Driftkompensation auf Basis von Besetztheitsgittern

– Diplom-Arbeit –

am Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel
Lehrstuhl für kognitive Systeme
Prof. Dr. Ing. Gerald Sommer

vorgelegt von

Timm Haase

Betreuer: Dipl. Inform. Jörg Bruske

25. März 1998

Timm Haase:
Driftkompensation auf Basis von Besetztheitsgittern

Institut für Informatik
Technische Fakultät
Christian-Albrechts-Universität zu Kiel

Inhaltsverzeichnis

Abbildungsverzeichnis

Teil I

Einleitung und Grundlagen

Kapitel 1

Einleitung

Es ist der ewige Traum der Menschheit, Geräte zu besitzen, die selbständig schwere Arbeiten übernehmen können. Der heute dafür übliche Begriff *Roboter* leitet sich daher von dem slavischen Verb *robota* ab und bedeutet *"schwer arbeiten"*.

Die fortschreitende Automatisierung hat eine ganze Reihe von Industrierobotern hervorgebracht. Diese können ohne weitere Hilfe fest vorgegebene Aufgaben erledigen. In der Regel sind sie allerdings noch nicht in der Lage, eigenständige Entscheidungen zu treffen oder auf mehr als triviale Ereignisse zu reagieren. Autonome Roboter hingegen können sich veränderten Gegebenheiten anpassen, da sie auf Basis eingehender Informationen selbständig neue Lösungs-Strategien entwickeln. Damit dies zu einer sinnvollen Tätigkeit führt, muß das System über eine breite Informationsbasis verfügen. Es sind dazu eine Reihe von Problemstellungen zu lösen.

Der Schwerpunkt dieser Arbeit liegt auf der Driftkompensation und der Selbstlokalisation, die im Bereich der autonomen mobilen Roboter, neben der Navigation, die zentrale Aufgabenstellung wiederspiegelt. Während die Navigation die Planungsebene darstellt, sorgt die Lokalisation für die geeignete Datenbasis. In erstgenannter werden sowohl kostengünstigste[1] Wege für die Zwischenziele geplant als auch für eine elementare Hindernis-Vermeidung gesorgt. Der Roboter muß zu jedem Zeitpunkt der Planungsphase seine Position wissen, um einen optimalen Pfad generieren zu können. Die Lokalisation wiederum spaltet sich in die beiden Teile der globalen Lokalisation und der Driftkompensation. Letztere geht von einer bekannten Startposition aus, während erstere darauf verzichtet und ohne jegliches Vorwissen die Position ermittelt. Da im allgemeinen die Startposition des Roboters bekannt ist, beschränkt sich diese Arbeit auf das Problem der Driftkompensation, die beispielsweise durch ein geeignetes Fehlermodell gelöst werden kann. Es wird eine Referenzstrecke festgelegt, die der Roboter hin und zurück fährt. Die gemessene Abweichung dient als Parameter für das Modell und

[1] kostengünstig bezüglich einer der Aufgabe angemessenen Kostenfunktion

gilt als repräsentativ für die Bodenbeschaffenheit. In der anschließenden Anwendungsphase werden die von den internen Sensoren des Roboters aufgenommenen Daten, die Odometrie, gemäß des Fehlermodells korrigiert. Ein Problem dieses Ansatzes liegt in der "Blindheit" des Verfahrens begründet. Es werden keine weiteren Sensorinformationen zur Driftbestimmung aufgenommen. Ist also das anfangs festgelegte Fehlermodell nur leicht abweichend von der Realität, kommt es bei Langstreckenfahrten zu starken Verschiebungen zwischen der korrigierten und der realen Position des Roboters. Außerdem ist die Wahl der Referenzstrecke nicht einfach. Sie muß zum einen Rotation und Verschiebung bezüglich der x- und y-Komponente berücksichtigen und zum anderen ausreichend lang sein.

Es bietet sich deshalb die Einbeziehung der jeweiligen Umgebungsdaten an. Die Datenerhebung kann dabei durch die verschiedensten Sensoren stattfinden. In der Regel wird allerdings auf Ultra-Schall-Sensoren bzw. Laser-Entfernungsmesser zurückgegriffen. Die so erhobenen Daten werden mit einem Modell der Umwelt verglichen. Dieses Umweltmodell kann beispielsweise als CAD-Zeichnung vorliegen.

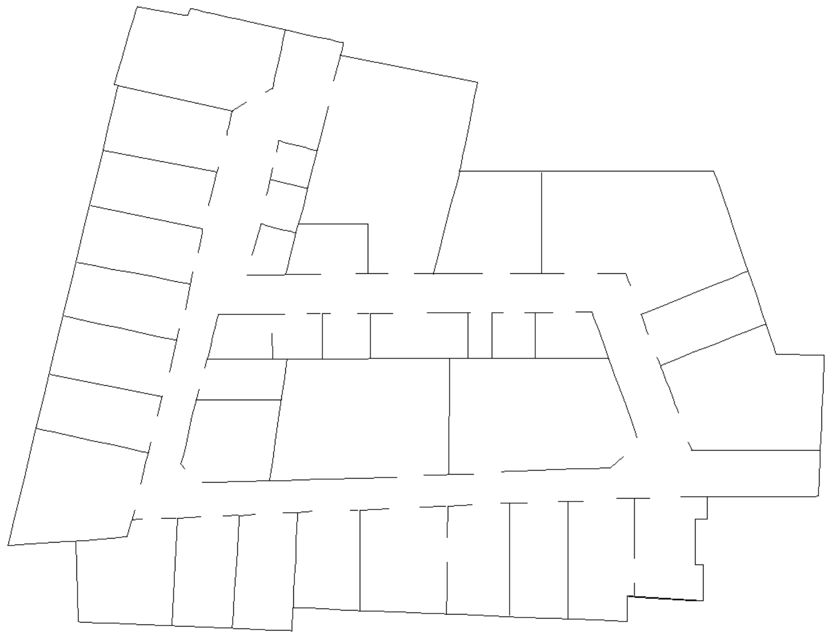

Abbildung 1.1: Die globale Karte als CAD-Modell

Abbildung 1.1 zeigt die linke Hälfte des dritten Stocks des Instituts für Informatik, Haus II der Christian-Albrechts-Universität, als CAD-Karte. Wie in dieser Abbildung zu sehen ist, sind keine Möbel in der Karte verzeichnet. Diese gelten als bewegliche Hindernisse und stören die vorgestellten Verfahren nicht. Bei den Ansätzen zur Driftkompensation wird, soweit möglich, auf schon etablierte Verfahren zurückgegriffen. Wo dies, z.B. durch einen zu theoretischen Charakter[2] oder einen fehlerhaften Ansatz, wie z.B. [Cox, Kruskal 1988], nicht möglich ist, werden die Verfahren entsprechend angepaßt bzw. nur der Grundgedanke fortgeführt. Alle genauer vorgestellten Verfahren zur Driftkompensation werden auf einem Roboter des Typs *B21* der Firma *Real World Interface* implementiert und ausführlich getestet, um den direkten Vergleich zu schon existierenden Verfahren herstellen zu können.

[2]Die Verfahren sind in der gegebenen Form nicht praxistauglich.

Kapitel 2

Der Forschungsstand

Im folgenden werden unterschiedliche Ansätze zur Lokalisation bzw. Driftkompensation mit deren Vor- und Nachteilen kurz vorgestellt. Diese zeigen exemplarisch die Hauptstrategien zur Lösung des Problems und basieren im wesentlichen auf den folgenden Daten:

- Topologische Karten
- Sektorkarten
- Positions-Wahrscheinlichkeitsgitter
- Objekterkennung
- Kalmanfilter
- Sensor-Korrelation
- Geradenbasierte Karten
- Hough-Transformation

Die beiden letzten Ansätze werden in dieser Arbeit ausführlicher behandelt und daher erst in den nachfolgenden Kapiteln beschrieben. Ein detaillierter Vergleich verschiedener Driftkompensations-Verfahren, der sowohl auf die geradenbasierte als auch auf die Hough-basierte Lokalisation eingeht, ist in [Schiele, Crowley 1994] zu finden.

2.1 Topologische Lokalisation

Der Ansatz von [Zimmer 1995] erstellt eine globale Karte mit Hilfe von sogenannten "Situationen". Eine Situation beinhaltet die Sensorwerte aller Sensoren und die dazugehörige Position. Der Aufbau der globalen Karte erfolgt dynamisch, d.h.

es wird mit einer Situation begonnen (der Startsituation) und jede weitere wird auf ihren Abstand zu den vorherigen überprüft. Übersteigt dieser einen Schwellwert, so wird die Situation zur Karte hinzugenommen und Verbindungen zu anderen Situationen, die zu dieser einen vorgegebenen topologischen Mindestabstand haben, hergestellt. Situationen bzw. Cluster von Situationen, die keine Verbindungen besitzen, werden wieder aus der Karte entfernt. Der Roboter kann sich in dieser Karte selbst lokalisieren, indem er die Situation (in der Karte) mit dem kleinsten Abstand zu seiner aktuellen Position findet.

Vorteile:

- Geringe Anforderungen an Art und Qualität der genutzten Sensoren
- Geringer Rechenaufwand (Echtzeit-Anforderungen)
- Sensorfusion einfach realisierbar
- Einfache Implementation möglich

Nachteile:

- Kein genaues Modell der Umwelt vorhanden
- Modell muß erst in einer Explorationsphase aufgebaut werden
- Nur topologische und keine geometrischen Daten vorhanden
- Stark abhängig von der Odometrie
- Topologische Verbindungen unter Umständen falsch

2.2 Umgebungseinteilung in Sektoren

In [Edlinger, Weiß 1995] wird die unmittelbare Umgebung des Roboters in Sektoren eingeteilt[1], in denen der jeweils kleinste Abstand zu einem Hindernis gespeichert ist. Falls die lokale Karte (die Sektorkarte) außerhalb des bekannten Bereiches liegt, wird sie der globalen Karte hinzugefügt. Die Zentren der einzelnen lokalen Karten innerhalb der globalen Karte bilden eine topologische Darstellung der Umwelt, d.h. die globale Karte kann auch als ungerichteter Graph dargestellt werden[2]. Selbstlokalisation erfolgt über Referenzpunkte, d.h. es wird der Fehler der Position und Orientierung bzgl. eines solchen Punktes berechnet. Die Referenzpunkte werden einer Karte entnommen (beispielsweise werden die

[1] Im Beispiel von *MOBOT IV* 60 Sektoren

[2] Zentren der lokalen Karten entsprechen den Knoten und die Verbindungen zwischen ihnen den Kanten.

Sensorwerte eines Laser-Rangefinders als Referenz zur Erzeugung einer solchen Karte verwendet).

Vorteile:

- Interne Karte ist vollständig und konsistent
- Navigation auf interner Karte möglich
- Algorithmen entsprechen Echtzeit-Anforderungen

Nachteile:

- Problem dynamischer Umgebungen (Menschen) nicht gelöst
- Genauer Sensor als Referenz notwendig

2.3 Positions-Wahrscheinlichkeits-Gitter

Gitter, die mit den Besetztheitswahrscheinlichkeiten belegt sind, bilden die Basis von [Burgard et al. 1996]. Es wird darüber hinaus eine Positions-Wahrscheinlichkeit für jede Zelle des Gitters berechnet. Durch die Integration über die Zeit werden die Zelleneinträge (Besetztheits- bzw. Positions-Wahrscheinlichkeiten) immer genauer, d.h. es bildet sich nach einer bestimmten Zeit die tatsächliche Position heraus. Zu jedem Zeitpunkt stellt das Feld mit der größten Wahrscheinlichkeit die beste Schätzung der Position dar.

Vorteile:

- Sehr genaue Bestimmung der Position (wenige Zentimeter Abweichung)
- Sensorfusion möglich
- Integration der Sensordaten über die Zeit
- Geringe Anforderungen an die Qualität der Sensoren
- Positionsbestimmung mit zunehmender Integration genauer

Nachteile:

- Positions-Wahrscheinlichkeit stark vom Umweltmodell abhängig
- Verfahren noch nicht optimiert
- Langsame Lokalisation (10 Sekunden)

2.4 Lokalisation mittels Objekterkennung

Mit Hilfe von Objekterkennung wird in [Holenstein et al. 1992] eine Klassifizierung der gegebenen Umwelt vorgenommen. Es werden hierfür die vier Objekt-Klassen *Ecken*, *Kanten*, *Wände* und *Unbekannt* eingeführt. Klassifizierte Objekte werden in Form von vier Parametern in einer Daten-Matrix abgelegt. Um nun den Roboter zu lokalisieren, wird der euklidische Abstand zwischen den aktuell gefundenen Objekten und denen in der Daten-Matrix gemessen. Hierbei bilden sich Cluster heraus, die für die gegebene Konstellation Positions-Schätzungen darstellen. Geeignete Cluster-Algorithmen bestimmen schließlich die aktuelle Roboter-Position.

Vorteile:

- Robust gegenüber unbekannten Hindernissen

Nachteile:

- Langsame Lokalisation (10 Sekunden)
- Verfahren nur lokal anwendbar
- Nur im Stillstand anwendbar
- Wahl der Objekt-Parameter umweltabhängig

2.5 Sensorfusion mit Hilfe von Kalmanfiltern

Der in [Curran, Kyriakopoulos 1993] vorgestellte Algorithmus nutzt einen erweiterten Kalman-Filter, um die Position und Orientierung aus den gegebenen Sensordaten zu bestimmen. Mit Hilfe dieser Daten (einschließlich der odometrischen Daten) werden die zukünftige Position und die zu erwartenden Sensorwerte berechnet. Starke Abweichungen, wie z.B. das plötzliche bzw. unvorhersehbare Auftauchen von Hindernissen, werden durch den erweiterten Kalman-Filter bezüglich der Lokalisation ausgeglichen und können zur reaktiven Hindernisvermeidung herangezogen werden. Die odometrische Position wird bei diesem Ansatz ständig durch eine Rauschkorrektur verbessert.

Vorteile:

- Sensorfusion möglich
- Robust gegenüber unbekannten Hindernissen
- Zykluszeit ca. 150-200 Millisekunden

Nachteile:

- Startposition muß bekannt sein
- Sensorwerte müssen unabhängig voneinander sein
- Anfällig gegenüber systematischen Fehler der Sensoren

2.6 Sensoreindruck-Korrelation

Der Algorithmus aus [Drumheller 1987] korreliert Segmente[3] mit dem Umwelt-Modell. Nicht mögliche Konfigurationen (Position und Orientierung) werden aus dem Suchraum entfernt. Hierfür wird ein Verfahren, welches Drumheller "sonar-barrier-test" nennt, genutzt. Um die Position des Roboters festzustellen, wird nur ein einziger Sensor-Eindruck verwendet[4], mit dem der Suchbaum aufgebaut wird. Dieser, auch Interpretationsbaum genannt, enthält als Blätter die einzelnen Interpretationen, d.h. Zuordnungen der Wände zu Segmenten.

Vorteile:

- Verfahren tolerant gegenüber Rauschen und unbekannten Objekten
- Verfahren zur Suchraumverkleinerung enthalten

Nachteile:

- Messungen nur im Stillstand möglich
- Relativ lange Messdauer (ca. 15 Sekunden)
- Schlechte Ergebnisse in Gängen und rechteckigen Räumen
- Keine Integration der Daten

[3]Segmente bestehen aus Sensor-Messungen, die in einem bestimmten Entfernungsbereich liegen.

[4]Ein Sensor-Eindruck beinhaltet 100 Messungen um 360°.

Kapitel 3

Beitrag dieser Arbeit

Wie eingangs schon erwähnt, wird für die nachfolgenden Teilprobleme, soweit dies möglich ist, auf schon etablierte Verfahren zurückgegriffen. So ist beispielsweise die erweiterte Hough-Transformation durchaus nicht neu, aber eher unüblich im Bereich der Driftkompensation. Ebenso verhält es sich bei dem geradenbasierten Ansatz. Die Neuerung liegt in einer geeigneten Kombination und Verbesserung der herkömmlichen Verfahren. Es soll somit erreicht werden, daß Fehler bisheriger Ansätze ausgeräumt und die üblichen Probleme, wie das Fehlen der Echtzeitfähigkeit, gelöst werden. Die hier vorgestellten Verfahren haben deshalb gegenüber den traditionellen Methoden der Driftkompensation die folgenden Vor- und Nachteile.

Vorteile:

- Verfahren implementiert und getestet
- Geringe Anforderungen an die Qualität der Sensoren
- Genaue Positions-Bestimmung
- Geringer Rechenaufwand (echtzeitfähig)
- Integration der Daten
- Sensorfusion
- Keine Modifikation der Umwelt notwendig

Nachteile:

- Verfahren nur zur Driftkompensation anwendbar
- Globale Karte notwendig

Aus dieser Aufstellung läßt sich einfach erkennen, daß viele Einschränkungen anderer Ansätze durch die geeignete Kombination und entsprechende Verbesserung der schon existierenden Verfahren entfallen oder zumindest abgeschwächt werden. Die globale Karte liegt dabei als CAD-Modell vor, während die über die Zeit integrierten Sensoreindrücke die lokale Karte darstellen.

3.1 Hough-basierte Driftkompensation

Die lokale Karte wird als zu suchendes Segment innerhalb der globalen Karte aufgefaßt. Hier bietet sich eine Erweiterung der Hough-Transformation an. Diese ist dahingehend erweitert, daß nicht nur einfache Linien detektiert werden, sondern auch ganze Segmente gefunden werden können. Da die Hough-Transformation weder skalen- noch rotationsinvariant ist, muß das gesuchte Segment in verschiedenen Rotationsstufen dem Verfahren als Eingabe übergeben werden. Die Skaleninvarianz spielt in diesem Zusammenhang keine Rolle, da beide Karten schon in derselben Skalierung vorliegen. Das Ergebnis der Hough-Transformation ist ein sogenanntes Akkumulatorfeld, welches Aufschluß über die aktuelle Position gibt. Dieses wird anschließend mit geeigneten Verfahren analysiert, um den Driftkompensations-Vektor bestimmen zu können.

3.2 Geradenbasierte Driftkompensation

Bei der geradenbasierten Driftkompensation wird jedem Pixel der lokalen Karte ein Liniensegment der globalen Karte zugeordnet. Anschließend müssen die Abstände der einzelnen Punkte zu ihren zugeordneten Linien minimiert werden. Der Ergebnisvektor dieses Minimierungsproblems stellt den Driftkompensations-Vektor dar. Für diesen Ansatz ist es notwendig, daß die globale Karte, die als CAD-Modell[1] vorliegt, in eine geradenbasierte Karte transformiert wird. Hierfür bietet sich eine vorangehende Eckpunktextraktion und daran anschließende Geradenbildung an.

3.3 Der Raum-Matching Algorithmus

Bei diesem Ansatz wird für jede Gitterzelle in der globalen Karte der Abstand zum nächsten Hindernis in den absoluten Orientierungen 90°, 180°, 270° und 360° bestimmt. Während der Laufzeit mißt auch der Roboter den Abstand in den jeweiligen Orientierungen zum nächsten Hindernis. Um den Roboter zu lokalisieren, wird das von ihm erzeugte 4-Tupel mit den in der globalen Karte enthaltenen verglichen. Der Driftkompensations-Vektor ergibt sich aus der aktuellen und der durch dieses Verfahren neu bestimmten Position.

[1] d.h. als pixelorientierte Bitmap

Kapitel 4

Modul-Aufbau der Software

Die Aufgabe, einen Roboter zielorientiert von einem Raum in einen anderen fahren zu lassen, ohne die Umwelt zu präparieren, erfordert die Kooperation mehrerer Software-Module. So müssen die von der Hardware gelieferten Sensor-Daten geeignet aufbereitet werden, um entsprechende Aktionen einleiten zu können. Abbildung 4.1 zeigt den konzeptionellen Modul-Aufbau auf der Informationsebene. Während Ovale und Kreise Daten symbolisieren, stellen Rechtecke Module dar. Der *Mapmaker* bereitet die von den Ultraschall-Sensoren gemessenen Entfernungswerte zu einer Kurzzeit-Karte auf, welche den unmittelbaren Sensoreindruck in Kartenform darstellt. Die Kurzzeit-Karten werden dann mit Hilfe des *Integrators* in ein Besetztheitsgitter eingegliedert. Ein solcher Ansatz beugt Fehlmessungen vor, die in der lokalen Karte nur geringfügig berücksichtigt werden. Anschließend wird mit Hilfe der globalen und der lokalen Karte die Drift, die mit zunehmend zurückgelegtem Weg größer wird, berechnet. Der resultierende Driftkompensations-Vektor dient dem *Navigator* zur korrekten Bestimmung seines Ziels. Außerdem wird die korrigierte odometrische Position im *Piloten* und im *Driver* für die Festlegung kurzfristiger Zwischenziele benötigt.

Der *Driver* und das *Hardwareinterface* sind in einem einzigen Prozeß realisiert, um den Kommunikationsaufwand möglichst gering halten zu können. Dies stellt sicher, daß die Reaktionszeiten des *Drivers* nicht durch zu lange Latenzzeiten der Kommunikation blockiert werden. Jedes Modul ist ein selbständiger Prozeß, der somit auf einem Cluster von Rechnern verteilt werden kann. Dies erhöht bei Prozessen, deren Kommunikations-Overhead gering ist, erheblich die Performance. Für einen reibungslosen Ablauf ist zusätzlich ein Startprozeß notwendig, der die anderen Prozesse auf den entsprechenden Rechnern startet und die Ports für die Interprozeß-Kommunikation festlegt. Es bieten sich hier verschiedene Kommunikationsebenen an. Prozesse, die auf unterschiedlichen Computern laufen, kommunizieren via *TCP/IP*. Rechnen zwei Programme auf dem gleichen Rechner, so tauschen sie ihre Informationen über *Shared-Memory-Segmente* oder *Message Queues* aus. Das für die Kommunikation zuständige Modul setzt dabei auf den von Unix zur Verfügung gestellten *Sockets* auf und ist auf die speziellen

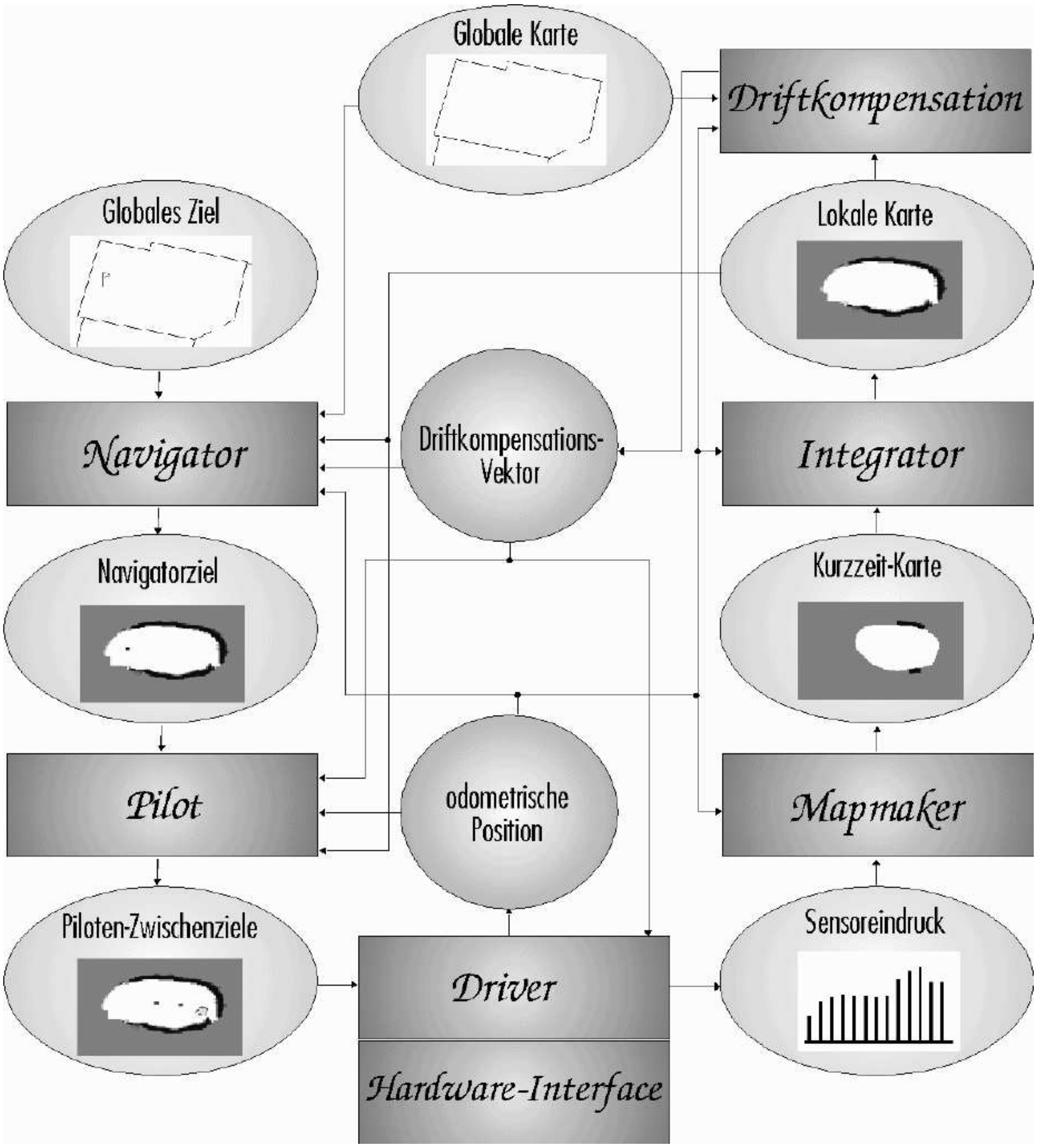

Abbildung 4.1: Der Software-Modul-Aufbau

Bedürfnisse der Robotik (Echtzeitfähigkeit[1]) zugeschnitten. Die Modulstruktur stellt zudem sicher, daß das Gesamtsystem nicht zwangsläufig ausfällt, falls einzelne Module stehen bleiben.

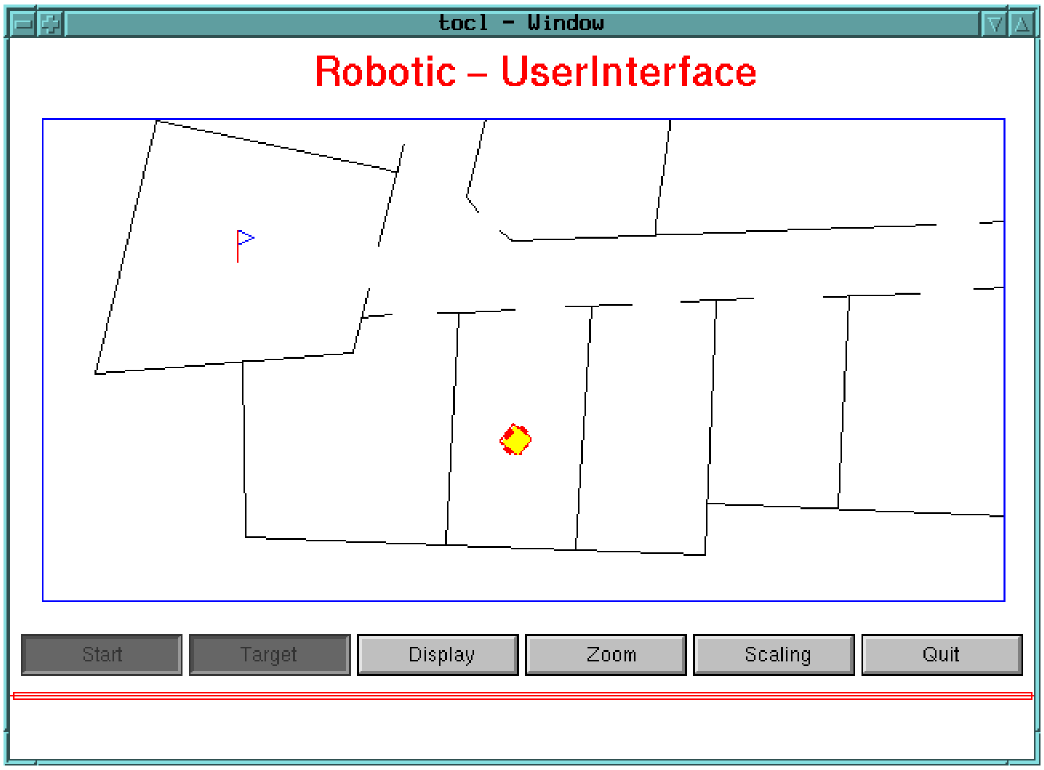

Abbildung 4.2: Das *Robotic-User-Interface*

Als Benutzerschnittstelle dient das in Abbildung 4.2 dargestellte *Robotic-User-Interface*, welches auf der Klassenbibliothek *tocl* basiert (s. [Haase 1997]). Hier können dem *Navigator* das globale Ziel und die aktuelle Roboter-Position mitgeteilt werden. Der *Navigator* reicht dann die Position an alle übrigen Module weiter.

Um den Kommunikationsaufwand zu verringern, werden alle notwendigen Daten in einem Paket zusammengefaßt versendet. Für die Driftkompensation ist dies von entscheidender Bedeutung. Da Integration der Sensorwerte und die Kompensationsberechnung einige Zeit in Anspruch nehmen, müssen odometrische Daten und Kartendaten übereinstimmen. Es wird daher bei jedem Sensoreindruck die odometrische Position mit verschickt. Die versendeten Karten besitzen somit eine Kennung, die durch den *Integrator* zur *Driftkompensation* geschleust wird. Da nicht nur die odometrischen Daten, sondern auch die Zeit in der jeweiligen Karte

[1] Da Unix kein Echtzeit-Betriebssystem ist, wird lediglich das Verzögern von Nachrichten verhindert.

kodiert ist, kann vom *Mapmaker* immer der letzte Sensoreindruck zur Kartenbildung herangezogen werden. Ältere Sensoreindrücke werden einfach verworfen.

Wie oben schon erwähnt, ist das System wegen des Modulaufbaus robust gegenüber Störungen. Fällt beispielsweise ein kompletter Rechner im Cluster aus, ist trotzdem, durch die reaktive Steuerung, sichergestellt, daß der Roboter sich kollisionsfrei bewegt. Das gilt allerdings nur mit der Einschränkung, daß der ausgefallene Rechner nicht der des Roboters ist. Bleibt dieser stehen, so stoppt das komplette System, da keine weiteren Instruktionen an die Hardware gesendet werden kann. Auf dieser untersten Ebene wird der Roboter mit Hilfe des *TCX*-Kommunikationspaketes der *Carnegie Mellon University* gesteuert. Es wurde für die Interprozeß-Kommunikation im Bereich der mobilen Roboter entwickelt, um Anwendungen über verschiedene Hardware-Architekturen, Betriebssysteme und Programmiersprachen zu unterstützen.

Kapitel 5

Grundlagen

5.1 Sensormodelle

Da alle folgenden Ansätze von dem Sensormodell abhängen, ist eine genaue Kenntnis der Beschaffenheit und Genauigkeits des verwendeten Sensors von entscheidender Bedeutung. Diese beiden Eigenschaften beinflussen maßgebend das zugehörige Modell. So besitzen Ultraschall-Sensoren gegenüber Laserentfernungsmessern eine wesentlich geringere Winkelauflösung. Das angewandte Modell muß dieser Eigenschaft Rechnung tragen. Der Öffnungswinkel wiederum bestimmt sich aus der Frequenz und dem Durchmesser des Wandlers. Je größer die Frequenz ist, desto kleiner ist der Öffnungswinkel. Die Verbesserung in der Winkelauflösung wird allerdings mit einer geringeren Reichweite des Sensors erreicht. Um nun den Öffnungswinkel ξ eines bestimmten Ultraschall-Wandlers in Bogenmaß zu bestimmen, ergibt sich nach [Brown 1985] Ausdruck (5.1). Dazu sei d der Durchmesser des Wandlers in Millimetern.

$$\xi = 1.22 \cdot \frac{\lambda}{d} \tag{5.1}$$

Wird für die Schallgeschwindigkeit die Konstante c gewählt, so läßt sich die akustische Wellenlänge λ invers proportional zur Frequenz f angeben[1]:

$$\lambda = \frac{c}{f}. \tag{5.2}$$

Abbildung 5.1 zeigt ein typisches Richtdiagramm eines Ultraschall-Wandlers bei einer Frequenz von 50 kHz. Die Pegel in dieser Darstellung sind in dB auf eine Achse normalisiert. Diese Ausgangsstrahl-Charakteristik fließt letztendlich in die Bildung des Sensormodells ein. Während neuronale Netze in [Thrun 1996] und [van Dam et al. 1996] für die Transformation der Sensorwerte in Wahrscheinlichkeits-Karten verwendet werden, lernen sie bei [Moravec, Blackwell 1992] lediglich

[1] Die Schallgeschwindigkeit ist dabei in mm/s und die Frequenz in Hz anzugeben. Für die Wellenlänge ergibt sich somit die Einheit mm.

die Parameter des Sensormodells, die von der jeweilig gemessenen Entfernung abhängig sind. Obwohl die beiden Ansätze auf dem ersten Blick sehr unterschiedlich zu sein scheinen, zeigt sich bei näherer Betrachtung, daß sie Gemeinsamkeiten besitzen. So werden auch bei ersterem Ansatz implizit für verschiedene Entfernungen unterschiedliche Sensormodelle eingesetzt. Zunächst jedoch wird im nächsten Abschnitt das einfachste Sensormodell besprochen.

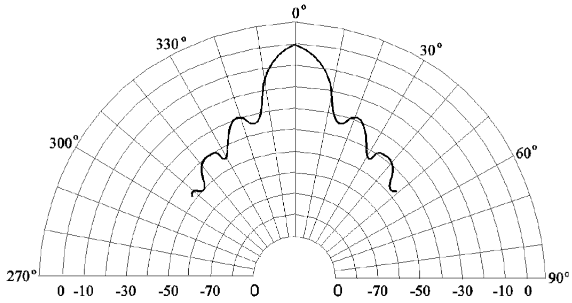

Abbildung 5.1: Ausgang eines Ultraschall-Wandlers (aus [Polaroid 1993])

5.1.1 Das naive Sensormodell

Das naive Sensormodell geht von einer kegelförmigen Ausbreitung der Schallwellen aus. Abbildung 5.2 zeigt die vereinfachte Interpretation der Sensorinformation. Während links der tatsächliche Ausgang des Ultraschall-Wandlers dargestellt ist, ist rechts der schematisierte Kegel zu sehen[2]. Letzterer liegt schließlich dem Modell zu Grunde.

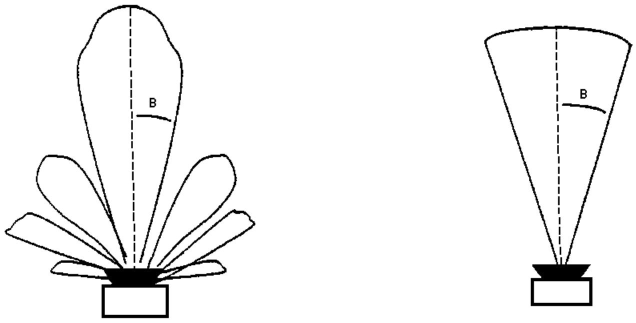

Abbildung 5.2: Interpretation der Sensorinformation

Mißt der Sensor nun in einer bestimmten Entfernung ein Hindernis, so wird dieses auf der gesamten Kegelbreite in der lokalen Karte eingetragen. Dabei spielt

[2]Der Parameter B gibt den Öffnungswinkel des Ultraschall-Wandlers an.

es keine Rolle, ob das detektierte Objekt im Zentrum oder im Randbereich des Kegels liegt. Die Belegtheits-Wahrscheinlichkeit[3] einer Zelle G_i ist auf die folgenden fünf Stufen diskretisiert:

1. *EMPTY*: kein Hindernis ($p(G_i) = 0.0$)

2. *MEDUNKNOWN*: kein Hindernis ($p(G_i) = 0.25$)

3. *UNKNOWN*: unbekanntes Gebiet ($p(G_i) = 0.5$)

4. *MEDOBSTACLE*: Hindernis ($p(G_i) = 0.75$)

5. *OBSTACLE*: Hindernis ($p(G_i) = 1.0$)

Da dieses Modell für eine einfache und schnelle Kartenbildung geeignet ist (siehe Anhang A), hat es in der Robotik eine hohe Akzeptanz erreicht. Eine erste Verfeinerung dieses Modells wird durch die Verwendung von Gaußfunktionen für das Hindernis und dem freien Bereich erzielt.

