Diplomarbeit

im Fach **Naturwissenschaftliche Informatik**

Universität Bielefeld
Technische Fakultät

Untersuchung von Algorithmen zur visuellen Roboterlokalisierung

vorgelegt von

Christian Lange
(Matrikelnummer: 1109548)

November 2000

Betreuung:

Dr. J. Zhang

Prof. Dr. A. Knoll

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Bielefeld, den 10. November 2000

Christian Lange

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1

Einführung und Aufgabenstellung

1.1 Motivation

Roboter könnten viele einfache Tätigkeiten ausführen, wenn sie sich in unserer alltäglichen Umgebung zurechtfänden. Reinigungsarbeiten, Museumsführungen oder Botendienste wie Austeilen von Briefen oder Speisen sind typische Beispiele hierfür. Maschinen, die einige dieser Tätigkeiten ausführen können, gibt es zwar, aber sie sind bisher nicht im Stande, sich selbstständig zu orientieren.

Orientieren bedeutet, sich in evtl. unbekannter Umgebung zurechtzufinden. Dies ist nötig, damit der Roboter

- einen Einsatzort aufsuchen,
- zu seinem Depot zurückkehren,
- einen Weg um Hindernisse herum finden,
- Menschen ausweichen und
- neue Räume erkunden kann.

Um den Weg zu einem bestimmten Ziel zu finden, wird eine **Karte** der Umgebung benötigt. Dies muss nicht notwendigerweise eine metrische Karte (ein Grundriss) sein. Erforderlich ist nur, dass in ihr sowohl die Informationen, anhand derer sich die verschiedenen Positionen von einander unterscheiden lassen, als auch die relative Lage der Positionen zueinander, abgespeichert werden. Die

Karte sollte idealerweise automatisch erstellt werden, da dies manuell zu aufwendig und fehleranfällig ist. Vorteilhaft wäre, wenn die enthaltenen Informationen in einem interpretierbaren Format ausgegeben werden könnten. Der Algorithmus, der mit Hilfe der Sensordaten die Position in der Karte bestimmen kann, wird im Folgenden *Lokalisationsalgorithmus* genannt.

In Räumen, die robotergerecht gestaltet sind – wie Fabrikhallen – bewegen sich autonome Fahrzeuge mit Hilfe von Markierungen, die speziell zu deren Orientierung angebracht wurden. Wenn Roboter in den oben genannten alltäglichen Bereichen eingesetzt werden sollen, ist eine andere Lösung erforderlich, denn das Installieren von Markierungen in allen Räumen einer Wohnung oder eines Bürogebäudes ist zu aufwendig und wenig ästhetisch. Hier sollten die Maschinen einer menschenfreundlichen Umgebung angepasst werden und nicht umgekehrt.

In Räumen, die auch oder überwiegend von Menschen genutzt werden, gibt es noch eine weitere Schwierigkeit: Der Roboter sollte die Menschen und andere bewegliche Hindernisse (z.B. Stühle) beachten. Diese können nicht in eine Karte eingezeichnet werden, sondern müssen während der Fahrt von den Sensoren bemerkt werden. Die Umgebung mit Sensoren zu untersuchen ist auch dann erforderlich, wenn der Roboter eine neue Karte eines unbekannten Raumes erstellen soll.

1.2 Anforderungen

Aus den genannten Punkten ergeben sich die folgenden Anforderungen an einen autonomen, mobilen Roboter:

- Er soll minimales Vorwissen über die Umgebung benötigen.
- Er soll unbekannte Räume eigenständig erkunden.
- Er soll möglichst viele Räume lernen können.
- Er soll auch nach totaler Desorientierung seinen Standort bestimmen können.
- Er erfordert keine Veränderung der Umgebung durch z.B. Markierungen.
- Er soll robust gegenüber Störungen wie verschobenen Stühlen, geschlossenen Türen, usw. sein.
- Zu seiner Bedienung soll kein Experte erforderlich sein.

Ein Robotersystem, dass diesen Anforderungen gerecht werden soll, muss Daten über seine Umgebung sammeln, auswerten und speichern. Hierbei muss darauf

geachtet werden, Daten zu verwenden, die eine eindeutige und effiziente Bestimmung der Position des Roboters und der Hindernisse in seiner Umgebung ermöglichen. Das Datenformat muss dabei so gewählt werden, dass sie – auch wenn der Roboter ein großes Gebiet kennt – in seinem begrenzten Speicher untergebracht werden können.

Zum Sammeln der Daten benötigt der Roboter geeignete **Sensoren**, die Signale aus der Umgebung des Roboters aufnehmen. Neben Bildern können hier auch Laser, Infrarotlicht oder Ultraschall verwendet werden. Die letzten drei arbeiten allerdings meist nur mit einzelnen Strahlen und können somit nur sehr kleine Ausschnitte ihrer Umgebung wahrnehmen. Sie übersehen deshalb z.B. leicht Stühle oder Tische, da diese im unteren Bereich oft nur dünne Beine haben. Außerdem werden z.B. Ultraschallwellen von Textilien, oder Infrarotlicht von schwarzen Flächen nicht reflektiert.

In dieser Arbeit sollen daher Bilder als Informationsquelle dienen. In einem Bild sind Informationen über einen größeren Umgebungsausschnitt enthalten, besonders, wenn zur Kamera eine Linse oder ein Spiegel hinzugefügt wird, sodass sie ein Bild ihrer kompletten Umgebung aufnimmt. So ein Aufbau wird *omnidirektionales Sichtsystem* genannt (siehe Abbildung 1.1).

Abbildung 1.1: Omnidirektionales Sichtsystem mit hyperbolischem Spiegel

Um aus den omnidirektionalen Bildern die zur Lokalisation nötigen Informationen zu gewinnen, müssen diese Bilder verarbeitet werden. Es müssen Merkmale, anhand derer Positionen wiedererkannt werden können, in ihnen gefunden und aus ihnen extrahiert werden.

1.3 Aufgabenstellung und Gliederung

Diese Diplomarbeit beschäftigt sich damit, wie gut verschiedene Algorithmen geeignet sind, um lokalisationsrelevante Daten aus omnidirektionalen Bildern zu extrahieren. Dazu werden verschiedene Methoden der Bildverarbeitung auf mehrere Serien von omnidirektionalen Testbildern angewandt und die dabei gewonnenen Merkmale miteinander verglichen.

In Kapitel 2 werden zunächst einige vorhandene Arbeiten aus dem Gebiet der visuellen Lokalisation vorgestellt. Die Systeme erlauben aber meist nur eine eingeschränkte Bewertung der dort verwandten Algorithmen.

Das Gesamtsystem, in dessen Rahmen diese Arbeit entsteht und in dem die Ergebnisse dieser Arbeit angewendet werden sollen, wird in Kapitel 3 beschrieben.

Die folgenden drei Kapitel erklären verschiedene Merkmale und zeigen, mit welchen Algorithmen diese berechnet werden, wobei Kapitel 4 verschiedene Möglichkeiten der Helligkeitsnormierung vergleicht, im 5. Kapitel verschiedene Möglichkeiten, Farbinformationen zu nutzen, beschrieben werden, und Kapitel 6 Algorithmen zur Komprimierung der Daten vergleicht.

Untersuchungsmethoden zur Bewertung der Lokalisationstauglichkeit der Merkmale werden in Kapitel 7 erörtert, und das 8. Kapitel dient der Beschreibung ihrer Anwendung und der dabei ermittelten Ergebnisse. Eine Zusammenfassung aller Ergebnisse und ein Ausblick darauf, welche weiteren Untersuchungen gemacht werden sollten, wird in Kapitel 9 gegeben.

2

Stand der Technik

Nachfolgend werden einige bisher existierende Arbeiten vorgestellt, die sich mit visueller Lokalisation beschäftigen. Diese sind so ausgewählt, dass möglichst viele wichtige Ansätze und Verfahren aktueller Forschung erwähnt werden. Die von den jeweiligen Autoren gemachten Angaben über die Schwierigkeiten der Systeme sind schlecht miteinander vergleichbar und daher nicht aufgeführt. Erfolgreich erscheinende Methoden werden in den anderen Kapiteln dieser Arbeit näher untersucht.

2.1 Vogelperspektive für freien Fußboden

S. Wei, Y. Yagi und M. Yachida verwenden in [WYY98] ein omnidirektionales Sichtsystem mit einem hyperbolischen Spiegel und zusätzlich einen Ring aus Ultraschallsensoren. Ihr Roboter soll daraus den freien Fußboden um sich herum bestimmen. Das Kamerabild wird hierzu in einen Blick aus der Vogelperspektive über dem Roboter transformiert (nach einem Algorithmus aus [YYY95]). Dieses als *Fußbodenkarte* (engl. floormap) bezeichnete Bild ist allerdings keine wirkliche Vogelperspektive, sondern entspricht dieser nur für den ebenen Boden um den Roboter herum und nur, wenn die begrenzenden Hindernisse auf dem Boden stehen.

Die Fußbodenkarte verwenden die Autoren zweifach: Erstens wird ein Kantenbild berechnet, und zweitens werden die Farben segmentiert. Zur Berechnung des Kantenbildes kommt ein Laplace-Operator zum Einsatz, dessen Ausgabe mittels eines Grenzwertes und eines Nachbarschaftsoperators geglättet wird. Anschlie-

ßend wird durch Dilatation und Erosion ein besserer Zusammenhang der Kantenpunkte erreicht.

Die Farbsegmentierung geschieht im (L,a,b)-Farbraum mit dem K-mean-Algorithmus, wobei in jeder Radiallinie des Bildes drei Farbcluster gesucht werden. Das System geht dann davon aus, dass das innere Cluster zum Roboter selbst gehört, das mittlere Cluster der freie Fußboden ist und das äußere Cluster einem Hindernis entstammt. Als Initialwerte des K-mean-Algorithmus verwendet das System den hellsten Punkt, den dunkelsten Punkt und den Durchschnitt der Punkte, die laut Ultraschallkarte zum Fußboden gehören.

Der Roboter erhält somit drei Karten seiner Umgebung (Ultraschall, Kanten und Farbregionen), aus denen er den freien Fußboden ermittelt.

Dieses Verfahren ermittelt zwar nicht die Position des Roboters, aber zur Exploration unbekannter Räume kann es sehr nützlich sein.

2.2 Histogrammvergleich

I. Ulrich und I. Nourbakhsh verwenden in [UN00] ein omnidirektionales Sichtsystem als einzigen Sensor. Verschiedene Positionen werden mit Hilfe von Farbhistogrammen unterschieden.

Dazu nimmt das System während einer Trainingsphase von jeder möglichen Position ein Bild auf und berechnet zu jedem sechs Histogramme. Zunächst die für den Rot-, Grün-, und Blaukanal und nach einer Transformation in den HLS-Farbraum auch die für Farbwert, Helligkeit und Sättigung. Anstelle der Bilder werden anschließend nur deren Histogramme verwendet.

Um ein Testbild zu klassifizieren, berechnet das System zunächst die sechs neuen Histogramme und dann deren Abstände zu den gelernten Histogrammen. Da die Autoren davon ausgehen, dass die bisherige Position bekannt ist, vergleichen sie nur mit deren Histogramm und mit denen der Nachbarpositionen. Als Abstandsmaß wird die Jeffrey-Divergenz benutzt, die von Y. Rubner u.a. in [RTG98] erläutert wird.

Mehrere Experimente der Autoren zeigen die fehlerfreie Klassifizierung von bis zu zehn Räumen. Die Bestimmung der genauen Position innerhalb eines Raumes ist nicht Gegenstand ihrer Arbeit. Ein weiterer Nachteil dieses Systems ist, dass die Nachbarschaftsbeziehungen der Positionen von Hand modelliert werden müssen.

2.3 Output Relevant Features

V. Schwert verwendet in [Sch98] ein Kompressionsverfahren, mit dem aus den Eingangsbildern direkt lokalisationsrelevante Komponenten extrahiert werden.

Das System erstellt sich klassifizierte Lerndaten, indem es durch den zu lernenden Raum fährt und in gleichmäßigen Abständen Bilder aufnimmt. Dabei vermeidet es Kollisionen mit Hindernissen mit Hilfe seiner Infrarotsensoren. Die Bestimmung der jeweiligen Aufnahmeposition erfolgt hierbei ausschließlich durch interne Sensoren (Radencoder) des Roboters.

Die von einem konischen Spiegel erzeugten, etwas unscharfen, omnidirektionalen Bilder werden zunächst helligkeitsnormiert (siehe Kapitel 4.1.1). Dann lernt das System einen Unterraum des Bildvektorraumes, dessen Basisvektoren in die Richtungen zeigen, in denen sich die Bilder ändern, wenn sich die Position ändert. Diese Basisvektoren nennt er *Output relevant Features*. Mit einem B-Spline-Fuzzy-Controller werden dann den in diesen Unterraum projizierten Bildern die Aufnahmepositionen zugeordnet.

Um die Aufnahmeposition eines neuen Bildes zu bestimmen, muss dieses ebenfalls energienormiert und in den Unterraum projiziert werden. Der Fuzzy-Controller kann dann sogar Positionen bestimmen, die zwischen denen der Lernbilder liegen.

Um geeignete *Output relevant Features* zu finden, dürfen die Eingangsbilder sich nicht zu stark voneinander unterscheiden. Daher teilt Schwert den zu lernenden Raum in mehrere Abschnitte (so genannte *Situationen*) ein.

Da das System aussehensbasiert ist, reagiert es empfindlich auf Veränderungen der Beleuchtung oder Objekte, die Teile der Szene verdecken. Die oben erwähnte Energienormierung soll Beleuchtungsänderungen unterdrücken, und, damit störende Objekte nicht das Gesamtsystem beeinträchtigen, teilt Schwert das omnidirektionale Bild in mehrere Sektoren ein, die getrennt verarbeitet werden.

2.4 Kanten mit omnidirektionalem Stereosichtsystem

In [DDPC99] wird von C. Drocourt, C. Pegard und anderen ein Roboter beschrieben, der an jeder Position zwei omnidirektionale Bilder aufnimmt. Eine Kamera mit einem konischen Spiegel wird dazu entlang des Roboters bewegt und nimmt so Bilder an zwei unterschiedlichen Positionen auf.

Das System extrahiert aus den Bildern eine Grauwertkurve (einen Grauwert pro Radiallinie) und daraus die vertikalen Kanten der Umgebung. Dann muss entschieden werden, welche Bereiche der jeweils zwei Bilder zusammengehören.

Dazu verknüpft das System verschiedene Kriterien probabilistisch. Eines davon ist die Übereinstimmung der Position der gefundenen Kanten. Ein weiteres ist die Übereinstimmung der mittleren Grauwerte der zwischen den Kanten liegenden Bereiche, und das dritte Kriterium sind geometrische Bedingungen, die sich aus der Lage der zwei Aufnahmepositionen ergeben.

Im Experiment der Autoren bestimmt das System seine Position in einem von Hand modellierten Raum recht genau und robust. Das eigenständige Erstellen eines Modells ist hingegen nicht so genau, und zur Lokalisation in einer größeren Umgebung werden keine Angaben gemacht.

Die mechanische Bewegung der Kamera ist unpraktisch, da sie Zeit benötigt, Ungenauigkeiten verursacht und evtl. Wartung erfordert. Die Installation von zwei omnidirektionalen Sichtsystemen ist aber keine Alternative, denn diese würden sich gegenseitig sehen und wären außerdem teurer.

2.5 Aufmerksamkeitsbereiche als Landmarken

Sowohl M. Knapek und andere ([KOK00]) als auch R. Sim und G. Dudek ([SD99]) beschreiben Verfahren, bei denen „interessante" Ausschnitte der Bilder verwendet werden, um eine Position wiederzuerkennen.

Während der Lernphase wählt das System mit Hilfe eines Eckendetektors Bildbereiche aus, in denen besonders viele Ecken liegen. Solche Bereiche, von denen es bis zu 30 pro Bild gibt, werden als *Landmarkenkandidaten* bezeichnet. Aus den Kandidaten einer kompletten Bildserie verwendet das System dann nur diejenigen, die eindeutig sind (von allen anderen unterscheidbar) zur späteren Lokalisation.

Bei der Lokalisation werden aus dem aufgenommenen Bild die Landmarkenkandidaten extrahiert und mit den zuvor gelernten verglichen. Mit jeder wiedergefundenen Landmarke erhält das System eine Positionsschätzung und aus diesen mittelt es dann die Position des Roboters.

Die oben genannten Autoren verwenden dieses Prinzip auf perspektivischen Bildern je eines Raumes. S. Bögershausen erweitert in [Bög00] den Ansatz auf omnidirektionale Bilder und schlägt andere Aufmerksamkeitsoperatoren vor (z.B. Farbsättigung). In dieser Arbeit befindet sich auch eine ausführlichere Beschreibung der Vorgehensweise.

2.6 Fazit

Die verschiedenen vorgestellten visuellen Lokalisationssysteme sind bei der Lösung der jeweiligen Aufgabe recht erfolgreich, aber keines erfüllt sowohl die Voraussetzung, ein größeres dynamisches Einsatzgebiet autonom zu erkunden, als auch die Forderung, seine Position darin eindeutig und genau zu bestimmen.

