Diplomarbeit

Ein System zur visuellen landmarkenbasierten Roboternavigation

durchgeführt am

Lehrstuhl für Kognitive Systeme
Institut für Informatik und Praktische Mathematik
Christian–Albrechts–Universität zu Kiel
Prof. Dr. G. Sommer

vorgelegt von
Daniel Grest

Betreuer: Bodo Rosenhahn

Dezember 2001

Zusammenfassung

Ziel dieser Arbeit ist ein System zur visuellen landmarkenbasierten Roboternavigation. Ein autonomer, mobiler Roboter soll sich anhand von dreidimensionalen CAD-Modellen der Umgebung orientieren. Die globale Position und Lage dieser Landmarken ist a-priori bekannt. Um die Odometriefehler des Roboters zu korrigieren werden verschiedene Unterprobleme behandelt. Zunächst werden Geraden (mittels *Houghtransformation*) aus Stereobildpaaren extrahiert. Für die Zuordnung der Bildgeraden zum CAD-Modell (*Korrespondenzproblem*) wird eine Vorhersage der Modellprojektion benutzt, die aus den Odometriedaten gewonnen wird. Die Zuordnung liefert dann eine Schätzung der relativen Bewegung des Modells (*2D-3D Poseschätzung*) und ermöglicht die Korrektur des Odometriefehlers (*Selbstlokalisation*).

Das Systemdesign ist von besonderer Relevanz. Es wird ein paralleler Ansatz gewählt mit unterschiedlichen Kompetenzen in einem hierarchisch organisierten Schichtenmodell. Angelehnt an verhaltensbasierte Architekturen ermöglicht dieses Design die zeitlich asynchrone Funktionsweise einzelner Schichten und Robustheit gegenüber Ausfall oder Funktionsstörung höherer Schichten.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die vorliegende Arbeit beschäftigt sich mit der visuellen Navigation für autonome Roboter. Es wird ein System entwickelt, das anhand einer bekannten Landmarke seine Position verifizieren und korrigieren kann. Zur Navigation werden ausschließlich Bilddaten verwendet. Als Landmarke eignen sich normale Möbel aus Büroumgebungen, wie Tische oder Schränke, genauso wie *CAD*-Modelle von Durchgängen oder Flurkreuzungen.
Der Begriff Navigation wird in [23] folgendermaßen definiert:
Ein bewegliches Objekt soll ausgehend von einer momentanen Position auf der Basis von unvollständiger Information unter Berücksichtigung vorgegebener Randbedingungen zu einem vorgegebenen Ziel gebracht werden.

1.1 Motivation

Selbstlokalisation ist die Grundlage für erfolgreiche Aktionen eines mobilen Systems. Ohne Wissen über seine Position und Orientierung in der Welt ist der Roboter nicht in der Lage, sinnvoll mit der Welt zu interagieren. Diese Arbeit beschäftigt sich deshalb mit dem Problem der Positionsverfolgung (engl. *position tracking*), bei dem die initiale Pose (Position und Orientierung) bekannt ist und jederzeit bis auf einen Fehler vorhergesagt werden kann. Die Aufgabe ist diesen Fehler mittels visueller Information zu korrigieren. Wie in vielen anderen Arbeiten zu diesem Thema [12, 27, 43] wird angenommen, dass der Positionsfehler begrenzt ist.

Gegenüber der lokalen Positionsverfolgung stehen Verfahren zur globalen Positionsbestimmung (engl. *global localization*), bei denen die initiale Pose des Roboters nicht bekannt ist [11, 53]. Es wird aber eine globale Karte der Umgebung benötigt. Verfahren zur globalen Positionsbestimmung nutzen in der Regel Sonarsensoren oder Laserentfernungsmesser, weil diese Sensoren Informationen über einen größeren Winkelbereich liefern. Dies ist mit Kameras nicht ohne Bewegung des Fahrzeugs oder der Kameras möglich.

Für viele Anwendungen sind Ultraschallsensoren oder Laserentfernungsmesser nicht optimal geeignet, da sie die Welt nur unzureichend abbilden. Der Einsatz von Kameras dagegen eröffnet einen deutlich größeren Anwendungsbereich, wie z.B. Objekterkennung, Objektverfolgung, Greifen von Objekten, Hindernisvermeidung, Selbstlokalisation etc. Deshalb konzentrieren wir uns in dieser Arbeit auf die Verarbeitung von Bildinformationen und bieten damit die Grundlage für weitere visuelle Anwendungen.

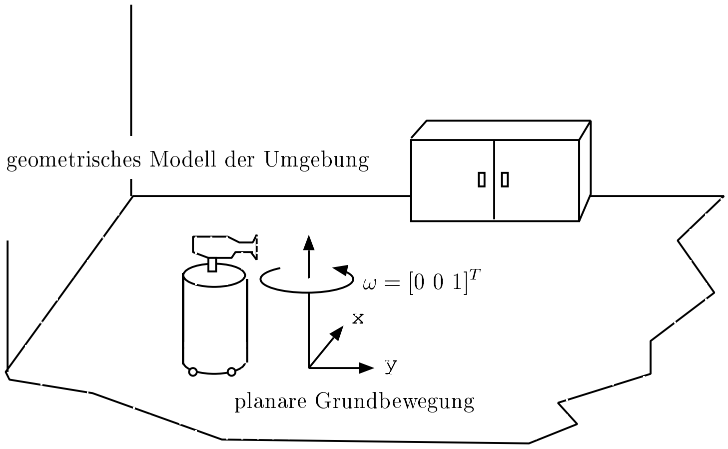

Abbildung 1.1: Der Roboter bewegt sich im Sichtbarkeitsbereich der Landmarke

Im Rahmen der visuellen Lokalisation werden folgende Themen behandelt:
Als Bildmerkmale dienen Geraden, die mittels der *Houghtransformation* [24] und eines neuen Algorithmus zur Extraktion der Geraden aus dem Akkumulatorfeld gewonnen werden [1]. Mit Hilfe der Projektionsmatrizen von kalibrierten Kameras können aus den Bildgeraden dreidimensionale Ebenen konstruiert werden. Zur räumlichen Anpassung (engl. *spatial fitting*) des projizierten CAD-Modells müssen Korrespondenzen zwischen Bild- und Modellentitäten gefunden werden (engl. *correspondence problem*). Gefundene Korrespondenzen erlauben eine Poseschätzung (engl. *pose estimation*), die durch Annahme kleiner Abweichungen (engl. *tracking*) zu einem *Matching* führt [43, 44].

Autonome mobile Roboter, die sich in der Welt bewegen, um ein bestimmtes Ziel zu erreichen, müssen in der Lage sein auf Veränderungen ihrer Umgebung in kurzer Zeit zu reagieren. Jederzeit muss das System Situationen ausweichen können, die seine Funktionalität beeinträchtigen, zu Unfällen führen oder Lebewesen gefährden könnten, wie z.B. sich schnell näherende Fahrzeuge, Fußgänger oder sich öffnende Türen etc. Diese Notwendigkeit führt zu Systemarchitekturen mit Modulen auf unterschiedlichen Abstraktionsebenen, die zeitlich unterschiedlich schnell die Situation erfassen und Maßnahmen ergreifen können. Module auf niedrigeren Abstraktionsebenen haben entsprechend weniger komplexe Aufgaben. Auf diese Weise sind sie fähig, in genügend kurzer Zeit zu reagieren [7]. So sollte z.B. die globale Wegplanung oder das Erkennen von Objekten dem Ausweichen vor Hindernissen untergeordnet sein.
Deshalb wird ein hierarchisches Sytem vorgestellt, das die obigen Anforderungen erfüllt.

1.2 Übersicht

Dieser Abschnitt soll kurz darstellen, wie das gesamte System arbeitet.
Der Roboter folgt einem vorgegebenen Weg, z.B. einem Rechteck innerhalb eines Raumes. Während der Bewegung wird Hindernissen ausgewichen. Hierbei dienen Sonar-, Infrarot- und taktile Sensoren zur Kollisionsvermeidung. Während der Bewegung wird der zurückgelegte Weg durch Messung der Umdrehungsanzahl der Rollen akkumuliert und dadurch

Abbildung 1.2: Projiziertes Modell der Landmarke (links) und erfolgreiches Matching (rechts)

die Position bestimmt (*Odometrie*). Durch Schlupf und Ungenauigkeiten in den Messeinheiten ist diese Position jedoch mit einem Fehler behaftet. Deshalb orientiert sich der Roboter an einer bekannten Landmarke (z.B. dem Schrank in Abbildung 1.1). Diese befindet sich nicht im Gesichtsfeld und muss deshalb zunächst zentriert werden. Ist die Landmarke sichtbar, wird ein Bild von ihr aufgenommen, anhand dessen die Roboterposition aktualisiert werden kann. Nach der Bildaufnahme wird die Bewegung fortgesetzt, während parallel die Bildverarbeitung erfolgt.

Mit Hilfe der durch die Odometrie ungefähr bekannten Position des Roboters wird das Modell der Landmarke in das Bild projiziert. Diese Projektion wird als Vorhersage für die Pixelkoordinaten der Landmarke im realen Bild verwendet. Der Odometriefehler erzeugt dabei eine Abweichung zwischen der Projektion und dem realen Bild.

Aus dem Bild der Landmarke werden mittels der Hough-Transformation Geraden extrahiert, die den Modelllinien zugeordnet werden. Jede dieser möglichen Zuordnungen definiert eine Pose, d.h. eine Transformation des Modells. Gesucht ist die Transformation, die das projizierte Modell mit dem realen Bild in Übereinstimmung bringt. Ein Beispiel für eine erfolgreich bestimmte Transformation ist in Abbildung 1.2 zu sehen. Die exakte Transformation zwischen Roboter und Landmarke wird nicht berechnet, sondern nur die relative 3D-Bewegung des Modells, die das projizierte Modell mit dem aufgenommenen Bild in Übereinstimmung bringt. Die gefundene, relative Transformation des Modells kann dann in eine Bewegung des Roboters umgerechnet werden, weil die Transformation zwischen Welt- und Roboterkoordinatensystem initial angegeben wird und die Kamera auf das Weltkoordinatensystem kalibriert ist.

Die Arbeit gliedert sich in 8 Kapitel. Nach der Einleitung erfolgt die Darstellung der mathematischen Grundlagen. Kapitel 3 erläutert wichtige Grundsätze der verhaltensbasierten Robotik, die die Grundlage für das Design des Gesamtsystems sind. Darauf erfolgt die Beschreibung der Bildvorverarbeitung und in Kapitel 5 wird das Matchingverfahren detailliert vorgestellt. Danach folgt die Beschreibung des verwendeten Roboters und der Systemarchitektur. Am Ende der Arbeit werden die implementierten Verfahren in verschiedenen Experimenten getestet, die im letzten Kapitel nochmal zusammengefasst und bewertet werden. Den Abschluss bildet ein Ausblick auf mögliche Erweiterungen.

Kapitel 2

Mathematischer Hintergrund

Dieses Kapitel gibt dem Leser die zum Verständnis der Arbeit notwendigen mathematischen Grundlagen. Ist der Leser mit den Begriffen *homogene Koordinaten, starre Bewegung, Abbildungseigenschaften einer Kamera, Hand-Auge-Kalibrierung* und *Poseschätzung kinematischer Ketten* bereits vertraut, so kann er dieses Kapitel überspringen und mit dem nächsten Kapitel fortfahren.

Zunächst erfolgt die in dieser Arbeit verwendete Notation von mathematischen Symbolen. Dann wird auf die Darstellungsformen von Rotationen eingegangen, insbesondere mittels Matrizen und Quaternionen. Darauf folgt eine Einführung in homogene Koordinaten und die damit leicht darzustellenden Abbildungsgleichungen für Kameras sowie die Bedeutung der Projektionsmatrix. Außerdem sind in dieser Arbeit Koordinatentransformationen im Zusammenhang der Hand-Auge-Kalibrierung von Bedeutung. Abschließend wird das Verfahren zur Poseschätzung für kinematische Ketten vorgestellt, das die Grundlage des Matching-Algorithmus darstellt (siehe Kapitel 5).
Dieses Kapitel gibt nur eine grobe Einführung in die einzelnen Themen. Der interessierte Leser kann in folgenden Arbeiten detailliertere Darstellungen finden [19, 20, 28, 29, 31, 35].

2.1 Notation

In dieser Arbeit werden folgende Schreibweisen verwendet:

- Allgemeine Parameter werden Kursiv dargestellt.
  Beispiel: ein Fehlerwert err_m.

- Vektoren und Punkte des $\mathbb{R}^n$ werden durch fett gedruckte Buchstaben beschrieben.
  Beispiel: ein 3D-Translationsvektor $\mathbf{t} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$.

- Matrizen sind durch Großbuchstaben zu erkennen:
  Beispiel: eine Projektionsmatrix $P^{3\times4} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$.

- Da oft homogene Koordiaten verwendet werden, unterscheiden wir zwischen dem Vektor in euklidischen Koordinaten $\mathbf{t} = (x, y, z)^T$ und dem entsprechenden homogenen Vektor $\bar{\mathbf{t}} = (x, y, z, 1)^T$ durch einen Strich über dem Buchstaben.
- Funktionen werden ebenfalls durch fett gedruckte Buchstaben beschrieben. Sie lassen sich von Vektoren durch die nachgestellten Funktionsparameter in Klammern unterscheiden.
  Beispiel: $\mathbf{f}(x)$ oder falls $\mathbf{x} \in \mathbb{R}^n$: $\mathbf{f}(\mathbf{x})$.

2.2 Rotationen

Zur Beschreibung von Rotationen gibt es unterschiedliche Modelle. Häufige Verwendung, vor allem in der Physik, findet die Beschreibung durch Matrizen mit *Eulerwinkeln*. Dabei werden Rotationen um eine beliebige Achse durch den Ursprung als Kombination von Rotationen um die Koordinatenachsen beschrieben.

Eine weitere Möglichkeit Rotationen zu beschreiben ist durch die Angabe der Rotationsachse und des Winkels. Die Achse kann z.B. als normalisierter Vektor angegeben werden. Oft werden diese Angaben in einer *Quaternion* [28] kodiert. Rotationsquaternionen werden vor allem in der Computergraphik [17] verwendet.

Das Verfahren zur Poseschätzung benutzt die Exponentialdarstellung für Rotationen, auf die ebenfalls kurz eingegangen wird.

2.2.1 Rotation mit Matrizen

Rotationsmatrizen sind orthogonale Matrizen, d.h. Matrizen A, für die gilt $A^T = A^{-1}$. Die Zeilen- und Spaltenvektoren sind also orthonormal zueinander. Diese Menge von Matrizen heißt auch *spezielle orthogonale Gruppe* $\mathbf{SO(n)}$ [31].

Zur Rotation von Punkten im zweidimensionalen hat die Rotationsmatrix R^2 folgende Gestalt,

$$R^2(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{2.1}$$

Dabei ist θ der Rotationswinkel, um den gedreht werden soll. Zu bemerken ist, dass die Rotation in der Ebene kommutativ ist, also für beliebige Rotationsmatrizen $R_1, R_2 \in \mathbf{SO(2)}$ gilt

$$R_1 R_2 = R_2 R_1. \tag{2.2}$$

Rotationen im 3D ($\mathbf{SO(3)}$) sind dagegen schwieriger zu beschreiben und sind insbesondere nicht kommutativ. Aus der Rotationsmatrix der Ebene lassen sich die drei Grundmatrizen R_x, R_y, R_z zur Beschreibung von 3D-Rotation herleiten,

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{2.3}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{2.4}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.5}$$

Dabei enstpricht die Multiplikation mit $R_x(\alpha)$ einer Rotation um die x-Achse, also in der y-z-Ebene, des Koordinatensystems mit dem Winkel α. Analoges gilt für die Matrizen $R_y(\beta)$ und $R_z(\gamma)$. Kombiniert man diese Grundmatrizen, so lassen sich Rotationen um beliebige Achsen durch den Ursprung beschreiben. Üblich ist die Darstellung mit *Eulerwinkeln*, d.h. die Reihenfolge der Rotationen ist festgelegt auf:

$$R(\alpha,\beta,\gamma) = R_z(\gamma)R_y(\beta)R_z(\alpha). \tag{2.6}$$

Wendet man R auf einen Punkt p an, so entspricht dies der Rotation des Punktes um die z-Achse mit dem Winkel α, dann der Rotation um die y-Achse mit Winkel β und zuletzt wieder der Rotation um die z-Achse mit Winkel γ.

2.2.2 Rotation mit Quaternionen

Definition 2.2.1 (Quaternion)
Eine Quaternion ist ein Quadrupel $q = (a,b,c,d)$ *mit* $a,b,c,d \in \mathbb{R}$. *Die Menge der Quaternionen wird mit* $\mathbb{H}$ *bezeichnet.*

Quaternionen können als Erweiterung der komplexen Zahlen aufgefasst werden. Es gibt drei imaginäre Zahlen, die zu -1 quadrieren und als $\mathbf{i}$, $\mathbf{j}$ und $\mathbf{k}$ bezeichnet werden. Für die Imaginärzahlen gilt

$$\mathbf{i^2} = \mathbf{j^2} = \mathbf{k^2} = -\mathbf{1} \qquad \mathbf{ij} = -\mathbf{ji} = \mathbf{k} \qquad \mathbf{jk} = -\mathbf{kj} = \mathbf{i} \qquad \mathbf{ki} = -\mathbf{ik} = \mathbf{j}. \tag{2.7}$$

Diese Regeln definieren die Multiplikation in $\mathbb{H}$, die Addition ist äquivalent zu den Komplexen Zahlen.

Bemerkung 2.2.1 *Die Menge der Quaternionen* $(\mathbb{H},+,\cdot)$ *bildet einen Schiefkörper.*

Zur Definition von Schiefkörpern siehe [31].

Die konjugierte Quaternion für $\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k};\ q_0,q_1,q_2,q_3 \in R$ ist definiert als $\mathbf{q}^\star = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}$. Der Betrag einer Quaternion ist:

$$\|\mathbf{q}\| = \sqrt{\mathbf{q}\mathbf{q}^\star} = \sqrt{q_0^2 + q_1^2 + q_3^2 + q_4^2} \tag{2.8}$$

Eine Quaternion $\mathbf{q_r}$, die eine Rotation um eine Achse (beschrieben durch den Richtungsvektor $\mathbf{v} = (v_1,v_2,v_3)$; $\|\mathbf{v}\| = 1$) mit dem Winkel θ durchführt, hat die Form,

$$\mathbf{q_r} = \cos\frac{\theta}{2} + \sin\frac{\theta}{2}(v_1\mathbf{i} + v_2\mathbf{j} + v_3\mathbf{k}). \tag{2.9}$$

Insbesondere gilt $\|\mathbf{q_r}\| = 1$, d.h. Rotationsquaternionen sind Einheitsquaternionen. Den rotierten Punkt $\mathbf{p'} = (p'_1,p'_2,p'_3)$ eines Punktes $\mathbf{p} = (p_1,p_2,p_3)$ erhält man durch,

$$(p'_1\mathbf{i} + p'_2\mathbf{j} + p'_3\mathbf{k}) = \mathbf{q_r}(p_1\mathbf{i} + p_2\mathbf{j} + p_3\mathbf{k})\mathbf{q_r^\star}. \tag{2.10}$$

2.2.3 Exponentialdarstellung für Rotationen

Eine weitere Möglichkeit zur Modellierung von Rotationen bietet die Exponentialschreibweise, die auf den dynamischen Prozess der Bewegung zurückgeht [35]. Betrachtet man die Bewegung eines Punktes $\mathbf{p}$ auf einer Ebene, die mit Einheitsgeschwindigkeit rotiert, dann gilt für den Geschwindigkeitsvektor (die Ableitung der Bewegung),

$$\dot{\mathbf{p}}(t) = \omega \times \mathbf{p}(t). \tag{2.11}$$

Dabei ist ω der auf der Ebene senkrecht stehende Normalenvektor und $\mathbf{p}(t)$ ist die Position des Punktes $\mathbf{p}$ zur Zeit t. Das Kreuzprodukt aus Gleichung (2.11) lässt sich auch als Matrixmultiplikation schreiben,

$$\omega \times \mathbf{p}(t) = \Omega\mathbf{p}(t) \qquad \mathrm{m}it \qquad \Omega = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{2.12}$$

Durch Integration der Gleichung (2.11) erhählt man die Exponentialdarstellung,

$$\mathbf{p}(t) = e^{\Omega t}\mathbf{p}(0). \tag{2.13}$$

Die Exponentialfunktion einer Matrix Ω wird definiert über die Reihenentwicklung für die Exponentialfunktion,

$$e^{\Omega} = \sum_{n=0}^{\infty} \frac{1}{n!}\Omega^n. \tag{2.14}$$

Die Rotation eines Punktes $\mathbf{p}$ um den Winkel θ erhält man, wenn man ihn eine Zeit t lang rotiert, dabei ist $t = \theta$,

$$e^{\Omega\theta} = \sum_{n=0}^{\infty} \frac{1}{n!}(\Omega\theta)^n. \tag{2.15}$$

Der Grenzwert dieser Reihe ist eine Matrix aus der Menge der Rotationsmatrizen $\mathbf{SO}(3)$.

2.3 Homogene Koordinaten

2.3.1 Einführung

Normalerweise drücken wir die Position eines Punktes in kartesischen Koordinaten aus. Z.B. benutzen wir im 2D-Raum die Schreibweise $\mathbf{x} = (x, y)$. Bei kartesichen Koordinaten haben wir aber Schwierigkeiten, wenn man Punkte im Unendlichen modellieren möchte. Außerdem möchten wir Richtungen von Punkten unterscheiden können. Dies führt zur Definition des *affinen Raums*:

Definition 2.3.1 (Affiner Raum)
Ein affiner Raum ist ein Tripel $(V, \vec{V}, +)$, bestehend aus einer nichtleeren Menge von Punkten V, einem Vektorraum $\vec{V}$ (mit Translationen und Richtungen) und einer Addition $+ : V \times \vec{V} \mapsto V$, das folgende Bedingungen erfüllt:
$(B1)$ $\exists!\mathbf{0} \in V, \forall \mathbf{a} \in V : \mathbf{a} + \mathbf{0} = \mathbf{0} + \mathbf{a} = \mathbf{a}$
$(B2)$ $\forall \mathbf{a} \in V,\ \mathbf{u}, \mathbf{v} \in \vec{V} : \mathbf{a} + (\mathbf{u} + \mathbf{v}) = (\mathbf{a} + \mathbf{u}) + \mathbf{v}$
$(B3)$ $\forall \mathbf{a}, \mathbf{b} \in V,\ \exists \mathbf{u} \in \vec{V} : \mathbf{a} + \mathbf{u} = \mathbf{b}$

Eine Möglichkeit, Punkte von Richtungen in der Darstellung zu unterscheiden, ist die Einführung einer weiteren Koordinate (einer *homogenen* Koordinate). Ein Punkt $\mathbf{x}$ im 2D-Raum wird dann beschrieben durch $\mathbf{x} = (x, y, 1)$ und eine Richtung durch $\mathbf{t} = (x, y, 0)$.

Die Verwendung homogener Koordinaten führt zur Einbettung des affinen Raums in den *projektiven Raum*:

Definition 2.3.2 (Projektiver Raum)
Sei V ein Vektorraum über dem Körper K. Der projektive Raum $P(V)$ ist die Menge $(V\backslash\{0\}, \sim)$ der Äquivalenzklassen von Vektoren in V ungleich Null mit der Äquivalenzrelation $\sim$, für die gilt,

$$\forall \mathbf{u}, \mathbf{v} \in V\backslash\{\mathbf{0}\} : \mathbf{u} \sim \mathbf{v} \Leftrightarrow \exists \lambda \in \mathbb{R} : \mathbf{v} = \lambda \mathbf{u}. \tag{3.16}$$

Zur weiteren Erklärung und Bedeutung des projektiven Raums siehe [19].
2D-Punkte des projektiven Raumes werden also nicht nur mit zwei Werten $\mathbf{x} = (x, y)$, sondern mit drei Werten $\overline{\mathbf{x}} = (x_1, x_2, x_3)$ ausgedrückt, sodass

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix} \tag{3.17}$$

Sei nun $\lambda \in \mathbb{R} : \lambda \neq 0$, dann beschreiben

$$\begin{pmatrix} \lambda x_1 \\ \lambda x_2 \\ \lambda x_3 \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

den äquivalenten Punkt. Entscheidend sind also nur die Verhältnisse der Elemente von $\overline{x}$.

Diese Koordinaten werden als *homogone Koordinaten* bezeichnet, weil eine Gleichung in (x, y) äquivalent ist mit einer *homogenen* Gleichung in (x_1, x_2, x_3).

Definition 2.3.3 (Lineare homogene Gleichung)
Sei o.B.d.A.

$$ax + by + cz + dw = h \tag{3.18}$$

mit den bekannten Konstanten a, b, c, d, h und den unbekannten $x, y, z, w \in \mathbb{R}$. Dann wird obige Gleichung als **lineare Gleichung** *bezeichnet. Ist zusätzlich $h = 0$, wird obige Gleichung als* **lineare homogene Gleichung** *bezeichnet.*

Als Beispiel folgt eine Gleichung, bei der alle Terme den gleichen Grad haben, wie die lineare Gleichung für Geraden im $\mathbb{R}^2$,

$$ax + by + c = 0. \tag{3.19}$$

Ersetzt man x durch x_1/x_3 und y durch x_2/x_3 so wird Gleichung (3.19) zu einer *homogenen* Gleichung, da:

$$\begin{aligned} & a\tfrac{x_1}{x_3} + b\tfrac{x_2}{x_3} + c = 0 \\ \Leftrightarrow \quad & ax_1 + bx_2 + cx_3 = 0 \end{aligned} \tag{3.20}$$

2.3.2 Homogene Koordinaten von 3D-Punkten

Dreidimensionale Punkte $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ werden analog zu Punkten des 2D-Raumes $\mathbf{y} \in \mathbb{R}^2$ homogenisiert, sodass für den homogenen Punkt $\overline{\mathbf{x}} = (x_1, x_2, x_3, x_4)$ gilt,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1/x_4 \\ x_2/x_4 \\ x_3/x_4 \end{pmatrix}. \tag{3.21}$$

Sei $\lambda \in \mathbb{R}\backslash\{0\}$ beliebig, dann enstprechen alle Punkte $\lambda\overline{\mathbf{x}}$ einer Äquivalenzklasse des Projektiven Raumes des $\mathbb{R}^3$. Für die Konvertierung von homogenen nach kartesischen Koordinaten muss lediglich das Verhältnis genommen werden,

$$\overline{\mathbf{x}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \rightarrow \mathbf{x} = \begin{pmatrix} x_1/x_4 \\ x_2/x_4 \\ x_3/x_4 \end{pmatrix}. \tag{3.22}$$

Ist $x_4 = 0$, so repräsentiert $\overline{\mathbf{x}}$ einen Punkt im Unendlichen, bzw. einen Richtungsvektor. Zu beachten ist, dass $\overline{\mathbf{x}} = \mathbf{0}$ keine Bedeutung hat und undefiniert ist.

Zur Konvertierung von kartesischen nach homogenen Koordinaten wird eine Dimension hinzugenommen und eine beliebige Skalierung $\lambda \in \mathbb{R}\backslash\{0\}$ eingeführt.

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \overline{\mathbf{x}} = \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \\ \lambda \end{pmatrix} \tag{3.23}$$

Da λ beliebig ist, wird üblicherweise $\lambda = 1$ gesetzt. Oft wird vereinfacht $\overline{\mathbf{x}} = (x, y, z, 1)^T$ als die Schreibweise für homogene Koordinaten zu $\mathbf{x} = (x, y, z)^T$ bezeichnet.

