JOHANNES KEPLER
UNIVERSITÄT LINZ

Eingereicht von
Lukas Probst, BSc

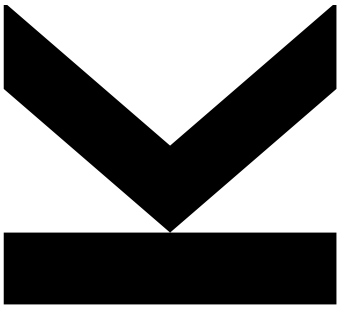
Angefertigt am
Institut für Robotik

Betreuer
Assoz. Univ.-Prof. DI Dr.
Hubert Gattringer

Mitbetreuung
DI Tobias Marauli

November 2024

Genauigkeitssteigerung von Industrierobotern mittels visuellen Methoden



Masterarbeit
zur Erlangung des akademischen Grades
Diplom-Ingenieur
im Masterstudium
Mechatronik

JOHANNES KEPLER
UNIVERSITÄT LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Abstract

In modern manufacturing applications, industrial robots must operate with increasing precision to meet rising demands. This thesis presents and compares two distinct approaches for enhancing the position and orientation accuracy of robots using vision-based methods.

The first approach, geometric calibration, focuses on optimizing the kinematic model using a dataset of end-effector poses and corresponding joint configurations. The second approach explores various control methods, utilizing real-time data of the end-effector pose for dynamic path correction. In both cases, a marker-based tracking method in the near-infrared range is employed to capture the end-effector pose.

The theoretical foundations of each method are first discussed, followed by testing within a simulation environment, and concluding with an evaluation on a physical robotic system.

Results indicate that both methods offer cost-effective solutions for improving both static and dynamic accuracy due to their relatively low hardware costs and high flexibility. However, further refinements are required to achieve optimal performance.

Finally, additional advantages of using camera systems for this application are discussed, and future optimization opportunities are presented.

Kurzfassung

Industrieroboter müssen in modernen Fertigungsanwendungen zunehmend präziser arbeiten, um den steigenden Anforderungen gerecht zu werden. In der folgenden Arbeit werden zwei verschiedene Konzepte zur Erhöhung der Positions- und Orientierungsgenauigkeit von Robotern basierend auf visuellen Methoden vorgestellt und einem Vergleich unterzogen.

Das erste Konzept, die geometrische Kalibrierung, umfasst die Optimierung des Kinematikmodells auf Basis eines Messdatensets aus Endeffektor-Posen und zugehörigen Achskonfigurationen. Das zweite Konzept widmet sich verschiedener Regelungsmethoden, bei der Echtzeit-Aufnahmen der Endeffektor-Pose für eine dynamische Korrektur der Roboterbahn genutzt werden. Die Erfassung der Endeffektor-Pose erfolgt in beiden Fällen über ein im nahen Infrarotbereich operierendes markerbasiertes Trackingverfahren.

Zuerst werden die theoretischen Grundlagen erläutert anschließend werden die beiden Konzepte in einer Simulationsumgebung getestet, gefolgt von einer Evaluierung an einem realen System.

Die Ergebnisse zeigen, dass sich beide Methoden durch die vergleichsweise niedrigen Hardwarekosten und ihre hohe Flexibilität als wirtschaftliche Lösung zur Verbesserung der statischen und dynamischen Genauigkeit bewähren, jedoch weitere Verbesserungen notwendig sind.

Zum Schluss werden zusätzliche Vorteile für den Einsatz von Kamerasystemen in diesem Aufgabenbereich diskutiert und ein Ausblick auf zukünftige Optimierungsmöglichkeiten gegeben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Vorwort

Diese Arbeit wurde in Zusammenarbeit mit der Firma Fill Gesellschaft m.b.H. am Institut für Robotik der Johannes Kepler Universität Linz im Rahmen des EU-Projekts SPRINTER (Projektnummer 101070581) erstellt.

Mein besonderer Dank gilt zuallererst meinem Teamleiter, Dipl.-Ing. Harald Sehrschön, sowie der Firma Fill Gesellschaft m.b.H., die mir ihr Vertrauen geschenkt haben, diese Arbeit im Rahmen meiner beruflichen Tätigkeit durchführen zu dürfen. Die Möglichkeit, sich während der Arbeitszeit einem Forschungsthema widmen zu können, ist keineswegs selbstverständlich und hat es mir erlaubt, mich intensiver auf mein Studium zu konzentrieren. Dafür bin ich äußerst dankbar.

Ein herzliches Dankeschön gilt meinem Betreuer, Assoz. Univ.-Prof. DI Dr. Hubert Gattringer, der mich während der gesamten Bearbeitung dieser Arbeit tatkräftig unterstützt und stets mit seiner fachlichen Expertise und Geduld zur Seite gestanden hat. Seine Begeisterung für die Robotik war für mich eine kontinuierliche Quelle der Inspiration und Motivation.

Auch meinen Kollegen am Institut für Robotik möchte ich herzlich danken. Sie waren immer bereit, mir mit Rat und Tat zur Seite zu stehen, wann immer mein Wissen an seine Grenzen stieß. Besonders hervorheben möchte ich meinen Mitbetreuer, DI Tobias Marauli, sowie DI Christian Zauner und Ing. Kurt Stenzel, die mit ihrer Unterstützung maßgeblich zum Gelingen dieser Arbeit beigetragen haben.

Ein großer Dank gilt auch meinen Studienkollegen, die mich sowohl akademisch als auch persönlich begleitet haben. Die gemeinsamen Gespräche und die Unterstützung während unserer Studienzeit haben diese unvergesslich gemacht.

Zu guter Letzt möchte ich meiner Familie danken – meinen Eltern, meinem Bruder, meiner Freundin und meinen Großeltern. Ohne eure bedingungslose Unterstützung wäre es mir weitaus schwerer gefallen, das Studium erfolgreich abzuschließen. Ihr habt mir in den schwierigsten Phasen immer wieder Kraft gegeben, und dafür bin ich euch unendlich dankbar.

Lukas Probst
Linz, am 11.Oktober 2024

Kapitel 1

Einleitung

In der heutigen Industrie ist die Robotik in vielen Sektoren essenziell, um durch ein hohes Maß an Automatisierung Produktqualität, Wirtschaftlichkeit und Wettbewerbsfähigkeit sicherzustellen. So auch in der Zerspannungstechnik, wo langsam die Anwendung von Industrierobotern Einzug findet und herkömmliche Bearbeitungszentren (BAZ) und Portalmaschinen verdrängen. Besonders in der Nachbearbeitung von komplexeren Hybridbauteilen in der Automobil- und Luftfahrtindustrie durch Sägen, Bohren und Schleifen ist dies der Fall. Anwendungsfeldern in denen der Industriepartner Fill, der diese Arbeit unterstützt, durch Produkte wie den Accubot für Bohranwendungen an Aerospace-Komponenten (dargestellt in Abbildung 1.1), oder den Grind Performer zur Nachbearbeitung von Guss und Schmiedeteilen im Automotive Bereich (dargestellt in Abbildung 1.2) stark vertreten ist.

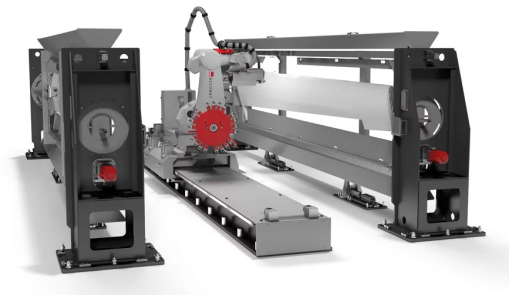


Abbildung 1.1: Accubot bei Bohrung einer Landeklappe [30]

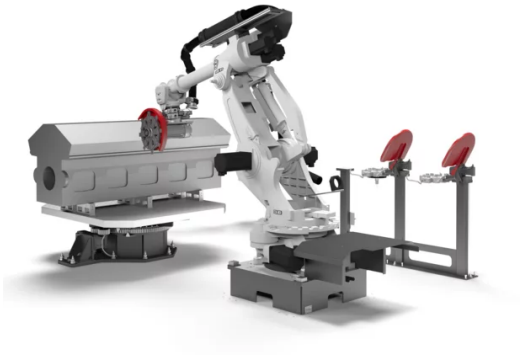


Abbildung 1.2: Grind Performer bei Nachbearbeitung eine Gussteils [31]

Mit der kontinuierlichen Verkleinerung der Fertigungstoleranzen, die durch Produktoptimierungen erforderlich wird, steigen aber die Anforderungen an die Genauigkeit der Fertigungsprozesse wobei herkömmliche Industrieroboter im Gegensatz zu BAZs Schwierigkeiten aufweisen diesen Anforderungen gerecht zu werden. Dennoch würde ein Einsatz von Industrierobotern viele Vorteile aufgrund geringer Investitionskosten, geringem Bauvolumen und hoher Flexibilität bieten. Die Herausforderung besteht nun darin, kostengünstige Lösungen zu finden, die die Fertigungsgenauigkeit bestehender Industrieroboter und Anlagen unter minimalem Aufwand verbessern. Eine bewährte

Methode zur Erhöhung der Genauigkeit ist die Kalibrierung von Robotern [25] und [26], mit welcher absolute Genauigkeiten im Bereich von 0.3 mm - 0.5 mm erreicht werden können. Sie weist jedoch auch einige Schwachstellen auf wie, beispielsweise den zeitlichen Aufwand der Kalibrierung und die begrenzte Flexibilität bei Veränderungen der Fertigungsumgebung. Zusätzlich sind diese Systeme verhältnismäßig teuer und somit für manche Anwendungen nicht immer wirtschaftlich. Eine andere Methode laut [43] ist der Verbesserung der Sensorsysteme z.B. durch externe Encoder mit welcher Genauigkeiten um 0.3 mm erreicht werden können. Hier liegt der Nachteil darin, dass die Installation dieser Sensorsysteme jedoch sehr aufwendig und kostenintensiv ist und nicht alle Fehlerarten kompensiert werden können. Eine weitere Methode ist die online Regelung auf Positionsdaten mit Genauigkeiten im Bereich von 0.2 mm wie sie z.B. in [34] und [25] beschrieben sind. Hier werden vorwiegend Lasertracking Systeme zur Detektion der Endeffektorpose des Industrieroboters verwendet, welche relativ kostenintensiv und einige Einschränkungen in der Platzierung und Detektion im Arbeitsraum haben.

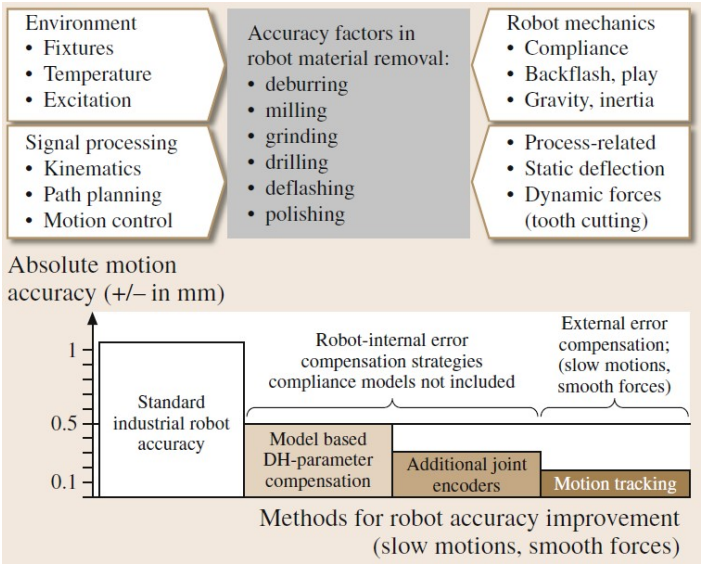


Abbildung 1.3: Absolute Genauigkeit in der Robotik, abgleitet aus [43], zur Verfügung gestellt von Fill Gesellschaft m.b.H

Durch die Fortschritte in der Bildverarbeitung und die revolutionären Entwicklungen im Bereich des maschinellen Lernens – angetrieben durch die stetig wachsende Rechenleistung und den technologischen Fortschritt – eröffnet sich ein neues Feld für den Einsatz kamerabasierter Systeme, um die beschriebenen Probleme in der Genauigkeit von Industrierobotern zu lösen. Der Einsatz hier führt allerdings zu neuen Herausforderungen. Um die geforderte Genauigkeitssteigerung erzielen zu können, sind hohe Bildraten und Auflösungen erforderlich, was zu enormen Datenmengen führt. Hier setzt das EU-Projekt Sprinter an, welches die vorliegende Arbeit unterstützt und sich dem Ziel verschrieben hat, robuste und extrem schnelle Technologien zur Datenübertragung mittels Glasfaser für den industriellen Einsatz zu entwickeln.

Ziel dieser Arbeit ist es nun die traditionellen Methoden zur Steigerung der Fertigungsgenauigkeit mit kamerabasierten Ansätzen zu kombinieren, um flexible und leistungsfähige Lösungen zu entwickeln, die den gestiegenen Anforderungen an die Präzision in der modernen Fertigung gerecht werden. Hierzu werden zwei Konzepte vorgestellt die miteinander verglichen werden, um herauszufinden, welche sich am besten eignen und welche Genauigkeiten erzielbar sind.

Der Fokus liegt hierbei darauf, eine höhere absolute Genauigkeit als bisherige Verfahren zu erzielen, welche nach Abbildung 1.3 im Bereich von 0.3 mm liegen und es werden somit Genauigkeiten im Zehntelmillimeterbereich angestrebt.

Im Aufbau der Arbeit wird zuerst in Kapitel 2 auf die für diese Konzepte zu Grunde liegende Modellbildung eingegangen. Anschließend wird der Einsatz von Kamerasystemen beim Tracken der Roboterpose in Kapitel 3 genauer beleuchtet. In Kapitel 4 wird auf das erste Konzept, welches die geometrische Kalibrierung umfasst eingegangen und in Kapitel 5 auf das zweite betrachtete Konzept, das die kamerabasierten Regelung, behandelt. Diese Konzepte werden anhand von Versuchen in Kapitel 6 miteinander verglichen. In Kapitel 7 wird außerdem ein weiteres Einsatzgebiet der Kamerasysteme für industrielle Anwendungen aufgezeigt, bis schlussendlich in Kapitel 8 die Ergebnisse interpretiert und eingeordnet werden.

Kapitel 2

Modellbildung

Die Steuerung und Regelung von Robotern stützt sich primär auf kinematische Modelle. In vielen Regelungskonzepten werden zusätzlich dynamische Modelle eingesetzt, die nicht nur zur Verbesserung der Regelgenauigkeit beitragen, sondern auch zur Vorabsimulation von Bewegungsabläufen und zur Erstellung eines digitalen Zwillings des Roboters dienen. Standardmodelle weisen jedoch aufgrund notwendiger Annahmen und Vereinfachungen Fehler auf, die zu Ungenauigkeiten in der Positionierung führen. Oft ist es hilfreich diese Fehler mit zu modellieren, um eine Erhöhung der Positioniergenauigkeit zu erreichen. Im Folgenden wird die Modellierung eines Sechsachs-Industrieroboters anhand eines ABB IRB1200 mit einer Reichweite von 0.7 m und einer maximalen Traglast von 7 kg beschrieben.

2.1 Kinematik

Dieser Abschnitt befasst sich mit der Beschreibung der Bewegung des für die Versuche verwendeten Roboters. Allgemein kann ein Roboter durch Strukturelemente modelliert werden, die über Gelenke miteinander verbunden sind. Die Bewegung des IRB1200 kann hierbei gut durch 7 Starrkörper und 6 unabhängige Drehgelenke beschrieben werden. Jeder freie Starrkörper hat im 3 dimensionalen Raum 6 Freiheitsgrade (3 in der Position und 3 in der Orientierung). Bei der Kopplung eines Starrkörpers mit einem anderen über ein unabhängiges Drehgelenk ergeben sich 5 Zwangsbedingungen. Ist die Basis fix mit dem Boden verbunden, so ergeben sich, für diese 6 weitere Zwangsbedingungen. Somit verfügt der IRB1200 insgesamt über 6 Freiheitsgrade, was bedeutet, dass 6 Koordinaten ausreichen, um seine Position und Orientierung eindeutig zu beschreiben. Zur Beschreibung dieser Freiheitsgrade werden die Gelenkwinkel q_i, $i = 1..6$ als Minimalkoordinaten verwendet. Diese Minimalkoordinaten bilden die Gelenkposition $\mathbf{q}$ im Konfigurationsraum, welcher durch die Gelenkwinkelgrenzen beschränkt ist.

Meist ist jedoch die Position und Orientierung des Endeffektors E in einem Inertialsystem I interessant. Die Position ${}_I\mathbf{r}_{IE} = (x\, y\, z)^T$ wird durch kartesische Koordinaten beschrieben. Für die Beschreibung der Orientierung gibt es jedoch eine Vielzahl an gängigen Methoden, im Zuge der Arbeit wurde sich hier auf Euler Winkel ${}_I\boldsymbol{\varphi}_{IE} = (\alpha\, \beta\, \gamma)^T$

nach der ZYX Konvention festgelegt, da auch ABB diese verwendet. Position und Orientierung wird als Pose $\mathbf{z}_{IE}$ zusammengefasst und liegt im Operationsraum.

Nachteil der Euler Winkel ist jedoch, dass diese darstellungsbedingt eine Singulärstelle bei $\beta = \pm 90°$ aufweisen und zusätzlich in der Nähe dieser eine kleine Änderung der Orientierung im Raum zu einer großen Änderung der Euler Winkel führen kann. Aus diesem Grund wird in Abschnitt 2.2 der Abstand zwischen zwei Posen über die Achse-Winkeldarstellung definiert.

Das Pendant zur Pose auf Geschwindigkeitsebene ist der Twist $\mathbf{V}_{IE}$. Dieser umfasst die Translationsgeschwindigkeit des Endeffektors ${}_I\mathbf{v}_{IE}$ und die Winkelgeschwindigkeit ${}_I\boldsymbol{\omega}_{IE}$ des Endeffektors. Bei der zeitliche Änderung der Orientierung wird auf die Winkelgeschwindigkeit übergegangen, dadurch ist es möglich Darstellungssingularitäten, die sich aus Repräsentation der Orientierung durch Euler Winkel ergeben, auf Geschwindigkeitsebene zu umgehen.

2.1.1 Vorwärtskinematik

Die Vorwärtskinematik ist eine Abbildung vom Konfigurationsraum in den Operationsraum. Zur Ermittlung dieser gibt es verschiedene Methoden. Bei der verwendeten Methode angelehnt an [16], werden an jedem der 7 Starrkörper (0 - 6), nach Abbildung 2.1 ein körperfestes Koordinatensystem eingeführt. Zusätzlich gibt es auch ein Inertialsystem I und ein körperfestes Endeffektor-Koordinatensystem E, welches am letzten Körper sitzt und eine Verschiebung des Werkzeug darstellt.

Die Orientierung von einem Koordinatensystem relativ zum Vorgänger wird über Drehmatrizen beschrieben. Zur Bildung dieser wird die Euler-Rodrigues Formel (2.1), mit dem Einheitsvektor ${}_i\mathbf{e}_{i,i+1}$ entlang der Drehachse und dem Drehwinkel $\varphi_{i,i+1}$ angewandt

$$\mathbf{R}_{i,i+1} = \exp\left({}_i\tilde{\mathbf{e}}_{i,i+1}\,\varphi_{i,i+1}\right) = \mathbf{I} + (1 - \cos(\varphi_{i,i+1}))\ {}_i\tilde{\mathbf{e}}^2_{i,i+1} + {}_i\tilde{\mathbf{e}}_{i,i+1}\,\sin(\varphi_{i,i+1})\,. \quad (2.1)$$

Der Operator $\sim$ beschreibt die Abbildung eines Vektors in eine zugehörige schiefsymmetrische Matrix

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \leftrightarrow \tilde{\mathbf{v}} = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}. \quad (2.2)$$

Diese schiefsymmetrische Matrix, repräsentiert das Kreuzprodukt eines Vektors $\mathbf{v}$ mit einem anderen Vektor als Matrixmultiplikation. Umgekehrt lässt sich eine solche Matrix auch durch einen Vektor repräsentieren.

Über die Drehmatrizen der relativen Verdrehung der einzelnen körperfesten Koordinatensysteme

$$\mathbf{R}_{I0} = \exp(\tilde{\mathbf{e}}_z\,\alpha_{I0})\,, \qquad \mathbf{R}_{01} = \exp(\tilde{\mathbf{e}}_z\,q_1)\,, \qquad \mathbf{R}_{12} = \exp(\tilde{\mathbf{e}}_y\,q_2)\,, \quad (2.3)$$

$$\mathbf{R}_{23} = \exp(\tilde{\mathbf{e}}_y\,q_3)\,, \qquad \mathbf{R}_{34} = \exp(\tilde{\mathbf{e}}_x\,q_4)\,, \qquad \mathbf{R}_{45} = \exp(\tilde{\mathbf{e}}_y\,q_5)\,, \quad (2.4)$$

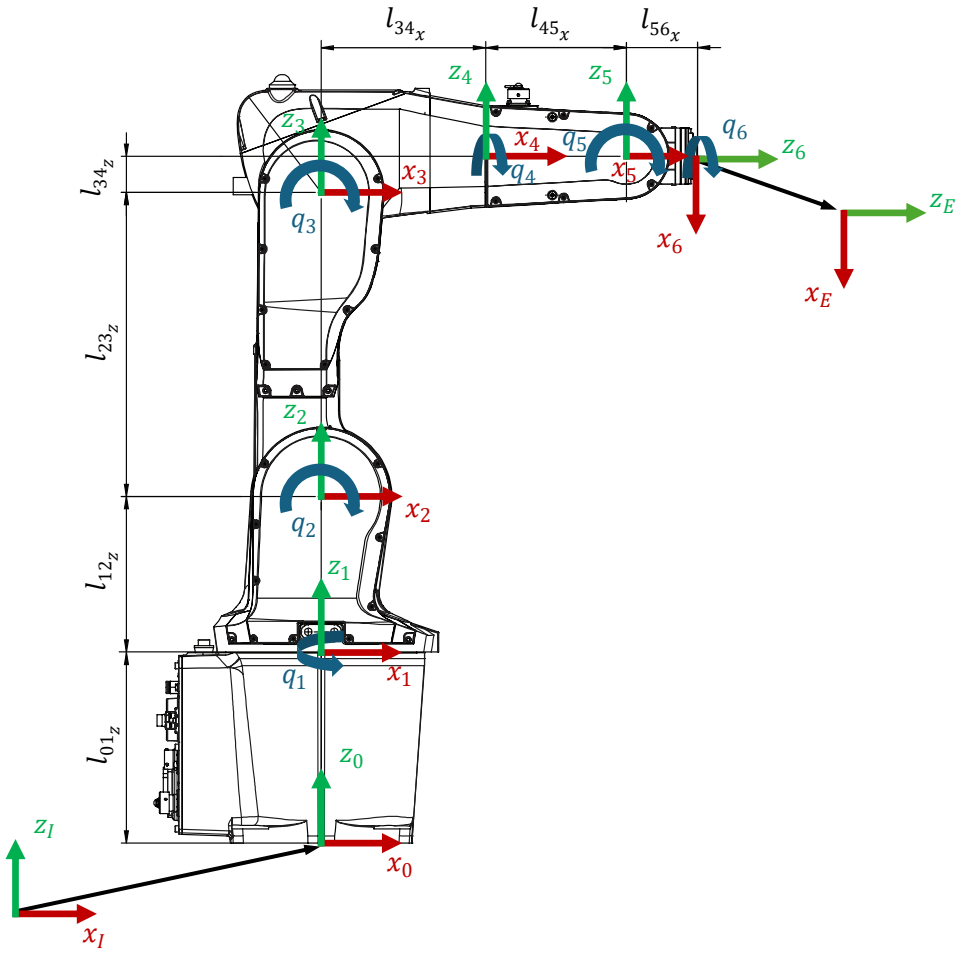


Abbildung 2.1: Definition der Koordinatensysteme des ABB - IRB1200

$$\mathbf{R}_{56} = \exp\left(\tilde{\mathbf{e}}_y \frac{\pi}{2}\right) \exp(\tilde{\mathbf{e}}_z\, q_6)\,, \quad \mathbf{R}_{6E} = \underbrace{\exp(\tilde{\mathbf{e}}_z\, \alpha_{6E}) \exp(\tilde{\mathbf{e}}_y\, \beta_{6E}) \exp(\tilde{\mathbf{e}}_x\, \gamma_{6E})}_{\mathbf{f}_{E2R}({}_6\boldsymbol{\varphi}_{6E})} \tag{2.5}$$

ergibt sich die Verdrehung des Endeffektors gegenüber des Inertialsystems

$$\mathbf{R}_{IE}\,(\mathbf{q}) = \prod_{i=I}^{6} \mathbf{R}_{i,i+1} = \mathbf{R}_{I0}\,\mathbf{R}_{01}\,...\,\mathbf{R}_{6E}\,. \tag{2.6}$$

Hierbei stellen $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ die elementaren Einheitsvektoren dar. Durch Transformation der Drehmatrix in Eulerwinkel ergibt sich schlussendlich die Orientierung ${}_I\boldsymbol{\varphi}_{IE}$. Diese Transformation ist in [16] beschrieben und es wird darauf im Folgenden nicht eingegangen. Die Bezeichnung erfolgt mit ${}_i\boldsymbol{\varphi}_{i,i+1} = \mathbf{f}_{R2E}\,(\mathbf{R}_{i,i+1})$. Die umgekehrte Funktion sieht wie folgt aus

$$\mathbf{R}_{i,i+1} = \mathbf{f}_{E2R}\,({}_i\boldsymbol{\varphi}_{i,i+1}) = \exp(\tilde{\mathbf{e}}_z\, \alpha)\, \exp(\tilde{\mathbf{e}}_y\, \beta)\, \exp(\tilde{\mathbf{e}}_x\, \gamma)\,. \tag{2.7}$$

Die relative Verschiebungen von einem Koordinatensystem zum nächsten wird mit den Ortsvektoren ${}_i\mathbf{r}_{i,i+1} = (l_x\, l_y\, l_z)^T$ beschrieben. Dies erfolgt mittels den geometrischen

Abmessungen aus den Konstruktionsdaten des Roboters und wird dargestellt in Tabelle 2.1. Diese definierten Längen bilden zusammen mit der Orientierung des Endeffektor System ${}_6\boldsymbol{\varphi}_{6E}$ und der Verdrehung des Inertialsystems α_{I0} den Vektor der nominellen Parameter $\mathbf{p}$.

Tabelle 2.1: Relative Verschiebung der Koordinatensysteme des IRB1200

${}_i\mathbf{r}_{i,i+1}$	l_x		l_y		l_z	
	$\mathbf{p}$	Wert	$\mathbf{p}$	Wert	$\mathbf{p}$	Wert
${}_I\mathbf{r}_{I0}$	l_{I0_x}	-0.265 m	l_{I0_y}	0.4 m	l_{I0_z}	0.498 m
${}_0\mathbf{r}_{01}$	-	0	-	0	l_{01_z}	0.219 m
${}_1\mathbf{r}_{12}$	-	0	-	0	l_{12_z}	0.1801 m
${}_2\mathbf{r}_{23}$	-	0	-	0	l_{23_z}	0.350 m
${}_3\mathbf{r}_{34}$	l_{34_x}	0.1885 m	-	0	l_{34_z}	0.042 m
${}_4\mathbf{r}_{45}$	l_{45_x}	0.1625 m	-	0	-	0
${}_5\mathbf{r}_{56}$	l_{56_x}	0.082 m	-	0	-	0
${}_6\mathbf{r}_{6E}$	l_{6E_x}	0 m	l_{6E_y}	0 m	l_{6E_z}	0 m

Über

$$ {}_I\mathbf{r}_{IE}(\mathbf{q}) = \sum_{i=I}^{6} \left[\mathbf{R}_{I,i}\, {}_i\mathbf{r}_{i,i+1}\right] = {}_I\mathbf{r}_{I0} + \mathbf{R}_{I0}\, {}_0\mathbf{r}_{01} + \ldots + \mathbf{R}_{I6}\, {}_6\mathbf{r}_{6E} \tag{2.8} $$

wird die Endeffektorposition ermittelt und somit ergibt sich die Funktion der Vorwärtskinematik zu

$$ \mathbf{z}_{IE} = \begin{pmatrix} {}_I\mathbf{r}_{IE}(\mathbf{q}) \\ {}_I\boldsymbol{\varphi}_{IE}(\mathbf{q}) \end{pmatrix} = \mathbf{z}_{IE}(\mathbf{q})\,. \tag{2.9} $$

Ebenfalls existiert eine Abbildung zwischen dem Tangentialraum des Konfigurationsraumes und dem des Operationsraumes, welche als Vorwärtskinematik auf Geschwindigkeitsebene bezeichnet wird. Zur Ermittlung dieser wird wieder nach [16] vorgegangen.

Die Endeffektor Geschwindigkeit im Inertialsystem erhält man durch die zeitliche Ableitung der Position

$$ {}_I\mathbf{v}_{IE} = \frac{\mathrm{d}\,{}_I\mathbf{r}_{IE}(\mathbf{q}(t))}{\mathrm{d}t} = \frac{\partial\, {}_I\mathbf{r}_{IE}(\mathbf{q})}{\partial \mathbf{q}}\,\dot{\mathbf{q}}\,. \tag{2.10} $$

Die Winkelgeschwindigkeit des Endeffektors ${}_I\boldsymbol{\omega}_{IE}$ wird über die relative Winkelgeschwindigkeit der einzelnen Körper ${}_i\boldsymbol{\omega}_{i,i+1}$ gebildet und erfolgt analog zu (2.8), wobei sich diese über die Drehachsen ${}_i\mathbf{e}_{i,i+1}$ und die Gelenkgeschwindigkeiten $\dot{q}_i$ wie folgt berechnen lassen

$$ {}_I\boldsymbol{\omega}_{I0} = \mathbf{0}\,, \quad {}_0\boldsymbol{\omega}_{01} = \mathbf{e}_z\,\dot{q}_1\,, \quad {}_1\boldsymbol{\omega}_{12} = \mathbf{e}_y\,\dot{q}_2\,, \quad {}_2\boldsymbol{\omega}_{23} = \mathbf{e}_y\,\dot{q}_3\,, $$
$$ {}_3\boldsymbol{\omega}_{34} = \mathbf{e}_x\,\dot{q}_4\,, \quad {}_4\boldsymbol{\omega}_{45} = \mathbf{e}_y\,\dot{q}_5\,, \quad {}_5\boldsymbol{\omega}_{56} = \mathbf{e}_x\,\dot{q}_6\,, \quad {}_6\boldsymbol{\omega}_{6E} = \mathbf{0}\,. $$

Die Vorwärtskinematik auf Geschwindigkeitsebene ergibt sich somit zu

$$\mathbf{V}_{IE} = \begin{pmatrix} {}_I\mathbf{v}_{IE}\left(\mathbf{q}, \dot{\mathbf{q}}\right) \\ {}_I\boldsymbol{\omega}_{IE}\left(\mathbf{q}, \dot{\mathbf{q}}\right) \end{pmatrix} = \mathbf{V}_{IE}\left(\mathbf{q},\, \dot{\mathbf{q}}\right) \tag{2.11}$$

und ist linear in $\dot{\mathbf{q}}$, weshalb sie sich ebenfalls durch die geometrische Jacobimatrix

$$\mathbf{J}\left(\mathbf{q}\right) = \frac{\partial \mathbf{V}_{IE}\left(\mathbf{q},\, \dot{\mathbf{q}}\right)}{\partial \dot{\mathbf{q}}} \tag{2.12}$$

darstellen lässt.

2.1.2 Inverskinematik

Die Umkehrfunktion der Vorwärtskinematik wird als Inverskinematik bezeichnet und bildet vom Operationsraum in den Konfigurationsraum ab. Im Allgemeinen ist die Existenz dieser Umkehrfunktion für Industrieroboter nicht garantiert. Für die spezielle Klasse der Sechsachs-Zentralhandroboter unter die der IRB1200 fällt, lässt sich jedoch eine analytische Umkehrfunktion finden. Um eine eindeutige Zuordnung der Gelenkposition zur Pose zu ermöglichen, müssen die Parameter $c_1, c_2 \in \{-1, 1\}$ eingeführt werden, die Aufschluss über die Konfiguration des Roboters geben. Durch eine Entkopplung um den Schnittpunkt der Achsen der Zentralhand (bei dem IRB1200 ist dies der 5te Gelenkpunkt) lassen sich über geometrische Überlegungen basierend auf [16], analytische Gleichungen für die Gelenkposition ermitteln. Im Folgenden wird auf diese analytischen Gleichungen kurz eingegangen.

Der Vektor zwischen erstem und fünften Gelenkpunkt, lassen sich durch Kenntnis der Pose und der nominellen Parameter nach

$${}_I\mathbf{r}_{15}(\mathbf{z}_{IE}) = {}_I\mathbf{r}_{IE} - {}_I\mathbf{r}_{I1} - \left(\mathbf{R}_{IE}\,\mathbf{R}_{6E}^T\right)\left({}_6\mathbf{r}_{56} + {}_6\mathbf{r}_{6E}\right) = \begin{pmatrix} l_{15_x} \\ l_{15_y} \\ l_{15_z} \end{pmatrix} \tag{2.13}$$

berechnen. Hiermit ist es wie in Abbildung 2.2 ersichtlich, möglich die Gelenkposition q_1 zu ermitteln. Diese ergibt sich aus

$$\tan\left(q_1 + \alpha_{I0} - \frac{c_1 - 1}{2}\pi\right) = \frac{l_{15_y}}{l_{15_x}} \tag{2.14}$$

zu

$$q_1 = \operatorname{atan2}(l_{15_y}, l_{15_x}) + \frac{c_1 - 1}{2}\pi - \alpha_{I0}\,. \tag{2.15}$$

Durch Kenntnis von q_1 lässt sich nun ${}_1\mathbf{r}_{25}$ mit

$${}_1\mathbf{r}_{25}\left(q_1,\, \mathbf{z}_{IE}\right) = \mathbf{R}_{I1}^T\,{}_I\mathbf{r}_{15} - {}_1\mathbf{r}_{12} = \begin{pmatrix} l_{25_x} \\ 0 \\ l_{25_z} \end{pmatrix}, \quad l_{25} = ||{}_1\mathbf{r}_{25}|| \tag{2.16}$$

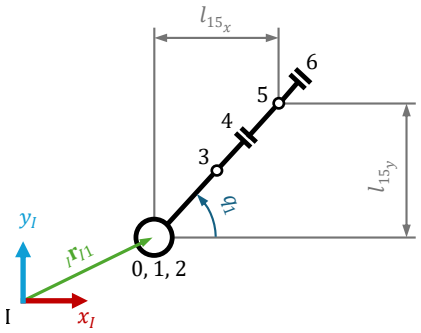


Abbildung 2.2: Vereinfachte Geometrie des IRB1200 (Draufsicht)

berechnen und in weiterer Folge wie in Abbildung 2.3 ersichtlich q_2 durch

$$q_2 = \frac{\pi}{2} - \beta - c_2\alpha\,, \tag{2.17}$$

wobei sich α aus dem Kosinus-Satz zu

$$\alpha = \arccos\left(\frac{l_{23_z}^2 + l_{25}^2 - l_{35}^2}{2\,l_{23_z}\,l_{25}}\right) \tag{2.18}$$

ergibt. Die Länge l_{35} lässt sich hierbei durch

$${}_3\mathbf{r}_{35} = {}_3\mathbf{r}_{34} + \mathbf{R}_{34}\,{}_4\mathbf{r}_{45} = \begin{pmatrix} l_{35_x} \\ 0 \\ l_{35_z} \end{pmatrix} = \begin{pmatrix} l_{34_x} + l_{45_x} \\ 0 \\ l_{34_z} \end{pmatrix}, \quad l_{35} = ||{}_3\mathbf{r}_{35}|| \tag{2.19}$$

berechnen. Der Winkel β ergibt sich über die in Abbildung 2.3 dargestellte Geometrie zu

$$\beta = \operatorname{atan2}(l_{25_z}, l_{25_x})\,. \tag{2.20}$$

Des Weiteren lässt sich über die Umformung der Gleichung

$$\cos(\pi - \delta) = \frac{l_{23_z}^2 + l_{35}^2 - l_{25}^2}{2\,l_{23_z}\,l_{35}}\,, \tag{2.21}$$

welche sich ebenfalls aus einer Anwendung des Kosinus-Satzes ergibt, der Ausdruck

$$q_3 = \frac{\pi - \arccos\left(\frac{l_{23_z}^2 + l_{35}^2 - l_{25}^2}{2\,l_{23_z}\,l_{35}}\right)}{c_2} + \gamma - \frac{\pi}{2} \tag{2.22}$$

für die Berechnung von q_3 ermitteln. Hier ergeben sich die Winkel δ und γ zu

$$\delta = c_2\left(\frac{\pi}{2} - \gamma + q_3\right), \quad \gamma = \operatorname{atan2}(l_{35_z}, l_{35_x})\,. \tag{2.23}$$

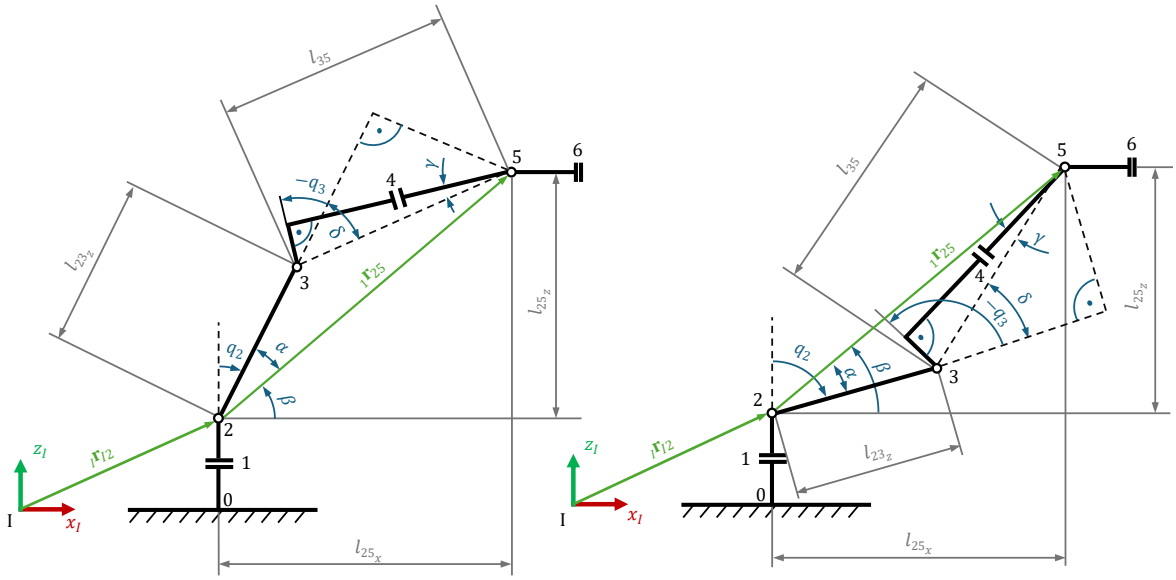


Abbildung 2.3: Vereinfachte Geometrie des IRB1200 für unterschiedliche Konfigurationen (Seitenansicht)

Durch die Kenntnis von q_1, q_2 und q_3 lassen sich die restlichen Gelenkswinkel über die Drehmatrix

$$\mathbf{R}_{36}\left(q_1,\, q_2,\, q_3\right) = \mathbf{R}_{I3}^{T}\, \mathbf{R}_{IE}\, \mathbf{R}_{6E}^{T} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{2.24}$$

bestimmen. Hierfür ergeben sich nach den in [16] ausgeführten Überlegungen, die Zusammenhänge

$$q_4 = \operatorname{atan2}(a_{23}, -a_{33})\,, \tag{2.25}$$

$$q_5 = \operatorname{atan2}(\sin(q_4)\, a_{23} - \cos(q_4)\, a_{33}, a_{13})\,, \tag{2.26}$$

$$q_6 = \operatorname{atan2}(a_{12}, -a_{11})\,. \tag{2.27}$$

für die restlichen Gelenkswinkel.

Auf Geschwindigkeitsebene existiert für den IRB1200 ebenfalls eine Inverskinematik. Da es sich hier um eine in $\dot{\mathbf{q}}$ lineare Abbildung handelt ist hier die Inverskinematik wesentlich leichter zu finden und zu berechnen. Aus diesem Grund wird $\mathbf{q}$ auch oft numerisch aus der Inverskinematik der Geschwindigkeitsebene ermittelt. Jedoch existiert nicht für jede Gelenksposition $\mathbf{q}$ eine Inverse, sondern es treten Gelenkspositionen $\mathbf{q}_S$ auf, die Singularitäten, an denen die Jacobimatrix singulär ist und somit nicht frei in jede beliebige Richtung des Operationsraumes Geschwindigkeit aufgebaut werden kann. In der Nähe dieser Konfigurationen existieren zusätzlich Bereiche in denen die theoretische Lösung außerhalb der Gelenkwinkelgeschwindigkeitsgrenzen liegt. Prinzipiell gibt es hier verschiedene Methoden dieses Problem zu lösen. Eine Methode ist eine Abbildung die eine Näherungslösung in diesen Bereichen definiert, ansonsten die inverse Funktion exakt repliziert. Diese Abbildung wird als robuste

Inverse $\mathbf{J}^+$ bezeichnet und wird in [16] beschrieben. Kurz zusammengefasst wird dabei der Geschwindigkeitsfehler minimiert mit dem quadratischen Optimierungsproblem

$$\dot{\mathbf{q}}^* = \arg\left(\min_{\dot{\mathbf{q}}}\left(\frac{1}{2}||\mathbf{V}_{IE} - \mathbf{J}\left(\mathbf{q}\right)\,\dot{\mathbf{q}}||_2^2 + \frac{1}{2}||\alpha\,\dot{\mathbf{q}}||_2^2\right)\right) \tag{2.28}$$

und der notwendigen Optimalitätsbedingung

$$\nabla_{\dot{\mathbf{q}}}\,\frac{1}{2}\left(||\mathbf{V}_{IE} - \mathbf{J}\,\dot{\mathbf{q}}||_2^2 + ||\alpha\,\dot{\mathbf{q}}||_2^2\right) = 0\,. \tag{2.29}$$

Diese führt zu

$$\mathbf{J}^T\,\mathbf{J}\,\dot{\mathbf{q}} - \mathbf{J}^T\,\mathbf{V}_{IE} + \alpha^2\,\dot{\mathbf{q}} = 0\,, \tag{2.30}$$

woraus sich die explizite Lösung

$$\dot{\mathbf{q}}^* = \left(\mathbf{J}^T\mathbf{J} + \alpha^2\mathbf{I}\right)^{-1}\mathbf{J}^T\,\mathbf{V}_{IE} \tag{2.31}$$

ergibt. Ein Dämpfungsfaktor $\alpha > 0$ bewirkt eine Reduktion der maximalen Gelenkgeschwindigkeiten, führt aber automatisch zu einem Fehler im Endeffektor-Twist, weshalb dieser dynamisch gewählt wird, um in der Nähe von Singularitäten die Geschwindigkeiten zu dämpfen, ansonsten jedoch keine Änderung vorzunehmen. Die Bewertung der Nähe zu einer Singularität erfolgt mit der kinematischen Manipulierbarkeit

$$w_{kin}(\mathbf{q}) = \sqrt{\det\left(\mathbf{J}(\mathbf{q})\,\mathbf{J}^T(\mathbf{q})\right)} \tag{2.32}$$

und führt somit zur folgenden dynamischen Wahl des Dämpfungsfaktors

$$\alpha\left(\mathbf{q}\right) = \begin{cases} \alpha_0\left(1 - \frac{w_{kin}(\mathbf{q})}{w_0}\right)^2 & w_{kin}\left(\mathbf{q}\right) < w_0 \\ 0 & w_{kin}\left(\mathbf{q}\right) \geq w_0 \end{cases} \tag{2.33}$$

wobei $\alpha_0 = 1 \cdot 10^{-4}$ und $w_0 = 1 \cdot 10^{-4}$ gewählt wurden.

Es ergibt sich somit die robuste Invertierung

$$\dot{\mathbf{q}} = \mathbf{J}^+\left(\mathbf{q}\right)\,\mathbf{V}_{IE} \tag{2.34}$$

$$\mathbf{J}^+\left(\mathbf{q}\right) = \left(\mathbf{J}^T\left(\mathbf{q}\right)\,\mathbf{J}\left(\mathbf{q}\right) + \alpha^2\left(\mathbf{q}\right)\mathbf{I}\right)^{-1}\mathbf{J}^T\left(\mathbf{q}\right)\,. \tag{2.35}$$

2.2 Abstand zwischen Posen

Der Abstand zwischen zwei Posen ist sowohl für die Regelung und Kalibrierung als auch für deren Bewertung von entscheidender Bedeutung und sollte möglichst frei von Fehlern aufgrund von Darstellungssingularitäten sein. Daher wird der Abstand über die Achse-Winkel-Darstellung wie folgt definiert

$$\Delta\mathbf{z} = \mathbf{e}\left(\mathbf{z}_d, \mathbf{z}\right) = \begin{pmatrix} {}_I\mathbf{r}_d - {}_I\mathbf{r} \\ \mathbf{e}_O \end{pmatrix}\,, \tag{2.36}$$

wobei $\mathbf{z}$ und $\mathbf{z}_d$ zwei beliebige Posen sind zum Beispiel eine Sollpose d und die gemessene Pose. Der Orientierungsfehler $\mathbf{e}_O$ ergibt sich zu

$$\mathbf{e}_O = \mathbf{e}_R \, \Delta\varphi \,, \tag{2.37}$$

mit dem Einheitsvektor $\mathbf{e}_R$ entlang der Drehachse und dem Drehwinkel $\Delta\varphi$. Diese lassen sich wiederum aus den Eulerwinkeln $\boldsymbol{\varphi}_d$ bzw. $\boldsymbol{\varphi}$ ermitteln, indem die zugehörigen Drehmatrizen $\mathbf{R}_d$ bzw. $\mathbf{R}$ berechnet werden und die Beziehungen

$$\Delta\varphi = \cos^{-1}\left(\frac{\mathrm{tr}(\mathbf{R}_d\,\mathbf{R}^T) - 1}{2}\right) \tag{2.38}$$

und

$$\tilde{\mathbf{e}}_R = \frac{1}{2\,\sin\Delta\varphi}\left(\mathbf{R}_d\,\mathbf{R}^T - \mathbf{R}\,\mathbf{R}_d^T\right) \tag{2.39}$$

angewandt werden. Auf die Transformation der in der Pose enthaltenen Eulerwinkel in die Drehmatrizen $\mathbf{R}_d$ bzw. $\mathbf{R}$ wird genauer in Abschnitt 2.1.1 eingegangen.

Als Maß für nur kleine Änderungen in der Orientierung wird auch

$$\Delta\mathbf{z} = \bar{\mathbf{e}}\,(\mathbf{z}_d, \mathbf{z}) = \begin{pmatrix} {}_I\mathbf{r}_d - {}_I\mathbf{r} \\ \bar{\mathbf{e}}_O \end{pmatrix} \qquad\qquad \bar{\mathbf{e}}_O = \mathbf{e}_R \,\sin\Delta\varphi \tag{2.40}$$

verwendet.

Für den Abstand auf Geschwindigkeitsebene gilt hier zudem

$$\Delta\mathbf{V} = \dot{\bar{\mathbf{e}}}\,(\mathbf{V}_d, \mathbf{V}) = \begin{pmatrix} {}_I\mathbf{v}_d - {}_I\mathbf{v} \\ \tilde{\bar{\mathbf{e}}}_O{}^T \, {}_I\boldsymbol{\omega}_d - \tilde{\bar{\mathbf{e}}}_O \, {}_I\boldsymbol{\omega} \end{pmatrix} = \mathbf{L}^T\mathbf{V}_d - \mathbf{L}\mathbf{V}, \quad \mathbf{L} = \mathrm{diag}\,(\mathbf{I}, \tilde{\bar{\mathbf{e}}}_O) \tag{2.41}$$

was sich wie folgt zeigen lässt

$$\tilde{\bar{\mathbf{e}}}_O = \tilde{\mathbf{e}}_R \sin\Delta\varphi = \frac{1}{2}\left(\mathbf{R}_d\,\mathbf{R}^T - \mathbf{R}\,\mathbf{R}_d^T\right) \tag{2.42}$$

$$\dot{\tilde{\bar{\mathbf{e}}}}_O = \frac{1}{2}\left(\left(\dot{\mathbf{R}}_d\,\mathbf{R}^T + \mathbf{R}_d\,\dot{\mathbf{R}}^T\right) - \left(\dot{\mathbf{R}}_d\,\mathbf{R}^T + \mathbf{R}_d\,\dot{\mathbf{R}}^T\right)^T\right) \tag{2.43}$$

$$= \frac{1}{2}\left(\left(\underbrace{\dot{\mathbf{R}}_d\mathbf{R}_d^T}_{\tilde{\boldsymbol{\omega}}_d}\,\mathbf{R}_d\,\mathbf{R}^T + \mathbf{R}_d\,\left(\underbrace{\dot{\mathbf{R}}\,\mathbf{R}^T}_{\tilde{\boldsymbol{\omega}}}\,\mathbf{R}\right)^T\right) - \left(\dot{\mathbf{R}}_d\,\mathbf{R}^T + \mathbf{R}_d\,\dot{\mathbf{R}}^T\right)^T\right) \tag{2.44}$$

$$= \tilde{\boldsymbol{\omega}}_d \underbrace{\frac{1}{2}\left(\mathbf{R}_d\,\mathbf{R}^T - \mathbf{R}\,\mathbf{R}_d^T\right)}_{\tilde{\bar{\mathbf{e}}}_O} + \underbrace{\frac{1}{2}\left(\mathbf{R}_d\,\mathbf{R}^T - \mathbf{R}\,\mathbf{R}_d^T\right)}_{\tilde{\bar{\mathbf{e}}}_O}\tilde{\boldsymbol{\omega}}^T \tag{2.45}$$

$$= \left(\widetilde{(\tilde{\bar{\mathbf{e}}}_O)^T\,\boldsymbol{\omega}_d} - \tilde{\bar{\mathbf{e}}}_O\,\boldsymbol{\omega}\right). \tag{2.46}$$

2.3 Fehlerkinematik

Das in Abschnitt 2.1 beschriebene Modell bildet die tatsächliche Kinematik nur mit einer gewissen Genauigkeit ab. Grund liegt hier darin, dass die exakte Geometrie eines Roboters selten bekannt ist und sich diese mit der Zeit ändern kann, außerdem müssen Vereinfachungen bzgl. der Drehgelenke getroffen werden. Eine Möglichkeit zur Steigerung der Positioniergenauigkeit eines Industrieroboters basiert auf einem umfassenderen Modell, in diesem jenes aus Abschnitt 2.1 um Fehlerparameter $\bar{\mathbf{p}}_e$ (siehe Abbildung 2.4) erweitert wird. Die in dieser Arbeit berücksichtigten Fehlerparameter können folgendermaßen klassifiziert werden:

- Nulllagenfehler: Hiermit wird die Abweichung der tatsächlichen Achsnullstellung von der modellierten Nullstellung definiert, diese kann unter anderem an Fertigungstoleranzen, Verformungen auf Grund von Abnutzung oder Crashs, fehlerhafter Kalibrierung oder Alterung der Encoder, sowie an Softwarefehlern liegen. Laut [8] liegt der Anteil der Nulllagenfehler an der Positionsungenauigkeit meist im Bereich von 80 % - 90 % und stellt somit den größten Anteil dar. Berücksichtigt werden sie durch die Substitution von q_i durch

$$q_i + p_{q_i}. \tag{2.47}$$

- Längsabweichungen: Als Längsabweichung wird die Abweichung der tatsächlich auftretenden Position der definierten körperfesten Koordinatensysteme von der nominellen Position beschrieben. Gründe für Längsabweichungen können in Fertigungstoleranzen, in fehlerhaftem Zusammenbau, Temperaturausdehnungseffekten, Verformungen auf Grund von Abnutzung oder Crashs liegen. Dieser Fehler macht laut [8] in der Regel zwischen 5 % - 10 % der Positionsungenauigkeit aus. Die Längsabweichung werden durch Erweiterung der Ortsvektoren ${}_i\mathbf{r}_{i,i+1}$ um Fehlerparameter nach der Gleichung

$${}_i\mathbf{r}_{i,i+1_{err}} = \begin{pmatrix} l_x + p_{l_x} \\ l_y + p_{l_y} \\ l_z + p_{l_z} \end{pmatrix} \tag{2.48}$$

  modelliert.

