HOCHSCHULE FURTWANGEN UNIVERSITY

Fraunhofer IPA

Bachelor-Thesis

Automatische Kalibrierung eines mobilen Serviceroboters

Jannik Sven Abbenseth

14. März 2013

1. Betreuer: Prof. Dr. rer. nat Edgar Seemann

2. Betreuer: Dipl.-Ing. Florian Weißhardt

Kurzfassung

Thema der Arbeit: Automatische Kalbrierung eines mobilen Serviceroboters

Bearbeiter: Jannik Sven Abbenseth

1. Betreuer Prof. Dr. rer. nat Edgar Seemann
2. Betreuer Dipl.-Ing. Florian Weißhardt

Semester Maschinenbau und Mechatronik, WS 2012/2013

Kurzfassung

Das Ziel dieser Arbeit ist es, die automatische Kalibrierung eines mobilen Serviceroboters anzupassen und zu generalisieren. Hierdurch soll ein einheitliches Kalibrierverfahren geschaffen werden, mit dem alle am Institut für Produktionstechnik und Automatisierung eingesetzten Serviceroboter kalibriert werden können.

Dazu wird das bisher eingesetzte Verfahren analysiert und Verbesserungspotentiale festgestellt.

Die beim vorhandenen Verfahren notwendigen Datenaufnahmeschritte konnten nach Abschluss der Arbeit auf einen Schritt reduziert werden. Bisher wurden Denavit-Hartenberg Parameter verwendet. Diese wurden durch Transformationen ersetzt, dadurch ist die Kalibrierung an weiteren Robotern einsetzbar.

Zusätzlich wurde ein Verfahren entwickelt, mit dem ein Kalibrierobjekt durch 2D Laserscanner erkannt werden kann, um diese in späteren Arbeiten zu kalibrieren.

Schlüsselwörter: Robotik, Serviceroboter, Kalbrierung

Abstract

Title of Bachelor-Thesis: Automatic calibration of a mobile service robot

Author:	Jannik Sven Abbenseth
1. Examiner	Prof. Dr. rer. nat Edgar Seemann
2. Examiner	Dipl.-Ing. Florian Weißhardt
Semester	Mechanical Engineering and Mechatronics, WS 2012/2013

Abstract

This Bachelor-Thesis aims for adapting and generalizing the automatic calibration process of a mobile service robot. Hereby a consistent calibration process for all used service robots at the Fraunhofer Institute for Manufacturing Engineering and Automation should be developed.

Therefore the former used calibration prozess was analyzed and its potential for improvement determined.

During the implementation the two data collection steps for the camera and the hand eye calibration were combined into a single step of data collection. Another improvement was made by reorganizing the calculation step to compute the hand eye transformations with recorded transformations instead of Denavit-Hartenberg parameters, which led to a wider range of application on service robots.

Additionally a detector for detecting a calibration pattern in 2D laser range finder was implemented. It could be used in future works for calibrating the mount position of these laser range finders.

Keywords: Calibration, Robotics

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige fremde Hilfe angefertigt habe.

Die verwendeten Literaturquellen sind im Literaturverzeichnis vollständig zitiert.

Villingen-Schwenningen, 14. März 2013

Jannik Sven Abbenseth

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

baselink Ursprungskoordinatensystem des Roboters

DH-Parameter Denavit-Hartenberg-Parameter

DOF Degree of Freedom (Freiheitsgrad)

FhG Fraunhofer Gesellschaft

IPA Institut für Produktionstechnik und Automatisierung

LBR3 Kuka Leichtbauroboter 3

LWA Schunk Light Weight Arm

ROS Robot Operating System

SDH Schunk Dextrous Hand

1 Einleitung

Ein Forschungsschwerpunkt am Institut für Produktionstechnik und Automatisierung (IPA) der Fraunhofer Gesellschaft (FhG) ist das Thema Servicerobotik. Die 1996 vom IPA verfasste Definition für Serviceroboter ist auch heute noch gültig:

> "Ein Serviceroboter ist eine frei programmierbare Bewegungseinrichtung, die teil- oder vollautomatisch Dienstleistungen verrichtet. Dienstleistungen sind dabei Tätigkeiten, die nicht der direkten industriellen Erzeugung von Sachgütern, sondern der Verrichtung von Leistungen für Menschen und Einrichtungen dienen."[1]

Das Ziel der Servicerobotikforschung ist es, den Menschen bei alltäglichen Aufgaben zu unterstützen. Weit verbreitete Beispiele für den Einsatz von Servicerobotern im häuslichen Bereich sind autonome Staubsauger oder Rasenmäher. Der häusliche Einsatz von Servicerobotern ist ein wichtiges Ziel der heutigen Forschung. Zusätzlich zu den oben angesprochenen einfachen Bauformen für Serviceroboter gibt es Roboter für komplexere Aufgaben, die aber noch nicht verbreitet oder erhältlich sind. Beispiele hierfür sind Tankroboter, die dem Kunden beim Tanken die "Schmutzarbeit" abnehmen oder persönliche Assistenten, wie dem PR2 von Willow Garage[1] oder der Care-O-bot®des IPA. Die beiden letztgenannten wurden als vielseitige Assistenten entwickelt, die in verschiedenen Tätigkeitsfeldern eingesetzt werden können.

[1] Willow Garage ist ein Kalifornisches Unternehmen, das die Entwicklung ziviler Roboter durch Hardware und Open-Source Software beschleunigen will.

Der in dieser Arbeit behandelte Care-O-bot®soll es älteren oder körperlich eingeschränkten Menschen ermöglichen, ein unabhängiges Leben zu führen. Auch in Pflegeheimen oder Krankenhäusern könnte der Care-O-bot®eingesetzt werden. Hier können Hol- und Bringdienste ausgeführt oder die Trinkgewohnheiten der Bewohner und Patienten überwacht und verbessert werden. Dadurch bleibt dem Personal mehr Zeit für den persönlichen Kontakt zu den Patienten. Die Forschung am IPA richtet sich aber auch an industriellen Einsatzzwecken dieser persönlichen Assistenten aus. Der ebenfalls von IPA entwickelten Roboter rob@work®kann Hol- und Bringdienste ausführen sowie bei Arbeiten unterstützen, die viel Erfahrung und Präzision benötigen. Wichtig für diese Anwendungen sind die bisherigen Entwicklungen in den Bereichen Navigation und Manipulation sowie die Realisierung kognitiver Fähigkeiten für Roboter.

Um in einem sich stetig ändernden Umfeld sicher und erfolgreich agieren zu können, ist eine genaue und sichere Wahrnehmung und Manipulation essentiell. Um dieses zu ermöglichen, muss der Roboter kalibriert werden. Diese Kalibrierung muss sowohl intrinsisch, also für den einzelnen Sensor oder Aktor an sich, als auch extrinsisch, also für die Komponente im Bezug auf das Gesamtsystem, erfolgen. Wenn die Kalibrierung falsch oder nicht ausgeführt wird, entsteht eine Gefährdung für den Roboter selbst und auch für seine Umwelt.

Im Fall des Care-O-bot®, der neben seinem eigentlichen Einsatzzweck als Serviceroboter auch als Forschungsroboter, auf dem verschiedene neue Robotertechnologien getestet und evaluiert werden können, genutzt wird, ist eine zuverlässige Kalibrierung besonders wichtig. Der Care-O-bot® besteht aus einer omnidirektional bewegbaren Plattform, auf der ein Torso mit mehreren Freiheitsgraden und ein Roboterarm mit Greifer angebracht sind. Außerdem befinden sich auf der Plattform drei Laserscanner zur Navigation und Hindernis-/Objekterkennung. Am Torso sind drei Kameras, die den aufwendigsten und wichtigsten Teil der Sensorik ausmachen, montiert.

In vorhergehenden Arbeiten wurde ein Verfahren implementiert, mit dem es möglich ist, eines der zwei in Stuttgart eingesetzten Modelle des Care-O-bot® automatisch zu kalibrieren. Für diese Kalibrierung muss der Roboter teilweise demontiert werden, um

bestimmte Parameter manuell ablesen und einstellen zu können. Außerdem müssen die Denavit-Hartenberg-Parameter (DH-Parameter)[2] des Torsos und des Arms von Hand ausgerechnet und in eine Konfigurationsdatei übertragen werden. Die zur Kalibrierung notwendigen Positionen des Arms werden in einer weiteren Datei als Gelenkwinkel abgelegt. Um einen neuen Roboter zu kalibrieren, müssen die Berechnungen der Gelenkwinkel und der DH-Parameter erneut ausgeführt werden.

Diese Bachelorthesis soll den Funktionsumfang der bisherigen Kalibrierung erweitern und generalisieren. Hierzu sollen die Konfigurationsdateien vereinfacht, automatisch generiert oder überflüssig gemacht werden. Außerdem soll eine einfache Möglichkeit entwickelt werden, Roboter, die sich in ihrem Aufbau, zum Beispiel durch einen zweiten Arm, vom Care-O-bot® unterscheiden, zu kalibrieren.

[2] Siehe Kapitel 2.2.4

1.1 Aufbau und Beschreibung des Care-O-bot®

Das Projekt Care-O-bot® wurde vom IPA bereits 1998 ins Leben gerufen. Bisher sind daraus drei Robotergenerationen hervorgegangen. Von der aktuellen dritten Generation gibt es bislang sieben Modelle, von denen zwei in Stuttgart am IPA eingesetzt werden. Dies sind die beiden Modelle Care-O-bot®3-3 und Care-O-bot®3-6. Die anderen Modelle wurden an externe Forschungseinrichtungen verkauft und waren für Tests an der Hardware nicht verfügbar. Der Grundaufbau ist für jeden Care-O-bot® der dritten Generation derselbe und am Beispiel des Care-O-bot®3-2 in Abbildung 1.1 dargestellt.

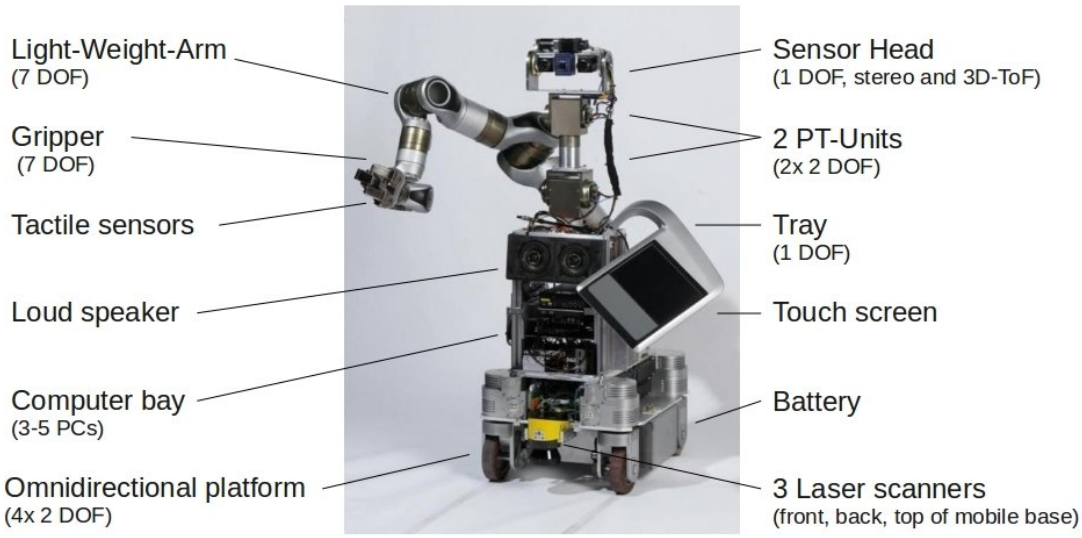

Abbildung 1.1: Hardwarekomponenten des Care-O-bot® [3]

Die einzelnen Modelle unterscheiden sich aber in ihren eingesetzten Komponenten. So ist der offensichtlichste Unterschied zwischen den in Abbildung 1.2 abgebildeten Care-O-bot®3-3 und Care-O-bot®3-6 der von verschiedenen Herstellern stammende Arm.

(a) Care-O-bot®3-3 mit LBR3

(b) Care-O-bot®3-6 mit LWA

Abbildung 1.2: Care-O-bot®3-3 und Care-O-bot®3-6 im Vergleich[4]

1.1.1 Aktoren

Die verschiedenen Care-O-bot® Distributionen sind mit den folgenden Aktoren und Sensoren ausgestattet, für die jeweils auch die Unterschiede zwischen den Modellen Care-O-bot®3-3 und Care-O-bot®3-6 aufgelistet sind.

Arm Für den an der Rückseite des Roboters angebrachten Arm werden zwei Modelle eingesetzt. Einmal der Kuka Leichtbauroboter 3 (LBR3) von Kuka sowie der Schunk Light Weight Arm (LWA) von Schunk.