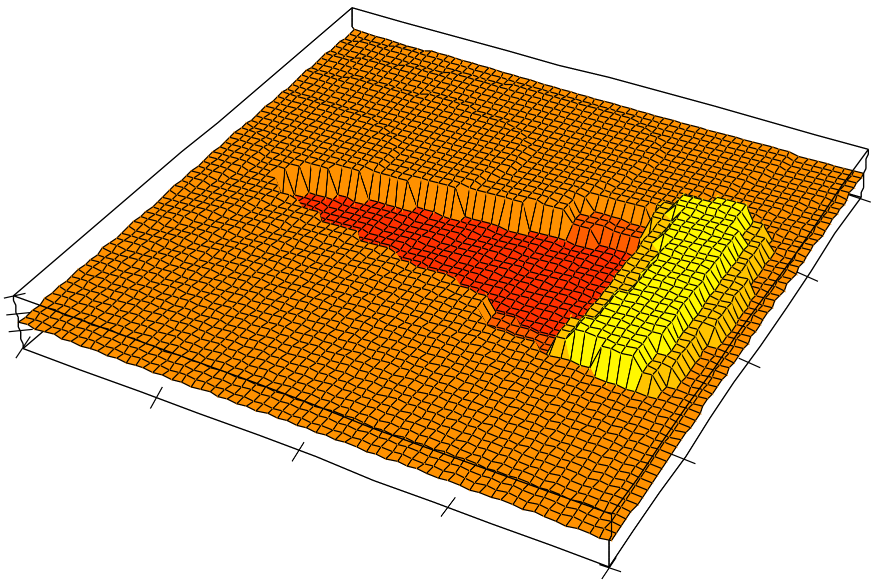

Abbildung 5.3: Das naive Sensormodell

[3]siehe Abschnitt 5.2

Abbildung 5.3 zeigt den diskretisierten Eindruck eines einzigen Sensors. Der Hauptnachteil des naiven Sensormodells zeigt sich jedoch in unerwünschten Störungen, die insbesondere in Gang- und Türsituationen auftreten können. Entfernt sich die lokale Karte zu weit von der Realität, versagt schließlich die Navigation[4].

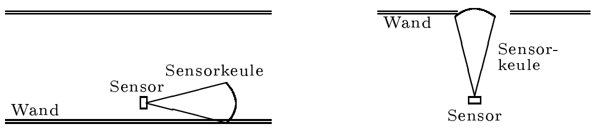

Abbildung 5.4: Problem des Zuwachsens

Dieser Effekt wird in Abbildung 5.4 verdeutlicht. Die beiden Sensoren messen in ihrem jeweiligen Randbereich noch die Wand. Da diese in der vollen Kegelbreite in den Sensoreindruck eingetragen wird, kommt es zu Störungen. Sind überlappende Sensoren am Roboter befestigt, so läßt sich dieser Effekt auf den nicht überlappenden Bereich beschränken. Das geschieht, indem im Sensoreindruck nur auf Zellen, die noch nicht dem freien Bereich zugeordnet sind, Hindernisse verzeichnet werden dürfen. Trotz dieser Verbesserung ist das Problem an sich nicht gelöst. Es bietet sich also die Modifikation des Sensormodells an.

5.1.2 Das modifizierte Sensormodell

Ultraschall-Sensoren sind im Nahbereich exakt. Mit zunehmender Entfernung nimmt jedoch auch die Genauigkeit der Sensoren ab. Es ist also notwendig, ein Sensormodell einzuführen, welches auf diese Eigenschaften der Sensoren eingeht. Während das in [Moravec, Blackwell 1992] vorgestellte Verfahren dynamisch das geeignete Modells bestimmt, wird bei der modifizierten Version der Entfernungsbereich auf fünf Ebenen diskretisiert. Im Nahbereich, bis $60cm$, ist der Sensor genau genug, um das naive Sensormodell verwenden zu können. Aber schon bei größeren Entfernungen empfiehlt es sich, den Bereich des Sensoreindruckes, der das Hindernis darstellt, zu verkleinern und den Kegel schmaler werden zu lassen. Um nun die Breite der Kegel zu reduzieren, wird ein zweiter, schmalerer in den ersten gelegt. Dabei ist darauf zu achten, daß sich das jeweilige Modell der Vorgänger-Entfernung in das nachfolgende ohne Lücken einbettet, da ansonsten Teile von ursprünglich freien Bereichen mit dem Wert *UNKNOWN* belegt werden würden. Abbildung 5.5 zeigt das modifizierte Sensormodell. Ein vom Sensor aufgenommener Entfernungswert läßt sich nun in eine der folgenden Kategorien einstufen und somit dem entsprechendem Modell zuführen. Das modifizierte Sensormodell hält verschiedene Modelle für die in der folgenden Liste aufgeführten Entfernungsbereiche bereit:

[4]Türen werden nicht mehr als solche erkannt.

1. kleiner gleich $60cm$
2. größer $60cm$ und kleiner gleich $90cm$
3. größer $90cm$ und kleiner gleich $130cm$
4. größer $130cm$ und kleiner gleich $160cm$
5. größer $160cm$

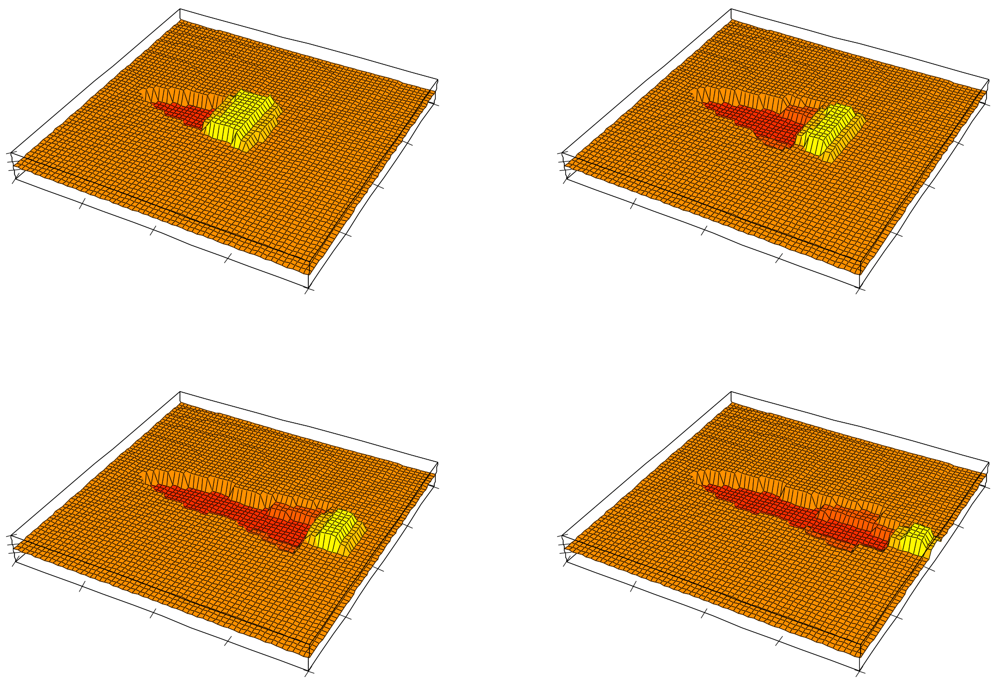

Abbildung 5.5: Das modifizierte Sensormodell

Bei Entfernungswerten, die größer $160cm$ sind, wird in der zugehörigen Diskretisierungsebene[5] kein Hindernis markiert, sondern lediglich der frei zu befahrende Bereich verzeichnet.

Bemerkung 5.1 *Bei der Implementierung kann die erforderliche Diskretisierung des Sensoreindruckes zu unerwünschten Störungen führen.*

[5]Diese ist nicht in Abbildung 5.5 zu sehen.

5.1.3 Das Prioritäten-Modell

Das Ziel des Prioritäten-Modells ist, ähnlich dem modifizerten Sensormodell, die stärkere Gewichtung kurzer Entfernungen gegenüber längeren. Dies resultiert, wie im Abschnitt 5.1.2 bereits erwähnt, aus der Tatsache, daß die Sensorik im Nahbereich wesentlich genauer ist. Der zu untersuchende Bereich wird deshalb in zwei Entfernungsgebiete aufgeteilt. Eins stellt den Nahbereich $\mathcal{N}$ dar, während das andere Gebiet für den Fernbereich $\mathcal{F}$ steht. Jeder Punkt des Sensoreindruckes hält nun die folgenden Zähler bereit:

- $\mathcal{N}_{miss}$: Anzahl der *EMPTY*-Messungen im Nahbereich
- $\mathcal{N}_{hit}$: Anzahl der *OBSTACLE*-Messungen im Nahbereich
- $\mathcal{F}_{miss}$: Anzahl der *EMPTY*-Messungen im Fernbereich
- $\mathcal{F}_{hit}$: Anzahl der *OBSTACLE*-Messungen im Fernbereich

Anschließend wird auf demselben Wege wie im naiven Sensormodell das umschließende Rechteck gebildet. Ein Punkt p, der innerhalb des Sensorkegels liegt und dessen Entfernung kleiner als die gemessene ist[6], wird auf seinen Zugehörigkeitsbereich untersucht. Gehört dieser zu $\mathcal{N}$ wird der entsprechende Zähler $\mathcal{N}_{miss}$ von p erhöht, andernfalls der Zähler $\mathcal{F}_{miss}$. Punkte, die im naiven Modell den Wert *OBSTACLE* zugewiesen bekämen, erhöhen die Zähler $\mathcal{N}_{hit}$ bzw. entsprechend $\mathcal{F}_{hit}$.

Die einzelnen Zellen des Sensoreindruckes enthalten nun Häufigkeiten. Diese können zur Festlegung des Belegtheitsgrades einer Zelle herangezogen werden. Damit die Belegtheits-Wahrscheinlichkeit nicht durch Störungen bzw. Ausreißer verfälscht werden kann, wird zuvor die Gesamtzahl aller Messungen für die entsprechende Zelle ermittelt:

$$\mathcal{N}_{weight} = \mathcal{N}_{miss} + \mathcal{N}_{hit} \quad \text{und} \tag{5.3}$$

$$\mathcal{F}_{weight} = \mathcal{F}_{miss} + \mathcal{F}_{hit} \tag{5.4}$$

Ist das Gewicht einer Zelle größer null, so kann das Verhältnis zwischen detektiertem Hindernis und Gesamtzahl der Messungen gebildet werden:

$$\mathcal{N}_{frac} = \frac{\mathcal{N}_{hit}}{\mathcal{N}_{weight}} \quad \text{und} \tag{5.5}$$

$$\mathcal{F}_{frac} = \frac{\mathcal{F}_{hit}}{\mathcal{F}_{weight}}, \tag{5.6}$$

andernfalls wird der Wert $0,5$ angenommen. Anschließend können die Werte $\mathcal{N}_{weight}$, $\mathcal{F}_{weight}$, $\mathcal{N}_{frac}$ und $\mathcal{F}_{frac}$ zur letztendlichen Bestimmung der Zell-Besetztheit verwendet werden. Sind beispielsweise nicht ausreichend viele Messungen

[6] und somit im naiven Sensormodell den Wert *EMPTY* zugeteilt bekäme.

bezüglich der betrachteten Gitterzelle eingegangen[7], d.h. $\mathcal{N}_{weight} + \mathcal{F}_{weight} < \theta$, so kann grundsätzlich der Wert *UNKNOWN* angenommen werden. Die Verhältnisse $\mathcal{N}_{frac}$ und $\mathcal{F}_{frac}$ eignen sich für die Einteilung der Gitterwerte in die Bereiche *EMPTY* bzw. *OBSTACLE*.

Werden zusätzlich Alterungsfaktoren für die zuvor eingeführten Zähler verwendet, ist sichergestellt, daß einzelne Zellen nicht überproportional stark gewichtet werden. Dies kommt häufig in Situationen vor, in denen der Roboter über einen längeren Zeitraum nahezu an der gleichen Position steht.

Bemerkung 5.2 *Da erst nach einer vorbestimmten Anzahl von Messungen die Besetztheits-Wahrscheinlichkeit neu gesetzt wird, trägt dieses Modell zu einer erhöhten Fehlertoleranz bei.*

5.2 Besetztheitsgitter

Besetztheitsgitter bilden den Grundstock für die Repräsentation der Sensordaten. Jede Gitterzelle stellt einen kleinen Bereich des Raumes dar[8] und beinhaltet dessen Besetztheits-Wahrscheinlichkeit. Es stellt sich somit das Problem der Kombination zweier Gitterwerte: Zum einen die schon vorhandene Besetztheits-Wahrscheinlichkeit und zum anderen die mit den Sensoren neu gemessene. In den folgenden Abschnitten wird auf die, in den Artikeln von [Moravec 1988] und [Elfes 1989] beschriebene, Methode näher eingegangen.

5.2.1 Eigenschaften der Wahrscheinlichkeitsrechnung

Die Formel von Bayes ist ein geeignetes Mittel für die Kombination zweier Wahrscheinlichkeiten. Seien hierzu die beiden Ereignisse A_1 und A_2 derart gegeben, daß $E = A_1 \cup A_2$ das sichere Ereignis darstellt. Sind weiterhin die beiden Ereignisse unvereinbar, d.h. $A_1 \cap A_2 = \emptyset$, so sind alle Voraussetzungen für den Satz von Bayes erfüllt und es kann folgender Ausdruck angegeben werden:

$$p(A_i|B) = \frac{p(B|A_i) \cdot p(A_i)}{p(B|A_1) \cdot p(A_1) + p(B|A_2) \cdot p(A_2)} \ , \quad i \in \{1, 2\}. \tag{5.7}$$

Dabei bezeichnet $p(A|B)$ die bedingte Wahrscheinlichkeit, die wie folgt definiert ist:

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \ , \quad \text{falls} \quad p(B) \neq 0. \tag{5.8}$$

[7]Der Grenzwert θ liegt in der Regel zwischen drei und fünf.

[8]Die Besetztheitsgitter, die in den folgenden Ansätzen verwendet werden, haben eine Zellengröße von $10cm \cdot 10cm$.

Im Fall von Besetztheitsgittern ist von Interesse, welche Belegtheits-Wahrscheinlichkeiten bestimmte Gitterzellen besitzen. Sei hierfür $p(o)$ die Wahrscheinlichkeit, daß die betrachtete Zelle besetzt ist. Entsprechend sei $p(\neg o)$ die Wahrscheinlichkeit, daß die Zelle nicht belegt ist. Für eine neue Messung B und die alte Information A ergibt sich mittels (5.7):

$$p(o|B \cap A) = \frac{p(B|o \cap A) \cdot p(o|A)}{p(B|o \cap A) \cdot p(o|A) + p(B|\neg o \cap A) \cdot p(\neg o|A)}, \tag{5.9}$$

$$p(\neg o|B \cap A) = \frac{p(B|\neg o \cap A) \cdot p(\neg o|A)}{p(B|o \cap A) \cdot p(o|A) + p(B|\neg o \cap A) \cdot p(\neg o|A)}. \tag{5.10}$$

Aus (5.9) und (5.10) läßt sich direkt das Verhältnis

$$\frac{p(o|B \cap A)}{p(\neg o|B \cap A)} = \frac{p(B|o \cap A)}{p(B|\neg o \cap A)} \cdot \frac{p(o|A)}{p(\neg o|A)} \tag{5.11}$$

zwischen der Besetztheitswahrschlichkeit bzw. der Nicht-Besetztheitswahrschlichkeit bilden. Um den Formalismus weiter zu vereinfachen, wird in [Moravec 1988] der Begriff *Odds* eingeführt:

$$Odds(A) = \frac{p(A)}{p(\neg A)} = \frac{p(A)}{1 - p(A)}. \tag{5.12}$$

Somit ergibt sich für die unbedingte Wahrscheinlichkeit:

$$p(A) = \frac{Odds(A)}{1 + Odds(A)}. \tag{5.13}$$

Ausdruck (5.11) läßt sich nun mittels des neu eingeführten Begriffes *Odds* vereinfacht darstellen:

$$Odds(o|A \cap B) = Odds(B|o \cap A) \cdot Odds(o|A). \tag{5.14}$$

5.2.2 Kombination zweier Wahrscheinlichkeitswerte

Mit Hilfe des in Abschnitt 5.2.1 eingeführten Formalismus lassen sich nun einfache Regeln für die Kombination zweier Wahrscheinlichkeitswerte angeben. Der Satz von Bayes kombiniert zwei unabhängige Informationsquellen A und B, die für die Besetztheit einer bestimmten Zelle von Bedeutung sind, zu einer neuen Schätzung $p(o|A \cap B)$. Die Information B enthält dabei die neu gemessene Besetztheits-Wahrscheinlichkeit $p(B|o)$ dieser Zelle. Aus dieser läßt sich unmittelbar die Wahrscheinlichkeit der Nichtbesetztheit $p(B|\neg o)$ ableiten. Diese Negation stellt sicher, daß die Voraussetzungen des Satzes von Bayes erfüllt sind[9].

[9] Es gilt: $\{B|o\} \cap \{B|\neg o\} = \emptyset$ und $E = \{B|o\} \cup \{B|\neg o\}$ stellt das sichere Ereignis dar.

Wird nun Ausdruck (5.11) ausschließlich auf die Information B angewandt, d.h. A enthält für diese Zelle keine weitere Information, so ergibt sich

$$\frac{p(o|B)}{p(\neg o|B)} = \frac{p(B|o)}{p(B|\neg o)} \cdot \frac{p(o)}{p(\neg o)}. \tag{5.15}$$

An dieser Stelle wäre es von großem Vorteil, wenn (5.15) in der ursprünglichen Bayes Formel (Ausdruck (5.9)) verwendet werden könnte. Dies würde die Bestimmung von $p(o|A \cap B)$ nur unter Verwendung von $p(o|A)$ und $p(o|B)$ ermöglichen. Diese Unabhängigkeit ist allerdings im allgemeinen nicht gegeben, da sich Messungen beispielsweise überschneiden können. Wird trotzdem die Annahme getroffen, daß B unabhängig von A ist, d.h. es gilt (5.16), können die Ausdrücke (5.11), (5.15) und (5.16) zur Kartenbildung herangezogen werden:

$$\frac{p(B|o \cap A)}{p(B|\neg o \cap A)} = \frac{p(B|o)}{p(B|\neg o)} \quad \text{und} \tag{5.16}$$

$$\frac{p(o|A \cap B)}{p(\neg o|A \cap B)} = \frac{p(o|B)}{p(\neg o|B)} \cdot \frac{p(o|A)}{p(\neg o|A)} \cdot \frac{p(\neg o)}{p(o)}. \tag{5.17}$$

Wird für die a-priori-Wahrscheinlichkeit der Besetztheit einer Zelle der Wert $0,5$ angenommen, d.h. $p(o) = p(\neg o) = 0,5$, so vereinfacht sich (5.17) zu:

$$\frac{p(o|A \cap B)}{p(\neg o|A \cap B)} = \frac{p(o|B)}{p(\neg o|B)} \cdot \frac{p(o|A)}{p(\neg o|A)}. \tag{5.18}$$

Gleichung (5.18) läßt sich nun, mit Hilfe des in Abschnitt 5.2.1 definierten Ausdrucks *Odds* (Ausdruck (5.12)), darstellen:

$$Odds(o|A \cap B) = Odds(o|A) \cdot Odds(o|B). \tag{5.19}$$

5.2.3 Der Integrator

Der Integrator basiert nun auf den von [Moravec 1988] und [Elfes 1989] beschriebenen Besetztheitsgittern, d.h. es werden mehrere Sensor-Eindrücke für die Kartenbildung herangezogen[10]. Zusätzlich stellt sich das Problem einer geeigneten Karten-Verschiebung. Da der Roboter stets im Kartenzentrum positioniert ist, muß die lokale Karte gemäß der Roboter-Bewegung angepaßt werden.

Abbildung 5.6 zeigt das Integrator-Fenster nach einer erfolgten Verschiebung der lokalen Karte. Die Farbe gibt dabei Auskunft über die jeweilige Besetztheits-Wahrscheinlichkeit[11]. Für die Transformation der Besetztheitsgitter existieren

[10] Die Sensor-Eindrücke stellen einen kontinuierlichen Datenstrom dar und werden deshalb über die Zeit integriert.

[11] Je dunkler die Kartenfarbe, desto höher die Besetztheits-Wahrscheinlichkeit.

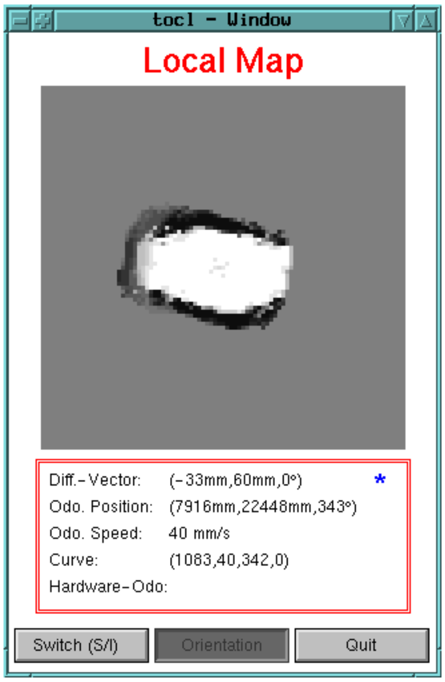

Abbildung 5.6: Die Integratorkarte

nun verschiedene Ansätze. So wird in [van Dam et al. 1990] ein neuronales Netz verwendet, um Gitter ineinander zu überführen. Wird bereits bei der Bildung des Sensoreindrucks die Rotation berücksichtigt, d.h. diese liegen immer in der gleichen absoluten Orientierung vor, so handelt es sich bei der zu betrachteten Transformation lediglich um eine Translation. Dies kann durch eine einfache Verschiebung der lokalen Karte, entsprechend dem zurückgelegten Weg, vorgenommen werden. Zeilen bzw. Spalten, die bei der Translation den Kartenbereich verlassen, werden verworfen, neu hinzukommende mit dem Wert *UNKNOWN* belegt. Der Integrator wendet das zuletzt vorgestellte Verfahren zur Karten-Verschiebung an.

Bemerkung 5.3 *Alterungsfaktoren sorgen auch hier für eine schwächere Gewichtung älterer Daten gegenüber neu aufgenommenen.*

5.3 Neuronale Netze

Während in der klassischen KI versucht wird, Vorgänge im Gehirn mit Hilfe von Regeln auszudrücken, lernen neuronale Netze an Hand von Beispielen. Dabei lassen sie sich, wie in [Köhle 1990] bzw. [Haykin 1994] ausgeführt, folgendermaßen charakterisieren.

1. Das Netz besteht aus elementar modellierten Neuronen, die Eingangssignale aufsummieren und ein Ausgangssignal bilden.

2. Es existieren gewichtete Verbindungen zwischen diesen Neuronen in einer vorgegebenen Topologie.

3. Die Gewichte der Verbindungen werden aufgrund von Algorithmen verändert.

Die Wechselwirkung der einzelnen Neuronen übernimmt die Informationsverarbeitung. Dabei können exzitatorische oder auch inhibitorische Signale[12] ausgesendet werden. Diese werden wiederum von Neuronen empfangen, die ihrerseits Signale generieren und verschicken. Die Verbindungen zwischen den Nervenzellen sind außerdem in der Lage, die zu transportierenden Signale abzuschwächen oder auch zu verstärken. Nicht selten werden sie deshalb auch kurz als Gewichte bezeichnet. Das *Wissen* eines derartigen Systems wird nun in erster Linie in den einzelnen Gewichten repräsentiert.

5.3.1 Biologisch motivierter Ansatz

Die Idee der neuronalen Netze liegt in der Biologie bzw. der Neurophysiologie begründet. So hat Cajal um 1908 Neuronen als elementaren Bestandteil des Gehirns identifiziert. In Abbildung 5.7, welche ein Neuron darstellt, sind deutlich die Dendriten, das Axon und die Synapsen zu erkennen. Sie stellen auf Informationsaustausch spezialisierte Zellen dar, die diesen via elektrisch geladener Teilchen[13] vornehmen. Übersteigt die einströmende Ladung einen Schwellwert, so werden Aktionspotentiale[14] ausgesendet. Diese Aktionspotentiale wurden sechs Jahre später von Adrian entdeckt. Frühe Modelle, die die Signalverbreitung beschreiben, basieren auf Untersuchungen der relativ großen Axonen im Tintenfisch. [Hodgkin, Huxley 1952] legten in Form von mathematischen Gleichungen den Ionenfluß, der diese Aktionspotentiale zur Folge hat, fest. Diese Gleichungen liefern eine exakte Beschreibung der Erzeugung und Verbreitung von Aktionspotentialen.

Künstliche neuronale Netze modellieren nun die biologischen auf einem gewissen Abstraktionsniveau. Dabei bleibt die Netzwerk-Topologie in der Regel unverändert[15]. Die informationsverarbeitenden Einheiten, die Neuronen, wurden schon 1943 in [McCulloch, Pitts 1943] definiert. Ein künstliches Neuron hat eine bestimmte Anzahl von gewichteten Eingängen, über die Signale aufgenommen werden können, und nur einen Ausgang. Das Neuron operiert zu diskreten Zeiteinheiten, so daß zum Zeitpunkt t ein Impuls ausgesendet wird, falls zum

[12] Exzitatorisch entspricht erregend und dem gegenüber inhibitorisch hemmend.
[13] Dies sind die sogenannten Ionen.
[14] Diese stellen sich als Stromimpulse dar.
[15] Die Möglichkeit, künstliche neuronale Netze auch dynamisch aufzubauen, besteht prinzipiell, wird aber, wegen der damit verbundenen Komplexheit, selten genutzt.

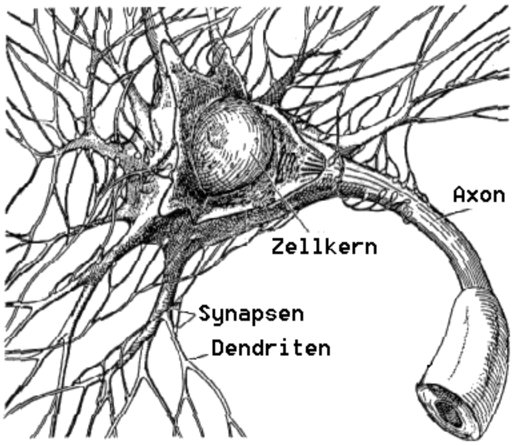

Abbildung 5.7: Aufbau eines Neurons (aus [Steven 1992])

Zeitpunkt $t-1$ der Schwellwert des Neurons überschritten wurde, d.h. genügend große Eingangssignale anlagen.

5.3.2 Das Backpropagation-Netz

Einen besonderen Netztyp stellen die Backpropagation-Netze dar. Diese versuchen, den Fehler zwischen gewünschter und tatsächlicher Ausgabe durch Änderung der Gewichte zu minimieren. Hierzu sei der Fehler $e_j(n)$ des j-ten Ausgabe-Neurons in der n-ten Iteration, wie folgt definiert:

$$e_j(n) = d_j(n) - y_j(n), \tag{5.20}$$

wobei $d_j(n)$ die gewünschte und $y_j(n)$ die tatsächliche Ausgabe für Neuron j liefern. Mit (5.20) läßt sich nun die Fehlerfunktion des gesamten Netzes angeben. Sei dazu $\mathcal{C}$ die Menge der Ausgabe-Neuronen:

$$\mathcal{E}(n) = \frac{1}{2} \cdot \sum_{j \in \mathcal{C}} e_j^2(n). \tag{5.21}$$

Der Backpropagation-Algorithmus weist nun jedem Gewicht $w_{ji}(n)$ zwischen zwei Neuronen j und i eine Korrektur $\Delta w_{ji}(n)$ zu, die zum Gradienten $\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$ proportional ist. Mit Hilfe der Kettenregel, kann dieser angegeben werden als:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathcal{E}(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \cdot \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \tag{5.22}$$

Hierbei stellt

$$v_j(n) = \sum_{i=0}^{P} w_{ji}(n) \cdot y_i(n), \tag{5.23}$$

die gewichtete Summe der P Eingänge dar, die als Argument der Aktivierungsfunktion Ψ_j übergeben wird, das heißt es gilt $\Psi_j(v_j(n)) = y_j(n)$. Werden obige Ausdrücke differenziert, ergibt sich somit:

$$\frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = e_j(n) \cdot (-1) \cdot \frac{\partial \Psi_j(v_j(n))}{\partial v_j(n)} \cdot y_i(n). \tag{5.24}$$

Die Korrektur $\Delta w_{ji}(n)$, die auf das Gewicht $w_{ji}(n)$ addiert wird, ist durch die *Delta-Regel* definiert:

$$\Delta w_{ji}(n) = -\eta \cdot \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)} = \eta \cdot \delta_j(n) \cdot y_i(n), \tag{5.25}$$

dabei fließt η als Lernrate in den Ausdruck ein. Für die Aktivierungsfunktion in (5.24) existieren verschiedene Ansätze. So kann diese als Schwellwert-, stückweise lineare oder auch als sigmoide Funktion modelliert werden. Letztere hat sich gegenüber den anderen Funktionen durchgesetzt und kann beispielsweise als Tangens Hyperbolicus angegeben werden:

$$\Psi(v) = \tanh\left(\frac{v}{2}\right). \tag{5.26}$$

Teil II

Driftkompensation

Kapitel 6

Motivation

Um einen Roboter auf einem gewünschten Weg halten zu können, ist es notwendig, zu jedem Zeitpunkt seine genaue Position zu wissen. Da die von der Odometrie gelieferten Daten aber mit zunehmender Zeit durch Drift und Schlupf unweigerlich ungenauer werden, müssen sie in einer geeigneten Art und Weise korrigiert werden. Während die Korrektur an Hand eines Fehlermodells, wie in [Wang 1988] beschrieben, keine Daten der unmittelbaren Umgebung nutzt, hat sich durchgesetzt, den Sensoreindruck zur Lokalisation heranzuziehen. Lokale Karten spielen in diesem Zusammenhang eine zentrale Rolle. Sie werden durch Sensoren, die am Roboter befestigt sind, aufgenommen und spiegeln die unmittelbare Umgebung des Roboters wieder. Mit Hilfe dieser Karten und einer dem System vorgegebenen globalen Karte ist es möglich, die genaue Position des Roboters zu ermitteln.

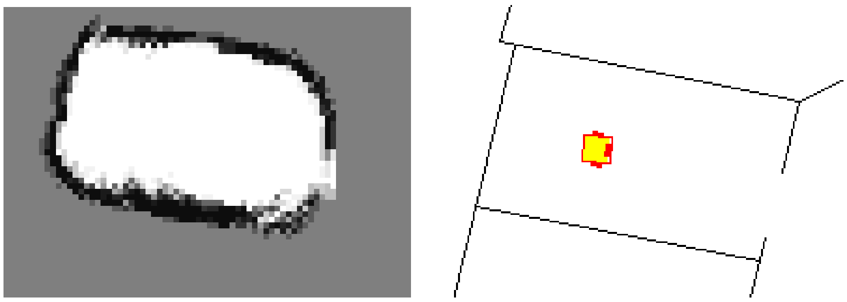

Abbildung 6.1: Das Matching-Problem

Hierbei läuft es letztendlich immer auf ein geeignetes Matching-Verfahren, welches Sensordaten mit Kartendaten vergleicht, hinaus. Wie in Abbildung 6.1 zu sehen ist, liegt dem System eine lokale Karte vor (in der Abbildung links), die aus den Sensoreindrücken erstellt wird. Diese muß mittels geeigneter Verfahren mit der globalen Karte (in der Abbildung rechts) korreliert werden. Da eine ungefähre Positions-Schätzung vorliegt, wird der Suchraum auf eine verhältnismäßig kleine Umgebung eingeschränkt.

Zur Bestimmung der genauen Roboter-Position existieren mehrere Ansätze, auf die in den folgenden Abschnitten näher eingegangen wird. Dabei konnte bei der geradenbasierten Driftkompensation durch die Hinzunahme eines zusätzlichen Kriteriums[1] die Eigenschaften des Verfahrens wesentlich verbessert werden. Auch die Hough-Transformation, die im Bereich der Lokalisation eher unüblich ist, zeigt sich in Räumen als robuste Methode zur Positions-Bestimmung. Jedes dieser Verfahren ist lediglich zur Korrektur kleinerer Positionsfehler geeignet, d.h. der zurückgelegte Weg zwischen zwei Lokalisationsphasen darf einen bestimmten Grenzwert nicht überschreiten. Abbildung 6.2 stellt die allgemeine Struktur der Driftvektor-Bestimmung dar.

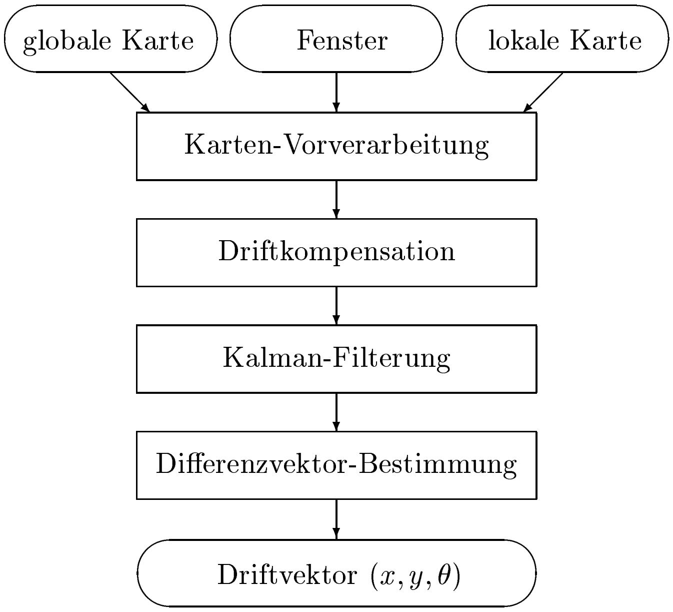

Abbildung 6.2: Allgemeine Struktur der Driftvektor-Bestimmung

Die Driftkompensation erhält als Eingangswerte die beiden vorverarbeiteten Karten und den zu betrachtenden Fensterausschnitt[2]. Als Ergebnis liefert sie die, um die Drift bereinigte, Position. Diese wird anschließend mittels eines Kalman-Filters zu einer neuen Schätzung der aktuellen Position herangezogen. Abschließend wird der Korrekturvektor (x, y, θ) aus der odometrischen und der berechneten Position gebildet. Die odometrische Position ist dabei als zusätzliche Information in der lokalen Karte enthalten. Steht der Driftvektor zur Verfügung, so kann dieser für die erneute Positions-Bestimmung verwendet werden, indem die odometrischen Daten um den Driftvektor bereinigt werden. Dies ist auch unbe-

[1] der Abstand zum Linien-Segment-Mittelpunkt
[2] Dieser wird unverändert von der Vorverarbeitungsebene übernommen.

dingt erforderlich, da, wie oben schon erwähnt, die Driftkompensations-Ansätze nur für kleinere Positionskorrekturen geeignet sind.

Kapitel 7

Karten-Vorverarbeitung

Um das Datenaufkommen zu verringern, empfiehlt sich die Ausdünnung der globalen Karte, d.h. die Linienstärke wird auf ein Pixel reduziert. Ein Operator, der dies leistet, wird in [Haralick, Shapiro 1992] beschrieben. Dieser *Ausdünnungs-Operator* $\mathcal{O}_4$ läßt sich als Konkatenation der folgenden drei Operatoren darstellen:

- Markierung von Rand- und inneren Punkten ($\mathcal{O}_1$)
- Bestimmung von Paarbeziehungen ($\mathcal{O}_2$)
- Schrumpfen zusammenhängender Regionen ($\mathcal{O}_3$)

Als Ausgangsbasis für die Anwendung der Operatoren dient die binarisierte Karte, d.h. Konturpunkte besitzen der Wert eins, Hintergrundpunkte dagegen den Wert null. In den folgenden Abschnitten seien die Nachbarschafts-Beziehungen gemäß Abbildung 7.1 festgelegt, wobei x_0 den Aufpunkt darstellt.

x_7	x_2	x_6
x_3	x_0	x_1
x_8	x_4	x_5

Abbildung 7.1: Nachbarschaft-Beziehungen

Die ausgedünnte Karte $\mathcal{M}'$ ergibt sich somit aus der ursprünglichen Karte $\mathcal{M}$ durch die Anwendung folgender Operationen:

$$\mathcal{M}' = \mathcal{O}_3(\mathcal{O}_2(\mathcal{O}_1(\mathcal{M}))). \tag{7.1}$$

Der Operator $\mathcal{O}_1$ unterteilt alle Konturpunkte in innere (Markierung i) und restliche Punkte (Markierung b). Die Menge der mit b markierten Punkte wird anschließend weiter verfeinert. Nur Punkte, die eine ausreichend große Anzahl von inneren Nachbarn besitzen, werden mit der Markierung p versehen. Alle anderen Konturpunkte erhalten den Wert eins. Zuletzt entscheidet $\mathcal{O}_3$ auf der Basis der mit p markierten Punkte, ob ein spezieller Punkt dem Hintergrund zugewiesen wird oder den Wert eins erhält.

