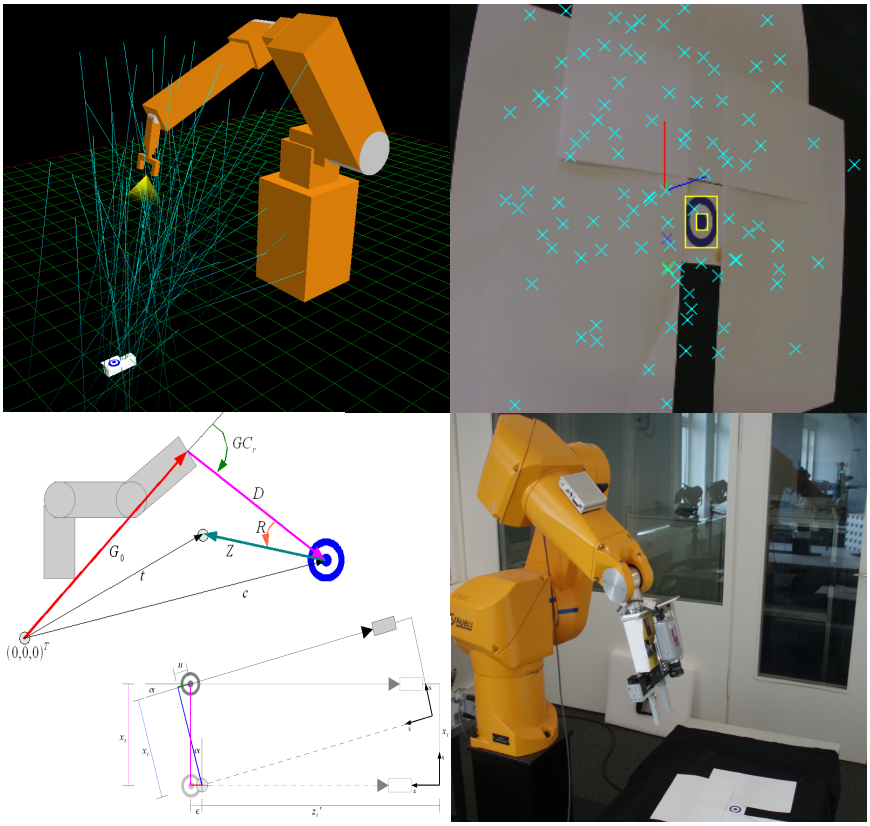
Diplomarbeit

Selbstkalibrierung eines Kamera-Robotersystems



Andreas S. Jordt
(Matrikelnummer: 603012)

Betreuer: Dr. Nils T. Siebel

August 2008

Kognitive Systeme
Prof. Dr. Sommer
Institut für Informatik
Christian-Albrechts-Universität zu Kiel

Abstract

In visual robotics, systems using wrist mounted cameras are very common. To enable such eye-in-hand robot systems to work accurately the knowledge of intrinsic and extrinsic camera parameters is required. Due to the fact that the hardware is inaccessable to direct measurements in most cases, calibration is done by algorithms analysing the camera images shot at different poses of the wrist.
In this thesis a new method for eye-in-hand calibration is introduced and tested. The algorithm falls into the class using point markers, i.e. the class of algorithms requiring only one identifiable point in the environment of the robot system. The method has a two-step approach: First a simple environment model is roughly estimated by analysing 6 poses. This model is then used to generate a set of poses of arbitrary size from which the calibration pattern is most probably visible. Defining the parameters of interest as the solution of an optimisation problem based on the acquired images enables a global optimiser to yield the requested calibration parameters. Hence, unlike common calibration methods, this algorithm does not apply an iterative approach to station far off poses and it does not use local optimisation. An overview over related works on eye-in-hand calibration is given and existing algorithms are compared to the one introduced in this thesis. Experiments show that the algorithm introduced here is not only more versatile in application but also provides more accurate results than the common calibration methods it is compared to.

Zusammenfassung

Im Bereich der visuellen Roboterregelung sind Systeme mit am Greifer montierten Kameras weit verbreitet. Um die präzise Funktion eines Kamera-Robotersystems zu gewährleisten, ist es nötig, im Vorwege die genaue Position der Kamera zum Greifer und deren optische Eigenschaften zu kennen. Da diese auf Grund der Unzugänglichkeit der Komponenten nur bedingt messbar sind, wird diese Kalibrierung üblicherweise von Algorithmen durch Analyse der Kamerabilder an verschiedenen Greiferposen durchgeführt.
In dieser Arbeit wird eine neue Methode zur Kalibrierung von Kamera-Robotersystemen vorgestellt und getestet. Der Algorithmus gehört zu der Klasse der Punktmarker-Verfahren, die zur Kalibrierung lediglich einen optisch wiedererkennbaren Punkt in der Umgebung des Robotersystems benötigen. Das Verfahren besteht aus zwei unterschiedlichen Phasen. Zunächst werden wichtige Umgebungsparameter auf Basis von sechs Greiferbewegungen geschätzt, welche es erlauben, anschließend eine beliebige Anzahl von Posen zu generieren, die das Kalibrierungsmuster mit hoher Wahrscheinlichkeit im Sichtfeld der Kamera haben. Die gesuchten Kalibrierungswerte werden dann als Lösung eines aus den Aufnahmen erzeugten Optimierungsproblems definiert und mit Hilfe eines globalen Optimierers ermittelt. Damit unterscheidet sich das Verfahren grundsätzlich von gängigen Algorithmen, die das Anfahren äußerer Kamerapositionen schrittweise realisieren und meist lokal optimierbare Systeme verwenden. Eine Übersicht über bekannte Verfahren zur Kamera-Roboterkalibrierung wird gegeben und es werden Vergleiche zwischen bereits existierenden und dem hier vorgestellten Verfahren angestellt. Es wird gezeigt, dass der im Rahmen dieser Arbeit entwickelte Algorithmus nicht nur weit aus vielseitiger und flexibler im praktischen Einsatz ist als alle bekannten Methoden, sondern zudem genauere Ergebnisse im Vergleichstest mit gängigen Kalibrierungsverfahren liefert.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Im Bereich der visuellen Roboterregelung ist es häufig erforderlich, die genaue Position und Orientierung einer am Greifer montierten Kamera zu bestimmen. Dadurch, dass man die Kamera mit Hilfe des Roboters sehr genau in alle Raumrichtungen bewegen kann, ist es möglich, diese Kalibrierung mit hoher Genauigkeit völlig automatisch durchzuführen. In dieser Arbeit wird ein neues System zur autonomen Kamerakalibrierung vorgestellt und mit bereits bestehenden Verfahren verglichen. Der hier beschriebene Algorithmus berechnet die Pose der Kamera, die intrinsischen Parameter der Kamera sowie Parameter für Linsenverzerrungsmodelle.

1.1 Aufgabenstellung

Um ein vollständig kalibriertes und einsatzfähiges Kamera-Robotersystem zu erhalten, ist es nötig, folgende Kalibrierungen durchzuführen:

1. Kalibrierung des Robotersystems. Die inverse Kinematik des Roboterarmes ist anschließend in der Lage, den Greifer beliebige Posen im Raum präzise anfahren zu lassen.

2. Kalibrierung der Kamera. Anschließend sind Öffnungswinkel, optisches Zentrum sowie sämtliche Verzerrungsparameter der Kamera bekannt.

3. Kalibrierung der Kamera zum Roboter. Nach dieser Kalibrierung ist die exakte Transformation vom Tool-Koordinatensystem des Roboters zum optischen Zentrum der Kamera bekannt.

In dieser Arbeit wird ein Verfahren vorgestellt, das die unter den Punkten 2. und 3. genannten Aufgaben durchführt. Voraussetzung dafür ist, dass der verwendete Roboterarm bereits vollständig kalibriert ist. Da die meisten Robotersysteme bereits kalibriert ausgeliefert werden, stellt dies keine nennenswerte Einschränkung für den Einsatz der autonomen Kalibrierung dar. Ebenfalls kein Bestandteil dieser Arbeit ist das Bildverarbeitungsverfahren, das in der Lage ist, das Kalibrierungsobjekt in allen aufgenommenen

Bildern zu identifizieren und zu lokalisieren. Es wird hierfür ein voll funktionsfähiges Verfahren vorausgesetzt[1].

1.2 Bekannte Verfahren

Die Aufgabenstellung der Kalibrierung von Kamera-Robotersystemen existiert schon seit mehr als 30 Jahren[2]. Mit dem Einsatz von visuellen Roboterregelungen folgte auch die Notwendigkeit der exakten Kalibrierung des Kamera-Robotersystems. Die bisher entwickelten Methoden lassen sich in drei Klassen unterteilen, abhängig von den verwendeten Kalibrierungsobjekten:

1. Verfahren, die auf Basis eines Schachbrettmusters im Bild eine Schätzung der Lageveränderung vornehmen,
2. Verfahren, die einen oder mehrere markierte Punkte im Raum observieren,
3. Verfahren, die komplett ohne Kalibrierhilfen arbeiten, allein auf Basis von wiedererkennbaren Bildausschnitten.

Diese Klassifizierung unterscheidet sich von der Gliederung Wangs' in [21]. Die von Wang definierte Klasse der Kalibrierungen bei bekannter Position des Kalibrierungsobjekts ist rein theoretischer Natur und nicht praktikabel. Wiederum existierte zur Zeit der Veröffentlichung 1992 noch kein Algorithmus, der komplett ohne Kalibrierungsobjekt arbeitete.

1.2.1 Kalibrierung mit Kalibrierflächen

Die klassische Variante des Kalibrierungsobjekts ist das Schachbrettmuster. In diese Klasse der Kalibrierungsobjekte fallen ebenfalls Punktflächen und alle anderen Muster, die mehr als eine optisch erfassbare Referenz bieten, deren geometrische Lage zueinander äquidistant oder in einer anderen Form im Vorwege bekannt ist (siehe auch Abb. 1.1). Bei einem Kalibriervorgang erlaubt die Verwendung eines Schachbrettmusters eine komplette Schätzung der Lageveränderung zwischen zwei Kamerabildern, also der Transformation zwischen Ausgangs- und Zielpose. Der 1988 von Tsai und Lenz [18, 19] vorgestellte Algorithmus zur Kalibrierung eines Kamera-Robotersystem sowie das von Shiu und Ahmad [13] und Albada, Lagerberg und Visser [20] entwickelte Verfahren verwenden diese Form des Kalibrierungsmusters. Diese Methoden waren die ersten, die die Roboterkalibrierung explizit von der intrinsischen und extrinsischen Kamerakalibrierung zugunsten der Echtzeitfähigkeit ihrer Verfahren trennt. Ein präzise funktionierendes Robotersystem wird hier als vorausgesetzt angesehen.

[1] Informationen zu dem in der Implementierung dieser Methode verwendeten Bildverarbeitungsalgorithmus sind im Kapitel 5.4.2 zu finden.

[2] Erste Verfahren zu diesem Thema wurden 1988/1989 von Tsai und Lenz [18] und von Shiu und Ahmad [13] vorgestellt.

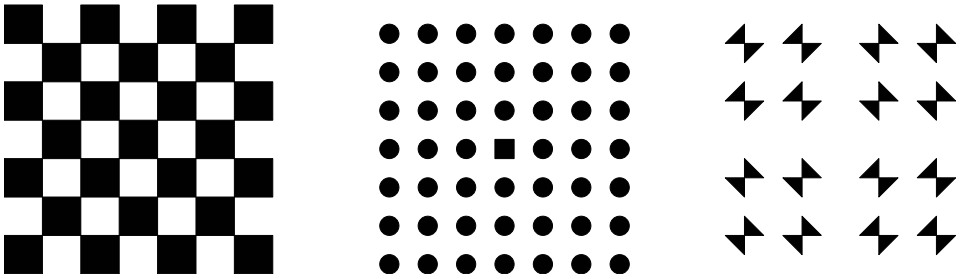


Abbildung 1.1: Die Klasse der Kalibrierflächen zeichnet sich dadurch aus, dass sich mehrere leicht detektierbare Punkte in gleichem oder vorher bekanntem Verhältnis auf einer planen Fläche befinden. Dies erlaubt es, bereits aus einer Bildaufnahme Informationen über die Lage der Kamera zum Kalibrierobjekt zu gewinnen.

Die Verfahren zur Lösung dieser Klasse von Problemen verwenden, wie fast alle, die bis Mitte der 90-er Jahre entwickelt wurden, einen lokalen Optimierungsansatz zur Lösung der aufgestellten Gleichungssysteme. Die hierfür nötigen Anfangsschätzungen müssen meist im Vorwege von Hand eingegeben werden. Dafür war jedoch bereits der Algorithmus von Tsai und Lenz in einer Zeit echtzeitfähig, in der die Rechenleistung der Computer aus heutiger Sicht als verschwindend gering gelten kann.

Das von Shiu und Ahmad entwickelte Verfahren basiert auf der Lösung der Gleichung $AX = XB$ homogener [15] Matrizen, welche direkt aus der Schätzung der Lageveränderung zwischen zwei Kamerabildern resultiert. Wenn A die bekannte Transformation zwischen den Greiferposen beschreibt und B die aus den Bildern errechnete Lageveränderung der Kamera, so ist eine Lösung von X die Transformation vom Greifer zur Kamera und enthält somit die extrinsischen Parameter des Kamera-Robotersystems. $AX = XB$ gilt bis heute als Standardformel der extrinsischen Kamera-Roboterkalibrierung. Park und Martin veröffentlichten 1994 einen analytischen Lösungsansatz für Gleichungen der Form $AX = XB$ [12], die eine mathematische Grundlage zur Berechnung des Einflusses von Bildrauschen liefert und Aussagen über die Konvergenz erlaubt.

1992 veröffentlichten Weng, Cohen und Herniou [23] eine Methode, die die extrinsische und intrinsische Kalibrierung kombinierte. Der Suchraum wurde hier zunächst auf die extrinsischen Parameter beschränkt. Die resultierenden Parameter wurden anschließend als Anfangsschätzung für die Suche im Gesamtparameterraum verwendet. Das Schachbrettmuster ist bis heute aufgrund der Fülle an Informationen, die aus einem einzelnen Bild gewonnen werden können, das Standardmuster der Kamerakalibrierung.

Die Grundkonzepte von Tsai und Lenz sowie die von Shiu und Ahmad werden auch heutzutage noch genutzt und finden Verwendung in modernen Kalibrierungsmethoden (siehe [16]).

1.2.2 Kalibrierung mittels Markerpunkten

Die zweite Klasse der Kamera-Roboterkalibrierung verwendet einen oder mehrere im Raum verteilte Markerpunkte (siehe auch Abb. 1.2). Der Vorteil dieser Verfahren gegenüber der Kalibrierung mit einem Schachbrettmuster ist der einfache Einsatz in der

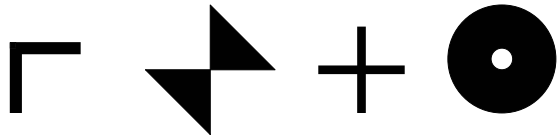


Abbildung 1.2: Markerpunkte zur Kalibrierung können beliebige Formen annehmen. Einzig die Detektierbarkeit im Bild muss gegeben sein. Oft wird zur Klassifizierung neben Form auch Farbe verwendet.

Praxis. Ist die Platzierung eines Schachbrettmusters im industriellen Umfeld eines Roboters oft schwierig, so können Markerpunkte an beliebige Stellen der Umgebung befestigt werden, ohne zusätzlichen Raum zu beanspruchen.
Der Lösungsweg über das Gleichungssystem $AX = XB$ ist in dieser Problemklasse nur bedingt einsetzbar, da hierfür mindestens drei Marker pro Bild zur Aufstellung einer Gleichung notwendig sind. Horaud und Dornaika [9] formulierten daher in ihrem Verfahren ein neues Gleichungssystem, $MY = M'YB$, welches es erlaubt, die intrinsischen und extrinsischen Parameter im Zuge der Optimierung zu bewerten, ohne für jedes Bild explizit eine Projektionsvorschrift liefern zu müssen.
Eine weitere Methode unter Verwendung beliebig vieler Markerpunkte lieferten Wei, Arbter und Hirzinger [22]. Dieses Verfahren stellt pro detektierten Marker eine Fitnessfunktion auf, die für die Gesamtbewertung aufsummiert wird. Durch speziell angepasste Roboterbewegung wird das dabei generierte Optimierungsproblem möglichst stabil gehalten. Die Anwendung einer auf dem Gauss-Markov-Theorem beruhenden Optimierung führt zu sehr genauen Kalibrierungsergebnissen. Erstmals kommt bei diesem Verfahren versuchsweise ein globaler Monte Carlo Optimierer [17] zum Einsatz, dessen Ergebnisse bleiben jedoch hinter denen der Gauss-Markov-Theorem-Optimierung zurück.

1.2.3 Kalibrierung ohne bekannte Marker

Die Klasse der markerlosen Kalibrierungsalgorithmen ist den Verfahren der Markerpunkt basierten Methoden sehr ähnlich, jedoch werden hier anstelle platzierter Muster im Raum Objekte und Texturen im Bild durch Tracking- oder Merkmalspunkt-Verfahren identifiziert, um Korrespondenzen zu generieren, z.B. mittels SIFT-Merkmalspunkten[3] [11]. Neben den Faktoren des Bildrauschens und der Ungenauigkeit durch die Kinematik kommt hier die schwer zu erzielende Positionsgenauigkeit der markanten Punkte bei Betrachtung aus verschiedenen Blickwinkeln hinzu, wie beschrieben in [10]. Eines der wenigen Verfahren aus diesem Bereich von Andreff, Horaud und Espiau [2] liefert dementsprechend Genauigkeiten, die weit hinter den Verfahren der anderen Klassen zurückbleiben.

1.3 Anforderungen an den Algorithmus und Konzeptionierung

Im folgenden Abschnitt werden die an den Algorithmus gestellten Anforderungen und ihre Auswirkung auf die Konzeptionierung des Verfahrens beschrieben. Die im vorange-

[3] SIFT - Scale Invariant Feature Transform

gangenen Kapitel beschriebenen Verfahren und ihre Ergebnisse dienen hierbei als Entscheidungsbasis.

1.3.1 Verwendung von Markerpunkten

Eine der Hauptanforderungen an den Algorithmus besteht in dem Ziel, möglichst genaue Ergebnisse zu liefern. Des Weiteren sollte das Verfahren sehr flexibel sein, um in möglichst vielen verschiedenen Umgebungen einsetzbar zu sein. Da die Verwendung von Schachbrettmustern in vielen Einsatzgebieten unpraktisch ist und eine Kalibrierung ohne im Raum platzierte Markierungen ungenau ausfällt, verwendet der hier vorgestellte Algorithmus Markerpunkte (siehe Kapitel 1.2.2).

1.3.2 Generierung der Greiferposen

Bei der Akquisition der Daten zur Kalibrierung gibt es bei den bekannten Verfahren starke Ähnlichkeiten. Die bisher entwickelten Methoden verwenden zur Generierung der Greiferposen inkrementelle Verfahren, die meist, einer virtuellen Geraden im Raum folgend, die Orientierung der Kamera anhand der observierten Bewegungen im Bild anpassen [22]. Zudem werden meist viele Punktmarker im Raum um den Roboter verteilt, so dass mit jedem Bild eine nicht leere Untermenge der Marker erfasst werden kann.
Diese Methode der Datengewinnung funktioniert, jedoch birgt sie zwei Nachteile:

1. Tsai und Lenz wiesen in [19] nach, dass für eine konstante Anzahl von Roboterbewegungen maximal unterschiedliche Posen im Raum zu einem maximal stabilen Gleichungssystem führen. Die meisten bestehenden Verfahren bewegen den Greifer schrittweise entlang senkrecht aufeinander stehender Achsen im Raum, um das Kalibrierungsobjekt im Kamerabild zu halten. Die Endpunkte dieser Bewegungen erfüllen das Kriterium von Tsai und Lenz nur bedingt. Ohne genaue Analyse der im Vorwege unbekannten Kinematik ist es nicht möglich, diejenigen Strecken im Raum zu finden, die unter Berücksichtigung der Kinematik des Roboters maximal voneinander entfernte Endpunkte haben. Zudem kann eine beliebige Wahl dieser Geraden zu unbeabsichtigt kurzen Strecken führen, etwa durch Einschränkung der Orientierungsfreiheit des Greifers aufgrund der Geometrie des Roboters.

2. Die Verwendung mehrerer Marker garantiert zwar ein Maximum an generierten Daten, jedoch steigt bei einigen Verfahren die Menge der meist implizit vorhandenen Unbekannten[4] mit jedem erkannten Punkt ebenfalls. Punkte, die zu weit voneinander entfernt sind, können schwer in Beziehung gebracht werden; Punkte, die eng beieinander liegen, erzeugen meist redundante Informationen, die eine Gewichtung der Daten notwendig macht. Nur einmal observierte Punkte generieren zudem keinerlei Informationen.

[4]Zur Aufstellung eines Gleichungssystems bestimmen einige Verfahren die Positionen der Markerpunkte im Raum. Da diese nicht bekannt sind, fügt jeder observierte Markerpunkt dem Gleichungssystem drei neue Unbekannte hinzu und erhöht so die Dimension des Suchraums um 3 (z.B. in [22]). Probleme dieser Art können jedoch z.B. mit Hilfe der Epipolargeometrie [7] umgangen werden.

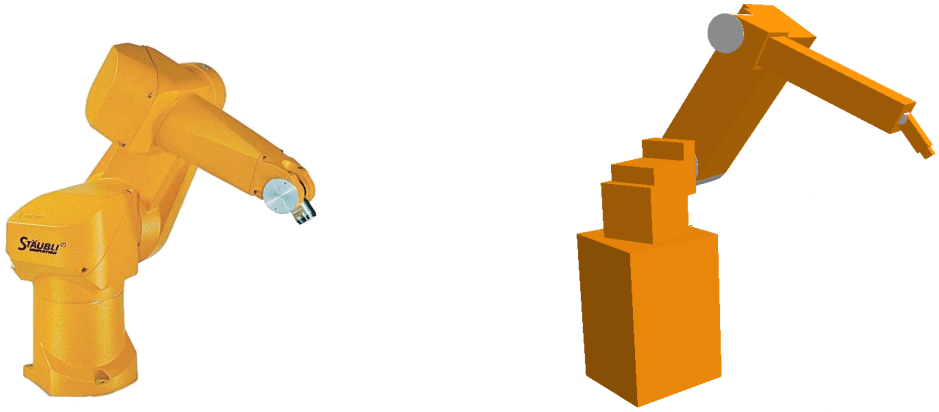


Abbildung 1.3: Links: Der zum Test des Algorithmus verwendete Unimotion Stäubli RX90. Rechts: Eine ebenfalls zu Testzwecken eingesetzte Simulation des Roboters, basierend auf der Geometrie des Stäubli RX90.

Um diese Probleme zu vermeiden, wird in dem hier vorgestellten Verfahren ein neuer Weg der Posengenerierung vorgestellt, und es wird lediglich ein Marker im Sichtfeld des Kamerasystems platziert. Es gilt, mit wenigen Posen – sechs Transformationen – ein Umgebungsmodell zu erstellen, welches von jeder Position im Raum eine Fokussierung des Kalibrierungsmusters gewährleistet. Auf diese Weise kann eine Menge maximal unterschiedlicher Posen generiert werden, die den von der Kinematik erreichbaren Posenraum ausfüllt.

1.3.3 Globale Optimierung

Der Anspruch, Einsetzbarkeit in möglichst vielen Arbeitsumgebungen zu ermöglichen, erfordert die Kompatibilität zu einer Vielzahl von Bildsensorklassen. In [22] wird gezeigt, dass ein komplett universelles Kameramodell mit einer hohen Anzahl von Parametern eine mächtige Fehlerquelle darstellt (Überparametrisierung). Ein möglichst vielseitiges Verfahren sollte daher modular aufgebaut sein, um den einfachen Austausch der Kameramodelle zu gewährleisten. Jedoch ist für beliebige Kameramodelle keine Konvergenz einer lokalen Optimierung zu gewährleisten. Ebenso wird durch die unvorhersehbare Auswahl der Greiferposen, wie in 1.3.2 beschrieben, eine generelle lokale Optimierbarkeit erschwert.
Die Verwendung des globalen Optimierers CMA-ES (***C****ovariance* ***M****atrix* ***A****daptation* - ***E****volution* ***S****trategy*) [6] liefert hier einen entscheidenden Beitrag (siehe auch Kapitel 6.2). Das Verfahren ist in der Lage, für die durch das Gleichungssystem gegebenen hochdimensionalen Suchräume optimale Parameter zu bestimmen, ohne auf eine Anfangsschätzung angewiesen zu sein. Ein analytischer Konvergenznachweis kann wegen der universellen Struktur des Suchraumes nicht gegeben werden, jedoch zeigen die in Kapitel 7 angeführten Testergebnisse die hohe Stabilität und Genauigkeit dieser Methode.

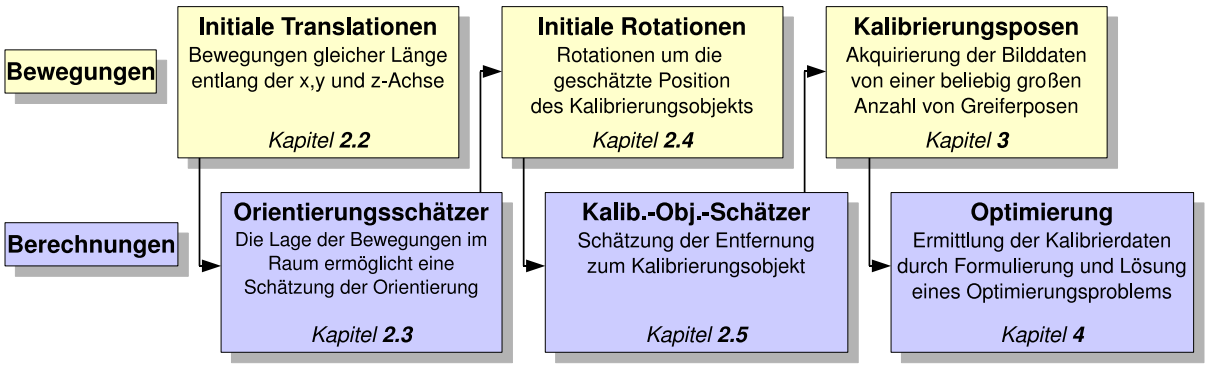


Abbildung 1.4: Schematischer Ablaufplan der autonomen Kalibrierung

1.3.4 Verwendung eines Robotersimulators

Die Bewertung realer Testergebnisse der Kalibrierung von Kamera-Robotersystemen ist nicht ohne weiteres möglich. Der Ursprung des Greiferkoordinatensystems liegt selten auf der Oberfläche des Roboters, und das optische Zentrum einer Kamera befindet sich ebenfalls innerhalb eines Gehäuses. Des Weiteren ist das optische Zentrum einer Kamera nicht einfach zu bestimmen. Weiter sind moderne Verfahren in einen Genauigkeitsbereich vorgedrungen, der unterhalb von per Hand messbaren Größenordnungen liegt.
Ergebnisse realer Tests werden daher lediglich in Varianzen der Parameter für mehrere variierende Testläufe angegeben.