Torso Der Torso ist je nach Modellnummer ein 3 Degree of Freedom (Freiheitsgrad) (DOF) oder ein 4 DOF Modell. Sowohl beim Care-O-bot®3-3 als auch beim Care-O-bot®3-6 wird das 3 DOF Modell eingesetzt.

Base Als Basis für den Roboter dient eine omnidirektionale Plattform. Durch die vier lenkbaren Räder ist es dem Roboter möglich, aus dem Stand in alle Richtungen

zu fahren oder zu drehen.

Tablett Das Tablett kann beim Care-O-bot®3-3 hoch und runter geklappt werden und hat einen Touchscreen eingelassen, mit dem der Roboter bedient werden kann. Beim Care-O-bot®3-6 stehen mehr Freiheitsgrade zur Verfügung. Hier kann das Tablett zusätzlich entlang zwei Achsen gedreht werden. Dadurch kann die Bedienung vorallem für sitzende Personen bequemer gemacht werden. Zur Bedienung des Roboters ist im Tablett ein Touchscreen eingebaut.

Hand Als Hand wird die Schunk Dextrous Hand (SDH) mit drei Fingern und sieben Freiheitsgraden eingesetzt.

Kopfachse Mit der Kopfachse kann die Kamera um 180 ° nach hinten geschwenkt werden, um sowohl im Manipulationsbereich hinter dem Roboter als auch im Servicebereich vor dem Roboter Bildinformationen zu erhalten.

Lautsprecher Um dem Nutzer Rückmeldung über die aktuelle Tätigkeit des Roboters zu geben, sind in der Plattform Lautsprecher eingebaut.

1.1.2 Sensoren

Kameras Für die 3-dimensionale Bilderfassung werden verschiedene Systeme eingesetzt. Alle Care-O-bot® der dritten Generation besitzen ein Stereokamerapaar aus zwei hochauflösenden Kameras. Darüber hinaus wird im Care-O-bot®3-3 eine Microsoft Kinect, im Care-O-bot®3-6 eine Asus X-Tion Pro und in anderen Modellen eine Time-of-flight Kamera eingesetzt.

Laserscanner Hinten auf der Plattform ist ein Laserscanner von Hokuyo verbaut, der für die Navigation eingesetzt wird. Außerdem ist vorne und hinten je ein Sicherheitslaserscanner von SICK verbaut, die zusätzlich den Not-Aus aktivieren, wenn sie ein Hindernis in unmittelbarer Nähe detektieren.

Taktile Sensoren Damit der Roboter unterschiedliche Dinge sicher greifen kann, sind an der Hand taktile Sensoren angebracht, die die Greifkraft ermitteln können.

Mikrofon Unter anderem zur Sprachsteuerung ist im Kopf des Roboters ein Mikrofon verbaut. Je nach 3D Kamera kann es auch in diese integriert sein.[5]

1.2 Kalibrierungsbedarf

Wie jeder andere Roboter auch muss der Care-O-bot® kalibriert werden. Durch Ungenauigkeiten bei der Montage oder Kollisionen beim Transport oder im Betrieb entspricht der Aufbau des Care-O-bot® nie dem auf der technischen Zeichnung. Obwohl diese Ungenauigkeiten selten größer als ein paar Millimeter oder Grad sind, können zum Beispiel Winkelfehler bei der Montage des Arms zu Abweichungen führen, die es dem unkalibrierten Roboter unmöglich machen, ein Objekt, dessen genaue Position bekannt ist, zu greifen. Durch den Aufbau und Einsatz des Care-O-bot® müssen Objekte erst von der Sensorik erkannt und im Raum lokalisiert werden. Wenn in diesem Schritt die Kameras an unbekannten Stellen des Roboters angebracht sind, kann aus den gewonnenen Daten keine zielführende Aktion des Roboters berechnet werden. Ein erkanntes Objekt muss also erst im Kamerakoordinatensystem lokalisiert werden. Diese Position muss dann in ein zentrales Roboterkoordinatensystem umgerechnet werden. Teil dieser Transformation sind beim Care-O-bot® mindestens zwei unbekannte Einzeltransformationen. Dies ist die Montageposition des Torsos sowie die Position der Kameras auf dem drehbaren Kameraträger. Zusätzlich kann noch die Montageposition des Kopfes auf dem Torso unbekannt sein. Ein Fehler an dieser Stelle kann aber auch durch die zwei anderen Transformationen ausgeglichen werden. Um das Objekt schließlich greifen zu können, muss die Position des Objekts relativ zum Ursprung des Arms bekannt sein, was eine genaue Kenntnis der Montageposition des Arms vorraussetzt. Auf diesem Weg können sich kleine Fehler zu großen Ungenauigkeiten beim Griff aufaddieren.[6]

2 Grundlagen

Zu Begin der Arbeit liegt bereits eine vollständige Beschreibung aller relevanten Komponenten der zu kalibrierenden Roboter vor. Daraus lässt sich sowohl die Vorwärtskinematik, also die Transformation eines Koordinatensystems einer Komponente, beispielsweise der Befestigungspunkt der Roboterhand am Roboterarm, zu einem anderen Koordinatensystem derselben Komponente zum Beispiel der Befestigungspunkt des Arms an der Roboterplattform, anhand von Gelenkwinkeln berechnen. Außerdem kann die inverse Kinematik, also die benötigten Gelenkwinkel zum Erreichen eines bestimmten Punktes, für den Arm berechnet werden.

2.1 Kalibrierung

Der Ausgangszustand der automatischen Kalibrierung liefert für den Care-O-bot®3-3 gute und zuverlässige Ergebnisse. Zur Kalibrierung muss ein Kalibrierungsmuster, das für den Care-O-bot® bisher ein Schachbrettmuster mit neun mal sechs inneren Ecken [7] ist, anstatt der SDH an den Arm montiert werden. Nach dem anschließenden Einstellen der Nullwerte für die Aktoren beginnt die erste Datenaufnahme für die Kalibrierung der Stereokameras. Die Kameras „Microsoft Kinect“ oder „Asus XTion Pro“ müssen aufgrund der guten Kalibrierung vom Hersteller nicht kalibriert werden.

Mit den gewonnenen Daten kann der Kalibrieralgorithmus für Stereokameras, der von OpenCV bereitgestellt wird, sowohl die intrinsischen Kameraparameter als auch den Versatz der Kameras zueinander, den sogenannten Baselineshift, berechnen.

Mit den jetzt kalibrierten Kameras werden ein zweites mal Daten aufgenommen. Hierzu werden der Roboterarm und der Torso in vorher festgelegte Positionen gefahren. An jeder Position werden die aktuellen Gelenkwinkel zusammen mit der, von den Kameras erkannten, Position des Kalibrierungsmusters aufgenommen.

Im letzten Schritt berechnet ein Optimierer anhand der aufgenommenen Daten die gesuchten Transformationen[3].[6]

Im Folgenden werden die vorhanden Konfigurationsdateien `system.yaml`, `sensors.yaml` und die drei `free_x.yaml` mit den in ihnen gespeicherten Parametern erklärt.

2.1.1 Konfigurationsdatei system.yaml

Die im Package `cob_robot_calibration` für den Care-O-bot®3-3 vorhandene `system.yaml` Datei enthält alle für die Berechnungen des Optimierers benötigten Daten. Dazu gehören:

Transformationen Feste Transformationen zwischen dem Ursprungskoordinatensystem des Roboters (baselink) und den Koordinatensystemen der Kameras und des Kalibrierungsmusters sind abgelegt. Dabei werden sowohl bekannte als auch Näherungen für unbekannte Transformationen angegeben.

Beschreibungen der Aktoren Die Denavit-Hartenberg-Parameter und Informationen zum Übersetzungsverhältnis und zur Genauigkeit der einzelnen Gelenke der Aktoren sind abgelegt. Das Übersetzungsverhältnis wird bereits im Treiber des Aktors berücksichtigt und ist hier immer mit 1.0 anzugeben.

Kameraparameter Hier können Kameraparameter definiert werden. Da bei der vorliegenden Kalibrierung schon verarbeitete Bildinformationen eingesetzt werden, sind hier nur Standardwerte eingetragen.

[3] Siehe Kapitel 2.3.2

2.1.2 Konfigurationsdatei sensors.yaml

In der Konfigurationsdatei werden die Ketten, aus denen der Roboter zusammengesetzt werden kann, definiert. Dazu wird in `camera_chains`, die die Ketten zu den Kameras angeben und `chains`, die die kinematische Kette zum Kalibrierobjekt beschreiben, unterschieden. Dazu wird jeder Kette eine eindeutige `sensor_id` sowie eine Beschreibung der Transformation zum Endpunkt zugeordnet.

Die Kette besteht aus den zwei festen Transformationen `before_chain` und `after_chain`, die die Transformation vom Roboterursprung zum Aktor und vom Endpunkt des Aktors zum Sensor beschreiben. Zusätzlich ist eine eindeutige `chain_id` zur Zuordnung der Kette zu den in der `system.yaml` angegebenen Beschreibung der Aktoren, sowie die Anzahl der Gelenke des Aktors definiert.

2.1.3 Konfigurationsdateien free_x.yaml

Die drei `free_x.yaml` Dateien haben den gleichen Aufbau wie die `system.yaml`. Sie werden benötigt, um dem Optimierer mitzuteilen, wann welche Parameter variabel sind.

2.2 Koordinatentransformationen

Ein wichtiges Thema bei der Robotik ist die Transformation eines Koordinatensystems in ein anderes. In der Robotik bekommt jedes Objekt (z.B. Roboter, Kamera, Gegenstände), das maßgeblich an einer Operation beteiligt ist, ein eigenes Koordinatensystem um seine Lage, bestehend aus Position und Orientierung, im Raum zu beschreiben. Dadurch lässt sich berechnen, wie ein Objekt gegriffen werden muss oder Kollisionen vermieden werden können.

Die Transformation von einem Koordinatensystem in ein anderes, wie sie zum Beispiel vorkommt, wenn die Kameras ein Gegenstand erkennen, kann durch eine Translation und eine Rotation dargestellt werden. Wenn die Lage der Kameras relativ zur Basis des Roboters bekannt ist, ist auch die Lage des Gegenstand im Roboterkoordinatensystem bekannt.

Für den Care-O-bot® ist jedes Koordinatensystem ein Kind eines sogenannten Parent-Frames, das wiederum ein Koordinatensystem ist. Ein Parent-Frame kann mehrere Kinder haben, aber ein Kind kann nur einen Parent-Frame haben. Außerdem sind alle Koordinatensysteme des Care-O-bot® orthonormale Koordinatensysteme.

2.2.1 Translation

Die Translation beschreibt nur die Position eines Punktes im Raum. Für die Koordinatentransformation ist dieser Punkt der Ursprung des neuen Koordinatensystems. Der Punkt P wird im n-Dimensionalen durch einen Koordinatenvektor mit n Elementen angegeben. Jedes Element p_n gibt die Entfernung entlang der n-ten Achse des Parent-Frames an. Räumliche Transformationen beschränken sich auf drei Dimensionen und damit den Punkt $P = [p_x p_y p_z]^T$.

Da für jeden Punkt festgelegt sein muss, in welchem Koordinatensystem A er angegeben ist, wird das Bezugskoordinatensystem in der Form ${}^{A}P$ angegeben.

Translationen lassen sich durch Addition zusammenfassen. Wenn ein Punkt im Koordinatensystem A, das um den Vektor Z vom Koordinatensystem B verschoben ist, angegeben ist und seine Position abhängig vom Koordinatensystem B angegeben werden soll, errechnet sich seine neue Position aus ${}^{B}P = {}^{A}P + Z$.

2.2.2 Rotation

Da in der Robotik die Position eines Objektes selten ausreichend ist, wird zusätzlich die Orientierung benötigt. Dafür bekommt das Objekt ein Koordinatensystem bestehend aus dem Ursprung P und drei Achsen. Die Ausrichtung der drei Achsen werden durch drei orthogonale Vektoren der Länge „1“ im Parent-Frame angegeben. Die drei Vektoren, die die Achsen des Koordinatensystems B im Parent-Frame A angeben, werden eindeutig mit ${}^{A}\hat{X}_B$, ${}^{A}\hat{Y}_B$ und ${}^{A}\hat{Z}_B$ angegeben. Diese drei Vektoren werden zu der sogenannten Rotationsmatrix

$$ {}^{A}_{B}R = [{}^{A}\hat{X}_B\, {}^{A}\hat{Y}_B\, {}^{A}\hat{Z}_B] = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} $$

zusammengefasst.

Für die Rotation ${}^{B}_{A}R$ gilt wie in [8] angegeben:

$$ {}^{B}_{A}R = {}^{A}_{B}R^T $$

Zur vereinfachten Darstellung von Rotationen genügen drei Winkel, aus denen die Rotationsmatrix berechnet werden kann. Am Care-O-bot® wird hierfür die

Roll-Pitch-Yaw

Präsentation gewählt. Weitere verbreitete Methoden, sind wie in [9] dargestellt, die zwölf Versionen der Euler-Winkel.

