



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Christoph Scholl

Optimierung bei der Kalibrierung von Sensor-Roboter-Systemen

*Fakultät Technik und Informatik
Department Maschinenbau und Produktion*

*Faculty of Engineering and Computer Science
Department of Mechanical Engineering and
Production Management*

Christoph Scholl

**Optimierung bei der Kalibrierung von
Sensor-Roboter-Systemen**

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Berechnung und Simulation
am Department Maschinenbau und Produktion
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
Die Fraunhofer-Einrichtung für Additive Produktionstechnologien IAPT
AM Factory
Am Schleusengraben 14
21029 Hamburg-Bergedorf

Erstprüfer/in: Prof. Dr.-Ing. habil. Frank Helmut Schäfer
Zweitprüfer/in: Dr.-Ing. Dirk Herzog

Abgabedatum: 20. Juni 2019

Zusammenfassung

Name des Studierenden

Christoph Scholl

Thema der Masterarbeit

Optimierung bei der Kalibrierung von Sensor-Roboter-Systemen

Stichworte

Industrieroboter, Kalibration, Sensor, numerische Umsetzung, MATLAB, Triangulation, Optimierung, Gauß-Newton, Newton, Trust-Region, Minimierung, Vermessungsstrategie,

Kurzzusammenfassung

In der angeführten Masterthese wird eine Optimierung der Hand-Auge-Kalibration von Robotern vorgenommen. Diese umfasste eine Analyse der bekannten Kalibrationsmethoden sowie deren Lösungsansätze. Des Weiteren wird ein physikalisches Modell definiert, welches in ein mathematisches Modell überführt wird. Die Minimierung des mathematischen Modells wird in einem Minimierungsalgorithmus umgesetzt. Abschließend werden die gewonnenen Ergebnisse gegenübergestellt und ein Ausblick auf weiterführende Entwicklungen und Forschungsfragen wird gegeben.

Name of Student

Christoph Scholl

Master Thesis title

Optimization of the calibration of sensor-robot-systems

Keywords

Industrial robots, calibration, sensor, numerical implementation, MATLAB, triangulation, optimization, Gauss-Newton, Newton, trust-region, minimization, measurement strategy,

Abstract

This master thesis shows an optimization of the hand-eye-calibration of a robot-sensor-systems. This included an analysis of the known calibration methods and their solutions. On the basis of the analysis a physical model is defined. The defined model ist transfomed into an equivalent mathematical model. The minimization of the mathematical model is implemented in a minimization algorithm. At the end of the thesis, a comparison of the results obtained and an outlook on further developments and research will be given.

Aufgabenstellung

Die Bedeutung der Vollautomatisierung in der Industrie gewinnt immer mehr an Bedeutung. Besonders in der Serienfertigung ist dies schon zu großen Teilen umgesetzt. Durch die Umsetzung der Teil- und Vollautomatisierung in der Kleinserienfertigung wird ein enormes Potential zur Steigerung der Produktivität erreicht. Die Umsetzung der Automatisierung ist eng mit der Verwendung von Robotern verknüpft. Die Umrüstzeiten der Roboter zur Anpassung an die unterschiedlichen Kleinserien gehen mit einem hohen Aufwand an Positionierung und Programmierung der einzelnen Positionen des Werkzeuges einher. Diese hohen Umrüstzeiten machen eine Automatisierung in der Kleinserien- und Einzelteilerfertigung unwirtschaftlich.

Mittels einer Hand-Auge-Kalibration kann eine bessere Allgemeingenaugigkeit erzielt werden. Aufgrund der gesteigerten Genauigkeit können wesentliche Anwendungen in der Robotertechnik und Handhabung vereinfacht werden. Die Probleme zwischen der Offline-Programmierung und der Abweichung im realen Verfahren werden minimiert. Durch die Verwendung einer Hand-Auge-Kalibration zwischen einem Sensor und einem Endeffektor ergeben sich weitere Möglichkeiten, die Automatisierung zu verbessern und wirtschaftlicher umzusetzen. Eine Erkennung von Bauteilgeometrien und deren Lage im Raum ist mit dieser Kalibrierung umsetzbar. Aufgrund der möglichen Lagenerkennung und der Information, wo sich der Roboter im Raum befindet, können diese Informationen zur Regelung des Roboters herangezogen werden.

Um eine Umsetzung der angedeuteten Vorteile zu nutzen, wird eine Hand-Auge-Kalibration numerisch durchgeführt. In dem Modell des Kalibrieralgorithmus wird beispielweise eine Messkugel als Kalibrierkörper verwendet. Die numerische Umsetzung wird an einer Roboter-Sensor-Kombination validiert, um eine Aussage über die Genauigkeit zu erlangen.

| | | |
|------------|---|------------|
| I | Inhaltsverzeichnis | |
| II | Abbildungsverzeichnis | IV |
| III | Tabellenverzeichnis | V |
| IV | Abkürzungsverzeichnis | VI |
| V | Nomenklatur | VII |
| 1 | Einleitung | 1 |
| 2 | Stand der Technik | 2 |
| 2.1 | Sechs-Achs-Industrieroboter für industrielle Anwendungen | 2 |
| 2.1.1 | Robotergenauigkeit | 3 |
| 2.1.2 | Orientierung und Bewegung des Roboters im Raum | 5 |
| 2.2 | Messprinzip der Triangulation | 8 |
| 2.3 | Grundlagen der Optimierung | 10 |
| 2.4 | Optimalitätsbedingungen | 11 |
| 2.4.1 | Minimum | 11 |
| 2.4.2 | Konvexität | 12 |
| 2.4.3 | Erste und zweite Ordnung der Optimalitätsbedingung | 15 |
| 2.5 | Minimierungsalgorithmen | 17 |
| 2.5.1 | Algorithmische Struktur der numerischen Verfahren | 17 |
| 2.5.2 | Newton-Verfahren | 18 |
| 2.5.3 | Gauß-Newton-Verfahren | 19 |
| 2.5.4 | Trust-Region-Verfahren | 20 |
| 2.6 | Hand-Auge-Kalibration mittels 3D Sensors | 21 |
| 2.6.1 | Hand-Auge-Kalibration nach Jianfeng Li, Ming Chen | 21 |
| 2.6.2 | Hand-Auge-Kalibration nach Min Yung Kim, Hyung Suck Cho | 23 |
| 3 | Methodische Betrachtung | 25 |
| 4 | Physikalisches und mathematisches Modell | 27 |
| 4.1 | Analyse bekannter physikalischer Systeme und mathematischer Modelle | 27 |
| 4.2 | Physikalisches System | 30 |
| 4.3 | Kalibration im physikalischen System | 31 |
| 4.4 | Mathematisches Modell | 32 |
| 4.4.1 | Entwickeltes mathematisches Modell | 32 |
| 4.4.2 | Lösung des mathematischen Modells | 34 |
| 5 | Analyse aktueller Lösungsalgorithmen | 37 |
| 5.1 | Potentialanalyse der Lösungsalgorithmen | 37 |
| 5.2 | Vergleich der Lösungsalgorithmen | 39 |

| | | |
|-----------|---|-----------|
| 6 | Umsetzung und Implementierung | 42 |
| 6.1 | Integration und Lösung in MATLAB | 42 |
| 6.1.1 | Import und Datenstruktur..... | 44 |
| 6.1.2 | Umrechnung von Posen in homogene Transformationsmatrizen | 44 |
| 6.1.3 | Berechnung des Residuums..... | 45 |
| 6.1.4 | Berechnung der Jacobi-Matrix..... | 46 |
| 6.1.5 | Hand-Auge-Kalibrationsalgorithmus mittels Gauß-Newton-Verfahren | 47 |
| 6.1.6 | Hand-Auge-Kalibrationsalgorithmus mittels Trust-Region-Verfahren..... | 48 |
| 6.2 | Theoretische Betrachtung einer definierten Potenzfunktion..... | 50 |
| 6.3 | Strategie zur Ermittlung von Simulations-Inputargumenten..... | 53 |
| 6.4 | Experimentelle Datenaufnahme..... | 55 |
| 6.4.1 | Sphärischer Kalibrierkörper..... | 55 |
| 6.4.2 | Vermessungsstrategie..... | 58 |
| 6.4.3 | Aufnahme der benötigten Inputargumente | 60 |
| 7 | Ergebnisse und Diskussion | 67 |
| 7.1 | Gegenüberstellung der Minimierungsalgorithmen anhand theoretischer Modelle .. | 67 |
| 7.1.1 | Vergleich der Minimierungsalgorithmen anhand einer Potenzfunktion | 67 |
| 7.1.2 | Vergleich der Minimierungsalgorithmen auf Grundlage des Simulationsmodells..... | 70 |
| 7.2 | Minimierung des realen mathematischen Modells | 73 |
| 7.3 | Zusammenfassung und Fazit der gewonnenen Ergebnisse | 74 |
| 8 | Zusammenfassung und Ausblick | 75 |
| VI | Anhang | 79 |
| A | Besetzung der D-Matrix, des U- und F-Vektors nach [KCK2001]..... | 79 |
| B | Anhang aus Kapitel 4 | 80 |
| B1 | Sensor scanCONTROL 2910-100 Technische Daten | 80 |
| B2 | Sensor scanCONTROL 2910-100 Datenblatt | 81 |
| C | Anhang aus Kapitel 6 | 82 |
| C1 | Entwickelte Funktion zum Importieren der Messdaten..... | 82 |
| C2 | Skript zur Umrechnung von Posen in h.Transformationsmatrizen..... | 83 |
| C3 | Entwickelte Funktion zur Berechnung der Differenz/Epsilon | 84 |
| C4 | Entwickelte Funktion zur Ermittlung der Jacobimatrix..... | 85 |
| C5 | Entwickeltes Skript des Gauß-Newton-Minimierungsalgorithmus | 88 |
| C6 | Ansteuerung der MATLAB-Toolbox des realen mathematischen Modells | 89 |
| C7 | Hessematrix der Potenzfunktion, $\nabla^2 f(x)$ | 91 |
| C8 | Entwickelter Newton-Algorithmus zur Minimierung der Potenzfunktion..... | 92 |
| C9 | Entwickelte Funktion / Implementierung der Potenzfunktion..... | 93 |
| C10 | Simulationsposen und Inputargumente..... | 94 |
| C11 | Zertifikat der Kalibrierkugel TW DK60.0 GE_M8 | 96 |

| | | |
|-----|--|-----|
| C12 | Abnahme-Protokoll Wenzel, 3D-Koordinatenmessmaschine X0 107 3D..... | 97 |
| C13 | Messprotokoll 3D-Koordinatenmessmaschine X0 107 3D, Kalibrierkörper..... | 98 |
| C14 | Posen der gemessenen Inputparameter TFA | 99 |
| C15 | Funktion zur Bestimmung der Mittelpunktkoordinaten einer Kugel | 101 |
| C16 | Entwickeltes Skript zur Ermittlung des Kugelmittelpunktes und der graphischen Darstellung des Kreises | 102 |
| C17 | Messungen des Inputparameters des gemessenen Vektors XS | 104 |
| D | Anhang CD..... | 106 |