7.1 Der Operator $\mathcal{O}_1$

Der Operator $\mathcal{O}_1$ markiert alle inneren Punkte mit i und alle äußeren sowie Randpunkte mit b. Dabei werden aus Effizienzgründen nur Konturpunkte betrachtet[1]. Grundsätzlich basiert der Operator auf den zwei Funktionen $\mathcal{R}$ und $\mathcal{Q}$:

$$\mathcal{R}(u,v) = \begin{cases} u & falls \ \ u = v \\ b & falls \ \ u \neq v \end{cases}, \tag{7.2}$$

$$\mathcal{Q}(u) = \begin{cases} b & falls \ \ u = b \\ i & falls \ \ u \neq b \end{cases}. \tag{7.3}$$

Für die Markierung eines bestimmten Pixels x_0 wird die Ausgabe von $\mathcal{Q}(a_n)$ herangezogen, wobei $a_m = \mathcal{R}(a_{m-1}, x_m)$ mit $a_0 = x_0$ gilt. Entsprechend der Nachbarschaft wird $n \in \{4, 8\}$ festgelegt, d.h. es gilt $m \in \{1, \ldots, 4\}$ bzw. $m \in \{1, \ldots, 8\}$.

Bemerkung 7.1 *Die Funktion $\mathcal{Q}$ stellt sicher, daß jeder Konturpunkt einen der beiden Werte $\mathfrak{i}$ bzw. $\mathfrak{b}$ zugewiesen bekommt und ist nicht äquivalent zur Identität.*

7.2 Der Operator $\mathcal{O}_2$

Auch dieser Operator basiert auf zwei einfachen Funktionen, $\mathcal{T}$ und $\mathcal{X}$, und betrachtet lediglich Vordergrundpunkte. Die Funktionen markieren einen Punkt x_0 mit p, falls dieser die bestimmte Markierung b und mindestens ϑ Nachbarn mit der Markierung i besitzt. Wie üblich gibt $n \in \{4, 8\}$ die betrachtete Nachbarschaft an:

$$\mathcal{T}(u,i) = \begin{cases} 1 & falls \ \ u = i \\ 0 & sonst \end{cases}, \tag{7.4}$$

$$\mathcal{X}(x_0) = \begin{cases} 1 & falls \ \ \sum_{i=1}^{n} \mathcal{T}(x_i, i) < \vartheta \ oder \ x_0 \neq b \\ p & falls \ \ \sum_{i=1}^{n} \mathcal{T}(x_i, i) \geq \vartheta \ und \ x_0 = b \end{cases}. \tag{7.5}$$

Nach Abschluß der Operation sind alle Konturpunkte entweder mit dem Wert eins belegt, d.h. diese Punkte werden nicht weiter modifiziert, oder mit dem ausgezeichneten Buchstaben p markiert.

[1] Die Anzahl der Hintergrundpunkte ist i.A. wesentlich größer als die der Konturpunkte.

7.3 Der Operator $\mathcal{O}_3$

Zuletzt werden zusammenhängende Bereiche geschrumpft, ohne einfach nur die Randpunkte zu entfernen. Die Randpunkt-Elimination führt im allgemeinen zu einer lückenhaften Kontur, d.h. es geht Information verloren oder falsche Information entsteht[2]. Um dies auszuschließen, werden nur die zuvor mittels des Operators $\mathcal{O}_2$ mit p markierten Punkte betrachtet. Hierfür werden die zwei Funktionen $\mathcal{Y}$ und $\mathcal{B}$ eingeführt, wobei $\mathcal{Y}$ jeweils für die 4-Nachbarschaft bzw. für die 8-Nachbarschaft definiert ist:

$$\mathcal{Y}_4(t,u,v,w) = \begin{cases} 1 & falls \;\; t = u \; und \; (v \neq t \; oder \; w \neq t) \\ 0 & sonst \end{cases} , \tag{7.6}$$

$$\mathcal{Y}_8(t,u,v,w) = \begin{cases} 1 & falls \;\; t \neq u \; und \; (v = t \; oder \; w = t) \\ 0 & sonst \end{cases} , \tag{7.7}$$

$$\mathcal{B}(a_1,a_2,a_3,a_4,x_0) = \begin{cases} 0 & falls \;\; \sum_{i=1}^4 a_i = 1 \\ 1 & sonst \end{cases} . \tag{7.8}$$

Wird die Nachbarschafts-Beziehung aus Abbildung 7.1 zu Grunde gelegt, so ergeben sich für die Variablen a_1 bis a_4 folgende Ausdrücke. Sei dazu $i \in \{4,8\}$:

$$a_1 = \mathcal{Y}_i(x_0,x_1,x_6,x_2), \tag{7.9}$$

$$a_2 = \mathcal{Y}_i(x_0,x_2,x_7,x_3), \tag{7.10}$$

$$a_3 = \mathcal{Y}_i(x_0,x_3,x_8,x_4) \quad \text{und} \tag{7.11}$$

$$a_4 = \mathcal{Y}_i(x_0,x_4,x_5,x_1). \tag{7.12}$$

Die Funktion $\mathcal{B}$ ordnet letztendlich die mit der Markierung p versehenen Punkte dem Hintergrund bzw. der Kontur zu.

[2]Ursprünglich nicht vorhandene Türöffnungen kommen hinzu.

Kapitel 8

Der Hough-basierte Ansatz

Das erste der hier vorgestellten Verfahren zur Driftkompensation basiert auf der erweiterten Hough-Transformation, wie sie in [Haberäcker 1995] beschrieben ist. Da diese weder rotations- noch größeninvariant ist, müssen die lokalen Karten[1] in diskreten Rotationsstufen transformiert werden. Die fehlende Größeninvarianz ist zu vernachlässigen, da die globale und die lokale Karte in derselben Skalierung vorliegen.

Zur Lokalisation des Roboters wird ein Fensterausschnitt der globalen Karte um die vermutete Position gewählt. Ein Akkumulatorfeld der gleichen Dimension wird anfangs mit null initialisiert. Dieses gibt nach Abschluß des Verfahrens Aufschluß über die richtige Position des Roboters. Durch die Transformation in diskreten Rotationsstufen ergibt sich für jede untersuchte Orientierung ein Akkumulatorfeld. Dies führt zur Notwendigkeit eines Gütemaßes, um letztendlich das richtige Akkumulatorfeld und somit die korrekte Position feststellen zu können.

8.1 Datenfluß

Abbildung 8.1 zeigt das Datenfluß-Diagramm für den Hough-basierten Ansatz, wobei *ODO* die odometrischen Daten des Systems repräsentiert. Es gehen folgende Eingangsgrößen in das Verfahren ein:

1. Die als Bitmap vorliegende globale Karte
2. Die lokale Karte (ebenfalls pixelorientiert)
3. Einen Fensterausschnitt

Der Fensterausschnitt schränkt den Suchraum, sowohl in der globalen als auch in der lokalen Karte, erheblich ein und erlaubt so den sinnvollen Einsatz der

[1]Eine lokale Karte wird in diesem Zusammenhang aus den über die Zeit integrierten Sensoreindrücken gebildet (siehe Kapitel 5.2).

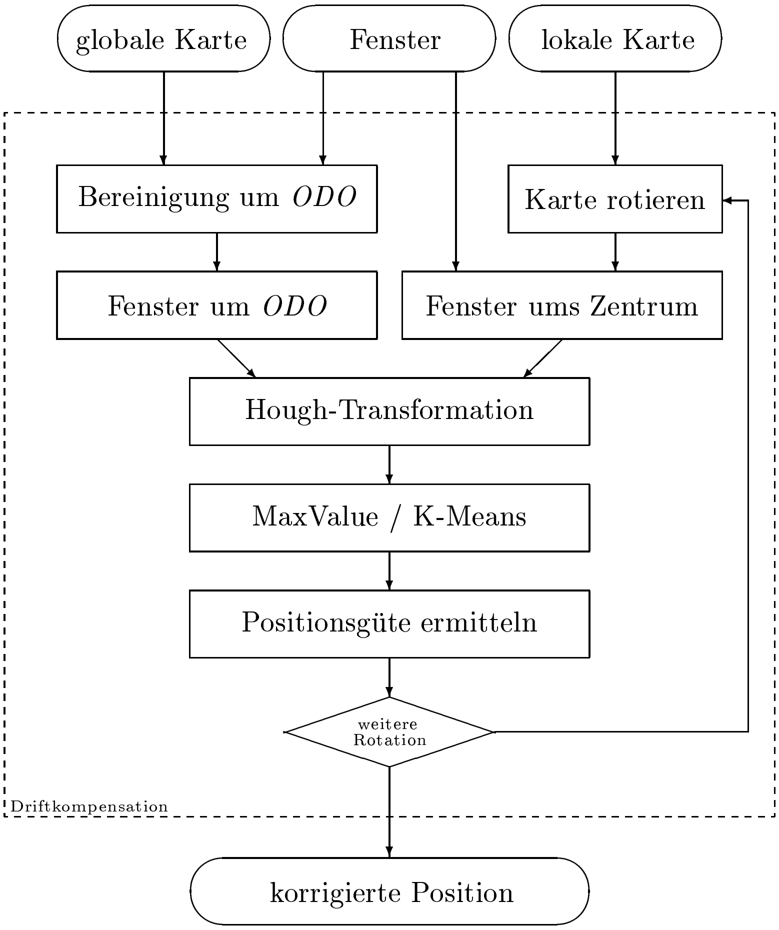

Abbildung 8.1: Datenfluß-Diagramm der Hough-basierten Lokalisation

Hough-Transformation. Dies ist möglich, da eine ungenaue Schätzung der Position vorliegt und lediglich die Drift innerhalb kurzer Zeitabstände berechnet werden soll. Nachdem die odometrischen Positionsdaten in Kartenkoordinaten transformiert worden sind, kann ein Fensterausschnitt in der globalen Karte, um die vermutete Position, gewählt werden. In der lokalen Karte wird ebenfalls ein Fensterausschnitt um das Zentrum[2] bestimmt. Beide Fensterausschnitte werden anschließend mittels der Hough-Transformation verglichen. Als Ergebnis liefert diese ein sogenanntes Akkumulatorfeld, das Aufschluß über die derzeitige Position gibt. Für eine Untersuchung dieses Feldes bieten sich zum einen

[2] Der Roboter befindet sich in der lokalen Karte stets im Zentrum.

Clusteranalyse-Verfahren und zum anderen Maximums-Strategien an. Da die Hough-Transformation nicht rotationsinvariant ist, muß für die jeweilige gefundene Position ein Gütemaß bestimmt werden, das für die gesuchte Orientierung einen Extremwert annimmt. Abschließend werden obige Schritte solange wiederholt, bis alle Rotationsstufen durchlaufen sind. Als Ergebnis wird die korrigierte Position wieder ausgegeben.

8.2 Die Hough-Transformation

Die Hough-Transformation wurde ursprünglich zur Detektion von Geraden und Kurven in Binärbildern verwendet (siehe [Duda, Hart 1972] und [Hough 1962]). Durch die in [Merlin, Farber 1975] vorgeschlagene Erweiterung können als Referenzstruktur nicht nur Geraden, sondern auch Segmente verwendet werden. Im Fall der Driftkompensation stellen diese Segmente die lokale Karte dar. Voraussetzung ist ein Binärbild[3], in dem das gesuchte Segment[4] enthalten ist. Dabei darf das vorgegebene Segment verrauscht oder unvollständig sein. Der Algorithmus kann durch die folgenden fünf Schritte skizziert werden ([Merlin, Farber 1975]). Eine genaue Beschreibung wird anschließend gegeben.

1. Festlegung eines Bezugspunktes $\mathcal{A}$ im Segment $\mathcal{S}$
2. Rotation des Segmentes $\mathcal{S}$ um 180° zu einem Segment $\mathcal{S}'$
3. Zuordnung von $\mathcal{S}'$ ausgehend von $\mathcal{A}$ für jeden Bildpunkt
4. Bestimmung des Punktes mit der größten Zuordnungsrate
5. Gesuchte Position des Bezugspunktes $\mathcal{A}$ ist die in Schritt 4 gefundene

Zunächst muß der Bezugspunkt des Segmentes festgelegt werden. Hier bietet sich das Kartenzentrum an, da dieses die vermutete Roboter-Position darstellt. Danach werden alle Bildpunkte der globalen Karte, die innerhalb eines Fensters um diese Position liegen, untersucht. Jeder Vordergrundpunkt könnte einer des gesuchtes Segmentes sein. Trifft das zu, liegt der oben gewählte Bezugspunkt in einer bestimmten relativen Lage zu dem betrachteten Punkt. Wird dies für alle Segmentpunkte durchgeführt, ergibt sich für jeden Vordergrundpunkt eine Menge von möglichen Bezugspunkt-Positionen, die punktsymmetrisch zum gesuchten Segment liegen. Das Akkumulatorfeld, das dieselbe Dimension wie das Fenster hat, und genau darunter liegt, wird nun jeweils an den entsprechenden Stellen um eins inkrementiert.

Der folgende Pseudocode soll das Verfahren verdeutlichen. Aus Gründen der Übersichtlichkeit wird auf notwendige Gültigkeits-Überprüfungen der einzelnen

[3] die globale Karte
[4] die lokale Karte

Indizes verzichtet. Außerdem werden die Schritte 4 und 5 mit Hilfe von Clusteranalyseverfahren, wie in den Kapiteln 8.3 und 8.5 beschrieben, gelöst und sind deshalb nicht im Code enthalten.

Variablen:

$fenster$	Fenstergröße
odo_x, odo_y	vermutete Position
$gmap$	globale Karte
$lmap$	lokale Karte
lw	Breite der lokalen Karte
lh	Höhe der lokalen Karte
$Akkumulator$	Akkumulatorfeld
x, y, i, j	Laufvariablen

Code:

```
for every x ∈ {−fenster/2, ..., fenster/2} do
    for every y ∈ {−fenster/2, ..., fenster/2} do
        if gmap(odo_x + x, odo_y + y) = 1 then
            for every i ∈ {0, ..., lw} do
                for every j ∈ {0, ..., lh} do
                    if lmap(lw − i, lh − j) = 1 then
                        Akkumulator(x − lw/2 + i, y − lh/2 + j) += 1
                    endif
                endfor
            endfor
        endif
    endfor
endfor
```

Bei der Auswertung des Akkumulatorfeldes ist es unter Umständen nicht ausreichend, das globale Maximum heranzuziehen, da vereinzelte Ausreißer, die nicht repräsentativ für die gesuchte Position sind, existieren können. Es bietet sich deshalb ein Clusteranalyse-Verfahren, wie das in Kapitel 8.3 beschriebene K-Means-Verfahren, an.

8.3 K-Means-Clusteranalyse

Der K-Means-Ansatz läßt sich, wie in [Tou, Gonzalez 1974] beschrieben, auf die Minimierung eines Performance-Index zurückführen. Ein in diesem Zusammenhang häufig verwendeter Index ist die Summe des quadratischen Fehlers:

$$\mathcal{I} = \sum_{j=1}^{K} \sum_{x \in \mathcal{Z}_j} \|x - m_j\|^2. \tag{8.1}$$

K stellt die Anzahl der Cluster, $\mathcal{Z}_j$ die Punktemenge des j-ten Clusters und m_j das j-te Clusterzentrum dar. Dieses läßt sich als Schwerpunkt[5] aller N_j Punkte des j-ten Clusters bestimmen:

$$m_j = \frac{1}{N_j} \sum_{x \in \mathcal{Z}_j} x. \tag{8.2}$$

Ausdruck (8.1) summiert die Abstände zwischen jedem einzelnen Punkt des Clusters zu dessen Zentrum auf. Der resultierende Gesamtabstand wird auch als Fehler bezeichnet, den es zu minimieren gilt. Vier Schritte, die iterativ durchlaufen werden müssen, sind hierzu notwendig:

1. Anfangs werden K Clusterzentren bestimmt. Diese seien mit z_j, $j \in \mathcal{S}$, durchnummeriert, wobei $\mathcal{S} = \{1, 2, \ldots, K\}$ die Menge der Clusterindizes ist. Die Menge $\mathcal{Z}_j$ sei die Punktmenge des Clusters j, deren Mächtigkeit N_j ist.

2. Im k-ten Iterationsschritt werden alle positiven Akkumulatorpunkte x auf die K Cluster unter Verwendung folgender Relation verteilt:

$$(x \in \mathcal{Z}_j(k)) \Longleftrightarrow (\forall i \in \mathcal{S}: \quad (i \neq j) \Rightarrow \|x - z_j(k)\| < \|x - z_i(k)\|). \tag{8.3}$$

3. Die neuen Clusterzentren $z_j(k+1)$, $j \in \mathcal{S}$, müssen so berechnet werden, daß der Performance-Index minimiert wird:

$$\mathcal{I}_j = \sum_{x \in \mathcal{Z}_j(k)} \|x - z_j(k+1)\|^2. \tag{8.4}$$

Clusterzentren, die dies erfüllen, stellen den Schwerpunkt der clusterzugehörigen Punkte dar:

$$\forall j \in \mathcal{S}: \quad z_j(k+1) = \frac{1}{N_j} \sum_{x \in \mathcal{Z}_j(k)} x. \tag{8.5}$$

4. Solange noch Zentrumsverschiebungen stattfinden, d.h. es existiert mindestens ein Clusterzentrum, für das $z_j(k+1) \neq z_j(k)$ ist, muß mit Schritt 2 fortgefahren werden. Da der Algorithmus konvergiert, gibt es nach endlich vielen Schleifendurchläufen keine Zentrumsverschiebungen mehr und das Verfahren kann abgebrochen werden.

[5] Mittelwert der einzelnen Komponenten der Clusterpunkte

Das oben angegebene Verfahren stellt eine gute Lösung zur Verfügung für Cluster, deren zugehörige Punkte die gleiche Wertigkeit besitzen. Es gibt allerdings auch Fälle, in denen die einzelnen Clusterpunkte eine verschiedene Wertigkeit haben. Hier ist es sinnvoll, die Clusterzentren entsprechend der Wertigkeit zu verschieben. Abbildung 8.2 verdeutlicht diesen Zusammenhang. Während beim

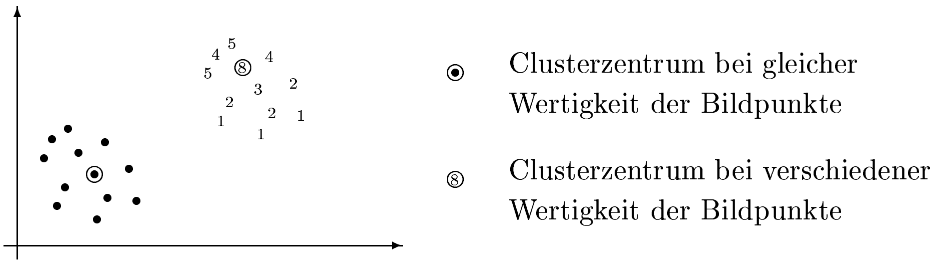

Abbildung 8.2: Clusterzentrums-Bestimmung

linken Cluster alle Bildpunkte die gleiche Wertigkeit besitzen und somit das Clusterzentrum als komponentenweiser Mittelwert aller Punkte berechnet werden kann, haben die Punkte im Rechten eine verschiedene Wertigkeit. Es ist also sinnvoll, Bildpunkte, die einen höheren Wert aufweisen, stärker in der Zentrums-Bestimmung zu berücksichtigen. Ausdruck (8.5) muß deshalb angepaßt werden. Sei dazu $\mathcal{V}(x)$ der Wert des Clusterpunktes x. Die Clusterzentren $z_j(k+1)$, $j \in \mathcal{S}$, lassen sich dann durch

$$z_j(k+1) = \frac{1}{\sum_{x \in \mathcal{Z}_j(k)} \mathcal{V}(x)} \sum_{x \in \mathcal{Z}_j(k)} x \cdot \mathcal{V}(x) \tag{8.6}$$

berechnen. Der Performance-Index in Ausdruck (8.4) ergibt sich dann zu

$$\mathcal{I}_j = \sum_{x \in \mathcal{Z}_j(k)} (\mathcal{V}(x) \cdot \|x - z_j(k+1)\|)^2. \tag{8.7}$$

Der folgende Pseudocode löst das K-Means-Clusterverfahren für ein zweidimensionales Feld, in dem die einzelnen Punkte verschiedene Wertigkeiten besitzen.

Variablen:

$fenster$	Fenstergröße
$Akkumulator$	Akkumulatorfeld
z_j	$j-tes$ Clusterzentrum
z_{j^*}	Clusterzentrum mit geringstem Abstand zum Bildpunkt
z_j^x	x-Komponente des Clusterzentrums z_j
z_j^y	y-Komponente des Clusterzentrums z_j
z_j^w	Gewichts-Komponente des Clusterzentrums z_j

$\mathcal{S}$	Menge der Clusterindizes
$\mathcal{C}$	Menge der Clusterzentren
$\mathcal{C}_{old}$	alte Menge der Clusterzentren
$\mathcal{C}_{new}$	neu gebildete Menge der Clusterzentren
$\mathcal{Z}_j$	Menge der zum Cluster j gehörigen Punkte
$Buffer_1, Buffer_2$	Zwischenspeicher
x, y, i, j	Laufvariablen

Code:

$\mathcal{C} := \bigcup_j z_j$
<u>repeat</u>
    $\mathcal{C}_{old} := \mathcal{C}$
    <u>for</u> every $x \in \{-\frac{fenster}{2}, \ldots, \frac{fenster}{2}\}$ <u>do</u>
        <u>for</u> every $y \in \{-\frac{fenster}{2}, \ldots, \frac{fenster}{2}\}$ <u>do</u>
            <u>if</u> $Akkumulator(x, y) > 0$ <u>then</u>
                $j^* \in \mathcal{S}$
                <u>for</u> all $z_j \in \mathcal{C}$ <u>do</u>
                    $Buffer_1 = Akkumulator(x, y) \cdot \|(x, y) - z_{j*}\|$
                    $Buffer_2 = Akkumulator(x, y) \cdot \|(x, y) - z_j\|$
                    <u>if</u> $Buffer_1 > Buffer_2$ <u>then</u>
                        $j^* := j$
                    <u>endif</u>
                <u>endfor</u>
                $\mathcal{Z}_{j^*} := \mathcal{Z}_{j^*} \cup (x, y)$
            <u>endif</u>
        <u>endfor</u>
    <u>endfor</u>
    $\mathcal{C}_{new} := \emptyset$
    <u>for</u> all $z_j \in \mathcal{C}$ <u>do</u>
        $z_j := 0$
        <u>for</u> every $(i, j) \in \mathcal{Z}_j$ <u>do</u>
            $z_j^x := z_j^x + i \cdot Akkumulator(i, j)$
            $z_j^y := z_j^y + j \cdot Akkumulator(i, j)$
            $z_j^w := z_j^w + Akkumulator(i, j)$
        <u>endfor</u>
        $z_j^x := \frac{z_j^x}{z_j^w}$
        $z_j^y := \frac{z_j^y}{z_j^w}$
        $\mathcal{C}_{new} := \mathcal{C}_{new} \cup z_j$
    <u>endfor</u>
    $\mathcal{C} := \mathcal{C}_{new}$
<u>until</u> $\mathcal{C}_{old} = \mathcal{C}_{new}$

Nach Abschluß der Clusteranalyse sind die vorgegebenen Zentren auf die einzelnen Cluster verteilt und spiegeln deren Schwerpunkte wieder. Ziel muß es nun sein, das zum größten Cluster zugehörige Zentrum zu bestimmen, wobei die Größe sich auf die Anzahl der clusterzugehörigen Punkte und deren Wertigkeit bezieht ($\sum_{x \in \mathcal{Z}_j} \mathcal{V}(x)$). Da mehrere Cluster etwa dieselbe Größe haben können, spielt die Verteilung der Clusterpunkte eine entscheidende Rolle. Dies führt zu der in Kapitel 8.6 zu bestimmenden Varianz.

Bemerkung 8.1 *Der Name K-Means liegt in der schrittweisen Berechnung der Clusterzentren begründet [Tou, Gonzalez 1974].*

8.4 Nachteile des Cluster-Verfahrens

Ein Nachteil des Cluster-Verfahrens von [Tou, Gonzalez 1974] ist die unter Umständen fehlerhafte Bestimmung der Varianz[6]. Dies führt zu einer falschen Kalman--Filterung[7] und somit letztendlich zu einer inkorrekten Position. Dies tritt bei ausgeprägten Richtungen im Akkumulatorfeld, wie sie üblicherweise in Gängen oder länglichen Räumen vorkommen (siehe Abbildung 8.7), am häufigsten auf. Abbildung 8.3 zeigt diesen Zusammenhang. Links ist das Akkumulatorfeld dar-

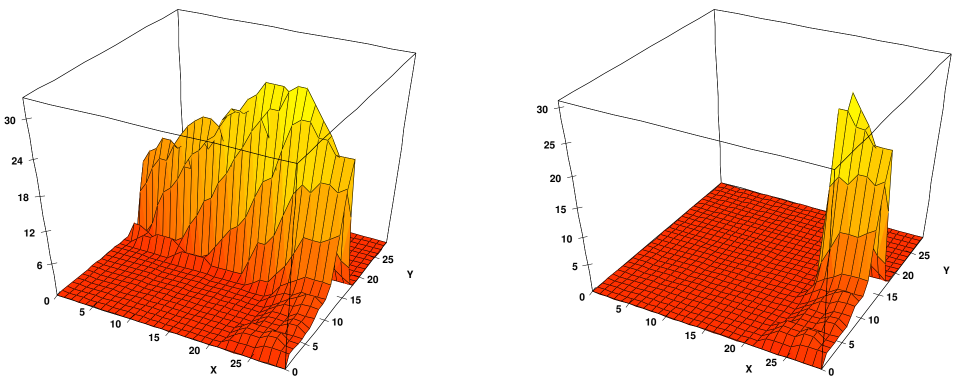

Abbildung 8.3: Fehlerhafte Clusterbildung

gestellt, rechts der Cluster, welches zur Positionskorrektur herangezogen wird. Es läßt sich deutlich erkennen, daß der Cluster nicht repräsentativ, für die im Akkumulator enthaltenen Daten, ist. Die somit ermittelte Varianz ist folglich falsch. Eine Verschmelzung der Cluster kann diesem Problem entgegenwirken, löst es aber

[6] siehe Abschnitt 8.6

[7] siehe Kapitel 11

nicht. Zur Verschmelzung zweier Cluster werden in einer Vierer-Nachbarschaft die Randpunkte des Clusters betrachtet. Beträgt die Wertigkeit dieser mehr als ein Viertel des Zentrums-Wertes, so wird der angrenzende Cluster hinzugenommen. Bezogen auf die in Abbildung 8.3 dargestellte Situation, würde dies eine Hinzunahme eines in etwa gleich großen Clusters bedeuten[8]. Wird die Clusterverschmelzung fortgesetzt, entsteht am Ende ein, aus den einzelnen Clustern gebildeter, großer Cluster. Dies führt zu einer verbesserten Varianz-Bestimmung, aber das ursprünglich gewählte Clusterzentrum kann noch falsch sein. Es bedarf also ein Verfahren, welches Varianz und Clusterzentrum so bestimmt, daß diese dem vorliegendem Akkumulatorfeld entsprechen.

8.5 Max-Value-Strategie

Es hat sich gezeigt, daß große Werte im Akkumulatorfeld immer eine signifikante Umgebung besitzen. Hierauf beruht die *Max-Value*-Strategie. Ausgehend vom größten Akkumulator-Wert, welcher das gesuchte Zentrum darstellt, wird ein Flood-Fill ausgeführt. Dabei gehören alle Punkte im alten Akkumulatorfeld zum neu gebildeten, sobald ihr Wert ein Viertel des zuvor gewählten Zentrum-Wertes übersteigt. Dies gewährleistet, daß

1. isolierte Peaks im Akkumulatorfeld entfernt werden,
2. das Akkumulatorfeld um das Rauschen bereinigt wird.

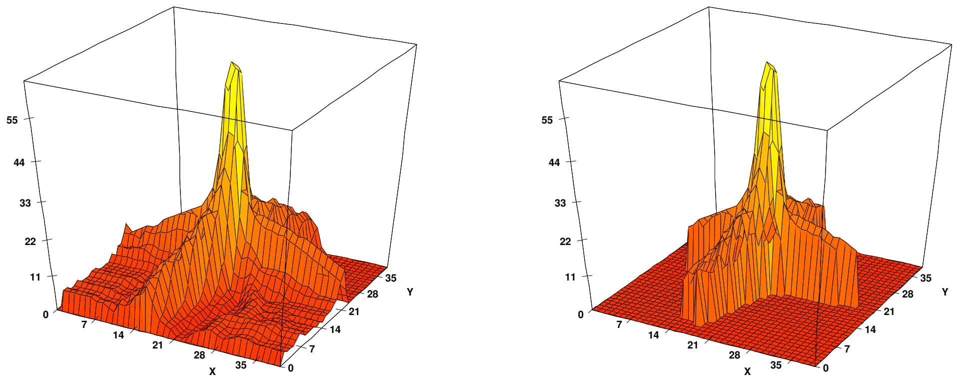

Abbildung 8.4: Rauschbeseitigung durch Flood-Fill

Abbildung 8.4 verdeutlicht dies. Während links das Akkumulatorfeld noch durch Rauschen gestört ist, kann rechts durch den vorangegangen Flood-Fill die Varianz

[8] Alle fünf Clusterzentren liegen auf einer Linie.

richtig berechnet werden. Im folgenden Pseudo-Code wird der, für die *Max-Value*-Strategie verwendete, Flood-Fill beschrieben. Die Funktion NEW erzeuge dabei ein Listenelement mit der gegebenen Struktur.

Variablen:

$Akkumulator$	Ausgangs-Akkumulatorfeld
$Akkumulator2$	Ziel-Akkumulatorfeld
max_x, max_y	größter Akkumulatorwert
nb_x, nb_y	unmittelbarer Nachbar des betrachteten Punktes
$list$	Liste noch abzuarbeitender Punkte
neu	neu zu bearbeitender Punkt
$start$	betrachteter Punkt
ϑ	Schwellwert für die Zugehörigkeit

Code:

```
NEW(list)
list.x = max_x
list.y = max.y
list.next = NULL
start = list
while start ≠ NULL do
    for each neighbor nb of start do
        if Akkumulator(nb_x, nb_y) > ϑ then
            NEW(neu)
            neu.x = nb_x
            neu.y = nb_y
            neu.next = NULL
            list.next = neu
            list = neu
            Akkumulator2(nb_x, nb_y) = Akkumulator(nb_x, nb_y)
        endif
    endfor
    start = start.next
endwhile
```

8.6 Bestimmung der Varianz

Für eine Auswertung des K-Means- bzw. des *Max-Value*-Ergebnisses ist die Verteilung innerhalb der Cluster von großer Bedeutung[9]. Diese gibt die mittlere

[9] Bei der *Max-Value*-Strategie existiert nur ein Cluster.

Entfernung eines Clusterpunktes zum Zentrum an. Es ist daher sinnvoll, für jedes Cluster j mit der Punktmenge $\mathcal{Z}_j$ ein Wahrscheinlichkeitsmaß $\mathcal{P}_j$ einzuführen. Hierfür wird eine Funktion benötigt, die zu jedem Clusterpunkt dessen Wertigkeit ausgibt. Die in Kapitel 8.3 beschriebene Funktion $\mathcal{V}$ leistet dies. $\mathcal{P}_j$ läßt sich nun mittels folgenden Ausdrucks definieren:

$$\mathcal{P}_j : \wp(\mathcal{Z}_j) \longrightarrow [0,1]\ ,\ \ x \longmapsto \frac{\mathcal{V}(x)}{\sum_{x\in\mathcal{Z}_j} \mathcal{V}(x)}. \tag{8.8}$$

Durch die Erweiterung von $\mathcal{V}$ auf Argumentmengen $M \subseteq \mathcal{Z}_j$ mit Hilfe von $\mathcal{V}(M) = \sum_{x\in M} \mathcal{V}(x)$, genügt $\mathcal{P}_j$ offensichtlich den Kolmogorow-Axiomen.

Axiom 1:	$\mathcal{P}_j(A) \geq 0$	$A \in \wp(\mathcal{Z}_j)$
Axiom 2:	$\mathcal{P}_j(\Omega) = 1$	$\Omega = \mathcal{Z}_j$
Axiom 3:	$A \cap B = \emptyset \Rightarrow \mathcal{P}_j(A \cup B) = \mathcal{P}_j(A) + \mathcal{P}_j(B)$	$A, B \in \wp(\mathcal{Z}_j)$

Sei nun X eine Zufallsgröße, deren Wertebereich $\mathcal{Z}_j$ entspricht. Der Erwartungswert $\mathcal{E}_j(X)$, der bei endlichen Mengen auch häufig nur mit μ_j bezeichnet wird, läßt sich dann unmittelbar aus (8.6) ableiten ($\mathcal{S}$ sei die Cluster-Indexmenge):

$$\forall j \in \mathcal{S} :\ \ \mathcal{E}_j(X) = \sum_{x\in\mathcal{Z}_j} x \cdot \mathcal{P}_j(X = x). \tag{8.9}$$

Mit Hilfe des Erwartungswertes kann die Varianz

$$\forall j \in \mathcal{S} :\ \ Var_j(X) \ = \ \sum_{x\in\mathcal{Z}_j} (x - \mu_j)^2 \cdot \mathcal{P}_j(X = x) \ = \ \mathcal{E}_j((X - \mu_j)^2) \tag{8.10}$$

berechnet werden. Ist die Zufallsvariable X eindeutig gewählt, wird für die Varianz $Var_j(X)$ oft auch kurz nur σ_j^2 geschrieben. σ_j wird dann die Standardabweichung oder Streuung des j-ten Clusters genannt.

8.7 Gütekriterium der Position

Für die Positions-Schätzung liegt dem System für jede untersuchte Orientierung genau ein Akkumulatorfeld zu Grunde[10]. Zur Bestimmung der Position muß das Akkumulatorfeld gefunden werden, welches sowohl die Translation als auch die Rotation des Roboters korrekt wiederspiegelt. Während die Translation direkt durch das Clusterzentrum vorgegeben ist, fließt die Rotation implizit durch das verwendete Akkumulatorfeld ein. Dabei gibt die Standardabweichung σ Aufschluß über die Verteilung der Punkte innerhalb eines Clusters. Ein für die Position repräsentatives Zentrum z_j^* sollte die folgenden Eigenschaften erfüllen:

[10]Das nachfolgende Experiment hat für die Positions-Bestimmung eine Basis von zwanzig Akkumulatorfeldern (zwanzig Ein-Grad-Schritte).

- Der Wert des Clusterzentrums $\mathcal{V}(z_j^*)$ sollte möglichst groß sein.
- Die Standardabweichung σ_j sollte dagegen eher klein sein.

Es bietet sich also an, ein Gütekriterium

$$\forall j \in \mathcal{S}: \quad \mathcal{G}(\mathcal{Z}_j) = \beta_1 \cdot \theta + \beta_2 \cdot \sigma_j + \beta_3 \cdot \frac{1}{\mathcal{V}(z_j^*)} \tag{8.11}$$

einzuführen. β_1, β_2 und β_3 seien Skalierungsfaktoren, die eine Gewichtung der Komponenten vornehmen. Zunächst fließt der relative Winkel θ in das Gütemaß ein. Dieser gibt über die Orientierung des Roboters Aufschluß und ist direkt mit dem untersuchten Akkumulatorfeld verknüpft. Da im Kurzstrecken-Bereich die Odometrie relativ genau ist[11], werden somit stärkere Abweichungen von der aktuellen Orientierung bestraft. Die beiden folgenden Summanden sorgen für die Erfüllung der oben geforderten Eigenschaften. Gesucht ist nun das Clusterzentrum z_j, dessen Clusterpunkte $\mathcal{Z}_j$ für fest vorgegebenen β_1, β_2 und β_3 das Gütekriterium $\mathcal{G}$ minimieren:

$$z^* \in \mathcal{Z} \;\wedge\; \mathcal{G}^*(\mathcal{Z}) = \min\{\mathcal{G}(\mathcal{Z}_j) \mid j \in \mathcal{S}\}. \tag{8.12}$$

Für die Positions-Schätzung wird schließlich das mit Hilfe von (8.12) bestimmte Akkumulatorfeld verwendet. Die Translations- und Rotationskorrektur lassen sich zum einen direkt aus dem Akkumulatorfeld ablesen und zum anderen indirekt durch den assoziierten Winkel bestimmen.