Roll-Pitch-Yaw

Die Roll-Pitch-Yaw Beschreibung stammt aus der Luft- und Seefahrt und wird genutzt, um die Orientierung eines Objekts mit den Winkeln Rollwinkel, Gierwinkel und Nickwinkel anzugeben.

ϕ Rollwinkel: Winkel, der eine Drehung um die X-Achse des Referenzkoordinatensystems angibt

ψ Gierwinkel: Winkel, der die Drehung um die Z-Achse eines Referenzkoordinatensystems beschreibt

θ Nickwinkel: Winkel, der die Rotation um die feste Y-Achse beschreibt.

Die Rotationsmatrix berechnet sich daraus mit $c_x = cos(x)$ und $s_x = sin(x)$ zu:

$$R_{RPY} = \begin{pmatrix} c_\phi * c_\theta & c_\phi * s_\psi * s_\theta - s_\phi * c_\theta & c_\phi * s_\psi * c_\theta - s_\phi * s_\theta \\ s_\phi * s_\psi & s_\phi * s_\psi * s_\theta + c_\phi * c_\theta & s_\phi * s_\psi * c_\theta - c_\phi * s_\theta \\ -s_\psi & c_\psi * s_\theta & c_\phi * c_\theta \end{pmatrix}$$

[9]

Mit dieser Rotationsmatrix kann die Position eines Punktes in einem zu seinem Ursprungskoordinatensystem gedrehten Koordinatensystem angegeben werden. Der neue Punkt P' errechnet sich dabei aus $P' = R * P$. Der Vorteil der Angabe einer Rotation als Matrix anstatt als Tripel von Winkeln entsteht aus der einfachen Matrixmultiplikation zur Berechnung einer Rotation. [8]

Alle für den Care-O-bot® angegebenen Orientierungen in der Roboterbeschreibung sind als Roll-Pitch-Yaw Winkel angegeben. Außerdem kann durch vorhandene Nodes[4] die Orientierung von jedem Koordinatensystem des Care-O-bot® relativ zu jedem anderen Koordinatensystem des Care-O-bot® in einem Translationsvektor und den Roll-Pitch-Yaw Winkeln berechnet werden.

[4] siehe Abschnitt 2.4

2.2.3 Verkettung von Transformationen

Die Transformation eines Koordinatensystems zu einem anderen lässt sich also durch eine Matrixmultiplikation und eine Vektoraddition ausdrücken. Um dies zu vereinfachen werden die Rotationsmatrix und der Translationsvektor zu einer Translationsmatrix zusammengefasst. Die Zusammenfassung ist nur möglich, wenn die dadurch entstehende 3×4 Matrix zu einer homogenen 4×4 erweitert wird. Die Translationsmatrix wird dann mit

$$ {}^{A}_{B}T = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

angegeben. Dadurch lässt sich der Übergang von einem Koordinatensystem in ein anderes durch eine Matrixmultiplikation angeben.[8]

2.2.4 Denavit-Hartenberg Parameter

Die Denavit-Hartenberg Notation basiert auf einem von Denavit und Hartenberg entwickelten Verfahren zur Berechnung von Transformationen entlang einer kinematischen Kette. Dazu werden für jedes Gelenk vier Parameter benötigt. Die sogenannten DH-Parameter. Aus diesen Parametern kann dann für jedes Gelenk die Transformationsmatrix zum vorherigen Gelenk berechnet werden. Das Verfahren vereinfacht vor allem die Berechnung der Vorwärtskinematik, also die Berechnung von Positionen anhand der Gelenkzustände.

Um eine Transformation mit vier statt den sonst üblichen sechs Parametern zu beschreiben, werden die möglichen Freiheitsgrade der Gelenke limitiert. Um einen Aktor mit DH-Parametern zu beschreiben dürfen alle Gelenke nur einen Freiheitsgrad

haben. In der Anwendung führt dies aufgrund der gebräuchlichen Roboterbauformen - Schubgelenk oder Drehgelenk - aber selten zu Einschränkungen.

Zur Angabe einer Transformation werden die Parameter $[a, \alpha, d, \Theta]$ angegeben. Daraus kann die homogene Translationsmatrix berechnet werden.[8]

2.3 Kalibrieralgorithmen

Zu Beginn der Arbeit gab es bereits einige Algorithmen zur automatischen Kalibrierung der am Roboter eingesetzten Hardwarekomponenten. Die wichtigsten und bereits am Care-O-bot® implementierten sind die Verfahren zur Kalibrierung von Mono- und Stereokameras, die OpenCV[5] zur Verfügung stellt und das von Vijay Pradeep entwickelte Verfahren[10] zur Kalibrierung von Robotern mit mehreren Armen und mehreren Sensoren.

2.3.1 Kamerakalibrierung

Um Objekte, die von den Kameras erkannt werden, in der dreidimensionalen Welt einzuordnen, muss ein möglichst exaktes mathematisches Modell entwickelt werden, nach dem Objektpunkte auf die Sensorfläche projiziert werden. Außerdem müssen die Parameter des Stereokamerasystems für die Bestimmung von Tiefendaten aus zwei Einzelbildern bekannt sein. Da diese Parameter nicht am Roboter oder der Kamera abgemessen werden können, wird auf Methoden der Bildverarbeitung zurückgegriffen.

[5] Open Computer Vision: Eine Open-Source Softwarebibliothek die Funktionen zur Bildverarbeitung bereitstellt

Monokamerakalibrierung

Zur Berechnung der Projektion eines Punktes im Raum auf einen Film oder Bildsensor muss ein Modell erzeugt werden, das die Abbildungseigenschaften einer Kamera möglichst einfach aber genau genug wiedergibt. Dieses Modell kann durch eine Kombination des sogenannten Lochkameramodells und des modellierten Einflusses von optischen Linsen beschrieben werden.[11]

Die Lochkamera besteht, wie in Abbildung 2.1 dargestellt, aus einer geschlossenen Box, in der auf einer Seite ein Sensor oder Film angebracht ist. Gegenüber vom Sensor befindet sich eine kleine Öffnung, das optische Zentrum C, durch das idealerweise nur ein Lichtstrahl pro Objektpunkt auf den Sensor trifft. Gemäß den geometrischen Gesetzen ergibt sich ein auf dem Kopf stehendes Abbild des Objektes auf der Bildebene oder dem Sensor. Zur Vereinfachung der Rechnung kann eine virtuelle Bildebene vor der Kamera angenommen werden, die durch eine Punktspiegelung der Bildebene am optischen Zentrum erzeugt werden kann. Im Vergleich zur Berechnung für die Bildebene sind die Werte hierbei invertiert.[12]

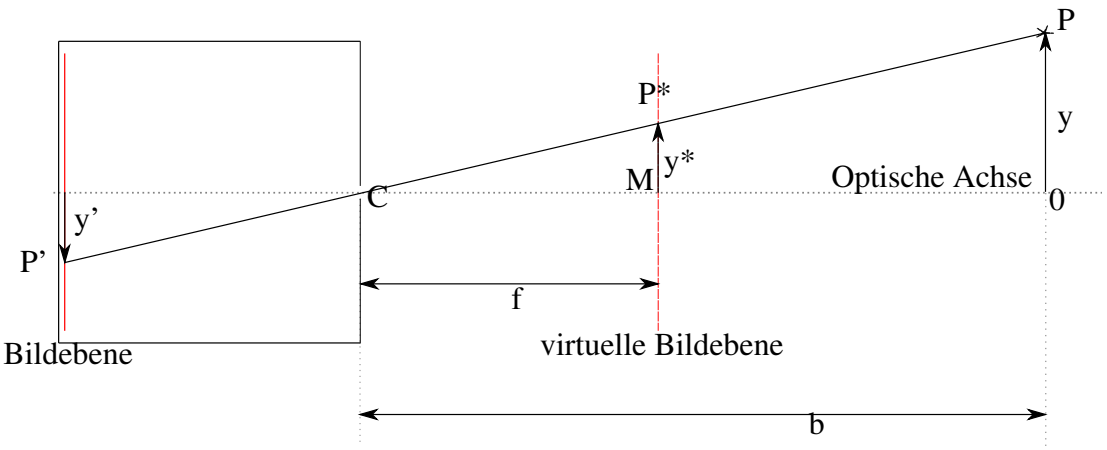

Abbildung 2.1: Prinzipskizze einer Lochkamera

Der Abstand von optischem Zentrum zur (virtuellen) Bildebene wird Fokallänge f genannt. Der Punkt P wird durch die Projektion auf den Sensor zum Punkt P' und

gespiegelt zum Punkt P^* mit den Koordinaten (x^*, y^*). Da die Dreiecke $C, M, P*$ und $C, 0, P$ ähnlich sind gilt:

$$\frac{y^*}{f} = \frac{y}{b} \Rightarrow y^* = \frac{y}{b} * f$$

Genauso gilt für x^*:

$$x^* = \frac{x}{b} * f$$

Da der Ursprung des Sensorkoordinatensystems aufgrund von Konventionen und Ungenauigkeiten bei der Kameramontage in den seltensten Fällen mit der optischen Achse der Kamera übereinstimmt, werden die projizierten Punkte noch um c_x und c_y verschoben.

Eine weitere Besonderheit bei digitalen Kameras ist die Form der Pixel auf dem Sensor. Da diese rechteckig sind, müssen im Modell verschiedene Werte f_x und f_y für die Fokallänge angenommen werden.[11]

Wenn diese beiden Besonderheiten berücksichtigt und in Vektorschreibweise dargestellt werden, ergibt sich

$$\begin{pmatrix} x^* \\ y^* \end{pmatrix} = \begin{pmatrix} \frac{x}{b} * f_x + c_x \\ \frac{y}{b} * f_y + c_y \end{pmatrix}$$

Um die Projektion durch eine Matrixmultiplikation zu berechnen, wird die homogene Projektionsmatrix Pr berechnet. Dazu werden die Vektoren $P = (x, y, b)^T$ und $P^* = (x^*, y^*)^T$ zu den homogenen Vektoren $\tilde{P} = (x, y, b, 1)^T$ und $\tilde{P^*} = (u^*, v^*, w^*)^T$ erweitert.

Die Berechnung der Projektion erfolgt jetzt durch

$$\tilde{P^*} = Pr * \tilde{P}$$

$$\tilde{P^*} = \begin{pmatrix} u^* \\ v^* \\ w^* \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{pmatrix} x \\ y \\ b \\ 1 \end{pmatrix}$$

.[11]

Zur Umrechnung von homogenen in nicht homogene Koordinaten berücksichtigt man, dass Punkte mit dem gleichen Verhältnis der Komponenten gleich sind. Außerdem gilt für $w^* = 1$

$$\tilde{P^*} \begin{pmatrix} u^* \\ v^* \\ w^* \end{pmatrix} = \begin{pmatrix} x^* \\ y^* \\ 1 \end{pmatrix}$$

.

Daraus folgt:

$$P^* = \begin{pmatrix} x^* \\ y^* \end{pmatrix} = \frac{1}{w^*} * \begin{pmatrix} u^* \\ v^* \end{pmatrix} = \begin{pmatrix} \frac{u^*}{w^*} \\ \frac{v^*}{w^*} \end{pmatrix}$$

Bei der Kalibrierung einer Kamera wird also die Projektionsmatrix Pr oder die Kameramatrix K berechnet, wobei gilt, dass

$$Pr = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \left[K \, (0,0,0)^T\right]$$

.[11]

Der größte Nachteil bei Lochkameras ist die geringe Lichtmenge, die durch das Loch auf den Sensor fällt. Dadurch würde der Aufnahmevorgang für ein Bild sehr lange brauchen, bis ausreichend Licht auf den Sensor gefallen ist.

Um diesen Nachteil auszugleichen, werden in Kameras Linsen verwendet, die das Licht bündeln, um mehr Licht auf den Senor fallen zu lassen. Durch Linsen werden allerdings neue Abbildungscharakteristiken eingeführt, die im Modell berücksichtigt

werden müssen. Da es nur zwei Charakteristiken gibt, die einen nennenswerten Einfluss auf das Bild haben, kann das Modell einfach erweitert werden. Diese beiden Einflüsse entstehen aufgrund der Linsenform und der Einbaulage des Sensors gegenüber der optischen Achse.

Eine parabolische Linse oder ein komplexes Linsensystem hat nur einen geringen Einfluss auf die Abbildung. Da aber häufig nur günstige sphärische Linsen eingesetzt werden, muss die radiale Verzeichnung korrigiert werden. Der Effekt nimmt ausgehend vom optischen Zentrum zu und ist rotationssymetrisch dazu. Die radiale Verzeichnung kann durch die ersten sieben Glieder einer Taylorreihe $f(x) = a_0 + a_1 * x + a_2 * x^2 + \cdots + a_n * x^n$ beschrieben werden. Aus der Symmetrie folgt, dass alle ungeraden Glieder gleich Null sind. Außerdem ist die Verzeichnung am Mittelpunkt gleich Null. Deswegen ist auch $a_0 = 0$.