- Achsschiefstellungen: Diese beschreiben eine Abweichung der Orientierung der Drehachse der Drehgelenke. Ungenauigkeiten im Zusammenbau, Verformungen durch Crashs sind Beispiele für die Ursache von Achsschiefstellung und machen wiederum laut [8] ca. 5 % - 10 % der Positionsungenauigkeit aus. Modelliert wird sie durch Erweiterung der Drehmatrizen $\mathbf{R}_{i,i+1}$ um Verdrehungen in den 2 Drehfreiheitsgraden die nicht um die modellierte Drehachse ${}_i\mathbf{e}_{i,i+1}$ erfolgen. Dabei wird angenommen, dass es sich bei den Verdrehungen um sehr kleine Winkel

handelt und somit die resultierenden Drehmatrizen durch ihre Linearisierung ersetzt werden können. Als Beispiel ergibt sich für ${}_i\mathbf{e}_{i,i+1} = \mathbf{e}_y$

$$\mathbf{R}_{i,i+1_{err}} = \left.\frac{\partial \exp\left(\tilde{\mathbf{e}}_x \alpha\right)}{\partial \alpha}\right|_{\alpha=0} p_{\alpha_i}\, \mathbf{R}_{i,i+1} \left.\frac{\partial \exp\left(\tilde{\mathbf{e}}_z \gamma\right)}{\partial \gamma}\right|_{\gamma=0} p_{\gamma_i}\,. \tag{2.49}$$

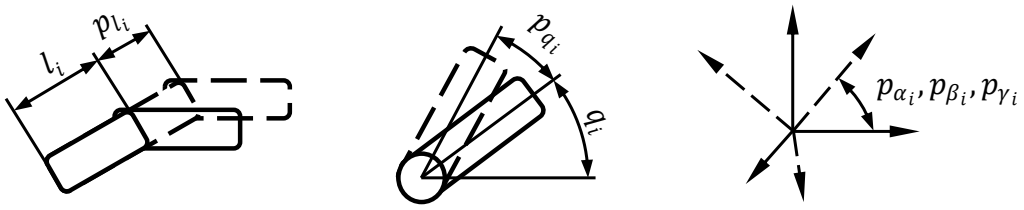


Abbildung 2.4: Fehlerparameter

Somit ergeben sich für jeden Körper 6 Fehlerparameter die zum Fehlerparametervektor $\bar{\mathbf{p}}_e \in \mathbb{R}^{48}$ zusammengefasst werden und mittels der Gleichungen (2.47) - (2.49) folgt die erweiterte Vorwärtskinematik

$$\mathbf{z}_{IE} = \mathbf{z}_{IE}\left(\mathbf{q}, \bar{\mathbf{p}}_e\right) \tag{2.50}$$

und auf Geschwindigkeitsebene

$$\mathbf{V}_{IE} = \mathbf{V}_{IE}\left(\mathbf{q}, \dot{\mathbf{q}}, \bar{\mathbf{p}}_e\right)\,. \tag{2.51}$$

In der Literatur [16] findet man unter anderem auch noch eine weitere geometrische Fehlerart, das Gelenkspiel. Dieses wurde in dieser Arbeit nicht berücksichtigt da es nur einen sehr kleinen Teil der Positionsungenauigkeit ausmacht.

Die Fehlerparameter sind unbekannt und müssen identifiziert werden, beschrieben in Kapitel 4. Die Abbildung vom Raum der Fehlerparameter in den Operationsraum ist jedoch nicht injektiv, was ein Problem bei der Identifikation darstellt. Lösung bietet eine Vereinfachung von (2.50), bei der die Fehlerparameter, die nicht oder nur schwer zu identifizieren sind ausgeschlossen werden. Zusätzlich wird angenommen, dass $\bar{\mathbf{p}}_e$ sehr klein ist wodurch eine Näherung des Modells durch eine Linearisierung um $\bar{\mathbf{p}}_e = \mathbf{0}$ zulässig ist. Es ergibt sich somit die Gleichung

$$\mathbf{z}_{IE} \approx \mathbf{z}_{IE}\left(\mathbf{q}, \mathbf{0}\right) + \left.\frac{\partial \mathbf{z}_{IE}\left(\mathbf{q}, \bar{\mathbf{p}}_e\right)}{\partial \bar{\mathbf{p}}_e}\right|_{\bar{\mathbf{p}}_e = \mathbf{0}} \bar{\mathbf{p}}_e = \mathbf{z}_{IE}\left(\mathbf{q}\right) + \bar{\mathbf{\Theta}}\left(\mathbf{q}\right)\, \bar{\mathbf{p}}_e\,. \tag{2.52}$$

Für das linearisierte Modell können die linear abhängigen Parameter über eine QR-Zerlegung von $\mathbf{\Theta}$ ermittelt werden, hier wird nach [16] vorgegangen. $\mathbf{\Theta} \in \mathbb{R}^{n,p}$ mit $n \geq p$ setzt sich hierbei wie folgt zusammen

$$\mathbf{\Theta} = \begin{pmatrix} \bar{\mathbf{\Theta}}\left(\mathbf{q}_1\right) \\ \vdots \\ \bar{\mathbf{\Theta}}\left(\mathbf{q}_N\right) \end{pmatrix}\,. \tag{2.53}$$

Wobei für die QR-Zerlegungen zufällige aber nicht gleiche Gelenkpositionen $\mathbf{q}_i$ verwendet werden.

Kurz zusammengefasst wird $\boldsymbol{\Theta}$ mittels der QR-Zerlegung $\boldsymbol{\Theta} = \hat{\mathbf{Q}}\,\hat{\mathbf{R}}$, in die obere Dreiecksmatrix $\hat{\mathbf{R}} \in \mathbb{R}^{p,p}$ ($\lambda_i = \hat{r}_{i,i}$) und die spaltenorthonormale Matrix $\hat{\mathbf{Q}} \in \mathbb{R}^{n,p}$ ($\hat{\mathbf{Q}}^T\,\hat{\mathbf{Q}} = \mathbf{I}$) zerlegt. Für die genaue Vorgehensweise bei der QR-Zerlegung sei auf [15] verwiesen.

Es ergibt sich somit die Beziehung

$$\boldsymbol{\vartheta}_i = \sum_{j=1}^{i} \hat{\mathbf{q}}_j\,\hat{r}_{j,i} \quad i = 1 \ldots p \tag{2.54}$$

für die Spalten $\boldsymbol{\vartheta}_i$ von $\boldsymbol{\Theta}$, mit den Spalten $\hat{\mathbf{q}}_j$ von $\hat{\mathbf{Q}}$ und Einträgen $\hat{r}_{j,i}$ von $\hat{\mathbf{R}}$. Für alle Eigenwerte $\lambda_k = \hat{r}_{k,k} = 0$ von $\hat{\mathbf{R}}$ folgt somit, dass jene Spalten $\boldsymbol{\vartheta}_k$ linear abhängig von den vorherigen Spalten sind, da sie sich aus einer Linearkombination der selben Basisvektoren $\hat{\mathbf{q}}_1 \ldots \hat{\mathbf{q}}_{k-1}$ wie die Vorgänger zusammensetzen. Folglich sind auch die zugehörigen Parameter $\bar{p}_{e_k}$ linear abhängig. $\boldsymbol{\Theta}$ kann nun so umsortiert werden, dass sie sich zuerst aus den linear unabhängigen Spalten und anschließend den linear abhängigen zusammensetzt $\boldsymbol{\Theta}_{sort} = (\boldsymbol{\Theta}_{un},\, \boldsymbol{\Theta}_{ab})$. Analog ist dies auch für $\hat{\mathbf{Q}}$ und $\bar{\mathbf{p}}_e$ möglich. Wie in [16] beschrieben lässt sich eine Matrix $\boldsymbol{\kappa}$ finden die den Zusammenhang $\boldsymbol{\Theta}_{un} = \boldsymbol{\Theta}_{ab}\boldsymbol{\kappa}$ beschreibt. Über die lässt sich das Problem wie folgt vereinfachen

$$\boldsymbol{\Theta}_{sort}\,\bar{\mathbf{p}}_{e,sort} = (\boldsymbol{\Theta}_{un},\, \boldsymbol{\Theta}_{ab}) \begin{pmatrix} \mathbf{p}_{un} \\ \mathbf{p}_{ab} \end{pmatrix} = (\boldsymbol{\Theta}_{un},\, \boldsymbol{\Theta}_{un}\boldsymbol{\kappa}) \begin{pmatrix} \mathbf{p}_{un} \\ \mathbf{p}_{ab} \end{pmatrix} = \boldsymbol{\Theta}_{un} \underbrace{(\mathbf{p}_{un} + \boldsymbol{\kappa}\,\mathbf{p}_{ab})}_{\mathbf{p}_e}\,.$$

Es ergibt sich somit das neue Modell für die Fehlerkinematik

$$\mathbf{z}_{IE} = \mathbf{z}_{IE}\,(\mathbf{q}, \mathbf{p}_e\,) \tag{2.55}$$

$$\mathbf{V}_{IE} = \mathbf{V}_{IE}\,(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{p}_e\,) = \mathbf{J}\,(\mathbf{q}, \mathbf{p}_e\,)\;\dot{\mathbf{q}} \tag{2.56}$$

mit dem reduzierten Fehlerparametervektor $\mathbf{p}_e \in \mathbb{R}^{30}$. Außerdem wird im Weiteren $\boldsymbol{\Theta}_{un}$ als neues $\boldsymbol{\Theta}$ bezeichnet.

Für den Fall, dass die Orientierung des Endeffektor nicht bekannt ist, oder nur schlecht gemessen werden kann, werden analog jene Fehlerparameter $\mathbf{p}_{e_{red}} \in \mathbb{R}^{25}$ ermittelt die rein über ${}_I\mathbf{r}_{IE}$ identifizierbar sind. Die Zusammensetzung des reduzierten Fehlerparametervektor ist in Tabelle 2.2 ersichtlich.

Tabelle 2.2: Reduzierte Fehlerparameter

mit Orientierung			ohne Orientierung		
$\mathbf{p}_e$	$\mathbf{p}_{un}$	$\boldsymbol{\kappa}\,\mathbf{p}_{ab}$	$\mathbf{p}_{e_{red}}$	$\mathbf{p}_{un}$	$\boldsymbol{\kappa}\,\mathbf{p}_{ab}$
p_{e_1}	p_{q_1}	p_{γ_0}	p_{e_1}	p_{q_1}	p_{γ_0}
p_{e_2}	p_{q_2}	0	p_{e_2}	p_{q_2}	0
p_{e_3}	p_{q_3}	0	p_{e_3}	p_{q_3}	0
p_{e_4}	p_{q_4}	0	p_{e_4}	p_{q_4}	0
p_{e_5}	p_{q_5}	$p_{\beta_4} + p_{\beta_6}$	p_{e_5}	p_{q_5}	p_{β_4}
p_{e_6}	p_{q_6}	p_{α_E}	p_{e_6}	$p_{l_{I0_x}}$	$-l_{01_z}\,p_{\beta_1} + p_{l_{01_x}}$
p_{e_7}	$p_{l_{I0_x}}$	$-l_{01_z}\,p_{\beta_1} + p_{l_{01_x}}$	p_{e_7}	$p_{l_{I0_y}}$	$l_{01_z}\,p_{\alpha_1} + p_{l_{01_y}}$
p_{e_8}	$p_{l_{I0_y}}$	$l_{01_z}\,p_{\alpha_1} + p_{l_{01_y}}$	p_{e_8}	$p_{l_{I0_z}}$	$p_{l_{01_z}} + p_{l_{12_z}}$
p_{e_9}	$p_{l_{I0_z}}$	$p_{l_{01_z}} + p_{l_{12_z}}$	p_{e_9}	$p_{l_{12_x}}$	0
$p_{e_{10}}$	$p_{l_{12_x}}$	0	$p_{e_{10}}$	$p_{l_{12_y}}$	$p_{l_{23_y}} + p_{l_{34_y}}$
$p_{e_{11}}$	$p_{l_{12_y}}$	$p_{l_{23_y}} + p_{l_{34_y}}$	$p_{e_{11}}$	$p_{l_{23_z}}$	0
$p_{e_{12}}$	$p_{l_{23_z}}$	0	$p_{e_{12}}$	$p_{l_{34_x}}$	$p_{l_{45_x}}$
$p_{e_{13}}$	$p_{l_{34_x}}$	$p_{l_{45_x}}$	$p_{e_{13}}$	$p_{l_{34_z}}$	0
$p_{e_{14}}$	$p_{l_{34_z}}$	0	$p_{e_{14}}$	$p_{l_{45_y}}$	$l_{56_x}\,p_{\gamma_5} + p_{l_{56_y}}$
$p_{e_{15}}$	$p_{l_{45_y}}$	$l_{56_x}\,p_{\alpha_6} + p_{l_{56_y}}$	$p_{e_{15}}$	$p_{l_{45_z}}$	$-l_{45_x}\,p_{\beta_4}$
$p_{e_{16}}$	$p_{l_{45_z}}$	$-l_{45_x}\,p_{\beta_4}$	$p_{e_{16}}$	$p_{l_{56_x}}$	$p_{l_{6E_z}}$
$p_{e_{17}}$	$p_{l_{56_x}}$	$p_{l_{6E_z}}$	$p_{e_{17}}$	$p_{l_{6E_x}}$	0
$p_{e_{18}}$	$p_{l_{56_z}}$	$l_{56_x}\,p_{\beta_6}$	$p_{e_{18}}$	$p_{l_{6E_y}}$	0
$p_{e_{19}}$	$p_{l_{6E_x}}$	0	$p_{e_{19}}$	p_{α_0}	p_{α_1}
$p_{e_{10}}$	$p_{l_{6E_y}}$	0	$p_{e_{20}}$	p_{β_0}	p_{β_1}
$p_{e_{21}}$	p_{α_0}	p_{α_1}	$p_{e_{21}}$	p_{α_2}	0
$p_{e_{22}}$	p_{β_0}	p_{β_1}	$p_{e_{22}}$	p_{γ_2}	0
$p_{e_{23}}$	p_{α_2}	0	$p_{e_{23}}$	p_{α_3}	0
$p_{e_{24}}$	p_{γ_2}	0	$p_{e_{24}}$	p_{γ_3}	0
$p_{e_{25}}$	p_{α_3}	0	$p_{e_{25}}$	p_{γ_4}	0
$p_{e_{26}}$	p_{γ_3}	0			
$p_{e_{27}}$	p_{γ_4}	0			
$p_{e_{28}}$	p_{γ_5}	$-p_{\alpha_6}$			
$p_{e_{29}}$	p_{β_E}	0			
$p_{e_{30}}$	p_{γ_E}	0			

2.4 Dynamik

Für den Entwurf der in Kapitel 5 beschriebenen Regler wird ein Modell für die Dynamik des Roboters benötigt, welches den Zusammenhang zwischen Motormomente und der daraus resultierenden Gelenkbeschleunigung beschreibt und somit das dynamische Verhalten des IRB1200 nachbildet. Für die Modellierung wird hier die Subsystem-Modellierung aus [36] mit dem in Abbildung 2.5 dargestellten Subsystem angewandt. Dieses besteht aus zwei Körper, einem Motor welcher über ein elastisches Getriebe mit einem Arm verbunden ist. Für den Entwurf der Regler würde grundsätzlich ein steifes Getriebe reichen, jedoch wird durch das Aufnehmen der Getriebeelastizität in die Modellierung, ein Fehler in der Positionierung erreicht und es eignet sich somit gut um die Robustheit des Reglers zu prüfen. Das gesamte System ergibt sich aus 6

Motor-Arm-Einheiten und einer Endmasse. Zur Modellierung der Getriebeelastizität ist es notwendig pro Einheit einen weiteren Freiheitsgrad q_M einzuführen, wodurch sich für das dynamische Modell die erweiterten Minimalkoordinaten

$$\mathbf{q}_{ext} = (\underbrace{q_1, \ldots, q_6}_{\mathbf{q}_A}, \underbrace{q_{M,1}, \ldots, q_{M,6}}_{\mathbf{q}_M})^T \tag{2.57}$$

ergeben.

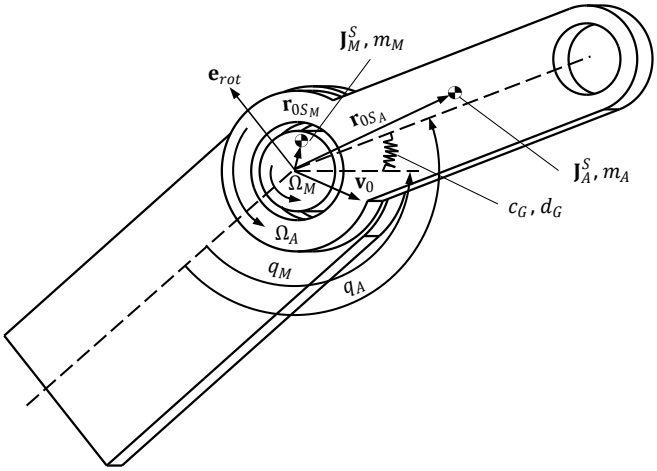


Abbildung 2.5: Motor-Arm Subsystem

Die Projektionsgleichungen für den IRB1200 sehen deshalb wie folgt aus

$$\sum_{n=1}^{7} \underbrace{\left(\frac{\partial \dot{\mathbf{y}}_n}{\partial \dot{\mathbf{q}}_{ext}}\right)^T}_{(2.62)} \underbrace{\left\{ \sum_{i \in \{A,M\}} \overbrace{\left[\left(\frac{\partial {}_R\mathbf{v}_{S_i}}{\partial \dot{\mathbf{y}}_n}\right)^T \quad \left(\frac{\partial {}_R\boldsymbol{\omega}_{S_i}}{\partial \dot{\mathbf{y}}_n}\right)^T \right]_i}^{(2.59)} \overbrace{\left[\begin{array}{c} {}_R\dot{\mathbf{p}} + {}_R\tilde{\boldsymbol{\omega}}_{IR}\, {}_R\mathbf{p} - {}_R\mathbf{f}^e \\ {}_R\dot{\mathbf{L}} + {}_R\tilde{\boldsymbol{\omega}}_{IR}\, {}_R\mathbf{L} - {}_R\boldsymbol{\tau}^e \end{array} \right]_i}^{(2.58)} \right\}}_{(2.60)} = \mathbf{0}$$

wobei durch eine Darstellung bezüglich der Bewegungsrichtung $\dot{\overline{\mathbf{y}}}_i = \left({}_R\mathbf{v}_{S_i}^T,\, {}_R\boldsymbol{\omega}_{S_i}^T\right)^T$, sich der Impuls- und Drallsatz vereinfacht darstellen lassen als

$$\overline{\mathbf{M}}_i\, \ddot{\overline{\mathbf{y}}}_i + \tilde{\boldsymbol{\Omega}}_R\, \overline{\mathbf{M}}_i\, \dot{\overline{\mathbf{y}}}_i + \dot{\overline{\mathbf{M}}}_i \dot{\overline{\mathbf{y}}}_i - \overline{\mathbf{Q}}_i \tag{2.58}$$

mit $\overline{\mathbf{M}}_i = \mathrm{diag}\left(m_i\, \mathbf{I},\, \mathbf{J}_i^S\right)$ und $\tilde{\boldsymbol{\Omega}}_R = \mathrm{diag}\left({}_R\tilde{\boldsymbol{\omega}}_{IR},\, {}_R\tilde{\boldsymbol{\omega}}_{IR}\right)$. Hierbei ist ${}_R\mathbf{v}_{S_i} = {}_R\mathbf{v}_0 + {}_R\tilde{\mathbf{r}}_{0S_i}^T\, {}_R\boldsymbol{\omega}_{IR} + {}_R\dot{\mathbf{r}}_{0S_i}$ die Schwerpunktsgeschwindigkeit und ${}_R\boldsymbol{\omega}_{S_i} = {}_R\boldsymbol{\omega}_F + {}_R\mathbf{e}_{rot}\, \Omega_i$ die Schwerpunktswinkelgeschwindigkeit. Die generalisierten Kräfte $\overline{\mathbf{Q}}_i$ werden erst in (2.60) eingeführt. Diese Bewegungsgleichungen der beiden Körper lassen sich mit

$$\overline{\mathbf{F}}_i^T = \left(\frac{\partial \dot{\overline{\mathbf{y}}}_i}{\partial \dot{\mathbf{y}}_n}\right)^T \tag{2.59}$$

in Richtung $\dot{\mathbf{y}}_n = \left({}_R\mathbf{v}_0^T,\ {}_R\boldsymbol{\omega}_F^T,\ \Omega_M,\ \Omega_A\right)^T$ projizieren und dort zusammenfügen, wodurch sich die Bewegungsgleichungen des Subsystems

$$\mathbf{M}_n\ddot{\mathbf{y}}_n + \mathbf{G}_n\dot{\mathbf{y}}_n - \mathbf{Q}_n \tag{2.60}$$

ergeben. $\dot{\mathbf{y}}_n$ setzt sich aus der Geschwindigkeit des Koppelpunktes ${}_R\mathbf{v}_0$, der Winkelgeschwindigkeit des Führungskoordinatensystems ${}_R\boldsymbol{\omega}_F$, und den relativen Motor- und Armgeschwindigkeiten $\Omega_M = i_G\,\dot{q}_M$, $\Omega_A = \dot{q}_A$ zusammen. Für das zusammengefügte System ergeben sich die generalisierten Kräfte zufolge des Motormoments u und der elastischen Kopplungen des elastischen Getriebes mit der Federsteifigkeit c_G, der Dämpfung d_G und der Getriebeübersetzung i_G zu

$$\mathbf{Q}_n = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ u + \frac{1}{i_G}\left(c_G\left(q_A - q_M\right) + d_G\left(\dot{q}_A - \dot{q}_M\right)\right) \\ -c_G\left(q_A - q_M\right) - d_G\left(\dot{q}_A - \dot{q}_M\right) \end{pmatrix}. \tag{2.61}$$

Die Gleichungen für die verschiedenen Motor-Arm-Einheiten lassen sich aus dem Subsystem ableiten, durch Spezifizierung der Koppelpunkte, Führungswinkelgeschwindigkeiten, Schwerpunktsabstände, der verschiedenen dynamischen Parameter sowie der Drehachse ${}_R\mathbf{e}_{rot}$. Eine Auflistung der dynamischen Parameter befindet sich im Anhang B.2. Das Referenzkoordinatensystem R ist, hierbei das jeweilige zur Einheit gehörige körperfeste Koordinatensystem aus Abbildung 2.1. Diese lassen sich mit

$$\mathbf{F}_n^T = \left(\frac{\partial\dot{\mathbf{y}}_n}{\partial\dot{\mathbf{q}}_{ext}}\right)^T \tag{2.62}$$

in den Raum der Minimalgeschwindigkeiten projizieren, wo sie zusammengefügt werden können. Zusätzlich werden hier auch noch generalisierte Reibungskräfte $\mathbf{Q}_f$ und Gewichtskräfte $\mathbf{Q}_g$ eingeführt. Die Reibung wird hierbei motorseitig als eine Kombination aus viskoser Reibung und Coulomb Reibung modelliert. Eine Zustandsumschaltung zwischen Gleiten und Haften findet nicht statt und die Haftreibung wird nicht mit modelliert. Die viskose Reibung lässt sich als Rayleigh Funktion

$$R_v = \sum_{n=1}^{6} \frac{1}{2}\, d_{v_n}\, \dot{q}_{M,n}^2 \tag{2.63}$$

angeben, wobei d_{v_n} den viskosen Reibungsparameter des jeweiligen Körpers darstellt.

Die Coulomb Reibung wird durch die projizierte Reibkraft

$$\mathbf{Q}_{f,c} = \begin{pmatrix} 0 \\ \vdots \\ Q_{f,c_1} \\ \vdots \\ Q_{f,c_6} \end{pmatrix}, \quad Q_{f,c_n} = d_{c_n} \tanh\left(\frac{\dot{q}_{M,n}}{\epsilon}\right), \tag{2.64}$$

mit dem coulomschen Reibungsparameter d_{c_n} modelliert. Zusammen ergeben sich die generalisierten Reibungskräfte zu

$$\mathbf{Q}_f = \frac{\partial R_v}{\partial \dot{\mathbf{q}}_{ext}} + \mathbf{Q}_{f,c}\,. \tag{2.65}$$

Eine Auflistung der verwendeten Reibungsparameter befindet sich in Tabelle B.2.

Die generalisierten Gewichtskräfte ergeben sich aus dem Potenzial

$$V_g = \sum_{n=1}^{7} -m_{A_n}\, {}_I\mathbf{g}\, {}_I\mathbf{r}_{S_{A,n}} - m_{M_n}\, {}_I\mathbf{g}\, {}_I\mathbf{r}_{S_{M,n}}\,, \tag{2.66}$$

wobei für jeden Körper die Masse des Arms m_{A_n} und des Motors m_{M_n}, sowie die zugehörigen Schwerpunktabstände ${}_I\mathbf{r}_{S_{A,n}}$ und ${}_I\mathbf{r}_{S_{M,n}}$ benötigt werden. Der Vektor ${}_I\mathbf{g}$ repräsentiert den Gravitationsvektor im Inertialsystem mit der Gravitationskonstanten g. Die generalisierten Gewichtskräfte ergeben sich durch

$$\mathbf{Q}_g = \frac{\partial V_g}{\partial \mathbf{q}_{ext}}\,. \tag{2.67}$$

Die aus $\mathbf{Q}_n$ resultierende generalisierten Kräfte können außerdem aufgeteilt werden in Kopplungskräfte des elastischen Getriebes $\mathbf{Q}_c$ und Antriebskräfte in der Form $\mathbf{B}\,\mathbf{u}$, wobei $\mathbf{u}$ die Motormomente der 6 Antriebsachsen und $\mathbf{B}$ die Stelleingriffsmatrix darstellt. Die gesamten Bewegungsgleichungen ergeben sich somit zu

$$\mathbf{M}(\mathbf{q}_{ext})\,\ddot{\mathbf{q}}_{ext} + \mathbf{G}(\mathbf{q}_{ext}, \dot{\mathbf{q}}_{ext})\,\dot{\mathbf{q}}_{ext} + \mathbf{Q}_g(\mathbf{q}_{ext}) + \mathbf{Q}_f(\dot{\mathbf{q}}_{ext}) + \mathbf{Q}_c(\mathbf{q}_{ext}, \dot{\mathbf{q}}_{ext}) = \mathbf{B}\,\mathbf{u}\,. \tag{2.68}$$

Daraus lässt sich für das dynamische Verhalten des IRB1200 das folgende nichtlineare Zustandsraummodel

$$\dot{\mathbf{x}} = \mathbf{f}\,(\mathbf{x},\, \mathbf{u}) = \begin{pmatrix} \dot{\mathbf{q}}_{ext} \\ \mathbf{M}^{-1}\left(\mathbf{B}\,\mathbf{u} - \mathbf{G}\,\dot{\mathbf{q}}_{ext} + \mathbf{Q}_g + \mathbf{Q}_f + \mathbf{Q}_c\right) \end{pmatrix}, \tag{2.69}$$

$$\mathbf{y} = \mathbf{g}\,(\mathbf{x}) = \mathbf{q}_{ext}\,, \tag{2.70}$$

mit dem Zustand

$$\mathbf{x} = \begin{pmatrix} \mathbf{q}_{ext} \\ \dot{\mathbf{q}}_{ext} \end{pmatrix} \tag{2.71}$$

ableiten.

Kapitel 3

Endeffektor Tracking

Zur präzisen Erfassung der Pose eines Industrieroboters existieren eine Vielzahl an Messsystemen. Die wichtigsten Systeme und ihre Eigenschaften lassen sich wie folgt zusammenfassen:

- **Drei-Kabel-System:** Dieses System fixiert drei Kabel am Endeffektor und verbindet sie mit Spannsystemen, die an festen Positionen im Arbeitsraum platziert sind. Die Spannsysteme sorgen für eine gleichmäßige Spannung der Kabel. Hochauflösende Längenmessgeräte messen die Kabellängen, sodass die Pose des Endeffektors rekonstruiert werden kann.
  Vorteile: Kostengünstig, relativ einfach in der Implementierung.
  Nachteile: Stark eingeschränkte Bewegungsfreiheit, aufwendiger wenig flexibler Aufbau, hoher Aufwand zur Kollisionsvermeidung und relativ begrenzte Genauigkeit ($\approx 0.1\,\text{mm}$).
- **Messarme:** Hierbei handelt es sich um Serienkinematiken, ähnlich zu Industrierobotern, ohne Antriebe, die jedoch mit hochpräzisen Winkelmesssystemen ausgestattet sind. Durch die fixe Verbindung mit dem Endeffektor des Roboters kann eine genaue Posenbestimmung erfolgen.
  Vorteile: Hohe Genauigkeit ($\approx 0.02\,\text{mm}$), kostengünstig.
  Nachteile: Starke Einschränkung in der Bewegungsfreiheit des Roboters, hoher Aufwand zur Kollisionsvermeidung und keine direkte Posenerfassung.

Abbildung 3.1: Messarm Faro Prime von Faro [13]

- **Externe Encoder:** Externe, hochpräzise Winkelgeber können die internen Winkelmessungen des Roboters ersetzen, um eine höhere Genauigkeit in der Posenerfassung zu erreichen, ähnlich der Funktionsweise der Messarme nur mit direkter Integration in den Industrierobotern.
  Vorteile: Bietet eine verbesserte Genauigkeit ($\approx 0.03\,\text{mm}$) ohne Einschränkungen der Roboterbewegungen. Direkte Integration in den Industrieroboter.
  Nachteile: Aufwendig und kostenintensiv im Einbau, indirekte Erfassung der Pose über die Gelenkwinkel.

Abbildung 3.2: Einsatz von externen Encodern im Accubot von Fill Gesellschaft m.b.H. [29]

- **Lasertracker:** Der Lasertracker kombiniert Winkelmessung und interferometrische Laserdistanzmessung, um die Position von Reflektoren im 3D-Raum zu bestimmen. Die Reflektoren müssen jedoch stets korrekt zum Tracker orientiert sein.
  Vorteile: Berührungslose Messung, hohe Flexibilität und Präzision ($\approx 0.03\,\text{mm}$).
  Nachteile: Sehr kostenintensiv, einschränkend in verwendbaren Roboterkonfigurationen, da Reflektoren korrekt ausgerichtet sein müssen.

Abbildung 3.3: Lasertracker Leica AT960 der Firma Hexagon AB [1]

- **3D Creator:** Dieses System verwendet drei Zeilenkameras im Infrarotbereich zur Erfassung eines Detektionskörpers mit aktiven Infrarotmarkern. Aus den Bildern der Zeilenkameras wird auf die Pose des Detektionskörpers rückgeschlossen.
  Vorteile: Kostengünstig, flexibel und berührungslos, für viele Anwendungsfälle geeignet.
  Nachteile: Geringe Robustheit gegenüber Verdeckung der Marker und ein

eingeschränkter Messbereich mit einer maximalen Genauigkeit von $\approx 0.1\,\text{mm}$, abhängig von der Entfernung zur Sensoreinheit.

Abbildung 3.4: 3D Creator der Firma Boulder Innovation Group [18]

In dieser Arbeit wird die Erfassung der Endeffektor-Pose mittels eines Kamerasystems bestehend aus Flächensensoren behandelt, was den Vorteil hoher Flexibilität bei geringem Preis bietet. Die Funktionsweise ist ähnlich zu der des 3D Creators mit dem großen Vorteil einer höheren Genauigkeit, einem größeren Messbereich und einer höheren Robustheit gegen Verdeckung. Wie die Funktionsweise eines solchen Systems aussehen könnte wird in den folgenden Abschnitten erläutert.

3.1 Kamera- und Optik

Beim kamerabasierten Endeffektor Tracking bildet die Digitalkamera, die Grundlage für die präzise Erfassung der Bewegung und Position von Objekten. Sie projiziert die einfallenden Lichtstrahlen auf einen Sensor und erzeugt so ein zweidimensionales Abbild des dreidimensionalen Raums. Wenn an 2 oder mehr verschiedenen Punkten im Raum zur gleichen Zeit durch Kameras Fotos aufgenommen werden, kann dadurch die Lage eines Punktes im dreidimensionale Raum rekonstruiert werden und falls eindeutig 3 oder mehr Punkte eines freibeweglichen Körper in diesem Raum erfasst werden können, so ist es möglich auf die Position und Lage von diesem Körper im Raum zu schließen. Die hierzu notwendigen Schritte auf die in den folgenden Abschnitten eingegangen wird, sind in Abbildung 3.5 zusammengefasst. Für diese benötigt es aber zunächst ein Modell der Kamera, auf welches in Abschnitt 3.1.3 eingegangen wird. Dieses Modell umfasst Parameter, auf deren Identifizierung wird in Abschnitt 3.1.7 eingegangen.

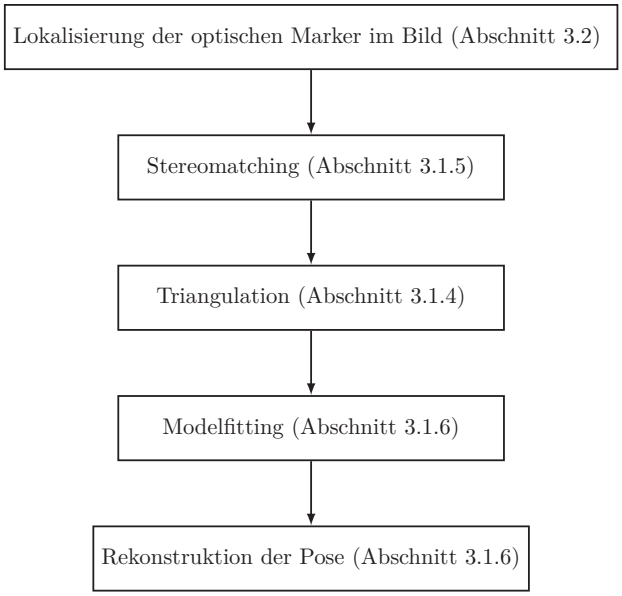


Abbildung 3.5: Ablauf der Erfassung der Pose eines Körpers

3.1.1 Bestandteile einer Digitalkamera

Um genaue Positionsdaten zu gewinnen, ist es wichtig, die einzelnen Bestandteile der Kamera und die damit verbundenen Parameter optimal abzustimmen und an den Arbeitsbereich anzupassen. Bei den Hauptbestandteilen handelt es sich hierbei um den Sensor, das Objektiv, die Blende, den Verschluss und die Filterlinse.

Der Sensor ist zuständig das Licht zu detektieren, um ein digitales Bild zu erzeugen. Hier gibt es unterschiedliche Sensortypen und Sensorgrößen die sich in ihrem Bildrauschen, Farbempfindlichkeit, Detektionsgeschwindigkeit und detektierbarem Spektrum unterscheiden. Die meisten Sensoren decken hier ein Spektrum ab das über den sichtbaren Bereich hinaus geht. Aus diesem Grund ist es möglich die Endeffektor Detektion im nahen Infrarotbereich (NIR) durchzuführen, was so einige Vorteile bietet. Es muss jedoch darauf geachtet werden, dass der gewählte Sensor eine ausreichende Sensitivität in diesem Spektrum aufweist.

Das Objektiv bündelt das Licht auf den Sensor und sorgt dafür, dass die Lichtstrahlen eines Objekts in einem bestimmten Abstand korrekt auf dem Sensor zusammenlaufen, wodurch eine scharfe Darstellung erreicht wird. Weiters lässt sich dadurch die Brennweite f einstellen, der Abstand zwischen der Linsenebene und dem Brennpunkt. Eine kurze Brennweite verursacht einen großen Öffnungswinkel und es kann somit ein größerer Bereich abgebildet werden, allerdings zu Lasten einer höheren perspektivischen Verzerrung. Eine große Brennweite bewirkt hingegen einen kleinen Öffnungswinkel, womit Objekte aus der Ferne vergrößert abgebildet werden. Hier gilt es einen Kompromiss zwischen einem breiten Sichtfeld und ausreichender Detailgenauigkeit zu finden.

Durch die Blende wird der Lichtdurchlass geregelt, wodurch die Schärfentiefe beeinflusst werden kann. Die Schärfentiefe stellt sicher, dass sich Objekte in einem weiten Bereich klar erfassen lassen. Die Berechnung ist in Abschnitt 3.1.2 ausgeführt. Eine kleine

Blende führt zu mehr Schärfe, hingegen eine große Blende zu mehr Lichteinfall. Die Öffnung der Blende wird über die Blendenzahl k' beschrieben und gibt das Verhältnis zwischen Brennweite und Durchmesser der Eingangspupille an. Für die Endeffektor Erkennung ist hier darauf zu achten, dass die Schärfentiefe möglichst groß ist, jedoch Über- und Unterbelichtung vermieden wird.

Der Verschluss steuert die Belichtungsdauer des Sensors und dadurch zusätzlich zur Blende die Belichtung des Bildes. Neben dem Sensor bestimmt sie auch wie schnelle Bildraten erzeugt werden können. Ist die Belichtungsdauer zu groß führt dies außerdem dazu, dass bewegte Objekte, abhängig von ihrer Geschwindigkeit verschwommen dargestellt werden. Beim Endeffektor Tracking ist eine möglichst kleine Belichtungsdauer von Vorteil, jedoch muss auch eine Unterbelichtung vermieden werden.

Durch optische Filter kann gesteuert werden, welches Spektrum detektiert werden soll. Bei der Anwendung im NIR-Bereich ist hier eine möglichst gute Dämpfung des sichtbaren Bereichs und ein möglichst guter Durchlass im NIR-Bereich notwendig.

3.1.2 Schärfentiefe

Die Schärfentiefe gibt den Bereich zwischen dem Nahpunkt und dem Fernpunkt an, dargestellt in Abbildung 3.6. Der Abstand des Nahpunktes zur Bildebene lässt sich näherungsweise durch

$$d_n = \frac{g\, d_h}{d_h + g - f} \tag{3.1}$$

berechnen, mit der Gegenstandsweite g, jener Entfernung von der Linsenebene zur fokussierten Ebene und der Hyperfokaldistanz

$$d_h = \frac{f^2}{k'\, Z} + f \tag{3.2}$$

in welche der Durchmesser des Zerstreuungskreises Z mit einfließt. Für den Fall, dass eine Unschärfe kleiner als ein Pixel auftritt ist Z die Sensorgröße. Der Fernpunkt ergibt sich näherungsweise zu

$$d_f = \frac{g\, d_h}{d_h - g - f}\,, \tag{3.3}$$

für $d_h > g - f$ und ansonsten liegt er im Unendlichen. Liegt ein Objekt innerhalb dieser Abstände so wird es scharf dargestellt.

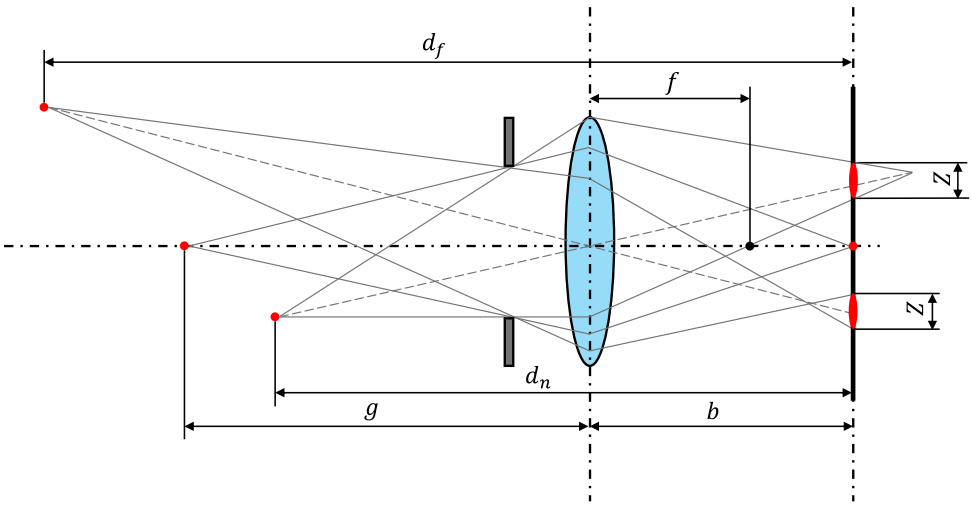


Abbildung 3.6: Schärfentiefe anhand eines einfachen Kameramodells bestehend aus einer Linse, einer Blende und des Sensors.

3.1.3 Kameramodell

Um die in den Bildern erfasste Position des Endeffektors zuverlässig bestimmen zu können, ist ein Kameramodell erforderlich. Dieses wird in [17] und [41] ausführlich beschrieben und wird im Folgenden kurz zusammengefasst.

Ein Punkt M im Raum der von der Kamera erfasst wird lässt sich über die Zentralprojektion in die Bildebene projizieren und erscheint dort als Punkt m, dargestellt in Abbildung 3.7. Die Bildebene liegt hierbei per Definition im Abstand f (die Brennweite) vor dem optischen Zentrum C und ist parallel zur Linsenebene wobei die z-Achse normal vom optischen Zentrum der Kamera zur Bildebene zeigt.

Die Beschreibung erfolgt laut Literatur in homogenen Koordinaten, diese werden in Folge durch $\hat{\mathbf{r}}$ gekennzeichnet. Es ergibt sich somit die Projektion

$$ {}_F\hat{\mathbf{r}}_{cm} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \frac{1}{z_{CM}} \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathbf{P}_0} {}_C\hat{\mathbf{r}}_{CM} \tag{3.4} $$

vom räumlichen Kamera-Koordinatensystem C in die Bildebene mit dem Bildkoordinatensystem F von einem Vektor vom optischen Zentrum C zu einem erfassten Punkt M.

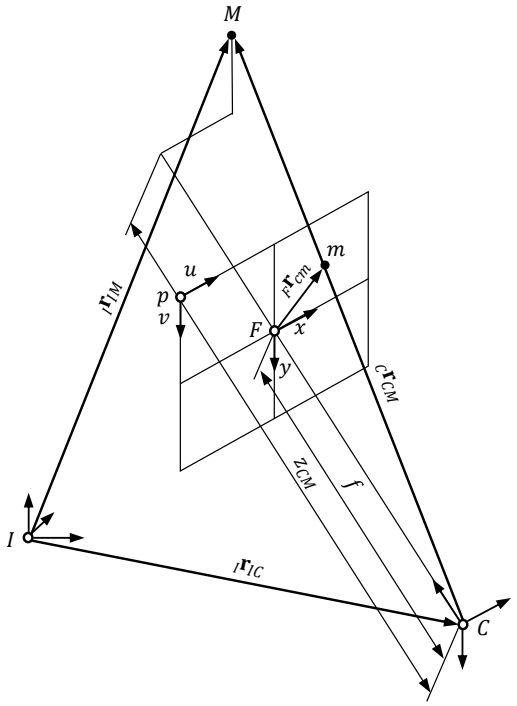


Abbildung 3.7: Projektion eines Punktes im Raum auf die Bildebene via Zentralprojektion

Die Detektion eines Punktes erfolgt in der Digitalkamera jedoch diskret in Pixel mit einer Auflösung ρ_u in x und ρ_v in y. Außerdem liegt der Ausgangspunkt der Pixel in der Regel nicht im optischen Zentrum sondern in der linken oberen Bildecke wodurch eine Verschiebung um u_0 und v_0 notwendig ist. Um dies zu berücksichtigen, ist die weitere Transformation

$$ {}_p\hat{\mathbf{r}}_m = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\rho_u} & 0 & u_0 \\ 0 & \frac{1}{\rho_v} & v_0 \\ 0 & 0 & 1 \end{pmatrix} {}_F\hat{\mathbf{r}}_{cm} \Rightarrow {}_p\hat{\mathbf{r}}_m = \frac{1}{z_{CM}} \underbrace{\begin{pmatrix} \frac{f}{\rho_u} & 0 & u_0 \\ 0 & \frac{f}{\rho_v} & v_0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{K}_C} \mathbf{P}_0 \, {}_C\hat{\mathbf{r}}_{CM} \tag{3.5} $$

notwendig. Zusätzlich ist eine Darstellung in einem Inertialsystem von Vorteil was durch die Transformation

$$ {}_p\hat{\mathbf{r}}_m = \frac{1}{z_{CM}} \mathbf{K}_C \mathbf{P}_0 \begin{pmatrix} \mathbf{R}_{CI} & -\mathbf{R}_{CI} \, {}_I\mathbf{r}_{IC} \\ \mathbf{0}^T & 1 \end{pmatrix} {}_I\hat{\mathbf{r}}_{IM} = \underbrace{\mathbf{K}_C \mathbf{P}_0 \, \mathbf{T}_{IC}^{-1}}_{\mathbf{P}} \, {}_I\hat{\mathbf{r}}_{IM} \tag{3.6} $$

erzielt wird, wobei $\mathbf{P}$ als Kamera-Projektionsmatrix bezeichnet wird und aus der Kamera-Kalibrierungsmatrix $\mathbf{K}_C$ mit den intrinsischen Kamera-Parametern und der Transformation $\mathbf{T}_{IC}$ mit den extrinsischen Kamera-Parametern besteht. In homogenen Koordinaten kann die Skalierung $\frac{1}{z_{CM}}$ weggelassen werden, da alle Punkte die sich in der letzten Koordinate auf 1 normieren lassen auf einer Gerade liegen und sich nicht unterscheiden lassen.

Zusätzlich weisen die Linsen in der Optik in der Regel sowohl radiale Verzerrung $\mathbf{f}_{dis,r}$ als auch tangentiale Verzerrungen $\mathbf{f}_{dis,t}$ auf. Diese lassen sich durch folgendes nichtlineares Modell beschreiben

$$ {}_{\bar{p}}\hat{\mathbf{r}}_m = \begin{pmatrix} u_{dis} \\ v_{dis} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta u \\ 0 & 1 & \Delta v \\ 0 & 0 & 1 \end{pmatrix} {}_{p}\hat{\mathbf{r}}_m = \mathbf{f}_{dis}\left({}_{p}\hat{\mathbf{r}}_m\right) \tag{3.7} $$

mit

$$ \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix} = \underbrace{\begin{pmatrix} u\left(k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ v\left(k_1 r^2 + k_2 r^4 + k_3 r^6\right) \end{pmatrix}}_{\mathbf{f}_{dis,r}} + \underbrace{\begin{pmatrix} 2p_1 uv + p_2\left(r^2 + 2u^2\right) \\ 2p_2 uv + p_1\left(r^2 + 2v^2\right) \end{pmatrix}}_{\mathbf{f}_{dis,t}} , \tag{3.8} $$

wobei $r^2 = u^2 + v^2$ gilt. Die Berechnung der Umkehrfunktion $\mathbf{f}_{dis}^{-1}$ erfolgt hierbei nach [47] numerisch mit der Iterationsvorschrift

$$ {}_{p}\hat{\mathbf{r}}_m^{(i+1)} = \frac{1}{1 + k_1\left(r^2\right)^{(i)} + k_2\left(r^4\right)^{(i)} + k_3\left(r^6\right)^{(i)}} \left({}_{p}\hat{\mathbf{r}}_m^{(i)} - \mathbf{f}_{dis,t}\left({}_{p}\hat{\mathbf{r}}_m^{(i)}\right)\right) , \tag{3.9} $$

wobei für den Startwert ${}_{p}\hat{\mathbf{r}}_m^{(0)} = {}_{\bar{p}}\hat{\mathbf{r}}_m$ gilt.

3.1.4 Triangulation

Durch die Reduktion der Dimension ist es mit nur einer Kamera nicht möglich die Position eines Punktes im Raum eindeutig zu rekonstruieren, hierfür benötigt es mindestens 2 Kameras, oder es müssen weitere Beziehungen bekannt sein. Bei zwei oder mehr Kameras lässt sich nach [11] und [32] die Position über die sogenannte Triangulation wie folgt bestimmen.

Ein Punkt M der im Abstand ${}_{I}\hat{\mathbf{r}}_{IM}$ vom Inertialsystem entfernt ist, wird in der Kamera als Punkt m erfasst, woraus sich der Abstand ${}_{p}\hat{\mathbf{r}}_m$ im entzerrten Bild ergibt. Die Entzerrung des Bildes erfolgt hierbei nach (3.9). Der Zusammenhang zwischen den zwei Vektoren ergibt sich nach (3.6) über die Kamera-Projektionsmatrix $\mathbf{P}$. Dem zufolge muss gelten

$$ {}_{p}\hat{\mathbf{r}}_m \times \mathbf{P}\, {}_{I}\hat{\mathbf{r}}_{IM} = {}_{\bar{p}}\tilde{\hat{\mathbf{r}}}_m \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} {}_{I}\hat{\mathbf{r}}_{IM} = \mathbf{0} , \tag{3.10} $$

da das Kreuzprodukt eines Vektors mit sich selbst einen Nullvektor[1] ergibt. Bei $\mathbf{p}_i \in \mathbb{R}^{1\times 4}$ handelt es sich um die Einträger der $i$ten Zeile der Projektionsmatrix $\mathbf{P}$. Aus diesem Zusammenhang lassen sich 2 unabhängige Gleichungen ableiten, die zu

$$ \underbrace{\begin{pmatrix} v\,\mathbf{p}_3 - \mathbf{p}_2 \\ u\,\mathbf{p}_3 - \mathbf{p}_1 \end{pmatrix}}_{\mathbf{A}_i \in \mathbb{R}^{2\times 4}} {}_{I}\hat{\mathbf{r}}_{IM} = \mathbf{0} \tag{3.11} $$

[1] In der Regel ist dies nie exakt der Fall da aufgrund von Diskretisierung und Messungenauigkeiten sich ein gewisser Fehler einstellt.

zusammengefasst werden können. Selbiges gilt für weitere Kameras mit anderen Kamera-Projektionsmatrizen und anderen Projektionen des Punktes M. Dies führt zum Gleichungssystem

$$\underbrace{\begin{pmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{pmatrix}}_{\mathbf{A}\in\mathbb{R}^{2n\times 4}} {}_I\hat{\mathbf{r}}_{IM} = \mathbf{0}\,, \tag{3.12}$$

welches entweder eindeutig bestimmt ist falls 2 Kameras im Einsatz sind, oder überbestimmt ist bei mehr als zwei Kameras. Prinzipiell lässt sich dieses Gleichungssystem durch eine Singulärwertzerlegung lösen, wobei die Lösung dem normierten Singulärvektor des kleinsten Singulärwertes entspricht. Diese Methode ist zwar sehr schnell, aber laut [32] nicht sehr robust und unpräzise, weshalb ein iteratives Verfahren empfohlen wird.

Bei diesem Verfahren wird der Einfluss der einzelnen Bildpunkte auf die Rekonstruktion der 3D-Pose gewichtet und die Gewichte werden iterativ bestimmt. Zusätzlich wird von homogenen Koordinaten auf die regulären Koordinaten gewechselt, wodurch sich das Gleichungssystem

$$\mathbf{A}\begin{pmatrix} {}_I\mathbf{r}_{IM} \\ 1 \end{pmatrix} = \bar{\mathbf{A}}\,{}_I\mathbf{r}_{IM} + \bar{\mathbf{b}} = \mathbf{0} \tag{3.13}$$

ergibt. Daraus lässt sich das Optimierungsproblem

$${}_I\mathbf{r}^*_{IM} = \arg\left(\min_{{}_I\mathbf{r}_{IM}}\left(\frac{1}{2}\left(\bar{\mathbf{A}}\,{}_I\mathbf{r}_{IM}+\bar{\mathbf{b}}\right)^T\mathbf{W}\left(\bar{\mathbf{A}}\,{}_I\mathbf{r}_{IM}+\bar{\mathbf{b}}\right)\right)\right)$$

mit der Gewichtungsmatrix $\mathbf{W} = \mathrm{diag}\,(w_1,\, w_1,\, \ldots,\, w_n,\, w_n)$ und den Gewichten w_i formulieren. Die Gewichte werden iterativ nach

$${w_i}^{(j)} = \left(\mathbf{p}_3^i\,{}_I\hat{\mathbf{r}}_{IM}^{(j-1)}\right)^{-2} \tag{3.14}$$

aus dem Ergebnis der letzten Iteration unter der Verwendung der 3ten Zeile der Kamera-Projektionsmatrix der jeweiligen Kamera i bestimmt wobei das Startgewicht ${w_i}^{(0)} = 1$ gewählt wird.

Über die Optimalitätsbedingung

$$\nabla\left(\frac{1}{2}\left(\bar{\mathbf{A}}\,{}_I\mathbf{r}_{IM}+\bar{\mathbf{b}}\right)^T\mathbf{W}\left(\bar{\mathbf{A}}\,{}_I\mathbf{r}_{IM}+\bar{\mathbf{b}}\right)\right) = \bar{\mathbf{A}}^T\mathbf{W}\,\bar{\mathbf{A}}\,{}_I\mathbf{r}_{IM} + \bar{\mathbf{A}}^T\mathbf{W}\,\bar{\mathbf{b}} = \mathbf{0} \tag{3.15}$$

ergibt sich die Iterationsvorschrift für das Minimum zu

$${}_I\mathbf{r}_{IM}^{(j)} = -\left(\bar{\mathbf{A}}^T\mathbf{W}^{(j)}\,\bar{\mathbf{A}}\right)^{-1}\bar{\mathbf{A}}^T\mathbf{W}^{(j)}\,\bar{\mathbf{b}} \tag{3.16}$$

mit der die 3D Position eines Punktes rekonstruiert werden kann. Zusammengefasst wird bei diesem Algorithmus zuerst ${}_I\mathbf{r}_{IM}^{(0)}$ mit $\mathbf{W}^{(0)} = \mathbf{I}$ nach (3.16) berechnet. Anschließend werden die neuen Gewichte ${w_i}^{(1)}$ nach (3.14) bestimmt, gefolgt von der

erneuten Berechnung von ${}_I\mathbf{r}_{IM}^{(1)}$ mit $\mathbf{W}^{(1)}$ nach (3.16). Dies wird solange wiederholt bis die Änderung von $\left\| {}_I\mathbf{r}_{IM}^{(j)} - {}_I\mathbf{r}_{IM}^{(j-1)} \right\|_2^2$ unter einem definierten Wert liegt.