II Abbildungsverzeichnis

| | |
|---|----|
| Abbildung 2-1: Achsen- / Roboterbezeichnung..... | 2 |
| Abbildung 2-2: Wiederhol-/Absolutgenauigkeit | 3 |
| Abbildung 2-3: Absolut- und Wiederholgenauigkeit | 4 |
| Abbildung 2-4: Schematische Darstellung einer Pose zwischen zwei Koordinatensystemen..... | 5 |
| Abbildung 2-5: Körperfestes Koordinatensystem, Roll-Pitch-Yaw Konvention | 7 |
| Abbildung 2-6: Grundlegendes Messprinzip der triangulatorischen Abstandsmessung | 8 |
| Abbildung 2-7: Schematischer Aufbau eines Triangulationssensors | 9 |
| Abbildung 2-8: Verschiedene Minima einer beliebigen Funktion $f(x)$ | 12 |
| Abbildung 2-9: Beispiele von konvexen Mengen..... | 13 |
| Abbildung 2-10: Beispiel von konkaven und konvexen Funktionen..... | 14 |
| Abbildung 2-11: Rechnergestützte Suche nach einem Minimum | 17 |
| Abbildung 2-12: Schematische Abbildung des verwendeten physikalischen Systems | 21 |
| Abbildung 2-13: Relative Roboterbewegung für die Hand-Auge-Kalibration nach | 23 |
| Abbildung 4-1: Reales System bestehend aus Industrieroboter, 2-A-Positionierer | 30 |
| Abbildung 4-2: Detaillierte Betrachtung der Sensoranbringung am Endeffektor | 30 |
| Abbildung 4-3: Schematische Darstellung der Kalibrationen am Robotersystem..... | 31 |
| Abbildung 4-4: Angepasstes physikalisches System, Grundlage des mathemat. Modells..... | 32 |
| Abbildung 4-5: Schematische Darstellung des entwickelten mathematischen Modells..... | 33 |
| Abbildung 6-1: Dreidimensionale Abbildung der Potenzfunktion mit allen Extremstellen | 50 |
| Abbildung 6-2: Höhenlinien der Potenzfunktion mit allen Extremstellen | 52 |
| Abbildung 6-3: Ideales System mit einer Drehung um 90° um die y-Achse (A) | 53 |
| Abbildung 6-4: Schematische Darstellung des Kalibrierkörpers | 55 |
| Abbildung 6-5: Befestigung des Kalibrierkörpers auf dem 2-Achs-Positionierer..... | 56 |
| Abbildung 6-6: Defi. Messwinkel und Koordinatensysteme für die Vermessungsstrategie | 59 |
| Abbildung 6-7: Schema. Darstellung einer Messung, die um 45° um die x-Achse gedreht ist..... | 59 |
| Abbildung 6-8: Schema. Darstellung der Positionsermittlung des Kalibrierkörpers..... | 60 |
| Abbildung 6-9: Koordinatensystem Flansch und Sensor, Identifizierung des Startparameter..... | 63 |
| Abbildung 6-10: Schematische Darstellung der Bestimmung des Vektors x_s | 64 |

III Tabellenverzeichnis

| | |
|---|----|
| Tabelle 2-1: Roboter Bewegungsbereich und Geschwindigkeit HR 60 HA | 3 |
| Tabelle 2-2: Problemklassen der Optimierung nach [Gri2013, Mes2015, NoWr2006a] | 11 |
| Tabelle 3-1: Strategie zur Bestimmung geeigneter Optimierungsalgorithmen | 26 |
| Tabelle 4-1: Analysen Bewertung der einzelnen Kalibrieransätze..... | 28 |
| Tabelle 4-2: Optimierungsstrategie des mathematischen Modells..... | 36 |
| Tabelle 5-1: Einteilung Minimierungsalgorithmen, Lösungsalgorithmen der Potentialanalyse | 38 |
| Tabelle 5-2: Bewertung der Lösungsalgorithmen der Potentialanalyse | 40 |
| Tabelle 6-1: Schematische Struktur des entwickelten Optimierungsskriptes | 42 |
| Tabelle 6-2: Posen/Vektoren des idealen Systems für die Simulations-Inputargumente..... | 54 |
| Tabelle 6-3: Vergleich der theoretischen und realen Kalibrierkörperabmessungen..... | 57 |
| Tabelle 6-4: Werte des Inputparameters TPA ; Pose und homogene Transformationsmatrix... | 61 |
| Tabelle 6-5: Werte des Inputparameters xP' ; Vektor..... | 61 |
| Tabelle 6-6: Inputargumente der homogenen Transformationsmatrix TAF um 45° gedreht.... | 62 |
| Tabelle 6-7: Start-Pose $p2SF$ und homogene Transformationsmatrix TFS | 63 |
| Tabelle 6-8: Inputargumente des Vektors xs um jeweils 45° gedreht | 65 |
| Tabelle 7-1: Ergebnisse der Minimierungsalgorithmen an einer definierten Potenzfunktion .. | 69 |
| Tabelle 7-2: Startposen $p2i$ des Simulationsmodells | 70 |
| Tabelle 7-3: Ergebnisse der finalen Minimierungsalgorithmen am Simulationsmodell | 72 |
| Tabelle 7-4: Ergebnisse der Minimierung des realen mathematischen Modells..... | 73 |
| Tabelle 7-5: Ergebnis und Bewertung der minimierten Pose $p2$ | 74 |

IV Abkürzungsverzeichnis

| Abkürzung | Bedeutung |
|--------------|---|
| CP | Bahnsteuerung |
| FDM | Finite Differenzen Methode |
| G-Verfahren | Gradienten-Verfahren |
| KQ-Verfahren | Verfahren der kleinsten Quadrate |
| KS | Koordinatensystem |
| MATLAB | Programmiersprache der Firma MatheWorks |
| mse | mean squared error / mittlere quadratische Abweichung |
| MP | Multipunktbewegung |
| n-Verfahren | newton-Verfahren |
| NN-Verfahren | Neuronale Netze |
| PTP | Punkt zu Punkt Bewegung |
| RPY | Roll-Pitch-Yaw / Roll-Nick-Gier Konvention |
| TCP | Tool Center Point |
| TR-Verfahren | Trust-Region-Verfahren |

V Nomenklatur

| Lateinische Symbole | Einheit | Bedeutung |
|---------------------|---------|---|
| A1-6 | [-] | Roboterachswinkel |
| A | [°] | Posenwinkel um die z-Achse |
| B | [°] | Posenwinkel um die z-Achse |
| C | [°] | Posenwinkel um die z-Achse |
| c | [-] | Konstante |
| d' | [mm] | Objektabstand |
| D | [-] | Schrittweite Newton Verfahren |
| E_L | [mm] | Brennweite |
| F | [-] | Zielfunktionen |
| $f(x)$ | [-] | Zielfunktion |
| F' | [mm] | Abstand Lichtquelle zur Optik |
| G | [-] | Gradient |
| $g(x)$ | [-] | Gleichungsnebenbedingung |
| H | [-] | Hessematrix |
| $h(x)$ | [-] | Ungleichungsnebenbedingung |
| h | [-] | Schrittweite Trust-Region Verfahren |
| h_c | [-] | Chauchy Schrittweite |
| H_T | [-] | Homogene Transformationsmatrix |
| I | [-] | Iteration |
| K | [-] | Iteration |
| $L_{1,2}$ | [mm] | Variable Länge |
| P | [-] | Roboterposition "1" |
| p | [-] | Pose |
| Q | [-] | Roboterposition "2" |
| R | [-] | Rotationsmatrix |
| T | [-] | Translationsmatrix |
| X_S | [mm] | Messstrecke Sensor |
| X_W | [mm] | Messstrecke Welt-KS zum Kalibrierkörper |
| x | [-] | x-Koordinatenrichtung |
| x^0 | [-] | Starparametersatz |
| x^* | [-] | Extremwert der Zielfunktion |
| y | [-] | y-Koordinatenrichtung |
| z | [-] | z-Koordinatenrichtung |

Formelverzeichnis

| Griechische Symbole | Einheit | Bedeutung |
|---------------------|---------|--------------------------|
| α | [°] | Winkel um die z-Achse |
| β | [°] | Winkel um die y-Achse |
| γ | [°] | Winkel um die x-Achse |
| ΔT | [-] | Relative Roboterbewegung |
| χ | [-] | Grundmenge |

1 Einleitung

Die industrielle Serienfertigung von maschinenbaulichen Komponenten birgt besondere Herausforderungen für Ingenieure, Werker und Automatisierungstechniker. Um die wirtschaftliche Fertigung von komplexen Bauteilen in optimaler Losgröße umsetzen zu können, ist ein hoher Automatisierungsgrad des gesamten Fertigungsprozesses erforderlich. Diese Automatisierung wird häufig durch den Einsatz von Portalen, 6-Achs-Robotern und Delta-Robotern realisiert, welche aufgrund ihrer kinematischen Flexibilität und der Umrüstbarkeit von Bearbeitungswerkzeugen eine wichtige Komponente in der Umsetzung einer wirtschaftlichen Prozesskette darstellen. Oft ist es sinnvoll, ein Robotersystem durch den Einsatz von Sensoren zu erweitern. Ein Beispiel hierzu ist in der additiven Fertigung zu finden, in welcher roboterbasierte Auftragsschweißprozesse durch optoelektronische Distanzsensoren ergänzt werden. Diese ermöglichen neben der Bauteil- und Positionserkennung zudem die Implementierung eines geschlossenen Regelkreises für die Lagenoffsetkorrektur und die Nahtverfolgung. Die Übergabe korrekter Positionsdaten des Sensors an die Robotersteuerung ist eine kritische Systemgrenze, welche besondere Aufmerksamkeit verlangt. Während Roboter und Sensor eine verlässliche interne Kalibrierung besitzen, muss die Kalibration der beiden Systemkomponenten zueinander ergänzt werden, welche bei jedem Werkzeugwechsel erneut durchgeführt werden muss. Diese Hand-Auge-Kalibration kann mittels numerischer Verfahren umgesetzt werden, wobei nichtlineare Optimierungsalgorithmen eingesetzt werden.