Die korrigierten Bildpunkte ergeben sich damit zu

$$x_{korr} = x * (1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$
$$y_{korr} = y * (1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6)$$

.

Die Verzerrung durch eine nicht parallele Einbaulage des Sensors zur Linse lässt sich durch die zwei Parameter p_1 und p_2 beschreiben.[11]

Der am Care-O-bot® eingesetzte Kalibrieralgorithmus wurde von Zhang und Sturm entwickelt und basiert auf der Erkennung eines definierten Musters. Dazu werden Bilder von dem Objekt aufgenommen und wichtige Punkte - im Fall des Care-O-bot® Ecken eines Schachbretts - erkannt und zusammen mit einem Modell des Musters an den Kalibrieralgorithmus übergeben. Dieser geht im ersten Schritt von einer Lochkamera ohne Verzerrungen durch Linsen aus. Anschließend wird der Parametervektor $[k_1, k_2, p_1, p_2, k_3]$ berechnet.[13]

Man erhält also alle wichtigen Parameter, um einen Punkt im Raum auf einen Punkt auf dem Sensor zu projizieren oder einem Punkt auf dem Sensor eine Linie im Raum

zuzuordnen. Außerdem kann dadurch die Pose, also die Position und die Orientierung, eines bekannten Objekts im Raum berechnet werden.

Stereokamerakalibrierung

Ein Stereokamerasystem besteht, ähnlich wie ein menschliches Augenpaar, aus zwei Kameras mit überlappendem Sichtfeld. Mit den aus den zwei Kamerabildern gewonnenen Informationen über einen Punkt lässt sich die genaue Position des Punktes im Raum berechnen. Bei einem idealen Stereokamerasystem, wie in Abbildung 2.2 dargestellt, liegen beide Bildebenen auf einer Ebene und die Transformation von einer zur anderen Kamera kann durch eine Translation entlang der X-Achse ausgedrückt werden. Dadurch sind auch alle Pixelreihen aneinander ausgerichtet. Wenn diese Bedingungen erfüllt sind, kann der Abstand eines Punktes von der Ebene parallel zu den Bildebenen durch die optischen Zentren der Kameras durch

$$Z = \frac{f * T}{a - b}$$

berechnet werden. Dabei ist a die X-Koordinate des Punktes in Kamera 1 und b die X-Koordinate des Punktes in Kamera 2.

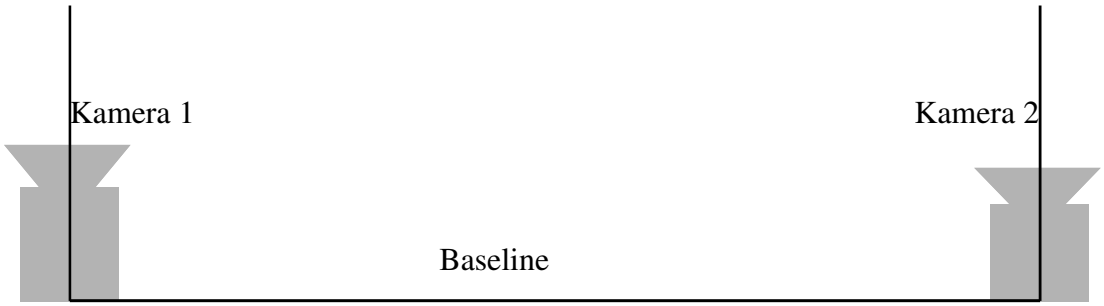

Abbildung 2.2: Draufsicht eines idealen Stereokamerapaares

Da in der Praxis keine ideale Ausrichtung der Kameras möglich ist, muss die Transformation durch eine 6D Transformation angegeben werden. Aus dieser Transformation und den intrinsischen Kameraparametern kann dann ein neues Bild berechnet werden, in dem alle Bedingungen erfüllt sind.

Die Kalibrierung eines Stereokamerapaares benötigt also, zusätzlich zu den Parametern für eine Kamera, die Transformation zwischen den Kameras bestehend aus einem 3D Translationsvektor und einer 3x3 Rotationsmatrix.

Zur Kalibrierung werden wie bei der Monokamerakalibrierung Bilder von einem definierten Objekt aufgenommen. Hierbei ist es wichtig, dass das Objekt auf beiden Bildern zu sehen ist. Daraus wird die 6D Pose des Objekts für jede Kamera berechnet. Wie in Figur 2.3 dargestellt ergibt sich für die Rotationsmatrix

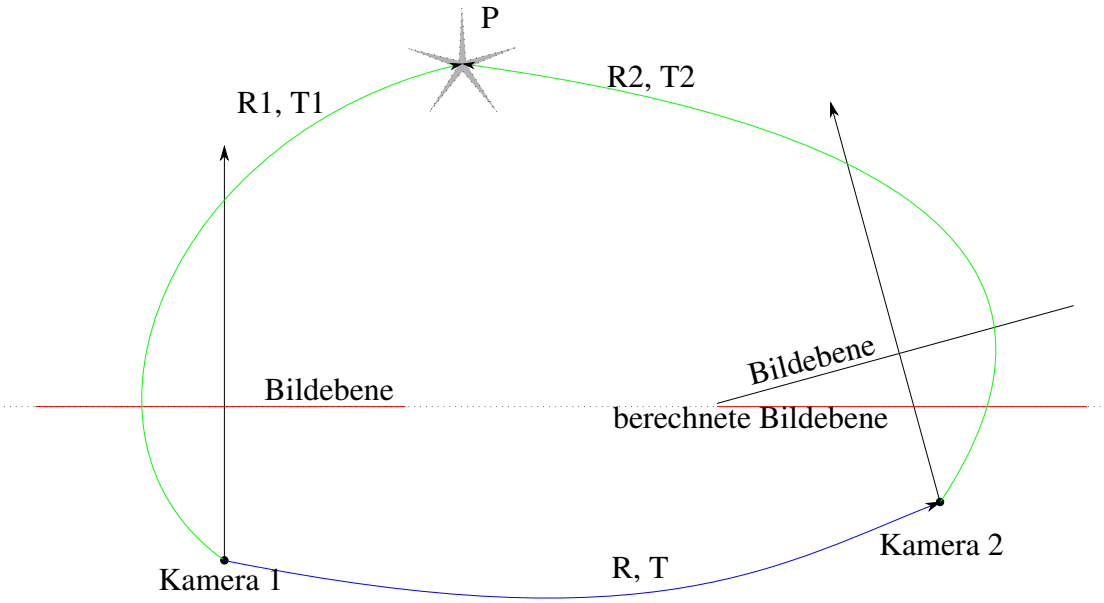

Abbildung 2.3: Transformationen im Stereokamerasystem

$$R = R_1 * (R_2)^T$$

und für den Translationsvektor

$$T = T_1 - R * T_2$$

Obwohl es möglich ist, die gesuchte Transformation aus einem Bildpaar zu berechnen, werden mehrere Paare aufgenommen, um Bildrauschen und andere Störeinflüsse zu kompensieren. Dazu wird der Median der gefundenen Transformationen als Startwert für ein Optimierungsverfahren verwendet, das zuverlässig die beste Lösung findet.

Mit den gefundenen Parametern können anschließend die Rektifizierungsmatrizen für beide Kameras berechnet werden, die auf die Bilder angewendet ein Bildpaar ergeben, bei dem alle Bedingungen für ein ideales Stereokamerasystem zutreffend sind.[11]

2.3.2 Kinematische Kalibrierung

Zur kinematischen Kalibrierung des Roboters wird ein Verfahren verwendet, das auf dem Kalibrierverfahren für den PR2 von Vijay Pradeep basiert.[10]

Nach der Festlegung, welche Transformationen kalibriert werden sollen, werden Pfade festgelegt, über die transformiert wird. Diese Pfade haben einen gemeinsamen Ausgangspunkt und als Endpunkt entweder einen Sensor oder ein detektierbares Objekt. Außerdem müssen alle vorher festgelegten Transformationen in den Pfaden enthalten sein. In Abbildung 2.4 sind zwei Pfade dargestellt. Pfad 1 geht vom Ursprungskoordinatensystem zu dem detektierbaren Objekt und enthält neben der bekannten Koordinatentransformation des Roboterarmes auch die zwei unbekannten Montagepositionen des Arms auf der Basis und des Kalibrierobjekts am Arm. In Pfad 2 sind die Transformation von der Basis zum Torso und vom Torso zur Kamera unbekannt. Die gesamten Pfade 1 und 2 sind in Grün dargestellt und setzen sich aus den roten festen Transformationen und den schwarzen variablen Transformationen zusammen.

Im Anschluss werden Positionen für die in den Pfaden eingeschlossenen Aktoren festgelegt, in denen das Kalibrierobjekt von allen Sensoren detektiert werden kann. Außerdem müssen vorgegebene Transformationen und Konfigurationen für den Optimierer bereitgestellt werden. Diese Schritte müssen für jeden Roboter einmalig ausgeführt werden und sind direkt von der Hardware abhängig.

Im darauf folgenden Datenaufnahmeschritt werden an jeder vorher definierten Position die Sensordaten zusammen mit den aktuellen Gelenkwinkeln der Aktoren aufgenommen. Diese Daten werden dann an den Optimierer übergeben.

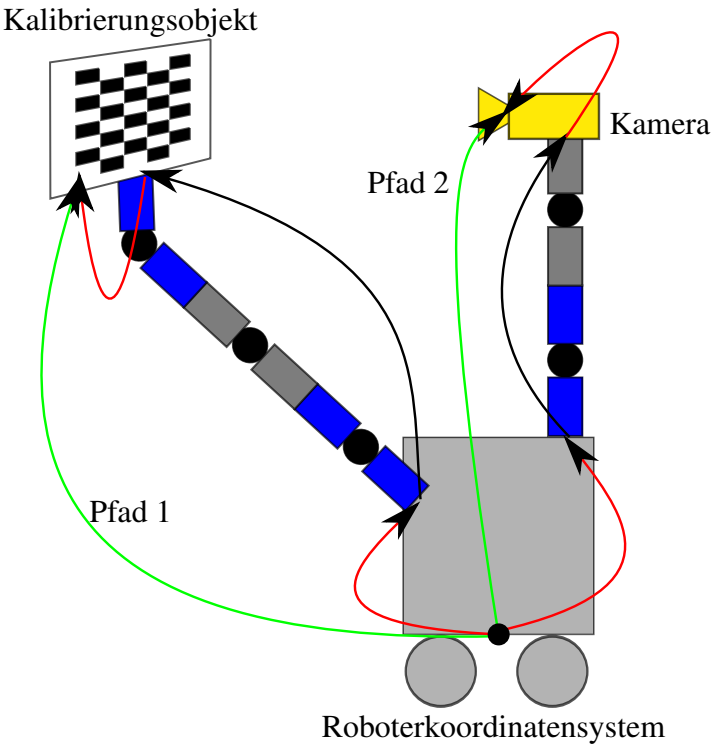

Abbildung 2.4: Transformationspfade des Care-O-bot®

Der Optimierer basiert auf dem Levenberg-Marquardt Verfahren[14, Abschnitt 1]. Hierbei werden für den Care-O-bot® drei Durchläufe des Optimiers genutzt, wobei in jedem Schritt weitere Parameter zur Optimierung freigegeben werden. Dadurch können lokale Minima vermieden werden.

Zur Bestimmung des aktuellen Fehlers werden die während der Datenaufnahme gespeicherten Gelenkwerte in ein Modell des Roboters eingesetzt. Dazu müssen die DH-Parameter der Aktoren sowie alle anderen Transformationen zumindest näherungsweise bekannt sein. Durch das Modell können dann die markanten Punkte des Kalibrierobjektes und die Position der Sensoren im Raum berechnet werden. Die modellierten Punkte lassen sich anschließend über die Sensorparameter in simulierte Messwerte umrechnen. Der Fehler, der durch Modifikationen an einzelnen Transformationen am Modell minimiert werden soll, ist die Differenz der berechneten und der gemessenen Messwerte.[15] Die Fehlerberechnung wird außerdem genutzt, um die Jacobimatrix für die Optimierung zu berechnen. Dazu werden alle vorhandenen Parameter einzeln

variiert um den Einfluss auf den Fehler zu berechnen. Im nächsten Schritt werden die Parameter dann gezielt in die richtige Richtung geändert [12].

2.4 Framework

Das Robot Operating System (ROS) ist ein Framework, das als Grundlage für Forschungs- und Entwicklungsroboter dienen soll. ROS wird als Open-Source Software von Willow-Garage sowie einigen weiteren Einrichtungen entwickelt. Das Hauptziel bei der Entwicklung von ROS ist es, ein Framework zu schaffen, in dem geschriebener Code nicht an Hardware gebunden ist, sondern wiederverwendet werden kann. Dazu wird die Software für den Roboter in einzelne „Nodes“ aufgeteilt, die jeweils eigene Aufgaben erfüllen.
Nodes werden zu Packages zusammengefasst, die einen größeren Funktionsblock einer Anwendung darstellen. Beispielhaft hierfür ist das Package `cob_calibration_executive`, mit dem alle Bewegungen, die der Roboter während der Kalibrierung macht, gesteuert werden.
Alle für eine Anwendung notwendigen neuen Funktionsblöcke werden in Stacks zusammengefasst.

