Technische Universität Darmstadt
Fachbereich Informatik
Fachgebiet Simulation und Systemoptimierung

Entwicklung eines Stereokamerasystems zur Objekterkennung und Entfernungsbestimmung in mobilen Robotersystemen

Development of a Stereo Vision System for Object Recognition and Distance Measurement for Mobile Robots

Diplomarbeit von Andreas Klemp

Aufgabensteller:	Prof. Dr. Oskar von Stryk
Betreuer:	Dipl.-Math. Jutta Kiener
Abgabetermin:	16.08.2005

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, August 2005

Andreas Klemp

Kurzzusammenfassung

Mobile Robotersysteme müssen ein Modell ihrer Umgebung erstellen, um Hindernisse oder Missionsziele zu erkennen. Auf der Grundlage eines solchen Modells werden Aktionen geplant und ausgeführt. Neben Laserscannern, Ultraschallsensoren und weiteren Techniken bietet die Erfassung der Umwelt mit Videokameras eine flexible Alternative.

Am Beispiel des humanoiden Roboters „Mr. DD“ der *Darmstadt Dribblers* wird in dieser Diplomarbeit ein neues, passives Stereokamerasystem entwickelt, welches zur Objekterkennung und Abstandsbestimmung verwendet wird. Hierfür kommen erstmalig zwei einfache USB-Webcams zum Einsatz. Weiterhin werden verschiedene Algorithmen zur Kalibrierung des Systems, Farbsegmentierung und Positionsberechnung untersucht und die geeignetsten implementiert. Ein Hauptkriterium für die Verwendbarkeit der Algorithmen ist die begrenzte Rechenleistung mobiler Robotersysteme. Da die untersuchten Algorithmen zu langsam sind, werden schließlich zwei neue merkmalsbasierte Verfahren zur Korrespondenzberechnung entwickelt.

Die Leistungsfähigkeit des Stereokamerasystems wird in der definierten Umgebung eines *RoboCup*-Feldes demonstriert, in welcher zum Beispiel der Ball relativ zu den Kameras lokalisiert wird.

Abstract

Mobile robots need a model of their environment to detect obstacles or mission goals. Using such a model actions are planned and executed. Besides laser scanners, ultrasonic detectors and other technologies video cameras are a flexible alternative.

Using the humanoid robot „Mr. DD“ of the *Darmstadt Dribblers* a new passive stereo vision system is developed in this diploma thesis, which is used for object recognition and distance measurements. For the first time two simple USB webcams are employed for this purpose. Furthermore different algorithms for calibrating the system, color segmentation and position estimation are investigated and the most suitable are implemented. A main criterion for the applicability of the algorithms is the limited computing power of mobile robots. Two new feature based methods for calculating correspondences are eventually developed because the investigated algorithms are too slow.

The abilities of the stereo vision system are demonstrated in the defined environment of a *RoboCup* field where the ball is localized relative to the cameras as proof of concept.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Einführung 1

1.2 Aufgabenbeschreibung 2

1.3 Gliederung 4

2 Stand der Forschung **5**

2.1 Existierende Kamerasysteme 5

2.1.1 Binokulare Systeme 5

2.1.2 Monokulare Systeme 7

2.1.3 Pseudo-Stereokamerasysteme 8

2.1.4 Fazit 9

2.2 Effizienz des Bildverstehens 10

2.2.1 Farbsegmentierung 10

2.2.2 Berechnung der Disparitäten 10

2.2.3 Implementierung in Hardware 11

2.2.4 Fazit 12

3 Theorie **15**

3.1 Modellbeschreibung 15

3.1.1 Das Kameramodell 15

3.1.2 Stereoskopie 18

3.2 Epipolargeometrie 22

3.3 Rekonstruktion 24

3.3.1 Rekonstruktion mit starker Kalibrierung 24

3.3.2 Rekonstruktion mit schwacher Kalibrierung 25

3.4 Bildverstehen 26
3.4.1 Bilderzeugung 27
3.4.2 Bildvorverarbeitung 27
3.4.3 Bildverarbeitung 28
3.4.4 Bildbeschreibung 30
3.4.5 Objekterkennung 31
3.4.6 Bildanalyse 32

4 Hardware **35**
4.1 Kameras 35
4.2 Konzeption des Stereogestells 37
4.3 Konstruktion der Stereogestelle 39

5 Software **43**
5.1 Bilderzeugung 45
5.2 Kalibrierung 48
5.2.1 Videoparameter 49
5.2.2 Farbbereiche für Segmentierung 50
5.2.3 Berechnung der Projektionsmatrizen 52
5.2.4 Berechnung der Fundamentalmatrix 59
5.3 Segmentierung 65
5.3.1 Klassifizierung der Pixel 66
5.3.2 Abstraktion zu Objekten 67
5.3.3 Optimierungen 68
5.4 Korrespondenzbestimmung 69
5.4.1 Flächenbasierte Methode 69
5.4.2 Merkmalsbasierte Methoden 71
5.4.3 Optimierungen 73

6 Ergebnisse **75**
6.1 Kalibrierung 75
6.1.1 Implementierte Algorithmen 75
6.1.2 Camera Calibration Toolbox 80

6.2 Segmentierung 81
6.3 Positionsbestimmung 82
6.3.1 Flächenbasierte Methode 83
6.3.2 Merkmalsbasierte Methoden 86
6.3.3 Fazit 91

7 Zusammenfassung und Ausblick **93**
7.1 Zusammenfassung 93
7.2 Erweiterungsmöglichkeiten 94

A Technische Daten **95**
A.1 Entwicklungssystem 95
A.2 Humanoider Roboter „Mr. DD“ 95

B Systemleistung **97**

C Konfiguration von udev **101**

D Verwendung von *RoboFrame* **107**

E Transformation der Doppelebene **109**

F Inhalt der CD **111**

G Singulärwertzerlegung **113**

H Levenberg-Marquardt-Algorithmus **115**

Literaturverzeichnis **117**

Abbildungsverzeichnis

1.1 Humanoider Roboter „Mr. DD“ 2
1.2 Verwendete Kamera Philips ToUcam PRO II 3

3.1 Kameramodell 16
3.2 Projektion auf die Bildebene 17
3.3 Stereoskopie-Modell 19
3.4 Disparität zweier Pixel 22
3.5 Beispiele für Epipolargeometrie 22

4.1 Mindestabstand einer Kugel 38
4.2 Dimensionen des Aluminium-Winkelprofils 39
4.3 Dimensionen des Gestells für Fernsicht 39
4.4 Dimensionen des Gestells für Nahsicht 40
4.5 Dimensionen der Halterung 40
4.6 Stereogestelle und Befestigung am Kopf 41

5.1 Struktur von *RoboFrame* 44
5.2 Überblick über die Software-Komponenten 45
5.3 Dialog zur Überwachung des Bildverstehens 46
5.4 Dialog zur Bildkalibrierung 49
5.5 Dialog zur Farbkalibrierung 51
5.6 Ebenes Kalibrierungsmuster 53
5.7 Zur Kalibrierung verwendete Doppelebene 60
5.8 Dialog zur Stereokalibrierung 65

6.1 Stereobildpaare für Kalibrierung 76

6.2 Beispiele für Farbsegmentierung 81
6.3 Beispiel für flächenbasierte Korrespondenzbestimmung 83
6.4 Beispiel mit zufälligem Stereobildpaar 84
6.5 Fehler bei der Korrespondenzbestimmung 85
6.6 Merkmalsbasierte Disparitäten 86
6.7 *RoboCup*-Marker für die Rekonstruktion 87

E.1 Anordnung der Doppelebene und des Kameragestells 109

Tabellenverzeichnis

4.1 Technische Daten der Kamera 36
4.2 Parameter und Sichtweiten der Stereogestelle 38

5.1 Fehler bei der Berechnung der Homografien 54
5.2 Beispiel für Farbkodierung 66

6.1 Kalibrierungsfehler des Testfalls Fernsicht 77
6.2 Kalibrierungsfehler des Testfalls Nahsicht 78
6.3 Laufzeit der Farbsegmentierung 82
6.4 Rekonstruktion für verschiedene Entfernungen 87
6.5 Laufzeit der Positionsberechnung 88
6.6 Rekonstruktionsfehler bei Bewegung 89
6.7 Rekonstruktionsfehler bei verdeckten Objekten 90

A.1 Technische Daten des Entwicklungssystems 95
A.2 Technische Daten des Humanoiden 96

B.1 Ausführungszeit ausgewählter Operationen 98

C.1 Technische Daten des Testsystems für udev 101

D.1 Zur Kompilierung verwendete Symbole 108

Listingverzeichnis

5.1 Einfaches Programm zum Zugriff auf Videodaten 47
C.1 Ausgabe von udevinfo zur linken Kamera 101
C.2 Ausgabe von udevinfo zur rechten Kamera 103
C.3 Regeln von udev zur Identifizierung der Kameras 105
D.1 Verwendung der Module 108

Notation und Symbole

$\mathbb{N}$	Menge der natürlichen Zahlen
$\mathbb{Z}$	Menge der ganzen Zahlen
$\mathbb{R}$	Menge der reellen Zahlen
$a, b, c, \ldots$	Skalare
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$	Vektoren
$A, B, C, \ldots$	Matrizen
$E_n \in \mathbb{R}^{n \times n}$	Einheitsmatrix
$\alpha, \beta, \gamma, \ldots$	Winkel
$S_a, S_b, S_c \ldots$	Koordinatensysteme
x_a, y_a, z_a	Achsen des Koordinatensystems S_a
${}^a\mathbf{p} \in \mathbb{R}^n$	Koordinatenvektor dargestellt bezüglich S_a $n = 2$ in der Ebene, $n = 3$ im Raum
${}^a\hat{\mathbf{p}} \in \mathbb{R}^{n+1}$	homogene Koordinaten von ${}^a\mathbf{p} \in \mathbb{R}^n$
${}^a\mathbf{r}_b \in \mathbb{R}^3$	Ursprung von S_b dargestellt bezüglich S_a
${}^aR_b \in \mathbb{R}^{3 \times 3}$	Orientierung von S_b dargestellt bezüglich S_a
${}^aT_b = ({}^aR_b; {}^a\mathbf{r}_b) \in \mathbb{R}^{3 \times 4}$	homogene Transformation von S_b nach S_a
$A^{-T} = (A^T)^{-1} = (A^{-1})^T$	invertierte und transponierte Matrix von A
$A^+ \in \mathbb{R}^{m \times n}$	Pseudoinverse von $A \in \mathbb{R}^{n \times m}$
$\mathbf{a}_\times \in \mathbb{R}^{3 \times 3}$	antisymmetrische Matrix mit $\mathbf{a}_\times \mathbf{b} = \mathbf{a} \times \mathbf{b}$
$\text{atan2}(x, y) = \alpha$	eindeutiger Winkel mit $\tan \alpha = \frac{y}{x}$ und $0 \leq \alpha < 2\,\pi$ durch Quadranteninformation
$\text{med}(a_1, \ldots, a_n)$	Median der Werte $a_1, \ldots, a_n$
$\ker(A)$	Kern einer linearen Abbildung A
$\text{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}$	Diagonalmatrix mit den Elementen $d_1, \ldots, d_n$
$\det(A)$	Determinante einer quadratischen Matrix A
$\|A\| = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}$	euklidische Norm einer Matrix $A \in \mathbb{R}^{m \times n}$
$\text{dist}(\hat{\mathbf{p}}, \mathbf{l})$	Abstand des Punktes $\hat{\mathbf{p}}$ von der Geraden $\mathbf{l}$
$\lceil a \rceil = \min\{b \in \mathbb{Z} \mid a \leq b\}$	Skalar $a \in \mathbb{R}$ aufgerundet

Alle angeführten Dimensionen gelten, sofern im Kontext nichts anderes angegeben wird.

Kapitel 1

Einleitung

1.1 Einführung

Mobile autonome Robotersysteme sind so unterschiedlich wie vielseitig. Um sie zu perfektionieren und ihre Fähigkeiten auszuweiten, unterhalten viele Universitäten Fachgebiete, die sich mit der Robotik beschäftigen. Auch kommerzielle Forschung ist in diesem Bereich vertreten.

Alle existierenden Roboter verfügen über externe Sensoren, mit denen sie ihre Umgebung wahrnehmen. Bei Fahrzeugen sind dies häufig Ultraschallsensoren, die sich in einem Ring um das Fahrzeug befinden, Laserscanner oder Kameras. Im Gegensatz zu Fahrzeugen orientiert sich das Design humanoider – d.h. menschenähnlicher – Roboter am Menschen. Da beim Menschen ein Großteil der Informationen über die Augen aufgenommen wird, bilden hier Kameras den Schwerpunkt der externen Sensorik. Während sich Ultraschallsensoren und Laserscanner für die räumliche Wahrnehmung eignen, ist dies mit einer Kamera im Allgemeinen nicht möglich. Ohne eine dreidimensionale Erfassung der Umgebung kann ein mobiler Roboter jedoch nicht in möglicherweise fremden Umgebungen navigieren und Hindernissen ausweichen.

Aus diesem Grund beschäftigen sich viele Wissenschaftler mit der Stereoskopie. Der Herkunft aus dem Griechischen nach bedeutet *stereo* fest, räumlich, körperlich und *skopeia* Betrachtung. Stereoskopie heißt also eigentlich allgemein Raumbetrachtung. In der einschlägigen Literatur und auch in dieser Arbeit wird der Begriff jedoch synonym zum binokularen Sehen, also dem Sehen mit zwei optischen Geräten, benutzt.

Ein übliches Sichtsystem besteht aus einer Kamera und einem Computerprogramm zum Bildverstehen. Dabei entspricht die Kamera einem biologischen Auge und das Programm übernimmt die Aufgaben des Gehirns. Ein solches

Programm erkennt einzelne Objekte und stellt gegebenenfalls einen Zusammenhang zwischen ihnen her. Dreidimensionale Informationen können aus einer zweidimensionalen Abbildung nicht ohne weitere Kenntnisse über die betrachteten Objekte ermittelt werden. Verwendet man zwei Kameras für das stereoskopische Sichtsystem, so erhält man ein Stereobildpaar, das die betrachtete Szene aus unterschiedlichen Blickwinkeln zeigt. Daher wird das Programm zum Bildverstehen erweitert, um aus diesen zusätzlichen Daten eines Stereokamerasystems räumliche Informationen zu gewinnen.

1.2 Aufgabenbeschreibung

Ziel dieser Diplomarbeit ist es, ein Stereokamerasystem für den humanoiden autonomen Roboter „Mr. DD“ der *Darmstadt Dribblers* [55] zu entwickeln. Abbildung 1.1 zeigt den Humanoiden mit dem angefertigten Kamerasystem.

Abbildung 1.1: Humanoider Roboter „Mr. DD“.

Unter Verwendung zweier Webcams aus Abbildung 1.2 ist ein passives Stereokamerasystem mit starr an der Halterung befestigten Kameras zu konstruieren, dass sich an dem Kopf des Roboters befestigen lässt. Dadurch gibt es keine beweglichen Teile am Stereosystem, womit auch kein zusätzlicher

Regelungsaufwand nötig ist. Das komplette Stereokamerasystem kann mit den zwei vorhandenen Kopfgelenken des Roboters gesteuert werden, was in dieser Diplomarbeit jedoch nicht behandelt wird.

Abbildung 1.2: Verwendete Kamera Philips ToUcam PRO II.

Der Humanoide soll die Stereosicht in verschiedenen Szenarien einsetzen. Im Umfeld des *RoboCup* [60] soll er die definierten Objekte, wie den Ball und die Tore, anhand ihrer Farbe erkennen und ihre Position bestimmen. Desweiteren soll er sich in den Gängen eines Gebäudes bewegen und dabei Türen erkennen. Schließlich soll die Position von Objekten in Griffweite bestimmt werden, da auch Interaktion im Nahbereich geplant ist. Aufgrund dieser Einsatzgebiete muss ein geeignetes Stereokamerasystem konzipiert werden.

Nach dem Bau eines geeigneten Kamerasystems ist ein Computerprogramm zu entwickeln, welches die Stereobildpaare der Kameras verarbeitet. Dieses Programm muss zunächst Objekte aufgrund einer Farbsegmentierung erkennen und eine Positionsbestimmung durchführen. Die erhaltenen Daten müssen dann anderen Komponenten des Roboters zur Verfügung gestellt werden, damit diese Informationen für die Planung und Ausführung von Aktionen verwendet werden können. Darüber hinaus ist eine Schnittstelle zur Fehlersuche und Überwachung notwendig. Eine solche Datenauswertung ist auf einem anderen Rechner durchzuführen, da der Humanoide nicht über ein Display und Eingabegeräte verfügt.

Die implementierte Software zur Realisierung der beschriebenen Aufgaben soll nach Möglichkeit effizient genug sein, um auf dem System des Humanoiden (Anhang A.2) ohne externen Rechner ausgeführt zu werden.

Schließlich soll sich das Stereokamerasystem einfach an andere mobile autonome Robotersysteme, wie zum Beispiel das differentialgetriebene autonome Roboterfahrzeug Pioneer 2DX, anpassen lassen.

1.3 Gliederung

Im Folgenden werden kurz die Inhalte der verbleibenden Kapitel erläutert. In Kapitel 2 werden verschiedene bestehende Stereokamerasysteme und die Effizienz existierender Algorithmen beschrieben. Kapitel 3 enthält die theoretischen Grundlagen für das weitere Verständnis der implementierten Algorithmen. Die Hardware-Komponenten des entwickelten Stereokamerasystems sind in Kapitel 4 dokumentiert. In Kapitel 5 werden die implementierte Software und die damit realisierten Algorithmen vorgestellt. Die erzielten Ergebnisse werden in Kapitel 6 aufgeführt und bewertet. Schließlich fasst Kapitel 7 die Ergebnisse der Arbeit zusammen. Darüber hinaus werden Möglichkeiten für zukünftige Erweiterungen diskutiert.

Kapitel 2

Stand der Forschung

Die Stereoskopie ist ein umfangreiches und aktives Forschungsgebiet. Mit allen Teilbereichen und Varianten, wie zum Beispiel Konstruktion, Bildverstehen, Lösungen in Hard- und Software, Rekonstruktion, Optimierung und neuronale Netze, ist sie zu komplex, um sie vollständig im Rahmen dieser Diplomarbeit zu beschreiben. Auf den folgenden Seiten werden daher einige existierende Stereokamerasysteme und Anwendungen vorgestellt, welche die Möglichkeiten der Umwelterfassung mit zwei Kameras anreißen. Auch einige Algorithmen zum Bildverstehen werden illustriert und ihre Effizienz diskutiert. Dies gibt einen Einblick in die mögliche Leistungsfähigkeit aktueller Systeme.

2.1 Existierende Kamerasysteme

Die bestehenden Kamerasysteme werden in aktive und passive Systeme unterteilt. Die Verwendung dieser Begriffe ist dabei nicht einheitlich. Mit einem aktiven System sind je nach Quelle entweder getrennt bewegliche Kameras oder die Verwendung von strukturiertem Licht gemeint. Mit letzterem wird die betrachtete Szene in definierter Weise beleuchtet, um das Bildverstehen zu erleichtern oder zusätzliche Informationen in die Bilder einzufügen.

2.1.1 Binokulare Systeme

Ein Beispiel für ein passives System ist der Stereokamerakopf des Lehrstuhls für Steuerungs- und Regelungstechnik der Technischen Universität München [30, 40]. Dieser besteht aus zwei Kameras, die in einem festen Konvergenzwinkel montiert sind. In diesem System ist weiterhin ein Genick

mit zwei Freiheitsgraden zum Drehen um die Senkrechte zu den beiden optischen Achsen und um die Kameras verbindende Achse integriert. Dieser Kopf wird zum Beispiel auf den zweibeinigen Laufrobotern Johnnie [28] der TU München und BARt-UH [19] der Universität Hannover eingesetzt. Darüber hinaus wird das System in dem Projekt ViGWaM [29] verwendet. In diesem Projekt werden Laufroboter simuliert, die über die räumliche Wahrnehmung mittels Stereoskopie navigieren. Der Vorteil dieses Kamerakopfes ist die einfache Konstruktion, da die Kameras nicht einzeln bewegt werden.

Ein aktives Stereokamerasystem mit getrennt beweglichen Kameras wird in [59] beschrieben. Dieses verfügt über die gleichen Drehachsen im Genick wie der passive Kopf. Zusätzlich können die Kameras um jeweils eine zu den optischen Achsen Senkrechte gedreht werden. Dadurch können einzelne Objekte fixiert werden, wie es bei vielen Tieren möglich ist. Erscheint ein Objekt in beiden Bildern etwa in der Bildmitte, so befindet es sich am Schnittpunkt der optischen Achsen und die Position des Objektes kann über die Gelenkstellungen des Genicks und der Kameras berechnet werden. Mit einem solchen Stereokamerasystem kann außerdem ein bewegliches Objekt verfolgt werden, das dadurch länger in beiden Bildern des Stereobildpaares sichtbar ist. Ein ähnliches System findet sich auch in [53, 54], wobei die Genickachse entlang der Verbindungslinie der Kameras weggelassen wird. Dadurch sind nur noch Drehungen in der Ebene der optischen Achsen möglich. In aktiven Systemen dieser Art ist ein erhöhter Regelungsaufwand nötig, um die Kameras auf ein Objekt auszurichten.

In [7] wird zur Vermessung gekrümmter Metallplatten ein Stereokamerasystem mit einer strukturierten Lichtquelle verwendet. Da die Metallplatten einfarbig sind, kann ein passives System keine korrespondierenden Punkte (siehe Kapitel 3.1.2) in einem Stereobildpaar finden, um mit diesen eine Rekonstruktion durchzuführen. Aus diesem Grund werden die passiven Kameras zunächst benutzt, um die Konturen des Objektes zu finden. Eine Lichtquelle projiziert dann ein regelmäßiges Muster aus Quadraten auf das Objekt, deren höhere Lichtintensität von den Kameras zur Korrespondenzbestimmung verwendet wird.

Aktive Kamerasysteme dieser Art liefern sehr genaue Rekonstruktionen der betrachteten Objekte und werden daher meist in der automatischen Produktion zur Qualitätskontrolle genutzt. Hier befindet sich oft nur ein Objekt auf dem Prüfstand, dass immer an der gleichen Position liegt. Außerdem sind die Lichtverhältnisse der Umgebung konstant und bekannt, so dass sie die Vermessung nicht beeinträchtigen. In manchen Anwendungen müssen die Objekte aus verschiedenen Richtungen oder mit unterschiedlichen Lichtmustern erfasst werden. Daher dürfen sich die betrachteten Objekte in diesen Fällen nicht bewegen. Aus den genannten Gründen lässt sich strukturiertes

Licht im Allgemeinen nicht für mobile Roboter verwenden. Auch der erhöhte Stromverbrauch zum Betreiben der Lichtquelle ist nachteilig für mobile Systeme.

Eine Diskussion verschiedener Methoden, Objekte mit strukturierten Lichtquellen zu vermessen, findet sich in [36] mit einer Darstellung der jeweiligen Anforderungen an das Einsatzgebiet.

2.1.2 Monokulare Systeme

Eine Alternative zu binokularen System sind monokulare Kamerasysteme, bei denen nur eine Kamera eingesetzt wird. Diese bieten einige Vorteile, haben aber auch Einschränkungen. Bei einer einzelnen Kamera gibt es keine Synchronisationsprobleme, wie sie bei den in dieser Arbeit verwendeten Webcams auftreten (siehe Kapitel 5.1). Desweiteren ist die elektronische Verarbeitung schneller, da nur ein Bild analysiert werden muss. Allerdings können räumliche Informationen nur dann aus einem einzelnen Bild gewonnen werden, wenn zusätzliche Bedingungen erfüllt sind. Die folgenden monokulare Systeme zeigen Beispiele für solche Bedingungen.

Trotz der im vorigen Beispiel genannten Nachteile kann man strukturiertes Licht bedingt zur Hinderniserkennung in mobilen Robotersystem verwenden, wie [15] zeigt. Hier sendet ein Roboterfahrzeug eine Lichtebene in Blickrichtung aus, die auf dem Boden vor dem Fahrzeug eine helle Linie erzeugt und von einer Kamera detektiert wird. Befindet sich ein Hindernis zwischen dieser Linie und dem Fahrzeug, so wird die Lichtlinie unterbrochen und erscheint im Bild höher oder niedriger als die Linie des Bodens. Auf diese Weise können Hindernisse bemerkt und umfahren werden. Mit der bekannten Anordnung der Lichtquelle und einer kalibrierten Kamera sind genügend Informationen vorhanden, um aus der Verschiebung der unterbrochenen Lichtlinie im Bild die Entfernungen der Hindernisse zu berechnen. Der Einsatz eines solchen Systems im Freien ist bei direkter Sonneneinstrahlung nicht möglich, da die generierte Linie dann nicht mehr erkennbar ist.

Mit dem System in Kapitel 4.5.1 von [9] kann die Position von Punkten rekonstruiert werden, wenn sie sich in einer bekannten Grundebene befinden. Dies kann zum Beispiel in autonomen Fahrzeugen verwendet werden, die anhand der Fahrbahnmarkierungen navigieren. Da hier nur die Position von ebenen Objekten bestimmt werden kann, eignet sich das System nicht für den in dieser Diplomarbeit geplanten Einsatz im *RoboCup*.

In [52] wird ein weiteres monokulares System vorgestellt, bei dem eine Linse mit großer sphärischer Aberration (siehe Kapitel 6.3 in [16]) vor der Kamera positioniert wird. Dadurch werden Punkte des Raumes zu Ringen in der

Bildebene. Wenn die Parameter der Linse bekannt sind und der Mittelpunkt und Radius eines Ringes bestimmt wird, kann die Position des Punktes rekonstruiert werden. Demonstriert wird das System mit einer Leuchtdiode als Raumpunkt und einem zusätzlichen Frequenzfilter vor der Linse. Dadurch wird nur das Licht der Diode abgebildet und es entsteht ein deutlich erkennbarer Ring. Die Entfernungsbestimmung von nichtleuchtenden Objekten wird in [52] nicht diskutiert. Hier existieren im Allgemeinen keine kontrastreichen Punkte, die auf deutliche Ringe abgebildet werden können. Auch Überlagerungen der abgebildeten Ringe stellen in diesem Fall ein Problem dar.

Ein abschließendes Beispiel für monokulare Positionsberechnungen wird in [58] illustriert. Bei dieser Methode kann die Position fester Objekte bestimmt werden, deren Eckpunkte immer einen konstanten Abstand zueinander besitzen. Sind diese Abstände bekannt und wird ein abgebildetes Objekt erfolgreich erkannt, so können aus den Pixelkoordinaten der Eckpunkte die Raumpositionen berechnet werden. Hierfür sind nichtlineare Gleichungssysteme zu lösen. Voraussetzung für dieses System sind Modelle der verwendeten Objekte und eine leistungsfähige Objekterkennung, welche die Orientierung der Objekte feststellen kann. Schließlich müssen die erkannten Ecken korrekt den Eckpunkten der Objekte zugewiesen werden, damit die Gleichungssysteme aufgestellt werden können. Auch dieses System lässt sich nicht im *RoboCup* einsetzen, da zum Beispiel der Ball keine markante Eckpunkte besitzt, die identifiziert und zum Erzeugen der Gleichungen verwendet werden können.

2.1.3 Pseudo-Stereokamerasysteme

Als Bindeglied zwischen monokularer und binokularer Sicht sind die Pseudo-Stereokamerasysteme zu nennen. Diese Systeme beinhalten nur eine Kamera, mit der aber zwei verschiedene Ansichten einer Szene aufgenommen werden. Zu den Vorteilen solcher Systeme gehört wie bei den monokularen Systemen, dass keine Synchronisationsprobleme auftreten können, da beide Ansichten exakt zu dem selben Zeitpunkt entstehen. Auch die Bildeigenschaften wie Sättigung und Helligkeit sind immer identisch, was bei der Bestimmung von Korrespondenzen hilfreich ist. Gegenüber monokularen Systemen haben Pseudo-Stereosysteme den Vorteil, dass für die betrachteten Objekte keine Einschränkungen existieren, um deren Position berechnen zu können.

In [27] werden verschiedene Ansätze für Pseudo-Stereokameras skizziert. Anschließend wird ein System unter Verwendung eines Prismas konstruiert. Durch das Prisma werden zu einem Punkt im Raum zwei virtuelle Punkte erzeugt, die in einer Ebene parallel zur Basis des Prismas durch den realen Punkt liegen. Dies führt zu einem Bild, in dem die beiden horizontalen

Bildhälften einem Stereobildpaar virtueller Kameras mit der halben horizontalen Auflösung entsprechen. In der Mitte befindet sich lediglich ein schmaler Interferenz-Streifen, der bei der Verarbeitung ausgelassen wird. Bei diesem System besteht ein höherer Aufwand bei der Konstruktion, da die optischen Eigenschaften des Prismas auf die Kamera abgestimmt sein müssen, damit das Licht auf die Bildebene trifft.

Ein anderes Pseudo-Stereokamerasystem wird in [41, 42] vorgestellt. Durch ein System von (halbdurchlässigen) Spiegeln werden zwei Abbildungen einer Szene in der Kamera überlagert. Auch dieses System ist äquivalent zu zwei virtuellen Kameras. Im Gegensatz zu der Variante mit dem Prisma entstehen hier aber nicht zwei getrennte Teilbilder. Dadurch bietet das System zwar die Vorteile einer Pseudo-Stereokamera, aber existierende Standardverfahren zum Bildverstehen können durch die Komplexität des überlagerten Bildes nicht angewendet werden.

Schließlich realisiert BIRIS in Kapitel 4.7.2 von [9] ein weiteres Pseudo-Stereokamerasystem. Die verwendete Kamera verfügt über zwei Blenden und kann dadurch als Lochkamera mit zwei Löchern modelliert werden. Wie bei der Variante mit den Spiegeln werden hier zwei Bilder in der Bildebene überlagert und erfordern eine komplexere Verarbeitung.

2.1.4 Fazit

Die Einschränkungen der monokularen Systeme für den geplanten Einsatz des humanoiden Roboters sind wesentlich größer als ihre Vorteile. Die Verarbeitung eines komplexen überlagerten Bildes, wie es einige der Pseudo-Stereokamerasysteme erzeugen, ist mit der beschränkten Rechenleistung des Humanoiden nicht möglich. Das Pseudo-Kamerasystem mit dem Prisma sowie die Varianten aktiver binokularer Systeme sind zu aufwändig und das resultierende System wird zu schwer und verbraucht im Fall aktiver Systeme viel Energie.
Folglich kann nur ein echtes Stereokamerasystem die Anforderungen der vorliegenden Diplomarbeit erfüllen.

Bei keinem der zitierten Stereokamerasysteme werden explizit Webcams verwendet und daraus resultierende Probleme untersucht. Da auch bei der gesamten Recherche keine Veröffentlichungen mit solcher Hardware gefunden worden sind, wird diese Variante des räumlichen Sehens erstmalig in der vorliegenden Diplomarbeit behandelt.

2.2 Effizienz des Bildverstehens

In Kapitel 3.4 wird das elektronische Bildverstehen ausführlich beschrieben. Auch Beispiele und Quellen für verschiedene Algorithmen werden angeführt. In diesem Abschnitt wird die Leistungsfähigkeit einiger Verfahren anhand ihrer Laufzeit und damit die Verwendbarkeit in mobilen Robotersystemen mit beschränkter Rechenleistung diskutiert. Die Leistung des in dieser Arbeit verwendeten Humanoiden (siehe Anhang A.2) ist geringer als die in den nachfolgend zitierten Quellen benutzte Hardware. Dadurch sind bei der Implementierung der Algorithmen längere Laufzeiten zu erwarten.

Die Umrechnungen der angegebenen Laufzeiten auf die vorliegende Hardware des humanoiden Roboters sind nur Näherungen und dienen der Orientierung. Durch verschiedene Architekturen und Optimierungen können die Zeiten nicht direkt verglichen werden und die Effizienz der Verfahren muss durch Messungen geprüft werden.

2.2.1 Farbsegmentierung

Der Algorithmus zur Farbsegmentierung in [6], der in dieser Diplomarbeit implementiert und in Kapitel 5.3 beschrieben wird, kann Bilder sehr schnell segmentieren. In der Quelle ist angegeben, dass bei einer Auflösung von 160×120 auf einem Pentium mit 200 MHz und einer Auslastung von nur 12% 30 Bilder pro Sekunde verarbeitet werden können. Rechnet man das linear auf 133 MHz und 100% Last um, so erhält man eine Segmentierungszeit von etwa 0,006 Sekunden pro Bild. Zum Vergleich ist in Kapitel 6.2 die Laufzeit der neuen, für den Humanoiden optimierten, Implementierung gegeben.

In [1] wird ein Segmentierungsverfahren auf Basis von Fuzzy-Logik vorgeschlagen. Dieses weist jedem Pixel durch Minimierung einer Abstandsfunktion eine Wahrscheinlichkeit zu, mit der es in verschiedenen Bildbereichen liegt. Dafür wird die Anzahl der Bereiche mit jeweils einem Farbwert vorgegeben. Nach eigenen Angaben ist dieses Verfahren „hocheffizient“. Für einige Bilder sind auch die Verarbeitungszeiten gegeben. Bei einer Auflösung von 512×512 und einem Pentium III mit 1600 MHz werden jedoch durchschnittlich 8,75 Sekunden benötigt. Wird diese Zeit linear auf 160×120 Pixel und 133 MHz umgerechnet, so benötigt das Verfahren etwa 7,71 Sekunden.

2.2.2 Berechnung der Disparitäten

Auch für die Berechnung der Disparitäten korrespondierender Pixel in einem Stereobildpaar (siehe Kapitel 3.1.2) werden in einigen Quellen Laufzeiten angegeben. Dies ist der Fall für den flächenbasierten Algorithmus in [37], der

auch in dieser Arbeit implementiert und in Kapitel 5.4.1 illustriert wird. Die Autoren testen den für ihre Hardware optimierten Algorithmus auf einem Dual-Pentium III mit jeweils 800 MHz und Bildern verschiedener Größe. Für die Auflösung von 160×120 und einem Disparitätssuchraum von 32 werden 0,05 Sekunden benötigt. Durch den verhältnismäßig kleinen Suchraum ist es nicht möglich, sehr nahe Objekte erkennen zu können. Eine Vergrößerung des Suchraumes führt zu einer längeren Laufzeit. Umgerechnet auf die vorliegende Hardware ergibt sich eine Laufzeit von etwa 0,6 Sekunden. In Kapitel 6.3.1 wird zum Vergleich die Laufzeit der neuen Implementierung auf dem verwendeten Entwicklungssystem angegeben.

Zwei weitere Beispiele für die Disparitätsberechnung in Graustufenbildern sind in [10] aufgeführt. Eine merkmalsbasierte Version klassifiziert einzelne Pixel als Merkmal, wenn sich in der 4-Nachbarschaft die Helligkeit stark ändert. Für diese Pixel werden anschließend die Disparitäten bestimmt. Mit einem Pentium II und 400 MHz benötigt dieses Verfahren bei einer Bildgröße von 384×256 nur 0,023 Sekunden. Dies entspricht bei 160×120 Pixeln schätzungsweise 0,014 Sekunden auf dem humanoiden Roboter. Eine Voraussetzung für diesen Algorithmus ist jedoch ein gleichgerichtetes Stereobildpaar (siehe Kapitel 3.4.2). Die flächenbasierte Variante verwendet eine Gaußpyramide (siehe auch Kapitel 5.3.2 in [23]) des Stereobildpaares. Diese beinhaltet verschiedene Auflösungen des Bildes, die zur Beschleunigung des Verfahrens benutzt werden. In dem Originalbild werden Pixel hohen Kontrasts gesucht, deren Anzahl in jeder Pyramidenebene durch die geringere Auflösung reduziert wird. Daher werden auf der Ebene mit der geringsten Auflösung nur wenige Disparitäten berechnet, die bei jeder Iteration in die nächst größere Auflösung verfeinert werden. Mit den gleichen Voraussetzungen wie bei der ersten Version ist hier eine Laufzeit von 0,09 Sekunden möglich, was bei einer Auflösung von 160×120 etwa 0,053 Sekunden auf dem Humanoiden entspricht.

In vielen anderen Veröffentlichungen werden effiziente oder echtzeitfähige Algorithmen vorgestellt. Wenn Messungen der Laufzeit und eine Beschreibung der Testumgebung vorliegen, stellt man jedoch fest, dass häufig aktuelle Rechner mit weit mehr als 500 MHz oder sogar mehrere vernetzte Computer verwendet werden. Dadurch steht wesentlich mehr Rechenleistung zur Verfügung als in mobilen Robotersystemen.

2.2.3 Implementierung in Hardware

Eine Möglichkeit zur Beschleunigung des Bildverstehens besteht darin, die Algorithmen nicht in Software sondern in Hardware auszuführen. In [66] wird ein flächenbasiertes Verfahren zur Berechnung der Disparitäten vorgeschla-

gen, dass eine moderne Grafikkarte nutzt. Das Verfahren ist dem in Kapitel 2.2.2 skizzierten Algorithmus von [10] sehr ähnlich und verwendet auch eine Gaußpyramide des Stereobildpaares in Graustufen, die bei entsprechender Grafikkarte automatisch erzeugt werden kann. Mit einigen Optimierungen für die benutzte Karte Radeon 9700 Pro werden für Bilder der Größe 256×256 und einem Disparitätssuchraum von 150 Laufzeiten bis zu 0,02 Sekunden erzielt.

Nicht nur Grafikkarten können die Effizienz des Bildverstehens steigern. Auch programmierbare Bausteine, wie zum Beispiel Field Programmable Gate Arrays (FPGA), können die Leistung erhöhen. In diesen Bausteinen werden Algorithmen durch entsprechende Verschaltung der einzelnen Transistoren implementiert. Dadurch kann ein Algorithmus schneller ausgeführt werden als auf einem Allzweck-Prozessor. In [3] werden die Möglichkeiten von FPGAs ausführlich diskutiert. Ein wichtiges Ergebnis ist hierbei, dass die Leistungssteigerung stark vom Datentransfer mit dem Baustein abhängt. Insbesondere die Speicherarchitektur und die Bus-Geschwindigkeit können den FPGA ausbremsen.

Im Gegensatz zu modernen Grafikkarten benötigen FPGAs nicht so viel Strom, was für den mobilen Einsatz vorteilhaft ist. Darüber hinaus können Grafikkarten im Allgemeinen nur auf handelsüblichen Motherboards eingesetzt werden. Ein FPGA kann auch mittels einer eigenen Schaltung platzsparend in einen mobilen autonomen Roboter integriert und angesteuert werden. Beiden Varianten ist gemeinsam, dass sie den Prozessor entlasten, dem so mehr Zeit für andere Aufgaben, wie Bahnplanung und Selbstlokalisierung, zur Verfügung steht.

In [22] wird schließlich ein kompaktes trinokulares Kamerasystem vorgestellt. Es verfügt über drei Schwarz-Weiß-Kameras, einen FPGA und einen IEEE1394-Anschluss, um Bilddaten weiterleiten zu können. Auf dem programmierbaren Baustein läuft ein umfangreiches Programm zur Verarbeitung der Bildtripel. Zunächst werden die Bilder gleichgerichtet, gefiltert und komprimiert. Anschließend wird ein flächenbasierter Algorithmus zur Berechnung der Disparitäten mit einem Suchraum von 64 angewendet. Für Bilder der Größe 320×240 werden nur etwa 0,008 Sekunden für einen kompletten Zyklus benötigt. Das System ist damit sehr schnell und außerdem klein, wodurch es hervorragend für den Einsatz in mobilen autonomen Robotern geeignet ist.

2.2.4 Fazit

Wie man an den genannten Beispielen sieht, nimmt das Bildverstehen einen großen Teil der Rechenleistung eines mobilen Roboters in Anspruch, der ne-

ben dem Bildverstehen noch viele andere Aufgaben lösen muss. Die aktuellen Verfahren sind kaum geeignet, um in Software implementiert und auf den relativ langsamen Systemen der Roboter ausgeführt zu werden. Durch eine starke Optimierung der Algorithmen kann zwar die Leistung auf einzelnen Systemen gesteigert werden, aber dadurch lassen sich die Komponenten nicht auf anderen Robotern effizient einsetzen. Eine effektive Verbesserung ist die Implementierung der Verfahren in programmierbaren Bausteinen.

Für die meisten veröffentlichten Verfahren zum Bildverstehen gibt es keine freien Implementierungen. Wenn dies der Fall ist, sind sie jedoch für spezielle Hardware optimiert oder nicht schnell genug für den Einsatz auf dem humanoiden Roboter. Aus diesem Grund werden in dieser Arbeit die verwendeten Algorithmen neu implementiert und können so an die vorliegende Hardware angepasst werden.

Kapitel 3

Theorie

3.1 Modellbeschreibung

Um die Umgebung eines Roboters zu erfassen und elektronisch zu verarbeiten, sind mathematische Modelle notwendig, welche in den folgenden Abschnitten beschrieben werden. Kameramodelle werden zum Beispiel auch in den Kapiteln 2.4 in [63], 6.2 in [14] und 7.4 in [11] diskutiert. Stereoskopie ist Bestandteil der Kapitel 7.1 in [63] und 2.2 in [40].