Für die Entwicklung eines Systems, das alle Anforderungen aus Kapitel 1 erfüllt, können evtl. einige der vorgestellten Algorithmen verwendet oder mit neuen kombiniert werden. Um entscheiden zu können, welche der Algorithmen und Merkmale besonders geeignet sind, müssen sie unter den gleichen Bedingungen getestet werden. Dieses geschieht in Kapitel 7 dieser Arbeit, denn die Ergebnisse aus den unterschiedlichen Testszenarien der jeweiligen Autoren nützen hierzu wenig.

3

Systemaufbau

Diese Arbeit ist Teil eines Projektes, in dem für einen mobilen Roboter selbstständige Orientierung und Navigierung entwickelt werden. Die Arbeit [Bög00] von S. Bögershausen entstand ebenfalls im Rahmen dieses Projektes. Sie behandelt ergänzende Fragestellungen mit Hilfe des gleichen Versuchsaufbaus.

3.1 Hardware

Der verwendete Roboter ist mit einem omnidirektionalen Sichtsystem und Ultraschallsensoren ausgestattet (siehe Abb. 3.2). Da die Kamera durch ihre höhere Auflösung wesentlich mehr Informationen zur Verfügung stellt, ist sie der wichtigste Sensor.

Das Sichtsystem besteht aus einer Kamera, deren Objektiv senkrecht nach oben ausgerichtet ist, und einem hyperbolischen Spiegel, dessen Rotationsachse mit der optischen Achse der Kamera zusammenfällt (siehe Abb. 3.1). Eine ausführliche Beschreibung sowie Untersuchungen verschiedener Spiegel und Kameras befinden sich in [Bög00].

3.2 Software

Abbildung 3.3 stellt die Bestandteile des Systems und deren Zusammenspiel mit der Umgebung dar.

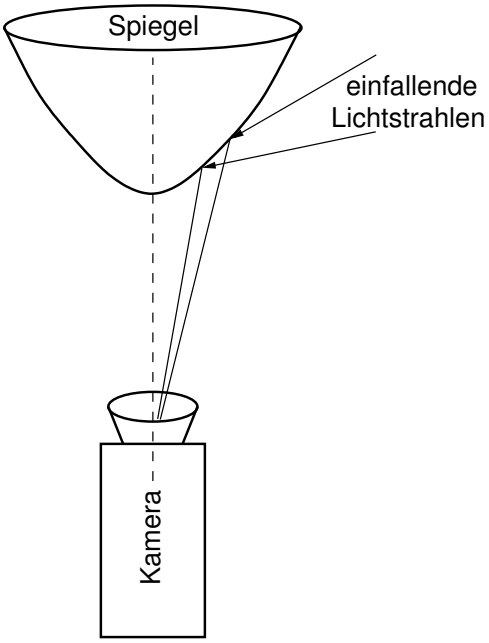

Abbildung 3.1: Anordnung von Kamera und Spiegel

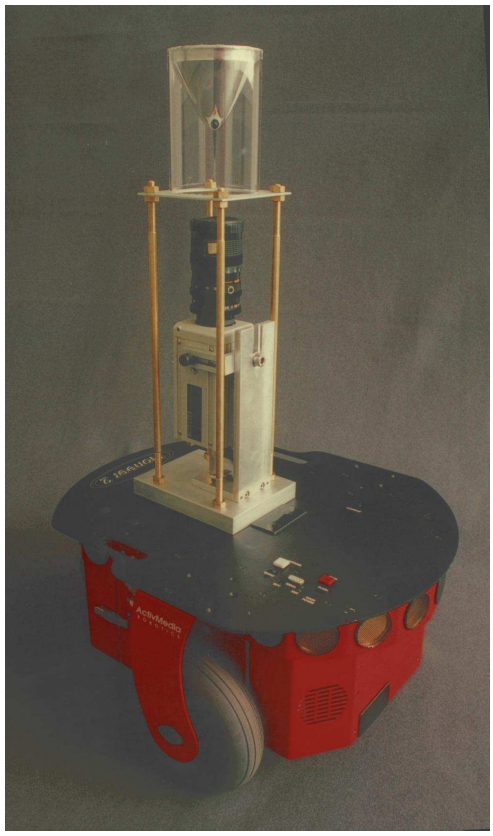

Abbildung 3.2: Pioneer 2 DX Roboter mit omnidirektionalem Sichtsystem und Ultraschallsensoren

Eine Kombination mehrerer Methoden der **Bildverarbeitung** wird benötigt, um die Kamerabilder aufzubereiten und um aus ihnen die lokalisationsrelevanten Informationen (auch *Merkmale* genannt) zu extrahieren. Verschiedene mögliche Merkmale werden in den Kapiteln 4, 5 und 6 vorgestellt.

Der **Positionsschätzer** versucht aus den Merkmalen und der gespeicherten Karte die Position des Roboters zu schätzen. Er besteht aus einem universellen Funktionsapproximator (z.B. Fuzzy-Controller).

Eine übergeordnete **Orientierungssteuerung** verwendet dann die geschätzte Position, um zu entscheiden, wie der Roboter sich verhält. Sie ist für die Verwaltung der Karte zuständig und stellt diese dem Positionsschätzer zur Verfügung. Falls bekannt ist, wo der Roboter sich bei der letzten Positionsschätzung befand, wird evtl. nur ein Ausschnitt der Karte zur neuen Schätzung vorgegeben, um den Such-

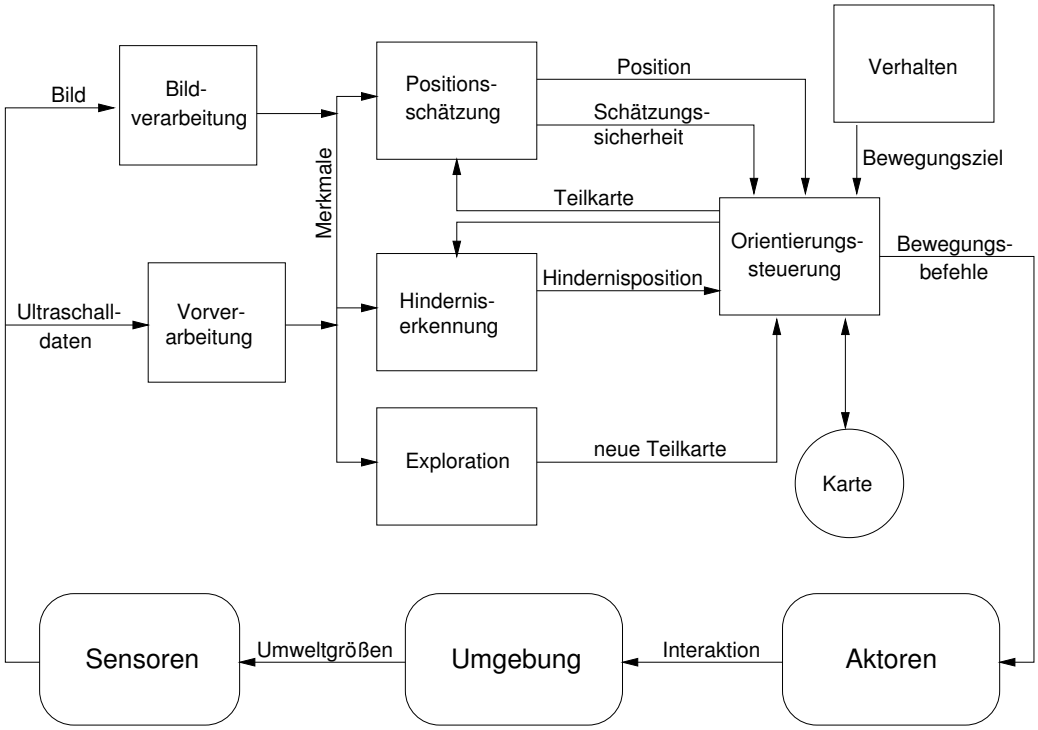

Abbildung 3.3: Bestandteile des Systems

raum des Positionsschätzers einzuschränken. Die Orientierungssteuerung ist auch für die Vermeidung von Zusammenstößen mit Hindernissen (sowohl in der Karte eingetragenen als auch nicht eingetragenen) verantwortlich.

Das Modul **Hinderniserkennung** liefert mit Hilfe der Merkmale der aktuellen Umgebung und den in der Teilkarte eingetragenen Merkmalen die Position möglicher Hindernisse.

Falls die aktuelle Umgebung zu keiner Teilkarte passt, muss die Orientierungssteuerung die Umgebung neu erkunden. Der Algorithmus zur **Exploration** teilt dazu die neuen Daten in Teilkarten ein, die später zur Lokalisierung verwendet werden.

Der **Verhaltensalgorithmus** ist allen anderen übergeordnet und verantwortlich für die Ausführung der Aufgabe des Roboters (Reinigen des Fußbodens, Austeilen der Post oder dgl.). Er liefert das Ziel für die jeweiligen Bewegungen. Außerdem muss er mit Hilfe anderer – nicht eingezeichneter – Module die Möglichkeit bieten, dem Roboter seine Aufgabe zu stellen.

Die Bewegungsbefehle der Orientierungssteuerung werden von den Aktoren (Motoren an den Rädern) umgesetzt und bewirken so eine Interaktion mit der Umgebung. Dadurch ändern sich die von den Sensoren aufgenommenen Umwelt-

größen, und der Roboter erhält neue Daten über seine Umwelt.

Die Untersuchungen zur Extraktion von Merkmalen aus Bildern in dieser Arbeit beziehen sich auf den hier vorgestellten Aufbau eines Systems zur Orientierung eines mobilen Roboters. Daher wird bei der Bewertung der Algorithmen Wert darauf gelegt, dass sie in Echtzeit arbeiten und mit dem Speicher des Roboters auskommen. Auch der Vergleich der Merkmale ist auf die Anforderungen dieses Systems ausgerichtet (siehe auch Kapitel 7).

4

Reduktion der Beleuchtungseinflüsse

4.1 Grundlagen

Von der Kamera aufgenommene Bilder des gleichen Objektes unterscheiden sich teilweise schon bezüglich ihrer Helligkeit sehr stark. Dafür gibt es mehrere Gründe:

- Das Objekt wird unterschiedlich stark beleuchtet.
- Das Objekt wird von verschiedenfarbigem Licht beleuchtet (z. B. Tageslicht oder Kunstlicht).
- Durch Lichtquellen oder Fenster im Bild wird die Gesamthelligkeit beeinflusst. Daraufhin passt die Kamera ihre Empfindlichkeit automatisch an, was zu einem veränderten Bild führt.

Insbesondere aussehensbasierte Verfahren werden durch diese Probleme stark beeinträchtigt. Um die Objekte und damit die Räume und Positionen leichter erkennen zu können, wäre es nützlich, diese Unterschiede zwischen den Bildern zu beseitigen oder zumindest zu reduzieren. Eine wesentliche Verbesserung wäre erreicht, wenn die Helligkeit der einzelnen Bilder angeglichen würde.

Um die Helligkeit darzustellen, eignen sich Y-Histogramme (Helligkeitskomponente des YUV-Farbraumes) und V-Histogramme (Helligkeitskomponente des HSV-Farbraumes). Die häufiger verwendeten Y-Histogramme werden auch als Grauwerthistogramme bezeichnet. Analog dazu wird der Begriff *Grauwert* als Synonym für Helligkeit verwandt. Da die untersuchte Büroumgebung kaum

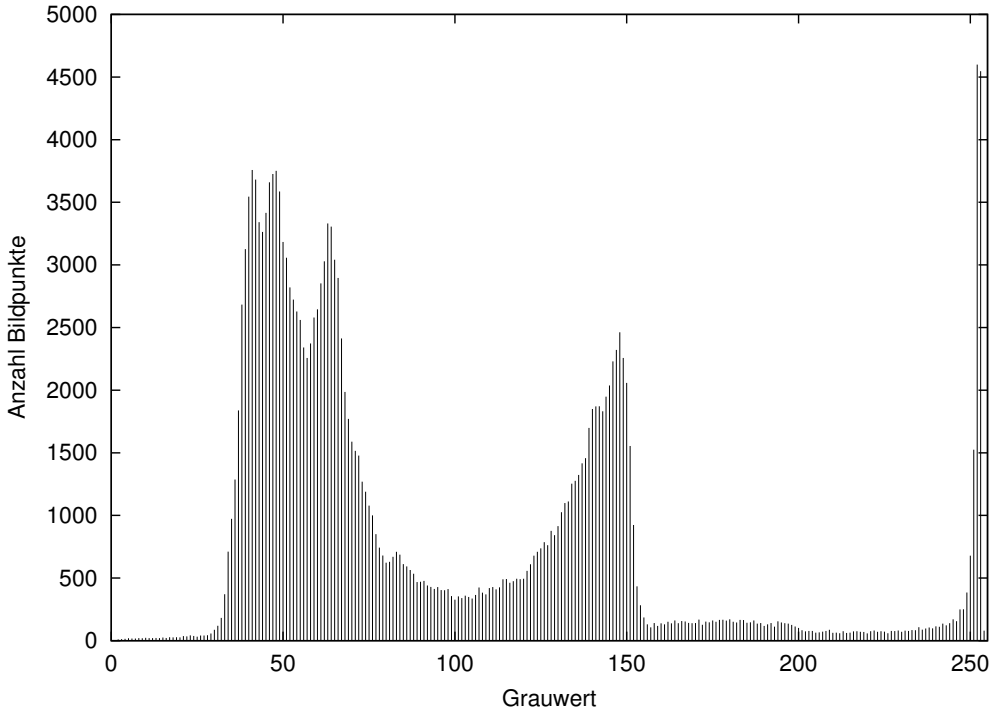

Abbildung 4.1: Grauwerthistogramm eines typischen Testbildes

gesättigte Farben enthält, sind zwischen Y- und V-Histogrammen kaum Unterschiede zu erkennen. Abbildung 4.1 zeigt ein typisches Grauwerthistogramm.

Auf der Abszisse wird die Helligkeit abgetragen und auf der Ordinate die Anzahl der Bildpunkte, die diese Helligkeit haben. Wenn im Bild viele Punkte nahezu die gleiche Farbe haben, ergibt sich im Histogramm an dieser Stelle ein Maximum. Viele der Flurbilder zeigen solche Maxima für den Fußboden und die Wand. In Abbildung 4.1 sind dies die Helligkeiten 50 und 150. Am rechten Rand sind als drittes Maximum die überbelichteten Punkte zu finden.

Wenn die gleiche Szene unter anderen Lichtverhältnissen aufgenommen wird, bewirkt das eine Verschiebung oder Verzerrung der Kurve.

Um die Bilder bzw. die darin enthaltenen Flächen besser vergleichen zu können, ist es wünschenswert, dass identische Flächen in unterschiedlichen Aufnahmen gleich aussehen, also die Maxima im Histogramm an den gleichen Stellen liegen. Dazu muss zumindest die Helligkeit der Bildpunkte angepasst werden. Bei einer Szene, die mit unterschiedlich gefärbter Beleuchtung fotografiert wurde, wäre die Angleichung der Helligkeit natürlich nicht ausreichend. Um die Helligkeitsschwankungen auszugleichen, werden verschiedene Verfahren getestet:

- Streckung des interessanten Bereiches des Histogramms

- Flächen-Übereinstimmung
- Energienormierung
- spaltenweise Energienormierung
- Kantenbild
- Bild mit hervorgehobenen Kanten

Diese werden im Folgenden vorgestellt.

4.1.1 Streckung des relevanten Histogrammbereiches

In [Sch98] wird ein Verfahren verwendet, bei dem der mittlere Bereich des Histogramms, in welchem sich die meisten Bildpunkte befinden, soweit gestreckt wird, dass er die gesamte Breite des Histogramms ausfüllt. Der Autor geht davon aus, dass die relevanten Bildinhalte jeweils soviele Bildpunkte besitzen, dass sie im Histogramm auffallen. Es wird dann ein Bereich mitten im Histogramm ausgewählt, dessen Breite auf eine Normbreite gestreckt wird. Wenn die Aufnahmen sich stark unterscheiden, ist es recht schwierig, die Grenzen dieses Bereiches zu finden. Für diesen Algorithmus muss das gesamte Bild zweimal durchlaufen werden, und die Schwellwerte im Histogramm müssen festgelegt werden.

4.1.2 Flächen-Übereinstimmung

Bei diesem Verfahren wird davon ausgegangen, dass bestimmte Farben in jedem Bild enthalten sind. Jedes der Testbilder enthält nah am Roboter ein Stück Fußboden und etwas höher die Wände des Flurs. Von diesen Farben beansprucht jede so große Bildteile, dass sie im Histogramm aller Bilder auffindbar sein sollten. Diese Helligkeiten dienen dann zur Kalibrierung der Helligkeitsnormierung, indem jedes Bild so modifiziert wird, dass seine Fußboden- und Wandfarbe mit einer Normfußbodenfarbe und einer Normwandfarbe übereinstimmen. Die Änderungen für die Histogrammbereiche am Rand und zwischen den Kalibrierungswerten müssen interpoliert werden.

