Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Oliver Dreschke

Ein System zur Entwicklung kostengünstiger omnidirektionaler Kameras für mobile Roboter

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Oliver Dreschke

Ein System zur Entwicklung kostengünstiger omnidirektionaler Kameras für mobile Roboter

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Kai von Luck
Zweitgutachter : Prof. Gunter Klemke

Abgegeben am 5. April 2007

Oliver Dreschke

Thema der Bachelorarbeit

Ein System zur Entwicklung kostengünstiger omnidirektionaler Kameras für mobile Roboter

Stichworte

omnidirektional, Kamera, Rundumsicht, Simulation, Spiegel

Kurzzusammenfassung

Um sich in ihrer Umgebung zurechtzufinden, benötigen autonome mobile Roboter Sensoren. Hier gibt es verschiedene Varianten. Eine Sensorart mit viel Potenzial ist die omnidirektionale Kamera. Omnidirektionale Kameras werden zum Beispiel seit einigen Jahren in großen Wettbewerben im Robotfußball genutzt. Bisher waren diese hauptsächlich großen Teams zugänglich. In dieser Arbeit ist ein System entwickelt worden, welches eine einfache omnidirektionale Kamera für die Verfolgung farblich eindeutiger Objekte ermöglicht. Damit soll diese Technologie auch kleinen Laboren in Classroom Kits zugänglich gemacht werden.

Hierzu bietet das System Werkzeuge, welche das Spektrum von der Idee einer omnidirektionalen Kamera bis zu ihrem Einsatz abdecken. Im Fokus stehen omnidirektionale Kameras, welche die Rundumsicht über den Blick auf einen Spiegel realisieren. In einer Simulationssoftware können verschiedene Spiegelformen getestet werden. Das ermöglicht eine Idee davon wie Informationen auf der Kamera vorliegen. Gleichzeitig liefert die Software alle notwendigen Daten, um den simulierten Spiegel fertigen zu können. Die Fertigung einfacher Spiegel für die Kamera kann hochschulintern geschehen, sofern eine Werkstatt mit Drehmaschine existiert. Für den Betrieb der Kamera gibt es eine fertige Bibliothek für Grundfunktionen, die erweitert werden kann. Eine Testumgebung rundet das System ab.

Oliver Dreschke

Title of the paper

A system for building low cost omnidirectional cameras

Keywords

omnidirectional, camera, robot, simulation, mirror

Abstract

Self-governed robots need sensors to behave in unknown environments. There are several different types of sensors like distance or infrared sensors. One flexible and interesting approach is the usage of omnidirectional cameras for finding and tracking colored objects, giving back their position to a robot. These cameras are used by big teams in robot soccer competitions. Unfortunately there is no such camera for little money that allows them to enter classrooms or little robot labs. This is where this thesis comes into place. It offers a system that can bring the technology of omnidirectional cameras into classrooms.

For that goal this thesis brings several tools. They start at the beginning and help understanding the view in which omnidirectional cameras see the world and go on to building a fitting, ready to use solution. For the first step there is a simulator that shows four basic mirror shapes and the effects on the picture the camera sees. At the same time the simulation creates all the information needed to physically build that shown mirror. Basic mirrors can be built in every workshop that has access to a turning lathe. For testing the camera this thesis brings a simple library and some software for monitoring the output and using the camera on a robot.

Danksagung

Zunächst möchte ich mich bei Prof. Dr. Kai von Luck für die sehr engagierte Unterstützung, die weit über diese Bachelorarbeit hinausreichte und die spendierte Currywurst bedanken. Damals beim RoboCup hatte ich wirklich Hunger.

Als Nächstes möchte ich meinen Eltern für die langjährige Unterstützung danken, ohne die mein Studium in dieser Form nicht möglich gewesen wäre.

Bei meinem Großvater möchte ich mich für den festen Händedruck und die strahlenden Augen zum Abschied bedanken.

Bei Robert Radke und Familie möchte ich mich dafür bedanken, dass ich ihre Werkstatt zustauben durfte und für das Korrekturlesen dieser Arbeit.

Bei Karin Munstermann und Familie möchte ich mich dafür bedanken, dass ich immer noch willkommen war, selbst nachdem ich die Schuld dafür zu tragen hatte, dass diverse Familienmitglieder öfter zu spät zum Abendbrot gekommen sind.

Bedanken möchte ich mich auch bei allen Mitarbeitern der Zentralwerkstatt, die mir das benötigte Werkzeug erklärt haben und trotz ungünstig bemaßter Zeichnungen, Teile für mich gefertigt haben, die später perfekt gepasst haben.

Bei Edita möchte ich mich für die Einführung in das SMD-Löten bedanken.

Dazu möchte ich mich bei allen guten Professorinnen und Professoren an der HAW bedanken, die mir sehr viel beigebracht haben.

Zum Abschluss möchte ich mich bei Herrn Wissenschafts- und Forschungssenator Jörg Dräger bedanken, dafür, dass er nicht früher in die Politik gekommen ist und Studiengebühren eingeführt hat, denn in dem Fall hätte ich dieses Studium nicht angetreten. Das Studium möchte ich jedoch nicht missen, denn es hat meine Sicht auf viele Dinge grundlegend geformt und neben sehr viel Arbeit auch Spaß gemacht.

Inhaltsverzeichnis

1 Einleitung **9**
- 1.1 Motivation . . . 9
- 1.2 Analyse . . . 10
  - 1.2.1 Bisherige Technik . . . 10
  - 1.2.2 Möglichkeiten bessere Rundumsicht zu erhalten . . . 11
- 1.3 Ziel dieser Arbeit . . . 11

2 Omnidirektionales Sehen mit Kameras **13**
- 2.1 Mehrere Kameras nutzen . . . 13
- 2.2 Besondere Optiken . . . 14
- 2.3 Kamera mit Spiegel . . . 15
- 2.4 Spiegel improvisieren oder bauen . . . 16
- 2.5 Anforderungen an die Kamera . . . 17

3 Simulationssoftware **19**
- 3.1 Wege zur Spiegelform . . . 19
- 3.2 Pro/Contra Simulation . . . 20
  - 3.2.1 Was ist eine Simulation . . . 20
  - 3.2.2 Welche Schritte umfasst eine Simulation . . . 21
  - 3.2.3 Simulation vs. Realität . . . 21
  - 3.2.4 Ist eine Simulation nötig? . . . 22
- 3.3 Anforderungen . . . 24
- 3.4 Entwurf der Simulationssoftware . . . 25
- 3.5 Implementation der Simulationssoftware . . . 33
  - 3.5.1 Raytracing . . . 33
  - 3.5.2 POV-Ray . . . 36
  - 3.5.3 Konkreter Aufbau der Simulationssoftware . . . 37
- 3.6 Fazit über Entwurf und Implementation der Simulation . . . 42

4 Nutzung der Simulationssoftware **44**
- 4.1 Modellbildung . . . 44
- 4.2 Die Platinenkamera . . . 44
  - 4.2.1 Bestimmung des Öffnungswinkels aus der Brennweite . . . 46

4.2.2 Praktische Bestimmung des Öffnungswinkels anhand eines Bildes . . 47
4.2.3 Bestimmung der Schärfentiefe 49
4.3 Testszenario 53
4.3.1 Aufbau der Versuchsanordnung 53
4.3.2 Berechnungsgrundlage für das Gestell und die Spiegel in der Simulation 56
4.3.3 Das Spielfeld 60
4.3.4 Ergebnis der Überprüfung 61
4.4 Simulationsszenario für Spiegel 62
4.4.1 Szenario der Beispielanwendung 62
4.4.2 Auswahl Spiegelformen 63
4.4.3 Entscheidung für einen Spiegel nach Lauf 70
4.4.4 Bewertung des Simulationsergebnisses und fertigen Spiegels 71
4.5 Fazit: Bewertung der implementierten Simulation 72

5 Aufbau der Kamera 74
5.1 Ausgangspunkt 74
5.2 Mechanische Designentscheidungen 74
5.3 Spiegelbau 76
5.3.1 Bestimmung der Oberfläche des Spiegels 79
5.4 Auswahl des Steuermodules 82
5.5 Kameralogik 85
5.5.1 Umwandlung Positionsangaben in Richtungsinformationen 85
5.5.2 Automatisches Einlesen der gesuchten Objektfarbe 88
5.5.3 Aufbau der Kameralogik 90
5.6 Fazit: Was kann die Kamera 93

6 Integration der Kamera in einen Roboter 95
6.1 Ausgangspunkt 95
6.2 Veränderungen an bestehender Plattform 95
6.3 Integration der Kamera 97
6.4 Fazit: Was kann der Roboter 98
6.5 Testumgebung 100
6.5.1 Einleitung 100
6.5.2 Überwachungsprogramm 100
6.5.3 Roboterlogik 101
6.5.4 Fazit: Was kann die Testumgebung 101

7 Zusammenfassung 103
7.1 Zusammenfassung 103
7.2 Fazit 104
7.3 Ausblick 105

Literaturverzeichnis **107**

A Bauanleitung Erweiterung AKSEN-Board auf entkoppelte Motorspannung V1.12 **109**
A.1 Vorwort . . . 109
A.2 Technische Details . . . 109
A.2.1 Grundlegende Idee . . . 109
A.2.2 Umbau . . . 110
A.2.3 Benötigte Teile für zwei Adapter . . . 110
A.3 Aufbau der Teilstücke des Adapters . . . 110
A.4 Bauteil 1 . . . 110
A.4.1 Was gemacht werden muss . . . 110
A.4.2 Bauteil 2 . . . 112
A.5 Zusammenbau . . . 117
A.5.1 Was gemacht werden muss . . . 117
A.6 empfohlene Vorgehensweise . . . 117
A.7 Anmerkung zum Umbau beider Motortreiber . . . 118

B Inhalt der CD **119**

C Verwendete Software **120**
C.1 Für die Simulation erforderliche Software . . . 120
C.2 Programmierung . . . 120
C.3 Grafiken und Diagramme . . . 120

1 Einleitung

1.1 Motivation

Roboter erobern immer mehr Bereiche des Lebens. In Fahrzeugen kommen zunehmend Fahrerassistenzsysteme zum Einsatz. Diese sollen die Sicherheit erhöhen und Fahrzeugen erlauben automatisch Notbremsungen zu starten, den passenden Abstand zum vorhergehenden Fahrzeug einzuhalten, Fahrzeuge in der Spur der Fahrbahn zu halten und Kollisionen zum Beispiel bei Spurwechseln zu verhindern. Systeme, die Fahrzeuge in Extremsituationen beherrschbar machen wie ABS und ESP kennt heute bereits jeder.

Im industriellen Bereich kommen zunehmend mehr fahrerlose Transportsysteme zum Einsatz. Diese versorgen zum Beispiel die Montage von Baugruppen mit Material und transportieren die fertigen Baugruppen an die nächste Station im Produktionsprozess. Diese fahrerlosen Transportsysteme navigieren zum Beispiel mit Hilfe von Linien oder Magnetmarkierungen. Zur Vermeidung von Unfällen durch Kollisionen mit Objekten und Personen haben diese Systeme verschiedene Sensoren, wie zum Beispiel Laserscanner und intelligente Kamerasysteme. Ein solches System findet sich an der HAW-Hamburg zurzeit in einem System mit dem Namen FAUST. An diesem System werden verschiedene Technologien zur Fahrerassistenz, Sensorik und Telemetrie in herkömmlichen und autonomen Fahrzeugen erforscht [1].

In einer viel kleineren, aber nicht weniger interessanten Liga, spielt der „RoboCup“. Die Initiatoren des „RoboCup“ haben das Ziel ausgesprochen, bis zum Jahr 2050 eine Fußballmannschaft vollständig autonomer Roboter zu entwickeln. Diese soll gegen den dann amtierenden Meister im Fußball gewinnen können. In 2006 gab es hier verschiedene Ligen mit unterschiedlichen Rahmenbedingungen und Zielsetzungen. Diese Zielsetzungen gehen zum Teil weit über das Fußballspielen hinaus. Etwa in der „Rescue League“, in der autonome Roboter in unwegsamen Gelände Verwundete finden sollen.

Bei den Fußball spielenden Robotern hat sich der omidirektionale Antrieb durchgesetzt. Dieser erlaubt eine Bewegung aus dem Stand in jede beliebige Richtung. Hierzu gab es bereits an der HAW-Hamburg zwei Diplomarbeiten. Eine von Michael Manger[2], der sich mit den

[1] Projektseite unter http://www.informatik.haw-hamburg.de/faust.html

[2] Design und realisierung einer experimentellen Plattform für Roboterfußball, Mangerr (2004)

Grundlagen solcher Systeme beschäftigt hat und eine von Michael Ziener[3], der eine größere Experimentierplattform entwickelte. Inzwischen wird der omnidirektionale Antrieb an der HAW-Hamburg im Rahmen des Projektes „Mobile Roboter“ erfolgreich eingesetzt. Hier gibt es eine eigene hochschulübergreifende Roboterliga von Robotlaboren, in denen kleine Teams von Studenten mit Fußballrobotern gegeneinander antreten.

Der omnidirektionale Antrieb lässt den Wunsch nach adäquater omnidirektionaler Sensorik aufkommen.

1.2 Analyse

1.2.1 Bisherige Technik

Omnidirektionale Plattformen können stark von senso-optischen Systemen profitieren. Hier werden unterschiedliche Sensoren durch einen optischen Sensor ergänzt. Für die Entfernungsmessung zu Wänden etc. werden, wie bisher üblich, Infrarot oder Ultraschallsensoren eingesetzt, da diese für den Zweck sehr gute Ergebnisse liefern können. Für eine omnidirektionale Erkennung eines Balles oder Tores sind diese Sensoren hingegen weniger geeignet. Darum wird sich dem Konzept der „Sensorfusion“ bedient und die bestehenden Sensoren mit einem weiteren spezialisierten Sensor ergänzt.

Bisher wird im Robotlabor der HAW-Hamburg ein Spezialball verwendet, der Infrarot ausstrahlt. Die Position des Balles wird mit preisgünstigen Infrarotsensoren ermittelt, die an der Außenkante der Roboter angebracht sind. Hinter der Ballführungs- und Schussvorrichtung befindet sich eine Leiste mit Infrarotsensoren. Diese ist empfindlicher, als die übrigen Ballkennungssensoren. Die Leiste kann den Ball aber nur in einer Richtung detektieren.

Die Tore sind durch zwei Infrarotsender gekennzeichnet, die unterschiedlich gepulste Signale aussenden. Diese werden mit einer Spezialkonstruktion und weiteren Infrarotempfängern empfangen. Damit wird die Position der Tore ermittelt. Die Konstruktion besteht aus drei Sensoren, die durch Abteilungen voneinander getrennt sind. Damit ist jeder Sensor auf eine bestimmte Richtung beschränkt. Der mittlere Sensor hat ein sehr kleines Abteil, wodurch er den Torsender nur dann empfangen kann, wenn er genau darauf gerichtet ist. Die beiden anderen Sensoren befinden sich links und rechts. Ihre Abteilungen haben einen größeren Öffnungswinkel. Dadurch kann früher erkannt werden, ob sich ein Tor links oder rechts vom Roboter befindet. Diese Konstruktion kann Tore sehr sicher erkennen, hat allerdings den Nachteil, dass gezielt nach den Toren gesucht werden muss. Die Richtungsinformation ist nicht immer verfügbar.

[3] Konstruktion einer programmierbaren, omnidirektionalen Roboterplattform, Ziener (2005)

1.2.2 Möglichkeiten bessere Rundumsicht zu erhalten

Um dem Roboter eine Rundumsicht zu ermöglichen, in der er den Ball und die Tore zu jeder Zeit im Blick haben kann, kann man die Zahl und die Qualität der Sensoren auf dem Roboter vergrößern. Das hat allerdings den Nachteil, dass dieses zusätzlichen Platz, mehr Ports und mehr Rechenleistung beansprucht. Ein großes Problem stellt hierbei die Art der Ballsensoren dar. Bei den Infrarotsensoren handelt es sich um analoge Sensoren. Sie liefern bei unterschiedlich stark eintreffender infraroter Strahlung unterschiedliche Spannungen an die Spannungswandler des Steuerungsmoduls. Die analogen Eingänge von Mikrocontrollern haben jedoch den Nachteil, dass eine Auswertung vergleichsweise langsam ist. Je mehr analoge Sensoren man einlesen möchte, desto länger dauert es, bis Ergebnisse für das Verhalten des Roboters verfügbar sind. Zusätzlich gibt es das Problem, dass die Sensoren nur begrenzte Öffnungswinkel haben dürfen. Dadurch entstehen tote Winkel, in denen die Position des Balles verloren geht.

Eine bessere Lösung stellt hier die omnidirektionale Kamera da. Diese ist deutlich flexibler und kann als einzelner Sensor viele Richtungsinformationen vorverarbeiten und liefern. Sie kann theoretisch nicht nur den Ball und Tore erkennen, sondern auch die Position anderer Roboter bereitstellen. Eine omnidirektionale Kamera ist mithilfe eines Filters in der Lage Infrarot zu erkennen. Damit wäre es immer noch möglich den bisher eingesetzten Ball zu nutzen. Allerdings hat dieser Ball den Nachteil, dass seine Batterie nach kurzer Zeit erschöpft ist. Sollte sich die omnidirektionale Kamera durchsetzen, so würde es Sinn machen diesen Ball durch ein besseres System zu ersetzen.

Das Problem der omnidirektionalen Kamera besteht darin, dass diese Art der Kamera lediglich für die Panoramafotografie erhältlich und sehr teuer ist. Einfache, preiswerte Kameras für den Einsatz in kleinen mobilen Robotern sind auf dem Markt nicht verfügbar. Eine Übersicht verschiedener kommerzieller omnidirektionaler Kameras findet man auch der Seite von Thomas B. Kunz Kunz. Zum Zeitpunkt dieser Arbeit war die preiswerteste Kamera im Bereich um 500 Euro angesiedelt. Aus diesem Grund ist es notwendig, einen Spiegel selber zu konstruieren und zu bauen.

1.3 Ziel dieser Arbeit

Omnidirektionale Kameras stellen einen guten Sensor für omnidirektionale Plattformen da. Diese Bachelorarbeit soll die Grundlagen zur Eröffnung dieser Technologie im Hochschuleinsatz schaffen. Hierfür wird ein System entwickelt, mit dem spezialisierte, omnidirektionale Kameras für beliebige Zwecke kostengünstig geschaffen werden können. Als Funktionsdemonstration wird der Bereich Robotfußball aufgegriffen. Hierbei soll mithilfe der omnidirektionalen Kamera eine Richtungsinformation zu einem Ball verfügbar gemacht werden.

Auf das Gewinnen von Entfernungsinformationen mithilfe der Kamera wird bewusst verzichtet, da es für diesen Zweck spezialisierte Sensoren gibt, die gut funktionieren. Durch das Verknüpfen verschiedener Sensorinformationen können Richtungs- und Entfernungsdaten später zusammengebracht werden.

2 Omnidirektionales Sehen mit Kameras

Omnidirektionales Sehen mit Kameras kann auf verschiedene Weisen erreicht werden. Dieses Kapitel beschäftigt sich mit den Konzepten zu omnidirektionalem Sehen ohne bewegliche Teile und stellt drei Möglichkeiten zum Erreichen dieser Fähigkeit vor.

2.1 Mehrere Kameras nutzen

Die einfachste Möglichkeit simuliert die Panoramafunktion von handelsüblichen, digitalen Fotokameras. Um ein 360° Panoramabild zu erzeugen, kann eine Kamera auf ein Stativ gestellt werden. Anschließend nimmt man in jeder Richtung mehrere Bilder auf und verschmelzt sie zu einem großen Bild. Dieses Vorgehen wird in der Fotografie als „Stitching“ bezeichnet. Das Resultat zeigt ein Rundumbild ohne nennenswerte Verzerrungen.

Das Drehen der Kamera kann vermieden werden, indem man mehrere Kameras einsetzt. Hierfür wird als Erstes der Öffnungswinkel der Linse der Kamera betrachtet. Dieser bestimmt die Anzahl der benötigten Kameras. Liegt eine Kamera mit einem Öffnungswinkel von 60° vor, so berechnet sich die Anzahl der Kameras aus folgender Formel.

$$\frac{360}{\ddot{O}ffnungswinkel} = AnzahlderKameras \tag{2.1}$$

$$\Rightarrow \frac{360}{60} = 6 \tag{2.2}$$

Die Rechnung zeigt, dass von dieser Kameraart sechs Stück benötigt würden, um eine omnidirektionale Kamera zu bauen.

Omnidirektionales Sehen mit der Nutzung mehrerer Kameras zu implementieren hat den Vorteil, dass sehr einfach auszuwertende Bilder entstehen. Jede Kamera stellt einen bestimmten Bereich einer Richtung da. Die Unterteilung dieses Bereiches ist beliebig fein und Linsenverzerrungen können bei diesem System vernachlässigt werden.

Die Nachteile finden sich im vergrößerten Platzbedarf, einem höheren Energiebedarf und der komplizierteren Ansteuerung der Kameras. Informationen müssen nicht nur von einer, sondern gleichzeitig von mehreren Kameras geholt und verarbeitet werden.

Ein Beispiel für eine solche Kamera ist die „Ladybug2“ 2.1 der Firma Point Grey Research. Sie kombiniert 6 Kameras zu einer und bietet eine hohe Auflösung bei geringen Ausmaßen. Allerdings ist sie sehr teuer.

Abbildung 2.1: Ladybug2 von Point Grey Research vereint sechs Kameras zu einer omnidirektionalen Kamera

2.2 Besondere Optiken

Spezielle Objektive reduzieren die Zahl der benötigten Kameras. Zu diesen gehören so genannte „Fischaugenlinsen“, englisch „Fisheyes“. Diese Linsen ermöglichen durch ihre besondere Form größere Sichtwinkel. Für das Kleinbildformat haben Brennweiten um 16mm einen diagonalen Bildwinkel von 180° und ein vollformatiges Bild. Alternativ gibt es auch Vorsatzlinsen, wie den Fisheye-Vorsatz FC-E9 der Firma Nikon 2.2. Mit solchen Linsen kann die Zahl der benötigten Kameras reduziert werden. Allerdings verursachen diese Optiken Verzerrungen, die eine Richtungsbestimmung erschweren.

Abbildung 2.2: Nikon Fisheye Vorsatz FC-E9 bietet einen Blickwinkel von 183°

Alternativ kann mit 180°-Fischaugenlinsen auch eine Rundumsicht mit nur einem Bild erzeugt werden. Dafür sieht die Kamera mit Linse direkt nach oben. Diese Bilder sind als

Abbildung 2.3: Rundumsicht mit 180° Fisheyelinse

Sensor zur Richtungsbestimmung auf mobilen Robotern allerdings nicht geeignet. Ein Beispielbild ist in Abbildung 2.3 [1] zu sehen. Man sieht dort, dass auf dem Bild beinahe nur Himmel aufgenommen ist. Der Bereich mit Nutzdaten wäre nur bei sehr hoch auflösenden Kameras, wie sie bei der Panoramafotografie eingesetzt werden, zu gebrauchen.

2.3 Kamera mit Spiegel

Einen anderen Ansatz verfolgen die beim RoboCup 2006 genutzten omnidirektionalen Kameras. Diese bestehen aus einer Kamera, die senkrecht auf einen Spiegel blickt. Dieser Spiegel ist meist konisch, kugelförmig oder in Form eines Paraboloiden. Der Ansatz ist sehr flexibel. Durch unterschiedliche Spiegelformen ist es möglich, die Sichtweite oder die Auflösung in bestimmten Bereichen des Bildes zu verändern. Zusätzlich ist diese Art der omnidirektionalen Kamera preisgünstig, da je nach Anwendungsbereich bereits mit sehr kostengünstigen Kameras angefangen werden kann. Da es nur eine Kamera gibt, entfällt die Problematik Informationen mehrerer Kameras auszulesen, die Informationen zusammen zu bringen und auszuwerten. Ein Beispiel für eine solche omnidirektionale Kamera zeigt Abbildung 2.4. Sie zeigt eine kommerzielle Kamera, welche auf dem RoboCup 2004 eingesetzt wurde.

[1] Das Bild kommt von der Internetseite www.crosus.de von Herrn Tilo Kunze, die sich unter anderem mit dem Thema Panoramafotografie beschäftigt. Eine weitere interessante Webadresse zum Thema Panoramafotografie ist von Herrn Thomas B. Kunz. Beide Seiten sind im Literaturverzeichnis zu finden.

Abbildung 2.4: Vstone Kamera, beim Robocup 2004 eingesetzt, das Bild stammt von der Website von Vstone Vstone

Bei dieser Art der Kamera gibt es zwei Schwierigkeiten. Die erste Schwierigkeit besteht darin, einen geeigneten Spiegel für die jeweilige Anwendung zu finden. Die zweite Schwierigkeit stellt das Aufhängen und justieren des Spiegels über der Kameralinse dar. Die Aufhängung soll nach Möglichkeit nicht im Bild zu sehen sein, gleichzeitig jedoch eine hohe Festigkeit aufweisen und den Spiegel sicher fixieren. Kleine Verschiebungen der Lage, zwischen Spiegel und Kamera, können Informationen bereits stark verfälschen und sind im Betrieb sehr negativ. Je größer die Halterung des Spiegels auf der Reflexionsfläche zu sehen ist, desto größer sind die toten Winkel der Kamera.

Das Ergebnis dieser Art der Rundumsicht ist ein unterschiedlich stark verzerrtes Bild. Diese Verzerrung wird bei der Panorama Fotografie durch Algorithmen aus dem Bild heraus gerechnet. Als Sensor für Roboter spielt die Verzerrung eine geringere Rolle. Hier interessieren Farbschwerpunkte und Linien, die auch verzerrt genügend Aussagekraft haben.

Besonders die Flexibilität des Sichtbereiches durch den Spiegel und die Möglichkeit der sehr kleinen Ausmaße, machen diese Methode ideal für eine omnidirektionale Kamera auf Robotern. Darum wird dieser Ansatz in dieser Arbeit weiterverfolgt. Es stellt sich die Frage, wie eine omnidirektionale Kamera mit Spiegel gebaut werden kann.

2.4 Spiegel improvisieren oder bauen

Um eine omnidirektionale Kamera mit Spiegel zu bauen, gibt es zwei Möglichkeiten. Es ist möglich einen Spiegel zu improvisieren oder selber zu bauen. Improvisieren bedeutet hier die Suche eines geeigneten Spiegels aus anderen Bereichen, zum Beispiel des Lampenbaus. Die Software, sowie die Halterung des Spiegels wird dann an dieses Bauteil angepasst.

Es ist nicht notwendig einen Spiegel zu bauen. Die Anforderungen an den Spiegel sind relativ gering. Dadurch, dass nur eine Richtungsinformation gesucht wird, benötigt man lediglich eine spiegelnde Form. Dieses haben Andere bereits gemacht, um damit Panoramafotografien zu fertigen. Einer von ihnen, der eine sehr gute Internetseite zum Thema gemacht hat, ist Herr Tilo Kunze Kunze. Diese Anleitungen können problemlos auf die Robotik übertragen werden und bieten einen schnellen Einstieg.

Wird der Spiegel improvisiert, so spart dieses viel Entwicklungsarbeit und damit Geld. Im Heimbereich ist es nur schwer möglich, einen Spiegel in einer genau definierten Form zu fertigen. Improvisierte Spiegel kommen ohne externe Ressourcen aus.

Auf der anderen Seite kann es kostengünstiger sein, einen Spiegel selber zu bauen, da nur die Materialien bezahlt werden müssen.

Bei selbst gebauten Spiegeln, gibt es eine langfristige Versorgungssicherheit. Auch nach längerer Zeit ist es möglich einen Spiegel, mit den gleichen Spezifikationen, bereits vorhandener Spiegel zu bauen. Sollte ein neues Projekt eine zusätzliche omnidirektionale Kamera benötigen, kann der Spiegel nach bestehenden Plänen nachgefertigt werden. Die Abhängigkeit von einem Fremdhersteller entfällt. Damit verschwindet auch die Notwendigkeit einer Lagerhalterung von Spiegeln, die vielleicht einmal benötigt werden könnten.

Das Bauen eigener Spiegel, ermöglicht einen speziell auf die Situation angepassten Spiegel zu fertigen. Hierbei können Optionen wie Größe, Form und Gewicht selber beeinflusst werden. Das erhöht die Flexibilität für den Einsatz omnidirektionaler Kameras.

Für einen schnellen Einstieg, eignen sich improvisierte Spiegel sehr gut. Man hat nach kurzer Zeit etwas in der Hand, mit dem man experimentieren kann. Das geht auf Kosten der Flexibilität. Bei improvisierten Spiegeln hat man keinen direkten Einfluss auf Form, Größe und Gewicht. Ebenfalls verliert man die Versorgungssicherheit. Ist ein Objekt gefunden, welches sich für einen Kameraspiegel besonders gut eignet, kann es sich um einen Saisonartikel handeln oder um ein Modeprodukt. Diese werden unter Umständen nur ein Jahr lang produziert, danach wird es unmöglich oder sehr aufwendig Ersatz zu beschaffen.

Dauerhafter und zukunftssicherer sind Spiegel, die selber gefertigt werden können. Zusätzlich bieten sie die Möglichkeit, einen Spiegel individuell den Bedürfnissen eines Projektes anzupassen. Für den Einsatz in einem Robot Labor ist das sehr wichtig. Aus diesem Grund gehört die Konstruktion eines Spiegels in den Umfang dieser Arbeit.

2.5 Anforderungen an die Kamera

Die Kamera sollte möglichst klein und robust sein, da dieses die Möglichkeit eröffnet, sie zum Beispiel auf einem im Labor vorhandenen „c't-Bot“ einzusetzen. Sie könnte einen Ball

auf einem Fußballfeld verfolgen und nutzbare Daten an ein AKSEN-Board liefern, welches einen Roboter steuert.

Schön wäre es, wenn diese Daten in sehr schneller Folge kommen könnten, da sich Bälle im Spiel manchmal sehr schnell bewegen. Vorteilhaft ist es ebenfalls, wenn der Spiegel leicht austauschbar ist. Das eröffnet die Möglichkeit, die Kamera mit wenig Aufwand für andere Einsatzgebiete zu modifizieren.

3 Simulationssoftware

Dieses Kapitel beschäftigt sich mit Simulationssoftware. Simulationen sind eine Möglichkeit omnidirektionale Kameras greifbar zu machen. Dieses Kapitel beschäftigt sich mit der Frage, wie ein Weg zu einer omnidirektionalen Kamera führen kann. Dabei wird Simulationssoftware als möglicher Weg angenommen. Aufbauend auf diese Annahme wird auf das Thema Simulationen näher eingegangen. Dabei werden das abstrakte Wort Simulation konkretisiert und Vor- und Nachteile einer Simulation erörtert.

Nach der generellen Betrachtung von Simulationen wird auf die Konzepte und Gedanken eingegangen, die zur Umsetzung der Simulationssoftware für omnidirektionale Kameras in dieser Arbeit führen. Begonnen wird mit der Auseinandersetzung mit den Anforderungen. Hier wird beschrieben, welche Eckpunkte eine Simulationssoftware für diesen Zweck erfüllen muss, um sinnvoll eingesetzt werden zu können. Zu dem wird eingeschränkt, welche Funktionen aus welchem Grund zur Verfügung gestellt werden müssen. Nach Definition der Anforderungen wird auf den Aufbau einer Software eingegangen, die diesen Anforderungen genügt. Am Ende dieses Kapitels steht ein Entwurf eines Programmes.

3.1 Wege zur Spiegelform

Der Spiegel einer omnidirektionalen Kamera ist in der Regel nicht flach. Es wäre möglich, einen Spiegel zu konstruieren, der sich sehr schnell dreht und so, praktisch augenblicklich, Informationen für alle Richtungen liefert. Das wäre allerdings deutlich anfälliger, als die Möglichkeit, mit der Kamera auf einen plastischen Spiegel zu blicken und dieses Bild dann auszuwerten.

Geschaffen werden muss ein plastischer Spiegel. Für die Auswertung hilfreich ist ein Drehkörper, also ein Objekt, das durch das Drehen eines Querschnittes entsteht. Ein Beispiel dafür ist ein Kegel oder Paraboloid. Auf diesen Spiegelformen steht jeder Punkt in einfacher Abhängigkeit zur Richtung.

Nachdem die Anforderungen feststehen, kann die Spiegelform berechnet werden. Relativ einfach ist es die Sichtweite zu berechnen, die mit einer Spiegelform erreicht werden kann.

Ebenso ist es kein großes Problem die benötigte Größe des Spiegels, sowie seine optimale Höhe über der Kamera zu bestimmen.

Das Problem besteht darin, dass der Blick von unten auf eine Form wie einen Paraboloiden oder Kegel ungewohnt ist. In der Regel bestehen keine Erfahrungswerte dafür, wie das Umgebungsbild durch die Form verzerrt wird. Eine Kugel zum Beispiel wird durch eine omndirektionale Kamera mit einem kegelförmigen Spiegel zu einem Kegel. Omnidirektionale Kameras, die als Spiegel eine Kugel nutzen, haben eine eher aufgeblasene Repräsentation der Kugel.

Durch verschiedene Formen können verschiedene Effekte erzeugt werden. Diese vorher zu berechnen, kann sehr kompliziert werden. Eine große Hilfe wäre die Möglichkeit, solche Effekte in der Konstruktionsphase sichtbar zu machen. Hierzu kann eine Simulation beitragen.

3.2 Pro/Contra Simulation

In diesem Unterkapitel wird beschrieben, um was es sich in diesem Fall bei Simulationen handelt, welche Schritte nötig sind, wo die Schwierigkeiten liegen und welche Vor- und Nachteile eine Simulation gegenüber einem Modell in der Realität aufweist.

3.2.1 Was ist eine Simulation

Bei einer Simulation geht es darum, ein System mit seinen dynamischen Prozessen in einem vereinfachten Modell nachzubilden. Die Erkenntnisse aus Experimenten an diesem Modell, können dann auf die Wirklichkeit übertragen werden (Steinbuch, 2004, S. 17).

Simulationen können mit Blick auf verschiedene Ziele eingesetzt werden. Ein Ziel ist es, das Verhalten von Systemen zu ermitteln, bevor diese gebaut werden. Ein weiteres Ziel kann es sein, eingetretene Ereignisse nachzuvollziehen. Schließlich kann die Simulation dazu genutzt werden, Parameter zu identifizieren, die gewünschte Eigenschaften erzeugen sollen.

Steinbuch betrachtet die Simulation speziell im Bereich des konstruktiven Maschinenbaus. Dort würden sie genutzt, um die „Funktionsfähigkeit von Bauelementen und Baugruppen zu überprüfen, zu verbessern oder sie zu optimieren“ (Steinbuch, 2004, S. 35).

3.2.2 Welche Schritte umfasst eine Simulation

Der erste Schritt besteht darin, alle entscheidenden Eigenschaften des abzubildenden Systems zu definieren. Hierfür muss das System zunächst abgegrenzt und analysiert werden, bis alle Eingabe- und Ausgabegrößen, sowie die wesentlichen Parameter bekannt sind.

Der zweite Schritt besteht darin, ein experimentierfähiges Modell zu schaffen, welches auch die Übertragbarkeit auf die Wirklichkeit erlaubt. Ein experimentierfähiges Modell ermöglicht Untersuchungen des Systems mit vertretbarem Aufwand. Für die Übertragbarkeit auf die Wirklichkeit ist es notwendig, dass die ermittelten Aussagen für die gleichen Aktionen im realen System die gleiche Bedeutung haben. Diese gleiche Bedeutung muss durch Vergleiche mit dem realen System überprüft werden (Steinbuch, 2004, S. 18). Bei der Prüfung wird das Modell an das bestehende, reale System geeicht.

Der dritte Schritt besteht darin, Experimente am Modell durchzuführen. Die dabei erzielten Ergebnisse müssen auf das reale System übertragen und überprüft werden.

3.2.3 Simulation vs. Realität

Eine Simulation bietet gegenüber der Realität einige Vorzüge. Ist die Simulation fertig, lassen sich anhand ihrer Ergebnisse bestimmte Formen testen und analysieren.

Bei einer Simulation werden Experimente in den virtuellen Bereich verschoben. Dadurch können Ergebnisse ohne die in der Realität benötigten Werkzeuge entstehen. Die Simulation wird jedoch Abweichungen von der Realität beinhalten. Das liegt in der Natur der Simulation.

Simulationen können keine Informationen enthalten und aufbereiten, die ihr nicht gegeben wurden, sondern bereiten Informationen lediglich auf. Steinbuch nennt das den „Informationserhaltungssatz“ und weist besonders darauf hin, dass keine neuen Informationen entstehen (Steinbuch, 2004, S. 46). Zusätzlich zur Herausforderung genügend Daten für eine Simulation bereitzustellen, gibt es das Problem der Parametervielfalt. Was in der Realität manchmal nicht auffällt, ist die schiere Zahl an Parametern. Diese Zahl muss für eine Simulation reduziert werden, um in absehbarer Zeit zu brauchbaren Ergebnissen zu kommen. Die Unterscheidung zwischen wichtigen und unwichtigen Parametern erfordert eine gute Kenntnis der Funktionsweise des zu modellierenden Systems. Es kann leicht passieren, dass wichtige Parameter übersehen und Unwichtige genutzt werden. Dieser Bereich der Parameterfestlegung ist fehleranfällig und entscheidet über die Qualität der Simulation. Steinbuch spricht hier von dem „GIGO- (=garbage in, garbage out) Prinzip“ (Steinbuch, 2004, S. 46). Übersetzt bedeutet das „Müll hinein, Müll hinaus“ Prinzip und besagt, dass die Ergebnisse nur so gut sein können, wie die zur Berechnung zur Verfügung gestellten Ausgangsdaten.