3.1.5 Stereomatching

Werden mehrere Punkte im Raum erfasst so muss erkannt werden, welche Bildpunkte zu dem selben Punkt im Raum gehören, dies wird als Stereomatching bezeichnet. Hierzu gibt es verschiedene Möglichkeiten zum einen kann die Zuordnung durch eine mitgelieferte Kodierung der Marker erfolgen (näheres in Abschnitt 3.2), zum anderen ist es unter bestimmten Fällen möglich aufgrund der Lage in den Bildern automatisch eine Zuordnung zu treffen. Bei dieser Methode ist es notwendig die Lage eines projizierten Punktes im jeweilig anderen Kamerabild ermitteln zu können, was durch die Fundamentalmatrix ermöglicht wird. Vorgegangen wird hier nach [32] und [11].

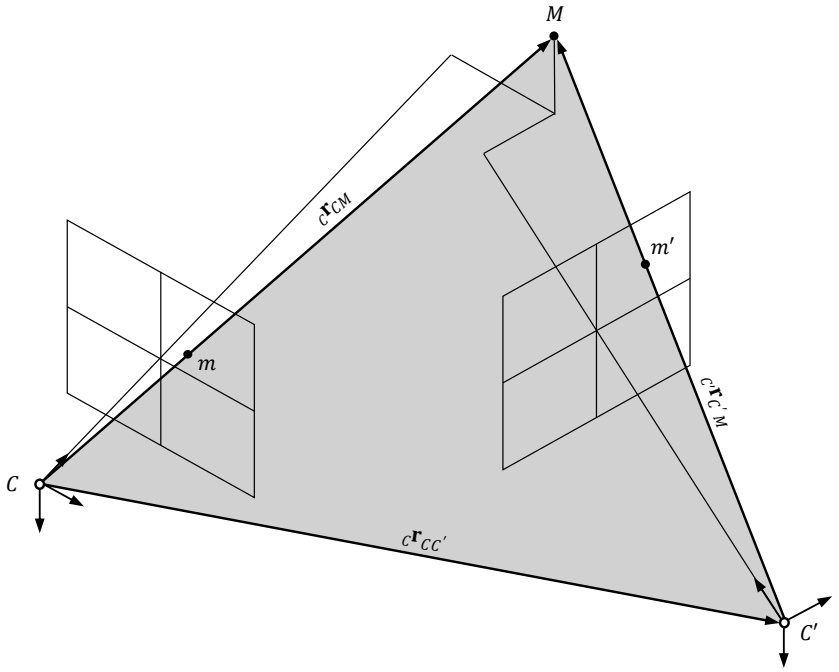


Abbildung 3.8: Darstellung eines Punktes im Raum in Bildern zweier Kameras

Die Fundamentalmatrix beschreibt den Zusammenhang zwischen den Projektionen eines Punktes M in verschiedenen Bildebenen. Die projizierten Punkte m und m' spannen hierbei mit dem betrachteten Punkt M die Epipolarebene auf, welche in Abbildung 3.8 grau hinterlegt ist. Für diese gilt

$$ {}_{C'}\mathbf{r}_{C'M}{}^T \left({}_{C'}\tilde{\mathbf{r}}_{C'C}\, \mathbf{R}_{C'C}\, {}_C\mathbf{r}_{CM} \right) = \mathbf{0}\,. \tag{3.17} $$

Außerdem ergibt sich aus (3.5) der Zusammenhang

$$ {}_p\hat{\mathbf{r}}_m = \mathbf{K}_C\, \mathbf{P}_0\, {}_C\hat{\mathbf{r}}_{CM} = \mathbf{K}_{CC}\mathbf{r}_{CM}\,. \tag{3.18} $$

Selbiges gilt auch für die zweite Kamera, wodurch sich die Beziehung

$$ {}_{p'}\hat{\mathbf{r}}_{m'}^{T} \underbrace{\mathbf{K}_{C'}^{-T}\, {}_{C'}\tilde{\mathbf{r}}_{C'C}\, \mathbf{R}_{C'C}\, \mathbf{K}_{C}^{-1}}_{\mathbf{F}} {}_{p}\hat{\mathbf{r}}_{m} = \mathbf{0} \tag{3.19} $$

ableiten lässt und somit die Fundamental Matrix $\mathbf{F}$ bestimmt werden kann. Für die umgekehrte Transformation gilt

$$ {}_{p}\hat{\mathbf{r}}_{m}^{T} \underbrace{\mathbf{K}_{C}^{-T}\, {}_{C}\tilde{\mathbf{r}}_{CC'}\, \mathbf{R}_{CC'}\, \mathbf{K}_{C'}^{-1}}_{\mathbf{F}^{T}} {}_{p'}\hat{\mathbf{r}}_{m'} = \mathbf{0}\,. \tag{3.20} $$

Die mögliche Lage eines Punktes im einem anderen Kamerabild ist auf eine Linie beschränkt. Diese Linie ergibt sich für den Bildpunkt m im Bild der anderen Kamera zu

$$ {}_{p'}\mathbf{l}' = \mathbf{F}\, {}_{p}\hat{\mathbf{r}}_{m} \tag{3.21} $$

bzw. umgekehrt zu

$$ {}_{p}\mathbf{l} = \mathbf{F}^{T}\, {}_{p'}\hat{\mathbf{r}}_{m'}\,, \tag{3.22} $$

wobei für einen Punkt auf dieser Linie gilt

$$ {}_{p}\mathbf{l}^{T}\, {}_{p}\hat{\mathbf{r}}_{l} = \mathbf{0}\,. \tag{3.23} $$

Durch die Diskretisierung des Bildes durch Pixel, sowie weitere Ungenauigkeiten liegt der Bildpunkt nicht exakt auf dieser Linie, jedoch in einem kleinen Bereich um diese Linie, wodurch eine klare Zuordnung möglich ist, sofern sich der Raumpunkt M nicht in der Verbindungsgerade der beiden optischen Zentren C und C' der Kameras befindet.

3.1.6 Optische Posenbestimmung eines Körpers

Bis jetzt wurde nur die Rekonstruktion einzelner Punkte im Raum betrachtet. Um die Position und Orientierung eines Körpers im Raum bestimmen zu können, bedarf es der Kenntnis von mindestens 3 Punkten[2] auf diesem Körper. Es muss sowohl die Lage der Punkte im körperfesten Koordinatensystem durch ein Modell dieses Körpers bekannt sein, zum Beispiel durch Vermessung des Körpers, als auch die rekonstruierte 3D Position der Punkte im Raum. Die Bestimmung der Pose erfordert zuerst eine korrekte Zuordnung der im Raum detektierten Punkte zu den bekannten Punkten des Modells des Körpers. Diese Aufgabe wird als Modelfitting bezeichnet und darauf wird in dieser Arbeit nicht genau eingegangen. Es gibt hierbei verschiedene Methoden zum einen kann durch eine Kodierung mitgegeben werden, um welchen Punkt es sich handelt, zum anderen kann wie in [32] beschrieben, falls die Punkte eine asymmetrische Anordnung aufweisen auf eine Zuordnung geschlossen werden.

[2] Diese Punkte dürfen nicht auf einer Gerade liegen, sondern müssen eine Ebene aufspannen.

Wenn einmal die Zuordnung der Punkte bestimmt wurde, kann daraus die Pose des Körpers ermittelt werden. Hier kann basierend auf [7], [23] und [32] wie folgt vorgegangen werden.

Die Position der N Marker im körperfesten Koordinatensystem ${}_K\mathbf{r}_{KM_i}$ ist durch die hochpräzise Vermessung des Detektionskörpers bekannt, sowie die Position der selben Marker im Inertialsystem ${}_I\mathbf{r}_{IM_i}$ durch die vorangegangene Analyse der Kamerabilder. Es gilt nun die Transformation zwischen den beiden Datensets zu finden, was auf das Optimierungsproblem

$$\mathbf{R}_{IK}{}^*,\, {}_I\mathbf{r}_{IK}{}^* = \arg\min_{\mathbf{R}_{IK},\, {}_I\mathbf{r}_{IK}} \left(\frac{1}{N} \sum_{i=1}^{N} ||\mathbf{R}_{IK}\, {}_K\mathbf{r}_{KM_i} + {}_I\mathbf{r}_{IK} - {}_I\mathbf{r}_{IM_i}||_2^2 \right) \tag{3.24}$$

führt, wobei $\mathbf{R}_{IK}$ eine Drehmatrix sein muss. Dieses Problem lässt sich mit der Schätzung für die Translation

$${}_I\mathbf{r}_{IK}{}^* \approx \underbrace{\frac{1}{N} \sum_{i=1}^{N} ({}_I\mathbf{r}_{IM_i})}_{{}_I\bar{\mathbf{r}}_{IM}} - \mathbf{R}_{IK} \underbrace{\frac{1}{N} \sum_{i=1}^{N} ({}_K\mathbf{r}_{KM_i})}_{{}_K\bar{\mathbf{r}}_{KM}} \tag{3.25}$$

und den Umformungsschritten

$$\begin{aligned}
\mathbf{R}_{IK}{}^* &\approx \arg\min_{\mathbf{R}_{IK}} \left(\frac{1}{N} \sum_{i=1}^{N} ||\mathbf{R}_{IK} \underbrace{({}_K\mathbf{r}_{KM_i} - {}_K\bar{\mathbf{r}}_{KM})}_{{}_K\bar{\bar{\mathbf{r}}}_{KM}} - \underbrace{({}_I\mathbf{r}_{IM_i} - {}_I\bar{\mathbf{r}}_{IM})}_{{}_I\bar{\bar{\mathbf{r}}}_{IM}} ||_2^2 \right) \\
&= \arg\min_{\mathbf{R}_{IK}} \left(\frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{R}_{IK}\, {}_K\bar{\bar{\mathbf{r}}}_{KM} - {}_I\bar{\bar{\mathbf{r}}}_{IM} \right)^T \left(\mathbf{R}_{IK}\, {}_K\bar{\bar{\mathbf{r}}}_{KM} - {}_I\bar{\bar{\mathbf{r}}}_{IM} \right) \right) \\
&= \arg\min_{\mathbf{R}_{IK}} \left(\frac{1}{N} \sum_{i=1}^{N} {}_K\bar{\bar{\mathbf{r}}}_{KM}{}^T\, {}_K\bar{\bar{\mathbf{r}}}_{KM} + {}_I\bar{\bar{\mathbf{r}}}_{IM}{}^T\, {}_I\bar{\bar{\mathbf{r}}}_{IM} - 2\, {}_I\bar{\bar{\mathbf{r}}}_{IM}{}^T\, \mathbf{R}_{IK}\, {}_K\bar{\bar{\mathbf{r}}}_{KM} \right) \\
&= \arg\max_{\mathbf{R}_{IK}} \left(\frac{1}{N} \sum_{i=1}^{N} {}_I\bar{\bar{\mathbf{r}}}_{IM}{}^T\, \mathbf{R}_{IK}\, {}_K\bar{\bar{\mathbf{r}}}_{KM} \right) = \arg\max_{\mathbf{R}_{IK}} \operatorname{tr} \left(\mathbf{R}_{IK}{}^T \underbrace{\frac{1}{N} \sum_{i=1}^{N} {}_I\bar{\bar{\mathbf{r}}}_{IM}{}^T\, {}_K\bar{\bar{\mathbf{r}}}_{KM}}_{\mathbf{C}} \right),
\end{aligned}$$

zu einem Maximierungsproblem vereinfachen, wobei hier $\mathbf{C}$ die Kovarianzmatrix darstellt. In weiterer Folge lässt sich durch die Singulärwertzerlegung $\mathbf{C} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^T$ das Maximierungsproblem lösen. Hier sind $\mathbf{U}$ und $\mathbf{V}$ orthogonale Matrizen und $\mathbf{W}$ ist eine Diagonalmatrix bestehend aus den Singulärwerten von $\mathbf{C}$. Für die Spur gilt

$$\operatorname{tr}\left(\mathbf{R}_{IK}{}^T\, \mathbf{C} \right) = \operatorname{tr}\left(\mathbf{R}_{IK}{}^T\, \mathbf{U}\,\mathbf{W}\,\mathbf{V}^T \right) = \operatorname{tr}\left(\underbrace{\mathbf{V}^T\, \mathbf{R}_{IK}{}^T\, \mathbf{U}}_{\mathbf{Q}}\, \mathbf{W} \right) = \operatorname{tr}\left(\mathbf{Q}\,\mathbf{W} \right), \tag{3.26}$$

mit der orthogonalen Matrix $\mathbf{Q}$. Diese kann nur maximal sein wenn $\mathbf{Q} = \mathbf{I}$ gilt und es ergibt sich somit die Lösung des Optimierungsproblem zu

$$\mathbf{R}_{IK}{}^{*} \approx \mathbf{U} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det\left(\mathbf{U}\,\mathbf{V}^{T}\right) \end{pmatrix} \mathbf{V}^{T} \tag{3.27}$$

und (3.25).

3.1.7 Kamerakalibrierung

Das Kamera-Modell umfasst Parameter wie die Brennweite, Verzerrungen durch die Linsen und die Position der Kamera im Raum. Diese Parameter müssen durch eine Kalibrierung ermittelt werden, um die Genauigkeit der Lokalisierung zu gewährleisten. Dazu wird nach [32] und [11] vorgegangen. Für die Kalibrierung muss die Lage mehrerer Punkte M_i eines Kalibrierungskörpers zueinander eindeutig bekannt sein, wobei dies in der Matrix

$${}_{K}\mathbf{K} = \begin{pmatrix} {}_{K}\hat{\mathbf{r}}_{KM_1} & {}_{K}\hat{\mathbf{r}}_{KM_2} & \dots & {}_{K}\hat{\mathbf{r}}_{KM_n} \end{pmatrix} \tag{3.28}$$

zusammengefasst wird. Dieser Kalibrierungskörper liegt in einem Abstand ${}_{I}\mathbf{r}_{IK}$ zum Inertialsystem verschoben und ist um $\mathbf{R}_{IK}$ verdreht, wodurch sich die Transformation

$${}_{I}\hat{\mathbf{r}}_{IM_i} = \underbrace{\begin{pmatrix} \mathbf{R}_{IK} & {}_{I}\mathbf{r}_{IK} \\ \mathbf{0}^{T} & 1 \end{pmatrix}}_{\mathbf{T}_{IK}} {}_{K}\hat{\mathbf{r}}_{KM_i} \tag{3.29}$$

für einen Punkt auf diesem Körper ergibt. Nun gibt es zwei Möglichkeiten entweder ist die Transformation vom Inertialsystem zum Detektionskörper bekannt, oder es wird das Inertialsystem in diesen Punkt gelegt. Für den ersten Fall können die Referenzen direkt in das Inertialsystem überführt werden mit ${}_{I}\mathbf{K} = \mathbf{T}_{IK}\,{}_{K}\mathbf{K}$ und im zweiten Fall gilt $\mathbf{T}_{KC} = \mathbf{T}_{IC}$ und somit $\mathbf{T}_{IK} = \mathbf{I}$, ${}_{I}\mathbf{K} = {}_{K}\mathbf{K}$. Jeder diese Punkte muss sich eindeutig zu seinem Bildpunkt zuordnen lassen, wodurch auch

$${}_{\bar{p}}\mathbf{k} = \begin{pmatrix} {}_{\bar{p}}\hat{\mathbf{r}}_{m_1} & {}_{\bar{p}}\hat{\mathbf{r}}_{m_2} & \dots & {}_{\bar{p}}\hat{\mathbf{r}}_{m_n} \end{pmatrix} \tag{3.30}$$

bekannt ist.

In der Identifizierung der Kameraparameter wird zunächst von dem rein linearen Kameramodell ausgegangen. Dieses ergibt nach (3.5) für den Kalibrierungskörper

$${}_{\bar{p}}\mathbf{k} \approx \mathbf{P}\,{}_{I}\mathbf{K}\,, \tag{3.31}$$

was sich in das Gleichungssystem

$$\mathbf{A}\,\mathbf{p}_c = \begin{pmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_n \end{pmatrix} \mathbf{p}_c = \mathbf{0} \tag{3.32}$$

überführen lässt, mit

$$\mathbf{A}_i = \begin{pmatrix} {}_I\hat{\mathbf{r}}_{IM_i}^T & \mathbf{0}^T & -u_{m_i}\,{}_I\hat{\mathbf{r}}_{IM_i} \\ \mathbf{0}^T & {}_I\hat{\mathbf{r}}_{IM_i}^T & -v_{m_i}\,{}_I\hat{\mathbf{r}}_{IM_i} \end{pmatrix} \in \mathbb{R}^{12\times 2}\,. \tag{3.33}$$

Die Einträge der Kamera- Projektionsmatrix sind dabei

$$\bar{\mathbf{p}}_C = \begin{pmatrix} p_{11} & \dots & p_{14} & p_{21} & \dots & p_{34} \end{pmatrix}^T, \quad \mathbf{P} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix}. \tag{3.34}$$

Es werden nun jene Einträge der Kamera- Projektionsmatrix gesucht, die ein Fehlermaß minimieren, welches auf das quadratische Optimierungsproblem

$$\bar{\mathbf{p}}_C^* = \arg\left(\min_{\bar{\mathbf{p}}_C}\left(\frac{1}{2}||\mathbf{A}\,\bar{\mathbf{p}}_C||_2^2\right)\right) = \arg\left(\min_{\bar{\mathbf{p}}_C}\left(\frac{1}{2}\left(\mathbf{A}\,\bar{\mathbf{p}}_C\right)^T\left(\mathbf{A}\,\mathbf{p}_C\right)\right)\right) \tag{3.35}$$

führt, mit einem Minimum bei

$$\nabla\frac{1}{2}||\mathbf{A}\,\mathbf{p}_C||_2^2 = 0\,, \tag{3.36}$$

was weiter zu der Gleichung

$$\mathbf{A}^T\mathbf{A}\mathbf{p}_C = \mathbf{0} \tag{3.37}$$

führt.

Für $\bar{\mathbf{p}}_C \neq \mathbf{0}$ muss $\bar{\mathbf{p}}_C$ somit ein Eigenvektor beim Eigenwert 0 sein, welcher sich numerisch bestimmen lässt.

Dieser Wert dient nun als Ausgangspunkt für die Identifizierung des nichtlinearen Kameramodells. Der Parametervektor wird um die nichtlinearen Parameter erweitert zu

$$\mathbf{p}_C = \begin{pmatrix} \bar{\mathbf{p}}_C^T & k_1 & k_2 & k_3 & p_1 & p_2 \end{pmatrix}^T \tag{3.38}$$

und ein nichtlinearer Fehler

$$\boldsymbol{\epsilon}\left(\mathbf{p}_C\right) = \begin{pmatrix} {}_{\bar{p}}\hat{\mathbf{r}}_{m_1} - \mathbf{f}_{dis}\mathbf{P}\ \left({}_I\hat{\mathbf{r}}_{IM_1}\right) \\ \vdots \\ {}_{\bar{p}}\hat{\mathbf{r}}_{m_n} - \mathbf{f}_{dis}\mathbf{P}\ \left({}_I\hat{\mathbf{r}}_{IM_n}\right) \end{pmatrix} \tag{3.39}$$

wird eingeführt, was auf das Optimierungsproblem

$$\mathbf{p}_C^* = \arg\left(\min_{\mathbf{p}_C}\left(\frac{1}{2}||\boldsymbol{\epsilon}\left(\mathbf{p}_C\right)||_2^2\right)\right) \tag{3.40}$$

führt. Dieses kann numerisch z.B. durch das Gauß-Newton Verfahren auf welches genauer in Abschnitt 4.1 eingegangen wird gelöst werden.

Aus der so bestimmten Kamera-Projektionsmatrix $\mathbf{P}$ kann die Kamera-Kalibrierungsmatrix $\mathbf{K}_C$ und die Transformation $\mathbf{T}_{IC}$ ermittelt werden.

3.2 Marker

Zur Lokalisierung des Endeffektors im Kamerabild gibt es verschiedene Ansätze. Eine Möglichkeit besteht darin, den Endeffektor direkt durch Objektdetektion mithilfe von Convolutional Neural Networks (CNNs) zu identifizieren. Diese Methode ist jedoch sehr rechenintensiv, wenig robust gegenüber Störungen und bietet nicht die erforderliche Genauigkeit, um die Anforderungen an die Präzisionssteigerung in industriellen Anwendungen zu erfüllen. Eine praktikablere Alternative ist der Einsatz von Markern, die am Endeffektor angebracht werden und eine präzise Lokalisierung im Kamerabild ermöglichen.

Die Marker können entweder im sichtbaren Lichtbereich oder im nahen Infrarotbereich (NIR) erkannt werden. Der NIR-Bereich bietet den Vorteil, dass die Detektion weniger anfällig für Störungen durch Umgebungslicht ist, da die Infrarotstrahlung von den Markern klarer erfasst wird. Für die Erkennung im NIR-Bereich werden häufig spezielle NIR-Filter verwendet, die eine zuverlässige Identifikation ermöglichen. Eine Übersicht liefert [32].

Es gibt verschiedene Arten von Markern, die sich in ihrer Funktionsweise unterscheiden. Einen Überblick liefert hier die Quelle [11], nach welcher vorgegangen wird. Auf die Unterschiede der Marker wird im Folgenden kurz eingegangen.

Aktive Marker emittieren selbst Licht, entweder im sichtbaren Spektrum oder im NIR-Bereich, und können durch den Einsatz von speziellen Filteralgorithmen sehr effizient und genau im Kamerabild erkannt werden. Diese Methode ist zwar effektiv, erhöht jedoch die Komplexität des Aufbaus, da zusätzliche Lichtquellen erforderlich sind. Eine alternative Lösung sind retroreflektive Marker, die Licht von einer externen Quelle, beispielsweise einer Infrarot-LED, reflektieren. Diese Marker bieten eine hohe Erkennungsgenauigkeit, da das reflektierte Licht leicht von der Kamera erkannt werden kann, ohne dass die Marker eine eigene Stromversorgung benötigen.

Passive Marker sind eine weitere Möglichkeit, wobei die Marker spezielle Muster oder kodierte Formen aufweisen, die mithilfe von Bildverarbeitungsalgorithmen erkannt und interpretiert werden. Diese Methode erfordert keine externe Lichtquelle, ist jedoch anfälliger für Störungen durch ungünstige Lichtverhältnisse und setzt aufwendigere Algorithmen für eine genaue Positionsbestimmung voraus. Dies führt zu längeren Rechen- und Detektionszeiten, was die Echtzeitfähigkeit einschränken kann.

Um die Position und Orientierung der Marker eindeutig zu bestimmen, gibt es verschiedene Ansätze. Eine Möglichkeit besteht darin, die Marker mit spezifischen Mustern oder kodierten Signalen zu versehen, die von der Kamera erkannt werden können. Bei aktiven Markern kann dies durch gepulste Lichtsignale erfolgen wie in [45], während passive Marker auf gedruckte Muster oder Formen zurückgreifen (zusammengefasst in [24]). Beispiele für solche passive kodierte Marker sind in Abbildung 3.9 dargestellt. Eine andere Methode basiert auf der Anordnung der Marker: Indem sie asymmetrisch positioniert werden, kann ihre Identifikation vereinfacht und Verwechslungen können vermieden werden.

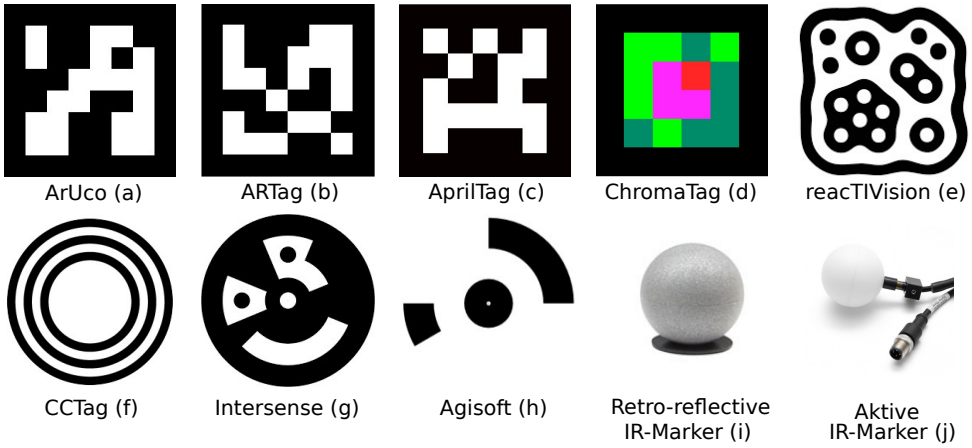


Abbildung 3.9: Auswahl diverser Marker für Computervision-Anwendungen: (a) ArUco Marker [40], (b) ARTag [14], (c) AprilTag [39], (d) ChromaTag [10], (e) reacTIVision Marker [22], (f) CCTag [9], (g) Intersense Marker [37], (h) Agisoft Marker [27], (i) passiver IR-Marker [4], (j) aktiver IR-Marker [3].

Für die präzise Bestimmung der Orientierung eines Objekts sind mindestens drei Marker erforderlich. In der Praxis werden jedoch oft mehr Marker verwendet, um die Messgenauigkeit weiter zu steigern und die Robustheit des Systems zu erhöhen. Dadurch bleibt die Positionsbestimmung auch dann zuverlässig, wenn einzelne Marker verdeckt sind oder eine fehlerhafte Erkennung vorliegt.

In dieser Arbeit wurde sich für kugelförmige infrarot retro-reflektive Marker entschieden. Diese weisen den Vorteil auf, dass sie sehr schnell im Bild erkannt werden können und die Erkennung sehr robust ist.

Für diesen Typ von Marker erfolgt die Segmentierung der Marker von der Umgebung durch einfaches Helligkeitsfiltern. Die Helligkeit der Pixeln der Bilder der verschiedenen Kameras werden mit einem Threshold-Wert verglichen und wenn sie diesen übersteigen, zählen sie zur Klasse der Marker ansonsten zur Klasse Umgebung. Der Threshold-Wert ist durch Versuche zu ermitteln, wobei sich dieser für verschiedene Kameras unterscheiden kann, je nach Positionierung dieser im Raum.

Nach der Segmentierung muss die Position des Markers ermittelt werden, was durch das Erkennen von Ellipsen in den segmentierten Bereichen erfolgt. Für dieses Problem existiert eine Vielzahl an Algorithmen. Eine Methode, die Segmentierung und Ellipsenerkennung in einem Schritt durchführt und somit sehr rechen-effizient ist, wurde in [32] beschrieben und wird hier kurz zusammengefasst.

Um die Position eines Markers im Bild zu bestimmen, wird der Schwerpunkt des Markers anhand der gewichteten Beiträge der zum Marker gehörenden Pixel berechnet. Bei kugelförmigen Markern entspricht der berechnete Schwerpunkt im Allgemeinen gut dem Zentrum des Markers, wobei die Helligkeit der Pixel zur Mitte hin zunimmt.

Sei $F(j,k)$ die Helligkeit eines Pixels an der Position (j,k). Dann lässt sich die Position des Markers in einem Bereich von j_0 bis J in u und von k_0 bis K in v wie folgt berechnen

$$ {}_p\hat{\mathbf{r}}_m = \begin{pmatrix} \frac{1}{K} \frac{\sum_{j=j_0}^{J} \sum_{k=k_0}^{K} j\, F(j,k)}{\sum_{j=j_0}^{J} \sum_{k=k_0}^{K} F(j,k)} \\ \frac{1}{J} \frac{\sum_{j=j_0}^{J} \sum_{k=k_0}^{K} k\, F(j,k)}{\sum_{j=j_0}^{J} \sum_{k=k_0}^{K} F(j,k)} \end{pmatrix} . \tag{3.41} $$

Kapitel 4

Geometrische Kalibrierung

Die in der Robotik für die Steuerung und Regelung verwendeten Modelle für die Kinematik, wie im Fall dieser Arbeit (2.9), sind mit geometrischen Fehlern behaftet. Grund hierfür liegt darin, dass die genaue Geometrie eines Roboters selten bekannt ist sondern meist nur die nominellen Parameter aus der Konstruktion. Es können jedoch umfassendere Modelle (wie in Abschnitt 2.3 beschrieben) erstellt werden, die geometrischen Fehler mit modellieren (2.50). Ziel der geometrischen Kalibrierung ist es nun diese Fehler zu ermitteln, damit das Verhalten des Fehlermodells möglichst genau dem des realen Roboters entspricht und so die Genauigkeit des Roboters verbessert werden kann.

Hierfür müssen zunächst mit einem Messsystem die tatsächliche Endeffektor-Pose $\mathbf{z}_{IE,m}$ und die Gelenkposition $\mathbf{q}_m$ an verschiedenen Punkten des Arbeitsraumes ermittelt werden. Die Wahl dieser Kalibrierungskonfiguration $\boldsymbol{\xi}$ hat dabei einen maßgeblichen Einfluss auf die spätere erzielbare Genauigkeit und wird in Abschnitt 4.2 behandelt. Über die Messungen werden die Fehlerparameter ermittelt, wobei die Abweichung der Endeffektorpose zum Fehlermodell (2.55) minimiert wird, siehe Abschnitt 4.1. Mit den identifizierten Fehlerparametern können die Fehler in der Steuerung und Regelung der Roboter kompensiert werden. Meist ist es jedoch nicht möglich die Fehlerparameter direkt der Roboter-Steuerung zu übergeben, da diese nur selten Zugriff auf die unterlagerten Modelle gewähren, wie es auch bei der verwendeten Steuerung ABB-IRC5 der Fall ist. Aus diesem Grund muss die der Roboter-Steuerung übergebene Bahn so angepasst werden, dass die kompensierte Soll-Bahn abgefahren wird, beschrieben in Abschnitt 4.4. Der Ablauf der geometrischen Kalibrierung lässt sich durch Abbildung 4.1 zusammenfassen.

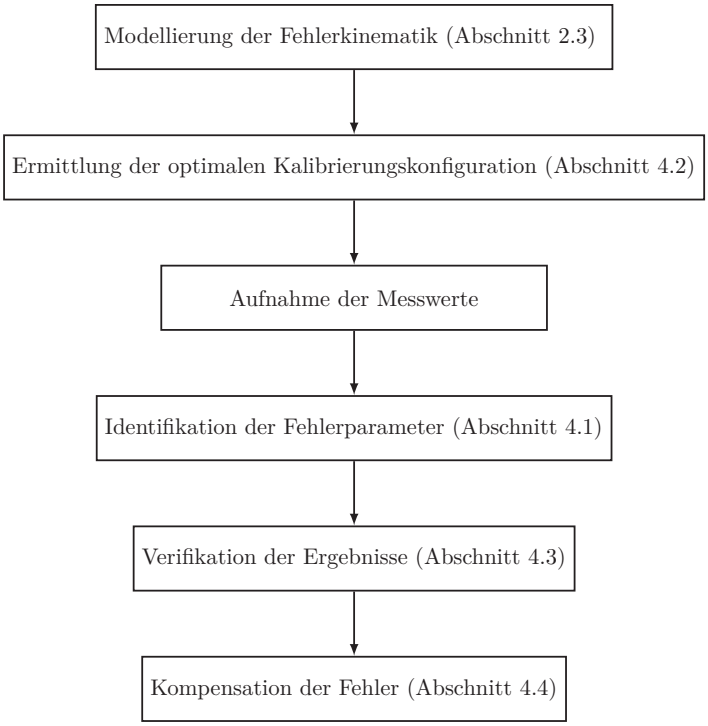


Abbildung 4.1: Ablauf geometrische Kalibrierung

4.1 Identifikation der Fehlerparameter

Für die Identifikation der Fehlerparameter von Abschnitt 2.3 wird nach [16] vorgegangen. Die Abweichung

$$\Delta \mathbf{z}_{IE} = \bar{\mathbf{e}}\left(\mathbf{z}_{IE,m}, \mathbf{z}_{IE}\left(\mathbf{q}_m, \mathbf{p}_e\right)\right) \tag{4.1}$$

einer gemessenen Pose zum Fehlermodell (2.50) liefert 6 nichtlineare Gleichungen die für die Identifikation herangezogen werden. Die Messungen sind jedoch mit Messfehlern behaftet. Um eine bessere Kalibrierung zu erreichen, die den Einfluss der Messfehler reduziert, wird ein überbestimmtes Gleichungssystem

$$\mathbf{Q}(\mathbf{p}_e) = \begin{pmatrix} \bar{\mathbf{e}}\left(\mathbf{z}_{IE,1}, \mathbf{z}_{IE}\left(\mathbf{q}_1, \mathbf{p}_e\right)\right) \\ \vdots \\ \bar{\mathbf{e}}\left(\mathbf{z}_{IE,N}, \mathbf{z}_{IE}\left(\mathbf{q}_N, \mathbf{p}_e\right)\right) \end{pmatrix} \in \mathbb{R}^{6N} \tag{4.2}$$

mit $N > 5$ Messungen formuliert. Die Fehlerparameter werden nun so variiert, dass ein definiertes Fehlermaß dieser Gleichungen minimiert wird. Hierfür kommen verschiedene Normen in Frage. In dieser Arbeit wird der quadratische Fehler verwendet, wodurch sich das nichtlineare freie Optimierungsproblem

$$\mathbf{p}_e^* = \arg\left(\min_{\mathbf{p}_e}\left(\frac{1}{2}||\mathbf{Q}(\mathbf{p}_e)||_2^2\right)\right) \tag{4.3}$$

ergibt.

Für die Lösung dieser Klasse an Problemen können verschiedene numerische Verfahren verwendet werden, wobei hier auf die verwendete Erweiterung des Gauß-Newton-Verfahren eingegangen wird, basierend auf [46] und [16].

Bei dieser Methode wird das nichtlineare Optimierungsproblem (4.3) mittels der Taylorreihenapproximation

$$\mathbf{Q}\left(\mathbf{p}_e\right) = \overbrace{\underbrace{\mathbf{Q}\left(\mathbf{p}_e^{(0)}\right)}_{\mathbf{Q}^{(0)}} + \underbrace{\nabla\mathbf{Q}\left(\mathbf{p}_e^{(0)}\right)}_{\mathbf{\Theta}^{(0)}} \underbrace{\left(\mathbf{p}_e - \mathbf{p}_e^{(0)}\right)}_{\Delta\mathbf{p}_e}}^{\bar{\mathbf{Q}}(\Delta\mathbf{p}_e)} + \mathcal{O}\left(||\mathbf{p}_e - \mathbf{p}_e^{(0)}||^2\right) \tag{4.4}$$

durch das quadratische Problem

$$\mathbf{p}_e^* \approx \arg\left(\min_{\Delta\mathbf{p}_e}\left(\frac{1}{2}||\bar{\mathbf{Q}}(\Delta\mathbf{p}_e)||_2^2\right)\right) = \arg\left(\min_{\Delta\mathbf{p}_e}\left(\frac{1}{2}\left(\mathbf{\Theta}^{(0)}\,\Delta\mathbf{p}_e + \mathbf{Q}^{(0)}\right)^T\left(\mathbf{\Theta}^{(0)}\,\Delta\mathbf{p}_e + \mathbf{Q}^{(0)}\right)\right)\right)$$

angenähert. Das Minimum des quadratischen Problems lässt sich wiederum mit der notwendigen Optimalitätsbedingung

$$\nabla\frac{1}{2}||\bar{\mathbf{Q}}(\Delta\mathbf{p}_e)||_2^2 = 0 \tag{4.5}$$

bestimmen, was zu

$$\left(\mathbf{\Theta}^{(0)}\right)^T\left(\mathbf{\Theta}^{(0)}\,\Delta\mathbf{p}_e + \mathbf{Q}^{(0)}\right) = 0 \tag{4.6}$$

führt. Daraus ergibt sich die explizite Lösung zu

$$\Delta\mathbf{p}_e = -\left(\left(\mathbf{\Theta}^{(0)}\right)^T\,\mathbf{\Theta}^{(0)}\right)^{-1}\left(\mathbf{\Theta}^{(0)}\right)^T\mathbf{Q}^{(0)}\,. \tag{4.7}$$

Hiermit lässt sich eine Iterationsvorschrift formulieren, die sich der optimalen Lösung $\mathbf{p}_e^*$ annähert. Um die numerische Lösung des Gleichungssystems zu erleichtern wird dieses normalisiert. Dies geschieht mit der Transformation $\mathbf{N} = \mathbf{\Theta}\mathbf{V} = \mathbf{I}\,\frac{1}{n_i}$, wobei n_i der größte Wert der $i$ten Spalte von $\mathbf{\Theta}$ ist. Somit folgt für (4.7)

$$\begin{aligned}
\Delta\mathbf{p}_e &= -\left(\left(\mathbf{N}\mathbf{V}^{-1}\right)^T\,\mathbf{N}\mathbf{V}^{-1}\right)^{-1}\left(\mathbf{N}\mathbf{V}^{-1}\right)^T\mathbf{Q}^{(0)}\\
&= -\left(\mathbf{V}^{-T}\mathbf{N}^T\,\mathbf{N}\mathbf{V}^{-1}\right)^{-1}\mathbf{V}^{-T}\mathbf{N}^T\mathbf{Q}^{(0)}\\
&= -\,\mathbf{V}\left(\mathbf{N}^T\,\mathbf{N}\right)^{-1}\mathbf{V}^T\mathbf{V}^{-T}\mathbf{N}^T\mathbf{Q}^{(0)}\\
&= -\,\mathbf{V}\left(\mathbf{V}^T\,\left(\mathbf{\Theta}^{(0)}\right)^T\,\mathbf{\Theta}^{(0)}\,\mathbf{V}\right)^{-1}\mathbf{V}^T\,\left(\mathbf{\Theta}^{(0)}\right)^T\,\mathbf{Q}^{(0)}\,.
\end{aligned}$$

Zusätzlich wird zur Verbesserung der numerischen Stabilität die Schrittweite $0 < d < 1$ eingeführt. Es ergibt sich somit aus (4.7) die Iterationsvorschrift zu

$$\mathbf{p}_e^{(i+1)} = \mathbf{p}_e^{(i)} - d\,\mathbf{V}\left(\mathbf{V}^T\left(\mathbf{\Theta}^{(i)}\right)^T\,\mathbf{\Theta}^{(i)}\,\mathbf{V}\right)^{-1}\mathbf{V}^T\left(\mathbf{\Theta}^{(i)}\right)^T\mathbf{Q}^{(i)}\,. \tag{4.8}$$

Da die Bestimmung der Orientierung des Endeffektors mit Fehlern behaftet ist, kann auch analog zur Verwendung der Abweichung der Pose (4.1), rein der Fehler in der Position

$$\Delta \mathbf{r}_{IE} = {}_I\mathbf{r}_{IE,m} - {}_I\mathbf{r}_{IE}\left(\mathbf{q}_m, \mathbf{p}_e\right) \tag{4.9}$$

für die Identifikation verwendet werden, wobei sich die Anzahl von identifizierbaren Fehlerparametern reduziert. Es entsteht somit ein Trade-off zwischen Verringerung der Modellfehler und Verringerung der Messfehler. In Abschnitt 6.4 werden die beiden Methoden verglichen.

4.2 Optimale Kalibrierungskonfigurationen

Für die in Abschnitt 4.1 beschriebene Methode zur Identifizierung der geometrischen Fehlerparameter wird eine Menge an Kalibrierungskonfigurationen $\boldsymbol{\xi} = \{\mathbf{q}_1, \ldots, \mathbf{q}_N\}$ benötigt. Grundsätzlich eignen sich hier nicht alle Konfigurationen gleich gut für die Kalibrierung. Außerdem ist die Größe der Menge auf eine gewisse Anzahl N_{max} an Kalibrierungskonfigurationen beschränkt, da eine höhere Anzahl zu mehr Zeitaufwand in der Messung, sowie Identifikation führen würde und mehr Speicherplatz benötigt. Ziel ist es, Konfigurationen zu finden, die eine möglichst präzise Kalibrierung ermöglichen, bei der sowohl Messrauschen als auch numerische Fehler minimiert werden. Dies erfolgt unter der Annahme, dass alle Konfigurationen im Arbeitsraum des Roboters messtechnisch gleichwertig erfasst werden können.

Um die Qualität der Konfigurationen $\boldsymbol{\xi}$ zu bewerten, wird die Informationsmatrix $\boldsymbol{\Lambda} = \left(\boldsymbol{\Theta}^0\right)^T \boldsymbol{\Theta}^0$ herangezogen, welche aus der Regressionsmatrix $\boldsymbol{\Theta}^0 = \boldsymbol{\Theta}^0\left(\boldsymbol{\xi}\right)$ gemäß Gleichung (4.4) gebildet wird. Als geeignete Metrik zur Bewertung wird gemäß [20] der kleinste nicht-null Singulärwert der Matrix $\boldsymbol{\Lambda}$ herangezogen, da dieser Auskunft über die Stabilität der Invertierung von $\boldsymbol{\Lambda}$ gibt und aufzeigt, wie gut der am schlechtesten identifizierbare Parameter beobachtbar ist, gleichbedeutend einer Abbildung dieses Parameters in den Nullraum. Dieser Singulärwert, der numerisch über die Singulärwertzerlegung berechnet wird, wird im Folgenden, in Anlehnung an [16], als Beobachtungsindex $B_3 = B_3\left(\boldsymbol{\Lambda}\right)$ bezeichnet. Ein größerer Wert von B_3 deutet auf eine höhere Eignung der Konfigurationen $\boldsymbol{\xi}$ für die Kalibrierung hin.

Zusätzlich muss definiert werden, welche Konfigurationen gültig sind. Diese werden durch die Gelenksgrenzen des Roboters, die zulässigen Bereiche des Messsystems sowie durch Kollisionsvermeidung bestimmt. Damit das Problem numerisch gelöst werden

kann, muss zusätzlich dieser Bereich diskretisiert werden. In dieser Arbeit wird deshalb der zulässige Bereich wie folgt definiert

$$\begin{aligned} q_1 &\in [-100, 100], \\ q_2 &\in [-45, 90], \\ q_3 &\in [-90, 30], \\ q_4 &\in [-180, 180], \\ q_5 &\in [-90, 90], \\ q_6 &\in [-180, 180]. \end{aligned}$$

Jede Achse wird äquidistant in $k = 13$ Samples unterteilt, was zur Menge $\mathbf{\Pi} = \{\mathbf{q}_1, \ldots, \mathbf{q}_\mathcal{N}\}$ mit der gesamten Anzahl an möglichen Konfigurationen von $\mathcal{N} = k^6 = 4\,826\,809$ führt.

Das Problem lässt sich somit als ein nichtlineares, beschränktes Optimierungsproblem

$$\begin{gathered} \boldsymbol{\xi}^* = \arg\left(\max_{\boldsymbol{\xi}} \left(B_3\left(\mathbf{\Lambda}\left(\boldsymbol{\xi}\right)\right)\right)\right) \\ \text{s.t. } \boldsymbol{\xi} \subset \mathbf{\Pi} \\ N \leq N_{max} \end{gathered}$$

formulieren und kann mit einer Erweiterung des DetMax-Algorithmus [33] gelöst werden. Kurz zusammengefasst handelt es sich hierbei um einen Brute-Force-Algorithmus der durch Auswechseln der Gelenkpositionen B_3 maximiert. Der Ablauf des DetMax-Algorithmus ist in Abbildung A.22 schematisch dargestellt.

Zuerst wird ein Startkonfigurationsset $\boldsymbol{\xi}_0$ der Größe $N_0 \leq N$ zufällig aus $\mathbf{\Pi}$ gewählt, für das die Regressionsmatrix $\mathbf{\Theta}_0^{(0)}$, die Informationsmatrix $\mathbf{\Lambda}_0$ und der Beobachtbarkeitsindex B_3 bestimmt werden. Anschließend werden in einem iterativen Schritt die Konfigurationen aus $\boldsymbol{\xi}_0$ durch solche aus $\mathbf{\Pi}$ ersetzt, die den Beobachtungsindex B_3 verbessern, bis keine weitere Steigerung möglich ist. Dabei wird in jeder Iteration die Konfiguration entfernt, deren Ausschluss den geringsten Abfall des Beobachtungsindex B_3 verursacht, und durch jene ersetzt, die die größte Steigerung bewirkt. Die Informationsmatrix $\mathbf{\Lambda}_i$ wird schrittweise aktualisiert, um den Rechenaufwand zu minimieren. Beim Hinzufügen wird hier zunächst die Regressionsmatrix $\mathbf{\Theta}_i^{(0)}$ um den neuen Anteil

$$\mathbf{\Theta}_{i+1}^{(0)} = \begin{pmatrix} \mathbf{\Theta}_i^{(0)} \\ \boldsymbol{\vartheta}_{i+1}^{(0)} \end{pmatrix} \quad \text{mit} \quad \boldsymbol{\vartheta}_{i+1}^{(0)} = \nabla_{\mathbf{p}_e} \bar{\mathbf{e}}\left(\mathbf{z}_{IE}\left(\mathbf{q}_j\right), \mathbf{z}_{IE}\left(\mathbf{q}_j, \mathbf{p}_e\right)\right)\big|_{\mathbf{p}_e = \mathbf{0}}$$

erweitert. Es ergibt sich somit folgende Iterationsvorschrift für die Informationsmatrix

$$\mathbf{\Lambda}_{i+1} = \left(\mathbf{\Theta}_{i+1}^{(0)}\right)^T \mathbf{\Theta}_{i+1}^{(0)} = \left(\mathbf{\Theta}_i^{(0)}\right)^T \mathbf{\Theta}_i^{(0)} + \left(\boldsymbol{\vartheta}_{i+1}^{(0)}\right)^T \boldsymbol{\vartheta}_{i+1}^{(0)} = \mathbf{\Lambda}_i + \Delta\mathbf{\Lambda}\,.$$

Analog folgt aus

$$\mathbf{\Lambda}_i = \left(\mathbf{\Theta}_i^{(0)}\right)^T \mathbf{\Theta}_i^{(0)} \quad \text{mit} \quad \mathbf{\Theta}_i^{(0)} = \begin{pmatrix} \mathbf{\Theta}_{i+1}^{(0)} \\ \boldsymbol{\vartheta}_i^{(0)} \end{pmatrix}$$

die Vorschrift fürs Entfernen von Konfigurationen zu

$$\mathbf{\Lambda}_{i+1} = \left(\mathbf{\Theta}_i^{(0)}\right)^T \mathbf{\Theta}_i^{(0)} - \left(\boldsymbol{\vartheta}_i^{(0)}\right)^T \boldsymbol{\vartheta}_i^{(0)} = \mathbf{\Lambda}_i - \Delta\mathbf{\Lambda}\,.$$

Der größte Abfall oder die größte Steigerung von B_3 wird durch einen Brute-Force-Ansatz ermittelt, bei dem die Auswirkungen aller Konfigurationen aus $\boldsymbol{\xi}_i$ bzw. $\mathbf{\Pi}$ auf B_3 berechnet werden. Nach dem Austauschschritt, folgt eine Erweiterung von $\boldsymbol{\xi}$ um die bestmöglichen Konfigurationen zu finden, bis entweder durch weiteres Hinzufügen von Konfigurationen B_3 nicht mehr gesteigert werden kann oder $\boldsymbol{\xi}$ die Größe von N_{max} erreicht. Anschließend wird das neue Konfigurationsset erneut einem Austauschschritt unterzogen. Sobald sich der Wert von B_3 nicht weiter verbessern lässt, hat der Algorithmus das Optimum gefunden.

Wie in [33] beschrieben, ist der Algorithmus jedoch anfällig dafür, lediglich ein lokales Maximum zu finden, was stark von der Wahl des Startkonfigurationssets abhängt. Aus diesem Grund wird der Algorithmus mehrfach durchlaufen, um das Ergebnis zu verifizieren.

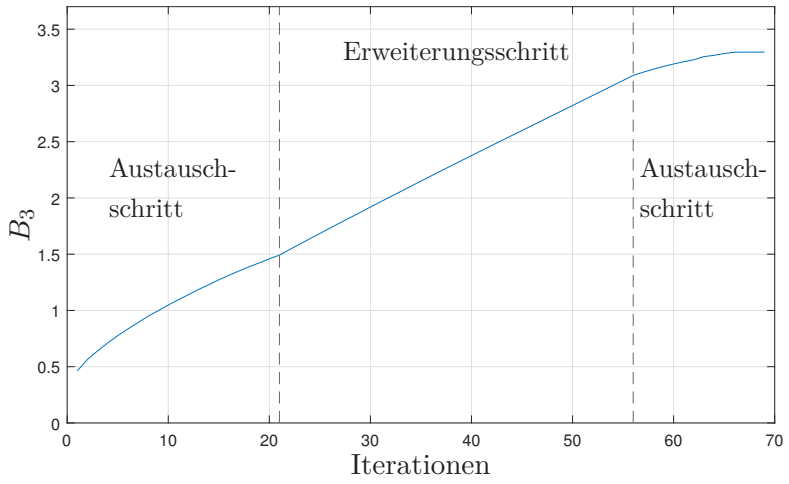


Abbildung 4.2: Verlauf von B_3

Für den IRB1200 werden in dem zuvor definierten zulässigen Bereich mit einer Diskretisierung von $k = 13$, $N = 75$ Konfigurationen gesucht, wobei mit $N_0 = 40$ gestartet wird. Der Verlauf von B_3 während der Optimierung ist in Abbildung 4.2 dargestellt. Es lässt erkennen, dass dabei der Beobachtbarkeitsindex von ca. 0.5 auf 3.3 gesteigert werden konnte, was mehr als dem 6-fachen entspricht. Des Weiteren zeigt der erste Austauschschritt und der Erweiterungsschritt die meiste Verbesserung. Durch die annähernd kontinuierliche Verbesserung von B_3 im Erweiterungsschritt lässt sich erkennen, dass die Grenze, an der das Hinzufügen einer neuen Konfiguration zu $\boldsymbol{\xi}$ keine Verbesserung bringt, noch weit entfernt ist und mit jeder weiteren Konfiguration

in dieser Phase eine Steigerung von B_3 um ca. 0.045 erzielt werden kann. Die gesamte Berechnung[1] ist jedoch mit ca. 26 h bereits sehr zeitaufwendig. Die Verteilung der Konfigurationen im Raum ist in Abbildung 4.3 dargestellt. Es lässt sich erkennen, dass sie sehr gut im Raum verteilt sind.

Für die ermittelten optimalen Konfigurationen muss anschließend ein Messprozess implementiert werden, hierzu werden die Konfigurationen noch einmal mittels Matlab auf Kollisionen überprüft und sortiert. Die Sortierung erfolgt nach minimalen Gelenkwinkelabstand wobei die ersten drei Achsen höher gewichtet werden. Anschließend wird ein ABB Rapid - Programm generiert das diese anfährt und bei jeder Konfiguration 1 s verweilt, um Schwingungen des Endeffektors ausklingen zu lassen und um die Konfiguration als solche identifizieren zu können. Die Planung der kollisionsfreien Trajektorien zwischen den Konfigurationen übernimmt hierbei die ABB Steuerung.

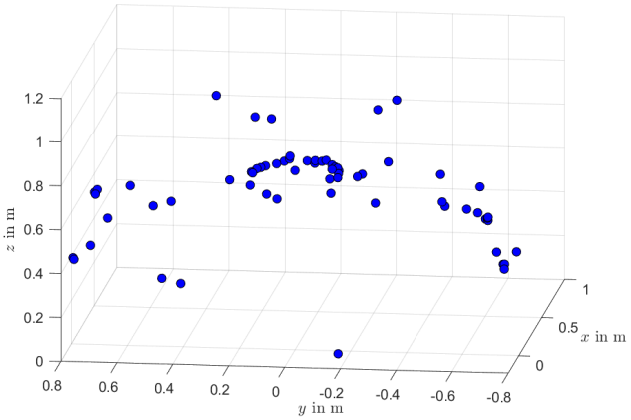


Abbildung 4.3: Optimale Konfigurationen für die geometrische Kalibrierung

4.3 Validierung

Um die Korrektheit und Qualität der Kalibrierung beurteilen zu können, werden zusätzlich N_{val} Validierungskonfigurationen $\boldsymbol{\xi}_{val}$ benötigt. Diese sollen möglichst gut im Arbeitsraum verteilt und unabhängig von den Kalibierungskonfigurationen sein. Hierfür gibt es verschiedene Möglichkeiten diese festzulegen. In dieser Arbeit wurden die $N_{val} = 125$ Validierungskonfigurationen dargestellt in Abbildung 4.4 verwendet.

[1] Durchgeführt wurde die Berechnung mit einem IntelCore i5 2.4 GHz Prozessor und 16 GB RAM.