4.1.3 Energienormierung

Grundlage bei diesem Ansatz aus [Kum96] ist die Gesamtintensität, die das jeweilige Bild hat. Sie wird berechnet als Summe der Helligkeiten aller Bildpunkte:

$$E = \sum_{i=1}^{N} v_i, \tag{4.1}$$

wobei N die Anzahl der Bildpunkte ist, und deren Helligkeit v_i größer als Null sein muss. Um diese Energie zweier Bilder anzupassen, werden in einem der Bilder alle Bildpunkte (bzw. deren Helligkeit) mit einem geeigneten Faktor multipliziert:

$$v_i{}' = v_i \frac{E_n}{E}. \tag{4.2}$$

Die *Normintensität* E_n wird hierbei z.B. auf die Hälfte der maximal möglichen Intensität oder mit Hilfe einiger Beispielbilder festgelegt. Jedes Bild, dessen Helligkeiten nach dieser Formel modifiziert werden, hat anschließend die Gesamtintensität E_n. Um ein Bild anzugleichen, benötigt dieser Algorithmus zwei Durchgänge durch das gesamte Bild mit einfachen Rechenoperationen.

4.1.4 Spaltenweise Energienormierung

Durch unterschiedlich große, überbelichtete Bereiche in den Bildern führt die Normierung der Gesamtintensität nicht zum gewünschten Ergebnis. Damit sich überbelichtete Punkte in einem Bildteil nicht auf die Normierung des gesamten Bildes auswirken können, ist eine Unterteilung des Bildes erforderlich.

Bei der spaltenweisen Energienormierung wird jede Bildspalte des Panoramabildes (Darstellung in Polarkoordinaten) einzeln normiert. Dabei wird – analog zur Normierung der Gesamtenergie – die Summe aller Bildpunkte der Spalte gebildet. Aus dieser wird dann ein Faktor berechnet, um alle Punkte der Spalte zu modifizieren. Die Bilder 4.2 und 4.3 veranschaulichen das Resultat des Algorithmus. Auch er benötigt zwei Durchgänge für jede Spalte und erfordert damit zwei Durchgänge durch das gesamte Bild. Es werden die gleichen einfachen Rechenoperationen wie bei der Energienormierung verwendet.

Abbildung 4.2: Originalbild vor der Normierung

4.1.5 Kantenbilder

Bei diesem Verfahren werden die Bildpunkte umso intensiver gezeichnet, je stärker sich die Helligkeit ihrer Nachbarn voneinander unterscheidet. Dadurch

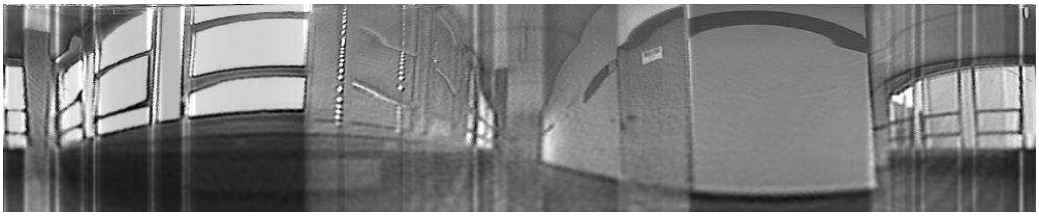

Abbildung 4.3: Spaltenweise normiertes Bild

Abbildung 4.4: Kanten in Bild 4.2

werden nur Helligkeitsunterschiede verarbeitet, aber die absolute Helligkeit bleibt unbeachtet und damit auch ihre Schwankungen. In den Ausgabebildern dieses Verfahrens sind nur die Kanten zwischen verschieden hellen Flächen zu sehen (siehe Abb. 4.4). Die Berechnung des Kantenbildes erfordert pro Bild nur einen Durchgang mit einfachen Rechenoperationen.

4.1.6 Hervorgehobene Kanten

Da bei reinen Kantenbildern ein Großteil der Information verlorengeht, ist es evtl. sinnvoll, zusätzlich zu den Kanten das eigentliche Bild zu verwenden. Ein Beispiel für ein solches Bild mit hervorgehobenen Kanten ist mit Abbildung 4.5 gegeben. Hier kann auch der Grauwert der Flächen zwischen den Kanten als Informationsquelle dienen.

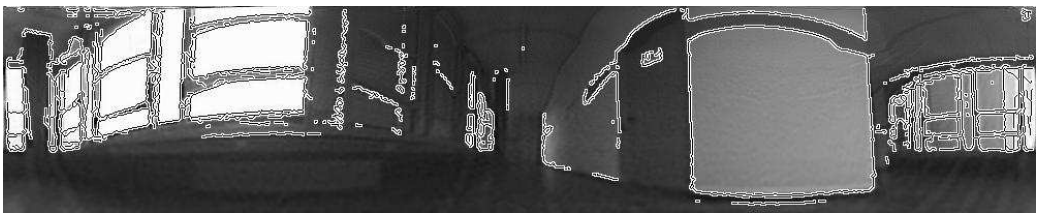

Abbildung 4.5: Bild mit hervorgehobenen Kanten

4.2 Methoden

Histogramme der von den verschiedenen Verfahren angepassten Bilder zeigen die erreichten Verbesserungen qualitativ. Bei einer fortlaufenden Serie von Bildern (einer Kamerafahrt entlang eines Flurs) bemerkt der Betrachter die Helligkeitsänderung von einem Bild zum nächsten besonders deutlich. Ein ideales Angleichungsverfahren sollte diese Bilder so modifizieren, dass zumindest dem (menschlichen) Betrachter keine Helligkeitsunterschiede mehr auffallen.

Eine genauere Angabe der Leistung der Algorithmen (insbesondere bezüglich der Verwendbarkeit als Vorverarbeitung zur Lokalisation) wird durch das in Kapitel 7 vorgestellte Verfahren erreicht.

5

Extraktion der Farbinformation

5.1 Motivation

Die Farbinformation in den Bildern kann auf verschiedene Weise zur Lokalisation verwendet werden. Eine Möglichkeit ist, dass es ein Objekte einer bestimmten Farbe nur an genau einem Ort (oder an sehr wenigen Orten) gibt. Sobald diese Farbe erkannt wird, befindet der Roboter sich also an einem dieser Orte.

Eine andere Möglichkeit besteht darin, die **Farbverteilung** eines Raumes als Merkmal zu verwenden. Dabei wird gespeichert, welche Farben wie häufig auftreten. Unterschiedlich eingerichtete Räume haben unterschiedliche Farbzusammensetzungen, sodass der Roboter zumindest zwischen den Räumen unterscheiden können sollte.

Die Farbinformation kann auch dazu verwendet werden, **einfarbige Flächen** als solche zu erkennen und damit Kanten zwischen ihnen zu detektieren. In Abschnitt 2.1 wird ein Verfahren vorgestellt, bei dem unter anderem anhand der Farbe des Fußbodens der freie Bereich um den Roboter herum gefunden wird.

Um Farbinformationen nutzen zu können, müssen sie in einem geeigneten Datenformat bearbeitet werden. Eine Farbe wird üblicherweise als dreidimensionaler Vektor dargestellt, wobei es verschiedene Vektorräume (sog. *Farbräume*) gibt. Zur Lokalisation sollte ein Farbraum folgende Eigenschaften haben:

- Beleuchtungs- und Farbinformation sollten separat dargestellt werden, damit Änderungen der Beleuchtung ignoriert werden können.
- Vom Betrachter als ähnlich empfundene Farben sollten im Farbraum eine geringe Distanz haben.

- Die Information sollte effizient verarbeitet werden können.

5.2 Farbräume

Farbräume können eingeteilt werden in solche, die für Menschen erdacht wurden, und solche, die Farben für technische Geräte beschreiben. Letztere sind meist wenig intuitiv, sondern von technischen Eigenschaften der Bildröhren, Kameras oder Drucker beeinflusst.

5.2.1 RGB-Farbraum

Der RGB-Farbraum ist der bei Computern am häufigsten verwendete Farbraum, weil er den technischen Eigenschaften der Kathodenstrahlröhre der meisten Monitore angepasst ist. Eine im RGB-Raum angegebene Farbe kann ohne Umwandlung auf dem Monitor dargestellt werden. Da der RGB-Raum auch der Farbraum ist, in dem die Kamera die Bilder ausgibt, werden alle Umrechnungsformeln dieses Kapitels ausgehend von RGB angegeben.

Die Komponenten des 3D-RGB-Farbvektors sind die Anteile an Rot, Grün und Blau, aus denen die jeweilige Farbe zusammengesetzt ist. Eine solche Darstellung, die auf drei Primärfarben beruht, wird auch *Tristimulus* genannt (siehe [Jä97]).

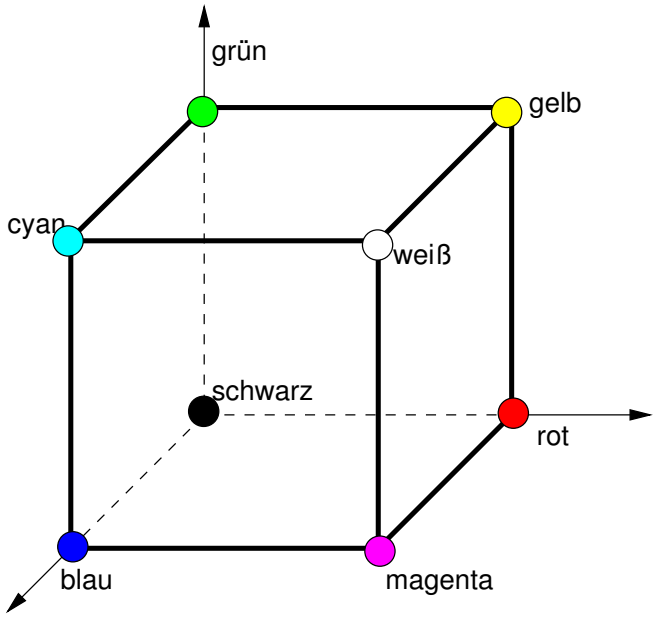

Abbildung 5.1: Würfel der im RGB-Raum darstellbaren Farben

Alle im RGB-Raum verfügbaren Farben befinden sich in einem Würfel (siehe Abb. 5.1). Eine der Ecken dieses Würfels befindet sich im Nullpunkt des Koordinatensystems. Diese entspricht Schwarz, bei dem alle drei Anteile gleich Null sind. Die diagonal gegenüberliegende Ecke stellt Weiß dar und die Koordinatenachsen zeigen auf die Ecken mit Rot, Grün und Blau.

5.2.2 HSV-Farbraum

Der HSV-Farbraum ist weit intuitiver als der RGB-Raum. Da er der menschlichen Wahrnehmung entgegenkommt, wird er von den meisten Bildverarbeitungsprogrammen unterstützt. Hier sind die Komponenten zur Farbdarstellung der **Farbton** (engl. Hue), die **Sättigung** (engl. Saturation) und die **Helligkeit** (auch Intensität, engl. Value, Lightness oder Intensity). In der Literatur gibt es für diesen Farbraum sowohl unterschiedliche Bezeichnungen (HSL, HSI) als auch unterschiedliche Definitionen für gleich bezeichnete Farbräume.

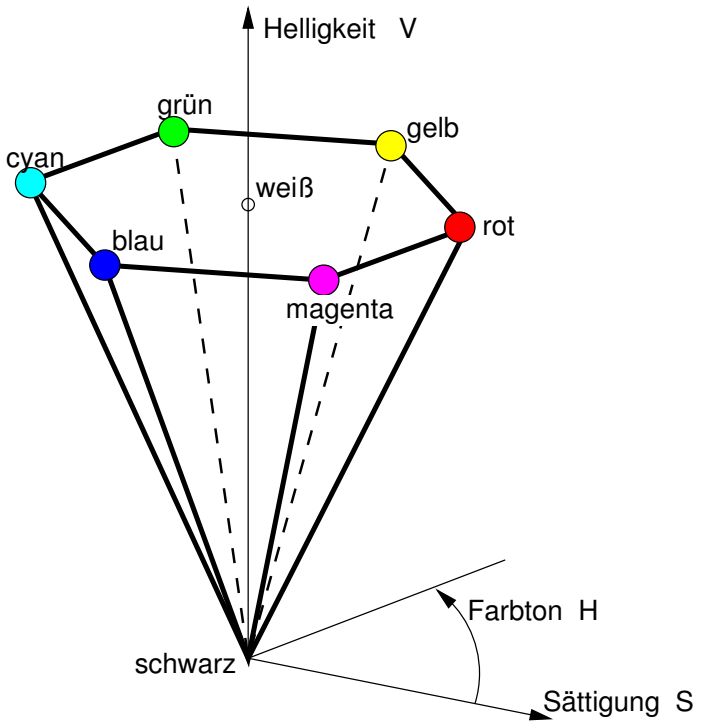

Abbildung 5.2: Der HSV-Bildraum als Pyramide

Die Idee ist aber gleich: Der RGB-Würfel wird in eine Pyramide (siehe Abb. 5.2) umgewandelt, deren Achse der Geraden von Schwarz nach Weiß entspricht. In der Spitze der Pyramide liegt Schwarz. Der Parameter Helligkeit gibt an, wie weit eine Farbe entlang dieser Achse von Schwarz entfernt ist. Die beiden anderen

Parameter beziehen sich auf die Farbe und geben deren Position in einer Ebene parallel zur Grundfläche der Pyramide an. Die Sättigung gibt an, wie rein eine Farbe ist. Je weniger Weiß die Farbe enthält, desto gesättigter ist sie und desto weiter ist sie von der Pyramidenachse entfernt. Die Farbtöne werden als Winkel in der Ebene angegeben, wobei Rot als Referenz ($H = 0$) dient.

Die folgende Definition wird in [Fel92] und [FR98] übereinstimmend angegeben:

$$Max = max(R, G, B) \tag{5.1}$$

$$Min = min(R, G, B) \tag{5.2}$$

Helligkeit: $$V = Max \tag{5.3}$$

Sättigung: $$S = \begin{cases} \frac{Max-Min}{Max} & \text{falls} \quad V > 0, \\ 0 & \text{sonst} \end{cases} \tag{5.4}$$

$$R' = 60\frac{Max - R}{Max - Min} \tag{5.5}$$

$$G' = 60\frac{Max - G}{Max - Min} \tag{5.6}$$

$$B' = 60\frac{Max - B}{Max - Min} \tag{5.7}$$

Farbton: $$H = \begin{cases} \text{undefiniert} & \text{falls } S = 0, \\ 300 + B' & \text{falls } R = Max \text{ und } G = Min, \\ 60 - G' & \text{falls } R = Max \text{ und } G \neq Min, \\ 60 + R' & \text{falls } G = Max \text{ und } B = Min, \\ 180 - B' & \text{falls } G = Max \text{ und } B \neq Min, \\ 180 + G' & \text{falls } B = Max \text{ und } R = Min, \\ 300 - R' & \text{falls } B = Max \text{ und } R \neq Min \end{cases} \tag{5.8}$$

5.2.3 YUV-Farbraum

Da der HSV-Farbraum die Helligkeit nicht besonders gut trennt, wird in manchen Bildverarbeitungssystemen der YUV-Farbraum verwendet. Er wurde für das Fernsehen entwickelt und enthält in seiner Y-Komponente die Helligkeitsinformation, während der U- und V-Kanal die Farbe speichern. Das YUV-Signal wird aus

dem RGB-Signal nach folgender Formel errechnet (nach [PM92]):

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \tag{5.9}$$

$$U = B - Y \tag{5.10}$$

$$V = R - Y \tag{5.11}$$

Der Y-Kanal mit der Helligkeit des Bildpunktes wird an anderen Stellen dieser Arbeit benutzt, wenn Grauwerte benötigt werden. Für alle Grautöne (einschließlich schwarz und weiß) sind U und V gleich Null. Bei einer Speicherung als Integer muss beachtet werden, dass U und V negativ sein können.

5.2.4 Lab-Farbraum

Alle bisher beschriebenen Farbräume haben einen Nachteil: Eine bestimmte Differenz im Farbraum bedeutet keinen gleich wahrgenommenen Unterschied. Der geringste vom Benutzer wahrgenommene Farbunterschied zwischen zwei Farben wird daher bei unterschiedlichen Farben auf unterschiedlich große Differenzen im Farbraum abgebildet.

Der von der CIE (Commission Internationae de l'Eclairage) entwickelte Lab-Farbraum bildet gleich wahrgenommene Unterschiede auf gleiche Distanzen im Farbraum ab, und ist daher besonders geeignet, Unterschiede zwischen Farben zu untersuchen. Darüber hinaus bietet die Linearität zur Wahrnehmung die Möglichkeit einer effizienten Kodierung (siehe [PM92]).