Zu diesen Informationen, die der Simulation übergeben werden, gehören Informationen über Bauteile. Das Ergebnis einer Simulation ist immer ein idealisierter Zustand. Die reale Welt bietet so etwas nicht. Zwei gleiche Werkstoffe sind in der Praxis nie identisch. Maße variieren mehr oder weniger stark und Materialien haben mehr oder weniger große Verunreinigungen und Fehler (Steinbuch, 2004, S. 42). Es ist praktisch nicht möglich für die omnidirektionale Kamera zwei Spiegel zu bauen, die zu einhundert Prozent identisch sind. Man kann die Abweichungen jedoch so gering wie möglich halten.

Hinzu kommen Messungenauigkeiten und Werte, die lediglich geschätzt werden können. Bei der Kamera sind Beispiele für Messungenauigkeiten, die Höhe des Spiegels über der Linse, die genaue Einstellung der Linse über dem Sensorchip oder die Ausrichtung des Spiegels über dem Mittelpunkt der Linse. Werte die nicht genau ermittelt werden können, sind der tatsächliche Öffnungswinkel der verbauten Linse auf der Platinenkamera und die genaue Form der in der Überprüfung der Simulation eingesetzten Kugel. Hier kommt die Genauigkeit der Eingabe nicht über die einer vernünftigen Annahme, wie Steinbuch es nennt, hinaus. Diese Messungenauigkeiten führen zu einer mehr oder weniger starken Abweichung des real gefertigten und des simulierten Objektes.

Simulationen können gegenüber der Realität dazu führen, dass falsche Ergebnisse zu leicht anerkannt werden. Dieses ist laut Steinbuch besonders bei so genannten „Push Button Systemen“ zu beobachten. Soll ein Objekt gefertigt werden, ohne dass eine Simulation besteht, so ist es wahrscheinlich, dass die zugrunde liegenden Berechnungen eher mehrmals kontrolliert werden. Schließlich kosten Fehler hier Zeit und Geld. Simulationsergebnisse jedoch sind plastisch erfassbar und können oft richtig wirken und dennoch falsch sein. Einem Bild zu vertrauen ist einfacher, als Formeln. Fehler, die in der Realität bereits beim Bauen eines Spiegels oder spätestens beim Sichten des fertigen Spiegels aufgefallen wären, könnten ein Projekt längere Zeit begleiten.

Nutzt man eine Simulation, muss man sich diese Punkte klarmachen und bei der Analyse der Simulationsergebnisse berücksichtigen.

3.2.4 Ist eine Simulation nötig?

Betrachtet man eine omnidirektionale Kamera mit Spiegel, so wirkt diese sehr einfach. Die Bilder dieser Kameras ähneln sich oft. Daher stellt sich die Frage des Sinnes einer Simulation zur Entwicklung omnidirektionaler Kameras.

Eine Simulation ist nicht zwangsweise nötig. Die Form der Spiegel für diese Art der Kamera lässt sich mathematisch errechnen. Damit ist es möglich für verschiedene Zwecke, speziell angepasste Spiegel zu entwerfen. Dieser Schritt entfällt nicht mit der Nutzung einer Simulation. Für spezielle Spiegel muss eine Simulation angepasst werden. Das kostet zusätzliche

Zeit. Neben der Zeit, die für das Anpassen der Simulation benötigt wird, bietet die Simulation ein weiteres Feld für Fehlerquellen. Bei jeder Anpassung muss die Simulation erneut überprüft werden. Kostengünstige omnidirektionale Kameras müssen nicht zwangsweise perfekt sein. Oft reicht es ein spiegelndes Objekt zu nehmen, welches man kostengünstig im Baumarkt oder auf dem Sperrmüll finden kann. Es würde für die Arbeit reichen, die Vor- und Nachteile verschiedener Grundformen zu erarbeiten und zur Verfügung zu stellen. Die Zeit zum Entwickeln und Fertigen des Spiegels entfällt hier.

Eine Simulation kann jedoch auch hilfreich sein. Soll ein fertiger Spiegel gekauft werden, lassen sich vorher verschiedene Spiegelformen simulieren. Damit lassen sich Fehlkäufe vermeiden. Da mit dem simulierten Bild bereits vor dem Eintreffen des Spiegels gearbeitet werden kann, ist es hier möglich Wartezeit zu überbrücken. Soll eine Kamera für einen speziellen Zweck konstruiert werden und eine Form steht fest, kann das Zusammenspiel von Kameraoptik und Spiegelform, für ein spezielles Szenario kreiert werden. Dem Modell stehen verschiedene Parameter zur Optimierung des Systems zur Verfügung, dieses ermöglicht ein besseres Ergebnis. Simulationen helfen es zu vermeiden, suboptimale Spiegel aus Zeit- oder Kostengründen einzusetzen. Wird auf eine Simulation verzichtet, fällt die Entscheidung auf wenige Spiegel, die tatsächlich produziert und getestet werden. Es ist jedoch möglich, dass einer der Entwürfe ein besseres Ergebnis geliefert hätte und aus Zeit oder Kostengründen nicht gefertigt wurde. Bei Spiegeln für omnidirektionale Kameras kann es passieren, dass durch Kombination verschiedener Formen, komplizierte Oberflächen entstehen. Bei diesen Formen kann es schwierig werden, die Form der Reflexion einzuschätzen. Hier ermöglichen Simulationen einen Zugang zu Stellgrößen, um Ergebnisse schnell zu erzielen und Formen einschätzen zu können.

Für eine omnidirektionale Kamera, mit gekauften oder improvisiertem Spiegel, macht die Simulation tatsächlich wenig Sinn. Hier gibt es nur wenig Einfluss auf das, was die Kamera später sieht. Kameras mit improvisierten Spiegel, werden in dieser Arbeit außen vor gelassen. Die Gründe hierfür sind in Kapitel 2.4 zu finden.

Ist die Entscheidung getroffen einen Spiegel selber herzustellen, ist es möglich, die Eigenschaften mathematisch zu ermitteln. Das entstehende Bild auf dem Spiegel wird durch Verzerrungen abstrakt und ist fürs menschliche Auge ungewohnt. Dadurch wird es schwierig bestimmte Eigenschaften festzulegen. Es ist jedoch gut möglich, mathematisch zu bestimmen, welchen Bereich des Spiegels die Kamera, bei vorgegebenem Abstand, sehen kann. Ebenfalls unproblematisch ist es, den Sichtradius zu bestimmen. Die Simulation benötigt nur diese Größen, um ein komplexes System zu beschreiben. Alles Weitere kann fertigen Programmen entnommen werden. Dadurch vereinfacht sich die Berechnung der Spiegel sehr stark. Die vom Roboter zu interpretierenden Bilder können leicht erzeugt und ausgewertet werden. Komplizierte Berechnungen für Reflexionen und Objektgrößen auf dem Spiegel können entfallen. Zusätzlich können für spezielle Anwendungsgebiete verschiedene Formen

praktisch begriffen und ausprobiert werden. Objekte wie Spiegelhalterungen können gleich bei der Simulation berücksichtigt und in das Gesamtbild eingebunden werden.

Diese Arbeit beschäftigt sich mit kostengünstigen omnidirektionalen Kameras. Diese sollen in späteren Projekten eingebunden werden. Das Thema ist dann nicht mehr die Kamera, sondern eine Anwendung der Kamera. Aus diesem Grund ist es sinnvoll die Parameter zur Konstruktion der Kamera so einfach wie möglich zu halten. Eine erweiterbare Simulation mit vorgegebenen Grundformen, ist hier ein sinnvoller Ansatz, da komplizierte mathematische Zusammenhänge abstrahiert werden und lediglich einfache Stellgrößen errechnet werden müssen. Sollten die bisher verfügbaren Formen nicht ausreichen, ist es immer noch möglich, die Mathematik hinzuzuziehen, das Resultat zu überprüfen und zu verbessern. Einfach zugängliche Simulationsergebnisse bestehender Spiegel können hier die Grundlage für Überlegungen bilden.

Aus diesem Grund ist eine Simulation trotz des zusätzlichen Aufwandes und der beschriebenen Gefahren ein sinnvolles Mittel, um kostengünstige omnidirektionale Kameras für mobile Roboter zu konstruieren und ist Teil dieser Arbeit. Der nächste Teil dieses Kapitels beschäftigt sich mit dem ersten Schritt einer eigenen Simulation im Rahmen dieser Arbeit.

3.3 Anforderungen

Der erste Schritt zu einer Simulationssoftware ist die Klarstellung des Aufgabengebietes. Die in dieser Arbeit entstehende Simulation soll es Studenten ermöglichen, in sehr kurzer Zeit, eine Idee davon zu bekommen, wie eine omnidirektionale Kamera für ihre Aufgabenstellung aussehen muss. Die Simulation soll bereits nutzbare Resultate liefern, auf deren Basis nach kurzer Zeit eine omnidirektionale Kamera gefertigt werden kann. Aus diesem Grund wird trotz der in Kapitel 3.2.3 beschriebenen Gefahren, in dieser Arbeit ein „Push Button System“ angestrebt. Hierbei soll anhand eines gegebenen Szenarios und einer kleinen Auswahl an möglichen Spiegeln, ein möglichst realitätsnahes Bild geschaffen werden. Realitätsnah bedeutet, dass Objekte für die Simulation abstrahiert vorliegen und die Simulation ein Kamerabild mit diesen Objekten physikalisch korrekt darstellt. Nach diesem Bild soll eine Entscheidung für eine Spiegelform, mit bestimmten Eigenschaften, möglich sein. Zusätzlich muss die Simulation alle für die Fertigung des Spiegels relevanten Daten zur Verfügung stellen.

Die Simulation sollte Experimente mit unterschiedlichen Variablen vereinfacht ermöglichen. Zusätzlich ist die Reproduzierbarkeit vorheriger Läufe zu Vergleichszwecken wichtig. Da eine komplett dynamische Simulationsumgebung den Zeitrahmen dieser Arbeit übersteigen würde, wird auf die Feinabstimmung aus der GUI heraus verzichtet und die Zahl der simulierbaren Arten von Spiegeln klein gehalten. Dennoch muss es möglich sein, die Simulation

bei einem tieferen Einsteigen in die Materie, nutzen zu können. Dafür muss es eine Möglichkeit geben, besondere Feinabstimmungen, weitere Kombinationen und ganz neue Spiegelformen zu erschaffen und mit der Simulation testen zu können. Die Simulationssoftware soll also erweiterbar sein.

Für das Ergebnis reicht es nicht aus, die Reflexion in einem Spiegel zu kennen. Beim Bau einer omnidirektionalen Kamera für mobile Roboter ist besonders die Größe der Kamera von Bedeutung. In der Regel wird eine Roboterplattform bestehen. Die Sensorik soll in diesem Fall durch eine omnidirektionale Kamera ergänzt werden. Das begrenzt die Größe der Kamera und macht diese Größe zu einem entscheiden Faktor. Aus diesem Grund ist es wichtig, nach einem Simulationslauf eine direkte Gegenüberstellung zwischen der Sicht der Kamera und der Größe des Aufbaues zu erzeugen.

Zu den funktionalen Anforderungen kommt die Wartbarkeit und Erweiterbarkeit der Simulationssoftware. Hierfür ist es zielführend die Software in sinnvolle Teile zu gliedern, die miteinander arbeiten. Dieses erleichtert die Fehlersuche sowie die Fehlervermeidung und das Anpassen des Programmes.

Mit den hier aufgeführten Anforderungen geht es weiter mit dem Entwurf und die Implementation der Simulationssoftware.

3.4 Entwurf der Simulationssoftware

Ein komplexeres Computerprogramm sollte immer in verschiedene Komponenten aufgeteilt werden. Dieses Kapitel beschäftigt sich mit dem Entwurf des Aufbaus der Simulationssoftware und beschreibt ihre einzelnen Komponenten. Begonnen wird hierbei mit einer abstrakteren Darstellung in Abbildung 3.1. Die Hauptkomponenten sind in verschiedenen Farben gehalten, um die Zugehörigkeit einer Funktionalität zu einem bestimmten Bereich zu verdeutlichen.

Als Ausgangspunkt für eine Software gibt es immer ein Hauptprogramm. Diese in grau gehaltene Komponente ist der Einstiegspunkt in das Programm. Hier werden Übergabeparameter beim Programmstart geprüft und auf Basis der Überprüfung eine Reihe von Komponenten aktiviert. Bei einer Simulationssoftware, wie sie in dieser Arbeit entstehen soll, werden zwei weitere Komponenten benötigt. Diese bestehen aus einem Modul zur Erzeugung von Spiegeln und einer grafischen Benutzerschnittstelle (GUI). Die Komponenten, welche mit der GUI zu tun haben, werden in den folgenden Abbildungen blau dargestellt, die Komponenten der Spiegelerzeugung grün.

Zwischen den Komponenten sorgen Schnittstellen für den notwendigen Datenaustausch. Die Spiegelerzeugung sollte hierbei über eine einzige Schnittstelle verfügen, welche von

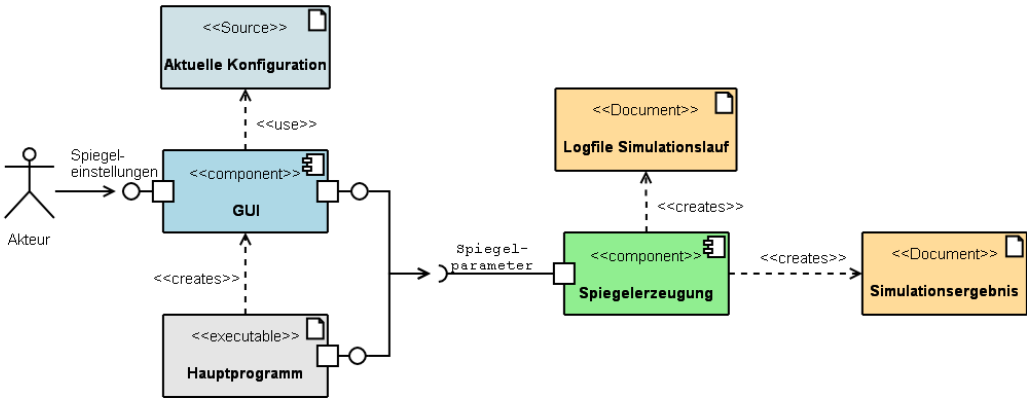

Abbildung 3.1: Die Simulationssoftware benötigt drei Hauptkomponenten und eine Konfigurationsdatei. Als Ergebnis entstehen ein Logfile und das Simulationsergebnis.

der GUI- und der Hauptprogrammkomponente genutzt wird. Das erhöht die Wartbarkeit und reduziert die Fehleranfälligkeit.

Diese Aufteilung in GUI und Funktionalität bietet bei der Entwicklung der Simulationssoftware mehrere Vorteile. Ein wichtiger Vorteil besteht darin, dass zunächst ein sehr einfaches Programm geschaffen werden kann, welches frühe Tests ermöglicht. Man erhält eine lauffähige Software und kann die Qualität der Ergebnisse prüfen, bevor viel Zeit in die Entwicklung einer grafischen Benutzerschnittstelle investiert wird. Durch die Trennung von GUI und Funktionalität bleibt die GUI austauschbar, da sie nur auf definierte Schnittstellen der Software zugreift. Damit können Beschränkungen einer ersten GUI beseitigt werden, ohne die Notwendigkeit die Simulationssoftware komplett neu zu schreiben. Das erfüllt die Forderung nach Erweiterbarkeit.

Das durch die Aufteilung im ersten Schritt entstandene Kommandozeilenprogramm kann auch nach Implementierung einer GUI nutzbar gehalten werden. Dadurch entsteht zusätzliche Flexibilität. Sie erlaubt die fertige Simulationssoftware mit einer Oberfläche für bequeme Nutzereingaben zu starten oder das Programm in Skripte und andere Programme zu integrieren. Der für den Endnutzer wichtigere Punkt ist die höhere Performance einer reinen Konsolenanwendung. Diese muss keine Rechenzeit für die GUI aufbringen. Dadurch kann etwas Zeit pro Bild gespart werden. Sollte ein späterer Anwender einige tausend Bilder erzeugen wollen, zum Beispiel für eine Animation, fällt das ins Gewicht.

Das Kommandozeilenprogramm benötigt mindestens zwei Komponenten der Software. Das Hauptprogramm und die Spiegelerzeugung. Die Spiegelerzeugung ist in der Lage aus verschiedenen Parametern zu einem Simulationsergebnis zu kommen. Dieses besteht aus den

verschiedenen, simulierten Spiegeln eines Laufes. Zusätzlich erstellt die Spiegelerzeugungskomponente ein Logfile zum Simulationslauf. In diesem befinden sich alle Daten, welche für die Reproduzierbarkeit des Laufes benötigt werden. Des Weiteren enthält das Logfile genaue Angaben darüber, wie ein spezieller Spiegel kreiert wurde. Diese Angaben können genutzt werden, um später einen bestimmten Spiegel zu wählen und zu verändern. Damit ist eine feinere Anpassung des Simulationsergebnisses möglich.

Eine sinnvolle Simulationssoftware für omnidirektionale Kameras benötigt viele Parameter. Diese Parameter über eine Kommandozeile einzugeben ist sehr umständlich und fehlerträchtig. Das macht eine grafische Benutzerschnittstelle notwendig. Hier sollten auf möglichst übersichtliche und einfache Weise Parameter für das Programm definiert werden können. Zusätzlich sollten vorhandene Standardwerte vorgehalten werden. Einen Nutzer wird es nach kurzer Zeit stören, muss er bei jedem Programmstart die gleichen Konfigurationen eingeben. Darum wird eine Konfigurationsdatei benötigt, welche diese Werte speichert und beim Programmstart wieder herstellt. Diese kann auch für eine spätere Reproduzierbarkeit von Ergebnissen genutzt werden. Die GUI sollte in der Lage sein die Ergebnisse zu präsentieren. Gleichzeitig sollte eine einfache Erweiterbarkeit der Oberfläche gewährleistet sein. Ist die einfache Erweiterbarkeit nicht gegeben, könnte es passieren, dass zwar weitere Funktionalität in die Simulationssoftware integriert wird, diese jedoch nicht über die GUI verfügbar gemacht wird. Zu dieser Erweiterbarkeit gehört die Abstraktion der Dateizugriffe und des Parsens für Konfigurationsdateien. Im Gegensatz zur Kommandozeilenanwendung ist es bei einem Programm mit GUI etwas komplizierter dem Nutzer eine Rückmeldung zu geben. Reichen beim Kommandozeilenprogramm einfache Ausgaben über fertig gestellte Spiegel, so müssen bei der Anwendung mit grafischer Benutzerführung Schaltflächen gesperrt und auf andere Art Rückmeldungen kreiert werden. Diese Rückmeldungen benötigen die oben aufgeführte Rechenzeit.

Das Ergebnis der Simulation ist in jedem Fall eine Reihe von Bildern und eine Liste von Informationen. Die Simulationssoftware nutzt für die Erzeugung der Ergebnisse immer die gleichen Komponenten des Programmes. Dabei ist es unerheblich, ob mit GUI oder von der Kommandozeile gestartet wurde.

Es gibt auf dem Markt verschiedene Bildbetrachtungsprogramme. Aktuelle Betriebssysteme bieten eine Miniaturansicht, die es erlaubt alle Bilder in einem Ordner zu betrachten. Ein Bildbetrachter ist darum nicht direkt Teil der Simulation. Es ist dennoch eine Überlegung wert, ob die GUI bereits eine Vorschau anzeigen sollte. Das hat den Vorteil, dass hier eine spezielle Sortierung implementiert werden kann. Es ist sinnvoll die wichtigsten Informationen, die zum Bau des Spiegels benötigt werden, in dem jeweiligen Bildnamen zu integrieren. Das bietet zum einen den Vorteil, dass die Bilder automatisch über einen sinnvollen Namen verfügen und nicht einfach durchnumeriert sind, zum Anderen gibt es keinen einfacheren Weg, um die Beziehung zwischen Daten und Bild herzustellen. Verwechslungen durch Verrutschen in Zeilen etc., wird vorgebeugt, die Ergebnisse sind schneller greifbar. So stehen auch nach

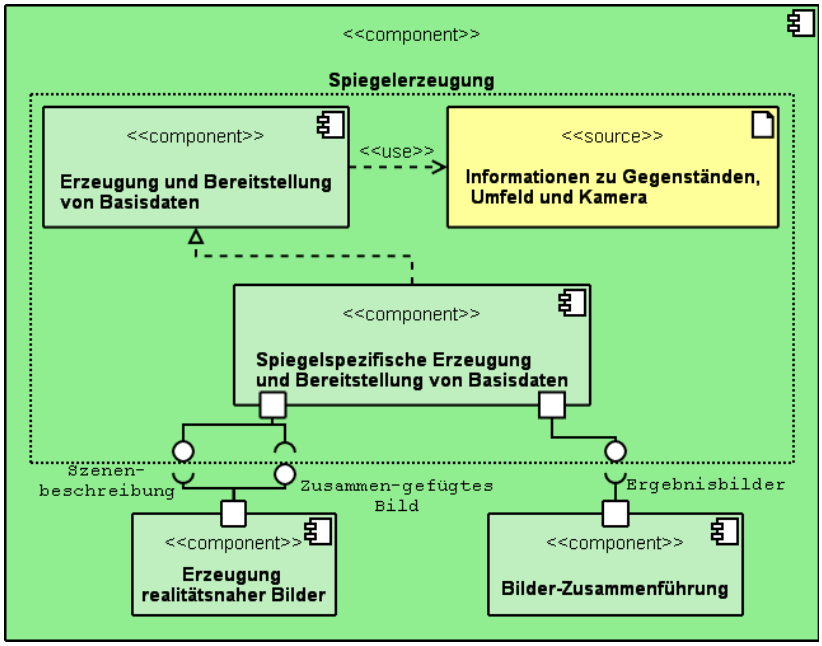

Abbildung 3.2: Aufteilung der Spiegelerzeugung

einfachem Lauf des Simulationsprogrammes aus der Kommandozeile heraus, sofort alle relevanten Daten auf einem Blick bereit. Eine weitere Möglichkeit die Daten augenblicklich verfügbar zu machen, wäre es die Daten direkt in das Bild hinein zu schreiben. Dafür könnte ein Balken unter dem Bild hinzugefügt werden.

Die Komponente für die Spiegelerzeugung wird weiter geteilt. Für die Spiegelerzeugung fallen verschiedene Aufgabengebiete an, welche in Abbildung 3.2 gezeigt werden. Hierbei ist der wichtigste Teil die Komponente zur Erzeugung realitätsnaher Bilder. Sie erzeugt die Bilder, welche das Ergebnis der Simulation darstellen. Der Teil des Systems innerhalb des gepunkteten Kastens erstellt die Informationen, welche von der Komponente zur Erzeugung realitäsnaher Bilder benötigt werden. Hier arbeiten zwei Teile zusammen. Die Komponente zur Erzeugung und Bereitstellung von Basisdaten enthält oder erzeugt Informationen, die für die Schaffung aller Spiegelformen benötigt werden. Die Komponente zur spiegelspezifischen Erzeugung und Bereitstellung von Basisdaten muss für jede neue Spiegelform ausgetauscht werden. Sie enthält ausschließlich die speziellen Informationen, die für die Erzeugung einer Spiegelform benötigt werden.

Für den Zugriff auf die Komponente „Erzeugung realitätsnaher Bilder“ gibt es zwei Möglichkeiten. Entweder es wird direkt über eine Schnittstelle auf den Algorithmus zugegriffen oder der Algorithmus in einem selbstständigen Programm implementiert. Die Nutzung eines zusätzlichen Programmes benötigt eine externe Szenenbeschreibungsdatei. Diese entspricht der Komponente „Informationen zu Gegenständen, Umfeld und Kamera. Dieser Weg wirkt

für die Implementierung sehr indirekt er bietet jedoch eine höhere Flexibilität für den Endnutzer. Durch die Nutzung einer normalen Szenenbeschreibungsdatei anstelle des direkten Zugriffes auf einen Bilderzeugungsalgorithmus, wird die in den Anforderungen beschriebene Feinabstimmung von Ergebnissen und Nutzbarkeit von komplett neuen Spiegelformen erfüllt. Es ist also möglich omnidirektionale Kameras zu simulieren, die weit über die Funktionalität eines „Push Button Systems“ hinausreichen. Zusätzlich eröffnet sich die Möglichkeit die Grundlagen des Resultates genau überprüfen und reproduzieren zu können. Das ist besonders mit der weiter oben beschriebenen Logdatei ein wichtiges Werkzeug. Aus diesem Grund wird der Umweg über eine externe Szenenbeschreibungsdatei für diese Software gewählt. Auf jeden Fall muss bei einem einfachen Lauf der Simulation über die GUI die Nutzung des Modules zur Erzeugung realitätsnaher Bilder transparent gehalten werden.

Die Szenenbeschreibungsdatei muss von der Komponente für die Erzeugung und Bereitstellung von Basisdaten bereitgestellt werden. Die in gelb gehaltene Szenenbeschreibungsdatei liegt getrennt im Dateisystem. In ihr befinden sich alle Informationen, die der Bilderzeugungsalgorithmus benötigt, um ein realitätsnahes Bild zu erzeugen.

Wie in den Anforderungen beschrieben, sollen auf Knopfdruck eine Reihe von Spiegeln simuliert werden. Mithilfe dieser Bilder soll eine Aussage darüber getroffen werden können, wie ein Spiegel geschaffen sein muss, damit er eine bestimmte Aufgabe erfüllt.

Um eine Reihe von Spiegeln erzeugen zu können, gibt es zwei Möglichkeiten. Entweder die Komponente zur Erzeugung realitätsnaher Bilder übernimmt diese Aufgabe oder die Simulation ruft diese Komponente mehrmals auf. Die in dieser Arbeit kreierte Simulationssoftware folgt dem zweiten Ansatz. Dieser hat den Nachteil, dass die Simulationssoftware außerhalb der Bilderzeugungskomponente deutlich komplexer wird. Dafür bietet er den Vorteil, dass die Szenenbeschreibung maximal einfach gehalten wird. Ein späterer Anwender muss sich damit nicht mit der Syntax von Schleifen etc. in der Szenenbeschreibung beschäftigen. Das ist für die Akzeptanz der Software wichtig. Zusätzlich wird es einem späteren Anwender dadurch vereinfacht mithilfe der Simulation schnell zu einem brauchbaren Ergebnis zu kommen.

Damit die externe Szenenbeschreibungskomponente aus Bild 3.2 von der Simulationssoftware verändert werden kann und gleichzeitig vom Anwender bequem anpassbar bleibt, muss sie weiter aufgeteilt werden. Eine sinnvolle Aufteilung zeigt Abbildung 3.3.

Hier wird die Komponente Informationen zu Gegenständen und Umfeld der Kamera in drei Komponenten aufgeteilt:

- Eine Basiskomponente „Szenenbeschreibung Gegenstände und Umfeld der Kamera“, in der sich die Beschreibung der generellen Umgebung befindet. Dazu gehören die Lichtquelle und verschiedene Objekte einer Grundszene. Objekte einer Grundszene sind Dinge, wie eine Spiegelhalterung und Objekte als Landmarken, die sowohl in

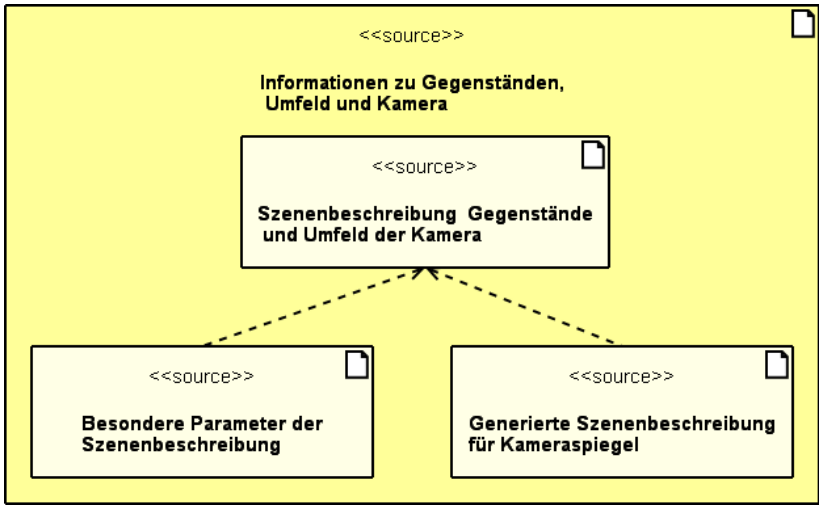

Abbildung 3.3: Aufteilung der Szenenbeschreibungsdatei in drei Teile

der Realität, als auch in der Simulation vorhanden sind. Bei diesen Objekten handelt es sich um Gegenstände, die für verschiedene Simulationsgänge gleich sind. Somit müssen diese Gegenstände nur selten verändert werden.

- Dazu kommt eine Konfigurationsdatei „Besondere Parameter der Szenenbeschreibung“, in der sich Objekte befinden, die austauschbar sind und nicht zum Szenario gehören, in dem die Kamera später eingesetzt werden soll. Das sind zum Beispiel die verwendete Kamera mit ihren Maßen und ihrer speziellen Bauform.
- Schließlich gibt es eine Datei „Generierte Szenenbeschreibung für Kameraspiegel“, in der die häufig zu verändernden Dinge stehen. Dazu gehören die Spiegel und weitere Objekte oder Variablen, die während der automatischen Simulationsgänge verändert werden.

Diese Aufteilung erlaubt eine Abstraktion von der Szenenbeschreibungsdatei der Bilderzeugungskomponente und einen Ansatzpunkt für ein externes Programm.

Für eine Umsetzung der Simulationssoftware auf dem hier beschriebenen Weg ist der Ansatzpunkt für ein externes Programm von besonderer Bedeutung. Ohne diesen Ansatzpunkt müsste die gesamte Szenenbeschreibungsdatei innerhalb der Simulationssoftware bearbeitet werden. Damit ginge jedoch Flexibilität verloren. Diese Flexibilität entsteht gerade dadurch, dass die Szenenbeschreibungsdatei direkt vom Anwender angepasst werden kann. Durch die Abspaltung eines Bereiches, der durch ein externes Programm erzeugt werden kann, ist es möglich alle anderen Bereiche der Szenenbeschreibungsdatei offen zu halten.

Die Abstraktion von der Szenenbeschreibungsdatei im Modul „Besondere Parameter der Szenenbeschreibung“, erlaubt es verschiedene Parameter außerhalb des eigentlichen Quell-

codes zu definieren. Damit ist es einem Anwender möglich, Parameter zu verändern, ohne sich in die Grundlagen einer Szenenbeschreibungsdatei einarbeiten zu müssen. Zusätzlich erlaubt die Aufteilung der Szenendatei, die Bündelung wichtiger Parameter an einem Ort. Somit muss für Veränderungen an bestimmten Eigenschaften der Szene nicht in der langen Basisdatei gesucht werden. Bei Änderungen der wichtigen Objekte der Szene reduziert sich zusätzlich die Gefahr, einzelne Einstellungen zu vergessen.

Jetzt ist es interessant, wie der Anforderung nach zwei Bildern aus einer Szene nachgekommen werden kann. Diese sind notwendig, um einen Bezug aus Kamerasicht und Größe des Spiegels herzustellen. Die Bilderzeugungskomponente sollte möglichst einfach gehalten werden. Das bedeutet allerdings, dass sie das Bild der Szene nur von einer Kamera pro Lauf zurückgeben kann. Das macht es notwendig die Bilderzeugungskomponente zwei Mal auf die Szenenbeschreibungsdatei anzuwenden. Anschließend müssen beide Bilder gemeinsam betrachtet werden können. Hier steckt ein großes Gefahrenpotenzial, da bei großer Anzahl an Bildern durchaus Verwechslungen vorkommen können. Um dieser Gefahr zu begegnen, sollte ein drittes Bild erzeugt werden. Dieses muss direkt im Bilderzeugungsprozess erstellt werden und verbindet die beiden Bilder einer Szene. Diesen Zweck erfüllt die Komponente „Bilderzusammenfügen“. Anhand des Resultates dieser Komponente kann der Nutzer eine sinnvolle Vorauswahl treffen.

Ein möglicher Lauf mit Nutzung der GUI wird in Abbildung 3.4 gezeigt.

Gestartet wird in jedem Fall mit dem grau dargestellten Hauptprogramm. Es hat in diesem speziellen Fall keine Übergabeparameter erhalten und erzeugt darum ein in blau dargestelltes GUIObjekt. Das Einlesen der Daten aus der Konfigurationsdatei wurde hier weggelassen. Es geschieht beim Erzeugen der GUI. Nach einiger Zeit hat der Anwender sich dafür entschieden genau einen Spiegel zu simulieren und den Prozess gestartet. Die GUI erzeugt dadurch ein Objekt zur Erzeugung von Spiegeldaten. Dieser Bereich wird in dem Sequenzdiagramm grün dargestellt. Bei der Erzeugung werden dem Objekt alle notwendigen Informationen übergeben, welche für die Aufgabe nötig sind. Dazu gehören der Ort, an dem sich die Szenenbeschreibungsdatei befindet und die Bereiche für die Erzeugung der Spiegel. Diese Bereiche beschreiben zum Beispiel die Spiegel mit unterschiedlichen Radien, die simuliert werden sollen. Nach seinem Start berechnet die Komponente „Erzeugung von Spiegeldaten“ die jeweiligen Parameter für die Spiegel und erstellt fertige Szenenbeschreibungsdateien. Diese werden dann der Komponente zur Erzeugung realitätsnaher Bilder übergeben, die daraus ein Bild generiert und sich selbst beendet. Ist das Bild fertig, wird die Kamera in der Szenenbeschreibung geändert. Die neue Szenenbeschreibungsdatei zeigt die Szene aus einer Draufsicht und wird der Komponente zur Erzeugung realtiätsnaher Bilder übergeben. Diese erzeugt daraufhin das zweite Bild und beendet sich abermals. An dieser Stelle macht eine Parallelisierung keinen Sinn, da die Erzeugung von Bildern selber sehr gut parallelisiert werden kann. Dazu mehr in Kapitel 3.5.1. Damit braucht diese Komponente bereits die gesamte Rechenkraft des Systems. Die fertigen Bilder werden im letzten

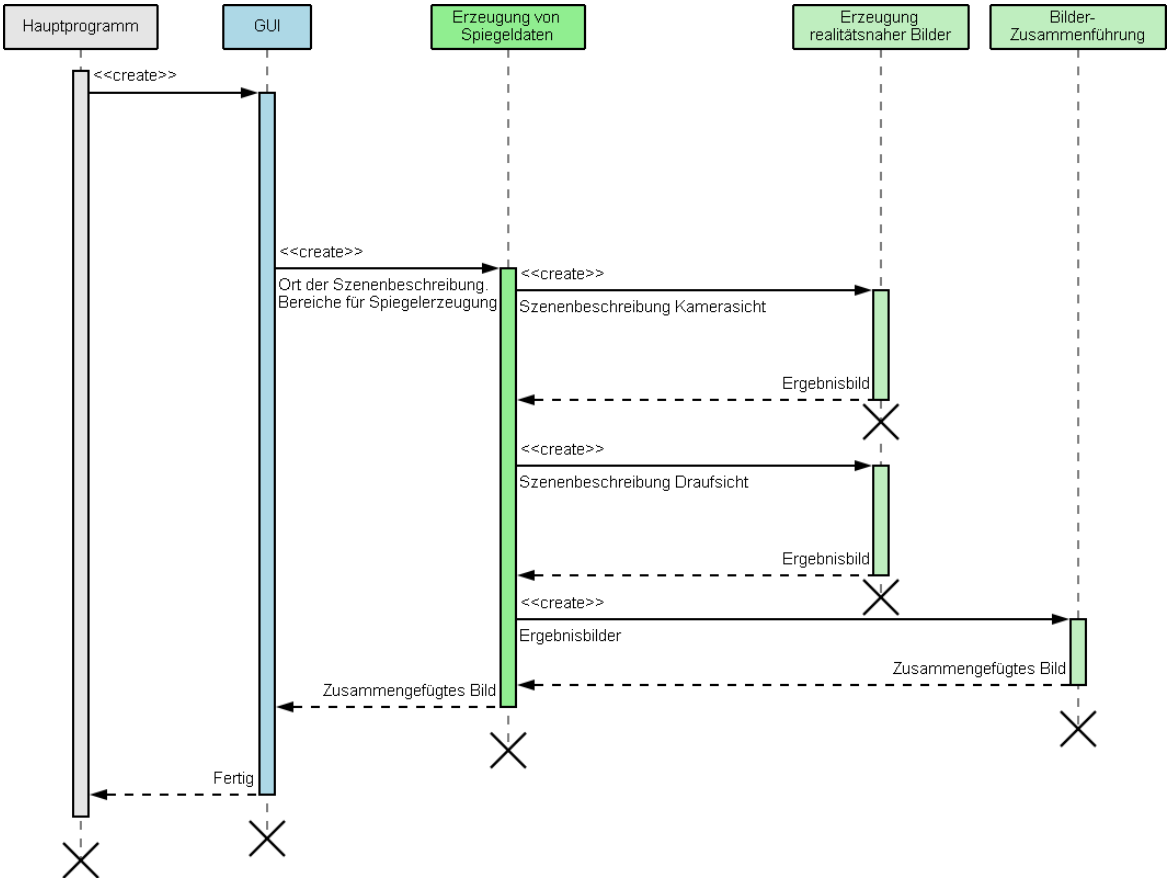

Abbildung 3.4: Beispiel eines Laufes der Software mit GUI

Schritt der Komponente „Bilder Zusammenführen“ übergeben, welche beide Bilder wie oben beschrieben in einen Bezug setzt. Anschließend übergibt sie der Komponente „Erzeugung von Spiegeldaten“ die Bilder und beendet sich. An dieser Stelle würden weitere Spiegel berechnet und der Prozess erneut beim Erzeugen der Bilderzeugungskomponente beginnen. In diesem Beispiel sollte nur ein Spiegel simuliert werden. Darum hat die Komponente „Erzeugung von Spiegeldaten“ ihre Arbeit abgeschlossen. Sie übergibt das zusammengefügte Bild an die GUI und beendet sich. Hier könnte der Anwender aus der GUI heraus weitere Spiegel simulieren. In diesem Beispiel hat er sich gegen diese Möglichkeit entschieden und das Programm beendet.

