DISS. ETH Nr. 9803

Aufdatierung der Position und der Orientierung eines mobilen Roboters

ABHANDLUNG
Zur Erlangung des Titels
Doktor der technischen Wissenschaften

der
EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE
ZÜRICH

vorgelegt von
ALOIS ALBERT HOLENSTEIN
dipl. Elektroingenieur ETH
geboren am 3.Oktober 1960
von Fischingen TG

Angenommen auf Antrag von:
Prof.Dr.M.Mansour, Referent
Prof.Dr.G.Schweitzer, Korreferent
Dr.E.Badreddin, Korreferent

1992

Dank

Die vorliegende Arbeit entstand in den Jahren 1987-1991 unter der Aufsicht von Herrn Prof.M.Mansour während meiner Tätigkeit als Assistent am Institut für Automatik der ETH Zürich. Ihm möchte ich dafür danken, dass er mir diese Arbeit ermöglicht hat.

Herrn Prof.G.Schweitzer bin ich zu Dank verpflichtet für die Übernahme des Korreferats sowie für die konstruktive Durchsicht der vorliegenden Arbeit.

Besonders danken möchte ich auch meinem direkten Vorgesetzten und Leiter des RAMSIS-Projektes, Herrn Dr.E.Badreddin, für die Anregung zu dieser Arbeit, für die Betreuung und für die vielen klärenden Diskussionen.

Nicht vergessen zu danken möchte ich meinen beiden langjährigen Bürokollegen, Dr.Remo Bless und Markus Müller, die während der Durchführung dieser Arbeit immer ein offenes Ohr hatten für kleinere und grössere Probleme. In diesen Dank eingeschlossen sind auch alle anderen Mitarbeiter des Instituts für Automatik.

Leer - Vide - Empty

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Bestimmung der Position und der Orientierung eines mobilen Roboters in einer strukturierten Umgebung. Bekannte Techniken, wie z.B. die Führung mittels Leitdraht oder das Anbringen von passiven oder aktiven Positionierhilfen, sind weit verbreitet, bedeuten aber eine wesentliche Einschränkung der Autonomie und der Flexibilität. Odometrie, d.h. die Positionsbestimmung mittels Aufintegration der Radumdrehungen, ist für längere Fahrstrecken zu ungenau. Mit dem hier vorgestellten Verfahren kann die Position ohne künstliche Markierungen bestimmt werden. Es werden lediglich Ultraschallabstandsmessungen und ein Referenzmodell der Umgebung benötigt. Ergänzt werden die Ultraschallsensoren durch ein Bildverarbeitungssystem, welches für das Heranfahren an eine Andockstelle eingesetzt wird. Die Positionsbestimmung ist Teil einer verhaltensbasierten Regelungsstruktur und ist auf dem institutseigenen mobilen Roboter RAMSIS implementiert und ausgetestet worden.

Die Positionsbestimmung ist deshalb ein Problem, weil dafür geeignete Objekte in der Umgebung gefunden werden müssen. Diese Suche ist meistens schwierig und benötigt viel Rechenzeit, vor allem beim Start, wenn die Roboterposition noch völlig unbekannt ist. Oft kann auch nicht garantiert werden, dass der Roboter genügend Objekte sehen kann. Aus diesen Gründen ist es notwendig, die Art der Positionsbestimmung der jeweiligen Fahrsituation anzupassen.

Beim Start wird die ganze Umgebung nach erkennbaren Objekten abgesucht, welche in einem sensorbasierten Modell abgelegt werden. Danach wird dieses Modell mit dem Referenzmodell verglichen und dabei die wahrscheinlichste Roboterposition bestimmt. Das dafür

üblicherweise eingesetzte Suchverfahren ist sehr rechenaufwendig, wenn mehr als zwei oder drei Hindernisse vorhanden sind. Wir verwenden deshalb ein von uns definiertes Clusteringverfahren, mit welchem ein schneller und sicherer Vergleich möglich ist.

Fährt der Roboter, so wird die Position odometrisch bestimmt und die dabei entstehenden Fehler anhand von Ultraschallabstandsmessungen, die in ein Kalmanfilter eingespiesen werden, so oft wie möglich korrigiert. Dies setzt aber voraus, dass der Odometriefehler geschätzt werden kann, und dass es gelingt zu erkennen, welche Abstandsmessungen von Referenzobjekten stammen. Diese Erkennungsaufgabe wird hier auf die Frage reduziert, ob der untersuchte Abstandswert ungefähr dem erwarteten Abstand zu einem bekannten Objekt entspricht. Ein weiteres Problem ist, dass das Kalmanfilter den Orientierungsfehler nur schlecht schätzen kann, wenn nur Abstandsmessungen vorliegen. Wir stellen deshalb eine ergänzende Methode vor, die für diese Aufgabe besser geeignet ist.

Beim Heranfahren an eine Andockstelle werden die Abstandsmessungen durch ein Bildverarbeitungssystem ergänzt. Dies erlaubt eine sichere Identifikation der Andockstelle und bringt zusätzliche Winkelmessungen.

Ein wichtiger Teil der Arbeit ist die Implementation auf einem mobilen Roboter und das Austesten in einer realen Umgebung. Die dabei erzielten Resultate zeigen, dass eine gute Positionsschätzung ohne Hilfe von künstlichen Markierungen erreicht werden kann, auch wenn der Roboter mit normaler Geschwindigkeit fährt.

Summary

This thesis treats the problem of mobile robot position determination in a structured environment. Known techniques such as wire-guidance or the application of passive or active beacons are widely used, but they severely restrict the mobile robot's autonomy and flexibility. Odometry, that is positioning by measuring the wheel revolutions, is assumed to be not accurate enough for longer distances. With the method presented in this thesis, the position can be determined without any artificial beacons. Only ultrasonic range measurements and an a priori known reference model are needed. The ultrasonic sensors are complemented by a vision system that supports docking operations. The positioning method is part of a behavior-based control structure and has been implemented and tested on our mobile robot RAMSIS built at our laboratory.

The position determination is an interesting problem, because it implies that reference objects must be found in the environment. The search for these objects is difficult and requires a lot of processing power, especially at initialization time, when the robot position is completely unknown. Moreover, there are many situations where it is impossible to see enough objects. These reasons make it necessary to adapt the position determination method to the actual driving situation.

At the beginning the whole environment is looked for identifiable objects that are put into a sensor based model. This model is then matched with the reference model. Thereby, the most likely robot position is calculated. The normally used search algorithm is very time consuming if there are more than two or three obstacles. We therefore use a time optimized clustering algorithm which guarantees a fast model

matching.

When the robot moves, its position is calculated odometrically and the errors are corrected by the help of ultrasonic range measurements, that are fed into a Kalman filter. This means that the errors from odometry have to be estimated, and that it must be possible to determine which range measurements originate from reference objects and which do not. This distinction is done by comparing measured distances with distances expected from the reference model and the actual robot position. Another problem is that Kalman filters are not suited to estimate orientation errors when only range information is available. We therefore introduce a complementary method that is better suited to solve this problem.

When approaching a docking station, the range measurements are complemented by angle measurements to the docking station. This allows for a reliable identification of the docking station and gives additional orientation information.

An important part of this thesis is the implementation and the testing of the algorithms on a real mobile robot in a real environment. We have shown that a sufficient position accuracy can be reached without using artificial beacons, although the robot moves at working speed.

Inhaltsverzeichnis

Figurenverzeichnis

Notationen

Die Bedeutung der meisten Bezeichnungen ist aus dem jeweiligen Zusammenhang ersichtlich. Unten sind deshalb nur diejenigen Notationen angegeben, die über die ganze Arbeit die gleiche Bedeutung haben.

Vektoren und Matrizen

$\mathbf{x}$	$\equiv [x_1, x_2, .., x_n]^T$ Vektor mit den Komponenten $x_1, .., x_n$
$\mathbf{M}$	Matrix mit den Elementen $\{m_{ij}\}$

in der Ebene:

$\|\mathbf{p}\|$	$\equiv \sqrt{p_x^2 + p_y^2}$, Vektornorm
$\angle\mathbf{p}$	Vektorrichtung
$\mathbf{pq}$	$\equiv [p_x q_x - p_y q_y, p_y q_x + p_x q_y]^T$, komplexe Multiplikation
$\|\mathbf{pq}\|$	$= \|\mathbf{p}\|\|\mathbf{q}\|$
$\mathbf{p}^*$	$\equiv [p_x, -p_y]^T$, komplexe Konjugierte
$\mathbf{p}^{-1}$	$\equiv \mathbf{p}^*/\|\mathbf{p}\|^2$, Inverse
$\mathbf{p}'\mathbf{q}$	$\equiv p_x q_x + p_y q_y$, Skalarprodukt

Roboter

$\mathbf{x}_R$	$\equiv [x_R, y_R, \phi_R, \psi_R]^T$, Position des Roboters
$\mathbf{x}_{Odo}$	$\equiv [x_{Odo}, y_{Odo}, \phi_{Odo}, \psi_{Odo}]^T$, Odometrisch bestimmte Position des Roboters
$\Delta\mathbf{x}$	$\equiv [\Delta x, \Delta y, \Delta\phi, \Delta\psi]^T$, Positionskorrektur
v_R	Geschwindigkeit des Roboters
$\dot{\phi}_R$	Drehgeschwindigkeit des unteren Teils

$\dot{\psi}_R$	Drehgeschwindigkeit des oberen Teils

Mengen

A	$\equiv \{a_i\}$, Menge mit den Elementen a_i
$\{a, b, c\}$	Menge bestehend aus den Elementen a, b, c

Komplexität von Algorithmen

$O(f(n))$	ist n ein Mass für die Grösse eines Problemes, so ist der asymptotische Anstieg der Berechnungszeit $T(n)$ höchstens von der Grössenordnung von n, d.h. es existieren zwei Konstanten $k_1 > 0$ und k_2, so dass gilt: $T(n) \leq k_1 f(n) + k_2, \quad \forall n \in N$
$\Omega(f(n))$	$T(n) \geq k_1 f(n) + k_2, \quad \forall n \in N$
$O(f(m,n))$	die Komplexität eines Algorithmus', angewendet auf ein Problem, dessen Grösse durch die Parameter m und n gegeben ist, verlangt einen Rechenaufwand $T(m,n)$, so dass $T(m,n) \leq k_1 f(m,n) + k_2, \quad \forall m \in N, \forall n \in N$ (wobei $k_1 > 0$ und k_2 zwei von m und n unabhängige Konstanten sind).
$\Omega(f(m,n))$	$T(m,n) \geq k_1 f(m,n) + k_2, \quad \forall m \in N, \forall n \in N$

Wahrscheinlichkeitsrechnung

X	Zufallsvariable
$\hat{x}$	Schätzwert
$\mathbf{C}, \mathbf{P}$	Covarianzmatrizen
$P(A)$	Wahrscheinlichkeit, dass ein Ereignis A eintritt

Kapitel 1

Einführung

Als Roboter bezeichnet man eine flexibel programmierbare Arbeitsmaschine, mit der ein Objekt in eine definierte Lage im Raum gebracht werden kann, oder mit der mit dem Objekt eine definierte Bewegung ausgeführt werden kann, um eine bestimmte Arbeitsaufgabe durchzuführen. Kann sich diese Arbeitsmaschine selber in der Ebene oder im Raum verschieben, so wird sie als *mobiler Roboter* bezeichnet. Ein stationärer Roboter kann mobil gemacht werden, indem er auf eine mobile Plattform aufgebaut wird. Die Entwicklung einer solchen Plattform ist heute zu einem eigenständigen Forschungsgebiet geworden. Häufig wird deshalb der Begriff mobiler Roboter verwendet, obwohl man nur die Plattform meint.

1.1 Ziel der Arbeit

Das Ziel dieser Arbeit ist es, die Bestimmung der Position und der Orientierung von mobilen Robotern zu verbessern. Dies ist notwendig, weil heutige Verfahren nur funktionieren, wenn entweder die Umgebung entsprechend präpariert wird, oder, bei unveränderter Umgebung, wenn die Robotergeschwindigkeit sehr niedrig gehalten wird.
Die vorliegende Arbeit ist folgendermassen aufgebaut. Nach einigen einführenden Bemerkungen zur Umgebung und zur Regelungsstruk-

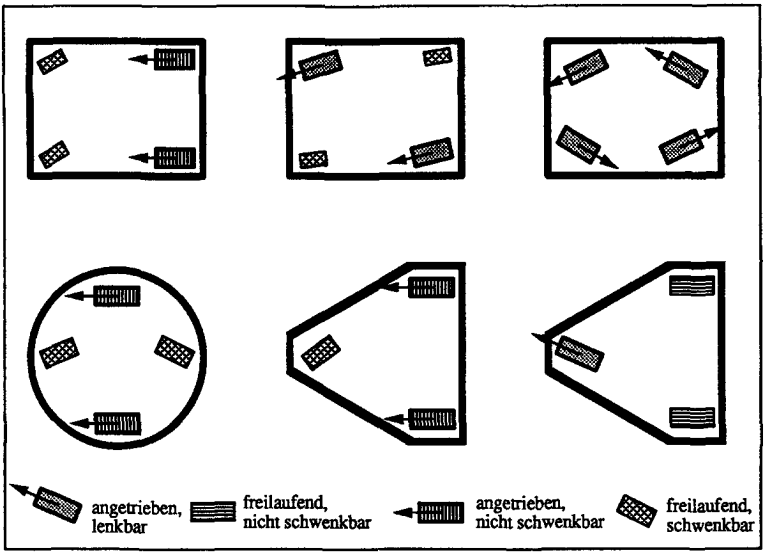
tur von mobilen Robotern, sowie zur Aufdatierung von Messungen konzentrieren wir uns in Kapitel 2 auf die eigentliche Positionsaufdatierungsaufgabe. Bestehende Methoden zur Positionsbestimmung von mobilen Robotern werden vorgestellt und beurteilt. Die Vorstellung unseres Verfahrens schliesst dieses Kapitel ab. Das nächste Kapitel ist der Umgebungsmodellierung und der Modellierung von Sensordaten gewidmet. Auch hier werden verschiedene Ansätze vorgestellt, beurteilt und in eine für die Positionsaufdatierung geeignete Form gebracht. In den drei nächsten Kapiteln werden je eine Methode vorgestellt, mit denen die Position im Stillstand, beim Fahren und vor einer Andockstelle bestimmt bzw. aufdatiert werden kann. Die Notwendigkeit dieser Aufteilung wird in Kapitel 2 gezeigt. Alle Algorithmen wurden auf dem institutseigenen Roboter RAMSIS implementiert und ausgetestet. Details dazu finden sich in Kapitel 7. Kapitel 8 schliesst die Arbeit ab mit einer Diskussion der erreichten Resultate und einem Ausblick auf wünschenswerte Weiterentwicklungen.

1.2 Mobile Roboter in strukturierter Umgebung

Meistens ist es sinnvoll, das Einsatzgebiet von mobilen Robotern einzuschränken, da sich dadurch ihre Konstruktion wesentlich vereinfachen lässt. Hier wird als Einsatzgebiet eine sogenannt strukturierte Umgebung angenommen. Die wichtigsten Merkmale solcher Umgebungen sind ebene Böden, sowie das Vorhandensein von ortsfesten Ecken, Kanten und ebenen Wänden. Diese Bedingungen erscheinen restriktiv, sind aber in Fabriken, Lagerhallen, Warenhäusern, usw. oft erfüllt. Dort liegen auch die meisten heute vorstellbaren Aufgabenbereiche, wie z.B. das Transportieren von Waren und Werkzeugen, das Reinigen von Böden, das Warten und Reparieren von anderen Maschinen, das Bearbeiten von sehr grossen Werkstücken, usw. (siehe z.B. [Miller 88], S.6).

Als **Antriebe** werden für mobile Roboter vorwiegend Elektromotoren eingesetzt, die ihre Energie von eingebauten Batterien beziehen. Vor allem bezüglich Sauberkeit und Einfachheit in der Anwendung sind elektrische Antriebe allen anderen Antriebsarten klar überlegen.

Für das Verschiebbarmachen der Plattform in der Ebene sind **Räder**

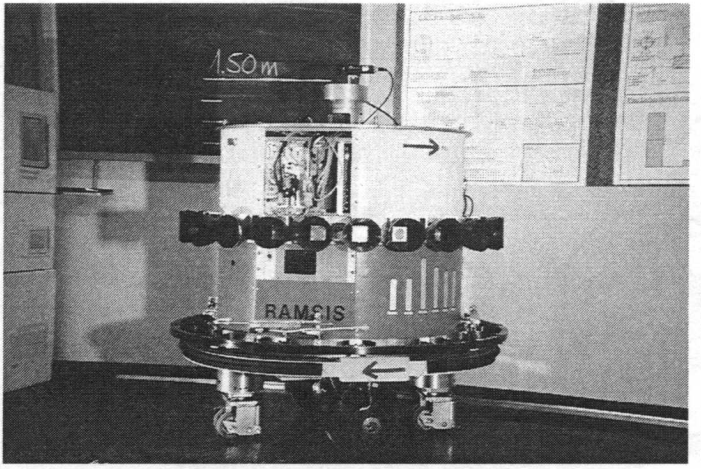


Figur 1.1: *Formen von mobilen Robotern und die Anordnung ihrer Räder.*

am besten geeignet. Bewegungsmittel wie z.B. Beine oder Raupen sind viel komplizierter zu konstruieren und sind deshalb höchstens in unwegsamem Gelände, bzw. für die Überwindung von Treppen oder steilen Rampen gerechtfertigt. Räder lassen sich auf verschiedene Arten anordnen. Einige gebräuchliche Formen sind in Fig. 1.1 abgebildet.

Je flexibler und autonomer ein mobiler Roboter ist, umso grösser sind seine Einsatzmöglichkeiten. Leitliniengeführte Fahrzeuge gelten als Stand der Technik, haben aber den Nachteil, dass sie nur hardwaremässig vorgegebene Wege abfahren können. Als nächste Entwicklungsstufe gilt die freie Programmierbarkeit des abzufahrenden Weges. Schlussendlich soll der mobile Roboter seinen Weg zu einem vorgegebenen Ziel aber selber finden können, unabhängig davon, ob Hindernisse vorhanden sind oder nicht.

Eng verknüpft mit den Fähigkeiten eines mobilen Roboters ist dessen **Sensorik**. Aus verschiedensten Gründen ist es sinnvoll, nur soviel Sensorik zu verwenden, wie unbedingt notwendig ist. Diese Notwendigkeit

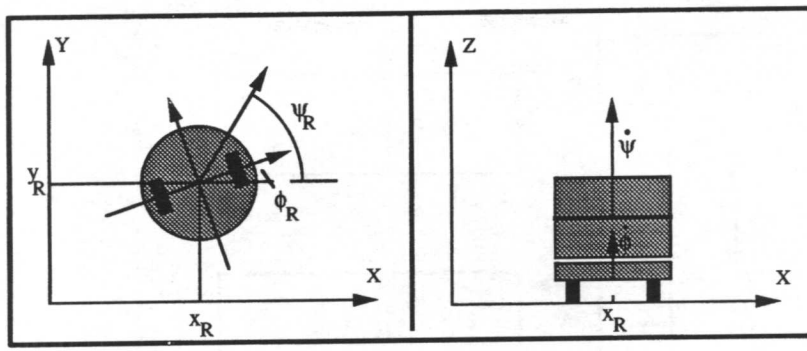


Figur 1.2: *Der mobile Roboter RAMSIS*

wiederum wird von verschiedenen Randbedingungen bestimmt, wie z.B. dem Einsatzgebiet oder den Aufgabenstellungen. Heute werden vor allem inkrementelle Weggeber, Ultraschalldistanzmesser, Bildverarbeitungssysteme, Laserdistanzmesser, taktile Sensoren, usw. ([Everett 89], [Tsumura 86a]) eingesetzt.

1.3 RAMSIS

Ein grosser Teil der nachfolgenden Ausführungen gilt für alle Arten von mobilen Robotern. Vieles kann aber nur anhand eines konkreten Beispieles erklärt werden. In diesen Fällen beziehen wir uns immer auf RAMSIS, den an unserem Institut gebauten mobilen Roboter (Figuren 1.2 und 1.3). Als Besonderheit ist der obere Teil, der sogenannte Turm, gegenüber dem unteren Teil verdrehbar. Die Orientierung des oberen Teils wird durch ψ, die Orientierung des unteren Teils durch ϕ angegeben.



Figur 1.3: *Kinematik von RAMSIS*

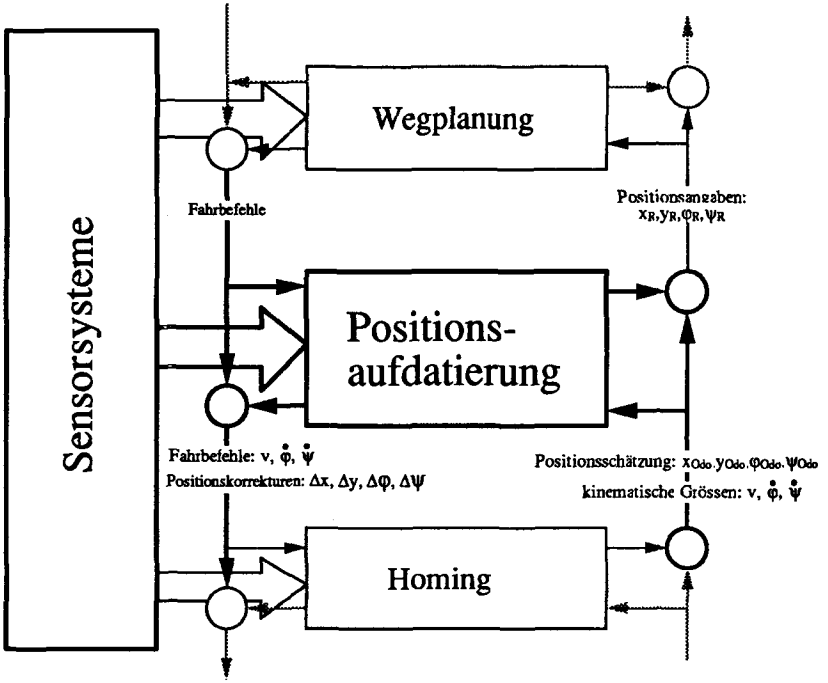
1.4 Verhaltensbasierte Regelungsstruktur

Für die Regelung von mobilen Robotern existieren verschiedene Ansätze. Wir verwenden eine sogenannte *Verhaltensbasierte Regelungsstruktur* ([Badreddin 89b]). Dabei wird das System in verschachtelte Ebenen aufgeteilt, wobei jede Ebene einen Vorwärtspfad zur tieferen Ebene und einen Rückwärtspfad zur höheren Ebene besitzt (Figur 1.4). Die *Verhaltensbasierte Regelungsstruktur* besitzt folgende wesentlichen Eigenschaften:

1. Sie ist streng rekursiv. Interaktionen sind nur möglich zwischen benachbarten Verhaltensebenen. Die Nachteile einer solchen Restriktion werden durch den Vorteil aufgehoben, dass es so wesentlich einfacher ist, günstige Hardware- und Softwareschnittstellen zu definieren. Dies erleichtert zudem die Aufteilung in einzelne Arbeitspakete.

2. Die Verhaltensebenen sind verschachtelt. Die Verhalten mit grösserer Bandbreite, d.h. die reflexiven Verhalten, befinden sich in den innersten Regelschleifen, während die langsameren, aber "intelligenteren" Verhalten aussen sind.

3. Sie lässt sich von innen nach aussen, bzw. von unten nach oben entwerfen (bottom-up).



Figur 1.4: *Positionsaufdatierung innerhalb der verhaltensbasierten Regelungsstruktur. Auf jeder Ebene sind ganz bestimmte Aufgaben zu lösen, wobei der Datenaustausch zu anderen Ebenen so gering wie möglich zu halten ist. Der Zugriff auf die Sensordaten ist auf jeder Ebene möglich.*

4. Im Gegensatz zu anderen Vorschlägen besitzt sie eine feste Struktur, wodurch ihre Analyse erleichtert wird.

5. Die Modellierung ist über alle Ebenen verteilt, d.h. auf jeder Ebene wird ein Modell benutzt, welches der jeweiligen Aufgabe angepasst ist.

6. Eine explizite Vermischung von Sensordaten ist nicht notwendig.

Die Positionsaufdatierung wird als eigenständige Ebene betrachtet (Figur 1.4). Sie liegt direkt über der Homingebene, auf welcher bereits eine grobe odometrische Bestimmung der Position erfolgt (x_{Odo}, y_{Odo},

ϕ_{Odo}, ψ_{Odo}). Noch tiefer liegende Ebenen sind die Kollisionsverhinderung und die Geschwindigkeitsregelung auf Roboterniveau (v, $\dot{\phi}$, $\dot{\psi}$). An die höhere Ebene (Wegplanung) wird die genaue Roboterposition übergeben, während nach unten nur Korrekturen (Δx, Δy, $\Delta\phi$, $\Delta\psi$) gemacht werden. Wichtig ist, dass auf der Positionsaufdatierungsebene auch Fahrbefehle erzeugt werden können. Dies kann z.B. dazu ausgenützt werden, den Roboter am Wegfahren zu hindern, wenn die Anfangsposition noch nicht fertig bestimmt worden ist.

1.5 Aufdatierung von Messungen

Daten, die mit der Zeit ihre Gültigkeit verlieren, müssen laufend wieder auf den neuesten Stand gebracht werden, d.h. es muss eine Aufdatierung erfolgen. Wann und wie oft ein Aufdatierungsschritt durchzuführen ist, hängt davon ab, wie gross der Anteil der falschen Daten sein darf. Messwerte sind eine spezielle Form von Daten. Bei Verfahren, die zur Bestimmung der gesuchten Grösse eine gemessene Grösse aufsummieren oder aufintegrieren müssen, wird der dabei auftretende Messfehler ebenfalls aufsummiert. Dadurch steigt der resultierende Fehler kontinuierlich. Wird das Messverfahren über längere Zeit eingesetzt, sind die ermittelten Resultate nicht mehr zuverlässig. Damit dieser Fall nicht eintritt, müssen die immer ungenauer werdenden Resultate rechtzeitig korrigiert werden, d.h. die Messung muss aufdatiert werden. Warum wird aber nicht gleich jenes Messverfahren verwendet, welches die genaueren Resultate liefert? Dafür kann es verschiedene Gründe geben:

- Die Bedingungen für die Anwendung des genaueren Messverfahrens sind nicht immer erfüllt. Beispielsweise sind für die Positionsaufdatierung notwendige Referenzpunkte nicht aus allen Lagen sichtbar.
- Das genaue Verfahren ist rechenaufwendig und kann deshalb nur in grösseren Zeitabständen benutzt werden.

Bei der Wahl eines Aufdatierungsverfahrens sind verschiedene Aspekte zu betrachten. Ist die aufzudatierende Grösse kontinuierlich oder diskret, eindimensional oder mehrdimensional, verläuft sie stetig oder unstetig, gehört sie zu einem linearen oder nichtlinearen System, ist die

Dauer zwischen zwei Aufdatierungsschritten kurz oder lang, usw. Die Position und Orientierung eines mobilen Roboters ist eine kontinuierliche, mehrdimensionale und stetig verlaufende Grösse. Die Dauer zwischen zwei Aufdatierungsschritten ist von der Verfügbarkeit neuer Sensordaten und von der geforderten Positionsgenauigkeit abhängig.

Kapitel 2

Positionsbestimmung bei mobilen Robotern

Die Position und die Orientierung eines Körpers müssen immer relativ zu einem im voraus definierten Bezugssystem angegeben werden. Im Raum benötigt man dazu drei Grössen, die die Position des Körpers beschreiben, sowie drei Grössen für die Orientierung. Mobile Roboter bewegen sich in der Ebene. Dadurch genügen zwei Werte für die Position und ein Wert für die Orientierung. Wie ein mobiler Roboter diese drei Grössen am besten bestimmen kann, ist das Thema dieser Arbeit. Der Einfachheit halber werden wir in der Folge oft nur von Positionsbestimmung sprechen, obwohl wir damit die Bestimmung der Position und der Orientierung meinen.

2.1 Genauigkeitsanforderungen

Wie genau und wie oft die Position und die Orientierung eines mobilen Roboters bestimmt werden muss, hängt vom Verwendungszweck und von der Umgebung ab. Sinnvoll ist es zu unterscheiden, ob der Roboter unterwegs ist, ob er irgendeine Arbeit ausführen muss, oder ob er an eine Arbeitsstelle heranfahren muss. Ein Vorschlag für die Einteilung in Genauigkeitsklassen ist in Tab. 2.1 angegeben. Dabei verzichten wir

Genauigkeitsklasse	Einsatzmöglichkeiten
$\pm 10cm$	Fahrt zwischen zwei Punkten
$\pm 1cm$	Fahrt entlang einer Trajektorie
$\pm 1mm$	Anfahren einer Andockstelle
$\pm 0.1mm$	Bearbeiten von Werkstücken Pick-and-Place-Operationen

Tabelle 2.1: *Genauigkeitsklassen*

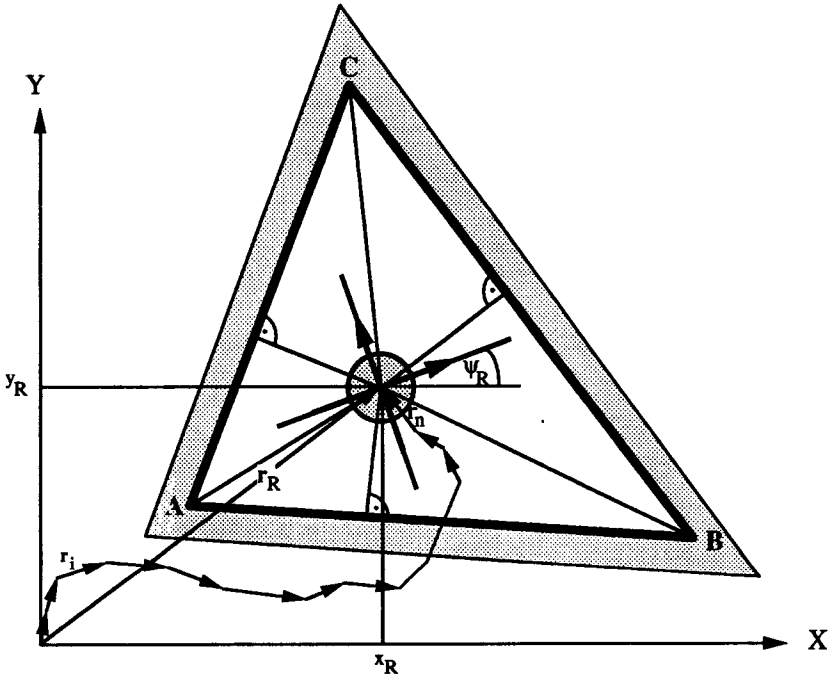
auf die Angabe der Orientierungsgenauigkeit, weil beim Fahren bereits geringe nicht korrigierte Orientierungsfehler zu grossen Positionsfehlern führen. Z.B. ergibt sich bei einer Geradeausfahrt von einem Meter und einem nicht korrigierten Orientierungsfehler von 1° bereits ein Positionsfehler von über 1.7cm.

2.2 Bekannte Methoden

2.2.1 Grundsätzliche Möglichkeiten

Setzt man voraus, dass nur Messungen vom Roboter aus möglich sind, so ergeben sich prinzipiell die in Fig. 2.1 angedeuteten Möglichkeiten, die Roboterposition zu bestimmen. Dabei wird angenommen, dass die Positionen der Ecken (A, B und C) und der Wände, sowie die Zuordnung der Messungen zu diesen, bekannt seien. Die möglichen Methoden sind:

1. 3 Winkelmessungen:
   Aus drei Winkelmessungen kann mittels Triangulation die Position und die Orientierung eindeutig bestimmt werden.
2. 2 Abstands- und 1 Winkelmessung:
   Aus den beiden Abständen kann die Position und aus der Winkelmessung die Orientierung des Roboters ebenfalls eindeutig bestimmt werden.
3. 2 Distanz- und 1 Winkelmessung:
   Aus den Distanzen zu zwei verschiedenen Eckpunkten erhält man



Figur 2.1: *Mobiler Roboter bestimmt seine Position in einem dreieckigen Raum. Dazu können folgende Messungen benutzt werden: Distanzmessungen zu den Eckpunkten A, B und C - Abstandsmessungen zu den Wänden AB, AC und BC - Winkelmessungen zu den Eckpunkten A, B und C.*
Aus jeder beliebigen Kombination von zwei Distanz-, Abstands- und/oder Winkelmessungen lässt sich die Position des Roboters bestimmen. Um die Orientierung zu erhalten, ist mindestens eine zusätzliche Winkelmessung notwendig. Bereits in diesem einfachen Raum sind 16 verschiedene Messkombinationen möglich, aus denen die Roboterposition bestimmt werden kann.

zwei mögliche Roboterpositionen. Die falsche der beiden Lösungen muss durch Einschränkung des Lösungsraumes oder durch eine zusätzliche Messung ausgeschieden werden. Mit der Winkelmessung kann die Orientierung eindeutig bestimmt werden.

4. 1 Distanz-, 1 Abstands- und 1 Winkelmessung:
   Aus der Distanz- und aus der Abstandsmessung ergeben sich wiederum zwei mögliche Positionen, von denen die falsche elimiert werden muss.

5. Aufsummieren zurückgelegter Wegstücke, ausgehend von bekannter Anfangsposition (Odometrie).

6. Integration von Geschwindigkeits- oder Beschleunigungsmessungen, ausgehend von bekanntem Anfangszustand (Dead Reckoning).

Analytisch gesehen sind alle diese Möglichkeiten gleichwertig. In der Praxis müssen aber die Ungenauigkeiten von Distanz-, Abstands- und Winkelmessungen berücksichtigt werden. Bei den Möglichkeiten 5 und 6 hängt die Genauigkeit zudem von der Integrationszeit und von der Fahrdynamik ab.

Nachfolgend soll eine Übersicht bekannter Lösungsvorschläge gegeben werden. Eine Gruppierung dieser Vorschläge könnte nach folgenden Kriterien geschehen:

- *Anfangsposition*: Ist diese bekannt?
- *Umgebung*: Gibt es darin künstliche Marken als Positionierhilfen?
- *Sensorsysteme*: Was und wie wird gemessen?
- *Genauigkeit*: Was gelten für Anforderungen?
- *Rechenaufwand*: Ist dieser zu gross für eine on-line-Aufdatierung?

Betrachten wir die beiden ersten Kriterien, so ergeben sich drei Gruppen von Lösungsvorschlägen, welche sich nicht nur in Details, sondern prinzipiell unterscheiden, und die im folgenden kurz vorgestellt werden.

2.2.2 Positionsbestimmung bei bekannter Anfangsposition

Kennt man die Anfangsposition des mobilen Roboters, so kann durch Hinzuaddieren des gefahrenen Weges die aktuelle Position laufend bestimmt werden. Wird dieser gefahrene Weg mittels Aufsummierung der Radumdrehungen ermittelt, so spricht man von *Odometrie*, wird er durch Integration der Robotergeschwindigkeit bestimmt, so nennt man dies *Dead Reckoning*.
Odometrische Positionsbestimmung ist dann nützlich und sinnvoll, wenn die gefahrenen Wege kurz sind und wenn die Räder wenig Schlupf haben. Diese Bedingungen sind im Labor meistens erfüllt, weshalb man sich gerne damit zufrieden gibt. Trotzdem fehlt es nicht an Versuchen, die Eigenschaften dieser Methode zu verbessern. Diese laufen meistens darauf hinaus, den Radschlupf irgendwie zu messen und dadurch die Wegstücke genauer bestimmen zu können ([Tsumura et al. 83], [Wang 88]). Diese Verfahren funktionieren allerdings nur, wenn der Schlupf relativ klein ist. Treten Situationen auf, bei denen die Räder durchdrehen, so helfen nur noch Messmethoden, die unabhängig sind von den Radumdrehungen. In [Köbbing 89] sind drei Vorschläge zu finden, wie die Geschwindigkeit gegenüber Oberflächen optisch und damit berührungslos gemessen werden kann. Das erste Verfahren arbeitet nach dem Doppler-Prinzip, das zweite nach dem Prinzip der Ortsfrequenzmessung und das dritte nach dem Korrelations-Prinzip. Das Doppler-Prinzip wird bei sogenannten Interferometern ausgenützt. Der Abstand zur gemessenen Oberfläche muss aber in so engen Grenzen gehalten werden, dass ein Einsatz für mobile Roboter nicht in Frage kommt. Das zweite Verfahren lässt sich für niedere Geschwindigkeiten nur bedingt verwenden und ist überdies zu ungenau, um daraus die Position ermitteln zu können. Am vielversprechendsten ist das dritte Verfahren. Es ist aber sehr aufwendig und benötigt entsprechend viel Rechenzeit. Einfacher ist ein völlig anderes Verfahren, nämlich die Beschleunigung eines Körpers zu messen. Da für die Berechnung des Weges zweimal integriert werden muss, sind hohe Anforderungen an die Messgenauigkeit zu stellen. Mit heutigen Sensoren ist es nicht möglich, die für mobile Roboter erforderliche Genauigkeit über längere Distanzen zu gewährleisten. Hochpräzise Beschleunigungsmesser reagieren zudem empfindlich auf Erschütterungen. Für die Stabilisierung der Orientierung kann ein Kreisel eingesetzt werden. Deren Genauigkeit reicht aus,

um den mobilen Roboter während mehreren Minuten auf dem richtigen Kurs zu halten.

2.2.3 Positionsbestimmung mit künstlichen Umgebungsmerkmalen

Das Problem der Positionsbestimmung kann wesentlich vereinfacht werden, wenn Positionierhilfen bzw. Landmarken in die Umgebung eingebaut werden, die sich leicht von anderen Umgebungsmerkmalen unterscheiden lassen. Solche Positionierhilfen können aktiv oder passiv sein und müssen auf die auf dem mobilen Roboter vorhandenen Sensorsysteme abgestimmt sein. Ausser in einfachen Fällen, wo alle Landmarken jederzeit sichtbar sind, braucht der Roboter unbedingt eine Landkarte, auf der die Positionen der einzelnen Landmarken verzeichnet sind. Alle Verfahren, die mit künstlichen Umgebungsmerkmalen arbeiten, profitieren von der Tatsache, dass die Positionierung eines Körpers trivial ist, sobald einige wenige Winkel- und/oder Abstandsmessungen zu Punkten auf der Landkarte vorliegen. Durch Einführung der künstlichen Landmarken kann man sich die viel schwierigere Aufgabe ersparen oder zumindest wesentlich erleichtern, die durchgeführten Messungen den richtigen Punkten auf der Landkarte zuordnen zu müssen.

Beispiele passiver Positionierhilfen sind auf den Boden gemalte Leitlinien oder an den Wänden befestigte Katzenaugen, Strichmuster, Klötze, Spiegel usw. (siehe z.B. [Larcombe 79], [Heiz 83], [Boegli 85], [Tsumura 85], [Tsumura 86a]). Diese Markierungen können auch an der Decke angebracht sein. Obwohl sie sich deutlich von der Umgebung abheben, ist es meistens doch nicht ganz einfach, sie richtig aufzunehmen. Für die Erkennung geeignete Sensorsysteme sind Laserscanner und vor allem Visionsysteme.

Einfacher ist die Erkennung bei aktiven Positionierhilfen. Diese sind meistens über das ganze befahrbare Gebiet verteilt und senden akustische oder elektromagnetische Signale aus. Auf dem mobilen Roboter sind spezielle Empfänger aufgebaut, die die ankommenden Signale identifizieren und deren Richtung bestimmen. Die anschliessende Positionsbestimmung erfolgt in den meisten Fällen mit Hilfe eines Triangulationsverfahrens. Beispiele für Verfahren mit aktiven Positionierhilfen finden sich in [Boegli 85], [Heiz 83] und [Tsumura 86a].

2.2.4 Positionsbestimmung mit natürlichen Umgebungsmerkmalen

Mit natürlichen Umgebungsmerkmalen sind Merkmale gemeint, die auch ohne das Positioniersystem des mobilen Roboters vorhanden wären, d.h. der mobile Roboter soll seine Position bestimmen können, ohne dass Veränderungen an der Umgebung angebracht werden müssen. Diese Aufgabe ist wesentlich anspruchsvoller als die im vorherigen Kapitel beschriebene. Entsprechend kleiner ist auch die Anzahl der dafür bekannten Lösungsvorschläge.

Jeder Lösungsvorschlag macht implizit oder explizit irgendwelche Annahmen über die Beschaffenheit der Umgebung. Je allgemeiner diese Annahmen sind, umso brauchbarer ist die Methode und umso weniger sieht man sich dem Vorwurf ausgesetzt, dass die Methode sowieso nur in dem vorgegebenen Gebiet anwendbar ist. In den meisten Fällen wird angenommen, dass die Umgebung ganz bestimmte geometrische Eigenschaften aufweist, welche einfach in ein Umgebungsmodell übertragen werden können. Dieses wird entweder als geometrisches Modell ([Chatila and Laumond 85], [Miller 85], [?]) oder als zweidimensionales Gitter mit besetzten und unbesetzten Feldern ([Crowley 89], [Marce and Julliere 86], [Microbe 85]) aufgebaut. Mit zweidimensionalen Gittermodellen lassen sich auch unstrukturierte Umgebungen relativ einfach beschreiben, was mit geometrischen Modellen praktisch unmöglich ist.

Ob das Umgebungsmodell vorgegeben ist oder nicht, ist eine weitere wesentliche Frage bei jeder Art von Positionsbestimmung. Wird eine hohe Genauigkeit gefordert, so kommen meistens nur vorgegebene Modelle in Frage, da selber aufgenommene Modelle oft mit zu grossen Fehlern behaftet sind.

Im folgenden werden einige vielversprechende Lösungsvorschläge etwas genauer vorgestellt.

Die einfachste Art, eine Positionsaufdatierung durchzuführen, ist in [Severin and Klein 88] angegeben. Zusammen mit den Fahrbefehlen werden dem Roboter Angaben darüber mitgegeben, wo er welche Abstandsmessungen für die Korrektur der odometrisch bestimmten Position durchzuführen hat. Hindernisse werden nicht berücksichtigt. Alle Abstandsmessungen werden als richtig angenommen.

In [Chatila and Laumond 85] wird der Raum in Sichtbarkeitspolygone aufgeteilt, welche zu einem Umgebungsmodell zusammengefügt werden. Aus dem Vergleich von Kanten und Eckwinkeln aus diesem Umgebungsmodell mit Kanten und Eckwinkeln, die mit einem Laserdistanzmesser aufgenommen werden, kann nach einigen Zwischenschritten die Roboterposition bestimmt werden. Damit die angegebene Methode funktioniert, müssen die Messungen vollständig sein.

Ähnliche Ideen liegen den in [Crowley 89], [Marce and Julliere 86] und [Microbe 85] beschriebenen Methoden zugrunde. Die Information über die Umgebung wird in einem zweidimensionalen Gittermodell abgelegt, indem jedes Feld dieses Gittermodelles entweder als leer, besetzt oder noch nicht untersucht bezeichnet wird. Für die Positionsbestimmung versucht man, vom Roboter aus aufgenommene Abstandsmessungen so gut wie möglich mit dem Umgebungsmodell zur Deckung zu bringen. Eine Modellaufdatierung erfolgt, indem die Einteilung in leere, besetzte und noch nicht erforschte Felder laufend den Sensormessungen angepasst wird.

Wie man eine Positionsbestimmung allein auf Grund von Winkelmessungen durchführen könnte, wird in [Sugihara 87] untersucht. Vorgegeben wird ein Umgebungsmodell, in dem die Positionen aller senkrechter Kanten der Testumgebung abgelegt sind. Die Positionsbestimmungsaufgabe besteht nun darin, ein Büschel von Winkelmessungen so in die Ebene zu legen, dass von der gefundenen Position aus jedem Strahl des Büschels eine Kante im Umgebungsmodell zugeordnet werden kann. Falls die Umgebung einfach und die Messfehler nicht zu gross sind, ist das Verfahren recht vielversprechend. Weniger günstig sieht es aus, wenn die Winkelmessungen zu ungenau sind, weil dadurch die Anzahl der möglichen Roboterpositionen sehr schnell ansteigt. Auch hat dies negative Auswirkungen auf die Geschwindigkeit des angegebenen Algorithmus'. Mit einem Visionsystem kann die geforderte Genauigkeit kaum errreicht werden. Die Qualität des Verfahrens könnte aber wesentlich gesteigert werden, wenn die Kanten unterscheidbar gemacht werden könnten, was z.B. mit Farben einfach zu bewerkstelligen wäre. Dies wäre aber bereits wieder eine Änderung an der Umgebung und müsste deshalb unter dem vorhergehenden Kapitel eingeordnet werden.

2.2.5 Beurteilung

Nimmt man an, dass die Umgebung mit künstlichen Merkmalen präpariert werden darf, so kann das Problem der Positionsbestimmung von mobilen Robotern als gelöst betrachtet werden. Aus einer Reihe gleichwertiger Verfahren kann je nach Bedarf das geeignetste ausgesucht werden. Ein Vergleich der Lösungsmethoden ist aber nur bedingt möglich, da die erreichten Resultate praktisch nie mit überprüfbaren Zahlen angegeben werden. Dies ist auch schwierig, da scheinbar nebensächliche Faktoren wie z.B. die Beschaffenheit der Bodens, die Konstruktion des mobilen Roboters, oder auch die gefahrene Teststrecke einen grossen Einfluss haben auf die Qualität der Resultate. Die meisten Lösungsvorschläge werden, wenn überhaupt, nur mittels Simulationen verifiziert. Dabei werden oft Annahmen gemacht, die nirgends dokumentiert sind.

Weniger gut sieht es aus, wenn aus Gründen der Autonomie keine Änderungen an der Umgebung gemacht werden sollen. Am allgemeinsten anwendbar sind diejenigen Verfahren, bei denen die Umgebung in einem zweidimensionalen Gittermodell abgelegt wird. Diese sind am tolerantesten gegenüber Sensorfehlern und nicht modellierten Hindernissen. Dafür sind diese Verfahren relativ ungenau und sehr langsam. Schnellere Verfahren, die mit einer geometrischen Modellierung der Umgebung arbeiten, funktionieren nur, wenn diese Umgebung eine Reihe von Anforderungen erfüllt. Die wichtigste ist dabei, dass keine oder nur sehr wenige Hindernisse vorhanden sein dürfen.

Ist die Anfangsposition einmal bestimmt, so kann der weitere Positionsverlauf mittels Koppelnavigation am schnellsten berechnet werden. Um die sich addierenden Fehler in Grenzen halten zu können, muss in gewissen Zeitabständen die Anfangsposition neu bestimmt werden. Aus betriebstechnischer Sicht wäre aber eine laufende Positionskorrektur besser. In [Wiklund 89] und in [Leonard and DurrantWhyte 91] werden dazu zwei verschiedene Lösungen vorgeschlagen. Der erste Vorschlag benutzt für die Positionskorrektur Winkelmessungen zu bekannten Reflektoren, weshalb er in die Gruppe mit den künstlichen Umgebungsmerkmalen gehört. Interessanter ist der zweite Ansatz. Von einer bekannten Anfangsposition startend bewegt sich der Roboter in einer Umgebung, von der ein Referenzmodell bekannt ist. Die Odometriefehler werden anhand von Abstandsmessungen zu bekannten Umgebungsmerkmalen

korrigiert. Wir werden in Kapitel 5 noch näher auf dieses Verfahren eingehen.

Von den vorgestellten Verfahren interessieren uns von jetzt an nur noch diejenigen, die ohne künstliche Umgebungsmerkmale auskommen. Zusammenfassend kann man zu diesen Verfahren Folgendes feststellen:

- Sobald die Umgebung als nicht vollständig bekannt angenommen werden muss, benötigt die Bestimmung der Anfangsposition unverhältnismässig viel Zeit.
- Odometrie ist zu ungenau und muss durch einen Aufdatierungsalgorithmus ergänzt werden.
- Bei nicht vollständig bekannter Umgebung ist es unmöglich, strenge Genauigkeitsanforderungen einzuhalten.
- Die vorgestellten Lösungsvorschläge werden meistens nur anhand von Simulationen untersucht. Implementationstechnische Erfahrungen fehlen weitgehend.
- Die Positionsbestimmung wird immer nur für sich allein betrachtet, statt sie als Teil der gesamten Robotersteuerung zu sehen.

2.3 Lösungskonzept

2.3.1 Annahmen, Forderungen, Wünsche

Nachfolgend haben wir einen Katalog von Annahmen [A], Forderungen[F] und Wünschen [W] zusammengestellt, die sich teilweise aus den oben gemachten Feststellungen ableiten, teilweise aber auch praktischen Überlegungen entspringen. Klar ist, dass ein direkter Zusammenhang zwischen der Allgemeinheit der Annahmen und der Komplexität des Positioniersystemes besteht. Die unten angegebenen Annahmen und Forderungen sind so allgemein wie möglich formuliert, ohne dass eine Lösung verunmöglicht wird.

A: Die Anfangsposition des mobilen Roboters ist nicht bekannt.

A: Die Umgebung ist strukturiert und teilweise bekannt. Es gibt darin auch Objekte, die sich verschieben können.

A: Odometrie und Dead Reckoning ist für längere Fahrten zu ungenau.

F: Der Roboterstandort muss relativ zur bekannten Umgebung angegeben werden.

F: Das Positionierverfahren muss in die *Verhaltensbasierte Regelungsstruktur* integriert werden.

F: Die Genauigkeitsanforderungen aus Tab. 2.1 sind einzuhalten.

W: Die notwendigen Sensorsysteme sollen auch für andere Zwecke eingesetzt werden können.

2.3.2 Grundidee

Die Grundidee unserer Methode besteht darin, dass wir die Positionsbestimmung nicht als eine reine Messaufgabe, sondern als ein Verhalten des Roboters betrachten. Dies hat den Vorteil, dass die Positionsaufdatierung auch eine aktive Rolle übernehmen kann. Statt nur Daten zu sammeln und daraus die Position zu bestimmen, kann sie durch Vorgabe von anzufahrenden Sollpositionen auch einen Einfluss auf die Roboterbewegungen ausüben. Dies erlaubt dem Roboter zum Beispiel, für die Bestimmung der Anfangsposition an einen dafür günstigen Ort zu fahren.

Ein weiterer wichtiger Punkt ist die Aufteilung der Positionsbestimmung in die drei Phasen Initialisierung - Aufdatierung - Andocken. Damit können die unterschiedlichen Anforderungen bezüglich Rechenzeit und Genauigkeit erfüllt werden.

Damit ist das Problem der Positionsbestimmung eines mobilen Roboters strukturiert. Fahrbefehle werden erst ausgeführt, wenn die Anfangsposition gefunden ist. Während dieser Zeit bewegt sich der Roboter nur, um einem eventuell auf ihn zukommenden Hindernis auszuweichen, oder um zusätzliche Daten zu sammeln. Theoretisch kann er sich für die Positionsbestimmung beliebig viel Zeit lassen, ohne dass etwas Nachteiliges passiert. Ist die Anfangsposition gefunden, so

können mit einem wesentlich weniger aufwendigen Verfahren die Positionsänderungen beim Fahren ermittelt werden. Ist dieses Verfahren aus irgendeinem Grund überfordert, so kann die Geschwindigkeit gedrosselt werden oder, falls notwendig, mit dem ersten Verfahren die Position von Grund auf neu berechnet werden. Fährt der Roboter an eine Andockstelle heran, so kann der Roboter künstliche Positionierhilfen benutzen, da die Andockstelle selber ein künstliches Gebilde darstellt. Nur so kann die Einhaltung der Genauigkeitsanforderungen garantiert werden.

2.3.3 Neue Lösungswege

Das Hauptziel dieser Arbeit ist es, die Positionsbestimmung von mobilen Robotern schneller, flexibler und sicherer zu machen, und diese in das Steuerungssystem eines mobilen Roboters zu integrieren. Um dies zu erreichen, müssen neue Lösungswege beschritten werden.

Am Anfang ist die Roboterposition noch völlig unbekannt und muss durch Vergleich eines sensorbasierten Modelles mit dem vorgegebenen Referenzmodell gefunden werden. Die offensichtliche Methode, die beiden Modelle durch eine systematische Suche sich entsprechender Objekte aufeinander abzubilden, ist sehr zeitaufwendig und lässt sich in der Praxis nur durchführen, wenn die Modelle nicht zu gross sind und wenn höchstens zwei bis drei Hindernisse bzw. falsch erkannte Objekte vorliegen. Diese Bedingungen können aber nicht eingehalten werden, wenn die Objekterkennung nicht absolut zuverlässig ist, oder wenn die Umgebung nur teilweise bekannt ist. In Kapitel 4 werden wir deshalb ein Clusteringverfahren einführen, mit dem die oben erwähnten Nachteile umgangen werden können. Dieses Verfahren erlaubt uns zudem, einen einfachen und effizienten Objekterkennungsalgorithmus zu formulieren.