Bemerkung 8.2 *Wird nur das Verhältnis zwischen $\mathcal{V}(z_j^*)$ und σ_j als Gütemaß verwendet, fließt die Wertigkeit des Clusterzentrums, die die allgemeine Güte der Messung angibt, nicht ausreichend ein.*

8.8 Neuronale Bestimmung der Positionsgüte

Neuronale Netze[12] stellen einen weiteren Ansatz zur Bestimmung der Positionsgüte dar. Dabei ist es nicht notwendig eine konkrete Gütefunktion zu ermitteln, sondern das Netz lernt vielmehr auf Basis von Beispielen die gewünschte Ausgabe zu produzieren. Es bieten sich hier insbesondere Multilayer-Perzeptron-Netze an. Diese besitzen gegenüber anderen Netzwerk-Architekturen folgende Vorteile.

- Zum einen kann die Aufgabe an Hand von Beispielen erlernt werden. Architekturen, die unüberwacht lernen[13], klassifizieren die Eingabe nach statistisch signifikanten Eigenschaften und produzieren somit nicht unbedingt die gewünschte Ausgabe.

[11] Zwischen zwei Lokalisations-Phasen können nur kurze Strecken zurückgelegt werden.
[12] siehe z.B. [Haykin 1994] oder auch [Köhle 1990]
[13] Als Beispiel sei hier auf Kohonen-Karten verwiesen.

- Zum anderen zeichnet Multilayer-Perzeptron-Netze die schnelle Fehlerminimierung aus, d.h. das Netz kann schnell an neue Situationen angepasst werden[14].

Das für die Bestimmung der Positionsgüte eingesetzte Multilayer-Perzeptron besteht aus einer Eingabeschicht von vier Neuronen, zehn verdeckten und einer Ausgabeschicht von ebenfalls zehn Neuronen. Jede Neuronenschicht ist mit der ihr nachfolgenden vollständig verbunden, wobei die Neuronen eine sigmoide Aktivierungsfunktion besitzen. Abbildung 8.5 zeigt diese Struktur.

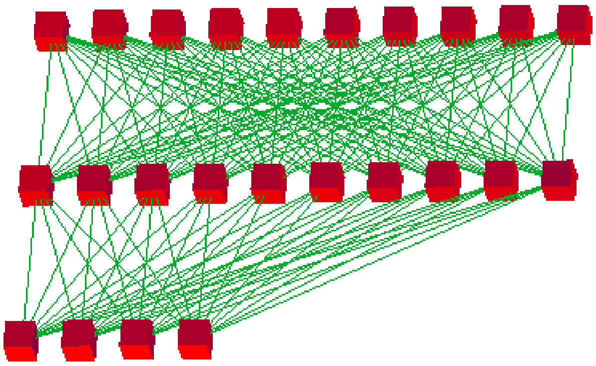

Abbildung 8.5: Die Netzwerk-Architektur

Das Netz erhält nun die folgenden vier Werte als Eingangsgrößen:

1. σ_x^2: Varianz der x-Komponente
2. σ_y^2: Varianz der y-Komponente
3. θ: relativer Winkel
4. *center*: Wertigkeit des Zentrums

Die Ausgabeschicht stellt eine lineare Repräsentation der Positionsgüte dar, d.h. die Ausgabe-Neuronen können direkt als Werteskalar interpretiert werden. Um zusätzlich Positions-Bewertungen abschwächen zu können, werden Aktivierungen sowohl von höher- als auch niederwertige Neuronen zugelassen[15], wobei niederwertige Neuronen bei der Güte-Bestimmung einen negativen Faktor besitzen. Es ergibt sich somit für die Ausgabeschicht $A = \{a_1, \dots, a_{10}\}$ das Gütemaß

$$\mathcal{G}'(A) = \sum_{a_i \in A} (i-5) * a_i. \tag{8.13}$$

[14]Netze mit langsamen Lernverhalten, wie die Boltzmann-Maschine, eignen sich hierfür nicht.

[15]Mit höherwertigen Neuronen sind die letzten fünf Neuronen gemeint, während die niederwertigen die ersten fünf Neuronen darstellen.

Abbildung 8.6: Die Ausgabeschicht des neuronalen Netzes

Abbildung 8.6 zeigt eine Netz-Ausgabe, in der sowohl das zweite als auch das achte Neuron eine den restlichen Neuronen gegenüber höhere Aktivierung aufweist[16].

Bemerkung 8.3 *Dem neuronalen Netz werden die Daten eines Akkumulatorfeldes als Eingabe präsentiert. Aus diesen generiert das Netz eine Güte, die mit allen anderen auf diese Weise ermittelten verglichen werden muß.*

Die Verteilung der Antwort auf zehn Neuronen bringt mehrere Vorteile mit sich. So läßt sich an Hand der Ausgabe-Aktivierung die Unterstützung des Netzes für bestimmte Antworten ermitteln. Bei einer hohen Aktivierung konnte das Netz mit einem großen Zuverlässigkeitsgrad die Antwort generieren. Ist hingegen die Ausgabe schwach, so ist die Unsicherheit des Systems hoch. Bei einem einzelnen Neuron stellt im Gegensatz hierzu schon jede Aktivierung eine konkrete Ausgabe dar.

Um nun Trainingsmuster generieren zu können, muß zu jedem Eingabe-Tupel ein Ausgabe-Vektor erzeugt werden. Dabei ist die Güte der vorliegenden Eingabe und damit das zugehörige Ausgabe-Neuron bekannt (Für das Experiment in Abschnitt 8.10 wurden fünfzig Trainingsmuster, die für den Eingaberaum repräsentativ sind, von Hand erstellt.). Wird dieses mit j bezeichnet, stellt sich die Frage, welche Aktivierung den restlichen Neuronen zuzuordnen ist. Dean Pomerleau hat in [Pomerleau 1989] gezeigt, daß eine gaußförmige Repräsentation der Ausgabe gegenüber anderen Darstellungsformen erhebliche Stärken aufweist. So kommt es bei einer kleinen Änderung der Eingabe-Aktivierung auch nur zu einer entsprechend kleinen Änderung der Ausgabe. Zudem haben neurophysiologische Studien ergeben, daß diese Art der Darstellung auch im Gehirn anzutreffen ist ([Andersen et al. 1985]). Für die Ausgabe-Aktivierung der Trainingsmuster ergibt sich somit:

$$a_i = e^{-\frac{|i-j|^2}{3}}, \tag{8.14}$$

wobei der Nenner des Exponenten eine statistisch ermittelte Größe darstellt (siehe [Pomerleau 1989]).

[16]Helle Neuronen besitzen gegenüber dunklen einen größeren Aktivierungsgrad.

8.9 Interpretation des Akkumulatorfeldes

Die Daten innerhalb des Akkumulatorfeldes können nicht nur Aufschluß über die aktuelle Position, sondern auch über die Umgebung geben.

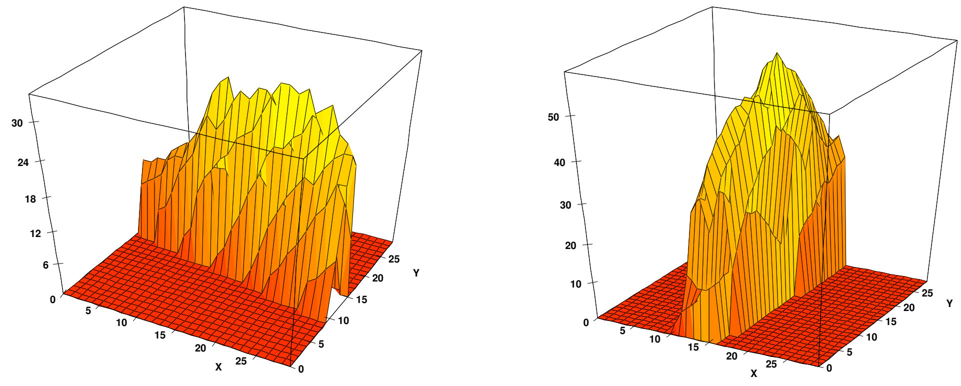

Abbildung 8.7: Horizontaler und vertikaler Gang

Abbildung 8.7 zeigt typische Gangsituationen. Während links die horizontalen Wände wesentlich näher zusammen sind als die vertikalen, ist rechts das Seitenverhältnis umgekehrt. Der Roboter befindet sich also in einem horizontalen bzw. vertikalem Korridor. Es ist deutlich zu erkennen, daß jeweils die zur Gangorientierung liegende Richtung nicht genau bestimmt werden kann, während die dazu orthogonal liegende relativ genau repräsentiert ist. Sind in dem Sensoreindruck nicht ausschließlich parallel zueinander liegende Wände, sondern auch weitere Merkmale verzeichnet, so kann sich der Roboter bezüglich der x- und y-Komponente lokalisieren. Abbildung 8.4 zeigt einen solchen Fall.

8.10 Experiment

Die erweiterte Hough-Transformation, die in diesem Kapitel besprochen wurde, ist in der Programmiersprache C++ implementiert, um sie ausführlichen Tests unterziehen zu können. Der betrachtete Suchraum beträgt dabei 3 Meter. Experimente, die eine größere Fensterumgebung beinhalten, zeigen grundsätzlich ein schlechteres Verhalten. So kommt es in Gangsituationen zu einem starken Oszillieren, bedingt durch parallel dazu verlaufende Räume, die in etwa die gleiche Breite haben. Dem Verfahren wird in zwanzig Ein-Grad-Schritten das zu suchende Segment zugeführt. Auch hier bringt eine Erweiterung des Suchraums[17]

[17] diesmal bzgl der Rotationsstufen

keine entscheidende Verbesserung. Durch den erhöhten Rechenaufwand ist das Driftkompensations-Modul erwartungsgemäß erheblich langsamer, was schließlich zu einer falschen Bestimmung des gesuchten Rotations- und Verschiebungsvektors führt. Längere Lokalisationszyklen geben dem System einen größeren Freiraum, in dem sich der Roboter bewegen kann. Es ist aber im allgemeinen nur für kurze Strecken sichergestellt, daß die Odometrie zuverlässig ist. Bei längeren Wegstrecken werden daher auch zunehmend die odometrischen Daten ungenauer. Überschreitet der zwischen zwei Lokalisations-Phasen zurückgelegte Weg eine bestimmte Länge, so wird anschließend durch die falsch vorgegebene Position der Suchraum außerhalb der realen Position des Roboters gesetzt. Der Roboter ist dann nicht mehr in der Lage, seine Position richtig zu bestimmen. Für die Cluster-Analyse des Akkumulatorfeldes wird die Max-Value-Strategie gewählt, und zuletzt bestimmt ein neuronales Netz die repräsentative Position.

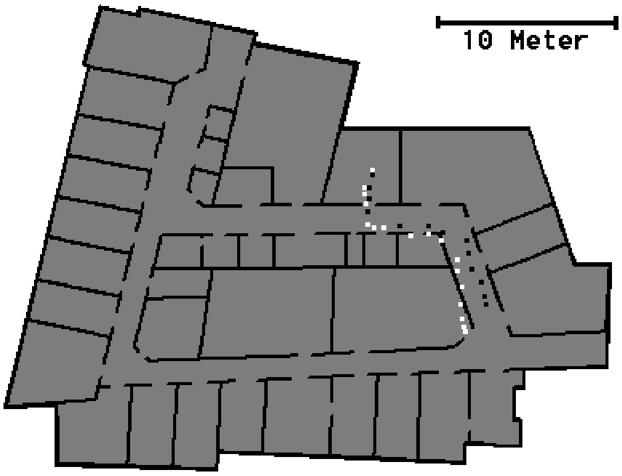

Abbildung 8.8: Testlauf des Hough-basierten Ansatzes

Abbildung 8.8 zeigt einen Versuch des Hough-basierten Ansatzes. Der Roboter fährt dabei, ausgehend vom oberen Raum, in die Mitte des Ganges und kommt dort schließlich an seinem Ziel rechts unten an. (Weiße Punkte stellen die odomotrische, schwarze hingegen die korrigierte Position dar.)

Die Analyse des Driftkompensations-Vektors (Abbildung 8.9) zeigt, daß dieser zu starken Sprüngen neigt. So sind anfangs innerhalb der Fensterumgebung nicht genügend Informationen für die Hough-Transformation vorhanden. Diese ist deshalb nicht in der Lage, die Position richtig zu bestimmen. Stattdessen kann es zu starken Sprüngen der Positions-Schätzung innerhalb des Suchraums kommen. Die werden allerdings durch die Verfahren in Kapitel 12 ausgeglichen bzw. nicht

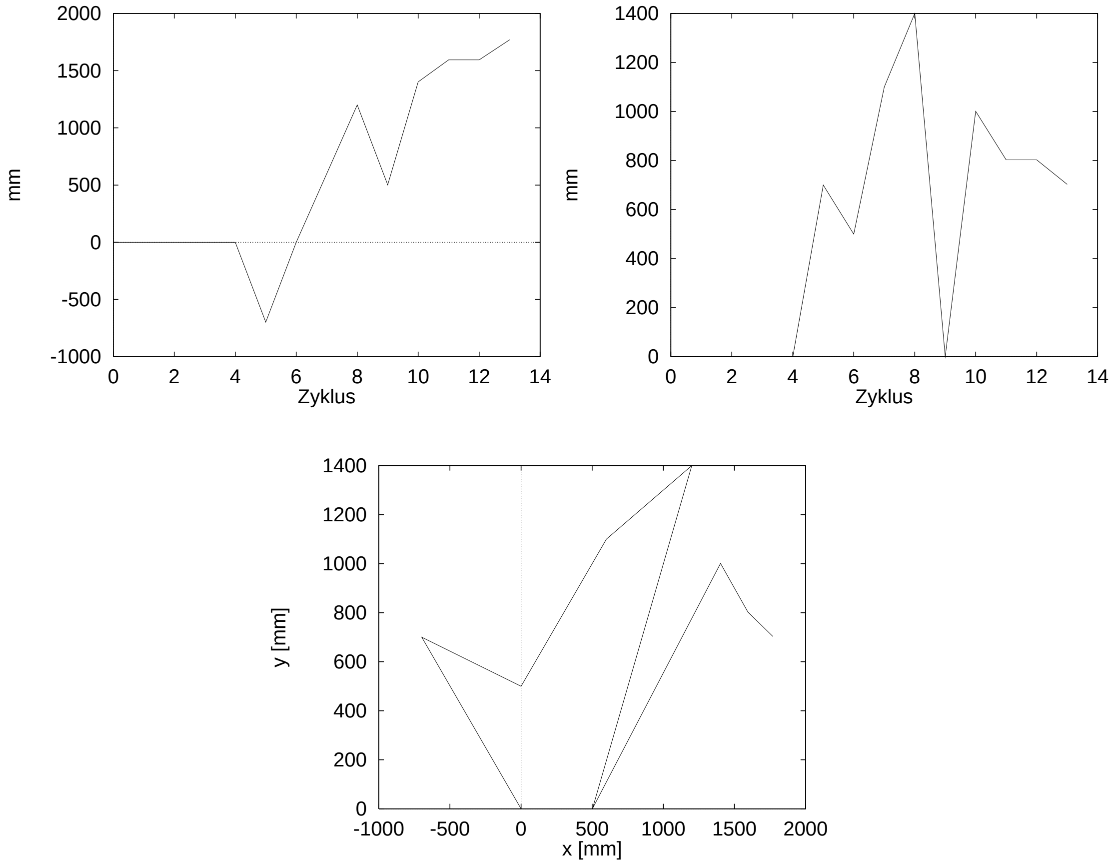

Abbildung 8.9: Verschiebungsgraphen des Experimentes

zugelassen. Die x- und die y-Komponente[18] des Vektors (Abbildung 8.9 oben) nehmen solange den Wert null an, bis der Roboter im Türbereich in eine ausreichend signifikante Umgebung gefahren ist. Befindet sich der Roboter nun im Gangbereich, d.h. dieser ist in der Suchumgebung enthalten, greift das Verfahren zur Driftkompensation.

An dieser Stelle ist es wichtig, daß die odometrische Position noch nicht zu weit von der realen Roboter-Position abweicht, da ansonsten keine korrekte Lokalisation des Roboters möglich ist. Aber auch bei einem richtig gesetzten Suchraum, sind noch Sprünge in der Positions-Bestimmung zu erkennen. Dies liegt zum einen an der relativ langen Zykluszeit der Hough-Transformation, die durch die zwanzig Rotationsstufen bedingt ist, und zum anderen an Fehlmessungen bedingt durch die uncharakteristische Umgebung. Der Rotations-Anteil des Drift-Vektors ist in der Abbildung nicht berücksichtigt, da dieser nur leicht um den Wert null oszilliert. Während in der Abbildung 8.9 oben der Driftkompensations-

[18]hier in Abhängigkeit des Lokalisations-Zyklus dargestellt

Vektor komponentenweise in Abhängigkeit des Lokalisations-Zyklus dargestellt ist, fließen in den unteren Graphen beide Komponenten ein. Dieser verdeutlicht die Tendenz des Vektors. Bei einer genauen Odometrie sollte diese nur geringfügig vom Ursprung abweichen.

Ein Problem, welches immer wieder in Gangsituationen auftritt, ist die richtige Bestimmung der in Korridor-Orientierung liegenden Komponente des Driftkompensations-Vektors. Solange nur die beiden gegenüberliegenden Wände und keine weiteren Hinweise innerhalb des Suchraums enthalten sind, kann der gesuchte Anteil des Vektors nur schwer bzw. überhaupt nicht korrekt bestimmt werden. Existieren andererseits markante Ecken oder andere natürliche Landmarken, wie in diesem Experiment, so ist das Verfahren durchaus in der Lage, die Position zu ermitteln.

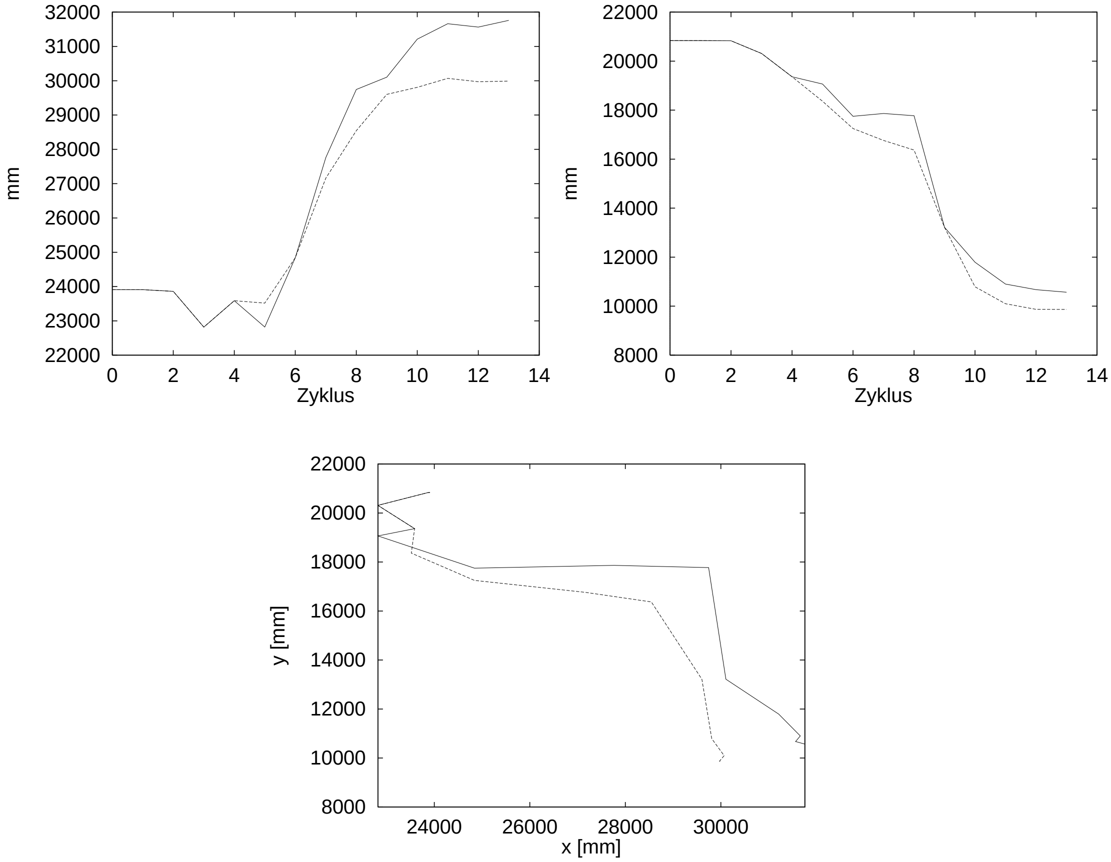

Abbildung 8.10: Positionsgraphen des Experimentes

Abbildung 8.10 (oben) zeigt jeweils für die x- und die y-Komponente die odometrische bzw. die korrigierte Position des Roboters[19]. Es läßt sich deutlich

[19] gestrichelte Linie ≡ Odometrie, durchgezogene Linie ≡ korrigierte Position.

die gegenseitige Verschiebung beider Positionen zueinander erkennen. Der untere Graph stellt jeweils den zurückgelegten Weg dar, wobei die korrigierte Position in etwa die reale Roboter-Position wiederspiegelt.

Kapitel 9

Geradenbasierte Driftkompensation

Bei der geradenbasierten Driftkompensation wird den einzelnen Bildpunkten der lokalen Karte[1] die jeweils nächstliegende Gerade der globalen Karte zugeordnet ([MacKenzie, Dudek 1994]). Anschließend kann, wie in [Cox, Kruskal 1988] beschrieben, der Korrekturvektor, bestehend aus der Rotations- und den Translations-Komponenten, bestimmt werden. Für diese Bestimmung ergibt sich ein lineares Gleichungssystem, welches mit den üblichen numerischen Lösungsverfahren gelöst werden kann. Der Vorteil dieses Ansatzes gegenüber der Hough-basierten Driftkorrektur liegt in der Einbeziehung der Rotations-Komponente, d.h. die lokale Karte muß dem Verfahren nicht in diskreten Rotationsstufen zugeführt werden. Außerdem hat sich in den Experimenten zu der Hough-Transformation gezeigt, daß diese nur in charakteristischen Umgebungen zuverlässig die Position des Roboters bestimmen kann. In Korridoren kommt es unter Umständen zu fehlerhaften Driftkorrekturen. Hier wendet sich der Vorteil der Hough-Transformation, auch noch gestörte Segmente zuordnen zu können, in einen Nachteil. Türen werden lediglich als Störungen interpretiert. Der geradenbasierte Ansatz hingegen liefert selbst in Fluren noch die richtige Position, da enthaltene Informationen, wie das Vorhandensein einer Tür, nicht mehr verloren gehen.

9.1 Datenfluß

Das Datenfluß-Diagramm aus Abbildung 9.1 verdeutlicht den Vorteil der geraden- gegenüber der Hough-basierten Driftkompensation. Während bei letzterem Ansatz wesentlich mehr Schleifendurchläufe erforderlich sind[2], um den gewünschten Korrekturvektor und die damit verbundene korrigierte Position zu ermitteln, wird dies bei der geradenbasierten Positionsvektor-Korrektur in nur wenigen Schritten

[1]Die lokale Karte wird aus den über die Zeit integrierten Sensoreindrücken gewonnen.
[2]Es müssen in jedem Fall alle Rotationsstufen durchlaufen werden.

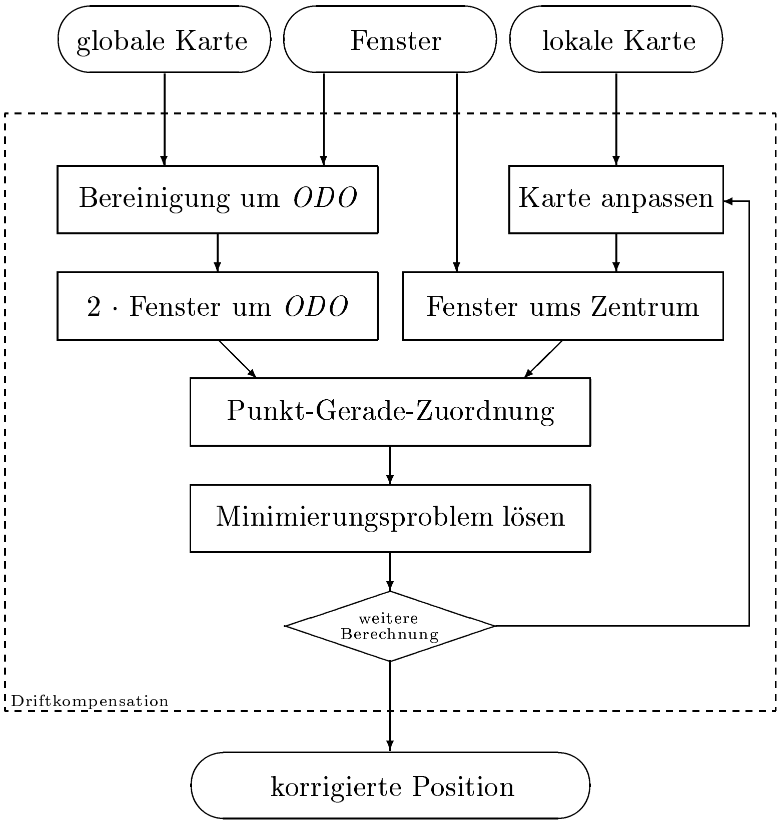

Abbildung 9.1: Datenfluß-Diagramm der geradenbasierten Lokalisation

erreicht. Dazu trägt im wesentlichen die zusätzliche Einbeziehung der Rotations-Komponente bei. Es werden nun folgende Daten von diesem Ansatz benötigt:

1. die globale Karte (bzw. die Liste der Eckpunkte),
2. die lokale Karte,
3. ein Fensterausschnitt.

Mit Hilfe des Fensterausschnitts läßt sich der Suchraum innerhalb der globalen und der lokalen Karte einschränken. Es wird ein Fenster in der globalen Karte um die mit der lokalen Karte gelieferte odometrische Position gebildet. Bei der lokalen Karte wird nur ein Fensterausschnitt um das Zentrum gewählt, sofern dieser die Kartengröße nicht übersteigt[3]. Für den Kartenausschnitt der globalen Karte

[3] Es ist durchaus möglich, den zu untersuchenden Bereich größer als die lokale Karte zu wählen. In solchen Fällen wird die vollständige lokale Karte verwendet.

ist zuvor eine Transformation von Welt-Koordinaten in Karten-Koordinaten notwendig. Diese wird zur Bereinigung der odometrischen Daten herangezogen. Die weitere Verarbeitung findet nur noch in Karten-Koordinaten statt, wobei globale und lokale Karte zwar dieselbe Skalierung, aber dennoch ihr eigenes Koordinatensystem besitzen. D.h. die beiden Koordinatensysteme lassen sich durch Translation und Rotation ineinander überführen. Das Zentrum der lokalen Karte stellt die odometrische Position, um die in der globalen Karte der Ausschnitt gewählt wird, dar. Beide Punkte werden nun für die Punkt-Gerade-Zuordnung in Übereinstimmung gebracht[4]. Nach Abschluß der Zuordnung wird jedem Punkt der lokalen Karte[5] eine Gerade der globalen Karte[6] zugeordnet. Diese wird zur Lösung des linearen Gleichungssystems, welches aus dem Abstand-Minimierungsproblems resultiert, herangezogen. Ist die Differenz zum vorhergehenden Lösungsvektor kleiner einer vorbestimmten Schwelle, so bricht das Verfahren ab. Als Ergebnis wird schließlich die korrigierte Position ausgegeben.

9.2 Eckpunktextraktion

Um die geradenbasierte Driftkompensation anwenden zu können, ist es notwendig, die globale Karte in ein Geradenbild zu transformieren. D.h. für jedes Liniensegment in der globalen Karte muß die zugehörige Gerade bestimmt werden. Es ist sinnvoll, dies auf Grund einer vorangegangenen Eckpunktextraktion vorzunehmen. Anfangs- und Endpunkte einzelner Liniensegmente sind ausreichend, die zugehörige Gerade zu beschreiben. Das hier vorgestellte Verfahren zur Extraktion von Eckpunkten ist eine Erweiterung des von Haye Hähne in [Hähne 1995] vorgestellten Operators *vcd*. Ausgangspunkt für die Eckenerkennung ist die binarisierte globale Karte, wie sie in Abbildung 9.2 zu sehen ist. Dabei stellen Bildpunkte mit dem Wert eins Konturpunkte und Bildpunkte mit dem Wert null Hintergrundpunkte dar. Mit Hilfe eines Konturfolge-Algorithmus, der auf den Konturpunkten entlang läuft, wird die globale Karte in mehrere Teilkonturen zerlegt. Diese Zerlegung findet immer genau dann statt, wenn das Konturfolge-Verfahren wegen einer zu starken Krümmung abbricht. Für die Konturfolge sind die zwei Funktionen $V(P,d)$ und $R(P,d)$ vorgegeben, die d Pixel entfernte Punkte auf der Kontur in Vorwärts- bzw. Rückwärts-Richtung ausgeben. Für jeden Konturpunkt P werden nun folgende Schritte ausgeführt:

1. Der Parameter d wird solange erhöht, bis der Krümmungswinkel zwischen $(P, R(P,d))$ und $(P, V(P,d))$ nicht weiter steigt und ein lokales Maximum vorliegt. Der so ermittelte Winkel sei P zugeordnet.

[4]Dies entspricht einer Translation eines der beiden Koordinatensystemen.

[5]Punkte, die sich innerhalb des Fensterauschnittes befinden.

[6]Die Geraden müssen lediglich durch den Fensterauschnitt gehen, d.h. Anfangs- bzw. Endpunkt ist innerhalb des Fensters oder das Fenster wird durch die Gerade geschnitten.

2. P ist ein Eckpunkt, falls ein lokales Maximum für die den Konturpunkten zugeordneten Winkel vorliegt oder das Ende der Teilkontur erreicht ist.

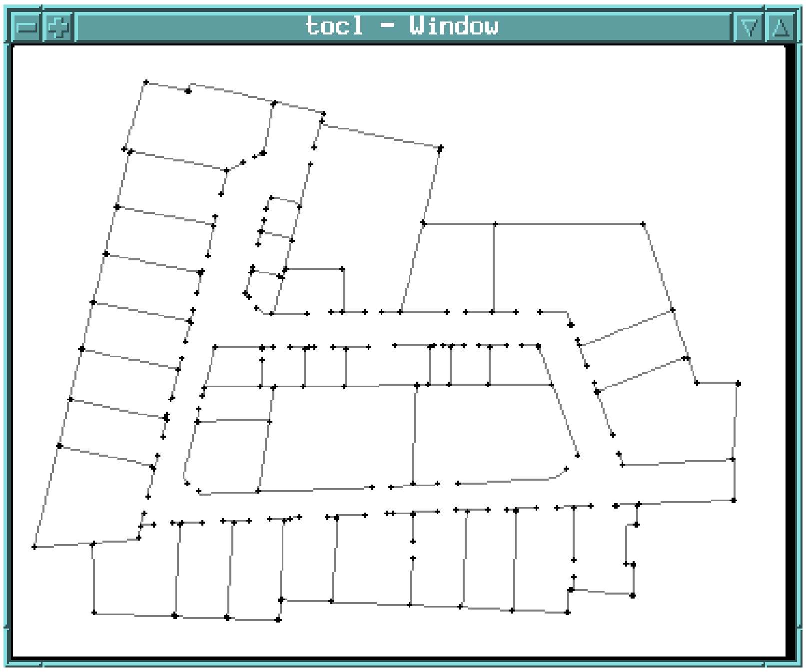

Abbildung 9.2: Quantisierte globale Karte

Die Verwendung eines Skalenraumes[7], wie im ersten Schritt vorgenommen oder auch in [Rattarangsi, Chin 1990] beschrieben, löst das Problem der richtigen Skalierung des Einzugsbereichs. Ist dieser zu groß gewählt, kann es zu einer Unterabtastung von Ecken kommen. In diesem Fall befinden sich mehrere Ecken im Einzugsbereich, so daß die Krümmungsmessung an Hand eines überlagerten Krümmungswinkels vorgenommen wird. Ist d zu klein gewählt, so ist der gemessene Winkel wegen Rauschens nicht aussagekräftig. Es wird daher im Schritt eins für d ein d_{min} vorgegeben, welches oberhalb eines eventuell vorhandenen Rauschens liegt. Abbildung 9.2 zeigt die globale Karte nach Abschluß der Eckendetektion. Die erkannten Eckpunkte sind in Form von Kreuzen markiert. Zum Vergleich zeigt Abbildung 9.3 die aus den Eckpunkten rekonstruierte globale Karte.

9.3 Rekonstruktion als Vektorbild

Die im vorangegangenen Abschnitt gefundenen Eckpunkte können nun zur Rekonstruktion der geradenbasierten globalen Karte herangezogen werden. Hierfür

[7]hier: Krümmungswinkel eines Konturpunktes im Bezug auf den Einzugsbereich d.

ist es notwendig, daß die Eckpunkte in einer geordneten Liste der Form

$$[x_{1_1}, x_{1_2}, \ldots, x_{1_n}, -1, x_{2_1}, x_{2_2}, \ldots, x_{2_m}, -1, \ldots, -1, x_{z_1}, x_{z_2}, \ldots, x_{z_l}] \tag{9.1}$$

vorliegen. Jedes x_{i_j} entspricht einem Eckpunkt und läßt sich, da die Bildbreite bekannt ist, wieder in kartesische Koordinaten zurücktransformieren.

$$\begin{aligned} x &= x_{i_j} \quad \text{Modulo} \quad \mathit{Bildbreite} \\ y &= x_{i_j} \quad \text{Div} \quad \mathit{Bildbreite} \end{aligned}$$

Die Rekonstruktion basiert auf der besonderen Struktur dieser Liste (Ausdruck (9.1)). Jeder Punkt x_{i_j} wird mit dem Punkt $x_{i_{j-1}}$ verbunden, sofern $j > 1$ gilt. Tritt als Vorgängerwert eines Punktes eine -1 auf, so existiert keine Verbindung, d.h. dieser Punkt ist ein neuer Startpunkt. Abbildung 9.3 zeigt die geradenbasierte Karte nach erfolgter Rekonstruktion.