2.3.3 Starre Bewegung in homogenen Koordinaten

Eine *starre Bewegung* (RBM aus engl. *Rigid Body Motion*) ist eine Transformation, welche die Abstände innerhalb eines Objekts unverändert lässt. Eine beliebige RBM lässt sich ausdrücken als Hintereinanderausführung einer Translation $\mathbf{t}$ und einer Rotation R. Dazu werden die Elemente des starren Körpers jeweils einzeln transformiert. Besteht der Körper z.B. aus einer Menge von 3D-Punkten, so lässt sich eine RBM, die die 3D-Punkte $\mathbf{p_i}$ des Körpers auf 3D-Punkte $\mathbf{p'_i}$ abbildet, folgendermaßen ausdrücken,

$$\mathbf{p'_i} = R\mathbf{p_i} + \mathbf{t}. \tag{3.24}$$

Dabei ist R eine 3×3 Rotationsmatrix und $\mathbf{t} = (t_1, t_2, t_3)^T$ ein Translationsvektor. Die Menge dieser Tranformationen wird auch als *spezielle Euklidische Gruppe* bezeichnet $\mathbf{SE(3)}$, siehe auch [31].

Benutzt man nun homogene Koordinaten für die Punkte $\mathbf{p}$ und $\mathbf{p}'$, dann lässt sich die Gleichung (3.24) als Multiplikation mit einer 4×4 Matrix schreiben,

$$\overline{\mathbf{p}'} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \overline{\mathbf{p}}. \tag{3.25}$$

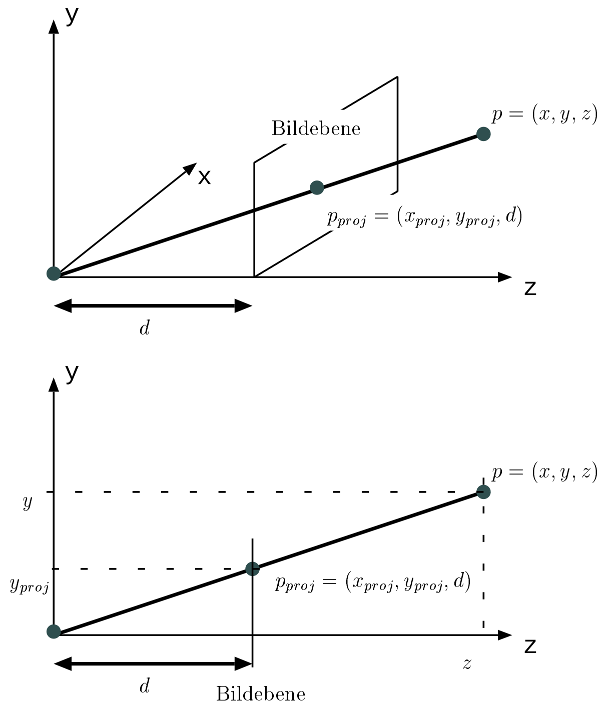

Abbildung 2.1: Perspektivische Projektion mit Lochkameramodell

Zu bemerken ist, dass durch die Erhöhung der Dimension um eins (homogene Koordinaten) die zuvor nichtlineare starre Bewegung linearisiert werden kann.

2.4 Abbildungseigenschaften einer Kamera

Eine Kamera bildet die 3D-Welt auf eine 2D-Ebene ab. Diese Abbildung ist im wesentlichen eine Transformation von 3D-Weltkoordinaten nach 2D-Pixelkoordinaten. Transformationen, die eine solche Dimensionsreduktion durchführen, nennt man auch Projektionen [15]. Zur Modellierung der Projektion gibt es verschiedene Möglichkeiten. Wir verwenden das *Lochkameramodell*, das im Folgenden dargestellt wird. Danach wird die Zusammensetzung der Projektionsmatrix P und die Bedeutung der *internen* und *externen* Kameraparameter erläutert.

2.4.1 Das Lochkameramodell

Das Lochkameramodell ist geeignet geeignet für die Modellierung von parallelen und perspektivischen Projektionen. Während die Parallelprojektion alle Punkte im 3D-Raum identisch auf die Bildebene abbildet (es entstehen keine Verzerrungen), ist die perspektivische Projektion besser geeignet für die Modellierung der tatsächlichen Abbildungseigenschaften einer Kamera, da sie die Entfernung der Punkte im Raum berücksichtigt und die fokale Länge der Kamera einbezieht. Es lässt sich zwar nicht jede Kamera mit dem Lochkameramodell beschreiben, durch die Verwendung geeigneter Objektive kann dies

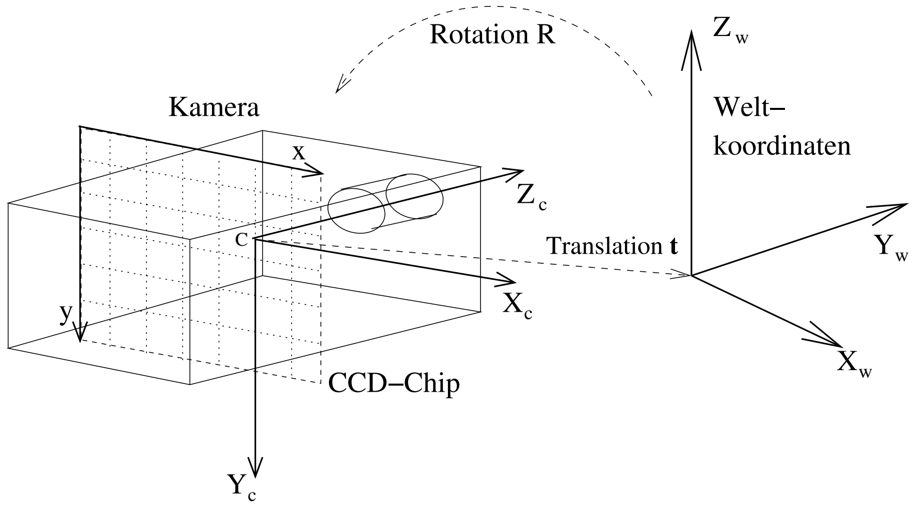

Abbildung 2.2: Für die Projektion relevante Koordinatensysteme und CCD-Chip (stark vergrößert dargestellt)

aber zumindest näherungsweise erreicht werden. Zu bemerken ist, dass eine Lochkamera keinen Unschärfebereich hat, wie es jedoch bei den meisten Kameras mit Objektiven der Fall ist.

Abbildung 2.1 zeigt die perspektivische Projektion. Ein Punkt $\mathbf{p} = (x, y, z)$ wird auf den Punkt $\mathbf{p_{proj}} = (x_{proj}, y_{proj})$ abgebildet. Unter Verwendung homogener Koordinaten lässt sich diese Abbildung schreiben als,

$$\overline{\mathbf{p_{proj}}} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \overline{\mathbf{p}} = P_p \overline{\mathbf{p}}. \tag{4.26}$$

Dabei ist $\overline{\mathbf{p_{proj}}} = (sx, sy, s)$ die homogene Repräsentation des Bildpunktes $\mathbf{x} = (x, y)$ und d ist die fokale Länge, d.h. der Abstand vom optischen Zentrum zur Bildebene.

2.4.2 Projektionsmatrix

In diesem Abschnitt wird die Zusammensetzung der Projektionsmatrix P erläutert, die 3D-Punkte $\mathbf{p_w}$ der Szene auf 2D-Pixel (u, v) des CCD-Chips abbildet.

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = P\ \overline{\mathbf{p_w}} \tag{4.27}$$

Die eigentliche Abbildung aus Gleichung (4.27) lässt sich in drei Teilschritte aufteilen. Der erste Schritt ist die Transformation von Weltkoordinaten (X_w, Y_w, Z_w) nach Kamerakoordinaten (X_c, Y_c, Z_c) (vgl. Abbildung 2.2). Diese Transformation ist äquivalent zu einer starren Bewegung, wie sie im vorigen Abschnitt beschrieben wurde. Um für einen

Punkt $\mathbf{p}_w$, beschrieben im Weltkoordinatensystem, die entsprechenden Koordinaten $\mathbf{p}_c$ im Kamerasystem zu ermitteln, ist also folgende Transformation notwendig,

$$\mathbf{p}_c = \begin{bmatrix} R & \mathbf{t} \\ 0 & 1 \end{bmatrix} \mathbf{p}_w = P_r \mathbf{p}_w. \tag{4.28}$$

Die drei Rotationswinkel der Matrix R und die drei Werte des Vektors t werden auch als *externe Kameraparameter* bezeichnet.

Der zweite Teilschritt ist die eigentliche Projektion, die die Punkte von 3D-Kamerakoordinaten (X_c, Y_c, Z_c) auf die Bildebene abbildet, wie in Gleichung (4.26) des Lochkameramodells beschrieben. Die Bildebene liegt in der Ebene des CCD-Chips; sie ist jedoch im Unterschied zum CCD-Chip nicht diskret und unendlich ausgedehnt.

Der dritte Teilschritt ist die Konvertierung nach Pixelkoordinaten $\mathbf{w} = (u, v)$.

$$\overline{\mathbf{w}} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} sx \\ sy \\ s \end{pmatrix} = P_c \begin{pmatrix} sx \\ sy \\ s \end{pmatrix} \tag{4.29}$$

Dabei ist $\overline{\mathbf{w}} = (su, sv, s)$ die homogene Repräsentation der Pixelkoordinaten (u, v). Die Skalierung k_u, k_v ist notwendig, falls die Zellen des CCD-Chips ein ungleiches Seitenverhältnis haben, also nicht quadratisch, sondern rechteckig sind. Zusätzlich beinhalten k_u, k_v eine Skalierung. Die Verschiebung u_0, v_0 berücksichtigt, dass der Nullpunkt des Pixelkoordinatensystems i. A. nicht identisch ist mit der Position des optischen Zentrums. Für die gesamte Projektionsmatrix P des Kameramodells ergibt sich,

$$\begin{aligned} P &= P_c P_p P_r \\ &= \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} & R & & \mathbf{t} \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{4.30}$$

Die Projektionsmatrix P ist eine 3×4 Matrix mit 11 Freiheitsgraden. Das Matrixprodukt aus $P_c P_p$ enthält die *internen* Kameraparameter d, k_u, k_v, u_0, v_0 und P_r die *externen* Parameter $R, \mathbf{t}$.

2.4.3 Kamerakalibrierung

Kamerakalibrierung beinhaltet die Bestimmung der Projektionsmatrix, welche die Abbildung von 3D-Punkten der Szene nach Pixelkoordinaten der Kamera als lineare Projektion beschreibt (bei Verwendung homogener Koordinaten). Wird nur die Projektionsmatrix geschätzt, so wird dies als *implizite* Kalibrierung bezeichnet. Werden außerdem die internen und externen Parameter berechnet, spricht man von *expliziter* Kalibrierung [55]. Die Bestimmung der 11 Werte der 3×4 Projektionsmatrix geschieht i. A. durch Lösung eines überstimmten linearen Gleichungssystems.

Dazu benötigt werden Messungen von korrespondierenden Punkten $(\mathbf{b^i}, \mathbf{w^i})$, dabei ist $\mathbf{b}^i = (u^i, v^i)$ der projizierte 2D-Punkt im Bild für den korrespondierenden 3D-Punkt $\mathbf{w}^i = (w_1^i, w_2^i, w_3^i)$ der Szene.

Zur Erzeugung des Gleichungssystems wird zunächst die Projektionsmatrix P in den Vektor $\mathbf{q} = (\mathbf{q_1}^T, q_{14}, \mathbf{q_2}^T, q_{24}, \mathbf{q_3}^T, q_{34})$ umnotiert,

$$P = \begin{bmatrix} \mathbf{q_1}^T & q_{14} \\ \mathbf{q_2}^T & q_{24} \\ \mathbf{q_3}^T & q_{34} \end{bmatrix}. \tag{4.31}$$

Damit folgt für ein Paar $(\mathbf{b^i}, \mathbf{w^i})$ und der Projektionsgleichung $\overline{\mathbf{b}} = P\overline{\mathbf{w}}$,

$$\begin{array}{rcl} \mathbf{q}_1^T\mathbf{w}^i - u^i\mathbf{q}_3^T\mathbf{w}^i + q_{14} - u^i q_{34} & = & 0 \\ \mathbf{q}_1^T\mathbf{w}^i - v^i\mathbf{q}_3^T\mathbf{w}^i + q_{24} - v^i q_{34} & = & 0 \end{array}. \tag{4.32}$$

Für n Paare erhält man aus (4.32) und der $2n \times 12$ Matrix

$$A = \begin{bmatrix} w_1^1 & w_2^1 & w_3^1 & 1 & 0 & 0 & 0 & 0 & -u^1w_1^1 & -u^1w_2^1 & -u^1w_3^1 & -u^1 \\ 0 & 0 & 0 & 0 & w_1^1 & w_2^1 & w_3^1 & 1 & -v^1w_1^1 & -v^1w_2^1 & -v^1w_3^1 & -v^1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ w_1^n & w_2^n & w_3^n & 1 & 0 & 0 & 0 & 0 & -u^nw_1^n & -u^nw_2^n & -u^nw_3^n & -u^n \\ 0 & 0 & 0 & 0 & w_1^n & w_2^n & w_3^n & 1 & -v^1w_1^n & -v^1w_2^n & -v^1w_3^n & -v^n \end{bmatrix} \tag{4.33}$$

das i. A. überbestimmte Gleichungssystem

$$A\mathbf{q} = 0. \tag{4.34}$$

Um die triviale Lösung $\mathbf{q} = \mathbf{0}$ auszuschließen und um die numerische Stabilität zu erhöhen, fordern wir, dass $q_{34} = 1$ gilt. Dies ist möglich, weil die Projektionsmatrix bis auf einen Skalierungswert definiert ist [15]. Die Forderung $q_{34} = 1$ schließt die *explizite* Kalibrierung aus, d.h. aus der skalierten Projektionsmatrix kann man die *externen* und *internen* Kameraparameter nicht mehr berechnen. Wir benötigen aber in unserer Arbeit nur die implizite Kalibrierung. Mit dieser Forderung wird Gleichung (4.33) zu

$$\begin{bmatrix} w_1^1 & w_2^1 & w_3^1 & 1 & 0 & 0 & 0 & 0 & -u^1w_1^1 & -u^1w_2^1 & -u^1w_3^1 \\ 0 & 0 & 0 & 0 & w_1^1 & w_2^1 & w_3^1 & 1 & -v^1w_1^1 & -v^1w_2^1 & -v^1w_3^1 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ w_1^n & w_2^n & w_3^n & 1 & 0 & 0 & 0 & 0 & -u^nw_1^n & -u^nw_2^n & -u^nw_3^n \\ 0 & 0 & 0 & 0 & w_1^n & w_2^n & w_3^n & 1 & -v^1w_1^n & -v^1w_2^n & -v^1w_3^n \end{bmatrix} \mathbf{q} = \begin{pmatrix} u^1 \\ v^1 \\ \cdot \\ u^n \\ v^n \end{pmatrix}. \tag{4.35}$$

Die Lösung des Gleichungsystems (4.35) $A\mathbf{q} = \mathbf{c}$ lässt sich mit der Pseudoinversen $A^+ = (A^TA)^{-1}A^T$ berechnen,

$$\mathbf{q} = A^+\mathbf{c}. \tag{4.36}$$

Aus Gründen der numerischen Stabilität wird jedoch ein auf dem Householder-Verfahren basierendes iteratives Verfahren aus [40] auf das Gleichungsystem

$$(A^TA)\mathbf{q} = A^T\mathbf{c} \tag{4.37}$$

angewendet.

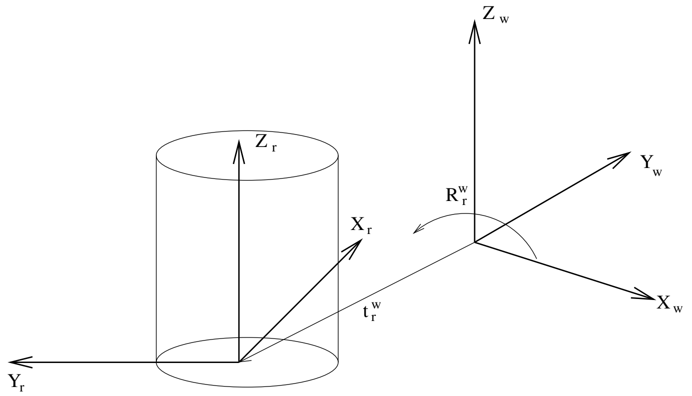

Abbildung 2.3: Koordinatentransformation der Hand-Auge-Kalibrierung

2.5 Hand-Auge-Kalibrierung

Die Hand-Auge-Kalibrierung beschreibt die Beziehung zwischen Koordinatensystemen [29]. In dieser Arbeit sind dies das Weltkoordinatensystem, auf das die Kameras kalibriert sind, und das Roboterkoordinatensystem, in dem die Bewegung erfolgt. Die Transformation T_r^w zwischen beiden Systemen ist eine starre Bewegung, die sich durch eine 4×4 Matrix beschreiben lässt,

$$T_r^w = \begin{bmatrix} R_r^w & \mathbf{t}_r^w \\ \mathbf{0} & 1 \end{bmatrix}. \tag{5.38}$$

Dabei ist R_r^w die Rotationsmatrix, die das Weltkoordinatensystem so dreht, dass die Achsen parallel zu den Achsen des Roboterkoordinatensystems stehen. Der Translationsvektor $\mathbf{t}_r^w$ beschreibt die notwendige Verschiebung, sodass beide Koordinatensysteme übereinanderliegen. Benötigt man die Roboterkoordinaten eines Punktes $\mathbf{p_w}$ in Weltkoordinaten, so erhält man diese durch

$$\overline{\mathbf{p_r}} = T_r^w \overline{\mathbf{p_w}}. \tag{5.39}$$

In Abbbildung 2.3 ist die Transformation von Welt- nach Roboterkoordinaten sichtbar, der Zylinder soll einen Roboter andeuten. Die Rotation R_r^w findet bei der abgebildeten Transformation nur in der x-y-Ebene statt.

2.6 Poseschätzung starrer Körper

Poseschätzung bedeutet die Orientierung und Position eines Objektes innerhalb eines gegeben Koordinatensystems zu bestimmen [15, 41, 49]. Dabei lassen sich drei Varianten unterscheiden. Erstens die 2D-2D Poseschätzung, bei der ein 2D-Modell so verschoben, rotiert und skaliert wird, dass es mit einem aufgenommenen Bild übereinstimmt.

Zweitens die 3D-3D Poseschätzung, die eine 3D-Modelltransformation berechnet, sodass dieses Modell mit gegebenen 3D-Informationen übereinstimmt. Diese Schätzung ist ähnlich zum 2D-2D Fall, mit dem Unterschied, dass eine größere Anzahl freier Parameter bestimmt werden muss.

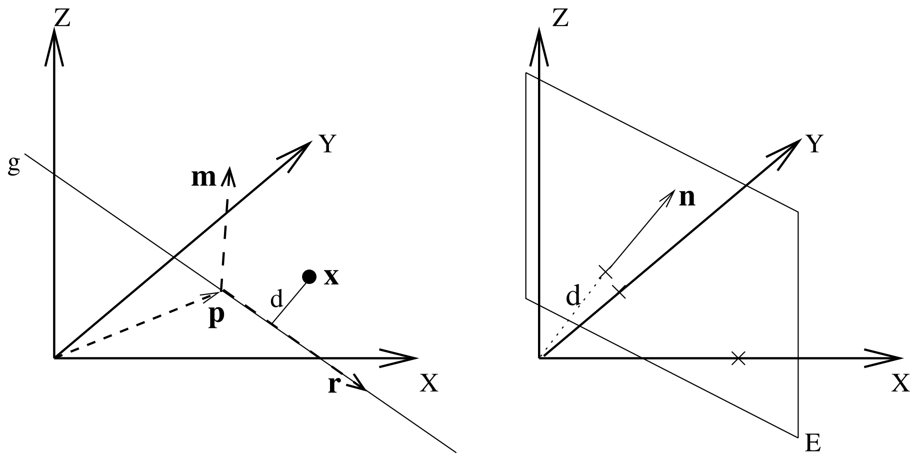

Abbildung 2.4: Darstellung von Geraden (links) und Ebenen(rechts)

In dieser Arbeit wird die dritte Variante, die 2D-3D Poseschätzung verwendet. Eine 3D-Modelltransformation muss ermittelt werden, sodass die Projektion dieses Modells mit einem aufgenommenen 2D-Bild übereinstimmt. Um das 3D-Modell mit den 2D-Bildinformationen zu vergleichen, rekonstruieren wir aus den extrahierten Bildelementen dreidimensionale Entitäten. Wir benutzen eine implizite Beschreibung des 2D-3D Poseschätzungproblems. Dabei werden Gleichungen für korrespondierende Entitäten aufgestellt. Der zu minimierende Gesamtfehler beschreibt die Abstände zwischen 3D-Ebenen und 3D-Punkten. Die Ebenen werden aus den extrahierten Bildgeraden und der Projektionsmatrix gewonnen (siehe schraffierte Fläche in Abbildung 2.6). Die Minimierung dieser Abstände liefert eine Pose, die abhängig ist von der Zuordnung der extrahierten Bildgeraden zu Modellgeraden.

Im folgenden beschreiben wir kurz das Verfahren zur Poseschätzung aus [20, 49], soweit es zum Verständnis dieser Arbeit notwendig ist.

2.6.1 Darstellung von 3D-Ebenen und Geraden

Eine Gerade im Raum kann durch zwei Vektoren dargestellt werden, z.B. durch Richtungs- und Ortsvektor. Wir nutzen jedoch implizite Beschreibungen, bei der Geraden des $\mathbb{R}^3$ durch Richtung und Moment dargestellt werden, was auch als *Plücker-Koordinatendarstellung* bekannt ist. Für die Darstellung mit *Plücker-Koordinaten* benötigt man das Moment $\mathbf{m}$, das sich ergibt aus dem Kreuzprodukt von Ortsvektor $\mathbf{p}$ und Richtungsvektor $\mathbf{r}$,

$$\mathbf{m} = \mathbf{p} \times \mathbf{r}. \tag{6.40}$$

Um zu testen, ob ein Punkt $\mathbf{x}$ auf einer Geraden $g = (\mathbf{r}, \mathbf{m})$ liegt, muss lediglich folgende Gleichheit überprüft werden,

$$g: \quad \mathbf{x} \times \mathbf{r} - \mathbf{m} = \mathbf{0}. \tag{6.41}$$

Ist der Richtungsvektor $\mathbf{r}$ vor der Berechnung des Moments normalisiert worden und ergibt sich ein Ergebnisvektor $\mathbf{x}'$ ungleich Null, so liegt der Punkt mit dem Abstand $d = |\mathbf{x}'|$

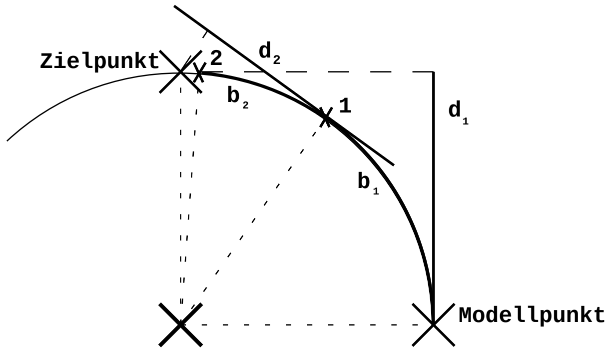

Abbildung 2.5: Iterative, lineare Annäherung der Rotation

neben der Geraden. Dieser Abstand d ist der minimale Euklidische Abstand des Punktes zur Geraden. Abbildung 2.4 links zeigt die geometrischen Verhältnisse der Geradendarstellung.

Ebenen lassen sich genau wie Geraden durch die sogenante *Hesseform* implizit beschreiben. Diese benutzt den zur Ebene $E = (\mathbf{n}, d)$ senkrecht stehenden Normalenvektor $\mathbf{n}$ und einen Skalar d (Hessedistanz), der dem minimalen Abstand der Ebene zum Nullpunkt des Koordinatensystems entspricht. Um zu testen, ob ein Punkt $\mathbf{x}$ auf der Ebene liegt, muss lediglich folgende Gleichheit überprüft werden:

$$E : \quad \mathbf{x} \cdot \mathbf{n} - d = 0 \tag{6.42}$$

Auch hier entspricht ein Wert ungleich Null dem minimalen Abstand des Punktes $\mathbf{x}$ zur Ebene. Abbildung 2.4 rechts zeigt die geometrischen Zusammenhänge der Ebenendarstellung.

2.6.2 Kinematische Kette

Eine kinematische Kette ist eine Aneinanderreihung von starren Körpern. Dabei können die starren Körper untereinander durch Gelenke miteinander verbunden sein. Diese Gelenke können nahezu beliebige Formen haben. In dieser Arbeit verwenden wir aber nur Rotationsgelenke, d.h. Gelenke die einen freien Parameter haben, den Rotationswinkel. Ein typisches Beispiel für eine solche Kette ist ein Roboterarm.

Zur Poseschätzung einer solchen kinematischen Kette gehört also nicht nur die Bestimmung der Position und Orientierung, sondern auch die Ermittlung der Rotationswinkel jedes Gelenks. Für diese Art der Parameterschätzung gibt es verschiedene Ansätze. Das in dieser Arbeit verwendete Verfahren aus [20] bestimmt alle Parameter gleichzeitig mittels Lösung und Iteration linearer Gleichungssysteme.

Iterative Berechnung der Bewegung

Die Exponentialdarstellung für Rotationen als Reihenentwicklung hat unendlich viele Summanden (siehe Abschnitt 2.2.3). Die unendliche Reihe lässt sich aber durch linea-

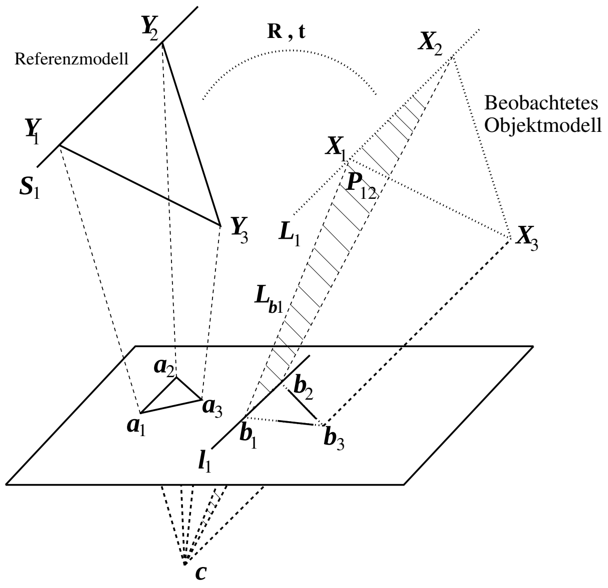

Abbildung 2.6: Referenz- und beobachtets Modell zur Erzeugung der Zwangsbedingungen

risieren und iterieren annähern.

Das Prinzip dieses iterativen Ansatzes zeigt Abbildung 2.5 am Beispiel einer Rotation in der Ebene. Die durchzuführende Bewegung des Modellpunktes zum Zielpunkt ist eine Drehung um 90°. Diese wird schrittweise linear angenähert.

Der minimale Abstand zwischen der Tangentialgeraden durch den Modellpunkt und dem Zielpunkt liefert einen Punkt, der den Abstand $\mathbf{d_1}$ zum Modellpunkt hat. Der Kreisbogen $\mathbf{b_1}$ mit der gleichen Länge wie $\mathbf{d_1}$ liefert den Ausgangspunkt für den nächsten Iterationsschritt (gekennzeichnet mit **1**). Nach dem gleichen Verfahren gelangt man zu Punkt **2** etc. Für unsere Arbeit sind 10 Iterationen ausreichend. Zu bemerken ist, dass ein Winkel von exakt 180° nicht linear angenähert werden kann, weil in dem Fall der Abstand zwischen Tangentialgerade und Zielpunkt bereits im Startpunkt minimal ist.

2.6.3 Geometrische Zwangsbedingungen

Die 2D-3D Poseschätzung hat das Ziel, ein 3D-Modell an gegebene 2D-Bildinformationen anzupassen. Da die extrahierten Bildelemente meist ungenau oder verrauscht sind, muss das poseschätzende System mit Abweichungen umgehen können [46].

Abbildung 2.6 zeigt die Situation des aufgenommenen Objektes mit den Bildpunkten $\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}$ und der angenommenen Position des Objekts $\mathbf{Y_1}, \mathbf{Y_2}, \mathbf{Y_3}$. Die unendlich ausgedehnte projektive Ebene $\mathbf{L_{b1}}$ wurde berechnet mittels des optischen Zentrums $\mathbf{c}$ und der von $\mathbf{b_1}, \mathbf{b_2}$ gebildeten Geraden $\mathbf{l_1}$.

