Dissertation

Robuste Navigation autonomer mobiler Systeme

Jens–Steffen Gutmann

16. Oktober 2000

Betreuer:

Prof. Dr. Bernhard Nebel

Zusammenfassung

Die Navigation eines autonomen mobilen Roboters läßt sich grob durch die drei Fragen „Wo bin ich?“, „Wohin gehe ich?“ und „Wie gelange ich dorthin?“ zusammenfassen. In dieser Dissertation werden allen drei Fragen intensiv untersucht, Lösungen erarbeitet und Vergleiche verschiedener Methoden präsentiert. Hauptsächliches Augenmerk wird dabei auf Verfahren gelegt, welche Daten eines 2d-Laserscanners verarbeiten. Es wird gezeigt, daß Scan-Matching eine robuste Methode für die Selbstlokalisierung eines mobilen Roboters in einer dynamischen Umgebung und über einen längeren Zeitraum ist und es wird ein Vergleich mit Markov-Lokalisierung vorgestellt. Weiterhin wird gezeigt, wie mittels Scan-Matching hochgenaue metrische Karten in großen, zyklischen Umgebungen erstellt werden können und wie durch Auswerten der Sichtbarkeit zwischen Scans eine topologische Karte für die Pfadplanung berechnet werden kann. Alle Verfahren werden in Innenraumumgebungen und unter realen Bedingungen erprobt. Weiterhin wird Roboterfußball als eine Anwendung der drei Navigationsprobleme vorgestellt und gezeigt, wie auf Basis einer schnellen, robusten und genauen Selbstlokalisierung Weltmodellierung, Pfadplanung und Kooperation effektiv realisiert werden können. Dies führt zu einer sehr erfolgreichen Roboter-Fußballmannschaft, die sowohl den Titel des Deutschen Meisters als auch den Weltmeistertitel errang.

Danksagung

Für die hervorragende Unterstützung beim Anfertigen dieser Dissertation möchte ich mich bei allen Beteiligten bedanken. Mein Dank gilt Bruno Welsch für die Rechnerbetreuung, Roswitha Hilden für die Administration und Jochen Renz für eine vergnügsame Zeit im gemeinsamen Büro. Dank gilt auch dem Graduiertenkolleg „Menschliche und Maschinelle Intelligenz“ für die Förderung dieser Arbeit.

Weiterer Dank gilt allen Mitwirkenden des *CS-Freiburg*-Projekts, insbesondere Wolfgang Hatzack, der die Projektleitung für 1999 und viele organisatorische Tätigkeiten übernahm, Markus Dietl für die hervorragende Sensordatenfusion, Frank Rittinger für die Benutzeroberfläche, Burkhard Dümler für die Erweiterungen der Scannerschnittstelle, Augustinus Topor für die mühevolle Ballerkennung und den sehr leistungsfähigen Wegeplaner, Maximiliam Thiel für die Verbesserung der Ballerkennung, Stefan Rahmann für dessen wissenschaftliche Betreuung bei der Bildverarbeitung und Kornel Marko für die Entwicklung des RoboCup-Simulators. Weiterer Dank gilt Christian Reetz für die Modernisierung der Aktionsauswahl, für die endlosen Stunden gemeinsamen Testens, sowie für seine niemals endende Motivation, auch noch 48 Stunden vor Abflug zur WM'99 mit der Realisierung des Paßspiels zu beginnen. Immanuel Herrmann gilt weiterer Dank für den genialen Entwurf der Ballschußvorrichtung sowie weiterer Hardware-Arbeiten und Gimmicks. Immanuel ist nicht nur handwerklich ein Genie sondern auch auf theoretischem Gebiet, was seine Erfolge bei den ACM-Wettbewerben belegen. Ganz besonderer Dank gilt Thilo Weigel, der die Hauptarbeit bei der Entwicklung des RoboCup'98-Teams übernahm und der mehr als nur ein hervorragender Diplomand war.

Weiterer Dank gilt Sebastian Thrun, Dieter Fox und Wolfram Burgard für die fruchtbaren Gespräche und die gute Zusammenarbeit. Wer einmal mit diesem Dreierteam zusammengearbeitet hat, weiß wie Wissenschaft betrieben werden kann. Wolfram Burgard verdient weiteren Dank für seine wertvollen Tips, die Qualität dieser Arbeit zu steigern, und für seinen darüber hinaus reichenden Einsatz. Kurt Konolige gilt ganz besonderer Dank. Erst durch die Besuche an seinem Institut und durch seine Ideen sind die richtig wichtigen Resultate dieser Arbeit entstanden.

Zuletzt bedanke ich mich bei Bernhard Nebel, welcher es mir ermöglicht hat, diese Arbeit überhaupt anzufertigen und ein hervorragender Betreuer war. Es ist ein Vergnügen in seinem Lehrstuhl zu arbeiten und mitzuwirken. Ich bedanke mich für diese schöne Zeit.

Für meine Eltern
und für Freunde.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Navigation mobiler Roboter ist eine Domäne, die seit vielen Jahren von vielen Forschungsgruppen untersucht wird. Es gibt bereits mehrere Robotersysteme, die hervorragend in ihrer Umgebung navigieren können, so z.B. Roboter, die in einem Museum als Museumsführer arbeiten, oder die Fußball auf einem mit Banden abgesicherten Spielfeld spielen.

Man könnte also vermuten, das Problem der Navigation für mobile Roboter sei gelöst. Und tatsächlich sind viele theoretische Grundlagen bereits gelegt. In der Praxis sind die meisten theoretischen Ergebnisse jedoch nicht direkt umsetzbar und müssen durch Näherungen ersetzt werden. Ein Grund hierfür ist, daß die Sensoren eines Roboters ungenaue oder unscharfe Information liefern, z.B. hat ein Entfernungssensor immer eine Meßungenauigkeit und gemessenen Abstände sind mit Fehlern behaftet.

In der Praxis gibt es daher oft Lösungen, die auf spezielle Gegebenheiten zugeschnitten sind. Lösungen, die in einer Umgebung gut arbeiten, müssen nicht zwangsläufig genauso gut in einer anderen Umgebung funktionieren. Die Fußballroboter des CS-Freiburg-Teams beispielsweise finden sich sehr gut in einem durch Banden abgesicherten Spielfeld zurecht. Stellt man sie jedoch außerhalb des Spielfeldes auf, so verlieren sie die Orientierung und können keine sinnvollen Tätigkeiten mehr ausführen.

Leonard und Durrant-Whyte [1991] fassen das allgemeine Problem der Roboternavigation durch die drei Fragen „Wo bin ich?“, „Wohin gehe ich?“ und „Wie gelange ich dorthin?“ zusammen. Anders formuliert müssen für ein vollständiges Navigationssystem die Probleme *Selbstlokalisierung*, *Kartenerstellung* und *Pfadplanung* gelöst werden. Bei der Selbstlokalisierung geht es darum, herauszufinden, wo sich der Roboter in seiner Umgebung befindet. Hierfür können die Messungen der auf dem Roboter befindlichen Sensoren eingesetzt werden. Bei der Kartenerstellung möchte man mit den Sensoren des Roboters eine Karte erstellen, die der Roboter z.B. für Selbstlokalisierung und Pfadplanung verwenden kann. Schließlich wird bei der Pfadplanung versucht, kollisionsfreie Wege von einer Start- zu einer Zielposition zu finden, die der Roboter in der realen Welt abfahren soll.

In dieser Arbeit werden alle drei Gebiete untersucht, sowie Lösungen und Ergebnisse zu jedem der Teilprobleme präsentiert. Insbesondere werden für die Gebiete Selbstlokalisierung und Kartenerstellung Resultate vorgestellt, die an der Spitze weltweiter Forschung liegen. Neben den drei Gebieten der Navigation mobiler Roboter wird weiterhin in einer Anwendung gezeigt, wie die verschiedenen Methoden zusammengesetzt werden können, um ein vollständiges Navigationssystem zu erhalten. Als Anwendung wurde Roboterfußball gewählt, da hier unter Echtzeitbedingungen agiert werden muß und weitere Probleme wie kooperatives Verhalten eine Rolle spielen.

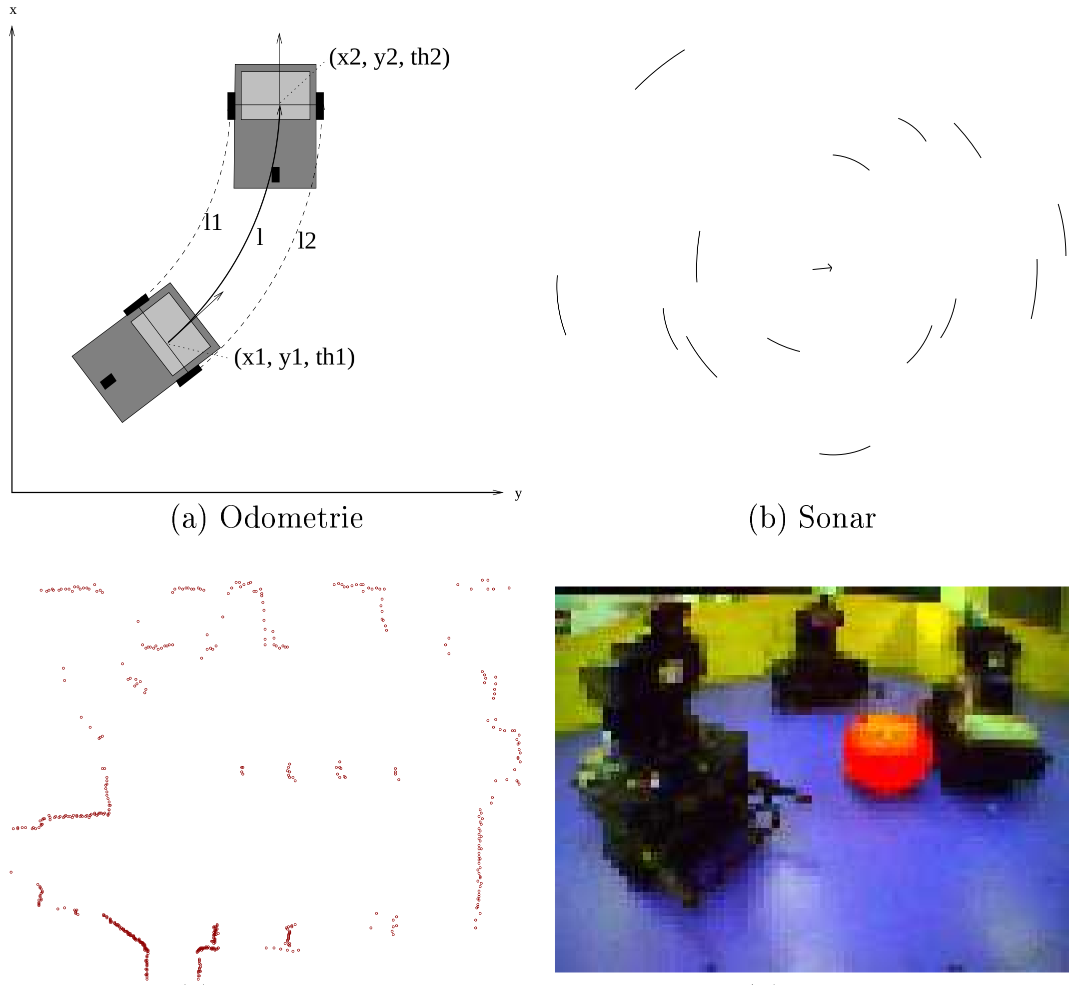
Die in dieser Arbeit vorgestellten Verfahren sind für Innenraumumgebungen, in denen sich dynamische Hindernisse wie Menschen oder andere Roboter bewegen, zugeschnitten. Alle Methoden werden mit realen Daten eines Roboters getestet, wobei zum Teil aufgezeichnete Daten verwendet werden, da so reproduzierbare Ergebnisse erzeugt werden können und statistische Resultate durch nachträgliches Verrauschen der Daten gewonnen werden können.

Diese Einleitung ist eine Gesamtübersicht dieser Arbeit. Im nächsten Abschnitt werden verschiedene Sensoren vorgestellt, die sehr gut für die Navigation eines mobilen Roboters geeignet sind. Danach werden die Daten eines Sensors, dem Laserscanner, näher untersucht und Methoden für die Verarbeitung und das Überdecken von Laserscans vorgestellt. Im darauffolgenden Abschnitt wird die Scanüberdeckung dazu benutzt, die Position eines Roboters zu bestimmen. Weitere Verfahren zur Selbstlokalisierung werden vorgestellt und ein Vergleich der Verfahren vorgeschlagen. Danach werden verschiedene Verfahren zur Erstellung von Umgebungskarten skizziert und in einem weiteren Abschnitt wird auf die Pfadplanung in einer dynamischen Büroumgebung eingegangen. Schließlich wird die Realisierung einer Roboter-Fußballmannschaft vorgestellt.

1.1 Sensorik

Damit ein mobiler Roboter autonom in einer Umgebung navigieren kann, muß er mit verschiedenen Sensoren, die seine Umwelt wahrnehmen, ausgestattet sein. Vier im Bereich Roboternavigation sehr beliebte Sensoren hierfür sind Odometrie, Sonar, Laserscanner und Videokameras, die im folgenden genauer beschrieben werden. Die in dieser Arbeit vorgestellten Verfahren verwenden nur Daten, die von einem oder mehreren dieser vier Sensoren aufgenommen wurden. Abbildung 1.1 zeigt Beispieldaten der verschiedenen Sensoren.

Odometrie: Die Odometrie besteht aus einem oder mehreren Rädern, die mit Sensoren ausgestattet sind, welche die Drehbewegung der Räder messen. Auf diese Weise wird bestimmt, wie lang der zurückgelegte Weg des jeweiligen Rades ist. Natürlich kann so noch nicht die Richtung bestimmt werden, in der sich das Fahrzeug bewegt hat. Bei einer Odometrie mit nur einem



(a) Odometrie (b) Sonar

(c) Laserscan (d) Vision

Abbildung 1.1: Verschiedene Sensoren, die für die Navigation eines mobilen Roboters verwendet werden können.

Rad kann man die Richtung z.B. durch einen zusätzlichen Sensor bestimmen, der den Winkel des Rades zur Längsachse des Fahrzeugs mißt (das Rad muß hierbei frei drehbar sein).

Falls die Odometrie wie beim *Pioneer-I*-Roboter aus zwei Rädern besteht, so kann die Richtungsänderung des Fahrzeugs aus der Differenz der beiden gemessenen Wegstrecken ermittelt werden (siehe Abbildung 1.1a). In Abschnitt 4.1 wird die zugrundeliegende Mathematik für diese Odometrie genauer untersucht.

Die Odometrie ist auf kurzen Strecken sehr genau, sie hat aber den gravierenden Nachteil, daß die Messung des zurückgelegten Weges mit zunehmender Strecke immer ungenauer und somit die Positionsschätzung des

Fahrzeuges zunehmend schlechter wird.

Zur Verbesserung der Positionsbestimmung wird in vielen Robotersystemen zusätzlich ein Kreiselkompaß eingesetzt. Mit diesem Sensor kann recht genau die Orientierung des Fahrzeugs bestimmt werden (meist wesentlich genauer als mit einer Odometrie). Ein Nachteil dieses Sensors ist, daß die Orientierung mit der Zeit ungenau wird (Drift), selbst wenn sich das Fahrzeug im Stillstand befindet, und daß große Beschleunigungen, z.B. beim Überfahren einer Schwelle, zu Störungen führen können.

Sonar: Sonar- oder Ultraschallsensoren werden zur Entfernungsbestimmung benutzt. Ein Sensor sendet dabei ein Signal im Ultraschallbereich aus und wartet auf ein rückkehrendes Echo. Das Signal breitet sich kegelförmig in die Abstrahlrichtung des Sensors aus, wobei der Öffnungswinkel gebräuchlicher Sensoren im Bereich von $5°$ bis $30°$ liegt. Das Signal wird an der Oberfläche eines in diesem Bereich befindlichen Objekts reflektiert und kehrt zum Sensor zurück. Der Ultraschallsensor mißt nun die Zeit zwischen Aussenden und Empfangen des Signals und bestimmt so die Entfernung. Abbildung 1.1b zeigt einen Scan, der von einem Ring von 24 Sonarsensoren aufgenommen wurde.

Der Vorteil von Ultraschallsensoren ist, daß sie sehr preiswert sind und wegen ihrer handlichen Größe nahezu überall montiert werden können. Ein Problem bei Ultraschallmessungen ist, daß man wegen des großen Öffnungswinkels nicht die genaue Richtung kennt, in die eine Messung durchgeführt wurde, da das erste Objekt in dem Öffnungsbereich die Entfernung bestimmt (vgl. Abbildung 1.1b). Oft wird versucht, dieses Problem durch probabilistische Methoden in den Griff zu bekommen.

Ein weiteres und sehr unangenehmes Problem ist, daß das Ultraschallsignal an glatten Oberflächen abgelenkt wird und kein Echo an den Sensor zurückgeht (*specular reflection*). Das Signal wird erst an einem weiteren Objekt reflektiert. Der Sensor mißt also „um die Ecke“ und erkennt das vor ihm liegende Objekt nicht.

Ein ähnliches Problem tritt beim gleichzeitigen Einsatz mehrerer Sensoren auf. Hier kann es passieren, daß ein Sensor die ausgestrahlten Signale eines anderen Sensors empfängt und als seine eigenen interpretiert (*cross-talk*), was natürlich zu einer falschen Entfernungsmessung führt.

Laserscanner: Laserscanner haben ein ähnliches Meßprinzip wie Sonarsensoren. Auch hier wird ein Signal in eine Richtung ausgesandt und auf die Rückkehr dieses Signals gewartet. Die ausgesandten Signale bewegen sich jedoch mit Lichtgeschwindigkeit, was zu einer deutlich kürzeren Signallaufzeit führt, aber auch schnellere und genauere Zeitmessungen erfordert. Das Signal wird wieder kegelförmig ausgestrahlt, der Kegel hat nun aber einen sehr kleinen

Öffnungswinkel, so daß man näherungsweise von einem Strahl ausgehen kann. Durch Rotation des Sende- und Empfangkopfes oder Ablenken des Lasersignals an einem Spiegel kann man Entfernungsmessungen in mehrere Richtungen durchführen. Auf diese Weise können zwei- und dreidimensionale Abstandsprofile aufgenommen werden. Laserscanner können sehr genaue Abstandsmessungen (kleiner $1cm$ Genauigkeit) liefern. Abbildung 1.1c zeigt einen Beispielscan eines horizontal montierten 360°-Scanners.

Probleme treten auf, wenn der Laserstrahl auf stark absorbierende Oberflächen (z.B. schwarze Oberflächen), auf Glas oder auf Spiegel trifft. Absorbierende Oberflächen „verschlucken" das Signal und an den Scanner gelangt kein Echo mehr zurück. Glas reflektiert den Laserstrahl in vielen Situationen nicht und der Scanner „schaut" auf Objekte hinter dem Glas. Spiegel haben das gleiche Verhalten wie glatte Oberflächen bei Ultraschallmessungen: das ausgesandte Signal wird an ihnen abgelenkt und der Sensor „mißt um die Ecke".

Es muß jedoch festgehalten werden, daß die Zahl der Fälle, in denen der Einsatz eines Laserscanners in einer Büroumgebung ungünstig ist, wesentlich kleiner ist, als die für Ultraschallsensoren kritischen Fälle. Während Glas noch recht häufig auftritt, sind Spiegel und stark absorbierende Oberflächen relativ selten. Dagegen treten Ablenkungen von Ultraschallsignalen an glatten Oberflächen (z.B. Wänden) sehr häufig auf.

Videokamera: Mit Videokameras können Farbbilder der Umgebung aufgenommen und digitalisiert werden. Die Ausgabe ist ein Pixelbild, das aus Farbwerten besteht. Abbildung 1.1d zeigt das Bild einer Videokamera, das von einem Fußballroboter des CS-Freiburg-Teams während eines Testspiels aufgenommen wurde.

Videosysteme können keine direkte Entfernungswerte liefern, die Pixelbilder sind von der Beleuchtung abhängig, die Datenmenge ist nicht unerheblich und die Extraktion von Merkmalen ist schwierig und zeitaufwendig. Dafür können mit den Daten einer Videokamera in einem Raum befindliche Objekte z.B. anhand von Form oder Farbe erkannt werden.

Die Verarbeitung von Videodaten ist eine große Herausforderung in den Navigationsverfahren mobiler Roboter, welche menschliches Verhalten als Vorbild verwenden. Die Erkennung und Klassifikation von Objekten, z.B. für die Navigation mit natürlichen Landmarken, ist seit Jahren ein großes Forschungsgebiet.

Die Wahl der Sensoren für die Navigation eines mobilen Roboters hängt von dem angestrebten Navigationsverfahren ab. Beispielsweise ist es leichter, eine in einem Raum speziell angebrachte Markierung (künstliche Landmarke) in einem Videobild zu erkennen, als diese in einem Abstandsprofil von einem Laserscanner zu detektieren. Dagegen ist es wesentlich einfacher, einen Positionsunterschied durch den Vergleich zweier Laserscans zu bestimmen, als durch den Vergleich zweier Videobilder.

Für viele Navigationsbereiche scheinen jedoch Laserscanner der ideale Kompromiß zwischen Ultraschall und Video zu sein. Ultraschall ist wegen der genannten Gründe ungenau und unzuverlässig. Video liefert eine zu große Datenmenge, die nur auf sehr schnellen Rechnern verarbeitet werden kann und ist anfällig gegen verschiedene Beleuchtungsverhältnisse.

Aus diesen Gründen wird in dieser Arbeit der Schwerpunkt auf Verfahren gelegt, die Daten eines Laserscanners verarbeiten. Dabei beschränken sich die Daten des Scanners auf zweidimensionale Abstandsprofile, die in der Horizontalen aufgenommen wurden.

1.2 Scan-Matching

Ein Grundbaustein, der für die Lösung vieler Navigationsaufgaben verwendet werden kann, ist *Scan-Matching*. Die Aufgabe besteht darin, einen Scan so zu verdrehen und zu verschieben, daß eine Überdeckung mit einem Referenzmodell zustande kommt.

Scan-Matching wird in Kapitel 3 genauer untersucht. Dort werden zunächst Vorverarbeitungsschritte, Filter und die Projektion von Scans vorgestellt, um Scans für eine schnelle, robuste und genaue Überdeckung vorzubereiten. Unter anderem wird beschrieben, wie aus einem Laserscan Merkmale wie Liniensegmente oder Ecken extrahiert werden können, wie die Anzahl der Meßpunkte durch Filter verkleinert und die Qualität der Messungen verbessert werden kann, und wie „unsichtbare“ Scanpunkte (Scanpunkte, die von einer anderen Aufnahmeposition nicht sichtbar sind) erkannt und entfernt werden können.

Danach werden verschiedene Methoden für die Überdeckung von Scans vorgestellt. Eine Methode extrahiert Liniensegmente aus einem Scan und ordnet Scanpunkte des anderen Scans diesen Liniensegmenten zu. Durch Minimieren einer Fehlersumme wird so die Verdrehung und Verschiebung der beiden Scans bestimmt. Ein anderes Verfahren ordnet direkt Scanpunkte des einen Scans den Scanpunkten des anderen Scans zu und berechnet so Rotation und Translation. Abbildung 1.2 zeigt ein Beispielpaar von Scans vor und nach der Überdeckung.

Das Kapitel schließt mit einer Diskussion der verschiedenen Verfahren ab. Dabei wird eine Kombination zweier Methoden präsentiert, welche die Vorteile der beiden Verfahren zusammenführt. Weiterhin wird ein Überblick von anderen in der Literatur entwickelten Verfahren gegeben.

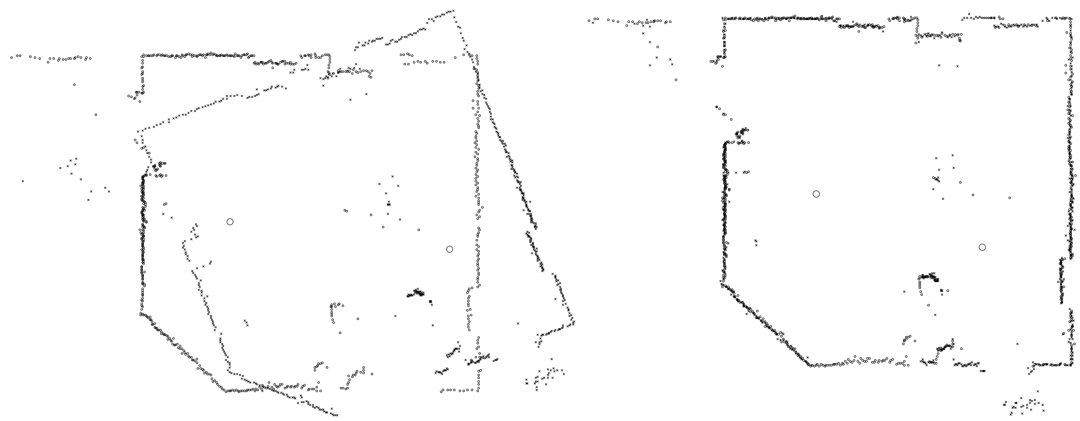


Abbildung 1.2: Scanpaar vor und nach Scanüberdeckung.

1.3 Selbstlokalisierung

Die Selbstlokalisierung eines mobilen Roboters in einer bekannten aber dynamischen Umgebung ist Gegenstand des 4. Kapitels. Scan-Matching eignet sich sehr gut für die Positionsbestimmung eines mobilen Roboters. Ein aufgenommener Scan wird mit einem Referenzmodell, z.B. einer Karte von Liniensegmenten oder mit einem zuvor aufgenommenen Referenzscan abgeglichen. Aus der hierzu notwendigen Verdrehung und Verschiebung des Scans läßt sich dann leicht eine Korrektur für die Roboterposition berechnen. Hierbei wird angenommen, daß der Roboter z.B. durch Auswerten von Odometrieinformation bereits ungefähr weiß, an welcher Position er sich befindet. Abbildung 1.3 zeigt eine Beispielfahrt eines Roboters mit erfolgreicher Lokalisierung durch Einsatz von Scan-Matching.

Ohne jegliches Vorwissen über die ungefähre Roboterposition kann Scan-Matching jedoch keine eindeutige Position bestimmen. Ein anderes Verfahren, das in der Lage ist, eine solche *globale* Selbstlokalisierung durchzuführen, ist *Markov-Lokalisierung*. Hier wird explizit eine Positionsaufenthaltswahrscheinlichkeit verwendet, die mit jeder Messung der Robotersensoren fortgeschrieben wird. Je nach Umgebung kann das Verfahren schon nach wenigen gefahrenen Metern die globale Position des Roboters bestimmen.

Es gibt eine Vielzahl anderer Verfahren für die Lokalisierung mobiler Roboter. Jedes dieser Verfahren wurde implementiert und erfolgreich in einer Laborumgebung erprobt. Doch welches Verfahren ist eigentlich das beste? Und wie stark hängt das Verfahren von Umgebung oder verwendeter Sensoren ab?

Um dieser Fragestellung nachzugehen, werden die beiden Verfahren Scan-Matching und Markov-Lokalisierung experimentell miteinander verglichen. Dabei stellt sich heraus, daß Scan-Matching genauere Resultate liefert, Markov-Lokalisierung unter großen Störeinflüssen dafür aber wesentlich robuster als Scan-Matching arbeitet.

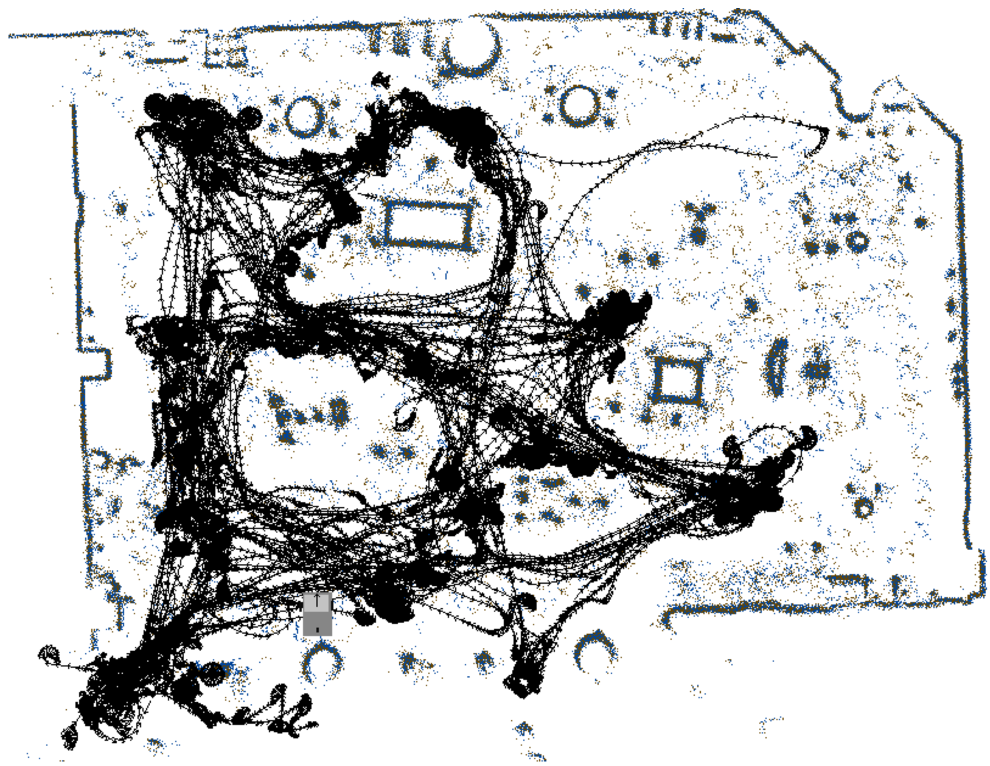


Abbildung 1.3: Beispielfahrt mit erfolgreicher Lokalisierung des Roboters *RHINO* im Deutschen Museum Bonn.

1.4 Kartenerstellung

Damit sich ein Roboter lokalisieren kann, benötigt er eine Karte seiner Umgebung. Eine solche Karte kann natürlich aus einem CAD-Modell erstellt oder von Hand vermessen werden. Jedoch sind CAD-Modelle meist nicht verfügbar, unvollständig oder gar falsch, und eine Vermessung der Umgebung von Hand ist sehr aufwendig. Wünschenswert wäre es, den Roboter selbst mittels seiner Sensoren die Umgebung vermessen zu lassen.

Dieses Aufgabe der Kartenerstellung wird in Kapitel 5 ausführlich besprochen. Hierbei stellt sich das Problem, daß für das Eintragen neuer Sensorinformation in eine bestehende Karte die Position des Roboters bekannt sein muß. Um aber die Position des Roboters zu kennen, wird wiederum eine Karte benötigt. Um dieses Problem zu lösen, wird für gewöhnlich ein rekursiver Weg gewählt, d.h. es wird zunächst die Roboterposition aufgrund der bisher erstellten Karte bestimmt und anschließend neue Sensorinformation in das Modell eingetragen.

Diese inkrementelle Kartenerstellung führt jedoch schnell zu Problemen, wenn die zu kartierende Umgebung Zyklen enthält, d.h. der Roboter über einen langen Weg wieder zu einer bereits besuchten Position zurückkehrt. An dieser Stelle kann der Roboter einen aufgenommenen Sensordatensatz bezüglich des zuletzt erstellen Kartenbereiches oder bezüglich des älteren an dieser Stelle erstellten Bereichs abgleichen. Viele einfache Verfahren erlauben zwar den Abgleich bezüglich mehrerer Referenzbereiche, doch ist es unklar, wie die neue Sensorinformation in die bestehende Karte eingetragen und die Kartenteile selbst aktualisiert werden sollen.

Um diesen Problem zu begegnen, wurde von Lu und Milios [1997a] ein Verfahren entwickelt, das alle Sensordaten entlang eines Zyklus in Betracht zieht und das alle ursprünglich durch Odometrie ermittelten Roboterpositionen entlang des Pfades so korrigiert, daß die Kartenteile zusammenpassen. Die Methode verwendet als Basisbaustein Scan-Matching, um einzelne Sensorscans zusammenzupassen. Die hieraus entstehenden Karten sind sehr genau.

Ein Problem des Ansatzes von Lu und Milios ist, daß die rohen Sensordaten bereits grob zueinander passen müssen, da Scan-Matching immer eine initiale Positionsschätzung benötigt. Ist die initiale Positionsschätzung jedoch sehr ungenau, so können Scans nicht mehr miteinander überdeckt werden, oder es werden nicht-zusammenpassende Scans miteinander überdeckt. Dies kann zu inkonsistenten oder falschen Karten führen. Mit anderen Worten müssen die rohen Sensordaten bereits *topologisch korrekt* sein.

Ein anderes Verfahren wurde von Thrun *et al.* [1998b] entwickelt. Dieses Verfahren benutzt eine Erwartungs- und Maximierungsstrategie, welche aus einem aufgenommenen Satz von Sensormessungen abwechselnd Roboterpositionen und Umgebungskarte berechnet, bis die entstandene Karte sich nicht mehr verändert. Das Verfahren ist in der Lage, topologisch korrekte Karten zu erstellen, jedoch ist es nicht so genau wie die Methode von Lu und Milios.

Durch Kombination der beiden Methoden erhält man ein Verfahren, das rohe Sensormessungen in eine hochgenaue Karte transformieren kann. Hierzu wird zunächst die Methode von Thrun *et al.* verwendet, um eine topologisch korrekte Karte zu erhalten, und anschließend die Methode von Lu und Milios, um die Genauigkeit zu verbessern. Abbildung 1.4 zeigt einen rohen Datensatz von Laserscannerdaten und die durch Anwendung des kombinierten Verfahrens entstandene Karte.

Schließlich wird in diesem Kapitel ein neues Verfahren vorgestellt, das in der Lage ist, inkrementell und in Echtzeit eine Karte aus einem Strom von Sensorwerten zu berechnen. Diese Methode benutzt Scan-Matching für die Erstellung kleiner Kartenstücke und die Feinkorrektur der verwalteten Karte, Kartenkorrelation für die Erkennung von topologischen Zusammenhängen, sowie das Verfahren von Lu und Milios für das Schließen von Zyklen.

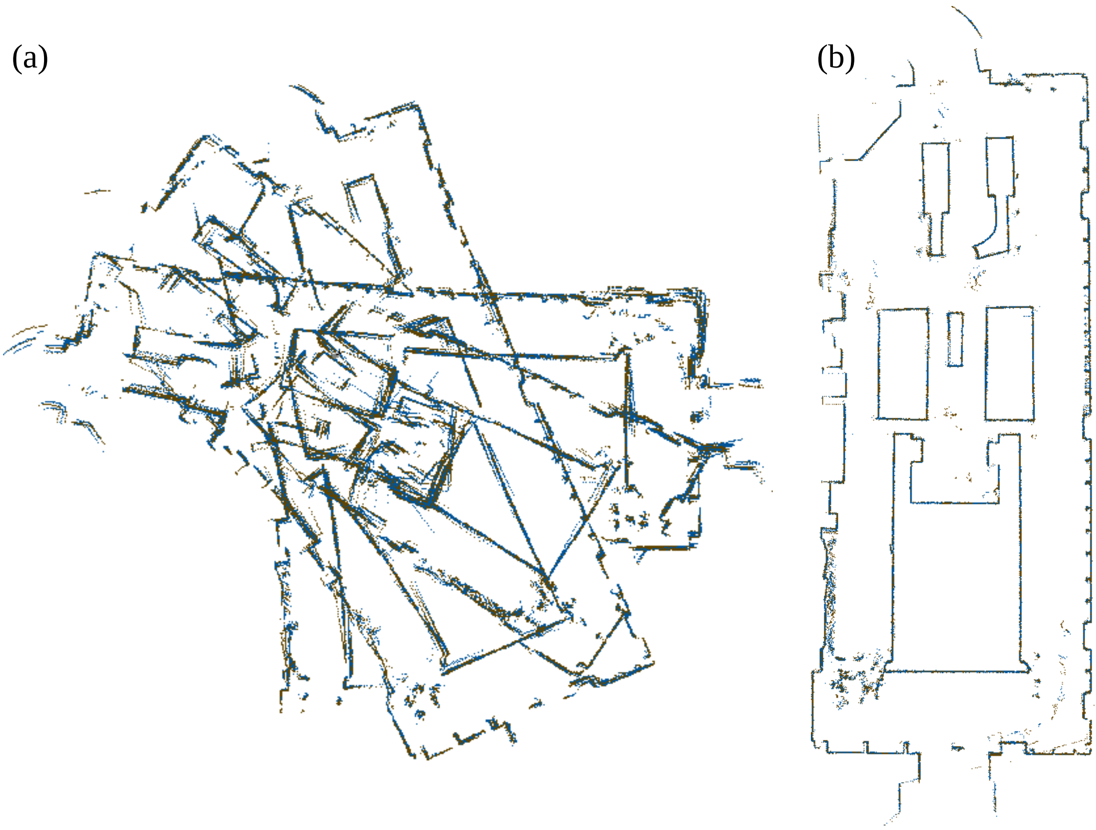


Abbildung 1.4: Beispiel für Kartenerstellung. (a) Rohdaten. (b) berechnete Karte.

1.5 Pfadplanung

In einem weiteren Kapitel wird auf die Pfadplanung, dem letzten der drei Navigationsprobleme, eingegangen. Zunächst wird in Anlehnung an die Kategorisierung von Latombe [1991] ein Überblick der verschiedenen Pfadplanungsansätze gegeben. Danach wird auf Verfahren, welche die Umgebung in ein Gitter mit Zellen gleicher Größe einteilen, näher eingegangen. Diese Verfahren sind in der Lage, mittels A^*-Suche oder Dynamischen Programmierens kollisionsfreie Pfade zu finden und haben den Vorteil, daß sie vollständig und korrekt sind, d.h. ein Lösungspfad wird auch tatsächlich gefunden, wenn einer vorhanden ist und ein gefundener Pfad führt den Roboter auch tatsächlich kollisionsfrei zum Ziel. Weiterhin erlauben diese Methoden *optimale*, d.h. kürzeste Pfade zu finden und sie können auf dynamische Veränderungen der Umwelt, z.B. umherlaufende Menschen, reagieren. Weiterhin stellen lokale Minimas, die z.B. durch eine Sackgasse hervorgerufen werden, im Gegensatz zu vielen verhaltensbasierten Methoden kein Problem für diese Methoden dar.

Das große Problem der gitterbasierten Pfadplanungsverfahren ist jedoch die Größe des Zustandraumes. Der Zustandsraum hängt direkt von der Größe der Einsatzumgebung und der verwendeten Zellgröße ab. Aus diesem Grund sollte ein gitterbasierter Wegeplaner nur in einer relativ kleinen Umgebung verwendet werden. Eine Möglichkeit, in größeren Umgebungen Wege zu planen, ist, ein zweistufiges Verfahren zu verwenden, das aus einem globalen Planer für einen Grobentwurf mit Zwischenzielen und einem gitterbasierten, lokalen Planer für die Planung zu den Zwischenzielen besteht.

Eine solche globale Wegeplanungsmethode ist weiterer Gegenstand dieses Kapitels. Hierfür wird aus einer Karte von Scans eine Wegekarte erzeugt. Die Knoten der Wegekarte sind die Aufnahmepositionen von Scans und die Verbindungslinien zwischen Knoten werden durch die Bestimmung einer sogenannten *Sichtbarkeit* zwischen den beiden beteiligten Scans bestimmt. Abbildung 1.5 zeigt die Büroumgebung der Abteilung der Künstlichen Intelligenz der Universität Freiburg zusammen mit dem hieraus berechneten Wegegraph.

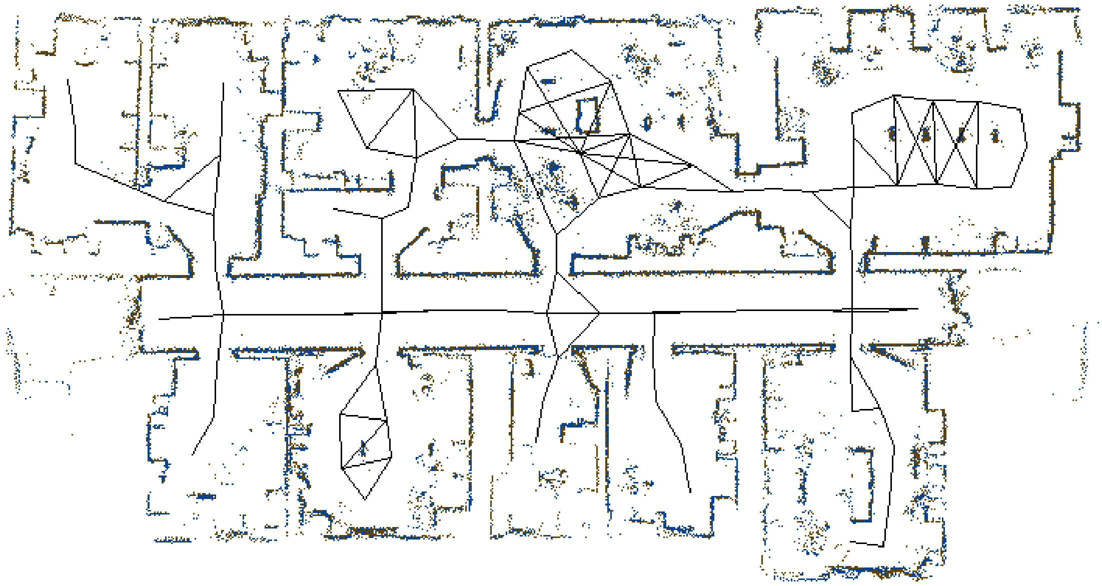


Abbildung 1.5: Wegegraph in der KI-Abteilung der Universität Freiburg.

Weiterhin wird auf Probleme des vorgestellten Ansatzes eingegangen und eine weitere Methode vorgeschlagen, die versucht, eine „natürliche“ Aufteilung der Umgebung in Regionen zu berechnen.

1.6 RoboCup

Roboterfußball. Dies ist das letzte große Kapitel dieser Arbeit. Hier wird gezeigt, wie die einzelnen Navigationskomponenten zusammengesetzt werden müssen, um ein robustes, effizientes und zugleich sehr erfolgreiches Team von Roboter-Fußballspielern zu erhalten. Da die RoboCup-Umgebung sehr einfach strukturiert ist, können einfache Algorithmen für die Selbstlokalisierung, Weltmodellierung und Pfadplanung verwendet werden.

Zunächst werden die verschiedenen Ligen, in denen Roboterfußball gespielt wird, und der Ansatz des CS-Freiburg-Teams für die Teilnahme in der Liga der mittelgroßen Roboter vorgestellt. Danach wird die allgemeine Architektur und die verwendete Hardware beschrieben. Abbildung 1.6 zeigt drei der fünf CS-Freiburg-Spieler.

Abbildung 1.6: Zwei Feldspieler und der Torwart des CS-Freiburg-Teams.

Auf die Realisierung einer schnellen, robusten und genauen Lokalisierung wurde ein besonderes Augenmerk gelegt. Da die Umgebung rechtwinklig ist und aus langen geraden Seitenwänden besteht, wurde ein neues Scan-Matching-Verfahren entwickelt, das aus einem Scan Liniensegmente extrahiert und diese mit einem apriori Linienmodell der RoboCup-Umgebung überdeckt. Dies führt zu einer sehr schnellen und genauen Lokalisierung, die in der Lage ist, die globale Roboterposition modulo der Symmetrie des Spielfeldes zu bestimmen.

Um zu erfahren wie gut das Verfahren im Vergleich zu anderen Scan-Matching-Methoden ist, wurde das Verfahren mit den Methoden aus Kapitel 4 experimentell verglichen. Hierbei stellte sich heraus, daß das Verfahren wesentlich schneller ist, eine höhere Robustheit (Ausfallsicherheit) bei stark verrauschten Odometriedaten hat, und dennoch eine vergleichbare Genauigkeit wie die anderen Methoden besitzt.

Auf Basis dieser Selbstlokalisierungskomponente lassen sich alle weiteren Systemkomponenten einfach und auf naheliegende Weise realisieren. Für die Spielererkennung werden Scanpunkte, die nicht zu den Spielfeldwänden gehören, gruppiert und der Schwerpunkt jeder Gruppe wird als der Mittelpunkt eines Spielers interpretiert. Für die Ballerkennung werden die Daten einer Videokamera farblich ausgewertet und die globale Ballposition auf dem Spielfeld bestimmt. Zusätzlich werden Geschwindigkeit und Richtungsbewegung aller Objekte durch Integration über die Zeit berechnet.

Die Steuerung der einzelnen Roboter geschieht lokal mit einem verhaltensbasierten Kontrollmodul. Situationen im Weltmodell werden auf auszuführende Aktionen abgebildet. Zusätzlich wird Kommunikation verwendet, um einzelne Aktionen, z.B. welcher Spieler zum Ball geht, abzusprechen.

Einzelne Aktionen können es erfordern, daß der Spieler zu einem bestimmten Zielpunkt fährt. Hierfür wird ein Pfadplanungsalgorithmus benötigt. Wegen der einfachen Struktur der RoboCup-Umgebung kann ein spezialisierter Algorithmus eingesetzt werden, z.B. die Methode des erweiterten Sichtbarkeitsgraphen aus [Latombe, 1991].

Zuletzt wird in diesem Kapitel über Erfahrungen bei Turnieren berichtet und es werden die erzielten Spielergebnisse bei der Teilnahme an der Weltmeisterschaft '98 in Paris (1. Platz), der Teilnahme an der Deutschen Meisterschaft '98 (1. Platz), der Teilnahme an der Weltmeisterschaft '99 in Stockholm (3. Platz) und der Teilnahme an der Deutschen Meisterschaft '99 (1. Platz) vorgestellt.

Diese Einleitung stellt eine Gesamtübersicht dieser Arbeit dar. Das folgende Kapitel beschäftigt sich mit Grundlagen, die in den weiteren Kapiteln benötigt werden. Danach folgen die Kapitel über Scan-Matching, Selbstlokalisierung, Kartenerstellung, Wegeplanung und RoboCup. Zuletzt wird die Arbeit zusammengefaßt und es wird ein Ausblick gegeben. Der Anhang informiert über die parallel zu dieser Arbeit entwickelte Software-Umgebung.

Kapitel 2

Grundlagen

Dieses Kapitel beschäftigt sich mit den mathematischen Grundlagen, die in dieser Arbeit verwendet werden. Zunächst werden Symbole und Notationen eingeführt. Danach werden stochastische Grundlagen wie Normalverteilung, Mahalanobis-Abstand und die Transformation von Dichtefunktionen definiert. Weiterhin wird der Kalman-Filter vorgestellt, welcher zur Fusion von Beobachtungen verschiedener Sensoren dient. Viele der in diesem Kapitel präsentierten Grundlagen finden sich auch in anderen Arbeiten [Smith *et al.*, 1990; Knoll, 1994; Gutmann, 1996].

2.1 Notationen

Die folgenden Notationen werden in dieser Arbeit verwendet.

Die Position von Objekten, z.B. des Roboters, wird durch einen Zeilenvektor $(x, y, \theta)^T$ angegeben, welcher Ort und Orientierung enthält. Für eine Matrix M wird durch M^T die transponierte Matrix angegeben. Ist M quadratisch und regulär, dann bezeichnet M^{-1} die Inverse Matrix zu M und $\det(M)$ ihre Determinante.

Für einen Winkel α definieren $R_2(\alpha)$ und $R_3(\alpha)$ die zugehörigen Rotationsmatritzen:

$$R_2(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{2.1}$$

$$R_3(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

2.2 Normalverteilung

In Bereich der Sensordatenverarbeitung wird oftmals angenommen, daß der Auftrittswahrscheinlichkeit von Meßwerten eine Normalverteilung zugrunde liegt.

Diese Annahme kommt daher, daß viele Störgrößen in die Messung eingehen. Jede dieser Störungen hat eine bestimmte, aber meist unbekannte Verteilung. Nach dem zentralen Grenzwertsatz der Stochastik gilt nun, daß die resultierende Messung als normalverteilt angenommen werden kann. Ein Vorteil der Normalverteilung ist, daß man sie vollständig durch die Angabe von Erwartungswert und Varianz definieren kann.

Die Dichtefunktion der Normalverteilung im skalaren Fall mit Erwartungswert μ_x und Varianz σ_x^2 ist wie folgt definiert:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_x}{\sigma_x}\right)^2} \tag{2.3}$$

Schreibweise: $x \sim N(\mu_x, \sigma_x^2)$

Im n-dimensionalen Fall wird dies zu:

$$p(x) = \frac{1}{\sqrt{(2\pi)^n \det(\Sigma_x)}} e^{-\frac{1}{2}(x-\mu_x)^T \Sigma_x^{-1}(x-\mu_x)} \tag{2.4}$$

Schreibweise: $x \sim N(\mu_x, \Sigma_x)$, wobei μ_x ein n-dimensionaler Erwartungswert und Σ_x eine $n \times n$-dimensionale Kovarianzmatrix ist.

2.3 Mahalanobis-Abstand

Für die Darstellung von normalverteilten Dichtefunktionen im mehrdimensionalen Fall sind Punktemengen mit gleicher Wahrscheinlichkeitsdichte interessant. Um diese zu bestimmen, setzt man die Dichtefunktion $p(x)$ konstant und bestimmt x.

$$\begin{aligned} \frac{1}{\sqrt{(2\pi)^n \det(\Sigma_x)}} e^{-\frac{1}{2}(x-\mu_x)^T \Sigma_x^{-1}(x-\mu_x)} &= \text{konstant} \\ (x-\mu_x)^T \Sigma_x^{-1}(x-\mu_x) &= \text{konstant} = r^2 \end{aligned} \tag{2.5}$$

Im zweidimensionalen Fall beschreibt diese Gleichung eine Höhenlinie des dreidimensionalen Graphen von $p(x)$. Diese Höhenlinie hat die Form einer Ellipse. Der Wert der Formel $(x-\mu_x)^T\Sigma_x^{-1}(x-\mu_x)$ heißt quadrierter Mahalanobis-Abstand zwischen dem Vektor x und dem Erwartungswert μ_x. Die Bedeutung dieses Maßes kann am einfachsten im skalaren Fall für $r = 1$ verdeutlicht werden. Die Gleichung reduziert sich dann zu $(x-\mu_x)^2\sigma_x^{-2} = 1$ oder $(x-\mu_x)^2 = \sigma_x^2$. Dies bedeutet, daß der Mahalanobis-Abstand von x zum Erwartungswert μ_x genau dann 1 wird, wenn x von μ_x um genau $1 \cdot \sigma_x$ abweicht.

2.4 Transformation von Dichtefunktionen

In der Sensordatenverarbeitung ist oftmals das Problem gegeben, Beobachtungen eines Sensors in ein anderes einheitliches Beobachtungssystem zu transformieren. Dabei liegen die gemessenen Daten in einer Normalverteilung mit bekanntem Erwartungswert und bekannter Kovarianzmatrix vor. Die Frage ist nun, welche Wahrscheinlichkeitsverteilung nach der Transformation vorliegt.

Zum Beispiel werden die Winkel- und Abstandsinformationen eines 2d-Laserscanners oftmals in kartesische Koordinaten umgerechnet. Für die Winkel- und Abstandsinformationen nimmt man eine Normalverteilung an, welche man z.B. durch eine Meßreihe bestimmen kann. Durch die Umrechnung in kartesische Koordinaten ergibt sich nun eine neue Verteilung, welche zu bestimmen ist.

Formal betrachtet ist eine m-dimensionale Größe u, mit $u \sim N(\mu_u, \Sigma_u)$ und eine Transformation $F : \mathbb{R}^m \to \mathbb{R}^n$ mit $F(u) = x$, $x \sim N(\mu_x, \Sigma_x)$ gegeben und der Erwartungswert μ_x sowie die Kovarianzmatrix Σ_x gesucht.

Zur Bestimmung der neuen Normalverteilung unterscheidet man zwischen den beiden Fällen, ob F linear oder nichtlinear ist.

2.4.1 Lineare Transformation

Im linearen Fall läßt sich F darstellen als $F(u) = Au + b$, wobei A eine konstante $n \times m$ Matrix und b ein n-dimensionaler Spaltenvektor ist. Für den Erwartungswert μ_x und die Kovarianzmatrix Σ_x gilt dann:

$$\begin{aligned}
\mu_x &= E(x) \\
&= E(Au + b) \\
&= AE(u) + b \\
&= A\mu_u + b \qquad (2.6)
\end{aligned}$$

$$\begin{aligned}
\Sigma_x &= E((x - E(x))(x - E(x))^T) \\
&= E((Au + b - AE(u) - b)(Au + b - AE(u) - b)^T) \\
&= E((A(u - E(u)))(A(u - E(u)))^T) \\
&= E((A(u - E(u)))((u - E(u))^T A^T)) \\
&= AE((u - E(u))(u - E(u))^T)A^T \\
&= A\Sigma_u A^T \qquad (2.7)
\end{aligned}$$

2.4.2 Nichtlineare Transformation

Falls F nicht linear ist, so nähert man es durch ein Taylorpolynom an einer geeigneten Stützstelle $\hat{u}$ an und läßt Terme höherer Ordnung weg:

$$F(u) \approx F(\hat{u}) + \nabla F(\hat{u})(u - \hat{u}) \qquad (2.8)$$

Hierbei ist $\nabla F(\hat{u}) = \frac{\partial F}{\partial u}(\hat{u})$ eine $n \times m$ Matrix mit den partiellen Ableitungen von F an der Stelle $\hat{u}$. Diese Matrix wird auch Jacobi-Matrix genannt.

F wird so an einer geeignete Stützstelle $\hat{u}$ linearisiert. Für die Stützstelle empfiehlt sich der Erwartungswert von u, da man nur an Punkten nahe dem Erwartungswert interessiert ist, also $\hat{u} = \mu_u$. Bringt man F nun in die Schreibweise für den linearen Fall, dann ergibt sich für die Matrix A und den Vektor b:

$$\begin{aligned} A &= \nabla F(\mu_u) && (2.9) \\ b &= F(\mu_u) - \nabla F(\mu_u)\mu_u && (2.10) \end{aligned}$$

Unter Zuhilfenahme der Ergebnisse des linearen Falls ergibt sich dann für den Erwartungswert μ_x und die Kovarianzmatrix Σ_x:

$$\begin{aligned} \mu_x &= A\mu_u + b \\ &= \nabla F(\mu_u)\mu_u + F(\mu_u) - \nabla F(\mu_u)\mu_u \\ &= F(\mu_u) && (2.11) \end{aligned}$$

$$\begin{aligned} \Sigma_x &= A\Sigma_u A^T \\ &= \nabla F(\mu_u)\Sigma_u \nabla F(\mu_u)^T && (2.12) \end{aligned}$$

2.4.3 Beispiel für eine Transformation

Betrachtet man den Meßvorgang eines 2d-Laserscanners, so stellt man fest, daß die Entfernungsmessungen und der Winkel, unter dem die Messungen stattfinden, mit einem Fehler behaftet sind. Für diese Fehler nimmt man nun an, daß sie normalverteilt und unabhängig voneinander sind. Sei d der gemessene Abstand und α der Winkel, unter dem die Messung stattfand, dann sind d und α normalverteilt mit $d \sim N(\mu_d, \sigma_d^2)$ und $\alpha \sim N(\mu_\alpha, \sigma_\alpha^2)$, wobei der Erwartungswert μ_d der tatsächlichen Entfernung und μ_α dem tatsächlichen Winkel entspricht.

Eine Messung $(d, \alpha)^T$ wird nun in kartesische Koordinaten umgerechnet, indem die Transformation F angewendet wird (siehe Abb. 2.1):

$$F(\begin{pmatrix} d \\ \alpha \end{pmatrix}) = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} d\cos\alpha \\ d\sin\alpha \end{pmatrix} \tag{2.13}$$

Durch die Transformation ergibt sich für die kartesischen Koordinaten eine neue Verteilung

$$\begin{pmatrix} x \\ y \end{pmatrix} \sim N(\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \Sigma_{xy}) \tag{2.14}$$

mit

$$\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} = F(\begin{pmatrix} d \\ \alpha \end{pmatrix}) = \begin{pmatrix} d\cos\alpha \\ d\sin\alpha \end{pmatrix} \tag{2.15}$$

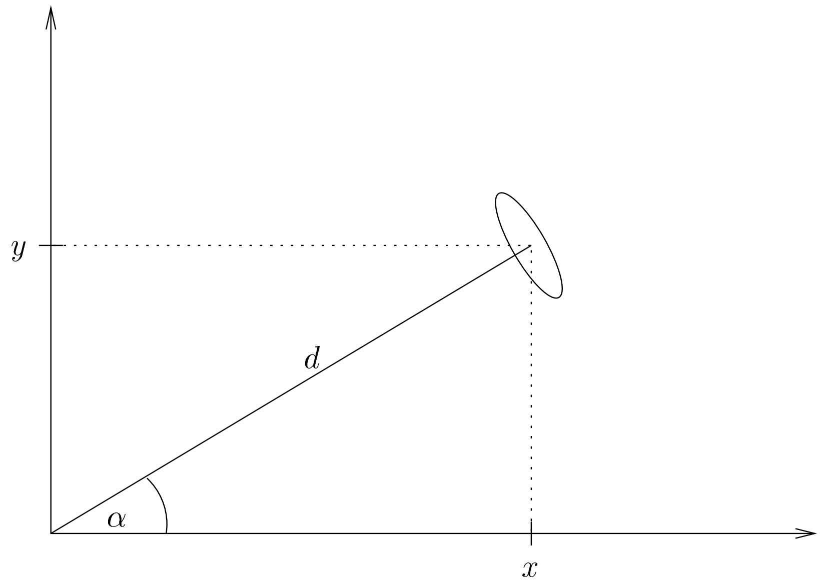


Abbildung 2.1: Transformation einer Messung in kartesische Koordinaten.

$$\Sigma_{xy} = \nabla F_{d\alpha} \Sigma_{d\alpha} \nabla F_{d\alpha}{}^T \tag{2.16}$$

$$\nabla F_{d\alpha} = \begin{pmatrix} \cos\alpha & -d\sin\alpha \\ \sin\alpha & d\cos\alpha \end{pmatrix} \tag{2.17}$$

$$\Sigma_{d\alpha} = \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\alpha^2 \end{pmatrix} \tag{2.18}$$

Zahlenbeispiel

Setzt man z.B. für $d = 3000mm$, $\alpha = \frac{\pi}{6}$ $(30°)$, $\sigma_d = 100mm$ und $\sigma_\alpha = 0.1$ $(5.730°)$ ein, so ergibt sich für den Erwartungswert und die Kovarianzmatrix (Einheiten in mm, bzw. Bogenmaß):

$$\begin{aligned}
\begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix} &= \begin{pmatrix} d\cos\alpha \\ d\sin\alpha \end{pmatrix} = \begin{pmatrix} 2598 \\ 1500 \end{pmatrix} \\
\nabla F_{d\alpha} &= \begin{pmatrix} \cos\alpha & -d\sin\alpha \\ \sin\alpha & d\cos\alpha \end{pmatrix} = \begin{pmatrix} 0.866 & -1500 \\ 0.500 & 2598 \end{pmatrix} \\
\Sigma_{d\alpha} &= \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\alpha^2 \end{pmatrix} = \begin{pmatrix} 10000 & 0 \\ 0 & 0.01 \end{pmatrix} \\
\Sigma_{xy} &= \nabla F_{d\alpha} \Sigma_{d\alpha} \nabla F_{d\alpha}{}^T = \begin{pmatrix} 30000 & -34640 \\ -34640 & 70000 \end{pmatrix}
\end{aligned}$$

Betrachtet man alle Punkte in der Ebene, deren Mahalanobis-Abstand zum Erwartungswert 1 beträgt, so ergibt sich die in Abbildung 2.1 dargestellte Ellipse. Die Punkte innerhalb der Ellipse besitzen einen Mahalanobis-Abstand kleiner 1.

Zur Bestimmung der Ellipse berechnet man die Inverse von Σ_{xy} und bringt diese auf ihre Hauptachsen. Die Hauptachsen betragen $a = d \cdot \sigma_\alpha = 300mm$ und $b = \sigma_d = 100mm$. Die Ellipse ist um $30°$ gegenüber der x-Achse gedreht.

2.5 Kalman-Filter

Oftmals ist die Situation gegeben, daß mehrere Messungen von verschiedenen Sensoren über den gleichen Sachverhalt vorliegen. Jede dieser Messungen liefert eine normalverteilte Beobachtung mit Erwartungswert und Varianz. Die Frage ist nun, wie man die Meßergebnisse der Sensoren miteinander verknüpft, um ein endgültiges Ergebnis zu bekommen, das genauer ist und einen kleineren Fehler besitzt.

Hierzu wird der Kalman-Filter eingesetzt. Voraussetzung ist, daß alle Schätzungen mit einem normalverteilten Fehler vorliegen. Der Kalman-Filter fusioniert die Schätzungen dann zu einer neuer Schätzung mit neuem Fehler, der wiederum normalverteilt ist. Als Optimalitätskriterium für die Wahl der neuen Schätzung dient das Kriterium der minimalen Varianz, d.h. die neue Schätzung wird so gewählt, daß der resultierende Fehler minimiert wird.

2.5.1 Eindimensionaler Fall

Die Vorgehensweise und Wirkung des Kalman-Filters kann am besten anhand eines Beispiels im skalaren Fall gezeigt werden. Angenommen zwei Sensoren zur Entfernungsmessung seien so aufgestellt, daß sie den gleichen Abstand zu dem vor ihnen liegenden Objekt, z.B. einer Wand, besitzen. Die Sensoren müßten also den gleichen Entfernungswert messen. Weiterhin nimmt man an, daß die Messungen der Sensoren mit einem Fehler behaftet sind. Die Messungen l_1 von Sensor 1 und l_2 von Sensor 2 seien wie folgt verteilt:

$$\begin{aligned} l_1 &\sim N(\mu_1, \sigma_1^2) \\ l_2 &\sim N(\mu_2, \sigma_2^2) \end{aligned}$$

Hierbei entsprechen μ_1 und μ_2 den realen Entfernungen und σ_1 und σ_2 sind die jeweiligen Standardabweichungen, die man durch eine Meßreihe bestimmen kann. Das Problem ist nun, wie man die Daten der beiden Sensoren verknüpft, um aus je einer vorliegenden Messung der beiden Sensoren einen Entfernungswert zu berechnen, der die Eigenschaften beider Sensoren berücksichtigt. Es ist naheliegend, hier ein gewichtetes arithmetisches Mittel der beiden Entfernungsmessungen zu verwenden. Die jeweiligen Gewichte sind dabei umgekehrt proportional zu der zugehörigen Varianz.

Seien l_1 und l_2 die Messungen der beiden Sensoren, dann berechnet sich der durch Kalman-Filterung bestimmte neue Abstandswert l mit Varianz σ^2 zu

$$l = \frac{1}{\frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2}}\left(\frac{1}{\sigma_1^2}l_1+\frac{1}{\sigma_2^2}l_2\right) \tag{2.19}$$

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_1^2}+\frac{1}{\sigma_2^2} \tag{2.20}$$

Im Fall $\sigma_1 = \sigma_2$ (z.B. beim Einsatz zweier identischer Meßgeräte) reduzieren sich die Gleichungen zu

$$l = \frac{1}{2}(l_1+l_2) \tag{2.21}$$

$$\sigma^2 = \frac{\sigma_1^2}{2} \tag{2.22}$$

Es wird also erwartungsgemäß das arithmetische Mittel gebildet und die Varianz halbiert sich.

Angenommen Sensor 1 ist gegenüber Sensor 2 sehr viel genauer, d.h. $\sigma_1 \ll \sigma_2$, so wird der Messung von Sensor 1 sehr viel mehr Gewicht gegeben, da nun der Reziprokwert von σ_1^2 sehr viel größer als der Reziprokwert von σ_2^2 ist. Die resultierende Varianz ist jedoch immer kleiner als jede der beiden Varianzen, was intuitiv auch richtig ist, da durch die zweite Messung der Fehler ja verkleinert werden soll.

Ein weiteres sehr anschauliches Beispiel findet man in in der Arbeit von Maybeck [1990].

2.5.2 Mehrdimensionaler Fall

Im n-dimensionalen Fall liegen zwei n-dimensionale Vektoren m_1 und m_2 vor, mit

$$m_1 \sim N(\mu_1, \Sigma_1)$$
$$m_2 \sim N(\mu_2, \Sigma_2)$$

Die durch Kalman-Filterung berechnete verbesserte Schätzung m mit Kovarianzmatrix Σ ergibt sich dann zu [Maybeck, 1990]:

$$m = (\Sigma_1^{-1}+\Sigma_2^{-1})^{-1}(\Sigma_1^{-1}m_1+\Sigma_2^{-1}m_2) \tag{2.23}$$

$$\Sigma = (\Sigma_1^{-1}+\Sigma_2^{-1})^{-1} \tag{2.24}$$

An dieser Stelle sei angemerkt, daß der Kalman-Filter in der Regel so angewendet wird, daß zunächst eine Vorhersage aufgrund des aktuellen Zustandes stattfindet und die Vorhersage mit der tatsächlichen Sensormessung verglichen wird. Hieraus wird dann eine Aktualisierung für den Systemzustand berechnet. Das Beispiel von Maybeck [1990] und die in Kapitel 4 vorgestellten Verfahren zur Selbstlokalisierung eines mobilen Roboters verwenden diese Vorgehensweise.

Nachdem nun die mathematischen Grundlagen für diese Arbeit bekannt sind, werden im nächsten Kapitel Methoden zum Überdecken von Scans vorgestellt.

Kapitel 3

Scan-Matching

Scan-Matching ist der Prozeß, einen Scan so zu drehen und zu verschieben, daß ein maximale Überlappung mit einem Referenzmodell entsteht. Scan-Matching spielt in der mobilen Robotik eine wichtige Rolle bei der Berechnung der eigenen Position und bei der Erstellung von hochgenauen Umgebungskarten. Insbesondere in dieser Dissertation wird Scan-Matching für fast alle Bereiche der Navigation mobiler Roboter eingesetzt.

Dieses Kapitel beschreibt zunächst, was unter einem Scan genau verstanden wird.

Danach werden Algorithmen vorgestellt, welche direkt auf den Scandaten operieren, z.B. Extraktion von Liniensegmenten oder die Projektion eines Scans auf eine andere Position.

Anschließend werden Methoden beschrieben, die Punkte eines Scans klassifizieren und ungewünschte Scanpunkte herausfiltern. Auf diese Weise können z.B. Scans geglättet werden, die Datenmenge kann reduziert werden ohne dabei viel an Information zu verlieren, oder es kann nur der polygonale Anteil eines Scans betrachtet werden.

Darauf folgend wird der Begriff des Scan-Matching definiert und verschiedene Lösungsverfahren für das Überdecken zweier Scans vorgestellt. Wichtige Eigenschaften der einzelnen Verfahren werden festgehalten und Verfahren werden miteinander verglichen.

Teile dieses Kapitels sind bereits in [Gutmann, 1996; Gutmann und Schlegel, 1996] zu finden. Jedoch wurden viele der dort präsentierten Verfahren erweitert (z.B. für 180° Scans) und verbessert (z.B. Verbesserung des erweiterten Cox-Algorithmus).

3.1 Scan

Ein Scan ist ein Menge von Meßwerten $\{m_i = (\alpha_i, r_i)^T \mid i = 0 \ldots n-1\}$, welche in Polarkoordinaten $(\alpha_i, r_i)^T$ angegeben sind. Im folgenden werden Meßwerte eines

Scans als Scanpunkte bezeichnet.

Ein Scanpunkt $m_i = (\alpha_i, r_i)^T$ kann für eine gegebene Aufnahmeposition $l = (x, y, \theta)^T$ wie folgt in absolute kartesische Koordinaten umgerechnet werden.

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + R_2(\theta) \begin{pmatrix} r_i \cos \alpha_i \\ r_i \sin \alpha_i \end{pmatrix} \tag{3.1}$$

Abbildung 3.1 zeigt einen typischen Scan wie er von einem kommerziellen 180°-Laserscanner mit 0.5° Winkelauflösung und $5cm$ Entfernungsauflösung aufgenommen wird.

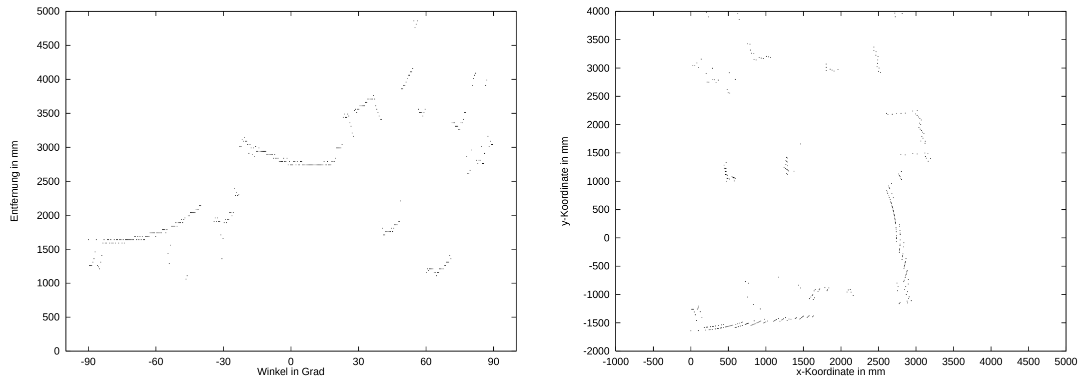


Abbildung 3.1: Typischer Scan, welcher von dem 180° Laserscanner SICK PLS-200 erfaßt wird. Links in Polarkoordinaten, rechts in kartesischen Koordinaten.

Die Meßdaten eines Laserscanners zeichnen sich durch eine hohe Winkelauflösung (meist $< 1°$) und eine hohe Genauigkeit in der Entfernungsmessung aus und sind daher wesentlich reichhaltiger, exakter und zuverlässiger als die vieler anderer Abstandssensoren wie z.B. Ultraschall.

Die Messungen eines Scans erfolgen für gewöhnlich in einer festen Reihe, z.B. mit aufsteigendem Winkel. Diese Eigenschaft wird in den folgenden Verfahren ausgenutzt.

3.2 Merkmalsextraktion in Scandaten

Oftmals ist es von Vorteil, nicht direkt auf den einzelnen Scanpunkten zu arbeiten, sondern Merkmale aus den Scans zu extrahieren. In diesem Abschnitt wird beschrieben, wie aus einem Scan Linien und Ecken extrahiert werden können.

3.2.1 Extraktion von Linien

Folgender Algorithmus beschreibt wie aus einem Scan Liniensegmente gewonnen werden können. Dazu werden Scanpunkte gruppiert und für jede Gruppe mittels der Funktion *split* Liniensegmente extrahiert.