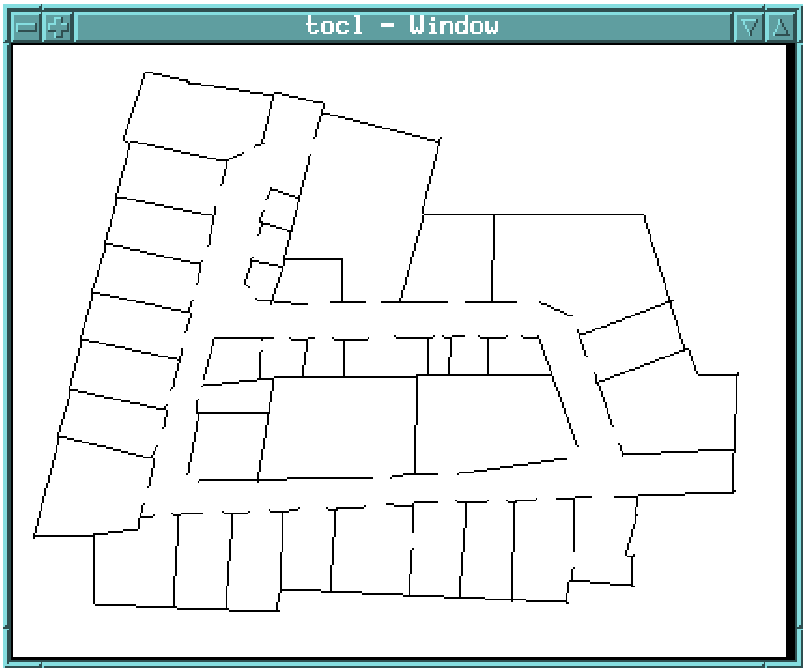

Abbildung 9.3: Geradenbasierte globale Karte

9.4 Punkt-Gerade-Zuordnung

Die Rekonstruktion in Abschnitt 9.3 stellt für jede Linie in der globalen Karte Anfangs- und Endpunkt zur Verfügung. Diese können zur Beschreibung der zugehörigen Geraden herangezogen werden. Seien nun $u, v \in I\!R^2$ mit $u = (x_1, y_1)$

und $v = (x_2, y_2)$ Anfangs- bzw. Endpunkt einer solchen Linie. Die Differenz $w = v - u$ der beiden Linienpunkte entspricht dann einem Richtungsvektor, der parallel zu diesem Liniensegment liegt. Die zugehörige Gerade A läßt sich mittels folgender Parameterdarstellung beschreiben:

$$A = v + \mathbb{R}w := \{p \in \mathbb{R}^2 \mid \exists \lambda \in \mathbb{R} \;:\; p = v + \lambda w\}. \tag{9.2}$$

Es wird nun ein Vektor gesucht, der orthogonal zu dieser Geraden steht. Ein möglicher Kandidat ist der um 90 Grad gedrehte Vektor w. Sei also der Vektor $s = (y_2 - y_1, x_1 - x_2)$ ein zur Geraden A orthogonaler Vektor. Der zugehörige Normalenvektor $s' = (a, b)$ läßt sich durch die Normierung

$$s' = \frac{1}{\| s \|} \cdot s \tag{9.3}$$

ermitteln. Mit $c = a \cdot x_1 + b \cdot y_1$ ergibt sich dann mit Hilfe der Hesseschen Normalform für (9.2) folgende Geradengleichung:

$$A = \{(x, y) \in \mathbb{R}^2 \mid (x, y) \cdot \begin{pmatrix} a \\ b \end{pmatrix} - c = 0\}. \tag{9.4}$$

Sei nun $\mathcal{A}$ die Menge der Geraden, die genau einem Liniesegment in der globalen Karte entsprechen, und $\mathcal{V}$ die Menge der Bildpunkte der lokalen Karte. Sind $u = (x_1, y_1)$ und $v = (x_2, y_2)$ Anfangs- bzw. Endpunkt des zur Geraden A zugehörigen Liniensegmentes, so ist m als der Mittelpunkt dieser Linie definiert:

$$m = \left(x_1 + \frac{x_2 - x_1}{2}, y_1 + \frac{y_2 - y_1}{2} \right). \tag{9.5}$$

Der Abstand eines Punktes $v \in \mathcal{V}$ zu einer Geraden $A \in \mathcal{A}$ ist

$$d_G(v, A) = (v \cdot \begin{pmatrix} a \\ b \end{pmatrix} - c)^2. \tag{9.6}$$

Der Abstand zweier Bildpunkte ist gegeben durch den euklidischen Abstand

$$d_P(v, m) = \| v - m \|_2 . \tag{9.7}$$

Jeder Bildpunkt $v_i \in \mathcal{V}$ wird nun genau einer Geraden $A_j \in \mathcal{A}$ in Form der Menge

$$\mathcal{Z} = \{(v_i, A_j) \in \mathcal{V} \times \mathcal{A} \mid \forall v_k \in \mathcal{V} : v_k \neq v_i \Rightarrow d_L(v_i, A_j) < d_L(v_k, A_j)\} \tag{9.8}$$

zugeordnet, wobei $d_L(v, A)$ der Abstand zwischen einem Bildpunkt $v \in \mathcal{V}$ und einer, dem Liniensegment zugehörigen, Geraden $A \in \mathcal{A}$ ist[8]:

$$d_L(v, A) = \alpha \cdot d_G(v, A) + \beta \cdot d_P(v, m). \tag{9.9}$$

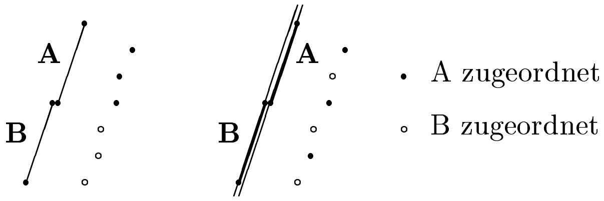

Abbildung 9.4: Problem der fehlerhaften Geradenzuordnung

Die Hinzunahme des Abstandes zwischen Bildpunkt und Mittelpunkt des zugehörigen Liniensegments in (9.8) ist erforderlich, um Zuordnungsfehler zu vermeiden. Abbildung 9.4 verdeutlicht diesen Zusammenhang. Es sind zwei Liniensegmente A und B vorgegeben. Findet eine Zuordnung ausschließlich auf der Basis des Abstandes zwischen Bildpunkt und nächstliegender Geraden statt, kommt es zu einem Zuordnungsfehler (Abbildung 9.4 rechts). Wird nun zusätzlich der Abstand zum Linienmittelpunkt berücksichtigt, werden diese Zuordnungsfehler vermieden (Abbildung 9.4 links). Außerdem wird die richtige Bestimmung des Driftkompensations-Vektors in Gangsituationen unterstützt. Befinden sich die Türöffnungen in dem zu lokalisierenden Bereich, wird durch die Hinzunahme des Linienmittelpunktes eine beliebige Verschiebung in Gangrichtung vermieden, da die Liniensegmente vor und nach der Türöffnung verschiedene Mittelpunkte besitzen. Die Punkte der lokalen Karte werden dann entsprechend der einen bzw. der anderen Seite zugeordnet. Die Berücksichtigung des Abstandes zum Linienmittelpunkt fließt lediglich in die Punkt-Gerade-Zuordnung ein. Wurde jedem Bildpunkt im Suchraum eine Gerade zugeordnet, kann eine Korrektur der Position entweder nur bezüglich der Translation oder aber bezüglich der Translation und Rotation vorgenommen werden.

9.5 Translationskorrektur

In diesem Abschnitt wird die Translationskorrektur besprochen. Kann die Orientierung mittels eines Kompasses oder anderen Meßgerätes genau bestimmt werden, so reicht es aus, die Verschiebung zur tatsächlichen Position zu berechnen. Der Vorteil liegt, durch die Reduzierung der Rechenoperationen bedingt, in einer wesentlich kürzeren Zykluszeit.

In Abbildung 9.5 ist zu erkennen, daß lediglich ein Translationsvektor notwendig ist, um die größte Übereinstimmung der Bildpunkte mit dem Linienmodell

[8] Die Faktoren α und β dienen der geeigneten Skalierung.

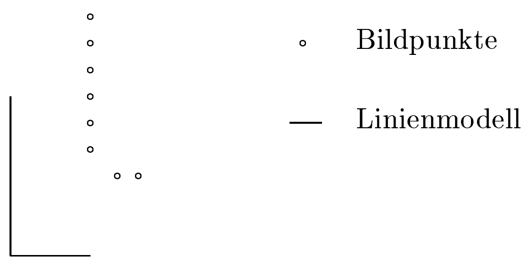

Abbildung 9.5: Translationskorrektur von Bildpunkten und dem Linienmodell

herzustellen. Jeder Bildpunkt kann dann mit Hilfe des Vektors t wie folgt transformiert werden:

$$v \to v + t. \tag{9.10}$$

Hieraus ergibt sich für alle in Abschnitt 9.4 getroffenen Zuordnungen $(v, A) \in \mathcal{Z}$ folgendes Minimierungsproblem:

$$MIN \;\; \mathcal{S} = \sum_{(v,A)\in\mathcal{Z}} d(v + t, A). \tag{9.11}$$

Wird für die Abstandsmessung in (9.11) der Abstand zwischen einem Punkt und einer Gerade (Ausdruck (9.6)) herangezogen, so ergibt sich:

$$\begin{aligned} \sum_{(v,A)\in\mathcal{Z}} d(v+t,A) &= \sum_{(v,A)\in\mathcal{Z}} (a^2(v_1+t_1)^2 + 2ab(v_1+t_1)(v_2+t_2) \\ &\quad -2ac(v_1+t_1) - 2bc(v_2+t_2) + b^2(v_2+t_2)^2 + c^2). \end{aligned} \tag{9.12}$$

Falls die Ableitungen von $\sum_{(v,A)\in\mathcal{Z}} d(v + t, A)$ nach t_1 bzw. t_2 in einem Punkt den Wert null annehmen, liegt die notwendige Bedingung für die Existenz eines relativen Minimums vor. Die Ableitung nach t_1 stellt sich wie folgt dar:

$$\frac{\partial \sum_{(v,A)\in\mathcal{Z}} d(v+t,A)}{\partial t_1} = \sum_{(v,A)\in\mathcal{Z}} (2a^2v_1 + 2a^2t_1 + 2abv_2 + 2abt_2 - 2ac). \tag{9.13}$$

Die Ableitung nach t_2 läßt sich entsprechend berechnen:

$$\frac{\partial \sum_{(v,A)\in\mathcal{Z}} d(v+t,A)}{\partial t_2} = \sum_{(v,A)\in\mathcal{Z}} (2b^2v_2 + 2b^2t_2 + 2abv_1 + 2abt_1 - 2bc). \tag{9.14}$$

Um nun die hinreichende Bedingung für die Existenz eines relativen Minimums zu erfüllen, muß die dazugehörige Jacobimatrix[9]

$$\mathcal{J} = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix} = \begin{pmatrix} \frac{\partial\left(\frac{\partial \sum_{(v,A)\in\mathcal{Z}} d(v+t,A)}{\partial t_1}\right)}{\partial t_1} & \frac{\partial\left(\frac{\partial \sum_{(v,A)\in\mathcal{Z}} d(v+t,A)}{\partial t_1}\right)}{\partial t_2} \\ \frac{\partial\left(\frac{\partial \sum_{(v,A)\in\mathcal{Z}} d(v+t,A)}{\partial t_2}\right)}{\partial t_1} & \frac{\partial\left(\frac{\partial \sum_{(v,A)\in\mathcal{Z}} d(v+t,A)}{\partial t_2}\right)}{\partial t_2} \end{pmatrix} \tag{9.15}$$

positiv definit sein. Die Matrix in (9.15) ist genau dann in einem Punkt positiv definit, wenn die Jacobische Determinante[10]

$$det(\mathcal{J}) = \begin{vmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{vmatrix} = J_{11} \cdot J_{22} - J_{12} \cdot J_{21} \tag{9.16}$$

in diesem Punkt positiv ist. Mit (9.13) und (9.14) läßt sich die Jacobimatrix konkret angeben:

$$\mathcal{J} = \begin{pmatrix} \sum_{(v,A)\in\mathcal{Z}} 2a^2 & \sum_{(v,A)\in\mathcal{Z}} 2ab \\ \sum_{(v,A)\in\mathcal{Z}} 2ab & \sum_{(v,A)\in\mathcal{Z}} 2b^2 \end{pmatrix}. \tag{9.17}$$

Die Determinante aus Ausdruck (9.16) berechnet sich dann wie folgt:

$$det(\mathcal{J}) = 4 \cdot \left(\sum_{(v,A)\in\mathcal{Z}} a^2 \sum_{(v,A)\in\mathcal{Z}} b^2 - \left(\sum_{(v,A)\in\mathcal{Z}} ab \right)^2 \right). \tag{9.18}$$

Die hinreichende Bedingung für das Vorkommen eines Minimums dient nur zur Überprüfung des ermittelten Ergebnisses. Sie fließt nicht unmittelbar in das Gleichungssystem[11] $(a_{ij})(t_i) = (b_i)$ zur Lösung des Minimierungsproblems ein. Dieses wird aus den Ableitungen (9.13) und (9.14) gebildet, die zur Bestimmung des Lösungsvektors t gleich null gesetzt werden:

$$\left(\sum_{(v,A)\in\mathcal{Z}} \begin{pmatrix} a^2 & ab \\ ab & b^2 \end{pmatrix} \right) \cdot \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \sum_{(v,A)\in\mathcal{Z}} \begin{pmatrix} -a^2 v_1 - abv_2 + ac \\ -b^2 v_2 - abv_1 + bc \end{pmatrix}. \tag{9.19}$$

Es ergibt sich schließlich das folgende Gleichungssystem:

$$\left(\sum_{(v,A)\in\mathcal{Z}} \left(\begin{pmatrix} a \\ b \end{pmatrix} \cdot (a,b) \right) \right) \cdot \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} = \sum_{(v,A)\in\mathcal{Z}} \begin{pmatrix} a \\ b \end{pmatrix} \cdot \left(c - (a,b) \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \right). \tag{9.20}$$

[9] Auch unter dem Begriff *Funktionalmatrix* bekannt.

[10] In diesem Zusammenhang wird oft von der *Funktionaldeterminate* gesprochen.

[11] Es gilt: $(a_{ij})(t_i) = (\sum_{(v,A)\in\mathcal{Z}}(a'_{ij}))(t_i) = \sum_{(v,A)\in\mathcal{Z}}(b'_i) = (b_i)$.

Somit lassen sich für t_1 und t_2 bei gegebenem Gleichungssystem $(a_{ij})(t_i) = (b_i)$ die folgenden zwei Ausdrücke angeben:

$$t_1 = \frac{b_1 a_{22} - b_2 a_{12}}{a_{11} a_{22} - a_{12} a_{21}} \quad \text{und} \tag{9.21}$$

$$t_2 = \frac{b_2 a_{11} - b_1 a_{21}}{a_{11} a_{22} - a_{12} a_{21}}. \tag{9.22}$$

Nimmt die Determinate den Wert null an, so besitzt obige Matrix nicht den vollen Rang und das Gleichungssystem müßte mittels einer Singulärwert-Zerlegung gelöst werden. Dieser Weg wird, wegen des unverhältnismäßg hohen Rechenaufwands, nicht gegangen. Vielmehr wird das Ergebnis einfach verworfen.

Bemerkung 9.1 *Singuläre Matrizen liegen vor, wenn ein großer Teil an parallel verlaufenden Geraden in das Gleichungssystem eingegangen ist.*

9.6 Translations- und Rotationskorrektur

Bei der Translations- und Rotationskorrektur wird zusätzlich, wie der Name schon vermuten läßt, eine Rotation des Roboters berücksichtigt. Die Einbeziehung der Drehung des Roboters hat einen erhöhten Rechenaufwand und damit eine längere Zykluszeit zur Folge. Diese liegt, da die Zykluszeit ca. eine Sekunde beträgt[12], in einem Bereich, der für die Lokalisation unkritisch ist.

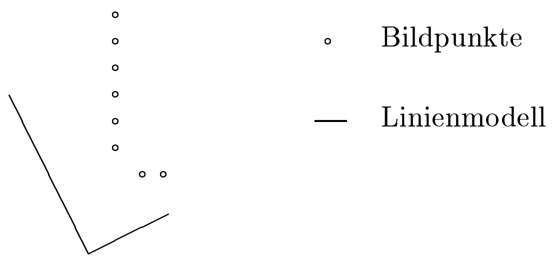

Abbildung 9.6: Translations- und Rotationskorrektur

Abbildung 9.6 veranschaulicht die Notwendigkeit einer zusätzlichen Rotationskorrektur. Um eine gute Übereinstimmung der Bildpunkte mit dem Linienmodell zu erhalten, ist eine Verschiebung und eine zusätzliche Rotation notwendig. Bei der Rotation muß darauf geachtet werden, daß das Rotationszentrum m dem Bildzentrum entspricht. Dies hat den Vorteil, daß die Hintereinanderausführung von (t_1, θ_1) und (t_2, θ_2) äquivalent zu $(t_1 + t_2, \theta_1 + \theta_2)$ ist. Jeder Bildpunkt v wird dann mit Hilfe des Tupels (t, θ) in der folgenden Form transformiert:

$$v \rightarrow R(\theta) \cdot (v - m) + (m + t), \tag{9.23}$$

[12] auf einer Sun Sparc Station 10

wobei

$$R(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \tag{9.24}$$

die übliche Rotationsmatrix darstellt. Gesucht wird nun das Tupel (t, θ), welches für alle Bildpunkte der lokalen Karte den folgenden Ausdruck minimiert:

$$MIN \ \ \mathcal{S} = \sum_{(v,A)\in\mathcal{Z}} d(R(\theta)\cdot(v-m)+(m+t), A). \tag{9.25}$$

Ausdruck (9.25) stellt allerdings, wegen der Rotationsmatrix (9.24), kein lineares Minimierungsproblem dar. Um dieses nun zu linearisieren, bietet sich die Verwendung einer sogenannten Pseudo-Rotation an, die durch das Heranziehen des linearen Anteils der entsprechenden trigonometrischen Funktionen (Ausdruck (9.26) und (9.27)) hervorgeht.

$$sin(\theta) = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \cdots \quad \text{und} \tag{9.26}$$

$$cos(\theta) = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \cdots . \tag{9.27}$$

Während (9.24) jeden Punkt auf einem Kreisbogen um das Zentrum rotiert, werden bei (9.28) die Punkte auf einer Tangente des Kreisbogens bewegt. Als Rotationsmatrix ergibt sich somit:

$$R'(\theta) = \begin{pmatrix} 1 & -\theta \\ \theta & 1 \end{pmatrix}. \tag{9.28}$$

Sei nun $f(t1, t2, \theta) = \sum_{(v,A)\in\mathcal{Z}} d(R'(\theta)\cdot(v-m)+(m+t), A)$. Dann ergibt sich mit Hilfe von (9.6) folgende Gleichung:

$$f(t1, t2, \theta) = \sum_{(v,A)\in\mathcal{Z}} (a(v_1 - \theta v_2 + \theta m_2 + t_1) + b(\theta v_1 - \theta m_1 + v_2 + t_2) - c)^2. \tag{9.29}$$

Um, wie schon im Abschnitt 9.5 erwähnt, das notwendige Kriterium für das Vorliegen eines Minimums an einem bestimmten Punkt überprüfen zu können, müssen die entsprechenden Ableitungen nach t_1, t_2 und θ gebildet werden. Somit ergibt sich für die Ableitung nach t_1:

$$\begin{aligned} \frac{\partial f(t1, t2, \theta)}{\partial t_1} = \textstyle\sum_{(v,A)\in\mathcal{Z}} \quad & 2\cdot(-ac + ab(t_2 + v_2 - \theta m_1 + \theta v_1) \\ & + a^2(t_1 + v_1 + \theta m_2 - \theta v_2)). \end{aligned} \tag{9.30}$$

Die Ableitung nach t_2 läßt sich entsprechend angeben:

$$\begin{aligned}\frac{\partial f(t1,t2,\theta)}{\partial t_2} = \textstyle\sum_{(v,A)\in\mathcal{Z}} \quad & 2\cdot(-bc+ab(t_1+v_1+\theta m_2-\theta v_2) \\ & +b^2(t_2+v_2-2\theta m_1+\theta v_1)).\end{aligned} \tag{9.31}$$

Und schließlich gibt Ausdruck (9.32) die Ableitung nach θ an:

$$\begin{aligned}\frac{\partial f(t1,t2,\theta)}{\partial \theta} = \textstyle\sum_{(v,A)\in\mathcal{Z}} \quad & 2\cdot(b(-m_1+v_1)+a(m_2-v_2)) \\ & \cdot(-c+a(t_1+v_1+\theta m_2-\theta v_2) \\ & +b(t_2+v_2-\theta m_1+\theta v_1)).\end{aligned} \tag{9.32}$$

Für die hinreichende Bedingung muß die dazugehörige Jacobimatrix

$$\mathcal{J} = \begin{pmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{pmatrix} = \begin{pmatrix} \frac{\partial\left(\frac{\partial f(t1,t2,\theta)}{\partial t_1}\right)}{\partial t_1} & \frac{\partial\left(\frac{\partial f(t1,t2,\theta)}{\partial t_1}\right)}{\partial t_2} & \frac{\partial\left(\frac{\partial f(t1,t2,\theta)}{\partial t_1}\right)}{\partial \theta} \\ \frac{\partial\left(\frac{\partial f(t1,t2,\theta)}{\partial t_2}\right)}{\partial t_1} & \frac{\partial\left(\frac{\partial f(t1,t2,\theta)}{\partial t_2}\right)}{\partial t_2} & \frac{\partial\left(\frac{\partial f(t1,t2,\theta)}{\partial t_2}\right)}{\partial \theta} \\ \frac{\partial\left(\frac{\partial f(t1,t2,\theta)}{\partial \theta}\right)}{\partial t_1} & \frac{\partial\left(\frac{\partial f(t1,t2,\theta)}{\partial \theta}\right)}{\partial t_2} & \frac{\partial\left(\frac{\partial f(t1,t2,\theta)}{\partial \theta}\right)}{\partial \theta} \end{pmatrix} \tag{9.33}$$

positiv definit sein. Diese ist positiv definit, wenn ihre Determinante größer null ist. Die Determinante läßt sich mittels der Regel von Sarrus direkt aus der Jacobimatrix berechnen:

$$\begin{aligned} det(\mathcal{J}) &= +J_{11}\cdot J_{22}\cdot J_{33}+J_{12}\cdot J_{23}\cdot J_{31}+J_{13}\cdot J_{21}\cdot J_{32} \\ & \quad -J_{11}\cdot J_{23}\cdot J_{32}-J_{12}\cdot J_{21}\cdot J_{33}-J_{13}\cdot J_{22}\cdot J_{31}.\end{aligned} \tag{9.34}$$

Mit Hilfe von (9.30) bis (9.32) kann die Jacobimatrix angegeben werden:

$$
\begin{aligned}
J_{11} &= \sum_{(v,A)\in\mathcal{Z}} 2a^2, && (9.35)\\
J_{12} &= \sum_{(v,A)\in\mathcal{Z}} 2ab, && (9.36)\\
J_{13} &= \sum_{(v,A)\in\mathcal{Z}} (2ab(-m_1+v_1)+2a^2(m_2-v_2)), && (9.37)\\
J_{21} &= \sum_{(v,A)\in\mathcal{Z}} 2ab, && (9.38)\\
J_{22} &= \sum_{(v,A)\in\mathcal{Z}} 2b^2, && (9.39)\\
J_{23} &= \sum_{(v,A)\in\mathcal{Z}} (2ab(m_2-v_2)+2b^2(-m_1+v_1)), && (9.40)\\
J_{31} &= \sum_{(v,A)\in\mathcal{Z}} 2a(b(-m_1+v_1)+a(m_2-v_2)), && (9.41)\\
J_{32} &= \sum_{(v,A)\in\mathcal{Z}} 2b(b(-m_1+v_1)+a(m_2-v_2)) \quad \textit{und} && (9.42)\\
J_{33} &= \sum_{(v,A)\in\mathcal{Z}} 2(b(-m_1+v_1)+a(m_2-v_2))^2. && (9.43)
\end{aligned}
$$

Bemerkung 9.2 *Im Rotations- und Translationsmodell werden die Ergebnisse, ähnlich dem einfacheren Translationsmodell, im Falle einer singulären Matrix einfach verworfen.*

Werden (9.30), (9.31) und (9.32) gleich null gesetzt, kann das Gleichungssystem[13] $(a_{ij})(t_i) = (b_i)$ konkret angegeben werden:

$$
\left(\sum_{(v,A)\in\mathcal{Z}} \begin{pmatrix} a^2 & ab & A \\ ab & b^2 & B \\ A & B & C \end{pmatrix}\right) \cdot \begin{pmatrix} t_1 \\ t_2 \\ \theta \end{pmatrix} = \sum_{(v,A)\in\mathcal{Z}} \begin{pmatrix} -a^2v_1 - abv_2 + ac \\ -b^2v_2 - abv_1 + bc \\ D \end{pmatrix}.
$$

A, B, C und D nehmen dabei folgende Gestalt an:

$$
\begin{aligned}
A &= a^2(m_2-v_2)+ab(v_1-m_1), && (9.44)\\
B &= b^2(v_1-m_1)+ab(m_2-v_2), && (9.45)\\
C &= ab(v_1m_1 - m_1^2 - v_1v_2 + v_2m_1 - m_1m_2 + v_2m_1 + v_1m_2 - v1v_2) && (9.46)\\
&\quad +a^2(m_1m_2 - v_2m_1 - v_2m_2 + v_2^2) + b^2(m_1^2 - v_1m_1 + v_1^2 - v_2m_1),\\
D &= ac(m_2-v_2)+bc(v_1-m_1)+a^2(v_1v_2-v_1m_2) && (9.47)\\
&\quad +b^2(v_2m_1 - v_1v_2) + ab(m_1v_1 - v_1^2 - v_2m_2 + v_2^2).
\end{aligned}
$$

[13]Wiederum gilt: $(a_{ij})(t_i) = (\sum_{(v,A)\in\mathcal{Z}}(a'_{ij}))(t_i) = \sum_{(v,A)\in\mathcal{Z}}(b'_i) = (b_i)$.

Somit lassen sich t_1, t_2 und θ direkt angeben:

$$t_1 = \frac{b_2a_{13}a_{32}+b_3a_{12}a_{23}-b_3a_{13}a_{22}-b_2a_{12}a_{33}+b_1a_{22}a_{33}-b_1a_{23}a_{32}}{a_{12}a_{31}a_{23}+a_{11}a_{22}a_{33}-a_{11}a_{23}a_{32}-a_{13}a_{31}a_{22}-a_{12}a_{21}a_{33}+a_{21}a_{13}a_{32}}, \tag{9.48}$$

$$t_2 = \frac{b_3a_{21}a_{13}+b_1a_{31}a_{23}-b_3a_{11}a_{23}-b_1a_{21}a_{33}+b_2a_{11}a_{33}-b_2a_{13}a_{21}}{a_{12}a_{31}a_{23}+a_{11}a_{22}a_{33}-a_{11}a_{23}a_{32}-a_{13}a_{31}a_{22}-a_{12}a_{21}a_{33}+a_{21}a_{13}a_{32}}, \tag{9.49}$$

$$\theta = \frac{b_2a_{12}a_{31}+b_1a_{21}a_{32}-b_1a_{31}a_{22}-b_2a_{11}a_{32}+b_3a_{11}a_{22}-b_3a_{12}a_{21}}{a_{12}a_{31}a_{23}+a_{11}a_{22}a_{33}-a_{11}a_{23}a_{32}-a_{13}a_{31}a_{22}-a_{12}a_{21}a_{33}+a_{21}a_{13}a_{32}}. \tag{9.50}$$

Bemerkung 9.3 *Da ein Pixel in der globalen Karte, durch die Quantisierung, einer Grundfläche von* $10cm \cdot 10cm$ *entspricht, wird in den obigen Verfahren mit Subpixelgenauigkeit gerechnet.*

9.7 Mathematischer Lösungsweg

Soll nun die Problemstellung aus Abschnitt 9.6 mathematisch korrekt gelöst werden, d.h. es wird über alle Bildpunkte und Geraden minimiert ohne eine vorhergehende Zuordnung zu treffen, ergibt sich folgendes Minimierungsproblem[14]:

$$MIN \sum_{v\in\mathcal{V}} \sum_{A\in\mathcal{A}} d(R(\theta)\cdot(v-m)+(m+t), A). \tag{9.51}$$

Dieses Minimierungsproblem besitzt $|\mathcal{V}|\cdot|\mathcal{A}|$ Gleichungen. Da in der Regel die Kardinalität der Mengen durch $|\mathcal{V}| > 300$ und $|\mathcal{A}| > 5$ gegeben ist, ergibt sich somit ein Gleichungssystem mit mindestens $300 \cdot 5 = 1500$ Gleichungen. Dieses relativ große Gleichungssystem läßt sich beim derzeitigen Stand der Technik nicht zufriedenstellend schnell lösen. Dies führt zu einer iterativen Lösungsstrategie.

9.8 Iterativer Lösungsweg

Beim iterativen Lösungsweg wird jedem Bildpunkt, wie in Abschnitt 9.4 beschrieben, die nächstliegende Gerade zugeordnet. Anschließend wird auf der Basis dieser Zuordnung das Minimierungsproblem gelöst. Dieses besteht nun aus einem Gleichungssystem mit zwei bzw., bei Hinzunahme der Rotation, drei Gleichungen. Das gewonnene Ergebnis wird auf das Bild[15] angewendet, d.h. das Bild wird entsprechend verschoben und, falls erforderlich, auch rotiert. Nun kann eine erneute Zuordnung stattfinden mit anschließender Lösung des Gleichungssystems und Anwendung des Ergebnisses auf das Bild. Dies wird solange wiederholt, bis das Verfahren konvergiert. Aus diesen Betrachtungen ergeben sich die folgenden wesentlichen Schritte:

[14] Für die Lösung des Minimierungsproblems aus Abschnitt 9.5 ergibt sich Entsprechendes.
[15] In diesem Fall die lokale Karte.

1. Bildung der Menge $\mathcal{Z} \subseteq \mathcal{V} \times \mathcal{A}$ gemäß Ausdruck (9.8).
2. Lösung des Minimierungsproblems 9.11 bzw. 9.25.
3. Anwendung des Ergebnisses auf die Bilddaten.
4. Wiederholung der Schritte 1 - 3, bis das Verfahren konvergiert.

Bemerkung 9.4 *In Schritt 3 des obigen Verfahrens wird, im Gegensatz zur Lösung des Minimierungsproblems 9.25, keine Pseudo-Rotation sondern die übliche Rotationsmatrix (Ausdruck (9.24)) verwendet.*

9.9 Bestimmung der Varianz

Da aus der Bestimmung des Rotations- und Translationsvektors nicht unmittelsbar eine Güte der Position hervorgeht, muß die Varianz mittels heuristischer Verfahren bestimmt werden. Hierfür wird die Genauigkeit der odometrischen Position im Kurzstreckenbereich genutzt. Da die Zykluszeit der Driftkompensation in der Regel ca. eine Sekunde beträgt, kann der Roboter, ausgehend von einer Maximalgeschwindigkeit von $100mm/s$, höchstens $10cm$ zurückgelegt haben. In diesem kleinen Bereich ist die durch die Sensoren gemessene Strecke noch relativ genau. Dies wiederum hat zur Folge, daß Korrekturen, die weiter von der odometrischen Position entfernt liegen, mit einer größeren Unsicherheit behaftet sind. Als Annäherung an die Varianz kann somit der quadrierte Verschiebungsvektor dienen.

9.10 Experiment

Das in diesem Kapitel vorgestellte Verfahren ist in einer Unix-Umgebung auf einem *RWI*-Roboter in der Programmiersprache C++ implementiert. In dem folgenden Experiment wird zum einen das Raum-zu-Raum-Fahrverhalten und zum anderen die Langstreckentauglichkeit des Systems getestet. Dabei untersucht das Verfahren eine Umgebung von 3 Metern um die vermutete Roboter-Position auf Drift bzw. Schlupf[16]. Abbildung 9.7 zeigt die für den Testlauf verwendete Strecke. Der zurückgelegte Weg ist dabei jeweils in weiß (reine odometrische Daten) bzw. schwarz (korrigierte Position) in der Karte eingetragen. Ausgehend vom oberen Raum soll der Roboter in den unteren Raum fahren. Dies kann bei einer starken Verschiebung der Positionsdaten gerade im Türbereich zu einem Fehlverhalten des Roboters und schließlich zum Versagen der Navigation führen. Während das System "glaubt", direkt vor der Tür zu stehen, befindet es sich in der Realität stark versetzt zu ihr. Im anderen Fall glaubt der Roboter den Türbereich schon

[16] D.h. der Fensterausschnitt beträgt 3 Meter.

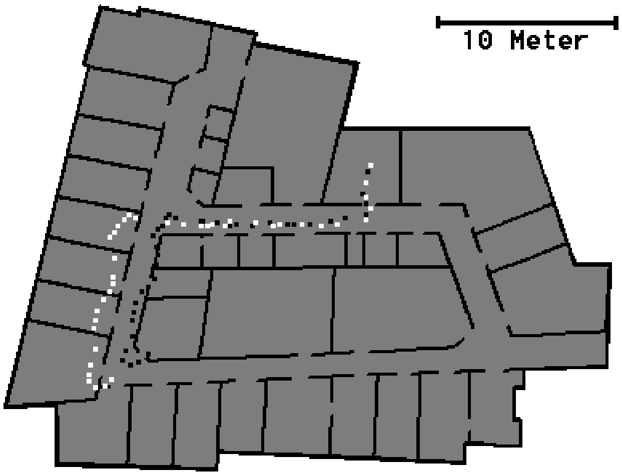

Abbildung 9.7: Die Teststrecke

passiert zu haben, steht aber noch davor. In beiden Fällen kann es zu Fehlplanungen kommen, die ihn nicht an das gewünschte Ziel führen.

Ein Problem, welches bei längeren Gängen immer wieder auftritt, ist die Berechnung der in Gangorientierung liegenden Komponente des Driftkompensations-Vektors. Die Hinzunahme der Mittelpunkte der einzelnen Liniensegmente stellt eine weitere Information für das System dar, die Fehlberechnungen entgegenwirkt. Kommt es trotzdem zu leichten Korrektur-Verschiebungen im Bezug auf die reale Position, wie dies in diesem Experiment der Fall ist[17], so gleicht das Verfahren den Fehler mit der Zeit wieder aus (siehe Abbildung 9.7).

In Abbildung 9.8 oben ist der Driftkompensations-Vektor mit seinen einzelnen Komponenten dargestellt. Dabei wird auf die Rotations-Komponente gänzlich verzichtet, da diese während des gesamten Testlaufs nahe null ist. In der linken oberen Grafik, in der nur die x-Komponente über die Zeit aufgetragen ist, läßt sich deutlich das Ansteigen des Schlupf-Fehlers erkennen. Erst als der Roboter seine Orientierung von der x- zur y-Richtung ändert, bleibt der Fehler konstant bzw. nimmt langsam ab. Genau umgekehrt verhält es sich mit der y-Komponente (Abbildung 9.8 rechts oben). Diese nimmt genau ab dem Zeitpunkt stark zu, ab dem die x-Komponente nur noch stagniert. Schließlich ist in der unteren Grafik, die die beiden Komponenten gleichzeitig darstellt, zu sehen, daß der Roboter im horizontalem Gang bezüglich der x- und im vertikalen bezüglich der y-Richtung einen sehr hohen Schlupf aufzuweisen hat. Dieser wird mit Hilfe des Driftkompensations-Vektors wieder ausgeglichen. Die Graphik stellt auch hier, wie im Fall des Hough-basierten Ansatzes, die Tendenz des Vektors dar. Falls die

[17]Der Korrekturvektor ist in der x-Komponente um ca. 45 Zentimeter zu groß.

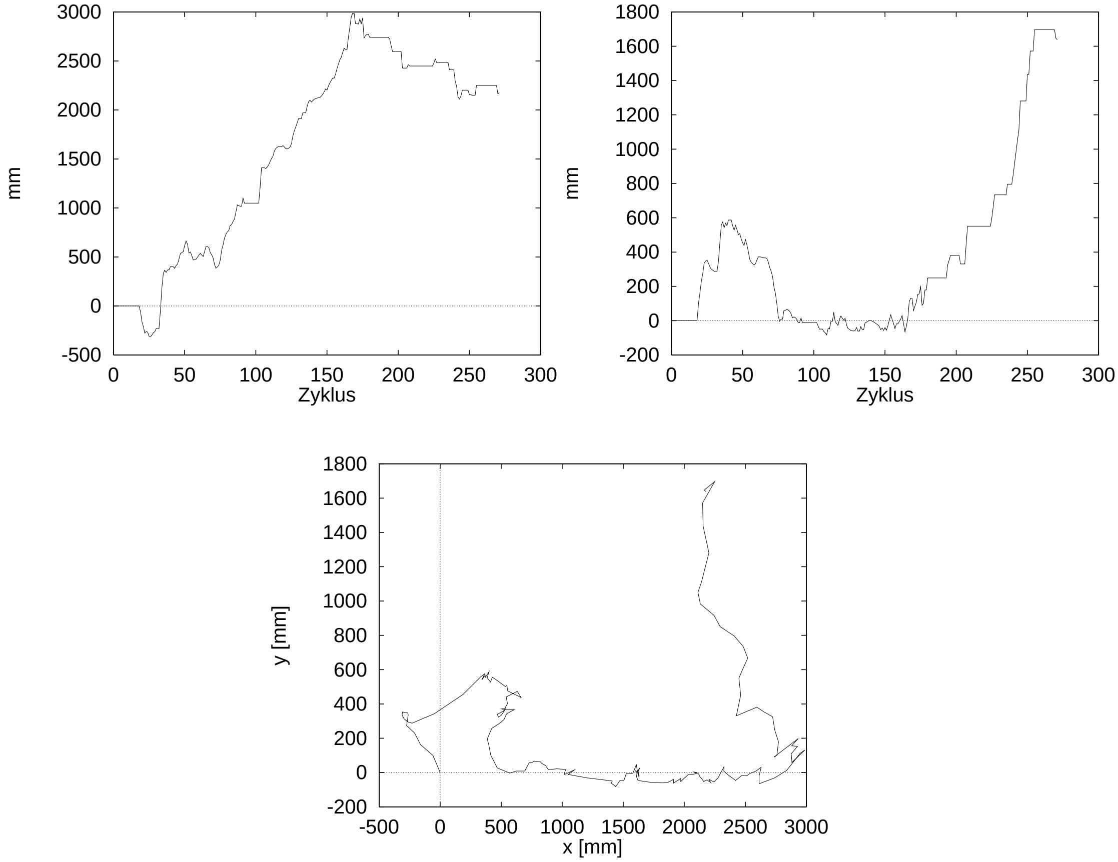

Abbildung 9.8: Der Driftkompensations-Vektor

Odometrie die Position während des gesamten Testlaufes genau liefert, sollte der Driftkompensations-Vektor nur um den Ursprung oszillieren, da keine Korrektur notwendig ist.