Aufgrund der Verwendung eines globalen Optimierers und der nicht vorhandenen Konvergenztheorie ist es nötig, dass das implementierte Verfahren in einer kontrollierbaren Umgebung getestet wird. Daher wurde für diese Arbeit eine Simulationssoftware entwickelt, die in der Lage ist, beliebige Einsatzszenarien zu simulieren. Somit kann hinreichend belegt werden, dass das Verfahren unter verschiedenen Rahmenbedingungen funktioniert.

1.4 Übersicht über den Algorithmus

Die hier vorgestellte autonome Roboter-Kamerakalibrierung ist in drei Schritte unterteilt, an deren Ende jeweils die Berechnung einiger Parameter durchgeführt wird, wie in Abb. 1.4 zu sehen ist. Voraussetzung für die Durchführung des Algorithmus ist, dass sich das Kalibrierungsobjekt im Zentrum des Kamerabildes befindet oder zumindest im Blickfeld der Kamera ist, so dass eine autonome Fokussierung (siehe Anhang A.3) durchgeführt werden kann. Die drei Bewegungsphasen werden in dieser Arbeit adressiert als:

1. Initiale Translationen
2. Initiale Rotationen
3. Generierte Posen

Die initialen Translationen bestehen aus drei separat durchgeführten Bewegungen entlang der senkrecht aufeinander stehenden Achsen des Weltkoordinatensystems. Die Auswertung der gewonnenen Bilddaten ermöglicht dann eine Schätzung der Kameraorientierung im Raum. Da die Pose des Greifers zu jeder Zeit bekannt ist, erhält man zusätzlich auch die Differenz der Orientierungen zwischen Kamera und Greifer.
Auf der Basis dieser Information lassen sich nun in der zweiten Bewegungsphase Rotationen relativ zum Kamerakoordinatensystem durchführen. Durch geschickte Wahl der Rotationsachsen und Winkel lässt sich dann mit drei Rotationen die Entfernung zwischen Greifer und Kalibrierungsmuster approximieren. Das Ergebnis der Auswertung dieser gewonnenen Daten parametrisiert ein Umgebungsmodell, genannt lineares Modell, welches erlaubt, zu einer beliebigen Position im Raum eine entsprechende Greiferpose zu ermitteln, in der die Kamera auf das Kalibrierungsobjekt ausgerichtet ist. Anhand dieses Modells lassen sich nun beliebig viele Posen im Raum generieren. Die Bild- und Posedaten können dann zu einer Fitnessfunktion kombiniert werden, deren Optimum ein Vektor mit den gesuchten Kalibrierungsparametern ist. Dieser Vektor wird von dem globalen Optimierer CMA-ES ermittelt und als Kalibrierungsergebnis ausgegeben.

Kapitel 2

Die Anfangsschätzungen

2.1 Definitionen

Für die mathematische Beschreibung der Anfangschätzungen seien zunächst einige mathematische Definition angegeben:

2.1.1 Transformationsmatrizen

Im Folgenden sei, wenn von einer Matrix M als homogene Transformation oder Transformationsmatrix gesprochen wird, M von der Form:

$$M \in \mathbb{R}^{4\times4}, M = \begin{pmatrix} & & & x \\ & R & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.1}$$

mit $x, y, z \in \mathbb{R}$ und $R \in \mathbb{R}^{3\times3}$ ist eine orthonormale Matrix mit $\det(R) = 1$. Diese restriktive Form der Definition schließt Transformationen wie Scherung, Spiegelung oder Skalierung aus und beschränkt sich somit auf Manipulationen, die auch im Realen vorkommen können.

2.1.2 Zerlegung einer Transformationsmatrix

In den folgenden Abschnitten wird es nötig sein, homogene Transformationen im dreidimensionalen Raum, die in Form einer Matrix vorliegen, in eine Translation und eine Rotation aufzuteilen. Seien hierfür die Indizes $_t$ für den translativen und $_r$ für den rotierenden Teil eingeführt, wobei für eine beliebige Transformationsmatrix M gelte:

$$T = \begin{pmatrix} r_{0,0} & r_{0,1} & r_{0,2} & t_0 \\ r_{1,0} & r_{1,1} & r_{1,2} & t_1 \\ r_{2,0} & r_{2,1} & r_{2,2} & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}_t = \begin{pmatrix} 1 & 0 & 0 & t_0 \\ 0 & 1 & 0 & t_1 \\ 0 & 0 & 1 & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

und

$$M_r = \begin{pmatrix} r_{0,0} & r_{0,1} & r_{0,2} & t_0 \\ r_{1,0} & r_{1,1} & r_{1,2} & t_1 \\ r_{2,0} & r_{2,1} & r_{2,2} & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}_r = \begin{pmatrix} r_{0,0} & r_{0,1} & r_{0,2} & 0 \\ r_{1,0} & r_{1,1} & r_{1,2} & 0 \\ r_{2,0} & r_{2,1} & r_{2,2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.3}$$

Homogene Transformationen M können so durch ihren rotierenden und translativen Teil beschrieben werden, in der Form, dass für beliebige M gilt:

$$M = M_t \cdot M_r. \tag{2.4}$$

Anmerkung:
Im Allgemeinen gilt nicht:

$$(M_t)^{-1} = (M^{-1})_t, \tag{2.5}$$

jedoch gilt immer:

$$(M_r)^{-1} = (M^{-1})_r. \tag{2.6}$$

2.1.3 Subtraktion homogener Vektoren

Des Weiteren werden räumliche Abstände und Positionsdifferenzen betrachtet. Hierzu sei im Gegensatz zu der im $\mathbb{R}^4$ üblichen Definition das Ergebnis einer Subtraktion im nicht homogenen, dreidimensionalen Ortsraum berechnet. Sei $a, b \in \mathbb{R}^4$, dann gelte

$$a - b = \begin{pmatrix} x_a \\ y_a \\ z_a \\ s_a \end{pmatrix} - \begin{pmatrix} x_b \\ y_b \\ z_b \\ s_b \end{pmatrix} := \begin{pmatrix} \frac{x_a}{s_a} - \frac{x_b}{s_b} \\ \frac{y_a}{s_a} - \frac{y_b}{s_b} \\ \frac{z_a}{s_a} - \frac{z_b}{s_b} \\ 1 \end{pmatrix}. \tag{2.7}$$

Mit dieser Definition ist es möglich, die translative Differenz zwischen zwei homogenen Ortsvektoren durch eine Subtraktion zu beschreiben, ohne die Skalierungskomponente des Ergebnisvektors zu kompromittieren.

2.1.4 Norm homogener Vektoren

Um die Entfernung zweier durch homogene Vektoren beschriebene Punkte im dreidimensionalen Raum zu notieren und um die Länge eines homogenen Ortsvektors sinngemäß angeben zu können, sei die h-Norm $\|\cdot\|_h$ im $\mathbb{R}^4$ eingeführt, bei der die Norm eines homogenen Ortsvektors seiner Norm im dreidimensionalen euklidischen Raum entspricht: Sei $a \in \mathbb{R}^4$, dann gelte

$$\|a\|_h = \left\| \begin{pmatrix} x_a \\ y_a \\ z_a \\ s_a \end{pmatrix} \right\|_h = \sqrt{x_a^2 + y_a^2 + z_a^2}. \tag{2.8}$$

Die Normeigenschaften sind für die h-Norm $\|\cdot\|_h$ offensichtlich gegeben.

2.1.5 Ursprung eines Koordinatensystems

Mit dem Ursprung eines Koordinatensystems sei im Folgenden die translative Komponente einer Transformationsmatrix beschrieben. Der Ursprung von $M \in \mathbb{R}^{4\times4}$ ist somit gegeben durch:

$$M \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.9}$$

2.2 Initiale Translationen

Sei die Grundstellung des Greifers durch eine homogene Matrix $G_0 \in \mathbb{R}^{4\times4}$ gegeben. In dieser Grundstellung sei das Kalibrierungsobjekt fokussiert[1]. Für jede weitere Pose des Greifers im Algorithmus ist die entsprechende Pose im Weltkoordinatensystem durch die fortlaufende Indizierung gekennzeichnet (G_1, G_2,...). Sei $N \subset \mathbb{N}$ die dazu gehörige Indexmenge entsprechend der Anzahl der Posen/Bilder.

Der Algorithmus führt nun von der Grundpose G_0 separat Translationen gleicher Länge in die Richtung der Achsen des Weltkoordinatensystems aus. Die Länge dieser Translation sei als δ_t definiert.
Diese Bewegungen definieren gleichzeitig die homogenen Matrizen der drei Posen als G_1, G_2 und $G_3 \in \mathbb{R}^{4\times4}$. Ziel dieser Bewegungen ist es, eine erste Schätzung der Orientierung der Kamera im Weltkoordinatensystem zu erhalten. Sei dafür zunächst die Bewegung des Kalibrierungsobjekts im Kamerakoordinatensystem betrachtet.

Für die Transformation von einer Greiferposen G_i zu einer anderen G_j sei nun die Schreibweise $G_{i,j}$ für $i, j \in N$ eingeführt. Diese Transformation sei für G_i und G_j definiert, so dass gilt:

$$\forall i, j \in N : G_j = G_i \cdot G_{i,j}. \tag{2.10}$$

Da nach Definition $G_{0,1}$, $G_{0,2}$ und $G_{0,3}$ äquidistante Translationen entlang der x- ,y- und z-Achse sind, fährt der Greifer des Roboters mit diesen Bewegungen die um δ_t skalierten Achsen des Weltkoordinatensystems ab.

Das Kamerakoordinatensystem sei durch die homogene Matrix C im Weltkoordinatensystem angegeben. Da die Kamera fest am Greifer montiert ist, muss das Kamerakoordinatensystem für jede Greiferpose separat definiert werden. Sei für alle Greiferposen G_i das entsprechende Kamerakoordinatensystem durch $C_i \in \mathbb{R}^{4\times4}$ definiert. Da die Transformation zwischen Greifer und Kamera fest ist, existiert eine homogene Transformationsmatrix GC (siehe auch Abb. 2.1), für die gilt:

$$C_i = G_i \cdot GC, \forall i \in N \tag{2.11}$$

[1]Für den Fall, dass das Kalibrierungsobjekt sich lediglich im Bild, jedoch nicht im Fokus des Bildes befindet, ist unter Kapitel A.3 ein Verfahren zur autonomen Fokussierung zu finden.

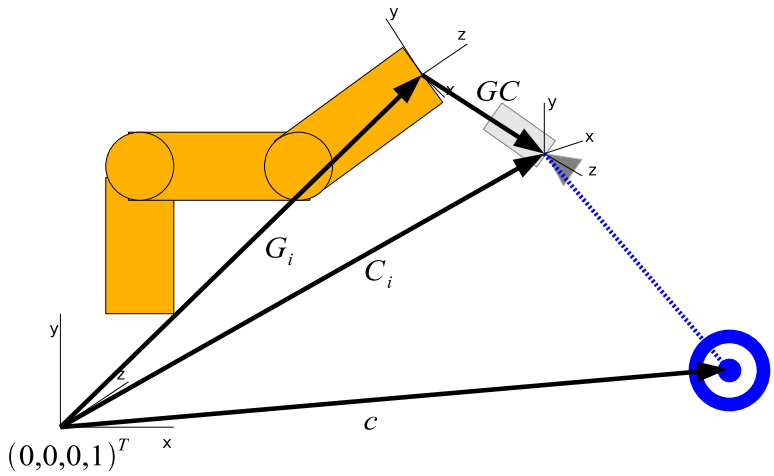


Abbildung 2.1: Die Anordnung der Vektoren/Koordinatensysteme für eine Pose i.

Des Weiteren seien analog zu $G_{i,j}$ die Transformationen $C_{i,j}$ definiert. Es gelte:

$$\forall i, j \in N : C_j = C_i \cdot C_{i,j}. \tag{2.12}$$

Sei nun $c \in \mathbb{R}^4$ die Position des Kalibrierungsobjekts im Weltkoordinatensystem. Die von der Kamera aus beobachteten Differenzen der Position des Kalibrierungsobjekts im Kamerakoordinatensystem seien als Vektoren $t_1 \in \mathbb{R}^4$ für $G_{0,1}$, $t_2 \in \mathbb{R}^4$ für $G_{0,2}$ und $t_3 \in \mathbb{R}^4$ für $G_{0,3}$ definiert, also als Ursprung der Transformationen nach Definition 2.1.5. Es gilt:

$$\begin{aligned} t_1 &= (C_1^{-1} \cdot c) - (C_0^{-1} \cdot c) = ((G_1 \cdot GC)^{-1} \cdot c) - ((G_0 \cdot GC)^{-1} \cdot c) \\ &= (GC^{-1} \cdot G_1^{-1} \cdot c) - (GC^{-1} \cdot G_0^{-1} \cdot c) \\ &= (GC^{-1} \cdot (G_{1,r})^{-1} \cdot (G_{1,t})^{-1} \cdot c) - (GC^{-1} \cdot (G_{0,r})^{-1} \cdot (G_{0,t})^{-1} \cdot c) \end{aligned} \tag{2.13}$$

und da $G_{1,r} = G_{0,r}$, gilt

$$t_1 = GC^{-1} \cdot (G_{0,r})^{-1} \cdot ((G_{1,t})^{-1} \cdot c - (G_{0,t})^{-1} \cdot c). \tag{2.14}$$

Für $((G_{1,t})^{-1} \cdot c - (G_{0,t})^{-1} \cdot c)$ gilt:

$$\begin{aligned} &((G_{1,t})^{-1} \cdot c - (G_{0,t})^{-1} \cdot c) \\ &= \begin{pmatrix} 1 & 0 & 0 & x_{G_0} + \delta_t \\ 0 & 1 & 0 & y_{G_0} \\ 0 & 0 & 1 & z_{G_0} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & x_{G_0} \\ 0 & 1 & 0 & y_{G_0} \\ 0 & 0 & 1 & z_{G_0} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \cdot \begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} x_c - x_{G_0} - \delta_t \\ y_c - y_{G_0} \\ z_c - z_{G_0} \\ 1 \end{pmatrix} - \begin{pmatrix} x_c - x_{G_0} \\ y_c - y_{G_0} \\ z_c - z_{G_0} \\ 1 \end{pmatrix} = \begin{pmatrix} -\delta_t \\ 0 \\ 0 \\ 1 \end{pmatrix} \end{aligned} \tag{2.15}$$

und somit folgt:

$$t_1 = (GC_r)^{-1} \cdot (G_{0,r})^{-1} \cdot \begin{pmatrix} -\delta_t \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.16}$$

Die von der Kamera beobachtete Bewegung t_1 lässt einen Rückschluss auf $GC_r^{-1} \cdot G_{0,r}^{-1}$ zu, die Orientierung der Kamera im Weltkoordinatensystem. Analog folgt nun des Weiteren:

$$t_2 = (GC_r)^{-1} \cdot (G_{0,r})^{-1} \cdot \begin{pmatrix} 0 \\ -\delta_t \\ 0 \\ 1 \end{pmatrix} \tag{2.17}$$

und

$$t_3 = GC_r^{-1} \cdot (G_{0,r})^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ -\delta_t \\ 1 \end{pmatrix}. \tag{2.18}$$

Diese Gleichungen lassen sich zu einer äquivalenten Matrixgleichung zusammenfassen:

$$\begin{pmatrix} & & & 0 \\ t_1 & t_2 & t_3 & 0 \\ & & & 0 \\ & & & 1 \end{pmatrix} = GC_r^{-1} \cdot (G_{0,r})^{-1} \cdot \begin{pmatrix} -\delta_t & 0 & 0 & 0 \\ 0 & -\delta_t & 0 & 0 \\ 0 & 0 & -\delta_t & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.19}$$

Da bis auf die gesuchte Transformation GC_r alle Variablen dieser Gleichung bekannt sind, lässt sich nun GC_r durch einfache Umformung angeben. Da t_1, t_2 und t_3 paarweise orthogonal zueinander sind und $\delta_t = \|t_1\|_h = \|t_2\|_h = \|t_3\|_h$, folgt, dass

$$\begin{pmatrix} & & & 0 \\ t_1 & t_2 & t_3 & 0 \\ & & & 0 \\ & & & 1 \end{pmatrix} \cdot \begin{pmatrix} \frac{-1}{\delta_t} & 0 & 0 & 0 \\ 0 & \frac{-1}{\delta_t} & 0 & 0 \\ 0 & 0 & \frac{-1}{\delta_t} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.20}$$

eine orthonormale Matrix ist. Da die Inverse einer orthonormalen Matrix gleich ihrer transponierten ist, folgt aus 2.19 direkt:

$$GC_r = (G_{0,r})^{-1} \cdot \begin{pmatrix} \frac{-1}{\delta_t} & 0 & 0 & 0 \\ 0 & \frac{-1}{\delta_t} & 0 & 0 \\ 0 & 0 & \frac{-1}{\delta_t} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} & t_1^T & & \\ & t_2^T & & \\ & t_3^T & & \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.21}$$

Demnach lässt die Beobachtung der Bewegungen des Kalibrierungsobjekts im Kamerakoordinatensystem während der initialen Translationen des Greifers theoretisch eine komplette Berechnung der Orientierung von der Kamera zum Greifer zu. Jedoch liefert eine

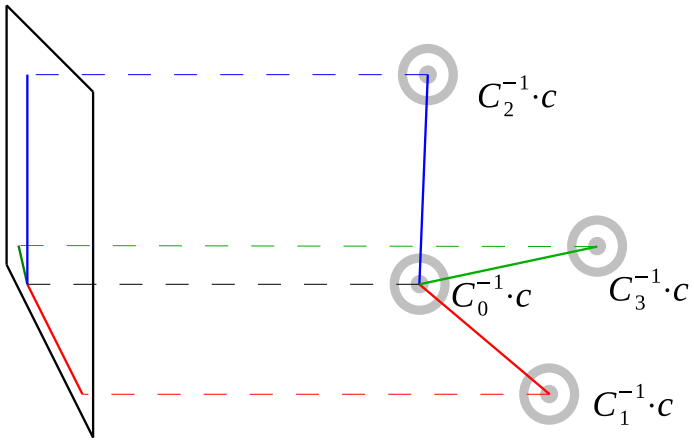


Abbildung 2.2: Die Rekonstruktion der dreidimensionalen Basisvektoren aus den Informationen der zweidimensionalen Bildebene ist möglich durch die bekannten Informationen der Kinematik.

Kamera immer nur eine zweidimensionale Projektion der Umgebung, so dass Teile der benötigten Informationen verloren gehen. Im folgenden Abschnitt wird eine Methode vorgestellt, die die bei dieser Projektion verlorengegangenen Komponenten von t_1,t_2 und t_3 rekonstruieren kann und dann unter Verwendung von 2.21 eine Schätzung für GC_r angibt.

2.3 Der Orientierungsschätzer

Im Folgenden sei die zu kalibrierende Kamera durch die Funktion $P : \mathbb{R}^4 \longrightarrow \mathbb{R}^2$ gegeben, eine Abbildung eines homogenen Ortsvektors im Kamerakoordinatensystem auf die Bildebene der Kamera, wobei das optische Zentrum im Punkt $(0, 0, 0, 1)^T$ liege. Da zu Anfang des Kalibrierungsprozesses weder Informationen über P noch über die Entfernung zwischen Kamera und Kalibrierungsobjekt vorliegen, wird zunächst ein möglichst einfaches Kameramodell verwendet.
Im Folgenden sei deshalb P^* definiert: P^* sei als vorläufiges Kameramodell durch eine Parallelprojektion definiert. Diese Parallelprojektion sei parametrisiert durch einen Skalierungsvektor $\lambda \in \mathbb{R}^2$:

$$P^*_\lambda := \mathbb{R}^4 \longrightarrow \mathbb{R}^2, \begin{pmatrix} x \\ y \\ z \\ s \end{pmatrix} \longmapsto \begin{pmatrix} x \cdot \lambda_x \\ y \cdot \lambda_y \end{pmatrix}, \lambda = \begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix} \tag{2.22}$$

Im Folgenden wird P^* als Approximation des Kameramodells P verwendet. Dadurch entstehende Fehler werden an dieser Stelle vernachlässigt. Eine Analyse der entstehenden Diskrepanzen und der Fortpflanzung des Fehlers durch den Algorithmus ist in Anhang B.1 aufgeführt. Es zeigt sich, dass die Annahme eines bestimmten Kameramodells nur einen sehr geringen Einfluss auf die damit berechneten Daten nimmt. Die Annahme von P^* als tatsächlichem Kameramodell ermöglicht es, die durch die Projektion verloren gegangenen Informationen über die relativen z-Komponenten von t_1,t_2 und t_3 zu rekonstruieren.

t_1, t_2, t_3 seien, wie im vorherigen Kapitel, definiert als Differenzvektoren der initialen Transformationen. Aus Kapitel 2.2 ist bekannt, dass die Kenntnis von t_1, t_2, t_3 für die Bestimmung von GC_r ausreichend ist. Anhand der Beobachtungen des Kamerabildes und unter der Annahme von P^* lassen sich jedoch nur $P^*(t_1)$, $P^*(t_2)$ und $P^*(t_3)$ ermitteln. Es gilt:

$$P^*(t_1) = \begin{pmatrix} (t_1)_x \cdot \lambda_x \\ (t_1)_y \cdot \lambda_y \end{pmatrix}, P^*(t_2) = \begin{pmatrix} (t_2)_x \cdot \lambda_x \\ (t_2)_y \cdot \lambda_y \end{pmatrix}, P^*(t_3) = \begin{pmatrix} (t_3)_x \cdot \lambda_x \\ (t_3)_y \cdot \lambda_y \end{pmatrix} \tag{2.23}$$

Um aus diesen Beobachtungen t_1, t_2 und t_3 komplett zu rekonstruieren, ist es notwendig, λ_x und λ_y zu kennen, sowie die von P^* vernachlässigte z-Komponenten der Vektoren, denn es gilt:

$$\exists \lambda_x, \lambda_y, (t_1)z, (t_2)z, (t_3)z \in \mathbb{R} : t_1 = \begin{pmatrix} \frac{P^*(t_1)_x}{\lambda_x} \\ \frac{P^*(t_1)_y}{\lambda_y} \\ (t_1)_z \\ 1 \end{pmatrix} \wedge t_2 = \begin{pmatrix} \frac{P^*(t_2)_x}{\lambda_x} \\ \frac{P^*(t_2)_y}{\lambda_y} \\ (t_2)_z \\ 1 \end{pmatrix} \wedge t_1 = \begin{pmatrix} \frac{P^*(t_3)_x}{\lambda_x} \\ \frac{P^*(t_3)_y}{\lambda_y} \\ (t_3)_z \\ 1 \end{pmatrix} \tag{2.24}$$

Die Herleitung der gesuchten Komponenten ist ausführlich in Anhang A.1 beschrieben. An dieser Stelle seien lediglich die Ergebnisse dieser Herleitung aufgeführt.
Für die Skalierung der Parallelprojektion gilt:

$$\lambda_x = \frac{-1}{\delta_t} \cdot \left\| \begin{pmatrix} P^*(t_1)_x \\ P^*(t_2)_x \\ P^*(t_3)_x \\ 0 \end{pmatrix} \right\|_2 \tag{2.25}$$

und

$$\lambda_y = \frac{-1}{\delta_t} \cdot \left\| \begin{pmatrix} P^*(t_1)_y \\ P^*(t_2)_y \\ P^*(t_3)_y \\ 0 \end{pmatrix} \right\|_2, \tag{2.26}$$

wobei δ_t die in Kapitel 2.2 definierte Länge der initialen Translationen ist.
Die z-Komponenten der Vektoren t_1, t_2 und t_3 lassen sich durch ein Kreuzprodukt berechnen. Es gilt:

$$\begin{pmatrix} (t_1)_z \\ (t_2)_z \\ (t_3)_z \end{pmatrix} = -\delta_t \cdot \left(\begin{pmatrix} \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_x}{\delta_t \cdot \lambda_x} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_x}{\delta_t \cdot \lambda_x} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_x}{\delta_t \cdot \lambda_x} \end{pmatrix} \times \begin{pmatrix} \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_y}{\delta_t \cdot \lambda_y} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_y}{\delta_t \cdot \lambda_y} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_y}{\delta_t \cdot \lambda_y} \end{pmatrix} \right) \tag{2.27}$$

Durch Einsetzen der Vektoren t_1, t_2 und t_3 in die Formel (2.21) erhält man GC_r, die Orientierung der Kamera zum Greifer. Aufgrund der anfänglich getätigten Annahme über das Kameramodell P handelt es sich bei der berechneten Matrix GC_r lediglich um eine Schätzung, deren Genauigkeit in Anhang B.1 analysiert ist. Diese Schätzung erlaubt es nun, Translationen und Rotationen entweder relativ zum Weltkoordinatensystem oder relativ zum Koordinatensystem der Kamera durchzuführen.

2.4 Initiale Rotationen

Um eine Fokussierung des Kalibrierungsobjekts zu gewährleisten, ist notwendig, einen Anhaltspunkt über die Position des Objektes zu erhalten. Zusätzlich zu der im vorangegangen Kapitel berechneten Orientierung der Kamera ist die Distanz zwischen Greifer und Kalibrierungsobjekt zu berechnen.
Zur Ermittlung dieser Distanz wird die Tatsache ausgenutzt, dass eine Rotation des Greifers um eine die Objektposition $c \in \mathbb{R}^4$ durchlaufende Achse keine Änderung der Objektposition im Kamerakoordinatensystem hervorruft. Sei $G_4^* \in \mathbb{R}^{4\times4}$ so gewählt, dass $G_4^* \neq G_0$ jedoch gilt:

$$G_4^{*-1} \cdot c = G_0^{-1} \cdot c. \tag{2.28}$$

Somit folgt sofort

$$C_4^{*-1} \cdot c = C_0^{-1} \cdot c, \tag{2.29}$$

da

$$GC^{-1} \cdot G_4^{*-1} \cdot c = GC^{-1} \cdot G_0^{-1} \cdot c. \tag{2.30}$$

Findet man eine solche Transformation $G_{0,4}^*$, lässt sich die Menge der möglichen Positionen von c auf einen eindimensionalen Unterraum beschränken, identisch mit der Rotationsachse von $G_{0,4}^*$. Da c ebenfalls auf der Sichtlinie von der Kamera zum Kalibrierungsobjekt liegt, lässt sich c eindeutig ermitteln. Voraussetzung hierfür ist jedoch, dass die Sichtlinie bekannt ist, was neben der bekannten Orientierung auch Kenntnis über die translative Komponente von GC erfordert, die zu diesem Zeitpunkt noch unbekannt ist.