In der vorliegenden Arbeit wird eine solche numerische Umsetzung der Hand-Auge-Kalibration eines Systems bestehend aus 6-Achs-Industrieroboter, Laser-Linien Triangulationssensor und einem Bearbeitungskopf für das Laser-Pulver-Auftragsschweißverfahren betrachtet. Dazu wird das physikalische System in ein mathematisches Modell überführt, in welchem ein Minimierungsproblem definiert werden kann. Die zur Lösung dieses Problems notwendigen Optimierungsalgorithmen werden analysiert und ein ausgewähltes Verfahren wird in der Software MATLAB programmiert. Die Hand-Auge-Kalibration, bestehend aus der Transformationsmatrix zwischen den internen Roboter- und Sensorkoordinatensystemen, wird anschließend anhand von Messdaten validiert.

Im Anschluss an die Einleitung dient das Kapitel 2 der Erläuterung des Stands der Technik. Darin werden die Funktionsweise eines 6-Achs-Industrieroboters und eines Triangulationssensors sowie die mathematischen Grundlagen der numerischen Optimierung erläutert. In Kapitel 3 wird die methodische Vorgehensweise für die Bearbeitung der Aufgabenstellung dargestellt. Die Analyse des physikalischen Systems und die Umsetzung in ein äquivalentes mathematisches Modell wird im Kapitel 4 erläutert. Auf die Analyse, Implementierung und Umsetzung der Minimierungsalgorithmen wird im Kapitel 5 und 6 eingegangen. In Kapitel 7 werden die Ergebnisse der Kalibrierung validiert und bewertet. Die Arbeit schließt mit dem Kapitel 8 ab, in welchem ein Ausblick auf die Weiterentwicklung des Optimierungsalgorithmus gegeben wird.

2 Stand der Technik

In diesem Abschnitt wird der aktuelle Stand der Technik dargestellt. Zunächst wird die Funktionsweise von Sechs-Achs-Industrierobotern und die Grenze der möglichen Genauigkeit betrachtet. Anschließend werden die Grundlagen der Abstandsmessung mittels der Triangulation erläutert. Im Abschluss des Kapitels werden die benötigten mathematischen Methoden der Optimierung eingeführt. Ausgehend von dieser Basis erfolgt die Erstellung des mathematischen Modells und die Auswahl der Algorithmen.

2.1 Sechs-Achs-Industrieroboter für industrielle Anwendungen

Unter Industrierobotern werden Roboter mit mindestens drei frei programmierbaren Achsen verstanden. Die Hauptaufgabe eines Industrieroboters ist es, ein Werkzeug (Wirkorgan) effizient in einem definierten Raum zu führen. Die Montage eines Wirkorgans, wie z.B. Greifer, Roboterwerkzeuge, Montagevorrichtungen und Sensoren, wird am Flansch des Roboters vorgenommen. Dabei wird der Teil des Roboters als Endeffektor bezeichnet, welcher im direkten Kontakt mit der Umgebung steht. Der Endeffektor befindet sich am Ende der kinematischen Kette des Roboters und ist meist mit dem Wirkorgan gleichzusetzen [HeMa2016, Web2017]. Das zu führende Werkzeug besitzt ein frei definierbares Koordinatensystem, welches als Tool-Center-Point (TCP) bezeichnet wird. [HeMa2016, Web2017] Für eine schweißtechnische Anwendung werden Sechs-Achs-Knickarm-Roboter verwendet. Die ersten drei Achsen werden Hauptachsen (A1-A3) genannt und sind im Wesentlichen für die Positionierung des TCP zuständig. Die nachfolgenden drei Achsen (A4-A6) werden als Handachsen bezeichnet. Diese Achsen sind im Wesentlichen für die Positionierung des Endeffektors im Raum zuständig (Abbildung 2-1). Die vollständige Beschreibung der Position des TCP im kartesischen Koordinatensystem besteht aus drei geometrischen ($[x, y, z]^T$) und rotatorischen Freiheitsgraden ($[A, B, C]^T$). Durch die zusätzliche Angabe der sechs Roboterachswinkel ($[\phi_1, \dots, \phi_6]^T$) kann eine Position des TCP vollumfänglich beschrieben werden. Eine so beschriebene Position wird als Pose bezeichnet [HeMa2016, Web2017]. Eine schematische Darstellung des Sechs-Achs-Vertikal Knickarm-Roboters wird in der Abbildung 2-1 dargestellt.

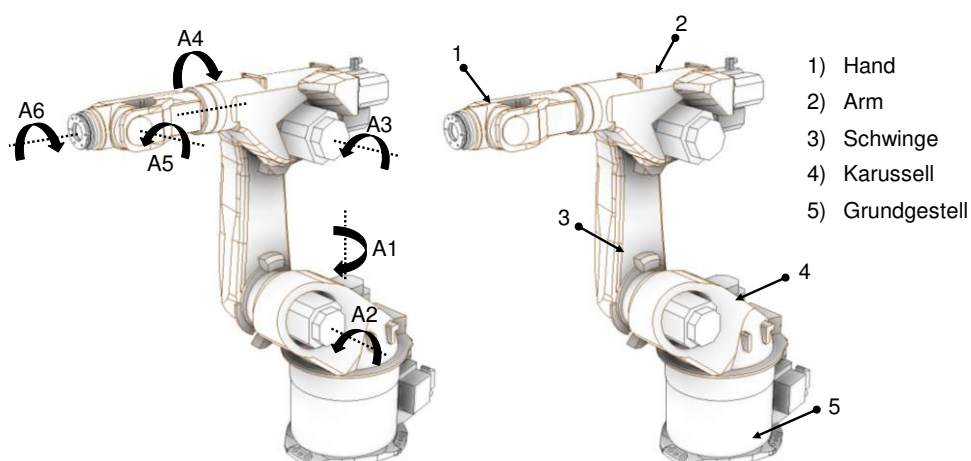


Abbildung 2-1: Achsen- / Roboterbezeichnung [KUR2005, Mai2016]

Der Roboter wird in 5 Grundbauteile unterteilt. Mit dem Grundgestell wird eine Fixierung im Boden und somit die Ableitung der Kräfte ermöglicht. Die Drehung der Schwinge um die Achse (A1) wird durch das Karussell mit einem maximalen Verstellwinkel von $\pm 185^\circ$ bewirkt. Der Arm und die Schwinge werden durch die Achse (A3) verbunden. Am Ende des Armes befindet sich die sechste Achse des Roboters. An dieser Achse wird das Werkzeug montiert. In der nachfolgenden Tabelle 2-1 werden die maximalen Verstellwinkel und Verstellgeschwindigkeiten der einzelnen Achsen dargestellt. [KUR2005]

Tabelle 2-1: Roboter Bewegungsbereich und Geschwindigkeit HR 60 HA [KUR2005]

| Achse | Bewegungsbereich [°] | Geschwindigkeit [°/s] |
|-------|----------------------|-----------------------|
| 1 | ± 185 | 128 |
| 2 | + 35 bis -135 | 102 |
| 3 | + 158 bis -120 | 128 |
| 4 | ± 350 | 260 |
| 5 | ± 119 | 245 |
| 6 | ± 350 | 322 |

2.1.1 Robotergenauigkeit

Die erreichten Genauigkeiten eines Roboters hängen von der Traglast und der Vorschubgeschwindigkeit ab. Die allgemeinen Angaben zur Nennlast beziehen sich auf die Anschlussfläche des Endeffektors. Durch das Verwenden von Werkzeugen, Greifern oder Sensoren, verschiebt sich der Schwerpunkt der zu führenden Last vom Flanschschwerpunkt weg. Dies hat zur Folge, dass sich bei einer gleichbleibenden Genauigkeit die maximal zu führenden Lasten verringern. Sollten diese Vorschriften umgangen werden, verringern sich die Genauigkeiten des Roboters. [HeMa2016, Bey2004]

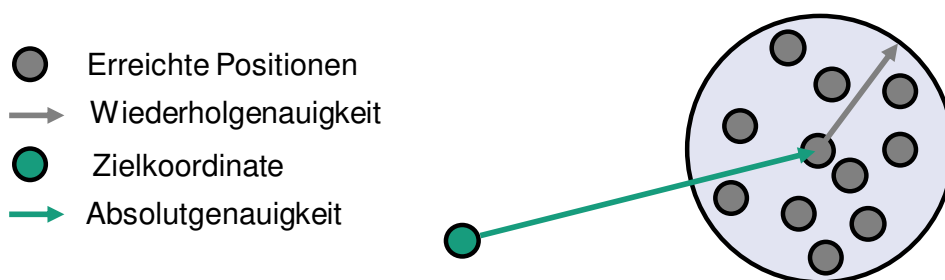


Abbildung 2-2: Wiederhol-/Absolutgenauigkeit [Bey2004]

In der Genauigkeitsbetrachtung wird in Absolutgenauigkeit und Wiederholgenauigkeit unterschieden (Abbildung 2-2). Die Absolutgenauigkeit gibt die Differenz zwischen der erwarteten Sollposition und dem Mittelwert der angefahrenen Istposition an. Der Mittelwert der Istposition ergibt sich aus mehrmaligem Anfahren der gleichen Sollpose aus verschiedenen Richtungen. Durch den Bezug dieses Mittelwertes auf das Basiskoordinatensystem des Roboters wird ein Wert für die Absolutgenauigkeit ermittelt. [HeMa2016, Sch1993]

Die Wiederholgenauigkeit wird durch das mehrfache Anfahren einer Sollpose aus einer definierten Richtung ermittelt. Aufgrund der gleichen Verfahrenswege ergeben sich Abweichungen in den Istposen. Diese Differenz wird als Wiederholgenauigkeit definiert. Eine solche Genauigkeitsprüfung kann ohne bekannte Lage des Basiskoordinatensystems des Roboters durchgeführt werden, da nur ein Abgleich der Istposen benötigt wird. [Bey2004, Sch1993]