Beim Fahren wird die Position odometrisch bestimmt (Kapitel 5). Dabei entstehende Positions- und Orientierungsfehler müssen möglichst schnell anhand von Abstandsmessungen korrigiert werden. Diese Aufgabe werden wir mit einem Extended Kalman Filter lösen. Wir werden zeigen, wie die dazu notwendige Schätzung der odometrischen Positionsunsicherheit durchgeführt werden kann. Zudem werden wir eine Methode angeben, wie Abstandsmessungen zu Referenzobjekten von Abstandsmessungen zu Hindernissen genügend schnell, d.h. on-line, unterschieden werden können. Ein weiteres Problem ist, dass wenn der

Orientierungsfehler allein auf Grund von Abstandsmessungen korrigiert wird, dies zu unbefriedigenden Resultaten führt. Wir haben deshalb ein Verfahren entworfen, mit welchem aus dem Verlauf des Positionsfehlers auf den Orientierungsfehler geschlossen werden kann.

Beim Andocken müssen Positionsfehler ausgeschlossen werden können. Wir ergänzen deshalb die Abstandsmessugen durch ein Bildverarbeitungssystem, welches eine sichere Identifikation der Andockstelle erlaubt, und welches zusätzliche Winkelmessungen bringt. Die Details dazu sind in Kapitel 6 beschrieben.

Ein wichtiger Teil der Arbeit ist die Integration der Positionsaufdatierung in die Gesamtsteuerung eines mobilen Roboters. Diese wird dadurch erleichtert, dass die oben beschriebenen Lösungswege auch unter diesem Gesichtspunkt konzipiert wurden.

Leer - Vide - Empty

Kapitel 3

Umgebungsmodelle

A ist ein *Modell* von B, wenn A benutzt werden kann, um Fragen über B beantworten zu können. Mit einem *Umgebungsmodell* müssen Fragen über die Umgebung beantwortet werden können. Mit einem *Sensorbasierten Modell* müssen Fragen über die gemessenen Daten beantwortet werden können.

Das interne Umgebungs- bzw. *Referenzmodell* kann mit einer Landkarte verglichen werden, welche die Position bzw. die Koordinaten von ausgewählten Umgebungsmerkmalen wie Strassen, Häusern, Flüssen, usw. enthält. Versucht jemand, seine Position in einer ihm unbekannten Umgebung zu bestimmen, so geht er etwa folgendermassen vor. Als erstes betrachtet er diese Umgebung und merkt sich einige besondere Umgebungspunkte, d.h. er bildet sich ein *Sensorbasiertes Modell*. Als Sensoren dienen ihm dabei vor allem seine Augen. Dann richtet er seinen Blick auf die Landkarte und versucht, diese Punkte auf der Karte wiederzufinden. Dabei achtet er vor allem auf die gegenseitige Lage der verschiedenen Merkmale. Hat er seine Position entdeckt, so vergewissert er sich nochmals, indem er einige weitere Umgebungsmerkmale zu erkennen versucht.

3.1 Modellierung einer strukturierten Umgebung

Entsprechend der in Kapitel 1.4. eingeführten *Verhaltensbasierten Regelungsstruktur*, nach der auf jeder Ebene nur soviel Information wie nötig vorhanden sein soll, wird hier ein Modell gesucht, welches für die Bestimmung der Position und der Orientierung des mobilen Roboters ausreicht. Nach einem kleinen Abstecher in das Gebiet der geometrischen Modellierung und der Darstellung einiger im Zusammenhang mit mobilen Robotern eingeführten Umgebungsmodellen, folgt eine Beschreibung desjenigen Umgebungsmodelles, welches in dieser Arbeit verwendet wird.

3.1.1 Geometrische Modellierung

Geometrische Modellierung ist die Technik, die benutzt wird, um die Form eines Objektes zu beschreiben. Die Beschreibung komplizierter Objekte wird dabei meistens aus der Beschreibung einfacherer Objekte synthetisiert. Das so gebildete Modell ist eine analytische, mathematische und damit abstrakte Beschreibung eines realen Objektes, die auch benutzt werden kann, um Informationen über dieses Objekt weiterzugeben. Breite Anwendung findet die geometrische Modellierung in Gebieten wie *CAD/CAM*, *Computer Graphics*, *Computer Art*, *Computer Vision* und auch *Robotertechnik*. Der Fortschritt auf diesen Gebieten ist eng damit verknüpft, ob effiziente geometrische Modelle für die Problembeschreibung gefunden werden können.

In den meisten Anwendungen müssen die Objekte dreidimensional modelliert werden. Je nach Aufgabenstellung kommen dabei verschiedene Methoden in Frage, wie z.B. *Graphenbasierte Modelle*, *Boole'sche Modelle*, *Rastermodelle*, *Sweep-Darstellungen*, *Grenzflächendarstellungen* und *Drahtmodelle* ([Mortenson 85]). Lässt sich das zu beschreibende Objekt in einfachere Teilobjekte zerlegen, so ist eine Modellierung relativ problemlos möglich. Dies ist in technischen Anwendungen häufig der Fall. Sehr schwierig wird die Aufgabe hingegen, wenn eine Zerlegung nicht möglich ist. In diesen Fällen hilft meistens nur noch eine approximative Modellierung, d.h. komplizierte geometrische Objekte müssen durch vereinfachte Objekte beschrieben werden.

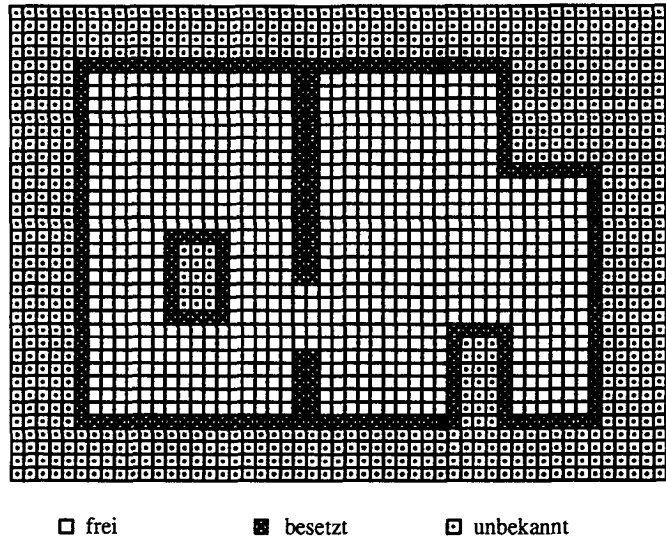
Ein mobiler Roboter bewegt sich in einer Umgebung, die eine Fülle von komplizierten und weniger komplizierten Objekten enthält. Jedes dieser Objekte genau zu modellieren ist in jedem Fall unmöglich und zum Glück auch nicht nötig. Bevor ein Modell gebildet wird, muss man sich sowieso zuerst überlegen, welche Fragen mit diesem Modell beantwortet werden müssen. Meistens zeigt es sich, dass es genügt, nur die wichtigsten Objekte zu modellieren. Eine weitere wesentliche Vereinfachung ergibt sich, wenn statt einer dreidimensionalen eine zweidimensionale Darstellung ausreicht.

Für die Positionsbestimmung genügt ein zweidimensionales Modell, sofern sich der mobile Roboter in einer ebenen strukturierten Umgebung bewegt. Je nach Anordnung der Sensoren ist es sinnvoll, dieses zweidimensionale Modell in mehreren Schichten anzulegen, was eine Zwischenstufe zu einem dreidimensionalen Modell darstellt.

Nachfolgend wird eine Auswahl von Umgebungsmodellen, wie sie in anderen Arbeiten eingeführt wurden, vorgestellt. Dabei lässt sich eine grobe Unterteilung in Rastermodelle und in Polygonmodelle machen. Zu beachten ist, dass die Modelle oftmals nicht primär für die Positionsbestimmung, sondern für Navigation und Wegplanung ausgelegt sind.

3.1.2 Rastermodelle

In einem Rastermodell (siehe Fig. 3.1) wird die Umgebung in quadratische Felder eingeteilt. Diese Felder werden als frei, besetzt oder als unbekannt bezeichnet, je nachdem, ob sich ein Hindernis auf diesem Feld befindet oder nicht ([Crowley 89], [Marce and Julliere 86], [Microbe 85]). Diese Information wird in einer $n \times m$-Matrix abgelegt. Je nach Feinheit der Rasterung und Grösse des Raumes kann diese Matrix beachtliche Dimensionen annehmen, was ungünstige Auswirkungen auf den Speicherbedarf hat. Weniger speicherintensiv ist die Darstellung des Rastermodelles in einer Baumstruktur, einer sogenannten *Quadtree*-Darstellung. Der Spareffekt ergibt sich durch das Zusammenfassen von nebeneinander liegenden Feldern gleicher Art.



Figur 3.1: *Zweidimensionale Rasterdarstellung einer strukturierten Umgebung.*

3.1.3 Polygonmodelle

Ein Polygonmodell ist eine Grundrissdarstellung der Umgebung. Konturen und Hindernisse werden durch n-eckige Polygone dargestellt. Je nach Verwendungszweck wird die Modellinformation in einer Matrix oder in verketteten Listen (z.B. [Chatila and Laumond 85]) abgelegt. Bleibt das Umgebungsmodell unverändert, so ist eine Matrixdarstellung vorzuziehen, da die Operationen am Modell einfacher und effizienter durchgeführt werden können. Muss der Roboter sein Umgebungsmodell selber lernen, so ist eine Darstellung mit verketteten Listen besser, weil dann beliebige Erweiterungen möglich sind. In diesem Fall ist aber auch zu berücksichtigen, dass aus Sensormessungen entstandene Umgebungsmodelle fehlerbehaftet sind.

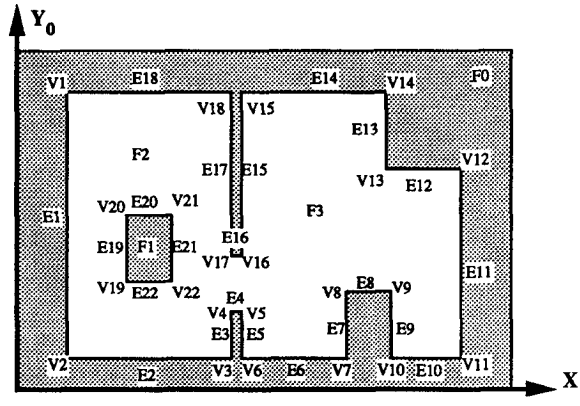
3.2 Das Referenzmodell

Für das Umgebungsmodell wählen wir eine Polygondarstellung, die beschrieben wird durch eine doppelt verbundene Liste (*DCEL*: Fig. 3.2). Diese Darstellung ist eindeutig und hat eine relativ einfache Form. Erweiterungen sind ohne grossen Aufwand möglich. Besondere Vorteile bringt diese Darstellung, wenn effiziente geometrische Algorithmen zu programmieren sind ([Lee and Preparata 84]).

Die wichtigste Komponente der DCEL ist die Kante (Edge). Jede Kante wird durch genau eine Zeile der Tabelle dargestellt. Jede Kantenzeile besteht aus vier Informationsfeldern (*Vertex 1*, *Vertex 2*, *Face 1*, *Face 2)* und zwei Pointerfeldern *(Pointer 1*, *Pointer 2*). Für die Abspeicherung von bestimmten Eigenschaften der Kanten können weitere Felder vorgesehen werden (*Additional Properties*). Die Felder der Kantenzeilen haben die folgenden Bedeutungen:
Das Feld *Vertex 1* enthält den Anfang der Kante und das Feld *Vertex 2* enthält das Ende der Kante. Dadurch erhält die Kante eine Richtung. Interpretiert man diese Kante als Wand, so liegt links dieser Kante immer der freie Raum.
Die Felder *Face 1* und *Face 2* enthalten die Namen der Flächen, die auf der linken bzw. auf der rechten Seite der Kante liegen.
Die Pointerfelder *Pointer 1* bzw. *Pointer 2* zeigen auf die vorhergehende bzw. auf die nachfolgende Kante.

Mit dieser Darstellung ist es einfach, diejenigen Kanten ausfindig zu machen, die eine gewünschte Fläche umschliessen. Auch kann schnell und einfach bestimmt werden, welche Kanten von einem gegebenen Schnittpunkt weglaufen.

Das Modell enthält keine Angabe von Unsicherheiten. Dies bedingt, dass nur solche Elemente darin Platz finden können, welche sich nicht verschieben. Dies sind aber gleichzeitig diejenigen Elemente, die für die Positionsbestimmung überhaupt in Frage kommen.



Vertex	1	2	3		21	22
x	0.91	0.91	4.37		3.11	2.28
y	5.88	0.62	0.62		4.46	4.46

Edge	Vertex 1	Vertex 2	Face 1	Face 2	Pointer 1	Pointer 2	Additional Properties
1	1	2	2	0	18	2	
2	2	3	2	0	1	3	
3	3	4	2	0	2	4	
4	4	5	2	0	3	5	
5	5	6	3	0	4	6	
6	6	7	3	0	5	7	
7	7	8	3	0	6	8	
8	8	9	3	0	7	9	
9	9	10	3	0	8	10	
10	10	11	3	0	9	11	
11	11	12	3	0	10	12	
12	12	13	3	0	11	13	
13	13	14	3	0	12	14	
14	14	15	3	0	13	15	
15	15	16	3	0	14	16	
16	16	17	2	0	15	17	
17	17	18	2	0	16	18	
18	18	1	2	0	17	1	
19	19	20	2	1	22	20	
20	20	21	2	1	19	21	
21	21	22	2	1	20	22	
22	22	19	2	1	21	19	

Figur 3.2: *Zweidimensionale Darstellung einer Umgebung in einer doppelt verbundenen Liste (***DCEL***: Doubly Connected Edge List). In der ersten Tabelle sind die Koordinaten der Punkte V_i angegeben (Geometrische Daten). Die zweite Tabelle enthält Angaben darüber, wie Punkte, Kanten und Flächen miteinander verbunden sind (Topologische Daten).*

3.3 Das sensorbasierte Modell

3.3.1 Sensorsysteme für mobile Roboter

Die Sensoren eines mobilen Roboters sind charakterisiert durch ihre verschiedenartigen Formen und die komplexen Aufgaben, die sie zu bewältigen haben. Eine Kamera z.B. nimmt rohe digitalisierte Bilder einer Umgebung auf. Auf diese Rohdaten können die verschiedensten Algorithmen angesetzt werden, die versuchen, verschiedene Merkmale wie Ecken, Kanten, Oberflächen usw. zu erkennen. Jeder dieser Algorithmen liefert eine andere Interpretation der Sensordaten. Die Vermischung aller Sensordaten und deren Interpretationen geschieht durch eine geeignete Darstellung in einem Modell.

Wir sind an einem Modell interessiert, in welchem geometrische Beobachtungen der Umwelt abgelegt werden können. Dabei ist zu beachten, dass Sensordaten immer mit Fehlern behaftet sind, die berücksichtigt werden müssen. Für die Positionsbestimmung sind vor allem Umgebungsmerkmale interessant, die erstens stationär sind und zweitens eine definierte geometrische Form haben. Im *Sensorbasierten Modell* wird dem Rechnung getragen, indem nur solche Merkmale eingetragen werden. Andere Merkmale werden ignoriert. Dieses Vorgehen lässt sich damit begründen, dass für die Positionsberechnung selber nur sehr wenige Merkmale gemessen werden müssen (siehe Kapitel 2.2.1). Die geometrischen Elemente, die wir verwenden, sind *Kanten*,*Ecken* und *Wände*. Diese lassen sich mit den verschiedensten Sensorsystemen relativ einfach und sicher erkennen und auch lokalisieren.

Die Wahl des richtigen Sensorsystemes ist nicht ganz einfach, weil dabei eine ganze Reihe von Gesichtspunkten gegeneinander abgewogen werden müssen ([Everett 89]):

- *Sichtfeld*: Für die Positionsbestimmung ist es vorteilhaft, wenn das Sichtfeld gross ist, weil so die Genauigkeit der Positionsbestimmung weniger sensitiv ist auf Messfehler.
- *Reichweite*: Je grösser der Raum, umso grösser sollte auch die Reichweite des verwendeten Sensorsystemes sein.
- *Genauigkeit und Auflösung*: Diese hängt von der geforderten

Genauigkeit der Positionsbestimmung ab.

- *Fähigkeit, alle Objekte der Umwelt zu erkennen*: Objekte können die ausgesendete Energie absorbieren oder in eine falsche Richtung reflektieren.

- *Laufzeit*: Die Häufigkeit der durchführbaren Messungen muss mit der Fahrgeschwindigkeit des Roboters im Einklang stehen.

- *Knappheit und Interpretierbarkeit der Daten*: Ein System, welches wenige, aber direkt verwendbare Daten liefert, ist einem System vorziehen, bei welchem die Information erst nach mehreren Verarbeitungsschritten vorliegt.

- *Einfachheit*: Das System sollte einfach zu warten und nicht zu teuer sein.

- *Leistungsaufnahme*: Damit der mobile Roboter möglichst lange ohne Nachladen der Batterien betrieben werden kann, ist ein niedriger Stromverbrauch wichtig.

- *Grösse*: Das System sollte auf einem mobilen Roboter Platz finden.

Je nachdem, ob die Sensoren gleichzeitig für andere Aufgaben wie z.B. Kollisionsverhinderung eingesetzt werden, spielen diese Gesichtspunkte eine unterschiedlich wichtige Rolle.

Für die Positionsbestimmung und für die Beschaffung von geometrischen Daten sind Abstands- und/oder Winkelmessungen wichtig. Vor allem für die Abstandsmessung gibt es eine grosse Auswahl von Möglichkeiten. Der zugehörige Winkel kann meistens aus der Messrichtung abgeleitet werden.

3.3.2 Abstandsmessung

Für die Bestimmung des Abstandes von einer bestimmten Position aus zu einem entfernt liegenden Gegenstand sind die unterschiedlichsten Verfahren bekannt:

- *Triangulation*: Der Ausfallwinkel und der Einfallwinkel eines reflektierten Strahles werden gemessen. Ist der Abstand zwischen Sender und Empfänger bekannt, so kann die Distanz zum reflektierenden Objekt berechnet werden. Die Genauigkeit ist vom gemessenen Abstand abhängig.

- *Messen der Laufzeit*: Die Zeit wird gemessen, die verstreicht, bis ein Signal wieder zum Sender zurückkehrt. Diese ist direkt proportional zur gemessenen Distanz.
  Als Signale werden Ultraschallwellen oder elektromagnetische Wellen verwendet. Die Messung der Laufzeit von Ultraschallwellen ist wesentlich einfacher, da sie relativ langsam sind. Für die Bestimmung der Laufzeit von elektromagnetischen Wellen sind aufwendige Apparaturen notwendig.

- *Verstellbarer Fokus*: Mit einer Fokussiereinrichtung wird eine Kamera scharf eingestellt. Die Einstellung ist ein Mass für die Entfernung des betrachteten Gegenstandes.

- *Messung der Intensität des reflektierten Signales*: Aus dem Verhältnis von gesendeter und empfangener Leistung kann auf den Abstand geschlossen werden.

Diese Aufzählung ist nicht vollständig, sollte aber einen Eindruck geben über die Vielfalt der Möglichkeiten. Für mobile Roboter am besten geeignet sind Verfahren, die mit Triangulation oder mit Laufzeitmessung arbeiten. Die anderen Verfahren sind entweder zu aufwendig, zu ungenau oder funktionieren nur unter bestimmten Bedingungen.

3.3.3 Winkelmessung

Für die Bestimmung des Winkels zu einem gewünschten Punkt sind die folgenden zwei Methoden am gebräuchlichsten:

1. *Auswerten eines Bildes:*
   Nachdem mit einem Erkennungsverfahren der gewünschte Punkt auf dem Bild identifiziert wurde, kann aus der Orientierung der Kamera und aus der Lage des Punktes innerhalb des Bildes der gesuchte Winkel bestimmt werden.

2. *Ausrichten eines Sensors:*
   Ein Sensor wird auf den Punkt ausgerichtet. Die Stellung des Sensors entspricht der gesuchten Richtung.

Für mobile Roboter ist das erste Verfahren eher besser geeignet, da meistens nicht nur ein Winkel, sondern die Winkel zu mehreren Punkten bestimmt werden müssen.

3.3.4 Darstellung

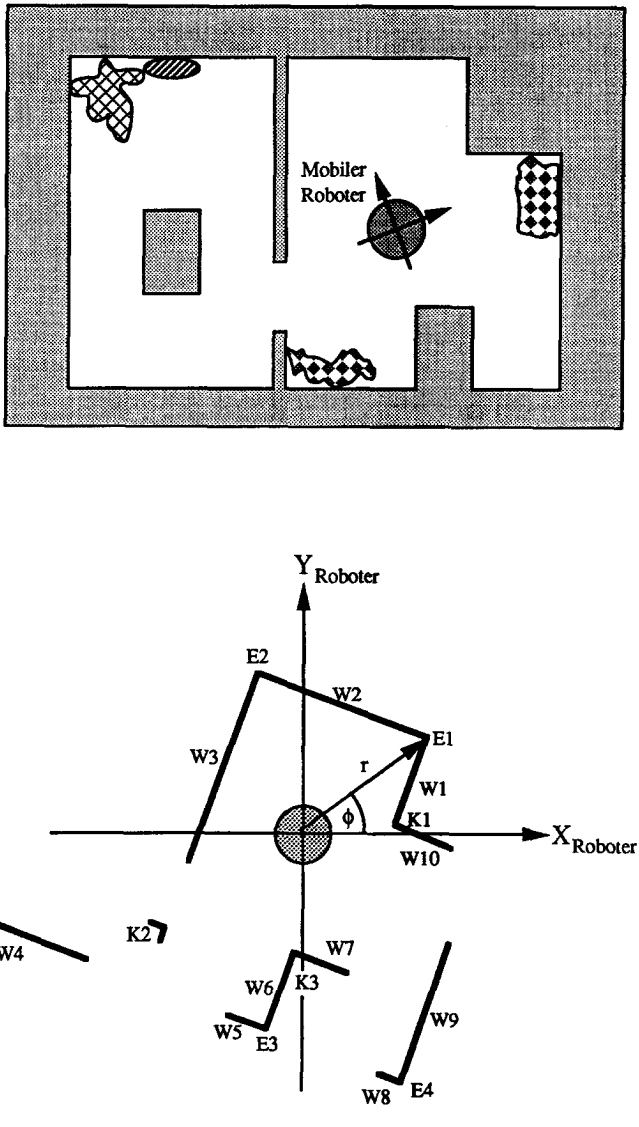
Das *Sensorbasierte Modell* soll alle einfach beschreibbaren geometrischen Elemente enthalten, die im Sichtbereich des mobilen Roboters liegen. In der oberen Hälfte von Figur 3.3 ist der Grundriss eines Raumes abgebildet, in dem sich ein mobiler Roboter aufhält. Nebst diversen Wänden, Kanten und Ecken sind auch Hindernisse mit nicht näher spezifizierbaren Formen vorhanden. Das dazugehörige vollständige *Sensorbasierte Modell* ist in Fig. 3.3 abgebildet. Im wesentlichen entspricht es dem Sichtbarkeitspolygon des Raumes aus der Position des mobilen Roboters, wobei unförmige Hindernisse ausgeklammert sind.

Die Bildung des *Sensorbasierten Modelles* ist problemlos, wenn man annimmt, dass fehlerfreie Abstandsmessungen möglich sind. Führt man solche Messungen vom Roboter aus in alle Richtungen durch, so ergibt deren Eintragung in ein Koordinatensystem automatisch eine Art Sichtbarkeitspolygon. Daraus können die vorgegebenen geometrischen Elemente extrahiert werden.

Die Darstellung erfolgt natürlicherweise in Polarkoordinaten, da alle Messungen vom gleichen Standort aus gemacht werden (Tab. 3.1). Die Position der Objekte wird durch r und ϕ angegeben. Da *Kanten* und *Ecken* Punkte sind, entspricht dies direkt der Position im Raum. Beim Objekt *Wand* hingegen bezeichnet r den Abstand der *Wand* vom Ursprung und ϕ den Winkel von dessen Normale. Diese Darstellung stimmt mit der Geradengleichung in Polardarstellung überein:

$$r - x\cos\phi - y\sin\phi = 0$$

Die Grösse einer *Wand* wird durch die Anfangs- und Endwinkel



Figur 3.3: *Der mobile Roboter befindet sich in einer strukturierten Umgebung. Das vollständige Sensorbasierte Modell ist von dessen Position abhängig und entspricht im wesentlichen dem Sichtbarkeitspolygon.*

Art	Nr	r[m]	ϕ[°]	ϕ_A[°]	ϕ_E[°]	$\sigma_r[m]$	σ_ϕ[°]
K	1	1.5	8	-	-	0.0	0.0
W	1	1.3	-22	8	34	0.0	0.0
E	1	2.5	34	-	-	0.0	0.0
W	2	2.1	68	34	105	0.0	0.0
E	2	2.6	105	-	-	0.0	0.0
W	3	1.6	158	105	187	0.0	0.0
W	4	3.1	248	187	210	0.0	0.0
K	2	2.7	212	-	-	0.0	0.0
W	5	3.1	248	247	260	0.0	0.0
E	3	3.2	260	-	-	0.0	0.0
W	6	0.5	-22	260	268	0.0	0.0
K	3	1.9	268	-	-	0.0	0.0
W	7	1.8	248	268	288	0.0	0.0
W	8	3.1	248	288	291	0.0	0.0
E	4	4.2	291	-	-	0.0	0.0
W	9	2.7	-22	291	325	0.0	0.0
W	10	0.6	68	-3	2	0.0	0.0

Tabelle 3.1: *Vollständiges Sensorbasiertes Modell*

ϕ_A und ϕ_B beschrieben. Für die spätere Verwendung des *Sensorbasierten Modelles* ist auch wichtig zu wissen, wie genau die Angaben sind. Schliesst man systematische Messfehler aus, so können r und ϕ als normalverteilte Zufallsgrössen mit der Messung als Mittelwert betrachtet werden. Deren Standardabweichungen σ_r und σ_ϕ werden deshalb als Mass für die Unsicherheit der Messungen ins Modell eingetragen. In Tab. 3.1 sind die Messungen als vollständig und fehlerfrei angenommen, weshalb σ_r und σ_ϕ immer 0.0 sind.

Eine wichtige Rolle spielt auch die Reihenfolge der Einträge ins *Sensorbasierte Modell*, weil dies die einzige Information über die Topologie der erkannten Objekte ist. Zwar kann nicht gewährleistet werden, dass sich im Modell benachbarte Objekte berühren, aber ihre gegenseitige Lage ist von grosser Bedeutung bei der Positionsfindung.

Leider sind Sensoren nicht ideal, so dass wir annehmen müssen, dass

Art	Nr	r[m]	ϕ[°]	ϕ_A[°]	ϕ_E[°]	$\sigma_r[m]$	σ_ϕ[°]
K	1	1.5	8	-	-	0.03	3.0
E	1	2.5	34	-	-	0.05	4.0
W	1	2.1	68	58	78	0.02	3.0
E	2	2.6	105	-	-	0.05	4.0
W	2	1.6	158	148	168	0.02	3.0
K	2	2.7	212	-	-	0.06	3.0
K	3	1.9	268	-	-	0.04	3.0
W	3	2.7	338	310	320	0.10	10.0

Tabelle 3.2: *Modell aus Ultraschallmessungen*

wir das Sichtbarkeitspolygon nicht vollständig und fehlerfrei erkennen können. Zwei Beispiele sollen dies veranschaulichen.

Modell aus Ultraschallmessungen Abstandsmessungen mit Ultraschallsensoren haben den Nachteil, dass sie mit einem grossen Öffnungswinkel erfolgen, und dass Ultraschallwellen unter schiefen Winkeln wegreflektiert werden können, was eine Messung verunmöglicht. Soll ein Modell allein an Hand von solchen Messungen gebildet werden, so ist in etwa das in Tab 3.2 angegebene Resultat zu erwarten.

Gut erkannt werden können demnach *Kanten*, während schief liegende *Wände* und versteckte *Ecken* nur schwierig zu detektieren sind. Die Numerierung der einzelnen Objekte stimmt nicht mehr mit dem vollständigen Modell überein.

Modell aus Bildinformation Aus einem einzelnen Bild allein können nur Winkelangaben entnommen werden. Um daraus weitere Informationen herauslesen zu können, ist entweder ein zweites Bild oder ein Referenzbild notwendig. Dies bedeutet, dass Abstände aus einem Einzelbild nicht bestimmt werden können. Das zu erwartende Modell ist in Tab 3.3 dargestellt. *Kanten* und *Ecken* können nicht unterschieden werden. *Wände* werden keine erkannt. Abstandsinformationen fehlen vollständig.

Art	Nr	r[m]	ϕ[°]	ϕ_A[°]	ϕ_E[°]	$\sigma_r[m]$	σ_ϕ[°]
K/E	1	?	8	-	-	-	1.0
K/E	2	?	34	-	-	-	1.0
K/E	3	?	105	-	-	-	1.0
K/E	4	?	187	-	-	-	1.0
K/E	5	?	210	-	-	-	1.0
K/E	6	?	212	-	-	-	1.0
K/E	7	?	260	-	-	-	1.0
K/E	8	?	268	-	-	-	1.0
K/E	9	?	288	-	-	-	1.0
K/E	10	?	291	-	-	-	1.0

Tabelle 3.3: *Modell aus Bildinformation*

Kapitel 4

Bestimmung der Anfangsposition

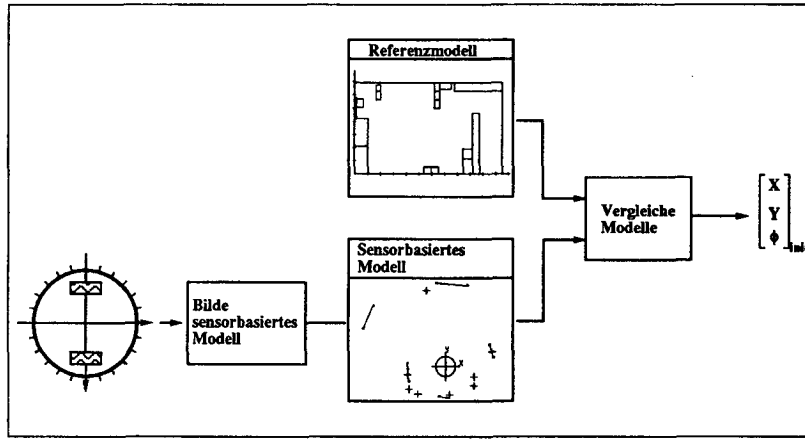
4.1 Problemstellung

Ein mobiler Roboter befinde sich irgendwo in einer strukturierten Umgebung. Davon kenne er ein geometrisches Referenzmodell. Anhand von Winkel- und/oder Abstandsmessungen soll er seine Position und Orientierung bestimmen. Dieses Problem wollen wir in zwei Schritten lösen (Fig 4.1):

1. Bilden eines *Sensorbasierten Modelles*, welches die Winkel- und/oder Abstandsmessungen geeignet darstellt.

2. Vergleich des *Sensorbasierten Modelles* mit dem vorgegebenen *Referenzmodell* der Umgebung.

Wie das *Sensorbasierte Modell* geeignet dargestellt werden kann, wurde im vorherigen Kapitel besprochen. Wie es im konkreten Fall erzeugt wird, ist sensorabhängig. Wir haben uns für Ultraschallsensoren entschieden, weil sie kostengünstig sind und relativ genaue Abstandsmessungen liefern. Gleichzeitig eignen sie sich auch als Sensoren für



Figur 4.1: *Bestimmung der Anfangsposition*

die Kollisionsverhinderung. Nachteilig sind die lange Messdauer wegen der niedrigen Geschwindigkeit der Schallwellen, sowie ihr grosser Öffnungswinkel. Vor allem letzteres erweist sich bei der Erkennung von Umgebungsobjekten als erschwerend.

In einem zweiten Schritt wird das *Sensorbasierte Modell* mit dem *Referenzmodell* verglichen, d.h. die beiden Modelle sind so gegeneinander zu verschieben, dass eine möglichst gute Übereinstimmung resultiert. Dabei gibt es folgende Schwierigkeiten:

- Vom Standort des Roboters aus kann nicht die ganze Umgebung eingesehen werden, weshalb nur ein Teil davon modelliert werden kann.
- Hindernisse werden im *Referenzmodell* nicht berücksichtigt, erscheinen aber im *Sensorbasierten Modell.*
- Das *Sensorbasierte Modell* muss aus fehlerbehafteten Messungen gebildet werden.

Natürlich kann die Position des Roboters nur dann bestimmt werden, wenn eine minimale Anzahl von Objekten aus dem *Sensorbasierten Modell* auch im *Referenzmodell* enthalten ist. Wir wollen annehmen,

dass vom momentanen Standort des Roboters aus mindestens **drei** Objekte erkannt werden können, die im *Referenzmodell* enthalten sind. Die restlichen Objekte dürfen Hindernisse sein. Kann diese Bedingung nicht erfüllt werden, so muss der Roboter einen günstigeren Standort aufsuchen.

4.2 Bildung des sensorbasierten Modelles aus Ultraschallmessungen

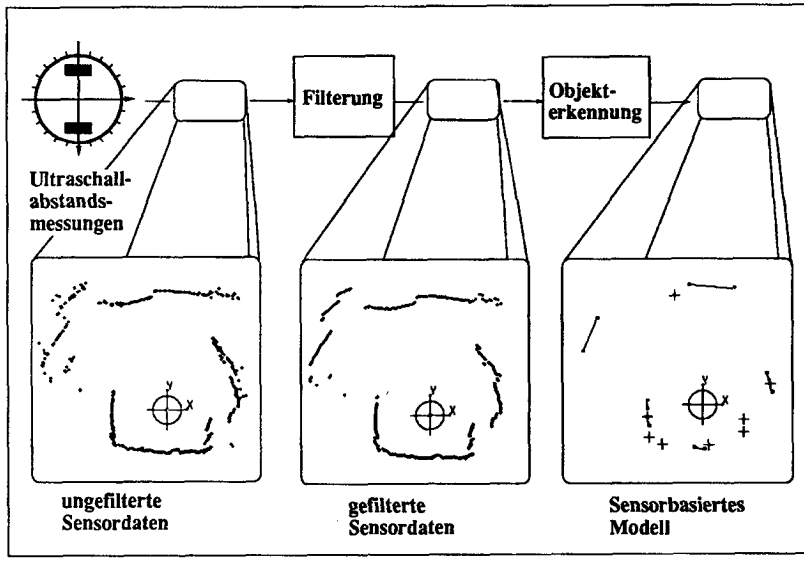
4.2.1 Einführung

Für die Bildung des *Sensorbasierten Modelles* sind Abstandsmessungen in mehrere Richtungen notwendig. Erfolgen diese von einem festen Standort aus, so erhält man ein Bild der Umgebung (Fig. 4.2). Dieses bezeichnen wir in der Folge als *Ultraschallbild*. Daraus können geometrische Objekte extrahiert und in das Modell eingetragen werden.

Das Finden von Objekten aus einem Ultraschallbild ist eine typische Aufgabe der Objekterkennung. Diesem Thema werden wir uns deshalb im nächsten Unterkapitel zuwenden. Dort werden wir sehen, dass die Lösung dieser Aufgabe wesentlich von den verfügbaren Sensordaten abhängt. Dies führt uns zur Analyse der von uns verwendeten Ultraschallmessungen. Die daraus gefundenen Unterscheidungsmerkmale werden anschliessend für die Erkennung von Objekten und deren Darstellung im *Sensorbasierten Modell* benutzt.

4.2.2 Objekterkennung

Das Erkennen von Objekten der Umwelt kann als spezieller Problemlösungsprozess interpretiert werden. Sein Ziel ist es, vorliegende Umweltsituationen bzw. Objekte gleicher Art jeweils einer Gruppe bzw. Klasse mit definierten Eigenschaften zuzuordnen. Die drei Phasen eines allgemeinen Problemlösungsprozesses treffen prinzipiell auch für jeden Erkennungsprozess zu. Diese Phasen enthalten lediglich speziellere Aufgabenstellungen:



Figur 4.2: *Die Bildung des Sensorbasierten Modelles lässt sich in die Teilschritte Datenaufnahme, Filterung und Objekterkennung unterteilen.*

1. Erfassen von charakteristischen Messgrössen der zu erkennenden Umweltsituationen bzw. Objekte

2. Analysieren der gemessenen charakteristischen Grössen

3. Zuordnen einer Umweltsituation bzw. eines Objektes zu der Gruppe bzw. Klasse, der es auf Grund der vorgenommenen Analyse mit der grössten Wahrscheinlichkeit entstammt.

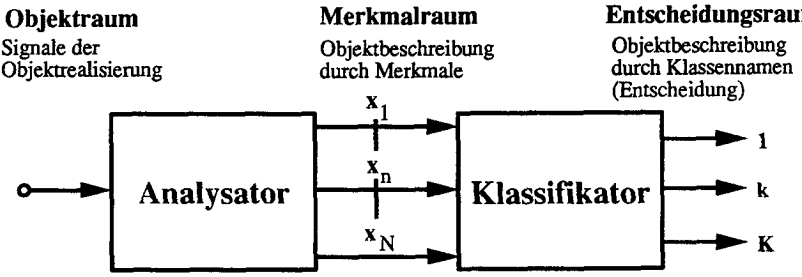
Problemmodell der Erkennung

Bei der Lösung von Erkennungsproblemen hat sich ein Modell als zweckmässig erwiesen ([Steinhagen und Fuchs 76]), welches zwei grundlegende Transformationen - Analyse und Klassifikation - enthält, die zwischen Objekt- und Merkmalraum bzw. zwischen Merkmal- und Entscheidungsraum wirken (Figur 4.3). Dieses Modell genügt der

gegebenen Abgrenzung des Erkennungs- vom allgemeineren Beschreibungsproblem und auch der Definition des Erkennungsprozesses. Die Struktur des Modells ist rückkopplungsfrei, womit kein Lernen möglich ist. Das ist eine Einschränkung der Allgemeingültigkeit, vor allem gegenüber leistungsfähigen biologischen Systemen. Damit ist jedoch eine klare Darstellung der Grundfunktionen möglich.

Hauptprobleme der Objekterkennung

Das **Merkmalproblem** beinhaltet die Frage, welcher Art die Merkmale sein sollen und wie viele zu verwenden sind. In der Regel kann man "nahezu vollständige" Objektbeschreibungen für jede Objektart angeben. Allerdings führt dies meist auf sehr grosse Merkmalsvektoren, die eine Klassifikation sehr erschweren. Es sind deshalb besser nur wenige Merkmale zu definieren, die aber trotzdem eine eindeutige Klassifikation ermöglichen. Daneben sollten diese Merkmale auch robust sein gegenüber kleinen Veränderungen.
Die Frage nach dem Aufbau des Klassifikators, das **Klassifikationsproblem**, lässt sich in zwei Teilprobleme aufspalten: das Entscheidungsproblem und das Lernproblem. Das *Entscheidungsproblem* ist die Frage nach der Struktur des Klassifikators. Diese wird durch die geometrische Struktur der in den Merkmalraum abgebildeten Gesamtheit der Objektrealisierungen bestimmt. Schwierigkeiten bei der Lösung des Problems entstehen meist dadurch, dass nicht genügend Kenntnisse über diesen Raum vorliegen. Das *Lernproblem* in bezug auf den Klassifikator ist die Frage nach den Parametern der Unterscheidungsfunktionen, d.h. nach dem Referenzwissen des Klassifikators. Es hat konkrete Zahlenwerte zur Lösung, die aus der Statistik der Objektrealisierungen zu gewinnen sind. Für das Klassifikationsproblem ist durch die statistische Entscheidungstheorie und die statistischen Schätz- und Lernmethoden ein optimaler, allerdings aus Aufwandsgründen oft nicht gangbarer Weg bekannt. Daneben existiert eine Vielzahl von Näherungslösungen.



Figur 4.3: *Grundmodell der Erkennung*
Objektraum: *Das Objekterkennungssystem ist eingangsseitig an die Umwelt gekoppelt. Da wir eine spezielle Erkennungsaufgabe lösen wollen, ist nur derjenige Teil der Umwelt von Interesse, der die entsprechende Objektart enthält. Diesen Teil bezeichnet man als Objektraum. Die Objekte sind dem Erkennungssystem durch Objektsignale zugänglich.*
Analysator: *Die für das System interessanten Signale werden vom Analysator aufgenommen und in eine Menge zahlenmässiger Grössen $\{x_n\}$ transformiert. Die Transformationsfunktionen können als Messvorschriften interpretiert werden, die auf die Signalfunktionen angewendet werden.*
Merkmalraum: *Die Menge der möglichen Merkmalwerte in einem Objekterkennungssystem wird als Merkmalraum bezeichnet. Für das Verständnis erweist es sich als vorteilhaft, diesen Raum als geometrischen N-dimensionalen Raum mit euklidischer Metrik zu interpretieren. Jedes Merkmal bildet dann eine Koordinatenrichtung. Die Menge der Merkmalwerte einer Objektrealisierung, d.h. ein Merkmalvektor, entspricht dem Ortsvektor zu einem bestimmten Punkt des Merkmalraums.*
Klassifikator: *Jeder Objektrealisierung einen bestimmten Klassennamen zuzuordnen verbleibt als Aufgabe für den Klassifikator. Diesem liegen die Objektrealisierungen in interner Beschreibung vor, d.h. als Merkmalsvektoren. Seine Aufgabe ist deshalb interpretierbar als Zuordnung eines Klassennamens zu jedem Punkt des Merkmalraums.*
Entscheidungsraum: *Die Menge der möglichen Entscheidungen über die Klassenzugehörigkeit ist der Entscheidungsraum. Die Anzahl der Entscheidungen ist in der Regel gleich der durch das innere Modell festgelegten Klassenanzahl K. Oft ist es sinnvoll, als zusätzliche Entscheidung die Rückweisung einzuführen, um unsicher klassifizierbare Objekte nicht irgend einer Klasse zuordnen zu müssen.*

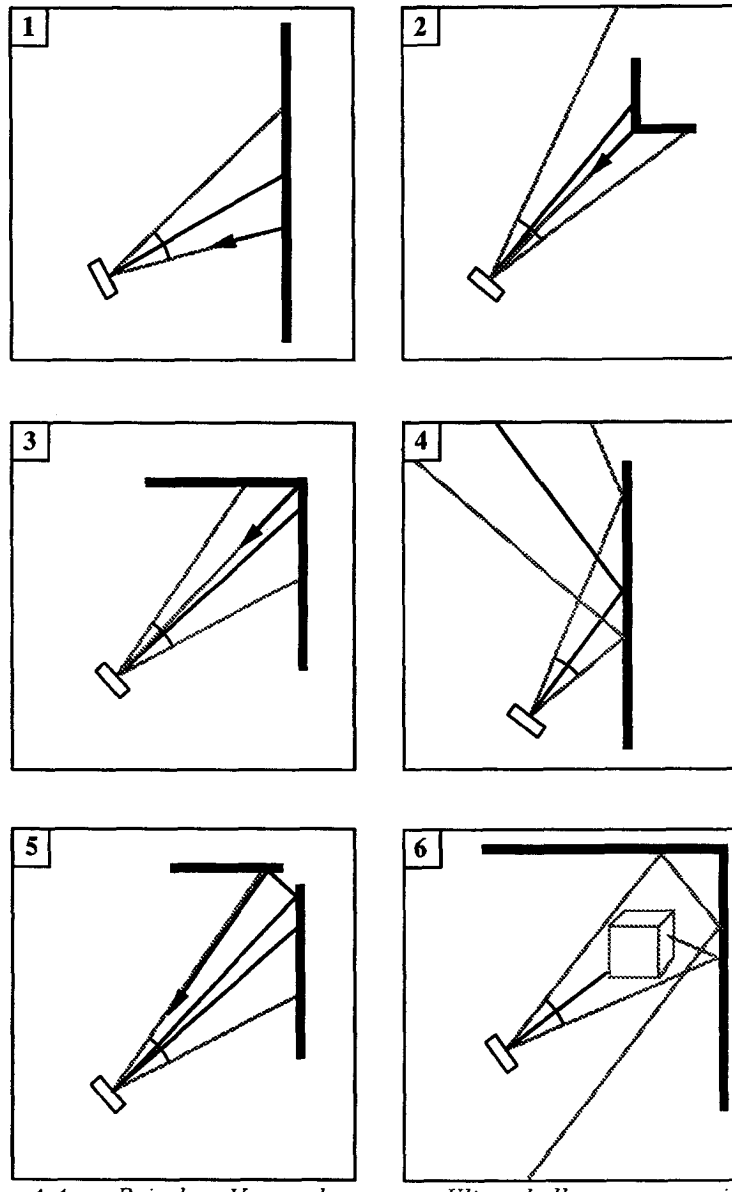
4.2.3 Ultraschallabstandsmessungen

Physikalische Grundlagen

Für die akustische Distanzmessung kommen entweder Triangulationsverfahren oder die Messung der Laufzeit zur Anwendung. Meistens werden dazu gepulste Schallwellen ausgesendet, deren Frequenz typischerweise grösser als 20kHz ist.
Die Leistung von Ultraschallsystemen wird wesentlich beeinflusst von der Konstruktion der Ultraschallsensoren und von den Eigenschaften der Umgebung. Die wichtigste Eigenschaft ist die Verminderung der Schalleistung über die Distanz. Bei der Fortpflanzung einer Schallwelle verringert sich deren Energiedichte entsprechend dem umgekehrten Quadrat der zurückgelegten Distanz. Ein Teil der Energie wird zudem von der Luft absorbiert, was von der Luftfeuchtigkeit, von der Luftverschmutzung und vor allem auch von der Frequenz der Schallwelle abhängig ist. Absorption kann auch beim angepeilten Objekt auftreten und ist eine Funktion von dessen Oberflächenbeschaffenheit. Aus all diesen Eigenschaften ergibt sich, dass die Reichweite eines Ultraschallsensors im wesentlichen von der gesendeten Energie und von der Sendefrequenz abhängt: je tiefer die Frequenz, desto grösser die Reichweite.
Ein anderer wichtiger Parameter, der durch die Eigenschaften der Luft beeinflusst wird, ist die Schallgeschwindigkeit. Diese hängt von der Lufttemperatur und von den Luftbewegungen ab, welche in unserem Fall aber vernachlässigt werden können. Die Schallgeschwindigkeit ändert sich proportional zur Quadratwurzel der Temperaturveränderung. Dies ergibt die einfache Formel

$$wahreDistanz = R_a = R_m \sqrt{\frac{T_a}{T_c}} \tag{4.1}$$

mit R_m gleich der gemessenen Distanz, T_a gleich der gemessenen Temperatur in Grad Kelvin und T_c gleich der Referenztemperatur. Allerdings besteht die Möglichkeit, dass die Temperatur in der Nähe des Sensors nicht gleich hoch ist wie in der Umgebung des reflektierenden Objektes.
Eine weitere wichtige zu betrachtende Grösse ist der Öffnungswinkel des Ultraschallsensors. Dieser ist abhängig vom Durchmesser des Sensors und von der benutzten Frequenz. Je höher die Frequenz ist, umso



Figur 4.4: *Bei der Verwendung von Ultraschallmessungen sind die Auswirkungen von Divergenz(1), Fokussierung(2,3), Reflektion(4), Mehrfachreflektion(5) und Absorption(6) der Signale zu berücksichtigen. Die Pfeile bezeichnen dabei den Weg des in Richtung Sensor reflektierten Ultraschallpulses.*

gebündelter ist die gesendete Schallwelle und umso höher ist die Winkelauflösung. Leider bedeutet aber eine Erhöhung der Frequenz wie gesagt eine Verringerung der Reichweite. Der Öffnungswinkel ist direkt proportional zur gesendeten Wellenlänge:

$$\Theta = 1.22 * \lambda / D \tag{4.2}$$

Θ ist der gewünschte Öffnungswinkel, λ ist die Wellenlänge und D ist der Durchmesser des Senders. Dies ergibt z.B. für den von uns verwendeten Sensor einen Öffunungswinkel von

$$\Theta = 1.22 * \frac{343\frac{m}{s}}{50kHz} / 3.5cm = 0.24 \ (\hat{=}13.7°)$$

Eigenschaften

In Figur 4.4 sind einige weitere wichtige Eigenschaften von Ultraschalldistanzmessungen und deren Auswirkungen zusammengefasst. Daraus lässt sich erkennen, dass man die besten Resultate erhält, wenn die Messrichtung senkrecht zur Oberfläche des Objektes steht. Ist dies nicht der Fall, so entspricht die gemessene Distanz nicht mehr dem Abstand in Richtung der Sensormittellinie, sondern der Distanz zu demjenigen Teil des Objektes, der am nächsten zum Sensor steht und von dem gerade noch ein Echo empfangen werden kann (Figur 4.4.1). Weitere Auswirkungen des grossen Öffnungswinkels sind in Figur 4.4.2 und in Figur 4.4.3 dargestellt. Im ganzen Öffnungsbereich wird nicht die tatsächliche Distanz, sondern die Distanz zum nächstgelegenen Punkt innerhalb dieses Bereiches gemessen. Figur 4.4.4 zeigt, dass überhaupt nichts mehr gemessenen wird, wenn der Einfallswinkel der Ultraschallsignale zu flach ist, weil dann alle Schallwellen wegreflektiert werden. Unter gewissen Umständen können auch Mehrfachreflektionen auftreten (Figur 4.4.5), was zu weiteren Fehlmessungen führt. Besonders schwierig sind diejenigen Fälle zu behandeln, wo die Absorption so stark ist, dass keine Schallwellen mehr reflektiert werden (Figur 4.4.6).

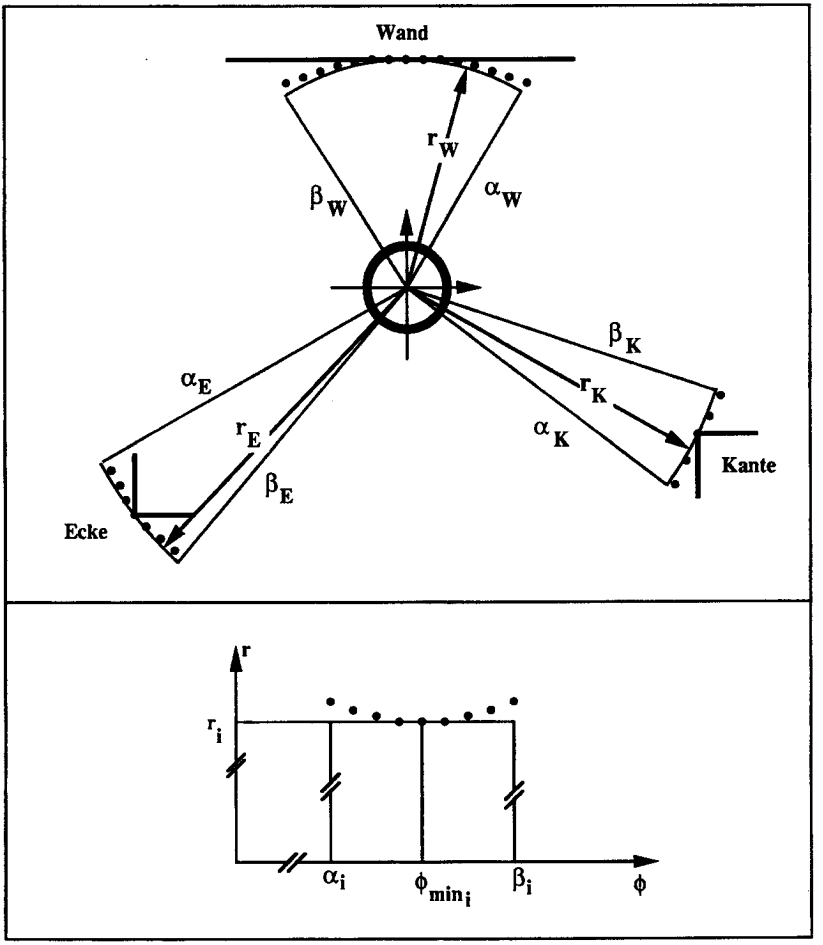
4.2.4 Ultraschallbild der elementaren Umgebungsobjekte

Um aus Ultraschallbildern einzelne Objekte erkennen zu können, müssen wir zuerst untersuchen, wie ein solches Objekt im Ultraschallbild aussieht (Fig. 4.5). Eine *Kante* erscheint darin als kleiner Kreisbogen, dessen Ausdehnung vom Öffnungswinkel der Messung abhängt (Fokussierung: Fig 4.4.2). Sehr ähnlich sieht das Bild einer *Ecke* aus (Fokussierung: Fig 4.4.3). Allerdings ist dieses Bild oftmals nicht so ideal, weil je nach Beschaffenheit und Anstellwinkel der angrenzenden Wände Mehrfachreflektionen oder andere Störungen auftreten können. Ebenfalls wenig unterscheidet sich das Bild einer *Wand*. Die Grösse des Kreisbogens hängt hier aber von deren Oberflächenbeschaffenheit ab.