Aus diesem Grund sei das lineare (Umgebungs-)Modell eingeführt. Das lineare Modell wird durch die Annahme definiert, die translative Komponente von GC beschränke sich auf die eindimensionale Untermenge der Translationen entlang der optischen Achse der Kamera. Die unter dieser Annahme verwendete Transformation von Greifer zu Kamera sei als GC^* definiert. Sei $z \in \mathbb{R}$ der Wert dieser translativen Komponente, so bedeutet dies GC^* ist von der Form

$$GC^* = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot GC_r \tag{2.31}$$

und damit

$$(GC^*)^{-1} = (GC^*)_r^{-1} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.32}$$

Somit ist gleichzeitig die translative Komponente von $(GC^*)^{-1}$ gegeben:

$$GC_t^{*-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{2.33}$$

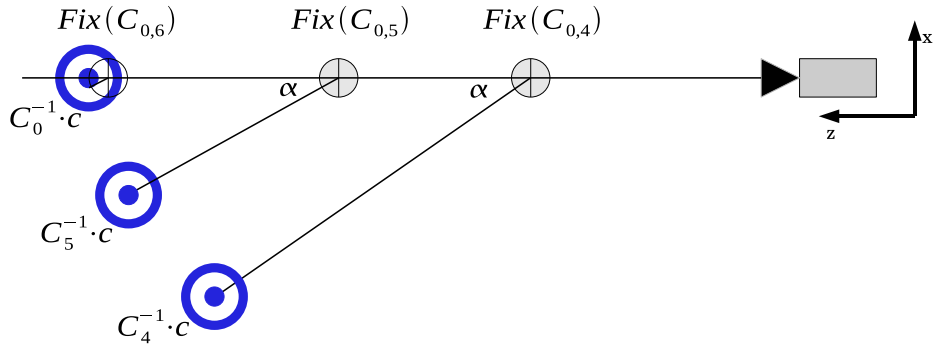


Abbildung 2.3: Die Rotation um die Schätzung der Kalibrierungsobjektposition liefert ein Fehlermaß für den Schätzer durch die resultierende Bewegung im Kamerakoordinatensystem. Die Bewegung des Kalibrierungsobjekts im Bild der Kamera ist dabei abhängig von der Distanz zwischen der Rotationsachse der Transformation (Menge der Fixpunkte) und der Position des Kalibrierungsobjekts.

Unter dieser Annahme beschränkt sich die Suche nach c auf die Suche nach der Distanz zwischen c und dem Ursprung von G_0, da die Sichtlinie bekannt ist.
Der durch das lineare Modell verursachte Fehler in der Schätzung und seine Fortpflanzung im Algorithmus sind in Anhang B.2 und B.3 zu finden.

Sei nun $d \in \mathbb{R}$ eine erste Schätzung der Distanz zwischen c und dem Ursprung von G_0. Definiere nun $G_{0,4}$ als Rotation um die zur x-Achse des Kamerakoordinatensystems parallelen Achse durch den Punkt $G_0 \cdot (0,0,d,1)^T$ mit dem Winkel α:

$$G_{0.4} := GC^* \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ 0 & -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot (GC^*)^{-1} \tag{2.34}$$

und

$$G_4 := G_0 \cdot G_{0,4} \tag{2.35}$$

Sei nun $\epsilon \in \mathbb{R}$ der Fehler in der Distanzschätzung d, so dass $d + \epsilon$ die gesuchte Distanz zwischen c und dem Ursprung von G_0 ist. Demnach gilt:

$$c = C_0^* \cdot \begin{pmatrix} 0 \\ 0 \\ d+\epsilon \\ 1 \end{pmatrix} \tag{2.36}$$

Die resultierende Bewegung des Kalibrierungsobjekts im Kamerakoordinatensystem ist somit gegeben durch:

$$((C_0^*)^{-1} \cdot c) - ((C_4^*)^{-1} \cdot c) = (C_0^*)^{-1} \cdot C_0^* \cdot \begin{pmatrix} 0 \\ 0 \\ d+\epsilon \\ 1 \end{pmatrix} - (C_4^*)^{-1} \cdot C_0^* \cdot \begin{pmatrix} 0 \\ 0 \\ d+\epsilon \\ 1 \end{pmatrix}$$

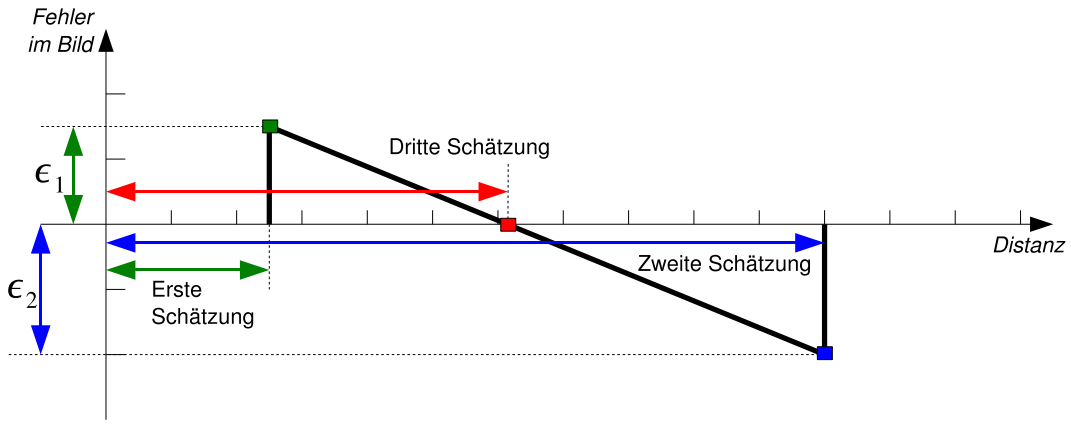


Abbildung 2.4: Aufgrund der linearen Beziehung zwischen geschätzter Distanz und dem resultierendem Bildfehler genügen zwei Schätzungen für eine Berechnung.

$$= \begin{pmatrix} 0 \\ 0 \\ d+\epsilon \\ 1 \end{pmatrix} - (C^*_{0,4})^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ d+\epsilon \\ 1 \end{pmatrix} =$$

$$\begin{pmatrix} 0 \\ 0 \\ d+\epsilon \\ 1 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ 0 & -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ d+\epsilon \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \\ d+\epsilon \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ \sin(\alpha) \cdot \epsilon \\ d + \cos(\alpha) \cdot \epsilon \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -\sin(\alpha) \cdot \epsilon \\ \epsilon - \cos(\alpha) \cdot \epsilon \\ 1 \end{pmatrix} \tag{2.37}$$

Die aus den Rotationsposen entstehende Bewegung des Kalibrierungsobjekts im Kamerakoordinatensystem hat offensichtlich die Eigenschaft, dass ihre y-Komponente für jede Distanzschätzung d und konstantem Rotationswinkel α linear zum Fehler ϵ ist. Diese Erkenntnis führt zu einem Verfahren, mit dem sich die Distanz zwischen Greifer und Kalibrierungsobjekt sehr genau bestimmen lässt.

2.5 Schätzung der Entfernung zum Kalibrierungsobjekt

Unter der Verwendung eines beliebigen Lochkameramodells und einer Gesamtentfernung $d' \in \mathbb{R}_{\geq 0}$ folgt aus (2.37), dass die Bewegung im unverzerrten Kamerabild linear zu

$$\begin{pmatrix} 0 \\ \frac{-\sin(\alpha)\cdot\epsilon}{d'+\epsilon\cdot(1-\cos(\alpha))} \end{pmatrix} - \begin{pmatrix} 0 \\ \frac{0}{d'} \end{pmatrix} \tag{2.38}$$

ist, unter Vernachlässigung der Linsenverzerrung.
Das bis auf einen kleinen Fehler durch die Distanzänderung lineare Verhältnis zwischen

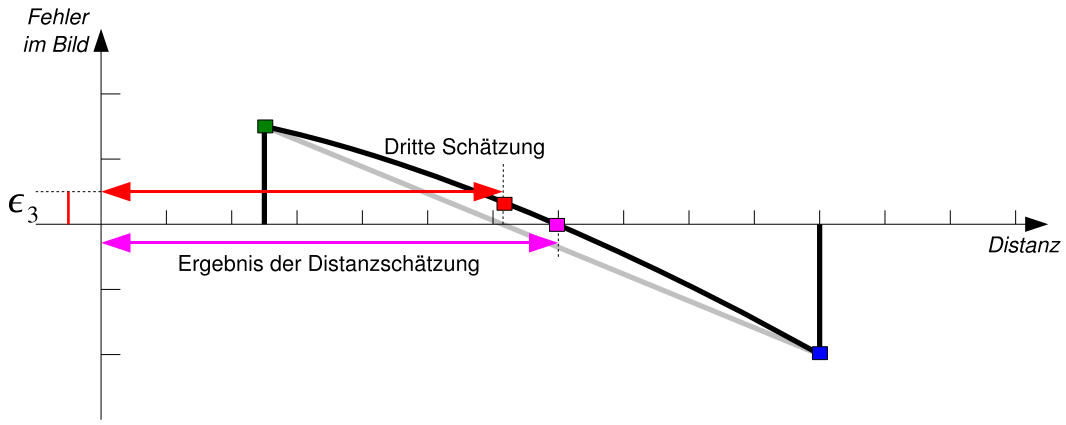


Abbildung 2.5: Ist das Ergebnis der linearen Interpolation ungenügend, so kann anhand des dritten Bildfehlers eine quadratische Interpolation durchgeführt werden.

beobachteter Bewegung im Bild und dem Distanzfehler kann nun ausgenutzt werden, indem zwei ungleiche Entfernungsschätzungen die Basis für eine lineare Interpolation von Fehler und Distanz erlauben (siehe Abb. 2.4).
Die entsprechenden Posen des Greifers seien durch G_4 und G_5 gegeben. Nun ermittelt man die Schätzung d, für die nach linearer Interpolation $\epsilon = 0$ gilt. Sei die Pose dieser Schätzung als G_6 definiert. Sollte die Differenz der Position des Kalibrierungsobjekts in G_0 und G_6 nicht minimal sein, lässt sich G_6 zur Verfeinerung der Distanzschätzung mittels quadratischer Interpolation verwenden (siehe Abb. 2.5). Linsenverzerrungen und anderen, nichtlinearen Komponenten der Bewegung im Bild wird somit Rechnung getragen, da deren Einflüsse in der Bildmitte gegen 0 gehen. Bei einer exakten Distanzschätzung würde nach (2.37) keine Bewegung des Kalibrierungsobjekts im Bild stattfinden und somit kann es sich auch nicht im Bild der Kamera bewegen.

Kapitel 3

Generierung weiterer Bilddaten

Nachdem die sechs initialen Transformationen durchgeführt sind, liegen folgende Daten über das Kamera-Robotersystem und seine Umgebung als Schätzung vor:

- Die Orientierung der Kamera im Weltkoordinatensystem $(C_0)_r$
- Die Orientierung der Kamera relativ zum Greifer $(GC)_r$
- Die Entfernung zwischen Greifer und Kalibrierungsobjekt d

Auf Basis dieser Daten können nun beliebig viele Greiferposen berechnet werden, in denen die am Arm des Roboters montierte Kamera das Kalibrierungsmuster im Kamerabild zentriert (bis auf eine geringe Abweichung, siehe Fehlerabschätzung in Kapitel B.3).

3.1 Berechnung der Greiferposen

Sei $d \in \mathbb{R}_{\geq 0}$, wie im Kapitel 2, die Distanz zwischen dem Ursprung des Greiferkoordinatensystems und dem Kalibrierungsmuster, G_0 die Ausgangspose des Greifers und GC^* die geschätzte Transformation zwischen Greifer und Kamera. Um auf Basis des linearen Modells eine Menge von Posen zu generieren, ist eine Abbildung zu definieren, die jede Position im Weltkoordinatensystem auf eine entsprechende Greiferpose abbildet. Sei diese Funktion K im Folgenden beschrieben durch:

$$K : \mathbb{R}^4 \to \mathbb{R}^{4\times 4} \tag{3.1}$$

K wird auf Basis des bereits in Kapitel 2.4 verwendeten linearen Modells definiert, durch die in Abb. 3.1 skizzierten Transformationen. Hierfür sei zunächst die Fokussierung in dem Koordinatensystem $G_0 \cdot (GC^*)_r$ betrachtet, also dem Kamerakoordinatensystem des linearen Modells. Dieses System hat seinen Ursprung im Greifer, und die z-Achse zeigt auf das Kalibrierungsobjekt. Nach dem linearen Modell gilt hier:

$$(G_0 \cdot (GC^*)_r)^{-1} \cdot c = \begin{pmatrix} 0 \\ 0 \\ d \\ 1 \end{pmatrix}. \tag{3.2}$$

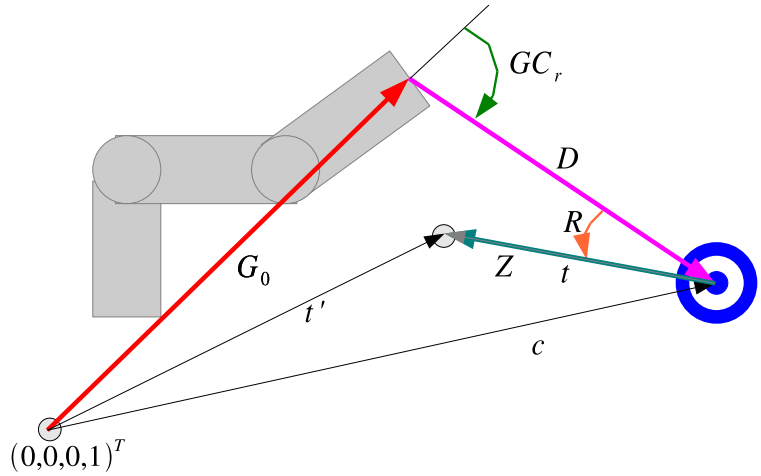


Abbildung 3.1: Die Berechnung einer fokussierenden Pose im Raum. G_0 (rot), GC_r (grün) und die Distanz d für die Transformation D (Magenta) sind bekannt. Die Rotation R (orange) und die Translation entlang der durch R definierten Achse Z (türkis) sind so zu wählen, dass die Hintereinanderausführung aller Transformationen ihren Ursprung in t hat.

Weiter gilt, dass jeder Punkt p, dessen x- und y-Komponente im Kamerakoordinatensystem verschwinden, also die Form $(0, 0, \cdot, 1)^T$ hat, von der Kamera fokussiert wird, da er auf der optischen Achse der Kamera liegt. K muss demnach zwei Bedingungen erfüllen: Sei $t' \in \mathbb{R}^4$ ein homogener Vektor, dann muss gelten

1. $K(t')$ sei eine Pose mit Ursprung in t'

$$K(t') \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = t' \tag{3.3}$$

2. Mit Pose $K(t')$ fokussiere die Kamera das Objekt c

$$\exists z \in \mathbb{R} : (GC)_r^{-1} \cdot K(t')^{-1} \cdot c = \begin{pmatrix} 0 \\ 0 \\ z \\ 1 \end{pmatrix}. \tag{3.4}$$

Sei dafür zunächst die Matrix D eingeführt:

$$D := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.5}$$

mit (3.2) folgt:

$$(G_0 \cdot GC_r \cdot D)^{-1} \cdot c = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{3.6}$$

$(G_0 \cdot GC_r \cdot D)$ beschreibt somit ein Koordinatensystem mit dem Ursprung in c. Für jede beliebige homogene Rotationsmatrix $R \in \mathbb{R}^{4\times4}$ mit $R \cdot (0,0,0,1)^T = (0,0,0,1)^T$ gilt somit

$$R^{-1} \cdot (G_0 \cdot GC_r \cdot D)^{-1} \cdot c = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{3.7}$$

Für jedes $z \in \mathbb{R}$ und

$$Z := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

gilt demnach:

$$Z^{-1} \cdot R^{-1} \cdot (G_0 \cdot GC_r \cdot D)^{-1} \cdot c = \begin{pmatrix} 0 \\ 0 \\ z \\ 1 \end{pmatrix}. \tag{3.9}$$

Sei $t' \in \mathbb{R}^4$ ein beliebiger homogener Vektor, dessen Greiferpose mit K berechnet werden soll. Sei

$$t := (G_0 \cdot GC_r \cdot D)^{-1} \cdot t' \tag{3.10}$$

die betrachtete Position relativ zum Kalibrierungsobjekt. Es gilt eine Rotationsmatrix $R \in \mathbb{R}^{4\times4}$ und ein $z \in \mathbb{R}$ für Z zu finden (siehe Abb. 3.1), für die die Bedingung (3.3) gilt, also nach Umformung:

$$Z^{-1} \cdot R^{-1} \cdot t = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \tag{3.11}$$

was gleichzeitig für

$$K : t' \mapsto G_0 \cdot GC_r \cdot D \cdot R \cdot Z \cdot (GC_r)^{-1} \tag{3.12}$$

die Bedingung (3.4) erfüllt. Dies lässt sich durch einfaches Einsetzen verifizieren:

$$\begin{aligned} (GC)_r^{-1} \cdot K(t')^{-1} \cdot c &= (GC)_r^{-1} \cdot (G_0 \cdot GC_r \cdot D \cdot R \cdot Z \cdot (GC_r)^{-1})^{-1} \cdot c \\ &= Z^{-1} \cdot R^{-1} \cdot (G_0 \cdot GC_r \cdot D)^{-1} \cdot c. \end{aligned} \tag{3.13}$$

Mit (3.7) gilt dann

$$(GC)_r^{-1} \cdot K(t')^{-1} \cdot c = Z^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ z \\ 1 \end{pmatrix}, \tag{3.14}$$

was von Bedingung (3.4) gefordert wurde.

Somit bleibt zur hinreichenden Definition von K entsprechendes R und z zu definieren, so dass (3.11) erfüllt ist. Diese Herleitung lässt sich mittels zweier Givens Rotationen

realisieren. Dies ist ausführlich in Anhang A.2 beschrieben. Die resultierende Matrix ist die Komposition dieser zwei Givens Rotationen und ist gegeben durch:

$$R := \begin{pmatrix} \frac{t_3}{\sqrt{t_1^2+t_3^2}} & 0 & \frac{t_1}{\sqrt{t_1^2+t_3^2}} & 0 \\ 0 & \frac{\sqrt{t_1^2+t_3^2}}{\sqrt{t_2^2+(t_1^2+t_3^2)}} & \frac{t_2}{\sqrt{t_2^2+(t_1^2+t_3^2)}} & 0 \\ -\frac{t_1}{\sqrt{t_1^2+t_3^2}} & -\frac{t_2}{\sqrt{t_2^2+(t_1^2+t_3^2)}} & \frac{\sqrt{t_1^2+t_3^2}}{\sqrt{t_2^2+(t_1^2+t_3^2)}} \cdot \frac{t_3}{\sqrt{t_1^2+t_3^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.15}$$

sowie das gesuchte Skalar durch

$$z := \|t\|_h \tag{3.16}$$

gegeben ist. Somit kann für den beliebig angegebenen homogenen Vektor t' eine Greiferpose angegeben werden, dessen Ursprung in t' liegt und in der (nach dem linearen Modell) die Kamera das Kalibrierungsmuster fokussiert.

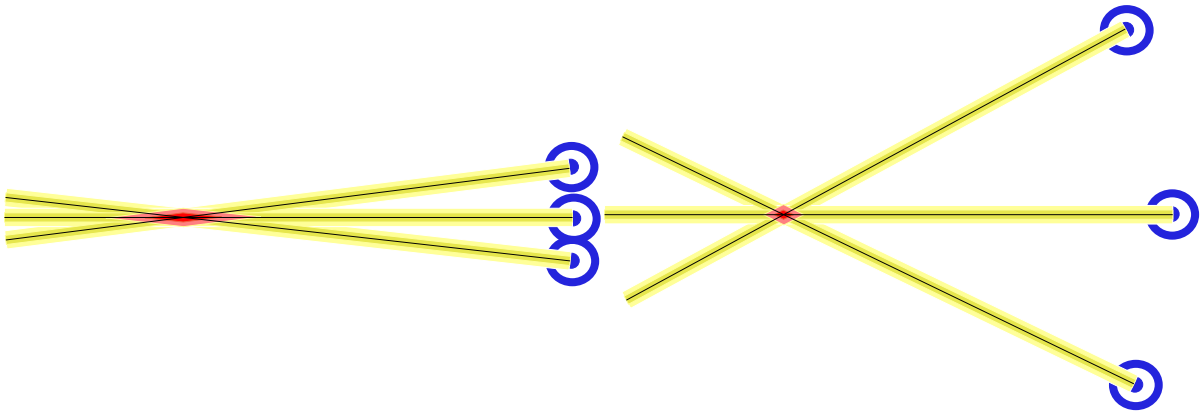


Abbildung 3.2: Betrachtung der Sichtstrahlen mit Varianzen im Kamerakoordinatensystem. Links: Kleine Einfallswinkel führen zu einer starken Varianz in der Positionsbestimmung des optischen Zentrums in Blickrichtung (roter Bereich). Rechts: Maximal große Einfallswinkel sorgen für eine minimale Empfindlichkeit gegenüber den Varianzen der Sichtstrahlen.

3.2 Maximierung des Einfallswinkels

Nach dem in Kapitel 3.1 beschriebenen Verfahren können nun beliebig viele auf das Kalibrierungsmuster ausgerichtete Posen generiert werden. In jeder der Posen wird dann eine Aufnahme gemacht und das Kalibrierungsobjekt im Bild lokalisiert. Diese Daten werden daraufhin zur Formulierung eines Optimierungsproblems verwendet. Jedoch führt die permanente Fokussierung des Kalibrierungsobjekts durch die Kamera zu Stabilitätsproblemen:

- Wenn bei der Betrachtung im Kamerakoordinatensystem alle Sichtstrahlen nahezu parallel und nahe aneinander verlaufen, so ist die Bestimmung des Schnittpunktes, also des optischen Zentrums, instabil. Wie in Abb. 3.2 grafisch dargestellt ist, führen größere Einfallswinkel zu einer stabileren Bestimmung des optischen Zentrums. Der Posengenerator erzeugt jedoch möglichst geringe Abweichungen zwischen optischer Achse und Sichtlinie (Kamera-Kalibrierungsmuster). Dadurch konzentrieren sich die resultierenden Musterpositionen im Zentrum des Kamerabildes. Dies erschwert die Lokalisierung des optischen Zentrums entlang der optischen Achse.
- Die Bestimmung der Linsenverzerrungsparameter kann nicht auf Basis von Daten geschehen, die sich nur auf das Verhalten des Kamerasystems nahe der optischen Achse beziehen. Die Auswirkungen der Linsenverzerrung sind meist in der Mitte des Bildes gering und nehmen nach außen hin zu.

Um diesen Problemen entgegenzuwirken, ist es notwendig, den vollen Öffnungswinkel der Kamera zu verwenden, also die Objektposition im Kamerabild zu „streuen“. Diese Streuung lässt sich durch Manipulation der Orientierung der generierten Greiferposen realisieren. Für eine exakte Gleichverteilung der Objektpositionen im Bild ist es jedoch notwendig, die Anzahl der Posen zu kennen. Da jedoch im Vorwege keine Informationen über die Erreichbarkeit der einzelnen Posen durch die inverse Kinematik vorliegt, ist eine genaue Kalkulation nicht möglich.

Deaktivierte Winkelmaximierung Aktivierte Winkelmaximierung

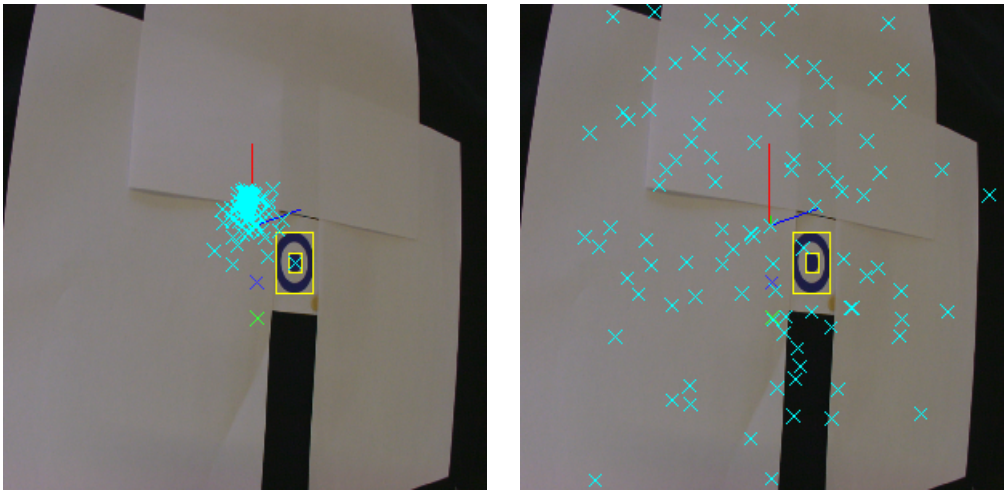


Abbildung 3.3: Links: Die Markierungen der Positionen des Kalibrierungsmusters eines Kalibriervorgangs mit deaktivierter Winkelmaximierung. Rechts: Die Verteilung der Kalibrierungsmusterpositionen mit aktivierter Winkelmaximierung.

Um trotzdem eine möglichst hohe Gleichverteilung der Objektpositionen im Bild zu erhalten, werden daher drei gleichverteilte Zufallsvariablen auf die Orientierungsparameter Yaw, Pitch und Roll aufaddiert. Die Skalierung dieser Variablen wird während des Abfahrens der Posen angepasst: Jede Aufnahme, in der sich das Objekt im inneren Drittel des Bildes befindet, sorgt für eine Inkrementierung der Streuungsstärke. Jede Aufnahme, die kein Objekt im Bild findet, sorgt für eine Dekrementierung. In allen anderen Fällen bleibt die Streuungsstärke konstant.
Dieses Verfahren führt zu einer gleichmäßigen Verteilung der Objektpositionen im Kamerabild, wie in Abb. 3.3 zu sehen ist.