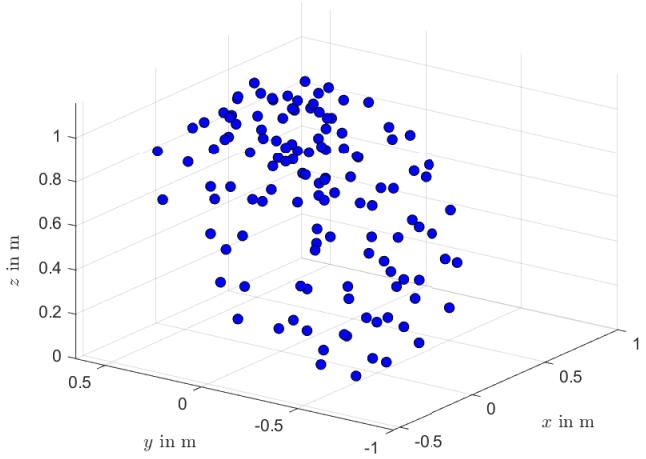


Abbildung 4.4: Validierungskonfigurationen

4.4 Kompensation geometrischer Fehler

Da in der Robotersteuerung keine direkte Korrektur des Kinematikmodells möglich ist, wird die Solltrajektorie im Konfigurationsraum so angepasst, dass die Fehler im Operationsraum kompensiert werden. Hierzu bedarf es der korrigierten Inverskinematik. Diese ist jedoch aufgrund der Achsschiefstellungen nicht mehr analytisch lösbar, weshalb auf eine numerische Lösung übergegangen werden muss. Diese basiert auf der Inverskinematik auf Geschwindigkeitsebene. Analog zu (2.12) und (2.35) im Abschnitt 2.1.2 ergibt sich diese zu

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\left(\mathbf{q}, \mathbf{p}_e\right) \mathbf{V}_{IE}\,. \tag{4.10}$$

Wird nun diese Lösung numerisch integriert, führt dies zu einem Fehler zufolge des geometrischen Fehlers auf Positionsebene und des numerischen Drifts aus der numerischen Integration. Dieser Fehler muss mit einem Regelkreis kompensiert werden. Hierfür wird das Fehlersystem

$$\dot{\bar{\mathbf{e}}} + \mathbf{K}\,\bar{\mathbf{e}} = 0 \tag{4.11}$$

definiert mit dem Fehler $\bar{\mathbf{e}} = \bar{\mathbf{e}}\left(\mathbf{z}_{IE,d}, \mathbf{z}_{IE}\left(\mathbf{q}, \mathbf{p}_e\right)\right)$ nach (2.40) mit (2.55), welches asymptotisch stabil ist, wenn $\mathbf{K}$ positiv definit ist. Setzt man hier (2.41) ein so ergibt sich

$$\mathbf{L}^T\mathbf{V}_{IE,d} - \mathbf{L}\,\mathbf{V}_{IE} + \mathbf{K}\,\bar{\mathbf{e}} = 0\,, \tag{4.12}$$

woraus sich der Endeffektor Twist zu

$$\mathbf{V}_{IE} = \mathbf{L}^{-1}\left(\mathbf{L}^T\mathbf{V}_{IE,d} + \mathbf{K}\,\bar{\mathbf{e}}\right) \tag{4.13}$$

bestimmen lässt. Mit (2.56) ergibt sich in weiterer Folge

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\left(\mathbf{q}, \mathbf{p}_e\right)\left(\mathbf{L}^{-1}\left(\mathbf{L}^T\mathbf{V}_{IE,d} + \mathbf{K}\,\bar{\mathbf{e}}\right)\right)\,, \tag{4.14}$$

was unter der Annahme kleiner Abweichungen vereinfacht werden kann zu

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\left(\mathbf{q}, \mathbf{p}_e\right)\left(\mathbf{V}_{IE,d} + \mathbf{K}\,\bar{\mathbf{e}}\left(\mathbf{z}_{IE,d},\, \mathbf{z}_{IE}\left(\mathbf{q}, \mathbf{p}_e\right)\right)\right) . \tag{4.15}$$

Durch Integration ergibt sich die kompensierte Trajektorie in Gelenkkoordinaten. Der gesamte Aufbau ist in Abbildung 4.5 dargestellt.

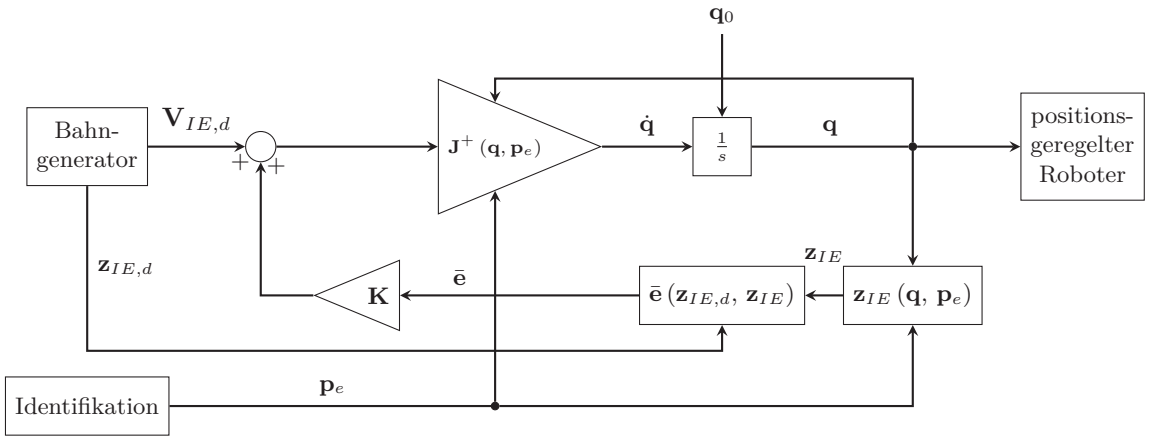


Abbildung 4.5: Kompensation der geometrischen Fehler

Kapitel 5

Kamerabasierte Regelung

Eine weitere Möglichkeit zur Steigerung der absoluten Genauigkeit besteht in der direkten Regelung auf die durch die Kamera erfasste Pose des Endeffektors. Dies bietet den Vorteil, sowohl die statische Positioniergenauigkeit als auch die dynamische Präzision zu verbessern. Allerdings können einzelne Messfehler bei der Posen-Erfassung in diesem Ansatz erheblich größere Auswirkungen haben.

Die kamerabasierte Regelung baut auf einer Kaskadenregelung auf, da bei vielen industrietauglichen Robotersteuerungen kein direkter Zugriff auf die vom Roboterhersteller implementierte Regelung besteht, sondern lediglich Sollpositionen und/oder Geschwindigkeiten vorgegeben werden können, oder nur die Verwendung von high-level Bewegungsbefehle möglich ist, welche wenig Spielraum bieten. Dies ist auch bei der in dieser Arbeit verwendeten ABB - Steuerung IRC5 der Fall. Die Struktur der Regelung ist in Abbildung 5.1 dargestellt, wobei zur Vereinfachung der Roboter direkt als System mit Motormomenten als Eingang modelliert wurde. In der Praxis werden diese Momente durch Elektromotoren aufgebracht, was eine zusätzliche innere Kaskade zur Stromregelung erfordert. Da die exakte implementierte Regelstruktur in der ABB-Steuerung jedoch nicht bekannt ist, hat dies keinen großen Einfluss auf das simulierte Regelverhalten, das ohnehin auf Modellannahmen basiert. Zur Untersuchung und Vergleich des Regelverhaltens werden in Matlab Simulink, zwei unterschiedliche Modelle betrachtet: ein geschwindigkeitsgeregelter Roboter, beschrieben in Abschnitt 5.3 und ein positionsgeregelter Roboter, beschrieben in Abschnitt 5.2.

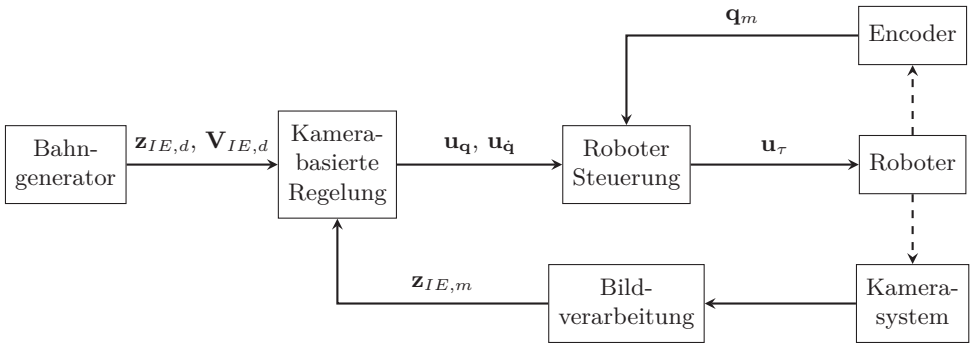


Abbildung 5.1: Regelungskonzept

Für die äußere Kaskade, die die kamerabasierte Regelung darstellt, werden drei unterschiedliche Konzepte betrachtet, welche in den Abschnitten 5.4 - 5.6 beschrieben werden. Zusätzlich wird zwischen äußerer und inneren Stufe die Stellgröße $\mathbf{u_q}$ bzw. $\mathbf{u_{\dot{q}}}$ beschränkt, wobei die Grenzwerte aus dem Datenblatt entnommen wurden und in Anhang B.1 zusammengefasst sind.

Als Kameramodell wird in der Simulation die Vorwärtskinematik gemäß (2.9) verwendet, mit einem zusätzlichen Messfehler bei der Positionsmessung in Form eines weißen Rauschen mit einer Rauschleistung von $1 \cdot 10^{-11}$. Außerdem wird eine Zeitverzögerung aufgrund der Datenübertragung und Bildverarbeitung mit 4 ms simuliert.

Zum Vergleich der Performance der verschiedenen Regelkonzepte wird im Folgenden der Fehler $\mathbf{e} = \mathbf{e}\left(\mathbf{z}_{IE,d}, \mathbf{z}_{IE,m}\right)$ nach (2.36) verwendet. Verglichen wird dabei die Abweichung von der mittels Kameras gemessenen Pose $\mathbf{z}_{IE,m}\left(t\right)$ von der in Abschnitt 5.1 beschriebenen Trajektorie $\mathbf{z}_{IE,d}\left(t\right)$, bei einem anfänglichen Positionsfehler von 0.01 m in jeder Richtung. Die Abweichung der Orientierung wird außerdem mit

$$\boldsymbol{\rho} = \begin{pmatrix} \rho_x \\ \rho_y \\ \rho_z \end{pmatrix} = \mathbf{e}_O\left(\mathbf{z}_{IE,d}, \mathbf{z}_{IE,m}\right) \tag{5.1}$$

abgekürzt.

5.1 Test-Trajektorie

Als Sollbahn wird die in Abbildung 5.2 dargestellte rechteckförmige Bahn, mit einer Orientierungsänderung von einem Eckpunkt zum nächsten um 90° in z gewählt. Die genauen Bahn-Zwischenpunkte sind in Tabelle 5.1 zusammengefasst. Als Basisprofil wird ein $\sin^2$-Beschleunigungsprofil gewählt mit $k_a = 0.7\,\mathrm{m/s^2}$, welches durch folgenden Beschleunigungsverlauf definiert ist

$$\ddot{\sigma}(t) = \begin{cases} 0 & t < t_0 \\ k_a \sin^2\left(\pi\sqrt{\frac{k_a}{2}}(t-t_0)\right) & t < t_0 + \sqrt{\frac{2}{k_a}} \\ -k_a \sin^2\left(\pi\sqrt{\frac{k_a}{2}}(t-t_0)\right) & t < t_0 + \frac{2}{\sqrt{\frac{k_a}{2}}} \\ 0 & \text{sonst} \end{cases}. \tag{5.2}$$

Für eine Bahnplanung in Weltkoordinaten ergibt sich der zeitliche Verlauf der Pose somit zu

$${}_I\mathbf{r}_{IE}(t) = {}_I\mathbf{r}_{IE,i} + \sigma(t-t_i)\left({}_I\mathbf{r}_{IE,i+1} - {}_I\mathbf{r}_{IE,i}\right), \quad t_i \le t < t_{i+1},\, i = 0,..E\,, \tag{5.3}$$

$${}_I\boldsymbol{\varphi}_{IE}(t) = \mathbf{f}_{R2Euler}\underbrace{\left(\exp\left(\tilde{\mathbf{e}}_{i,i+1}\,\varphi_{i,i+1}\,\sigma(t-t_i)\right)\right)}_{\mathbf{R}_{i,i+1}}, \quad t_i \le t < t_{i+1},\, i = 0,..E\,, \tag{5.4}$$

wobei $\mathbf{e}_{i,i+1}$ die Drehachse der Drehung vom $i$ten zum $i+1$ten Zwischenpunkt ist und $\varphi_{i,i+1}$ der notwendige Drehwinkel. Beide lassen sich mit (2.38), (2.39) und (2.7) aus den Eulerwinkeln der Zwischenpunkte berechnen. Für den Twist gilt

$$ {}_I\mathbf{v}_{IE}(t) = \dot{\sigma}(t-t_i)\,({}_I\mathbf{r}_{IE,i+1} - {}_I\,\mathbf{r}_{IE,i})\,, \qquad t_i \leq t < t_{i+1},\, i = 0,..E \tag{5.5} $$

$$ {}_I\tilde{\boldsymbol{\omega}}_{IE}(t) = \mathbf{R}_{i,i+1}\,\tilde{\mathbf{e}}_{i,i+1}\,\varphi_{i,i+1}\,\dot{\sigma}(t-t_i)\,\mathbf{R}_{i,i+1}^{T}, \qquad t_i \leq t < t_{i+1},\, i = 0,..E. \tag{5.6} $$

Tabelle 5.1: Zwischenpunkte der Testtrajektorie

Zwischenpunkte	$\mathbf{z}_{IE,1}$	$\mathbf{z}_{IE,2}$	$\mathbf{z}_{IE,3}$	$\mathbf{z}_{IE,4}$	$\mathbf{z}_{IE,5}$
x in mm	350	550	550	350	350
y in mm	450	450	−450	−450	450
z in mm	400	400	400	400	400
α in °	180	90	0	90	180
β in °	0	0	0	0	0
γ in °	90	90	90	90	90

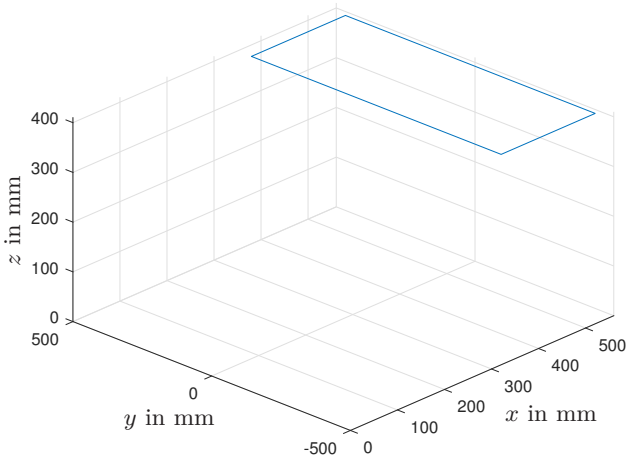


Abbildung 5.2: Solltrajektorie

5.2 Positionsgeregelter Roboter

Das positionsgeregelte Robotermodell (dargestellt in Abbildung 5.3) besteht aus dem in Abschnitt 2.4 beschriebenen dynamischen Modell des IRB1200, welches mit einem PI-Regler, mit den Parametern $\mathbf{K}_P = 80$ und $\mathbf{K}_I = 1$ geregelt wird, wobei zusätzlich die Schwerkrafteinflüsse kompensiert werden. Die Regelung erfolgt auf Grundlage der vorgegebenen Gelenkpositionen $\mathbf{q}_d$, wobei eine Übertragungsverzögerung von $T_{del} = 4\,\text{ms}$ berücksichtigt wurde.

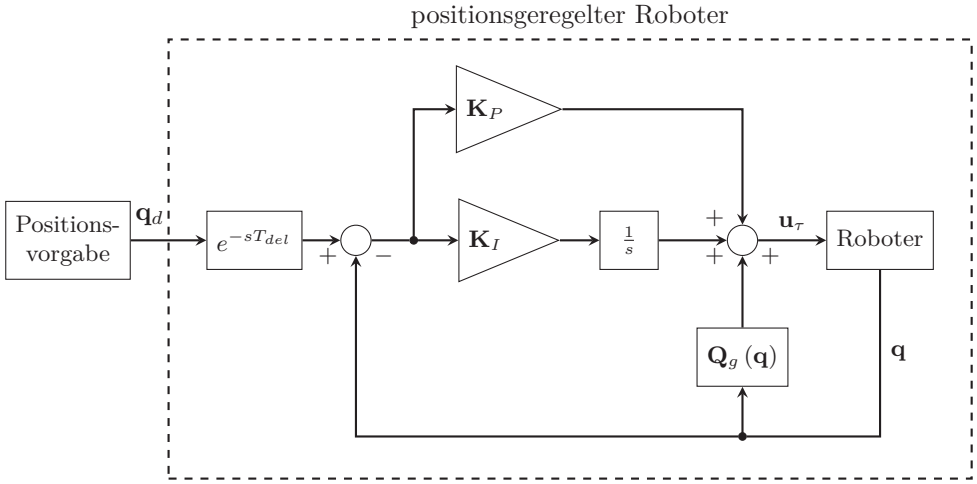


Abbildung 5.3: Positionsgeregelter Roboter

Für den geregelte Roboter ergibt sich in der Matlab Simulink Simulation der in Abbildung 5.4 dargestellte Verlauf des Positionsfehlers, sowie der in Abbildung 5.5 dargestellte Verlauf des Orientierungsfehlers.

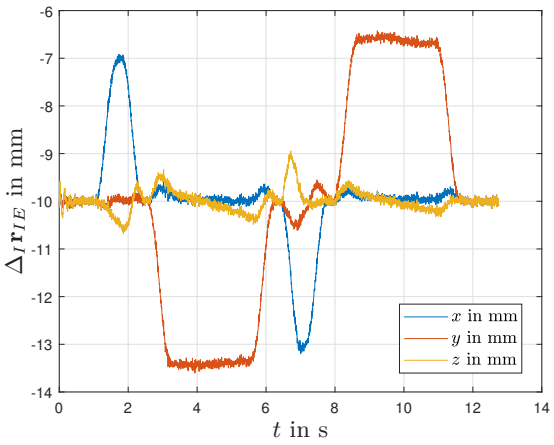


Abbildung 5.4: Simulink Simulation: Positionsfehler des positionsgeregelten Roboters

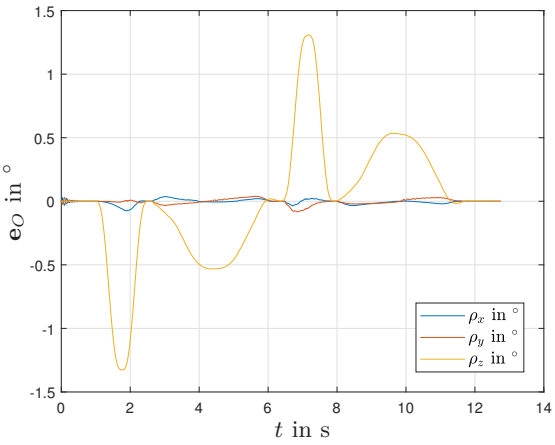


Abbildung 5.5: Simulink Simulation: Orientierungsfehler des positionsgeregelten Roboters

5.3 Geschwindigkeitsgeregelter Roboter

Der geschwindigkeitsgeregelte Roboter verwendet ebenfalls einen PI-Regler mit Schwerkraftkompensation. Für die Regelung wurden die Parameter $\mathbf{K}_P = 100$ und $\mathbf{K}_I = 1$ festgelegt. Die Vorgabe der Soll-Gelenkgeschwindigkeit $\dot{\mathbf{q}}_d$ erfolgt unter Berücksichtigung einer Übertragungsverzögerung von $T_{del} = 4\,\text{ms}$.

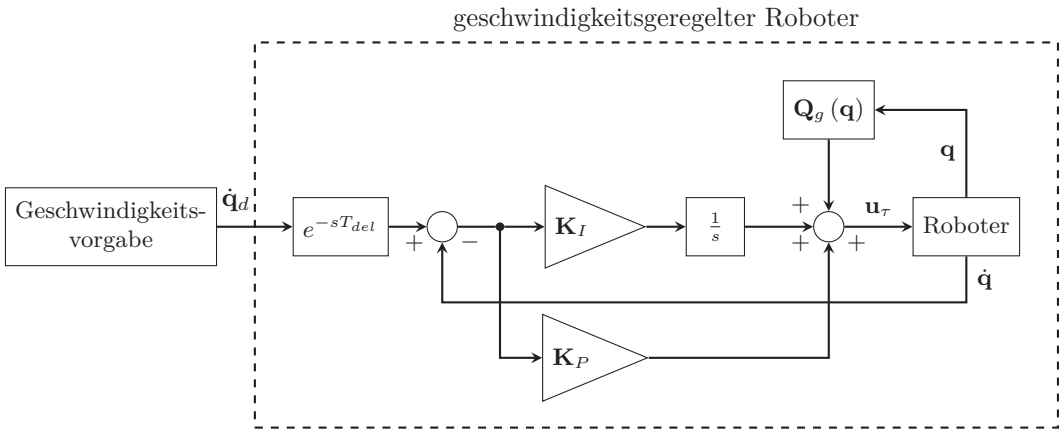


Abbildung 5.6: Geschwindigkeitsgeregelter Roboter

In der Matlab-Simulink Simulation ergeben sich die Fehlerverläufe Abbildung 5.7 und Abbildung 5.8.

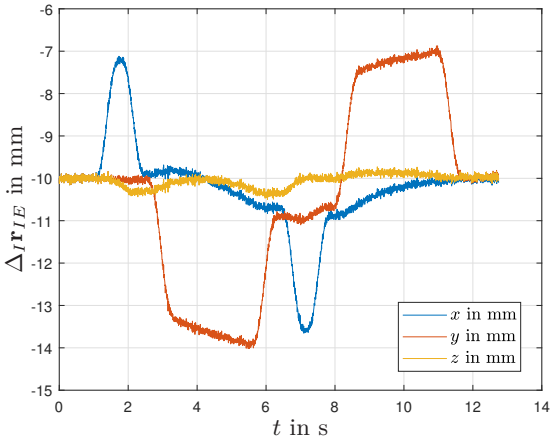


Abbildung 5.7: Simulink Simulation: Positionsfehler des geschwindigkeitsgeregelten Roboters

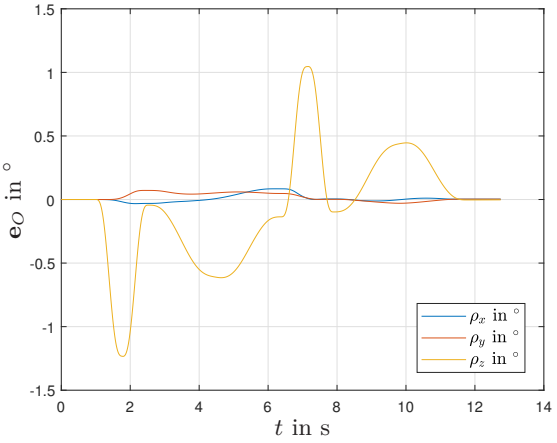


Abbildung 5.8: Simulink Simulation: Orientierungsfehler des geschwindigkeitsgeregelten Roboters

5.4 Konzept 1 (Numerische Inverskinematik)

Die in Abschnitt 4.4 beschriebene Methode zur Kompensation geometrischer Fehler eignet sich nach einer kleinen Abwandlung auch für die direkte Regelung auf die kameraerfasste Pose des Endeffektors. Es wird wiederum vom Fehlersystem

$$\dot{\bar{\mathbf{e}}} + \mathbf{K}\,\bar{\mathbf{e}} = 0 \tag{5.7}$$

ausgegangen, diesmal mit dem Fehler $\bar{\mathbf{e}} = \bar{\mathbf{e}}\,(\mathbf{z}_{IE,d}, \mathbf{z}_{IE,m})$ der aus der gemessenen Pose $\mathbf{z}_{IE,m}$ nach (2.40) ermittelt wird. Das Fehlersystem ist asymptotisch stabil wenn $\mathbf{K}$ positiv definit ist. Nach der in (4.12) ausgeführten Umformung lässt sich ein Ausdruck für den Twist $\mathbf{V}_{IE}$ ermitteln. Dieses Ergebnis lässt sich mit der Inverskinematik

(2.34) in $\dot{\mathbf{q}}$ transformieren. Aus diesem lässt sich durch Integration die Stellgröße des positionsgeregelten Roboters ermitteln

$$\mathbf{u_q} = \int_0^t \left(\mathbf{J}^+ (\mathbf{q})\, \mathbf{L}^{-1} \left(\mathbf{L}^T \mathbf{V}_{IE,d} + \mathbf{K}\, \bar{\mathbf{e}} \left(\mathbf{z}_{IE,d}, \mathbf{z}_{IE,m} \right) \right) \right) \mathrm{d}\tau + \mathbf{q}_0 \,. \tag{5.8}$$

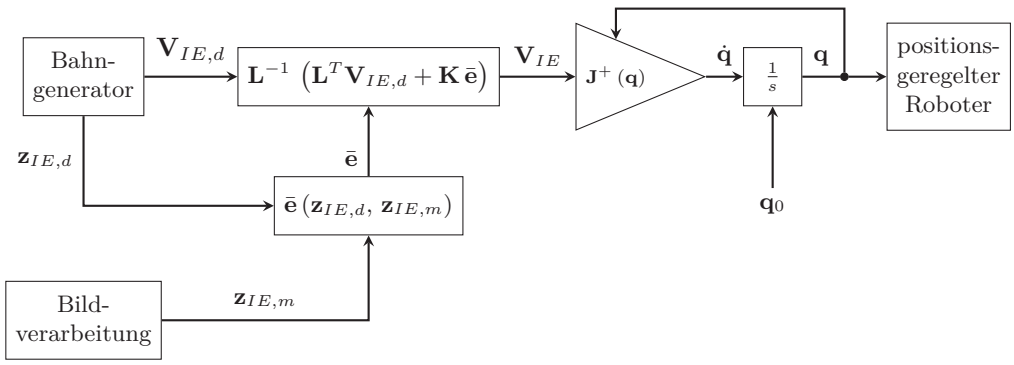


Abbildung 5.9: Regelungskonzept 1

Bei der Simulation mittels Matlab-Simulink ergibt sich, bei einer Wahl von $\mathbf{K} = \mathrm{diag}(2.5, 2.5, 2.5, 20, 20, 20)$ der in Abbildung 5.10 bzw. Abbildung 5.11 dargestellte Verlauf des Positionsfehlers und des Orientierungsfehlers.

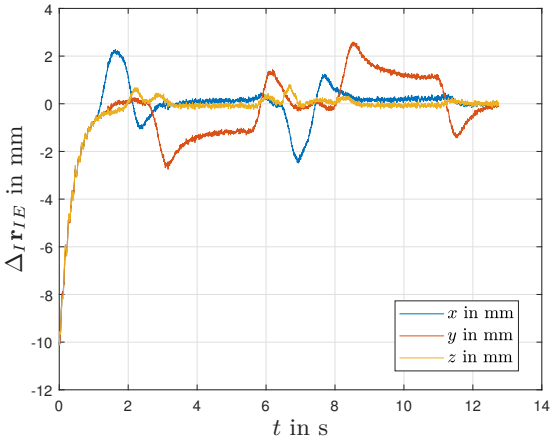


Abbildung 5.10: Simulink Simulation: Positionsfehler Regelungskonzept 1

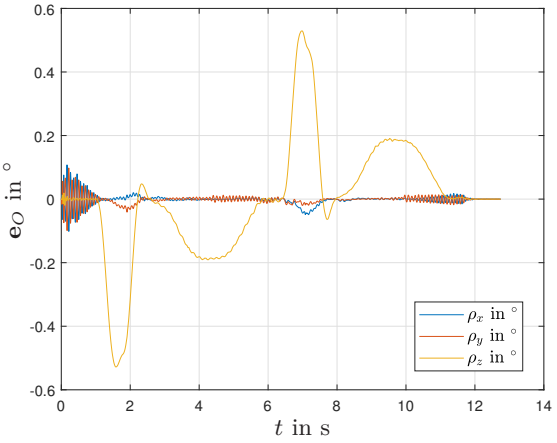


Abbildung 5.11: Simulink Simulation: Orientierungsfehler Regelungskonzept 1

5.5 Konzept 2 (Position-based Visual Servoing)

Dieses Regelungskonzept, dargestellt in Abbildung 5.12, basiert auf dem Position-based Visual Servoing nach [16]. Hierzu wird das Fehlersystem

$$\dot{\underline{\mathbf{e}}} + \mathbf{K}\, \underline{\mathbf{e}} = 0 \tag{5.9}$$

mit dem Fehler

$$\underline{\mathbf{e}} = \mathbf{z}_{IE,d} - \mathbf{z}_{IE,m} = \begin{pmatrix} {}_I\mathbf{r}_{IE,d} - {}_I\mathbf{r}_{IE,m} \\ {}_I\boldsymbol{\varphi}_{IE,d} - {}_I\boldsymbol{\varphi}_{IE,m} \end{pmatrix} \tag{5.10}$$

eingeführt, welches asymptotisch stabil ist insofern $\mathbf{K}$ positiv definit ist. Durch Einsetzen und Umformen ergibt sich

$$\dot{\mathbf{z}}_{IE} = \dot{\mathbf{z}}_{IE,d} + \mathbf{K}\,\underline{\mathbf{e}}\,, \tag{5.11}$$

was sich wiederum durch die Transformation

$$\mathbf{V}_{IE} = \mathbf{T}\left(\mathbf{z}_{IE}\right)\,\dot{\mathbf{z}}_{IE}\,, \quad \mathbf{T} = \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^T & 0 & -\sin(\alpha_{IE}) & \cos(\alpha_{IE})\cos(\beta_{IE}) \\ \mathbf{0}^T & 0 & \cos(\alpha_{IE}) & \sin(\alpha_{IE})\cos(\beta_{IE}) \\ \mathbf{0}^T & 1 & 0 & -\sin(\beta_{IE}) \end{pmatrix} \tag{5.12}$$

als Twist

$$\mathbf{V}_{IE} = \mathbf{T}\left(\mathbf{z}_{IE}\right)\left(\dot{\mathbf{z}}_{IE,d} + \mathbf{K}\,\underline{\mathbf{e}}\right) \approx \dot{\mathbf{V}}_{IE,d} + \mathbf{T}\left(\mathbf{z}_{IE}\right)\left(\mathbf{K}\,\underline{\mathbf{e}}\right) \tag{5.13}$$

darstellen lässt. Dieser Twist lässt sich nach (2.34) in die Gelenksgeschwindigkeitsebene transformieren mit der Vereinfachung $\mathbf{q} \approx \mathbf{q}_d$ ergibt sich somit das Regelgesetz für einen geschwindigkeitsgeregelten Roboter

$$\mathbf{u}_{\dot{\mathbf{q}}} = \underbrace{\mathbf{J}^+\left(\mathbf{q}_d\right)\,\dot{\mathbf{V}}_{IE,d}}_{\dot{\mathbf{q}}_d} + \underbrace{\mathbf{J}^+\left(\mathbf{q}\right)\,\mathbf{T}\left(\mathbf{z}_{IE}\right)\,\left(\mathbf{K}\,\left(\mathbf{z}_{IE,d} - \mathbf{z}_{IE,m}\right)\right)}_{\Delta\dot{\mathbf{q}}}\,. \tag{5.14}$$

Zur Erhöhung der Sicherheit, falls in der Bildverarbeitung eine Pose falsch identifiziert wird, wird zusätzlich die Korrektur der soll-Gelenksgeschwindigkeit $\Delta\dot{\mathbf{q}}$ beschränkt.

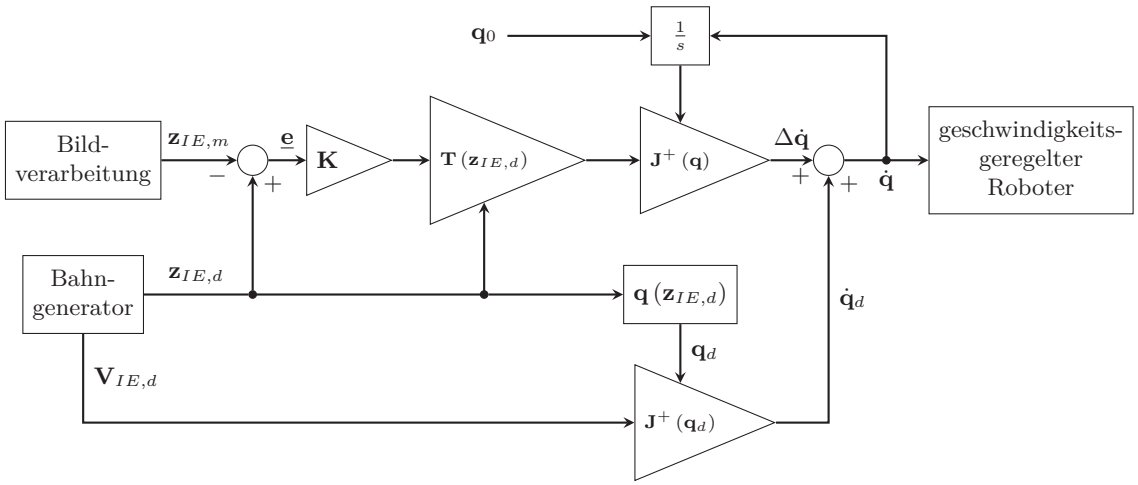


Abbildung 5.12: Regelungskonzept 2

Bei einer Wahl von $\mathbf{K} = 25\,\mathbf{I}$, ergibt sich in der Simulation, der in Abbildung 5.13 und Abbildung 5.14 dargestellte Fehlerverlauf.

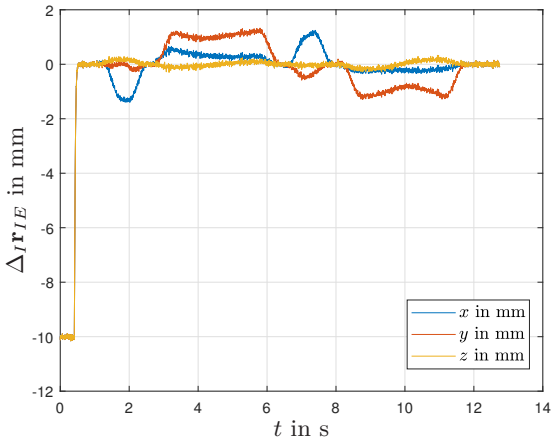


Abbildung 5.13: Simulink Simulation: Positionsfehler Regelungskonzept 2

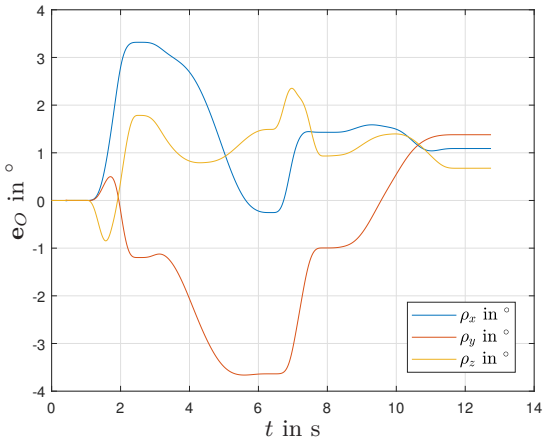


Abbildung 5.14: Simulink Simulation: Orientierungsfehler Regelungskonzept 2

5.6 Konzept 3 (Fehlersystem auf Positionsebene)

Das letzte Regelkonzept basierte auf dem selben Fehlersystem wie jenes aus Abschnitt 5.5

$$\dot{\underline{\mathbf{e}}} + \mathbf{K}\,\underline{\mathbf{e}} = 0\,, \tag{5.15}$$

mit dem Fehler $\underline{\mathbf{e}}$ nach (5.10). Durch Einsetzen der zeitlichen Ableitung von $\underline{\mathbf{e}}$ ergibt sich

$$\dot{\mathbf{z}}_{IE,d} - \dot{\mathbf{z}}_{IE} + \mathbf{K}\,\underline{\mathbf{e}} = 0 \tag{5.16}$$

und durch weiteres Umformen lässt sich ein Ausdruck für $\dot{\mathbf{z}}_{IE}$ finden, der sich wiederum integrieren lässt

$$\mathbf{z}_{IE} = \int_0^t \left(\dot{\mathbf{z}}_{IE,d} + \mathbf{K}\,\underline{\mathbf{e}}\right) \mathrm{d}\tau + \mathbf{z}_{IE,d_0} + \underline{\mathbf{e}}_0\,. \tag{5.17}$$

Die dadurch erhaltene Pose kann mittels der Inverskinematik auf Positionsebene in die Stellgröße $\mathbf{q}$ des positionsgeregelten Roboters transformiert werden, was zum Regelgesetz

$$\mathbf{u}_\mathbf{q} = \mathbf{q}\left(\mathbf{z}_{IE}\right) \tag{5.18}$$

$$\mathbf{z}_{IE} = \mathbf{z}_{IE,d} + \int_0^t \mathbf{K}\,\left(\mathbf{z}_{IE,d} - \mathbf{z}_{IE,m}\right) \mathrm{d}\tau + \underline{\mathbf{e}}_0 \tag{5.19}$$

führt. Um die Sicherheit zu erhöhen, im Fall einer falschen Identifikation der Pose durch die Bildverarbeitungsalgorithmen, wird zusätzlich die Korrektur der soll-Pose $\Delta\mathbf{z}_{IE}$ beschränkt.

Für einen fairen Vergleich wird auch hier analog zu Abschnitt 5.4 für $\mathbf{K} = \mathrm{diag}(2.5, 2.5, 2.5, 20, 20, 20)$ gewählt, wodurch sich in der Simulation der in Abbildung 5.16 und Abbildung 5.17 dargestellte Fehlerverlauf ergibt.

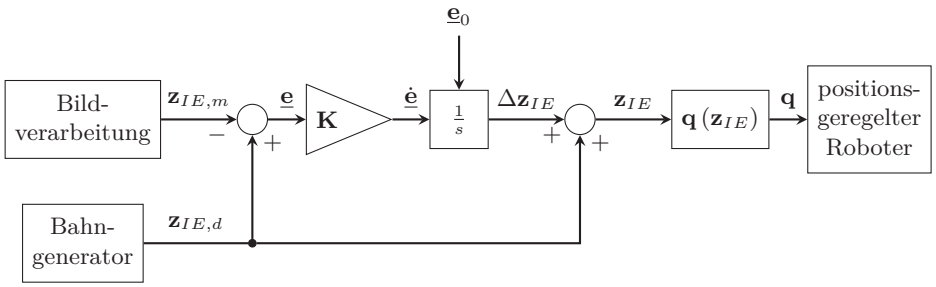


Abbildung 5.15: Regelungskonzept 3

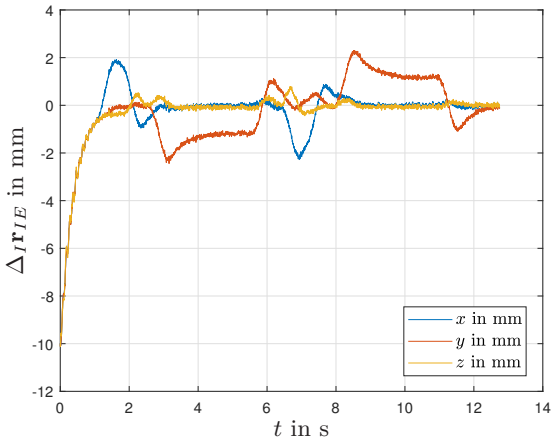


Abbildung 5.16: Simulink Simulation: Positionsfehler Regelungskonzept 3

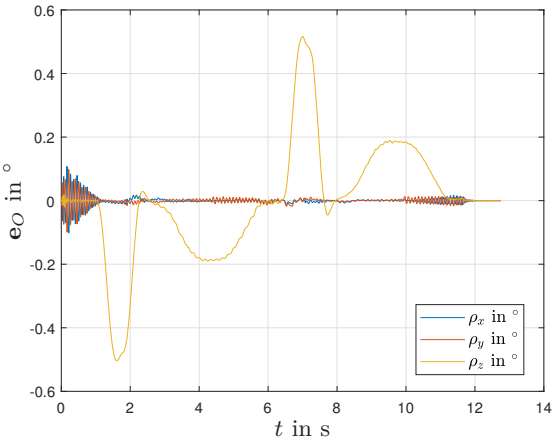


Abbildung 5.17: Simulink Simulation: Orientierungsfehler Regelungskonzept 3

5.7 Vergleich der Konzepte

In der Simulation zeigen Konzept 1 und Konzept 3 ein nahezu identisches Regelverhalten. Beide Konzepte ermöglichen es dem Regler, den Fehler bei der Sollposition schnell zu kompensieren, wobei das Überschwingen geringer ausfällt als bei der herkömmlichen Positions- oder Geschwindigkeitsregelung. Konzept 2 erreicht die Sollposition am schnellsten und weist das geringste Überschwingen auf. Allerdings treten bei diesem Konzept Schwierigkeiten bei der Ausregelung der Orientierung auf, was wahrscheinlich auf die in Abschnitt 5.5 getroffenen Annahmen sowie auf numerische Fehler zurückzuführen ist.

Kapitel 6

Durchführung

Das in Kapitel 4 vorgestellte Konzept zur Steigerung der Positioniergenauigkeit mittels geometrischer Kalibrierung wird in diesem Kapitel mit den in Kapitel 5 beschriebenen Regelungskonzepten verglichen. Dies erfolgt in zwei Schritten: Zunächst wird in einer Simulation mit RobotStudio (RS), der Simulations- und Entwicklungsumgebung von ABB, die Funktionalität und das Konzept überprüft (Abschnitt 6.4.1 und Abschnitt 6.5.1). Hierzu wurde in RS ein digitaler Zwilling der verwendeten IRC5-Robotersteuerung mit dem verwendeten IRB1200 Roboter erstellt und die verschiedenen Geometrien sowie der Detektionskörper importiert. Im zweiten Schritt werden die Versuche direkt am beschriebenen Versuchsaufbau durchgeführt (Abschnitt 6.4.2 und Abschnitt 6.5.2). Sowohl der Versuchsaufbau des ersten und des zweiten Schrittes sind in Abschnitt 6.1 definiert, außerdem wird in Abschnitt 6.2 auf das verwendete Messsystem eingegangen. Die Kommunikation mit dem Roboter wird hierbei über die ABB Schnittstelle Externally Guided Motion realisiert, wobei die Funktionalität dieser Schnittstelle in Abschnitt 6.3 beschrieben wird mit weiterer Ausführung in Anhang A.1, A.2 und A.3.

6.1 Versuchsaufbau

Das zentrale Element des Versuchsaufbaus bildet der Industrieroboter IRB 1200 von ABB mit einer Reichweite von 0.7 m und einer maximalen Traglast von 7 kg. Am Endeffektor ist ein selbstgefertigter Detektionskörper mittels eines Schunk-Werkzeugwechselsystems befestigt, welcher in Abschnitt 6.2.1 beschrieben wird. Die infrarot-reflektierenden Marker des Detektionskörper[1] werden von 10 high-Speed Kameras vom Typ Oqus 6+ des Motion-Capturingsystems Qualisys erfasst. An den Kameras ist direkt eine IR-Beleuchtung und ein IR-Filter integriert, außerdem befindet sich an der Kamera ein kleiner Prozessor der für die erste Bildauswertung zuständig ist. Die Kameras sind über eine Daisy-chained Ethernet-Verbindung mit der zentralen Recheneinheit verbunden. Die Recheneinheit wird durch einen PC[2] mit Windows 10 Education Betriebssystem dargestellt. Auf diesem läuft der Qualisys

[1] Designed mit Creo Parametric Student Edition Version 10.0.0.0.

[2] Der PC ist ausgestattet mit einem Intel Core i9-9900K 3.6 GHz Prozessor und verfügt 32 GB RAM.

DHCP Server (QDS)[3], welcher für die Kommunikation mit den Kameras zuständig ist, sowie der Qualisys Track Manager (QTM)[4], welcher für die Steuerung der Kameraeinstellungen, Definierung der erfassten Körper, Kamerakalibrierung, ect. zuständig ist. Eine genaue Beschreibung der Funktionen des QTM erfolgt in Abschnitt 6.2.2. Des Weiteren läuft auf dem PC eine Echtzeitanwendung[5] die mittels ABB Externally Guided Motion (EGM) über eine Ethernet-Verbindung mit der ABB Steuerung IRC5 kommuniziert. Die EGM - Kommunikation wird genauer in Abschnitt 6.3 beschrieben. Die Robotersteuerung ist wiederum mit dem ABB IRB1200 verbunden. Sowohl die EGM-Echtzeitanwendung als auch der QTM weisen eine Schnittstelle zu einem Simulink[6] Modell mit Realtime-Pacer auf in welchem die in den vorherigen Abschnitten erläuterten kamerabasierten Konzepte zur Genauigkeitssteigerung implementiert sind. Eine Überblick über den Versuchsaufbau liefert Abbildung 6.1.

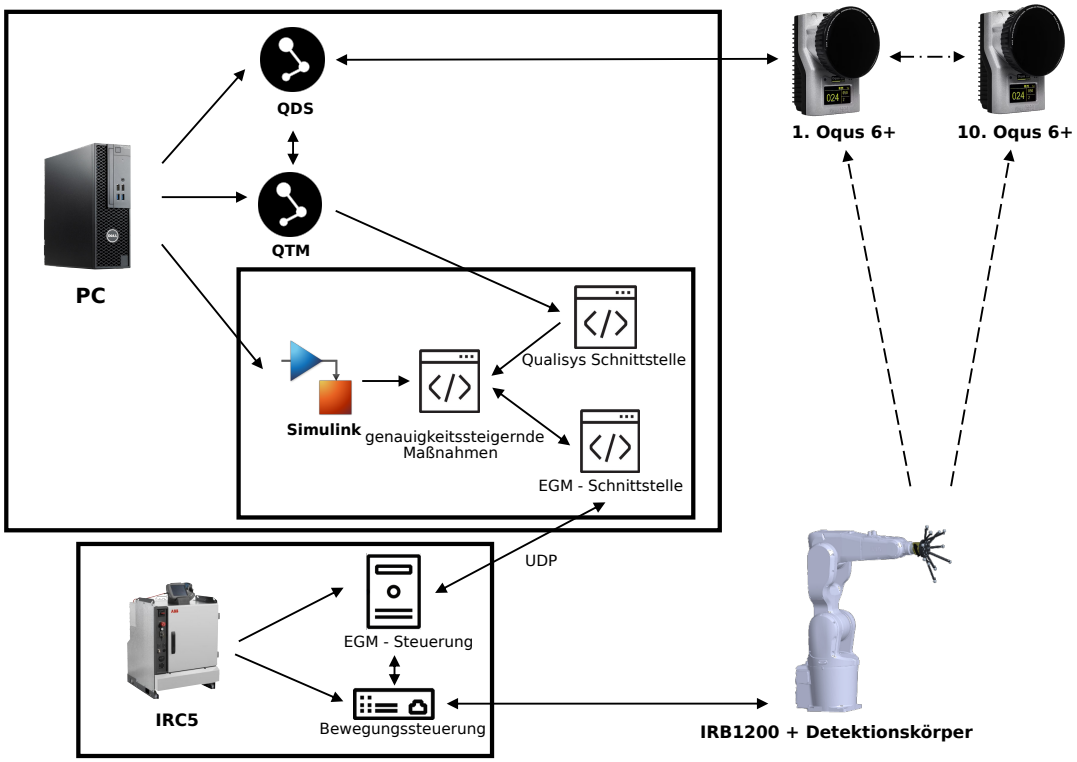


Abbildung 6.1: Versuchsaufbau

Um die Funktionalität der verschiedenen Versuche im Vorfeld zu überprüfen und potenzielle Risiken für das reale Versuchssetup zu minimieren, werden diese zunächst in einer Simulationsumgebung getestet. Der Kern dieser Simulation besteht aus der Verwendung von RobotStudio[7], der Simulationsumgebung von ABB, in der sowohl der Roboter als auch die Robotersteuerung virtuell abgebildet werden. Die virtuelle

[3]QDS-Version 3.0 (build 6020).
[4]QTM-Version 2020.3 (build 6020) kompatibel mit RT-Protocol Versionen 1.0 - 1.21.
[5]Entwickelt mit Visual Studio Community 2022 Version 17.2.3.
[6]Verwendet wurden die Matlab Version R2023b Update 3 23.2.0.2409890 und die Simulink Version 23.2 (R2023b).
[7]Verwendet wurde RobotStudio 2023.2.1 Version 23.2.10478.1.

Robotersteuerung kommuniziert lokal über EGM mit der EGM-Echtzeitanwendung. Diese Echtzeitanwendung bietet eine Schnittstelle zu einem Simulink-Modell, das mit einem Realtime-Pacer ausgestattet ist, um die Simulation in Echtzeit durchzuführen. In diesem Simulink-Modell wird die Endeffektorpose, die normalerweise aus den Kameradaten ermittelt würde, durch die über EGM gestreamte Pose simuliert. Um realistische Bedingungen nachzubilden, werden diese Pose-Daten zusätzlich mit einem Offset sowie einem Rauschen versehen. Ebenso werden die Gelenkwinkelvorgaben, die über EGM an das Robotermodell übermittelt werden, mit einem Offset versehen, um Nulllagenfehler der Gelenke zu simulieren. Die angenommenen Fehler für diese Simulation sind in Tabelle 6.1 zusammengefasst. Der gesamte Simulationsaufbau ist in Abbildung 6.2 skizziert.

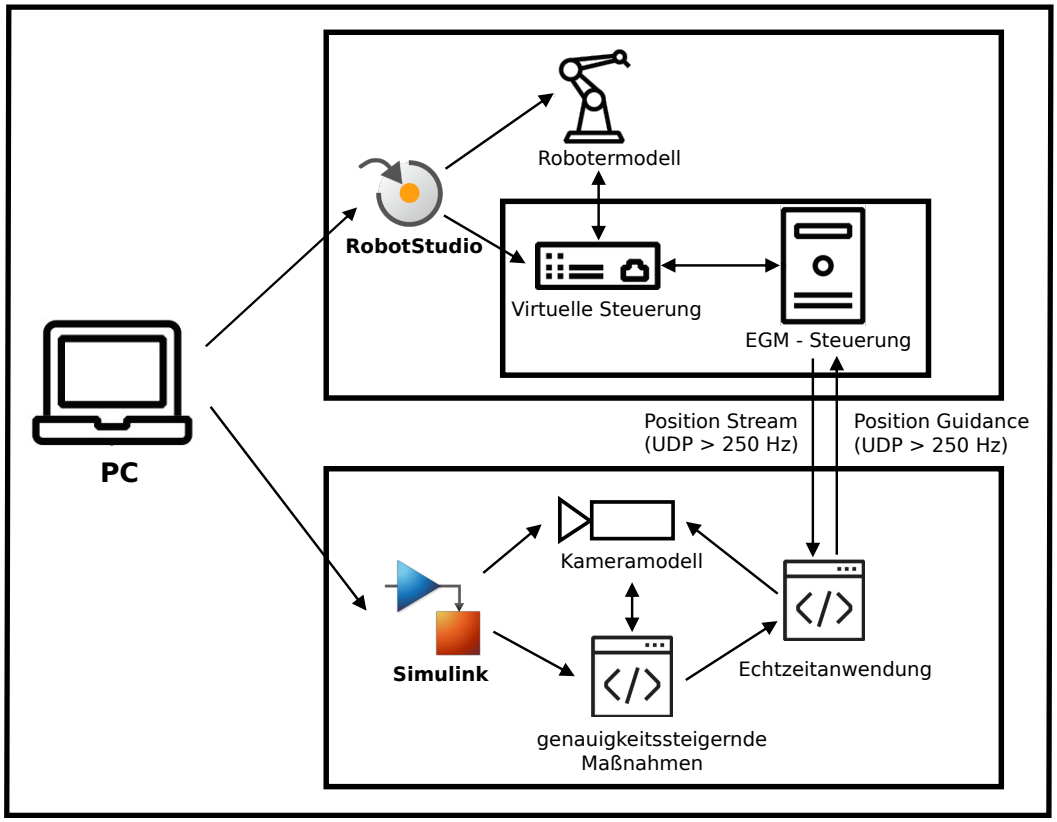


Abbildung 6.2: Aufbau der Simulationsumgebung

Tabelle 6.1: Simulierte Fehler

Mit Fehler beaufschlagte Größe	entsprechender Fehlerparameter	Offset	Rauschen
q_1	p_{e_1}	1°	0.01°
q_2	p_{e_2}	1.5°	0.01°
q_3	p_{e_3}	2°	0.01°
q_4	p_{e_4}	2.5°	0.01°
q_5	p_{e_5}	3°	0.01°
q_6	p_{e_6}	3.5°	0.01°
x_E	p_{e_7}	10 mm	0.01 mm
y_E	p_{e_8}	10 mm	0.01 mm
z_E	p_{e_9}	10 mm	0.01 mm
α_E	p_{e_6}	0°	0.001°
β_E	$p_{e_{29}}$	0°	0.001°
γ_E	$p_{e_{30}}$	0.1°	0.001°

6.2 Messsystem

Als Messsystem wurde sich für das Motioncapturingsystem der Firma Qualisys entschieden. Mit diesem wird ein eigensangefertigter Detektionskörper erfasst, welcher am Endeffektor des Industrieroboters befestigt ist. Die Beschreibung der Komponenten erfolgt in Abschnitt 6.2.1 und Abschnitt 6.2.2.