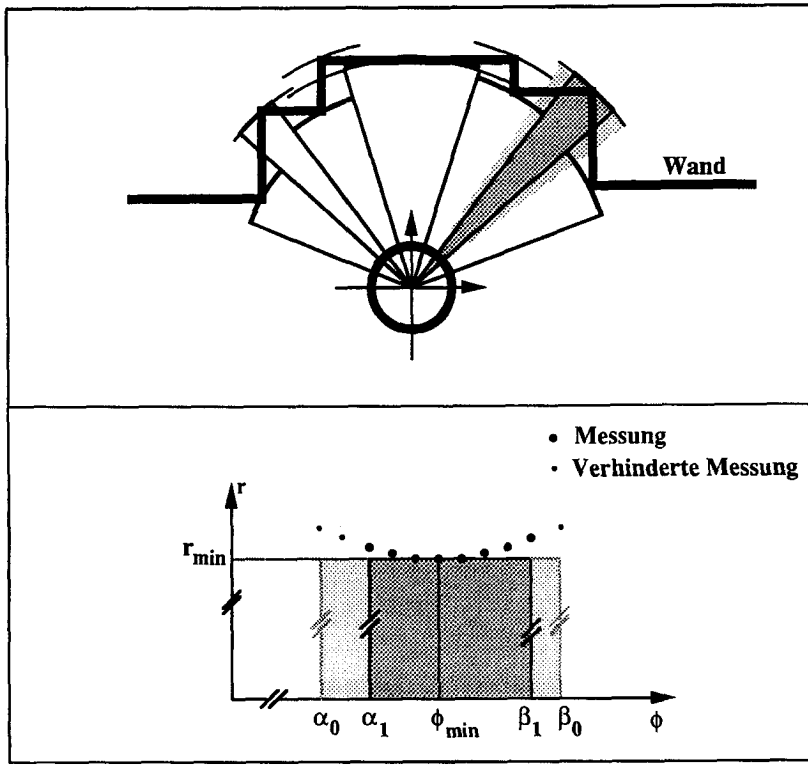
In strukturierten Umgebungen gibt es kaum Einzelobjekte. Vielmehr sind *Ecken*, *Kanten* und *Wände* Teil eines ganzen Raumes. Dies hat Auswirkungen auf deren Ultraschallbild (Fig 4.6). Als Einzelobjekt wäre dieses begrenzt durch α_0 und β_0. Da es aber im Schatten eines anderen Objektes liegt, ist es nur in den Grenzen α_1 und β_1 sichtbar. Wichtig ist auch die Grösse ϕ_{min}, welche die exakte Berechnung der Position des Objektes erlaubt. Im allgemeinen ist ϕ_{min} ungefähr gleich $\frac{1}{2}(\alpha_0 + \beta_0)$. Daraus können Rückschlüsse auf die tatsächlichen Objektgrenzen gezogen werden.

4.2.5 Erkennung der Umgebungsobjekte

Das Erkennen von Umgebungsobjekten soll als klassisches Objekterkennungsproblem gelöst werden (Kap.4.2.2). Dabei bedienen wir uns des zuvor eingeführten Grundmodelles der Erkennung (Figur 4.3). Als Objektklassen verwenden wir die für das *Sensorbasierte Modell* vorgesehenen geometrischen Elemente, nämlich *Kanten*, *Ecken* und *Wände*. Alle anderen geometrischen Elemente werden in einer vierten Klasse vereinigt. Dies erlaubt uns, nur sicher erkannte geometrische Elemente der entsprechenden Klasse zuordnen zu müssen. Während die Definition der Objektklassen im wesentlichen unabhängig ist von den Signalen der Objektrealisierung, ist die Lösung der weiteren Aufgaben wesentlich von den vorhandenen Sensordaten abhängig. Die Ausführungen in diesem Unterkapitel gelten deshalb nur für das von uns verwendete Ultraschallmesssystem ([Albatros Ultrasonic Network]).



Figur 4.5: *Im oberen Teil der Figur sind die Ultraschallbilder der drei Grundobjekte Wand, Kante und Ecke eingetragen. In der ϕ-r-Ebene sind diese weitgehend identisch.*

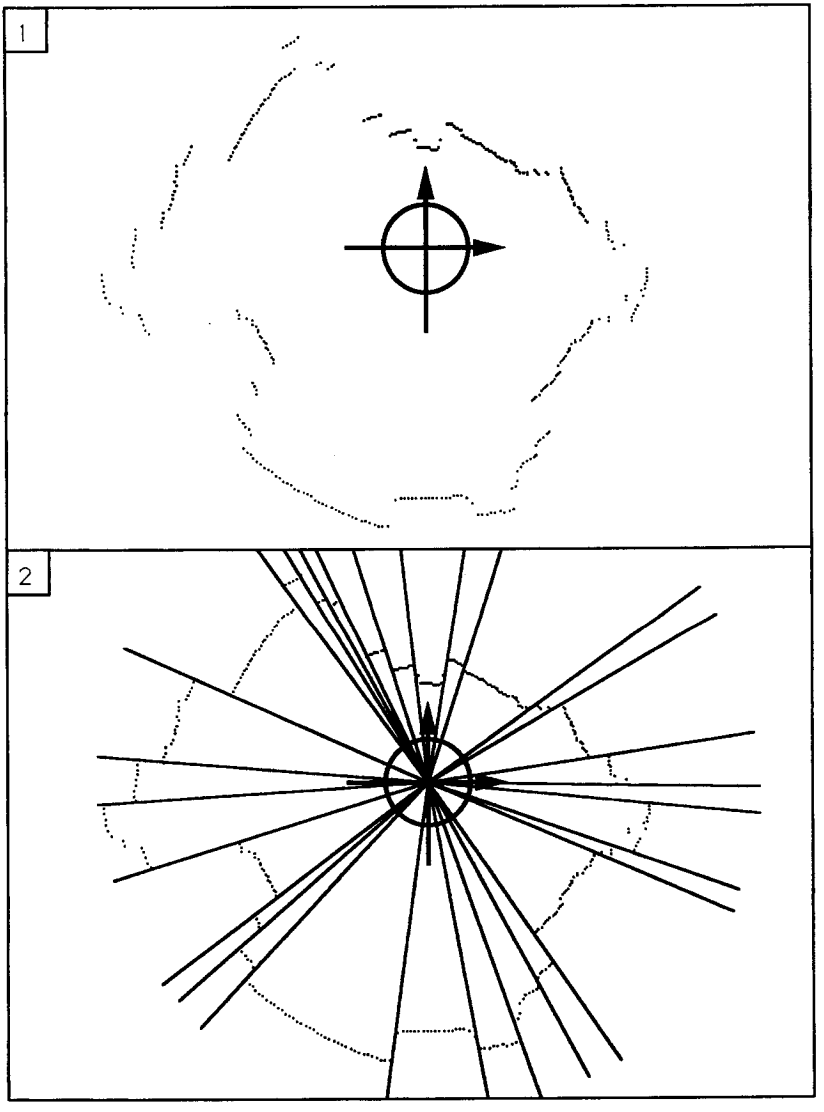


Figur 4.6: *Veränderung des elementaren Ultraschallbildes durch benachbarte Objekte.*

Bestimmung der Objektgrenzen

Ein Ultraschallbild beinhaltet mehrere Objekte, während diese bei klassischen Objekterkennungsaufgaben einzeln vorliegen. Unsere erste Aufgabe besteht also darin, die Grenzen zwischen den einzelnen Objekten zu finden.

In Figur 4.7 ist als Beispiel ein gefiltertes und normalisiertes Ultraschallbild dargestellt. D.h. in diesem Bild liegt pro Winkeleinheit genau eine Abstandsmessung vor und einzelne Ausreisser wurden durch den Abstandswert ihres Nachbarn ersetzt. Nun geht es darum, Punk-



Figur 4.7: *Aufteilung eines Ultraschallbildes in einzelne Objekte.*

tegruppen zu finden, die zum selben geometrischen Objekt gehören. Eine von Hand vorgenommene Gruppeneinteilung ist im zweiten Teil des Bildes angegeben. Teilweise ist diese Einteilung eindeutig, teilweise bekommt man aber je nach Strategie auf ein anderes Resultat. Für das *Sensorbasierte Modell* sind nur die eindeutig extrahierbaren Objekte interessant.

Für die automatische Gruppeneinteilung könnten folgende zwei Methoden verwendet werden:

- Gradientenverfahren:
  Abstandsmessungen, die zur selben Gruppe gehören, haben ungefähr den gleichen Wert. Ist der Unterschied zwischen zwei benachbarten Messungen grösser als eine vorgegebene Höchstgrenze, so können diese beiden Messungen nicht zur selben Gruppe gehören.
- Split-and-Merge-Algorithmus ([Pavlidis 73]:
  Die Abstandsmessungen werden als Punkte in der r-ϕ-Ebene eingetragen und durch möglichst wenige Geradenstücke approximiert. Die endgültige Einteilung der Geradenstücke wird iterativ bestimmt. Überschreitet ein vorgeschlagenes Geradenstück eine vorgegebene Fehlernorm, so wird es aufgeteilt (Split), unterschreitet es diese Norm, so wird versucht, es mit einem benachbarten Stück zu verbinden (Merge). Dies wird so lange versucht, bis keine weiteren Umverteilungen mehr möglich sind.

Bei stark verrauschten Daten ist das zweite Verfahren dem ersten vorzuziehen. In unserem Fall lohnt sich dieser Mehraufwand aber nicht, weil die Messungen bereits vorgefiltert sind.

Beim ersten Verfahren muss als einziger Parameter die maximale Differenz zwischen zwei Messungen angegeben werden. Sein Wert muss experimentell bestimmt werden. Wir haben mit einer Abstandsdifferenz von 2cm die besten Resultate erhalten.

Definition von Merkmalen

Sind die Objekte separiert, so müssen sie anhand von charakteristischen Merkmalen klassiert werden. Nach Analyse der Figuren 4.5 und 4.7

ergeben sich folgende vielversprechenden Merkmale:

- Breite des Objektes
- Varianz der Messpunkte
- Krümmung der die Messpunkte verbindenden Kurve
- Abstand vom Roboter
- Varianz der Randpunkte
- Relative Lage zu den benachbarten Objekten

Die Bestimmung dieser Merkmale aus den Sensordaten ist mathematisch eindeutig definiert und deshalb einfach. Die Auswertung von mehreren Ultraschallbildern hat aber ergeben, dass sie sich für die Objekte *Kante*, *Ecke* und *Wand* kaum unterscheiden, was die nachfolgende Klassifikation wenn nicht verunmöglicht, so doch sehr erschwert. Für die Modellbildung ist aber vor allem die Unterscheidung wichtig, ob es sich beim untersuchten Objekt überhaupt um ein modellierbares Objekt handelt. Dafür genügen die beiden Merkmale *Breite* und *Relative Lage*. Diese werden so definiert (siehe auch Fig. 4.6):

$$\begin{array}{lll} \mathit{Objektbreite}: & s_{Obj} & \in \{1°..90°\} \\ \mathit{RelativeLage}: & p_{rel} & \in \{\mathit{vorne}, \mathit{mitte}, \mathit{hinten}\} \end{array}$$

Berechnung der relativen Lage p_{rel}, wobei der Index i die Reihenfolge der Objekte innerhalb eines Ultraschallbildes bezeichntet:

$$p_{rel} = \begin{cases} \mathit{vorne} & \text{wenn } (r_{min_i} < r_{min_{i-1}}) \wedge (r_{min_i} < r_{min_{i+1}}) \\ \mathit{hinten} & \text{wenn } (r_{min_i} > r_{min_{i-1}}) \wedge (r_{min_i} > r_{min_{i+1}}) \\ \mathit{mitte} & \text{sonst} \end{cases}$$

Berechnung der Objektgrösse s_{Obj}:

$$s_{Obj} = \begin{cases} \beta_1 - \alpha_1 & \text{wenn } p_{rel} = \mathit{vorne} \\ 2 * max(\phi_{min} - \alpha_1, \beta_1 - \phi_{min}) & \text{sonst} \end{cases}$$

Es gibt auch Ansätze, durch Variation der Ultraschallfrequenz ([Kuc 87]) oder durch Messen von mehreren Standorten aus ([Brown 85]), sicherere Merkmale für die Objekterkennung zu finden. Unser Ultraschallsystem arbeitet mit einer festen Schallfrequenz. Standortwechsel sind ungünstig, weil der dabei gemachte Positionsfehler mitberücksichtigt werden muss.

Klassifikation der Objekte

Anhand der beiden Merkmale s_{Obj} und p_{rel} sollen die Objekte in die Klassen *Kante*, *Ecke*, *Wand* oder *Unbekannt* eingeteilt werden.

$$Objektklassen : k_{Obj} \in \{Kante, Wand, Ecke, Unbekannt\}$$

Einteilung in Klassen:

$$k_{Obj} = \begin{cases} Kante & \text{wenn } (10° < s_{Obj} < 17°) \wedge (p_{rel} \neq hinten) \\ Ecke & \text{wenn } (10° < s_{Obj} < 18°) \wedge (p_{rel} = hinten) \\ Wand & \text{wenn } (15° < s_{Obj} < 90°\ ) \\ Unbekannt & \text{sonst} \end{cases}$$

In einzelnen Fällen kann ein Objekt nicht einer einzigen Klasse zugeteilt werden, weil wir keine Merkmale zu deren Unterscheidung kennen. Dann wird so weitergefahren, als ob zwei Objekte erkannt worden wären.

4.2.6 Bildung des Sensorbasierten Modelles

Sind die gemessenen Objekte klassifiziert, so müssen noch deren geometrische Daten berechnet werden, bevor sie ins *Sensorbasierte Modell* eingetragen werden können. Die Angabe der Unsicherheiten σ_r und σ_ϕ basiert auf Werten, die wir bei unserem Ultraschallsystem gemessen haben.

Kante:

$$\begin{aligned} r_K &= r_{min} \\ \phi_K &= \phi_{min} \\ \sigma_{r_K} &= \begin{cases} 0.01m & \text{wenn } p_{rel} = vorne \\ 0.03m & \text{wenn } p_{rel} = mitte \end{cases} \\ \sigma_{\phi_K} &= \begin{cases} 1° & \text{wenn } p_{rel} = vorne \\ 3° & \text{wenn } p_{rel} = mitte \end{cases} \end{aligned}$$

Ecke:

$$\begin{aligned} r_E &= r_{min} \\ \phi_E &= \phi_{min} \\ \sigma_{r_E} &= 0.03m \\ \sigma_{\phi_E} &= 3° \end{aligned}$$

Wand:

$$
\begin{aligned}
r_W &= r_{min} \\
\phi_W &= \phi_{min} \\
\sigma_{r_W} &= \begin{cases} 0.01m & \text{wenn } p_{rel} = vorne \\ 0.03m & \text{sonst} \end{cases} \\
\sigma_{\phi_W} &= \begin{cases} 1^\circ & \text{wenn } p_{rel} = vorne \\ 3^\circ & \text{sonst} \end{cases} \\
\phi_A &= \phi_{min} - \tfrac{1}{2} s_{Obj} \\
\phi_E &= \phi_{min} + \tfrac{1}{2} s_{Obj}
\end{aligned}
$$

Bei der Eintragung der Objekte in die Liste des Modelles ist die Reihenfolge der Objekte beizubehalten.

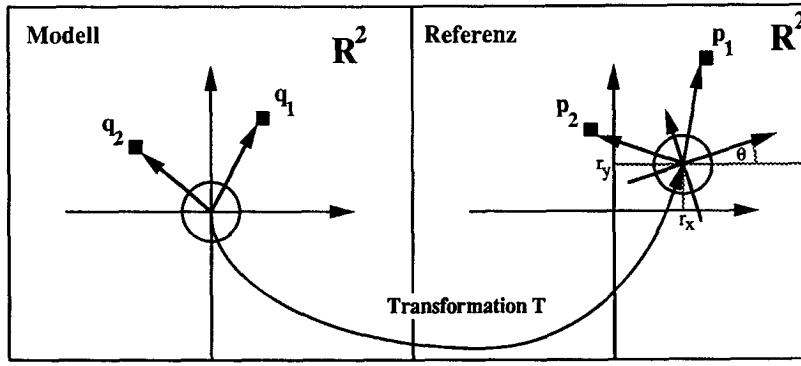
4.3 Vergleich von unvollständigen und fehlerhaften geometrischen Modellen

Wie können zwei aus verschiedenen Positionen aufgenommene geometrische Modelle, die die selbe Umgebung darstellen, aufeinander abgebildet werden? D.h. wie muss das erste Modell transformiert werden, dass es sich mit dem zweiten deckt? Was geschieht, wenn die Modelle unvollständig und fehlerhaft sind? Und wie verhält sich der Lösungsaufwand in Funktion der Anzahl Objekte? Die Beantwortung dieser Fragen ist der Inhalt dieses Abschnittes. Das dabei verwendete Material stammt zu grossen Teilen von [Baird 84].

4.3.1 Abbildungsgeometrie

Die Transformation **T** sei eine eineindeutige Abbildung einer zweidimensionalen Ebene $\mathbf{R}^2$ auf sich selber und besteht aus einer Translation, einer Rotation und einer positiven Skalierung (Fig. 4.8).
Transformationen können auf zwei verschiedene Arten dargestellt werden, wobei in beiden Fällen eine Parametrisierung in vier unabhängigen Variabeln erfolgt.

Trigonometrische Parametrisierung Die vier reellen Parameter r_x, r_y, s und θ ($s > 0$) beschreiben eine Transformation **T**, die auf



Figur 4.8: *Transformation T*

einen Punkt $\mathbf{q} = \begin{bmatrix} q_x \\ q_y \end{bmatrix}$ wie folgt wirkt:

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} \equiv \mathbf{T}(\mathbf{q}) \equiv \begin{bmatrix} r_x \\ r_y \end{bmatrix} + s * \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} q_x \\ q_y \end{bmatrix}.$$

Dies ist eine offensichtliche Darstellung: r_x und r_y sind Verschiebungen, s ist ein Skalierungsfaktor und θ ist der Winkel der Verdrehung um den Ursprung. Zu beachten ist hier, dass $\mathbf{T}(\mathbf{q})$ nichtlinear ist in θ.
Äquivalent ist die Darstellung mittels homogener Transformation.

$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \cos\theta & -\sin\theta & r_x \\ \sin\theta & \cos\theta & r_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ 1 \end{bmatrix}.$$

Der Vorteil dieser Darstellung ist, dass sich eine Folge von Transformationen durch eine einzige homogene Transformationsmatrix ausdrücken lässt, welche durch einfache Matrixmultiplikationen berechnet werden kann.

Affine Parametrisierung Die vier reellen Parameter r_x, r_y, r_1 und r_2 (r_1 und r_2 nicht beide 0) beschreiben eine Transformation $\mathbf{T}$ wie folgt:

$$\mathbf{T}(\mathbf{q}) \equiv \begin{bmatrix} r_x \\ r_y \end{bmatrix} + \begin{bmatrix} r_1 & -r_2 \\ r_2 & r_1 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \end{bmatrix}.$$

Setzt man $r_1 = s\cos\theta$ und $r_2 = s\sin\theta$ so sieht man, dass die Matrix eine Skalierung und eine Verdrehung bewirkt. Der Vorteil dieser Darstellung

ist, dass die Transformation linear ist in allen vier Parametern.
Oftmals ist es sinnvoll, Translation und Rotation getrennt zu betrachten. Wir führen dazu die folgenden Bezeichnungen ein:

$$\mathbf{r}_{xy} \equiv \begin{bmatrix} r_x \\ r_y \end{bmatrix} ; \mathbf{r}_{12} \equiv \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

Damit kann die affine Transformation $\mathbf{T}$ in der $\mathbf{R^2}$-Ebene einfach dargestellt werden:

$$\mathbf{T}(\mathbf{q}) = \mathbf{r}_{xy} + \mathbf{r}_{12}\mathbf{q}$$

Wenn r_1 und r_2 beide nicht gleich 0 sind, so hat die Abbildung $\mathbf{T}$ eine Inverse $\mathbf{T}^{-1}$, so dass $\mathbf{T}^{-1}(\mathbf{T}(\mathbf{q})) = \mathbf{q}$ für jedes $\mathbf{q}$.

$$\mathbf{T}^{-1}(\mathbf{q}) = (\mathbf{q} - \mathbf{r}_{xy})\mathbf{r}_{12}^{-1} \tag{4.3}$$

4.3.2 Berechnung der Transformation T

Jetzt geht es darum, diejenige Transformation zu bestimmen, welche das Modell auf die Referenz abbildet. Sind je ein Punktepaar $(\mathbf{q}_1, \mathbf{q}_2)$ und $(\mathbf{p}_1, \mathbf{p}_2)$ gegeben, so ist die Transformation $\mathbf{T}$, welche $\mathbf{q}_1$ auf $\mathbf{p}_1$ und $\mathbf{q}_2$ auf $\mathbf{p}_2$ abbildet, eindeutig bestimmt. Ebenfalls eindeutig bestimmt ist $\mathbf{T}$, wenn je ein Punkt und eine Gerade gegeben sind, da dieser Fall auf den oberen Fall zurückgeführt werden kann. Je zwei Geraden genügen nicht, um $\mathbf{T}$ eindeutig zu definieren. Ist jedoch s genügend genau bekannt, was in technischen Anwendungen meistens der Fall ist, dann kann $\mathbf{T}$ auch aus zwei Geradenpaaren bestimmt werden.
Die vier Parameter von $\mathbf{T}$ können nacheinander berechnet werden:

1. Skalierung berechnen:

$$s = \frac{\|\mathbf{p}_2 - \mathbf{p}_1\|}{\|\mathbf{q}_2 - \mathbf{q}_1\|}$$

2. Drehung θ bestimmen:

$$\theta = \angle(\mathbf{p}_2 - \mathbf{p}_1) - \angle(\mathbf{q}_2 - \mathbf{q}_1)$$

3. Modell umskalieren (mit s) und um den Winkel θ drehen (mit der Rotationsmatrix $\mathbf{T}_\theta$):

$$\mathbf{q'}_{1r} = \mathbf{T}_\theta * s * \mathbf{q}_1$$

4. Verschiebung $\mathbf{r}$ berechnen

$$\mathbf{r} = \mathbf{p}_1 - \mathbf{q}'_{1r}$$

Eine direktere Formel ist in [Baird 84] angegeben:

$$\mathbf{T} = \begin{pmatrix} \overline{\mathbf{p}} - \Delta\mathbf{p}(\Delta\mathbf{q})^{-1}\overline{\mathbf{q}} \\ \Delta\mathbf{p}(\Delta\mathbf{q})^{-1} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_{xy} \\ \mathbf{r}_{12} \end{pmatrix}$$

mit $\overline{\mathbf{p}} \equiv 0.5 * (\mathbf{p}_1 + \mathbf{p}_2)$, $\Delta\mathbf{p} \equiv (\mathbf{p}_2 - \mathbf{p}_1)$, $\overline{\mathbf{q}} \equiv 0.5 * (\mathbf{q}_1 + \mathbf{q}_2)$ und $\Delta\mathbf{q} \equiv (\mathbf{q}_2 - \mathbf{q}_1)$.

4.3.3 Behandlung von Unsicherheiten

Soll eine Modellpunktmenge auf eine Referenzpunktmenge abgebildet werden, so ist zu berücksichtigen, dass Sensordaten immer verrauscht sind und somit eine Abbildung nur unter Zulassung gewisser Toleranzen möglich ist. Für die Angabe dieser Toleranzen gibt es verschiedene Möglichkeiten:

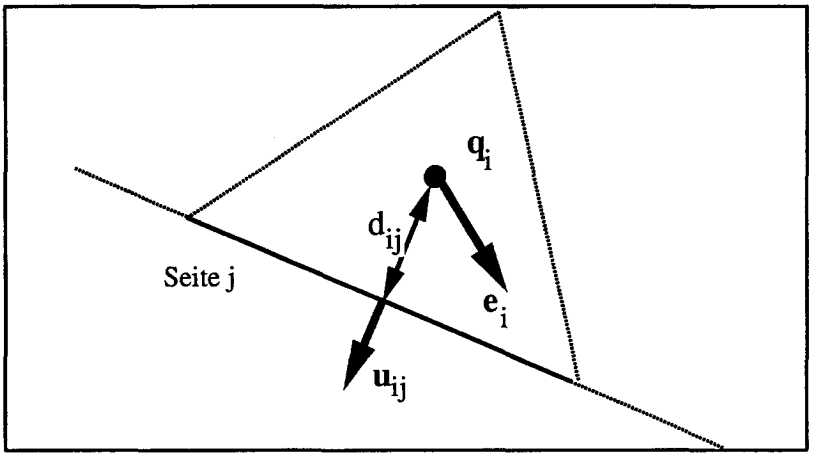
Fehlerpolygone im $\mathbf{R}^2$-Raum Nimmt man eine Transformation $\mathbf{T}$ als gegeben an, so ergibt sich für einen zum Sensorpunkt $\mathbf{q}_i$ gehörenden Modellpunkt $\mathbf{p}_{m_i}$ folgender Fehlervektor:

$$\mathbf{e}_i \equiv \mathbf{T}^{-1}(\mathbf{p}_{m_i}) - \mathbf{q}_i \tag{4.4}$$

Mit dieser Darstellung wird der Messfehler automatisch im Sensormodell skaliert. Für jeden Sensorpunkt wird der maximale Messfehler als konvexes Polygon um diesen Punkt beschrieben, d.h. der tatsächliche Wert muss innerhalb dieses Polygons liegen. Beschreibt man das Polygon als Schnittmenge von Halbebenen, so stellt jede Seite des Polygons eine lineare Ungleichung dar. Diese Ungleichungen können als Bedingungen interpretiert werden.

$$\mathbf{u}'_{ij}\mathbf{e}_i \leq d_{ij} \tag{4.5}$$

$\mathbf{u}_{ij}$ ist ein Einheitsvektor rechtwinklig zur Polygonseite, d_{ij} ist der Abstand von $\mathbf{q}_i$ zu dieser Seite und $'$ bedeutet das Skalarprodukt. Formell wird 4.5 dargestellt als $< \mathbf{u}_{ij}, d_{ij} >$. Die ganze Menge der Fehlergrenzen



Figur 4.9: *Formales Modell der Fehlergrenzen*

von $\mathbf{q}_i$ kann geschrieben werden als $< \mathbf{U}_{ij}, \mathbf{d}_i >$ mit $\mathbf{U}_{ij}$ als l_ix2-Matrix und $\mathbf{d}_i$ als Vektor der Länge l_i. Um ein Polygon einzuschliessen, muss $l_i \geq 3$ sein. Es ist sinnvoll, l_i nicht wesentlich grösser als 3 zu wählen. Eine Abbildung von k Punkten des Modelles auf k Punkte der Referenz ist möglich, wenn eine Abbildung $\mathbf{T}$ existiert, so dass alle punkteweisen Positionsfehler folgende Beschränkungen erfüllen:

$$\mathbf{U}_{ij}\mathbf{e}_i \leq \mathbf{d}_i \text{ für alle i} = 1,...,k$$

Abweichung im $\mathbf{R}^4$-Raum Ob eine Abbildung von k Punkten des Modelles auf k Punkte der Referenz möglich ist, wird dadurch entschieden, ob die Abbildungen im R^4-Raum innerhalb einer Hyperkugel mit definiertem Radius liegen oder nicht. Die Grösse dieses Radius' ist davon abhängig, wie genau die Positionen der Modellpunkte bestimmt werden können.
Mit der Angabe der Fehlerpolygone ist es relativ einfach, die Charakteristik von verschiedenen Sensorsystemen zu berücksichtigen. Dafür ist der Test, ob eine Abbildung möglich ist oder nicht, aufwendiger.

4.3.4 Matchingverfahren

Ein *Matching M der Grösse k* ist eine direkte Abbildung einer Untermenge von P (Referenzmenge) der Grösse k auf Q (Modellmenge). Das gleiche gilt für eine Abbildung von Q auf P. Ist ein Matching bekannt, so können Punkte in der Modellmenge Q sofort bekannten Objekten in P zugeordnet werden. Ein Matching der Grösse k wird geschrieben

$$m_1\ m_2\ \ldots\ m_k$$

und gelesen: $\mathbf{pm_1}$ passt zu $\mathbf{q_1}$, $\mathbf{pm_2}$ passt zu $\mathbf{q_2}$, usw.

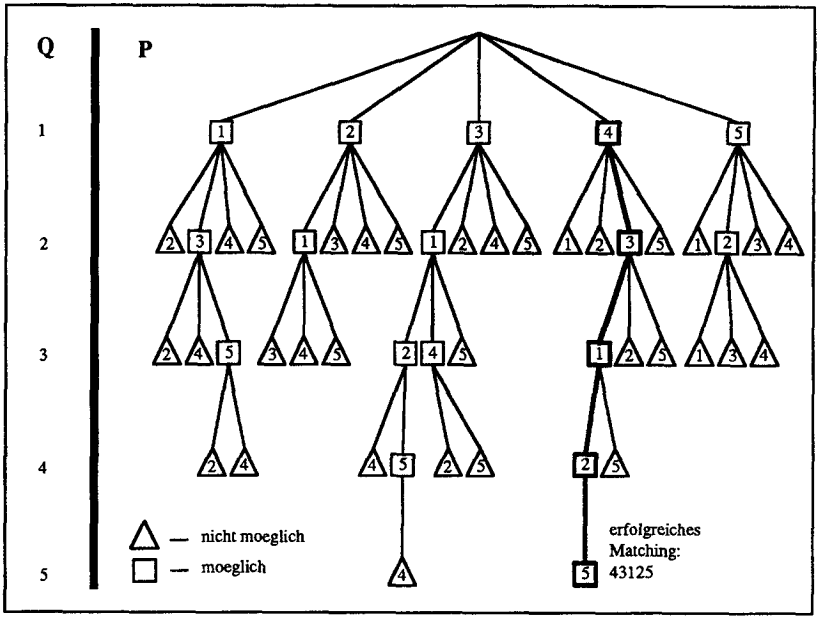
Nach [Baird 84] ist es nicht möglich, das oben formulierte Matchingproblem mittels Methoden zu lösen, bei denen die Positionsinformation nicht beibehalten wird, wie etwa bei der Verwendung von Fourierkoeffizienten oder bei der Verwendung von Flächenmomenten. Vielmehr muss theoretisch jede Kombination von Modell- und Referenzpunkten auf ihre Zulässigkeit untersucht werden.

Für die systematische Suche werden die Sensorobjekte $\mathbf{q}_i$ zufällig numeriert ($\mathbf{q}_1\ \mathbf{q}_2\ \ldots\ \mathbf{q}_m$). Die Numerierung hat keinen Einfluss auf die gefundene Lösung, aber die Suchzeit kann dadurch beeinflusst werden. Die Reihenfolge der Modellpunkte $\mathbf{p}_i$ ist von Anfang an gegeben: $\mathbf{p}_1\ \mathbf{p}_2\ \ldots\ \mathbf{p}_n$.

Für das Finden eines totalen Matchings wird ein Baum von Teilmatchings in die Tiefe abgesucht, wobei für jedes Matching M alle kürzeren Vorgänger von M vorher geprüft worden sind (siehe Figur 4.10). Eine mögliche Suchordnung ist die lexikographische Suche ([Reingold et. al 77]). Für n=3 sähe die Reihenfolge so aus:

1 12 123 13 132 2 21 213 231 3 31 312 32 321

Jedes Teilmatching wird auf seine Zulässigkeit überprüft. Ist ein Teilmatching nicht möglich, so muss auf diesem Pfad nicht mehr tiefer gesucht werden, d.h. dieser Ast kann abgeschnitten werden. Da alle totalen Matchings gefunden werden sollen, wird die Suche weitergeführt, auch wenn bereits ein Pfad bis zum Ende durchlaufen worden ist.



Figur 4.10: *Tiefensuche in einem Baum von teilweisen Matchings.*

Die Grösse des Suchbaumes ist gleich der Anzahl getesteter Teilmatchings und wird bestimmt durch die Geometrie der Referenz, durch die Geometrie des Modelles, durch die Art des Tests und durch die Reihenfolge der Sensorpunkte. Eine wichtige Rolle spielen aber auch die Grössen m (Anzahl Modellobjekte), m_H (Anzahl Hindernisse, d.h. Objekte, die in der Referenz nicht modelliert sind) und n (Anzahl Referenzobjekte).

- $(m = n) \wedge (m_H = 0)$:
  Ist $T(n)$ die totale Anzahl der getesteten Matchings und wird angenommen, dass mindestens ein totales Matching existiert, so hat der kleinstmögliche Suchbaum die Grösse:

$$T(n) \geq \sum_{k=1}^{n}(n-k+1) = \Omega(n^2). \tag{4.6}$$

  Eine obere Grenze für die Grösse des Suchbaumes anzugeben, ist wesentlich schwieriger. Falls das Sensorrauschen sehr gross ist,

so dass jedes Teilmatching möglich ist, so hat der Suchbaum eine Grösse von $T(n) = O(n^n)$. Sind die Fehler kleiner, so reduziert sich der Suchbaum entsprechend.

- $(m < n) \wedge (m_H = 0)$:
  Die Tiefe des Suchbaumes ist m. Der kleinstmögliche Suchbaum hat immer noch die Grösse $T(n) = \Omega(n^2)$, während sich der grösstmögliche Suchbaum auf $T(m,n) = O(n^m)$ reduziert.
- $(m > n) \wedge (m_H = 0)$:
  Dieser Fall ist kann auf den Fall oben zurückgeführt werden.
- $(m < n) \wedge (m_H \neq 0)$:
  Gibt es Objekte im Modell, zu dem kein passendes Objekt in der Referenz vorhanden ist, so existiert kein zulässiges Teilmatching und die Suche bricht in jedem Fall ab. Um dieses Problem zu umgehen, müssen in der Referenz sogenannte *Attrappenobjekte* eingeführt werden, die zu jedem Objekt im Modell passen. Damit wird verhindert, dass die Tiefensuche frühzeitig abbricht. Allerdings vergrössert sich dadurch der kleinstmögliche Suchbaum beträchtlich: $T(n, m_H) = \Omega(n^{2+m_H})$, während der grösstmögliche Suchbaum sich nur wenig ändert: $T(m, n, m_H) = O(n^{m+m_H})$.

Die in [Baird 84], [Miller 84] und [Drumheller 87] vorgeschlagenen Matchingmethoden unterscheiden sich vor allem durch die Art der gewählten Objekte und der Testmethode für die Teilmatchings.
Baird nimmt als Objekte Punkte an, die sich allein durch ihre Position unterscheiden. Ihren maximalen Messfehler grenzt er durch die oben beschriebenen Fehlerpolygone ein. Von jeder Polygonseite kann eine Bedingung für eine mögliche Transformation $\mathbf{T}$ abgeleitet werden. Angenommen $< \mathbf{u}, d >$ sei eine Fehlerbegrenzung für $\mathbf{q}$, so ist M möglich, falls eine Abbildung $\mathbf{T}$ existiert, so dass gilt:

$$\mathbf{u}'(\mathbf{T}^{-1}_{(\mathbf{p})} - \mathbf{q}) \leq d \tag{4.7}$$

Diese Ungleichung kann zu folgender Bedingung umgeformt werden:

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{up}^* \end{bmatrix}' \mathbf{T}^{-1}_{(\mathbf{p})} \leq d + \mathbf{u}'\mathbf{q} \tag{4.8}$$

Damit kann das Testen eines Teilmatchings auf die Frage zurückgeführt werden, ob das System der aus allen beteiligten Polygonseiten abgeleiteten Ungleichungen eine Lösung hat oder nicht. Dafür stehen leistungsfähige Verfahren wie z.B. der Simplexalgorithmus zur Verfügung.

Während in Baird das Matchingproblem möglichst allgemein angegangen wird, wollen Miller und Drumheller die Position eines mobilen Roboters an Hand von Ultraschallmessungen bestimmen. Sie nehmen an, dass das mit Ultraschall am besten zu erkennende Objekt die Wand ist. Dadurch ändert sich vor allem die Methode für das Testen eines Teilmatchings. Dies geschieht durch Testen der Richtungen, der Abstände und der Reihenfolge der detektierten Wände.

Der Zeitbedarf von Matchingverfahren wird durch zwei Faktoren bestimmt: der Grösse des Suchbaumes und dem Aufwand für das Testen eines Teilmatchings. Diese können separat betrachtet werden, da die Grösse des Suchbaumes unabhängig ist von der Methode, wie ein Teilmatching getestet wird.

4.3.5 Clusteringverfahren

Aus allen möglichen Kombinationen von Modell- und Referenzobjekten werden die Transformationen T berechnet und in den R^4-Raum übertragen. Da richtige Paarungen immer die gleiche Transformation T ergeben, ist zu erwarten, dass um die richtige Position eine Anhäufung von Punkten zu finden ist. Diese Anhäufung soll mit einem Clusteringverfahren gefunden werden.
Der Aufwand des Verfahrens setzt sich zusammen aus der Berechnung der Transformationen und aus dem Suchen der grössten Punktanhäufung im R^4-Raum.

Berechnung der Transformationen Ist m die Anzahl Modellobjekte und n die Anzahl Referenzobjekte, so können daraus

$$T(m,n) = \frac{1}{2}m(m-1)n(n-1) \tag{4.9}$$

mögliche Transformationen berechnet werden. Längst nicht alle Transformationen sind auch geometrisch sinnvoll. Wird der Skalierungsfaktor eingeschränkt, so müssen sehr viele Transformationen gar nicht berechnet werden. Von den berechneten Transformationen sind zudem nur jene sinnvoll, die auf Roboterpositionen führen, die geometrisch überhaupt möglich sind.

Suchen der grössten Punktanhäufung Ein Cluster ist eine Gruppe zusammengehöriger Elemente einer statistischen Menge. Betrachten wir die Objekte in einem Raum als zufällig verteilt, so sollten die daraus berechneten Transformationen T ebenfalls zufällig verteilt sein. Es ist also naheliegend, für die Suche der grössten Punktanhäufung eines der bekannten Clusteranalyseverfahren anzuwenden.

Die meisten Verfahren können entweder den *Hierarchischen*, den *Optimierungs-* oder den *Dichtesuchmethoden* zugeordnet werden. Alle diese Methoden gehen davon aus, dass eine Menge von Elementen gegeben ist, welche in eine vordefinierte Anzahl von Gruppen unterteilt werden soll. Ein erstes Hauptproblem stellt denn auch die Definition der Anzahl der Gruppen dar.

Bei den *Hierarchischen Verfahren* bildet am Anfang jedes Element ein Cluster. Sukzessive werden immer grössere Clusters gebildet, indem alle Distanzen zwischen den einzelnen Clusters getestet werden und die beiden Clusters mit der kleinsten Distanz zu einem einzigen Cluster verbunden werden. Diese Verfahren sind vergleichsweise schnell. Nachteilig ist aber, dass eine Einteilung, die sich später als schlecht erweist, nicht mehr korrigiert werden kann.

Wesentlich aufwendiger sind die *Optimierungstechniken.* Mit einem einfachen aber nicht optimalen Clusterverfahren wird eine Starteinteilung gesucht. Diese Einteilung versucht man zu verbessern, indem systematisch Elemente zwischen bereits definierten Clusters ausgetauscht werden. Die Qualität einer Einteilung wird mittels eines Gütekriteriums angegeben.

Ähnlich aufwendig sind die *Dichtesuchverfahren.* Dabei werden Gebiete möglichst grosser Dichte gesucht. Allerdings ist es praktisch unmöglich zu entscheiden, ob man ein lokales oder ein globales Maximum gefunden hat.

Bei der Wahl des richtigen Clusterverfahrens sind die Art der Eingangsdaten, die Art der gewünschten Ausgangsdaten, der Rechenaufwand und die Korreliertheit der Daten zu berücksichtigen. Für unsere Anwendung ist vor allem die Minimierung des Rechenaufwandes von zentraler Bedeutung. Eine schnelle empirische Methode ist deshalb den oben vorgestellten Methoden vorzuziehen.

4.3.6 Vergleich des Matching- und des Clustering-Verfahrens für die Bestimmung von T

Dies sind die Vorteile (**V**) und Nachteile (**N**) der beiden Verfahren:

Matchingverfahren

V Liefert direkt eine Liste von aufeinander abbildbaren Objekten.

V Die Messgenauigkeit kann durch das Fehlerpolygon exakt charakterisiert werden.

N Grosser Rechenaufwand, der mit der Anzahl der Hindernisse überproportional ansteigt. (Vergleiche den Aufwand für den kleinstmöglichen Suchbaum: $T(n, m_H) = \Omega(n^{2+m_H})$).

N Die Anzahl Hindernisse muss für die Berechnung im voraus angenommen werden.

Clusteringverfahren

V Anzahl der nicht modellierten Hindernisse hat keinen Einfluss auf den Rechenaufwand.

V Nicht eindeutig erkannte Umgebungsobjekte können verwendet werden.

N Die Genauigkeit der Messungen kann nicht einzeln berücksichtigt werden.

Für die Aufgabe der Positionsbestimmung ist das Clusteringverfahren vorzuziehen, weil es die Anforderungen bezüglich voraussagbarer Rechenleistung und vor allem bezüglich der Toleranz gegenüber Hindernissen viel besser erfüllt.

4.4 Clusteringverfahren für die Grobbestimmung der Roboterposition

Mit dem Clusterverfahren soll die ungefähre Position des Roboters bestimmt werden. Dabei werden wir uns an das bei der Clusteranalyse

	1	2	3	...	n
r_x	0.2	2.1	-3.3	...	7.6
r_y	8.3	3.6	3.0	...	-2.2
r_1	1.2	5.1	0.3	...	5.5
r_2	0.3	0.1	1.1	...	3.6

Figur 4.11: *Datenmatrix X*

übliche Vorgehen halten:

1. Datenmatrix aufstellen
2. Datenmatrix standardisieren
3. Ähnlichkeitsmass festlegen
4. Clusters bilden
5. Clusters analysieren

4.4.1 Standardisierte Datenmatrix

In der Datenmatrix **X** werden die Objekte zusammengefasst, die der Clusterbildung unterzogen werden sollen. Dabei bedeutet X_{ij} das i-te Attribut des j-ten Objektes. Unsere Objekte sind die Punkte im R^4-Raum mit den Attributen r_x, r_y, r_1 und r_2 (Fig. 4.11). Bei der Bestimmung der Ähnlichkeit der einzelnen Objekte sollen alle Attribute im gleichen Masse berücksichtigt werden. Haben die Attribute nicht alle die gleiche Dimension, so muss die Datenmatrix standardisiert werden. Die standardisierte Datenmatrix bezeichnen wir mit **Z**. Fehlen Angaben über die Grösse und die Verteilung der Attribute, so wird diese meistens nach folgender Vorschrift berechnet:

$$Z_{ij} = \frac{X_{ij} - \overline{X}_i}{S_i} \tag{4.10}$$

mit

$$\overline{X}_i = \frac{\sum_{j=1}^{t} X_{ij}}{t} \text{ und } S_i = \sqrt{\frac{\sum_{j=1}^{t}(X_{ij} - \overline{X}_i)^2}{t-1}}$$

Dabei beizeichnet t die Anzahl der Objekte. Mit der Standardisierung werden die Attribute dimensionslos und verteilen sich in einem ungefähr gleich grossen Intervall.
Um Rechenzeit zu sparen und damit alle Attribute im Intervall [0..1] liegen, werden wir die Datenmatrix nach einer einfacher zu berechnenden Formel standardisieren:

$$Z_{ij} = \frac{X_{ij} - RMIN_i}{RMAX_i - RMIN_i} \tag{4.11}$$

In unserem Fall ist diese Standardisierung sinnvoll, da wir annehmen können, dass die Punkte im R^4-Raum gleichmässig verteilt sind und weil die Abschätzung von $RMIN_i$ und $RMAX_i$ einfach möglich ist: $RMIN_{rx}$, $RMAX_{rx}$, $RMIN_{ry}$ und $RMAX_{ry}$ entsprechen den Ausdehnungen der strukturierten Umgebung. $RMIN_{r1}$ und $RMIN_{r2}$ betragen -1, $RMAX_{r1}$ und $RMAX_{r2}$ sind gleich +1.
Hier zeigt sich auch der Vorteil der affinen Parametrisierung von T. Bei der trigonometrischen Parametrisierung lägen z.B. die Winkel 1° und 359° im R^4-Raum weit auseinander, obwohl sie in Wirklichkeit sehr ähnlich sind. Bei der affinen Parametrisierung tritt dieses Problem gar nicht auf, weil r_1 und r_2 über den ganzen Winkelbereich stetig sind.

Als Ähnlichkeitsmass verwenden wir die euklidische Distanz zwischen den Objekten der standardisierten Datenmatrix.

$$e_{jk} = \sqrt{\sum_{i=1}^{m}(Z_{ij} - Z_{ik})^2}$$

e_{jk} bezeichnet die Distanz zwischen den Objekten j und k. m ist die Anzahl Attribute und beträgt in unserem Fall vier.

4.4.2 Clusterbildung

Für die Clusterbildung benutzen wir nicht einen der früher vorgestellten Algorithmen, sondern ein heuristisches Verfahren. Dieses Verfahren ist bedeutend schneller und liefert bei zufälliger Verteilung der Objekte

in den meisten Fällen sehr gute Resultate, besonders wenn wie hier kugelförmige Clusters gefunden werden müssen.

Algorithmus Die Verteilung und die Numerierung der zu analysierenden Punktmenge sei zufällig. Wir benutzen die Mengen

$PTS = \{pts_i\}$ Menge der noch zu analysierenden Punkte
$REM = \{rem_i\}$ Menge der noch nicht einteilbaren Punkte
$CEN = \{cen_i\}$ Menge der bereits gefundenen Clusters
$LON = \{lon_i\}$ Menge der Clusters mit nur einem Punkt

mit den Einzelelementen

$pts = [rx, ry, r1, r2]^T$,
$rem = [rx, ry, r1, r2]^T$,
$cen = [rx, ry, r1, r2, n]^T$ und
$lon = [rx, ry, r1, r2]^T$.

Daneben werden die Grössen n, npts, matched, *newcen*, *curpt* und *rCluster*, sowie die Funktion $d(x, y)$ verwendet, welche die euklidische Distanz zwischen zwei Punkten x und y berechnet. Der Algorithmus lautet:

$LON = \emptyset$; $CEN = \emptyset$; npts = n; (Initialisierung)
while npts > 1,
    $REM = \emptyset$;
    $newcen = \{pts_1\}$;
    matched = 0;
    for j = 2 .. npts,
        $curpt = \{pts_j\}$;
        if $d(newcen, curpt) < rCluster$,
            matched = matched + 1;
            $newcen = \frac{\text{matched}*newcen + curpt}{\text{matched} + 1}$;
        else
            $REM = REM + \{curpt\}$;

```
        end;
    end;
    if matched > 0,
        newcen.n = matched;
        CEN = CEN + {newcen};
    else
        LON = LON + {newcen};
    end;
    PTS = REM;
end;
LON = LON + PTS;
```

Das Verfahren ist deshalb relativ schnell, weil jeder Punkt nur einmal einem Cluster zugeteilt wird, womit aufwendige Austauschschritte entfallen. Dafür kann nicht garantiert werden, dass die gefundene Einteilung irgend ein Optimalitätskriterium erfüllt.
Der Aufwand für die Clusterbildung hängt von der Anzahl Punkte im R^n-Raum ab. Indirekt ist der Aufwand auch von *rCluster* abhängig, denn der Algorithmus läuft umso schneller, je weniger Clusters gebildet werden müssen. Im besten Fall, wenn alle Punkte zum selben Cluster gehören, beträgt der Aufwand $T(n) = \Omega(n)$. Im schlechtesten Fall, wenn nur alleinstehende Punkte gefunden werden, was aber je nach Grösse des Raumes und von *rCluster* gar nicht möglich ist, wächst der Aufwand auf $T(n) = O(n^2)$.

Der einzige wählbare Parameter ist *rCluster*. Damit wird die Grösse der Hyperkugel definiert, innerhalb welcher die Punkte liegen müssen, die zum selben Cluster gehören. Weil *rCluster* sowohl die Geschwindigkeit als auch das Ergebnis der Clusteranalyse entscheidend beeinflusst, muss dieser Parameter sorgfältig abgeschätzt werden. Man kann für diese Aufgabe von zwei Betrachtungsweisen ausgehen. Entweder macht man *rCluster* von der Sensorgenauigkeit abhängig oder man benutzt das Verhältnis zwischen der Anzahl einzuteilender Punkte und der Grösse des R^n-Raumes.
Im ersten Fall geht man davon aus, dass sich die Unsicherheitsgebiete um die Objekte im Modell direkt in ein Unsicherheitsgebiet im R^4-Raum übertragen. Wie dieses Unsicherheitsgebiet berechnet werden kann, ist in [DurrantWhyte 88] angegeben. Die geometrischen Objekte und die Transformationen werden dabei als Zufallsgrössen betrachtet, deren Unsicherheiten durch Varianzmatrizen charakterisiert werden. Da

Unsicherheitsgebiete und Varianzmatrizen einfach ineinander überführt werden können, kann aus der Varianzmatrix von T direkt auf *rCluster* geschlossen werden.
Im zweiten Fall nimmt man an, dass die Punkte gleichmässig im R^4-Raum verteilt seien. Dann wird der Raum in so viele Einheiten unterteilt, dass pro Einheit deutlich weniger Punkte vorhanden sind, als im gesuchten Cluster zu erwarten sind. So wird die Wahrscheinlichkeit klein gemacht, dass ein zufälliger Cluster mehr Punkte enthält als der gesuchte.
Aus zweierlei Gründen ziehen wir eine Abschätzung nach der zweiten Methode vor. Die Berechnung ist einfacher und es sind nicht so viele schlecht abschätzbare Annahmen zu machen.

$$rCluster = \frac{1}{2}\sqrt[n]{\frac{V_{Raum} * n_0}{n_{Punkte}}} \tag{4.12}$$

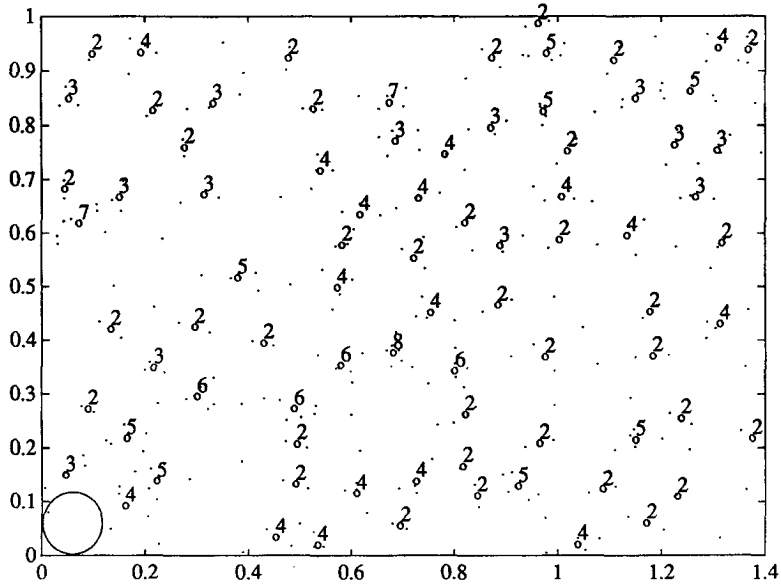
(n: Dimension des Raumes, V_{Raum}: Volumen des Raumes, n_0: mittlere Anzahl Punkte pro Cluster, n_{Punkte}: Anzahl der einzuteilenden Punkte).
Diese Formel berechnet die halbe Kantenlänge jenes Würfels, in welchem bei einer Gleichverteilung der Punkte genau n_0 Punkte liegen würden. Diese entspricht dem Radius der Kugel, welche in diesem Würfel Platz hätte. Wenn alle Attribute auf das Intervall [0..1] standardisiert werden, so ist V_{Raum} immer gleich 1.0.

Das Verhalten des Algorithmus' soll nun am Beispiel von 300 zufällig in der Ebene verteilten Punkten gezeigt werden (Fig. 4.12). Die Abmessungen der Ebene betragen $RMIN_x = 0.0$, $RMAX_x = 1.4$, $RMIN_y = 0.0$ und $RMAX_y = 1.0$. Weiter wählen wir n_0 gleich 4. Nach der Standardisierung erhalten wir für *rCluster*:

$$rCluster = \frac{1}{2}\sqrt[2]{\frac{(1.0 * 1.0) * 4}{300}} = 0.0577$$

In Fig 4.12 ist das Resultat dargestellt. Es zeigt sich, dass die Punkte sicher und zuverlässig in Gruppen eingeteilt werden. Die Grösse der einzelnen Cluster schwankt zwischen 1 und 8 und beträgt im Durchschnitt ziemlich genau 4, was der Grösse n_0 entspricht.