Kapitel 4

Formulierung des Optimierungsproblems

Nachdem alle benötigten Bilder der generierten Posen aufgenommen und ausgewertet sind, ist es nun möglich, die gesuchten Parameter als Argument-Maximum einer Fitnessfunktion F zu definieren. Dafür ist es notwendig, ein zugrunde liegendes Umgebungsmodell zu definieren und dessen Parameter zu bestimmen. Diese Parameter seien im Parametervektor Θ zusammengefasst.
Das in der Fitnessfunktion F verwendete Umgebungsmodell bildet jeden zu testenden Parametervektor Θ auf seinen negativen Fehlerwert ab. Der Fehlerwert entspricht dabei den aufsummierten quadratischen Bildfehlern[1] des aus den Parametern resultierenden Modells. Der Fehler wird somit direkt an der Stelle minimiert, wo er detektiert wird: auf der Bildebene. Der Optimierungsalgorithmus liefert den Vektor Θ_{max}, für den der Bildfehler minimal ist.

4.1 Der Parametervektor Θ

Die Liste der Umgebungsvariablen eines Kamera-Roboter-Systems lässt sich in extrinsische und intrinsische Kameraparameter aufteilen.
Die extrinsischen Parameter beschreiben die Lage des Kamerakoordinatensystems zum Greiferkoordinatensystem und somit die Einträge der Matrix GC. Da GC eine homogene Transformationsmatrix ist, lässt sie sich über die folgenden sechs Parameter definieren:

- Translationen in x-, y- und z-Richtung
- Die Rotationen γ_x, γ_y und γ_z (yaw, pitch und roll)

Hinzu kommen die internen Eigenschaften der verwendeten Kamera, die intrinsischen Parameter:

[1] Als Bildfehler wird die euklidische Distanz zwischen der observierten Position des Kalibrierungsmusters im Bild und der durch das Modell berechneten Position bezeichnet.

- Öffnungswinkel der Kamera in x- und y-Richtung β_x^*, β_y^*
- Die in Kapitel 4.2 beschriebenen Parameter des Linsenverzerrungsmodells, gegeben durch die Verzerrungsfunktionen δ_u^* und δ_v^*

Zudem wird zur Bestimmung der Güte einer Schätzung die (vermutliche) Position des Mittelpunktes des Kalibrierungsobjekts c^* benötigt[2].
Θ ist somit definiert durch:

$$\Theta := \begin{pmatrix} x^* & y^* & z^* & \gamma_x^* & \gamma_y^* & \gamma_z^* & \beta_x^* & \beta_y^* & c_x^* & c_y^* & c_z^* & \delta_u^* & \delta_v^* \end{pmatrix}^T \tag{4.1}$$

Das Optimierungsproblem ist somit durch die Suche nach Θ_{max} beschrieben, mit

$$\Theta_{max} := argmax_{\Theta}(F(\Theta)). \tag{4.2}$$

4.2 Das Linsenverzerrungsmodell

Generell ist der in dieser Arbeit vorgestellte Algorithmus unabhängig von dem verwendeten Linsenverzerrungsmodell. Zur Evaluierung des Verfahrens und für die realen Tests wurden jedoch einige Verzerrungsmodelle verwendet. Neben den polynomiellen Standardmodellen [14] wurden auch die Verzerrungsmodelle von Heikkilä und Silvén [8] und Weng, Cohen und Herniou [23] implementiert. Das Modell von Weng, Cohen und Herniou stellte sich während der praktischen Tests als stabiles Modell heraus, während sich das Modell von [8] als ungeeignet erwies. Die polynomiellen Modelle und das Verfahren von Weng, Cohen und Herniou seien daher hier definiert. Ein ausführlicher Test und Vergleiche der Verzerrungsmodelle sind in Kapitel 7.6 zu finden.

4.2.1 Verzerrung nach Weng, Cohen und Herniou

Das Verzerrungsmodell von Weng, Cohen und Herniou modelliert die folgenden Linsenverzerrungseffekte:

- Radiale Bildverzerrung
- Tangentiale Verschiebung des Mittelpunktes
- Radiale Verschiebung des Mittelpunktes
- Prismaverzerrung

[2]Es ist nicht zwingend notwendig, c zu schätzen, da Berechungsvorschriften existieren, c auf Basis der anderen Komponenten von Θ zu berechnen. Warum diese nicht zur Anwendung kommen, wird ausführlich in Kapitel 6.3 erläutert.

Aus der Kombination der Verzerrungsvorschriften ergibt sich dann die Gesamtverzerrungsformel, aus der dann alle Terme der Ordnung 3 und höher gestrichen werden, da sie von den Autoren in [23] als vernachlässigbar angesehen werden. Sei die unverzerrte Position im Bild gegeben durch $u, v \in [-1, 1]$, so modelliert das Modell für die u- und $v-$ Komponente eine Verzerrung von

$$\delta_u(u, v) = s_1(u^2 + v^2) + 3p_1u^2 + p_1v^2 + 2p_2uv + k_1u(u^2 + v^2) \tag{4.3}$$

und

$$\delta_v(u, v) = s_2(u^2 + v^2) + 2p_1uv + p_2u^2 + 3p_2v^2 + k_1v(u^2 + v^2). \tag{4.4}$$

für die Verzerrungsparameter $s_1, s_2, p_1, p_2, k_1 \in \mathbb{R}$. In der Implementierung kommen die etwas effizienteren und zu 4.3 und 4.4 äquivalenten Formeln

$$\delta_u(u, v) = (g_1 + g_3)u^2 + g_4uv + g_1v^2 + k_1u(u^2 + v^2) \tag{4.5}$$

$$\delta_v(u, v) = g_2u^2 + g_3uv + (g_2 + g_4)v^2 + k_1v(u^2 + v^2) \tag{4.6}$$

zum Einsatz, die man durch die Substitutionen $g_1 = s_1 + p_1$, $g_2 = s_2 + p_2$, $g_3 = p_2$ und $g_4 = 2p_2$ erhält.

4.2.2 Das polynomielle Verzerrungsmodell

Im Unterschied zu der expliziten Formulierung einzelner Verzerrungsarten durch das Modell von Weng, Cohen und Herniou, versucht das polynomielle Verzerrungsmodell [14] durch Interpolation eine Entzerrungsgrundlage zu liefern. Ein Standardmodell der polynomiellen Verzerrungsfunktion ist das Polynom sechsten Grades:

$$\delta_u(u, v) = sign(u) \cdot \left(s_1 \cdot \sqrt{u^2 + v^2}^2 + s_2 \cdot \sqrt{u^2 + v^2}^4 + s_3 \cdot \sqrt{u^2 + v^2}^6\right) \tag{4.7}$$

$$\delta_v(u, v) = sign(v) \cdot \left(s_1 \cdot \sqrt{u^2 + v^2}^2 + s_2 \cdot \sqrt{u^2 + v^2}^4 + s_3 \cdot \sqrt{u^2 + v^2}^6\right) \tag{4.8}$$

Um eventuellen Verschiebungen des Linsensystems oder des CCD-Chips Rechnung zu tragen, kann vor der Anwendung der radialen Verzerrung ein Versatz (o_u, o_v) auf die Bildkoordinaten addiert werden:

$$\delta_u(u, v)' = \delta_u(u + o_u, v + o_v) \tag{4.9}$$

$$\delta_v(u, v)' = \delta_v(u + o_u, v + o_v) \tag{4.10}$$

Da der Versatz des CCD-Chips ähnliche Auswirkungen auf das Gesamtmodell hat wie eine leichte Drehung der Kamera, besteht bei der Verwendung dieses Verzerrungsmodells die Gefahr, dass sich die Genauigkeit der extrinsischen Parameter verringert. Aus diesem Grund wird im Vergleichstest der Linsenverzerrungsmodelle in Kapitel 7.6 das polynomielle Verzerrungsmodell separat mit und ohne den CCD-Versatz getestet.

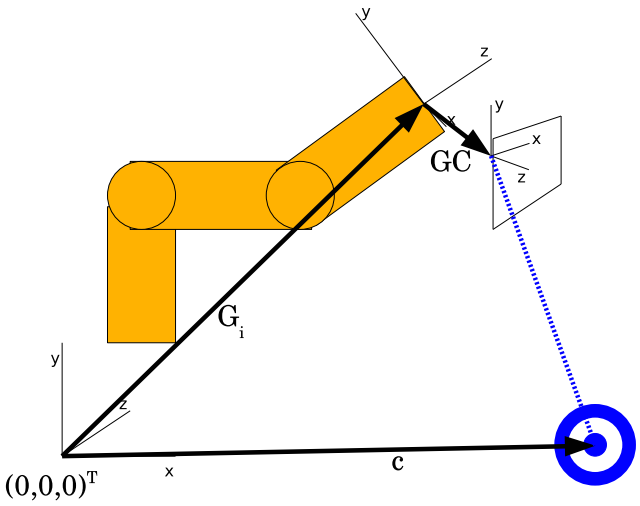


Abbildung 4.1: Die Berechnung des Fehlers in einem einzelnen Bild wird auf Basis der doppelten Darstellbarkeit der Position des Kalibrierungsobjekts durchgeführt.

4.3 Die Fitnessfunktion F

Die Fitness eines Parametervektors Θ wird durch Einsetzen der einzelnen Komponenten in das zugrunde liegende Umgebungsmodell berechnet[3]. Ein über Θ parametrisiertes Modell liefert zu jeder Greiferpose eine Bildposition, an der, unter Annahme dieser Parameter, das Kalibrierungsobjekt beobachtet werden müsste. Für jede gespeicherte Greiferpose-Bild-Kombination wird dann die tatsächlich beobachtete Position des Kalibrierungsobjekts mit der berechneten Bildposition verglichen, und die euklidische Distanz auf der Bildebene definiert den Bildfehler. Die Quadrate der Bildfehler aller Bilder werden aufsummiert und negiert, um die Fitness zu erhalten. Der Bildbereich von F ist somit $\mathbb{R}_{\leq 0}$, wobei im Optimalfall $F(\Theta_{max}) = 0$ gilt.

Abb. 4.1 veranschaulicht, wie aus der Festlegung der Einträge des Parametervektors und damit der Kamerapose und des Kalibrierungsobjekts unmittelbar die Position des Sichtstrahls durch die Bildebene der Kamera festgelegt wird.

Für jede Greiferposition G_i ist die Position des Kalibrierungsobjekts im Kamerakoordinatensystem gegeben durch

$$(G_i \cdot GC)^{-1} \cdot c. \tag{4.11}$$

Zur Modellierung des Lochkameramodells seien die Funktion P_u und P_v eingeführt:

$$P_u : \mathbb{R}^4 \to \mathbb{R}, \begin{pmatrix} x \\ y \\ z \\ s \end{pmatrix} \mapsto \frac{x}{z} \tag{4.12}$$

[3]Das Grundkonzept der Fitnessfunktion basiert auf den in Kapitel 6.4 aufgeführten Beobachtungen.

$$P_v : \mathbb{R}^4 \to \mathbb{R}, \begin{pmatrix} x \\ y \\ z \\ s \end{pmatrix} \mapsto \frac{y}{z} \tag{4.13}$$

Die Parameter in Θ,

$$\Theta = \begin{pmatrix} x^* & y^* & z^* & \gamma_x^* & \gamma_y^* & \gamma_z^* & \beta_x^* & \beta_y^* & c_x^* & c_y^* & c_z^* & \delta_u^* & \delta_v^* \end{pmatrix}^T \tag{4.14}$$

werden zunächst zur Konstruktion der Schätzung von GC verwendet:

$$GC^* := \begin{pmatrix} \cos(\gamma_z) & \sin(\gamma_z) & 0 & 0 \\ -\sin(\gamma_z) & \cos(\gamma_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos(\gamma_y) & 0 & \sin(\gamma_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\gamma_y) & 0 & \cos(\gamma_y) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.15}$$

$$\cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\gamma_x) & \sin(\gamma_x) & 0 \\ 0 & -\sin(\gamma_x) & \cos(\gamma_x) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & x^* \\ 0 & 1 & 0 & y^* \\ 0 & 0 & 1 & z* \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.16}$$

Damit ist es möglich, die Position des Kalibrierungsmusters im Bild durch Anwendung des Lochkameramodells zu definieren:

$$u_i' := P_u((G_i \cdot GC^*)^{-1} \cdot c^*) \cdot \tan(\beta_x^*) \tag{4.17}$$

$$v_i' := P_v((G_i \cdot GC^*)^{-1} \cdot c^*) \cdot \tan(\beta_y^*) \tag{4.18}$$

Zusammen mit den in Kapitel 4.2 vorgestellten Funktionen zur Modellierung der Linsenverzerrung δ_u und δ_v lässt sich die erwartete Bildposition des Kalibrierungsmusters definieren durch:

$$u_i^* := u_i' + \delta_u^*(u_i', v_i') \tag{4.19}$$

$$v_i^* := v_i' + \delta_v^*(u_i', v_i') \tag{4.20}$$

Seien u_i und v_i die tatsächlich beobachteten Bildpositionen, dann ist der Bildfehler für ein Bild i gegeben durch

$$\sqrt{(u_i^* - u_i)^2 + (v_i^* - v_i)^2} \tag{4.21}$$

Die Fitness einer Schätzung Θ kann nun als negierte Summe aller durch Θ implizierten Bildfehlerquadrate formuliert werden:

$$F(\Theta) = -\sum_i \sqrt{(u_i^* - u_i)^2 + (v_i^* - v_i)^2}^2 \tag{4.22}$$

Für ein optimal gewähltes Θ gilt also in einem perfekten System $F(\Theta) = 0$.

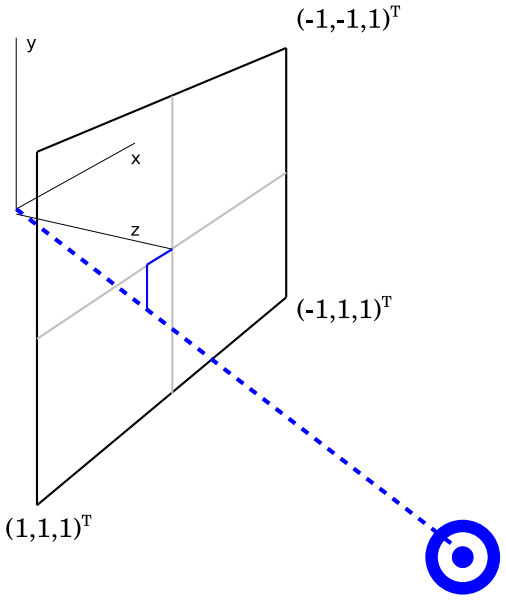


Abbildung 4.2: Zu jedem Bild gibt $(u_i, v_i), i \in N$ die beobachtete Objektposition als Abweichung von der z-Achse um die Bildkoordinaten an. Auf diese Weise fließen die Bilddaten in das zu lösende Gleichungssystem ein.

4.4 Optimierung

Die Optimierung des Fitnesswertes $F(\Theta)$ wird durch den globalen Optimierer CMA-ES durchgeführt. Bei CMA-ES handelt es sich um ein derandomisiertes Optimierungsverfahren, das eine multivariate Gaußsche Normalverteilung adaptiert zur Verteilung von Individuen im mehrdimensionalen Suchraum. Die Behandlung der Individuen geschieht dabei nach dem Prinzip evolutionärer Algorithmen. CMA-ES ist kein Quasi-NewtonVerfahren, jedoch ähnelt die Kovarianzmatrix-Adaption der Approximation der inversen Hessematrix dieser Optimiererklasse. CMA-ES ist in der Lage, unabhängig von der gegebenen Anfangsschätzung das absolute Minimum der in Kapitel 4.3 definierten Fitnessfunktion F zu ermitteln[4].
Die Linsenverzerrungsfunktionen δ_u und δ_v, die in Θ enthalten sind, werden für die Optimierung durch ihre Parameter ersetzt, so dass z. B. für das Verzerrungsmodell von Weng und Cohen fünf reelle Parameter anstelle der Verzerrungsfunktionen zu optimieren sind.
Für die Anzahl der zu verwendenden Individuen zur Optimierung wird in [6] der Wert $4 + 3 \cdot \log(n)$, n Anzahl der Dimensionen des Suchraums, als Empfehlung angegeben. Während der Testläufe stellte sich dieser Wert auch zur Optimierung von F als gute Wahl heraus. In über 95% aller Testläufe konvergierte CMA-ES zum korrekten Optimum, in allen anderen konnte durch Analyse des Fitnesswertes ein erneuter Optimierungsvorgang initiiert werden. Ein Optimierungsvorgang dauerte durchschnittlich weniger als 30 Sekunden. Die mit CMA-ES ermittelten Testergebnisse sind in Kapitel 7 zu finden.
Argumente, die zur Wahl von CMA-ES als Optimierer geführt haben, sind in Kapitel 6.2 aufgeführt.

[4]Dies wurde im Zuge der in 7.1 beschriebenen Tests hinreichend belegt.

Kapitel 5

Implementierung

Der Algorithmus wurde zusammen mit einer Steuerungs-, Visualisierungs- und Simulationssoftware in der Programmiersprache C++ implementiert. Die hierfür entwickelte Software HECTOR (***H**and **E**ye **C**alibration **T**oolkit for **O**ne-arm **R**obotsystems*) wurde auf Basis der Bildverarbeitungs- und Hardwareanbindungsbibliothek PACLib realisiert. Um bei der Auswertung der Testergebnisse Unterschiede zwischen physikalisch bedingten Unzulänglichkeiten der Hardware und Eigenschaften des Algorithmus zu analysieren, wurde das Roboter-Simulationsmodul RobSim der PACLib weiterentwickelt und in die Software HECTOR integriert. Im Folgenden werden die einzelnen Komponenten der Software kurz vorgestellt:

5.1 Die Perception Action Components Library

Die Perception Action Components Library (PACLib) ist eine Softwarebibliothek zur Erstellung von Programmen und Modulen rund um den Bereich der kognitiven Systeme. Sie umfasst Module zur Ansteuerung diverser Kamerasysteme und Sensoren sowie Pan-Tilt-Units und Roboter. Dazu kommen Anwendungen und Module aus dem Bereich der Bildverarbeitung, Optimierung und Mathematik.
Eine in der PACLib entwickelte Anwendung ist in der Lage, durch die Ansteuerung vielfältiger Hardware als Perzeptions-Aktions-Zyklus zu fungieren und die verarbeiteten Daten grafisch auf beliebigen Fenstersystemen (Windows / KDE / Gnome) darzustellen. HECTOR ist eine PACLib-Anwendung. Die PACLib wurde am Lehrstuhl Kognitive Systeme an der Christian-Albrechts-Universität zu Kiel entwickelt. Eine vollständige Dokumentation findet sich unter:

`http://www.ks.informatik.uni-kiel.de/~paclib/PACLibDoc_released/html/`

5.2 Der Robotersimulator RobSim

Das Modul RobSim der PACLib ist eine Bibliothek zur Simulation von stationären Roboterarmen. Beliebige Geometrien und Anordnungen von Linear- und Rotationsgelenken

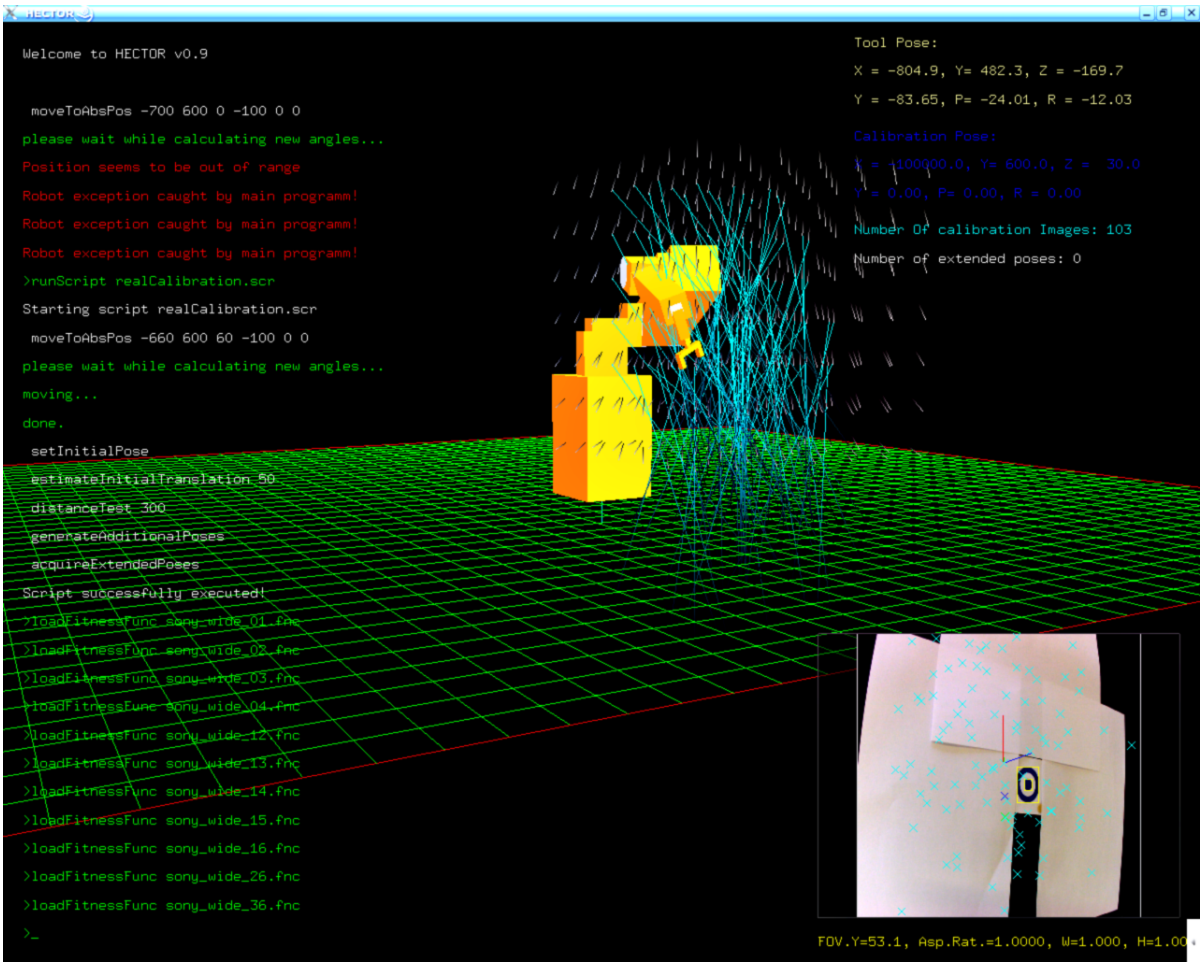


Abbildung 5.1: Screenshot von HECTOR während eines Kalibriervorgangs: Die Visualisierung ist simuliert, das Kamerabild (rechts unten) stammt von dem realen Roboter, der parallel die Kalibrierung durchführt.

können via OpenGL dargestellt und in Echtzeit gesteuert werden. Die Basisversion von RobSim wurde u.a. vom Autor dieser Arbeit entwickelt, und im Zuge der Implementierung von HECTOR wurde RobSim um eine effektive inverse Kinematik (auf Basis von CMA-ES) und die Fähigkeit, beliebige Kamera-Robotersysteme inklusive diverser Linsenverzerrungsmodelle zu simulieren, erweitert. Zusätzlich wurde die Fähigkeit der passiv-synchronen Simulation implementiert, die es erlaubt, die Bewegungen eines realen Roboterarms zeitgleich zu spiegeln und so parallel mit identischen Ausgangsdaten einen realen und einen simulierten Versuch durchzuführen.

5.3 Die Software HECTOR

Die im Zuge dieser Arbeit entwickelte Software HECTOR (***H**and **E**ye **C**alibration **T**oolkit for **O**ne-arm **R**obotsystems*) wurde zur Unterstützung des Forschungs- und Entwicklungsprozesses für den Kalibrierungsalgorithmus entwickelt.
Mit HECTOR ist es möglich, die Vorteile von realen und simulierten Tests zu kombinieren. So können bei einem realen Testlauf die geplanten Posen und die ermittelten

Sichtlinien in Echtzeit in der parallel laufenden 3-D-Animation eingeblendet werden. Auf diese Weise können Mess- und Kalkulationsergebnisse in einer frei schwenk- und zoombaren Umgebung visualisiert und verifiziert werden. Die eingebaute Konsole ermöglicht das gleichzeitige Steuern des Roboters und Ausführen von Berechnungen und bietet zusätzlich komfortable Eigenschaften wie Autovervollständigung und Online-Hilfe.
Ein universelles Funktions-Management-System ermöglicht das einfache Erweitern der Funktionalität mit eigenem C++ Quelltext, welcher ebenfalls automatisch in die Software eingebunden wird und über eine dafür entwickelte Zwischenschicht Konsolenbefehle in bereits vorverarbeiteter, geparster Form erhält. Zusätzlich ist die Konsole skriptfähig, erlaubt also die komplette Automatisierung von Vorgängen mittels einfacher Textdateien. Die in Kapitel 7 beschriebenen Testläufe wurden durch ein jeweiliges Testskript vollkommen autonom durchgeführt. Laufende Prozesse können jederzeit manipuliert oder gestoppt werden, genauso wie ermittelte oder generierte Datensätze über die Konsole in Dateien zwischengespeichert und zu einem späteren Zeitpunkt wieder geladen werden können.
Eine speziell für HECTOR entwickelte Ereignis-Verwaltung erlaubt die gesteuerte serielle Ausführung von Befehlsketten. So kann sich z. B. eine 'Funktion zur Aufnahme eines Bildes an einer bestimmten Roboterpose' nach dem Anfahr-Befehl an den Roboter zur erneuten Ausführung bei Erreichen der Zielpose oder eines anderen Ereignisses anmelden. Fehlerbehandlungsroutinen können sich beispielsweise für die Ausführung bei bestimmten Exceptions anmelden. Da HECTOR komplett als Einzel-Thread-Anwendung programmiert wurde, ist es in der Lage, durch internes Scheduling die Ausführung einzelner Programmteile zu priorisieren.
Um aussagekräftige Simulatortests durchführen zu können, ist HECTOR in der Lage, Kameramodelle mit beliebigen Öffnungswinkeln, Seitenverhältnissen und Auflösungen des CCD-Chips zu simulieren. Zudem erlaubt ein Rauschgenerator das Simulieren von unsauberen Kameradaten und schlecht kalibrierten Robotersystemen.

5.4 Implementierung des Algorithmus

Der Kalibrierungsalgorithmus wurde parallel als eigenständige Applikation entwickelt sowie durch eine Reihe von Klassen und Skripte in HECTOR integriert. Die komplette Dokumentation und Teile des Quelltextes liegen dieser Arbeit in Form einer Doxygen-Dokumentation[1] bei.
Wie in Abb. 5.2 zu sehen ist, ist die Implementierung auf mehrere Klassen verteilt. Die Klasse *Calibration* stellt dabei die verwaltende Klasse dar, die die Kommunikation nach außen führt und die anderen Klasseninstanzen der Kalibrierung verwaltet. In der Klasse *Estimate* ist die in Kapitel 2 beschriebene Anfangsschätzung implementiert. Der Optimierer ist in der Klasse *CMA-ES* implementiert[2]. Die Implementierung des Optimierers existierte bereits vor der Entwicklung dieses Verfahrens und wurde lediglich in das Programm

[1] Doxygen ist ein in der PACLib verwendetes Dokumentationswerkzeug, das aus speziellen Kommentaren im Quelltext eine Dokumentation zusammenstellt. Weitere Informationen sind unter `www.doxygen.org` zu finden.