6.2.1 Detektionskörper

Der Detektionskörper besteht aus 11 Infrarotmarkern, die über modulare Arme an einer Grundplatte befestigt sind. Die modulare Bauweise der Arme ermöglicht eine flexible Positionierung der Infrarotmarker, sodass verschiedene Anordnungen getestet werden können, um die optimale Positionierung der Marker zu ermitteln. Ein weiterer Vorteil dieses Designs ist, dass im Falle einer Beschädigung lediglich einzelne Arme ersetzt werden müssen. Die Grundplatte bietet die Möglichkeit, den Detektionskörper entweder direkt am Roboterflansch oder indirekt über ein Schunk-Werkzeugwechselsystem zu befestigen. Aus Kostengründen und zur Zeitersparnis wurden die Arme und die Grundplatte im 3D-Druckverfahren hergestellt. Als Infrarotmarker werden 16 mm große, infrarotreflektierende Kugeln von Qualisys verwendet. Diese Marker bieten im Vergleich zu anderen passiven Markern eine schnelle und robuste Lokalisierung im Bild. Die Orientierung des gesamten Detektionskörpers wird durch die asymmetrische Anordnung der Marker eindeutig bestimmt. Zwar wären theoretisch nur 3 Marker erforderlich, um die Orientierung zu berechnen, doch durch die Erhöhung der Markeranzahl auf 11 wird die Robustheit des Systems deutlich gesteigert. Dies bedeutet, dass nicht immer alle Marker im Sichtfeld der Kameras sein müssen. Zudem verbessert die höhere Markeranzahl dank der in Abschnitt 3.1 beschriebenen Subpixeltechnik sowohl die Genauigkeit der Positions- als auch der Orientierungsbestimmung.

Der gesamte Detektionskörpers weist eine Größe von $\varnothing 350 \times 110$ mm auf und ist in Abbildung 6.3 dargestellt.

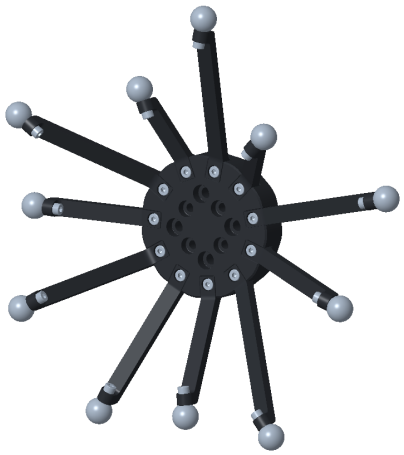


Abbildung 6.3: Detektionskörper

6.2.2 Qualisys

Qualisys bietet umfassende Motion-Capturing-Lösungen an, bestehend aus spezialisierten Kameras und Detektionssoftware. Die Kameras sind hochauflösende High-Speed-Kameras mit integriertem Vorverarbeitungssystem, das bereits einen Großteil der Objekterkennung, welche in Kapitel 3 beschrieben wurde, übernimmt. Sie arbeiten im Infrarotbereich und sind mit einem IR-Beleuchtungsring sowie einem IR-Filter ausgestattet. Verwendet werden in dieser Arbeit die in Abbildung 6.4 dargestellten Oqus 6+ Kameras mit 6 MP Auflösung und einer maximalen Bildrate von 450 fps.

Abbildung 6.4: Kamera Oqus 6+ von Qualisys

Die Software von Qualisys besteht aus dem Qualisys DHCP Server und dem Qualisys Track Manager:

- **Qualisys DHCP Server (QDS)**: Der QDS kümmert sich um die Kommunikation zwischen dem PC, der die Datenauswertung durchführt, und den Kameras.

Er stellt sicher, dass alle Kameras im Netzwerk erkannt werden, und weist diesen automatisch IP-Adressen zu. Der Server läuft im Hintergrund und sorgt für eine reibungslose Verbindung zwischen den Komponenten.

- **Qualisys Track Manager (QTM)**: Der QTM liefert die Werkzeuge zur Objekterkennung und Kamerakalibrierung. Hiermit ist es möglich die Kameras zu kalibrieren, sowie mehrere zu detektierende Objekte zu definieren. Für die Kalibrierung wird das vom Hersteller mitgelieferte Koordinatensystem und das Kalibrierwerkzeug benötigt. Während des Kalibrierungsprozesses wird der Arbeitsbereich mit dem Kalibrierwerkzeug abgefahren, die Software nutzt anschließend die Aufzeichnung, um die extrinsischen Kameraparameter zu bestimmen und zeigt im Anschluss die Genauigkeit der Kalibrierung an, welche in Folge bei den Ergebnissen in Abschnitt 6.4 und 6.5 angegeben wird. Das verwendete Kalibrierungssystem dient als globales Bezugssystem für alle weiteren Messungen. In QTM lassen sich auch die Kameraeinstellungen, wie Auflösung, Bildrate, Belichtung und Treshhold der Markererkennung anpassen. Für die in dieser Arbeit durchgeführten Versuche werden die in Tabelle A.8 dargestellten Einstellungen verwendet. Zudem können erkannte Marker zu Starrkörpern zusammengefasst werden, wodurch die Software automatisch die Position und Orientierung dieser Körper relativ zum Kalibrierungskoordinatensystem berechnet. Die Genauigkeit der Detektion hängt von der Position und Anzahl der Kameras, der Anzahl und Position der Marker sowie von den gewählten Kameraparametern ab (z. B. Auflösung, Bildrate, Belichtung, Markererkennungsschwelle). Zudem können Umwelteinflüsse wie Lichtverhältnisse oder reflektierende Oberflächen die Erkennung stören. QTM liefert außerdem eine Schnittstelle zum Datenexport an Matlab, wobei entschieden werden kann welche Daten gestreamt werden. Dazu zählen Bildpositionen der einzelnen Marker, sowie Position und Orientierung definierter Körper. Diese Schnittstelle wird in weiterer Folge auch in der Simulink Echtzeitanwendung verwendet.

Für eine detaillierte Erklärung der Funktionsweise von QDS und QTM wird auf das Benutzerhandbuch von Qualisys verwiesen [2]. Die Konfiguration von QTM für die Verwendung im Versuchsaufbau ist in Anhang A.4 beschrieben.

6.3 Externally Guided Motion

Der folgende Abschnitt stützt sich auf dem Benutzerhandbuch Externally Guided Motion von ABB [5] und fasst dessen Inhalte zusammen, mit Bezug auf die in dieser Arbeit verwendeten Funktionen. Für weiterführende Details wird auf das Handbuch verwiesen.

Externally Guided Motion (EGM) ist eine Schnittstelle, die eine direkte Kommunikation zwischen einem PC oder Server und einer ABB-Robotersteuerung ermöglicht. Es handelt sich um eine Punkt-zu-Punkt Kommunikation (unicast), bei der der PC als Sensor fungiert und die Robotersteuerung als Server. Diese Kommunikation basiert

auf dem User Datagram Protocol (UDP), einem Übertragungsprotokoll. Da bei UDP kein Protokoll für den Verbindungsaufbau existiert, kann weder garantiert werden, dass die Datenpakete (Datagramme) zugestellt werden, noch dass sie in der richtigen Reihenfolge ankommen oder Duplikate vermieden werden. Der Vorteil von UDP liegt jedoch in seiner geringen Latenzzeit und der effizienten Nutzung der Netzwerkressourcen. Um zusätzlich die Daten zwischen PC und Robotersteuerung effizient zu übermitteln, werden sie mit Google Protocol Buffers (Protobuf) serialisiert. Protobuf ist eine sprach- und plattformunabhängige Methode zur Serialisierung strukturierter Daten, ähnlich wie XML, jedoch weitaus kompakter und schneller. Die genaue Datenstruktur muss einmalig in einer .proto-Datei definiert werden, die den Aufbau der Nachrichten festlegt. ABB stellt diese .proto-Datei bereit, und anschließend kann der Source-Code generiert werden, der die Serialisierung und Deserialisierung der Daten übernimmt. Weitere Details sind in [28] zu finden, eine detaillierte Struktur der verschiedenen Kommunikationsschichten ist in Tabelle 6.2 dargestellt.

Tabelle 6.2: Kommunikationsschichten der EGM Kommunikation

OSI-Schicht	**Technologie**	**Beschreibung**
Schicht 7: Anwendungsschicht	Externally Guided Motion (EGM) Anwendung	Kommunikation von Messwerten und Bewegungsvorgaben zwischen PC und Robotersteuerung.
Schicht 6: Darstellungsschicht	Protobuf	Serialisierung und Deserialisierung der Daten zur effizienten Übertragung in binärem Format.
Schicht 5: Sitzungsschicht	-	Nicht verwendet in diesem Szenario.
Schicht 4: Transportschicht	UDP	Verbindungsloses Protokoll zur Übertragung von Datagrams
Schicht 3: Vermittlungsschicht	IP	Verantwortlich für die Weiterleitung der Pakete zwischen PC und Robotersteuerung über das Netzwerk.
Schicht 2: Sicherungsschicht	Ethernet	Stellt Frames für die physische Übertragung bereit und sorgt für Fehlererkennung.
Schicht 1: Physische Schicht	Ethernet-Kabel	Übertragung der elektrischen Signale über das Kabel.

EGM bietet drei verschiedene Moden an:

- **EGM Position Stream**: EGM Position Stream, ermöglicht das kontinuierliche Streamen der aktuellen Lage des Roboters, sowie zukünftiger Zielpunkte. Dabei können sowohl die Gelenkwinkel als auch die Pose in einem vordefinierten Koordinatensystem übertragen werden. Für die Darstellung der Orientierung können entweder Eulerwinkel (nach der ZYX-Konvention) oder Quaternionen verwendet werden. Dies wird ebenfalls für externe Achsen ermöglicht. Darüber hinaus können Statusinformationen wie der aktuelle Motorstatus oder der Programmstatus abgefragt werden als auch Daten externer Kraftsensoren. Jede Nachricht enthält einen Zeitstempel, der die Systemzeit der Steuerung zum Sendezeitpunkt wiedergibt. Die Daten werden in einem Intervall von etwa 4.032 ms übertragen,

was einer Frequenz von ungefähr 250 Hz entspricht, oder in einem Vielfachen dieses Intervalls, abhängig von der Konfiguration.

- **EGM Position Guidance**: Mit EGM Position Guidance kann die Roboter Trajektorie in Echtzeit vorgeben werden. Dies erfolgt entweder durch eine Kombination aus einer Darstellung in der Positionsebene und eine in der Geschwindigkeitsebene oder durch lediglich eine der beiden. Die Darstellung der Zielwerte kann entweder in Form von Gelenkwinkeln oder als Pose erfolgen. Wobei bei einer Darstellung als Pose, das Koordinatensystem frei definiert werden kann und die Darstellung der Orientierung entweder durch Eulerwinkel (nach der ZYX-Konvention) oder Quaternionen erfolgt. Laut ABB ist die Datenübertragung ebenfalls mit einer Übertragungsrate von 4.032 ms $\approx$ 250 Hz möglich, jedoch ist mit einer Kontrollverzögerung von $10 - 20$ ms zu rechnen.
- **EGM Path Correction**: Kleine Abweichungen von der offline programmierten Roboterbahn können auch mittels EGM Path Correction korrigiert werden. Dabei können Positionskorrekturen im Bahn-Koordinatensystem in Y und Z durchgeführt werden, wobei Z der Z-Richtung des aktiven Werkzeug-Koordinatensystems entspricht. Eine Korrektur in Bahnrichtung (X) oder der Orientierung ist nicht möglich. Die Kommunikation erfolgt mit einem Vielfachen von 48 ms.

Um EGM nutzen zu können, muss die Software-Option 689-1 *Externally Guided Motion* für die ABB-Robotersteuerung erworben werden. Außerdem muss die Steuerung mit einer Betriebssoftware (RobotWare) in Version 6.10 oder höher ausgestattet sein, wobei eine eingeschränkte Nutzung auch mit RobotWare-Versionen ab 6 möglich ist. Bevor EGM verwendet werden kann, muss es zunächst freigeschaltet und konfiguriert werden, Details sind in Anhang A.1 zu finden.

Zusätzlich wäre es im EGM Position Stream möglich, die sogenannten ABB-Testsignale zu streamen. Zum Zeitpunkt dieser Arbeit wurde diese Funktion jedoch noch nicht vollständig von ABB implementiert, weshalb ein Workaround über das ABB PC-Interface verwendet wird. Die Testsignale umfassen verschiedene von der Steuerung erfasste oder berechnete Werte, wie die Gelenkgeschwindigkeit und die Motormomente. Eine detaillierte Übersicht der Testsignale ist in Tabelle B.3 zu finden. Für die Nutzung des ABB PC-Interfaces muss die zusätzliche Software-Option 616-1 *PC Interface* freigeschaltet sein.

Für die EGM-Kommunikation muss auf der Roboter-Steuerung, als auch auf dem externen Gerät (PC) eine EGM-Anwendung implementiert sein. Für beide Seiten bietet ABB jedoch keine fertig entwickelte Programme an die diese bewerkstelligen, die Entwicklung ist dem Nutzer überlassen, wodurch mehr Flexibilität gegeben ist.

Die Implementierung auf der Robotersteuerung erfolgt in der ABB-Programmiersprache RAPID. ABB stellt hierfür vorgefertigte EGM-Funktionen zur Verfügung, die Bausteine zur Initialisierung und zum zyklischen Aufruf der verschiedenen EGM-Modi (wie EGM Position Stream und EGM Position Guidance) enthalten. Darüber hinaus ist es möglich, ABB-Testsignale zu streamen, allerdings

muss dieses Programm in einem separaten Task ausgeführt werden. Eine detaillierte Beschreibung befindet sich in Anhang A.2.

Auf der PC-Seite ist es notwendig, eine Echtzeitanwendung zu implementieren, die zyklisch die empfangenen Daten verarbeitet und neue Bewegungsbefehle an die Robotersteuerung sendet. Dazu wurde im Rahmen dieser Arbeit das C++-Projekt *egm_comm_src* entwickelt, das die Kommunikation über UDP ermöglicht. Die empfangenen Daten werden in Echtzeit verarbeitet und an Simulink übergeben, wo die zuvor beschriebenen Maßnahmen zur Genauigkeitssteigerung durchgeführt werden. Die Kommunikation zwischen der C++-Anwendung und Simulink erfolgt über einen Simulink-S-Function Block, der in Anhang A.3.2 näher beschrieben wird, während die Implementierung der C++-Anwendung in Anhang A.3.1 behandelt wird.

Um die Schwachstellen der UDP-Übertragung zu beheben, werden die Messdaten mit einem Zeitstempel und einer Sequenznummer versehen. Diese Informationen ermöglichen es der C++-Anwendung, die Reihenfolge der eintreffenden Datenpakete zu überprüfen. Sollte die Reihenfolge nicht korrekt sein, wird das fehlerhafte Datenpaket verworfen. Darüber hinaus kann die Anwendung durch die Überwachung der Sequenznummern und Zeitstempel einen möglichen Abriss des Datenstreams erkennen und auf einen Verbindungsverlust schließen. Im Gegensatz hat die Robotersteuerung selbst keine Möglichkeit, einen solchen Verbindungsverlust zu identifizieren.

In dieser Arbeit wird sowohl der EGM Position Stream als auch die EGM Position Guidance verwendet. Dies bedeutet, dass die aktuelle Position des Roboters kontinuierlich an den PC übertragen wird, der daraufhin neue Sollpositionen berechnet und an die Steuerung zurücksendet. Die Kommunikation und der Ablauf sind in Abbildung 6.5 schematisch dargestellt. Darüber hinaus lässt sich der gesamte Kommunikationsprozess auch in der Simulationsumgebung RobotStudio nachbilden. In diesem Fall kommuniziert die Simulink-Echtzeitanwendung über eine virtuelle Steuerung mit dem digitalen Zwilling der Anlage, der das reale Verhalten des Roboters simuliert.

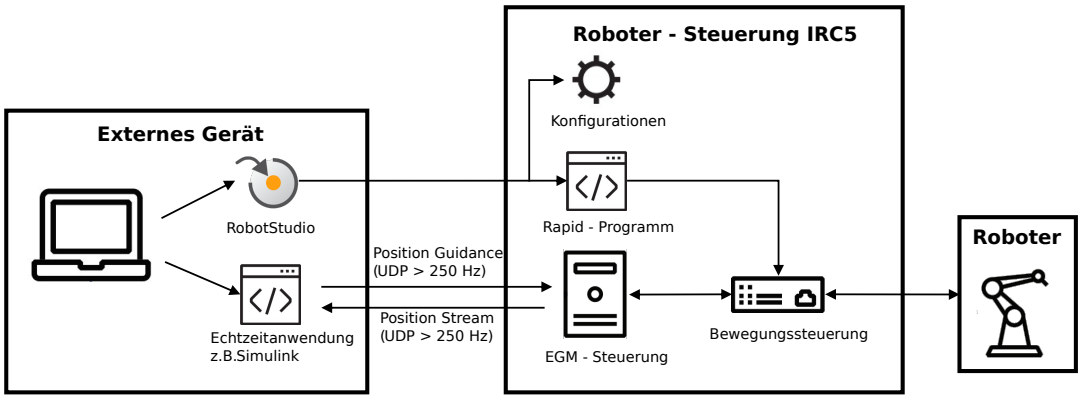


Abbildung 6.5: Aufbau der Kommunikation via EGM

Zur Überprüfung des Echtzeitverhaltens wurden Tests mit dem Simulink Realtime Pacer durchgeführt, bei denen eine vorgegebene Sollbahn in Gelenkwinkeln und Gelenkgeschwindigkeit über die EGM-Schnittstelle an den Roboter übertragen wurde. Dabei

wurden die Übertragungszeitpunkte der Sollwerte wie folgt erfasst: t_{sent} bezeichnet den Zeitpunkt, zu dem die Sollwerte laut Systemzeit des PCs an den Roboter übermittelt wurden, und t_{sim} den Zeitpunkt, zu dem dies laut Simulationszeit von Simulink erfolgt ist. Zusätzlich wurde die Systemzeit der Robotersteuerung t_{rob} erfasst, welche den Zeitpunkt angibt, zu dem die Messwerte vom Roboter an den PC zurückübermittelt wurden. Alle Zeiten wurden zu Beginn der Tests synchronisiert.

Aus diesen Daten wurden die Zeitdifferenzen $\Delta t_{sent} = t_{sent} - t_{sim}$ und $\Delta t_{rob} = t_{rob} - t_{sim}$ ermittelt, die in Abbildung 6.6 dargestellt sind.

Die Analyse des Verlaufs von Δt_{sent} zeigt, dass die Simulationszeit anfangs langsamer als die Systemzeit des PCs fortschreitet und später schneller wird, bis die Simulationszeit die Systemzeit bei Sekunde 1.092 erreicht und ab diesem Zeitpunkt ungefähr gleich verläuft. Diese Differenz lässt darauf schließen, dass Simulink anfangs Schwierigkeiten hat, die Berechnungen in Echtzeit durchzuführen und anstelle von Überschreibungen die Berechnungen in längeren Zeitschritten ausführt. Anschließend läuft die Simulation schneller als die Solltaktzeit und startet die nächste Berechnung sofort, was zu einem beschleunigten Fortschritt der Simulationszeit führt. Dieses Verhalten entspricht jedoch nicht dem Echtzeitstandard.

Der Verlauf von Δt_{rob} zeigt anfangs ein ähnliches Muster: Die Differenz steigt zunächst stark an und verringert sich dann allmählich, wobei ein Offset von etwa 80 ms bestehen bleibt. Um diesen Offset zeigt das Signal leichte, sägezahnartige Schwankungen, was darauf hinweist, dass die reale Taktzeit des EGM nicht exakt 4 ms beträgt, sondern leicht darüber liegt. Dadurch baut sich die Differenz auf, bis kein neues Signal zur Verfügung steht, woraufhin die Differenz abrupt abnimmt und der Zyklus erneut beginnt. Diese Schwankungen treten in einer Breite von 4 ms auf. Zusammenfassend lassen sich hier drei Einflüsse identifizieren: die Nicht-Einhaltung des Echtzeittakts, eine Übertragungsverzögerung von etwa 80 ms und eine leicht ungenaue Abstimmung der Übertragungsfrequenz.

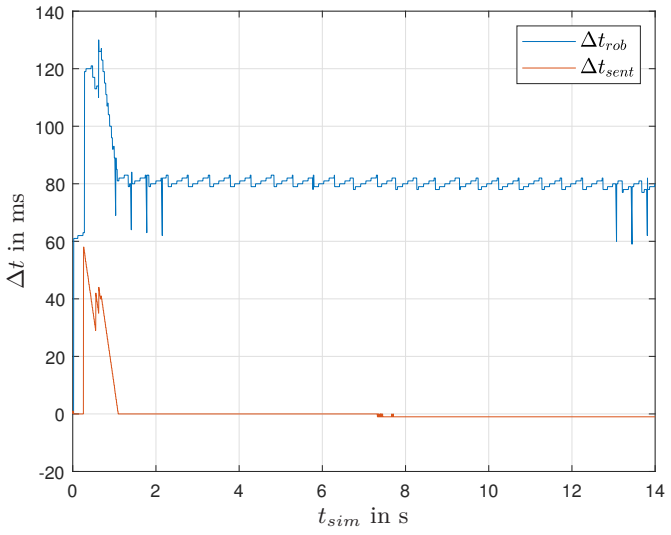


Abbildung 6.6: Unterschiede in der Zeitbasis

Zusätzlich wurde der Zeitunterschied zwischen der Sollbahn $\mathbf{q}_d$ und der tatsächlich abgefahrenen Bahn $\mathbf{q}_m$ durch zeitliche Verschiebung der Zeit-Werte-Paare ermittelt, sodass beide Verläufe bestmöglich übereinanderliegen. Dies ist in Abbildung 6.7 dargestellt und ergibt eine Totzeit von 140 ms.

Die Ergebnisse zeigen, dass die reale Totzeit die in [5] angegebenen Werte von 10 bis 20 ms überschreitet und dass der Simulink Realtime Pacer kein ideales Echtzeitverhalten bietet. Insgesamt führt dies zu einer Totzeit von etwa 140 ms.

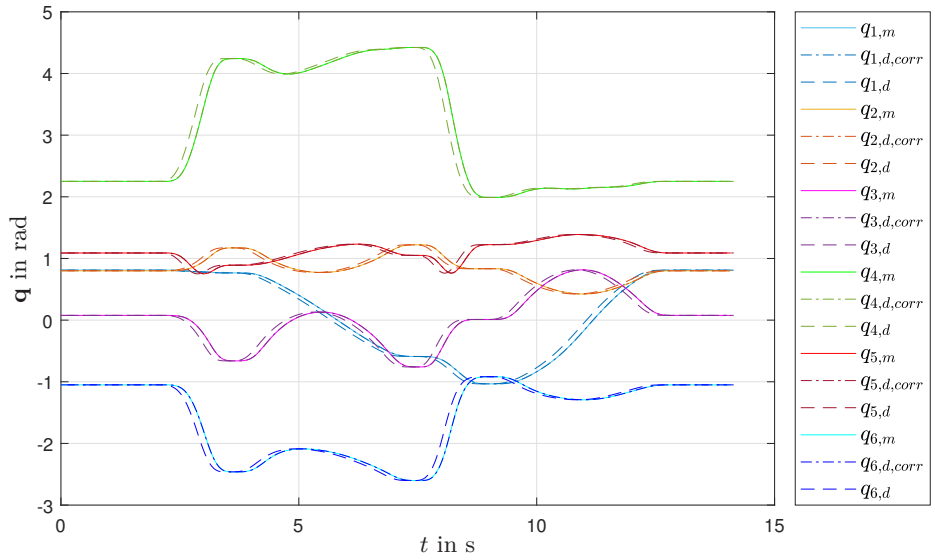


Abbildung 6.7: Ermittlung der Totzeit

6.4 Geometrische Kalibrierung

Das in Kapitel 4 beschriebene Konzept zur Steigerung der Positioniergenauigkeit eines Roboters durch die geometrische Kalibrierung wird im Folgenden zuerst anhand einer Simulation in RobotStudio und anschließend anhand des in Abschnitt 6.1 beschriebenen Versuchsaufbau getestet.

6.4.1 Simulation

Im ersten Schritt werden die Messdaten bestehend aus den in Abschnitt 4.2 beschriebenen optimalen Konfigurationen und den in Abschnitt 4.3 beschriebenen Validierungskonfiguration generiert. Dies geschieht indem die Konfigurationen über ein Rapid-Programm auf einer virtuellen Steuerung von dem in RobotStudio simulierten IRB1200 angefahren werden. Über EGM-Position Streaming lassen sich die simulierten EE-Posen an diesen Konfigurationen an Simulink übertragen, wo das Messdatenset erstellt wird. Die Messdaten werden künstlich mit den in Tabelle 6.1 angeführten Fehlern beaufschlagt. Diese Fehler müssen im Verlauf der Kalibrierung identifiziert werden.

Die Kalibrierungsdaten weisen hierbei die in Abbildung 6.8 dargestellten Fehler auf. Der Positionsfehler wird hierbei als der Abstand des Kinematikmodell (2.9) von der

gemessenen Position dargestellt und der Orientierungsfehler als die Winkeldifferenz nach (2.38) zwischen Kinematikmodell und gemessener Orientierung.

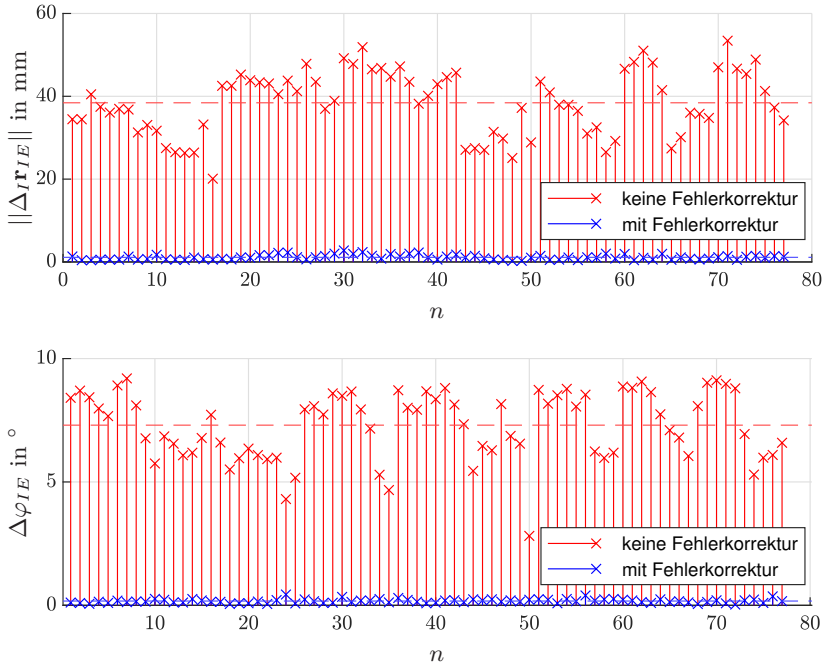


Abbildung 6.8: RobotStudio Simulation: Vergleich der Positions- und Orientierungsfehler anhand der simulierten Kalibrierungsdaten für das reguläre Kinematikmodell (*keine Fehlerkorrektur*) und das an diesen Daten kalibrierte Fehlermodell (*mit Fehlerkorrektur*)

Die Fehler zwischen dem Kinematikmodell und den Messergebnissen aus der Simulation, an den Validierungskonfigurationen sind in Abbildung 6.9 dargestellten. Die Fehler variieren hier in beiden Datensets sehr stark wobei der Mittelwert des Positionsfehlers der beiden Datensets in etwa gleich bei 37 mm liegt und der Mittelwert des Orientierungsfehlers bei 6.7° liegt. Beide Mittelwerte sind in Abbildung 6.9 und Abbildung 6.9 rot-strichliert eingezeichnet.

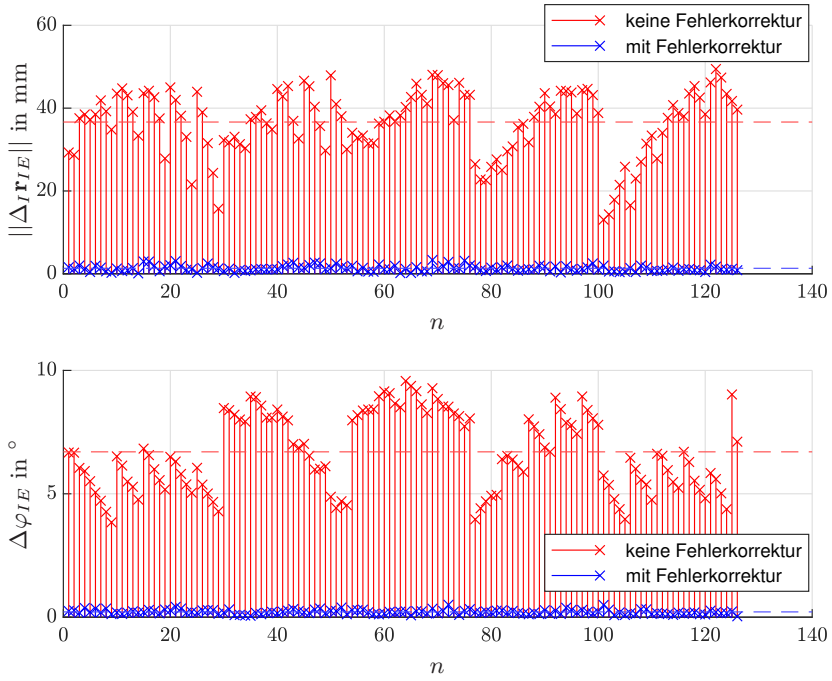


Abbildung 6.9: RobotStudio Simulation: Vergleich der Positions- und Orientierungsfehler anhand der simulierten Validierungsdaten für das reguläre Kinematikmodell (*keine Fehlerkorrektur*) und das kalibrierte Fehlermodell (*mit Fehlerkorrektur*)

Anhand der Kalibrierungsdaten wurden die Fehlerparameter des Fehlermodells (2.55) gemäß Abschnitt 4.1 identifiziert. Die resultierenden Fehler nach der Kalibrierung sind ebenfalls in Abbildung 6.8 und Abbildung 6.9 für die beiden Datensets dargestellt. Es lässt sich bereits erkennen, dass die Kalibrierung eine wesentliche Verkleinerung des Fehlers bewirkt hat. Vergleicht man die identifizierten Fehlerparameter mit den künstlich eingebrachten Fehlern nach Tabelle 6.1 in Abbildung 6.10 und Abbildung 6.12, so zeigt sich, dass diese mit einer Genauigkeit im Bereich von $1 \cdot 10^{-5} - 1.6 \cdot 10^{-3}$ ermittelt werden konnten. Dies spricht für eine hohe Qualität der Identifizierung und zeigt, dass der Identifikationsalgorithmus funktioniert. Die erreichbare Genauigkeit ist dabei von der Wahl des künstlich eingebrachten Rauschens abhängig, welches ebenfalls in Tabelle 6.1 angeführt wird. Ohne diesem Rauschen ist eine Genauigkeit im Bereich von $1.3 \cdot 10^{-10} - 3.6 \cdot 10^{-8}$ erzielbar, was Abbildung 6.11 zeigt.

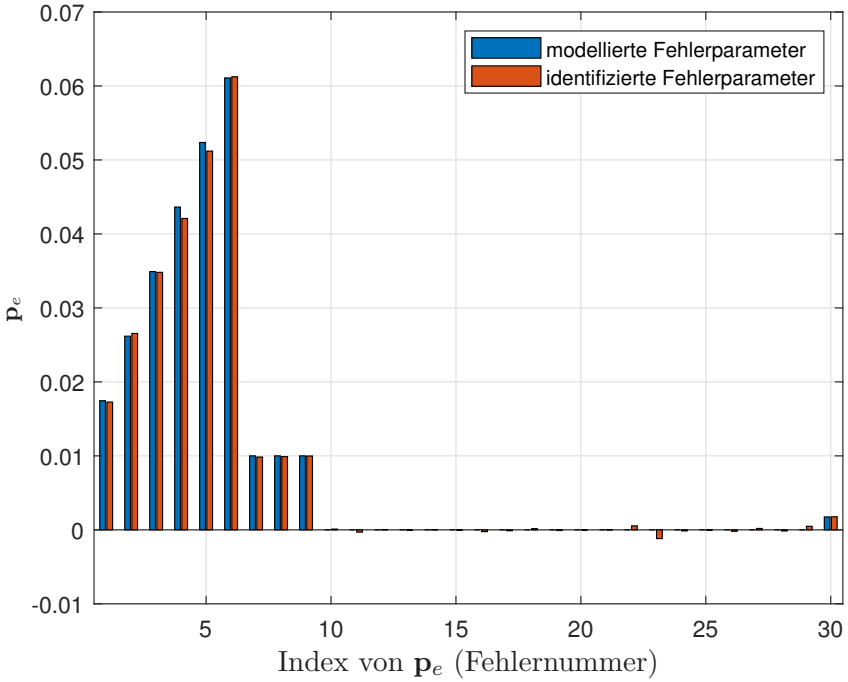


Abbildung 6.10: RobotStudio Simulation: Vergleich der identifizierten mit den in der Simulation vorgegeben Fehlerparametern

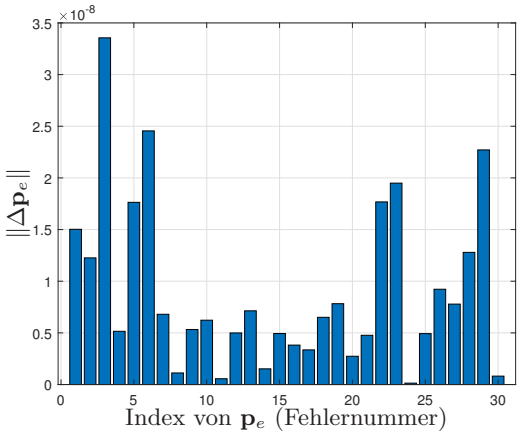


Abbildung 6.11: RobotStudio Simulation: Differenz zwischen modellierten und identifizierten Fehlerparametern ohne Rauschen

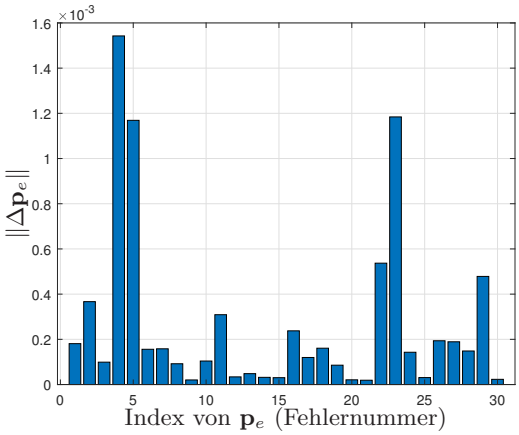


Abbildung 6.12: RobotStudio Simulation: Differenz zwischen modellierten und identifizierten Fehlerparametern mit Rauschen

Eine detaillierte Betrachtung des Positions- und des Orientierungsfehlers des kalibrierten Fehlermodells anhand der Validierungsdaten zeigen die Histogramme in Abbildung 6.15 und Abbildung 6.16. Wobei an den meisten der 125 Validierungskonfigurationen ein absoluter Positionsfehler im Bereich von $0.5 - 2\,\text{mm}$ auftritt und ein Orientierungsfehler im Bereich von $0.1 - 0.3°$. Tabelle 6.3 liefert hierbei eine detaillierte Zusammensetzung der Fehler. Wie bei der Betrachtung der Genauigkeit der Identifizierung der Fehlerparameter in Abbildung 6.11 sind auch hier die Werte der Positions-, oder Orientierungsfehler maßgeblich vom künstlich eingebrachten Rauschen abhängig. Ohne diesem tritt faktisch kein Fehler auf, mit Ausnahme minimaler Abweichungen aufgrund numerischer Ungenauigkeiten, was Abbildung 6.13 und Abbildung 6.14 ver-

deutlicht und dies somit für eine einwandfreie Funktion der beschrieben Algorithmen spricht.

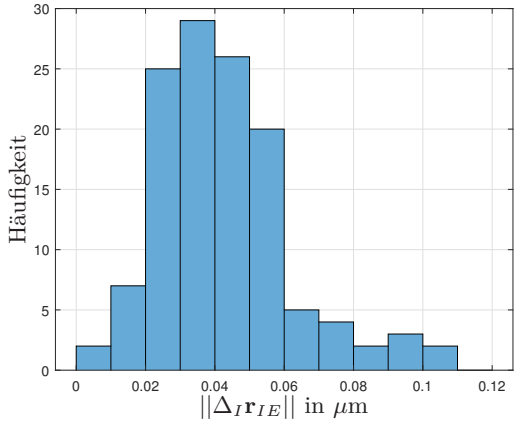


Abbildung 6.13: RobotStudio Simulation: Histogramm des Positionsfehlers anhand der simulierten Validierungsdaten ohne Rauschen

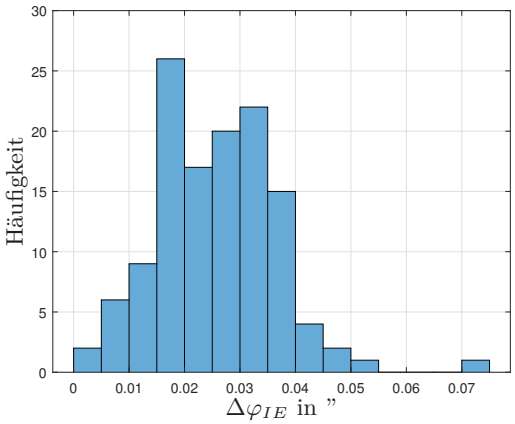


Abbildung 6.14: RobotStudio Simulation: Histogramm des Orientierungsfehlers anhand der simulierten Validierungsdaten ohne Rauschen

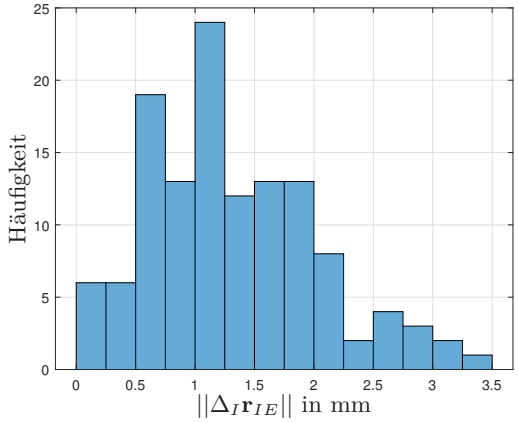


Abbildung 6.15: RobotStudio Simulation: Histogramm des Positionsfehlers anhand der simulierten Validierungsdaten

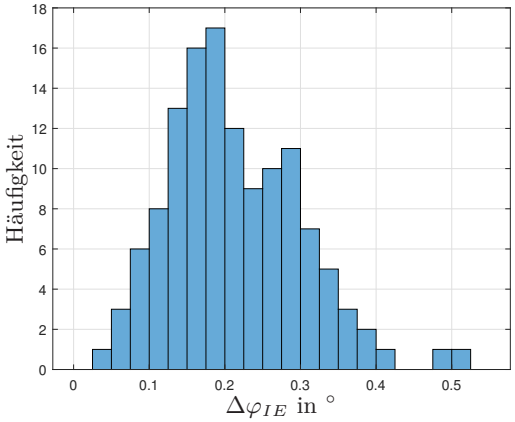


Abbildung 6.16: RobotStudio Simulation: Histogramm des Orientierungsfehlers anhand der simulierten Validierungsdaten

Tabelle 6.3: RobotStudio Simulation: Fehlerverhalten auf Basis des Validierungsdatensets

		Δx in mm	Δy in mm	Δz in mm	ρ_x in °	ρ_y in °	ρ_z in °
mittlerer absoluter Fehler	Kinematik-modell	11.83	14.62	26.07	2.81	3.69	3.46
	Fehler-modell	0.73	0.58	0.70	0.11	0.10	0.11
maximaler absoluter Fehler	Kinematik-modell	33.68	41.05	47.91	8.30	7.69	7.59
	Fehler-modell	2.16	2.28	2.94	0.42	0.40	0.36
90 % Grenze	Kinematik-modell	24.92	33.20	41.43	7.11	6.41	5.79
	Fehler-modell	1.41	1.20	1.57	0.23	0.21	0.21

Für den Fall, dass die Orientierung nicht genau genug gemessen werden kann, können wie in Abschnitt 4.1 beschrieben für die Identifikation lediglich die Positionsdaten verwendet werden. Dies führt für das selbe Datenset zu einem Positionsfehler im Bereich von $0-3.5\,\text{mm}$ dargestellt in Abbildung 6.17 und einen Orientierungsfehler im Bereich von $3.05-3.85°$ dargestellt in Abbildung 6.18. Es lässt sich somit erkennen, dass selbst für den simulierten Fall in dem die Orientierungsdaten ohne große Fehler gemessen werden können, die Identifikation ohne Orientierungen ziemlich gute Ergebnisse liefert.

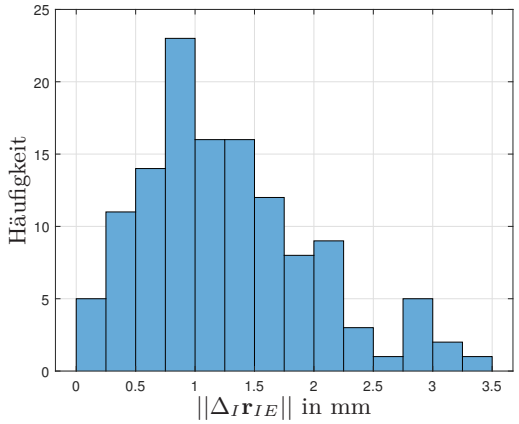


Abbildung 6.17: RobotStudio Simulation: Histogramm des Positionsfehlers anhand der simulierten Validierungsdaten bei Identifikation ohne Orientierung

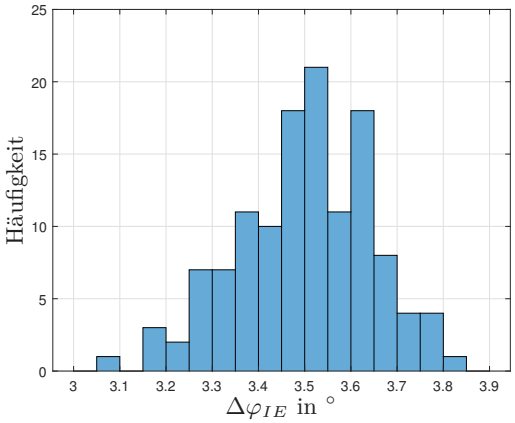


Abbildung 6.18: RobotStudio Simulation: Histogramm des Orientierungsfehlers anhand der simulierten Validierungsdaten bei Identifikation ohne Orientierung

Im nächsten Schritt wird das identifizierte Modell verwendet, um die Kompensation der geometrischen Fehler nach Abschnitt 4.4 für die Testbahn laut Abschnitt 5.1 vorzunehmen. Dies geschieht in der in Abschnitt 6.1 beschriebenen Simulationsumgebung. Dabei wurde die Verstärkung mit $\mathbf{K} = 2\,\mathbf{I}$ gewählt. Es ergibt sich der in Abbildung 6.19 dargestellte Positionsfehler und der in Abbildung 6.20 dargestellte Orientierungsfehler. Der Berechnung des Orientierungsfehlers erfolgt hier nach (2.37).

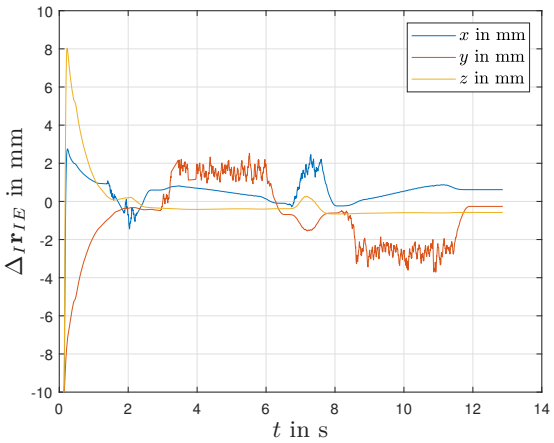


Abbildung 6.19: RobotStudio Simulation: Positionsfehler des kalibrierten Roboters

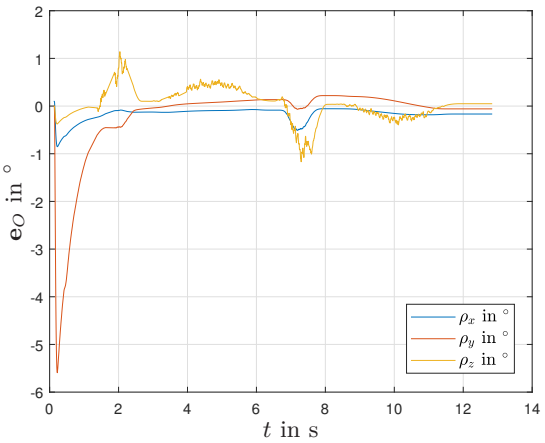


Abbildung 6.20: RobotStudio Simulation: Orientierungsfehler des kalibrierten Roboters

Als Vergleich ergibt sich in der Simulation mit den in Tabelle 6.1 dargestellten Fehlern, bei Steuerung des unkalibrierten Roboters der in Abbildung 6.21 gezeigte Positionsfehler, sowie der Orientierungsfehler in Abbildung 6.22. Es wird deutlich,

dass selbst relativ kleine geometrische Fehler zu erheblichen Abweichungen in der Endeffektorposition und -orientierung führen. Diese Abweichungen treten sowohl während des Bahnfahrens als auch beim Erreichen der Endkonfiguration auf.

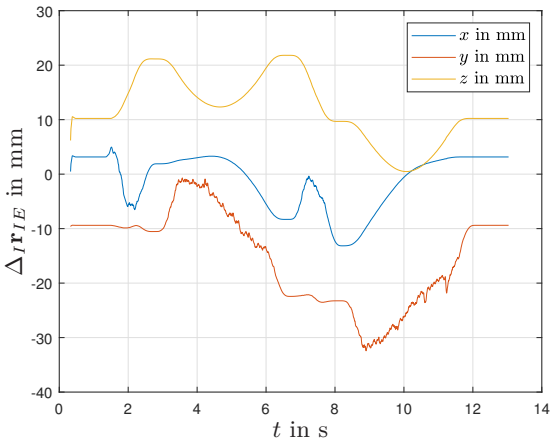


Abbildung 6.21: RobotStudio Simulation: Positionsfehler des unkalibrierten Roboters

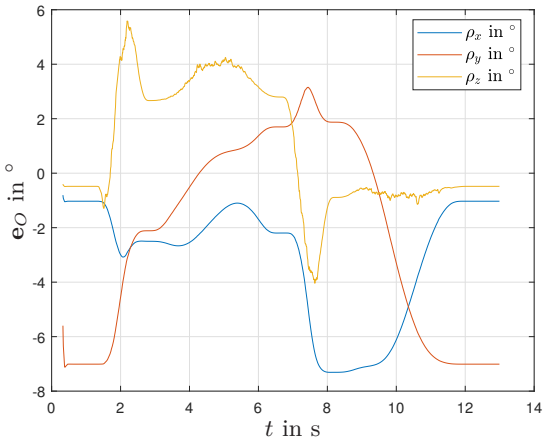


Abbildung 6.22: RobotStudio Simulation: Orientierungsfehler des unkalibrierten Roboters

Durch die Kalibrierung ließen sich diese Fehler signifikant reduzieren, insbesondere jener der Endkonfiguration. Der maximale Fehler während des Abfahrens der Bahn bleibt jedoch relativ hoch. Dieser Effekt lässt sich hauptsächlich auf zwei Ursachen zurückführen:

- **Geometrische vs. dynamische Fehler**: Bei der geometrischen Kalibrierung werden ausschließlich die geometrischen Fehler des Robotermodells identifiziert und korrigiert. Die dynamischen Fehler bleiben unkompensiert. Dies zeigt sich insbesondere daran, dass die statische Genauigkeit nach der Kalibrierung sehr gut ist, während die dynamische Genauigkeit, also während des Bewegens, weiterhin unter den dynamischen Einflüssen leidet.
- **Totzeiten und Regelung**: Wie bereits in Abschnitt 6.3 erwähnt, führen die bei der EGM-Kommunikation auftretenden Totzeiten zu Verzögerungen und Störungen in der Regelung. Darüber hinaus spielt die Einstellung des unterlagerten Regelkreises des positionsgeregelten Roboters eine wesentliche Rolle, da dieser die Stabilität und Genauigkeit der Bewegung beeinflusst. Diese lassen sich jedoch nur bedingt beeinflussen und sind nicht im Detail bekannt.

In Tabelle 6.4 sind die unterschiedlichen Fehlerwerte für den kalibrierten und unkalibrierten Roboter im Detail aufgelistet, was die Auswirkungen der Kalibrierung auf die Positions- und Orientierungsfehler nochmals verdeutlicht.

Tabelle 6.4: RobotStudio Simulation: Fehlerverhalten der Kompensation

		Δx in mm	Δy in mm	Δz in mm	ρ_x in °	ρ_y in °	ρ_z in °
maximaler absoluter Fehler	unkalibriert	13.16	32.42	21.81	7.31	7.01	5.59
	kalibriert	2.45	3.71	0.66	0.51	0.46	1.14
statische Genauigkeit	unkalibriert	3.15	9.39	10.23	1.03	7.01	0.48
	kalibriert	0.62	0.26	0.58	0.17	0.06	0.05

6.4.2 Versuch am realen System

Nach der erfolgreichen Simulation werden die Tests unter realen Bedingungen mit dem in Abschnitt 6.1 beschriebenen Versuchsaufbau wiederholt. Anstelle der Simulation des Kamerasystems wird nun das reale Qualisys-System verwendet, und die Kommunikation erfolgt direkt mit der Robotersteuerung. Bei der Kalibrierung des Kamerasystems konnte eine Genauigkeit von 0.880 31 mm erzielt werden.

Im ersten Schritt werden, wie bereits in der Simulation, die in Abschnitt 4.2 beschriebenen optimalen Roboterkonfigurationen mittels eines Rapid-Programms angefahren. Die genaue Vermessung der Roboterpose erfolgt hierbei über das Qualisys-Motion-Capturing-System. Gleichzeitig werden die über EGM übermittelten Gelenkwinkelwerte aufgezeichnet. Beide Messungen werden synchronisiert. Sobald der Roboter die jeweilige optimale Konfiguration erreicht hat und die Schwingungen nach einer Abklingzeit von 0.5 s abgeklungen sind, werden die übermittelte Pose und die Gelenkwinkel als Messpunkt in das Datenset aufgenommen. Nach selbiger Vorgehensweise werden das Validierungsdatenset ermittelt für die Validierungskonfigurationen nach Abschnitt 4.3.

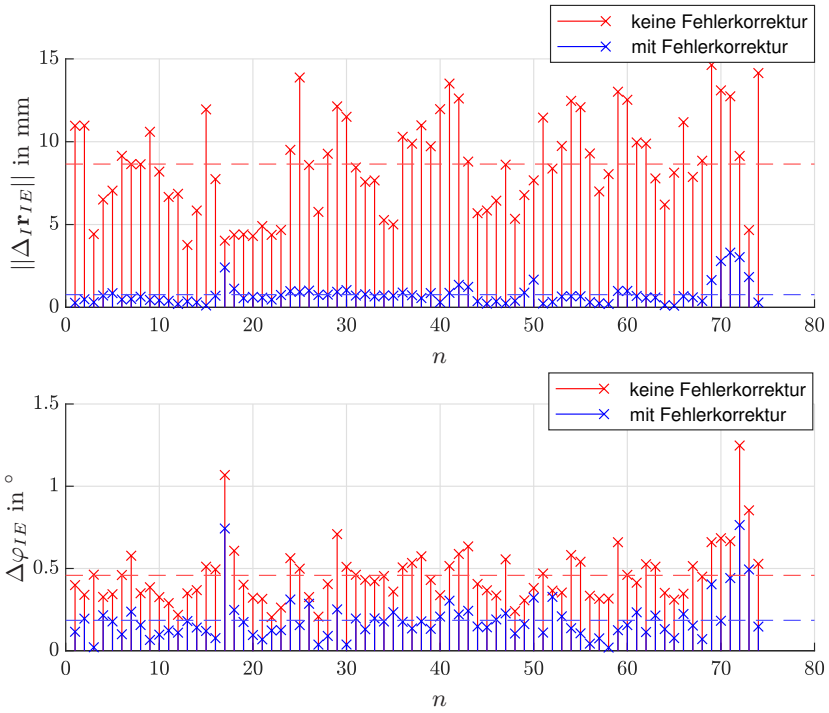


Abbildung 6.23: Experiment: Vergleich der Positions- und Orientierungsfehler anhand der Kalibrierungsdaten für das reguläre Kinematikmodell (*keine Fehlerkorrektur*) und das an diesen Daten kalibrierte Fehlermodell (*mit Fehlerkorrektur*)

Der Fehler zwischen dem Kinematikmodell (2.9) und der gemessenen Pose, ist in Abbildung 6.23 dargestellt, wobei die Berechnung der Winkeldifferenz nach (2.38) erfolgt. Dieser ist mit einem mittleren Positionsfehler von 8.65 mm und einen mittleren Orientierungsfehler von 0.46° (gekennzeichnet durch rot-strichlierte Linien) für industrielle Zwecke sehr groß. Im Vergleich sind die resultierenden Fehler jedoch kleiner

als jene Fehler die sich aus der Simulation in Abschnitt 6.4.1 ergeben, bei welcher Werte für die geometrischen Fehler angenommen wurden. Außerdem schwanken die Fehler sehr stark mit einem maximale Ausschlag um den Mittelwert von 5.98 mm in der Position und von 0.79° in der Orientierung. Sehr ähnlich verhält es mit den aus den Validierungsdaten ergebenden Fehlern, die in Abbildung 6.24 dargestellt sind.