Mit diesem Entwurf der Simulationssoftware geht es jetzt in die Implementierung.

3.5 Implementation der Simulationssoftware

Der erste Schritt bei der Implementierung ist der Blick über den Tellerrand. Welche Komponenten müssen selber geschrieben und welche können aus anderen Quellen übernommen werden. Hierbei stellt sich heraus, dass es für zwei Komponenten bereits Umsetzungen gibt. Diese sind zum einen die Komponente zur Erzeugung realitätsnaher Bilder, zum Anderen die Komponente zum Zusammenfügen von Bildern.

Auf die Komponente zur Erzeugung realitätsnaher Bilder wird an dieser Stelle in den nächsten zwei Unterkapiteln näher eingegangen. Anschließend wird mit der Komponente zum Zusammenfügen von Bildern und der Konstruktion der späteren Simulationssoftware fortgefahren.

3.5.1 Raytracing

Eine sehr brauchbare Methode, um realitätsnahe Bilder künstlich zu erschaffen ist die Raytracing Technologie. Auf diese wird im Folgenden näher eingegangen. Anschließend wird im Kapitel 3.5.2 auf eine spezielle Implementierung dieser Technologie aus der Open Source Welt eingegangen.

Das Raytracing wird überall dort eingesetzt, wo besonders realistische Szenen mit beliebigen Materialien erzeugt werden sollen. Ein solches Bild mit Hilfe von Raytracing zu erzeugen, kann sehr zeitintensiv sein. Die benötigte Zeit hängt hierbei stark von der Komplexität der Szene ab. Da es unter Umständen lange dauern kann, bis ein Bild fertig erstellt wurde, wird Raytracing bisher hauptsächlich für die Filmproduktion eingesetzt. Im Bereich der virtuellen Realität konnte es sich noch nicht durchsetzen. Dennoch wird daran weiterhin gearbeitet. Die Universität des Saarlandes, welche die Grundlagen für das Projekt OpenRT gelegt hat,

möchte auf der CeBIT 2007, ihre Forschungen auf dem Gebiet des Realtime Raytracings vorstellen golem OpenRT. Auf einer älteren Version des OpenRT Projektes bauen die Arbeiten von Daniel Pohl, welcher sein Informatik Studium an der Universität Erlangen in dem Thema abgeschlossen hat Pohl (2006).

Dieses Kapitel stützt sich auf die Werke „3D-Grafik und Animation“ Plenge (1989) und „Fotorealismus und Ray Tracing in C“ Christopher D. Watkins (1993). Diese bieten eine detaillierte Beschreibung für die Implementierung von Raytracing Programmen. Das macht sie zu idealen Anlaufstellen, um die Funktionsweise solcher Programme zu verstehen.

Raytracing ist eine Technologie, mit deren Hilfe man realistische, drei dimensionale Abbildungen erzeugen kann. Übersetzt bedeutet Raytracing „Strahl Verfolgung. Das menschliche Auge nimmt die Strahlen wahr, die es von einer Lichtquelle aus erreichen. Beim Raytracing wird die umgekehrte Richtung verwendet. Strahlen werden vom Auge des Betrachters oder einer Kamera zur Lichtquelle hin verfolgt. Damit wird gewährleistet, dass lediglich die Strahlen berechnet werden, die für das Bild relevant sind (Plenge, 1989, S. 264).

Die realistische, drei dimensionale Abbildung von Objekten, wird mithilfe der Vektorrechnung und grundlegender Gesetze der Optik erreicht. Strahlen können reflektiert, gebrochen, gestreut und gebündelt werden. Diese mathematischen Grundlagen können von einem Algorithmus beschrieben werden. Für die Erzeugung der Abbildung gibt es drei Ebenen. In der Mitte befindet sich die Projektionsebene. Diese Ebene besteht aus einer bestimmten Anzahl von Feldern oder Pixeln. Sie ist praktisch der Ausschnitt des fertigen Bildes. Der Betrachter befindet sich vor der Projektionsebene und kann ein Auge oder eine Kamera sein. Aus diesem Grund spricht man vom Betrachter auch als „Augpunkt“. Er legt den Winkel der Strahlen und damit das sichtbare Bild, auf die dritte Ebene fest. Die dritte Ebene ist die Szene, welche dargestellt wird. Sie besteht aus Objekten und einer oder mehreren Lichtquellen. Die drei Ebenen und die in der Szene enthaltenen Objekte werden in einer Szenenbeschreibung definiert. Aus ihr muss der Raytracingalgorithmus die Farbe für jedes Feld der Projektionsebene ermitteln und damit das spätere Bild generieren.

Einige wichtige Eigenschaften der Objekte in einer Szene sind in folgender Aufzählung aufgeführt.

- Augpunkt: Enthält alle Informationen zur Bestimmung der Richtung der Strahlen und der Projektionsfläche.
- Projektionsfläche: Enthält die Größe der Projektionsfläche in Pixeln für die x- und y-Richtung.
- Lichtquellen: Beinhalten unter anderem Informationen zur Position, Strahlrichtung, Abstrahlwinkel und Farbe.
- Szenenobjekte: Sind zusammengesetzt aus „Primitiven“ und beinhalten Informationen zur Position, Farbe und Eigenschaften ihrer Materialien.

Primitive: Sind Objekte wie Kugeln, Kegel, Würfel oder andere Polygone. Sie stellen die Basis für kompliziertere Objekte.

Ein möglicher Raytracingalgorithmus ist rekursiv und beinhaltet fünf Schritte.

- Im ersten Schritt, wird ein Strahl vom Betrachter durch die Projektionsebene verfolgt, bis er ein Objekt trifft. Bei dem Objekt kann es sich um einen Teil der Szene, eine Lichtquelle oder den Hintergrund handeln. Handelt es sich um eine Lichtquelle oder den Hintergrund, nimmt das Feld dessen Farbe an. Trifft der Strahl auf einen Teil der Szene, wird der Schnittpunkt bestimmt und zum zweiten Schritt übergegangen.
- Der zweite Schritt besteht darin, die Farbe der Objektfläche im Schnittpunkt zu bestimmen. Hierfür werden die Eigenschaften des Objektes und der Lichtquellen hinzugezogen, was in einem nächsten Schritt rekursiv bearbeitet werden kann.
- Im dritten Schritt werden Strahlen vom vorher berechneten Schnittpunkt zu den Lichtquellen der Szene geschickt. Damit wird die Helligkeit und Farbe des Punktes bestimmt und eine diffuse und spiegelnde Beleuchtung erzeugt.
- Schritt vier überprüft die Eigenschaften der Objektoberfläche.

  Handelt es sich um eine reflektierende Oberfläche, so wird ein neuer Strahl nach den Reflexionsgesetzen der Physik berechnet. Auf diesem neuen Strahl beginnt der Algorithmus mit dem ersten Schritt.

  Handelt es sich um eine transparente Oberfläche, so wird der Strahl nach den Brechungsgesetzen gebrochen. Dabei entsteht, wie bei der reflektierenden Oberfläche, ein neuer Strahl. Auf diesen neuen Strahl beginnt auch hier der Algorithmus mit dem ersten Schritt.
- Der fünfte und letzte Schritt beinhaltet das Summieren der Farben aller Substrahlen eines Schnittpunktes. Die Summe beschreibt die Farbe des Pixels des berechneten Strahles. Handelt es sich um einen Strahl, der vom Betrachter auf die Projektionsebene stößt, so beschreibt die Summe die Farbe der Fläche auf der Projektionsebene. Der rekursive Algorithmus terminiert für diesen Pixel in der Projektionsebene.

Dieser Algorithmus kann für jedes Feld der Projektionsebene angewandt werden. Da er in sich abgeschlossen ist, ist es trivial Raytracing zu parallelisieren. Die Projektionsfläche muss für das Parallelisieren lediglich in mehrere Teile geteilt werden, die Teile können dann an unterschiedliche Rechner oder Prozessoren zur Bearbeitung übergeben werden. Am Ende werden die fertigen Teilstücke zu einem Gesamtbild zusammengesetzt. Dieses unterstützt die Aussage in Kapitel 3.4, dass eine Parallelisierung zur Erzeugung von zwei Bildern zur gleichen Zeit keinen Sinn macht.

3.5.2 POV-Ray

Eine mögliche Implementierung der Raytracing Technologie ist POV-Ray. POV-Ray bietet einige sehr interessante Vorzüge gegenüber anderen Raytracing Programmen. Zum Einen ist POV-Ray Open Source, damit ist es frei zugänglich und kostenlos. Der offene Quellcode erlaubt es theoretisch Raytracing direkt im eigenen Programm zu integrieren. Hinzu kommt, dass POV-Ray für alle wichtigen Plattformen verfügbar ist und eine weite Verbreitung genießt.

Für POV-Ray gibt es von verschiedenen Drittherstellern grafische Benutzerschnittstellen. Diese haben einige Vor- und Nachteile. Ein Vorteil ist, dass man theoretisch mit der Arbeit beginnen kann, ohne sich in eine Beschreibungssprache einarbeiten zu müssen. Zudem ist es sehr schnell möglich zu ansehnlichen Ergebnissen zu kommen. Dieser Vorteil relativiert sich jedoch sehr bald, wenn man die Oberfläche einrichten und bedienen muss. Die meisten Programme zur Erstellung von 3D-Szenen haben den Nachteil, dass sie in der Bedienung sehr komplex sind. Das bedeutet, man muss sich erst eine gewisse Zeit einarbeiten. Das Gleiche gilt für Programme, welche 3D-Szenen anhand einer Beschreibungsdatei erstellen. Für Animationen oder Bilder reicht es aus, ungefähre Positionen zu finden. Objekte werden nach optischen Gesichtspunkten zusammengefügt. Wenn sie dem ästhetischen Empfinden des Künstlers entsprechen, stimmt die Position. Die meisten kostenlosen Programme, für dreidimensionale Kreationen, sind für diesen Aufgabenbereich gemacht. Für die Simulation im Rahmen dieser Arbeit ist das nicht ausreichend. Ein zentraler Punkt ist die Reproduzierbarkeit. Es soll ein möglichst genaues Abbild der entscheidenden Gegebenheiten der Wirklichkeit geschaffen werden. Dort beginnen die Schwierigkeiten bei den grafischen Bedienoberflächen der Raytracingprogramme. Bei CAD-Programmen ist es möglich, Objekte sehr genau anhand eines Maßstabes zu positionieren. Allerdings sind die Ergebnisse dieser Programme für die Simulation oft nicht zu gebrauchen. Diese Programme sind auf die Konstruktion von Objekten spezialisiert. Physikalisch richtige Bilder des fertigen Objektes sind hier oft nur in sehr teuren Versionen zu finden. Oft muss für diese spezielle Funktion, eine zusätzliche Erweiterung des CAD-Programmes erworben werden.

Der Vorteil eines Programmes ohne grafische Bedienoberfläche liegt darin, dass man sehr genau weiß, an welcher Position sich Objekte befinden. Da die Szenenbeschreibung in einem für Menschen lesbaren Format sein muss, ist es möglich sich Änderungen durch externe Programme anzusehen und zu bewerten. Falls nötig können an dieser Stelle auch Änderungen an der generierten Datei vorgenommen werden. Das macht die Möglichkeit Szenen in Form von Text zu beschreiben deutlich flexibler, als die Nutzung eines Programmes, welches nur über eine GUI bedient werden kann. Diese Flexibilität war einer der Ausschlag gebenden Punkte für die Verwendung von POV-Ray in dieser Arbeit. Genau dieser Punkt ist für die Einbindung in die Simulationssoftware und spätere Veränderung der Szene wichtig.

Der zweite Punkt ist der geringe Aufwand sich in die Beschreibungssprache von POV-Ray einzuarbeiten. Für jemanden, der programmieren kann, stellt die Einarbeitung keine große Hürde da. Die Sprache unterstützt Schleifen und Makros, was auch kompliziertere Gebilde erlaubt. Zusätzlich handelt es sich bei POV-Ray um etablierte Software mit starker Benutzerbasis. Es gibt viele frei zugängliche Werkzeuge, Texturen, Modelle, Szenen, Anleitungen und Anlaufstellen für Fragen.

3.5.3 Konkreter Aufbau der Simulationssoftware

Mit den in Kapitel 3.5.2 beschriebenen Eigenschaften passt POV-Ray ideal in das Konzept und kann als Komponente für die Erzeugung realitätsnaher Bilder verwendet werden. Wie oben beschrieben gibt es POV-Ray für die meisten großen Betriebssysteme wie Windows, MacOSX und Linux. Leider scheint POV-Ray lediglich unter Linux die Möglichkeit der Parameterübergabe zu geben. Die Windows und MacOSX-Versionen von POV-Ray beinhalten eine eigene IDE. Das ist zwar für das Erstellen von Bildern hilfreich, jedoch schließt es externe Anwendungen oder Skripte aus. Da die Parameterübergabe als Schnittstelle für die Simulation essenziell ist, wird die dieser Arbeit erstellte Software ausschließlich unter Linux funktionieren.

Bei der Nutzung des Raytracing Programmes POV-Ray, sollte darauf geachtet werden, dass die Simulation mit einem normal installierten POV-Ray funktioniert. Das bedeutet, die Simulation nutzt eine übliche Szenenbeschreibungsdatei. Das erlaubt die in den Anforderungen und Entwurf beschriebene Feinabstimmung von Ergebnissen und Nutzbarkeit von komplett neuen Spiegelformen. Zusätzlich kommt es dem Wunsch nach die Simulationsergebnisse leicht nachvollziehbar zu machen. So können Codeteile im Logfile abgelegt sein, die zur Überprüfung in die Szenendatei eingefügt werden. Wird diese Szenenbeschreibungsdatei mit POV-Ray direkt genutzt, muss ein identisches Bild entstehen.

Für das Zusammenfügen von Bildern gibt es das Open Source Programm ImageMagick ImageMagick. ImageMagick ist ein Programm für die Kommandozeile und erlaubt verschiedene Bildmanipulationen. Es gibt unter anderem Portierungen für Windows, Linux, MacOSX, BSD und BeOS/Zeta. Dadurch, dass es sich hierbei um ein Komandozeilenprogramm handelt, ist es kein Problem dieses aus einer eigenen Anwendung heraus aufzurufen.

Nachdem alle Teile identifiziert wurden, für die bereits fertige Lösungen bestehen, wird jetzt auf die eigentliche Implementierung eingegangen. Hier konnten die oben genannten Punkte weitestgehend umgesetzt werden.

Für die Trennung von GUI und Funktionalität sind, wie in Abbildung 3.5 dargestellt, zwei Packages geschaffen worden. Dabei stellt das Package „MirrorMaker“ alle Klassen zur Verfügung, die für das Kommandozeilenprogramm benötigt werden. Das Package „GUI“ bein-

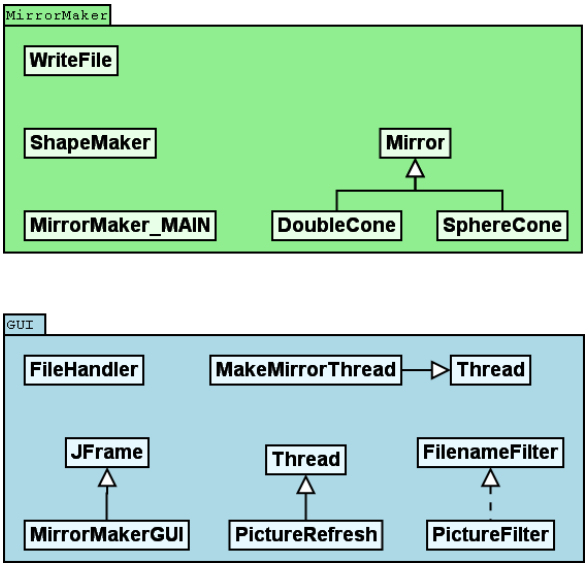

Abbildung 3.5: Aufteilung der Simulationssoftware in GUI und Funktionalität

haltet alle Klassen, die für die grafische Benutzerschnittstelle des Programmes notwendig sind.

Das Kommandozeilenprogramm MirrorMaker besteht aus sechs Klassen.

- Die Klasse „WriteFile“ ist eine sehr einfach gehaltene Klasse. Objekte dieser Klasse bekommen bei der Erzeugung einen String übergeben. Dieser enthält einen kompletten Pfad und den Dateinamen für eine Datei in die mit Hilfe des Objektes geschrieben werden kann. Die Klasse enthält alle wichtigen Exceptionhandler und gibt Fehlermeldungen auf der Konsole aus.
- Die Klasse „ShapeMaker“ ist für die Berechnung von Spiegelformen zuständig. Sie enthält zwei verschiedene Arten von Methoden. „getShape“-, bzw. „getPartofShape“-Methoden beinhalten Berechnungen, die für Spiegelformen notwendig sind. Ein Beispiel hierfür wäre die Höhe eines Kegels mit 32°. „getPovShape“-Methoden geben einen String zurück, der genau einer Beschreibung des entsprechenden Spiegels für die Szenendatei entspricht.
- Die Klasse „Mirror“ ist die Basisklasse für alle Spiegel, die mit der Simulation erstellt werden. Sie stellt viele, für das Erstellen der Szenendatei benötigten, Konstanten zur Verfügung. Dazu bietet sie Methoden für alle Aufgaben, die zur Erstellung der späteren Bilder notwendig sind. Zu den Aufgaben gehören der Zusammenbau der Dateinamen für die Simulationsergebnisse, der Aufruf von POV-Ray und die zur Verfügung stehenden Informationen zur Auflösung oder des Kamerawinkels. Die Klasse Mirror ist nicht für den Aufbau der Spiegel verantwortlich. Diesen Aufbau in Form einer Szenenbe-

schreibung, bekommen ihre Methoden als String übergeben. Die Klasse Mirror ist als Superklasse der verschiedenen Spiegelformen konstruiert worden. Die Idee dahinter ist eine klare Trennung zwischen Erzeugung der Spiegel und Informationen zum Projektpfad und Namen von Simulationsergebnissen. Um diese Informationen aus der Spiegellogik raus zu halten, ist es sinnvoll, diese in eine Superklasse zu kapseln. Das verhindert einen späteren Wildwuchs an Helferklassen. Damit sorgt es dafür, dass zum Beispiel Veränderungen an der Art, wie Simulationsergebnisse abgelegt werden sollen, global für alle Spiegel gelten.

- Die Klassen „DoubleCone“ und „SphereCone“, sind von der Klasse Mirror abgeleitet. Sie beinhalten die Brechungsgrundlagen und Methoden, um Kegel und Doppelkegel, beziehungsweise Kugeln und Kugeln mit Kegel als Szenenbeschreibung zu kreieren. Damit haben Spiegel alle Werkzeuge beisammen, um aus Bereichsinformationen und Berechnungsgrundlagen, Simulationsergebnisse zu schaffen. In dieser Arbeit wurde auf die Klassen für einfache kegelförmige und einfache kugelförmige Spiegel verzichtet. Bei diesen Spiegeln handelt es sich um Sonderfälle der jeweiligen Doppelspiegel Formen. Der Kern dieser beiden Klassen ist beispielhaft für den Doppelkegel in Abbildung 3.6 gezeigt. Da sowohl beim Doppelkegel, als auch bei der Kugel mit Kegel je drei Parameter Bereiche berechnet werden können, funktionieren die Algorithmen analog zu einander.
- Mit der Klasse „MirrorMaker_MAIN“ wird das Programm gestartet. Hier werden Übergabeparameter gesichtet und Entscheidungen zum Programmstart gefällt. Wird das Programm ohne Parameter gestartet, erscheint die GUI. Andernfalls wird anhand der Parameter entweder eine Fehlermeldung ausgegeben, oder bei richtigen Parametern ein Spiegel Objekt erzeugt, welches die Spiegel generiert.

Die GUI kann nicht selbstständig aufgerufen werden und besteht aus fünf Klassen.

- Die Klasse „FileHandler“ ist deutlich komplexer, als die Datei WriteFile. Bei der GUI gibt es verschiedene Felder, die alle Informationen beinhalten. Damit diese Informationen beim nächsten Programmstart noch da sind und nicht ständig neu eingegeben werden müssen, gibt es die Klasse FileHandler. Sie beinhaltet je eine Konstante und eine Variable pro Feld der GUI. Für alle Variablen gibt es Get- und Set-Methoden in der FileHandler Klasse. Zusätzlich beinhaltet die Klasse FileHandler Methoden zum Speichern und öffnen von Konfigurationsdateien. Diese Konfigurationsdateien sind in Klartext gehalten, sollten aber nicht verändert werden. Für das Übernehmen der Informationen aus den Konfigurationsdateien in die Variablen der Klasse gibt es in der Klasse einen einfachen Parser.
- Die Objekte der Klasse PictureRefresh sind Threads, die im Actionhandler der GUI-Schaltflächen erzeugt werden. Sie erstellen ein Spiegelobjekt und überprüfen in regelmäßigen Abständen, ob neue Bilder im Bilderordner existieren. Neue Bilder werden

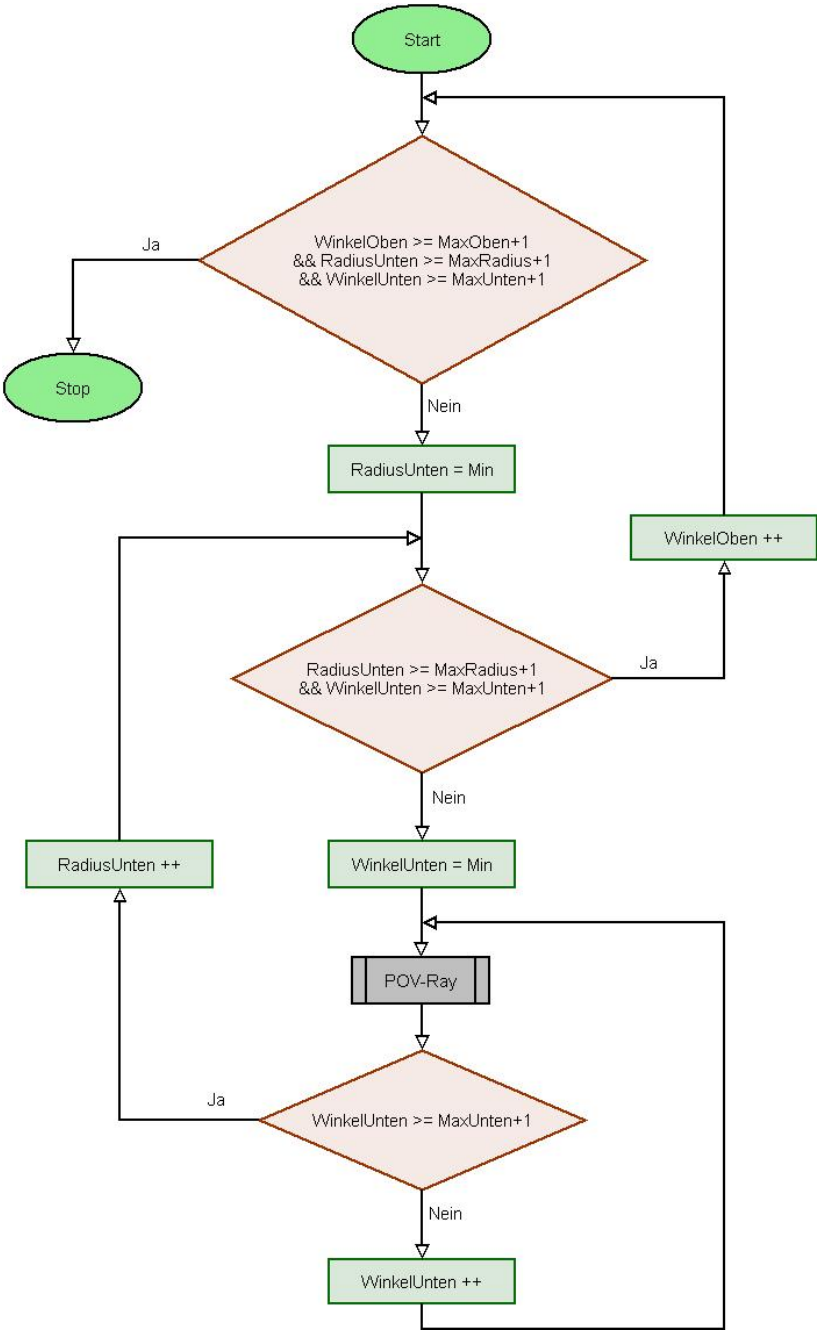

Abbildung 3.6: Flussdiagramm des Algorithmus zur Erzeugung von Doppelkegeln in der Klasse DoubleCone

in einer Liste gespeichert und in einem Bereich der GUI ausgegeben. Der Mechanismus hierfür ist aus Zeitgründen sehr primitiv und unperformant implementiert worden. Dadurch entstehen zwei Probleme. Zum Einen wird viel Zeit verschwendet, da regelmäßig durch eine komplette Liste akzeptierter Bilddateien iteriert wird. Zum Anderen gibt es noch einen kritischen Fehler beim Einfügen sehr vieler Bilder in die GUI. Hier fehlt eine Überprüfung des vorhandenen Speichers. Darum wurde die Anzahl der Bilder, die in die GUI geladen wird, willkürlich auf 100 begrenzt. Auf sehr alter Hardware mag das zu wenig sein, auf so alter Hardware möchte man jedoch keinen Raytracer einsetzen, da die Berechnung der Bilder zu lange brauchen würde.

- PictureFilter wird für die in PictureRefresh beschriebene Ausgabe der Bilder benötigt. Es handelt sich hierbei um eine einfache Klasse, welche das Interface FilenameFilter implementiert. Sie ist dafür zuständig, dass die aus POV-Ray resultierenden *.png Bilder ignoriert werden und nur die Zusammengesetzten *.jpg Bilder aus ImageMagick, in die GUI geladen werden.
- Die Klasse MakeMirrorThread ist für die Spiegelerstellung aus der GUI heraus verantwortlich. MakeMirrorThread Objekte werden im Actionhandler der GUI-Schaltflächen erzeugt. Sie bilden einen weiteren Handlungsstrang und erlauben es die GUI aktiv zu halten während Spiegel erzeugt werden.
- Die Klasse MirrorMakerGUI ist von JFrame abgeleitet. Sie beinhaltet die sichtbare grafische Benutzerschnittstelle mit allen Schaltflächen und Actionhandlern. Sie wurde mit Hilfe des GUI-Builders der NetBeans IDE, in Version 5.5, der Firma Sun Microsystems erstellt. Diese IDE ist kostenlos von der Internetpräsenz von Sun Microsystems zu bekommen Microsystems.

Wie schon beschrieben, werden alle Objekte aus dem MirrorMaker_MAIN Objekt des Packages MirrorMaker erzeugt. Hier gibt es eine Verzweigung, die entscheidet, ob direkt Spiegel erzeugt, oder die GUI gestartet werden soll. Die Grafik 3.7 zeigt die daraus resultierenden Beziehungen. Die Farbe der Objekte zeigt die Zugehörigkeit zu den oben beschriebenen Packages. Grüne Objekte sind aus den Klassen des Packages MirrorMaker erzeugt, blaue Objekte den Klassen des Packages GUI. Das Diagramm zeigt zwei Dinge. Zum Einen bringt das Package MirrorMaker alle Klassen mit, die für die Erstellung der Spiegel benötigt werden. Zum Anderen nutzt die Benutzerschnittstelle, welche durch das Objekt MirrorMakerGUI repräsentiert wird, die gleichen Objekte, wie das MirrorMaker_MAIN Programm. Beide Objekte stellen einen Einsprungpunkt in die Erzeugung von Spiegeln dar.

Alle Objekte sind von ihren aufrufenden Objekten abhängig. Kein Objekt existiert doppelt. Bei den DoubleCone Objekten findet eine Auswahl statt. Es existieren nicht alle Objekte für die Spiegelgenerierung gleichzeitig. An der Stelle im Programm wird genau die Klasse ausgewählt, die benötigt wird, um eine bestimmte Folge von Spiegeln zu kreieren. Genau

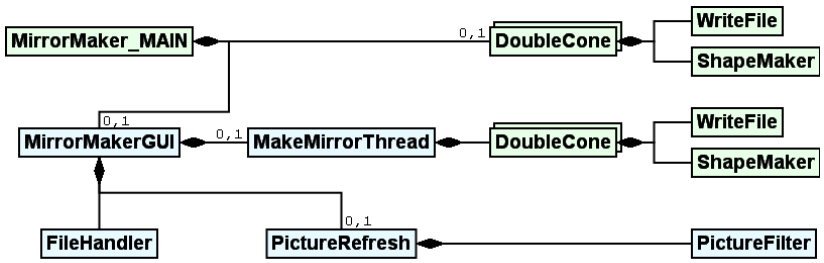

Abbildung 3.7: Beziehungen zwischen den einzelnen Objekten im laufenden Programm

aus dieser einen benötigten Klasse, wird ein Objekt erzeugt. Es wird dort aus einer Reihe von Spiegeln ausgewählt.

3.6 Fazit über Entwurf und Implementation der Simulation

Die entstandene Simulationssoftware erfüllt alle wichtigen Anforderungen an den Aufbau der Simulation. Sie bietet eine schnelle Zielführung zum Bau omnidirektionaler Kameras mit Hilfe von vier Standardspiegelformen. Die Software kann auf verschiedene Weisen eingesetzt werden. Entweder als Kommandozeilen Programm oder als Programm mit grafischer Benutzerführung, wie in Abbildung 3.8 zu sehen. In beiden Fällen ist das unter der Simulation liegende POV-Ray für den Anwender transparent. Dennoch liefert die Simulationssoftware durch die Nutzung von POV-Ray als Komponente zur Erzeugung realitätsnaher Bilder dem Anwenders die größtmögliche Flexibilität. Alle Dateien sind für Menschen lesbar und können verändert und angepasst werden. Die Szenebeschreibungsdatei ist in drei Teile geteilt. Das ermöglicht einen weiteren Zwischenschritt zwischen dem sehr einfach zu bedienenden Programm und dem Einstieg in POV-Ray. Alle Ergebnisdateien sind zu POV-Ray kompatibel. Damit ist es dem Anwender möglich nach kurzer Einarbeitung in POV-Ray die Simulation beliebig anzupassen und ganz neue Formen zu bilden.

Der modulare Aufbau der Simulationssoftware vereinfacht die Erweiterung des Programmes. Für die Erweiterung ist es lediglich notwendig sich die Schnittstellen der Mirror Basisklasse anzusehen und eine neue Subklasse zu schreiben. Allerdings sind hierfür grundlegende POV-Ray Kenntnisse erforderlich, da an dieser Stelle leider auch die Daten in Form der POV-Ray Szenenbeschreibungsdateien erstellt werden müssen. Das funktioniert leider nicht sehr übersichtlich, da Strings zusammengebaut werden müssen, die sowohl POV-Ray spezifischen Code, als auch die berechneten Werte aus der Simulation beinhalten müssen.

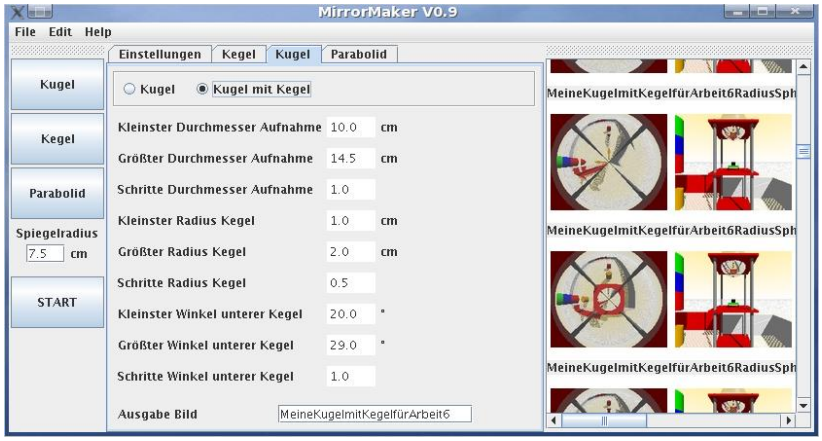

Abbildung 3.8: In dieser Arbeit entstandene Simulationssoftware

Die Ergebnisse der Simulationssoftware liegen als Bilder vor. Diese sind schnell vergleichbar und stellen eine recht gute physikalische Annäherung an die Realität dar. Durch die Bindung der wichtigsten Werte für den Spiegel im Dateinamen liegen sofort alle Daten vor, die für den Bau des Spiegels notwendig sind. Eine zusätzliche Logdatei liefert genauere Informationen über alle im Lauf erzeugten Spiegel und zeigt die POV-Ray Beschreibung der Spiegel. Damit kann das Ergebnis in POV-Ray reproduziert und für weitere Tests mit anderen Objekten etc. genutzt werden. Zusätzlich bietet die Logdatei eine weitere Möglichkeit für die Überprüfung der Werte.

Eine unschöne Sache ist die starre GUI-Vorschau. Hier könnte ein sinnvoller Mechanismus gefunden werden, der die erzeugten Resultate des Simulationslaufes dynamischer darstellen kann. Im Bereich des Anzeigens von Bildern innerhalb der GUI ist noch Platz für Verbesserungen.

Diese Simulationssoftware bietet eine generelle Lösung für die Simulation von omnidirektionalen Kameras. Damit kann zum nächsten Teil übergegangen werden, in dem sich die Simulation an einem Testlauf in der Wirklichkeit beweisen muss. Dieser nächste Schritt ist für die Bewertung der Simulationssoftware entscheidend.

4 Nutzung der Simulationssoftware

4.1 Modellbildung

Bevor mit der Simulation begonnen werden kann, ist es nötig die relevanten Parameter des Modells zu bestimmen und ein Testszenario aufzubauen. Folgende Fragen müssen mindestens beantwortet werden können:

- Welche Kamera wird eingesetzt und welche Eigenschaften hat sie?
- In welchen Entfernungen sollen Objekte erfasst werden können?
- Wie groß sind die Objekte?
- Welche Größe müssen Objekte auf dem Spiegel haben, damit sie von der Kamera erfasst werden können?
- Wie groß soll die Kamera werden?
- Gibt es erschwerende Bedingungen, die eine bestimmte Größe verlangen? Das kann besonders im Außeneinsatz passieren, wo Schmutz eine besondere Rolle spielen kann.
- Welche Punkte in der realen Welt, können als Bezugspunkte genutzt werden, um die Simulation daran zu überprüfen?

Um diese Punkte geht es in den folgenden Kapiteln. Begonnen wird mit den Eigenschaften der eingesetzten Platinenkamera.

4.2 Die Platinenkamera

Bei dieser Arbeit wird auf Bildanalyse verzichtet. Dennoch soll ein System entstehen, welches als Sensor für einen mobilen Roboter geeignet ist. Für diesen Zweck bietet es sich an, ein fertiges Kameramodul zu verwenden, welches über eine eigene Logik zur Bildanalyse verfügt. Zu Beginn dieser Arbeit gab es zwei Kameras auf dem Markt, welche zum Einen preisgünstig sind, zum Anderen bereits grundlegende Verfahren, wie Color Tracking mitbringen. Die Entscheidung ist zugunsten der CMU-CAM2 der Carnegie Mellon University

getroffen worden. Dieses Kameramodul bietet eine Plattform, die kompakte Ausmaße hat, sowie unter Anderem, Farben und Linien verfolgen kann, zu einem günstigen Preis.

Die CMU-CAM2 hat eine Größe von 55 Mal 55 Millimetern. Beim Bildsensor handelt es sich um das Modell OV6620 der Firma Omnivision. Er hat eine Chipfläche von 3,1x2,5 Millimetern, bei einer maximalen Auflösung von 352x288 Pixeln Kameramodul-Sensor. Die Brennweite der verwendeten Linse beträgt 3,8mm und bietet einen Öffnungswinkel von 45°x40° Kameramodul-Kamera.