Ziel der Poseschätzung ist nun die Transformation $(\mathbf{R}, \mathbf{t})$ so zu bestimmen, dass der Abstand zwischen den Modellpunkten $\mathbf{Y_1}, \mathbf{Y_2}, \mathbf{Y_3}$ und den projektiven Ebenen $\mathbf{L_{b1}}, \mathbf{L_{b2}}, \mathbf{L_{b3}}$ (nur $\mathbf{L_{b1}}$ eingezeichnet) minimal wird. Die Pose des Objekts mit minimalem Abstand ist zu sehen als $\mathbf{X_1}, \mathbf{X_2}, \mathbf{X_3}$.

Die in dieser Arbeit benutzte Zwangsbedingung ist die Eigenschaft, dass ein transformierter 3D-Modellpunkt auf der korrespondierenden, beobachteten projektiven Ebene liegen muss, bzw. der Abstand minimal sein muss.

Neben der Verwendung der Punkt/Ebene Bedingung können zur Poseschätzung auch Punkt/Punkt und Punkt/Linien Zwangsbedingungen benutzt werden. Dies ist z.B. nützlich, wenn einzelne markante Bildpunkte extrahiert werden, die in eine Beziehung zu Modellpunkten gesetzt werden sollen.

Aus diesen Zwangsbedingungen werden lineare Gleichungssysteme gebildet, die dann mit einem auf dem Householder-Ansatz basierenden Verfahren gelöst werden [40].

Minimalvoraussetzungen

Zur Lösung des Gleichungssystems muss genug Information vorhanden sein, d.h. es müssen genügend Korrespondenzen zwischen observiertem Objekt (Bildelemente) und Modellelementen hergestellt werden. Zur Schätzung der Basiskonfiguration ist es notwendig, die sechs Parameter der starren Bewegung $(R, \mathbf{t})$ zu bestimmen. Bei Verwendung der Punkt/Ebene Bedingung sind sechs Korrespondenzen notwendig. Für jedes weitere Glied der Kette ist eine weitere Korrespondenz nötig, damit der Gelenkwinkel bestimmt werden kann.

Wir fordern, dass wenigstens 7 Korrespondenzen benutzt werden, weil sich dadurch die Wahrscheinlichkeit erhöht, dass die gefundene Pose bei guter Anpassung der gesuchten Pose entspricht. Abhängig vom verwendeten Modell ist eine noch größere Anzahl von Korrespondenzen notwendig.

Kapitel 3

Design von Verhaltensbasierten Systemen

Das Design eines Robotersystems ist von entscheidender Bedeutung für seine Funktionsweise und beeinflusst Eigenschaften wie *Robustheit, Adaptivität* und *Effektivität* [9]. Deshalb wird in diesem Kapitel auf zwei verschiedene Konstruktionen für künstliche, intelligente Systeme eingegangen. Zunächst erfolgt eine kurze Beschreibung der klassischen Systeme und darauf eine Vorstellung der verhaltensbasierten Systeme und ihre Bedeutung für diese Arbeit. Dieses Kapitel orientiert sich inhaltlich an Brooks und Teilaspekte sind entnommen aus [10].

3.1 Klassische Architektur

Klassiche Systeme wie Wissensbasierte oder Experten-Systeme zeichnen sich durch eine *Top-Down* Architektur aus. In der Robotik werden dabei oft Informationen mehrerer unterschiedlicher Sensoren in eine abstrakte Repräsentation der Welt integriert (*Sensor Fusion* [34]). Basierend auf dieser Repräsentation und dem zu erreichenden Ziel ermittelt eine zentrale Entscheidungseinheit die nächste Handlung. Die Verarbeitung der Daten erfolgt dabei meist streng sequentiell mit folgenden Teilschritten (vgl. Abb. 3.1):
Wahnehmung - Modellierung - Planung - Handlung.

Die Planungseinheit ist der zentrale Sitz der Intelligenz, die anhand einer symbolischen Repräsentation der Welt Entscheidungen trifft. Die Art der Symbole und ihr Zusammenhang zur realen Welt sind maßgebend für die Funktionsweise des gesamten Systems.

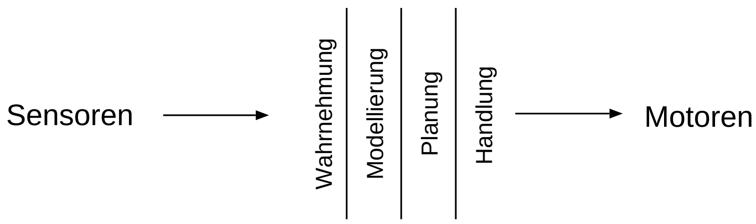

Abbildung 3.1: Übliche Reihenfolge der Verarbeitung bei klassischen Systemen

Obwohl klassische Systeme in künstlich konstruierten Umbegungen bemerkenswerte Ergebnisse erzielten, haben sie doch ernste Probleme in dynamischen, realen Umgebungen. Einerseits ist eine explizite Modellierung und eine Definition von Regeln für alle möglichen Situationen nicht praktikabel und andererseits muss das System u.U. sehr kurzfristig auf veränderte Situationen reagieren können. Eine Berechnung des gesamten Verarbeitungszyklus ist dabei oft zu aufwendig und langwierig. Deshalb fehlt es klassischen Systemen oft an *Adaptivität* und *Robustheit* besonders in Real-Zeit-Anwendungen mit sich schnell verändernden Rahmenbedingungen.

3.2 Verhaltensbasierte Architektur

Die verhaltensbasierte Architektur beruht auf der Betrachtung von realen, biologischen Lebewesen, die in der Welt situiert sind und autonom Aufgaben erledigen können. Diese Betrachtung führt zu einer *bottom-up* Strategie für intelligente Systeme. Verschiedene Schichten stehen dabei für unterschiedliche Kompetenzen mit eigenständigen Verhaltensweisen und befinden sich in einer hierarchischen Struktur, in der höhere Schichten komplexere Verhalten besitzen.

In [7] wird erstmals der Begriff *subsumption architecture* genannt, auf dem sich die verhaltensbasierten Systeme begründen. Die *subsumption architecture* basiert auf der Idee, dass Schichten, die in der Hierarchie weiter unten stehen, die Aktionen von höheren Schichten unterdrücken können, d.h. die unteren Schichten sollen vollständig autonom agieren können, ohne etwas von den höheren, komplexeren Schichten wissen zu müssen. Insbesondere müssen für die unteren Schichten immer genug Ressourcen vorhanden sein (Energie, Rechenkapazität, Sensorzugriff), sodass ihre Funktionsweise gewährleistet ist. Jede Schicht arbeitet deshalb in einem eigenen Zeitrahmen, also asynchron zu den anderen Schichten und ist in der Lage, die Aktionen von höheren Schichten zu unterdrücken. So sollte z.B. die Kollisionsvermeidung immer Vorrang haben gegenüber Lokalisierungsmodulen. Ein Beispiel für Schichten eines mobilen Roboters ist in Abbildung 3.2 dargestellt.

Ein System mit einer *subsumption* ähnlichen Architektur garantiert die Funktionsweise niedrigerer Schichten. Die Intelligenz des Systems ist nicht in einer zentralen Einheit zu finden und es wird keine abstrake symbolische Repräsentation der Welt aufgebaut oder zur Planung von Aktionen verwendet.

Die Merkmale von verhaltensbasierten Systemen nach Brooks [8, 9] lassen sich wie folgt zusammenfassen:

- Die biologische Evolution ist die Grundlage für Designmethoden, in denen Verbesserung der Leistung aus inkrementell hinzugefügter, situationsbedingter Verschaltung (als Hard- oder Software) entsteht.
- Situiertheit: Der Roboter befindet sich in der realen Welt und nicht in ihrer abstrakten Beschreibung. Es existiert keine symbolische Repräsentation der Welt.
- Körperlichkeit: Der Roboter hat einen physisch existierenden Körper und erfährt die Welt direkt. Seine Aktionen sind Teil dieser Welt und er erhält augenblicklich Feedback durch seine Aktivitäten.

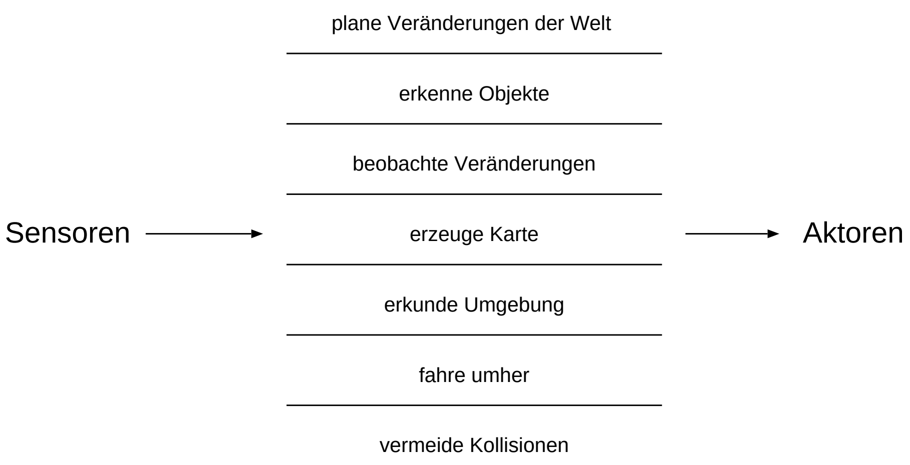

Abbildung 3.2: Mögliche Schichten einer verhaltensbasierten Architektur (aus [7])

- Emergenz: Es existiert kein zentraler Ort der Intelligenz. Die einzelnen Teile des Systems erzeugen erst als Gesamtes für den externen Beobachter das intelligente Verhalten. (Das Ganze ist mehr als die Summe seiner Teile [2].)
- Das Gesamtverhalten des Systems wird in einzelne Module von Wahrnehmung und zugehöriger Handlung zerlegt.
- Die Verbindungen/der Austausch zwischen einzelnen Schichten soll so klein wie möglich sein.
- Jede Schicht sollte eine enge Verknüpfung von Wahrnehmung und Handlung haben.
- Das System ist durch Hinzufügen von Modulen hierarchisch erweiterbar.

Nach vielen Jahren Erfahrung mit verhaltensbasierten Ansätzen lassen sich im Bereich der künstlichen Intelligenz und Robotik verschiedene verhaltensbasierte Systeme bezüglich ihrer Motivation einordnen.
Dies sind z.B.:

- **Biologische Motivation**: Untersuchungen aus der Verhaltenslehre und der Neurophysiologie bilden die Grundlage für die Zerlegung des Problems in Einzelverhalten. Die Reaktion des Systems wird mit denen von Tieren verglichen, um die Zerlegung zu verifizieren oder zu verbessern.

- **Experimentell ermitteltes Design**: Das System wird zunächst mit einer Reihe von Grundfähigkeiten ausgestattet, die dann in Experimenten getestet und deren Art des Zusammenwirkens überprüft wird. Entspricht das Gesamtverhalten den Erwartungen, so werden weitere Einzelverhalten hinzugefügt. Diese Vorgehensweise entspricht einer *Bottom-Up* Strategie.

Außerdem lassen sich zwei Varianten der Verknüpfung von Einzelverhalten unterscheiden:

- **Kooperativ**: Verschiedene Verhalten produzieren Ausgaben, die sich verknüpfen lassen, so z.B. Positionsbestimmungen verschiedener Navigationsmodule.
- **Konkurierend**: Verschiedene Verhalten erzeugen Aktionen, die nicht gleichzeitig vom System ausführbar sind. Einige diese Aktionen müssen deshalb unterdrückt werden.

Die Einzelverhalten, auch Module genannt, werden oft als deterministische Automaten implementiert. Die Anzahl der Zustände ist dabei oft sehr klein, in dieser Arbeit genügen 3 Zustände zur Beschreibung des Verhaltens einzelner Module.

3.3 Bemerkungen zum verhaltensbasierten Ansatz

Im Rahmen der KI (*Künstliche Intelligenz*) und besonders in der Robotik wird oft ein verhaltensbasierter Ansatz zum Systemdesign verwendet, jedoch gibt es keine klare Strategie zur Zerlegung eines Problems in einzelne Verhalten. Insbesondere ist eine solche Zerlegung immer abhängig vom gewünschten Ziel und muss sich an gegebene Bedingungen anpassen. Unterschiedliche physikalische Gegebenheiten der Umgebung oder andere Sensoren/Aktuatoren können eine völlig neue Zerlegung des Gesamtverhalten notwendig machen.

Viele Arbeiten beschäftigen sich deshalb mit der Strategie zur Entwicklung verhaltensbasierter Systeme, z.B. [4, 39, 48]. Einige für uns interessante Aspekte sind folgende:

- Hält man sich strikt an die von Brooks aufgestellten Grundsätze, so ist es schwierig komplexe Aufgaben zu implementieren, die Planung und Schlussfolgerungen beinhalten. Deshalb ist es manchmal notwendig, um ein bestimmtes Ziel zu erreichen, dieses in Teilziele zu unterteilen und auf verschiedene Module zu verteilen [54]. Dies trifft besonders auf Probleme der Roboternavigation zu.
- Das System sollte ein Erinnerungsvermögen besitzen. Zur Lösung von hochkomplexen Aufgaben ist es absolut notwendig, dass Sensordaten oder Erkenntnisse/ Erfahrungen, die in der Interaktion mit der Umgebung gemacht werden, zumindest für kurze Zeit gespeichert werden.
- Das Erlernen von Kompetenzen spielt eine entscheidende Rolle für das erfolgreiche Verhalten eines Systems. Lernaspekte können sein: Anpassung von Parametern, z.B. für die Bildvorverarbeitung, die Unterdrückung von konkurrierenden Verhalten oder die Zerlegung in Einzelverhalten. Im Extremfall kann sogar das Gesamtverhalten in der Interaktion gelernt werden, z.B. Navigation oder Kollisionsvermeidung durch *Reinforcement-Learning* [21].

Zusammenfassend lässt sich sagen, dass die von Brooks ursprünglich geforderten Merkmale zur Konstruktion von verhaltensbasierten Systemen sich so nur auf einfache Verhalten anwenden lassen, die vergleichbar sind mit den motorischen Fähigkeiten des Menschen, wie z.B. das Gehen mit Gewichtsverlagerung ohne umzufallen oder das Fixieren von interessanten Objekten im Gesichtsfeld [10] etc. Viele diese Verhalten werden vom Menschen unbewusst ausgeführt. Komplexere Verhalten, deren Hauptaufgabe nicht die Ansteuerung von Motoren ist, lassen sich so nur schwer realisieren.

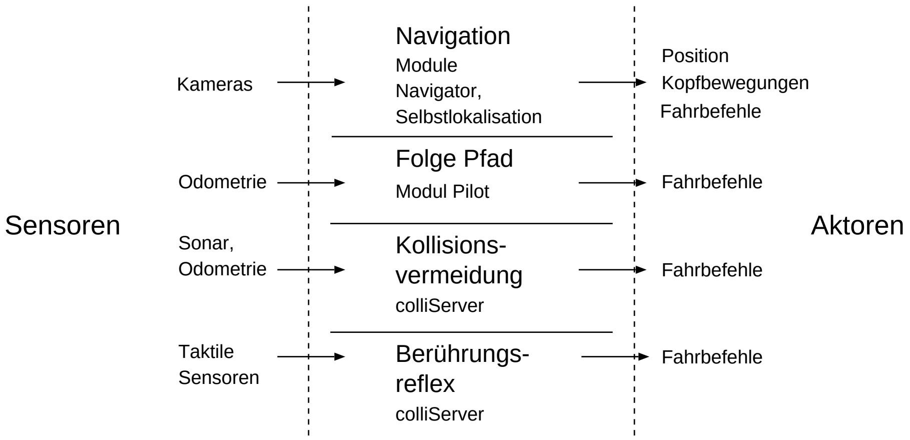

Abbildung 3.3: Implementierte Verhaltensschichten in dieser Arbeit

3.4 Einordnung dieser Arbeit

Der Aufbau unseres Systems zur visuellen landmarkenbasierten Navigation soll nicht nur die Selbstlokalisation ermöglichen, sondern auch als Grundlage für weitere Arbeiten dienen. Deshalb gestalten wir das gesamte System gut strukturiert, schnell nachvollziehbar und leicht erweiterbar.

Die implementierte Struktur ist angelehnt an die *subsumption architecture* nach Brooks. Wir legen auch besonderen Wert auf ein hierarchische Schichtenmodell, bei dem jede Schicht autonom und in einem asynchronen, parallelen Zeitrahmen arbeitet. Die Funktionsweise unterer Schichten soll durch einen Ausfall oberer Schichten nicht beeinflusst werden. Untere Schichten haben die Möglichkeit die Aktionen höher liegender Schichten zu unterdrücken.

Die Zerlegung der Verhalten dieser Arbeit wird in Abbildung 3.3 deutlich. Zur detaillierten Beschreibung einzelner Module siehe Kapitel 6.

- Auf der untersten Ebene steht der **Berührungsreflex**. Registrieren die frontalen taktilen Sensoren eine Kollision, so setzt der Roboter automatisch etwa $10cm$ zurück in die entgegengesetzte Richtung.

- **Kollisionsvermeidung**. Diese Schicht ist selbstständig in der Lage Hindernisse zu umfahren und die Bewegungsfreiheit zu gewährleisten. Es werden Ultraschall- und taktile Sensoren verwendet. Dieses Verhalten erzeugt ausschliesslich Fahrbefehle.

- Die dritte Abstraktionsschicht ist das Verhalten **Folge Pfad**, dessen Aufgabe es ist, entlang von Wegpunkten einem vorgegeben Pfad zu folgen. Bislang wird dieser Pfad manuell angegeben, er kann aber auch von einem komplexeren Verhalten erzeugt sein, das in der Lage ist in globalem Rahmen Wegpunkte zu berechnen. Diese Schicht benutzt nur die Odometriedaten des Roboters und erzeugt Fahrbefehle, die an die Kollisionsvermeidung weitergegeben werden. Wie die Wegpunkte von der unteren Schicht erreicht werden ist nicht genau bekannt und muss deshalb ständig anhand der Odometriewerte geprüft werden.

- Die höchste Schicht bildet das Verhalten **Navigation**, dessen Aufgabe die Positionsbestimmung des Roboters ist. Dazu werden die Kameras des Roboters und a-priori-Wissen über Landmarken in der Umgebung genutzt. Diese Schicht erzeugt Bewegungsbefehle für den Kamerakopf zur Zentrierung des Objekts und die Abweichung der Odometriedaten von der tatsächlichen Position. In dieser Schicht können auch andere Navigationsverfahren implementiert werden.

Die Module der Schicht Navigation besitzen ein kooperatives Verhalten, d.h. ihre Ergebnisse (Positionsbestimmungen) lassen sich kombinieren. Z.B. realisiert Arkin [3] die Kooperation dadurch, dass alle Verhalten Vektorfelder zur Navigation erzeugen. Kooperation findet durch Addition der Vektoren statt.

Verhalten, die Aktionen für die gleichen Motoren erzeugen, sind meistens konkurrierend. Hier wird das konkurrierende Verhalten der drei Schichten bezüglich der Fahrbefehle realisiert, indem jede Schicht die Fahrbefehle an die nächste untere weitergibt. In der unteren Schicht wird diese Aktion dann evtl. ausgeführt, weitergegeben oder unterdrückt. Es gibt also keine nachgeschaltete Entscheidungseinheit, die abwägt, welche Aktion gerade Vorrang hat.

Diese Implementation des Schichtmodells von konkurrierenden Verhalten ermöglicht eine hierarchische Erweiterung des Gesamtsystems durch Hinzufügen höherer Ebenen, ohne dass alle unteren Schichten angepasst werden müssen.

Kapitel 4

Bildvorverarbeitung

In diesem Kapitel erläutern wir die Schritte der Bildverarbeitung, die zu den von uns genutzten Bildprimitva führen (unendlich ausgedehnte Hough-Geraden [24]). Diese Geraden werden zur Poseschätzung benutzt.

Die Bildvorverarbeitung beinhaltet die Extraktion von Bildprimitiva, mit dem Ziel, die zu verarbeitende Gesamtinformation des Bildes zu reduzieren.
Bildprimitiva können z.B. sein:

- Markante Punkte (Ecken und Kreuzungen),
- Kanten (Bereiche mit hohem Grauwertgradientenbetrag),
- Liniensegmente mit Orientierung und Länge,
- Gaborwaveletantworten oder Frequenzkoeffizienten bei der Fouriertransformation.

In dieser Arbeit beschränken wir uns auf die Extraktion von Grauwertkanten, die als unendlich ausgedehnte Geraden beschrieben werden. Die extrahierte Bildgerade wird zur Rekonstruktion der projektiven Ebenen herangezogen. Diese Ebenen sind unendlich ausgedehnt (siehe Kapitel 2).

Zur Erkennung und Bestimmung der Bildgeraden wird eine Erweiterung der klassischen Houghtransformation verwendet, die gewichtete, orientierungsselektive Houghtransformation [1]. Dieses globale Verfahren ist sehr robust, und im Gegensatz zu lokalen Verfahren weniger sensitiv gegenüber Rauschen und Okklusion. Eine Eigenschaft dieses globalen Verfahrens ist der Verlust der lokalen Information, da die gefundenen Geraden unendlich ausgedehnt sind. Die Anfangs- und Endpunkte der Kanten stehen also nicht zur Verfügung. Dieser Informationsverlust ist jedoch unerheblich, weil zur Bestimmung der Pose unendlich ausgedehnte Ebenen benutzt werden.
Die Bildvorverarbeitung unterteilt sich in folgende Schritte:

i. Bildglättung mit 3×3 Binomialfilter,

ii. Berechnung des Gradienten bei Unterdrückung von Nebenmaxima,

iii. Erzeugung des Akkumulatorfelds mittels der gewichteten, orientierungsselektiven Houghtransformation,

iv. Maximumsuche im Akkumulatorfeld.

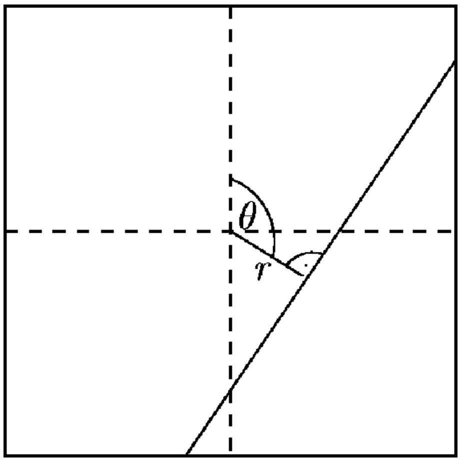

Abbildung 4.1: Darstellung der Geraden in (r, θ)-Polarform

4.1 Kantenextraktion

Zunächst wird das Bild mit einem Tiefpassfilter (3×3 Binomialfilter) geglättet, mit dem Ziel das Bildrauschen zu verringern. Danach erfolgt die Berechnung des Grauwertgradienten für jeden Pixel des Bildes mit Speicherung des Gradientenbetrags und Orientierung.

Um nun ein binäres Kantenbild zu erhalten, wird eine Schwellwertsegmentierung verwendet. Dazu werden Pixel mit einem Gradientenbetrag oberhalb eines Schwellwertes ausgewählt. Grauwertkanten weisen jedoch in der Regel einen glatten und keinen sprunghaften Grauwertanstieg auf, d.h. eine einfache Schwellwertbildung auf dem Gradientenbild liefert keine scharfen, sondern in der Breite leicht ausgedehnte Grauwertkanten. Ist eine Kante im Binärbild beispielsweise 3 Pixel breit, so liefert diese Kante nach der Houghtransformation 3 Maxima (1 Haupt- und 2 Nebenmaxima), die im Akkumulatorfeld sehr dicht beieinander liegen, sodass eine exakte Lokalisierung der Kante nicht möglich ist. Um diese Nebenmaxima zu unterdrücken werden zur Houghtransformation nur solche Punkte des Binärbildes herangezogen, die in Richtung des ansteigenden Gradienten den höchsten Betrag haben.

4.2 Houghtransformation

Die Houghtransformation gehört zu den sogenannten *Votierungsverfahren* und dient der Erkennug von Strukturen in Binärbildern, die durch eine explizite Funktion beschreibbar sind. Für Geraden ist eine geeignete Funktion $\mathbf{f}_g$ die Beschreibung in Polarform,

$$\mathbf{f}_g(r, \theta) = x \cos\theta + y \sin\theta - r = 0. \tag{2.1}$$

Die Polarform stellt eine Gerade durch den kürzesten Abstand r zum Ursprung (Bildmitte) und den Winkel θ der Normalen zur y-Achse dar (siehe Abbildung 4.1).
Die Houghtransformation arbeitet folgendermaßen:
Jeder Punkt $p = (x, y)$ im Binärbild votiert für alle Parametersätze (r, θ), die ihn erzeugt haben können, d.h. für alle Geraden, die durch den Punkt p verlaufen. Für alle

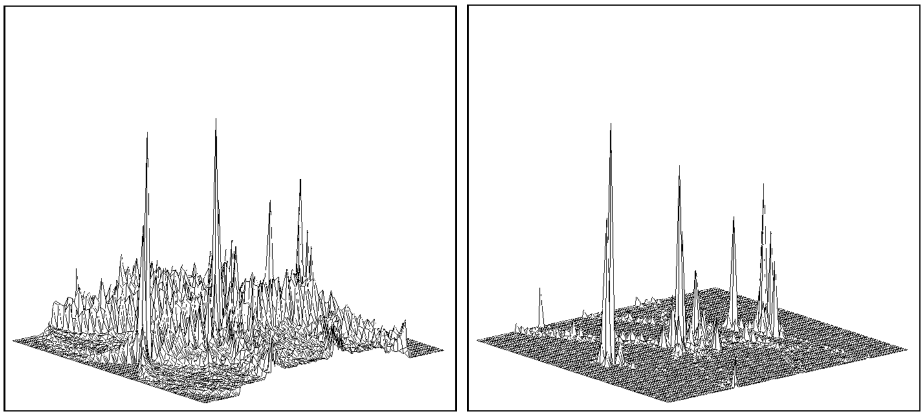

Abbildung 4.2: Akkumulatorfeld der Standard-Houghtransformation (links) und der gewichteten, orientierungsselektiven Houghtransformation(rechts)

Punkte des Binärbilds wird eine Kurve in einem Akkumulatorfeld gezeichnet (die Werte werden erhöht), Maxima in diesem Feld sind Kandidaten für im Bild vorhandene Geraden. Je mehr Punkte zu einer Grauwertkante gehören, desto stärker ist das Maximum im Akkumulatorfeld.

4.3 Erweiterte Houghtransformation

Die orientierungsselektive Houghtransformation ist eine Erweiterung der normalen Houghtransformation und benutzt zusätzlich zum Betrag des Gradienten auch dessen Orientierung. Durch diese Erweiterung votieren nur Punkte $p = (x, y)$ im Bild für eine Gerade beschrieben durch (r, θ), wenn der zum Punkt p gehörende Gradientenwinkel β_p in einem Toleranzbereich um θ liegt. Durch diese Modifikation nimmt das Rauschen im Akkumulatorfeld deutlich ab, sodass signifikante Maxima leichter zu bestimmen sind (siehe Abbildung 4.2).

Die gewichtete Houghtransformation unterscheidet sich zur normalen Houghtransformation, die Werte im Akkumulatorfeld jeweils um 1 erhöht, durch eine variable Erhöhung der Akkumulatorwerte, d.h. ein geeigneter Bildpunkt votiert für eine Gerade mit den Parametern (r, θ) abhängig von der Qualität seiner Eignung, sodass die Werte im Akkumulatorfeld nicht grundsätzlich um 1, sondern um Werte aus dem Intervall [0..1] erhöht werden. Diese Gewichtung des Beitrags eines geeigneten Punktes p wird beeinflusst durch den normalisierten Betrag des Gradienten des Punktes p und durch dessen Gradientenwinkel β_p. Durch die Erweiterung der Houghtransformation zur gewichteten, orientierungsselektiven Houghtransformation wird das Rauschen im Akkumulatorfeld stark verringert, ohne dass sich die Position der lokalen Maxima verändert. Eine ausführlichere Beschreibung und Analyse der gewichteten, orientierungsselektiven Houghtransformation ist zu finden in [1].