In den oberen Grafiken der Abbildung 9.9, in der jeweils die x- und y-Komponente der Position in Abhängigkeit des Lokalisations-Zyklus dargestellt ist, läßt sich der systematische Fehler bezüglich den odometrischen gegenüber den korrigierten Daten erkennen[18]. Abbildung 9.9 unten zeigt letztendlich den zurückgelegten Weg auf Basis der reinen odometrischen und der korrigierten Positionsdaten. Auch hier kann beobachtet werden, daß der Fehler immer in der Haupt-Fahrt-Richtung stark zunimmt, während die Drift bezüglich den restlichen Orientierungen verschwindend gering ausfällt.

Bemerkenswert ist in diesem Zusammenhang, daß, im Gegensatz zu den von Thrun gemachten Beobachtungen in [Thrun 1996], der Fehler in der Rotation des Roboters verschwindend gering ist. Dies kann unter anderem an der Beschaffen-

[18]Der große Schlupf ist auch in den weiteren Experimenten im Anhang gut zu erkennen.

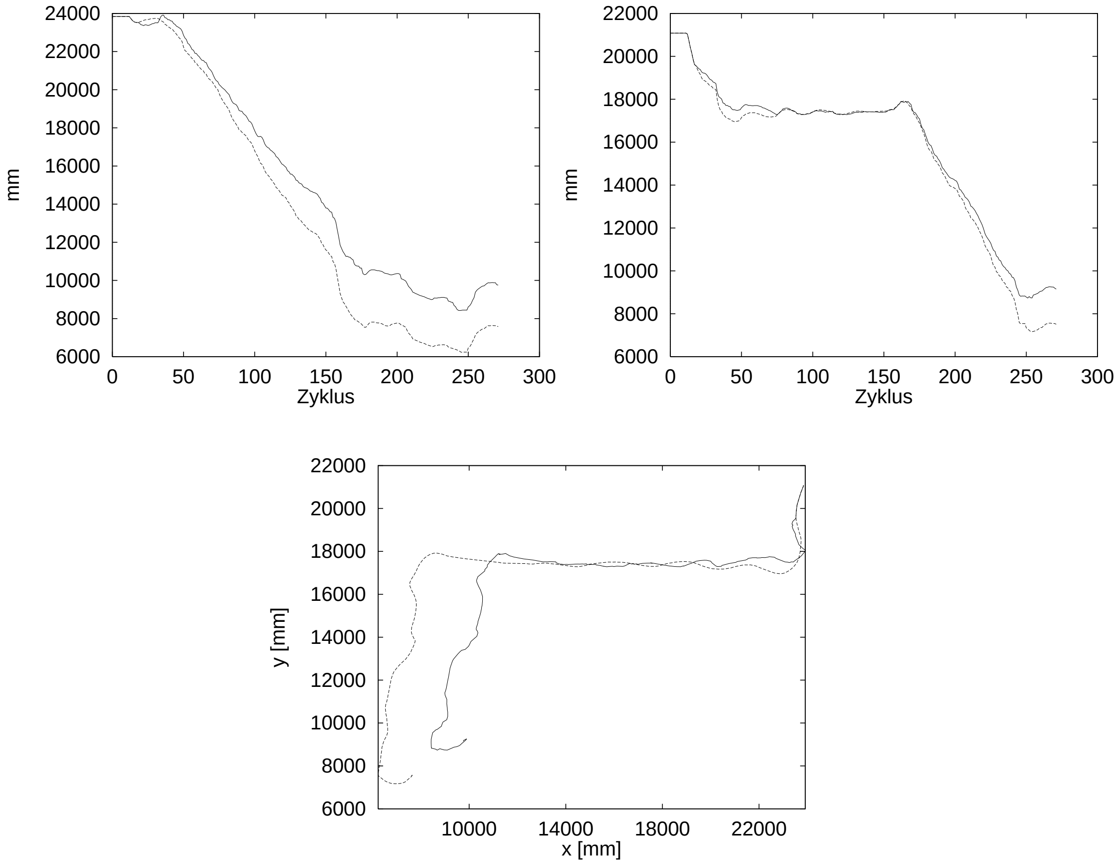

Abbildung 9.9: Die odometrische und die korrigierte Position

heit des Bodens liegen. Während Thruns Roboter auf Estrich-Boden manövrierte, wurde dieser Testlauf auf dünner Auslegware ausgeführt. Bei dieser kann es bei langen Strecken und höheren Geschwindigkeiten leicht zu einem Durchdrehen der Antriebsräder kommen. Ein zweiter möglicher Grund für das Auftreten der Abweichung könnte in einer fehlerhaften Skalierung der globalen Karte begründet liegen. Dieser Schlupf wird vollständig durch das Verfahren kompensiert.

Bemerkung 9.5 *Weitere Experiment-Daten sind im Anhang C zu finden.*

Kapitel 10

Raum-Matching-Algorithmus

Dieses letzte hier vorgestellte Verfahren eignet sich nur für eine Driftkompensation innerhalb von rechtwinkeligen Räumen und ist daher nur eingeschränkt mit den bisherigen Ansätzen vergleichbar. Trotzdem zeigt es, daß auch relativ einfache Ansätze durchaus ihre Berechtigung haben.

Im wesentlichen werden für die vier Orientierungen $0°$, $90°$, $180°$ und $270°$ Entfernungskarten gebildet, die den Abstand eines Bildpunktes zur, in dieser Richtung, nächstliegenden Wand wiedergeben. Somit ergibt sich für jeden Bildpunkt der globalen Karte ein 4-stelliger Vektor. Zur Laufzeit werden dann die entsprechenden Sensorwerte für die jeweiligen Orientierungen zu einem Vektor zusammengefaßt. Dieser spiegelt die aktuelle Position des Roboters wieder. Anschließend kann durch ein geeignetes Matching-Verfahren[1] die Position des Roboters innerhalb des Raumes bestimmt werden.

Bemerkung 10.1 *Da die zur Verfügung stehende Umgebung nicht rechtwinkelig ist, werden auf Experimente in diesem Kapitel verzichtet.*

10.1 Bildung der Entfernungskarten

Die Bildung der Entfernungskarten basiert auf der pixelorientierten globalen Karte. Um nun für die Orientierungen $0°$, $90°$, $180°$ und $270°$ eine Richtungskarte zu erstellen, werden für jeden Punkt p der globalen Karten die folgenden Schritte ausgeführt:

1. Setze θ auf $0°$ und i auf 1.
2. Ausgehend von p wird in Richtung θ die nächstliegende Wand gesucht. Der resultierende Wert d stellt die Entfernung zu dieser Wand dar und wird in dem Vektor E_p an der i-ten Position gespeichert, d.h. $E_p[i] = d$.
3. Erhöhe θ um $90°$ und i um 1.

[1] Beispielsweise der euklidische Abstand zwischen den Vektoren

4. Falls θ kleiner 360° ist, wiederhole Schritte 2 - 3.

Nach Abschluß des Verfahrens liegt für jeden Bildpunkt p der globalen Karte ein Vektor E_p vor, der für die jeweilige Orientierung den kürzesten Abstand zum nächsten Hindernis[2] wiedergibt.

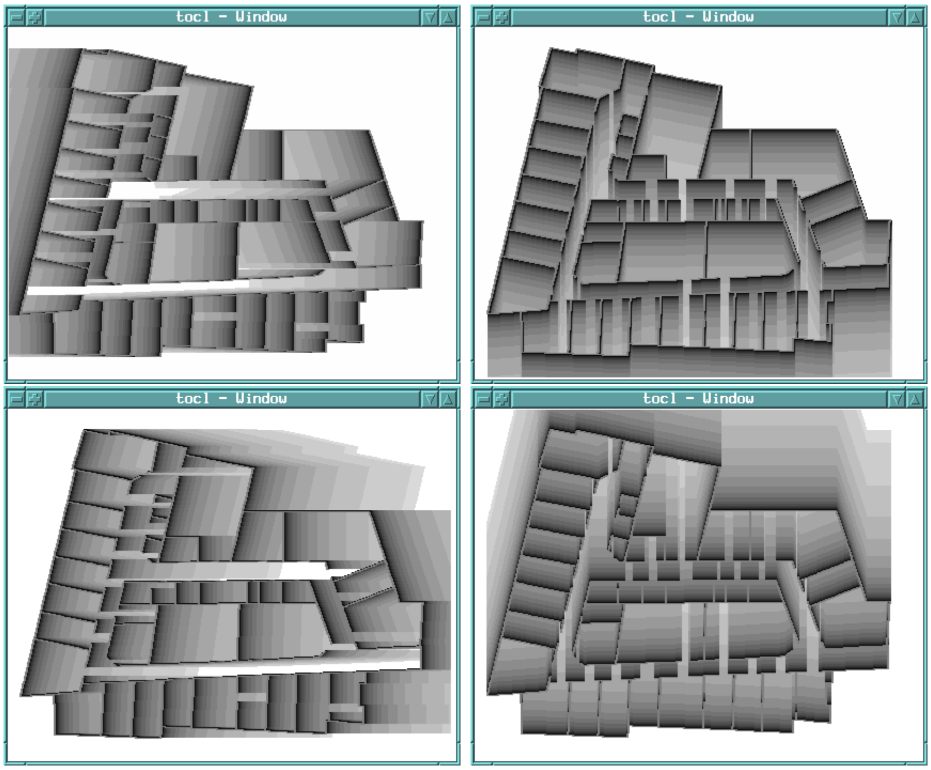

Abbildung 10.1: Entfernungskarten für die Orientierungen 0°, 90°, 180° und 270°

In Abbildung 10.1 sind die einzelnen Komponenten des Vektors E_p visualisiert. Die Farbwerte der einzelnen Bildpunkte spiegeln die Entfernung zur nächstliegenden Wand wieder, wobei diese einen logarithmischen Verlauf haben und hellere Farbwerte einen größeren Abstand zum nächsten Hindernis bedeuten. Es läßt sich deutlich an den weißen Flächen erkennen (linke Abbildungen), daß die Entfernungswerte in den Gängen unbrauchbar sind, da diese die Entfernungsauflösung der Sensoren übersteigen.

10.2 Sensordaten-Matching

Nachdem die Entfernungskarten in einem Vorverarbeitungs-Schritt erstellt wurden, können diese nun zur Positions-Bestimmung des Roboters herangezogen wer-

[2] In diesem Fall nur Wände.

den. Hierzu wird in der lokalen Karte[3], ähnlich dem oben beschriebenen Verfahren, für die Orientierungen 0°, 90°, 180° und 270° der Entfernungsvektor E_{Robot} gebildet. Für den unmittelbaren Vergleich der Vektoren E_{Robot} und E_p bietet sich als erster, naiver Ansatz der euklidische Abstand an:

$$M_p = \|E_{Robot} - E_p\|_2. \tag{10.1}$$

Um die Position des Roboters bestimmen zu können, wird die unmittelbare Umgebung um die odometrische Position O_{Robot} als Suchraum $\mathcal{U}(O_{Robot})$ gewählt. Für jeden Punkt q innerhalb des Suchraumes wird das zugehörige Abstandmaß M_q bestimmt. Die gesuchte Position P_{Robot} des Roboters stellt nun das Minimum aller auf dieser Weise ermittelten Abstandmaße dar:

$$p = P_{Robot} \Rightarrow (\forall q \in \mathcal{U}(O_{Robot}) \ : \ M_p < M_q). \tag{10.2}$$

10.3 Nachteile des Verfahrens

Trotz des sehr schnellen Matching-Algorithmus hat diese Vorgehensweise auch eine Reihe von Nachteilen. So ist das Verfahren nur in rechtwinkeligen Räumen anwendbar. In Gängen kann nicht einmal annäherungsweise eine genaue Schätzung der aktuellen Position abgegeben werden[4]. Weiterhin eignet sich das Verfahren in dieser Form nur für eine Translationskorrektur und nicht zu der dazugehörigen Rotationskorrektur. Zuletzt ist es im allgemeinen sinnvoller Seitenverhältnisse, d.h. das Verhältnis der Entfernungen bei 0° und 180° bzw. bei 90° und 270°, an Stelle des euklidischen Abstands als Güte heranzuziehen. Bei letzterem fließt die Orientierung nur implizit ein. Im nächsten Abschnitt werden nun Lösungsvorschläge für die obigen Problemstellungen besprochen.

10.4 Optimierungs-Möglichkeiten

Zunächst muß das Problem der Sensor-Ungenauigkeit gelöst werden. Unterschiedliche Material-Oberflächen haben im Bezug auf Schall ein unterschiedliches Abstrahlungs-Verhalten (Absorber-Verhalten). So haben glatte Materialien[5] eher einen kleinen Bereich, indem der Sensor sie zuverlässig detektiert. Abbildung 10.2 stellt diesen Zusammenhang dar. Das Material kann nur innerhalb des durch θ bestimmten Kegels vom Sensor zuverlässig detektiert werden. Ist der Sensor außerhalb dieses Kegels, sind die gemessenen Entfernungswerte nicht aussagekräftig.

[3] Diese stellt den Sensoreindruck des Roboters dar.

[4] Es stellt sich dann das Problem der Gang-Erkennung dar, um fehlerhafte Messungen ausschließen zu können. Dieses Problem ist, ausgehend von der lokalen Karte, nicht trivial.

[5] Wie z.B. Glas, Metalle, kunstoffbeschichtete Türen usw.

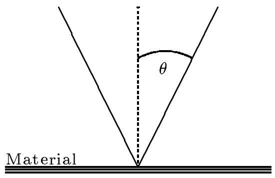

Abbildung 10.2: Absorber-Verhalten von Materialien

Das obige Problem wirkt sich nachteilig auf die fest vorgegebenen Orientierungen aus. Im ungünstigsten Fall liegen alle vier für die Entfernungsmessung verantwortlichen Sensoren außerhalb des Bereiches, in der sie das nächstliegende Hindernis zuverlässig detektieren können. Der somit gewonnene Vektor ist für die Positions-Bestimmung unbrauchbar. Da nach Auslegung der Sensoren keine zu kurz gemessenen Entfernungen auftreten können, bietet es sich an, möglichst kurze Entfernungen in den Vektor einzubringen. Dies wird durch eine optimierte Ausrichtung der einzelnen Orientierungen erreicht. Anstatt die vorweg fest definierten Richtungen 0°, 90°, 180° und 270° heranzuziehen, werden diese solange gedreht, bis sie orthogonal zu den Wand-Orientierungen stehen. Dies ist natürlich nur möglich, wenn der entsprechende Raum rechtwinklig ist. Jeder Bildpunkt hat dann in dem zugehörigen Vektor nur die kürzesten Abstände zu den nächstliegenden Wänden gespeichert. Für die Roboter-Position gilt, wie weiter oben schon erwähnt, daß kleine Sensormeßwerte einen höheren Zuverlässigkeitsgrad besitzen. Werden diese für den Vektor herangezogen, so ist es möglich, innerhalb eines kleinen Bereiches nicht nur eine Translations- sondern auch eine Rotationskorrektur durchzuführen. Für den Vergleich des Roboter-Vektors mit den Vektoren der globalen Karte kann an Stelle des euklidischen Abstand auch ein Verhältnis der einzelnen Seiten eingehen:

$$h_p = \frac{E_p[1]}{E_p[3]}, \tag{10.3}$$

$$h_{Robot} = \frac{E_{Robot}[1]}{E_{Robot}[3]}, \tag{10.4}$$

$$v_p = \frac{E_p[2]}{E_p[4]} \quad \text{und} \tag{10.5}$$

$$v_{Robot} = \frac{E_{Robot}[2]}{E_{Robot}[4]}. \tag{10.6}$$

Ein Matching zwischen Roboter- und Kartenvektor stellt sich als gewichtete Summe der Verhältnis-Abstände dar:

$$M_p = \alpha \cdot |h_p - h_{Robot}| + \beta \cdot |v_p - v_{Robot}|, \tag{10.7}$$

wobei α und β Skalierungsfaktoren sind. Die letztendliche Bestimmung der Roboter-

Position bleibt unverändert:

$$p = P_{Robot} \Rightarrow (\forall q \in \mathcal{U}(O_{Robot}) \;:\; M_p < M_q). \tag{10.8}$$

Die unmittelbare Umgebung um die odometrische Position des Roboters, O_{Robot}, wird durch $\mathcal{U}(O_{Robot})$ dargestellt.

Kapitel 11

Kalman-Filter

Liegen Meßwerte mit gegebener Genauigkeit vor, so stellt sich häufig das Problem einer geeigneten Kombination dieser Werte. Kalman-Filter stellen eine Lösung dieses Problems dar.

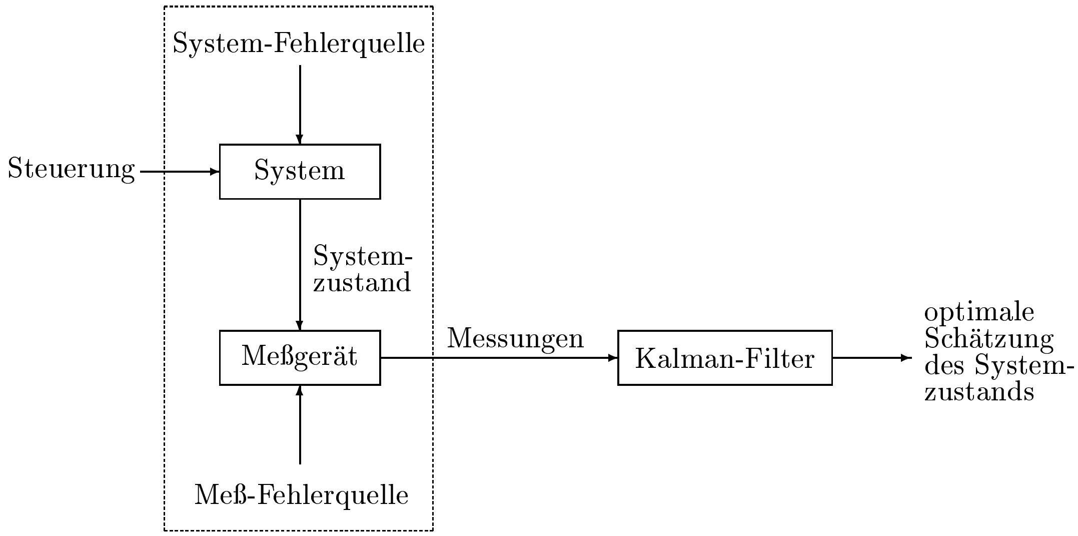

Abbildung 11.1: Eine Kalman-Filter-Anwendung (aus [Maybeck])

Abbildung 11.1 zeigt den konzeptionellen Aufbau einer Kalman-Filter-Anwendung. Der Systemzustand soll zu jedem Zeitpunkt an Hand der Ein- und Ausgaben berechnet werden.

11.1 Einführung

Wie eingangs schon erwähnt bzw. in [Maybeck] beschrieben, kombiniert ein Kalman-Filter alle zu einem Zeitpunkt vorhandenen Meßdaten mit dem Vorwissen über das System und der Art der Sensoren, um eine Schätzung der gesuchten Größe abzugeben. Der damit behaftete Fehler wird mit zunehmender Zeit statistisch minimiert. Es ist sinnvoll hierfür eine bedingte Wahrscheinlichkeitsdichte einzuführen, d.h. daß die Dichtefunktion von dem jeweiligen Meßwert abhängig ist. Sind die Meßwerte mit einem genauen Sensor aufgenommen, so nimmt die Wahrscheinlichkeit für Werte mit kleiner werdendem Abstand zum Meßwert stark zu. Andererseits nimmt bei ungenauen Werten die Wahrscheinlichkeit mit zunehmender Entfernung vom Meßwert nur langsam ab.

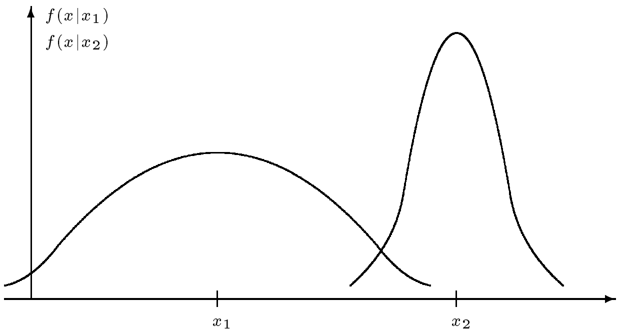

Abbildung 11.2: Wahrscheinlichkeitsdichte bei zwei gegebenen Meßwerten

Abbildung 11.2 verdeutlicht diesen Zusammenhang an Hand von zwei Meßwerten. Während bei x_1 ein ungenauer Sensor verwendet wurde und somit die Messung einer vorgegebenen Unsicherheit unterliegt, ist x_2 mit einem höherauflösendem Sensor aufgenommen worden. Der Funktionswert $f(x|x_1)$ gibt die Wahrscheinlichkeit an, daß die Position x die gesuchte Position x_1 ist ($f(x|x_2)$ analog). Es sind deutlich die zwei daraus resultierenden Wahrscheinlichkeitsdichten zu erkennen. Mit zunehmenden Abstand von x_1 verändert sich die Wahrscheinlichkeit des Wertes nur geringfügig. Bei Werten, die um x_2 angesiedelt sind, existiert nun genau der entgegengesetzte Fall. Bei größerer Entfernung nimmt die Wahrscheinlichkeit der Werte rapide ab. Weiterhin ist zu erkennen, daß die Amplitude bei x_2 viel größer als bei x_1 ist. Dies folgt unmittelbar mit Ausdruck (11.1) und der Forderung, daß 68.3% des Wahrscheinlichkeits-Gewichtes[1] innerhalb der σ-Umgebung um den Meßwert liegen müssen (Ausdruck (11.2)):

$$\int_{-\infty}^{+\infty} f(x|x_i) = 1 \quad \text{und} \tag{11.1}$$

$$\int_{x_i-\sigma_{x_i}}^{x_i+\sigma_{x_i}} f(x|x_i) = 0.683. \tag{11.2}$$

[1] Das Gewicht ergibt sich aus der Summierung der einzelnen Positions-Wahrscheinlichkeiten (im kontinuierlichen Fall entspricht dies Ausdruck (11.2)).

Bemerkung 11.1 *σ_{x_i} ist ein direktes Maß der Meßunsicherheit, d.h. je größer die Streuung σ_{x_i} ist, desto niedriger ist die Amplitude und somit die Gaußfunktion stärker gestreckt.*

Kalman-Filter propagieren nun diese bedingten Wahrscheinlichkeitsdichten in Systemen, die durch ein lineares Modell beschrieben werden können. System- und Meßwertrauschen müssen sich dann als weißes bzw. gaußförmiges Rauschen darstellen lassen, wobei weißes Rauschen die Unkorreliertheit mit der Zeit impliziert. Es werden daher keine Annahmen über die Stärke des weißen Rauschens gemacht.

11.2 Positionsbestimmung

Im Fall der Robotik eignen sich Kalman-Filter hervorragend als Hilfsmittel für die Bestimmung der aktuellen Position. Sei beispielsweise eine Position (x_1, y_1, θ_1) des Roboters mit einer Standardabweichung von $(\sigma_{x_1}, \sigma_{y_1}, \sigma_{\theta_1})$ gegeben. Die bedingte Wahrscheinlichkeit $x(t_1)$ kann durch die Messung (x_1, y_1, θ_1) bestimmt werden. Die Dichteverteilung der Position zum Zeitpunkt t_1 ergibt sich direkt aus der Standardabweichung und (11.2), da für die Meßwerte ein gaußförmiges Rauschen angenommen wird.

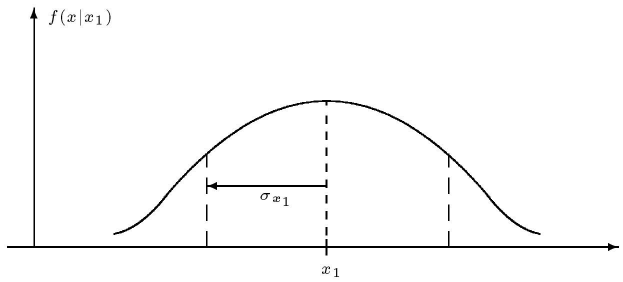

Abbildung 11.3: Positions-Wahrscheinlichkeit (hier nur die x-Komponente)

Wie in Abbildung 11.3 zu sehen ist, ist die Positions-Wahrscheinlichkeit an der Stelle $x = x_1$ am größten[2]. Für alle weiteren Positionen nimmt die Wahrscheinlichkeit entsprechend der Funktion $f(x|x_1)$ in Abhängigkeit der Messung (x_1, y_1, θ_1) ab. Es ergibt sich somit zum Zeitpunkt t_1 als genaueste Schätzung der aktuellen Position

$$\hat{x}(t_1) = (x_1, y_1, \theta_1). \tag{11.3}$$

Für die Varianz des Meßwertfehlers gilt:

$$\sigma_x^2(t_1) = (\sigma_{x_1}^2, \sigma_{y_1}^2, \sigma_{\theta_1}^2). \tag{11.4}$$

[2]Für die weiteren Komponenten y_1 und θ_1 gilt entsprechendes.

Bemerkung 11.2 *Beim Vorliegen nur eines Meßwertes ist $\hat{x}$ Modalwert, Median und Mittelwert zugleich.*

Wird nun nach einer bestimmten Zeit ein zweiter Meßwert erhoben, muß die zurückgelegte Strecke berücksichtigt werden. Sei dazu folgendes einfaches Bewegungsmodell

$$\frac{\partial x}{\partial t} = u + w \tag{11.5}$$

gegeben. Die Geschwindigkeit sei dabei mit u bezeichnet. w sei der Rauschterm, der das fehlende Wissen der tatsächlichen Geschwindigkeit wiederspiegelt. Im allgemeinen wird dieser als Gaußglocke mit Modalwert null und Varianz σ_w^2 modelliert. Kurz vor der Messung zum Zeitpunkt t_2 kann eine Schätzung der voraussichtlichen Position zum Zeitpunkt t_2^- abgegeben werden:

$$\begin{aligned} \hat{x}(t_2^-) &= \hat{x}(t_1) + u[t_2 - t_1] \quad \text{und} &(11.6)\\ \sigma_x^2(t_2^-) &= \sigma_x^2(t_1) + \sigma_w^2[t_2 - t_1]. &(11.7)\end{aligned}$$

Die beste Vorhersage der Position zum Zeitpunkt t_2^- stellt der Wert $\hat{x}(t_2^-)$ dar. Die erwartete Varianz für die abgegebene Schätzung wird durch $\sigma_x^2(t_2^-)$ angegeben. Wird nun eine Messung zur Zeit t_2 genommen, sind wiederum zwei gauß'sche Verteilungen, die Informationen über die Position beinhalten, vorhanden. Diese werden in der Form

$$\hat{x}(t_2) = \hat{x}(t_2^-) + \mathcal{K}(t_2) \cdot [x_2 - \hat{x}(t_2^-)] \tag{11.8}$$

miteinander kombiniert. Für die Varianz ergibt sich:

$$\sigma_x^2(t_2) = \sigma_x^2(t_2^-) - \mathcal{K}(t_2) \cdot \sigma_x^2(t_2^-). \tag{11.9}$$

Der Kalman-Faktor $\mathcal{K}$ berechnet sich dann wie folgt:

$$\mathcal{K}(t_2) = \frac{\sigma_x^2(t_2^-)}{\sigma_x^2(t_2^-) + \sigma_{x_2}^2}. \tag{11.10}$$

11.3 Der Algorithmus

Im folgenden wird der Pseudo-Code für die Bestimmung einer dreidimensionalen Position mittels eines Kalmanfilters angegeben.

Variablen:

odo	odometrische Position, abgelegt in einer Struktur mit den Komponenten x, y, rot sowie deren Varianz
kp	kalmangefilterte Position, abgelegt in einer Struktur

	mit den Komponenten x, y, rot sowie deren Varianz
pos	gemessene Position, abgelegt in einer Struktur mit den Komponenten x, y, rot sowie deren Varianz
est	geschätzte Position, abgelegt in einer Struktur mit den Komponenten x, y, rot sowie deren Varianz
$diff$	geschätzter zurückgelegter Weg, abgelegt in einer Struktur mit den Komponenten x, y, rot
β	Streuungsfaktor

Code:

$diff.x(t+1) = odo.x(t+1) - kp.x(t)$
$diff.y(t+1) = odo.y(t+1) - kp.y(t)$
$diff.rot(t+1) = odo.rot(t+1) - kp.rot(t)$

$est.x(t+1) = kp.x(t) + diff.x(t+1)$
$est.y(t+1) = kp.y(t) + diff.y(t+1)$
$est.rot(t+1) = kp.rot(t) + diff.rot(t+1)$
$est.x.\sigma^2(t+1) = kp.x.\sigma^2(t) + ((diff.x(t+1)) \cdot \beta)^2$
$est.y.\sigma^2(t+1) = kp.y.\sigma^2(t). + ((diff.y(t+1)) \cdot \beta)^2$
$est.rot.\sigma^2(t+1) = kp.rot.\sigma^2(t) + ((diff.rot(t+1)) \cdot \beta)^2$

$\mathcal{K}x(t+1) = (est.x.\sigma^2(t+1))/(est.x.\sigma^2(t+1) + pos.x.\sigma^2(t+1))$
$\mathcal{K}y(t+1) = (est.y.\sigma^2(t+1))/(est.y.\sigma^2(t+1) + pos.y.\sigma^2(t+1))$
$\mathcal{K}rot(t+1) = (est.rot.\sigma^2(t+1))/(est.rot.\sigma^2(t+1) + pos.rot.\sigma^2(t+1))$

$kp.x(t+1) = est.x(t+1) + \mathcal{K}x(t+1) \cdot (pos.x(t+1) - est.x(t+1))$
$kp.y(t+1) = est.y(t+1) + \mathcal{K}y(t+1) \cdot (pos.y(t+1) - est.y(t+1))$
$kp.rot(t+1) = est.rot(t+1) + \mathcal{K}rot(t+1) \cdot (pos.rot(t+1) - est.rot(t))$
$kp.x.\sigma^2(t+1) = est.x.\sigma^2(t+1) - \mathcal{K}x(t+1) \cdot est.x.\sigma^2(t+1)$
$kp.y.\sigma^2(t+1) = est.y.\sigma^2(t+1) - \mathcal{K}y(t+1) \cdot est.y.\sigma^2(t+1)$
$kp.rot.\sigma^2(t+1) = est.rot.\sigma^2(t+1) - \mathcal{K}rot(t+1) \cdot est.rot.\sigma^2(t+1)$

Bemerkung 11.3 *Die odometrische Position stellt die Schätzung der Position zum Zeitpunkt t dar, deren Genauigkeit mit dem zurückgelegten Weg abnimmt.*

Kapitel 12

Vermeidung von Meßfehlern

Trotz der oben angegebenen relativ genauen Verfahren zur Driftkompensation, kann es immer noch zu Meß- bzw. Berechnungsfehlern kommen. Diese gilt es nun zu identifizieren und von dem Differenzvektor auszuschließen. Im folgenden werden die häufigsten Fehler angegeben.

12.1 Verlassen des Gültigkeitsbereiches

Das Verlassen des Gültigkeitsbereiches, z.B. der globalen Karte, muß in jedem Fall verhindert werden, da es ansonsten zu unvorhersehbaren Effekten kommen kann. Diese bestehen im wesentlichen aus einem nicht zulässigen Speicherzugriff auf die globale Karte. Ein genügend großer Rand, der diese säumt, verhindert in der Regel das Verlassen der globalen Karte. Tritt dieser Fall dennoch ein, so muß er abgefangen werden. Ein Korrekturvektor, der dies verursacht, wird so angepaßt, daß sich die neue, durch den Vektor festgelegte, Position innerhalb der globalen Karte befindet.

12.2 Korrektur-Vektor zu groß

In der Regel existieren zwei mögliche Analysemöglichkeiten für zu große Korrektur-Vektoren:

1. Der Korrektur-Vektor übersteigt den Suchbereich.

2. Der Korrektur-Vektor ist größer als die zurückgelegte Strecke.

Der erste Fall wird schon direkt bei der Driftkompensation abgefangen. Vektoren, die diese Eigenschaft besitzen, werden nicht zugelassen und fließen somit nicht in die Differenzvektor-Berechnung ein. Der zweite Fall wird bei der Bestimmung der neuen Position berücksichtigt. Da die Lokalisations-Zyklus-Zeit bekannt ist, kann die zurückgelegte Strecke S geschätzt werden. Seien dazu v_i und v_{i-1} die zwei zuletzt gelieferten Geschwindigkeiten und Δt die entsprechende Zeitdifferenz:

$$S \approx \Delta t \cdot \frac{v_i + v_{i-1}}{2}. \tag{12.1}$$

Übersteigt die x- bzw. die y-Komponente des Differenz-Vektors diesen Schätzwert, so wird die entsprechende Komponente auf die Schätzung angepaßt.

12.3 Driftkorrektur durch Wände

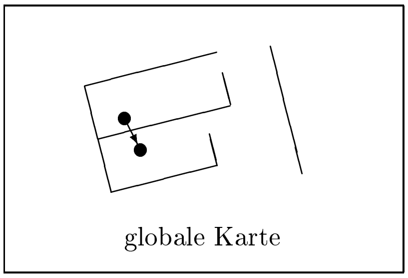

Abbildung 12.1: Das Room-Hopper Problem

Zuletzt muß die Driftkorrektur durch Wände hindurch verhindert werden. Es soll damit ein von Raum-zu-Raum-Springen (Room-Hopper) vermieden werden. Ein Room-Hopper führt insbesondere bei der geradenbasierten Driftkompensation zu Meßfehlern[1], die das Verfahren selbst nicht wieder korrigieren kann[2]. Abbildung 12.1 zeigt einen Ausschnitt der globalen Karte. Der Roboter, hier als schwarzer Punkt dargestellt, erhält einen Korrektur-Vektor, der zu einer Position innerhalb eines anderen Raumes führt. Dieser Vektor kreuzt somit mindestens eine Wand. Solche Korrektur-Vektoren sind nicht zulässig und werden von der Differenzvektor-Berechnung ausgeschlossen. Zur Überprüfung dieser Eigenschaft werden alle Bildpunkte der globalen Karte untersucht, die in Richtung des Korrektur-Vektors zwischen alter und neuer Position liegen. Der folgende Pseudo-Code verdeutlicht dies.

Variablen:

Karte globale Karte

[1] Auch beim Hough-basierten Ansatz kann dies zu schlechten Meßergebnissen führen.

[2] Falls beispielsweise zwei annähernd identische Räume nebeneinander liegen, so führt ein sogenannter Room-Hopper im ungünstigsten Fall dazu, daß die lokale Karte optimal mit dem falschen Raum gematcht wird.

odo_x, odo_y	vermutete Position
$dx, dy, drot$	Korrektur-Vektor
vx, vy	normierter Korrektur-Vektor
xx, yy	Laufvariablen
$tangente$	Tangente des Steigungsdreiecks (dx, dy)

Code:

$tangente = \sqrt{dx^2 + dy^2}$
$vx = dx/tangente$
$vy = dy/tangente$
$yy = odo_y$
<u>for</u> every $xx \in \{odo_x - |dx|, odo_x + |dx|\}$ <u>do</u>
  $xx = xx + vx$
  $yy = yy + vy$
  <u>if</u> $Karte(xx, yy) = 1$ <u>then</u>
    $dx = 0$
    $dy = 0$
    $drot = 0$
  <u>endif</u>
<u>endfor</u>

Kapitel 13

Vergleich der Verfahren

Es werden nun die vorgestellten Verfahren miteinander verglichen, um sie schließlich in einen breiteren Kontext bisheriger Lösungs-Strategien einzuordnen.

13.1 Hough- und geradenbasierter Ansatz

Dieser Abschnitt befaßt sich mit dem Vergleich der vorgestellten Ansätze zur Lösung des Driftkompensations-Problems. Dabei steht der Raum-Matching-Algorithmus nicht in unmittelbarer Konkurrenz zu den anderen beiden Verfahren, da dieser nur für eine Positions-Bestimmung innerhalb rechtwinkeliger Räume geeignet ist. Er wird deshalb im folgenden vernachlässigt.

In den ausführlichen Testreihen zur Hough-Transformation hat sich gezeigt, daß diese in markanten Umgebungen eine gute Schätzung der aktuellen Position liefert. Befindet sich der Roboter allerdings in einem Korridor, so versagt in der Regel das Verfahren. Dies liegt direkt am Hough-Ansatz, der eine Zuordnung zwischen Segment[1] und Bild[2] sucht. In Flur-Situationen führt das Matching-Problem daher zu einer relativ genauen Bestimmung der orthogonal zu Gang-Orientierung liegenden Komponente, während die jeweils andere nur schlecht bestimmt werden kann (s. Abbildung 8.7). Im ungünstigsten Fall wird eine Komponente des Driftkompensations-Vektors fortlaufend falsch bestimmt, so daß nach Verlassen des Ganges die Position zu stark von der realen abweicht. Der Roboter ist dann nicht mehr in der Lage, sich mittels der Driftkompensation zu relokalisieren.