Bei sehr geringer Auflösung ist die Kamera in der Lage Farbschwerpunkte mit bis zu 50 Frames pro Sekunde zu verfolgen. Es gibt zwei Standard Auflösungen. Einen „Low-Res-Mode“ mit einer Auflösung von 142x84 Pixeln und einen „Hi-Res-Mode“ mit einer Auflösung von 253x172 Pixeln. Im „Low-Res-Mode“ können Farbschwerpunkte mit bis zu 24 Frames pro Sekunde verfolgt werden. Im Hi-Res-Mode bleiben noch um die acht Frames pro Sekunde. Linien können selten mit mehr als acht Frames pro Sekunde verfolgt werden.

Die Kamera wird über die serielle Schnittstelle mit dem Zielgerät verbunden. Hierbei gibt es die Auswahl zwischen einem fünf Volt TTL Anschluss oder einem Anschluss zum PC, bei dem die Pegel mithilfe eines MAX232 angepasst werden. Die verfügbaren Baudraten liegen zwischen 1200 und 115200 Baud und können per Jumper auf der Platine gewählt werden.

Bilder verlassen das Modul nicht. Es ist jedoch möglich, zum Justieren der Kamera ein Standbild übertragen zu lassen. Die CMU-CAM2 wird über verschiedene Datenpakete angesprochen und gibt ihre Ergebnisse als Datenpaket zurück.

Versendet und empfangen werden die Datenpakete entweder als String oder als RAW-Paket. Standard ist die Kommunikation über Strings. Diese hat den Vorteil, dass die Funktionen der Kamera in beliebigen Terminal Programmen ausprobiert werden können. Das ermöglicht einen Vergleich einer Implementierung mit einem vorher festgelegten Schema und hilft bei der Fehlersuche. Die Kamera erlaubt das Mischen der Modi, etwa um Befehle als RAW-Pakete zu senden und Strings zu empfangen. RAW-Pakete haben den Vorteil, dass sie im Programm besser ausgewertet und verschickt werden können. Lästiges und Zeit raubendes Umwandeln von String-Zahlen in ein Computer-Lesbares-Format entfällt. Zusätzlich müssen weniger Bytes übertragen werden. Die CMU-CAM2 verschickt zum Beispiel Farbinformationen als Zahlen zwischen 16 und 240. Diese werden im RAW-Modus in einem Byte übertragen, im String-Modus bestehen sie aus drei Bytes.

Für die Kamera spricht auch eine umfangreiche Dokumentation und ein mitgeliefertes Java-Programm, mit dem die Funktionen der Kamera erprobt werden können. Dieses Java-Programm liegt im Quellcode vor und ist so angelegt, dass es sowohl unter Linux, als auch unter Windows mit der CMU-CAM2 funktioniert.

4.2.1 Bestimmung des Öffnungswinkels aus der Brennweite

Inzwischen ist bekannt, warum eine Simulation Teil dieser Arbeit ist und welche Technologie hinter der Simulation steht. Um eine Szene richtig darstellen zu können, wird noch etwas Mathematik benötigt. Damit beschäftigt sich dieser Unterpunkt.

Bei den meisten Linsen werden Angaben zur Brennweite gemacht. Für die Simulation in POV-Ray wird zur Nachbildung jedoch der Öffnungswinkel benötigt. Dieser beschreibt den Sichtwinkel einer Kamera und kann aus der Brennweite ermittelt werden. Für die Umrechnung werden die Brennweite der Linse, sowie die Diagonale des Bildsensors benötigt.

Für die Umrechnung wird etwas Trigonometrie benötigt. Das Schaubild 4.1 veranschaulicht die Rechnung.

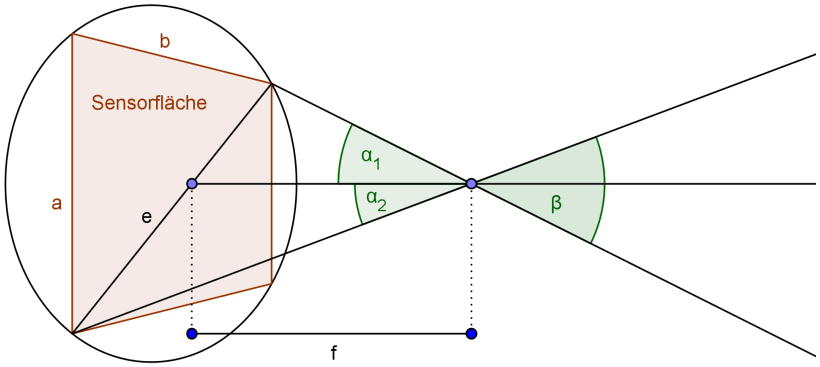

Abbildung 4.1: Größen für die Berechnung des Öffnungswinkels aus der Brennweite

Variable	Bedeutung
e	Diagonale der Sensorfläche
b	Breite der Sensorfläche
a	Höhe der Sensorfläche
f	Brennweite
beta	Öffnungswinkel

Tabelle 4.1: Definition der Variablen für die Bestimmung des Öffnungswinkels

Der Öffnungswinkel einer Kamera hängt von der Brennweite *f* der Linse und der Größe der Sensorfläche ab. Der Kreis um die Sensorfläche in 4.1, ist der Sichtbereich der Kamera. In diesen Bereich fallen die Lichtstrahlen durch die Linse auf die hintere Wand der Kamera. Die Kamera sieht jedoch nur diejenigen Lichtstrahlen, welche auf die Sensorfläche treffen. Die Sensorfläche begrenzt damit den Öffnungswinkel. Der größtmögliche Öffnungswinkel ist von

der Diagonalen *e* der Sensorfläche abhängig. Allerdings ist die Angabe des horizontalen und vertikalen Öffnungswinkels hilfreicher, da dieser Wert direkt in der Simulationssoftware verwendet werden kann. Darum wird im Folgenden mit der Breite der Sensorfläche *b* gerechnet. Die Brennweite *f* bestimmt den Abstand von der Linse, bei dem parallel einfallendes Licht auf einen Punkt gebündelt wird. Werden die Strahlen auf einen Punkt der Sensorfläche gebündelt, wirkt das Bild scharf. Die Angabe der Brennweite aus dem Datenblatt der Kamera ist also die Entfernung der Linse zur Sensorfläche in der Optik der Kamera für ein scharfes Bild. Weiterhin gibt die Brennweite den Winkel an, in der Licht auf die Rückwand der Kamera trifft.

Der Öffnungswinkel kann mithilfe einfacher trigonometrischer Berechnungen bestimmt werden. Die Formel hierfür ist in Formel 4.1 aufgeführt. Die einzelnen Variablen sind in Tabelle 4.1 beschrieben. Die Daten stammen aus dem Datenblatt zur Kameralinse (Kameramodul-Kamera) und dem Datenblatt zum eingesetzten Sensorchip (Kameramodul-Sensor).

Bestimmung des horizontalen Öffnungswinkels β:

$$\begin{aligned} \beta &= 2 \cdot \arctan\left(\frac{b}{2f}\right) \\ \beta &= 2 \cdot \arctan\left(\frac{3{,}1\,mm}{2 \cdot 3{,}8\,mm}\right) \\ \beta &= 44{,}38 \end{aligned} \tag{4.1}$$

Der Unterschied zwischen der Herstellerangabe von 45° und dem hier berechneten Öffnungswinkel von 44,38° kommt daher, dass im Datenblatt zur Kameralinse (Kameramodul-Kamera) eine Sensorfläche von 3,2mm x 2,5mm angegeben ist. Die hier verwendete CMUCAM2 verwendet jedoch ein anderes Kameramodul, welches laut Datenblatt eine Sensorfläche von 3,1mm x 2,5mm hat (Kameramodul-Sensor).

4.2.2 Praktische Bestimmung des Öffnungswinkels anhand eines Bildes

Der Öffnungswinkel einer Kamera kann wie in Kapitel 4.2.1 beschrieben rechnerisch aus den Spezifikationen der Kamera ermittelt werden. Bei Platinenkameras kommt es jedoch oft vor, dass die Angaben sehr wage oder nicht vorhanden sind. In diesen Fällen wird eine Lösung benötigt, die anhand der gegebenen Kamera den Öffnungswinkel bestimmen lässt.

Eine mögliche Lösung für dieses Problem ist ein Bild eines definierten Objektes zu machen. Hierfür eignet sich Millimeterpapier sehr gut, da es eine genaue Einteilung besitzt und sehr viele rekonstruierbare Punkte enthält, deren Position genau bekannt ist. Alternativ kann ein

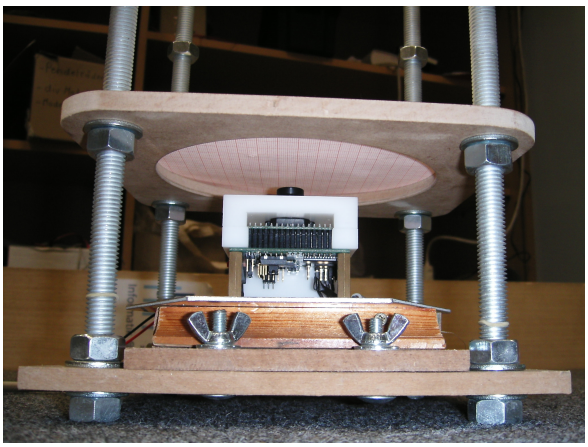

Abbildung 4.2: Aufbau zur Messung des Öffnungswinkels einer Platinenkamera

Lineal oder ein Gegenstand mit exakt definierter Größe verwendet werden. Die Lösung mit einem Gegenstand definierter Größe ist eher unpraktisch, da hier der Messaufbau so lange verschoben werden muss, bis das Objekt genau an die Ränder des Bildes reicht. Im Folgenden wird die Lösung über Millimeterpapier gezeigt.

Der Aufbau ist in diesem Fall sehr einfach gewesen. Für die omnidirektionale Kamera, welche Inhalt dieser Arbeit ist, wurde bereits ein sehr stabiles Gestell benötigt. Dieses konnte für die Messung verwendet werden. Wie Abbildung 4.2 zeigt, ist die Kugel entfernt worden. An ihre Stelle kam das Millimeterpapier. Die obere Plattform wurde auf eine Höhe von genau 160mm über dem Boden eingestellt. Die Höhe zwischen Kameralinse und dem Millimeterpapier betrug dabei ca. 20,4mm. Die Linse wurde 2,7mm aus der Halterung gedreht. Abbildung 4.3 zeigt die Sicht der Kamera.

Aus dem Bild und den Einstellungen des Gestelles kann der Öffnungswinkel analog zur Rechnung in Kapitel 4.2.1 ermittelt werden. Als Grundlage dient wieder die Skizze aus Abbildung 4.1. An Stelle der Sensorfläche steht bei dieser Rechnung das Bild aus Abbildung 4.3. Es ergibt sich für den Winkel β die Rechnung aus Gleichung 4.2.

$$\begin{aligned} \beta &= 2 \cdot \arctan\left(\frac{e}{2f}\right) \\ \beta &= 2 \cdot \arctan\left(\frac{17mm}{2 \cdot 20{,}4mm}\right) \\ \beta &= 45{,}24 \end{aligned} \tag{4.2}$$

Der Unterschied zwischen dem hier berechneten Wert und dem in Kapitel 4.2.1 bestimmten Öffnungswinkel ist in den Messungenauigkeiten zu suchen. Wird anstelle der Höhe 20,4mm

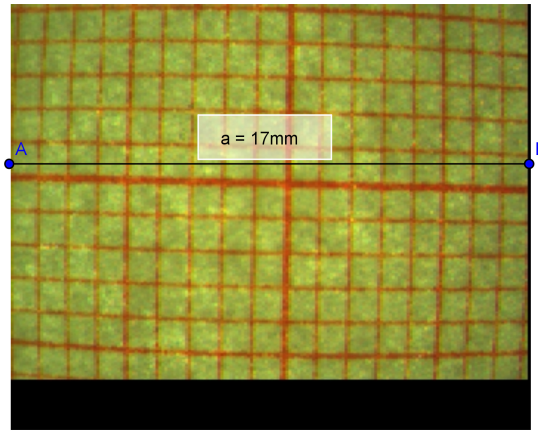

Abbildung 4.3: Messaufnahme auf Millimeterpapier 20,4mm über der Kameralinse

mit 20,84mm gerechnet, stimmen die Öffnungswinkel überein. Das ist ein Unterschied von weniger als einem halben Millimeter, was außerhalb der Genauigkeit des Messaufbaues liegt.

4.2.3 Bestimmung der Schärfentiefe

Neben dem Öffnungswinkel ist die Schärfentiefe eine weitere Größe, die von Bedeutung ist[1]. Wikipedia nennt die Schärfentiefe „die Ausdehnung des scharf abgebildeten Bereichs entlang der optischen Achse eines optischen Systems Wikipedia-Schaerfentiefe (2007). Die notwendige Berechnungsgrundlage wird in diesem Unterkapitel zur Verfügung gestellt. Die Formeln stammen aus dem deutschen Wikipedia Wikipedia-Schaerfentiefe (2007).

Diese Berechnung wird anhand des Beispiels des in Kapitel 5.3 gefertigten Spiegels und dem „Hi-Resolution“-Modus der in Kapitel 4.2 vorgeführt. Das ist sinnvoll, da mit Abnahme der Auflösung die Tiefenschärfe zunimmt. Wird ein Spiegel für eine geringere Auflösung gebaut, wird eine künstliche Beschränkung erzeugt, welche später Probleme bereiten kann.

Im ersten Schritt muss die Bildsensordiagonale d ermittelt werden, Sie wird anschließend zur Berechnung des Zerstreuungskreisdurchmessers Z benötigt. In diesem Beispiel ist die Breite des Bildsensors $b = 3,1\,mm$ und die Höhe des Bildsensors $c = 2,5\,mm$.

[1] Leider wurde sie bei der Erstellung des Simulationsszenarios nicht berücksichtigt, was zu einem unscharfen Bild der Kamera führte.

$$d^2 = b^2 + c^2 \tag{4.3}$$
$$d = \sqrt{b^2 + c^2} \tag{4.4}$$
$$d = \sqrt{(3{,}1mm)^2 + (2{,}5mm)^2} \tag{4.5}$$
$$d = 3{,}98mm \tag{4.6}$$
$$\tag{4.7}$$

Im zweiten Schritt muss der Zerstreuungskreisdurchmesser ermittelt werden. Er stellt den Bereich dar in dem das Licht auf nur einen Pixel des Bildsensors trifft. Damit ist seine Größe entscheidend für die Schärfe des Bildes. Je größer ein Pixel des Bildsensors ist, desto geringer ist die Auflösung bei gleicher Chipfläche. Gleichzeitig nimmt die optische Schärfe mit größeren Pixeln zu. Für die Auflösung der Sensordiagonalen wird die kleinere Seitenauflösung des Chips verwendet. Damit ist sichergestellt, dass das Bild in jedem Fall scharf ist, da die Zerstreuungskreise in jedem Fall kleiner sind, als die einzelnen Pixel des Bildsensors. In diesem Beispiel wird die maximale Auflösung des Sensorchips der eingesetzten Platinenkamera von 288 Pixel verwendet. Nach dem oben Gesagten ist ein Bild, wenn es bei hoher Auflösung scharf ist, auch bei kleineren Auflösungen scharf.

$$Z = \frac{d}{Auflösung der Sensordiagonalen} \tag{4.8}$$
$$Z = \frac{3{,}98mm}{288} \tag{4.9}$$
$$Z = 0{,}014mm \tag{4.10}$$
$$\tag{4.11}$$

Im dritten Schritt wird die hyperfokale Entfernung d_h berechnet. Sie wird für später für die Berechnung des Nahpunktes benötigt. Für die Berechnung dieser Größe werden die Brennweite und die Blende der in Kapitel 4.2 verwendeten Platinenkamera genommen. Ihre Brennweite beträgt $f = 3{,}8mm$, ihre Blende $K = 2{,}0$.

$$d_h = \frac{f^2}{K \cdot Z} \tag{4.12}$$

$$d_h = \frac{(3{,}8mm)^2}{2{,}0 \cdot 0{,}014mm} \tag{4.13}$$

$$d_h = 515{,}71mm \tag{4.14}$$

$$\tag{4.15}$$

Im fünften Schritt muss die Fokusentfernung g bestimmt werden. Hier können die Werte des bereits in Kapitel 5.3 gefertigten Spiegel aus der Simulation verwendet werden. Dieser hat eine Höhe von $h_s = 4{,}78mm$ und ist in einer Höhe von $h = 18{,}56mm$ aufgehängt. Da am Ende der Rechnung ein Schärfebereich mit seinen Extrema herauskommen soll, sollte hier die Hälfte der Spiegelhöhe von der Aufhängungshöhe abgezogen werden. Damit befindet sich der scharfe Bereich auf jeden Fall in der Mitte des Spiegels.

$$g = h - \frac{h_s}{2} \tag{4.16}$$

$$g = 18{,}56mm - \frac{4{,}78mm}{2} \tag{4.17}$$

$$g = 16{,}17mm \tag{4.18}$$

$$\tag{4.19}$$

Im sechsten Schritt folgt die Berechnung des Nahpunktes d_n. Das ist Punkt, welcher am Dichtesten vor der Linse liegt und scharf dargestellt wird.

$$d_n = \frac{g \cdot d_h}{d_h + (g - f)} \tag{4.20}$$

$$d_n = \frac{16{,}17mm \cdot 515{,}71mm}{515{,}71mm + (16{,}17mm - 3{,}8mm} \tag{4.21}$$

$$d_n = 15{,}79mm \tag{4.22}$$

$$\tag{4.23}$$

Im siebten Schritt findet die Berechnung des Fernpunktes d_f statt. Das ist der Punkt, der am weitesten von der Linse entfernt liegt und noch scharf dargestellt wird.

$$d_f = \begin{cases} \frac{g \cdot d_h}{d_h - (g-f)} & \text{, wenn } d_h > (g-f) \\ \infty & \text{, wenn } d_h \leq (g-f) \end{cases} \tag{4.24}$$

$$x = (g-f) \tag{4.25}$$

$$x = (16{,}17\,mm - 3{,}8\,mm) \tag{4.26}$$

$$x = 12{,}37\,mm \tag{4.27}$$

$$\tag{4.28}$$

$$135{,}71\,mm > 12{,}37\,mm \tag{4.29}$$

$$\tag{4.30}$$

$$d_f = \frac{g \cdot d_h}{d_h - (g-f)}$$

$$d_f = \frac{16{,}17\,mm \cdot 515{,}71\,mm}{515{,}71\,mm - (16{,}17 - 3{,}8\,mm)} \tag{4.31}$$

$$d_f = 16{,}57\,mm \tag{4.32}$$

$$\tag{4.33}$$

Im achten und letzten Schritt kann schließlich der Schärfentiefebereich Δd bestimmt werden.

$$\Delta d = d_f - d_n = \begin{cases} 2 \cdot \frac{g \cdot (g-f) \cdot d_h}{d_h^2 - (g-f)^2} & \text{, wenn } d_h > (g-f) \\ \infty & \text{, wenn } d_h \leq (g-f) \end{cases} \tag{4.34}$$

$$\Delta d = 16{,}57\,mm - 15{,}79\,mm \tag{4.35}$$

$$\Delta d = 0{,}78\,mm \tag{4.36}$$

$$\tag{4.37}$$

Der Schärfebereich beträgt bei dieser verwendeten Kamera mit ihrem verwendeten Objektiv $0{,}78$mm.

Bei gegebener Spiegelhöhe kann die Formel 4.34, so umgestellt werden, dass die benötigte Höhe berechnet werden kann. Dafür muss die Gleichung, wie in 4.38 nach g aufgelöst werden[2]. Beim Einsetzen in 4.39 wurde der Term unter dem Wurzelzeichen zusammengefasst, damit die Gleichung auf die Seite passt.

[2] Für diese Umrechnung wurde ein Taschenrechner genutzt, es gibt zwei Lösungen, wovon die für diesen Zweck richtige Lösung gewählt wurde.

$$\Delta d = 2 \cdot \frac{g \cdot (g - f) \cdot d_h}{d_h^2 - (g - f)^2} \Rightarrow \tag{4.38}$$

$$g = \frac{d_h \cdot \sqrt{\Delta d^2 + 2 \cdot d_h \cdot \Delta d + f^2} + f \cdot (\Delta d + d_h)}{\Delta d + 2 \cdot d_h} \tag{4.39}$$

$$g = \frac{515{,}71mm \cdot \sqrt{1334{,}676mm^2} + 3{,}8mm \cdot (4{,}78mm + 515{,}71mm)}{4{,}78mm + 2 \cdot 515{,}71mm} \tag{4.40}$$

$$g = 20{,}09mm \tag{4.41}$$

Das macht bei dem gefertigten Spiegel eine minimale Höhe von 20,09mm erforderlich, damit der gesamte Bereich des Spiegels von der Kamera scharf dargestellt werden kann. Das verändert allerdings die Sicht. Wie in Kapitel 4.2.1 gezeigt, verändert sich der Bildbereich mit der Entfernung zur Linse. Der gefertigte Spiegel zwingt daher zu Kompromissen.

Nachdem die Eigenschaften der eingesetzten Kamera bekannt sind, kann zum Testszenario übergegangen werden.

4.3 Testszenario

Beim Testszenario geht es darum eine Szene in der Realität und der Simulation zu schaffen, welche eine möglichst gute Aussage darüber zulässt, ob die Simulation funktioniert. Hierzu ist es notwendig Überlegungen zu Landmarken anzustellen. Diese müssen möglichst genau durch die Simulation wiedergegeben werden. Die genauen Objekte für das spätere Zielszenario sind an dieser Stelle noch nebensächlich, können aber bereits integriert werden. Farben spielen für diese Simulation ebenfalls eine untergeordnete Rolle. Sie müssen nicht der Realität entsprechen, sondern haben die Aufgabe Landmarken zu markieren. Die folgenden Unterkapitel beschäftigen sich mit den Objekten innerhalb des Testszenarios.

4.3.1 Aufbau der Versuchsanordnung

Zur Überprüfung der Simulation ist eine Kamera mit improvisiertem Spiegel zum Einsatz gekommen. Eine solche Kamera konnte mit relativ geringem Aufwand gebaut werden. Zum Einsatz ist die CMU-CAM2 aus Kapitel 4.2 gekommen. Diese Platinenkamera sollte ebenfalls in der fertigen Kamera genutzt werden. Die CMU-CAM2 blickte auf eine Gartenkugel mit einem Durchmesser von 150mm, welche in einer Halterung fixiert war. Im Folgenden wird dieser Aufbau als Kamera bezeichnet.

Abbildung 4.4: Eingesetztes Fußballfeld

Als Eingrenzung wird das Spielfeld für Robotfußball an der HAW-Hamburg gewählt, welches Abbildung 4.4 zeigt. Dieses ist nach den Normen der „RoboCup Junior Soccer League“[3] gebaut worden. Besonders hilfreich für die Überprüfung der Simulation, ist die Form des Spielfeldes. Es handelt sich um einen abgegrenzten Bereich von 1830x1220mm. Die Wände sind 145mm hoch. In die Ecken sind schräge Bauteile eingearbeitet, damit der Ball dort nicht hängen bleibt. In den kurzen Seiten befinden sich Tore. Sie ragen 80mm nach hinten und sind 450mm breit. Diese vielen Ausbuchtungen, Schrägen und Flächen bieten gute Landmarken, welche zur Kontrolle der Simulation verwendet werden können. Zusatzlich ist es gut möglich, dass die Kamera in dieser Umgebung später einmal eingesetzt werden soll. Dieses Spielfeld wird im Rahmen von Wettbewerben zwischen mehreren Hochschulen verwendet, an denen das Robotlabor der HAW-Hamburg teilnimmt. Dort treten Roboter gegeneinander an, die in Projekten konstruiert wurden. Damit ist das Spielfeld mit großer Wahrscheinlichkeit noch für einen längeren Zeitraum verfügbar. Auch als Einsatzort für spätere Kameras ist es denkbar, womit es Sinn macht eine Nachbildung des Spielfeldes in der Simulation vorzuhalten.

Zusammen mit einer halbtransparenten, orangen Tasse, konnten diese Bezugspunkte mit der Simulation verglichen werden. Die Tasse hat einen Durchmesser von ca. 77 Millimetern und eine Höhe von ca. 100 Millimetern. Um die Tasse in einem definierten Abstand zur Kamera aufzustellen, wurde der Messaufbau mit einem Maßband und einem Anschlag erweitert. In diesem Aufbau wurde die Tasse verfolgt, während sie mit dem Anschlag von der Kamera entfernt wurde. Dabei wurde ermittelt, in welcher Entfernung die Tasse noch gut erkannt werden konnte. Das ergab den maximalen Sichtradius. Anschließend wurde von der maximalen Position ein Bild gespeichert. Auf diesem Bild erkennt man die Anzahl der Pixel, welche die Tasse auf dem Bildsensor eingenommen hatte.

Für den ersten Versuch wurde die Kamera mit einem Gestell aus Fischertechnik gebaut. Im zweiten Schritt wurde dieses Gestell durch eine deutlich stabilere Konstruktion ersetzt.

Die Halterung aus Fischertechnik bot nicht die benötigte Stabilität. Trotz zusätzlicher Verstrebungen konnten Verwindungen nicht ausreichend verhindert werden. Aus diesem Grund ist

[3] Nähere Informationen zur RoboCup Junior Soccer League auf der Homepage der RoboCup Federation RoboCupJunior

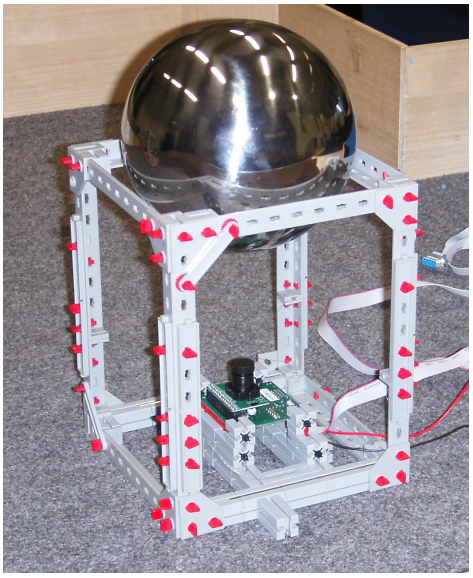

Abbildung 4.5: Omnidirektionale Kamera im Gestell aus Fischertechnik mit Gartenkugel

ein zweites Gestell konstruiert worden. Dieses deutlich stabilere Gestell besteht aus 4 Gewindestangen, einer Bodenplatte mit Kamerahalterung und einer oberen Platte mit kreisförmigen Loch, in das die Kugel gelegt werden kann. Das Loch ist mithilfe einer Drehmaschine entstanden, um eine besonders runde Form zu ermöglichen. Der Auflagepunkt der Kugel wurde in das Material der oberen Halterung gefräst. Damit liegt die Kugel möglichst nah an der unteren Kante der Halterung auf. So sollte die Reflexion der Halterung in der sichtbaren Spiegelfläche minimiert werden.

Bei der Konstruktion dieser Halterung ist aufgefallen, dass die Kugel bei genauer Vermessung nicht ganz rund ist. Dieses ist bei der Analyse der Simulationsqualität zu berücksichtigen.

Die Gewindestangen ermöglichen eine sehr präzise Einstellbarkeit der Kamera. Die obere Halterung für die Kugel ist durch die Gewindestangen höhenverstellbar. Sie wird durch Muttern oberhalb und unterhalb der Halterung fixiert. Die Probleme mit der Flexibilität innerhalb der Konstruktion aus Fischertechnik konnten mit dieser Konstruktion behoben werden.

Die CMU-CAM2 ist in einer Halterung eingeklemmt und kann in der Ebene verschoben werden. Dadurch ist es möglich die Kamera auf die Mitte der Kugel auszurichten. Als weitere Hilfe wurde an jede der vier Gewindestangen ein Gummiband befestigt. In diese Halterungen sind kreuzförmig dünne Drähte gespannt. Das ermöglicht eine einfachere Ausrichtung der CMU-CAM2 auf die Mitte der Kugel.

Die Kamera ist an eine definierte Position auf dem Spielfeld gesetzt worden. Dabei lag die

Abbildung 4.6: Stabiles Gestell aus Gewindestangen und MDF-Platten

hintere Kante der Kamera genau an der Wand des Spielfeldes. Die linke Seite der Kamera schloss bündig mit der Kante des Tores ab.

Zur Bestimmung der Entfernung der halb transparenten Tasse verlief ein Maßband parallel zur Wand des Spielfeldes. Der Nullpunkt war die vordere Kante der Kamera. Das genaue Bestimmen der Entfernung ermöglichte ein Anschlag aus Legosteinen. Dieser bestand aus einer großen quadratischen Lego Grundplatte, auf der mit einigen Teilen ein zweiseitiger Anschlag für die Tasse gebaut wurde. Die Grundplatte konnte an der Wand entlang über das Maßband geschoben werden. Die Entfernung wurde an der vorderen Kante der Lego Grundplatte abgelesen. Damit entsprechen die gemessenen Werte die hinterste Kante der Tasse.

Nachdem das Szenario in der Realität aufgebaut ist, müssen die Objekte in der Simulation nachgebildet werden.

4.3.2 Berechnungsgrundlage für das Gestell und die Spiegel in der Simulation

In diesem Abschnitt wird der Aufbau der verschiedenen Objekte beschrieben, die in der Simulation verwendet wurden. Dazu gehören Abstrahierungen der Szene aus Kapitel 4.3.1 auf geometrische Objekte und Berechnungen von Abständen und Höhen bei der Kamera.

Abbildung 4.7: Versuchsaufbau zur Überprüfung der Simulation

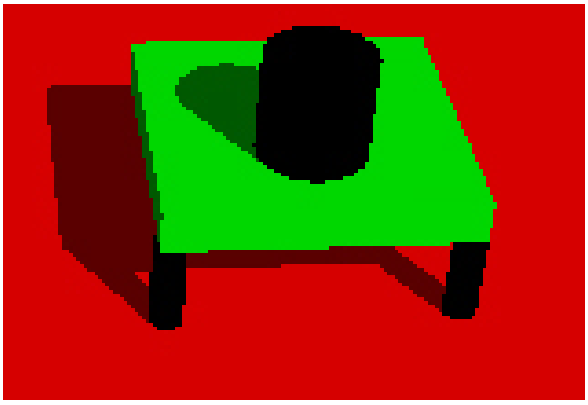

Abbildung 4.8: Abbildung der Kamera in der Simulation

CMU-CAM2

Die CMU-CAM2 ist auf einen Quader reduziert worden. Das Objektiv wird durch einen Zylinder abgebildet. Der Augpunkt liegt genau auf der Kante dieses Zylinders und ist nach oben auf den Spiegel gerichtet. Die Nachbildung der Kamera ist wichtig, da diese bei vielen Spiegelformen im Bild erscheint. Sie erfüllt zwei wichtige Funktionen. Zum Einen dient sie als Landmarke, die von der Simulation korrekt wiedergegeben werden muss. Zum Anderen gibt die Kamera einen wichtigen Hinweis darauf, wie viel Spiegelfläche für die Objekterkennung verloren geht. Abbildung 4.8 zeigt die Kamera in der Simulation.

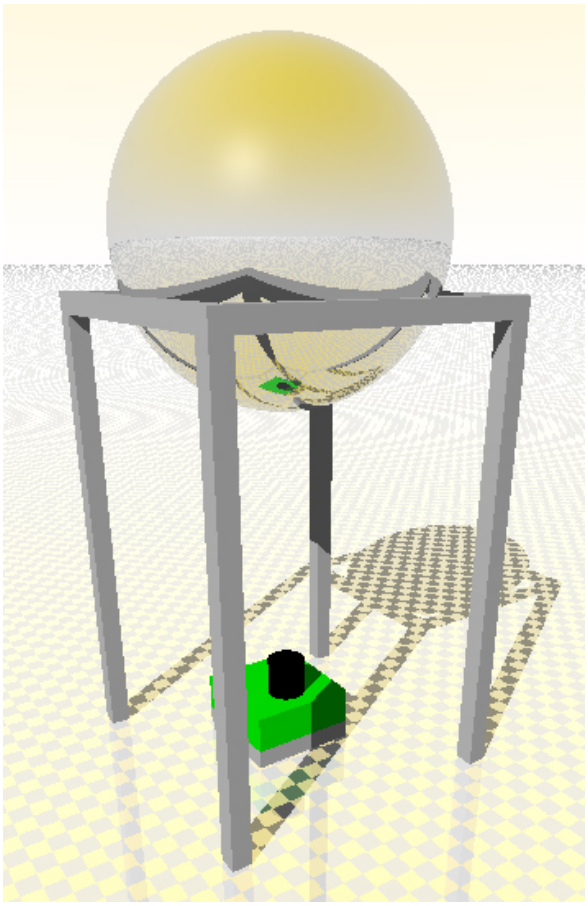

Abbildung 4.9: Abbildung des Gestelles aus Fischertechnik in der Simulation

Gestell aus Fischertechnik

Das Gestell aus Fischertechnik ist einfach nachzubilden. In der Simulation wird die Form auf acht Quader reduziert. Jeder Quader hat eine quadratische Grundfläche mit Seitenlänge 10mm. Das Innenmaß zwischen den Stützen beträgt 135mm. Dieser Wert legt auch die Tiefe fest, in der die Kugel in das Gestell eintaucht. Das ergibt die Größe des sichtbaren Spiegels. Die Maße werden hierbei durch die Möglichkeiten der Teile aus dem Fischertechnik Baukasten im Robotlabor vorgegeben. Ein Bild dieses Gestelles in der Simulation ist in Abbildung 4.9 gezeigt.

Für die Ermittlung der Position und Größe der Kugel in der Simulation ist die Kugelform auf eine Kreisform reduziert worden. Der Kreis hat den gleichen Durchmesser wie die Kugel, seine Sehne ist durch den Abstand der Stützen auf 135mm festgelegt. Hier liegt der Kreis auf den Stützen. Aus diesen Informationen lässt sich die Eintauchtiefe h der Kugel in die

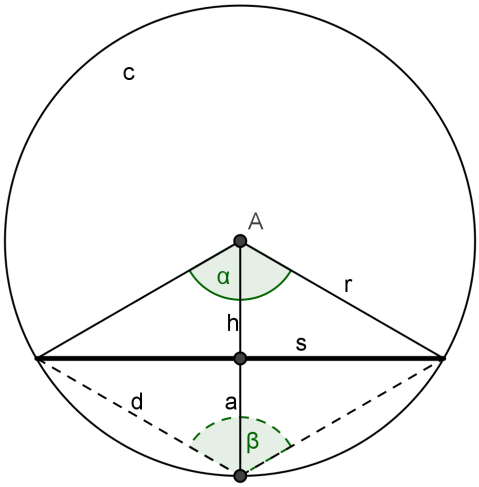

Abbildung 4.10: Größen für Kreissehnenberechnung

Halterung berechnen. Abbildung 4.10 zeigt die für die einfache Formel notwendigen Zusammenhänge.

$$h = \sqrt{r^2 - \frac{s^2}{4}}$$

Die Höhe der Kugel über der Kamera hängt mit dem Radius der Kugel und dem Öffnungswinkel der Kamera zusammen. Die Kugel befindet sich genau dann in der richtigen Höhe, wenn sich die Außenseiten der Kugel genau am Rand des Sichtkegels befinden. Diesen Wert kann die Simulationssoftware eigenständig berechnen. Wie die Berechnung aussieht, kann in Kapitel 4 nachgelesen werden.

Verbessertes Gestell

Das verbesserte Gestell besteht aus zwei Quadern mit einer Seitenlänge von 200mm und vier Zylindern mit einem Durchmesser von 12mm und einer Länge von 300mm. Die Quader bilden die Boden- und Deckenplatte ab, die Zylinder die Gewindestangen des Gestells. Das kreisrunde Loch in der oberen Platte muss nicht hinzugefügt werden. Die Kugel darf die Platte bei POV-Ray überlappen und kann einfach an die richtige Stelle definiert werden. Das Ergebnis ist für die Bewertung des Spiegels ausreichend genau. Dieser Aufbau wird in Abbildung 4.11 gezeigt.

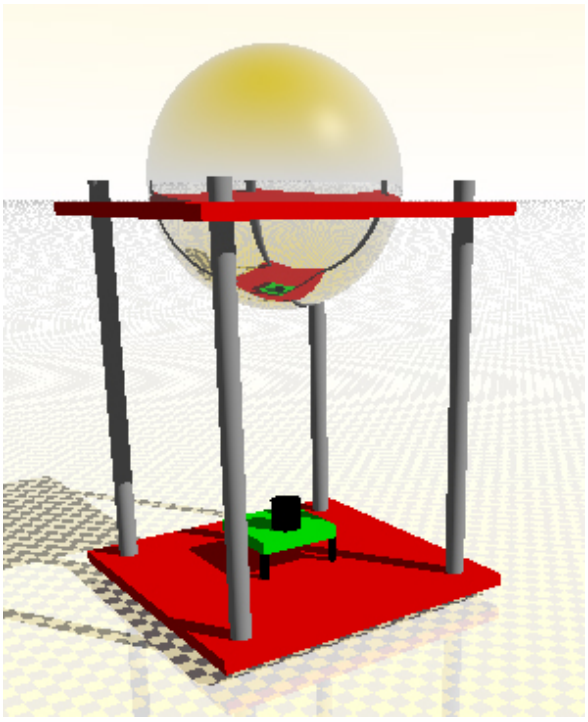

Abbildung 4.11: Abbildung des verbesserten Gestelles in der Simulation

4.3.3 Das Spielfeld

Abbildung 4.12 zeigt die fertige Draufsicht auf die mit der Simulation erstellten Szene. Zu sehen sind das oben beschriebene Feld sowie eine Abbildung einer Tasse und einer Kugel, die sich an definierten Positionen der Szene befinden. Beide Objekte wurden genau vor oder hinter der Kamera platziert. In der Realität müssen zwei Bilder gemacht werden, dort muss auch die Kugel auf dem Spielfeld liegen. In der Simulation soll die Kugel nur eine Vorabidee davon geben, wie ein Ball am hinteren Ende der Szene zu sehen wäre. Die Kamera befindet sich an der hintersten Wand des Feldes an der linken Kante des Tores. Diese Position ist in der Realität sehr leicht zu reproduzieren. Das Zentrum der Szene befindet sich im Zentrum der Linse der Kamera. Der Nullpunkt für die Entfernungsmessung ist die vordere Kante der Kamerahalterung. Damit lassen sich Positionen von Objekten sehr leicht in einen Bezug zur Wirklichkeit bringen. Die verwendete Einheit in der POV-Ray Szenendatei ist bei dieser Simulation Zentimeter.