Algorithmus 3.1: *line-extraction*(s)
Eingabe: Scan s
Ausgabe: Menge von Liniensegmenten l
Ablauf:

```
l := empty
start = 0
for i := 1 to numpoints(s) - 1 do
    p_1 := n-th-scanpoint(s, i - 1)
    p_2 := n-th-scanpoint(s, i)
    if distance(p_1, p_2) > MAX-DISTANCE then
        l := l ∪ split(s, start, i - 1)
        start := i
    endif
endfor
l := l ∪ split(s, start, numpoints(s) - 1)
return l
```

Die Konstante *MAX-DISTANCE* bestimmt den maximalen Abstand zweier aufeinanderfolgender Scanpunkte für die Gruppierung. Die Funktion *split* erledigt die eigentliche Aufgabe der Linienextraktion:

Algorithmus 3.2: *split*$(s, start, end)$
Eingabe: Gruppe von Scanpunkten, die durch s, $start$ und end festgelegt ist
Ausgabe: Menge von Liniensegmenten l
Ablauf:

```
l := empty
line := make-line(s, start, end)
if numpoints(line) ≥ MIN-POINTS-ON-LINE then
    if σ(line) < MAX-SIGMA then
        l := l ∪ {line}
    else
        p_start := n-th-scanpoint(s, start)
        p_end := n-th-scanpoint(s, end)
        i_split := start
        d := 0
        for i := start + 1 to end - 1 do
            p := n-th-scanpoint(s, i)
            if distance-to-line(p, p_start, p_end) > d then
```

$i_{split} := i$
$d :=$ *distance-to-line*(p, p_{start}, p_{end})
endif
endfor
$l := l \cup split(s, start, i_{split})$
$l := l \cup split(s, i_{split}, end)$
endif
endif
return l

Die *split*-Funktion ist rekursiv. Zunächst wird durch die gesamte Gruppe von Scanpunkten eine Ausgleichsgerade (*make-line*) gelegt. Sofern die erzeugte Linie genügend Scanpunkte enthält und die Standardabweichung σ nicht zu groß ist, wird die Linie in die Linienmenge aufgenommen und zurückgesprungen. Falls jedoch die Standardabweichung zu groß ist, aber noch genügend Punkte vorhanden sind, um mindestens eine Linie zu bilden, so wird die Gruppe von Scanpunkten an einem bestimmten Punkt in zwei Untergruppen aufgeteilt und die *split*-Funktion mit beiden Untergruppen rekursiv aufgerufen. Es wird dann die Vereinigung der beiden von der *split*-Funktion bestimmten Linienmengen zurückgegeben. Der Scanpunkt, an dem die Gruppe aufgeteilt wird, ist durch den Punkt festgelegt, welcher den maximalen Abstand zu der Geraden durch den Start- und Endpunkt der Gruppe hat. Er wird in der **for**-Schleife bestimmt. Die Funktion *distance-to-line*(p, p_1, p_2) liefert dabei den Abstand des Punktes p zu der Geraden, die durch die Punkte p_1 und p_2 festgelegt ist.

Die Konstanten *MIN-POINTS-ON-LINE* und *MAX-SIGMA* bestimmen die Anzahl und Qualität der erzeugten Linien. Ein kleiner Wert für *MAX-SIGMA* erzeugt im Mittel mehr Linien als ein hoher Wert. Dafür ist die Genauigkeit der Linien für kleinere Werte besser als für größere. Für den SICK-Laserscanner PLS 200 und eine typischen Büroumgebung wie die des KI-Lehrstuhl der Universität Freiburg liefern Werte zwischen $30mm$ und $50mm$ für *MAX-SIGMA* einen guten Kompromiß zwischen niedriger Linienzahl und hoher Genauigkeit.

MIN-POINTS-ON-LINE legt die minimale Anzahl von Punkten pro Linie fest. Je kleiner der Wert gewählt wird, umso mehr Linien werden erzeugt. Allerdings wird dadurch auch die Gefahr größer, daß Linien in verauschte Sensordaten gelegt werden, die durch nicht-polygonale Objekte erzeugt wurden. Brauchbare Werte für *MIN-POINTS-ON-LINE* liegen zwischen 5 und 20.

Die *split*-Funktion ruft zu Beginn die Funktion *make-line* auf. Diese legt eine Ausgleichsgerade durch die übergebene Gruppe von Scanpunkten. Die Funktion kann wie folgt realisiert werden: Seien (x_i, y_i), $i = 0 \dots n-1$ die kartesischen Koordinaten der Scanpunkte, durch die eine Gerade gelegt werden soll. Eine Gerade sei durch die Parameter α und d festgelegt, so daß für alle Punkte (x, y) auf der Geraden gilt:

$$x \cos\alpha + y \sin\alpha - d \;=\; 0$$

Die Ausgleichsgerade wird so gelegt, daß die Summe der Abstandsquadrate

$$E_{fit} = \sum_{i=0}^{n-1} (x_i \cos\alpha + y_i \sin\alpha - d)^2$$

minimiert wird. Die Lösungen für α und d und die damit verbundene Standardabweichung σ berechnen sich dann zu [Lu, 1995, Seiten 41-43]:

$$\begin{aligned}
\alpha &= \frac{1}{2}\tan^{-1}\frac{-2S_{xy}}{S_{y^2} - S_{x^2}} \\
d &= \bar{x}\cos\alpha + \bar{y}\sin\alpha \\
\sigma^2 &= \frac{1}{2n}\left(S_{x^2} + S_{y^2} - \sqrt{4S_{xy}^2 + (S_{y^2} - S_{x^2})^2}\right)
\end{aligned}$$

mit

$$\begin{aligned}
\bar{x} &= \frac{1}{n}\sum_{i=0}^{n-1} x_i \\
\bar{y} &= \frac{1}{n}\sum_{i=0}^{n-1} y_i \\
S_{x^2} &= \sum_{i=0}^{n-1}(x_i - \bar{x})^2 \\
S_{y^2} &= \sum_{i=0}^{n-1}(y_i - \bar{y})^2 \\
S_{xy} &= \sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})
\end{aligned}$$

Abbildung 3.2 zeigt, wie der Algorithmus auf einer gegebenen Punktemenge arbeitet. Im obersten Bild wird versucht, eine Ausgleichsgerade durch alle Punkte zu legen (durchgezogene Linie). Die Standardabweichung ist hier jedoch sehr groß, also wird die Punktemenge an dem Punkt, welcher mit einem kleinen Pfeil markiert ist, aufgeteilt. Dieser Punkt hat von der Strecke, die durch den Start- und Endpunkt geht (gestrichelte Linie), den maximalen Abstand. Im zweiten Bild paßt die Ausgleichsgerade für den linken Teil bereits, der rechte Teil muß jedoch nochmals unterteilt werden. In Bild 3 sind dann alle Linien erzeugt worden.

Abbildung 3.3 zeigt einen Beispielscan und die daraus extrahierten Liniensegmente. Da die Umgebung, in welcher die Scanaufnahme erfolgte, hauptsächlich polygonal ist, kann beinahe der gesamte Scan in Liniensegmente umgewandelt werden.

Die Linienextraktion ist ein typischer *divide and conquer* Algorithmus. Die Zeitkomplexität ist mit der des *Quicksort*-Algorithmus vergleichbar, d.h. $O(n^2)$ im

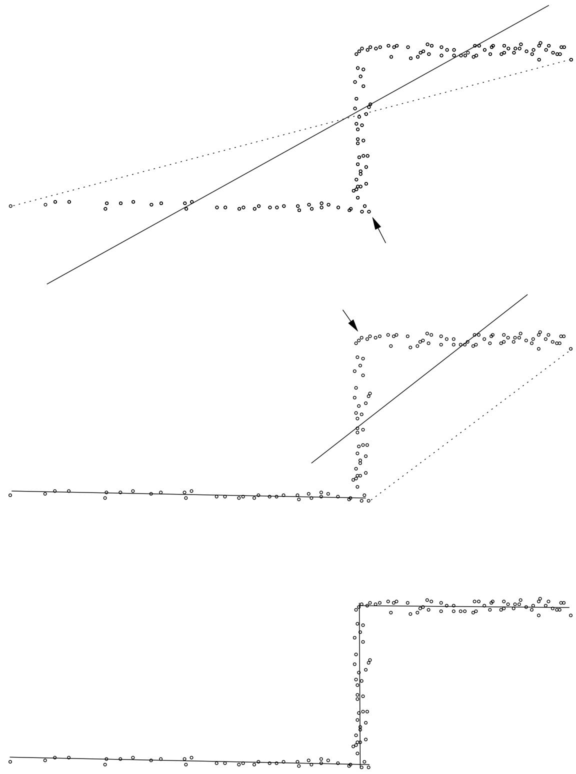


Abbildung 3.2: Extraktion von Linien aus einer Gruppe von Scanpunkten.

schlechtesten und $O(n \log n)$ im mittleren Fall, wenn n die Anzahl der Scanpunkte ist.

In der Literatur befinden sich zahlreiche weitere Verfahren zur Extraktion von Liniensegmenten aus Scandaten. Das Verfahren von Castellanos und Tardós [1996] ist dem hier beschriebenen Verfahren sehr ähnlich. Einziger Unterschied ist, daß nicht sofort eine Ausgleichsgerade in eine Gruppe von Scanpunkten gelegt wird, sondern zunächst geprüft wird, ob die Gruppe an dem Punkt mit maximalen Abstand zu der Geraden durch Start- und Endpunkt weiter aufgeteilt werden muß. Weitere Verfahren zur Extraktion von Liniensegmenten aus Scandaten befinden sich in [Vestli, 1995; Arras *et al.*, 1996; Arras und Siegwart, 1997]. Diese Verfahren sind weitaus komplexer bieten dafür aber weitere Eigenschaften, wie z.B. Robustheit gegenüber Ausreißern, Zusammenlegen von co-linearen Segmenten, und die Extraktion anderer geometrischer Primitive (z.B. Kreise).

3.2.2 Extraktion von Ecken

Auf ähnliche Weise wie Linien aus den Scandaten extrahiert werden, können auch Ecken generiert werden, indem aufeinanderfolgende Linien miteinander geschnitten werden. Hierzu muß nur die Funktion *split* in der Linienextraktion durch

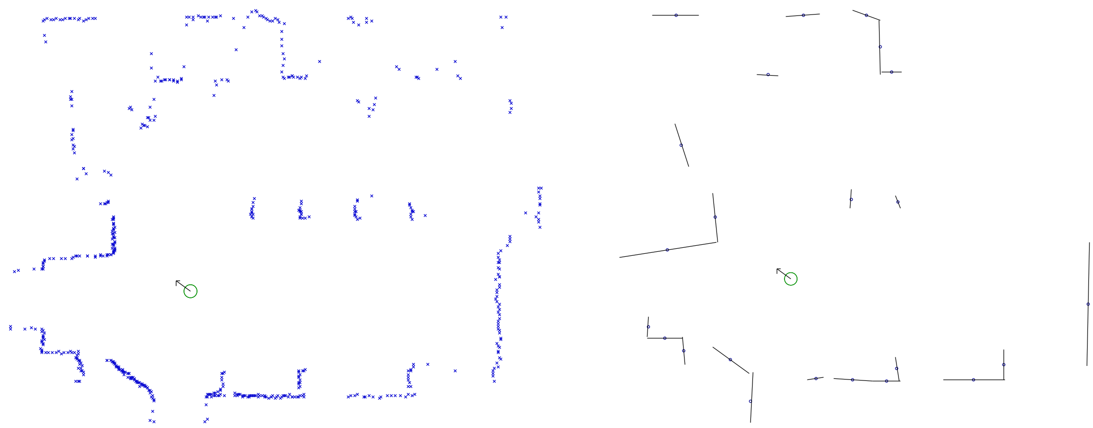


Abbildung 3.3: Extraktion von Linien aus einem Scan. Links roher Scan, rechts extrahierte Linien.

folgende Funktion ersetzt werden.

Algorithmus 3.3: $split(s, start, end)$
Eingabe: Gruppe von Scanpunkten, die durch s, $start$ und end festgelegt ist
Ausgabe: Menge von Ecken e
Ablauf:

$e := empty$
if $(end - start) \geq 2MIN\text{-}POINTS\text{-}CORNER$ **then**
  $p_{start} := n\text{-}th\text{-}scanpoint(s, start)$
  $p_{end} := n\text{-}th\text{-}scanpoint(s, end)$
  $i_{split} := start$
  $d := 0$
  for $i := start + 1$ **to** $end - 1$ **do**
    $p := n\text{-}th\text{-}scanpoint(s, i)$
    if $distance\text{-}to\text{-}line(p, p_{start}, p_{end}) > d$ **then**
      $i_{split} := i$
      $d := distance\text{-}to\text{-}line(p, p_{start}, p_{end})$
    endif
  endfor
  if $(i_{split} - start) \geq MIN\text{-}POINTS\text{-}CORNER$ **and**
  $(end - i_{split}) \geq MIN\text{-}POINTS\text{-}CORNER$ **then**
    $line_1 := make\text{-}line(s, i_{split} - MIN\text{-}POINTS\text{-}CORNER, i_{split})$
    $line_2 := make\text{-}line(s, i_{split}, i_{split} + MIN\text{-}POINTS\text{-}CORNER)$
    if $\sigma(line_1) < MAX\text{-}SIGMA$ **and** $\sigma(line_2) < MAX\text{-}SIGMA$ **then**
      $e := e \cup \{make\text{-}corner(line_1, line_2)\}$
    endif
  endif
  $e := e \cup split(s, start, i_{split})$

$e := e \cup split(s, i_{split}, end)$
endif
return e

Die Funktion untersucht zunächst, ob noch genügend Scanpunkte für die Extraktion einer Ecke vorhanden sind. Falls dem so ist, so wird wie bei der Linienextraktion der Scanpunkt gesucht, der von der Geraden durch Anfangs- und Endpunkt der Punktegruppe den maximalen Abstand hat. An diesem Punkt wird jeweils eine Untergruppe von *MIN-POINTS-CORNER* Scanpunkten vor und nach diesem Punkt gebildet und, falls genügend Scanpunkte in beiden Untergruppen sind, jeweils eine Gerade durch diese Punkte gelegt. Falls schließlich die Standardabweichungen beider Geraden nicht zu groß sind, so wird mittels der Funktion *make-corner* ein Ecke generiert. Dazu berechnet *make-corner* Schnittpunkt, Lage und Öffnungswinkel der beiden Geraden. Die Parameter *MAX-SIGMA* und MIN-POINTS-CORNER bestimmen wie bei der Linienextraktion Anzahl und Güte der extrahierten Ecken.

Abbildung 3.4 zeigt einen Beispiel-Scan und die daraus extrahierten Ecken. In diesem Beispiel sind nur rechte Winkel dargestellt, d.h. nur Ecken mit einem Öffnungswinkel von ungefähr 90° Grad.

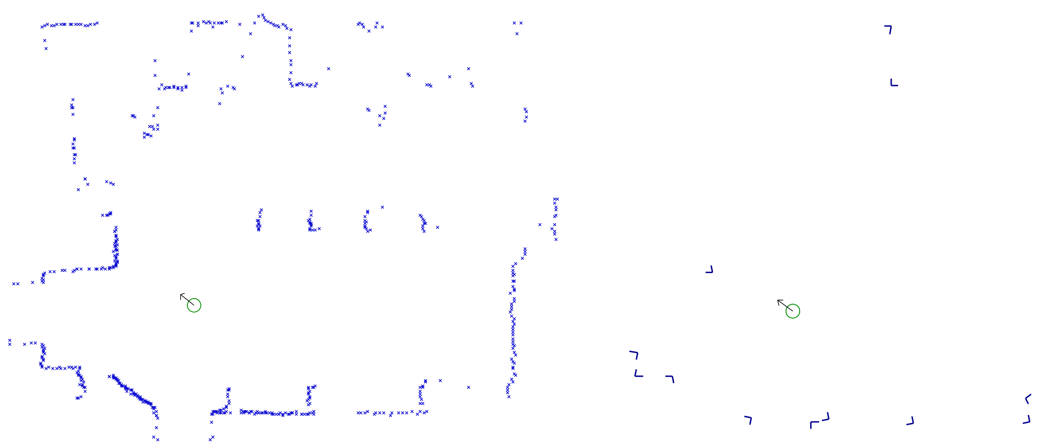


Abbildung 3.4: Extraktion von 90° Ecken aus einem Scan. Links roher Scan, rechts extrahierte Ecken.

Die Extraktion von Ecken hat die gleiche Komplexität wie die Linienextraktion, d.h. $O(n^2)$ im schlechtesten und $O(n \log n)$ im mittleren Fall.

Weitere Verfahren zur Extraktion von Ecken aus Scandaten, welche ebenfalls Ecken durch Schneiden aufeinanderfolgender Geraden bestimmen, befinden sich in [Vestli, 1995; Castellanos *et al.*, 1996a].

3.3 Projektion von Scans

Bei der Projektion eines Scans wird für jeden Punkt des Scans untersucht, ob er von einer gegebenen, anderen Aufnahmeposition aus sichtbar ist [Lu, 1995; Lu und Milios, 1997b].

Abbildung 3.5, oberes Bild, zeigt einen Scan, der an Position x aufgenommen wurde und nun von Position y aus betrachtet werden soll. Von Position x aus liegen alle Punkte im aufsteigendem Winkel vor, von Position y aus liegen die Punkte innerhalb der gepunkteten Ellipse jedoch im absteigenden Winkel vor, d.h. von y aus sieht man die Punkte „von hinten“, sie sind daher nicht sichtbar. Gleichzeitig wird durch diese Punkte ein großer Teil des restlichen Scans verdeckt, diese Punkte sind ebenfalls nicht sichtbar. Entfernt man alle nicht sichtbaren Punkte, so erhält man den Scan, der in Abbildung 3.5, unteres Bild, dargestellt ist.

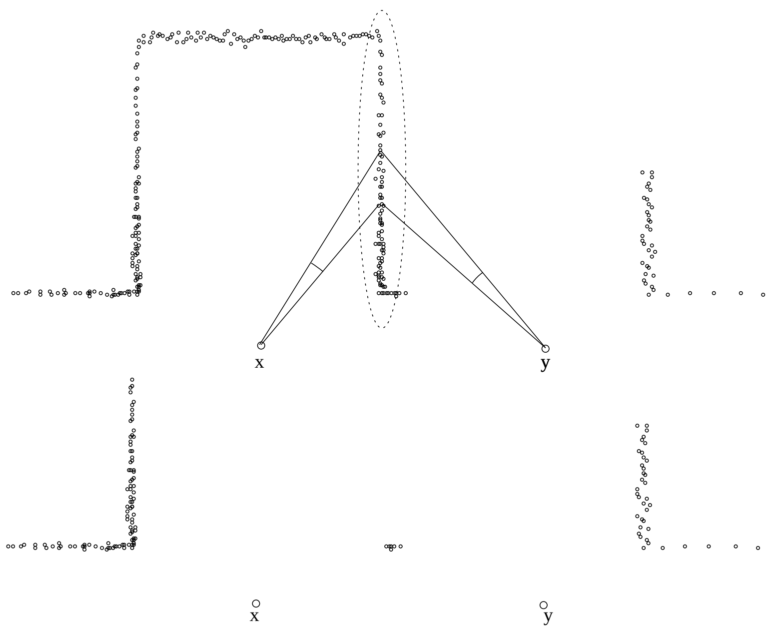


Abbildung 3.5: Projektion eines Scans: Die Punkte in der Mitte des oberen Bilds (gepunktete Ellipse) sind von y aus nicht sichtbar. Nach Entfernen aller von y aus nicht sichtbaren Punkte erhält man das untere Bild.

Der Algorithmus zur Kennzeichnung der unsichtbaren Punkte geht wie folgt vor:

Algorithmus 3.4: *scan-project*(s, y)
Eingabe: Scan s; Position y
Ausgabe: Markierung der sichtbaren Scanpunkte
Ablauf:

for $i := 0$ **to** $numpoints(s) - 1$ **do**
  $p :=$ *n-th-scanpoint*(s, i)
  $p_{pred} := pred(p)$
  $p_{succ} := succ(p)$
  if *angle-to-position*$(p_{pred}, y) <$ *angle-to-position*(p_{succ}, y) **then**
    $visible(i) := TRUE$
  else
    $visible(i) := FALSE$
  endif
endfor
$visible :=$ *mark-occluded*$(s, visible, y, s, 0)$
return *visible*

Der Algorithmus überprüft für alle Scanpunkte, ob sie im aufsteigenden Winkel vorliegen. Dazu werden die Funktionen $pred(p)$ und $succ(p)$ benötigt, welche den Vorgänger bzw. Nachfolger eines Scanpunktes liefern. Hier könnte jeweils der direkte Nachbarpunkt bestimmt werden. Da der Scan jedoch verrauschte Anteile enthält, kann es von Vorteil sein, nicht den unmittelbaren Nachbarn zu verwenden, sondern einen weiter entfernten Punkt. Für Punkte, die mit aufsteigendem Winkel vorliegen, wird die Sichtbarkeitsmarke gesetzt, für alle anderen gelöscht.

Zuletzt wird die Funktion *mark-occluded* aufgerufen, welche Verdeckungen erkennt und die Sichtbarkeitsmarke von verdeckten Scanpunkten löscht. Diese Funktion nimmt als Parameter den zu untersuchenden Scan s, die zugehörigen Sichtbarkeitsmarken *visible*, die Position y, auf welche der Scan projiziert wird, den Rahmenscan, welcher potentiell Scanpunkte verdeckt (hier der Scan s selbst) und einen Mindestabstand, um zu entscheiden, wann ein Scanpunkt verdeckt ist (hier 0, da der Scan s selbst Rahmenscan ist). Der Algorithmus für *mark-occluded* sieht wie folgt aus.

Algorithmus 3.5: *mark-occluded*$(s, visible, y, frame, dist)$
Eingabe: Scan s; Marken *visible*; Position y; Scan $frame$; Entfernung *dist*
Ausgabe: Markierung der sichtbaren Scanpunkte
Ablauf:
for $i := 0$ **to** $numpoints(s) - 1$ **do**
  if $visible(i) = TRUE$ **then**
    $p :=$ *n-th-scanpoint*(s, i)
    $a :=$ *angle-to-position*(p, y)
    $d :=$ *distance-to-position*(p, y)
    for $j := 0$ **to** $numpoints(frame) - 1$ **do**
      $k := (j + 1) \bmod numpoints(frame)$
      if $s \neq frame$ **or** $(i \neq j$ **and** $i \neq k)$ **then**
        $p :=$ *n-th-scanpoint*$(frame, j)$
        $a_1 :=$ *angle-to-position*(p, y)
        $p :=$ *n-th-scanpoint*$(frame, k)$

$a_2 := \textit{angle-to-position}(p, y)$
if $a_1 > a_2$ **then**
$a_2 := a_1$
$a_1 := \textit{angle-to-position}(p, y)$
endif
if $a_1 < a$ **and** $a < a_2$ **then**
if $d - \textit{distance-to-position}(p, y) > \textit{dist}$ **then**
$\textit{visible}(i) := \textit{FALSE}$
endif
endif
endfor
endif
endfor
return *visible*

Die Komplexität dieses Algorithmus ist $O(n^2)$, wobei n die Anzahl der Scanpunkte ist. Durch eine Modifikation der Funktion *mark-occluded* kann die Konstante, die in $O(n^2)$ steckt, stark verkleinert werden. Dazu werden Scanpunkte geeignet zu Winkelbereichen zusammengefaßt. Nun müssen nur noch die Winkelbereiche überprüft und Überschneidungen genauer untersucht werden.

Die Projektion eines Scans auf eine andere Position ist sehr nützlich für das Überdecken zweier Scans. Hierzu sollte vorher stets der Projektionsfilter angewandt werden, welcher im Abschnitt 3.4.5 erläutert wird.

3.4 Filterung von Scandaten

Für viele scanverarbeitenden Algorithmen ist es von Vorteil, den zu bearbeitenden Scan vorher mit verschiedenen Filtern zu behandeln, um z.B. ungewünschte Scanpunkte zu entfernen, oder um die Datenmenge zu reduzieren. In diesem Abschnitt werden verschiedene Filterverfahren vorgestellt. Dies sind der Medianfilter, welcher zum Glätten der einzelnen Messungen dient, der Reduktionsfilter, der Punktewolken zu einem Punkt zusammenfaßt und die Verteilung der Punkte verbessert sowie ihre Anzahl reduziert, der Winkelreduktionsfilter, welcher wie der Reduktionsfilter Punkte zusammenfaßt, der Linienfilter, der Punkte eliminiert, die nicht auf einer Linie liegen, und der Projektionsfilter, welcher für zwei Scans, diejenigen Punkte bestimmt, die potentiell einen Korrespondenzpartner im jeweils anderen Scan besitzen.

3.4.1 Medianfilter

Der Medianfilter ist in der Lage, Ausreißer in einem Scan zu erkennen und durch eine geeignete Messung zu ersetzen. Hierzu wird für jeden Scanpunkt ein Fenster um den Scanpunkt gelegt, das neben dem Scanpunkt selbst die letzten paar Messungen vor und nach diesem Punkt enthält. Der Scanpunkt wird dann durch einen neuen Scanpunkt ersetzt, welcher denselben Aufnahmewinkel besitzt, jedoch für die Entfernung den Median der Entfernungswerte im betrachteten Fenster nimmt.

Folgender Algorithmus realisiert den Medianfilter:

Algorithmus 3.6: *median-filter*(s)
Eingabe: Scan s
Ausgabe: Scan s'
Ablauf:
    for $i := 0$ **to** $numpoints(s) - 1$ **do**
        $p :=$ *n-th-scanpoint*(s, i)
        for $j := 0$ **to** *MEDIAN-NUM-POINTS* $- 1$ **do**
            $k := (i + j -$ *MEDIAN-NUM-POINTS*$/2)$ mod $numpoints(s)$
            $p_k :=$ *n-th-scanpoint*(s, k)
            $d(j) :=$ *distance-value*(p_k)
        endfor
        $d_{median} := median(d)$
        n-th-scanpoint$(s', i) := ($*angle-value*$(p), d_{median})$
    endfor
    return s'

Der Parameter *MEDIAN-NUM-POINTS* bestimmt die Fenstergröße, mit welcher der Medianfilter arbeitet. Ein großer Wert bedeutet eine starke Glättung des Scans, d.h. es gibt keine großen Entfernungsschwankungen mehr. Ein kleiner Wert erlaubt große Schwankungen in der Entfernung, gleichzeitig werden aber weniger Ausreißer erkannt. In der Praxis hat sich ein Wert von *MEDIAN-NUM-POINTS* $= 5$ bewährt.

Die Komplexität des Medianfilters ist $O(n*m)$, wenn n die Anzahl Scanpunkte und m die Fenstergröße ist (die Bestimmung des Medians kann in $O(m)$ erfolgen).

Abbildung 3.6 zeigt einen Beispielscan vor und nach Anwendung des Medianfilters mit einer Fenstergröße von 5 Scanpunkten. Es ist gut zu sehen, wie das Sensorrauschen durch Anwendung des Medianfilters reduziert wird.

Die Vorteile des Medianfilters liegen in der Entfernung von Ausreißern und in der Reduktion von Sensorrauschen. Ein Nachteil des Medianfilters ist, daß Ecken abgerundet werden. Zum Einsatz kommt der Medianfilter, z.B. bei der Objekterkennung. Hier wird vor der eigentlichen Objekterkennung der Medianfilter angewandt, um glattere Konturen zu erhalten. Bei der Extraktion von Merkmalen und dem Überdecken von Scans bietet die vorherige Anwendung des Medianfilters auf die beteiligten Scans keine Vorteile.

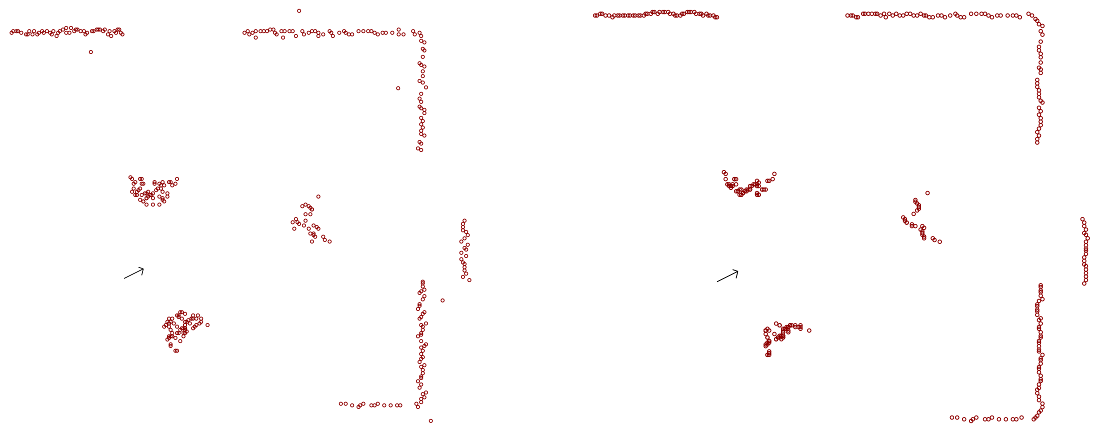


Abbildung 3.6: Anwendung des Medianfilters auf einen Scan. Links vor, rechts nach Anwendung des Filters.

3.4.2 Reduktionsfilter

Der Reduktionsfilter faßt Punktwolken zu einem Punkt zusammen. Eine Punktwolke wird durch einen Radius angegeben und nach folgendem Schema bestimmt: Der erste Punkt eines Scans gehört zu einer Wolke. Dieser Punkt sei im folgenden Ausgangspunkt genannt. Alle nachfolgenden Punkte, die einen Abstand zum Ausgangspunkt kleiner dem zweifachen Radius haben, werden der Wolke des Ausgangspunkts hinzugefügt. Beim ersten Punkt, der einen Abstand größer oder gleich dem zweifachen Radius hat, wird diese Wolke geschlossen und eine neue Wolke mit dem aktuellen Punkt als Ausgangspunkt geöffnet. Jede Wolke wird durch den Schwerpunkt der in ihr enthaltenden Scanpunkte ersetzt.

Folgender Algorithmus realisiert den Reduktionsfilter:

Algorithmus 3.7: *reduction-filter*(s, r)
Eingabe: Scan s; Radius r
Ausgabe: Scan s'
Ablauf:

$j := 0$
$p_0 :=$ *n-th-scanpoint*$(s, 0)$
$p_{sum} := p_0$
$n := 1$
for $i := 1$ **to** *numpoints*$(s) - 1$ **do**
  $p :=$ *n-th-scanpoint*(s, i)
  if *distance*$(p_0, p) < 2r$ **then**
    $p_{sum} := p_{sum} + p$
    $n := n + 1$
  else
    n-th-scanpoint$(s', j) := p_{sum}/n$

$j := j + 1$
$p_0 := p$
$p_{sum} := p_0$
$n := 1$
endif
endfor
$n\text{-}th\text{-}scanpoint(s', j) := p_{sum}/n$
$j := j + 1$
$numpoints(s') := j$
return s'

Der Algorithmus hat eine Komplexität von $O(n)$, wenn n die Anzahl der Scanpunkte ist.

Abbildung 3.7 zeigt einen Scan vor und nach Durchlaufen des Reduktionsfilters mit $r = 5cm$. Man sieht, daß nach der Filterung deutlich weniger Scanpunkte zurückbleiben und diese nun besser über den gesamten Bereich verteilt sind als vorher.

Abbildung 3.7: Anwendung des Reduktionsfilters mit $r = 5cm$ auf einen Scan. Links vor, rechts nach Anwendung des Filters.

Der Vorteil von Scans, die mit dem Reduktionsfilter behandelt wurden, ist eine Reduzierung der Anzahl der Scanpunkte ohne dabei wesentliche Information verloren zu haben. Dies führt bei nachfolgenden Algorithmen zu kürzeren Laufzeiten. Weiterhin ist die Verteilung der Punkte über den Scan nach Behandlung mit dem Reduktionsfilter besser (im Sinne von Gleichverteilung) als vorher. Allerdings ist die Extraktion von Merkmalen nicht mehr so einfach, da möglicherweise zu wenige Punkte für ein Merkmal übrig bleiben. Eine Merkmalsextraktion sollte daher vor Einsetzen des Reduktionsfilters geschehen.

3.4.3 Winkelreduktionsfilter

Ein zum Reduktionsfilter sehr ähnlicher Filter ist der Winkelreduktionsfilter, welcher Scanpunkte mit ähnlichem Aufnahmewinkel gruppiert und durch den Punkt ersetzt, dessen Entfernungswert gleich dem Median der Entfernungswerte ist.

Folgender Algorithmus beschreibt den Winkelreduktionsfilter:

Algorithmus 3.8: *angle-reduction-filter*(s, α)
Eingabe: Scan s; Winkel α
Ausgabe: Scan s'
Ablauf:

$j := 0$
$q(0) :=$ *n-th-scanpoint*$(s, 0)$
$n := 1$
for $i := 1$ **to** *numpoints*$(s) - 1$ **do**
  $p :=$ *n-th-scanpoint*(s, i)
  if *abs*(*angle-value*(p) − *angle-value*$(q(0))) < \alpha$ **then**
    $q(n) := p$
    $n := n + 1$
  else
    n-th-scanpoint$(s', j) :=$ *median-dist*(q, n)
    $j := j + 1$
    $q(0) := p$
    $n := 1$
  endif
endfor
n-th-scanpoint$(s', j) :=$ *median-dist*(q, n)
$j := j + 1$
numpoints$(s') := j$
return s'

Hierbei liefert *median-dist*(q, n) den Scanpunkt aus der Menge q von Scanpunkten, dessen Entfernungswert gleich dem Median der Entfernungswerte ist.

Die Komplexität des Winkelreduktionsfilters ist $O(n)$, wenn n die Anzahl der Scanpunkte ist.

Abbildung 3.8 zeigt einen Beispielscan vor und nach Anwendung des Winkelreduktionsfilters.

Der Winkelreduktionsfilter kann verwendet werden, um die Datenmenge eines Scans, welcher mit hoher Winkelauflösung aufgenommen wurde, gleichmäßig zu reduzieren. Dies wird z.B. nach der Verschmelzung mehrere Scans zu einem Scan eingesetzt.

Abbildung 3.8: Anwendung des Winkelreduktionsfilters mit $\alpha = 2°$ auf einen Scan. Links vor, rechts nach Anwendung des Filters.

3.4.4 Linienfilter

Ein weiterer Filter kann realisiert werden, indem man aus dem Scan nach dem Verfahren aus Abschnitt 3.2.1 Linien extrahiert und alle Scanpunkte entfernt, die auf keinem Liniensegment liegen (siehe Abbildung 3.9).

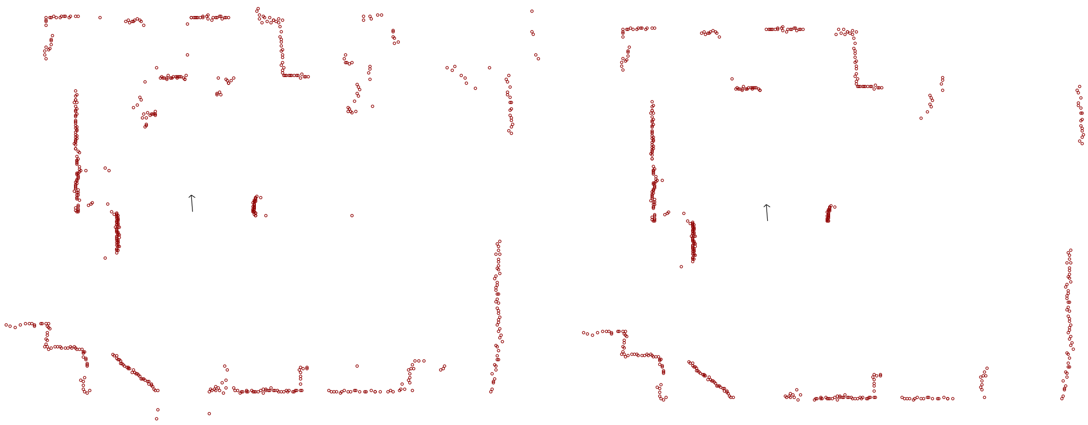


Abbildung 3.9: Anwendung des Linienfilters auf einen Scan. Links vor, rechts nach Anwendung des Filters.

Die Komplexität des Linienfilters ist gleich der Komplexität der Linienextraktion ($O(n \log n)$ im mittleren Fall).

Der Linienfilter findet hauptsächlich dann Verwendung, wenn Algorithmen, die eine polygonale Umwelt erfordern, angewendet werden sollen, z.B. in einem Scan-Matching-Verfahren, welcher jedem Scanpunkt eine Linie aus einem apriori Modell zuordnet. Hier ist es sinnvoll, zuerst den Linienfilter anzuwenden, um falsche Zuordnungen zu vermeiden.

3.4.5 Projektionsfilter

Der Projektionsfilter bestimmt diejenigen Scanpunkte eines Scans, die potentiell einen korrespondierenden Punkt in einem anderen Scan besitzen. Hierzu wird der Scans auf die Aufnahmeposition des anderen Scans mit dem Verfahren aus Abschnitt 3.3 projiziert und Scanpunkte, die durch den anderen Scan verdeckt werden oder nicht in dessen Aufnahmebereich liegen, entfernt.

Folgender Algorithmus beschreibt das Vorgehen des Projektionsfilters:

Algorithmus 3.9: *projection-filter*(s, t)
Eingabe: Scan s, t;
Ausgabe: Scan s'
Ablauf:
  $y := \textit{most-likely-pose}(t)$
  $visible := \textit{scan-project}(s, y)$
  $visible := \textit{mark-occluded}(s, visible, y, t, \textit{MAX-DISTANCE})$
  $visible := \textit{mark-field-of-view}(s, visible, y, t)$
  $s' := \textit{extract-visible}(s, visible)$
  return s'

Hierbei bestimmt *most-likely-pose* die wahrscheinlichste Position, an der der Scan aufgenommen wurde, welche z.B. durch Odometrieinformation gewonnen wird. Diese Position wird als Näherung der Aufnahmeposition von t für die Projektion von s benutzt. Die Funktionen *scan-project* und *mark-occluded* sind die beiden Funktionen aus Abschnitt 3.3, wobei *mark-occluded* hier mit dem Scan t als Rahmenscan aufgerufen wird und der letzte Parameter (Mindestabstand zur Erkennung von Verdeckungen) auf *MAX-DISTANCE* gesetzt wird. Dieser Wert sollte großzügig dimensioniert werden (z.B. $1000mm$), da die Schätzungen der Aufnahmepositionen der beiden Scans ungenau sein können und lieber zu wenige als zuviele Scanpunkte als verdeckt markiert werden sollen. Im nächsten Schritt werden durch die Funktion *mark-field-of-view* alle Scanpunkte als nicht sichtbar markiert, die nicht im Aufnahmebereich des Scans t liegen. Schließlich wird mittels *extract-visible* der gefilterte Scan zusammengesetzt. Hierzu werden einfach alle Scanpunkte von s genommen, für die die *visible*-Markierung wahr ist.

Folgender Algorithmus verwirklicht die Funktion *mark-field-of-view*:

Algorithmus 3.10: *mark-field-of-view*$(s, visible, y, frame)$
Eingabe: Scan s; Marken $visible$; Position y; Scan $frame$
Ausgabe: Markierung der sichtbaren Scanpunkte
Ablauf:
  for $i := 0$ **to** $numpoints(s) - 1$ **do**
    if $visible(i) = TRUE$ **then**
      $visible(i) := FALSE$
      $p := \textit{n-th-scanpoint}(s, i)$

```
        a := angle-to-position(p, y)
        for j := 0 to numpoints(frame) − 1 do
            p := n-th-scanpoint(frame, j)
            a1 := angle-to-position(p, y) − MAX-ANGLE
            a2 := angle-to-position(p, y) + MAX-ANGLE
            if a1 < a and a < a2 then
                visible(i) := TRUE
            endif
        endfor
    endif
endfor
return visible
```

Hier gibt *MAX-ANGLE* die maximale Winkelabweichung an, die für die Zuordnung eines Scanpunktes zu einem Scanpunkt im Rahmenscan erlaubt ist. Da die Scans verdreht sein können, sollte dieser Wert nicht zu klein sein. In der Praxis hat sich ein Wert von $MAX\text{-}ANGLE = 20°$ bewährt.

Die Komplexität des Projektionsfilters ist $O(n^2)$, wenn n die Anzahl Scanpunkte ist. Durch die in Abschnitt 3.3 beschriebene Erweiterung (Zusammenfassen von Scanpunkten in Winkelbereiche) kann die Laufzeit in der Praxis jedoch reduziert werden.

Abbildung 3.10 zeigt die Anwendung des Projektionsfilters auf ein Scanpaar wobei hier der Filter wechselseitig auf die beiden Scans angewandt wurde. Wie zu sehen ist, bleiben im wesentlichen die gemeinsamen Punkte der beiden Scans übrig.

Der Projektionsfilter ist sehr nützlich für die Überdeckung von Scans, da durch die Entfernung nicht sichtbarer Scanpunkte falsche Zuordnungen vermieden werden. Insbesondere werden Zuordnungen von Scanpunkten zu einer Wand, die „von hinten“ gesehen wird, vermieden.

3.5 Überdecken von Scans

In diesem Abschnitt werden Scan-Matching-Verfahren vorgestellt, d.h. Verfahren, die einen Scan mit einem anderen Scan überdecken. Zunächst wird die Problemstellung näher spezifiziert und Eigenschaften der Überdeckung anhand von Beispielen gezeigt. Danach folgen verschiedene Lösungsverfahren, die in der Vergangenheit entwickelt und angewandt wurden. Es handelt sich hierbei um den Algorithmus von Cox, welcher einen Scan mit einem Linienmodell überdeckt und in dieser Arbeit für das Überdecken von Scans erweitert wurde, der IDC-Algorithmus, welcher Scans direkt miteinander überdeckt, und eine Kombination dieser beiden Verfahren, welche Vorteile beider Verfahren nutzt und Nachteile vermeidet. Schließlich folgt ein Überblick über andere in der Literatur zu findende Scan-Matching-Verfahren.

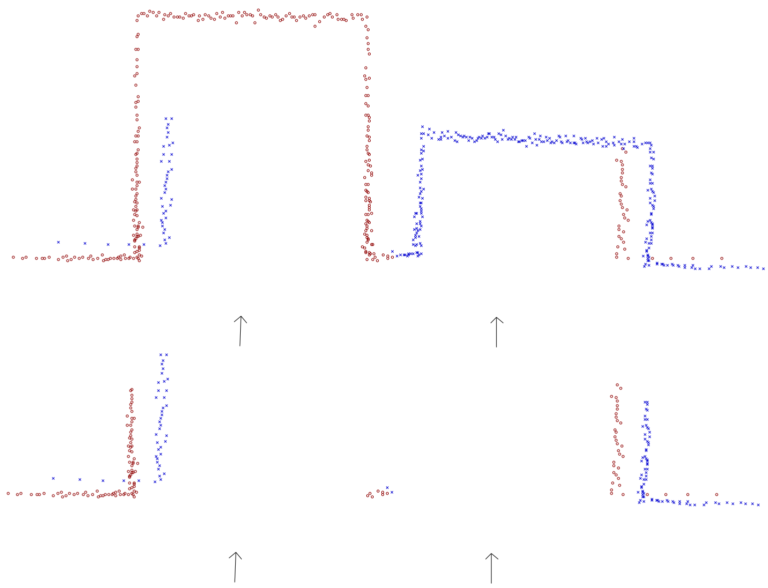


Abbildung 3.10: Wechselseitige Anwendung des Projektionsfilters auf ein Scanpaar. Oben vor, unten nach Anwendung des Filters.

3.5.1 Problemstellung

Gegeben seien zwei Scans s und t. Scan s heiße aktueller Scan, Scan t Referenzscan. Die Aufnahmeposition von t definiere das Koordinatensystem von t. Gesucht ist nun eine Abbildung $match : l \to p$, welche Positionen l im Koordinatensystem von t auf Wahrscheinlichkeiten $p \in [0, 1]$ abbildet, die angeben, wie gut sich Scan s mit Scan t deckt, wenn s an Position l verschoben und verdreht wird.

Scan-Matching ist somit ein Suchproblem in einem mehrdimensionalen Raum. Durch die Bestimmung einer Wahrscheinlichkeitsverteilung für die Scanposition können Mehrdeutigkeiten ausgedrückt werden, z.B. kann es sein, daß es mehrere Positionen gibt, an denen sich die Scans decken.

Die meisten Verfahren zum Überdecken von Scans vereinfachen das Suchproblem, indem folgende zwei Annahmen getroffen werden.

Gauß-Verteilung: Die Verteilungsfunktion, welche durch *match* definiert ist, kann durch eine Gauß'sche Funktion approximiert werden. Dies hat zur Folge, daß nur noch die ersten beiden Momente (Erwartungswert und Varianz) bestimmt werden müssen. Allerdings können dadurch nicht mehr beliebige Verteilungen modelliert werden, was vor allem dann zum Nachteil wird, wenn mehrere Hypothesen vorliegen.

Lokalitätsannahme: Es wird angenommen, daß die ungefähre Aufnahmeposition von s im Koordinatensystem von t bekannt ist (z.B. durch Odometriedaten) und nur noch ein lokaler Abgleich notwendig ist. Dies ermöglicht den Einsatz von lokalen Suchverfahren, die oftmals auch eine Lösung in geschlossener Form besitzen.

Durch diese beiden Annahmen kann Scan-Matching auch als eine Funktion *scan-match* definiert werden, die zwei Scans auf eine Gauß'sche Funktion mit Mittelwert μ_{match} und Kovarianzmatrix Σ_{match} abbildet.

$$\begin{aligned} \textit{scan-match}(s,t) &= (\mu_{match}, \Sigma_{match}) \\ \mu_{match} &= (\hat{x}, \hat{y}, \hat{\theta})^T \\ \Sigma_{match} &= \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{pmatrix} \end{aligned}$$

Der Vektor $(\hat{x}, \hat{y}, \hat{\theta})^T$ gibt die Stelle an, an welche der Scan s im Koordinatensystem von t verschoben und verdreht werden muß, damit eine maximale Überdeckung der Scans zustande kommt. Die Fehlerkomponente Σ_{match} ist ein Maß für die Genauigkeit der berechneten Überdeckung. Die berechnete Wahrscheinlichkeitsfunktion ist somit eine Normalverteilung, welche sich aus berechnetem Mittelwert und Fehlerkovarianzmatrix ergibt:

$$(x, y, \theta)^T \sim N((\hat{x}, \hat{y}, \hat{\theta})^T, \Sigma_{match})$$

3.5.2 Eigenschaften der Gauß'schen Positionsverteilung

Je nach Szenario, in der die beiden Scans aufgenommen werden, ergeben sich verschiedene Ausprägungen der Positionsverteilung. Im folgenden werden anhand von zwei Beispielen die Eigenschaften dieser Gauß-Verteilungen aufgezeigt.

In Abbildung 3.11a wurden zwei Scans in einer Umgebung aufgenommen, in welcher für das Überdecken von Scans alle drei Freiheitsgrade fest sind. Ein Scan wurde im linken unteren Bereich, der andere im rechten oberen Bereich aufgenommen. Die Aufnahmepositionen sind jeweils durch einen kleinen Pfeil gekennzeichnet.

Die durch Scan-Matching berechnete Gauß'sche Positionsverteilung ist im rechten oberen Bereich zu sehen. Diese ist durch eine Ellipse und einen Fehlertrichter an der Stelle des Erwartungswertes dargestellt. Ellipse und Fehlertrichter geben qualitativ die Varianz in Ort und Orientierung an (Höhenlinie mit festem Mahalanobis-Abstand zum Erwartungswert). Da in diesem Beispiel alle Freiheitsgrade fest sind, sind die Fehlerkomponenten recht klein, was einer hohen Genauigkeit der Positionsschätzung entspricht.

Abbildung 3.11b zeigt eine Korridorumgebung, in der Scans aufgenommen wurden. Hier kann die Positionsverteilung jedoch nur die Orientierung und den Ort senkrecht zum Korridor bestimmen. Dies wird in der Fehlerkovarianzmatrix

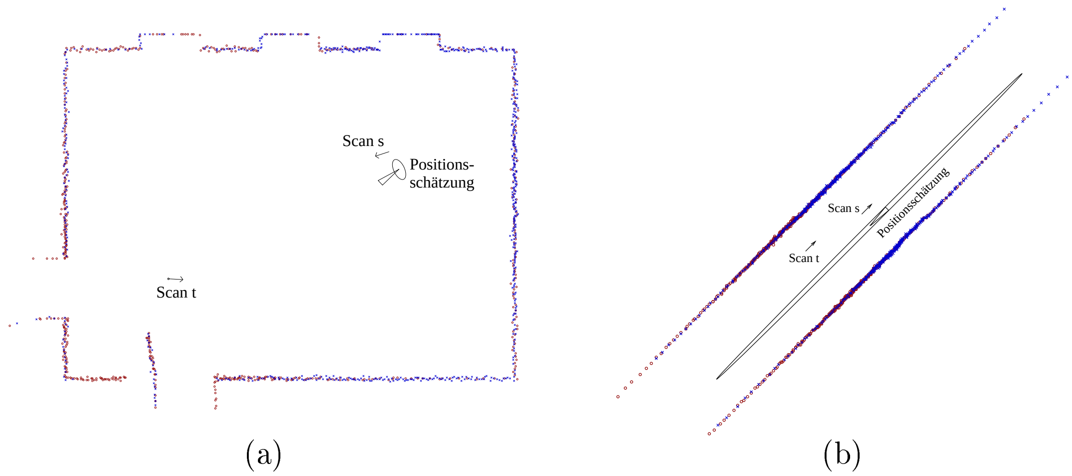


Abbildung 3.11: Eigenschaften der durch Scan-Matching bestimmten Gauß-Verteilungen. Links eine Umgebung, in der alle drei Freiheitsgrade fest sind, rechts eine Korridorumgebung, in der ein Freiheitsgrad frei ist.

dadurch berücksichtigt, daß die Fehlerellipse sich nun in Richtung des Korridors stark ausdehnt.

Eine solche Extremsituation tritt in der Realität nur selten auf. Meistens enthalten die Scans genügend Information, um die Position eindeutig zu bestimmen. Trotzdem enthalten viele Büroumgebungen Situationen, die ähnlich zu der skizzierten Extremsituation sind, z.B. kann ein langer Korridor, von dem nur wenige Türen abgehen, eine ähnliche Fehlerkovarianzmatrix liefern.

Allgemein liefert eine Überdeckung von Scans eine Fehlerellipse, deren Hauptachsen in etwa den Anteilen an horizontaler und vertikaler Positionsinformation entsprechen. Die Fehlerellipse paßt sich also der Positionsinformation der Umgebung an.

Nachdem die Problemstellung des Scan-Matching nun bekannt ist, werden im folgenden Verfahren zur Lösung dieses Problems vorgestellt.

3.6 Erweiterter Cox-Algorithmus

Ein Verfahren zum Überdecken eines Scans mit einem Linienmodell wurde von Cox [1990; 1991] entwickelt. Hierzu werden Scanpunkte Modellinien zugeordnet und daraus eine Positionskorrektur berechnet.

Zunächst wird das ursprüngliche Verfahren von Cox vorgestellt. Danach werden eigene Erweiterungen präsentiert, die den Ansatz für die paarweise Überdeckung von Scans einsatzfähig machen.

3.6.1 Verfahren von Cox

Das ursprüngliche Verfahren überdeckt einen Scan mit einem apriori Linienmodell. Hierzu wird jedem Scanpunkt eine Linie des apriori Modells zugeordnet. Aus dieser Zuordnung läßt sich dann die Verschiebung und Verdrehung des Scans gegenüber dem Linienmodell bestimmen. Das Verfahren benötigt eine ungefähre Anfangsschätzung der Aufnahmeposition, welche aus Odometriedaten gewonnen wird.

Das Verfahren läßt sich in folgende Schritte einteilen:

1. Setze $(\hat{x}, \hat{y}, \hat{\theta})^T = (s_x, s_y, s_\theta)^T$. Hierbei ist $(s_x, s_y, s_\theta)^T$ die initiale Positionsschätzung der Scanaufnahme, welche durch Odometriedaten gegeben ist.

2. Verschiebe und drehe Scan auf Position $(\hat{x}, \hat{y}, \hat{\theta})^T$.

3. Bestimme für jeden Scanpunkt die Modellinie, die dem Punkt am nächsten liegt. Diese Modellinie wird im folgenden Ziellinie genannt.

4. Berechne die Transformation $\hat{b} = (\Delta x, \Delta y, \Delta\theta)^T$, welche die Summe der Abstandsquadrate zwischen Scanpunkten und jeweiliger Ziellinie minimiert.

5. Setze $(\hat{x}, \hat{y}, \hat{\theta})^T = (\hat{x}, \hat{y}, \hat{\theta})^T + (\Delta x, \Delta y, \Delta\theta)^T$.

6. Wiederhole Schritte 2 – 5 bis das Verfahren konvergiert. Das Ergebnis der Überdeckung ist $(\hat{x}, \hat{y}, \hat{\theta})^T$.

7. Berechne Fehlerkovarianzmatrix Σ_{match}.

Bestimmung der Ziellinie

Bei der Bestimmung der Ziellinie für einen Scanpunkt wird diejenige Linie ausgewählt, die den kleinsten (euklidischen) Abstand zum Scanpunkt besitzt. Falls dieser Abstand einen vorher festgelegten maximalen Abstand d_{max} übersteigt, findet keine Zuordnung statt. Der Scanpunkt wird in diesem Fall als Ausreißer behandelt und für die weiteren Schritte entfernt. In der Praxis haben sich Werte zwischen $1000mm$ – $2000mm$ für d_{max} bewährt.

Bestimmung der Transformation

Zur Berechnung der Transformation in Schritt 4 wird eine Funktion $trans$ aufgestellt, die den Scan um einen Winkel $\Delta\theta$ dreht und mit einem Vektor $\Delta t = (\Delta x, \Delta y)^T$ verschiebt. Gedreht wird dabei immer um den Ort $l_s = (s_x, s_y)^T$ der Scanaufnahme. Die Funktion $trans$ bildet einen Scanpunkt $p_i = (p_{ix}, p_{iy})^T$ wie folgt ab:

$$trans(\Delta t, \Delta\theta)(p_i) = \begin{pmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{pmatrix} (p_i - l_s) + l_s + \Delta t \quad (3.2)$$

Unter der Annahme, daß der Verdrehungswinkel $\Delta\theta$ klein ist, läßt sich die Funktion approximieren zu:

$$trans(\Delta t, \Delta\theta)(p_i) \approx \begin{pmatrix} 1 & -\Delta\theta \\ \Delta\theta & 1 \end{pmatrix} (p_i - l_s) + l_s + \Delta t \tag{3.3}$$

Zu jedem Scanpunkt sei die Ziellinie durch die Parameter $u_i = (u_{ix}, u_{iy})^T$ und r_i gegeben, so daß für alle Punkte z auf der Linie gilt: $z^T u_i = r_i$. Hier wird zur Vereinfachung angenommen, daß die Linie unendlich lang ist (bei der Bestimmung der Ziellinie in Schritt 1 wird aber von den endlichen Liniensegmenten ausgegangen). Der quadrierte Abstand eines Scanpunktes p_i zur jeweiligen Ziellinie läßt sich dann wie folgt berechnen:

$$\begin{aligned}
dist^2 &= ((trans(\Delta t, \Delta\theta)(p_i))^T u_i - r_i)^2 \\
&\approx ((\begin{pmatrix} 1 & -\Delta\theta \\ \Delta\theta & 1 \end{pmatrix} (p_i - l_s) + l_s + \Delta t)^T u_i - r_i)^2 \\
&= ((\Delta t + \Delta\theta \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (p_i - l_s) + p_i)^T u_i - r_i)^2 \\
&= ((\Delta x - \Delta\theta(p_{iy} - s_y) + p_{ix})u_{ix} + \\
&\quad (\Delta y + \Delta\theta(p_{ix} - s_x) + p_{iy})u_{iy} - r_i)^2 \\
&= ((x_{i1}, x_{i2}, x_{i3})b - y_i)^2 \qquad (3.4)
\end{aligned}$$

mit

$$x_{i1} = u_{ix} \tag{3.5}$$
$$x_{i2} = u_{iy} \tag{3.6}$$
$$x_{i3} = -(p_{iy} - s_y)u_{ix} + (p_{ix} - s_x)u_{iy} \tag{3.7}$$
$$y_i = r_i - p_{ix}u_{ix} - p_{iy}u_{iy} \tag{3.8}$$
$$b = (\Delta x, \Delta y, \Delta\theta)^T \tag{3.9}$$

Die Summe der Abstandsquadrate E_{fit} zwischen Scanpunkten und jeweiliger Ziellinie für n Scanpunkte $p_1, \ldots, p_n$ ergibt sich dann zu:

$$\begin{aligned}
E_{fit}(b) &= \sum_{i=1}^{n} ((x_{i1}, x_{i2}, x_{i3})b - y_i)^2 \\
&= (Xb - Y)^T (Xb - Y) \qquad (3.10)
\end{aligned}$$

mit

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{n3} \end{pmatrix} \tag{3.11}$$

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{3.12}$$

Die Funktion E_{fit} muß nun minimiert und die zugehörige Transformation $\hat{b}$ bestimmt werden. Dazu differenziert man E_{fit} nach b und setzt den entstehenden Ausdruck gleich Null:

$$\begin{aligned}
\frac{\partial E_{fit}(\hat{b})}{\partial b} &= 0 \\
2X^T(X\hat{b} - Y) &= 0 \\
X^T X\hat{b} &= X^T Y \\
\hat{b} &= (X^T X)^{-1} X^T Y
\end{aligned} \tag{3.13}$$

Der Vektor $\hat{b}$ ist genau die in Schritt 4 gesuchte Transformation.

Konvergenz und Fehlerkovarianzmatrix

Die Schritte 2 – 5 des Cox-Algorithmus werden solange wiederholt, bis das Verfahren konvergiert. Für die Konvergenz betrachtet man den berechneten Transformationsvektor $\hat{b}$ und prüft, ob er klein genug ist. Hierzu müssen die folgenden beiden Bedingungen gelten:

$$\sqrt{\Delta x^2 + \Delta y^2} < \epsilon_{dist} \tag{3.14}$$

$$|\Delta\theta| < \epsilon_\theta \tag{3.15}$$

Die Werte von ϵ_{dist} und ϵ_θ hängen vom verwendeten Laserscanner ab und müssen geeignet gewählt werden. Für kommerziell erhältliche Laserscanner liefern Werte von $1mm - 10mm$ für ϵ_{dist} und $0.1° - 1.0°$ für ϵ_θ gute Ergebnisse.

Zusätzlich zum Ergebnisvektor $(\hat{x}, \hat{y}, \hat{\theta})^T$ kann mit dem Verfahren von Cox auch eine Fehlerkovarianzmatrix Σ_{match} berechnet werden, welche die Genauigkeit der geschätzten Transformation angibt. Diese berechnet sich zu [Cox, 1990; Cox, 1991]:

$$\Sigma_{match} = \frac{1}{n-4}(Y - X\hat{b})^T(Y - X\hat{b})(X^T X)^{-1} \tag{3.16}$$

Zeit-Komplexität

Für die Betrachtung der Zeitkomplexität seien n Scanpunkte und m Modellinien gegeben und das Verfahren benötige k Iterationsschritte. Zur Berechnung der Komplexität werden zunächst die einzelnen Schritte untersucht und dann zur Gesamtkomplexität zusammengesetzt.

Schritt 2 verschiebt und dreht den Scan. Dies benötigt $O(n)$ viele Operationen. In Schritt 3 wird jedem Scanpunkt eine Modellinie zugeordnet. Ein einfacher Algorithmus benötigt hierzu $O(nm)$ viele Schritte (die Komplexität kann aber durch den Einsatz von Voronoi-Diagrammen verbessert werden). Die Transformationsberechnung in Schritt 4 kann nach obigem Lösungsansatz mit Aufwand $O(n)$ berechnet werden. Ebenso benötigt die Berechnung der Fehlerkovarianzmatrix $O(n)$ Zeit. Zusammen ergibt sich also eine Komplexität von $O(nmk)$.

3.6.2 Erweiterung des Cox Algorithmus

Eine naheliegende Erweiterung des Cox-Algorithmus für den Einsatz der paarweisen Überdeckung von Scans ist, aus dem Referenzscan Linien zu extrahieren und diese als apriori Modell für den Cox-Algorithmus zu verwenden.

Abbildung 3.12 zeigt den Aufbau des erweiterten Cox-Algorithmus. Aus dem Referenzscan wird ein Linienmodell gewonnen und dem Cox-Algorithmus als apriori Modell weitergereicht. Der aktuelle Scan wird vor Anwendung des Cox-Algorithmus noch mit einem Linienfilter behandelt. Auf diese Weise werden falsche Zuordnungen von Scanpunkten zu Modellinien minimiert. Als Ergebnis erhält man neben Erwartungswert und Kovarianzmatrix der Scanüberdeckung auch einen Fehlerwert $error$, welcher die Qualität der berechneten Überdeckung anzeigt. Dieser Wert berechnet sich als der Median aller Abstände der Zuweisungen von Scanpunkten zu Modellinien nach der Überdeckung. Ein hoher Wert für $error$ bedeutet also eine schlechte Überdeckung, da die Abstände von Scanpunkten zu Modellinien groß sind, ein kleiner Wert bedeutet eine gute Überdeckung.

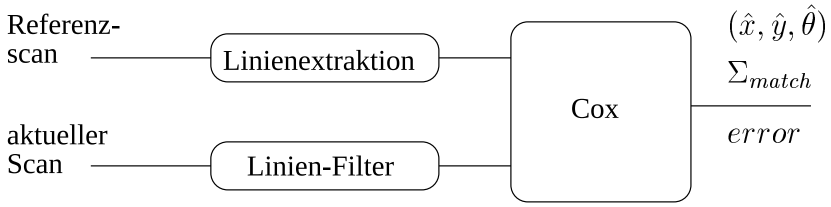


Abbildung 3.12: Schema des erweiterten Cox-Algorithmus.

Um die Anzahl falscher Zuordnungen zu reduzieren, wurde des Cox-Verfahren selbst noch um zwei Heuristiken erweitert. Zum einen wurde neben der fest implementierten Grenze d_{max} zur Entfernung von Ausreißern zusätzlich noch eine variable Schranke entwickelt. Hierzu werden alle Zuordnungen von Scanpunkten zu Modellinien nach ihrem Abstand aufsteigend sortiert und nur die ersten $q \cdot n$ viele Zuordnungen ($q \in [0, 1]$, $n =$ Anzahl Zuordnungen) zur Bestimmung der Transformation und der Fehlerkovarianzmatrix benutzt. Hierbei wird q je nach Situation gewählt. Es werden mindestens q_0 viele Zuordnungen verwendet. Falls weitere Zuordnungen mit geringem Abstand ($< e_{max}$) vorhanden sind, so werden diese ebenfalls hinzugenommen. Diese Methode paßt sich der Situation der beiden Scans an. Hat sich die Umgebung zwischen den beiden Scanaufnahmen stark geändert, so wird ein kleines q gewählt, d.h. es werden mehr Ausreißer entfernt. Hat sich die Umgebung nicht geändert, so liegt q nahe 1 und es werden nahezu alle Scanpunkte für die Bestimmung der Überdeckung benutzt. In der implementierten Version wurde $d_{max} = 2000mm$, $e_{max} = 500mm$ und $q_0 = 0.66$ gesetzt, und e_{max} nimmt mit jeder Iteration exponentiell ab.

Die zweite Heuristik, um Fehlzuordnungen zu vermeiden, ist so realisiert, daß jedem Scanpunkt des aktuellen Scans eine Orientierung zugeordnet wird. Die

Orientierung wird im Linienfilter bestimmt, welcher vor Anwendung des Cox-Algorithmus angewandt wird, und ergibt sich als die Orientierung des Liniensegmentes, auf welcher der Scanpunkt liegt. Bei der Bestimmung der Ziellinie für einen Scanpunkt wird nun nicht mehr die Linie mit kleinstem euklidischen Abstand ausgewählt, sondern die, welche die folgende Abstandsfunktion $dist_\alpha$ minimiert:

$$dist_\alpha(p,l) = \frac{dist(p,l)}{abs(\cos(heading(p) - heading(l))} \tag{3.17}$$

wobei $dist(p,l)$ die euklidische Distanz von Scanpunkt p zu Linie l ist und *heading* die Orientierung von Scanpunkt und Linie wiedergibt. Durch diese Heuristik wird die Zuordnung von Scanpunkten zu Linien vermieden, die eine große Abweichung in der Orientierung besitzen. Dies kann von Vorteil sein, wenn der aktuelle Scan gegenüber dem Referenzmodell nur wenig verdreht ist, es aber Veränderungen in der Umgebung gibt, z.B. in Abbildung 3.13 besitzen die durch eine Ellipse gekennzeichneten Scanpunkte eine kleine euklidische Distanz zur unteren Linie, jedoch einen großen $dist_\alpha$-Wert zu allen Linien. Überdeckt man den Scan mit dem hier beschriebenen Verfahren, so werden die Scanpunkte innerhalb der Ellipse als Ausreißer erkannt und entfernt.

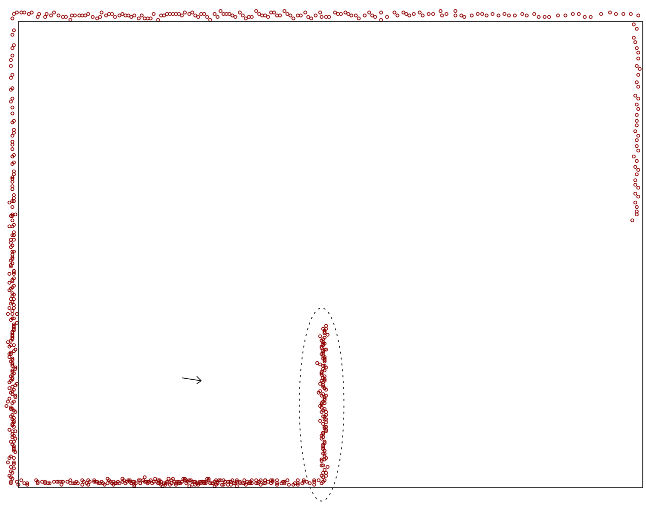


Abbildung 3.13: Heuristik der modifizierten Abstandsfunktion. Die Scanpunkte innerhalb der Ellipse werden keiner Modellinie zugeordnet.

Wie in den Arbeiten [Gutmann und Schlegel, 1996; Gutmann, 1996] gezeigt wurde, eignet sich der erweiterte Cox-Algorithmus sehr gut für Umgebungen mit hohem polygonalen Anteil. Je weniger jedoch die Umgebung polygonal ist desto schlechter werden die Ergebnisse. Ein Scan-Matching-Algorithmus, der keine polygonale Umgebung erfordert, wird im nächsten Kapitel beschrieben.

3.7 IDC-Algorithmus

Ein leistungsstarker Algorithmus zur paarweisen Überdeckung von Scans, der ohne geometrische Interpretation der beiden Scans auskommt, wurde von Lu und Milios entwickelt [Lu und Milios, 1994a; Lu und Milios, 1994b; Lu und Milios, 1994c; Lu, 1995; Lu und Milios, 1997b]. Insbesondere setzt der Algorithmus keinerlei Merkmale in den Scandaten voraus und ist somit universell in vielen Umgebungen einsetzbar.

3.7.1 Idee

Die Idee des Verfahrens ist, Scanpunkte des aktuellen Scans den Punkten des Referenzscans zuzuordnen. Durch diese Zuordnung kann dann ähnlich wie beim Cox-Algorithmus eine Fehlersumme minimiert und die Verschiebung und Verdrehung der Scans bestimmt werden. Für die Zuordnung von Scanpunkten werden zwei Heuristiken benutzt: die Regel *nächster-Punkt* (*closest-point rule*) und die Regel *gleiche-Entfernung* (*matching-range rule*).

3.7.2 Regel *nächster-Punkt*

Für jeden Scanpunkt des aktuellen Scans wird der Punkt im Referenzscan bestimmt, der dem Scanpunkt am nächsten liegt. Dabei wird zwischen den Scanpunkten des Referenzscans interpoliert, d.h. man verbindet die Scanpunkte mit kurzen Liniensegmenten. Ein Partner muß daher nicht genau ein Scanpunkt des Referenzscans sein, sondern kann ein Punkt auf der Verbindungslinie zweier benachbarter Scanpunkte sein.

Der Algorithmus zur Bestimmung des Zuordnungspartners zu einem Scanpunkt p des aktuellen Scans ist sehr einfach. Man betrachtet dazu immer zwei aufeinanderfolgende Scanpunkte p_1, p_2 des Referenzscans und berechnet den Abstand von p zum Liniensegment (p_1, p_2). Das Liniensegment mit dem kleinsten Abstand enthält dann den gesuchten Punkt. Unter der Annahme, daß die Scans nur wenig verdreht sind, kann der Suchbereich nach dem am nächsten liegenden Punkt erheblich eingeschränkt werden. Dazu werden nur Scanpunkte p_i des Referenzscans betrachtet, deren Aufnahmewinkel bezüglich der Aufnahmeposition des aktuellen Scans in einem Bereich um den Aufnahmewinkel von p liegen. Mathematisch heißt das, daß der Absolutwert des Winkels zwischen den Geraden (l_s, p) und (l_s, p_i) kleiner einer Winkeltoleranz ω sein muß, wobei l_s die Aufnahmeposition des aktuellen Scans ist.

Für eine Implementierung ist es von Vorteil, wenn die Scanpunkte des Referenzscans mit aufsteigendem Aufnahmewinkel (bezüglich l_s) vorliegen, da die Ausnutzung dieser Eigenschaft eine effiziente Implementierung der Suchbereichseinschränkung erlaubt. Daher wird im folgenden von dieser Eigenschaft ausgegangen. Dies bedeutet, daß vor Anwendung des hier beschriebenen Überdeckungsverfahrens auch dafür gesorgt werden muß, daß die Punkte mit aufsteigendem Winkel vorliegen, was z.B. durch Anwenden des Projektionsfilters erreicht werden kann.

3.7.3 Regel *gleiche-Entfernung*

Bei der zweiten Heuristik wird jedem Scanpunkt des aktuellen Scans der Punkt im Referenzscan zugeordnet, der die gleiche Entfernung zum Aufnahmepunkt l_s besitzt. Dabei wird wieder zwischen den Scanpunkten des Referenzscans interpoliert. Falls kein Punkt gefunden wurde, der exakt die gleiche Entfernung besitzt, so wird derjenige Punkt gewählt, dessen Entfernung am nächsten zu der gesuchten Entfernung liegt.

Der Algorithmus zur Bestimmung des Zuordnungspartners für einen Scanpunkt p des aktuellen Scans gestaltet sich ähnlich dem Verfahren der Regel *nächster-Punkt*. Auch hier werden nur Punkte p_i des Referenzscans untersucht, die in dem oben beschriebenen Winkelintervall liegen. Auf diese Weise werden außerdem völlig falsche Zuordnungen vermieden. Die Bestimmung des Zuordnungspartners läuft wie folgt ab:

Algorithmus 3.11: *find-matching-range*(p, p_i)
Eingabe: Scanpunkt p; Scanpunkte $p_i, i \in \{1, \dots n\}$
Ausgabe: Punkt p'
Ablauf:

```
d := distance(p, l_s)
d' := ∞
d_pp' := ∞
for i := 1 to n − 1 do
    d_1 := distance(p_i, l_s)
    d_2 := distance(p_{i+1}, l_s)
    if (d_1 < d and d_2 < d) or (d_1 > d and d_2 > d) then
        if |d_1 − d| < |d_2 − d| then
            d_h := |d_1 − d|
            p_h := p_i
        else
            d_h := |d_2 − d|
            p_h := p_{i+1}
        endif
        if d_h < d' then
            d' := d_h
            p' := p_h
        endif
    else
        p_h := interpolate(p_i, p_{i+1}, d)
```

```
        if d' > 0 or distance(p, p_h) < d_pp' then
            d' := 0
            d_pp' := distance(p, p_h)
            p' := p_h
        endif
    endif
endfor
return p'
```

Der Algorithmus untersucht alle $n-1$ Paare von aufeinanderfolgenden Scanpunkten im zulässigen Winkelintervall. Für jedes Paar werden die Entfernungen d_1 und d_2 zum Aufnahmepunkt l_s berechnet. Liegen beide Entfernungen über oder unter der gesuchten Entfernung d, so kann mit diesem Paar kein Punkt der gleichen Entfernung gefunden werden. Sofern noch kein besserer Punkt gefunden wurde, wird in diesem Fall derjenige Punkt von p_i und p_{i+1} als vorläufiger Zuordnungspartner gewählt, dessen Abstand näher am gesuchten Abstand liegt.