Ein Ziel dieser Arbeit ist die Erfassung der Umwelt durch elektronische Verarbeitung der Bilder von Digitalkameras. Es werden dreidimensionale reale Objekte in der Nähe des Roboters in zweidimensionale, digitale Bilder umgewandelt, die dann weiter verarbeitet werden. Um aus diesen Bildern Informationen über die abgebildete Umgebung zu erhalten, ist ein mathematisches Modell der Kamera aus Abbildung 1.2 nötig. Hierfür wird die perspektivische Projektion verwendet.

Eine Kamera wird durch intrinsische und extrinsische Parameter beschrieben. Mit ersteren wird die Transformation dreidimensionaler Objekte im Kamerakoordinatensystem S_k in die zweidimensionale Bildebene S_b modelliert. Letztere bestehen aus einem Translationsvektor ${}^k\mathbf{r}_0$ und den freien Parametern einer Rotationsmatrix ${}^kR_0$. Diese Werte definieren die Transformation zwischen einem beliebigen Referenzkoordinatensystem S_0 und dem Kamerakoordinatensystem S_k und werden in Kapitel 3.1.2 behandelt.

3.1.1 Das Kameramodell

In diesem Abschnitt werden die intrinsischen Parameter untersucht, welche die Transformation zwischen Kamerakoordinaten S_k mit Ursprung im Fokus der Kamera und Pixelkoordinaten S_b mit Ursprung in der oberen, linken

Bildecke definieren und in Abbildung 3.1 verdeutlicht werden. Hierzu gehören die Brennweite f, der Schnittpunkt der optischen Achse mit der Bildebene (Hauptpunkt) in Pixelkoordinaten ${}^b\mathbf{o} = (o_x; o_y)^T \in \mathbb{R}^2$, die Größe der Pixel $(s_x; s_y) \in \mathbb{R}^2$ und die Verkippung der Bildkoordinatenachsen φ.

Da die Größe der Pixel oft nicht direkt bestimmbar ist, können stattdessen die Auflösung $(a_x; a_y)^T \in \mathbb{N}^2$ und der horizontale bzw. vertikale Öffnungswinkel $(\theta_x; \theta_y)^T \in \mathbb{R}^2$ der Kamera verwendet werden. Die beschriebenen Parameter sind erweiterbar, um Abbildungsfehler wie zum Beispiel sphärische Aberration oder Koma (Kapitel 6.3 in [16]) zu berücksichtigen, welche in dieser Arbeit jedoch vernachlässigt werden.

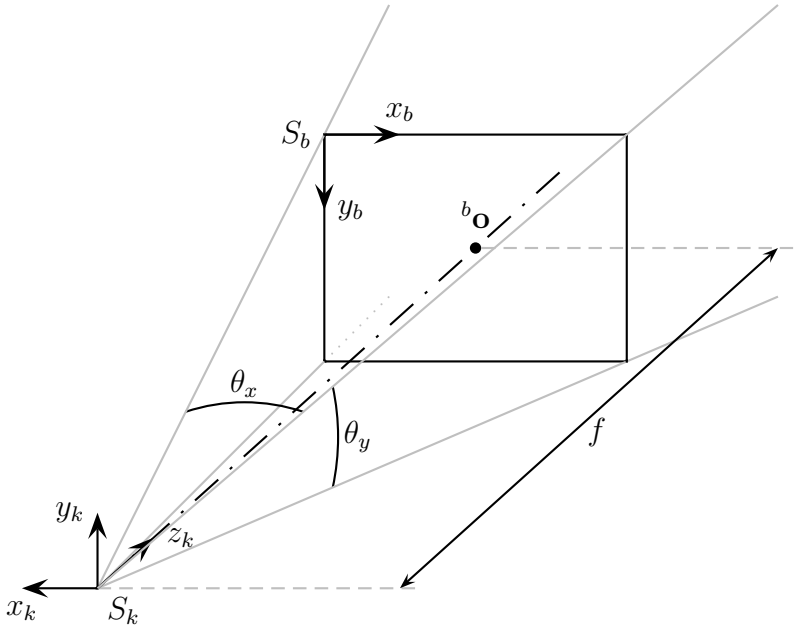


Abbildung 3.1: Kameramodell mit den bestimmbaren intrinsischen Parametern.

Die verschiedenen Orientierungen der Kamera- und Bildkoordinatensysteme ergeben sich aus den üblichen Konventionen. In der Bildverarbeitung liegt der Ursprung oben links mit den x- bzw. y-Achsen nach rechts bzw. unten. In dem dreidimensionalen Kamerasystem erscheint es natürlicher, dass positive y-Werte eine Höhe und positive z-Werte einen Abstand in Blickrichtung darstellen.

Um das Bild ${}^b\mathbf{p} = ({}^bp_x; {}^bp_y)^T$ eines Punktes ${}^k\mathbf{p}$ wie in Abbildung 3.2 zu bestimmen, sind einige einfache Rechnungen nötig. Sei ${}^k\mathbf{p} = ({}^kp_x; {}^kp_y; {}^kp_z)^T$ ein Punkt der physikalischen Welt in Koordinaten mit einer Längeneinheit. Weiterhin sei angenommen, dass der Ursprung von S_k und damit der Brenn-

punkt im Gehäuse der Kamera liegt, also die z-Koordinate aller abbildbaren Punkte größer 0 ist. Bei einem perspektivischen Kameramodell, bei dem der Ursprung von S_b auf der optischen Achse liegt, wird ${}^k\mathbf{p}$ mit

$$ {}^b\tilde{p}_x = -f\frac{{}^k p_x}{{}^k p_z}, \qquad {}^b\tilde{p}_y = -f\frac{{}^k p_y}{{}^k p_z} \tag{3.1} $$

abgebildet.

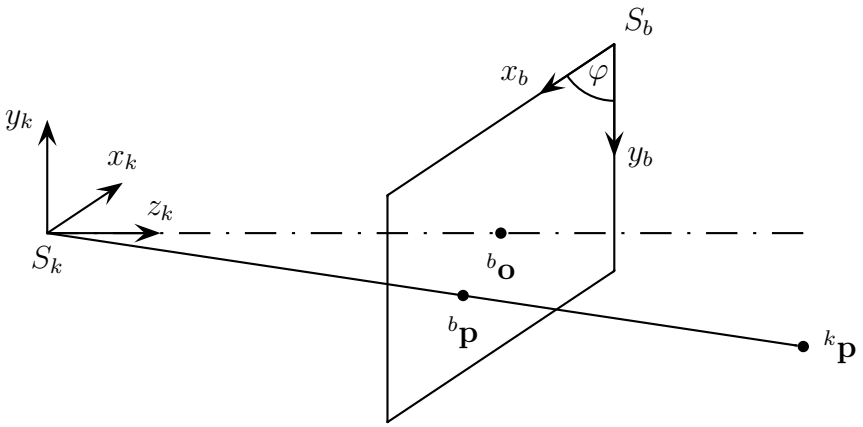


Abbildung 3.2: Projektion eines Punktes im Raum auf die Bildebene.

In diesem Fall liegt aber der Ursprung von S_b nicht auf der optischen Achse und die Koordinaten ${}^b p_x$ und ${}^b p_y$ sind Pixel und keine metrischen Längen. Für diese Umrechnungen werden der Hauptpunkt ${}^b\mathbf{o}$ und die Pixelgröße $(s_x; s_y)$ verwendet.

$$ {}^b p_x = o_x + \frac{{}^b\tilde{p}_x}{s_x} = o_x - \frac{f\,{}^k p_x}{{}^k p_z\, s_x}, \qquad {}^b p_y = o_y + \frac{{}^b\tilde{p}_y}{s_y} = o_y - \frac{f\,{}^k p_y}{{}^k p_z\, s_y} \tag{3.2} $$

Bei CCD-Kameras kommt es außerdem vor, dass die Bildkoordinatenachsen einen Winkel $\varphi \neq \frac{\pi}{2}$ einschließen. Dieser zusätzliche Parameter wird durch eine Scherung

$$ {}^b p_x = o_x - \frac{f\,{}^k p_x}{{}^k p_z\, s_x} + \frac{f\,{}^k p_y}{{}^k p_z\, s_x \tan\varphi}, \qquad {}^b p_y = o_y - \frac{f\,{}^k p_y}{{}^k p_z\, s_y} \tag{3.3} $$

eingeführt.

Gleichung (3.3) kann auch als Matrixprodukt geschrieben werden, da dieses in symbolischen Gleichungen kürzer und damit lesbarer ist.

$$ {}^b\hat{\mathbf{p}} = s \begin{pmatrix} {}^b p_x \\ {}^b p_y \\ 1 \end{pmatrix} = K\,{}^k\mathbf{p} = \begin{pmatrix} -\frac{f}{s_x} & \frac{f}{s_x \tan\varphi} & o_x \\ 0 & -\frac{f}{s_y} & o_y \\ 0 & 0 & 1 \end{pmatrix} {}^k\mathbf{p} \tag{3.4} $$

Hierbei ist $K \in \mathbb{R}^{3\times3}$ die Transformationsmatrix der intrinsischen Kameraparameter und $s \in \mathbb{R}$ ein beliebiger Skalar. Gleichung (3.4) bildet inhomogene Koordinaten im dreidimensionalen Kamerasystem auf homogene Koordinaten (z.B. Kapitel 2.2.6 in [11]) in der Bildebene ab. Weil die perspektivische Projektion keine lineare Abbildung ist, müssen die ersten beiden Komponenten von ${}^b\hat{\mathbf{p}}$ durch die dritte Komponente dividiert werden, um ${}^b\mathbf{p}$ zu erhalten. Die Koordinaten der Bildpunkte werden dabei auf ganze Zahlen gerundet, da Bilder in einem Computer nur diskret und nicht kontinuierlich dargestellt werden können.

Wenn die Pixelgröße nicht bekannt ist, so kann man diese als Verhältnis

$$s_x = \frac{b}{a_x}, \qquad s_y = \frac{h}{a_y} \tag{3.5}$$

der Breite b bzw. Höhe h der Bildebene zur horizontalen bzw. vertikalen Auflösung bestimmen. Liegt zusätzlich der Hauptpunkt in der Bildmitte

$${}^b\mathbf{o} = \begin{pmatrix} o_x \\ o_y \end{pmatrix} = \begin{pmatrix} a_x/2 \\ a_y/2 \end{pmatrix}, \tag{3.6}$$

so ergibt sich aus geometrischen Überlegungen

$$s_x = \frac{f \tan\frac{\theta_x}{2}}{o_x}, \qquad s_y = \frac{f \tan\frac{\theta_y}{2}}{o_y}. \tag{3.7}$$

Setzt man (3.7) in (3.4) ein, erhält man die Kameramatrix in Abhängigkeit der Öffnungswinkel und der halben Auflösung.

$$K = \begin{pmatrix} -\frac{o_x}{\tan\frac{\theta_x}{2}} & \frac{o_x}{\tan\frac{\theta_x}{2}\tan\varphi} & o_x \\ 0 & -\frac{o_y}{\tan\frac{\theta_y}{2}} & o_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

Um von den gewählten intrinsischen Parametern zu abstrahieren, werden folgende Variablen eingeführt.

$$K = \begin{pmatrix} d_x & c & o_x \\ 0 & d_y & o_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.9}$$

3.1.2 Stereoskopie

Wie man in Kapitel 3.1.1 sieht, werden Geraden des Raumes durch den Brennpunkt der Kamera auf Punkte in der Ebene abgebildet. Daher ist es

umgekehrt nicht möglich, die Position eines Objektes aus den Pixeln eines Bildes zu bestimmen. Aus diesem Grund sind mindestens, im Fall des binokularen Sehens genau, zwei Kameras nötig, um räumliche Informationen aus digitalen Bildern zu ermitteln. Diese Bedingung trifft zu, wenn keine weiteren Informationen über die betrachteten Objekte, wie zum Beispiel deren Größe, vorliegen.

Wird ein Punkt im Raum in die Bildebenen abgebildet, so erhält man ein Paar von Pixelkoordinaten. Die betroffenen Pixel werden korrespondierende Pixel genannt. Die zur Abbildung verwendeten Kameras müssen nicht zwangsläufig identisch sein. Es ist jedoch vorteilhaft, wenn die Gewichtsverteilung des Stereokamerasystems symmetrisch ist und die Bilder ähnliche Farbeigenschaften sowie identische Auflösungen aufweisen.

Abbildung 3.3 zeigt eine mögliche Anordnung von zwei Kameras, wie sie in dieser Arbeit verwendet wird. Sie orientiert sich an der Anordnung der Augen vieler Lebewesen (z.B. des Menschen), welche sich im Laufe der Evolution besonders bei Raubtieren als nützlich für die Entfernungsbestimmung erwiesen hat.

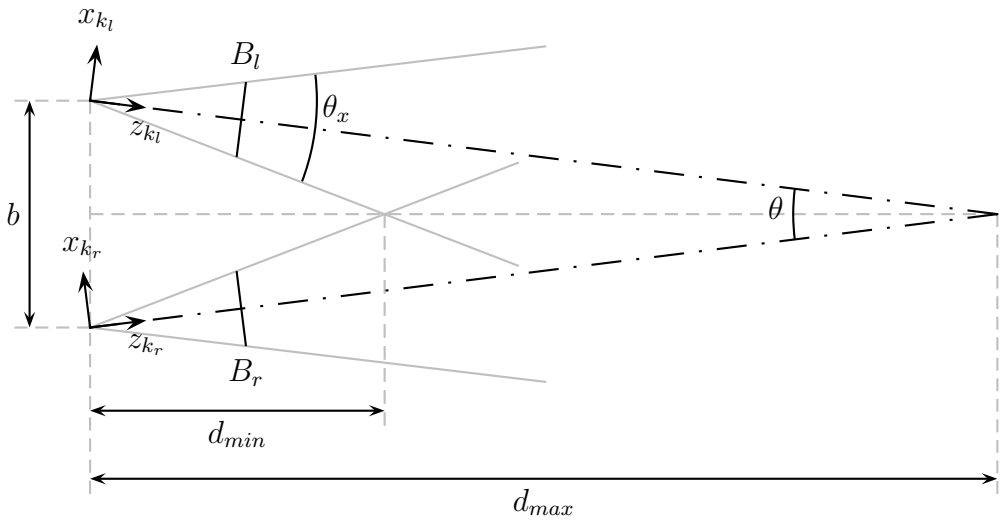


Abbildung 3.3: Sicht von oben auf das Stereoskopie-Modell mit den Bildebenen B_l und B_r.

Die Foki der Kameras haben einen horizontalen Abstand b voneinander und sind symmetrisch angeordnet. Die optischen Achsen der Kameras schließen dabei einen Konvergenzwinkel θ ein. Damit definieren die Parameter b und θ die extrinsischen Parameter der beiden Kameras zueinander und legen die minimale Entfernung d_{min} sowie die praktisch maximale Entfernung d_{max} der sichtbaren Objekte fest.

Mit praktisch maximaler Entfernung ist gemeint, dass auch ein Objekt in größerer Entfernung in den Bildern sichtbar ist, aber die Beziehung der korrespondierenden Pixel sich ändert. Dieser Effekt, welcher in Kapitel 5.4.1 noch einmal näher besprochen wird, erhöht die Laufzeit einiger Algorithmen zur Bestimmung korrespondierender Pixel und kann durch Einschränkung des Suchraumes vermieden werden.

Der mathematische Zusammenhang der Größen aus Abbildung 3.3 wird durch folgende Gleichungen beschrieben:

$$d_{min} = \frac{b}{2\tan\frac{\theta+\theta_x}{2}}, \qquad d_{max} = \frac{b}{2\tan\frac{\theta}{2}}. \tag{3.10}$$

Diese Gleichungen sind ein wichtiger Bestandteil bei der Festlegung der Parameter b und θ. Aber auch zwei weitere Auswirkungen dieser Wahl müssen bedacht werden.

Zum einen ist die Auflösung der Entfernungen von Objekten besser, je größer b ist. Wenn die Kameras sehr eng zusammen stehen, befinden sich Objekte ungeachtet ihrer Entfernung immer an ähnlichen Bildkoordinaten, wodurch eine Differenzierung der Position nur schwer möglich ist. Ist dagegen der Abstand der Kameras groß, ändern sich die Bildkoordinaten stark, wenn sich ein Objekt in verschiedenen Entfernungen befindet.

Zum anderen dürfen die dynamischen Eigenschaften des Kamerakopfes nicht vernachlässigt werden. Die Konstruktion muss stabiler und damit schwerer sein, wenn der Abstand b erhöht wird. Außerdem müssen die Motoren, welche den Kopf bewegen, mehr Leistung bringen, da die Trägheitsmomente größer sind.

Da nicht beide Auswirkungen gleichermaßen gut behandelt werden können, ist die Bestimmung der extrinsischen Parameter nicht theoretisch, sondern nur im konkreten Anwendungsfall lösbar.

Sind die Parameter des Kamerakopfes festgelegt, so bestimmt man die homogene Transformation des Referenzkoordinatensystems S_0 in das linke bzw. rechte Kamerasystem S_{k_l} bzw. S_{k_r}. Sei ohne Beschränkung der Allgemeinheit angenommen, dass das Referenzkoordinatensystem S_0 und das linke Kamerakoordinatensystem identisch sind, dann sind

$$\begin{aligned} {}^{k_l}T_0 &= ({}^{k_l}R_0; {}^{k_l}\mathbf{r}_0) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \\ {}^{k_r}T_0 &= ({}^{k_r}R_0; {}^{k_r}\mathbf{r}_0) = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & b\cos\theta \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & b\sin\theta \end{pmatrix} \end{aligned} \tag{3.11}$$

die Transformationsmatrizen der extrinsischen Kameraparameter. Kombiniert man diese Matrizen mit Gleichung (3.4), so erhält man die Transformation vom Referenzkoordinatensystem in die linke bzw. rechte Bildebene in Abhängigkeit der intrinsischen und extrinsischen Kameraparameter.

$$
\begin{aligned}
{}^{b_l}\hat{\mathbf{p}} &= s_l \begin{pmatrix} {}^{b_l}p_x \\ {}^{b_l}p_y \\ 1 \end{pmatrix} = K_l \, {}^{k_l}T_0 \, {}^0\hat{\mathbf{p}} = P_l \, {}^0\hat{\mathbf{p}} \\
{}^{b_r}\hat{\mathbf{p}} &= s_r \begin{pmatrix} {}^{b_r}p_x \\ {}^{b_r}p_y \\ 1 \end{pmatrix} = K_r \, {}^{k_r}T_0 \, {}^0\hat{\mathbf{p}} = P_r \, {}^0\hat{\mathbf{p}}
\end{aligned} \tag{3.12}
$$

Die so definierten Matrizen $P_l, P_r \in \mathbb{R}^{3\times 4}$ sind die Projektionsmatrizen, welche homogene Punkte in dem Referenzkoordinatensystem in homogene Punkte der Bildebenen transformieren.

Da im Folgenden nur noch Koordinaten in den Bildebenen und im Referenzkoordinatensystem betrachtet werden, wird eine vereinfachte Schreibweise der verwendeten Matrizen eingeführt, indem die Indizes der Bildkoordinaten und die Transformationsmatrizen umbenannt werden.

$$
{}^l\hat{\mathbf{p}} = K_l \, (E_3; \mathbf{0}) \, {}^0\hat{\mathbf{p}} = P_l \, {}^0\hat{\mathbf{p}}, \qquad {}^r\hat{\mathbf{p}} = K_r \, (R; \mathbf{t}) \, {}^0\hat{\mathbf{p}} = P_r \, {}^0\hat{\mathbf{p}} \tag{3.13}
$$

Bildet man mit Gleichung (3.13) einen Punkt ${}^0\hat{\mathbf{p}} = ({}^0p_x; {}^0p_y; {}^0p_z, 1)^T$ auf die inhomogenen Punkte ${}^l\mathbf{p}$ und ${}^r\mathbf{p}$ ab, so kann man deren Differenz $\mathbf{d}_{rl} \in \mathbb{R}^2$ berechnen. Hierbei handelt es sich um die so genannte Disparität vom rechten ins linke Bild. Gleichung 3.14 zeigt die Disparität, falls die Kameras identisch sind und somit $K_l = K_r$ gilt.

$$
\begin{aligned}
\mathbf{d}_{rl} &= {}^l\mathbf{p} - {}^r\mathbf{p} \\
&= \begin{pmatrix} \frac{{}^0p_z({}^0p_y\, c - b\, d_x)\cos\theta + (({}^0p_x(b+{}^0p_x) + {}^0p_z^2)d_x + (b+{}^0p_x){}^0p_y\, c)\sin\theta - {}^0p_y\, {}^0p_z\, c}{{}^0p_z({}^0p_z\, \cos\theta + (b+{}^0p_x)\sin\theta)} \\ {}^0p_y\, d_y \left(\frac{1}{{}^0p_z} - \frac{1}{{}^0p_z\, \cos\theta + (b+{}^0p_x)\sin\theta} \right) \end{pmatrix}
\end{aligned} \tag{3.14}
$$

Überlagert man das rechte und linke Bild, so kann man die Disparität wie in Abbildung 3.4 veranschaulichen.

Im Fall $\theta = 0$ vereinfacht sich Gleichung (3.14) zu

$$
\mathbf{d}_{rl} = \begin{pmatrix} -\frac{b\, d_x}{{}^0p_z} \\ 0 \end{pmatrix}. \tag{3.15}
$$

Der Zusammenhang $d_{rl_x} \propto \frac{1}{{}^0p_z}$ dieser Gleichung verdeutlicht, dass Entfernungen im Nahbereich der Kameras besser aufgelöst werden. Objekte in der

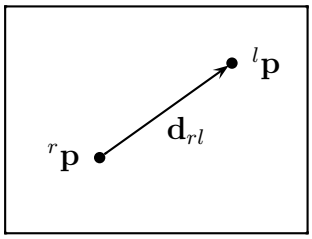


Abbildung 3.4: Disparität $\mathbf{d}_{rl}$ bei einem überlagerten Stereobildpaar.

Nähe der Kameras befinden sich an sehr unterschiedlichen Stellen in den Bildern. Damit ist eine große Disparität gegeben. Ändert sich die Disparität nur um wenige Pixel, so ist auch die Entfernungsänderung des Objektes gering. Ist das Objekt jedoch weit entfernt, so ist die Disparität gering. Ändert sich diese nun um wenige Pixel, so ändert sich die Entfernung des Objektes stark.

3.2 Epipolargeometrie

Die Epipolargeometrie (z.B. Kapitel 9 in [14] und Kapitel 7.3 in [63]) ist ein wichtiger Bestandteil der Stereoskopie, da sie Zusammenhänge in einem Stereobildpaar charakterisiert. Abbildung 3.5 zeigt zwei Fälle des in Kapitel 3.1.2 definierten Stereosystems.

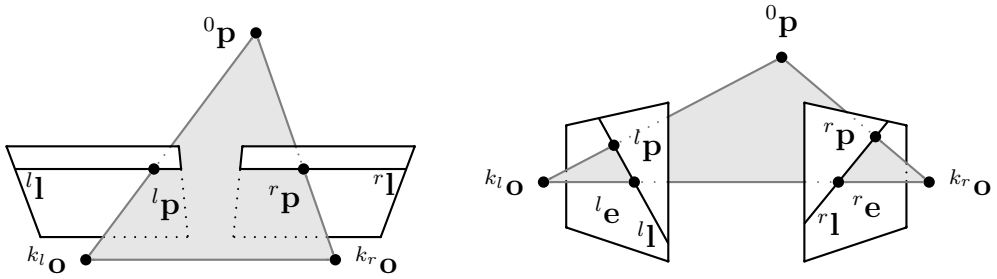


Abbildung 3.5: Beispiele für Epipolargeometrie: (a) $\theta = 0$, (b) $\theta \neq 0$.

Die grau eingezeichnete Epipolarebene ist durch einen Punkt $^{0}\mathbf{p}$ und die Foki der Kameras $^{k_l}\mathbf{o}$ und $^{k_r}\mathbf{o}$ bestimmt und schneidet die Bildebenen in den Epipolarlinien $^{l}\mathbf{l}$ und $^{r}\mathbf{l}$. Die beiden Bildpunkte $^{l}\mathbf{p}$ und $^{r}\mathbf{p}$, auf welche $^{0}\mathbf{p}$

abgebildet wird, liegen genau auf diesen Linien. Umgekehrt schneiden sich die Strahlen durch die Foki und die Bildpunkte in ${}^0\mathbf{p}$. Alle Epipolarlinien gehen dabei durch die Epipole ${}^l\mathbf{e}$ und ${}^r\mathbf{e}$. Im Fall paralleler Kameras mit $\theta = 0$ haben die Epipole die homogenen Koordinaten ${}^l\hat{\mathbf{e}} = (-1; 0; 0)^T$ und ${}^r\hat{\mathbf{e}} = (1; 0; 0)^T$.

Die Epipolargeometrie ist vollständig durch die intrinsischen und extrinsischen Kameraparameter aus Gleichung (3.12) definiert und wird durch die so genannte Fundamentalmatrix

$$F = \left(P_r \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \right)_\times P_r \, P_l^+ = K_r^{-T} \, \mathbf{t}_\times \, R \, K_l^{-1} \in \mathbb{R}^{3\times 3} \tag{3.16}$$

beschrieben. Dabei ist $P_l^+ = (K_l^{-T}; \mathbf{0})^T$ die Pseudoinverse von P_l und $\mathbf{t}_\times$ eine antisymmetrische Matrix, für die

$$\mathbf{t}_\times \, \mathbf{b} = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \mathbf{b} = \mathbf{t} \times \mathbf{b} \tag{3.17}$$

für beliebige Vektoren $\mathbf{b} \in \mathbb{R}^3$ gilt.

Die Fundamentalmatrix ist homogen, also eindeutig bis auf einen skalaren Faktor, und besitzt den Rang 2, da die Matrix $\mathbf{t}_\times$ den Rang 2 hat. Damit verfügt F über sieben Freiheitsgrade. Außerdem weist diese Matrix weitere wichtige Eigenschaften auf. Korrespondierende Pixel im Stereobildpaar erfüllen

$${}^r\hat{\mathbf{p}}^T \, F \, {}^l\hat{\mathbf{p}} = 0. \tag{3.18}$$

Pixel im linken bzw. rechten Bild definieren je eine Epipolarlinie im rechten bzw. linken Bild gemäß

$$F \, {}^l\hat{\mathbf{p}} = {}^r\mathbf{l}, \qquad\qquad F^T \, {}^r\hat{\mathbf{p}} = {}^l\mathbf{l}. \tag{3.19}$$

Die Vektoren ${}^r\mathbf{l}$ und ${}^l\mathbf{l}$ sind Repräsentationen von Geraden in der Ebene. Es definiert der Vektor $\mathbf{l} = (a; b; c)^T \in \mathbb{R}^3$ die Gerade $a\,x + b\,y + c = 0$, welche eindeutig ist bis auf einen skalaren Faktor ungleich 0. Schließlich erfüllen die beiden Epipole die Gleichungen

$$F \, {}^l\hat{\mathbf{e}} = \mathbf{0}, \qquad\qquad F^T \, {}^r\hat{\mathbf{e}} = \mathbf{0}. \tag{3.20}$$

Gleichung (3.20) hat eine nichttriviale Lösung, da die Fundamentalmatrix den Rang 2 besitzt. Andernfalls würden sich die Epipolarlinien nicht in einem Epipol schneiden. Ist die Singulärwertzerlegung von F bekannt, sind die Epipole ${}^l\hat{\mathbf{e}}$ bzw. ${}^r\hat{\mathbf{e}}$ die Rechts- bzw. Linkssingulärvektoren, die zum Singulärwert 0 gehören.

Für den Fall $\theta = 0$ ergibt sich aus (3.16) nach geeigneter Skalierung mit $\frac{d_{y\,l}\, d_{y\,r}}{b}$ die Fundamentalmatrix

$$F = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -d_{y\,l} \\ 0 & d_{y\,r} & d_{y\,l}\, o_{y\,r} - d_{y\,r}\, o_{y\,l} \end{pmatrix}. \tag{3.21}$$

Bei identischen Kameras $K_l = K_r$ und einer weiteren Skalierung mit $\frac{1}{d_{y\,l}} = \frac{1}{d_{y\,r}}$ erhält man die Matrix

$$F = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{3.22}$$

3.3 Rekonstruktion

Der folgende Abschnitt beschäftigt sich mit der Rekonstruktion von Objekten und ihrer Position im Referenzkoordinatensystem S_0. Dadurch wird es möglich, aus den Stereobildpaaren, welche das Stereokamerasystem aufzeichnet, räumliche Informationen zu ermitteln. Das vorgestellte Verfahren für diese Berechnung wird auch in Kapitel 12.2 von [14] ausführlich beschrieben.

Seien die korrespondierenden Pixel ${}^{l}\mathbf{p}$ und ${}^{r}\mathbf{p}$ eines Stereobildpaares gegeben, welche den Punkt ${}^{0}\mathbf{p}$ im Referenzkoordinatensystem beschreiben. Damit der Punkt ${}^{0}\mathbf{p}$ rekonstruiert werden kann, müssen die intrinsischen und extrinsischen Kameraparameter bekannt sein. Die Bestimmung der Parameter wird Kalibrierung genannt. Je nachdem, wie viele Informationen über das Stereokamerasystem nach der Kalibrierung vorliegen, sind zwei Varianten der Berechnung zu unterscheiden. Diese sind sich zwar sehr ähnlich, unterscheiden sich aber stark in der Bedeutung der Ergebnisse und damit der Rekonstruktion.

3.3.1 Rekonstruktion mit starker Kalibrierung

Während einer starken Kalibrierung werden alle Kameraparameter bestimmt. Diese Informationen erlauben eine metrische Rekonstruktion, bei welcher ${}^{0}\mathbf{p}$ im euklidischen Referenzkoordinatensystem S_0 liegt und die Koordinaten über aussagekräftige Längeneinheiten verfügen. Seien also im Folgenden die Projektionsmatrizen P_l und P_r bekannt.

Die Rekonstruktion wird durch eine lineare Triangulation berechnet. Aus Gleichung (3.13) wird am Beispiel des linken Bildes der skalare Faktor durch

ein Kreuzprodukt herausgerechnet. Dadurch erhält man pro Bild drei Gleichungen, von denen zwei linear unabhängig sind.

$$
\begin{aligned}
{}^{l}\hat{\mathbf{p}} \times (P_l\, {}^{0}\hat{\mathbf{p}}) &= \begin{pmatrix} {}^{l}p_x \\ {}^{l}p_y \\ 1 \end{pmatrix} \times \left(\begin{pmatrix} \bar{\mathbf{p}}_{l1} \\ \bar{\mathbf{p}}_{l2} \\ \bar{\mathbf{p}}_{l3} \end{pmatrix} {}^{0}\hat{\mathbf{p}} \right) \\
&= \begin{pmatrix} {}^{l}p_y(\bar{\mathbf{p}}_{l3}\, {}^{0}\hat{\mathbf{p}}) - (\bar{\mathbf{p}}_{l2}\, {}^{0}\hat{\mathbf{p}}) \\ (\bar{\mathbf{p}}_{l1}\, {}^{0}\hat{\mathbf{p}}) - {}^{l}p_x(\bar{\mathbf{p}}_{l3}\, {}^{0}\hat{\mathbf{p}}) \\ {}^{l}p_x(\bar{\mathbf{p}}_{l2}\, {}^{0}\hat{\mathbf{p}}) - {}^{l}p_y(\bar{\mathbf{p}}_{l1}\, {}^{0}\hat{\mathbf{p}}) \end{pmatrix} = \mathbf{0}
\end{aligned} \tag{3.23}
$$

Die Vektoren $\bar{\mathbf{p}}_{l\,i} \in \mathbb{R}^4$ sind dabei die Zeilen der Matrix P_l. Analog erhält man drei Gleichungen für das rechte Bild. Kombiniert man jeweils die ersten beiden Gleichungen in einer Matrix $A \in \mathbb{R}^{4\times 4}$, so erhält man das lineare Gleichungssystem $A\, {}^{0}\hat{\mathbf{p}} = 0$ mit

$$
A = \begin{pmatrix} {}^{l}p_x\, \bar{\mathbf{p}}_{l3} - \bar{\mathbf{p}}_{l1} \\ {}^{l}p_y\, \bar{\mathbf{p}}_{l3} - \bar{\mathbf{p}}_{l2} \\ {}^{r}p_x\, \bar{\mathbf{p}}_{r3} - \bar{\mathbf{p}}_{r1} \\ {}^{r}p_y\, \bar{\mathbf{p}}_{r3} - \bar{\mathbf{p}}_{r2} \end{pmatrix}. \tag{3.24}
$$

Dieses Gleichungssystem ist überbestimmt, da die Lösung bis auf einen skalaren Faktor eindeutig ist. Aufgrund von numerischen Fehlern und der Kenntnis der Bildpunkte auf ganze Pixel genau besitzt dieses Gleichungssystem im Allgemeinen keine Lösung. Geometrisch bedeutet das, dass sich die Strahlen durch ${}^{k_l}\mathbf{o}$ und ${}^{l}\mathbf{p}$ bzw. ${}^{k_r}\mathbf{o}$ und ${}^{r}\mathbf{p}$ nicht schneiden, und somit nicht denselben Punkt ${}^{0}\mathbf{p}$ beschreiben. Es sind also Lösungsverfahren nötig, welche das Gleichungssystem unter Minimierung eines Fehlers lösen. Ein mögliches numerisches Verfahren ist die Singulärwertzerlegung (siehe Anhang G).

In Kapitel 12 von [14] werden weitere nichtlineare Verfahren beschrieben, welche den Punkt ${}^{0}\mathbf{p}$ rekonstruieren. Sie basieren darauf, zunächst Punkte ${}^{l}\tilde{\mathbf{p}}$ und ${}^{r}\tilde{\mathbf{p}}$ zu berechnen, welche die Epipolarbedingung (3.18) besser bzw. exakt erfüllen. Im Anschluss daran wird das beschriebene Gleichungssystem mit den neuen Bildpunkten gelöst.

Eine Verbesserung der Rekonstruktion durch diese Verfahren hängt von der Genauigkeit der bestimmten Korrespondenzen und der Kameraparameter ab. Es ist jedoch sicher, dass die Berechnungen komplexer sind und damit mehr Rechenzeit benötigen als die lineare Lösung allein. Es ist also im konkreten Fall zu untersuchen, ob eine eventuelle Verbesserung der Rekonstruktion den Mehraufwand rechtfertigt.

3.3.2 Rekonstruktion mit schwacher Kalibrierung

Im Gegensatz zur starken Kalibrierung werden bei einer schwachen Kalibrierung nicht alle Kameraparameter bestimmt. Welche der Parameter nicht

berechnet werden, hängt vom verwendeten Kalibrierungsverfahren ab und ist nicht festgelegt. Da in diesem Fall einige Informationen über das Stereokamerasystem fehlen, ist keine metrische Rekonstruktion mehr möglich. Somit ist die Rekonstruktion eindeutig bis auf eine projektive Abbildung. Die Rekonstruktion ${}^0\mathbf{p}$ liegt also in einem Referenzkoordinatensystem S_0, welches nicht zwangsläufig euklidisch ist und die Koordinaten haben auch keinen Bezug mehr zu Längeneinheiten. Ein solcher Fall liegt vor, wenn während der Kalibrierung nur die Fundamentalmatrix F bestimmt wird.

In Ergebnis 9.14 auf Seite 256 von [14] und in [69] wird die Zerlegung von F in kanonische Projektionsmatrizen P_l und P_r beschrieben. Hierfür wird der rechte Epipol als Lösung des linearen Gleichungssystems $F^T\,{}^r\hat{\mathbf{e}} = 0$ benötigt. Damit ergeben sich die kanonischen Projektionsmatrizen zu

$$P_l = (E_3; \mathbf{0}) \qquad\qquad P_r = (-{}^r\hat{\mathbf{e}}_\times\, F; {}^r\hat{\mathbf{e}}) \tag{3.25}$$

Im Gegensatz zu [14] enthält P_r hier ein negatives Vorzeichen, da sonst nach Gleichung 3.16

$$\tilde{F} = \left(P_r \begin{pmatrix}\mathbf{0}\\ 1\end{pmatrix}\right)_\times P_r\, P_l^+ = {}^r\hat{\mathbf{e}}_\times\, {}^r\hat{\mathbf{e}}_\times\, F = -F \tag{3.26}$$

die negierte Fundamentalmatrix ergibt. Diese Änderung hat keine Auswirkungen, da F eindeutig bis auf einen Skalar ist. Die Skalierung mit $\|{}^r\hat{\mathbf{e}}\|^2$ aus [69] ist nach [14] nicht notwendig. Darüber hinaus gilt $\|{}^r\hat{\mathbf{e}}\| = 1$, wenn ${}^r\hat{\mathbf{e}}$ mit der Singulärwertzerlegung berechnet wird.

Die beschriebene Berechnung der Projektionsmatrizen muss nur einmal durchgeführt werden. Im Anschluss berechnet man die Rekonstruktionen wie in Kapitel 3.3.1. Es sei nochmals erwähnt, dass die so erhaltene Rekonstruktion im Allgemeinen nicht metrisch ist.

3.4 Bildverstehen

Der Vorgang des Bildverstehens lässt sich in sechs Schritte einteilen (Kapitel 7.1 in [11]), welche in den folgenden Abschnitten erläutert werden. Außerdem wird gezeigt, wie diese Schritte mit den implementierten Algorithmen zusammenhängen. Eine ausführliche Darstellung der mathematischen Grundlagen und die theoretische Beschreibung vieler Verfahren findet sich in [23], wobei die Kategorisierung von der hier verwendeten Unterteilung abweicht.

Während der sechs Schritte des Bildverstehens werden die Informationen über die betrachtete Szene unterschiedlich dargestellt. Zu Beginn sind sie physikalischer Natur, da sie in Form von Licht vorliegen. Im ersten Schritt

werden diese Informationen digitalisiert. Die nächsten drei Schritte beinhalten die numerische Verarbeitung der so gewonnenen Daten. In den letzten beiden Schritten erfolgt die Verarbeitung vorwiegend symbolisch mit Verfahren aus dem Bereich der künstlichen Intelligenz.

Wenn schließlich das Bild von einem Computersystem verstanden wird und ein Ziel für das mobile Robotersystem definiert ist, können Aktionen geplant und ausgeführt werden. Während der Aktionen werden neue Bilder erzeugt und verarbeitet. Damit ist der Sense-Plan-Act-Zyklus als hierarchisches Steuerungsparadigma eines Roboters gegeben (Kapitel 6.1 in [9]).

3.4.1 Bilderzeugung

Die Bilderzeugung ist eine wesentliche Voraussetzung in jedem System zur grafischen Datenverarbeitung. Sie beschreibt die Umwandlung des Lichts, das von der betrachteten Szene ausgeht, in diskrete Daten, die ein Computer verarbeiten kann. Hier kommen zum Beispiel Charge-Coupled Devices (CCD) zum Einsatz. Sie bestehen aus einem zweidimensionalen Array von Silizium-Fotodioden, welche einfallende Photonen absorbieren und als messbare elektrische Ladung in MOS-Kondensatoren speichern. Mit einer entsprechenden Elektronik wird diese Ladung ausgelesen und diskretisiert (näheres hierzu beispielsweise in Kapitel 4.5.3 von [8] und Kapitel 7.2 von [11]).

Das Ergebnis dieser Digitalisierung ist eine rechteckige Matrix natürlicher Zahlen, die Informationen über die betrachtete Szene enthalten. Die Einträge der Matrix werden Pixel (kurz für Picture Element) genannt. Die Bedeutung der gespeicherten Zahlen hängt von dem verwendeten CCD-Sensor, der Kodierung und der Komprimierung ab. Gängige Darstellungen sind zum Beispiel ein Byte $(0 - 255)$ für Graustufen oder drei Byte $(0 - 16777215)$ mit je einem Byte für drei Farbkanäle RGB oder YUV.

Für das Stereokamerasystem werden zwei USB-Kameras mit CCD-Sensor verwendet (siehe auch Kapitel 4.1) und die in dieser Arbeit neu entwickelte Klasse `VideoDevice` kapselt alle zur Bilderzeugung notwendigen Methoden (Kapitel 5.1).

3.4.2 Bildvorverarbeitung

Die Bildvorverarbeitung kompensiert Fehler bei der Bilderzeugung und stellt eventuell nötige Bildeigenschaften für die Algorithmen der nächsten Verarbeitungsschritte her.

Zu dem ersten Anwendungsfall gehören beispielsweise Rauschunterdrückung durch Mittelwert- oder Medianfilter. Auch Farbverfälschungen oder radiale

Verzerrungen lassen sich aus den Bildern herausfiltern, wenn ein entsprechendes Modell für die jeweiligen Fehler des Sensors bekannt ist. Im Falle eines Stereokamerasystems ist auch eine Interpolation von asynchronen Bildern möglich, wenn der Entstehungszeitpunkt der Bilder bekannt ist (näheres hierzu in Kapitel 5.1).

Zur zweiten Gruppe gehört die Grauwerttransformation. Mit dieser wird ein Graustufenbild so umgewandelt, dass Maxima im Histogramm deutlich voneinander getrennt liegen, um später als verschiedene Objekte erkannt zu werden. Auch eine Binarisierung ist möglich, falls ein Algorithmus nicht auf Farbbildern, sondern nur auf Binärbildern operiert.