In der nachfolgenden Abbildung 2-2 wird eine schematische Darstellung der Trefferbilder von angefahrenen Posen aufgezeigt. Des Weiteren lässt sich erkennen, dass sich trotz einer schlechten Absolutgenauigkeit dennoch eine gute Wiederholgenauigkeit erreichen lässt. [HeMa2016, Sch1993]

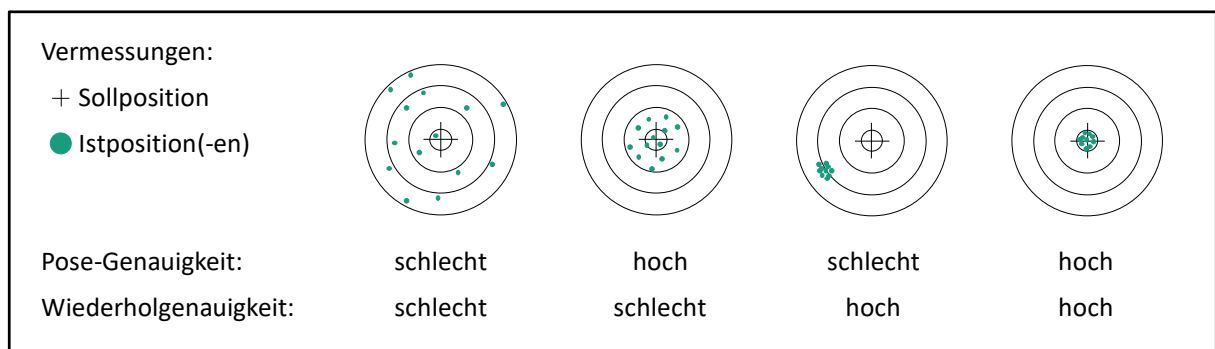


Abbildung 2-3: Absolut- und Wiederholgenauigkeit [Bey2004, HeMa2016]

Aufgrund der Konstruktion des Roboters und der Umgebung, in welcher der Roboter aufgestellt ist, ergeben sich Störeinflüsse, die sich negativ auf die Genauigkeit auswirken. In der nachfolgenden Aufzählung wird ein Ausschnitt aus möglichen Störgrößen gezeigt. Sie resultieren aus der charakteristischen Konstruktion, den verwendeten Materialien sowie aus den Umwelteinflüssen. [HeMa2016]

- Gelenkspiel in den Lagen und den Führungen
- Elastische Verformung in der statischen und dynamischen Belastung
- Unterschiedliche Reibung bei verschiedenen Verfahrgeschwindigkeiten
- Spiel in den Übertragungsgetrieben (z.B. Zahnspiel bei Stirnradgetrieben)
- Temperaturschwankungen in Strukturelementen
- Begrenztes Auflösungsvermögen der inkrementellen Wegmesssysteme

2.1.2 Orientierung und Bewegung des Roboters im Raum

Die Bewegung eines Roboters im Raum kann durch drei verschiedene Bewegungsarten durchgeführt werden. Diese lassen sich in die Punkt-zu-Punkt (PTP-Bewegungen), Multipunkt (MP-Bewegung) und Bahnsteuerung (CP-Bewegung) unterteilen. [HeMa2016, Web2017]

Punkt-zu-Punkt Steuerung

In der Punkt-zu-Punkt Steuerung werden nur die Bahnstützpunkte angefahren. Es wird keine Information über den abzufahrenden Weg, einzelner Gelenke oder des TCP's, zwischen den Punkten hinterlegt. Der Verfahrweg des TCP zwischen den Punkten wird durch die Robotersteuerung automatisch generiert und ist dem Anwender unbekannt. [HeMa2016, Web2017]

Multipunkt-Steuerung

In der Multipunkt-Steuerung werden die Bewegungsbahnen nicht durch eine typische Bahnbeschreibung oder vordefinierten Punkten beschrieben. In der Methode werden die Bahnpunkte in definierten Zeitintervallen im manuellen Verfahrprozess abgespeichert und bilden am Ende der Bewegung die definierten Bahnbewegungen ab. [HeMa2016, Web2017]

Bahnsteuerung

Die Methode der Bahnsteuerung wird für eine genaue Führung des TCP's im Raum benötigt. Durch die Vorgabe der Bahnstützpunkte (PTP) wird die Bahn definiert. Aufgrund einer weiteren Information über die Bewegungsart zwischen den Punkten wird eine definierte Bewegung des TCP's im Raum ermöglicht. Bei den Informationen handelt es sich um die Bewegungsart, Beschleunigungen und Geschwindigkeiten zwischen den Bahnstützpunkten. [HeMa2016, Web2017]

Zur vollständigen Bestimmung der Position des Endeffektors / TCP im dreidimensionalen Raum wird die Lage und Orientierung des körperfesten Koordinatensystems (Körper-KS, P') zu einem weiteren Welt- / Körperkoordinatensystem (Welt-KS, B) benötigt (Abbildung 2-4). Durch diese Beschreibung wird die Führung des Endeffektors im Raum ermöglicht.

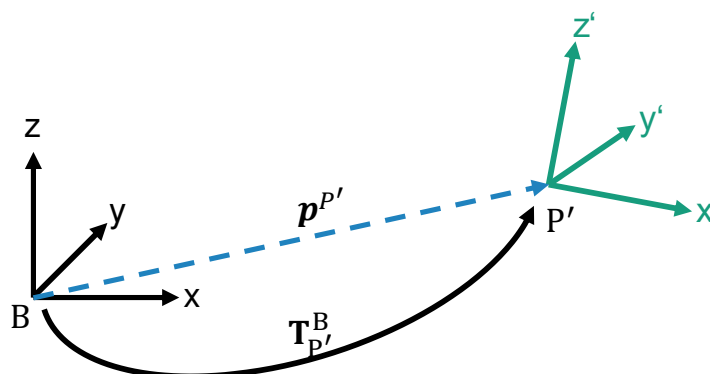


Abbildung 2-4: Schematische Darstellung einer Pose zwischen zwei Koordinatensystemen

Die Pose beschreibt die räumliche Position eines Punktes im Raum, welche sich aus der Kombination von Orientierung und Lage im dreidimensionalen Raum zusammensetzt [DIN2012]. Die Beschreibung eines Punktes bezüglich eines Welt-KS ist allein durch die Abstände in den Koordinatenachsen $(x, y, z)^T$ nicht möglich (Formel (2-1)). Um eine vollständige Beschreibung des Punktes zu erhalten, werden die Winkelversätze zwischen dem Welt-KS und dem Punkt-KS benötigt. Daher besteht eine vollständig beschriebene Pose aus sechs Einträgen und wird als Spaltenvektor abgebildet. Die ersten drei Einträge beziehen sich auf die translatorische Verschiebung des Punkt-KS im Welt-KS $(x, y, z)^T$. Die letzten drei Einträge der Pose besitzen die Werte aus den jeweiligen Winkelversätzen zu dem Welt-Koordinatensystem (Formel (2-2)) (Abbildung 2-4). [HeMa2016, Web2017]

$$a = (x_p \quad y_p \quad z_p)^T \quad (2-1)$$

$$p^{P'} = (x_p \quad y_p \quad z_p \quad A'_{p'} \quad B'_{p'} \quad C'_{p'})^T \quad (2-2)$$

Aufgrund der Darstellungsmöglichkeit der Pose als homogene Koordinaten ist eine rechnerische Bearbeitung möglich. Dabei lassen sich die Translation und Rotation des Koordinatensystems durch eine Multiplikation in homogenen Koordinaten darstellen (Formel (2-3) bis (2-8)). Da die Multiplikation von Drehungen nicht kommutativ ist, muss die Reihenfolge der Multiplikationen berücksichtigt werden. Zur Berücksichtigung der Reihenfolge sind mehrere Vereinbarungen möglich. Im Nachfolgenden wird die Roll-, Nick- und Gierwinkel (Roll-Pitch-Yaw; RPY) Konvention verwendet (Formel (2-6)) (Abbildung 2-5). Aufgrund der Matrixbeschreibung können alle Transformationen im dreidimensionalen Raum durch eine 4x4 Matrix dargestellt werden. [Bey2004, HLN2012, HeMa2016]

$$H_T = T * R \quad (2-3)$$

$$T = T_z(t_z) * T_y(t_y) * T_x(t_x) = T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2-4)$$

$$T_x(t_x) = \begin{bmatrix} E & (t_x \ 0 \ 0)^T \\ 0 & 1 \end{bmatrix}; \quad T_y(t_y) = \begin{bmatrix} E & (0 \ t_y \ 0)^T \\ 0 & 1 \end{bmatrix}; \quad T_z(t_z) = \begin{bmatrix} E & (0 \ 0 \ t_z)^T \\ 0 & 1 \end{bmatrix} \quad (2-5)$$

$$R = R_{RPY} = T_{Rz}(C) * T_{Ry}(B) * T_{Rx}(A) = \begin{bmatrix} R_{RPY} & 0 \\ 0 & 1 \end{bmatrix} \quad (2-6)$$

$$T_{Rx}(A) = \begin{bmatrix} R_x(A) & 0 \\ 0 & 1 \end{bmatrix}; \quad T_{Ry}(B) = \begin{bmatrix} R_y(B) & 0 \\ 0 & 1 \end{bmatrix}; \quad T_{Rz}(C) = \begin{bmatrix} R_z(C) & 0 \\ 0 & 1 \end{bmatrix} \quad (2-7)$$

$$R_x(A) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos A & -\sin A \\ 0 & \sin A & \cos A \end{bmatrix}; \quad R_y(B) = \begin{bmatrix} \cos B & 0 & \sin B \\ 0 & 1 & 0 \\ -\sin B & 0 & \cos B \end{bmatrix}; \quad (2-8)$$

$$R_z(C) = \begin{bmatrix} \cos C & -\sin C & 0 \\ \sin C & \cos C & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Die allgemeine Koordinatentransformationsmatrix H_T (Formel (2-3)) setzt sich aus den jeweiligen Rotationsmatrizen (Formel (2-8)) und der Translation der zugehörigen Achsen zusammen. Dabei setzt sich die Translationsmatrix T durch die elementaren Translationen in den Koordinatenrichtungen zusammen. Die allgemeine homogene Translationsmatrix T (Formel (2-4)) setzt sich aus der Multiplikation der einzelnen homogenen Translationsmatrizen der einzelnen Koordinatenrichtungen zusammen. Die Zusammensetzung der Drehmatrix R (Formel (2-6)) wird durch die Multiplikation der elementaren homogenen Rotationstransformationen (Formel (2-6)) um die jeweiligen Achsen umgesetzt (Formel (2-7)) [Bey2004, HeMa2016]