Dieses Problem stellt die Frage nach einem geeigneten Verfahren, welches selbst in Gängen zu einer korrekten Bestimmung der Position führt. Der geradenbasierte Ansatz, wie er in Kapitel 9 beschrieben worden ist, gibt hierauf eine Antwort. Im Vergleich zur Hough-basierten Driftkompensation kann die geraden-

[1] die lokale Karte
[2] die globale Karte

basierte Version in charakteristischen Gängen[3] die Position sicher bestimmen. Dies liegt nicht zuletzt an dem neu eingeführtem Zuordnungs-Kriterium.

Abbildung 13.1 stellt die Problematik dar. Während links ein Ausschnitt aus der globalen Karte zu sehen ist, sind in der Mitte und rechts die entsprechenden lokalen Karten und deren Zuordnung aufgeführt. Der geradenbasierte Ansatz mit zusätzlichem Zuordnungskriterium beweist an dieser Stelle die Überlegenheit gegenüber der Hough-Transformation. Diese ist, ebenso wie der geradenbasierte Ansatz ohne Zusatzkriterium, nicht in der Lage, die vorhandene Tür im oberen Teil des Ganges als solche zu erkennen. Vielmehr stellt die Türöffnung lediglich eine Störung des Segmentes dar und die entsprechenden Punkte der lokalen Karte werden den Wand-Segmenten **A** und **B** zugeordnet (Abbildung 13.1 rechts).

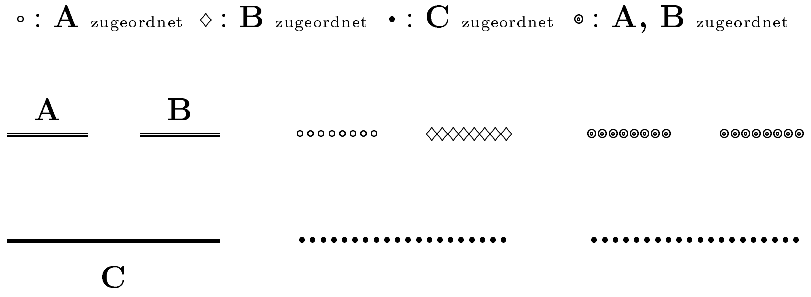

Abbildung 13.1: Problem der korrekten Lokalisation in Korridoren

Der geradenbasierte Ansatz verhält sich ohne das neu eingeführte Zuordnungskriterium nicht anders. Die den beiden Liniensegmenten **A** und **B** zugehörigen Geraden sind identisch. Der Abstand von Punkten in der lokalen Karte zu diesen ist deshalb gleich. Die Information, daß im oberen Teil des Korridors eine Tür existiert, geht somit verloren (Abbildung 13.1 rechts).

Die Hinzunahme des Liniensegment-Mittelpunktes führt schließlich zu einer korrekten Zuordnung zwischen Punkten der lokalen und Geraden der globalen Karte. Abbildung 13.1, Mitte, zeigt die Nutzung der zusätzlichen Information.

Zuletzt ist die Einbeziehung der Rotation ohne erheblichen Mehraufwand in der geradenbasierten Lokalisation ein weiterer wesentlicher Vorteil. Zum einen ist bei kleinen Fehlern der Odometrie[4] der Rechenaufwand wesentlich geringer, da der Driftkompensations-Vektor schnell gefunden werden kann. Zum anderen können, in ungünstigen Situationen, die Akkumulatorfelder der einzelnen Rotations-Stufen nahezu identisch sein. Eine Bestimmung der Orientierung des Roboters wird somit zusätzlich erschwert und führt unter Umständen zu falschen

[3] d.h. die eine ausreichende Anzahl von Türen besitzen.
[4] und somit einer kleinen Driftkorrektur, wie sie im allgemeinen üblich ist.

Ergebnissen.

Werden die Einschränkungen der Hough-Transformation zu Grunde gelegt, ergibt sich ein eindeutiger Vorteil des geradenbasierten Ansatzes. Dieser bezieht zum einen die Rotations-Komponente ein und zum anderen ist das Verfahren in der Lage, selbst in Fluren die Position richtig zu bestimmen. Da globale und lokale Karte stets in der gleichen Skalierung vorliegen, fällt die fehlende Skaleninvarianz nicht negativ auf.

13.2 Bisherige Driftkompensations-Verfahren

Der nun folgende Vergleich basiert im wesentlichen auf den in den Artikeln veröffentlichten Ergebnissen. Diese gelten in der Regel für relativ kleine Umgebungen. Die direkte Gegenüberstellung ist deshalb, bis auf die vorgestellten Experimente in [Burgard et al. 1996], nicht ohne weiteres möglich.

Wegen der besseren Driftkompensations-Eigenschaften des geradenbasierten Ansatzes in Korridoren gegenüber der Hough-Transformation, beziehen sich die folgenden Ergebnisse nur auf diesen.

Das erste Verfahren ist der in [Cox, Kruskal 1988] beschriebene Ansatz. Dieser nutzt einen Laserentfernungsmesser, um die unmittelbare Umgebung abzubilden. Als Testgebiet stand ein ca. $45m^2$ großer Raum zur Verfügung[5]. Der genauere Sensor und die wesentlich kleinere Umgebung stellen in diesem Fall einen klaren Vorteil dar. So ist der durchschnittliche Abstand zwischen Punkten des Sensoreindrucks und dem Modell auch nur ca. $5cm$ groß. Dies gilt für eine Messung bei Stillstand des Roboters. Die Art des Sensors erlaubt dem Verfahren allerdings nicht, Messungen während der Fahrt vorzunehmen. Zudem sagen diese Ergebnisse nichts über die tatsächliche Roboter-Position aus, die unter Umständen stark versetzt sein kann. Werden diese Ergebnisse trotzdem zu Grunde gelegt, ergibt sich eine Abweichung von einem Prozent bezüglich der Breite des Raumes.

Dies entspricht der Abweichung des geradenbasierten Ansatzes, der eine Verschiebung von nur 45 Zentimetern bei einer zurückgelegten Strecke von 45 Metern hat. Der Roboter selbst hingegen nimmt an Hand seiner unkorrigierten odometrischen Daten an, daß er 6 Meter versetzt von seiner realen Position steht. In Büroumgebungen entspricht dies in der Regel einem anderen Raum, d.h. das gewünschte Ziel kann nicht mehr angefahren werden. Bei der ermittelten Abweichung der korrigierten zu der realen Roboter-Position muß zudem die Quantisierung der globalen Karte berücksichtigt werden, die einem Pixel eine Grundfläche von $10cm \cdot 10cm$ zuordnet.

Eine Lokalisations-Iteration in [Holenstein et al. 1992] benötigt zehn Sekunden und ist nur während des Stillstands des Roboters möglich. Obwohl die ge-

[5] Der Raum hatte die Ausmaße $9m \cdot 5m$ und beinhaltete keine Möbel.

fundene Position in dem relativ kleinen Raum von $20m^2$ zufriedenstellend ist, sind die langsame Lokalisation und die Notwendigkeit des Roboter-Stillstandes nicht akzeptabel. Außerdem wird der Ansatz nicht in Gang-Situationen getestet, welche für die Stabilität des Verfahrens entscheidend sind.

Auch [Drumheller 1987] testet seinen Ansatz ausschließlich in kleinen Räumen. Dies ist nicht weiter verwunderlich, da die Sensoreindruck-Korrelation für Korridore ungeeignet ist. Obwohl nur ein Sensoreindruck verwendet wird, benötigt das Verfahren Rechenzeiten bis zu 15 Sekunden. Dies liegt unter anderem an der verwendeten Hardware, einer *Symbolics 3600 Lisp Machine*, und der Programmierspache Lisp. Die Auswertung eines einzelnen Sensoreindrucks kann zudem bei Fehlmessungen der Ultraschall-Sensoren zu einem völligen Versagen des Algorithmus führen.

Der in [Burgard et al. 1996] vorgestellte Ansatz, mittels Positions-Wahrscheinlichkeitsgitter Driftkompensation zu betreiben, wird als einziges Verfahren auch in größeren Büro-Umgebungen[6] getestet. Der Algorithmus erweist sich als robost gegenüber unbekannten Störungen und in Korridor-Situationen. Bei einem zurückgelegten Weg von zehn Metern kann bei diesem Ansatz eine Abweichung von $30cm$ zur realen Roboter-Position gemessen werden.

Abschließend läßt sich feststellen, daß die meisten bisher veröffentlichten Ansätze zur Driftkompensation diese nur in verhältnismäßig kleinen Umgebungen zufriedenstellend lösen. Da Büro-Umgebungen aber in der Regel eine nicht unerhebliche Anzahl von langen Gängen besitzen, sind die Verfahren hierfür nicht geeignet. Der geradenbasierte Ansatz löst nun zu einem großen Teil dieses Problem und läßt sich sowohl in Räumen als auch Fluren erfolgreich einsetzen.

[6] Das Test-Gebiet des 1994 ausgeführten Roboter-Wettbewerbs betrug ca. $20m \cdot 30m$.

Teil III

Globale Lokalisation

Kapitel 14

Motivation

Ist kein Vorwissen über die Position des Roboters vorhanden, ist es erforderlich diese zu lokalisieren, um vorhandene Umweltmodelle nutzen und eine zielorientierte Navigation einleiten zu können. Zudem kann die so ermittelte Position als Initialisierung der Driftkompensations-Verfahren verwendet werden, falls diese versagt haben und den so entstandenen Fehler nicht mehr von selbst beheben können.

Für die globale Lokalisation existieren verschiedene geeignete Sensoren. So können Funkfeuer, die zusätzlich auf verschiedenen Frequenzen senden, eine sogenannte Kreuzpeilung des Roboters erlauben. Hierfür ist allerdings eine Modifikation der Umwelt erforderlich. Ist dies nicht erwünscht bzw. möglich, muß auf andere Sensoren zurückgegriffen werden. Es bieten sich Ultraschall-Sensoren und insbesondere die sehr viel genaueren Laserentfernungsmesser an. Diese zeichnen ein Abbild der aktuellen Umgebung und können so für die Positions-Bestimmung herangezogen werden. Um die globale Position ermitteln zu können, muß der schon zurückgelegte Weg berücksichtigt werden. Dieser schränkt den Raum der Positions-Hypothesen mit zunehmend zurückgelegtem Weg immer weiter ein, bis sich schließlich die absolute Position herauskristallisiert.

Das Problem der absoluten Positionsbestimmung ist durchaus nicht neu. Trotzdem existieren nur wenige Ansätze (z.B. [Fox et al. 1996]), die die absolute Roboter-Position in größeren Umgebungen zufriedenstellend lokalisieren. In den meisten Fällen, wie etwa in [MacKenzie, Dudek 1994] bzw. [Zimmer 1995], wird die absolute Position des Roboters in einer sehr kleinen Umgebung (circa 4 bis 8 Meter) bestimmt. Einen in diesem Zusammenhang zunehmend interessant werdender Bereich stellt die aktive Lokalisation des Roboters, wie sie in [Burgard et al. 1997] beschrieben ist, dar. Diese führt das System zielorientiert in vorbestimmte Positionen, um die Zahl der Positions-Hypothesen zu veringern. Der Nachteil liegt allerdings darin, daß die Hauptziel-Verfolgung erst nach Abschluß dieser Lokalisationsphase eingeleitet werden kann.

Wie oben schon erwähnt, ist die absolute Positions-Bestimmung nicht ausschließlich in der Startphase des Roboters interessant. Ist die Startposition des Roboters bekannt, kann sich im Laufe der Fahrt durch Drift und Schlupf ein so großer Fehler akkumulieren, daß dieser nicht mehr durch die Driftkompensations-Verfahren ausgeglichen werden kann.

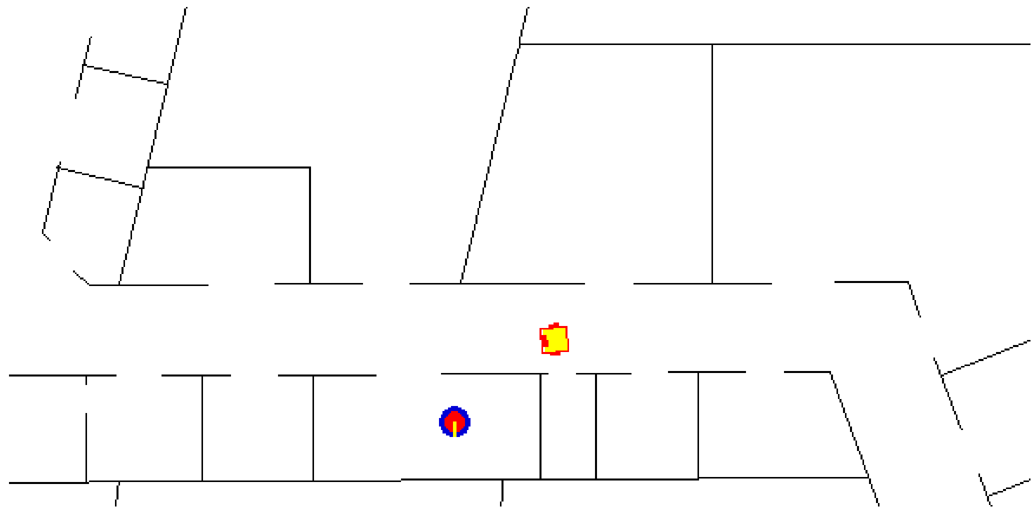

Abbildung 14.1: Notwendigkeit der Relokalisation

Abbildung 14.1 zeigt einen solchen Fall. Der Roboter glaubt an Hand seiner korrigierten odometrischen Daten, in einem Raum zu sein (Kreis). Tatsächlich befindet er sich aber noch im Korridor (Quadrat). Die üblichen Driftkompensations-Verfahren versagen in dieser Situation. Sie vergleichen lediglich die lokale mit der globalen Karte und können nicht zwischen ähnlich aussehenden Kartenausschnitten unterscheiden. Eine permanente Lokalisation des Roboters auf einer übergeordneten Ebene löst dieses Problem.

Kapitel 15

Positions-Wahrscheinlichkeitsgitter

Positions-Wahrscheinlichkeitsgitter, wie sie in [Fox et al. 1996] beschrieben werden, geben in jeder Zelle die Wahrscheinlichkeit der Position des Roboters an. Hierfür wird das Umweltmodell[1] $\mathcal{M}$, das beispielsweise als CAD-Karte vorliegt, mit den Sensorwerten verglichen. Sei nun $p(x|s_1 \cap \ldots \cap s_{n-1} \cap \mathcal{M})$ die Wahrscheinlichkeit, daß die Zelle x die Position des Roboters darstellt. Um die neue Sensorinformation s_n in das Gitter zu integrieren, wird gemäß der Formel von Bayes:

$$p(x|s_1 \cap \ldots \cap s_{n-1} \cap s_n \cap \mathcal{M}) = \frac{p(s_1 \cap \ldots \cap s_{n-1} \cap s_n|x \cap \mathcal{M}) \cdot p(x|\mathcal{M})}{p(s_1 \cap \ldots \cap s_{n-1} \cap s_n|\mathcal{M})} \tag{15.1}$$

vorgegangen. Die Wahrscheinlichkeit $p(s_1 \cap \ldots \cap s_{n-1} \cap s_n|\mathcal{M})$ dient als Normierungsfaktor und ist deshalb von der jeweiligen Zelle x unabhängig. Sie wird durch die Bedingung, daß die Summe der Wahrscheinlichkeiten über alle Zellen des Gitters den Wert eins annehmen muß, bestimmt:

$$\sum_x p(x|s_1 \cap \ldots \cap s_{n-1} \cap s_n \cap \mathcal{M}) = 1. \tag{15.2}$$

Wird die Unabhängigkeit von s_n gegenüber $s_1 \cap \ldots \cap s_{n-1}$ bei gegebenen x und $\mathcal{M}$ vorausgesetzt, so läßt sich (15.1) wie folgt darstellen:

$$p(x|s_1 \cap \ldots \cap s_{n-1} \cap s_n \cap \mathcal{M}) = \frac{p(s_1 \cap \ldots \cap s_{n-1}|x \cap \mathcal{M}) \cdot p(s_n|x \cap \mathcal{M}) \cdot p(x|\mathcal{M})}{p(s_1 \cap \ldots \cap s_{n-1} \cap s_n|\mathcal{M})}. \tag{15.3}$$

Es bleibt die Frage zu klären, wie die Wahrscheinlichkeit $p(s|x \cap \mathcal{M})$ zu berechnen ist. Hierbei wird die von Moravec in [Moravec 1988] vorgeschlagene Diskretisierung genutzt. Ein Sensor kann nun die Entfernungen R_1 bis R_k messen. Die

[1] Dies ist die globale Karte.

Wahrscheinlichkeit, daß die Entfernung R_i an der Position x in der gegebenen Karte $\mathcal{M}$ gemessen wird, läßt sich wie folgt angeben:

$$p(R_i|x \cap \mathcal{M}) = 1 - \prod_{\tilde{x} \in R_i} [1 - \mathcal{M}(\tilde{x}) \cdot p(h(\tilde{x}|x \cap \mathcal{M})] \cdot [1 - (1 - \mathcal{M}(\tilde{x})) \cdot p(e(\tilde{x})|x \cap \mathcal{M})], \tag{15.4}$$

wobei der Schall durch die Gitterzelle $\tilde{x}$ reflektiert wird und $\mathcal{M}(\tilde{x})$ die Belegtheits-Wahrscheinlichkeit von $\tilde{x}$ in $\mathcal{M}$ darstellt.

Die zwei Wahrscheinlichkeiten $p(h(\tilde{x}|x \cap \mathcal{M})$ und $p(e(\tilde{x}|x \cap \mathcal{M})$ geben wieder, ob der Sensor die besetzte bzw. unbesetzte Zelle $\tilde{x}$ detektiert, unter der Voraussetzung, daß x die Roboter-Position ist. Wird nun die Entfernung R_i gemessen, d.h. die Entfernungen R_j mit $j < i$ werden nicht detektiert, ergibt sich die gesuchte Wahrscheinlichkeit

$$p(s|x \cap \mathcal{M}) = p(R_i|x \cap \mathcal{M}) \cdot \prod_{j=1}^{i-1} (1 - p(R_j|x \cap \mathcal{M})). \tag{15.5}$$

Bei der Initialisierung des Positions-Wahrscheinlichkeitsgitters werden die Werte des Besetztheitsgitters zu Grunde gelegt. Je größer die Besetztheits-Wahrscheinlichkeit ist, desto kleiner ist die Positions-Wahrscheinlichkeit:

$$p(x|\mathcal{M}) = \frac{1 - \mathcal{M}(x)}{\sum_{\tilde{x} \in \mathcal{M}} (1 - \mathcal{M}(\tilde{x}))}. \tag{15.6}$$

Teil IV

Diskussion und Ausblick

Kapitel 16

Diskussion

Zum Abschluß dieser Arbeit werden die in den vorherigen Abschnitten erzielten Ergebnisse zusammengefaßt und bewertet. Dabei wird bei den Experimenten[1] auf die Praxistauglichkeit der Verfahren besonderen Wert gelegt, so daß alle Testergebnisse mit dem Roboter *B21* und nicht nur im Simulator, der lediglich einen Ausschnitt der Umwelt modellieren kann, erzielt werden.

Das erste vorgestellte Verfahren beschreibt die erweiterte Hough-Transformation, wie sie in [Haberäcker 1995] zu finden ist. Diese wird auf die zu untersuchenden Umgebungen angepaßt. Die nachfolgende Analyse des resultierenden Akkumulator-Feldes stellt dann eine Positions-Schätzung zur Verfügung. In den zugehörigen Testreihen zeigt sich, daß der Hough-basierte Ansatz in markanten Umgebungen, d.h. Umgebungen, die nicht ausschließlich aus langen Korridoren bestehen, gute Ergebnisse erzielt. Zu einem großen Teil ist das der nachgeschalteten Kalman-Filterung zuzurechnen, da diese nicht nur die Position, sondern auch die zugehörige Varianz für die Positions-Bestimmung berücksichtigt. So haben Akkumulator-Felder, die in Fluren aufgenommen werden, immer eine ausgeprägte Vorzugs-Orientierung. Durch die Verwendung der Max-Value-Strategie spiegelt sich diese auch in der Varianz wieder. Da für jede Komponente des Driftkompensations-Vektors ein eigener Kalman-Filter genutzt wird, können einzelne Komponenten der Positions-Schätzung durchaus mit unterschiedlicher Unsicherheit behaftet sein[2].

Müssen während der Fahrt lange, uncharakteristische Gänge durchlaufen werden, zeigt sich der Hough-Ansatz allerdings von seiner negativen Seite. Sobald nur noch die beiden gegenüberliegenden Wände in der lokalen Karte enthalten sind, ist das Verfahren erwartungsgemäß nicht mehr in der Lage, die Position bzgl. der Korridor-Ausrichtung zu bestimmen. Deshalb wird auf die geradenbasierte Driftkompensation, welche diesen Schwachpunkt ausgleicht, in Kapitel 9 näher eingegangen.

[1] siehe Anhang C
[2] Dies tritt häufig in Gangsituationen auf.

Der zweite Ansatz, die geradenbasierte Driftkompensation, basiert auf den in den beiden Artikeln [Cox, Kruskal 1988] und [MacKenzie, Dudek 1994] angegebenen Verfahren. Diese wurden auf die zur Verfügung stehende Hardware angepaßt und in entscheidenden Punkten verbessert bzw. erweitert. Hierzu ist es notwendig, die als Bitmap vorliegende globale Karte in eine geradenbasierte zu transformieren. Als Basis dient ein Eckendetektor, der auf die besonderen Bedürfnisse der Karten-Rekonstruierbarkeit zugeschnitten ist (siehe Abschnitt 9.3).

Ein großes Problem stellt die Verschiedenartigkeit der verwendeten Sensoren dar. Während Cox und Kruskal einen Laserentfernungsmesser, der bis auf wenige Millimeter Abweichung punktgenau Entfernungen wahrnehmen kann, zur Verfügung haben, ist der *B21* nur mit Ultraschall-Sensoren ausgerüstet. Diese sind gerade bei steigender Entfernung zunehmend ungenau und haben dabei eine kegelförmige Ausbreitung. Es wird deshalb, statt einer einzigen Messung zu vertrauen, die Integration der Sensordaten vorgezogen. Somit gelingt es, auftretende Störungen oder Phantommessungen[3] auszugleichen. Dieser Vorteil für die Lokalisation stellt gleichzeitig einen Nachteil für die Hindernisvermeidung dar, die insbesondere für das Ausweichen beweglicher Objekte zuständig ist.

Da die Verfahren zur Positions-Bestimmung in den als Grundlage dienenden Artikeln nur innerhalb kleiner Räume[4] ihre Gültigkeit bewiesen haben, müssen diese auf beliebig große Umgebungen angepaßt werden. Die Einführung von Fensterausschnitten stellt dabei sicher, daß nur ein kleiner Teil der globalen Karte für die jeweilige Messung herangezogen wird. Der Kartengröße werden somit keine Grenzen gesetzt.

Die Beschränkung der Verfahren auf kleine Räume führt zu einem Fehlverhalten in Gangsituationen. Es ist mit den unmodifizierten Verfahren nicht möglich, die Position nach einer längeren Fahrt durch einen Korridor noch genau zu bestimmen. Deshalb ist es erforderlich, ein weiteres Kriterium in der Zuordnungsebene einzuführen. Durch die Hinzunahme des Liniensegment-Mittelpunktes kann sogar in Fluren eine korrekte Lokalisation sicher gestellt werden. Das Modell für die Translations- und Rotationskorrektur mußte schließlich neu erarbeitet werden, da das in [Cox, Kruskal 1988] angegebene Gleichungssystem fehlerhaft ist.

Desweiteren hat sich gezeigt, daß eine ungefilterte Weiterverarbeitung der Meßergebnisse zu einem fehlerhaften Driftkompensations-Vektor führt. Deshalb wird eine weitere Kontrollebene eingeführt, die die berechnete Position auf Plausibilität überprüft. In den zugehörigen Testreihen können auf diese Weise störende Ausreißer eliminiert werden.

Die Wahl eines geeigneten Sensormodells stellt sich bei genauerer Betrachtung als nicht trivial heraus. Da dieses aber für die Driftkompensation von entscheidender Bedeutung ist, wurden drei verschiedene Ansätze in dieser Arbeit

[3] Zu kurz bzw. zu lang gemessene Entfernungen.

[4] In der Regel zwischen $6m^2$ und $20m^2$.

besprochen. Während das naive Sensormodell über alle Entfernungsbereiche unverändert bleibt, nutzen das modifizierte und das Prioritäten-Modell die besonderen Eigenschaften des Sensors aus. So werden kurze Entfernungen gegenüber längeren grundsätzlich bevorzugt, da Ultraschall-Sensoren im Nahbereich genauer messen können.

Besetztheitsgitter, wie sie in [Moravec 1988] erstmals vorgestellt wurden, bilden anschließend die Grundlage für die Integration der durch das Sensormodell gelieferten Sensoreindrücke. Die Integration über die Zeit stellt sicher, daß Störungen, wie beispielsweise bewegliche Objekte, nicht dauerhaft in der Karte verbleiben. Andererseits wird mit der Einführung von Besetztheits-Graden zusätzliche Information für die Navigation des Roboters bereit gestellt. Mit Hilfe der neuen Sensormodelle und der Sensor-Integration ist es in Experimenten mit dem *B21* gelungen, eine Umgebungskarte aufzubauen, die abgesehen von unwesentlichen Abweichungen der Realität entspricht.

Wie schon weiter oben erwähnt, dienen Kalman-Filter zur Kombination von Meßwerten. Liegt eine ungenaue Messung mit einer großen Varianz vor, so wird auch das Gesamtergebnis nur leicht beeinflußt.

Der Raum-Matching-Algorithmus kann als eine einfache Variante der Driftkompensation angesehen werden. So basiert dieser ausschließlich auf Sensorwerten und benötigt keine lokalen Karten. Das Verfahren führt aber nur in rechtwinkligen Räumen zu akzeptablen Ergebnissen und läßt sich nicht, wie das vorherige Verfahren, universell einsetzen. Wegen der kurzen Zykluszeit und dem Wegfallen der Kartenbildung bleibt dieser Ansatz dennoch interessant. Dieser Algorithmus könnte beispielsweise unterstützend auf weitere Lokalisations-Verfahren wirken.

Die vorgestellten Verfahren eignen sich für kleine Korrekturen der Drift. Da die Zykluszeit der Lokalisation kurz und die Odometrie auf kleinen Strecken genau ist, werden die Verfahren dadurch nicht eingeschränkt. Ein weiterer Vorteil der hier vorgestellten Verfahren liegt in der Fehlertoleranz begründet. Die lokale Karte muß nicht genau mit dem entsprechenden Abschnitt der globalen Karte übereinstimmen. Vielmehr kann das Segment durchaus durch Rauschen gestört oder mit Lücken behaftet sein. Die Maßgabe, daß jeweils eine globale Karte der zu befahrenden Umgebung vorhanden sein muß, stellt in der Realität keine weitere Einschränkung dar, weil in der Regel Grundrisse des entsprechenden Bereiches vorhanden sind.

Die globale Lokalisation wird zum Schluß kurz behandelt. Dabei verspricht der von [Fox et al. 1996] vorgestellte Ansatz, die Aufgabe gut lösen zu können. Dieser Ansatz hat seine Praxistauglichkeit in der Bonner Außenstelle des Deutschen Museums bereits bewiesen.

Zuletzt ist eine weitere wichtige Zielsetzung dieser Arbeit, das Einsatzgebiet des Roboters in keinster Weise zu verändern. Ausdrücklich werden keine künstliche Landmarken erlaubt oder bestimmte Architekturen vorausgesetzt. Dies schließt Verfahren aus, die Vielfache eines bestimmten Winkels zwischen zwei Wänden als gegeben annehmen (z.B. [Thrun 1996]). Die Hough-Transformation sowie der geradenbasierte Ansatz erfüllen diese Bedingungen. Da sie keine weiteren Forderungen an das Zielgebiet stellen, können diese Verfahren als universell einsetztbar angesehen werden.

Kapitel 17

Ausblick

Wie im vorangegangenen Kapitel kurz ausgeführt, erzielen die vorgestellten Verfahren im allgemeinen gute Ergebnisse. Ungünstige Konstellationen[1] können trotzdem noch zu fehlerhaften Korrekturen führen. Weicht die korrigierte Position von der realen zu stark ab, kann dies nicht mehr durch die Driftkompensations-Verfahren ausgeglichen werden. Die in Kapitel 12 vorgestellten Ansätze verhindern dies. Tritt dennoch ein derartiger Fall auf, muß die Position des Roboters in einem relativ kleinen Bereich von beispielsweise sechs Metern absolut bestimmt werden. Anschließend kann mittels der Driftkompensations-Verfahren die Korrektur der Drift und des Schlupfes erneut fortgesetzt werden.

Eine einfache Lokalisation könnte in der direkten Auswertung der Frontsensoren vorgenommen werden. Hierbei wird über eine vorbestimmte Anzahl von Messungen die Entfernung des in Fahrtrichtung liegendem Hindernis vorgenommen. Das so ermittelte Entfernungsmaß wird mit dem aus der korrigierten Position und der globalen Karte berechneten verglichen und der Korrekturvektor entsprechend angepaßt. Das kann natürlich nur innerhalb des Genauigkeitsbereichs der verwendeten Sensoren vorgenommen werden (bei Ultraschall-Sensoren nur wenige Meter). Diese einfache absolute Positionskorrektur verhindert zu starkes Anwachsen des Driftkompensations-Vektors insbesondere in Gangsituationen.

Auch die Verfahren selbst geben noch Spielraum für Verbesserungen. So kann das für die Bestimmung der Positionsgüte genutzte neuronale Netz mit Hilfe der in [Pomerleau 1989] vorgestellten Sensitivitäts-Analyse weiter verfeinert werden. Sie gibt Aufschluß über die Gewichtsverteilung innerhalb des Netzes und erlaubt eine optimale Anpassung der Netzparameter (Anzahl der Neuronen). Außerdem kann an Hand des Akkumulatorfeldes erkannt werden, ob sich der Roboter innerhalb eines Raumes oder eines Ganges befindet (siehe Abschnitt 8.9). Diese Information kann bei einer starken Verschiebung der Position zur Korrektur der selbigen herangezogen werden. Befindet sich der Roboter beispielsweise in einem Raum, nimmt aber an Hand der Daten an, im Gang zu sein, kann dies mit Hilfe

[1] z.B. Umgebungen mit ausschließlich langen Gängen ohne Türen

des Akkumulatorfeldes festgestellt werden.

Auch die geradenbasierte Driftkompensation läßt sich weiter verbessern. So stellt die Zuordnung der Pixel zur nächstliegenden Geraden ein Problem dar. Da jedes Pixel genau einem Liniensegment zugeordnet werden muß, kann es bei Störpixeln zu Zuordnungsfehlern führen. Eine vorhergehende Filterung der lokalen Karte kann diesem Problem entgegenwirken. Desweiteren kann das Verfahren beschleunigt werden, indem nicht alle Punkte der lokalen Karte zur Positionsbestimmung, sondern nur ein sogenanntes Sampling[2] herangezogen wird.

Auch an der Hardware-Plattform selbst, die ansonsten keine Wünsche offen läßt, können weitere Verbesserungen zur Steigerung der Performance vorgenommen werden. So würde ein Kompaß, der die absolute Orientierung des Roboters zuverlässig ausgibt, die Nutzung des einfachen Translations-Ansatzes des geradenbasierten Verfahrens zulassen. Dies hätte eine kürzere Zykluszeit und damit eine genauere Positionsbestimmung zur Folge. Die derzeitige Rechenleistung der beiden $150MHz$-Pentium-Rechner könnte durch leistungsfähigere Computer weiter ausgebaut werden. Dies würde den Flaschenhals 'Funk-Ethernet' beseitigen und die Interprozeß-Kommunikation in wesentlichen Punkten vereinfachen[3].

[2] Wird nur jeder k-te Punkt herangezogen, so wird von einem k-Sampling gesprochen.

[3] Es könnte zum größten Teil via Shared-Memory kommuniziert werden.

Teil V

Anhang

Anhang A

Bestimmung der Sensorkegel

In diesem Abschnitt wird auf die Erzeugung des Sensoreindrucks, der schließlich der lokalen Karte als Basis dient, näher eingegangen. Hierzu müssen aus der Orientierung θ des Sensors und der gemessenen Entfernung ϱ die Sensorkegel berechnet werden. Als erster Schritt bietet sich die Bildung des umschließenden Rechtecks an:

$$\begin{aligned}
x_1 &= \min\{s_x, sm_{x_1}, sm_{x_2}, s_x + b_x\}, && \text{(A.1)}\\
y_1 &= \min\{s_y, sm_{y_1}, sm_{y_2}, s_y + b_y\}, && \text{(A.2)}\\
x_2 &= \max\{s_x, sm_{x_1}, sm_{x_2}, s_x + b_x\} \quad \text{und} && \text{(A.3)}\\
y_2 &= \max\{s_y, sm_{y_1}, sm_{y_2}, s_y + b_y\}. && \text{(A.4)}
\end{aligned}$$

Die Position des Sensors ist dabei mit (s_x, s_y) festgelegt. Die Punkte (sm_{x_1}, sm_{y_1}), (sm_{x_2}, sm_{y_2}) und (b_x, b_y) seien wie folgt definiert:

$$\begin{aligned}
sm_{x_1} &= s_x + \cos\left(\theta - \frac{1}{2} \cdot \xi\right) \cdot \chi, && \text{(A.5)}\\
sm_{y_1} &= s_x + \sin\left(\theta - \frac{1}{2} \cdot \xi\right) \cdot \chi, && \text{(A.6)}\\
sm_{x_2} &= s_x + \cos\left(\theta + \frac{1}{2} \cdot \xi\right) \cdot \chi, && \text{(A.7)}\\
sm_{y_2} &= s_x + \sin\left(\theta + \frac{1}{2} \cdot \xi\right) \cdot \chi, && \text{(A.8)}\\
b_x &= \cos(\xi) \cdot \chi \quad \text{und} && \text{(A.9)}\\
b_y &= \sin(\xi) \cdot \chi, && \text{(A.10)}
\end{aligned}$$

wobei ξ den Winkel der Kegelöffnung und χ die Maximalentfernung beschreiben. Nachdem nun die linke untere und die rechte obere Ecke des Rechteckes definiert sind, ist es vollständig beschrieben. Abbildung A.1 zeigt ein solches.

Nun muß für alle Punkte innerhalb des Rechteckes die Zugehörigkeit bestimmt werden. Hierfür erweist sich das Skalarprodukt als geeignet. Für zwei Vektoren

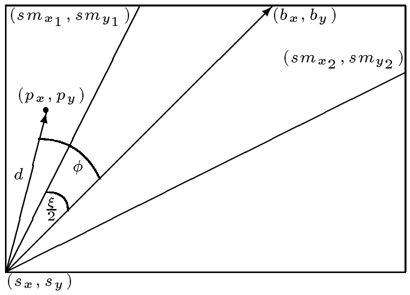

Abbildung A.1: Umschließendes Rechteck für den Sensoreindruck

v und w bezeichnen $\|v\|$ und $\|w\|$ die Länge der Vektoren und α den Winkel zwischen ihnen. Somit ergibt sich für das Skalarprodukt

$$< v, w > = \|v\| \cdot \|w\| \cdot \cos(\alpha). \tag{A.11}$$

Werden die Längen der Vektoren auf eins normiert, liefert das Skalarprodukt direkt den Cosinus des eingeschlossenen Winkels:

$$< v', w' > = \cos(\alpha). \tag{A.12}$$

Sei nun (p_x, p_y) ein beliebiger Punkt innerhalb des umschließenden Rechtecks mit Abstand d von (s_x, s_y). Der Cosinus des Winkels ϕ, zwischen den beiden Vektoren (p_x, p_y) und (b_x, b_y), ergibt sich aus dem auf eins normierten Skalarprodukt, d.h.

$$\cos(\phi) = \frac{< (p_x, p_y), (b_x, b_y) >}{\chi \cdot d}. \tag{A.13}$$

Der Punkt (p_x, p_y) ist genau dann innerhalb des Sensorkegels, wenn der Cosinus von ϕ größer als der Cosinus von $\frac{\xi}{2}$ ist:

$$\cos(\phi) > \cos\left(\frac{\xi}{2}\right), \tag{A.14}$$

d.h. $\phi < \frac{\xi}{2}$ gilt. Um eine weitere Verfeinerung des Sensoreindruckes zu realisieren, ist es sinnvoll, einen zweiten, schmaleren Kegel mit dem Öffnungswinkel ζ einzuführen. Punkte, die innerhalb dieses Kegels liegen, erhalten die Wertigkeit *EMPTY*. Punkte, die außerhalb des ersten, aber noch innerhalb des zweiten Kegels[1] liegen, erhalten den Wert *MEDUNKNOWN*, und schließlich werden die außerhalb liegenden Punkte mit *UNKNOWN* belegt.