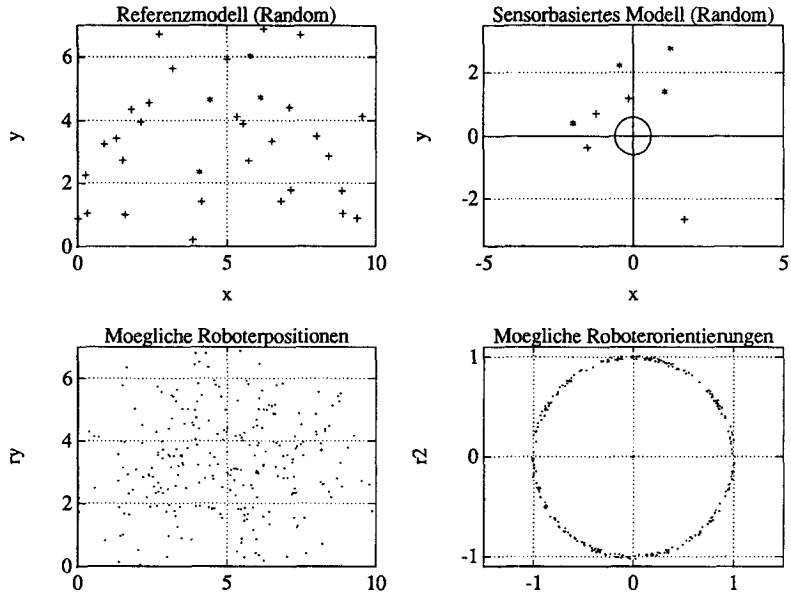
Das obige Beispiel zeigte das Verhalten des Algorithmus', sowie den Zusammenhang zwischen *rCluster* und der durchschnittlichen Grösse



Figur 4.12: *Einteilung von 300 zufällig in der Ebene verteilten Punkten in Clusters mit rCluster = 0.0577 (siehe Kreis unten links). Die Clusters sind durch ihren Mittelpunkt und die Anzahl zugehöriger Punkte dargestellt.*

der gefundenen Clusters. Die Brauchbarkeit des Algorithmus' für die Positionsbestimmung soll an Hand eines anderen Beispieles demonstriert werden (Fig 4.13). Die Aufgabe besteht dabei darin, ein *Sensorbasiertes Modell* so zu transformieren, dass es sich möglichst gut mit einem *Referenzmodell* deckt, welches aus 34 zufällig in der Ebene verteilten Punkten besteht. Unter der Annahme, dass in der Realität etwa die Hälfte der erkannten Objekte Hindernisse sind, wird das *Sensorbasierte Modell* aus 4 zufälligen Punkten und aus 4 vom Referenzmodell abgeleiteten Punkten zusammengesetzt. Die 4 abgeleiteten Punkte werden durch die Transformation $T = [6.0, 3.0, 0.86, 0.5]^T$ vom *Referenzmodell* ins *Sensorbasierte Modell* übertragen. Um das Beispiel möglichst realistisch zu machen, werden diese Punkte in r- und in ϕ-Richtung gestört ($\sigma_r = 0.03$m, $\sigma_\phi = 5°$).

Fig 4.13 zeigt das *Referenz-* und das *Sensorbasierte Modell*, sowie die daraus berechneten Punkte im R^4-Raum. Aus darstellungstechnischen Gründen ist dieser in zwei Teile geteilt. In der $r_x - r_y$-Ebene sieht

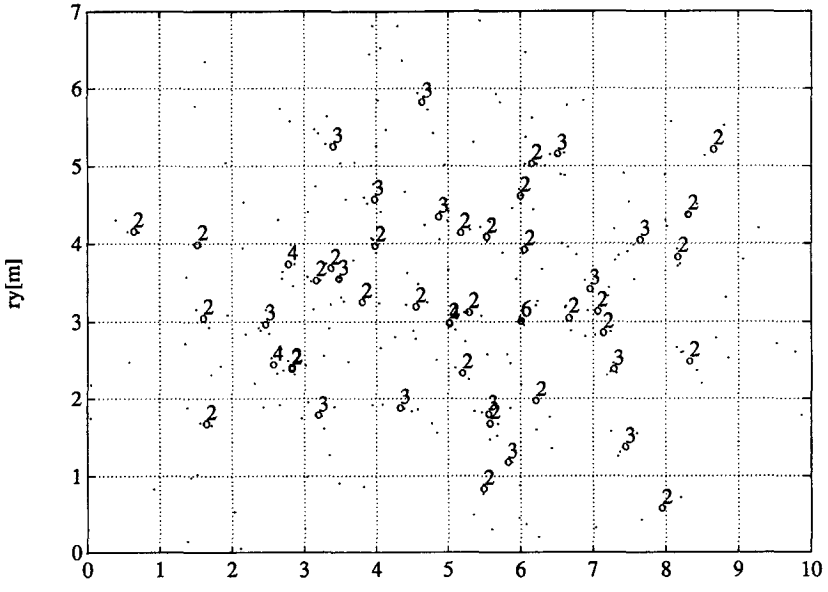


Figur 4.13: *Bestimmung der Punkte im R^4-Raum. Die Modelle bestehen aus zufälligen (+) und aus ineinander überführbaren Punkten (*).*

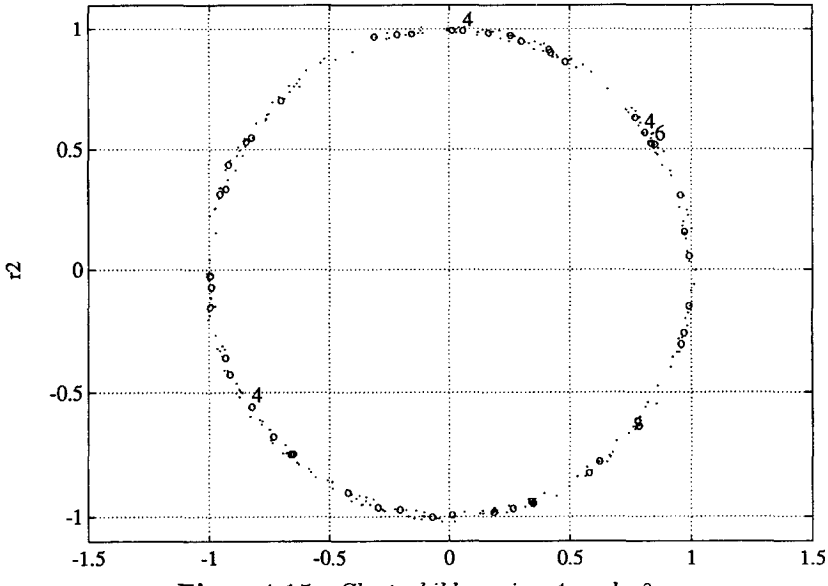
man bereits relativ deutlich die dichte Punkteansammlung bei Position [6.0,3.0]. Ob diese Punkte in der $r_1 - r_2$-Ebene auch zusammenliegen, kann von Auge nicht beurteilt werden. Das Resultat der Clusterbildung ist aus den beiden Figuren 4.14 und 4.15 ersichtlich. Der grösste Cluster enthält 6 Punkte und befindet sich ziemlich genau an der gesuchten Position [6.0, 3.0, 0.86, 0.5].

4.4.3 Clusteranalyse

Die Clusterbildung allein stellt erst einen Zwischenschritt zur Lösung eines Problems dar. Die Lösung selber kann erst durch die Analyse der dabei gemachten Gruppeneinteilung gefunden werden. Für uns heisst das, dass aus allen Clustern cen_i derjenige gefunden werden muss, dessen Mittelpunkt am nächsten bei der gesuchten Roboterposition liegt. Im allgemeinen sollte der gesuchte Cluster derjenige mit den meisten Punkten sein. Diese Annahme ist aber zu unsicher und muss durch zusätzliche Tests untermauert werden.



Figur 4.14: *Clusterbildung in rx und ry*



Figur 4.15: *Clusterbildung in r1 und r2*

posFound = FALSE;
sort CEN, s.t. $cen_i.n > cen_{i+1}.n$
if $cen_1.n \geq n_0 + 6$,
    $r_{Robot} = cen_1$;
    posFound = TRUE;
else if $(cen_1.n - cen_2.n) > 2n_0$,
    $r_{Robot} = cen_1$;
    posFound = TRUE;
else
    posFound = FALSE;

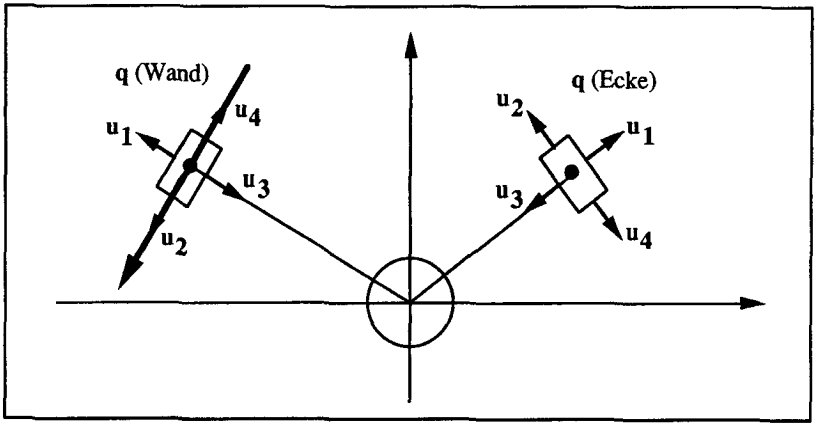
Kann die Position des Roboters so nicht gefunden werden, bleiben verschiedene Möglichkeiten, wie weiter verfahren werden soll.

- Die Clusterbildung und -analyse wird mit halb so grossem n_0 wiederholt. Dadurch kann die Wahrscheinlichkeit verkleinert werden, dass der grösste Cluster zufällig entstanden ist.
- Man betrachtet cen_1 als mögliche Roboterposition und führt damit den im nächsten Unterkapitel beschriebenen Schritt durch. Ist die Zahl der gefundenen Objekte genügend gross, so ist cen_1 die gesuchte Position.
- Der Roboter wechselt seinen Standort und versucht von dort aus seine Position zu bestimmen.

4.4.4 Zuordnung der gefundenen Objekte

Für die exakte Bestimmung der Position ist es unerlässlich zu wissen, welche Objekte im *Sensorbasierten Modell* welchen Objekten im *Referenzmodell* entsprechen. Man geht dabei folgendermassen vor.

Gegeben sei das *Sensorbasierte Modell* $Q = \{q_i\}$ und das *Referenzmodell* $P = \{p_i\}$, sowie eine Transformation $\mathbf{T}$, welche Q auf P abbildet. Gesucht ist eine Liste $m_1, m_2, ... m_n$, so dass das i-te Objekt von Q dem m_i-ten Objekt von P entspricht. Mit der Ungleichung 4.8 kann getestet werden, ob ein von P nach Q transformierter Punkt innerhalb des Fehlerpolygons um q_i liegt oder nicht. Für punktförmige



Figur 4.16: *Bestimmung des Fehlerpolygons.*

Objekte (Kante, Ecke) wird das Fehlerpolygon so definiert (Fig 4.16), dass dessen Ausdehnung proportional ist zu den Unsicherheiten in r- und ϕ-Richtung.

$$
\begin{aligned}
\mathbf{u}_{i1} &= \frac{\mathbf{q}_i}{\|\mathbf{q}_i\|}, & d_{i1} &= \sigma_{ir} \\
\mathbf{u}_{i2} &= \begin{bmatrix} 0 & -1 \\ +1 & 0 \end{bmatrix} \mathbf{u}_{i1}, & d_{i2} &= \sigma_{i\phi} \\
\mathbf{u}_{i3} &= \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \mathbf{u}_{i1}, & d_{i3} &= \sigma_{ir} \\
\mathbf{u}_{i4} &= \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \mathbf{u}_{i1}, & d_{i4} &= \sigma_{i\phi}
\end{aligned}
$$

Geradenförmige Objekte (Wände) können auf die gleiche Art getestet werden, wenn $\mathbf{q}_i = [r_i \cos\phi_i, r_i \sin\phi_i]^T$ und $\mathbf{p}_i$ gleich dem Fusspunkt des Lotes gesetzt wird, welches von der geschätzten Roboterposition auf die betrachtete Gerade gezeichnet wird.

Da nur Objekte gleicher Art aufeinander abgebildet werden können, wird jede Objektart einzeln getestet. Für die einzelnen Objekte geht man folgendermassen vor:

Ecken:

for i=1..m_E, (nehme gemessene Ecke)

compute $\mathbf{u}_{i1}..\mathbf{u}_{i4}, d_{i1}..d_{i4}$; (berechne Fehlerpolygon)
$m_{Ei} = 0$;
for j=1..n_E, (nehme Referenzecke)
if $\forall k \in \{1..4\}$: $\begin{bmatrix} \mathbf{u}_{ik} \\ \mathbf{u}_{ik}\mathbf{p^*}_j \end{bmatrix}' \mathbf{T}^{-1} \leq d_{ik} + \mathbf{u}'_{ik}\mathbf{q}_i$,
$m_{Ei} = j$;
end;
end;

Genau gleich sieht der Algorithmus für das Objekt *Kante* aus, während beim Objekt *Wand* als Ergänzung der Fusspunkt des Lotes der Geraden berechnet werden muss.

4.5 Feinbestimmung der Roboterposition mittels Ausgleichsrechnung

Sobald zu den Sensormessungen passende Punkte im *Referenzmodell* gefunden worden sind, kann die Position des Roboters berechnet werden (Figur 4.17). Wir schlagen eine Methode vor, die die Summe der Abstandsquadrate minimiert. Da der Abstand zwischen zwei Punkten anderst berechnet wird als der Abstand eines Punktes von einer Geraden, müssen wir zwei Fälle unterscheiden: Positionsberechnung aus Punkten und Positionsberechnung aus Geraden.

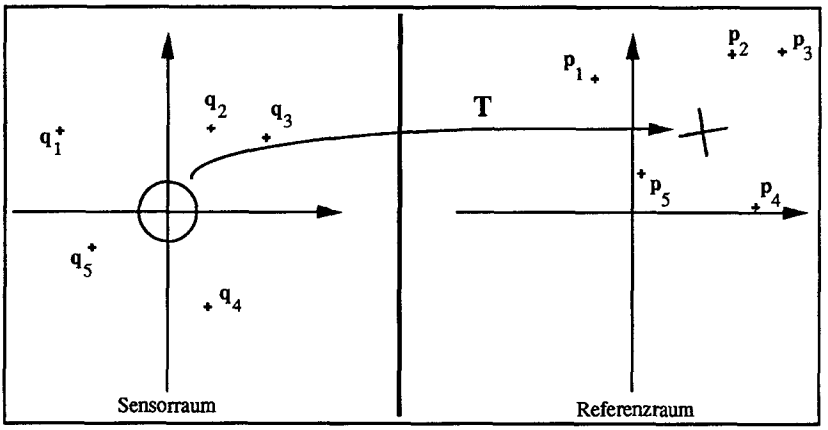
Punkte Soll die Roboterposition auf Grund von Messungen zu Punkten bestimmt werden, so haben wir folgendes Problem zu lösen:

Gegeben sind n Sensorpunkte $\mathbf{q}_i$ und n Referenzpunkte $\mathbf{p}_i$. Gesucht ist eine Abbildung $\mathbf{T}$, so dass folgende Bedingung erfüllt ist:

$$\sum_{i=1}^{n} \|\mathbf{p}_i - \mathbf{T}(\mathbf{q}_i)\|^2 = Min \tag{4.13}$$

wobei

$$\mathbf{T}(\mathbf{q}_i) \equiv \begin{bmatrix} r_x \\ r_y \end{bmatrix} + \begin{bmatrix} r_1 & \\ & r_2 \end{bmatrix} \begin{bmatrix} q_{xi} \\ q_{yi} \end{bmatrix}.$$



Figur 4.17: *Durch die Transformation* **T** *werden die gemessenen Punkte so über die Referenzpunkte gelegt, dass die Summe der Quadrate der verbleibenden Abstände minimal wird.*

Das Quadrat des Abstandes zwischen einem Sensorpunkt und einem Referenzpunkt berechnet sich zu

$$\begin{aligned} d_i^2 &= \|\mathbf{p}_i - \mathbf{T}(\mathbf{q}_i)\|^2 \\ &= \left\| \begin{array}{l} p_{xi} - r_x - r_1 q_{xi} + r_2 q_{yi} \\ p_{yi} - r_y - r_2 q_{xi} - r_1 q_{yi} \end{array} \right\|^2 \\ &= \left\| \begin{array}{l} d_{xi} \\ d_{yi} \end{array} \right\|^2 = d_{xi}^2 + d_{yi}^2 \end{aligned}$$

Damit lautet unsere Bedingung für die Berechnung von $[r_x, r_y, r_1, r_2]$:

$$\sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} (d_{xi}^2 + d_{yi}^2) = Min \tag{4.14}$$

Da d_{xi} und d_{yi} beide linear von $[r_x, r_y, r_1, r_2]$ abhängig sind, kann dieses Problem mit der Gauss'schen Methode der kleinsten Quadrate gelöst werden. Wir setzen $\mathbf{y} = [p_{x1}, p_{y1}, p_{x2}, p_{y2} \ldots p_{xn}, p_{yn}]^T$ und $\mathbf{x} =$

$[r_x, r_y, r_1, r_2]^T$ und kommen so auf die gewünschte Form $\mathbf{y} = \mathbf{Ax}$:

$$\begin{bmatrix} p_{x1} \\ p_{y1} \\ p_{x2} \\ p_{y2} \\ \vdots \\ p_{xn} \\ p_{yn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & q_{x1} & -q_{y1} \\ 0 & 1 & q_{y1} & q_{x1} \\ 1 & 0 & q_{x2} & -q_{y2} \\ 0 & 1 & q_{y2} & q_{x2} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & q_{xn} & -q_{yn} \\ 0 & 1 & q_{yn} & q_{xn} \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_1 \\ r_2 \end{bmatrix} \tag{4.15}$$

Als Lösung der Gauss'schen Normalengleichung

$$\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{y} \tag{4.16}$$

erhält man die nach dem Kriterium der kleinsten Quadrate ausgeglichenen Werte $[x_1, ..., x_n]$

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} \tag{4.17}$$

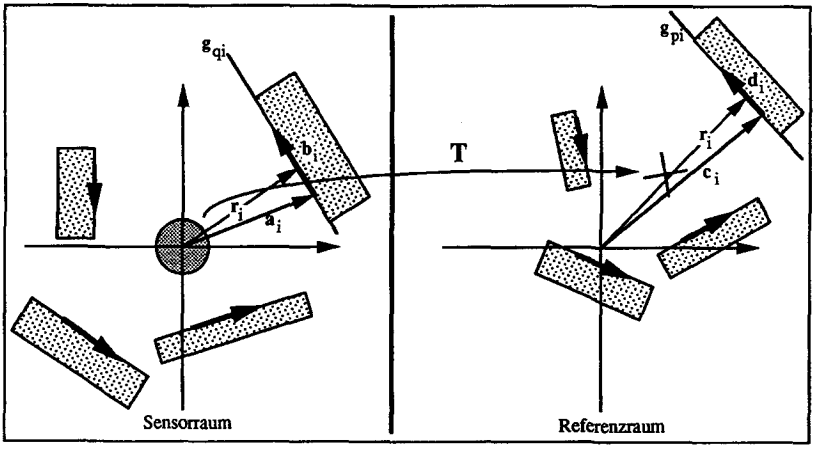
Möchte man die verschiedenen Punkte unterschiedlich gewichten, so werden die Elemente von $\mathbf{A}$ und $\mathbf{y}$ mit einem Gewichtungsfaktor γ_i multipliziert. Man erhält dann $\tilde{\mathbf{A}} = [\tilde{a}_{ij}]$ mit $\tilde{a}_{ij} = \gamma_i a_{ij}$, sowie $\tilde{\mathbf{y}} = [\tilde{y}_i]$ mit $\tilde{y}_i = \gamma_i y_i$.

$$\mathbf{x} = (\tilde{\mathbf{A}}^T\tilde{\mathbf{A}})^{-1}\tilde{\mathbf{A}}^T\tilde{\mathbf{y}} \tag{4.18}$$

Der Gewichtungsfaktor γ_i wird je nach Genauigkeit der Positionsangabe eines Punktes gross (> 1) oder klein ($0 \,..\, 1$) gewählt und ist in unserem Falle von den Messunsicherheiten σ_r und σ_ϕ abhängig.

Geraden Für die Berechnung der Roboterposition aus n Sensorgeraden und n Referenzgeraden kann ein ähnliches Verfahren verwendet werden wie bei den Punkten. Weil der Abstand zwischen zwei Geraden nur definiert ist, wenn diese parallel sind, ergibt sich eine etwas andere Berechnung. Gegeben seien n gerichtete Sensorgeraden g_{qi}, die jede im *Referenzmodell* eine entsprechende, ebenfalls gerichtete Referenzgerade g_{pi} hat (siehe Figur 4.18). Gesucht ist wiederum eine Transformation $\mathbf{T}$, so dass gilt:

$$\sum_{i=1}^{n} \|d_{pi} - d_{qi}\|^2 = Min \tag{4.19}$$



Figur 4.18: *Durch die Transformation* **T** *werden die gerichteten Sensorgeraden so über die Referenzgeraden gelegt, dass die Summe der Quadrate der verbleibenden Abstände minimal wird.*

Die Geraden werden durch einen Basisvektor und einen Richtungsvektor dargestellt:

$$\begin{array}{lll} \textit{Sensorraum}: & g_{qi}: & \mathbf{r}_i = \mathbf{a}_i + u_i\mathbf{b}_i \\ \textit{Referenzraum}: & g_{pi}: & \mathbf{r}_i = \mathbf{c}_i + v_i\mathbf{d}_i \end{array}$$

Der Abstand eines beliebigen Punktes $\mathbf{p}$ zu einer Geraden g ($\mathbf{r} = \mathbf{a}+u\mathbf{b}$) ist

$$d = \frac{\begin{bmatrix} -b_y \\ b_x \end{bmatrix}\begin{bmatrix} p_x \\ p_y \end{bmatrix}}{\|\mathbf{b}\|} - \frac{\begin{bmatrix} -b_y \\ b_x \end{bmatrix}\begin{bmatrix} a_x \\ a_y \end{bmatrix}}{\|\mathbf{b}\|}$$

Wie erwähnt müssen zwei Geraden parallel sein, damit ein Abstand zwischen ihnen definiert werden kann. Dies zwingt uns, die Transformation **T** in zwei Schritten zu berechnen:

1. Berechnung der Drehung Θ
2. Berechnung der Verschiebung $\begin{bmatrix} r_x \\ r_y \end{bmatrix}$ und des Skalierungsfaktors s

Die Verdrehung Θ kann einfach berechnet werden, da sie unabhängig ist von den drei anderen Grössen:

$$\Theta = \frac{\sum_{i=1}^{n}(\angle \mathbf{d}_i - \angle \mathbf{b}_i)\gamma_i}{\sum_{i=1}^{n}\gamma_i}$$

γ_i ist wiederum ein Gewichtungsfaktor, in dem zum Beispiel die Länge der durch die Geraden dargestellten Wand mitberücksichtigt werden kann.
Für den zweiten Teil der Berechnung drehen wir zuerst jede Sensorgerade so um den Ursprung des Sensorkoordinatensystemes, dass sie parallel zur entsprechenden Modellgeraden zu liegen kommt. Mit diesen gedrehten Geraden führen wir eine Berechnung von $\mathbf{T}$ durch, die die Quadrate der Abstände minimiert.
Nach der Verdrehung der einzelnen Sensorgeraden schreiben wir:

$$\text{Gedrehtes Sensormodell:} \quad g_{qi}^{\square}: \quad \mathbf{r}_i^{\square} = \mathbf{a}_i^{\square} + u_i^{\square}\mathbf{b}_i^{\square}$$

Die zu minimierende Grösse lautet nun:

$$\sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} \|d_{pi} - d_{qi}^{\square}\|^2$$

mit

$$d_{pi} = \frac{c_{xi}d_{yi} - c_{yi}d_{xi}}{\|\mathbf{d}_i\|} \text{ und}$$

$$d_{qi}^{\square} = \frac{b_{yi}^{\square}r_x - b_{xi}^{\square}r_y}{\|\mathbf{b}_i^{\square}\|} + \frac{a_{xi}^{\square}b_{yi}^{\square} - a_{yi}^{\square}b_{xi}^{\square}}{\|\mathbf{b}_i^{\square}\|}s$$

Da d_i linear abhängig ist von r_x, r_y und s, kann dieses Problem ebenfalls mit der Methode der kleinsten Quadrate gelöst werden. Wir setzen diesmal $\mathbf{y}^T = [d_{p1}, d_{p2}...d_{pn}]$ und $\mathbf{x}^T = [r_x, r_y, s]$ und kommen so wieder

auf die Form $\mathbf{y} = \mathbf{A}\mathbf{x}$:

$$\begin{bmatrix} d_{p1} \\ d_{p2} \\ \vdots \\ d_{pn} \end{bmatrix} = \begin{bmatrix} \frac{b^{\square}_{y1}}{\|\mathbf{b}^{\square}_1\|} & \frac{-b^{\square}_{x1}}{\|\mathbf{b}^{\square}_1\|} & \frac{a^{\square}_{x1}b^{\square}_{y1} - a^{\square}_{y1}b^{\square}_{x1}}{\|\mathbf{b}^{\square}_1\|} \\ \frac{b^{\square}_{y2}}{\|\mathbf{b}^{\square}_2\|} & \frac{-b^{\square}_{x2}}{\|\mathbf{b}^{\square}_2\|} & \frac{a^{\square}_{x2}b^{\square}_{y2} - a^{\square}_{y2}b^{\square}_{x2}}{\|\mathbf{b}^{\square}_2\|} \\ \vdots & \vdots & \vdots \\ \frac{b^{\square}_{yn}}{\|\mathbf{b}^{\square}_n\|} & \frac{-b^{\square}_{xn}}{\|\mathbf{b}^{\square}_n\|} & \frac{a^{\square}_{xn}b^{\square}_{yn} - a^{\square}_{yn}b^{\square}_{xn}}{\|\mathbf{b}^{\square}_n\|} \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ s \end{bmatrix} \tag{4.20}$$

Punkte und Linien Wie gesagt ist es nicht möglich, eine Ausgleichsberechnung anzugeben, die Punkte und Linien gleichzeitig berücksichtigt. Um eine möglichst gute Schätzung der Roboterposition zu erhalten, berechnen wir deshalb diese zuerst auf Grund der Punkte allein und dann auf Grund der Linien allein. Diese beiden Resultate werden anschliessend durch Mittelwertbildung zum Endresultat verbunden.

1. Aus n_P Punktpaaren wird mittels Ausgleichsrechnung die Transformation $\mathbf{T}_P = [r_{xP}, r_{yP}, r_{1P}, r_{2P}]^T$, bzw. r_{xP}, r_{yP}, s_P und Θ_P berechnet.

2. Aus n_L Linienpaaren wird mittels Ausgleichsrechnung die Transformation $\mathbf{T}_L = [r_{xL}, r_{yL}, r_{1L}, r_{2L}]^T$ bzw. r_{xL}, r_{yL}, s_L und Θ_L berechnet.

3. Durch Mittelung der Transformationsparameter erhält man das Endresultat:

$$\begin{aligned} r_x &= \frac{n_P r_{xP} + n_L r_{xL}}{n_P + n_L} \\ r_y &= \frac{n_P r_{yP} + n_L r_{yL}}{n_P + n_L} \\ s &= \frac{n_P s_P + n_L s_L}{n_P + n_L} \\ \Theta &= \frac{n_P \Theta_P + n_L \Theta_L}{n_P + n_L} \end{aligned}$$

4.6 Ein Beispiel

Die Ausführungen in diesem Kapitel sollen nun an Hand eines konkreten Beispieles illustriert werden. Die Sensordaten wurden mit RAMSIS in unserem Labor aufgenommen und off-line weiterverarbeitet.

Die Umgebung

Ein Photo und eine Grundrissdarstellung des Labors sind in Fig. 4.19 und Fig. 4.20 dargestellt. Die Grundrissdarstellung entspricht dem in der Folge verwendeten *Referenzmodell*. Die für strukturierte Umgebungen typischen Elemente *Wand*, *Kante* und *Ecke* sind darin in ausreichender Anzahl vorhanden. Natürlich gibt es in diesem Labor auch viele andere Objekte, die im *Referenzmodell* keine Aufnahme finden, weil sie entweder von den Ultraschallsensoren nicht erkannt werden können, oder weil sie leicht verschiebbar sind und deshalb nicht für die Positionsbestimmung in Frage kommen.
Die Abmessungen des Raumes sind mit 6.5x9m klein im Verhältnis zur Grösse des Roboters, der einen Durchmesser von 1.2m hat. Man kann aber annehmen, dass eine grössere Umgebung aus einer Vielzahl solcher kleiner Räume zusammengesetzt sei.
Zu beachten ist, dass die Objekte des Labors aus den verschiedensten Materialien bestehen. W1 ist ein Fenstersims aus Holz, W6 ist eine Holzwand, W8 und W10 sind plastifizierte Schränke, W9 ist aus Blech, W14 ist eine verputzte Wand und W18 ist die Kante zweier nebeneinanderstehender Holztische. Die restlichen Objekte sind Kartonschachteln.

Bildung des Sensorbasierten Modelles

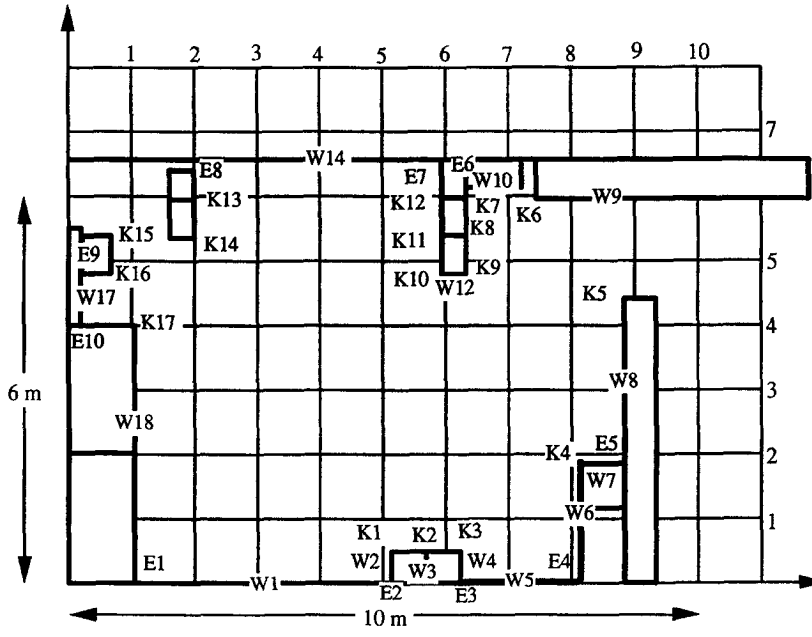
Der Roboter macht von einem festen, aber noch zu bestimmenden Standort aus Messungen in alle Richtungen. Die in Fig. 4.21 eingezeichneten Abstände sind bereits normalisiert und gefiltert, d.h. pro Richtungseinheit gibt es genau eine Messung und allfällige Ausreisser sind durch den Mittelwert seiner Nachbarn ersetzt.
Das Resultat der Objekterkennung ist ebenfalls bereits eingezeichnet. Dabei wurden die in Kapitel 4.2 hergeleiteten Regeln angewendet. Es

Figur 4.19: *Aufnahme unseres Labors*

ist ersichtlich, dass im Vordergrund stehende Objekte am besten erkannt werden. Damit ist auch klar, dass *Kanten* besser als die relativ gesehen immer hinten liegenden *Ecken* erkannt werden.
Das aus den erkannten Objekten abgeleitete *Sensorbasierte Modell* ist in Fig. 4.22 dargestellt.

Grobpositionierung mit dem Clusteringverfahren

Wie in Kapitel 4.4 beschrieben, werden aus allen möglichen Objektpaaren des *Sensorbasierten Modelles* alle Transformationen berechnet, welche diese Paare auf Objektpaare im *Referenzmodell* abbilden. Die Transformationen werden als Punkte in den R^4-Raum eingetragen. Ohne Einschränkungen wird die Anzahl dieser Punkte sehr gross, was die nachfolgende Clusterbildung erheblich verlangsamt. Deshalb werden nur plausibel erscheinende Punkte überhaupt berechnet. Die wichtigste Bedingung ist, dass der Skalierungsfaktor s im Bereich [0.97..1.03] liegen muss, was bei einem gut kalibrierten Sensorsystem sicher eine vernünftige Annahme ist. Des weiteren werden nur Punkte



Figur 4.20: *Die strukturierte Umgebung*

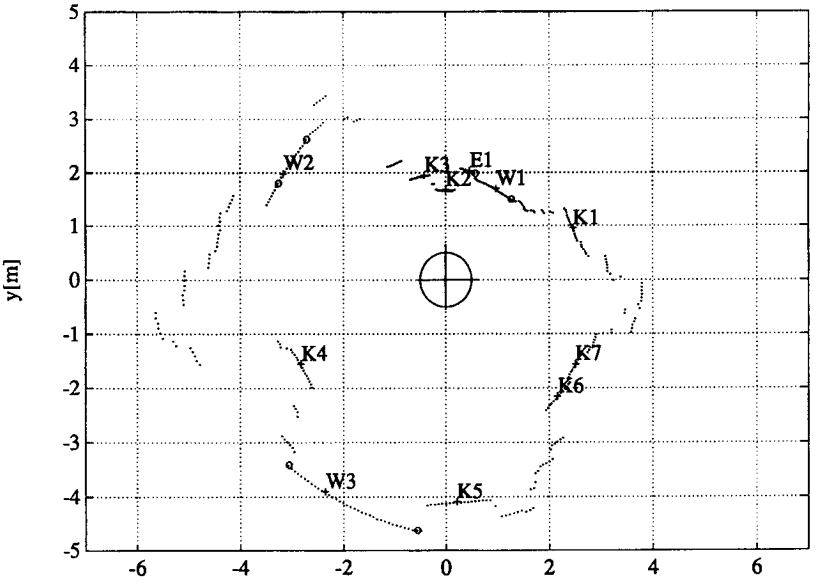
berücksichtigt, welche als Roboterposition überhaupt in Frage kommen, d.h. welche innerhalb der Grenzen des Raumes liegen.
In den Figuren 4.23 und 4.24 sind die Verteilung der Punkte und das Resultat der Clusterbildung eingetragen. Da die Punkte nicht wie in Kapitel 4.4 angenommen gleichmässig über den R^4-Raum verteilt sind, muss die dort angegebene Formel für $r_{Cluster}$ etwas abgeändert werden:

$$rCluster = \frac{1}{2}\sqrt[3]{\frac{1.0 * n_0}{n_{Punkte}}}$$

Statt der vierten wird nur die dritte Wurzel genommen. Damit wird berücksichtigt, dass bei festem s nur noch drei Parameter variieren. Dies bedingt aber auch eine Änderung in der Berechnung von Z_{r1j} und Z_{r2j}, weil sonst das Attribut ϕ ein zu grosses Gewicht erhält.

$$Z_{r1j} = \frac{X_{r1j} - 1.0}{6.28} \text{ bzw. } Z_{r2j} = \frac{X_{r2j} - 1.0}{6.28}$$

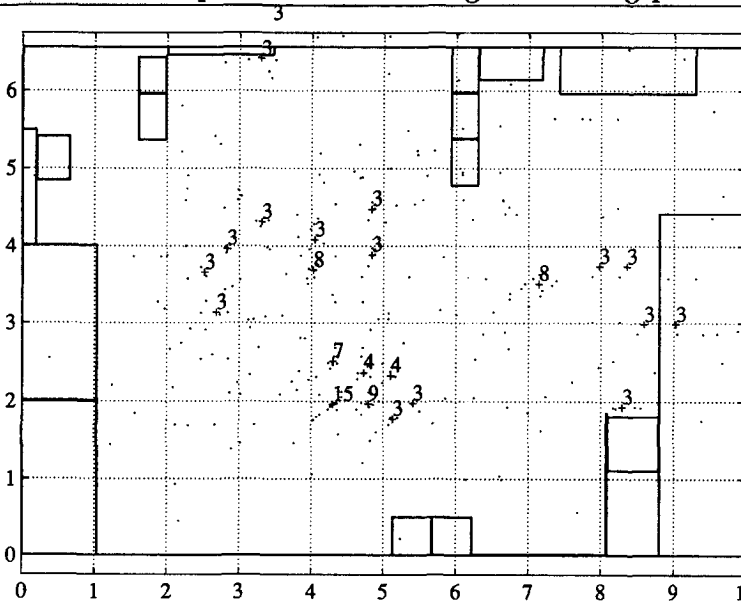
Neuer Divisor ist nicht mehr 2.0 sondern 6.28, was dem Umfang des Einheitskreises entspricht.



Figur 4.21: *Objekterkennung aus Ultraschallabstandsmessungen*

Art	Nr	r[m]	ϕ[°]	ϕ_A[°]	ϕ_E[°]	$\sigma_r[cm]$	σ_ϕ[°]
K	1	2.642	21	-	-	3.0	3.0
W	1	1.954	60	50	74	3.0	3.0
E	1	2.078	77	-	-	3.0	3.0
K	2	1.678	90	-	-	1.0	1.0
K	3	2.000	102	-	-	3.0	3.0
W	2	3.726	148	136	151	3.0	3.0
K	4	3.223	209	-	-	1.0	1.0
W	3	4.565	239	227	263	3.0	3.0
K	5	4.108	273	-	-	1.0	1.0
K	6	3.060	315	-	-	3.0	3.0
K	7	2.958	328	-	-	1.0	1.0

Figur 4.22: *Das Sensorbasierte Modell*

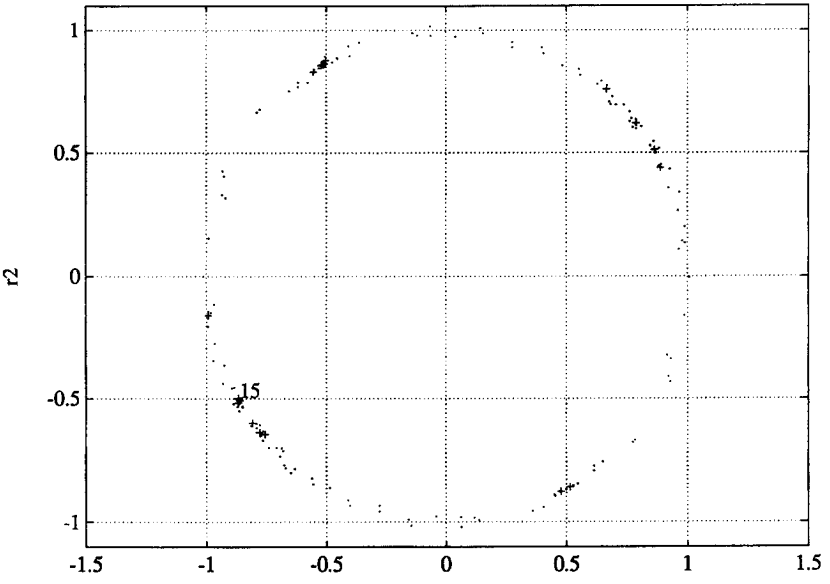


Figur 4.23: *Punkte im R^4-Raum: $r_x - r_y$*

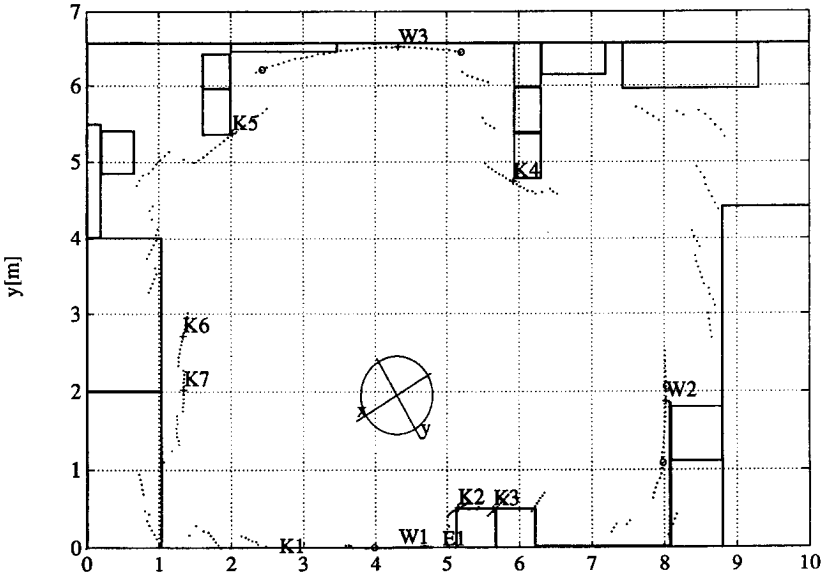
Das Resultat der Clusterbildung ist ebenfalls aus den Figuren 4.23 und 4.24 ersichtlich. Die grösste Gruppe umfasst 15 Punkte. Deren Mittelpunkt entspricht demnach (siehe Kapitel 4.4) der ungefähren Roboterposition. Sein Wert beträgt [4.300, 1.955, -0.862, -0.511]. Dies entspricht einem Lagewinkel von 210.7°.

Als nächstes müssen die erkannten Objekte den entsprechenden Objekten im *Referenzmodell* zugeordnet werden (Kapitel 4.5). Mit Hilfe von Fig. 4.25 kann diese Aufgabe auch von Hand gelöst werden. Als Ergebnis erhalten wir:

$m_{K1} = 0$, $m_{K2} = 1$, $m_{K3} = 2$, $m_{K4} = 10$,
$m_{K5} = 14$, $m_{K6} = 0$, $m_{K7} = 0$,
$m_{E1} = 0$ und
$m_{W1} = 1$, $m_{W2} = 6$, $m_{W3} = 14$.



Figur 4.24: *Punkte im R^4-Raum: $r_1 - r_2$*



Figur 4.25: *Roboterposition aus dem Clusteringverfahren*

Feinpositionierung mit Ausgleichsrechnung

Als letztes wird mittels Ausgleichsrechnung eine genauere Position des Roboters bestimmt. Aus den punktförmigen Objekten berechnen wir:

$$\begin{bmatrix} p_{x1} \\ p_{y1} \\ p_{x2} \\ p_{y2} \\ p_{x10} \\ p_{y10} \\ p_{x14} \\ p_{y14} \end{bmatrix} = \begin{bmatrix} 1 & 0 & q_{x2} & -q_{y2} \\ 0 & 1 & q_{y2} & q_{x2} \\ 1 & 0 & q_{x3} & -q_{y3} \\ 0 & 1 & q_{y3} & q_{x3} \\ 1 & 0 & q_{x4} & -q_{y4} \\ 0 & 1 & q_{y4} & q_{x4} \\ 1 & 0 & q_{x5} & -q_{y5} \\ 0 & 1 & q_{y5} & q_{x5} \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_1 \\ r_2 \end{bmatrix}$$

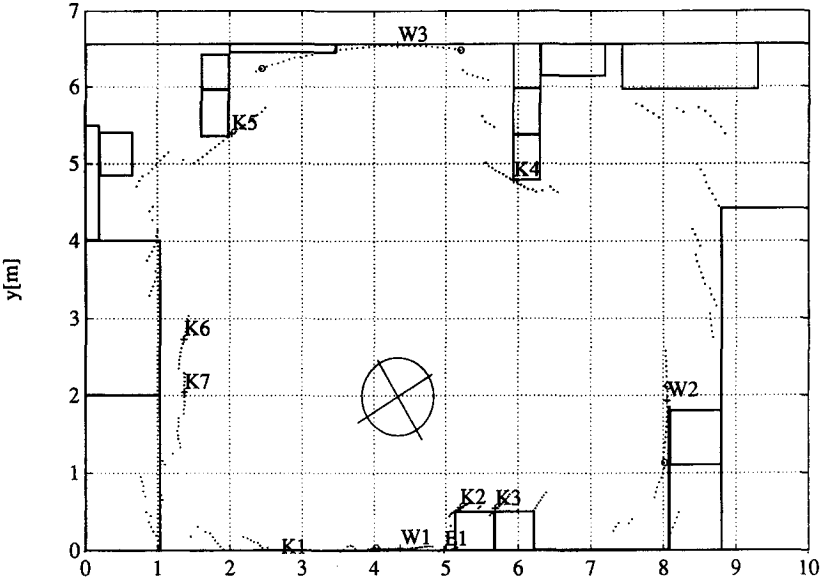
$$\begin{bmatrix} 5.123 \\ 0.505 \\ 5.673 \\ 0.505 \\ 5.930 \\ 4.790 \\ 1.992 \\ 5.360 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0.000 & -1.6781 \\ 0 & 1 & 1.678 & 0.000 \\ 1 & 0 & -0.416 & -1.956 \\ 0 & 1 & 1.956 & -0.416 \\ 1 & 0 & -2.819 & 1.563 \\ 0 & 1 & -1.563 & -2.819 \\ 1 & 0 & 0.215 & 4.102 \\ 0 & 1 & -4.102 & 0.215 \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_1 \\ r_2 \end{bmatrix}$$

Dies ergibt $[r_{xK}, r_{yK}, r_{1K}, r_{2K}]$ = [4.292, 1.963, -0.851, -0.510] bzw. $[r_{xK}, r_{yK}, \phi_K]$ = [4.292, 1.963, 210.96°].

Für die Berechnung der Roboterposition aus den Geraden berechnet man zuerst die Orientierung ϕ_W, was in unserem Beispiel 211° ergibt. Nach Anwendung der weiteren Schritte erhält man schliesslich

$$\begin{bmatrix} 0.000 \\ 8.067 \\ 6.570 \end{bmatrix} = \begin{bmatrix} 0.017 & -1.000 & 1.955 \\ 1.000 & -0.017 & 3.724 \\ 0.000 & 1.000 & 4.565 \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ s \end{bmatrix}$$

was auf $[r_{xW}, r_{yW}, s_W]$ = [4.391, 2.022, 0.993] führt.



Figur 4.26: *Roboterposition nach der Ausgleichsrechnung*

Aus den beiden oben erhaltenen Resultaten berechnen wir die Roboterposition:

$$r_x = \frac{4 * 4.292m + 3 * 4.391m}{7} = 4.334m$$

$$r_y = \frac{4 * 1.963m + 3 * 2.022m}{7} = 1.988\grave{m}$$

$$\phi = \frac{4 * 211° + 3 * 211°}{7} = 211°$$

Die Position wurde also nur noch um $4.334m - 4.300m = 3.4cm$ in x, um $1.988m - 1.955m = 3.3cm$ in y und um $211° - 210.7° = 0.3°$ in θ korrigiert. Je nach Genauigkeitsanforderungen könnte in diesem Falle auf die Nachkorrektur mittels Ausgleichsrechnung verzichtet werden.

Das Schlussresultat ist in Fig. 4.26 abgebildet. Rein optisch lässt sich erkennen, dass die Positionsgenauigkeit recht gut ist. Die Abweichung in x und in y liegt im Bereich von 2 bis 3 cm und der Orientierungsfehler ist kleiner als 2°.

Mit Hilfe von Fig. 4.26 kann auch die Qualität der Objekterkennung beurteilt werden. Kanten und Wände werden erstaunlich sicher

erkannt, während die Position der einzigen erkannten Ecke nur ungenau bestimmt werden konnte.

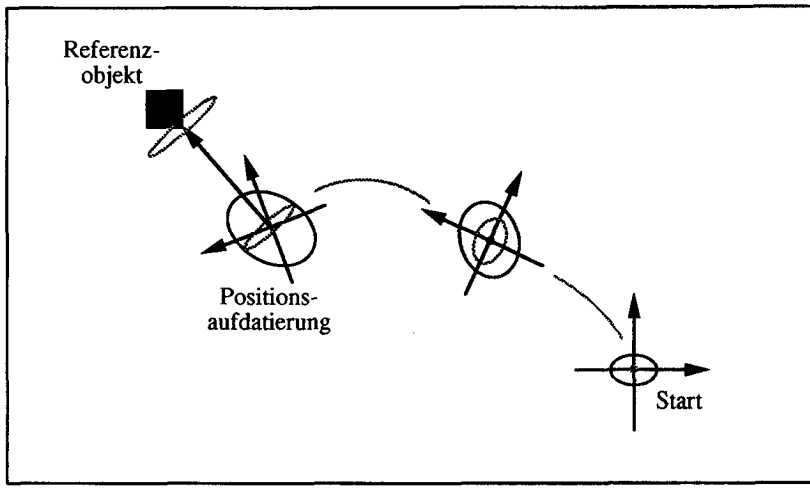
Kapitel 5

Positionsaufdatierung beim Fahren

5.1 Einführung

Die Problemstellung kann mit Hilfe von Fig. 1.4 formuliert werden. Gegeben sei eine durch Aufintegration der Radumdrehungen berechnete Schätzung $[x_{Odo}, y_{Odo}, \phi_{Odo}, \psi_{Odo}]$ der Roboterposition, sowie ein *Referenzmodell*, in welchem stationäre geometrische Objekte der Umgebung eingetragen sind. Gesucht ist eine Positionskorrektur $[\Delta x, \Delta y, \Delta\phi, \Delta\psi]$, so dass die Unterschiede zwischen erwarteten und gemessenen Abstands- und Winkelmessungen verringert werden.

Aufdatierung der Position heisst in unserem Fall, dass auf Grund von unsicheren Sensormessungen die Unsicherheit der Roboterposition verringert werden soll. Wie aus Fig. 5.1 zu ersehen ist, muss dabei mit drei Arten von Fehlern gerechnet werden: mit Fehlern bei der Berechnung der Anfangsposition, mit Fehlern bei der odometrischen Positionsbestimmung und schliesslich mit Fehlern bei der Abstandsmessung. Die Grösse der Fehler wird durch ein Fehlerellipsoid angedeutet. Ein relativ geringer Anfangsfehler wird beim Fahren wegen der unsicheren odometrischen Positionsbestimmung immer grösser, um nach jeder Messung wieder abzunehmen. Eine Aufdatierung hat so häufig zu er-



Figur 5.1: *Unsicherheiten in der Roboterposition*

folgen, dass der odometrische Positionsfehler eine gewisse Obergrenze nicht überschreitet. Diese hängt im wesentlichen von der geforderten Genauigkeit ab.

Ein Instrumentarium, welches die Darstellung und die Manipulation von unsicheren geometrischen Beziehungen ermöglicht, wurde von [Smith and Cheeseman 86] entwickelt. Diejenigen Teile daraus, die für uns nützlich sind, werden im nächsten Kapitel wiedergegeben. Die Relationen zwischen den einzelnen Koordinatensystemen werden dabei als normalverteilte Zufallsgrössen angenommen, deren Unsicherheiten mit einer Covarianzmatrix beschrieben werden. Die Grösse dieser Covarianzmatrix und damit der Unsicherheit wird durch Ellipsen dargestellt (Fig. 5.1). Mit den beiden Operationen *compound* (Addition zweier Transformationen) und *merge* (Verbindung zweier Transformationen) ist es möglich, aus jedem beliebigen Netzwerk von Relationen die resultierende Transformation mit zugehöriger Unsicherheit zu berechnen. Aus Fig. 5.1 sieht man, dass sich die Start- und die Odometriefehler addieren (*compound*), während durch die Sensormessung möglicherweise eine Reduktion der Unsicherheit (*merge*) erfolgt.

Auf den gleichen Ideen basiert die Kalmanfilterung, die für das Schätzen von Zuständen eingesetzt wird. Ausgehend von einer Schätzung des Anfangszustandes vergrössert sich dessen Unsicherheit

mit zunehmender Zeit, um nach jeder Messung wieder abzunehmen. Im Unterschied zu [Smith and Cheeseman 86] kann eine Korrektur auch erfolgen, wenn nur eine Messung in eine bestimmte Richtung vorliegt. Die Hauptschwierigkeit ist dabei nicht die Anwendung des Algorithmus', sondern die korrekte Modellierung der Sensor- und Modellunsicherheiten. Weniger kritisch, aber ebenfalls wichtig, ist die Schätzung der Anfangsunsicherheit.

Über die Positionsaufdatierung von mobilen Robotern gibt es relativ wenige Arbeiten. Der Einfachheit halber wird meistens angenommen, dass die Methoden, die im Stillstand funktionieren, auch für unterwegs geeignet seien. Dass dies nicht der Fall ist, ergibt sich allein schon auf Grund ihres hohen Zeitbedarfes. Intensiver mit dem Problem beschäftigt hat man sich in der Luft- und Raumfahrt (siehe z.B. [Hostetler 83]). Die dabei entwickelten Ideen können auf das Aufdatierungsproblem von mobilen Robotern übertragen werden.

Ein vielversprechender Ansatz ist in [Wiklund 89] beschrieben. Darin werden für die Positionsaufdatierung Winkelmessungen zu vorgegebenen Markierungen verwendet. Diese Lösung kommt für uns aber höchstens für das Anfahren einer Andockstelle in Frage, da wir sonst keine Veränderungen der Umgebung vornehmen wollen. Allerdings könnte das Verfahren allgemeiner verwendet werden, wenn die Winkel zu vorhandenen Umgebungsmerkmalen statt zu Markierungen gemessen würden.

In [Steer 89] ist ein Versuch beschrieben, die Position eines Fahrzeuges mit Hilfe von Ultraschall zu korrigieren. Aus dem Unterschied zwischen erwarteten und gemessenem Abständen werden Differenzvektoren gebildet, die dann zu einer Positionskorrektur weiterverarbeitet werden. Der vorgeschlagene Ansatz ist sinnvoll. Wesentliche Probleme werden aber nicht angesprochen, weil nur einfachste Beispiele beschrieben werden.