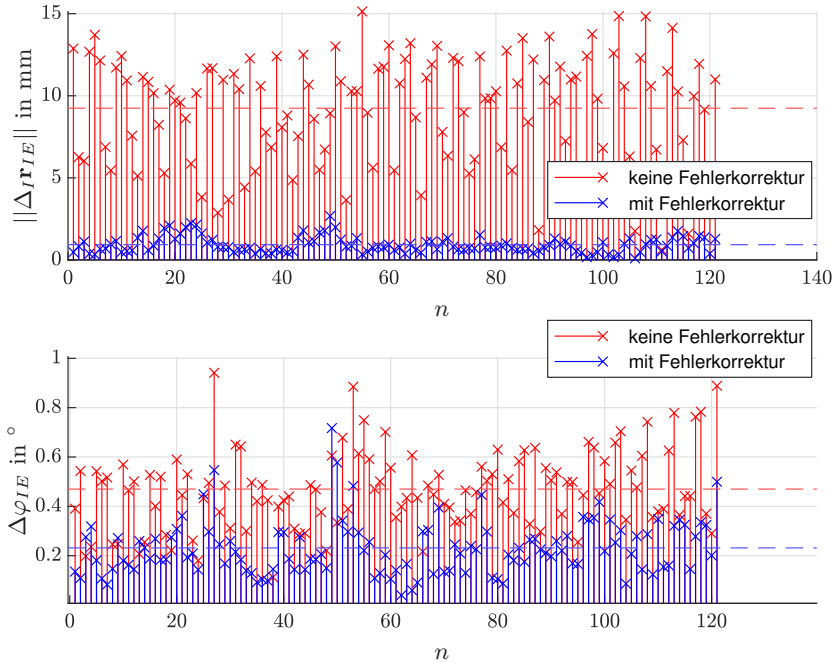


Abbildung 6.24: Experiment: Vergleich der Positions- und Orientierungsfehler anhand der Validierungsdaten für das reguläre Kinematikmodell (*keine Fehlerkorrektur*) und das kalibrierte Fehlermodell (*mit Fehlerkorrektur*)

Mit den aufgenommenen Kalibrierungsdaten wird anschließend das in Abschnitt 2.3 beschriebene Fehlermodell identifiziert. Die Fehler des identifizierten Modells sind ebenfalls in Abbildung 6.23 und Abbildung 6.24 dargestellt. Es lässt sich hier erkennen, dass vor allem in der Position eine enorme Verbesserung der Genauigkeit erzielt wird. Hier konnte der mittlere Fehler der Validierungskonfigurationen von 9.25 mm und 0.47° auf 0.93 mm und 0.23° gesenkt werden. In der Position somit in etwa auf ein Zehntel reduziert und in der Orientierung um ca. die Hälfte. Betrachtet man die Verteilung der Fehler in den Histogrammen Abbildung 6.25 und Abbildung 6.26 so zeigt sich, dass die meisten Positionsfehler des identifizierten Modells kleiner als 1.5 mm sind und die meisten Orientierungsfehler kleiner als 0.4°. Diese recht guten Ergebnisse sind jedoch noch weit von den geplantem Ziel von einer Genauigkeit im Zehntelmillimeter Bereich entfernt. Es besteht jedoch großes Potential indem man die Genauigkeit der Detektion der Endeffektorpose durch bessere Kamerasysteme und effizientere Algorithmen steigert.

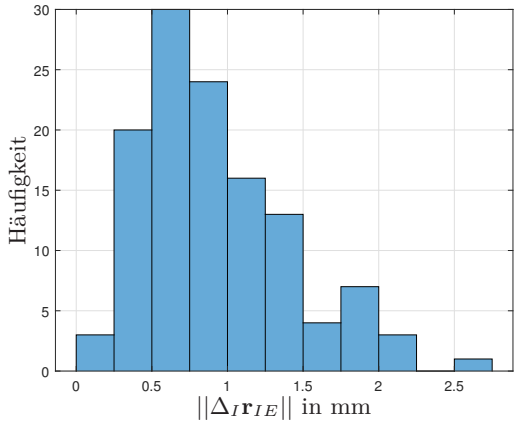


Abbildung 6.25: Experiment: Histogramm des Positionsfehlers anhand der Validierungsdaten

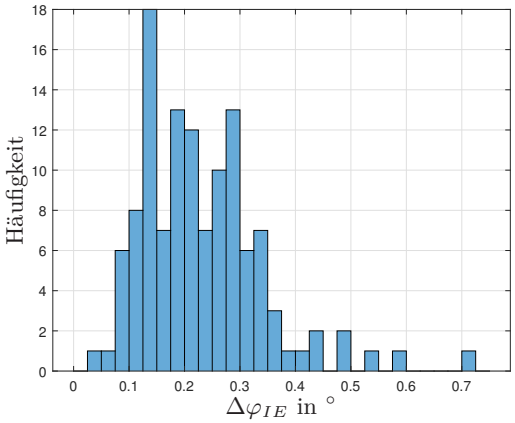


Abbildung 6.26: Experiment: Histogramm des Orientierungsfehlers anhand der Validierungsdaten

Die aus der Identifikation gewonnenen Fehlerparameter sind in Abbildung 6.27 dargestellt. Im Vergleich zu den Ergebnissen aus der Simulation sind die identifizierten Parameter signifikant geringer. Ein auffälliger Punkt ist der Unterschied zwischen der Identifikation der Parametern mit und ohne Berücksichtigung der Orientierung. Während die Fehlerparameter ohne die Verwendung der Orientierungsdaten stark von denen mit Orientierungsdaten abweichen, bleibt die Genauigkeit der Position nahezu unverändert. Die Genauigkeit der Orientierung konnte mit der Methode ohne Orientierungsdaten jedoch im Gegensatz zur Methode mit Orientierungsdaten nicht signifikant verbessert werden, weist jedoch auch keine Verschlechterung im Vergleich zum unkalibrierten Modell auf. Die Verteilung des Positionsfehlers und des Orientierungsfehlers bei der Identifikation ohne Orientierungen sind in den Histogrammen Abbildung 6.28 und Abbildung 6.29 dargestellt, zusätzlich sind die auftretenden Fehler noch in Tabelle 6.5 zusammengefasst.

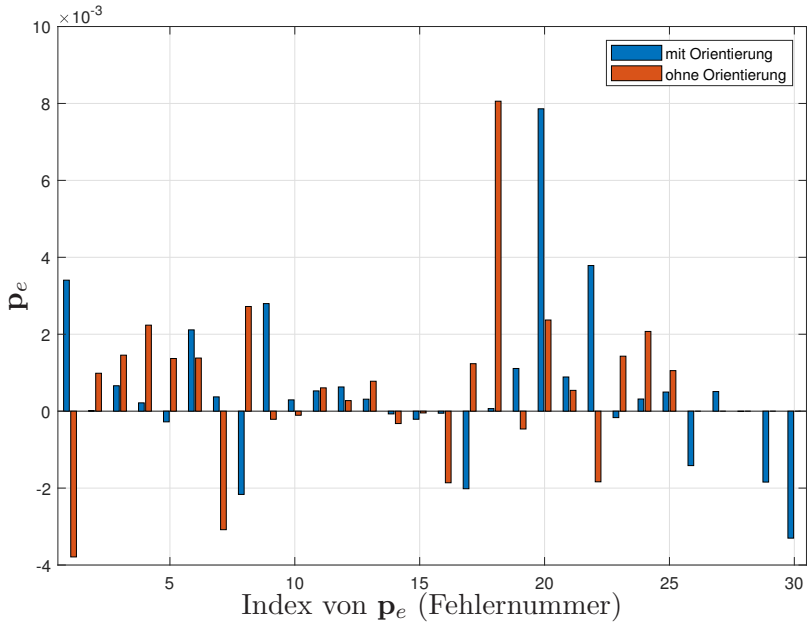


Abbildung 6.27: Experiment: Vergleich der Fehlerparameter bei Identifizierung mit und ohne Orientierung

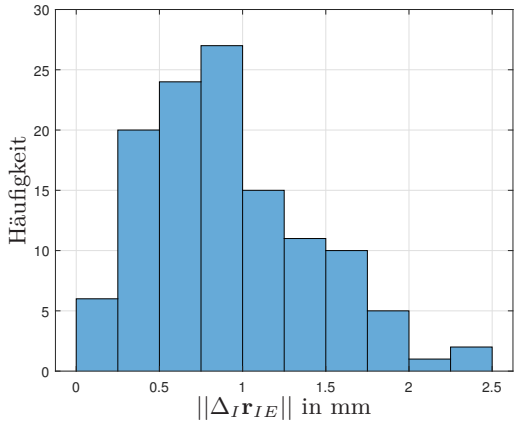


Abbildung 6.28: Experiment: Histogramm des Positionsfehlers anhand der Validierungsdaten bei Identifikation ohne Orientierung

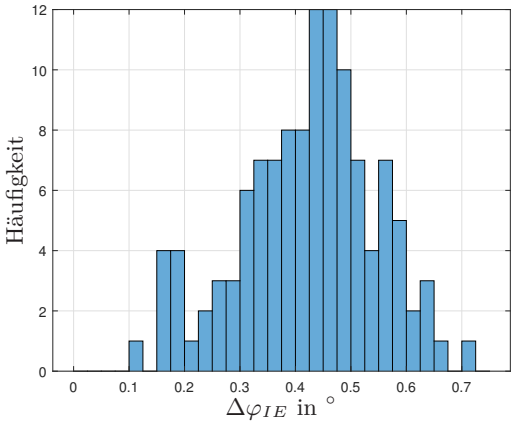


Abbildung 6.29: Experiment: Histogramm des Orientierungsfehlers anhand der Validierungsdaten bei Identifikation ohne Orientierung

Tabelle 6.5: Experiment: Fehlerverhalten auf Basis des Validierungsdatensets

		Δx in mm	Δy in mm	Δz in mm	ρ_x in °	ρ_y in °	ρ_z in °
mittlerer absoluter Fehler	Kinematikmodell	4.39	4.64	4.60	0.17	0.21	0.31
	Fehlermodell	0.44	0.58	0.38	0.08	0.10	0.15
maximaler absoluter Fehler	Kinematikmodell	13.10	12.95	12.63	0.57	0.61	0.89
	Fehlermodell	1.64	2.28	1.56	0.49	0.40	0.54
90 % Grenze	Kinematikmodell	9.63	9.77	9.78	0.32	0.40	0.58
	Fehlermodell	1.10	1.18	0.91	0.19	0.22	0.30

Im Anschluss wird das identifizierte Modell zur Kompensation der in Abschnitt 5.1 beschrieben Testbahn angewendet. Hierbei kommt die in Abschnitt 4.4 beschriebene Methode zur Kompensation der geometrischen Fehler zum Einsatz, wobei die Verstärkung $\mathbf{K} = 100\,\mathbf{I}$ gewählt wurde. Der resultierende Positionsfehler und der Orientierungsfehler nach (2.37) sind in Abbildung 6.30 und Abbildung 6.31 dargestellt.

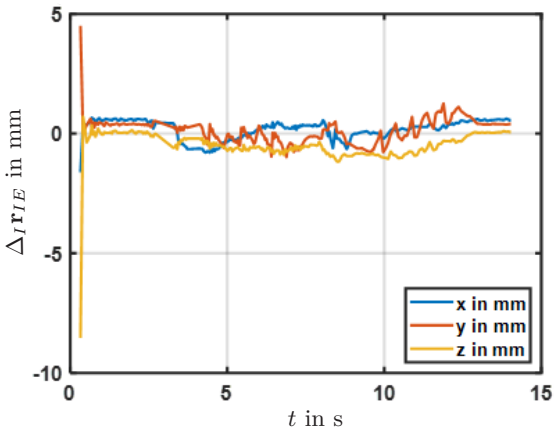


Abbildung 6.30: Experiment: Positionsfehler des kalibrierten Roboters

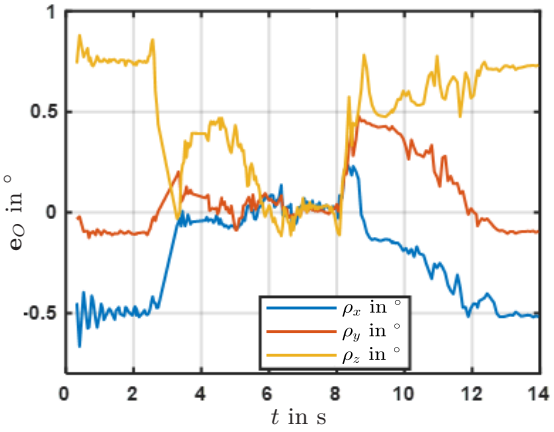


Abbildung 6.31: Experiment: Orientierungsfehler des kalibrierten Roboters

Zum Vergleich wurde dieselbe Testbahn auch ohne Kompensation abgefahren. Die entsprechenden Fehlerverläufe sind in Abbildung 6.32 und Abbildung 6.33 zu sehen. Es lässt sich erkennen, dass durch die Kompensation der maximale Positionsfehler beim Abfahren der Testbahn von 9.82 mm auf 1.36 mm reduziert werden konnte. Dieser Fehler ist sogar geringer als der maximale Fehler, der bei der statischen Messung der Validierungskonfigurationen ermittelt wurde. Der Grund dafür liegt darin, dass das Validierungsdatenset einige Ausreißer enthält.

Der maximale Orientierungsfehler konnte jedoch nur geringfügig verbessert werden – von 1.06° auf 0.89° – und bleibt damit schlechter als der maximale Orientierungsfehler der Validierungsdaten. Dies ist auf zwei Hauptursachen zurückzuführen: Erstens gibt es bei bestimmten Konfigurationen Probleme bei der exakten Berechnung der Orientierung aus den von Qualisys erfassten Markern. Zweitens gibt es, wie in Abschnitt 6.3 beschrieben, Herausforderungen bei der Kommunikation über EGM, die zu Ungenauigkeiten führen.

Außerdem führt sowohl für den maximalen Orientierungsfehler als auch für den maximalen Positionsfehler, wie bereits erwähnt, die geometrische Kalibrierung nur zu einer Verbesserung der geometrischen Fehler. Es treten jedoch zusätzlich auch dynamische Fehler auf, welche unkompensiert bleiben und somit Einfluss auf das dynamische Verhalten haben. Dies zeigt sich vorallem bei dem Vergleich mit der statischen Genauigkeit, jene Genauigkeit mit der der Endpunkt der Bahn angefahren wird. Diese hat sich vom Betrag von 9.84 mm auf 0.69 mm verbessert und liegt somit weitaus näher beim Ziel.

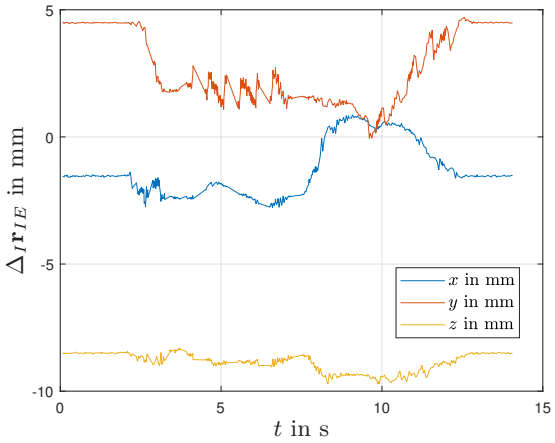


Abbildung 6.32: Experiment: Positionsfehler des unkalibrierten Roboters

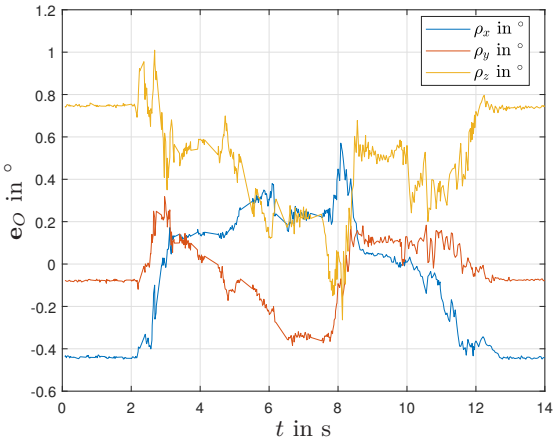


Abbildung 6.33: Experiment: Orientierungsfehler des unkalibrierten Roboters

Die resultierenden Fehler aus der Kompensation sind noch einmal detailliert in Tabelle 6.6 angeführt. Vergleicht man dieses Verhalten des realen Systems mit der Simulation (Tabelle 6.4), wird deutlich, dass am realen System eine höhere Genauigkeit erreicht wurde. Dies liegt zum einen daran, dass die Fehler in der Simulation nur auf Annahmen und Schätzungen beruhen. Zum anderen erreicht der verwendete PC bei der Simulation seine Leistungsgrenzen, was die Echtzeitkommunikation teilweise verfälscht und zu schlechteren Ergebnissen führt.

Tabelle 6.6: Experiment: Fehlerverhalten der Kompensation

		Δx in mm	Δy in mm	Δz in mm	ρ_x in °	ρ_y in °	ρ_z in °
maximaler absoluter Fehler	unkalibriert	3.15	4.86	9.82	0.61	0.36	1.06
	kalibriert	1.19	1.25	1.36	0.67	0.49	0.89
statische Genauigkeit	unkalibriert	2.85	4.75	8.13	0.46	0.11	0.79
	kalibriert	0.55	0.38	0.15	0.51	0.15	0.78

6.5 Regelung

Die in Kapitel 5 beschriebenen Konzepte zur Steigerung der Positioniergenauigkeit eines Roboters durch kamerabasierte Regelungsmethoden wird im Folgenden zuerst anhand einer Simulation in RobotStudio und anschließend anhand des in Abschnitt 6.1 beschriebenen Versuchsaufbau getestet.

6.5.1 Simulation

Zuerst werden in der in Abschnitt 6.1 beschriebenen Simulationsumgebung die Regelkonzepte getestet. Die aktuelle Pose des in RobotStudio simulierten Roboters wird hierbei von der virtuellen Robotersteuerung über EGM an die Simulink Echtzeitanwendung übertragen wo die Daten zur Simulation der Bilderfassung dienen und mit den Fehlern dargestellt in Tabelle 6.1 beaufschlagt werden. Über das Regelgesetz basierend auf einem der 3 Regelkonzepte wird die Stellgröße ermittelt und an die virtuelle Steuerung übertragen.

Angefangen wird mit dem ersten Konzept nach Abschnitt 5.4, bei dem in der Simulation der in Abbildung 6.34 dargestellte Positionsfehler und der in Abbildung 6.35 dargestellte Orientierungsfehler, basierend auf (2.37) erzielt wurde. Hierbei wurde die Verstärkung variiert, um das optimale Ergebnis hinsichtlich des kleinsten Fehlers zu erzielen, wobei sich $\mathbf{K} = 7\,\mathbf{I}$ als beste Wahl herausstellte.

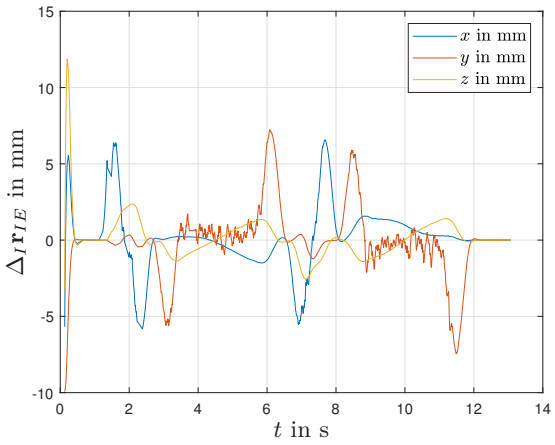


Abbildung 6.34: RobotStudio Simulation: Positionsfehler des Regelungskonzepts 1

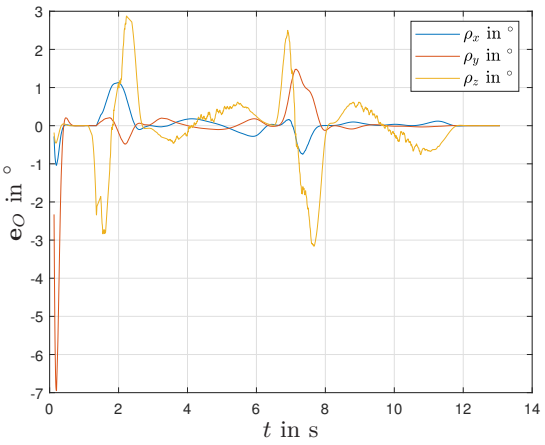


Abbildung 6.35: RobotStudio Simulation: Orientierungsfehler Regelungskonzepts 1

Anschließend wurde das zweite Konzept beschrieben in Abschnitt 5.5 simuliert. Auch hier wurde die Verstärkung angepasst, wobei das beste Ergebnis bei $\mathbf{K} = 5\,\mathbf{I}$ erzielt wurde. Die resultierenden Fehler sind in Abbildung 6.36 für die Position und Abbildung 6.37 für die Orientierung dargestellt.

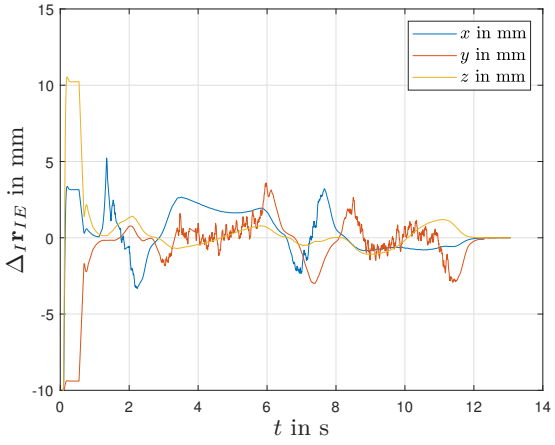


Abbildung 6.36: RobotStudio Simulation: Positionsfehler des Regelungskonzepts 2

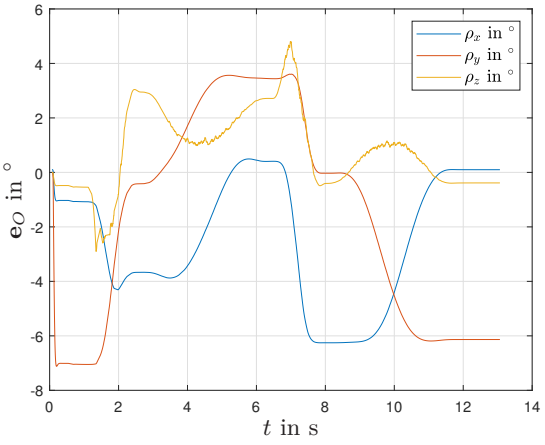


Abbildung 6.37: RobotStudio Simulation: Orientierungsfehler Regelungskonzepts 2

Zum Abschluss wurde das dritte Konzept nach Abschnitt 5.6 simuliert. Das optimale Ergebnis wurde mit $\mathbf{K} = 4\,\mathbf{I}$ erzielt, das in den Fehlerkurven in Abbildung 6.38 für die Position und Abbildung 6.39 für die Orientierung zu sehen ist.

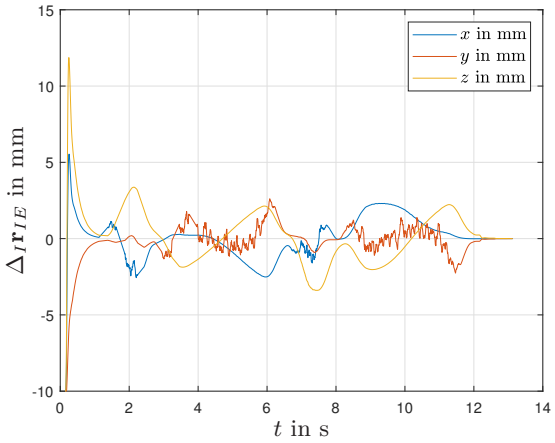


Abbildung 6.38: RobotStudio Simulation: Positionsfehler des Regelungskonzepts 3

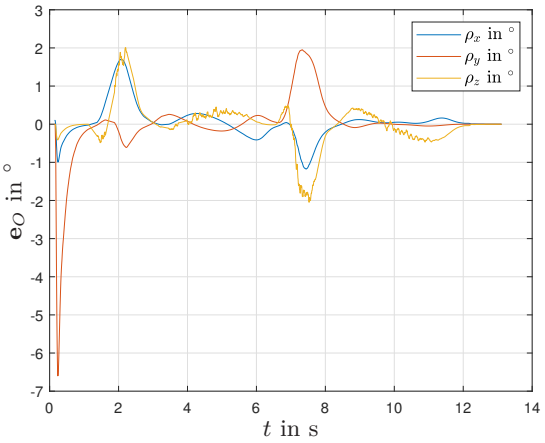


Abbildung 6.39: RobotStudio Simulation: Orientierungsfehler Regelungskonzepts 3

Um den Aufwand bei der Variierung der Verstärkung zu minimieren, wurden alle Verstärkungsgewichte gleich gewählt ($\mathbf{K} = k\,\mathbf{I}$). Durch differenzierte Gewichtungen könnten jedoch unter Umständen noch bessere Ergebnisse erreicht werden.

Die Ergebnisse der drei Regelungskonzepte werden mit den Resultaten der Steuerung des Roboters auf die Solltrajektorie, die in der Simulation der geometrischen Fehlerkompensation in Abschnitt 6.4.1 bereits simuliert wurde, verglichen. Die resultierenden Positionsfehler sind in Abbildung 6.21 und die Orientierungsfehler in Abbildung 6.22 dargestellt.

Es fällt sofort auf, dass sich andere Fehlerverläufe wie in Kapitel 5 ergeben, wo auf das dynamische Modell zurückgegriffen wird. Die starken Abweichungen lassen sich dadurch erklären, dass in Kapitel 5 die Regelstruktur angenommen werden musste, da ABB keine genauen Informationen über die in der EGM-Steuerung implementierte Regelung herausgibt. Zudem ist das dynamische Modell nicht exakt, da viele Parameter entweder geschätzt oder aus Datenblättern entnommen wurden, ohne einer präzisen Identifikation (ähnlich der der geometrischen Parameter) zu unterliegen. Hinzu kommen Übertragungsfehler und Totzeiten durch die Kommunikation über die EGM-Schnittstelle. Trotz dieser Unterschiede lassen sich Gemeinsamkeiten in den Fehlerverläufen erkennen, insbesondere an den Stellen, an denen Spitzen auftreten. In der Simulation mit RobotStudio fallen diese jedoch deutlich stärker aus.

Vergleicht man die Fehlerverläufe der Regelungskonzepte mit der Steuerung, so zeigt sich, dass das Konzept 3 hinsichtlich der maximalen Fehler die beste Performance aufweist. Der Positionsfehler konnte von 32.42 mm auf 3.4 mm und der Orientierungsfehler von 7.31° auf 2.05° reduziert werden. Konzept 2 schneidet hingegen am schlechtesten in der Orientierung ab, mit einem maximalen Fehler von 7.05°, während Konzept 1 mit 7.24 mm den höchsten Positionsfehler aufweist.

Vergleicht man diese Ergebnisse mit der Simulation der Kompensation der geometrischen Fehler (Abbildung 6.21 und Abbildung 6.22), so zeigt sich ein minimal schlechteres dynamisches Verhalten für Konzept 1 und ein deutlich schlechteres Verhalten für die anderen Konzepte. Dies lässt sich auf die Totzeiten in der Kommunikation

zurückführen, da der PC an seine Leistungsgrenzen stößt, da er sowohl die Regelung als auch das RobotStudio-Modell simuliert. Zusätzlich führen die bereits erwähnten Schwächen der EGM-Schnittstelle zu weiteren Abweichungen.

Besonders auffällig ist die ausgezeichnete Performance in der statischen Genauigkeit. Alle drei Konzepte erreichen eine Positionsgenauigkeit von unter 0.02 mm, was die Ergebnisse der simulierten Kalibrierung deutlich übertrifft. Wie bereits in Abschnitt 5.5 festgestellt, schafft es Regelkonzept 2 aber nicht, die Orientierung erfolgreich auszuregeln. Im Gegensatz dazu erreichen die anderen Konzepte eine sehr gute Performance, mit einem statischen Orientierungsfehler von unter 0.01°. Sowohl in Bezug auf die Position als auch die Orientierung übertreffen diese Konzepte somit die Leistung der geometrischen Kalibrierung.

In Tabelle 6.7 ist das Fehlerverhalten der drei Regelungskonzepte und der Steuerung zum Vergleich zusammengefasst.

Tabelle 6.7: RobotStudio Simulation: Vergleich des simulierten Fehlerverhaltens der kamerabasierten Regelungen

		Δx in mm	Δy in mm	Δy in mm	ρ_x in °	ρ_y in °	ρ_z in °
maximaler absoluter Fehler	Steuerung	13.16	32.42	21.81	7.31	7.01	5.59
	Regelungskonzept 1	6.57	7.24	2.61	1.13	1.48	2.87
	Regelungskonzept 2	5.23	3.60	1.42	6.25	7.05	4.80
	Regelungskonzept 3	2.56	2.61	3.40	1.70	1.95	2.05
statische Genauigkeit	Steuerung	3.15	9.39	10.23	1.03	7.01	0.48
	Regelungskonzept 1	0.01	0.01	0.01	0.01	0.01	0.01
	Regelungskonzept 2	0.01	0.02	0.01	0.10	6.14	0.39
	Regelungskonzept 3	0.01	0.01	0.01	0.01	0.01	0.01

6.5.2 Versuch am realen System

Nach Abschluss der Simulation werden die Versuche mit dem in Abschnitt 6.1 beschriebenen Versuchsaufbau am realen System durchgeführt. Anstelle der Simulation des Kamerasystems wird nun das tatsächliche Motion-Capturing-System Qualisys zur Erfassung der Endeffektorpose des Roboters eingesetzt. Die Vorgabe der Stellgröße erfolgt direkt mit der realen Robotersteuerung, anstelle der virtuellen Steuerung in der Simulation. Bei der Kalibrierung des Qualisys-Systems konnte eine Positionsgenauigkeit von 0.533 41 mm erzielt werden.

Zuerst wird das Regelungskonzept nach Abschnitt 5.4, basierend auf der numerischen Inverskinematik getestet. Hierbei wird für die Verstärkung $\mathbf{K} = 7\,\mathbf{I}$ gewählt. Die resultierenden Fehlerverläufe in Bezug auf die Position und Orientierung sind in Abbildung 6.40 und Abbildung 6.41 dargestellt. Hierbei und nachfolgend wurde für die Berechnung der Orientierungsabweichung nach (2.37) vorgegangen.

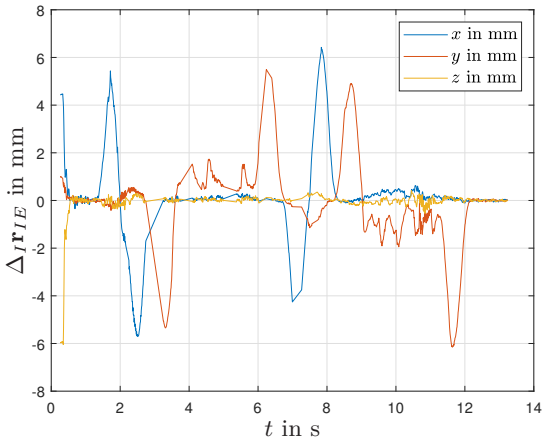


Abbildung 6.40: Experiment: Positionsfehler Regelungskonzept 1

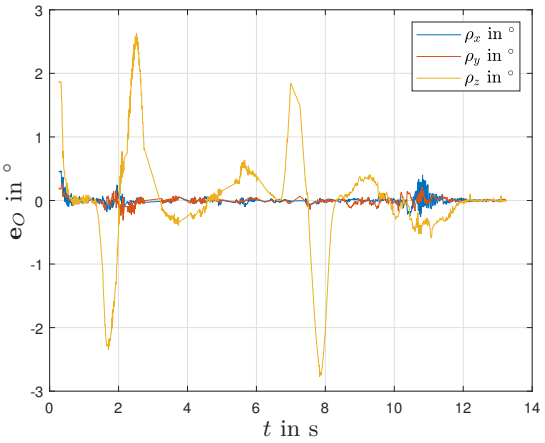


Abbildung 6.41: Experiment: Orientierungsfehler Regelungskonzept 1

Im Anschluss daran folgt die Prüfung des in Abschnitt 5.5 beschriebenen Regelungskonzepts, basierend auf Position-based Visual Servoing. Die Verstärkung wird hier $\mathbf{K} = 5\,\mathbf{I}$ gewählt. Die Fehlerverläufe für Position und Orientierung sind in Abbildung 6.42 und Abbildung 6.43 aufgeführt.

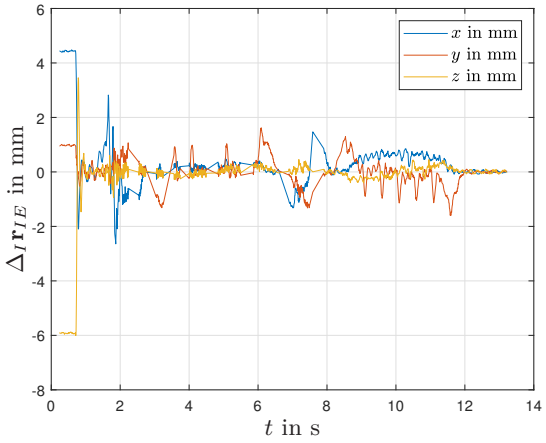


Abbildung 6.42: Experiment: Positionsfehler Regelungskonzept 2

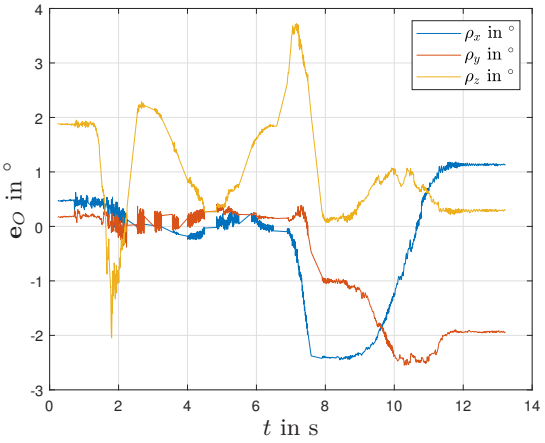


Abbildung 6.43: Experiment: Orientierungsfehler Regelungskonzept 2

Zum Schluss wird das dritte Regelkonzept, welches in Abschnitt 5.6 beschrieben wird und auf einem Fehlersystem auf Positionsebene basiert, mit einer Verstärkung von $\mathbf{K} = 4\,\mathbf{I}$ getestet. Für dieses Konzept ergeben sich der Positionsfehler in Abbildung 6.44 und der Orientierungsfehler in Abbildung 6.45.

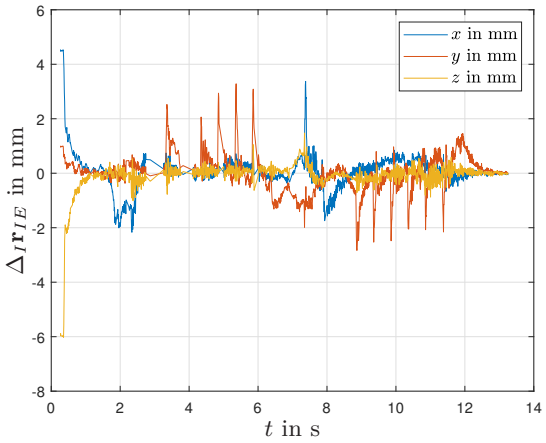


Abbildung 6.44: Experiment: Positionsfehler Regelungskonzept 3

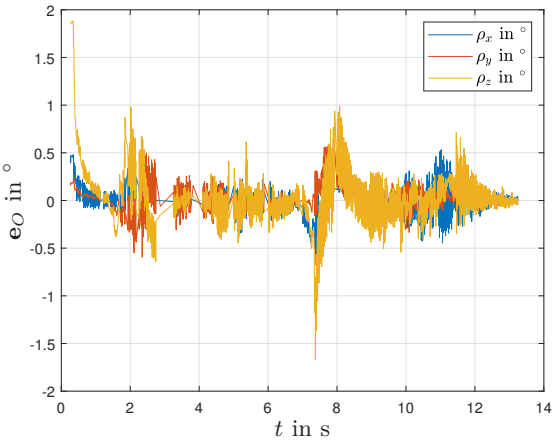


Abbildung 6.45: Experiment: Orientierungsfehler Regelungskonzept 3

Zusätzlich wird zum Vergleich eine einfache Steuerung auf die geplante Test-Trajektorie durchgeführt. Hier werden die Gelenkwinkel, welche sich aus der Bahnplanung nach Abschnitt 5.1 ergeben, direkt über EGM an den Roboter übermittelt, welcher diese anfährt. Gleichzeitig wird über das Motion-Capturing System die Pose erfasst. Das resultierende Fehlerverhalten aus gemessener Pose und Soll-Pose nach (2.36) ist in Abbildung 6.46 und in Abbildung 6.47 dargestellt und dient als Referenz, um die Wirksamkeit der Regelungskonzepte zu evaluieren.

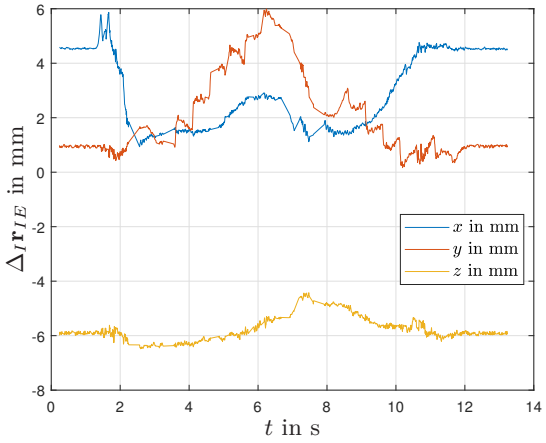


Abbildung 6.46: Experiment: Positionsfehler der Steuerung, für den Vergleich mit den Regelungskonzepten

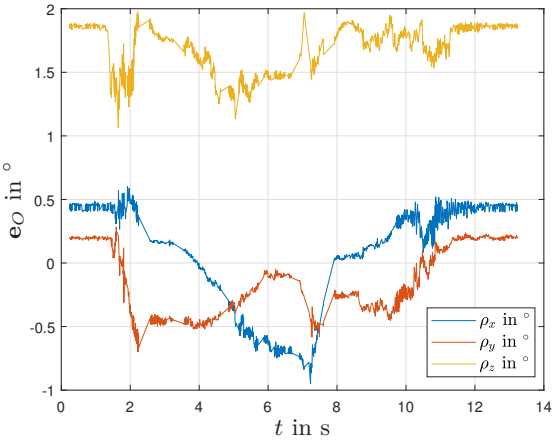


Abbildung 6.47: Experiment: Orientierungsfehler der Steuerung, für den Vergleich mit den Regelungskonzepten

Wie bereits bei der Kompensation der geometrischen Fehler zeigt sich auch bei der kamerabasierten Regelung, dass während des Abfahrens der Bahn signifikante Positions- und Orientierungsfehler auftreten. Dies ist hauptsächlich auf die hohe Totzeit in der Kommunikation über die EGM-Schnittstelle zurückzuführen, aber auch auf Fehler bei der Detektion des Endeffektors. Da außerdem das Regelverhalten der EGM-Steuerung als auch die dynamischen Parameter des Roboters nicht genau bekannt sind, kann auch keine Vorsteuerung implementiert werden, um ein besseres dynamisches Verhalten zu erreichen. Bei den getesteten Regelungskonzepten liefert das Regelungskonzept 1 die schlechteste Performance in Bezug auf den Positionsfehler, mit einem maximalen Fehler, der sogar über dem der reinen Steuerung ohne Regelung liegt.

Regelungskonzept 2 erzielt die beste Performance in Bezug auf den Positionsfehler, weist jedoch die schlechteste Leistung bei der Orientierungskontrolle auf, während hier Regelungskonzept 1 am besten abschneidet.

Das Regelungskonzept 3 scheint einen guten Kompromiss zwischen beiden Fehlerarten darzustellen. Dennoch bleibt die Gesamtperformance, mit einem maximalen Positionsfehler von 3.37 mm und einem maximalen Orientierungsfehler von 1.06°, aufgrund der genannten Probleme deutlich hinter den angestrebten Zielen und der Leistung der geometrischen Kalibrierung zurück.

Hinsichtlich der statischen Genauigkeit schneidet die kamerabasierte Regelung jedoch wesentlich besser ab als die geometrische Kalibrierung. Hier zeigt das Regelungskonzept 1 die beste Performance mit einem Positionsfehler von 0.09 mm und einem Orientierungsfehler von 0.08°, gefolgt von Regelungskonzept 3. Das gesetzte Ziel konnte hier somit erreicht werden.

Wie bereits in Kapitel 5 festgestellt, gelingt es hingegen Regelungskonzept 2 nicht, die Orientierung effektiv auszuregeln. Dies liegt an den in der Orientierungsdarstellung getroffenen Vereinfachungen, die eine genaue Regelung erschweren. In Tabelle 6.8 sind die gemessenen Fehler der verschiedenen Regelungskonzepte zum Vergleich angeführt.

Tabelle 6.8: Experiment: Vergleich des Fehlerverhaltens der kamerabasierten Regelungen

		Δx in mm	Δy in mm	Δy in mm	ρ_x in °	ρ_y in °	ρ_z in °
maximaler absoluter Fehler	Steuerung	5.87	6.00	6.48	0.95	0.70	1.98
	Regelungs-konzept 1	6.42	6.15	0.54	0.40	0.31	2.77
	Regelungs-konzept 2	2.82	1.62	0.94	2.45	2.55	3.72
	Regelungs-konzept 3	3.37	2.83	1.48	0.68	0.65	1.06
statische Genauigkeit	Steuerung	4.56	1.02	5.98	0.40	0.18	1.89
	Regelungs-konzept 1	0.07	0.09	0.07	0.03	0.02	0.08
	Regelungs-konzept 2	0.09	0.08	0.11	1.16	1.95	0.32
	Regelungs-konzept 3	0.12	0.20	0.11	0.06	0.02	0.08

Beim Vergleich des realen Systems mit der Simulation zeigt sich, ähnlich wie bei der geometrischen Kalibrierung, eine höhere Genauigkeit im realen System. Dies deutet darauf hin, dass die Echtzeitfähigkeit der Simulation unter dem hohen Rechenaufwand leidet, wodurch das Simulationsergebnis verfälscht wird. Außerdem basieren die Fehler, die in der Simulation ausgeregelt werden, auf vereinfachten Annahmen, die nicht mit den im realen System auftretenden Abweichungen übereinstimmen.

Kapitel 7

Arbeitsraumüberwachung

Ein wesentlicher Vorteil bei der Verwendung von Kamerasystemen zur Erfassung der Pose des Endeffektors liegt in der vielseitigen Verwendbarkeit der Bilddaten. Diese Daten können in einem separaten, langsameren Zyklus zusätzlich für andere Aufgaben genutzt werden. Beispielsweise können die Kamerabilder zur Personenerkennung eingesetzt werden, wenn eine Person innerhalb des Arbeitsbereichs des Roboters detektiert wird, kann automatisch ein Not-Stopp ausgelöst oder die maximal zulässige Geschwindigkeit des Roboters reduziert werden, um die Sicherheit zu gewährleisten.

Darüber hinaus ermöglichen die Bilddaten die Erkennung von Veränderungen in der Umgebung oder am Roboter selbst. Sollte sich die Struktur des Roboters verändern, etwa durch ein gelöstes Sensorkabel, das Abfallen einer Verkleidung, eine Beschädigung oder Verformung des Werkzeugs am Endeffektor, kann dies frühzeitig festgestellt werden. In solchen Fällen kann das System eine Warnung oder einen Wartungshinweis an den Bediener der Anlage senden. Diese Benachrichtigung kann durch ein Bild ergänzt werden, das die genaue Position und Art der Anomalie zeigt, was eine schnelle und gezielte Reaktion ermöglicht.

Um die Möglichkeiten dieser Technologie zu demonstrieren, wurde in dieser Arbeit eine Personenerkennung implementiert. Hierfür wird ein vortrainiertes tiny-YOLO (You Only Look Once) Version 3 Netzwerk verwendet, das von Matlab bereitgestellt und auf dem COCO-Datensatz trainiert wurde [19]. Ein YOLO-Netzwerk ist eine spezielle Form von einem Convolutional Neural Network (CNN), die es ermöglicht Objekte in Bildern in Echtzeit zu erkennen und zu klassifizieren. Die Funktionsweise des YOLO-Netzwerks lässt sich nach [44] und [12] folgendermaßen zusammenfassen.

Es teilt das Bild in ein Gitter auf und bewertet jede Zelle, um festzustellen, ob ein Objekt vorhanden ist. Dabei kombiniert es sowohl die Erkennung als auch die Lokalisierung in einem einzigen Netzwerk, was zu einer extrem schnellen und genauen Objekterkennung führt.

Es besteht dabei aus folgenden Komponenten:

- **2-D Convolution**:
  Der Convolutional Layer ist verantwortlich für die Extraktion relevanter Merk-

male aus dem Bild. Durch die mathematische Operation der Faltung werden lokale Merkmale wie Kanten, Ecken oder Texturen erkannt. Dazu werden Filter (Kernels) einer bestimmten Größe (z.B. 3×3) verwendet, die über das Bild "gleiten". Die Einträge im Kernel entsprechen den Weights die sich durch Training ergeben. Der Stride gibt dabei an, wie viele Positionen der Filter nach jedem Schritt weiterbewegt wird. Ein Stride von 1 bedeutet beispielsweise, dass der Filter um eine Position horizontal und eine Position vertikal verschoben wird.

- **2-D Max Pooling**:
  Hierbei handelt es sich um einen Filter, ähnlich dem 2-D Convolution-Layer, mit dem Ziel der Minimierung der Dimensionen für bessere Rechenleistung sowie Extraktion der wichtigsten Merkmale. Beim Max Pooling wird innerhalb eines definierten Fensters (z.B. 2×2) der maximale Wert ausgewählt und als repräsentativer Wert für dieses Fenster übernommen. Der Stride gibt auch hier an, um wie viele Positionen das Fenster verschoben wird.

- **Batch Normalization**:
  Diese Technik wird angewendet, um den Lernprozess zu stabilisieren und zu beschleunigen. Batch Normalization normalisiert die Ausgaben jedes Layers, wodurch Schwankungen in den Eingabewerten ausgeglichen werden. Dies führt zu einem effizienteren Training, indem es die Notwendigkeit einer sorgfältigen Initialisierung der Gewichte reduziert und die Abhängigkeit von der Lernrate verringert.

- **Leak ReLU**:
  Hierbei handelt es sich um die Activation Function des Netzwerks. Diese hat die Aufgabe, die Ausgabe eines Neurons zu transformieren, um Nichtlinearität in das ansonsten lineare Netzwerk einzuführen. Dadurch kann das Netzwerk komplexe nichtlineare Muster in den Daten erfassen.

- **Upsample und Depth Concatenation**:
  Die Depth Concatenation ermöglicht es verschiedene Schichten eines Netzwerks zusammenzufügen, um relevante semantischen Merkmale erkennen zu können. Hierfür ist es notwendig, dass die ersten beiden Dimensionen der Schichten übereinstimmen (z.B. für dieses Netzwerk 26×26). Um dies zu ermöglichen, kann über ein Upsampling die Dimension erhöht werden.

Das verwendete tiny-YOLO-Netzwerk besteht aus 8.8 Millionen trainierbaren Parametern (Learnables) und umfasst 44 Schichten (Layers). Als Eingang wir ein RGB - Bild mit 416×416 Pixel verwendet. Da die Kamera eine höhere Auflösung hat, wird das Bild zuerst in einem Downsamplig-Schritt auf diese Größe angepasst. Als Output liefert das Neuronale Netz zwei Feature Tensoren, welche jeweils die Lokalisation der Objekte im Bild (bounding boxes), die Klassifizierung der erkannten Objekte (classes) und den Grad der Übereinstimmung (score) enthalten. Der erste Tensor nutzt eine geringere Auflösung (13×13) und erkennt große Objekte gut, der andere hingegen eine höhere Auflösung (26×26) für die Erkennung kleiner Objekte im Bild. Die Struktur

ist in Abbildung 7.1 dargestellt, mit der detaillierten Beschreibung der einzelnen Schichten in Tabelle A.9.

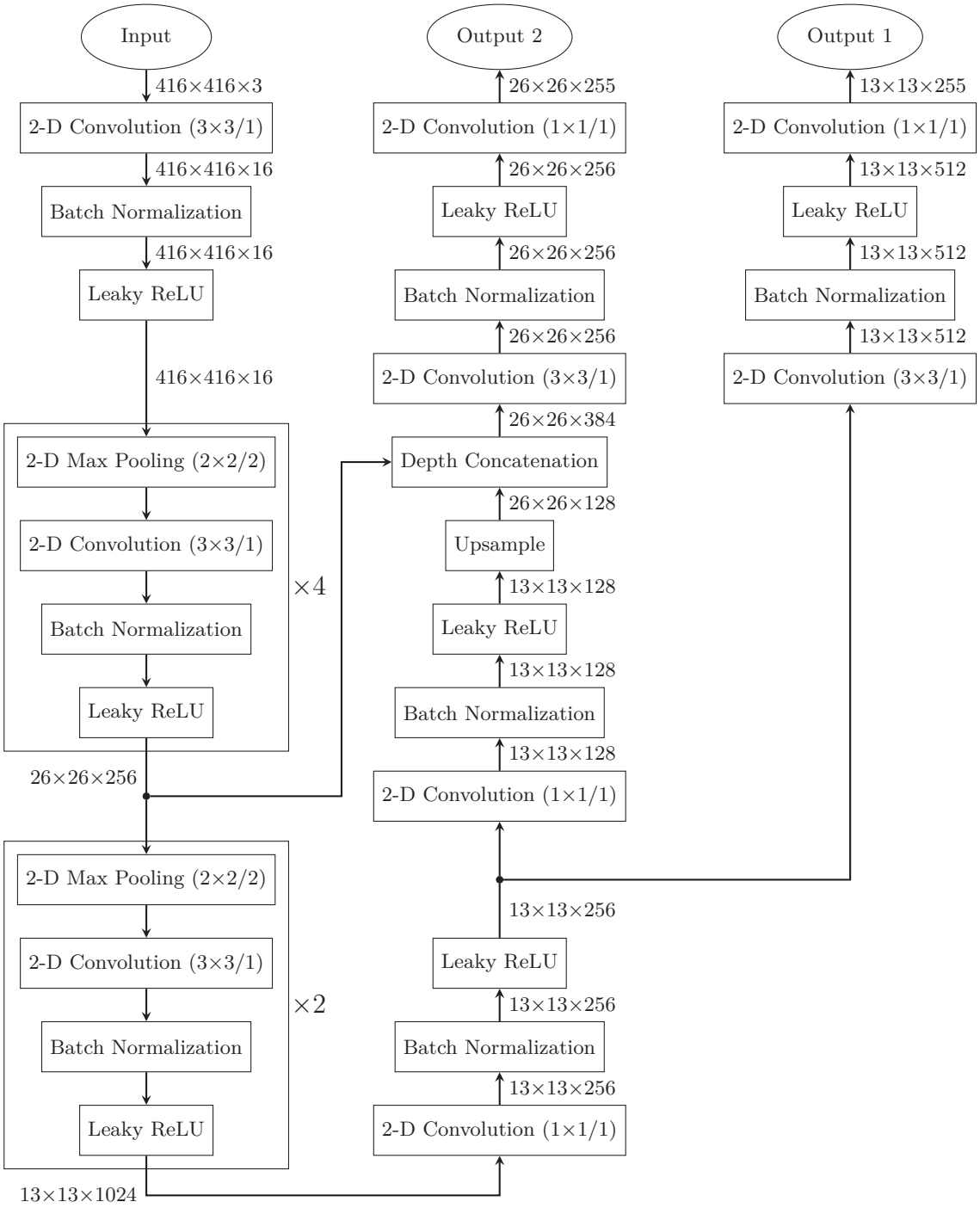


Abbildung 7.1: Ablauf tiny YOLO Netzwerk

Das beschriebene Netzwerk wird anschließend in der Simulink Realtime Toolbox betrieben. Hierfür wird der Simulinkblock Object-Detector verwendet. Ein weiterer Block filtert die Output-Tensoren nach der Klasse Person, überprüft ob der Score über einen gewissen Grenzwert liegt und vergleicht ob das Objekt sich im Arbeits-

raum befindet. Gewählt wurde ein Threshold von 0.5, wobei dies auf einer Annahme beruht. Eine optimale Wahl muss erst noch ermittelt werden, prinzipiell besteht hier ein Trade-Off zwischen Falschdetektion einer Person und nicht Identifikation einer Person. Falls alle Abfragen positiv sind liefert diese Funktion einen positiven Output, ansonsten einen negativen. Der positive Output kann dann anschließend verwendet werden, um geeignete Maßnahmen zu triggern. In dem Testaufbau dient er jedoch als Endergebnis. Als Eingang dient die Webcam Flex 1080P HD No. 133-97 der Firma Sandberg, das Kamerasystem des Motion-Capturing-Systems Qualisys konnte hier aufgrund von Kompatibilitätsproblemen leider nicht verwendet werden. Zur Überprüfung der Machbarkeit ist das verwendete Setup jedoch ausreichend. Eine mögliche Erweiterung der Versuche ist in Kapitel 8 ausgeführt. Bei den Versuchen konnten die Person im Arbeitsraum gut identifiziert werden, jedoch trat das Problem auf das manchmal 2 überlappende Bounding-Boxes über die selbe Person gezeichnet wurden, wie in Abbildung 7.2 erkennbar. Grundsätzlich lässt sich sagen, dass dieser zusätzliche Anwendungsfall für die Kameras relativ leicht implementiert ist und eine gute Möglichkeit bietet die Sicherheit einer Produktionsanlage zu verbessern. In dieser Ausführung können jedoch die bisherigen Sicherheitsmechanismen noch nicht ersetzt werden, sondern sie dient lediglich als Add-On.