Ein weiterer Anwendungsfall der Bildvorverarbeitung ist die Tiefenbestimmung bei einem Stereokamerasystem. Viele Algorithmen zur Entfernungsberechnung [13, 37, 65, 66] haben die Vorbedingung, dass das Stereobildpaar gleichgerichtet ist. Das bedeutet, dass die Epipolarlinien (siehe Kapitel 3.2) waagerecht und für korrespondierende Punkte auf gleicher Höhe in beiden Bildern sind. Diese Eigenschaft ist durch eine Fundamentalmatrix wie in Gleichung (3.22) gegeben und kann beispielsweise durch die Verfahren in [20, 31, 46] und Kapitel 11.12 von [14] erzeugt werden. In [46] wird die Fundamentalmatrix zur verlustfreien Korrektur der Bilder verwendet. Mit dem Verfahren in [20] werden aus mindestens acht Korrespondenzen des Stereobildpaares zwei Homografien für die Umrechnung ermittelt. Diese zwei Homografien werden in [14, 31] indirekt über eine Zerlegung in einfachere Abbildungen bestimmt.

Aufgrund der beschränkten Rechenleistung in mobilen Robotersystemen sollten in der Bildvorverarbeitung nur Transformationen an den Bildern durchgeführt werden, die für die weitere Verarbeitung unbedingt nötig sind.

3.4.3 Bildverarbeitung

Verfahren in diesem Schritt sind dadurch gekennzeichnet, dass sie einzelne Pixel zu gröberen Strukturen zusammenfassen, die dann weiterverarbeitet werden. Wichtige Beispiele für die Bildverarbeitung sind das Erkennen von Linien und Ecken sowie die Segmentierung auf Basis der Farben im Bild.

Für die Linien- und Eckenerkennung sei ohne Beschränkung der Allgemeinheit ein Graustufenbild gegeben, da die gebräuchlichen Algorithmen nur die Helligkeitsänderung im Bild verwenden. Das Bild wird zum Beispiel mit einem Sobel- oder Laplace-Operator (Kapitel 7.6.4 in [11]) so gefiltert, dass Pixel in der Nähe einer starken Helligkeitsänderung hervorgehoben und die verbleibenden Pixel abgeschwächt werden. In Kapitel 4.2 von [63] ist weiterhin der Canny-Kantendetektor beschrieben, welcher nicht nur aufgrund der

Intensität der näheren Umgebung ein Pixel als Kante klassifiziert, sondern dieses Ergebnis mit weiteren Operationen verbessert.

Auf das gefilterte Bild kann beispielsweise die Hough-Transformation (Kapitel 8.2.1 in [11]) angewendet werden. Hierfür werden die als Kanten klassifizierten Pixel $(x; y)^T \in \mathbb{N}^2$ in den Parameterraum $(\theta; r)^T \in \mathbb{R}^2$ der Geraden $x \cos\theta + y \sin\theta = r$ transformiert und erhöhen einen zweidimensionalen Akkumulator für alle Lösungen der Geradengleichung. Die Geraden des Bildes sind dann einfach durch die Maxima im Akkumulator definiert. Bildbereiche werden schließlich als Objekte erkannt, wenn sie beispielsweise von Geraden umgeben sind.

Punkte in Pixelkoordinaten entsprechen sinusförmigen Funktionen im Hough-Raum. Diese Eigenschaft wird in einer Erweiterung der Hough-Transformation zur Detektion von Ecken [2] ausgenutzt. Im Akkumulator werden sinusförmige Funktionen gesucht, auf denen mindestens zwei Maxima liegen. Da ein Maximum eine Gerade repräsentiert, entsprechen die gesuchten Funktionen Punkten, die auf zwei oder mehr Geraden liegen und damit Ecken sind. Ein Nachteil dieses Verfahrens ist, dass die Schnittpunkte von Geraden nicht immer Schnittpunkte der endlichen Strecken im Bild sind.

Für die Eckenerkennung während der Kalibrierung in Kapitel 5.2.4 wird eine Funktion aus der Camera Calibration Toolbox for Matlab [5] in C++ portiert.

Auch aufgrund der Farb- bzw. Grauwerte können Objekte identifiziert werden (Kapitel 8.2.2 und 8.2.3 in [11]). Eine Möglichkeit ist hier, Schwellwerte aus der Verteilung der Farbwerte im Bild abzuleiten. Alle benachbarten Pixel, deren Farbwert zwischen zwei Schwellwerten liegen, bilden ein Objekt. Eine weitere Variante ist das Bereichswachstum (zum Beispiel [43]). Dabei werden benachbarte Pixel zu Objekten zusammengefasst, wenn sie ähnliche Farbeigenschaften aufweisen. Diese Methode wird während der Kalibrierung in Kapitel 5.2.2 zur ersten Bestimmung der Farbbereiche für die spätere Farbsegmentierung verwendet. Diese wiederum ist eine Implementierung des Algorithmus aus [6] und wird in Kapitel 5.3 näher besprochen. Eine weitere Variante in [1] verwendet Fuzzy-Logik zur Segmentierung.

Bei dem Stereosehen ist außerdem eine Bildsegmentierung auf Basis einer Verschiebung der Pixel in einem Stereobildpaar möglich. Wird ein Objekt mit zwei Kameras unterschiedlicher Position abgebildet, erscheint es in beiden Bildern an unterschiedlichen Stellen. Diese Verschiebung oder Disparität, die in Gleichung (3.14) formuliert wird, beinhaltet bei Kenntnis der Kameraparameter genügend Informationen, um die Position des Objektes im Raum festzustellen (siehe Kapitel 3.3). Der schwierige Teil der Rekonstruktion ist das so genannte Korrespondenzproblem. Dabei geht es um die möglichst genaue Bestimmung von korrespondierenden Punkten und damit der Dis-

paritäten in einem Stereobildpaar. Schließlich werden zusammenhängende Bildbereiche, die Raumpunkte konstanter Entfernung abbilden, zu je einem Objekt zusammengeschlossen.

Die Algorithmen zur Korrespondenzbestimmung werden in flächen- und merkmalsbasierte Methoden eingeteilt. Zur ersten Gruppe gehören zum Beispiel [37, 66]. Das Verfahren in [66] wird in Kapitel 2.2.3 skizziert und [37] wird implementiert und in Kapitel 5.4.1 näher erläutert. Algorithmen dieser Gruppe haben den Vorteil, dass sie großflächig Disparitäten im gesamten Stereobildpaar bestimmen. Das geschieht, indem die Ähnlichkeit von Pixelfenstern in beiden Bildern maximiert wird. Allerdings sind die Methoden sehr rechenintensiv und für den Einsatz in mobilen Robotersystemen demnach nur bedingt einsetzbar. Beispiele für die zweite Gruppe sind [10, 13, 65]. Hier werden zunächst Merkmale der Bilder wie Kanten oder Ecken extrahiert, deren Korrespondenzen im Anschluss bestimmt werden. Der Algorithmus in [10] wird in Kapitel 2.2.2 kurz beschrieben. Während in [65] Korrespondenzen für Bildbereiche mit ähnlichen Intensitäten berechnet werden, wird in [13] mit genetischen Algorithmen ein ungewöhnliches Verfahren zur Korrespondenzberechnung vorgeschlagen. Diese Algorithmen verhalten sich gegensätzlich zur ersten Gruppe, da sie normalerweise schneller sind, aber dafür nur in wenigen Bildbereichen Disparitäten bestimmen.

Für den humanoiden Roboter werden zwei neue, merkmalsbasierte Verfahren entwickelt, welche die Flächenschwerpunkte und Größen der farbsegmentierten Blobs zur Rekonstruktion verwenden. Diese Algorithmen werden in Kapitel 5.4.2 beschrieben.

Schließlich liegen die segmentierten Objekte als so genannte Blobs vor. Dabei handelt es sich um Strukturen, welche die ermittelten Informationen der entsprechenden Bildbereiche enthalten. Optional werden die Blobs komprimiert, um Speicherplatz zu sparen und auch effizienter weiterverarbeitet zu werden. In dieser Arbeit wird hierfür die Lauflängenkodierung (Run-Length Encoding, RLE) benutzt, die beispielsweise in [47] beschrieben wird.

3.4.4 Bildbeschreibung

Nachdem im vergangenen Verarbeitungsschritt Bildbereiche als potentiell interessante Objekte bestimmt worden sind, werden nun Charakteristika dieser Bildbereiche berechnet. Hierzu gehören zum Beispiel die Bounding Box (ein Rechteck, das den kompletten Bereich enthält), die Fläche in Pixeln, der Flächenschwerpunkt, die Flächenmomente, die Kontur und die konvexe Hülle. Für das Stereosehen kommt noch die räumliche Position hinzu. Einige dieser Werte sind für Blobs im RLE sehr effizient zu berechnen und in [47]

näher erläutert. Auch in den Kapiteln 8.3 und 8.4 von [11] sind diese und andere Merkmale illustriert.

Allgemein sollten die Charakteristika invariant gegen Skalierung, Rotation und Translation sein. Außerdem müssen sie robust gegen Rauschen und aussagekräftig für die betrachteten Objekte sein. Für den Einsatz in mobilen Robotersystemen ist einmal mehr die effiziente Berechnung wichtig.

In Kapitel 5.3.2 wird die Berechnung der Bounding Box, der Blob-Größe und des Flächenschwerpunkts während der Farbsegmentierung erläutert. Auf die Positionsberechnung wird in Kapitel 5.4.2 eingegangen.

3.4.5 Objekterkennung

Die Objekterkennung ist ein vielfältiger und komplexer Schritt des Bildverstehens, der zum Beispiel in den Kapiteln 8.5 von [11] und 10 von [63] behandelt wird. Damit Objekte erkannt werden können, muss eine Wissensbasis vorliegen, in der Modelle bekannter und damit erkennbarer Objekte beschrieben werden. Eine solche Wissensbasis enthält beispielsweise die Merkmale aus Kapitel 3.4.4. Damit ist die Aufgabe der Objekterkennung, das Modell in der Wissensbasis zu finden, das am besten mit einem Blob übereinstimmt.

Es ist auch möglich, ein so genanntes Template-Matching durchzuführen. Dabei enthält die Wissensbasis Beispielaufnahmen der bekannten Objekte aus verschiedenen Blickwinkeln. Die Objekterkennung muss hier die Schablone in der Wissensbasis finden, die dem Blob am ähnlichsten sieht. Wissensbasen dieser Art sind allerdings sehr groß, was bei dem beschränkten Speicherplatz eines mobilen Robotersystems problematisch ist.

Ein weiteres Problem der Objekterkennung ist, dass Objekte bei unterschiedlicher Beleuchtung oder aus verschiedenen Richtungen sehr unterschiedlich aussehen können und verschiedene Merkmale aufweisen. Dies ist bei dem Einsatz des Humanoiden im *RoboCup* einfacher. Die wichtigen Objekte sind hier vorrangig durch ihre Farbe gekennzeichnet. Wird beispielsweise ein orangefarbener Blob detektiert, so ist es bereits sehr wahrscheinlich, dass es sich dabei um den Ball handelt, da diese Farbe (außer eventuell im Zuschauerraum) sonst nicht verwendet wird. Lediglich die Farbbereiche für die Segmentierung müssen den jeweiligen Lichtverhältnissen angepasst werden, damit die Erkennung zuverlässig funktioniert. Dieser Vorgang ist Teil der Kalibrierung und wird in Kapitel 5.2.2 dargestellt. Auch bei der Bewegung in den Gängen eines Gebäudes kommen nur einfache Objekte, wie zum Beispiel Türen und Wände, vor. Alle anderen nicht erkannten Objekte können als Hindernisse betrachtet werden.

Schließlich ist die Erstellung der Wissensbasis keine Trivialität. Bei wenigen

einfachen Objekten kann sie manuell angelegt werden. Es ist jedoch möglich und ratsam, automatische Lernverfahren einzusetzen, um eine Datenbank anzulegen oder zu pflegen. Ein Überblick über verschiedenste Lernverfahren ist in den Kapiteln 18 bis 21 von [49] gegeben.

Besonders interessant und vielfach untersucht sind neuronale Netze [26, 34]. In [34] werden Objekte zunächst in logarithmische Polarkoordinaten transformiert, normalisiert und zentriert. Anschließend lernt ein neuronales Netz, verschiedene geometrische Formen zu erkennen, die in entsprechender Weise dargestellt werden. Das vorgestellte Verfahren kann sehr ähnliche Formen, wie Quadrate und nahezu quadratische Rechtecke oder Kreise und Achtecke, unterscheiden. Bei [26] lernt ein neuronales Netz mit rotations- und skalierungsinvarianten Merkmalen der Fouriertransformierten von Objekten.

Im Rahmen dieser Diplomarbeit werden Objekte ausschließlich an ihrer Farbe erkannt, um eine schnelle Verarbeitung zu ermöglichen.

3.4.6 Bildanalyse

Den Abschluss des Bildverstehens bildet die Bildanalyse. Die segmentierten Blobs aus Kapitel 3.4.3 besitzen inzwischen verschiedene Eigenschaften aus Kapitel 3.4.4 und einen Objekttyp aus Kapitel 3.4.5.

Wie bei der Objekterkennung ist auch bei der Bildanalyse eine Wissensbasis notwendig. Im *RoboCup* enthält diese zum Beispiel, dass es genau ein blaues Tor gibt und dass der Ball immer auf dem Rasen liegt. Es sind auch Regeln in der Wissensbasis gespeichert, mit deren Hilfe neues Wissen hergeleitet werden kann. Eine dieser Regeln könnte besagen, dass der Ball vor dem Tor liegt, wenn der Humanoide steht und der Ball im Bild unterhalb vom Tor erscheint. Wenn die Wissensbasis ständig aktualisiert wird, kann sie auch die Positionen der Gegner und des Balls zu vergangenen Zeitpunkten enthalten.

Je nachdem, wie die Informationen der vergangenen Verarbeitungsschritte vorliegen, ist ein Formalismus für die Wissensrepräsentation zu wählen. Wenn ein orangefarbener Blob einfach als Ball erkannt wird, so kann beispielsweise die Prädikatenlogik erster Stufe zur Darstellung verwendet werden. Allerdings können die Objektbezeichnungen und ihre Merkmale mit einer gewissen Unsicherheit behaftet sein. In diesem Fall ist die Prädikatenlogik nicht mehr geeignet und es muss ein Formalismus mit Wahrscheinlichkeiten gewählt werden.

Zur Analyse des Bildes wird schließlich eine Hypothese gesucht, die den Fakten der Wissensbasis und den Beobachtungen aus den vergangenen Verarbeitungsschritten nicht widerspricht. Darüber hinaus müssen aus den Fakten

und der Hypothese die Beobachtungen herleitbar sein. Je nach dem gewählten Formalismus kommen hierfür andere Verfahren zum Einsatz. Bei der Prädikatenlogik sind zum Beispiel Konnektion und Resolution zu nennen. Wenn die Informationen mit Zeitpunkten versehen sind, eignet sich das Fluentenkalkül. Für unsichere Daten können beispielsweise bayessche Netze und Fuzzy-Logik benutzt werden.

Die genannten Formalismen und Verfahren sind in [4] und den Kapiteln 8 bis 10 und 13 bis 15 von [49] ausführlich erläutert. Außerdem findet sich weiterführende Literatur in den genannten Quellen.

Kapitel 4

Hardware

4.1 Kameras

Zur Bilderzeugung kommen zwei USB-Kameras des Modells ToUcam PRO II (PCVC840K) von Philips zum Einsatz (Abbildung 1.2). Auf der offiziellen Webseite [45] ist zu Beginn der Diplomarbeit noch ein Datenblatt angeboten worden, welches auf der beigelegten CD archiviert ist. Inzwischen finden sich dort nur noch Treiber und keine weiteren Informationen über die verwendete Kamera.

Angesprochen werden die Kameras über das PWC Kernel-Modul von [38]. Während der Diplomarbeit ist die Betreuung dieses Moduls von einem anderen Entwickler übernommen worden. Die aktuelle Version ist auf der Webseite [50] zu finden.

Die von den Kameras gelieferten Bilder der Größe 160×120 Pixel werden mittels Double Buffering gespeichert (näheres hierzu in Kapitel 5.1). Um die Daten mit dem Maximum von 30 Bildern pro Sekunde zu erhalten, wird das Kernel-Modul mit dem Befehl `modprobe pwc size=qsif fps=30 mbufs=2` geladen.

Für die verwendeten Algorithmen zur Tiefenbestimmung ist es wichtig, welches Bild von der linken bzw. rechten Kamera kommt. Hierfür müssen die Kameras aus einem Programm heraus unterschieden werden. Sind keine weiteren Video-Geräte angeschlossen, erhalten die Kameras die Gerätenamen `/dev/video0` und `/dev/video1`. Welche Kamera zu welchem Gerätenamen gehört, lässt sich nicht feststellen und hängt unter anderem davon ab, in welcher Reihenfolge die USB-Kabel eingesteckt werden. Sind beim Systemstart bereits beide Kameras angeschlossen, so ist nicht definiert, welcher Name welchem Gerät zugeordnet wird. Hierfür unterstützt das PWC-Modul den Parameter `dev_hint`, der in Abhängigkeit der Seriennummer der Kamera

bestimmte Gerätenamen vergibt. Die Datei `/proc/bus/usb/devices` zeigt bei angeschlossenen Kameras jedoch, dass beide identische Seriennummern besitzen, wodurch eine Identifikation der Kameras aufgrund der Seriennummern unmöglich ist.

Eine eingeschränkte Lösung für das Problem bietet das Programm udev. Dieser Bestandteil aktueller Linux-Distributionen vergibt anhand vieler Eigenschaften von Geräten bestimmte Gerätenamen. Sind die Kameras angeschlossen, so werden mit dem Befehl `udevinfo -a -p ‘udevinfo -q path -n /dev/videoX‘` alle verfügbaren Informationen über das Gerät angezeigt, wobei `/dev/videoX` der aktuell vergebene Gerätename ist. Die Informationen der Kameras unterscheiden sich nur in den verwendeten USB-Steckplätzen. Es ist also möglich, anhand des physikalischen USB-Anschlusses die Kameras zu unterscheiden, solange immer derselbe Anschluss für die linke bzw. rechte Kamera verwendet wird. Für die Nutzung von udev ist mindestens ein Linux-Kernel der Version 2.6 nötig, der weder auf dem Entwicklungsrechner noch auf dem Humanoiden installiert ist. Genauere Informationen über udev und das benutzte Testsystem stehen in Anhang C.

Aus dem Datenblatt der Kamera kann man die technischen Daten in Tabelle 4.1 entnehmen. Für die Berechnung des vertikalen Öffnungswinkel wird das Seitenverhältnis eines Bildes von 4 : 3 verwendet. Die angegebene Abmessung gilt für den zusammengeklappten Zustand ohne Objektiv.

Dimension	Wert
Auflösung	160×120
θ_x	$33°$
θ_y	$\frac{3}{4} \cdot \theta_x = 24{,}75°$
Abmessung	$70 \times 47 \times 27\,mm$
Masse	$100\,g$

Tabelle 4.1: Technische Daten der verwendeten Kamera ToUcam PRO II von Philips.

Unter der Annahme aus Gleichung (3.6) sowie rechtwinkliger Bildkoordinatenachsen mit $\varphi = \frac{\pi}{2}$ ergibt sich aus Gleichung (3.8) eine theoretische Kameramatrix

$$K = \begin{pmatrix} -270{,}075 & 0 & 80 \\ 0 & -273{,}465 & 60 \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.1}$$

Diese Kameramatrix wird bei der Konzeption des Stereogestells in Kapitel 4.2 und zum Einschätzen der Kalibrierungsergebnisse in Kapitel 5.2.3 benutzt.

4.2 Konzeption des Stereogestells

In diesem Abschnitt werden die Dimensionen des Stereogestells aus den Anwendungsgebieten und technischen Voraussetzungen entwickelt.

Der Humanoide soll sich auf einem *RoboCup*-Feld der Mid Size League [48] und in den Gängen eines Gebäudes bewegen und im Nahbereich mit seiner Umgebung interagieren. In Gängen muss er möglichst weit und bei der Interaktion möglichst nah sehen.

Aufgrund der Breite des Kopfes mit 18,1 mm und der Kameras mit 47 mm ist der kleinste mögliche Abstand der Foki beider Kameras 65,1 mm. Da die Halterung selbst noch etwas Platz benötigt, ist der minimale Abstand $b_{min} = 95\,mm$.

Verwendet man Gleichung (3.10) und den einfachen Fall paralleler Kameras, so ergibt sich bei $\theta_x = 33°$

$$d_{min} = 1{,}688\,b, \qquad d_{max} = \infty. \tag{4.2}$$

Theoretisch kann man mit einer solchen Konfiguration beliebig weit sehen. Wie in Kapitel 3.1.2 erwähnt, ist die Auflösung der Entfernung bei kleiner Disparität gering, wodurch eine verlässliche Positionsberechnung unmöglich ist.

Setzt man Gleichung (4.1) in (3.15) ein, so erhält man die Disparität in Abhängigkeit des Stereoparameters b und des Objektabstandes ${}^0p_z$

$$\mathbf{d}_{rl} = \begin{pmatrix} 270{,}075\frac{b}{{}^0p_z} \\ 0 \end{pmatrix}. \tag{4.3}$$

Im Folgenden wird eine minimale Disparität von fünf Pixeln angenommen. Dadurch ist eine praktisch maximale Objektentfernung

$$\tilde{d}_{max} = \frac{270{,}075}{5} b = 54{,}015\,b \tag{4.4}$$

gegeben.

Soll zum Beispiel der AIBO-Ball [57] mit einem Radius von $r = 43\,mm$ in beiden Bildern vollständig sichtbar sein, so erhöht sich der minimale Abstand gemäß Abbildung 4.1 um d_b auf

$$\tilde{d}_{min} = d_{min} + d_b = d_{min} + \frac{r}{\sin\frac{\theta+\theta_x}{2}} = 1{,}688\,b + 151{,}4\,mm. \tag{4.5}$$

Da für $b = 95\,mm$ der Bereich von 311,8 mm bis 5131,4 mm nicht den Nahbereich für die Interaktion abdeckt, muss eine Konfiguration mit $\theta \neq 0$ gewählt werden.

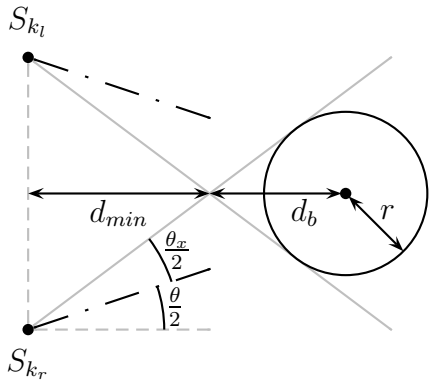


Abbildung 4.1: Mindestabstand einer Kugel.

Aus Gleichung (3.10) folgt mit minimalem b

$$\tilde{d}_{min} = d_{min} + d_b = \frac{95}{2\tan\frac{\theta+33^\circ}{2}}mm + \frac{43}{\sin\frac{\theta+33^\circ}{2}}mm. \tag{4.6}$$

Setzt man für $\tilde{d}_{min} = 200\,mm$ ein, so ergibt sich $\theta = 17{,}87^\circ$. Daraus erhält man wiederum eine maximale Sichtweite von

$$d_{max} = \frac{95}{2\tan\frac{17{,}87^\circ}{2}}mm = 302{,}14\,mm. \tag{4.7}$$

Diese Konfiguration ermöglicht nicht die nötige Fernsicht, wenn das Verfahren zur Bestimmung der Disparitäten nur Objekte in einer Entfernung bis zum Schnittpunkt der optischen Achsen verarbeiten kann.

Durch den kleinen horizontalen Öffnungswinkel von $\theta_x = 33^\circ$ sind also je nach Anwendung zwei Gestelle nötig. Mit den bisher beschriebenen Formeln für den minimalen und maximalen Abstand werden zwei Gestelle mit den Eigenschaften in Tabelle 4.2 gebaut, da auf dem Markt kein passendes Equipment verfügbar ist.

Anwendung	b	θ	d_{min}	$\tilde{d}_{min}$	d_{max}
Nahsicht	$95\,mm$	15°	$106{,}7\,mm$	$212{,}4\,mm$	$360{,}8\,mm$
Fernsicht	$140\,mm$	0°	$236{,}3\,mm$	$387{,}7\,mm$	$\infty\,(7562{,}1\,mm)$

Tabelle 4.2: Parameter und Sichtweiten der Stereogestelle.

Die Notwendigkeit von zwei Gestellen würde durch Kameras mit einem größeren Öffnungswinkel bzw. einer vorgeschalteten Weitwinkeloptik oder einem aktiven Stereokamerasystem [53, 54, 59] entfallen. Letzteres bringt allerdings einen Mehraufwand bei der Konstruktion und Regelung mit sich.

4.3 Konstruktion der Stereogestelle

Da keine Befestigungsteile in den passenden Dimensionen erhältlich sind und der Aufbau möglichst leicht sein soll, werden die beiden in Kapitel 4.2 konzipierten Stereogestelle aus einem Winkelprofil aus Aluminium gebaut. Das hier verwendete Profil aus einem Baumarkt weist die Dimensionen in Abbildung 4.2 auf.

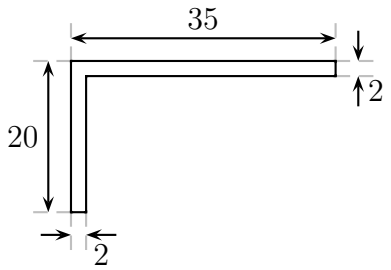


Abbildung 4.2: Dimensionen des Aluminium-Winkelprofils in mm.

Die Gestelle bestehen aus je drei Teilen. Zur Befestigung der Kameras dienen die Teile in Abbildung 4.3 bzw. 4.4. Für die Befestigung am Kopf des Humanoiden sind jeweils zwei Ausführungen von Abbildung 4.5 nötig, welche sich durch eine Spiegelung unterscheiden.

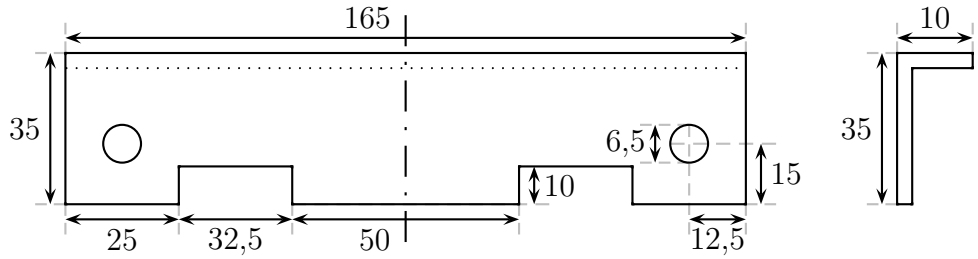


Abbildung 4.3: Dimensionen des Gestells für Fernsicht in mm.

Die Kameras werden mit Stativschrauben mit einem Gewindedurchmesser von $\frac{1}{4}'' = 6{,}35\,mm$ und einer Länge von $9\,mm$ an den Gestellen befestigt. Solche Schrauben sind in einem Fotofachgeschäft erhältlich. Da die Gewinde der Kameras nicht tief genug sind, werden zusätzlich ein bis zwei Unterlegscheiben verwendet. Die angegebene Entfernung der Bohrungen vom hinteren Rand ist mit $15\,mm$ etwas zu groß. Nach dem Bohren werden die Löcher langsam aufgefeilt, bis die Kameras exakt passen. Dadurch wird sichergestellt, dass die Kameras torsionsfest fixiert sind und die optischen Achsen

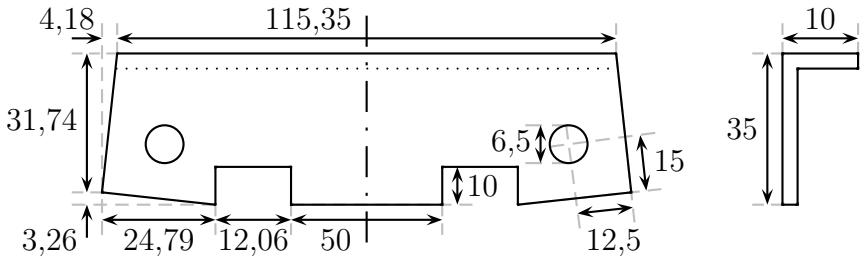


Abbildung 4.4: Dimensionen des Gestells für Nahsicht in mm.

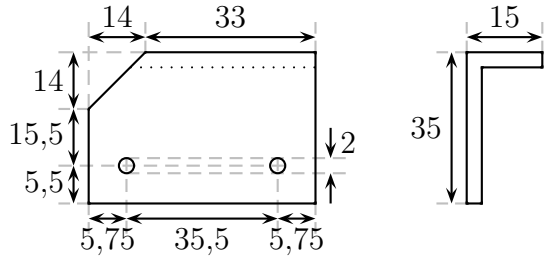


Abbildung 4.5: Dimensionen der Halterung in mm.

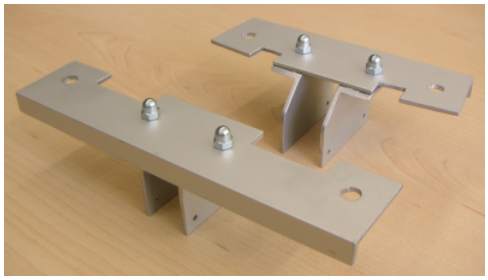
somit den definierten Konvergenzwinkel θ einschließen. Da das Gewinde der Kamera an dem beweglichen Ständer befestigt ist, kippt die eigentliche Kamera leicht nach vorne und hinten. Durch Fixierung der Kameras an den Ständern mithilfe von einfachen Gummiringen wird dieses Problem gelöst.

Die überstehenden Winkel der Länge $10\,mm$ an den Vorderseiten der Gestelle dienen der Stabilisierung. Der Träger der Kameras kann sich so nicht verbiegen. Der Originalwinkel wird an diesen Stellen um $10\,mm$ gekürzt, um Gewicht zu sparen ohne die Stabilität zu verringern. Die kurze Seite der Halterungen wird ebenfalls gekürzt, um Gewicht zu sparen und im Falle der Nahsicht die Kameras näher zusammenrücken zu können.

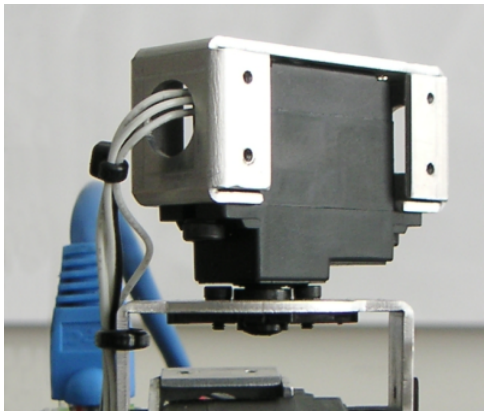
Sind die einzelnen Teile fertig bearbeitet, müssen sie zusammengefügt werden. Hierzu werden je zwei Halterungen an dem Kopf festgeschraubt. Daraufhin wird das Gestell mit Zweikomponentenkleber zentriert an die Halterungen geklebt. Dadurch ist eine exakte Passform der Stereogestelle am Kopf des Humanoiden gegeben. Wenn der Klebstoff ausgehärtet ist, werden die Halterungen mit je einer Schraube fixiert, damit sie auch größeren Belastungen standhalten.

Ein Stereogestell mit beiden Kameras hat seinen Schwerpunkt relativ genau über dem Mittelpunkt der Grundfläche. Dadurch werden die Belastungen der Genickmotoren gering gehalten.

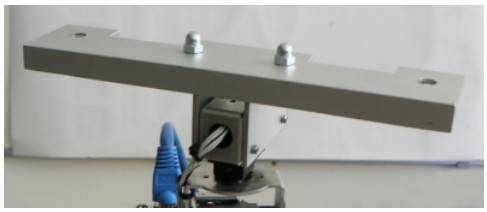
Abbildung 4.6 zeigt die fertig gestellten Stereogestelle und schrittweise ihre Befestigung an dem Kopf des humanoiden Roboters.



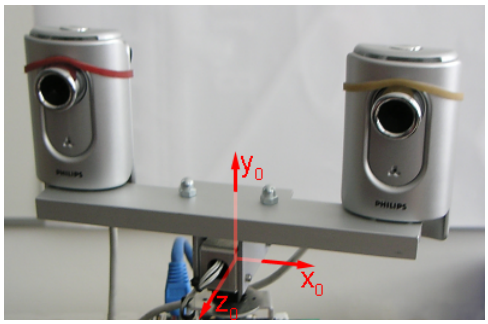
(a)



(b)



(c)



(d)

Abbildung 4.6: (a) beide Stereogestelle (b) Kopf des Humanoiden (c) Gestell für Fernsicht am Kopf (d) vollständiges Stereokamerasystem mit dem verwendeten Referenzkoordinatensystem S_0

Kapitel 5

Software

Im Rahmen dieser Diplomarbeit sind über 27000 Zeilen Programm-Code inklusive Kommentaren erstellt worden. Aufgrund des Umfangs wird der Quelltext hier nicht abgedruckt. Vielmehr ist er auf der beigelegten CD und in dem Subversion-Repository [56] auf einem Server des Fachgebiets Simulation und Systemoptimierung verfügbar. Mithilfe des Tools Doxygen [17] kann aus speziellen Kommentaren im Quellcode eine API-Referenz der entwickelten Klassen generiert werden. Auch diese befindet sich auf der CD.

Entwickelt und getestet wird die Software auf einem Notebook, dessen technische Daten in Anhang A.1 zu finden sind. Abschließende Tests werden auf dem Humanoiden (siehe Anhang A.2) durchgeführt.

Gemäß der Aufgabenbeschreibung in Kapitel 1.2 können überwachende Tätigkeiten nicht auf dem System des Humanoiden durchgeführt werden. Darüber hinaus soll die Software möglichst einfach mit anderen Robotern benutzbar sein. Beide Anforderungen lassen sich mit *RoboFrame* [44] erfüllen. Dabei handelt es sich um ein Softwareframework, das momentan am selben Fachgebiet der Technischen Universität Darmstadt entwickelt wird.

Die Struktur von Anwendungen im *RoboFrame* ist in Abbildung 5.1 dargestellt. *RoboFrame* besteht aus den zwei Teilen *RoboApp* und *RoboGui*. Auf dem Robotersystem läuft eine Applikation unter Verwendung von *RoboApp*, welche beliebige Module enthält. Module sind eigenständige Programmteile, die beispielsweise das Bildverstehen, die Bahnplanung oder die Selbstlokalisierung realisieren und sich einfach zu neuen Applikationen für andere Systeme kombinieren lassen. Auf einem Kontrollsystem kann optional eine grafische Oberfläche ausgeführt werden, die *RoboGui* verwendet. In dieser sind beliebige Dialoge registriert, die bestimmte Daten darstellen können. Auch die Dialoge können zu neuen Oberflächen zusammengestellt werden. Alle genannten Komponenten verfügen über definierte Schnittstellen, über die sie Informationen zur Verfügung stellen oder empfangen. Dabei ist die Kommu-

nikation zwischen den einzelnen Komponenten transparent durch *RoboFrame* realisiert.

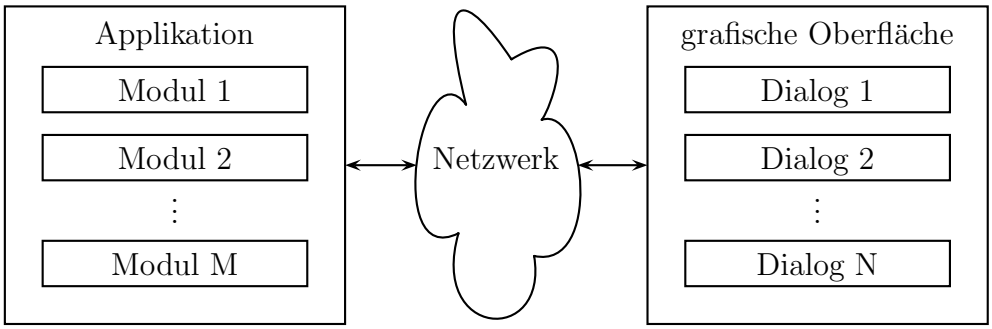


Abbildung 5.1: Struktur des Frameworks *RoboFrame*.

Das Framework selbst ist plattformunabhängig und *RoboApp* unterstützt zur Zeit die Betriebssysteme Unix, Linux, Mips, Windows32 und WindowsCE. *RoboGui* benötigt die Bibliothek Qt [62], um grafische Oberflächen zu erzeugen. Dadurch sind die Oberflächen nur unter Unix und Linux mit einem X-Server sowie Windows32 ausführbar. Bei neuem Code ist lediglich auf einige Besonderheiten der jeweiligen C++-Compiler zu achten, um ebenfalls plattformunabhängige Applikationen und Oberflächen zu entwickeln.

Da der PWC-Treiber für die Webcams nur für Unix, Linux und Mips verfügbar ist, können die Applikationen zum Stereosehen nur auf diesen Systemen ausgeführt werden. Die grafische Oberfläche zur Kommunikation mit den Applikationen ist zusätzlich unter Windows32 kompilier- und ausführbar.

In Anhang D ist beschrieben, wie sich die implementierten Module und Dialoge in neue Anwendungen integrieren lassen.

In den folgenden Abschnitten werden ausgewählte Komponenten der entwickelten Anwendungen beschrieben. Einen Überblick über das Zusammenspiel der Komponenten ist in Abbildung 5.2 ersichtlich, während detaillierte Informationen über die konkreten Implementierungen in der API-Referenz enthalten sind.

Das Licht der betrachteten Szene wird von beiden Kameras in ein digitales Stereobildpaar umgewandelt, das anschließend segmentiert wird. Aus den so gewonnenen Blobs beider Bilder werden Korrespondenzen bestimmt, die zur Rekonstruktion verwendet werden, anhand derer die Blobs um eine metrische Position im Referenzkoordinatensystem zu erweitern. Diese können dann von beliebigen Modulen verarbeitet werden, damit das Robotersystem auf die erkannten Objekte reagieren kann. Für diesen Vorgang müssen zuvor die

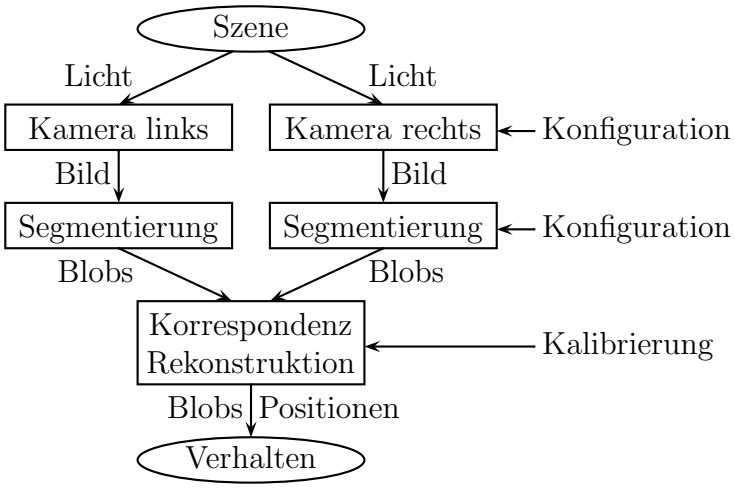


Abbildung 5.2: Überblick über das Zusammenspiel der Software-Komponenten.

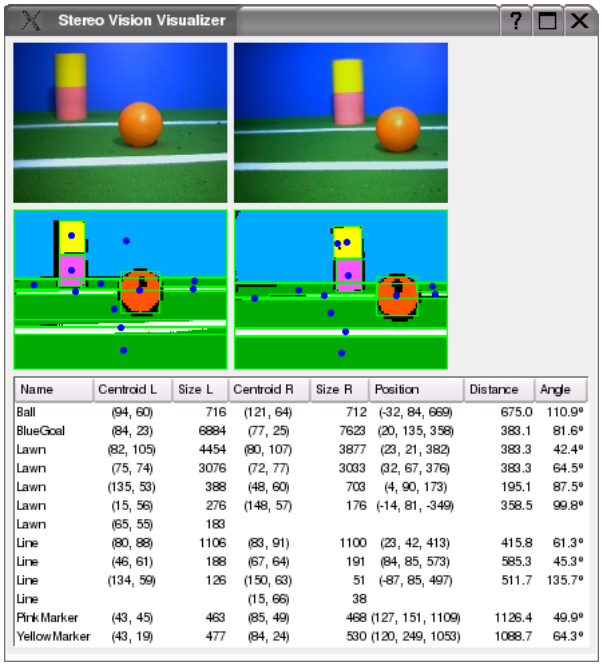
Kameras und die Farbbereiche für die Segmentierung konfiguriert und das Stereokamerasystem für die Rekonstruktion kalibriert werden.

Zur Überwachung des durchgeführten Bildverstehens wird der Dialog `Visualizer` verwendet, der in Abbildung 5.3 zu sehen ist. Im oberen Teil wird zunächst das Stereobildpaar von den Kameras angezeigt. Darunter befindet sich das segmentierte Stereobildpaar mit den hellgrünen Bounding Boxen und den dunkelblauen Flächenschwerpunkten der erkannten Objekte. Im unteren Teil befindet sich schließlich eine Tabelle, welche die Pixelkoordinaten der Schwerpunkte sowie die Objektgrößen anzeigt. Für detektierte Korrespondenzen ist weiterhin die metrische Position, die Entfernung und ein Winkel angegeben. Letzterer wird in Kapitel 6.3.2.3 definiert.

5.1 Bilderzeugung

Die Verbindung zu den Kameras wird über das Kernel-Modul PWC [38, 50] und Video4Linux [24] hergestellt.