Falls die Entfernungen der beiden Scanpunkte jedoch einmal so liegen, daß ein Wert kleiner und der andere größer als der gesuchte Wert ist, so kann mit der Funktion *interpolate* ein Punkt zwischen p_i und p_{i+1} interpoliert werden, der genau die gesuchte Entfernung besitzt. Möglicherweise gibt es aber mehrere Paare (p_i, p_{i+1}) mit dieser Entfernungsverteilung. In diesem Fall gibt es also keinen eindeutigen Korrespondenzpartner. Man kann dann entweder gar keinen Korrespondenzpartner zulassen oder aber, wie es in dieser Implementierung vorgenommen wurde, den Punkt benutzen, der am nächsten zum Scanpunkt p liegt.

3.7.4 Bestimmung der Verdrehung und Verschiebung

Nachdem man eine Menge von Korrespondenzpaaren $(p_i, p'_i), i = 1 \dots n$ nach einer der oben beschriebenen Regeln bestimmt hat, kann daraus durch Minimieren einer Summe von Abstandsquadraten die Verdrehung und Verschiebung der Punkte berechnet werden. Dazu werden die Punkte p_i des aktuellen Scans um $\Delta\theta$ gedreht und mit $\Delta t = (\Delta x, \Delta y)^T$ verschoben. Gedreht wird wieder um die Aufnahmeposition l_s des Scans. Nun wird die Summe der quadrierten Abstände zwischen den gedrehten und verschobenen Punkten p_i und p'_i aufgestellt:

$$E_{fit}(\Delta t, \Delta\theta) = \sum_{i=1}^{n} |R_2(\Delta\theta)(p_i - l_s) + l_s + \Delta t - p'_i|^2 \tag{3.18}$$

Diese Funktion muß nun minimiert werden. Die hierzu passenden Parameter Δt und $\Delta\theta$ berechnen sich zu [Lu, 1995, Seiten 50,51]:

$$\Delta\theta = \tan^{-1} \frac{S_{xy'} - S_{yx'}}{S_{xx'} + S_{yy'}} \tag{3.19}$$

$$\Delta t = \bar{p}' - l_s - R_2(\Delta\theta)(\bar{p} - l_s) \tag{3.20}$$

mit

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i \tag{3.21}$$

$$\bar{p}' = \frac{1}{n}\sum_{i=1}^{n} p_i' \tag{3.22}$$

$$\begin{pmatrix} S_{xx'} & S_{xy'} \\ S_{yx'} & S_{yy'} \end{pmatrix} = \sum_{i=1}^{n} (p_i - \bar{p})(p_i' - \bar{p}')^T \tag{3.23}$$

Um Ausreißer bei der Berechnung der Verdrehung und Verschiebung zu eliminieren, wird vorher die Menge der Punktepaare nach ihrem Abstand aufsteigend sortiert. Anschließend werden nur die ersten $q \cdot n$ vielen Zuordnungen ($q \in [0, 1]$, n = Anzahl Zuordnungen) zur Bestimmung der Transformation benutzt. Wie beim erweiterten Cox-Algorithmus wird q nicht konstant gesetzt sondern je nach Situation gewählt. Dies ermöglicht Werte von q nahe 1, wenn die beiden Scans gut zueinander passen und Werte nahe q_0, falls viele Ausreißer vorliegen.

In [Lu, 1995] wurden beide Heuristiken zum Überdecken von Scans untersucht. Dabei stellte sich heraus, daß die Regel *nächster-Punkt* besonders gut zur Bestimmung der Verschiebung und die Regel *gleiche-Entfernung* besonders gut zur Bestimmung der Verdrehung geeignet ist. Daher wurden beide Verfahren kombiniert und der IDC-Algorithmus entworfen.

3.7.5 IDC-Algorithmus

Der IDC[1]-Algorithmus kann wie folgt formuliert werden:

1. Setze $(\hat{x}, \hat{y}, \hat{\theta})^T = (s_x, s_y, s_\theta)^T$. Hierbei ist $(s_x, s_y, s_\theta)^T$ die initiale Schätzung der Aufnahmeposition des aktuellen Scans, welche durch Odometriedaten gegeben ist.

2. Verschiebe und drehe aktuellen Scan auf Position $(\hat{x}, \hat{y}, \hat{\theta})^T$.

3. Bestimme für jeden Scanpunkt p_i den Korrespondenzpartner p_i' nach der Regel *nächster-Punkt* und p_i'' nach der Regel *gleiche-Entfernung*.

4. Berechne $(\Delta x_1, \Delta y_1, \Delta\theta_1)^T$ als Lösung des Minimums der Summe von Abstandsquadraten zwischen den Punktepaaren (p_i, p_i').

5. Berechne $(\Delta x_2, \Delta y_2, \Delta\theta_2)$ als Lösung des Minimums der Summe von Abstandsquadraten zwischen den Punktepaaren (p_i, p_i'').

6. Setze $(\hat{x}, \hat{y}, \hat{\theta})^T = (\hat{x}, \hat{y}, \hat{\theta})^T + (\Delta x_1, \Delta y_1, \Delta\theta_2)^T$.

[1] iterative dual correspondence

7. Wiederhole Schritte 2 – 6 bis das Verfahren konvergiert. Das Ergebnis der Überdeckung ist $(\hat{x}, \hat{y}, \hat{\theta})^T$.

8. Berechne Fehlerkovarianzmatrix Σ_{match}.

Der IDC-Algorithmus benutzt also das Translationsergebnis der Regel *nächster-Punkt* zusammen mit dem Rotationsergebnis der Regel *gleiche-Entfernung* als Gesamttransformation.

Für die Konvergenz betrachtet man wie beim Cox-Algorithmus den berechneten Transformationsvektor $(\Delta x_1, \Delta y_1, \Delta\theta)^T$ und prüft, ob er klein genug ist. Hierzu müssen die beiden folgenden Bedingungen gelten:

$$\sqrt{\Delta x_1^2 + \Delta y_1^2} < \epsilon_{dist} \tag{3.24}$$

$$|\Delta\theta_2| < \epsilon_\theta \tag{3.25}$$

Die Werte von ϵ_{dist} und ϵ_θ hängen vom verwendeten Laserscanner ab und müssen geeignet gewählt werden. Für kommerziell erhältliche Laserscanner liefern Werte von $1mm - 10mm$ für ϵ_{dist} und $0.1° - 1.0°$ für ϵ_θ gute Ergebnisse.

3.7.6 Fehlerkovarianzmatrix

Die Fehlerkovarianzmatrix Σ_{match} berechnet sich aus den Korrespondenzpaaren $(p_i, p_i'), i = 1 \ldots n$ wie folgt [Lu, 1995, Seiten 92-96]:

$$\Sigma_{match} = s^2(M^T M)^{-1} \tag{3.26}$$

mit

$$M = \begin{pmatrix} M_1 \\ \vdots \\ M_n \end{pmatrix} \tag{3.27}$$

$$M_i = \begin{pmatrix} 1 & 0 & -y_i \\ 0 & 1 & x_i \end{pmatrix} \tag{3.28}$$

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \frac{1}{2}(p_i + p_i') \tag{3.29}$$

$$s^2 = \frac{(Z - M\bar{D})^T(Z - M\bar{D})}{2n - 3} \tag{3.30}$$

$$Z = \begin{pmatrix} p_1 - p_1' \\ \vdots \\ p_n - p_n' \end{pmatrix} \tag{3.31}$$

$$\bar{D} = (M^T M)^{-1} M^T Z \tag{3.32}$$

Aufgrund der gleichzeitigen Anwendung beider Heuristiken ergeben sich zwei Mengen von Korrespondenzpaaren. Nachdem das Verfahren konvergiert ist, ist jedoch kein großer Unterschied mehr zwischen den beiden Mengen von Korrespondenzpaaren vorhanden. Daher kann eine beliebige dieser beiden Korrespondenzmengen für die Berechnung der Fehlerkovarianzmatrix benutzt werden.

3.7.7 Zeit-Komplexität

Der zeitaufwendigste Teil des IDC-Algorithmus besteht in der Bestimmung der Korrespondenzpartner. Hier muß für jeden Scanpunkt des aktuellen Scans ein entsprechender Punkt im Referenzscan bestimmt werden. Somit ergibt sich in erster Näherung ein Aufwand von $O(n^2)$, wenn n die Anzahl der Scanpunkte ist. Durch die Festlegung des zulässigen Winkelbereichs kann die Anzahl der Korrespondenzpartner erheblich eingeschränkt werden. Allerdings ist die Anzahl der Scanpunkte innerhalb dieses Intervalls immer noch von n abhängig, so daß der Aufwand zur Bestimmung aller Korrespondenzpartner unverändert bleibt. Insgesamt ergibt sich also bei k Iterationen des IDC-Algorithmus ein Aufwand von $O(kn^2)$.

3.7.8 Erweiterter IDC-Algorithmus

Eine naheliegende Optimierung des IDC-Algorithmus ist beide Scans vorher mit dem Reduktionsfilter zu bearbeiten. Auf diese Weise wird die Anzahl der Scanpunkte stark reduziert ohne dabei wesentliche Information zu verlieren. Weiterhin kann wie beim Cox-Algorithmus ein Fehlerwert *error* bestimmt werden, der sich als Median der Abstände der Scanpunktzuordnungen nach der Überdeckung ergibt. Dieser Wert ist klein, wenn sich die beiden Scans gut decken, und groß, wenn die Überdeckung schlecht ist. Abbildung 3.14 zeigt den schematischen Aufbau des erweiterten IDC-Algorithmus.

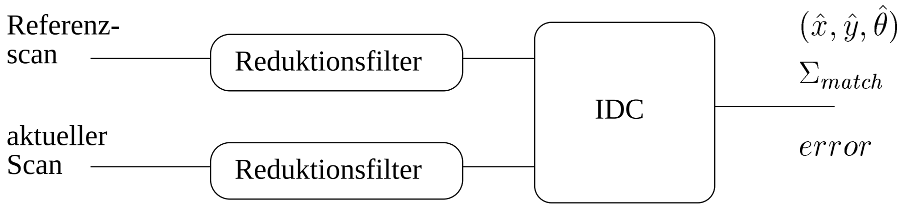


Abbildung 3.14: Schema des erweiterten IDC-Algorithmus.

Der IDC-Algorithmus eignet sich wegen seiner universellen Einsetzbarkeit sehr gut für viele Umgebungen und setzt keine geometrischen Annahmen an die Umgebung voraus. Jedoch kann der IDC-Algorithmus in Korridorumgebungen eine falsche oder zu optimistische Positionsschätzung liefern. Im nächsten Abschnitt wird daher ein weiteres Scan-Matching-Verfahren vorgestellt, das die Vorteile von Cox- und IDC-Algorithmus kombiniert und Nachteile vermeidet.

3.8 Kombinierter Scan-Matching-Algorithmus

Die beiden in den vorigen Abschnitten vorgestellten Scan-Matching-Verfahren wurden in einer Arbeit von Gutmann und Schlegel experimentell miteinander verglichen [Gutmann und Schlegel, 1996; Gutmann, 1996]. Dabei stellte sich heraus, daß beide Algorithmen für polygonale Umgebungen, in denen für die Überdeckung alle Freiheitsgrade fest sind, sehr genaue Resultate liefern und im Vergleich keiner der beiden Algorithmen signifikant bessere Ergebnisse liefert. Der Cox-Algorithmus benötigt aber wegen der geringeren Komplexität eine geringere Laufzeit als der IDC-Algorithmus.

In nicht-polygonalen Umgebungen liefert der IDC-Algorithmus erwartungsgemäß bessere Resultate als der Cox-Algorithmus. Jedoch ist das IDC-Verfahren in einer polygonalen Umgebung, die nicht alle Freiheitsgrade fixiert, also z.B. eine Umgebung mit langem Korridor, zu optimistisch. Hier wird eine Positionsverteilung berechnet, die den aktuellen Scan fest an einen Ort bindet und nicht wie man erwarten würde eine Positionsverteilung, die eine lange Ellipse entlang des Korridors beschreibt.

Aufgrund dieser Resultate wurde in [Gutmann und Schlegel, 1996; Gutmann, 1996] ein kombiniertes Scan-Matching-Verfahren (CSM) vorgeschlagen, das in polygonalen Umgebungen die bessere Laufzeit des Cox-Algorithmus ausnutzt und zu optimistische Positionsschätzungen des IDC-Algorithmus verhindert, und in nicht-polygonalen Umgebungen die Universalität des IDC-Algorithmus verwendet.

Abbildung 3.15 zeigt den schematischen Aufbau dieses kombinierten Scan-Matching-Verfahrens. Kern des Verfahrens ist eine Entscheidungslogik, welche die zu überdeckenden Scans untersucht und dann einen der beiden Verfahren Cox-Scan-Matching oder IDC-Scan-Matching verwendet, um die Überdeckung zu berechnen. Für die Entscheidungslogik werden aus beiden Scans jeweils Liniensegmente extrahiert und der prozentuale Anteil der Liniensegmente am Gesamtumfang der Scans berechnet. Ist dieser Anteil für beide Scans genügend groß, so liegt eine hauptsächlich polygonale Umgebung vor und der erweiterte Cox-Algorithmus wird zum Überdecken der Scans verwendet. Andernfalls wird der erweiterte IDC-Algorithmus benutzt. In der Praxis haben sich Werte zwischen 40 und 60 Prozent für den prozentualen Anteil bewährt.

3.9 Andere Scan-Matching-Verfahren

In der Literatur befinden sich eine ganze Reihe weiterer Scan-Matching-Verfahren. Dieser Abschnitt stellt ein paar dieser Verfahren vor und ordnet sie gegenüber den bereits beschriebenen Methoden ein.

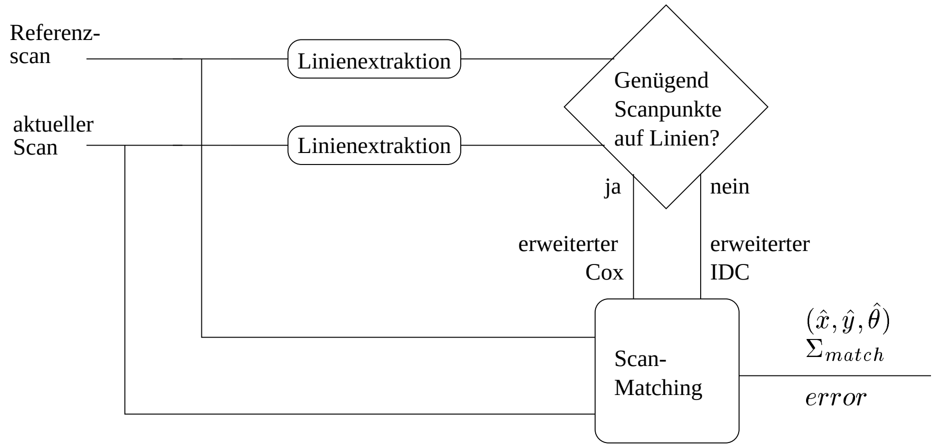


Abbildung 3.15: Schema des kombinierten Scan-Matching-Verfahrens CSM.

3.9.1 Kreuzkorrelations-Algorithmus

Ein histogrammbasiertes Verfahren zur Überdeckung zweier Scans wurde von Weiß *et al.* entwickelt [Weiß *et al.*, 1994a; Weiß *et al.*, 1994b; Weiß und Puttkamer, 1995]. Voraussetzung ist, daß in den Scans genügend viele gerade Linien vorhanden sind, d.h. die Umgebung polygonal ist, und es zwei senkrecht zueinander stehende Hauptrichtungen gibt.

Um die beiden Scans zur Überdeckung zu bringen, wird zunächst die Verdrehung der Scans bestimmt. Dazu wird von jedem Scan ein sogenanntes Winkelhistogramm angelegt. Ein Winkelhistogramm ist eine Statistik über die Verteilung der in dem Scan auftretenden Winkel zwischen aufeinanderfolgenden Scanpunkten. Zu jedem Scanpunkt betrachtet man seinen Ortsvektor bezüglich der Aufnahmeposition des Scans und bildet die Differenz aufeinanderfolgender Ortsvektoren. Der Winkel dieses Differenzvektors gegenüber der x-Achse wird auf einen diskreten Wert gerundet. Die Verteilung dieser diskreten Winkel wird dann Winkelhistogramm genannt. Abbildung 3.16 zeigt für einen Beispielscan die Bestimmung der auftretenden Winkel und das daraus gewonnene Histogramm.

Das Winkelhistogramm weist lokale Maxima für die im Scan enthaltenen Hauptrichtungen, z.B. lange Wände, auf. Um die Verdrehung der beiden Scans zu bestimmen, wird nun je ein Winkelhistogramm h_t des Referenzscans und h_s des aktuellen Scans angelegt und die Kreuzkorrelation k berechnet:

$$k(j) \quad = \quad \sum_{i=0}^{n-1} h_s(i) h_t(i + j \bmod n) \tag{3.33}$$

Hierbei geben $h_s(i)$ und $h_t(i)$, $0 \leq i < n$ die Werte des jeweiligen Histogramms an der Stelle i und n die Größe der Histogramme an.

Die Stelle, an der die Kreuzkorrelation ihr Maximum annimmt, entspricht

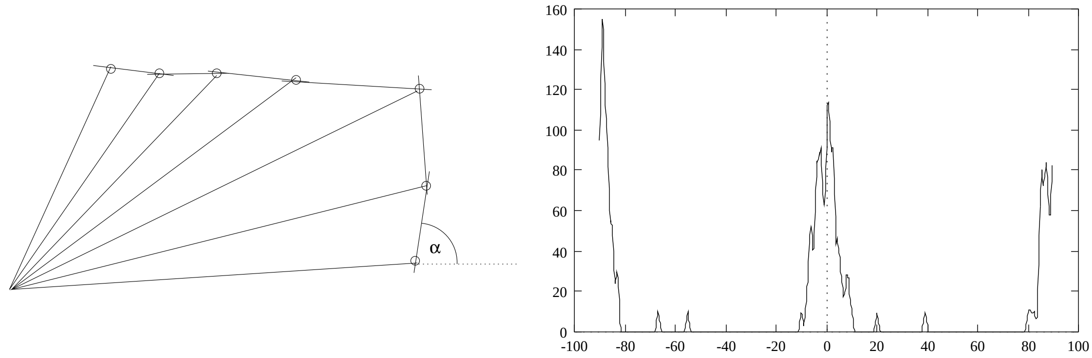


Abbildung 3.16: Erstellung eines Winkelhistogramms. Links Bestimmung der auftretenden Winkel, rechts das gewonnene Histogramm.

dann mit hoher Wahrscheinlichkeit der tatsächlichen Verdrehung. Die Kreuzkorrelation wird möglicherweise mehrere lokale Maxima aufweisen. Wenn man davon ausgeht, daß die Scans nur wenig verdreht sind, kann der Suchbereich aber auf wenige Grad Abweichung beschränkt und das richtige Maximum gefunden werden.

Nachdem die Verdrehung bekannt ist, werden nun beide Scans so gedreht, daß die Hauptrichtung des Referenzscans parallel zur x-Achse liegt. Da beide Scans in der gleichen Umgebung aufgenommen wurden, kann angenommen werden, daß h_s ebenfalls ein (zumindest lokales) Maximum an der entsprechenden Stelle der Hauptrichtung von h_t aufweist. Auf diese Weise kann nun durch Betrachten von x- und y-Histogrammen die Verschiebung der beiden Scans bestimmt werden.

Ein x-Histogramm ist eine Statistik über die Verteilung der x-Koordinate der Scanpunkte. Wie bei der Erstellung der Winkelhistogramme wird hierbei wieder auf diskrete Werte gerundet. Damit nicht ein unendlich großer Definitionsbereich vorliegt, werden die Werte auf eine endliche Länge gefaltet, d.h. es wird der Wert x modulo $size$ berechnet, wobei $size$ hinreichend groß gewählt werden muß, damit die Phasenverschiebung zwischen den x-Histogrammen noch eindeutig ist.

Durch Bestimmen des Maximums der Kreuzkorrelation kann die Verschiebung $\hat{d}_1$ in x-Richtung berechnet werden. Entsprechend kann für die y-Koordinate die Verschiebung $\hat{d}_2$ bestimmt werden. Durch Zurückdrehen der so berechneten Ergebnisse $\hat{d}_1$ und $\hat{d}_2$ kann man die Verschiebung der Scans bestimmen.

Der Aufwand für die Berechnung der Verdrehung und Verschiebung ist bei Scans mit n Punkten und einer Histogrammgröße von m Zellen gleich $O(n + m^2)$. Dies setzt sich aus dem Aufwand zur Erstellung der Histogramme und Drehen der Scans ($O(n)$) und dem Aufwand zur Bestimmung des Maximums der Kreuzkorrelation ($O(m^2)$) zusammen.

In [Gutmann, 1996; Gutmann und Schlegel, 1996] wurden zahlreiche Probleme dieses Ansatzes aufgezeigt und Erweiterungen entwickelt, um das Verfahren zu verbessern. So wurde beispielsweise das Verfahren für nicht-rechtwinklige Umgebungen erweitert und eine Heuristik erstellt, mit der sich die Genauigkeit der Überdeckung in Form einer Kovarianzmatrix berechnen läßt.

Dieser erweiterte Kreuzkorrelations-Algorithmus wurde ebenfalls experimentell mit dem Cox-, IDC- und dem kombinierten Scan-Matching-Verfahren verglichen. Dabei stellte sich heraus, daß das Kreuzkorrelations-Verfahren wie das Cox-Verfahren erwartungsgemäß nur in polygonalen Umgebungen gute Ergebnisse liefert. Der Cox-Algorithmus (und somit auch der kombinierte Scan-Matching-Algorithmus) ist hier jedoch sowohl in Rechenzeitbedarf wie auch in der berechneten Genauigkeit dem Kreuzkorrelations-Algorithmus überlegen [Gutmann, 1996; Gutmann und Schlegel, 1996].

3.9.2 Varianten des Cox-Algorithmus

Weitere Arbeiten beschäftigen sich mit der Optimierung des Cox-Algorithmus. Gonzalez *et al.* [1992] entwickelten ein zum Cox-Algorithmus äquivalentes Verfahren, bei dem die Zeit für die Zuordnung von Scanpunkten zu Modellinien optimiert wurde. Dies wurde durch Verwenden eines Gitters möglich, das für jede Zelle die Linien enthält, welche die Zelle schneiden. Dadurch wird eine Art Hash-Tabelle angelegt. Für die Zuordnung von Scanpunkten zu Linien müssen nun nur noch die Linien der Zelle, auf die der Scanpunkt fällt, und umliegende Zellen, untersucht werden. Dies reduziert die Gesamtzeit zum Überdecken des Scans, erfordert aber einen Vorverarbeitungsschritt.

3.9.3 Varianten des IDC-Algorithmus

In einer Arbeit von Bengtsson und Baerveldt [1999] wurden verschiedene Varianten des IDC-Algorithmus für den Einsatz in dynamischen Umgebungen entwickelt und miteinander verglichen. Hierzu wird ein Scan in verschiedene, sich überlappende Winkelbereiche eingeteilt und für jeden Winkelbereich separat eine Überdeckung bestimmt. Aus der Qualität der Überdeckung (hier der durchschnittliche Abstand der Scanpunktzuordnungen) wird entschieden, welche Winkelbereiche für die tatsächliche Bestimmung der Scanüberdeckung benutzt werden. Dieses erweiterte Verfahren wird experimentell in mehreren Varianten mit dem ursprünglichen IDC-Verfahren verglichen und eine Kombination von Varianten vorgeschlagen, welche die besten Resultate liefert.

3.9.4 Verwendung von Belegtheitsgittern

Eine andere von Moravec und Elfes entwickelte Überdeckungsmethode ist, die Scans in ein Belegtheitsgitter einzutragen und durch Korrelation die Verschiebung und Verdrehung zu bestimmen [Moravec und Elfes, 1985]. Hierzu wird im einfachsten Fall für jede Zelle ein Zustand *belegt* oder *frei* bestimmt und für verschiedene diskrete Verschiebungen und Winkel das Skalarprodukt der beiden

Gitter bestimmt. Hieraus ergibt sich direkt die Wahrscheinlichkeitsverteilung der Position des aktuellem Scan im Koordinatensystem des Referenzscans.

Rechenzeit und Genauigkeit dieses Verfahrens hängen von der Wahl der verwendeten Zellgröße ab. Eine kleine Zellgröße verbessert die Genauigkeit, erfordert aber mehr Rechenzeit. Ein Vorteil dieses Verfahrens ist, daß nicht nur ein Scan mit einem anderen Scan überdeckt werden kann, sondern es kann ein ganzer Satz von Scans mit einem Belegtheitsgitter überdeckt werden.

Eine kürzlich von Konolige entwickelte und optimierte Variante dieses Verfahrens erlaubt eine schnelle Berechnung der Positionsverteilung und wird zur globalen Positionsbestimmung eines Roboters eingesetzt [Konolige und Chou, 1999]. Rechenzeit und Genauigkeit sind aber weiterhin dem kombinierten Scan-Matcher unterlegen.

Ein weiterer gitterbasierter Ansatz zum Überdecken von Scans trägt beide Scans in je ein Gitter ein und führt anschließend eine sogenannte *Distanztransformation* durch [Prassler und Milios, 1995]. Auf diese Weise gewinnt man eine Karte von *Höhenlinien*, welche für die Überdeckung benutzt wird. Dieses Verfahren besitzt einen hohen Rechenbedarf und ist ungenau.

3.9.5 Feature-Matching

Shaffer *et al.* [1992] extrahieren Merkmale wie Liniensegmente oder Ecken aus Scandaten und ordnen ihnen Merkmale in einem apriori Modell zu. Für die Zuordnung von Merkmalen zueinander wird eine ungefähre Aufnahmeposition benötigt, um die sichtbaren Modellmerkmale vorherzusagen. Für die Überdeckung werden unäre Bedingungen (Winkel einer Ecke) und binäre Bedingungen (Winkel zwischen zwei Liniensegmenten) aufgestellt und die Korrespondenz gesucht, welche die meisten Bedingungen erfüllt.

Castellanos *et al.* [1996b] extrahieren Liniensegmente aus Laserdaten, weisen diesen Linien aus einem apriori Modell zu und bestimmen so die Überdeckung des Scans mit apriori Modell. Das Verfahren ist prinzipiell in der Lage, alle möglichen Positionen zu finden, für die es eine Überdeckung des Scans mit apriori Karte gibt. Weiterhin werden experimentell verschiedene algorithmische Varianten dieses Verfahrens miteinander verglichen. Jedoch liegen keine experimentellen Ergebnisse über Genauigkeit und Robustheit dieses Verfahrens vor.

Gutmann *et al.* [1999b] und Weigel [1999] entwickelten ein ähnliches Verfahren, um einen Scan schnell, genau und zuverlässig mit einem apriori Linienmodell zu überdecken. Dieses Verfahren wird in Kapitel 7 genauer beschrieben und mit anderen Scan-Matching-Verfahren experimentell verglichen.

3.9.6 Polygon-Matching

Eine theoretische Arbeit zur Lokalisation eines Roboters in einer polygonalen Umgebung wurde von Guibas *et al.* [1995] vorgestellt. Gegeben sind ein Polygon (die Umgebungskarte) und ein sternförmiges Polygon V (das Sichtbarkeitspolygon des Roboters). Gesucht sind dann alle Punkte in der Umgebungskarte, deren Sichtbarkeitspolygon gleich V sind. Das Verfahren benötigt einen sehr rechenintensiven Vorverarbeitungsschritt und erfordert zahlreiche Vorraussetzungen wie exakte Sensorik, exakte Umgebungskarte und einen Kompaß zur Bestimmung der Orientierung, was einen Einsatz unter realen Bedingungen ausschließt.

Karch *et al.* [Karch und Noltemeier, 1996; Karch *et al.*, 1997] konnten die Zeitkomplexität des Vorverarbeitungsschrittes verbessern und das Verfahren für Systeme ohne Kompaß und ungenauer Sensoren weiterentwickeln. Offen bleibt jedoch, ob das Verfahren auch in der Praxis eingesetzt werden kann, da z.B. keine Hindernisse in den Sichtbarkeitspolygonen liegen dürfen.

3.9.7 3d-Scan-Matching

Horn und Schmidt [1995] nutzen einen 3d-Laserscanner, um einen damit aufgenommenen Scan mit einem 3 dimensionalen apriori Modell zu überdecken. Zunächst werden aus dem 3d-Scan Flächen extrahiert und jeweils einer Fläche des apriori Modells zugeordnet. Aus diesen Zuordnungen wird die Summe der Abstandsquadrate gebildet und mittels einem analytischen Verfahren die Transformation gesucht, welche die Summe minimiert. Weiterhin wird eine Formel zur Berechnung der Kovarianzmatrix der berechneten Überdeckung hergeleitet.

Das Verfahren wurde in einer realen Umgebung getestet und besitzt eine Genauigkeit von ca. $5cm$ für den Ort und $3°$ in der Orientierung. Jedoch erfordert es den Einsatz eines 3d-Laserscanners, welcher teuer ist, eine hohe Aufnahmezeit erfordert oder nur einen geringen Sichtbereich hat.

3.9.8 Feature-Tracking

Eine zum Scan-Matching sehr ähnliche Methode ist Feature-Tracking. Hierzu wird für jedes Merkmal aus dem aktuellen Scan ein Merkmal aus einer apriori Umgebungskarte aufgrund einer initialen Positionsschätzung vorhergesagt und zugewiesen. Die Zuweisung geschieht für jedes Merkmal separat, d.h. es werden keine binären Bedingungen zwischen Merkmalen berücksichtigt. Da dieses Verfahren nicht versucht, den gesamten Scan einzupassen, sondern nur Scanteile jeweils separat mit der Umgebungskarte überdeckt, ist zu erwarten, daß diese Methode in dynamischen Umgebungen weniger robust als Scan-Matching ist.

Beispiele, welche diese Methode für die Lokalisierung eines mobilen Roboters benutzen, findet man in [Vestli *et al.*, 1994; Holenstein *et al.*, 1992; Crowley, 1989; Borthwick und Durrant-Whyte, 1994].

Kapitel 4

Selbstlokalisierung

Dieses Kapitel diskutiert Methoden zur Selbstlokalisierung eines mobilen Roboters. Die Aufgabe hierbei ist, die Position des Roboters aufgrund einer apriori Umgebungskarte und Sensordaten des Roboters zu bestimmen.

Generell unterscheidet man zwischen zwei verschiedenen Selbstlokalisierungsproblemen: globale Selbstlokalisierung und lokale Selbstlokalisierung. Bei der globalen Selbstlokalisierung wird der Roboter an einen beliebigen Ort gestellt und dem System wird die Gelegenheit gegeben, die Umwelt mit den Sensoren des Roboters zu beobachten. Das System muß dann durch Auswerten der Sensorinformationen entscheiden, an welchen möglichen Positionen der Roboter sich befinden kann. Der Prozeß, um diese Entscheidung zu treffen, ist im allgemeinen aufwendig und benötigt je nach Größe des Suchraums entsprechend viel Rechenzeit.

Bei der lokalen Selbstlokalisierung dagegen ist die ungefähre Position des Roboters bereits bekannt und es soll „nur“ eine Positionskorrektur berechnet werden. Dies ist der Fall, wenn der Roboter an einer ungefähr bekannten Position aufgestellt wird und dann fortlaufend seine Position durch Abgleich der Sensordaten mit Umgebungskarte bestimmt.

Diese Arbeit beschäftigt sich hauptsächlich mit lokalen Selbstlokalisierungsmethoden. Es werden jedoch auch globale Methoden präsentiert und ein Vergleich eines globalen mit einem lokalen Verfahren durchgeführt.

Zunächst wird eine bekannte Selbstlokalisierungsmethode, die Koppelnavigation, vorgestellt, welche Sensordaten der Odometrie auswertet. Dieses Verfahren wird in den meisten Robotersystemen eingesetzt.

Da diese Methode auf langen Fahrstrecken ungenaue Positionsinformationen liefert, werden für die Selbstlokalisierung Daten weiterer Sensoren benötigt. Die Auswertung dieser Daten kann auf unterschiedliche Art und Weise geschehen, anhand derer sich verschiedene Kategorien von Selbstlokalisierungsmethoden definieren lassen. Diese Kategorien werden in einem weiteren Abschnitt beschrieben.

Eine Kategorie, die dichte Sensordaten vergleichende Kategorie, wird näher betrachtet und verschiedene Methoden daraus kurz vorgestellt.

Danach wird auf eine Methode näher eingegangen, welche Scan-Matching und Kalman-Filterung benutzt, um eine lokale Positionsbestimmung zu realisieren. In dieser Arbeit werden zwei Scan-Matching-Methoden für die Selbstlokalisierung genauer untersucht, nämlich die Methode von Cox [1990], welche Scans mit einem apriori Linienmodell der Umgebung überdeckt, und das kombinierte Scan-Matching-Verfahren [Gutmann und Schlegel, 1996], welches Scans mit zuvor aufgenommenen Referenzscans abgleicht. Es wird ein mehrstündiger Beispiellauf eines Roboters in einer dynamischen Umgebung präsentiert, welcher die Robustheit der Scan-Matching-Methoden demonstriert.

Anschließend wird eine weitere, bereits entwickelte Methoden vorgestellt, welche in der Lage ist, den Roboter global zu lokalisieren. Die Methode heißt Markov-Lokalisierung und verwendet ein Positionsgitter oder Partikel zur Modellierung der Roboterposition.

Aufgrund der Vielzahl verschiedener Verfahren zur Selbstlokalisierung eines mobilen Roboters, stellt sich die Frage, welche Methode unter welchen Gegebenheiten die beste ist. In dieser Arbeit werden daher zwei verschiedene Verfahren (Scan-Matching und Markov-Lokalisierung) in verschiedenen Umgebungen und unter variierenden Bedingungen experimentell miteinander verglichen. Dabei stellt sich heraus, daß eine Kombination der beiden Methoden die Vorteile beider Verfahren miteinander verbinden kann. Nach Wissensstand des Autors ist dieser Vergleich bisher der erste Vergleich von robusten Selbstlokalisierungsmethoden, die erfolgreich in dynamischen Umgebungen eingesetzt wurden.

4.1 Koppelnavigation

Bei der Koppelnavigation[1] wird die Positionsänderung eines Fahrzeugs durch Messen des zurückgelegten Weges eines oder mehrere Räder bestimmt. Zu diesem Zweck sind an den Rädern Sensoren angebracht, welche die Drehbewegung des jeweiligen Rades messen. Zusätzlich kann ein Kreiselkompaß verwendet werden, um die Orientierung des Roboters zuverlässiger zu bestimmen.

4.1.1 Dreiradkinematik

Abbildung 4.1 zeigt einen Roboter mit Dreiradkinematik. Diese Kinematik wird in der mobilen Robotik häufig verwendet. Beispielsweise kommt sie im *Pioneer-I*-Roboter zum Einsatz, welcher für zahlreiche Experimente in dieser Arbeit benutzt wurde. Im folgenden wird nur auf diese Kinematik näher eingegangen. Die Resultate können aber auch auf andere Kinematiksysteme übertragen werden.

Die beiden Vorderräder beim *Pioneer-I*-Roboter sind die Antriebsräder und sind mit Sensoren zur Bestimmung der gefahrenen Strecke ausgestattet. Das Hinterrad ist ein frei drehendes und leer laufendes Rad, wie es beispielsweise bei

[1] Im englischen auch als *dead-reckoning* bezeichnet.

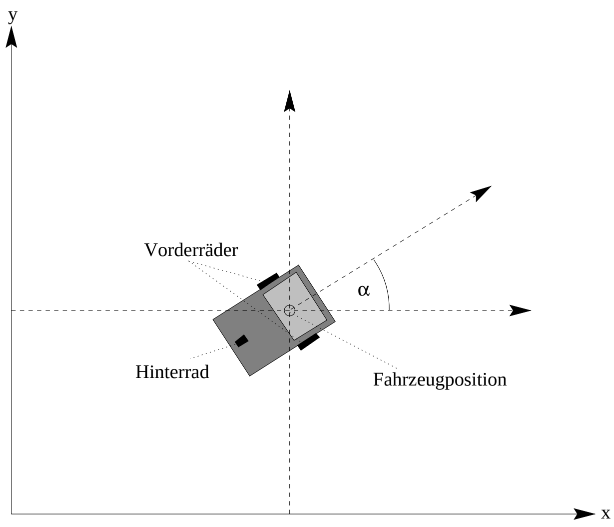


Abbildung 4.1: Dreiradkinematik wie sie beispielsweise auf dem *Pioneer-I*-Roboter eingesetzt wird.

Bürostühlen verwendet wird. Üblicherweise wird bei dieser Kinematik der Referenzpunkt der Fahrzeugposition in der Mitte der Achse der Räder mit Odometriesensoren gewählt. Dies führt zu einer einfachen Berechnung der Positionsänderung aus den Strecken, die von den beiden Rädern zurückgelegt wurden. Der zurückgelegte Weg ergibt sich als Mittelwert der Wegstrecken der beiden Räder und die Orientierungsänderung ist proportional zu der Differenz der beiden Werte. Im folgenden wird beschrieben, wie sich die aktuelle Position aus diesen beiden Informationen berechnen läßt.

Bewegt sich das Fahrzeug, so wird in regelmäßigen Zeitabständen die Fahrzeugposition aktualisiert. Hierzu wird der zurückgelegte Weg δ und die Änderung in der Orientierung α seit dem letzten Zeitpunkt gemessen und mit der aktuellen Fahrzeugposition verrechnet. Zur Vereinfachung nimmt man an, daß sich das Fahrzeug in dieser Zeitspanne nahezu geradlinig bewegt. Man erhält dadurch zwar einen gewissen Fehler, dieser kann aber durch die Wahl eines kleineren Zeitabstands zwischen den Messungen beliebig verkleinert werden.

Die Roboterposition $l = (x, y, \theta)^T$ wird durch die Eingabe $a = (\delta, \alpha)^T$ nach folgender Formel aktualisiert:

$$l \;\leftarrow\; F(l,a) = \begin{pmatrix} x + \delta\cos(\theta) \\ y + \delta\sin(\theta) \\ \theta + \alpha \end{pmatrix} \tag{4.1}$$

Ein Problem dieser Positionsbestimmung ist, daß Fehler in der Odometrie auftreten. Beispielsweise können die Räder rutschen (Schlupf), unrund sein, es können durch Bodenunebenheiten falsche Entfernungen gemessen werden, oder

der Boden, auf dem gefahren wird, erlaubt keine genau Messung der zurückgelegten Strecke (z.B. weicher Teppichboden). Diese Fehler sind normalerweise recht gering, d.h. auf kurzen Strecken liefert die Koppelnavigation sehr genaue Ergebnisse. Jedoch nehmen die Fehler mit der gefahrenen Strecke zu und wachsen praktisch ohne obere Grenze.

Abbildung 4.2a zeigt eine Route in der Abteilung *Grundlagen der künstlichen Intelligenz* der Universität Freiburg, die von einem *Pioneer-I*-Roboter abgefahren wurde. Der Roboter startete im rechten oberen Raum, fuhr durch die gesamte Abteilung und endete im linken oberen Raum. Abbildung 4.2b zeigt die Positionsschätzungen aufgrund von Odometrieinformationen. Es ist gut zu sehen, wie anfangs die Positionsschätzung noch recht gut ist, jedoch am Ende deutlich von der tatsächlichen Position abweicht.

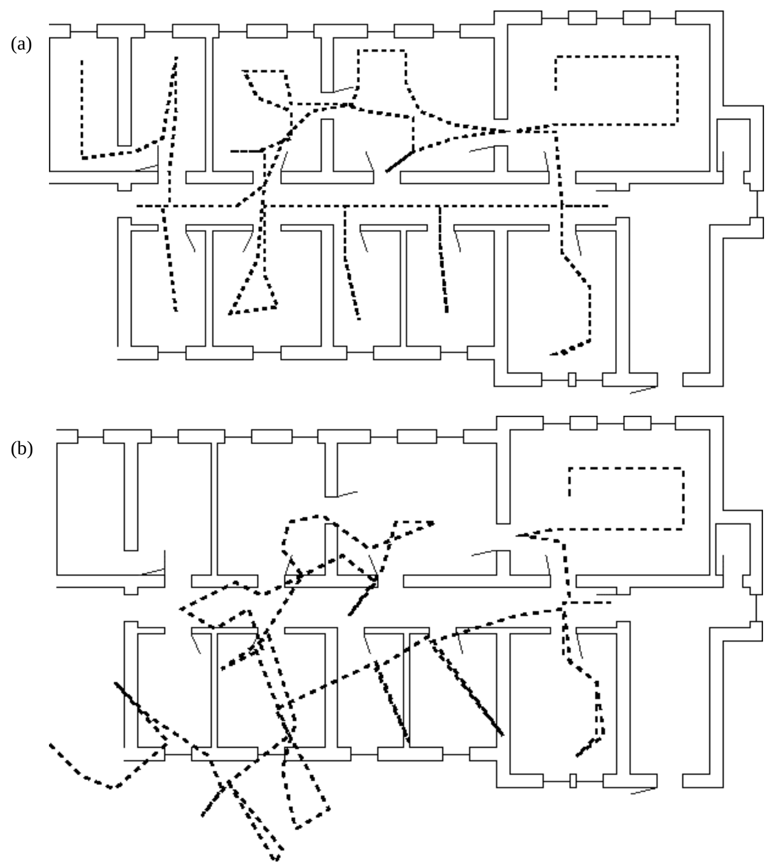


Abbildung 4.2: Driftfehler in der Koppelnavigation. (a) Tatsächlich abgefahrener Pfad. (b) Positionsschätzungen der Koppelnavigation.

Natürlich können die in der Odometrie auftretenden Fehler reduziert werden, indem beispielsweise eine zweite Odometrie verwendet wird oder ein Anhängersystem zum Einsatz kommt [Borenstein, 1994; Feng *et al.*, 1996]. Jedoch wird dadurch nur der Fehler verkleinert, nicht aber das grundsätzliche Problem gelöst, daß die Position über größere Strecken ungenau wird. Um dieses Problem zu lösen, müssen weitere Sensoren einbezogen werden, beispielsweise entfernungsmessende Sensoren wie Laserscanner. Es kann jedoch der Positionsfehler, welcher in der Koppelnavigation entsteht, modelliert werden. Im folgenden soll hierauf näher eingegangen werden.

4.1.2 Positionsfehler

Es wird angenommen, daß die Fehler in der Positionsbestimmung durch Koppelnavigation normalverteilt sind, d.h. gemessene Entfernung und Rotation unterliegen einer Gauß-Verteilung. Weiterhin wird in vielen Positionierungssystemen (z.B. Systeme, die auf Kalman-Filterung beruhen) die Roboterposition ebenfalls durch eine Gauß-Verteilung modelliert. Dies führt zu folgender Repräsentation.

$$
\begin{aligned}
l &\sim N(\mu_l, \Sigma_l) && (4.2)\\
\mu_l &= (\hat{x}, \hat{y}, \hat{\theta})^T && (4.3)\\
\Sigma_l &= \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 \end{pmatrix} && (4.4)\\
a &\sim N(\mu_a, \Sigma_a) && (4.5)\\
\mu_a &= (\hat{\delta}, \hat{\alpha})^T && (4.6)\\
\Sigma_a &= \begin{pmatrix} \sigma_\delta^2 & 0 \\ 0 & \sigma_\alpha^2 \end{pmatrix} && (4.7)
\end{aligned}
$$

Bei der Eingabe a geht man davon aus, daß der zurückgelegte Weg δ nicht mit der Änderung in der Orientierung α korreliert, d.h. $\sigma_{\delta\alpha} = 0$.

Werden die Parameter der Funktion F als ein Vektor $(x, y, \theta, \delta, \alpha)^T$ betrachtet, so können direkt die Formeln 2.11 und 2.12 verwendet werden, um aus der alten Fahrzeugposition und der Eingabe die neue Fahrzeugposition zu bestimmen.

$$
\begin{aligned}
\mu_l &\leftarrow F(\mu_l, \mu_a) && (4.8)\\
\Sigma_l &\leftarrow \nabla F_{la} \Sigma_{la} \nabla F_{la}^T && (4.9)\\
\nabla F_{la} &= \begin{pmatrix} 1 & 0 & -\hat{\delta}\sin(\hat{\theta}) & \cos(\hat{\theta}) & 0 \\ 0 & 1 & \hat{\delta}\cos(\hat{\theta}) & \sin(\hat{\theta}) & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} && (4.10)\\
\Sigma_{la} &= \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & 0 & 0 \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & 0 & 0 \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\delta^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\alpha^2 \end{pmatrix} && (4.11)
\end{aligned}
$$

Auch hier wird angenommen, daß die Fahrzeugposition nicht mit der Eingabe korreliert. Durch Umformen vereinfacht sich Formel 4.9 zu:

$$\Sigma_l \leftarrow \nabla F_l \Sigma_l \nabla F_l^T + \nabla F_a \Sigma_a \nabla F_a^T \tag{4.12}$$

wobei ∇F_l und ∇F_a die entsprechenden Teilmatrizen von ∇F_{la} sind.

Das Verhalten dieses Systems wird nun anhand einer simulierten Beispielfahrt demonstriert. Abbildung 4.3a zeigt mehrere Stationen dieser Roboterfahrt in einer zyklischen Umgebung. Das Fahrzeug startete links unten und fuhr im Uhrzeigersinn in einem Quadrat in die Nähe der Ausgangsposition zurück. Das Quadrat hat in etwa eine Kantenlänge von 10 Metern. An mehreren Stationen wurde die Fahrzeugposition in Form von Schätzwert (Erwartungswert) und Fehler (Kovarianzmatrix) notiert. Die geschätzte Position ist durch einen länglichen Trichter markiert. Die Spitze des Trichters gibt den Ort des Fahrzeugs an, während die Richtung der Trichteröffnung die Fahrzeugorientierung zeigt. Um den Fehler der Fahrzeugposition darzustellen, wurde aus der jeweiligen Kovarianzmatrix die Untermatrix

$$\Sigma_{xy} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{pmatrix} \tag{4.13}$$

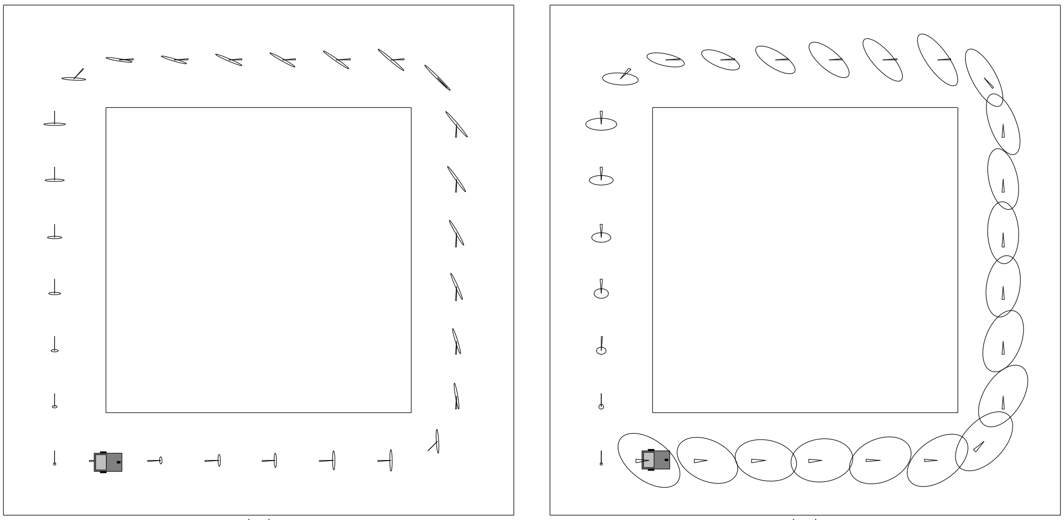
und der Wert von σ_θ^2 extrahiert. Σ_{xy} wurde benutzt, um alle Punkte der Ebene zu bestimmen, die einen Mahalanobis-Abstand zum Schätzwert kleiner gleich 1 haben. Diese Punkte sind jeweils durch eine Ellipse angedeutet, d.h. der tatsächliche Ort des Fahrzeugs befindet sich mit hoher Wahrscheinlichkeit in diesem Bereich. Der Wert von σ_θ^2 wurde benutzt, um die Öffnung des Trichters darzustellen. Auch hier befinden sind alle Orientierungen, die einen Mahalanobis-Abstand zum Schätzwert kleiner gleich 1 haben, innerhalb der Richtungen der beiden Trichterkanten.

In dem Beispiel wurde der anfängliche Positionsfehler auf den Wert

$$\Sigma_l = \begin{pmatrix} 50^2 & 0 & 0 \\ 0 & 50^2 & 0 \\ 0 & 0 & 0.001 \end{pmatrix}$$

gesetzt, d.h. der Ort in x- und y-Richtung wurde mit einer Standardabweichung von jeweils $5cm$ und die Orientierung mit einer Standardabweichung von 2° bestimmt. Der Fehler der Eingabe wurde zunächst auf die Nullmatrix gesetzt, d.h. die Koppelnavigation arbeitete fehlerfrei.

Man erkennt deutlich, daß der Positionsfehler des Fahrzeugs in der rechten oberen Ecke maximal wird. Dieser Fehler wird alleine durch die anfängliche Unsicherheit in der Orientierung so groß. Bei der Rückkehr des Fahrzeugs in die Nähe der Ausgangsposition nimmt der Fehler der Fahrzeugposition wieder auf seinen ursprünglichen Wert ab. Dies ist intuitiv auch richtig, da in diesem Beispiel die Koppelnavigation ja als fehlerfrei angenommen wurde.



(a) (b)

Abbildung 4.3: Positionsschätzungen mit Kovarianzmatrizen einer Beispielfahrt mit fehlerfreien (a) und fehlerbehafteten (b) Koppelnavigation.

In einem weiteren Versuch wurde nun der Fehler der Eingabe berücksichtigt. Hierbei wurde angenommen, daß das Fahrzeug über einen Kreiselkompaß verfügt und sich der Fehler in der Entfernungsmessung linear zur gefahrenen Entfernung, und der Fehler in der Orientierung linear zur verstrichenen Zeit verhält. Dies ist vernünftig, da die Genauigkeit einer Odometrie umso mehr abnimmt, je weiter gefahren wurde, und die Orientierungsmessung eines Kreiselkompasses mit der Zeit ungenau wird. In diesem Versuch wurde für σ_δ der Wert 50mm pro 1000mm, und für σ_α der Wert 0.1° pro Zeiteinheit gesetzt, wobei von einem Aufnahmepunkt zum nächsten ungefähr 160 Zeiteinheiten vergehen.

Abbildung 4.3b zeigt das Ergebnis dieser Fahrt. Man sieht, daß der Fehler der Fahrzeugposition am Ende der Fahrt deutlich größer ist als zu Beginn. Der Fehler ist in diesem Fall hauptsächlich auf den Fehler in der Koppelnavigation zurückzuführen.

Wang [1990] hat das Modell zur Koppelnavigation noch verbessert. In seiner Arbeit wird die Annahme getroffen, daß sich das Fahrzeug in der verstrichenen Zeitspanne nicht geradlinig bewegt sondern einen Kreisbogen beschreibt. Dadurch wird die Berechnung der neuen Fahrzeugposition genauer. Viel Sorgfalt wurde bei der Bestimmung des Fehlers betrieben. Es wurden mehrere Formeln zur Fehlerberechnung hergeleitet und miteinander verglichen.

Zusammenfassend ist die Lokalisierungsmethode durch Koppelnavigation eine einfache und schnelle Positionierungsmethode, die in den meisten Robotersystemen eingesetzt wird. Auf kurzen Strecken erweist sie sich als sehr genau, auf größeren Strecken wird die Positionsschätzung aber unbrauchbar. Daher müssen für ein komplettes Selbstlokalisierungssystem weitere Sensoren und Verfahren miteinbezogen werden. Der nächste Abschnitt stellt verschiedene Kategorien von Selbstlokalisierungsmethoden vor, die in der Vergangenheit zum Einsatz kamen.

4.2 Kategorien von Lokalisierungsmethoden

Lokalisierungsmethoden lassen sich allgemein in die drei Kategorien verhaltensbasierte Ansätze, Landmarkenlokalisierung und dichte Sensordaten vergleichende Verfahren einteilen.

Verhaltensbasierte Ansätze beruhen auf der Interaktion von Roboteraktionen mit der Umwelt, um zu navigieren. Zum Beispiel folgte Connell's Roboter Herbert einer Faustregel, um durch eine Büroumgebung zu fahren und seinen Weg zurückzufinden, indem die Prozedur der Hinfahrt rückwärts angewandt wurde [Connell, 1990]. Weiter verfeinerte Systeme lernen interne Strukturen, die „abgespielt“ werden können, um Aktionen zu wiederholen oder rückgängig zu machen [Arkin, 1990]. Während verhaltensbasierte Systeme für bestimmte Aufgaben sehr nützlich sind, ist ihre Fähigkeit einen Roboter geometrisch zu lokalisieren begrenzt, da die Navigationsfähigkeit implizit in der Sensor- und Aktionshistorie liegt.

Landmarkenlokalisierung beruht auf der Erkennung von Landmarken, um den Roboter geometrisch zu lokalisieren. Die Landmarken können können *a priori* gegeben sein (z.B. die Satelliten im GPS-Navigationssystem oder in der Umgebung angebrachte Marken, die mit speziellen Mustern oder auffälligen Farben versehen sind), oder vom Robotersystem in einer Explorationsphase gelernt werden (z.B. Sonar-Landmarken [Leonard *et al.*, 1990].) Während Landmarkenlokalisierung beeindruckende Ergebnisse in geometrischer Lokalisierung erreichen kann, muß entweder die Einsatzumgebung vorher eingerichtet werden oder natürliche Landmarken effizient und robust erkannt werden können.

Im Gegensatz hierzu, versuchen dichte Sensordaten vergleichende Verfahren (z.B. [Burgard *et al.*, 1996; Gutmann und Schlegel, 1996; Lu und Milios, 1997b; Schultz und Adams, 1996; Zhang und Faugeras, 1992]) die gesamte verfügbare Sensorinformation zu verwenden, um die Roboterposition zu bestimmen. Dies wird dadurch bewerkstelligt, daß dichte Sensorscans mit einer Oberflächenkarte der Umgebung verglichen werden ohne dabei Landmarken aus den Sensordaten zu extrahieren. Daher können dichte Sensordaten vergleichende Verfahren sich beliebige in den Sensordaten vorhandene Merkmale zum Vorteil machen, ohne dabei explizit diese Merkmale zu definieren.

Im folgenden wird auf dichte Sensordaten vergleichende Verfahren näher eingegangen und verschiedene Verfahren, die erfolgreich in letzter Zeit eingesetzt wurden, vorgestellt.

4.3 Dichte Sensordaten vergleichende Verfahren

Allgemein in probabilistischen Termen ausgedrückt ist Lokalisierung der Prozeß die Wahrscheinlichkeit $p(l \mid O^n, A^n)$ zu bestimmen, daß der Roboter sich an einer Position l befindet, gegeben eine Historie von geschätzten Bewegungen A^n und eine Historie von Sensormessungen O^n in einer bekannten Umgebung. In der Praxis ist es zu aufwendig den wechselseitigen Effekt aller Odometrie- und Sensormessungen zu bestimmen. Daher wird eine rekursive Approximation verwendet, welche die Positionswahrscheinlichkeit zum Zeitpunkt n lediglich aus der Wahrscheinlichkeitsverteilung zum Zeitpunkt $n-1$ und den Odometrie- und Sensormessungen zum Zeitpunkt n berechnet, wobei angenommen wird, daß die einzelnen Beobachtungen bei gegebener Karte unabhängig voneinander sind:

$$p(l \mid O^n, A^n) \;=\; \alpha \cdot \int p(l \mid o_n, a_n, l')p(l' \mid O^{n-1}, A^{n-1})\; dl'. \qquad (4.14)$$

Hierbei ist l' die Roboterposition zum Zeitpunkt $n-1$ und α ein Normalisierungsfaktor, der sicherstellt, daß die Summe $p(l \mid O^n, A^n)$ über alle Positionen l eins ergibt.

Für gewöhnlich wird die rekursive Bestimmung der Positionswahrscheinlichkeit in zwei Schritten durchgeführt.

1. Vorhersage der neuen Roboterposition l mit zugehöriger Unsicherheit aus der vorherigen Position l' und den Messungen der Odometrie.

2. Aktualisierung der Roboterposition l und zugehöriger Unsicherheit durch Vergleich von Sensorinformation mit einer Umgebungskarte.

Im ersten Schritt wird die Unsicherheit in der Roboterposition für gewöhnlich vergrößert, da die Odometriemessungen wie in diesem Kapitel beschrieben fehlerbehaftet sind. Im zweiten Schritt wird die Unsicherheit im allgemeinen verkleinert.

Der Vorhersageschritt wird durch eine bedingte Wahrscheinlichkeit $p(l \mid a_n, l')$, dem Bewegungsmodell, beschrieben, welche die Wahrscheinlichkeit ausdrückt, daß Aktion a_n ausgeführt an Position l' den Roboter an Position l bewegt. Nachdem sich der Roboter bewegt hat, wird die neue Positionswahrscheinlichkeit wie folgt berechnet:

$$p(l) \;\leftarrow\; \int p(l \mid a_n, l')p(l')\; dl' \qquad (4.15)$$

Hierbei nehmen die meisten Verfahren für den Odometriefehler (also $p(l \mid a_n, l')$) eine Normalverteilung an.

Im Aktualisierungsschritt wird die neue Positionswahrscheinlichkeit durch Anwenden der Bayes'schen Regel berechnet:

$$p(l) \;\leftarrow\; p(l \mid o_n) = \alpha \cdot p(o_n \mid l)p(l) \qquad (4.16)$$

Hierbei bestimmt das Sensormodell $p(o_n|l)$ die Wahrscheinlichkeit der Sensormessung o_n gegeben der Roboter befindet sich an Position l.

Verfahren, die das soeben beschriebene rekursive Modell benutzen, müssen nur noch festlegen, wie sie die Wahrscheinlichkeitsverteilung $p(l)$ repräsentieren und wie Bewegungs- und Sensormodell aussehen. Im folgenden werden hierzu drei verschiedene Verfahren vorgestellt: Scan-Matching mit Kalman-Filterung, das eine Gauß-Verteilung als Wahrscheinlichkeitsverteilung verwendet[2], gitterbasierte Markov-Lokalisierung, das stückweise lineare Funktionen (auch Gitter genannt) hierfür einsetzt, und Monte-Carlo-Lokalisierung, welches Partikel verwendet.

4.4 Scan-Matching-Lokalisierung

Ein Verfahren, das erfolgreich zur Lokalisierung eines mobilen Roboters eingesetzt wird, ist Lokalisierung durch Scan-Matching [Cox, 1990; Weiß und Puttkamer, 1995; Lu und Milios, 1997b; Gutmann und Schlegel, 1996; Gutmann und Nebel, 1997]. Oftmals wird ein Kalman-Filter verwendet, der Positionsschätzungen von Odometrie und Scan-Matching fusioniert.

Lokalisierung durch Scan-Matching mit Kalman-Filterung repräsentiert die Wahrscheinlichkeitsverteilung der Roboterposition durch eine Gauß-Verteilung:

$$p(l) \;=\; N(\mu_l, \Sigma_l) \tag{4.17}$$

Ebenso werden Odometriefehler und Scan-Matching-Korrektur mit Gauß-Verteilungen modelliert. Dies hat den Vorteil, daß Roboterpositionen mit hoher Präzision berechnet werden können und eine effiziente Fusionsmethode, nämlich Kalman-Filterung, verwendet werden kann.

Bewegt sich der Roboter um $a = (\delta, \alpha)^T$, so wird im Vorhersageschritt die neue Roboterposition nach den Formeln 4.8 und 4.12 berechnet, d.h.

$$\mu_l \;\leftarrow\; F(\mu_l, \mu_a) = \begin{pmatrix} \hat{x} + \hat{\delta}\cos(\hat{\theta}) \\ \hat{y} + \hat{\delta}\sin(\hat{\theta}) \\ \hat{\theta} + \hat{\alpha} \end{pmatrix} \tag{4.18}$$

$$\Sigma_l \;\leftarrow\; \nabla F_l \Sigma_l \nabla F_l^T + \nabla F_a \Sigma_a \nabla F_a^T \tag{4.19}$$

Im Aktualisierungsschritt wird diese Positionsschätzung mit dem Ergebnis $o \sim N(\mu_o, \Sigma_o)$ einer Scanüberdeckung fusioniert. Hierzu werden die Kalman-Filter-Gleichungen 2.23 und 2.24 angewandt:

$$\mu_l \;\leftarrow\; (\Sigma_l^{-1} + \Sigma_o^{-1})^{-1}(\Sigma_l^{-1}\mu_l + \Sigma_o^{-1}\mu_o) \tag{4.20}$$

$$\Sigma_l \;\leftarrow\; (\Sigma_l^{-1} + \Sigma_o^{-1})^{-1} \tag{4.21}$$

[2] Natürlich muß dem Kalman-Filter nicht notwendigerweise eine Gaußverteilung zugrunde liegen, sondern es können die ersten beiden Momente beliebiger Verteilungsfunktionen geschätzt werden.

Diese Gleichungen belegen, daß Selbstlokalisierung durch Kalman-Filterung effizient implementiert werden kann. Solange die Fehlermodelle genau sind, liefert Kalman-Filterung eine sehr gute Positionsschätzung.

Der Erfolg des Kalman-Filters hängt jedoch stark vom Resultat der Scanüberdeckung ab. Liefert Scan-Matching eine falsche Positionsschätzung, so berechnet der Kalman-Filter ebenfalls eine falsche, fusionierte Schätzung. Dies kann zu katastrophalen Fehlern führen, da die Positionsschätzungen des Kalman-Filters als initiale Scanposition für die Scanüberdeckung dienen. Im schlimmsten Fall kann der Roboter sich nicht mehr lokalisieren oder liefert falsche Positionswerte.

Um diese Fälle weitestgehend zu vermeiden, sollten weitere Mechanismen vorgesehen werden, die beispielsweise schlechte Ergebnisse der Scanüberdeckung erkennen und in diesen Fällen die Anwendung des Kalman-Filters verhindern. Im folgenden werden zwei verschiedene Lokalisierungsverfahren, die Scan-Matching benutzen, vorgestellt.

4.4.1 Scan-Matching mit Linienmodell

Beim Scan-Matching mit Linienmodell wird ein aufgenommener Scan mit einem apriori Modell von Liniensegmenten überdeckt. Hierzu werden entweder direkt Scanpunkte Liniensegmenten aus der Umgebungskarte zugeordnet [Cox, 1990; Gonzalez *et al.*, 1992] oder es werden zunächst Merkmale aus den Laserdaten extrahiert und dann Merkmalen der Umgebungskarte zugeordnet [Shaffer *et al*, 1992; Castellanos *et al.*, 1996b; Gutmann *et al.*, 1999b; Weigel, 1999].

In diesem Abschnitt wird das in Abschnitt 3.6 beschriebene Verfahren von Cox [1990], welches Scanpunkte direkt Liniensegmenten der Umgebungskarte zuordnet, für die Selbstlokalisierung benutzt. Da die Komplexität dieses Verfahrens direkt von der Anzahl Liniensegmente der Umgebungskarte abhängt, ist es sinnvoll Erweiterungen zu entwerfen, welche die Bestimmung des zu einem Scanpunkt zugehörigen Liniensegmentes optimieren.

Neben dem von Gonzalez *et al.* [1992] vorgeschlagenen gitterbasierten Ansatz (siehe Abschnitt 3.9.2) zur effizienten Bestimmung der Korrespondenzlinien, gibt es noch eine weitere effiziente Methode, welche aus der bestehenden Menge von Liniensegmenten eine begrenzte Untermenge auswählt. Sofern angenommen werden kann, daß die ungefähre Roboterposition bekannt ist, können als Untermenge die Liniensegmente hergenommen werden, die von dieser Position aus sichtbar sind. Die Bestimmung dieser Liniensegmente kann durch Aufnahme eines simulierten Scans an der Stelle der ungefähren Roboterposition erfolgen, indem z.B. ein *Ray-Tracing*-Verfahren eingesetzt wird. Neben der Reduzierung der Anzahl Liniensegmente hat diese Methode noch einen weiteren Vorteil. Da nur die Segmente ausgewählt werden, die von der ungefähren Roboterposition aus sichtbar sind, werden unsinnige Zuordnungen von Scanpunkten zu Liniensegmenten, wie z.B. die Zuordnung eines Scanpunktes zur Rückseite einer Wand, verhindert.

Nach dieser Vorverarbeitung kann der aktuelle Scan mit der Untermenge von

Liniensegmenten überdeckt werden. Ergebnis dieser Überdeckung ist eine Positionsschätzung o und ein Fehlerwert $error$ (siehe Abschnitt 3.6). Liegt eine gute Überdeckung vor, d.h. $error < MAX\text{-}ERROR$, so kann die von Scan-Matching berechnete Positionsschätzung o direkt mit der Odometrieschätzung nach den Formeln 4.20 und 4.21 verrechnet werden. Ansonsten wird das Ergebnis ignoriert und die Positionsbestimmung alleine der Odometrie überlassen.

Das soeben beschrieben Verfahren wurde in mehreren Umgebungen getestet und Genauigkeit und Robustheit experimentell bestimmt. Diese Ergebnisse sowie ein Vergleich mit anderen Verfahren folgen in Abschnitt 4.6. Abbildung 4.4 zeigt ein typisches Linienmodell einer Büroumgebung mit einer Beispiellokalisierung eines mobilen Roboters durch Anwendung dieses Verfahrens.

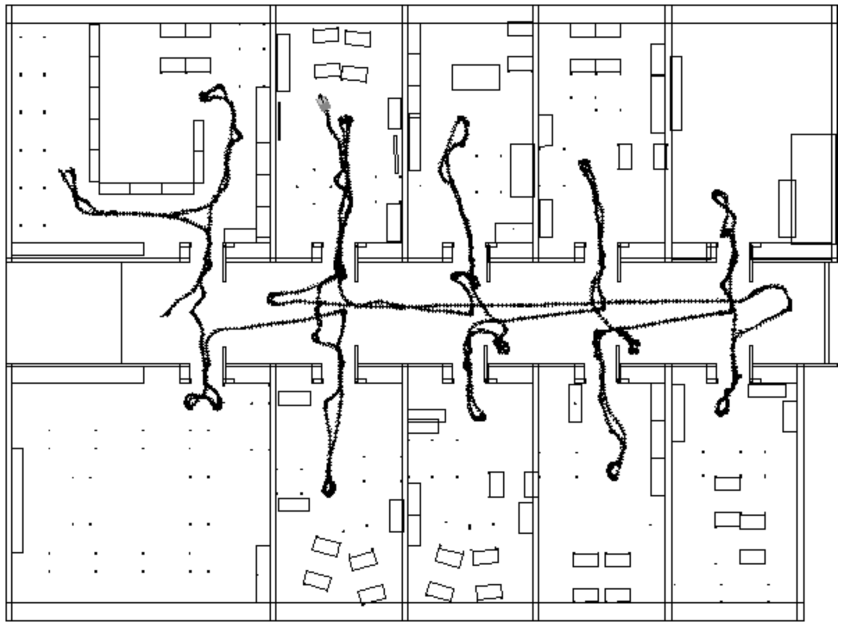


Abbildung 4.4: Linienmodell der Abteilung Informatik der Universität Bonn mit einer Beispiellokalisierung des Roboters *RHINO*.

4.4.2 Scan-Matching mit Referenzscans

Eine andere Möglichkeit, die Roboterposition durch Scan-Matching zu bestimmen, ist einen aufgenommenen Scan nicht mit einem Linienmodell, sondern mit einem Referenzscan zu überdecken [Weiß und Puttkamer, 1995; Lu und Milios, 1997b; Gutmann und Nebel, 1997]. Basis dieses Verfahrens ist eine Karte von Referenzscans, welche nicht weiter interpretiert werden [Weiß und Puttkamer, 1995]. Dies hat den Vorteil, daß keine expliziten Merkmale für die Lokalisation vorausgesetzt werden, also z.B. die Umgebung nicht unbedingt polygonal sein muß [Lu und Milios, 1997b].

Lokalisierung

Zunächst muß die Karte von Referenzscans erstellt werden. Dies kann dadurch realisiert werden, daß der Roboter in einer Explorationsfahrt Scans aufnimmt. Da die Aufnahmepositionen dieser Scans nur durch Odometrieinformationen bekannt sind, sind diese für gewöhnlich zu ungenau, um direkt verwendet werden zu können. Daher erfolgt zunächst eine Korrektur der Aufnahmepositionen. In nicht allzu großen Umgebungen kann hierfür das Verfahren von Lu und Milios [1997a] verwendet werden. Dieses Verfahren sowie weitere Verfahren zur Erstellung von Umgebungskarten in großen zyklischen Umgebungen werden im Kapitel 5 beschrieben.

Sei nun also angenommen, daß eine konsistente Karte von Referenzscans vorliegt. Eine Lokalisation des Roboters kann dadurch erfolgen, daß ein neu aufgenommener Scan mit einem der Referenzscans überdeckt wird. Für die Auswahl des Referenzscans sollte derjenige Scan herangezogen werden, welcher die größte Überlappung basierend auf der ungefähren Roboterposition besitzt. In der Regel ist dies der Scan, welcher der aktuellen Roboterposition am nächsten liegt.

Die Aktualisierung der Roboterposition geschieht nun wie folgt. Zum neu aufgenommenen Scan s wird der Referenzscan t als der Scan bestimmt, dessen Aufnahmeposition am nächsten zur Aufnahmeposition von s liegt. Auf beide Scans wird der Projektionsfilter wie er in Abschnitt 3.4.5 beschrieben ist angewandt. Hierdurch enthalten beide Scans nur noch diejenigen Punkte, die einen potentiellen Korrespondenzpartner im jeweils anderen Scan besitzen.

Bevor nun die beiden Scans überdeckt werden, erfolgt eine Überprüfung, ob noch genügend Scanpunkte in beiden Scans enthalten sind. Hierzu wird für jeden Scan die Summe aller Winkelbereiche, welche Scanpunkte enthalten, bezüglich der Aufnahmeposition bestimmt. Abbildung 4.5 zeigt zwei Beispielscans und die Bestimmung der Winkelbereiche für den auf der rechten Seite aufgenommenen Scan.

Unterschreitet die Summe der Winkelbereiche eine bestimmte Schranke für einen der beiden Scans, so ist die Gefahr groß, daß Scan-Matching ein falsches Ergebnis berechnet, da nicht mehr genug zueinander korrespondierende Scanpunkte vorliegen. In diesem Fall wird keine Selbstlokalisierung mit den beiden Scans durchgeführt. In der Praxis hat sich ein Wert zwischen $60°$ und $90°$ für diese Schranke bewährt.