Die Berechnung der Koordinaten in diesem Farbraum aus denen im RGB-Raum ist etwas komplexer als für die bisherigen Farbräume. Zunächst erfolgt eine Umrechnung in den XYZ-Farbraum, der die Grundlage aller CIE-Farbräume ist:

$$X = 0.607\,R + 0.174\,G + 0.201\,B \tag{5.12}$$

$$Y = 0.299\,R + 0.587\,G + 0.114\,B \tag{5.13}$$

$$Z = 0\,R + 0.066\,G + 1.117\,B \tag{5.14}$$

Aus diesem werden dann die Koordinaten des Lab-Farbraumes bestimmt (siehe [Pra78]):

$$L = 25\,\sqrt[3]{100\frac{Y}{Y_0}} - 16 \tag{5.15}$$

$$a = 500\,\left(\sqrt[3]{\frac{X}{X_0}} - \sqrt[3]{\frac{Y}{Y_0}}\right) \tag{5.16}$$

$$b = 200 \left(\sqrt[3]{\frac{Y}{Y_0}} - \sqrt[3]{\frac{Z}{Z_0}} \right) \tag{5.17}$$

Hierbei steht $(X_0, Y_0, Z_0) = (0.982, 1.000, 1.183)$ für die Koordinaten von Weiß im CIE-XYZ-Farbraum. Die L-Komponente dieses Farbraumes enthält die Helligkeit (Luminance) der Farbe. Die Komponenten a und b beschreiben deren Position in einer Farbebene. Das Sechseck, das alle darstellbaren Farben der HSV-Pyramide enthält, wird in der ab-Farbebene dieses Farbraumes zu einem unsymmetrischen Polygon (siehe Abb. 5.3).

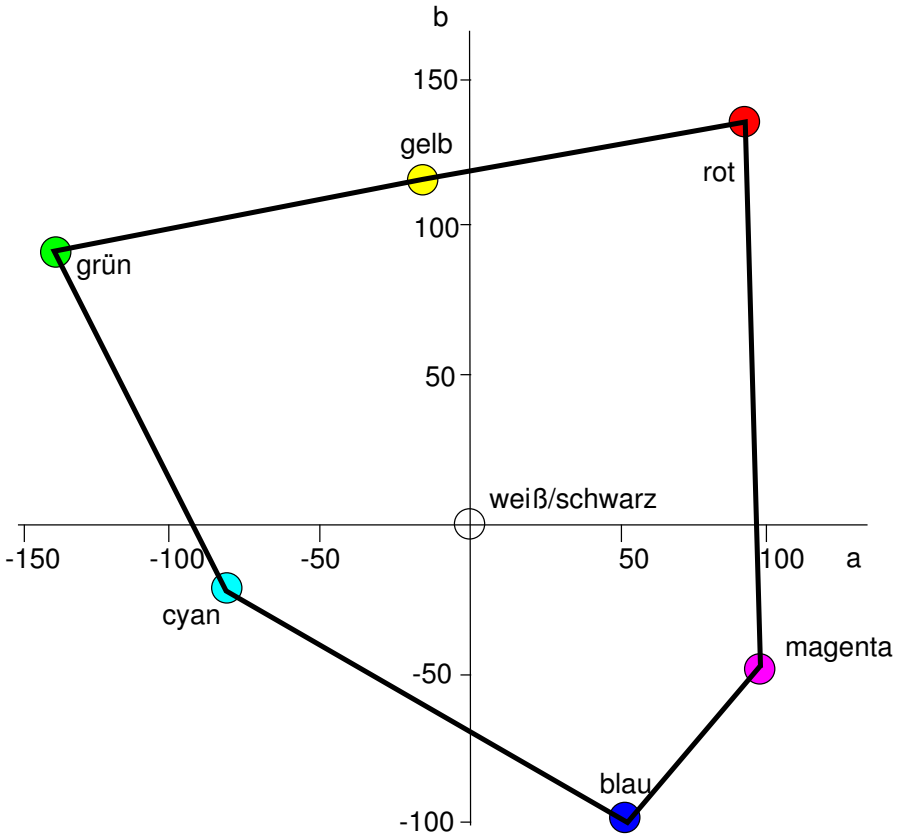

Abbildung 5.3: Darstellung einiger Farben in der Farbebene des Lab-Farbraumes

Ob Linearität zur Wahrnehmung für die Lokalisation einen so großen Vorteil bietet, dass diese komplexen Rechenoperationen gerechtfertigt sind, müssen die Experimente zeigen. Die Umrechnung muss schließlich für jeden Punkt durchgeführt werden. Immerhin wird in Abschnitt 2.1 ein System beschrieben, das mit dem Lab-Farbraum erfolgreich arbeitet.

5.2.5 Problemangepasster Farbraum

Evtl. lässt sich ein völlig neuer Farbraum finden, der zur Lokalisation besser geeignet ist, als die bisher beschriebenen. Dazu muss eine Abbildung aus dem RGB-Raum in den neuen Farbraum gefunden werden, die lokalisationsrelevante Informationen hervorhebt. Der dabei gebildete neue Farbraum benötigt möglicherweise sogar weniger Dimensionen als die oben erwähnten.

5.3 Methoden

Um den Informationsgehalt der Größen der verschiedenen Farbräume zu untersuchen, werden einkanalige Bilder dieser Größen berechnet. Sie geben dem Betrachter die Möglichkeit, den Informationsgehalt der jeweiligen Größe abzuschätzen. Bild 5.4 soll hierzu als Vorlage dienen. Es wurde mit der Hitachi HV-C20-Kamera aufgenommen.

Abbildung 5.4: Ausgangsbild für Farbuntersuchungen

5.3.1 Sättigung

Abbildung 5.5: Sättigung (je stärker gesättigt desto dunkler)

In Abbildung 5.5 sind die gesättigten Bildpunkte hervorgehoben. Das Sättigungshistogramm (Abb. 5.6) zeigt, dass die meisten Punkte eine geringe Sättigung haben, also Grautöne sind. Dies liegt in erster Linie daran, dass die Testumgebung nur schwach gesättigte Farben enthält, aber auch daran, dass die Kameras Farben weniger gesättigt wiedergeben, als wir sie wahrnehmen.

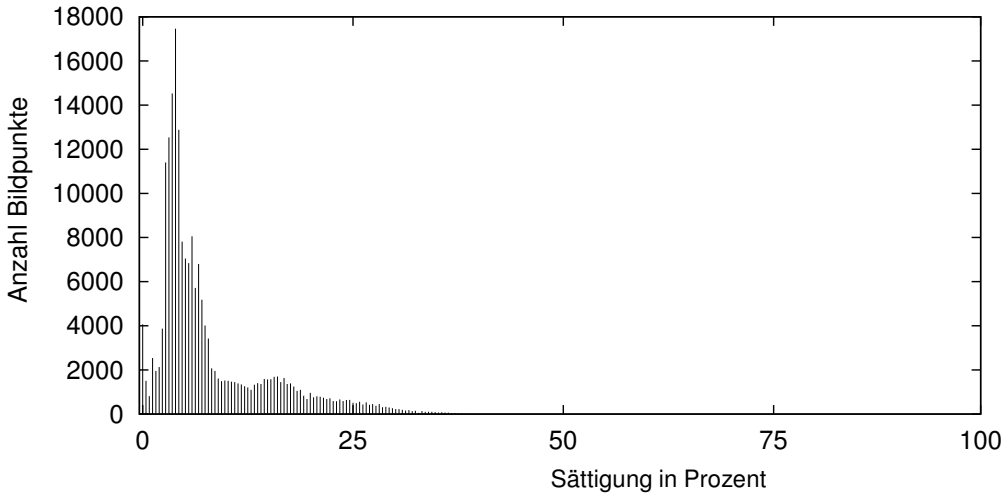

Abbildung 5.6: Sättigungs-Histogramm zu Bild 5.4

Die durchschnittliche Sättigung $\overline{S}$ der Pixel eines Bildes ist ein Maß für die Menge vorhandener Farbinformation. Sie wird berechnet als

$$\overline{S} = \frac{1}{N} \sum_{i=1}^{N} S(i), \tag{5.18}$$

wobei N die Anzahl der Bildpunkte angibt und $S(i)$ die Sättigung der Farbe des Punktes i.

5.3.2 Farbton

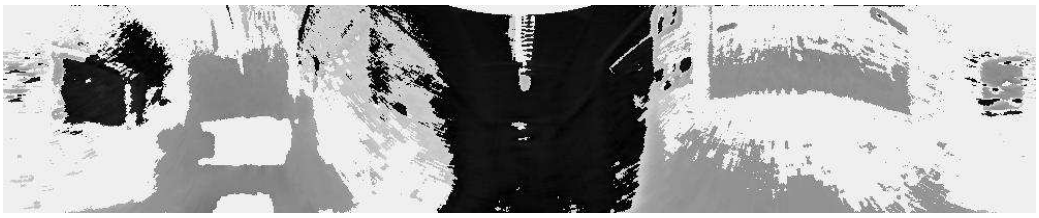

Abbildung 5.7: Farbtöne aus Bild 5.4 (jeder Grauwert entspricht einem Winkel)

In Abbildung 5.7 ist der Farbton der Bildpunkte dargestellt. Punkte, die in Bild 5.4 rot sind, deren Farbtonwinkel also Null ist, sind mit der Helligkeit Null (schwarz) dargestellt. Für die Farben mit größeren Hue-Werten werden entsprechend hellere Grautöne verwendet, wobei weiß für Punkte mit undefiniertem Farbton steht.

Per Definition (Gl. 5.8) ist der Farbton undefiniert, falls die Sättigung gleich Null ist. Im Experiment hat sich darüberhinaus gezeigt, dass die Kamera ein Farbrau-

schen bewirkt. Daher sind auch Punkte mit einer Sättigung unter 5% weiß dargestellt. In Bild 5.4 trifft dies für 45% der Bildpunkte zu.

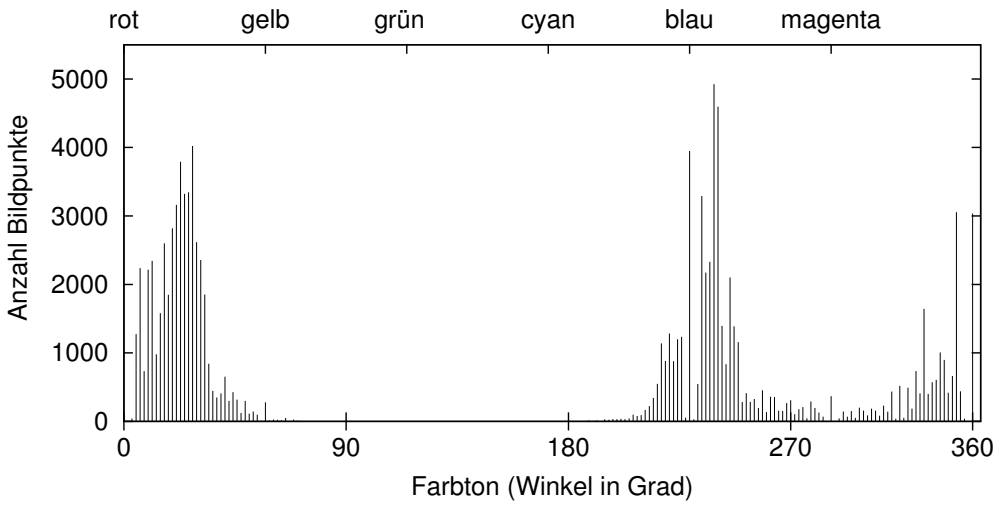

Abbildung 5.8: Farbton-Histogramm zu Bild 5.4

Das Histogramm (Abb. 5.8) zu Bild 5.7 zeigt, wieviele Punkte welches Farbtons es gibt. Im Bereich der gelb-roten Farben und im Bereich der Blautöne sind deutliche Maxima zu erkennen. Die Punkte mit undefiniertem Farbton sind in diesem Histogramm nicht aufgeführt. Histogramme dieser Art können verwendet werden, um Positionen anhand ihrer Farbverteilung zu unterscheiden.

6

Dimensionsreduktion

Es gibt viele Gründe die Kamerabilder zu komprimieren, bevor sie verarbeitet werden:

- Für viele Aufgaben wird nur ein Teil der Bildinformationen benötigt.
- Unter Echtzeitbedingungen können komplexe Algorithmen keine großen Datenmengen bearbeiten.
- Ein großer Teil der Bildinformation ändert sich von Bild zu Bild nicht, er ist also nicht lokalisationsrelevant.
- Auf dem Roboter steht nur begrenzter Speicher zur Verfügung.

Idealerweise sollte das Kompressionsverfahren Informationen, die zur Lokalisation benötigt werden, erhalten oder sogar hervorheben, und alle anderen Bildinhalte weglassen. Der lokalisationsrelevante Teil der Bildinformation ist nicht notwendigerweise ein Bildausschnitt, sondern allgemein das Kriterium, welches in der Menge aller auftretenden Bilder von der Aufnahmeposition abhängt. Ein Algorithmus, der diese Informationen extrahieren soll, benötigt klassifizierte Lerndaten, also Bilder mit Positionsinformationen. Dann kann er entscheiden, welche Bildinformationen trotz unterschiedlicher Position gleich bleiben und welche der Veränderungen zwischen den Bildern durch unterschiedliche Positionen hervorgerufen werden.

Da es recht aufwendig ist, den lokalisationsrelevanten Teil der Bilder zu bestimmen, werden zunächst einfachere Verfahren vorgestellt.

6.1 Bildorientierte Verfahren

In diesem Abschnitt werden Verfahren aus der klassischen Bildverarbeitung aufgezählt, die auch als Vorstufe anderer Verfahren Verwendung finden. Sie können auch miteinander kombiniert werden.

Bildausschnitte

In [Sch98] wird nur der obere Teil der Bilder verwendet, weil dort Möbel und andere Gegenstände, die verschoben werden könnten, seltener sind. Schwert teilt außerdem das Rundumsichtbild in Segmente ein und schätzte dann für jedes Segment einzeln die Position. Dadurch kann die Position bei einer Störung in einem der Sektoren (z.B. Personen) anhand der anderen noch immer bestimmt werden.

Um solche Bildausschnitte zu ermitteln, wird nur sehr wenig Rechenzeit benötigt. Wenn sie sich als geeignete Merkmale herausstellen, wäre es evtl. sinnvoll, das Sichtsystem zu verändern, sodass es nur einen Teil des Bildes aufnimmt.

Skalierung des Bildes

Mehrere Pixel des Bildes werden hierbei zu einem Pixel zusammengefasst. Dies geschieht durch Bildung ihres Mittelwertes. Ein günstiger Nebeneffekt hierbei ist, dass das Rauschen, welches die Kameras produzieren, geglättet wird. Die Skalierung gehört ebenfalls zu den mit wenig Rechenleistung auskommenden Bildoperationen.

6.2 Hauptkomponentenanalyse

Für die folgenden Verfahren werden die Bilder wie Vektoren in einem hochdimensionalen Vektorraum verwendet. Die einzelnen Bildpunkte werden zeilenweise aneinandergereiht, sodass ein langer Vektor entsteht. Für Grauwertbilder entspricht die Dimension dieses Vektors der Anzahl der Bildpunkte. Die Helligkeiten der Bildpunkte werden zu den Komponenten des Bildvektors. Um Farbbilder darzustellen, ist die Dimension dreimal so groß wie die Anzahl der Bildpunkte, da ein Farbpunkt üblicherweise mit drei Werten dargestellt wird. Eine Menge von Bildern wird so zu einer Mannigfaltigkeit oder Punktmenge, deren Form bzw. Verteilung Gegenstand der folgenden Verfahren ist.

Die verwendeten Bilder haben 200 000 Bildpunkte. Da es sich um Farbbilder handelt, müssten sie in einem Vektorraum mit 600 000 Dimensionen bearbeitet werden. Aus dieser Größe wird deutlich, dass der Einsatz aufwendiger Algorithmen

mit ganzen Bildern problematisch ist. Daher sollte zunächst ein anderes einfacheres Verfahren vorgeschaltet und dann die Hauptkomponentenanalyse angewendet werden.

Die Hauptkomponentenanalyse (engl. Principal Component Analysis, PCA) bildet die Vektoren in einen Unterraum mit geringerer Dimension ab, wobei das Ziel ist, möglichst wenig Informationen zu verlieren.

Ausgangspunkt für die Abbildung der Bildvektoren in einen Unterraum ist die Verteilung der Vektoren im hochdimensionalen Bildraum. Der Mittelpunkt dieser Verteilung bildet dabei den Ursprung des neuen kartesischen Koordinatensystems. Dessen erste Koordinatenachse wird in Richtung der größten Varianz der Menge der Bildvektoren gelegt, die Zweite entlang der Richtung der zweitgrößten Varianz usw.