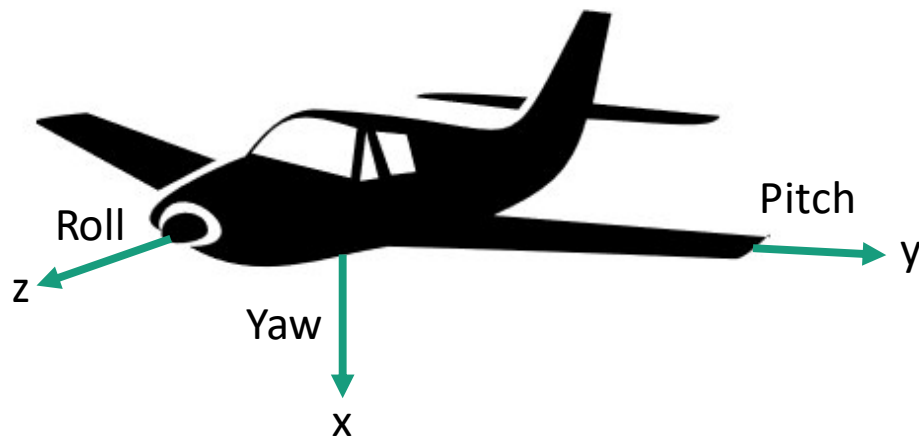


Abbildung 2-5: Körperfestes Koordinatensystem, Roll-Pitch-Yaw Konvention [HeMa2016]

2.2 Messprinzip der Triangulation

Das Messprinzip der Triangulation basiert auf dem Verfahren der geometrischen Triangulation zur Ermittlung von Abständen. Durch das Verwenden der Trigonometrie können über Winkelfunktionen Seitenlängen und Winkel eines Dreiecks ermittelt werden. Bei der Triangulation wird dieser Zusammenhang verwendet, um die Unbekannten eines Dreiecks zu ermitteln.

Das Messprinzip triangulatorischer Wegsensoren basiert auf dem oben genannten geometrischen Prinzip. Um den Abstand eines Objektes zu bestimmen wird durch eine Lichtquelle Strahlung emittiert. Diese Strahlung wird von der Oberfläche des zu messenden Objektes reflektiert. Die reflektierte Strahlung kann durch den Detektor aufgenommen werden. Aufgrund der bekannten geometrischen Beziehung von Detektor, Lichtquelle und Objekt kann der Abstand H berechnet werden (Abbildung 2-6 und Formel (2-12)). [Wol2016, Ga-Ta2011]

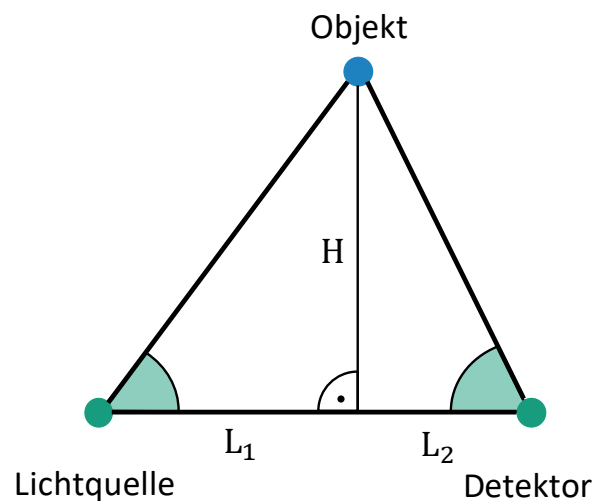


Abbildung 2-6: Grundlegendes Messprinzip der triangulatorischen Abstandsmessung [Wol2016]

$$\tan \alpha = \frac{H}{L_1}; \quad (2-9)$$

$$\tan \beta = \frac{H}{L_2}; \quad (2-10)$$

$$L = L_1 + L_2 = \frac{H}{\tan \alpha} + \frac{H}{\tan \beta} \quad (2-11)$$

$$H = \frac{L \cdot \tan \alpha \cdot \tan \beta}{\tan \alpha + \tan \beta} \quad (2-12)$$

Technische Oberflächen reflektieren Licht teilweise diffus. Wird ein diffus reflektierendes Material durch einen Lichtstrahl beleuchtet, so erfolgt eine ungerichtete Rückstreuung, welche sich gemäß des Lambertischen Kosinussatzes in alle Raumrichtungen verteilt.

Durch diese Eigenschaft kann auf eine Ausrichtung des Objektes zur einfallenden Strahlung verzichtet werden, da davon ausgegangen werden kann, dass das Licht anteilig in Richtung des Detektors gestreut wird. [Wol2016, GaTa2011]

In der nachfolgenden Abbildung 2-7 wird der schematische Aufbau eines Triangulationsensors für den Anwendungsfall einer diffusen Reflexion dargestellt. Um den Objektabstand d bestimmen zu können, müssen folgende Größen bekannt sein: der Abstand F / A_L zwischen der Lichtquelle und der Optik, der Abstand E_L (Brennweite / Bildweite), welcher identisch ist mit dem Abstand F und A_L sowie der Basisabstand B von der optischen Achse der Lichtquelle zum Detektor. Die Messung der Strecke x , an welcher der reflektierte gebündelte Lichtstrahl auf den Detektor trifft, wird für die Bestimmung des momentanen Objektabstandes d verwendet (Abbildung 2-7). [Wol2016, GaTa2011]

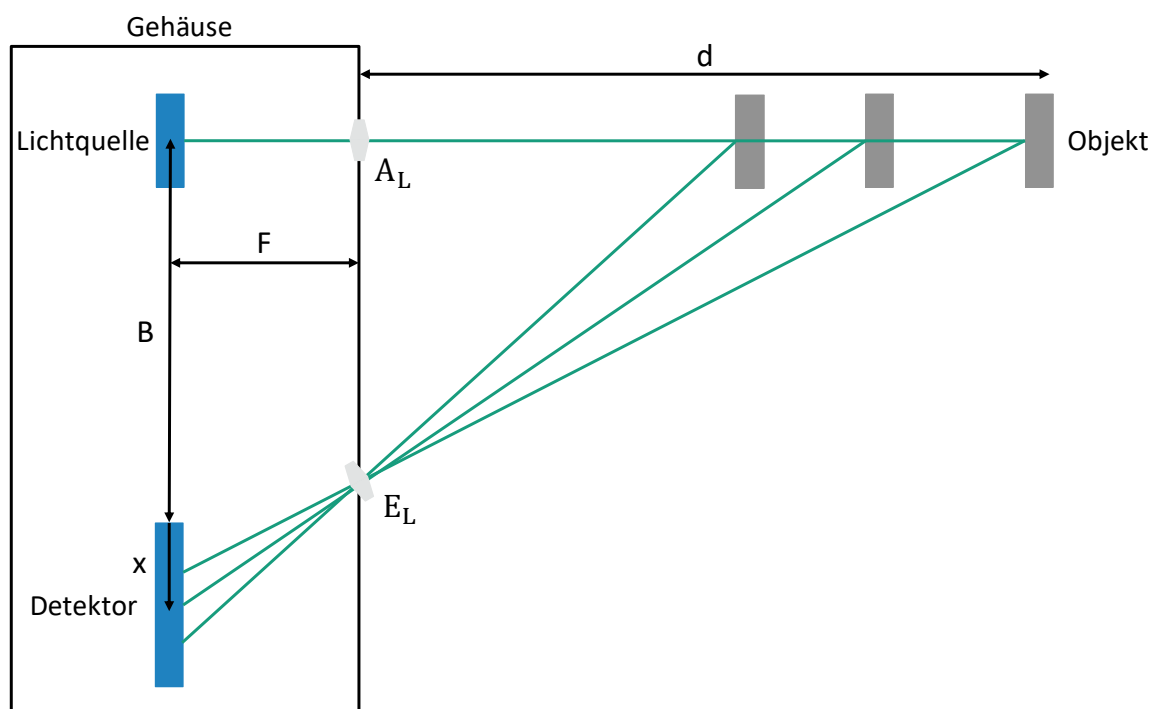


Abbildung 2-7: Schematischer Aufbau eines Triangulationssensors mit Lichtquelle, Objekt und Detektor [Wol2016]

2.3 Grundlagen der Optimierung

Mathematische Methoden zur Findung des Minimums einer bekannten Funktion werden in vielen Bereichen der Wirtschaftswissenschaften, der Technik und der Naturwissenschaft eingesetzt. Der Begriff Optimierung wird im Allgemeinen als die Suchen nach einer optimalen Lösung eines definierten Problems verstanden. Das definierte Problem wird in ein mathematisches Modell umgewandelt und stellt damit die Grundlage zur Optimierung dar. Aus dem Modell lassen sich mithilfe der Optimierungstechniken die unbekanntes Modellparameter oder -funktionen bestimmen. [Gri2013, Ste2018]

Die Optimierung unterteilt sich zunächst in zwei Optimierungsprobleme. Im ersten Bereich werden die statischen Optimierungsprobleme eingeordnet. Unter diesen Problemen wird die Minimierung einer Funktion mit Optimierungsvariablen verstanden. Während sich in den zweiten Bereich die dynamischen Optimierungsprobleme eingliedern, in denen eine Funktion im Zeitbereich optimiert wird. Im Nachfolgenden werden die statischen Optimierungsprobleme detaillierter betrachtet. [Gri2013, Ste2018]

Um eine Problemstellung in ein mathematisches Modell umzuformen, wird eine Standardformulierung des Optimierungsproblems definiert. In der Standardform wird die Zielfunktion $f(x)$, die Gleichungsnebenbedingungen $g_i(x)$, die Ungleichungsnebenbedingungen $h_i(x)$ und die Grundmenge X definiert. In den nachfolgenden Formeln wird ein restringiertes Problem angegeben:

$$\min f(x) \tag{2-13}$$

$$g_i(x) = 0 \quad i \in Y \tag{2-14}$$

$$h_i(x) \leq 0 \quad i \in I \tag{2-15}$$

$$X \in \Omega \tag{2-16}$$

Wird ein Optimierungsproblem ohne Gleichungs- und Ungleichungsnebenbedingungen (Formel (2-14),(2-15)) angegeben, so entsteht ein unrestringiertes Optimierungsproblem. [Ste2018, Mes2015, NoWr2006a, Ven2009] Wenn es unter Umständen nur eine Optimallösung gibt, wird diese mit einem * gekennzeichnet (s. Formel (2-17))

$$f^* = f(x^*) = \inf_{x \in X} f(x) \tag{2-17}$$

Durch die große Anzahl der unterschiedlichen Probleme und der damit entstehenden Vielfalt an Charakteristiken der mathematischen Modelle, werden Optimierungsprobleme im Allgemeinen in die untenstehenden Klassen eingeteilt. Dabei werden die Merkmale der Zielfunktion, der Grundmenge, der Form der Inputargumente und die Ansprüche an die Lösung berücksichtigt. In der Betrachtung der Lösungen wird in die lokale, globale und multikriterielle Lösung unterschieden (Tabelle 2-2).