Die Änderungen und Erweiterungen, die in dieser Bachelorthesis gemacht werden, betreffen hauptsächlich das Stack `cob_calibration`, in dem alle Funktionen für die Kalibrierung abgelegt sind.

Die Kommunikation der Nodes kann durch die Systeme ROS-Services, ROS-Topics oder ROS-Parameters erfolgen. Die drei Schnittstellen zwischen den Nodes unterscheiden sich vorallem durch die Funktion der beteiligten Kommunikationspartner.

ROS Services Durch die „Services“ können zwei Nodes direkt miteinander kommunizieren. Hierbei sendet der Client-Node eine Anfrage an den Server-Node, in der bestimmte Parameter übergeben werden. Der Server-Node antwortet auf dem gleichen Weg.

ROS Topics In den „Topics“ können beliebig viele Nodes Nachrichten hinterlassen. Die jeweils aktuellste Nachricht kann dann wiederum von beliebig vielen Nodes gelesen werden. Die Nachrichten enthalten keine Informationen, von wem sie gesendet werden, also gibt es für den Empfänger keine Möglichkeit, zu antworten oder zu quittieren. Dieses Verfahren wird zum Beispiel beim Verbreiten von Sensordaten eingesetzt.

ROS Parameters Der „Parameter Server“ wird für die Speicherung von Daten verwendet. Hier können Daten hinterlegt werden, die immer den gleichen Wert haben und von mehreren Nodes benutzt werden. Ein Beispiel hierfür sind die Kameraparameter oder die Beschreibung des Roboters.

Die Schnittstellen sind dabei so entwickelt, dass es problemlos möglich ist, die Nodes auf unterschiedlichen vernetzten Computern auszuführen. Um geschriebenen Code auf anderen Robotern zu verwenden, muss also lediglich dafür gesorgt werden, dass alle Kommunikationsschnittstellen vorhanden sind und richtig angewandt werden. Durch die modulare Struktur von ROS können viele vorhandene Lösungen für die eigenen Aufgaben genutzt und angepasst werden. Außerdem bietet ROS eine Simulationsumgebung und eine nahtlose Anbindung an andere Softwarepakete wie OpenCV.

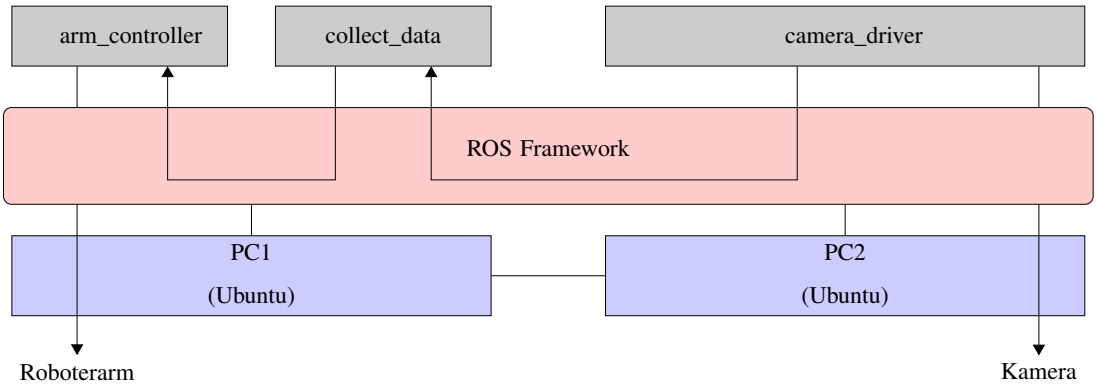

Abbildung 2.5: ROS Architektur nach[6]

In Abbildung 2.5 ist der Ablauf der Datenaufnahme schematisch dargestellt. Der ROS-Node `collect_data` schickt über ROS eine Nachricht an den Node

`arm_controller` der den am PC1 angeschlossenen Roboterarm steuert. Außerdem werden die Daten der am PC2 angeschlossenen Kamera vom `camera_driver` verarbeitet und über ein ROS-Topic an den `collect_data` Node weitergegeben.

Während der Bearbeitungszeit der Thesis wurde auf allen Computern die Version „Electric Emys“ eingesetzt.

3 Problemstellung und Entwicklung einer Lösung

Die Aufgabe, das vorhandene Kalibrierungsverfahren an alle Care-O-bot® der dritten Generation anzupassen und zu generalisieren erfordert eine Analyse des bestehenden Verfahrens. Mit dem Ansatz, für einen neuen Roboter neue Konfigurationsdateien mit fest einprogrammierten Werten konnte keine genaue Kalibrierung berechnet werden. Die vorhandenen Probleme waren in den Bereichen der Datenaufnahme sowie der kinematischen Kalibrierung des Roboters angesiedelt. Außerdem sollten Lösungen gefunden werden, die Montagepositionen der Laserscanner und das Tablett zu Kalibrieren.

3.1 Vorbereitung der Kalibrierung

Vor der Kalibrierung müssen die Gelenke des Torsos und des Arms in eine definierte „Home“-Position gebracht werden. Dazu müssen diese Positionen manuell angefahren und anhand von Markierungen an den Gelenken überprüft werden. Danach muss aus den aktuellen Werten der Gelenke und den eingestellten Werten manuell ein neuer Referenzwert berechnet und eingegeben werden. Fehler in diesem Schritt wirken sich auf die gesamte nachfolgende Kalibrierung aus und lassen sich nicht beheben. Das

Fehlerpotential, das durch die manuelle Verarbeitung entsteht, kann durch eine automatisierte Lösung gesenkt werden. Dadurch wird die Kalibrierung vereinfacht und beschleunigt.

3.2 Datenaufnahme

Zur Datenaufnahme werden Armpositionen mit den dazugehörigen Torsostellungen in einer Datei abgelegt. Diese Datei wird anhand vorher festgeleter kartesischer Koordinaten berechnet. Durch die unterschiedlichen Abmessungen der eingesetzten Arme ist es dem LWA nicht möglich, alle dieser für den LBR3 festgelegten Positionen zu erreichen. Desweiteren ist eine Berechnung der Positionen für andere Roboter, die sich wie der rob@work® durch den gesamten Aufbau vom Care-O-bot® unterscheiden, nicht möglich, da dessen Kameras und Aktoren einen vollkommen anderen Sicht- und Wirkungsbereich haben. Es werden nicht genügend Daten für die spätere Optimierung aufgenommen, was die Genauigkeit der Kalibrierung reduziert. Außerdem mussten für die Kamerakalibrierung und die kinematische Kalibrierung des Care-O-bot® zweimal nacheinander Daten von denselben Positionen des Arms und des Torsos aufgenommen werden.

3.2.1 Bestimmung der Sample-Positionen

Eine Lösung für die unterschiedlichen Hardwarekonfigurationen ist die Erstellung neuer Positionsdaten entweder durch manuelles Einlernen, automatisches Einlernen oder berechnen und überprüfen in Simulation.

Manuelles Einlernen Zum manuellen Einlernen der Samplepositionen werden die Aktoren des Roboters von einem Bediener in Positionen gefahren, an denen das am Arm montierte Kalibrierobjekt in den Kameras sichtbar ist. Anschließend

werden die zu den Positionen gehörenden Gelenkwinkel für spätere Kalibrierungen gespeichert. Das manuelle Einlernen der Positionen ist für jeden Roboter durchführbar, aber sehr aufwendig.

Automatisches Einlernen Hierbei wird ausgehend von der optischen Achse einer Kamera ein spiralförmiger Pfad fesgelegt, der vom Roboterarm abgefahren und regelmäßig auf die Sichtbarkeit des Kalibrierobjekts überprüft wird. Hierdurch wird ein sichtbares Feld abgesteckt und gleichzeitig Samples für die Optimierung aufgenommen.

Ein Vorteil dieser Lösung ist, dass zur Berechnung der Samplepositonen nur ein grobes Modell der Lage der optischen Achsen benötigt wird, das aus den ungenauen CAD-Daten berechnet werden kann. Aus diesem Modell kann dann der Ursprung des Pfades bestimmt werden.

Problematisch ist, dass Kollisionen mit dem Roboter oder der Umwelt erst bei der Bewegung des Arms erkannt und durch einen Abbruch der Datenaufnahme verhindert werden. Eine kollisionsfreie geplante Bewegung kann wegen der zu ungenauen Kalibrierung noch nicht bestimmt werden.

Berechnen und Überprüfen in Simulation Es werden im Voraus Positionen anhand einer Strategie berechnet und abgespeichert.

Der Vorteil bei der Berechnung und Speicherung der Gelenkwinkel in der Simulation ist, dass weniger ungültige Samples angefahren werden und damit der Zeitaufwand der Datenaufnahme reduziert wird. Außerdem können alle Trajektorien des Arms und des Torsos in der Simulation auf Kollisionen überprüft werden. Eine rechnerische Überprüfung auf Kollisionen ist wegen der noch nicht erfolgten Kalibrierung nicht zuverlässig möglich.

Ein Nachteil ist, dass Modelle zur Strategie der Berechnung erzeugt werden müssen, die genug Toleranz für Fehler zulassen, um die nicht erfolgte Kalibrierung auszugleichen.

Die Lösung, Positionen durch manuelles Einlernen zu bestimmen, wurde verworfen, da diese für jeden neuen Roboter zeitaufwendige Schritte bedeutet.

Der Unterschied der beiden anderen ist der Zeitpunkt, wann die Berechnung der Gelenkwinkel ausgeführt wird. Während bei dem Abfahren eines Pfades die Gelenkwinkel unmittelbar vor der Bewegung berechnet werden, werden sie bei der Simulation schon vorher berechnet.

Die Entscheidung fiel wegen der erheblich sichereren Durchführung für die Berechnung der Samples, die durch manuelle Überprüfung bestätigt werden können.

3.2.2 Doppelte Datenaufnahme vermeiden

Die doppelte Datenaufnahme war notwendig, da der Optimierer für die kinematische Kalibrierung bereits verarbeitete, also entzerrte Bilder des Kalibrierungsobjekts und die Projektionsmatrix der Kameras benötigt, um den Fehler zu bestimmen. Dazu wurden nur die Daten des zweiten Aufnahmeschritts für die kinematische Kalibrierung des Roboters verwendet. Für die Aufnahme wurden unter anderem die Nachrichten, die die Kameraparameter und die Kamerabilder enthalten, aufgezeichnet und in einer Datei gespeichert. Die Entzerrung der Bilddaten kann aber auch nach den angegebenen Formeln[6] erst bei der Optimierung erfolgen. Genauso können die Kameraparameter auch nachträglich aus der entsprechenden Kalibrierungsdatei gelesen werden. Durch diese Modifikationen können alle für die Kalibrierung notwendigen Daten in einem Aufnahmeschritt erfasst werden.

Durch den fehlenden Aufnahmeschritt kann der Zeitaufwand der Datenaufnahme und damit der gesamten Kalibrierung reduziert werden. Weil Armbewegungen ohne Kollisionsberechnung zu Schäden an der Hardware und der Umwelt führen können, kann durch die geringere Anzahl an Bewegungen außerdem die Sicherheit der Kalibrierung erhöht werden.

[6] Siehe Kapitel 2.3.1

3.3 Optimierung

Für die Optimierung wird aus den Konfigurationsdateien ein Modell des zu kalibrierenden Roboters erstellt. Dafür müssen die DH-Parameter der Aktoren sowie alle anderen Transformationen bekannt sein. Außerdem muss zu jeder Transformation angegeben werden, ob sie kalibrierbar oder festgelegt ist.

3.3.1 Doppelte Modellierung des Roboters

Da das ROS-Modell eines Roboters nicht nur zur Berechnung der Transformationen verwendet wird, sondern für die Visualisierung auch genaue Informationen über die Lage eines Gelenks benötigt, sind keine DH-Parameter sondern genaue Transformationen mit Rotations- und Translationsvektor angegeben. Die für die Kalibrierung benötigten DH-Parameter wurden bisher manuell berechnet und in die `system.yaml` eingegeben. Die Beschränkungen der DH-Parameter sind dadurch auch am Roboter vorhanden. Außerdem gibt es nicht für alle am Care-O-bot® eingesetzten Komponenten ausreichend genaue DH-Parameter. Die bisher manuell ausgeführte Berechnung der DH-Parameter soll deswegen automatisiert oder ersetzt werden.

Dazu bieten sich die automatische Berechnung der DH-Parameter, die Berechnung der Vorwärtskinematik durch den entsprechenden ROS-Service oder das Abspeichern der Transformationen während der Datenaufnahme an:

Berechnung der DH-Parameter Um die Kalibrierung auch für andere Roboter zu nutzen, besteht die Möglichkeit, die Parameter automatisiert berechnen zu lassen.