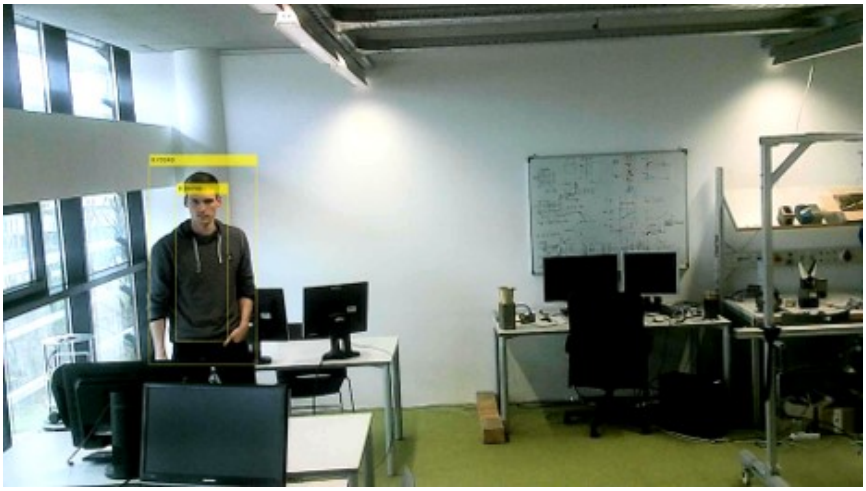


Abbildung 7.2: Personenerkennung im Arbeitsraum

Kapitel 8

Zusammenfassung und Ausblick

In dieser Arbeit wurden zwei Methoden zur visuellen Steigerung der Genauigkeit eines Industrieroboters untersucht: die geometrische Kalibrierung und die kamerabasierte Regelung. Beide Ansätze wurden zunächst in einer Simulationsumgebung getestet und anschließend auf das reale System übertragen. Die Ergebnisse zeigen, dass die Simulationen zwar zur Verifikation der Funktionsweise der Algorithmen geeignet sind, die exakte Vorhersage des realen Verhaltens jedoch eingeschränkt bleibt. Dies ist auf verschiedene Faktoren zurückzuführen:

- **Leistungsgrenzen:** Die Simulation stieß aufgrund des hohen Rechenaufwands an die Leistungsgrenzen des verwendeten PCs, was zu einer eingeschränkten Echtzeitfähigkeit führte. Im Vergleich dazu wiesen die realen Tests eine bessere Echtzeitperformance auf, da weniger Berechnungen erforderlich waren.
- **EGM-Kommunikation:** Die EGM-Schnittstelle verursachte Kommunikationsverzögerungen von etwa 140 ms. Diese Totzeiten könnten sowohl auf die nicht vollständig ausgereiften Schnittstellen als auch auf die Beschränkungen der ABB-Technologie zurückzuführen sein.
- **Annahme der Fehlerparameter:** Die in der Simulation verwendeten Fehlerparameter und die angenommene Genauigkeit der Messwerte basierten auf idealisierten Annahmen. Dies erklärt die großen Abweichungen zwischen Simulation und realen Versuchen, da die tatsächlichen Fehlerquellen in der Simulation nicht vollständig abgebildet wurden.

Bei den Tests am realen System konnte eine statische Genauigkeit von 0.69 mm bei der geometrischen Kalibrierung und unter 0.1 mm bei der kamerabasierten Regelung erreicht werden. Die dynamische Genauigkeit konnte durch die vorgestellten Methoden ebenfalls verbessert werden, das Ziel einer Genauigkeit im Bereich von Zehntelmillimetern wurde jedoch aufgrund einiger Einschränkungen nicht erreicht:

- **Probleme bei der Orientierungsdetektion:** In bestimmten Konfigurationen konnte die Orientierung des Endeffektors nicht zuverlässig bestimmt werden. Eine Verbesserung könnte durch das Finetuning der Detektionsparameter des Motion-

Capturing-Systems oder durch den Einsatz eines anderen Systems mit höherer Genauigkeit erzielt werden. Außerdem war die Positionierung der Kameras nicht optimal und hier besteht ebenfalls Verbesserungspotential.

- **Begrenzte Genauigkeit der Positionsmessung:** Das verwendete Qualisys-System war für andere Anwendungen ausgelegt und bot nicht die erforderliche Präzision. Die Kameraauflösung war im Verhältnis zum Abstand zum Roboter zu gering, was durch hochauflösendere Kameras verbessert werden könnte.
- **EGM-Kommunikation:** Auch im realen System hatten die Kommunikationsverzögerungen über die EGM-Schnittstelle einen signifikanten Einfluss auf die Genauigkeit.
- **Unkenntnis der genauen Regelstruktur:** Die fehlende Kenntnis über die interne Regelstruktur des Roboters machte es schwierig, die kamerabasierte Regelung zu optimieren.
- **Identifikation der Dynamik:** Durch Identifikation der Dynamik des Roboters kann durch Vorsteuerung, oder Modifizierung der Regelkonzepte ein besseres dynamisches Verhalten erzielt werden.
- **Optimierung der Regelparameter:** Die gewählten Regelparameter lieferten zwar gute Ergebnisse, könnten aber durch weitergehendes Finetuning noch verbessert werden.

Die Analyse der verschiedenen Ansätze ergab, dass sich das Regelkonzept 1 am besten zur Verbesserung der statischen Genauigkeit eignete, jedoch weniger robust als die geometrische Kalibrierung war. Bei der dynamischen Genauigkeit zeigte die geometrische Kalibrierung die besten Ergebnisse, da sie direkt die Steuerung des Roboters optimierte und dadurch weniger durch die EGM-Totzeiten beeinflusst wurde. Wenn die Kommunikationsverzögerungen beseitigt werden könnten, würde voraussichtlich das Regelkonzept 3 am besten abschneiden.

Die geometrische Kalibrierung ohne Verwendung der Orientierungsdaten sowie das Regelkonzept 2 eignen sich nur, wenn primär die Position des Endeffektors wichtig ist und die Orientierung eine untergeordnete Rolle spielt

Trotz der bestehenden Einschränkungen konnte eine wesentliche Verbesserung der Genauigkeit durch geometrische Kalibrierung und kamerabasierte Regelung erreicht werden. Mit weitergehenden Maßnahmen, die in Abschnitt 8.1 beschrieben sind, erscheint die Erreichung einer Genauigkeit im Bereich von Zehntelmillimetern durchaus realistisch. Es wurde jedoch nur eine spezifische Testbahn untersucht (siehe Abschnitt 5.1), weshalb eine Erweiterung der Tests notwendig ist (siehe Abschnitt 8.3), um ein umfassenderes Bild des Systemverhaltens zu erhalten.

An dieser Stelle wird außerdem noch einmal hervorgehoben das die kamerabasierte Kalibrierung und Regelung gegenüber anderen Messmethoden den Vorteil bietet, dass visuelle Daten eine Vielzahl an Informationen enthalten, die für verschiedene industrielle Anwendungen genutzt werden können. Im Kapitel 7 wurde gezeigt, dass

selbst einfache Methoden zur Auswertung dieser Daten bereits vielversprechend sind, jedoch für den industriellen Einsatz noch weiter verbessert werden müssen. Weitere Details dazu werden in Abschnitt 8.2 erläutert.

8.1 Geplante Verbesserungen

Zur Verbesserung der Endeffektor Detektion kann auf ein eigens für diese Anwendung zusammengestelltes Kamerasetup übergegangen werden. Dies hat den Vorteil, dass mehr Flexibilität bei der Auswahl der Komponenten besteht, wodurch zum einen die Genauigkeit der Bilder gesteigert werden kann zum anderen eine gezieltere Optimierung der Algorithmen zur Objekterkennung ermöglicht wird. Zudem wird ein direkter Zugriff auf die Roh-Bilddaten ermöglicht, was die Implementierung zusätzlicher Funktionen wie der in Kapitel 7 beschriebenen Arbeitsraumüberwachung erleichtert.

Ein Nachteil bei der Übertragung ganzer hochauflösender Bilder in Echtzeit ist jedoch die enorm hohe Datenrate, die erforderlich wäre. Dies resultiert aus der Kombination von hoher Auflösung und schneller Bildwiederholrate. Um dieses Problem zu bewältigen, kann die Sprinter-Glasfasertechnologie eingesetzt werden, die eine Echtzeit-Datenübertragung mit einer Bandbreite von bis zu 10 Gbit/s ermöglicht.

Eine zusätzliche Möglichkeit zur Handhabung der Datenrate besteht darin, nur die relevanten Bildausschnitte in hoher Bildrate zu übertragen. Hierzu wird im Vorfeld berechnet, in welchem Bereich des Kamerabildes sich der Endeffektor voraussichtlich befinden wird, basierend auf dem Modell (2.9). Anschließend wird lediglich dieser Bildbereich mit einem definierten Toleranzrahmen in hoher Auflösung übertragen. Das restliche Bild, welches zur Arbeitsraumüberwachung dient, kann mit einer niedrigeren Bildrate und Auflösung gesendet werden. Auf diese Weise lässt sich die benötigte Datenmenge erheblich reduzieren, ohne dass wichtige Informationen verloren gehen.

Der Ablauf dieses optimierten Erfassungs- und Übertragungsalgorithmus ist in Abbildung 8.1 schematisch dargestellt.

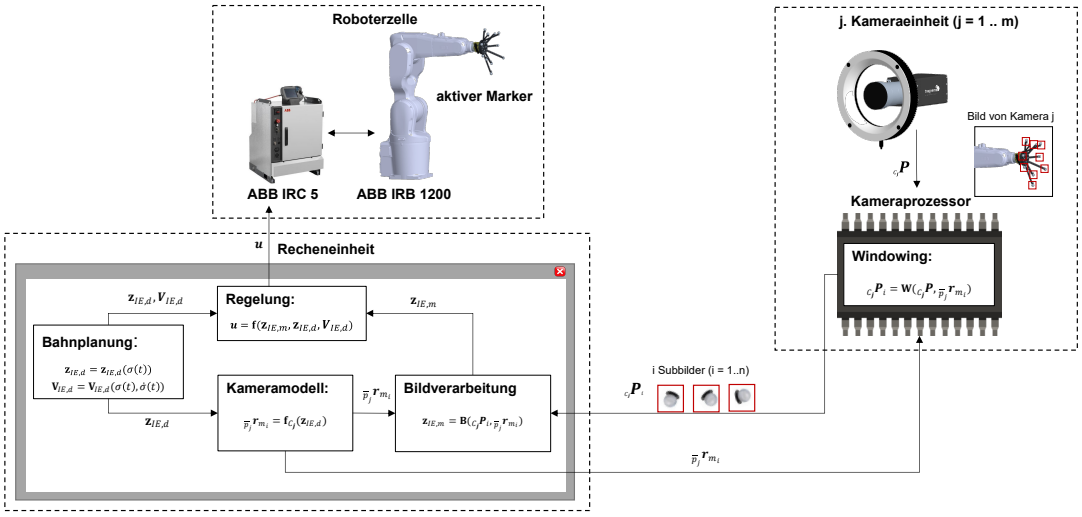


Abbildung 8.1: Ablauf des optimierten Erfassungsalgorithmus

Zum Testen dieser Verbesserungen sind weitere Versuche mit dem in Abbildung 8.2 dargestellten Aufbau geplant. Dabei wird auf die Highspeed-Kamera HB-25000-G-N von Emergent Vision übergegangen. Diese Kamera bietet die Möglichkeit, Auflösung und Framerate flexibel zu variieren, wodurch verschiedene Testfälle entstehen. So kann bestimmt werden, welche Rahmenbedingungen für eine präzise Detektion des Endeffektors erforderlich sind. Im Detail besteht der verbesserte Testaufbau aus folgenden Komponenten:

- **Kamerasysteme**: Vier Highspeed-Kameras des Typs HB-25000-G-N streamen die Bilddaten. Jede Kamera ist mit Infrarot-Beleuchtung, IR-Filtern und passenden Objektiven ausgestattet, um eine präzisere Erfassung des Endeffektor zu gewährleisten.
- **Datenübertragung**: Die Kameradaten werden über einen Glasfaserlink mit einer Gesamt-Datenrate von 10 Gbit/s an ein zyklisches Arrayed Waveguide Grating (cAWG) übertragen. Dieses cAWG sorgt dafür, dass die Daten an unterschiedliche Recheneinheiten weitergeleitet werden, indem es die Datenströme nach ihrer Wellenlänge sortiert. Auf diese Weise wird die parallele Datenverarbeitung simuliert.
- **Rechenzentrum und Steuerungseinheiten**: Die Daten werden in einem externen Industrie-Server verarbeitet, der speziell für die Anforderungen der Echtzeitsteuerung und Bildverarbeitung konfiguriert ist. Hierbei ist es wichtig, die in Abschnitt 6.3 beschriebene Schnittstelle zu verbessern, um die Totzeit unter 10 ms zu senken.
- **Roboter und Steuerung**: Der in diesem Testaufbau verwendete Roboter ist der ABB IRB 1200, gesteuert durch den ABB IRC5 Controller, wie bereits im vorherigen Setup. Der Roboter und die Steuerungseinheit sind auf einer beweglichen Basis montiert, um maximale Flexibilität bei der Positionierung des Aufbaus zu gewährleisten.

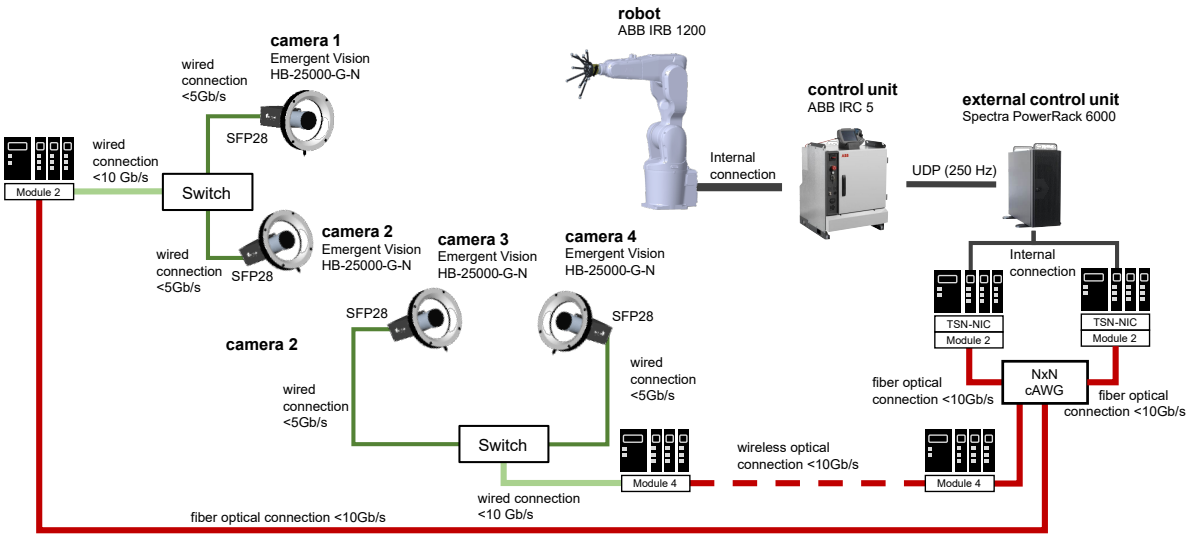


Abbildung 8.2: Erweiterter Versuchsaufbau

Dieser Testaufbau erlaubt eine genaue Analyse der Anforderungen an die Erkennungsgenauigkeit des Endeffektors unter verschiedenen Kamerakonfigurationen. Zudem wird untersucht, wie die Datenverarbeitung von einem dezentralen Anlagenbereich in einen zentralen Serverraum verlagert werden kann. Dies ermöglicht die Ausführung rechenintensiver Algorithmen an einem zentralen Ort und gewährleistet eine effiziente Ressourcennutzung sowie eine hohe Flexibilität im System.

8.2 Arbeitsraumüberwachung

Der neue Versuchsaufbau ermöglicht es, die Personenerkennung parallel zur Erfassung des Endeffektors in Echtzeit zu erproben. Darüber hinaus kann die in Kapitel 7 vorgestellte Idee zur automatisierten Erkennung von Schäden an der Anlage durch ein weiteres Convolutional Neural Network umgesetzt werden. Dieses CNN soll kontinuierlich den Zustand der Anlage überwachen und Anomalien wie gelöste Kabel, gebrochene Komponenten oder defekte Werkzeuge frühzeitig erkennen.

8.3 Erweiterung der Tests

Für zukünftige Tests und Optimierungen ist es vorgesehen, zusätzliche Testbahnen wie Kreis- und Achterbahnen zu implementieren, um so die Leistungsfähigkeit unter realistischen Bedingungen zu evaluieren. Ein weiterer Fokus liegt im Vergleich der Kameramessung mit etablierten Messmethoden, wie dem Lasertracker. Ebenso sollen gezielte Versuche den Einfluss von Kameraauflösung und Bildrate auf die Genauigkeit der Endeffektorerfassung untersuchen, um zusammen mit der Weiterentwicklung des Detektionskörpers die Basis für eine Empfehlungen der optimalen Wahl der Komponenten und Einstellungen abzuleiten. Die Weiterentwicklung des Detektionskörpers wird eine Optimierung der Markerpositionen, sowie den Übergang von passiven auf aktive Marker und eine Integration der Lasertracker Markierungen beinhalten.

Anhang A

Ergänzungen zu den verwendeten Programmen

A.1 Konfiguration der EGM –Schnittstelle

Zum Aktivieren und Konfigurieren von EGM muss zunächst die Robotersteuerung über die Service-Schnittstelle mit einem PC verbunden, auf dem ABB RobotStudio (RS) installiert ist. In RS im *Datei*-Tab unter dem Menüpunkt *Online* lässt sich eine Verbindung mit der Steuerung herstellen.

Im *Steuerungs*-Tab von RS kann anschließend die Robotersteuerung per Rechtsklick ausgewählt werden, woraufhin sich eine Auswahlliste öffnet (siehe Abbildung A.1). Wählt man dort *Optionen ändern*, öffnet sich ein neues Fenster. In diesem Fenster kann unter dem Abschnitt *Engineering Tools* die Option *Externally Guided Motion (EGM)* aktiviert werden (Abbildung A.2).

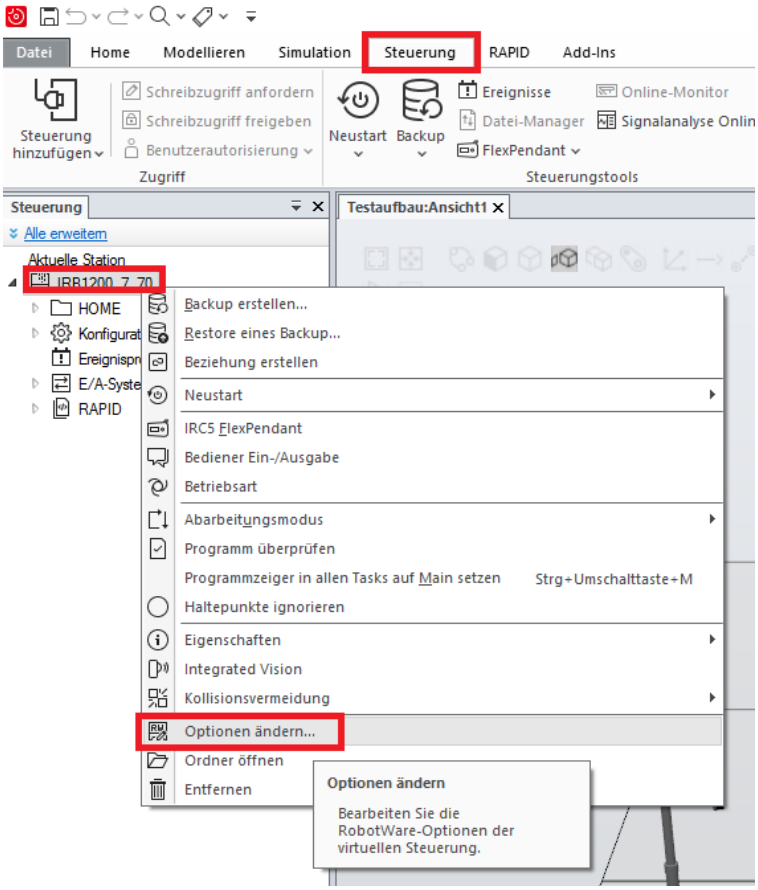


Abbildung A.1: Erster Schritt zur Aktivierung von EGM

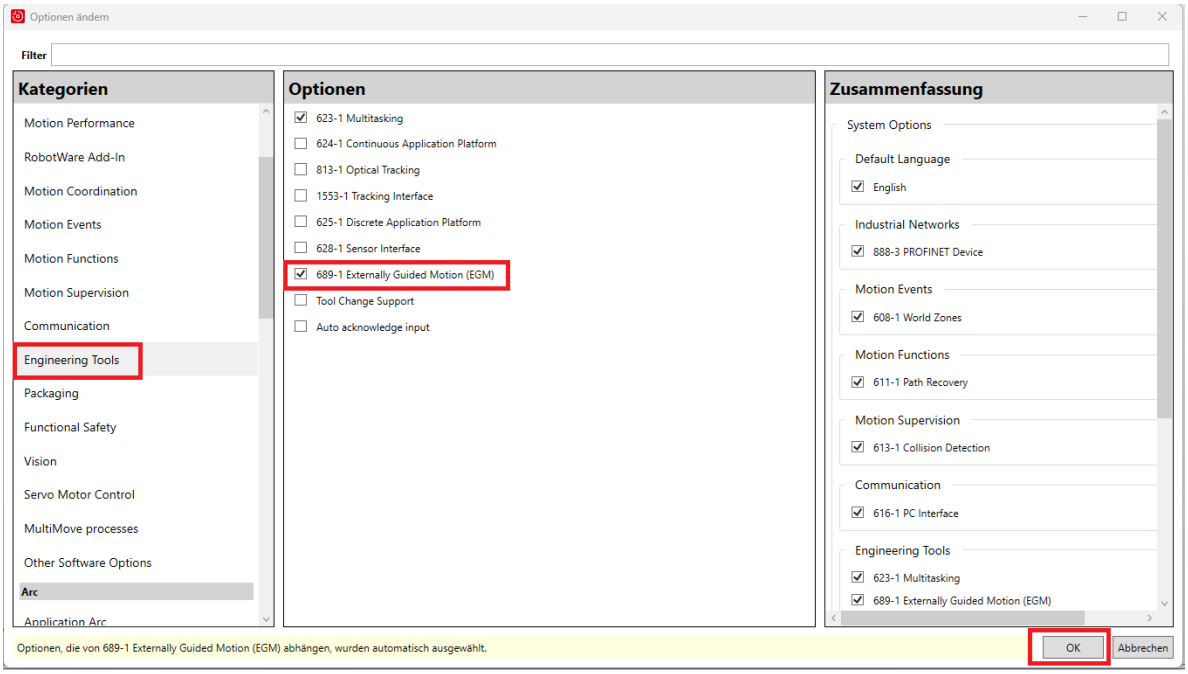


Abbildung A.2: Zweiter Schritt zur Aktivierung von EGM

Zusätzlich sind die folgenden Konfigurationen vorzunehmen:

1. **UDP-Kommunikation konfigurieren**: Unter *Konfiguration/Communication* muss im Abschnitt *Transmission Protocol* ein neues UDP-Protokoll angelegt werden, in dem die entsprechende IP-Adresse und der gewünschte Port definiert werden. Die genaue Belegung der IP und Port-Konfiguration ist in Anhang B.4 zusammengefasst.

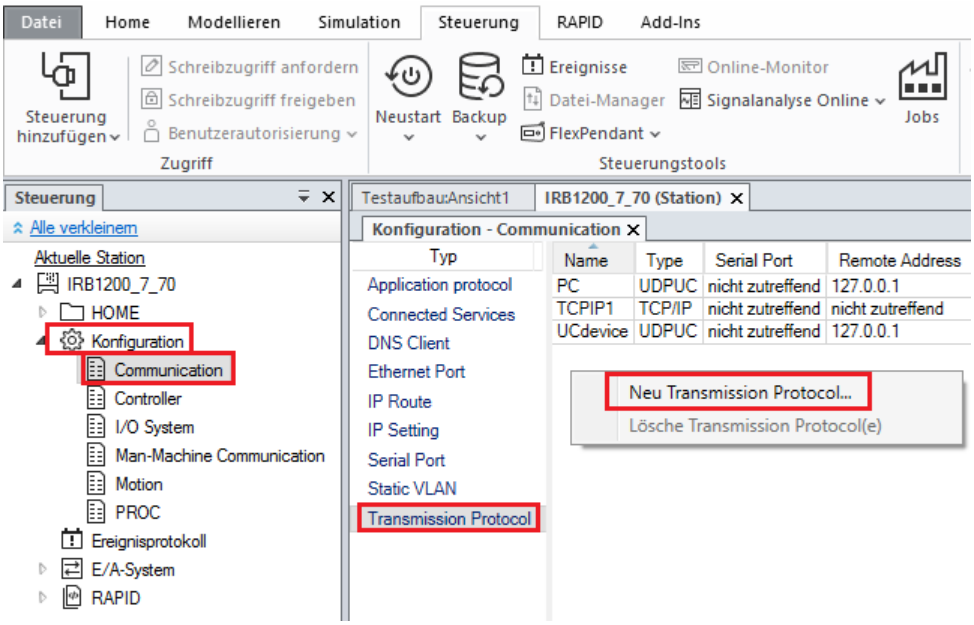


Abbildung A.3: UDP-Kommunikation konfigurieren

2. **EGM-Steuerung konfigurieren**: Unter *Konfiguration/Motion* muss im Abschnitt *External Motion Interface Data* eine neue Konfiguration angelegt, bei der Parameter wie Filterlevel, Proportional Gain, LP-Bandbreite und Ramp Time eingestellt werden. Eine vollständige Liste der verwendeten Konfigurationsparameter ist in Tabelle A.1 angeführt.

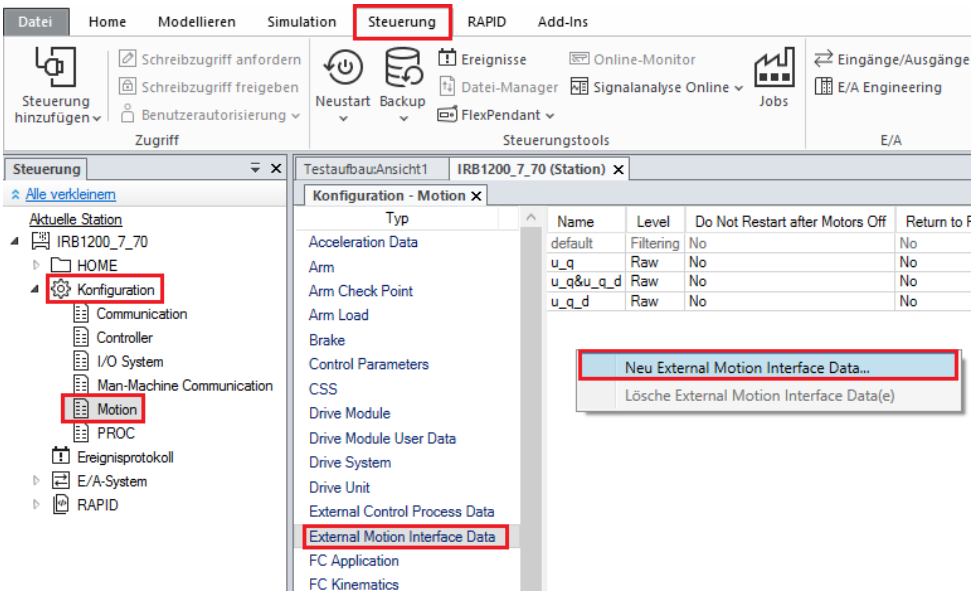


Abbildung A.4: EGM - Einstellungen konfigurieren

Tabelle A.1: Einstellungen von External Motion Interface Data

Name	Level	Default Ramp Time	Default Proportional Gain	Default Low Pass Filter Bandwidth
u_q	Raw	0.005	20	100
u_q&u_q_d	Raw	0.005	20	100
u_q_d	Raw	0.005	0	100

A.2 EGM Implementierung auf der Steuerungsseite

Die Implementierung von EGM auf der Robotersteuerung folgt einem klar definierten Ablauf: Zuerst wird EGM durch die Bewegungssteuerung aufgerufen. Im nächsten Schritt liest EGM die Feedbackdaten, wie beispielsweise die aktuelle Position, aus der Bewegungssteuerung aus. Diese Daten werden anschließend über den Position Stream an das externe Gerät gesendet. Im weiteren Verlauf überprüft EGM die UDP-Warteschlange, um neue Meldungen vom externen Gerät zu empfangen. Diese neuen Soll-Positionen werden durch die Position Guidance übermittelt. Wenn eine neue Meldung vorliegt, liest EGM die entsprechenden Daten aus und schreibt die neuen Positionsinformationen in die Bewegungssteuerung. Falls jedoch keine neuen Positionsdaten verfügbar sind, setzt die Steuerung die Bewegung mit den zuletzt empfangenen Soll-Positionen fort. Für diesen Ablauf stellt ABB verschiedene Rapid-Funktionen zur Verfügung. Mit diesen wurde das Modul *EGM_Communication* verfasst in dem die für diese Arbeit notwendigen Anwendungen implemetiert wurden. Das Modul enthält hierfür die Prozeduren *EGM_POSITION_STREAM* und *EGM_MOVEMENT*, als auch Felder für verschiedene Einstellungen, dargestellt in Tabelle A.2.

Die Prozedur *EGM_MOVEMENT* mit dem Eingabeparameter *mode* dient zur Anwendung von EGM Position Guidance zusammen mit EGM Position Stream, wobei

[1]Das UDP-Gerät muss nach Anhang A.1 eingerichtet sein.

Tabelle A.2: Beschreibung der Felder des Moduls *EGM_Communication*

Name	Beschreibung
UCDevice	Name des UDP-Geräts[1]
egm_minmax_joint	Konvergenzbedingung bei der Gelenkbewegung
egm_minmax_lin1	Konvergenzbedingung bei der Position
egm_minmax_rot1	Konvergenzbedingung bei der Orientierung

der ganzzahlige Eingabeparameter im Bereich von 0 bis 5 definiert, welche Daten übertragen und empfangen werden sollen. Die Auswahlmöglichkeit ist in Tabelle A.3 dargestellt.

Für die Nutzung von lediglich EGM Position Stream dient die Prozedur *EGM_POSITION_STREAM* mit dem Eingabeparameter *stream_jointdata*. Der boolsche Eingabeparameter legt fest, ob Gelenkwinkel oder Posen gestreamt werden sollen. Dieser Befehl dient zur Initialisierung und muss nur einmal vor Verwendung aufgerufen werden.

Tabelle A.3: Beschreibung des Eingabeparameters *mode*

Wert	Beschreibung
0	Übertragung der Gelenkwinkel
1	Übertragung der Gelenkwinkel und Gelenkgeschwindigkeit
2	Übertragung der Gelenkgeschwindigkeit
3	Übertragung der Pose
4	Übertragung von Pose und Twist
5	Übertragung des Twists

Der Ablauf der Testsignal-Übertragung sieht ähnlich aus. Die Testsignale werden zyklisch ausgelesen, in einem Datenpaket zusammengefasst und über das PC-Interface via UDP an das externe Gerät gesendet. Wie bei der EGM-Kommunikation werden die ausgelesenen Daten mit einem Zeitstempel versehen, um eine exakte zeitliche Zuordnung zu gewährleisten. Für die Testsignal-Übertragung wurde das Modul *TEST_SIG_Communication* entwickelt, welches in einem eigenen Task laufen muss. Gestartet wird das Streaming indem die globale Variable *start_streaming_tsig* gesetzt wird. Zusätzlich müssen durch die globalen Variablen *remote_address_tsig* und *remote_port_tsig* vorher die IP-Adresse und der Port des UDP-Geräts definiert werden.

Tabelle A.4: Variablen zur Steuerung der Testsignal Übertragung

Name	Beschreibung	Datentyp
start_streaming_tsig	Aktivieren/ Deaktivieren des Testsignal-Streamings	bool[2]
remote_address_tsig	IP-Adresse des UDP-Geräts	string
remote_port_tsig	Port des UDP-Geräts	num

[2] true ... Aktivieren, false ... Deaktivieren.

A.3 EGM Implementierung auf der PC-Seite

Auf der PC-Seite beginnt der Prozess mit der Initialisierung der Kommunikation, einschließlich der Festlegung von IP-Adressen, dem Öffnen von Ports, dem Setzen von Timeouts sowie Definition der Modi. In der anschließenden zyklischen Kommunikation werden neue Positionen, abhängig von der gewählten Konfiguration, an den Roboter gesendet, wobei der genaue Sendezeitpunkt für die spätere Analyse gespeichert wird. Anschließend werden die EGM-Daten und/ oder die Testsignale ausgelesen. Sollte innerhalb eines festgelegten Timeouts keine Datenübertragung stattfinden, wird der Schritt übersprungen und ein Fehlerzähler erhöht. Anschließend werden die empfangenen Positionsdaten und Geschwindigkeitsdaten an die Ausgangsports weitergeleitet, und der aktuelle Status des Roboters wird verarbeitet und notfalls werden Maßnahmen ergriffen.

A.3.1 C++ Projekt

Das C++ Projekt liefert eine Klasse *Robot*, angelehnt an die Arbeit [42], in welcher die UDP-Übertragung auf PC-Seite umgesetzt ist. In dieser Klasse sind zwei Modus-Typen definiert: *MODE* und *UDP_MODE*. Diese Modi legen fest, welche Daten zwischen Roboter und PC über UDP übertragen und empfangen werden und nach welcher Regelung das Buffering der empfangenen Daten stattfindet. Sie sind in Tabelle A.6 und Tabelle A.5 beschrieben. Eine detaillierte Beschreibung der öffentlichen Felder liefert Tabelle A.7. Im Folgenden werden die Funktionen von *Robot* genauer beschrieben:

- *setIPAddress*: Diese Funktion definiert die IP-Adresse der Robotersteuerung, mit der über UDP kommuniziert wird. Die gewünschte IP-Adresse wird als *String* (z.B. "127.0.0.0") übergeben. Wenn die IP-Adresse ungültig ist, erfolgt die Kommunikation mit allen Geräten auf dem ausgewählten Port.
- *setPort*: Über diese Funktion wird der Kommunikationsport festgelegt. Es können entweder der EGM-Kommunikationsport und der Testsignal-Kommunikationsport als *int* übergeben werden oder nur ein Port, wobei ein Flag angibt, ob es sich um den EGM- oder den Testsignal-Port handelt.
- *setTimeOut*: Diese Funktion setzt die Timeout-Werte für die UDP-Kommunikation, also wie lange auf eine neue Nachricht gewartet wird. Die Funktion erhält zwei Eingaben: *timeOut* in Millisekunden für die EGM-Kommunikation und *timeOut_tsig* für die Testsignal-Kommunikation.
- *setMode*: Hiermit werden die Modi für die EGM- und Testsignal-Kommunikation eingestellt. Zu den Eingabewerten gehören:
  - Der Lese-Modus *new_readMode* für EGM, welcher den Modus *MODE* annehmen kann.
  - Der Schreib-Modus für EGM *new_writeMode*, welcher ebenfalls den Modus *MODE* annehmen kann.

  - Der Flag *new_enableTsigStreaming* zur Aktivierung des Testsignal-Streamings.
  - Der Modus für den Feedback-Zyklus *new_FeedbackCycleMode*, welcher einen Wert nach *UDP_MODE* annehmen kann.
  - Der Flag *new_enableNonBlockingMode* mit welchem der UDP-Sockets in den non-blocking Modus geschalten wird.

- *initRobot*: Hiermit wird die UDP-Kommunikation initialisiert, und die Anfangswerte (Position, Gelenkwinkel und Testsignalwerte) vom Roboter werden ausgelesen. Ebenfalls wird ein Fehlercode mit einer optionalen Fehlermeldung zurückgegeben.
- *WriteCycle*: Diese Funktion definiert den Schreibzyklus, der neue Positionen an den Roboter übermittelt. Sie muss in einer Schleife ausgeführt werden. Es gibt zwei Varianten dieser Funktion:
  1. Die erste Version sendet Positionsdaten ohne zusätzliche Eingaben und gibt bei einem Fehler eine Fehlermeldung sowie einen Fehlercode zurück.
  2. Die zweite Version erlaubt es, spezifische Positionsdaten über den Eingabewert *input_val* zu übergeben und ebenfalls Fehlermeldungen und Fehlercodes auszugeben.
- *FeedbackCycle*: Diese Funktion definiert den Lesezyklus der Daten des Roboters. Sie muss ebenfalls in einer Schleife ausgeführt werden. Es gibt zwei Varianten:
  1. Die erste Version speichert die Daten direkt in die Felder der *Robot* Instanz und gibt bei Bedarf eine Fehlermeldung und eine Feedback-Nachricht zurück. Der Rückgabewert zeigt an, ob ein Fehler bei der EGM-Kommunikation (1) oder bei den Testsignalen (2) aufgetreten ist.
  2. Die zweite Version gibt zusätzlich die gelesenen Daten als Rückgabewert *return_val* zurück. Der Fehlercode bleibt identisch.
- *cleanRobot*: Diese Funktion beendet die UDP-Kommunikation und gibt den belegten Speicher frei. Wie bei den anderen Funktionen kann eine Fehlermeldung ausgegeben werden, und der Rückgabewert zeigt an, ob ein Fehler aufgetreten ist (0 für keinen Fehler, 1 bei Fehler).

Tabelle A.5: Modus zur Spezifikation des Umganges mit dem UDP-Lesebuffer (UDP_MODE)

Modus	**Wert**	**Beschreibung**
NORMAL	0	Liest in jedem Zyklus das nächste Datagramm aus dem Puffer.
CLEAR_EACH_CYCLE	1	Stellt sicher, dass der Puffer in jedem Zyklus geleert wird, bevor das neueste Datagramm gelesen wird.
CLEAR_ONCE	2	Leert den Puffer einmal zu Beginn des Feedback-Zyklus, danach wird das nächste Datagramm in jedem Zyklus gelesen.

Tabelle A.6: Modus zur Spezifikation der gesendeten Daten (MODE)

Modus	**Wert**	**Beschreibung**
JOINT	0	Überträgt/empfängt nur Gelenkdaten.
JOINT_SPEED	1	Überträgt/empfängt Gelenkdaten und Gelenkgeschwindigkeiten.
JOINT_SPEED_ONLY	2	Überträgt/empfängt nur Gelenkgeschwindigkeiten.
POSE / POSE_EULER	3	Überträgt/empfängt nur kartesische Koordinaten und Euler-Winkel.
POSE_SPEED / POSE_EULER_SPEED	4	Überträgt kartesische Koordinaten, Euler-Winkel und Geschwindigkeiten, empfängt kartesische Koordinaten und Euler-Winkel.
POSE_SPEED_ONLY / POSE_EULER_SPEED_ONLY	5	Überträgt nur Geschwindigkeiten für kartesische Koordinaten und Euler-Winkel, empfängt kartesische Koordinaten und Euler-Winkel.
POSE_QUAT	6	Überträgt/empfängt nur kartesische Koordinaten und Quaternionen.
POSE_QUAT_SPEED	7	Überträgt kartesische Koordinaten, Quaternionen und Geschwindigkeiten, empfängt kartesische Koordinaten und Quaternionen.
POSE_QUAT_SPEED_ONLY	8	Überträgt nur Geschwindigkeiten für kartesische Koordinaten und Quaternionen, empfängt kartesische Koordinaten und Quaternionen.
SPEED	9	Überträgt/empfängt Gelenkgeschwindigkeiten und Geschwindigkeiten für kartesische Koordinaten sowie Euler-Winkel.
ALL	10	Überträgt/empfängt alle Daten: Gelenkpositionen, kartesische Koordinaten, Euler-Winkel, Quaternionen, Geschwindigkeiten, Gelenkgeschwindigkeiten und Motordrehmomente.

Tabelle A.7: Felder der *Robot* Klasse und deren Beschreibung

Kategorie	**Bezeichnung**	**Datentyp**	**Beschreibung**
Initiale Positionswerte	*RobotPos_init*	double[3]	Initiale Endeffektorposition (x, y, z in mm).
	RobotEuler_init	double[3]	Initiale Orientierung des Endeffektors (Euler-Winkel nach ZYX - Konvention in °).
	RobotQuaternions_init	double[4]	Initiale Orientierung des Endeffektors (Quaternionen).
	RobotJoint_init	double[6]	Initiale Gelenkposition (Gelenkwinkel $q_1 \dots q_6$ in °).
Gewünschte Positionswerte	*RobotPos_des*	double[3]	Gewünschte Endeffektorposition (x, y, z in mm).
	RobotEuler_des	double[3]	Gewünschte Orientierung des Endeffektors (Euler-Winkel nach ZYX - Konvention in °).
	RobotQuaternions_des	double[4]	Gewünschte Orientierung des Endeffektors (Quaternionen).
	RobotJoint_des	double[6]	Gewünschte Gelenkposition (Gelenkwinkel $q_1 \dots q_6$ in °).
Gewünschte Geschwindigkeitswerte	*RobotPosSpeed_des*	double[3]	Gewünschte Endeffektor-Geschwindigkeit (in mm/s).
	RobotEulerSpeed_des	double[3]	Gewünschte Winkelgeschwindigkeit (Euler-Winkel nach ZYX - Konvention in °/s).
	RobotQuaternionsSpeed_des	double[4]	Gewünschte Winkelgeschwindigkeit (Quaternionen).
	RobotJointSpeed_des	double[6]	Gewünschte Gelenkgeschwindigkeit (in °/s).
Geplante Positionswerte	*RobotPos_plan*	double[3]	Empfangene geplante Endeffektorposition (x, y, z in mm).
	RobotEuler_plan	double[3]	Empfangene geplante Orientierung (Euler-Winkel nach ZYX - Konvention in °).
	RobotQuaternions_plan	double[4]	Empfangene geplante Orientierung (Quaternionen).
	RobotJoint_plan	double[6]	Empfangene geplante Gelenkposition (Gelenkwinkel $q_1 \dots q_6$ in °).
Aktuelle Positionswerte	*RobotPos_curr*	double[3]	Empfangene aktuelle Endeffektorposition (x, y, z in mm).
	RobotEuler_curr	double[3]	Empfangene aktuelle Orientierung des Endeffektors (Euler-Winkel nach ZYX - Konvention in °).
	RobotQuaternions_curr	double[4]	Empfangene aktuelle Orientierung des Endeffektors (Quaternionen).
	RobotJoint_curr	double[6]	Empfangene aktuelle Gelenkposition (Gelenkwinkel $q_1 \dots q_6$ in °).
Testsignaldaten	*tsig_vals*	double[]	Empfangene Testsignalwerte.
	RobotTorque_curr	double[6]	Empfangene aktuelle Motor-Drehmomente (in Nm).
	RobotJointSpeed_curr	double[6]	Empfangene Gelenkgeschwindigkeit (in °/s).

Andere Roboterparameter	*RobotTime_curr*	double[4]	Zeitstempel der aktuellen Positionen (h, min, s, ms).
	RobotTime_plan	double[4]	Zeit für geplante Position (h, min, s, ms).
	SendTime_curr	double[4]	Zeitstempel der gesendeten Positionen (h, min, s, ms).
	TsigTime_curr	double	Zeitpunkt der letzten Testsignal-Messung (ms).
	nJoints	int	Anzahl der Gelenke.
	RobotMotorState	int	Motorzustand des Roboters (0 = undefiniert, 1 = an, 2 = aus).
	RobotEgmState	int	EGM-Zustand (0 = undefiniert, 1 = Fehler, 2 = Stopp, 3 = in Betrieb).
	RobotRapidState	int	Ausführungszustand des Programms (0 = undefiniert, 1 = gestoppt, 2 = in Betrieb).
	RobotConvergenceMet	bool	Flag, ob Konvergenzkriterien erreicht wurden.
	RobotUtilRate	double	Auslastungsrate des Roboters (in Prozent der benötigten Leistung).
	RobotCollision	int	Flag, ob eine Kollision erkannt wurde.

A.3.2 Simulink Block

Für eine Einbindung der EGM-Schnittstelle in Simulink wurde eine EGM-Block (dargestellt in Abbildung A.5) entwickelt, der auf einem S-Function Block basiert und auf die Roboterklasse Anhang A.3.1 zurückgreift, angelehnt an die Arbeit [38].

Dieser Block verfügt über die folgende Eingänge:

- *q_des* ... Vorgabe für die Gelenkposition in °
- *q_d_des* ... Vorgabe für die Gelenkgeschwindigkeit in °/s
- *z_IE_des* ... Vorgabe für die Pose mit der Position in mm und der Orientierung als Eulerwinkel (ZYX - Konvention) in °
- *z_d_IE_des* ... Vorgabe für die Geschwindigkeit mit der translatorischen Geschwindigkeit in mm/s und der Orientierungsänderung als Eulerwinkel (ZYX - Konvention) in °/s

Der Block besitzt außerdem die folgenden Ausgänge:

- *q* ... ausgelesene Gelenkposition in °
- *q_d* ... ausgelesene Gelenkgeschwindigkeit in °/s
- *z_IE* ... ausgelesene Pose mit der Position in mm und der Orientierung als Eulerwinkel (ZYX - Konvention) in °
- *M_mot* ... ausgelesene Motormomente in Nm
- *t_rob* ... Zeitpunkt (der Robotersteuerung), zu dem die EGM-Auslesung stattfand in ms
- *t_sent* ... Zeitpunkt (des PCs), zu dem die Vorgaben vom PC an die Steuerung gesendet wurden in ms

Je nach den getroffenen Einstellungen müssen nicht alle Eingänge verbunden sein, bzw. liefern nicht alle Ausgänge einen Wert. Diese Einstellungen können im Menü des Blocks (welches in Abbildung A.6 dargestellt ist) vorgenommen werden, das in zwei Bereiche unterteilt ist: *Settings* und *Parameters*.

Im Bereich *Settings* können einfache Klick-Boxen verwendet werden, um festzulegen, ob EGM-Streaming aktiv sein soll, ob neue Zielpositionen über EGM vorgegeben werden sollen oder ob Testsignale gestreamt werden sollen. Zusätzlich lässt sich über ein Drop-Down-Menü auswählen, welche Daten übertragen werden sollen (analog zu *MODE* in Tabelle A.6).

Im Bereich *Parameters* lassen sich die Übertragungsparameter festlegen. Diese sind die IP-Adresse (als String, z.B. '127.0.0.0'), die Ports und Time-Outs für die EGM- und Testsignal-Übertragung.

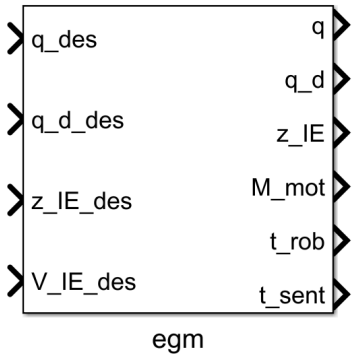


Abbildung A.5: Simulink EGM Block

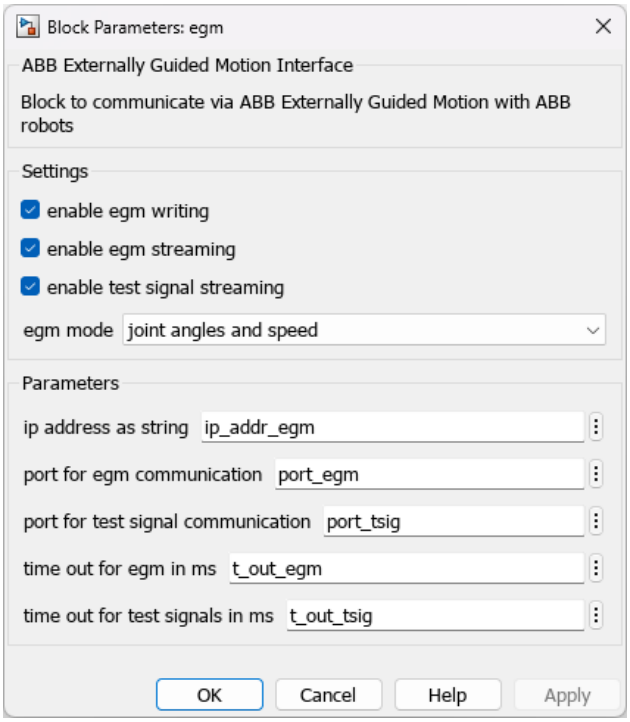


Abbildung A.6: Menü des Simulink EGM Blocks

A.4 Konfiguration des Qualisys Track Managers

Zum Konfigurieren muss zuerst der Roboter in Grundstellung $\mathbf{q}_0 = (0, 0, 0, 0, 0, 0)^T$ gebracht werden. Anschließend wird mit der Kamerakalibrierung begonnen.

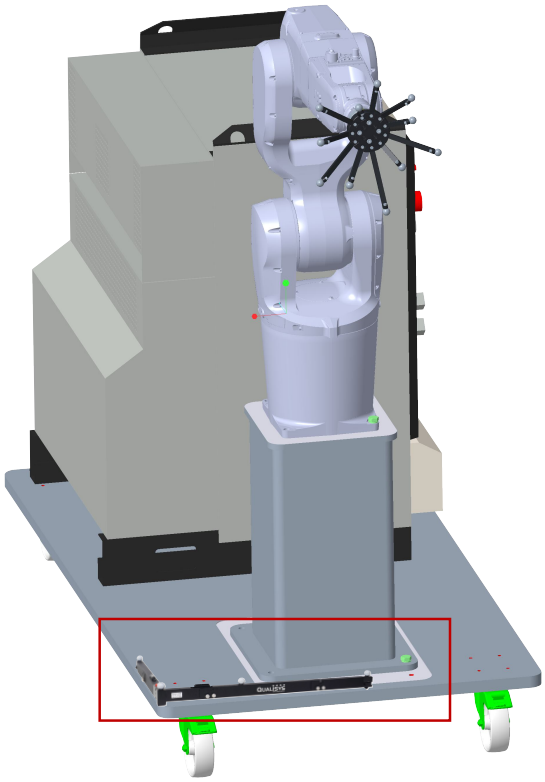


Abbildung A.7: Platzierung des L-Frames für die Kamera-Kalibrierung

Hierzu muss das mitgelieferte Koordinatenkreuz (L-Frame) am Fuß der Basis platziert werden, siehe Abbildung A.7. Mittels den integrierten Stellschrauben und Wasserwagen kann die Lage des Kreuz verändert werden, so dass diese parallel zum Boden ist. Anschließend wird der Kamera-Kalibiervorgang gestartet. Hierzu muss auf das Kalibriersymbol dargestellt in Abbildung A.8 geklickt werden.