Eine Instanz der in dieser Diplomarbeit entwickelten Klasse `VideoDevice` repräsentiert eine Kamera und kapselt alle Einstellungsmöglichkeiten des Treibers. Bei der Instantiierung der Klasse wird der Gerätename angegeben, wie zum Beispiel `/dev/video0`. Wird das Gerät mit der Methode `openDevice()` geöffnet, so liefert die Kamera Bilder in der eingestellten Rate. Mit der Methode `captureImage()` wird ein Bild vom Treiber angefordert, auf welches man schließlich mit der Methode `getImage()` zugreift. Die Methode



Name	Centroid L	Size L	Centroid R	Size R	Position	Distance	Angle
Ball	(94, 60)	716	(121, 64)	712	(-32, 84, 669)	675.0	110.9°
BlueGoal	(84, 23)	6884	(77, 25)	7623	(20, 135, 358)	383.1	81.6°
Lawn	(82, 105)	4454	(80, 107)	3877	(23, 21, 382)	383.3	42.4°
Lawn	(75, 74)	3076	(72, 77)	3033	(32, 67, 376)	383.3	64.5°
Lawn	(135, 53)	388	(48, 60)	703	(4, 90, 173)	195.1	87.5°
Lawn	(15, 56)	276	(148, 57)	176	(-14, 81, -349)	358.5	99.8°
Lawn	(65, 55)	183					
Line	(80, 88)	1106	(83, 91)	1100	(23, 42, 413)	415.8	61.3°
Line	(46, 61)	188	(67, 64)	191	(84, 85, 573)	585.3	45.3°
Line	(134, 59)	126	(150, 63)	51	(-87, 85, 497)	511.7	135.7°
Line			(15, 66)	38			
PinkMarker	(43, 45)	463	(85, 49)	468	(127, 151, 1109)	1126.4	49.9°
YellowMarker	(43, 19)	477	(84, 24)	530	(120, 249, 1053)	1088.7	64.3°

Abbildung 5.3: Dialog zur Überwachung des Bildverstehens.

`captureImage()` blockiert die Anwendung so lange, bis ein Bild vollständig vorliegt.

In der Methode `captureImage()` wird der Aufruf `VIDIOCSYNC` an den Treiber gesendet, um das nächste Bild anzufordern. Dieser Aufruf terminiert nicht immer und die Programmausführung wird dadurch nicht reproduzierbar angehalten. Auf dem Humanoiden tritt dieses Problem reproduzierbar auf, sobald die Kameras mehr als fünf Bilder pro Sekunde liefern sollen. Dies ist eventuell in der aktuellen Version des PWC-Treibers behoben, die jedoch nicht mit dem momentan installierten Linux-Kernel zusammenarbeitet.

Ein einfaches Programm zur Verarbeitung der Videodaten ist in Listing 5.1 gegeben. Benötigt die Verarbeitung des Bildes mehr Zeit, als zwischen zwei Bildern des Datenstroms zur Verfügung steht, werden überschüssige Bilder vom Treiber verworfen.

Der Zugriff auf die Bilddaten erfolgt hierbei über Memory Mapping. Dabei greift die Anwendung direkt auf die Bildspeicher des Treibers zu, wodurch die Zeit zum Kopieren der Daten und Speicherplatz gespart werden. Weiterhin wird Double Buffering verwendet. Das bedeutet, dass ein Bild in einem Puffer verarbeitet werden kann, während die Kamera das nächste Bild in einem anderen Puffer ablegt.

Die verwendeten Kameras liefern die Bilddaten im Format YUV420P. Der

Listing 5.1: Einfaches Programm zum Zugriff auf Videodaten.

```
VideoDevice vd("/dev/video0");
vd.openDevice();
const unsigned char *data;
while (running)
{
  vd.captureImage();
  data = vd.getImage();
  processImage(data);
}
vd.closeDevice();
```

Farbraum ist also YUV und der Y-Kanal (Helligkeit) wird in x- und y-Richtung jeweils doppelt so oft abgetastet, wie die U- und V-Kanäle (Farbe). Zu einem Block von 2×2 Y-Werten gehört je ein U- und V-Wert, da das menschliche Auge Helligkeitsunterschiede besser wahrnimmt als Farbunterschiede. Zusätzlich sind die Daten planar, das heißt bei einem Bild der Größe $w \times h$ Pixel werden jeweils zeilenweise zunächst alle $w \cdot h$ Y-Werte gefolgt von je $\frac{w \cdot h}{4}$ U- und V-Werten, gespeichert.

Bei der Verwendung von zwei Kameras für das Stereosehen tritt das Problem der Synchronisierung auf. Für die Positionsbestimmung von Objekten müssen die Bilder eines Stereobildpaares vom selben Zeitpunkt stammen, da sonst durch die Bewegung des Objektes oder des Roboters unter Umständen große Fehler in der berechneten Position möglich sind. Die benutzten Webcams können nicht synchronisiert werden. Sind die Videogeräte geöffnet, liefern sie Bilder in einer konstanten Rate. Ohne direkt auf die Steuerelektronik der Kameras zuzugreifen, ist es daher nicht möglich, ein Bild zu einem bestimmten Zeitpunkt anzufordern. Die Geräte in einzelnen Threads gleichzeitig zu öffnen, bringt keine Verbesserung, da beide Videoströme und Steuerungsbefehle über einen USB-Bus transportiert werden.

Dieses Problem kann durch den Einsatz anderer Hardware gelöst werden. Zum einen kann man ein Pseudo-Stereokamerasystem wie in [27, 41, 42] oder Kapitel 4.7.2 von [9] verwenden. Bei einigen dieser Lösungen sind allerdings andere Algorithmen zur Verarbeitung nötig, da nicht zwei eigenständige Bilder vorliegen, sondern ein Stereobildpaar überlagert wird. Eine weitere Möglichkeit der Synchronisierung bietet die Verwendung von Kameras, die programmgesteuert und damit gleichzeitig einzelne Bilder liefern.

Durch die Auswertung von Zeitstempeln der Bilder in der Software bietet sich die Möglichkeit, synchrone Stereobildpaare zu interpolieren. Die benutzten Kameras und Video4Linux unterstützen dies allerdings nicht. In der Methode

`captureImage()` der Klasse `VideoDevice` wird der Zeitpunkt gespeichert, zu dem das Bild von dem Treiber geholt wird. Dieser Zeitpunkt ist allerdings ungeeignet, wenn die Verarbeitung der Bilder länger dauert als zwischen zwei Bildern des Videostroms zur Verfügung steht. In diesem Fall bekommt `captureImage()` sofort das bereits fertiggestellte Bild vom Treiber, speichert aber den Zeitpunkt des Methodenaufrufs. Damit ist kein Zusammenhang zwischen Bild und Zeitstempel mehr gegeben.

In der neuen Version Video4Linux2 ist in der Struktur `v4l2_buffer` ein Feld `timestamp` vorgesehen, welches in der Version 10.0.7 des PWC-Moduls erstmals unterstützt wird. Dieses Modul benötigt aber mindestens einen Linux-Kernel der Version 2.6, welcher auf dem Humanoiden bisher nicht installiert ist. Laut der Spezifikation von `v4l2_buffer` in [24] sollte die Zeit gespeichert werden, wenn das erste Byte des Bildes im Puffer abgelegt wird. Eine Untersuchung des Quelltextes des PWC-Moduls zeigt allerdings, dass die Zeitstempelung sehr ähnlich realisiert ist, wie in der Klasse `VideoDevice`. Es wird ebenfalls der Zeitpunkt hinterlegt, zu dem das Bild vom Treiber abgeholt wird. Die Qualität der Zeitstempel wird dadurch nicht verbessert.

Das Problem der ungenauen Zeitstempel wird im Rahmen dieser Arbeit nicht gelöst. Dazu ist die Anpassung des Treibers PWC nötig, so dass dieser die Spezifikation von Video4Linux2 erfüllt. Außerdem wird ein aktueller Linux-Kernel auf dem Humanoiden benötigt. Die Software zur Positionsbestimmung, welche ein synchrones Stereobildpaar benötigt, wird dadurch nur mit einem unbewegten Stereokamerasystem und stationären Objekten getestet.

5.2 Kalibrierung

Die Kalibrierung des Stereokamerasystems erfolgt in drei Schritten. In Kapitel 5.2.1 werden die Parameter erläutert, welche die Eigenschaften der Bilder beeinflussen und vom Kamera-Treiber zur Verfügung gestellt werden. Kapitel 5.2.2 beschreibt die Definition der Farbbereiche, die für die Segmentierung benötigt werden. In Kapitel 5.2.3 und 5.2.4 werden schließlich verschiedene Verfahren und ihre Probleme beschrieben, um die Projektionsmatrizen P_l und P_r aus Gleichung (3.13) bzw. die Fundamentalmatrix F aus Gleichung (3.16) zu bestimmen.

Die ermittelten Parameter werden in einem System von Konfigurationsdateien gespeichert, die dann von den Algorithmen zum Bildverstehen verwendet werden.

5.2.1 Videoparameter

Die verwendeten Webcams verfügen durch den PWC-Treiber über viele Parameter, mit denen die Bildqualität gesteuert wird. In Abbildung 5.4 ist der Dialog zu sehen, mit dem diese Parameter für beide Kameras getrennt eingestellt werden können, wobei nur die Bedienelemente für die linke Kamera in der Abbildung sichtbar sind.

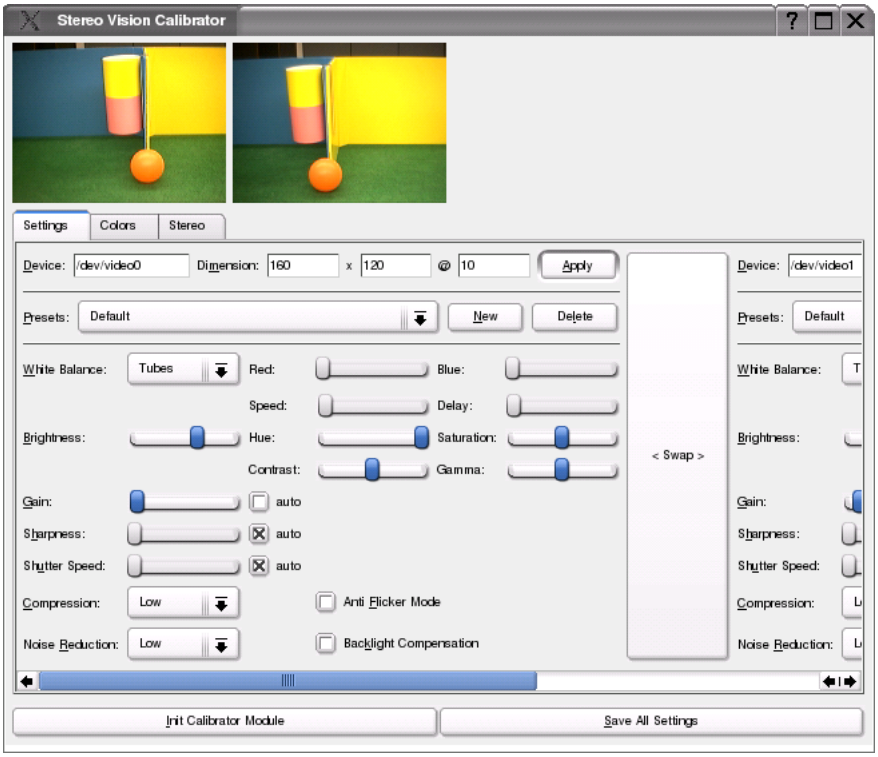


Abbildung 5.4: Dialog zur Kalibrierung der Videobilder.

In dem Dialog werden oben die beiden Kamerabilder angezeigt, die stets unter Verwendung der aktuellen Einstellungen erzeugt werden. In der ersten Zeile werden der Gerätename sowie die Auflösung und die Bildwiederholrate eingestellt. Darunter wird ein gewünschter Parametersatz ausgewählt. Dadurch ist es möglich, verschiedene Konfigurationen zu speichern, die sich beispielsweise für unterschiedliche Beleuchtungen eignen. Zwischen diesen Parametersätzen kann dann einfach umgeschaltet werden.

Mit den weiteren Reglern müssen die Kameras so konfiguriert werden, dass sie farblich möglichst identische Bilder liefern. Dies ist nötig, damit eine Farbsegmentierung in beiden Bildern zu ähnlichen Objekten führt. Eine entsprechende Kalibrierung wird im Rahmen dieser Diplomarbeit manuell durchgeführt.

Mit geeigneten automatischen Verfahren kann dieser Vorgang beschleunigt werden.

Die Bedienelemente für die einzelnen Parameter sind in der Regel selbsterklärend. Den größten Einfluss auf die Bildqualität haben die Einstellungen von Weißabgleich (White Balance) bis Verstärkung (Gain) in der oberen Hälfte der Regler. Den Weißabgleich und die Verstärkung kann die Webcam automatisch regeln. Diese Automatismen sind aber nur zur ersten Grobeinstellung zu verwenden. Bei dem Einsatz des Stereokamerasystems führen die automatischen Werte zu falsch segmentierten Objekten, da die Kameras auf wechselnde Beleuchtungen reagieren und die Farbbereiche (vergleiche Kapitel 5.2.2) nicht mehr dieselben Objekte repräsentieren.

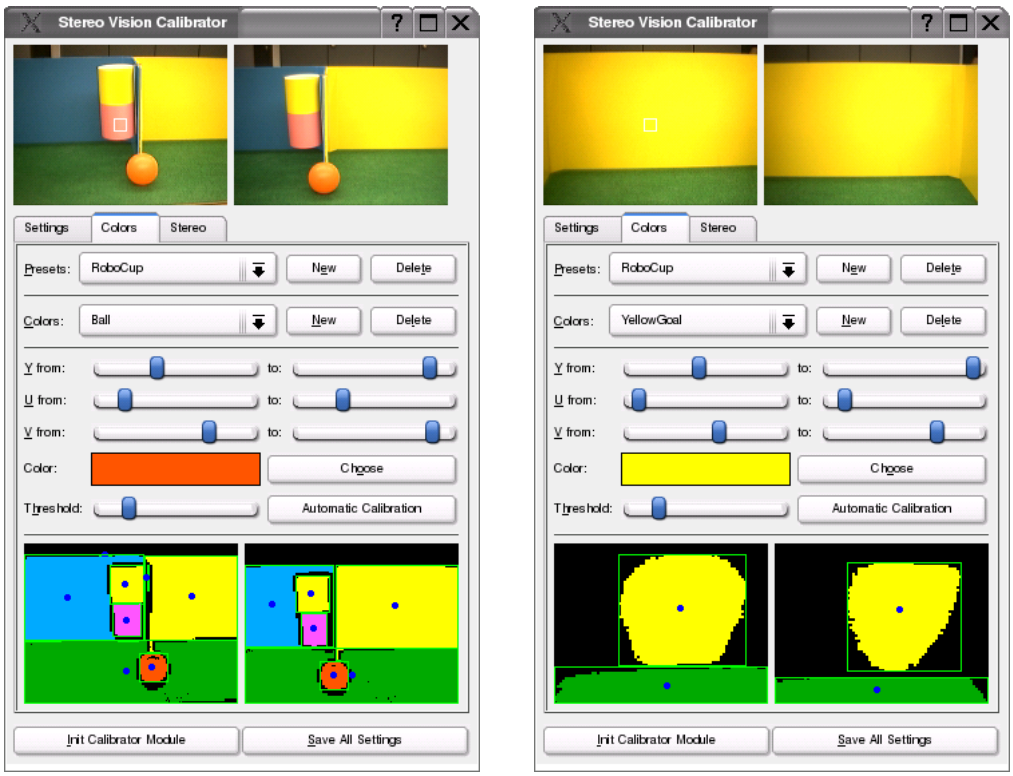
Die besten Ergebnisse werden ausschließlich mit künstlicher Beleuchtung erzielt, da sich beispielsweise beim Wechsel von Sonnenschein und Bewölkung die Farben in den Kamerabildern stark verändern. Die Kameraeinstellung „Tubes“ für den Weißabgleich, die speziell für die Beleuchtung mit Leuchtstoffröhren gedacht ist, erzeugt sehr ähnliche Kamerabilder. Die Intensität der Bilder muss anschließend mit den Reglern für Helligkeit (Brightness) und Verstärkung angeglichen werden.

Wie in Kapitel 4.1 beschrieben wird, ist es momentan nicht möglich, die Kameras eindeutig zu identifizieren. Falls die rechte und linke Webcam nach dem Anschließen vertauschte Gerätenamen zugewiesen bekommen, können diese mit der Schaltfläche in der Mitte zwischen den Einstellungen für die Kameras gewechselt werden.

5.2.2 Farbbereiche für Segmentierung

Der Algorithmus der Segmentierung wird in Kapitel 5.3 beschrieben. Für die Kalibrierung ist nur wichtig, dass das verwendete Verfahren zur Zuordnung von Pixeln zu einer bestimmten Farbklasse Quader im dreidimensionalen YUV-Farbraum benötigt. Der Dialog zur Kalibrierung dieser Quader ist in Abbildung 5.5 zu sehen.

Auch in diesem Dialog können verschiedene Parametersätze gespeichert werden, die zum Beispiel die Farben des *RoboCup* oder zur Erkennung von Türen in einem Gebäude enthalten. Dabei besteht jeder Satz aus bis zu 32 Farben, die zur Segmentierung verwendet werden. Im mittleren Bereich des Fensters befinden sich sechs Schieberegler, mit denen ein Quader im YUV-Raum über die minimalen und maximalen Werte des jeweiligen Farbkanals definiert wird. Darunter kann eine Farbe ausgewählt werden, mit der Objekte des aktuellen Farbbereichs dargestellt werden.



(a) (b)

Abbildung 5.5: Dialog zur Kalibrierung der Farbbereiche: (a) Objekte aus dem *RoboCup* (b) Farbfehler an den Bildrändern.

Im unteren Teil des Dialogs sind die segmentierten Versionen des Stereobildpaares sichtbar. Diese werden unter Verwendung des Algorithmus aus Kapitel 5.3 und der aktuellen Farbbereiche des aktiven Parametersatzes generiert. Außerdem werden die Bounding Boxen der einzelnen Blobs hellgrün und die Flächenschwerpunkte dunkelblau eingezeichnet.

Für jeden einzelnen Farbbereich kann ein Quader automatisch bestimmt werden. Hierzu ist in der Mitte des linken Kamerabilds ein kleines, weißes Quadrat eingezeichnet. Wird eine automatische Kalibrierung durchgeführt, so werden die Pixel in diesem Quadrat zur Bestimmung des Quaders verwendet. Zunächst bilden diese Pixel einen quadratischen Blob sowie die mittleren Farbwerte U_0 und V_0. Die Farbkanäle U und V der Pixel in der 4-Nachbarschaft des Blobs werden mit den mittleren Farbwerten verglichen. Wenn die Ungleichung

$$(U - U_0)^2 + (V - V_0)^2 < s \tag{5.1}$$

gilt, wird das Pixel zu dem Blob hinzugefügt und die mittleren Farbwerte aktualisiert. Der Skalar $s \in \mathbb{R}$ definiert einen Schwellwert und wird oberhalb der segmentierten Bilder im Dialog eingestellt. Diese und ähnliche Varianten einer Segmentierung mit Bereichswachstum sind in [43] zu finden. Der beschriebene Vorgang wird so lange durchgeführt, bis es in der 4-Nachbarschaft keine ähnlichen Pixel mehr gibt. Die YUV-Werte des so ermittelten Blobs definieren einen Quader, der die Farben des Blobs vollständig enthält.

In Abbildung 5.5(a) sind einige Objekte des *RoboCup* zu sehen, deren Farbbereiche zunächst automatisch bestimmt und dann manuell verbessert worden sind. Abbildung 5.5(b) zeigt eine Großaufnahme des gelben Tores und den entsprechenden Quader, wie er von dem automatischen Verfahren bestimmt worden ist. Man sieht hier sehr gut, dass die verwendeten Kameras die Farben am Bildrand teilweise stark verfälschen, obwohl die Torfläche gleichmäßig ausgeleuchtet ist. Dadurch wird nur der zentrale Bereich des Tores erkannt. Dieses Problem kann mit einer Vergrößerung der Farbquader umgangen werden. Damit ist es jedoch schwieriger, ähnliche Farben wie gelb und orange auseinander zu halten, da die Quader sich möglichst nicht im YUV-Raum überschneiden sollten. Liegt die Farbe eines Pixels in der Schnittmenge von zwei oder mehr Quadern, kann es nicht eindeutig einer Farbklasse zugeordnet werden.

5.2.3 Berechnung der Projektionsmatrizen

Es existieren viele verschiedene Verfahren, die aus unterschiedlichen Informationen eine Kamera oder ein Stereokamerasystem kalibrieren können. In diesem Abschnitt werden die Methoden aus [32, 64, 67] besprochen und ihre Probleme geschildert.

5.2.3.1 Verwendung einer Kalibrierungsebene

In dem Algorithmus von [67] wird die Kalibrierung einer Kamera durch die Aufnahme eines Kalibrierungsmusters in einer Ebene aus mindestens drei Orientierungen vorgeschlagen. Das Muster besteht dabei aus einem Raster schwarzer Quadrate auf weißem Grund. Die Koordinaten der Eckpunkte i der Quadrate sind in einem Koordinatensystem S_m in der Ebene bekannt. Dadurch sind die Punkte ${}^m\mathbf{p}_i = ({}^m p_{i\,x}; {}^m p_{i\,y}; 0)^T$ in euklidischen Koordinaten gegeben. Abbildung 5.6 zeigt eine Aufnahme des Kalibrierungsmusters mit eingezeichnetem Koordinatensystem. Die Quadrate haben eine Kantenlänge von $13{,}5\,mm$ und einen Abstand von $10\,mm$ voneinander.

Seien $n \geq 3$ Aufnahmen der Ebene aus verschiedenen Orientierungen gegeben und die Eckpunkte i der Quadrate in Aufnahme j in Pixelkoordinaten

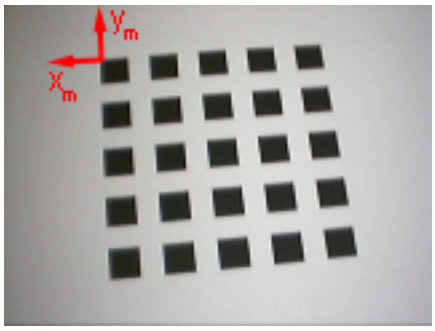


Abbildung 5.6: Kalibrierungsebene mit Koordinatensystem S_m mit Ursprung in der oberen linken Ecke.

${}^b\mathbf{p}_{ij} = ({}^bp_{ijx}; {}^bp_{ijy})^T$ bestimmt. Anschließend werden die Korrespondenzen ${}^b\mathbf{p}_{ij} \leftrightarrow {}^m\mathbf{p}_i$ definiert. Für jedes Bild wird eine Abbildung

$$ {}^b\hat{\mathbf{p}}_{ij} = H_j \begin{pmatrix} {}^mp_{ix} \\ {}^mp_{iy} \\ 1 \end{pmatrix} = H_j \, {}^m\hat{\mathbf{p}}_i \tag{5.2} $$

berechnet. Die Matrix $H_j \in \mathbb{R}^{3\times3}$ bildet die Punkte der Kalibrierungsebene auf Punkte der Bildebene j ab und wird Homografie genannt. Ist der Zusammenhang zwischen Muster- und Bildebene beschrieben durch $K\,(R_j; \mathbf{t}_j)$ mit einer Kameramatrix K aus (3.9), einer Rotationsmatrix R_j und einem Translationsvektor $\mathbf{t}_j$, so gilt $H_j = s\,K\,(\mathbf{r}_{j1}; \mathbf{r}_{j2}; \mathbf{t}_j)$. Dabei ist H_j eindeutig bis auf einen beliebigen Skalar $s \in \mathbb{R}$.

Eine lineare Näherung von H_j wird folgendermaßen berechnet. Der Vektor $\mathbf{x} = (\bar{\mathbf{h}}_{j1}; \bar{\mathbf{h}}_{j2}; \bar{\mathbf{h}}_{j3})^T \in \mathbb{R}^9$ enthält die Zeilen der Homografie H_j und für jede gegebene Korrespondenz wird ein Gleichungssystem

$$ \begin{pmatrix} {}^m\hat{\mathbf{p}}_i & \mathbf{0}^T & -{}^bp_{ijx}\,{}^m\hat{\mathbf{p}}_i \\ \mathbf{0}^T & {}^m\hat{\mathbf{p}}_i & -{}^bp_{ijy}\,{}^m\hat{\mathbf{p}}_i \end{pmatrix} \mathbf{x} = \mathbf{0} \tag{5.3} $$

angegeben, welches Gleichung (5.2) entspricht. Schreibt man diese Gleichungen für alle n_j Punkte untereinander, erhält man ein Gleichungssystem der Dimension $2\,n_j \times 9$, das mit der Singulärwertzerlegung (siehe Anhang G) gelöst wird.

Die Berechnung von H_j wird nun als nichtlineares Least-Squares-Problem

$$ \min_{H_j} \sum_i \left\| {}^b\mathbf{p}_{ij} - {}^b\tilde{\mathbf{p}}_{ij} \right\|^2 \tag{5.4} $$

formuliert und mit dem Levenberg-Marquardt-Algorithmus (Anhang H) gelöst. Dabei ist

$$ {}^b\tilde{\mathbf{p}}_{ij} = \frac{1}{\bar{\mathbf{h}}_3\,{}^m\hat{\mathbf{p}}_i} \begin{pmatrix} \bar{\mathbf{h}}_1\,{}^m\hat{\mathbf{p}}_i \\ \bar{\mathbf{h}}_2\,{}^m\hat{\mathbf{p}}_i \end{pmatrix} \tag{5.5} $$

die Projektion des Punktes ${}^m\mathbf{p}_i$ unter der Homografie H_j. Für die Initialisierung des Lösungsverfahrens wird die lineare Lösung des kombinierten Gleichungssystems aus (5.3) verwendet. Tabelle 5.1 zeigt die Fehler bei der Berechnung der Matrizen H_j bei der Aufnahme des Kalibrierungsmusters mit dem Stereokamerasystem bei zwei Testfällen. Da die mittleren Fehler der Rekonstruktion in der Regel unter einem Pixel liegen und die Eckpunkte ${}^b\mathbf{p}_{ij}$ nur mit Pixelgenauigkeit vorliegen, werden die Homografien mit hoher Genauigkeit berechnet.

j	H_{lj}		H_{rj}	
	Ecken	Fehler	Ecken	Fehler
1	43	$0{,}39\,px \pm 0{,}23\,px$	68	$0{,}44\,px \pm 0{,}18\,px$
2	58	$0{,}53\,px \pm 0{,}32\,px$	39	$0{,}37\,px \pm 0{,}18\,px$
3	58	$0{,}47\,px \pm 0{,}19\,px$	57	$0{,}37\,px \pm 0{,}19\,px$
4	58	$0{,}56\,px \pm 0{,}30\,px$	59	$0{,}39\,px \pm 0{,}19\,px$
5	70	$0{,}50\,px \pm 0{,}26\,px$	63	$0{,}52\,px \pm 0{,}24\,px$

(a)

j	H_{lj}		H_{rj}	
	Ecken	Fehler	Ecken	Fehler
1	50	$0{,}79\,px \pm 0{,}29\,px$	42	$0{,}92\,px \pm 0{,}29\,px$
2	62	$1{,}08\,px \pm 0{,}35\,px$	67	$1{,}15\,px \pm 0{,}49\,px$
3	59	$0{,}78\,px \pm 0{,}28\,px$	51	$1{,}05\,px \pm 0{,}38\,px$
4	67	$0{,}87\,px \pm 0{,}29\,px$	64	$0{,}91\,px \pm 0{,}32\,px$
5	73	$0{,}84\,px \pm 0{,}29\,px$	63	$0{,}98\,px \pm 0{,}38\,px$

(b)

Tabelle 5.1: Durchschnittlicher Fehler und Standardabweichung der Projektionen von den erkannten Ecken im Bild: (a) Fernsicht (b) Nahsicht.

5.2.3.2 Berechnung der Stereoparameter

Aus den Homografien wird nun eine lineare Berechnung der Kameramatrix K und der räumlichen Anordnung $(R_j; \mathbf{t}_j)$ illustriert. Sei $\omega = K^{-T} K^{-1} \in \mathbb{R}^{3\times3}$ eine symmetrische Matrix und das Bild des absoluten Kegelschnitts (Kapitel 8.5.1 in [14]). Der Vektor $\mathbf{x}_\omega = (\omega_{11}; \omega_{12}; \omega_{22}; \omega_{13}; \omega_{23}; \omega_{33})^T \in \mathbb{R}^6$ enthält die Einträge der oberen Dreiecksmatrix von ω und definiert die symmetrische Matrix damit eindeutig. Aus der Definition von H_j und der Orthonormalität

von $\mathbf{r}_{j1}$ und $\mathbf{r}_{j2}$ lässt sich für jede Aufnahme ein Gleichungssystem

$$\begin{pmatrix} \mathbf{v}_{12} \\ \mathbf{v}_{11} - \mathbf{v}_{22} \end{pmatrix} \mathbf{x}_\omega = \mathbf{0} \tag{5.6}$$

mit

$$\mathbf{h}_a^T \, \omega \, \mathbf{h}_b = \mathbf{v}_{a\,b} \, \mathbf{x}_\omega = \big(h_{1a} \, h_{1b}; h_{1a} \, h_{2b} + h_{2a} \, h_{1b}; h_{2a} \, h_{2b}; h_{3a} \, h_{1b} + h_{1a} \, h_{3b}; \\ h_{3a} \, h_{2b} + h_{2a} \, h_{3b}; h_{3a} \, h_{3b} \big) \, \mathbf{x}_\omega \tag{5.7}$$

ableiten. Dabei gilt $\mathbf{v}_{a\,b} \in \mathbb{R}^6$. Gleichung (5.6) wird zu einem Gleichungssystem der Dimension $2\,n \times 6$ kombiniert und mit der Singulärwertzerlegung gelöst. Es sind mindestens drei Aufnahmen der Ebene nötig, um das Gleichungssystem zu lösen.

Mithilfe einer Cholesky-Zerlegung, Matrix-Invertierung und Normierung (so dass $k_{33} = 1$ gilt) wird die Kameramatrix K aus ω berechnet. Eine geschlossene Form für diese drei Rechnungen ist in Anhang B von [67] gegeben. Anschließend wird die räumliche Anordnung

$$\begin{aligned} \mathbf{r}_{j1} &= s \, K^{-1} \, \mathbf{h}_{j1} \\ \mathbf{r}_{j2} &= s \, K^{-1} \, \mathbf{h}_{j2} \\ \mathbf{r}_{j3} &= \mathbf{r}_{j1} \times \mathbf{r}_{j2} \\ \mathbf{t}_j &= s \, K^{-1} \, \mathbf{h}_{j3} \end{aligned} \tag{5.8}$$

mit $s = \frac{1}{\|A^{-1} \, \mathbf{h}_{j1}\|}$ berechnet. Mit der Singulärwertzerlegung wird schließlich die Orthonormalität von $(\mathbf{r}_{j1}; \mathbf{r}_{j2}; \mathbf{r}_{j3})$ erzwungen, da sie numerisch im Allgemeinen nicht gegeben ist.

Das Originalverfahren wird nun auf das Stereokamerasystem erweitert. Es sind die Matrizen $H_{l\,j}$, $H_{r\,j}$, K_l, K_r, $R_{l\,j}$, $R_{r\,j}$, $\mathbf{t}_{l\,j}$ und $\mathbf{t}_{r\,j}$ der linken und rechten Kamera für jedes Stereobildpaar gegeben. Aus diesen Daten kann die Orientierung R_j und Position $\mathbf{t}_j$ der rechten relativ zur linken Kamera näherungsweise bestimmt werden. Mit dem Levenberg-Marquardt-Algorithmus wird jetzt die Optimierung des nichtlinearen Least-Squares-Problems

$$\min_{K_l, K_r, R_j, \mathbf{t}_j, R, \mathbf{t}} \sum_{j=1}^{n} \sum_{i=1}^{n_j} \left(\left\| {}^l\mathbf{p}_{i\,j} - {}^l\tilde{\mathbf{p}}_{i\,j} \right\|^2 + \left\| {}^r\mathbf{p}_{i\,j} - {}^r\tilde{\mathbf{p}}_{i\,j} \right\|^2 \right) \tag{5.9}$$

durchgeführt. Die Projektion eines Punktes i in Aufnahme j der linken bzw. rechten Kamera ist ${}^l\tilde{\mathbf{p}}_{i\,j}$ bzw. ${}^r\tilde{\mathbf{p}}_{i\,j}$. Als Startwert für die Optimierung werden die linearen Lösungen des kombinierten Gleichungssystems aus (5.6) von Gleichung (5.8) verwendet.

Ein Problem an dem beschriebenen Kalibrierungsverfahren ist die Berechnung der Kameramatrix K aus ω. Die Cholesky-Zerlegung ist nur für positiv

definite Matrizen durchführbar. Die Lösung für ω ist im Allgemeinen jedoch nicht positiv definit. In [67] wird dieses Problem nicht erwähnt und dementsprechend nicht gelöst.

Für positiv definite Matrizen $A \in \mathbb{R}^{k\times k}$ gilt, dass ihre Eigenwerte positiv sind. Eine alternative Bedingung ergibt sich aus dem charakteristischen Polynom $\det(A - t\,E_k) = (-1)^k\,t^k + a_{k-1}\,t^{k-1} + \ldots + a_1\,t + a_0$, bei der $(-1)^l\,a_l > 0$ für $0 \leq l \leq k-1$ gelten muss. Eine dritte mögliche Bedingung für positive Definitheit besagt, dass die oberen linken Untermatrizen $A_l \in \mathbb{R}^{l\times l}$ für $1 \leq l \leq k$ die Bedingung $\det(A_l) > 0$ erfüllen. Alle drei Formulierungen beinhalten nichtlineare Ungleichungen in den Einträgen von A und lassen sich nicht in einem linearen Gleichungssystem darstellen.

Die letzte der Bedingungen lässt sich im Fall der Matrix $\omega \in \mathbb{R}^{3\times 3}$ am leichtesten berechnen. Es werden Residuen $r_l \in \mathbb{R}$ für den Abstand von positiven Determinanten eingeführt.

$$r_l = \begin{cases} 0 & \det(\omega_l) > 10^{-5} \\ e^{\det(\omega_l)} - \det(\omega_l) & \text{sonst} \end{cases} \tag{5.10}$$

Eine nichtlineare Minimierung mit Gleichung (5.6) und den Residuen aus Gleichung (5.10) bringt nicht den gewünschten Erfolg.

Die beschriebene Kalibrierung wird mit den Homografien getestet, deren Fehler in Tabelle 5.1 gegeben sind. Die berechneten Matrizen ω sind bis auf ω_r der Nahsicht nicht positiv definit, die Berechnung der Kameramatrix K kann jedoch trotzdem ausgeführt werden, ohne dass zum Beispiel die Quadratwurzeln komplex werden. Für diesen Test werden die erwarteten Dimensionen der Stereogestelle aus Tabelle 4.2 als Startwert verwendet. Folgende Kalibrierungsergebnisse werden nach der nichtlinearen Optimierung von (5.9) für die Fernsicht

$$K_l = \begin{pmatrix} 369{,}87 & -33{,}6 & 153{,}92 \\ 0 & 309{,}83 & -25{,}88 \\ 0 & 0 & 1 \end{pmatrix}, \quad K_r = \begin{pmatrix} 300{,}21 & -20{,}4 & 23{,}996 \\ 0 & 273{,}8 & -60{,}74 \\ 0 & 0 & 1 \end{pmatrix},$$

$$R = \begin{pmatrix} 0{,}99 & 0{,}05 & 0{,}16 \\ -0{,}08 & 0{,}99 & 0{,}15 \\ -0{,}15 & -0{,}16 & 0{,}98 \end{pmatrix}, \qquad \mathbf{t} = \begin{pmatrix} 4{,}06 \\ 1{,}7 \\ 3{,}75 \end{pmatrix},$$

$$\tilde{\theta} = 13{,}12°, \qquad {}^l\mathbf{r} = (-0{,}16; 0{,}15; -0{,}07)^T$$

beziehungsweise für die Nahsicht

$$K_l = \begin{pmatrix} 284{,}14 & -19{,}53 & 49{,}04 \\ 0 & 267{,}9 & 24{,}7 \\ 0 & 0 & 1 \end{pmatrix}, \quad K_r = \begin{pmatrix} 241{,}06 & -30{,}33 & -26{,}02 \\ 0 & 245{,}27 & 41{,}47 \\ 0 & 0 & 1 \end{pmatrix},$$
$$R = \begin{pmatrix} 0{,}97 & 0{,}01 & 0{,}23 \\ 0{,}01 & 0{,}996 & -0{,}09 \\ -0{,}23 & 0{,}09 & 0{,}97 \end{pmatrix}, \qquad \mathbf{t} = \begin{pmatrix} -11{,}65 \\ -0{,}76 \\ -10{,}18 \end{pmatrix},$$
$$\tilde{\theta} = 14{,}47°, \qquad {}^{l}\mathbf{r} = (0{,}09; 0{,}23; 0{,}00054)^T$$

erzielt. Hier beschreibt R die Drehung von $\tilde{\theta}$ um den Vektor ${}^{l}\mathbf{r}$. Diese Ergebnisse haben offensichtlich nicht viel mit den erwarteten Werten in den Gleichungen (3.11) und (4.1) gemeinsam. Ein weiteres Anzeichen für die geringe Qualität der Ergebnisse ist, dass der Levenberg-Marquardt-Algorithmus nicht in einem (lokalen) Minimum abbricht, sondern aufgrund der maximalen Anzahl an Funktionsauswertungen.

5.2.3.3 Erweiterungen des Verfahrens

In [32, 64] wird das Verfahren von [67] auf zwei bzw. N Kameras erweitert. Diese Erweiterungen werden kurz erläutert.

Das Verfahren in [32] verwendet den statischen Zusammenhang der beiden Kameras, um eine Verknüpfung der Homografien $H_{l\,j}$ und $H_{r\,j}$ und damit die unendliche Homografie $H_\infty = K_r\, R\, K_l^{-1} \in \mathbb{R}^{3\times 3}$ (vergleiche Kapitel 13.4 in [14]) herzuleiten. Für jedes Stereobildpaar lässt sich ein Gleichungssystem

$$\begin{pmatrix} \mathbf{h}_{l\,j\,1}^T & \mathbf{0} & \mathbf{0} & 0 & \dots & -h_{r\,j\,11} & \dots & 0 \\ \mathbf{0} & \mathbf{h}_{l\,j\,1}^T & \mathbf{0} & 0 & \dots & -h_{r\,j\,21} & \dots & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{h}_{l\,j\,1}^T & 0 & \dots & -h_{r\,j\,31} & \dots & 0 \\ \mathbf{h}_{l\,j\,2}^T & \mathbf{0} & \mathbf{0} & 0 & \dots & -h_{r\,j\,12} & \dots & 0 \\ \mathbf{0} & \mathbf{h}_{l\,j\,2}^T & \mathbf{0} & 0 & \dots & -h_{r\,j\,22} & \dots & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{h}_{l\,j\,2}^T & 0 & \dots & -h_{r\,j\,32} & \dots & 0 \end{pmatrix} \mathbf{x} = \mathbf{0} \tag{5.11}$$

angeben, wobei die Einträge von $H_{r\,j}$ in der Spalte $9+j$ stehen und $\mathbf{x} = (\bar{\mathbf{h}}_{\infty 1}; \bar{\mathbf{h}}_{\infty 2}; \bar{\mathbf{h}}_{\infty 3}; s_1; \dots; s_n)^T \in \mathbb{R}^{9+n}$ die Zeilen von H_∞ und einen skalaren Faktor für jedes Stereobildpaar enthält. Aus $n \geq 2$ Stereobildpaaren wird durch Kombination von Gleichung (5.11) ein Gleichungssystem der Dimension $6\,n \times 9 + n$ erzeugt und mit der Singulärwertzerlegung gelöst.

Nun wird der Zusammenhang zwischen den Bildern der absoluten Kegel-

schnitte[1] $\omega_l = H_\infty^T \, \omega_r \, H_\infty$ als Gleichungssystem

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & \mathbf{v}_{11} \\ 0 & -1 & 0 & 0 & 0 & 0 & \mathbf{v}_{12} \\ 0 & 0 & -1 & 0 & 0 & 0 & \mathbf{v}_{22} \\ 0 & 0 & 0 & -1 & 0 & 0 & \mathbf{v}_{13} \\ 0 & 0 & 0 & 0 & -1 & 0 & \mathbf{v}_{23} \\ 0 & 0 & 0 & 0 & 0 & -1 & \mathbf{v}_{33} \end{pmatrix} \mathbf{x} = \mathbf{0} \tag{5.12}$$

mit $\mathbf{x} = (\mathbf{x}_{\omega\,l}^T; \mathbf{x}_{\omega\,r}^T)^T \in \mathbb{R}^{12}$ geschrieben. Die Vektoren $\mathbf{v}_{a\,b}$ aus Gleichung (5.7) enthalten hier die Einträge von H_∞. Kombiniert man (5.12) mit (5.6) zu einem Gleichungssystem der Dimension $4\,n + 6 \times 12$, so können ω_l und ω_r simultan berechnet werden.

Schließlich wird auch bei der Berechnung von $R = K_r^{-1} \, H_\infty \, K_l$ und $\mathbf{t}$ die feste Anordnung der Kameras ausgenutzt.