Jetzt können die beiden Scans überdeckt werden. Als Überdeckungsverfahren bietet sich das kombinierte Scan-Matching-Verfahren aus Abschnitt 3.8 an. Ergebnis dieser Überdeckung ist eine Schätzung $o' = ((\hat{x}, \hat{y}, \hat{\theta})^T, \Sigma_{match})$ der relativen Position von s im Koordinatensystem von t und ein Fehlerwert $error$, welcher die Güte der Überdeckung angibt.

Wie beim Scan-Matching-Lokalisierungsverfahren mit Linienmodell wird der Fehlerwert benutzt, um schlechte Überdeckungen zu erkennen und zu verwerfen. Sofern eine gute Überdeckung, d.h. $error < MAX\text{-}ERROR$, vorliegt und

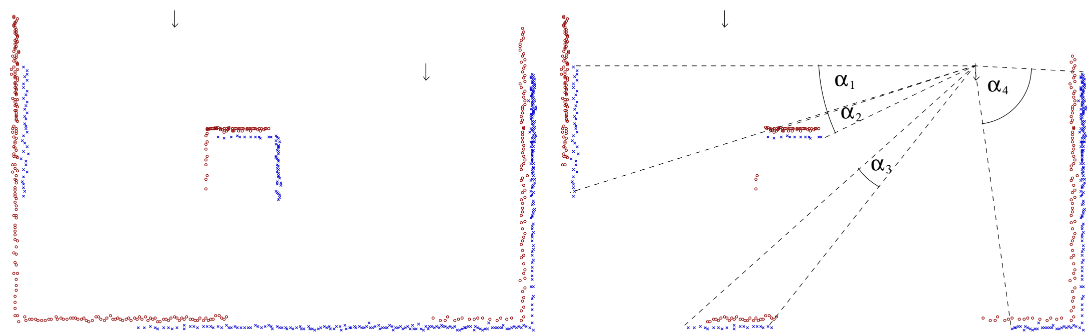


Abbildung 4.5: Beispiel für die Anwendung des Projektionsfilters auf zwei Scans. Links Scans vor Anwendung des Projektionsfilters, rechts resultierende Scans und Bestimmung der Winkelbereiche der noch verbleibenden Scanpunkte für den auf der rechten Seite aufgenommenen Scan.

$((\hat{x}_t, \hat{y}_t, \hat{\theta}_t)^T, \Sigma_t)$ die Schätzung der Aufnahmeposition von t ist, welche von der Kartenerstellung berechnet wurde, so kann die absolute Positionsschätzung $o = ((\hat{x}_s, \hat{y}_s, \hat{\theta}_s)^T, \Sigma_s)$ von s für die Aktualisierung der Roboterposition wie folgt berechnet werden:

$$\begin{pmatrix} \hat{x}_s \\ \hat{y}_s \\ \hat{\theta}_s \end{pmatrix} = \begin{pmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{\theta}_t \end{pmatrix} + R_3(\hat{\theta}_t) \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{\theta} \end{pmatrix} \tag{4.22}$$

$$\Sigma_s = S \Sigma_t S^T + R_3(\hat{\theta}_t) \Sigma_{match} R_3^T(\hat{\theta}_t) \tag{4.23}$$

$$S = \begin{pmatrix} 1 & 0 & \hat{x} \cos \hat{\theta}_t - \hat{y} \sin \hat{\theta}_t \\ 0 & 1 & \hat{x} \sin \hat{\theta}_t + \hat{y} \cos \hat{\theta}_t \\ 0 & 0 & 1 \end{pmatrix} \tag{4.24}$$

Die Positionsschätzung o kann nun mit der Odometrieschätzung durch Anwenden der Kalman-Filtergleichungen 4.20 und 4.21 für die Aktualisierung der Roboterposition fusioniert werden.

Resultate

Das soeben beschriebene Verfahren wurde erfolgreich in verschiedenen Umgebungen und mit verschiedenen Robotern und Sensoren eingesetzt. Im Projekt *AMOS* des Forschungsinstituts für anwendungsorientierte Wissensverarbeitung (FAW) wurde das Verfahren auf einem fahrerlosen Transportsystem (FTS) mit einem 360° *IBEO* 3D-Laserscanner eingesetzt [Gutmann und Schlegel, 1996; Gutmann, 1996; Corsépius *et al.*, 1997].

Im Nachfolgeprojekt *SMART* findet es auf einem *B21* Roboter mit *SICK* Laserscanner Verwendung [Schlegel, 1998; Schlegel und Wörz, 1998].

An der Universität Freiburg wurde das System für die Navigation eines Bürobotens in der Abteilung *Grundlagen der Künstlichen Intelligenz* eingesetzt [Gutmann und Nebel, 1997].

Das System wurde auch mit aufgezeichneten Daten des Roboters *RHINO* der Universität Bonn getestet. Abbildung 4.6 zeigt eine Karte von Referenzscans des Deutschen Museums in Bonn sowie eine Fahrt von *RHINO* in dieser Umgebung unter Einsatz des hier beschriebenen Lokalisierungsverfahrens.

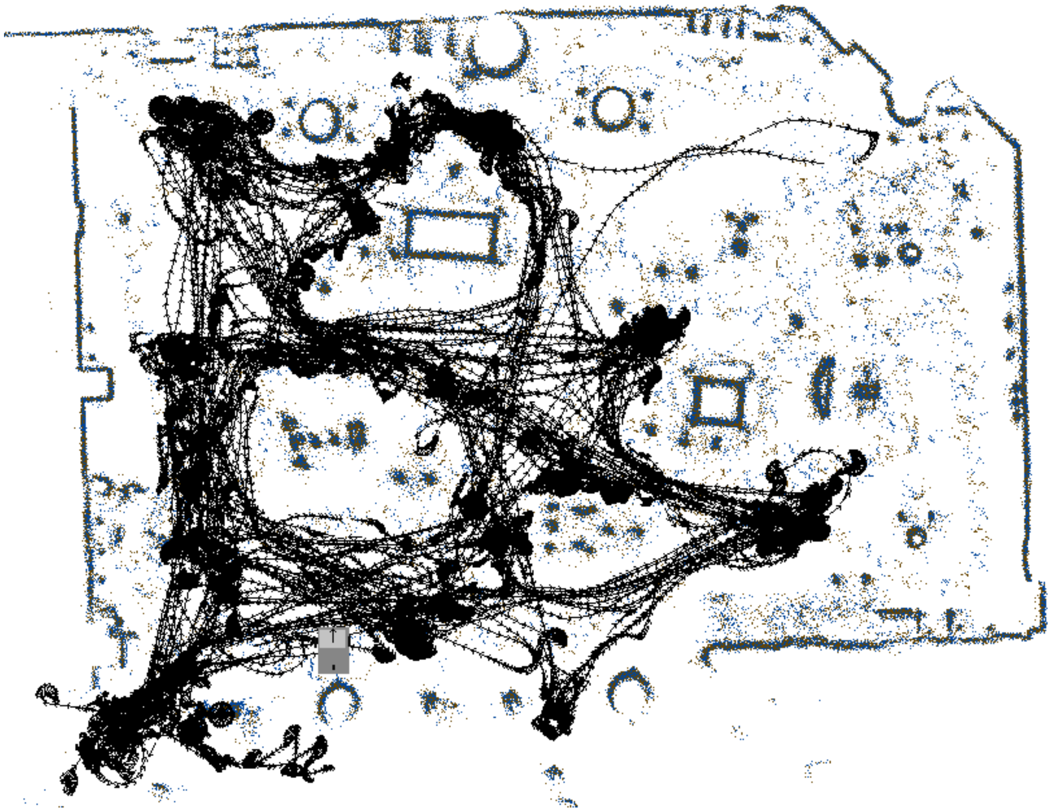


Abbildung 4.6: Karte von Referenzscans des Deutschen Museums in Bonn mit einer Beispiellokalisierung des Roboters *RHINO*.

In dieser Fahrt diente *RHINO* als Tourguide für die Besucher des Museums [Burgard *et al.*, 1998a]. Dabei fuhr *RHINO* zu verschiedenen Ausstellungsgegenständen, um diese dem Besucher näher zu erläutern. Zu manchen Zeiten war das Museum stark besucht, so daß nur noch wenig Sensorinformation für die Lokalisierung zur Verfügung stand. Trotzdem ist das System in der Lage, die Position des Roboters zu verfolgen, was dadurch ersichtlich ist, daß alle Positionen auf freien Flächen liegen (der Roboter also gegen keine Hindernisse gefahren ist) und die berechnete Endposition der tatsächlichen Position am Ende der Fahrt entspricht. Bemerkenswert ist auch, daß im linken unteren und mittleren unteren

Bereich keine Referenzscans vorliegen, so daß hier dem System nur Odometrieinformation zur Verfügung stand. Die Länge dieser Museumsfahrt betrug insgesamt $4\frac{1}{2}$ Stunden.

Es folgen ein paar Statistiken zu diesem Lokalisierungsbeispiel. Von den insgesamt 70890 aufgenommenen Scans wurden 20276 (28.6%) für die Positionsaktualisierung verworfen, da entweder zuviele Scanpunkte durch den Projektionsfilter entfernt wurden oder aber der Fehlerwert $error$ zu groß war. Dies kann z.B. im linken unteren und mittleren unteren Bereich oder durch Blockieren der Sensoren durch zuviele Besucher passieren. In den restlichen 50614 Fällen sind von den translativen Positionsaktualisierungen

- 24199 (47.8%) zwischen $0.0mm$ und $0.2mm$.
- 11371 (22.5%) zwischen $0.2mm$ und $1.0mm$.
- 10006 (19.8%) zwischen $1.0mm$ und $10.0mm$.
- 4878 (9.6%) zwischen $10mm$ und $100mm$.
- 152 (0.3%) zwischen $100mm$ und $300mm$.

Die verbleibenden 8 translativen Aktualisierungen größer $300mm$ sind $301.7mm$, $367.4mm$, $420.2mm$, $430.6mm$, $454.1mm$, $606.9mm$, $687.8mm$ und $746.0mm$.

Die Rotationsaktualisierungen teilen sich wie folgt auf:

- 23167 (45.8%) liegen zwischen $0.0°$ und $0.2°$.
- 19475 (38.5%) liegen zwischen $0.2°$ und $1.0°$.
- 7043 (13.9%) liegen zwischen $1.0°$ und $5.0°$.
- 848 (1.7%) liegen zwischen $5°$ und $10°$.
- 72 (0.1%) liegen zwischen $10°$ und $20°$.

Die verbleibenden 8 Rotationsaktualisierungen größer $20°$ sind $20.8°$, $20.8°$, $22.0°$, $22.6°$, $22.7°$, $22.9°$, $23.7°$ und $24.5°$.

Bei dieser Statistik ist zu beachten, daß die vielen kleinen Positionsaktualisierungen kein Zufall sind, da der Roboter viel Zeit im Stehen verbrang, um den Besuchern die Ausstellungsgegenstände zu erläutern.

Weitere experimentelle Untersuchungen dieses Verfahrens und ein Vergleich mit anderen Selbstlokalisierungsmethoden finden sich im Abschnitt 4.6.

4.5 Markov-Lokalisierung

Die Schlüsselidee bei Markov-Lokalisierung ist, eine diskrete Approximation der Wahrscheinlichkeitsverteilung über alle Positionen der Umgebung zu berechnen. Die Wahrscheinlichkeitsverteilung kann dann direkt nach den Formeln 4.15 und 4.16 fortgeschrieben werden. Markov-Lokalisierung ist nicht auf dichte Sensordaten vergleichende Methoden beschränkt und es existiert eine Vielzahl von Varianten dieser Methode [Burgard *et al.*, 1996; Kaelbling *et al.*, 1996; Nourbakhsh *et al.*, 1995; Simmons und Koenig, 1995; Thrun, 1998].

Folgende Eigenschaften dieser Lokalisierungsmethode wurden experimentell nachgewiesen.

- Markov-Lokalisierung ist in der Lage, die Position des Roboters zu bestimmen, selbst wenn die initiale Position unbekannt ist. Diese Eigenschaft ist essentiell für völlig autonome Roboter, da die initiale Position nicht mehr von Hand eingegeben werden muß, wenn der Roboter eingeschaltet wird oder seine Position verliert.

- Rauschende Sensoren wie beispielsweise Sonarsensoren können für die Lokalisierung verwendet werden.

- Es können Mehrdeutigkeiten repräsentiert werden und es existieren Erweiterungen, welche Mehrdeutigkeiten aktiv auflösen [Burgard *et al.*, 1997].

- Die Zeitkomplexität dieser Verfahren wird von der Modellierung der Wahrscheinlichkeitsverteilung dominiert. Bei einer gitterbasierten Repräsentation beispielsweise hängt die benötigte Rechenzeit von der Dimension des Gitters und der verwendeten Zellgröße ab.

Die in der Literatur veröffentlichten Markov-Lokalisierungsverfahren können aufgrund der verwendeten Diskretisierung unterschieden werden. Während [Kaelbling *et al.*, 1996; Nourbakhsh *et al.*, 1995; Simmons und Koenig, 1995; Thrun, 1998] eine topologische Diskretisierung der Umgebung vornehmen und Landmarkenerkennung für die Lokalisierung des Roboters verwenden, benutzt das Verfahren von Burgard und Fox *et al.* [Burgard *et al.*, 1996; Fox *et al.*, 1998; Fox, 1998] eine feine gitterbasierte Approximation der Verteilung, d.h.

$$p(l) = p(x,y,\theta) \quad = \quad \begin{cases} c_{ijk} & \text{falls} \quad \begin{array}{l} i \cdot d_x \leq x - x_0 < (i+1) \cdot d_x, \\ j \cdot d_y \leq y - y_0 < (j+1) \cdot d_y, \\ k \cdot d_\theta \leq \theta - \theta_0 < (k+1) \cdot d_\theta \end{array} \\ 0 & \text{sonst} \end{cases} \tag{4.25}$$

wobei c_{ijk} die Zellen des Gitters sind, $(x_0, y_0, \theta_0)^T$ der Ursprung des Gitters ist und d_x, d_y und d_θ die Zellgröße bestimmen. Positionen außerhalb des Positionsgitters erhalten eine Wahrscheinlichkeit von 0, was sich aus der Annahme begründet, daß sich der Roboter nie außerhalb seiner Umgebung befindet.

Um mit dem riesigen Zustandsraum effizient umgehen zu können, wurden zahlreiche Optimierungen vorgenommen. In der Praxis wird für gewöhnlich nur ein kleiner Bereich um den Roboter herum während der Lokalisierung aktualisiert.

Die Umgebungskarte, die diesem Verfahren zugrunde liegt, ist eine metrische Karte, die entweder aus einer von Hand erstellten und aus Linienzügen bestehenden CAD-Karte oder aus einem Belegtheitsgitter [Moravec und Elfes, 1985] besteht. Die Karte wird für die Vorhersage von Sensormessungen benutzt, indem für jede Zelle des Zustandraumes die erwartete Sensormessung bestimmt wird. Die Ähnlichkeit der vorhergesagten Messung zur tatsächlichen Messung gibt ein Maß für die Beobachtungswahrscheinlichkeit $p(o_n|l)$.

Eine wichtige Eigenschaft von Markov-Lokalisierung ist die Fähigkeit den Roboter global innerhalb der Umgebung zu lokalisieren. Abbildung 4.7 zeigt den Grundriß einer 27 auf 20 Meter großen Sektion des Informatikinstituts der Universität Bonn. Beide Bilder zeigen die Wahrscheinlichkeitsverteilungen der Roboterposition projiziert auf die xy-Ebene einer globalen Lokalisierung des Roboters. Helle Werte markieren eine geringe, dunkle Werte eine hohe Positionswahrscheinlichkeit. Rote Werte markieren Peaks.

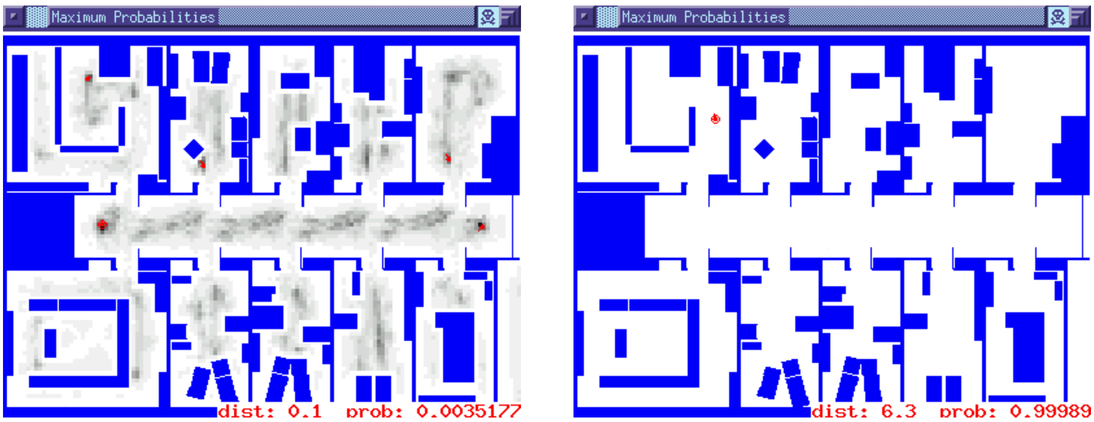


Abbildung 4.7: Globale Positionsbestimmung mittels Markov-Lokalisierung im Informatikinstitut der Universität Bonn. Links Zustand nach Integration von zwei Sonarscans. Rechts nach einer Fahrt von 6.3 Metern und Integration von 6 Sonarscans.

Zu Beginn wurde dem Roboter keinerlei Information über seine Position gegeben, d.h. die Positionsverteilung war eine Gleichverteilung über den gesamten Zustandsraum. Im linken Bild von Abbildung 4.7 ist der Zustand nach Integration von zwei Sonarscans mit jeweils 24 Ultraschallsensoren dargestellt. Nach einer Fahrt von 6.3 Metern und der Integration von 6 Sonarscans ist sich der Roboter absolut sicher über seine Position (rechtes Bild in Abbildung 4.7).

Das soeben beschriebene Markov-Lokalisierungsverfahren wurde von Wissenschaftlern der Universität Bonn entwickelt und in einer Vielzahl von Roboteranwendungen erfolgreich eingesetzt [Buhmann *et al.*, 1995; Burgard *et al.*, 1998a; Thrun *et al.*, 1999]. Im Abschnitt 4.6 werden experimentelle Ergebnisse zu dieser gitterbasierten Markov-Lokalisierung und ein Vergleich mit Scan-Matching-Verfahren präsentiert.

Da Genauigkeit und Rechenbedarf von gitterbasierter Markov-Lokalisierung direkt von der verwendeten Zellgröße abhängen, wurde auch eine dynamische Variante entwickelt, welche die Zellgröße automatisch der aktuellen Positionsverteilung anpaßt [Burgard *et al.*, 1998b]. Dies hat den Vorteil, daß zur Bestimmung der globalen Roboterposition automatisch ein grobes Gitter und zum Verfolgen der Position automatisch ein feines Gitter verwendet wird.

Eine erst kürzlich vorgestellte Variante von Markov-Lokalisierung verwendet anstatt eines Positionsgitters Partikel zur Repräsentation der Positionswahrscheinlichkeiten [Dellaert *et al.*, 1999; Fox *et al.*, 1999]. Dieses als Monte-Carlo-Lokalisierung benannte Verfahren verwaltet eine Menge von Partikeln

$$s_i \;=\; ((x_i, y_i, \theta_i)^T, p_i),$$

welche jeweils die Wahrscheinlichkeit p_i ausdrücken, daß sich der Roboter an Position $(x_i, y_i, \theta_i)^T$ befindet. Das Verfahren ist sehr effizient (Vorhersage und Aktualisierung der Roboterposition können in $O(n)$ in der Anzahl Partikel realisiert werden), und kann den Roboter präzise lokalisieren. Problematisch ist, wieviele Partikel für die Lokalisierung benötigt werden und es existieren Methoden diese Anzahl dynamisch an die augenblickliche Positionsverteilung anzupassen [Fox *et al.*, 1999].

4.6 Vergleich von Lokalisierungsmethoden

Neben Scan-Matching-Lokalisierung und Markov-Lokalisierung existieren in der Literatur noch eine Vielzahl weiterer dichte Sensordaten vergleichende Lokalisierungsverfahren (z.B. [Rencken, 1993; Schultz und Adams, 1996; Borthwick und Durrant-Whyte, 1994; Leonard und Durrant-Whyte, 1991; Crowley, 1989; Holenstein *et al.*, 1992; Feng *et al.*, 1996]). Jedes dieser Verfahren wurde in einer gewissen Umgebung und mit einem gewissen Robotersystem getestet. Jedoch liegen keinerlei Informationen über die Anwendbarkeit der Verfahren in anderen Umgebungen und mit anderen Robotern vor.

Aus diesem Grunde sollen in diesem Abschnitt zwei Verfahren systematisch miteinander verglichen werden, um herauszufinden, wie sich die Verfahren unter verschiedenen Einsatzbedingungen und mit verschiedenem Sensorrauschen verhalten. Ziel ist es, die verschiedenen Stärken und Schwächen der einzelnen Verfahren zu ermitteln, um hieraus Richtlinien abzuleiten, in welcher Umgebung und mit welchen Sensoren ein Robotersystem sich robust und genau lokalisieren kann.

Folgende Punkte sollen dabei genauer untersucht werden:

1. Unter welchen Umständen gibt es katastrophale Fehler, d.h. der Roboter verliert seine Position?

2. Wie genau lokalisiert das jeweilige Verfahren den Roboter?

3. Wie gut kann das Verfahren mit Mehrdeutigkeiten umgehen im Falle unzureichender Sensorinformation für die eindeutige Lokalisierung des Roboters?

Für den Vergleich soll ein Kalman-Filter basiertes Verfahren und eine Markov-Lokalisierungsmethode herangezogen werden. Als Kalman-Filter basiertes Verfahren bieten sich die in Abschnitt 4.4 vorgestellten Verfahren an. Da hier immer ein ganzer Scan mit einem Referenzmodell überdeckt wird, ist zu erwarten, daß dies zu zuverlässigeren Ergebnissen führt, als Methoden, welche die Lokalisierung nur auf einzelne Merkmale basieren und keine binären Bedingungen an Merkmale stellen [Vestli *et al.*, 1994; Holenstein *et al.*, 1992; Crowley, 1989; Borthwick und Durrant-Whyte, 1994].

Für Markov-Lokalisierung wurde die gitterbasierte Variante aus Abschnitt 4.5 gewählt, da diese universal ist und sehr viel Erfahrung in der Realisierung dieser Variante steckt.

4.6.1 Lokalisierungsexperimente

Um die beiden Selbstlokalisierungsmethoden miteinander zu vergleichen, wurden zahlreiche Experimente mit dem mobilen Roboter *RHINO* [Buhmann *et al.*, 1995; Thrun *et al.*, 1998a]. (siehe Abbildung 4.8) der Universität Bonn durchgeführt. Als Testumgebung diente einerseits die büroartige Umgebung des Informatikinstituts der Universität Bonn als auch eine ziemlich unstrukturierte Umgebung im Deutschen Museum Bonn, in welcher *RHINO* als Museumsführer diente [Burgard *et al.*, 1998a] (siehe Abbildung 4.16). *RHINO* ist ein *RWI B21* Roboter mit 2 *SICK*-Laserscannern, welche zusammen einen Bereich von 360° abdecken, und einem Ring von 24 Ultraschallsensoren, die einen Öffnungswinkel von jeweils 15° besitzen.

Für die Untersuchung der beiden Lokalisierungsmethoden wurden aufgenommene Odometrie- und Sensordaten mit Rauschen versehen. Für gewöhnlich gibt es mehrere verschiedene Arten von Rauschen, die ein Roboter in einer realen Welt wahrnimmt. Auf der einen Seite gibt es ein typisches Gauß'sches Rauschen in Odometrie und abstandsmessenden Sensoren, welche von den inhärenten Ungenauigkeiten der Sensoren stammt. Auf der anderen Seite gibt es Fehler, die sich nicht durch Gauß'sche Funktionen modellieren lassen, beispielsweise Fehler, die durch nicht-modellierte Hindernisse, durch Sensorübersprechen (*cross-talk*) oder durch die Kollision des Roboter mit einem Hindernis und dadurch verursachtem Positionsversatz entstehen.

Odometriefehler, die von schlecht haftenden Rädern, unebenen Böden oder verschiedenen Nutzlasten kommen, werden durch die folgenden drei Parameter

Abbildung 4.8: Mobiler Roboter *RHINO* der Universität Bonn.

charakterisiert (siehe Abbildung 4.9 links)

Entfernungsrauschen: Der Entfernungsfehler $\Delta_\delta(\delta)$ nachdem der Roboter die Strecke δ zurückgelegt hat.

Rotationsrauschen: Der Orientierungsfehler $\Delta_\alpha(\alpha)$ nachdem sich der Roboter um den Winkel α gedreht hat.

Driftrauschen: Der Orientierungsfehler $\Delta_\alpha(\delta)$ nachdem der Roboter die Strecke δ zurückgelegt hat.

Eine andere Quelle für Odometrierauschen, die jedoch wesentlich seltener auftritt, sind Situationen, in denen der Roboter mit einem Hindernis zusammenstößt. Diese abrupten Fehler können durch die folgenden Parameter charakterisiert werden (siehe Abbildung 4.9 rechts).

Stoßfehler: Der Positionsfehler x, y und α, der zur Odometrieposition hinzuaddiert wird.

Stoßhäufigkeit: Die Wahrscheinlichkeit, daß ein Zusammenstoß mit einem Hindernis auftritt.

Schließlich wird noch eine weitere Art von Rauschen betrachtet, die ensteht, wenn ein entfernungsmessender Sensor ungenaue oder unbrauchbare Daten liefert. Dies kann durch eine ungenaue Umgebungskarte, fehlerhafte Sensoren oder

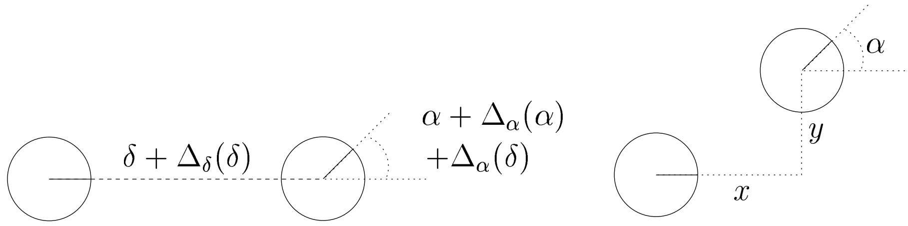


Abbildung 4.9: Verschiedene Quellen, die Odometriedaten verrauschen. Links Hinzufügen von Gauß'schen Rauschen $\langle \Delta_\delta(\delta), \Delta_\alpha(\alpha), \Delta_\alpha(\delta) \rangle$. Rechts Hinzufügen von Stoßrauschen $\langle x, y, \alpha \rangle$.

durch dynamische Hindernisse wie umherlaufende Personen verursacht werden. Da dieser Typ von Rauschen schwer zu charakterisieren ist, werden in den folgenden Experimenten Daten eines Roboterlaufs in einer stark besuchten Umgebung verwendet.

4.6.2 Ergebnisse in einer Büroumgebung

Abbildung 4.10 zeigt eine Büroumgebung im Informatikinstitut der Universität Bonn, in welcher Selbstlokalisierungsexperimente durchgeführt wurden. Die Umgebung besteht aus einem Korridor und zehn verschiedenen Büroräumen. Für die Experimente wurde der Roboter auf der linken Seite des Korridors gestartet und in alle Räume bewegt. An 22 verschiedenen Orten wurde die Position des Roboters mit einem Maßband von Hand gemessen und später durch Verwenden von Scan-Matching genauer ermittelt. Diese 22 Positionen werden im folgenden als Referenzpositionen für die Bestimmung von Genauigkeit und Robustheit benutzt.

Für die Experimente wurden Sensor- und Odometriedaten dieser Roboterfahrt aufgezeichnet und verschiedene Arten von Rauschen den Odometriedaten hinzugefügt. Für das Verrauschen der Daten wurde ein Zufallszahlengenerator verwendet und für jeden Satz von Parametern 26 verschiedene Startwerte für die Initialisierung verwendet. Wann immer der Roboter an einer der Referenzpositionen ankam, was durch Vergleich von Zeitstempeln in den aufgezeichneten Daten festgestellt werden kann, wurde der Abstand der vom System geschätzten Position zur tatsächlichen Position gemessen. Abbildung 4.11 zeigt die Trajektorie, welche vom Roboter tatsächlich abgefahren wurde, und eine typische Trajektorie nach starkem Verrauschen der Odometriedaten. In diesem Beispiel wurden für das Verrauschen die Parameter $\langle 400, 20, 20 \rangle$ benutzt, welche die Standardabweichungen von $\langle \Delta_\delta(\delta), \Delta_\alpha(\alpha), \Delta_\alpha(\delta) \rangle$ mit den Einheiten $\sqrt{mm^2/m}$, $\sqrt{Grad^2/360°}$ und $\sqrt{Grad^2/m}$ angeben.

Für die Experimente wurde die Verfahren Scan-Matching mit Linienmodell und Scan-Matching mit Referenzscans, sowie gitterbasierte Markov-Lokalisierung

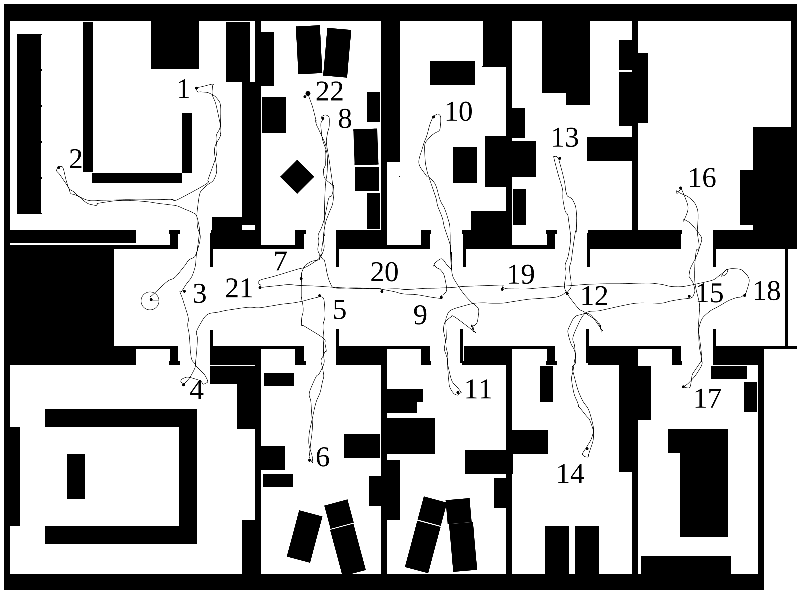


Abbildung 4.10: Büroumgebung im Informatikinstitut der Universität Bonn ($27 \times 20m^2$) mit abgefahrener Robotertrajektorie und 22 Referenzpositionen.

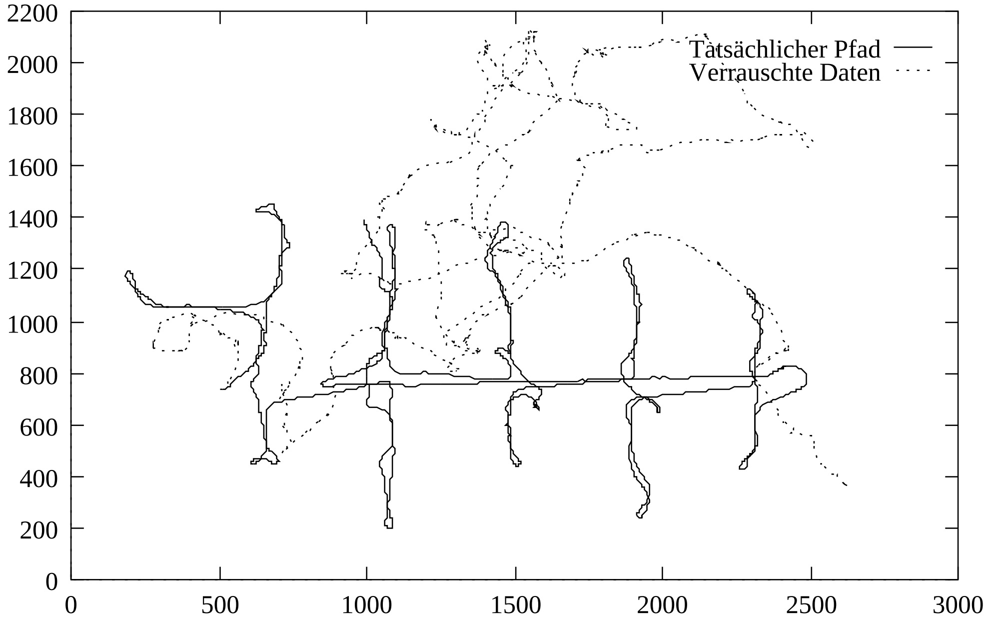


Abbildung 4.11: Tatsächlich gefahrene Trajektorie und typische Trajektorie, wie sie durch starkes Verrauschen mit den Parametern $\langle 400, 20, 20 \rangle$ entsteht.

mit Sonardaten und Laserdaten verwendet. Den Verfahren wurde jeweils mitgeteilt, mit welchen Parametern die Odometriedaten verrauscht wurden, so daß diese sich entsprechend anpassen konnten. Zum Beispiel reichte die Zellgröße bei Markov-Lokalisierung von $15cm \times 15cm \times 3°$ bei niedriger Rauschstufe bis zu $30cm \times 30cm \times 10°$ bei der höchsten Rauschstufe. Für jede Methode wurde der mittlere Abstand der geschätzten Positionen zu den jeweils zugehörigen Referenzpositionen ermittelt. Die Abstände wurden über alle Positionen gemittelt, bei denen der Roboter seine Position nicht verloren hatte. Für die Entscheidung, ob der Roboter seine Position verloren hat oder nicht, wurde eine Schranke von einem Meter benutzt, d.h. der Roboter verliert seine Position, wenn der Abstand der geschätzten Position zur Referenzposition größer einem Meter ist.

Abbildung 4.12 zeigt den mittleren Abstand zu den Referenzpositionen für verschiedene Stufen von Gauß'schen Rauschen. Die Werte auf der x-Achse entsprechen den Standardabweichungen der Parameter $\langle \Delta_\delta(\delta), \Delta_\alpha(\alpha), \Delta_\alpha(\delta) \rangle$. Die Fehlerbalken geben in dieser und allen folgenden Abbildungen das 95%-Konfidenzintervall des Mittelwertes an, d.h. der wahre Mittelwert befindet sich mit hoher Wahrscheinlichkeit in diesem Intervall. Wie zu sehen ist, ist Scan-Matching-Lokalisierung sehr viel genauer als Markov-Lokalisierung, sofern die Roboterposition nicht verloren wurde.

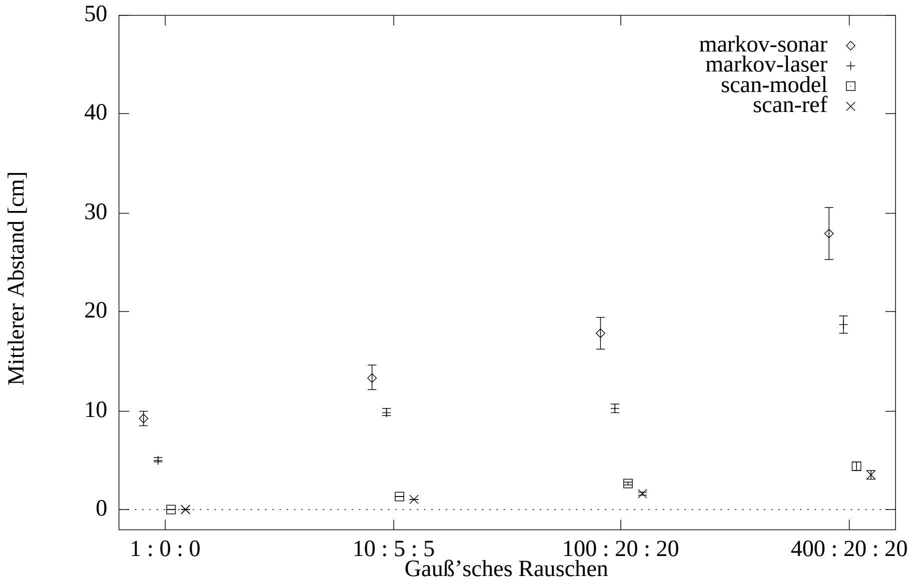


Abbildung 4.12: Abstände zu Referenzpositionen in der Büroumgebung für verschiedene Stufen von Gauß'schen Odometrierauschen.

Betrachtet man jedoch die Robustheit der Verfahren, d.h. die Anzahl der Positionen, an denen der Roboter seine Position nicht verloren hat, so ergibt sich ein anderes Verhalten. Abbildung 4.13 zeigt die durchschnittliche Anzahl der Positionen, an denen der Roboter seine Position verloren hat, d.h. der Abstand zur Referenzposition größer einem Meter ist. Für geringes Rauschen können alle Verfahren zu allen betrachteten Zeitpunkten die Roboterposition bestimmen. Bei stärkerem Rauschen verliert Markov-Lokalisierung jedoch sehr viel weniger oft den Roboter als Scan-Matching. Selbst bei dem stärksten verwendeten Rauschen verliert Markov-Lokalisierung mit Laserdaten nur in 0.3% der Fälle die Roboterposition, was signifikant besser als die Resultate von Scan-Matching ist. Dies liegt daran, daß Markov-Lokalisierung in der Lage ist, den Roboter zu relokalisieren, falls die Position einmal verloren geht. Scan-Matching mit Kalman-Filterung kann in diesem Fall die Position jedoch nur per Zufall wiederfinden. Eine interessantes Fakt ist, daß Markov-Lokalisierung mit Sonardaten in diesem Experiment eine ähnliche Robustheit wie Scan-Matching aufweist.

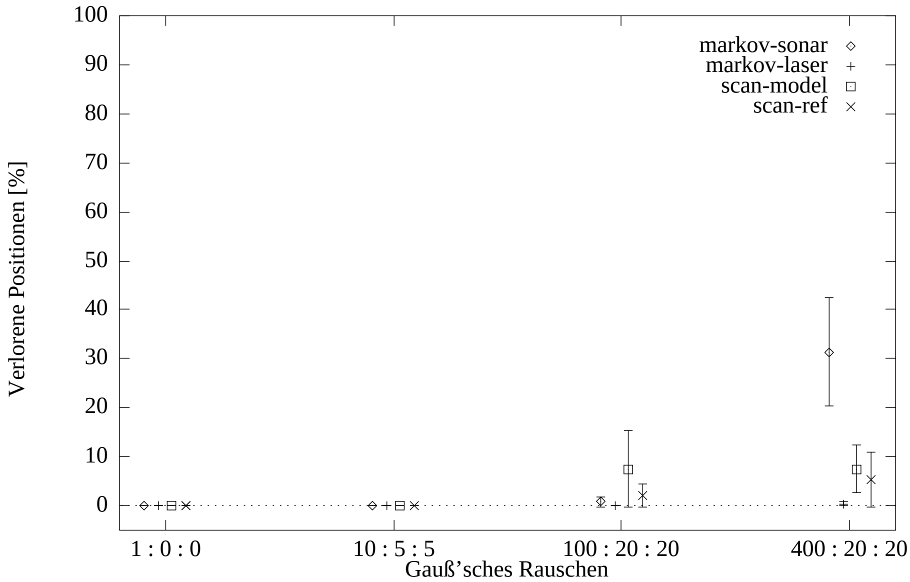


Abbildung 4.13: Prozentuale Anzahl der Positionen, an denen die Position des Roboters verloren war, für verschiedene Stufen von Gauß'schen Rauschen.

Auf gleiche Weise wurde das Verhalten der Verfahren bei Hinzunahme von Stoßrauschen untersucht. Abbildung 4.14 zeigt den mittleren Abstand zu den Referenzpositionen bei verschiedenen Stufen von Stoßrauschen. Werte auf der x-Achse geben die verwendeten Standardabweichungen des Stoßfehlers $\langle x, y, \alpha \rangle$ an. Als Stoßhäufigkeit p für einen Zusammenstoß mit einem Hindernis wurde $p = 0.05$ pro Meter gesetzt. Zusätzlich zum Stoßrauschen wurde ein geringes Gauß'sches Rauschen der Stärke $\langle 10, 5, 5 \rangle$ hinzuaddiert. Wie in Abbildung 4.14 zu sehen ist, ist Scan-Matching wieder sehr viel genauer als Markov-Lokalisierung.

Abbildung 4.15 zeigt die durchschnittliche Anzahl von Referenzpositionen, an denen die Roboterposition von den verschiedenen Verfahren als verloren angesehen wurde. Die Scan-Matching-Methoden können hier nur bei geringem Stoßrauschen die Roboterposition halten, was hauptsächlich darauf rückzuführen ist, daß Stoßrauschen die Annahme von Gauß'schen Odometrierauschen sehr schlecht

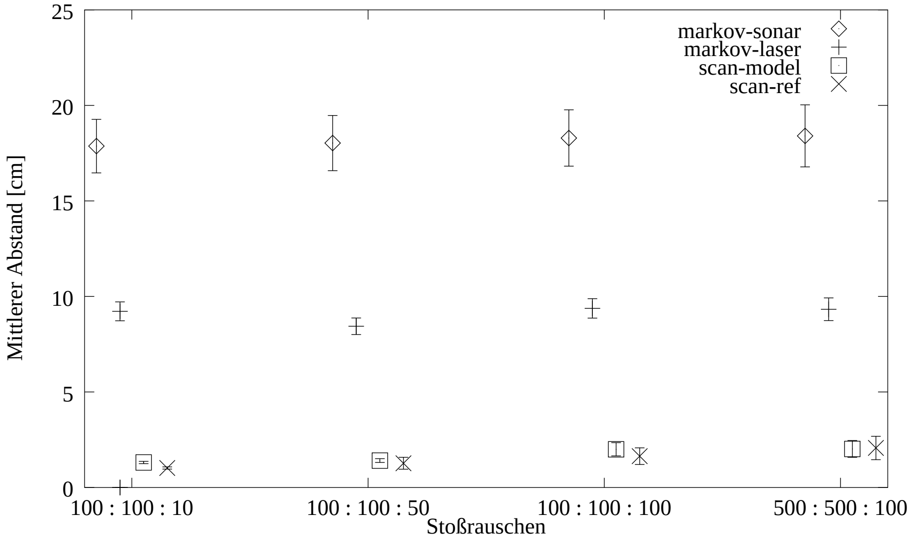


Abbildung 4.14: Abstände zu Referenzpositionen in der Büroumgebung für verschiedene Stufen von Stoßrauschen.

modelliert. Markov-Lokalisierung ist hier wesentlich robuster, selbst wenn nur Sonardaten verwendet werden.

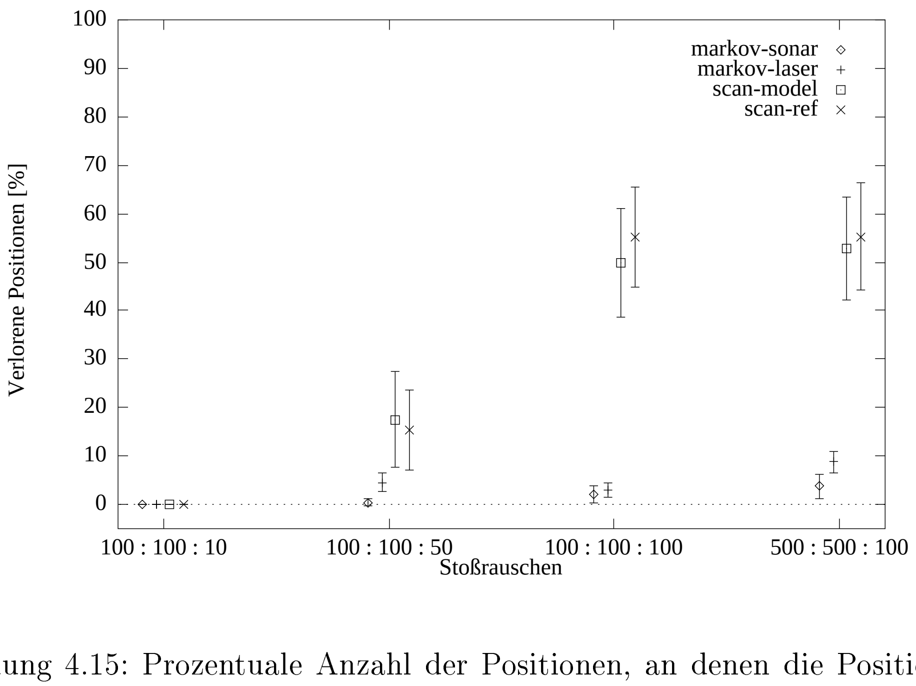


Abbildung 4.15: Prozentuale Anzahl der Positionen, an denen die Position des Roboters verloren war, für verschiedene Stufen von Stoßrauschen.

4.6.3 Ergebnisse in einer unstrukturierten Umgebung

Die Selbstlokalisierungsverfahren wurden auch in einer unstrukturierten Umgebung untersucht, um die Sensitivität zur verwendeten Umgebung und zu Kartenfehlern zu testen. Dies ist wichtig, da die Anwendung von mobilen Robotern nicht nur auf strukturierte Büroumgebungen beschränkt ist und Umgebungen, in denen der Roboter agiert, nicht statisch sind oder teilweise nur ungenau modelliert werden können.

Die Daten dieser Experimente wurden während des Einsatzes von *RHINO* als Museumsführer im Deutschen Museum in Bonn [Burgard *et al.*, 1998a] aufgenommen. In dieser Umgebung war der Roboter oftmals komplett umzingelt von Besuchern, so daß nahezu alle Sensordaten kürzere Abstände als erwartet lieferten. Abbildung 4.16 zeigt eine typische Situation, in der die Sensoren von *RHINO* durch umherstehende Personen blockiert waren.

Abbildung 4.16: Typische Situation, in der viele Besucher um den Roboter herumstehen und die Sensoren falsche Meßwerte liefern.

Für diese Experimente wurden zwei verschiedene Datensätze benutzt, deren Trajektorien in Abbildung 4.17 zu sehen sind. Die gestrichelte Trajektorie stammt von einem Roboterlauf in leerem Museum, die durchgezogene wurde in normalen Betrieb aufgenommen, d.h. Besucher konnten die Sicht der Sensoren blockieren. Abbildung 4.18 zeigt einen typischen Scan, wie er von den Laserscannern des Roboters in einer stark mit Besuchern gefüllten Situation erfaßt wird.

Da bei der Fahrt des Roboters im Museum keine Referenzpositionen vermessen wurden, wurden als Referenz die Positionsergebnisse der Scan-Matching-Lokalisierung mit den Rohdaten verwendet, da Scan-Matching nach ersten Untersuchungen wieder genauer als Markov-Lokalisierung war.

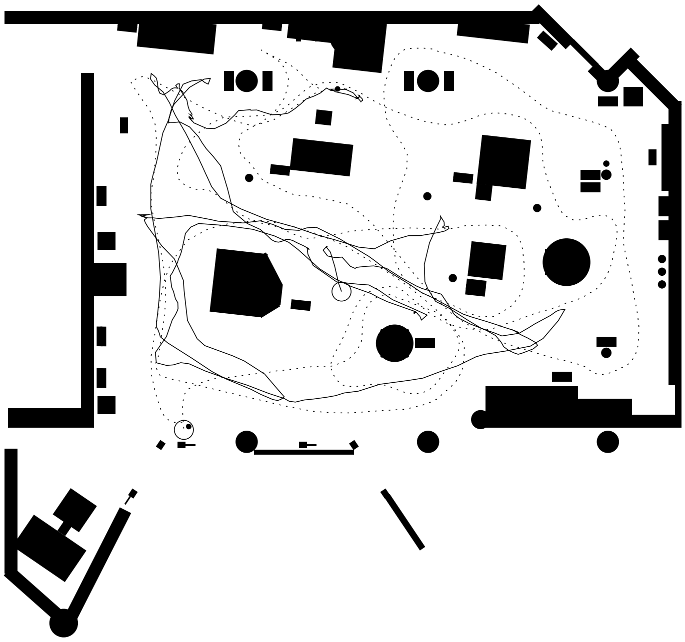


Abbildung 4.17: Trajektorien der Roboterläufe für die Selbstlokalisierungsexperimente. Die gestrichelte Linie zeigt die Trajektorie in leerem Museum, die durchgezogene in gefülltem Museum.

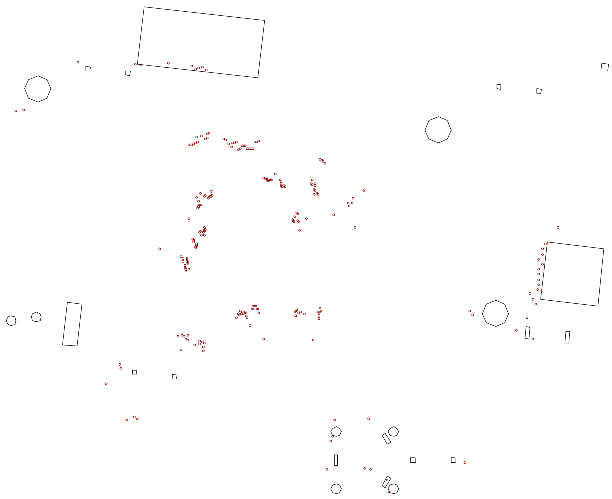


Abbildung 4.18: Typischer Scan, der viele zu kurze Messungen enthält, wie er in der Situation aus Abbildung 4.16 enstehen kann. Die Rechtecke und Kreise beschreiben Objekte der Umgebungskarte während Punkte die Messungen des Laserscanners sind.

Wie in der Büroumgebung wurden Genauigkeit und Robustheit der Verfahren bei Hinzunahme von Gauß'schen Rauschen bestimmt. Für die Genauigkeit der Verfahren ergab sich dasselbe Ergebnis wie für die Büroumgebung, d.h. Scan-Matching ist wesentlich genauer als Markov-Lokalisierung, vorausgesetzt der Roboter verliert seine Position nicht.

Abbildungen 4.19 zeigt für das leere Museum die relative Häufigkeit, daß ein Verfahren die Roboterposition verliert. Wiederum ist Markov-Lokalisierung öfters als Scan-Matching in der Lage den Roboter zu lokalisieren.

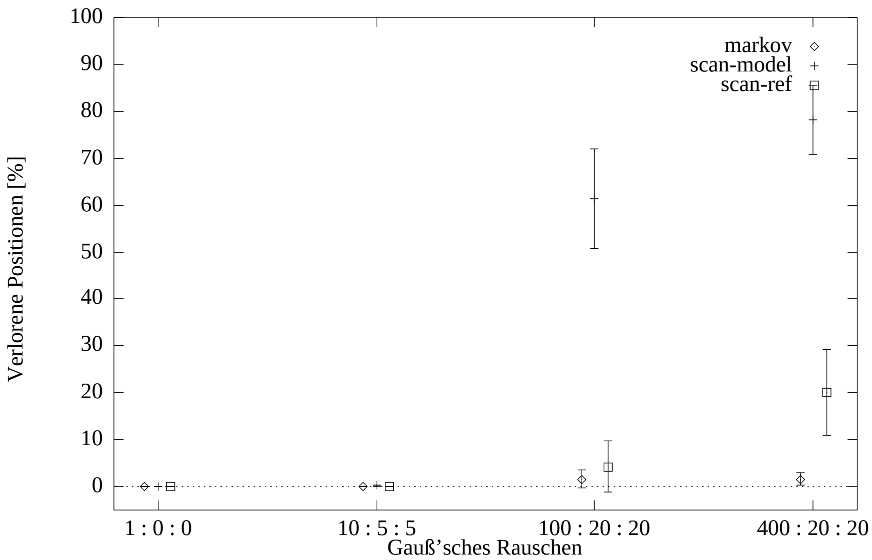


Abbildung 4.19: Anzahl verlorener Positionen für verschiedene Stufen von Gauß'-schen Rauschen in leerem Museum.

Schließlich zeigt Abbildung 4.20 das Verhalten, wenn viele Besucher im Museum sind. Hier haben beide Verfahren Probleme, die Roboterposition selbst bei geringem Odometrierauschen nicht zu verlieren. Markov-Lokalisierung ist hier jedoch wieder robuster als Scan-Matching.

4.6.4 Diskussion

In diesem Abschnitt wurden zwei verschiedene, populäre Methoden für die Selbstlokalisierung eines mobilen Roboters empirisch miteinander verglichen: Markov-Lokalisierung, das beliebige Wahrscheinlichkeitsverteilungen mittels einem Positionsgitter repräsentieren kann, und Scan-Matching mit Kalman-Filterung, das Positionen mit Normalverteilungen modelliert. Während andere Arbeiten [Shaffer *et al.*, 1992; Schiele und Crowley, 1994; Gutmann und Schlegel, 1996; Bengtsson und Baerveldt, 1999] hauptsächlich verschiedene Matching-Verfahren für Kal-

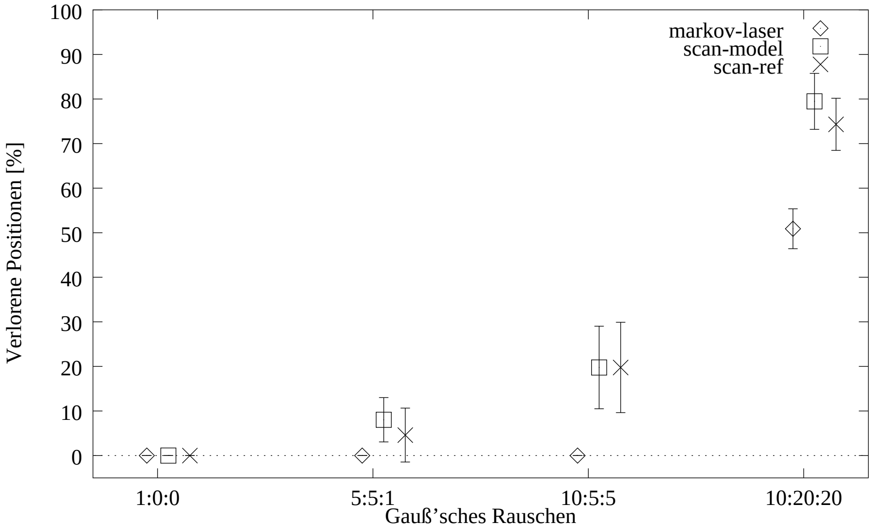


Abbildung 4.20: Anzahl verlorener Positionen für verschiedene Stufen von Gauß'-schen Rauschen in gefülltem Museum.

man-Filterung miteinander vergleichen, wurden hier zwei verschiedene Lokalisierungsansätze miteinander verglichen.

Obwohl beide Verfahren auf den gleichen Bayes'schen Formeln aufbauen, unterscheiden sie sich durch die Wahl der Repräsentation der Wahrscheinlichkeitsverteilung signifikant im Verhalten. Die Ergebnisse der empirischen Evaluation können wie folgt grob zusammengefaßt werden.

- Sofern genügend Sensorinformation zur Verfügung steht ist Scan-Matching-Lokalisierung wesentlich genauer als Markov-Lokalisierung.

- Markov-Lokalisierung ist robuster als Scan-Matching, falls Odometrie- oder Sensordaten unzuverlässig sind.

Die experimentellen Ergebnisse legen es nahe, die beiden untersuchten Verfahren zu kombinieren, so daß die Robustheit von Markov-Lokalisierung und die Effizienz und Genauigkeit von Scan-Matching zusammengeführt werden. Dies könnte dadurch realisiert werden, daß Markov-Lokalisierung nur ein sehr grobes Gitter benutzt, um die Ergebnisse von Scan-Matching-Lokalisierung zu überwachen. Zu Beginn wird Markov-Lokalisierung benutzt, den Roboter global zu lokalisieren. Sobald die Position des Roboters ungefähr bekannt ist wird Scan-Matching hinzugeschaltet, um den Roboter exakt zu lokalisieren. Verliert Scan-Matching die Position, was durch Vergleich der Positionsschätzung mit der Schätzung von Markov-Lokalisierung bestimmt werden kann, so übernimmt Markov-Lokalisierung die Positionsbestimmung und setzt Scan-Matching wieder auf, sobald sich eine eindeutige globale Position wieder ergibt.

Kapitel 5

Kartenerstellung

In diesem Kapitel wird das Problem der Kartenerstellung näher untersucht. Die Aufgabe ist, von einer unbekannten Umgebung eine für das Robotersystem nutzbare Umgebungskarte zu erstellen. Umgebungskarten werden aus folgenden Gründen für die Navigation eines Roboters benötigt:

1. Die Umgebungskarte kann für die Selbstlokalisierung des Roboters durch Einsatz eines der im vorigen Kapitel beschriebenen Verfahren benutzt werden.

2. Für den Roboter wichtige Positionen wie Postfächer, Kaffeemaschine, etc. können in die Karte eingetragen werden.

3. Die Umgebungskarte wird für die Planung von Pfaden von Start- zu Zielorten benötigt, z.B. im Pfadplanungsverfahren, das im nächsten Kapitel beschrieben wird.

Für viele Robotersysteme wurde eine für die Navigation nutzbare Umgebungskarte durch Vermessen der Umgebung mit einem Maßband und Eintragen der einzelnen Längen in ein CAD-Modell von Hand erstellt. Beispielsweise wurde die Umgebungskarte für *RHINO* im Deutschen Museum in Bonn auf diese Weise gewonnen [Burgard *et al.*, 1998a]. Nach Aussage von Wolfram Burgard wurde für die Erstellung dieser Umgebungskarte in etwa eine Woche benötigt.

Die Umgebungskarte kann jedoch auch aus den Sensordaten des Roboters in einer Explorationsfahrt gewonnen werden. Die Aufgabe hierbei ist, den Roboter in alle Bereiche der Umgebung zu bewegen und aus den anfallenden Odometrie- und Sensordaten eine Umgebungskarte zu erstellen. Das Problem hierbei ist, daß für die Erstellung der Karte die Roboterpositionen bekannt sein müssen, da sonst nicht klar ist, an welche Stelle die aufgenommenen Sensordaten einzutragen sind. Um die Roboterpositionen aber bestimmen zu können, wird wiederum eine Umgebungskarte benötigt. Für die Lösung dieses konkurrierenden Problems wird für gewöhnlich ein rekursiver Ansatz gewählt, d.h. es wird mit einer leeren Karte und beliebiger Position gestartet, Sensordaten werden aufgrund der aktuellen Positionsschätzung in die Karte integriert, und die Position wird nach Bewegen des Roboters aufgrund der bisher erstellten Karte aktualisiert.

Zahlreiche Methoden zur Lösung dieses Problems wurden in der Vergangenheit vorgeschlagen. Manche Methoden stellen Annahmen an die Umgebung, wie z.B. daß die Umgebung rechtwinklig ist. Dadurch kann das Problem erheblich vereinfacht werden und es existieren Lösungen, die erfolgreich für die Navigation von Robotern eingesetzt wurden [Buhmann *et al.*, 1995; Thrun *et al.*, 1998a; Konolige *et al.*, 1997; Kunz *et al.*, 1997]. In dieser Arbeit sollen solche Annahmen vermieden werden, d.h es werden nur solche Verfahren untersucht, die in fast beliebigen Umgebungen eingesetzt werden können.

Bei der Erstellung von Karten in Innenräumen treten die Begriffe der *topologisch korrekten Karten* und der *konsistenten Karten* auf. Diese Begriffe werden meist im „gewöhnlichen“ Sinn benutzt und sollen in dieser Arbeit durch die beiden folgenden Beschreibungen konkretisiert werden.

Topologisch korrekte Karten: Eine Karte heißt topologisch korrekt, wenn die in der Karte enthaltenen Beobachtungen genau dann miteinander identifizierbar sind, wenn die zugehörigen Objekte in der realen Welt identisch sind.

Konsistente Karten: Eine Karte heißt konsistent, wenn sie einer realen Umgebung entsprechen könnte, d.h. es darf z.B. keine Wand mehrfach in der Karte auftauchen. Viele Verfahren zur Kartenerstellung versuchen die Beobachtungen nur minimal zu ändern, um aus einer inkonsistenten eine konsistente Karte zu berechnen.

Beispielsweise ist die in Abbildung 5.1 abgebildetet Karte topologisch korrekt aber nicht konsistent. Ziel eines vollständigen Verfahrens zur Erstellung von Karten ist es, topologisch korrekte und konsistente Karten zu erstellen.

Im folgenden wird zunächst das Problem von Zyklen in der Umgebung vorgestellt. Ein einfacher, inkrementeller Ansatz, welcher die Umgebungskarte nur lokal aktualisiert, kann zwar effizient implementiert werden, im allgemeinen aber nicht in Umgebungen eingesetzt werden, die Zyklen enthalten.

Weiterhin werden eine ganze Reihe von existierenden Verfahren zusammengefaßt, die sich mit dem Schließen von Zyklen näher auseinandergesetzt haben, jedoch entweder suboptimale Lösungen liefern (z.B. ungenaue Karten), oder inhärent Probleme bei ungenauer Sensorik (z.B. schlechter Odometrie) aufweisen.

Anschließend werden zwei in letzter Zeit erfolgreich eingesetzte Verfahren zur Erstellung von Karten in großen, zyklischen Umgebungen vorgestellt. Lu und Milios [1997a] formulieren die Kartenerstellung als ein Optimierungsproblem und bestimmen mit Hilfe von Scan-Matching konsistente Positionsschätzungen der Aufnahmepositionen. Das Verfahren liefert hochgenaue Karten, kann aber nur verwendet werden, wenn die Daten bereits topologisch korrekt sind. Dies ist jedoch in großen, zyklischen Umgebungen im allgemeinen nicht gegeben.

Thrun *et al.* [1998b] verwenden Erwartung und Maximierung für die Kartenerstellung. Hierdurch können aus Rohdaten topologisch korrekte Karten erzeugt werden. Die Karten besitzen jedoch eine niedrige Genauigkeit, welche in vielen Fällen für die robuste Navigation eines Roboters nicht ausreichen.

In dieser Arbeit wird eine Kombination der Verfahren von Thrun *et al.* [1998b] und Lu und Milios [1997a] vorgeschlagen, welche in der Lage ist, aus Rohdaten topologisch korrekte und konsistente Karten mit hoher Genauigkeit zu erstellen.

Das Verfahren benötigt jedoch sehr viel Rechenzeit, kann nicht inkrementell eingesetzt werden (alle Daten müssen zuerst gesammelt werden) und der Benutzer muß zusätzliche Eingaben (Landmarken), die bisher nicht vom Roboter selbst erkannt werden, dem System mitteilen. Ein nahezu ideales Verfahren müßte jedoch schnell, inkrementell, autonom und genau sein. In dieser Arbeit wird ein neues Verfahren beschrieben [Gutmann und Konolige, 1999], welches Scan-Matching, Kartenkorrelation und konsistente Positionsschätzung verwendet und das nach Wissensstand des Autors das erste inkrementelle Verfahren ist, das autonom und in Echtzeit hochgenaue Karten in großen, zyklichen Umgebungen erstellt.

5.1 Verfahren mit lokaler Aktualisierung

Eine naheliegende Methode, eine Umgebungskarte aufzubauen, besteht darin, inkrementell neue Daten in das bisher erstellte Modell zu integrieren. Wann immer ein neuer Satz von Daten eines Sensors aufgenommen wird, so wird dieser zuerst mit dem vorherigen Sensorbild oder der bisher erstellten Karte überdeckt und anschließend in das Umgebungsmodell integriert, indem zwischen den durch Überdeckung und Odometrie bestimmten Positionen gemittelt wird oder ein Kalman-Filter zum Einsatz kommt. Beispiele für solche Verfahren findet man zahlreich in der Literatur [Ayache und Faugeras, 1989; Leonard *et al.*, 1990; Crowley, 1989; Gonzalez *et al.*, 1994; Borthwick und Durrant-Whyte, 1994; Edlinger und Weiß, 1995; Weiß und Puttkamer, 1995; Kwon und Lee, 1997].

Ein großes Problem dieses Ansatzes ist, daß die entstehende Karte inkonsistent werden kann, da verschiedene Kartenteile immer nur unabhängig voneinander aktualisiert werden. Weiterhin ist es schwierig diese Inkonsistenzen später aufzulösen, vor allem, wenn die aufgenommenen Daten bereits permanent in der Karte integriert sind.

Abbildung 5.1 verdeutlicht dieses Problem an einem Beispiel. Der Roboter startete an Position 0 in der linken unteren Ecke und fuhr in einer Schleife in die Nähe der Ausgangsposition zurück. Für die Erstellung der Karte wurde neue Sensorinformation immer nur mit den bis dahin zuletzt aufgenommenen Sensordaten abgeglichen. Am Ende der Fahrt können die Sensordaten von Position 15 sowohl mit den Daten von Position 14 als auch mit denen der Startposition 0 überdeckt werden. Beide Überdeckungen liefern aber aufgrund des angesammelten Positionsfehlers verschiedene Lösungen für Position 15. Die Unterschiede in den Lösungen können signifikant groß werden, insbesondere wenn die abgefahrene Schleife groß ist. Um Position 15 zu bestimmen, kann ein gewichtetes Mittel der beiden Lösungen (wie es beispielsweise ein Kalman-Filter berechnen würde) verwendet werden. Dann müssen jedoch auch die Schätzungen für Position 14 und weiter zurückliegenden Positionen aktualisiert werden, da sonst Inkonsistenzen entstehen. Im allgemeinen müssen alle Positionen entlang des gefahrenen Pfades aktualisiert werden.

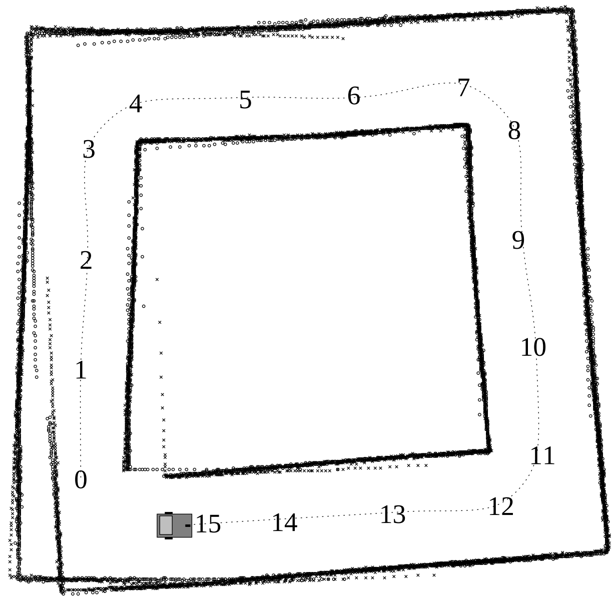


Abbildung 5.1: Problem von Zyklen bei der inkrementellen Kartenerstellung mit lokaler Aktualisierung. Die entstehende Karte wird inkonsistent.

Ein Verfahren zur Erstellung von konsistenten Karten muß daher alle aufgenommenen Sensordaten mit den zugehörigen Positionsschätzungen verwalten und neu eintreffende Sensorinformation so integrieren, daß eine konsistente Repräsentation der Umgebung erreicht wird. Verfahren, die einen solchen Ansatz verfolgen, werden im nächsten Abschnitt besprochen.

5.2 Verfahren für zyklische Umgebungen

Das erste Projekt, das sich systematisch mit konsistenter Kartenerstellung beschäftigt hat, ist das *Hillare*-Projekt [Chatila und Laumond, 1985]. In diesem Projekt werden Entfernungsdaten zu Objekten segmentiert und jedem Objekt wird ein lokales Koordinatensystem zugeordnet. Jedes lokale Koordinatensystem wird zusammen mit der zugehörigen Positionsunsicherheit des Roboters in einem globalen Koordinatensystem referenziert. Wird ein Objekt zu einem späteren Zeitpunkt wiedererkannt, also z.B. nach Abfahren einer Schleife, so wird die Roboter- und Objektposition durch Verwenden eines gewichteten Mittels aktualisiert. Weiterhin wird die Positionsänderung mit einem abnehmenden Effekt rückwärtspropagiert, so daß alle Positionen entlang des Roboterpfades bis zum Anfang der Schleife korrigiert werden. Die potentiellen Probleme dieses Ansatzes liegen darin, daß einerseits Objekte erkannt und wiedererkannt werden müssen, also eine genaue und zuverlässige Objekterkennung benötigt wird, und andererseits nur Positionen von Objekten entlang eines Pfades korrigiert werden, nicht aber die globale Konsistenz aller Objektpositionen erzwungen wird.

In den Arbeiten von Durrant-Whyte [1987; 1988a; 1988b] wird das Problem der Konsistenzerhaltung in Netzwerken mit räumlichen Relationen genauer untersucht. Hier wird das Umgebungsmodell durch räumliche Beziehungen zwischen Objekten repräsentiert und ein probabilistischer Fusionsalgorithmus ähnlich dem Kalman-Filter eingesetzt, um neue Beobachtungen in das Modell zu integrieren. Werden Relationen aufgrund neuer Informationen aktualisiert, so wird die Konsistenz des Gesamtnetzwerkes durch Bedingungen entlang von Schleifen des Netzwerks erzwungen. Die Aktualisierungsprozedur ist als ein Constraint-Optimierungsproblem formuliert und erlaubt es, neue Beobachtungen in das Netzwerk einfließen zu lassen, während zugleich die Konsistenz zwischen bestehenden Bedingungen und neuer Information gewahrt wird. Das Problem relationenbasierter Ansätze ist, daß die Relationen nicht unabhängig voneinander betrachtet werden dürfen und daher weitere Bedingungen hinzugefügt werden müssen. Daher ist dieser Ansatz in der Praxis kompliziert und schwierig anzuwenden.

Smith, Self und Cheesman [1990] beschreiben eine Repräsentation für räumliche Information, welche sie stochastische Karte nennen, und stellen Prozeduren für die Erstellung, Abfrage und inkrementelle Erweiterung dieser Karte vor. Die Karte enthält die Schätzung von räumlichen Beziehungen zwischen Objekten und deren Unsicherheit. Die Prozeduren bieten eine allgemeine Lösung für das Problem der Schätzung unsicherer relativer räumlicher Beziehungen. Die Idee dabei ist, alle räumlichen Beziehungen zusammen mit ihren Unsicherheiten und gegenseitigen Abhängigkeiten in einem Zustand zu repräsentieren. Hierzu wird ein Systemvektor zusammen mit der zugehörigen Kovarianzmatrix verwaltet. Wird neue Information in die Karte eingetragen, so müssen Systemvektor und Kovarianzmatrix um eine Dimension erweitert werden und alle Abhängigkeiten der neuen Beobachtung zu vorhandenen Beobachtungen berechnet werden. Wird eine räumliche Beziehung erneut observiert, so werden Systemvektor und Kovarianzmatrix durch Verwenden eines Kalman-Filters aktualisiert. Vorraussetzungen für die Anwendung dieser Repräsentation sind, daß Winkelfehler wegen Linearisierungsschritten klein sind und die beobachteten Wahrscheinlichkeitsverteilungen sich adäquat durch die ersten beiden Momente (Erwartungswert und Varianz) beschreiben lassen.