[2] Da es sich bei der Implementierung von CMA-ES um C-Quelltext handelt, ist CMA-ES keine Klasse im genauen Sinne, sondern eine instanzierbare Datenstruktur mit verwaltenden Funktionen.

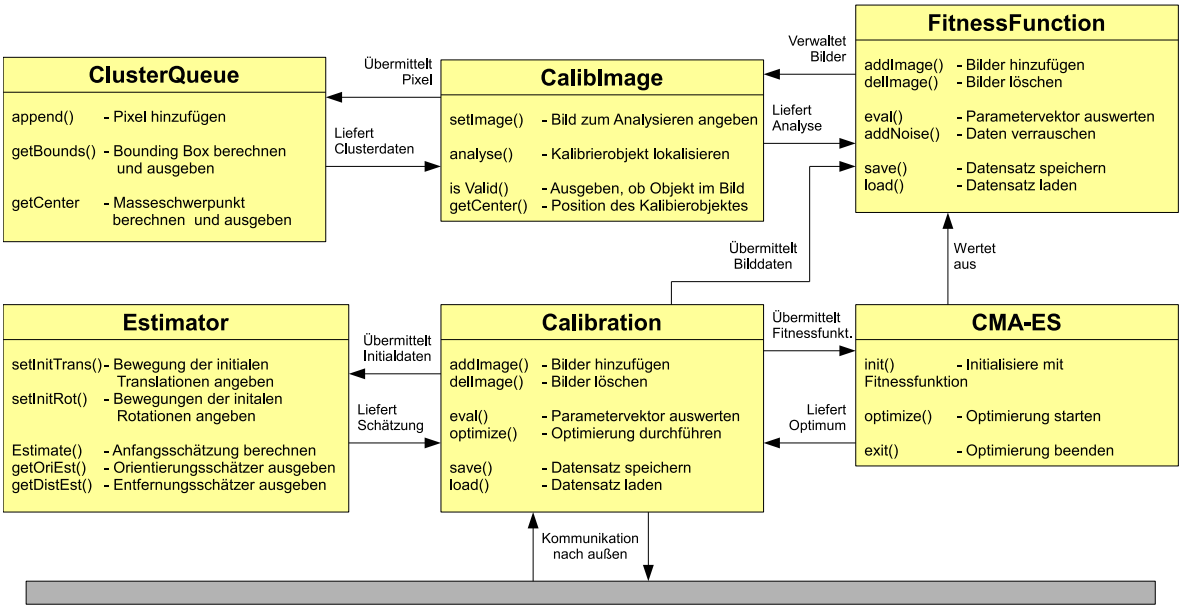


Abbildung 5.2: Eine vereinfachte Darstellung der Klassen und ihrer Funktionen zur Beschreibung der Aufgabenteilung

eingebunden. Die Klassen *CalibImage* und *ClusterQueue* implementieren die Bildverarbeitung. Ihre Funktion ist in Abschnitt 5.4.2 näher beschrieben. Die Klasse *FitnessFunction* stellt die Implementierung der Fitnessfunktion und des Modells dar, auf dessen Basis der Fitness-Wert für eine Parameterschätzung berechnet wird (siehe Kapitel 4.3). Sie verwaltet ebenfalls die Instanzen der *CalibImage*-Klasse, da das Umgebungsmodell und somit der Fitnesswert durch die einzelnen Bilder definiert werden.

Da eine vollständige Auflistung aller Funktionen und deren Erklärung den Rahmen dieser Arbeit sprengen würde, seien im Folgenden nur einige wichtige Implementierungsdetails aufgeführt:

5.4.1 Implementierung der Fitnessfunktion

Als einziger Ausschnitt des Quelltextes sei hier die Evaluierungsfunktion der *FitnessFunction*-Klasse aufgeführt, da dieser Prozedur eine sehr zentrale Rolle zukommt. Durch die Evaluierung eines Parametervektors wird sowohl das Umgebungsmodell und dessen Optimum definiert sowie die Auswertungsgeschwindigkeit der Gesamtoptimierung anhand der Komplexität dieser Funktion bestimmt. Aus diesem Grund wurde bei der Implementierung auf eine möglichst schlanke, effiziente Verarbeitung geachtet, was u.a. zu der Einbindung der Kalibrierungsobjektposition in den Parametervektor führte (siehe dazu auch Kapitel 6.3):

```
double FitnessFunc::eval(double* sigma){

   // Define the inverse of GC from sigma:
   CMatrix<double> GC_inv;
   GC_inv.SetDim(4,4); GC_inv.Id();
```

```
   // Calculate the rotational part of GC_inv
   CMatrix<double> rot,rotX,rotY,rotZ;
   rotZ.RotZ(-sigma[3] / 180.0 * M_PI);
   rotY.RotY(-sigma[4] / 180.0 * M_PI);
   rotX.RotX(-sigma[5] / 180.0 * M_PI);
   rot = (rotZ *rotY) * rotX;

   // Set the translation part of GC_inv and compile
   GC_inv(0,3) = - sigma[0];
   GC_inv(1,3) = - sigma[1];
   GC_inv(2,3) = - sigma[2];
   GC_inv = rot.Homog() * GC_inv;

   // Initialise error value
   double error_value = 0.0;

   // Set the calibration object vector c
   CVector<double> c; c.SetDim(4);
   c(0) = sigma[10];
   c(1) = sigma[11];
   c(2) = sigma[12];
   c(3) = 1.0;

   // Allocate vector to store the individual
   // G_i_inv * GC_inv * c for each image
   CVector<double> transformed_calib;

   // Cycle trough all images and sum up the error value
   for (int i = 0; i < numberOfImages; i++){
      // Calculate the calibration object location
      // in the assumed camera coordinate system
      transformed_calib = GC_inv * G_i_inv[i] * c;

      // Do the pinhole projection
      transformed_calib(1) /= transformed_calib(0);
      transformed_calib(2) /= transformed_calib(0);
      transformed_calib(1) *= sigma[6];
      transformed_calib(2) *= sigma[7];

      double d_x; // Image distortion x
      double d_y; // Image distortion y

      /* image distortion parameters d_x, d_y are calculated here */

      error_value += pow(y - y_intersec[i] + d_y ,2) * imageWeight[i];
      error_value += pow(x - x_intersec[i] + d_x ,2) * imageWeight[i];
      }
   return error_value;
}
```

5.4.2 Lokalisierung des Kalibrierungsmusters

Obwohl die Bildverarbeitung nicht Teil des Algorithmus ist, war es für die Implementierung notwendig, eine Methode zur Lokalisierung des Kalibrierungsmusters im Bild zu erstellen. Die Position des in 6.1 beschriebenen, blauen Kalibrierungsobjekts soll nach Möglichkeit subpixelgenau ermittelt werden. Die Verarbeitungsdauer sollte dabei sehr kurz sein. Zu diesem Zweck wurde ein Verfahren zur Lokalisierung in $O(n), n$ Anzahl der Bildpixel, entwickelt. Im Folgenden sind die einzelnen Schritte aufgeführt:

1. **Generierung eines Binärbildes zur Markierung der farbigen Pixel:** Ein Element des Binärbildes wird auf 1 gesetzt, falls das entsprechende Pixel im Ausgangsbild eine Farbe aus dem Bereich um die Kalibrierungsmusterfarbe im HSV-Raum besitzt, andernfalls wird es auf 0 gesetzt.

2. **Zusammenfassung benachbarter blauer Pixel:** Eine serielle Überprüfung aller Pixel startet für jedes Pixel mit Wert 1 eine Breitensuche, welche alle sich in der Zusammenhangskomponente[3] befindende Pixel an eine Queue-Struktur anhängt. Sobald ein Pixel in einer Queue registriert ist, wird sein Wert im Binärbild auf 0 gesetzt. Auf diese Weise wird jedes Pixel maximal zweimal behandelt.

3. **Ermittlung der Clusterdaten:** Die Pixel jedes Clusters werden nun seriell abgearbeitet und für jedes Cluster wird eine Bounding Box sowie der Masseschwerpunkt ermittelt. Kleine Cluster werden gelöscht.

4. **Auswertung der Clusterdaten:** Aus den ermittelten Clustern werden nun die zwei herausgesucht, von denen die Bounding Box des einen die Bounding Box des anderen komplett umgibt und in einem bestimmten Größenverhältnis steht. Der Masseschwerpunkt des inneren Clusters wird als Objektposition ausgegeben.

Im Simulator konnte so eine Genauigkeit der Lokalisierung im Bild erzielt werden, die unter 1/4 Pixelkantenlänge liegt.

5.4.3 Generierung eines normalverteilten Rauschens

Die für die Simulatortests benötigte Erzeugung eines normalverteilten Rauschens basiert auf der von Abramowitz und Stegun aufgestellten approximierten Inversen der kumulativen Verteilungsfunktion der Gaußschen Normalverteilung [1]. Demnach ist eine Zufallsvariable x approximativ normalverteilt um 0 in $\mathbb{R}$ mit Varianz $\sigma = 1$, falls es eine in $[-0.5, 0.5] \setminus \{0\}$ gleichverteilte Zufallsvariable x' gibt und

$$x = sign(x') \cdot \sqrt{\ln(\frac{1}{x'^2})} \tag{5.1}$$

gilt. Die multivariate Normalverteilung ist dann durch die separate Anwendung der Verteilung auf jede Dimension gegeben.

[3] Mit Zusammenhangskomponente sind die Pixel adressiert, die in der Topologie der Bildebene eine zusammenhängende Fläche an Pixeln ergeben, die im Binärbild mit einer 1 markiert worden sind.

Kapitel 6

Konzeptionelle Entscheidungen

In diesem Kapitel werden einige der wichtigen, im Laufe des Forschungs- und Entwicklungsprozesses getroffenen Entscheidungen erläutert:

1. Die Wahl des Kalibrierungsobjekts
2. Die Verwendung von CMA-ES als Optimierer
3. Die Einbindung der Objektposition in den Parametervektor Θ
4. Die Minimierung in der Bildebene

6.1 Wahl des Kalibrierungsobjekts

Zur Kalibrierung von Kameras und Kamera-Robotersystemen existieren drei Klassen von Kalibrierungsobjekten:

- Ebene Flächen mit Schachfeld- oder Punktmuster
- Farblich und/oder anhand der Form wiedererkennbare Punkt-Kalibrierungsobjekte
- Eine farblich heterogene Umgebung zur Verwendung von wiedererkennbaren Merkmalspunkten

Wie bereits in Kapitel 1.3 erwähnt, sollte das hier verwendete Kalibrierungsmuster aus Gründen der Flexibilität ein Punktmarker sein. Das bedeutet, dass die Bildverarbeitung lediglich eine Position pro Bild ausgeben darf und evtl. vom Bild entnommene Daten wie Ausrichtung und Größe des Kalibrierungsmusters vernachlässigt werden. Dies geschieht zum einen, um eine klare Abstraktion der Bildverarbeitung zu gewährleisten (da diese kein Teil des Algorithmus ist), zum anderen um die Vielseitigkeit des Verfahrens zu erhalten, da jegliche Zusatzinformation über evtl. variante und invariante Eigenschaften des Musters im Vorwege dem System zugeführt werden müssten.
Zum Testen und Validieren der Methode wurde für den Algorithmus eine Bildverarbeitung

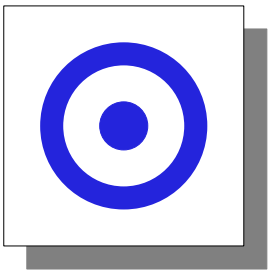


Abbildung 6.1: Die Form des Kalibrierungsmusters ist für die Kamerakalibrierung optimiert.

implementiert. Diese sei als Beispielverfahren im Folgenden kurz motiviert und beschrieben.

Zur Erläuterung der Wahl des Kalibrierungsobjekts sei zunächst die genaue Aufgabe des Musters betrachtet: Es gilt, einen abstrakten Punkt im Raum zu simulieren. Da die Eigenschaft eines Punktes, eine unendlich kleine Fläche zu besitzen, offensichtlich nicht nachempfunden werden kann, gilt es, eine möglichst kleine Fläche im Bild der Kamera zu beanspruchen. Je größer die Fläche im Bild ist, desto eher ist sie geneigt, Eigenschaften der Linsenverzerrung der Nachbarschaft des Punktes wiederzugeben und nicht der Linsenverzerrung des Punktes an sich. Ein kleiner Punkt einer bestimmten Farbe ist jedoch nur schwer stabil zu erkennen.

Aus diesem Grund wurde das in Abb. 6.1 dargestellte Kalibrierungsmuster verwendet. Segmentiert man das Bild in seine blauen Farbflächen, so tritt bei diesem Objekt der sonst sehr seltene Fall ein, dass ein Segment ein anderes komplett umschließt. Verwendet man diese Eigenschaft und die Relationen der Ausdehnung der Flächen zur Klassifizierung, so ist es möglich, trotz seiner geringen Größe den mittleren blauen Punkt im Bild zu detektieren und den Masseschwerpunkt der Pixel des Punktes als abstrahierten Punkt im Bild zu verwenden.
Während der Tests wurde so eine Lokalisierung mit einer durchschnittlichen Abweichung von 1/16 Pixellänge realisiert (siehe Kapitel 7.1). Details zur Implementierung sind in Kapitel 5.4.2 zu finden.

6.2 Verwendung von CMA-ES als Optimierer

Nahezu alle gängigen Verfahren zur automatischen Kalibrierung von Kamera-Robotersystemen arbeiten auf der Basis der Generierung eines lokalen Optimierungsproblems. Der Vorteil eines lokal optimierbaren Systems liegt in der Genauigkeit und der Geschwindigkeit der anwendbaren Verfahren zur Approximierung des Optimums. Die Nebenbedingung der lokalen Optimierbarkeit bei der Generierung des Problems stellt jedoch eine starke Einschränkung für den Algorithmus dar. Entweder ist ein Gleichungssystem zu generieren, welches nur ein einziges lokales Optimum besitzt, das somit auch global ist, oder es

muss eine Anfangsschätzung vorliegen, die nahe genug am globalen Optimum liegt, um eine Konvergenz des Optimierers zu gewährleisten. Diese Einschränkung wirkt sich auf die einzuholenden Daten (Greiferposen) sowie auf deren Verwertung bei der Bestimmung der Fitness aus, was auf Kosten der Robustheit und der Vielseitigkeit geht.
Bei dem in dieser Arbeit vorgestellten Verfahren sollten die Nachteile der lokalen Optimierung durch Verwendung eines globalen Optimierers wegfallen. Der in [5] aufgeführte Vergleich globaler Optimierer zeigt deutlich den Vorteil von CMA-ES gegenüber anderen gängigen Verfahren in den Bereichen Genauigkeit, Geschwindigkeit und Robustheit. Ebenso ist CMA-ES in der Lage, bei lokaler Optimierbarkeit schnell exakte Ergebnisse zu erzielen [3]. Die praktischen Tests an dem hier vorgestellten Verfahren ergaben, dass mittels CMA-ES in über 95% der Optimierungen das globale Optimum auch ohne Angabe einer Anfangsschätzung gefunden wurde. In allen übrigen Testläufen war der Punkt der Konvergenz so weit von dem Argumentenoptimum entfernt, dass mittels eines einfachen Schwellenwertes[1] für den Fitnesswert des Optimums eine erneute Optimierung ausgelöst werden kann. Auf diesem Weg konnte die Zahl der fehlerhaften Konvergenzen während der gesamten Testphase auf Null reduziert werden.

6.3 Einbindung der Objektposition in den Parametervektor Θ

Das in Kapitel 4 vorgestellte Umgebungsmodell dient als Basis für die Fitnessfunktion der Optimierung. Ein durch Θ parametrisiertes Modell wird von der Fitnessfunktion auf seine Konsistenz geprüft, und die berechnete Fitness entspricht dem negativen Fehler dieses Modells (siehe auch Kapitel 4.3).
Die intrinsischen und extrinsischen Parameter definieren so auch die Sichtstrahlen vom Ursprung der Kamera zum Zentrum des Kalibrierungsobjekts. Für eine beliebige Menge an Strahlen im dreidimensionalen Raum lässt sich ein Punkt finden, dessen Summe[2] der minimalen Abstände zu diesen Strahlen minimal wird. Somit existiert eine Separabilität innerhalb des Optimierungproblems zwischen Kamera-Robotsystemparametern und den Parametern für die Objektposition c_x, c_y und c_z. Es handelt sich sogar bei Letzterem nicht um ein Optimierungsproblem im eigentlichen Sinne, da eine direkte Berechnungsvorschrift für c_x, c_y und c_z für alle $\Theta \setminus \{c_x, c_y, c_z\}$ angegeben werden kann.
Die in Kapitel 4.3 beschriebene Fitnessfunktion führt pro Aufruf und pro Bild eine Matrix-Vektor-Multiplikation, eine Evaluierung der Linsenverzerrungsfunktionen und eine geringe, konstante Anzahl von Additionen und Multiplikationen durch. Somit ist die Gesamtlaufzeit der Modellevaluierung linear zur Anzahl der Bilder, also in $O(|I|)$. Die Komplexität der Berechnung des minimalen Abstands eines Punktes zu einer Menge von Geraden (in der durch die euklidische Norm induzierten Metrik) ist quadratisch zu der Anzahl der Geraden. Somit würde eine Fitnessfunktion, die eine Berechnung der Objektposition selbst durchführt, in $O(|I|^2)$ liegen. Diese Eigenschaft würde eine starke Einschränkung der Skalierbarkeit des Algorithmus bedeuten.
Bei der praktischen Erprobung beider Fitnessfunktionen zeigte sich, dass bereits bei einer

[1] Das Verhältnis der Fitness von korrektem und ermitteltem Optimum war im Fall einer Fehlkonvergenz stets höher als Faktor 500.

[2] Genauer: Der RMS, also die Wurzel der Summe der Quadrate der minimalen Abstände.

Anzahl von 30 Bildern die Optimierung mit der expliziten Berechnung der Objektposition mehr als zehnmal länger dauert als es der Fall ist, wenn die Objektposition von dem Optimierer selbst geschätzt wird. Eine geringere Zuverlässigkeit der Optimierung durch die von 15 auf 18 erhöhte Dimensionalität des Suchraumes konnte nicht festgestellt werden.

6.4 Minimierung in der Bildebene

Eine wichtige Frage bei der Entwicklung dieses Kalibrierungsverfahrens war, an welcher Stelle des Modells der Fehler zwischen realer Messung und parametrisiertem Modell erhoben werden soll, um bestmöglichste Ergebnisse zu erzielen. Bei der Erstellung einer Fitnessfunktion zur Bestimmung von Modellparametern ist es wichtig, das Wissen über Unsicherheiten des Modells, wie z.B. das Rauschen der Eingabedaten, in das Optimierungsproblem mit einfließen zu lassen.
Der bei Optimierungsproblemen übliche RMS-Fehlerwert wird auch in diesem Verfahren zur Mittelung der Fehler verwendet. Er hat die mathematische Eigenschaft, dass genau die Parameter das minimale Argument definieren, die zu einer minimalen Standardabweichung der einzelnen Fehlerwerte führen, nimmt man eine Normalverteilung dieser Werte an. Der RMS-Fehler gleicht in seinem Minimum der Standardabweichung der Verteilung, und das Argumentenminimum gleicht dem Erwartungswert der Verteilung.
Diese Tatsache sollte in die Erstellung einer Metrik zur Erhebung des Fehlers einfließen. Die einfachste Form, dieser Forderung gerecht zu werden, ist, den Fehler da zu minimieren, wo er entsteht: auf der Bildebene.
Dieses Konzept wurde in der in Kapitel 4.3 beschriebenen Fitnessfunktion umgesetzt. Diese Herangehensweise sorgt für einen kleinst-möglichen Fehler, sind die Bilddaten mit einem normalverteilten Rauschen belegt, was sich ebenfalls in den praktischen Tests (Kapitel 7.7) bestätigt.

Kapitel 7

Tests

Um einen Einblick in die Leistungsfähigkeit des Algorithmus zu gewinnen, wurden simulierte und praktische Testläufe durchgeführt. Insbesondere dienten die Tests dazu, folgende Fragen zu beantworten:

- Wie genau sind die Ergebnisse unter optimalen Bedingungen?
- Wie genau sind die Ergebnisse im praktischen Einsatz?
- Wie viele Bildaufnahmen werden für eine Kalibrierung benötigt?
- Welchen Einfluss hat Bildrauschen auf die Ergebnisse?
- Welchen Einfluss haben verrauschte Greiferposen auf das Ergebnis?
- Welchen Einfluss hat die Wahl des Linsenverzerrungsmodells auf die ermittelten extrinsischen Parameter?

Sämtliche zum Test benötigten Daten wurden mit der Software HECTOR erstellt. Einige Datenverarbeitungsschritte sowie die Visualisierung der Ergebnisse erfolgte mit MATLAB[1].

7.1 Testergebnisse im Simulator

Für eine Bewertung des Algorithmus unter optimalen Bedingungen wurden 40 simulierte Testläufe mit HECTOR (siehe Kapitel 5.3) durchgeführt. Die simulierte Kamera hatte eine Auflösung von 1024×1024 Pixel bei einem Öffnungswinkel von 51 Grad. Die Geometrie des simulierten Roboterarms war identisch mit der des für die praktischen Tests verwendeten Unimotion Stäubli RX90 Roboterarms. Das verwendete Linsenverzerrungsmodell entsprach dem in Kapitel 4.2 beschriebenen Modell von Weng, Cohen und Herniou [23].

[1] MATLAB ist ein Numerik-, Simulations- und Visualisierungsprogramm. Es stellt eine Hochsprache zur einfachen Implementierung numerischer Algorithmen zur Verfügung. Weitere Informationen sind unter `www.mathworks.com/products/matlab` zu finden.

Extrinische Parameter (Simulation):

Parameter	z	y	x	γ_y	γ_p	γ_r
Einheit	mm	mm	mm	Grad	Grad	Grad
RMS	0.063	0.017	0.021	0.0014	0.027	0.025

Tabelle 7.1: Durchschnittlicher RMS-Fehler der extrinsischen Parameter in den simulierten Testläufen

Intrinsische Parameter (Simulation):

Parameter	β_x	β_y	g_1	g_2	g_3	g_4	k_1
Einheit	Grad	Grad	-	-	-	-	-
RMS	0.0054	0.0069	0.00009	0.00007	0.00008	0.00010	0.00011

Tabelle 7.2: Durchschnittlicher RMS-Fehler der intrinsischen Parameter unter Verwendung des Verzerrungsmodells nach Weng und Cohen in den simulierten Testläufen

Da innerhalb der Simulation die Ground Truth Daten zur Verfügung stehen, kann der Fehler über alle Testläufe als RMS (Wurzel der Mittelwerte der Fehlerquadrate) angegeben werden. Die einzelnen Fehlerwerte sind in Tabelle 7.1 und 7.2 aufgeführt.

Der durchschnittliche Wert der Fitnessfunktion für das Optimum lag über -10^{-6} bei durchschnittlich 70 Bildern, was einem durchschnittlichen Bildfehler von unter 0.00012 entspricht. Bei der verwendeten Auflösung der simulierten Kamera von 1024×1024 Pixel bedeutet dies eine Pixelgenauigkeit von

$$\frac{1024}{2} \cdot 0.00012 \approx 0.06 \tag{7.1}$$

bei der Lokalisierung des Kalibrierungsmusters (etwa 1/16 der Pixelkantenlänge), was eine Bestätigung der Güte des verwendeten Kalibrierungsmusters ist. Durch die Verwendung höherer Auflösungen könnten noch genauere Kalibrierungsergebnisse generiert werden, jedoch zeigt sich in den folgenden Tests, dass Auswirkungen der Diskretisierung durch den CCD-Chip nicht den limitierenden Faktor der Genauigkeit darstellen.

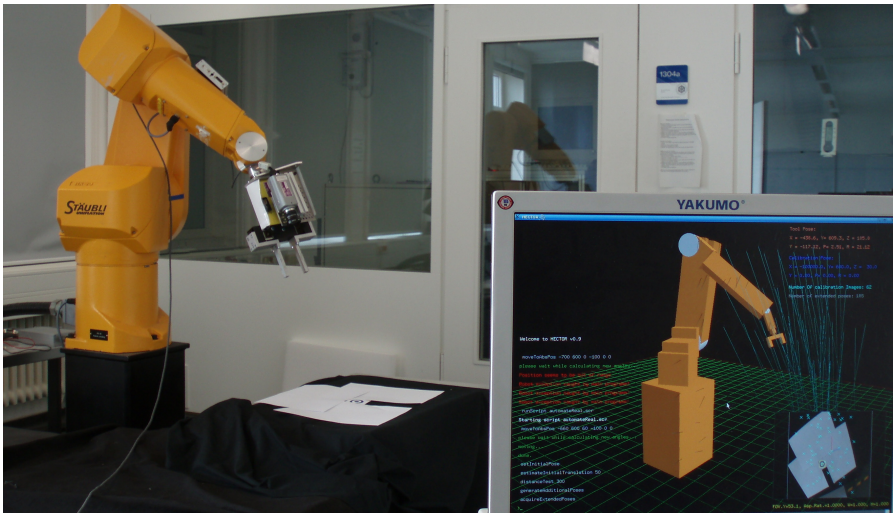


Abbildung 7.1: Die Testumgebung: Der Unimotion Stäubli RX90 mit montierter Sony DFW-X710 Kamera am Greifer. Die weiße Fläche um das Kalibrierungsmuster dient dem automatischen Weißabgleich.

Extrinsische Parameter (Real):

Parameter Einheit	z mm	y mm	x mm	γ_y Grad	γ_p Grad	γ_r Grad
SD	2.54	1.08	1.5	0.08	0.15	0.13

Tabelle 7.3: Standardabweichung der extrinsischen Parameter in den realen Testläufen

Intrinsische Parameter (Real):

Parameter Einheit	β_x Grad	β_y Grad	g_1 -	g_2 -	g_3 -	g_4 -	k_1 -
SD	0.57	0.28	0.0026	0.0040	0.0065	0.0040	0.0049

Tabelle 7.4: Standardabweichung der intrinschen Parameter unter Verwendung des Verzerrungsmodells nach Weng und Cohen in den realen Testläufen

7.2 Testergebnisse im realen Umfeld

Die realen Testläufe wurden mit einem Unimotion Stäubli RX90 durchgeführt. Die verwendete Kamera war eine Sony DFW-X710 Firewire Kamera mit einer Auflösung von 1024×768 Pixel und integriertem Hardware Bayer Filter. Das bei dem Test verwendete Linsenverzerrungsmodell entsprach dem Modell von Weng, Cohen und Herniou [23]. Es wurden 40 Testläufe durchgeführt mit jeweils ca. 100 Posen. Da für dieses Testszenario keine Ground Truth Daten verfügbar waren, werden in den folgenden Tabellen die Standardabweichungen über die 40 Testläufe angegeben. Die einzelnen Fehlerwerte sind in Tabelle 7.4 und 7.3 aufgeführt.