Bei [64] werden Matrizen $Q_j \in \mathbb{R}^{4\times 3}$ eingeführt, die einen Punkt ${}^m\hat{\mathbf{p}}_i = ({}^m p_{i\,x}; {}^m p_{i\,y}; 1)^T$ in das Koordinatensystem der linken Kamera transformieren. Damit sind die Abbildungen ${}^l\hat{\mathbf{p}}_{i\,j} = P_l \, Q_j \, {}^m\hat{\mathbf{p}}_i$ und ${}^r\hat{\mathbf{p}}_{i\,j} = P_r \, Q_j \, {}^m\hat{\mathbf{p}}_i$ gegeben.

Die Homografien $H_{r\,j}$ mit $j > 1$ werden gemäß dem Verfahren in Anhang B von [64] normalisiert, bevor alle Homografien in der Matrix

$$W = \begin{pmatrix} H_{l1} & \dots & H_{l\,n} \\ H_{r1} & \dots & H_{r\,n} \end{pmatrix} = \begin{pmatrix} \tilde{P}_l \\ \tilde{P}_r \end{pmatrix} (\tilde{Q}_1; \dots; \tilde{Q}_n) \in \mathbb{R}^{6\times 3\,n} \tag{5.13}$$

kombiniert werden. Da diese bei perfekten Homografien den Rang 4 besitzt, wird sie mit der Singulärwertzerlegung wie in Gleichung (5.13) zerlegt (vergleiche Abschnitt 3.4 in [61]). Bei den numerischen Homografien ist die Qualität der Zerlegung gegeben durch das Verhältnis $\frac{d_4}{d_5}$ der Singulärwerte von W. Je größer das Verhältnis, desto weniger ist W durch Rauschen beeinflusst. Die so erhaltenen Matrizen werden mit $S = (\tilde{P}_l^T (\tilde{P}_l \, \tilde{P}_l^T)^{-1}; \mathbf{s}) \in \mathbb{R}^{4\times 4}$ und $\mathbf{s} \in \ker(\tilde{P}_l)$ transformiert in

$$\tilde{P}_{l/r} = \tilde{P}_{l/r} \, S \qquad\qquad \tilde{Q}_j = S^{-1} \, \tilde{Q}_j, \tag{5.14}$$

so dass $\tilde{P}_l = (E_3; \mathbf{0})$ gilt. Der Zusammenhang der berechneten Matrizen zu den gewünschten Matrizen ist gegeben durch

$$P_{l/r} = t_{l/r} \, \tilde{P}_{l/r} \, U^{-1}, \qquad Q_j = t_j \, U \, \tilde{Q}_j, \qquad U = \begin{pmatrix} K_l^{-1} & \mathbf{0} \\ \mathbf{u}^T & u \end{pmatrix} \tag{5.15}$$

[1] Bei dieser Gleichung sind in [32] aufgrund der fehlerhaften Gleichung (18) die Indizes l und r vertauscht.

mit $t_{l/r}, t_j \in \mathbb{R}$. Die Matrix K_l wird wie bisher aus den Matrizen Q_j berechnet und die Komponenten $\mathbf{u} \in \mathbb{R}^3$ sowie $u \in \mathbb{R}$ von $U \in \mathbb{R}^{4\times4}$ werden durch Aufstellen eines Gleichungssystems und Lösen mittels der Singulärwertzerlegung berechnet.

Abschließend wird ein nichtlineares Least-Squares-Problem formuliert und mit dem Levenberg-Marquardt-Algorithmus minimiert.

Die beschriebenen Verfeinerungen des ursprünglichen Algorithmus durch [32, 64] benötigen nach wie vor die Berechnung der Matrix K_l mit einer Cholesky-Zerlegung von ω_l, ohne auf die positive Definitheit einzugehen. Die mit den erweiterten Verfahren erzielten Ergebnisse sind noch weit von den theoretischen Werten entfernt. In Kapitel 5.2.4 wird ein Verfahren mit einem völlig anderen Ansatz beschrieben, welches schließlich zu akzeptablen Resultaten führt.

5.2.4 Berechnung der Fundamentalmatrix

In [68, 69] und Kapitel 11 von [14] werden Verfahren vorgeschlagen, um ein Stereokamerasystem über die Berechnung der Fundamentalmatrix F zu kalibrieren. Diese werden im Folgenden beschrieben.

5.2.4.1 Verwendung einer Doppelebene

Der Nachteil dieser Verfahren ist, dass nicht alle Punkte in einer Ebene wie in Kapitel 5.2.3 liegen dürfen. Wenn dies der Fall ist, kann die Berechnung von D in Kapitel 5.2.4.3 nicht durchgeführt werden. Daher sind zwei Ebenen nötig, die rechtwinklig zueinander stehen sollten (im Folgenden Doppelebene genannt), damit die Koordinaten der Eckpunkte leichter bestimmt werden können. Abbildung 5.7 zeigt die Doppelebene mit dem Koordinatensystem S_m. Die Quadrate haben eine Kantenlänge von $85\,mm$ und einen Abstand von $25\,mm$ vom Rand. Dadurch sind die Quadrate auch aus großer Entfernung noch deutlich zu erkennen, was in dem endgültigen Kalibrierungsverfahren nützlich ist. Es können auch mehrere Quadrate auf einer größeren Doppelebene verwendet werden.

Nachfolgend sind ein Stereobildpaar der Doppelebene und n Korrespondenzen ${}^l\mathbf{p}_i \leftrightarrow {}^r\mathbf{p}_i \leftrightarrow {}^m\mathbf{p}_i$ mit $1 \leq i \leq n$ in den Bildebenen S_l bzw. S_r und der Doppelebene S_m gegeben. Die Koordinaten in der Doppelebene sind bekannt. In den Bildebenen werden die Korrespondenzen halbautomatisch ausgewählt. Hierzu wird im Kalibrierungsdialog von Abbildung 5.8 die aktuelle Position des Mauszeigers als erste Schätzung einer Ecke verwendet. Diese wird dann mithilfe des Gradienten einen quadratischen Bereichs um die Schätzung iterativ bis auf Subpixelgenauigkeit verbessert. Die verwendete Methode ist eine

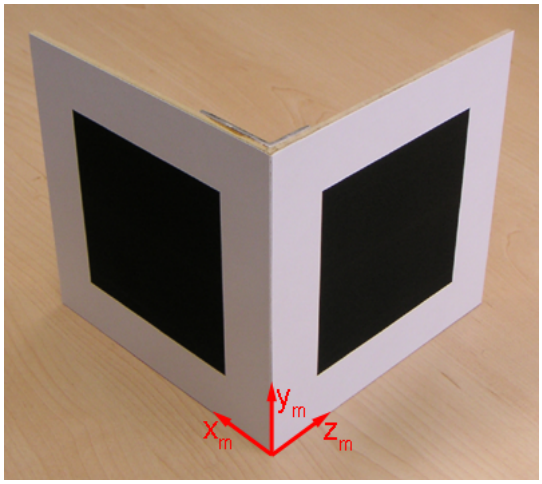


Abbildung 5.7: Zur Kalibrierung verwendete Doppelebene mit Koordinatensystem S_m mit Ursprung in der unteren vorderen Ecke.

Portierung der Matlab-Funktion `cornerfinder()` aus der Camera Calibration Toolbox for Matlab [5] in C++.

Auf der Webseite zu [68] wird eine Implementierung ohne Quelltext der automatischen Korrespondenzbestimmung aus Abschnitt 4 und 5 zum Download angeboten. Diese bricht jedoch mit einem Speicherzugriffsfehler ab. Der Algorithmus kann auch online verwendet werden. Hier zeigen Tests, dass die Ecken des Kalibrierungsmusters ein bis drei Pixel zu weit innen liegen und damit zu ungenau sind.

Damit die Fundamentalmatrix berechnet werden kann, muss zunächst Gleichung (3.18) umgeschrieben werden zu

$$({}^{r}p_x\,{}^{l}p_x; {}^{r}p_x\,{}^{l}p_y; {}^{r}p_x; {}^{r}p_y\,{}^{l}p_x; {}^{r}p_y\,{}^{l}p_y; {}^{r}p_y; {}^{l}p_x; {}^{l}p_y; 1)\,\mathbf{f} = \mathbf{0}, \tag{5.16}$$

wobei $\mathbf{f} = (\bar{\mathbf{f}}_1; \bar{\mathbf{f}}_2; \bar{\mathbf{f}}_3)^T \in \mathbb{R}^9$ die Zeilen der Fundamentalmatrix enthält. Mit $n \geq 8$ Punkten wird Gleichung (5.16) zu einem Gleichungssystem der Dimension $n \times 9$ kombiniert und mit der Singulärwertzerlegung gelöst. Anschließend wird gemäß Anhang G der Rang 2 von F erzwungen, da dies eine Eigenschaft der Fundamentalmatrix ist, wie in Kapitel 3.2 gezeigt wird.

Durch Parametrisierung von F mit $\mathbf{x} = (a; b; c; d; u_l; v_l; u_r; v_r)^T \in \mathbb{R}^8$ ergibt sich

$$F = \begin{pmatrix} b & a & -a\,v_l - b\,u_l \\ -d & -c & c\,v_l + d\,u_l \\ d\,v_r - b\,u_r & c\,v_r - a\,u_r & a\,v_l\,u_r + b\,u_l\,u_r - c\,v_l\,v_r - d\,u_l\,v_r \end{pmatrix}. \tag{5.17}$$

Der Vorteil dieser Darstellung ist, dass F garantiert den Rang 2 hat. In dieser Form kann jedoch der Fall aus Gleichung (3.22) nicht dargestellt werden. Es

zeigt sich aber, dass diese Konstellation nicht auftritt, da die Kameras und Gestelle nicht so exakt sind.

Aus der linearen Lösung $\mathbf{f}$ wird die Parametrisierung $\mathbf{x}$ berechnet und das nichtlineare Least-Squares-Problem

$$\min_{\mathbf{x}} \sum_{i=1}^{n} \left(\mathrm{dist}({}^{r}\hat{\mathbf{p}}_i, F\, {}^{l}\hat{\mathbf{p}}_i)^2 + \mathrm{dist}({}^{l}\hat{\mathbf{p}}_i, F^T\, {}^{r}\hat{\mathbf{p}}_i)^2 \right) \tag{5.18}$$

mit dem Levenberg-Marquardt-Algorithmus gelöst. Die Funktion

$$\mathrm{dist}(\hat{\mathbf{p}}, \mathbf{l}) = \frac{\left|\hat{\mathbf{p}}^T \mathbf{l}\right|}{\sqrt{l_x^2 + l_y^2}} \tag{5.19}$$

berechnet den kürzesten Abstand des Punktes $\mathbf{p}$ von der Geraden $\mathbf{l}$ in Pixeln.

5.2.4.2 Erweiterungen des Verfahrens

Zu der beschriebenen Berechnung von F werden noch drei Erweiterungen vorgestellt. In [68] geht man nicht von perfekten Korrespondenzen aus, sondern nimmt einen bestimmten Anteil Ausreißer an. Damit die Rechnung nicht von diesen beeinflusst wird, werden nicht alle n Korrespondenzen für die Berechnung von F verwendet. Stattdessen werden m Fundamentalmatrizen aus jeweils acht Korrespondenzen berechnet. Die Anzahl

$$m = \left\lceil \log_{1-(1-e)^8}(1-p) \right\rceil \tag{5.20}$$

ergibt sich aus dem Anteil e von Messfehlern und der Wahrscheinlichkeit p, dass bei m Versuchen mindestens einer ohne Ausreißer ist. Für $p = 0{,}99$ und $e = 0{,}4$ ergeben sich beispielsweise $m = 272$ Versuche. Die beste Fundamentalmatrix F_j mit $1 \leq j \leq m$ erfüllt die Bedingung

$$M = \min_{F_j} \operatorname*{med}_{i=1,\ldots,n} \left(\mathrm{dist}({}^{r}\hat{\mathbf{p}}_i, F_j\, {}^{l}\hat{\mathbf{p}}_i)^2 + \mathrm{dist}({}^{l}\hat{\mathbf{p}}_i, F_j^T\, {}^{r}\hat{\mathbf{p}}_i)^2 \right). \tag{5.21}$$

Die acht Korrespondenzen für jede Iteration werden zufällig, aber räumlich möglichst verteilt, gewählt. Details dieser Auswahl finden sich am Ende von Abschnitt 6.3 in [68]. Ein solches Auswahlverfahren wird auch Random Samples genannt.

Weiterhin wird ein Schwellwert auf der Basis von M vorgeschlagen, der bei einer abschließenden Optimierung des Problems (5.18) verwendet wird. Mithilfe dieses Schwellwertes wird in jeder Iteration geprüft, ob die Punkte einen zu großen Abstand von ihren Epipolarlinien haben und damit als Ausreißer herausgefiltert werden. Wird diese Rechnung als gewichtetes Least-Squares-Problem mit dem Levenberg-Marquardt-Algorithmus durchgeführt, so wird

das Minimum mit einer Fundamentalmatrix gefunden, bei der alle Korrespondenzen als Ausreißer gewertet werden und dadurch die Summe der quadrierten Fehler null ist. Eine solche Fundamentalmatrix ist für eine Rekonstruktion ungeeignet.

Die dritte Erweiterung ist eine Normalisierung der Punktkoordinaten ${}^{l}\mathbf{p}_i$ und ${}^{r}\mathbf{p}_i$ gemäß Kapitel 11.2 in [14], die für beide Bilder unabhängig durchgeführt wird.

Zunächst wird der Schwerpunkt

$$\mathbf{c} = \frac{1}{n} \sum_i \mathbf{p}_i \tag{5.22}$$

berechnet. Anschließend wird der Root-Mean-Square-Abstand (RMS)

$$d = \sqrt{\frac{1}{n} \sum_i \|\mathbf{c} - \mathbf{p}_i\|^2} \tag{5.23}$$

der Punkte vom Schwerpunkt ermittelt. Mit $s = \frac{\sqrt{2}}{d}$ wird die Transformation

$$T = \begin{pmatrix} s & 0 & -s\,c_x \\ 0 & s & -s\,c_y \\ 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 3} \tag{5.24}$$

verwendet, um die normalisierten Punkte $\tilde{\mathbf{p}}_i = T\,\hat{\mathbf{p}}_i$ zu bestimmen. Diese haben ihren Schwerpunkt im Ursprung und einen RMS-Abstand von $\sqrt{2}$ vom Ursprung. Durch die Normalisierung sind die numerischen Verfahren besser konditioniert.

Jetzt werden die Berechnungen in Gleichung (5.16) und (5.18) durchgeführt, um die normalisierte Fundamentalmatrix $\tilde{F}$ zu ermitteln, die schließlich $F = T_r^T\,\tilde{F}\,T_l$ definiert.

Wie in Kapitel 3.3.2 beschrieben wird, ist mit der Fundamentalmatrix nur eine schwache Kalibrierung des Stereokamerasystems gegeben und eine metrische Rekonstruktion nicht möglich. Der folgende Abschnitt illustriert, wie durch Informationen über die Doppelebene dennoch eine starke Kalibrierung durchführbar ist.

5.2.4.3 Starke Kalibrierung mit der Fundamentalmatrix

Aus der Fundamentalmatrix werden wie in Kapitel 3.3.2 zwei kanonische Projektionsmatrizen $\tilde{P}_l$ und $\tilde{P}_r$ berechnet. In [69] wird ein Verfahren vorgestellt, mit dem aus der Kenntnis der Punkte ${}^{m}\mathbf{p}_i$ metrische Projektionsmatrizen P_l und P_r bestimmt werden.

Aus den Korrespondenzen des Stereobildpaares ${}^{l}\mathbf{p}_i$ und ${}^{r}\mathbf{p}_i$ werden zunächst projektive Rekonstruktionen ${}^{m}\tilde{\mathbf{p}}_i = ({}^{m}\tilde{p}_{i\,x}; {}^{m}\tilde{p}_{i\,y}; {}^{m}\tilde{p}_{i\,z}; {}^{m}\tilde{p}_{i\,t})^T$ wie in Kapitel 3.3.1 bestimmt. Gesucht ist nun eine Matrix $D \in \mathbb{R}^{4\times4}$, welche die Gleichung

$$s_i \, {}^{m}\tilde{\mathbf{p}}_i = D \, {}^{m}\hat{\mathbf{p}}_i \tag{5.25}$$

erfüllt. Ist D bekannt, so werden $P_l = \tilde{P}_l \, D$ und $P_r = \tilde{P}_r \, D$ einfach berechnet.

Rechnet man den skalaren Faktor s_i aus (5.25) heraus, so ergeben sich die drei Bedingungen

$$\begin{aligned}
{}^{m}\tilde{p}_{i\,y} \, \bar{\mathbf{d}}_1 \, {}^{m}\hat{\mathbf{p}}_i^T - {}^{m}\tilde{p}_{i\,x} \, \bar{\mathbf{d}}_2 \, {}^{m}\hat{\mathbf{p}}_i^T &= 0 \\
{}^{m}\tilde{p}_{i\,z} \, \bar{\mathbf{d}}_1 \, {}^{m}\hat{\mathbf{p}}_i^T - {}^{m}\tilde{p}_{i\,x} \, \bar{\mathbf{d}}_3 \, {}^{m}\hat{\mathbf{p}}_i^T &= 0 \\
{}^{m}\tilde{p}_{i\,t} \, \bar{\mathbf{d}}_1 \, {}^{m}\hat{\mathbf{p}}_i^T - {}^{m}\tilde{p}_{i\,x} \, \bar{\mathbf{d}}_4 \, {}^{m}\hat{\mathbf{p}}_i^T &= 0,
\end{aligned} \tag{5.26}$$

welche als Gleichungssystem

$$\begin{pmatrix}
{}^{m}\tilde{p}_{i\,y} \, {}^{m}\hat{\mathbf{p}}_i^T & -{}^{m}\tilde{p}_{i\,x} \, {}^{m}\hat{\mathbf{p}}_i^T & \mathbf{0}^T & \mathbf{0}^T \\
{}^{m}\tilde{p}_{i\,z} \, {}^{m}\hat{\mathbf{p}}_i^T & \mathbf{0}^T & -{}^{m}\tilde{p}_{i\,x} \, {}^{m}\hat{\mathbf{p}}_i^T & \mathbf{0}^T \\
{}^{m}\tilde{p}_{i\,t} \, {}^{m}\hat{\mathbf{p}}_i^T & \mathbf{0}^T & \mathbf{0}^T & -{}^{m}\tilde{p}_{i\,x} \, {}^{m}\hat{\mathbf{p}}_i^T
\end{pmatrix} \mathbf{x} = \mathbf{0} \tag{5.27}$$

mit $\mathbf{x} = (\bar{\mathbf{d}}_1; \bar{\mathbf{d}}_2; \bar{\mathbf{d}}_3; \bar{\mathbf{d}}_4)^T \in \mathbb{R}^{16}$ geschrieben werden.

Kombiniert man dieses Gleichungssystem für $n \geq 5$ Punkte, die nicht in einer Ebene liegen, so erhält man ein Gleichungssystem der Dimension $3\,n \times 16$, welches mit der Singulärwertzerlegung gelöst wird.

Wie die Koordinaten im Stereobildpaar können die Punkte ${}^{m}\tilde{\mathbf{p}}_i$ mit $\tilde{T}_m$ und ${}^{m}\mathbf{p}_i$ mit T_m normalisiert werden, um $\tilde{D}$ zu berechnen. Wie in Gleichung (5.22) bzw. (5.23) werden $\mathbf{c}$ und d berechnet. Mit $s = \frac{\sqrt{3}}{d}$ sind die Transformationen gegeben mit

$$T = \begin{pmatrix}
s & 0 & 0 & -s\,c_x \\
0 & s & 0 & -s\,c_y \\
0 & 0 & s & -s\,c_z \\
0 & 0 & 0 & 1
\end{pmatrix} \in \mathbb{R}^{4\times4}. \tag{5.28}$$

Mit dieser Normalisierung befindet sich der Schwerpunkt im Ursprung und die Punkte haben einen RMS-Abstand vom Ursprung von $\sqrt{3}$. Damit ergibt sich die gesuchte Matrix aus $D = \tilde{T}_m^{-1} \, \tilde{D} \, T_m$.

Wird mit dem kalibrierten Stereokamerasystem eine Rekonstruktion berechnet, so ist diese im Koordinatensystem S_m der Doppelebene gegeben, in dem auch die Punkte ${}^{m}\mathbf{p}_i$ definiert sind. Die Rekonstruktion ist also eindeutig bis auf eine homogene Transformation. Ist die Orientierung und Position der Doppelebene in einem Referenzkoordinatensystem bekannt, wird die Transformation bestimmt und P_l sowie P_r entsprechend transformiert. Damit ist

schließlich eine absolute metrische Rekonstruktion im Referenzkoordinatensystem möglich.

Eine so durchgeführte Kalibrierung ermöglicht eine genaue Rekonstruktion von Objekten, die sich an einer ähnlichen Position zum Stereokamerasystem befinden, wie die Doppelebene während der Kalibrierung. In anderen Abständen treten große Fehler in der berechneten Position auf. Dieses Problem wird gelöst, indem mehrere Aufnahmen der Doppelebene in unterschiedlichen Entfernungen gemacht und zur Kalibrierung verwendet werden. Hierfür ist es hilfreich, große Quadrate in der Doppelebene zu haben, damit diese bei der geringen Auflösung auch in großen Entfernungen noch zur Eckenbestimmung verwendet werden können. Bei der Verwendung mehrerer Aufnahmen ist immer die räumliche Anordnung der Doppelebene anzugeben, da die Koordinaten vor der metrischen Kalibrierung in das gemeinsame Referenzkoordinatensystem transformiert werden.

In Kapitel 6.1.1 werden die Kalibrierungsergebnisse mit den beschriebenen Verfahren aufgeführt und bewertet.

Abbildung 5.8 zeigt die grafische Oberfläche, mit der ein Stereokamerasystem kalibriert werden kann, im Zustand nach einer erfolgreichen Kalibrierung. Im oberen Teil kann zwischen verschiedenen Kalibrierungen gewechselt werden, wie es beispielsweise für die Nah- und Fernsicht sinnvoll ist. Darunter befinden sich die Parameter der räumlichen Anordnung der Doppelebene relativ zum Referenzkoordinatensystem. In Anhang E ist ein Beispiel für die Bestimmung dieser Parameter gegeben. Es folgt ein Parameter, mit dem der quadratische Bereich für die Eckenerkennung eingestellt wird. Schließlich können die beschriebenen Erweiterungen der Kalibrierung wie Datennormalisierung und Random Samples aktiviert oder deaktiviert werden.

Im unteren Teil des Dialogs ist das Stereobildpaar mit den Rückprojektionen aller bekannten Eckpunkte als grüne Kreuze zu sehen. Bewegt man den Mauszeiger über die Bilder, so werden die Epipolarlinien zur aktuellen Zeigerposition rot eingezeichnet. Wenn die Kreuze auf den Eckpunkten liegen und beide Epipolarlinien durch die gleichen Punkte gehen, kann man schnell die Qualität der durchgeführten Kalibrierung einschätzen. Rechts sind schließlich die numerischen Ergebnisse aufgelistet. Diese bestehen aus der Fundamentalmatrix und den Projektionsmatrizen sowie den mittleren Fehlern und Standardabweichungen der Kalibrierung, die in Kapitel 6.1.1 näher beschrieben werden.

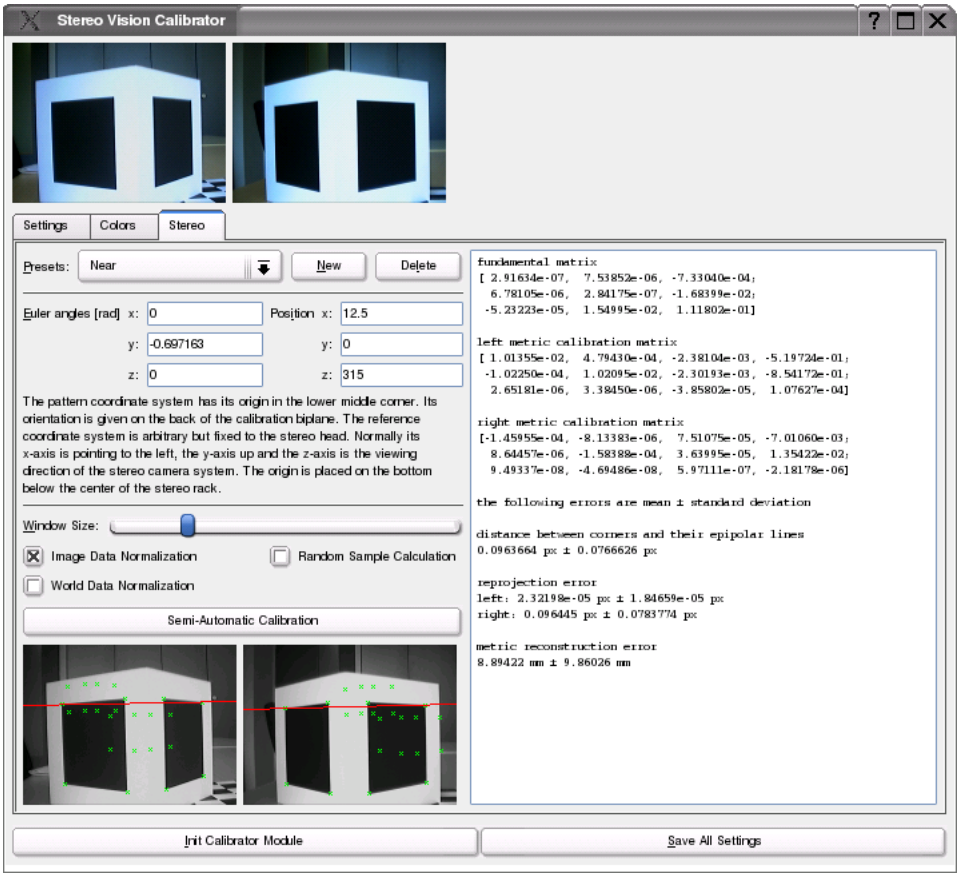


Abbildung 5.8: Dialog zur Kalibrierung des Stereokamerasystems.

5.3 Segmentierung

Der erste Schritt des Bildverstehens nach der Bilderzeugung besteht in dieser Diplomarbeit aus einer Segmentierung der Stereobildpaare. Diese wird aufgrund der Farbinformationen durchgeführt. In der Klasse `Segmentation` wird ein Interface zur Farbsegmentierung von Digitalbildern definiert. Klassen mit Implementierungen von neuen Algorithmen erben von dieser Basisklasse und werden mit der statischen Methode `build()` erzeugt. Die verschiedenen Algorithmen werden durch Namen unterschieden und können so über die verwendeten Konfigurationsdateien gewechselt werden, ohne das Programm neu zu kompilieren. Diese Implementierung wendet das bekannte Strategy Design Pattern [12] an. In der abgeleiteten Klasse `SegmentationVeloso` ist der Algorithmus zur Farbsegmentierung aus [6] implementiert. Im Folgenden werden der Algorithmus und einige Implementierungsdetails beschrieben. Zwei Beispiele für das Ergebnis dieses Verarbeitungsschrittes sind in Abbildung 6.2

zu sehen.

Das Verfahren wird nacheinander auf die beiden Bilder eines Paares angewendet und operiert direkt auf dem Bildformat YUV420P der Kameras. Dieses Format wird in Kapitel 5.1 erläutert. Durch die Verwendung des nativen Bildformats wird die Rechenzeit für eine Konvertierung eingespart und man kann die Komprimierung der U- und V-Kanäle ausnutzen.

Die Implementierung besteht aus vier Schleifen, die schrittweise einzelne Pixel zu Blobs zusammenfassen und einige Charakteristika berechnen. Diese Schritte werden in den nächsten Abschnitten illustriert.

5.3.1 Klassifizierung der Pixel

Der erste Durchlauf iteriert zeilenweise über die Pixel des Bildes. Dabei werden die Pixel klassifiziert und so den definierten Farbbereichen zugeordnet. Diese Bereiche sind als Quader im dreidimensionalen YUV-Farbraum definiert. Durch eine binäre Kodierung der Bereiche, die nur einmal durchgeführt werden muss, können die Pixel sehr effizient klassifiziert werden.

In dem Format YUV420P werden die drei Kanäle mit je einem Byte, also den Werten von 0 bis 255, abgespeichert. Für die Kodierung der Farbbereiche werden daher drei Arrays der Länge 256 erzeugt, die natürliche Zahlen mit 32 Bit enthalten. Die Bits 0 bis 31 entsprechen dabei jeweils einem Farbbereich und jedes der drei Arrays einem Farbkanal. Liegt ein möglicher Farbwert in einem Bereich, wird das entsprechende Bit auf 1 gesetzt und 0 sonst.

An einem Beispiel mit 2 Bit und möglichen Werten von 0 bis 3 wird dies verdeutlicht. Es sollen die beiden Farbbereiche F_0 mit $Y_0 = 3$, $U_0 \in \{1, 2, 3\}$, $V_0 = 1$ bzw. F_1 mit $Y_1 \in \{2, 3\}$, $U_1 \in \{0, 1\}$, $V_1 = 2$ in den Arrays Y, U und V kodiert werden. Tabelle 5.2 zeigt die Erstellung der drei Arrays.

	Y	U	V
Initialisierung	$\{00, 00, 00, 00\}$	$\{00, 00, 00, 00\}$	$\{00, 00, 00, 00\}$
mit F_0	$\{00, 00, 00, 01\}$	$\{00, 01, 01, 01\}$	$\{00, 01, 00, 00\}$
mit F_1	$\{00, 00, 10, 11\}$	$\{10, 11, 01, 01\}$	$\{00, 01, 10, 00\}$

Tabelle 5.2: Beispiel für binäre Kodierung von Farbbereichen.

Ein Pixel wird klassifiziert, indem anhand der Farbwerte jeweils ein Eintrag der Arrays ausgewählt wird, die daraufhin mit einem bitweisen *und* verknüpft werden. Das Ergebnis dieser Operation ist eine Zahl, welche die Zugehörigkeit des Pixels zu allen Farbbereichen enthält. Dadurch wird das Pixel mit nur zwei binären Operationen gleichzeitig in bis zu 32 Farbbereiche

klassifiziert, was mit einer einfachen Implementierung bis zu 192 Vergleiche benötigen würde. Wird beispielsweise ein Pixel $(3, 1, 2)$ untersucht, so wird die Rechnung

```
Y[3] & U[1] & V[2] = 11 & 11 & 10 = 10
```

ausgeführt. Das Ergebnis `10` besagt, dass das Pixel in dem Farbbereich F_1 aber nicht in F_0 liegt. Ist in dem Ergebnis kein Bit gesetzt, so liegt es in keinem der definierten Bereiche und wird nicht weiter verarbeitet.

Für die Segmentierung ist es problematisch, wenn sich zwei oder mehr Farbbereiche überlappen. In einem solchen Fall kann es vorkommen, dass in dem Ergebnis der Klassifizierung mehrere Bits gesetzt sind und das Pixel nicht eindeutig einem Bereich zugeordnet werden kann. Es wird dann das höchstwertige gesetzte Bit verwendet. Da die Farbbereiche in Konfigurationsdateien gespeichert sind und diese beim Start des Programmes in eine Map gelesen werden, liegen sie alphabetisch aufsteigend sortiert im Speicher und werden in dieser Reihenfolge in den Arrays kodiert. Dadurch wird im Fall einer Überlappung immer der Farbbereich mit dem lexikografisch größten Namen ausgewählt.

Die klassifizierten Pixel werden nun lauflängenkodiert (Run-Length Encoding, RLE [47]), damit die weitere Verarbeitung beschleunigt wird. Sobald sich die Klassifizierung aufeinanderfolgender Pixel ändert oder die Bildzeile beendet ist, wird das höchstwertige Bit der Klassifizierung bestimmt sowie ein Run erzeugt und gespeichert. Ein solcher Run ist durch seine Bildzeile y sowie die erste Spalte x_1 und die letzte Spalte x_2 gleichfarbiger, zusammenhängender Pixel charakterisiert. Darüber hinaus enthält er einen Zeiger auf einen Eltern-Run, der zunächst mit dem Run selbst initialisiert wird. Es werden nur Runs gespeichert, die einem Farbbereich entsprechen, wodurch der Hintergrund keinen Speicherplatz belegt.

5.3.2 Abstraktion zu Objekten

In der zweiten Schleife werden jeweils benachbarte Zeilen des RLE-Bildes verglichen. Überlappen sich in den Zeilen Runs gleicher Farbe, so wird der Eltern-Zeiger des unteren Runs auf Eltern-Run des oberen Runs gesetzt. Werden durch den Run in der unteren Zeile zwei Teilbereiche des oberen Bildteils vereinigt, so wird der Vorfahr des rechten Bereichs auf den Vorfahren des linken gesetzt. Ein Beispiel dieses Vorgangs ist in [6] aufgeführt.

In einem weiteren Durchlauf über die Runs werden die Eltern-Zeiger bis zur Wurzel verfolgt, um auf eine gemeinsame Datenstruktur zuzugreifen. Diese enthält unter anderem die Bounding Box $\mathbf{b} \in \mathbb{N}^4$, den Flächenschwerpunkt $\mathbf{c} \in \mathbb{N}^2$ und die Größe $s \in \mathbb{N}$ in Pixeln eines gleichfarbigen Bildbereichs, dem

so genannten Blob. Wenn dieser noch nicht existiert, wird er angelegt und mit initialen Werten

$$\begin{aligned} b_{left} &= w & c_x &= 0 \\ b_{right} &= 0 & c_y &= 0 \\ b_{top} &= y & s &= 0 \end{aligned}$$

gefüllt, wobei $w \in \mathbb{N}$ die Bildbreite ist. Für jeden Run werden die Charakteristika unter Verwendung der Run-Länge $l = x_2 - x_1 + 1$ aktualisiert.

$$\begin{aligned} b_{left} &\leftarrow \min(b_{left}, x_1) & c_x &\leftarrow c_x + \frac{l\,(x_1 + x_2)}{2} \\ b_{right} &\leftarrow \max(b_{right}, x_2) & c_y &\leftarrow c_y + y\,l \\ b_{bottom} &= y & s &\leftarrow s + l \end{aligned}$$

Insbesondere wird dadurch mit dem ersten Run eines Blobs b_{bottom} initialisiert.

In einer letzten Schleife wird über die erzeugten Blobs iteriert. Wenn sie eine gegebene Mindestgröße s_{min} nicht erreichen, werden sie gelöscht. Ist ein Blob groß genug, wird die Berechnung des Flächenschwerpunkts

$$\mathbf{c} \leftarrow \frac{1}{s}\mathbf{c}$$

abgeschlossen.

5.3.3 Optimierungen

Auf dem System des Humanoiden „Mr. DD“ sind einige Operationen besonders rechenintensiv, da dieser zum Beispiel nicht über eine Hardwareunterstützung für Fließkommazahlen verfügt. In Anhang B sind die Ergebnisse einiger Messungen zu finden, die zu den nachfolgenden Optimierungen führen.

Das Bildformat YUV420P ermöglicht die Wiederverwendung der Klassifizierung des U- und V-Kanals. Da ein UV-Wert zu einem Block von 2×2 Pixeln gehört, ändert sich nur der Anteil des Y-Kanals. In der vorliegenden Implementierung wird für zwei benachbarte Pixel einer Zeile die Klassifizierung des UV-Wertes zwischengespeichert. Man kann auch die Teilergebnisse für eine ganze Zeile speichern und in der nächsten abrufen. Allerdings führt der Aufwand durch die Array-Zugriffe nicht zu einer weiteren Verringerung der Laufzeit.

Bei der Berechnung des höchstwertigen Bits für die Klassifizierung der Pixel wird der Logarithmus zur Basis 2 durch eine neue Implementierung ersetzt.

Bei dem Verfahren wird in zwei Schritten durch bitweise Operationen entschieden, in welchem der vier Bytes sich das höchstwertige Bit befindet. Mit einer Lookup-Tabelle wird in dem entsprechenden Byte das gesuchte Bit identifiziert und mit der Information des betroffenen Bytes kombiniert.

Für die Speicherung der Runs und Blobs werden Listen der Standard Template Library (STL) eingesetzt. Bei der Verwendung von Threads, wie es in *RoboFrame* der Fall ist, sind die Listen sehr ineffizient, wenn Elemente eingefügt werden. Daher werden anstelle von Listen Vektoren verwendet, die nur wenig durch Threads beeinflusst werden. Wird jedoch ein Element zu einem Vektor hinzugefügt, sind die Iteratoren unter Umständen nicht mehr gültig, was zum Beispiel für die Eltern-Zeiger der Runs wichtig ist. Aus diesem Grund wird zu Beginn Speicher für die Vektoren reserviert, so dass sie sich nicht neu allokieren müssen. Durch die Reservierung wird Laufzeit eingespart, aber es kann nur noch eine begrenzte Zahl von Runs pro Zeile und Blobs pro Bild gespeichert werden.

Durch diese und einige kleinere Optimierungen wird die Segmentierung komplexer Bilder auf dem humanoiden Roboter bis zu 100-mal schneller durchgeführt. In Kapitel 6.2 werden Beispiele der Segmentierung gezeigt sowie Ergebnisse der abschließenden Messungen von Laufzeit und Speicherverbrauch aufgeführt.

5.4 Korrespondenzbestimmung

In diesem Abschnitt werden verschiedene Verfahren vorgestellt, mit denen das Korrespondenzproblem in einem Stereobildpaar gelöst wird. Dadurch kennt man Paare von Pixelkoordinaten, die jeweils den gleichen Punkt im Raum beschreiben. Mit diesen Paaren werden anschließend gemäß Kapitel 3.3.1 die dreidimensionalen Positionen rekonstruiert.

Die Methoden zur Berechnung der Korrespondenzen kann man in flächen- und merkmalsbasierte Algorithmen einteilen. Beide Gruppen werden in Kapitel 3.4.3 näher charakterisiert. Vertreter beider Kategorien werden in den folgenden Abschnitten beschrieben.

5.4.1 Flächenbasierte Methode

Ähnlich der Farbsegmentierung wird auch hier das Strategy Design Pattern [12] angewendet. Im Rahmen dieser Diplomarbeit ist eine Basisklasse `DisparityArea` entwickelt worden, die ein Interface für flächenbasierte Methoden definiert, wodurch mehrere Algorithmen einfach ausgetauscht werden können. Die flächenbasierte Methode aus [37] wird in der Klasse

`DisparityMaenner` realisiert und im Folgenden illustriert. In den Abbildungen 6.3 und 6.4 sind Beispiele für die Ausgabe des Verfahrens ersichtlich.

Das verwendete Verfahren kann nur gleichgerichtete Stereobildpaare verarbeiten, deren Zusammenhang durch die Fundamentalmatrix aus Gleichung (3.22) charakterisiert wird. Dadurch besitzen die Pixel einer Korrespondenz dieselbe y-Koordinate.

Betrachtet man ein Pixel $(x; y)$ im rechten Bild, so kann aufgrund der Vorbedingung des Verfahrens das korrespondierende Pixel nur die Koordinaten $(x + d; y)$ besitzen. Dabei ist $d \in \mathbb{Z}$ die Disparität. Die Laufzeit des Verfahrens hängt stark von dem zu durchsuchenden Bereich $d_{min} \leq d \leq d_{max}$ ab. Im Fall paralleler Kameras mit $\theta = 0$ erscheinen Objekte im linken Bild immer weiter rechts als im rechten Bild. Daher kann $d_{min} \geq 0$ gesetzt werden. Für den Fall $\theta \neq 0$ bedeutet diese Einschränkung, dass nur Objekte in einer Entfernung bis zum Schnittpunkt der optischen Achsen detektiert werden können. In größeren Abständen befinden sich die Objekte im linken Bild weiter links als im rechten. Der Wert d_{max} ist maximal die Bildbreite und kann verringert werden, wenn Objekte ab einer bestimmten Entfernung detektiert werden sollen.

Durch die beschränkten Informationen können zu einzelnen Pixeln keine Korrespondenzen bestimmt werden. Daher werden quadratische Bildbereiche der ungeraden Kantenlänge $s \in \mathbb{N}$ um die Pixel betrachtet, um Paarungen zu berechnen. Zu einem Pixel $(x; y)$ im rechten Bild und den Pixeln $(x + d; y)$ im linken Bild werden Ähnlichkeiten der entsprechenden Bereiche berechnet, deren Minimum die Korrespondenz definiert. Als Ähnlichkeitsmaß wird die Summe absoluter Differenzen (SAD) verwendet, da sie effizient berechenbar ist.

$$\begin{aligned} SAD(x, y, d) = \sum_{i=-\frac{s-1}{2}}^{\frac{s-1}{2}} \sum_{j=-\frac{s-1}{2}}^{\frac{s-1}{2}} & (|Y_r(x+i, y+j) - Y_l(x+i+d, y+j)| \\ & + |U_r(x+i, y+j) - U_l(x+i+d, y+j)| \\ & + |V_r(x+i, y+j) - V_l(x+i+d, y+j)|) \end{aligned} \tag{5.29}$$

Die Werte $Y_{l/r}, U_{l/r}, V_{l/r}$ sind die YUV-Farbkanäle des linken und rechten Bildes. Teile dieser Summe werden in benachbarten Bildbereichen mehrfach benötigt, was für eine effiziente, inkrementelle Berechnung ausgenutzt wird.