Die stochastische Karte von Smith, Self und Cheesman ist eine beliebte Methode für die konsistente Erstellung von Karten. Castellanos *et al.* [1997; 1999] benutzen das Verfahren für die Erstellung einer merkmalsbasierten Umgebungskarte aus Sensorinformationen eines mobilen Roboters und zeigen experimentell, daß die Berücksichtigung von Abhängigkeiten zwischen Merkmalen durch Kovarianzen zu besseren Ergebnissen führt. Hébert *et al.* [1995] zeigen in einer Simulation, daß Abhängigkeiten zwischen Merkmalen untereinander und zwischen Merkmalen und der Roboterposition nicht ignoriert werden dürfen, da sonst bei der Kartenerstellung Positionsunsicherheiten unterschätzt werden. Weiterhin werden Experimente durchgeführt, bei denen die Forderung nach einem kleinen Winkelfehler verletzt ist. Dabei stellte sich heraus, daß schon ein relativ kleiner Winkelfehler von 7° katastrophale Ergebnisse bei der Kartenerstellung erzeugen kann. Ein Grundrahmen für die Untersuchung verschiedener Explorationsstrategien für die stochastische Karte wurde in [Wullschleger *et al.*, 1999] vorgestellt.

Für eine Übersicht weiterer Verfahren, die sich mit dem Problem des Schließens von Zyklen beschäftigen, sei auf [Hébert *et al.*, 1995; Lu und Milios, 1997a] verwiesen. Der nächste Abschnitt beschreibt das Verfahren von Lu und Milios [1997a], welches sehr erfolgreich für die Erstellung von Karten in großen, zyklischen Umgebungen eingesetzt wurde.

5.3 Konsistente Positionsschätzung

Lu und Milios [1995; 1997a] folgen einem anderen Ansatz. Für die Erstellung der Karte werden keine Objekte, die aus Sensordaten gewonnen werden müssen, verwendet, sondern es werden direkt die rohen Daten eines Laserscanners verarbeitet. Die Umgebungskarte selbst wird dabei nicht explizit erstellt, sondern durch die aufgenommenen Scans zusammen mit deren Aufnahmeposition repräsentiert. Der Vorteil hierbei ist, daß das Verfahren in beliebigen Umgebungen eingesetzt werden kann und nicht irgendwelche Merkmale, die in der Umgebung vorhanden sein müssen, erfordert. Der Ansatz benutzt eine Kombination von relationenbasierter und positionsbasierter Repräsentation. Relationen zwischen Positionen werden durch Odometrieinformation und Scan-Matching gewonnen, während die Positionen selbst freie Variablen sind und durch Lösen eines Optimierungsproblems bestimmt werden.

Für die Überdeckung zweier Scans können verschiedene Verfahren eingesetzt werden. Lu und Milios verwenden in ihrer Arbeit den ebenfalls von ihnen entwickelten *IDC*-Algorithmus [Lu und Milios, 1997b]. Aufgrund der Ergebnisse von Gutmann und Schlegel [1996] wird in dieser Arbeit hierfür der kombinierte Überdeckungsalgorithmus aus Abschnitt 3.8 verwendet. Bevor Scans miteinander überdeckt werden, wird wie bei den in Abschnitt 4.4 beschrieben Selbstlokalisierungsverfahren geprüft, ob beide Scans genügend viele gemeinsame Scanpunkte besitzen (Projektionsfilter), und schlechte Scan-Matching-Ergebnisse werden verworfen.

Aus den durch Odometrie und Scan-Matching gewonnenen Beziehungen zwischen Positionen wird ein Netzwerk erstellt, das aus Positionen als Knoten und Beziehungen als Kanten besteht. Im allgemeinen sind die Beziehungen in diesem Netzwerk inkonsistent bzw. widersprüchlich, da Beziehungen keine unabhängigen Variablen sind und mit Fehlern behaftet sind. Die Aufgabe ist nun, alle Positionen so zu bestimmen, daß die Inkonsistenzen weitestgehend aufgelöst werden, d.h. minimal werden. Dies wird dadurch erreicht, daß eine Fehlersumme aufgestellt wird, die alle Positionen als freie Variablen enthält und jede Beziehung in einen Term umgewandelt wird, der wie eine Feder zwischen zwei Positionen angesehen werden kann. Eine Feder besitzt minimale Energie, wenn die beteiligten Positionen genau der Beziehung entsprechen. Die Fehlersumme berechnet dann die Gesamtenergie im Netzwerk und die Positionen werden so bestimmt, daß diese Energie minimal wird.

5.3.1 Definition des Schätzproblems

Lu und Milios betrachten das folgende allgemeine Optimierungsproblem. Gegeben sei ein Netzwerk von unsicheren Messungen über $n+1$ Knoten $X_0, X_1, \ldots, X_n$, wobei jeder Knoten X_i ein d-dimensionaler Positionsvektor ist. Eine Kante D_{ij} zwischen zwei Knoten X_i und X_j repräsentiert eine meßbare Differenz der beiden Positionen. Im allgemeinen ist D_{ij} eine Funktion von X_i und X_j, die möglicherweise nichtlinear ist. Im folgenden wird D_{ij} Meßgleichung genannt und der einfache Fall betrachtet, daß diese linear ist, d.h. $D_{ij} = X_i - X_j$ gilt.

Eine Beobachtung von D_{ij} wird als $\bar{D}_{ij} = D_{ij} + \Delta D_{ij}$ modelliert, wobei ΔD_{ij} ein Gauß-verteilter Fehler mit Mittelwert 0 und bekannter Kovarianzmatrix C_{ij} ist. Das Ziel ist nun, die optimale Schätzung aller Positionen zu bestimmen, gegeben ein Satz von Messungen $\bar{D}_{ij}$ mit Kovarianzmatrizen C_{ij}. Außerdem sollen die Kovarianzmatrizen der geschätzten Positionsvektoren aus den Kovarianzmatrizen der Beobachtungen bestimmt werden.

Das Optimalitätskriterium ist hierbei das der minimalen Varianz (oder maximum likelihood), d.h. die Knoten X_i (und damit auch die Meßgleichungen D_{ij}) werden so bestimmt, daß die bedingte Wahrscheinlichkeit der abgeleiteten D_{ij}, gegeben die Beobachtung $\bar{D}_{ij}$, maximiert wird. Nimmt man an, daß alle Beobachtungsfehler Gauß-verteilt und unabhängig voneinander sind, so ist das Optimalitätskriterium äquivalent zur Minimierung der folgenden Mahalanobis-Distanz:

$$W = \sum_{(i,j)} (D_{ij} - \bar{D}_{ij})^T C_{ij}^{-1} (D_{ij} - \bar{D}_{ij}) \tag{5.1}$$

wobei die Summe über alle Meßgleichung läuft. Im Falle, daß zu einem Knotenpaar keine Beobachtung $\bar{D}_{ij}$ vorliegt, wird die Inverse der zugehörigen Kovarianzmatrix auf Null gesetzt, d.h. $C_{ij}^{-1} = 0$.

Durch die Linaritätsannahme der Meßgleichung läßt sich die letzte Formel umschreiben zu:

$$W = \sum_{(i,j)} (X_i - X_j - \bar{D}_{ij})^T C_{ij}^{-1} (X_i - X_j - \bar{D}_{ij}) \tag{5.2}$$

Da nur relative Information in der Funktion W benutzt wird, kann eine Position frei gewählt werden. Ohne Beschränkung der Allgemeinheit sei $X_0 = 0$ und $X_1, \ldots X_n$ seien Positionen relativ zu X_0.

Notiert man die Meßgleichungen D_{ij} in Matrixform, d.h.

$$\mathbf{D} = \mathbf{HX} \tag{5.3}$$

wobei $\mathbf{X}$ ein nd-dimensionaler Vektor bestehenden aus der Konkatenation von $X_1, \ldots, X_n$, $\mathbf{D}$ die Konkatenation aller Positionsdifferenzen der Form $D_{ij} = X_i - X_j$ und $\mathbf{H}$ eine Inzidenzmatrix ist, deren Elemente 1, -1 oder 0 sind, so kann die Funktion W wie folgt in Matrixform formuliert werden:

$$W = (\bar{\mathbf{D}} - \mathbf{HX})^T \mathbf{C}^{-1} (\bar{\mathbf{D}} - \mathbf{HX}) \tag{5.4}$$

Hierbei ist $\bar{\mathbf{D}}$ die Konkatenation aller Beobachtungen D_{ij} und $\mathbf{C}$ die Kovarianzmatrix von $\bar{\mathbf{D}}$, welche aus den einzelnen C_{ij}'s als Untermatrizen besteht.

Die Lösung $\mathbf{X}$, welche W minimiert, ergibt sich dann als

$$\mathbf{X} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{C}^{-1} \bar{\mathbf{D}} \tag{5.5}$$

und die Kovarianz von $\mathbf{X}$ als

$$\mathbf{C_X} = (\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H})^{-1} \tag{5.6}$$

Sind die Meßfehler unabhängig voneinander, so ist $\mathbf{C}$ blockdiagonal und die Lösung vereinfacht sich. Sei $\mathbf{G}$ die $nd \times nd$ Matrix $\mathbf{H}^T \mathbf{C}^{-1} \mathbf{H}$. Die $d \times d$ Untermatrizen von $\mathbf{G}$ können bestimmt werden durch

$$G_{ii} = \sum_{j=0}^{n} C_{ij}^{-1} \tag{5.7}$$

$$G_{ij} = -C_{ij}^{-1} \tag{5.8}$$

Sei weiterhin $\mathbf{B}$ der nd-dimensionale Vektor $\mathbf{H}^T \mathbf{C}^{-1} \bar{\mathbf{D}}$. Die d-dimensionalen Untervektoren von $\mathbf{B}$ lassen sich bestimmen durch

$$B_i = \sum_{j=0; j \neq i}^{n} C_{ij}^{-1} \bar{D}_{ij} \tag{5.9}$$

Dann können die Positionsschätzung $\mathbf{X}$ und Kovarianzmatrix $\mathbf{C_X}$ bestimmt werden durch:

$$\mathbf{X} = \mathbf{G}^{-1}\mathbf{B}; \qquad \mathbf{C_X} = \mathbf{G}^{-1} \tag{5.10}$$

Obige Gleichung erfordert, daß die Matrix $\mathbf{G} = \mathbf{H}^T \mathbf{C}^{-1} \mathbf{H}$ invertierbar ist. Lu und Milios vermuten, daß es möglich ist die Invertierbarkeit von $\mathbf{G}$ zu beweisen, sofern das Netzwerk vollständig verbunden ist und sich die individuellen Fehlerkovarianzen „normal“ verhalten.

5.3.2 Anwendung für die Kartenerstellung

Eine typische Anwendung dieses Optimierungsverfahrens besteht in der Navigation eines mobilen Roboters zur Schätzung der Roboterpositionen $(x_i, y_i, \theta_i)^T$ mit Positionsunsicherheiten Σ_i zu verschiedenen Zeitpunkten i. Als Beobachtungen dienen relative Positionsinformationen, welche aus Odometriedaten und durch Scan-Matching bestimmt werden. Mit dem Verfahren können dann die Aufnahmepositionen von während der Fahrt aufgenommen Entfernungsdaten bestimmt werden und durch Eintragen der Daten in ein globales Koordinatensystem kann eine konsistente Umgebungskarte erstellt werden.

In dieser Anwendung ist die Meßgleichung D_{ij} wegen der Winkelkomponente θ nicht-linear. Lu und Milios [1997a] zeigen jedoch, wie die Meßgleichung in diesem Fall linearisiert und das Optimierungsverfahren angewendet werden kann. Da durch die Linearisierung Fehler entstehen, die proportional zur initialen Positionsschätzung sind, wird das Verfahren iterativ angewendet. In der Praxis reichen vier bis fünf Iterationen aus bis das Ergebnis auf Maschinengenauigkeit konvergiert ist.

Die Zeitkomplexität der konsistenten Positionsschätzung bei einem Datensatz von n Scans setzt sich aus der Zeit für den Aufbau des Netzwerkes und der Bestimmung der optimalen Positionen zusammen. Für den Aufbau muß für jedes Paar von Scans eine Überprüfung, ob genügend Überlappung vorliegt, und ggf. eine Scanüberdeckung durchgeführt werden. Nimmt man an, daß die Anzahl von Scanpunkten pro Scan konstant ist, d.h. Scan-Matching $O(1)$ Zeit benötigt, dann beträgt die Zeit für den Aufbau des Netzwerkes $O(n^2)$ im schlechtesten Fall. Für die Bestimmung der optimalen Positionen muß die Matrix $\mathbf{G}$ invertiert werden, was $O(n^3)$ viel Zeit benötigt. Zusammen ergibt sich für die konsistente Positionsbestimmung also eine Zeitkomplexität von $O(n^3)$.

In vielen Fällen ist die Matrix $\mathbf{G}$ wegen der Struktur der Umgebung dünn besetzt, z.B. in einem engen Gang gibt es meist nur Überlappungen von Scans, die dicht beieinander liegen. Dies dünne Besetztheit der Beobachtungsmatrix kann ausgenutzt werden, um die Positionsschätzung $\mathbf{X}$ nach Gleichung 5.10 schneller als $O(n^3)$ zu bestimmen, z.B. durch Einsatz des *LASPack*-Systems [Skalicky, 1996]. Auf diese Weise kann jedoch nicht die Kovarianzmatrix von $\mathbf{X}$ bestimmt werden, da hierfür in jedem Fall $\mathbf{G}$ invertiert werden muß.

5.3.3 Resultate

Im folgenden wird ein Beispiel für die Kartenerstellung mittels konsistenter Positionsschätzung präsentiert. Das Verfahren wird auf einen Datensatz von 74 Scans angewendet, der in einer Explorationsfahrt des Roboters *AMOS* des Forschungsinstituts für anwendungsorientierte Wissensverarbeitung (FAW) aufgezeichnet wurde. *AMOS* verfügt neben zahlreicher anderer Sensorik über einen 360° *IBEO*-3d-Scanner, der für diese Aufnahmen in einem 2d Modus betrieben wurde.

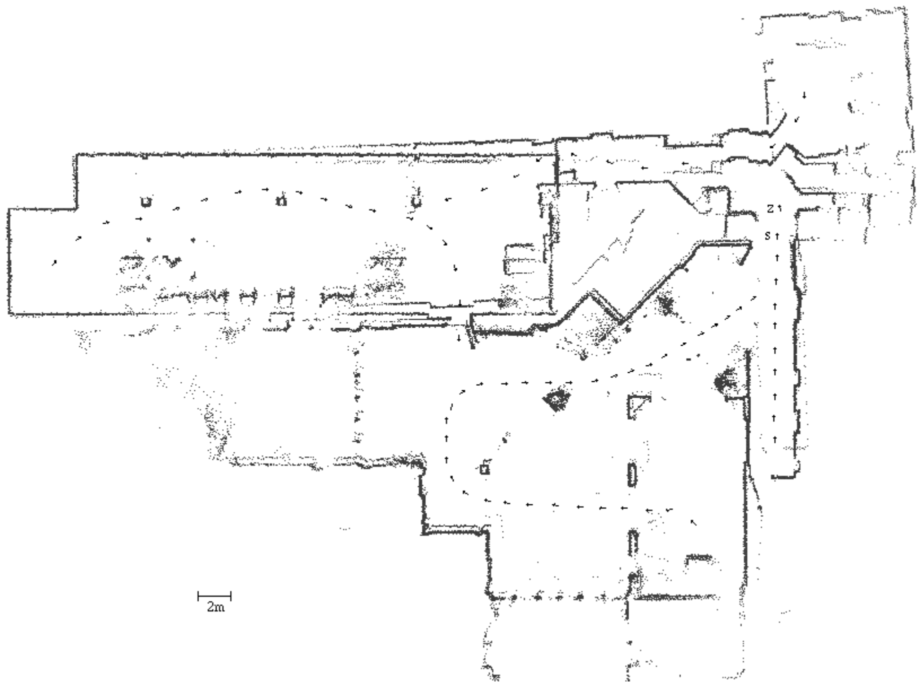


Abbildung 5.2: Satz von 74 Scans, welche in einer Explorationsfahrt im 39 auf 25 Meter großen Untergeschoß des FAW Ulm aufgezeichnet wurden. Pfeile geben die durch Odometrie bestimmten Aufnahmepositionen an.

In Abbildung 5.2 sind die rohen Scandaten zu sehen, die lediglich aufgrund von Odometriedaten in ein Koordinatensystem eingetragen wurden. Das Fahrzeug startete an der mit **S** markierten Stelle und fuhr den rechten Gang entlang. Anschließend wurde in einem Bogen der größte Bereich des Gebäudes besucht und hinterher der obere linke Raum befahren. Zuletzt wurde das Fahrzeug durch den oberen Gang zurück zur Ausgangsposition **S** gefahren. Durch Odometriefehler hat sich hier ein Positionsfehler von ca. $1400mm$ und $2.5°$ angesammelt und die durch Koppelnavigation bestimmte Position **Z** weicht deutlich von der tatsächlichen Position **S** ab.

Abbildung 5.3 zeigt die Umgebungskarte, wie sie ensteht, wenn man das Verfahren von Lu und Milios zusammen mit dem kombinierten Scanüberdeckungsalgorithmus auf diesen Datensatz anwendet und die Scans aufgrund der verbesserten Positionsschätzung in das Koordinatensystem einträgt. Wie zu sehen ist, fallen die Positionen **S** und **Z** zusammen und eine konsistente Umgebungskarte entsteht.

Ein weiteres Beispiel für die Anwendung der konsistenten Positionsschätzung ist in Abbildung 5.4 zu sehen. Diese Daten wurden in der Universität Toronto mit einem *B21*-Roboter mit *SICK*-Laserscanner aufgenommen [Hähnel *et al.*, 1998].

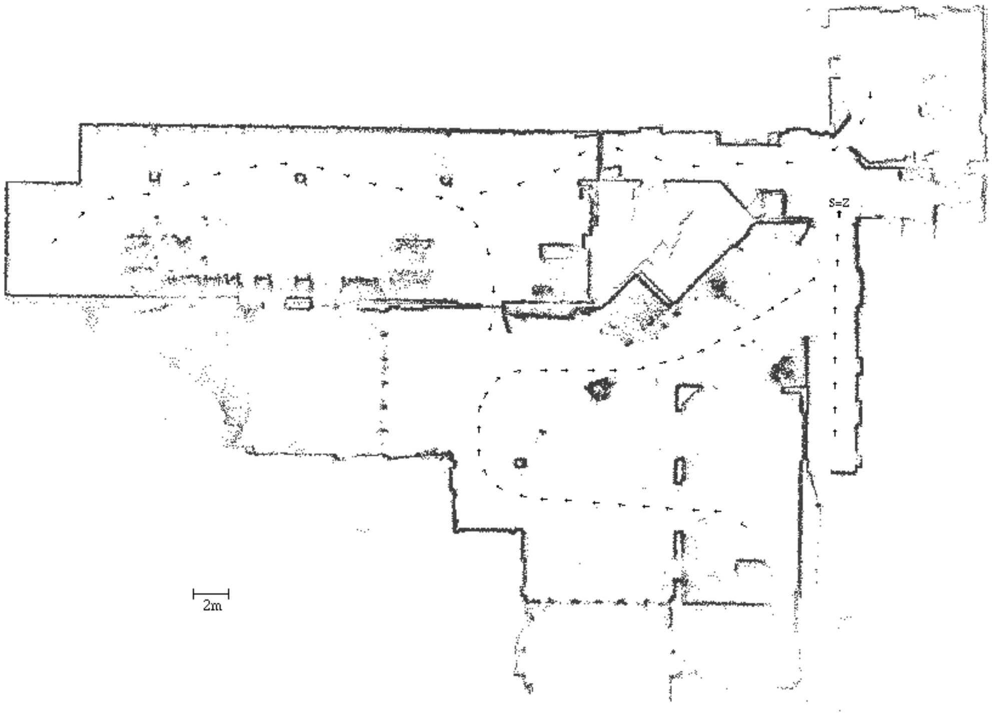


Abbildung 5.3: Scans nach Korrektur der Aufnahmepositionen durch Anwendung der konsistenten Positionsschätzung mit kombiniertem Scan-Matching-Verfahren.

5.3.4 Bewertung

Die konsistente Positionsschätzung ist ein Verfahren zur Erstellung von hochgenauen Umgebungskarten aus Odometrie- und Laserscannerdaten. Das Verfahren verlangt, daß zunächst alle Daten aufgezeichnet und anschließend verarbeitet werden. Wünschenswert wäre jedoch ein inkrementeller Ansatz, der Scandaten nach und nach in die Umgebungskarte integriert. Obwohl Lu und Milios hierfür auch eine inkrementelle Variante beschreiben, ist diese nicht praktikabel, da jedesmal der gesamte Datensatz verarbeitet wird.

Weiterhin müssen die initialen Aufnahmepositionen der Scans bereits nahe ihrer tatsächlichen Position liegen, da die Scanüberdeckung eine ungefähre Position der beiden Scans zueinander benötigt. Die Daten müssen also bereits topologisch korrekt sein. Dies ist jedoch nicht immer gegeben, insbesondere wenn die Umgebung groß und Zyklen lang sind, oder die Odometrie schlecht ist. Wendet man das Verfahren in solchen Fällen an, so werden Zyklen entweder nicht geschlossen, da keine Überlappung der entsprechenden Scans vorhanden ist, oder aber es werden Scans miteinander überdeckt, die topologisch nicht zusammengehören. In diesem

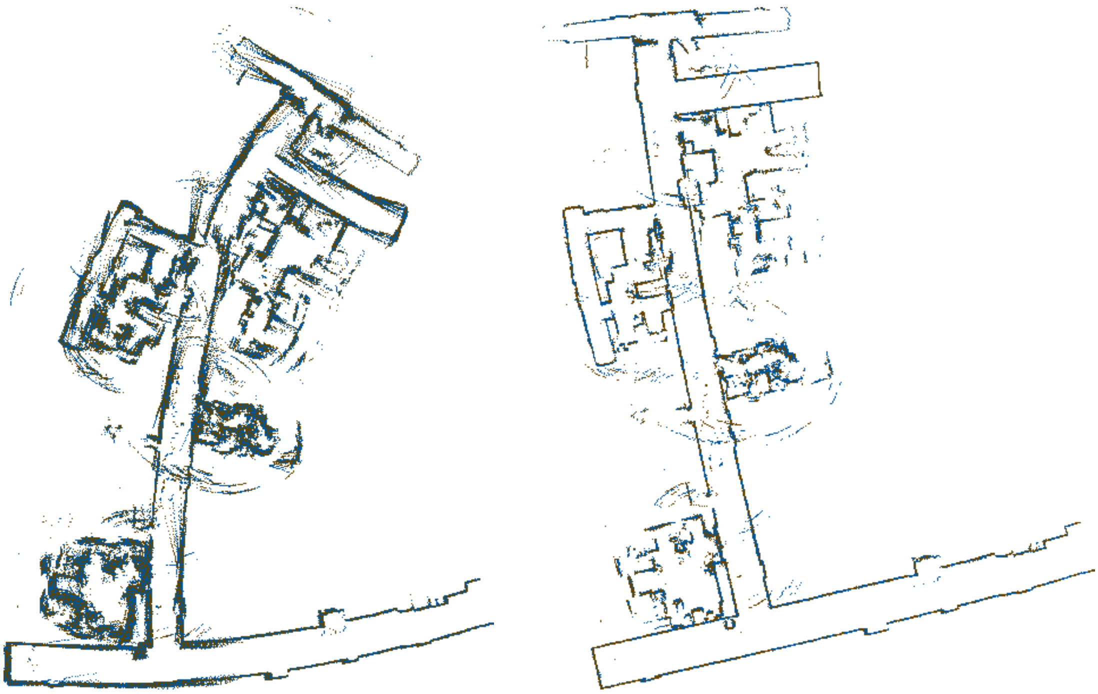


Abbildung 5.4: Weiteres Beispiel für die Anwendung der konsistenten Positionsschätzung auf Laserdaten, welche in einer 29 auf 39 Meter großen Umgebung der Universität Toronto aufgenommen wurden. Links die rohen Daten, rechts das Ergebnis nach konsistenter Positionsschätzung.

Fall können katastrophale Resultate die Folge sein, da die Scans möglicherweise zufällig zusammenpassen.

Die Daten der Explorationsfahrt sollten also vor Anwendung der konsistenten Positionsschätzung topologisch korrekt sein. Ein Verfahren, das dies ermöglicht wird im nächsten Abschnitt beschrieben.

5.4 Erwartung und Maximierung

Ein auf Landmarken basiertes, statistisches Verfahren zur Erstellung von topologisch korrekten Karten in großen Umgebungen wurde von Thrun *at al.* [1998c; 1998b] entwickelt. Diese Verfahren versucht aus einer Menge von observierten, nicht unterscheidbaren Landmarken und dazwischenliegenden Odometriemessungen herauszufinden, welche Landmarken zueinander gehören. Hierzu wird der Roboter per Joystick durch die Umgebung bewegt. An verschiedenen Stellen wird dem Roboter per Tastendruck mitgeteilt, daß eine Landmarke an der augenblicklichen Position vorliegt. Die Aufgabe des Systems ist es nun herauszufinden, welche Landmarken zueinandergehören. Hieraus können durch Interpolation zwischen den Landmarkenpositionen alle Positionen des Roboters während der Explorationsfahrt bestimmt werden und somit eine topologisch korrekte Karte erstellt werden.

5.4.1 Stochastische Formulierung

In statistischen Termen läßt sich das Problem der Kartenerstellung als das Problem der Suche nach der wahrscheinlichsten Karte m^*, gegeben Odometrie- und Sensordaten, formulieren.

$$m^* = \operatorname*{argmax}_m P(m \mid d) \tag{5.11}$$

wobei $d = (A^n, O^n)$ die aufgezeichneten Odometrie- und Sensordaten sind und m über alle möglichen Karten läuft.

In [Thrun *et al.*, 1998b] wird gezeigt wie sich die Bestimmung der wahrscheinlichsten Karte durch Anwenden der Bayes'schen Regel und Ausnutzung von Unabhängigkeitsannamhen auf Bewegungs- und Sensormodell reduzieren läßt. Die gesuchte Karte läßt sich hierdurch berechnen als

$$m^* = \operatorname*{argmax}_m \int \dots \int \prod_{t=1}^{n} P(o_t \mid m, \xi_t) \prod_{t=1}^{n-1} P(\xi_{t+1} \mid a_t, \xi_t) d\xi_1, \dots d\xi_n \tag{5.12}$$

Hierbei ist $P(o_t \mid m, \xi_t)$ das Sensormodell, welches die Wahrscheinlichkeit angibt, daß o_t beobachtet wird, gegeben der Roboter befindet sich an Position ξ_t in der Karte m, und $P(\xi_{t+1} \mid a_t, \xi_t)$ das Bewegungsmodell, welches die Wahrscheinlichkeit angibt, daß der Roboter sich an Position ξ_{t+1} befindet gegeben er stand zum Zeitpunkt t an Position ξ_t und führte Aktion a_t aus.

Leider ist die Bestimmung der wahrscheinlichsten Karte nach Formel 5.12 äußerst rechenzeitaufwendig. Zum einen müssen im Suchraum alle möglichen Karten ausprobiert werden, was für große Umgebungen 10^6 oder mehr Möglichkeiten bedeutet, selbst wenn nur grobe Näherungen benutzt werden. Zum anderen muß für die Berechnung der Wahrscheinlichkeit einer Karte über alle Positionen zu allen betrachteten Zeiten integriert werden, was bei üblichen Datensätzen bedeutet, daß mehr als 10^5 unabhängige Positionsvariablen vorliegen, die jeweils ca. 10^8 verschiedene Werte annehmen können.

Glücklicherweise gibt es eine effiziente Technik, um Formel 5.12 durch eine *hill-climbing*-Methode auszuwerten: der *EM*-Algorithmus [Dempster *et al.*, 1977]. Der *EM*-Algorithmus besteht aus zwei Schritten, einem Erwartungsschritt (E-Schritt) und einem Maximierungsschritt (M-Schritt).

1. Im E-Schritt werden Wahrscheinlichkeiten $P(\xi \mid m, d)$ für die Roboterposition ξ zu allen Zeitpunkten aufgrund der als bisher besten bestimmten Karte m berechnet. Im ersten Schritt gibt es noch keine Karte.

2. Im M-Schritt wird die wahrscheinlichste Karte durch Maximierung von $\text{argmax}_m P(m \mid \xi, d)$ und Verwenden der im E-Schritt bestimmten Positionen berechnet.

Der E-Schritt entspricht einem Lokalisierungsschritt mit fester Karte, während der M-Schritt einer Kartenerstellung mit festen Roboterpositionen entspricht. Durch Iteration der beiden Schritte können so die Karte und die abgefahrenen Positionen verbessert werden. Der *EM*-Algorithmus ist ein *hill-climbing*-Verfahren, was bedeutet, daß nicht garantiert ist, daß ein globales Maximum gefunden wird.

5.4.2 Kartenerstellung

Das Verfahren von Thrun *et al.* benutzt für das Sensormodell signifikante Stellen (Landmarken) in der Umgebung des Roboters. Es wird angenommen, daß der Roboter über einen Sensor verfügt, der ihm mitteilt, ob er an einer signifikanten Stelle steht oder nicht. In den nachfolgenden Experimenten wurde dies so realisiert, daß dem System per Tastendruck signifikante Stellen mitgeteilt wurden. Das System weiß weder wieviele signifikante Stellen es in der Umgebung gibt, noch wie oft eine Landmarke besucht wurde. Die Aufgabe ist es herauszufinden, welche Landmarken zueinander gehören.

In der Implementierung von Thrun *et al.* werden für die Repräsentierung aller Wahrscheinlichkeiten (Positionen, Karten, ...) Gitter mit grober Auflösung benutzt (ca. $1m$ und $5°$). Wie gut das Verfahren ist und wie genau die berechneten Karten sind wird im nächsten Abschnitt gezeigt.

Da das Verfahren von Thrun *et al.* topologisch korrekte Karten und die konsistente Positionsschätzung von Lu und Milios aus topologisch korrekten Karten metrisch hochgenaue Karten erzeugt, ist es naheliegend, die beiden Verfahren zu kombinieren. Thrun und Gutmann *et al.* [1998d] benutzen diese Vorgehensweise zur Erstellung von hochgenauen Navigationskarten in großen Umgebungen.

5.4.3 Ergebnisse

Im folgenden werden mehrere Beispiele für die Anwendung des eben beschriebenen Verfahrens gezeigt. Abbildung 5.5a zeigt Daten, die in einer Explorationsfahrt im Carnegie Museum for Natural Science mit einem Roboter mit *SICK*-Laserscanner aufgenommen wurden. Wie zu sehen ist, hat sich hierbei ein deutlicher Odometriefehler angesammelt. Abbildung 5.5b zeigt die Karte nach Anwendung des *EM*-Verfahrens. Es entsteht eine topologisch korrekte aber relativ ungenaue Karte. Schließlich zeigt Abbildung 5.5 das Resultat nach Anwendung der konsistenten Positionsschätzung. Es entsteht eine sehr genaue metrische Karte, die direkt für Navigationszwecke eingesetzt werden kann.

Abbildungen 5.6 und 5.7 zeigen weitere Beispiele für die Anwendung des kombinierten Verfahrens zur Erstellung von Umgebungskarten in der Wean Hall der

Carnegie Mellon University, Pittsburgh, USA. Diese Umgebung hat eine Größe von ungefähr 80 auf 25 Meter und wurde mit einem *B21*-Roboter mit *SICK*-Laserscanner exploriert.

5.4.4 Bewertung

Das *EM*-Verfahren zur Erstellung von Umgebungskarten ist in der Lage, topologisch korrekte Karten aus Odometriedaten und Landmarkeninformationen zu generieren. Zusammen mit der konsistenten Positionsschätzung können dadurch hochgenaue Karten erstellt werden.

Ein Nachteil des *EM*-Verfahrens ist, daß in der gegenwärtigen Version zusätzliche Informationen von außen gegeben werden müssen, nämlich die Nachricht, daß eine signifikante Stelle (Landmarke) erreicht wurde. Eine erst kürzlich veröffentlichte Variante des *EM*-Verfahrens benutzt Sonardaten anstatt Landmarken [Burgard *et al.*, 1999]. Das Verfahren wurde aber bisher nur in relativ kleinen Umgebungen eingesetzt und besitzt nur eine geringe Genauigkeit.

Ein weiterer Nachteil ist, daß das *EM*-Verfahren sehr rechenintensiv ist, was eine Echtzeitanwendung ausschließt. Weiterhin müssen zunächst alle Daten gesammelt werden, bevor mit der Kartenerstellung begonnen werden kann.

Wünschenswert wäre ein Verfahren, das Odometrie- und Sensordaten während der Explorationsfahrt in Echtzeit in eine Karte integriert, ohne dabei weitere Informationen von außen zu benötigen. Ein solches Verfahren wird im nächsten Abschnitt beschrieben.

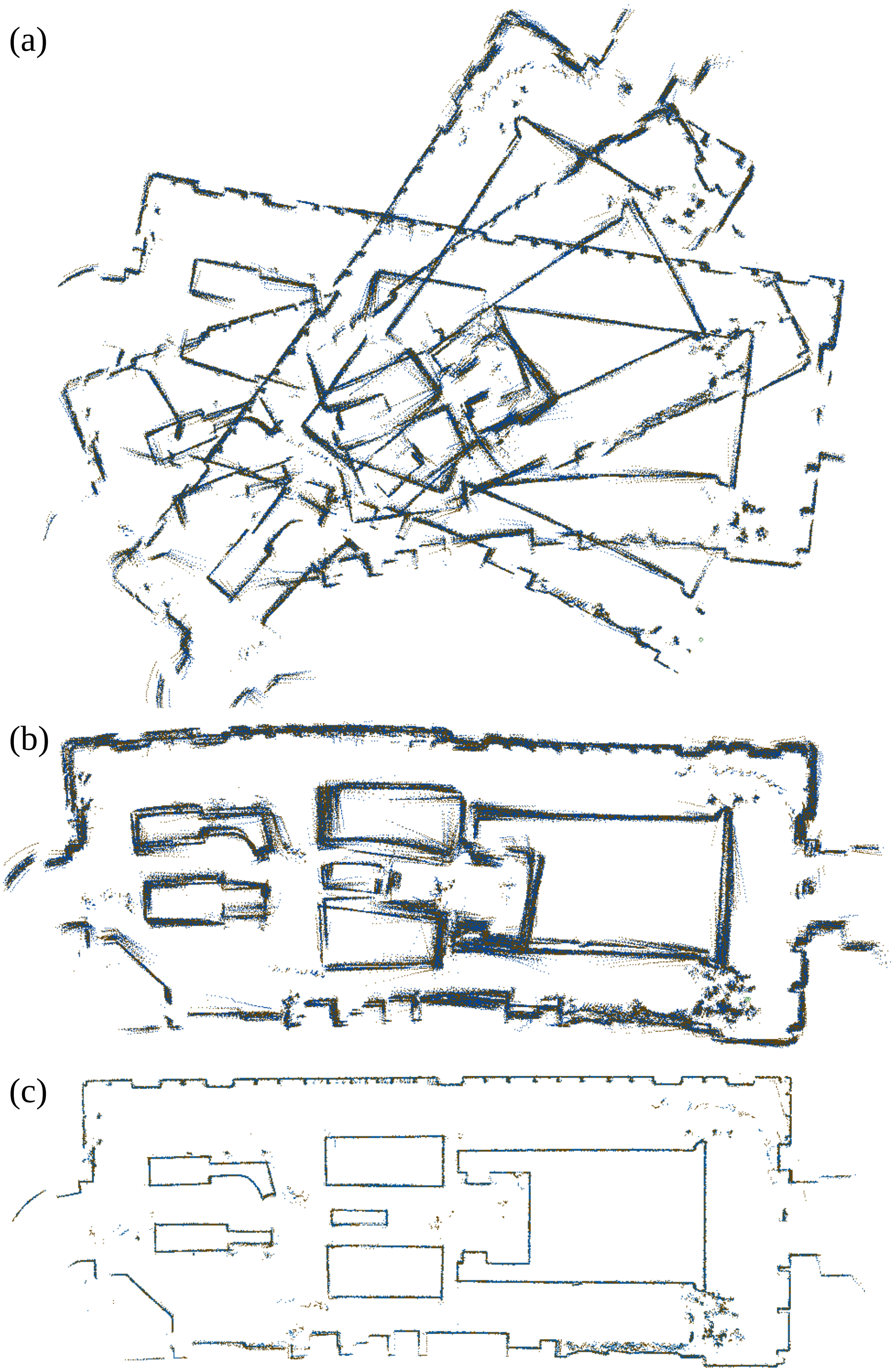


Abbildung 5.5: Kartenerstellung im Carnegie Museum of Natural Science (45×15 Meter). (a) Karte, die durch rohe Odometriedaten entsteht. (b) Karte nach Korrektur durch *EM*-Verfahren. (c) Ergebnis nach konsistenter Positionsschätzung.

(a)

(b)

(c)

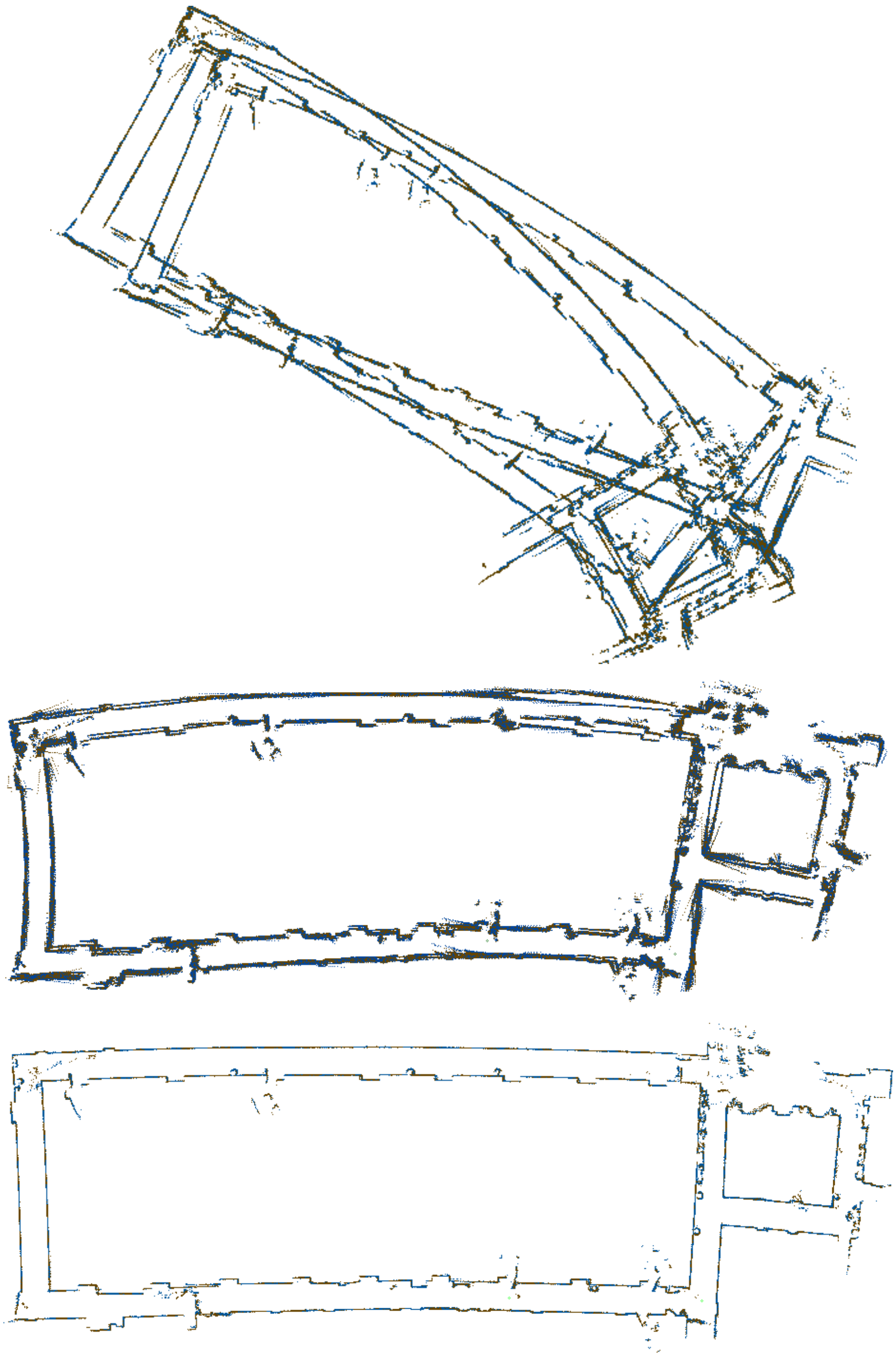


Abbildung 5.6: Kartenerstellung in der Wean Hall der Carnegie Mellon University (80×25 Meter). (a) Karte, die durch rohe Odometriedaten entsteht. (b) Karte nach Korrektur durch *EM*-Verfahren. (c) Ergebnis nach konsistenter Positionsschätzung.

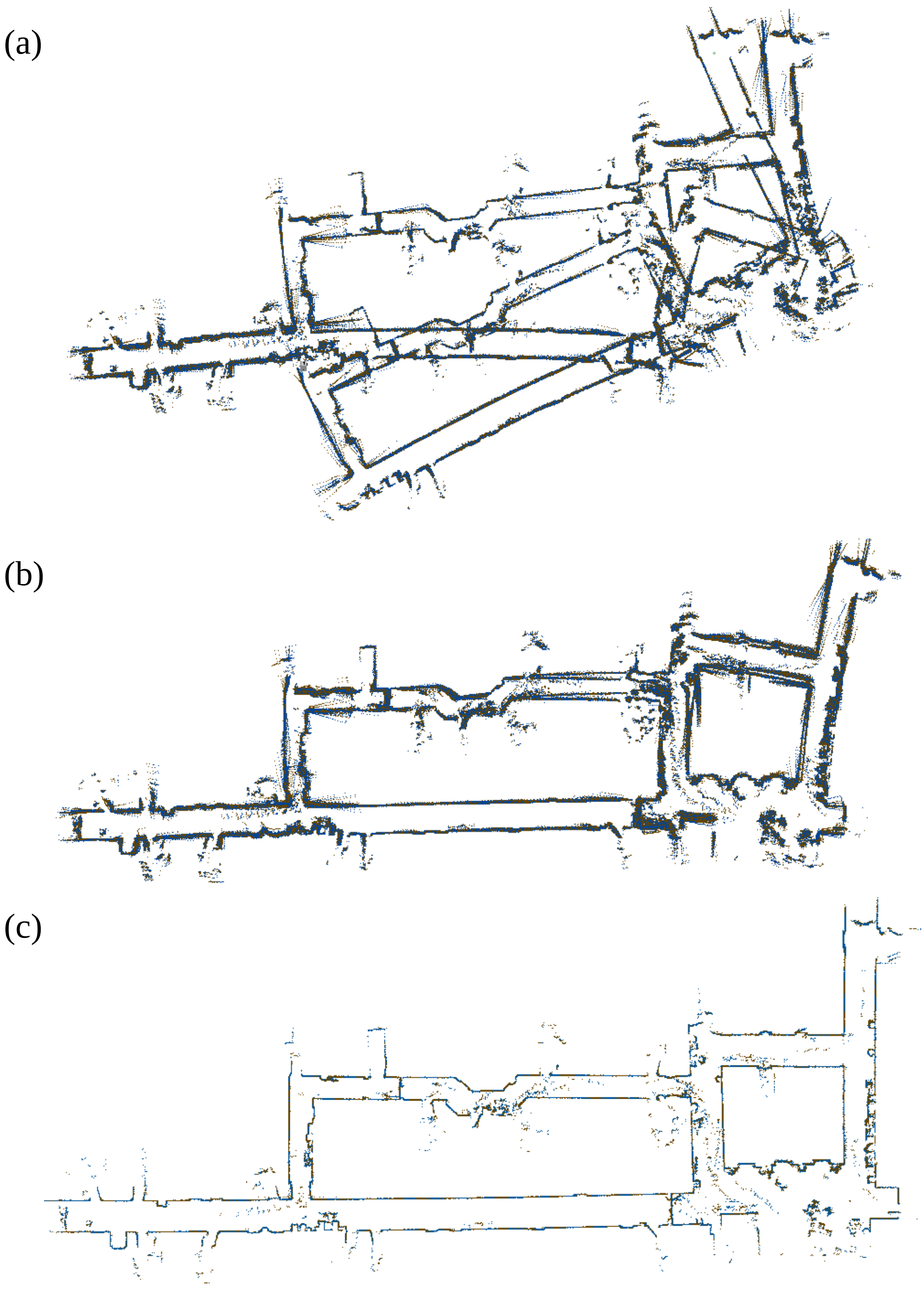


Abbildung 5.7: Weiteres Beispiel für die Kartenerstellung in der Wean Hall. (a) rohe Daten, (b) nach *EM*-Korrektur, (c) Ergebnis nach konsistenter Positionsschätzung.

5.5 LRGC-Verfahren

In diesem Abschnitt wird ein neues Verfahren zur Bestimmung einer konsistenten globalen Positionsschätzung vorgestellt, das *LRGC*-Verfahren (*Local Registration and Global Correlation*) [Gutmann und Konolige, 1999]. Das Verfahren baut direkt auf der Methode der konsistenten Positionsschätzung von Lu und Milios [1997a] (siehe Abschnitt 5.3) auf und benutzt zwei weitere Techniken, um effizient neue Daten in eine bestehende Karte zu integrieren, und um topologische Zusammenhänge zu entdecken.

Sei zunächst der Fall gegeben, daß eine konsistente Karte bereits erstellt wurde und eine neue Position l_n hinzugefügt werden soll (siehe Abbildung 5.8). Die neue Position besitzt eine Verbindung zur vorherigen Position aufgrund von Odometriemessungen und mehrere Verbindungen zu vorhergehenden Positionen aufgrund ausreichender Überlappung von Scans und zugehörigen Scan-Matching-Ergebnissen. Diese Beziehungen sind in Abbildung 5.8 durch dicke Bögen markiert. Solange der Roboter neue Bereiche exploriert, sind immer nur zeitlich zusammenliegende Positionen miteinander verbunden und es gibt keine Beziehungen über längere Distanzen.

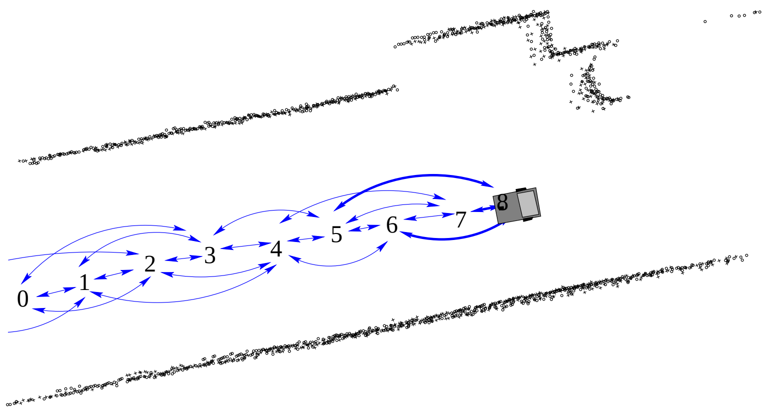


Abbildung 5.8: Hinzufügen einer neuen Position zu einer bestehenden Karte in einer noch nicht explorierten Umgebung.

Solange der Roboter nur neue Bereiche exploriert, reicht es aus, immer nur eine lokale Aktualisierung der Roboterposition vorzunehmen. Wenn der Roboter jedoch einen großen Zyklus schließt, so tritt die topologische Korrektheit in den Vordergrund, da neue Positionen plötzlich zu früheren in Bezug gesetzt werden müssen. Hier ist es wichtig, topologische Zusammenhänge zuverlässig zu bestimmen, da eine falsche Entscheidung zu katastrophalen Ergebnissen führen kann. Einzelne Scans alleine enthalten für gewöhnlich nicht genug Information, um topologische Zusammenhänge zu entscheiden, insbesondere dann nicht, wenn die Umgebung relativ uniform in einer Richtung ist wie z.B. in einem langen Korridor. Daher werden in dieser Arbeit mehrere Scans zu einem Kartenstück zusammengefaßt und als Muster benutzt, um Überdeckungen in der bisher erzeugten Karte zu finden.

Diese Technik ist offensichtlich zuverlässiger als einzelne Scans miteinander zu vergleichen, aber es stellt sich die Frage, wie die Suche nach der passenden Überdeckung effizient realisiert werden kann. Glücklicherweise wurde dieses Problem kürzlich von Konolige und Chou [1999] im Kontext der Selbstlokalisierung eines mobilen Roboters genauer untersucht und eine Methode zur Korrelation eines Kartenstücks mit einer apriori Karte entwickelt. Ergebnisse zeigen, daß diese Methode effizient und zuverlässig ist und aus diesem Grund wird sie hier für die Bestimmung der topologischen Korrektheit zur Kartenerstellung in zyklischen Umgebungen benutzt. Die Korrelation arbeitet dabei im Hintergrund, prüft von Zeit zu Zeit, ob ein topologischer Zusammenhang vorliegt, und fügt ggf. neue Beziehungen zwischen alten und neuen Kartenteilen ein.

Der *Local Registration and Global Correlation* Algorithmus beruht auf drei unterschiedlichen Techniken: Scan-Matching, konsistente Positionsschätzung und Kartenkorrelation. Im folgenden werden diese Techniken nochmals genauer vorgestellt und erklärt, welche Modifikationen für das *LRGC*-Verfahren vorgenommen wurden. Danach wird der Algorithmus selbst beschrieben und gezeigt wie die einzelnen Komponenten zu einem praktikablen Echtzeitsystem für die Kartenerstellung aus dichter Entfernungsinformation zusammengesetzt werden. Zuletzt werden Ergebnisse mit realen Daten präsentiert und Erweiterungen vorgestellt.

5.5.1 Scan-Matching

Scan-Matching ist der Prozeß, einen Scan so zu verschieben und zu verdrehen, daß eine maximale Überdeckung mit einem apriori Modell, z.B. einem anderen Scan entsteht. In Kapitel 3 wurden verschiedene Verfahren vorgestellt, um Scans miteinander zu überdecken. Dabei stellte sich heraus, daß das kombinierte Scan-Matching-Verfahren wegen seiner Universalität und Ausnutzung evt. polygonaler Eigenschaften der Umgebung die besten Resultate liefert. Aus diesem Grund wird für das *LRGC*-Verfahren diese Scan-Matching-Methode ausgewählt.

Die kritischen Seiten von Scan-Matching liegen in der Gefahr, daß Ungenauigkeiten von Positionen unterschätzt werden, was Schwierigkeiten in der konsistenten Interpretation von Positionen sich überlappender Scans hervorrufen kann (siehe [Hébert *et al.*, 1995] über dieses spezielle Problem). Scan-Matching sollte außerdem quantitativ hochwertige Resultate liefern, z.B. sollten gerade Linien in einem Korridor auch zu entsprechend ausgerichteten Scans führen.

5.5.2 Konsistente Positionsschätzung

Das Schlüsselkonzept um Scan-Matching-Ergebnisse zu fusionieren, ist die konsistente Positionsbestimmung von Lu und Milios [1997a] wie sie in Abschnitt 5.3

beschrieben wurde. Wie dort bereits ausgeführt wurde, benötigt diese Methode eine gute Anfangsschätzung der Scanpositionen, um brauchbare Resultate zu erzeugen. Daher wird diese Methode für zwei verschieden Zwecke benutzt:

1. Um lokale Kartenstücke aus den letzten paar vom Roboter aufgenommenen Scans zu erstellen. In diesem Fall liegen immer topologisch korrekte Daten vor, da nur sehr wenig Odometriefehler akkumuliert wurde. Selbst wenn größere Fehler in den Odometriedaten vorhanden sind, z.B. bei einem rutschenden Roboter oder bei Synchronisationsproblemen zwischen Scanner und Odometrie, kann die Anwendung von Scan-Matching und konsistenter Positionsschätzung oftmals wieder die richtige Geometrie zurückgewinnen.

2. Um einen Zyklus zu schließen, nachdem topologische Beziehungen durch Kartenkorrelation bestimmt wurden. In diesem Fall werden zunächst die neuen Beziehungen zur Karte hinzugefügt und die konsistente Positionsschätzung darauf angewandt. Danach, wenn der Zyklus geschlossen wurde und eine topologisch korrekte Karte entstanden ist, wird die konsistente Positionsschätzung nochmals benutzt, wobei zuvor die neuen Beziehungen durch Scan-Matching-Ergebnisse ersetzt werden. Dies führt zu einer weiteren quantitativen Verbesserung der berechneten Karte.

Eine typische Netzwerktopologie ist in Abbildung 5.8 dargestellt. Wenn eine neue Position l_n der Kette hinzugefügt wird, so werden immer nur die letzten K Scanpositionen für die Aktualisierung der Karte benutzt. Um herauszufinden, ob die Aktualisierung dieser K-Nachbarschaft der globalen Aktualisierung aller Positionen gleichwertig ist, wurden eine Reihe von Experimenten durchgeführt, welche beide Methoden miteinander vergleichen. Abbildung 5.9 zeigt den Effekt der Variierung von K für eine Karte von 150 Positionen, wobei der Abstand zwischen aufeinanderfolgenden Positionen ungefähr 0.3 Meter beträgt. Für $K \geq 7$ ist der durchschnittliche Positionsfehler kleiner einem Millimeter und bleibt unverändert für größere K.[1]

Fehler in der lokalen Aktualisierung können enstehen, wenn nicht genug lokaler Kontext vorhanden ist, um die Scans richtig auszurichten. Abbildung 5.10 zeigt die Differenzfehler für $K = 5$ und $K = 10$ während eine Karte erstellt wird. Ungefähr bei Position 30 erzeugt ein schwieriger Scan einen großen Fehler. Während die größere Nachbarschaft ($K = 10$) durch Hinzunahme von weiterem Kontext sich schnell wieder erholt, kann die kleinere ($K = 5$) den Fehler nicht mehr korrigieren.

Ein paar Eigenschaften dieser inkrementellen Aktualisierung sollten festgehalten werden. Zum einen ist der Rechenaufwand in jedem Schritt konstant, da die

[1] Positionsfehler werden aufgrund der Differenz zweier aufeinanderfolgender Positionen berechnet. Liefert z.B. die lokale Aktualisierung die Werte $\Delta x = 500mm$ und $\Delta y = 50mm$ für zwei Positionen, und die globale Aktualisierung $\Delta x = 550mm$ und $\Delta y = 50mm$, so ergibt sich ein Positionsfehler von $50mm$.

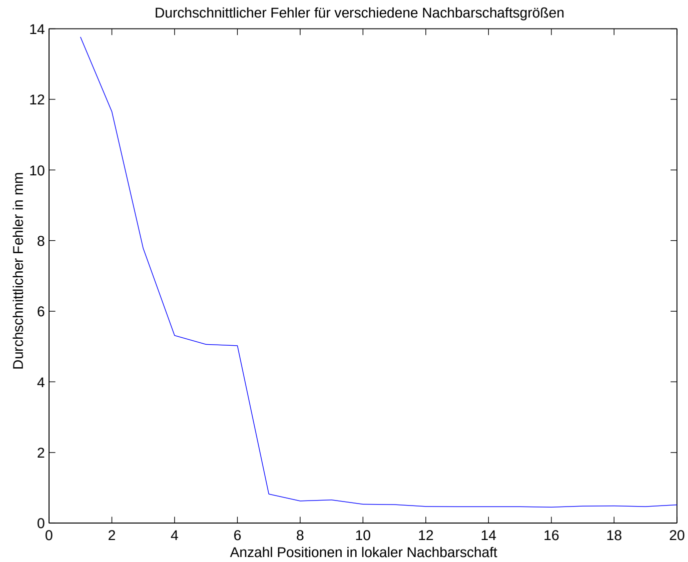


Abbildung 5.9: Durchschnittlicher Positionsfehler in Abhängigkeit der Nachbarschaftsgröße K.

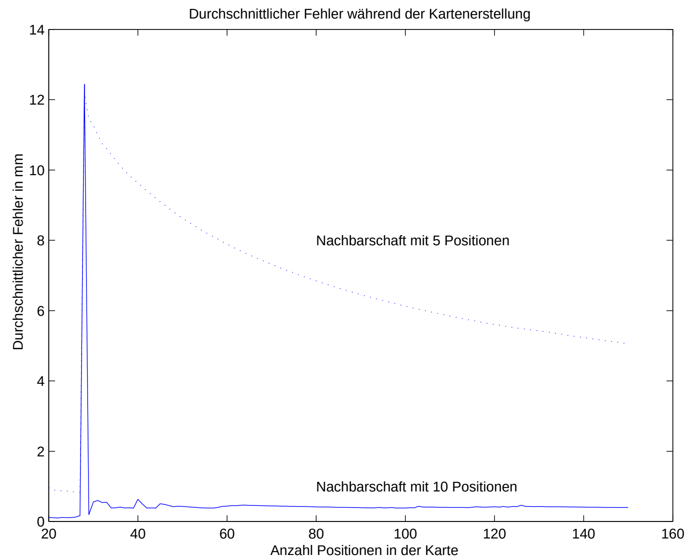


Abbildung 5.10: Durchschnittlicher Positionsfehler während der Erstellung einer Karte aus 150 Positionen.

Anzahl von Knoten begrenzt ist. Insbesondere ist der Rechenaufwand unabhängig von der Kartengröße. Da die Überdeckung der einzelnen Scans sehr effizient ist, ist die gesamte Prozedur sehr schnell und benötigt weniger als $100ms$ in typischen Situationen.

Ein zweiter Punkt ist, daß lokale Aktualisierung wegen der Nichtlinearität von Scan-Matching manchmal Resultate produziert, die stark von dem Ergebnis der globalen Positionsschätzung abweichen. In diesen Fällen verursacht der neue Scan eine kleine Verschiebung in früheren Positionen l_i außerhalb dem lokalen Aktualisierungsbereich. Diese Verschiebung verursacht dann ein ganz anderes Ergebnis bei der Überdeckung des Scans an Position l_i mit einem Nachbarscan. Das Ergebnis der lokalen Aktualisierung ist in diesem Fall nicht unbedingt schlechter, da die globale Aktualisierung in ein lokales Minimum gelaufen sein kann, aus dem es erst später wieder herauskommt. Dieser Aspekt wurde jedoch bisher noch nicht näher untersucht.

Anders verhält es sich bei der Kartenaktualisierung nach Schließen eines Zyklus. Da die konsistente Positionsschätzung ein $O(n^3)$-Prozeß in der Anzahl Knoten ist und für das Schließen des Zyklus alle Positionen entlang des Zyklus betrachtet werden müssen, werden zahlreiche Optimierungen benötigt, damit die Methode effizient wird. Normalerweise hat eine Position nur wenige Verbindungen zu anderen Positionen, welche zudem in der lokalen Umgebung der betrachteten Position liegen. Daher können effiziente lineare Algorithmen, wie z.B. das *LASPack*-System [Skalicky, 1996], eingesetzt werden, welche die spärliche Besetztheit ausnutzen, um konsistente Positionsschätzungen zu berechnen.

Außerdem kann die Größe der Beobachtungsmatrix nach einer Idee von Lu und Milios [1997b] reduziert werden, indem das Netzwerk nach Verbindungen mit geringer Unsicherheit untersucht wird. Eine solche Verbindung wird dann als konstante Beziehung behandelt und die beteiligten Positionen können voneinander abgeleitet werden. Daher kann eine Position entfernt werden, was die Beobachtungsmatrix verkleinert. Obwohl diese Vorgehensweise zu einer suboptimalen Lösung führt, konnte kein großer Unterschied zur optimalen Lösung mit voller Beobachtungsmatrix festgestellt werden, sofern die Beschränkung der maximalen Anzahl von Positionen auf eine relative große Zahl wie z.B. 200 gesetzt wurde.

Alle diese Optimierungen führen dazu, daß in fast allen Fällen das Schließen eines Zyklus weniger als 10 Sekunden selbst für große Zyklen benötigt.

5.5.3 Kartenkorrelation

Um die topologischen Beziehungen von Positionen zum Schließen eines Zyklus zu bestimmen, wird das zuletzt vom Roboter erstellte Kartenstück mit älteren Teilen der bereits erstellten Karte verglichen. An den Stellen, an denen das lokale Kartenstück gut in die Karte paßt, ist es wahrscheinlich, daß die neue Position topologisch mit einer diesen Stellen verbunden ist.

In der gegenwärtigen Version des *LRGC*-Verfahrens ist es nicht möglich, eine einmal eingefügte topologische Verbindung später wieder herauszunehmen, da beim Schließen des Zyklus alle Positionen aktualisiert werden und keine Historie geführt wird. Aus diesem Grund darf das Einfügen einer solchen Verbindung nur dann geschehen, wenn die topologische Beziehung auch wirklich sicher ist. Dies ist der hauptsächliche Grund, warum ein ganzer Satz von Scans mit der Karte verglichen und eine zusätzliche Entscheidungslogik zum Verwerfen von falschen Beziehungen benötigt wird.

Eine weitere Bedingung für den Kartenvergleich ist, daß sie effizient sein muß, da sie die ganze Zeit im Hintergrund läuft, während der Roboter in einem Zyklus zu einer bereits besuchten Position zurückkehrt. Kürzliche Untersuchungen von Konolige und Chou [1999] stellen eine schnelle und genaue Überdeckungstechnik basierend auf Korrelation bereit. Die Rechtfertigung für dieses Verfahren liegt in der Bayes'schen Analyse der Überdeckungswahrscheinlichkeit. Für ein gegebenes lokales Kartenstück r und einer Karte m wird die posteriori Wahrscheinlichkeit $p(l \mid r, m)$, daß sich der Roboter an Position l befindet, gesucht. Durch Anwenden der Bayes'schen Regel erhält man:

$$p(l \mid r, m) = k \cdot p(r \mid l, m) p(l, m) \tag{5.13}$$

Hierbei gibt das Sensormodell $p(r \mid l, m)$ die Wahrscheinlichkeit an, daß man das Kartenstück r von der Roboterposition l aus sieht, gegeben die Karte m. Wie in [Konolige und Chou, 1999] gezeigt wurde, kann das Sensormodell durch einen Korrelationsoperator approximiert werden. Hierbei wird ein Gitter auf die Karte gelegt und für jede Zelle i die Belegtheitswahrscheinlichkeit $p(r_i)$ für das lokale Kartenstück und $p(m_i)$ für die Karte an der Zelle i berechnet. Der Korrelationsoperator ist dann:

$$corr(r, m) = \sum_i p(r_i) p(m_i) \tag{5.14}$$

Für die Praxis ist es günstig, die gesamte Unsicherheit in die Belegtheitswahrscheinlichkeit $p(m_i)$ der Karte zu stecken, was obige Summe vereinfacht und zu einer optimierten Implementierung führt (für Einzelheiten siehe [Konolige und Chou, 1999]).

Abbildung 5.11 zeigt die Korrelationsantwort für ein typisches lokales Kartenstück mit einer typischen Karte. Die gepunktete Umrandung deutet das lokale Kartenstück hinter der aktuellen Roboterposition an. Die Ellipse kreist die Positionen ein, an denen sich der Roboter aufgrund von Odometriefehlern befinden könnte. Und die Korrelationsantwort ist an Stellen mit guter Überdeckung schwarz und wird heller je schlechter das lokale Kartenstück zur alten Karte paßt.

Die priori Wahrscheinlichkeit $p(l, m)$ reduziert sich zu $p(l)$, da die Karte m fest ist. Der Suchbereich von $p(l)$ wird mit einer Gleichverteilung initialisiert. Hier könnte auch eine Gauß-Verteilung aufgrund der Kovarianz der Roboterposition verwendet werden. Der Normalisierungsfaktor k ist schwieriger zu bestimmen. Im allgemeinen sollten sich die Wahrscheinlichkeiten $p(l \mid r, m)$ zu einer Zahl kleiner eins aufsummieren, da die Möglichkeit besteht, daß das lokale Kartenstück an keine Stelle in der Karte paßt, z.B. wenn das aktuelle Kartenstück nicht mit

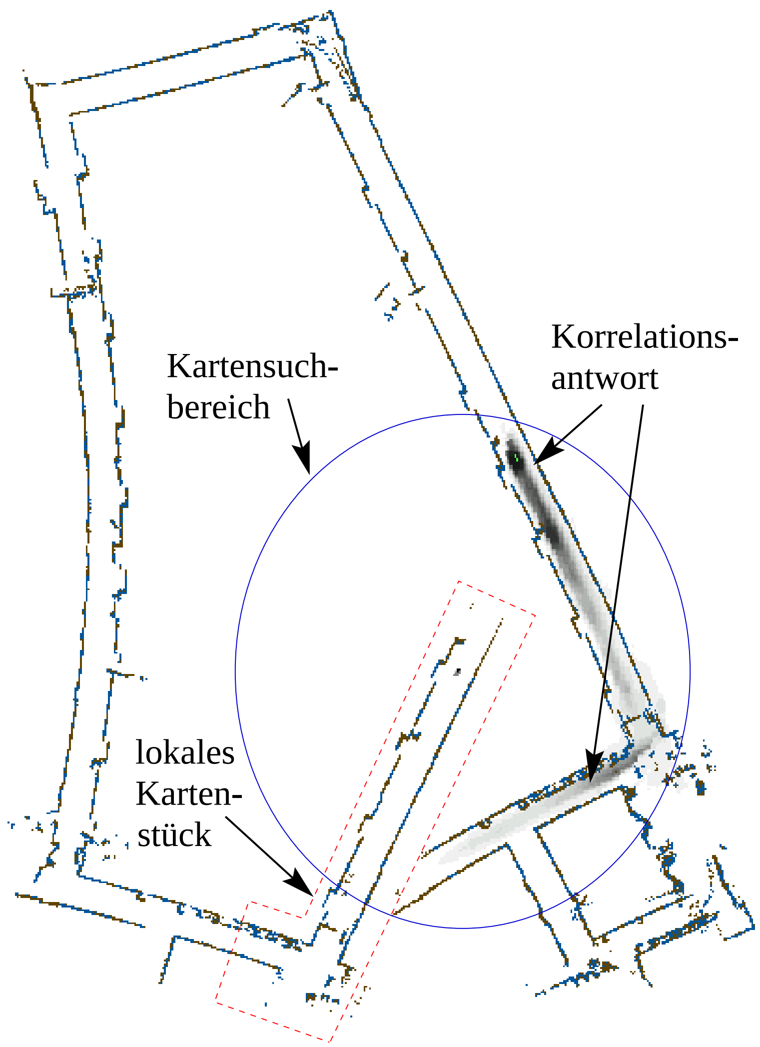


Abbildung 5.11: Kartenkorrelation. Die Korrelationsantwort $p(r \mid l, m)$ ist innerhalb der Ellipse dargestellt.

der bisherigen Karte überlappt. Es ist jedoch sehr schwierig abzuschätzen, wie wahrscheinlich es ist, daß das lokale Kartenstück nirgendwo paßt. Daher wird für die Praxis die Korrelationsantwort normalisiert und einem anschließenden Filter die Entscheidung überlassen, ob eine topologisch eindeutige Relation vorliegt oder nicht. Folgende Bedingung für diesen Filter ergaben gute Resultate:

1. Hoher Überdeckungswert. Der nichtnormalisierte Korrelationswert sollte groß sein.

2. Keine Mehrdeutigkeiten. Falls mehrere Bereiche mit hoher Wahrscheinlichkeit vorhanden sind, so sollte der Unterschied des größten Peaks zum nächstgrößten groß sein.

3. Niedrige Varianz. Der beste Bereich sollte einen steilen Peak besitzen.

Zum Beispiel sind in Abbildung 5.11 mehrere Bereiche mit hoher Wahrscheinlichkeit zu sehen. Der oberste Bereich hat einen sehr hohen Korrelationswert und ist ungefähr fünf mal so hoch wie der nächst beste. Schließlich ist die Varianz dieses Bereichs klein, weniger als $20cm$ in x- und y-Richtung.

Mehrere Faktoren können die Qualität der Korrelationsmethode beeinflussen. Die wichtigsten sind die Größe des lokalen Kartenstücks und der Bereich, der für die Suche nach einer passenden Überdeckung benutzt wird. Der nächste Abschnitt beschreibt, wie diese Parameter gewählt und die einzelnen Techniken zum *LRGC*-Algorithmus zusammengesetzt werden.

5.5.4 LRGC-Algorithmus

Abbildung 5.12 zeigt das Basisschema, das für die Kartenaktualisierung benutzt wird, wenn ein neuer Scan vom Laserscanner aufgenommen wird. Eine Karte wird hier als ungerichteter Graph repräsentiert, wobei Knoten Roboterpositionen mit zugehörigen Scans und Kanten Bedingungen zwischen Position sind, die durch Odometrie, Scan-Matching oder Kartenkorrelation gewonnen werden. Der leere Graph wird als initiale Karte verwendet.

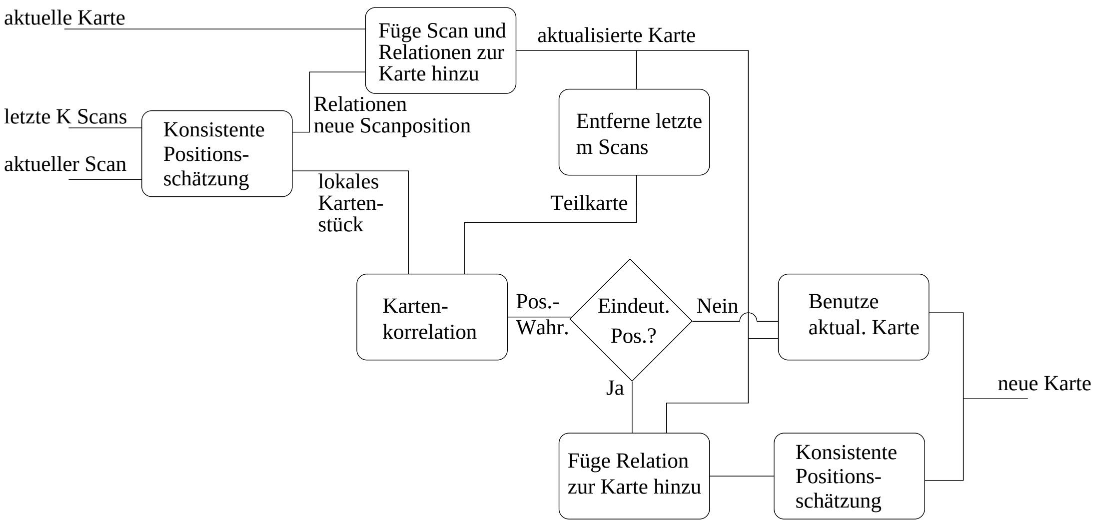


Abbildung 5.12: Datenfluß des *LRGC*-Algorithmus.

Wenn ein neuer Scan zur Karte hinzugefügt wird, so wird er zuerst mit den letzten K Scans (der lokalen Nachbarschaft) registriert, um ihn sauber auszurichten und um die Positionsschätzung der Odometrie zu verbessern. Die neue Scanposition wird dann zusammen mit ihren Beziehungen zur aktuellen Karte hinzugefügt. Hierdurch entsteht die aktualisierte Karte.

Die Erkennung von Zyklen wird im restlichen Teil des Ablaufschemas realisiert. Von der aktualisierten Karte wird ein „altes“ Kartenstück extrahiert, von dem angenommen werden kann, daß es topologisch korrekt ist. Dies wird dadurch realisiert, daß die letzten m Scans (mit $m > K$) entfernt werden, wodurch vermieden wird, daß neuere Scans in der alten Karte vorhanden sind. Ein lokales Kartenstück wird außerdem von den letzten aufgenommenen Scans erstellt und mit der alten Karte korreliert. Die dadurch entstandene Positionswahrscheinlichkeitsverteilung wird dann von dem zuvor beschriebenen Filter untersucht. Sofern der größte Peak den Filter passiert, kann angenommen werden, daß eine topologische Beziehung gefunden wurde. In diesem Fall wird die Relation zur Karte hinzugefügt und konsistente Positionsschätzung verwendet, um den Zyklus zu schließen und um die Karte feinzujustieren.