[Leonard and DurrantWhyte 91] arbeiten ebenfalls mit Ultraschallabstandsmessungen für die Korrektur von Odometriefehlern. Dabei gehen sie ähnlich vor wie wir. Sie benutzen ein Kalmanfilter, wozu Sie die Positionsunsicherheit und die Sensorfehler ebenfalls mittels Covarianzmatrizen beschreiben müssen. Die Unterschiede liegen in der Berechnung des Odometriefehlers, im Suchen von gültigen Messungen und schliesslich bei der Art der Orientierungskorrektur. Zudem lassen

sich ihre Ergebnisse nicht richtig vergleichen, da sie nur im Stillstand messen.

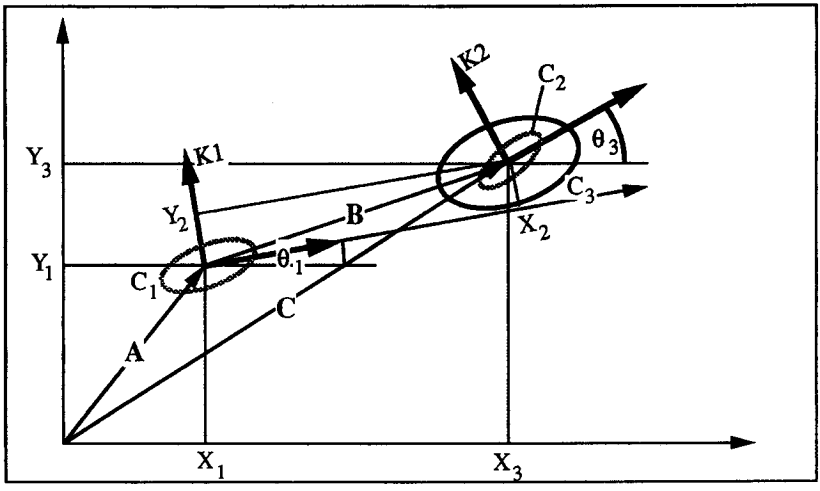
Vielversprechend schien uns der Ansatz, die Differenzen von erwarteten und gemessenen Abständen als Zeitreihen darzustellen und mittels Polynomen zu approximieren. Entsprechende Versuche waren aber wenig erfolgreich. Vor allem die Tatsache, dass die Messdaten in unregelmässigen Zeitabständen anfallen, verunmöglichte eine brauchbare Positionskorrektur.

Wie bereits angedeutet spielt der Rechenzeitbedarf bei der Positionsaufdatierung eine wichtige Rolle, da sie in Echtzeit ablaufen soll. Ein grosser Teil davon entsteht dadurch, dass für jede Abstandsmessung das passende Objekt im Referenzmodell gesucht werden muss. Naheliegend wäre es deshalb, ein Verfahren zu wählen, das ohne Referenzmodell auskommt. Leider ist dies aber aus prinzipiellen Überlegungen nicht möglich. In jedem Fall müssten für eine genaue Positionsbestimmung über einen längeren Zeitraum verschwindend kleine Sensorfehler angenommen werden, da bei jeder Positionsbestimmungsmethode, die ohne Modell arbeitet, die Sensorfehler aufsummiert werden.

Die nachfolgenden Kapitel beschäftigen sich mit der Darstellung und Manipulation von geometrischen Unsicherheiten, mit der Modellierung des Odometriefehlers, mit der Berechnung der Abstandswerte, sowie mit der Anwendung des Kalmanfilters für die Positionsaufdatierung. Abschliessend werden an Beispielen die erreichten Resultate erläutert.

5.2 Rechnen mit geometrischen Unsicherheiten

Eine unsichere oder approximative Transformation (AT) besteht aus einer geschätzten mittleren Verschiebung eines Koordintensystemes zu einem anderen, sowie aus einer Covarianzmatrix, mit welcher die Unsicherheit der Schätzung dargestellt wird. ATs entstehen typischerweise aus Sensormessungen oder aus Relativbewegungen. Bildlich kann die Unsicherheit durch eine Ellipse dargestellt werden, deren Mittelpunkt dem Nominalwert der Schätzung entspricht, und deren Grenze ein Gebiet umschliesst, in welchem der tatsächliche Wert mit einer definierten



Figur 5.2: *Addition approximativer Transformationen*

Wahrscheinlichkeit liegt.

Um aus einem Netzwerk von unsicheren Transformationen eine resultierende Transformation mit zugehöriger Unsicherheit berechnen zu können, sind die Basisoperationen *compound*, *merge* und *reverse* zu definieren, welche zwei serielle bzw. parallele Transformationen zu einer einzigen Transformation vereinfachen, bzw. die Transformationsrichtung umkehren.

Für die Positionsaufdatierung werden wir nur die *compound*-Operation benötigen.

5.2.1 Verbindung serieller ATs - *compound*

Unser Ziel ist es, die Koordinaten von K2 (siehe Fig. 5.2) im globalen Koordinatensystem auszudrücken. Dazu müssen die Transformationen $\mathbf{A}(X_1, Y_1, \Theta_1)$ und $\mathbf{B}(X_2, Y_2, \Theta_2)$ zur Transformation $\mathbf{C}(X_3, Y_3, \Theta_3)$ verbunden werden. Die exakte Lösung lautet:

$$\begin{aligned} X_3 &= f(X_1, Y_1, \Theta_1, X_2, Y_2, \Theta_2) \\ &= X_2 \cos\Theta_1 - Y_2 \sin\Theta_1 + X_1 \\ Y_3 &= g(X_1, Y_1, \Theta_1, X_2, Y_2, \Theta_2) \end{aligned} \tag{5.1}$$

$$
\begin{aligned}
&= X_2 \sin\Theta_1 + Y_2 \cos\Theta_1 + Y_1 && (5.2)\\
\Theta_3 &= h(X_1, Y_1, \Theta_1, X_2, Y_2, \Theta_2) \\
&= \Theta_1 + \Theta_2 && (5.3)
\end{aligned}
$$

Wir möchten die Mittelwerte und die Covarianzen dieser drei Funktionen schätzen. Die Variabeln werden nun als Zufallsvariabeln angenommen. Die Funktionen werden durch eine Taylorentwicklung 1.Ordnung um die Mittelwerte der Variabeln beschrieben, z.B. $\hat{X}_3 \approx f(\hat{X}_1, \hat{Y}_1, \hat{\Theta}_1, \hat{X}_2, \hat{Y}_2, \hat{\Theta}_2)$. Neben der geschätzten Verschiebung gehört auch eine Covarianzmatrix zu jeder AT. Um die Covarianzmatrix schätzen zu können, drücken wir die Taylorentwicklung in Matrixform aus:

$$
\begin{pmatrix} \Delta X_3 \\ \Delta Y_3 \\ \Delta\Theta_3 \end{pmatrix} \approx \mathbf{J} * (\Delta X_1, \Delta Y_1, \Delta\Theta_1, \Delta X_2, \Delta Y_2, \Delta\Theta_2)^T \qquad (5.4)
$$

J ist die (3x6)-Jacobimatrix der Transformation um die Mittelwerte der Variabeln:

$$
\mathbf{J} = \left(\begin{array}{ccc|ccc}
\frac{\partial f}{\partial X_1} & \frac{\partial f}{\partial Y_1} & \frac{\partial f}{\partial \Theta_1} & \frac{\partial f}{\partial X_2} & \frac{\partial f}{\partial Y_2} & \frac{\partial f}{\partial \Theta_2} \\
\frac{\partial g}{\partial X_1} & \frac{\partial g}{\partial Y_1} & \frac{\partial g}{\partial \Theta_1} & \frac{\partial g}{\partial X_2} & \frac{\partial g}{\partial Y_2} & \frac{\partial g}{\partial \Theta_2} \\
\frac{\partial h}{\partial X_1} & \frac{\partial h}{\partial Y_1} & \frac{\partial h}{\partial \Theta_1} & \frac{\partial h}{\partial X_2} & \frac{\partial h}{\partial Y_2} & \frac{\partial h}{\partial \Theta_2}
\end{array}\right) \qquad (5.5)
$$

$$
= \left(\begin{array}{ccc|ccc}
1 & 0 & -(Y_3 - Y_1) & \cos\Theta_1 & -\sin\Theta_1 & 0 \\
0 & 1 & (X_3 - X_1) & \sin\Theta_1 & \cos\Theta_1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1
\end{array}\right) \qquad (5.6)
$$

$$
= [\mathbf{H} \mid \mathbf{K}] \qquad (5.7)
$$

Covarianz ist definiert als der Erwartungswert der quadrierten Abweichungen. Wir quadrieren beide Seiten von Gl. 5.4 durch Multiplikation der beiden Seiten der Gleichung mit ihren zugehörigen Transponierten. Nehmen wir den Erwartungswert des Resultates, so erhalten wir die Covarianzmatrix $\mathbf{C}_3$:

$$
\mathbf{C}_3 = E\left[\begin{bmatrix} \Delta X_3 \\ \Delta Y_3 \\ \Delta\Theta_3 \end{bmatrix} \begin{bmatrix} \Delta X_3 \\ \Delta Y_3 \\ \Delta\Theta_3 \end{bmatrix}^T\right]
$$

$$\approx \mathbf{J}\begin{pmatrix} \mathbf{C}_1 & 0 \\ 0 & \mathbf{C}_2 \end{pmatrix}\mathbf{J}^T = \mathbf{H}\mathbf{C}_1\mathbf{H}^T + \mathbf{K}\mathbf{C}_2\mathbf{K}^T \qquad (5.8)$$

Die (3x3)-Matrix $\mathbf{C}_3$ beschreibt die Covarianzen der Koordinaten von K2 im globalen Koordinatensystem in Abhängigkeit der Covarianzmatrizen $\mathbf{C}_1$ und $\mathbf{C}_2$, welche die Fehler in den Variabeln von **A** und von **B** ausdrücken. Man beachte, dass wenn Θ_1 und Θ_2 fehlerfrei gemessen werden können, dass dann die Dimensionen von **H**, **K** und von den zugehörigen Covarianzmatrizen reduziert werden können. Dies geschieht durch Wegstreichen derjenigen Reihen und Kolonnen, die mit Θ_1 und Θ_2 assoziiert sind. Wenn keine Winkelfehler vorhanden sind, werden obige Gleichungen linear in den Zufallsvariabeln. Damit ist das obige Resultat nicht länger eine Näherung, sondern das erste und das zweite Moment werden korrekt transformiert.

5.2.2 Darstellung einer AT in einem anderen Koordinatensystem

Als Beispiel soll gezeigt werden, wie eine in Polarkoordinaten (r, ϕ, Θ) ausgedrückte AT in eine äquivalente kartesische Form gebracht werden kann. Diese Abbildung ist notwendig, wenn z.B. eine Kamera den Ort eines Objektes in Polarkoordinaten angibt, und wenn auch der Kamerafehler in Polarkoordinaten gegeben ist, die Weiterverarbeitung des Resultates aber in kartesischen Koordinaten erfolgen soll. Die Transformation ist gegeben durch:

$$\hat{x} \approx f_1(\hat{r}, \hat{\phi}) = \hat{r}\cos\hat{\phi} \qquad (5.9)$$

$$\hat{y} \approx f_2(\hat{r}, \hat{\phi}) = \hat{r}\sin\hat{\phi} \qquad (5.10)$$

$$\hat{\Theta} = f_3(\hat{\Theta}) = \hat{\Theta} \qquad (5.11)$$

J, die Jacobimatrix der Transformation, ist gegeben durch:

$$\mathbf{J} = \begin{pmatrix} \frac{\partial f_1}{\partial r} & \frac{\partial f_1}{\partial \phi} & \frac{\partial f_1}{\partial \Theta} \\ \frac{\partial f_2}{\partial r} & \frac{\partial f_2}{\partial \phi} & \frac{\partial f_2}{\partial \Theta} \\ \frac{\partial f_3}{\partial r} & \frac{\partial f_3}{\partial \phi} & \frac{\partial f_3}{\partial \Theta} \end{pmatrix} = \begin{pmatrix} \frac{x}{r} & -y & 0 \\ \frac{y}{r} & x & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (5.12)$$

In einer Näherung 1.Ordnung kann die kartesische Covarianzmatrix ausgedrückt werden:

$$\mathbf{C}(x, y, \Theta) \approx \mathbf{J} * \mathbf{C}(r, \phi, \Theta) * \mathbf{J}^T \tag{5.13}$$

5.2.3 Das Fehlerellipsoid

Zur Beantwortung der Frage, ob die Wahrscheinlichkeit, dass ein bestimmtes Objekt an einem bestimmten Ort ist oder nicht, einen gewissen Wert überschreitet oder nicht, setzen wir voraus, dass die Wahrscheinlichkeitsverteilung der Objektposition eine mehrdimensionale Gaussverteilung ist. Die allgemeine Form ist gegeben durch:

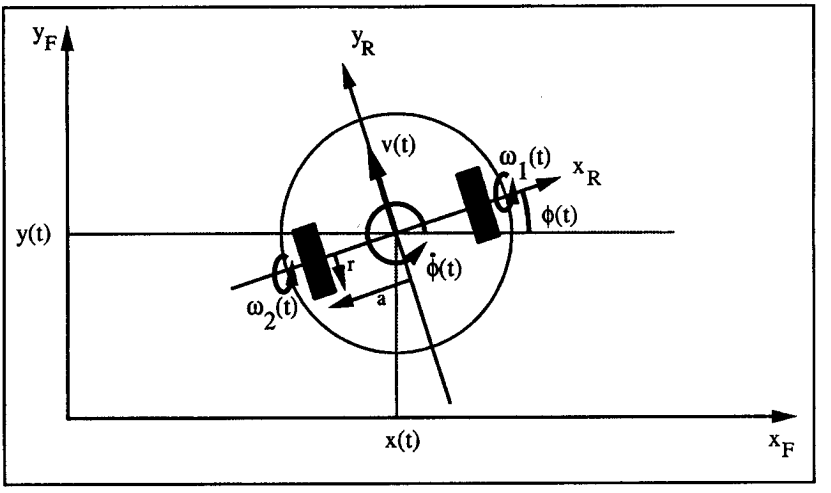
$$P(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n \det \mathbf{C}}} e^{-\frac{1}{2}[(\mathbf{x}-\hat{\mathbf{x}})^T \mathbf{C}^{-1}(\mathbf{x}-\hat{\mathbf{x}})]} \tag{5.14}$$

Dabei ist n die Dimension, C die Covarianzmatrix, $\hat{\mathbf{x}}$ der nominelle Mittelwert und $\mathbf{x}$ ein Vektor zu einem bestimmten Punkt. Die Konturen gleicher Wahrscheinlichkeit dieser Verteilung bilden Ellipsoide im n-dimensionalen Raum. Deren Zentrum liegt bei $\hat{\mathbf{x}}$. Sie können durch die Formel

$$(\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{C}^{-1} (\mathbf{x} - \hat{\mathbf{x}}) = k^2 \tag{5.15}$$

beschrieben werden. Dabei wird durch k eine Vertrauensgrenze definiert. Für den Spezialfall, dass aus einer dreidimensionalen Covarianzmatrix $\mathbf{C}_3(x, y, \phi)$ die zweidimensionale Fehlerellipse in x und y gesucht wird, muss zuerst die zweidimensionale Covarianzmatrix $\mathbf{C}_2(x, y)$ aus $\mathbf{C}_3(x, y, \phi)$ gebildet werden. Dies geschieht durch Wegstreichen der Kolonne und der Reihe mit der nicht gewollten Variable. Anschliessend wird k berechnet:

$$Pr(x, y \in Ellipse) = 1 - e^{\frac{-k^2}{2}} \text{ bzw.} k^2 = -2\ln(1 - Pr) \tag{5.16}$$



Figur 5.3: *Kinematik des Roboters mit zwei Antriebsrädern*

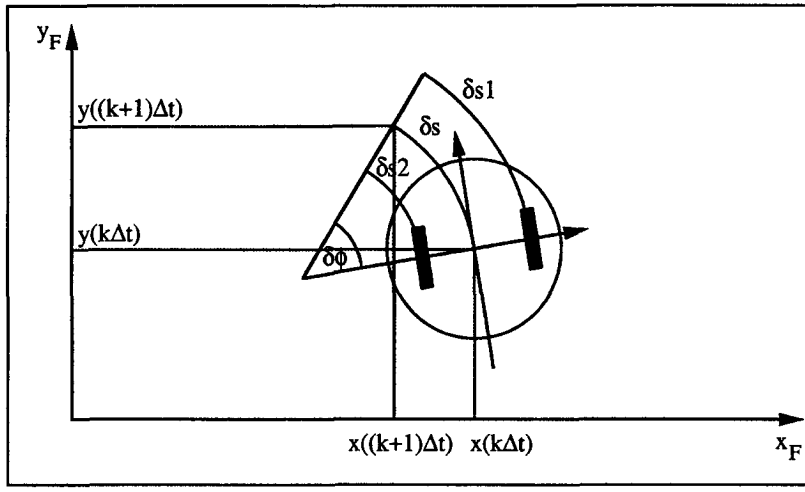
5.3 Schätzung der odometrischen Positionsunsicherheit

Unser Roboter wird durch zwei Räder angetrieben, welche sich mit einer Winkelgeschwindigkeit ω_1 bzw. ω_2 drehen (Fig. 5.3). Die Radbewegungen $\delta s1 = \omega_1 * r * \Delta t$ und $\delta s2 = \omega_2 * r * \Delta t$ werden mit Inkrementalencodern erfasst. Daraus lassen sich folgende Grössen ableiten:

$$\begin{aligned} \delta s_{k+1} &= \frac{\delta s1_{k+1} + \delta s2_{k+1}}{2} \\ \delta \phi_{k+1} &= \frac{\delta s1_{k+1} - \delta s2_{k+1}}{2a} \\ \tilde{\phi}_{k+1} &= \phi_k + \frac{\delta \phi_{k+1}}{2} \end{aligned}$$

Die Roboterposition wird durch Integration folgender Gleichung berechnet (Fig 5.4):

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \Delta \mathbf{s}_{k+1}) \tag{5.17}$$



Figur 5.4: *Weg des Roboters zwischen zwei Abtastpunkten*

$$= \mathbf{x}_k + \begin{bmatrix} -\frac{\sin(\tilde{\phi}_{k+1})}{2} & -\frac{\sin(\tilde{\phi}_{k+1})}{2} \\ \frac{\cos(\tilde{\phi}_{k+1})}{2} & \frac{\cos(\tilde{\phi}_{k+1})}{2} \\ \frac{1}{2a} & -\frac{1}{2a} \end{bmatrix} * \Delta \mathbf{s}_{k+1}$$

wobei $\mathbf{x}_k = [x_k, y_k, \phi_k]^T$ die alte Roboterposition und $\Delta \mathbf{s}_{k+1} = [\delta s1_{k+1}, \delta s2_{k+1}]^T$ den Weg der beiden Räder beschreiben.

Aus verschiedenen Gründen kann $\Delta \mathbf{s}_{k+1}$ nicht fehlerfrei gemessen werden:

- **Schlupf:** Bei einem Gleiten zwischen Rad und Boden stimmt $\delta s_i = \omega_i * r * \Delta t$ nicht mehr. Da der Schlupf von der Beschleunigung des Roboters und von der Bodenbeschaffenheit des Raumes abhängig ist, kann dieser Effekt nur mit zusätzlichen Messungen erfasst werden.

- **Skalierungsfehler:** Der Radius des Rades kann nicht exakt angegeben werden, weil sich dieser wegen der Elastizität des Rades je nach Belastung ändert.

- **Quantisierungsfehler:** Die Auflösung der Inkrementalencoder ist beschränkt. Dadurch wird die Position nicht aus dem exakten, sondern aus einem auf- oder abgerundeten Wert berechnet. Durch entsprechende Wahl der Encoder kann dieser Fehler gering gehalten werden.
- **Bodenunebenheiten:** Die Positionsberechnungen gehen von einem absolut ebenen Boden ohne Verunreinigungen aus. Ist der Boden uneben oder liegen irgendwelche kleine Teilchen auf dem Boden, so ist der vom Roboter zurückgelegte Weg länger als dessen Projektion auf die Ebene.

Um alle diese Fehler zu berücksichtigen, wird $\Delta \mathbf{s}_{k+1}$ als normalverteilte Zufallsgrösse angenommen, deren Unsicherheit von der Covarianzmatrix $\mathbf{C}(\Delta \mathbf{s}_{k+1})$ beschrieben wird.

Aus den Varianzen von $\delta s1$ und $\delta s2$ kann eine Schätzung der Varianz der Roboterposition berechnet werden. Wir werden diese Berechnung nach der in Kapitel 5.2.2 vorgestellten Methode durchführen. Eine genauere, aber aufwendigere Berechnungsmethode, wie sie z.B. [Wang 88] vorschlägt, ist hier nicht nötig, da diese erst bei Winkeländerungen von $\delta\phi > 10°$ wesentliche Vorteile bringt. Für uns heisst das aber, dass die Neuberechnung der Positionsunsicherheit zu erfolgen hat, bevor sich der Roboter um mehr als 10° gedreht hat.

Nimmt man an, dass die Fehler in $\delta s1$ und $\delta s2$ zu einem globalen Fehler führen, welcher proportional ist zur Distanz, die jedes einzelne Rad zurücklegt, so können wir die Positionsunsicherheit folgendermassen berechnen (siehe auch [Moutarlier and Chatila 89]):

$$\mathbf{C}(\mathbf{x}_{k+1}) \simeq \mathbf{J}(\mathbf{x}_k)\mathbf{C}(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)^T + \mathbf{J}(\Delta \mathbf{s}_{k+1})\mathbf{C}(\Delta \mathbf{s}_{k+1})\mathbf{J}(\Delta \mathbf{s}_{k+1})^T \tag{5.18}$$

$\mathbf{C}(\Delta \mathbf{s}_{k+1})$ ist die Covarianzmatrix der Radfehler und beträgt

$$\mathbf{C}(\Delta \mathbf{s}_{k+1}) = \alpha^2 * \begin{bmatrix} (\delta s1_{k+1})^2 & 0 \\ 0 & (\delta s2_{k+1})^2 \end{bmatrix} \tag{5.19}$$

α ist ein Parameter, welcher das Verhältnis zwischen dem gefahrenen Weg und der Vergrösserung der Positionsunsicherheit beschreibt.

$\mathbf{J}(\mathbf{x}_k)$ ist die Jacobimatrix von F um die Roboterposition $\mathbf{x}_k$ und $\mathbf{J}(\Delta \mathbf{s}_{k+1})$ ist die Jacobimatrix von F um die Verschiebung der Räder

Δs_{k+1}:

$$\mathbf{J}(\mathbf{x}_k) = \begin{bmatrix} 1 & 0 & -\delta s_{k+1} * \cos\left(\tilde{\phi}_{k+1}\right) \\ 0 & 1 & -\delta s_{k+1} * \sin\left(\tilde{\phi}_{k+1}\right) \\ 0 & 0 & 1 \end{bmatrix} \tag{5.20}$$

und

$$\mathbf{J}(\Delta s_{k+1}) = \begin{bmatrix} j_{11} & j_{12} \\ j_{21} & j_{22} \\ \frac{1}{2a} & -\frac{1}{2a} \end{bmatrix} \tag{5.21}$$

mit

$$\begin{aligned} j_{11} &= -\frac{1}{2} * \sin\left(\tilde{\phi}_{k+1}\right) - \frac{\delta s_{k+1}}{4a} * \cos\left(\tilde{\phi}_{k+1}\right) \\ j_{12} &= -\frac{1}{2} * \sin\left(\tilde{\phi}_{k+1}\right) + \frac{\delta s_{k+1}}{4a} * \cos\left(\tilde{\phi}_{k+1}\right) \\ j_{21} &= +\frac{1}{2} * \cos\left(\tilde{\phi}_{k+1}\right) - \frac{\delta s_{k+1}}{4a} * \sin\left(\tilde{\phi}_{k+1}\right) \\ j_{22} &= +\frac{1}{2} * \cos\left(\tilde{\phi}_{k+1}\right) + \frac{\delta s_{k+1}}{4a} * \sin\left(\tilde{\phi}_{k+1}\right) \end{aligned}$$

Es muss nochmals betont werden, dass diese Berechnungen nur richtig sind, falls der Radschlupf kompensiert wird und dessen Auswirkung hier als normalverteilt angenommen werden kann. Ohne Schlupfkompensation muss man sich damit behelfen, im Sinne einer worst-case Abschätzung α so gross zu wählen, dass der Schlupf in $\mathbf{C}(\Delta s_{k+1})$ enthalten ist, was aber bei normaler Fahrweise zu einer schlechten Abschätzung der Genauigkeit der Roboterposition führt.

Basiert die Berechnung des Odometriefehlers auf Annahmen über $\delta s1$ und $\delta s2$, so wird die Unsicherheit in ϕ im allgemeinen eher zu gross geschätzt. Diesem Problem kann man damit begegnen, dass für die Abschätzung des Odometriefehlers nicht Annahmen über $\delta s1$ und $\delta s2$, sondern über δs und $\delta\phi$ verwendet werden. Ausgehend von den Gleichungen für die Bestimmung der Roboterposition

$$x_{k+1} = x_k + f(\delta s_{k+1}, \delta\phi_{k+1}) = x_k - \delta s_{k+1} * \sin(\tilde{\phi}_{k+1}) \tag{5.22}$$

$$y_{k+1} = y_k + g(\delta s_{k+1}, \delta\phi_{k+1}) = y_k + \delta s_{k+1} * \cos(\tilde{\phi}_{k+1}) \tag{5.23}$$

$$\phi_{k+1} = \phi_k + h(\delta s_{k+1}, \delta\phi_{k+1}) = \phi_k + \delta\phi_{k+1} \tag{5.24}$$

erhalten wir in diesem Falle die Jacobimatrix $\mathbf{J}(\delta s_{k+1}, \delta\phi_{k+1})$:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f}{\partial \delta s_{k+1}} & \frac{\partial f}{\partial \delta \phi_{k+1}} \\ \frac{\partial g}{\partial \delta s_{k+1}} & \frac{\partial g}{\partial \delta \phi_{k+1}} \\ \frac{\partial h}{\partial \delta s_{k+1}} & \frac{\partial h}{\partial \delta \phi_{k+1}} \end{bmatrix} = \begin{bmatrix} -\sin(\tilde{\phi}_{k+1}) & -\frac{\delta s_{k+1}}{2} * \cos(\tilde{\phi}_{k+1}) \\ \cos(\tilde{\phi}_{k+1}) & -\frac{\delta s_{k+1}}{2} * \sin(\tilde{\phi}_{k+1}) \\ 0 & 1 \end{bmatrix}$$

und mit

$$\mathbf{C}(\delta s_{k+1}, \delta\phi_{k+1}) = \begin{bmatrix} \sigma^2_{\delta s_{k+1}} & 0 \\ 0 & \sigma^2_{\delta\phi_{k+1}} \end{bmatrix}$$

erhalten wir wieder

$$\begin{aligned} \mathbf{C}(\mathbf{x}_{k+1}) \simeq\ & \mathbf{J}(\mathbf{x}_k)\mathbf{C}(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)^T \\ & + \mathbf{J}^T(\delta s_{k+1}, \delta\phi_{k+1}) * \mathbf{C}(\delta s_{k+1}, \delta\phi_{k+1}) * \mathbf{J}(\delta s_{k+1}, \delta\phi_{k+1}) \end{aligned}$$

$(\sigma_{\delta s1}, \sigma_{\delta s2})$ und $(\sigma_{\delta s}, \sigma_{\delta\phi})$ sind nicht unabhängig voneinander, sondern über die Beziehungen

$$\sigma^2_{\delta s} = \frac{\sigma^2_{\delta s1} + \sigma^2_{\delta s2}}{4} \text{ und } \sigma^2_{\delta\phi} = \frac{\sigma^2_{\delta s1} + \sigma^2_{\delta s2}}{(2a)^2} \tag{5.25}$$

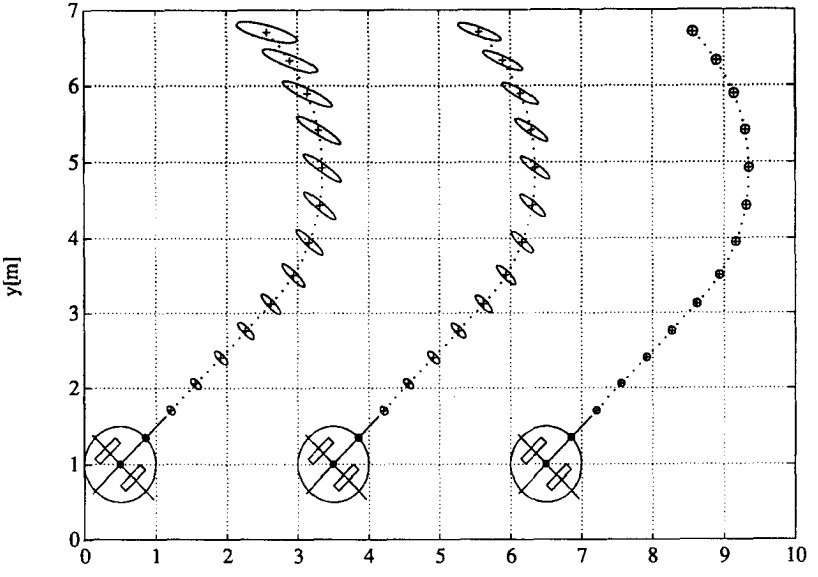
miteinander verbunden. Dies bedeutet, dass wenn $\sigma^2_{\delta s}$ und $\sigma^2_{\delta\phi}$ so gewählt werden, dass dann die Positionsunsicherheit gleich gross wird, wie wenn $\sigma^2_{\delta s1}$ und $\sigma^2_{\delta s2}$ gegeben wären.

Wird die odometrische Positionsbestimmung durch eine Orientierungsmessung, z.B. von einem Kreisel, unterstützt, so ist die Varianz $\sigma^2_{\delta\phi}$ des Orientierungsfehlers nicht mehr eine Funktion von δs und $\delta\phi$ und kann deshalb als von den Positionsfehlern unabhängig betrachtet werden. Dadurch können folgende Elemente von $\mathbf{C}(\Delta\mathbf{x}_{k+1})$ gesetzt werden:

$$c_{13}(\Delta\mathbf{x}_{k+1}) = c_{23}(\Delta\mathbf{x}_{k+1}) = c_{31}(\Delta\mathbf{x}_{k+1}) = c_{32}(\Delta\mathbf{x}_{k+1}) = 0$$

$$c_{33}(\Delta\mathbf{x}_{k+1}) = \sigma^2_{Kreisel}$$

Zur Veranschaulichung der Berechnung von $\mathbf{C}(\mathbf{x}_{k+1})$ ist in Fig. 5.5 das Resultat von drei simulierten Fahrten dargestellt. Die linke Spur



Figur 5.5: *Verlauf der Positionsunsicherheit bei verschiedenen Berechnungsarten*

gehört zur ersten Berechnungsart (Fehler ausgedrückt in $\delta s1$ und in $\delta s2$) und die mittlere Spur gehört zur zweiten Berechnungsart (Fehler ausgedrückt in δs und in $\delta\phi$). Bei dieser wurde eine leicht geringere Unsicherheit in ϕ angenommen als bei der linken Kurve. Bei der rechten Spur wurde die Positionsmessung von einem Kreisel unterstützt. Die Grösse der Covarianzmatrizen werden durch Fehlerellipsen dargestellt, welche das Gebiet umschreiben, in welchem die tatsächlichen Positionen mit einer Wahrscheinlichkeit von 50% liegen. Für α wurde ein Wert von 0.04 angenommen. Die Berechnung der Position und der Covarianzmatrix $C(\Delta x_{k+1})$ erfolgte in Abständen von ca.12cm, was bei einer Geschwindigkeit von 1m/sec einer Abtastzeit von 120msec entspricht.

Obwohl der Fehler in ϕ nicht dargestellt ist, kann man aus dem Verlauf der Positionsunsicherheit auf dessen Grösse schliessen. Augenfällig ist die Verbesserung durch die Benützung eines Kreisels, obwohl dieser mit einer Standardabweichung von $\sigma = 2°$ relativ ungenau ist.

5.4 Stützwerte für die Positionskorrektur

Die Roboterposition wird auf Grund des Unterschiedes zwischen erwarteten und gemessenen Abständen zu bekannten Objekten korrigiert. Da wir wegen der Einbettung in die Verhaltensbasierte Regelungsstruktur die Richtung und den Zeitpunkt der Messungen nicht steuern können, müssen wir jeden Messwert darauf hin untersuchen, ob er als Stützwert für die Positionsaufdatierung in Frage kommt. Wir müssen also jederzeit und unabhängig von der Position des Roboters und unabhängig von der Messrichtung entscheiden können, ob die betrachtete Abstandsmessung brauchbar ist oder nicht. Diese Aufgabe umfasst die Bestimmung der vom Roboterstandort aus sichtbaren Merkmale, sowie deren Richtung. Liegt ein sichtbares Objekt in Richtung der untersuchten Messung, so wird der Laufzeitfehler der Messung korrigiert und der erwartete Abstand mit dem gemessenen Abstand verglichen. Liegt der Unterschied innerhalb einer noch festzulegenden Toleranz, so wird angenommen, dass die Messung tatsächlich von diesem Objekt stammt, womit sie als Stützwert verwendet werden kann.

5.4.1 Bestimmung der sichtbaren Objekte

Die naheliegendste Methode wäre, aus dem *Referenzmodell* und dem momentanen Roboterstandort das Sichtbarkeitspolygon zu berechnen. *Kanten*, *Ecken* und *Wände*, die auf diesem Polygon lägen, kämen dann als sichtbare Objekte in Frage. Leider ist diese Methode für Echtzeitanwendungen zu langsam, da nach [Lee and Preparata 84] der Aufwand für die Bestimmung des Sichtbarkeitspolygons *mindestens* $\Omega(m \log n + F)$ beträgt (m: Anzahl Polygone der Umgebung, n: totale Anzahl der Umgebungspunkte, F: Grösse des resultierenden Sichtbarkeitspolygons).

Es gibt keine Methode, mit der die sichtbaren Punkte schneller bestimmt werden könnten. Um trotzdem in Echtzeit arbeiten zu können, müssen die Werte im voraus berechnet und abgespeichert werden. Ein Problem ist dabei die geeignete Darstellung und Verwendung dieser Vorausberechnungen. Wir schlagen hier eine Aufteilung der Umgebung in quadratische Felder vor (siehe Figur 5.6), weil dadurch die Zuweisung der Roboterposition zum entsprechenden Feld trivial wird. Zu jedem

Feld wird angegeben, welche *Kanten*, *Ecken* und *Wände* von innerhalb dieses Feldes sichtbar sind. Befindet sich der Roboter in irgend einem solchen Feld, so können die entsprechenden Umgebungsobjekte einfach aufgerufen werden. Im Betrieb können aber nicht alle diese Objekte auch tatsächlich verwendet werden, da erstens nicht von jeder Stelle innerhalb eines Feldes alle sichtbar sind, und zweitens weil Hindernisse die Sicht verdecken können. Die Auswahl der brauchbaren Merkmale geschieht erst bei der Auswertung der Sensordaten.

5.4.2 Messrichtung und Distanz zu sichtbaren Objekten

Aus der Position des Roboters und der Objekte wird die Richtung berechnet, in die eine Messung zu erfolgen hat. Dies soll an Hand von Figur 5.7 gezeigt werden.

Ausrichtung auf eine Wand

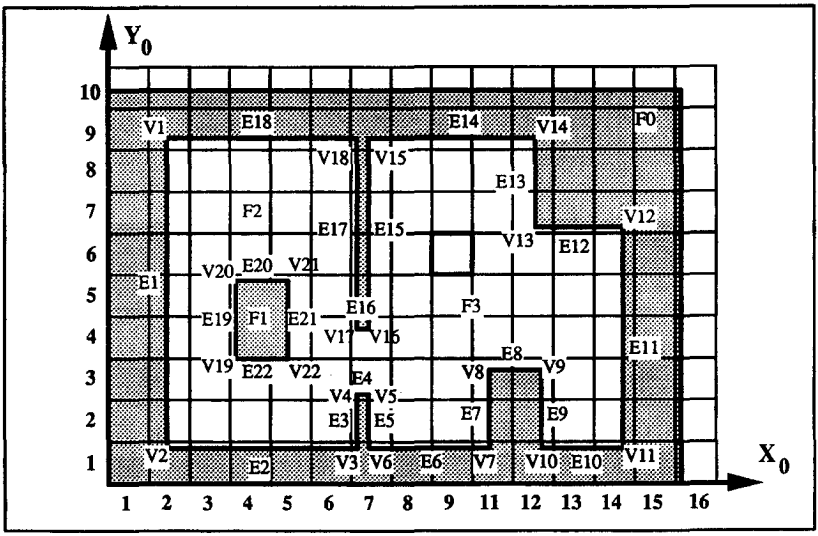
Die Ausrichtung eines Sensors auf eine Wand ist sehr einfach, da die Messrichtung beim Fahren konstant bleibt, falls der mobile Roboter seine Orientierung nicht ändert.

$$\gamma_{Wi} = \beta_i - \frac{\pi}{2} - \psi_R \tag{5.26}$$

Ausrichtung auf eine Kante oder Ecke

Etwas schwieriger ist das Ausrichten eines Sensors auf eine Kante oder Ecke. Jede Verschiebung des Roboters bedeutet eine Änderung der Messrichtung. Für die Berechnung werden die Koordinaten der Kante ${}^0\mathbf{r}_K$ ins Roboterkoordinatensystem transformiert. ${}^R\mathbf{r}_K$ zeigt dann in die gesuchte Richtung.

$${}^R\mathbf{r}_K = {}^0_R\mathbf{T}_H^{-1} * {}^0\mathbf{r}_K$$



Figur 5.6: *Darstellung sichtbarer Merkmale:*
Vom Feld 96 aus sind z.B. folgende Objekte sichtbar:

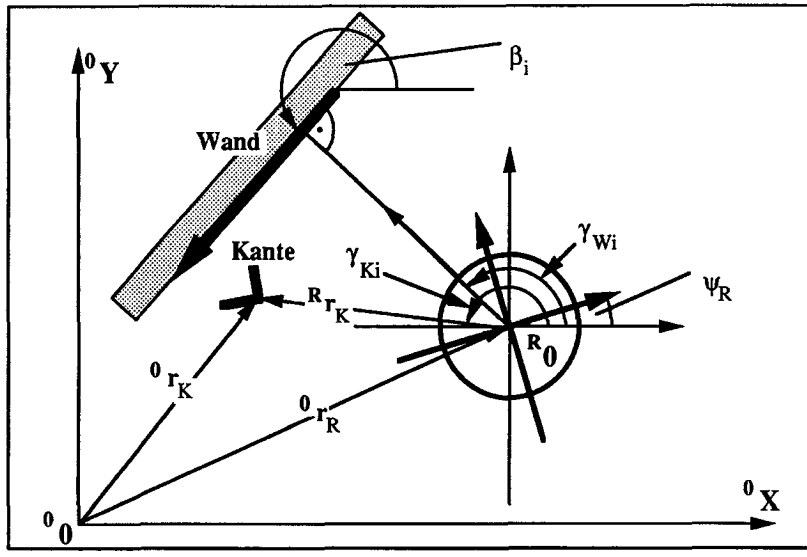
Feld 96:

Kanten: *V5 V8 V9 V13 V16*
Ecken: *V2 V6 V7 V11 V12 V14 V15*
Wände: *E1 E2 E4 E5 E6 E7 E8 E10 E11 E12 E13 E14 E15*

Diese Liste kann wesentlich reduziert werden, wenn man nur diejenigen Merkmale berücksicht, die mit einem bestimmten Sensorsystem überhaupt gesehen werden können. Für die Erkennung mit Ultraschall ergäbe sich:

Feld 96:

Kanten: *V5 V8 V13*
Ecken: *V6 V14 V15*
Wände: *E6 E11 E14 E15*



Figur 5.7: *Bestimmung der Messrichtung*

bzw.

$$\begin{bmatrix} {}^{R}x_K \\ {}^{R}y_K \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\psi_R & \sin\psi_R & -x_R * \cos\psi_R - y_R * \sin\psi_R \\ -\sin\psi_R & \cos\psi_R & x_R * \sin\psi_R - y_R * \cos\psi_R \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{0}x_K \\ {}^{0}y_K \\ 1 \end{bmatrix} \tag{5.27}$$

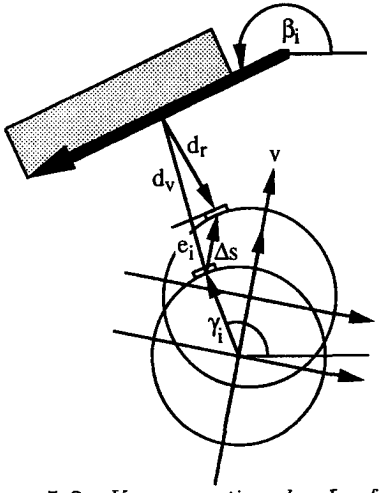
Der gesuchte Winkel γ_{Ki} ist dann

$$\gamma_{Ki} = \arctan \frac{{}^{R}y_K}{{}^{R}x_K} \tag{5.28}$$

Als Erwartungswert der Messung wird der Abstand des Roboters zum angepeilten Objekt genommen.

5.4.3 Korrektur des Laufzeitfehlers

Als Mass für den Abstand zu einem Objekt wird die Zeit verwendet, die ein ausgesendeter Strahl bis zu seiner Rückkehr benötigt. Dieser Wert stimmt nicht exakt, wenn sich der Sender in der Zwischenzeit selber verschoben hat. Die gemessene Distanz muss dann um dessen



Figur 5.8: *Kompensation der Laufzeitfehler*

Eigenbewegung kompensiert werden. Natürlich ist eine Kompensation nur sinnvoll, wenn die Geschwindigkeit des Strahls nicht sehr viel höher als diejenige des Senders ist. Bei den Geschwindigkeiten, die ein mobiler Roboter erreichen kann, muss die Laufzeit von Schallwellen, nicht aber diejenige von elektromagnetischen Wellen berücksichtigt werden.

Für die Berechnung der Kompensation des Laufzeitfehlers betrachten wir zuerst Figur 5.8 und die darin angeschriebenen Grössen. Der Roboter bewege sich mit einer Geschwindigkeit v vorwärts. Der während der Messung zurückgelegte Weg sei Δs. γ_i bezeichne die Messrichtung, d_v den Abstand zu Beginn der Messung, d_r den Abstand am Schluss der Messung und e_i sei der Unterschied zwischen dem Abstand am Anfang und dem Abstand am Schluss. Mit d_{US} wird der vom Sensor gemessene Abstandswert bezeichnet. Weitere Grössen sind die Schallgeschwindigkeit c_{US} und die Laufzeit t_c. Berechnen wollen wir d_v, den Abstand zu Beginn der Messung.

$$\Delta \mathbf{s} = \mathbf{v} * t_c \text{ , wobei } t_c = \frac{d_{US}}{c_{US}}$$

und

$$e_i = \gamma_i' \Delta \mathbf{s}$$

Mit

$$d_{US} = \frac{d_v + d_r}{2} \text{ und } d_r \simeq d_v - e_i$$

folgt für d_v:

$$d_v = d_{US} + \frac{e_i}{2} \tag{5.29}$$

Diese Berechnung ist richtig, falls sich der Roboter während der Messung mit gleicher Geschwindigkeit fortbewegt. Diese Annahme ist sicher korrekt, da die Messzeit bei einem Abstand von 10m nur ungefähr 60msec beträgt. Andernfalls müsste Δs odometrisch bestimmt werden, was aber auch nicht genauer wäre, weil dies auch nur in diskreten Zeitabständen möglich ist.
Ein anderer Effekt ist das sich Drehen des Roboters während der Messung. Dadurch wird auch der Sensor um den Robotermittelpunkt gedreht. Wegen der relativ grossen Sensoröffnung kann dieser Effekt vernachlässigt werden. Im schlimmsten Fall würde nicht etwas Falsches, sondern gar nichts gemessen.

5.4.4 Modellierung des Messfehlers

Aus einem Ultraschallbild lassen sich Abstand und Winkel zu einem Objekt relativ genau bestimmen. Aus einer einzigen Abstandsmessung kann aber nur bedingt auf die Richtung geschlossen werden, in der das Objekt liegt. Experimentell haben wir folgende Werte für $\sigma_{d_{US}}$ und für $\sigma_{\phi_{US}}$ ermittelt:

	$\sigma_{d_{US}}$	$\sigma_{\phi_{US}}$
Wand	0.08m	7.5deg
Kante	0.03m	7.5deg
Ecke	0.1m	7.5deg

Wir nehmen also an, dass die Messwerte normalverteilte Zufallsgrössen sind. Die relativ grossen Fehler ergeben sich dadurch, dass die Sensoren nicht genau auf die Objekte ausgerichtet werden können.

5.4.5 Bestimmung der Stützwerte

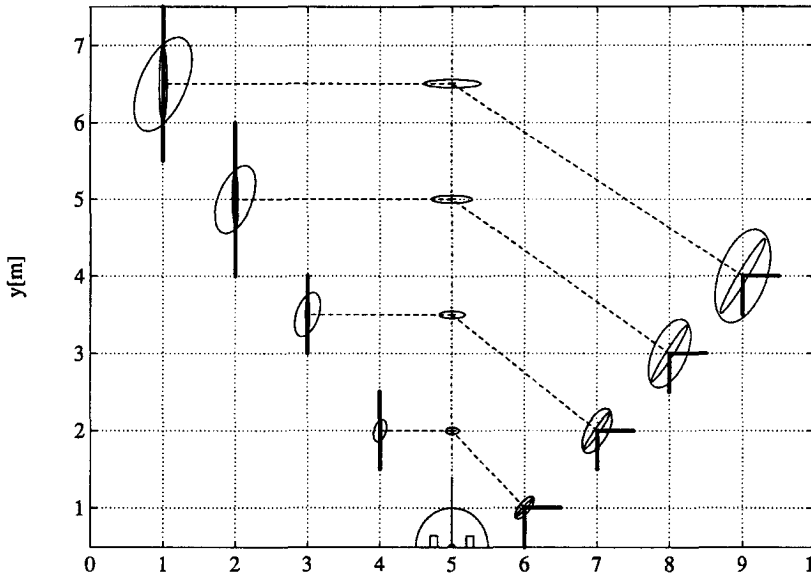
Als letztes muss untersucht werden, ob der erwartete Abstand mit dem Messwert übereinstimmt. Eine Abweichungen deutet darauf hin, dass die Messung nicht vom angepeilten Objekt, sondern von einem Hindernis stammt. Eine gute Positionsaufdatierung ist nur möglich, wenn falsche Werte eliminiert werden.

Theoretisch ist es nicht möglich, allein auf Grund des Abstandes zu entscheiden, ob das Echo des Ultraschallsignales von einem Hindernis stammt oder nicht. Wir können nur kontrollieren, ob die Differenz zwischen gemessenem und berechnetem Abstand innerhalb einer bestimmten Grenze liegt. Dabei müssen die Positionsunsicherheit des Roboters und die Ungenauigkeit der Messung berücksichtigt werden. Aus $\mathbf{C}(\mathbf{x}_k)$ und $\mathbf{C}(\mathbf{d}_{US})$ kann $\mathbf{C}(\mathbf{x}_k, \mathbf{d}_{US})$ berechnet werden. Um zu entscheiden, ob eine Messung verwendet werden kann oder nicht, muss untersucht werden, ob das angepeilte Objekt innerhalb der Fehlerellipse von $\mathbf{C}(\mathbf{x}_k, \mathbf{d}_{US})$ liegt oder nicht. Dazu testen wir, ob die Ungleichung

$$(\mathbf{x}_{Obj} - \mathbf{x}_{Obj_{US}})^T \mathbf{C}_2(\mathbf{x}_k, \mathbf{d}_{US})^{-1} (\mathbf{x}_{Obj} - \mathbf{x}_{Obj_{US}}) < k_{xd}^2 \qquad (5.30)$$

erfüllt ist. Wenn ja, liegt die vermutete Objektpositon $\mathbf{x}_{Obj_{US}}$ innerhalb der Fehlerellipse, deren Mittelpunkt die nominelle Objektposition $\mathbf{x}_{Obj}$ ist. Für die Objekte *Kante* und *Ecke* entspricht $\mathbf{x}_{Obj}$ deren Position, während wir beim Objekt *Wand* für $\mathbf{x}_{Obj}$ den Fusspunkt des von der Roboterposition aus gezogenen Lotes nehmen.

Fig 5.9 zeigt den Verlauf von $\mathbf{C}(\mathbf{x}_k, \mathbf{d}_{US})$ in Abhängigkeit der Messrichtung, der gemessenen Distanz und der odometrischen Positionsunsicherheit. Der Roboter startet bei der Position (5m/0.5m) und fährt mit konstanter Geschwindigkeit parallel zur y-Achse. Bei den Positionen (5.0m/2.0m), (5.0m/3.5m), (5.0m/5.0m) und (5.0m/6.5m) werden je eine Abstandsmessung zu einer Wand und zu einer Kante durchgeführt. Jede Messung ist durch eine gestrichelte Linie angedeutet. Durch Ellipsen werden die Odometriefehler, die Sensorfehler und die Messunsicherheit in Funktion des Odometrie- und des Sensorfehlers angedeutet.



Figur 5.9: *Verlauf von* $\mathbf{C}(\mathbf{x}_k, \mathbf{d}_{US})$ *in Abhängigkeit der Messrichtung, der gemessenen Distanz und der odometrischen Positionsunsicherheit.*

5.5 Positionskorrektur mittels Kalmanfilterung

Nach den Ausführungen in den vorhergehenden Kapiteln sind wir nun bereit, die Positionsaufdatierung mittels Kalmanfilterung durchzuführen. Nach einer kurzen Einführung in die entsprechende Theorie werden die Anpassungen besprochen, die durch die Trennung von Odometrie und Positionsaufdatierung notwendig sind.

5.5.1 Theorie

Das Kalmanfilter wird für die Schätzung von Zustandsgrössen verwendet, wobei diese Schätzung optimal ist im Sinne einer Minimierung des Erwartungswertes der Summe der Quadrate der Schätzfehler. Für ein lineares System basiert das Kalmanfilter auf folgenden Annahmen und Gleichungen ([Gelb 84]):

Modellgleichung:

$$\mathbf{x}_k = \Phi_{k-1}\mathbf{x}_{k-1} + \mathbf{w}_k, \quad \mathbf{w}_k \sim N(0, \mathbf{Q}_k) \tag{5.31}$$

Messgleichung:

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k \sim N(0, \mathbf{R}_k) \tag{5.32}$$

Anfangsbedingungen:

$$E[\mathbf{x}_0] = \hat{\mathbf{x}}_0, \; E[(\mathbf{x}_0 - \hat{\mathbf{x}}_0)(\mathbf{x}_0 - \hat{\mathbf{x}}_0)^T] = \mathbf{P}_0$$

Andere Annahmen:

$$E[\mathbf{w}_k\mathbf{v}_j^T] = 0 \text{ für alle j,k}$$

Extrapolation des Zustandsvektors:

$$\hat{\mathbf{x}}_{k/k-1} = \Phi_{k-1}\hat{\mathbf{x}}_{k-1/k-1} \tag{5.33}$$

Extrapolation der Covarianzmatrix:

$$\mathbf{P}_{k/k-1} = \Phi_{k-1}\mathbf{P}_{k-1/k-1}\Phi_{k-1}^T + \mathbf{Q}_{k-1} \tag{5.34}$$

Aufdatierung der Zustandsschätzung:

$$\hat{\mathbf{x}}_{k/k} = \hat{\mathbf{x}}_{k/k-1} + \mathbf{K}_k[\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k/k-1}] \tag{5.35}$$

Aufdatierung der Fehlercovarianzmatrix:

$$\mathbf{P}_{k/k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\mathbf{P}_{k/k-1} \tag{5.36}$$

Kalman-Verstärkungsmatrix:

$$\mathbf{K}_k = \mathbf{P}_{k/k-1}\mathbf{H}_k^T[\mathbf{H}_k\mathbf{P}_{k/k-1}\mathbf{H}_k^T + \mathbf{R}_k]^{-1} \tag{5.37}$$

Das eigentliche Kalmanfilter umfasst die Gleichungen 5.33 bis 5.37. Bei jedem Aufdatierungsschritt wird mit Hilfe der Modellgleichungen eine Schätzung des Zustandes $\hat{\mathbf{x}}_{k/k}$ und der neuen Covarianzmatrix $\mathbf{P}_{k/k-1}$ berechnet. Diese beiden Grössen werden anschliessend unter Berücksichtigung von Messungen $\mathbf{z}_k$ korrigiert.