Weitere und umfangreichere Aufschlüsselungen der einzelnen Optimierungsklassen werden in den nachfolgenden Quellen erläutert [Gri2013, Mes2015, NoWr2006a, Ste2018, Ven2009].

Tabelle 2-2: Problemklassen der Optimierung nach [Gri2013, Mes2015, NoWr2006a, Ven2009]

| Optimierungsproblemklassen | Eigenschaften |
|----------------------------|--|
| Lineare Optimierung | Zielfunktion und Restriktionen sind linear |
| Quadratische Optimierung | Zielfunktion ist quadratisch und die Restriktionen sind linear |
| Nichtlineare Optimierung | Zielfunktion oder mindestens eine Restriktion ist nichtlinear |
| Integer-Optimierung | Alle Variablen sind diskret |

2.4 Optimalitätsbedingungen

Im Nachfolgenden werden die Kriterien betrachtet, die zur Beschreibung der Optimalitätsbedingungen benötigt werden. Aufgrund dieser Bedingungen können die Zielfunktion und deren Lösungen bewertet werden. Des Weiteren kann eine Aussage über die Lösbarkeit des mathematischen Modells und die Auswahl des optimalen Minimierungsalgorithmus getroffen werden.

2.4.1 Minimum

Um Aussagen über ein globales oder lokales Minimum in der zulässigen Menge X zu erhalten, werden die nachfolgenden Definitionen nach [Pie2017, Ste2018] betrachtet. Aus der Definition lässt sich erkennen, dass ein globales Minimum stets auch ein lokales Minimum ist. Eine visuelle Darstellung der unterschiedlichen Minima wird in der Abbildung 2-8 dargestellt. Dabei wird in das lokale Minimum, das globale Minimum und den Sattelpunkt unterschieden. [Pie2017, Ste2018]

Definition des globalen und lokalen Minimums nach [Pie2017, Ste2018]:

Für eine Funktion $f(x)$ mit $f: X \rightarrow \mathbb{R}$ und $X \subseteq \mathbb{R}^n$ besitzt an der Stelle $x^* \in X$

- Ein lokales Minimum, falls $f(x^*) \leq f(x)$ für alle x in hinreichend kleiner Umgebung um x^*
- Ein striktes lokales Minimum, falls $f(x^*) < f(x)$ für alle x in hinreichend kleiner Umgebung um x^*
- Ein globales Minimum, falls $f(x^*) \leq f(x)$ für alle $x \in X$
- Ein eindeutiges globales Minimum, falls $f(x^*) < f(x)$ für alle $x \in X$

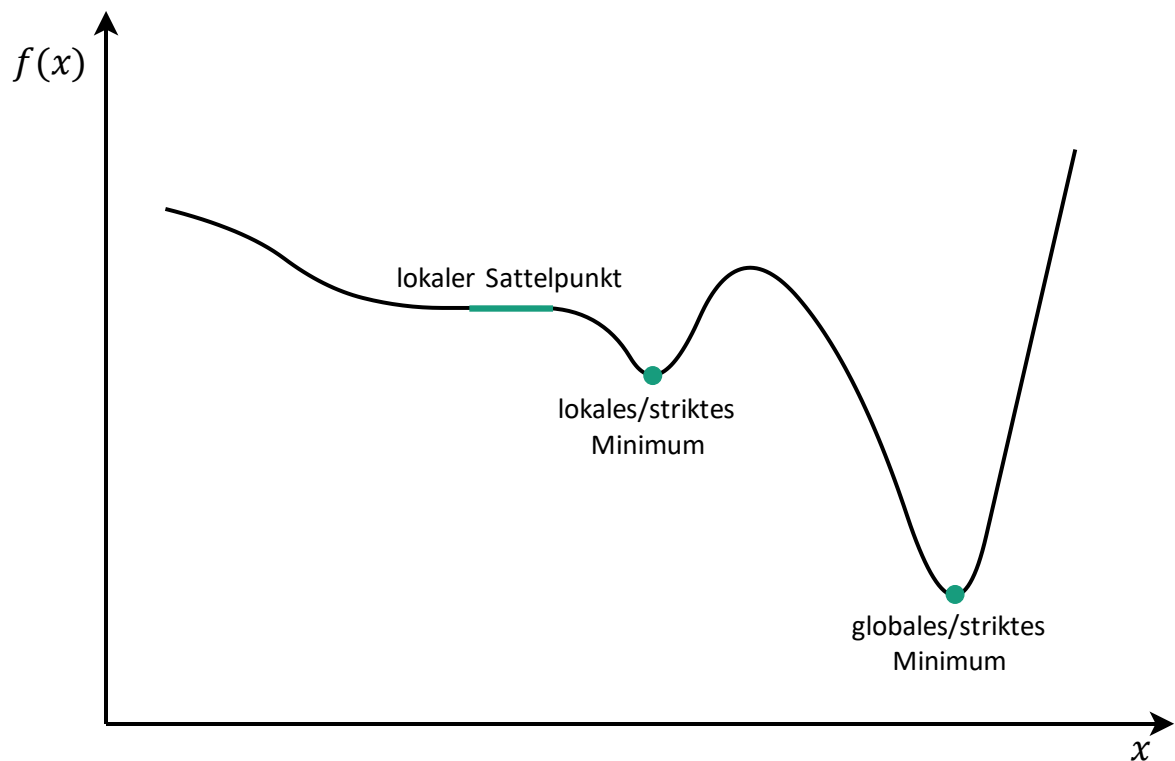


Abbildung 2-8: Verschiedene Minima einer beliebigen Funktion $f(x)$ [Bec2014]

2.4.2 Konvexität

Zur Bestimmung der Lösbarkeit des mathematischen Modells wird die Konvexität einer Menge, einer Funktion oder eines Optimierungsproblems herangezogen. Aufgrund einer Konvexitätsuntersuchung kann eine Aussage über die eindeutige Lösung des Optimierungsproblems getroffen werden. Dabei kann der Begriff der Konvexität auf Mengen, Funktionen und ganzen Optimierungsproblemen angewendet werden.

Konvexe Mengen

In der Betrachtung von Mengen kann eine geometrische Interpretation angewendet werden, welche nachfolgend erläutert wird. In der geometrischen Betrachtung wird eine Verbindungsline zwischen den Punkten $x, y \in X$ definiert. Sind diese Verbindungslinien komplett in $X \subseteq \mathbb{R}^n$ enthalten, so ist die Menge X konvex. Des Weiteren stellen die Schnittmengen einer konvexen Menge ebenfalls eine konvexe Menge dar. In der Abbildung 2-9 werden Beispiele zu konvexen und nichtkonvexen Mengen gezeigt. [Bec2014, Ven2009, Gri2013, Ste2018]

Definition der konvexen Menge [Gri2013, Ste2018]:

Eine Menge $X \subseteq \mathbb{R}^n$ ist konvex, falls die folgende Bedingung für beliebige Punkte $x, y \in X$ erfüllt ist:

$$\text{a) } (1 - \lambda)x + \lambda y \in X \quad \forall x, y \in X, \lambda \in [0, 1]$$

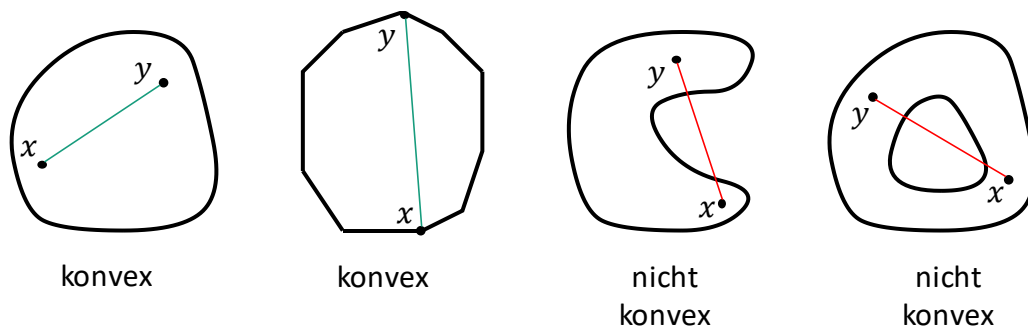


Abbildung 2-9: Beispiele von konvexen Mengen [Ste2018]

Konvexe Funktionen

Die Konvexität einer Funktion wird geometrisch durch die Definition des epi-Graphen, der alle Punkte oberhalb der Funktion abbildet oder die Sekanten durch die Punkte $P_1 = (x_1, f(x_1))$ und $P_2 = (x_2, f(x_2))$ immer oberhalb des Funktionsgraphen liegen (s. Formel (2-18)) [Kab1996, Kön1995, Ven2009]. Aufgrund der nachfolgenden Definitionen wird die Konvexität der Funktion in konvex, streng konvex und streng konkav unterteilt. In der Abbildung 2-10 werden Beispiele zu konvexen und konkaven Funktionen abgebildet. [Ste2018, Ven2009]

$$\text{epi } f := \left\{ \begin{pmatrix} \mathbf{x} \\ r \end{pmatrix} : \mathbf{x} \in \mathbb{R}^n, r \geq f(\mathbf{x}) \right\} \quad (2-18)$$

Definition der konvexen und konkaven Funktionen [Kab1996, Kön1995, Ste2018]:

Ist $X \subseteq \mathbb{R}^n$ eine konvexe Menge. Die Funktion $f : X \rightarrow \mathbb{R}$ ist konvex auf X falls, die folgenden Bedingungen erfüllt sind:

a) Funktion ist konvex

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) \quad \forall x, y \in X, \quad \lambda \in [0,1]$$

b) Funktion ist streng konvex

$$f((1-\lambda)x + \lambda y) < (1-\lambda)f(x) + \lambda f(y) \quad \forall x, y \in X, \quad \lambda \in [0,1]$$

c) Funktion ist (streng) konkav

$-f(x)$ (strikt) konvex

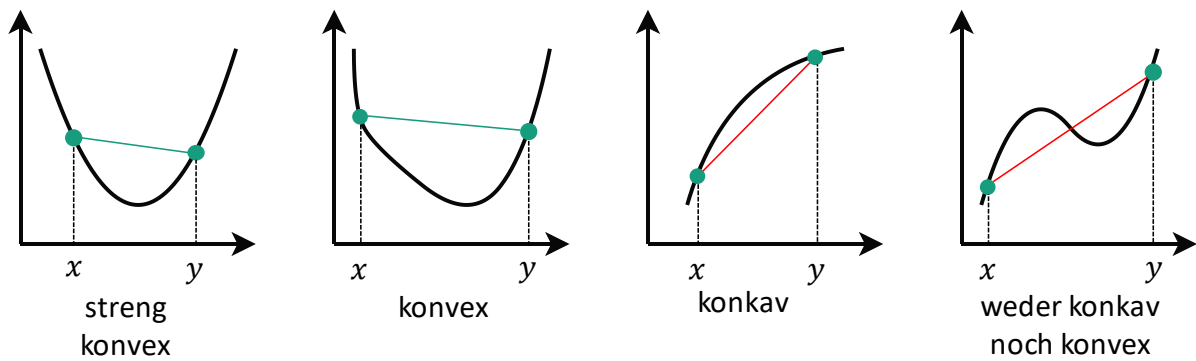


Abbildung 2-10: Beispiel von konkaven und konvexen Funktionen [PLB2012b, Ste2018]

Eine weitere Möglichkeit zur Bestimmung der Konvexität einer Funktion kann über die Definitheit der Hessematrix (s. Kapitel 2.4.3) erreicht werden. In den nachfolgenden Definitionen werden die vier Möglichkeiten der Definitheit der Hessematrix $\nabla^2 f(\mathbf{x})$ aufgezeigt. [PLB2012a, Ven2009, Bec2014]

Definition der Konvexität durch die Hessematrix [Ven2009, Bec2014]:

Sei $\mathbf{X} \subseteq \mathbb{R}^n$ konvex und $f: \mathbf{X} \rightarrow \mathbb{R}$ zweimal stetig differenzierbar:

- f ist konvex: $\nabla^2 f(\mathbf{x})$ für alle $\mathbf{x} \in \mathbf{X}$ positiv semidefinit ist
- f ist streng konvex: $\nabla^2 f(\mathbf{x})$ für alle $\mathbf{x} \in \mathbf{X}$ positiv definit ist
- f ist konkav: $\nabla^2 f(\mathbf{x})$ für alle $\mathbf{x} \in \mathbf{X}$ negativ semidefinit ist
- f ist streng konkav: $\nabla^2 f(\mathbf{x})$ für alle $\mathbf{x} \in \mathbf{X}$ negativ definit ist

Konvexe Optimierungsprobleme

Der Begriff der Konvexität von Mengen und Funktionen lässt sich auch auf die Optimierungsprobleme anwenden. Um ein konvexes Optimierungsproblem zu erhalten, werden konvexe Bedingungen an die Zielfunktion, die zulässige Menge, den Gleich- und Ungleichungsnebenbedingungen gestellt. Dabei wird von der Zielfunktion, der Ungleichungsnebenbedingung und der zulässigen Menge eine Konvexität verlangt. Die Gleichungsbedingung benötigt eine lineare Form, die damit auch das Kriterium einer konvexen Funktion erfüllt. [Bec2014]

Definition der konvexen Optimierungsprobleme [Bec2014]:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} f(x) \\ \text{s.t. } g(x) = 0, \quad i = 1, \dots, p \\ h(x) \leq 0, \quad i = 1, \dots, p \end{aligned}$$

Ist (streng) konvex, wenn:

- a) $g_i : \mathbb{R}^n \rightarrow \mathbb{R}, \quad i = 1, \dots, p \rightarrow \text{linear}$
- b) $h_i : \mathbb{R}^n \rightarrow \mathbb{R}, \quad i = 1, \dots, p \rightarrow \text{konvex}$
- c) $f(x)$ (streng) konvex auf $X \subseteq \mathbb{R}^n$

Aufgrund der angeführten Definition lassen sich weitere Eigenschaften ableiten. Mit der Struktur des konvexen Optimierungsproblems ist ein lokales Minimum immer ein globales Minimum. Des Weiteren besitzt ein streng konvexes Optimierungsproblem nur ein Minimum. Dieses stellt wiederum das globale Minimum dar. [PLB2012a]

2.4.3 Erste und zweite Ordnung der Optimalitätsbedingung

Die Optimalitätsbedingungen dienen zur Auffindung von (lokalen) Extremstellen und dem Überprüfen, ob es sich dabei um ein Minimum handelt. Die x-Werte der Extremstelle werden mit der Variable x^* beschrieben. Um ein Minimum zu finden, muss die nachfolgende Bedingung erfüllt sein. Dabei befinden sich die betrachteten x-Werte hinreichend nahe am x^* . (s. Formel (2-19)) [Bec2014, Ven2009, Ste2018]

$$f(x) \geq f(x^*) \tag{2-19}$$

Für eine Funktion, die stetig zweifach differenzierbar ist, lassen sich die Optimalitätsbedingungen der ersten und zweiten Ordnung definieren. Um eine Aussage über die Minima zu erhalten, wird der Gradient der Funktion gebildet (Formel (2-20)). [Bec2014, Ven2009, Ste2018]

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2-20)$$

Die erste Ordnung erfüllt das Kriterium des Maximums, des Minimums und des Sattelpunktes. Aufgrund der Mehrdeutigkeit kann im optimalen Punkt \mathbf{x}^* keine Aussage über die Art des Extrempunktes gemacht werden. Um eine Aussage über den Extrempunkt zu bekommen, wird die Bedingung der zweiten Ordnung eingeführt (Formel (2-21)). [Bec2014, Ven2009]

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial^2 x_2^2} \end{bmatrix} \geq \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (2-21)$$

Diese Definition bestimmt eine notwendige Optimalitätsbedingung, in welche keine Umkehrbarkeit zulässig ist. Dieses kann zu einer Fehlinterpretation des Minimums führen. Um dieses zu vermeiden, kann ein hinreichendes Optimalitätskriterium definiert werden. Die Definitionen der notwendigen und hinreichenden Optimalitätskriterien werden im Nachfolgenden aufgezeigt. Aufgrund dieser Beschreibung lassen sich Sattelpunkte und Minima unterscheiden. [Bec2014, Ven2009]

Definition der notwendigen und hinreichenden Optimalitätskriterien [Ste2018, Ven2009]:

Ist $f: \mathbb{R}^n \rightarrow \mathbb{R}$ eine zweimal stetig differenzierbare Funktion und $\mathbf{x}^* \in X$ ein lokales Minimum von f gelten folgende Optimalitätsbedingungen der ersten und zweiten Ordnung

- a) Notwendige Optimalitätsbedingungen

$$\begin{aligned} \nabla f(\mathbf{x}^*) &= 0 \\ \nabla^2 f(\mathbf{x}) &\geq 0 \text{ p. s. d} \end{aligned}$$

- b) Hinreichende Optimalitätsbedingungen

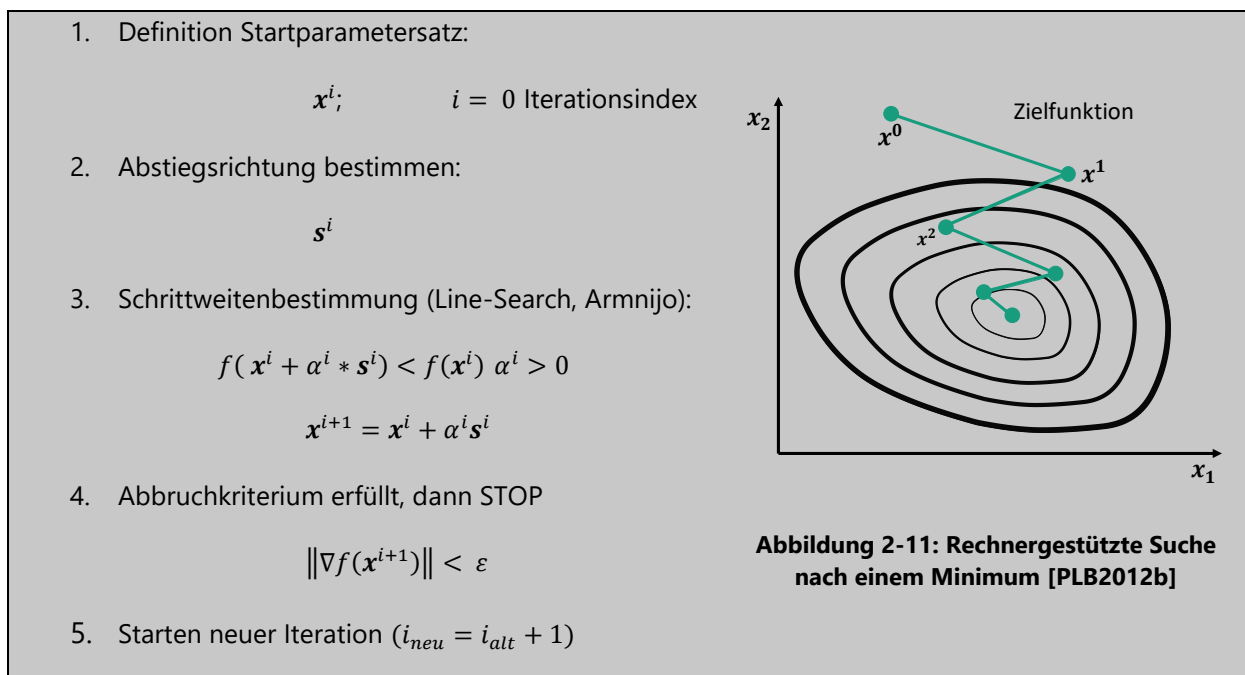
$$\begin{aligned} \nabla f(\mathbf{x}^*) &= 0 \\ \nabla^2 f(\mathbf{x}) &> 0 \text{ p. d} \\ \mathbf{x}^* &= \text{ein strenges lokales Minimum} \end{aligned}$$

2.5 Minimierungsalgorithmen

Um ein mathematisches Modell zu lösen und ein lokales oder globales Minimum bzw. Maximum zu finden, werden Lösungsalgorithmen benötigt. Diese Lösungsalgorithmen können analytisch oder numerisch angewendet werden. Aufgrund der hohen Komplexität und Vielfalt der gesuchten Parameter wird eine analytische Lösung sehr aufwändig. In den nachfolgenden Unterkapiteln werden ausgewählte Minimierungsalgorithmen erläutert.