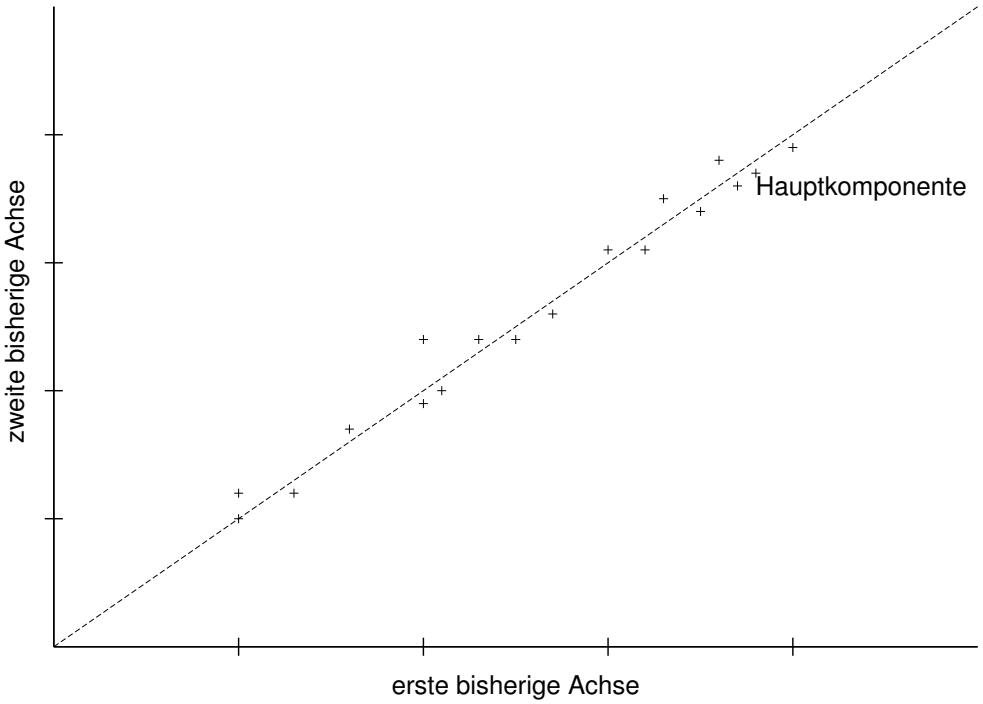

Abbildung 6.1: Diese Punktwolke erstreckt sich hauptsächlich in Richtung der eingezeichneten Achse.

Es lässt sich zeigen (siehe [Loe55]), dass diese Wahl des Unterraums den mittleren quadratischen Fehler zwischen den Originaldaten und den aus dem Unterraum zurücktransformierten Daten minimiert. Durch die Wahl der Dimension des Unterraumes wird beeinflusst, wie groß der Informationsverlust ist. Je mehr Dimensionen der Unterraum hat, um so genauer ist die Darstellung der Daten. Der Algorithmus nutzt die Beobachtung, dass hochdimensionale Datenwolken häufig

nur in wenigen Richtungen variieren wie die in Abbildung 6.1. Bei einer derartigen Verteilung bleibt ein großer Teil der Information erhalten, wenn die Daten auf die eingezeichnete Achse projiziert werden.

Im Folgenden werden unterschiedliche Methoden zur Bestimmung der Hauptachsen einer Menge von Datenvektoren $\{\vec{v}_1, \ldots, \vec{v}_n\}$ vorgestellt.

6.2.1 Analytische Bestimmung der Hauptkomponenten

Bei diesem Verfahren werden die Hauptachsen mit Hilfe der Kovarianzmatrix berechnet. Als erstes muss der Mittelwert der Datenvektoren bestimmt und von allen abgezogen werden:

$$\overline{\vec{v}} = \frac{1}{n} \sum_{i=1}^{n} \vec{v}_i \tag{6.1}$$

$$\vec{x}_i = \vec{v}_i - \overline{\vec{v}} \tag{6.2}$$

Nun werden die Datenvektoren (Spaltenvektoren) nebeneinander in eine Matrix $\mathbf{X}$ geschrieben, die zur Berechnung der Kovarianzmatrix $\mathbf{K}$ dient:

$$\mathbf{X} := [\vec{x}_1 \vec{x}_2 \ldots \vec{x}_n] \tag{6.3}$$

$$\mathbf{K} := \mathbf{X}\,\mathbf{X}^{\mathsf{T}} \tag{6.4}$$

Für die Komponente $k_{i,j}$ von $\mathbf{K}$ werden also die Produkte der i-ten mit der j-ten Komponente aller Datenvektoren aufsummiert. Für mittelwertsfreie Daten ist $\mathbf{K}$ gleich der Korrelationsmatrix.

Der Eigenvektor von $\mathbf{K}$ mit dem größten Eigenwert zeigt in die Richtung, in der die Daten die größte Varianz haben; der Eigenvektor mit dem zweitgrößten Eigenwert in Richtung der zweitgrößten Varianz usw. Die Größe der Kovarianzmatrix beträgt $n \times n$, wobei n die Dimension der Eingangsdaten ist. Da diese sehr groß ist, ist es problematisch, $\mathbf{K}$ aufzustellen und die Eigenvektoren zu bestimmen.

Ein Ausweg, der von Murakami und Kumar in [MK82] beschrieben wird, ist, statt der Kovarianzmatrix die implizite Kovarianzmatrix $\widetilde{\mathbf{K}}$ zu verwenden.

$$\widetilde{\mathbf{K}} := \mathbf{X}^{\mathsf{T}}\,\mathbf{X} \tag{6.5}$$

Ihre Dimension ist nur $m \times m$, wobei m für die Anzahl der Eingangsdaten steht, die normalerweise nur bei einigen Hundert oder Tausend liegt. Aus den Eigenvektoren $\widetilde{\vec{e}}$ und Eigenwertenwerten $\widetilde{\lambda}$ von $\widetilde{\mathbf{K}}$ lassen sich dann die der Kovarianzmatrix

berechnen:

$$\lambda_i = \widetilde{\lambda}_i \tag{6.6}$$

$$\vec{e}_i = \widetilde{\lambda}_i^{-\frac{1}{2}} \; \mathbf{X} \, \widetilde{\vec{e}}_i \tag{6.7}$$

Zur Berechnung der Eigenwerte und -vektoren der Matrizen werden meist Standardverfahren wie das von Jacobi (siehe [Fis86]) benutzt. C. Berger z.B. wendet in [Ber00] eben dieses an. Da hierfür mit steigender Anzahl der Eingabedaten sehr viel Arbeitsspeicher erforderlich ist, wird im Folgenden eine Möglichkeit der Hauptkomponentenbestimmung vorgestellt, für die keine Matrix aufgestellt zu werden braucht.

6.2.2 Iterative Bestimmung der Hauptkomponenten

In [HKP91] wird ein neuronales Netz von Yuille und anderen beschrieben, welches die Hauptkomponenten lernt. Dieses wird auch von V. Schwert in [Sch98] verwendet. Da die Daten hierfür mittelwertsfrei sein müssen, wird der Mittelwert nach Gleichung 6.2 abgezogen. Dann wird der Vektor $\vec{w}$ zufällig initialisiert. Jetzt beginnt der eigentliche *Lernalgorithmus*, hierbei werden die Eingabedaten evtl. mehrfach durchgegangen, wobei $\vec{w}$ jedes Mal jedem Datenvektor um ein kleines Stück angenähert wird. Dies geschieht nach der Formel

$$\vec{w}\,' = \vec{w} + \eta \, (\, V \, \vec{x}_i - |\vec{w}|^2 \, \vec{w} \,) \tag{6.8}$$

$$\text{mit} \quad V = \vec{w}^{\mathsf{T}} \vec{x}_i \qquad \text{(Skalarpodukt).}$$

Hierbei regelt η, wie stark $\vec{w}$ geändert wird. Dies sollte zunächst groß sein, um vom Initialwert in die richtige Richtung zu kommen, und dann immer kleiner werden, damit ein einzelnes untypisches Datum $\vec{w}$ nicht zu stark beeinflusst. Nach genügend vielen Durchgängen zeigt $\vec{w}$ in die gleiche Richtung wie die Mehrheit der Datenvektoren. Dies ist die erste Hauptkomponente.

Um die zweite Hauptkomponente zu finden, muss von den Datenvektoren ihre Projektion auf die erste Hauptkomponente abgezogen werden:

$$\vec{x}_i\,' = \vec{x}_i - V\vec{w} \tag{6.9}$$

Dadurch wird die Datenwolke in einen Unterraum projiziert, der um eine Dimension kleiner ist als der Ausgangsraum und auf dem die bereits gefundene Achse senkrecht steht. In Abb. 6.1 entspricht dies einer Geraden orthogonal zur eingezeichneten ersten Hauptkomponente. Nun wird wieder mit Formel 6.8 die Hauptrichtung bestimmt, welche der zweiten Hauptkomponente $\vec{w}_2$ der ursprünglichen Vektoren entspricht.

Zur Bestimmung weiterer Achsen werden die Vektoren erneut in einen um eine Dimension kleineren Unterraum projiziert, um dort wieder die Hauptrichtung zu ermitteln.

Dieses Verfahren benötigt – bei sparsamer Implementierung – nur Hauptspeicher für die Eingangsdaten und die gerade gesuchte Hauptkomponente. Außerdem ermittelt es nur soviele Achsen, wie benötigt werden, im Gegensatz zur analytischen Methode, bei der immer alle Eigenvektoren berechnet werden. Ein weiterer Vorteil gegenüber dieser ist, dass die Hauptkomponenten auch nach der Lernphase angepasst werden können. Dadurch besteht die Möglichkeit, sich auf eine verändernde Umwelt einzustellen.

7

Merkmalsvergleich

Klassischerweise wird die Qualität von Algorithmen anhand des Bedarfs an Speicher und Rechenzeit bewertet. Diese Eigenschaften sagen nur wenig über den Nutzen bei der Lokalisation aus. Darum werden in dieser Arbeit weitere Kriterien betrachtet.

7.1 Aufgaben der Merkmale

Durch Bildverarbeitung werden Merkmale aus den Bildern extrahiert. Merkmale können z.B. ganze Bilder, Kantenbilder, Farb- oder Grauwerthistogramme oder Positionen von Markierungen sein. Sie müssen die verschiedenen Positionen eindeutig von einander unterscheiden und sollten unabhängig von Störungen (z.B. Beleuchtungsänderungen) sein.

Alle oben erwähnten Merkmale lassen sich als (evtl. hochdimensionale) Vektoren darstellen. Da auch die Eingangsbilder hochdimensionale Vektoren sind (wenn alle Bildpunkte untereinander geschrieben werden), kann die Bildverarbeitung als Abbildung von einem Vektorraum (den *Bildraum*) in einen anderen (dem *Merkmalsraum*) betrachtet werden. Da aus den Merkmalen mit Hilfe der Karte die Position geschätzt wird, muss die Karte eine Abbildung vom Merkmalsraum in den Raum der Positionen ermöglichen. Damit hängt der von der Karte benötigte Speicher von der Dimension des Merkmalsraumes ab.

Aus den Anforderungen an autonome mobile Roboter (siehe Abschnitt 1.2) ergeben sich einige Anforderungen an die Bildverarbeitung und die Merkmale:

- Durch die Merkmale müssen sich die Positionen möglichst deutlich voneinander unterscheiden.
- Im Merkmalsraum sollten die Vektoren von Bildern, die zu benachbarten Positionen gehören, näher zusammenliegen als jene, deren Aufnahmepositionen weit voneinander entfernt sind.
- Die Berechnung der Merkmale eines Bildes muss in Echtzeit möglich sein.
- Die Dimension des Merkmalsraumes sollte möglichst klein sein, damit erstens die Karte nicht zu viel Speicher benötigt und zweitens der Positionsschätzer nicht zu komplex wird.
- Zur Bildverarbeitung sollten keine fest vorgegebenen Schwellwerte verwendet werden, sondern die Algorithmen sollten benötigte Schwellwerte automatisch aus den Daten ermitteln.

Der ideale Algorithmus extrahiert also aus den Bildern mit wenig Rechenaufwand wenige, aussagekräftige Merkmale, durch die die Aufnahmeposition unabhängig von Störungen eindeutig charakterisiert wird.

7.2 Messverfahren

7.2.1 Aufgabe des Messverfahrens

Ein komplettes Lokalisationssystem kann daran gemessen werden, wie genau der Roboter seine Position im Durchschnittsfall bestimmen kann, oder in wievielen Fällen der Fehler der Positionsbestimmung unter einer Grenze liegt. Volkmar Schwert ist in [Sch98] so vorgegangen. Um einen Teilalgorithmus zu bewerten, integriert er ihn in das Gesamtsystem und testet dieses dann. Solch eine Vorgehensweise ist recht aufwendig, besonders wenn eine Lernkomponente (z.B. ein Fuzzy-Controller als Positionsschätzer) beteiligt ist oder weitere Parameter (z.B. Umweltbedingungen) variiert werden sollen. Wenn verschiedene Teilalgorithmen des Systems untersucht und ggf. verändert werden, ist so ein Verfahren auch deshalb problematisch, weil die Änderungen an einer Verarbeitungsstufe die Eingabedaten für die folgenden Verarbeitungsstufen beeinflussen. Am Anfang der Entwicklung muss die Bewertung schon deshalb auf einem anderen Weg erfolgen, weil noch kein Gesamtsystem existiert.

Um verschiedene Positionsschätzer einsetzen zu können, werden hier nur Merkmale verglichen und nicht die Genauigkeit der Positionsschätzung. Mit guten Merkmalen sollte es immer möglich sein, einen Positionsschätzer zu implementieren.

Folgende Größen zeigen, wie gut die oben aufgeführten Ziele für Merkmale erreicht werden:

- **Unterscheidbarkeit** der Positionen,
- **Interpolierbarkeit** eines Merkmalsvektors anhand seiner Nachbarn,
- **Rechenaufwand**, um Merkmale aus einem Bild zu gewinnen,
- **Dimension** des Merkmalsraums,
- Verhältnis der **Abstände** im Merkmalsraum zu denen der Aufnahmepositionen,
- **Unempfindlichkeit** gegenüber Störungen.

Die Unterscheidbarkeit sollte im Merkmalsraum nicht schlechter sein als im Bildraum, sonst verliert der Algorithmus wichtige Informationen. Der Rechenaufwand sollte zwar möglichst gering gehalten werden, aber da sich die Lokalisation als recht schwierig herausstellt, kann ihm keine sehr hohe Priorität gegeben werden. Die wichtigeren Ziele sind: Die Dimension des Merkmalsraumes möglichst gering zu halten und störungsunempfindliche Merkmale zu finden.

7.2.2 Abstandsmatrix

Die wichtigste Größe ist die Unterscheidbarkeit der Positionen anhand der Merkmale. Falls die Abstände im Merkmalsraum sich ähnlich verhalten wie die der Aufnahmepositionen, so erleichtert dies die Konstruktion des Positionsschätzers. Um den Zusammenhang dieser beiden Größen zu untersuchen werden zunächst für alle Bilder einer Serie die Merkmalsvektoren bestimmt, und dann wird für jeden Merkmalsvektor der Abstand zu jedem anderen in eine Matrix eingetragen.

Diese *Abstandsmatrix* ist die Grundlage der weiteren Vergleiche. Die naheliegendste Möglichkeit ihren Inhalt zu veranschaulichen ist, sie als Gebirge zu plotten. Dies ist in Abbildung 7.1 beispielhaft für die Grauwerthistogramme (256-dimensionale Daten) einer Bildserie dargestellt. Dieses Diagramm sowie alle folgenden wurden automatisch vom Programm *matrix* erstellt.

In der Diagonale ist der Abstand jedes Vektors zu sich selbst dargestellt. Dieser ist selbstverständlich für jeden Vektor gleich Null. Direkt neben der Diagonale sind die Abstände zu den Merkmalsvektoren benachbarter Aufnahmepositionen eingezeichnet. Sie sollten alle größer Null sein (sonst wären die betreffenden Positionen nicht unterscheidbar). Je weiter ein Punkt von der Diagonale entfernt ist, umso weiter liegt seine Aufnahmeposition von der Bezugsposition entfernt. Die

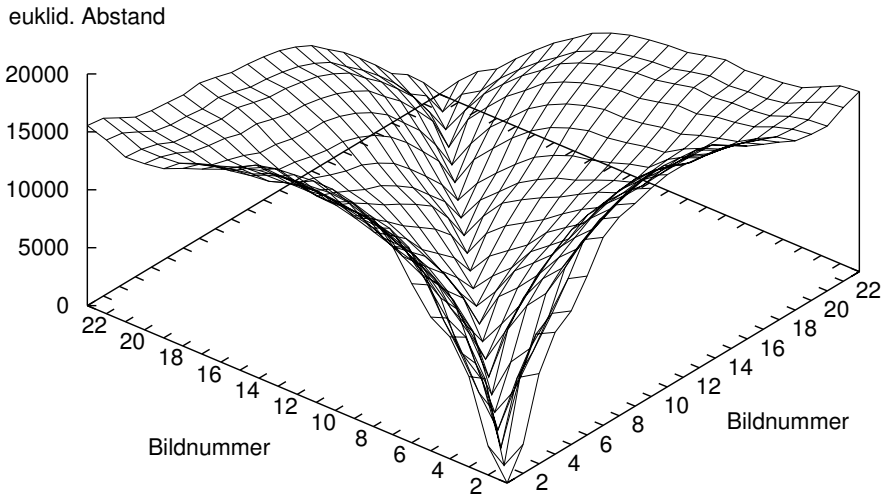

Abbildung 7.1: Abstandsgebirge für Grauwerthistogramme einer Serie mit 23 Bildern

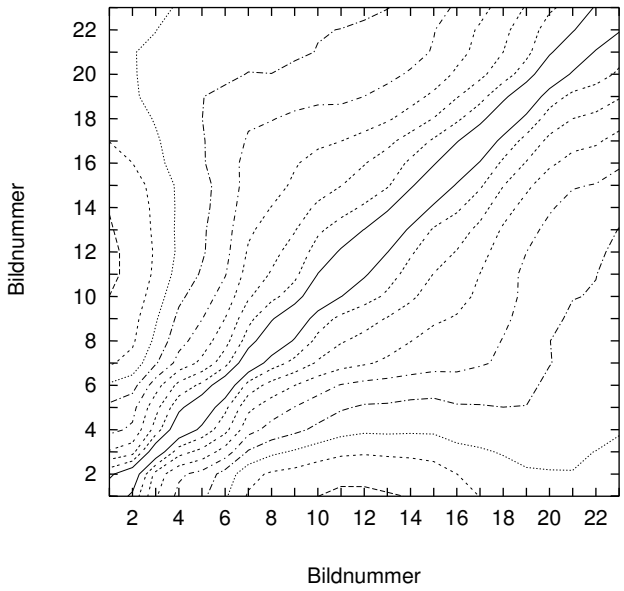

Abbildung 7.2: Höhenlinien des Abstandsgebirges aus Abb. 7.1

gezeichnete Oberfläche sollte also – bei einem idealen Merkmal – von der Diagonale aus monoton steigen.