Die Gleichungen für die Berechnung der Roboterposition und die hier verwendeten Messungen sind nichtlinear. In diesem Fall muss das

sogenannte *Extended Kalman Filter* verwendet werden:
Modellgleichung:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{w}(t), \quad \mathbf{w}(t) \sim N(0, \mathbf{Q}(t)) \tag{5.38}$$

Messgleichung:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}(t_k)) + \mathbf{v}_k, \quad k = 1, 2, ...; \quad \mathbf{v}_k \sim N(0, \mathbf{R}_k) \tag{5.39}$$

Anfangsbedingungen:

$$E[\mathbf{x}_0] \sim N(\hat{\mathbf{x}}_0, \mathbf{P}_0)$$

Andere Annahmen:

$$E[\mathbf{w}(t)\mathbf{v}_j^T] = 0 \text{ für alle j,k}$$

Extrapolation des Zustandsvektors:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), t) \tag{5.40}$$

Extrapolation der Covarianzmatrix:

$$\dot{\mathbf{P}}(t) = \mathbf{F}(\hat{\mathbf{x}}(t))\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}^T(\hat{\mathbf{x}}(t)) + \mathbf{Q}(t) \tag{5.41}$$

Aufdatierung der Zustandsschätzung:

$$\hat{\mathbf{x}}_{k/k} = \hat{\mathbf{x}}_{k/k-1} + \mathbf{K}_k[\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k/k-1})] \tag{5.42}$$

Aufdatierung der Fehlercovarianzmatrix:

$$\mathbf{P}_{k/k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1})]\mathbf{P}_{k/k-1} \tag{5.43}$$

Kalman-Verstärkungsmatrix:

$$\mathbf{K}_k = \mathbf{P}_{k/k-1}\mathbf{H}_k^T(\hat{\mathbf{x}}_{k/k-1})[\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1})\mathbf{P}_{k/k-1}\mathbf{H}_k^T(\hat{\mathbf{x}}_{k/k-1}) + \mathbf{R}_k]^{-1} \tag{5.44}$$

Definitionen:

$$\mathbf{F}(\hat{\mathbf{x}}(t)) = \frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \Big|_{\mathbf{x}(t)=\hat{\mathbf{x}}(t)}$$

$$\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1}) = \frac{\partial \mathbf{h}_k(\mathbf{x}(t_k))}{\partial \mathbf{x}(t_k)} \Big|_{\mathbf{x}(t_k)=\hat{\mathbf{x}}_{k/k-1}}$$

Die Gleichungen 5.40 bis 5.44 bilden das *Extended Kalman Filter* für nichtlineare Systeme mit diskreten Messungen. Ein Vergleich mit

dem konventionellen Kalmanfilter zeigt, dass die Verstärkungen $\mathbf{K}_k$ (Gl. 5.44) Zufallsgrössen sind, welche über die Matrizen $\mathbf{F}(\hat{\mathbf{x}}(t))$ und $\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1})$ von der Schätzung $\hat{\mathbf{x}}_{k/k}$ abhängig sind. Dies ergibt sich aus der Tatsache, dass $\mathbf{f}$ und $\mathbf{h}_k$ um den aktuellen Schätzwert von $\mathbf{x}(t)$ linearisiert werden. Daraus folgt, dass eine Vorausberechnung von $\mathbf{K}_k$ nicht möglich ist und deshalb in Echtzeit erfolgen muss. Zudem ist auch die Folge der Schätzfehlercovarianzmatrizen $\mathbf{P}_k$ vom Verlauf von $\hat{\mathbf{x}}(t)$ abhängig und damit zufällig. D.h. die Genauigkeit der Schätzung ist vom Verlauf der Trajektorie von $\mathbf{x}(t)$ abhängig.

5.5.2 Anpassung an die Problemstellung

Im Gegensatz zu den Gleichungen des *Extended Kalman Filters* müssen $\hat{\mathbf{x}}_{k/k-1}$ odometrisch und $\mathbf{P}_{k/k-1}$ mit Gleichung 5.18 bestimmt werden, weil wir aus praktischen Gründen kein dynamisches Modell des mobilen Roboters verwenden. Problemlos ist die Berechnung von $\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1})$, dessen Wert $[\cos\hat{\gamma}_{k/k-1}, \sin\hat{\gamma}_{k/k-1}, 0]$ beträgt, wobei $\gamma_{k/k-1}$ dem Winkel vom Roboter zum angepeilten Objekt entspricht. Als Aufdatierungsgleichungen erhalten wir so:

$$\hat{\mathbf{x}}_{k/k-1} = \text{odometrisch bestimmte Position} \quad (5.45)$$

$$\mathbf{P}_{k/k-1} = \mathbf{C}(\mathbf{x}_k) \quad (5.46)$$

$$\mathbf{R}_k = \sigma_{\Delta d} \quad (5.47)$$

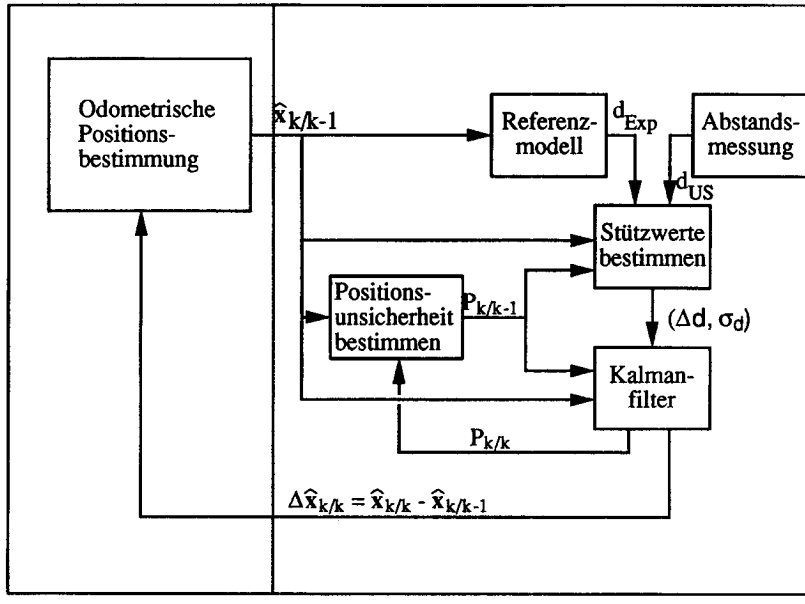
$$\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1}) = \left[\cos\hat{\gamma}_{k/k-1}, \sin\hat{\gamma}_{k/k-1}, 0\right] \quad (5.48)$$

$$\mathbf{K}_k = \mathbf{P}_{k/k-1}\mathbf{H}_k^T(\hat{\mathbf{x}}_{k/k-1}) \left[\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1})\mathbf{P}_{k/k-1}\mathbf{H}_k^T(\hat{\mathbf{x}}_{k/k-1}) + \mathbf{R}_k\right]^{-1} \quad (5.49)$$

$$\hat{\mathbf{x}}_{k/k} = \hat{\mathbf{x}}_{k/k-1} + \mathbf{K}_k\Delta d \quad (5.50)$$

$$\mathbf{P}_{k/k} = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1})]\mathbf{P}_{k/k-1} \quad (5.51)$$

Der Aufdatierungsvorgang soll durch Fig. 5.10 verdeutlicht werden. Sobald neue Abstandsmessungen vorliegen, wird untersucht, ob diese als Stützwerte verwendet werden können oder nicht. Dazu muss zuerst die neue Positionsunsicherheit $\mathbf{P}_{k/k-1}$ bestimmt werden. Ist ein neuer Stützwert gefunden, so wird eine neue Schätzung der Position und der Fehlercovarianzmatrix berechnet. Erfolgen, wie im Falle unseres Ultraschallsystemes, mehrere Abstandsmessungen gleichzeitig, so wird pro Stützwert das Kalmanfilter einmal durchlaufen und erst am Ende

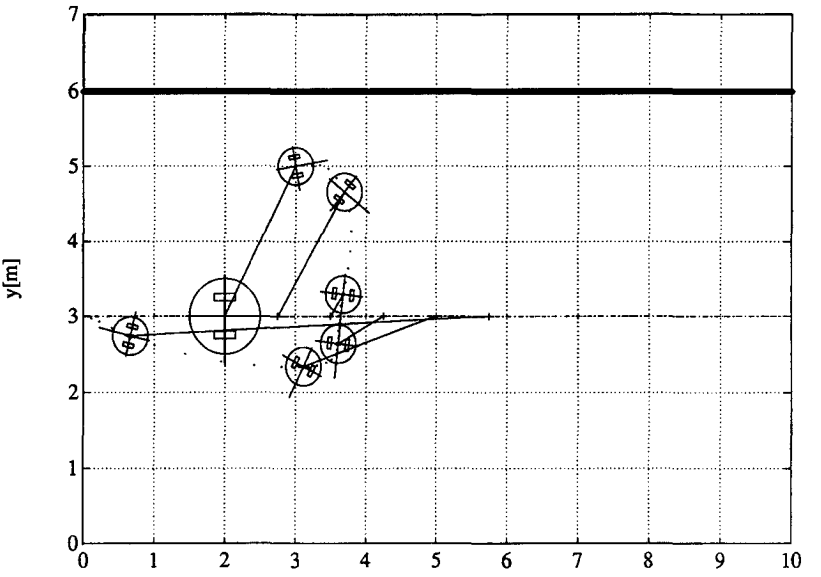


Figur 5.10: *Positionsaufdatierung mit Kalmanfilter*

die Positionskorrektur an die odometrische Positionsbestimmung weitergeleitet.

5.5.3 Gleichzeitige Positions- und Orientierungskorrektur

In der oben angegebenen Formulierung werden mit einer einzigen Abstandsmessung die drei Grössen x, y und ϕ korrigiert. Wie Figur 5.11 zeigt, führt dies zu unbrauchbaren Ergebnissen. Der Grund dafür ist, dass das Kalmanfilter versucht, die Abstandsdifferenz Δd nicht nur durch Verschiebung des Roboters, sondern auch durch Drehung zu reduzieren. Dies führt aber schon bei kleinen Abstandsdifferenzen zu grossen Korrekturen in ϕ.



Figur 5.11: *Gleichzeitige Positions- und Orientierungskorrektur: Der mobile Roboter steht an der Stelle [2.0, 3.0]. Die odometrisch bestimmte Position sei aber [3.0, 5.0], was einem relativ grossen Fehler entspricht. Dieser Fehler soll durch Abstandsmessungen senkrecht zur Wand bei y=6.0 abgebaut werden. Der Roboter bewegt sich dabei mit konstanter Geschwindigkeit auf der Geraden y=3.0 in x-Richtung.*

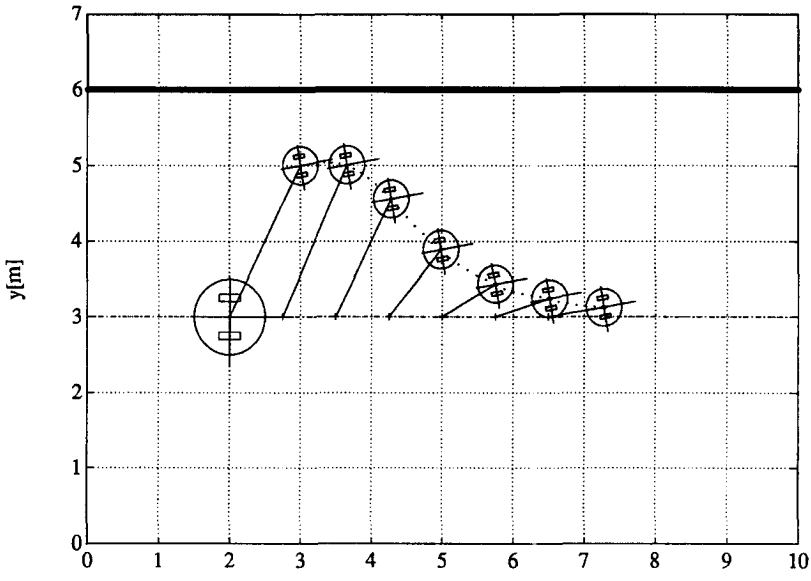
5.5.4 Entkopplung von Positions- und Orientierungskorrektur

Abstandsmessungen sind für eine Korrektur der Orientierung ungeeignet, weshalb die Kalmanfiltergleichungen nur für x und y einzusetzen sind. Dies setzt aber voraus, dass die Orientierung entweder mit einem Kreisel stabilisiert wird, oder dass in geeigneten Zeitabständen eine Korrektur des Orientierungsfehlers erfolgt, die sich auf eine andere Methode abstützt.

Für die Positionskorrektur ohne Korrektur des Orientierungsfehlers müssen folgende Anpassungen gemacht werden:

$$\mathbf{P}_{k/k-1} = \mathbf{C}_2(\mathbf{x}_k) \text{ und } \mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1}) = \left[\cos\hat{\gamma}_{k/k-1}, \sin\hat{\gamma}_{k/k-1}\right] \quad (5.52)$$

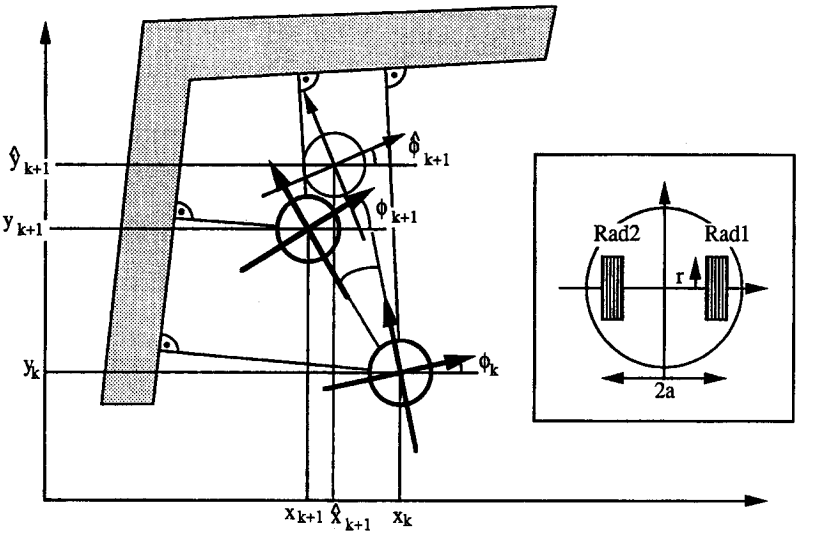
Bei den anderen Gleichungen ändern sich die Dimensionen der Ma-



Figur 5.12: *Entkoppelte Positionskorrektur: Der mobile Roboter steht an der Stelle [2.0, 3.0]. Die odometrisch bestimmte Position sei aber [3.0, 5.0]. Nebst diesem grossen Positionsfehler liege ein Orientierungsfehler von 18° vor. Der Positionsfehler soll durch Abstandsmessungen senkrecht zur Wand bei y=6.0 abgebaut werden. Der Roboter bewegt sich dabei mit konstanter Geschwindigkeit auf der Geraden y=3.0 in x-Richtung.*

trizen, nicht aber die Ausdrücke. Offen bleibt noch die Frage, wie nach der Filterung $\mathbf{C}(\mathbf{x}_k)$ aus $\mathbf{C}_2(\mathbf{x}_k)$ und der Orientierungsunsicherheit wieder zusammengesetzt werden soll. Verfügt der mobile Roboter über einen Kreisel, so sind die Messungen in (x, y) und in (ϕ) entkoppelt und die Bildung von $\mathbf{C}_2(\mathbf{x}_k)$ aus $\mathbf{C}(\mathbf{x}_k)$ ist offensichtlich. Fehlen Richtungsmessungen, so nehmen wir an, dass Positions- und Orientierungsfehler durch die Filterung entkoppelt werden. Bei der odometrischen Positionsschätzung entsteht aber sofort wieder eine Kopplung dieser beiden Grössen.

Die erzielte Verbesserung soll mit Fig. 5.12 gezeigt werden. Obwohl ein nicht auskorrigierter Orientierungsfehler vorliegt, ist die Korrektur in x und y wesentlich besser als in Fig. 5.11.



Figur 5.13: *Positionsfehler auf Grund des Orientierungsfehlers*

5.6 Korrektur des Orientierungsfehlers

Im vorherigen Kapitel haben wir gesehen, dass das Kalmanfilter nicht in der Lage ist, aus Abstandsmessungen eine befriedigende Korrektur des Orientierungsfehlers zu ermitteln. Dies hängt unter anderem damit zusammen, dass die Positionsbestimmung nichtlinear ist in ϕ. Es gibt aber eine andere Möglichkeit, die Schätzung der Orientierung zu verbessern. Man geht dabei von der Beobachtung aus, dass bei einem Orientierungsfehler die Position des fahrenden Roboters immer in die gleiche Richtung korrigiert werden muss. Natürlich kann es sich dabei auch um einen systematischen Fehler bei der Odometrie handeln. Wir wollen aber davon ausgehen, dass der Effekt solcher Fehler im Vergleich zu den Auswirkungen von Orientierungsfehlern gering sei. Als weitere Einschränkung nehmen wir an, dass der Roboter keinen Schlupf in Richtung der Radachsen habe.

Für die Herleitung des Verfahrens wollen wir uns zuerst an die Positionsbestimmung mittels Koppelnavigation zurückerinnern. Die Gleichungen für ein Wegstück zwischen zwei Abtastpunkten lauten (siehe

auch Figur 5.13):

$$\delta\hat{s} = \frac{\delta\hat{s}_1 + \delta\hat{s}_2}{2} \; ; \; \delta\hat{\phi} = \frac{\delta\hat{s}_1 - \delta\hat{s}_2}{2a} \tag{5.53}$$

Daraus wird die neue Position des Roboters berechnet:

$$\hat{x}_{k+1} = x_k - \delta\hat{s} * \sin(\phi_k + \frac{\delta\hat{\phi}}{2}) \tag{5.54}$$

$$\hat{y}_{k+1} = y_k + \delta\hat{s} * \cos(\phi_k + \frac{\delta\hat{\phi}}{2}) \tag{5.55}$$

$$\hat{\phi}_{k+1} = \phi_k + \delta\hat{\phi} \tag{5.56}$$

$\delta\hat{s}_1$ und $\delta\hat{s}_2$ bezeichnen die Wegstücke der beiden Räder seit der letzten Aufdatierung. Diese entsprechen aber nicht genau dem gefahrenen Weg, da der Schlupf der beiden Räder noch nicht berücksichtigt ist. Um die tatsächlich zurückgelegten Wegstücke zu erhalten, subtrahieren wir deshalb bei jedem Rad einen Betrag e_i:

$$\delta s_1 = \delta\hat{s}_1 - e_1 \text{ und } \delta s_2 = \delta\hat{s}_2 - e_2$$

Für die tatsächlichen Werte δs und $\delta\phi$ erhalten wir somit:

$$\delta s = \frac{\delta\hat{s}_1 - e_1 + \delta\hat{s}_2 - e_2}{2} \tag{5.57}$$

$$\delta\phi = \frac{(\delta\hat{s}_1 - e_1) - (\delta\hat{s}_2 - e_2)}{2a} \tag{5.58}$$

Uns interessiert vor allem die tatsächliche Winkeländerung $\delta\phi$. Diese kann so umgeschrieben werden:

$$\delta\phi = \frac{\delta\hat{s}_1 - \delta\hat{s}_2}{2a} - \frac{e_1 - e_2}{2a} \tag{5.59}$$

Um die tatsächliche Winkeländerung $\delta\phi$ bestimmen zu können, brauchen wir also nebst den odometrisch gemessenen Radumdrehungen nur die Differenz $e_1 - e_2$ des Schlupfes der beiden Räder zu kennen. Diese kann man berechnen, da man sowohl die odometrisch bestimmte als auch die aufdatierte Position des Roboters kennt:
Odometrisch bestimmte Position:

$$\hat{x}_{k+1} = x_k - \frac{\delta\hat{s}_1 + \delta\hat{s}_2}{2} * \sin(\phi_k + \frac{\delta\hat{s}_1 - \delta\hat{s}_2}{2 * 2a}) \tag{5.60}$$

$$\hat{y}_{k+1} = y_k + \frac{\delta\hat{s}_1 + \delta\hat{s}_2}{2} * \cos(\phi_k + \frac{\delta\hat{s}_1 - \delta\hat{s}_2}{2 * 2a}) \tag{5.61}$$

Tatsächliche Position:

$$x_{k+1} = x_k - \frac{\delta\hat{s}_1 - e_1 + \delta\hat{s}_2 - e_2}{2} * \sin(\phi_k + \frac{\delta\hat{s}_1 - e_1 - \delta\hat{s}_2 + e_2}{2 * 2a})$$

$$y_{k+1} = y_k + \frac{\delta\hat{s}_1 - e_1 + \delta\hat{s}_2 - e_2}{2} * \cos(\phi_k + \frac{\delta\hat{s}_1 - e_1 - \delta\hat{s}_2 + e_2}{2 * 2a})$$

Diese Gleichungen können dividiert werden:

$$-\frac{\hat{x}_{k+1} - x_k}{\hat{y}_{k+1} - y_k} = \tan\left(\phi_k + \frac{\delta\hat{s}_1 - \delta\hat{s}_2}{2 * 2a}\right)$$

$$-\frac{x_{k+1} - x_k}{y_{k+1} - y_k} = \tan\left(\phi_k + \frac{\delta\hat{s}_1 - \delta\hat{s}_2 - (e_1 - e_2)}{2 * 2a}\right)$$

Nach weiteren Umformungen erhält man:

$$\begin{aligned} e_1 - e_2 = \; & 4a * \{\arctan[-(\hat{x}_{k+1} - x_k), (\hat{y}_{k+1} - y_k)] \\ & - \arctan[-(x_{k+1} - x_k), (y_{k+1} - y_k)]\} \end{aligned}$$

Diese Gleichung muss schliesslich noch in Gleichung 5.59 eingesetzt werden, woraus man die Winkelkorrektur erhält:

$$\begin{aligned} \delta\phi - \delta\hat{\phi} = \; & -\{\arctan[-(\hat{x}_{k+1} - x_k), (\hat{y}_{k+1} - y_k)] \\ & - \arctan[-(x_{k+1} - x_k), (y_{k+1} - y_k)]\} \end{aligned} \tag{5.62}$$

Der Vorteil dieses Verfahrens ist, dass für die Berechnung von $\delta\phi$ keine früher berechneten Winkelwerte verwendet werden. Damit ist garantiert, dass die Orientierungsfehler nicht aufintegriert werden. Zu erwähnen ist auch, dass sich Ultraschallmessungen und die hier beschriebene Methode ideal ergänzen, weil diese richtige Abstandswerte liefern, solange der Orientierungsfehler nicht grösser als die Messkeule ist.

Die obige Berechnung gilt für stückweise lineare Wegstücke. Diese Bedingung ist aber nur bei Geradeausfahrt erfüllt. Bei Kurvenfahrt könnte eine Approximation mit Geradenstücken vorgenommen werden. Eine Korrektur des Orientierungsfehlers sollte aber nur in grösseren Abständen erfolgen, weil sonst jede kleinste Positionskorrektur zu einer grossen Orientierungskorrektur führen würde. Dann kann aber die Kurvenfahrt nicht mehr linearisiert werden. Wir lösen dieses Problem, indem wir die Fahrtkurve im roboterfesten Koordinatensystem eintragen und Buch führen über die Korrekturen in Fahrtrichtung (δs) und

in Querrichtung (δds). Die Orientierung wird dann jeweils in festen Abständen S korrigiert.

Die Formel für die Winkelkorrektur vereinfacht sich dadurch:

$$\delta\phi - \delta\hat{\phi} = -\{\arctan[0, S] - \arctan[\sum \delta ds, S + \sum \delta s]\} \qquad (5.63)$$

Die Grösse der Orientierungsunsicherheit ist von der gefahrenen Trajektorie abhängig und kann deshalb nicht im voraus berechnet werden. Wir begnügen uns deshalb mit einer groben Abschätzung.

$$\sigma^2_{\delta\phi} \approx \frac{\sigma^2_{\delta ds_0} + \sigma^2_{\delta ds_1}}{S^2} \qquad (5.64)$$

Nehmen wir für die Positionsunsicherheit $\sigma_{\delta ds_0}$ am Anfang und für die Unsicherheit $\sigma_{\delta ds_1}$ am Ende des Wegstückes je einen Wert von 0.05m und setzen wir $S = 1.5m$, so erhalten wir als Orientierungsunsicherheit

$$\sigma_{\delta\phi} \approx \sqrt{\frac{0.05^2 + 0.05^2}{1.5^2}} = 0.047 = 2.7°$$

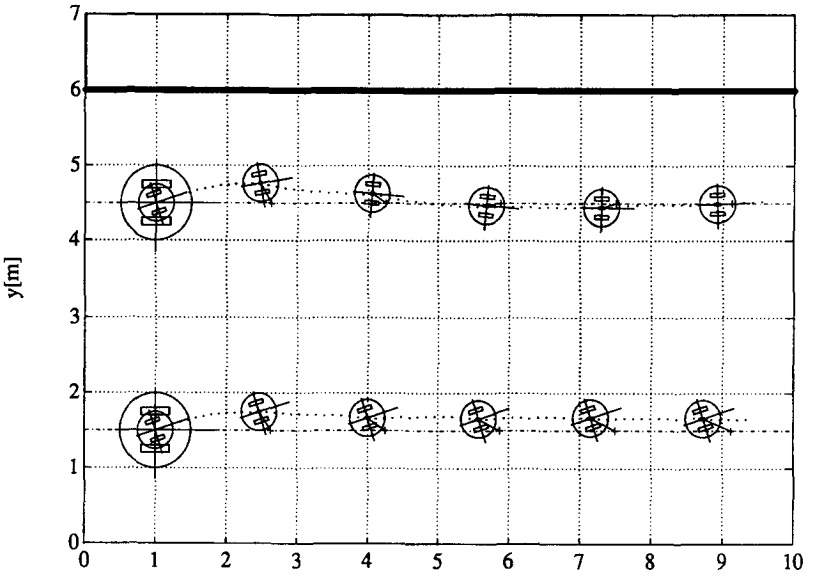
Anhand von Fig. 5.14 können eine Fahrt *ohne* und eine Fahrt *mit* Orientierungskorrektur miteinander verglichen werden. Die korrigierte Fahrt ist oben dargestellt und die Korrektur erfolgte nach jeweils 1.5m.

5.7 Beispiele

5.7.1 Einstellung der Parameter

Folgende Parameter müssen richtig gewählt werden, damit die Positionsaufdatierung erfolgreich durchgeführt werden kann:

- α - bestimmt die Genauigkeit der odometrischen Positionsschätzung
- C_0 - ist ein Mass für die Genauigkeit der Anfangsposition
- σ_{US} - steht für die Standardabweichung der Abstandsmessung



Figur 5.14: *Korrektur der Orientierung basierend auf Abstandsmessungen senkrecht zur Wand bei y=6.0m.*

- $\sigma_{Kreisel}$ - Standardabweichung der Orientierungsmessung
- k_{xd} - bestimmt das Vertrauensintervall, innerhalb welchem ein gemessenes Objekt als Referenzobjekt vermutet wird.
- S - Weg bis zur nächsten Orientierungskorrektur (bei fehlender Winkelmessung)
- Δs - Aufdatierungsintervall

Diese Parameter variieren je nach Qualität und Ausstattung des Roboters, nach der Fahrweise und nicht zuletzt auch nach der Umgebung. C_0, σ_{US} und $\sigma_{Kreisel}$ sind im wesentlichen von der Sensorgenauigkeit abhängig und können ziemlich genau vorausbestimmt werden. Schwieriger ist die Abschätzung von α. Weil wir kein dynamisches Modell verwenden, müssen wir annehmen, dass die Positionsunsicherheit unabhängig ist vom Fahrstil des Roboters. Dies zwingt uns, für α einen geeigneten Mittelwert zu wählen. Indirekt abhängig von α und von σ_{US} ist k_{xd}. Dieser Parameter ist so zu wählen, dass möglichst wenige falsche Messungen für die Aufdatierung verwendet werden. Wird aber k_{xd} zu

klein gewählt, so werden auch gute Messungen verworfen. S ist ebenfalls von α abhängig und ist so zu wählen, dass die Orientierungskorrektur möglichst unabhängig wird von den Sensorungenauigkeiten.

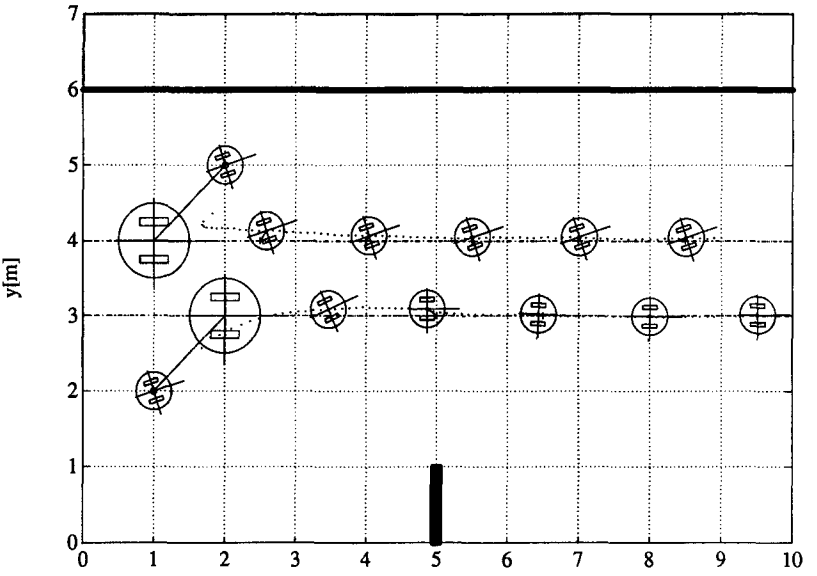
Eine Sonderstellung nimmt der Parameter Δs ein, weil er nicht vom Roboter, sondern von der verfügbaren Rechnerleistung und von der Geschwindigkeit der Sensoren abhängig ist. Eine hohe Rechenleistung garantiert aber noch keine gute Aufdatierung. Ebenso wichtig sind die Qualität und die Anzahl vorhandener Messdaten. Dies wiederum ist von der Umgebung abhängig.

Bei den folgenden zwei Beispielen gelten die unten angegebenen Parametereinstellungen:

$$
\begin{aligned}
\alpha &= 0.06 \text{ für Bsp.1 und} \\
&\quad\ 0.10 \text{ für Bsp.2} \\
C_0 &= \begin{bmatrix} (0.03m)^2 & 0 & 0 \\ 0 & (0.03m)^2 & 0 \\ 0 & 0 & (\frac{2}{57.3}rad)^2 \end{bmatrix} \\
\sigma_{US} &= 0.03m \\
\sigma_{Kreisel} &= (\frac{2}{57.3}rad) \\
k_{xd}^2 &= 4.6 \text{ (entspricht 90\%)} \\
S &= 1.5m \\
\Delta s &= 0.10 \text{ für Bsp.1 und} \\
&\quad\ 0.20 \text{ für Bsp.2}
\end{aligned}
$$

5.7.2 Beispiel 1

Hier soll gezeigt werden, dass sich die Objekte *Wand* und *Kante* für die Positionsaufdatierung ideal ergänzen. Fig. 5.15 zeigt zwei Fahrten. Oben startet der Roboter beim Punkt [1.0m,4.0m] und fährt mit konstanter Geschwindigkeit der Geraden [y=4m] entlang. Die geschätzte Anfangsposition beträgt [2.0m,5.0m] und die geschätzte Orientierung ist 10°, d.h. der Schätzfehler ist sehr gross. Es zeigt sich, dass die Positionsschätzung schon nach wenigen Aufdatierungsschritten wesentlich verbessert wird. Eine Orientierungskorrektur wird nicht durchgeführt.

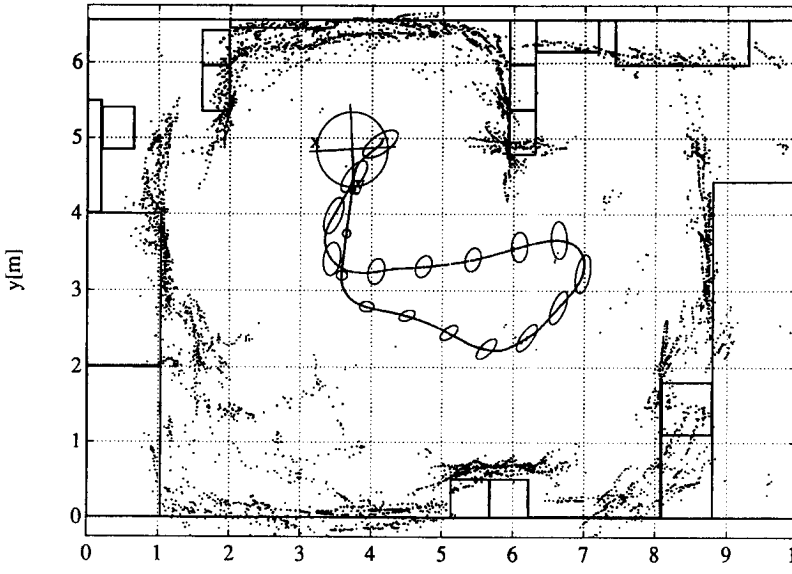


Figur 5.15: *Positionsaufdatierung mit Abstandsmessungen zu zwei verschiedenen Objekten (Wand: $x = 6m$, Kante: x=5m/y=1m). Parametereinstellungen:* $\alpha = 0.06$, $\Delta s = 0.1m$

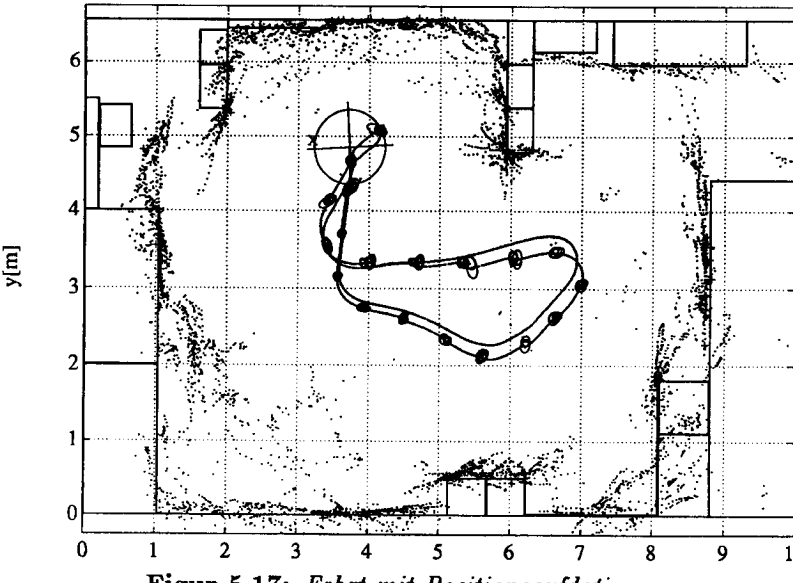
Eine ähnliche Situation haben wir unten in Fig. 5.15. Der Positionsfehler und der Orientierungsfehler beim Start sind betragsmässig wieder gleich gross. Diesmal startet der Roboter aber beim Punkt [2.0m,3.0m] und fährt der Geraden [y=3m] entlang. Im Gegensatz zu oben wird hier eine Orientierungskorrektur durchgeführt, welche die Genauigkeit der Positions- und Orientierungsschätzung wesentlich verbessert.

5.7.3 Beispiel 2

Anhand der Figuren 5.16 und 5.17 soll die Leistungsfähigkeit der Positionsaufdatierung in einer realen strukturierten Umgebung gezeigt werden. In Fig. 5.16 ist eine Fahrt ohne Aufdatierung abgebildet. Eingezeichnet sind der mobile Roboter beim Start (3.7m/4.8m), die Fahrtrajektorie, die Fehlerellipsen der odometrischen Positionsschätzung, die Umrisse des Referenzmodelles und schliesslich die Ultraschallmessungen (durch Punkte angedeutet). Wäre die Positionsschätzung gut, so müssten Abstandsmessungen zu einer *Wand* auf einer Geraden liegen.



Figur 5.16: *Fahrt ohne Positionsaufdatierung*



Figur 5.17: *Fahrt mit Positionsaufdatierung*

Abstandsmessungen zu *Kanten* müssten auf Linien liegen, die sich in einem Punkt schneiden. Dies ergibt sich aus den früher besprochenen Eigenschaften von Ultraschallmessungen. Dass der Roboter etwas von seinem Kurs abgekommen ist, sieht man am besten anhand der Messungen zu den Wänden bei y=0m, bei y=0.5m, bei y=4.8m und bei y=6.5m. Dort gibt es grosse Differenzen zwischen den Messungen und den tatsächlichen Positionen der Wände. Auffallend ist auch die grosse Anzahl von Fehlmessungen. Diese kommen z.T. davon, dass sich nicht abgebildete Personen im Raum befanden, oder auch davon, dass der Boden gewisse rauhe Stellen hat, welche die Ultraschallwellen reflektieren können. Dies ist vor allem bei grösseren Abständen ein Problem. Fig. 5.17 unterscheidet sich von Fig. 5.16 dadurch, dass nun die Position laufend aufdatiert wird. Die Trajektorie *ohne* Fehlerellipsen zeigt den Verlauf der unkorrigierten Fahrt. Die Trajektorie *mit* den Fehlerellipsen zeigt die korrigierte Positionsschätzung. Anhand der Abstandsmessungen an den oben erwähnten Stellen sieht man deutlich die Verbesserung der Positionsschätzung.

Eine Abschätzung der Positionsgenauigkeit ist analytisch schwierig und erfolgt deshalb nur optisch. Im schlimmsten Fall ist mit einem Fehler von ca. 5cm zu rechnen. Dieser kann bei langsamer Fahrt oder wenn viele gute Messungen vorhanden sind, wesentlich reduziert werden.

Leer - Vide - Empty

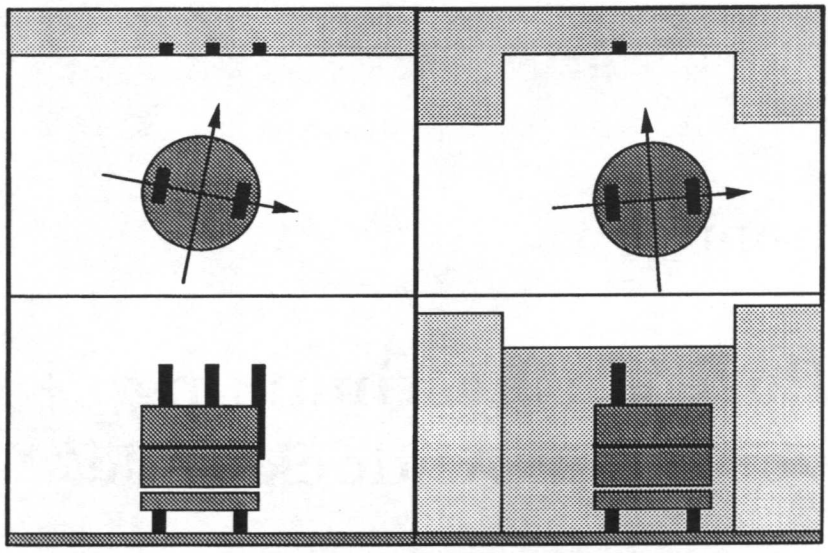
Kapitel 6

Positionsbestimmung vor einer Andockstelle

Sobald der mobile Roboter gezielt mit der Umgebung in Interaktion treten muss, ist eine höhere Positionsgenauigkeit gefordert, die mit den bis jetzt vorgestellten Methoden nur in günstigen Fällen erreicht werden kann. Um dieses Problem zu lösen, definieren wir Orte, wo der Roboter in gezielten Kontakt mit der Umgebung treten kann. Diese sogenannten *Andockstellen* können den Bedürfnissen des Roboters angepasst werden. Insbesondere soll es erlaubt sein, Markierungen zu installieren, um die Positionsbestimmung zu erleichtern.

6.1 Gestaltung der Andockstelle

Die Andockstelle muss so gestaltet sein, dass die Bestimmung der relativen Roboterposition unabhängig von der weiteren Umgebung erfolgen kann. Nehmen wir an, dass die Andockstelle passiv ist und damit die Messungen vom Roboter aus zu erfolgen haben, so ergeben sich, bei Anwendung von Abstands- und/oder Winkelmessungen, die in Kapitel 2.1 vorgestellten prinzipiellen Möglichkeiten. Diese werden aber dadurch eingeschränkt, dass der verfügbare Platz knapp ist. Entscheidend bleibt, dass die Andockstelle den Sensorsystemen des Roboters angepasst ist.



Figur 6.1: *Mögliche Gestaltungen einer Andockstelle: Links für eine Abstands- und drei Winkelmessungen, rechts für drei Abstands- und eine Winkelmessung*

Die beiden in Fig. 6.1 dargestellen Andockstellen sind ausgelegt für den kombinierten Einsatz von Ultraschallabstandssensoren zusammen mit einem Bildverarbeitungssystem, mit welchem Winkelmessungen gemacht werden können. Im linken Bild wird die Position mittels Triangulation berechnet. Mit Abstandsmessungen zur Wand kann deren Genauigkeit verbessert werden. Im rechten Bild wird die Position auf Grund von einer Winkel- und drei Abstandsmessungen bestimmt. Eine Abstandsmessung erfolgt in Richtung Wand, die beiden anderen in Richtung der beiden vorgeschobenen Ecken. Beide Lösungen haben Vor- und Nachteile. Die linke Andockstelle ist sehr einfach zu bauen und die Positionsbestimmung ist von allen Seiten möglich. Ungünstig ist, dass schon kleine Messfehler zu grossen Positionsfehlern führen und dass das Andockmuster dem Öffnungswinkel der Kamera angepasst werden muss. Bei der Variante rechts kann einfacher eine genügende Genauigkeit erreicht werden und der Sichtbereich der Kamera ist kein Problem. Unbefriedigend ist, dass die Positionsbestimmung aus seitlichen Lagen nicht möglich ist. Der Hauptnachteil ist aber, dass diese Andockstelle viel

mehr Platz braucht und ihr Aufbau relativ aufwendig ist. Aus diesen Gründen haben wir uns für die Variante links entschieden.

6.2 Bestimmung der Position relativ zur Andockstelle

An der Andockstelle sind drei Streifen in einem Abstand s angebracht (Fig. 6.2). Vom Roboter aus werden die drei Winkel β, δ und ϵ gemessen. Die Berechnung der Roboterposition (x, y, ϕ) erfolgt dann unter Verwendung der folgenden trigonometrischen Gleichungen:

$$\pi = \epsilon + \delta + \xi + \zeta \tag{6.1}$$

$$\frac{\sin\epsilon}{s} = \frac{\sin\zeta}{a} \quad \text{(Sinussatz)} \tag{6.2}$$

$$\frac{\sin\delta}{s} = \frac{\sin\xi}{a} \tag{6.3}$$

Daraus können ξ und a berechnet werden:

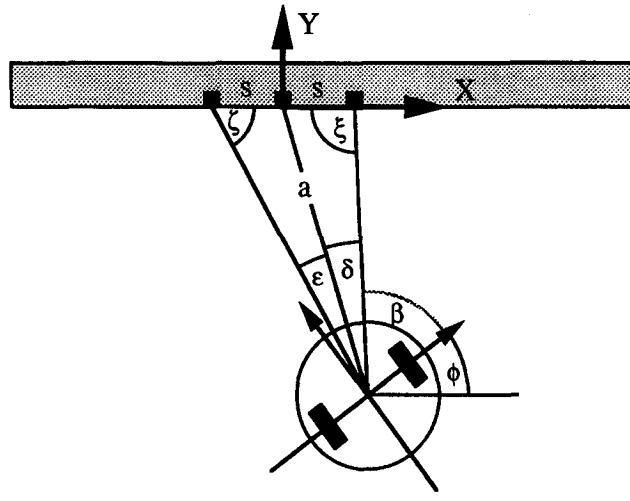
$$\begin{aligned} \frac{\sin\epsilon}{\sin\zeta} &= \frac{\sin\delta}{\sin\xi} \\ \xi &= \arctan\left(\frac{\sin(\epsilon+\delta)}{\frac{\sin\epsilon}{\sin\delta} - cos(\epsilon+\delta)}\right) \\ a &= s * \frac{\sin\xi}{\sin\delta} \end{aligned} \tag{6.4}$$

Als Roboterposition ergibt sich somit:

$$x = -a * cos(\delta + \xi) \tag{6.5}$$

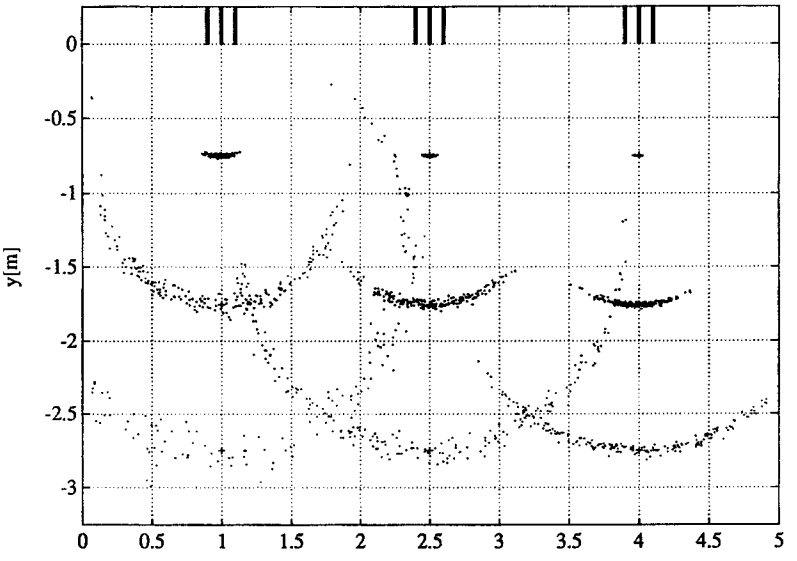
$$y = -a * sin(\delta + \xi) \tag{6.6}$$

$$\phi = \delta + \xi - \beta \tag{6.7}$$



Figur 6.2: *Positionsbestimmung vor der Andockstelle*

Die Schwierigkeit dieses Verfahrens liegt darin, dass schon kleine Fehler in δ und in ϵ zu grossen Fehlern bei der Positionsbestimmung führen. Dies kann direkt aus Gleichung 6.4 gefolgert werden. Wie gross diese Fehlerempfindlichkeit ist, kann mit Hilfe von Fig. 6.3 abgeschätzt werden. Dort wird von neun verschiedenen Standorten aus die Roboterposition jeweils mehrmals bestimmt. Die Messungen der beiden Zwischenwinkel δ und ϵ werden dabei als normalverteilte Zufallsvariabeln mit einer Standardabweichung σ_δ bzw. σ_ϵ betrachtet. Im Fall links ist $\sigma_\delta = \sigma_\epsilon = 0.1°$, im mittleren Fall sind beide $0.05°$ und im Fall rechts $0.025°$. Diese Standardabweichungen entsprechen damit grössenordnungsmässig der Genauigkeit einer Kamera mit einem Öffnungswinkel von 20° und einer Auflösung von 500 Pixeln pro Zeile. Der Abstand s der Streifen beträgt 10cm. Pro Standort wurde 200 Mal mit verrauschten δ und ϵ die Position bestimmt. Der Standort ist mit einem Kreuz bezeichnet.
Augenfällig ist, dass die Genauigkeit schnell abnimmt, wenn der Abstand zur Andockstelle wächst. Dies erklärt sich damit, dass bei zunehmendem Abstand δ und ϵ kleiner und dadurch die relativen Winkelfehler grösser werden. Das gleiche Problem ergibt sich bei Messungen aus seitlichen Lagen. Es ist also wichtig, dass der Roboter beim Heranfahren an die Andockstelle immer möglichst senkrecht zu dieser



Figur 6.3: *Positionsbestimmung vor drei Andockstellen basierend auf unsicheren Winkelmessungen. Die tatsächlichen Positionen sind durch ein Kreuz bezeichnet. Die Winkelunsicherheiten werden als normalverteilt angenommen und variieren von $\sigma_\delta = 0.1°$ (linke Seite) über $\sigma_\delta = 0.05°$ (Mitte) bis $\sigma_\delta = 0.025°$ (rechte Seite). Die Punkte bezeichnen die Verteilung der berechneten Positionen.*

steht.

Eine gute Positionsbestimmung ist also nur in der Nähe der Andockstelle möglich. Dies deckt sich mit unserer Forderung, dass die Genauigkeit vor allem bei der Andockstelle gross sein soll. Es erschwert aber ein befriedigendes Verhalten beim Heranfahren an die Andockstelle. Dies ist nur gewährleistet, wenn auch in Abständen von 2 bis 3 Metern die Positionsangabe nicht völlig daneben liegt.

Die berechneten Roboterpositionen liegen alle auf einem Ortbogen um die Andockstelle. Es wäre also möglich, die Genauigkeit dadurch zu verbessern, dass man nur Roboterpositionen zulässt, deren Abstand mit dem mittels Ultraschall gemessenen Abstand ungefähr übereinstimmt. Aus Fig. 6.3 ist aber ersichtlich, dass der Positionsfehler in x-Richtung immer noch bis zu 0.5m betragen kann. Vielversprechender ist es, das

Muster mehrmals zu messen und anschliessend die Position aus dem Mittelwert der gemessenen Winkel zu bestimmen.

6.3 Positionsaufdatierung beim Fahren

Ist die Anfangsposition vor der Andockstelle bestimmt, so können wir die Position wiederum unter Verwendung des *Extended Kalman Filters* aufdatieren. Während ohne Muster vor allem das Bestimmen gültiger Messungen ein Problem ist, liegt hier die Hauptschwierigkeit darin, dass alle Messungen in die gleiche Richtung gemacht werden müssen, worunter die Positionsgenauigkeit in Richtungen senkrecht dazu leidet.

Im folgenden schlagen wir drei Varianten vor, wie die Position vor einer Andockstelle aufdatiert werden kann. Die dabei erzielten Resultate sind in Fig 6.4 abgebildet.

Alle drei Varianten sind direkt vom Vorgehen abgeleitet, wie es in Kapitel 5.5 beschrieben ist. Sie unterscheiden sich nur darin, was für Messungen verwendet werden. Dies verlangt eine Anpassung bei der Bestimmung der Messmatrix $\mathbf{H}_k$ und der Sensorfehlermatrix $\mathbf{R}_k$.

Bei der ersten Variante benutzen wir je eine Abstands- und eine Richtungsmessung für die Verbesserung der Positions- und Orientierungsschätzung. Dadurch erhalten wir folgende Aufdatierungsgleichungen:

$$\hat{\mathbf{x}}_{k/k-1} = \text{odometrisch bestimmte Position} \tag{6.8}$$

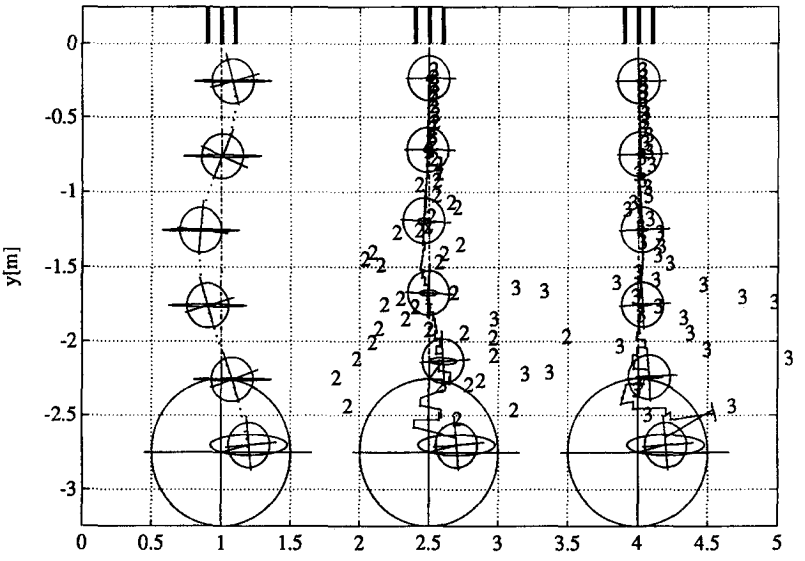
$$\mathbf{P}_{k/k-1} = \mathbf{C}(\mathbf{x}_k) \tag{6.9}$$

$$\mathbf{R}_k = \begin{bmatrix} \sigma^2_{\Delta d} & 0 \\ 0 & \sigma^2_{\Delta\beta} \end{bmatrix} \tag{6.10}$$

$$\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1}) = \begin{bmatrix} \cos\hat{\gamma}_{k/k-1} & \sin\hat{\gamma}_{k/k-1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

$$\mathbf{K}_k = \mathbf{P}_{k/k-1}\mathbf{H}_k^T(\hat{\mathbf{x}}_{k/k-1}) [\mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1})\mathbf{P}_{k/k-1}\mathbf{H}_k^T(\hat{\mathbf{x}}_{k/k-1}) + \mathbf{R}_k]^{-1} \tag{6.12}$$

$$\hat{\mathbf{x}}_{k/k} = \hat{\mathbf{x}}_{k/k-1} + \mathbf{K}_k \begin{bmatrix} \Delta d \\ \Delta\beta \end{bmatrix} \tag{6.13}$$



Figur 6.4: *Simulation von drei Andockmanövern unter Verwendung von verschiedenen Aufdatierungsvarianten. Bei der Fahrt links werden der senkrechte Abstand zum Muster und die Richtung zum mittleren Streifen gemessen. Bei der mittleren Fahrt werden die Richtungen zu den drei Streifen bestimmt und daraus die Position relativ zum Andockmuster berechnet. Die Fahrt rechts unterscheidet sich von der mittleren Fahrt dadurch, dass zusätzlich der senkrechte Abstand zum Muster gemessen wird.*

$$\mathbf{P}_{k/k} = [\mathbf{I} - \mathbf{K}_k \mathbf{H}_k(\hat{\mathbf{x}}_{k/k-1})]\mathbf{P}_{k/k-1} \tag{6.14}$$

Bei der zweiten Variante bestimmen wir die Roboterposition (x, y, ϕ) mit dem im vorigen Unterkapitel angegebenen Verfahren. Anschliessend wenden wir die Aufdatierungsgleichungen an, als ob wir (x, y, ϕ) einzeln gemessen hätten. Für $\mathbf{H}_k$ erhalten wir so die (3x3)-Einheitsmatrix und für $\mathbf{R}_k$ eine Diagonalmatrix mit den Elementen σ_x^2, σ_y^2 und σ_ϕ^2. Die Abschätzung von $\mathbf{R}_k$ ist etwas schwierig, weil (x, y, ϕ) nicht gemessene, sondern aus δ, ϵ und β abgeleitete Grössen sind. Für die Bestimmung von σ_x und σ_y begnügen wir uns deshalb mit einer Abschätzung, die sich auf die in Fig. 6.3 dargestellten Resultate abstützt: $\sigma_x = \sigma_y \approx 0.075 * a^3[m]$. σ_ϕ ergibt sich aus der Genauigkeit der Winkelmessung.