Das Hauptproblem sind sogenannte Geradenbüschel, die durch Nebenmaxima verursacht werden. Diese Nebenmaxima treten dabei charakteristisch im Akkumulatorfeld als sog. *Schmetterlingsstruktur* auf (siehe Abbildung 4.3).

Abbildung 4.3: Charakteristische Schmetterlingsstruktur der Maxima im Akkumulatorfeld

4.4 Maximumsuche im Akkumulatorfeld

Weiteren Einfluss auf die Qualität und Quantität der gefundenen, signifikanten Geraden hat die Maximumsuche im Akkumulatorfeld. Zu unterscheiden sind dabei zwei Ansätze, einerseits die reine Suche der lokalen Maxima im Akkumulatorfeld, andererseits Verfahren, die zusätzlich zur Suche im Akkumulatorfeld auf Informationen im Originalbild zurückgreifen. Wir haben Versuche mit zwei unterschiedlichen Suchverfahren gemacht.

4.4.1 Maximumsuche mit Bereichslöschung

Das Verfahren mit Bereichslöschung basiert auf dem bereits von Pauli [37] vorgeschlagenen Ansatz, einen rechteckigen Bereich um gefundene lokale Hauptmaxima zu löschen. Dieser Ansatz ist durch Ackermann [1] so verändert worden, dass ein elliptischer Bereich um lokale Maxima gelöscht wird, der den Vorteil bietet nur benachbarte Geraden auszuschliessen, die den gleichen Abstand bezüglich (r, θ) zum lokalen Maxima im Akkumulatorfeld haben. Weitere Verfahren zur Suche von lokalen Maxima im Akkumulatorfeld finden sich in [36, 29].

Sehr kritisch ist bei diesem Suchverfahren die Wahl der Größe des zu löschenden Bereichs. Bei zu groß gewähltem Bereich löscht man vielleicht dicht nebeneinanderstehende, parallele Geraden, die tatsächlich im Bild vorhanden sind. Wird der Bereich dagegen zu klein gewählt, erhält man mehrere Antworten an nur einer Kante in Form eines Geradenbüschels.

4.4.2 Maximumsuche mit Verifikation im Originalbild

Das Verfahren mit Verifikation im Originalbild, im folgenden Distanzverfahren genannt, prüft, ob zwei lokale Maxima im Akkumulatorfeld die gleiche reale Kante im Bild beschreiben oder ob beide Maxima von verschiedenen Kanten erzeugt wurden. Sind beide Maxima von der gleichen Kante erzeugt, wird das schwächere Maximum gelöscht. Abbildung 4.4 verdeutlicht die zu unterscheidenen Situationen. Um diese Fälle unterscheiden zu können, wird im Originalbild nach starken Bereichen der Hough-Geraden gesucht (schwarz markierte Bereiche in Abbildung 4.4). Stark ist ein Bereich in der Nähe einer Geraden dann, wenn das Produkt $c_g(\mathbf{p})$ aus dem normalisierten Gradientenbetrag $E_\mathbf{p}$ und der Abweichung des Gradientenwinkels θ_p vom Geradenwinkel θ hoch ist,

$$c_g(\mathbf{p}) = E_\mathbf{p} \cdot f_\theta(\|\theta - \theta_p\|). \tag{4.2}$$

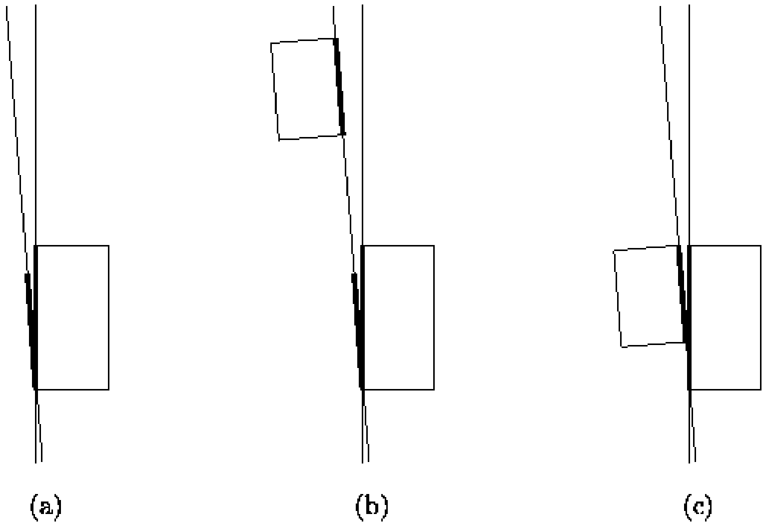

Abbildung 4.4: Durch Verifikation im Originalbild zu unterscheidene Fälle

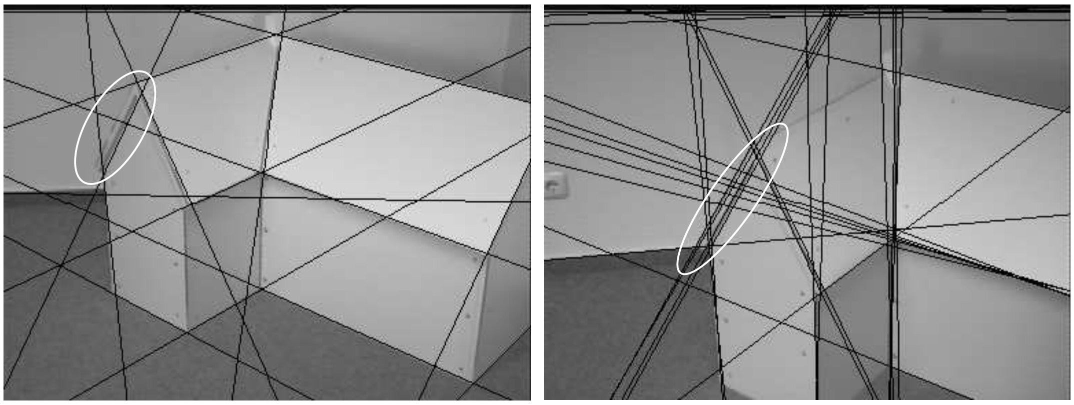

Abbildung 4.5: Ergebnis der Maximumsuche: links das Verfahren mit Bereichslöschung und rechts das Distanzverfahren

Dieses Produkt wird im Folgenden auch als Konfidenz bezeichnet. Ausgehend vom Schnittpunkt zweier Geraden wird nun in äquidistanten Schritten die Konfidenz berechnet. Die Summe dieser Konfidenzen ist ein Maß für die Signifikanz beider Geraden. Ist diese Summe größer als ein Schwellschwert t_{conf}, so bleibt die Gerade erhalten, ist sie kleiner, so ist die Gerade von der gleichen Kante erzeugt worden wie ihr Nachbar, sodass sie gelöscht werden kann. Neben dem Schwellwert t_{conf}, der bei zu kleiner Wahl Geradenbüschel erzeugt, ist ein weiterer Parameter t_{length} für dieses Verfahren wichtig. Der Schwellwert t_{length} bestimmt, wie stark Maxima im Akkumulatorfeld mindestens sein müssen, um zur Maximumsuche herangezogen zu werden. Also bestimmt dieser Wert die minimale Länge einer Kante im Bild.

Eine detailiertere Beschreibung des Distanzverfahrens und die Bedeutung der Parameter ist in [1] zu finden.

4.4.3 Vergleich der Verfahren

Die Vor- und Nachteile beider Verfahren sind in Tabelle 4.1 dargestellt.

Beim Vergleich beider Verfahren hat sich gezeigt, dass die Wahl der Parameter beim Verfahren mit Bereichslöschung im Akkumulatorfeld sehr kritisch ist. Abhängig von der aufgenommen Szene und dem Abstand der wichtigen Objekte, liefert ein fester Parametersatz teilweise unbrauchbare Ergebnisse, da signifikante Geraden, welche für die weitere Verarbeitung entscheidend sind, oft gar nicht gefunden werden oder es werden deutlich zu viele Geraden in Form von Geradenbüscheln detektiert, sodass die weitere Verarbeitung zeitlich zu intensiv wird oder es zu viele mögliche Lösungen gibt. Wird der zu löschende Bereich jedoch optimal angegeben, so liefert dieses Verfahren sehr gute Ergebnisse. In Abbildung 4.5 (links) sieht man das Ergebnis des Verfahrens mit Bereichslöschung. Der zu löschende Bereich war dabei nahezu optimal gewählt, zu erkennen daran, dass kaum Geradenbüschel vorhanden sind, trotzdem aber alle wichtigen Kanten detektiert wurden.

Ein großer Nachteil dieses Verfahren ist, dass dicht benachbarte, parallele Kanten nicht gefunden werden können, selbst wenn der Bereich optimal angegeben ist. So wurden z.B. die Dachkanten nicht korrekt detektiert, siehe Markierung in Abbildung 4.5 (links).

Das Distanzverfahren liefert im Allgemeinen zwar keine besseren Ergebnisse, jedoch ist die Qualität der Ergebnisse deutlich robuster gegenüber Veränderungen der Szene bei gleichem Parametersatz ($t_{length} = 20$,$t_{conf} = 60$). Diese Werte ergaben sich nach ausführlichen Tests mit unterschiedlichen Parametersätzen. Eine Veränderung von t_{conf} zeigt kaum Auswirkungen. Erhöht man dagegen t_{length}, so erhählt man insgesamt weniger Geraden, weil der Wert t_{length} die minimale Höhe eines Maximums im Akkumulatorfeld angibt, was der minimalen Länge einer Kante entspricht. Mit hohen Werten werden zu wenig Kanten detektiert, ein niedriger Wert führt zu sehr vielen Kanten, die die weitere Verarbeitung stark verlangsamen.

Geradenbüschel werden beim Distanzverfahren ebenfalls erfolgreich unterdrückt, wobei in Bildregionen mit dicht beieinander auftretenden, parallelen Kanten sehr viele Geraden detektiert werden, für die evtl. keine Kanten im Bild zu finden sind. Diese falsch detektierten Geraden erhöhen den Suchraum, ermöglichen aber die Bestimmung der Pose mit einem deutlich kleineren Gesamtfehler, da nur mit den korrekt detektierten Geraden die genaue Anpassung der Abbildungsparameter möglich wird (siehe folgendes Kapitel).

Für die weitere Verarbeitung der Geraden in unserer Arbeit ist die Eigenschaft sehr wichtig, dass auch dicht beieinander liegende parallele Geraden gefunden werden. So wurden z.B. in Abbildung 4.5 (rechts) alle Kanten des überstehenden Daches und der zugehörige Schatten detektiert, siehe hell markierter Bereich in Abbildung 4.5 (rechts).

Die Rechenzeit des Distanzverfahrens ist zwar etwas größer als beim Verfahren mit Bereichslöschung, da aber die vollständige Vorverarbeitung der Bilder (Größe: 384×288 Pixel) auf einem Pentium3 800Mhz System weniger als 500ms benötigt, ist die unterschiedliche Rechenzeit der Verfahren in unserer Arbeit irrelevant.

	Vorteile	Nachteile
Maximumsuche mit Bereichslöschung	- einfacher Algorithmus - geringe Rechenzeit - vorwiegend signifkante Geraden bei optimalen Parametern	- parallele benachbarte Kanten nicht detektierbar - sensibel für Parameteränderung - Geradenbüschel oder wenig Geraden bei schlecht gewählten Parametern
Distanzverfahren	- parallele benachbarte Kanten werden detektiert - gute Ergebnisse mit Standardparametern	- komplexer Algorithmus - Originalbild benötigt - viele nicht signifkante Geraden an parallelen benachbarten Kanten

Tabelle 4.1: Vergleich der beiden Verfahren zur Maximumsuche

Kapitel 5

Tracking

Als Tracking bezeichnet man die Verfolgung eines Objektes in einer Bildfolge. Verfolgung bedeutet die Bewegung des Objektes von einem Bild zum nächsten zu bestimmen. Die Bewegungsbestimmung kann im Bild erfolgen (2D-2D), wie z.B. bei Verfahren die den optischen Fluss verwenden [5] oder es wird die 3D-Bewegung eines 3D-Objektes gesucht (3D-3D).

Wir verstehen unter Tracking die Bestimmung der 3D-Bewegung eines Objektes innerhalb einer Szene durch Anpassung des projizierten Objektmodells an 2D-Bilddaten (2D-3D). Dabei ist die Bewegung des Objekts zwischen den Bildaufnahmen sehr klein, sodass man annehmen kann, dass sich das Objekt auch im Bild nur wenig bewegt, die Objektpunkte sich also in der gleichen Umgebung wiederfinden lassen.

Um die Bewegung des Objekts zu ermitteln, ist es notwendig Beziehungen zwischen den extrahierten Bildmerkmalen und den Modellelementen herzustellen. Diesen Vorgang bezeichnen wir als Korrespondenzfindung, auf den wir im Folgenden noch gesondert eingehen. Wird eine Korrespondenz angenommen, kann diese benutzt werden, um die Pose des Objekts zu bestimmen. Das Fehlermaß der Pose ermöglicht dann eine Bewertung der Korrespondenz. Es gibt also eine Wechselwirkung zwischen Korrespondenzfindung und Poseschätzung. Abbildung 5.1 verdeutlicht die Zusammenhänge der Teilprobleme.

In diesem Kapitel motivieren wir zunächst die Verwendung der Trackingannahme in unserer Arbeit. Darauf folgt die Beschreibung des Matchingproblems, das einerseits unser Verfahren zum Finden geeigneter Korrespondenzen erläutert und andererseits zeigt, wie

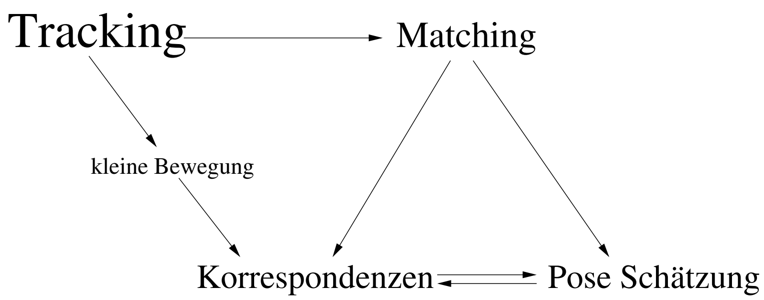

Abbildung 5.1:

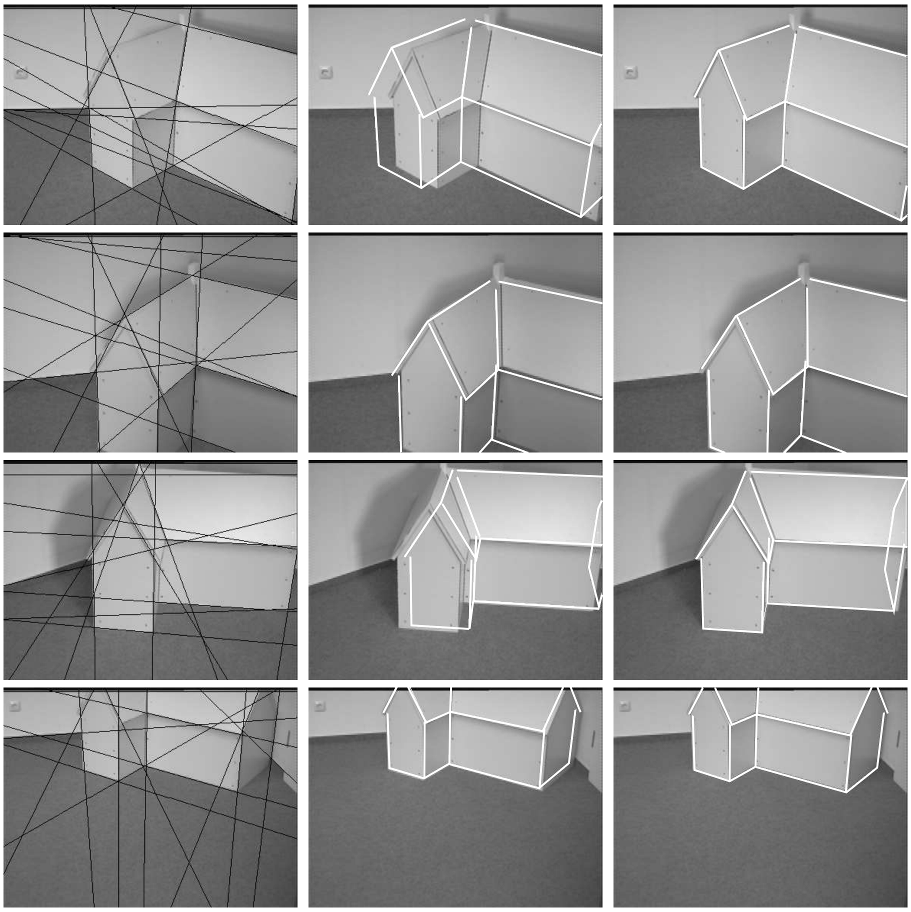

Abbildung 5.2: Bildvorverarbeitung (links), Vorhersage des Kamerbildes (mitte) und Visualisierung der gefundenen Bewegung (rechts). Jede Zeile zeigt die Aufnahmen an den Eckpunkten eines rechteckförmigen Pfades.

die Pose des Objekts unter Verwendung verschiedener Bedingungen bestimmt werden kann.

5.1 Trackingannahme

Wir verwenden die Eigenschaft von Bildfolgen, dass die Bewegung des Objektes von einem Bild zum nächsten nur sehr klein ist, wenn die Bildaufnahmen in genügend kurzen Zeitintervallen erfolgen und die relative Geschwindigkeit des Objektes zur Kamera ausreichend langsam ist. Sind diese Bedingungen gegeben, so kann man annehmen, dass sich das Objekt innerhalb des Zeitintervals zwischen zwei Bildaufnahmen nur wenig bewegt. Also lassen sich die im Bild extrahierbaren Merkmale des Objektes im nächsten Bild an ungefähr der gleichen Position und mit der gleichen Orientierung wiederfinden.

Werden in kurzen zeitlichen Abständen Bilder während der Fahrt aufgenommen, so ist die Trackingannahme erfüllt. Aufnahmen, die während der Fahrt des Roboters aufgenommen werden, sind jedoch nicht geeignet für die weitere Verarbeitung, weil der

Roboter im Allgemeinen nicht in Richtung der Landmarke blickt und es zu Verzerrungen der Szene kommt (*motion blur*). Deshalb ist es notwendig die Landmarke zunächst ins Blickfeld zu bekommen.

Wir benutzen in unserer Arbeit keine in kurzen Zeitabständen aufgenommene Bildfolge, sondern die Odometriedaten eines Roboters zur Vorhersage des Kamerabildes. Da die Odometriedaten verrauscht sind, ergibt sich i.A. eine kleine Abweichung des Originalbildes vom vorhergesagten Bild. Dieser Fehler entspricht einer Bewegung des Objektes, die äquivalent ist zu der Standard-Tracking Situation.

Abbildung 5.2 verdeutlicht die Situation. Die Bilder sind jeweils an den Eckpunkten eines Rechtecks aufgenommen, an dem Roboter entlang gefahren ist. Jede Zeile entspricht einer Position. Die linke Spalte zeigt das Ergebnis der Bildvorverarbeitung, in der mittleren Spalte ist das Originalbild (Hintergrund) mit dem vorhergesagten Bild des Objektmodells (hell überlagert) sichtbar und die rechte Spalte zeigt das projizierte Modell nach dem Matching. Die Bewegung, welche durch Rauschen der Odometriedaten entstanden ist, ist im mittleren Bild als Abweichung sichtbar. Das die Pose und damit die Bewegung erfolgreich bestimmt werden konnte ist in den rechten Bildern sichtbar, in denen das Modell mit der berechneten Bewegung transformiert und projiziert wurde.

Der Algorithmus zur Korrespondenzfindung basiert auf dieser Trackingannahme und lässt sich unverändert auf das Tracking von Objekten innerhalb einer Bildfolge anwenden. Da wir dieses Verfahren im Rahmen landmarkenbasierter Navigation verwenden, bewegt sich das Objekt/die Landmarke innerhalb der Szene nicht, sondern die auf dem Roboter angebrachte Kamera erzeugt die Veränderung des Bildes, wenn sich der Roboter bewegt.

5.2 Ausnutzung der Odometriedaten

Um mit dem Aufnahmesystem des Roboters Bilder von ausreichender Qualität zu erhalten, ist es notwendig, dass sich die Kamera während der Bildaufnahme im Ruhezustand befindet und die Landmarke sich auch tatsächlich im Blickfeld befindet. Um dies zu erreichen und gleichzeitig die Trackingannahme zu gewährleisten, nehmen wir häufig genug Bilder auf, damit die Unsicherheit der Roboterposition nicht zu hoch wird (typischer zeitlicher Abstand 30 Sekunden).

Die vom Roboter in dieser Zeit durchgeführte Bewegung ist ungefähr bekannt, da die Umdrehungen der Rollen und die Rotation des Rumpfes akkumuliert werden und so die aktuelle Position ungefähr bestimmt werden kann (Odometrie). Durch Schlupf und Ungenauigkeiten in der Messeinheit sind diese Daten jedoch verrauscht. Die verrauschten Odometriedaten werden verwendet, um die Position und Orientierung des Modells der Landmarke im Bild vorherzusagen. Da diese Vorhersage aufgrund der ungenauen Odometriedaten ebenfalls verrauscht ist, ergibt sich eine Positions- und Orientierungsdifferenz des projizierten Modells mit der tatsächlich aufgenommenen Landmarke. So erhält man eine vergleichbare Situation wie im Falle des Trackings von Objekten.

Im Folgenden wird beschrieben, wie die Vorhersage des Bildes erreicht wird. Abbildung 5.3 zeigt die Situation nach einer Bewegung $\mathbf{t_r}$, die aus den Odometriedaten ermittelt wird. Die beiden Kreise deuten den Rumpf des Roboters in der Aufsicht an. Rechts ist die Startposition des Roboters zu sehen, die das Roboterkoordinatensystem

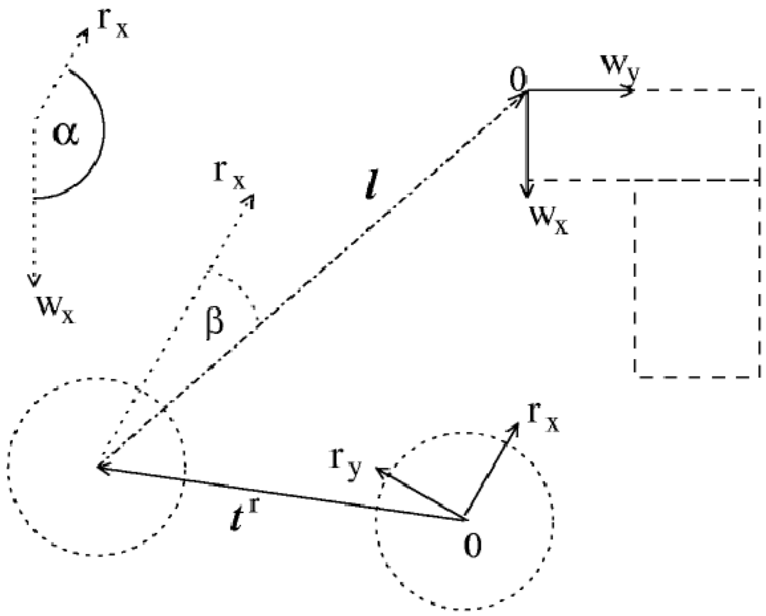

Abbildung 5.3: Verwendung der Odometriedaten zur Vorhersage des Kamerbildes

definiert, und links unten ist die Position nach der Bewegung $\mathbf{t_r}$ sichtbar. Oben rechts ist die Landmarke im Weltkoordinatensystem angedeutet. Die Transformation zwischen Welt- und Roboterkoordinatensystem, gegeben durch den Winkel α und den Vektor $\mathbf{l}$, ist als bekannt vorausgesetzt (siehe folgender Abschnitt: Bestimmung der Startwerte).

5.2.1 Ausrichtung zur Landmarke

Da der Roboter einem beliebigen Pfad folgen kann, ist die Landmarke im allgemeinen nicht im Blickfeld des Roboters. Deshalb muss der Roboter zunächst geeignet rotieren, um die Landmarke gut sichtbar im Blickfeld zu haben. Dabei wird die Position der Landmarke im Roboterkoordinatensystem benutzt, in Abbildung 5.3 durch $\mathbf{l} = (l_x, l_y)$ gekennzeichnet. Der Rotationswinkel berechnet sich dabei wie folgt,

$$tan\beta = \frac{l_y}{l_x}. \tag{2.1}$$

Da der arctan nur von $-\pi/2$ bis $\pi/2$ definiert ist, benutzen wir die atan2 Funktion [25], wobei

$$atan2(y,x) := \begin{cases} atan(\frac{y}{x}) & : \quad x > 0 \\ sign(y)(\pi - atan(\|\frac{y}{x}\|)) & : \quad x < 0 \\ sign(y)\frac{\pi}{2} & : \quad y \neq 0, x = 0 \\ undef & : \quad x = y = 0 \end{cases} \tag{2.2}$$

5.2.2 Anpassung der Odometriedaten an die Projektionsmatrix

Für die Vorhersage des Kamerabildes muss die vom Roboter durchgeführte Bewegung an die Projektionsmatrix angepasst werden. So kann das Modell der Landmarke in das Bild projiziert werden. Fehler in der Bestimmung der Bewegung verursachen eine Verschiebung des projizierten Modells im Bild, sodass ein Matching von Modell und Bildkanten erschwert oder sogar unmöglich wird. Eine Modellierung des zu erwartenden Fehlers der Odometrie und den Einfluss dieses Fehlers auf Verschiebungen im Bild findet man in [27].

Die Bewegung des Roboters beinhaltet eine Translation $\mathbf{t^r}$ im Roboterkoordinatensystem und eine Rotation R^r_β um die z-Achse des Roboterkoordinatensystems (r_x,r_y), siehe Abbildung 5.3. Im folgenden ist T^w das Äquivalent zur Matrixmultiplikation des Vektors $\mathbf{t^r}$.

Gesucht ist die äquivalente allgemeine Transformation $A^w = T^w R^w_\beta$ in Weltkoordinaten (w_x, w_y), welche die Projektionsmatrix P in eine Matrix P' überführt, sodass die abgebildeten Modellpunkte mit dem tatsächlichen Kamerabild übereinstimmen,

$$P' = PA^w = PT^w R^w_\beta. \tag{2.3}$$

Diese Multiplikation entspricht einer Transformation der Modellpunkte vor der eigentlichen Projektion. Sei X ein Modellpunkt in Weltkoordinaten. Seine Projektion x im Bild ist gegeben durch $x = PX$. Also gilt für P',

$$x' = P'X = PA^w X = PT^w R^w_\beta X. \tag{2.4}$$

D.h. die gesuchte Transformation wirkt auf die Modellpunkte, nicht auf die Kamera. Bewegt sich die Kamera um TR, so entspricht dies einer Transformation $(TR)^{-1}$ der Modellpunkte. Deshalb gilt

$$T^w = R_\alpha (T^r)^{-1}. \tag{2.5}$$

Die von der Kamera durchgeführte Rotation R^r_β muss von den Modellpunkten an der gleichen Position in die andere Richtung durchgeführt werden. Deshalb werden die Modellpunkte zunächst um $\mathbf{l}^w$ verschoben, rotiert und zurückgeschoben,

$$R^w_\beta = (R_\alpha L^r)^{-1} R^{-1}_\beta (R_\alpha L^r). \tag{2.6}$$

Insgesamt ergibt sich für die Projektionsmatrix P',

$$P' = PR_\alpha (T^r)^{-1} (R_\alpha L^r)^{-1} R^{-1}_\beta (R_\alpha L^r). \tag{2.7}$$

5.2.3 Bestimmung der Startwerte

Die Transformation T^w_r beschreibt die Beziehung zwischen Roboter- und Weltkoordinaten, welche äquivalent ist zur Problemstellung der *Hand-Auge-Kalibrierung*. Da die Kameras auf Modellpunkte im Weltkoordinatensystem kalibriert sind, die Bewegung des Roboters aber nur im Roboterkoordinatensystem bekannt ist, muss diese Bewegung mit T^w_r transformiert werden.