Dies kann anhand der Höhenlinien (siehe Abb. 7.2) leichter geprüft werden. Hier scheint es, als wäre das getestete Merkmal geeignet, die Bilder entsprechend ihrer Aufnahmeposition zu ordnen. Die Abstände im Merkmalsraum (also die Unterschiede zwischen den Histogrammen) steigen von der Diagonale aus. Dass der Abstand zwischen Bild 3 und Bild 2 größer ist als der zwischen Bild 3 und Bild 5, fällt hier kaum auf. Da der Abstand der Aufnahmepositionen sich aber umgekehrt verhält, sollte dies beachtet werden. An dieser Stelle versagt schließlich das getestete Merkmal.

7.2.3 Abstandsbalken

Um für ein Bild zu zeigen, dass alle seine Abstände im Merkmalsraum eine korrekte Einordnung der Aufnahmeposition ermöglichen, ist eine andere Darstellung wünschenswert. In Abbildung 7.3 ist für jedes Bild (Abszisse) die Größe aller seiner Abstände eingezeichnet. Dabei sind die beiden Abstände, die zu gleich weit entfernten Aufnahmepositionen gehören, durch eine Linie verbunden. Damit das Diagramm nicht unübersichtlich wird, sind nur die ersten zehn Bilder eingezeichnet.

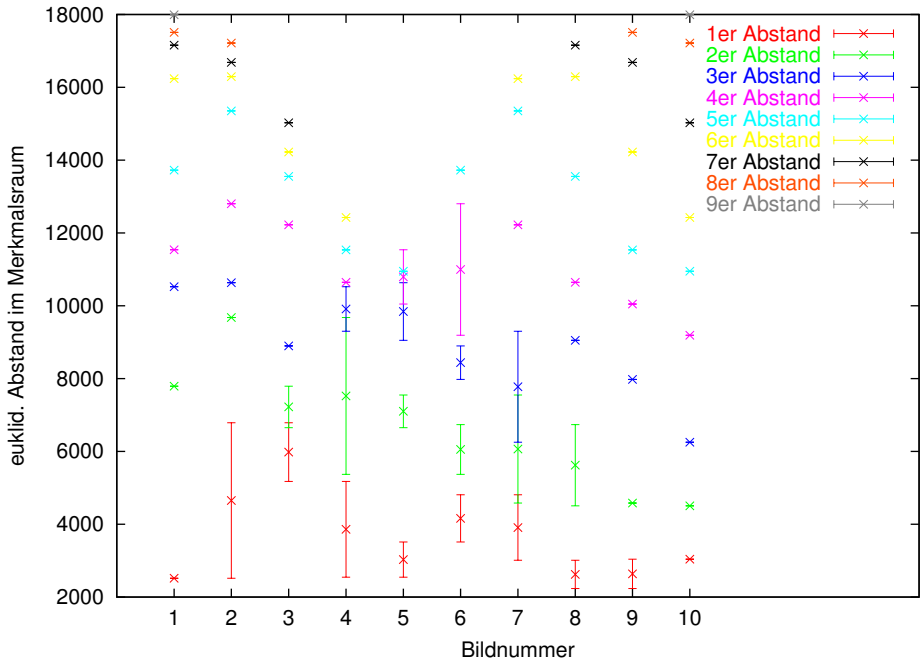

Abbildung 7.3: Abstände jeden Bildes zu jedem anderen

Der rote Balken bei Bild 2 besagt also, dass einer der direkten Nachbarn (Bild 1 oder 3) im Merkmalsraum ca. 2200 entfernt ist und der andere ca. 6800. Das alle anderen Markierungen dieses Bildes (grün, blau usw.) sich weiter oben befinden, zeigt an, dass alle nicht direkt benachbarten Bilder auch im Merkmalsraum weiter entfernt sind als die direkten. Der Abstand des entfernteren direkten Nachbarn (das obere Ende des roten Balkens) wird im Folgenden Radius $r(i)$ um Bild i genannt, in dem sich idealerweise kein anderer Merkmalsvektor befinden sollte.

Hier ist deutlich zu sehen, dass bei Bild 3 ein Problem besteht, denn die beiden untersten Balken (der 1er- und der 2er-Abstand) überschneiden sich. Das bedeutet, dass einer der direkten Nachbarn auf der Abszisse (Bild 2 oder 4) im Merkmalsraum weiter entfernt ist als einer der zweiten Nachbarn (Bild 1 oder 5).

Ein Merkmal ist dann ideal, wenn es bei keinem der Bilder zu einer solchen Überlappung der Abstände kommt. Wenn nur die kleineren Abstände (z.B. 1er, 2er und 3er) frei von solchen Problemen sind, so eignet sich der Merkmalsraum wenigstens, um ein Bild innerhalb dieser kleinen Umgebung einzuordnen.

7.2.4 Interpolierbarkeit

Die zur Abschätzung der Qualität des Merkmals wichtigen Informationen sind also die Überschneidungen der Balken, bzw. die Frage, bis zu welchem Abstand – beim kleinsten begonnen – die Abstände sich nicht überlappen. Diese Information ist in Abbildung 7.4 dargestellt.

Hier kann für jedes Bild abgelesen werden, bis zu welchem Nachbarn (bzgl. der Aufnahmeposition) die Abstände im Merkmalsraum eine korrekte Einordnung ermöglichen. Die Balkenlänge 2 für Bild 5 bedeutet also, dass die beiden 1er-Abstände kleiner sind als die 2er-Abstände und diese wiederum kleiner sind als alle anderen. Abbildung 7.3 bestätigt dies und zeigt auch, dass einer der 3er-Abstände größer ist als einer der 4er. Daraus ergibt sich, dass die Aufnahmeposition von Bild 5 aus seinen zweiten Nachbarn interpoliert werden kann, aber nicht mehr aus den dritten oder weiter entfernten. Die Zahl über dem Balken gibt an, welches Bild zu dem Problem führt. Bei Bild 5 ist also ein zu geringer Abstand zu Bild 9 der Grund, warum es anhand der dritten Nachbarn nicht korrekt eingeordnet würde.

Für Bilder am Rand (z.B. Bild 1) können nur Abstände zu einer Seite hin überprüft werden, daher fällt deren Bewertung meist unerwartet gut aus. Aus Abbildung 7.3 wäre für Bild 1 sogar zu erwarten, dass 9 Nachbarn passend steigende Abstände ergeben. Das in Abbildung 7.4 nur der Wert 5 eingezeichnet ist, liegt daran, dass Bild 23 (welches in Abbildung 7.3 nicht berücksichtigt wurde) einen kleineren Abstand hat als Bild 7. Dies ist im Höhenliniendiagramm (Abb. 7.2) zu erkennen.

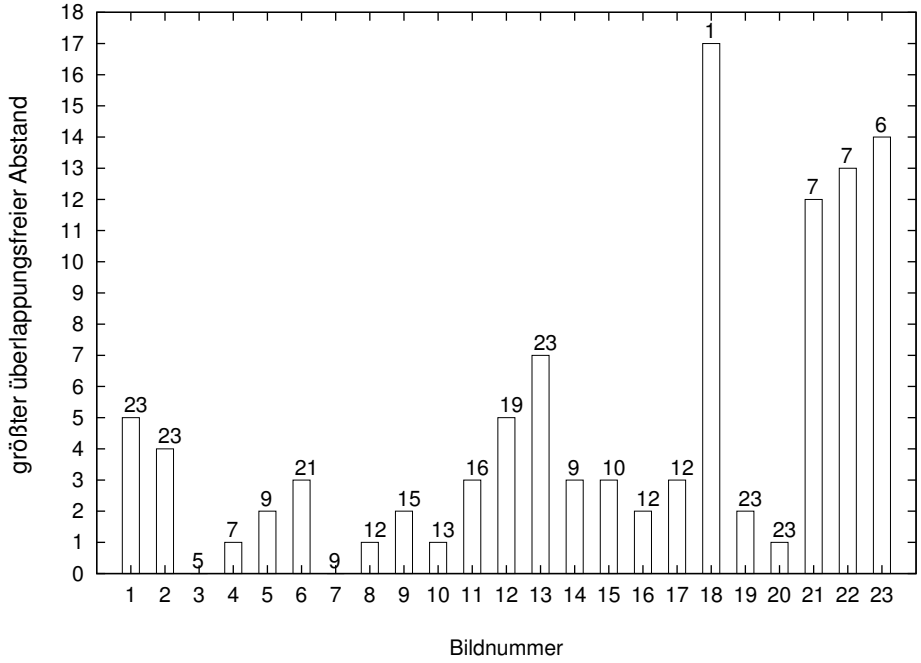

Abbildung 7.4: Interpolierbarkeit der Aufnahmepositionen anhand des Merkmals (die Zahlen über den Balken geben an, welches Bild das Problem verursacht)

Bei einer Bildserie, für die der größte überlappungsfreie Abstand (im Folgenden auch *Interpolierbarkeit* genannt) *aller* Bilder größer oder gleich 2 ist, kann ein Datum auch dann noch korrekt eingeordnet werden, wenn nur jedes zweite Bild zur Verfügung steht. Im obigen Beispiel trifft dies für die Teilserie von Bild 11 bis Bild 19 zu.

7.2.5 Unterteilung der Bildserie

Die Bilder 3 und 7 verursachen ein Problem, denn ihr Abstand im Merkmalsraum ermöglicht es nicht, sie korrekt einzuordnen. Solch ein Problem kann evtl. gelöst werden, wenn die Bildserie in **Teilserien** eingeteilt wird, innerhalb derer alle Bildpositionen interpolierbar sind. Dann ist ein zusätzliches anderes Merkmal erforderlich, um ein Bild seiner Teilserie zuzuordnen, aber innerhalb dieser funktioniert dann das untersuchte Merkmal.

Um die Stellen in der Gesamtserie zu finden, an denen sie geteilt werden sollte, helfen die Zahlen über den Balken. Die Teilung sollte schließlich so erfolgen, dass die Bilder, deren Abstände zueinander problematisch sind, in verschiedene Teilserien fallen. Die Beispielserie müsste also an zwei Stellen geteilt werden: Zwischen Bild 3 und Bild 5 sowie zwischen Bild 7 und Bild 9.

Es ist wünschenswert, ein Merkmal zu finden, dass möglichst wenige Unterteilungen erfordert, denn für die Zuordnung zur richtigen Teilserie müssen schließlich wieder andere Merkmale herangezogen werden.

7.2.6 Vergleichende Messung

Die Interpolierbarkeitsdiagramme zweier Merkmale geben die Möglichkeit, die Nützlichkeit zu vergleichen. Dabei ist die Länge des kleinsten Balkens das wichtigste Maß, denn diese gibt an, über wieviele Positionen an der schwierigsten Stelle interpoliert werden kann.

Falls bei beiden Merkmalen die Länge des kleinsten Balken Null ist, kann mit dem Interpolierbarkeitsdiagramm kein Vergleich mehr erfolgen. Um die Merkmale dennoch vergleichen zu können, wird die *relative Überlappung* U eines Merkmals bzgl. einer Bildserie definiert:

$$U = \sum_{i=1}^{N} \frac{1}{r(i)} \sum_{j=1}^{N} d(i,j) \tag{7.1}$$

$$d(i,j) = \begin{cases} r(i) - Abst(i,j) & \text{falls } Abst(i,j) < r(i) \text{ und } |j-i| > 1 \\ 0 & \text{sonst} \end{cases} \tag{7.2}$$

$$r(i) = max(Abst(i,i-1), Abst(i,i+1)) \tag{7.3}$$

Hierbei steht N für die Anzahl der Bilder, $Abst(i,j)$ für den Abstand der Bilder i und j im Merkmalsraum, r für den größten Abstand (im Merkmalsraum) zu einem der direkt benachbarten Bilder und d für die absolute Überlappung.

Die relative Überlappung eines Merkmals auf einer Bildserie ist gleich 0, wenn alle Positionen anhand des Merkmals aus den anderen Positionen interpoliert werden können. Je mehr sich bei einem Bild der 1er-Abstand mit einem oder mehreren der anderen (eigentlich größeren) Abstände überschneidet (vgl. Abb. 7.3), desto größer wird die relative Überlappung. Dabei verhindert der Faktor $1/r(i)$, dass die absolute Größe der Abstände im Merkmalsraum mit einfließt, denn diese hat für die Qualität eines Merkmals keine Bedeutung.

Die Überlappung höherer Abstände (z.B. der 2er mit dem 3er bei Bild 4) wird bei der Berechnung von U nicht berücksichtigt, da Merkmale, bei denen der 1er Abstand überlappungsfrei ist, nach ihrer Interpolierbarkeit bewertet werden sollten.

8

Experimentelle Ergebnisse

8.1 Verwendete Beispieldaten

8.1.1 Bildaufnahme

Um Algorithmen zu vergleichen, die Merkmale aus Bildern extrahieren, wurden mit dem in Kapitel 3 beschriebenen System mehrere Serien von Testbildern aufgenommen. Der Roboter wurde dazu entlang eines Flures der Universität gefahren, und in festen Abständen wurden Momentaufnahmen gemacht. Die Parameter, durch die sich die Serien voneinander unterscheiden, waren dabei: die Kamera, der Abstand der Bilder von einander, die Anzahl der gemachten Bilder sowie der verwendete Teil des Flures. Tabelle 8.1 zeigt die genauen Daten.

	Serie I	Serie II	Serie III	Serie IV	Serie V
Kamera	Sony EVI-G21		Hitachi HV-C20		
Auflösung	768×576 Pixel				
Abstand	50 cm		10 cm		
Anzahl	22 Bilder	21 Bilder	41 Bilder	23 Bilder	32 Bilder

Tabelle 8.1: Aufnahmedaten der Testbildserien

Die von den Kameras gelieferten Bilder zeigen den Blick von unten zum Spiegel (siehe Abb. 8.1). Der interessante Bereich eines solchen Bildes ist der Kreis, in dem der Spiegel zu sehen ist.

Abbildung 8.1: Blick der Kamera auf den Spiegel

8.1.2 Umrechnung in Panoramabilder

Wegen der hyperbolischen Form des Spiegels zeigt das Kamerabild die Umgebung nur verzerrt. In der Realität gleichgroße Flächen nehmen in so einem Bild also unterschiedlich viel Platz ein, je nachdem, von welchem Teil des Spiegels sie projiziert werden. Um dies zu korrigieren, werden die Bilder in eine Panoramaform umgerechnet, die in etwa so aussieht, als ob von der Spiegelmitte aus in alle Richtungen zugleich geblickt würde (siehe Abb. 8.2). Das genaue Verfahren, mit dem ein Programm von S. Köper die Umrechnung der Testserien durchführt, ist in [Bög00] dokumentiert.

Abbildung 8.2: In Panoramaformat umgerechnetes Bild

Die errechneten Panoramabilder haben ein Format von 1000×200 Pixeln und liegen im RGB-Format vor. Da für jeden Farbkanal ein Byte verwendet wird, sind sie 600 000 groß.

8.1.3 Berechnung der Merkmale

Für die Untersuchung wurden zunächst die in den Kapiteln 4 bis 6 erklärten Merkmale aus den Bildern der fünf Serien ermittelt. Die Programme hierzu sind im Anhang dieser Arbeit aufgeführt. Anschließend wurden die in Kapitel 7 vorgestellten Diagramme und die relative Überlappung berechnet.

Für fast alle Merkmale dienten Grauwertbilder als Eingangsdaten. Nur die ganzen Farbbilder und die Farbmerkmale wurden aus den RGB-Bildern gewonnen.