Der orange Zylinder entspricht in seiner Größe einer Tasse, welche im Labor verfügbar war und für erste Versuche zum Einsatz kommt. Es handelt sich hierbei lediglich um eine einfache Figur, die leicht durch eine Kugel oder ein anderes Objekt ausgetauscht werden kann.

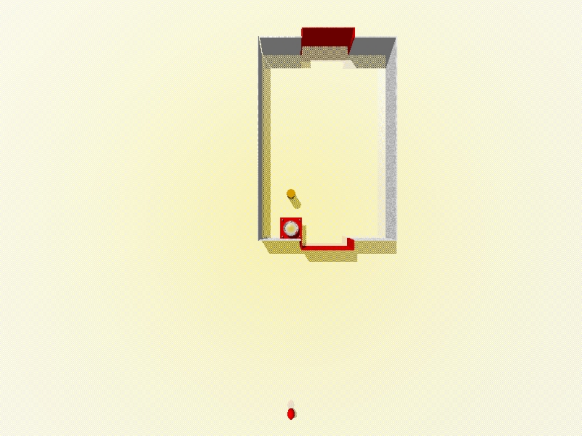

Abbildung 4.12: Szene zum Testen von Kamerabildern

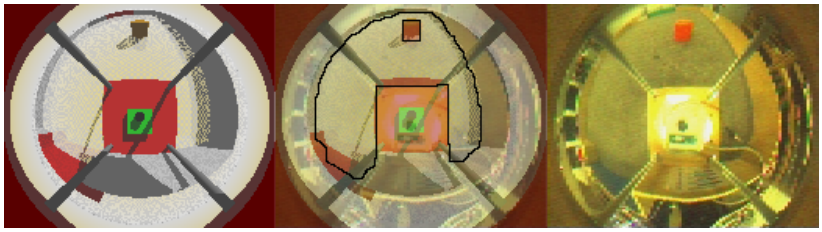

Abbildung 4.13: Vergleich Simulation und Realität beim kugelförmigen Spiegel

4.3.4 Ergebnis der Überprüfung

Wie im Vergleich zwischen der Simulation und dem Bild der Kamera in Abbildung 4.13 deutlich zu sehen ist, ist das simulierte Ergebnis sehr nah an der Realität. Die Abweichungen sind mit der nicht ganz perfekten Kugelform des Spiegels und leichten Ungenauigkeiten beim Justieren der Kamera zu erklären. Das Bild ist auf der gesamten Fläche scharf und das zu verfolgende Objekt klar zu erkennen. Die die Größenordnungen zwischen den Proportionen der Simulation und des Kamerabildes stimmen zufriedenstellend gut überein.

Damit kann das Ergebnis der Simulation als brauchbar angesehen werden und als Grundlage für einen weiteren selbst gefertigten Spiegel dienen.

4.4 Simulationsszenario für Spiegel

Nachdem die Überprüfung der Simulation in Kapitel 4.3.4 gezeigt hat, dass die Simulation ausreichend genaue Bilder liefert, kann dazu übergegangen werden, verschiedene Formen für den späteren Spiegel zu betrachten.

Hinzu kommen weitere Parameter aus dem Kapitel 4.1 auf die bei der Simulation eingegangen werden muss. Die Objekte aus der vorherigen Simulation in Kapitel 4.3 können übernommen werden. Sie entsprechen den Objekten, die später verwendet werden sollen. Weitere Vorgaben entstehen aus dem Szenario der Beispielanwendung. Diese wird im nächsten Unterkapitel vorgestellt. Anschließend werden einfache Spiegelformen vorgestellt, die mit der Simulation erzeugt werden können.

4.4.1 Szenario der Beispielanwendung

Die Beispielanwendung soll die Qualität und die Funktionsfähigkeit des hier beschriebenen Verfahrens zeigen. Zu diesem Zweck wird ein Spielfeld genommen, welches nicht größer ist, als ein im Robotlabor eingesetztes Spielfeld für Robotfußball. Die Beispielanwendung enthält einen Roboter, der in der Lage ist, Richtungsinformationen zu einem Ball zu erhalten und darauf zu reagieren. Als Sensor wird für diesen Zweck eine, im Rahmen dieser Arbeit geschaffene, omnidirektionale Kamera verwendet. Die Kamera wird auf eine Roboterplattform montiert, welche einem Ball folgen soll. Das Szenario besteht aus einem bunten Ball, welcher auf einer einfarbigen Fläche verfolgt werden soll.

Ein entscheidender Punkt hierbei ist die Auswahl eines geeigneten Balles. Der Ball muss eine Farbe haben, welche leicht von der Kamera erkannt werden kann. Diese Farbe sollte nach Möglichkeit nicht in der sonstigen Umgebung auftauchen. Einen Ball ausschließlich über Farberkennung zu identifizieren und zu verfolgen ist nicht trivial. Das Problem besteht darin, dass der Ball von oben beleuchtet wird. Durch seine Rundung hat der Ball für die Kamera keine einheitliche Farbe, sondern umfasst durch Licht und Schatten ein ganzes Spektrum unterschiedlicher Farben. Dieses erstreckt sich durch verschiedene Töne, die bei weiß beginnen und sich über die Hauptfarbe nach dunkelbraun oder Schwarz erstreckt.

Auch bei anderen Sensortypen gibt es zuweilen Schwierigkeiten, da unterschiedliche Lichtquellen unterschiedliches Licht erzeugen. So kommt es öfter vor, dass Roboter Aufgaben im Labor problemlos bewältigen, allerdings am Einsatzort Schwierigkeiten bekommen.

Um diese Probleme zu umgehen, wird ein leuchtender Ball eingesetzt. Dieser bietet den Vorteil, dass er über die komplette Fläche eine einheitliche Farbe bietet. Zusätzlich macht ein leuchtender Ball das System weniger anfällig gegenüber Lichtschwankungen, da er aktiv farbiges Licht ausstrahlt.

Erschwerende Bedingungen gibt es in diesem Szenario lediglich durch die geringe Größe des Spiegels. Dieser soll möglichst klein ausfallen. Der Einsatzort beschränkt sich auf den Bereich in Gebäuden. Durch den Einsatz eines Spezialballes ist die Lichtempfindlichkeit reduziert worden, was eine Vorführung an unterschiedlichen Orten ohne Laborlicht vereinfacht.

Als Spielfeld dient ein grüner Teppich mit aufgedrucktem Fußballfeld. Dieses Spielfeld hat zwar Markierungen, diese stören die Ballerkennung jedoch nicht. Zusätzlich bieten sie Material für weitere Versuche mit Linienerkennung und geben dem Szenario eine nette Gestalt.

4.4.2 Auswahl Spiegelformen

Nachdem das Szenario der Beispielanwendung feststeht, sollte eine Betrachtung möglicher Spiegelformen stattfinden. Die Simulationssoftware bringt vier verschiedene Spiegelformen mit. Diese bestehen aus zwei Grundformen, einer Kugel und einem Kegel. Diese Formen sind leicht zu fertigen und bieten sehr unterschiedliche Eigenschaften, die im Folgenden gezeigt werden sollen.

Zur besseren Veranschaulichung wurde für diesen Zweck ein kleines Extraszenario geschaffen, welches in Abbildung 4.14 gezeigt wird. Es handelt sich um die Simulation des Robotfußballfeldes aus dem Robotlabor, welches in Kapitel 4.3.1 bereits vorgestellt wurde. Hinzu kommen einige Zylinder, welche die Größe der dort beschriebenen Tasse haben. Die Kamerahalterung wurde in der Simulation um eine zweite Ebene erweitert. Damit ist es leichter zu erkennen, dass sich die Zylinder in einer Linie mit der Kamera befinden. Die Platinenkamera befindet sich in der Höhe, in der sie auf dem späteren Roboter verwendet wird. Die obere Spiegelhalterung ist für diese spezielle Versuchsanordnung entfernt worden, das ist für die Verdeutlichung der gleich gezeigten optischen Effekte hilfreich. Die braunen Zylinder stehen im jeweils gleichen Abstand zur Kamerahalterung. Alle Zylinder sind zehn Zentimeter hoch. Die in der Luft schwebenden Zylinder befinden sich an den gleichen x- und y-Koordinaten wie der darunter liegende braune Zylinder. Der untere rote Zylinder der schwebenden Gruppierung befindet sich in der Höhe der roten Plattform, auf der die Platinenkamera steht. Der blaue, damit etwa in Höhe des Spiegels und der grüne auf jeden Fall oberhalb des Spiegels.

Kugelform

Für Spiegel in Kugelform werden der Radius der Kugel und der Durchmesser der Halterungsöffnung benötigt, in der die Kugel liegt.

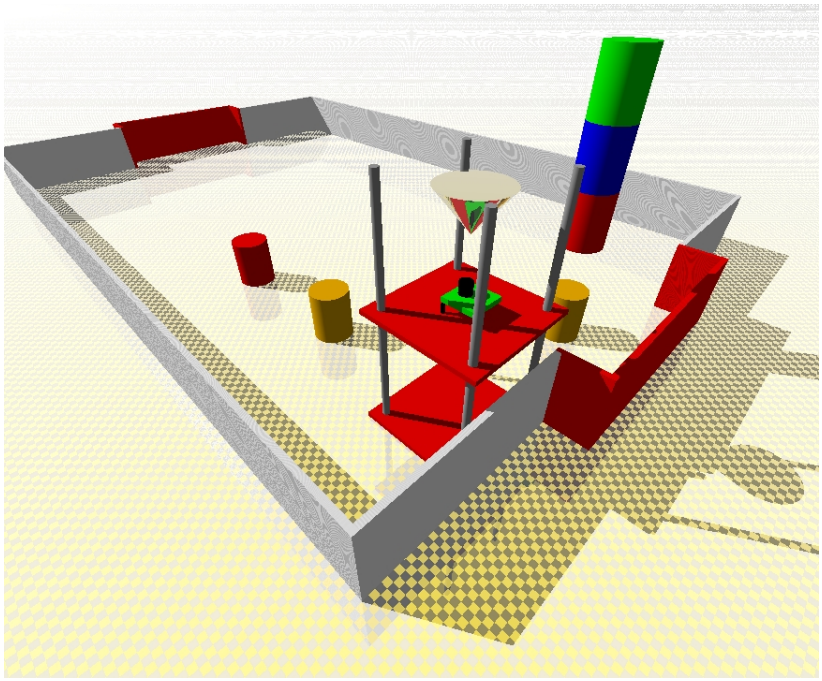

Abbildung 4.14: Testszenario für die Erläuterung der Spiegelformen

Für die Betrachtung der Kamerasicht auf kugelförmige Spiegel gibt es hier zwei Beispiele. In beiden Bildern wurde die gleiche Kugelgröße verwendet. Verändert wurde die Größe des Auflagedurchmessers auf der Kugelhalterung. Damit verändert sich der sichtbare Bereich auf die Kugel. Anders als in Abbildung 4.14 gezeigt, wurde hierfür die obere Spiegelhalterung im Bild verwendet. Die Spiegelhalterung schneidet den Spiegel an der richtigen Stelle ab und verdeckt höher liegende Bereiche.

Abbildung 4.15 zeigt eine Gegenüberstellung zweier Kamerabilder. Beim linken Bild handelt es sich um eine Kamera mit kugelförmigem Spiegel, der einen Durchmesser von 150mm hat. Das Loch der Auflagefläche hat hier einen Durchmesser von 145mm. Beim rechten Bild handelt es sich ebenfalls um einen kugelförmigen Spiegel mit einem Durchmesser von 150mm. Das Loch der Auflagefläche beträgt dieses Mal jedoch lediglich 7,5cm.

Die Simulation passt die Höhe des Spiegels automatisch so an, dass der Spiegel die größtmögliche Darstellung auf der Bildsensorfläche der Platinenkamera hat. Der geringere Durchmesser der sichtbaren Spiegelfläche sorgt also dafür, dass der zweite Spiegel deutlich näher zur Platinenkamera steht. Abbildung 4.16 zeigt diesen Zusammenhang.

Betrachtet man nur eines der Bilder aus Abbildung 4.15, sind zwei grundlegende Eigenschaften dieser Spiegelform zu erkennen. Die auffälligste Eigenschaft ist die, dass bauchige und kugelförmige Formen eine Reflexionsfläche zur Kamera haben. Das bedeutet, die Ka-

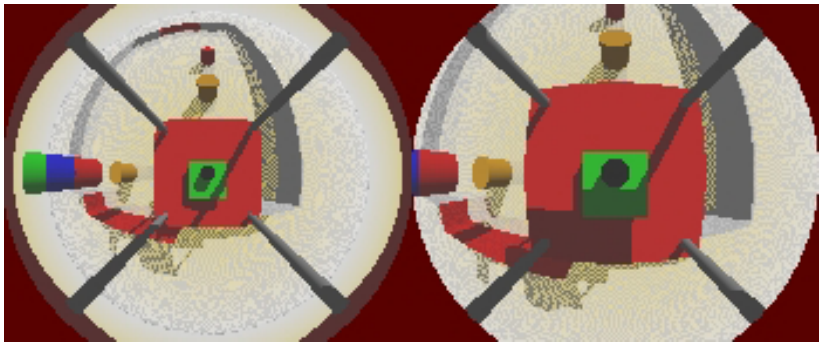

Abbildung 4.15: Simulation zweier kugelförmiger Spiegel mit 15cm Durchmesser mit verschiedenen Auflageflächen

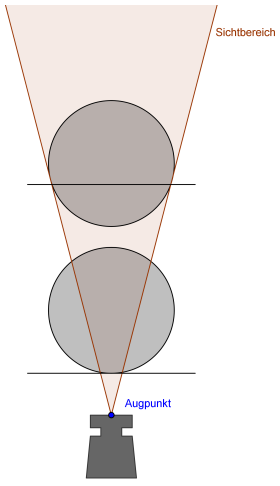

Abbildung 4.16: Zusammenhang Höhe und Größe des Spiegels im Bild

mera sieht sich selber, wodurch wertvolle Bildfläche verloren geht. Die zweite Eigenschaft ist die, dass Objekte durch die Wölbung auseinandergezogen und leicht verzerrt werden. Im Großen und Ganzen behalten sie jedoch ihre Grundform.

Vergleicht man beide Bilder, werden drei Effekte sichtbar. Am stärksten fällt der deutlich vergrößerte Flächenverbrauch durch die Kamera selber auf. Das kommt dadurch, dass der Spiegel wie oben beschrieben deutlich dichter zur Kamera steht. Auch sehr deutlich zu sehen ist die geringere Sichtweite im zweiten Bild. Das kommt daher, dass die Kugel weiter unten abgeschnitten wurde und die Seiten mit der geringeren Höhe eine geringere Steigung haben. Bei etwas genauerer Betrachtung fallen die seitlich schwebenden Kegel auf. Beim linken Bild sind alle drei Kegel vollständig zu erkennen. Zusätzlich sieht man den Horizont als helle Fläche über dem etwas pixeligen Boden. Die linke Kamera war also in der Lage nicht nur in der Ebene und nach unten zu sehen, sondern hat zusätzlich Bereiche gesehen, die über der Spiegelhöhe lagen. Bei der rechten Kamera mit der geringeren Kugelwölbung ist nur ansatzweise der blaue Kegel zu sehen. Sowohl der grüne Kegel als auch der Horizont bleiben unsichtbar. Die Sichtweite zu den Seiten entspricht lediglich der Spielfeldbreite. Die Kamera sieht nur Bereiche in einem bestimmten Radius unterhalb der Spiegelhöhe.

Ganz anders sieht das Bild einer omnidirektionalen Kamera mit kegelförmigem Spiegel aus.

Kegelform

Für die Kegelform wird bei der Simulation der Radius des Kegels sowie seine Höhe benötigt. Der Radius sowie der Winkel des Kegels sind vom Anwender vorgegeben. Die Berechnung der Höhe des Kegels über der Linse wird auch hier von der Simulationssoftware übernommen.

Abbildung 4.17 zeigt auf der linken Seite die Sicht einer omnidirektionalen Kamera mit kegelförmigen Spiegeln. Die rechte Seite zeigt eine Draufsicht auf die Spiegel und vermittelt einen Eindruck davon, wie der Spiegel beschaffen ist. Die Kegel haben alle einen Radius von 50mm. Die Winkel sind von oben nach unten betrachtet 20°, 37° und 45°. Dabei handelt es sich um Extrempositionen, welche die drei maßgeblichen Sichtformen repräsentieren.

Kegelförmige Spiegel haben zwei Eigenschaften gemeinsam. Das Auffälligste ist, dass die Kamera selber nicht im Sichtbereich ist. Zu sehen ist lediglich die Spiegelhalterung. Dafür verzerren kegelförmige Spiegel kleinere Objekte sehr stark und zwingen sie in eine dreieckige Form, deren Spitze auf das Zentrum des Bildes gerichtet ist.

Im direkten Vergleich der drei Bilder aus Kamerasicht sieht man drei verschiedene Variationen, für die man diese Art von Spiegel einsetzen kann. Der oberste Spiegel mit 20° zeigt den direkten Bereich um die Kamera herum. Die Sicht ist stark begrenzt. Dafür erscheinen

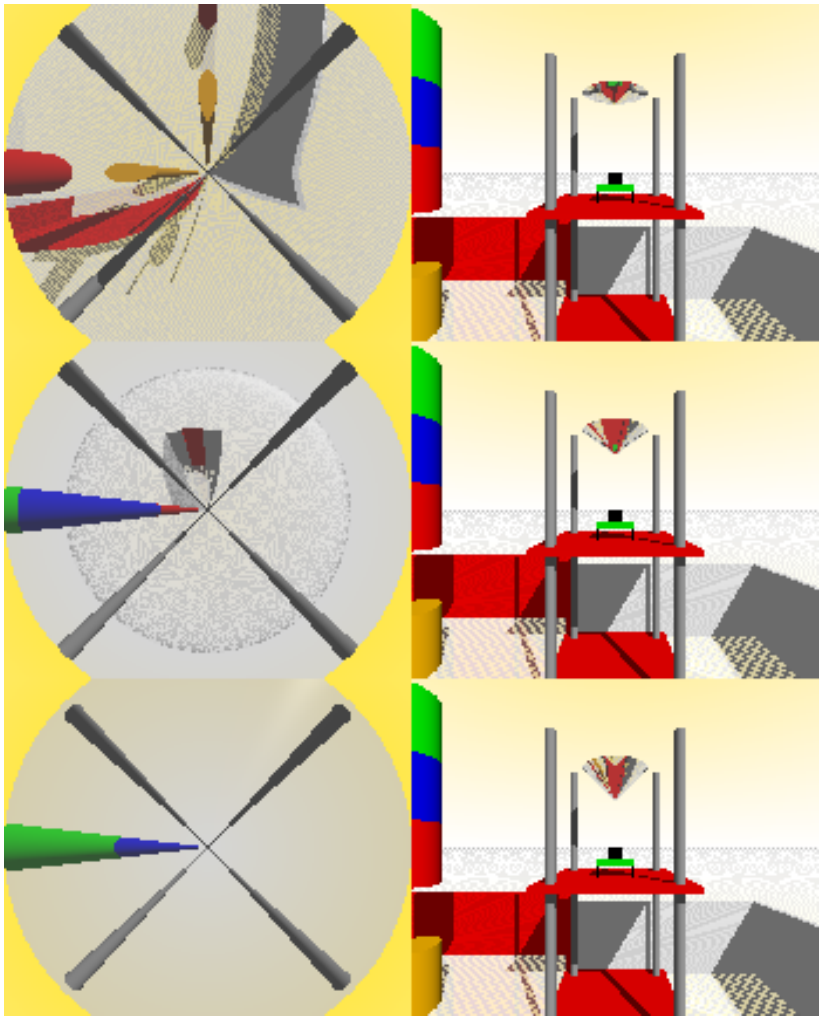

Abbildung 4.17: Zusammenhang Sichtweite, Größe und Winkel bei kegelförmigen Spiegeln

auch Objekte, die genau neben der Kamera stehen auf dem Kamerabild. Der mittlere Spiegel bietet eine sehr weite Sicht. Er ist in der Lage Objekte zu erkennen, die entweder sehr groß und dicht sind oder parallel zum Boden auf Spiegelhöhe liegen. Das erkennt man an der sehr großen Proportion des blauen Zylinders und dem breiten hellgrauen Streifen über dem Horizont. Der mittlere Spiegel ist nicht genau an den Öffnungswinkel der Kamera angepasst, weshalb er auch ein klein wenig nach oben sehen kann. Die unterste Kamera sieht fast ausschließlich nach oben. Der rote Zylinder ist nicht zu erkennen, genau so wenig der Boden der Szene. Dafür kann der Spiegel über die Enden der Spiegelhalterung sehen und enthält einen sehr großen Teil des grünen Zylinders.

Für die praktische Umsetzung sehr wichtig ist die rechte Seite der Abbildung. Hier zeigt sich, dass Spiegel mit kleinem Winkel sehr flach sind. Wird der Winkel jedoch sehr steil, vergrößert sich die Tiefe des Spiegels sehr stark. Hier muss auf Effekte der Tiefenunschärfe geachtet werden.

Die Kegelform fordert einen Kompromiss zwischen Weitsicht und der Sicht auf Objekte im nahen Umfeld. Dieses kann durch die Zweiteilung des Kegels verringert werden. Die Kombination von zwei kegelförmigen Spiegeln zu einem Doppelkegel verringert den toten Winkel.

Doppelkegel

Bei einem Spiegel in Doppelkegelform werden mehr Informationen benötigt. Zu den Werten des oberen Kegels in Form von generellem Radius und Winkel kommen noch ein Radius für den unteren Kegel, der natürlich kleiner sein muss, als der des oberen Kegels und ein zusätzlicher Winkel für den unteren Kegel.

Abbildung 4.18 zeigt zwei Spiegel der Doppelkegel Bauart. Der Obere der beiden Spiegel zeigt eine Variante, in der der obere Kegel eher flach und der untere Kegel eher steil ist. Der untere der beiden Spiegel zeigt eine Variante in welcher der obere Kegel eher steil und der untere Kegel eher flach ist.

Diese Kegelform bietet alle grundlegenden Eigenschaften kegelförmiger Spiegel aus Kapitel 4.4.2. Hinzu kommen die Eigenschaften für den oben aufgesetzten Spiegel. Bei dieser Spiegelform gibt es einen mehr oder weniger großen toten Bereich, welcher zwischen den Spiegeln liegt. Die Inhalte des unteren Spiegelteils sind immer deutlich kleiner als bei einem Spiegel in normaler Kegelform. Damit erfordert die Objekterkennung in dem Bereich eine höhere Auflösung. Der Obere Kegel verliert einen Teil seines Blickbereiches, da an dieser Stelle der kleinere Kegel sitzt.

Diese Spiegel können gut in Bereichen eingesetzt werden in denen Objekte gesucht werden, die sehr groß und weit entfernt sind. Hierbei kann der eine Spiegel die entfernten Objekte

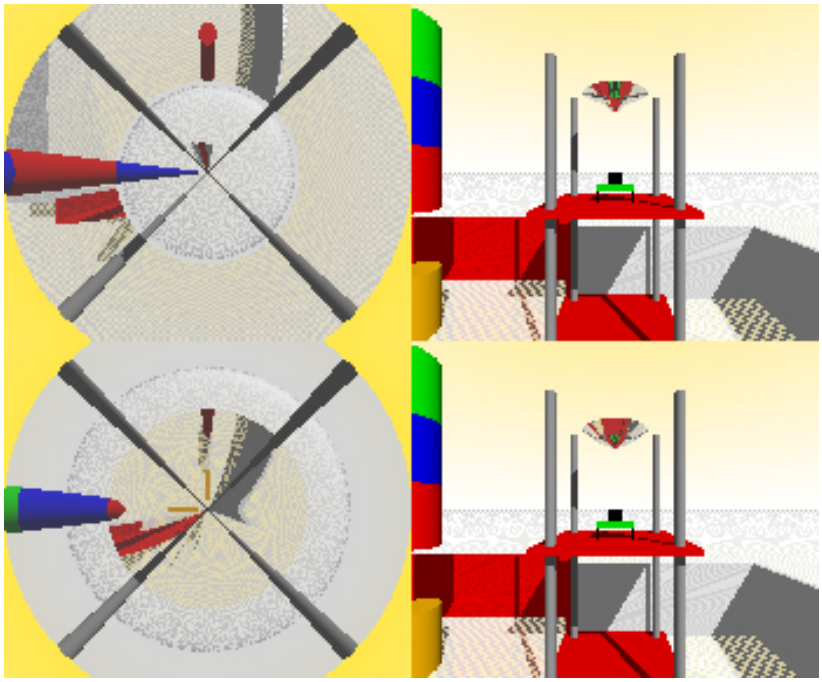

Abbildung 4.18: Spiegel in Doppelkegelform: obere Abbildung: kleiner Winkel unten großer Winkel; Untere Abbildung: oben großer Winkel, unten kleiner Winkel

sichtbar machen, der andere Spiegel ermöglicht das sichere Navigieren ohne gegen Hindernisse zu fahren. Ein sinnvoller Einsatz auf kleineren Flächen ist eher nicht gegeben, da der tote Bereich nur sehr schwer zu erfassen und zu schließen ist.

Eine weitere Kombinationsmöglichkeit der ist in Kapitel 4.4.2 beschriebene kugelförmige Spiegel mit dem in Kapitel 4.4.2 beschriebene Kegel.

Kugel mit Kegel

Für einen Spiegel in der Form Kugel mit Kegel müssen der Radius der Kugel, der Durchmesser der Öffnung in der Halterung, der Radius des Kegels und der Winkel des Kegels angegeben werden. Die Simulationssoftware kümmert sich auch hier um die Höhe des Spiegels und sorgt dafür, dass möglichst viel Bildfläche vom Spiegel ausgefüllt wird.

Die in Abbildung 4.19 gezeigte Kombination aus Kugel und Kegel hat einen bedeutenden Vorteil gegenüber dem in Kapitel 4.4.2 beschriebenen Doppelkegel. Da die Kamera an zentraler Stelle des Spiegels zu sehen ist und diese für die Objekterkennung nicht nötig ist, kann hier ein kegelförmiger Spiegel ergänzt werden, ohne Informationen gegenüber dem kugelförmigen Spiegel zu verlieren.

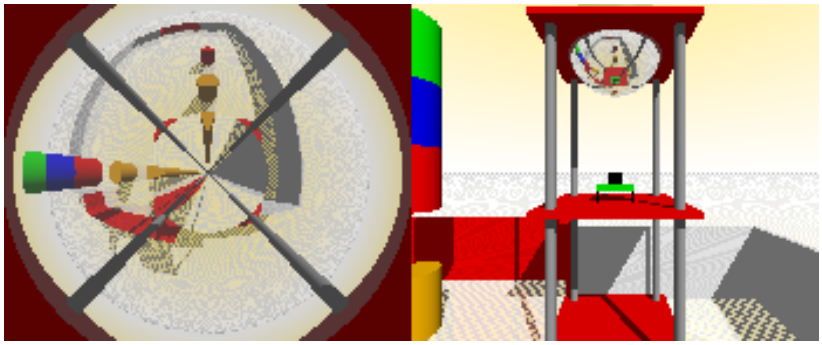

Abbildung 4.19: Kugel mit Kegel

Das führt dazu, dass diese Kombination alle Eigenschaften eines kugelförmigen Spiegels hat, wie sie in Kapitel 4.4.2 beschrieben werden. Der große Vorteil besteht darin, dass der verlorene Platz im Zentrum des Bildes bei dieser Spiegelform genutzt wird.

Die linke Seite von Abbildung 4.19 zeigt die Kamerasicht auf eine Kugel mit sehr kleinem Kegel. Dieser wurde an die Größe der Kamera angepasst. Da die Kamera auf einer größeren roten Basisplatte liegt, welche eine quadratische Form hat, sind auf der Kugelfläche noch rote Ecken erkennbar. Auffällig ist, dass die braunen Zylinder jeweils doppelt zu sehen sind. Damit erhöht sich die Anzahl der Pixel, die ein Objekt in dem Bereich im Kamerabild einnimmt. Das kann für eine sicherere Erkennung von Objekten in dem Bereich genutzt werden.

Die starke Verzerrung der Objekte auf dem Bild einer omnidirektionalen Kamera mit Spiegel in Kegelform ist für viele Anwendungen schädlich. Ein großer Vorteil ist jedoch die Eigenschaft des Kegels die Kamera zu verbergen. Das macht diese Kombination besonders hilfreich.

4.4.3 Entscheidung für einen Spiegel nach Lauf

Da bei dem fertigen Spiegel eine möglichst kleine Form angestrebt wird, kommt für den Spiegel nur eine Form in Frage, die unten Spitz zuläuft. Damit fallen die einfache Kugel oder eine paraboloide Form raus. Die Entscheidung fällt hier auf die einfache Kegelform. Dieses hat zwei Gründe. Ein einfacher kegelförmiger Spiegel ist am leichtesten zu fertigen und liefert nach Bewertung des Simulationsergebnisses ausreichende Sichteigenschaften. Wichtiger ist jedoch der Punkt, dass diese Simulationssoftware bisher nur an einem kugelförmigen Spiegel getestet wurde. Um Seiteneffekte beim Ergebnis des zweiten Spiegels nach Möglichkeit ausschließen zu können, macht es Sinn hier die Grundform zu überprüfen. Im Falle des Kegels sollte sie möglichst gleichmäßige, große Flächen liefern, um Fehler leichter erkennen zu können. Wird das Bild der Kamera mit dem kegelförmigen Spiegel von der Simulation

richtig wiedergegeben, so ist sichergestellt, dass die Simulation für diese Art von Spiegel generell geeignet ist. Eine Kombination der Spiegelformen wird also zum richtigen Ergebnis führen.

Die Bewertung dieses kegelförmigen Spiegels wurde in das nächste Kapitel vorgezogen.

4.4.4 Bewertung des Simulationsergebnisses und fertigen Spiegels

Bei der Bewertung des Spiegels und der Simulation gibt es zwei Ergebnisse. Wie im linken Bild von Abbildung 4.20 zu sehen ist, gibt die Kamera ein sehr unscharfes Bild zurück. Dieses entsteht durch die eingesetzte Optik der Kamera. Da der Spiegel sehr nah über der Linse steht, ist die Kamera nicht in der Lage ein scharfes Bild über die gesamte Tiefe des Spiegels zu erzeugen.

In Abbildung 4.20 wurden drei Aufnahmen des Spiegels nebeneinandergestellt. Vom ersten zum letzten Bild wurde jeweils die Höhe der Linse über dem Sensorchip um eine halbe Umdrehung verändert. Bei genauerer Betrachtung kann der scharfe Bereich besonders in der Darstellung der Gewindestangen erkannt werden. Um das Ergebnis aus Abbildung 4.20 nachzuvollziehen, wurde die Rechnung aus Kapitel 4.2.3 für die bei den Bildern verwendete Auflösung von 174 Mal 144 Pixeln durchgeführt. Wie in Kapitel 4.2.3 beschrieben, wurde für die Berechnung mit dem kleineren Wert 144 gearbeitet. Das Ergebnis ist eine berechnete Schärfentiefe Δd von 1,49mm.

Wird der berechnete Wert mit den Bildern aus Abbildung 4.20 verglichen, wird die Rechnung vom Bild bestätigt. Die Spiegeltiefe des Kegels aus Bild 4.20 beträgt 4,78 Millimetern. Der scharfe Bereich ist aufgrund der Verzerrungen nicht genau auszumachen. Er entspricht aber ungefähr einem Drittel der Spiegelfläche, der berechnete Schärfentiefebereich von 1,49mm ist in der Größenordnung sichtbar.

Da eine scharfe Darstellung der Umgebung mit diesem Spiegel nicht möglich ist, stellt sich die Frage in wie weit sich die Schärfe auf die konkrete Anwendung auswirkt und ob ein scharfes Bild zur Verfolgung eines Farbpunktes benötigt wird. Dieser Punkt könnte die Qualität des Gesamtsystems stark negativ beeinflussen. Die Lösung besteht darin entweder die Linse der Kamera durch eine besser Geeignete auszutauschen oder einen größeren Spiegel zu fertigen, welcher dann höher hängen würde.

Das Übersehen des Schärfebereiches bei der Berechnung des Spiegels unterstützt die Aussage in Kapitel 3.2.3, die besagt, dass Simulationen Informationen lediglich aufbereiten und keine zusätzlichen Erkenntnisse liefern können. Zusätzlich zeigt es, wie wichtig ein genaues Studium des zu simulierenden Szenarios für die Qualität einer Simulation ist. In einem nächsten Schritt sollte die Tiefenschärfe von der Simulation berücksichtigt werden.

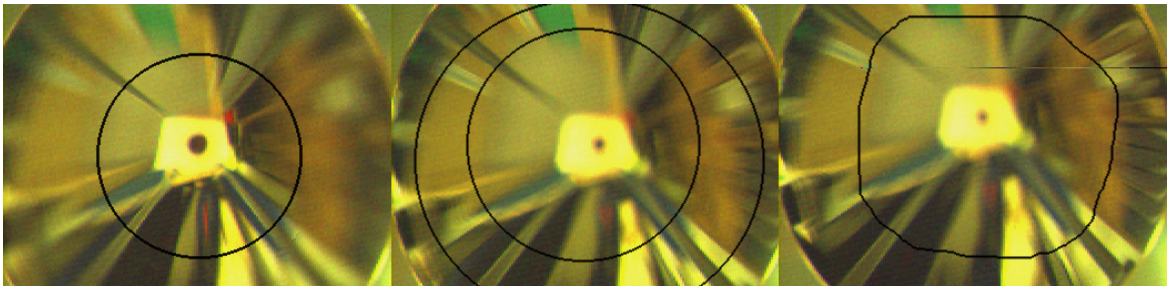

Abbildung 4.20: Wandernde Unschärfekreise auf kegelförmigen Spiegel

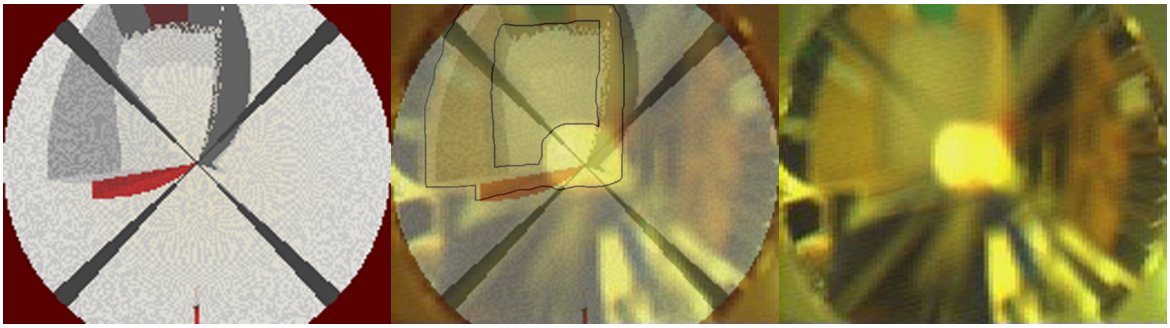

Abbildung 4.21: Vergleich Realität und Simulation bei kegelförmigen Spiegel

Wird der Sichtbereich betrachtet, so sieht das Ergebnis deutlich besser aus. Wie Abbildung 4.21 zeigt, nähern sich die Proportionen des Spielfeldes gut an. Wie auch schon beim vorherigen Spiegel aus Kapitel 4.3.4 kommen die Ungenauigkeiten, dass es in der Realität unmöglich ist, den Spiegel perfekt auszurichten. Bereits kleine Änderungen in Höhe, Neigung oder Position verändern das Bild maßgeblich. Das trifft besonders für diesen kleinen Spiegel zu. Trotz dieser Ungenauigkeit ist die Qualität der Simulation durchaus brauchbar. Die sichtbare Fläche ist sehr ähnlich. Auch die Entfernung zum hinteren Tor und die Einbuchtung des vorderen Tores kommen gut hin.

Deutlich zu erkennen ist der große helle Fleck im Zentrum des Spiegels. Dieser entsteht dadurch, dass die Spitze des Kegels an der Drehmaschine abgestumpft wurde. Das wird normalerweise getan um Verletzungen zu vermeiden. Bei dieser Anwendung ist es jedoch kontraproduktiv und sollte beim nächsten Spiegel unterlassen werden.