7.3 Tests mit variabler Anzahl von Bildern

Eine der wichtigsten Fragen für den praktischen Einsatz ist, wie viele Bilder zu einer vollständigen Kalibrierung benötigt werden. Durch die universale Form des Optimierungsproblems ist es nicht möglich, diese Frage auf dem analytischen Weg zu beantworten. Die einzig mögliche Antwort kann daher nur durch einen empirisch gewonnenen Richtwert gegeben werden. Zu diesem Zweck wurden simulierte und reale Tests mit einer variierenden Anzahl an Bildern durchgeführt. Die Ergebnisse des realen Tests entstammen 36 Testumgebungen (von 100 bis 10 Bildern in 2er Schritten) mit jeweils 40 Testläufen. Die Testergebnisse der Simulation beruhen ebenso auf 40 Testläufen für jedes der 36 Testumgebungen (von 80 bis 8 Bildern).

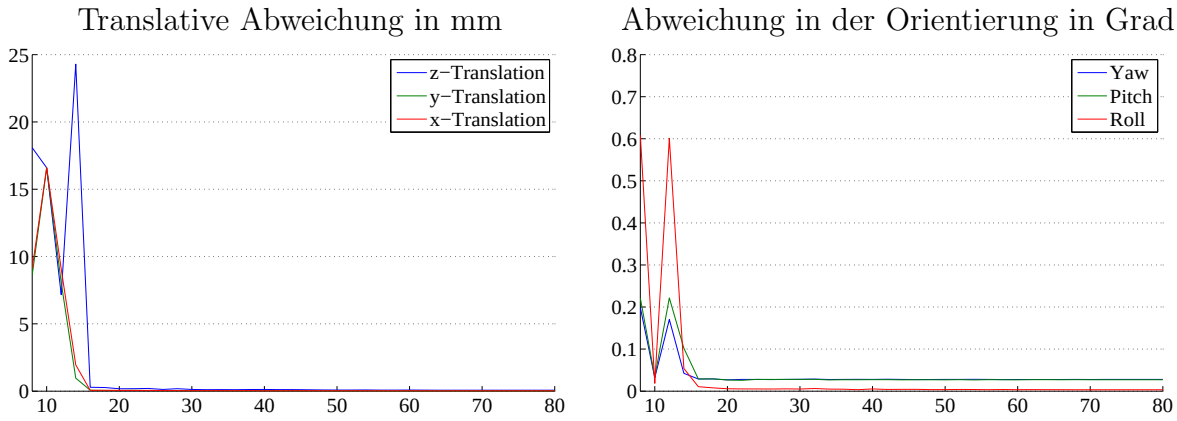


Abbildung 7.2: Links: Simulierte Testläufe mit einer variablen Anzahl von Bildern: Zu sehen ist die Anzahl der verwendeten Bilder gegen den RMS-Fehler der Ergebnisse. Bei weniger als 18 Bildern ist das Minimum, gegen das der Optimierer konvergiert, nicht mehr unbedingt das gesuchte Optimum.

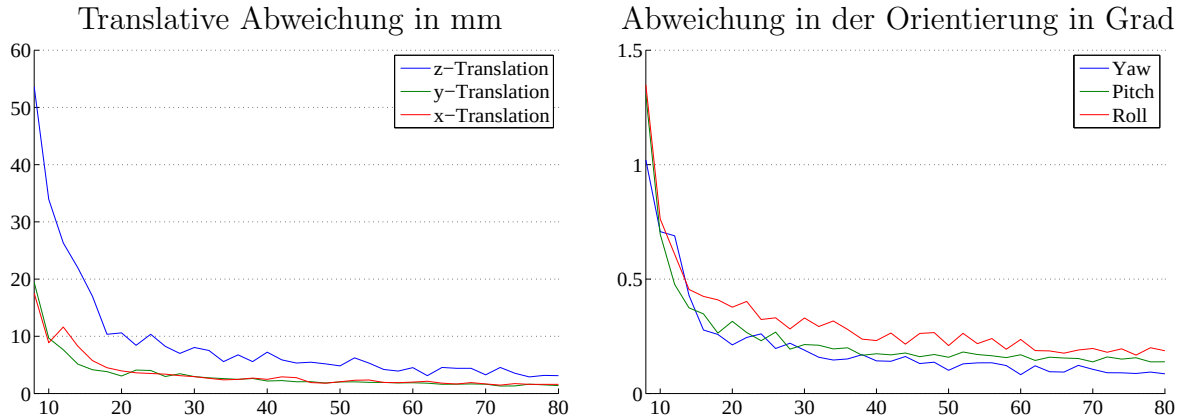


Abbildung 7.3: Links: Reale Testläufe mit einer variablen Anzahl von Bildern: Zu sehen ist die Anzahl der verwendeten Bilder gegen die Standardabweichung in den Ergebnissen. Eine Anzahl, ab der man von einem exakten Ergebnis ausgehen kann, gibt es nicht. Je mehr Bilder verwendet werden, desto genauer ist das Ergebnis.

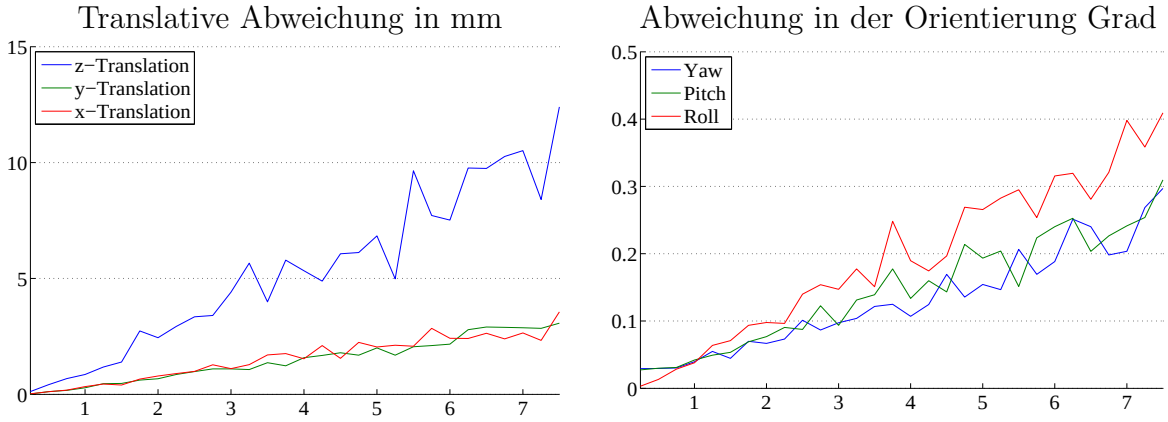


Abbildung 7.4: Links: Der Fehler der translativen Parameter in mm gegenüber künstlich verrauschten Bilddaten. Rechts: Die Fehler der Winkelbestimmung in Grad gegenüber künstlich verrauschten Bilddaten. Das Rauschen ist in Form der Varianz der Normalverteilung von der Abweichung in Pixellänge gegeben.

Während bei den Ergebnissen des Simulators eine klare Grenze zwischen korrekter und inkorrekter Konvergenz zu beobachten ist, liefern die Ergebnisse der realen Tests einen kontinuierlich ansteigenden Fehlerwert bei geringer werdender Bilderzahl. Diese Beobachtung ist auf das Rauschen in den realen Daten zurückzuführen, welches sich bei einer höheren Zahl von Bildern besser kompensieren lässt als bei wenigen Bildern.

7.4 Tests mit künstlich verrauschten Bilddaten

Zur Analyse der Stabilität gegenüber verrauschten Bilddaten wurden Simulatortests mit künstlich erzeugtem Rauschen durchgeführt. Da die Bildverarbeitung (Lokalisierung des Kalibrierungsmusters im Bild) nicht Teil des Algorithmus ist (siehe 1.3), werden nicht die Bilder selbst, sondern die von der Bildverarbeitung erhaltenen Positionsdaten verrauscht. Die addierten Bildfehler sind normalverteilt[2] bezüglich einer Standardabweichung σ. Es wurden für jeden Wert σ 40 Testläufe durchgeführt, und der entsprechende RMS-Fehler in den Parametern des Ergebnisses ermittelt. Abb. 7.4 zeigt den Fehler der extrinsischen Parameter. Die z-Achse entspricht der Sichtrichtung der Kamera.

Deutlich ist zu erkennen, dass die bereits in 7.1 und 7.2 beobachtete relative Ungenauigkeit der translativen Komponente in Blickrichtung der Kamera ebenfalls linear zu den übrigen translativen Parametern ansteigt. Diese Beobachtung deckt sich mit der in 3.2 beschriebenen Problematik.
Das Verhalten der intrinsischen Parameter hängt von dem verwendeten Linsenverzerrungsmodell ab. Die hier dargestellten Daten wurden unter Verwendung des Modells von Weng, Cohen und Herniou [23] erzeugt. In Abb. 7.5 sind die RMS-Fehler der Ergebnisse zu den entsprechenden σ Werten aufgeführt.

[2]Details zur Implementierung der Rauscherzeugung sind in 5.4.3 zu finden.

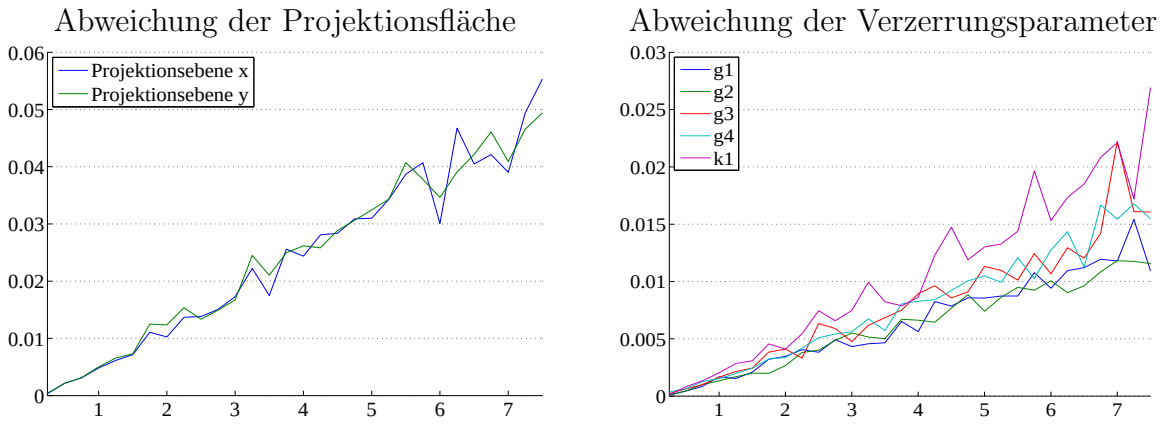


Abbildung 7.5: Links: Der RMS-Fehler der Skalierung der Einheitsbildebene gegenüber künstlich verrauschten Bilddaten. Rechts: Die Fehler der Linsenverzerrungsparameter. Die Verrauschung ist in Form der Varianz der Abweichung in Pixellänge gegeben.

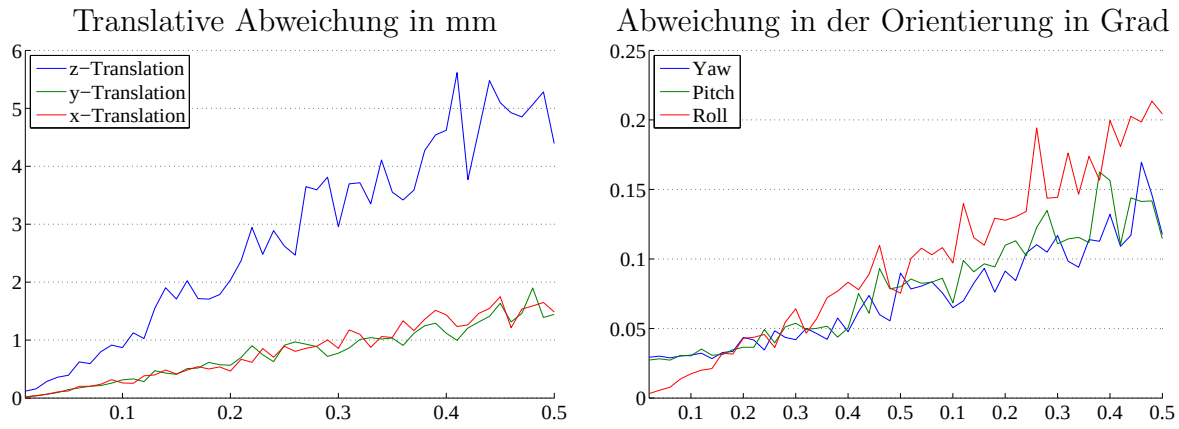


Abbildung 7.6: Links: Rauschen der Posedaten in mm (Varianz) gegenüber dem RMS-Fehler in den translativen extrinsischen Parametern. Rechts: Rauschen der Posedaten in mm (Varianz) gegenüber dem RMS-Fehler in der berechneten Orientierung in Grad.

7.5 Tests mit künstlich verrauschten Greiferposen

Um den Einfluss von Ungenauigkeiten dieser Größenordnung und das Verhalten der Kalibrierung bei nicht korrekt kalibrierten Robotersystemen zu beurteilen, wurden simulierte Tests mit künstlich verrauschten Greiferpositionsdaten durchgeführt. Fehlerhafte oder verrauschte Winkelstellungen des Roboters sind die Ursache für verrauschte Positionsdaten, der dadurch verursachte, effektive Fehler wird jedoch durch die Geometrie bestimmt. Insbesondere das durchschnittliche Verhältnis zwischen Winkel- und Positionsfehler hängt stark von Größe des Roboters ab. Für diese Tests wurden die Varianzen in Translation und Rotation der Posen in einem festen Verhältnis zueinander gesetzt:
1mm Translation = 0.25 Grad Rotation
Dieses Verhältnis entspricht dem eines Roboterarms mit einer Reichweite von 1.5 Metern im Durchschnitt.

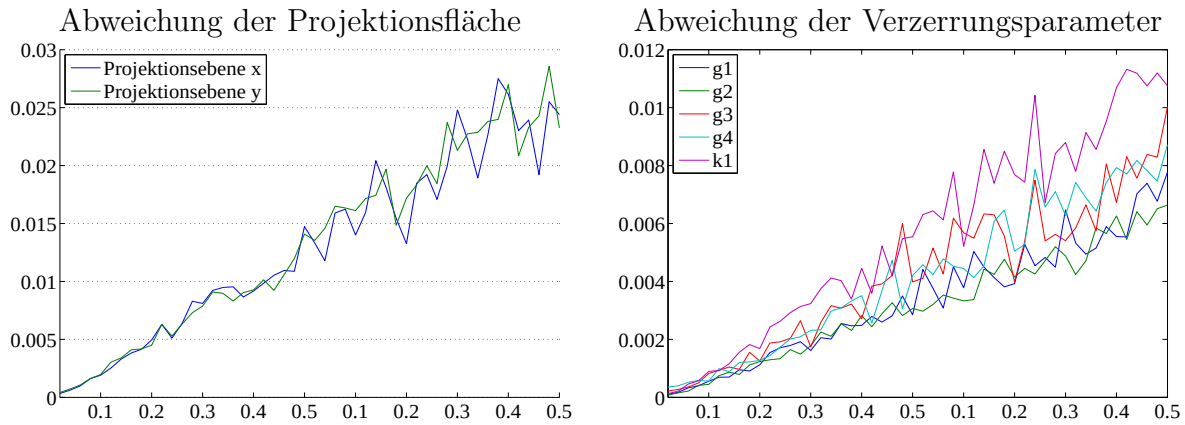


Abbildung 7.7: Links: Der RMS-Fehler der Skalierung der Einheitsbildebene gegenüber dem Rauschen in mm (Varianz) in den Posedaten. Rechts: Die Fehler der Linsenverzerrungsparameter gegenüber dem Rauschen in mm (Varianz) in den Posedaten.

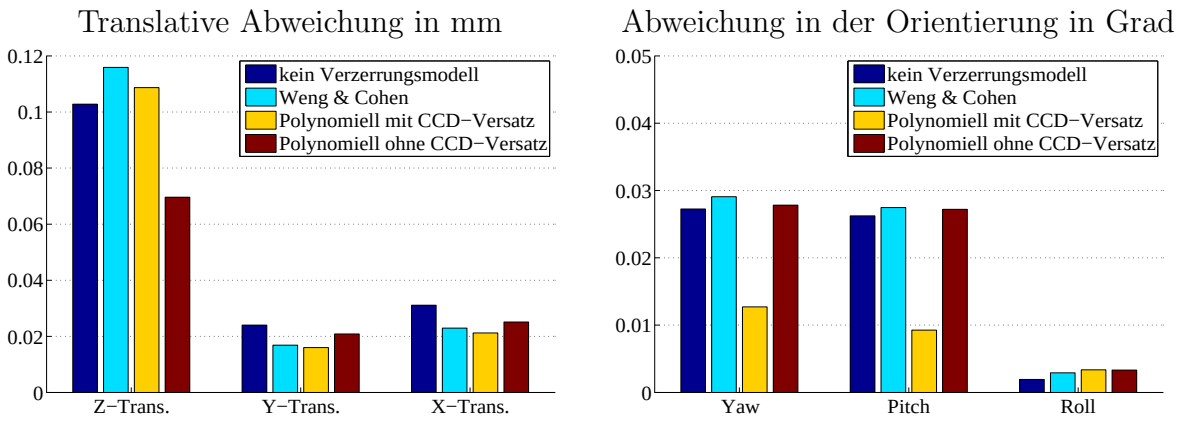


Abbildung 7.8: Simulierte Testläufe mit verschiedenen Linsenverzerrungsmodellen. Links: Die Modelle gegenüber ihrem translativen RMS-Fehler in mm. Rechts: Die Modelle gegenüber ihrem RMS-Fehler in der Orientierung in Grad.

Zu beobachten ist, dass bei verrauschten Positionsdaten die Abweichung in den translativen extrinsischen Parametern stark ansteigt. Moderne, kalibrierte Robotersysteme arbeiten jedoch mit Genauigkeiten um 0.02 mm. Schwankungen in dieser Größenordnung haben nur einen geringen Einfluss auf das Ergebnis. Die berechnete Orientierung bleibt indes auch bei verrauschten Posedaten sehr genau.

7.6 Vergleichstest der Linsenverzerrungsmodelle

Um einen Einblick in die Auswirkungen des Linsenverzerrungsmodells zu gewinnen, wurden 40 reale und 40 simulierte Tests mit je vier verschiedenen Linsenverzerrungsmodellen durchgeführt.

Gemessen wurde der RMS-Fehler der simulierten Testläufe (Abb. 7.8) und die Standard-

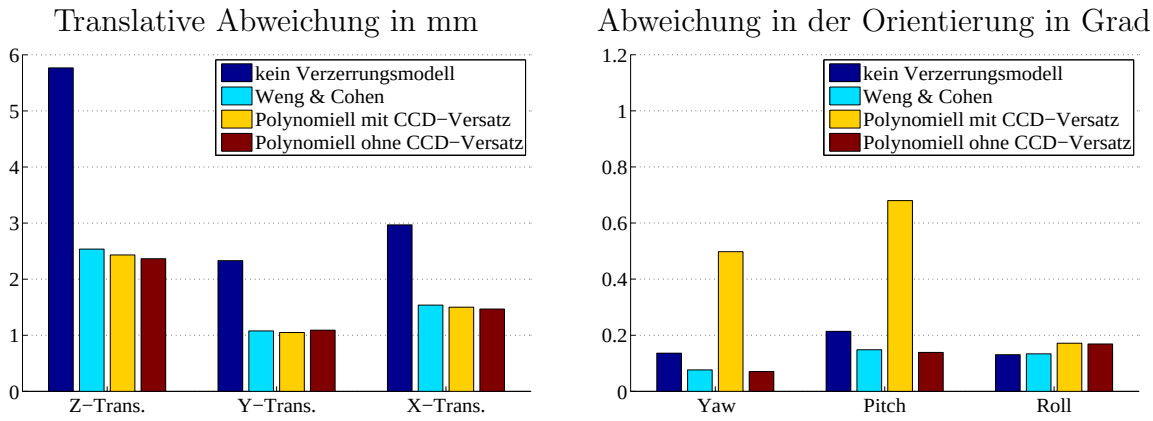


Abbildung 7.9: Reale Testläufe mit verschiedenen Linsenverzerrungsmodellen. Links: Die Modelle gegenüber ihrer translativen Standardabweichung in mm. Rechts: Die Modelle gegenüber ihrer Standardabweichung in der Orientierung in Grad.

abweichungen der realen Testläufe (Abb. 7.9) unter der Verwendung folgender Modelle:

1. Kein Linsenverzerrungsmodell, variable Öffnungswinkel in x- und y-Richtung
2. Linsenverzerrung nach dem Modell von Weng, Cohen und Herniou [23] (siehe Kapitel 4.2.1)
3. Polynomielles Linsenverzerrungsmodell sechsten Grades mit Berücksichtigung eines CCD-Chip-Versatzes und Scherung (siehe Kapitel 4.2.2)
4. Polynomielles Linsenverzerrungsmodell sechsten Grades mit Berücksichtigung der Bildscherung und ohne variablen CCD-Chip-Versatz (siehe Kapitel 4.2.2)

Zu beobachten ist, dass kein Linsenverzerrungsmodell in allen Bereichen den kleinsten Fehler generiert. Vielmehr ist die optimale Wahl des Linsenverzerrungsmodells abhängig von den Verrauschungseigenschaften. Das rechte Diagramm in Abb. 7.9 zeigt, dass ein ungeeignetes Linsenverzerrungsmodell einen Fehler erzeugen kann, der weit über dem einer Optimierung ohne Linsenverzerrungsmodell liegt. Die Verwendung einer CCD-Versatz Kompensierung in einer simulierten Umgebung ohne Linsenverzerrung ist offensichtlich in der Lage, auf die Aliasing-Eigenschaften des Renderes des Simulators einzugehen und diese zu kompensieren. Wie in Abb. 7.8 zu sehen, wird die Orientierung der Kamera etwas genauer bestimmt als bei den übrigen Verfahren, die einen Versatz des CCD-Chips nicht berücksichtigen. In den natürlich verrauschten, realen Testläufen zeigte sich jedoch, dass die Ähnlichkeit eines CCD-Versatzes mit der Drehung der Kamera zu Invarianzen dieser Parameter gegenüber dem System führen, was sich in einer erhöhten Standardabweichung der Yaw- und Pitch-Werte (Abb. 7.9) wiederspiegelt.

7.7 Vergleich mit existierenden Verfahren

Eine Gegenüberstellung von Kalibrierungsverfahren für Kamera-Robotersysteme ist ohne einen direkten Vergleich von Testergebnissen an identischen Systemen nicht sehr aus-

sagekräftig. Faktoren wie die Genauigkeit des Roboterarms, des Linsensystems und der Auflösung der Kamera haben einen großen Einfluss auf die Kalibrierung, wie in Kapitel 7.4 und 7.5 zu sehen ist. Aus diesem Grund ist es bei Veröffentlichungen üblich, zusammen mit den Ergebnissen realer Tests die Ergebnisse von simulierten Kalibriervorgängen anzugeben, da hier die Güte des Algorithmus unabhängig von den diversen physikalischen Unschärfen betrachtet werden kann. Seien daher nun die Testergebnisse des Simulators mit den Werten von zwei gängigen Verfahren verglichen, der Methode nach Tsai und Lenz [19] und dem Verfahren von Wei, Arbter und Hirzinger [22].

Als Grundlage dienen die in [22] aufgeführten Werte. Die Größe des translativen Fehlers der extrinsischen Parameter bei einem Bildrauschen von $\sigma_u = 0.5$ Pixel in der $u-$Komponente und $\sigma_v = 0.5$ Pixel in der $v-$Komponente wird in dieser Quelle als einziger Vergleichswert für die Genauigkeit der Verfahren aufgeführt. Vergleicht man diese Werte mit den für Kapitel 7.4 ermittelten Fehlerwerten des hier vorgestellten Verfahrens bei gleichem Rauschen, so ergibt sich folgende Tabelle:

RMS der Gesamt-Translation in mm:

Parameter	vorgestelltes Verfahren	Wei und Arbter	Tsai und Lenz
RMS	0.28	0.36	0.58

7.8 Zusammenfassung der Testergebnisse

Anhand der durchgeführten Tests ergibt sich ein Gesamtbild der Leistungsfähigkeit des Algorithmus und dessen Grenzen. Die Testergebnisse des Simulators (Kapitel 7.1) zeigen, dass unter optimalen Bedingungen sehr exakte Ergebnisse produziert werden. Das Verhalten des Algorithmus bei verrauschten Bildaufnahmen (Kapitel 7.4) und Posedaten (Kapitel 7.5) zeigt ein lineares Verhältnis zwischen Varianz des Rauschens und dem verursachten Fehler. Auffallend ist jedoch der rasch ansteigende Fehlerwert in den translativen extrinsischen Parametern bei verrauschten Pose-Daten. Aus diesem Grund sollte dieser Algorithmus, wie bereits in Kapitel 1.1 gefordert, nur auf einem bereits kalibrierten Robotersystem ausgeführt werden. Ein Berücksichtigen oder Kompensieren des Rauschens bei einem schlecht kalibrierten Robotersystem ist nicht möglich, da hierfür die Kenntnis der Robotergeometrie Voraussetzung ist. Die Eingabe der Geometrie vor der Anwendung des Algorithmus an einem Robotersystem steht jedoch im Konflikt mit der Forderung nach einem einfachen, sofort einsetzbaren System.

Die Ergebnisse der realen Testläufe (Kapitel 7.2) zeigen in der praktischen Anwendung einen größeren Fehlerwert als im Simulator. Dies liegt zum einem an dem nicht zu vermeidenden Rauschen in den Roboterposen des Stäubli RX90, zum anderen an dem durch den Bayer-Filter und der Linsenverzerrung verursachten Bildrauschen. Während in der Simulation jeweils ein vorher festgelegtes Verzerrungsmodell durch sein exakt inverses entzerrungsmodell kompensiert werden kann, gibt es für das reale Linsensystem kein exaktes Entzerrungsmodell, was zwangsweise zu einem, wenn auch meist minimalen, Bildfehler führt.