Die Kantenbilder und diejenigen mit hervorgehobenen Kanten wurden mit dem Programm *susan* von Stephen Smith (siehe auch [SB97]) erstellt.

Bei den skalierten Bildern bezieht sich die angegebene Skalierung auf die Länge und Breite der Bilder. Eine Skalierung auf 50% hat daher eine Reduzierung der Dimension auf 25% zur Folge.

Die Rechenzeit zur Berechnung der Hauptkomponenten aus ganzen Bildern ist – wie erwartet – sehr hoch. Außerdem ist die Anzahl der Bilder pro Serie (20 bis 40) zu gering, um eine statistische Analyse im 200 000 dimensionalen Raum der Grauwertbilder (Farbbilder wären noch aufwendiger) zu ermöglichen. Daher werden für die Experimente nur skalierte Bilder als Eingangsdaten der PCA verwendet.

Das Programm *learn* zum Finden der Hauptkomponenten verwendet die iterative Methode nach Gleichung 6.8. Anschließend bildet es die Eingangsvektoren auf die Hauptkomponenten ab.

8.2 Ergebnisse

8.2.1 Reduktion der Beleuchtungseinflüsse

Die Methode, einen interessanten Bereich des Histogramms auf eine Normbreite zu strecken, war nicht erfolgreich. Da die Grauwerthistogramme am Rand zum Teil sehr flach abfallen, gelingt es nicht einmal, manuell einen geeigneten Histogrammbereich festzulegen, der eine einheitliche Helligkeit der Bilder zur Folge hätte. Verschiedene Versuche, einen Grenzwert automatisch zu bestimmen (über einen Anteil an der max. Helligkeit, den Anteil beachteter Bildpunkte o.ä) führten ebenfalls nicht zum Erfolg. Abbildung 8.3 zeigt eines der problematischen Histogramme.

Schräg einfallendes Licht verleiht geraden Wänden einen kontinuierlichen Farbverlauf, der es (auch manuell) unmöglich macht, eine Farbe der Wand zu bestimmen, die dann zu einer **Normfarbe** modifiziert werden könnte. Bei der Fußbo-

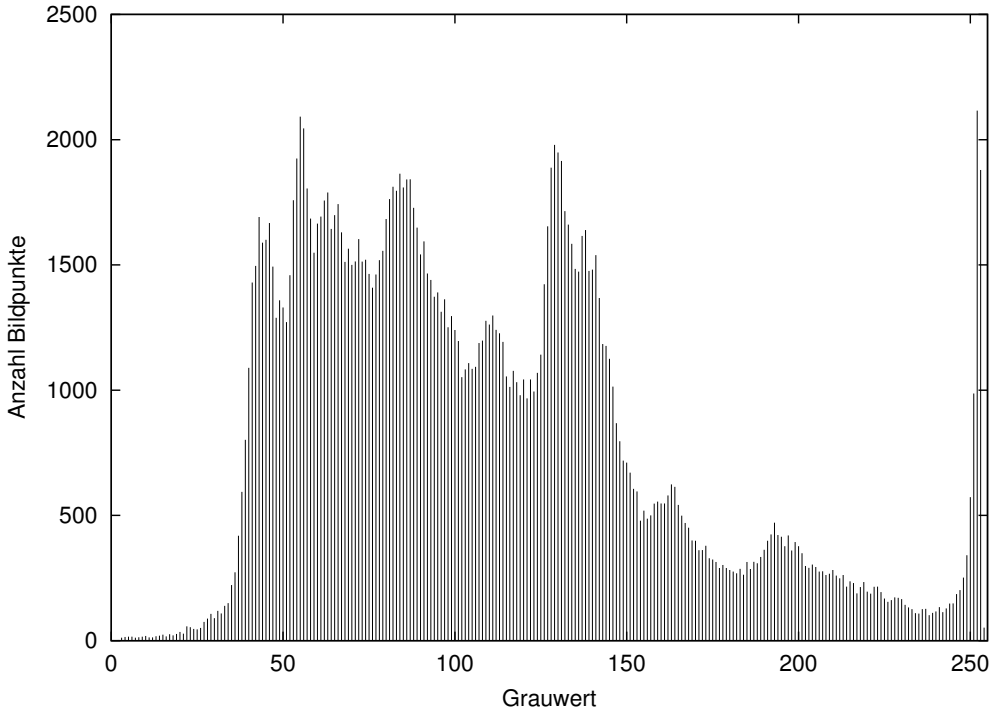

Abbildung 8.3: Flach auslaufendes Grauwerthistogramm

Abbildung 8.4: Originalbild vor der Normierung

denfarbe bewirken Lichtreflexionen ähnliches.

Auch für andere Verfahren ist es problematisch, dass manche Flächen dadurch, dass das Licht auf sie mit unterschiedlichem Winkel trifft, nicht so homogen sind, wie sie dem Betrachter erscheinen. Eine einfache Skalierung der Helligkeit kann eine solche Fläche natürlich auch nicht normieren. Die Wand links und rechts der Tür in Abbildung 8.4 zeigt dies deutlich.

Damit der Wand eine **einheitliche Farbe** zugewiesen werden könnte, müsste sie durch andere Merkmale (z.B. eine kantenbasierte Segmentierung) vom Rest des Bildes isoliert werden. Dies ist für vorverarbeitende Normierung aber viel zu

komplex und aufwendig. Eine andere Idee ist, die zur Wand gehörenden Punkte anhand ihres Farbtons zu erkennen. Dies wird im folgenden Abschnitt über Farben untersucht.

Dadurch, dass im Rundumbild sowohl dunkle als auch überbelichtete Teile enthalten sind, bewirkt die **Normierung der Gesamtintensität** nur wenig. Je mehr überbelichtete Bildpunkte vorhanden sind, desto stärker wird die Helligkeit gesenkt, obwohl dies für andere Bildteile häufig falsch ist. Die Gesamthelligkeit ist schließlich die Größe, die bereits in der Kamera zur Helligkeitskorrektur verwendet wird, und daher über die Bildserie hinweg recht konstant bleibt.

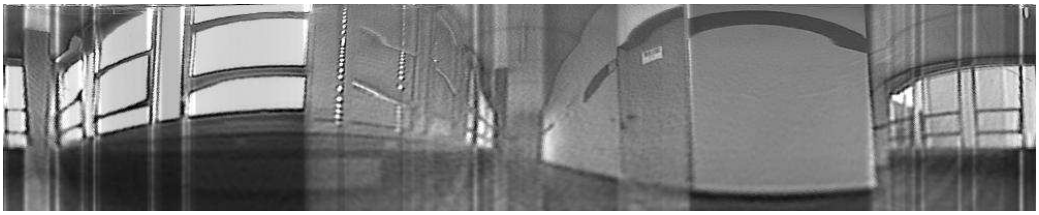

Abbildung 8.5: Spaltenweise normiertes Bild

Dies führt zur **spaltenweisen Energienormierung** (Bild 8.5). Hier ist ist die Wand links und rechts der Tür deutlich einheitlicher als im Originalbild. Jede Bildspalte einzeln zu normieren gleicht auch die Histogramme an, führt aber zu unerwünschten senkrechten Kanten. Dies ist im Beispielbild über der Tür besonders deutlich zu sehen.

Dies alles zeigt, dass zur Lokalisation vorzugsweise beleuchtungsunabhängige Merkmale verwendet werden sollten, denn eine perfekte Normierung ist schwierig.

8.2.2 Extraktion der Farbinformation

Wie in Kapitel 5 erläutert, gibt die durchschnittliche Sättigung $\overline{S}$ an, wieviel Farbinformation ein Bild enthält. In Tabelle 8.2 wird der durchschnittliche Wert von $\overline{S}$

	Serie I	Serie II	Serie III	Serie IV	Serie V
Durchschnitt der Serie	7,7%	7,2%	8,9%	7,9%	7,0%
kleinste durchschn. Sättig.	5,5%	6,0%	6,5%	6.4%	5.3%
größte durchschn. Sättig.	9.3%	8.5%	11%	9.4%	9.4%

Tabelle 8.2: Durchschnittliche Sättigung der Bilder der Testbildserien

jeder Bildserie angegeben sowie die durchschnittliche Sättigung des am stärksten gesättigten Bildes und des am wenigsten gesättigten Bildes. Diese Werte zeigen, dass die Farben aller Bilder eine sehr geringe Sättigung aufweisen.

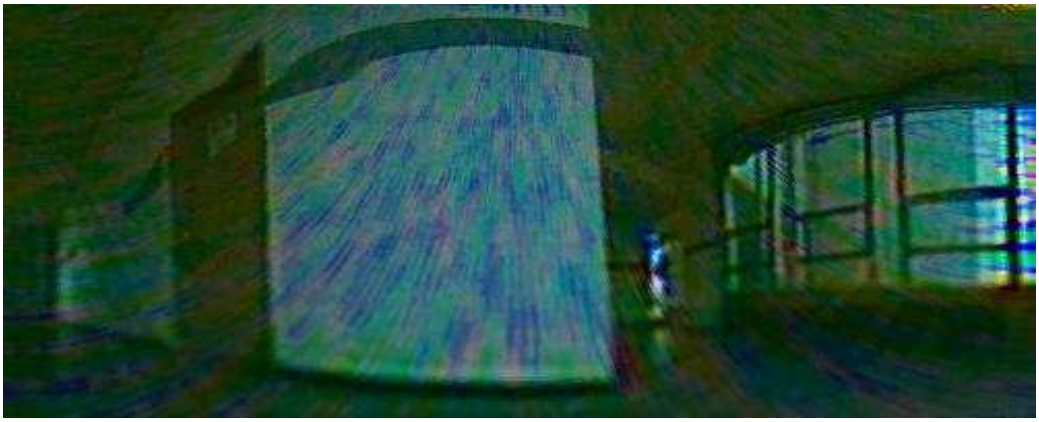

Abbildung 8.6: Sättigung auf das 8fache erhöht

Die Bilder beider Kameras weisen ein deutliches Farbrauschen auf. Um dies sichtbar zu machen, wurde in Abbildung 8.6 die Sättigung auf das 8fache erhöht. Das Muster auf der Wand neben der Tür zeigt die Störung deutlich. Wegen ihr ist die **Farbtoninformation** von Pixeln mit einer Sättigung unter 5% nutzlos.

Abbildung 8.7: Sättigung der nutzbaren Pixel auf das 4fache erhöht

Aus diesem Grund liefern bei einigen Bildern weniger als 50% der Bildpunkte einen verwertbaren Farbton. Diese sind in Bild 8.7 mit einer auf das 4fache verstärkten Sättigung dargestellt. Um das Farbrauschen nicht mit zu verstärken, sind Farben mit einer Sättigung unter 5% weiß belassen worden.

Es wird deutlich, dass die meiste Farbinformation durch einfallendes Licht verursacht wird: Im linken Bildteil scheint blaues Tageslicht durch ein Fenster, welches an der gegenüberliegenden Wand (rechter Bildteil) reflektiert wird. In dem Teil des Ganges, der nur durch Kunstlicht beleuchtet wird, sind die Wände dagegen eher gelb-rot.

Farbtonhistogramme, die aus Aufnahmen einiger Büros berechnet wurden, zeigen ebenfalls nur Maxima für die Farben des auftretenden Lichts und nicht – wie erhofft – für spezifische Farben in den Räumen. Sie eignen sich also nicht zur Unterscheidung der Räume, sondern allenfalls zur Unterscheidung der Lichtverhältnisse. Zwischen den Abbildungen 8.8, 8.9 und 8.10 sind zwar Differenzen

zu erkennen, aber nur in der Anzahl der orangen, der blauen und der daraus gemischten Punkte. Diese stammen also vom Licht und ändern sich daher unter anderem abhängig von der Tageszeit.

Aus Bildern, in denen sich Regionen verschiedener Farbe befinden, lassen sich im allgemeinen die **Kanten** zwischen diesen Regionen gut extrahieren. Wegen der geringen Sättigung der Flurbilder und des damit verbundenen geringen Informationsgehalts der Farbtonbilder werden aber bei den hier verwandten Bildserien weniger deutliche Kanten gefunden als im Grauwertbild. Die in Bild 8.7 gezeigte Farbtonverteilung macht deutlich, warum im Farbbild sogar falsche Kanten gefunden werden. Ein empfindlich reagierender Farbkantenextraktor findet die Grenzen zwischen den unterschiedlich beleuchteten Teilen der Wand und teilt dadurch die – eigentlich kantenfreie Wand – in mehrere Abschnitte ein.

Da die Bilder nur sehr wenig Sättigung aufweisen, wird durch eine Transformation in den YUV- oder den Lab-Farbraum nicht viel gewonnen. In beiden unterscheidet sich die Farbe der meisten Pixel nur wenig von weiß.

Der beste **problemangepasste Farbraum** bei Bildern mit einer so geringen Sättigung ist der Y-Kanal (siehe Gl. 5.9). Denn der Informationsverlust, wenn anstelle der Farbbilder nur Grauwertbilder verwendet werden, ist sehr gering, aber der Speicherverbrauch sinkt auf ein Drittel. Anstelle der drei Kanäle für Rot, Grün und Blau genügt ein Kanal für Y.

Die Position ausschließlich anhand der Farbinformation (z.B. Farbtonhistogramme) zu bestimmen, ist problematisch, denn das Tageslicht ändert seine Farbe abhängig vom Wetter und der Tageszeit (abends sieht der gesamte Flur gelb-rot aus). Die Information kann aber zur Unterstützung verwendet werden, denn es gibt Bereiche, in die niemals Tageslicht dringt.

8.3 Vergleich der Merkmale

Um die verschiedenen Merkmale zu vergleichen, wurde das in Kapitel 7 beschriebene Verfahren zur Berechnung der relativen Überlappung angewendet. Tabelle 8.3 zeigt die Ergebnisse. Von keinem der getesteten Merkmale können alle Bilder aller Serien anhand der Abstände im Merkmalsraum korrekt geordnet werden. Die Merkmale sind den meisten Bildserien also nicht gewachsen. Ein ideales Merkmal würde sich dadurch auszeichnen, dass es für alle Bildserien eine relative Überlappung von 0 erreicht.

Die verschiedenen Bildserien sind dabei offenbar unterschiedlich komplex. Serie IV scheint die am leichtesten zu ordnende zu sein, denn selbst nach einer Skalierung auf nur 20 Bildpunkte können noch alle Positionen interpoliert werden. Eine nähere Betrachtung dieser Serie zeigt, dass die Gesamtenergie der Bilder nahezu