4.5 Fazit: Bewertung der implementierten Simulation

Nachdem zwei sehr unterschiedliche Spiegel getestet wurden und zu ähnlich guten Ergebnissen geführt haben, kann die Simulationssoftware für diesen Zweck als ausreichend genau

betrachtet werden. Beachtet man die Toleranzen, bietet die Simulation auf jeden Fall eine gute Grundlage für einen Spiegel. Was jedoch wichtiger ist, die Simulationssoftware ist in der Lage nach sehr kurzer Zeit eine Idee davon zu vermitteln, wie die Bilder einer omnidirektionalen Kamera aussehen. Damit können bestimmte Spiegelformen für bestimmte Zwecke frühzeitig ausgeschlossen werden. In diesem Beispiel waren es die kugelförmigen Spiegel, die nicht in Frage kamen. In einem anderen Beispiel könnten es die kegelförmigen Spiegel sein, die sich für eine Linienverfolgung disqualifizieren.

5 Aufbau der Kamera

Nachdem die Simulationssoftware die Daten für einen brauchbaren Spiegel geliefert hat, kann der Bau der Kamera begonnen werden. Dieses Kapitel umfasst alle Punkte, die mit dem Bau zu tun haben, bis hin zur fertig einsetzbaren Kamera.

5.1 Ausgangspunkt

Grundlage für die Kamera bildet die in Kapitel 4.2 vorgestellte CMU-CAM2. Das hier verwendete Modell besteht aus einer 55 Mal 55 Millimeter großen Platine und einem darauf gesteckten Kameramodul. An der Kopfseite der Hauptplatine befinden sich zwei Bohrungen für Halterungen und die Buchse für das aufsteckbare Kameramodul. Auf der Unterseite der Platine befinden sich die Elektronik und diverse Stecker für Anschlüsse und Jumper für verschiedene Betriebsmodi. Die Bauteile nutzen die gesamte Platinenfläche, bis sehr kurz vor dem Rand der Platine. Im Lieferumfang befindet sich ein Datenkabel mit neunpoligem SUB-D Stecker für die serielle Verbindung mit einem Computer und ein Kabel mit passendem Stecker für die Buchse zur Spannungsversorgung der Kamera.

5.2 Mechanische Designentscheidungen

Die zwei Bohrungen für Halterungen sollten für die Aufhängung verwendet werden. Diese erlauben ein stabiles Verbinden der Hauptplatine mit dem Bauteil, welches die Kamera auf der obersten Ebene des Roboters hält. Die horizontale Lage der Kamera macht eine weitere Halterung auf der gegenüberliegenden Seite der Platine notwendig. Dieses erhöht die Stabilität der Kamera und bietet Auflageflächen für die Spiegelhalterung. Die Herausforderung bestand darin, eine Halterung zu konstruieren, welche die Platine fest an einer Position hält. Dieses wird sehr durch die starke Belegung der Platine bis zum Rand erschwert. An der Fußseite werden unter anderem die Spannungsversorgung und das Datenkabel von unten mit der Kamera verbunden. Dennoch muss hier die zweite Halterung die Platine greifen, was die Größe der angrenzenden Stecker limitiert. Da das mitgelieferte Datenkabel auf der Seite der Kamera einen recht großen Stecker hat, musste ein neues Datenkabel gefertigt

Abbildung 5.1: Halterung für Spiegel aus der Panoramafotografie

werden. Weiterhin wurde der Stecker für die Spannungsversorgung erneuert, weil die Kabel des ursprünglichen Steckers durch die mechanische Belastung im Testbetrieb gebrochen waren.

Da das Kameramodul auf der CMU-CAM2 lediglich an einer Stelle gesteckt ist, kann man es in den Grenzen des Steckverbinders, um einige Grad bewegen. Das bewirkt eine große Veränderung des Bildes bei einer omnidirektionalen Kamera. Bei dieser Art von Kamera ist das Objektiv idealerweise immer auf genau einen Punkt des Spiegels gerichtet. Um das Spiel zu beseitigen, ist eine komplizierte Kunststoffkonstruktion entstanden. Sie nutzt die Bohrungen in der unteren Platine und unterstützt das Kameramodul, sodass sich dieses nicht mehr bewegen kann. Für die Halterungen und Stützen auf den Kameramodulen wird Kunststoff verwendet, da sich viele Kontakte und Lötstellen auf beiden Seiten der Platinen befinden. Neben der Eigenschaft von Kunststoff sich leicht verarbeiten zu lassen, spart es als Nichtleiter die Notwendigkeit eine Abdeckung für die Platine zu bauen. Das macht Kunststoff im Bereich der Kamera zum bevorzugten Material.

Für die Konstruktion der Spiegelhalterung gibt es drei Möglichkeiten. Abbildung 5.1 zeigt eine typische Spiegelhalterung aus der Panoramafotografie. Dort werden Spiegel oft mithilfe einer Stange gehalten. Die Stange geht durch die Spitze des Spiegels und ist am Spiegel und an einer Scheibe über der Linse verschraubt. In der Fotografie stellt das eine gute Lösung dar, wenn parabolische oder halb kugelförmige Spiegel zum Einsatz kommen. Diese Spiegelformen zeigen die Kamera im Zentrum des Bildes. Die Kamera ist auf dem Zielbild nicht gewollt, womit dieser Ausschnitt des Bildes für die Aufnahme nicht relevant ist. Weiterhin haben die eingesetzten Fotoapparate eine deutlich größere Linse, als die in dieser Arbeit verwendete Platinenkamera. Dadurch stellt diese Art der Halterung für die Fotografie im Gegensatz zur Platinenkamera, kein Problem da.

Eine zweite Lösung ist eine Halterung aus drei oder mehr dünnen Stangen, welche sich wie ein Teelichthalter nach oben strecken. Auf diese Stangen kann eine Spiegelhalterung oder der Spiegel direkt befestigt werden. Dieses ist eine gute Konstruktion für größere Spiegel, da die Haltestangen hier nur einen geringen Teil der verfügbaren Reflexionsfläche bedecken.

Die Halterung kann so von der Objekterkennung ignoriert werden und bietet eine sehr hohe Stabilität. Weitere optische Verschlechterungen gibt es bei dieser Art der Halterung nicht.

Da der Spiegel für die omnidirektionale Kamera möglichst klein ausfallen sollte, ist die Entscheidung auf ein Rohr aus Acrylglas gefallen. Akrylglas bietet zwei Vorteile gegenüber anderen Materialien, wie Plexiglas oder Glas. Es hat eine höhere Lichtdurchlässigkeit als Plexiglas und ist leichter zu verarbeiten als Glas. Gegenüber Glas bietet Akrylglas zudem weniger Gefahrenpotenzial, da es nicht zerbricht und somit keine scharfen Splitter entstehen können. Auf diese Stützkonstruktion ist ein Deckel aus Aluminium gesetzt. In der Mitte des Deckels ist ein Gewinde geschnitten, in das eine Gewindestange gedreht wird. An der Gewindestange können verschiedene Spiegel befestigt werden. Das bietet zwei Vorteile. Zum Einen ist der Spiegel über die Gewindestange höhenverstellbar, zum Anderen ist es leicht möglich den Spiegel auszutauschen. Dafür muss lediglich der Spiegel von der Gewindestange gedreht werden und durch einen anderen ersetzt werden. Das erhöht die Flexibilität der Konstruktion. Gehalten wird die Gewindestange durch zwei Kontermuttern, die sich auf der Unter- und Oberseite der Aluminiumhalterung befinden.

Das Akrylrohr hat genau den Durchmesser des Kameraobjektives. Damit kann es über das Objektiv gesteckt werden und wird durch dieses zentriert und gehalten. Um eine höhere Stabilität und Sicherheit zu erreichen, ist eine Halterung aus Kunststoff gefertigt worden, welches an die Außenseite des Akrylrohres angepasst ist. Der Aufbau des Kameramoduls macht dieses notwendig. Das Kameramodul besteht aus zwei Teilen, der Platine mit dem Bildsensor und dem Objektiv. Das Objektiv befindet sich gerade über dem Sensor und ist lediglich an zwei kleinen Schrauben mit der Platine verbunden. Die gefertigte Halterung hat die Aufgabe, das Rohr zu stabilisieren und Kräfte aufzunehmen. So zerren zum Beispiel Fliehkräfte nicht am Objektiv und die Ausrichtung des Objektives zum Bildsensor bleibt stabil.

5.3 Spiegelbau

Die Ergebnisse aus Kapitel 4 führen zu der Notwendigkeit einen kegelförmigen Spiegel für die omnidirektionale Kamera zu fertigen. Bei der Fertigung gab es zwei Versuche. Der erste Versuch bestand darin, einen Spiegel aus Metall zu fertigen und diesen so stark zu polieren, dass eine ausreichend gute Reflexion im Spiegel entsteht. Als Material wurde zunächst Aluminium gewählt. Aluminium ermöglicht den Bau eines sehr leichten Spiegels, jedoch blieb die Oberfläche auch nach intensivem Polieren zu matt. Für den zweiten Versuch sollte das Material des Spiegels veredelt werden, was zu einem sehr guten Resultat geführt hat. Die jeweiligen Ergebnisse zeigt das Bild 5.2.

Für die Veredelung kommt die Galvanotechnik zum Einsatz, auf die hier kurz eingegangen wird. Beim Galvanisieren wird eine sehr dünne Schicht eines Metalles auf ein Objekt auf-

Abbildung 5.2: Links kegelförmiger Spiegel, hochpoliert aus Alu, rechts kegelförmiger Spiegel, hochpoliert aus Kupfer, veredelt mit Silber

getragen. Dafür gibt es verschiedene Techniken. Für diese Arbeit sind zwei Verfahren interessant, da diese für den Heimgebrauch verfügbar sind. Das eine Verfahren ist das „Abscheiden von Metall mit äußerer Stromquelle in einem Tauchbad“, das zweite Verfahren ist das so genannte „Tampongalvanisieren“. Für weitergehende Informationen können die Werke „Praktische Galvanotechnik“ Heinz Leuze (1984) und „Galvanotechnik in Frage und Antwort“ Gaida (1983) aus dem Literaturverzeichnis verwendet werden.

Beim Galvanisieren in einem Tauchbad wird das Elektrolyt in eine Wanne gegeben. Soll die Innenseite einer Schüssel, wie zum Beispiel bei einem Reflektor, galvanisiert werden, kann das Elektrolyt auch direkt in das Objekt gegeben werden. Das zu galvanisierende Objekt wird als Kathode an den negativen Pol der Spannungsquelle angeschlossen. Als Anode wird beim Versilbern eine Edelstahlplatte an den positiven Pol angeschlossen. Das Material der Anode ist abhängig vom Material, welches auf das Objekt aufgetragen werden soll. Ist alles angeschlossen und das Objekt in das Elektrolyt eingetaucht, wird eine vorher bestimmte Spannung angelegt. Diese ist abhängig von der Oberfläche des zu galvanisierenden Objektes. Näheres hierzu findet sich in der Anleitung zum Galvanisierset, welches im Anhang zu dieser Arbeit zu finden ist.

Beim Tampongalvanisieren wird das Elektrolyt an die Oberfläche des Objektes gebracht, ohne dass dieses eingetaucht werden muss. Das Elektrolyt wird in einem Schwamm gehalten, welcher an der Spitze einer Apparatur befestigt ist. Wie beim Galvanisieren im Tauchbad ist auch hier das Objekt als Kathode an den negativen Pol der Spannungsquelle angeschlossen. Die Anode ist an den positiven Pol der Spannungsquelle angeschlossen und befindet sich unter dem Schwamm an der Apparatur. Ist die Spannungsquelle aktiviert worden und der Schwamm mit Elektrolyt getränkt, wird der Schwamm mit leichtem Druck über die zu galvanisierende Fläche geführt. Dabei wird das Material nicht verstrichen. Es findet der gleiche elektrochemische Prozess, wie beim Tauchverfahren, statt. Es muss also ein gleichmäßiger Stromfluss stattfinden.

Bei beiden Verfahren müssen einige Vorbereitungen und Berechnungen getätigt werden. Zunächst muss das Material gewählt werden, mit dem der Spiegel veredelt werden soll. Das

war im Fall des Spiegels für diese Arbeit sehr einfach. Die für den Heimgebrauch verfügbaren Galvanisiersets bieten Möglichkeiten für das Verkupfern, Vernickeln, Versilbern oder Vergolden. Vergoldung und Verkupferung konnte von vornherein ausgeschlossen werden, da dort die Farben auf dem Spiegel verfälscht würden. Übrig blieben die Möglichkeit der Versilberung oder Vernickelung. Die Wahl fiel auf die Versilberung, da ein Einsteigerset mit Nickelelektrolyt nicht auffindbar war. Die Versilberung ist auch ohne diesen Zwang eine gute Wahl, da Silber ein sehr hohes Reflexionsvermögen von 80 bis 99 Prozent hat. Der Nachteil bei Silber gegenüber Nickel besteht darin, dass Silber durch die Feuchtigkeit und den Schwefel in der Luft anläuft und sich Silbersulfidschichten bilden (Heinz Leuze, 1984, S. 321). Das macht es notwendig den Spiegel gelegentlich zu polieren. Daher könnte es sich später lohnen einen Spiegel zu vernickeln und diesen auf Hochglanz zu polieren. Sollte die Reflexion eines Spiegels aus Nickel ausreichen, könnte das einen pflegeleichteren Spiegel ergeben. Alternativ könnte auch versucht werden, den Spiegel zu versiegeln, was naturgemäß die Reflexionseigenschaften des Silbers verringert.

Bei der Silbergalvanisierung wird oft mit giftigen Chemikalien wie Cyanid und Quecksilber gearbeitet. Das hier eingesetzte Verfahren ist laut Packungsbeilage ungefährlich. Auch giftige Gase sollen nicht entstehen. Leider ist weder auf den Flaschen, noch im Beiheft die Zusammensetzung des Elektrolyten beschrieben.

Als Material für den Spiegel wurde Kupfer gewählt. Kupfer ist für die Galvanisierung gut geeignet und in brauchbarer Form in der Zentralwerkstatt der HAW-Hamburg verfügbar. Der Spiegel wurde an der Drehmaschine gefertigt und stark poliert. Die Silberschicht wird sehr dünn und kann keine Rillen ausfüllen. Darum muss die Oberfläche der Reflexionsfläche besonders eben sein. Anschließend ist die Reflexionsfläche mit dem Reiniger aus dem Galvaniserset und Reinigungsspiritus gesäubert worden. Der gesäuberte Spiegel wurde dann sehr sorgfältig unter laufendem Wasser gespült, um die Reste der Reiniger zu entfernen. Damit waren die Vorbereitungen abgeschlossen.

Die im Galvanisierset enthaltene Wanne ist für das Galvanisieren des kleinen Spiegels sehr groß. Um Elektrolyt zu sparen, wurde darum ein anderes Gefäß eingesetzt. Angeboten hat sich ein Schnapsglas mit Einkerbungen durch welche die Kabel für die Spannung geführt werden können. Für die Halterung des Spiegels dienen Unterlegscheiben. Diese haben in der Mitte ein Loch in welches die Gewindestange, die auch später den Spiegel halten soll, befestigt werden kann. Als Spannungsquelle kommt ein Labornetzteil zum Einsatz. Dieses erlaubt es den Strom zu begrenzen was den Einsatz von Widerständen im Aufbau spart und die Genauigkeit erhöht. Spannung und Strom können am Gerät mithilfe eines Multimeters sehr genau eingestellt werden. Den Aufbau für das Galvanisieren zeigt Abbildung 5.3.

Bei größeren Spiegeln macht es Sinn auf das Tamponverfahren umzusteigen. Das spart Elektrolyt und macht die Berechnung der Oberfläche des Spiegels überflüssig.

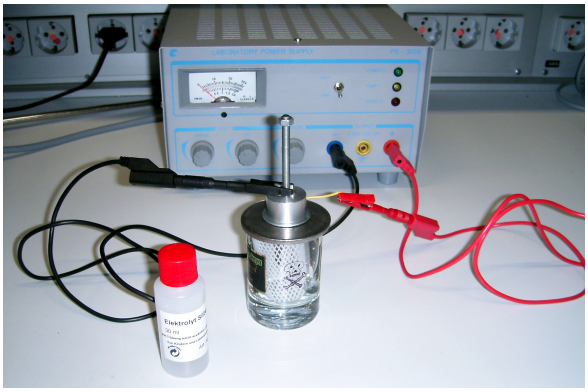

Abbildung 5.3: Aufbau für das Galvanisieren des Spiegels

Für die richtige Spannung beim Tauchverfahren ist es notwendig, die Oberfläche des Spiegels zu bestimmen. Diese Berechnung befindet sich als Beispiel im jetzt folgenden Kapitel.

5.3.1 Bestimmung der Oberfläche des Spiegels

Der Spiegel besteht aus zwei geometrischen Formen. Einem Kegel, welcher den eigentlichen Spiegel darstellt und einem Zylinder, mit dem der Spiegel gehalten werden soll. Die zu galvanisierende Oberfläche besteht daher aus der Summe der Mantelfläche des Zylinders und der Mantelfläche des Kegels. Die Basisfläche des Kegels und die Oberseite des Zylinders können hier außer acht gelassen werden, da diese nicht versilbert werden sollen.

Die für die Kegelberechnung notwendigen Größen, finden sich in Abbildung 5.4[1]. Die Größen für die Berechnung des Zylinders, befinden sich in Abbildung 5.5. Die Gleichungen, für die Bestimmung der Oberfläche des Spiegels, sind in Abbildung 5.3.1 aufgeführt.

[1] Die geometrischen Zeichnungen dieser Arbeit wurden mit dem kostenlosen und plattformunabhängigen Programm GeoGebra (GeoGebra) erstellt.

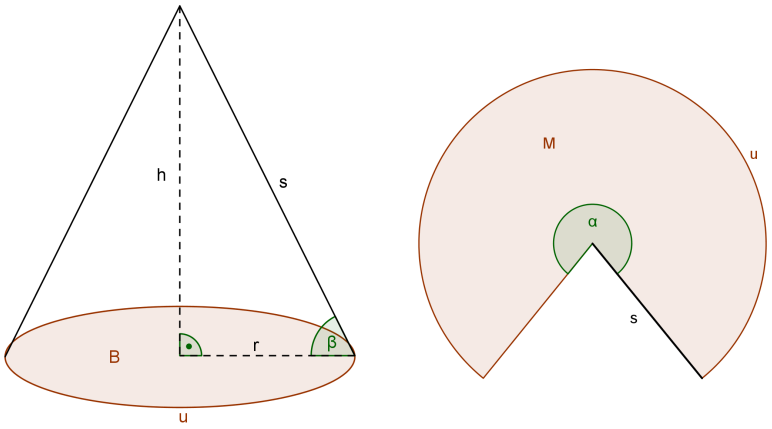

Abbildung 5.4: Größen für Kegelberechnung

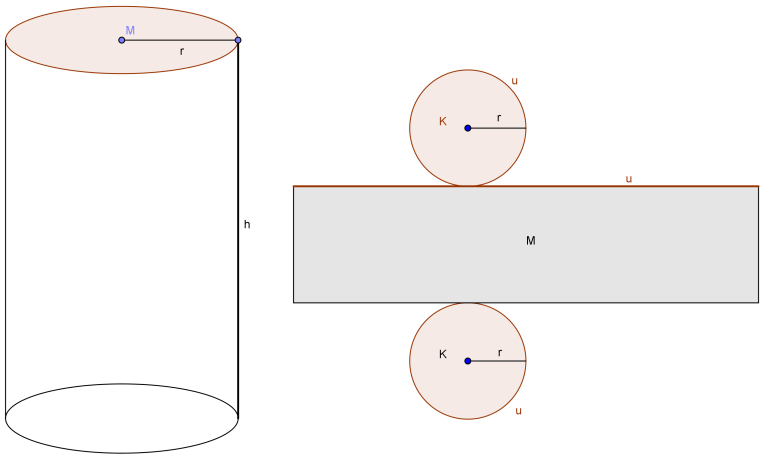

Abbildung 5.5: Größen für Zylinderberechnung

Mantelfläche Kegel: (5.1)

$$M_{Kegel} = \pi \cdot s \cdot r$$

Kantenlänge: (5.2)

$$s = \sqrt{r^2 + h^2}$$

Höhe: (5.3)

$$h = r \cdot \tan(\beta)$$

$$M_{Kegel} = \pi \cdot \sqrt{r^2 + (r \cdot \tan(\beta))^2} \cdot r \quad (5.4)$$

$$M_{Kegel} = \pi \cdot \sqrt{(7{,}5mm)^2 + (7{,}5mm \cdot \tan(32{,}48°))^2} \cdot 7{,}5mm$$

$$M_{Kegel} = 209{,}48mm^2$$

Mantelfläche Zylinder: (5.5)

$$M_{Zylinder} = 2\pi r \cdot h$$

$$M_{Zylinder} = 2\pi \cdot 7{,}5mm \cdot 5mm$$

$$M_{Zylinder} = 235{,}62mm^2$$

Mantelfläche Spiegel: (5.6)

$$M_{Spiegel} = M_{Kegel} + M_{Zylinder}$$

$$M_{Spiegel} = 209{,}48mm^2 + 235{,}62mm^2$$

$$M_{Spiegel} \approx 445{,}1mm^2$$

Nachdem die zu versilbernde Oberfläche bestimmt ist, muss der Strom ermittelt werden, der für die Galvanisierung benötigt wird. Diese richtige Spannung ist für die Glanzbildung von Bedeutung. Die benötigte Stromdichte für einen Quadratzentimeter kann aus Tabelle 5.1 entnommen werden, welche aus der Anleitung des Galvanisiersets stammt. Diese muss, wie im Beispiel 5.3.1, mit der Fläche des Objektes multipliziert werden.

Elektrolyt	Spannung	Stromdichte
Silber (sauer)	$1-3V$	$10-30mA/cm^2$
Kupfer (sauer)	$2-3V$	$30-35mA/cm^2$
Nickel (sauer)	$2-4V$	$10-50mA/cm^2$
Gold (alkalisch)	$2-3V$	$20-50mA/cm^2$

Tabelle 5.1: Für Galvanisierung benötigter Strom

Berechnung der benötigten Stromstärke für die Galvanisierung

Fläche der zu galvanisierenden Fläche in Zentimeter: (5.7)

$$M_{Spiegel} \approx 445{,}1mm^2 = 44{,}51cm^2 \quad (5.8)$$

Spannung: $3V$ (5.9)

Stromstärke: (5.10)

$$44{,}51cm^2 \cdot 30mA = 1335{,}3mA = 1{,}3353A$$

Spannung: $1V$ (5.11)

Stromstärke: (5.12)

$$44{,}51cm^2 \cdot 10mA = 445{,}1mA = 0{,}4451A$$

Bei der Versilberung des Spiegels wurden mithilfe des Labormessgerätes $1,33A$ eingestellt. Dieses hat zu einem sehr guten Ergebnis geführt. Der Spiegel hat eine gleichmäßige Oberfläche und gute Reflexionseigenschaften.

5.4 Auswahl des Steuermodules

An dieser Stelle ist es notwendig die Frage zu beantworten, wie die Informationen der Kamera zu Richtungsinformationen umgewandelt werden. Hierfür gibt es zwei Möglichkeiten. Entweder man überlässt diese Aufgabe dem Zielsystem oder man setzt ein zusätzliches System ein, welches die Daten vorverarbeitet und dem Zielsystem zur Verfügung stellt.

Überlässt man das Umrechnen der Informationen dem Zielsystem, spart man die Entwicklung weiterer Hardware. Zusätzlich spart diese Lösung Energie und Platz, da keine weitere Elektronik benötigt wird. Die Logik zur Gewinnung der Richtungsinformation befindet sich in

diesem Fall innerhalb einer Bibliothek. Diese Bibliothek wird in das Programm des Zielsystems integriert und genutzt.

Wird zusätzliche Hardware für die Auswertung und Kommunikation mit dem Zielsystem entwickelt, benötigt die Kamera mehr elektrische Energie und mehr Platz. Dafür wird andere Flexibilität gewonnen. Die zusätzliche Hardware kann als Middleware fungieren. Das erlaubt den Austausch der Platinenkamera gegen eine leistungsfähigere Kamera ohne das Programm auf dem Steuerungsmodul des Roboters zu erneuern. Hinzu kommt die Möglichkeit zusätzliche Schnittstellen zur Kamera, wie I^2C oder CAN, zu schaffen. Ein weiterer wichtiger Punkt ist die unkomplizierte Einsatzfähigkeit. Auswertungshardware innerhalb der Kamera erlaubt es genaue Angaben über die Leistungsfähigkeit der Kamera zu machen. Werden die Informationen der Platinenkamera auf dem Zielsystem umgewandelt, so hängt die Leistungsfähigkeit stark von der Geschwindigkeit des dort eingesetzten Systems ab. Eine Bibliothek kann alle Berechnungen ausführen, allerdings gibt es je nach eingesetztem Controller, Schwierigkeiten mit der Performance oder dem benötigten Speicher.

Die in dieser Arbeit gefertigte, omnidirektionale Kamera, soll später als Sensor zur Richtungserkennung dienen. Richtungen können auf zwei Weisen bestimmt werden. Entweder man erstellt ein Raster und definiert eine bestimmte Richtungsangabe für jedes Feld oder man gibt die Richtung als Winkel weiter.

Die Richtungserkennung über ein kleines Raster ist sehr einfach, jedoch nicht sehr präzise. Je feinmaschiger das Raster ist, desto präziser wird es. Allerdings nimmt mit der Anzahl der Felder die Komplexität in der Berechnung oder dem Speicherbedarf stark zu. Das liegt daran, dass jedes Feld im Raster eine bestimmte Position repräsentiert. Diese muss als Information gespeichert oder berechnet werden.

Wird das Raster sehr feinmaschig, macht es Sinn dieses als karthetisches Koordinatensystem zu betrachten. Dadurch entsteht die Möglichkeit aus dem Raster Polarkoordinaten zu gewinnen, die eine Entfernung und eine Richtung enthalten. Die Richtung ist genau das, was die Kamera als Sensor zurückgeben soll. Die Umrechnung eines solchen karthetischen Koordinatensystemes in ein Polarkoordinatensystem ist in der Mathematik beschrieben und kann leicht angewandt werden. Auf die Rechnung wird in 5.5.1 eingegangen. Bei der Umrechnung wird mit dem Arcus Tangens gerechnet, damit macht es keinen Sinn, für diese Art der Winkelbestimmung, einem Acht-Bit-Controller zu verwenden. Aufgrund des sehr großen Genauigkeitsgewinnes ist dieses Vorgehen gegenüber dem groben Raster vorgezogen worden.

Aus Zeitgründen ist in dieser Arbeit auf die Entwicklung einer Auswertungshardware für die Kamera verzichtet worden. Als Zielsystem für die Beispielimplementierung kommt ein im Labor verfügbares System mit dem Namen RCUBE zum Einsatz (Abbildung 5.6). Dieses befindet sich bereits auf der Ausgangsplattform des Roboters und hat verschiedene Vorteile. Es besteht aus drei Komponenten. Die erste Komponente ist das im Robotlabor eingesetzte

Abbildung 5.6: RCUBE, Bild stammt von der Homepage des Projektes der FH Brandenburg RCUBE

AKSEN-Board. Die zweite Komponente ist ein „VIO-Board“, welches für die Videodigitalisierung über 4 PAL-Eingänge und einen Videoausgang verfügt. Als dritte Komponente bringt der RCUBE ein CPU-Board mit. Die einzelnen Komponenten sind zu einem stabilen Modul verbunden und können über einen CAN-Bus miteinander kommunizieren.

Das oberste Modul ist das AKSEN-BOARD der FH-Brandenburg. Dieses wird im Rahmen des Projektes „Mobile Roboter“ eingesetzt. Es verfügt unter anderem über einen Acht-Bit-Mikrocontroller, einen CANanschluss, eine serielle Schnittstelle, 16 analoge und 16 digitale Ports, drei Anschlüsse für Servos, vier Anschlüsse für Getriebemotoren und vier Anschlüsse für LEDs. Für die Programmierung bietet es eine gut dokumentierte Bibliothek, mit deren Hilfe alle Funktionen des Boards genutzt werden können. Es muss also nicht in die Mikrocontroller Programmierung eingestiegen werden. Dieses Modul ist bekannt und hat sich bereits mehrere Semester in Robot Laboren verschiedener Hochschulen bewährt.

Das mittlere Modul ist das VIO-BOARD. Es wird in dieser Arbeit nicht verwendet, da die Logik für die Bilderkennung bereits in der CMU-CAM2 vorhanden ist. Sollte eine andere Arbeit auf einer Kamera ohne Logik aufgebaut sein, bietet dieses Modul eine Möglichkeit eine Kamera anzuschließen. Als Startpunkt könnte die Diplomarbeit von Lars Brandt dienen. Sie beschäftigt sich mit Objektverfolgung und nutzt diese Schnittstelle Brandt (2005).

Beim untersten Modul handelt es sich um das LART-Board. Dieses an der TU Delft in den Niederlanden entwickelte Modul arbeitet mit einer 220MHz digital SA-1100 StrongARM CPU. Es verfügt über 32 MB EDO-RAM und einem vier Mega Byte großen „Intel Fast boot block Flash Memory“. Damit hat es laut Herstellerangaben eine Leistungsaufnahme von weniger als einem Watt und eine Performance von mehr als 200 Millionen Instruktionen pro Sekunde (MIPS). Auf dem Modul läuft ein embedded Linux, auf das man über zwei serielle Schnitt-

stellen zugreifen kann. Für den Zugriff wird lediglich ein Terminal Programm benötigt. An eine dieser Schnittstellen kann die Kamera bequem angeschlossen werden. Die zweite serielle Schnittstelle ist noch frei und kann für die Kommunikation mit dem Roboter genutzt werden. Auf diese Weise kann das Programm des Roboters bequem auf den neusten Stand gebracht, oder eine Kommunikation während des Betriebes ermöglicht werden. Die Kommunikation während des Betriebes ist zum Beispiel für die Fehlererkennung von Vorteil. Interessanter ist sie jedoch für genetische Verfahren wie die Robotersteuerung über neuronale Netze oder die Kommunikation zwischen mehreren Robotern.

Der RCUBE bietet alle notwendigen Anschlüsse. Zusätzlich beinhaltet er das AKSEN-Board, welches sich im Labor und darüber hinaus bereits bewährt hat. Mit dem VIO-Board bietet er zusätzliche Funktionalität und einen Einstiegspunkt für weitere Arbeiten. Das LART-Board schließlich hat genügend Rechenleistung für die in dieser Arbeit gestellten Anforderungen und alle notwendigen Anschlüsse und ist mit seinem Linux Betriebssystem sehr flexibel. Das macht den RCUBE zu einer sinnvollen Grundlage für eine leistungsfähige Experimentierplattform mit omnidirektionaler Kamera.

5.5 Kameralogik

Die Kameralogik befindet sich zwischen den Anfragen des Roboters und der Platinenkamera, die Teil der omnidirektionalen Kamera ist. Sie initialisiert die Platinenkamera und ist anschließend für die Vorverarbeitung der Farbschwerpunktsinformationen aus der Objektverfolgung verantwortlich.

5.5.1 Umwandlung Positionsangaben in Richtungsinformationen

Die Kameralogik ist der Vermittler zwischen der eingesetzten Kamera und dem Roboter und liefert Richtungsangaben zu einem Objekt. Damit sie diese Aufgabe erledigen kann, müssen die Informationen der Kamera in Richtungsinformation umgewandelt werden. Das macht einige einfache Rechnungen notwendig.

Im Falle der in dieser Arbeit verwendeten CMU-CAM2 bekommt die Kameralogik Positionsangaben in Form von Koordinaten. Diese Koordinaten entsprechen der Pixelmatrix der Kamera, welche ihren Nullpunkt in der oberen linken Ecke hat. Da das Kamerabild aufgrund der Bauweise der Kamera im Zentrum immer den Roboter und außenrum die Umwelt hat, muss das Koordinatensystem des Bildes dahin verschoben werden. Abbildung 5.7 zeigt die Verschiebung der Punkte in dieses neue Koordinatensystem anhand eines Beispielbildes mit 50 Mal 50 Pixeln.

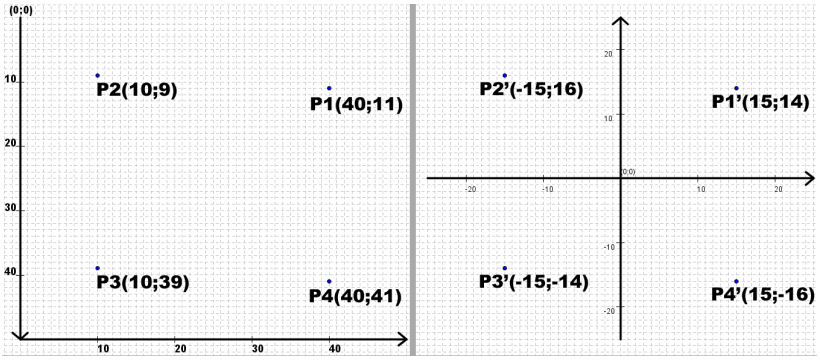

Abbildung 5.7: Verschiebung des Koordinatensystems für anschließende Berechnung des Winkels zu einem Punkt

Für die Umrechnung kann eine einfache Formel zu Hilfe genommen werden. Die folgende Rechnung zeigt den Rechenweg für den Punkt $P1$. Hierbei wird ein Koordinatensystem von 50 Mal 50 Pixeln angenommen. In der allgemeinen Form entspricht x_max die Anzahl der Pixel in x-Richtung und y_max die Anzahl der Pixel in y-Richtung. x und y sind jeweils die von der Kamera erhaltenen Koordinaten eines Objektes, während x' und y' die Entsprechung dieser Koordinaten im neuen Koordinatensystem repräsentieren.

Allgemeine Form:

$$x' = -\left(\frac{x_max}{2} - x\right) \tag{5.13}$$

$$y' = \frac{y_max}{2} - y \tag{5.14}$$

Konkretes Beispiel anhand von P1(40;11) und einem Pixelfeld von 50 Mal 50 Pixeln:

$$x_1' = -\left(\frac{50}{2} - 40\right) = 15 \tag{5.15}$$

$$y_1' = \frac{50}{2} - 11 = 14 \tag{5.16}$$

P1' liegt bei $(15, 14)$. (5.17)

Nachdem das Koordinatensystem auf die Eigenschaften des Kamerabildes umgeformt wurde, kann der Winkel des Objektes zur Kamera errechnet werden. Die Mathematik bietet zwei Modelle an, mit denen Punkte eindeutig bestimmt sind. Das eine Modell ist das des kartesischen Koordinatensystemes, das andere Modell das des Polarkoordinatensystemes. Die Pixelmatrix des Kamerabildes entspricht dem kartesischen Koordinatensystem, jeder Punkt ist durch einen x- und einem y-Wert eindeutig definiert. Bei dem Polarkoordinatensystem wird jeder Punkt durch eine Entfernung zum Zentrum und einem Winkel definiert. Dieser

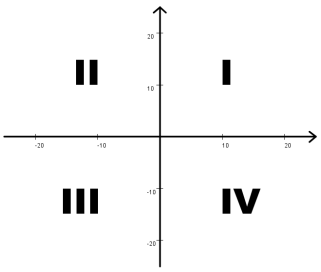

Abbildung 5.8: Quadranten eines Koordinatensystemes

Winkel ist genau die Information, die für die Kameralogik gesucht ist. Zwischen beiden Systemen gibt es einen Zusammenhang, der eine Umrechnung vom einen ins andere System erlaubt (Papula, 2001, S.42f). Diese Eigenschaft macht sich die Kameralogik zunutze.

Bevor die Objektkoordinaten aus dem kartesischen in das Polarkoordinatensystem überführt werden können, ist es sinnvoll eine weitere Umrechnung zu machen. Da die Kamera bei der verwendeten Form der omnidirektionalen Kamera auf einen Spiegel sieht, ist das Bild horizontal spiegelverkehrt. Das bedeutet, eine direkte Umrechnung der Koordinaten ergäbe eine Logik, welche zwar richtig erkennt, ob ein Objekt vor oder hinter der Kamera steht, welche allerdings links und rechts vertauscht. Aus diesem Grund muss die x-Koordinate des Objektpunktes mit minus Eins multipliziert werden.

Nach diesen Überlegungen kann die Rechnung beginnen. Für die Umrechnung vom kartesischen- in das Polarkoordinatensystem ist es wichtig, in welchem Quadranten sich der Punkt befindet. Die Quadranten sind wie in Abbildung 5.8 angeordnet. Die allgemeine Formel für die Umrechnung zeigt Tabelle 5.2.

Quadrant	I	II, III	IV
$\varphi =$	$\arctan\left(\frac{y}{x}\right)$	$\arctan\left(\frac{y}{x}\right) + \pi$	$\arctan\left(\frac{y}{x}\right) + 2\pi$

Tabelle 5.2: Umrechnung von Kartesischen in Polarkoordinaten

Mit dieser allgemeinen Formel aus Tabelle 5.2, kann der Winkel aus dem in 5.15 verwendeten Beispielpunkt P1'(15;14) ermittelt werden.