Hierfür ist bei einer Bildgröße von $w \times h$ Pixeln jedoch ein dreidimensionales Speichervolumen der Dimension $w \times h \times (d_{max} - d_{min} + 1)$ nötig. Unter den verschiedenen Möglichkeiten, dieses Volumen zu füllen, wird diejenige gewählt, bei welcher der Cache des Systems optimal ausgenutzt wird. Daher wird zu einem Pixel im rechten Bild die Ähnlichkeit zu allen Pixeln im linken

Bild mit den möglichen Werten d bestimmt. Die Pixel im rechten Bild werden zeilenweise bearbeitet. Dadurch ist es außerdem möglich, mithilfe eines Ringpuffers das Speichervolumen auf die Größe $w \times (s+2) \times (d_{max} - d_{min} + 1)$ zu reduzieren. Um Sonderbehandlungen an den Rändern zu vermeiden, wird ein schmaler Rand zu dem Volumen hinzugefügt und mit 0 initialisiert. Weitere Details zu der Berechnung der SAD-Werte sind [37] zu entnehmen.

Da zu einem rechten Pixel $(x; y)$ zunächst alle Disparitäten betrachtet werden, können gleichzeitig die Disparitäten der drei geringsten SAD-Werte gespeichert werden. Diese werden für einen Eindeutigkeitstest verwendet, weil in größeren einfarbigen Bereichen oder regelmäßig gemusterten Gebieten mehrere Korrespondenzen möglich sind. Der kleinste der drei Werte definiert einen Schwellwert, unter dem maximal ein weiterer Wert liegen darf. Ein solches doppeltes Minimum tritt häufig auf, da sich zwei benachbarte Suchfenster nur wenig unterscheiden. Ist das Minimum eindeutig und das korrespondierende Pixel im linken Bild liegt nicht am Bildrand, so wird es als Rechts-Links-Disparität d_{rl} an der Stelle $(x; y)$ eines neuen Bildes gespeichert.

Wenn die Berechnungen für eine Zeile beendet sind, werden die gefundenen Disparitäten einer Konsistenzprüfung unterzogen. Die Disparität vom rechten in das linke Bild muss identisch sein mit der Disparität der umgekehrten Richtung.

$$d_{rl}(x, y) = -d_{lr}(x + d_{rl}(x, y), y) \tag{5.30}$$

Für diese Prüfung werden die bereits berechneten SAD-Werte wiederverwendet. Es muss zu einem Pixel $(x; y)$ im linken Bild nur das Minimum der Werte $(x - i; y; i)$ im Speichervolumen gesucht werden. Entspricht das gefundene Minimum einer anderen Disparität, so wird das Pixel im Ergebnis als ungültig markiert.

Das Resultat dieser Berechnungen ist ein Graustufenbild, bei dem an der Stelle $(x; y)$ die Disparität $d_{rl}(x, y)$ gespeichert ist. Für die Positionsberechnung wird also das Koordinatenpaar ${}^l\mathbf{p} = (x + d_{rl}(x, y); y)^T$ und ${}^r\mathbf{p} = (x; y)^T$ verwendet. Resultate des beschriebenen Verfahrens werden in Kapitel 6.3.1 aufgeführt und bewertet.

5.4.2 Merkmalsbasierte Methoden

Auch hier wird das Strategy Design Pattern [12] angewendet. Von der Basisklasse `DisparityFeature` sind die beiden Klassen `DisparitySize` und `DisparityCentroid` abgeleitet, die zwei einfache, aber effiziente Methoden für eine merkmalsbasierte Positionsberechnung zur Verfügung stellen. Die entsprechenden Korrespondenzbestimmungen sind in dieser Diplomarbeit

neu entwickelt worden und bauen auf einem Stereobildpaar auf, dass beispielsweise mit dem Verfahren aus Kapitel 5.3 segmentiert wird. In Abbildung 6.6 werden beispielhafte Ergebnisse beider Methoden dargestellt.

In der Klasse `DisparitySize` werden die Korrespondenzen aufgrund der Farbe und der Größe der segmentierten Blobs bestimmt. Hierfür werden die Blobs der beiden Bilder zunächst aufsteigend nach ihrer Farbe und innerhalb einer Farbe absteigend nach der Größe sortiert. Anschließend iteriert man parallel über beide Blob-Listen. Gibt es Blobs einer Farbe nur im linken oder im rechten Bild, so kann es keine Korrespondenz geben. In diesem Fall iteriert man in der Liste mit dem lexikografisch kleineren Farbnamen weiter bis zum nächsten Blob einer anderen Farbe. Besitzen die aktuellen Blobs der beiden Listen denselben Namen, so handelt es sich um den jeweils größten unbearbeiteten Blob dieser Farbe. Mit den Flächenschwerpunkten dieser Blobs wird ein homogenes Gleichungssystem (3.24) aufgestellt und mit der Singulärwertzerlegung gelöst. Die berechnete Position wird auf Millimeter genau gerundet und in den beiden Blobs gespeichert. Danach wird in den Listen zum nächsten Blob gewechselt.

Der Algorithmus der Klasse `DisparityCentroid` verwendet ein etwas aufwändigeres Verfahren zur Korrespondenzberechnung. Wie im vorangegangenen Abschnitt werden die beiden Blob-Listen aufsteigend nach den Farbnamen sortiert. Im Gegensatz zu `DisparitySize` werden hier alle möglichen Paarungen von Blobs gleicher Farbe betrachtet. Mit der Funktion (5.19) wird für jede Paarung der Abstand der Flächenschwerpunkte ${}^{l/r}\mathbf{p}$ von ihren Epipolarlinien berechnet.

$$d_l = \mathrm{dist}({}^{l}\hat{\mathbf{p}}, F^T\,{}^{r}\hat{\mathbf{p}}) \qquad\qquad d_r = \mathrm{dist}({}^{r}\hat{\mathbf{p}}, F\,{}^{l}\hat{\mathbf{p}}) \tag{5.31}$$

Anschließend werden die Korrespondenzen ausgewählt, für welche die Abstände unter einem Schwellwert liegen und deren Abstandssumme minimal ist:

$$\min_{{}^{l/r}\mathbf{p}}(d_l + d_r). \tag{5.32}$$

Für die Berechnung des Minimums werden jeweils nur die Paarungen benutzt, deren Blobs noch nicht in einer Korrespondenz verwendet werden. Auch hier wird die Rekonstruktion wie in Kapitel 3.3.1 durchgeführt und auf Millimeter genau gerundet.

Bei dieser Variante wird die Positionsberechnung nicht durch größere Störungen beeinträchtigt, die nur in einem der beiden Bilder sichtbar und von den Epipolarlinien gleichfarbiger Objekte entfernt sind. Ein solcher Fall ist zum Beispiel gegeben, wenn im Zuschauerraum orangene Kleidung getragen wird. Diese Objekte erscheinen im Allgemeinen im oberen Teil der Bilder und haben somit einen großen Abstand zu den Epipolarlinien der möglichen Bälle

im unteren Bildbereich. Von Nachteil sind allerdings Störungen, die näher an einer Epipolarlinie liegen, als das tatsächlich korrespondierende Objekt.

Beide Verfahren ermitteln unter Umständen falsche Korrespondenzen, wenn Objekte in einem Bild verdeckt oder außerhalb des Bildes sind. Hierfür sind aufwändigere Methoden zu untersuchen.

Bei einer guten Segmentierung des Stereobildpaares ohne viele Störungen sind mit den beiden Verfahren nur wenige Informationen der Bilder zu verarbeiten. Dadurch ist es möglich, die kompletten Bilder nach Korrespondenzen abzusuchen. Somit ist für das Gestell der Nahsicht mit $\theta \neq 0$ nicht die Einschränkung des flächenbasierten Verfahrens gegeben, bei dem nur Objekte bis zum Schnittpunkt der optischen Achsen detektiert werden.

Zur Compile-Zeit kann durch Definition des Symbols `USE_SAMPSON` für beide Verfahren zusätzlich die Sampson-Näherung aus Kapitel 12.4 von [14] aktiviert werden. Diese berechnet zu den Pixelkoordinaten ${}^{l/r}\mathbf{p}$ einer Korrespondenz zunächst Koordinaten ${}^{l/r}\tilde{\mathbf{p}}$, die näherungsweise die Epipolarbedingung (3.18) erfüllen und verringert so die Störungen der verwendeten Punkte.

$$\mathbf{p} = \begin{pmatrix} ({}^{r}\mathbf{p}^T F)_x \\ ({}^{r}\mathbf{p}^T F)_y \\ (F\,{}^{l}\mathbf{p})_x \\ (F\,{}^{l}\mathbf{p})_y \end{pmatrix} \qquad \begin{pmatrix} {}^{l}\tilde{p}_x \\ {}^{l}\tilde{p}_y \\ {}^{r}\tilde{p}_x \\ {}^{r}\tilde{p}_y \end{pmatrix} = \begin{pmatrix} {}^{l}p_x \\ {}^{l}p_y \\ {}^{r}p_x \\ {}^{r}p_y \end{pmatrix} - \frac{{}^{r}\mathbf{p}^T F\,{}^{l}\mathbf{p}}{\mathbf{p}^T\mathbf{p}}\mathbf{p} \tag{5.33}$$

Anschließend wird mit den Koordinaten ${}^{l/r}\tilde{\mathbf{p}}$ wie bisher die Rekonstruktion durchgeführt. Der Nutzen dieses Verfahrens wird in Kapitel 6.3.2 diskutiert.

5.4.3 Optimierungen

Wie bei der Segmentierung in Kapitel 5.3.3 können die merkmalsbasierten Verfahren zur Positionsberechnung auf dem humanoiden Roboter beschleunigt werden. Anhang B enthält Messungen einiger elementarer Operationen, anhand derer die Positionsberechnung optimiert wird.

Für die Matrix des homogenen Gleichungssystems (3.24) werden zu Beginn des Programms zwei Matrizen $A_{var}, A_{const} \in \mathbb{R}^{4\times4}$ erzeugt, wobei A_{var} bei der Rekonstruktion durch die Pixelkoordinaten verändert wird und A_{const} unveränderlich ist. Dadurch werden die Array-Zugriffe bei der Konstruktion der Matrix reduziert. Weiterhin werden Zwischenergebnisse wiederverwendet. So unterscheiden sich bei den Abständen $d_{l/r}$ nur die Nenner, so dass man diese Abstände explizit schneller berechnen kann, als mit der generischen Funktion (5.19).

Der größte Teil des Zeitbedarfs für die Rekonstruktion wird allerdings durch die Singulärwertzerlegung verbraucht. Es wird in dieser Diplomarbeit eine

schnellere Variante entwickelt, da für die Lösung des homogenen Gleichungssystems nur der Rechtssingulärvektor zu dem kleinsten Singulärwert benötigt wird. Daher wird nicht die vollständige Singulärwertzerlegung $A = U\,D\,V^T$ ermittelt. So wird in der optimierten Version die Matrix U nicht berechnet und die Singulärwerte werden nicht in absteigender Reihenfolge sortiert. Es wird abschließend nur der kleinste Singulärwert gesucht und der entsprechende Rechtssingulärvektor von der Methode zurückgegeben.

Ein zeitaufwändiger Bestandteil der Singulärwertzerlegung ist die Anwendung der Wurzelfunktion, um den Satz des Pythagoras $\sqrt{a^2 + b^2}$ zu berechnen. Dieser ist so implementiert, dass kein auslöschender Überlauf und Unterlauf auftritt. Dadurch sind Wurzeln von Zahlen c mit $1 \leq c \leq 2$ zu berechnen. Wurzeln dieser Art können mit Ganzzahloperationen angenähert werden. Wenn $c = 2$ gilt, wird eine entsprechende Konstante zurückgegeben, da dieser Fall von dem folgenden Algorithmus nicht verarbeitet werden kann.

In dem IEEE-Standard 754 [18] wird die binäre Darstellung von Fließkommazahlen definiert. Die Zahlen $1 \leq c < 2$ besitzen alle den gleichen Exponenten und ihre Dezimalstellen sind bei einer Darstellung mit 64 Bit in den Bits 51 bis 0 gespeichert. Die Bits der Fließkommazahl werden als Ganzzahl gleicher Bitlänge interpretiert und so verschoben, dass die Bits der Dezimalstellen in den Bits 61 bis 10 stehen und so die binäre Darstellung `01xxxx...` erzeugen. Die so erhaltene Zahl sei i und besitzt 64 Bit. Die Wurzel dieser Zahl hat genau 32 Bit. Es werden die 32 Bit des Ergebnisses r mit `1000...` initialisiert. In einer Schleife werden nacheinander die Bits 30 bis 0 gesetzt. In jedem Durchlauf wird geprüft, ob $r^2 > i$ gilt. Wenn dies der Fall ist, so wird das eben gesetzte Bit wieder auf 0 gesetzt. Nach der Schleife wird r mit binären Operationen wieder in eine gültige Fließkommazahl umgewandelt. Da die verwendeten Fließkommazahlen eine Genauigkeit von 52 Bit haben, ist das Ergebnis mit 32 Bit nur eine Näherung und auf 10 Dezimalstellen genau.

Weitere Optimierungen sind bei der verwendeten Implementierung der Singulärwertzerlegung nur schwer möglich, da die Wurzelberechnungen und die schreibenden Array-Zugriffe nicht ohne weiteres reduziert werden können. Eine alternative Möglichkeit ein lineares, inkonsistentes, homogenes Gleichungssystem im Least-Squares-Sinn zu lösen, ist die Berechnung des Eigenvektors zum kleinsten Eigenwert. Aber auch diese Methode ist komplex und eine Beschleunigung ist nicht ohne umfangreiche Implementierung und Tests möglich. Beispiele sowie Zeit- und Speicherbedarf der beiden merkmalsbasierten Algorithmen sind in Kapitel 6.3.2 zu finden.

Kapitel 6

Ergebnisse

6.1 Kalibrierung

6.1.1 Implementierte Algorithmen

In diesem Abschnitt werden die numerischen Kalibrierungsergebnisse der Verfahren von Kapitel 5.2.4 aufgelistet und bewertet. Für die Tests werden je drei Stereobildpaare für die Fern- bzw. Nahsicht verwendet. Abbildung 6.1 zeigt die benutzten Testbilder.

Die räumlichen Anordnungen der Doppelebene bestehen in allen sechs Fällen aus einer Drehung θ_i um die y-Achse und einer Verschiebung ${}^0\mathbf{r}_{mi}$ in Millimetern. Die folgenden Werte werden für die Kalibrierung der Fernsicht

$$\theta_1 = -0{,}67 \qquad \theta_2 = -0{,}88 \qquad \theta_3 = -0{,}78$$

$${}^0\mathbf{r}_{m1} = \begin{pmatrix} 16 \\ 0 \\ 471{,}5 \end{pmatrix} \qquad {}^0\mathbf{r}_{m2} = \begin{pmatrix} -100 \\ 108 \\ 1002{,}5 \end{pmatrix} \qquad {}^0\mathbf{r}_{m3} = \begin{pmatrix} 95 \\ 0 \\ 1074{,}5 \end{pmatrix}$$

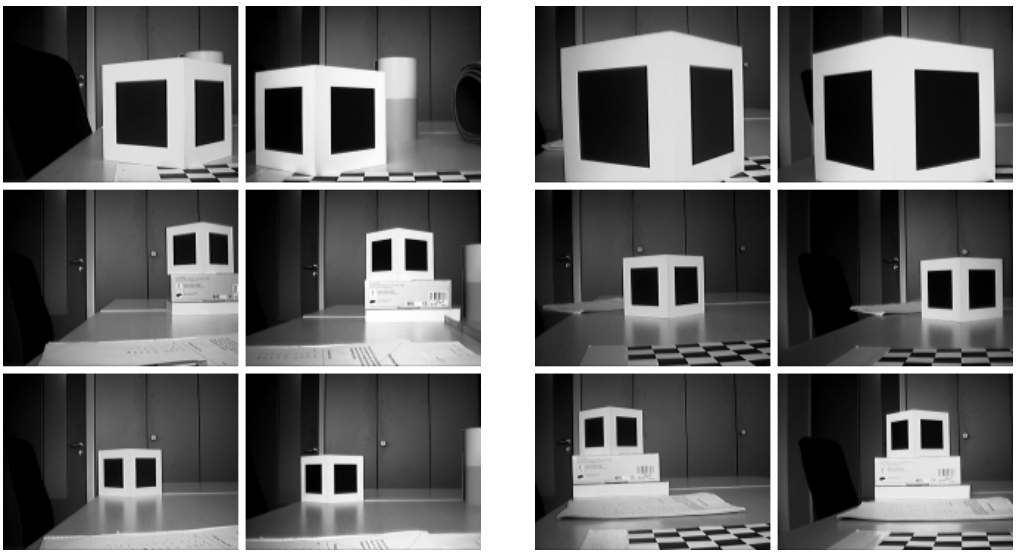
beziehungsweise der Nahsicht

$$\theta_1 = -0{,}70 \qquad \theta_2 = -0{,}80 \qquad \theta_3 = -0{,}77$$

$${}^0\mathbf{r}_{m1} = \begin{pmatrix} 12{,}5 \\ 0 \\ 315 \end{pmatrix} \qquad {}^0\mathbf{r}_{m2} = \begin{pmatrix} -37 \\ 0 \\ 818{,}5 \end{pmatrix} \qquad {}^0\mathbf{r}_{m3} = \begin{pmatrix} 94 \\ 108 \\ 1094{,}5 \end{pmatrix}$$

verwendet. In Anhang E ist ein Beispiel gegeben, wie diese Werte mit hinreichender Genauigkeit ermittelt werden können.

In den Tabellen 6.1 und 6.2 sind die mittleren Fehler und die Standardabweichungen der Kalibrierung aufgeführt. Die erste Spalte bezeichnet eine der folgenden Kalibrierungsvarianten.



(a) (b)

Abbildung 6.1: Stereobildpaare zum Testen der Kalibrierung: (a) Fernsicht (b) Nahsicht.

1. Kalibrierung ohne Datennormalisierung
2. Kalibrierung mit Normalisierung der Bilddaten
3. Kalibrierung mit Normalisierung der Weltdaten
4. Kalibrierung mit vollständiger Datennormalisierung

Die zweite Spalte enthält den Abstand der Bildpunkte von ihren Epipolarlinien in Pixeln. In den nächsten beiden Spalten stehen die Fehler der Rückprojektion der rekonstruierten Punkte in die linke bzw. rechte Bildebene in Pixeln. Der Fehler der rekonstruierten Punkte bezüglich der bekannten Punkte in der Doppelebene in Millimetern ist schließlich in der letzten Spalte aufgeführt.

In dem oberen Teil der beiden Tabellen finden sich die Ergebnisse für die direkte Kalibrierung unter Verwendung aller Korrespondenzen. Der untere Teil enthält jeweils die Ergebnisse der in Abschnitt 6.3 von [68] vorgeschlagenen Methode der Random Samples. Da in dieser Variante der Zufall eine große Rolle spielt, werden alle Tests dreimal durchgeführt und die Resultate aufgelistet.

Test	Epipolarfehler $[px]$	Rückprojektionsfehler $[px]$		Rekonstruktionsfehler $[mm]$
		Links	Rechts	
1	$6{,}57 \cdot 10^{-2} \pm 6{,}35 \cdot 10^{-2}$	$1{,}97 \cdot 10^{-3} \pm 1{,}92 \cdot 10^{-3}$	$6{,}39 \cdot 10^{-2} \pm 6{,}27 \cdot 10^{-2}$	$4{,}86 \pm 4{,}15$
2	$6{,}57 \cdot 10^{-2} \pm 6{,}35 \cdot 10^{-2}$	$3{,}38 \cdot 10^{-5} \pm 3{,}29 \cdot 10^{-5}$	$6{,}58 \cdot 10^{-2} \pm 6{,}46 \cdot 10^{-2}$	$4{,}87 \pm 4{,}17$
3	$6{,}57 \cdot 10^{-2} \pm 6{,}35 \cdot 10^{-2}$	$1{,}97 \cdot 10^{-3} \pm 1{,}92 \cdot 10^{-3}$	$6{,}39 \cdot 10^{-2} \pm 6{,}27 \cdot 10^{-2}$	$5{,}47 \pm 4{,}08$
4	$6{,}57 \cdot 10^{-2} \pm 6{,}35 \cdot 10^{-2}$	$3{,}37 \cdot 10^{-5} \pm 3{,}29 \cdot 10^{-5}$	$6{,}58 \cdot 10^{-2} \pm 6{,}46 \cdot 10^{-2}$	$5{,}48 \pm 4{,}08$
1	$8{,}50 \cdot 10^{-2} \pm 8{,}39 \cdot 10^{-2}$	$4{,}75 \cdot 10^{-2} \pm 4{,}72 \cdot 10^{-2}$	$3{,}75 \cdot 10^{-2} \pm 3{,}75 \cdot 10^{-2}$	$5{,}18 \pm 4{,}46$
	$6{,}84 \cdot 10^{-2} \pm 7{,}30 \cdot 10^{-2}$	$3{,}66 \cdot 10^{-2} \pm 3{,}93 \cdot 10^{-2}$	$3{,}18 \cdot 10^{-2} \pm 3{,}44 \cdot 10^{-2}$	$4{,}96 \pm 4{,}37$
	$9{,}71 \cdot 10^{-2} \pm 1{,}05 \cdot 10^{-1}$	$2{,}72 \cdot 10^{-2} \pm 2{,}98 \cdot 10^{-2}$	$6{,}99 \cdot 10^{-2} \pm 7{,}68 \cdot 10^{-2}$	$6{,}45 \pm 5{,}75$
2	$7{,}44 \cdot 10^{-2} \pm 8{,}01 \cdot 10^{-2}$	$2{,}00 \cdot 10^{-3} \pm 2{,}17 \cdot 10^{-3}$	$7{,}27 \cdot 10^{-2} \pm 7{,}96 \cdot 10^{-2}$	$5{,}62 \pm 4{,}98$
	$7{,}50 \cdot 10^{-2} \pm 7{,}35 \cdot 10^{-2}$	$2{,}25 \cdot 10^{-4} \pm 2{,}21 \cdot 10^{-4}$	$7{,}50 \cdot 10^{-2} \pm 7{,}46 \cdot 10^{-2}$	$5{,}20 \pm 4{,}60$
	$8{,}00 \cdot 10^{-2} \pm 7{,}84 \cdot 10^{-2}$	$7{,}00 \cdot 10^{-3} \pm 6{,}90 \cdot 10^{-3}$	$7{,}32 \cdot 10^{-2} \pm 7{,}28 \cdot 10^{-2}$	$5{,}39 \pm 4{,}78$
3	$7{,}18 \cdot 10^{-2} \pm 7{,}86 \cdot 10^{-2}$	$4{,}00 \cdot 10^{-3} \pm 4{,}41 \cdot 10^{-3}$	$6{,}81 \cdot 10^{-2} \pm 7{,}57 \cdot 10^{-2}$	$6{,}27 \pm 4{,}42$
	$8{,}99 \cdot 10^{-2} \pm 8{,}37 \cdot 10^{-2}$	$6{,}46 \cdot 10^{-2} \pm 6{,}05 \cdot 10^{-2}$	$2{,}52 \cdot 10^{-2} \pm 2{,}38 \cdot 10^{-2}$	$5{,}90 \pm 4{,}06$
	$7{,}48 \cdot 10^{-2} \pm 7{,}62 \cdot 10^{-2}$	$3{,}46 \cdot 10^{-3} \pm 3{,}55 \cdot 10^{-3}$	$7{,}15 \cdot 10^{-2} \pm 7{,}39 \cdot 10^{-2}$	$5{,}62 \pm 4{,}35$
4	$6{,}72 \cdot 10^{-2} \pm 8{,}40 \cdot 10^{-2}$	$4{,}24 \cdot 10^{-3} \pm 5{,}35 \cdot 10^{-3}$	$6{,}32 \cdot 10^{-2} \pm 8{,}02 \cdot 10^{-2}$	$6{,}17 \pm 4{,}50$
	$7{,}04 \cdot 10^{-2} \pm 8{,}07 \cdot 10^{-2}$	$9{,}95 \cdot 10^{-4} \pm 1{,}15 \cdot 10^{-3}$	$6{,}97 \cdot 10^{-2} \pm 8{,}12 \cdot 10^{-2}$	$6{,}37 \pm 4{,}69$
	$7{,}76 \cdot 10^{-2} \pm 8{,}24 \cdot 10^{-2}$	$4{,}37 \cdot 10^{-4} \pm 4{,}67 \cdot 10^{-4}$	$7{,}75 \cdot 10^{-2} \pm 8{,}35 \cdot 10^{-2}$	$5{,}66 \pm 4{,}51$

Tabelle 6.1: Mittlere Fehler und Standardabweichungen für Kalibrierungsvarianten 1 bis 4 im Testfall Fernsicht, oben: direkt, unten: Random Samples.

Test	Epipolarfehler $[px]$	Rückprojektionsfehler $[px]$		Rekonstruktionsfehler $[mm]$
		Links	Rechts	
1	$9{,}69 \cdot 10^{-2} \pm 7{,}70 \cdot 10^{-2}$	$1{,}24 \cdot 10^{-3} \pm 9{,}82 \cdot 10^{-4}$	$9{,}57 \cdot 10^{-2} \pm 7{,}78 \cdot 10^{-2}$	$8{,}89 \pm 9{,}85$
2	$9{,}68 \cdot 10^{-2} \pm 7{,}72 \cdot 10^{-2}$	$2{,}87 \cdot 10^{-5} \pm 2{,}29 \cdot 10^{-5}$	$9{,}69 \cdot 10^{-2} \pm 7{,}89 \cdot 10^{-2}$	$8{,}96 \pm 9{,}84$
3	$9{,}69 \cdot 10^{-2} \pm 7{,}70 \cdot 10^{-2}$	$1{,}24 \cdot 10^{-3} \pm 9{,}82 \cdot 10^{-4}$	$9{,}57 \cdot 10^{-2} \pm 7{,}78 \cdot 10^{-2}$	$11{,}30 \pm 13{,}12$
4	$9{,}68 \cdot 10^{-2} \pm 7{,}72 \cdot 10^{-2}$	$2{,}87 \cdot 10^{-5} \pm 2{,}29 \cdot 10^{-5}$	$9{,}69 \cdot 10^{-2} \pm 7{,}89 \cdot 10^{-2}$	$11{,}35 \pm 13{,}16$
1	$9{,}75 \cdot 10^{-2} \pm 9{,}71 \cdot 10^{-2}$	$4{,}91 \cdot 10^{-4} \pm 4{,}89 \cdot 10^{-4}$	$9{,}74 \cdot 10^{-2} \pm 9{,}87 \cdot 10^{-2}$	$9{,}12 \pm 10{,}12$
	$2{,}43 \cdot 10^{-1} \pm 3{,}24 \cdot 10^{-1}$	$3{,}30 \cdot 10^{-3} \pm 4{,}52 \cdot 10^{-3}$	$2{,}44 \cdot 10^{-1} \pm 3{,}31 \cdot 10^{-1}$	$9{,}31 \pm 6{,}42$
	$1{,}34 \cdot 10^{-1} \pm 1{,}23 \cdot 10^{-1}$	$6{,}93 \cdot 10^{-3} \pm 6{,}42 \cdot 10^{-3}$	$1{,}28 \cdot 10^{-1} \pm 1{,}20 \cdot 10^{-1}$	$9{,}17 \pm 10{,}76$
2	$1{,}34 \cdot 10^{-1} \pm 1{,}01 \cdot 10^{-1}$	$6{,}64 \cdot 10^{-5} \pm 5{,}07 \cdot 10^{-5}$	$1{,}34 \cdot 10^{-1} \pm 1{,}02 \cdot 10^{-1}$	$8{,}18 \pm 8{,}17$
	$3{,}09 \cdot 10^{-1} \pm 4{,}36 \cdot 10^{-1}$	$5{,}70 \cdot 10^{-4} \pm 8{,}28 \cdot 10^{-4}$	$3{,}16 \cdot 10^{-1} \pm 4{,}54 \cdot 10^{-1}$	$10{,}35 \pm 6{,}92$
	$1{,}35 \cdot 10^{-1} \pm 1{,}23 \cdot 10^{-1}$	$8{,}44 \cdot 10^{-5} \pm 7{,}81 \cdot 10^{-5}$	$1{,}35 \cdot 10^{-1} \pm 1{,}24 \cdot 10^{-1}$	$8{,}79 \pm 9{,}26$
3	$1{,}21 \cdot 10^{-1} \pm 9{,}32 \cdot 10^{-2}$	$9{,}88 \cdot 10^{-4} \pm 7{,}72 \cdot 10^{-4}$	$1{,}20 \cdot 10^{-1} \pm 9{,}38 \cdot 10^{-2}$	$10{,}38 \pm 10{,}91$
	$2{,}36 \cdot 10^{-1} \pm 2{,}88 \cdot 10^{-1}$	$8{,}38 \cdot 10^{-3} \pm 1{,}05 \cdot 10^{-2}$	$2{,}31 \cdot 10^{-1} \pm 2{,}87 \cdot 10^{-1}$	$10{,}38 \pm 8{,}56$
	$1{,}14 \cdot 10^{-1} \pm 1{,}01 \cdot 10^{-1}$	$4{,}74 \cdot 10^{-3} \pm 4{,}22 \cdot 10^{-3}$	$1{,}10 \cdot 10^{-1} \pm 9{,}92 \cdot 10^{-2}$	$11{,}23 \pm 12{,}99$
4	$1{,}52 \cdot 10^{-1} \pm 9{,}49 \cdot 10^{-2}$	$7{,}39 \cdot 10^{-5} \pm 4{,}66 \cdot 10^{-5}$	$1{,}53 \cdot 10^{-1} \pm 9{,}63 \cdot 10^{-2}$	$10{,}16 \pm 9{,}73$
	$1{,}16 \cdot 10^{-1} \pm 1{,}05 \cdot 10^{-1}$	$5{,}03 \cdot 10^{-5} \pm 4{,}59 \cdot 10^{-5}$	$1{,}16 \cdot 10^{-1} \pm 1{,}06 \cdot 10^{-1}$	$10{,}96 \pm 12{,}61$
	$1{,}12 \cdot 10^{-1} \pm 1{,}05 \cdot 10^{-1}$	$1{,}30 \cdot 10^{-5} \pm 1{,}21 \cdot 10^{-5}$	$1{,}13 \cdot 10^{-1} \pm 1{,}06 \cdot 10^{-1}$	$11{,}59 \pm 13{,}15$

Tabelle 6.2: Mittlere Fehler und Standardabweichungen für Kalibrierungsvarianten 1 bis 4 im Testfall Nahsicht, oben: direkt, unten: Random Samples.

Betrachtet man die Tabellen 6.1 und 6.2, so stellt man fest, dass die Kalibrierungsergebnisse ungeachtet der Datennormalisierung und der Methode der Random Samples ähnlich sind. Da die Ergebnisse der Random Samples in der Tendenz etwas schlechter und durch die zufällige Korrespondenzauswahl nicht reproduzierbar sind, ist die direkte Kalibrierung vorzuziehen. Darüber hinaus benötigt die Kalibrierung mit Random Samples ein Vielfaches der Rechenzeit, auch wenn sie sich gut parallelisieren ließe.

Eine Normalisierung der Weltdaten erhöht den mittleren metrischen Rekonstruktionsfehler um wenige Millimeter, während eine Normalisierung der Pixelkoordinaten den Rückprojektionsfehler im linken Bild verringert und die anderen Fehler praktisch unverändert lässt.

Eine Kalibrierung unter Verwendung einer Aufnahme der Doppelebene mit mehreren kleinen Quadraten zeigt andere Kalibrierungsergebnisse. Hier verbessert eine Normalisierung der Pixelkoordinaten die Ergebnisse sehr. Auch die Methode der Random Samples liefert stark unterschiedliche Ergebnisse. Für die Fernsicht werden mit 16 Korrespondenzen Rekonstruktionsfehler von $4{,}23\,mm \pm 2{,}13\,mm$ bis $98{,}34\,mm \pm 59{,}81\,mm$ erzielt. Bei der Nahsicht treten mit 47 Korrespondenzen Fehler von $3{,}52\,mm \pm 2{,}44\,mm$ bis $631{,}94\,mm \pm 1360{,}72\,mm$ auf. Der etwas größere aber stabile Rekonstruktionsfehler bei der Verwendung mehrerer Aufnahmen lässt sich durch die Messungenauigkeiten bei der Transformation der Koordinatensysteme der Doppelebenen in das Referenzkoordinatensystem aus Abbildung 4.6(d) erklären. Die Kalibrierung mit nur einer Aufnahme der Doppelebene erweist sich als zu speziell und bewirkt Fehler bei der Rekonstruktion von Objekten in anderen Abständen.

Aus den durchgeführten Tests lässt sich ableiten, dass die direkte Kalibrierung mit der Normalisierung der Bilddaten vor der Berechnung der Fundamentalmatrix F die besten Resultate erzielt. Außerdem sollten mindestens zwei Aufnahmen der Doppelebene aus unterschiedlichen Entfernungen verwendet werden. Daher wird das Stereokamerasystem mit dieser Kombination der Algorithmen, die durch entsprechende Einstellungen im Kalibrierungsdialog aus Abbildung 5.8 auswählbar ist, kalibriert und so für die dreidimensionale, metrische Rekonstruktion verwendet.

Durchgeführte Kalibrierungen für beide Gestelle werden nur wenig beeinflusst, wenn die Kameras zwischen der Nah- und Fernsicht gewechselt werden. Es ist jedoch darauf zu achten, dass immer dieselbe Kamera rechts bzw. links montiert wird, da ein Vertauschen größere Auswirkungen auf die Epipolargeometrie hat. Werden neue Gestelle oder Kameras verwendet, so ist eine erneute Kalibrierung unerlässlich. Kleinere Änderungen hingegen, wie eine Veränderung der Brennweite durch Drehen am Objektiv der verwendeten Webcams, haben auch nur geringe Auswirkungen und erfordern nicht

notwendigerweise eine Kalibrierung.

6.1.2 Camera Calibration Toolbox

Abschließend werden die Kalibrierungsergebnisse der in dieser Diplomarbeit entwickelten Software mit den Werten der Camera Calibration Toolbox for Matlab [5] verglichen. In dieser Toolbox wird unter anderem das Verfahren aus [67] verwendet, das in dieser Arbeit nicht eingesetzt wird, wie in Kapitel 5.2.3 begründet. Mit der Toolbox werden beide Gestelle durch jeweils fünf Stereobildpaare eines ebenen Schachbrettmusters aus unterschiedlichen Richtungen kalibriert. Die Ausgabe der Kalibrierung besteht aus den Kameramatrizen aus Gleichung (3.9) und der räumlichen Anordnung $(R;\mathbf{t})$ aus Gleichung (3.13) der Kameras zueinander. Dadurch ist eine Rekonstruktion nur in den Kamerakoordinatensystemen S_{k_l} und S_{k_r} möglich, die im Gegensatz zur hier verwendeten Orientierung um 180° um die z-Achse gedreht sind.

Die berechneten extrinsischen Parameter zeigen Abweichungen von den erwarteten Werten, die trotz beschränkter Genauigkeit bei der Gestellkonstruktion und den Messungen etwas groß erscheinen. So wird für die Nahsicht ein Winkel $\theta = 20{,}4°$ und für beide Gestelle eine Verschiebung in z-Richtung von $7{,}9\,mm$ bzw. $8{,}1\,mm$ bestimmt. Bei der Nahsicht wird außerdem eine Abweichung von $6{,}9\,mm$ in y-Richtung berechnet.

Wird mit den, für die in dieser Arbeit neu entwickelten Kalibrierung, verwendeten Korrespondenzen eine Rekonstruktion in der Toolbox durchgeführt und in ein Referenzkoordinatensystem transformiert, dass zwischen den Foki der Kameras liegt, so kann ein mittlerer Rekonstruktionsfehler berechnet werden. Für die Fernsicht bzw. Nahsicht beträgt dieser Fehler $50{,}66\,mm \pm 10{,}31\,mm$ bzw. $80{,}07\,mm \pm 23{,}04\,mm$. Diese Fehler lassen sich nur schwer mit den Tabellen 6.1 und 6.2 vergleichen, da das Referenzkoordinatensystem der Toolbox einige Zentimeter über dem in dieser Arbeit verwendeten liegt. Einen Anhaltspunkt für die Qualität dieser Kalibrierung erhält man durch die Berechnung des mittleren Fehlervektors

$$\mathbf{e}_{\text{Fern}} = \begin{pmatrix} 25{,}86 \\ -39 \\ -9{,}16 \end{pmatrix} mm \pm \begin{pmatrix} 8{,}79 \\ 15{,}8 \\ 9{,}24 \end{pmatrix} mm$$

$$\mathbf{e}_{\text{Nah}} = \begin{pmatrix} 15{,}27 \\ -20{,}57 \\ -68{,}26 \end{pmatrix} mm \pm \begin{pmatrix} 8{,}75 \\ 26{,}49 \\ 29{,}92 \end{pmatrix} mm$$

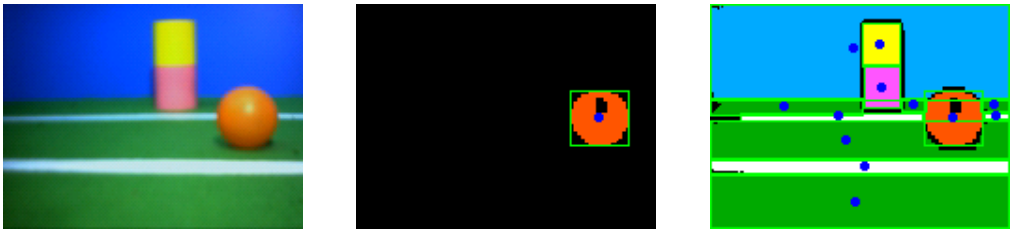
der Rekonstruktion, welcher der Verschiebung zwischen den beiden Koordinatensystemen entsprechen sollte, bei der kleine x- und z-Werte bis $10\,mm$

sowie y-Werte von etwa $85\,mm$ vorhanden sind.

Die Abweichungen der extrinsischen Parameter, die großen Standardabweichungen der mittleren Fehler sowie der fehlende Zusammenhang zwischen der erwarteten Verschiebung und den Fehlervektoren zeigen, dass die neu implementierten Verfahren etwas bessere Ergebnisse liefern als die bestehende Camera Calibration Toolbox for Matlab.

6.2 Segmentierung

Die Resultate der in dieser Diplomarbeit implementierten Farbsegmentierung (siehe Kapitel 5.3) werden im Folgenden aufgeführt und bewertet. Abbildung 6.2(a) zeigt das Bild der Kamera, das für die Tests verwendet wird. In Abbildung 6.2(b) wird die Szene mit nur einem Farbbereich segmentiert, wodurch ausschließlich der Ball erkannt wird. Durch die Verwendung mehrerer Farbbereiche wird die komplexe Szene wie in Abbildung 6.2(c) segmentiert. In beiden Beispielen sind sowohl die berechneten Flächenschwerpunkte dunkelblau als auch die Bounding Boxen hellgrün eingezeichnet.



(a) (b) (c)

Abbildung 6.2: Beispiele für die Farbsegmentierung: (a) Bild von der Kamera (b) Erkennung des Balls (c) Erkennung der komplexen Szene.

Mit einer geeigneten Beleuchtung und Konfiguration der Kamera sowie der Farbbereiche ist eine sehr gute Farbsegmentierung möglich. Eine Voraussetzung für eine solche Segmentierung sind allerdings deutlich unterscheidbare Farben der betrachteten Objekten, wie es im *RoboCup* der Fall ist. Wie in Kapitel 5.2.2 beschrieben wird, werden die Farben an den Bildrändern von der Kamera verfälscht, was durch eine Vergrößerung der Farbbereiche ausgeglichen wird.

Ein großer Vorteil des verwendeten Algorithmus zur Farbsegmentierung ist seine Geschwindigkeit. Zur Messung der Laufzeit wird die Szene aus Abbil-

dung 6.2 mit beiden Farbkonfigurationen jeweils 100-mal aufgenommen und segmentiert. Die durchschnittlichen Laufzeiten finden sich in Tabelle 6.3. Diese enthält die Messungen des ursprünglichen, nicht optimierten Algorithmus und die für den Humanoiden optimierte Variante. Die wichtigsten Optimierungen werden in Kapitel 5.3.3 erläutert.

System	einfache Szene		komplexe Szene	
	original	optimiert	original	optimiert
Entwicklungsrechner	3,86	2,79	7,30	4,45
humanoider Roboter	192,38	20,01	1634,41	30,46

Tabelle 6.3: Durchschnittliche Laufzeit für die Beispiele aus Abbildung 6.2 in Millisekunden mit und ohne Optimierungen.

Wie erwartet, benötigt die Segmentierung auf dem Humanoiden mehr Zeit als auf dem Entwicklungsrechner. Die unveränderte Version des Algorithmus ist viel zu langsam für den Einsatz in mobilen Robotersystemen. Jedoch führen die im Rahmen dieser Diplomarbeit entwickelten Optimierungen auf dem humanoiden Roboter zu einer drastischen Steigerung der Geschwindigkeit. Je nach Komplexität der betrachteten Szene ist damit eine Zeitersparnis von 89% bis 99% möglich, was einer Leistungssteigerung um den Faktor 9 bis 100 entspricht. Je mehr kleine Objekte erkannt werden müssen, desto länger benötigt der Algorithmus. Allerdings treten dann die Optimierungen deutlicher in Erscheinung.