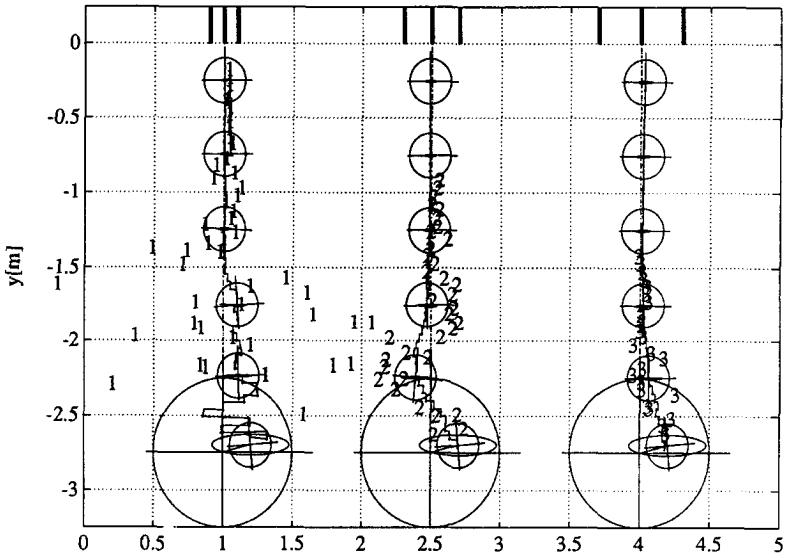
Die dritte Variante ist eine Erweiterung von Variante 2. Diese Erweiterung besteht darin, dass nach jedem Aufdatierungsschritt die Positionsschätzung mit einer Abstandsmessung senkrecht zum Muster verbessert wird.

Wie gesagt sind die Resultate der mit den drei Verfahren erzielten Resultate in Fig 6.4 dargestellt. Der Roboter startet dabei jeweils auf der Linie [y = -2.75m] und bewegt sich mit konstanter Geschwindigkeit auf die Andockstelle zu. Wir nehmen an, dass die geschätzte Anfangsposition um $(0.2m, 0.05m, 5°)$ von der richtigen Position abweicht. Dieser Positionsfehler muss auf der Fahrt in Richtung Andockstation abgebaut werden. Die Breite des Andockmusters ist so gewählt, dass im angedockten Zustand das ganze Muster noch im Sichtbereich der Kamera liegt. Die in der Mitte und rechts eingezeichneten Ziffern 2 und 3 zeigen die Positionen an, die auf Grund der Winkelmessungen errechnet wurden. Auch hier ist die vom Abstand abhängige Streuung der Werte sichtbar.

Bei Variante 1 ergibt sich das Problem, dass der Abstandsfehler zwar gut korrigiert wird, dass aber die Fehler in x und in ϕ nicht gleichzeitig abgebaut werden können. Ist die Position richtig, so ist die Schätzung der Orientierung schlecht und umgekehrt, was zu einer typischen Pendelbewegung um die richtige Position führt.

Variante 2 und Variante 3 heben sich deutlich vom ersten Verfahren ab, wobei beim letzten Verfahren die richtige Position schneller erreicht wird als bei Variante 2. Unschön ist allerdings, dass bei grösseren Distanzen, d.h. am Anfang des Andockvorgangs, die Positionsschätzung grosse Sprünge macht, was bei Betrachtung von Fig. 6.3 nicht verwundert.

Störend bei den Varianten 2 und 3 ist, dass das ganze Andockmuster bis am Schluss im Sichtbereich der Kamera liegen soll und deshalb sehr schmal sein muss. Dies zwingt uns zudem, die Kamera irgendwo in der Tiefe des Roboters zu plazieren, wo die Sichtverhältnisse ungünstig sind. Das Andockmuster könnte breiter ausgelegt und die Kamera ganz vorne montiert werden, wenn nicht bis am Schluss alle drei Streifen gesehen werden müssten. Dies ist durch eine Kombination von Variante 3 mit Variante 1 möglich. Beim Heranfahren an die Andockstelle wird die Position solange mit Variante 3 aufdatiert, bis nicht mehr das ganze Andockmuster sichtbar ist. In der Nähe der Andockstelle wird

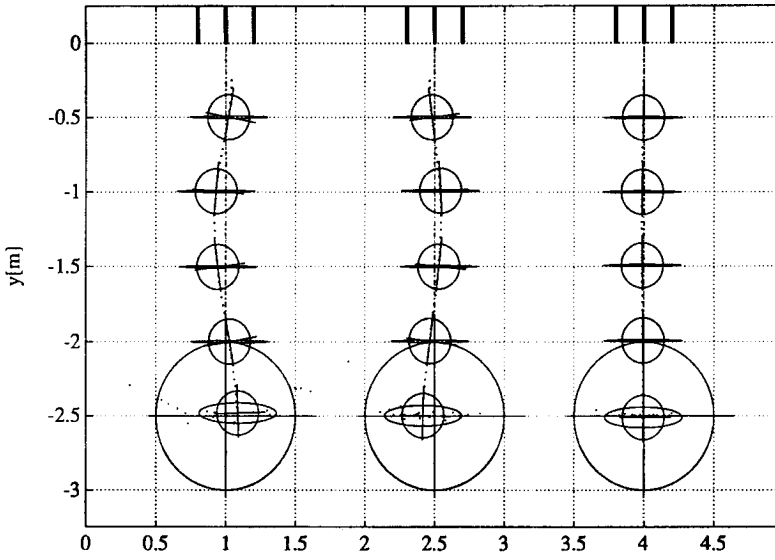


Figur 6.5: *Simulation von drei Andockmanövern unter Verwendung von verschieden grossen Andockmustern.*

nach Variante 1 vorgegegangen. Dies bedingt, dass die Position mit Variante 3 bereits recht genau bestimmt wurde, was durch die Breite des Andockmusters erleichtert wird. Der Nachteil dieser Methode ist allerdings, dass nicht mehr von überall angedockt werden kann. Steht der Roboter bereits zu nahe an der Andockstelle, so muss er sich soweit davon entfernen, bis das ganze Andockmuster wieder sichtbar ist.

Fig. 6.5 zeigt das Andocken mit verschieden grossen Andockmustern. Je grösser dieses ist, umso schneller kann eine gute Schätzung der Position erwartet werden.

Bis jetzt sind wir davon ausgegangen, dass die Roboterposition von Anfang an ungefähr bekannt sei. Andernfalls hätten wir keine gezielten Abstandsmessungen machen können. Lassen wir diese Annahme fallen, so müssen wir zuerst versuchen, die Position des Roboters zu ermitteln. Dazu bestimmen wir mehrere Male ξ und δ, berechnen deren Mittelwert und daraus die Position relativ zur Andockstelle. Nun könnten wir eine gezielte Abstandsmessung in Richtung Andockstelle durchführen, was uns nachher die Berechnung der Roboterposition erlaubt. In Fig 6.6



Figur 6.6: *Aufdatierung mit verbesserter Initialisierung und mit folgenden Unsicherheiten in den Winkelmessungen - linke Fahrt:* $\sigma_\delta = 0.1°$*,mittlere Fahrt:* $\sigma_\delta = 0.05°$*, rechte Fahrt:* $\sigma_\delta = 0.025°$*.* $(\alpha = 0.1,\ ds = 0.05m)$

sind drei Versuche mit verschiedenen Werten für σ_δ bzw. σ_ϵ dargestellt $(0.1°, 0.05°, 0.025°)$. Die Winkel ξ und δ wurden je 20 Mal berechnet und gemittelt. Die Streifen des Andockmusters liegen 0.2m auseinander. Bei einem Kameraöffnungswinkel von 20° kann damit bis in eine Entfernung von ca. 1.2m das ganze Muster gesehen werden. Die Aufdatierung beim Heranfahren an die Andockstelle erfolgte mit dem Verfahren 1. Als zusätzliche Schwierigkeit wurde der Radius des rechten Rades um 3% zu gross gewählt, womit ein systematischer Odometriefehler resultierte. Der Vergleich der drei Fahrten zeigt, dass ein gutes Andocken möglich ist, falls die Anfangsposition genau bestimmt wird. Sind die Winkelmessungen zu ungenau, so kann versucht werden, die Genauigkeit der Positionsbestimmung durch Erhöhung der Anzahl Messungen zu verbessern.

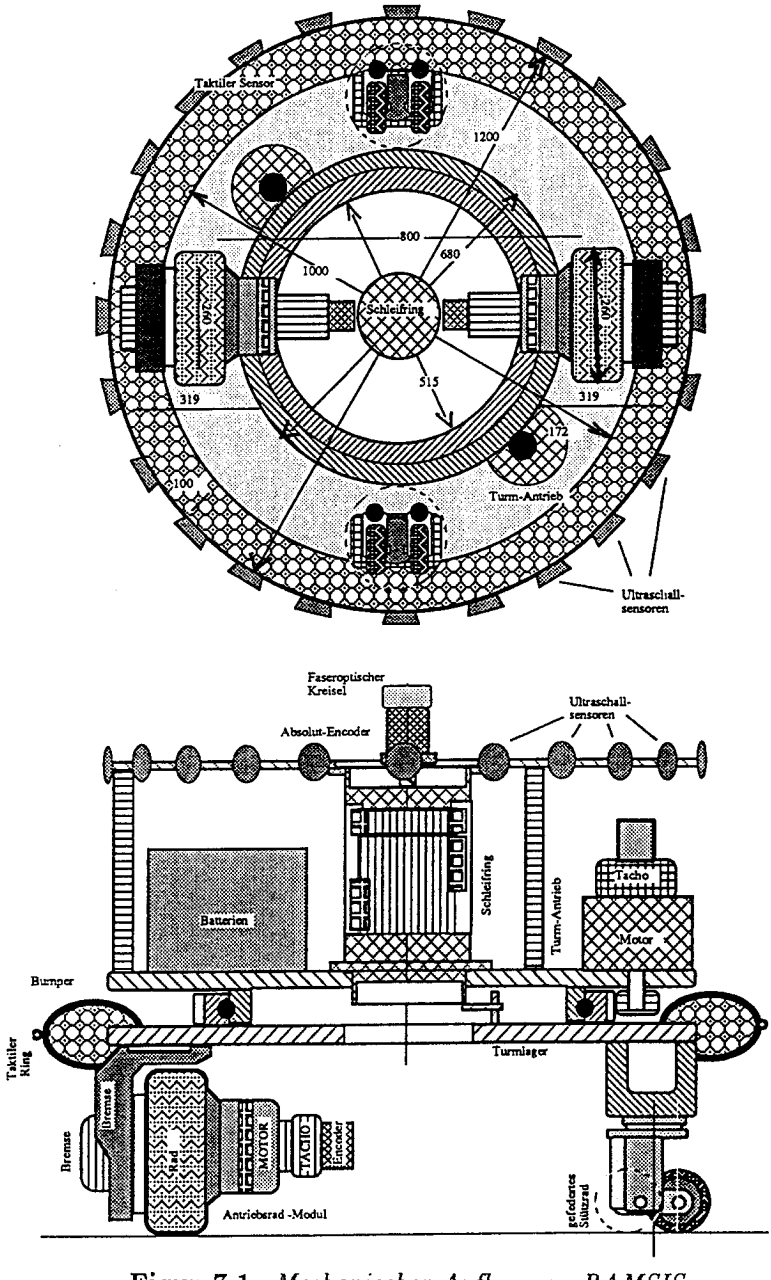
Kapitel 7

Implementation

Da sich ein mobiler Roboter in einer Umgebung bewegt, die sich nur unzureichend mathematisch beschreiben lässt, kommt der praktischen Erprobung des Positionsaufdatierungsverfahrens eine wichtige Bedeutung zu. In diesem Kapitel werden zuerst der Aufbau und die Sensorsysteme von RAMSIS vorgestellt. Nach einigen Ausführungen zur allgemeinen Softwarestruktur werden implementationstechnische Einzelheiten der Initialisierungsphase, der Aufdatierungsphase und des Andockens erklärt. Am Schluss dieser drei Unterkapitel werden jeweils die erreichten Resultate dargestellt.

7.1 Mechanischer Aufbau von RAMSIS

Der mechanische Aufbau von RAMSIS ist in Fig. 7.1 dargestellt. RAMSIS misst 1.2m in der Breite und 1.3m in der Höhe. Sein Gesamtgewicht beträgt über 400kg. RAMSIS hat zwei fest montierte angetriebene Räder und zwei frei bewegliche, gefederte Stützräder. Als Besonderheit ist der obere Teil gegenüber dem unteren verdrehbar. Durch geschickte Massenverteilung, d.h. indem der untere Teil möglichst leicht und damit beweglich gemacht wird, kann eine hohe Fahrdynamik erreicht werden. Zudem können durch die Verdrehbarkeit des Turmes darauf aufgebaute Sensorsysteme einfach in eine gewünschte Richtung gebracht werden.



Figur 7.1: *Mechanischer Aufbau von RAMSIS*

7.2 Das VME-Mehrprozessorsystem

7.2.1 Der VME-Bus

Der VME-Bus ist ein System, mit welchem Datenverarbeitungsgeräte, Datenspeichergeräte und Ein-/Ausgabegeräte eng gekoppelt miteinander verbunden werden können. Seine genauen Spezifikationen, bei deren Definition die untenstehenden Ziele verfolgt wurden, sind in [VME Specs] zu finden:

- Zwei am VME-Bus angeschlossenen Geräten sollte es möglich sein zu kommunizieren, ohne dass die internen Aktivitäten der anderen angeschlossen Geräte gestört werden.
- Elektrische und mechanische Systemeigenschaften sollten definiert sein, die für das einwandfreie Funktionieren der Kommunikation mit anderen am Bus angeschlossenen Geräten einzuhalten sind.
- Es sollte ein System zur Verfügung gestellt werden, dessen Leistungsfähigkeit in erster Linie von den Geräten selber und nicht von dessen Verbindungssystem begrenzt wird.

Auf dem VME-Bus werden die Daten asynchron mit einer theoretischen Höchstgeschwindigkeit von 40MByte/sec übertragen. Die Übertragung erfolgt parallel, wobei die Datenbreite variiert werden kann. Der exklusive Zugriff auf den Bus wird durch eine Arbitrierungsschaltung hardwaremässig gelöst.

7.2.2 Das VME-Mehrprozessorsystem von RAMSIS

Das VME-Mehrprozessorsystem von RAMSIS verfügt über 12 Steckplätze, die mit vier Prozessorkarten, drei Interfacekarten, zwei Speicherkarten und mit einer Koppelkarte zu einem ebenfalls VME-basierten Visionsystem aufgefüllt sind. Änderungen und Erweiterungen sind schnell und einfach möglich, was eine jederzeitige Anpassung an die jeweiligen Bedürfnisse an Rechenleistung erlaubt.

7.2.3 Das Echtzeitbetriebssystem ALBATROS

ALBATROS ist eines der vielen Betriebssysteme, die für VME-Mehrprozessorsysteme angeboten werden. Seine Besonderheiten sind (siehe [Albatros Manuals]):

- Unterstützung von Echtzeitprozessen
- Befehle, mit denen mechanische Einrichtungen wie z.B. Roboter direkt angesteuert werden können
- fehlende Unterstützung von Festspeichermedien
- Möglichkeit, eigene anwendungsspezifische Befehle zu definieren

7.3 Sensorsysteme und Datenaufnahme

Hier werden diejenigen Sensorsysteme von RAMSIS vorgestellt, welche für die Positionsbestimmung von Bedeutung sind.

7.3.1 Encoder

An den Achsen der beiden Antriebsmotoren sind optische Inkrementalencoder mit einer Auflösung von 1000 Strichen plaziert. Unter Einberechnung der Antriebsuntersetzungen, den Radradien, sowie der Vierfachauswertung der optischen Pulse ergibt sich eine Auflösung von $2\mu m$ pro Puls.

Die Verdrehung zwischen der unteren Plattform und dem Turm wird mit einem Absolutencoder mit 1024 Strichen gemessen, was eine Auflösung von 0.35° ergibt.

7.3.2 Ultraschallsensoren

Abstandsmessung mit Polaroid-Ultraschallsensoren

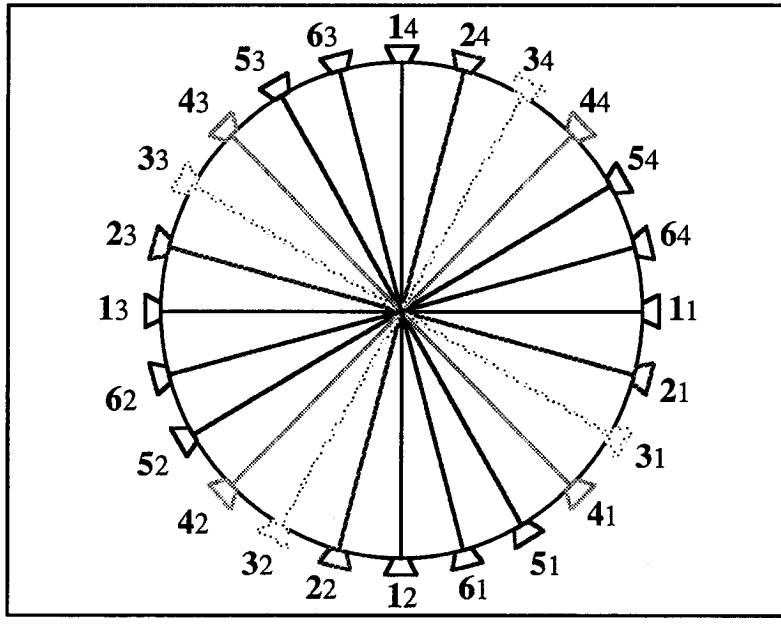
Ein akustischer Transducer (Sender/Empfänger) bildet zusammen mit einem elektronischen Schaltkreis eine Distanzmesseinheit. Damit kann der Abstand zu einem Objekt gemessen werden, welches zwischen 0.3m und 10m entfernt liegt. Zu diesem Zweck wird ein akustischer Puls ausgesendet und auf dessen Echo gewartet. Die dabei verstrichene Zeit dient als Mass für den Abstand zum Objekt.

Der wichtigere Teil der Messeinheit ist der Transducer. Dieser dient sowohl als Lautsprecher als auch als Mikrophon. Wenn die Einheit aktiviert wird, sendet der Transducer 56 Schallpulse mit einer Frequenz von 49.1kHz aus. Danach wird auf Mikrofonmodus umgeschaltet und auf das erste Echo gewartet. Weil die Stärke des Echos mit zunehmendem Abstand abnimmt, muss die Ansprechschwelle der verstrichenen Zeit angepasst werden. Ohne diese Anpassung könnten befriedigende Messwerte nur in einem sehr kleinen Distanzbereich erzielt werden.

Anordnung und Ansteuerung der Ultraschallsensoren

Aus Fig. 7.2 kann die Anordnung der 24 Ultraschallsensoren ersehen werden. Jeder Sensor ist mit seiner Gruppen- bzw. Knotennummer, sowie mit der Position innerhalb der Gruppe beschriftet. Die Sensoren einer Gruppe sind um 90° gegeneinander versetzt. Der Abstand der einzelnen Gruppen beträgt 15°. Sensormessungen erfolgen gruppenweise, d.h. es werden immer vier Sensoren gleichzeitig aktiviert. Um möglichst wenig Probleme mit Übersprechen und mit Mehrfachreflektionen zu haben, werden die Gruppen in der Reihenfolge 1-4-2-6-3-5-1.. angesprochen.

Hardwaremässig sind die 24 Sensoren über einen speziellen Bus miteinander verbunden. Über eine Zwischenelektronik ist dieser Bus an eine Prozessorkarte (MVME101 von Motorola) angekoppelt. Diese steuert die Messungen und wertet sie aus. Während den Messungen ist die Prozessorkarte blockiert und kann nicht für andere Zwecke eingesetzt werden. Verstreichen für eine Messung mehr als 60msec (entspricht einer Distanz von ca. 10m), so wird diese abgebrochen. Dadurch wird



Figur 7.2: *Anordnung und Gruppierung der Ultraschallsensoren*

verhindert, dass der Prozessor zu lange blockiert bleibt.

7.3.3 Faseroptischer Kreisel

In der Mitte von RAMSIS ist ein faseroptisches Kreisel plaziert. Damit können Drehungen um die Vertikalachse gemessen werden. Die erlaubte Drehgeschwindigkeit liegt im Bereich von $\pm 150°/sec$. Die Driftrate des Kreisels liegt im Bereich von 1..10°/Stunde. Gegenüber herkömmlichen mechanischen Kreiseln haben faseroptische Kreisel verschiedene Vorteile:

- keine beweglichen Teile und damit keine Anlaufzeit
- hohe Bandbreite
- kostengünstig im angegebenen Genauigkeitsbereich
- von der Beschleunigung unabhängige Drift

- wartungsfrei

7.3.4 Bildverarbeitungssystem

Das von uns benutzte Bildverarbeitungssystem ist in [Engel&Stiefvater 89] ausführlich beschrieben. Es besteht in seiner Grundausführung aus sieben Doppeleuropakarten, bis zu vier Kameras, einem Bildschirm und einem Terminal. Die Verbindung der Karten geschieht über einen VME-Bus, über mehrere Bildbusse und/oder über den Bildspeicherbus. Die sieben Karten haben folgende Aufgaben:

1. CPU-Karte (Motorola68000, 16MHz, 2MByte RAM):
   Steuert alle Bildverarbeitungsfunktionen. Zu diesem Zweck ist die CPU-Karte über den VME-Bus mit dem Bildspeicher verbunden.
2. Interfacebaugruppe:
   Digitalisiert das Videosignal der CCD-Kamera.
3. Videoadressierungsbaugruppe:
   Übernimmt über den Bildbus das digitalisierte Videosignal und schreibt es via den Bildspeicherbus an die richtige Stelle des Bildspeichers.
4. Bildspeicherbaugruppe:
   Auf 1MByte können die Bilddaten abgespeichert werden. Dieser Speicherbereich ist gleichzeitig vom Signalprozessor, über den VME-Bus, vom Bildeinzug und von der Bilddarstellung zugreifbar.
5. High-Speed-Interface:
   Ermöglicht wahlfreien, bidirektionalen 128bit breiten Zugriff des Signalprozessors auf den Bildspeicher.
6. Signalprozessorbaugruppe:
   Führt mit Hilfe eines ADSP2100 Signalprozessors Manipulationen an den Bilddaten durch. Die Befehle dazu werden von der CPU-Karten generiert und vom Signalprozessor via Bildspeicher übernommen.
7. Falsch-Farb-Graphik-Baugruppe:
   Stellt den Monitorbereich des Bildspeichers auf einem Bildschirm dar.

Die Lesegeschwindigkeit beträgt 30 Bilder/sec. Ein Bild besteht aus 512x485 Pixeln. Bei einer Auflösung von 8bit pro Pixel ergibt sich so ein Speicherbedarf von 256kByte. Im Bildspeicher können also höchstens vier Vollbilder abgespeichert werden.

Zum System gehört eine SONY XC77 CCD-Kamera mit einer Auflösung von 768x493 Pixeln. Dies bedeutet, dass vom Bildverarbeitungssystem nur 2/3 des aufgenommenen Bildes verarbeitet werden können. Das rechte Drittel des Bildes geht verloren. Ist die Kameralinse nicht verzerrungsfrei, so führt dies zu asymmetrischen Verzerrungen im verarbeiteten Teil des Bildes, was die Messgenauigkeit von der Lage der Kamera abhängig macht.
Für die Kamera werden verschiedene Objektive angeboten. Wir verwenden ein Weitwinkelobjektiv mit einer Öffnung von 33°.

Zum Bildverarbeitungssystem gehört auch eine Softwarebibliothek mit einer umfangreichen Sammlung von Funktionen für die Aufnahme, Manipulation und Darstellung von Bildern. Die für uns wichtigste Operation ist die Bildsegmentierung, mit welcher das Bild in Objekte eingeteilt wird, welche nach Grösse geordnet ausgegeben werden. Die Segmentierung kann allerdings nur am Binärbild durchgeführt werden.

7.4 Softwarekonzept von RAMSIS

RAMSIS hat eine *Verhaltensbasierte Regelungsstruktur*. Bei deren Entwurf wurde darauf geachtet, dass möglichst wenig Information zwischen den einzelnen Verhaltensebenen ausgetauscht werden muss. Das gleiche Ziel wird bei der Aufteilung einer Software auf ein Mehrprozessorsystem verfolgt - zwischen den einzelnen Prozessoren sollten möglichst wenig Daten verschoben werden müssen. Die naheliegende Lösung, pro Verhaltensebene eine Prozessorkarte einzusetzen, führt aber zu einer ineffizienten Ausnutzung der Rechenleistung, weil der Rechenaufwand in den einzelnen Verhaltensebenen sehr unterschiedlich ist. Aus diesem Grunde werden sogenannte Funktionseinheiten definiert. Darin werden logisch zusammenhängende Aufgaben so zusammengefasst, dass sie auf einer oder mehreren gut ausgelasteten Prozessorkarten ausgeführt werden können.

7.4.1 Einteilung in Funktionseinheiten

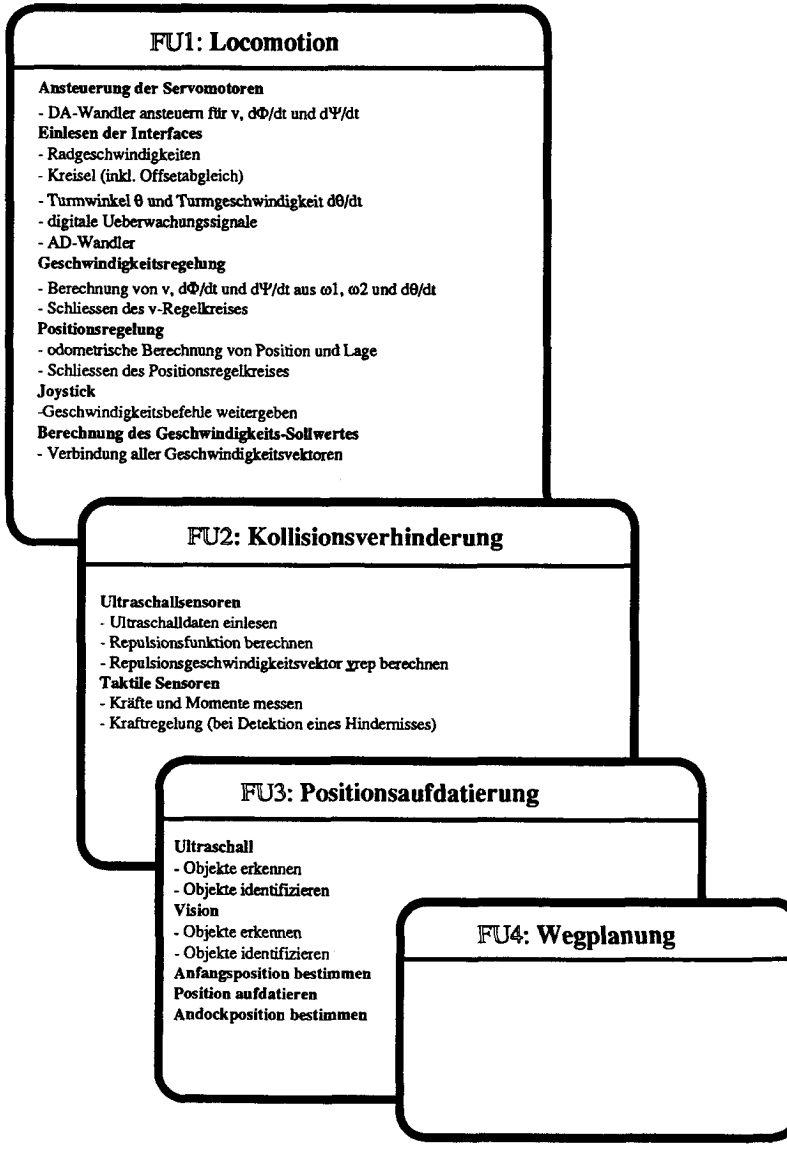
Die Einteilung in Funktionseinheiten entspricht weitgehend der Einteilung in Verhaltensebenen und kann aus Fig. 7.3 ersehen werden. In der ersten Funktionseinheit (FU1) sind die Verhaltensebenen *Roboterregelung* und *Positionsregelung* realisiert. Als weitere Funktion wird hier die Ein- und Ausgabe der von diesen Ebenen benötigten Sensorsignale ausgeführt. In FU2 sind die Funktionen für die Kollisionsverhinderung zusammengefasst. Ultraschallsensoren und taktile Sensoren werden auf dieser Ebene eingelesen. Die nächsten Funktionseinheiten beinhalten die Positionsaufdatierung und die Wegplanung. Die Positionsaufdatierung kann auf die in FU1 und FU2 aufgenommenen Sensordaten zugreifen. Weiter kann mit einem Visionsystem Bildinformation gewonnen werden.

Für die Implementation ist die Aufteilung der Funktionseinheiten innerhalb des VME-Mehrprozessorsystems von Bedeutung. Diese ist in der nächsten Figur dargestellt (7.4).

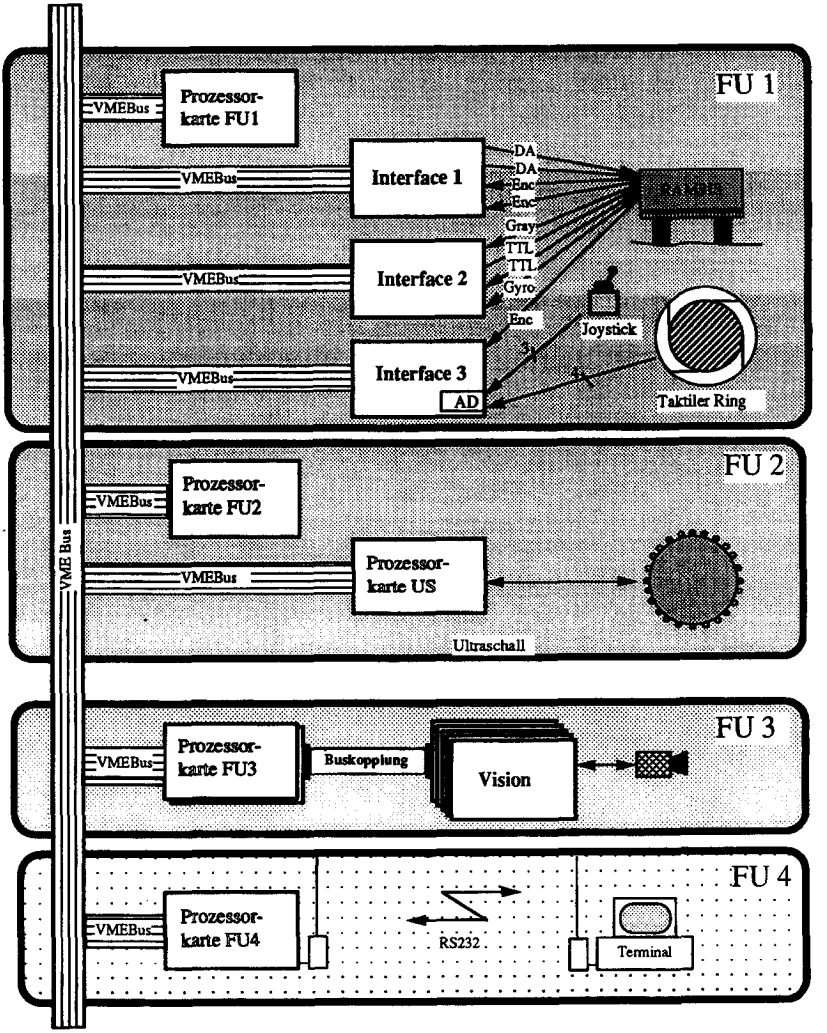
7.4.2 Datenkommunikation

Das Zusammenwirken der Funktionseinheiten ist nur möglich, wenn Daten untereinander ausgetauscht werden können. Dieser Datenaustausch soll aber erstens so gering wie nötig und zweitens so schnell wie möglich sein. Mit der Einteilung in Funktionseinheiten wird dem ersten Punkt entsprochen. Für einen schnellen Datentransfer garantiert der VME-Bus.

Im folgenden interessieren uns nur diejenigen Daten, die von mehr als einer Funktionseinheit benötigt werden. Wir bezeichnen sie als globale Daten. Globale Daten können entweder zentral oder dezentral organisiert werden. Bei der zentralen Lösung wird ein Speicherbereich definiert, wohin alle globalen Daten geschrieben und auch wieder gelesen werden können. Mit einem Verriegelungsmechanismus garantiert man, dass Datensätze nicht gleichzeitig gelesen und geschrieben werden können. Bei der dezentralen Lösung wird jeder Funktionseinheit ein Speicherbereich zugewiesen, in den die von ihr produzierten globalen Daten abgelegt werden können. Gleichzeitiger Datenzugriff ist auch hier zu verhindern.



Figur 7.3: *Einteilung in Funktionseinheiten (FU)*



Figur 7.4: *Funktionseinheiten auf dem VME-Mehrprozessorsystem*

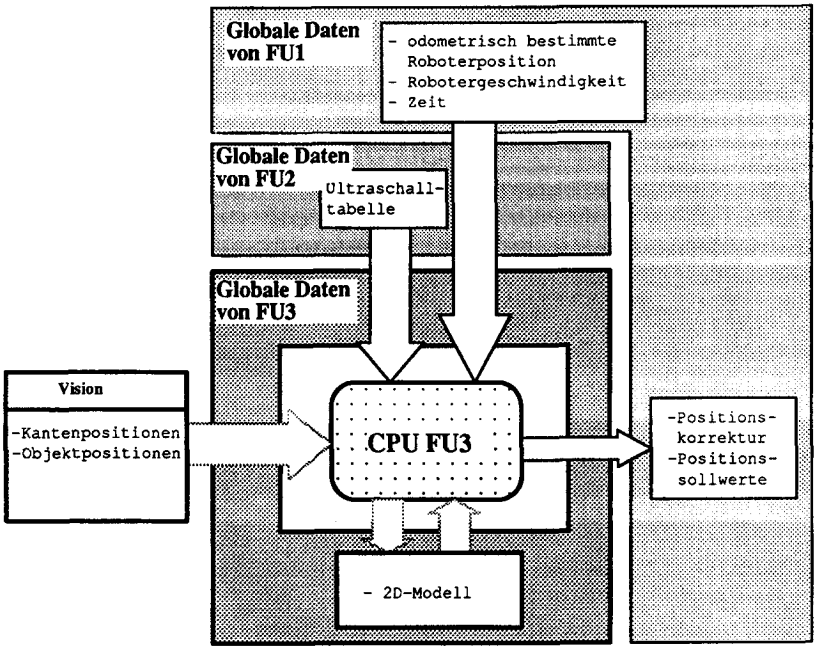
Auf einem VME-Mehrprozessorsystem sind beide Lösungsvarianten einfach zu realisieren. Die zentrale Ablage der globalen Daten ist organisatorisch einfacher, hat aber den Nachteil, dass viele Daten über den VME-Bus transportiert werden müssen. Mit der dezentralen Lösung können unsere Forderungen bezüglich Menge und Geschwindigkeit des Datenaustausches besser erfüllt werden. Dessen Organisation ist in unserem Fall einfach, wenn man sich an folgende Regeln hält:

- Global benötigte Sensordaten werden in derjenigen Funktionseinheit abgelegt, in der sie erzeugt werden.
- Sollwerte aller Art werden in der ersten Funktionseinheit abgelegt, in der sie wirksam werden. Z.B. werden die Geschwindigkeitssollwerte in FU1 abgelegt, da sie direkt von der Roboterregelung benötigt werden.
- Zustandsgrössen gehören in diejenige Funktionseinheit, in der sie durch Messung oder Berechnung ermittelt werden.

Die für die Positionsaufdatierung notwendigen Datentransfers sind in Fig. 7.5 dargestellt. Die Positionskorrektur muss nach FU1 geschrieben werden, da die Roboterposition dort abgelegt ist. Die Ultraschallmessungen werden in FU2 produziert und können von dort übernommen werden. Das bedeutet aber, dass auf die Ansteuerung der Ultraschallsensoren kein direkter Einfluss genommen werden kann. Ein indirekter Einfluss ist möglich, weil Sollwerte für den Positionsregelkreis gegeben werden können.

7.4.3 Benutzerinterface

RAMSIS kann über eine serielle Schnittstelle mit einem Leitrechner verbunden werden, wozu aber eine entsprechende Kommunikationssoftware benötigt wird. In der Testphase genügt es, wenn über ein Terminal auf die einzelnen Prozessorkarten zugegriffen werden kann. Mit der Positionsaufdatierungsebene (FU3) wird über einfache Menubefehle ohne irgendwelche Grafik kommuniziert. Die Eingabe des *Referenzmodelles* der Umgebung erfolgt off-line.

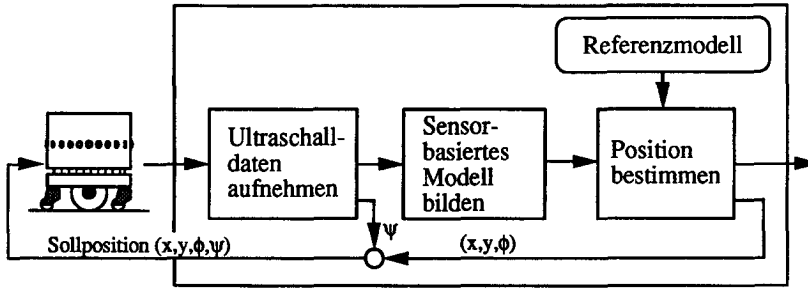


Figur 7.5: *Globale Daten für die Positionsaufdatierung*

7.5 Bestimmung der Anfangsposition

7.5.1 Übersicht

Die Bestimmung der Anfangsposition wird in drei Schritten durchgeführt (Fig 7.6). Als erstes müssen Ultraschalldaten aufgenommen werden, aus denen ein Ultraschallbild erzeugt werden kann. Für eine genaue Positionsbestimmung ist es wichtig, dass der Roboter in dieser Zeit seinen Standort nicht wechselt und dass Messungen in möglichst viele Richtungen erfolgen. Dies kann durch Drehen des Turmes (ψ) erreicht werden. Als zweites wird aus dem Ultraschallbild das *Sensorbasierte Modell* erzeugt. Durch Vergleich dieses Modelles mit dem *Referenzmodell* gelangt man zu der gesuchten Startposition des Roboters. Kann vom momentanen Standort aus die Position nicht er-



Figur 7.6: *Bestimmung der Anfangsposition*

mittelt werden, so müssen der Standort gewechselt und die drei Schritte nochmals durchgeführt werden.

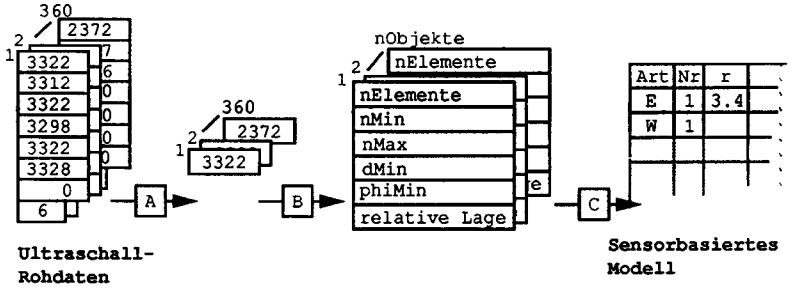
7.5.2 Aufnahme der Ultraschallmessungen

Die Ultraschallmessungen werden in FU2 durchgeführt, weil sie für die Kollisionsverhinderung benötigt werden. Um beim Auftreten von Hindernissen möglichst schnell reagieren zu können, ist die Ansteuerung der Sensoren von der Fahrgeschwindigkeit des Roboters abhängig. Im Stillstand werden alle 24 Sensoren aktiviert. Mit zunehmender Geschwindigkeit wird das Gesichtsfeld künstlich verengt und es werden nur noch Messungen in Fahrtrichtung und senkrecht dazu durchgeführt. Dies ist ebenfalls ein Grund, warum der Roboter seine Position während der Datenaufnahme nicht verschieben soll.

Für ein gutes Ultraschallbild sind etwa 500 Messungen notwendig, die möglichst gleichmässig in alle Richtungen verteilt sein sollen. Bei einer durchschnittlichen Messzeit von 40msec pro vier Messungen errechnet sich daraus ein Zeitbedarf von 5sec.

Die Drehung des Turmes könnte mit der Ansteuerung der Ultraschallsensoren synchronisiert werden, um die Weiterverarbeitung der Daten zu erleichtern. Dies erhöht aber die Messzeit, weshalb wir darauf verzichten.

7.5.3 Bildung des Sensorbasierten Modelles



Figur 7.7: *Bildung des Sensorbasierten Modells:*
A: Filtern und Ergänzen der Ultraschallmessungen
B: Segmentieren der gefilterten Daten und Berechnen von Merkmalen
C: Objekte erkennen und das sensorbasierte Modell aufstellen

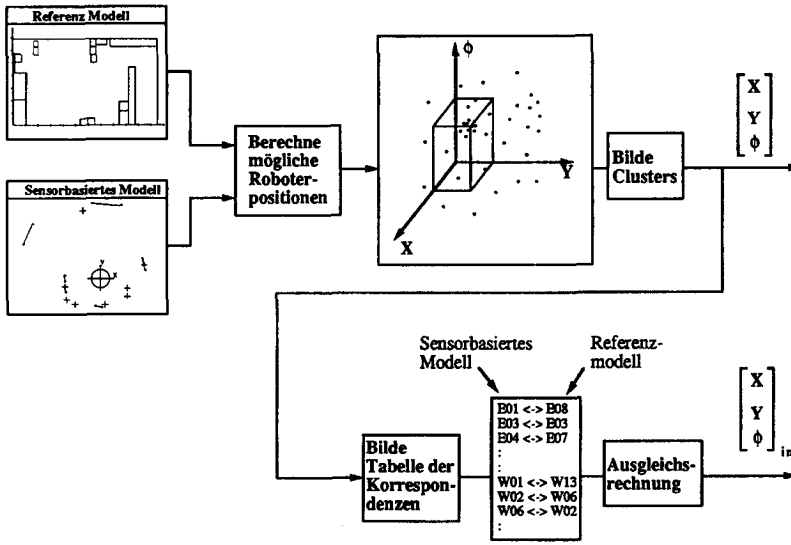
Während der Datenaufnahme werden alle Abstandsmessungen laufend nach der Messrichtung geordnet, wobei wir mit einer Auflösung von 1° arbeiten. Nach Beendigung der Messungen sind die Rohdaten in 360 Arrays abgelegt, wobei jedem Array eine Messrichtung entspricht (Fig. 7.7). Für die Berechnung des normalisierten Ultraschallbildes, in dem pro Richtung genau eine Messung gilt, sind die folgenden Schritte durchzuführen:

1. *Auswahl der besten Messung innerhalb jedes Arrays:* Dazu werden die Messungen geordnet und der mittlere Wert ausgewählt.

2. *Ergänzen von fehlenden Messungen:* Wegen der fehlenden Synchronisation von Messung und Turmdrehung ist es möglich, dass in einzelne Richtungen überhaupt keine Messungen gemacht werden. In diesem Fall wird der Wert der benachbarten Messung genommen

3. *Medianfilterung des Ultraschallbildes:* Liegt ein Messwert deutlich unter oder über den benachbarten Messwerten, so ist dieser als Ausreisser zu betrachten und durch den Mittelwert der benachbarten Messungen zu ersetzen.

Aus dem ausgeglichenen Ultraschallbild werden durch Segmentierung die Einzelobjekte extrahiert. Gleichzeitig werden für jedes dieser



Figur 7.8: *Berechnung der Startposition*

Einzelobjekte Merkmale berechnet, die für die Objektidentifikation und für die Bildung des *Sensorbasierten Modelles* benötigt werden. Dessen Berechnung erfolgt unter Anwendung des in Kapitel 4.2 eingeführten Klassifikators. Der gesamte Zeitbedarf für die Bildung des *Sensorbasierten Modelles* beträgt deutlich weniger als 1sec und kann damit gegenüber dem Zeitbedarf für die Datenaufnahme vernachlässigt werden.

7.5.4 Positionsbestimmung

Die Schritte für die Bestimmung der Anfangsposition sind in Fig. 7.8 zusammengefasst. Mit dem Clusteringverfahren aus Kapitel 4.4 wird die ungefähre Roboterposition bestimmt. Für die Berechnung der Datenmatrix werden alle Kombinationen von Modellobjektpaaren und Referenzobjektpaaren berücksichtigt. Dies führt aber schon bei relativ wenigen Objekten zu einer riesigen Datenmatrix. Diese kann durch Einschränkung des zulässigen Skalierungsbereichs und durch Wegstreichen von physikalisch unmöglichen Positionsvorschlägen wirkungsvoll verkleinert werden. Nach der Clusterbildung werden die berechneten

Clusters der Grösse nach sortiert. Der Mittelpunkt des grössten Clusters wird provisorisch als Roboterposition angenommen. Durch Plausibilitätstests wird diese Annahme bestätigt oder verworfen. Bei deren Ablehnung wird das gleiche mit dem zweitgrössten Cluster gemacht. Falls auch dessen Mittelpunkt nicht als Roboterposition in Frage kommt, so muss der Roboter seinen Standort wechseln und mit der Positionsbestimmung neu beginnen.

Ist die Clusteringmethode erfolgreich, so kann die Roboterposition, falls nötig, mittels Ausgleichsrechnung noch genauer bestimmt werden. Die Korrespondenzentabelle wird durch Anwendung des Matchingverfahrens berechnet. Dabei werden aber nur diejenigen Objekte berücksichtigt, die von der grob berechneten Roboterposition aus gesehen werden können.

Der Zeitbedarf für die Berechnung der möglichen Roboterpositionen und der Clusters variiert je nach Anzahl der Objekte im Sensorbasierten Modell und im Referenzmodell. Bei der hier verwendeten Laborumgebung wurden im Durchschnitt ca. 5sec benötigt.

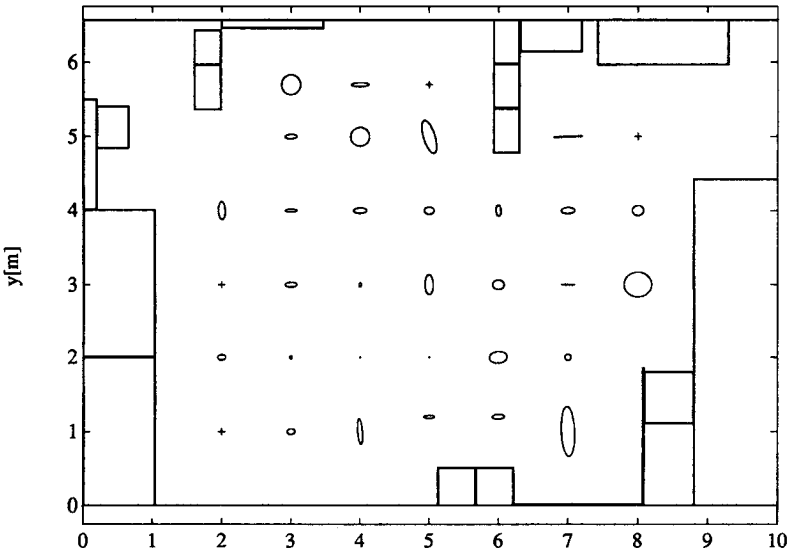
7.5.5 Resultate

Um Aussagen über den Zusammenhang zwischen dem Standort des Roboters und der Qualität der Positionsbestimmung machen zu können, wurde an 34 verschiedenen Orten die Position bestimmt und mit der tatsächlichen Position des Roboters verglichen. Die Parametereinstellungen und die strukturierte Umgebung stimmen dabei mit dem Beispiel in Kap.4.6 überein. Die genauen Resultate sind in Tab. 7.1 zusammengefasst. Die Abmessungen der Ellipsen in Fig. 7.9 vermitteln einen optischen Eindruck von der Grösse der Messfehler. Die Richtungen der Ellipsenachsen entsprechen den Orientierungsfehlern. Standorte, von denen aus die Positionsbestimmung nicht zum Erfolg führte, sind mit einem Kreuz(+) bezeichnet.

Statistisch gesehen lassen sich die Resultate in vier Gruppen einteilen. Gruppe 1 sind die sehr genauen Messungen, bei denen der Positionsfehler innerhalb von 3cm und der Orientierungsfehler innerhalb von 2° liegt. Am grössten ist Gruppe 2 mit Positionsfehlern um 5cm und Orientierungsfehlern um 2°. Gruppe 3 beinhaltet die Messungen mit so grossen Positions- und Orientierungsfehlern, dass diese nicht als

Position(x[m]/y[m])	Clusterpunkte	Δx[mm]	Δy[mm]	$\Delta\phi$[deg]
2.0/1.0	88	-	-	-
2.0/2.0	162	28	19	2
2.0/3.0	134	-	-	-
2.0/4.0	298	24	61	2
3.0/1.0	286	28	19	2
3.0/2.0	399	8	11	1
3.0/3.0	674	42	17	1
3.0/4.0	724	42	9	0
3.0/5.0	269	44	14	2
3.0/5.7	129	69	66	4
4.0/1.0	642	17	88	5
4.0/2.0	423	2	5	0
4.0/3.0	841	9	15	1
4.0/4.0	441	47	18	2
4.0/5.0	242	68	63	4
4.0/5.7	33	63	13	1
5.0/1.2	260	40	10	0
5.0/2.0	400	4	1	2
5.0/3.0	912	30	68	1
5.0/4.0	331	35	24	0
5.0/5.0	143	41	116	19
5.0/5.7	146	-	-	-
6.0/1.2	293	44	16	3
6.0/2.0	377	64	40	5
6.0/3.0	191	42	32	1
6.0/4.0	284	18	37	4
7.0/1.0	394	48	169	3
7.0/2.0	502	22	21	0
7.0/3.0	490	46	2	0
7.0/4.0	341	49	20	2
7.0/5.0	108	103	2	2
8.0/3.0	306	99	85	0
8.0/4.0	376	40	34	4
8.0/5.0	77	-	-	-

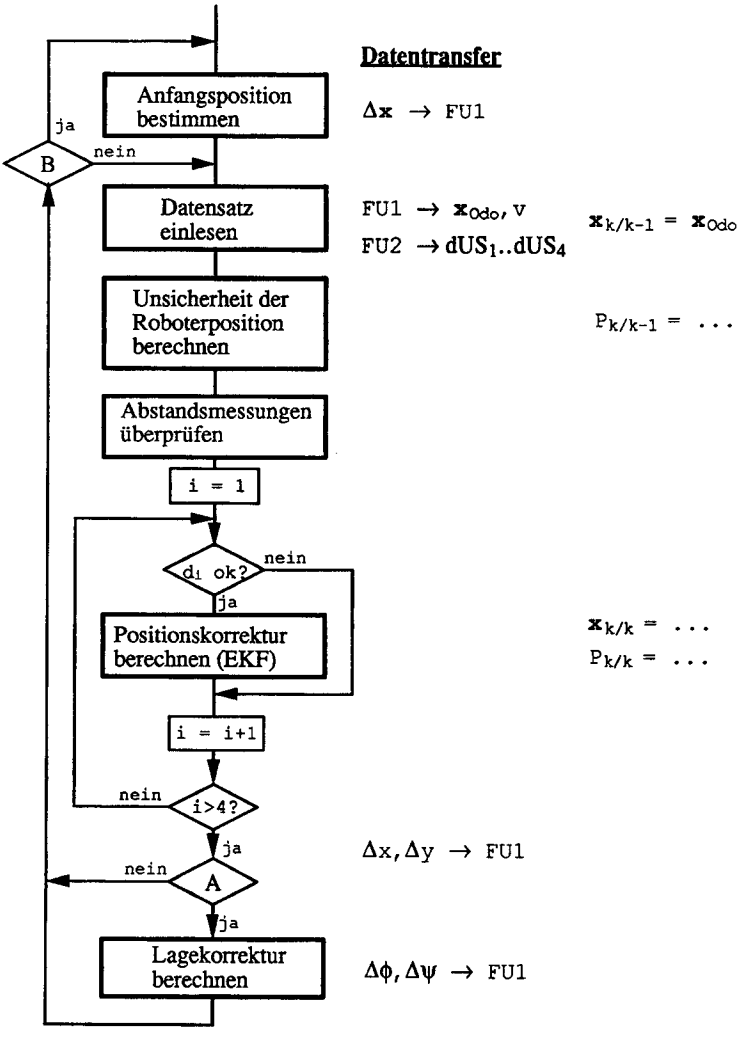
Tabelle 7.1: *Resultate der Bestimmung der Anfangsposition*



Figur 7.9: *Fehler bei der Bestimmung der Anfangsposition*

Startwerte für die Positionsaufdatierung verwendet werden können. In Gruppe 4 schliesslich sind die nicht erfolgreichen Messungen enthalten. Es fällt auf, dass die guten Messungen vor allem von Standorten in der Mitte des Raumes stammen, während es am Rand häufiger nicht möglich war, die Position zu bestimmen.