Der Vergleich der Position von P1' und Abbildung 5.8 zeigt, dass P1' im ersten Quadranten liegt. Mit Tabelle 5.2 ergibt sich daher folgende Formel zur Berechnung des Winkels zum

Punkt P1':

$$\varphi = \arctan\left(\frac{y}{x}\right) \tag{5.18}$$

$$\varphi = \arctan\left(\frac{14}{15}\right) \tag{5.19}$$

$$\varphi = 43{,}03 \tag{5.20}$$

Die Einteilung in 360 Grad für die Richtungen eines Objektes sind mehr als ausreichend. Aus Performancegründen wird daher auf Nachkommastellen verzichtet. Das gesuchte Objekt befindet sich also vorne links in Richtung 43 Grad zum Roboter. Dieser Wert wird auf Anfrage zurückgegeben.

Mit dieser Grundlage wird der Aufbau der Kameralogik interessant, welche die Koordinaten von der Platinenkamera bekommt und an die Robotsteuerung weiter gibt. Damit beschäftigt sich das nächste Kapitel.

5.5.2 Automatisches Einlesen der gesuchten Objektfarbe

Zum Einlesen der Farbe des zu suchenden Objektes müssen eine Reihe von Operationen durchgeführt werden. Vorbedingung ist hierbei, dass der Boden eine möglichst homogene klare Farbe hat, die sich von dem zu suchenden Objekt unterscheidet. Der Algorithmus zum Finden des Objektes soll hier mit Hilfe von Abbildung 5.9.erläutert werden.

Das erste Viertel in Abbildung 5.9 zeigt das Ausgangsbild aus Sicht der Kamera. Der braune Zylinder soll als Objekt für die spätere Verfolgung identifiziert werden. Wie man sehen kann, werden neben dem Objekt selber, viele zusätzliche Dinge angezeigt. Hierzu ist zu bedenken, dass in der Simulation der komplette Boden grün ist und das Bild nur wenig Farben enthält. Das gleiche Bild in der Wirklichkeit zeigt Millionen kleiner Farbkleckse in der Umgebung, die von Gegenständen oder Personen in der Umgebung der Kamera stammen. Aus dieser Vielfalt ist es nicht möglich auf das richtige Objekt zu schließen, denn zufällige, einfarbige Flächen sind nicht auszuschließen. Aus diesem Grund wird ein virtuelles Fenster definiert, welches wie im zweiten Bild von Abbildung 5.9 aussehen könnte. Das virtuelle Fenster beschränkt die Sicht auf einen definierten Bereich vor der Kamera und raubt ihr damit die Rundumsicht. Die Software der Kamera wird ihre Aktionen ausschließlich in diesem Bereich durchführen. Dieses Fenster muss für jede Kamera angepasst werden. Bei der Definition des Fensters sind zwei Dinge von Bedeutung. Das Fenster muss so gewählt werden, dass es einen Bereich definiert, der leicht zu finden ist und ausschließlich das zu suchende Objekt und den einfarbigen Boden zeigt. Hierfür kann beispielsweise eine DINA4 Seite als Maß verwendet werden, wie es Abbildung 5.10 zeigt. Als Zweites muss darauf geachtet werden,

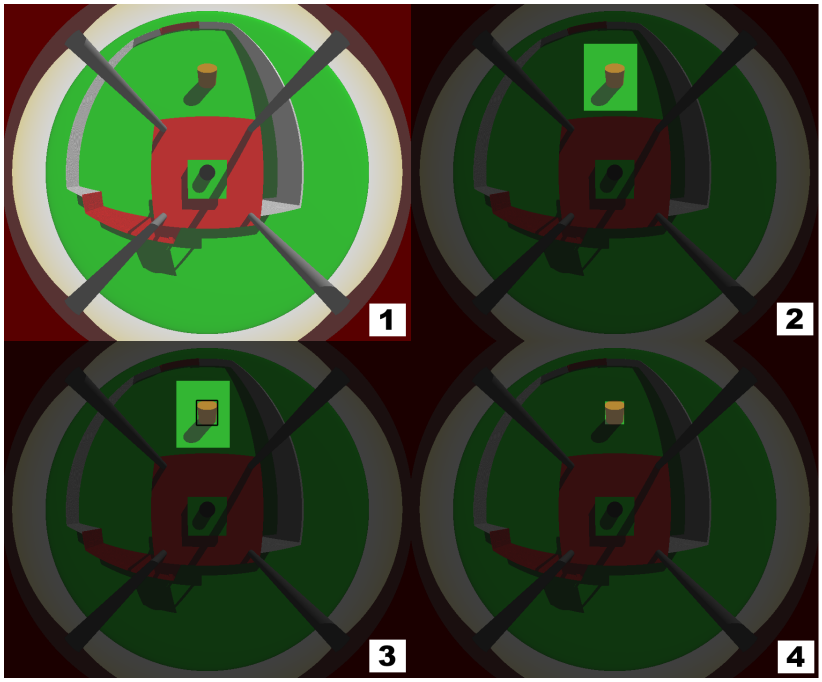

Abbildung 5.9: Bildreihe Objektfindung

dass die Bodenfläche gegenüber dem zu suchenden Objekt möglichst groß ist. Das ist für den nächsten Schritt entscheidend.

Nachdem das Fenster auf dem Objekt liegt, wird nach der Dominanten Farbe des Bildausschnittes gesucht. Da die einfarbige Fläche des Bodens deutlich größer ist, als die Fläche des gesuchten Objektes, ist das Ergebnis die Hintergrundfarbe. Nach dieser Aktion sind zwei Dinge bekannt. Das sind zum Einen die Hintergrundfarbe und zum Anderen die Information, dass es noch einen weiteren Farbbereich gibt, welcher sich von der Hintergrundfarbe unterscheidet. Das bedeutet im Umkehrschluss, dass eine invertierte Suche nach der im letzten Schritt gefundenen Farbe genau zum Objekt führen muss. Invertierte Suche bedeutet hierbei die Suche nach derjenigen Farbe, welche nicht angegeben ist. Das Ergebnis dieser Suche ist in Bild 3 von Abbildung 5.9 dargestellt. Neu hinzugekommen ist ein Kasten, der das Objekt genau umschließt. Zu diesem Zeitpunkt ist der Bereich bekannt, in dem die Farbe des zu suchenden Objektes dominant ist. Um die Farbe zu bekommen, muss das virtuelle Fenster auf diesen Bereich gesetzt werden. Damit ist die Kamera in der Lage mit der gleichen Funktionalität, wie zuvor beim Einlesen der Hintergrundfarbe, die Farbe des gesuchten Objektes einzulesen. Jetzt wo das Objekt und dessen Farbe identifiziert sind, wird die Kamera wieder auf ihre Betriebsauflösung gestellt und sieht wieder den Ausschnitt aus Bild eins in Abbildung 5.9. Die Objektverfolgung kann beginnen.

Nachdem in Kapitel 5.5.1 erläutert wurde, wie aus den von der Kamera zur Verfügung ge-

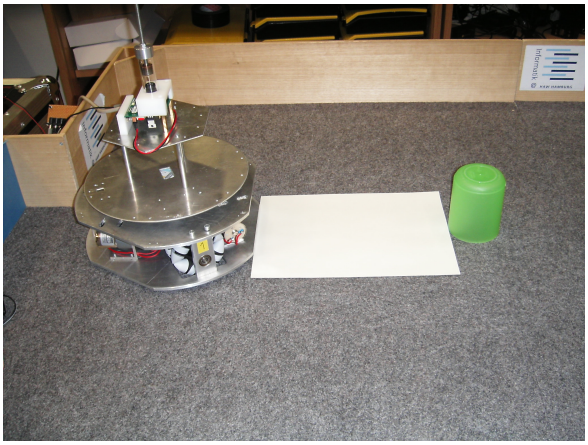

Abbildung 5.10: DINA4 Seite als Anschlag, um das Objekt in das virtuelle Fenster zu legen

stellten Daten ein Winkel berechnet werden kann und Kapitel 5.5.2 gezeigt hat, wie omnidirektionale Kameras Objekte für die spätere Verfolgung identifizieren können, kann jetzt auf den Aufbau der Kameralogik eingegangen werden.

5.5.3 Aufbau der Kameralogik

Der Aufbau der Kameralogik und ihr Platz im Gesamtsystem des Roboters werden in Abbildung 5.11 gezeigt. Der Hauptteil befindet sich im schraffierten Rahmen und ist in drei Teile geteilt. Das in grau gehaltene Hauptprogramm ist der Startpunkt des Programmes auf dem Steuerungsmodul der Kamera. Er verwendet den als Bibliothek gehaltenen Hauptteil. Die Komponente Testcases ist für die Entwicklung neuer Anwendungsfälle und Funktionalitäten hilfreich. Er gehört nicht in eine fertige Kameralogik. Da es bei dieser Arbeit jedoch um ein System zur Entwicklung omnidirektionaler Kameras geht, ist eine solche Komponente wichtig und sinnvoll. Damit kann die Qualität des ObjektTrackers gesichert werden, da neue Funktionalität zunächst in der Komponente Testcases implementiert wird. Hat sie sich dort bewährt, kommt der entstandene Code in die Komponente ObjektTracker und ist damit Teil der Kameralogik. Die in rot gehaltenen Komponenten „Logik Platinenkamera“ und „Robotsteuerung“ stellen zwei externe Systeme dar mit denen die Kamera später zusammenarbeiten muss. Im Beispiel der in dieser Arbeit implementierten omnidirektionalen Kamera wird die „Logik Platinenkamera“ von der CMU-CAM2[2] repräsentiert. Die „Übertragung über Schnittstelle zur Kamera“ ist in diesem Fall eine serielle Übertragung. Bei der Komponente „Robotsteuerung“ handelt es sich bei dieser konkreten Implementierung, um die Software, die auf dem Steuermodul des Roboters läuft. Dieses Steuermodul ist Teil des RCUBE, welcher in Kapitel 5.4

[2]Näheres zur Kamera im Kapitel 4.2

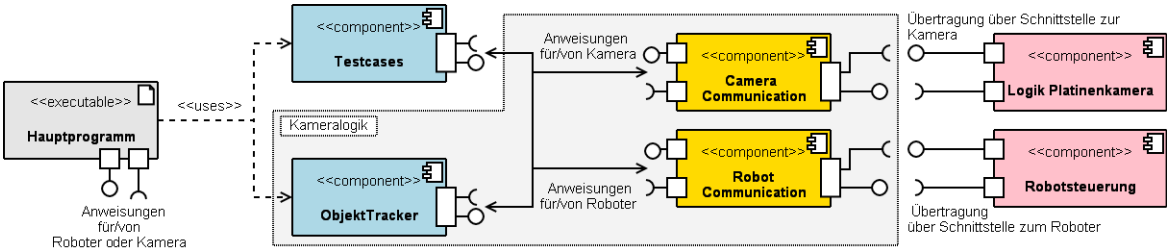

Abbildung 5.11: Aufbau Kameralogik im Kontext eines Gesamtsystemes einer omnidirektionalen Kamera für mobile Roboter

vorgestellt wurde. Die Kommunikation findet in diesem Fall über den CAN-Bus des verwendeten Systemes statt.

Der Hauptteil ist in die Bereiche ObjektTracker, CameraComunication und RobotComunication zu geteilt. Das soll die Portierbarkeit des Systems und den Komfort beim Programmieren erhöhen. Die in blau gehaltene Komponente ObjektTracker enthält die eigentliche Kameralogik. Hier werden alle Umrechnungen vorgenommen, die für die einfache Verwendung der Kamera als Sensor für einen Roboter notwendig sind.

Die Komponenten für die Kommunikation, welche in Gelb gehalten sind, beinhalten eine weitere Unterteilung. Diese wird in Abbildung 5.12 beispielhaft dargestellt. Jede dieser beiden Komponenten besteht aus zwei Teilen. Die Komponente „Command transformation“ abstrahiert die Befehle der Kamera oder des Roboters und stellt sicher, dass der Objekt Tracker Standardfunktionen abrufen kann ohne den eigentlichen Syntax für die dahinter stehende Komponente zu kennen. Die zweite Unterteilung wird von der Komponente „Use communication protocol“ repräsentiert. Sie führt die eigentliche Kommunikation auf dem Zielsystem durch. Diese Komponente nimmt auf der einen Seite Zeichenketten oder Ziffern an und kümmert sich darum, dass diese auf der anderen Seite über die zugrunde liegende Hardware verschickt werden können.

Die beschriebene Aufteilung innerhalb der Kommunikationskomponenten erlaubt es mit relativ wenig Aufwand die Kamera, die Robotsteuerung oder die Hardware, auf der die Kameralogik läuft, auszutauschen. Diese Portabilität ist sehr wichtig für ein System zur Entwicklung omnidirektionaler Kameras für mobile Roboter, wenn dieses System längerfristig nutzbar sein soll.

Wird das Hauptprogramm der Kameralogik gestartet, versucht es als Erstes einen Kontakt zur Kamera herzustellen. Hierfür schickt es zum Beispiel so lange „Reset“ Befehle zur Kamera, bis diese mit dem richtigen String antwortet oder eine bestimmte Zeit abgelaufen ist. Konnte keine Kamera gefunden werden, bricht das Programm mit einer entsprechenden Fehlermeldung ab. Wurde die Kamera gefunden, wird das Hauptprogramm in seinen Betriebszustand versetzt.

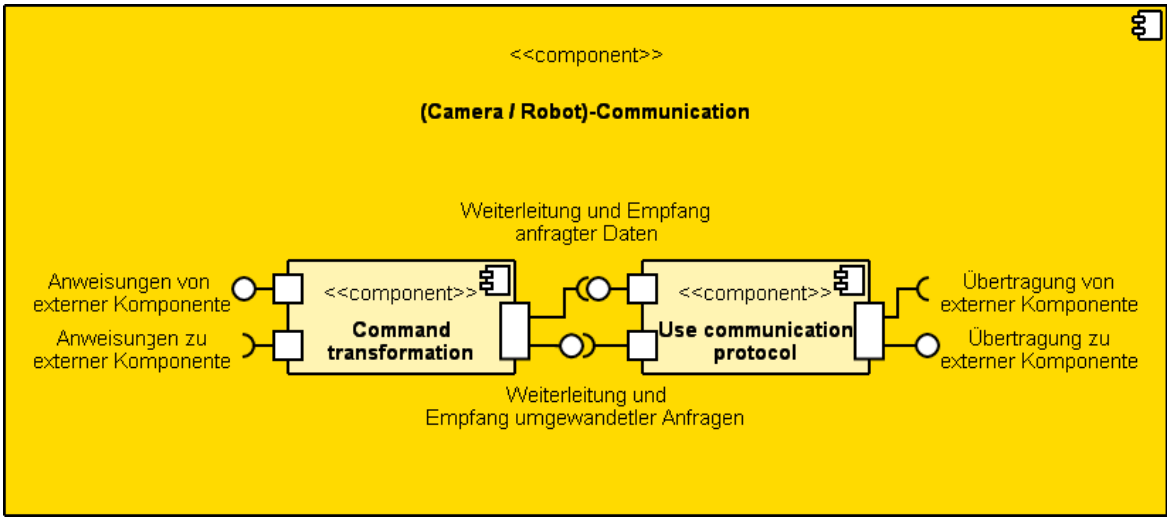

Abbildung 5.12: Aufbau der Komponenten für die Kommunikation

Abbildung 5.13 zeigt beispielhaft eine Kontaktaufnahme zwischen dem Hauptprogramm der Kameralogik und einer möglichen Platinenkamera. In dem Beispiel versucht die Kameralogik in Form des Hauptprogrammes die Kamera zu erreichen. Hierfür schickt es Reset Befehle in gleichmäßigen Abständen über die zur Verfügung gestellten Kommunikationswege der Camera Comunication. Andere Kameras können hier unterschiedliche Befehle erwarten. Zum Zeitpunkt des ersten Reset zur Kamera, war diese noch nicht bereit. Darum kommt an dieser Stelle keine Antwort. Das Gleiche passiert beim zweiten Versuch. Beim dritten Versuch ist die Platinenkamera empfangsbereit, allerdings ist bei der Übertragung ein Fehler passiert, weshalb eine unbekannte Zeichenkette die Kamera erreicht. Diese Zeichenkette wird mit einem „no acknowledge“ quittiert. Der vierte Versuch ist in diesem Beispiel schließlich erfolgreich. Die Kamera übermittelt ihre Kennung, in diesem Beispiel ein einfaches „hello“. Damit weiß das Hauptprogramm, dass die Platinenkamera betriebsbereit ist. Jetzt kann die Farbe des zu suchenden Objektes eingelesen werden, wie es in Kapitel 5.5.2 beschrieben ist.

Die Kommunikation während der Objektverfolgung zeigt Abbildung 5.14. Die zentrale Komponente ist hierbei der ObjektTracker. Er wurde vom Hauptprogramm gestartet. Beim Start hat er eine Farbe zum Verfolgen übermittelt bekommen. Als Erstes startet er die Objektverfolgung mit der Anweisung „TrackColor“. Daraufhin beginnt die Kamera „T-Pakete“ zu senden. Sie enthalten unter anderem die Koordinaten des gesuchten Farbschwerpunktes. Wie in dem Diagramm zu erkennen schickt die Platinenkamera ohne Anfrage in regelmäßigen Abständen diese Informationen zur Komponente CameraComunication. Ebenfalls in regelmäßigen Abständen wird das aktuelle Paket vom ObjektTracker abgeholt. Veraltete Pakete werden

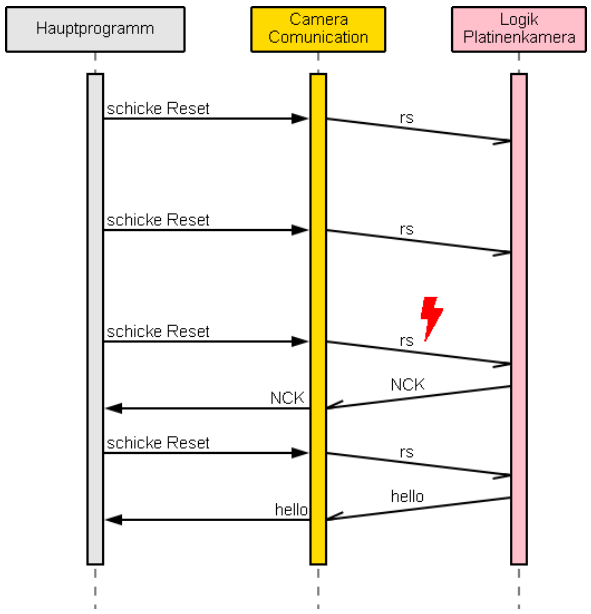

Abbildung 5.13: Beispiel einer Kontaktaufnahme der Kameralogik mit einer Platinenkamera

verworfen. Der ObjektTracker wandelt die frischen Koordinaten wie in Kapitel 5.5.1 gezeigt in eine Richtungsinformation um und hält diese für etwaige Abrufe bereit.

Die Abrufe kommen von der Robotsteuerung. Abbildung 5.14 zeigt zwei Versuche einen aktuellen Winkel zu erhalten. Die Anforderung geht über die Robot Comunication zum ObjektTracker. Im ersten Fall hat der ObjektTracker noch keine Daten und gibt einen leeren Winkel zurück. Die Robotsteuerung kümmert sich zunächst um weitere Aufgaben, um schließlich erneut Daten von der Kamera anzufordern. Dieses Mal liegt ein Winkel vor, welcher der Robotsteuerung zurückgegeben wird.

Nachdem die Komponenten der Kamera, ihr Aufbau und die Kameralogik fest stehen, wird es Zeit für eine Betrachtung dessen, was die Kamera leisten kann.

5.6 Fazit: Was kann die Kamera

Die hier beispielhaft erstellte omnidirektionale Kamera ist in der Lage Objekte mit einer Geschwindigkeit von acht bis 24 Bildern pro Sekunde zu verfolgen. Die Kameralogik erlaubt es die Kamera als Sensor für einen mobilen Roboter zu verwenden. Sie übermittelt auf Anfrage den Winkel zu einem Objekt, welches verfolgt wird. Der Aufbau der Kameralogik erlaubt es Komponenten der Kamera durch andere zu ersetzen, ohne die gesamte Software neu schreiben zu müssen. Die Beispielkamera wird über den CAN-Bus oder eine serielle

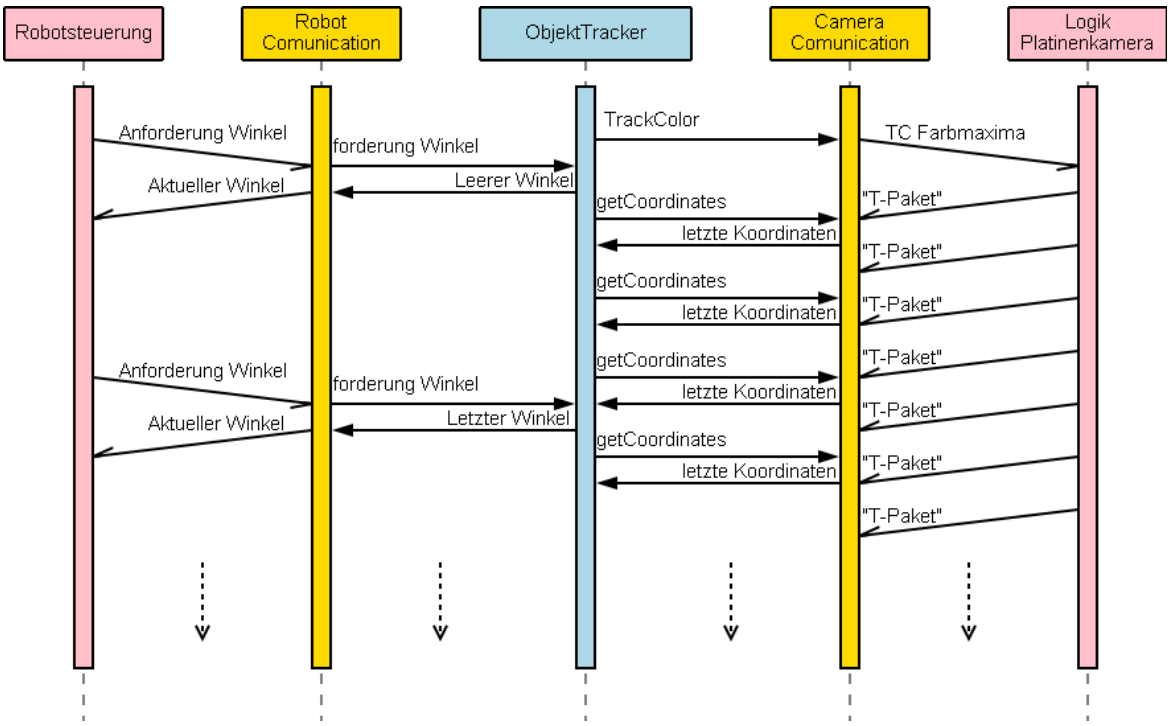

Abbildung 5.14: Normaler Betrieb der Kameralogik

Schnittstelle an einen Roboter angeschlossen. Objekte sind je nach Größe aus einer Kamerahöhe von 195 Millimetern, bis zu einer maximalen Entfernung von 1,8 Metern, erkennbar. Das entspricht der vorher durchgeführten Simulation und dem Blick über das im Robotlabor eingesetzte Spielfeld aus Kapitel 4.3.1. Mit diesem positiven Ergebnis kann die Kamera in einen Roboter integriert werden.

6 Integration der Kamera in einen Roboter

In Kapitel 5 ist eine Kamera für omnidirektionale Roboter entstanden. Diese soll sich in diesem letzten Kapitel der Arbeit beweisen. Dafür wurde ein bestehender Roboter angepasst und eine Beispielanwendung geschrieben. Diese Schritte werden in diesem Kapitel gezeigt.

6.1 Ausgangspunkt

Als Grundlage des in dieser Arbeit verwendeten Roboters dient die omnidirektionale Roboterplattform aus der Diplomarbeit von Michael Ziener Ziener (2005). Diese Plattform ist sehr stabil und bringt mit seinem „RCUBE“ ein für die Bildbearbeitung ausreichendes Mikrocontroller Modul mit. Angetrieben wird der Roboter über drei Motoren mit Planetengetriebe. Diese übertragen ihre Kraft auf die Antriebsräder über Zahnriemen. Der Roboter ist aus Aluminium gefertigt und bewegt sich auf Allseitenrädern mit Querrollen aus hartem Kunststoff.

6.2 Veränderungen an bestehender Plattform

An dieser Ausgangsplattform mussten einige Änderungen vorgenommen werden. Diese betreffen das Antriebssystem, die Spannungsversorgung und zusätzliche Aufbauten, welche durch den Einsatz der omnidriektonalen Kamera nötig geworden sind.

Durch die starke Belastung in verschiedenen Forschungsprojekten sind die Räder der Plattform mit der Zeit ausgeschlagen und haben Spiel entwickelt. Zusätzlich haben sich die Zahnriemen ausgedehnt und konnten nicht mehr genügend gespannt werden. Punkt vier in Abbildung 6.1 zeigt die neuen Räder, die angeschafft wurden, um das Spiel der Räder zu korrigieren. Die Entscheidung fiel auf mehrgliedrige Räder, welche derzeit im Hobbybereich eingesetzt werden. Diese Räder gibt es in der benötigten Größe von 80mm nur mit harten Querrollen und haben eine runde Nabe für Wellen mit einem Durchmesser von 8mm. Dieses macht eine Mitnahme des Rades auf der Welle notwendig.

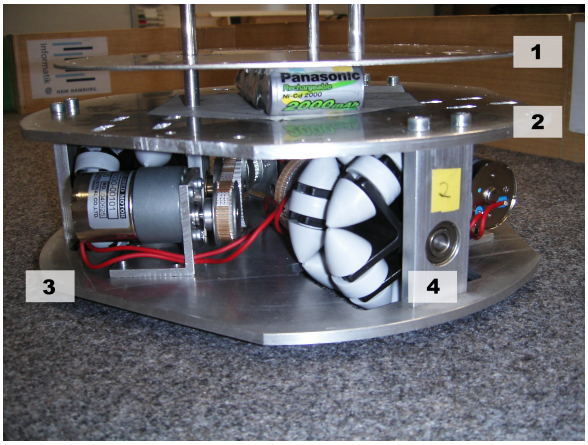

Abbildung 6.1: Veränderungen an Roboterplattform

Die Zahnriemen wurden durch Kürzere ersetzt, da diese bereits vorhanden waren. Die kürzeren Zahnriemen machten es notwendig die Motoren auf die Bodenplatte der Antriebseinheit zu verlegen (Punkt 3 Abbildung 6.1). Zusätzlich wurde mit der veränderten Position der Motoren der nutzbare Platz auf der oberen Ebene der Antriebseinheit, die in Abbildung 6.1 mit einer Zwei markiert ist, vergrößert. Der zusätzliche Platz entsteht durch den Wegfall der Schraubenköpfe, die vorher für die Halterung der Motoren nötig waren. Weiterhin entstand mit dem Umbau die Möglichkeit die obere Ebene der Antriebseinheit zu entfernen, um an die Mechanik des Antriebes zu gelangen. Damit kann der Antrieb im sicheren Stand gewartet werden, ohne den Roboter auf den Kopf stellen zu müssen. Das ist besonders praktisch, da die oberste Plattform nur selten eine Fläche bietet, auf der man den Roboter stellen könnte. Hinzu kommen lose Teile für Versuchszwecke, die sich oft auf dem Roboter befinden. Damit ist ein Umdrehen des Roboters für die Wartung des Antriebes ungünstig.

Die Spannungsversorgung ist überarbeitet worden. Die aus alten Bohrmaschinen Akkus gefertigten Akkupacks mit NiCd Akkus, wurden durch zwei sehr leistungsfähige neue Akkupacks aus dem Modellbau ersetzt. Diese bieten eine deutlich höhere Kapazität bei verringertem Platzbedarf. Es gibt zwei getrennte Spannungen, die von zwei unterschiedlichen Akkupacks geliefert werden. Ein zwölf-Volt-Akkupack ist für die Versorgung der Motoren verantwortlich. Dieser hat eine besonders hohe Kapazität. Für die Elektronik gibt es einen weiteren Akkupack mit sechs Volt. Dieses bietet verschiedene Vorteile. Zum Einen können Störungen durch die Motoren nicht auf die Elektronik überspringen. Zum Anderen ist eine ausreichende Versorgung der Sensoren und Mikrocontroller auch dann noch gewährleistet, wenn der Akku für den Antrieb schwach wird. Das hilft besonders den Sensoren, die auf Spannungsschwankungen empfindlich reagieren. Zusätzlich ist es möglich, die Spannung des Motorakkus vom Mikrocontroller überprüfen zu lassen und darauf zu reagieren.

Auf den speziellen Motortreiber der Originalplattform wurde verzichtet. Als Ersatz kommt

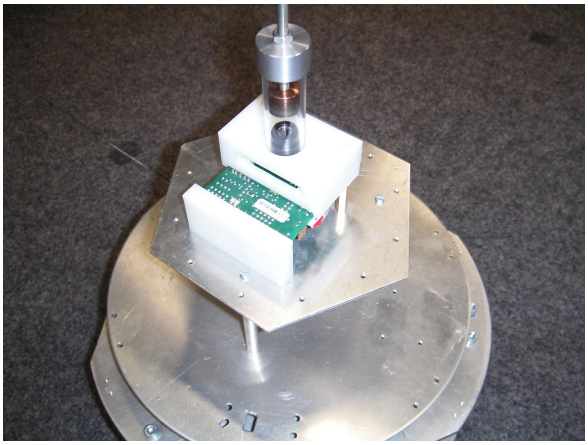

Abbildung 6.2: Kameraebene mit omnidirektionaler Kamera

eine Erweiterung des AKSEN-Boards zum Einsatz. Das AKSEN-Board wurde in Kapitel 5.4 beschrieben. Diese Erweiterung macht es sich zu nutzen, dass die auf dem Motortreiber des AKSEN-Boardes verbauten Motortreiber für eine weit höhere Spannung ausgelegt sind, als die vom AKSEN-Board vorgesehenen fünf bis acht Volt. Die Erweiterung trennt den Pin für die Motorspannung der Motortreiber vom AKSEN-Board und fügt dem Board einen weiteren Spannungskreis für die Motoren hinzu. Genaueres findet sich in den Anlage A. Das System hat sich inzwischen im Robotlabor bewährt. Zu Problemen mit der Erweiterung ist es bisher nicht gekommen.

Es macht Sinn omnidirektionale Kameras in der Mitte von Robotern zu platzieren. Dadurch ist es einfacher Richtungsinformationen zu verarbeiten. Weiterhin sollte die Effektivität der Sharp Entfernungssensoren erhöht werden, indem diese näher zum Zentrum des Roboters angebracht werden. Diese Sensoren arbeiten in den ersten Zentimetern nicht zuverlässig, weshalb es Sinn macht sie dem Mittelpunkt des Roboters näher zu bringen. Diese zwei Anforderungen wurden durch die in Abbildung 6.2 gezeigte Ebene gelöst. Sie bietet neben der Kamera Platz für die Sensoren. Da diese Ebene sehr hoch liegt, wurde die in Abbildung 6.1 mit Eins markierte alte Ebene wiederverwendet. Auf ihrer Unterseite findet sie Platz für zusätzliche sechs Entfernungssensoren. Diese können dort ebenfalls näher zum Zentrum des Roboters gebracht werden.

6.3 Integration der Kamera

Für die Integration der Kamera wurde die in Bild 6.2 gezeigte, stabile Halterung entworfen. Diese besteht aus zwei Messing Distanzstücken und einem Kunststoffteil, welche fest mit der sechseckigen, obersten Ebene verschraubt sind. Für den RCUBE musste ein neuer

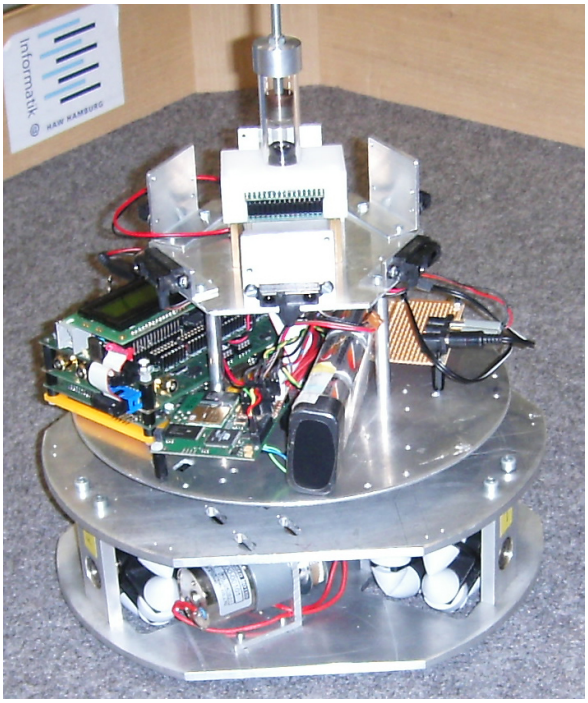

Abbildung 6.3: Plattform mit Elektronik

Platz gefunden werden. Die ursprüngliche Position am Rand der Plattform[1] reichte zu weit in den Mittelpunkt der obersten Ebene. Dadurch hätte die Kamera sehr hoch sitzen müssen. Aus diesem Grund wurde die Halterung für den RCUBE entfernt. Um Höhe zu sparen, ist die ursprünglich oberste Ebene deutlich tiefer gelegt worden. Dazu sind die Sharp Distanzsensoren von der oberen auf die untere Seite der Ebene verlegt worden.

6.4 Fazit: Was kann der Roboter

Der Roboter wie ihn Abbildung 6.3 zeigt, stellt eine stabile Plattform für weitere Arbeiten dar.

Der omnidirektionale Antrieb erlaubt es dem Roboter in alle Richtungen zu fahren, ohne dafür drehen zu müssen. Der Roboter ist in der Lage sich während der Fahrt zu einem Objekt auszurichten und kann bei Bedarf im Stand drehen. Diese Roboterplattform hat drei bevorzugte Richtungen, geradeaus, 120 Grad links und 120 Grad rechts.

Die im Rahmen dieser Arbeit entstandene omnidirektionale Kamera, ermöglicht es dem Roboter ein Objekt in einem sehr großen Bereich zu verfolgen. Dieser Bereich entspricht in etwa

[1] Siehe Bild in Arbeit von Michael Ziener (Ziener, 2005, S. 47)

dem Fußballfeld für Robotfußball an der HAW-Hamburg, wie es in Kapitel 4.3.1 beschrieben ist. Des Weiteren verfügt er über zwei Mal sechs Infrarot Distanzsensoren, mit deren Hilfe Wände und Hindernisse erkannt werden können. Sie sind in gleichmäßigen Abständen um den Roboter verteilt. Es ist jedoch nicht vorgesehen diese zwölf Sensoren gleichzeitig zu verwenden. Sie sind in zwei identischen Anordnungen in unterschiedlichen Höhen angebracht. Das erlaubt es die für die Umgebung jeweils geeignetere Höhe zu wählen.

Durch die Erneuerung des Fahrwerks und der Spannungsversorgung ist der Roboter für lange Einsätze vorbereitet. Autonomes Fahren über mehrere Stunden sollte ohne Probleme möglich sein. Das nächste Kapitel beschäftigt sich mit der Ansteuerung des Roboters und der Software der omnidirektionalen Kamera.

6.5 Testumgebung

6.5.1 Einleitung

Für diese Arbeit ist neben der Software für die Simulation, welche im Kapitel 3 vorgestellt wurde und der Kameralogik aus Kapitel 5.5, Software für den Betrieb des im Kapitel 6 beschriebenen Roboter entwickelt worden. Diese Software dient als Testumgebung für die omnidirektionale Kamera. Hier wurde Wert auf eine möglichst einfache Wiederverwendbarkeit gelegt. Die Software teilt sich in zwei Bereiche auf. Der erste Bereich ist ein einfaches Überwachungsprogramm. Dieses befindet sich auf einem PC und erhält Informationen über die Position gefundener Objekte von der Kameralogik. Das zweite Programm ist für die Fahrmanöver des Roboters zuständig und nutzt die Informationen der Kamera, um einen Ball zu verfolgen.

Die Kommunikation zwischen der Kameralogik und der Roboterlogik findet über den CAN-Bus statt, welcher das LART-Board mit dem AKSEN-Board verbindet. Das Überwachungsprogramm kann über die Serielle Schnittstelle oder Bluetooth Informationen von der Kameralogik erhalten. Das ermöglicht eine Kontrolle der Kameralogik, was für die Entwicklung hilfreich ist.

Im Folgenden wird auf die Besonderheiten und den Aufbau der Roboterlogik und des Überwachungsprogrammes eingegangen.

6.5.2 Überwachungsprogramm

Bei dem Überwachungsprogramm handelt es sich um ein sehr rudimentär zusammengesetztes Javaprogramm. Die Oberfläche zeigt zwei Fenster. Die Fenster beinhalten ein „Radar“ und einen Mittschnitt der Informationen von der Kameralogik. Das Radar zeigt mit einem Zeiger die Richtung an, wo die omnidirektionale Kamera ein Objekt erkennt. Das erleichtert und beschleunigt die Funktionsprüfung einer neuen Kamera und ermöglicht das gleichzeitige Bewegen eines Objektes in einem Szenario sowie die Überprüfung der Reaktion der Kamera. Im Logfenster können beliebige Strings ausgegeben werden, die über die serielle Schnittstelle den PC erreichen. Hier können bestimmte Punkte in der Kameralogik markiert und Daten zur Auswertung übermittelt werden. Das erleichtert eine spätere Fehlersuche.

Das Überwachungsprogramm stellt somit ein sinnvolles Werkzeug im Rahmen eines Systems zur Entwicklung omnidirektionaler Kameras für mobile Roboter dar.