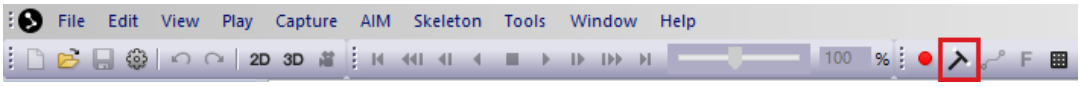


Abbildung A.8: Starten des Kamera-Kalibierungsvorganges

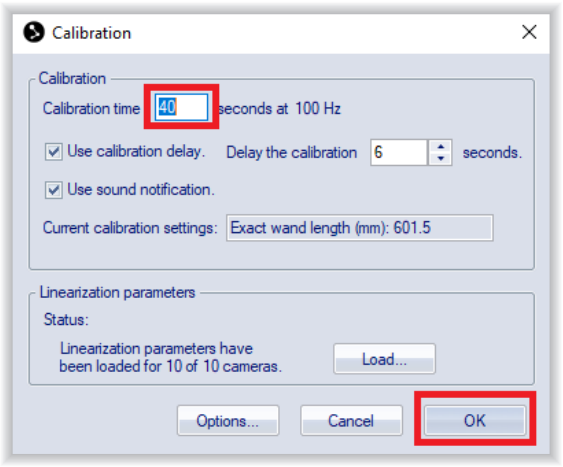


Abbildung A.9: Kalibrierungsmenü

Daraufhin öffnet sich das Kalibrierungsmenü siehe Abbildung A.9. Nach der Wahl der Kalibrierzeit (in dieser Arbeit wurden 40 s verwendet) kann die Kalibrierung gestartet werden und der Arbeitsraum muss mit dem Kalibrierstab abgefahren werden. Nach Ende des Erfassungszyklus führt QMT die Kamerakalibierung automatisch aus und liefert die Werte der Kalibrierung dargestellt in Abbildung A.10

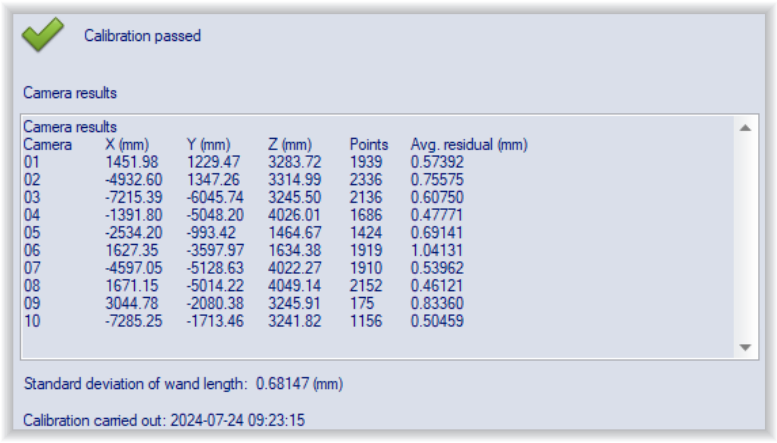


Abbildung A.10: Kalibrierungsergebnisse

Als nächstes muss die Standard-Orientierungsdarstellung auf die verwendeten Eulerwinkel nach ZYX-Konvention umgestellt werden. Hierzu wird das Einstellungsmenü geöffnet. Unter *Processing/Eulerangles* befindet sich die eingestellte Orientierungsdarstellung. Für Eulerwinkel nach ZYX-Konvention müssen die Felder *Custom*, *Local rotation axes* und die richtige Drehreihenfolge nach Abbildung A.11 gewählt werden.

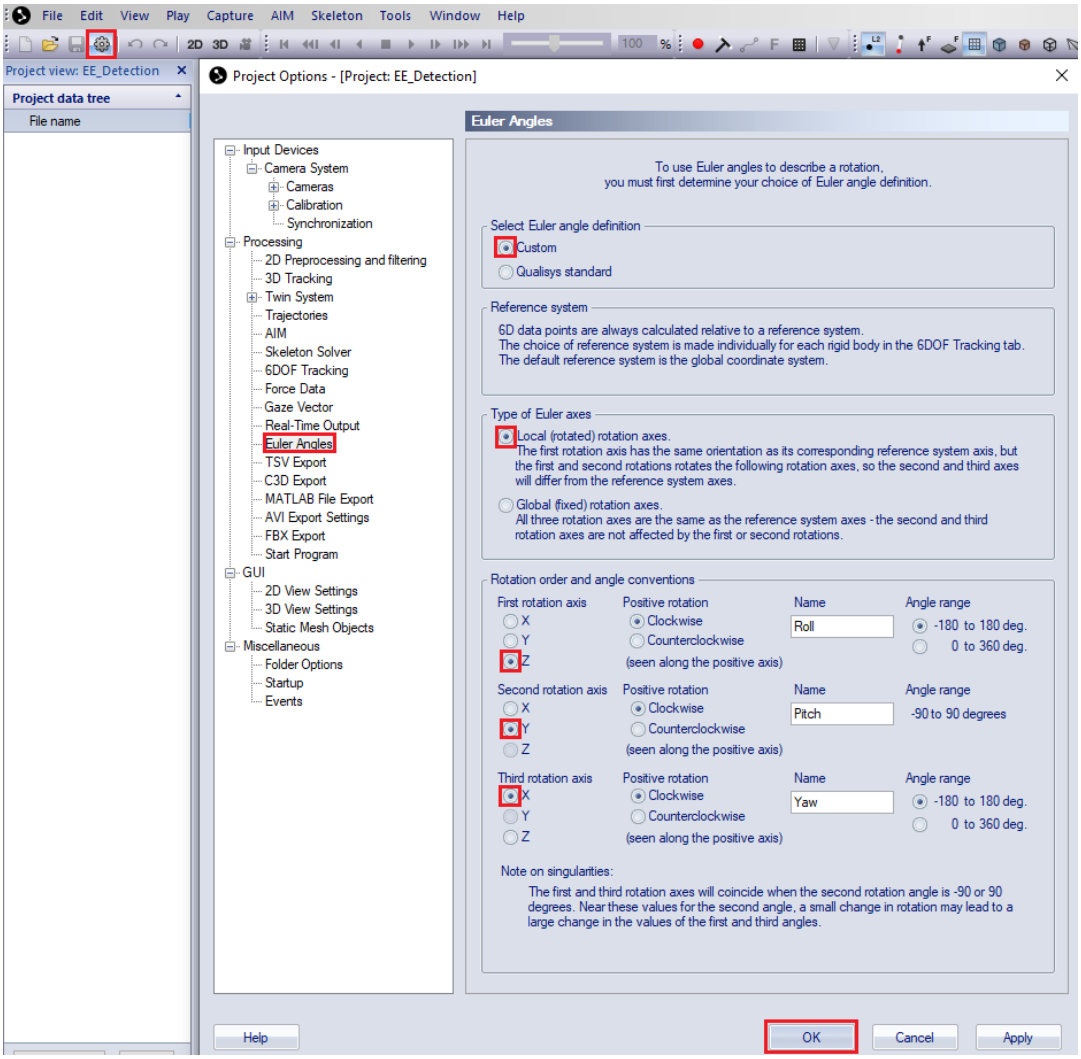


Abbildung A.11: Starten des Kamera-Kalibierungsvorganges

Anschließend können die Starrkörper definiert werden. In dieser Arbeit werden zwei Körper definiert zum einen das Koordinatenkreuz, stellvertretend für die Basis des Roboters und zum anderen der Abschnitt 6.2.1 beschriebene Detektionskörper des Endeffektors.

Die detektierten Reflektorkugeln erscheinen in der 3D-Ansicht als weiße Punkte. Mit *STRG* + Linksklick können mehrere Reflektorkugeln ausgewählt werden. Wenn alle Reflektoren eines Körpers gewählt wurden, kann mit Rechtsklick das Menü geöffnet werden wo mittels *Define rigid body (6DOF)* ein neuer Starrkörper erzeugt werden kann (Abbildung A.12). In dem anschließend erscheinenden Fenster muss die Bezeichnung des neuen Körpers eingetragen und mit *OK* bestätigt werden, daraufhin erscheint ein Skelett das die Punkte verbindet, sowie ein körperfestes Koordinatensystem.

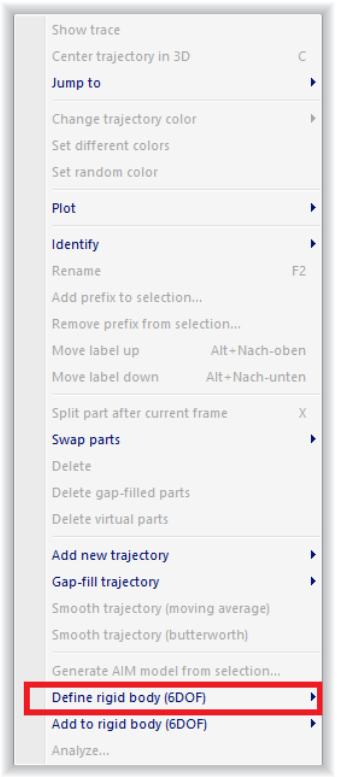


Abbildung A.12: Erstellen eines Starrkörpers

Abbildung A.13: Änderungen des körperfesten Koordinatensystems des Starrkörpers

Nun muss das Koordinatensystem noch ausgerichtet werden, hierzu wird dieses zuerst im Menüpunkt Einstellungen unter *Processing/6DOF Tracking* durch Wahl von *Reset Rotation* siehe Abbildung A.13 zurückgesetzt werden. Anschließend wird mit *Rotate* das Koordinatensysteme in die selbe Orientierung wie in Abbildung 2.1 dargestellt gebracht. Das Koordinatensystem des Detektionskörpers muss hierfür zuerst um $-90°$ um die z-Achse und anschließend um $90°$ um die y-Achse gedreht werden, die Basis um $-90°$ um die z-Achse. In dem erscheinenden Fenster muss hierfür für jede Drehung *Rotate the System* ausgewählt werden, die Achse und der Winkel angegeben werden und mit *OK* bestätigt werden, dargestellt in Abbildung A.14.

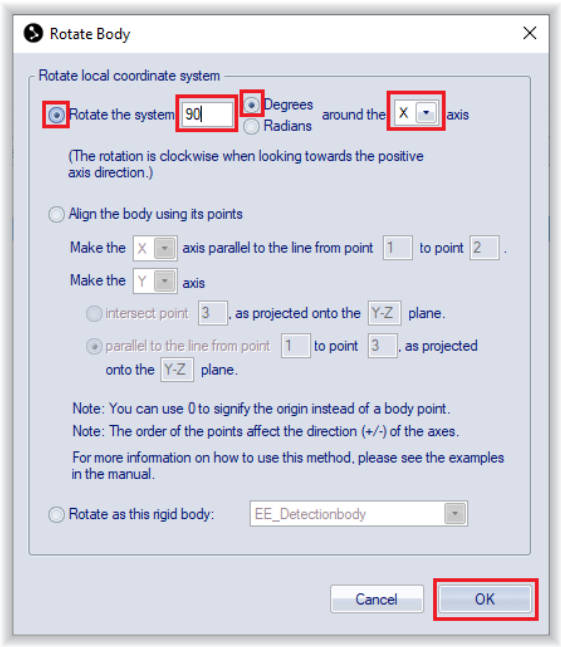


Abbildung A.14: Verdrehung des körperfesten Koordinatensystems

Als nächstes wird die aktuelle Position des jeweiligen Koordinatensystems ermittelt hierzu unter *Coordinate System* die Option *Use this coordinate system (relative the global coordinate system)* wählen und dort mit *Get position* die aktuelle Position ermitteln (Abbildung A.15). Aus diesen Werten wird die notwendige Verschiebung berechnet basierend auf der nominellen Verschiebung laut den CAD-Daten und der Platzierung des L-Frames. Für den Detektionskörper gilt

$$\begin{pmatrix} x_{detect} \\ y_{detect} \\ z_{detect} \end{pmatrix} = \begin{pmatrix} z_m - 1289.1 \\ 400 - x_m \\ y_m - 168 \end{pmatrix} \tag{A.1}$$

und für die Basis

$$\begin{pmatrix} x_{base} \\ y_{base} \\ z_{base} \end{pmatrix} = \begin{pmatrix} y_m + 265 \\ 400 - x_m \\ 498 - z_m \end{pmatrix} . \tag{A.2}$$

Nach der Messung werden für den Detektionskörper die Werte 400, 265, 498 für die Position und -90, 0, 0 für die Orientierung eingetragen und mit *OK* bestätigt. Für die Basis das selbe, dadurch wird die Verschiebung des in dieser Arbeit verwendeten Inertialsystems relativ zum Inertialsystem von Qualisys festgelegt.

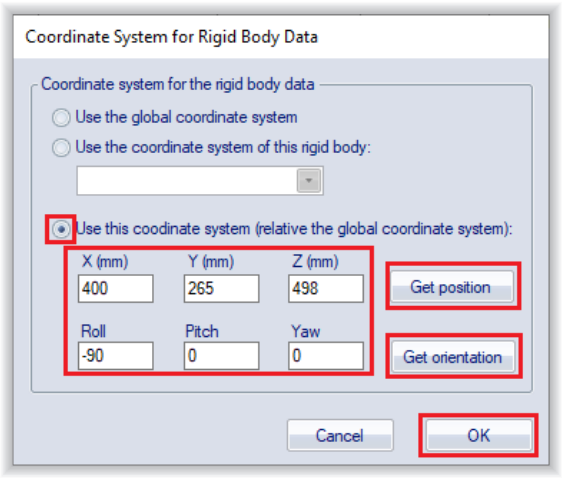


Abbildung A.15: Verschiebung des Inertialsystems

Für die relative Verschiebung der körperfesten Koordinatensysteme müssen die soeben ermittelten Werte aus (A.2) und (A.1) unter *Translate* (siehe Abbildung A.13) in der Option *To local coordinates* eingetragen werden (Abbildung A.16) und mit *OK* bestätigt werden.

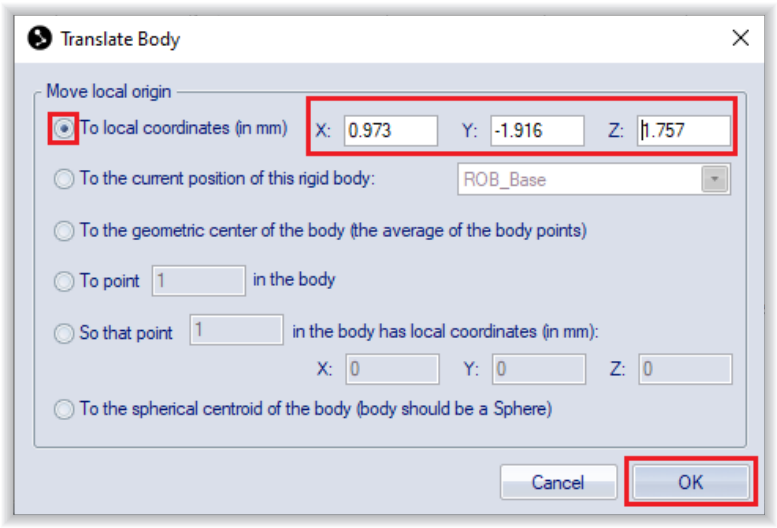


Abbildung A.16: Verschiebung des körperfesten Koordinatensystems

Der fertige angelegte Detektionskörper und die Roboterbasis in der Grundposition des Roboters ist in der Abbildung A.17 dargestellt. Hier sind die Skelette der beiden Körper, sowie ihre körperfesten Koordinatensystem dargestellt und das Inertialsystem von Qualisys.

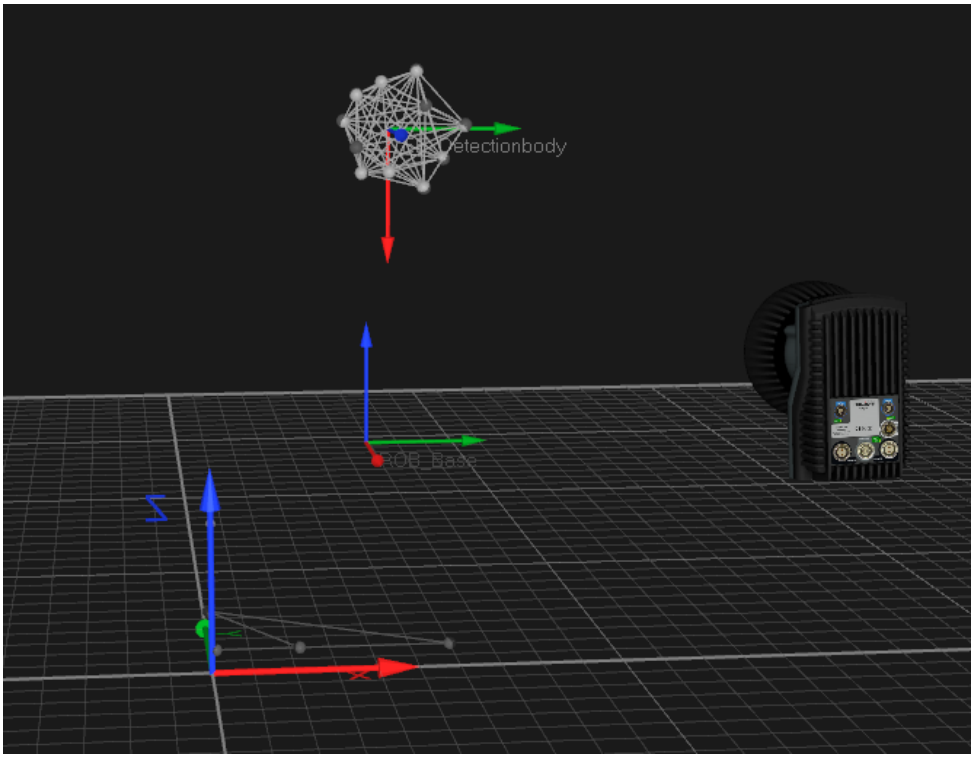


Abbildung A.17: Darstellung der Koordinatensysteme des Detektionskörpers und der Roboterbasis so wie des Inertialsystems

Des Weiteren lassen sich im Menüpunkt Einstellungen unter *Processing/6DOF Tracking* die Eigenschaften für die Erkennung der Körper festlegen. Für die Roboterbasis macht es Sinn hier unter *Filter Static pose* zu wählen, um eine robustere Erfassung zu ermöglichen. Zusätzlich kann die minimal notwendige Anzahl an Reflektoren damit der Körper erkannt wird unter *Min. markers* festgelegt werden, sowie die Genauigkeit der identifizierten Abstände zwischen den einzelnen Reflektoren in *Bone Tolerance*.

Label	X / Color	Y / Origin	Z / Orientation	Virtual / Min. markers	Id / Max. residual	Bone tolerance	Filter
⊞ EE_Detectio...		Fixed	Fixed	3	10.00	0.50	No filter
⊞ ROB_Base		Fixed	Fixed	3	10.00	5.00	Static pose

Abbildung A.18: Eigenschaften der Erkennung der Körper

Im nächsten Schritt werden die Kameraparameter eingestellt. Hierzu muss auf den 2D-Modus gewechselt werden (dies erfolgt mit dem in Abbildung A.19 dargestellten Knopf mit dem zwischen 2D und 3D Modus gewechselt werden kann). In diesem wird ein Feld zur Einstellung der Kameraparameter für jede verbundene Qualisyskamera angezeigt als Beispiel Abbildung A.20. Unter *Marker* sind die Einstellungen für die Markerdetektion zu treffen. Die in dieser Arbeit getroffenen Einstellungen sind in Tabelle A.8 zu finden.

Abbildung A.19: Navigationsmenü

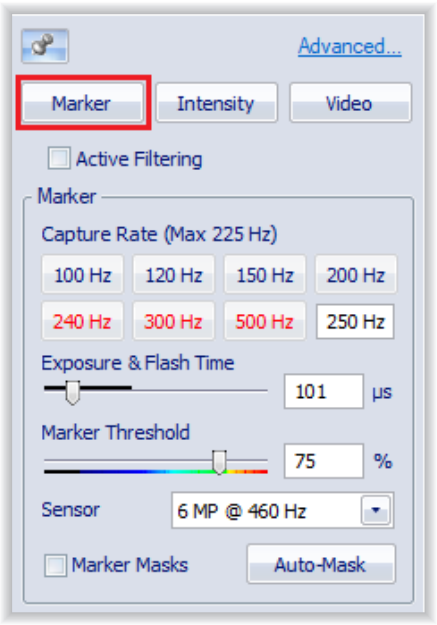


Abbildung A.20: Konfiguration der Kamera (Oqus 1)

In dieser Ansicht kann auch für jede Kamera ein Fenster festgelegt werden in dem sich der Marker befinden muss. Dies ist nützlich, um Störquellen vorzeitig auszufiltern und die Berechnung zu beschleunigen. Hierzu wird der in Abbildung A.19 dargestellte Windowing Befehl genutzt. Die in dieser Arbeit festgelegten Bereiche sind in Tabelle A.8 dargestellt, wobei immer der untere und der obere Bildpunkt des analysierten Bildausschnittes angegeben werden. Diese ergeben sich aus dem Arbeitsraum des Roboters.

Tabelle A.8: Kameraeinstellungen der Oqus 6+ Kameras in QTM

ID	**Framerate** in Hz	**Belichtungs-zeit** in µs	**Erkennungs-threshold** in %	**Bildausschnitt** in px (gesamt 3071 × 1983)
1	250	101	75	$\begin{pmatrix}192\\329\end{pmatrix} - \begin{pmatrix}3071\\1983\end{pmatrix}$
2	250	276	65	$\begin{pmatrix}0\\0\end{pmatrix} - \begin{pmatrix}1407\\1663\end{pmatrix}$
3	250	400	30	$\begin{pmatrix}864\\208\end{pmatrix} - \begin{pmatrix}2559\\1199\end{pmatrix}$
4	250	400	62	$\begin{pmatrix}1440\\0\end{pmatrix} - \begin{pmatrix}3071\\1375\end{pmatrix}$
5 M	250	171	61	$\begin{pmatrix}0\\0\end{pmatrix} - \begin{pmatrix}3071\\1983\end{pmatrix}$
6	250	66	75	$\begin{pmatrix}480\\112\end{pmatrix} - \begin{pmatrix}3071\\1887\end{pmatrix}$
7	250	398	62	$\begin{pmatrix}512\\0\end{pmatrix} - \begin{pmatrix}2623\\1455\end{pmatrix}$
8	250	400	84	$\begin{pmatrix}1248\\176\end{pmatrix} - \begin{pmatrix}3071\\1967\end{pmatrix}$
9	250	162	61	$\begin{pmatrix}1536\\768\end{pmatrix} - \begin{pmatrix}3071\\1983\end{pmatrix}$
10	250	400	44	$\begin{pmatrix}0\\0\end{pmatrix} - \begin{pmatrix}1759\\1151\end{pmatrix}$

Zum Schluss muss noch die Matlab-Schnittstelle konfiguriert werden. Hierzu muss man im Menüpunkt Einstellungen unter *Processing/MATLAB File Export* das Feld *6D data* wählen, siehe Abbildung A.21. (Die Auswahl von *Timestamps* ist nicht notwendig da diese Information automatisch mit *Frameinfo* übertragen wird.)

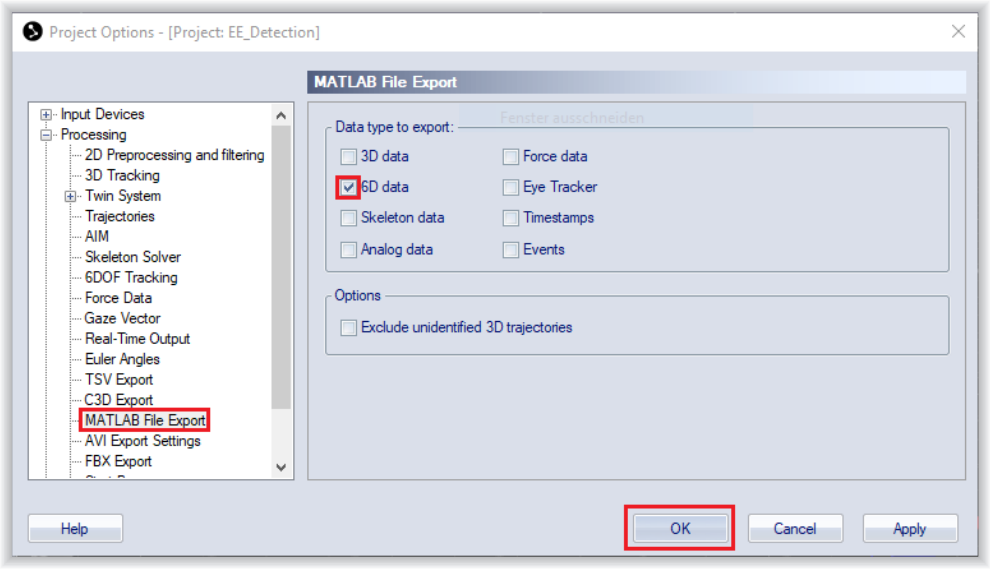


Abbildung A.21: Konfiguration der Schnittstelle von QTM zu Matlab

A.5 DetMax - Algorithmus

Im Folgenden wird der in der Arbeit verwendete DetMax - Algorithmus detailliert in Form eines Flussdiagramms dargestellt.

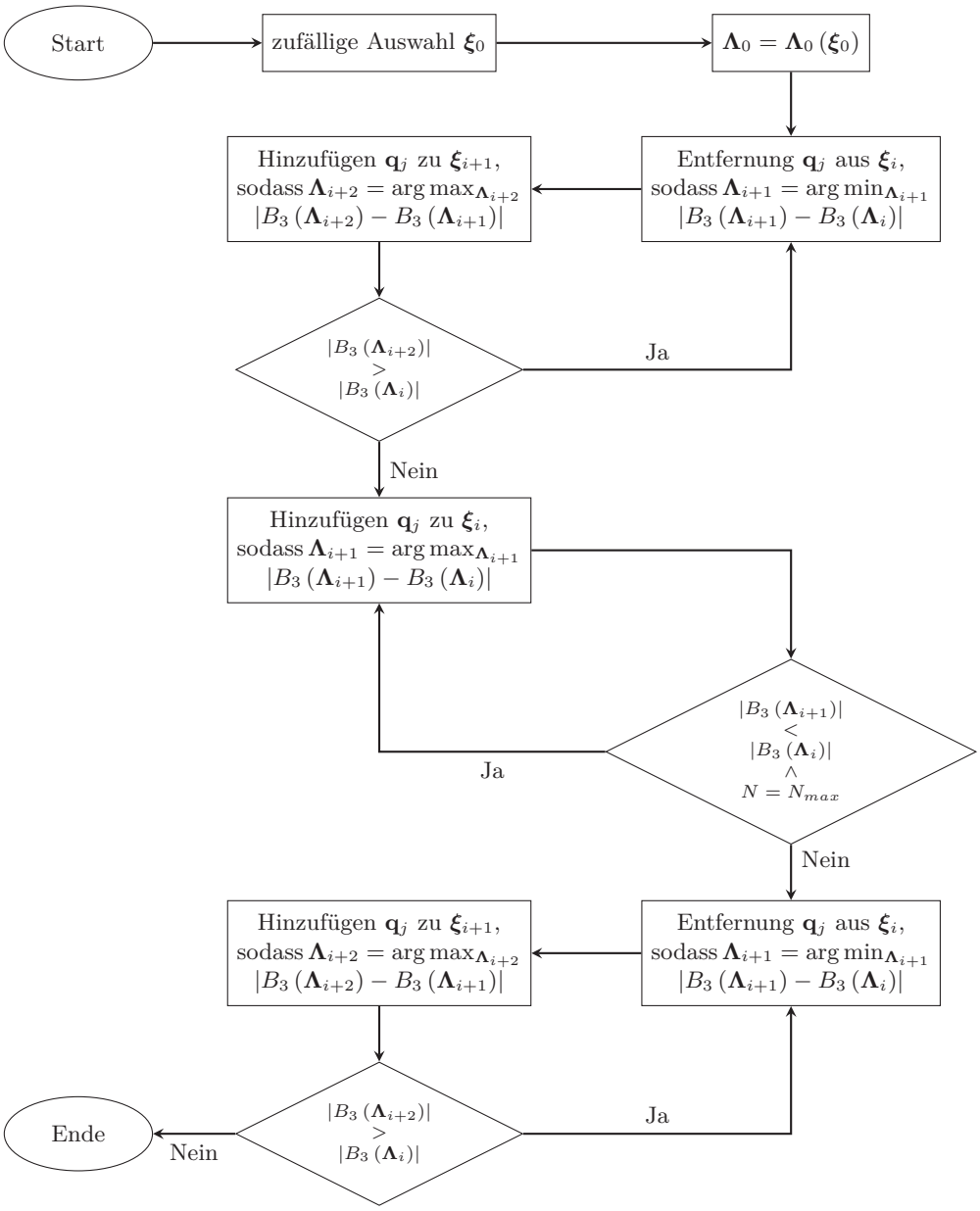


Abbildung A.22: Ablauf Ermittlung optimale Kalibrierungskonfigurationen

A.6 Struktur tiny YOLO-Netz

In diesem Abschnitt wird anhand der Tabelle A.9 detailliert die Struktur des tiny YOLO-Netzes, das in dieser Arbeit verwendet wurde, beschrieben.

Tabelle A.9: Struktur des tiny YOLO-Netzwerks

Layer	type	Padding	Size / Stride	Scale	Input-Size	Output - Size
1	2-D Convolution	same	3×3/1	-	416×416×3	416×416×16
2	Batch Normalization	-	-	-	416×416×16	416×416×16
3	Leaky ReLU	-	-	0.1	416×416×16	416×416×16
4	2-D Max Pooling	same	2×2/2	-	416×416×16	208×208×16
5	2-D Convolution	same	3×3/1	-	208×208×16	208×208×32
6	Batch Normalization	-	-	-	208×208×32	208×208×32
7	Leaky ReLU	-	-	0.1	208×208×32	208×208×32
8	2-D Max Pooling	same	2×2/2	-	208×208×32	104×104×32
9	2-D Convolution	same	3×3/1	-	104×104×32	104×104×64
10	Batch Normalization	-	-	-	104×104×64	104×104×64
11	Leaky ReLU	-	-	0.1	104×104×64	104×104×64
12	2-D Max Pooling	same	2×2/2	-	104×104×64	52×52×64
13	2-D Convolution	same	3×3/1	-	52×52×64	52×52×128
14	Batch Normalization	-	-	-	52×52×128	52×52×128
15	Leaky ReLU	-	-	0.1	52×52×128	52×52×128
16	2-D Max Pooling	same	2×2/2	-	52×52×128	26×26×128
17	2-D Convolution	same	3×3/1	-	26×26×128	26×26×256
18	Batch Normalization	-	-	-	26×26×256	26×26×256
19	Leaky ReLU	-	-	0.1	26×26×256	26×26×256
20	2-D Max Pooling	same	2×2/2	-	26×26×256	13×13×256
21	2-D Convolution	same	3×3/1	-	13×13×256	13×13×512
22	Batch Normalization	-	-	-	13×13×512	13×13×512
23	Leaky ReLU	-	-	0.1	13×13×512	13×13×512
24	2-D Max Pooling	same	2×2/1	-	13×13×512	13×13×512
25	2-D Convolution	same	3×3/1	-	13×13×512	13×13×1024
26	Batch Normalization	-	-	-	13×13×1024	13×13×1024
27	Leaky ReLU	-	-	0.1	13×13×1024	13×13×1024
28	2-D Convolution	same	1×1/1	-	13×13×1024	13×13×256
29	Batch Normalization	-	-	-	13×13×256	13×13×256
30	Leaky ReLU	-	-	0.1	13×13×256	13×13×256
31.1	2-D Convolution	same	3×3/1	-	13×13×256	13×13×512
32.1	Batch Normalization	-	-	-	13×13×512	13×13×512
33.1	Leaky ReLU	-	-	0.1	13×13×512	13×13×512
34.1	2-D Convolution	same	1×1/1	-	13×13×512	13×13×255
31.2	2-D Convolution	same	1×1/1	-	13×13×256	13×13×128
32.2	Batch Normalization	-	-	-	13×13×128	13×13×128
33.2	Leaky ReLU	-	-	0.1	13×13×128	13×13×128
34.2	Resize	-	-	-	13×13×128	26×26×128
35.2	Depth concatenation	-	-	-	26×26×128 26×26×256	26×26×384
36.2	2-D Convolution	same	3×3/1	-	26×26×384	26×26×256
37.2	Batch Normalization	-	-	-	26×26×256	26×26×256
38.2	Leaky ReLU	-	-	0.1	26×26×256	26×26×256
39.2	2-D Convolution	same	1×1/1	-	26×26×256	26×26×255

Anhang B

Ergänzungen zum Roboter

B.1 Robotergrenzen

Der verwendete ABB IRB1200 Roboter weist die in Tabelle B.1 dargestellten Gelenkwinkel- und Gelenkgeschwindigkeitsgrenzen auf. Diese wurden aus dem Produkthandbuch [6] entnommen.

Tabelle B.1: Gelenkwinkel- und Gelenkgeschwindigkeitsgrenzen

Beschränkte Größe	Untergrenze	Obergrenze	Einheit
q_1	-170	170	$^\circ$
q_2	-100	135	$^\circ$
q_3	-200	70	$^\circ$
q_4	-270	270	$^\circ$
q_5	-130	130	$^\circ$
q_6	-400	400	$^\circ$
$\dot{q}_1$	-288	288	$^\circ/\mathrm{s}$
$\dot{q}_2$	-240	240	$^\circ/\mathrm{s}$
$\dot{q}_3$	-297	297	$^\circ/\mathrm{s}$
$\dot{q}_4$	-400	400	$^\circ/\mathrm{s}$
$\dot{q}_5$	-405	405	$^\circ/\mathrm{s}$
$\dot{q}_6$	-600	600	$^\circ/\mathrm{s}$

B.2 Dynamische Parameter

In folgender Tabelle sind die verwendeten Parameter für die Dynamik dargestellt. Die Darstellung der Reibungskoeffizienten erfolgt motorseitig. Die Werte entstammen zum Teil aus Annahmen zum anderen wurden jene Werte aus [35] verwendet.

Tabelle B.2: Parameter des dynamischen Modells des IRB1200

Parameter	Beschreibung	Wert	Einheit
i_{G_1}	Getriebeübersetzung Achse 1	66.6111	
i_{G_2}	Getriebeübersetzung Achse 2	107.815	
i_{G_3}	Getriebeübersetzung Achse 3	53.7073	
i_{G_4}	Getriebeübersetzung Achse 4	60.0364	
i_{G_5}	Getriebeübersetzung Achse 5	69.923	
i_{G_6}	Getriebeübersetzung Achse 6	78.686	
m_0	Masse Sockel	15.57	kg
m_{A_1}	Masse Arm 1	13	kg
C_{A_1}	Trägheit in z von Arm 1	0.67	kgm^2
C_{M_1}	Trägheit in z von Motor 1	$200 \cdot 10^{-6}$	kgm^2
m_{A_2}	Masse Arm 2	17.4	kg
A_{A_2}	Trägheit in x von Arm 2	0.45	kgm^2
B_{A_2}	Trägheit in y von Arm 2	0.539	kgm^2
C_{A_2}	Trägheit in z von Arm 2	0.764	kgm^2
B_{M_2}	Trägheit in y von Motor 2	$200 \cdot 10^{-6}$	kgm^2
m_{A_3}	Masse Arm 3	4.8	kg
A_{A_3}	Trägheit in x von Arm 3	0.006	kgm^2
B_{A_3}	Trägheit in y von Arm 3	0.067 25	kgm^2
C_{A_3}	Trägheit in z von Arm 3	0.07	kgm^2
B_{M_3}	Trägheit in y von Motor 3	$200 \cdot 10^{-6}$	kgm^2
m_{A_4}	Masse Arm 4	0.82	kg
A_{A_4}	Trägheit in x von Arm 4	$1.3 \cdot 10^{-3}$	kgm^2
B_{A_4}	Trägheit in y von Arm 4	$1.8 \cdot 10^{-3}$	kgm^2
C_{A_4}	Trägheit in z von Arm 4	$1.8 \cdot 10^{-3}$	kgm^2
A_{M_4}	Trägheit in x von Motor 4	$33 \cdot 10^{-6}$	kgm^2
m_{A_5}	Masse Arm 5	0.32	kg
A_{A_5}	Trägheit in x von Arm 5	$0.3 \cdot 10^{-3}$	kgm^2
B_{A_5}	Trägheit in y von Arm 5	$0.4 \cdot 10^{-3}$	kgm^2
C_{A_5}	Trägheit in z von Arm 5	$0.3 \cdot 10^{-3}$	kgm^2
A_{M_5}	Trägheit in x von Motor 5	$33 \cdot 10^{-6}$	kgm^2
m_{A_6}	Masse Arm 6	0.09	kg
A_{A_6}	Trägheit in x von Arm 6	$0.15 \cdot 10^{-3}$	kgm^2
B_{A_6}	Trägheit in y von Arm 6	$0.15 \cdot 10^{-3}$	kgm^2
C_{A_6}	Trägheit in z von Arm 6	$0.15 \cdot 10^{-3}$	kgm^2
A_{M_6}	Trägheit in x von Motor 6	$33 \cdot 10^{-6}$	kgm^2
m_L	Masse Last	0 - 7	kg
A_L	Trägheit in x von Last	0.1	kgm^2
B_L	Trägheit in y von Last	0.1	kgm^2
C_L	Trägheit in z von Last	0.1	kgm^2
g	Gravitationskonstante	9.81	$\mathrm{m/s}^2$
d_{v_1}	Gleitreibungskoeffizient Achse 1	$1.48 \cdot 10^{-3}$	Nms/rad
d_{v_2}	Gleitreibungskoeffizient Achse 2	$1.48 \cdot 10^{-3}$	Nms/rad

d_{v_3}	Gleitreibungskoeffizient Achse 3	$1.48 \cdot 10^{-3}$	Nms/rad
d_{v_4}	Gleitreibungskoeffizient Achse 4	$1.48 \cdot 10^{-3}$	Nms/rad
d_{v_5}	Gleitreibungskoeffizient Achse 5	$1.48 \cdot 10^{-3}$	Nms/rad
d_{v_6}	Gleitreibungskoeffizient Achse 6	$1.48 \cdot 10^{-3}$	Nms/rad
d_{c_1}	Coulombreibungskoeffizient Achse 1	$4.15 \cdot 10^{-3}$	Nm
d_{c_2}	Coulombreibungskoeffizient Achse 2	$4.15 \cdot 10^{-3}$	Nm
d_{c_3}	Coulombreibungskoeffizient Achse 3	$4.15 \cdot 10^{-3}$	Nm
d_{c_4}	Coulombreibungskoeffizient Achse 4	$4.15 \cdot 10^{-3}$	Nm
d_{c_5}	Coulombreibungskoeffizient Achse 5	$4.15 \cdot 10^{-3}$	Nm
d_{c_6}	Coulombreibungskoeffizient Achse 6	$4.15 \cdot 10^{-3}$	Nm
ϵ	Übergang von Coulombreibung	$1 \cdot 10^{-2}$	rad/s
c_{G_1}	Federkonstante Getriebeelastizität Achse 1	25 000	Nm/rad
c_{G_2}	Federkonstante Getriebeelastizität Achse 2	45 000	Nm/rad
c_{G_3}	Federkonstante Getriebeelastizität Achse 3	45 000	Nm/rad
c_{G_4}	Federkonstante Getriebeelastizität Achse 4	45 000	Nm/rad
c_{G_5}	Federkonstante Getriebeelastizität Achse 5	45 000	Nm/rad
c_{G_6}	Federkonstante Getriebeelastizität Achse 6	45 000	Nm/rad
d_{G_1}	Dämpferkonstante Getriebeelastizität Achse 1	80	Nms/rad
d_{G_2}	Dämpferkonstante Getriebeelastizität Achse 2	80	Nms/rad
d_{G_3}	Dämpferkonstante Getriebeelastizität Achse 3	80	Nms/rad
d_{G_4}	Dämpferkonstante Getriebeelastizität Achse 4	80	Nms/rad
d_{G_5}	Dämpferkonstante Getriebeelastizität Achse 5	80	Nms/rad
d_{G_6}	Dämpferkonstante Getriebeelastizität Achse 6	80	Nms/rad

B.3 Testsignale

In Tabelle B.3 werden die in ABB Rapid definierten Testsignale beschrieben. Alle Daten werden hier auf der Armseite gemessen und stehen für alle Roboterachsen als auch externe Achsen zur Verfügung, wenn nicht anders gekennzeichnet.

Tabelle B.3: Auflistung der Testsignale von ABB

Art des Signals	ABB Testsignal ID	Einheit
Gelenkgeschwindigkeit externer Achsen	6	rad/s
Soll-Motormoment externer Achsen	9	Nm
Gelenkwinkel externer Achsen	1	rad
Soll-Gelenkgeschwindigkeit externer Achsen	4	rad/s
Digitaler Eingang 1 externer Achsen	102	boolean
Digitaler Eingang 2 externer Achsen	103	boolean
Position[1]	4000	° oder mm
Geschwindigkeit[2]	4001	°/s oder mm/s
Motormoment	4002	Nm
Geschätztes, von außen aufgebrachtes Drehmoment	4003	Nm

B.4 Adressenbelegung

Im Folgenden wird auf die Adressenbelegung für die verschiedenen Geräte eingegangen.

Tabelle B.4: Verwendete IP-Adressen und Ports

Verwendung	Gerät	IP-Adresse	Port
EGM Kommunikation (Simulation in RS)	PC	127.0.0.1	5514
Testsignal Kommunikation (Simulation in RS)	PC	127.0.0.1	5515
EGM Kommunikation	IRC5	192.168.125.1	5516
Testsignal Kommunikation	IRC5	192.168.125.1	5517
EGM Kommunikation	PC	192.168.125.211	5516
Testsignal Kommunikation	PC	192.168.125.211	5517
Kommunikation mit Kameras	PC	192.168.254.1	-

[1] je nach Einstellung als Gelenkswinkel oder Pose
[2] je nach Einstellung als Gelenkgeschwindigkeit oder Twist

Literatur

[1] Hexagon AB. *Leica Absolute Tracker AT960*. URL: https://hexagon.com/products/leica-absolute-tracker-at960 (besucht am 28. 10. 2024).

[2] Qualisys AB. *Qualisys User Manual*. Version 2020.3. Qualisys AB, Nov. 2020. URL: www.qualisys.com (besucht am 10. 09. 2024).

[3] Qualisys AB. *Long range active mocap marker*. URL: https://www.qualisys.com/accessories/markers/long-range-active-marker/ (besucht am 28. 10. 2024).

[4] Qualisys AB. *Super-spherical mocap markers*. URL: https://www.qualisys.com/accessories/markers/super-spherical-markers/ (besucht am 28. 10. 2024).

[5] ABB. *Anwendungshandbuch Externally Guided Motion*. 3HAC073319-003. Version E. ABB, Dez. 2021. URL: https://library.e.abb.com/public/c20fb51c2fc649e48397b81506a86a8c/3HAC073319%20AM%20Externally%20Guided%20Motion%20RW6-de.pdf (besucht am 10. 09. 2024).

[6] ABB. *Produkthandbuch IRB1200*. 3HAC046983-003. Version V. ABB, Sep. 2022. URL: https://search.abb.com/library/Download.aspx?DocumentID=3HAC046983-003&LanguageCode=de&DocumentPartId=&Action=Launch (besucht am 11. 09. 2024).

[7] K. S. Arun, T. S. Huang und S. D. Blostein. „Least-Squares Fitting of Two 3-D Point Sets“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.5 (1987), S. 698–700. URL: https://ieeexplore.ieee.org/abstract/document/4767965.

[8] A. Brennsteiner. „Geometrische Roboterkalibrierung mit optischen Methoden“. Masterarbeit. Institut für Robotik, Johannes Kepler Universität Linz, 2016. URL: https://resolver.obvsg.at/urn:nbn:at:at-ubl:1-13148.

[9] L. Calvet, P. Gurdjos und V. Charvillat. „Camera tracking using concentric circle markers: Paradigms and algorithms“. In: *2012 19th IEEE International Conference on Image Processing*. 2012, S. 1361–1364.

[10] J. DeGol, T. Bretl und D. Hoiem. „ChromaTag: A Colored Marker and Fast Detection Algorithm“. In: *CoRR* abs/1708.02982 (2017). arXiv: 1708.02982. URL: http://arxiv.org/abs/1708.02982.

[11] K. Dorfmüller-Ulhaas. „Optical Tracking-From User Motion To 3D Interaction“. Diss. Technische Universität Wien, 2002. URL: https://diglib.eg.org/items/86afd6d5-8334-4455-99c8-694ae1c17d43.

[12] I. Eldor u. a. „A Vision-Based System for Inspection of Expansion Joints in Concrete Pavement“. In: *Preprints* (2023). URL: https://doi.org/10.20944/preprints202305.0701.v1.

[13] Faro. *Infos über tragbare Messarme*. URL: https://www.faro.com/de-DE/Resource-Library/Article/understanding-portable-measurement-arms (besucht am 28. 10. 2024).

[14] M. Fiala. „ARTag, a fiducial marker system using digital techniques“. In: Bd. 2. 2005, 590–596 vol. 2. ISBN: 0-7695-2372-2.

[15] W. Gander. „Algorithms for the QR-Decomposition“. In: *research report no. 80-02* (1980). URL: https://people.inf.ethz.ch/gander/papers/qrneu.pdf.

[16] H. Gattringer. *Vorlesungsskriptum Robotik*. Institut für Robotik, Johannes Kepler Universität Linz, 2022.

[17] H. Gattringer. *Vorlesungsskriptum Steuerung und Regelung von Robotern II*. Institut für Robotik, Johannes Kepler Universität Linz, 2022.

[18] Boulder Innovation Group Inc. *3D Creator product information*. URL: https://boulderinnovators.com/products/product-information/ (besucht am 28. 10. 2024).

[19] The MathWorks Inc. *yolov3ObjectDetector*. URL: %5Curl%7Bhttps://de.mathworks.com/help/vision/ref/yolov3objectdetector.html%7D (besucht am 10. 09. 2024).

[20] Q. Jia u. a. „A novel optimal design of measurement configurations in robot calibration“. In: *Mathematical Problems in Engineering* 2018.1 (2018), S. 4689710. URL: https://doi.org/10.1155/2018/4689710.

[21] M. Kallinger. „Vergleich unterschiedlicher geometrischer Kalibrierungsansätze für Sechsachsroboter“. Masterarbeit. Institut für Robotik, Johannes Kepler Universität Linz, 2023. URL: https://resolver.obvsg.at/urn:nbn:at:at-ubl:1-70240.

[22] M. Kaltenbrunner und R. Bencina. „reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction“. In: *Proceedings of the first international conference on Tangible and Embedded Interaction (TEI07)* (2007). URL: https://reactivision.sourceforge.net/ (besucht am 28. 10. 2024).

[23] K. Kanatani. „Analysis of 3-D rotation fitting“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16.5 (1994), S. 543–549. URL: https://ieeexplore.ieee.org/abstract/document/291441.

[24] J . Köhler, A. Pagani und D. Stricker. „Detection and Identification Techniques for Markers Used in Computer Vision.“ In: Bd. 19. 2010, S. 36–44.

[25] T. Kubela, A. Pochyly und V. Singule. „Investigation of position accuracy of industrial robots and online methods for accuracy improvement in machining processes“. In: *2015 International Conference on Electrical Drives and Power Electronics (EDPE)* (2015), S. 385–388. URL: https://ieeexplore.ieee.org/abstract/document/7325325.

[26] Y. Liu u. a. „Improvement of Robot Accuracy with an Optical Tracking System“. In: *Sensors* 20.21 (2020). ISSN: 1424-8220. URL: https://www.mdpi.com/1424-8220/20/21/6341.

[27] Agisoft LLC. *Tutorial (Intermediate level): Coded Targets and Scale Bars in Agisoft PhotoScan Pro 1.1*. URL: https://www.agisoft.com/pdf/PS_1.1_Tutorial%20(IL)%20-%20Coded%20Targes%20and%20Scale%20Bars.pdf (besucht am 28. 10. 2024).

[28] Google LLC. *Official Google Protocol Buffers Documentation*. URL: https://protobuf.dev/ (besucht am 10. 09. 2024).

[29] Fill Gesellschaft m.b.H. *Accubot - NDT SYSTEMS*. URL: https://www.fill.co.at/de/produkte/accubot-ndt (besucht am 28. 10. 2024).

[30] Fill Gesellschaft m.b.H. *Accubot - The robot drilling cell*. URL: https://www.fill.co.at/en/products/accubot-drilling (besucht am 17. 10. 2024).

[31] Fill Gesellschaft m.b.H. *Grind Performer - The robot grinding machine*. URL: https://www.fill.co.at/en/products/grind-performer (besucht am 17. 10. 2024).

[32] M. Mehling. „Implementation of a low cost marker based infrared optical tracking system“. Diss. Fachhochschule Stuttgart - Hochschule der Medien, 2006. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=45fa1533ffa1f20e0b823e78efdd99c0a0019684.

[33] T. J. Mitchell. „An Algorithm for the Construction of “D-Optimal” Experimental Designs“. In: *Technometrics* 42.1 (2000), S. 48–54. URL: https://doi.org/10.1080/00401706.2000.10485978.

[34] C. Moeller u. a. *Real time pose control of an industrial robotic system for machining of large scale components in aerospace industry using laser tracker system*. 2017. URL: http://hdl.handle.net/11420/2801.

[35] E. Mok. *ABB-IRB-1200 Kinematische Analyse, MATLAB RVC Tool-Analyse + Simulink-Adams Gelenk-Simulation*. URL: %5Curl%7Bhttps://blog.csdn.net/Ezekiel_Mok/article/details/122358393%7D (besucht am 23. 09. 2024).

[36] A. Müller und H. Bremer. *Vorlesungsskriptum Höhere Kinetik - Mehrkörpersysteme*. Institut für Robotik, Johannes Kepler Universität Linz, 2020.

[37] L. Naimark und E. Foxlin. „Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker“. In: *Proceedings. International Symposium on Mixed and Augmented Reality*. 2002, S. 27–36.

[38] P. Obal und P. Gierlak. „EGM Toolbox—Interface for Controlling ABB Robots in Simulink“. In: *Sensors* 21.22 (2021). ISSN: 1424-8220. URL: https://www.mdpi.com/1424-8220/21/22/7463.

[39] Edwin Olson. „AprilTag: A robust and flexible visual fiducial system“. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, S. 3400–3407.

[40] F. Romero-Ramirez, R. Muñoz-Salinas und R. Medina-Carnicer. „Speeded Up Detection of Squared Fiducial Markers“. In: *Image and Vision Computing* 76 (2018).

[41] C. Rosner. „Lokalisierung distinktiver Merkmale eines Roboter-Endeffektors für das Tracking eines 3D-Structured-Light-Sensors“. Bachelorarbeit. Software Engineering IKM Fakultät Hagenberg, FH Oberösterreich, 2022.

[42] M. Sakthivel. *Example code for EGM control of ABB robots and external axes*. URL: https://github.com/mohith-sakthivel/abb_egm_motion_control.git (besucht am 12. 09. 2024).

[43] *Springer handbook of robotics*. eng. Springer, 2008. ISBN: 354023957X. URL: http://d-nb.info/973670681/04.

[44] H. Urban. „Entwicklung eines Systems zur Artikelerkennung mit Hilfe von Künstlicher Intelligenz zur Überprüfung kommissionierter Waren“. Masterarbeit. Institut für Konstruktionswissenschaften und Produktenwicklung, Technische Universität Wien, 2021.

[45] E. Urtans und A. Nikitenko. „Active infrared markers for augmented and virtual reality“. In: *Riga Technical University, Latvia* 9 (2016), S. 10. URL: https://www.iitf.lbtu.lv/conference/proceedings2016/Papers/N197.pdf.

[46] Wikipedia. *Gauß-Newton-Verfahren*. 2024. URL: https://de.wikipedia.org/wiki/Gau%C3%9F-Newton-Verfahren (besucht am 10. 09. 2024).

[47] H. Wu u. a. „An Accurate Recognition of Infrared Retro-Reflective Markers in Surgical Navigation“. In: *Journal of Medical Systems* 43 (2019), S. 1–13. URL: https://doi.org/10.1007/s10916-019-1257-x.

Lukas PROBST, BSc

05.06.1998

luprob@aon.at

+43 (0)650 5730209

Hans Berghammer
Siedlung 44
5230 Mattighofen
Österreich

https://www.linkedin.com/in/lukas-probst-b439a12a0

ZIELSETZUNG

Meinen Berufseinstieg plane ich im Bereich Prozessentwicklung und Prozessoptimierung.

QUALIFIKATIONEN

Durch mein Studium an der JKU konnte ich mir bereits unter anderem ein fundiertes Wissen in den Bereichen Automatisierungstechnik, Robotik, technische Mechanik und Modellbildung aneignen.

ERFAHRUNG

VOLLZEIT-ANSTELLUNGEN

FILL Gesellschaft m.b.H.
(seit Oktober 2023, Forschung und Entwicklung)

TEILZEIT-ANSTELLUNGEN

JKU Institut für Robotik
(Oktober 2022 bis Oktober 2023, Mitarbeiter in der Forschung)

JKU Institut für Maschinenlehre und Antriebstechnik
(Oktober 2021 bis Jänner 2023, Tutor)

BERUFSPRAKTIKA

FILL Gesellschaft m.b.H. (Juli, August 2022)
AMAG Austria Metall AG (Juli, August 2021 und August 2018)
WACKER Chemie AG (Juli, August 2019)
APTIV (Mai, Juni 2018)
PROMOTECH Kunststoff- und Metallverarbeitungs GmbH
(August 2017, August 2016, August 2015 und August 2014)

AUSBILDUNG

JOHANNES KEPLER UNIVERSITÄT LINZ

seit März 2022
Masterstudium Mechatronik – Schwerpunkt Robotik, Regelungstechnik und Automatisierung

JOHANNES KEPLER UNIVERSITÄT LINZ

2018 - 2022
Bachelorstudium Mechatronik – Vertiefung Robotik und Automatisierung

HTBLA BRAUNAU

2012 bis 2017
Zweig Mechatronik – Material Processing

BESONDERE KENNTNISSE UND FÄHIGKEITEN

Computerkenntnisse: Microsoft Office (Word, Excel, PowerPoint), Pro Engineer / CREO, Matlab, Maple, LaTeX

Programmierkenntnisse: Java, C++, AMD64 Assembler und Python

Hobbys und ehrenamtliche Tätigkeit: Bergsport (Tourenführer Alpenverein Sektion Mattighofen), Sporttauchen, Reisen