Die Testläufe mit einer variablen Anzahl von Bildern (Kapitel 7.3) ergaben in der Simulation, dass das Optimierungsproblem bis zu einer Anzahl von 18 Bildern ein eindeutig definiertes Optimum besitzt, welches vom Optimierer zuverlässig ermittelt wird. Jedoch zeigen die praktischen Tests, dass bei einer geringen Anzahl von Bildern die Auswirkung des Rauschens sehr groß ist, und mit einer höheren Anzahl von Bildern besser kompensiert werden kann. Für eine möglichst exakte Kalibrierung sollten daher möglichst viele Posen angefahren werden.

Im direkten Vergleich mit bestehenden Algorithmen zeigt sich neben der größeren Flexibilität des hier vorgestellten Verfahrens auch die höhere Genauigkeit (Kapitel 7.7).

Kapitel 8

Zusammenfassung

8.1 Ergebnisse

In dieser Arbeit wurde ein Algorithmus zur autonomen Kalibrierung eines Kamera-Robotersystems vorgestellt und getestet. Die Methode ist weitaus flexibler als bestehende Algorithmen:

- Die Verwendung nur eines Markerpunktes stellt einen minimalen Anspruch an die Einsatzumgebung des Kamera-Robotersystems. Verfahren, die mehrere Marker benötigen oder Kalibrierflächen verwenden, sind in diesem Punkt eingeschränkt.
- Die Anfangsschätzung ermöglicht es, selbst bei einer stark eingeschränkten Bewegungsfreiheit des Roboters ein Maximum an verschiedenen Posen anzufahren, da man im Gegensatz zu existierenden Verfahren nicht auf die schrittweise Annäherung angewiesen ist.
- Die Generierung der Posen durch die Anfangsschätzung im Vorwege erlaubt eine Kalibrierung selbst bei einer zeitweiligen Verdeckung des Kalibrierungsobjektes, die durch ungünstige Greiferbewegungen, Verdeckung durch bewegliche Fremdkörper oder sich ändernde Lichtverhältnisse verursacht werden können. Nahezu alle bestehenden Markerpunkt-Kalibrierungsverfahren arbeiten auf Basis einer permanenten Verfolgung des Objekts im Bild, was zu einer starken Anfälligkeit bezüglich zeitweiliger Verdeckungen führt.

Die Testergebnisse des hier vorgestellten Verfahrens zeigten eine hohe Präzision. Trotz des universellen modularen Aufbaus und der Verwendung eines globalen Optimierers sind die Resultate größtenteils genauer als die der bestehenden, lokal optimierende Verfahren. Durch die einfache Austauschbarkeit der Linsenverzerrungsmodelle ist es möglich, Ergebnisse weiterführender Forschung auf diesem Gebiet schnell und effizient in die bestehende Kalibrierungssoftware zu integrieren. Neue Erkenntnisse über die optimale Verteilung der Greifer- und Kameraposen im Raum können durch einfache Änderung der Parameter auf das Verfahren angewendet werden. Selbst die Verwendung eines komplett neuen Umgebungs- und Bewertungsmodells kann durch einen einfachen Austausch der

Fitnessfunktion realisiert werden.
Somit bietet die hier vorgestellte Methode ein Höchstmaß an Aktualisierbarkeit, Genauigkeit und Flexibilität. Die im Zuge dieser Arbeit implementierte Software bietet zudem vielseitige Möglichkeiten zur Visualisierung und Interaktion mit dem Verfahren.

8.2 Ausblick

Durch die von bestehenden Verfahren abweichende Struktur dieses Kalibrierungsverfahrens ergeben sich neue, unbeantwortete Fragestellungen, die sich als Grundlage für weiterführende Forschung anbieten:

- Ist es möglich, durch Gewichtung der einzelnen Bildfehler die Stabilität des Verfahrens zu erhöhen? Möglich wäre eine Gewichtung auf Basis der Verteilung der Objektpositionen im Bild oder der Kameraposen im Raum. Zu analysieren wäre, welche Bild-Posekombinationen sich wie auf die Stabilität der Methode auswirken.
- Ist die Verwendung eines farbigen Kalibrierungsobjekts sinnvoll? Welches Bayer-Filter-Verfahren bietet die höchste Genauigkeit bei der Lokalisierung des Kalibrierungsmusters? Gibt es eine Möglichkeit, Unzulänglichkeiten bei Systemen mit Hardware-Bayer-Filter zu umgehen?
- Wie sind die Posen der Kamera im Raum zu verteilen, um eine höchste Stabilität und maximale Invarianz gegenüber verrauschten Daten im erzeugten Bewertungsmodell zu erhalten? Die Aufstellung eines analytischen Modells zur Bewertung der Auswirkungen von Varianzen in Abhängigkeit von der Verteilung der Kamerapositionen im sechsdimensionalen Posenraum kann zur Erhöhung der Genauigkeit führen.
- Die Testergebnisse des Simulators zeigen einen Fehler bis zu 0.064 mm in der translativen Komponente der extrinsischen Parameter, der allein durch die Diskretisierung auf die Pixel in der Bildebene entsteht. Die Berechnung der Fitness basiert auf der Erstellung eines RMS-Bildfehlerwertes, der die Annahme eines normalverteilten Rauschens in den Bildfehlern impliziert. Ist diese Annahme gerechtfertigt oder gibt es ein Fehlermaß, das der Charakteristik des Rauschens eher Rechnung trägt als der RMS-Fehler?

Die Architektur des hier vorgestellten Verfahrens erlaubt es zudem, die Fehlermodelle bestehender Markerpunkt-Kalibrierungsverfahren zu integrieren, ohne dass dies auf Kosten der Flexibilität des Gesamtverfahrens geht. Die Implementierung dieses Verfahrens stellt somit einen Rahmen zur Verfügung, in dem mit geringem Aufwand die Kernelemente und die Fehlerbewertung existierender Verfahren direkt und repräsentativ miteinander verglichen werden können.

Anhang A

Herleitungen

A.1 Herleitung der initialen Translationsschätzung

In Kapitel 2.3 wird die Rekonstruktion der initialen Translationen im Kamerasystem t_1, t_2 und t_3 aus den Beobachtungen einer Parallelprojektion P^* gefordert, mit

$$P^*_\lambda := \mathbb{R}^4 \longrightarrow \mathbb{R}^2, \begin{pmatrix} x \\ y \\ z \\ s \end{pmatrix} \longmapsto \begin{pmatrix} x \cdot \lambda_x \\ y \cdot \lambda_y \end{pmatrix}, \lambda = \begin{pmatrix} \lambda_x \\ \lambda_y \end{pmatrix}, \lambda_x, \lambda_y \in \mathbb{R}. \tag{A.1}$$

Für die Berechnung von t_1, t_2 und t_3 werden zunächst die Skalierung λ_x und λ_y der Parallelprojektion bestimmt. Auf dieser Grundlage erfolgt die Berechnung von $(t_1)_z$, $(t_2)_z$ und $(t_3)_z$. Diese Berechnung sei gleichzeitig konstruktiver Beweis für die Eindeutigkeit des Ergebnisses. Zunächst sei die bereits in (2.21) betrachtete Matrix T definiert:

$$T = \begin{pmatrix} \frac{-1}{\delta_t} & 0 & 0 & 0 \\ 0 & \frac{-1}{\delta_t} & 0 & 0 \\ 0 & 0 & \frac{-1}{\delta_t} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} & t_1^t & & \\ & t_2^t & & \\ & t_3^t & & \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} & \frac{-1}{\delta_t} \cdot t_1^T & & \\ & \frac{-1}{\delta_t} \cdot t_2^T & & \\ & \frac{-1}{\delta_t} \cdot t_3^T & & \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{A.2}$$

Aus (2.21) folgt, dass T eine orthonormale Matrix ist.

Kombiniert man nun (A.2) und (2.24), so ergibt sich:

$$T = \begin{pmatrix} \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_x}{\lambda_x} & \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_y}{\lambda_y} & \frac{-1}{\delta_t} \cdot (t_1)_z & 0 \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_x}{\lambda_x} & \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_y}{\lambda_y} & \frac{-1}{\delta_t} \cdot (t_2)_z & 0 \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_x}{\lambda} & \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_y}{\lambda_y} & \frac{-1}{\delta_t} \cdot (t_3)_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{A.3}$$

Da T eine orthonormale Matrix ist, sind die Zeilenvektoren orthogonal zueinander und die euklidische Norm jedes Zeilenvektors ist 1. Sei zunächst die Schreibweise der zwei linken Zeilenvektoren durch Herausziehen der Skalare vereinfacht. Es gilt:

$$\begin{pmatrix} \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_x}{\lambda_x} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_x}{\lambda_x} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_x}{\lambda_x} \\ 0 \end{pmatrix} = \frac{-1}{\delta_t \cdot \lambda_x} \cdot \begin{pmatrix} P^*(t_1)_x \\ P^*(t_2)_x \\ P^*(t_3)_x \\ 0 \end{pmatrix} \tag{A.4}$$

und

$$\begin{pmatrix} \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_y}{\lambda_y} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_y}{\lambda_y} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_y}{\lambda_y} \\ 0 \end{pmatrix} = \frac{-1}{\delta_t \cdot \lambda_y} \cdot \begin{pmatrix} P^*(t_1)_y \\ P^*(t_2)_y \\ P^*(t_3)_y \\ 0 \end{pmatrix}. \tag{A.5}$$

Da

$$\left\| \frac{-1}{\delta_t \cdot \lambda_x} \cdot \begin{pmatrix} P^*(t_1)_x \\ P^*(t_2)_x \\ P^*(t_3)_x \\ 0 \end{pmatrix} \right\|_2 = \left\| \frac{-1}{\delta_t \cdot \lambda_y} \cdot \begin{pmatrix} P^*(t_1)_y \\ P^*(t_2)_y \\ P^*(t_3)_y \\ 0 \end{pmatrix} \right\|_2 = 1 \tag{A.6}$$

gilt und da δ_t bekannt ist, lassen sich λ_x und λ_y direkt angeben durch:

$$\lambda_x = \frac{-1}{\delta_t} \cdot \left\| \begin{pmatrix} P^*(t_1)_x \\ P^*(t_2)_x \\ P^*(t_3)_x \\ 0 \end{pmatrix} \right\|_2 \tag{A.7}$$

und

$$\lambda_y = \frac{-1}{\delta_t} \cdot \left\| \begin{pmatrix} P^*(t_1)_y \\ P^*(t_2)_y \\ P^*(t_3)_y \\ 0 \end{pmatrix} \right\|_2. \tag{A.8}$$

Da auf diese Weise λ_x und λ_y berechenbar sind, sind von der Matrix T (A.3) die 1. ,2. und 4. Spalte ebenfalls bekannt. Der 3. Spaltenvektor, in dem sich die z-Komponenten von t_1, t_2 und t_3 befinden, lässt sich nun rekonstruieren, da er orthogonal zu den übrigen Spaltenvektoren steht und eine euklidische Norm von 1 besitzt. Sei die linke obere 3×3 Submatrix von T betrachtet. Es gilt:

$$\begin{pmatrix} \frac{P^*(t_1)_x}{-\delta_t \cdot \lambda_x} & \frac{P^*(t_1)_y}{-\delta_t \cdot \lambda_y} & \frac{-1}{\delta_t} \cdot (t_1)_z \\ \frac{P^*(t_2)_x}{-\delta_t \cdot \lambda_x} & \frac{P^*(t_2)_y}{-\delta_t \cdot \lambda_y} & \frac{-1}{\delta_t} \cdot (t_2)_z \\ \frac{P^*(t_3)_x}{-\delta_t \cdot \lambda_x} & \frac{P^*(t_3)_y}{-\delta_t \cdot \lambda_y} & \frac{-1}{\delta_t} \cdot (t_3)_z \end{pmatrix} \tag{A.9}$$

ist eine orthonormale Matrix. Daraus folgt, dass der Spaltenvektor durch ein einfaches Kreuzprodukt bestimmt werden kann:

$$\begin{pmatrix} \frac{-1}{\delta_t} \cdot (t_1)_z \\ \frac{-1}{\delta_t} \cdot (t_2)_z \\ \frac{-1}{\delta_t} \cdot (t_3)_z \end{pmatrix} = \begin{pmatrix} \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_x}{\delta_t \cdot \lambda_x} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_x}{\delta_t \cdot \lambda_x} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_x}{\delta_t \cdot \lambda} \end{pmatrix} \times \begin{pmatrix} \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_y}{\delta_t \cdot \lambda_y} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_y}{\delta_t \cdot \lambda_y} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_y}{\delta_t \cdot \lambda_y} \end{pmatrix} \tag{A.10}$$

Der ebenfalls orthogonal auf der Ebene der x- und y-Komponenten stehende, normierte Vektor

$$(-1) \cdot \begin{pmatrix} \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_x}{\delta_t \cdot \lambda_x} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_x}{\delta_t \cdot \lambda_x} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_x}{\delta_t \cdot \lambda_x} \end{pmatrix} \times \begin{pmatrix} \frac{-1}{\delta_t} \cdot \frac{P^*(t_1)_y}{\delta_t \cdot \lambda_y} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_2)_y}{\delta_t \cdot \lambda_y} \\ \frac{-1}{\delta_t} \cdot \frac{P^*(t_3)_y}{\delta_t \cdot \lambda_y} \end{pmatrix} \tag{A.11}$$

kommt als Lösung nicht in Frage, da sonst gelten würde:

$$det(T) = -1 \tag{A.12}$$

Dies würde jedoch im Widerspruch zu einer Implikation der Formel (2.21) stehen, die besagt, dass

$$GC_r = (G_{0,r})^{-1} \cdot T. \tag{A.13}$$

Es gilt, dass $\det(GC_r) = 1$ und $\det(G_{0,r}) = 1$, da GC_r und $G_{0,r}$ reine Rotationsmatrizen sind. Daraus folgt, dass

$$\det(T) = \det(G_{0,r}) \cdot \det(GC_r) = 1. \tag{A.14}$$

A.2 Verwendung der Givens-Rotationen

In Kapitel 3.1 ist zu jedem homogenen Vektor $t \in \mathbb{R}^4$ eine Matrix R gesucht, für die gilt:

$$t = R \cdot \begin{pmatrix} 0 \\ 0 \\ \|t\|_h \\ 1 \end{pmatrix} \tag{A.15}$$

Dieses Problem ist äquivalent zu dem einer trivialen QR-Zerlegung. Die zur QR-Zerlegung verwendeten Givensrotationen [4] liefern auch für (A.15) eine Lösung. Sei $t \in \mathbb{R}^4$ ein homogener Vektor und $t = (x, y, z, 1)^T$. Nach (2.8) gilt:

$$\|t\|_h = \sqrt{x^2 + y^2 + z^2} \tag{A.16}$$

Wie bei einer QR-Zerlegung gilt nun:

$$t = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{z}{\sqrt{x^2+z^2}} & 0 & \frac{x}{\sqrt{x^2+z^2}} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{x}{\sqrt{x^2+z^2}} & 0 & \frac{z}{\sqrt{x^2+z^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ y \\ \sqrt{x^2+z^2} \\ 1 \end{pmatrix}. \tag{A.17}$$

Weiter gilt:

$$t = \begin{pmatrix} \frac{z}{\sqrt{x^2+z^2}} & 0 & \frac{x}{\sqrt{x^2+z^2}} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{x}{\sqrt{x^2+z^2}} & 0 & \frac{z}{\sqrt{x^2+z^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{x^2+z^2}}{\sqrt{y^2+(x^2+z^2)}} & \frac{y}{\sqrt{y^2+(x^2+z^2)}} & 0 \\ 0 & -\frac{y}{\sqrt{y^2+(x^2+z^2)}} & \frac{\sqrt{x^2+z^2}}{\sqrt{y^2+(x^2+z^2)}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ \sqrt{y^2+x^2+z^2} \\ 1 \end{pmatrix}. \tag{A.18}$$

Nach Satz über Givensrotation [4] gilt:

$$R := \begin{pmatrix} \frac{z}{\sqrt{x^2+z^2}} & 0 & \frac{x}{\sqrt{x^2+z^2}} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{x}{\sqrt{x^2+z^2}} & 0 & \frac{z}{\sqrt{x^2+z^2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{x^2+z^2}}{\sqrt{y^2+(x^2+z^2)}} & \frac{y}{\sqrt{y^2+(x^2+z^2)}} & 0 \\ 0 & -\frac{y}{\sqrt{y^2+(x^2+z^2)}} & \frac{\sqrt{x^2+z^2}}{\sqrt{y^2+(x^2+z^2)}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.19}$$

ist eine Rotationsmatrix und es gilt

$$t = R \cdot \begin{pmatrix} 0 \\ 0 \\ \|t\|_h \\ 1 \end{pmatrix}. \tag{A.20}$$

A.3 Autonome Zentrierung des Kalibrierungsobjekts

Im Folgenden sei der zur Fokussierung verwendete Algorithmus in Pseudocode angegeben. Dabei sei die Funktion $GetOriResponse(yaw, pitch, roll)$ die Funktion, die eine Pose anfährt, und die euklidische Differenz auf der Einheits-Bildebene $[-1, 1]^2$ zwischen Objektposition und Mittelpunkt zurückgibt. Sollte das Objekt nicht im Bild zu sehen seien, gibt $GetOriResponse(\cdot, \cdot\cdot, \cdot\cdot\cdot)$ den Wert 2 zurück.

Die Variablen `yaw`, `pitch` und `roll` enthalten zu Beginn des Algorithmus die aktuelle Orientierung des Greifers und am Ende die gewünschte Orientierung, in der die Kamera das Objekt fokussiert. Die Genauigkeit der Fokussierung kann durch die Variable `precision` als maximale Abweichung in der Bildebene angegeben werden. Die Variable `stepSize` enthält die initiale Schrittweite als Winkeldifferenz. Diese sollte zu Anfang mindestens die Hälfte des Öffnungswinkels der Kamera betragen. Eine zu klein gewählte Schrittweite kann dazu führen, dass der Algorithmus nicht zur gewünschten Winkelstellung vordringen kann. Ein zu groß gewählter Winkel sorgt lediglich für eine längere Approximationsdauer.

```
currentError = GetOriResponse(yaw,pitch,roll)

While (currentError < precision) do

   if (GetOriResponse(yaw + stepSize,pitch,roll) < currentError) then
      yaw := yaw + stepSize
   else if (GetOriResponse(yaw - stepSize,pitch,roll) < currentError) then
      yaw := yaw - stepSize
   end if

   currentError = GetOriResponse(yaw,pitch,roll)
```

```
   if (GetOriResponse(yaw,pitch + stepSize,roll) < currentError) then
      pitch := pitch + stepSize
   else if (GetOriResponse(yaw,pitch - stepSize,roll) < currentError) then
      pitch := pitch - stepSize
   end if

   currentError = GetOriResponse(yaw,pitch,roll)

   if (GetOriResponse(yaw,pitch,roll + stepSize) < currentError) then
      roll := roll + stepSize
   else if (GetOriResponse(yaw,pitch,roll - stepSize) < currentError) then
      roll := roll - stepSize
   end if

   currentError = GetOriResponse(yaw,pitch,roll)

   stepSize := stepSize * 0.5

loop
```

Anhang B

Fehlerabschätzungen

B.1 Das initiale Kameramodell

In diesem Abschnitt wird der Fehler zwischen der zur Initialschätzung verwendeten Parallelprojektion zu beliebigen anderen Kameramodellen analysiert. Zunächst wird die Äquivalenz der Modelle für Bewegungen entlang der zur Bildebene parallelen Achsen gezeigt. Anschließend wird die Differenz der Modelle bei beliebigen Bewegungen analysiert.
Da diese Abschätzung nur für die Berechnung der Orientierung des Initial-Modells benötigt wird, werden im Folgenden nur Fehler bezüglich reiner Translationen betrachtet.

B.1.1 Äquivalenz der Projektionsmodelle bei Bewegungen parallel zur Bildebene

Sei nun also der Spezialfall betrachtet, dass eine Translation parallel zur Bildebene erfolgt. Die folgenden Ortsvektoren seien im homogenen Kamerakoordinatensystem angegeben: Sei p die Position des Kalibrierungsobjekts in der Kamerastellung $C_0 = G_0 \cdot GC$, also:

$$p = C_0^{-1} \cdot c = (G_0 \cdot GC)^{-1} \cdot c \tag{B.1}$$

Sei des Weiteren für den hier betrachteten Spezialfall G_i, $i \in \{1, 2, 3\}$ eine Greiferpose, für die gilt, dass ihr der Differenzvektor zwischen ihrem Ursprung und dem Ursprung der Ausgangspose G_0 einen Vektor parallel zur Bildebene ist. Sei diese Differenz durch den Vektor a im Kamerakoordinatensystem definiert durch:

$$a = C_i^{-1} \cdot c = (G_i \cdot GC)^{-1} \cdot c \tag{B.2}$$

Sei nun die Parallelprojektion P_s sowie die Lochkameraprojektion L_r definiert als:

$$P_\lambda : \mathbb{R}^4 \longrightarrow \mathbb{R}^2, \begin{pmatrix} x \\ y \\ z \\ s \end{pmatrix} \longmapsto \begin{pmatrix} \frac{x}{\lambda_x} \\ \frac{y}{\lambda_y} \end{pmatrix} \tag{B.3}$$

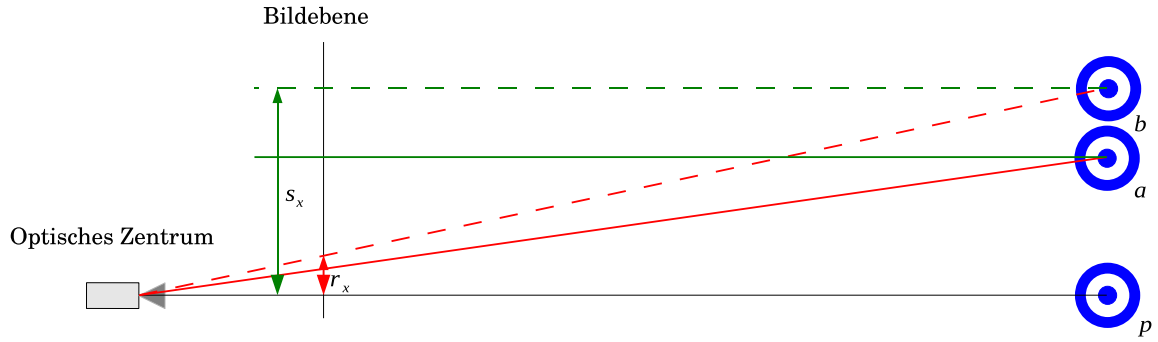


Abbildung B.1: Punkt a wird auf die Bildebene projiziert. Punkt b dient dabei als Referenz für den entsprechenden Randpunkt des Projektionsbildes. Parallelprojektion (grün) und Lochkameramodell (rot) gleichen sich dabei bis auf einen Skalierungsfaktor.

$$L_r : \mathbb{R}^4 \longrightarrow \mathbb{R}^2, \begin{pmatrix} x \\ y \\ z \\ s \end{pmatrix} \longmapsto \begin{pmatrix} \frac{x}{z} \cdot r_x \\ \frac{y}{z} \cdot r_y \end{pmatrix}, \tag{B.4}$$

Zur Unifizierung der Bildebenen von P_s und L_r, also zur Anpassung der Bildskalierungsvektoren $\lambda = (\lambda_x, \lambda_y)^T$ und $r = (r_x, r_y)^T$, sei Referenzpunkt $b \in \mathbb{R}^4$ im Kamerakoordinatensystem als „auf dem Einheitskreis zu projizierende“ Verlängerung der Verbindung von p und a definiert, so dass es einen Faktor $\sigma \in (R)$ gibt mit $a = p + \sigma \cdot (a - p)$, siehe auch Abb. B.1. Somit gelte:

$$\|P_\lambda(b - p)\|_2 = 1 \tag{B.5}$$

und

$$\|L_r(b - p)\|_2 = 1. \tag{B.6}$$

Somit gilt für die Projektionen:

$$P_\lambda(a) = \begin{pmatrix} a_x \cdot \lambda_x \\ a_y \cdot \lambda_y \end{pmatrix} = \begin{pmatrix} (p_x + \sigma \cdot (b_x - p_x)) \cdot \lambda_x \\ (p_y + \sigma \cdot (b_y - p_y)) \cdot \lambda_y \end{pmatrix}. \tag{B.7}$$

Da $p_x = 0$ und $p_y = 0$, da p auf $(0, 0)$ projiziert wird, gilt:

$$P_\lambda(a) = \sigma \cdot \begin{pmatrix} b_x \cdot \lambda_x \\ b_y \cdot \lambda_y \end{pmatrix} \tag{B.8}$$

Mit $\lambda = \frac{r}{p_z}$ gilt somit:

$$P_s(a) = \sigma \cdot \begin{pmatrix} \frac{b_x}{b_z} \cdot r_x \\ \frac{b_y}{b_z} \cdot r_y \end{pmatrix} = \begin{pmatrix} \frac{a_x}{a_z} \cdot r_x \\ \frac{a_y}{a_z} \cdot r_y \end{pmatrix} = L_r(a) \tag{B.9}$$

Demnach ist für den betrachteten Spezialfall die Äquivalenz zwischen Lochkameramodell und Parallelprojektion gegeben, jedoch gilt dies nicht für den allgemeinen Fall:

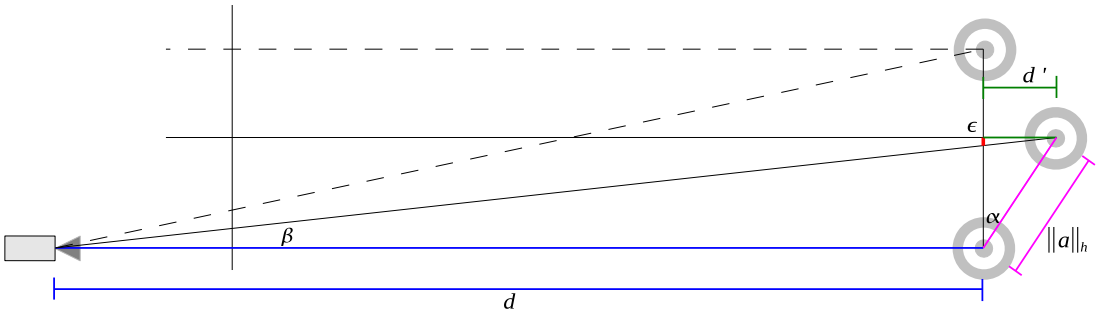


Abbildung B.2: Zur Analyse der Differenz zwischen Parallelprojektion und Lochkameramodell genügt es, die durch α verursachte Differenz der Modelle auf einer zur Bildebene parallelen Ebene betrachten, da hier beide Modelle äquivalent sind. Der Modellfehler kann so einem Bewegungsfehler ϵ (rot) gleichgesetzt werden.