6.5.3 Roboterlogik

Neben der Überwachung der Funktionen der Kamera aus Kapitel 6.5.2, soll anhand eines einfachen Beispiels gezeigt werden, dass die Kamera als Sensor auf einem Roboter funktioniert. Das macht eine Roboterlogik notwendig.

Bei der Roboterlogik handelt es sich um eine einfache Subsumption Architektur[2]. Das bedeutet, dass der Roboter keinem vorgefertigten Verhalten folgt, sondern direkt auf seine Sensordaten reagiert. Hierbei wird auf unterschiedliche Kombinationen von Sensorwerten mit unterschiedlicher Priorität reagiert. Das sorgt primär dafür, dass der Roboter sein Spielfeld nicht verlässt. Ist das sicher gestellt, wird der Roboter in Richtung des Balles bewegt.

Für die Ansteuerung der Motoren und die Bewegung in eine bestimmte Richtung soll auf eine Bibliothek von Michael Ziener zurückgegriffen werden. Sie ist im Rahmen seiner Diplomarbeit „Konstruktion einer programmierbaren omnidirektionalen Roboterplattform“ Ziener (2005) entstanden und soll die Motoren des Roboters so ansteuern, dass er sich in einem vorgegebenen Winkel mit einer bestimmten Geschwindigkeit bewegt. Dieser Bibliothek werden die Informationen der omnidirektionalen Kamera übergeben. Sie liefert die Richtungsinformationen in Grad, die genau für diese Funktionen benötigt werden.

Alternativ kann eine einfachere Ansteuerung programmiert werden. Hierbei würde der von der Kamera stammende Winkel einem Sektor zugewiesen. Der Roboter würde sich mithilfe seiner drei Hauptrichtungen dem Ball annähern. Das ist sehr einfach zu implementieren und zeigt die Funktionsfähigkeit der Kamera ausreichend gut.

6.5.4 Fazit: Was kann die Testumgebung

Die Testumgebung stellt eine sehr einfache Beispielimplementierung dar. Sie reicht aus, um die Funktionsfähigkeit der Kamera und die Kommunikation zwischen den Komponenten zu zeigen. Damit zeigt die Software, dass die entstandene omnidirektionale Kamera als Sensor funktioniert.

Das zusätzliche Überwachungsprogramm bietet einen schnellen Zugriff auf die Kamera und erlaubt es die Funktionsfähigkeit der implementierten Funktionen der in Kapitel 5.5 beschriebenen Bibliothek zu überprüfen.

Das Überwachungsprogramm ist bei Abgabe dieser Arbeit in rudimentärer Form fertig. Es kann allerdings Sinn machen hier neu anzusetzen und eine Software zu schreiben, welche

[2] Für nähere Informationen zur Subsumption Architektur siehe Paper von R. Brooks „A robust layered control system for a mobile robot“ Brooks (1968)

die Funktionen von Autonomen Robotern überwacht. Besonders ein auf diese Weise implementierter Notausschalter wäre neben grundlegenden Kontrollfeldern eine interessante Hilfe beim Programmieren von autonomen Robotern.

Die Roboterlogik ist aus Zeitgründen noch nicht fertig gestellt. Bisher existieren alle Schnittstellen für die Kommunikation mit der Kamera. Der nächste Schritt zu einem voll funktionsfähigen Roboter, der die in dieser Arbeit gefertigte omnidirektionale Kamera als Sensor nutzt, ist daher nicht allzu groß.

7 Zusammenfassung

7.1 Zusammenfassung

Ziel dieser Arbeit war es einen Weg zu bereiten, auf dem in kurzer Zeit omnidirektionale Kameras für den Einsatz in Robotlaboren geschaffen werden können. Diese sollten als Sensor genutzt werden können, um eine Richtungsinformation zu einem einmaligen, farbigen Objekt zu bekommen. Hinzu kommt der Aspekt der Kosten, da fertige omnidirektionale Kameras preislich extrem hoch angesiedelt sind. Dieses Ziel ist mit der vorliegenden Arbeit erreicht worden.

Es ist eine Simulationssoftware entstanden, die zuverlässig als „Push-Button-System“ vier verschiedene Spiegelarten simulieren kann und anschließend zu jedem simulierten Spiegel alle notwendigen Daten für die Produktion des Spiegels zur Verfügung stellt. Das Programm bietet eine sehr geringe Lernkurve, die zu ersten Ergebnissen führt und einen ersten Eindruck über das Bild, welches die Kamera später liefern wird, vermittelt. Reichen die vorgegebenen Werte nicht, ist die Software an eigene Gegebenheiten anpassbar. Nach einer mittleren Abstraktionsschicht gibt es im letzten Schritt die Möglichkeit Spiegelformen, die bisher nicht vorgesehen waren zu simulieren. Das Simulationsprogramm baut auf Standardmechanismen des Raytracing Programmes POV-Ray auf. Das macht es möglich vorher erzeugte Resultate mit etwas Einarbeitung selber zu manipulieren und anzupassen. Zusätzlich erlaubt ein modularer Aufbau das Erweitern der Simulationssoftware.

Neben der Simulationssoftware ist eine Kameralogik entstanden. Sie enthält eine Bibliothek für verschiedene Algorithmen zur Aufbereitung von einer Kamera stammender Daten. Hinzu kommen austauschbare Funktionen, welche die eben angesprochenen Algorithmen von den Kamerafunktionen und den Kommunikationswegen trennen. Damit ist die Software sehr portabel gehalten und kann für beliebige Plattformen umgebaut werden, solange es für diese C-Compiler gibt.

Zusätzlich zu diesen für die Entwicklung omnidirektionaler Kameras wichtigen Komponenten wurde ein bestehender Roboter so umgebaut, dass er als Versuchsplattform genutzt werden kann. Auf ihm befindet sich ein erster Prototyp einer mit diesem System gebauten omnidirektionalen Kamera. Neben dieser Kamera ist ein sehr stabiles Versuchsgestell entstanden, welches sehr gute Einstellungsmöglichkeiten bietet und vielseitig einsetzbar ist. Dieses

Gestell erlaubt es zum Beispiel Spiegel zu testen, bevor ein angepasstes Gestell gefertigt wurde.

Zum Testen einer fertigen omnidirektionalen Kamera sind ein Überwachungsprogramm und eine Roboterplattform entstanden. Das sehr rudimentäre und einfache ÜberwachungsProgramm ermöglicht es die erstellte omnidirektionale Kamera zu testen. Es dient als Mensch-Maschine-Schnittstelle und erzeugt aus dem Datenstrom einer (Platinen-) Kamera ein leicht auswertbares Bild. Als Ergänzung dazu stellt der Roboter ein fertiges Framework, in das in kurzer Zeit ein neues Verhalten integriert werden kann.

Abgerundet wird die Arbeit durch eine LiveCD auf Linuxbasis, auf der die komplette Software inklusive Programmierumgebungen installiert ist. Diese CD ist bootfähig und enthält ein komplettes Betriebssystem. Von hier aus können die Simulationsumgebung und das Überwachungsprogramm einfach gestartet werden. Zusätzlich enthält die CD die wichtigsten Programme zur Entwicklung und Dokumentation eigener Software, soweit dieses rechtlich möglich war. Damit kann sofort mit der Arbeit oder einem einfachen Blick in die Welt der omnidirektionalen Kameras begonnen werden, ohne langwierig mit der Installation eines Arbeitsplatzes aufgehalten zu werden.

7.2 Fazit

Mit dieser Arbeit konnte ein System entwickelt werden, dass omnidirektionale Kameras in jedes Robotlabor bringen kann. Die Baukosten beschränken sich auf ein Stück Kupfer, etwas Material für eine Kamerahalterung und eine Platinenkamera. Zudem bietet das System eine Lösung, die schnell zu einem Ergebnis kommt. Sie ist unabhängig von Drittherstellern und ermöglicht es damit jederzeit eine baugleiche Kamera zu einem bestehenden Pool hinzuzufügen.

Die im Rahmen der Arbeit erstellte Software bietet eine gute Grundlage für die Logik der Kamera und macht neue Kameras nach geringen Anpassungen der Software schnell einsetzbar.

Was nicht optimal funktioniert hat, war der erste nach den Daten der Simulation gebaute Spiegel. Dieser zeigt zwar genau den richtigen Bereich und passt gut zum Simulationsergebnis, allerdings ist das Bild unscharf. Hier fehlte die Information über den Schärfebereich der Kamera bei der Spiegelgröße und der besonderen Höhe über der Linse. Aus Zeitgründen konnte keine Halterung für einen zweiten, größeren Spiegel entstehen. Diese Information fehlt noch in der Simulationssoftware und sollte hinzugefügt werden. Die dazu nötigen Formeln finden sich in dieser Arbeit. Das Berücksichtigen des Schärfebereiches bei der Berechnung der Spiegelhöhe sollte kein Problem sein.

7.3 Ausblick

Als Ergänzung zu dieser Arbeit gibt es verschiedene Ansatzpunkte in ganz unterschiedlichen Gebieten.

Die derzeitige Lösung baut auf einem fertigen Mikrocontollermodul, dem R-CUBE der FH-Brandenburg auf. Dieser ist zum einen kostenintensiv, bietet Funktionen, die nicht benötigt werden, und ist im Moment nicht käuflich. Zudem ist er sehr groß. Hier wäre ein guter Ansatzpunkt ein eigenes Steuermodul auf Basis eines 32-Bit-Mikrocontrollers zu schaffen. Selbst preiswerte Mikrocontroller dieser Art bieten verschiedene Schnittstellen und können die von der Platinenkamera stammenden Informationen weiterverarbeiten. Damit entsteht wieder eine Unabhängigkeit von externen Dienstleistern und Zulieferern. Auch für den Einsatz der omnidirektionalen Kamera wird ein anderes Steuermodul für die Kamera benötigt, welches auch auf kleinere Roboter passt.

Ein weiterer Ort für Verbesserungen ist die eingesetzte Platinenkamera. Diese ist sehr leistungsstark, bietet sehr gut dokumentierte Schnittstellen und verfügt über eine sehr gute Testumgebung. Allerdings reduziert diese Einfachheit der Kamera die Fähigkeiten der omnidirektionalen Kamera. Es ist zum Beispiel nur umständlich möglich zwei Objekte unterschiedlicher Farbe gleichzeitig zu verfolgen, das könnte in einer eigenen Implementierung deutlich effektiver gestaltet werden. Hier ist es eine Überlegung wert, ob es Sinn macht eigene Software zur Bildanalyse zu schaffen, die auf die Bedürfnisse einer omnidirektionalen Kamera ausgelegt ist. Zusätzlich hat sich die Verfügbarkeit der Kamera als sehr unsicher herausgestellt, wodurch ein Umstieg auf eine andere Kamera möglicherweise notwendig wird. Einen Einstiegspunkt könnte hierfür die Diplomarbeit von Lars Brandt bieten Brandt (2005). Dort wurde ein Objekttracker für Embedded Systems zur Steuerung mobiler Roboter geschaffen. Als Grundlage diente auch dort der R-Cube, was einen schnellen Einstieg erlauben würde.

Im Bereich der Simulationssoftware ist es möglich weitere Spiegelformen und Objekte hinzuzufügen. Interessant wäre auch eine alternative, intuitivere Eingabemethode. Schön wäre etwas, wo Grundformen als „Wirenet“ oder bereits gerenderte Form dargestellt würden, an denen rumgezupft werden kann. Das würde das Verstehen des Bildes fördern und viel flexiblere und ausgefallenere Spiegel erlauben. Zusätzlich könnte die grafische Benutzerschnittstelle um eine bessere Anzeige der Simulationsergebnisse erweitert werden.

Die Roboterplattform könnte in verschiedene Richtungen erweitert werden. Die im Moment verbauten Räder haben Querrollen aus hartem Kunststoff. Das führt dazu, dass sie auf dem Boden der Labore nur sehr wenig Haftung aufbauen können. Hier könnte versucht werden die Querrollen aus einem anderen Material herstellen oder andere Räder zu entwickeln, die eine bessere Haftung bieten. Eine weitere, sinnvolle Erweiterung wäre der Ausbau der omnidirektionalen Kamera zu einem kombinierten Sensor. Hierfür sind bereits Halterungen für Ultraschall Sensoren an der sechseckigen Kameraplattform gefertigt worden. Ultraschall

Sensoren erlauben es dem Roboter Hindernisse auf größere Entfernungen zu erkennen. Dadurch könnte er sich zum Beispiel sicherer im Flur der Hochschule bewegen. Die Kamera könnte so für die Erkennung der Türen verwendet werden. Was der Plattform gänzlich fehlt, ist ein Mechanismus, welcher den Füllstand der Akkus überprüft und bei schwächer werdender Spannung das Verhalten ändert und eine Rückmeldung liefert. Das schützt vor Tiefentladung des Akkus und kann vermeintlichen Fehlern vorbeugen, die durch schwache Akkus hervorgerufen werden.

Interessant könnte eine spezielle omnidirektionale Kamera sein, welche auf Schiffen oder Rampen funktioniert. Hierfür müsste die Kamera flexibel gelagert sein und sich selbstständig zum Horizont ausrichten können. Zusätzlich müsste sie arretierbar sein, da es auf Rampen auch wichtig sein kann nach oben zu sehen.

Ich denke diese Arbeit eröffnet viele neue Möglichkeiten zur Weiterentwicklung und öffnet die Tür für interessante Projekte. Ich hoffe diese Arbeit erschließt die Technologie der omnidirektionalen Kamera einem weit größerem Anwenderkreis und nimmt ihr den Schrecken.

Literaturverzeichnis

[Brandt 2005] BRANDT, Lars: *Entwicklung eines Objekttrackers für Embedded Systems*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2005. – URL http://users.informatik.haw-hamburg.de/~kvl/brandt/diplom.pdf

[Brooks 1968] BROOKS, R.: *A robust layered control system for a mobile robot*. Robotics and Automation, IEEE Journal of [legacy, pre - 1988], 1968 ISSN 0882-4967

[Christopher D. Watkins 1993] CHRISTOPHER D. WATKINS, Mark F.: *Fotorealismus und Ray Tracing in C*. Verlag Heinz Heise GmbH u. Co KG Hannover, 1993. – ISBN 3-88229-024-2

[Dia] ALEXANDER LARSSON, Lars Clausen Hans B.: *Dia*. – URL http://www.gnome.org/projects/dia/

[Gaida 1983] GAIDA, Bernhard: *Galvanotechnik in Frage und Antwort für Selbststudium und Schulgebrauch*. 1983. – ISBN 387470-011-3

[Galvanisieren] UNBEKANNT: *Arbeitsanleitung Galvanisieren*. Lag Galvanisierset von Conrad mit Artikel Nr. 527983-62 bei.

[GeoGebra] MARKUS HOHENWARTER, Yves K.: *GeoGebra*. – URL http://www.geogebra.at

[golem] PSEUDONYM nie: *Raytracing fit für Computerspiele?*. – URL http://www.golem.de/0702/50770.html

[Heinz Leuze 1984] HEINZ LEUZE, et a.: *Praktische Galvanotechnik Ein Lehr- und Handbuch*. 1984. – ISBN 3-87480-017-2

[ImageMagick] IMAGEMAGICK: *ImageMagick*. – URL http://www.imagemagick.org

[Kameramodul-Kamera] LTD, COMedia: *Datenblatt Kameramodul C-CamA*. – URL http://www.comedia.com.hk

[Kameramodul-Sensor] OMNIVISION: *Datenblatt Kameramodul OV6620*. – URL http://www.openrt.de/

[Kunz] KUNZ, Thomas B.: *Digital Panorama Technologie*. – URL http://www.tbk.de/frameset.htm?panorama/show4.htm

[Kunze] KUNZE, Tilo: *www.crosus.de, Panoramafotografie und mehr*. – URL http://www.crosus.de/

[Mangerr 2004] MANGERR, Michael: *Design und realisierung einer experimentellen Plattform für Roboterfußball*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2004. – URL http://users.informatik.haw-hamburg.de/~kvl/manger/diplom.pdf

[Microsystems] MICROSYSTEMS, Sun: *Sun Microsystems*. – URL http://www.sun.com/

[OpenRT] SAARLANDS, Universität des: *The OpenRT Real-Time Ray-Tracing Project*. – URL http://www.openrt.de/

[Papula 2001] PAPULA, Lothar: *Mathematische Formelsammlung für Ingenieure und Naturwissenschaftler*. Vieweg und Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 2001, 2001. – ISBN 3-528-64442-7

[Plenge 1989] PLENGE, Axel: *3-D-Grafik und Animation für IBM-PC und Kompatible: Grundlagen, perspektivische Projektion, 3-D-Funktionsplotter, Ray-Tracing, Rotationskörper und vieles mehr*. Haar bei München: Markt-u.-Technik-Verlag, 1989. – ISBN 3-89090-678-8

[Pohl 2006] POHL, Daniel: *Raytracing in Spielen Einblicke in ein alternatives Rendering-Verfahren*. 2006. – URL http://www.computerbase.de/artikel/software/2007/bericht_raytracing_spielen/

[RCUBE] BRANDENBURG, FH: *RCUBE*. – URL http://ots.fh-brandenburg.de/?module=pagemaster&PAGE_user_op=view_page&PAGE_id=8&MMN_position=17:14

[RoboCupJunior] ROBOCUPJUNIOR: *RoboCupJunior Official Site*. – URL http://satchmo.cs.columbia.edu/rcj/

[Steinbuch 2004] STEINBUCH, Prof. Dipl.-Math. R.: *Simulation im konstruktiven Maschinenbau*. Fachbuchverlag Leipzig im Carl Hanser Verlag, 2004. – ISBN 3-446-22747-4

[Vstone] VSTONE: *Omnidirektionales Kamera von Vstone*. – URL http://www.vstone.co.jp/

[Wikipedia-Raytracing 2007] SYNONYM verschiedene unter: *Raytracing*. Wikipedia. 2007. – URL http://de.wikipedia.org/wiki/Raytracing

[Wikipedia-Schaerfentiefe 2007] SYNONYM verschiedene unter: *Schärfentiefe*. Wikipedia. 2007. – URL http://de.wikipedia.org/wiki/Sch%C3%A4rfentiefe

[Ziener 2005] ZIENER, Michael: *Konstruktion einer programmierbaren, omnidirektionalen Roboterplattform*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2005. – URL http://users.informatik.haw-hamburg.de/~kvl/ziener/diplom.pdf

A Bauanleitung Erweiterung AKSEN-Board auf entkoppelte Motorspannung V1.12

A.1 Vorwort

Das AKSEN-Board der FH-Brandenburg verfügt über Schnittstellen für 4 Gleichstromgetriebemotoren mit einer Spannung von 6V und einer maximalen Stromaufnahme von 1A Dauerstrom. Für größere Roboter reichen diese Motoren nicht. Stärkere Motoren mit mehr Spannung sind hier die erste Wahl. Dieses Dokument zeigt einen Weg, um mit einem kleinen Aufsatz stärkere Motoren an das AKSEN-Board anschließen zu können. Mit Aufsatz dürfen Motoren verwendet werden, die eine Spannung zwischen 4,5V und 32V benötigen. Dabei darf der maximale Strom 1A nicht überschreiten.

A.2 Technische Details

A.2.1 Grundlegende Idee

Das AKSEN-Board verfügt über zwei getrennte Stromkreise. VCC2 läuft auf der Batteriespannung von bis zu 8V, VCC1 wird geregelt und liefert eine Spannung von 5V. Möchte man die Spannung z.B. auf 12V heben, so beeinträchtigt dies verschiedene Komponenten des AKSEN-Boards. Daraus Folgt die größere Dimensionierung verschiedener Widerstände. Zusätzlich sind die Servo-Ports nach einer Anhebung von VCC2 nicht mehr nutzbar. Mit dem Adapter wird dieses Problem umgangen. Die Motorspannung wird von VCC2 getrennt und bekommt eine eigene Versorgung VCC3.

A.2.2 Umbau

Da die Motortreiber des AKSEN-Boards in Sockel gesteckt sind, muss am AKSEN-Board selber nichts verändert werden. Benötigt wird lediglich eine Vorrichtung, um die Motorversorgungsspannung an die Motortreiber zu bekommen. Dieses übernimmt ein Adapter.

A.2.3 Benötigte Teile für zwei Adapter

Für den Umbau werden benötigt:

- 4 16-polige Präzision-IC-Fassung
- 2 rote Kabel
- 1 schwarzes Kabel
- 1 Stecker

A.3 Aufbau der Teilstücke des Adapters

Als Grundlage des Adapters dienen zwei 16-polige Präzision-IC-Fassungen. Diese haben die gleiche Größe, wie die IC-Fassungen für die Motortreiber des AKSEN-Boards. Sie werden im Folgenden durch Bauteil 1 und Bauteil 2 bezeichnet.

A.4 Bauteil 1

A.4.1 Was gemacht werden muss

Um die Motorspannung des AKSEN-Boards (VCC2) vom Motortreiber zu entkoppeln, wird aus einem der vorliegenden Sockel Pin8 (Siehe Abb. A.1) entfernt. Dieses ist dann Bauteil 1 und stellt die Verbindung zum AKSEN-Board da.

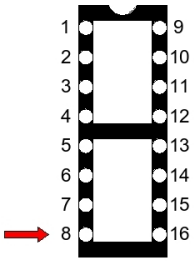

Abbildung A.1: Belegung Sockel, Pin8 (VCC) Markiert

empfohlene Vorgehensweise

Der Pin8, der zur Motorspannung auf dem AKSEN-Board geht (VCC2), wird mit einem kleinen Bohrer aus einem der beiden Fassungen getrennt. Somit gibt es keine Verbindung zwischen der neuen Motorspannung (VCC3) im Motortreiber und VCC2 des AKSEN-Boards.

Das Ergebnis sieht man in Abb. A.2

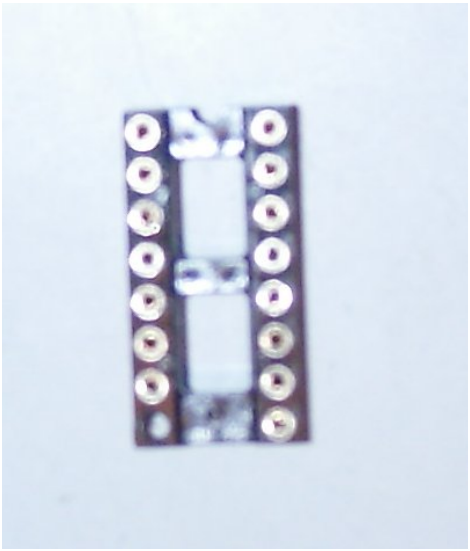

Abbildung A.2: Sockel mit entferntem Pin8

A.4.2 Bauteil 2

Was gemacht werden muss

Für die neue Motor-Versorgungsspannung (VCC3), wird an der zweiten Fassung ein Kabel an die breite Stelle des Pin8 gelötet. Gibt es keine breite Stelle, dann sollte das Kabel so weit wie möglich zum Sockel hin gelötet werden.

Die Lötstelle sollte nicht höher sein, als die Auflagefläche des Sockels (Abb. A.3). So wird ein passgenaues Zusammenstecken der Bauteile eins und zwei ermöglicht und die Gefahr von Wackelkontakten minimiert. Das übrige Stück des achten Pins wird abgeschnitten. (Abb. A.4)

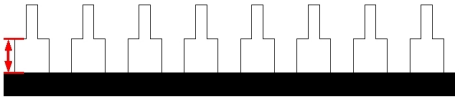

Abbildung A.3: Lötstelle nicht höher als Auflagefläche

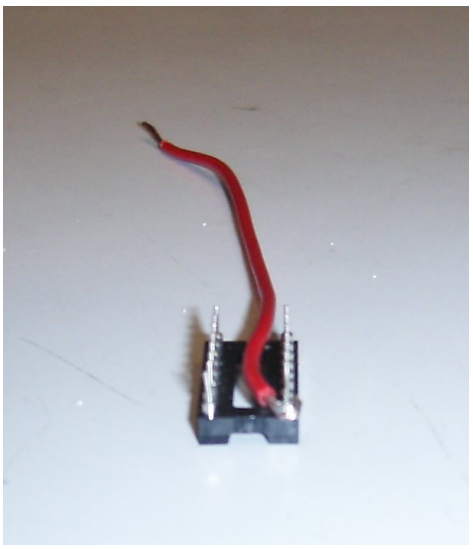

Abbildung A.4: Bauteil 2 mit angelötetem Spannungskabel

Empfohlene Vorgehensweise

Zum Anlöten des roten Kabels empfiehlt es sich, die zweite Fassung leicht in die erste zu stecken. Dabei sollte ein Pin der unteren Fassung in die Buchse des Pin 8 im Bauteil 2 gesteckt

werden. So kann sich Pin 8 durch die Erwärmung nicht so leicht aus seiner Richtung biegen. Das Ganze sollte man an dieser Stelle vorsichtig einspannen, um ein sicheres Arbeiten zu ermöglichen. Das rote Kabel sollte um ca. einen Zentimeter ab isoliert werden. Das Ende kann dann um Pin 8 gewickelt werden. (Abb. A.5)

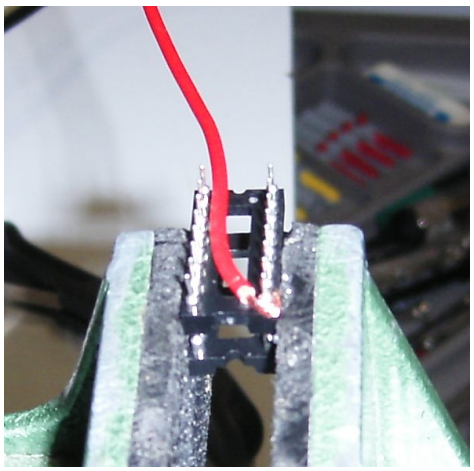

Abbildung A.5: Bauteil 2 eingespannt, Kabel fertig zum Löten

Ist das Kabel dran, wird der überstehende Rest des Pin 8 abgetrennt. (Abb. A.6)

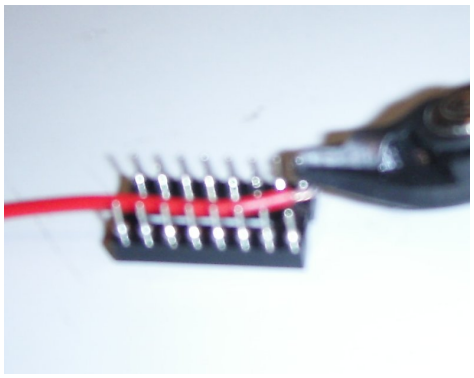

Abbildung A.6: überflüssige Rest des Pin 8 wird abgetrennt

Da der Pin sehr weit außen steht und hier eine recht große Spannung, sowie ein großer Strom anliegt, wird er mit Schrumpfschlauch isoliert. Dafür wird zunächst ein kleiner Schrumpfschlauch so weit wie möglich zur Lötstelle geschoben (Abb. A.7). Dort wird er geschrumpft.

Die äußere Seite wird mit großem Schrumpfschlauch isoliert. Dafür wird ein Stück Schrumpfschlauch über die Ecke gezogen. (Abb. A.8)

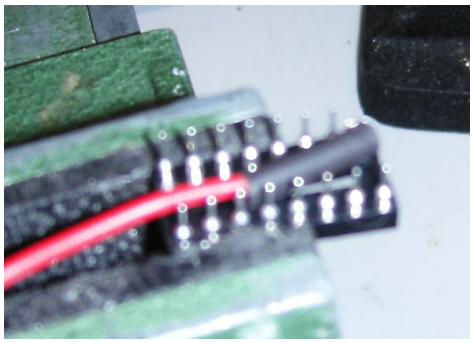

Abbildung A.7: kleiner Schrumpfschlauch bis Anschlag an Lötstelle geschoben

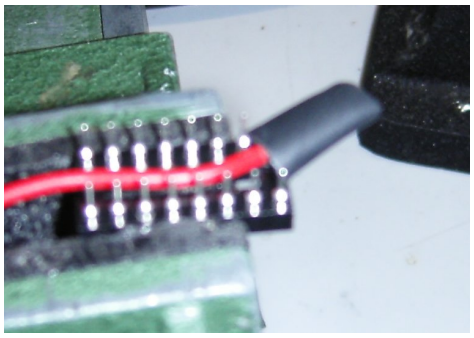

Abbildung A.8: großer Schrumpfschlauch über die Ecke gezogen

Dieser Schrumpfschlauch wird nun mit einem Heisluftföhn geschrumpft. So lange der Schrumpfschlauch noch weich ist, wird er dicht an der Kante der Fassung zusammengedrückt. Ein kurzes Stück dahinter wird der immer noch weiche Schrumpfschlauch abgeschnitten. Dadurch wird die Öffnung geschlossen. (Abb. A.9)

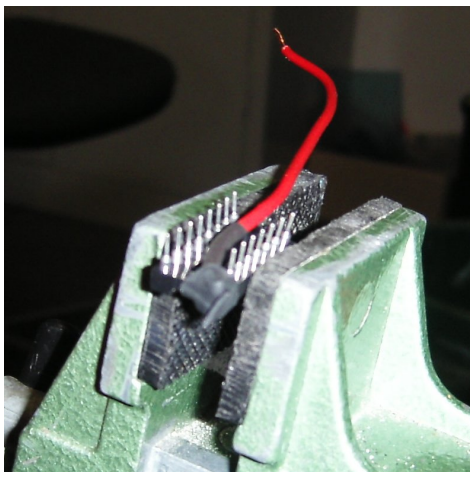

Abbildung A.9: Die Ecke des Adapters ist isoliert

Im nächsten Schritt muss die Buchse der Fassung wieder geöffnet werden, da sie vom Schrumpfschlauch überdeckt wird. Dazu mit einem scharfen Messer ein Stück der Oberseite ausschneiden. (Abb. A.10)

Abbildung A.10: Die Buchse wieder verfügbar gemacht für Motortreiber-IC

Nach Anbringen des Kabels für die Versorgungsspannung (VCC3) wird das Kabel für die Masse an die Pins 4 und 5 gelötet A.11.

Hierfür wird ca. 1cm Isolierung des blauen Kabels entfernt. Das Ende wird in zwei gleiche Bündel geteilt. Diese Bündel werden um die Pins gewickelt. (Abb. A.12) Hierbei muss wieder auf die Höhe geachtet werden. Die Kabel inklusive Lot müssen wie beim Kabel für die Versorgungsspannung innerhalb der Sockelauflagefläche bleiben.

An die Enden der Kabel sollte ein beliebiger verpolungssicherer Stecker gebaut werden.

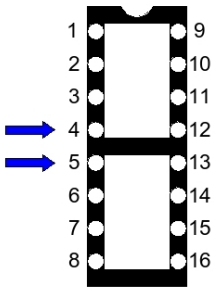

Abbildung A.11: Belegung Sockel, Masse Markiert

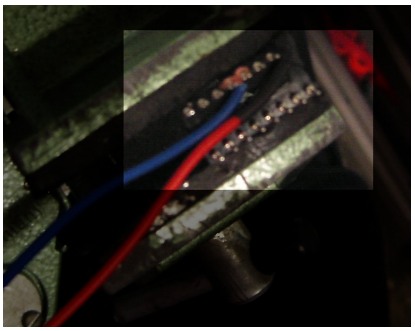

Abbildung A.12: Kabel für die Masse um Pin 4 und 5 gewickelt

Werden beide Motortreiber an der gleichen Spannungsquelle betrieben, müssen die roten Kabel beider Adapter verbunden werden. Das funktioniert am Stecker gut.

A.5 Zusammenbau

A.5.1 Was gemacht werden muss

Die fertigen Bauteile 1 und 2 können zusammen gesteckt werden. Die Kabel liegen im Freiraum zwischen den Fassungen. Die Isolierung der Kabel schützt vor Kurzschlüssen, die auftreten könnten, wenn die Kabel gegen die übrigen Beine der Fassungen kommen. Zum Zusammenstecken werden die fertigen Buchsenteile vorsichtig über einander gelegt und leicht an gedrückt. Hier empfiehlt es sich die Richtung der Bauteile zu überprüfen (Aussparung über Aussparung). Ist alles richtig, die Fassungen unter leichtem Druck zusammendrücken, bis sie richtig sitzen. sollte an einer Stelle eine größere Lücke entstanden sein, so besteht die Gefahr eines Wackelkontaktes. Um spätere Probleme zu vermeiden, sollte man in diesem Fall die entsprechende Lötstelle verkleinern.

A.6 empfohlene Vorgehensweise

Die Kabel verlaufen zwischen Bauteil 1 und Bauteil 2 und verlassen den Adapter über die Lücke zwischen Pin 1 und Pin 9 am Kopfende der Fassungen (Abb. A.13).

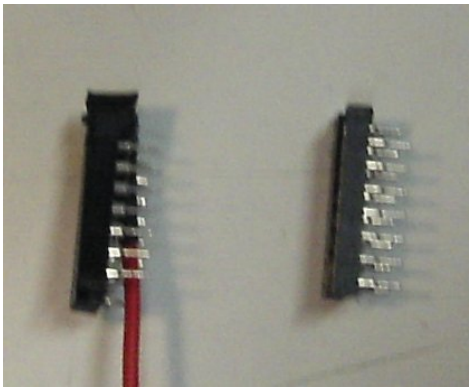

Abbildung A.13: Die Kabel (hier nur eins) liegen zwischen Bauteil 1 und Bauteil 2

Die Fassungen werden vorsichtig zusammengedrückt. Dabei ist unbedingt darauf zu achten, dass beide Fassungen in der richtigen Richtung montiert werden. Die Aussparungen auf den

Fassungen müssen über einander zeigen. Die Ecke mit dem Spannungskabel kommt über den aus gebohrten Pin des Bauteil 1.

Nach nochmaliger Überprüfung werden die Motortreiber des AKSEN-Board vorsichtig aus Ihren Fassungen gehoben und auf den Adapter gesteckt. Auch hier ist wieder sehr auf die Richtung zu achten (Aussparung auf Aussparung).

Anschließend wird der Adapter auf die Motortreiber-Fassungen des AKSEN-Boards gesteckt. Damit ist der Umbau abgeschlossen. Das rote Kabel kann nun an den Plus-Pol der Spannungsquelle angeschlossen werden, das blaue Kabel an den Minus-Pol der Spannungsquelle.

A.7 Anmerkung zum Umbau beider Motortreiber

Sollen beide Motortreiber mit der gleichen Spannung arbeiten, so benötigt nur einer der beiden Adapter das extra Kabel (schwarz/blau) zur Masse der neuen Spannungsquelle VCC3. Das reicht aus, da die Motortreiber nach wie vor mit der Masse des AKSEN-Boards verbunden sind. Der erste Adapter stellt bereits die Verbindung zwischen der Masse des Aksen-Boards und der zusätzlichen Spannungsquelle her. Durch das Weglassen des zweiten Massekabels, wird es bedeutend einfacher die Stecker zur Spannungsquelle zu fertigen.

Weiterhin ist darauf zu achten, dass beide roten Kabel an den Plus-Pol der Spannungsquelle angeschlossen werden.

Sollen Motortreiber 1 mit VCC3 und Motortreiber 2 mit VCC4 betreiben werden, also jeder Motortreiber eine eigene Spannung bekommen, muss jeder Adapter ein Kabel für Spannung und Masse haben. In dem Fall werden insgesamt zwei schwarze und zwei rote Kabel benötigt.

B Inhalt der CD

Die CD zu dieser Bachelorarbeit enthält diese Arbeit als PDF-Dokument, die Datenblätter zur CMU-CAM2, besuchte Webseiten sowie Bilder und Diagramme, welche in dieser Arbeit verwendet wurden.

Der Aufbau ist wie folgt:

- /**Bachelorarbeit** enthält diese Bachelorarbeit als PDF
- /**Datenblaetter** enthält die Datenblätter zu den Komponenten der CMU-CAM2
- /**Besuchte-Webseiten** enthält Kopien der Webseiten aus dem Literaturverzeichnis dieser Arbeit
- /**Bilder**/**Bachelorarbeit** enthält alle Bilder und Diagramme, die in dieser Bachelorarbeit vorkommen.
- /**Bilder**/**AKSEN-Erweiterung** enthält alle Bilder, die im Dokument zur AKSEN-Erweiterung verwendet wurden
- /**Diagramme** enthält alle Diagramme und Zeichnungen aus dieser Arbeit in ihren Originalformaten

C Verwendete Software

Hier ist eine Auflistung einiger hilfreicher Programme, die für diese Arbeit verwendet wurde.

C.1 Für die Simulation erforderliche Software

- **JAVA Link:** www.sun.com
- **POV-Ray** Raytracing Programm für sehr viele Plattformen **Link:** www.povray.org
- **ImageMagick** Kommandozeilenprogramm zum Manipulieren von Bildern **Link:** www.imagemagick.org

C.2 Programmierung

- **NetBeans** Javaprogrammierung besonders hilfreich für die Erzeugung kleiner GUIs **Link:** www.sun.com
- **Eclipse** Mit verschiedenen Erweiterungen als IDE für die C/C++ Programmierung **Link:** http://www.eclipse.org/

C.3 Grafiken und Diagramme

- **Dia** Programm zum Erstellen aller möglichen Diagrammtypen **Link:** http://www.gnome.org/projects/dia
- **GeoGebra** Dynamische Mathematik Software für Schulen **Link:** www.geogebra.org

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 5. April 2007	
Ort, Datum	Unterschrift