Die angegebenen Laufzeiten beziehen sich auf die Segmentierung eines Bildes inklusive der Bestimmung von Objektgröße, Flächenschwerpunkt und Bounding Box. Zur Bestimmung der Position von Objekten wie in Kapitel 5.4.2 wird ein Stereobildpaar segmentiert, wodurch sich der Aufwand verdoppelt.

Schließlich ist auf dem humanoiden Robotersystem nicht nur die Rechenzeit, sondern auch der Speicherplatz beschränkt. Daher werden Messungen durchgeführt, um den Verbrauch des Speichers zu bestimmen. Hierfür wird ein einfaches Programm entwickelt, das Stereobildpaare von den Kameras holt und diese segmentiert. Für Szenen wie in Abbildung 6.2(b) bzw. 6.2(c) werden etwa 337 Kilobyte bzw. 367 Kilobyte Speicher benötigt. Dieser relative hohe Speicherbedarf resultiert unter anderem aus den Laufzeitoptimierungen, da für die Speicherung der Daten ausreichend Platz reserviert wird.

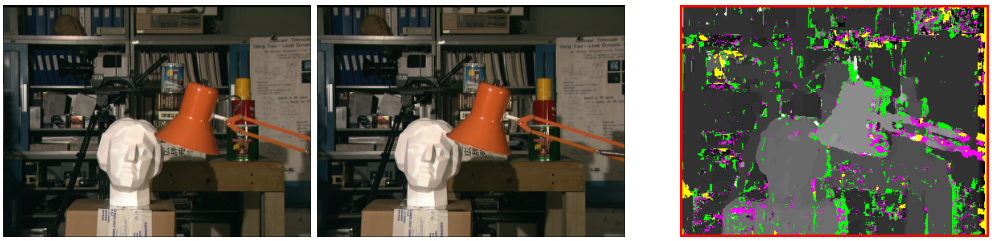
6.3 Positionsbestimmung

Die Ergebnisse der implementierten und in Kapitel 5.4 beschriebenen Verfahren zur Korrespondenz- und Positionsbestimmung werden im Folgenden

aufgeführt.

6.3.1 Flächenbasierte Methode

Mithilfe des Algorithmus aus Kapitel 5.4.1 werden Graustufenbilder mit den Disparitäten in einem gleichgerichteten Stereobildpaar erzeugt. Zu Testzwecken werden entsprechende Bildpaare [33] verwendet. Abbildung 6.3 zeigt ein Stereobildpaar mit dem berechneten Graustufenbild. Die Graustufen sind für die Anzeige mit dem Faktor 5 skaliert.



(a) (b)

Abbildung 6.3: (a) Stereobildpaar [33] zum Testen der flächenbasierten Korrespondenzbestimmung (b) berechnetes Graustufenbild der Disparitäten

In Abbildung 6.3(b) sind einige Pixel farbig hervorgehoben, die während der Berechnung aus verschiedenen Gründen als ungültig markiert werden.

Rot ist der Bildrand, der durch das quadratische Suchfenster nicht bearbeitet werden kann.

Grüne Pixel bestehen die Konsistenzprüfung aus Gleichung (5.30) nicht. Dies tritt vor allem auf, wenn Objekte im rechten Bild sichtbar aber, im linken Bild verdeckt sind.

Gelb sind die Pixel, deren Korrespondenz nach der Berechnung am Bildrand liegt. Da die tatsächliche Korrespondenz außerhalb des Bildes liegen könnte, werden sie als ungültig markiert.

Violett sind alle Pixel, zu denen bei der Berechnung der SAD-Werte kein eindeutiges Minimum gefunden werden kann (siehe Kapitel 5.4.1).

Obwohl es viele kleinere Störungen in dem Ergebnis gibt, sind die größeren Bereiche, wie die Lampe, der Kopf und der Tisch, deutlich zu erkennen.

Durch die Anwendung eines Medianfilters können viele der ungültigen Pixel entfernt werden, wenn genügend Rechenzeit zur Verfügung steht.

Die tatsächliche Leistungsfähigkeit zeigt der Algorithmus, wenn er auf ein Stereobildpaar aus zufälligen, farbigen Punkten angewendet wird. Durch Verschiebung einiger Bildbereiche werden künstlich Entfernungsinformationen eingefügt. Ein solches Bildpaar mit dem berechneten Graustufenbild ist in Abbildung 6.4(a) zu sehen. Das Ergebnis in 6.4(c) weicht nur in den verdeckten Bereichen und am rechten Bildrand von dem exakten Graustufenbild in 6.4(b) ab. Selbst wenn in dem Stereobildpaar starkes gaußsches Rauschen mit dem Mittelwert $\mu = 0$ und der Varianz $\sigma^2 = 0{,}04$ vorliegt (was bei 256 Werten pro Farbkanal einer Standardabweichung von 51 entspricht), treten nur wenige zusätzliche Fehler an den Sprüngen der Disparitäten auf, wie Abbildung 6.4(d) zeigt. Für die Anzeige sind die Graustufen mit dem Faktor 6 skaliert.

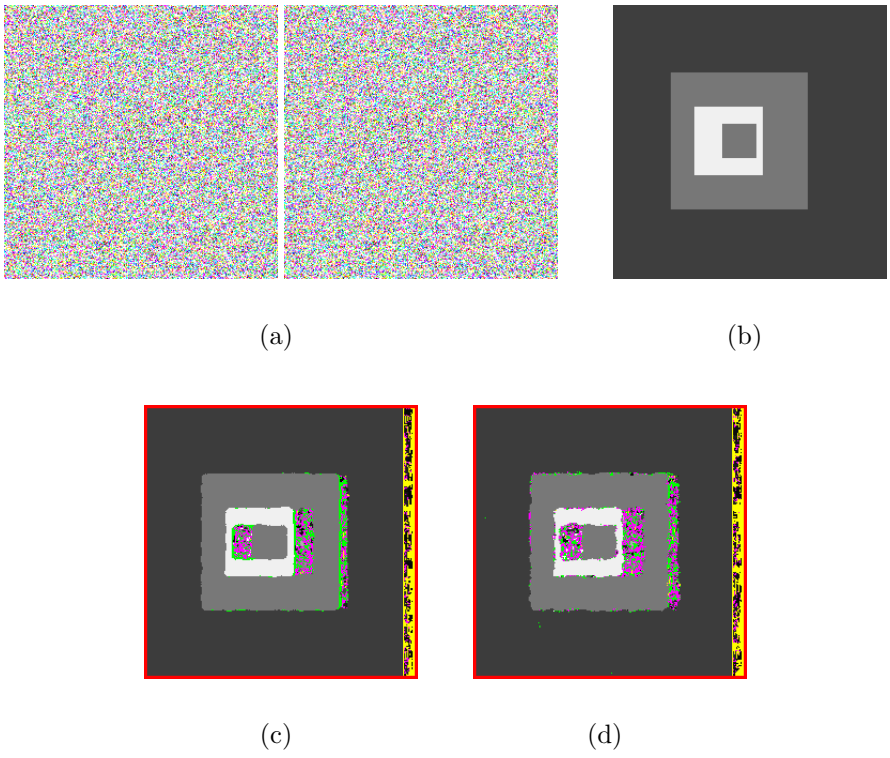
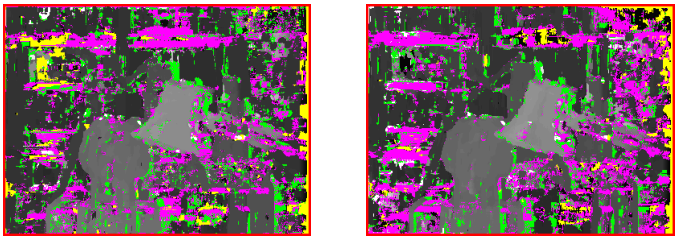


Abbildung 6.4: Verwendung zufälliger Punkte: (a) Stereobildpaar (b) exakte Disparitäten (c) berechnetes Graustufenbild (d) Resultat mit gaußschem Rauschen in der Eingabe.

Trotz der guten Ergebnisse ist das Verfahren nicht für den Einsatz auf dem

humanoiden Roboter geeignet. Es ist nicht sehr robust, wenn das Stereobildpaar die Fundamentalmatrix aus Gleichung (3.22) nicht exakt erfüllt. Abbildung 6.5 zeigt zwei Ergebnisse, wenn das linke Bild aus Abbildung 6.3(a) um ein Pixel nach unten verschoben oder um 0,5° im Uhrzeigersinn gedreht wird.



(a) (b)

Abbildung 6.5: Fehler bei der Korrespondenzbestimmung durch Veränderung des linken Bildes aus Abbildung 6.3(a): (a) 1 Pixel nach unten verschoben (b) 0,5° im Uhrzeigersinn gedreht.

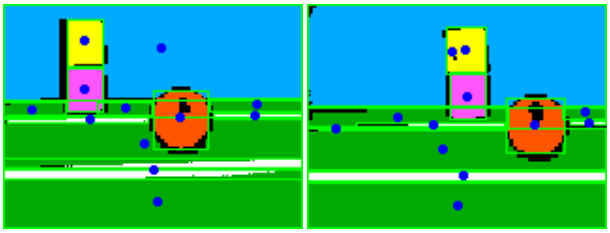
Fehler dieser Art bewirken mehr Störungen in dem berechneten Graustufenbild. Durch eine Vorverarbeitung des Stereobildpaares wie in [20, 31, 46] oder Kapitel 11.12 von [14] kann ein Stereobildpaar gleichgerichtet werden. Allerdings steht auf dem Humanoiden nicht genügend Rechenleistung zur Verfügung, um dies schnell durchführen zu können. Weiterhin muss geprüft werden, ob die Kalibrierung des Stereokamerasystems exakt genug ist, damit eine solche Vorverarbeitung ein Stereobildpaar in der benötigten Qualität erzeugen kann.

Darüber hinaus ist das Verfahren selbst zu langsam für den Einsatz auf dem autonomen Roboter. Das Entwicklungssystem (Anhang A.1) benötigt für Bilder der Größe 160×120 etwa 380 Millisekunden. Auch wenn diese Implementierung noch nicht für den Humanoiden optimiert ist, kommt noch der Zeitbedarf für die beschriebene Bildvorverarbeitung und die eigentliche Positionsberechnung wie in Kapitel 3.3.1 hinzu. Eine starke Beschleunigung nach einer Optimierung ist durch die vielen schreibende Array-Zugriffe nicht zu erwarten. Daher wird die Bearbeitungsdauer auf dem Humanoiden für ein Stereobildpaar einige Sekunden betragen und dieser Ansatz nicht weiter verfolgt.

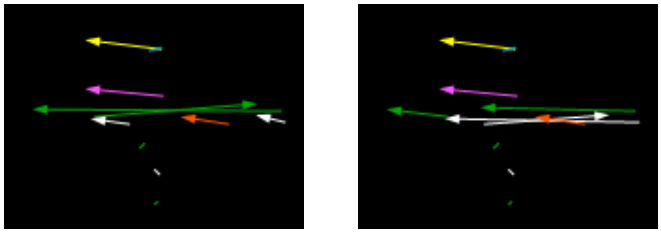
6.3.2 Merkmalsbasierte Methoden

6.3.2.1 Position mit optimalen Voraussetzungen

Für die Rekonstruktion mit den in Kapitel 5.4.2 beschriebenen Verfahren ist ein segmentiertes Stereobildpaar nötig. Ein solches Bildpaar ist in Abbildung 6.6(a) zu sehen. Abbildung 6.6(b) bzw. 6.6(c) zeigt die berechneten Disparitäten vom rechten ins linke Bild mit den Algorithmen `DisparitySize` bzw. `DisparityCentroid`. Die Farben der Pfeile verdeutlichen dabei das Objekt, dessen Disparität sie repräsentieren. Bei kleinen Disparitäten wird die Pfeilspitze aus Gründen der Übersichtlichkeit weggelassen.



(a)



(b) (c)

Abbildung 6.6: Merkmalsbasierte Disparitäten: (a) segmentiertes Stereobildpaar (b) Berechnung mit `DisparitySize` (c) Berechnung mit `DisparityCentroid`.

Berechnet man die Positionen der Objekte aus den bestimmten Korrespondenzen, so wird mit beiden Varianten die Position des Balls und des *RoboCup*-Markers identisch rekonstruiert. Da es schwierig ist, die euklidischen Koordinaten zu messen, wird nur der Abstand zu diesen Objekten gemessen und mit dem berechneten Abstand verglichen. Der Berechnung zufolge hat der Ball einen Abstand von 675 mm und der untere bzw. obere Teil des Markers einen Abstand von 1126,4 mm bzw. 1088,7 mm. Mit einer Messung wird die Entfer-

nung des Balls mit 660 mm und die des Markers mit 1110 mm bestimmt. Der Rekonstruktionsfehler liegt also zwischen 15 mm und 22 mm und ist damit sehr gering.

Die Positionen des Tores, der Linien und der Rasenstücke werden nicht erfolgreich bestimmt, da diese Objekte in keinem der Bilder vollständig sichtbar sind und damit die Flächenschwerpunkte nicht die selben Punkte des Raumes beschreiben. Die berechneten Korrespondenzen beider Varianten unterscheiden sich bei vier dieser Objekte wie man durch Vergleichen der Abbildungen 6.6(b) und 6.6(c) feststellt.

Für eine genauere Bewertung der Rekonstruktionsergebnisse wird die Position eines *RoboCup*-Markers wie in Abbildung 6.7 in verschiedenen Abständen bestimmt und mit seiner bekannten Position verglichen.

Abbildung 6.7: *RoboCup*-Marker für die Rekonstruktion.

Der Mittelpunkt des unteren bzw. oberen Teils des Markers befindet sich bei ${}^0\mathbf{p} = (39; 50; z)^T\, mm$ bzw. ${}^0\mathbf{p} = (39; 150; z)^T\, mm$ im Referenzkoordinatensystem. Tabelle 6.4 zeigt die rekonstruierten Positionen ${}^0\tilde{\mathbf{p}}$ für verschiedene Entfernungen z und die Abstände von den gemessenen Positionen.

z	unterer Teil		oberer Teil	
	${}^0\tilde{\mathbf{p}}^T$	$\|{}^0\mathbf{p} - {}^0\tilde{\mathbf{p}}\|$	${}^0\tilde{\mathbf{p}}^T$	$\|{}^0\mathbf{p} - {}^0\tilde{\mathbf{p}}\|$
500	(43; 52; 511)	11,87	(42; 155; 512)	13,34
1000	(46; 54; 1040)	40,80	(47; 160; 1033)	35,40
2000	(57; 53; 2066)	68,48	(54; 155; 2059)	61,08
3000	(60; 48; 3087)	89,52	(56; 165; 3041)	46,85

Tabelle 6.4: Rekonstruktion des Markers für verschiedene Entfernungen in Millimetern.

Mit zunehmender Entfernung wächst auch der Fehler in x- und z-Richtung, während die Position entlang der y-Achse nahezu unverändert bleibt. Mit einem relativen Fehler $\frac{\|{}^0\mathbf{p} - {}^0\tilde{\mathbf{p}}\|}{\|{}^0\mathbf{p}\|}$ zwischen 1,5% und 4% sind die erzielten Rekonstruktionen sehr genau.

Wie in Kapitel 3.1.2 beschrieben wird, werden geringe Entfernungen besser aufgelöst als große Entfernungen. Dies wird deutlich, wenn sich aufgrund von Rauschen die Flächenschwerpunkte ändern. Verschiebt sich im Fall $z = 500$ der Schwerpunkt des unteren Teils des Markers im rechten Bild um ein Pixel nach links, so verringert sich die Position entlang der z-Achse um $10\,mm$ auf ${}^0\tilde{p}_z = 501\,mm$. Verschiebt sich aber der Schwerpunkt des unteren Teils für den Fall $z = 3000$ im linken Bild um ein Pixel nach links, so ändert sich die Rekonstruktion ${}^0\tilde{\mathbf{p}} = (75; 43; 3528)^T$ um etwa einen halben Meter.

Die Algorithmen `DisparitySize` und `DisparityCentroid` können auf dem Entwicklungssystem sehr schnell ausgeführt werden, während sie auf dem Humanoiden lange benötigen. Tabelle 6.5 enthält die Laufzeiten der Verfahren auf beiden Systemen jeweils für eine einfache Szene mit nur einem Ball und eine komplexe Szene wie in Abbildung 6.6(a). Die angegebenen Zeiten sind Durchschnittswerte für 100 Durchläufe, wobei für diese Messungen die Optimierungen von Kapitel 5.4.3 verwendet werden.

System	einfache Szene		komplexe Szene	
	`Size`	`Centroid`	`Size`	`Centroid`
Entwicklungsrechner	0,14	0,16	0,87	1,15
humanoider Roboter	72,88	83,64	800,43	971,55
davon Singulärwertzerlegung	67,62	67,81	726,44	735,71

Tabelle 6.5: Durchschnittliche Laufzeit der Positionsberechnung in Millisekunden für die Algorithmen `DisparitySize` und `DisparityCentroid`.

Die Messungen zeigen, dass die Dauer der Positionsberechnung auf dem Entwicklungssystem zu vernachlässigen ist. Auf dem System des humanoiden Roboters ist sie jedoch sehr lang. Dabei wird der größte Anteil durch die eigentliche Rekonstruktion mit der Singulärwertzerlegung verbraucht, wie die dritte Zeile von Tabelle 6.5 verdeutlicht. Für sämtliche Rechnungen werden fast ausschließlich Elementaroperationen verwendet. Da in der Singulärwertzerlegung vergleichsweise wenig Quadratwurzeln berechnet werden, ist der Verbrauch der Laufzeit bei den schreibenden Array-Zugriffen mit Fließkommazahlen zu suchen (siehe Tabelle B.1). Eine Optimierung durch die Verwendung von Festkommazahlen mit 32 Bit ist kaum möglich, da sich die Einträge in der Fundamentalmatrix bzw. in den Projektionsmatrizen um sechs Größenordnungen unterscheiden. Damit ist keine hinreichende Genauigkeit mehr gegeben, weil in 32 Bit maximal neun Größenordnungen gespeichert werden können und damit höchstens drei Dezimalstellen erhalten bleiben.

Ähnlich wie bei der Segmentierung wird auch bei der Positionsberechnung der Speicherverbrauch gemessen. Hierzu werden mit einer einfachen Anwendung Stereobildpaare aufgenommen und segmentiert. Anschließend werden

die Korrespondenzen bestimmt und die Positionen rekonstruiert. Für Szenen mit der Komplexität von Abbildung 6.2(b) bzw. 6.2(c) belegt die Anwendung mit `DisparitySize` etwa 344 Kilobyte bzw. 368 Kilobyte und mit `DisparityCentroid` 349 Kilobyte bzw. 372 Kilobyte Speicherplatz. Vergleicht man diese Werte mit dem Verbrauch der Segmentierung aus Kapitel 6.2 so steigt der Speicherbedarf nur um 5 bis 12 Kilobyte.

6.3.2.2 Position mit Bewegung

Die verwendeten Webcams des in dieser Arbeit entwickelten Stereokamerasystems lassen sich nicht synchronisieren wie in Kapitel 5.1 ausgeführt wird. Daher treten Rekonstruktionsfehler auf, wenn sich die betrachteten Objekte oder das Kamerasystem bewegen. Die Größe dieses Fehlers hängt von der eingestellten Bildrate der Kameras und der Bewegungsrichtung ab.

Damit die Größenordnung des Rekonstruktionsfehlers eingeschätzt werden kann, werden Messungen mit 5 bzw. 30 Bildern pro Sekunde durchgeführt. Dabei wird ein stationärer Ball mit horizontalen und vertikalen Bewegungen des Kamerakopfes aufgenommen und seine Position rekonstruiert. Die Entfernung ohne Bewegung d sowie die kleinste bzw. größte berechnete Entfernung d_{min} bzw. d_{max} sind in Tabelle 6.6 aufgeführt.

Bewegungsrichtung	Bilder pro Sekunde	d_{min}	d	d_{max}
horizontal	5	324,4	675,0	3381,7
	30	550,8	673,5	806,3
vertikal	5	608,2	658,6	711,0
	30	643,9	680,3	699,5

Tabelle 6.6: Abstände eines Balls bei horizontaler und vertikaler Kopfbewegung in Millimetern.

Anhand der Ergebnisse der durchgeführten Tests stellt man fest, dass vor allem eine horizontale Bewegung große Fehler in der Rekonstruktion verursacht. Vertikale Bewegungen haben zwar auch Fehler zur Folge, die aber wesentlich geringer ausfallen. Außerdem bewirkt eine Erhöhung der Bildrate eine Verringerung der Rekonstruktionsfehler, da die Bilder eines Stereobildpaares dann einen kleineren zeitlichen Abstand voneinander haben. Allerdings können auf dem humanoiden Robotersystem die Kameras nicht mit mehr als 5 Bildern pro Sekunde betrieben werden, weil das PWC-Modul die Datenmengen nicht mehr verarbeiten kann.

6.3.2.3 Position mit Verdeckung

Wenn Objekte teilweise verdeckt sind, verschieben sich die Flächenschwerpunkte, die zur Rekonstruktion verwendet werden. In Tabelle 6.7(a) sind für ein Objekt die Koordinaten der Flächenschwerpunkte sowie die euklidische Rekonstruktion und deren Abstand vom Referenzkoordinatensystem gegeben. Die erste Zeile enthält die Daten für das vollständig sichtbare Objekt. In den weiteren Zeilen ist das Objekt im linken Bild aus vier verschiedenen Richtungen bis ungefähr zur Hälfte verdeckt. Tabelle 6.7(b) enthält die gleichen Berechnungen, bei denen aber auf die Flächenschwerpunkte die Sampson-Näherung aus Gleichung (5.33) angewendet wird.

sichtbar	${}^l\mathbf{p}^T\ [px]$	${}^r\mathbf{p}^T\ [px]$	${}^0\mathbf{p}^T\ [mm]$	$\|{}^0\mathbf{p}\|\ [mm]$
alles	$(\ 99; 64)$	$(135; 65)$	$(-73; 70;\ 863)$	868,9
oben	$(\ 99; 57)$	$(135; 65)$	$(-73; 93;\ 859)$	867,1
unten	$(\ 99; 72)$	$(135; 65)$	$(-73; 43;\ 866)$	870,1
rechts	$(105; 64)$	$(135; 65)$	$(-68; 72;\ 714)$	720,8
links	$(\ 94; 63)$	$(135; 65)$	$(-79; 71; 1046)$	1051,4

(a)

sichtbar	${}^l\mathbf{p}^T\ [px]$	${}^r\mathbf{p}^T\ [px]$	${}^0\mathbf{p}^T\ [mm]$	$\|{}^0\mathbf{p}\|\ [mm]$
alles	$(\ 98{,}99; 63{,}42)$	$(135{,}01; 65{,}56)$	$(-73; 72;\ 863)$	869,1
oben	$(\ 99{,}08; 60{,}01)$	$(134{,}95; 62{,}07)$	$(-73; 83;\ 859)$	866,1
unten	$(\ 98{,}88; 67{,}32)$	$(135{,}04; 69{,}56)$	$(-73; 59;\ 868)$	873,1
rechts	$(104{,}98; 63{,}34)$	$(135{,}01; 65{,}64)$	$(-68; 74;\ 714)$	721,0
links	$(\ 94{,}00; 63{,}00)$	$(135{,}00; 65{,}00)$	$(-79; 71; 1046)$	1051,4

(b)

Tabelle 6.7: Rekonstruktionsfehler bei teilweise verdeckten Objekten: (a) Flächenschwerpunkte (b) Sampson-Näherung aus Gleichung (5.33).

Wie man in Tabelle 6.7(a) sieht, ändert sich nur die y-Koordinate merklich um wenige Zentimeter, wenn der obere bzw. untere Teil des Objektes verdeckt ist. Auf die berechnete Entfernung des Objektes wirkt sich dies kaum aus, da die z-Koordinate den Abstand dominiert. Sehr große Fehler treten allerdings auf, wenn die rechte bzw. linke Objekthälfe verdeckt ist. Die z-Koordinate ändert sich in diesem Beispiel um $15\,cm$ bis $20\,cm$, was sich auch direkt im Objektabstand widerspiegelt.

Die Ergebnisse der Sampson-Näherung in Tabelle 6.7(b) sind denen der di-

rekten Rekonstruktion sehr ähnlich. Die einzelnen euklidischen Koordinaten und der Abstand ändern sich maximal um wenige Millimeter. Die einzige Ausnahme bildet die y-Koordinate bei einer Verdeckung der oberen oder unteren Objekthälfte. Der Fehler wird hier etwa halbiert. Dieses Verhalten liegt daran, dass die Epipolarlinien bei den in dieser Arbeit konstruierten Stereokamerasystemen nahezu waagerecht verlaufen und die Sampson-Näherung die Pixelkoordinaten senkrecht zu den Epipolarlinien korrigiert.

Aufgrund der beschränkten Messgenauigkeit kann nicht entschieden werden, ob im Fall vollständig sichtbarer Objekte die direkte Rekonstruktion oder die Verwendung der Sampson-Näherung exakter ist. Da letztere die horizontalen Fehler der Bildpunkte und damit die Abweichungen entlang der z-Achse nicht korrigieren kann, wird zugunsten der Laufzeit auf die Sampson-Näherung verzichtet.

Wenn bei einer korrekten Korrespondenz aufgrund von Störungen die Position ${}^0\mathbf{p}$ nicht genau rekonstruiert werden kann, so zeigen die Experimente, dass der größte Fehler im Allgemeinen entlang der z-Achse auftritt. Daher können die beiden anderen Koordinaten verwendet werden, um mit dem Winkel $\alpha = \text{atan2}({}^0p_x, {}^0p_y)$ eine verwertbare Information zu erhalten. Mit diesem Winkel kann entschieden werden, in welcher Richtung sich das Objekt befindet, obwohl keine Aussage getroffen werden kann, wie weit sich der Roboter in die entsprechende Richtung drehen muss.

Falsche Korrespondenzen und Rekonstruktionsfehler bei teilweise verdeckten Objekten lassen sich durch die zeitliche Verfolgung der Objekte über mehrere Bilder verringern. Hierfür wird angenommen, dass sich die Geschwindigkeit und Richtung der Objekte in aufeinanderfolgenden Bildern nur wenig ändert. Die Algorithmen der Objektverfolgung müssen dann Informationen über die Eigenbewegung des mobilen Robotersystems verarbeiten, damit diese in den Bildern kompensiert werden kann. Allerdings sind für solche Verfahren neben mehr Rechenleistung auch verlässliche Zeitstempel der Bilder nötig.

6.3.3 Fazit

Die Entfernungsmessungen in Tabelle 6.4 sind mit dem Gestell für Nahsicht durchgeführt worden, das nach Tabelle 4.2 für Entfernungen bis 360,8 mm konzipiert ist. Damit ist gezeigt, dass mit den in dieser Diplomarbeit entwickelten merkmalsbasierten Algorithmen zur Korrespondenzbestimmung ohne Erhöhung der algorithmischen Komplexität Stereokamerasysteme mit einem Konvergenzwinkel $\theta \neq 0$ weiter als bis zum Schnittpunkt der optischen Achsen sehen können. Dies ist mit dem flächenbasierten Verfahren aus Kapitel 5.4.1 nicht möglich.

Die Genauigkeit des in dieser Arbeit entwickelten Stereokamerasystems ist sehr hoch, wie die Messungen zeigen. Bei Objektentfernungen bis zu drei Metern unterscheiden sich die Rekonstruktionen nur um wenige Zentimeter von den gemessenen Positionen. Ein Vergleich der Genauigkeit mit bestehenden Systemen ist nicht möglich, da für diese keine Messungen angegebenen sind.

Aufgrund der hardwarebedingt hohen Laufzeit der Positionsberechnung ist ein praktischer Einsatz auf dem humanoiden Roboter „Mr. DD“ zum Beispiel im *RoboCup* kaum möglich. Bei einer Verarbeitungsgeschwindigkeit von etwa einer Sekunde pro Stereobildpaar ist eine zuverlässige Bestimmung der Position eines bewegten Objektes nur eingeschränkt realisierbar.

Kapitel 7

Zusammenfassung und Ausblick

7.1 Zusammenfassung

In dieser Diplomarbeit ist ein Stereokamerasystem entwickelt worden, dass sich für verschiedenste Einsatzmöglichkeiten autonomer mobiler Roboter eignet. So kann es im *RoboCup* eingesetzt werden, um Objekte an ihrer Farbe zu erkennen und ihre Position relativ zu einem Referenzkoordinatensystem zu bestimmen. Darüber hinaus kann es sehr nahe Objekte für die direkte Interaktion oder weit entfernte Objekte für die Navigation beispielsweise in einem Bürogebäude erkennen.

Weiterhin sind verschiedene Verfahren implementiert und getestet worden, um das Stereokamerasystem so zu kalibrieren, dass eine möglichst genaue metrische Rekonstruktion der Objektpositionen möglich ist. Messungen haben gezeigt, dass mit der verwendeten Kalibrierungsmethode eine Rekonstruktion mit Fehlern von wenigen Zentimetern bei einem Objektabstand bis zu drei Meter möglich ist.

Das Bildverstehen ist mit Algorithmen für die Farbsegmentierung sowie Korrespondenzbestimmung und Rekonstruktion realisiert worden. Für diese Verfahren sind in dieser Arbeit Schnittstellen definiert worden, so dass sie sehr einfach austauschbar sind. Dies ist sogar zur Laufzeit möglich, womit das Robotersystem auf unterschiedliche Situationen reagieren und die Stereobildpaare entsprechend verarbeiten kann. Dabei sind die Methoden auf dem System des humanoiden Roboters „Mr. DD“ der *Darmstadt Dribblers* getestet und für dieses optimiert worden, wodurch die Geschwindigkeit der Verarbeitung auf diesem System erheblich gesteigert wird. Allerdings ist durch die derzeitige Hardware des Humanoiden die Verarbeitungsgeschwindigkeit bezüglich einiger notwendiger Operationen stark eingeschränkt, wodurch der Einsatz bei bewegten Objekten nur bedingt möglich ist.

Darüber hinaus wird durch Verwendung des Frameworks *RoboFrame* ein hohes Maß an Erweiterbarkeit, Modularität und Plattformunabhängigkeit sichergestellt. So können die entwickelten Module einfach zu neuen Anwendungen kombiniert und außer für den Humanoiden „Mr. DD“ auch für andere Systeme verwendet werden. Mit *RoboFrame* sind schließlich grafische Oberflächen entwickelt worden, mit denen das Stereokamerasystem kalibriert und konfiguriert werden kann. Auch die Überwachung der Module und damit eine erleichterte Fehlersuche ist so möglich.

7.2 Erweiterungsmöglichkeiten

Mit den verwendeten Webcams ist eine Synchronisation nicht möglich, wodurch ein Stereobildpaar nicht zwei Abbildungen einer Szene zum selben Zeitpunkt enthält. Daher weisen die Rekonstruktionen teilweise große Fehler auf, wenn sich die betrachteten Objekte oder der Roboter bewegen. Hier können Kameras mit einer entsprechenden Steuerung die Qualität der Rekonstruktion ohne Mehraufwand für Verarbeitung steigern. Eine weitere Möglichkeit ist die Anpassung des Treibers, so dass dieser die Bilder direkt zeitstempelt, wenn sie aufgenommen werden. Damit können Objektpositionen in den Bildern interpoliert und korrekt rekonstruiert werden.

Wechselnde Lichtverhältnisse haben eine verfälschte Farbwiedergabe der betrachteten Objekte in den Bildern zur Folge. Dadurch ist eine korrekte Farbsegmentierung erst nach einer Anpassung der Konfigurationen von Kamera und Algorithmus möglich. Momentan werden die Parameter der Kameras manuell so eingestellt, dass sie möglichst ähnliche Bilder liefern. Auch die Farbbereiche für die Segmentierungen werden halbautomatisch angepasst. Für diese beiden Vorgänge sind automatische Verfahren sinnvoll, die eine Umstellung auf andere Beleuchtungen beschleunigen und die entsprechenden Parameter optimal wählen.

Werden mehrere Objekte betrachtet, so kommt es häufig zu Verdeckungen von Teilen eines Objektes oder eines ganzen Objektes in einem der beiden Bilder. Auch durch den kleinen Blickwinkel der Kameras ist ein Objekt oft nur in einem Bild zu sehen. In solchen Situationen kann die Rekonstruktion große Fehler aufweisen oder unmöglich sein. In all diesen Fällen schafft eine Objektverfolgung über mehrere Stereobildpaare Abhilfe.

Eine generelle Möglichkeit, die Software zu verbessern, besteht natürlich in weiteren Optimierungen für das verwendete Robotersystem. Auch neue, effizientere Algorithmen können entwickelt werden, was zu einer Leistungssteigerung führt. Durch die gewonnene Zeit kann der Roboter entweder schneller reagieren oder um fortgeschrittene Fähigkeiten erweitert werden.

Anhang A

Technische Daten

A.1 Entwicklungssystem

Für die Software-Entwicklung und einige der Laufzeitmessungen wird ein Notebook verwendet, dessen technische Daten in Tabelle A.1 aufgeführt sind.

Merkmal	Eigenschaften
Bezeichnung	Baycom Worldbook
Prozessor	Intel Celeron, Coppermine, 700 MHz
Arbeitsspeicher	240 MB
Architektur	IA32
Linux-Distribution	SuSE 9.0
Linux-Kernel	2.4.21-286-default
USB-Treiber	usb-ohci
PWC-Treiber	8.11

Tabelle A.1: Technische Daten des Entwicklungssystems.

A.2 Humanoider Roboter „Mr. DD“

Das Gestell des Stereokamerasystems wird für den humanoiden autonomen Roboter „Mr. DD“ [55] aus Abbildung 1.1 entwickelt. Dieser Roboter ist eine Spezialanfertigung der Firma iXs Research Corporation [21] für das Fachgebiet Simulation und Systemoptimierung des Fachbereichs Informatik der Technischen Universität Darmstadt. Auch bei der Software-Entwicklung werden die Besonderheiten dieses Systems berücksichtigt, wie zum Beispiel die

fehlende Hardwareunterstützung für Fließkommaberechnungen. Die technischen Daten des Roboters sind in Tabelle A.2 aufgelistet.

Merkmal	Eigenschaften
Prozessor	NEC VR4181A, 133 MHz
Arbeitsspeicher	32 MB
Ramdisk	8 MB
Architektur	Mips
Linux-Distribution	Red Hat 7.1
Linux-Kernel	2.4.19-rc1
USB-Treiber	usb-ohci
PWC-Treiber	8.6
Netzwerk	WLAN, LAN
Stromversorgung	Batterien
Höhe	65 cm
Breite	31 cm
Gewicht	4,8 kg
Freiheitsgrade	6 pro Bein, 4 pro Arm, 2 in der Hüfte, 2 im Genick
Sensoren	24 Kodierer für Gelenkwinkel
	4 Abstandssensoren in den Schultern
	3 Kraftsensoren pro Fuß

Tabelle A.2: Technische Daten des humanoiden Roboters „Mr. DD“.

Anhang B

Systemleistung

Zu Optimierungszwecken sind auf dem Entwicklungsrechner und dem humanoiden Roboter (siehe auch Anhang A.2) einige Tests durchgeführt worden, um aufwändige Operationen zu identifizieren und zu vermeiden. In Tabelle B.1 sind die Ergebnisse der Tests aufgeführt. Die angegebenen Werte sind Operationen pro Sekunde. Die Spaltenüberschriften „Single“ und „Multi“ beziehen sich auf die Verwendung eines oder mehrerer Threads, da *RoboFrame* [44] mit Threads kompiliert wird.

Einige der betrachteten Operationen sind nicht selbsterklärend und werden näher erläutert.

a. Es werden natürliche Zahlen an eine Liste bzw. einen Vektor angehängt.

b. Es wird über eine Liste bzw. einen Vektor iteriert.

c. Es werden natürliche Zahlen an einen Vektor angehängt, nachdem in diesem genügend Speicher reserviert worden ist, so dass er sich nicht neu allokieren muss.

d. Es werden Elemente aus einem Array gelesen.

e. Es werden Elemente in ein Array geschrieben.

f. Es wird mit $\sqrt{a^2 + b^2}$ der Satz des Pythagoras angewendet.

g. Es wird der Satz des Pythagoras mit der Wurzelfunktion aber ohne auslöschenden Überlauf und Unterlauf berechnet. Hierfür wird maximal eine Wurzel der Form $\sqrt{1 + c^2}$ berechnet, wobei $0 \leq c \leq 1$ gilt.

h. Wie im Fall g wird der Satz des Pythagoras berechnet. Die Wurzel wird jedoch durch eine Näherung mit Ganzzahloperationen berechnet. Details dieser Berechnung sind in Kapitel 5.4.3 beschrieben.

Operation	Entwicklungsrechner		Humanoider Roboter	
	Single	Multi	Single	Multi
list<long>::push[a]	$1{,}81 \cdot 10^6$	$1{,}05 \cdot 10^6$	$5{,}56 \cdot 10^5$	$1{,}98 \cdot 10^4$
list<long>::iterator[b]	$3{,}05 \cdot 10^6$	$3{,}06 \cdot 10^6$	$2{,}58 \cdot 10^6$	$2{,}58 \cdot 10^6$
vector<long>::push[a]	$4{,}47 \cdot 10^6$	$4{,}42 \cdot 10^6$	$1{,}01 \cdot 10^6$	$9{,}93 \cdot 10^5$
vector<long>::push[c]	$8{,}83 \cdot 10^6$	$8{,}84 \cdot 10^6$	$2{,}56 \cdot 10^6$	$2{,}54 \cdot 10^6$
vector<long>::iterator[b]	$7{,}34 \cdot 10^6$	$7{,}40 \cdot 10^6$	$5{,}09 \cdot 10^7$	$5{,}07 \cdot 10^7$
long +	$2{,}06 \cdot 10^8$	$2{,}07 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}44 \cdot 10^7$
long −	$2{,}08 \cdot 10^8$	$2{,}07 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}44 \cdot 10^7$
long ∗	$2{,}08 \cdot 10^8$	$2{,}08 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}44 \cdot 10^7$
long /	$2{,}08 \cdot 10^8$	$2{,}04 \cdot 10^8$	$9{,}32 \cdot 10^6$	$9{,}32 \cdot 10^6$
long >	$2{,}07 \cdot 10^8$	$2{,}08 \cdot 10^8$	$6{,}45 \cdot 10^7$	$6{,}44 \cdot 10^7$
double +	$2{,}08 \cdot 10^8$	$2{,}08 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}45 \cdot 10^7$
double −	$2{,}08 \cdot 10^8$	$2{,}08 \cdot 10^8$	$6{,}45 \cdot 10^7$	$6{,}44 \cdot 10^7$
double ∗	$2{,}07 \cdot 10^8$	$2{,}08 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}45 \cdot 10^7$
double /	$2{,}07 \cdot 10^8$	$2{,}07 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}44 \cdot 10^7$
double >	$2{,}08 \cdot 10^8$	$2{,}08 \cdot 10^8$	$6{,}45 \cdot 10^7$	$6{,}45 \cdot 10^7$
float +	$2{,}07 \cdot 10^8$	$2{,}07 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}45 \cdot 10^7$
float −	$2{,}04 \cdot 10^8$	$2{,}07 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}44 \cdot 10^7$
float ∗	$2{,}06 \cdot 10^8$	$2{,}07 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}45 \cdot 10^7$
float /	$2{,}07 \cdot 10^8$	$2{,}07 \cdot 10^8$	$6{,}44 \cdot 10^7$	$6{,}43 \cdot 10^7$
float >	$2{,}07 \cdot 10^8$	$2{,}07 \cdot 10^8$	$6{,}43 \cdot 10^7$	$6{,}44 \cdot 10^7$
get long[][d]	$3{,}70 \cdot 10^8$	$4{,}28 \cdot 10^8$	$1{,}28 \cdot 10^8$	$1{,}20 \cdot 10^8$
get double[][d]	$4{,}27 \cdot 10^8$	$4{,}30 \cdot 10^8$	$1{,}21 \cdot 10^8$	$1{,}22 \cdot 10^8$
get float[][d]	$4{,}24 \cdot 10^8$	$3{,}38 \cdot 10^8$	$1{,}20 \cdot 10^8$	$1{,}20 \cdot 10^8$
set long[][e]	$1{,}11 \cdot 10^7$	$1{,}10 \cdot 10^7$	$3{,}16 \cdot 10^6$	$3{,}15 \cdot 10^6$
set double[][e]	$6{,}84 \cdot 10^6$	$6{,}89 \cdot 10^6$	$2{,}81 \cdot 10^4$	$2{,}81 \cdot 10^4$
set float[][e]	$1{,}09 \cdot 10^7$	$1{,}08 \cdot 10^7$	$3{,}14 \cdot 10^4$	$3{,}10 \cdot 10^4$
sqrt()[f]	$7{,}01 \cdot 10^5$	$6{,}96 \cdot 10^5$	$9{,}55 \cdot 10^2$	$9{,}49 \cdot 10^2$
Math::pythagoras()[g]	$6{,}33 \cdot 10^5$	$6{,}33 \cdot 10^5$	$8{,}15 \cdot 10^2$	$8{,}13 \cdot 10^2$
Math::pythagoras()[h]	$2{,}76 \cdot 10^5$	$2{,}80 \cdot 10^5$	$2{,}50 \cdot 10^3$	$2{,}50 \cdot 10^3$
ilogb()[i]	$7{,}85 \cdot 10^5$	$7{,}56 \cdot 10^5$	$1{,}67 \cdot 10^4$	$1{,}40 \cdot 10^4$
Math::msb()[j]	$7{,}82 \cdot 10^5$	$7{,}82 \cdot 10^5$	$1{,}39 \cdot 10^5$	$1{,}30 \cdot 10^5$

Tabelle B.1: Systemleistung in Operationen pro Sekunde.

i. Es wird das höchstwertige Bit mit dem Logarithmus zur Basis 2 berechnet.

j. Es wird das höchstwertige Bit mit bitweisen Operationen und einer Lookup-Tabelle berechnet. Details dieser Funktion finden sich in Kapitel 5.3.3.