Der Rotationswinkel γ^w_r der Matrix T^w_r und die Translation $\mathbf{t}^w_r$ in der Ebene können aus der initialen Projektionsmatrix bestimmt werden. Dazu ist es lediglich notwendig den Roboter an etwa jene gleiche Position zu fahren, an welcher die Projektionsmatrizen (linke und rechte Kamera) bestimmt wurden. Die Abweichung wird durch den Matching-Algorithmus (siehe Kapitel 5) berechnet und die Projektionsmatrix angepasst. Es hat sich gezeigt, dass die explizite Bestimmung der *externen* Kameraparameter sehr ungenau ist, und sich hohe Abweichungen bei der Translation $\mathbf{t}_r^w$ ergeben[56]. Der Winkel γ_r^w wird jedoch mit ausreichend hoher Genauigkeit geschätzt. Deshalb geben wir nur die Translation $\mathbf{t}_r^w$ zwischen Roboter- und Weltkoordinatensystem manuell an.

5.3 Matching

Dieser Abschnitt erläutert das Matchingproblem und beschreibt den von uns verwendeten Algorithmus, der das Korrespondenzproblem löst und die tatsächliche Pose der Landmarke findet.

Matching ist der Vorgang, der die extrahierten Bildmerkmale (in unserem Fall unendlich ausgedehnte Geraden) auf das Modell der Landmarke abbildet und die Bestimmung der Objektpose ermöglicht. Dabei sind zwei Teilprobleme zu lösen:

- *Das Korrespondenzproblem* beschreibt die Zuordnung von Modellelementen (3D-Punkte/Geraden) und Bildelementen (2D-Geraden).

- *Die Anpassung* des Modells, d.h. die Abbildungsparameter zu finden (hier Rotation und Translation), sodass der räumliche Abstand von Modell- und Bildelementen möglichst klein wird.

Diese Teilprobleme definieren eine Optimierungsaufgabe mit dem Ziel einen Gesamtfehler (*match error*) zu minimieren. Dieser Fehler muss jedoch beide Teilprobleme berücksichtigen.

5.3.1 Der Suchraum aller möglichen Korrespondenzen

Sei M die Menge aller Modelllinien und sei B die Menge der extrahierten Bildgeraden. Der Raum aller möglichen Korrespondenzen ist dann $S = M \times B$. Jede Teilmenge $S' \subseteq S = M \times B$ stellt eine Korrespondenz dar, die eine Pose des Modells definiert. Zur Lösung des Optimierungsproblems ist die Menge S' zu finden, die den *match error* minimiert.

Zum Beispiel hat das Modell einer Tür mit Rahmen 9 Linien. Wie in Abbildung 5.4 sichtbar, liefert die Bildvorverarbeitung ca. 20 Geraden. Links sieht man die projizierten Modelllinien (8 sichtbar) und rechts die extrahierten Bildgeraden. Der gesamte Suchraum ist die Menge aller Teilmengen (Potenzmenge) mit der Größe $2^{\|M \times B\|} = 2^{9\times 20} = 1,5 \cdot 10^{54}$.

Selbst wenn die Berechnung einer Pose zu einer Teilmenge S' und dessen *match error* nur eine μs dauerte, bräuchte die Berechnung aller Teilmengen und der zugehörigen Korrespondenz/Pose mehrere Jahre !

Notwendig ist also eine Einschränkung des Suchraums und eine Suchstrategie, die nicht alle Möglichkeiten durchprobiert. Nimmt man an, dass eine Modelllinie jeweils mit nur einer Bildgerade korrespondieren kann, so reduziert sich die Größe des Suchraums erheblich. Da wir Bildgeraden mit Hilfe der Houghtransformation gewinnen, sind diese unendlich ausgedehnt. Sind von einer Modellkante nur Teilbereiche im Bild extrahiert bzw. sichtbar, z.B. wegen Okklusion, so werden beide extrahierten Liniensegmente durch

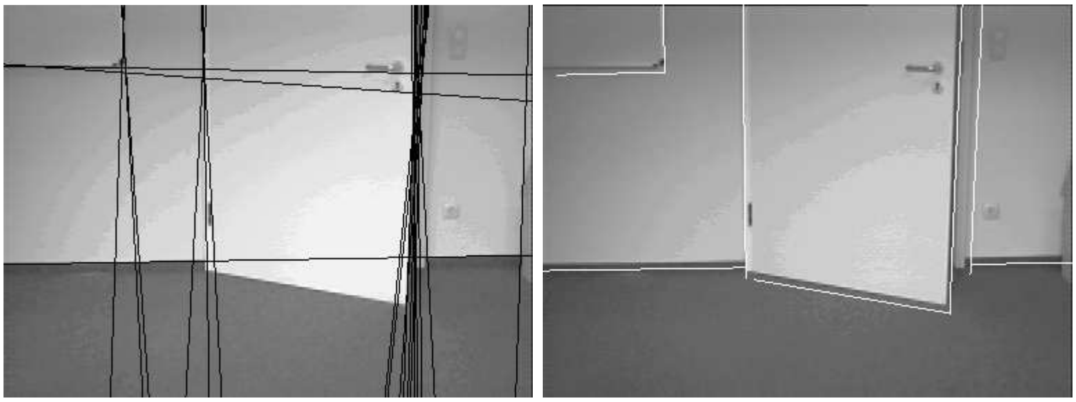

Abbildung 5.4: Kamerabild mit extrahierten Hough-Geraden (links) und Projektion des Modells vor dem Matching (rechts)

die Houghtransformation zu einer Linie gruppiert. Deshalb ist die Annahme in unserem Fall sinnvoll, dass eine Modelllinie jeweils mit nur einer Bildgerade korrespondieren kann.

Der andere Fall gilt jedoch nicht. Eine Bildgerade kann ohne weiteres mit mehreren Modelllinien korrespondieren. Dieser Fall tritt z.B. immer dann auf, wenn Modelllinien parallel in Folge definiert werden (linear abhängige Richtungsvektoren), wie bei der Teppichleiste in Abbildung 5.4. Selbst wenn man bei der Definition des Modells solche linear abhängigen Linien vermeidet, können Modelllinien, die in einer Ebene liegen, bei geeigneter Rotation des Modells nach der Projektion auf der gleichen Bildgeraden liegen, wie in Abbildung 5.5 dargestellt. Der Suchraum wird in einer Matrix kodiert, deren Einträge e_{ij} angeben, ob eine Korrespondenz zwischen Modelllinie i und Bildgerade j möglich ist.

Suchbereich und Trackingannahme

Wie im vorigen Abschnitt erläutert, nutzen wir die Trackingannahme, da die Eigenbewegung des Roboters ungefähr bekannt ist und die Projektion des Modells von einem Bild zum nächsten Bild grob geschätzt werden kann. Damit ist das projizierte Modell im Bild nur wenig verschoben vom tatsächlich aufgenommenen Modell.

Unter dieser Annahme kann der Suchbereich folgendermaßen schrittweise eingeschränkt werden.

- Sind im Bild Modelllinien nur zu einem bestimmten Teil oder gar nicht sichtbar, so kann zu ihnen auch keine Korrespondenz gefunden werden.
- Die Bildgeraden liegen in der Nähe der projizierten Modelllinien, d.h. ihr Abstand in Pixeln sowie der Winkel zwischen beiden Linien ist klein. Korrepondenzen mit einem Abstand oder einer Winkeldifferenz über einem bestimmten Grenzwert sind nicht möglich. Dies bewirkt eine sehr starke Einschränkung des Suchbereichs und wird auch in [6, 44] verwendet.
- Falls bekannt ist, dass die Bewegung des Modells in der Ebene stattgefunden hat, wie im Falle einer Roboterbewegung, und die Kamera nicht um ihre optische Achse rotiert, so kann für vertikale Modelllinien eine weitere Einschränkung getroffen

werden. Vertikal bedeutet, dass die Modelllinie orthogonal zum Boden ist. Bei einer Translation und Rotation in der Ebene ändert sich die Orientierung der Linie auch in der Projektion praktisch nicht, sodass für vertikale Modelllinien nur Korrespondenzen mit der nahezu gleichen Orientierung (Winkel in Hough-Koordinaten) möglich sind.

Die Wahl des Grenzwertes bei der Einschränkung über Winkeldifferenz und Abstand ist sehr kritisch in Bezug auf die Gesamtrechenzeit. Wird die erlaubte Abweichung sehr klein gewählt, so verringert sich die Anzahl möglicher Korrespondenzen zwar extrem, es besteht jedoch die Gefahr, dass die korrekte Korrespondenz nicht mehr gefunden wird und die Pose des Modells nicht berechnet werden kann, wenn die Abweichung des projizierten Modells vom realen Bild größer als angenommen ist.
Wird der Grenzwert dagegen sehr groß gewählt, können auch größere Modellbewegungen (Odometriefehler) erkannt werden. Da sich der Suchraum entsprechend vergrößert, wird mehr Rechenzeit benötigt bis ein geeignetes Minimum gefunden ist. Gleichzeitig erhöht sich auch die Wahrscheinlichkeit von falschen Posen, weil die Anzahl lokaler Minima mit gültiger Korrespondenzanzahl steigt oder das globale Minimum eine *falsche* Pose repräsentiert.

5.3.2 Anpassung der Abbildungsparameter

Im Allgemeinen sind alle Abbildungsparameter der Kamera, d.h. 3D-Rotation, Translation, Skalierung, Brennweite etc. (siehe Kapitel Math. Grundlagen/Kamerkalibrierung), so zu bestimmen, dass der räumliche Abstand von projizierten Modelllinien und extrahierten Bildgeraden minimal wird. Diese Parameter bestimmen gleichzeitig die Pose des Modells.

Wir bestimmen nicht alle Abbildungsparameter, sondern lediglich die Rotation und Translation des Modells. Die internen Kameraparameter (Brennweite, Skalierung, Mittelpunkt) werden als bekannt und konstant angenommen, da die verwendeten Kameras weder Zoom noch Autofocus besitzen. Die internen Kameraparameter werden von uns nicht explizit berechnet, sondern sie sind in der initial angegeben Projektionsmatrix enthalten. Die explizite Berechnung der internen Parameter hat sich als sehr rauschanfällig erwiesen und sollte deshalb vermieden werden [41, 56]. Im weiteren ändern wir nur die *externen* Parameter der Projektionsmatrix durch Rotation und Translation.

Fehlermaß

Der oben angesprochene *match error* (err_m), der vom Matching Algorithmus minimiert werden muss, wird einerseits von den gefunden Korrespondenzen und ihrer Anzahl beeinflusst, andererseits von der Qualität der gefundenen Pose, d.h. von den Abbildungsparametern (hier Rotation $\mathbf{R}$ und Translation $\mathbf{t}$).

Der räumliche Fehler err_r zwischen Modell- und Bildelementen wird in vielen Arbeiten [27] nur im Bild berechnet, d.h. die geschätzte Pose wird zur Projektion des Modells benutzt, sodass nur der räumliche 2D-Abstand von Bildelementen bestimmt werden muss. Wir verwenden als räumliches Fehlermaß dagegen den tatsächlichen Abstand von 3D-Modellelementen (Punkten) und den projektiven 3D-Geraden bzw. projektiven 3D-Ebenen, die aus den extrahierten Bildprimitiva mit Hilfe der Projektionsmatrix gewonnen

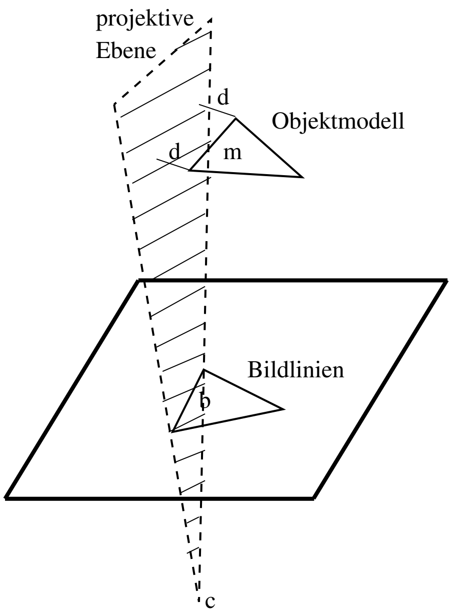

Abbildung 5.5: Darstellung des Hesseabstandes zwischen Punkt und Ebene

werden (siehe auch Kapitel 2). Der Abstand ist dabei der Hesseabstand zwischen Punkten und Geraden, bzw. zwischen Punkten und Ebenen. In Abbildung 5.5 ist dieser Abstand als d eingezeichnet.

Natürlicherweise ist der räumliche Fehler bei einer geringeren Anzahl von benutzten Korrespondenzen geringer als bei vielen Korrespondenzen. Deshalb mitteln wir den räumlichen Fehler auf die Anzahl der verwendeten Korrespondenzen,

$$err_r = \frac{1}{M} \sum_{i}^{M} err_i^c. \tag{3.8}$$

Dabei sind $err_1^c, err_2^c, .., err_M^c$ die Fehler für jeweils ein korrespondierendes Paar $(m, b) \in S'$ mit der Modelllinie $m \in M$ und dem zugehörigen Bildelement $b \in B$. Der Fehler setzt sich zusammen aus dem Abstand des Anfangs- und Endpunkts der Modelllinie zu der projektiven Ebene, für jedes korrespondierende Paar also aus zwei Abstandswerten.

Als Fehlermaß nur den räumlichen Abstand zu verwenden ist nicht ausreichend, weil eine größere Anzahl von Korrepondenzen die Wahrscheinlichkeit steigert, dass die gefundene Pose korrekt ist. Bei minimaler Korrespondenzanzahl führen viele Positionen und Orientierungen des Objektes zu einer Pose mit geringem Fehler err_r.

Je mehr Korrespondenzen zur Berechnung der Pose benutzt werden, desto höher ist die Wahrscheinlichkeit, dass die gefundene Pose die tatsächliche Lage des Objekts beschreibt. Deshalb wird der Fehler err_r zusätzlich noch mit der Anzahl der Korrespondenzen gewichtet, sodass Posen mit vielen Korrespondenzen einen kleineren Fehler haben. Damit ist das Fehlermaß für eine Pose,

$$err_r = \frac{1}{\sigma + C} err_r, \tag{3.9}$$

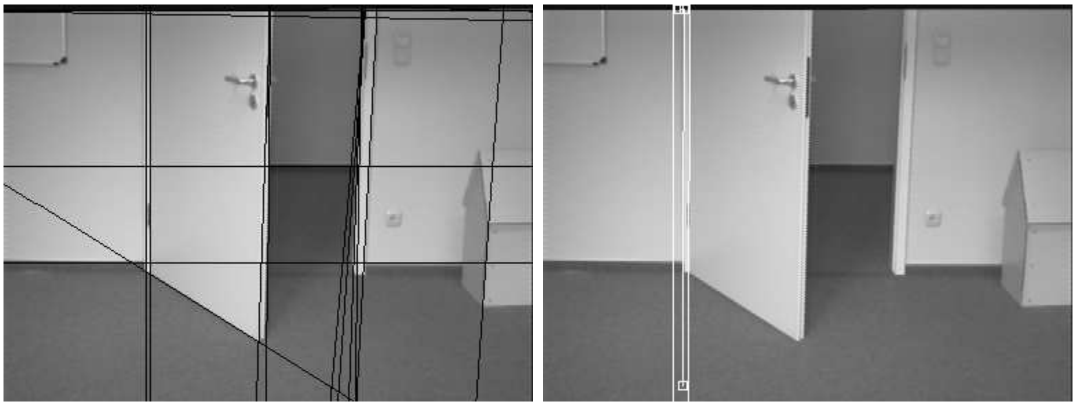

Abbildung 5.6: Fehlberechnung einer Pose (rechts) aufgrund von zuwenig benutzten Korrespondenzen

wobei C die Anzahl der Korrespondenzen und σ einen Gewichtungsfaktor darstellt, der bei hohen Werten die Bedeutung der Korrespondenzanzahl abschwächt.

Eine Extremsituation mit zuwenig Korrespondenzen ist in Abb. 5.6 sichtbar. Obwohl der räumliche Fehler err_r weniger als $1mm$ beträgt, ist die ermittelte Pose offensichtlich falsch. Da wir erlauben, dass mehrere Modelllinien mit nur einem Bildelement korrespondieren, ist es möglich, dass alle zur Berechnungen der Pose benutzten Modelllinien mit nur 1 oder 2 Bildgeraden korrespondieren. In einem solchen Fall ist der Fehler sehr klein und die Gewichtung aus Gleichung (3.9), die mehr Korrespondenzen belohnt, ist nicht in der Lage Posen mit mehr Korrespondenzen einen kleineren Fehler zuzuordnen. Um solche Fälle zu unterdrücken, geben wir eine Mindestanzahl von notwendigen Korrespondenzen vor, die höher ist als zur Berechnung der Pose erforderlich. Wir benutzen bei 16 Modelllinien z.B. 7 Korrespondenzen.

Das Poseschätzverfahren nimmt eine beliebige Rotation und Translation an. Da wir im Rahmen unserer Arbeit aber jeweils nur kleine Rotationen betrachten, ist es sinnvoll Posen, die eine große Rotation bedeuten, zu bestrafen. Dies geschieht durch einen Fehlerterm, der Posen mit kleine Rotationen unbeeinflusst lässt und größere Rotationen unwahrscheinlich macht. Dieser Fehlerterm ist multiplikativ gewählt, damit größere Rotationen nicht unmöglich werden. Der Gesamtfehler ist dann,

$$err_m = \mathbf{f}_{err}(\alpha) err_r. \tag{3.10}$$

Dabei ist α der Rotationswinkel und $\mathbf{f}_{err}$ ist eine angepasste Gaussfunktion (siehe Abbildung 5.7). Die Funktion $\mathbf{f}_{err}$ erhöht den Gesamtfehler von Posen, die eine Rotation von 100 Grad bis 260 Grad beinhalten, um das 5-fache, weil Rotationen größer als 90 Grad sehr unwahrscheinlich sind. Sie hat folgende Form,

$$\mathbf{f}_{err}(\alpha) = 1 + 4e^{(-\frac{(\alpha-180)^6}{295})}. \tag{3.11}$$

Dabei liegt der Rotationswinkel α im Intervall [0...360] Grad.

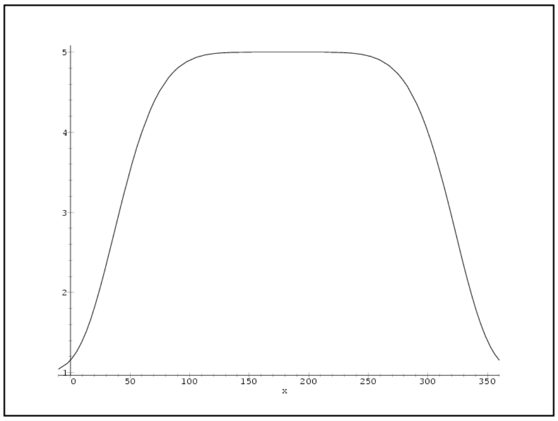

Abbildung 5.7: Funktion zur Bestrafung von großen Rotationen

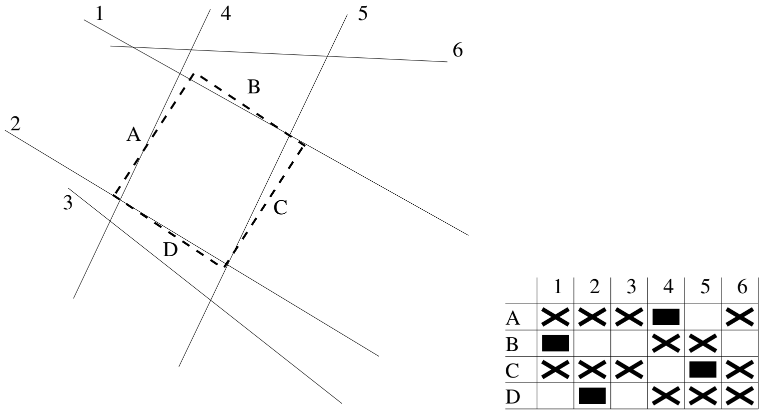

Abbildung 5.8: Suchbereicheinschränkung und zugehörige Fitmatrix am Beispiel eines Rechtecks

5.3.3 Matching Algorithmus

Der gesamte Algorithmus zur Berechnug der Pose ist in einer `C++`-Klasse implementiert [26], die als Eingaben das aufgenommene Kamerabild, ein Drahtgittermodell des Objekts und eine Projektionsmatrix erhält. Berechnet wird die Rotation und die Translation, die das abgebildete Modell mit dem Originalbild in Übereinstimmung bringt. Enthält das Modell bewegliche Gelenke, so wird zusätzlich der Gelenkwinkel bestimmt.

Der Suchraum wird repräsentiert durch eine Fitmatrix, dabei steht jede Zeile für eine Modelllinie und jede Spalte für ein Bildelement. Die Einträge der Matrix geben an, ob eine Korrespondenz zwischen Modelllinie und Bildelement möglich (Eintrag: 0), unmöglich (Eintrag: 2) oder gerade getestet/gefunden wird (Eintrag: 1). In Abbildung 5.8 ist sichtbar, wie die Trackingannahme den Suchbereich einschränkt. Die projizierten Modelllinien sind mit Buchstaben markiert und die extrahierten Bildgeraden mit Zahlen. In der Fitmatrix (rechts unten) ist sichtbar, welche Korrespondenzen möglich (weiß), welche unmöglich (Kreuz) und welche tatsächlich gefunden sind (schwarz). Die

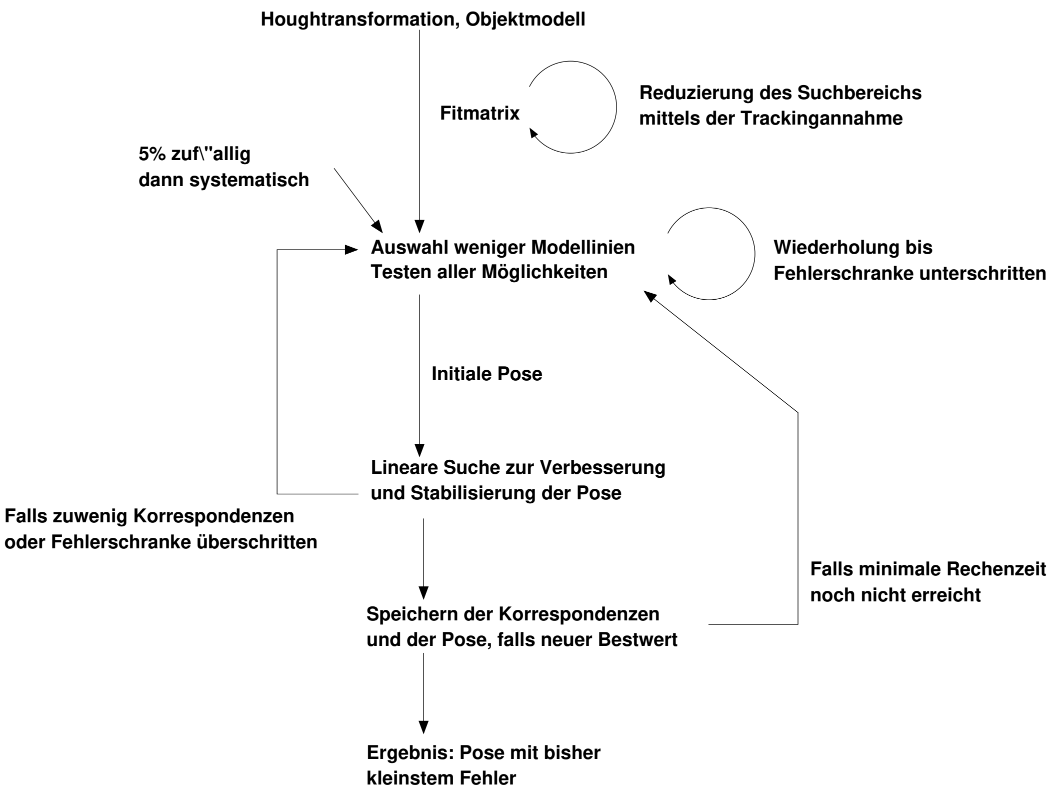

Abbildung 5.9: Schema des Matching Algorithmus

Einschränkung des Suchbereichs (Kreuze in der Fitmatrix) ist hier durch Pixelabstand und Winkeldifferenz geschehen.

Die Basis des Matchingalgorithmus ist ein lokales Suchverfahren mit zufälliger Startposition [6] und schematisch dargestellt in Abbildung 5.9. Das Prinzip der Heuristik liegt in einer Kombination aus iterativer Verbesserung und zufälliger Auswahl. Zunächst werden wenige Modelllinien ausgewählt (beispielsweise 5), dann wird für alle möglichen Korrespondenzen dieser Modelllinien mit allen möglichen Bildgeraden die Pose berechnet und anhand des Fehlers entschieden, ob diese Pose als initiale Pose verwendbar ist. Falls dem so ist, wird diese initiale Pose verbessert, indem jeweils eine weitere Modelllinie hinzugenommen wird und nur für diese Modelllinie alle Korrespondenzen mit allen möglichen Bildgeraden getestet werden. Wird für diese Modelllinie eine Bildgerade gefunden, sodass der Fehler unter einem Grenzwert bleibt, so wird dieses Paar (m, b) zur Korrespondenz hinzugenommen.

Die jeweils besten Posen werden gespeichert und eine minimale Rechenzeit wird angegeben. Für die Verwendung mit dem Roboter kann so die Rechenzeit an die Fahrtzeit, die der Roboter zwischen der Bildaufnahme benötigt, angepasst werden. D.h. während der Fahrt des Roboters kann parallel immer das letzte Bild ausgewertet und nach der besten Pose gesucht werden.

5.3.4 Stereo Matching

Das verwendete Robotersystem besitzt zwei Kameras, sodass zur Selbstlokalisation ein Stereobildpaar zur Verfügung steht. Wir führen das monokulare Verfahren des Matching getrennt an jedem Bild des Stereobildpaares durch und benutzen die gefundene Transformation aus jedem Bild zur Verifikation der Pose. Ist im linken und rechten Bild ein Matching erfolgreich, so sollte die entsprechende relative Bewegung von beiden Bildern nahezu identisch sein. Wir kombinieren die Informationen beider Bilder durch Bildung des Mittelwertes der beiden Einzeltransformationen. Sind die Abweichungen zwischen beiden Bildern groß, so kann man annehmen, dass in einem Bild eine falsche Pose bestimmt wurde.

Wir verwenden zur Korrektur der Odometriewerte die gefundene Transformation nur dann, wenn das Matching in beiden Bildern erfolgreich ist und die Abweichungen klein sind. Andernfalls kann eine falsch bestimmte Pose zu Korrekturen der Odometrie führen, die den Roboter in einen Zustand versetzt, in dem es ihm nicht mehr möglich ist, die Landmarke zu zentrieren.

Kapitel 6

Beschreibung des Robotersystems

In diesem Kapitel wird zunächst der verwendete Roboter mit seinen Eigenschaften vorgestellt. Anschliessend erfolgt eine detailiertere Beschreibung der Systemarchitektur, die bereits im Kapitel 3 erwähnt wurde. Insbesondere wird auf einzelne implementierte Module und deren Wechselwirkung eingegangen.

6.1 Verwendete Hardware

Die Basis für diese Arbeit bildet ein Roboter vom Typ B21 der Firma RWI [47]. Er hat einem maximalen Durchmesser von $53cm$ an der Basis und eine Höhe mit Aufbauten von ca. $175cm$. Damit befinden sich die Kameras in menschlicher Augenhöhe (vgl. Abbildung 6.1). Das Fahrwerk erlaubt unabhängige Translations- und Rotationsbewegungen, dadurch ist es möglich auf der Stelle zu rotieren.

Zur Kollisionsvermeidung werden 24 Sonarsensoren verwendet, die in einer Höhe von $80cm$ ringförmig angebracht sind, in einem Abstand von jeweils 15 Grad. Der Öffnungswinkel des jeweiligen Ultraschallkegels beträgt nach Herstellerangaben ebenfalls 15 Grad und hat eine Reichweite von bis zu 5 Metern. Außerdem besitzt der Roboter taktile Sensoren, die eine Berührung des Rumpfes mit Hindernissen detektieren. Zusätzlich sind 2 Ringe von Infrarotsensoren angebracht, die aber nur für sehr kurze Entfernungen verwendbar sind.