Nachdem alle Punkte innerhalb der Keule auf die obige Art bestimmt worden sind, muß nun das Hindernis[2] in den Sensoreindruck eingetragen werden. Dieses

[1] Dies ist der ursprüngliche Kegel mit dem Öffnungswinkel ξ.
[2] Insbesondere stellen auch Wände Hindernisse dar.

liegt, wie oben schon erwähnt, in einer Entfernung von ϱ vor. Somit folgt für alle Punkte (p_x, p_y), deren Abstand zu (s_x, s_y) größer oder gleich ϱ ist und die innerhalb des Kegels liegen, daß diese das Hinderniss darstellen. Da es im allgemeinen aber keinen Sinn macht, Wände beliebig dick in der lokalen Karte einzutragen, wird deren Stärke vorher festgelegt. Um für die lokale Karte *lmap* die Zugehörigkeit eines beliebigen Punktes (p_x, p_y) bestimmen zu können, seien zuvor die boolschen Variablen $bool_1$ bis $bool_6$ durch das folgende Regelsystem definiert. Dabei sei $\mathcal{W}$ die Wandstärke und τ der Anteil der Wand, der mit *MEDOBSTACLE* belegt werden soll.

- $\underline{\text{if}}\ \cos(\phi) > \cos\left(\frac{\zeta}{2}\right)\ \underline{\text{then}}\ bool_1 = True\ \underline{\text{else}}\ bool_1 = False$
- $\underline{\text{if}}\ \cos(\phi) > \cos\left(\frac{\xi}{2}\right)\ \underline{\text{then}}\ bool_2 = True\ \underline{\text{else}}\ bool_2 = False$
- $\underline{\text{if}}\ \|(p_x, p_y) - (s_x, s_y)\| < \varrho\ \underline{\text{then}}\ bool_3 = True\ \underline{\text{else}}\ bool_3 = False$
- $\underline{\text{if}}\ \|(p_x, p_y) - (s_x, s_y)\| < \varrho + \frac{\mathcal{W}}{\tau}\ \underline{\text{then}}\ bool_4 = True\ \underline{\text{else}}\ bool_4 = False$
- $\underline{\text{if}}\ \|(p_x, p_y) - (s_x, s_y)\| < \varrho + \mathcal{W} - \frac{\mathcal{W}}{\tau}\ \underline{\text{then}}\ bool_5 = True\ \underline{\text{else}}\ bool_5 = False$
- $\underline{\text{if}}\ \|(p_x, p_y) - (s_x, s_y)\| < \varrho + \mathcal{W}\ \underline{\text{then}}\ bool_6 = True\ \underline{\text{else}}\ bool_6 = False$

Aus obigen Regeln läßt sich nun die Zugehörigkeit der einzelnen Bildpunkte bestimmen. Um übersichtlich zu bleiben, wird der *MEDOBSTACLE*-Bereich in zwei Gleichungen unterteilt.

$$bool_1 \wedge bool_2 \wedge bool_3 \implies lmap(p_x, p_y) = EMPTY \tag{A.15}$$
$$\neg bool_1 \wedge bool_2 \wedge bool_3 \implies lmap(p_x, p_y) = MEDUNKNOWN \tag{A.16}$$
$$\neg bool_2 \vee \neg bool_6 \implies lmap(p_x, p_y) = UNKNOWN \tag{A.17}$$
$$bool_2 \wedge \neg bool_3 \wedge bool_4 \implies lmap(p_x, p_y) = MEDOBSTACLE \tag{A.18}$$
$$bool_2 \wedge \neg bool_5 \wedge bool_6 \implies lmap(p_x, p_y) = MEDOBSTACLE \tag{A.19}$$
$$bool_2 \wedge \neg bool_4 \wedge bool_5 \implies lmap(p_x, p_y) = OBSTACLE \tag{A.20}$$

Anhang B

Die verwendete Hardware

Um die in den vorhergehenden Kapiteln besprochenen Verfahren auf einem System unter Echtzeitbedingungen testen zu können, müssen diese auf einem realen Roboter implementiert werden. Die Experimente fanden auf einem Roboter des Typs *B21* der Firma *Real World Interface* statt (Abbildung B.1), der hier kurz charakterisiert wird. Der *B21* läßt sich in die folgenden drei Hauptkomponenten untergliedern:

- Base (ca. $30cm$ vom Boden bis zur oberen Kante),
- Enclosure (ca. $90cm$ ab der oberen Kante der *Base*) und
- Console (fester Aufsatz auf der *Enclosure*).

Die *Base* beinhaltet neben den Antriebsrädern, den mechanischen Komponenten und den Batterien auch die zugehörigen Motoren, 32 taktile und 32 Infrarot-Sensoren. Einfache Funktionen, wie die Feststellung der odometrischen Position, werden zusätzlich von ihr übernommen.

Die *Enclosure* kann bis zu zwei vollständige Computer beherbergen. Außerdem finden 24 Ultraschall-, 24 IR- und 24 taktile Sensoren in ihr Platz. Die Funkverbindung zur Außenwelt sowie die komplette Ansteuerungs-Logik sind zudem in der *Enclosure* eingebaut.

Die *Console* stellt schließlich die internen Zustände des Roboters dar. Zusätzlich kann ein Notebook, welches auf der *Enclosure* steht, an ihr angeschlossen werden, um eine größere Unabhängigkeit des Roboters zu gewährleisten.

Der *B21* wird standardmäßig mit vier Antriebsrädern, die synchron gelenkt werden, ausgeliefert. Dabei wird bei einer Drehung die absolute Orientierung der *Base* nicht verändert, während die *Enclosure* entsprechend der Ausrichtung der Antriebsräder rotiert wird. Die zylindrische Form des Roboters garantiert ein Drehen auf der Stelle, ohne mit Hindernissen jeglicher Art zu kollidieren. Der Roboter verfügt außerdem über ein globales Koordinaten-System, in welchem er sich bewegen kann.

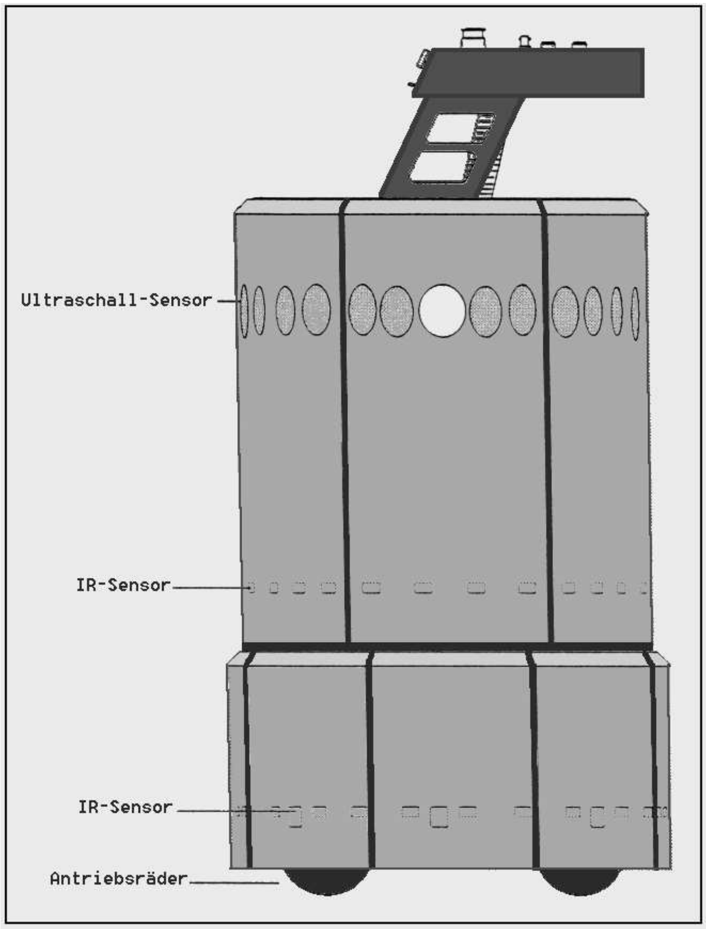

Abbildung B.1: Der Roboter des Typs *B21*

Die Ansteuerung des Roboters erfolgt über das *Hardware-Interface*. Dieses sendet *curve*-Kommandos zur *Base* und erhält im Gegenzug Ultraschall-Sensor-Reports. Ein Kommando hat dabei die folgende Gestalt.

$$curve(t, ts, \theta, \theta s) \tag{B.1}$$

Wobei t der zurückzulegende Weg, ts die Translations-Geschwindigkeit, θ der zudrehende Winkel und θs die Rotations-Geschwindigkeit darstellt. Mit Hilfe der mehrfachen Hintereinander-Ausführung dieses Befehls können alle Fahroperationen modelliert werden[1].

[1] Insbesondere sind Kreisbahnen einfach realisierbar.

Anhang C

Testläufe

Im folgenden werden weitere Testläufe in Form von Abbildungen kurz dargestellt. Wegen der besseren Eigenschaften des geradenbasierten Ansatzes gegenüber der Hough-Transformation, beschränken sich diese auf die geradenbasierte Driftkompensation.

Die obere Abbildung zeigt dabei jeweils den zurückgelegten Weg in der globalen Karte. Dieser entspricht in der Regel auch der tatsächlich gefahrenen Strecke. Der Driftkompensations-Vektor, der für alle aufsetzenden Verfahren von großer Bedeutung ist, ist in den folgenden drei Grafiken zu sehen. Die linke und mittlere Abbildung stellen die x- bzw. die y-Komponente des Vektors dar, während die rechte den Graph in Abhängigkeit beider Komponenten zeigt. Auch hier wird auf die Darstellung der Rotations-Verschiebung verzichtet, da diese bei allen Testläufen nahezu unverändert auf dem Wert null blieb. Schließlich zeigen die letzten drei Abbildungen die odometrische und die korregierte Position des Roboters in der Komponenten-Darstellung bzw. als Graph des zurückgelegten Weges.

Als Geraden-Zuordnungsraum wird eine Umgebung von drei Metern um die vermutete Roboter-Position gewählt. Alle Liniensegmente, die Anfangs- oder Endpunkt innerhalb dieses Fensters haben bzw. es schneiden, können Punkten der lokalen Karte zugeordnet werden, um anschließend das daraus resultierende Minimierungsproblem zu lösen.

Zur Kartenbildung wird ausschließlich auf die Ultraschall-Sensoren zurückgegriffen. Der Sensoreindruck liegt dabei als Besetztheitsgitter vor, welches mittels des Prioritäten-Modells erstellt wurde. Schließlich gliedert der *Integrator* die Sensordaten mit Hilfe des von Moravec in [Moravec 1988] vorgestellten Verfahrens über die Zeit ein.

Die Hardware-Plattform stand in Form eines Roboters des Typs *B21* der Firma *Real World Interface* zur Verfügung. Eine kurze Beschreibung der Hardware-Eigenschaften dieses Roboters ist in Anhang B zu finden.

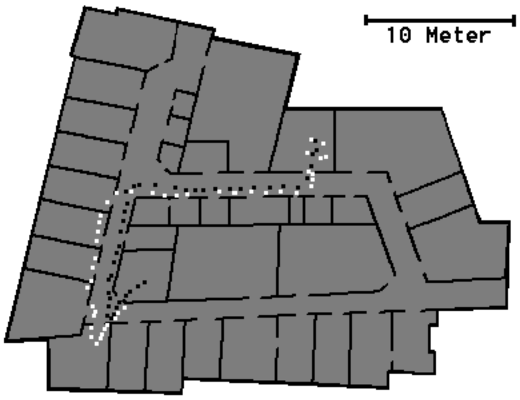

Abbildung C.1: Testlauf 2

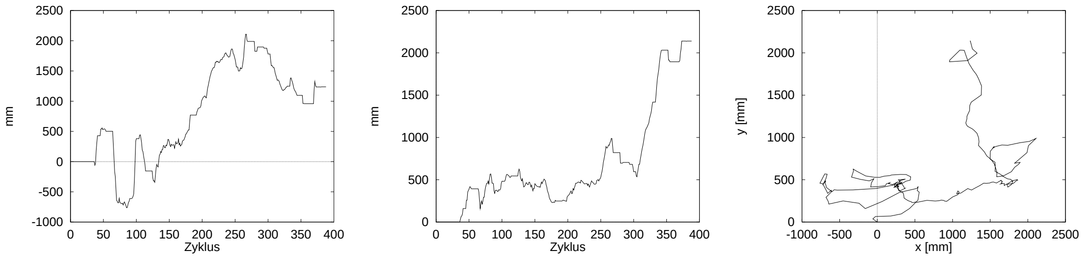

Abbildung C.2: Verschiebungsgraphen des 2. Testlaufes

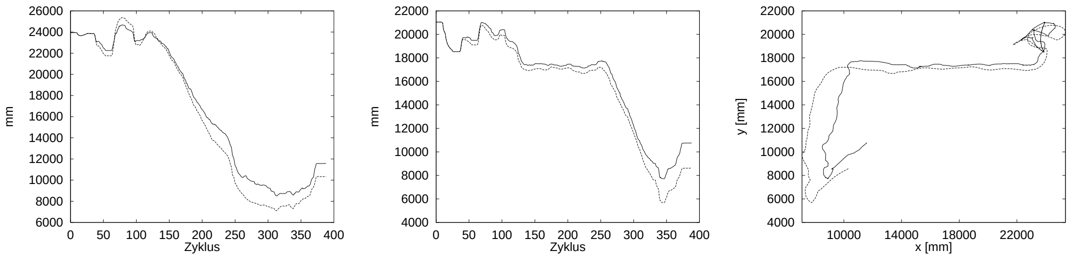

Abbildung C.3: Positionsgraphen des 2. Testlaufes

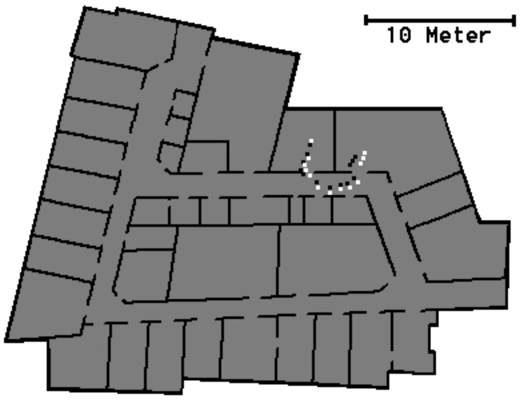

Abbildung C.4: Testlauf 3

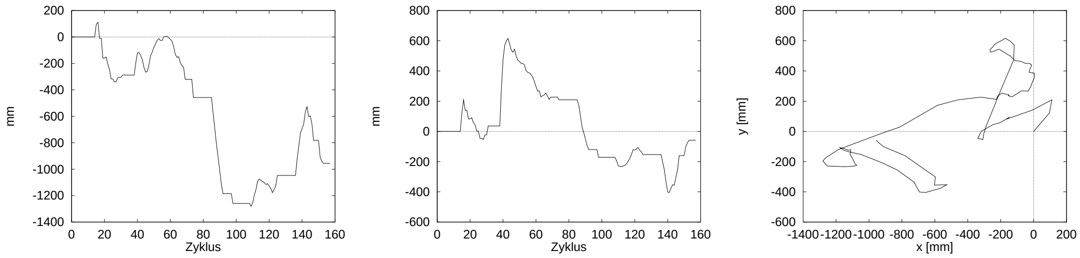

Abbildung C.5: Verschiebungsgraphen des 3. Testlaufes

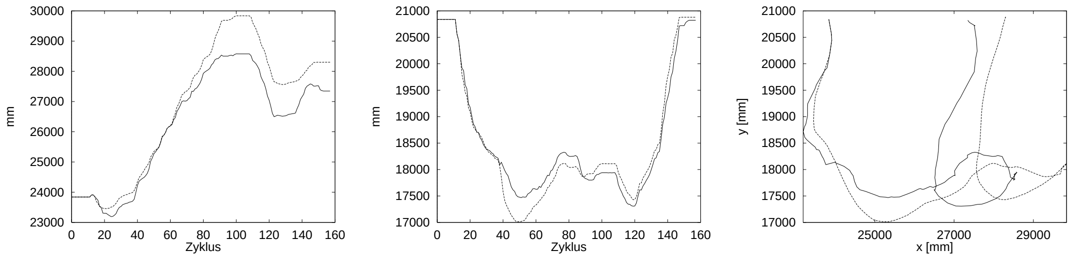

Abbildung C.6: Positionsgraphen des 3. Testlaufes

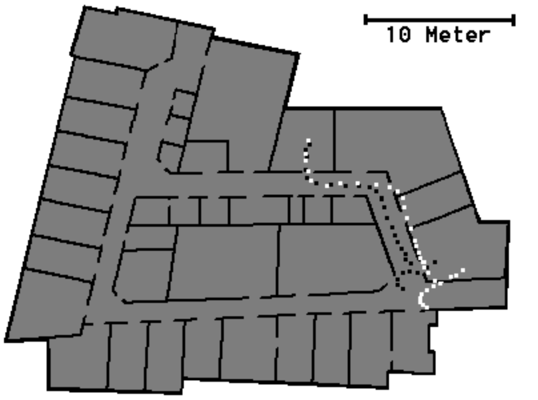

Abbildung C.7: Testlauf 4

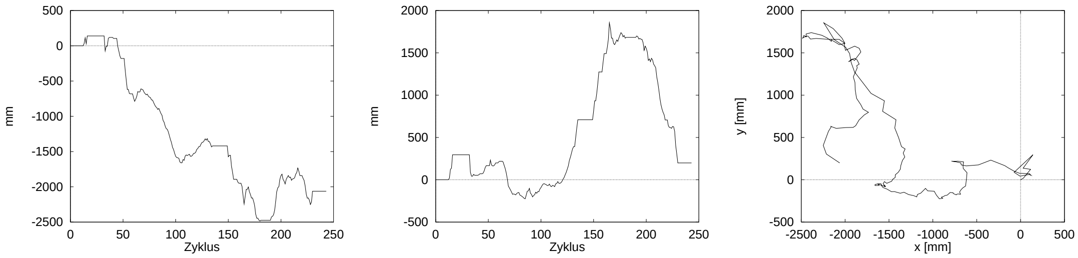

Abbildung C.8: Verschiebungsgraphen des 4. Testlaufes

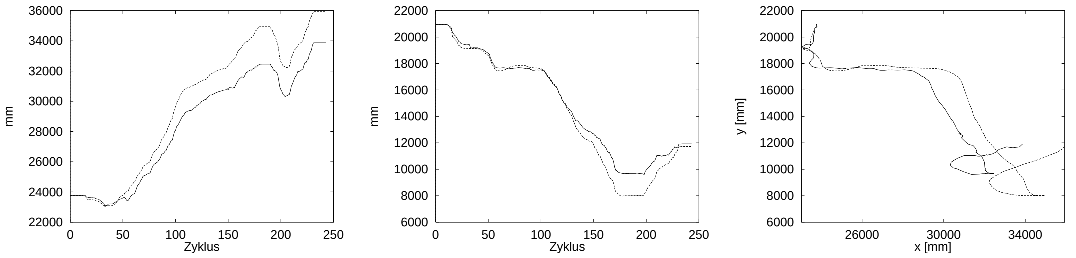

Abbildung C.9: Positionsgraphen des 4. Testlaufes

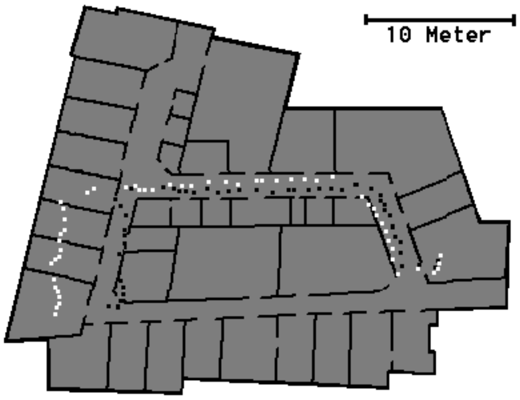

Abbildung C.10: Testlauf 5

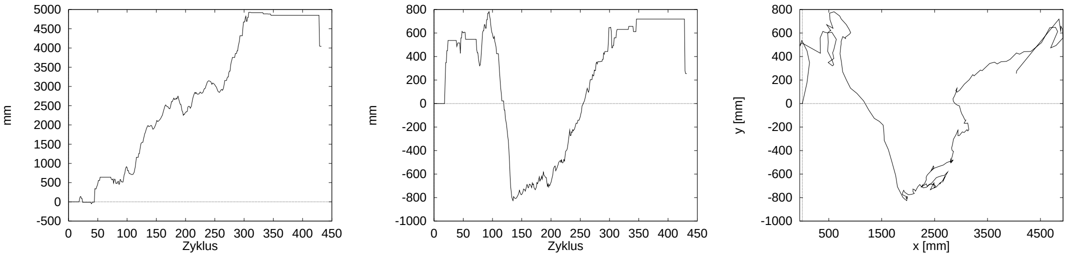

Abbildung C.11: Verschiebungsgraphen des 5. Testlaufes

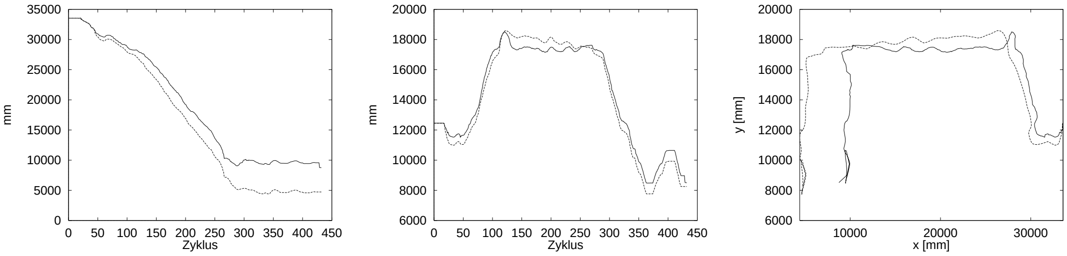

Abbildung C.12: Positionsgraphen des 5. Testlaufes

Abbildung C.13: Testlauf 6

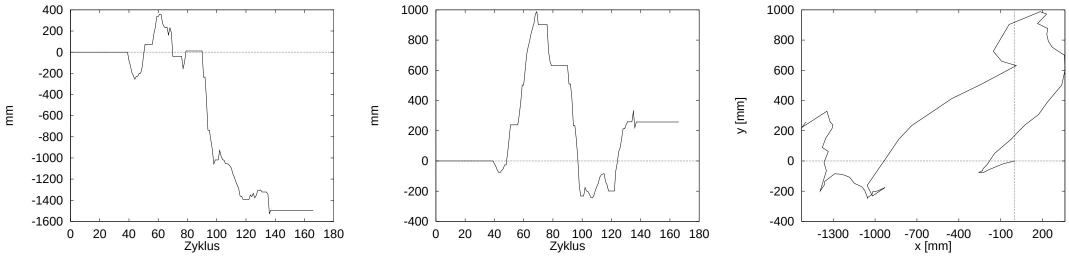

Abbildung C.14: Verschiebungsgraphen des 6. Testlaufes

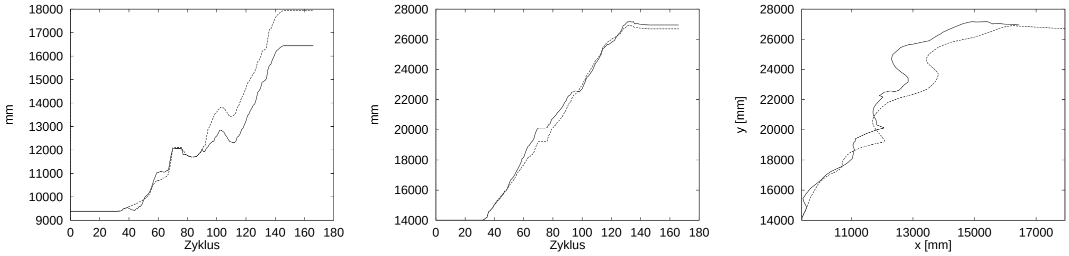

Abbildung C.15: Positionsgraphen des 6. Testlaufes

Anhang D

Testlauf-Bilder

Abbildung D.1: Oben: Korridor mit einer Tür. Unten: Einfahrt in den Zielraum.

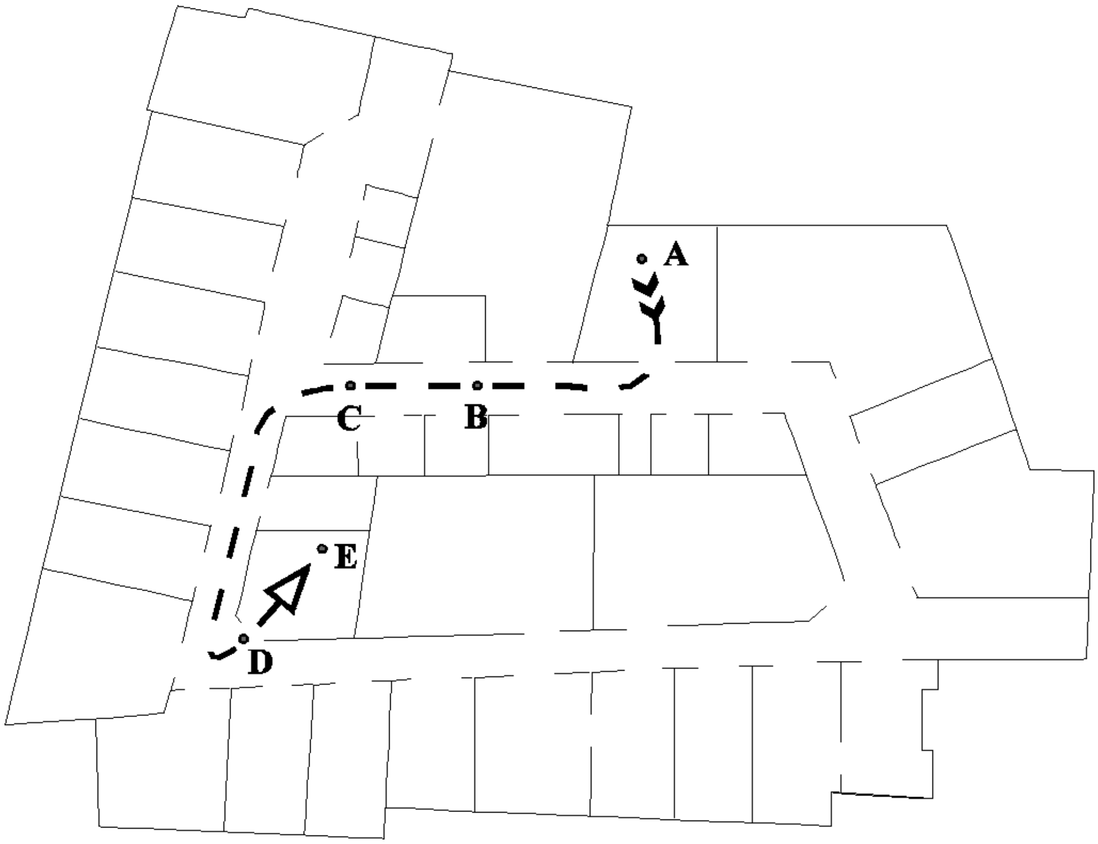

Abbildung D.2: Der Testlauf-Weg

Abbildung D.2 zeigt den zurückgelegten Weg des Testlaufes. Ausgehend vom derzeitigen Roboter-Labor *A* ist der RWI B21 in den Druckerraum *E* gefahren. Die Photos der Abbildung D.1 zeigen die Positionen *B*, *C* und *D*, wobei die beiden oberen Bilder den Punkten *B* und *C* entsprechen und die unteren dem Punkt *D*.

Anhang E

Die CD-ROM

Die beiliegende CD-ROM ist im ISO 9660 - Format und enthält die folgenden Daten.

- *DRIFT.PS* : Die vollständige Diplom-Arbeit im Postscript-Format
- *DRIFT.PDF* : Die vollständige Diplom-Arbeit im Acrobat-Format
- *DRIFT.TAR* : Die LateX-Sourcen und die Bilder der Diplom-Arbeit
- *FOLIEN.PS* : Die Vortragsfolien im Postscript-Format
- *FOLIEN.PDF* : Die Vortragsfolien im Acrobat-Format
- *FOLIEN.TAR* : Die LateX-Sourcen und die Bilder der Vortrags-Folien
- *SOURCE.TAR* : Der komplette Quell-Code als TAR-Archiv
- *TEST_B21.MPG* : Ein Film eines Testlaufes (Roboter)
- *TEST_SUN.MPG* : Ein Film eines Testlaufes (Computer)

Sowohl die Texte als auch die Filme liegen dabei in einem unkomprimierten Format vor, so daß diese ohne weitere Vorbereitung direkt von der CD-ROM betrachtet werden können. Die TAR-Archive hingegen müssen mittels der folgenden Befehlszeile wieder neu angelegt werden.

$$tar\ \ xf\ \ \ < NAME > .TAR \tag{E.1}$$

Während der Parameter f die Datei auswählt, stellt x das Archiv wieder her.

Bemerkung E.1 *Unter dem Betriebssystem* `Windows95` *lassen sich die Dateien mit Hilfe des Programmes* `WinZip` *rekonstruieren.*

Literaturverzeichnis

[Andersen et al. 1985] R.A. Andersen, G.K. Essik, R.M. Siegel: *Encoding of spatial location by posterior parietal neurons*, Science, Volume 230, S.456-458, 1985

[Brown 1985] M.K. Brown: *Locating Object Surfaces with an Ultrasonic Range Sensor*, IEEE Conference on Robotics and Automation, St. Louis, S.110-115, März 1985

[Burgard et al. 1996] W. Burgard, D. Fox, D. Henning, T. Schmidt: *Position Tracking with Position Probability Grids*, University of Bonn, 1996

[Burgard et al. 1997] W. Burgard, D. Fox, S. Thrun: *Active Mobile Robot Localization*, 15th International Joint Conference on Artificial Intelligence, 1997

[Cox, Wilfong 1990] I.J. Cox, G.T. Wilfong, Editoren: *Autonmous Robot Vehicles*, Springer Verlag, 1990

[Cox, Kruskal 1988] I.J. Cox, J.B. Kruskal: *On the Congruence of Noisy Images to Line Segment Models*, Proceedings of the 1988 IEEE, International Conference on Computer Vision, S.252-258, 1988

[Curran, Kyriakopoulos 1993] A. Curran, K. J. Kyriakopoulos: *Sensor-Based Self-Localization for Wheeled Mobile Robots*, Proccedings of the 1993 IEEE, International Conference on Robotics and Automation, S.8-13,1993

[van Dam et al. 1990] J.W.M. van Dam, B.J.A. Kröse, F.C.A. Groen: *Transforming Occupancy Grids under Robot Motion*, Faculty of Mathematics and Computer Science, University of Amsterdam, 1990

[van Dam et al. 1996] J.W.M. van Dam, B.J.A. Kröse, F.C.A. Groen: *Adaptive Sensor Models*, Faculty of Mathematics and Computer Science, University of Amsterdam, 1996

[Drumheller 1987] Michael Drumheller: *Mobile Robot Localization Using Sonar*, IEEE Transactions on Pattern Analysis and Machine Intelligence, S.325-332, 1987

[Duda, Hart 1972] R.O. Duda, P.E. Hart: *Use of the Hough Transform to Detect Lines and Curves in Pictures*, Communication of the ACM, Vol. 15, No. 1, S.11-15, 1972

[Edlinger, Weiß 1995] T. Edlinger, G. Weiß: *Exploration, Navigation and Self-Localization in an Autonomous Mobile Robot*, Autonome mobile Systeme '95 University of Kaiserslautern, 1995

[Elfes 1989] A. Elfes: *Using Occupancy Grids for Mobile Robot Perception and Navigation*, IEEE Computer, S.46-57, Juni 1989

[Fox et al. 1996] W. Burgard, D. Fox, D. Henning, T. Schmidt: *Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids*, University of Bonn, 1996

[Haase 1997] T. Haase: *Entwurf einer GUI-Klassenbibliothek für das X-Window-System*, Christian-Albrechts-Universität, Technische Fakultät, Kiel, 1997

[Haberäcker 1995] P. Haberäcker: *Praxis der Digitalen Bildverarbeitung und Mustererkennung*, Carl Hanser Verlag, Münschen, 1995

[Hähne 1995] H. Hähne: *Verfahren zur Detektion von Ecken*, Christian-Albrechts-Universität, Technische Fakultät, Kiel, 1995

[Haralick, Shapiro 1992] R.M. Haralick, L.G. Shapiro: *Computer and Robot Vision*, Addisson Wesley, 1992

[Haykin 1994] S. Haykin: *Neural Networks*, Macmillan College Publishing Company Inc., New York, 1994

[Hodgkin, Huxley 1952] A. Hodgkin, A. Huxley: *A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve*, J. Physiol., London, S.500-544, 1952

[Holenstein et al. 1992] A.A. Holenstein, M.A. Müller, E. Badreddin: *Mobile Robot Localization in a Structured Environment Cluttered with Obstacles*, Proccedings of the 1992 IEEE, International Conference on Robotics and Automation, S.2576-2581,1992

[Hough 1962] P.V.C. Hough: *Methods and Means for Recognizing Complex Patterns*, U.S. Patent 3069654, 1962

[Köhle 1990] M. Köhle: *Neuronale Netze*, Springer-Verlag, Wien, 1990

[MacKenzie, Dudek 1994] P. MacKenzie, G. Dudek: *Precise Positioning Using Model-Based Maps*, Proccedings of the 1994 IEEE, International Conference on Robotics and Automation, S.1615-1621, 1994

[Maybeck] P.S. Maybeck: *The Kalman Filter: An Introduction to Concepts*, Aus [Cox, Wilfong 1990], S.194-204

[McCulloch, Pitts 1943] W. McCulloch, W. Pitts: *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics 5, S.115-133, 1943

[Merlin, Farber 1975] P.M. Merlin, D.J. Farber: *A Parallel Mechanism for Detecting Curves in Pictures*, IEEE Transactions on Computers, Vol. 24, S.96-98, 1975

[Moravec 1988] H.P. Moravec: *Sensor fusion in certainty grids for mobile robots*, AI Magazine, S.61-74, Sommer 1988

[Moravec, Blackwell 1992] H.P. Moravec, M. Blackwell: *Learning Sensor Models for Evidence Grids*, Robotics Institute Research Review, Pittsburgh, 1992

[Polaroid 1993] Polaroid Corporation: *Polaroid Ultrasonic Ranging Developer's Kit*, Publication No. PXW6431 6/93, Cambridge, MA, June 1993

[Moravec 1988] H.P. Moravec: *Sensor fusion in certainty grids for mobile robots*, AI Magazine, S.61-74, Sommer 1988

[Moravec 1988] H.P. Moravec: *Sensor fusion in certainty grids for mobile robots*, AI Magazine, S.61-74, Sommer 1988

[Pomerleau 1989] D.A. Pomerleau: *ALVINN: An Autonomous Land Vehicle In a Neural Network*, Advances in Neural Information Processing Systems 1, D.S. Touretzky, Editor, Morgan Kaufmann, 1989

[Rattarangsi, Chin 1990] A. Rattarangsi, R.T. Chin: *Scale-based Detection of Corners of Planar Curves*, 10th International Conferenz on Pattern Recognition, S.923-930, 1990

[Schiele, Crowley 1994] B. Schiele, J.L. Crowley: *A Comparison of Position Estimation Techniques Using Occupancy Grids*, Proccedings of the 1994 IEEE, International Conference on Robotics and Automation, 1994, S.1628-1634

[Steven 1992] C.F. Stevens: *Die Nervenzelle*, Spektrum der Wissenschaft: Gehirn und Nervensystem, Spektrum der Wissenschaft Verlag, Heidelberg, 10. Auflage, S.2-13, 1992

[Thrun 1996] S. Thrun, A. Bücken: *Learning Maps for indoor Mobile Robot Navigation*, Technical Report, Mobile Robot Laboratory, CMU, 1997

[Tou, Gonzalez 1974] J.T. Tou, R.C. Gonzalez: *Pattern Recognition Prinziples*, Addision-Wesley, Advanced Book Program, Reading, S.86-107, 1974

[Wang 1988] C.M. Wang: *Location Estimation and Uncertainty Analysis for Mobile Robots*, IEEE Conferences on Robotics and Automation, S.529-533, 1988

[Zimmer 1995] Uwe R. Zimmer: *Self-Localization in Dynamic Enviroments*, IEEE/SOFT International Workshop BIES'95, 1995

Index

Eidestattliche Erklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Kiel, den 25. März 1998