Anhand der Messergebnisse erkennt man einige Besonderheiten des humanoiden Roboters, die zu ineffizienter Software führen, wenn die entsprechenden Operationen nicht vermieden werden können. Bezüglich der Container-Klassen `list` und `vector` wird deutlich, dass die erstere langsamer ist, wenn Elemente am Ende eingefügt werden. Dieser Unterschied tritt noch deutlicher hervor, wenn Threads verwendet werden. Außerdem kann man schneller durch einen `vector` iterieren. Eine weitere Verbesserung ist möglich, wenn in dem `vector` vor dem Füllen genügend Speicher reserviert wird. Dadurch wird die Reallokation zum Vergrößern des Speicherplatzes gespart.

Obwohl der Humanoide keine Hardwareunterstützung für Fließkommazahlen hat, sind die vier Grundrechenarten und ein Vergleich genauso schnell wie die entsprechenden Operationen für Ganzzahlen. Tatsächlich ist es so, dass die Ganzzahldivision sehr viel langsamer ist, als die anderen aufgeführten Operationen. Diese Besonderheit tritt auf dem Entwicklungssystem nicht auf, bei dem keine nennenswerten Geschwindigkeitsunterschiede zu beobachten sind.

Eine weitere Erkenntnis betrifft die Zugriffszeiten auf Arrays. Während sehr schnell aus diesen gelesen werden kann, ist der schreibende Zugriff bei Ganzzahlen etwa 40-mal langsamer und bei Fließkommazahlen sogar 4000-mal langsamer. Damit sind die elementare Rechenoperationen für Fließkommazahlen rund 2000-mal schneller als die Speicherung eines Ergebnisses in einem Array.

Komplexere mathematische Funktionen, wie die Quadratwurzel und der Logarithmus, sind auf beiden Systemen viel langsamer als die Elementaroperationen. Durch die fehlende Unterstützung für Fließkommazahlen auf dem Humanoiden tritt dieser Unterschied hier deutlicher hervor. Der benötigte Spezialfall der Wurzelberechnung kann, wie in Kapitel 5.4.3 skizziert, mit Ganzzahloperationen näherungsweise berechnet werden. Dieses Näherungsverfahren ist auf dem Entwicklungssystem zwar langsamer aber auf dem Humanoiden etwa 3-mal schneller als die Wurzelfunktion. Die Berechnung des höchstwertigen Bits einer Zahl (siehe Kapitel 5.3.3) kann ohne den Logarithmus etwa 9-mal schneller ausgeführt werden.

Anhang C

Konfiguration von udev

Auf dem verwendeten Entwicklungsrechner und dem Humanoiden ist der Linux-Kernel 2.4.21 bzw. 2.4.19 und damit kein udev [25] installiert. Auf einigen Rechnern des Robotik-Pools ist die nötige Software vorhanden, jedoch sind die Konfigurationsdateien nur lesbar. Aus diesem Grund wird zu Testzwecken leihweise ein anderes Notebook (Tabelle C.1) verwendet. Auf diesem sind die Komponenten udev und sysfs bereits vorhanden und das PWC-Modul von [50] wird kompiliert und installiert.

Merkmal	Eigenschaften
Bezeichnung	Asus S5600
Prozessor	Intel Pentium M, Dothan-Core, 1,6 GHz
Arbeitsspeicher	512 MB
Architektur	IA32
Linux-Distribution	Gentoo
Linux-Kernel	2.6.11-gentoo-r6
USB-Treiber	usb-uhci
PWC-Treiber	10.0.7

Tabelle C.1: Technische Daten des Notebooks für den Test von udev.

Nach dem Anschließen einer Kamera liefert der Befehl `udevinfo -a -p ‘udevinfo -q path -n /dev/video0‘` die Informationen in Listing C.1.

Listing C.1: Ausgabe von udevinfo zur linken Kamera.

```
looking at class device '/sys/class/video4linux/
   video0':
  SYSFS{dev}="81:0"
```

```
    SYSFS{name}="Philips 740 webcam"

follow the class device's "device"
  looking at the device chain at '/sys/devices/
     pci0000:00/0000:00:1d.0/usb1/1-2':
    BUS="usb"
    ID="1-2"
    SYSFS{bConfigurationValue}="1"
    SYSFS{bDeviceClass}="00"
    SYSFS{bDeviceProtocol}="00"
    SYSFS{bDeviceSubClass}="00"
    SYSFS{bMaxPower}="500mA"
    SYSFS{bNumConfigurations}="1"
    SYSFS{bNumInterfaces}=" 3"
    SYSFS{bcdDevice}="0003"
    SYSFS{bmAttributes}="a0"
    SYSFS{detach_state}="0"
    SYSFS{devnum}="2"
    SYSFS{idProduct}="0311"
    SYSFS{idVendor}="0471"
    SYSFS{maxchild}="0"
    SYSFS{serial}="01690000A5000000"
    SYSFS{speed}="12"
    SYSFS{version}=" 1.10"

  looking at the device chain at '/sys/devices/
     pci0000:00/0000:00:1d.0/usb1':
    BUS="usb"
    ID="usb1"
    SYSFS{bConfigurationValue}="1"
    SYSFS{bDeviceClass}="09"
    SYSFS{bDeviceProtocol}="00"
    SYSFS{bDeviceSubClass}="00"
    SYSFS{bMaxPower}="  0mA"
    SYSFS{bNumConfigurations}="1"
    SYSFS{bNumInterfaces}=" 1"
    SYSFS{bcdDevice}="0206"
    SYSFS{bmAttributes}="c0"
    SYSFS{detach_state}="0"
    SYSFS{devnum}="1"
    SYSFS{idProduct}="0000"
    SYSFS{idVendor}="0000"
```

```
    SYSFS{manufacturer}="Linux 2.6.11-gentoo-r6
       uhci_hcd"
    SYSFS{maxchild}="2"
    SYSFS{product}="Intel Corp. 82801DB/DBL/DBM (
       ICH4/ICH4-L/ICH4-M) USB UHCI Controller #1"
    SYSFS{serial}="0000:00:1d.0"
    SYSFS{speed}="12"
    SYSFS{version}=" 1.10"

  looking at the device chain at '/sys/devices/
     pci0000:00/0000:00:1d.0':
    BUS="pci"
    ID="0000:00:1d.0"
    SYSFS{class}="0x0c0300"
    SYSFS{detach_state}="0"
    SYSFS{device}="0x24c2"
    SYSFS{irq}="16"
    SYSFS{local_cpus}="1"
    SYSFS{subsystem_device}="0x1828"
    SYSFS{subsystem_vendor}="0x1043"
    SYSFS{vendor}="0x8086"

  looking at the device chain at '/sys/devices/
     pci0000:00':
    BUS=""
    ID="pci0000:00"
    SYSFS{detach_state}="0"
```

Nach Entfernen der Kamera und Anschließen der zweiten Kamera an einen anderen USB-Steckplatz erhält man die Informationen in Listing C.2.

Listing C.2: Ausgabe von udevinfo zur rechten Kamera.

```
  looking at class device '/sys/class/video4linux/
     video0':
    SYSFS{dev}="81:0"
    SYSFS{name}="Philips 740 webcam"

follow the class device's "device"
  looking at the device chain at '/sys/devices/
     pci0000:00/0000:00:1d.1/usb2/2-1':
    BUS="usb"
    ID="2-1"
    SYSFS{bConfigurationValue}="1"
```

```
    SYSFS{bDeviceClass}="00"
    SYSFS{bDeviceProtocol}="00"
    SYSFS{bDeviceSubClass}="00"
    SYSFS{bMaxPower}="500mA"
    SYSFS{bNumConfigurations}="1"
    SYSFS{bNumInterfaces}=" 3"
    SYSFS{bcdDevice}="0003"
    SYSFS{bmAttributes}="a0"
    SYSFS{detach_state}="0"
    SYSFS{devnum}="3"
    SYSFS{idProduct}="0311"
    SYSFS{idVendor}="0471"
    SYSFS{maxchild}="0"
    SYSFS{serial}="01690000A5000000"
    SYSFS{speed}="12"
    SYSFS{version}=" 1.10"

  looking at the device chain at '/sys/devices/
     pci0000:00/0000:00:1d.1/usb2':
    BUS="usb"
    ID="usb2"
    SYSFS{bConfigurationValue}="1"
    SYSFS{bDeviceClass}="09"
    SYSFS{bDeviceProtocol}="00"
    SYSFS{bDeviceSubClass}="00"
    SYSFS{bMaxPower}="  0mA"
    SYSFS{bNumConfigurations}="1"
    SYSFS{bNumInterfaces}=" 1"
    SYSFS{bcdDevice}="0206"
    SYSFS{bmAttributes}="c0"
    SYSFS{detach_state}="0"
    SYSFS{devnum}="1"
    SYSFS{idProduct}="0000"
    SYSFS{idVendor}="0000"
    SYSFS{manufacturer}="Linux 2.6.11-gentoo-r6
       uhci_hcd"
    SYSFS{maxchild}="2"
    SYSFS{product}="Intel Corp. 82801DB/DBL/DBM (
       ICH4/ICH4-L/ICH4-M) USB UHCI Controller #2"
    SYSFS{serial}="0000:00:1d.1"
    SYSFS{speed}="12"
    SYSFS{version}=" 1.10"
```

```
  looking at the device chain at '/sys/devices/
     pci0000:00/0000:00:1d.1':
    BUS="pci"
    ID="0000:00:1d.1"
    SYSFS{class}="0x0c0300"
    SYSFS{detach_state}="0"
    SYSFS{device}="0x24c4"
    SYSFS{irq}="19"
    SYSFS{local_cpus}="1"
    SYSFS{subsystem_device}="0x1828"
    SYSFS{subsystem_vendor}="0x1043"
    SYSFS{vendor}="0x8086"

  looking at the device chain at '/sys/devices/
     pci0000:00':
    BUS=""
    ID="pci0000:00"
    SYSFS{detach_state}="0"
```

Diese Daten sind bis auf die verwendeten USB-Ports und deren Ressourcen identisch. Es sind noch zwei Regeln nötig, die vor den normalen udev-Regeln verarbeitet werden. Auf dem Testsystem liegen die Regeln in dem Verzeichnis `/etc/udev/rules.d/`, in dem bereits eine Datei `50-udev.rules` mit den Standard-Regeln vorhanden ist. Da die Dateien in lexikografischer Reihenfolge verarbeitet werden, wird eine neue Datei `10-local.rules` hinzugefügt, welche die Regeln aus Listing C.3 enthält.

Listing C.3: Regeln von udev zur Identifizierung der Kameras.

```
BUS="usb", PLACE="1-2", KERNEL="video?", NAME="%k",
   SYMLINK="lefteye"
BUS="usb", PLACE="2-1", KERNEL="video?", NAME="%k",
   SYMLINK="righteye"
```

Diese Regeln besagen, dass ein USB-Gerät am Port 2 des Hubs 1 (1-2) bzw. Port 1 des Hubs 2 (2-1) und dem Standard-Gerätenamen `video?` (`video0` bis `video9`) angeschlossen ist. Wird ein solches Gerät verwendet, so erhält es seinen Standard-Namen und eine symbolische Verknüpfung mit dem Namen `lefteye` bzw. `righteye`. Jetzt können über die Namen `/dev/lefteye` bzw. `/dev/righteye` USB-Video-Geräte an den USB-Steckplätzen eindeutig identifiziert werden.

Anhang D

Verwendung von *RoboFrame*

Unter Verwendung des Frameworks *RoboFrame* [44] sind im Rahmen dieser Diplomarbeit die Module `CalibratorModule` und `StereovisionModule` sowie die Dialoge `Calibrator` und `Stereovision` entwickelt worden. Wird eines der Module auf dem Humanoiden verwendet, können über den entsprechenden Dialog Information mit dem Modul ausgetauscht und Daten visualisiert werden. Darüber hinaus gibt es die zwei ausführbaren Applikationen `CalibratorApp` und `StereovisionApp`, die das jeweilige Modul enthalten und eine grafische Oberfläche, welche die beiden Dialoge enthält. Zur Ausführung der Applikationen werden im gleichen Verzeichnis die Konfigurationsdateien benötigt, in denen die Einstellungen für das Stereokamerasystem festgehalten sind. Die genannten Komponenten sind auf der beigelegten CD verfügbar und können unter Linux mit den vorhandenen Build-Skripten erstellt werden. Die grafische Oberfläche kann zusätzlich unter Windows32 mit einem Projekt für Visual Studio erzeugt werden.

Die Daten, die von den Modulen und Dialogen verschickt und empfangen werden, sind in der Referenz zu den entsprechenden Klassen auf der CD detailliert beschrieben.

Mit den Build-Skripten werden unter Linux die Bibliotheken `libcalibratormodule`, `libcalibratorgui`, `libstereovisionmodule`, `libstereovisiongui`, `libstereocommon` und `libminpack` erstellt. Wenn neue Applikationen oder grafische Oberflächen erstellt werden, können einfach diese Bibliotheken verwendet werden, ohne die einzelnen Komponenten neu zu kompilieren. Falls die Bibliotheken nicht verwendet werden können, müssen die Komponenten kompiliert werden, wobei die Abhängigkeiten den Build-Skripten entnommen werden können. Einige Teile des Codes werden unterschiedlich kompiliert, wenn bestimmte Symbole definiert sind. Diese Symbole sind mit einer kurzen Beschreibung in Tabelle D.1 aufgeführt.

Werden die Module in neuen Anwendungen verwendet, so werden sie einem

Symbol	Beschreibung
DEBUG	ermöglicht die Ausgabe zusätzlicher Informationen über den Kommandozeilenparameter -d
_WIN32	ermöglicht die Verwendung mathematischer Konstanten unter Windows32
SYSTEM_WINCE	deaktiviert die Verarbeitung der Kommandozeilenargumente unter WindowsCE
SYSTEM_MIPSEL	aktiviert optimierten Code für den Humanoiden und leitet Exceptions in Log um
HAVE_FORTRAN	aktiviert die vollständige Fortran-Minpack-Bibliothek (andernfalls C++-Portierung einer Teilmenge der Funktionen)
PACKAGE_VERSION	wird während der Verarbeitung der Kommandozeilenargumente verwendet

Tabelle D.1: Verwendete Symbole während der Kompilierung.

Prozess im *RoboFrame* hinzugefügt. Dieser muss regelmäßig aufgerufen werden, damit Stereobildpaare von den Kameras geholt und verarbeitet werden. In Listing D.1 wird die Methode `init()` skizziert, mit der ein Modul in der Applikation `NewApplication` eingebunden wird.

Listing D.1: Verwendung eines Moduls in einer Applikation.

```
void NewApplication::init()
{
  Process *process = createProcess();
  process->setTimerMilliseconds(n);
  process->addModule(new CalibratorModule(config));
}
```

Dabei ist `CalibratorModule` das Modul und `n` eine Zeit in Millisekunden. Diese hängt von der gewählten Framerate der Kameras ab und sollte etwas geringer sein, als die Zeit zwischen zwei Bildern. Werden beispielsweise 30 Bilder pro Sekunde erzeugt, so wird `n=30` und bei 5 Bildern pro Sekunde `n=195` verwendet.

Anhang E

Transformation der Doppelebene

In Kapitel 5.2.4.3 wird die räumliche Anordnung einer Doppelebene mit Koordinatensystem S_m bezüglich eines Referenzkoordinatensystems S_0 benötigt. Mit einer entsprechenden Transformation kann die beschriebene metrische Kalibrierung relativ zu S_0 ausgeführt werden.

Abbildung E.1 zeigt eine mögliche Anordnung der Doppelebene links und des Stereokameragestells rechts auf einer Tischplatte, deren Kanten als Bezugspunkte verwendet werden. Von dem Gestell, dessen hinterer Rand direkt an der Tischkante liegt, ist nur die Grundfläche der Halterung eingezeichnet. Weiterhin sind die Orientierungen der Koordinatensysteme und die benötigten Maße ersichtlich. Dabei befindet sich der Ursprung von S_m in der vorderen, unteren Ecke der Doppelebene und von S_0 in der Mitte der Grundfläche auf der Tischplatte. Die Lage von S_0 ist auch in Abbildung 4.6(d) ersichtlich.

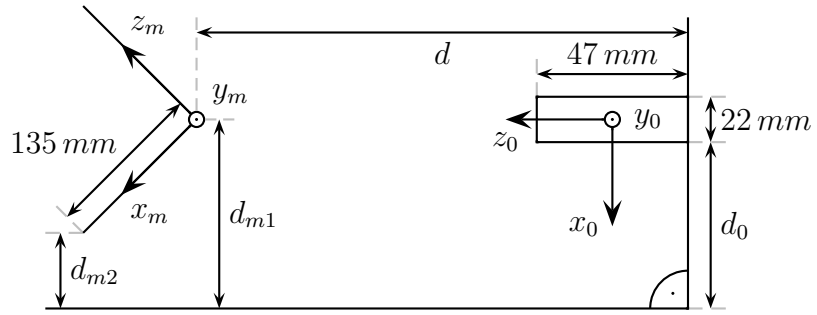


Abbildung E.1: Anordnung der Doppelebene und des Kameragestells zur metrischen Kalibrierung.

Die Entfernung d bezeichnet den Abstand von S_m rechtwinklig zu der Tischkante an der das Kameragestell steht. Der Wert d_0 enthält die Distanz des Gestells zur zweiten Tischkante. Bezüglich dieser Kante sind auch die Abstände d_{m1} von S_m und d_{m2} der nächsten Ecke definiert. Sind alle Entfernungen ausgemessen, ist die räumliche Anordnung der Doppelebene für die Eingabe im Dialog (Abbildung 5.8) gegeben durch die Eulerwinkel

$$\begin{pmatrix} 0 \\ -\cos^{-1}\frac{d_{m1}-d_{m2}}{135\,mm} \\ 0 \end{pmatrix} \tag{E.1}$$

und den Translationsvektor

$$\begin{pmatrix} 11\,mm + d_0 - d_{m1} \\ 0 \\ d - 23{,}5\,mm \end{pmatrix}. \tag{E.2}$$

Anhang F

Inhalt der CD

Alle elektronisch verfügbaren Daten der vorliegenden Diplomarbeit sind auf einer beigelegten CD verfügbar. Um das Auffinden bestimmter Informationen oder Dateien zu erleichtern, wird die oberste Verzeichnisebene der CD im Folgenden beschrieben.

Bibliotheken enthält die Bibliothek Minpack für nichtlineare Optimierung und die Camera Calibration Toolbox for Matlab.

Bilder enthält alle Bilder, die in dieser Diplomarbeit und in den dazugehörigen Präsentationen verwendet werden. Darüber hinaus sind hier Vorlagen für den Aufdruck der zur Kalibrierung verwendeten Doppelebene zu finden.

Daten enthält eine Beispielkonfiguration des Programmes udev zur Identifizierung der Kameras, verschiedene gleichgerichtete Stereobildpaare zum Testen der Software, Ergebnisse der durchgeführten Kalibrierung in Matlab sowie einige Log-Dateien der implementierten Module, die mit dem `LogRecorder` von *RoboFrame* verarbeitet werden können.

Dokumentation enthält die elektronische Form dieser Diplomarbeit, die Folien des Zwischen- und Abschlussberichtes sowie die generierte API-Referenz des Quellcodes.

Programme enthält die ausführbaren Applikationen und grafischen Oberflächen für verschiedene Systeme sowie die generierten Bibliotheken, um Module und Dialoge einfach in neuen Anwendungen verwenden zu können. Keins der Kompilate enthält Debug-Informationen, damit die Ausführung beschleunigt und die Dateigrößen reduziert werden.

PWC enthält den Quellcode des PWC-Moduls und seine Dokumentation sowie die Referenz von Video4Linux2.

Quellcode enthält die Quelldateien der in dieser Arbeit implementierten Software sowie die von *RoboFrame* in der zum Abschluss dieser Diplomarbeit aktuellen Version, so dass alle Dateien zum Kompilieren der Software vorhanden sind. Diese Dateien sind auch in dem Subversion-Repository [56] des Fachgebiets Simulation und Systemoptimierung in der Revision 1000 abgelegt.

Referenzen enthält die zitierten und elektronisch verfügbaren Quellen, die nach ihrem Inhalt sortiert in entsprechenden Unterverzeichnissen zu finden sind.

Anhang G

Singulärwertzerlegung

Die Singulärwertzerlegung (Singular Value Decomposition, SVD) ist eine Zerlegung beliebiger Matrizen in Diagonal- und Orthogonalmatrizen, mit deren Hilfe verschiedene numerische Probleme gelöst werden. Das Verfahren wird zum Beispiel für die lineare Ausgleichsrechnung verwendet. Kapitel 7.3 von [51] und Anhang 6 von [63] enthalten weitere Informationen über die Singulärwertzerlegung. In Anhang 4.4 von [14] finden sich Verweise und Beispiele zur Berechnungskomplexität.

Mit der Singulärwertzerlegung wird eine Matrix $A \in \mathbb{R}^{m \times n}$ in eine Diagonalmatrix $D = \mathrm{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n \times n}$ und zwei orthogonale Matrizen $U \in \mathbb{R}^{m \times n}$ und $V \in \mathbb{R}^{n \times n}$ zerlegt.

$$A = U\,D\,V^T \tag{G.1}$$

Durch die Orthogonalität von U und V folgt $U^T\,U = V^T\,V = V\,V^T = E_n$.

Die Elemente der Diagonalmatrix sind nie negativ und heißen Singulärwerte d_i. Die Zerlegung kann so permutiert werden, dass $d_1 \geq d_2 \geq \ldots \geq d_n \geq 0$ gilt, was ohne Beschränkung der Allgemeinheit vorausgesetzt wird. Die Spalten von U bzw. V heißen Links- bzw. Rechtssingulärvektoren.

Folgende Eigenschaften und Problemlösungen werden in der vorliegenden Arbeit verwendet:

- Die Singulärwerte d_i geben Aufschluss über die Singularität und den Rang der Matrix A. Wenn mindestens ein Singulärwert im Bereich der Maschinengenauigkeit liegt, also praktisch 0 ist, ist die Matrix A singulär. Die Anzahl der $d_i \neq 0$ gibt den Rang an.

- Es existiert zu jeder Matrix eine Pseudoinverse $A^+ = V\,D_0^{-1}\,U^T$. Die Matrix D_0^{-1} enthält die Diagonalelemente $\frac{1}{d_i}$, wenn d_i nicht nahe der Maschinengenauigkeit ist und 0 sonst. Wenn A nicht singulär ist, gilt

$D_0^{-1} = D^{-1}$ und $A^+ A = E_n$. Ist A zusätzlich quadratisch, so gilt außerdem $A^+ = A^{-1}$.

- Die Lösung eines überbestimmten, homogenen Gleichungssystems $A\,\mathbf{x} = \mathbf{0}$ mit $A = U\,D\,V^T$ ist der Rechtssingulärvektor $\mathbf{v}_n$, der zum kleinsten Singulärwert gehört. Das Verfahren findet auch eine Lösung, wenn das Gleichungssystem inkonsistent ist, wie es bei Messdaten oder numerischen Verfahren oft der Fall ist. Die Lösung erfüllt die Bedingungen $\min_{\mathbf{x}} \|A\,\mathbf{x}\|$ und $\|\mathbf{x}\| = 1$.

- Die Lösung eines überbestimmten, inhomogenen Gleichungssystems $A\,\mathbf{x} = \mathbf{b}$ ist gegeben durch $\mathbf{x} = A^+\,\mathbf{b}$. Auch hier wird ein Fehler minimiert mit $\min_{\mathbf{x}} \|A\,\mathbf{x} - \mathbf{b}\|$ und $\|\mathbf{x}\| = 1$.

- Es können bestimmte Eigenschaften numerischer Matrizen erzwungen werden. Zu einer Matrix $A = U\,D\,V^T \in \mathbb{R}^{3\times3}$ kann eine Rotationsmatrix $R = U\,V^T$ berechnet werden, die den Fehler $\min_R \|A - R\|$ minimiert. Weiterhin existiert zu $A = U\,D\,V^T \in \mathbb{R}^{3\times3}$ eine Matrix $F = U\ \mathrm{diag}(d_1, d_2, 0)\,V^T$ mit Rang 2 und $\min_F \|A - F\|$.

Anhang H

Levenberg-Marquardt-Algorithmus

Der Levenberg-Marquardt-Algorithmus ist ein Verfahren, mit dem nichtlineare Least-Squares-Probleme gelöst werden, die vor allem in nichtlinearen Ausgleichsrechnungen auftreten. Eine ausführliche Beschreibung dieser Methode findet sich in Kapitel 10.2 von [39]. In der vorliegenden Diplomarbeit wird die Minpack-Bibliothek [35] verwendet, die mehrere Varianten des Levenberg-Marquardt-Algorithmus implementiert. Da diese in Fortran geschriebene Bibliothek nicht direkt durch C++-Programme ansprechbar ist, wird im Rahmen dieser Diplomarbeit die C++-Klasse `Minpack` entwickelt, welche die Funktionalität der Bibliothek zur Verfügung gestellt.

Ein Least-Squares-Problem ist von der Form

$$\min_{\mathbf{x}} \sum_{j=1}^{m} r_j^2(\mathbf{x}) \tag{H.1}$$

mit $\mathbf{x} \in \mathbb{R}^n$. Die Funktionen $r_j(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ sind zweimal stetig differenzierbar und werden Residuen genannt.

Ein solches Problem muss gelöst werden, wenn beispielsweise ein mathematisches Modell zu durchgeführten Messungen gesucht wird. Das Modell ist eine nichtlineare Funktion $f(\mathbf{x}; \mathbf{t}) : \mathbb{R}^{n+k} \to \mathbb{R}^l$ mit den unbekannten Parametern $\mathbf{x}$ des Modells und den bekannten Parametern $\mathbf{t} \in \mathbb{R}^k$ der Messung. Werden in $m > n$ Messungen die Messwerte $\mathbf{y}_j \in \mathbb{R}^l$ zu den Parametern $\mathbf{t}_j$ für $1 \leq j \leq m$ bestimmt, so wird der Abstand der Messwerte vom Modell $r_j(\mathbf{x}) = \|\mathbf{y}_j - f(\mathbf{x}; \mathbf{t}_j)\|$ berechnet. Setzt man diese Residuen in Gleichung (H.1) ein, so erhält man direkt das zu lösende Least-Squares-Problem

$$\min_{\mathbf{x}} \sum_{j=1}^{m} \|\mathbf{y}_j - f(\mathbf{x}; \mathbf{t}_j)\|^2 . \tag{H.2}$$

Für die Lösung eines Least-Squares-Problems mit dem Levenberg-Marquardt-Algorithmus sind die analytischen Residuen und die Jacobimatrix der Residuen

$$J(\mathbf{x}) = \begin{pmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \cdots & \frac{\partial r_m}{\partial x_n} \end{pmatrix} \in \mathbb{R}^{m \times n} \tag{H.3}$$

erforderlich. Je nach Komplexität der r_j sind die partiellen Ableitungen ebenfalls analytisch gegeben. Wenn dies nicht der Fall ist, so wird die Jacobimatrix von der Minpack-Bibliothek numerisch angenähert.

Schließlich ist der Levenberg-Marquardt-Algorithmus ein iteratives Lösungsverfahren, das einen Startwert $\mathbf{x}_0$ in der Nähe des gesuchten Minimums $\mathbf{x}$ benötigt. Die Berechnung einer solchen Näherung ist je nach Problem unterschiedlich. Eine Möglichkeit ist, das Modell $f(\mathbf{x}; \mathbf{t})$ linear zu approximieren, so dass es als Matrixprodukt

$$F\,\mathbf{x}_0 = \begin{pmatrix} \tilde{f}(\mathbf{t_1}) \\ \vdots \\ \tilde{f}(\mathbf{t_m}) \end{pmatrix} \mathbf{x}_0 = \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_m \end{pmatrix} \tag{H.4}$$

gegeben ist. Dabei gilt $\tilde{f}(\mathbf{t}_j) \in \mathbb{R}^{l \times n}$ mit $f(\mathbf{x}; \mathbf{t}_j) \approx \tilde{f}(\mathbf{t}_j)\,\mathbf{x}_0$. Das überbestimmte lineare Gleichungssystem kann zum Beispiel mit der Singulärwertzerlegung von F gelöst werden (siehe Anhang G).

Literaturverzeichnis

[1] Abadpour, Arash ; Kasaei, Shohreh: A New FPCA-Based Fast Segmentation Method for Color Images. In: *Fourth IEEE International Symposium on Signal Processing and Information Technology* (2004), S. 72–75

[2] Barrett, William A. ; Petersen, Kevin D.: Houghing the Hough: Peak Collection for Detection of Corners, Junctions and Line Intersections. In: *Computer Vision and Pattern Recognition* 2 (2001), S. 302–309

[3] Benitez, Domingo: Performance of Remote FPGA-based Coprocessors for Image-Processing Applications. In: *Euromicro Symposium on Digital System Design* (2002), S. 268–275

[4] Bibel, Wolfgang ; Hölldobler, Steffen ; Schaub, Torsten: *Wissensrepräsentation und Inferenz: eine grundlegende Einführung.* 1. Auflage. Vieweg Verlagsgesellschaft, 1993. – ISBN 3528053747

[5] Bouguet, Jean-Yves: *Camera Calibration Toolbox for Matlab.* `http://www.vision.caltech.edu/bouguetj/calib_doc`

[6] Bruce, James ; Balch, Tucker ; Veloso, Manuela: Fast and Inexpensive Color Image Segmentation for Interactive Robots. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* 3 (2000), S. 2061–2066

[7] Chen, Chichyang ; Zheng, Yuan F.: Passive and Active Stereo Vision for Smooth Surface Detection of Deformed Plates. In: *IEEE Transactions on Industrial Electronics* 42 (1995), Nr. 3, S. 300–306

[8] Demtröder, Wolfgang: *Laserspektroskopie - Grundlagen und Techniken.* 4. Auflage. Springer Verlag, 2004. – ISBN 3540642196

[9] Dudek, Gregory ; Jenkin, Michael: *Computational Principles of Mobile Robots.* 1st edition. Cambridge Univerity Press, 2000. – ISBN 0521568765

[10] Franke, Uwe ; Joos, Armin: Real-time Stereo Vision for Urban Traffic Scene Understanding. In: *IEEE Intelligent Vehicles Symposium* (2000), S. 273–278

[11] Fu, King S. ; Gonzalez, Rafael C. ; Lee, C. S. G.: *Robotics: Control, Sensing, Vision, and Intelligence*. 1st edition. McGraw-Hill, 1987. – ISBN 0070226253

[12] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Design Patterns: Elements of Reusable Object-Oriented Software*. 1st edition. Addison-Wesley, 1995. – ISBN 0201633612

[13] Goulermas, John Y. ; Liatsis, Panos: Feature-based Stereo matching via coevolution of epipolar subproblems. In: *7th International Conference on Image Processing And Its Applications* 1 (1999), S. 23–27

[14] Hartley, Richard ; Zisserman, Andrew: *Multiple View Geometry in Computer Vision*. 2nd edition. Cambridge University Press, 2004. – ISBN 0521540518

[15] Haverinen, Janne ; Röning, Juha: An Obstacle Detection System Using a Light Stripe Identification Based Method. In: *IEEE International Joint Symposia on Intelligence and Systems* (1998), S. 232–236

[16] Hecht, Eugene: *Optics*. 4th edition. Addison Wesley, 2002. – ISBN 0805385665

[17] Heesch, Dimitri van: *Doxygen*. `http://www.stack.nl/~dimitri/doxygen`

[18] IEEE: *IEEE 754: Standard for Binary Floating-Point Arithmetic*. `http://grouper.ieee.org/groups/754`

[19] Institut für Regelungstechnik: *Zweibeiniger Autonomer Roboter BARt-UH*. `http://www.irt.uni-hannover.de/irt/asr/bart.html`

[20] Isgrò, Francesco ; Trucco, Emanuele: On Robust Rectification for Uncalibrated Images. In: *International Conference on Image Analysis and Processing* (1999), September, S. 297–302

[21] iXs Research Corporation: *Homepage*. `http://www.ixs.co.jp/index-e.html`

[22] Jia, Yunde ; Zhang, Xiaoxun ; Li, Mingxiang ; An, Luping: A Miniature Stereo Vision Machine (MSVM-III) for Dense Disparity Mapping. In: *17th International Conference on Pattern Recognition* 1 (2004), S. 728–731

[23] Jähne, Bernd: *Digitale Bildverarbeitung*. 5. Auflage. Springer Verlag, 2001. – ISBN 3540412603

[24] Knorr, Gerd: *video4linux2*. http://linux.bytesex.org/v4l2

[25] Kroah-Hartman, Greg: *udev*. http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html

[26] Kulkarni, Arun D. ; Byars, P.: Neural nets for invariant object recognition. In: *Symposium on Applied Computing* (1991), S. 336–344

[27] Lee, Doo H. ; Kweon, In S.: A Novel Stereo Camera System by a Biprism. In: *IEEE Transactions on Robotics and Automation* 16 (2000), Oktober, Nr. 5, S. 528–541

[28] Lehrstuhl für Angewandte Mechanik: *Johnnie - zweibeinige Laufmaschine der TUM*. http://www.amm.mw.tu-muenchen.de/Forschung/ZWEIBEINER/johnnie.html

[29] Lehrstuhl für Steuerungs- und Regelungstechnik: *Projekt ViGWaM*. http://lsr.ei.tum.de/team/vigwam

[30] Lehrstuhl für Steuerungs- und Regelungstechnik: *Stereo-Camera-Head for Humanoid Walking Machine*. http://www.lsr.ei.tum.de/team/vigwam/Head/Kopf.html

[31] Loop, Charles ; Zhang, Zhengyou: Computing Rectifying Homographies for Stereo Vision. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 1 (1999), Juni, S. 125–131

[32] Malm, Henrik ; Heyden, Anders: Stereo Head Calibration from a Planar Object. In: *Computer Vision and Pattern Recognition* 2 (2001), S. 657–662

[33] Middlebury College: *Stereo Vision Research Page*. http://cat.middlebury.edu/stereo/data.html

[34] Mikrut, Zbigniew: Recognition of Objects Normalized in Log-polar Space Using Kohonen Networks. In: *2nd International Symposium on Image and Signal Processing and Analysis* (2001), S. 308–312

[35] Moré, Jorge ; Garbow, Burt ; Hillstrom, Ken: *Minpack*. http://www.netlib.org/minpack

[36] Mouaddib, El M. ; Batlle, Jorge F. ; Salvi, Joaquim: Recent Progress in Structured Light in order to Solve the Correspondence Problem in Stereo Vision. In: *IEEE International Conference on Robotics and Automation* 1 (1997), S. 130–136

[37] Mühlmann, Karsten ; Maier, Dennis ; Hesser, Jürgen ; Männer, Reinhard: Calculating Dense Disparity Maps from Color Stereo Images, an Efficient Implementation. In: *IEEE Workshop on Stereo and Multi-Baseline Vision* (2001), Dezember, S. 30–36

[38] Nemosoft: *Drivers for Philips USB webcams.* `http://www.smcc.demon.nl/webcam`

[39] Nocedal, Jorge ; Wright, Stephen J.: *Numerical Optimization.* 1st edition. Springer, 1999. – ISBN 0387987932

[40] Ortega, Javier P.: *Construction and Integration of a Multiaxial Stereo-Camera Head in a Walking Machine*, Technische Universität München, Diplomarbeit, 2001

[41] Pachidis, Theodore ; Lygouras, John: A pseudo stereo vision system as a sensor for real time path control of a robot. In: *19th IEEE Instrumentation and Measurement Technology Conference* 1 (2002), Mai, S. 1589–1594

[42] Pachidis, Theodore ; Lygouras, John: Pseudo Stereo Vision System: modifications for accurate measurements in 3-D space using camera calibration. In: *2nd ISA/IEEE Sensors for Industry Conference* (2002), November, S. 66–70

[43] Palus, Henryk ; Bereska, Damian: Region-Based Colour Image Segmentation. In: *5th Workshop on Color Image Processing* (1999), S. 67–74

[44] Petters, Sebastian ; Thomas, Dirk: *RoboFrame - Softwareframework für mobile autonome Robotersysteme*, Technische Universität Darmstadt, Diplomarbeit, 2005

[45] Philips: *Royal Philips Electronics.* `http://www.philips.de`

[46] Pollefeys, Marc ; Koch, Reinhard ; Gool, Luc V.: A simple and efficient rectification method for general motion. In: *7th IEEE International Conference on Computer Vision* 1 (1999), September, S. 496–501

[47] Quek, Francis K. H.: An Algorithm for the Rapid Computation of Boundaries of Run-Length Encoded Regions. In: *Pattern Recognition* 33 (2000), Nr. 10, S. 1637–1649

[48] The RoboCup Federation: *Humanoid Kid Size League, Humanoid Medium Size League, The Rules & Setup for Osaka 2005.* 1st edition. 2005. `http://er04.ams.eng.osaka-u.ac.jp/humanoid\_webpage/humanoid.pdf`

[49] Russel, Stuart J. ; Norvig, Peter: *Artificial Intelligence: A Modern Approach*. 2nd edition. Pearson Education, Inc., 2003. – ISBN 0130803022

[50] Saillard, Luc: *Philips USB Webcam Driver for Linux*. `http://www.saillard.org/linux/pwc`

[51] Schwarz, Hans R.: *Numerische Mathematik*. 3. Auflage. B. G. Teubner, 1993. – ISBN 3519229609

[52] Seko, Yasuji ; Miyake, Hiroyuki ; Yamaguchi, Yoshinori ; Hotta, Hiroyuki: Optical 3D sensing by a single camera with a lens of large spherical aberration. In: *Intelligent Sensors, Sensor Networks and Information Processing Conference* (2004), S. 179–182

[53] Shibata, Masaaki ; Honma, Taiga: 3D object tracking on active stereo vision robot. In: *7th International Workshop on Advanced Motion Control* (2002), S. 567–572

[54] Shibata, Masaaki ; Kawasumi, Hajime: Solution for stereo correspondences on active stereo vision robot. In: *8th IEEE International Workshop on Advanced Motion Control* (2004), S. 665–670

[55] Simulation und Systemoptimierung: *Humanoid Robot Project: Darmstadt Dribblers*. `http://robocup.informatik.tu-darmstadt.de/humanoid`

[56] Simulation und Systemoptimierung: *Subversion Repository*. `https://svn.sim.informatik.tu-darmstadt.de/robocode/trunk/stereo`

[57] Sony: *Sony AIBO Europe*. `http://www.eu.aibo.com`

[58] Suzuki, Yosihiko ; Takano, Manabu: Estimation of the Position of a Rigid Body with a Single Camera. In: *International Conference on Industrial Electronics, Control and Instrumentation* 2 (1991), S. 1309–1311

[59] Tanaka, Maki ; Maru, Noriaki ; Miyazaki, Fumio: 3-D tracking of a moving object by an active stereo vision system. In: *20th International Conference on Industrial Electronics, Control and Instrumentation* 2 (1994), S. 816–820

[60] The RoboCup Federation: *RoboCup Official Site*. `http://www.robocup.org`

[61] Tomasi, Carlo ; Kanade, Takeo: Shape and Motion from Image Streams: a Factorization Method - Full Report on the Orthographic Case / Robotics Institute, Carnegie Mellon University. 1992 (CMU-CS-92-104). – Forschungsbericht. Cornell TR 92-1270

[62] Trolltech: *Qt Overview*. `http://www.trolltech.com/products/qt/index.html`

[63] Trucco, Emanuele ; Verri, Alessandro: *Introductory Techniques for 3-D Computer Vision*. 1st edition. Prentice Hall, 1998. – ISBN 0132611082

[64] Ueshiba, Toshio ; Tomita, Fumiaki: *Calibration of Multi-camera Systems Using Planar Patterns*. `http://www.suri.it.okayama-u.ac.jp/sov2002/ueshiba.pdf`. Version: 2002

[65] Veksler, Olga: Semi-Dense Stereo Correspondence with Dense Features. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2 (2001), Dezember, S. 490–497

[66] Zach, Christopher ; Karner, Konrad ; Bischof, Horst: Hierarchical Disparity Estimation With Programmable 3D Hardware. In: *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2004), S. 275–282

[67] Zhang, Zhengyou: A flexible new technique for camera calibration. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2002), S. 1330–1334

[68] Zhang, Zhengyou ; Deriche, Rachid ; Faugeras, Olivier ; Luong, Quang-Tuan: A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry. In: *Artificial Intelligence* 78 (1994), Nr. 1-2, 87-119. `http://www-sop.inria.fr/robotvis/personnel/zzhang/softwares.html`

[69] Zhang, Zhengyou ; Faugeras, Olivier ; Deriche, Rachid: An Effective Technique for Calibrating a Binocular Stereo Through Projective Reconstruction Using Both a Calibration Object and the Environment. In: *Journal of Computer Vision Research - VIDERE* 1 (1997), Nr. 1