Der Kamerakopf hat 4 Freiheitsgrade, auf ihm sind zwei Kameras angebracht. Schwenken und Heben/Senken des Kopfes ist möglich, sowie eine Vergenzbewegung der Kameras. Die Kameras erlauben die Montage unterschiedlicher Objektive und mittels der angeschlossenen Frame-Grabber-Karten sind Farbbildaufnahmen mit 20Hz möglich.

Alle Module und Algorithmen werden on-board berechnet von zwei parallel arbeitenden *Pentium III* Standard-PCs mit 800Mhz Taktfrequenz und Linux-Betriebssystem. Die Kontrollausgabe von Bildern und wichtigen Statusmeldungen erfolgt über eine Ethernet Funkverbindung auf einer SUN-Workstation.

Abbildung 6.1: Der verwendete Roboter B21 in der Frontalperspektive

6.2 Verwendete Software

Zur Lösung der Gesamtaufgabe wurden im Wesentlichen drei unterschiedliche Softwarepakete verwendet. Das erste sind die von RWI gelieferten Client/Server Programme (beeSoft [47]) zur Ansteuerung der Hardware, von der explizit nur Fahr- und Steuerbefehle verwendet werden. Außerdem geben diese Programme Zugriff auf die verschiedenen Sensorwerte.

Das zweite Softwarepaket wurde am *Lehrstuhl für Kognitve Systeme, Kiel* entwickelt (KIVIGraP) [26] und ist die Grundlage für die Bildvorverarbeitung, Poseschätzung und wesentliche grundlegende `C++`-Klassen, wie Matrizen und Bildklassen.

Das dritte Paket betrifft die Funktionen zum Ansprechen der Meteor *frame grabber* Karten und stammt wie die Karten von Matrox [32]. Es bildet die Hardware-Schnittstelle zur Bildaufnahme.

6.3 Systemarchitektur

In diesem Abschnitt werden einzelne, implementierte Module und deren Kommunikation untereinander detaillierter beschrieben.

Abbildung 6.2 zeigt das Schema der Architekur. Die Module Pilot und Navigator sind für die Erzeugung der Fahrbefehle und die Koordination verantwortlich. Das Modul TrackPose beinhaltet die Selbstlokalisation, also die Bildverarbeitung und das Tracking. Der *colliserver* führt die Fahrbefehle aus und vermeidet gleichzeitig Kollisionen. Die unteren drei Server Programme beinhalten die Hardwareansteuerung und sind Teil des Pro-

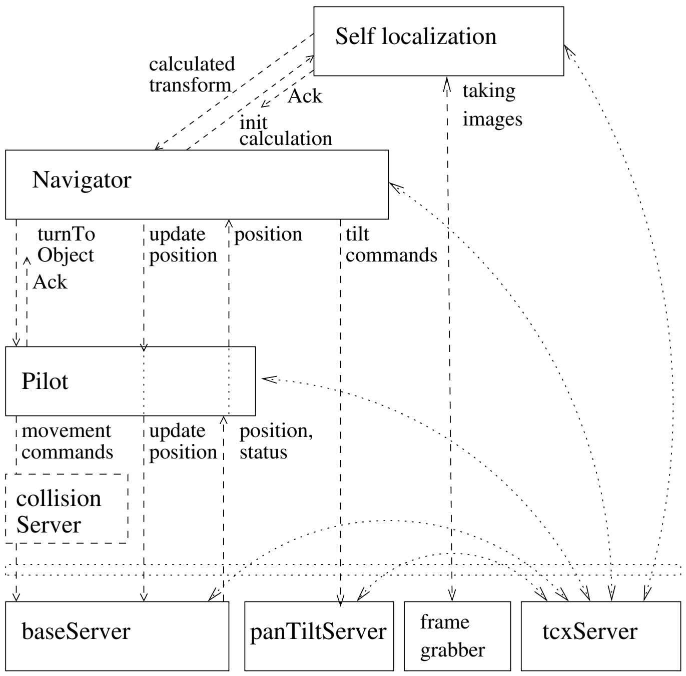

Abbildung 6.2: Schema der Systemarchitektur

grammpakets *beeSoft*, während der *frame grabber* zum Paket von Matrox und der übrige Teil zu *KIVIGrap* gehört.

Da alle Programmteile asynchron arbeiten, sind Nachrichten sowie Bestätigungen zwischen den Modulen notwendig, im Schema sind diese Nachrichten als Pfeile eingezeichnet.

6.3.1 Kommunikation

Zur Kommunikation zwischen den Clients und den Servern wird eine Inter- Prozesskommunikation mit dem Kürzel TCX benutzt [51]. Diese basiert auf dem Versenden von registrierten Nachrichten, für die der Empfänger eine Behandlungsfunktion beim tcx-Server registrieren muss. Die Clients sind hier die Module Pilot, Navigator und TrackPose.

Die Module Pilot und Navigator laufen in einem gemeinsamen Prozess auf dem gleichen PC zusammen mit den Hardware-Server-Programmen. Da beide Module in einem gemeinsamen Prozess ablaufen, geschieht die Kommunikation durch einfache Funktionsaufrufe.

Das Modul TrackPose, das die gesamte Bildverarbeitung und das Matching beinhaltet, benutzt den zweiten PC des Roboters.

Zum Datentransfer zwischen den Modulen Navigator und TrackPose wird das Linux-NFS (Net File System [13]) des Roboters verwendet. Der Navigator erzeugt aus den Odometriedaten eine Transformationsmatrix, welche die Bewegung des Roboters in Weltkoordinaten ausdrückt (siehe Kapitel 5). Diese Matrix wird im Modul TrackPose zur Anpassung der Projektionsmatrix verwendet und in einer Datei gespeichert. Das Modul

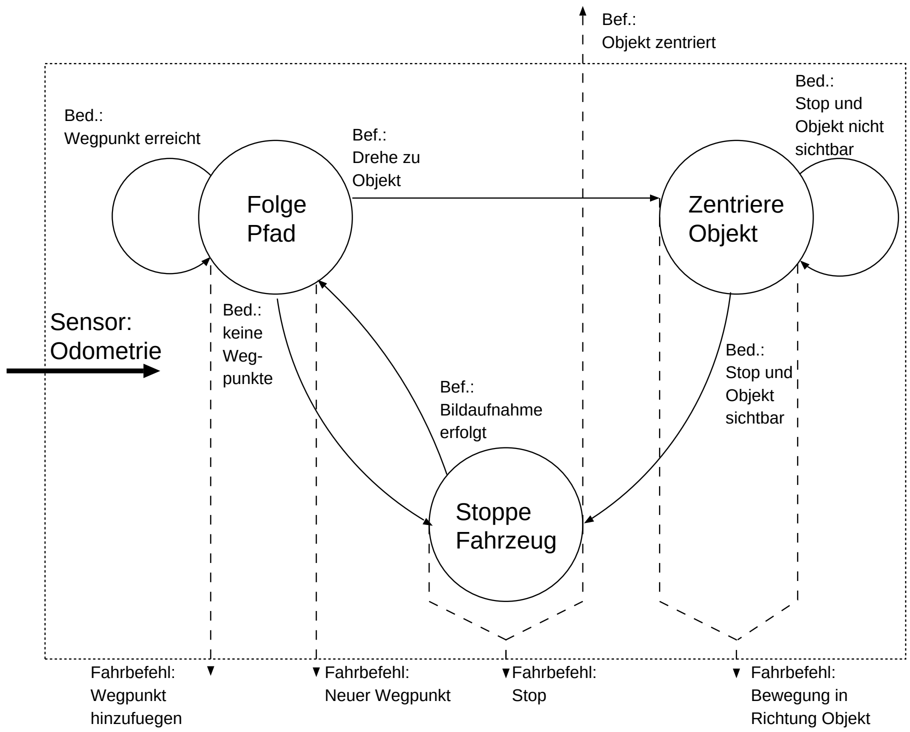

Abbildung 6.3: Beschreibung des Pilotverhaltens durch einen Zustandsautomat

TrackPose liest diese Datei und berechnet die tatsächliche Position des Roboters, führt also die Selbstlokalisation durch. Die Differenz (Rotationswinkel und 2D-Tranlsation) wird dann in einer Datei abgelegt, aus welcher der Navigator die zur Aktualisierung der Position notwendigen Werte ausliest.

6.3.2 Einzelne Module

Während Abbildung 6.2 den Daten- und Befehlsfluss des Gesamtsystems darstellt, wird in diesem Abschnitt auf das Verhalten einzelner Module detaillierter eingegangen. Zur Aufteilung des Gesamtsystems in Einzelverhalten wird auf Kapitel 3 verwiesen.

Modul Pilot

Die Aufgaben diese Programmteils ist, analog zu der Aufgabe eines Flugzeugpiloten, das tatsächliche Steuern des Fahrzeugs, d.h. das Folgen eines vordefinierten Pfades. Der Pfad kann von einem Verhaltensmodul höherer Komplexität erzeugt oder vom Benutzer vorgegeben sein. Der Pilot unterdrückt u.U. Bewegungsaktionen von höheren Schichten und erzeugt Fahrbefehle. Diese werden vom Modul Kollisionsvermeidung empfangen und ausgeführt. Der Pilot weiß also nicht genau, wie seine Aktion von der Schicht Kollisionsvermeidung ausgeführt wird. Er muss ständig prüfen, ob die gewünschte Situation erreicht ist.

Das Verhalten des Moduls Pilot lässt sich als endlicher Zustandsautomat [22] beschreiben, siehe Abbildung 6.3. Bislang sind drei Zustände definiert:

- **Folge Pfad** Dieser Zustand charakterisiert das Ziel, an einem vorgegebenen Pfad entlang zu einem Zielpunkt zu fahren.

- **Zentriere Objekt** Wenn ein Bild der nächsten sichtbaren Landmarke zur Selbstlokalisation aufgenommen werden soll, muss dieses zunächst ins Blickfeld geraten.

- **Stoppe Fahrzeug** Anhalten des Fahrzeugs.

Die Zustandsübergänge sind durch Pfeile angedeutet, tritt eine bestimmte Bedingung ein, so ändert sich der Zustand. Diese Bedingungen sind abhängig vom aktuellen Zustand und den Sensorwerten oder vom aktuellen Zustand und einem gegeben Befehl eines anderen Moduls (durch *Bed.* oder *Bef.* in der Abbildung abgekürzt).

Jeder Zustandswechsel erzeugt eine Aktion (gestrichelte Pfeile). Dies sind vorwiegend Fahrbefehle, die zum Modul *Kollisionsvermeidung* weitergegeben werden, oder Bestätigungen zum Erreichen eines gewünschten Zustands an andere Module. Beispiel: Nach dem Zentrieren der aufzunehmenden Landmarke muss das Fahrzeug im Stillstand sein, bis das Bild aufgenommen wurde. (Aktion: Befehl zentriert).

Alle Koordinaten, die innerhalb des Piloten benutzt werden, sind in Roboterkoordinaten angegeben.

Modul Navigator

Das Modul Navigator erzeugt ein Verhalten, das dem eines Flugzeugnavigators nachempfunden ist. Seine Aufgabe ist es, die Position zu bestimmen, dazu stehen ihm u.U. verschiedene Lokalisationstechniken zur Verfügung. Im Rahmen dieser Arbeit exisitert nur die Möglichkeit der Lokalisation anhand von bekannten Landmarken, für die ein Modell bekannt sein muss. Außerdem übernimmt der Navigator die Verwaltung und Transformation verschiedener Koordinatensysteme, hier Roboter- und Weltkoordinaten.

Das Gesamtverhalten lässt sich als endlicher Zustandsautomat beschreiben. Folgende Zustände sind definiert:

- **Planung** Die Hauptaufgabe des Navigators ist, anhand der Sensordaten zu entscheiden, ob eine Lokalisation notwendig und möglich ist. Wenn z.B. eine bestimmte Entfernung zurückgelegt wurde oder eine bestimmte Zeitspanne verstrichen ist.

- **Warten auf Zentrieren** Solange das Objekt sich nicht im Blickfeld befindet, kann die Bildaufnahme nicht erfolgen.

- **Warten auf update** Das Modul Selbslokalisation benötigt eine unterschiedliche Zeitspanne bis es ein Ergebnis erzeugt. In diesem Zustand wird auf das Ergebnis gewartet.

Die Zustandsübergänge sind durch Pfeile angedeutet (siehe Abbildung 6.4), tritt eine bestimmte Bedingung ein, so ändert sich der Zustand. Diese Bedingungen sind abhängig

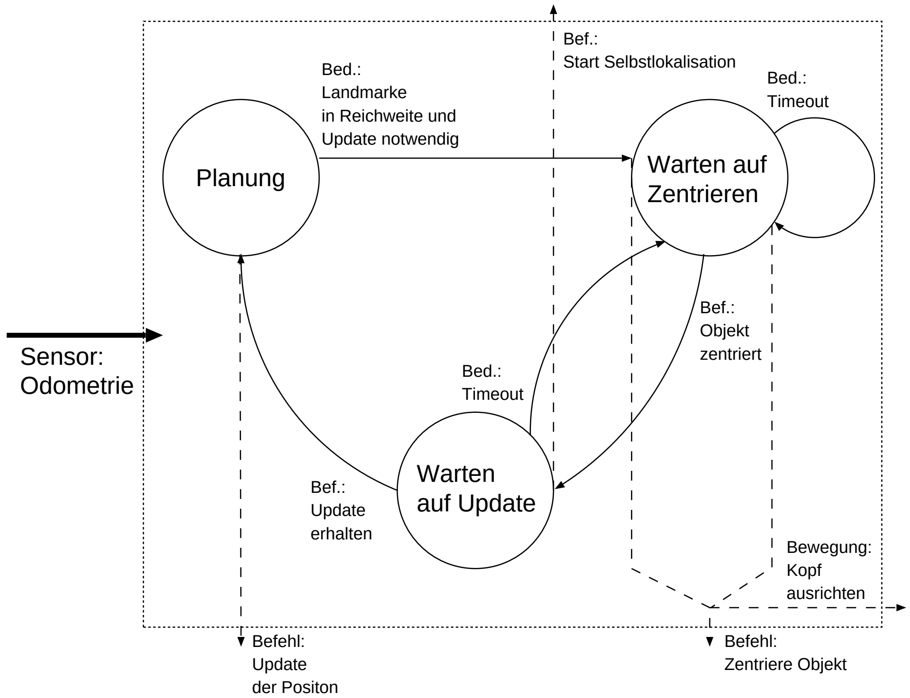

Abbildung 6.4: Beschreibung des Navigatorverhaltens durch einen Zustandsautomat

vom aktuellen Zustand und den Sensorwerten, oder vom aktuellen Zustand und einem gegebenen Befehl eines anderen Moduls (durch *Bed.* oder *Bef.* in der Abbildung abgekürzt).

Da niedrigere Schichten ein anderes, nicht erwünschtes Verhalten erzeugen können, gibt es für die Wartezustände Bedingungen, die nach einer gewissen Zeitspanne ausgelöst werden und den Lokalisationsprozess neu initieren.

Der Wechsel in den Zustand **Warten auf Zentrieren** löst eine Aktion aus, die den Piloten veranlasst das Fahrzeug in Richtung Landmarke zu bewegen.

Meldet der Pilot, dass der Roboter gestoppt ist und in Richtung Landmarke blickt, wird der Zustand nach **Warten auf update** gewechselt. Der Zustandswechsel mit dem Befehl *Start Selbstlokalisation* löst eine Aktion aus, welche die aus den Odometriedaten ermittelte Transformationsmatrix in eine Datei schreibt, die vom Modul Selbstlokalisation eingelesen wird. Befindet das Modul sich in diesem Zustand, so wird regelmäßig aus einer Datei gelesen, in der die ermittelte Transformation vom Modul *Selbstlokalisation* abgelegt ist.

Modul Kollisionsvermeidung

Dieser Programmteil ist in der Lage unter Verwendung unterschiedlicher Sensoren Kollisionen zu vermeiden und Hindernisse zu umfahren. Das Verfahren basiert auf dem dynamischen Fenster Ansatz [52] und wird genauso im RHINO-Projekt [11] benutzt.
Verwendbare Sensoren des Roboters sind:

- **Ultraschall- oder Sonarsensoren** 24 Stück mit 15 Grad Öffnungswinkel und einer Reichweite von $20 - 500cm$

- **Infrarotsensoren** 24 und 32 Stück in 2 Ringen für Reichweiten von $1 - 30cm$
- **taktile Sensoren** Diese Sensoren registrieren Kollisionen mit Hindernissen, pro Tür des Roboters 4, insgesamt 56.

Von jedem Sensor werden die Entfernungswerte über einen kurzen Zeitraum gemittelt und zur Berechnung von in der Nähe liegenden Hindernissen verwendet. Die taktilen Sensoren haben dabei die höchste Priorität, bei einer Kollsion stoppt der Roboter sofort und fährt ca. $10cm$ in die entgegengesetzte Richtung.

Die möglichen Bewegungen sind kurvenförmige Trajektorien, die auch von Fahrzeugen ausgeführt werden können, die eine PKW- ähnliche Anordnung von Rädern besitzen, d.h. Rotationen auf der Stelle sind nicht möglich. Fahrbefehle (v, w) bestehen aus einer Translationsgeschwindigkeit v und einer Rotationsgeschwindigkeit w. Für geeignete Fahrbefehle wird eine Suche im Raum der Geschwindigkeiten ausgeführt [18]. Entscheidend für die Suche ist eine Optimierungsfunktion:

$$\mathbf{f}(v, w) = \alpha \cdot \mathbf{angle}(v, w) + \beta \cdot \mathbf{dist}(v, w) + \gamma \cdot \mathbf{vel}(v, w) \tag{3.1}$$

Unter Beachtung der aktuellen Position und Orientierung des Roboters werden folgende Aspekte berücksichtigt:

- **Richtung zum Ziel**: Die Funktion **angle** ist ein Maß dafür, wie schnell das Ziel erreicht werden kann. Es ist maximal, wenn der Roboter exakt in Richtung Zielpunkt fährt.
- **Bewegungsfreiheit**: Die Funktion **dist** ist die Entfernung zum nächsten Hindernis. Je kleiner diese Distanz ist, desto größer ist die Notwendigkeit es zu umfahren.
- **Geschwindigkeit**: Die Funktion **vel** ist die Translationsgeschwindigkeit und belohnt hohe Geschwindigkeiten.

Mit den Werten α, β, γ kann das Verhalten des Moduls Kollisionsvermeidung beeinflusst werden, indem bestimmten Aspekten mehr Gewicht verliehen wird.
Setzt man z.B. $\alpha = 0$, so fährt der Roboter ziellos umher. Wählt man dagegen β zu hoch, so kann es passieren, das eine Türöffnung nicht als ausreichend breit erkannt wird und der Roboter nicht in der Lage ist, durch die geöffnete Tür zu fahren.

Modul Selbstlokalisation

Dieses Modul bestimmt die Differenzbewegung (Rotation und Translation) des Roboters zwischen der Position, die aus den Odometriedaten ermitteltet wurde, und der tatsächlichen Position relativ zu einer bekannten Landmarke. Das Verhalten lässt sich ebenfalls als endlicher Zustandsautomat beschreiben (siehe Abbildung 6.5). Die Zustände haben folgende Bedeutung:

- **Warten auf Odo Daten** Solange keine Positionsdaten der Odometrie vom Modul Navigator übermittelt werden, wird in diesem Zustand gewartet.

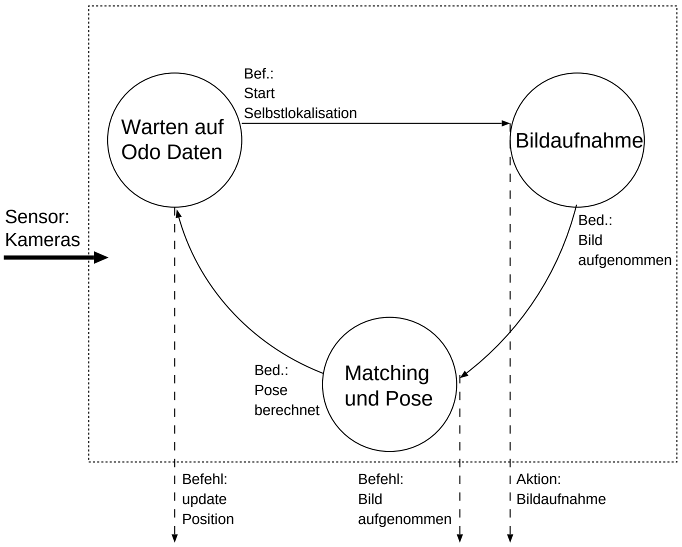

Abbildung 6.5: Beschreibung des Moduls Selbstlokalisation durch einen Zustandsautomat

- **Bildaufnahme** Ist die Bildaufnahme beendet, wird der Navigator darüber informiert.
- **Matching und Pose** Hauptverhalten des Moduls. Hier wird die Differenzbewegung in Weltkoordinaten berechnet, wie in Kapitel 4 und 5 beschrieben.

Das Verhalten des Moduls ist beim ersten Lokalisationsschritt etwas unterschiedlich, weil zu Beginn die geladenen Projektionsmatrizen an die aktuelle Startposition angepasst werden. Ausgehend von der Startposition werden die initialen Matrizen entsprechend der Roboterbewegung (Odometrie) in späteren Schritten angepasst. Die Umrechung der Odometriewerte von Roboter- nach Weltkoordinaten übernimmt das Modul Navigator.

Kapitel 7

Experimente

In diesem Kapitel werden verschiedene Experimente beschrieben, welche die Qualität und Quantität der implementierten Verfahren testen. Zunächst erfolgt ein Versuch zur quantitativen Bewertung der Selbstlokalisation. Anschließend wird das Gesamtsytem qualitativ untersucht. Zum Abschluß werden Versuche mit unterschiedlichen Landmarken vorgestellt.

7.1 Quantitative Analyse der Selbstlokalisation

Das Ziel des ersten Versuches ist die quantitative Bestimmung des Odometriefehlers und seiner Korrektur mit dem Verfahren zur Selbstlokalisation anhand einer Landmarke (siehe Kapitel 4 und 5).

Dazu verwenden wir als Landmarke das Modell eines Eckhauses mit den Maßen $L : 100cm \times B : 90cm \times H : 80cm$ (Abbildung 7.1). Dem Roboter wird ein Pfad vorgegeben, der einem Reckteck mit den Maßen $120cm \times 140cm$ enstpricht, die Eckpunkte des Rechtecks werden auf dem Fußboden markiert und die Position des Roboters relativ zu den Marken manuell bestimmt. Aus diesen Distanzen lässt sich der real zurückgelegte Weg berechnen. Das Rechteck wird mehrfach umfahren, so das der gesamte zurückge-

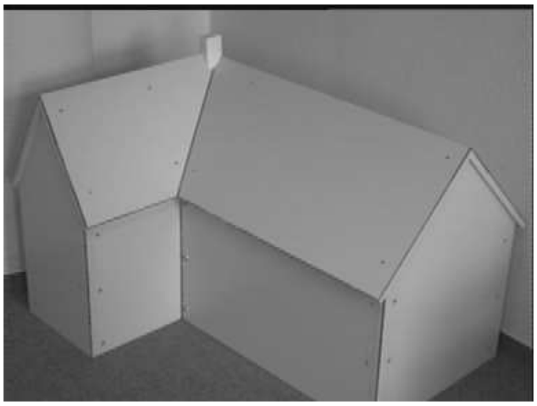

Abbildung 7.1: Die verwendete Landmarke

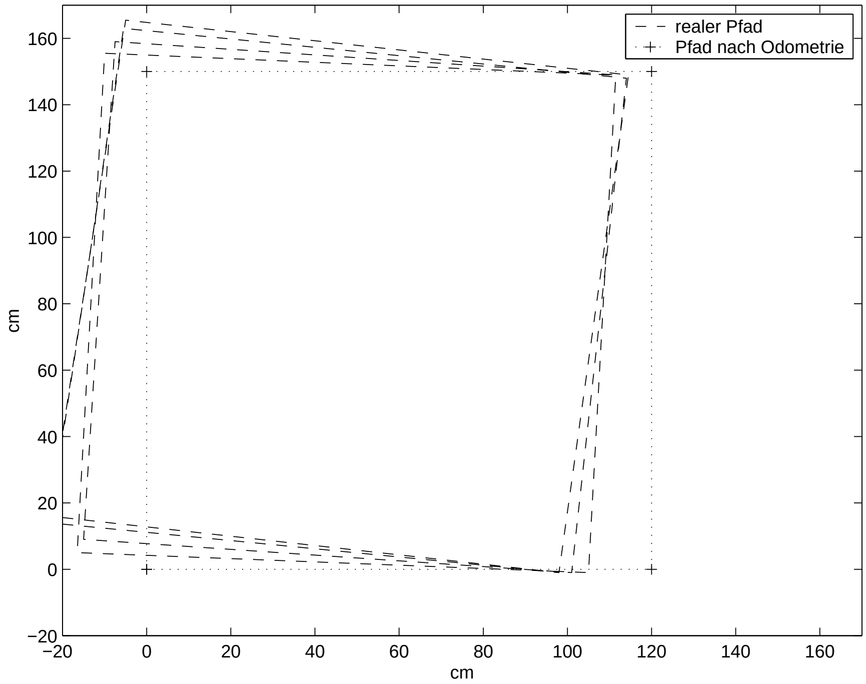

Abbildung 7.2: Realer und vorgegebenr Pfad

legte Weg etwa 50 Meter beträgt. Der vorgegebene Weg und der tatsächliche Weg sind in Abbildung 7.2 sichtbar. Die vorgegebenen Fahrbefehle des Roboters entsprechen dem idealen Rechteck, dem gefolgt werden soll. Die gestrichelte Linie in der Abbildung zeigt den real zurückgelegten Weg. Dabei wurde die Position nur an den Eckpunkten bestimmt und zwischen den Ecken interpoliert.

Zusätzlich wird an jeder Ecke die Selbstlokalisation durchgeführt und das interne Koordinatensystem des Roboters aktualisiert. Diese Veränderung zeigt sich in der Abbildung 7.2 nicht, weil die Fahrbefehle zum Folgen des Pfades relativ zur aktuellen Position angegeben sind. Anhand der Abbildung ist zu erkennen, dass die Abweichung des realen Pfades von dem idealen Pfad vor allem auf Rotationsfehler zurückzuführen ist.

Der Positionsfehler für jede Ecke ist in Abbildung 7.3 graphisch dargestellt. Zu sehen ist der Abstand von der realen Position (*Grundwahrheit*) zu der Soll-Position (dem idealen Rechteck) als gestrichelte Linie und der Abstand von der realen Position zur korrigierten Position des internen Koordinatensystems als durchgezogene Linie. Es fällt auf, dass der Positionsfehler auch nach der Korrektur um $10cm$ schwankt. Da der reale Fehler an der rechten oberen Ecke kleiner ist als $10cm$, wird hier der reale Fehler sogar kleiner als der Fehler nach der Korrektur. Dies lässt sich dadurch erklären, dass bereits beim ersten Schritt eine Abweichung von der Soll-Position von $10cm$ besteht und die rechte obere Ecke der Startpunkt ist. Da sich der Rotationsfehler konstant entwickelt hat, ist es verständlich, dass der Positionsfehler am Startpunkt relativ klein bleibt, weil der Startpunkt das Rotationszentrum ist, wie in Abbildung 7.2 ersichtlich. Zufällig nimmt der

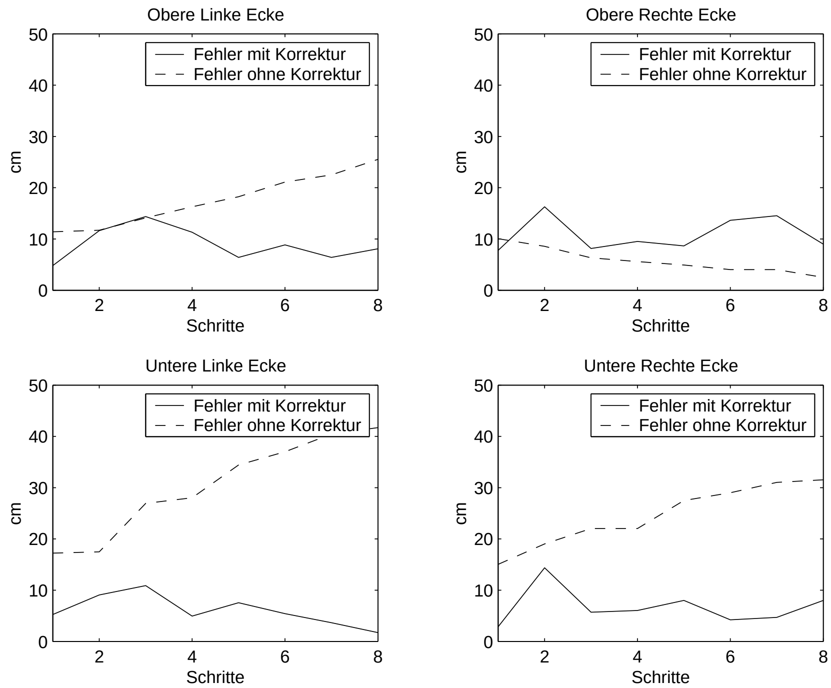

Abbildung 7.3: Fehler für die 4 Ecken des Rechtecks

Fehler deshalb sogar noch leicht ab.