Der Vorteil dieser Methode liegt in der einfachen Implementierung. Durch die Berechnung der Parameter wird eine neue Konfigurationsdatei erzeugt, mit der der Roboter kalibriert werden kann.

Die Nachteile dieser Methode liegen in der Berechnung der Vorwärtskinematik während der Optimierung sowie in der weiteren Verwendung der DH-Parameter. Dadurch werden alle Nachteile, die sich durch die vereinfachte Beschreibung einer Transformation durch DH-Parameter ergeben, auf den Roboter übertragen. Diese Vereinfachungen lassen nur reine Rotations- und Translationsgelenke zu.

Die Berechnung der Vorwärtskinematik während der Optimierung wirkt sich auf die Laufzeit der Kalibrierung aus. Die Umrechnung der aufgenommenen Gelenkwinkel mit den gegebenen DH-Parametern erfolgt etwa 250.000 mal. Durch den häufigen Funktionsaufruf wirken sich Änderungen in der Rechenzeit stark auf die Gesamtrechenzeit aus.

Berechnen der Vorwärtskinematik mit dem ROS-Service Eine Alternative hierzu ist die Nutzung des ROS-Services zur Berechnung der Vorwärtskinematik während der Optimierung. Dazu werden weiterhin die Gelenkwinkel erfasst und abgelegt. Es werden die Gelenkwinkel an den ROS-Node, der für die Berechnungen der Vorwärtskinematik zuständig ist, übergeben und die Transformation berechnet.

Dadurch müssen in den Konfigurationsdateien keine DH-Parameter abgelegt sein und es lassen sich alle modellierten Arten von Aktoren berechnen.

Problematisch hierbei ist, dass es für jeden Aktor einen Service geben muss, der die Vorwärtskinematik berechnen kann. Für den Care-O-bot® gibt es diesen Service nur für den Arm. Für Tests wurde dieser Node angepasst, um auch Lösungen für den Torso zu berechnen. Dabei konnte festgestellt werden, dass der Zeitbedarf der Kalibrierung ansteigt. Die Berechnung der Transformationen erfolgt hierbei wie mit DH-Parametern während der Optimierung. Durch die Nutzung der ROS-Services zur Berechnung der Transformationen und den damit verbundenen zusätzlichen Funktionsaufrufen erhöht sich der Overhead und damit auch die Gesamtrechenzeit.

Abspeichern der Transformationen während der Datenaufnahme Diese Möglichkeit erfordert Änderungen sowohl an der Datenaufnahme als auch an der Optimierung. Anstatt Gelenkwinkel aufzunehmen, werden hierbei die Transformationen eines Aktors während der Datenaufnahme berechnet. An dieser Stelle muss die Berechnung nur einmal pro Sample, also insgesamt etwa 30 bis 50 mal erfolgen. Es werden 250.000 Durchläuf für die Berechnung der Kovarianz- und der Jacobi-Matrix benötigt. Bei dieser Möglichkeit werden hierzu nicht die Gelenkwinkel, sondern die Rotations- und Translationsvektorelemente variiert.

Der Vorteil dieser Methode ist neben der verkürzten Rechenzeit, dass für alle mit ROS verwendeten Roboter ein Modell vorhanden sein muss, das hier verwendet werden kann. Die Modellierung des Manipulators muss also nicht zusätzlich in DH-Parametern vorhanden sein. Außerdem bietet ROS ohne zusätzliche Nodes die Möglichkeit, die aktuellen Transformationen zu berechnen. Die Transformation kann also für jeden Roboter ohne zusätzliche Änderungen berechnet werden.

Dafür muss bereits bei der Datenaufnahme bekannt sein, welche Transformationen für die Optimierung benötigt werden. Das bedeutet, dass die in Abbildung 2.4 dargestellten Pfade bereits genau definiert sein müssen und nur wenige Änderungen nachträglich gemacht werden können. Bei der Aufnahme von Gelenkwinkeln ist diese Festlegung erst bei der Kalibrierung von Bedeutung. Wenn nachträglich Änderungen an den angegebenen Pfaden gemacht werden, müssen erneut Daten aufgenommen werden. Bei den anderen Methoden ist eine Änderung nach der Datenaufnahme möglich.

Bei dieser Methode ändert sich die Dateistruktur, müssen viele Teile zur Berechnung des Modells des Optimierers angepasst oder neu entwickelt werden.

Da ein möglichst universell einsetzbares Verfahren implementiert werden soll, kann nicht davon ausgegangen werden, dass alle zu kalibrierenden Roboter die durch die weitere Verwendung von DH-Parametern entstehenden Anforderungen erfüllen.

Trotz dem erheblichen Mehraufwand wird die dritte Alternative umgesetzt. Verglichen mit den anderen Lösungen ist diese, durch die Aufhebung der Beschränkungen der DH-Parameter und den Verzicht auf zusätzliche Services, die universellste. Zudem müssen für andere neue Funktionen Änderungen am Modell des Optimierers vorgenommen werden, die die gleichen Funktionen betreffen.

3.3.2 Verkettung von Aktoren

Durch das Format der Parameterdateien des bisherigen Kalibrieralgorithmus konnten nicht alle Eigenschaften des Care-O-bot® abgebildet werden. Dazu gehören Aktoren, die an anderen Aktoren montiert sind. Beim Care-O-bot® führt das zu Einschränkungen, die die Kopfachse betreffen. Beim Care-O-bot® ist der Kopf mit einer Achse an einer unbekannten Positon auf dem Torso montiert. Für die bisherige Kalibrierung wurde, um dies zu umgehen, die Kopfachse als statische Transformation angesehen. Dadurch wurde eine Kalibrierung in einem bestimmten Bereich ermöglicht. Zur Steigerung der Genauigkeit sollte die Kalibrierung den gesamten Arbeitsraum des Roboters abdecken.

Um dies zu ermöglichen, muss der Dateiaufbau der Konfigurationsdatei `sensors.yaml` und das Erzeugen des Modells für die Optimierung geändert werden.

3.4 Extrinsische Kalibrierung der Laserscanner

Für die Kalibrierung von Laserscannern gibt es je nach Ausführung der Scanner verschiedene Verfahren zur Kalibrierung. 3D Laserscanner, die zusätzlich zu den Entfernungsdaten auch Intensitätswerte der reflektierten Strahlen liefern, können wie Kameras behandelt werden. Die Intensitäten lassen sich zu einem Bild zusammensetzen auf dem, wie auf dem Kamerabild, das Kalibrierobjekt erkannt werden kann. Die 6DOF

Pose des Objekts lässt sich durch die gemessenen Entfernungen noch genauer berechnen.[10]

Da die am Care-O-bot® eingesetzten Laserscanner nur 2D Scanner sind, die keine Helligkeitswerte messen, muss ein anderes Verfahren gefunden werden, mit dem die Position der Laserscanner kalibriert werden kann. Die Kalibrierung der Montageposition besteht auch hier aus dem Erkennen eines Kalibrierobjektes und der Optimierung der Parameter.

4 Umsetzung und Ablauf der Kalibrierung

Der neu festgelegte Ablauf der Kalibrierung lässt sich in die Einrichtung und Durchführung unterteilen. Die Einrichtung erfordert Kenntnisse über die Funktionsweise des Kalibrierablaufs, muss aber nur einmalig pro Roboter ausgeführt werden.

4.1 Einrichtung der Kalibrierung

Die Einrichtung der Kalibrierung kann vollständig in Simulation erfolgen und umfasst nur Änderungen der Dateien im Package `cob_calibration_config`.

4.1.1 Angeben der zu kalibrierenden Kameras und des Kalibrierobjekts

Ein weiterer neuer Schritt bei der Einrichtung der Kalibrierung auf einem neuen Roboter ist das Festlegen der zu kalibrierenden Kameras. Hierzu werden in der Datei `cameras.yaml` eine Referenzkamera und alle weiteren Kameras angegeben. Dazu wird für jede Kamera der Name des Sensorkoordinatensystems, das ROS-Topic,

auf dem die Bildinformationen verteilt werden und der Pfad zu den Kalibrierungsdateien angegeben. Außerdem muss das eingesetzte Kalibrierungsmuster angegeben werden.

Diese Einstellungen waren bei dem ursprünglichen Kalibrierverfahren an mehreren Stellen verteilt, was das Anpassen an neue Roboter erschwert hat.

4.1.2 Ausgangspunkt der Kalibrierung

Als erster Schritt muss die neue Konfigurationsdatei `calibration_seed.yaml` angelegt werden, in der eine Position des Kalibrierobjekts mittig im Sichtfeld der Kameras in Gelenkwinkeln der Aktoren angegeben wird. Diese Position ist der Ausgangspunkt der Kalibrierung und sollte genutzt werden, um den Endeffektor des Roboters durch das Kalibrierobjekt zu ersetzen. Die angegebene Position wird als Anfangspunkt für die Suche der Samplepositionen verwendet.

4.1.3 Berechnung der Samplepositionen

Die Samplepositionen werden automatisch berechnet. Dazu muss der Ausgangspunkt der Kalibrierung bekannt sein. Bei der Berechnung wird ein Bereich im Arbeitsraum der zu kalibrierenden Aktorkette festgelegt. Danach werden für einzelne Positionen in diesem Raum Berechnungen zur Sichtbarkeit und zur Erreichbarkeit der Positionen ausgeführt. Es wird mit einem allgemeinen Kameramodell, das nur den Öffnungswinkel des Bildes benötigt und einem Modell der Bewegung der optischen Achse berechnet, ob die Position sichtbar ist. Im Anschluss daran wird gegebenenfalls die inverse Kinematik des Torsos und des Arms berechnet. Die berechneten Gelenkwinkel werden in der Datei `calibration_positions.yaml` gespeichert. Die berechneten Positionen können anschließend durch Simulation überprüft und manuell um weitere Positionen erweitert werden. Im Gegensatz zur bisherigen Kalibrierung lassen sich

durch die neue Berechnung schneller Gelenkwinkel für Roboter mit unterschiedlichen Arbeitsräumen festlegen.

4.1.4 Festlegung der Kinematischen Pfade

Für jeden weiteren Schritt müssen Informationen, welche kinematischen Ketten an der Kalibrierung beteiligt sind, vorhanden sein. Dazu wird wie bei dem ursprünglichen Verfahren eine Konfigurationsdatei verwendet. Um auch hintereinander angeordnete Aktoren in der Kalibrierung zu berücksichtigen, wird die Struktur angepasst. Die neue `sensors.yaml` setzt sich aus den dreiBlöcken `chains`, `sensor_chains` und `camera_chains` zusammen.

Der neue Block `chains` beschreibt alle vorkommenden Aktorketten. Dazu werden die Transformation zum Ursprung des Aktors, das Ursprungs- und das Endkoordinatensystem sowie Transformationen nach dem Aktor angegeben. Zusätzlich bekommt jede Kette einen eindeutigen Namen.

Beim Care-O-bot® werden zwei von drei Ketten angegeben. Da die Kopfachse während der Kalibrierung nicht bewegt wird, kann diese Kette als feste Transformation angenommen werden. Der Aufbau dieser Ketten wird in Abbildung 4.1 dargestellt. Die gesamte dargestellte Kette ist vom Typ `sensor_chain` und zusammengesetzt aus der grün markierten `chain` und der schwarzen `after_chain`. Die `chain` setzt sich aus den zwei festen Transformationen `before_chain` und `after_chain` und der Aktorkette zusammen. Eine Aktorkette muss nicht nur einen Aktor, wie zum Beispiel den Arm beschreiben, sondern kann auch mehrere Aktoren zusammenfassen. Dabei muss darauf geachtet werden, dass Transformationen, die durch die Kette beschrieben werden, nicht kalibriert werden können.

Der Block `sensor_chains` beschreibt die gesamte Transformation vom baselink bis zum Kalibrierobjekt. Dazu wird eine Liste der `chains` sowie die Transformationen vor und nach den darin angegebenen Aktorketten angegeben. Mit dem gleichen

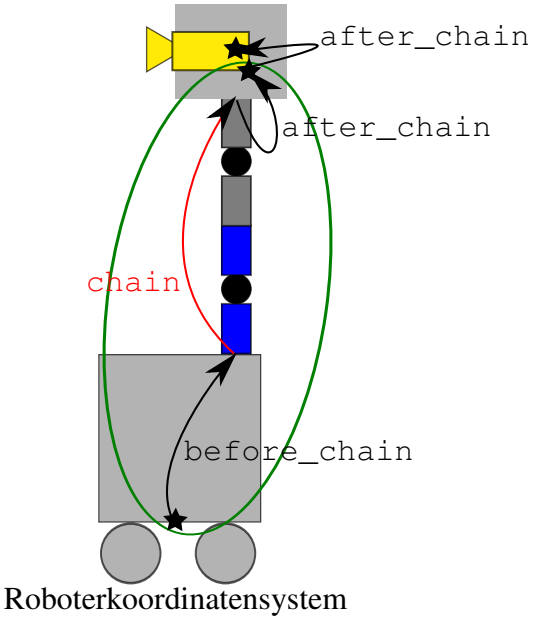

Abbildung 4.1: Pfade der Konfigurationsdatei

Aufbau werden im Block `camera_chains` die Transformationen zu den eingesetzten Kameras angegeben.