Die beiden Hauptfaktoren, welche die Positionsbestimmung beeinflussen, sind die Qualität des *Sensorbasierten Modelles* und die Einmaligkeit der Umgebung, in welcher sich der Roboter befindet.
Die Qualität des *Sensorbasierten Modelles* wird wesentlich von der Oberflächenbeschaffenheit, der relativen Lage und von der Menge der Umgebungsobjekte bestimmt. Die beiden ersten Faktoren spielen deshalb eine wichtige Rolle, weil sie die Reflektionseigenschaften von Ultraschall massgeblich beeinflussen. Der dritte Faktor ist insofern bedeutend, als bei grösserer Anzahl Objekte die Wahrscheinlichkeit steigt, dass genügend, d.h. mindestens drei Objekte richtig erkannt werden können.
Die Einmaligkeit der Umgebung ist entscheidend, weil die Positionsbestimmung letztlich auf der Unterscheidbarkeit von *Sensorbasierten Modellen* beruht, welche alle vom gleichen Raum aber von unterschiedlichen Standorten aus aufgenommen wurden.



A: Lageunsicherheit zu gross?
B: Positionsunsicherheit zu gross?

Figur 7.10: *Flussdiagramm der Positionsaufdatierung*

Wie bereits oben erwähnt, scheint es ungünstig zu sein, wenn der Roboter am Rand steht. Dies lässt sich dahingehend deuten, dass dort sowohl die Menge der sichtbaren Objekte kleiner ist als auch die Unterscheidbarkeit des *Sensorbasierten Modelles* abnimmt.

7.6 Aufdatierung der Position beim Fahren

7.6.1 Übersicht

Der gesamte Aufdatierungsvorgang lässt sich anhand von Fig. 7.10 erklären. Ist die Position des Roboters völlig unbekannt, so muss diese mit der Methode aus Kapitel 4 bestimmt werden. Dies entspricht der Initialisierung des Aufdatierungsvorganges. Ist die Anfangsposition bestimmt, so werden aus FU2 die vier aktuellsten Abstandsmessungen ($dUS_1..dUS_4$) und aus FU1 die Positionsschätzung $\mathbf{x}_{Odo}$ und die Geschwindigkeit v übernommen. Diese wird benötigt, um die Laufzeitfehler der Ultraschallmessungen kompensieren zu können. Danach wird mit dem in Kap.5.3 beschriebenen Verfahren die Unsicherheit der Roboterposition bestimmt. Als nächstes kommt der zeitaufwendigste Schritt, die Überprüfung der vier Abstandsmessungen. Diejenigen Messungen, die von einem Objekt im *Referenzmodell* stammen, werden dann für die Positionskorrektur verwendet. Bevor ein neuer Datensatz eingelesen wird, muss kontrolliert werden, ob die Bedingungen für eine Orientierungskorrektur erfüllt sind. Falls ja wird diese durchgeführt. Steigt aus irgendwelchen Gründen die Positionsunsicherheit über eine bestimmte Grenze, so muss der Aufdatierungsalgorithmus neu initialisiert werden. Dazu muss der Roboter zum Stehen gebracht werden. Ein Weiterfahren wäre sowieso nicht sinnvoll, da ohne genaue Positionsinformation das Ziel nicht erreicht werden kann.

7.6.2 Echtzeitaspekte

Für die erfolgreiche Realisierung der Positionsaufdatierung ist es wichtig, die zeitlichen Bedingungen der einzelnen Schritte zu analysieren. Damit wird auch die Frage beantwortet, ob die Aufdatierung synchron zu einem vorgegebenen Takt oder asynchron er-

folgen soll.

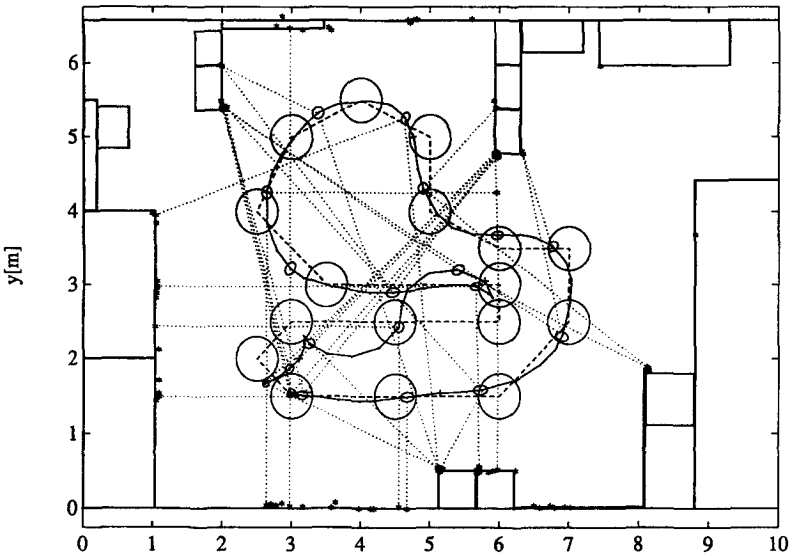
Bei einer synchronen Realisierung müsste in jedem Fall sichergestellt werden, dass auch im ungünstigsten Falle die Aufdatierungsschleife innerhalb des vorgegebenen Zeitintervalls beendet werden kann. Diese Lösung ist dann sinnvoll, wenn genügend Rechenzeitreserve vorhanden ist oder wenn nicht sehr häufig aufdatiert werden muss. Funktioniert die Aufdatierung nur, wenn so oft wie möglich korrigiert wird, so ist eine asynchrone Realisierung zu wählen, weil damit unnötiger Rechenzeitverlust vermieden wird.

7.6.3 Resultate

Ziel der Positionsaufdatierung ist es, die Roboterposition auch während dem Fahren jederzeit genügend genau zu kennen. Dies ergibt zusätzliche Probleme bei der Überprüfung der erzielten Resultate, weil nicht nur die Positionen, sondern auch deren Gültigkeitszeitpunkte berücksichtigt werden müssen. Dies bedeutet, dass es nicht mehr genügt, am Schluss eines Testlaufes die gemessene mit der tatsächlichen Position zu vergleichen. Idealerweise müsste der Weg des Roboters mit einer Referenzanlage aufgezeichnet und mit den mittels Positionsaufdatierung errechneten Werten verglichen werden. Weil wir nicht über eine solche Anlage verfügen, müssen wir uns damit begnügen zu untersuchen, ob die aufgenommenen Abstandsmessungen ins *Referenzmodell* passen. Berücksichtigt man, dass die Messgenauigkeit unserer Ultraschallsensoren besser als 1cm ist, so kann auch so gültig überprüft werden, ob der Positionsfehler innerhalb der geforderten Toleranz von 10cm liegt (Fahrt von Punkt zu Punkt). Die für das Abfahren von Trajektorien geforderte Genauigkeit von 1cm kann jedoch nicht ganz garantiert werden.

Alle Testfahrten haben wir in unserem Labor durchgeführt (Beschreibung siehe Kap.4.6). Sie wurden so ausgelegt, dass jeweils möglichst der ganze Raum abgefahren werden musste. Drei solcher Fahrten sollen im Folgenden genauer betrachtet werden:

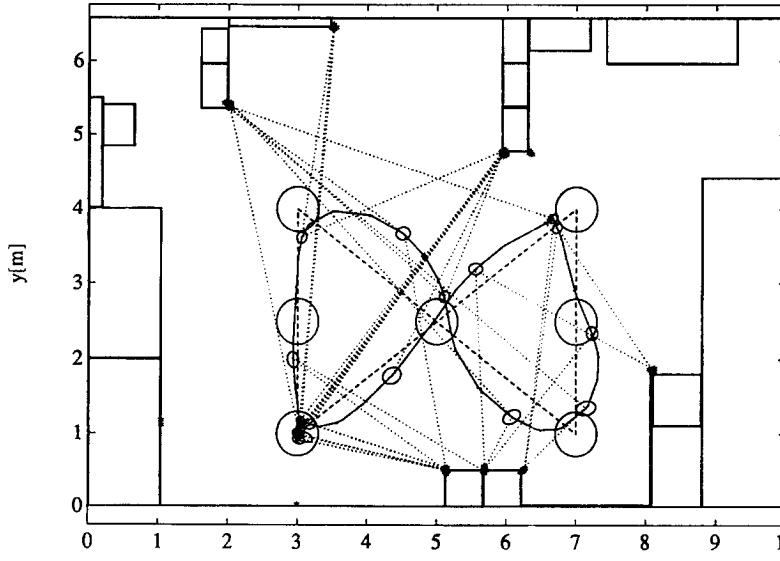
Fahrt 1: Bananenkurve (Kreisel eingeschaltet)
Fahrt 2: Achterkurve (Kreisel eingeschaltet)
Fahrt 3: Bananenkurve (Kreisel ausgeschaltet)



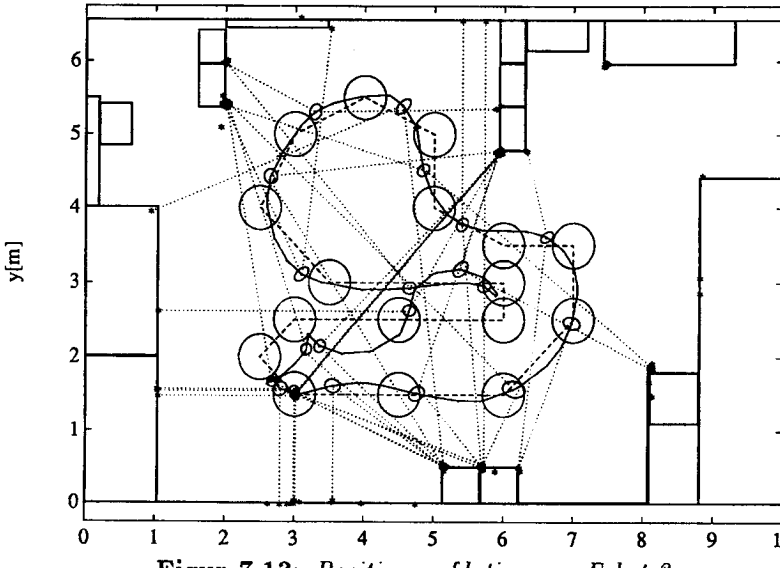
Figur 7.11: *Positionsaufdatierung: Fahrt 1*

Mit den beiden ersten Fahrten soll gezeigt werden, dass die Positionsaufdatierung unabhängig von der Kurvenform funktioniert. Die dritte Fahrt zeigt, dass unter Umständen auf einen Kreisel verzichtet werden kann, dass aber die Positionsunsicherheit dadurch grösser wird.

Die drei Fahrten sind in den Figuren 7.11, 7.12 und 7.13 dargestellt. Der abzufahrende Weg wurde nicht durch eine Trajektorie, sondern durch eine Liste von Sollpunkten vorgegeben. In den drei Figuren sind diese Sollpunkte zusammen mit ihrem kreisförmigen Einzugsgebiet dargestellt. Damit ist das Gebiet gemeint, in das der Roboter fahren muss, bevor der nächste Sollpunkt angesteuert werden kann. Mit der Grösse des Einzugsgebietes kann die Form des gefahrenen Weges beeinflusst werden. Zur Verdeutlichung des Sollpfades sind die Sollpunkte durch eine gestrichelte Linie verbunden. Der Sollpfad beginnt und endet bei allen drei Fahrten unten links, d.h. ungefähr bei Position [3.0m/1.5m]. Weiter sind in allen drei Figuren der tatsächlich gefahrene Weg, die Unsicherheitsellipsen (nach jeweils 3sec), sowie alle Abstandsmessungen (*) eingetragen. Um zu zeigen, von wo aus die Messungen gemacht wurden, sind diese teilweise mit der zugehörigen Roboter-



Figur 7.12: *Positionsaufdatierung: Fahrt 2*



Figur 7.13: *Positionsaufdatierung: Fahrt 3*

Fahrt	Endposition(x[m]/y[m])	Δx[mm]	Δy[mm]	$\Delta\phi$[deg]
1.1	2.978/1.485	23	15	1
1.2	2.980/1.475	36	19	1
1.3	2.969/1.460	36	38	1
2.1	2.998/0.985	2	20	0
2.2	3.010/0.950	7	48	3
2.3	3.005/0.950	6	49	2
3.1	3.000/1.460	2	42	2
3.2	2.985/1.487	15	13	2
3.3	3.005/1.490	11	31	1

Tabelle 7.2: *Vergleich der tatsächlichen mit den gemessenen Endpositionen*

position verbunden. Aus den drei Figuren lässt sich auch die jeweilige Geschwindigkeit des Roboters ableiten. Diese beträgt ungefähr 0.3m/sec bei den Fahrten 1 und 3, sowie 0.5m/sec bei Fahrt 2. Bei allen Fahrten haben wir die unten angegebenen Parametereinstellungen gewählt, die leicht von den in Kap. 5.7 verwendeten Werten abweichen.

$$
\begin{aligned}
\alpha &= 0.2 \\
C_0 &= \begin{bmatrix} (0.03m)^2 & 0 & 0 \\ 0 & (0.03m)^2 & 0 \\ 0 & 0 & (\frac{2}{57.3}rad)^2 \end{bmatrix} \\
\sigma_{US} &= 0.03m \text{ (Ecke,Kante) bzw.} 0.08m\text{(Wand)} \\
\sigma_{Kreisel} &= (\frac{5}{57.3}rad) \\
k_{xd}^2 &= 2.76 \text{ (entspricht 75\%)} \\
S_{Winkelkorrektur} &= 1.5m \\
\sigma_{Winkelkorrektur} &= (\frac{5}{57.3}rad)
\end{aligned}
$$

Die beiden ersten Fahrten zeigen ungefähr das gleiche Resultat, was die Abweichung der Abstandsmessungen von den Referenzobjekten betrifft. Bei der Achterkurve sind die Fehlerellipsen etwas grösser. Dies lässt sich mit der etwas grösseren Fahrgeschwindigkeit erklären. Auf Grund

```
typedef struct  {
  int time;                /* Aufnahmezeitpunkt der Daten  */
  int x, y, phi, psi;      /* Position des Roboters        */
  int a, b, c;             /* Positionen der drei Balken   */
  int searchPatternFlag;   /* Flags fuer die Steuerung     */
  int controlTowerFlag;    /* des Programmablaufs          */
  int findInitPosFlag;
  int findDockFlag;
  int newFU1PosFlag;
  int newPatternFlag;
}  *VISION_DATA_PTR;
```

Figur 7.14: *Struktur des Datenfeldes, auf welches VME-Rechner und Bildverarbeitungssystem gemeinsam zugreifen können.*

der Häufigkeit der Messungen kann man erkennen, welche Objekte für die Ultraschallerkennung günstig sind und deshalb häufig benutzt wurden. Bei der dritten Fahrt (Fig 7.13) ist die Positionsunsicherheit etwas grösser, weil die Winkelkorrekturmethode ungenauer ist als die Stützung mittels Kreisel.

Interessant sind auch die am Ende einer Fahrt festgestellten Abweichungen zwischen gemessener und tatsächlicher Position. In Tab. 7.2 sind die Resultate von mehreren Fahrten zusammengefasst. Es zeigt sich, dass die Fehler im Bereich der Initialisierungsfehler liegen. Bei allen von uns durchgeführten Fahrten haben wir am Schluss einen Positionsfehler gemessen, der kleiner als die geforderten 10cm ist.

7.7 Andocken

Bis jetzt wurden für die Positionsbestimmung nur Odometrie- und Ultraschalldaten verwendet. Beim Andocken setzen wir zusätzlich ein Visionsystem ein, mit welchem die Winkel zu einem dreiteiligen Balkenmuster bestimmt werden sollen. Dabei müssen wir die folgenden systembedingten Einschränkungen berücksichtigen:

- Ein Datenaustausch zwischen den beiden Rechnern ist nur mit

dem Visionsystem als Master und dem VME-Rechner als Slave möglich.

- Bildverarbeitung ist relativ zeitaufwendig, weshalb darauf zu achten ist, dass auf dem Visionsystem möglichst wenige zusätzliche Aufgaben erledigt werden müssen.

Die erste Einschränkung zwingt uns, einen für beide Rechner gemeinsamen Datenbereich auf dem VME-Rechner zu definieren (Fig. 7.14). Die Ablaufsteuerung erfolgt durch speziell dafür vorgesehene Flags.

7.7.1 Programmablauf

Nachdem der Roboter in die Nähe der Andockstelle gefahren ist, läuft der Andockvorgang nach dem in Fig. 7.15 dargestellten Prozess ab, der alle 20msec aufgerufen wird. Als erstes wird die Covarianzmatrix der Positionsfehler berechnet, welche beim Fahren durch die Odometriefehler entstehen. Danach wird das gemeinsame Datenfeld bei Bedarf mit den neuesten Positionswerten gefüllt. Zeigt die Kamera aus irgend einem Grunde noch nicht in die Richtung des Balkenmusters, so muss versucht werden, durch Drehen des Turmes dieses Muster zu finden. Da die Messgenauigkeit besser ist, wenn das Balkenmuster genau in der Mitte des Bildes liegt, wird der Turm entsprechend ausgeregelt. Als nächstes wird die Position des Roboters relativ zur Andockstation bestimmt. Um die Genauigkeit zu erhöhen, wird dazu der Durchschnitt aus 20 Messungen genommen. Damit sich diese Messungen unterscheiden, wird der Turm dabei ganz leicht gedreht. Zudem wird kontrolliert, ob das eingelesene Muster nicht schon einmal benutzt wurde. Erst jetzt beginnt der eigentliche Andockvorgang. Analog zum im vorherigen Kapitel beschriebenen Vorgehen wird versucht, die Positionsschätzung mittels Ultraschallmessungen zu verbessern. Dazu werden aber nur Abstandsmessungen in Richtung der Andockstation berücksichtigt. Dann wird kontrolliert, ob ein neues Balkenmuster eingelesen wurde. Falls ja, wird daraus die dazugehörige Kameraposition berechnet und die Roboterposition mittels Kalmanfilterung aufdatiert. Als letztes wird die neue Sollposition des Roboters berechnet, welche von der Position des Roboters relativ zur Andockstelle abhängt. Dabei muss darauf geachtet werden, dass der Roboter am Schluss senkrecht

```
* Positionsfehler-Covarianzmatrix berechnen *
if (newFU1PosFlag)  {
  * Roboterposition auf gemeinsames Datenfeld schreiben *
  newFU1PosFlag++;
}
if (searchPatternFlag)  * Turm drehen *
if (controlTowerFlag && newPatternFlag)  {
  * Turm ausrichten *
  if (Turm ausgerichtet)  {
    controlTowerFlag = 0; findInitPosFlag = 1;
  }
  newPatternFlag = 0; newFU1PosFlag = 1;
}
if (findInitPosFlag && newPatternFlag)  {
  * Turm ganz leicht drehen *
  * Balkenpositionen einlesen und summieren *
  iInit++;
  if (iInit > 20)  {
    * durchschnittliche Balkenpositionen berechnen *
    * Roboterposition berechnen *
    * Positionsfehler-Covarianzmatrix initialisieren *
    findInitPosFlag = 0;  findDockFlag = 1;
  }
  newPatternFlag = 0; newFU1PosFlag = 1;
}
if (findDockFlag) {
  * Ultraschalldaten einlesen *
  * Abstandsmessung Richtung Andockstelle kontrollieren *
  if (abs(dUS - dBerechnet) < 0.2)
    * Position aufdatieren *
}
if (findDockFlag && newPatternFlag)  {
  * Vision-Datensatz uebernehmen *
  * Roboterposition aus Balkenpositionen berechnen *
  * Roboterposition mittels Kalmanfilter aufdatieren *
  newPatternFlag = 0; newFU1PosFlag = 1;
if (findDockFlag) * Sollposition des Roboters berechnen *
}
```

Figur 7.15: *Pseudocode des Andockprozesses*

```
* Fenstergroesse definieren *
while (!ende) {
  * Bild aufnehmen *
  if ((newPatternFlag == 0) && (newFU1PosFlag > 1)) {
    newFU1PosFlag = 0;
    * Bild binarisieren *
    * Bild segmentieren *
    * Muster suchen *
    if (Muster gefunden)  {
      searchPatternFlag = 0;
      * Balkenpositionen uebertragen *
      newPatternFlag = 1;
    }
    else
      newFU1PosFlag = 1;
  }
}
```

Figur 7.16: *Pseudocode des Datenaufnahmeprozesses auf dem Visionsystem*

auf die Andockstelle zufährt. Wichtig ist auch, dass die Kamera in jeder Phase des Andockvorganges in Richtung des mittleren Balkens zeigt.

7.7.2 Aufnahme des Balkenmusters

Da das Visionsystem nicht mit dem VME-Rechner synchronisiert werden kann, wird für die Mustererkennung die in Fig. 7.16 dargestellte while-Schleife so oft wie möglich durchlaufen. Zuerst wird aber noch die Grösse des Bildausschnittes definiert, innerhalb welchem das Andockmuster gesucht werden soll. Damit kann der Zeitbedarf von Bildbinarisierung und -segmentierung beeinflusst werden.

Am Anfang der Schleife steht die Bildaufnahme. Das aufgenommene Bild wird aber nur dann weiterverarbeitet, wenn es kein früher erkanntes Muster gibt, welches noch nicht vom VME-Rechner übernommen wurde. Des weitern wird getestet, ob die Positionsangaben im gemein-

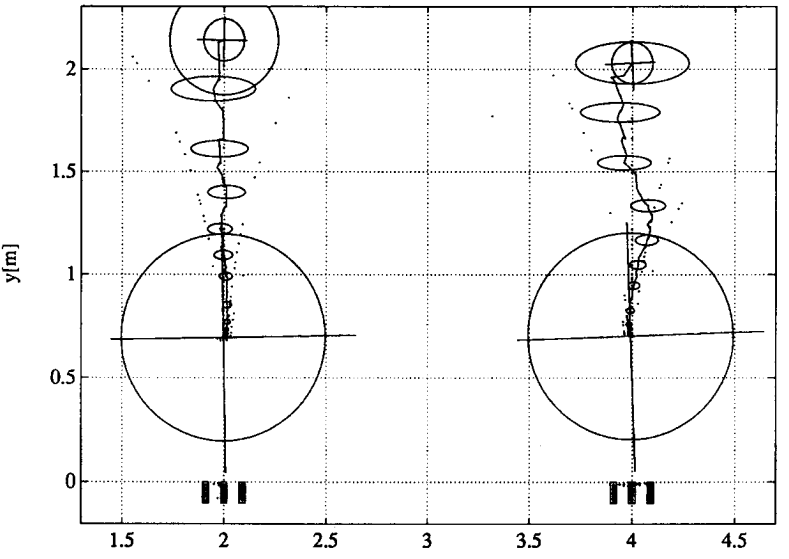
samen Datenfeld aktuell sind. Falls ja, werden diese Daten eingefroren, d.h. die Positionsangaben werden erst wieder aktualisiert, wenn das Muster nicht mehr benötigt wird. Dies ist besonders wichtig, weil das aufgenommene Muster und die geschätzte Roboterposition zeitlich übereinstimmen müssen. Danach wird das Bild analysiert und bei erfolgreicher Erkennung die Positionen der drei Balken auf den gemeinsamen Datenbereich geschrieben. Durch entsprechende Flags wird angezeigt, dass ein neues Muster aufgenommen worden ist. Falls kein Balkenmuster erkannt worden ist, wird das Positionsfeld wieder zum Überschreiben freigegeben.

7.7.3 Resultate

Zur Veranschaulichung der erreichten Resultate haben wir zwei Andockfahrten aufgezeichnet (Fig. 7.17). Beide Fahrten beginnen ca. 2m vor der Andockstelle. Die durch Mittelung von 20 Positionsmessungen errechnete Anfangsposition ist durch einen kleinen Roboter angedeutet. Die ausgezogene Linie entspricht dem Weg des Roboters zur Andockstelle. Die einzelnen Positionsmessungen beim Heranfahren sind durch Punkte dargestellt. Ebenfalls durch Punkte sind die Abstandsmessungen mittels Ultraschall angedeutet. Diese liegen alle in der Nähe der Andockstelle. Die Positionsunsicherheit ist durch Fehlerellipsen dargestellt. Die Wahrscheinlichkeit, dass die tatsächliche Position innerhalb dieser Ellipsen liegt, beträgt 50%. Der zeitliche Abstand der Ellipsen ist 1sec. Der grosse Roboter, der ca. 0.7m vor der Andockstelle liegt, markiert die erreichte Endposition.

Bei beiden Fahrten wurden die folgenden Parametereinstellungen gewählt:

$$
\begin{aligned}
\alpha &= 0.1 \\
C_0 &= \begin{bmatrix} (0.2m)^2 & 0 & 0 \\ 0 & (0.2m)^2 & 0 \\ 0 & 0 & (\frac{5}{57.3}rad)^2 \end{bmatrix} \\
\sigma_{x_{Vision}} &= 0.75 * d^3[m] \\
\sigma_{y_{Vision}} &= 0.75 * d^3[m] \\
\sigma_{\phi_{Vision}} &= 0.2 \\
\sigma_{d_{US}} &= 0.08m
\end{aligned}
$$



Figur 7.17: *Fahrt zur Andockstelle*

Anhand von Fig. 7.17 erkennt man den Einfluss der Pixelauflösung, welche eine Quantisierung der Winkelmessungen bewirkt. Die Verteilung der Positionsmessungen ist denn auch nicht gleichmässig, sondern verläuft ziemlich genau auf drei Bahnen, die vor der Andockstelle zusammenlaufen. Man sieht auch, dass die Kalmanfilterung eine Mittelung dieser Positionsmessung bewirkt, was eine erstaunlich gute Positionsschätzung erlaubt. Durch die Ultraschallabstandsmessungen kann die Positionsschätzung wirkungsvoll verbessert werden.

Die erreichte Endgenauigkeit lässt sich aus Fig. 7.17 leider nicht herauslesen. Mehrere Nachmessungen der Endposition haben aber durchwegs Positionsfehler von weniger als 1cm und Orientierungsfehler von weniger als 2° ergeben.

Leer - Vide - Empty

Kapitel 8

Schlussbemerkungen

Ziel dieser Arbeit war es, ein Verfahren zu entwickeln und zu implementieren, welches es einem mobilen Roboter erlaubt, jederzeit seine Position und seine Orientierung in einer strukturierten Umgebung zu bestimmen. Als Erweiterung gegenüber bekannten Methoden sollte dieses Verfahren auch ohne künstliche Orientierungshilfen und bei normaler Fahrgeschwindigkeit funktionieren. Das Hauptproblem lag deshalb darin, eine schnelle und sichere Positionsbestimmung zu ermöglichen, obwohl man sich nur auf eine relativ unsichere Objekterkennung abstützen konnte. Auf Grund der sich je nach Fahrsituation ändernden Anforderungen bezüglich Genauigkeit und verfügbarer Zeit wurde eine Dreiteilung dieser Aufgabe vorgenommen: Initialisierung - Aufdatierung beim Fahren - Andocken.

Die Initialisierung, d.h. die Bestimmung der Anfangsposition, läuft in zwei Phasen ab: 1. Datenaufnahme und Bildung des *Sensorbasierten Modelles* - 2. Vergleich mit dem *Referenzmodell.* Das Ergebnis des Modellbildungsprozesses hängt wesentlich vom eingesetzten Sensorsystem und von der Umgebung ab. Wir verwenden Ultraschallabstandsmessungen. Daraus können mit dem von uns definierten Erkennungsalgorithmus in den meisten Fällen genügend Objekte erkannt werden. Dies ist vor allem auf die guten Eigenschaften des Modellvergleiches mittels der von uns eingeführten Clusteranalyse zurückzuführen. Diese Methode ist bezüglich Robustheit gegenüber falsch erkannten Objek-

ten und bezüglich Geschwindigkeit allen andern bekannten Methoden vorzuziehen.

Für die Positionsaufdatierung beim Fahren wird ein Kalmanfilter eingesetzt. Zu lösende Probleme sind dabei die Schätzung der odometrischen Positionsunsicherheit, das Finden von für die Aufdatierung brauchbaren Messungen, sowie die Entkoppelung von Positions- und Orientierungsschätzung. Was die Aufdatierung überhaupt zum Problem macht, ist die Tatsache, dass jede Messung daraufhin untersucht werden muss, ob es sich um den Abstand zu einem Referenzobjekt oder zu einem Hindernis handelt. Einziges Unterscheidungsmerkmal ist die Differenz zwischen gemessenem und erwartetem Abstand. Dies funktioniert nur, wenn der Positionsfehler nicht zu gross wird. Dies wiederum kann nur garantiert werden, wenn die odometrische Positionsbestimmung genügend genau ist, oder wenn genügend stützende Abstandsmessungen gemacht werden können. Die Schätzung der Orientierung ist problemlos, wenn ein Kreisel eingesetzt werden kann. Muss darauf verzichtet werden, so kann mit der in Kapitel 5.6 beschriebenen Methode die Orientierung aus dem Verlauf der Positionskorrekturen in Funktion des gefahrenen Weges berechnet werden. Im Vergleich mit dem Kreisel erhöhen sich dadurch die Anforderungen an die Genauigkeit der Positionsbestimmung etwas.

Vor einer Andockstelle muss die Position mit grosser Sicherheit bestimmt werden können. Dies kann nur erreicht werden, wenn ein eindeutiges Orientierungsmittel, wie z.B. ein Andockmuster, vorhanden ist. Wir ergänzten deshalb die Abstandssensoren mit einem Bildverarbeitungssystem, mit welchem das Andockmuster erkannt werden kann, und welches zusätzliche Winkelmessungen liefert. Die Positionsschätzung wird wiederum mit einem Extended Kalman Filter durchgeführt. Die gestellten Genauigkeitsanforderungen können damit erfüllt werden.

Einen wichtigen Teil der Arbeit bildete die Implementation der oben erwähnten Verfahren. Ihre Einbettung in ein Gesamtsystem bedeutet eine wesentliche Erweiterung zu bisher bekannten Verfahren.

Für die praktische Verwendbarkeit der Positionsbestimmung sind deren Robustheit und deren Zeitbedarf von entscheidender Bedeutung. Ein weiteres wichtiges Kriterium ist, wie leicht sich die Positionsbestimmung an verschiedene Umgebungen anpassen lässt.

Die Bestimmung der Anfangsposition ist mit unserer Methode schneller und robuster als mit allen anderen bekannten Methoden. Auch die Genauigkeit ist ausreichend. Für den praktischen Einsatz würde man sich vielleicht einen Zeitbedarf von weniger als 10 Sekunden wünschen. Die Hälfte dieser Zeit muss dabei für die Abstandsmessungen mittels Ultraschall aufgewendet werden und kann aus physikalischen Gründen nicht reduziert werden. Eine wesentliche Verringerung des Zeitbedarfs ist deshalb nur mit schnelleren Sensoren, z.B. mit Laserdistanzmessern, möglich. Die Anpassung an verschiedene Umgebungen geschieht softwaremässig, d.h. ohne Veränderungen der neuen Umgebung. Die einzige Arbeit ist die Eingabe des neuen Referenzmodelles. Wenn der Roboter häufig seine Umgebung wechseln muss, wäre es sinnvoll, dies zu automatisieren.
Die Aufdatierung der Position anhand von Ultraschallmessungen funktioniert gut, auch wenn der Roboter mit normaler Geschwindigkeit fährt. Will man auf den Einsatz eines Kreisels verzichten, so erweist sich unser Verfahren für die Orientierungskorrektur als sehr nützlich. In Ausnahmesituationen kann immer wieder auf das Verfahren zur Bestimmung der Anfangsposition zurückgegriffen werden.
Relativ gross ist der Sensoraufwand beim Andocken. Das dafür benutzte Bildverarbeitungssystem sollte deshalb möglichst auch für andere Aufgaben eingesetzt werden können.

Die in dieser Arbeit vorgestellten Verfahren können ohne grossen Aufwand auch auf mobile Roboter mit einer anderen Kinematik und mit anderen Sensorsystemen angepasst werden. Entscheidend ist dabei, dass genügend Referenzobjekte erkannt werden können, und dass die odometrische Positionsunsicherheit geschätzt werden kann.

Mobile Roboter werden sich in Richtung grösserer Autonomie entwickeln. Für die Positionsbestimmung heisst dies, dass der Anwendungsbereich auf kompliziertere Umgebungen erweitert werden muss. Dies impliziert fähigere Sensorsysteme, sprich Bildverarbeitung. Kompliziertere Umgebungen bedeuten aber auch umfangreichere Referenzmodelle, was wiederum den Wunsch nach automatischer Generierung desselben nach sich zieht. Des weiteren gilt es aber auch zu berücksichtigen, dass grössere Bewegungsfreiheit nur erreicht werden kann, wenn die Fortbewegungsmechanik des mobilen Roboters entsprechend angepasst wird.

Leer - Vide - Empty

Literaturverzeichnis

[Albatros Ultrasonic Network] *Albatros - Local Area Ultrasonic Network User's Manual*, ROBOSOFT, Asnière, France, 1988

[Albatros Manuals] *Albatros - User's Manual, Albatros - C Interface Reference Manual, Albatros - Reference Manual*, ROBOSOFT, Asnière, France, 1989

[Ayache and Faugeras 88] N.Ayache, O.Faugeras: *Maintaining Representations of the Environment of a Mobile Robot*, Rapport de Recherche No.789, INRIA, France, Fevrier 1988, pp.1..37

[Badreddin 88] E.Badreddin: *RAMSIS Concept and Specifications*, Proc. 19th Int. Symp. on Allied Techn. & Automation (ISATA), Monte Carlo 1988, pp.103..121

[Badreddin 89b] E.Badreddin: *Recursive Nested Behavior Control Structure for Mobile Robots*, Int. Conf. on Intelligent Autonomous Systems, Amsterdam, The Netherlands, Dec. 1989, pp.586..596

[Badreddin 91] E.Badreddin: *Tailoring the Behavior of a Mobile Robot*, Proceedings, IEEE 1991 Int. Conf. on Robotics and Automation, Sacramento, April 1991, pp.2556..2561

[Baird 84] Henry S. Baird: *Model-based image matching using location*, (ACM distinguished dissertations, 1984) Diss., Univ.Princeton - N.J.1984 Cambridge - Mass. : MIT Press, 1985

[Boegli 85] P.Boegli: *A comparative evaluation of AGV navigation techniques*, Proc. of the 3rd Int. Conf. on Automated Guided Vehicle Systems, Stockholm, Sweden, Oct. 1985, pp.169..179

[Bronstein 81] I.N.Bronstein, K.A.Semendjajew: *Taschenbuch der Mathematik*, Verlag Harri Deutsch, Thun und Frankfurt/Main, 20.Auflage, 1981

[Brooks 85] R.Brooks: *Visual Map Making for a Mobile Robot*, Proceedings, IEEE 1985 Int. Conf. on Robotics and Automation, St.Louis, March 1985, pp.824..829

[Brown 85] M. K. Brown: *Locating Object Surfaces with an Ultrasonic Range Sensor*, Proceedings, IEEE 1985 Int. Conf. on Robotics and Automation, St.Louis, March 1985, pp.110..115

[Cassinis and Venuti 89] R.Cassinis, P.Venuti: *Sonar Range Data Processing and Enhancement*, Int. Conf. on Intelligent Autonomous Systems, Amsterdam, Dec. 1989, pp.168..177

[Chatila 81] R.Chatila: *Systeme de navigation pour un robot mobile autonome: modélisation et processus decisionnels*, Thèse de Docteur Ingénieur, Toulouse, July 1981

[Chatila and Laumond 85] R.Chatila, J.-P. Laumond: *Position Referencing and Consistent World Moldeling for Mobile Robots*, Proceedings, IEEE 1985 Int. Conf. on Robotics and Automation, St.Louis, March 1985, pp.138..145

[Crowley 87] James L. Crowley: *Using the Composite Surface Model for Perceptual Tasks*, Proceedings, IEEE 1987 Int. Conf. on Robotics and Automation, Raleigh, NC, March 1987, pp.929..934

[Crowley 89] James L. Crowley: *World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging*, Proceedings, IEEE 1989 Int. Conf. on Robotics and Automation, Scottsdale, May 1989, pp.674..680

[Darnell and Margolis 88] P.Darnell, Ph.Margolis: *Software Engineering in C*, Springer-Verlag, New York 1988

[Drumheller 87] Michael Drumheller: *Mobile Robot Localization Using Sonar*, IEEE 1987 Transactions on Pattern Analysis and Machine Intelligence, March 1987, pp.325..331

[DurrantWhyte 88] Hugh F. DurrantWhyte: *Integration, Coordination, and control of multi-sensor robot systems*, Kluwer Academic Publishers, 1988

[Elfes 86] Alberto Elfes: *A Sonar-Based Mapping and Navigation System*, Proceedings, IEEE 1986 Int. Conf. on Robotics and Automation, San Francisco, April 1986, pp.1151..1156

[Elfes and Matthies 88] A.Elfes, L.Matthies: *Integration of Sonar and Stereo Range Data Using a Grid-Based Representation*, Proceedings, IEEE 1988 Int. Conf. on Robotics and Automation, Philadelphia, April 1988, pp.727..733

[Engel&Stiefvater 89] *E&S Vision Workstation - Benutzerhandbuch*, Gesellschaft für Bildverarbeitung und flexible Automatisierung, Karlsruhe, Deutschland, 1989

[Everitt 80] Brian Everitt: *Cluster Analysis*, Halsted Press, New York, 1980

[Everett 89] H.R.Everett: *Survey of Collision Avoidance and Ranging Sensors for Mobile Robots*, robotics and autonomous systems, north-holland, amsterdam, Vol. 5, No. 1, May 1989, pp.5..67

[Gascoigne 85] A.Gascoigne: *Mobile Robots for Industrial and Commercial Applications - A Review of Possibilities and Design Requirements*, IEEE Trans. on Mechanical Engineering, 1985

[Gonzalez 87] R.Gonzalez: *Digital Image Processing*, Addison-Wesley Publishing Company, 1987

[Gelb 84] A.Gelb: *Applied optimal estimation*, Cambridge, MIT Press, 1984

[Haberaecker 85] Peter Haberaecker: *Digitale Bildverarbeitung: Grundlagen und Anwendungen*, Hanser-Verlag, München Wien, 1985

[Heiz 83] Ulrich Heiz: *Einrichtung für ortsbewegliche, ebengebundene Objekte zur Selbstbestimmung ihrer Lagekoordinaten und Richtungswinkel*, Swiss Patent No. EP 84 110 861.6, Oktober 1983

[Holenstein and Isenrich 84] A.Holenstein, R.Isenrich: *Fahrbarer Manipulator*, Institut für Automatik und Ind. Elektronik, ETH Zürich, Diplomarbeit AIE 8527, 1984

[Holenstein 88a] A.Holenstein: *AIE-Quattro-II: Ein Laborfahrzeug für die Forschung auf dem Gebiet der mobilen Roboter*, Institut für Automatik und Ind. Elektronik, ETH Zürich, Internal Report No.88-01, 1988

[Holenstein 88b] A.Holenstein: *Fahr-Software für einen mobilen Roboter*, SGA-Zeitschrift, Vol.8, No.4,1988, pp.3..13

[Holenstein 89] A.Holenstein: *RAMSIS - Software-Beschreibung*, Institut für Automatik und Ind. Elektronik, ETH Zürich, Interner Bericht, 1989

[Holenstein and Badreddin 91] A.Holenstein, E.Badreddin: *Collision Avoidance in a Behavior-based Mobile Robot Design*, Proceedings, IEEE 1991 Int. Conf. on Robotics and Automation, Sacramento, April 1991, pp.898..903

[Holenstein 92] A.Holenstein, M.Müller, E.Badreddin: *Mobile Robot Localization in an Environment Cluttered with Obstacles*, Proceedings, IEEE 1992 Int. Conf. on Robotics and Automation, Nizza, Mai 1992, pp.2576..2581

[Hostetler 83] Larry D. Hostetler, Ronald D. Andreas: *Nonlinear Kalman Filtering Techniques for Terrain Navigation*, Trans. on Automatic Control, Vol. AC-28, No.3, March 1983, pp.315..323

[Jorgensen et. al 86] Ch.Jorgensen, W.Hamel, Ch.Weisbin: *Autonomous Robot Navigation*, Oak Ridge National Laboratory, BYTE, Jan. 1986, pp.223..235

[Keller and Tarreghetta 86] P.Keller, S.Tarreghetta: *Andocken eines Wagens*, Institut für Automatik und Ind. Elektronik, ETH Zürich, Semesterarbeit AIE 8558, 1986

[Köbbing 89] H.Köbbing: *Völlig autonome Flurförderzeuge: Berührungslose Geschwindigkeitsbestimmung*, Elektronik 7, 31.3.1989, pp.46..51

[Kriegman 87] D.Kriegman, E.Triendl, Th.Binford: *A Mobile Robot: Sensing, Planning and Locomotion*, Proceedings, IEEE 1987 Int. Conf. on Robotics and Automation, Raleigh, March 1987, pp.402..408

[Krotkov 89] Eric Krotkov: *Mobile Robot Localization Using a Single Image*, Proceedings, IEEE 1989 Int. Conf. on Robotics and Automation, Scottsdale, May 1989, pp.978..983

[Kuc and Siegel 87] R.Kuc, M.Siegel: *Efficient Representation of Reflecting Structures for a Sonar Navigation Model*, Proceedings,

IEEE 1987 Int. Conf. on Robotics and Automation, Raleigh, March 1987, pp.1916..1923

[Kuc 87] Roman Kuc: *Physically Based Simulation Model for Accoustic Sensor Robot Navigation*, IEEE 1987 Transactions on Pattern Analysis and Machine Intelligence, Nov. 1987, pp.766..778

[Larcombe 79] M.H.E.Larcombe: *Mobile Robots for Industrial Use*, University of Warwick, U.K., The Industrial Robot, June 1979, pp.70..76

[Leonard and DurrantWhyte 91] John J.Leonard, Hugh F.Durrant-Whyte: *Mobile Robot Localization by Tracking Geometric Beacons*,IEEE Trans. on Robotics and Automation, Vol.7, Nr.3, Juni 1991

[Leu 89] M.Leu: *Kollisionsverhinderung mit Hilfe von Repulsionsfunktionen*, Institut für Automatik und Ind. Elektronik, ETH Zürich, Diplomarbeit AIE 8622, 1989

[Lee and Preparata 84] D.Lee, F.Preparata: *Computational Geometry - A Survey*, IEEE 1984 Trans. on Computers, Vol. C-33, No.12, Dec. 1984, pp.1072..1101

[Lovasz and Plummer 86] L.Lovasz, M.Plummer: *Matching Theory*, North-Holland, Amsterdam, 1986

[Marce and Julliere 86] L.Marce, M.Julliere: *Dynamic Localization of a Mobile Robot Through Range Measurements*, Proc. of SPIE, Vol. 727, Mobile Robots, Cambridge, Mass., Oct. 1986, pp.274..281

[Microbe 85] F.Freyberger, P.Kampmann, G.Karl, G.Schmidt: *Microbe - ein autonomes mobiles Robotersystem*, TU München, VDI-Zeitung, April 1985, S.231..236

[Miller 84] David Miller: *Two Dimensional Mobile Robot Positioning Using Onboard Sonar*, Proceedings, IEEE 1984 Int. Conf. on Robotics and Automation, Atlanta, March 1984, pp.362..369

[Miller 85] David Miller: *A Spatial Representation System for Mobile Robots*, Proceedings, IEEE 1985 Int. Conf. on Robotics and Automation, St.Louis, March 1985, pp.122..127

[Miller 88] Richard K. Miller: *Mobile Robots*, Emerging Industries No.1, Technical Insights, Inc., Englewood/Fort Lee, NJ, 1988

[Moravec 83] Hans P. Moravec: *The Stanford Cart and the CMU Rover*, Proceedings of the IEEE, Vol.71. No.7, July 1983, pp.872..884

[Moravec and Elfes 85] Hans P. Moravec, Alberto Elfes: *High Resolution Maps from Wide Angle Sonar*, Proceedings, IEEE 1985 Int. Conf. on Robotics and Automation, St.Louis, March 1985, pp.116..121

[Moravec et. al 85] C.Thorpe, L.Matthies, H.Moravec: *Experiments and Thoughts on Visual Navigation*, Proceedings, IEEE 1985 Int. Conf. on Robotics and Automation, St.Louis, March 1985, pp.830..835

[Moravec 88] Hans P.Moravec: *Sensor Fusion in Certainty Grids for Mobile Robots*, AAAI (American Association for Artificial Intelligence), AI Magazine, Summer 1988, pp.61..74

[Mortenson 85] Michael E. Mortenson: *Geometric Modeling*, John Wiley and Sons, Inc., New York, 1985

[Moutarlier and Chatila 89] Ph. Moutarlier, R. Chatila: *An Experimental System for Incremental Environement Modelling by an Autonomous Mobile Robot*, Experimental Robotics I (The First International Symposium, Montreal, 1989), Springer Verlag 1990, pp.327..346

[MPW C] *Macintosh Programmers Workshop C*, Apple Technical Publications, Apple Computer Inc., CA, 1987

[MPW Reference] *Macintosh Programmers Workshop Reference*, Apple Technical Publications, Apple Computer Inc., CA, 1987

[Papadimitriou and Steiglitz 82] Ch.Papadimitriou, K.Steiglitz: *Combinatorial Optimization - Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1982

[Pavlidis 73] Theodosios Pavlidis: *Waveform Segmentation Through Functional Approximation*, IEEE 1973 Transactions on Computers, Vol. C-22, No.7, July 1973, pp.689..697

[Pavlidis 74] Theodosios Pavlidis: *Segmentation of Plane Curves*, IEEE 1974 Transactions on Computers, Vol. C-23, No.8, Aug. 1974, pp.860..869

[Preparata and Shamos 85] F.Preparata, M.Shamos: *Computational Geometry - An Introduction*, Springer Verlag, New York Berlin Heidelberg Tokyo, 1985

[Press et. al 88] W.Press, B.Flannery, S.Teukolsky, W.Vetterling: *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge 1988

[Reingold et. al 77] E.Reingold, J.Nievergelt, N.Deo: *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1977

[Romesburg 84] H. Charles Romesburg: *Cluster Analysis for Researchers*, Lifetime Learning Publications, Belmont, California, 1984

[Scott 87] A.W.Scott: *The Sonar Ring: Obstacle Detection for a Mobile Robot*, Proceedings, IEEE 1987 Int. Conf. on Robotics and Automation, Raleigh, March 1987, pp.1574..1579

[Sedgewick 88] Robert Sedgewick: *Algorithms*, Princeton University, Addison-Wesley Publishing Company, 1988

[Severin and Klein 88] D.Severin, W.Klein: *Ein System zur leitlinienlosen Führung von Industriefahrzeugen*, TU Berlin, Automatisierungstechnische Praxis, Heft 3/1988, S.122..128

[Smith and Cheeseman 86] Randall C. Smith, Peter Cheeseman: *On the Representation and Estimation of Spatial Uncertainty*, The International Journal of Robotics Research, Vol.5, No.4, Winter 1986, Massachusetts Institute of Technology, pp.56..68

[Soetadji and Rembold 87] T.Soetadji, U.Rembold: *A Cube-Based Approach to Find-Path for an Autonomous Mobile Robot*, NATO ASI Series, Vol F-29, Springer-Verlag Berlin Heidelberg 1987, S.557..588

[Sorenson 66] H.W.Sorenson: *Kalman Filtering Techniques*, Advances in Control Systems Theory and Applications, Vol. 3, 1966, Academic Press Inc., pp.219..292

[Späth 83] Helmut Späth: *Cluster-Formation und -Analyse*, R.Oldenburg Verlag, München Wien, 1983

[Steer 89] Barry Steer: *Experiments and theory with a 0.5 ton mobile robot*, Experimental Robotics I (The First International Symposium, Montreal, 1989), Springer Verlag 1990, pp.362..387

[Steinhagen und Fuchs 76] H.E.Steinhagen, S.Fuchs: *Objekterkennung*, VEB Verlag Technik, Berlin, 1976

[Sugihara 87] K.Sugihara: *Location of a Robot Using Sparse Visual Information*, The MIT Press Series in Artificial Intelligence, Fourth Int. Symp. on Robotics Research, Santa Cruz, Aug. 1987, pp.319..326

[Teldix 79] *TELDIX - Taschenbuch der Navigation*, Teldix GmbH, Heidelberg, 1979

[Tsumura et al. 83] T.Tsumura, N.Fujiwara, M Hashimoto: *An Experimental System for Self-Contained Position and Heading Measurement of Ground Vehicle*, Proceedings, 83' Int. Conf. on Advanced Robotics, Tokyo, Japan, May 1983, pp.269..276

[Tsumura 85] T.Tsumura: *New Method for Position and Heading Compensation on Ground Vehicle*, Proceedings, 85' Int. Conf. on Advanced Robotics, Tokyo, Japan, Sept. 1985, pp.429..436

[Tsumura 86a] T. Tsumura: *Survey of Automated Guided Vehicle in Japanese Factory*, Proceedings, IEEE 1986 Int. Conf. on Robotics and Automation, San Francisco, April 1986, pp.1329..1334

[Tsumura 86b] T. Tsumura: *Positioning and Guidance of Ground Vehicle by Use of Laser and Corner Cube*, Proceedings, IEEE 1986 Int. Conf. on Robotics and Automation, San Francisco, April 1986, pp.1335..1342

[VME Specs] *The VMEbus SPECIFICATION*, VMEbus International Trade Association (VITA), PRINTEX Publishing Inc., Tempe, Arizona, 1985

[Wang 88] C. M. Wang: *Location Estimation and Uncertainty Analysis for Mobile Robots*, Proceedings, IEEE 1988 Int. Conf. on Robotics and Automation, Philadelphia, April 1988, pp.1230..1235

[Wiklund 89] A.Wernersson, U.Wiklund, U.Andersson, K.Hyyppä: *Vehicle Navigation using "image information": on Association Errors*, Intelligent Autonomous Systems, Amsterdam, Dec 1989, pp.814..822

Lebenslauf

Ich wurde am 3.Oktober 1960 in Niederuzwil geboren und wuchs in Zuckenriet, Kanton St.Gallen, auf. Dort besuchte ich 6 Jahre die Primarschule und 2 Jahre die Sekundarschule. Anschliessend besuchte ich von 1975-1979 die Kantonsschule St.Gallen, wo ich mit der Maturitätsprüfung Typus C abschloss. Ab Herbst 1980 studierte ich an der Abteilung für Elektrotechnik an der Eidgenössischen Technischen Hochschule Zürich und erlangte Ende 1984 das Diplom eines Elektroingenieurs.

Nach dem Studium begann ich als Assistent und wissenschaftlicher Mitarbeiter am Institut für Automatik und Industrielle Elektronik der ETH Zürich. Nebst der Mitarbeit im Unterricht befasste ich mich vor allem mit mechatronischen Systemen. Dort war ich massgeblich am Bau von drei verschiedenen Prototypen von mobilen Robotern beteiligt. Meine Nachdiplomarbeit beschreibt den Aufbau und die Steuerung von AIE-Quattro-II, dem zweiten Prototypen. Mit dem Bau von RAMSIS, einem mobilen Roboter im Industriemassstab, begannen wir 1988. Dort bin ich vor allem für die Steuerungssoftware zuständig. Als Teil dieser Aufgabe entstand die vorliegende Dissertation.

Von 1987-89 arbeitete ich als nebenamtlicher Dozent für Regelungstechnik am Abendtechnikum in Zürich.