Zusätzlich zur Position des Roboters liefert die Odometrie auch die Orientierung, die, wie bereits festgestellt, einen hohen Fehler aufweist (Rotationsfehler). An jedem Eckpunkt wird durch die Selbstlokalisation auch die aktuelle Orientierung bestimmt. Der Fehler der Orientierungsbestimmung ist in Abbildung 7.4 sichtbar. Der Fehler ohne Korrektur ist die absolute Differenz zwischen der Soll-Orientierung (ideales Rechteck) und der Orientierung, die aus jeweils zwei manuell gemessenen Positionen (*Grundwahrheit*) bestimmt wurde. Der Fehler mit Korrektur ist die absolute Differenz zwischen der Orientierung, die von der Odometrie nach der Aktualisierung geliefert wurde, und der manuell gemessenen Orientierung (Grundwahrheit).

Insgesamt lässt sich feststellen, dass die Bestimmung der Orientierung, die in einem Fehlerbereich von ca. 2 Grad liegt, gegenüber dem Orientierungsfehler von 12 Grad genauer ist als die Bestimmung der Position, die in einem Fehlerbereich von etwa $10cm$ liegt. Die Orientierung, die durch die Selbstlokalisation berechnet wird, ist noch etwas genauer als die bestimmte Abweichung in Abbildung 7.4, weil die von der Odometrie gelieferte Orientierung nicht direkt nach der Positionsbestimmung, sondern erst nach der Rotation in Richtung der nächsten Landmarke genommen wurde.

Der Fehler der Selbstlokalisation lässt sich im wesentlichen auf Fehler in der Tiefenbestimmung zurückführen. Da der Roboter sich in einer Entfernung von 2 bis 4 Metern von der Landmarke bewegt und die Kamera einen Öffnungswinkel von nur 30 Grad besitzt, ist eine Ungenauigkeit von $10cm$ in der Entfernungsbestimmung bei einer Auflösung von 384×288 nicht ungewöhnlich. Aus Stereoverfahren zur Tiefenbestimmung ist bekannt [15], dass Subpixel Genauigkeit notwendig ist, um exakte Tiefenwerte zu erhalten. Zusätz-

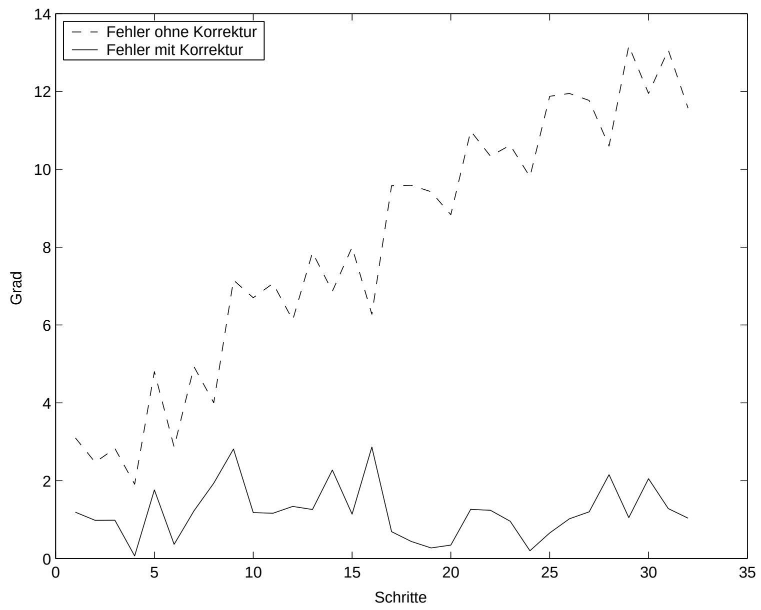

Abbildung 7.4: Orientierungsfehler während dem Folgen des Rechteckpfades

lich beeinflussen weitere Faktoren den Fehler der Positionsbestimmung. Diese werden im nächsten Kapitel diskutiert.

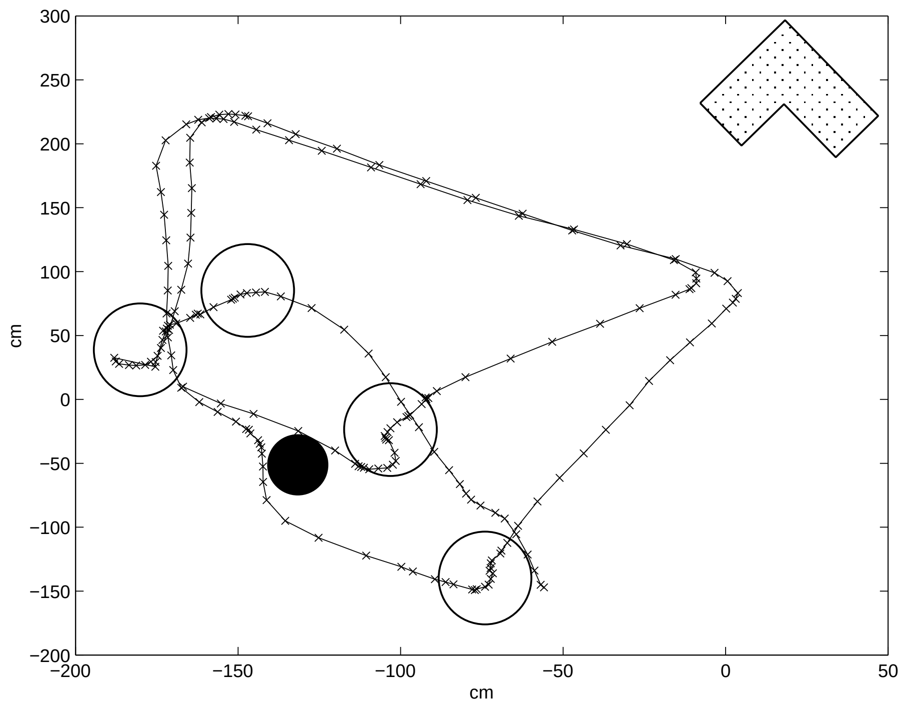

Abbildung 7.5: Gefahrener Weg im Zeitraum von 2,5 Minuten

7.2 Qualitative Analyse der Selbstlokalisation

Die Experimente in diesem Abschnitt dienen dem Ziel, die Funktionsweise der Selbstlokalisation im Gesamtsystem qualitativ beurteilen zu können. Dazu werden zwei Experimente zur Orientierung mit unterschiedlichen Landmarken durchgeführt.

7.2.1 Experiment in einem Laborraum

Dieser Versuch soll zeigen, dass der Roboter auch über einen längeren Zeitraum hinweg in der Lage ist, sich anhand einer gegeben Landmarke zu orientieren und Hindernissen auszuweichen.

Es werden fünf Wegpunkte innerhalb eines Raumes vorgegeben, die der Roboter in einer Endlosschleife versucht zu erreichen. Der Roboter ist ca. eine Stunde in Bewegung und hält nur an, um Bilder der Landmarke aufzunehmen. Der Kamerakopf bewegt sich während dieses Versuchs nicht. Der Neigungswinkel des Kopfes ist auf 25 Grad eingestellt und die Stellung der Vergenzen mit 4 Grad nach innen. Die Selbstlokalisation wird etwa alle 60 Sekunden ausgeführt, wenn die Entfernung zur Landmarke größer als $250cm$ ist. Außerdem wird die Position des Roboters zweimal pro Sekunde gespeichert, sodass der gefahrene Weg nachvollziehbar ist. Abbildung 7.5 zeigt einen Ausschnitt des Weges, der in etwa $2,5$ Minuten zurückgelegt wurde. Der Robter fuhr den dargestellten Weg entgegen dem Uhrzeigersinn. Kreisförmig markiert sind die Positionen, an denen die Landmarke zentriert und Bilder aufgenommen wurden. Die beiden Kreise auf der rechten Seite zeigen deutlich, wie der Roboter sich langsam in Richtung der Landmarke bewegt hat. Es erfolgte

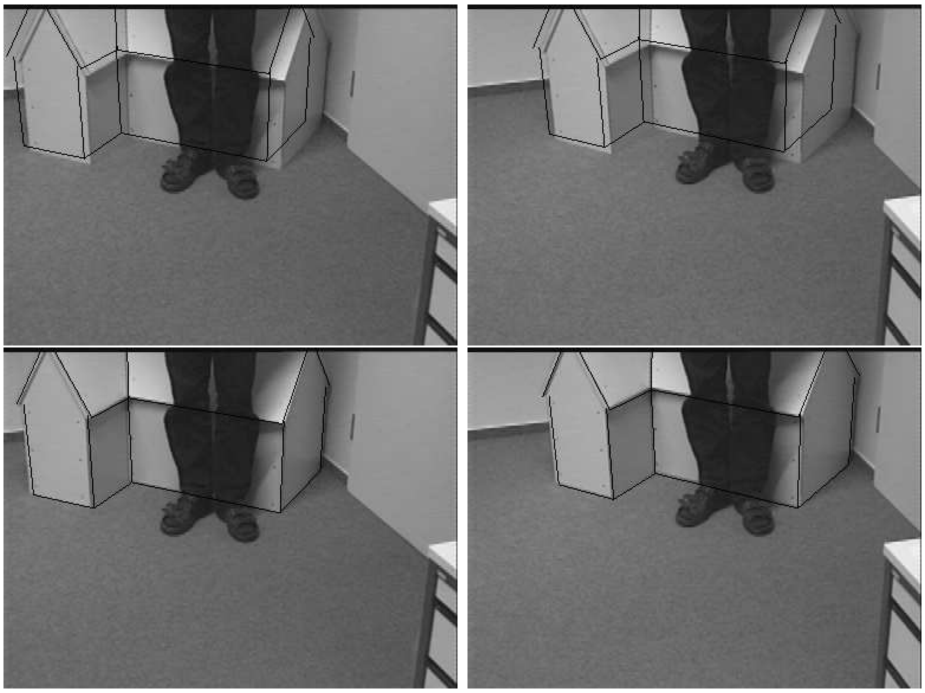

Abbildung 7.6: Erfolgreiches Matching mit Verdeckung (linke und rechte Kamera)

keine Rotation auf der Stelle, weil das Modul Kollisionsvermeidung nur kurvenförmige Fahrbefehle erzeugt.

Der schwarze Kreis deutet ein Hindernis an, dem ausgewichen wurde als eine Person sich dem Roboter in den Weg stellte. Gut zu erkennen ist das Bremsen (dichtere Ansammlung von Kreuzen in der Abbildung) und das seitliche Ausweichen.

Abbildung 7.6 zeigt Bilder der linken und rechten Kamera, die während des Matching aufgenommen wurden. Obwohl die Landmarke teilweise durch eine Person verdeckt war, konnte die Pose des Hauses erfolgreich bestimmt und daraus die Position des Roboters berechnet werden.

Während des gesamten Versuchs wurde an 94 Bildern die Selbstlokalisation durchgeführt. Davon war das Matching 9 mal nicht erfolgreich. Die ermittelte Tranformation wurde nur zur Aktualisierung der Position benutzt, wenn im linken und rechten Bild das Matching erfolgreich war und die gefundene Transformation sich nur wenig unterschied. Bilder, die bei einer großen Abweichung der Roboterposition von der angenommenen Position aufgenommen wurden, zeigt Abbildung 7.7. Die minimale Zeit für das Matching war mit 6 Sekunden pro Bild eingestellt und die maximale Zeit mit 15 Sekunden. Dazu kommen jeweils 0,5 Sekunden für die Bildvorverarbeitung und etwa 3-4 Sekunden für die Kontrollanzeige der Bilder auf der SUN- Workstation. Mit der Zeit, die zur Zentrierung der Landmarke gebraucht wurde, ergibt sich eine Gesamtzeit der Selbstlokalisation von etwa 50-60 Sekunden. Der Roboter kann jedoch bereits nach der Bildaufnahme seine Bewegung fortsetzen, weil die Selbstlokalisation parallel auf dem zweiten PC des Roboters abläuft.

Zusammenfassend lässt sich feststellen, dass das implementierte System in der Lage

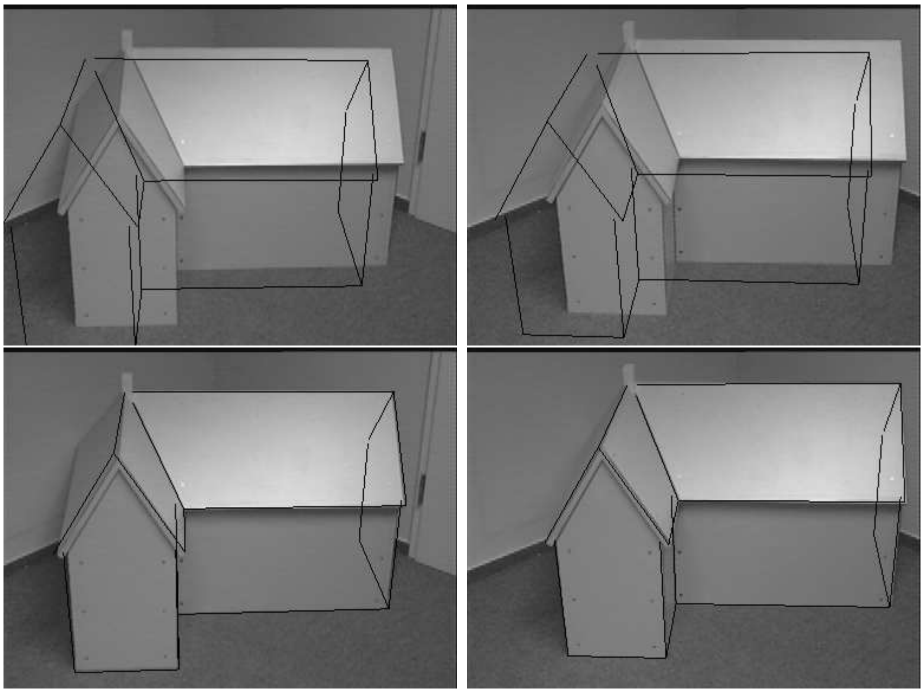

Abbildung 7.7: Erfolgreiches Matching trotz großem Odometriefehler (linke und rechte Kamera)

ist sich innerhalb des Sichtbarkeitsbereichs einer Landmarke zu orientieren und dabei Hindernissen auszuweichen.

Abbildung 7.8: Erfolgreicher Match des Durchgang-Modells

7.2.2 Experiment in einer Büroumgebung

Zusätzlich zu vorigem Versuch soll dieses Experiment zeigen, dass auch andere Landmarken zur Selbstlokalisation geeignet sind. Für diesen Versuch dient der in Abbildung 7.8 sichtbare Durchgang als Landmarke. In dieser Abbildung ist auch ein erfolgreiches Matching dargestellt.

Der Neigungswinkel des Kopfes war auf 10 Grad eingestellt und die Stellung der Vergenzen mit 0 Grad. Die Selbstlokalisation wurde alle 30 Sekunden ausgeführt, wenn die Entfernung zur Landmarke größer als $250cm$ war. Der Roboter bewegte sich nur in einem Bereich des Flures, von dem die Landmarke ausreichend sichtbar war. Er fuhr etwa $20min$, ohne sich zu verirren.

Ist das Matching im linken oder rechten Bild nicht erfolgreich, oder ist die Abweichung der berechneten Transformation zwischen dem linken und dem rechten Bild zu groß, wird das gesamte Ergebnis verworfen, um den Odometriefehler nicht noch zu vergrößern. Da bei diesem Versuch sehr häufig entweder nur im linken oder im rechten Bild das Matching erfolgreich war, wurde die gefundene Transformation auch verwendet, wenn nur in einem Bild die Tranformation bestimmt werden konnte.

Benutzt man nur ein Bild zur Aktualisierng der Position/der Odometrie, so kann ein Ergebnis wie in Abbildung 7.9 (rechts) dazu führen, dass der Roboter sich vollständig verirrt, d.h. der Odometriefehler so groß ist, das die Landmarke nach der Zentrierung nicht mehr im Bild ist. Deshalb wurde die gefundene Transformation von nur einem Bild limitiert auf $5cm$ in x- und y-Richtung sowie auf 1 Grad in der Rotation. Selbst wenn die Transformation in die falsche Richtung erfolgt, kann der Roboter den Fehler in späteren Schritten noch korrigieren.

Ein Auszug des in zwei Minuten zurückgelegten Weges ist in Abbildung 7.10 sichtbar. Das Zentrum der Landmarke befindet sich etwa an den Koordinaten $(500, 0)$ und ist deshalb nicht in der Abbildung eingezeichnet. Der Roboter bewegte sich nur in der Flurmitte, weil er versuchte einen Mindestabstand von $30cm$ zu allen Hindernissen einzuhalten. In der Abbildung ist außerdem der aktualisierte Positionsvektor zu erkennen (Sprung im Weggraphen durch Kreis gekennzeichnet). Der ausgefüllte Kreis deutet eine Person an, der ausgewichen wurde, als diese am Roboter vorbeiging.

Abbildung 7.9: korrekter Match links und fehlerhafter Match im rechten Bild

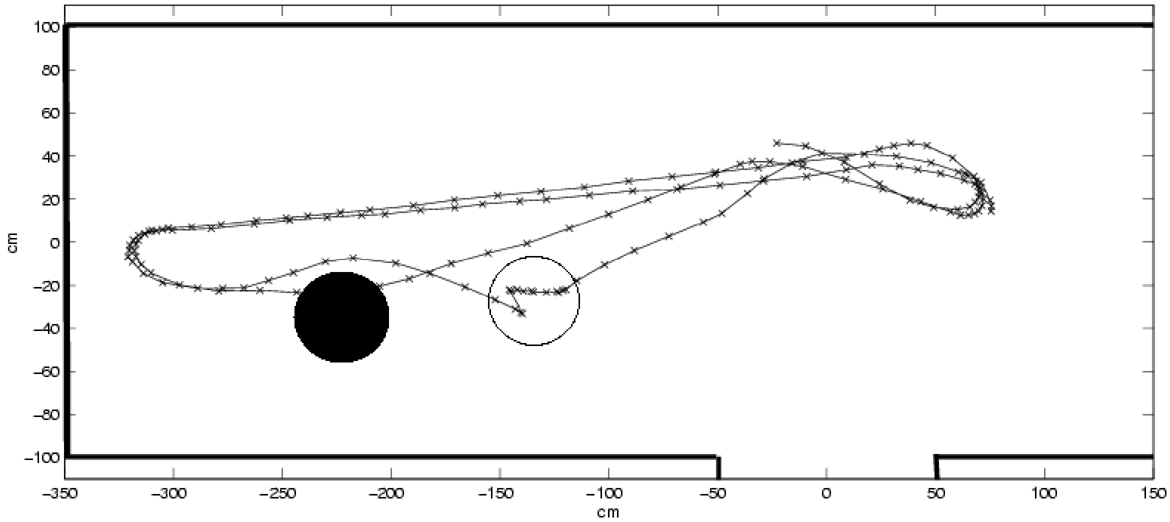

Abbildung 7.10: Gefahrener Weg im Zeitraum von 2 Minuten

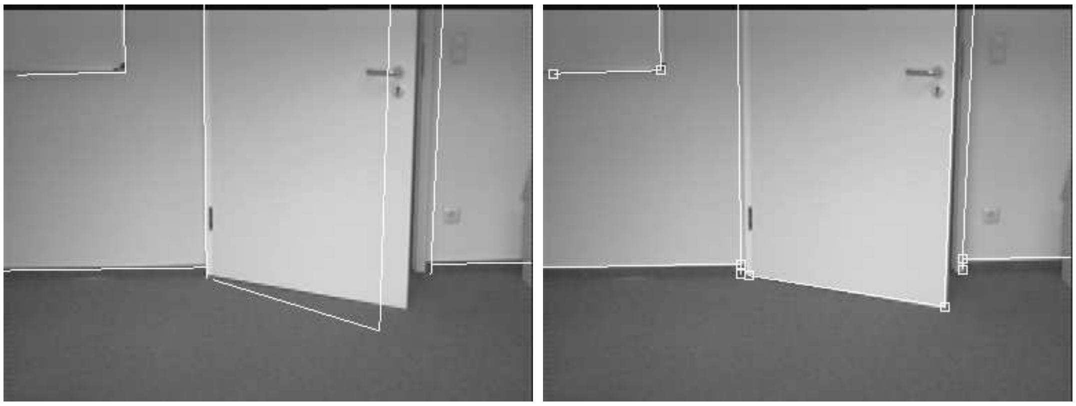

Abbildung 7.11: Bestimmung der Pose und des Rotationswinkels einer kinematischen Kette

7.3 Poseschätzung einer kinematischen Kette

Dieser Versuch soll das Matching und die Poseschätzung an einer kinematischen Kette (siehe Kapitel 2) demonstrieren. Als Landmarke dient eine normale Tür mit einer daneben hängenden Tafel. Zusätzlich zur Pose der Tür soll der Öffnungswinkel des Türblatts bestimmt werden.

In Abb. 7.11 ist links das noch nicht transformierte Modell (hell hervorgehoben) mit dem Originalbild zu sehen. Im rechten Bild ist die Pose nach der gefunden Transformation sichtbar. Der Öffnungswinkel des Türgelenks wurde ebenfalls erfolgreich bestimmt, sodass entschieden werden kann, ob der Roboter die Tür durchfahren kann oder nicht. Die Abbildung zeigt, dass ein erfolgreiches Matching mit Bestimmung des Öffnungswinkels prinzipiell möglich ist.

Während des Versuchs hat sich jedoch auch gezeigt, das die Erfolgsquote des Matching zu gering ist, um die Tür als Landmarke zur Navigation zu verwenden. Das Problem ist, die korrekten Korrespondenzen für die Modelllinien des Türblatts mit den Houghgeraden zu finden. Da das Modell des Türblatts nur 3 Linien enthält, ist die Wahrscheinlichkeit von Korrespondenzen dieser 3 Linen mit nicht korrekten Houghgeraden sehr hoch. Abbildung 7.12 zeigt ein Beispiel für falsch zugeordnete Geraden. Die Tür wurde nach hinten geöffnet. Obwohl dies mechanisch nicht möglich ist, ermittelte die Poseschätzung einen solchen Winkel, da keine Einschränkung des Winkels gemacht wurde und der Fehler der Pose bei dieser Konfiguration sehr gering ist.

Zusammenfassend lässt sich feststellen, dass die Tür mit variablen Öffnungswinkel als Landmarke zur Navigation nur geeignet ist, wenn der Suchraum der möglichen Korrespondenzen weiter eingeschränkt wird, z.B. durch Verwendung von hell-dunkel Übergängen oder durch spezielle Restriktionen für die Linien des Türblatts, wie mögliche Winkelbereiche etc.

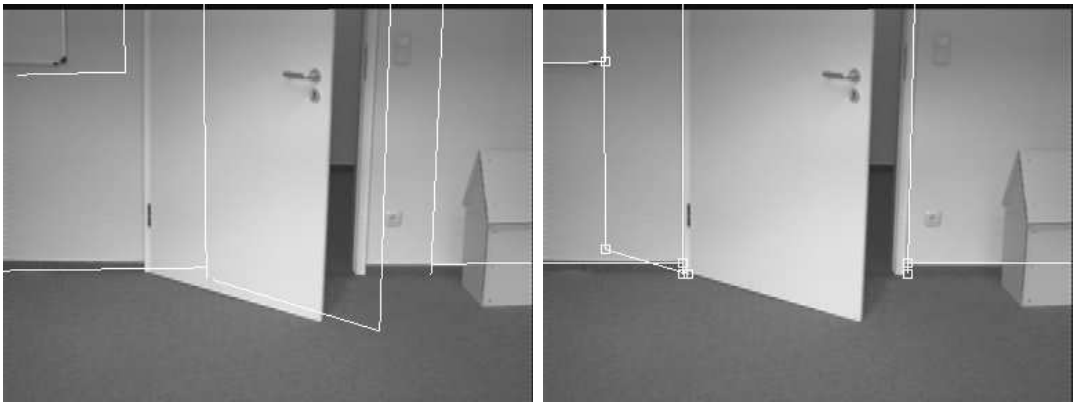

Abbildung 7.12: Der Rotationswinkel wurde falsch bestimmt (rechts) aufgrund nicht korrekter Korrespondenzen.

7.4 Kollisionsvermeidung

Zur Kollisionsvermeidung werden die Ultraschall- und Infrarotsensoren verwendet. Dabei hat sich gezeigt, dass die Infrarotsensoren den Ultraschallsensoren nur wenig Unterstützung bieten, weil diese nur in einer Entfernung von $0cm$ bis $30cm$ überhaupt etwas registrieren. Die Entfernung kann dabei nicht bestimmt werden, sondern lediglich, ob sich ein Hindernis in Reichweite befindet. Außerdem werden schwarze Objekte nicht erkannt, weil die Absorption des Infrarotlichts von schwarzen Flächen zu groß ist und die reflektierte Energie nicht ausreicht.

Zusätzlich kommt es vor, dass die Infrarotsensoren Falschmeldungen liefern, d.h. ein Hindernis detektieren, obwohl sich in $30cm$ Entfernung nichts befindet. Deshalb benutzen wir die Infrarotsensoren nicht, weil der Nutzen die Störreffekte nicht überwiegt.

Abbildung 7.13 zeigt eine Bildfolge, die das Ausweichverhalten darstellt. Eine Person läuft dabei dem Roboter in den Weg (Bild 1-3). Der Roboter stoppt, dreht sich nach links (Bild 4-9) und umfährt das Hindernis in großem Bogen (Bild 9-16).