B.1.2 Differenz der Projektionsmodelle bei Bewegungen nicht parallel zur Bildebene

Sei im Folgenden der Fall betrachtet, in dem die Transformation $G_{0,i}$ keine zur Bildebene parallele Translation ausführt. Dann gibt es einen Winkel $\alpha \neq 0$, in dem die Gerade durch den Punkt p_0 entlang a zur Bildebende steht. Da durch die Neigung die z-Komponente der Bewegungsvektoren im Kamerakoordinatensystem nicht mehr verschwindet, ist die beobachtete Äquivalenz zwischen Lochkameramodell und Parallelprojektion für diesen Fall nicht mehr gültig. Es gilt nun, die relative Abweichung des Lochkameramodells von der Parallelprojektion zu analysieren.

Seien dafür wie in Abb. B.2 die Differenz $\epsilon \in \mathbb{R}$ zwischen der Parallelprojektion und dem Lochkameramodell eingeführt. Die Differenz ist dabei auf der zur Bildebene parallelen Unifizierungsebene analysiert, um ein anschauliches Fehlermaß zu erhalten. Die Applikation des jeweils anderen Modells entspricht somit einer mit dem Fehler ϵ behafteten Bewegung.

Es gilt

$$d' = \sin(\alpha) \cdot \|a\|_h. \tag{B.10}$$

Für ϵ folgt somit:

$$\epsilon = \tan(\beta) \cdot d'. \tag{B.11}$$

$\tan(\beta)$ lässt sich nun abschätzen durch

$$\tan(\beta) \leq \frac{\cos(\alpha) \cdot \|a\|_h}{d} \tag{B.12}$$

und somit folgt:

$$\epsilon \leq \frac{\cos(\alpha) \cdot \|a\|_h \cdot \sin(\alpha) \cdot \|a\|_h}{d}. \tag{B.13}$$

Da

$$\max_{\alpha \in [0, \frac{\pi}{2}]} (\cos(\alpha) \cdot \sin(\alpha)) = \frac{1}{2} \tag{B.14}$$

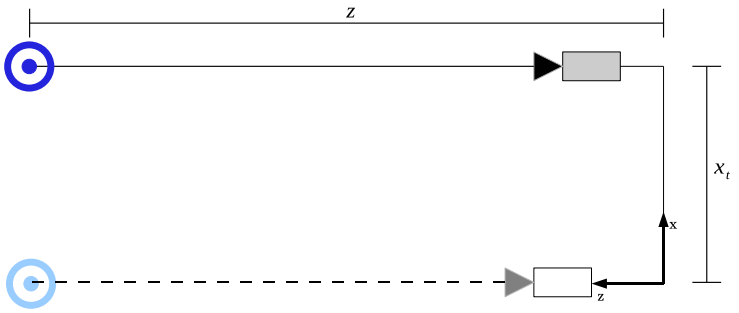
gilt für ϵ:

$$\epsilon \leq \frac{\|a\|_h^2}{2 \cdot d}. \tag{B.15}$$

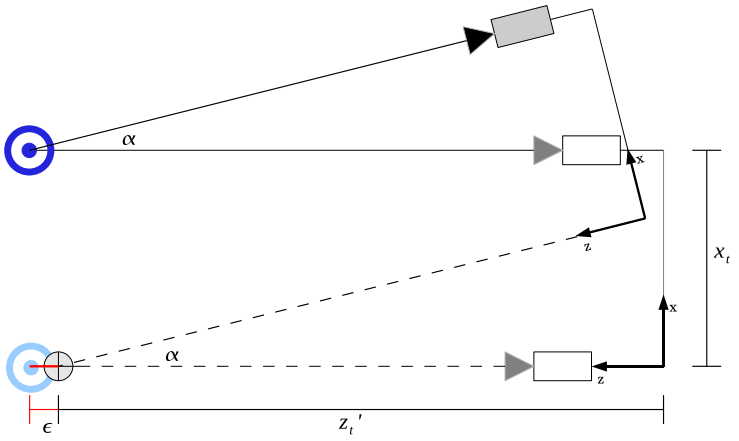
Zur Veranschaulichung:
Die Länge der initialen Translationen a beträgt in der Implementierung des Algorithmus 3 cm. Bei einer initialen Distanz zwischen Kamera und Kalibrierungsobjekt d von 80 cm tritt demnach im Berechnungsmodell eine maximale Differenz von $\epsilon = 0{,}5$ mm zwischen der Interpretation durch ein Lochkameramodell und einer Parallelprojektion auf. Eine Differenz dieser Größenordnung kann vernachlässigt werden. Beispielsweise liegt in einer solchen Umgebung die Diskretisierung einer Kamera mit einem Öffnungswinkel von 40° auf 1024 Pixel bei 0.656 mm.

B.2 Die initiale Distanzschätzung

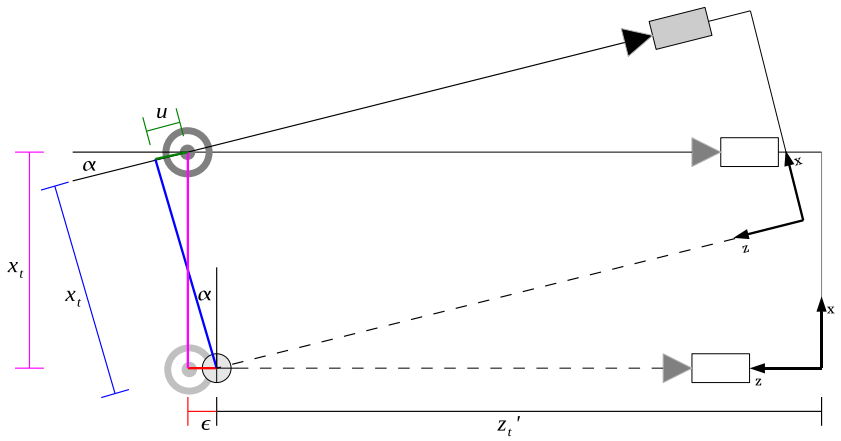
Die in Kapitel 2.4 beschriebene Schätzung der Distanz zwischen Greifer und Kalibrierungsmuster geht von der Annahme aus, dass sich die Position des optischen Zentrums der Kamera auf einer Linie zwischen dem Ursprung des Greiferkoordinatensystems und dem Kalibrierungsobjekt befindet. Durch Rotationen um Punkte auf dieser Sichtlinie wird die Entfernung approximiert. Innerhalb dieses linearen Modells (Die Sichtachse geht durch den Ursprung des Greiferkoordinatensystems) wird die Distanz dadurch approximiert, dass eine Rotation des Greifers um den gesuchten Punkt keine Positionionsänderung des Punktes im Kamerakoordinatensystem hervorruft. Ist jedoch die Sichtlinie um einen Wert $x_t \in \mathbb{R}$ verschoben (affines Modell), so wird eine falsche Distanz berechnet, deren Fehler im Folgenden analysiert wird.
Da die Transformationen in 2.4 innerhalb eines zweidimensionalen Unterraumes stattfinden, lässt sich die Abschätzung des Fehlers ebenfalls auf diese Ebene beschränken. Sei also $z_t \in \mathbb{R}_{\geq 0}$ die Distanz zwischen Greifer und Kalibrierungsmuster.



Im linearen Modell befindet sich die Kamera (helles Kameraobjekt) auf der Sichtlinie. Tatsächlich ist die Kamera aber auf einer um x_t verschobenen Gerade (affines Modell, dunkles Kameraobjekt). Eine Rotation um einen Punkt auf der Sichtlinie, für die der Blickwinkel im affinen Modell erhalten bleibt, führt zu einem Fehler $\epsilon \in \mathbb{R}$ in der Distanzschätzung:



Der Distanzfehler ϵ lässt sich nun mit Hilfe einer geometrischen Konstruktion berechnen:



Aus dieser Konstruktion lässt sich nun eine Berechnungsvorschrift für ϵ (rot) ablesen, indem man die horizontalen Komponenten über Magenta, Grün und Blau gemäß ihrer Ausrichtung addiert/subtrahiert:

$$\epsilon = 0 + \cos(\alpha) \cdot u - \sin(\alpha) \cdot x_t. \tag{B.16}$$

Wobei für die unbekannte Größe $u \in \mathbb{R}$ unter der Betrachtung der vertikalen Komponenten gilt (über blau und grün):

$$x_t = \cos(\alpha) \cdot x_t + \sin(\alpha) \cdot u \tag{B.17}$$

und somit

$$u = \frac{1 - \cos(\alpha)}{\sin(\alpha)} \cdot x_t. \tag{B.18}$$

Durch Einsetzen erhält man:

$$\epsilon = \frac{1 - \cos(\alpha)}{\sin(\alpha)} \cdot \cos(\alpha) \cdot x_t - \sin(\alpha) \cdot x_t. \tag{B.19}$$

Die Variable α kann von dem Algorithmus frei gewählt werden. In der folgenden Tabelle ist zur Anschauung der relativen Fehler ϵ für verschiedene Winkel α aufgeführt:

α	10	12	14	16	18	20
ϵ	$-0.08749 \cdot x_t$	$-0.1051 \cdot x_t$	$-0.1228 \cdot x_t$	$-0.1405 \cdot x_t$	$-0.1584 \cdot x_t$	$-0.1763 \cdot x_t$

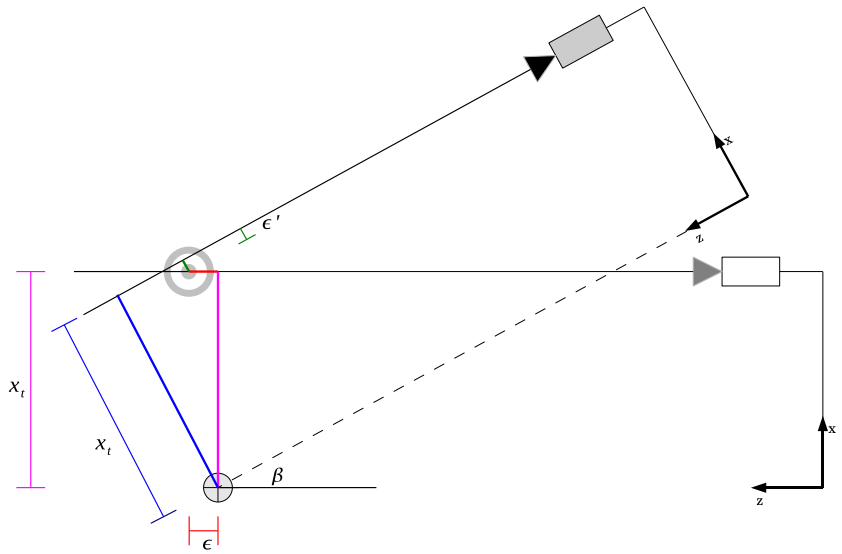
Der effektive, im Bild auftretende Fehlerwinkel, der bei einer Fokussierung des Objektes nach dem linearen Modell (wie in Kapitel 3.1 verwendet) auftritt, liegt jedoch weit unter diesen relativen Fehlerwerten. Die Abschätzung dieses Fehlers ist in Kapitel B.3 zu finden.

B.3 Der Winkelfehler des linearen Modells

Die in Kapitel 3.1 beschriebene Generierung der Greiferposen basiert auf einem vereinfachten Umgebungsmodell zur Fokussierung des Kalibrierungsobjekts. Dieses Modell geht davon aus, dass die optische Achse der Kamera durch den Ursprung des Greiferkoordinatensystems geht. Diese Annahme führt bereits bei der Distanzschätzung zu einem Fehler, dessen Berechnung in Kapitel B.2 zu finden ist. Die Fortpflanzung dieses Fehlers bei der Generierung der Posen wird in diesem Kapitel abgeschätzt.
Hierfür sei eine Annahme getroffen, die in der Praxis stets erfüllt sein sollte:

Die Entfernung des Greifers zu dem Kalibrierungsmuster sei nie kleiner als die doppelte translative Differenz zwischen Greifer und Kamera.

Dies sollte durch die inverse Kinematik gewährleistet werden, da andernfalls eine Kollision der Kamera mit dem Objekt, welches das Kalibrierungsmuster trägt, nicht ausgeschlossen werden kann. Diese Annahme wird zu einem späteren Zeitpunkt in die Abschätzung des Fehlers eingehen.
Zur Fokussierung des Objektes ist die geschätzte Distanz d zwischen dem Ursprung von G_0 und dem Kalibrierungsmuster gegeben. Diese Schätzung ist mit dem Fehler ϵ behaftet. Des Weiteren hängt der Fehler einer generierten Pose von der translativen Differenz x_t (siehe B.2) und der Winkeldifferenz β ab. β resultiert aus der Fokussierung des Objektes anhand des linearen Modells:



Sei zunächst die Abweichung $|\epsilon'|$ (grün) an der Stelle des Kalibrierungsmusters abgeschätzt. Die Entfernung k (türkis) zwischen Rotationsachse und Kalibrierungsmuster ist gegeben durch

$$k = \sqrt{x_t^2 + \epsilon^2}. \tag{B.20}$$

Betrachtet man die Position des Kalibrierungsmusters im rotierten Kamerasystem, so wird ϵ' abgeschätzt werden durch

$$\epsilon' = x_t - \cos(\beta) \cdot x_t - \sin(\beta) \cdot \epsilon \tag{B.21}$$

Da das Kalibrierungsmuster nicht von hinten betrachtet werden kann, sei β auf den Bereich

$$\beta \in [-\pi/2, +\pi/2] \tag{B.22}$$

beschränkt.

Für die folgende Abschätzung von $|\epsilon'|$ reicht es, den Fall $sign(\sin(\beta)) \neq sign(x_t))$ zu betrachten, da für alle $\beta \in [-\pi/4, +\pi/4]$ und $x_t \in \mathbb{R}_{\neq 0}$ gilt:

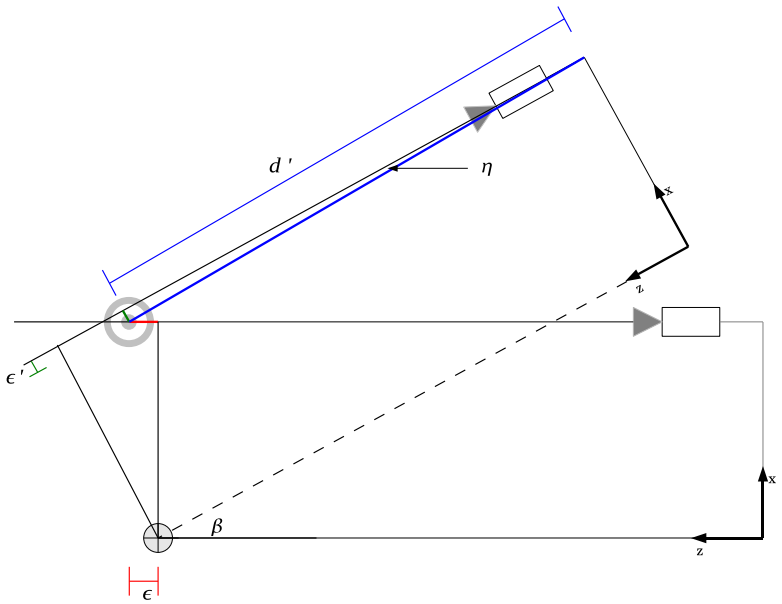
$$sign(x_t - \cos(\beta) \cdot x_t) = sign(x_t), \tag{B.23}$$

da $\cos(\beta) > 0$. Für ϵ'_{max}, das ϵ' mit dem höchstmöglichen Absolutwert, gilt somit

$$|\epsilon'_{max}| \leq |x_t - \cos(\beta) \cdot x_t| + |\sin(\beta) \cdot \epsilon|. \tag{B.24}$$

Daraus folgt

$$|\epsilon'_{max}| \leq |x_t \cdot (1 - \cos(\beta))| + |\sin(\beta) \cdot \epsilon|. \tag{B.25}$$



Für die Abschätzung des Fehlerwinkels η erhält man mittels der zuvor getroffenen Annahme über die Beziehung zwischen d' und x_t:

$$|\sin(\eta)| = \frac{|\epsilon'|}{d'} \leq \frac{|\epsilon'|}{2 \cdot x_t}. \tag{B.26}$$

Somit gilt

$$|\sin(\eta)| \leq |\frac{\epsilon'_{max}}{2x_t}| = |\frac{|x_t \cdot (1 - \cos(\beta))| + |\sin(\beta) \cdot \epsilon|}{2x_t}|. \tag{B.27}$$

Mit B.19 folgt dann

$$|\sin(\eta)| \leq |\frac{|x_t \cdot (1 - \cos(\beta))| + |\sin(\beta) \cdot \frac{1-\cos(\alpha)}{\sin(\alpha)} \cdot \cos(\alpha) \cdot x_t - \sin(\alpha) \cdot x_t|}{2x_t}|, \tag{B.28}$$

wobei α der vom System festgelegte Winkel der initialen Rotation ist. Aufgrund der Monotonie der Sinusfunktion auf dem Intervall $[-\pi, +\pi]$ gilt somit:

$$|\eta| \leq \left| \sin^{-1} \left(\frac{|(1 - \cos(\beta))| + |\sin(\beta) \cdot \frac{1-\cos(\alpha)}{\sin(\alpha)} \cdot \cos(\alpha) - \sin(\alpha)|}{2} \right) \right|. \tag{B.29}$$

Somit lässt sich zu jedem maximalen Fokussierungswinkel β ein theoretisch benötigter Öffnungswinkel η angeben, der garantiert, dass das Kalibrierungsmuster in jeder Pose im Bild zu sehen ist. Zur Veranschaulichung von (B.29) ist in der folgende Tabelle für initiale Rotationen α und maximale Neigungen während der Kalibrierung β die maximale Abweichung η berechnet. Eine Kamera mit einem Öffnungswinkel $\eta \cdot 2$ hat in jeder der generierten Posen das Kalibrierungsobjekt im Bild.

β	$\alpha = 10$	$\alpha = 12$	$\alpha = 14$	$\alpha = 16$	$\alpha = 18$	$\alpha = 20$
20	5.86838	6.69179	7.51241	8.32972	9.1432	9.95232
25	6.63038	7.41631	8.20019	8.98162	9.76018	10.5354
30	7.60087	8.35125	9.10034	9.84782	10.5934	11.3367
35	8.77407	9.49118	10.2077	10.9235	11.6383	12.3519
40	10.1433	10.8298	11.5164	12.2031	12.8896	13.576
45	11.7013	12.3601	13.0197	13.6801	14.3414	15.0033
50	13.4401	14.0746	14.7105	15.3479	15.987	16.6277
55	15.3519	15.9656	16.5814	17.1994	17.8198	18.4428
60	17.4285	18.0254	18.625	19.2275	19.8332	20.4422

Zu diesen Werten sei angemerkt, dass die in der Tabelle aufgeführte maximale Abweichung (Fehlerwinkel) lediglich in einer Pose der generierten Winkel auftreten kann. In allen anderen Posen wird dieser theoretische Fehlerwert deutlich unterschritten.

Anhang C

Tabellarische Auflistung der Symbole

C.1 Variablen

Symbol	Definiert als	Beschreibung
N	$\subset \mathbb{N}$	Indexmenge zur Indizierung der Bilder/Posen
GC	$\in \mathbb{R}^{4\times4}$	Homogene Transformation Greifer $\rightarrow$ Kamera
GC^*	$\in \mathbb{R}^{4\times4}$	Transformation Greifer $\rightarrow$ Kamera im linearen Modell
G_i	$\in \mathbb{R}^{4\times4}$	Koordinatensystem des Greifers in Position $i \in \mathbb{N}$
$G_{i,j}$	$\in \mathbb{R}^{4\times4}$	Matrix, definiert durch: $G_j = G_i \cdot G_{i,j}$ mit $i, j \in \mathbb{N}$
C_i	$\in \mathbb{R}^{4\times4}$	Koordinatensystem der Kamera in Position $i \in \mathbb{N}$
$C_{i,j}$	$\in \mathbb{R}^{4\times4}$	Matrix, definiert durch: $C_j = C_i \cdot C_{i,j}$ mit $i, j \in \mathbb{N}$
(u_i, v_i)	$\in \mathbb{R}^2$	Position des Kalibrierungsobjektes in Bild i
c	$\in \mathbb{R}^4$	Positionsvektor des Kalibrierungspunktes
c^*	$\in \mathbb{R}^4$	Schätzung des Positionsvektors des Kalibrierungspunktes
d	$\in \mathbb{R}$	Distanz zwischen Kamera und Kalibrierungsobjekt in C_0
α	$\in \mathbb{R}$	Winkel, um den G_0 von G_5 - G_7 abweicht
δ	$\in \mathbb{R}$	Entfernung, um die G_0 von G_1 - G_3 abweicht
t_1, t_2, t_3	$\in \mathbb{R}^4$	Die Translationspalten der Matrizen $G_{0,1}$, $G_{0,2}$ und $G_{0,3}$
T	$\in \mathbb{R}^{4\times4}$	Initiale Schätzung von $(C_0^{-1})_r$ zur Posengenerierung
R, D, Z	$\in \mathbb{R}^{4\times4}$	Teiltransformationen der Posengenerierungsfunktion K
Θ	$\in \mathbb{R}^{11} \times (\mathbb{R}^{\mathbb{R}})^2$	Parametervektor der Fitnessfunktion
$\beta_{x/y}$	$\in \mathbb{R}$	Öffnungswinkel in x- / y- Richtung der Kamera
$\lambda_{x/y}$	$\in \mathbb{R}$	Skalierung der Parallelprojektion P^*_λ
$o_{x/y}$	$\in \mathbb{R}$	Versatz des CCD Sensors zum Mittelpunkt der Bildebene
z_s	$\in \mathbb{R}$	Scherung der Bildebene in der Kamera
$z_{2/4/6}$	$\in \mathbb{R}$	Modellparameter der polynomiellen Linsenverzerrung
$g_{1/2/3/4}, k_1$	$\in \mathbb{R}$	Modellparameter der Linsenverzerrung nach Weng [23]

C.2 Operatoren und Funktionen

Symbol	Definiert als	Beschreibung
$\cdot_r$	$\mathbb{R}^{4\times4} \to \mathbb{R}^{4\times4}$	Operator zur Abbildung einer Matrix auf ihre Rotation
$\cdot_t$	$\mathbb{R}^{4\times4} \to \mathbb{R}^{4\times4}$	Operator zur Abbildung einer Matrix auf ihre Translation
$\cdot_{x/y/z/s}$	$\mathbb{R}^4 \to \mathbb{R}$	Abbildung eines homog. Vektors auf eine Komponente
$\cdot_{u/v}$	$\mathbb{R}^4 \to \mathbb{R}$	Abbildung eines Bildpunktes auf eine Komponente
$\|\cdot\|_h$	$\mathbb{R}^4 \to \mathbb{R}$	Norm homogener Vektoren
P	$\mathbb{R}^4 \to \mathbb{R}^2$	Abbildungsvorschrift der realen Kamera
P^*	$\mathbb{R}^4 \to \mathbb{R}^2$	Parallelprojektion als Kameramodell der Anfangsschätzung
δ_u,δ_v	$\mathbb{R}^2 \to \mathbb{R}$	Funktion der Linsenverzerrung für $u-$ und $v-$Komponente
K	$\mathbb{R}^4 \to \mathbb{R}^{4\times4}$	Funktion der Posengenerierung

Literaturverzeichnis

[1] Milton A. Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions.* U.S. Goverment Printing Office, 1964.

[2] Nicolas Andreff, Radu P. Horaud, and Bernard Espiau. On-line hand-eye calibration. In *Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling (3DIM'99), Ottawa, Canada*, pages 430–436, October 1999.

[3] Anne Auger and Nikolaus Hansen. Performance evaluation of an advanced local search evolutionary algorithm. *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2005.

[4] Gene H. Golub and Charles F. van Loan. *Matrix Computations, 2nd edn.* The Johns Hopkins University Press, 1989.

[5] Nikolaus Hansen and Stefan Kern. Evaluating the CMA evolution strategy on multimodal test functions. In *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 282–291, 2004.

[6] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaption in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[7] Richard Hartly and Andrew Zisserman. *Multiple View Geometry In Computer Vision.* Cambridge Press, Cambridge, United Kindom, 2003.

[8] Janne Heikkilä and Olli Silvén. Calibration procedure for short focal length off-the-shelf ccd cameras. In *Proceedings of the 13th International Conference on Pattern Recognition (CVPR'96), Wien, Austria*, pages 166–170, 1996.

[9] Radu P. Horaud and Fadi Dornaika. Hand-eye calibration. *International Journal on Robotics Research*, 14(3):195–210, June 1995.

[10] Vincent Lepetit and Pascal Fua. Monocular model-based 3d tracking of ridgid objects: A survey. *Fundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, volume 20, pages 91–110, 2003.

[12] Frank C. Park and Bryan J. Martin. Robot sensor calibration: solving $AX = XB$ on the Euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, October 1994.

[13] Yiu Cheung Shiu and Shaheen Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $AX = XB$. *IEEE Transactions on Robotics and Automation*, 5(1):16–29, 1989.

[14] Chester C. Slama. *Manual Of Photogrammetry*. American Society Of Photogrammetry, Falls Church, Virginia, 4th edition, 1980.

[15] Mark W. Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons, New York, Chichester, 2005.

[16] Klaus H. Strobl and Gerd Hirzinger. Optimal hand-eye calibration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4647–4653, 2006.

[17] Albert Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004.

[18] Roger Y. Tsai and Reimar K. Lenz. Real time versatile robotics hand/eye calibration using 3D machine vision. *IEEE Transactions on Robotics and Automation*, 1:554–561, 1988.

[19] Roger Y. Tsai and Reimar K. Lenz. A new technique fo fully autonomous end efficient 3D robotic hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(3):345–558, 1989.

[20] Dick van Albada, Jose Lagerberg, and Arnoud Visser. Eye in hand calibration. *Industrial Robot*, 21(6):14–17, 1994.

[21] Ching-Cheng Wang. Extrensic calibration of a vision sensor mounted on a robot. *IEEE Transactions on Robotics and Automation*, 8(2):161–175, 1992.

[22] Guo-Qing Wei, Klaus Arbter, and Gerd Hirzinger. Active self-calibration of robotic eyes and hand-eye relationships with model identification. *IEEE Transactions on Robotics and Automation*, 14(1):158–166, February 1998.

[23] Juyang Weng, Paul Cohen, and Marc Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, October 1992.

Kiel, den 22.8.2008

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

Andreas Stefan Jordt