Um topologische Beziehungen zu finden, wird der Suchbereich auf einen Bereich um die aktuelle Roboterposition herum beschränkt. Dieser Bereich wächst mit der Positionsunsicherheit des Roboters. Die Positionsunsicherheit wurde dabei als Gauß-Verteilung modelliert und für die Korrelation werden nur Positionen untersucht, die einen Mahalanobis-Abstand zur Roboterposition kleiner einer bestimmten Schranke besitzen. Außerdem wird die lokale Kartengröße linear der Positionsunsicherheit angepaßt, um mögliche Mehrdeutigkeiten bei größeren Suchbereichen zu kompensieren. Daher werden große Zyklen erst dann geschlossen, wenn eine genügend große Sicherheit für eine topologische Beziehung vorhanden ist. Nachdem ein Zyklus geschlossen wurde, verringert sich die Positionsunsicherheit automatisch und Suchgröße und Größe des lokalen Kartenstücks verringern sich wieder.

Am Schluß der Kartenerstellung, nachdem alle Scans integriert wurden, kann die Karte noch weiter optimiert werden, indem die konsistente Positionsschätzungsmethode erneut auf alle Positionen angewendet wird.

5.5.5 Ergebnisse

Der soeben vorgestellte Kartenerstellungsalgorithmus wurde in zahlreichen Umgebungen und mit verschiedenen Robotersystemen getestet. Alle in diesem Abschnitt vorgestellten Ergebnisse wurden mit dem *LRGC*-Verfahren erzeugt, das selbständig die Daten aus der realen Welt verarbeitete.

Abbildung 5.13a zeigt Rohdaten, die von einem *B21*-Roboter mit 180° *SICK*-Laserscanner in der Wean Hall der Carnegie Mellon University aufgenommen wurden. Diese Umgebung hat eine Größe von 80 auf 25 Meter und enthält zwei Zyklen, wobei einer der beiden Zyklen eine Länge von ca. 200 Meter besitzt.

Abbildung 5.13b zeigt die Situation vor Schließen des kleinen Zyklus. Da hier der akkumulierte Odometriefehler klein ist, ist das Schließen der Schleife eine einfache Aufgabe (siehe Abbildung 5.13c). Nachdem der Roboter jedoch den großen Zyklus abgefahren hat, ist wie in Abbildung 5.13d zu sehen ist ein signifikanter Odometriefehler angewachsen. Um diesen Zyklus zu schließen, wird ein großer

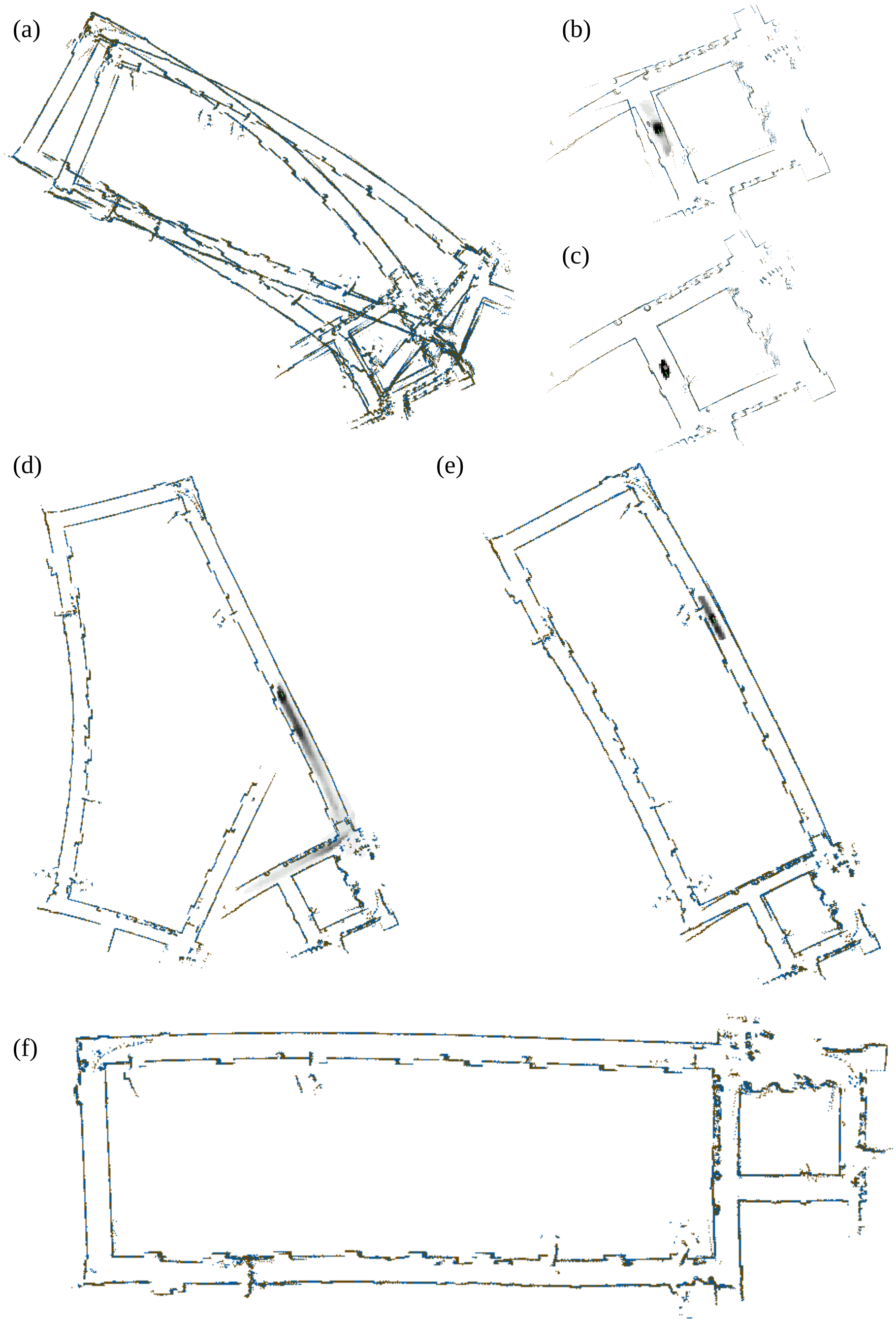


Abbildung 5.13: (a) Durch rohe Odometrie- und Laserdaten entstehende Karte in einer 80 auf 25 Meter großen Umgebung. (b) Vor Schließen des kleinen Zyklus. (c) Nach Schließen des kleinen Zyklus. (d) Vor Schließen des großen Zyklus. (e) Nach Schließen des großen Zyklus. (f) Resultierende Karte.

Suchbereich und ein großes lokales Kartenstück benötigt. Das System ist aber dennoch in der Lage, die Scans korrekt auszurichten (siehe Abbildung 5.13e). Beachtenswert hierbei sind die kleinen Ungenauigkeiten in der linken unteren Ecke. Hier weiß das System noch nicht, daß diese Kartenteile zusammengehören. Nachdem der große Zyklus ein zweites Mal aktualisiert wurde, werden diese Beziehungen erkannt und die Ungenauigkeiten verschwinden. Abbildung 5.13f zeigt die resultierende Karte.

Eine weitere Serie von Experimenten wurde im Artificial Intelligence Center des SRI International mit einem *Pioneer-II*-Roboter mit *SICK*-Laserscanner durchgeführt. Abbildung 5.14a zeigt die rohen Daten eines dieser Experimente. Hier wurden die Odometriedaten durch einen großen Driftfehler, welcher durch einen Teppich mit direktionalen Stoppeln verursacht wurde, stark verrauscht. Der Kartenerstellungsalgorithmus ist trotzdem in der Lage diesen Driftfehler zu korrigieren und eine topologisch korrekte und genau Karte zu berechnen (siehe Abbildung 5.14b).

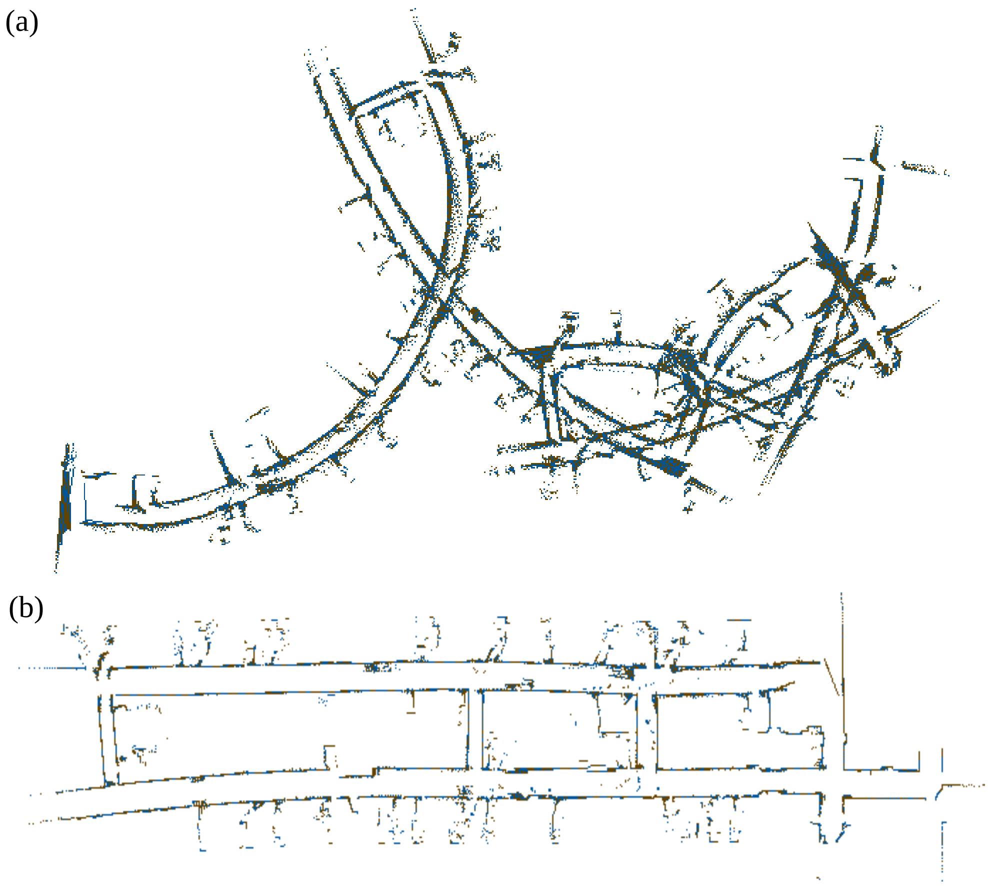


Abbildung 5.14: (a) Rohdaten mit großen Driftfehler in einer 85 auf 15 Meter großen Umgebung. (b) Berechnete Karte.

Abbildung 5.15 zeigt weitere Läufe des *LRGC*-Verfahrens mit Daten des Carnegie Museum of Natural Science, welche von einem *B21*-Roboter aufgenommen wurden, und mit Daten in der Abteilung „Grundlagen der künstlichen Intelligenz“ der Universität Freiburg, die von einem *Pioneer-I*-Roboter mit *SICK*-Laserscanner erfaßt wurden.

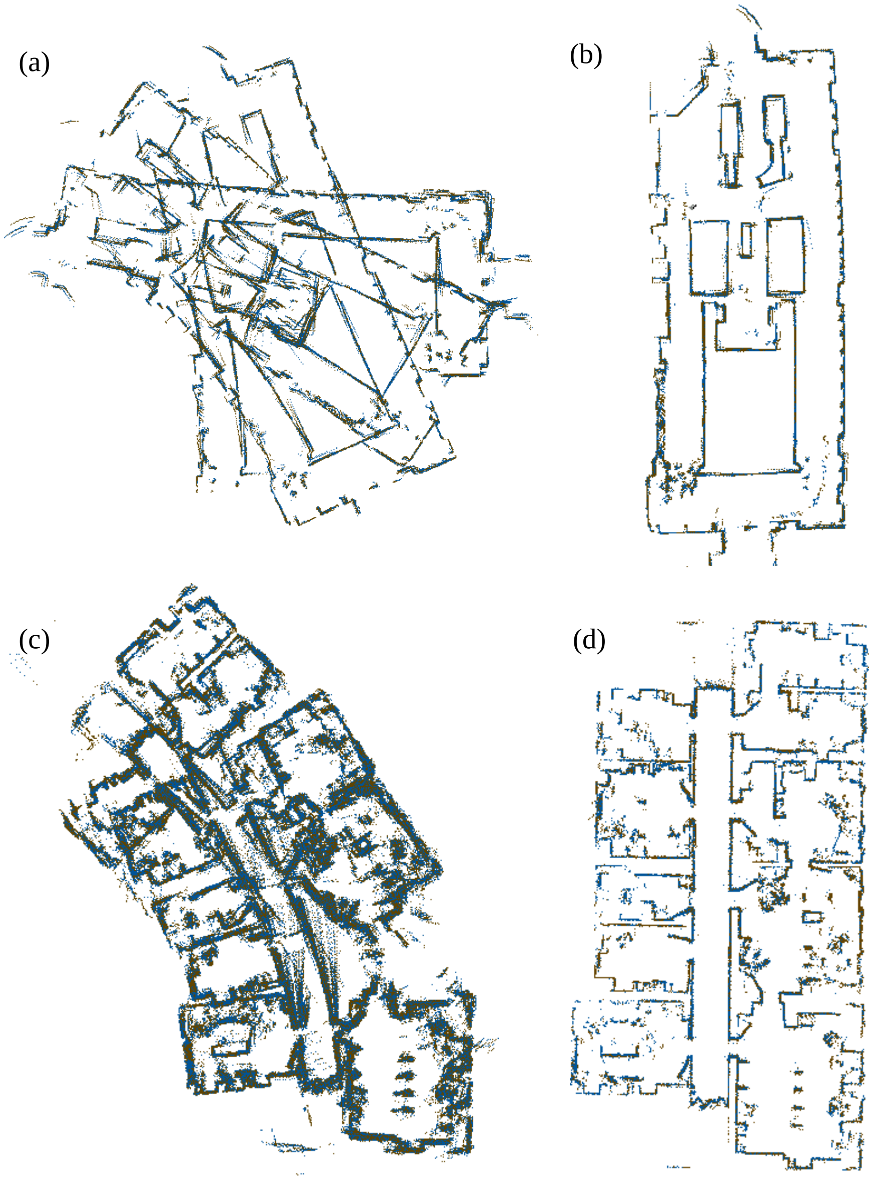


Abbildung 5.15: Kartenerstellung in weiteren Umgebungen. (a) Rohdaten des Carnegie Museum of Natural Science (45 auf 15 Meter). (b) Erstellte Karte. (c) Rohdaten der KI-Abteilung der Universität Freiburg. (24 auf 13 Meter). (d) Erstellte Karte.

5.5.6 Bewertung

In diesem Abschnitt wurde die *Local Registration and Global Correlation* Methode für die Erstellung von Karten in großen, zyklischen Umgebungen präsentiert. Das Verfahren benutzt drei verschiedene Techniken, um inkrementell Karten zu erstellen, topologische Beziehungen zu finden und Zyklen zu schließen. Mehrere Beispielläufe in verschiedenen Umgebungen und mit verschiedenen Robotersystemen wurden durchgeführt, welche die Möglichkeiten und die Genauigkeit des Verfahrens demonstrieren.

Das Verfahren ist inkrementell: Scans werden in Echtzeit in die Karte integriert, nachdem sie vom Laserscanner aufgenommen wurden. Nur beim Schließen eines Zyklus wird zusätzlicher Rechenaufwand benötigt. Dies ist nach Wissen des Autors, das erste selbständige Echtzeit-Kartenerstellungssystem, welches dichte, genaue metrische Karten in großen, zyklischen Umgebungen erstellt.

Einige wichtige Voraussetzungen dieses Ansatzes sollten festgehalten werden. Zum einen benötigt der Algorithmus gute Scan-Matching-Resultate, um genaue lokale Kartenstücke zu produzieren. Zum Beispiel kann die Kartenkorrelation keine steilen Peaks berechnen, wenn das lokale Kartenstück in einem geraden Korridor schiefe oder runde Wände erzeugt, da keine genügend große Überlappung mehr vorhanden ist. In diesen Situationen ist es auch schwierig andere Techniken wie z.B. Markov-Lokalisierung [Burgard *et al.*, 1996; Konolige und Chou, 1999] einzusetzen, da die robuste Erkennung von falschen topologischen Beziehungen durch große lokale Kartenstücke nicht mehr möglich ist. Ein möglicher Ansatz ist, mehrere kleinere lokale Kartenstücke zu verwenden.

Eine weitere Voraussetzung für den *LRGC*-Algorithmus ist die effiziente und robuste Erkennung der topologischen Zusammenhänge, da die von der Kartenkorrelation und dem daran anschließenden Filter erkannten Beziehungen auch den tatsächlichen Gegebenheiten entsprechen müssen. Falls der Filter zuviele gute Überdeckungen verwirft, so werden nicht genug Relationen zum Schließen von Zyklen generiert, falls er eine falsche Zuordnung akzeptiert, so wird die Karte inkonsistent. Während der in dieser Arbeit vorgeschlagene Filter in den vorgestellten Experimenten gut funktionierte, können noch weitere Techniken wie visuelle Referenzen oder 3d-Information hinzugenommen werden. Im unvermeidbaren Fall, daß eine falsche Überdeckung akzeptiert wird, kann die strikte Identifikation topologischer Beziehungen entschärft werden, indem mehrere Hypothesen verfolgt und eine Erkennung von falsch geschlossenen Zyklen hinzugefügt wird.

Schließlich sind die diskutierten Techniken auch anwendbar zur gemeinsamen Kartenerstellung für ein Team von Robotern. Jeder Roboter kann seine eigene lokale Karte erstellen und mit anderen Robotern über eine drahtlose Verbindung kommunizieren. Selbst wenn die Roboter anfangs keine Vorstellung über ihre relative Position zueinander besitzen, können sie durch Korrelation ihres lokalen Kartenstücks mit der Karte eines anderen Roboters ihre Position bestimmen und die Karten ähnlich wie beim Schließen eines Zyklus miteinander verschmelzen.

Kapitel 6

Pfadplanung

In den beiden vorherigen Kapiteln wurde beschrieben, wie ein Roboter seine Position anhand einer Karte der Umgebung bestimmen kann, und wie er diese Karte aus Daten einer Explorationsfahrt erstellen kann. Um ein vollständiges Navigationssystem zu erhalten, bei dem das Robotersystem autonom in seiner Umgebung agiert, muß das System auch noch in der Lage sein, Wege zu bestimmten Orten zu planen und abzufahren, d.h. der Roboter benötigt eine Pfadplanungskomponente. Dies ist Gegenstand dieses Kapitels.

Das Problem der Pfadplanung kann wie folgt formuliert werden. Gegeben sind eine Startposition (die augenblickliche Position des Roboters), eine Zielposition und eine Umgebungskarte. Gesucht ist eine Abfolge von Aktionen, z.B. eine Folge von Zwischenpositionen, die den Roboter störungsfrei (also z.B. ohne mit Hindernissen zusammenzustoßen) von der Start- zur Zielposition bewegen. Dieses Problem wurde ausführlich von Latombe [1991] untersucht. Dabei wurden zahlreiche Lösungsansätze vorgestellt und Komplexitätsabschätzungen bestimmt.

Eine Idee, das Problem der Bewegungsplanung zu abstrahieren, liegt in der Einführung eines Konfigurationsraumes. Der Konfigurationsraum besteht aus allen möglichen Positionen (Ort und Orientierung), an denen sich der Roboter in seiner Umgebung befinden kann. Durch diese Modellierung kann der Roboter als ein Punkt modelliert werden und die Wegeplanung reduziert sich auf die Planung eines Punktes im Konfigurationsraum.

Für die Bewegungsplanung in der Ebene ist der Konfigurationsraum 3 dimensional. Ist der Roboter ein Zylinder und kann sich auf der Stelle drehen, d.h. zu jeder Zeit in jede beliebige Richtung fahren, so kann der Konfigurationsraum auf 2 Dimensionen reduziert werden, da die Orientierung des Roboters keine Rolle mehr spielt. In vielen Verfahren wird daher der Roboter als kreisförmig angenommen oder die Form des Roboters zu einem Kreis ausgedehnt. In dieser Arbeit soll diese Annahme ebenfalls verwendet werden, um ein neues, zweistufiges Pfadplanungsverfahren zu entwerfen.

Der Rest dieses Kapitels ist wie folgt aufgebaut. Zunächst werden verschieden Ansätze der Bewegungsplanung wie sie in [Latombe, 1991] aufgeführt sind vorgestellt.

Danach wird auf einen Ansatz, der die Umwelt in Zellen gleicher Größe unterteilt, näher eingegangen und gezeigt, daß für große Umgebungen ein solches Verfahren nicht praktikabel ist.

Anschließend wird ein neues, zweistufiges Verfahren beschrieben, das aus einer Umgebungskarte von Laserscans eine topologische *Wegekarte* erstellt, auf der effizient Wege über längere Distanzen geplant werden können. Für die lokale Planung zu einzelnen Zwischenpunkten wird ein gitterbasierter Ansatz mit begrenztem Suchraum benutzt. Hierdurch entsteht ein effizientes und robustes Pfadplanungssystem, das dynamischen Hindernissen reaktiv ausweicht, nichtpassierbare Wege erkennt und alternative Wege findet. Für das Verfahren wird ein Beispiellauf mit einem *Pioneer-I*-Roboter präsentiert, bei dem das System mehrere Male einen neuen Weg planen muß. Weiterhin werden verwandte Verfahren vorgestellt und mit dem neuen Verfahren verglichen. Schließlich wird das Verfahren bewertet und es werden potentielle Probleme beschrieben.

6.1 Ansätze zur Pfadplanung

Es gibt eine Vielzahl von Verfahren zur Planung der Bewegung eines mobilen Roboters. Die meisten dieser Verfahren lassen sich in eine der folgenden drei Kategorien einteilen [Latombe, 1991]:

Wegekarten-Ansatz: Beim Wegekarten-Ansatz wird aus einer Umgebungskarte ein Graph erstellt, dessen Kanten aus hindernisfreien Wegen bestehen. Die Kanten sind quasi „Standardwege", auf denen der Roboter navigieren kann. Das Pfadplanungsproblem reduziert sich hierdurch auf das Verbinden von Start- und Zielposition zu Punkten in der Wegekarte und der Suche nach einem Pfad in diesem Graphen. Das Verfahren, welches in Abschnitt 6.3 vorgestellt wird, benutzt einen solchen Ansatz.

Zellunterteilungs-Ansatz: Hier wird der hindernisfreie Konfigurationsraum in einfache Regionen, genannt Zellen, eingeteilt, so daß der Wechsel von einer Zelle zu einer benachbarten durch einen einfachen Pfad realisiert werden kann. Die Planung eines Pfades geschieht dann, indem eine Folge von benachbarten Zellen gesucht wird, welche den Roboter von der Start- zur Zielposition führen. Es gibt eine ganze Reihe von Varianten, die diesen Ansatz verfolgen, z.B. können die Zellen exakt die freie Umgebung einschließen oder die Zellen können eine feste Größe haben. In Abschnitt 6.2 wird eine Variante, welche Zellen fester Größe verwendet, genauer beschrieben und Probleme aufgezeigt.

Potentialfeld-Ansatz: Die Idee dieses Ansatzes ist, den Roboter als Partikel in einem künstlichen Potentialfeld wandern zu lassen. Das Potentialfeld wird dabei von der Zielposition, welche eine anziehende Kraft, und den Hindernissen, welche abstoßend wirken, induziert. Der negierte Gradient dieser Potentialfunktion wirkt dann eine Kraft auf den Roboter aus und die Richtung dieser Kraft wird als vielversprechendste Richtung für die Roboterbewegung betrachtet. Potentialfeldmethoden können sehr effizient sein und eignen sich dadurch auch für die Bewegungsplanung im 3 dimensionalen Raum [Braun und Corsépius, 1996; Braun, 1995]. Jedoch haben diese Methoden den gravierenden Nachteil, daß der Roboter in lokale Minima laufen kann, aus denen er nur durch Einsatz weiterer Mechanismen wieder herausgelangen kann. Aus diesem Grund wird dieser Ansatz hier nicht weiter verfolgt.

Im folgenden wird zunächst ein gitterbasierter Ansatz zur Bewegungsplanung vorgestellt und Probleme dieses Ansatzes beschrieben. Danach wird ein neues, zweistufiges Verfahren zur Pfadplanung vorgestellt, das eine Wegekarte aus Laserdaten erstellt.

6.2 Gitterbasierte Pfadplanung

Viele Verfahren zur Wegeplanung eines mobilen Roboters unterteilen den Konfigurationsraum in Zellen gleicher Größe. Auf dem so entstehenden Gitter können dann mittels einer Suchstrategie Pfade von Start- zu Zielpositionen gefunden werden.

Nimmt man an, daß der Roboter durch einen Zylinder approximiert werden kann und in der Lage ist, sich im Stand in jede beliebige Richtung zu drehen, so schrumpft der Konfigurationsraum zu einem 2 dimensionalen Raum, da die Orientierung des Roboters keine Rolle mehr spielt. Die Zellen des Konfigurationraumes, die durch Hindernisse blockiert sind, können nun auf einfache Weise bestimmt werden, indem alle Hindernisse um den Radius des Roboters vergrößert werden[1] und die zugehörigen Zellen als belegt markiert werden. Der Suchalgorithmus muß dann nur noch einen Pfad innerhalb der nichtbelegten Zellen finden. Als Suchverfahren kann eine Breitensuche [Knick und Schlegel, 1994; Corsépius *et al.*, 1997] oder dynamisches Programmieren [Thrun *et al.*, 1998a] verwendet werden.

Abbildung 6.1 zeigt ein Beispiel für diesen Bewegungsplaner. Hier wurden Scandaten in ein Belegtheitsgitter eingetragen (dunkle Zellen) und alle Hindernisse um den Radius eines fiktiven Roboters vergrößert (dunkel umrandete Zellen). Anschließend wurde ein Pfad von einer Position im rechten oberen Raum zu einer Position im linken oberen Raum durch Anwenden einer Breitensuche geplant.

Belegtheitsgitter haben den Vorteil, daß sie einfach zu erstellen und zu verwalten sind und kürzeste Wege über sie bestimmt werden können. Sie sind außerdem

[1] In der Literatur wird diese Vorgehensweise *obstacle growing* genannt.

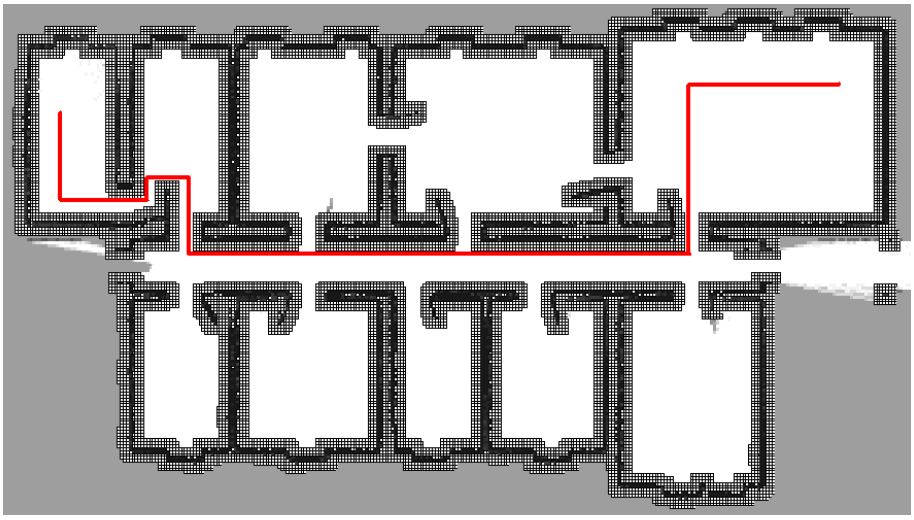


Abbildung 6.1: Wegeplanung auf einem Gitter. Hindernisse werden um den Radius des Roboters vergrößert in das Gitter eingetragen.

korrekt und vollständig, d.h. gefundene Wege sind auch tatsächlich befahrbar, es wird immer eine Lösung gefunden, sofern es eine gibt, und das Verfahren erkennt Situationen, in denen es keine Lösung gibt.

In großen Einsatzumgebungen benötigen Belegtheitsgitter jedoch viel Speicherplatz und die Wegeplanung wird wegen des großen Suchraumes ineffizient. Daher sollte dieser Ansatz nur in kleinen Umgebungen gewählt werden. Um auch in großen Umgebungen navigieren zu können, empfiehlt es sich ein zweistufiges Verfahren zu verwenden, wie es im nächsten Abschnitt vorgestellt wird.

6.3 Pfadplanung auf Sichtbarkeitsgraphen

In diesem Abschnitt wird eine neues Verfahren zur Pfadplanung eines mobilen Roboters beschrieben, das aus Laserdaten eine topologische Karte der Umgebung berechnet. Topologische Karten haben den Vorteil, daß sie planungseffizient sind und wenig Speicherplatz benötigen. Außerdem wurde in [Thrun und Bücken, 1996] festgestellt, daß die Wegeplanung auf topologischen Karten nicht unbedingt wesentlich längere Wege bestimmt als die gitterbasierte Wegeplanung. Die Problematik topologischer Karten ist jedoch ihre automatische Erstellung aus Sensorinformation.

Das in diesem Abschnitt beschrieben Verfahren berechnet aus einem Satz sich korrekt überdeckender 360°-Scans eine topologische Karte der Umgebung. Diese Scans können durch eines der im vorigen Kapitel vorgestellten Verfahren zur Kartenerstellung geliefert werden. Im Falle, daß keine Rundumscans vorliegen, z.B. nur 180°-Scans, können nahe zusammenliegende Scans zu einem Scan zusammengefaßt werden, indem sie auf einen Punkt (z.B. auf die Aufnahmeposition eines Scans) projiziert, nach Aufnahmewinkel sortiert, und durch Einsatz des Winkelreduktionsfilters aus Abschnitt 3.4.3 mit einem gleichmäßigen Winkelabstand versehen werden.[2]

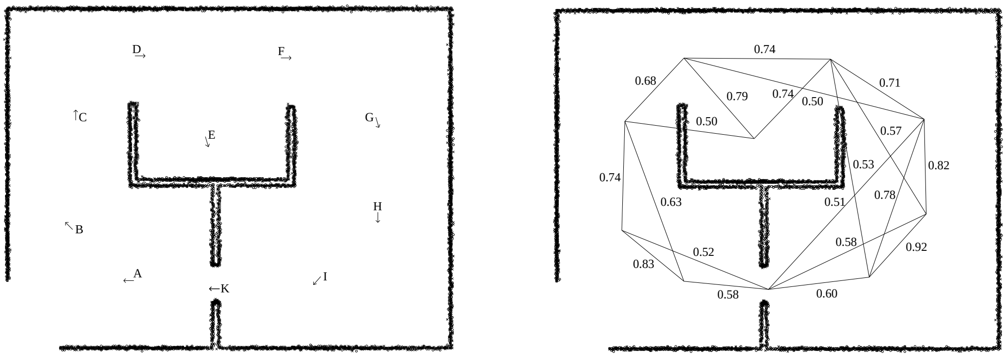
Die Laserscans können gleichzeitig als Referenzscans für die Selbstlokalisierung des Roboters nach dem Verfahren aus Abschnitt 4.4.2 benutzt werden, so daß eine gemeinsame Karte für Wegeplanung und Selbstlokalisierung verwendet wird.

Für die Wegeplanung wird aus den Scandaten eine topologische Karte berechnet, welche Scanpositionen miteinander verbindet. Auf diese Weise entsteht eine kompakte Wegekarte, auf der effizient geplant werden kann. Für die lokale Navigation zwischen den Scanpositionen wird der gitterbasierte Wegeplaner aus dem vorigen Abschnitt verwendet, welcher in der Lage ist, Hindernissen dynamisch auszuweichen und nicht befahrbare Wege zu erkennen.

Der nächste Abschnitt beschreibt, wie aus den Scandaten eine topologische Karte berechnet werden kann.

6.3.1 Sichtbarkeitsgraph

Im folgenden wird angenommen, daß das Robotersystem in einem initialen Explorationsschritt die gesamte Einsatzumgebung befahren und einen Satz von 360°-Laserscans gesammelt hat. Die Aufnahmepositionen der Scans sollen möglichst gleichverteilt über die Umgebung sein. Abbildung 6.2a zeigt einen solchen Satz von 10 simulierten Scans.



(a) (b)

Abbildung 6.2: Simulierte Laserscans (a) und zugehöriger Sichtbarkeitsgraph (b).

[2]Natürlich muß hierbei auch der volle Sichtbereich von 360° abgedeckt sein, was z.B. durch Drehen das Roboters an den einzelnen Stellen oder durch Abfahren der Umgebung in beiden Richtungen erfolgen kann.

Für alle Paare von Scans werden die gemeinsamen Scanpunkte durch Anwendung des in Abschnitt 3.4.5 beschriebenen Projektionsfilters bestimmt. Das Verhältnis der Anzahl gemeinsamer Punkte zur Gesamtanzahl Punkte zweier Scans wird *Sichtbarkeit* genannt.

Eine hohe Sichtbarkeit zweier Scans bedeutet, daß die Wahrscheinlichkeit, von einer Position zur anderen mittels einem lokalem Manöver fahren zu können, hoch ist. Dies begründet sich durch die folgende Überlegung. Falls zwei Scans einen gemeinsamen Punkt besitzen, so existiert für einen zu einem Punkt geschrumpften Roboter ein Weg von der einen Aufnahmeposition zur anderen über den gemeinsamen Punkt. Für reale Roboter wird jedoch ein entsprechend breiter Durchgang zwischen den Aufnahmepositionen benötigt, welcher umso wahrscheinlicher ist, je mehr gemeinsame Scanpunkte vorliegen.

Aus den berechneten Sichtbarkeiten wird ein ungerichteter Graph gebildet, dessen Knoten die Scanpositionen und die Kanten die zugehörigen Sichtbarkeiten sind. Um die Anzahl der Kanten möglichst klein zu halten, werden nur Sichtbarkeiten eingetragen, die größer als ein vorher definierter Mindestwert sind. Abbildung 6.2b zeigt den für die 10 simulierten Scans so entstandenen Graph mit einer Mindestsichtbarkeit von 50 Prozent.

Für jede Kante i mit Sichtbarkeit v_i wird die Wahrscheinlichkeit p_i, von der einen Scanposition zur anderen mit einem lokalen Manöver fahren zu können, durch die Heuristik $p_i = v_i^{kd_i}$ abgeschätzt, wobei d_i der euklidische Abstand der Scanpositionen und k eine Konstante zur Normalisierung ist. Bei gegebener Start- und Zielposition sucht der Wegeplaner nun den Pfad mit maximaler Gesamtwahrscheinlichkeit. Nimmt man an, daß alle Einzelwahrscheinlichkeiten voneinander unabhängig sind, so wird der Pfad so gelegt, daß das Produkt

$$\prod_i p_i \quad = \quad \prod_i v_i^{kd_i}$$

über alle beteiligten Kanten maximal wird. Die Bestimmung eines solchen Pfades kann durch Verwendung von uniformer Kostensuche effizient implementiert werden und ist unabhängig von k.

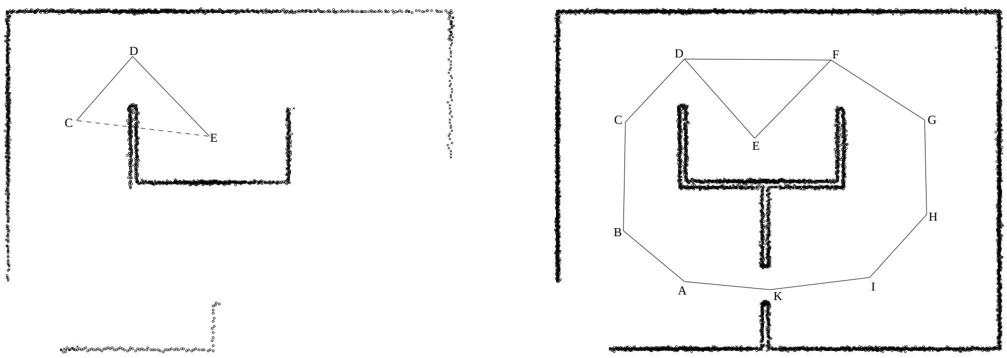
Die verwendete Schätzung der Wahrscheinlichkeiten bewirkt, daß der Planer z.B. für den Weg von Position C zu Position E nicht den direkten Weg sondern den Umweg über Position D wählt. Gleichzeitig werden jedoch auch unnötig lange Pfade vermieden, z.B. legt der Planer bei der Aufgabe von Position A zu Position I zu fahren, den Weg nicht über die Positionen B, C, D, F, G, H, sondern wählt den wesentlich kürzeren Weg über Position K.

Stellt sich während der Planausführung heraus, daß der gewählte Weg nicht befahrbar ist, z.B. wenn der schmale Durchgang an Position K durch ein Hindernis blockiert ist, so wird die nicht befahrbare Kante entfernt und ein neuer Pfad ausgehend von der zuletzt besuchten Position erstellt.

6.3.2 Entfernung unnötiger Kanten

Durch eine einfache Überlegung kann die Anzahl der Kanten im Sichtbarkeitsgraph verkleinert werden. Für jedes Knotentripel, das räumlich nahe genug beieinander liegt, wird der zugehörige Teilgraph untersucht (siehe Beispiel in Abbildung 6.3a). Eine Kante wird entfernt, wenn die zugehörige Wahrscheinlichkeit kleiner als das Produkt der Wahrscheinlichkeiten der beiden anderen Kanten ist, da in diesem Fall der Wegeplaner immer den längeren Weg wählt. Die Kante mit geringer Wahrscheinlichkeit käme nur in Betracht, wenn eine der beiden anderen Kanten durch ein Hindernis blockiert ist und eine Neuplanung stattfinden muß. Da nur Knotentripel betrachtet werden, die genügend nahe zusammen liegen, würde der lokale Wegeplaner aber für diese Kante keine neuen Wege finden. Daher kann die Kante $\overline{\mathrm{CE}}$ in Abbildung 6.3a entfernt werden.

Werden alle mit dieser Methode bestimmten unnötigen Kanten entfernt, so entsteht der Graph aus Abbildung 6.3b, welcher *Wegegraph* genannt wird.



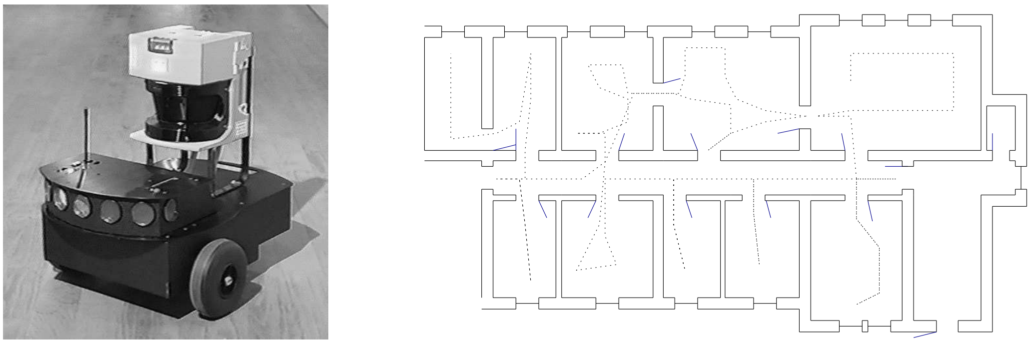
(a) (b)

Abbildung 6.3: Entfernung unnötiger Kanten aus dem Sichtbarkeitsgraph. Betrachtung eines Knotentripels (a) und entstandener Wegegraph (b).

6.3.3 Ergebnisse

Das in diesem Abschnitt beschriebene Verfahren wurde auf einem Roboter des Typs *Pioneer 1*, auf den ein *SICK PLS* 200 Laserscanner montiert wurde, implementiert und getestet. Das Fahrzeug und ein CAD-Modell der 24 auf 13 Meter großen Einsatzumgebung sind in Abbildung 6.4 zu sehen.

Roboter und Laserscanner sind über je ein Modem mit einer Sparc Ultra 1 Station verbunden, die das Fahrzeug über die Saphira-Umgebung [Konolige *et al.*, 1997] steuert. Das Fahrzeug verfügt über zwei einzeln angetriebene Vorderräder, die mit je einer Odometrie zur inkrementellen Positionsbestimmung bestückt sind. Das Fahrzeug ist in der Lage auf der Stelle zu drehen.



(a) (b)

Abbildung 6.4: Pioneer Roboter mit SICK Laserscanner (a) und CAD Modell der Einsatzumgebung mit Explorationspfad (b).

Der Laserscanner hat eine Entfernungsauflösung von $50mm$ und eine Winkelauflösung von 0.5°. Bei einem Blickfeld von 180° werden pro Scan 361 Meßwerte innerhalb $40ms$ bestimmt. Durch die relativ langsame Funkübertragung von 38400 baud können jedoch nur ca. 3 Scans pro Sekunde übertragen werden.

Das Fahrzeug wurde in einer Explorationsfahrt ausgehend vom rechten oberen Raum durch die gesamte Einsatzumgebung bewegt (siehe Abbildung 6.4b). An mehreren Stellen entlang der Fahrt wurden Scans aufgenommen. Da das Blickfeld des Scanners nur 180° beträgt, wurde der Roboter an den Aufnahmestellen in mehreren Schritten um 360° gedreht, damit die aufgenommenen Scans später zu vollen 360°-Scans zusammengesetzt werden können.

Die Aufnahmepositionen der Scans wurden anschließend mit der Methode der konsistenten Positionsschätzung aus Abschnitt 5.3 korrigiert, wodurch eine konsistente Umgebungskarte entstand. Nach dieser Korrektur wurden Scans, deren Aufnahmepositionen dicht beieinander lagen, zu einem neuen Scan zusammengefaßt und Scans mit weniger als 360°-Sichtfeld entfernt. Der so entstandene Satz von 85 Scans und der daraus berechnete Wegegraph mit 123 Kanten sind in Abbildung 6.5 dargestellt.

Realisierung

Dem Robotersystem wird die Karte von 360°-Scans, der Wegegraph und die initiale Fahrzeugposition als Vorabinformation übergeben. Das System führt einmal pro Sekunde eine Selbstlokalisierung durch Überdecken des aktuellen Scans mit dem am nächsten zur aktuellen Position liegenden 360°-Scan nach dem Verfahren aus Abschnitt 4.4.2 durch. Durch Verwendung von 360°-Scans als Referenzscans wird sichergestellt, daß die zu überdeckenden Scans immer einen genügend großen gemeinsamen Sichtbereich haben.

Wird dem System eine Zielposition übergeben, so wird von der aktuellen Po-

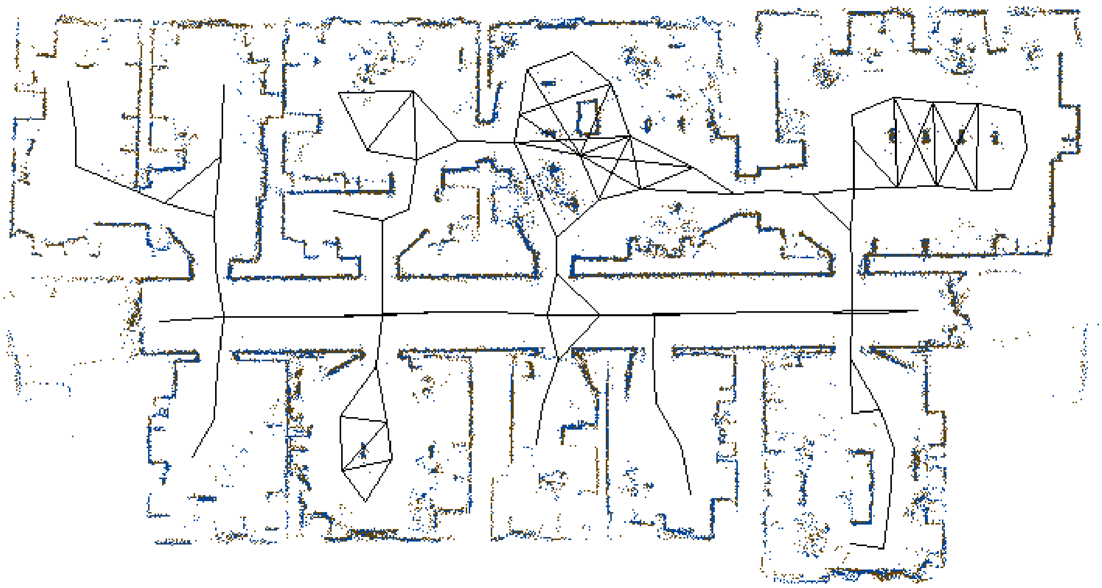


Abbildung 6.5: Korrigierte Karte von Scans mit Wegegraph.

sition aus ein Pfad im Wegegraph gesucht und die einzelnen Zwischenpositionen einem lokalen Wegeplaner übergeben. Der lokale Wegeplaner benutzt ein kleines Belegtheitsgitter hoher Auflösung, dessen Größe je nach Entfernung der anzufahrenden Position dynamisch gewählt wird. Für die Zellgröße wurde eine relativ kleine Kantenlänge von $50mm$ gewählt. Das Gitter wird durch Eintragen der letzten 5 aufgenommenen Scans vorbelegt. Für die lokale Wegeplanung kam der Algorithmus mit Breitensuche aus [Knick und Schlegel, 1994] zum Einsatz. Durch die geringe Größe des Gitters erfolgt die lokale Wegeplanung in Echtzeit. Treten Hindernisse während der Fahrt auf, so werden diese nach Eintragen in das Belegtheitsgitter erkannt und eine Neuplanung lenkt den Roboter um das Hindernis herum. Stellt der lokale Wegeplaner nach Absuchen des gesamten Gitters fest, daß innerhalb des lokalen Bereichs kein Weg vorhanden ist, so bricht er ab. Der globale Planer entfernt dann temporär die nicht befahrbare Kante und führt von der zuletzt besuchten Position aus eine Neuplanung durch.

Beispielfahrt

Das System wurde im rechten oberen Raum gestartet und die Aufgabe gegeben, an eine Position im links davon liegenden Büro zu fahren. Zuvor wurde die direkte Verbindungstür geschlossen, die Tür vom Gang zum Büro durch ein Hindernis blockiert und alle anderen Türen geöffnet (Abbildung 6.6).

Der Roboter wählte zunächst den direkten Weg durch die Tür zum Büro, stellte fest, daß der Weg durch die Tür nicht befahrbar war, und plante einen neuen Weg über die Tür vom Gang zum Büro (Abbildung 6.7).

Abbildung 6.6: Wegeplan vom Praktikumsraum ins Büro.

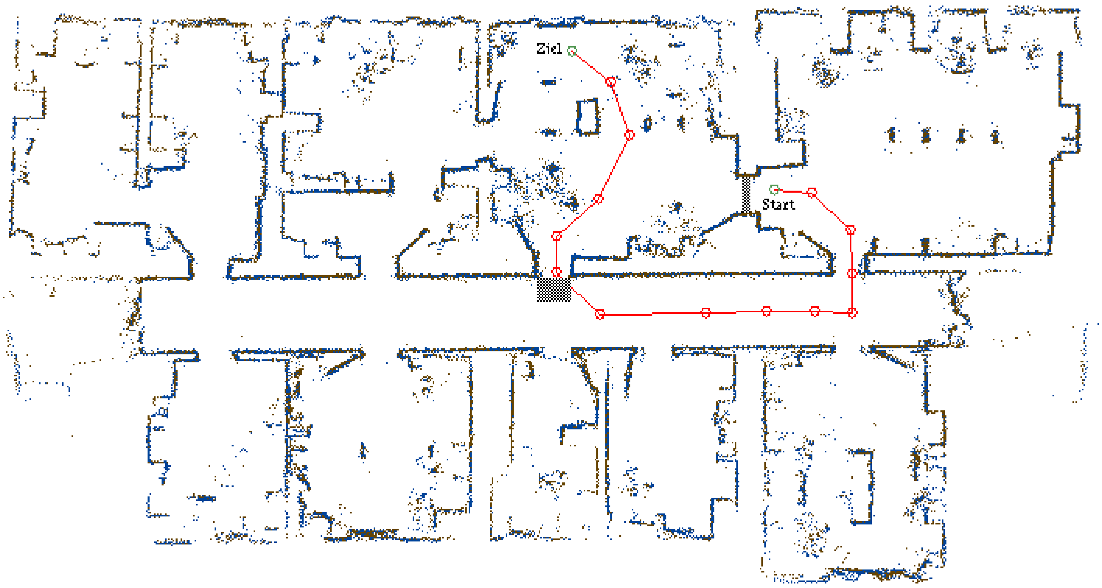


Abbildung 6.7: Neuer Plan vom Praktikumsraum über den Gang ins Büro.

Vor der Tür auf dem Gang angekommen versuchte der Roboter die rechte, schräg verlaufende Kante durch das Hindernis abzufahren. Wieder brach der lokale Planer ab und der globale Planer bestimmte einen neuen Weg über die etwas weiter links liegende Kante. Nach Erreichen der Startposition dieser Kante stellte der lokale Planer durch die Vorbelegung des Gitters mit den letzten 5 Scans sofort fest, daß auch dieser Weg nicht befahrbar ist. Eine erneute globale Neuplanung schließlich lenkte den Roboter über das links vom Büro liegenden Sekretariat zur Zielposition (Abbildung 6.8). Dieser Pfad war durch keine weiteren Hindernisse belegt und der Roboter konnte das Ziel erreichen.

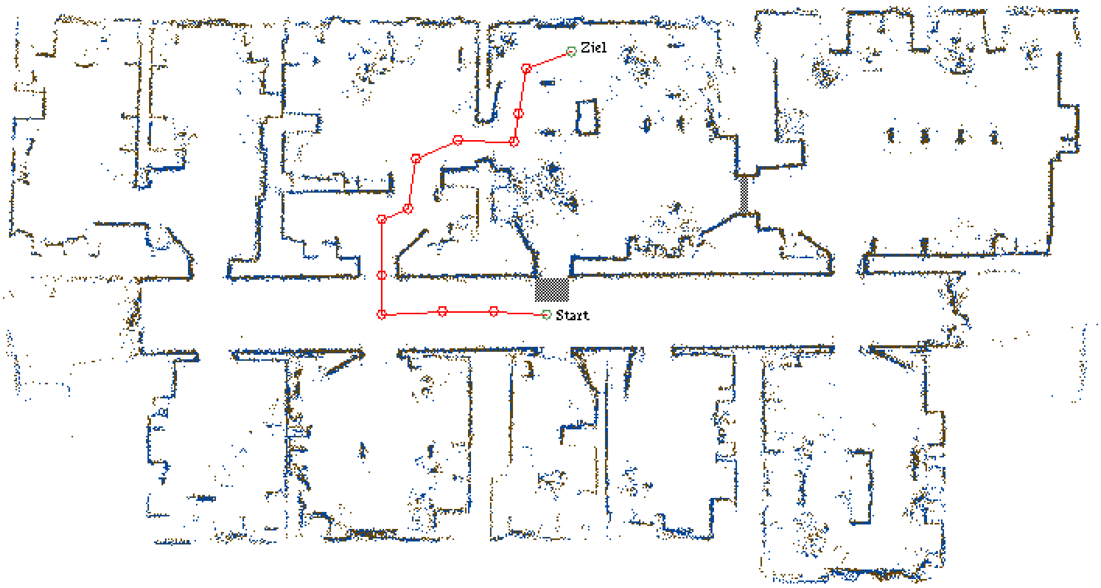


Abbildung 6.8: Neuer Plan vom Gang über das Sekretariat ins Büro

6.3.4 Diskussion

Das vorgestellte Robotersystem wurde in weiteren Beispielfahrten erfolgreich erprobt. Durch das Zusammenspiel des lokalen reaktiven Wegeplaners mit dem hier vorgestellten globalen Wegeplaner wird ein hoher Grad an Robustheit erlangt.

Durch Verwendung eines globalen Planers auf dem Wegegraph werden in der Regel keine kürzesten Wege gefunden, da einerseits eine Diskretisierung auf eine kleine Menge von Positionen stattfindet und andererseits Wege so gewählt werden, daß die Gesamtwahrscheinlichkeit für die Befahrbarkeit des geplanten Weges maximal wird.

In [Thrun und Bücken, 1996] wurde festgestellt, daß der dort beschriebene topologische Planer im Mittel nur unwesentlich längere Wege berechnet als ein gitterbasierter Wegeplaner. Dies bestärkt uns in der Annahme, daß auch der hier vorgestellte globale Wegeplaner keine wesentlich längeren Wege erzeugt, was auch in den realen Experimenten Bestätigung fand.

Ein Problem des vorgestellten Verfahrens tritt auf, wenn eine zu besuchende Zwischenposition durch ein Hindernis blockiert ist, der lokale Planer also nicht direkt zu dem Zwischenziel fahren kann. Dieses Problem wird vermieden, indem um das eigentliche Zwischenziel ein relativ großer Kreis gelegt wird und es hinreichend ist, wenn der lokale Wegeplaner einen Weg zu einer Gitterzelle innerhalb des Kreises findet.

Ein vermeintlicher Nachteil des Planens auf dem Wegegraph scheint die Tatsache zu sein, daß nur die im Graph vorhandenen Positionen angefahren werden können. Zum einen kann aber für eine beliebige Zielposition der im Graph am nächsten liegende Knoten angefahren und von dort aus mit dem lokalen Wegeplaner das Ziel erreicht werden, zum anderen kann der Sichtbarkeitsgraph um einen Knoten an der Zielposition erweitert werden, indem alle Scans im näheren Umfeld auf diese Position projiziert werden und der Graph um die zugehörigen Sichtbarkeiten erweitert wird.

6.3.5 Verwandte Arbeiten

In [Edlinger und Weiß, 1995] wird ein ähnlicher Ansatz verfolgt. Das dort vorgestellte Robotersystem exploriert die Einsatzumgebung autonom durch Auswertung von 360°-Laserscans, die während der Fahrt aufgenommen werden. Zu jedem Scan werden genügend breite Passagen bestimmt, in welche weitere Zielpositionen für die Exploration gelegt werden. Zwischen den Scanpositionen wird ein Mindestabstand eingehalten, der eine gleichmäßige Verteilung der Aufnahmepositionen sichert. Liegt der Aufnahmepunkt eines Scans in einer genügend breiten Passage eines anderen Scans, so wird eine Verbindungskante zwischen den Scans angelegt. Auf diese Weise entsteht eine topologische Navigationskarte, auf der mittels A^*-Algorithmus Wege geplant werden.

Das Verfahren exploriert die Umgebung inkrementell, daher werden auch die Positionen der Scans inkrementell bestimmt, was zu Problemen führt, wenn die zu explorierende Umgebung einen Zyklus enthält. Das hier vorgestellte Verfahren dagegen korrigiert die Aufnahmepositionen aller Scans nach der Explorationsphase und kann daher mit Zyklen in der Umgebung umgehen.

Die Erstellung der topologischen Karte in [Edlinger und Weiß, 1995] basiert auf der direkten Auswertung der beteiligten Scans. Ist während der Aufnahme dieser Scans ein störendes Hindernis vorhanden, so wird möglicherweise keine genügend breite Passage zu einem benachbarten Scan gefunden und in der topologischen Karte kann zwischen den Scans keine Kante eingefügt werden. Weiterhin kann keine Kante eingetragen werden, wenn sich direkt zwischen den Scans ein kleines stationäres Hindernis befindet. Unser Verfahren benutzt die Sichtbarkeit, um Kanten zwischen zwei Scans zu legen. Dies hat den Vorteil, daß selbst wenn Hindernisse direkt zwischen den Scans liegen, die Sichtbarkeit immer noch groß ist. Die eigentliche Wegeplanung zwischen den Scans geschieht erst zur Laufzeit durch Einsatz des lokalen Wegeplaners, welcher Hindernissen dynamisch ausweicht. Ein Nachteil unseres Verfahrens ist, daß eventuell Kanten im Wegegraph vorhanden sind, die nie befahrbar sind, z.B. Kanten zwischen zwei Positionen, die durch ein Gitter abgetrennt sind, durch das der Laserscanner stellenweise

„hindurchsieht“.

Der in [Edlinger und Weiß, 1995] verwendete Scanüberdecker, welcher auf Korrelation von Histogrammen beruht, kann nur in polygonalen Umgebungen eingesetzt werden. Der von uns verwendete kombinierte Scanüberdecker ist dagegen in der Lage, in nicht-polygonalen Umgebungen aufgenommene Scans zu überdecken und liefert zusätzlich ein Gütemaß für die Zuverlässigkeit der berechneten Überdeckung.

Ein weiterer Vorteil unseres Verfahrens ist die Möglichkeit des Neuplanens. Ist eine Kante nicht befahrbar, so wird dies vom lokalen Wegeplaner erkannt und eine globale Neuplanung im Wegegraph eingeleitet.

In [Thrun und Bücken, 1996; Thrun *et al.*, 1998a] wird ein weiteres Verfahren zur Erstellung topologischer Karten beschrieben. Dort wird ein globales Belegtheitsgitter über die gesamte Umgebung gelegt und durch Auswerten des zugehörigen Voronoi-Diagrammes die topologische Karte erstellt. Auch hier muß die Umgebung bei der Kartenerstellung statisch sein. Kleine Hindernisse bei der Aufnahme würden sonst zu Veränderungen in der topologischen Karte führen. Der Ansatz hat aber den Vorteil, daß die gesamte Umgebung in Bereiche unterteilt wird und eine eher räumliche Beschreibung entsteht.

In [Röfer, 1995] wird in einer Simulation die Erstellung einer topologischen Karte mit eindimensionalen 360°-Farbbildern vorgestellt. Farbbilder sind jedoch stark von der Beleuchtung abhängig und kleine Änderungen in der Umgebung können starke Änderungen in der Korrelation zweier Bilder bewirken. Daher scheint dieser Ansatz nicht praktikabel zu sein.

Ein weiterer Ansatz wird in [Zimmer, 1995] vorgestellt. Hier wird ein Roboter mit taktilen und lichtempfindlichen Sensoren eingesetzt und mittels einem erweiterten Kohonenmodell eine topologische Karte erstellt. Die entstehende Karte enthält jedoch relativ viele Knoten und scheint daher für größere Umgebungen ungeeignet zu sein.

6.3.6 Bewertung

Es wurde ein neues Verfahren zur Erstellung einer topologischen Karte aus Laserscans vorgestellt. Das Verfahren basiert auf der Auswertung der Sichtbarkeit zwischen Scans, um Wege mit maximaler Wahrscheinlichkeit für die Befahrbarkeit zu bestimmen. Zusammen mit einem lokalen gitterbasierten Wegeplaner wird ein hoher Grad an Robustheit erlangt. Unvorhergesehenen Hindernissen wird dynamisch ausgewichen, nicht passierbare Wege werden erkannt und alternative Wege gefunden. Das Verfahren wurde auf einem realen Roboter implementiert und erfolgreich getestet.

Nicht unerwähnt sollen folgende Beschränkungen und Probleme dieses Verfahrens sein.

- Falls eine anzufahrende Position vollständig durch ein Hindernis blockiert ist, so bricht der lokale Planer ab, obwohl es vielleicht einen Pfad um das Hindernis herum geben könnte. Das System führt dann entweder mit einer globalen Neuplanung fort oder bricht die Pfadplanung ergebnislos ab.

- Für die Pfadplanung wird nur lokale Information verwendet und eine geschlossen Türe, die schon von weitem als nichtpassierbar erkannt werden könnte, wird erst dann erkannt, wenn der Roboter sie passieren will.

- Die topologische Karte hängt stark von den der Wahl der Scanpositionen ab und es ist unklar, wie diese automatisch gewählt werden können. Beispielsweise könnte eine zufällige Wahl zu einem Zick-Zack-Kurs in einem Korridor führen.

- Der Graph enthält möglicherweise ähnliche Kanten, z.B. zwei Kanten, die beide durch die gleiche Türe führen. Sobald einer dieser Kanten nichtpassierbar ist, sind alle diese Kanten nicht passierbar. Es ist schwierig diesen Sachverhalt im Graphen zu repräsentieren. In der gegenwärtigen Implementierung wird versucht, jede der Kanten entlangzufahren.

Manche der soeben beschriebenen Probleme treten bei allen Verfahren zur Erstellung topologischer Navigationskarten auf, die Knoten als einzelne Punkte repräsentieren. Sinnvoller ist es daher, ganze Regionen, z.B. Räume, als Knoten und Verbindungen, z.B. Türen, als Kanten zu verwenden, ähnlich wie es im Verfahren von Thrun und Bücken [1996] vorgeschlagen wurde. Das dort beschriebene Verfahren ist aber sensitiv gegenüber Ausreißern in den Sensordaten, z.B. verursacht ein Hindernis in einem Raum die Erzeugung mehrere topologischer Regionen. Außerdem ist die Einteilung in Regionen nicht intuitiv.

Eine andere Idee ist in Abbildung 6.9 skizziert. Hier wurden alle Hindernisse in ein Belegtheitsgitter eingetragen und soweit vergrößert, daß alle Türbereiche vollständig geschlossen werden. Dadurch entstehen mehrere Regionen, welche in diesem Beispiel genau Räumen und Gängen entsprechen. Durch Vergrößern dieser Regionen ist es möglich, auch die Verbindungen zwischen Regionen zu bestimmen. Der Vorteil dieses Verfahrens ist, daß eine „natürliche“ Aufteilung in Räume und Gänge gefunden wird. Nachteilig ist, daß relativ große Regionen entstehen können, die für eine gitterbasierte Wegeplanung weiter aufgeteilt werden müssen. Außerdem hängt das Verfahren stark von dem verwendeten Wert zur Vergrößerung der Hindernisse ab. Insgesamt bedarf diese Methode weiterer Untersuchungen bevor sie in realen Umgebungen eingesetzt werden kann.

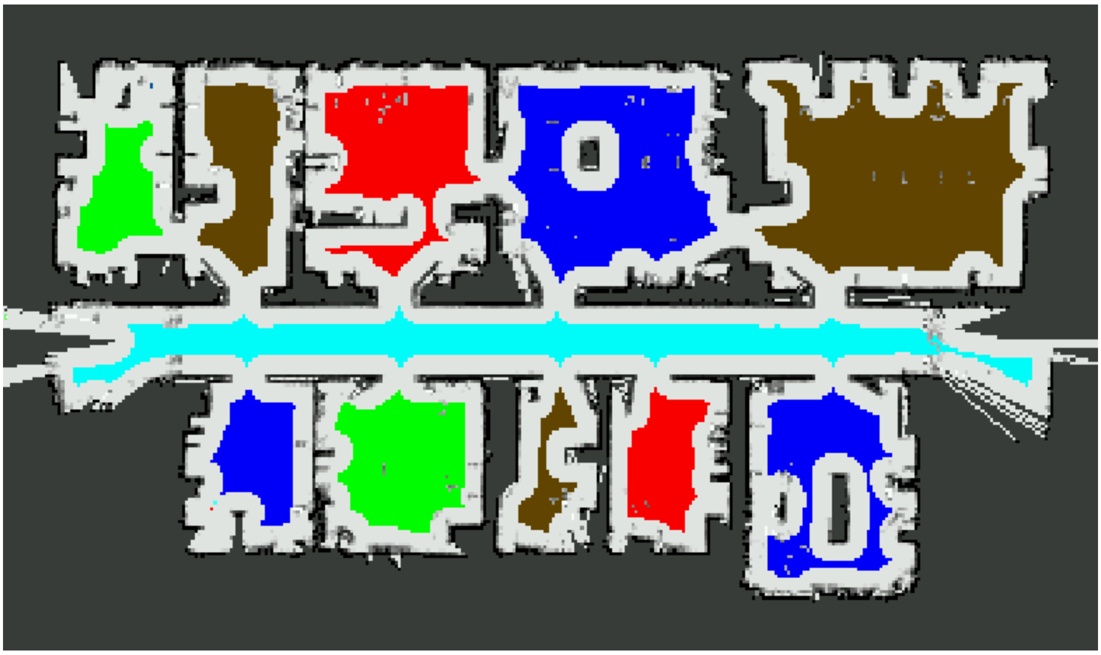


Abbildung 6.9: Aufteilung in topologische Regionen aus Sensordaten der KI-Abteilung der Universität Freiburg.

Kapitel 7

RoboCup

In dieser Arbeit wurden bisher einzelne Komponenten wie Selbstlokalisierung, Kartenerstellung und Wegeplanung für die robuste Navigation eines autonomen, mobilen Roboters präsentiert. In diesem Kapitel sollen diese Komponenten in der Beispieldomäne Roboterfußball angewendet und weitere Komponenten, die für ein erfolgreiches Zusammenspiel von Roboter-Fußballspielern notwendig sind, vorgestellt werden.

Roboterfußball ist ein herausfordernder Forschungsbereich, da für eine erfolgreiche Mannschaft Probleme in den Bereichen Robotik, Künstliche Intelligenz, Multi-Agentensysteme und Schlußfolgern unter Echtzeitbedingungen gelöst werden müssen [Kitano *et al.*, 1997]. Dieses Kapitel beschreibt die Schlüsselkomponenten des CS-Freiburg-Teams, wobei der Schwerpunkt auf den Modulen zur Selbstlokalisierung und Objekterkennung auf der Basis von Laserscannerdaten liegt. Aus diesen Informationen wird ein explizites Weltmodell aufgebaut, das für Wegeplanung, verhaltensbasierte Steuerung und Kooperation eingesetzt wird.

Zunächst werden verschiedene Ligen, in denen Roboterfußball ausgetragen wird, beschrieben. Danach wird der Ansatz des CS-Freiburg-Teams vorgestellt und begründet, warum dieser Ansatz verfolgt wurde. Anschließend werden die verwendete Hardware, sowie Team- und Spielerarchitektur präsentiert.

Auf die speziell für RoboCup zugeschnittene Selbstlokalisierung wird in einem weiteren Abschnitt eingegangen. Das Verfahren benutzt ein linienbasiertes Scan-Matching-Verfahren, welches schnell, robust und hochgenau ist. Das Verfahren wird mit anderen Scan-Matching experimentell verglichen.

Im Anschluß werden dann Weltmodellierung und Pfadplanung besprochen. Darauf aufbauend werden verhaltensbasierte Steuerung, sowie Kooperation der Spieler untereinander vorgestellt. Zuletzt wird über Erfahrungen bei der Teilname an Turnieren und über Spielergebnisse berichtet.

Die in diesem Kapitel vorgestellten Arbeiten sind größtenteils in Gutmann *et al.* [1998b; 1999a; 1999b; 2000] veröffentlicht. Ein Gesamtüberblick des CS-Freiburg-Teams mit weiteren Einzelheiten, die im Rahmen dieser Arbeit nur grob beschrieben werden, findet sich in der Arbeit von Weigel [1999].

7.1 RoboCup-Ligen

Ein junges, aber schnell an Popularität gewinnendes Standardproblem in Bereich der autonomen mobilen Robotik ist der Roboterfußball. Die *Robot World Cup Initiative (RoboCup)* wurde von einer Gruppe japanischer Forscher ins Leben gerufen mit dem Ziel, die Forschung in Künstliche Intelligenz und Robotik zu pflegen, indem ein Standardproblem benutzt wird, bei dem eine breite Menge von Technologien integriert und untersucht werden kann [Kitano *et al.*, 1998]. Dabei wird folgende Unterteilung in RoboCup-Ligen unternommen.

Simulationsliga: In der Simulationsliga spielen virtuelle Fußballspieler auf einem simulierten Fußballfeld. Die Simulation versucht, realistische Verhältnisse wie sie in einem realen Fußballspiel auftreten zu modellieren. Beispielsweise hat ein Spieler einen eingeschränkten Sichtbereich und kann nur eine kurze Distanz mit voller Geschwindigkeit laufen. Vorteil dieser Klasse ist, daß keine Hardware-Arbeiten anfallen und man sich ganz auf Grundfähigkeiten und Strategie konzentrieren kann.

Klasse der kleinen Roboter: In dieser Klasse spielen bis zu 5 Roboter pro Team auf einem Feld der Größe einer Tischtennisplatte. Die Grundfläche jedes Roboters ist dabei auf $180cm^2$ beschränkt. Über dem Spielfeld ist eine Videokamera montiert, welche das gesamte Spielfeld erfaßt und auf die beide Teams Zugriff haben. Durch diese Kamera vereinfachen sich viele Probleme wie Selbstlokalisierung (die Roboter dürfen farbige Markierungen auf der Oberseite besitzen) und Ballpositionsbestimmung. Hierdurch wird Perzeption und Erstellung eines Weltmodells realisierbar und die Entwicklung von komplexeren Verhalten wie beispielsweise Passen von Bällen, Freilaufen von Spielern, Kooperation zwischen Spielern, etc. möglich [Veloso *et al.*, 1998b; Veloso und Stone, 1998].

Klasse der mittelgroßen Roboter: Diese Klasse erlaubt pro Team vier[1] Roboter, die eine Größe von maximal $50cm$ im Durchmesser besitzen dürfen. Auf die Roboter können nahezu beliebige Sensoren montiert werden und die Rechenkapazität kann auf dem Roboter selbst oder über Funk auf einem externen Rechner vorhanden sein. In dieser Klasse gibt es sehr viel Spielraum bei der Gestaltung der Hardware, z.B. können Schußapparaturen, Ballführungsmechanismen, oder verschiedene Kombinationen von Sensoren ausprobiert werden. Daher wird viel Zeit für das Design der Hardware verwendet und es bleibt meist nur wenig Spielraum, Wahrnehmung, Basisverhalten und Spielstrategien zu implementieren. Das CS-Freiburg-Team, welches in diesem Kapitel beschrieben wird, tritt in dieser Klasse an.

[1] 1998 waren bis zu fünf Roboter pro Team erlaubt

Klasse der Roboter mit Beinen: Eine weitere Klasse ist die der Roboter, welche nicht auf Rädern auf dem Spielfeld fahren, sondern sich auf Beinen bewegen. Die Roboter sind sehr klein und bisher gibt es nur einen Typ, welcher von der Firma *SONY* hergestellt wird. Die Sensorik beschränkt sich auf eine einfache Videokamera, was die Wahrnehmung sehr stark einschränkt. Obwohl für die Positionsbestimmung der Roboter farbige Landmarken am Spielfeldrand aufgestellt sind, die von der Kamera erkannt werden können, erweist sich die Lokalisierung als sehr schwierig [Veloso *et al.*, 1998c].

7.2 Ansatz des CS-Freiburg

Während die meisten Teams in der Klasse der mittelgroßen Roboter als hauptsächlichen Sensor ein Videosystem verwenden, um Ball, Spieler und Tore zu erkennen, verwendet das CS-Freiburg-Team hierfür Laserscanner. Aus den Daten des Scanners werden die eigene Position und die Positionen anderer Spieler extrahiert und zusammen mit der Ballposition, welche durch Daten einer Videokamera bestimmt wird, in einem Weltmodell verwaltet. Das Weltmodell dient als zentraler Baustein für die gesamte Steuerung des Roboters, d.h. Aktionen werden aufgrund verschiedener Situationen (z.B. Ball sichtbar oder nicht sichtbar) ausgewählt.