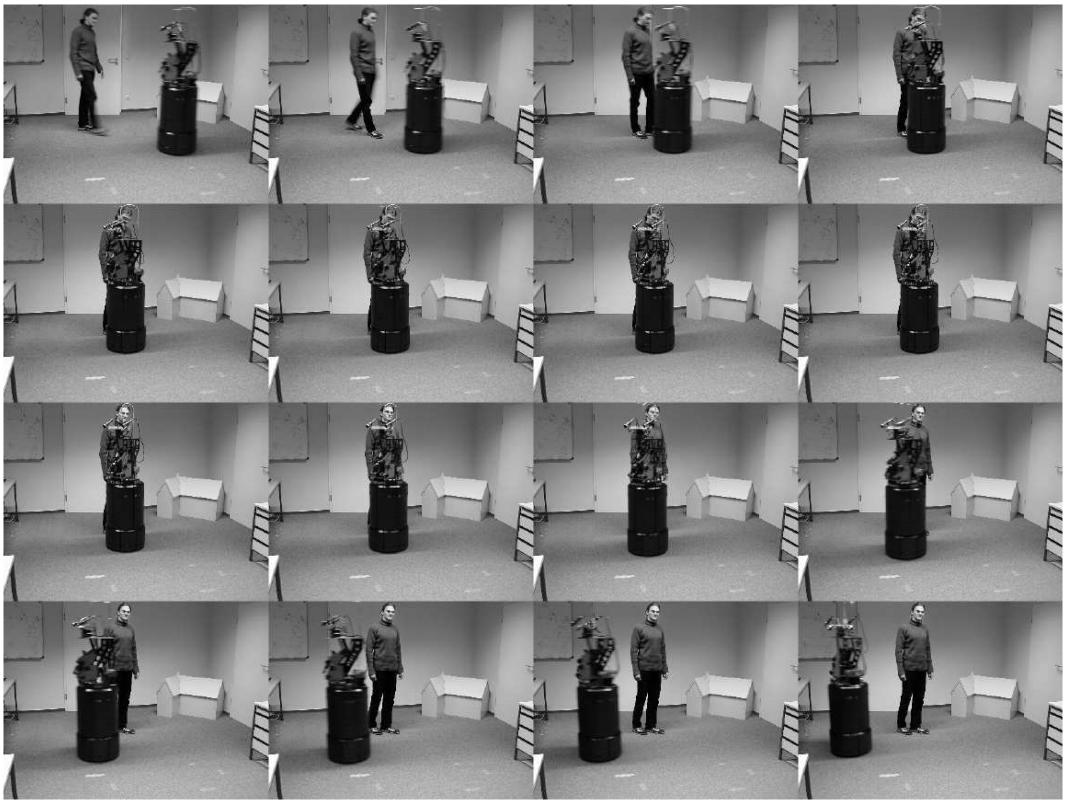

Abbildung 7.13: Ein Beispiel für die Kollisionsvermeidung

Kapitel 8

Diskussion und Ausblick

In diesem Kapitel werden die benutzten Verfahren und die Ergebnisse der Versuche zusammengefaßt und diskutiert. Anschließend wird ein Ausblick auf mögliche Erweiterungen gegeben.

8.1 Diskussion

Die Experimente des vorigen Kapitels haben gezeigt, dass das entwickelte System in der Lage ist, sich anhand einer Landmarke zu orientieren. Das implementierte Verfahren basiert auf der Positionsverfolgung, bei der die initiale Position bekannt ist. Die Genauigkeit der Positionsbestimmung reicht aus, um den Roboter im Langzeitexperiment bis zu einer Stunde lang im Sichtbarkeitsbereich der Landmarke zu bewegen, ohne dass dieser sich verirrt. Es werden keine besonderen Eigenschaften der Landmarken ausgenutzt. Das Verfahren ist geeignet für jede Art von Landmarke, solange sich sieben oder mehr Kanten im Bild extrahieren lassen.

Der erste Versuch zeigt, dass die Position bis auf einen durchschnittlichen Fehler von 10 cm und etwa 1 Grad in der Orientierung bestimmt werden kann. Unabhängig von Fehlern der Poseschätzung und des Matching tragen folgende Quellen zum Positionsfehler bei:

- **Genauigkeit der initialen Position** Diese wird manuell ermittelt. Die initiale Position definiert die Transformation zwischen den verwendeten Koordinatensystemen (Hand-Auge-Kalibrierung). Der Rotationsanteil der Transformation wird durch Zerlegung der Projektionsmatrix nach dem ersten Matching bestimmt. Diese Rotation ist bei dem Versuch im Flur mit einer Abweichung von ca. 8 Grad weit über der Genauigkeit, die manuell erreichbar ist. Eine Ausrichtung des Roboters zur Landmarke von Hand bedingt jedoch immer einen Fehler von 1-2 Grad.

- **Ausrichtung der Kameras** Zur Transformation der ermittelten relativen Bewegung in Weltkoordinaten nach Roboterkoordinaten wird angenommen, dass beide Kameras mit dem gleichen Vergenzwinkel nach innen ausgelenkt sind (siehe Anhang A). Eine Abweichung der Vergenzwinkel von linker und rechter Kamera ist äquivalent zu einer Orientierungsabweichung der Hand-Auge-Kalibirierung.

- **Kamerakalibrierung** Die Qualität der Projektionsmatrix hat entscheidenden Einfluss. Wir nehmen an, dass die *internen* Kameraparameter konstant sind und durch die *impilzite* Kalibrierung erfolgreich bestimmt werden können. Diese werden zwar nicht explizit berechnet, aber wir verändern durch Anpassung der Projektionsmatrix die externen Parameter. Deshalb verursachen Abweichungen der *internen* Kameraparameter von den realen Werten einen Fehler in der Bestimmung der relativen Transformation, auch wenn das Modell ideal an die Bilddaten angepaßt werden kann. Zusätzlich sind Bildaufnahmen aufgrund der Objektive oft leicht verzerrt. Solche Verzerrungen, wie sie bei Weitwinkelobjektiven sehr stark auftreten, sind durch das Lochkameramodell nicht beschreibbar und führen zu verstärkten Fehlern am Bildrand.

Trotz dieser unterschiedlichen Fehlerquellen, die einen kaum zu korrigierenden Grundfehler bei der Selbstlokalisation erzeugen, hat sich in den Versuchen zur qualitativen Analyse gezeigt, dass die Genauigkeit der Selbstlokalisation ausreicht, um den Roboter über einen längeren Zeitraum hinweg sicher zu navigieren. Wenig überraschend sind die Beobachtungen, dass der Roboter nach einem gewissen Zeitraum (ca. 20 Minuten beim Versuch in der Büroumgebung) in Situationen geraten kann, die eine ausreichend gute Zentrierung der Landmarke unmöglich machen, weil der Odometriefehler zu groß geworden ist. Dies geschieht meist dann, wenn die Selbstlokalisation nicht erfolgreich ist, d.h. ein Matching nicht möglich ist, weil die Landmarke nicht ausreichend im Bild sichtbar ist, z.B. wegen Verdeckung oder eines bereits zu großen Odometriefehlers. Der Roboter hat sich in einem solchen Falle verirrt.

Das Gesamtsystem wurde in Anlehnung an die *subsumption architecture* [7] in verschiedene Abstraktionsschichten aufgeteilt, sodass die Funktionalität einzelner Teilverhalten auch bei Ausfall höherer Schichten erhalten bleibt. Besonders auf die Möglichkeit der leichten Erweiterbarkeit wurde geachtet. Da untere Schichten die Aktionen höherer Schichten unterdrücken können, lassen sich weitere Abstraktionsschichten ohne große Anpassung anderer Schichten hinzufügen. Während der Versuche hat sich gezeigt, dass das Planungsverhalten der Navigator Schicht zu unflexibel ist, um den Roboter auch über längere Zeit hinweg sicher zu positionieren. Besonders während des Versuchs in der Büroumgebung verirrte der Roboter sich nach etwa 15 min. Ein Planungsverhalten, das auf solche Situationen reagieren kann, ist notwendig.

Auf der untersten Abstraktionschicht steht die Kollisionsvermeidung. Diese erzeugt ein zufriedenstellendes Verhalten. Personen, die während der Versuche um den Roboter herumliefen, konnte ausgewichen werden. Die Sonarsensoren sind für diese Aufgabe ausreichend, die Infrarot-Sensoren liefern dagegen keinen nennenswerten positiven Beitrag. Schwarze Objekte können von ihnen gar nicht detektiert werden, und die Entfernung kann auch nicht bestimmt werden. Sie geben lediglich an, ob sich etwas in ihrem Meßbereich befindet (1-30 cm). Zusätzlich melden sie gelegentlich Hindernisse, die gar nicht vorhanden sind.

Zur Bildvorverarbeitung wurden 2 Verfahren verwendet, die auf der Houghtransformation basieren. In den Versuchen hat sich gezeigt, dass die Maximumsuche mit Bereichslöschung oft besser geeignet ist, weil bei der Angabe von passenden Parametern weniger irrelevante Geraden gefunden werden als bei der Maximumsuche mit Verifikation im Originalbild. Dadurch ist weniger Zeit notwendig bis ein gültiger Match gefunden wird, weil der Suchraum möglicher Korrespondenzen kleiner ist. Die Maximumsuche mit Verifikation im Originalbild lässt jedoch meist eine genauere Bestimmung der Pose zu, weil auch dicht beieinander liegende, parallele Kanten extrahiert werden.

Die Beschreibung des Bildinhaltes durch Geraden mittels der Hough-Transformation bietet zwar den Vorteil einer gewissen Robustheit gegenüber Bildstörrungen, wie z.B. Verdeckungen, man verliert aber einen Teil der lokalen Information, da Hough-Geraden unendlich ausgedehnt sind und so Anfangs- und Endpunkte der Bildkanten nicht genutzt werden können. Außerdem grenzt die Verwendung von Geraden die Beschaffenheit der Landmarken ein. Die Verwendung des Strukturmultivektors [16] zur Beschreibung von Modellen oder von Bildinhalten ist flexibler und für weitere Untersuchungen vielversprechend. Wie bereits in [44] gezeigt wurde, ist die Poseschätzung von akkumulierten Modellen bestehend aus Liniensegmenten prinzipiell möglich. Zu bemerken ist, dass die Verwendung von Liniensegmenten den Suchraum drastisch vergrößert.

Mit dem verwendeten Verfahren zur Poseschätzung ist es auch möglich, die Position zusätzlicher Gelenke zu bestimmen. Für Landmarken oder Objekte, die bewegliche Gelenke enthalten, ist die Bestimmung geeigneter Korrespondenzen schwieriger, eine Anpassung des Matchingalgorithmus und die Beachtung von speziellen Eigenschaften der Landmarke und der Gelenke ist notwendig. In dieser Arbeit werden auschließlich Punkt/Ebene Bedingungen zur Anpassung der Pose verwendet. Extrahiert man nicht nur Geraden im Bild, sondern auch andere Elemente, wie Punkte oder Vertizes, können diese direkt zur Poseschäztung unter Verwendung der Punkt/Linie Zwangsbedingung benutzt werden.

8.2 Ausblick

Es existieren verschiedene Möglichkeiten das System zu erweitern:
Zunächst die Erzeugung einer Karte von Landmarken. Diese würde es dem Roboter gestatten sich in einer größeren Umgebung zu bewegen, die nur durch die Anzahl und Position der Landmarken beschränkt ist. Für eine solche Karte ist die Einführung eines neues Koordinatensystems notwendig, da bisher Welt- und Objektkoordinatensystem identisch sind. Außerdem wird der Roboter bei einer zeitlich lang andauernden Bewegung in Situationen geraten, an denen ein Planungsverhalten notwendig wird, das auf große Odometriefehler reagieren kann (siehe auch obiger Abschnitt). Dies könnte z.B. ein Verhalten sein, das die Selbstlokalisation an der gleichen Position mit unterschiedlichen Orientierungen mehrfach ausführt, solange bis die Lokalisation erfolgreich ist.

Eine schneller durchzuführende Erweiterung ist das Heben und Senken des Kamerakopfes zur besseren Zentrierung der Landmarke. Diese zusätzliche Transformation der Kamera unterscheidet sich lediglich von der Rotation des Roboters um das Rotationszentrum und die Rotationsachse (siehe auch Abschnitt 5.2.2 Anpassung der Odometriedaten an die Projektionsmatrix). Zu bemerken ist jedoch, dass ein Heben und Senken des Kamerakopfes eine weitere, fehlerbehaftete, zu korrigierende Bewegung ist, analog zur Rotation des Rumpfes. Außerdem wäre die relative Transformation des Modells nicht mehr nur in der Ebene, sodass eine vollständige 3D-Pose geschätzt werden muß.

Zur Poseschätzung wird der Abstand zwischen 3D-projektiven Ebenen, die aus den extrahierten Bildgeraden mittels der Projektionsmatrix gewonnen werden, und 3D-

Punkten des Modells minimiert. Benutzt man mehrere Kameras, so können die Zwangsbedingungen für projektive Ebenen von mehreren Kameraansichten in ein Gleichungsystem integriert werden. Zusätzlich muß der Matchingalgorithmus angepaßt werden, damit die richtigen Korrespondenzen zwischen Modellentitäten und Bildentitäten aus verschiedenen Bildern gefunden werden.

Zur Vereinfachung des Korrespondenzproblems und der Beschleunigung des Matching können spezielle Eigenschaften der Landmarken ausgenutzt werden. In [6] werden z.B. Übergänge von hellen nach dunklen Flächen ausgenutzt, um die möglichen Korrespondenzen für bestimmte Modellinien zu reduzieren. Eine weitere Reduzierung des Suchbereichs möglicher Korrespondenzen kann erreicht werden durch die Ausnutzung der topologischen Eigenschaften eines Modells [30]. Dies können z.B. parallele Eigenschaften bestimmter Linien sein oder Ecken, an denen sich mehrere Linien schneiden.

Langfristig wären zusätzliche Navigationsverfahren zur Unterstützung der Positionsbestimmung sehr nützlich. In der Schicht Navigation können kooperative Lokalisationsverhalten koordiniert werden. Z.B. werden in [50] Sonarsensoren zur Navigation benutzt. Der Laserentfernungsmesser könnte zur Bestimmung der initialen Position herangezogen werden, da er sehr genaue Abstandswerte liefert.

Abschließend lässt sich sagen, dass es in dieser Arbeit gelungen ist, ein visuelles Navigationssystem in einer verhaltensbasierten Architektur zu implementieren und ausführlich zu testen. Zur Navigation wurde ein Lokalisationsverfahren entwickelt, das auf der Positionsverfolgung basiert und den Odometriefehler als relative Modelltransformation bestimmt, ohne explizit Kameraparameter zu verwenden.

Anhang A

A.1 Kalibrierung der Vergenzeinheit

Zur genau parallelen Ausrichtung der Kameras wird ein roter Kalibrierstab verwendet, der in der Mitte zwischen den Kameras montiert ist (siehe Markierung in Abbildung A.1). Es wird angenommen, dass das optische Zentrum der Kameras in der Mitte des Bildes liegt. Deshalb kann ein Regelungsansatz verwendet werden, um den Stab in die Mitte der Bilder zu rücken. Dazu wird der rote Anteil der Bilder zunächst segmentiert und die Position des segmentierten Bereichs wird bestimmt. Dann werden die Kameras abhängig von der Differenz so bewegt, dass die nächste Abweichung geringer ist. Befindet sich der segmentierte Bereich in der Mitte des Bildes, so werden die Kameras abschließend um 90° rotiert, damit sie parallel stehen.

A.1.1 Segmentierung

Die RGB-Werte (Rot, Grün, Blau) des Farbbildes werden zunächst in den HSI Raum (Hue, Saturation, Intensity) transformiert. In diesem Farbraum lässt sich der zu segmentierende Farbbereich genau angeben, weil die Farbe in dem Hue-Wert kodiert ist und theoretisch unabhängig von der Beleuchtungsstärke (Intensity) und Sättigung (Saturation) der Farbe ist. Die Segmentierung geschieht dann durch Löschen (Grauwert null) aller Pixel, deren

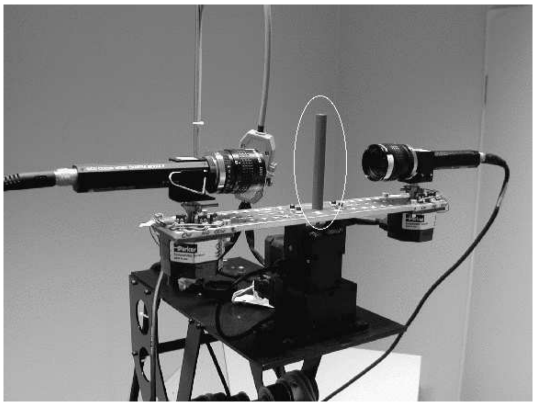

Abbildung A.1: Der Kamerakopf mit Kalibrierstab

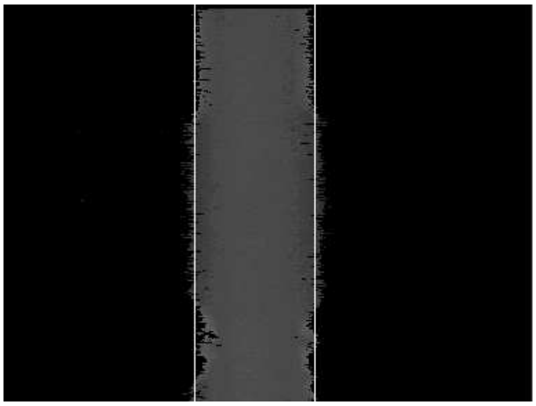

Abbildung A.2: Das segmentierte Bild des Kalibrierstabes

Farbwert nicht im angegebenen Bereich liegt oder deren Intensität und Sättigung unter einem Minimalwert liegt. Im Folgenden werden die nicht gelöschten Pixel als segmentierte Pixel bezeichnet.

A.1.2 Algorithmus zum Finden des Randes

Gesucht ist die Spaltenposition des linken und des rechten Randes im Bild.
Annahmen:

- Der Rand ist vertikal im Bild, weil die Kameras nicht um die optische Achse rotieren können und der Kopf nicht nach unten oder oben geneigt ist.
- Es sind keine weiteren roten Objekte im Hintergrund, sodass Pixel ungleich null ausserhalb des segmentierten Bereichs als Rauschen zu betrachten sind.

Algorithmus:

i. **Finden der Segmentmitte** durch Bilden des Durchschnitts der Spaltenposition aller segmentierten Pixel.

ii. **Linker Rand**: Ausgehend von der Mitte wird ein Marker so weit nach links verschoben, bis ein Fehler minimal wird. Jeder Pixel, der auf der falschen Seite der Marke steht, erhöht den Fehler um eins. Für segmentierte Pixel ist links vom Marker die falsche Seite und für die übrigen Pixel rechts von der Marke.

iii. **Rechter Rand**: Analog zum linken Rand mit Vertauschung von links und rechts bei der Fehlerberechnung.

Um beurteilen zu können, wie sicher die Bestimmung der Randpositionen ist, wird der Anteil der segmentierten Pixel, die sich zwischen den Rändern befinden, im Verhältnis zu der Gesamtanzahl aller segmentierten Pixel angegeben. Bei geeigneter Beleuchtung sind Werte um 90% erreichbar. Die zylindrische Form des Stabes erschwert eine exakte Segmentierung, weil am Rand des Stabes Glanzlichteffekte entstehen, die die Sättigung des Rot reduzieren, bzw. eine lokale Überbelichtung (weiß) erzeugen.

Literaturverzeichnis

[1] Ackermann M. Akkumulieren von Objektrepräsentationen im Wahrnehmungs–Handlungs Zyklus. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2000.

[2] Aristoteles (384-322 v. Chr.) *Aristoteles von O. Höffe* Taschenbuch, 2. Auflage, 1999

[3] Arkin R.C. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*, vol. 8, no. 4, pp. 92–112, 1989

[4] Arkin R. *Behaviour-Based Robotics.* MIT Press, Cambridge, Massachusetts, 1998

[5] Barron J.L. and Fleet D.J. and Beauchemin S.S. and Burkitt, T.A. Performance Of Optical Flow Techniques. *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, 1994.

[6] Beveridge J.R. Local search algorithms for geometric object recognition: Optimal correspondence and pose. *Ph.D. Thesis, Computer Science Department, University of Massachusetts, Amherst*, 1993. Available as Technical Report CS 93–5.

[7] Brooks R.A. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14–23, March 1986.

[8] Brooks R.A. Intelligence Without Reason. *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Sydney, Australia, pp. 569–595, 1991

[9] Brooks R.A. From earwigs to humans. *Robotics and Autonomous Systems*, vol. 20, no. 2-4, 1997.

[10] Bunten A. Ein autonomes Robotersystem zum Folgen einer Person. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2001.

[11] Burgard W. and Cremers A.B. and Fox D. and Hähnel D. and Lakemeyer G. and Schulz D. and Steiner W. and Thrun S. Experiences with an interactive tour-guide robot. Technical Report CMU-CS-98-139, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 1998

[12] Christensen H. and Kirkeby N. Model-driven vision for in-door navigation. *Robotics and Autonomous Systems*, vol. 12, pp. 199–207, 1994

[13] Curry D. *UNIX Systems Programming.* O 'Reilly Associates, 1996

[14] Daniilidis K. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research*, vol. 18, no.3, pp. 286–298, 1999.

[15] Faugeras O. *Three-Dimensional Computer Vision, A Geometric Viewpoint.* Mit Press, Cambridge, 1993

[16] Felsberg M. and Sommer G. Structure multivector for local analysis of images. *In R.Klette, Th. Huang, G.Gimmel'farb (eds.), Multi-Image Search and Analysis*, LNCS, Springer, pp. 93-104, 2001.

[17] Foley J. and van Dam A. and Feiner S. and Hughes J. *Computer Graphics Principles and Practice, 2nd ed. in C.* Addison-Wesley Publishing Company, Reading, Mass., 1996

[18] Fox D. and Burgard W. and Thrun, S. The Dynamic Window Approach to Collision Avoidance. *Technical Report*, IAI-TR-95-13, University of Bonn,1995

[19] Gallier J. *Geometric Methods and Applications for Computer Science and Engineering.* Springer Verlag, New York Inc. 2001

[20] Granert O. Poseschätzung Kinematischer Ketten. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2001.

[21] Hailu G. and Sommer G. Embedding knowledge in reinforcement learning. *International Conference on Artificial Neural Networks (ICANN)*, pages 1133–1138, Skovde, Sweden, 1998

[22] Hopcroft J. and Ullmann J. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, 1979

[23] Hoppen P. *Autonome mobile Roboter.* B.I. Wissenschaftsverlag, Mannheim, 1992

[24] Hough P.V.C. Methods and means for recognizing complex patterns. *U.S. Patent 3,069,654, Dec. 18*, 1962.

[25] Kernighan B.W. and Ritchie D.M. *The C Programming Language, Second Edition.* Prentice Hall, Inc., 1988

[26] KiViGraP (Homepage of the Kieler Vision and Grasping Project). http://www.ks.informatik.uni-kiel.de/~kivi/kivi.html.

[27] Kosaka A. and Kak A.C. Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties. *CVGIP: Image Understanding*, Vol. 56, No. 3, November, pp. 271–329, 1992.

[28] Kuipers J. *Quaternions and Rotation Sequences.* Princeton University Press, 1999.

[29] Kunze S. Ein Hand-Auge-System zur visuell basierten Lokalisierung und Identifikation von Objekten. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2000.

[30] Lanser S. and Zierl Ch. On The Use of Topological Constraints within Object Recognition Tasks. *In Proceedings of 13th ICPR*, volume 1, pp. 580–584, 1996

[31] Lorenz, F. *Lineare Algebra I+II.* BI Wissenschaftsverlag, 3. Auflage, 1992

[32] Matrox Imaging Systems. http://www.matrox.com/imaging/home.htm

[33] Lanser S. and Zierl C. and Munkelt O. and Radig B. MORAL-A Vision-Based Object Recognition System for Autonomous Mobile Systems. *Computer Analysis of Images and Patterns (CAIP) Sommer G., Daniilidis K., Pauli J. (Eds)*, pp. 33–41, Kiel 1997.

[34] Moravec Hans P. Sensor fusion in certainty grids for mobile robots. *Sensor Devices and Systems for Robotics, ed. Alicia Casals, Springer Verlag*, NATO Advanced Research Workshop, Catalonia, Spain, October 13-16, 1987

[35] Murray R.M. and Li Z. and Sastry S.S. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[36] o'Gorman L. and Sanderson A.C. The converging squares algorithm: an efficient method for locating peaks in multidimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 6, No. 3, pp. 280–288, 1984

[37] Pauli J. Geometric/photometric consensus and regular shape quasi-invariants for object localization and boundary extraction. *Technical Report 9805, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 1998.

[38] Pauli J. Development of Camera-Equipped Robot Systems. *Technical Report 9904,Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik*, 2000.

[39] Pauli J. *Learning-Based Robot Vision, Principles and Applications.* Springer Verlag, 2001

[40] Press W.H. and Teukolsky S.A. and Vetterling W.T. and Flannery B.P. *Numerical Recipes in C.* Cambridge University Press, 1994.

[41] Horaud R. Phong T.Q. and Tao P.D. Object pose from 2-d to 3-d point and line correspondences. *International Journal of Computer Vision*, vol. 15, pp. 225–243, 1995.

[42] Princen J.and Illingworth J. and Kittler J. An optimizing line finder using a Hough transform algorithm. *Computer Vision, Graphics, and Image Processing*, vol. 52, pp. 57–77, 1990.

[43] Riseman E. and Hanson A. and Beveridge J. and Kumar R. and Sawhney H. Landmark-based navigation and the acquisition of environmental models. *Visual Navigation: From Biological Systems to Unmanned Ground Vehicles*, Yiannis Aloimonos, Ed., Lawrence Erlbaum Associates, Inc., pp. 317–374, 1997

[44] Rosenhahn B. and Krüger N. and Rabsch T. and Sommer G. Tracking with a Novel Pose estimation algorithm. *International Workshop on Robot Vision; Auckland, New Zealand,(Eds.), R. Klette, G. Sommer, S. Peleg,Lecture Notes in Computer Science,* vol. 1998, pp. 9–19,2001 .

[45] Rosenhahn B. and Granert O. and Sommer G. Monocular pose estimation of kinematic chains. In *Applied Geometric Algebras for Computer Science and Engineering, Birkhäuser Verlag*, 2001. (to appear)

[46] Rosenhahn B. and Zhang Y. and Sommer G. Pose estimation in the language of kinematics. *Second International Workshop, Algebraic Frames for the Perception-Action Cycle, AFPAC 2000, LNCS 1888*, Springer-Verlag, pp. 284–293, 2000.

[47] iRobot: Real Robot Division Homepage. http://www.irobot.com/rwi/

[48] Sommer G. The global algebraic frame of the perception-action cycle. *Handbook of Computer Vision and Applications, vol. 3*, Academic Press, San Diego, pp. 221–264, 1999

[49] Sommer G. and Rosenhahn B. and Zhang Y. Pose estimation using geometric constraints. *In R.Klette, Th. Huang, G.Gimmel'farb (eds.), Multi-Image Search and Analysis*, LNCS, Springer, pp. 153–170, 2001.

[50] Stark H. Zielgerichtete Navigation und Kollisionsvermeidung eines mobilen Roboters. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 1998.

[51] Fedor Ch. TCX: An interprocess communication system for building robotic architectures, Programmer's Guide to version 10.xx. *CMU, Robotics Institute*, 1994

[52] Thrun S. and A. Bucken W. and Burgard D. and Fox T. and Frohlinghaus D. and Henning and T. Hofmann and M. Krell and T. Schmidt Map learning and high-speed navigation in RHINO *AI-Based Mobile Robots: Case Studies of Successful Robot Systems*, Boston, Mass., MIT Press, 1998

[53] Thrun S. and Fox D. and Burgard D. and Dellaert F. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence Journal*, vol. 128, no. 1-2, pp. 99–141, 2001

[54] Tsotsos J. Behaviorist Intelligence and the Scaling Problem. *Artificial Intelligence*, volume 75, pp. 135-160, 1995

[55] Wei Guo-Qing and Song De Ma Implicit and explicit camera calibration: Theory and experiments. *IEEE trans. on Pattern Analysis and Machine intelligence*,vol. 16, no. 5, pp. 469–480, May 1994.

[56] Wendtorf D. Integration visuell-aktorischer Kompetenzen im Wahrnehmungs-Handlungs-Zyklus. *Diplomarbeit, Lehrstuhl für Kognitive Systeme der Christian-Albrechts-Universität zu Kiel*, 2001.

Danksagung

Hiermit möchte ich mich bei all denen bedanken, die mich während meiner Diplomarbeit unterstützt haben.

Besonderer Dank gilt *Bodo Rosenhahn*, der mich während des gesamten Zeitraumes sehr engagiert betreut hat und immer für Diskussionen Zeit hatte.

Bei *Oliver Granert* bedanke ich mich für seine geleistete Arbeit zur *Poseschätzung* und für die Zusammenarbeit im Projekt *KiviGrap*, das durch die Anstrengungen von *Dr. Norbert Krüger* zustande gekommen ist.

Meinen Eltern möchte ich ganz herzlich danken für die langjährige Unterstützung während meines Studiums und besonders während meiner Diplomarbeit.

Bedanken möchte ich mich auch bei *Maja Kalden* und *Anett Grest-Wufka* für das Korrigieren der Arbeit.

Ganz besonderer Dank geht an meine Freundin *Anja Wilms* für ihre Unterstützung und Hilfe.

Hiermit versichere ich, dass ich die Arbeit selbstständig verfasst und ausschließlich die angegebenen Hilfsmittel und Quellen verwendet habe.

Kiel 18. Dezember 2001