Aus der erzeugten `sensors.yaml` kann durch die angewandten Änderungen die Datei `system.yaml` berechnet werden. Hierzu werden die benötigten festen Transformationen von einem Skript ausgelesen und umgerechnet. Diese Transformationen werden in die gesuchte Konfigurationsdatei eingetragen. Zusätzlich wird eine Vorlage für die Konfigurationsdateien zum Festlegen der Optimierungsschritte erzeugt.

4.1.5 Einstellen der freien Optimierungsparameter

Um lokale Minima im Ergebnis der Kalibrierung zu vermeiden, wird die Optimierung der Montagepositionen mehrmals ausgeführt. Hierbei werden bei jedem Schritt weitere Parameter zur Optimierung freigegeben. Welche Parameter in welchem Schritt freigegeben werden, kann durch die Konfigurationsdateien `free_x.yaml` eingestellt werden. `x` gibt dabei den Kalibrierungsdurchlauf an. Der Aufbau der Datei ist mit dem der `system.yaml` identisch. Die Zahlenwerte der Transformationen sind durch `0`

und 1 ersetzt, um anzugeben ob der Parameter optimiert werden soll (1) oder als fest angenommen wird (0).

Eine erfolgreiche Kalibrierung mit drei Optimierungszyklen kann erreicht werden, wenn im ersten Durchlauf nur die Montageposition des Kalibrierobjekts am Arm, im zweiten Durchlauf die Position der Kameras auf dem Kopf und im letzten Zyklus alle anderen gesuchten Transformationen zur Optimierung freigegeben werden.

4.2 Durchführen der Kalibrierung

Die Durchführung der Kalibrierung wurde durch die vorgenommenen Änderungen auf drei Schritte reduziert. Die drei Schritte gliedern sich in eine Datenaufnahme und zwei Berechnungsschritte.

4.2.1 Datenaufnahme

Vor der Datenaufnahme muss die „Home“-Position aller Gelenke der Aktoren festgelegt werden. Dazu wird die „Home“-Position am Aktor manuell angefahren und die Gelenkwinkel in dieser Position als neue Referenz gespeichert. Dazu werden die Gelenkwinkel ausgelesen und mit der aktuellen Referenz verrechnet. Dieser Schritt ist durch ein Skript automatisiert, wodurch die Fehleranfälligkeit der Kalibrierung gesenkt wird.

Zur Datenaufnahme muss das in der Konfigurationsdatei angegebene Kalibrierobjekt montiert werden. Dazu muss sich der Arm in einer der berechneten Sampleposition oder der Ausgangspositon der Kalibrierung befinden. Durch die Berechnung der Samplepositionen wird sichergestellt, dass alle Aktorstellungen ähnlich zu der Ausgangsposition sind. Dadurch werden Umkonfigurationen des Aktors während der Datenaufnahme vermieden. Durch die Montage des Kalibrierobjekts in der Ausgangsposition

findet auch zu Beginn und zum Ende der Kalibrierung keine Umkonfiguration mit dem angebauten Kalibrierobjekt statt und Kollisionen werden vermieden.

Zur Datenaufnahme werden die für die Kalbrierung benötigten Transformationen aus der `sensors.yaml` eingelesen. Während der Datenaufnahme werden die berechneten Samplepositionen angefahren. Im Anschluss wird überprüft, ob das Kalibrierobjekt in allen oder nur in der Referenzkamera sichtbar ist. Da für die kinematische Kalibrierung nur die Informationen der Referenzkamera verwendet werden, wird bei einfacher Sichtbarkeit ein gültiges Sample zur kinematischen Kalibrierung aufgenommen. Bei Sichtbarkeit an allen Kameras werden zusätzlich Bilder für die Kamerakalibrierung aufgenommen.

Für die Kamerakalibrierung werden keine kinematischen Informationen über den Roboter benötigt. Daher ist es für ein Sample zur Kamerakalibrierung ausreichend, dem Kalibrierer die Bilder aller zu kalibrierenden Kameras zur Verfügung zu stellen.

Die kinematische Kalibrierung benötigt zusätzlich zu den Bildinformationen der Referenzkamera die Transformationen der Aktoren. Dazu wird nach den Änderungen anstatt der Gelenkwinkel die Transformation direkt als Translationsvektor und den Roll-Pitch-Yaw Winkeln in ein Bagfile, ein ROS eigenes Datenformat abgespeichert. Nach der Datenaufnahme sollten mindestens 20 gültige Samples für beide Kalibrierungen vorhanden sein.

Die Datenaufnahme wird von dem ROS-Node `collect_robot_calibration_data` gesteuert. Alle am Roboter verwendeten Aktoren werden in die gewünschte Sampleposition gebracht. Danach wird mit einem ROS-Service Aufruf der ROS-Node `visibility_checker` angewiesen, die Sichtbarkeit des Kalibrierobjekts in jeder Kamera zu überprüfen. In der Serviceantwort ist angegeben, ob das Kalibrierobjekt in allen oder nur in der Referenzkamera zu sehen ist. Wenn das Objekt nur in der Referenzkamera sichtbar ist, wird nur ein Datensatz für die kinematische Kalibrierung aufgenommen. Hierzu wird der ROS-Service `/collect_data/capture` aufgerufen. Der Empfänger dieses Aufrufs ist der Node `collect_data`. Dieser berech-

net alle Transformationen, die in der `sensors.yaml` als Aktorketten eingetragen sind, anhand des aktuellen Roboterzustandes. Dazu wird eine Nachricht vom Typ `ChainMeasurement` erzeugt. Darin wird, wie in Abbildung 4.2 dargestellt, die Bezeichnung der Kette sowie der Translationsvektor und die Roll-Pitch-Yaw Winkel der Transformation gespeichert. Zusätzlich werden die Pixelkoordinaten der Ecken des

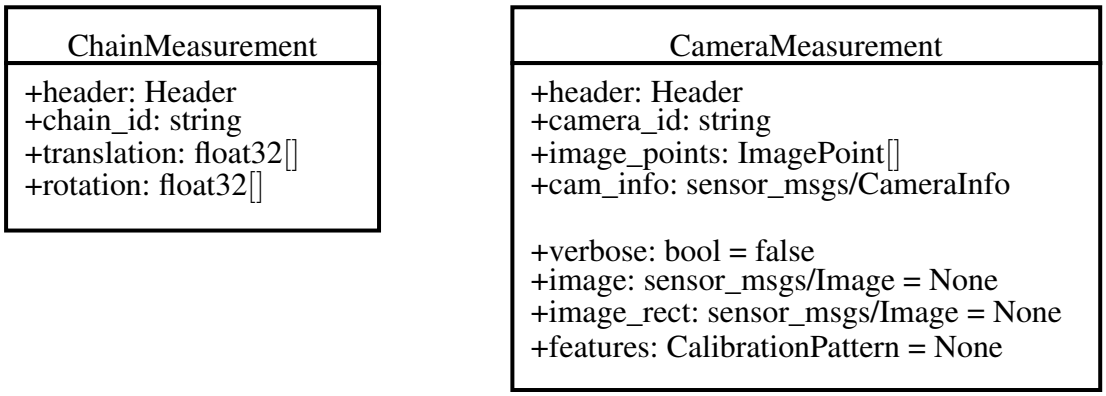

Abbildung 4.2: Die Messages `CameraMeasurement` und `ChainMeasurement`

Schachbretts aus dem Kamerabild der Referenzkamera extrahiert. Die Koordinaten werden anschließend zusammen mit der ID der Kamera und einem Zeitstempel zu einer Nachricht vom Typ `CameraMeasurement` zusammengefügt. Es besteht die Möglichkeit, weitere Informationen in der Nachricht zu speichern, um Fehler bei der Datenaufnahme zu erkennen. Zu den zusätzlichen Informationen gehören ein Bild und die Koordinaten des Kalibrierobjekts in dessen Koordinatensystem. Die zwei in Abbildung 4.2 dargestellten Messagesbilden einen Datensatz und werden zu einer Nachricht auf dem ROS-Topic `/robot_measurement` zusammengefasst. Alle Nachrichten, die auf diesem Topic veröffentlicht werden, werden in einem Bagfile gespeichert und bilden die Datengrundlage zur späteren Kalibrierung.

Wenn das Kalibrierobjekt in allen Kameras sichtbar ist, wird zusätzlich ein Datensatz für die Kamerakalibrierung aufgenommen. Dazu wird der ROS-Service `/image_capture/capture` aufgerufen, der im ROS-Node `image_capture` die Speicherung aller Kamerabilder als `*.jpg` veranlasst.

Alle Bilddaten, die in diesem Schritt verarbeitet oder abgespeichert werden, sind die unverabeiteten Rohbilder, bei denen noch keine Linseneffekte herausgerechnet sind.

Nachdem die Serviceaufrufe beantwortet sind, wird an der nächsten Kalibrierungsposition versucht, einen weiteren Datensatz aufzunehmen.

4.2.2 Berechnung der Kameraparameter

Aus den aufgenommen Daten werden im Anschluss die Kameraparameter aller Kameras sowie deren Transformationen zur Referenzkamera berechnet. Die wichtigste Neuerung der Kamerakalibrierung ist, dass nicht nur die Transformation des Stereokamerapaars sondern auch die Transformation zu den anderen Kameras berechnet wird. Dafür wurde ein neues Kamerareferenzkoordinatensystem für den Roboter eingeführt. Das Koordinatensystem der Referenzkamera ist identisch mit dem neuen Koordinatensystem.

Im ersten Berechnungsschritt werden alle Transformationen im Kopf des Roboters berechnet. Durch das zusätzliche Koordinatensystem muss im anschließenden kinematischen Kalibrierungsschritt nur noch die Montageposition des Kopfes und nicht mehr jeder einzelnen Kamera berechnet werden. Im nächsten Schritt wird zur Berechnung der Montagepostion der Kameras die Stellung des Aktors berücksichtigt. Die daraus resultierende Ungenauigkeit wird in dem neuen Berechnungsverfahren umgangen, wodurch eine genauere Kameraposition berechnet werden kann.

Für die Berechnung werden Stereokamerapaare gebildet, die jeweils aus der Referenzkamera und einer der anderen Kameras bestehen. Für dieses Kamerapaar werden die vorher aufgenommenen Bilder geladen und verarbeitet. Mit den daraus gewonnenen Pixelkoordinaten der wichtigen Punkte des Kalibrierobjekts wird der Stereokamerakalibrieralgorithmus von OpenCV aufgerufen, der die Kameraparameter und die Transformationen zwischen den Kameras berechnet.

Die berechneten Transformationen werden anschließend in die Kalibrierungsdatei eingetragen. Außerdem wird für jede Kamera eine Kalibrierungsdatei erstellt, die die Kameraparameter enthält.

4.2.3 Kinematische Kalibrierung

Im letzten Berechnungsschritt werden die Montagepositionen der einzelnen Komponenten des Roboters optimiert. Dazu muss ein Modell des Roboters erzeugt werden, das alle aufgenommenen Daten abbilden kann. Das Modell wird durch die Verkettung von drei Klassen erzeugt. Durch die Änderungen an der Datenaufnahme und den neuen Konfigurationsdateien müssen weitreichende Änderungen an den Klassen dieses Modells durchgeführt werden. Die drei existierenden Klassen dienen zur Abbildung einer Kamera, einer Kette von Transformationen und dem Kalibrierobjekt. Die beiden Klassen zur Darstellung der Kamera und des Kalibrierobjektes werden zur Berechnung der Abweichung eines virtuellen Kalibrierobjektes und den gemessenen Daten verwendet.

In der Kameraklasse werden dafür die Koordinaten der Merkmale des virtuellen Kalibrierobjekts auf den modellierten Sensor projiziert. Dafür müssen die Brennweiten und Verschiebungswerte bekannnt sein, die vor den Änderungen in den Messdatensätzen angegeben waren. Da die Kameras beim angepassten Verfahren während der Datenaufnahme noch nicht kalibriert und somit die benötigten Parameter noch nicht bekannt sind, müssen die Werte aus der, nach der Kamerakalibrierung erstellten Kalibrierungsdatei geladen werden. Zusätzlich muss die Linsenverzerrung modelliert werden, die bei kalibrierten Kameras schon ausgerechnet wurde. Dazu werden die Linsenparameter aus der Kalibrierungsdatei gelesen. Auf die berechneten projizierten Punkte wird zur Berechnung des Fehlers die Linsenverzerrung angewandt. Dadurch werden die Pixelkoordinaten der Punkte des virtuellen Schachbretts erzeugt, die mit den gemessenen Pixelkoordinaten verglichen werden.