In einem Papier über die Herausforderungen des Roboterfußballs mutmaßen Asada *et al.* [1998], daß Entfernungssensoren nicht ausreichen, um Ball, Hindernisse und Tore zu unterscheiden [Asada *et al.*, 1998, Seite 48]. Weiterhin wurde in diesem Papier vermutet, daß „konventionelle“ Ansätze, die ein explizites Weltmodell erstellen und auf diesem Pläne berechnen und ausführen, nicht passend für die dynamische Umgebung des Roboterfußballs sind [Asada *et al.*, 1998, Seite 49].

Während Sonarsensoren für diese Aufgabe sicherlich nicht ausreichend genau und zuverlässig sind, sind Laserscanner sehr wohl in der Lage, alles auf dem Spielfeld mit Ausnahme des Balles zuverlässig zu erkennen. Außerdem kann aus den Laserdaten ein explizites Weltmodell konstruiert werden, das ausgeklügelte Verhalten und Deliberation unterstützen kann.

Weiterhin scheint die Erstellung eines expliziten Weltmodells und Einsatz von Deliberation eine notwendige Voraussetzung zu sein, um ein ästhetisches und effektives Fußballspiel zu bestreiten. Diese Vermutung wird durch die Tatsache unterstützt, daß die beiden Gewinner der Simulationsliga und der kleinen Liga in RoboCup'97 einen solchen Ansatz verfolgten. [Burkhard *et al.*, 1998; Veloso *et al.*, 1998a; Veloso *et al.*, 1998b].

Die Leistung dieser Teams steht im krassen Gegensatz zu den Teams der Liga der mittelgroßen Robotern bei RoboCup'97. Obwohl viele der dort aufgetretenen Probleme an schlechter Funkverbindung und ungünstigen Lichtverhältnissen lagen [Noda *et al.*, 1998], ist die schlechte Leistung auch auf das Nichtvorhandensein eines expliziten Weltmodells zurückzuführen. Ein weiteres Indiz für die

Notwendigkeit eines Weltmodells ist die Tatsache, daß das CS-Freiburg-Team die Meisterschaft 1998 in der mittleren Klasse gewann.

7.3 Architektur

Die im CS-Freiburg-Team eingesetzten Hardware-Komponenten sowie die allgemeine Architektur sind sehr ähnlich zu denen anderer Teams. Der hauptsächliche Unterschied liegt in der Verwendung von Laserscannern, Ballsteuerungsmechanismen und einer globalen Sensorintegration, die ein globales Weltmodell bereitstellt.

7.3.1 Hardware-Komponenten

Da unsere Arbeitsgruppe nicht spezialisiert ist im Entwurf von Roboterplattformen, wurden handelsübliche Roboter vom Typ *Pioneer-I* benutzt. Die Basisversion des *Pioneer-I* ist jedoch kaum in der Lage guten Fußball zu spielen, da Sensoren und Effektoren hierfür nicht ausreichend sind. Aus diesem Grund wurden eine Reihe von zusätzlichen Komponenten auf den Robotern angebracht (siehe Abbildung 7.1).

Auf jeden Roboter wurde eine Videokamera montiert, die an das *Cognachrome*-Bildverarbeitungssystem von *Newton Labs* angeschlossen und für Ballerkennung und -verfolgung eingesetzt wurde. Um alle lokalen Informationen zu verarbeiten, wurde jeder Roboter mit einem kleinen *Toshiba* Notebook des Typs *Libretto 70CT*, auf dem das *Linux*-Betriebssystem installiert wurde, ausgerüstet. Der Roboter selbst wird über das *Saphira*-System [Konolige *et al.*, 1997], das mit jedem *Pioneer*-Roboter ausgeliefert wird, gesteuert. Um schließlich Kommunikation mit einem Rechner außerhalb des Spielfeldes zu ermöglichen, wurde das drahtloses Funk-Ethernet *WaveLan* der Firma *Lucent Technologies* verwendet.

7.3.2 Laserscanner

Zusätzlich zu den oben aufgeführten Komponenten wurde auf jeden Roboter ein *PLS200*-Laserscanner der Firma *SICK AG* montiert. Diese Scanner liefern Entfernungsinformation für ein Sichtfeld von 180° mit einer Winkelauflösung von 0.5° und einer Genauigkeit von $5cm$ bis zu einer Distanz von $30m$.

Die von diesem Laserscanner gelieferte Information ist sehr zuverlässig und nur wenig verrauscht. Nach den Spezifikationen der *SICK AG* sollten in einer Umgebung nicht mehrere Laserscanner gleichzeitig operieren oder es sollte sichergestellt sein, daß diese auf verschiedenen Höhen montiert sind. Für das CS-Freiburg-Team wurden die Scanner jedoch alle auf der gleichen Höhe betrieben und es wurden keine Inferenzprobleme beobachtet.

Abbildung 7.1: Drei der fünf Roboter-Fußballspieler des CS-Freiburg.

7.3.3 Ballsteuerungsmechanismus

In der Basisversion des *Pioneer-I*-Roboters gibt es keine sehr effektiven Möglichkeiten, um mit einem Ball umzugehen, ihn auf dem Feld zielgerichtet zu bewegen oder ins gegnerische Tor zu schießen. Aus diesem Grunde wurde ein Schußmechanismus aus Teilen des *Märklin Metallbaukasten* entwickelt [Herrmann, 1998]. Der Kicker selbst wird von zwei Elektrospulen ausgelöst und schießt den Ball über eine Distanz von etwa drei Metern. Weiterhin wurden flexible Federn an der Vorderseite angebracht, um den Ball führen zu können. Die Federn haben eine Länge von ca. 30 % des Balldurchmessers (siehe Abbildung 7.2).

Dieser Ballführungsmechanismus führte zu Diskussionen vor der RoboCup'98-Meisterschaft. Es wurde jedoch am Ende beschlossen, daß der Einsatz dieser Konstruktion nicht gegen die RoboCup-Regeln verstößt, da nicht alle Freiheitsgrade des Balles eingeschränkt werden und der Gegner immer noch Gelegenheit hat, den Ball zu bekommen.

Das CS-Freiburg-Team glaubt, daß solch eine Ballführungseinheit definitiv

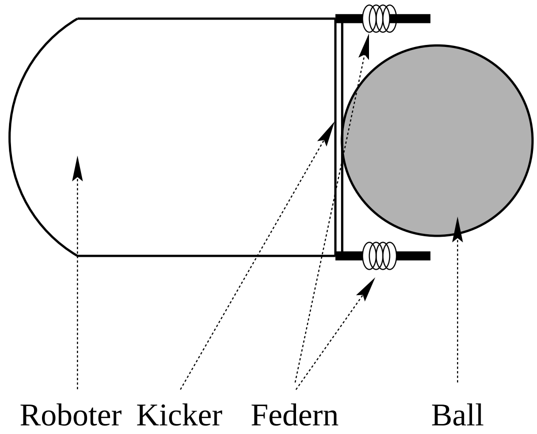


Abbildung 7.2: Mechanismus zur Ballsteuerung.

erlaubt sein sollte, da ohne sie es praktisch unmöglich ist, den Ball von der Bande weg wieder ins Spielfeld zu bringen, was bedeuten würde, daß der Schiedsrichter immer wieder den Ball neu positionieren müßte, was wiederum sehr störend ist, besonders für die Zuschauer. Außerdem verliert der Roboter ohne Ballführung den Ball sehr schnell, wenn er ihn bewegen will. Insbesondere wäre unser letztes Tor im letzten Spiel bei RoboCup'98 nicht möglich gewesen ohne diese Federn.

7.3.4 Allgemeine Architektur

Wie bereits erwähnt, ist die allgemeine Architektur des CS-Freiburg-Teams (siehe Abbildung 7.3) sehr ähnlich zu denen anderer Teams in der Klasse der mittelgroßen Roboter. Es gibt jedoch auch ein paar bemerkenswerte Unterschiede.

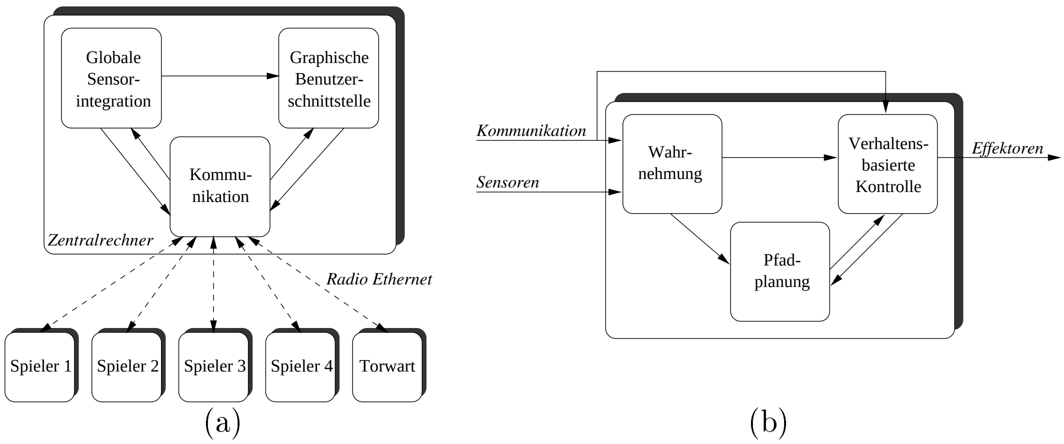


Abbildung 7.3: (a) Teamarchitektur (b) Spielerarchitektur

Die Roboter sind im Wesentlichen autonome Roboter-Fußballspieler. Jeder Spieler ist mit Sensoren, Effektoren und einem Computer ausgestattet und besitzt ein *Wahrnehmungsmodul*, welches ein lokales Weltmodell erstellt und verwaltet (siehe Abbildung 7.3b). Aufgrund des beobachteten Zustandes im Weltmodell und den durch Funk übertragenen Intentionen der Mitspieler entscheidet das *verhaltensbasierte Steuerungsmodul*, welches Verhalten aktiviert wird. Sofern das ausgewählte Verhalten den Roboter an einen bestimmten Punkt im Spielfeld bewegen möchte, so wird das *Pfadplanungsmodul* benutzt, das einen kollisionsfreien Weg zum Zielpunkt berechnet.

Um die Fußballspieler zu initialisieren, zu starten und zu stoppen, und um den Zustand aller Agenten zu überwachen, wird eine Radio-Ethernet-Verbindung zwischen den On-Board-Computern und einem Rechner außerhalb des Spielfeldes eingesetzt (siehe Abbildung 7.3a). Falls die Funkverbindung unbrauchbar ist, so kann das Fußballteam immer noch eingesetzt werden, indem jeder Roboter manuell gestartet wird. Das *AGILO*-Team [Klupsch *et al.*, 1999], das *ART*-Team [Nardi *et al.*, 1999], das *T-Team* [Zell *et al.*, 1999], das *Uttori*-Team [Yokota *et al.*, 1998] und andere verwenden einen sehr ähnlichen Ansatz.

Im Gegensatz zu manch anderem Team jedoch, verwendet das CS-Freiburg-Team den Computer außerhalb des Spielfeldes und die Funkverbindung, um eine *globale Sensorintegration* zu verwirklichen, welche ein *globales Weltmodell* erstellt und verwaltet. Dieses Weltmodell wird regelmäßig an alle Spieler versandt und diese können dadurch ihre lokale Sicht der Welt erweitern. Dies bedeutet, daß das Weltmodell der einzelnen Spieler sehr ähnlich dem Weltmodell ist, das durch eine alles überblickende Kamera entsteht, wie sie in der Klasse der kleinen Roboter z.B. vom Team *CMUnited* [Veloso *et al.*, 1998b] verwendet wird. Die Informationen des globalen Weltmodells im CS-Freiburg-Team sind jedoch weniger genau als die Informationen, die durch direkte Beobachtung gewonnen werden (siehe Abschnitt 7.5).

7.4 Selbstlokalisierung

Die Entwicklung des CS-Freiburg-Teams begann unter der Hypothese, daß es von großem Vorteil ist, wenn jeder der Spieler genau weiß, wo er sich auf dem Spielfeld befindet. Aufgrund der in dieser Arbeit erzielten Erfahrungen mit Methoden zur Selbstlokalisierung von Robotern (siehe Kapitel 4) ist es naheliegend eine dieser Methoden für den Roboterfußball zu verwenden.

Für den Roboterfußball muß eine Methode robust, genau und effizient sein. Robust bedeutet, daß auch größere Positionsfehler noch korrigiert werden können und daß das Verfahren in der Lage ist, den Roboter global zu lokalisieren, sofern die ungefähre Orientierung des Spielers bekannt ist. Genau heißt, daß die Position auf wenige Zentimeter und auf wenige Grad genau sein soll, um plazierte Pässe und Torschüsse durchführen zu können. Schließlich bedeutet effizient, daß das

Verfahren nur wenige Millisekunden Rechenzeit benötigen darf. Leider ist keines der in Kapitel 4 beschriebenen Verfahren in der Lage alle drei Anforderungen zu erfüllen. Aus diesem Grunde wurde ein neues Scan-Matching-Verfahren *LineMatch* entworfen, das die simple, polygonale Struktur der RoboCup-Umgebung ausnutzt und Allgemeingültigkeit gegen Geschwindigkeit und der Möglichkeit der globalen Lokalisierung abwägt.

7.4.1 LineMatch

Der *LineMatch*-Algorithmus extrahiert Liniensegmente aus einem Scan und überdeckt diese mit einem apriori Linienmodell der RoboCup-Umgebung ähnlich wie die Methoden von Shaffer *et al.* [1992] und Castellanos *et al.* [1996b]. Es wird erwartet, daß dieser Algorithmus schneller und robuster als andere Scan-Matching-Verfahren ist, aber dennoch so genau wie diese ist. In wie weit diese Vermutungen zutreffen wird in Abschnitt 7.4.2 untersucht.

Für die Extraktion von Linien aus den Scandaten wird der in Abschnitt 3.2.1 beschriebene Algorithmus verwendet. Um sicherzustellen, daß die extrahierten Linien auch wirklich den Spielfeldbanden entsprechen, werden nur solche Linien in Betracht gezogen, die signifikant länger als die erlaubte maximale Ausdehnung eines Roboters sind. Folgender Algorithmus zeigt, wie die Überdeckung von Scanlinien mit einem apriori Linienmodell durch rekursives Ausprobieren aller Paarungen von Scanlinien und Modellinien durchgeführt wird:

Algorithmus 7.1: *LineMatch*(M, S, P)
Eingabe: Modellinien M, Scanlinien S, Paarungen P
Ausgabe: Satz von Positionshypothesen H
Ablauf:

```
if |P| = |S| then
    H := {P}
else
    H := ∅
    s := SelectScanline(S, P)
    for all m ∈ M do
        if VerifyMatch(M, S, P ∪ {(m, s)}) then
            H := H ∪ LineMatch(M, S, P ∪ {(m, s)})
        endif
    endfor
endif
return H
```

SelectScanline wählt die nächste Scanlinie aus, die überdeckt werden soll und *VerifyMatch* überprüft durch Berechnung der Rotation und Verschiebung, ob die neue Paarung (m, s) kompatibel ist mit den bereits akzeptierten Paarungen in P.

Der Algorithmus liefert eine Menge von Positionshypothesen in Form von Paarungen, die einfach in mögliche Scanaufnahmepositionen transformiert werden können.

In der RoboCup-Umgebung ist dieser Algorithmus in der Lage, die globale Position des Roboters zu bestimmen modulo der Symmetrie des Spielfeldes. Dies bedeutet, daß man zwei Positionshypothesen erhält, wenn drei Spielfeldbanden sichtbar sind (siehe Abbildung 7.4), vier Hypothesen, wenn zwei Banden sichtbar sind, und vier Hypothesen mit einer entarteten Kovarianzmatrix, wenn nur eine Wand sichtbar ist.

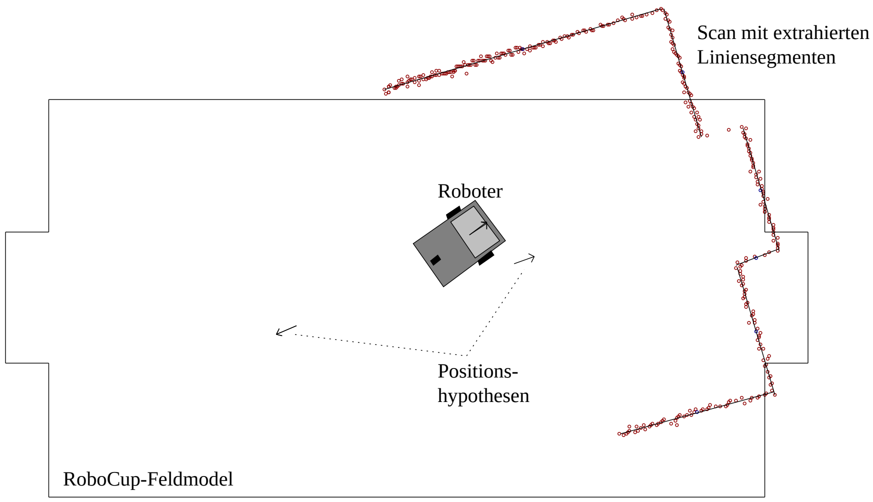


Abbildung 7.4: Der LineMatch-Algorithmus berechnet zwei Hypothesen für die Roboterposition.

Diese Scan-Matching-Methode ist ähnlich zu den Methoden von Castellanos *et al.* [1996b] und Shaffer *et al.* [1992]. Im Gegensatz zu deren Methoden jedoch werden im LineMatch-Algorithmus nur die *globalen Bedingungen* bezüglich Verschiebung und Verdrehung sowie die Längenrestriktion von Scanlinien überprüft. Dies ist ausreichend für die Bestimmung der Positionshypothesen und effizienter als die Methode von Castellanos *et al.* [1996b]. Weiterhin benötigt das LineMatch-Verfahren keine initiale Positionsschätzung, so daß sogar extreme Fehler korrigiert werden können.

Nachdem ein Scan überdeckt wurde, wird die plausibelste Positionshypothese mit Informationen der Odometrie durch Einsatz eines Kalman-Filters fusioniert. Hierzu werden direkt die Gleichungen 4.20 und 4.21 angewandt. Die Bestimmung der plausibelsten Positionshypothese wird aufgrund der aktuellen Positionsinformation der Odometrie durchgeführt. Die Positionshypothese, die am nächsten der aktuellen Odometrieposition liegt und ungefähr die gleiche Orientierung besitzt, wird als plausibelste Hypothese angesehen.

Um das Selbstlokalisierungssystem zu initialisieren, wird der Roboter an eine beliebige Position im Spielfeld aber grob in Richtung des gegnerischen Tores gestellt und der Mittelwert und die Fehlerkovarianzmatrix der Roboterposition auf die folgenden Werte gesetzt:

$$\mu_l := (0,0,0)^T \tag{7.1}$$

$$\Sigma_l := \begin{pmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{pmatrix} \tag{7.2}$$

Dies führt zu einer globalen Selbstlokalisierung bei Überdeckung des ersten Scans.

Da keine Ausreißer beim Überdecken der Scanlinien mit Modellinien erlaubt sind, kann der Algorithmus fehlschlagen, falls inkonsistente Scans, z.B. Scans, die eine lange diagonale Wand enthalten, aufgenommen werden. In wie weit diese Situation die Positionsschätzung des Roboters beeinflußt wird in Abschnitt 7.4.2 untersucht.

Wie sich herausstellt, ist der implementierte Algorithmus sehr schnell in der RoboCup-Umgebung (siehe Abschnitt 7.4.2). Da hier die Menge M der Modellinien nur sehr klein ist (12 Linien reichen aus, um alle Spielfeldbanden zu modellieren) ist es interessant, das Laufzeitverhalten in Abhängigkeit der Größe der Menge M zu bestimmen. Eine erste grobe Analyse ergibt, daß die Laufzeit des Algorithmus im schlechtesten Fall $O(|M|^{|S|})$ ist. Dies ergibt sich aus der Rekursionstiefe $|S|$ und der Tatsache, daß in jeder Rekursion von *LineMatch* $|M|$ verschiedene Paarungen ausprobiert werden.

Es stellt sich jedoch heraus, daß eine sehr viel besser Abschätzung der Laufzeit im schlechtesten Fall möglich ist. Nach der zweiten Rekursionsstufe, also nachdem zwei Paarungen festgelegt wurden, sind alle Freiheitsgrade für Rotation und Translation fest (*SelectScanline* ist so implementiert, daß in den ersten beiden Rekursionsstufen zwei nicht-parallele Linien zurückgegeben werden). Dies bedeutet, daß in tieferen Rekursionsstufen nur noch eine Paarung konsistent sein kann, welche dann den nächsten rekursiven Aufruf von *LineMatch* verursacht. Durch diese Betrachtung erhält man $|M|^2$ mögliche Paarungen in den ersten beiden Rekursionsstufen, die durch weitere rekursive Aufrufe gefolgt werden, die $|M||S|$ verschiedene Paarungen überprüfen. Schließlich, da *VerifyMatch* $O(|S|)$ viel Zeit benötigt, erhält man eine Gesamtkomplexität von $O(|M|^3|S|^2)$.

Im allgemeinen muß man mit der kubischen Laufzeit auskommen. Dennoch kann für realistische Umgebungen, in denen nicht alle Wände gleichzeitig sichtbar sind wie z.B. in gewöhnlichen Büroumgebungen, durch einen Vorverarbeitungsschritt die Laufzeit auf beinahe lineare Komplexität in $|M|$ verkleinert werden. Solch eine Vorverarbeitungsphase würde für jede Wand alle anderen Wände bestimmen, die gleichzeitig sichtbar sein können. Durch Verwenden dieser Datenstruktur kann die Zahl der zu untersuchenden Wände drastisch reduziert werden und bei Annahme einer konstanten Anzahl gleichzeitig sichtbarer Wände würde sich eine lineare Komplexität des Verfahrens ergeben.

7.4.2 Vergleich mit anderen Scan-Matching-Verfahren

Um den Vorteil des LineMatch-Algorithmus gegenüber anderen Scan-Matching-Verfahren zu zeigen, wurden der Cox-Algorithmus, wie er in Abschnitt 4.4.1 für das Überdecken eines Scans mit einem apriori Modell von Liniensegmenten eingesetzt wurde, das kombinierte Scan-Matching-Verfahren CSM, das in Abschnitt 4.4.2 für die Überdeckung eines Scans mit einem Referenzscan verwendet wurde, und der LineMatch-Algorithmus miteinander verglichen. Nicht im Vergleich eingeschlossen sind das IDC-Verfahren und die Kreuzkorrelationsmethode, da deren Eigenschaften bereits durch den CSM-Algorithmus abgedeckt sind [Gutmann und Schlegel, 1996].

Da der CSM-Algorithmus einen Satz von Referenzscans als apriori Karte benötigt, wurde ein kleiner Satz von Scans aufgenommen, der akkumulierte Odometriefehler der Aufnahmepositionen mittels der Methode der konsistenten Positionsschätzung aus Abschnitt 5.3 korrigiert, und die so berechneten Scans als Referenzscans verwendet.

Für den Vergleich der verschiedenen Methoden wurden reale Daten mit einem der mobilen Fußballroboter aufgenommen. Um Daten eines realistischen Spielszenarios zu erhalten, wurde der Fußballroboter in der RoboCup-Umgebung mit mehreren stationären und sich bewegenden Hindernissen betrieben. Aufgrund dieser Daten wurde die durchschnittliche Laufzeit der verschiedenen Lokalisierungsmethoden bestimmt und es wurde wie in den Experimenten in Abschnitt 4.6 verschiedene Arten von Rauschen hinzugefügt, um Genauigkeit und Robustheit der Verfahren zu bestimmen.

Ähnliche Arbeiten wurden von Shaffer *et al.* [1992] durchgeführt, welche zwei Scan-Matching-Methoden miteinander verglichen, die ähnlich der Cox- und der LineMatch-Methode sind. Jedoch wurden in deren Arbeit nur einzelne Scanüberdeckungen verwendet, während in dieser Arbeit alle Daten eines gesamten Roboterlaufs berücksichtigt werden. Zudem verglichen Shaffer *et al.* ihre Algorithmen in einer nahezu statischen Umgebung, während in den Experimenten in dieser Arbeit, Daten eines realistischen, dynamischen Szenarios mit vielen stationären und sich bewegenden Hindernissen, welche die Sicht der Robotersensoren blocken können, in Betracht gezogen werden. Daher sollten die Resultate in dieser Arbeit ein besseres Bild davon geben, wie gut die einzelnen Verfahren in einer dynamischen Umgebung wie RoboCup tatsächlich sind.

Laufzeitergebnisse

Für die Bestimmung der Laufzeiten der verschiedenen Scan-Matching-Verfahren wurde die durchschnittliche Laufzeit jeder Methode für die Berechnung der Positionsaktualisierung vor Fusion mit der Odometrieschätzung gemessen. Um Messungen zu erhalten, welche die Performanz unter realen Spielbedingungen zeigen, wurde ein realistisches Spielszenario in einem RoboCup-Spielfeld aufgebaut, in welches mehrere stationäre und sich bewegende Objekte plaziert wurden (siehe Abbildung 7.5). Hier wurde einer der Fußballroboter als rechter Verteidiger eingesetzt, wobei der Spieler mehrere Male über das gesamte Spielfeld fuhr. Insgesamt legte der Roboter einen Distanz von ungefähr 41 Metern zurück, drehte sich um 11000 Grad (ca. 30 Umdrehungen) und sammelte über 3200 Scans.

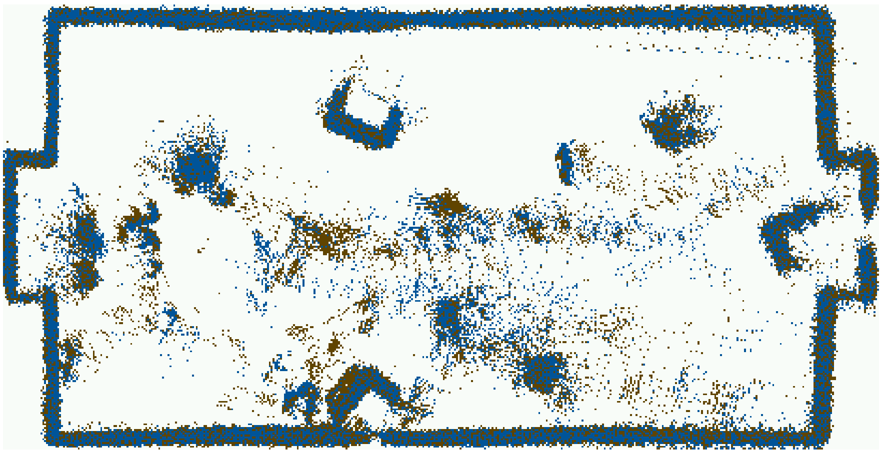


Abbildung 7.5: Experimentierumgebung: mehrere Hindernisse wurden in das RoboCup-Feld gestellt, um ein realistisches Szenario zu erzeugen. Verrauschte Messungen stammen von bewegten Hindernissen.

Abbildung 7.6 zeigt Laufzeitresultate, die auf dem On-Board-Rechner des Roboters, ein Pentium 120 MHz Laptop, gemessen wurden. Wie erwartet ist der LineMatch-Algorithmus schneller als die konkurrierenden Methoden. LineMatch ist 8 mal schneller als der Cox-Algorithmus und ca. 20 mal schneller als die CSM-Methode. Die sehr niedrige durchschnittliche Laufzeit von nur $2ms$ pro Scan erlaubt die Verarbeitung aller aufgenommenen Scans in Echtzeit.

Cox	CSM	LineMatch
$16ms$	$39ms$	$2ms$

Abbildung 7.6: Laufzeitergebnisse auf einem Pentium 120 MHz Laptop.

Ergebnisse in einem typischen Spielszenario

Um die Genauigkeit und Robustheit des LineMatch-Algorithmus zu bestimmen wurden die Daten des obigen Roboterlaufs aufgezeichnet und wie bei den Untersuchungen in Abschnitt 4.6 verschiedene Arten von Rauschen der Odometrieinformation hinzugefügt. Da für die Bestimmung der Genauigkeit der Positionsschätzungen Referenzpositionen nötig sind, müssen diese auf geeignete Weise ermittelt werden. Um dies zu vereinfachen, wurde einer der Scan-Matching-Methoden, der Cox-Algorithmus, auf die aufgenommenen Daten angewandt und die resultierenden Ausgaben als Referenzpositionen benutzt.

Für jeden Satz von Rauschwerten wurden 26 Läufe mit verschiedenen Startwerten für die Initialisierung eines Zufallszahlengenerators durchgeführt. Abbildung 7.7 zeigt die tatsächliche Trajektorie des Roboters und eine typische Trajektorie, wie sie durch starkes Verrauschen mit den Parametern $\langle 400, 100, 40\rangle$ ensteht. Die Werte entsprechen der Standardabweichung des Gauß'schen Rauschens $\langle \Delta_\delta(\delta), \Delta_\alpha(\alpha), \Delta_\alpha(\delta)\rangle$, wobei Symbole und Einheiten wie in Abschnitt 4.6 verwendet wurden.

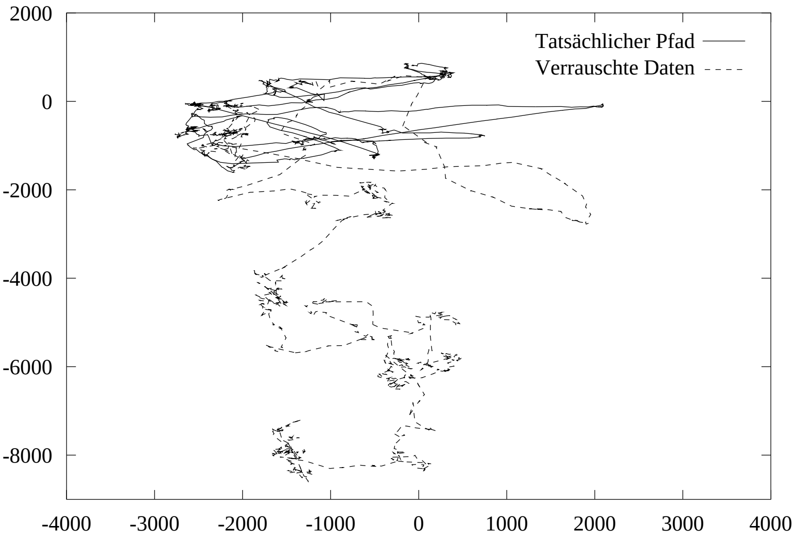


Abbildung 7.7: Tatsächliche Robotertrajektorie und typische Trajektorie, die durch Addition von starkem Gauß'schem Rauschen mit Standardabweichungen $\langle 400, 100, 40\rangle$ entsteht.

Für jede Scan-Matching-Methode wurden die Anzahl der Fälle, in denen der Roboter seine Position verloren hatte, gezählt und der Abstands- und Rotationsfehler zur Referenzposition bestimmt. Für die Entscheidung, wann eine Position als verloren angesehen wird, wurde eine Schranke von $0.5m$ für den Abstand und $30°$ für die Orientierung benutzt. Für die Bestimmung von Abstands- und Rotationsfehler wurden nur die Positionen betrachtet, bei denen der Roboter nicht verloren war.

Abbildung 7.8 zeigt den durchschnittlichen Abstandsfehler und Abbildung 7.9 den durchschnittlichen Rotationsfehler zu den Referenzpositionen für fünf verschiedene Stufen von Gauß'schem Odometrierauschen. Die Wertetripel auf der x-Achse entsprechen der Standardabweichung des Gauß'schen Rauschens $\langle \Delta_\delta(\delta), \Delta_\alpha(\alpha), \Delta_\alpha(\delta)\rangle$. In diesen und allen nachfolgenden Abbildungen entsprechen die Fehlerbalken dem 95% Konfidenzinterval des Durchschnittwertes.

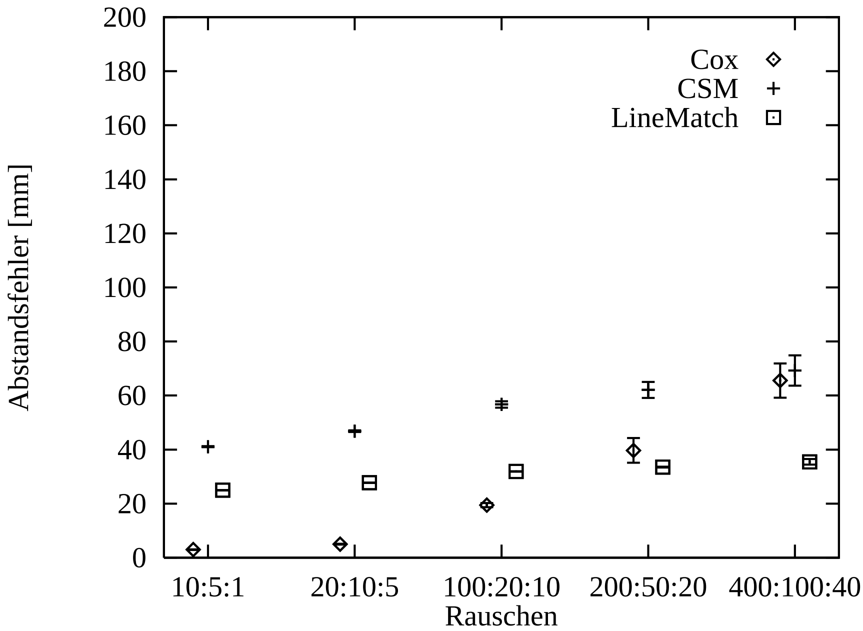


Abbildung 7.8: Abstandsfehler zu Referenzpositionen in typischen Spielszenario für verschiedene Stufen von Gauß'schem Odometrierauschen.

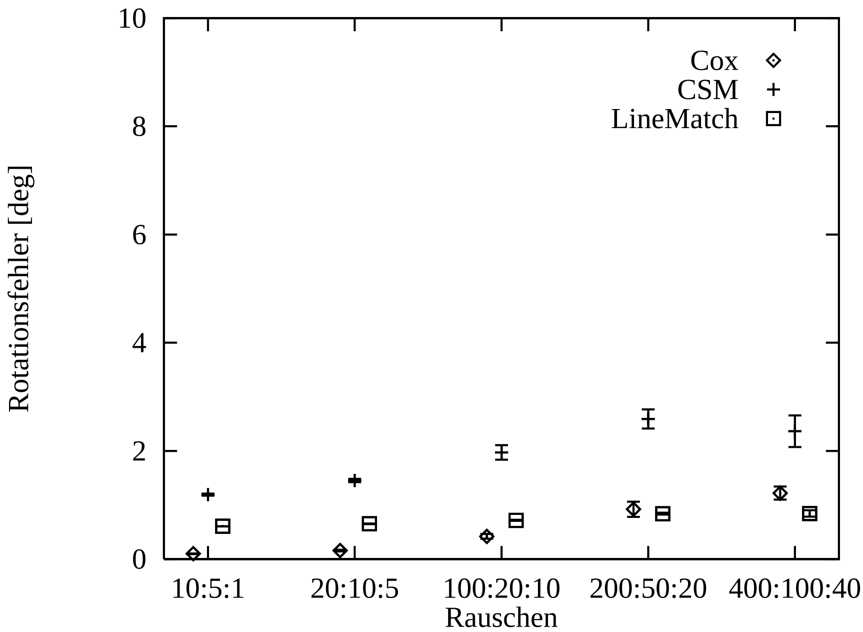


Abbildung 7.9: Rotationsfehler zu Referenzpositionen in typischen Spielszenario für verschiedene Stufen von Gauß'schem Odometrierauschen.

Aus beiden Abbildungen ist ersichtlich, daß alle drei Verfahren eine ähnliche Genauigkeit besitzen, die für gewöhnlich besser als $5cm$ und $2°$ ist. Nur der Cox-Algorithmus hat bei geringem Rauschen eine signifikant höhere Genauigkeit als die anderen Methoden, aber dies rührt daher, daß die Referenzpositionen ebenfalls mit dem Cox-Algorithmus bestimmt wurden.

Der LineMatch-Algorithmus ist jedoch wesentlich robuster als die anderen Methoden. Abbildung 7.10 zeigt die Anzahl der Positionen, an denen der Roboter seine Position verloren hatte, für die gleichen Stufen von Gauß'schem Rauschen wie in den vorigen Abbildungen. Hier liefert der LineMatch-Algorithmus sehr gute Resultate und ist in der Lage, den Roboter zu lokalisieren selbst bei hohem Odometrierauschen. Nur bei der größten verwendeten Rauschstufe fängt auch LineMatch an, die Roboterposition zu verlieren. Es wird vermutet, daß die höhere Robustheit des LineMatch-Verfahrens an dem größeren Suchbereich für das Finden von Überdeckungen liegt.

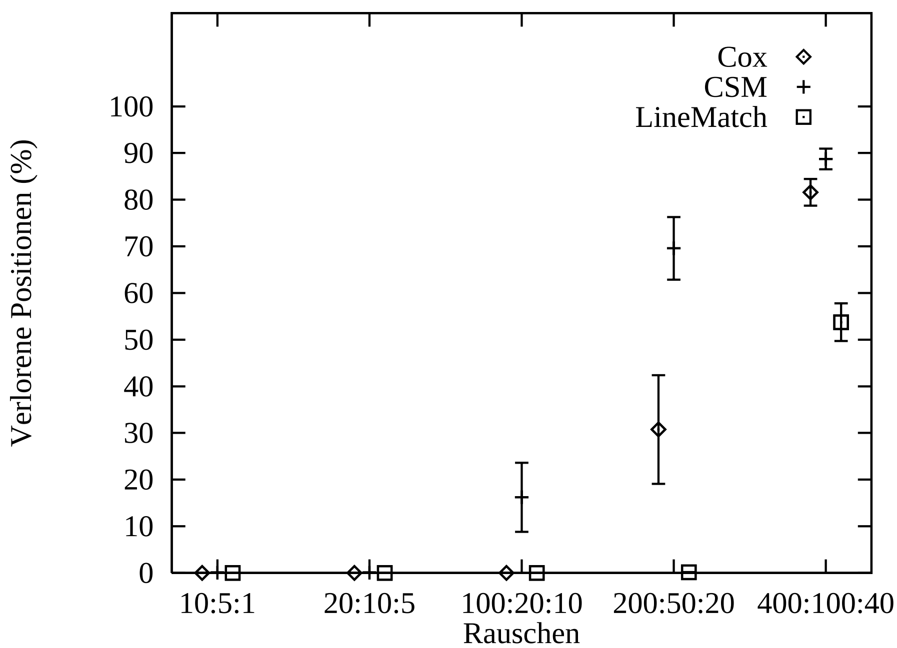


Abbildung 7.10: Anzahl der Positionen, an denen der Positionsfehler größer $0.5m$ oder größer $30°$ ist, für verschiedene Stufen von Gauß'schem Rauschen in einer typischen Spielsituation.

Auf die gleiche Weise wurde untersucht, wie die verschiedenen Methoden sich verhalten, wenn Stoßrauschen den Odometriedaten hinzugefügt wird. Bei der Genauigkeit ergaben sich ähnliche Resultate wie bei Hinzufügen von Gauß'schem Rauschen. Alle drei Methoden haben eine ähnliche Genauigkeit für den Abstands- und Rotationsfehler wie im Gauß'schen Fall. Abbildung 7.11 zeigt die Anzahl der

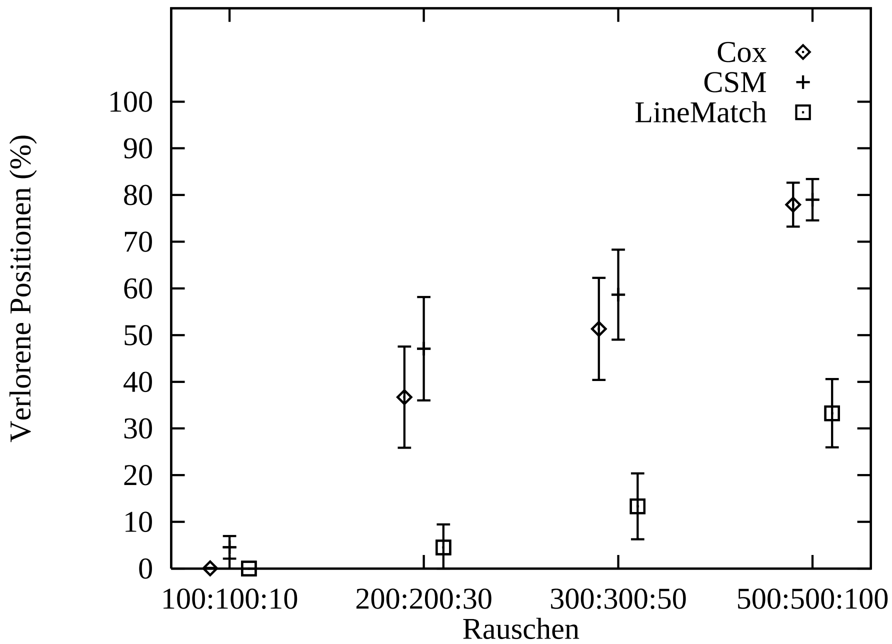


Abbildung 7.11: Anzahl der Positionen, an denen der Positionsfehler größer $0.5m$ oder größer 30° ist, für verschiedene Stufen von Stoßrauschen in einem typischen Spielszenario.

Positionen, an denen der Roboter verloren war, wenn Stoßrauschen den Odometriedaten hinzugefügt wird. Die Wertetripel auf der x-Achse entsprechen den Werten des hinzugefügten Stoßrauschens $\langle x, y, \alpha \rangle$ mit den Einheiten mm für x und y und Grad für α (siehe auch Abschnitt 4.6). Für das Stoßrauschen wurde eine Wahrscheinlichkeit von 0.2 pro Meter verwendet, d.h. mit einer Wahrscheinlichkeit von 20% stößt der Roboter gegen ein Hindernis, wenn er sich um einen Meter bewegt. Zusätzlich zu dem Stoßrauschen wurde ein geringes Gauß'sches Rauschen der Stärke $\langle 100, 5, 2 \rangle$ hinzugefügt. Wie in Abbildung 7.11 zu sehen ist, haben alle Scan-Matching-Verfahren Probleme, wenn Stoßrauschen vorhanden ist. Dies liegt daran, daß die Gaußverteilungsannahme bei der Fusion von Sensorbeobachtungen mit Odometriedaten das Stoßrauschen nicht besonders gut modelliert. Der LineMatch-Algorithmus zeigt jedoch weniger Ausfälle als die anderen Methoden und ist daher wieder robuster als die anderen Verfahren.

Ergebnisse in verwirrendem Spielszenario

Die verschiedenen Scan-Matching-Verfahren wurden auch in einem verwirrenden Spielszenario, bei dem eine lange Wand in das RoboCup-Spielfeld plaziert wurde, ausgewertet. Abbildung 7.12 zeigt die Daten, die in diesem Experiment gesammelt wurden.

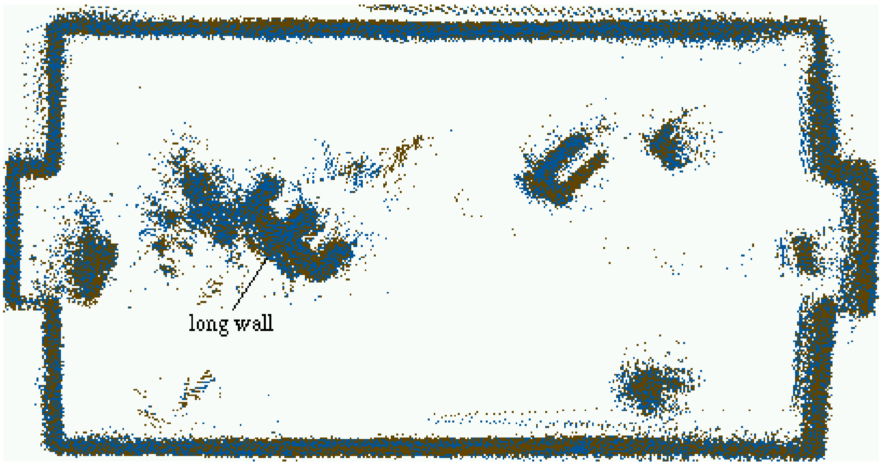


Abbildung 7.12: Verwirrendes Spielszenario: eine lange Wand wurde durch Aneinanderstellen von zwei Kartonschachteln im RoboCup-Spielfeld erzeugt, um den LineMatch-Algorithmus irre zu führen.

Es wird erwartet, daß der LineMatch-Algorithmus unter diesen Bedingungen irritiert wird, da die lange Wand nicht im Vorverarbeitungsschritt von LineMatch herausgefiltert wird und entweder eine falsche Überdeckung zur Folge hat oder die Positionsbestimmung alleine der Odometrie überlassen wird. Obwohl diese verwirrende Spielsituation sehr unwahrscheinlich zu sein scheint, kann sie dennoch auftreten, wenn ein Gegner versucht, zwei oder mehrere Roboter so zu plazieren, daß sie für die Laserscanner eine glatte Oberfläche bilden.

Glücklicherweise hat der LineMatch-Algorithmus keine allzu großen Probleme mit der verwirrenden Situation. Es wird vermutet, daß dies an der Tatsache liegt, daß es immer noch eine große Anzahl von Scans gibt, in denen die verwirrende Wand wegen des beschränkten Sichtbereichs des Laserscanners nicht vorhanden ist. Der Roboter ist daher nicht in der Lage sich zu lokalisieren, wenn die Wand in den Sensordaten vorhanden ist, kann sich aber wieder lokalisieren, sobald sie aus den Sensordaten verschwindet.

Für die Genauigkeit der verschiedenen Verfahren wurden wieder vergleichbare Ergebnisse wie in den vorherigen Experimenten erhalten. Die Robustheit des LineMatch-Algorithmus ist nun aber deutlich geringer und ist vergleichbar mit der des Cox- und CSM-Algorithmus. Abbildung 7.13 zeigt die Anzahl der Positionen, an denen der Roboter verloren war, für verschiedene Stufen von Gauß'schem Odometrierauschen. Hier zeigen alle drei Methoden eine ähnliche Robustheit.

Schließlich zeigt Abbildung 7.14 die Anzahl der Positionen, an denen der Roboter verloren war, bei Hinzunahme von Stoßrauschen. Auch hier ist der LineMatch-Algorithmus robuster als die anderen Methoden wegen des größeren Suchbereichs.

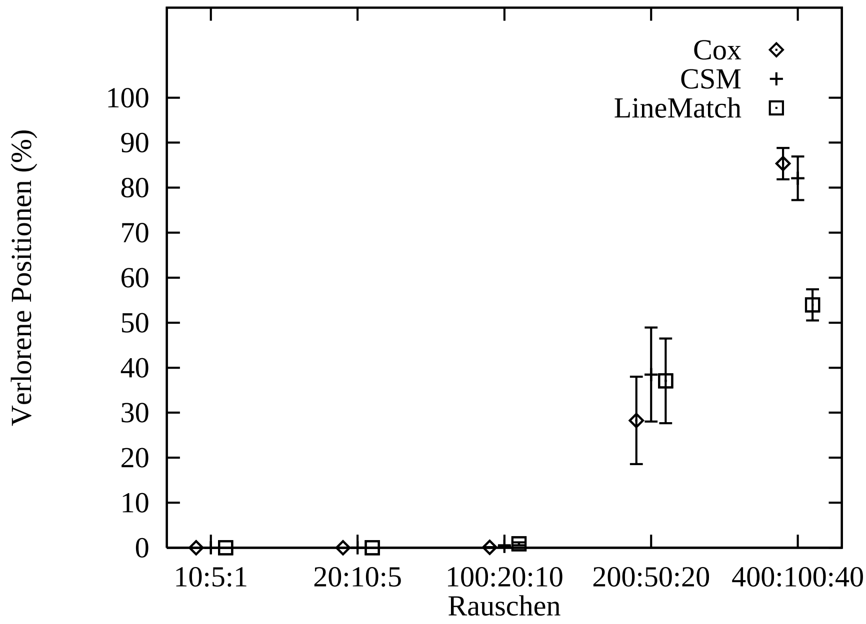


Abbildung 7.13: Anzahl Positionen, an denen der Positionsfehler größer $0.5m$ oder größer $30°$ war, in einem verwirrenden Spielszenario für verschiedene Stufen von Gauß'schem Rauschen.

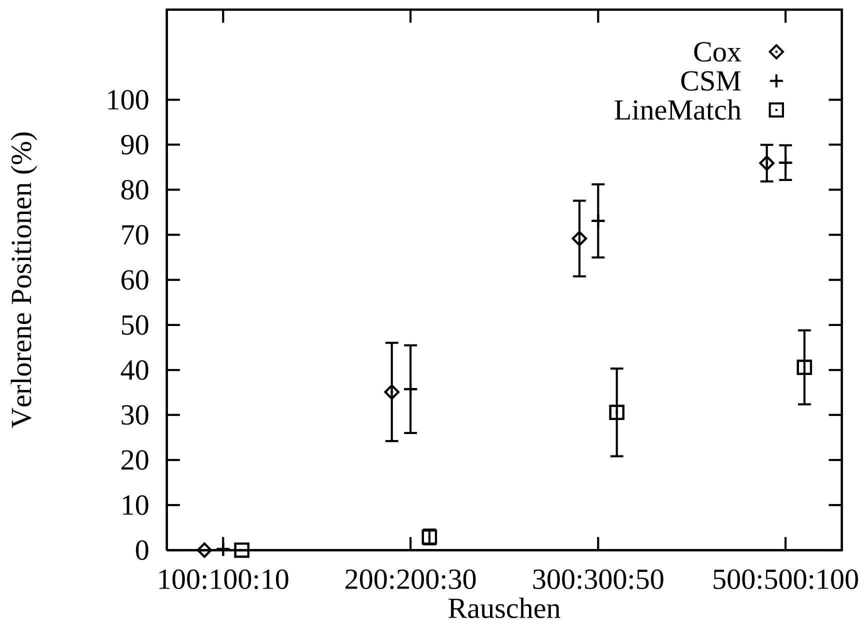


Abbildung 7.14: Anzahl Positionen, an denen der Positionsfehler größer $0.5m$ oder größer $30°$ ist, in einem verwirrenden Spielszenario für verschiedene Stufen von Stoßrauschen.

7.4.3 Diskussion

In diesem Abschnitt wurde eine neue Methode für die Überdeckung von Scans mit einem apriori Modell von Liniensegmenten präsentiert, die sehr gut für die Selbstlokalisierung in einer polygonalen und dynamischen Umgebung wie RoboCup geeignet ist. Experimentelle Untersuchungen bestätigen, daß die neue Methode wesentlich schneller und robuster als andere existierende Scan-Matching-Verfahren ist und trotzdem eine ähnliche Genauigkeit besitzt.

Die vorgeschlagene Methode wurde als eine der Schlüsselkomponenten des CS-Freiburg-Roboter-Fußballteams entwickelt und hat sich als schnell, zuverlässig, genau und robust erwiesen. Das Verfahren hat nie in einem offiziellen oder inoffiziellen Spiel versagt und führte das Team zu zahlreichen Erfolgen (siehe Abschnitt 7.9.

Obwohl die Methode bisher nur für RoboCup eingesetzt wurde, ist es ein offensichtlicher Schritt das Verfahren in anderen polygonalen Umgebungen, z.B. als Lokalisierungsmodul in dem in Kapitel 6 vorgestellten Navigationssystem, einzusetzen. Hierfür kann der Algorithmus auf verschiedene Weisen erweitert werden, um z.B. partielle Überdeckungen, bei denen nicht alle Linien des Scans mit Modellinien überdeckt werden, zu erlauben oder um den Algorithmus für den Einsatz in großen Umgebungen zu optimieren.

7.5 Erstellung lokaler und globaler Weltmodelle

Jeder Fußballagent interpretiert seine Sensorwerte mittels dem in Abbildung 7.15 abgebildetenen Wahrnehmungsmodul, um eigene Position, die Position der beobachteten Spieler und die Ballposition zu bestimmen.

Nachdem das Selbstlokalisierungsmodul einen Scan überdeckt hat, werden alle Scanpunkte, die zur Spielfeldumrandung gehören entfernt und die verbleibenden Punkte in Cluster aufgeteilt. Für jeden Cluster wird der Schwerpunkt berechnet und als ungefähre Position eines Roboters interpretiert (siehe Abbildung 7.16). Inhärent an diesem Ansatz ist der systematische Fehler aufgrund der verschiedenen Formen der Roboter.

Da die Laserscanner auf den Robotern so hoch montiert sind, daß sie über den Ball hinwegblicken, können diese nicht für die Ballerkennung benutzt werden. Selbst wenn die Scanner tiefer sitzen würden, ist es fraglich ob es möglich ist, den Ball von den Spielern aufgrund der Form zu unterscheiden, besonders dann, wenn man bedenkt, daß Spieler und Ball sehr dicht beieinander liegen können. Aus diesem Grund wird für die Ballerkennung ein kommerziell erhältliches Bildverarbeitungssystem eingesetzt.

Falls die Kamera ein Objekt einer bestimmten Farbe erkennt, so liefert des Bildverarbeitungssystem die Pixelkoordinaten des Objektzentrums zusammen mit dessen Breite, Höhe und Größe. Aus diesen Pixelkoordinaten wird die re-

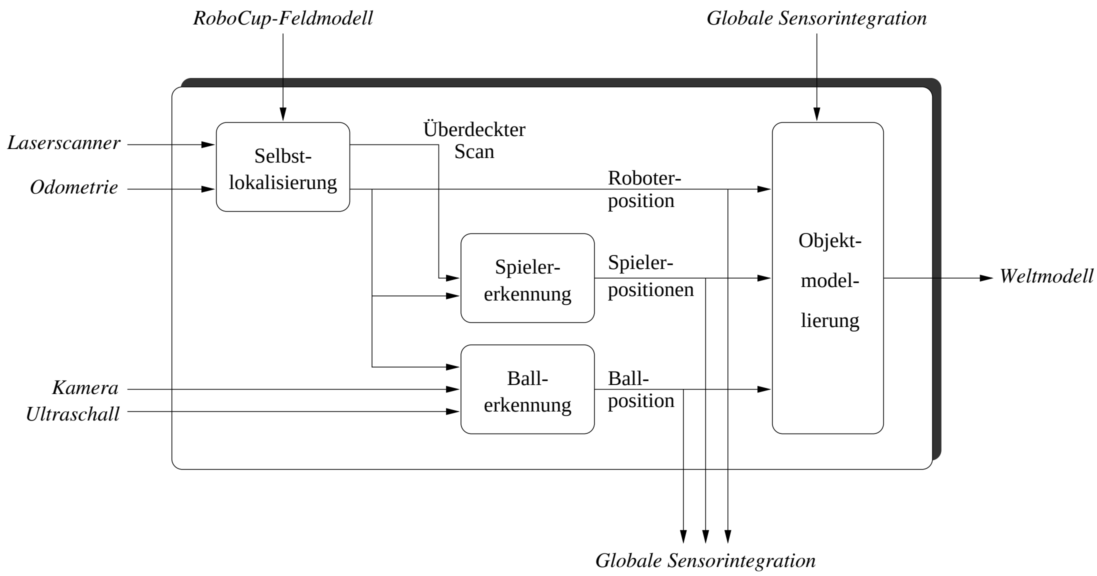


Abbildung 7.15: Das Wahrnehmungsmodul

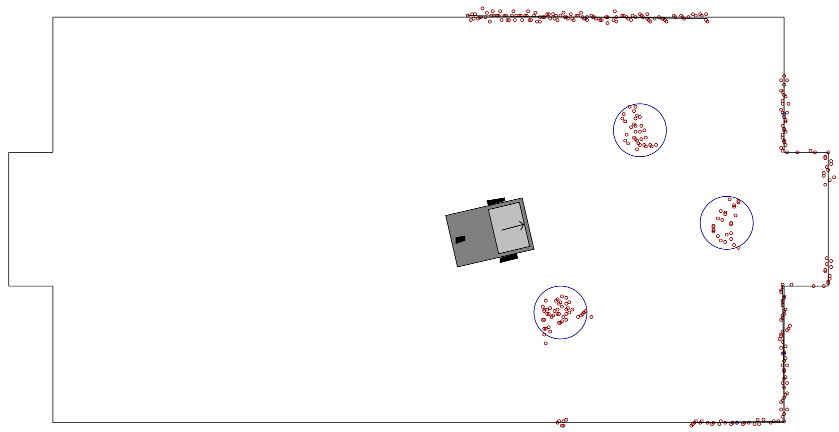


Abbildung 7.16: Spielererkennung: Liniensegmente werden aus einem Scan extrahiert, mit den Feldlinien überdeckt und drei Spieler werden aus den verbleibenden Scanpunkten generiert.

lative Ballposition bezüglich des Roboterkoordinatensystems berechnet, indem Pixelkoordinaten auf Polarkoordinaten (Abstand und Winkel zum Roboter) abgebildet werden. Die Abbildungsfunktion wird in einem Kalibrierungsschritt für einen Satz von vorgegebenen Positionen gelernt und für alle anderen Pixelkoordinaten wird interpoliert. Eine ausführliche Beschreibung dieses Ballerkennungssystem befindet sich in [Topor, 1998].

Da die robuste Ballerkennung eines der schwierigsten Probleme im Robotersystem des CS-Freiburg-Teams ist, wurden für die Weltmeisterschaft 1999 zahlreiche Erweiterungen wie z.B. Konturerkennung oder Plausibilitätsüberprüfungen vorgenommen, welche die Zuverlässigkeit, Genauigkeit und Reichweite des Bildverarbeitungssystems verbesserten. Diese Erweiterungen sind in der Arbeit von Thiel [1999] zu finden.

Aus der geschätzten Position des Spielers, der Positionen der anderen Objekte und des Balls – sofern sichtbar – konstruiert der Fußballagent sein eigenes *lokales Weltmodell*. Durch Verwalten einer Historie der Positionen aller Objekte können Orientierung und Geschwindigkeit der Objekte bestimmt werden. Ein Kalman-Filter wird eingesetzt, um Rauschen in den Sensorbeobachtungen zu reduzieren. Position, Orientierung und Geschwindigkeit werden an das Modul der *globalen Sensorintegration* geschickt.

Zusätzlich zu den direkt beobachtbaren Objekten, enthält das lokale Weltmodell auch Informationen über Objekte, die momentan nicht sichtbar sind. Zum einen wird ein Objekt nicht sofort aus dem Weltmodell entfernt, wenn es aus dem Sichtbereich des Roboters verschwindet. Durch Verwenden der zuletzt bekannten Position, Orientierung und Geschwindigkeit, wird die Objektposition noch für einige Sekunden fortgeschrieben. Zum anderen wird die Information des globalen Weltmodells verwendet, um das lokale Weltmodell zu erweitern. Objekte des globalen Weltmodells, die zu keinem Objekt des lokalen Weltmodells gehören und außerhalb des Sichtbereichs des Roboters liegen, werden zum lokalen Modell hinzugefügt, aber als nicht wirklich sichtbar markiert. Falls ein Objekt des globalen Weltmodells zu einem Objekt des lokalen Weltmodells korrespondiert, so hat die Information des lokalen Weltmodells Vorrang, da diese Daten höchstwahrscheinlich jünger und genauer sind. In diesem Fall wird die globale Information nur für die Bestimmung der Identität, z.B. ob Freund oder Gegner, verwendet.

Das *globale Weltmodell* wird aus den zeitgestempelten Daten (Position, Orientierung und Geschwindigkeit aller beobachteten Objekte), die jeder Spieler an die globale Sensorintegration schickt, generiert. Durch diese Schätzungen ist es einfach Freund und Gegner zu unterscheiden (siehe Abbildung 7.17).

Zu wissen wer und wo die eigenen Mitspieler sind, ist natürlich sehr hilfreich, um ein kooperatives Spiel zu ermöglichen. Eine andere Information, die sehr hilfreich ist, ist die globale Ballposition. Das Bildverarbeitungssystem kann den Ball nur bis zu einer Entfernung von 3–4 Metern erkennen. Die globale Ballposition zu wissen, selbst wenn der Ball nicht direkt sichtbar ist, ermöglicht dem Spieler seine Kamera in Richtung des Balles zu drehen, was eine Suche nach dem Ball durch Herumdrehen vermeidet. Dies ist insbesondere für den Torwart wichtig, der sonst möglicherweise einen Ball zu seiner linken übersieht, da er vielleicht gerade zu seiner rechten nach ihm sucht.

Es sollte jedoch bemerkt werden, daß durch den inhärenten Zeitversatz – von der Beobachtung eines Objekts bis zum Erhalt einer Nachricht über dieses Objekt von der globalen Sensorintegration – die Information des globalen Weltmodells

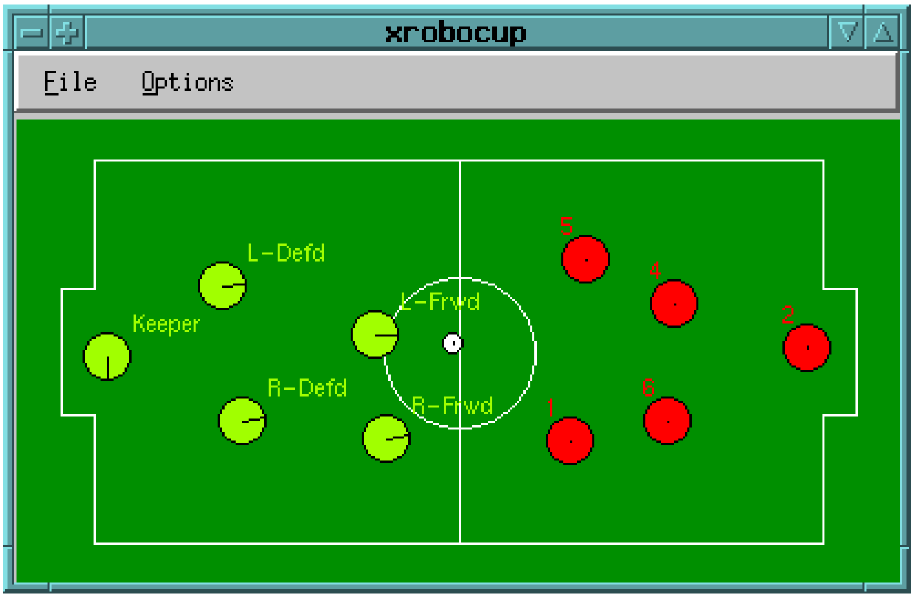


Abbildung 7.17: Visualisierung des Ergebnisses der globalen Sensorintegration.

immer 100–400 msec alt ist. Dies bedeutet, daß das Verhalten der Roboter nicht direkt über die Informationen des globalen Weltmodells zentral gesteuert werden kann. Trotzdem gibt es neben den bereits oben aufgeführten Vorzügen eine Zahl weiterer wichtiger Probleme, die anhand des globalen Weltmodells gelöst werden können.

Zum Beispiel könnte das globale Weltmodell dazu benutzt werden, disorientierte Mitspieler zu erkennen und neu zu orientieren. Obwohl diese Situation noch nie in einem Spiel aufgetreten ist, ist so ein Sicherheitsmechanismus sicherlich wertvoll und ist zukünftiger Forschungsgegenstand.

Weiterhin ermöglichen die Informationen des globalen Weltmodells die Erkennung unzuverlässiger Sensoren der Roboterfußballspieler. Beispielsweise wurde in manchen Spielen bei der deutschen Meisterschaft VisionCup'98 der Ball von einem Roboter an einem falschen Ort gesehen. Werden die einzelnen Beobachtungen der Spieler nur gemittelt, wie es in der damaligen Version implementiert war, so ist eine völlig inplausible Ballposition die Folge. Aus diesem Grund wurde für RoboCup'99 eine wesentlich robustere Fusion der Ballbeobachtungen entwickelt, die auf einer Kombination von Markov-Lokalisierung für eine hohe Robustheit und Kalman-Filterung für eine hohe Genauigkeit beruht [Dietl, 1999].

Schließlich kann das Weltmodell auch verwendet werden, um strategische Entscheidungen zu treffen, beispielsweise für die dynamische Rollenzuweisung [Veloso *et al.*, 1998a; Reetz, 1999].

7.6 Steuerung und Kooperation

Die Entscheidungen eines Fußballagenten basieren hauptsächlich auf der augenblicklichen Situation seines lokalen Weltmodells. Um jedoch ein kooperatives Teamverhalten zu verwirklichen, werden die tatsächlichen Entscheidungen auch aufgrund einer *Rollenzuteilung* und durch Austausch der Intentionen der Spieler getroffen.

Obwohl die Ausführungssteuerung als verhaltensbasiert beschrieben werden kann, weicht der hier vorgestellte Ansatz signifikant von Ansätzen ab, bei denen die Verhalten direkt von uninterpretierten Sensorwerten abgeleitet werden [Brooks, 1986] wie es beispielsweise im Team *Ullanta* [Werger *et al.*, 1998] der Fall ist. Im Ansatz des CS-Freiburg-Teams bestimmen Merkmale höherer Ordnung, die von Sensordaten und durch Kommunikation mit anderen Agenten abgeleitet werden, welches Verhalten aktiviert wird. Weiterhin können einzelne Verhalten komplexe Prozesse, wie z.B. die Planung eines kollisionsfreien Pfades zu einem bestimmten Zielpunkt (siehe Abschnitt 7.7), hervorrufen.

7.6.1 Basisfähigkeiten und verhaltensbasierte Steuerung

Das verhaltensbasierte Steuerungsmodul besteht aus einem regelbasierten System, welches Zustände auf Aktionen abbildet. Das Regelsystem wird alle 100 msec neu ausgewertet, so daß auf Situationsänderungen sehr schnell reagiert werden kann. Je nach Rolle des Spielers, ob Torwart oder Feldspieler, gibt es verschiedene Regelsysteme.

Der Torwart ist sehr einfach beschaffen und muß nur verhindern, daß der Ball ins eigene Tor rollt. Er beobachtet immer den Ball – mittels den Informationen des globalen Weltmodells, falls die eigene Kamera ihn nicht sieht – und fährt zu dem Punkt, an dem er den Ball aufgrund der geschätzten Bewegungsrichtung des Balles abzupassen erwartet. Ist der Ball auf der linken oder rechten Seite des Tors, so fährt der Roboter zur entsprechenden Seite und dreht sich zum Ball hin. Damit der Roboter schneller auf der Torlinie hin- und herfahren kann, wurde ein spezieller Aufbau gewählt, bei dem der „Kopf“ des Roboters um 90° gedreht wurde (siehe Abbildung 7.1). Trifft der Ball den Torwart, so wird er durch die automatische Schußvorrichtung zurück ins Feld befördert.

Die Feldspieler haben einen reicheren Satz von Basisfähigkeiten. In der folgenden Beschreibung behandeln die ersten vier Fähigkeiten, Situationen, in denen der Ball nicht direkt gespielt werden kann, während die letzten beiden den Umgang mit dem Ball betreffen.

Approach-position: Fahre eine Zielposition sorgfältig an.

Go-to-position: Plane (fortlaufend) einen kollisionsfreien Pfad von der aktuellen Roboterposition zu einer Zielposition und folge diesem Pfad bis die Zielposition erreicht ist.

Observe-ball: Richte den Roboter so aus, daß der Ball in der Fokusmitte liegt. Verfolge den Ball ohne ihn anzufahren.

Search-ball: Drehe den Roboter, um den Ball zu finden. Falls der Ball nach einer vollen Umdrehung nicht gefunden wurde, so fahre zu einer Heimatposition und suche erneut von dort.

Move-ball: Bestimme eine gerade Linie von der aktuellen Roboterposition zu einem Zielpunkt im gegnerischen Tor, welche den maximalen Abstand zu allen Objekten auf dem Spielfeld hat. Folge dieser Linie mit steigender Geschwindigkeit und bestimme die Linie erneut von Zeit zu Zeit.

Shoot-ball: Um den Ball zu beschleunigen, benutze entweder die Schußvorrichtung oder drehe den Roboter ruckartig. Welche Aktion zum Einsatz kommt und in welche Richtung gedreht werden soll hängt von der aktuellen Spielsituation ab.

Die Abbildung von Situationen zu Aktionen ist anhand eines Entscheidungsbaumes implementiert und ist in Abbildung 7.18 zu sehen. Das Regelsystem wird permanent (alle $100ms$) neu ausgewertet und berücksichtigt die gerade ausgeführte Aktion und das aktuelle Weltmodell. Mögliche Aktionen sind:

FreeFromStall: Wähle und folge einem Pfad zu einer freien Position auf dem Spielfeld.

GoToHomePos: Fahre zur Heimatposition durch Verwenden der *go-to-position*-Fähigkeit.

LineUp: „Freue Dich“ (Musik abspielen und auf der Stelle drehen), fahre anschließend zur Heimatposition.

SearchBall: Starte das *search-ball*-Verhalten .

PlayBall: Angriff durch Einsatz der Fähigkeiten *approach-position*, *move-ball* und *shoot-ball*.

ObserveBall: Beobachte den Ball durch das Verhalten *observe-ball*.

GoToBall: Fahre zum Ball mittels der Verhalten *go-to-position* und *approach-position*.

Es sollte bemerkt werden, daß die taktischen Details der Verhalten laufend geändert wurden, sogar noch, nachdem die Weltmeisterschaft 1998 in Paris bereits begonnen hatte. Zum Beispiel wurde das *FreeFromStall*-Verhalten nach dem ersten Spiel so verändert, daß ein Roboter nicht mehr zurückweicht, wenn er in Ballbesitz ist. In diesem Spiel kam es vor, daß der Gegner permanent mit dem Ball gegen einen unserer Spieler gedrückt hat, dieser dann aufgrund des Blockieren

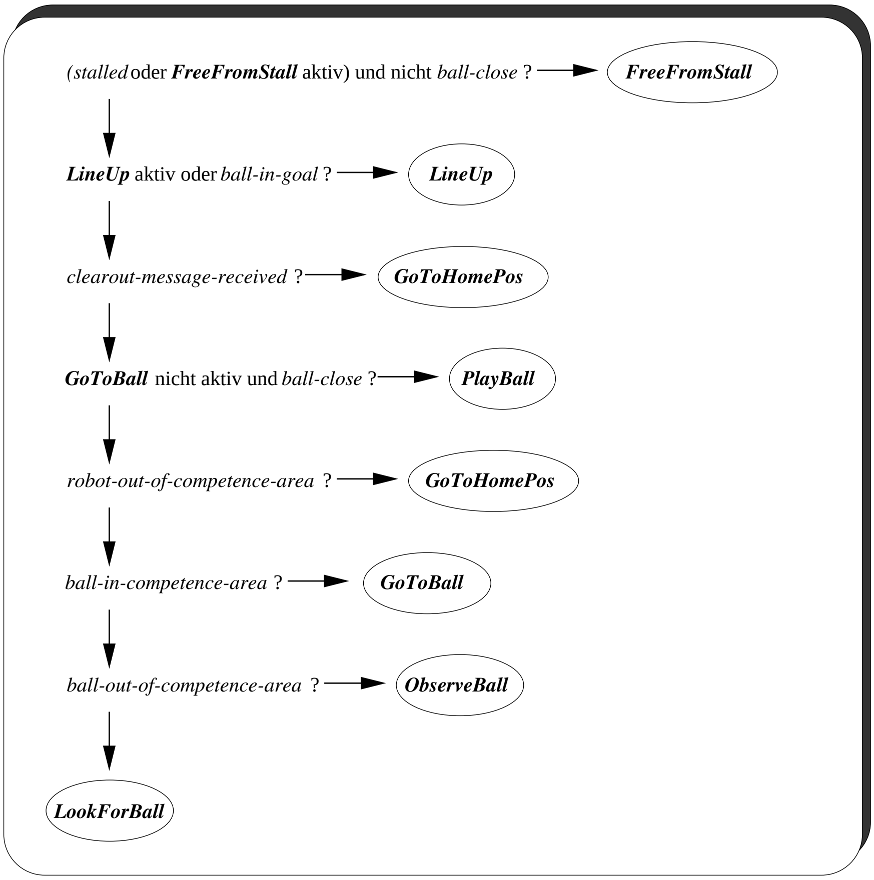


Abbildung 7.18: Entscheidungsbaum für die Aktionsauswahl. Linkspfeil für positive Antwort, Rechtspfeil für negative Antwort. Kreise zeigen neue Zustände an.

der Räder in den *FreeFromStall*-Zustand geriet und dem Gegner Platz machte. Die Fähigkeit ein geschossenes Tor zu erkennen und den Torschützen in einen „Zustand der Freude“ zu versetzen, war nicht sehr nützlich, da sofort nachdem der Ball über die Torlinie rollte, Personen das Spielfeld betraten, um die Roboter neu aufzustellen.