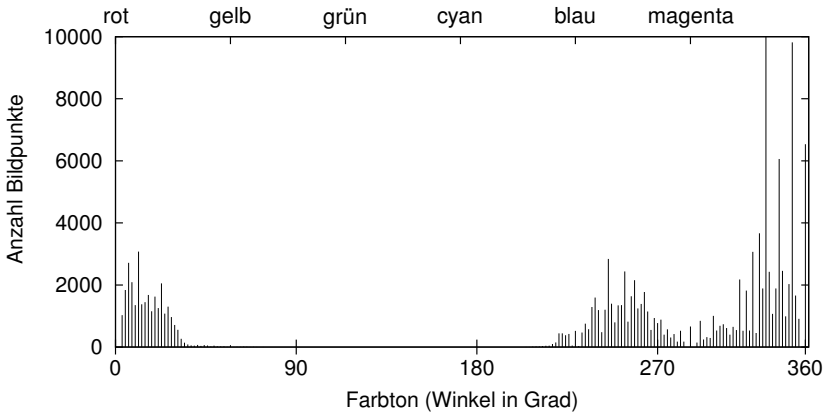

Abbildung 8.8: Farbton-Histogramm des Flures

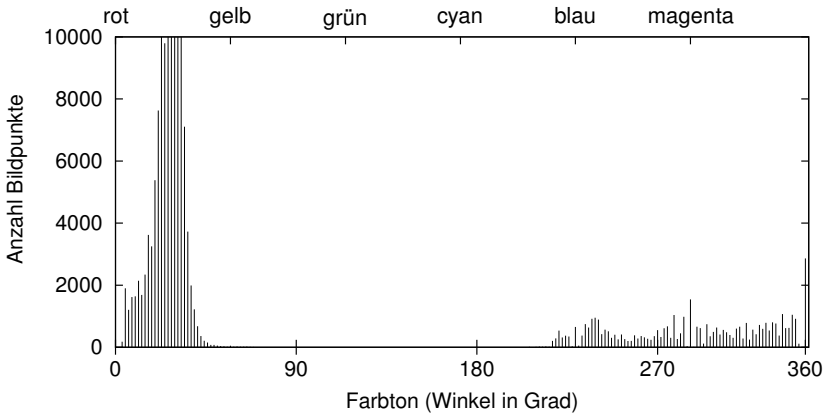

Abbildung 8.9: Farbton-Histogramm eines Büros

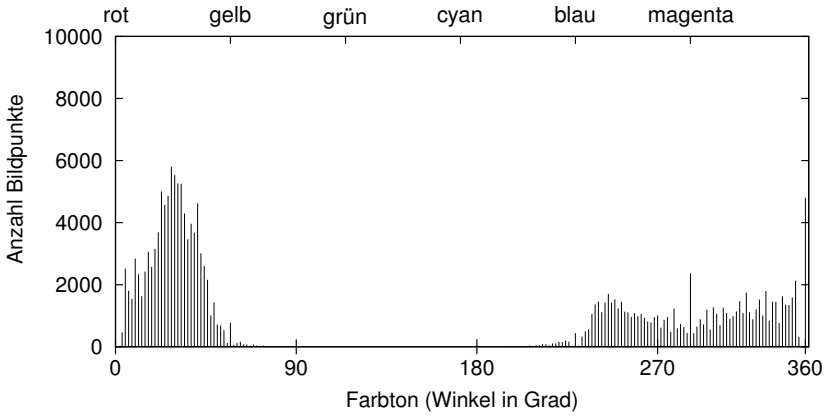

Abbildung 8.10: Farbton-Histogramm des Rechnerraumes

Merkmal	Dim.	Serie I	Serie II	Serie III	Serie IV	Serie V
Anzahl		22	21	41	23	32
ganze Farbbilder	600 000	2.5	5.4	0.35	0	2.8
ganze Graubilder	200 000	2.4	5.3	0.39	0	2.6
obere Hälfte	100 000	6.7	5.2	0.2	0	4.7
untere Hälfte	100 000	1.6	6.0	0.72	0.0083	0.68
Bildmitte	100 000	3.5	7.0	1.7	0	3.8
energienormiert	200 000	3.1	6.1	0.36	0	0.54
spaltenw. energ.	200 000	2.6	2.2	0.49	0	0.2
Kantenbild	200 000	3.3	2.4	2.6	0.52	0.55
hervorgeh. Kanten	200 000	1.0	2.8	0.54	0.3	0.87
Farbtonbild	200 000	6.5	1.8	0.43	0.025	1.5
Grauhistogramm	256	4.8	8.3	5.7	0.067	32.0
Farbtonhist.	256	18.0	37.0	37.0	26.0	31.0
Sättigungshist.	256	12.0	22.0	10.0	9.7	15.0
skaliert auf 50%	50000	2.4	5.4	0.39	0	2.6
skaliert auf 25%	12500	2.5	5.7	0.38	0	2.8
skaliert auf 5%	500	2.8	7.1	0.53	0	4.6
skaliert auf 1%	20	4.5	9.6	1.8	0	14.0
skal. 25% &PCA	10	3.7	6.3	0.78	0	4.4
skal. 25% &PCA	5	3.6	8.9	0.069	0.12	5.5
skal. 5% & PCA	10	3.2	7.5	0.82	0	5.5
skal. 5% & PCA	5	4.0	9.5	0.2	0.27	6.5
skal. 5% & PCA	3	5.0	11.0	1.2	0.31	9.4

Tabelle 8.3: Ergebnisse des Algorithmenvergleichs: relative Überlappung der Merkmale auf den Bildserien

monoton steigt.

Die relative Überlappung zeigt, welche Merkmale wie nah daran sind, die Interpolation zu ermöglichen: **Ganze Bilder** sowie der obere oder untere Teil davon haben eine geringe relative Überlappung, aber auch eine hohe Dimension. Die **Skalierung** auf 25% (entspricht einer Skalierung der Dimension auf 6.25%) liefert nahezu ebensogute Ergebnisse mit einer recht hohen Reduktion der Dimension und einem geringen Rechenaufwand.

Durch den Einsatz der **PCA** wird der Rechenaufwand beträchtlich erhöht, aber

er ermöglicht es, die Dimension weiter zu senken. Ob die Erhöhung der relativen Überlappung, die damit einhergeht, vertretbar ist, bleibt fraglich.

Unter den **Normierungsverfahren** liefert das in Abschnitt 4.1.4 entwickelte spaltenweise arbeitende die besten Ergebnisse. In einigen Serien erreicht es eine geringere relative Überlappung als die ganzen Bilder. Dies gilt auch für die Bilder mit **hervorgehobenen Kanten**. Beides ist ein klares Indiz dafür, dass Beleuchtungsunterschiede ein bedeutender Störfaktor sind, denn Helligkeitsnormierung und Kantenbilder erzeugen Bilder, die gegen unempfindlich gegenüber Beleuchtungsänderungen sind.

Die **Farbmerkmale** (Sättigung und Farbton) ergeben die schlechtesten Ergebnisse, was aufgrund der wenig gesättigten Farben der Bilder auch nicht anders zu erwarten ist.

9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

In dieser Diplomarbeit wurden Algorithmen, die Merkmale aus omnidirektionalen Bildern berechnen, miteinander verglichen. Das Ziel dabei war, geeignete Algorithmen und Merkmale zur Lokalisierung eines mobilen Robotersystems zu finden.

Nach der Beschreibung des Systems, in dem die Algorithmen eingesetzt werden sollen, folgte die Erklärung einiger Merkmale. Dann wurde ein Verfahren vorgestellt, dass die Korrelation zwischen Merkmalsvektoren und Aufnahmepositionen einer Bildserie bewertet. Mit Hilfe dieses Verfahrens und einiger anderer Beobachtungen erfolgten dann eine Bewertung und ein Vergleich der Merkmale unter Berücksichtigung der Eigenschaften der jeweiligen Extraktionsalgorithmen.

Die wichtigsten Ergebnisse dieser Arbeit sind:

- Es sollten vorzugsweise Merkmale verwendet werden, die unempfindlich gegenüber Beleuchtungsänderungen sind.
- Die Verwendung von Farbbildern und Farbmerkmalen ist nur in einer Umgebung mit Farben, die stärker gesättigt sind als diejenigen auf den hier verwendeten Testbildern, sinnvoll.
- Unterschiedliches, einfallendes Licht enthält viel Farbinformation und überlagert damit die Farbinformation der aufgenommenen Umgebung.

- Durch eine Skalierung der Bilder kann ihre Dimension auf weniger als 10% reduziert werden, ohne wichtige Informationen zu verlieren.
- Keines der untersuchten Merkmale ist in der Lage, alle Testbildserien korrekt zu ordnen. Also kann eine Lokalisierung nur durch eine Kombination mehrerer Merkmale oder den Einsatz anderer Methoden gelingen.

9.2 Ausblick

Die Untersuchung von Algorithmen und Merkmalen zur visuellen Lokalisierung ist mit dieser Arbeit noch nicht abgeschlossen. Weitere Schritte, um eine robuste Lokalisierung zu ermöglichen, können sein:

- Untersuchung weiterer Algorithmen und deren Merkmale mit dem vorgestellten Vergleichsverfahren.
- Untersuchen weiterer Kombinationen von Merkmalen.
- Evtl. Verwendung einer anderen Metrik als der euklidischen im Vergleichsverfahren, je nachdem, was für ein Positionsschätzer eingesetzt werden soll.
- Zusätzliche Versuche mit einem Positionsschätzer (z.B. Fuzzykontroller) zur Verifikation der Ergebnisse des vorgestellten Verfahrens.
- Erweiterung des Vergleichsverfahrens auf zweidimensionale Bildmengen statt eindimensionaler Bildserien.
- Testen der Eigenschaften der Algorithmen und Merkmale anhand einer Bildserie mit *unterschiedlichen* Bildern, die an der *gleichen* Aufnahmeposition gemacht wurden.
- Die Ausgabe des Vergleichsprogramms bzgl. der Interpolierbarkeit kann zur Einteilung der Umgebung verwendet werden, um Teilbereiche zu bestimmen, in denen eine lokale Positionsbestimmung erfolgen soll.

Zusammen mit der weiteren Forschung werden die Ergebnisse dieser Arbeit hoffentlich dazu beitragen, dass autonome, mobile Roboter dem Menschen in Zukunft bei einfachen Tätigkeiten helfen.

Eingesetzte Software

Helligkeitsnormierung

Eigenes Programm **norm** für Gesamtnormierung:
`norm < input.pgm > output.pgm`
Eigenes Programm **normcolumn** für spaltenweise Normierung:
`normcolumn < input.pgm > output.pgm`
Beiden Programmen kann die Solldurchschnittshelligkeit eingestellt werden:
`norm 130 < input.pgm > output.pgm`
Der Vorgabewert ist 120.

Kantenextraktor

SUSAN Version 2l von Stephen Smith (http://www.fmrib.ox.ac.uk/∼steve):
normale Kanten:
`susan input.pgm output.pgm -p -t 15`
Grauwertbild mit hervorgehobenen Kanten:
`susan input.pgm output.pgm -e -t 15`

Histogrammextraktion

Eigenes Programm **histo**: In einem X11-Fenster betrachten:
`histo -x < input.pgm`
Im PNG-Format in eine Datei schreiben:
`histo -p < input.pgm > output.png`
Mehrere solcher PNG-Dateien können als Trickfilm betrachtet werden mit:
`xv -wait 1 *.png`
Als ASCII-Zahlenfolge in eine Datei schreiben:
`histo -a < input.pgm > output.data`
Diese Daten können mit gnuplot und `plot "output.data"` betrachtet werden.

Farbraumumwandlung

Eigenes Programm **extract**: Farbton (H aus HSV) extrahieren:
`extract -h < input.ppm > output.pgm`
Sättigung (S aus HSV) extrahieren:
`extract -s < input.ppm > output.pgm`
UV-Ebene extrahieren:
`extract -u < input.ppm > output.ppm`

PCA-Programme

Eigenes Programm **learn** unter Verwendung einer Bibliothek aus [Sch98]:
`learn inputDim num outputDim input.vec output.pca`

Bedeutung der Parameter:

`inputDim`	Dimension der Eingabedaten
`num`	Anzahl der Eingabedaten
`outputDim`	Anzahl zu findender Hauptkomponenten
`input.vec`	Datei mit Eingabevektoren (`inputDim`·`num` durch Leerzeichen getrennte Floats)
`output.pca`	Dateiname zur Speicherung der Hauptkomponenten und der auf diese projizierten Eingabevektoren

Eigenes Programm **lookup** projiziert einen neuen Vektor auf die Hauptkomponenten und berechnet dessen Abstand zu den von **learn** projizierten Eingabevektoren:
`learn learned.pca input.vec`

Bildreduktion

Skalierung eines Bildes (skaliert die Dimension auf 1/4):
`pnmscale 0.5 < input.pgm > output.pgm`
Ausschneiden der Bildmitte eines 1000x200 großen Bildes:
`pnmcut 250 0 750 200 < input.pgm > output.pgm`

Umrechnung in Polarkoordinaten

Perspektive von Susanne Köper, Univ. Bielefeld, AG Techn. Informatik: Mit hyperbolischem Spiegel aufgenommenes Bild (Mittelpunkt: (x,y), Radius: r) in Panoramabild (Polarkoordinatendarstellung) umrechnen:
`Perspektive -i in.ppm -o out.ppm -s -x382 -y294 -r315`

Abstandsprüfprogramm

Eigenes Programm **matrix** zur Berechnung und Veranschaulichung der Abstände einer Menge von Bildern oder Merkmalsvektoren:
`matrix dim num input mode output`

Bedeutung der Parameter:

`dim`	Dimension der Eingabedaten
`num`	Anzahl der Eingabedaten
`input`	Quelle der Eingabedaten (entweder ein Dateiname oder `-r` zum Einlesen kompletter Bilder über stdin oder `-g` zum verwenden der Helligkeit über stdin eingelesener Farbbilder)
`mode`	Auswahl des Arbeitsmodus (1. Buchstabe Diagrammart, 2. Buchstabe Metrik (immer `e` für euklid. Abstand), 3. Buchstabe Ausgabeformat)
`output`	Basisdateiname für Ausgabedateien

Diagrammarten:

`g` Abstandsmatrix als Gebirge
`h` Höhenlinien des Gebirges
`f` Abstände in Form von Fehlerbalken
`i` Interpolierbarkeit mit Problemnummern
`a` alle Diagramme ausgeben

Ausgabeformate:

`x` X11-Fenster
`e` Encapsulated Postscript
`c` Farbiges Encapsulated Postscript

Beispiel (Interpolierbarkeit der Grauwertbilder aus 23 Farbbildern als eps):
`cat *.ppm | matrix 200000 23 -g iee graubild`

Konvertierungsprogramme

Konvertieren von PPM nach JPEG:
`cjpeg input.ppm > output.jpeg`
Konvertieren von JPEG nach PPM:
`djpeg input.jpeg > output.ppm`
Konvertieren von PPM nach PGM (mit $Y = 0.299\,R + 0.587\,G + 0.114\,B$):
`ppmtopgm input.ppm > output.pgm`
Konvertieren von PGM nach PPM (alle Kanäle auf den gleichen Wert setzen):
`pgmtoppm 1,1,1 input.pgm > output.ppm`

Literaturverzeichnis

[Ber00] BERGER, CHRISTIAN: *Ein robuster, farbbasierter Objekterkenner zur Roboterfeinpositionierung in einem realen Laborumfeld*. Diplomarbeit, Universität Bielefeld, 2000.

[Bög00] BÖGERSHAUSEN, SVEN: *Merkmalsgewinnung mittels eines omnidirektionalen Sichtsystems zur Lokalisation mobiler Roboter*. Diplomarbeit, Universität Bielefeld, 2000.

[DDPC99] DROCOURT, CYRIL, LAURENT DELAHOCHE, CLAUDE PEGARD und ARNAUD CLERENTIN: *Mobile Robot Localization Based on an Omnidirectional Stereoscopic Vision Perception System*. In: *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, Seiten 1329–1334, Detroid, Michigan, May 1999.

[Fel92] FELLNER, W. D.: *Computergrafik*. BI-Wiss.-Verl., Mannheim, Leipzig, Wien, Zürich, 2. Auflage, 1992.

[Fis86] FISCHER, GERD: *Lineare Algebra*. Vieweg, Braunschweig, Wiesbaden, 9. Auflage, 1986.

[FR98] FORD, ADRIAN und ALAN ROBERTS: *Colour Space Conversions*. Technischer Bericht University of Westminster, 1998. http://www.inforamp.net/∼poynton/PDFs/coloureq.pdf.

[HKP91] HERTZ, KROGH und PALMER: *Principal Component Analysis*. In: *Introduction to the theory of neural computation*, Kapitel 8.3, Seiten 204–209. Addison Wesley, 1991.

[Jä97] JÄHNE, BERND: *Digitale Bildverarbeitung*. Springer, Berlin, Heidelberg, 4. Auflage, 1997.

[KOK00] KNAPEK, MARKUS, RICARDO SWAIN OROPEZA und DAVID J. KRIEGMAN: *Selecting Promising Landmarks*. In: *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, Seiten 3771–3777, San Francisco, CA, April 2000.

[Kum96] KUMMERT, FRANZ: *Bildverarbeitungsmethoden*. Vorlesungsskript, Universität Bielefeld, 1996.

[Loe55] LOEVE, M.: *Probability Theory*. Van Nostrand, Princeton, 1955.

[MK82] MURAKAMI, H. und V. KUMAR: *Efficient Calculation of Primary Images from a Set of Images*. In: *Transactions on Pattern Analysis and Machine Intelligence*, Band 4, Seiten 511–515. IEEE, 1982.

[PM92] PENNEBAKER, WILLIAM B. und JOAN L. MITCHELL: *JPEG, still image data compression standard*. Van Nostrand Reinhold, New York, 1992.

[Pra78] PRATT, WILLIAM K.: *Digital Image Processing*. Wiley, New York, 1978.

[RTG98] RUBNER, YOSSI, CARLO TOMASI und LEONIDAS J. GUIBAS: *The Earth Mover's Distance as a Metric for Image Retrieval*. Technischer Bericht Computer Science Department, Stanford University, 1998.

[SB97] SMITH, S.M. und J.M. BRADY: *SUSAN - A New Approach to Low Level Image Processing*. Int. Journal of Computer Vision, 23(1):45–78, Mai 1997.

[Sch98] SCHWERT, VOLKMAR: *Entwicklung eines selbstlernenden Fuzzy-Systems zur autonomen Navigation und Lokalisierung eines mobilen Roboters*. Diplomarbeit, Universität Bielefeld, 1998.

[SD99] SIM, ROBERT und GREGORY DUDEK: *Learning Visual Landmarks for Pose Estimation*. In: *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, Seiten 1972–1978, Detroid, Michigan, May 1999.

[UN00] ULRICH, IWAN und ILLAH NOURBAKHSH: *Appearance-Based Place Recognition for Topological Localization*. In: *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, Seiten 1023–1029, San Francisco, CA, April 2000.

[WYY98] WEI, SHIH-CHIEH, YASUSHI YAGI und MASAHIKO YACHIDA: *Building Local Floor Map by Use of Ultrasonic and Omni-directional Vision Sensor*. In: *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, Seiten 2548–2553, Leuven, Belgien, May 1998.

[YYY95] YAMAZAWA, KAZUMASA, YASUSHI YAGI und MASAHIKO YACHIDA: *Obstacle Detection with Omnidirectional Image Sensor HyperOmni Vision*. In: *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, Seiten 1062–1067, Nagoya, Aichi, Japan, May 1995.