Die Klasse zur Berechnung des Fehlers am Kalibrierobjekt wurde nicht geändert. Zur Berechnung des Fehlers werden die Punkte des virtuellen Kalibrierobjektes mit den modellierten Punkten des an einer Kette von Transformationen angebrachten Kalibrierobjektes verglichen.

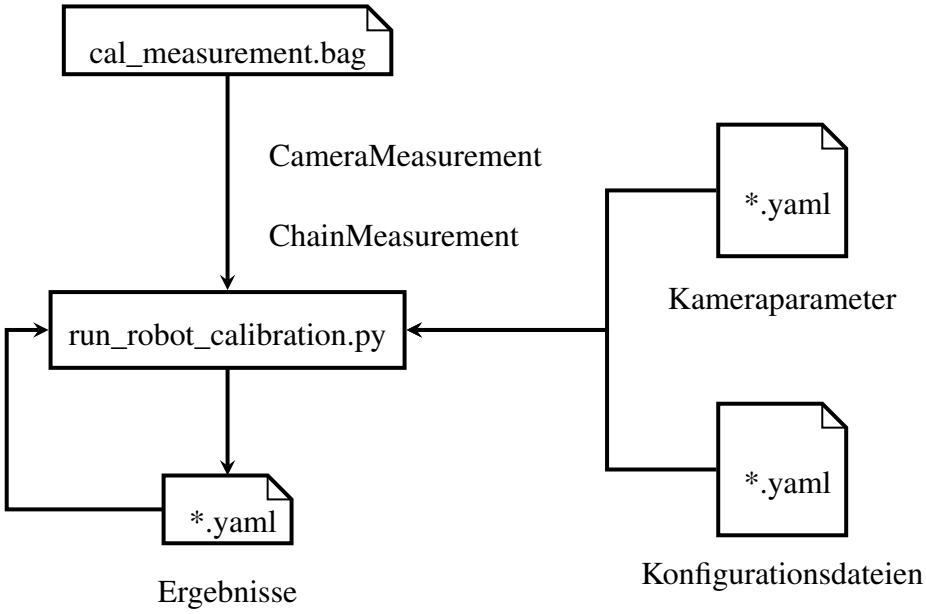

Abbildung 4.3: Kinematische Kalibrierung

Zur Berechnung der Transformationen zu den Sensoren gibt es eine Klasse, die unterteilt ist in die Berechnung der festen Transformationen und der variablen Aktortransformationen. Die Angabe der festen Transformationen erfolgt nach den Änderungen in dem für ROS typischen Format Translationsvektor und Roll-Pitch-Yaw Winkel. Vorher wird wie in [14, Abschnitt 3.3.2] ein Rotationsvektor angegeben, dessen Länge der Rotationswinkel ist. Durch die Änderung ist die Einrichtung eines neuen Roboters und die Berechnung der Rotationsmatrix im Modell vereinfacht worden. Das Modell der variablen Transformationen wurde von DH-Parametern zu gegebenen Transformationen im Format Translationsvektor und Roll-Pitch-Yaw Winkel geändert. Dadurch können neue Roboter einfacher in die Kalibrierung übernommen werden. Zusätzlich zu den geänderten Berechnungen der Transformationen der Ketten musste auch die Berechnung der Kovarianzmatrix geändert werden, um die neue Struktur zu unterstützen.

Nachdem die berechneten Transformationen in die Kalibrierungsdatei übernommen wurden, ist die Kalibrierung abgeschlossen.

4.3 Kalibrierung der Laserscanner

Die Kalibrierung der Laserscanner gliedert sich, wie die Kalibrierung der Kameras, in die intrinsische Kalibrierung und die Kalibrierung der Montagepositionen. Da die Laserscanner vom Hersteller ausreichend genau intrinsisch kalibriert werden, muss lediglich eine extrinsische Kalibrierung vorgenommen werden.

Die Laserscanner lassen sich durch den geringen Arbeitsraum des Armes im Sichtfeld der Laserscanner und dem Sichtfeld der Kameras nicht in den bisherigen Kalibrierablauf integrieren. Zur Kalibrierung wird also ein neuer Datenaufnahme- und ein Berechnungsschritt notwendig.

Zur Erkennnung des Kalibrierobjekts muss ein neues Verfahren entwickelt werden, das in der Lage ist das Kalibrierobjekt anhand eines 2D-Schnitts des Objekts zu erkennen. Dazu muss ein neues dreidimensionales Objekt benutzt werden, dessen Schnitt auf den Sensordaten erkannt werden kann. Die Sensordaten liegen in einer Nachricht vom Typ `LaserScan` vor, die die detektierten Punkte in einem Polarkoordinatensystem angibt[16]. Durch die Umrechnung in karthesische Koordinaten kann daraus ein Bild erzeugt werden, das von OpenCV weiterverarbeitet werden kann.

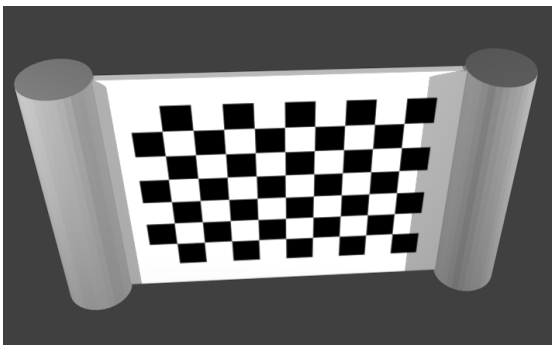

Abbildung 4.4: Kalibrierobjekt für die Laserscanner

Ein mögliches Kalibrierobjekt ist in Abbildung 4.4 dargestellt. Es besteht aus einem Schachbrett, an dessen Rändern Zylinder angebracht sind. Die Zylinder ergeben im 2D-Schnitt ein Kreissegment, das mithilfe einer Hough-Transformation [17] erkannt werden kann. Auf diese Weise können auch alle auftretenden Geraden erkannt werden. Da die Achsen der Zylinder des Kalibrierobjekts auf einer Ebene mit der Ebene des Schachbretts liegen, kann die Position des Kalibrierobjekts aus der Linie die den geringsten Fehler zu den Mittelpunkten der erkannten Kreise als Teil der Ebene des Kalibrierobjekts erkannt werden. Die erkannte Position liefert die Orientierung und den Abstand des Kalibrierobjekts zum Laserscanner.

Die Erkennung des Kalibrierobjekts funktioniert in der Simulation, muss aber noch stabilisiert und mit der Hardware überprüft werden.

Das Schachbrett am Kalibrierobjekt wird zur Erkennung durch die Kameras benötigt. Aus der erkannten 6DOF Pose der Kameras und der 2DOF Pose der Laserscanner kann mit mehreren dieser Datensätzen die Position der Laserscanner relativ zu den Kameras berechnet werden. Die Berechnung kann wie die kinematische Kalibrierung von einem Optimierer erfolgen. Die Umsetzung des Optimierers steht nach den Verbesserungen an der Objekterkennung noch aus.

5 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war es, die automatische Kalibrierung des Care-O-bot® anzupassen und zu generalisieren. Hierdurch sollte ein einheitliches Kalibrierverfahren entwickelt werden, mit dem alle Care-O-bot® und andere am IPA eingesetzte Serviceroboter kalibriert werden können. Die bereits vorhandene Lösung erzielt für den Care-O-bot®3-3 gute Ergebnisse, mit denen eine sichere Objekterkennung und Manipulation möglich ist. Diese Lösung konnte allerdings nicht auf andere Roboter angewandt werden.

Zur Generalisierung wurde das bisher am Care-O-bot®3-3 eingesetzte Verfahren analysiert und Optimierungsmöglichkeiten herausgearbeitet.

Die beiden Datenaufnahmeschritte zur Kamera- und kinematischen Kalibrierung des alten Verfahrens wurden zu einem Schritt zusammengefasst. Dafür werden zur kinematischen Kalibrierung die Merkmale eines Kalibrierobjekts auf den verzerrten Bildern der Kameras erkannt. Bei der Berechnung wird diese Verzerrung im Modell berücksichtigt. Die Aufnahme eines Datensatzes zur Kalibrierung mit unkalibrierten Kameras ist möglich und die Datenaufnahme zur Kamerakalibrierung entfällt.

Zur Berechnung der Kalibrierung wurden im Kalibrierverfahren DH-Parameter benötigt, die nicht für alle behandelten Roboter verfügbar waren. Somit konnte die Kalibrierung nur auf einem Roboter ausgeführt werden. Um Roboter ohne verfügbare DH-Parameter zu kalibrieren, wurde die Berechnung der Vorwärtskinematik angepasst, dass keine DH-Parameter mehr benötigt werden.

Die Implementierung erfolgt im ROS-Stack `cob_calibration`. Um neue Roboter einfacher in die Kalibrierung aufzunehmen, werden alle Konfigurationen im neuen ROS-Package `cob_calibration_config` zusammengefasst.

Anstatt festgelegter Positionen zur Datenaufnahme werden Samplepositionen für die Roboter berechnet. Dadurch können Roboter mit unterschiedlichen Aktoren in die Kalibrierung einbezogen werden.

Durch diese Änderungen wird das Hauptziel der Bachelorthesis, das Kalibrierungsverfahren für die Kameras und die beiden „relevanten Komponenten“[6] - Arm und Torso - anzupassen, um andere Roboter zu kalibrieren, erreicht.

Zusätzlich wurde ein Verfahren entwickelt, mit dem ein Kalibrierobjekt in den 2D Laserscan Daten erkannt werden kann. Zur Kalibrierung der Laserscanner zu den Kameras kann das Verfahren in zukünftigen Arbeiten verbessert und stabilisiert werden. Außerdem kann die mathematische Optimierung implementiert werden.

In Zukunft können weitere Komponenten des Care-O-bot® zum Beispiel die Montageposition des Tabletts, durch taktile Messungen oder angebrachte Marker kalibriert werden.

Literatur

[1] R.D. Schraft und H. Volz. *Serviceroboter: Innovative Technik in Dienstleistung und Versorgung*. Springer, 1996. ISBN: 9780387593593.

[2] tankpitstop. *The TankPitstop*. Englisch. tankpitstop. URL: (http://www.tankpitstop.nl/index.php?pid=8&lang=en).

[3] Fraunhofer Gesellschaft, Institut für Produktionstechnik und Automatisierung. URL: http://www.care-o-bot-research.org/care-o-bot-3/technical-data.

[4] Fraunhofer Gesellschaft, Institut für Produktionstechnik und Automatisierung. URL: http://www.care-o-bot.de/english/Cob3_Download.php.

[5] Fraunhofer Gesellschaft, Institut für Produktionstechnik und Automatisierung. URL: http://www.care-o-bot-research.org/care-o-bot-3/components.

[6] S.A. Haug. „Automatische Kalibrierung eines komplexen Serviceroboters“. Diplomarbeit. Institut für Automatisierungs- und Softwaretechnik Universität Stuttgart, 2012.

[7] *Camera calibration and 3D reconstruction*. Willow Garage. URL: http://opencv.willowgarage.com/documentation/python/calib3d_camera_calibration_and_3d_reconstruction.html#findchessboardcorners.

[8] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman, 2005. ISBN: 9780201543612.

[9] L. Sciavicco und B. Siciliano. *Modelling and Control of Robot Manipulators*. Advanced Textbooks in Control and Signal Processing Series. Springer-Verlag, 2000. ISBN: 9781852332211.

[10] V. Pradeep, K. Konolige und E. Berger. „Calibrating a multi-arm multi-sensor robot: A Bundle Adjustment Approach“. In: *International Symposium on Experimental Robotics (ISER)*. New Delhi, India, Dez. 2010. URL: http://www.willowgarage.com/sites/default/files/calibration.pdf.

[11] G. Bradski und A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Incorporated, Sep. 2008. ISBN: 9780596516130.

[12] D.A. Forsyth und J. Ponce. *Computer Vision: A Modern Approach*. Pearson, 2011. ISBN: 9780136085928.

[13] Z. Zhang. „A flexible new technique for camera calibration“. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.11 (2000), S. 1330–1334.

[14] Vijay Pradeep. URL: http://www.ros.org/wiki/pr2_calibration_estimation?distro=electric.

[15] P. Levi u. a. *Autonomous Mobile Systems 2012: 22. Fachgespräch Stuttgart, 26. Bis 28. September 2012*. Informatik Aktuell Series. Springer London, Limited, 2012. ISBN: 9783642322167.

[16] *sensor_msgs/LaserScan Message*. Englisch. Willow Garage. März 2013. URL: http://www.ros.org/doc/api/sensor_msgs/html/msg/LaserScan.html.

[17] B. Jähne. *Digitale Bildverarbeitung: und Bildgewinnung*. 7., neu bearbeitete Aufl. 2012. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-04952-1. URL: http://nbn-resolving.de/urn/resolver.pl?urn=10.1007/978-3-642-04952-1.