Da die Aktionsauswahl mittels dem hier vorgestellten regelbasierten System schwierig zu modellieren ist (der Entwurf des Entscheidungsbaumes in Abbildung 7.18 bedarf sorgfältiger Überlegungen) wurde für die Teilnahme an RoboCup'99 ein anderes Konzept für die Aktionsauswahl entwickelt. Dieses enthält für jede mögliche Aktion Vorbedingung und Invariante und berechnet eine *Aktivierungsenergie*. Die Aktion mit der höchsten Aktivierungsenergie wird ausgeführt. Dieses Schema erlaubt eine wesentlich einfachere Modellierung der Aktionsauswahl. Einzelheiten sind in der Arbeit von Reetz [1999] zu finden.

7.6.2 Multi-Agenten-Koordination

Würden sich alle Fußballspieler nach dem gleichen Regelwerk verhalten, so würde ein „Schwarm von Robotern“ dem Ball hinterherjagen und die Spieler würden sich sehr wahrscheinlich gegenseitig behindern. Eine Möglichkeit, dieses Problem zu vermeiden, ist es, verschiedene *Rollen* und *Kompetenzbereiche* den Spielern zuzuordnen (siehe Abbildung 7.19).

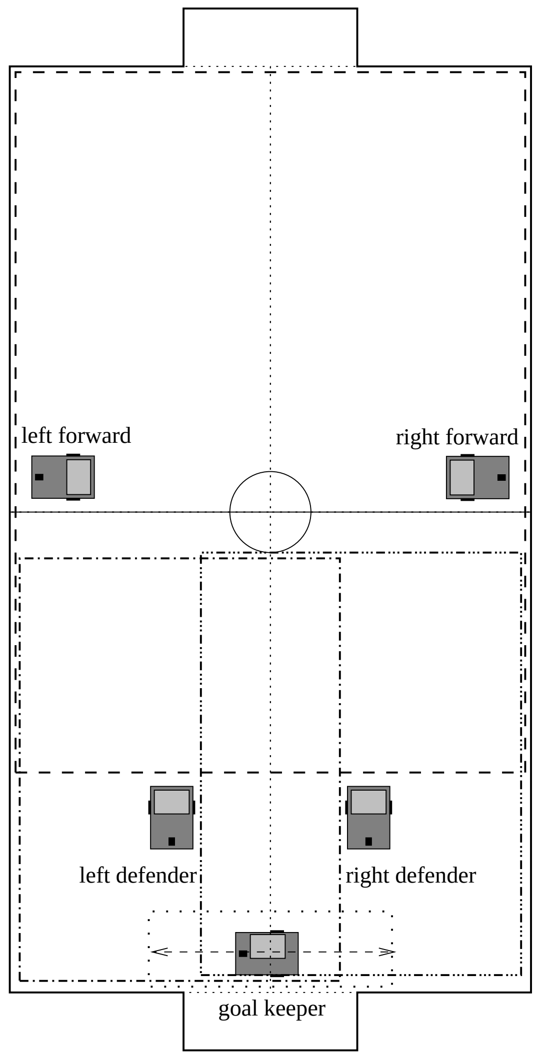


Abbildung 7.19: Rollen und Kompetenzbereiche.

Wären die Kompetenzbereiche nicht-überlappend, so sollte es keine Interferenz zwischen Mitspielern geben und es wäre auch keine Kommunikation zwischen den Spielern nötig. Jeder Spieler würde zum Ball fahren und ihn in den nächsten Kompetenzbereich oder ins gegnerische Tor spielen. In der Tat war dies der anfängliche Entwurf für die Kooperation zwischen den Robotern und wurde im ersten Spiel bei RoboCup'98 eingesetzt.

Die strikte Aufteilung in Kompetenzbereiche führt jedoch zu zahlreichen Problemen. Erstens können die Spieler sich immer noch an den Grenzen zwischen den Kompetenzbereichen gegenseitig stören. Zweitens, falls ein Spieler durch das andere Team blockiert wird, ausfällt oder aus dem Spielfeld herausgenommen wird, so übernimmt kein anderer Mitspieler diesen Kompetenzbereich und der Ball bleibt in diesem Bereich dem Gegner alleine überlassen. Drittens, falls ein Abwehrspieler die Chance besitzt, den Ball ins gegnerische Tor zu dribbeln, so wird einer der Stürmer sehr wahrscheinlich den Abwehrspieler blockieren. Aus diesen Gründen wurde der anfängliche Entwurf nach dem ersten Spiel in Paris signifikant geändert und weitere Mechanismen kamen zum Einsatz.

Sobald ein Spieler in einer guten Situation ist, um den Ball zu spielen, sendet er sogenannte *Clear-Out*-Nachrichten an alle Mitspieler. Empfängt ein Mitspieler eine solche Nachricht, so versucht er, dem angreifenden Spieler aus dem Weg zu gehen, indem er an den Spielfeldrand fährt (siehe Abbildung 7.20). Dies verhindert, daß sich Spieler gegenseitig behindern. Für die Teilnahme an RoboCup'99 wurden weitere Nachrichtentypen eingesetzt, um. z.B. bereits abzusprechen, wer überhaupt zum Ball fahren soll [Reetz, 1999]. Mit anderen Worten wird die Kooperation zwischen Spielern durch Kommunikation realisiert, ähnlich wie beispielsweise im *Uttori*-Team [Yokota *et al.*, 1999]. Durch den Einsatz von Kommunikation können sich die Kompetenzbereiche überlappen, die beiden Stürmer können z.B. die vorderen zwei Drittel des Spielfeldes gemeinsam nutzen (siehe Abbildung 7.19) und Angriffe auf das gegnerische Tor werden durch Kommunikation koordiniert. Für RoboCup'99 wurden die Kompetenzbereiche ganz abgeschafft und die Raumaufteilung nur durch verschiedene Heimatpositionen und durch Kommunikation realisiert [Reetz, 1999].

Für RoboCup'98 gab es keine spezielle Koordination für Defensiv-Bewegungen. In der Tat ergab sich ein Defensiv-Verhalten automatisch aus der verhaltensbasierten Steuerung. Sobald der Ball in die eigene Spielfeldhälfte rollte, fuhren die Abwehrspieler zum Ball und blockierten so den Angriff. Während diese einfache Verteidigungsstrategie recht gut funktionierte, wurden für RoboCup'99 Erweiterungen vorgenommen. Beispielsweise wurden Verhalten eingebaut, welche einen Spieler zum Blocken eines Angriffs zwischen Ball und eigenem Tor bewegen. Weiterhin werden den Spielern verschiedene Heimatpositionen je nach Spielsituation zugewiesen, wodurch taktische Aufstellungen möglich sind [Reetz, 1999].

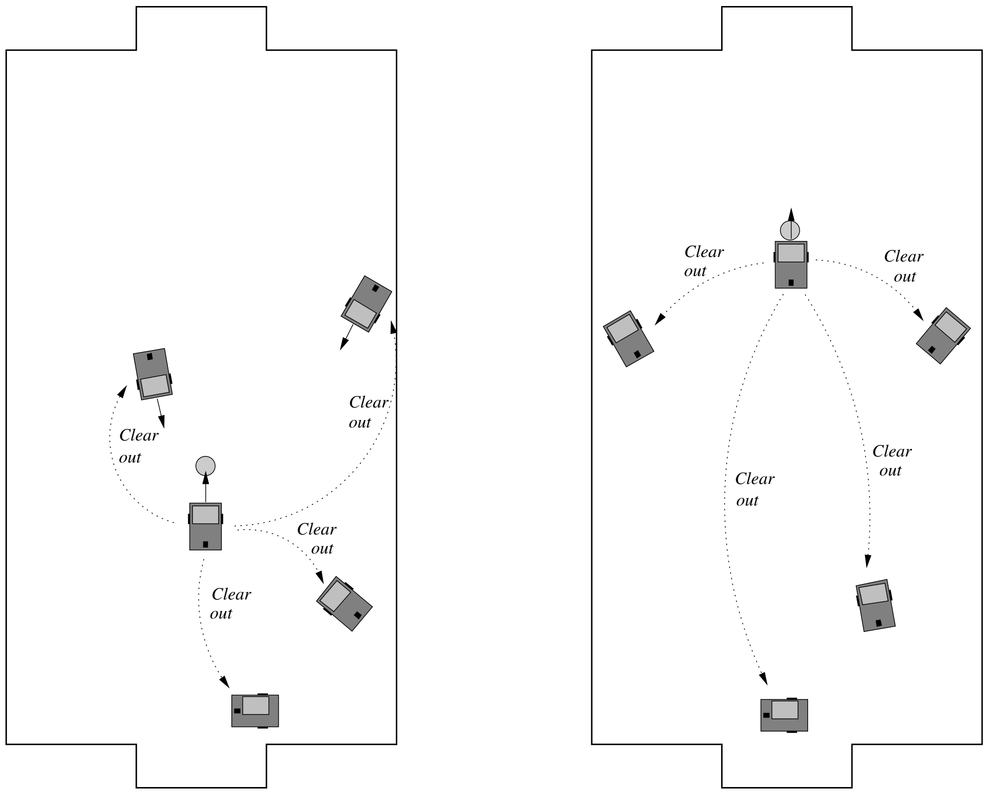


Abbildung 7.20: Koordination durch Versenden von *Clear-Out*-Nachrichten.

7.7 Wegeplanung

Manche der oben beschriebenen Basisfähigkeiten betreffen die Bewegung des Fußballroboters zu einer Zielposition auf dem Spielfeld. Während diese Aufgabe sicherlich verhaltensbasiert realisiert werden kann, wurde ein Bewegungsplaner hierfür eingesetzt, um Probleme wie beispielsweise lokale Minima zu vermeiden.

Bewegungsplanung unter dem Vorhandensein bewegter Hindernisse ist bekanntlich ein rechenzeitaufwendiges Problem [Latombe, 1991]. Weiterhin sind gefundene Lösungswege aufgrund der zeitintensiven Berechnung wahrscheinlich bereits veraltet, bevor sie generiert sind. Aus diesem Grund wurde das Problem der Pfadplanung mit bewegten Hindernissen approximiert, indem für die Wegesuche alle Objekte als stationär betrachtet werden. Obwohl solch eine Vorgehensweise unangemessen in einer Umgebung mit hoher Dynamik zu sein scheint, hat die Erfahrung gezeigt, daß gegnerische Spieler sehr oft als stationäre Objekte betrachtet werden können. Zudem ist die implementierte Pfadplanung so effizient (wenige Millisekunden für 4–5 Hindernisse), daß laufend neu geplant werden kann.

Um beliebige Pfade um Hindernisse zu planen, wird die Methode des *erweiterten Sichtbarkeitsgraphen* [Latombe, 1991] verwendet. Alle Objekte im Weltmodell werden vergrößert und das Spielfeld verkleinert, was die Wegesuche für einen zu einem Punkt geschrumpften Roboter erlaubt. Die eigentliche Pfadplanung wird durch einen A^*-Algorithmus realisiert, welcher kürzeste, kollisionsfreie Pfade findet, die den Roboter von seiner aktuellen Position auf geraden Liniensegmenten und Kreisbogensegmenten zum Zielpunkt führen (siehe Abbildung 7.21 (b)).

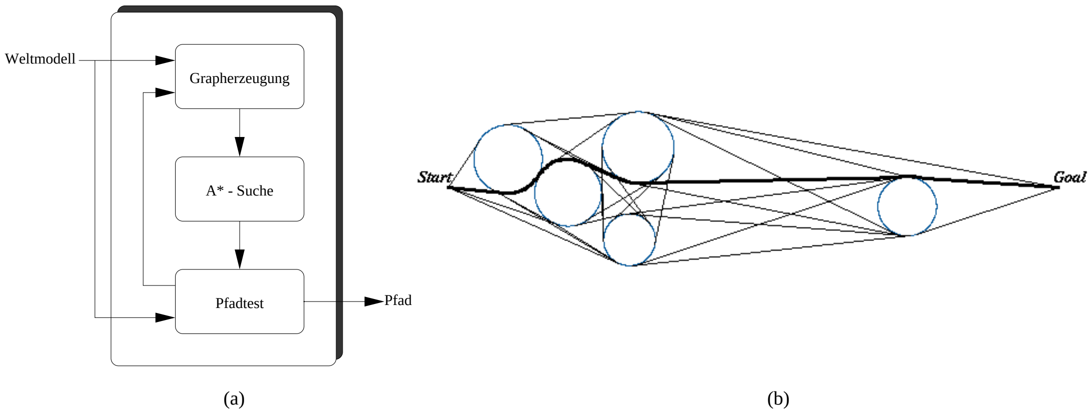


Abbildung 7.21: Pfadplanungsmodul (a) und erweiterter Sichtbarkeitsgraph (b).

Um die Laufzeit des Pfadplanungsalgorithmus zu verbessern, wird ein iteratives Verfahren verwendet (siehe Abbildung 7.21 (a)). Zunächst wird alleine mit Start- und Zielknoten ein Weg geplant und Objekte werden nur dann dem Graphen hinzugefügt, wenn sie mit einem gefunden Graphen in Konflikt stehen. Auf diese Weise können sehr schnell Pfade gefunden werden, z.B. wenn der direkte Weg zum Zielpunkt frei ist, ohne dabei den vollständigen Sichtbarkeitsgraphen aufbauen zu müssen.

Da der Pfadplaner in jedem Zyklus neu aufgerufen wird, besteht die Gefahr, daß Oszillationen bei ähnlich langen, aber verschiedenen Pfaden auftreten können. Um dies zu verhindern, werden Pfade, die eine große Änderung in der anfänglichen Roboterorientierung verursachen, mit Zusatzkosten bestraft.

Obwohl dieser Pfadplanungsalgorithmus sehr erfolgreich für RoboCup'98 eingesetzt wurde, besitzt er auch Nachteile. Die Vergrößerung der Hindernisse, um mit einem zu einem Punkt geschrumpften Roboter zu planen, wird über einen Parameter eingestellt. Ist dieser zu klein, so besteht die Gefahr, daß der Roboter mit einem Hindernis kollidiert, da der erweiterte Sichtbarkeitsgraph die Wege genau an den Rand der (vergrößerten) Objekte legt. Ist er zu groß, so kann es vorkommen, daß der Planer keinen Weg mehr findet, obwohl physikalisch Platz ist, um zwischen den Hindernissen durchzufahren. Aus diesem Grund wurde für RoboCup'99 ein potentialfeldbasierter Wegeplaner eingesetzt, welcher die Eigenschaft hat, bei viel Freiraum möglichst große Bögen um Hindernisse zu fahren aber bei engen Situationen dennoch in der Lage ist, vorsichtig zwischen Objekten hindurchzufahren. Da der potentialfeldbasierte Wegeplaner in lokale Minima laufen kann, wurde das Verfahren mit einem gitterbasierten A^*-Verfahren komplementiert, das den Roboter aus lokalen Minimas herausführt. Abbildung 7.22 zeigt eine Beispielkonfiguration, bei der ein Pfad von der rechten oberen Ecke zur linken unteren Ecke gesucht wird, der Algorithmus in ein lokales Minima läuft und aus diesem mittels A^*-Suche wieder herausfindet. Eine genaue Beschreibung des Verfahrens ist in der Arbeit von Topor [1999] zu finden.

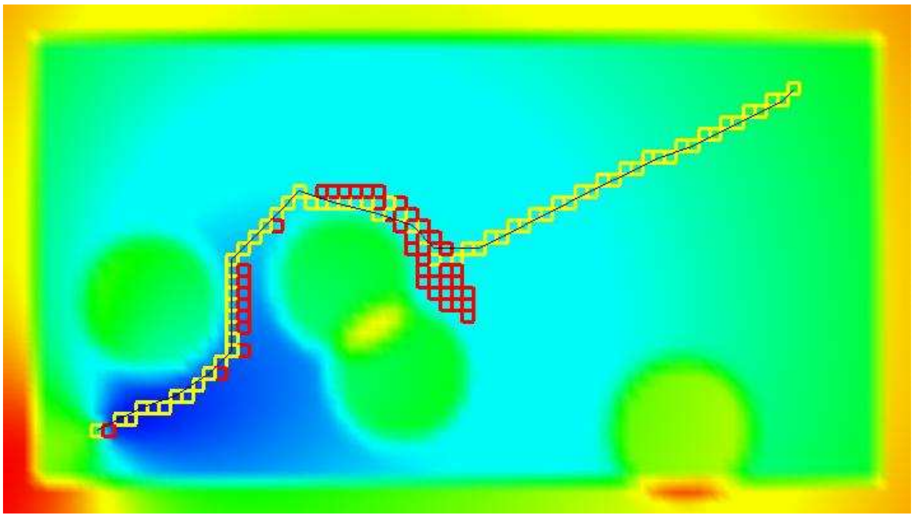


Abbildung 7.22: Pfadplanung mit Potentialfeldmethode und A^*-Algorithmus.

7.8 Erfahrungen bei Turnieren

Die Teilnahme an RoboCup'98 war für das CS-Freiburg-Team in zweierlei Hinsicht sehr fruchtbar. Zum einen gab es die Möglichkeit, Ideen mit anderen Teams auszutauschen und deren Lösungsansätze kennenzulernen. Zum anderen konnte sehr viel durch die Spiele gelernt werden. Wie bereits oben beschrieben, wurden Taktiken und Strategien neu entworfen nachdem die Erfahrungen von den Spielen gesammelt und ausgewertet wurden.

Die Erfahrungen mit Hard- und Software-Zuverlässigkeit sind gemischt. Laserscanner und Selbstlokalisierung funktionierten ohne Probleme, während die Funkverbindung manchmal gestört war, möglicherweise durch andere Teams, die zur gleichen Zeit auf einem anderen Feld spielten. Das anfälligste Teil des Systems ist das Bildverarbeitungssystem, nicht wegen etwaiger Hardware-Ausfälle, sondern weil leichte Veränderungen in den Lichtverhältnissen zu ernsthaften Problemen in der Ballerkennung führen. Dies scheint jedoch ein Problem für alle Manschaften zu sein.

Die Ergebnisse des CS-Freiburg-Teams sind sehr zufriedenstellend. Im ersten Auftritt bei RoboCup'98 gewann das Team die Weltmeisterschaft, hatte das beste

Torverhältnis (12:1), verlor keines der Spiele und schoß fast 25% aller Tore. Daß diese Leistung kein Zufall war, wurde auf der Deutschen Meisterschaft *VISION-RoboCup'98* bewiesen, bei dem das Team ebenfalls ungeschlagen gewann. Die Teilnahme an RoboCup'99 war ebenfalls erfolgreich und das Team schloß mit einem Torverhältnis von 33:2 und einem sehr spannenden Halbfinalspiel gegen die Mannschaft aus Italien mit dem dritten Platz ab.

Die Schlüsselkomponenten für diesen Erfolg sind höchstwahrscheinlich die Selbstlokalisierungs- und Objekterkennungsmethoden, die auf Daten des Laserscanners basieren und es erlauben, genaue und zuverlässige lokale und globale Weltmodelle zu erstellen. Auf Basis dieser Weltmodelle wurde es möglich, reaktive Pfadplanung, fein justierte Verhalten und Multi-Agenten-Kooperation zu verwirklichen, was sehr hilfreich für ein gutes und erfolgreiches Spiel ist. Schließlich spielte sicherlich auch der selbstgebaute Schußapparat und der Ballsteuerungsmechanismus eine Rolle für ein erfolgreiches Roboter-Fußballspiel.

7.9 Ergebnisse bei offiziellen Spielen

Weltmeisterschaft RoboCup'98, Paris, 2.7-9.7.98:

CSF - NAIST	=	1 : 1 (nach Elfmeterschießen)
CSF - USC	=	1 : 0
CSF - Real Magicol	=	3 : 0
CSF - Yale	=	2 : 0
CSF - Osaka	=	3 : 0
CSF - Tübingen	=	2 : 0

CS-Freiburg ist Weltmeister 98.

Deutsche Meisterschaft VisionCup'98, Stuttgart, 30.9-1.10.98:

CSF - Ulm	=	4 : 0
CSF - Tübingen	=	4 : 1 (nach Elfmeterschießen)
CSF - München	=	2 : 0

CS-Freiburg ist Deutscher Meister 98.

Weltmeisterschaft RoboCup'99, Stockholm, 28.7-4.8.99:

CSF - ISocRob	=	5 : 0
CSF - GMD	=	6 : 0
CSF - KIRC	=	6 : 0
CSF - RMIT	=	6 : 0
CSF - Wisley	=	4 : 0
CSF - NAIST	=	4 : 1
CSF - ART	=	$0 : 0^+$
CSF - Wisley	=	2 : 1

CS-Freiburg gewinnt Bronze 99.

Deutsche Meisterschaft VisionCup'99, Stuttgart, 12-13.10.99:

CSF - GMD	=	6 : 0
CSF - Tübingen	=	4 : 0
CSF - München	=	$2^+ : 2$
CSF - Stuttgart	=	4 : 1

CS-Freiburg ist Deutscher Meister 99.

Kapitel 8

Zusammenfassung und Ausblick

Die robuste Navigation autonomer mobiler Systeme ist ein großes Forschungsgebiet, das sich grob in die drei Bereiche Selbstlokalisierung, Kartenerstellung und Pfadplanung einteilen läßt. In dieser Arbeit wurden alle drei Bereiche umfassend untersucht und neue Methoden und Resultate erarbeitet. Dieses Kapitel faßt die gesamte Arbeit zusammen und gibt einen Ausblick auf zukünftige Arbeiten.

8.1 Zusammenfassung

Zunächst wurden in Kapitel 3 grundlegende Algorithmen für die Verarbeitung von Laserscannerdaten vorgestellt. Neben Algorithmen für die Extraktion von Merkmalen aus Laserscans, der Projektion von Scans und verschiedener Filterverfahren, wurde der Scan-Matching-Begriff für das Überdecken von Scans definiert. Zwei verschiedene Scan-Matching-Methoden, der Cox- [Cox, 1991] und der IDC-Algorithmus [Lu und Milios, 1997b], wurden vorgestellt und eine Kombination beider Methoden vorgeschlagen, welche die Vorteile beider Verfahren verbindet und ihre Nachteile vermeidet [Gutmann und Schlegel, 1996]. Schließlich wurde der Ansatz in Bezug zu anderen in der Literatur zu findenden Arbeiten gesetzt.

Ein auf Scan-Matching aufbauendes Verfahren zur Selbstlokalisierung eines mobilen Roboters wurde in Kapitel 4 vorgestellt. Ein besonderes Augenmerk wurde auf Scan-Matching mit der Methode von Cox und dem kombinierten Algorithmus gelegt und Ergebnisse von Lokalisierungsexperimenten aus einer Büroumgebung sowie einem Museum präsentiert. Das Verfahren schnitt hierbei in beiden Fällen erfolgreich ab und insbesondere in der Museumsumgebung konnte der Roboter trotz vieler störender Hindernisse und umherlaufender Personen seine Position auch über längere Zeit immer bestimmen.

Weiterhin wurde das Verfahren mit einer anderen Selbstlokalisierungsmethode, der Markov-Lokalisierung [Burgard *et al.*, 1996; Fox, 1998], experimentell verglichen. Hierzu wurden verschiedene Störquellen den Sensordaten hinzugefügt. Zum einen wurden den Odometriedaten Gauß'sches Rauschen und Stoßrauschen hinzugefügt. Zum anderen wurden Experimente mit den Sensordaten einer Roboterfahrt durchgeführt, bei der sehr viele umherlaufende Personen die Sensoren des Roboters blockierten. In zahlreichen Experimenten wurden so Genauigkeit und Robustheit der beiden Selbstlokalisierungsmethoden bestimmt. Hierbei stellte sich heraus, daß Scan-Matching wesentlich genauer als Markov-Lokalisierung ist, meistens sogar um mehr als eine Zehnerpotenz. Markov-Lokalisierung ist dagegen wegen des größeren Suchraums wesentlich robuster als Scan-Matching, besonders wenn die Sensor- und Odometriedaten stark verrauscht sind [Gutmann *et al.*, 1998a].

Aus diesem Grund wird eine Kombination der beiden Ansätze vorgeschlagen, welche die Vorteile der Verfahren verbindet. Hierbei wird Markov-Lokalisierung dazu verwendet, die globale Roboterposition zu bestimmen und die Positionsverfolgung der Scan-Matching-Methode zu überwachen. Falls die durch Scan-Matching bestimmte Position nicht mehr plausibel ist, so erkennt Markov-Lokalisierung dies und lokalisiert den Roboter gegebenenfalls neu.

Damit sich ein Roboter überhaupt lokalisieren kann benötigt er eine Umgebungskarte. Um diese nicht von Hand oder mittels eines CAD-Modells erstellen zu müssen, wird ein Verfahren benötigt, das eine Umgebungskarte aus den Sensordaten eines Explorationslaufs des Roboters erstellt. Diese Problematik der Kartenerstellung war Gegenstand des 5. Kapitels. Die Kartenerstellung erfordert eine rekursive Lösung, da für das Eintragen neuer Sensorinformation in eine bestehende Karte die genaue Roboterposition benötigt wird. Um diese aber zu bestimmen, wird wiederum eine Umgebungskarte benötigt.

Einfache Verfahren, welche die Karte nur lokal aktualisieren scheitern, wenn die zu kartierende Umgebung große Zyklen enthält. Um mit Zyklen umgehen zu können, muß ein Verfahren alle besuchten Positionen entlang des Pfades in Betracht ziehen. Der Ansatz der *Stochastic Map* [Smith *et al.*, 1990] und die *Konsistente Positionsschätzung* [Lu und Milios, 1997a] sind solche Verfahren.

Zunächst wurde die Methode der konsistenten Positionsschätzung vorgestellt. Diese betrachtet einen Satz von durch Odometrie bestimmten Positionen, an denen jeweils ein Laserscan aufgenommen wurde. Beziehungen zwischen Positionen werden durch Odometriemessungen und durch Scan-Matching bestimmt. Alle Positionen werden so geschätzt, daß eine Fehlersumme minimiert wird. Dadurch entsteht eine konsistente Karte der Umgebung. In mehreren Experimenten wurde gezeigt, daß diese Methode sehr genaue Karten liefert. Ein Nachteil ist jedoch, daß die Rohdaten bereits topologisch korrekt sein müssen.

Ein anderer Ansatz, welcher eine Erwartungs- und Maximierungsstrategie benutzt [Thrun *et al.*, 1998b], wurde anschließend präsentiert. Diese Methode ist in der Lage, aus einem Satz von rohen Sensordaten, eine topologisch korrekte Karte zu erstellen, die aber nicht so genau wie die durch konsistente Positionsschätzung berechnete Karte ist.

In dieser Arbeit wurde eine Kombination der Arbeiten von Thrun und Lu und Milios vorgestellt, welche zu einem sehr leistungsstarken Verfahren führt, das in der Lage ist, hochgenaue Karten in großen, zyklischen Umgebungen aus Laserscannerdaten zu erstellen [Thrun *et al.*, 1998d].

Leider ist dieses Verfahren sehr rechenintensiv, die Daten müssen erst gesammelt werden, bevor mit der Auswertung begonnen werden kann, und das Verfahren erfordert zusätzliche Informationen eines Benutzers, um dem Roboter mitzuteilen, wann er eine Landmarke erreicht hat. Wünschenswert wäre ein Verfahren, das autonom, in Echtzeit und ohne zusätzliche Informationen eines Benutzers eine genaue Karte der Umgebung inkrementell erstellt.

Ein solches Verfahren wurde in diesem Kapitel entwickelt. Es benutzt Scan-Matching, Kartenkorrelation und die Methode der konsistenten Positionsschätzung, um hochgenaue Karten aus einem Strom von Odometrie- und Laserscannerdaten zu generieren. Scan-Matching wird benutzt, um lokale Kartenstücke zusammenzusetzen und um die Genauigkeit der Gesamtkarte zu verbessern. Kartenkorrelation wird verwendet, um topologische Zusammenhänge zu erkennen, die für das Schließen von Zyklen wichtig sind. Schließlich wird die Methode der konsistenten Positionsschätzung dazu eingesetzt, lokale Kartenstücke zusammenzusetzen und um erkannte Zyklen zu schließen. Zahlreiche Experimente in verschiedenen Umgebungen und mit verschiedenen Robotern wurden präsentiert und zeigen die Leistungsfähigkeit dieser Methode [Gutmann und Konolige, 1999]. Nach Wissen des Autors ist dies das erste inkrementelle Verfahren, das in der Lage ist, autonom und in Echtzeit hochgenaue Karten in großen, zyklischen Umgebungen zu erstellen.

In Kapitel 6 wurde das Problem der Pfadplanung genauer untersucht. Gegeben ist eine Umgebungskarte, sowie eine Start- und eine Zielposition. Gesucht ist ein Pfad, der den Roboter kollisionsfrei von der Start- zur Zielposition führt. Zunächst wurden gitterbasierte Methoden, welche die Umwelt in Zellen gleicher Größe einteilen, genauer analysiert. Diese sind in der Lage, kürzeste Wege zu finden, Sackgassen zu erkennen, und mit ungenauen Informationen umzugehen. Jedoch hängt die Zustandsraumgröße und der Rechenbedarf direkt von der Größe der Umgebung ab, sodaß diese Methode nur in relativ kleinen Umgebungen für eine Echtzeitanwendung sinnvoll ist. Aus diesem Grund wurde ein zweistufiges Verfahren vorgeschlagen, das aus einem globalen und einem lokalen Planer besteht [Gutmann und Nebel, 1997].

Der globale Planer erstellt eine Wegekarte, welche durch Auswertung der *Sichtbarkeit* zwischen Scans bestimmt wird. Je mehr gemeinsame Scanpunkte zwei Scans besitzen, desto größer ist die Wahrscheinlichkeit, daß es einen kollisionsfreien Pfad zwischen den beiden Scanaufnahmepositionen gibt. Ein A^*-Algorithmus berechnet dann Wege von Start- zu Zielknoten auf diesem Graphen. Für die Bestimmung der tatsächlichen Pfade zu den Zwischenpositionen wird ein lokales, gitterbasiertes Verfahren eingesetzt. Dieses Verfahren kann Hindernissen dynamisch ausweichen und erkennt, wenn eine anzufahrende Position nicht erreichbar ist. In einem Beispielexperiment in der Umgebung der KI-Abteilung wurde die erfolgreiche Anwendung des Verfahrens gezeigt. Zuletzt wurde auf verschiedene

Nachteile des Verfahrens hingewiesen und ein Ausblick mit einer neue Methode präsentiert, welche versucht, eine „natürliche“ Aufteilung der Umgebung in Räume und Gänge zu berechnen.

Im letzten größeren Kapitel dieser Arbeit wurde Roboterfußball als eine Anwendung der zuvor beschriebenen Navigationsmethoden präsentiert. Es wurde ein klassischer Ansatz der Künstlichen Intelligenz gewählt, der explizite Weltmodellierung und Deliberation vorsieht [Gutmann *et al.*, 1999a]. Die verwendeten Roboter vom Typ *Pioneer-I* wurden dabei jeweils mit einem Laserscanner, einer Videokamera und einem Mini-Notebook mit Funk-Ethernet-Verbindung ausgestattet. Zusätzlich wurde eine selbstgebaute Schußvorrichtung mit Ballsteuerung entwickelt.

Jeder Roboter handelt als selbständiger Agent und ist in eine Team-Architektur integriert, welche die einzelnen Roboter per Funk mit einem Rechner außerhalb des Spielfeldes verbindet. Die Spielerarchitektur selbst ist in die drei Module Wahrnehmung, verhaltensbasierte Steuerung und Pfadplanung unterteilt.

Das Wahrnehmungsmodul sorgt für die Bereitstellung eines lokalen Weltmodells, aufgrund dessen Zustände Aktionen abgeleitet werden können. Grundlage dieses Moduls ist das Wissen über die genaue Roboterposition. Hierzu wird eine Selbstlokalisierungsmethode benutzt, welche ein neues Scan-Matching-Verfahren namens *LineMatch* verwendet. Durch einen Vergleich mit anderen Scanüberdeckungsmethoden konnte experimentell gezeigt werden, daß *LineMatch* wesentlich schneller und robuster als die konkurrierenden Methoden ist, ohne dabei an Genauigkeit gegenüber den anderen Verfahren einzubüßen [Gutmann *et al.*, 1999b].

Wurde ein Scan erfolgreich überdeckt, so können anschließend andere Spieler erkannt werden und zusammen mit der Ballposition, die über eine Videokamera bestimmt wird, in einem lokalen Weltmodell verwaltet werden. Auf der Grundlage dieses Weltmodells können durch relativ einfache geometrische Verfahren kollisionsfreie Pfade geplant werden. Das verhaltensbasierte Steuerungsmodul wertet Situationen im Weltmodell und über Funk erhaltene Nachrichten der anderen Spieler aus und entscheidet so, welche Aktion als nächste ausgeführt werden soll. Eine Kooperation der Spieler wurde durch die Definition von Kompetenzbereichen und den Einsatz von Kommunikation möglich. Das Kapitel schloß mit Erfahrungen bei Turnieren und den erzielten Spielergebnissen. Das CS-Freiburg Team errang den Titel des Deutschen Meisters 1998 und 1999, den Weltmeistertitel 1998 und erreichte den 3. Platz bei der Weltmeisterschaft 1999.

8.2 Ausblick

Bei allen in dieser Arbeit vorgestellten Bereichen der Navigation von autonomen mobilen Systemen sind zahlreiche Erweiterungen möglich. Das Gebiet der Selbstlokalisierung ist bereits sehr gut erforscht. Dennoch sind weitere Arbeiten denkbar, von denen neue vielversprechende Ergebnisse zu erwarten sind. Die vorgeschlagene Kombination der beiden Selbstlokalisierungsverfahren sollte tatsächlich realisiert werden, um experimentelle Evidenz der Kombination der Vorteile beider Verfahren zu erhalten. Weiterhin können Untersuchungen für die Selbstlokalisierung einer Gruppe von Robotern unternommen worden. Dieses Gebiet der Multi-Roboter-Lokalisierung ist ein noch sehr junges Forschungsgebiet.

Im Bereich der Kartenerstellung kann das vorgeschlagene, inkrementelle Verfahren an zahlreichen Stellen erweitert werden. Zum einen kann die Erkennung von topologischen Zusammenhängen verbessert werden, indem weitere Sensoren, wie z.B. Videokameras hinzugezogen werden. Zum anderen kann ein Mechanismus eingebaut werden, welcher nachträglich erkennt, daß ein Zyklus an der falschen Stelle geschlossen wurde. Durch Verwalten mehrerer Hypothesen der Stellen, an denen das Schließen eines Zyklus möglich ist, kann dann die richtige Kombination von topologischen Zusammenhängen gefunden werden. Weiterhin kann das Verfahren für den kooperativen Einsatz mehrerer Roboter weiterentwickelt werden [Konolige *et al.*, 1999]. Die Multi-Roboter-Kartenerstellung ist ebenfalls noch ein sehr junges Forschungsgebiet.

Der Bereich der Pfadplanung wurde in dieser Arbeit weniger intensiv wie die beiden anderen Navigationsthemen untersucht. Das zweistufige Verfahren mit globalem und lokalem Planer hat sich in der Praxis bei vielen Robotersystemen bewährt. Auf dem Gebiet des globalen Planens sind jedoch noch weitere Untersuchungen möglich. Die in Kapitel 6 angesprochene Aufteilung in Räume und Gänge ist eine sehr vielversprechende Methode, topologische Karten und damit Suchgraphen für die globale Planung zu erstellen.

Im Bereich Roboterfußball sind ebenfalls noch viele Verbesserungen möglich. Zum einen können die Spieler weiterentwickelt werden. Neben der Hardware, die durch einen stärkeren Schußmechanismus und schnellere Motoren aufgerüstet werden könnte, sind Verbesserungen an den Basisverhalten denkbar, die ein flüssigeres und schnelleres Spiel erlauben. Leider beschränkt hier die zur Verfügung stehende Plattform mögliche Leistungssteigerungen.

Zum anderen kann das Spielen im Team verbessert werden, indem z.B. die Roboter dynamisch aufgrund der aktuellen Situation im Weltmodell positioniert werden. So könnte z.B. das Freilaufen eines Spielers oder Offensiv- bzw. Defensiv-Spiel realisiert werden. Für die Teilnahme an RoboCup'99 wurden hier bereits Arbeiten verwirklicht, welche die Heimatposition der Spieler dynamisch aufgrund der aktuellen Ballposition festlegen. Weiterhin wäre die Realisierung eines Paßspiels eine Herausforderung an Wahrnehmung, Kooperation und reaktive Basisverhalten [Nebel *et al.*, 1999]. Für RoboCup'99 wurde ein solches Paßspiel bereits entwickelt und im Viertelfinalspiel gegen das Team aus Italien gab es eine Situation, in der ein Spieler einem anderen Spieler im gegnerischen Strafraum den Ball zupaßte. Leider konnte der annehmende Spieler den Ball aber nicht ins Tor befördern. Auf einem weiteren Turnier Ende September in Weimar im Finalspiel gegen Ulm war es dann aber soweit: Das erste Tor durch Paßspiel!

Anhang A

ScanStudio

Parallel zu den in dieser Arbeit beschrieben Verfahren wurde eine leistungsfähige Entwicklungsumgebung für die Verarbeitung von Scannerdaten entwickelt. Mit *ScanStudio* kann ein Roboter sich lokalisieren, Karten erstellen und kollisionsfreie Pfade zu Zielpositionen berechnen. Die meisten in dieser Arbeiten beschrieben Verfahren sowie weitere Algorithmen wurden mittels *ScanStudio* entwickelt und sind in diesem Software-Paket integriert.

Die Selbstlokalisierung kann durch eine der in *ScanStudio* vorhandenen Scan-Matching-Methoden realisiert werden. Zur Verfügung stehen eine Erweiterung der Cox-Methode [Cox, 1991], der IDC-Algorithmus [Lu und Milios, 1997b], eine Erweiterung der Kreuzkorrelationsmethode [Weiß und Puttkamer, 1995], das kombinierte Scan-Matching-Verfahren [Gutmann und Schlegel, 1996], sowie Scan-Matching durch Kartenkorrelation [Konolige und Chou, 1999]. Weiterhin enthält *ScanStudio* Verfahren zur globalen Positionsbestimmung mittels Markov-Lokalisierung und Kartenkorrelation.

Das LRGC-Verfahren zur Kartenerstellung in großen, zyklischen Umgebungen ist ebenfalls in *ScanStudio* enthalten. Durch einen interaktiven Modus kann der Benutzer außerdem die Kartenerstellung kontrollieren und z.B. das Schließen von falsch detektierten Zyklen vermeiden. Abbildung A.1 zeigt eine Bildschirmabbildung des *ScanStudio*-Simulators vor Schließen eines erkannten Zyklusses. Zusätzlich steht eine Implementierung der Methode der konsistenten Positionsschätzung zur Verfügung.

Weiterhin ist das in dieser Arbeit beschriebene Pfadplanungsverfahren in der Softwareumgebung enthalten und eine erste Implementierung des Verfahrens mit „natürlicher“ Raumaufteilung ist vorhanden.

Die *ScanStudio*-Software wurde seit Ende 1995 entwickelt und besteht aus drei Teilen, einer Funktionsbibliothek, die alle implementierten Verfahren enthält, einem Satz von Kommandozeilenbefehlen, mit denen auf Platte gespeicherte Scans verarbeitet werden können, und einer graphischen Benutzeroberfläche zur Visualisierung von Scandaten, Debuginformationen und Ergebnissen der verschiedenen Algorithmen.

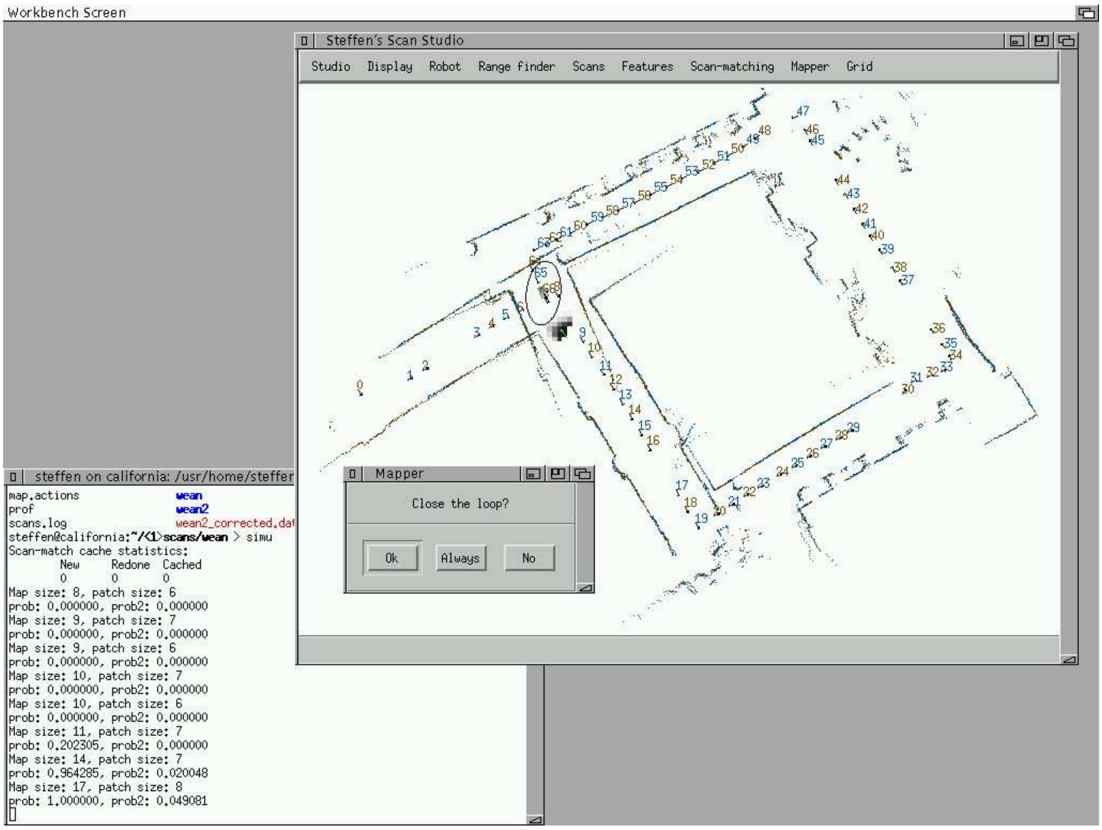


Abbildung A.1: Bildschirmabbildung der entwickelten Softwareumgebung.

Alle Teile wurden in ANSI-C implementiert, wobei teilweise auf Erweiterungen des gcc-Compilers zurückgegriffen wurde. Die graphische Benutzeroberfläche wurde unter Unix mit dem X-Window-System und Motif erstellt, ist aber relativ einfach auf andere Systeme portierbar.

ScanStudio darf nur mit der Erlaubnis des Authors eingesetzt werden. Folgende Institute und Personen sind im Besitz einer solchen Lizens:

- Universität Freiburg, Lehrstuhl für Grundlagen der Künstlichen Intelligenz. Prof. Dr. Bernhard Nebel.
- FAW Ulm, Arbeitsgruppe des SFB 527, Projekt C3. Christian Schlegel.
- Universität Bonn, Institiut für Informatik. Prof. Dr. Wolfram Burgard und Dr. Dieter Fox.
- Carnegie Mellon University, Pittsburgh, USA. Prof. Dr. Sebastian Thrun.
- SRI Int., Menlo Park, USA. Dr. Kurt Konolige.
- Institut für Automatisierungstechnik IAT, Universität Bremen. Kai Steffen.

- Autonomous Systems Lab, Ecole Polytechnique Federale de Lausanne EPFL, Schweiz. Prof. Dr. Roland Siegwart und Kai Arras.
- „Robots that kill“, Palo Alto, USA. Andy Rubin.
- Stanford University, USA. Prof. Dr. J.C. Latombe.
- University of Texas at Austin, USA. Prof. Dr. Ben Kuipers und Alisa Marzilli.
- Departement of Computer Science, University of Manchester M13 9PL. Stephen Marsland.
- Fachhochschule Konstanz. Prof. Dr. Bittel.

Weitere Lizensen sind auf Anfrage beim Autor, Steffen Gutmann, erhältlich.

Abbildungsverzeichnis

Literaturverzeichnis

[Arkin, 1990] R.C. Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6:105–122, 1990.

[Arras *et al.*, 1996] Kai O. Arras, Sjur Vestli, und Nadine Tschichold-Gürman. Echtzeitfähige Merkmalsextraktion und Situationsinterpretation aus Laserscannerdaten. In *Autonome Mobile Systeme (AMS'96)*, Seiten 57–66, München, 1996. Springer-Verlag.

[Arras und Siegwart, 1997] Kai O. Arras und Roland Y. Siegwart. Feature extraction and scene interpretation for map-based navigation and map-building. In *Proc. SPIE, Mobile Robotics XII*, volume 3210, Seiten 42–53, 1997.

[Asada *et al.*, 1998] Minoru Asada, Peter Stone, Hiroaki Kitano, Alexis Drogoul, Dominique Duhaut, Manuela Veloso, Hajime Asama, und Sho'ji Suzuki. The RoboCup physical agent challenge: Goals and protocols for phase I. In Kitano [1998], Seiten 42–61.

[Asada und Kitano, 1999] M. Asada und H. Kitano, editors. *RoboCup-98: Robot Soccer World Cup II.* Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, Heidelberg, New York, 1999.

[Ayache und Faugeras, 1989] N. Ayache und O.D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Transactions on Robotics and Automation*, 5(6):804–819, 1989.

[Bengtsson und Baerveldt, 1999] Ola Bengtsson und Albert-Jan Baerveldt. Localization in changing environments by matching laser range scans. In *Proc. 3rd European Workshop on Advanced Mobile Robots (EUROBOT'99)*, Zürich, September 1999.

[Borenstein, 1994] J. Borenstein. Internal correction of dead-reckoning errors with the smart encoder trailer. In *Proc. International Conference on Intelligent Robots and Systems (IROS'94)*, Seiten 127–134, September 1994.

[Borthwick und Durrant-Whyte, 1994] S. Borthwick und H. Durrant-Whyte. Simultaneous localisation and map building for autonomous guided vehicles. In *Proc. International Conference on Intelligent Robots and Systems (IROS'94)*, Seiten 761–768, September 1994.

[Braun und Corsépius, 1996] Bernhard Braun und Ralf Corsépius. AMOS: Schnelle Manipulator-Bewegungsplanung durch Integration potentialfeldbasierter lokaler und probabilistischer globaler Algorithmen. In *Autonome Mobile Systeme (AMS'96)*, München, 1996. Springer-Verlag.

[Braun, 1995] Bernhard Braun. Vergleich von Algorithmen zur Bahnplanung eines Manipulators. Diplomarbeit, Universität Ulm, 1995.

[Brooks, 1986] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics & Automation*, RA-2(1), 1986.

[Buhmann *et al.*, 1995] J. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, und S. Thrun. The mobile robot RHINO. *AI Magazine*, 16(2):31–38, Summer 1995.

[Burgard *et al.*, 1996] Wolfram Burgard, Dieter Fox, Daniel Hennig, und Timo Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proc. 14th National Conference on Artificial Intelligence (AAAI'96)*, Seiten 896–901, August 1996.

[Burgard *et al.*, 1997] Wolfram Burgard, Dieter Fox, und Sebastian Thrun. Active mobile robot localization. In *Proc. 15th International Conference on Artificial Intelligence (IJCAI'97)*, 1997.

[Burgard *et al.*, 1998a] W. Burgard, A.B. Cremers, D. Fox, G. Lakemeyer, D. Hähnel, D. Schulz, W. Steiner, und S. Thrun. The interactive museum tour-guide robot. In *Proc. 15th National Conference on Artificial Intelligence (AAAI'98)*, 1998.

[Burgard *et al.*, 1998b] W. Burgard, A. Derr, D. Fox, und A.B. Cremers. Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach. In *Proc. International Conference on Intelligent Robots and Systems (IROS 98)*, Victoria, Oktober 1998.

[Burgard *et al.*, 1999] W. Burgard, D. Fox, H. Jans, C. Matenar, und S. Thrun. Sonar-based mapping with mobile robots using EM. In *Proc. of the International Conference on Machine Learning (ICML'99)*, 1999.

[Burkhard *et al.*, 1998] Hans-Dieter Burkhard, Markus Hannebauer, und Jan Wendler. AT Humbold – development, practice and theory. In Kitano [1998], Seiten 357–372.

[Castellanos *et al.*, 1996a] José A. Castellanos, J. Neira, O. Strauss, und Juan D. Tardós. Detecting high level features for mobile robot localization. In *Proc. International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Seiten 611–618, Washington, Dezember 1996.

[Castellanos *et al.*, 1996b] José A. Castellanos, Juan D. Tardós, und José Neira. Constraint-based mobile robot localization. In *International Workshop on Advanced Robotics and Intelligent Machines*. University of Salford, April 1996.

[Castellanos *et al.*, 1997] José A. Castellanos, Juan D. Tardós, und G. Schmidt. Building a global map of the environment of a mobile robot: The importance of correlations. In *Proc. International Conference on Robotics and Automation (ICRA'97)*, Seiten 1053–1059, April 1997.

[Castellanos *et al.*, 1999] José A. Castellanos, J. M. M. Montiel, José Neira, und Juan D. Tardós. The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transaction on Robotics and Automation*, 15(5):948–952, Oktober 1999.

[Castellanos und Tardós, 1996] José A. Castellanos und Juan D. Tardós. Laser-based segmentation and localization for a mobile robot. In *Proc. Sixth International Symposium on Robotics and Manufacturing (ISRAM'96)*, Seiten 101–108, Montpellier, France, Mai 1996.

[Chatila und Laumond, 1985] R. Chatila und J.P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proc. International Conference on Robotics and Automation (ICRA'85)*, März 1985.

[Connell, 1990] J. Connell. *Minimalistic Mobile Robotics: A Colony-style Architecture for an Artificial Creature*. Academic Press, 1990.

[Corsépius *et al.*, 1997] Ralf Corsépius, Jörg Illmann, und Christian Schlegel. Abschlußbericht des Verbundprojektes AMOS. Technical report, Forschungsinstitut für anwendungsorientierte Wissensverarbeitung (FAW), 1997.

[Cox und Wilfong, 1990] I.J. Cox und G.T. Wilfong, editors. *Autonomous Robot Vehicles*. Springer-Verlag, 1990.

[Cox, 1990] Ingemar J. Cox. Blanche: Position estimation for an autonomous robot vehilce. In Cox und Wilfong [1990], Seiten 221–228.

[Cox, 1991] Ingemar J. Cox. Blanche—an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, April 1991.

[Crowley, 1989] J.L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proc. International Conference on Robotics and Automation (ICRA'89)*, Seiten 674–680, 1989.

[Dellaert *et al.*, 1999] F. Dellaert, D. Fox, W. Burgard, und S. Thrun. Monte carlo localization for mobile robots. In *Proc. International Conference on Robotics and Automation (ICRA'99)*, 1999.

[Dempster *et al.*, 1977] A. Dempster, A. Laird, und D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[Dietl, 1999] Markus Dietl. Globale Sensorintegration im Roboterfußball. Studienarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

[Durrant-Whyte, 1987] H.F. Durrant-Whyte. Consistent integration and propagation of disparate sensor observations. *International Journal of Robotics Research*, 6(3):3–24, 1987.

[Durrant-Whyte, 1988a] H.F. Durrant-Whyte. *Integration, Coordination and Control of Multisensor Robot Systems*. Kluwer Academic Publishers, Boston, Mass., 1988.

[Durrant-Whyte, 1988b] H.F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, 4(1):23–31, 1988.

[Edlinger und Weiß, 1995] T. Edlinger und G. Weiß. Exploration, navigation and self-localization in an autonomous mobile robot. In *Autonome Mobile Systeme (AMS'95)*, Seiten 142–151. Springer-Verlag, 1995.

[Feng *et al.*, 1996] L. Feng, J. Borenstein, und H.R. Everett. “Where am I?” Sensors and methods for autonomous mobile robot positioning, 1996.

[Fox *et al.*, 1998] D. Fox, W. Burgard, S. Thrun, und A.B. Cremers. Position estimation for mobile robots in dynamic environments. In *Proc. 15th National Conference on Artificial Intelligence (AAAI'98)*, 1998.

[Fox *et al.*, 1999] D. Fox, W. Burgard, F. Dellaert, und S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. National Conference on Artificial Intelligence (AAAI'99)*, 1999.

[Fox, 1998] Dieter Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation.* PhD thesis, Institute of Computer Science III, University of Bonn, Germany, 1998.

[Gonzalez *et al.*, 1992] J. Gonzalez, A. Stentz, und A. Ollero. An iconic position estimator for a 2d laser range finder. In *Proc. International Conference on Robotics and Automation (ICRA'92)*, Seiten 2646–2651, 1992.

[Gonzalez *et al.*, 1994] J. Gonzalez, A. Reina, und A. Ollero. Map building for a robot equipped with a 2D laser rangefinder. In *Proc. International Conference on Robotics and Automation (ICRA'94)*, 1994.

[Guibas *et al.*, 1995] L.J. Guibas, R. Motwani, und P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, und R. Wilson, editors, *Algorithmic Foundations of Robotics*, Seiten 269–282. A.K. Peters (Boston), 1995.

[Gutmann *et al.*, 1998a] Jens-Steffen Gutmann, Wolfram Burgard, Dieter Fox, und Kurt Konolige. An experimental comparison of localization methods. In *Proc. International Conference on Intelligent Robots and Systems (IROS'98)*, Victoria, Oktober 1998.

[Gutmann *et al.*, 1998b] Jens-Steffen Gutmann, Bernhard Nebel, Wolfgang Hatzack, Immanuel Herrmann, Frank Rittinger, Augustinus Topor, Thilo Weigel, und Bruno Welsch. The CS Freiburg team. In M. Asada, editor, *RoboCup-98: Robot Soccer World Cup II. Proceedings of the second RoboCup Workshop*, Paris, France, July 1998.

[Gutmann *et al.*, 1999a] Jens-Steffen Gutmann, Wolfgang Hatzack, Immanuel Herrmann, Bernhard Nebel, Frank Rittinger, Augustinus Topor, Thilo Weigel, und Bruno Welsch. The CS Freiburg team: Reliable self-localization, multirobot sensor integration, and basic soccer skills. In Asada und Kitano [1999].

[Gutmann *et al.*, 1999b] Jens-Steffen Gutmann, Thilo Weigel, und Bernhard Nebel. Fast, accurate, and robust self-localization in polygonal environments. In *Proc. International Conference on Intelligent Robots and Systems (IROS'99)*, Kyongju, Oktober 1999.

[Gutmann *et al.*, 2000] Jens-Steffen Gutmann, Wolfgang Hatzack, Immanuel Herrmann, Bernhard Nebel, Frank Rittinger, Augustinus Topor, und Thilo Weigel. The CS Freiburg team: Playing robotic soccer on an explicit world model. *AI Magazine*, 21(1):37–46, 2000.

[Gutmann und Konolige, 1999] Jens-Steffen Gutmann und Kurt Konolige. Incremental mapping of large cyclic environments. In *CIRA'99*, November 1999.

[Gutmann und Nebel, 1997] Jens-Steffen Gutmann und Bernhard Nebel. Navigation mobiler Roboter mit Laserscans. In *Autonome Mobile Systeme (AMS'97)*, Stuttgart, Oktober 1997. Springer-Verlag.

[Gutmann und Schlegel, 1996] Jens-Steffen Gutmann und Christian Schlegel. Amos: Comparison of scan matching approaches for self-localization in indoor environments. In *Proc. 1st Euromicro Workshop on Advanced Mobile Robots (EUROBOT'96)*, Seiten 61–67, 1996.

[Gutmann, 1996] Jens-Steffen Gutmann. Vergleich von Algorithmen zur Selbstlokalisierung eines mobilen Roboters. Diplomarbeit, Universität Ulm, 1996.

[Hébert *et al.*, 1995] P. Hébert, S. Betgé-Brezetz, und R. Chatila. Probabilistic map learning: Necessity and difficulties. In *Proc. International Workshop on Reasoning with Uncertainty in Robotics*, Amsterdam, 1995.

[Herrmann, 1998] Immanuel Herrmann. Roboter-Schußvorrichtung. Memo, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1998.

[Hähnel *et al.*, 1998] D. Hähnel, W. Burgard, und G. Lakemeyer. GOLEX – bridging the gap between logic (GOLOG) and a real robot. In *Proc. of the 22nd German Conference on Artificial Intelligence (KI'98)*, Bremen, Germany, 1998.

[Holenstein *et al.*, 1992] A. Holenstein, M. Muller, und E. Badreddin. Mobile robot localization in a structured environment cluttered with obstacles. In *Proc. International Conference on Robotics and Automation (ICRA'92)*, Seiten 2576–2581, 1992.

[Horn und Schmidt, 1995] J. Horn und G.K. Schmidt. Continuous localization for long-range indoor navigation of mobile robots. In *Proc. International Conference on Robotics and Automation (ICRA'95)*, Seiten 387–394, 1995.

[Kaelbling *et al.*, 1996] L. Kaelbling, A. Cassandra, und J. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proc. International Conference on Intelligent Robots and Systems (IROS'96)*, 1996.

[Karch *et al.*, 1997] Oliver Karch, Hartmut Noltemeier, Mathias Schwark, und Thomas Wahl. Relokalisation – Ein theoretischer Ansatz in der Praxis. In *Autonome Mobile Systeme (AMS'97)*, Seiten 119–130. Springer-Verlag, Stuttgart, Oktober 1997.

[Karch und Noltemeier, 1996] Oliver Karch und Hartmut Noltemeier. Zum Lokalisationsproblem für Roboter. In *Autonome Mobile Systeme (AMS'96)*, Seiten 128–137. Springer-Verlag, München, 1996.

[Kitano *et al.*, 1997] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, und Hitoshi Matsubara. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, 1997.

[Kitano *et al.*, 1998] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, und Hitoshi Matsubara. RoboCup: A challenge problem for AI and robotics. In Kitano [1998], Seiten 1–19.

[Kitano, 1998] H. Kitano, editor. *RoboCup-97: Robot Soccer World Cup I*, volume 1395 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York, 1998.

[Klupsch *et al.*, 1999] Michael Klupsch, Thorsten Bandlow, Marc Grimme, Ignaz Klerrer, Maximilian Lückenhaus, Fabian Schwarzer, und Christoph Zierl. Agilo RoboCuppers: RoboCup team description. In Asada und Kitano [1999].

[Knick und Schlegel, 1994] Manfred Knick und Christian Schlegel. AMOS: Active perception of an autonomous system. In *Proc. International Conference on Intelligent Robots and Systems (IROS'94)*, Seiten 281–289, September 1994.

[Knoll, 1994] A. Knoll. Sensordatenfusion für Anwendungen in der Robotik, 1994. Manuskript zum Spezialkurs, KI-Frühjahrsschule der GI.

[Konolige *et al.*, 1997] Kurt Konolige, Karen Myers, Enrique H. Ruspini, und Alessandro Saffiotti. The Saphira Architecture: A Design for Autonomy. *Journal of Experimental and Theoretical Artificial Intelligence*, 9:215–235, 1997.

[Konolige *et al.*, 1999] Kurt Konolige, Jens-Steffen Gutmann, Didier Guzzoni, Robert Ficklin, und Keith Nicewarner. A mobile robot sense net. In *SPIE Photonics East*, Boston, September 1999.

[Konolige und Chou, 1999] Kurt Konolige und Ken Chou. Markov localization using correlation. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI'99)*, Stockholm, 1999.

[Kunz *et al.*, 1997] Clayton Kunz, Thomas Willeke, und Illah R. Nourbakhsh. Automatic mapping of dynamic office environments. In *Proc. International Conference on Robotics and Automation (ICRA'97)*, Seiten 1681–1687, April 1997.

[Kwon und Lee, 1997] Young D. Kwon und Jin S. Lee. A stochastic environment modelling method for mobile robot by using 2-D laser scanner. In *Proc. International Conference on Robotics and Automation (ICRA'97)*, Seiten 1688–1693, April 1997.

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

[Leonard *et al.*, 1990] J. Leonard, H. Durrant-Whyte, und I.J. Cox. Dynamic map building for an autonomous mobile robot. In *Proc. International Conference on Intelligent Robots and Systems (IROS'90)*, Seiten 89–95, 1990.

[Leonard und Durrant-Whyte, 1991] J.J. Leonard und H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transaction on Robotics and Automation*, 7(3):376–382, 1991.

[Lu und Milios, 1994a] F. Lu und E.E. Milios. An iterative algorithm for shape registration. In *2nd Int. Workshop on Visual Form*, Seiten 344–353, 1994.

[Lu und Milios, 1994b] F. Lu und E.E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *IEEE Computer Vision and Pattern Recognition Conference (CVPR'94)*, Seiten 935–938, 1994.

[Lu und Milios, 1994c] F. Lu und E.E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. Technical Report RBCV-TR-94-46, University of Toronto, 1994.

[Lu und Milios, 1995] F. Lu und E.E. Milios. Optimal global pose estimation for consistent sensor data registration. In *Proc. International Conference on Robotics and Automation (ICRA'95)*, 1995.

[Lu und Milios, 1997a] Feng Lu und E.E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

[Lu und Milios, 1997b] Feng Lu und E.E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18:249–275, 1997.

[Lu, 1995] Feng Lu. *Shape Registration using Optimization for Mobile Robot Navigation.* PhD thesis, University of Toronto, 1995.

[Maybeck, 1990] Peter S. Maybeck. The Kalman filter: An introduction to concepts. In Cox und Wilfong [1990], Seiten 194–204.

[Moravec und Elfes, 1985] H.P. Moravec und A. Elfes. High resolution maps from wide-angle sonar. In *Proc. International Conference on Robotics and Automation (ICRA'85)*, März 1985.

[Nardi *et al.*, 1999] D. Nardi, G. Clemente, und E. Pagello. Azzurra robot team. In Asada und Kitano [1999].

[Nebel *et al.*, 1999] Bernhard Nebel, Jens-Steffen Gutmann, und Wolfgang Hatzack. The CS Freiburg'99 team. In *3rd International Workshop on RoboCup*, 1999.

[Noda *et al.*, 1998] Itsuki Noda, Shoji Suzuki, Hitoshi Matsubara, Minoru Asada, und Hiroaki Kitano. Overview of RoboCup-97. In Kitano [1998], Seiten 20–41.

[Nourbakhsh *et al.*, 1995] I. Nourbakhsh, R. Powers, und S. Birchfield. DERVISH an office-navigating robot. *AI Magazine*, 16(2):53–60, Summer 1995.

[Prassler und Milios, 1995] E.A. Prassler und E.E. Milios. Position estimation using equidistance lines. In *Proc. International Conference on Robotics and Automation (ICRA'95)*, Seiten 85–92, 1995.

[Reetz, 1999] Christian Reetz. Aktionsauswahl in dynamischen Umgebungen am Beispiel Roboterfußball. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

[Rencken, 1993] W.D. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proc. International Conference on Intelligent Robots and Systems (IROS'93)*, Seiten 2192–2197, 1993.

[Röfer, 1995] T. Röfer. Navigation mit eindimensionalen 360°-Bildern. In *Autonome Mobile Systeme (AMS'95)*, Seiten 193–202. Springer-Verlag, 1995.

[Schiele und Crowley, 1994] Bernt Schiele und James L. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proc. International Conference on Robotics and Automation (ICRA'94)*, Seiten 1628–1634, 1994.

[Schlegel und Wörz, 1998] Christian Schlegel und Robert Wörz. Der Softwarerahmen SmartSoft zur Implementierung sensomotorischer Systeme. In *Autonome Mobile Systeme (AMS'98)*, Seiten 208–217, Karlsruhe, November 1998. Springer-Verlag.

[Schlegel, 1998] Christian Schlegel. Fast local obstacle avoidance under kinematic and dynamic constraints for a mobile robot. In *Proc. International Conference on Intelligent Robots and Systems (IROS'98)*, Seiten 594–599, Victoria, Oktober 1998.

[Schultz und Adams, 1996] A. C. Schultz und W. Adams. Continuous localization using evidence grids. Technical Report AIC-96-007, Naval Center for Applied Research in Artificial Intelligence, 1996.

[Shaffer *et al.*, 1992] Gary Shaffer, Javier Gonzalez, und Anthony Stentz. Comparison of two range-based estimators for a mobile robot. In *SPIE Conf. on Mobile Robots VII*, volume 1831, Seiten 661–667, 1992.

[Shaffer *et al*, 1992] G. Shaffer *et al.* Position estimator for underground mine equipment. In *IEEE Transactions on Industry Applications*, volume 28, September 1992.

[Simmons und Koenig, 1995] R. Simmons und S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI'95)*, 1995.

[Skalicky, 1996] Tomas Skalicky. LASPack – package for solving large sparse systems of linear equations., 1996. Available in http://www.tu-dresden.de/mwism/skalicky/laspack/laspack.html.

[Smith *et al.*, 1990] R. Smith, M. Self, und P. Cheeseman. Estimating uncertain spatial relationships in robotics. In Cox und Wilfong [1990], Seiten 167–193.

[Thiel, 1999] Maximiliam Thiel. Roboter-Fußball: Zuverlässige Ballerkennung und Positionsschätzung. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

[Thrun *et al.*, 1998a] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, und T. Schimdt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, und R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, Cambridge, MA, 1998.

[Thrun *et al.*, 1998b] S. Thrun, D. Fox, und W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998. Also appeared in Autonomous Robots 5:253-271.

[Thrun *et al.*, 1998c] Sebastian Thrun, Dieter Fox, und Wolfram Burgard. Probabilistic mapping of an environment by a mobile robot. In *Proc. International Conference on Robotics and Automation (ICRA'98)*, 1998.

[Thrun *et al.*, 1998d] Sebastian Thrun, Jens-Steffen Gutmann, Dieter Fox, Wolfram Burgard, und Benjamin Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *Proc. 15th National Conference on Artificial Intelligence (AAAI'98)*, Madison, WI, 1998.

[Thrun *et al.*, 1999] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, und D. Schulz. MINERVA: A second generation mobile tour-guide robot. In *Proc. International Conference on Robotics and Automation (ICRA'99)*, 1999.

[Thrun und Bücken, 1996] Sebastian Thrun und Arno Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proc. 14th National Conference on Artificial Intelligence (AAAI'96)*, August 1996.

[Thrun, 1998] Sebastian Thrun. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 1998.

[Topor, 1998] A. Topor. Die Ballerkennung und Positionsschätzung des CS-Freiburg-Teams bei RoboCup'98. Studienarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1998.

[Topor, 1999] Augustinus Topor. Pfadplanung in dynamischen Umgebungen. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

[Veloso *et al.*, 1998a] Manuela Veloso, Peter Stone, und Kwun Han. The CMUnited-97 robotic soccer team: Perception and multiagent control. In *Autonomous Agents – Second Internation Conference (Agents'98)*. ACM Press, 1998.

[Veloso *et al.*, 1998b] Manuela Veloso, Peter Stone, Kwun Han, und Sorin Achim. The CMUnited-97 small robot team. In Kitano [1998], Seiten 242–256.

[Veloso *et al.*, 1998c] Manuela Veloso, William Uther, Masahiro Jujita, Minoru Asada, und Hiroaki Kitano. Playing soccer with legged robots. In *Proc. International Conference on Intelligent Robots and Systems (IROS'98)*, Victoria, Oktober 1998.

[Veloso und Stone, 1998] Manuela Veloso und Peter Stone. Indiviual and collaborative behaviors in a team of robotic soccer agents. In *Third International Conference on Multi-Agent Systems (ICMAS'98)*, Seiten 309–316, Paris, France, July 1998. IEEE Computer Society Press.

[Vestli *et al.*, 1994] S. Vestli, N. Tschichold-Gürman, und H. Andersson. Learning control and localisation of mobile robots. In *Autonome Mobile Systeme (AMS'94)*, Seiten 202–213. Springer-Verlag, 1994.

[Vestli, 1995] Sjur Jonas Vestli. *Fast, accurate and robust estimation of mobile robot position and orientation*. PhD thesis, Swiss Federal Institute of Technology Zürich, 1995.

[Wang, 1990] C.M. Wang. Location estimation and uncertainty analysis for mobile robots. In Cox und Wilfong [1990], Seiten 90–95.

[Weigel, 1999] Thilo Weigel. Roboter-Fußball: Selbstlokalisierung, Weltmodellierung, Pfadplanung und verhaltensbasierte Kontrolle. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Institut für Informatik, 1999.

[Weiß *et al.*, 1994a] G. Weiß, C. Wetzler, und E.v. Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *Proc. International Conference on Intelligent Robots and Systems (IROS'94)*, September 1994.

[Weiß *et al.*, 1994b] G. Weiß, C. Wetzler, und E.v. Puttkamer. Positions- und Orientierungsbestimmung von bewegten Systemen in Gebäuden durch Korrelation von Laserdaten. In *Autonome Mobile Systeme (AMS'94)*, Seiten 55–64. Springer-Verlag, 1994.

[Weiß und Puttkamer, 1995] G. Weiß und E.v. Puttkamer. A map based on laserscans without geometric interpretation. In U. Rembold et al., editor, *Intelligent Autonomous Systems*, Seiten 403–407. IOS Press, 1995.

[Werger *et al.*, 1998] Brian Werger, Pablo Funes, Miguel Schneider Fontan, Randy Sargent, Carl Witty, und Tim Witty. The spirit of bolivia: Complex behavior through minimal control. In Kitano [1998], Seiten 348–356.

[Wullschleger *et al.*, 1999] Felix H. Wullschleger, Kai O. Arras, und Sjur J. Vestli. A flexible exploration framework for map building. In *Proc. 3rd European Workshop on Advanced Mobile Robots (EUROBOT'99)*, Zürich, September 1999.

[Yokota *et al.*, 1998] K. Yokota, K. Ozaki, A. Matsumoto, K. Kawabata, H. Kaetsu, und H. Asama. Omni-directional autonomous robots cooperating for team play. In Kitano [1998], Seiten 333–347.

[Yokota *et al.*, 1999] K. Yokota, K. Ozaki, N. Watanabe, A. Matsumoto, D. Koyama, T. Ishikawa, K. Kawabata, H. Kaetsu, und H. Asama. Cooperative team play based on communication. In Asada und Kitano [1999].

[Zell *et al.*, 1999] Andreas Zell *et al.* The T-Team. In Asada und Kitano [1999].

[Zhang und Faugeras, 1992] Z. Zhang und O.D. Faugeras. Estimation of displacements from two 3d frames obtained from stereo. *Trans. Pattern Analysis and Machine Intelligence*, 14(2):1141–1156, 1992.

[Zimmer, 1995] U.R. Zimmer. Self-localization in dynamic environments. In *IEEE/SOFT Int. Workshop BIES*, 1995.