2.5.1 Algorithmische Struktur der numerischen Verfahren

Um ein unter- oder überbestimmtes mathematisches Modell zu minimieren, werden iterative Lösungsverfahren benötigt. Aufgrund der Definition eines Minimums (s. Kapitel 2.4.1) ist es nicht möglich, ein solches mathematisches Modell hinreichend in einem Schritt zu minimieren. Aufgrund der bestehenden Struktur des Optimierungsproblems und der möglichen Restriktionen ist eine Definition des Startparametersatzes \mathbf{x}^0 nötig. Durch den gegebenen Startpunkt können sich unterschiedliche lokale Minima durch die unterschiedlichen Suchrichtungen ergeben. Im Allgemeinen wird eine Abstiegsrichtung definiert, um in das Minimum der Zielfunktion zu gelangen. Eine allgemeine algorithmische Struktur wird in der Abbildung 2-11 und in der nachfolgenden Auflistung schematisch abgebildet. [PLB2012b, Ste2018]



In der Struktur der iterativen Algorithmen wird nach der Definition des Startparametersatzes \mathbf{x}^0 eine Abstiegsrichtung $\mathbf{s}^{(i)}$ bestimmt. Entlang dieser Abstiegsrichtung wird eine Schrittlänge definiert, die eine Minimierung der Zielfunktion bewirkt $f(\mathbf{x}^i + \alpha^i * \mathbf{s}^i) < f(\mathbf{x}^i)$. Wenn das Abbruchkriterium $\|\nabla f(\mathbf{x}^{i+1})\| < \varepsilon$ nicht erfüllt ist, wird eine weitere Iteration des Algorithmus ausgeführt ($i_{neu} = i_{alt} + 1$). [PLB2012b]

2.5.2 Newton-Verfahren

Dieses Verfahren beruht auf der zweiten Ordnung und benötigt damit die Funktionswerte $f(\mathbf{x})$, den Gradienten $\nabla f(\mathbf{x})$ und die Hessematrix $\nabla^2 f(\mathbf{x})$. Die Grundidee des Newton-Verfahrens basiert auf der Minimierung der quadratischen Approximation der Funktion $f(\mathbf{x})$ an der Stelle \mathbf{x}^k . Aus dieser Vorgehensweise ergibt sich die Formulierung des nächsten Schritts \mathbf{x}^{k+1} wie in der Formel (2-22) abgebildet [Bec2014, Mes2015, NoWr2006b]

$$\mathbf{x}^{k+1} = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^n} \left\{ f(\mathbf{x}^k) + \nabla f(\mathbf{x}^k)^T (\mathbf{x} - \mathbf{x}^k) + \frac{1}{2} (\mathbf{x} - \mathbf{x}^k)^T \nabla^2 f(\mathbf{x}^k) (\mathbf{x} - \mathbf{x}^k) \right\} \quad (2-22)$$

Da die Bedingung der ersten Ordnung erfüllt sein muss und damit $\nabla f(\mathbf{x}) = 0$, kann die Formel (2-22) folgendermaßen ausgedrückt werden. [Bec2014, Mes2015, NoWr2006b]

$$\nabla f(\mathbf{x}^k) + \nabla^2 f(\mathbf{x}^k) (\mathbf{x} - \mathbf{x}^k) = 0 \quad (2-23)$$

Unter Berücksichtigung der zweiten Ordnung - die Hessematrix ist positiv definit - ergibt sich die nachstehende Gleichung für den nächsten Schritt \mathbf{x}^{k+1} (Formel (2-24)). Wobei der zweite Term die Abstiegsrichtung des nächsten Schrittes abschätzt (Formel (2-25)). [Bec2014, Mes2015, NoWr2006b]

$$\mathbf{x}^{k+1} = \mathbf{x}^k - (\nabla^2 f(\mathbf{x}^k))^{-1} \nabla f(\mathbf{x}^k) \quad (2-24)$$

$$\mathbf{d}_k = (\nabla^2 f(\mathbf{x}^k))^{-1} \nabla f(\mathbf{x}^k) \quad (2-25)$$

Sollte die Inverse der Hessematrix nicht existieren, wird das nachstehende Gleichungssystem gelöst, um den nächsten Schritt \mathbf{x}^{k+1} zu erhalten (Formel (2-26) und (2-27)). [Bec2014, NoWr2006b]

$$(\nabla^2 f(\mathbf{x}^k))^{-1} \mathbf{d}^k = -\nabla f(\mathbf{x}^k) \quad (2-26)$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{d}^k \quad (2-27)$$

Wenn eine Hessematrix für alle k -Iterationen existiert, gleicht das Newton-Verfahren einem skalierten Gradientenverfahren. Nachteile bei der Verwendung des Newton-Verfahrens ist, dass der Startparametersatz \mathbf{x}^0 nahe an der gesuchten Lösung liegen muss. Aufgrund der Definition eines Minimums kann das Verfahren nicht zwischen einem Sattelpunkt oder Minimum unterscheiden. Dies führt dazu, dass ein Sattelpunkt als Minimum angesehen wird und das Abbruchkriterium erfüllt ist, ohne ein wirkliches Minimum gefunden zu haben. Ein weiterer Nachteil liegt in der aufwendigen Berechnung der benötigten Inversen der Hessematrix. Dieses wird besonders rechenintensive, wenn höherdimensionale Optimierungsprobleme gelöst werden. Dem gegenüber steht das quadratische Konvergenzverhalten in Richtung des Minimums [Bec2014, Mes2015, NoWr2006b]. Eine detaillierte Betrachtung des Verfahrens kann in den nachfolgenden Quellen entnommen werden. [Bec2014, Mes2015, NoWr2006b]

2.5.3 Gauß-Newton-Verfahren

Das Gauß-Newton-Verfahren ist eine Modifikation des allgemeinen Newton-Verfahrens (s. Kapitel 2.5.2). In der Modifikation handelt es sich im Allgemeinen um die Methode der kleinsten Quadrate. Die Lösung der Zielfunktion wird durch eine quadratische Näherung ersetzt. Durch diese Definition benötigt das Gauß-Newton-Verfahren keine Hessematrix $\nabla^2 f(\mathbf{x})$. Unter der Betrachtung einer nichtlinearen Least-Squares-Problem Formel (2-28)) ergibt sich der nächste Schritt \mathbf{x}^{k+1} der Gauß-Newtonlösung aus der nachstehenden Formel (2-29)). Dabei wird die Summe der quadratischen linearen Approximation $f_i(\mathbf{x}^k)$ minimiert. [Bec2014, NoWr2006b]

$$\min_{\mathbf{x} \in \mathbb{R}^n} \left\{ \sum_{i=1}^m (f_i(\mathbf{x}) - c_i)^2 \right\} = \min \|F(\mathbf{x})\|^2 \quad (2-28)$$

$$\mathbf{x}^{k+1} = \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^n} \left\{ \sum_{i=1}^m f_i(\mathbf{x}^k) + \nabla f_i(\mathbf{x}^k)^T (\mathbf{x} - \mathbf{x}^k) - c_i \right\} \quad (2-29)$$

Das oben dargestellte Minimierungsproblem zeigt im Wesentlichen ein lineares Least Square Problem der nachfolgenden Form. [Bec2014, NoWr2006b]

$$\min_{\mathbf{x} \in \mathbb{R}^n} \|A^k \mathbf{x} - b^k\|^2 \quad (2-30)$$

$$A^k = J(\mathbf{x}^k); \quad b(\mathbf{x}^k) = J(\mathbf{x}^k)\mathbf{x}^k - F(\mathbf{x}^k)$$

Wenn die Jacobi-Matrix keinen vollen Spaltenrang aufweist und nur positiv semidefinit ist, ist keine eindeutige Lösung möglich. Eine weitere explizite Form der Gleichung kann angegeben werden (Formel (2-32)), die sogenannte Least-Square-Solution (Formel (2-31)) für den nächsten Schritt \mathbf{x}^{k+1} . [Bec2014, NoWr2006b]

$$\mathbf{x} = (A^T A)^{-1} A^T b \quad (2-31)$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \left(J(\mathbf{x}^k)^T J(\mathbf{x}^k) \right)^{-1} J(\mathbf{x}^k) F(\mathbf{x}^k) = \mathbf{x}^k - \mathbf{d}^k \quad (2-32)$$

Durch eine weitere Modifizierung in der Schrittweite kann aus den Verfahren Newton und Gauß-Newton ein gedämpfter Algorithmus werden. Aufgrund der Dämpfungsparameter kann ein stabilerer und effizienterer Algorithmus erzeugt werden. [Bec2014] Eine detaillierte Betrachtung des Verfahrens kann in den nachfolgenden Quellen entnommen werden. [Bec2014, NoWr2006b]

2.5.4 Trust-Region-Verfahren

Das Trust-Region-Verfahren weicht von der eingeführten algorithmischen Struktur aus Kapitel 2.5.1 ab, wobei die Suchrichtung und die Schrittweite in dem Verfahren gemeinsam bestimmt werden. Dabei wird zuerst der Suchradius bestimmt und im Nachfolgenden dann die Abstiegsrichtung in dem vertrauenswürdigen Suchradius. Im Allgemeinen wird die Zielfunktion $f(x^k)$ durch das Trust-Region-Verfahren in einer Umgebung von x^k quadratisch approximiert (Formel (2-33)) [NoWr2006b, Ste2018, SuYu2006]

$$m^k(h) = f(x^k) + \nabla f(x^k)^T h + \frac{1}{2} h^T \nabla^2 f(x^k) h \quad (2-33)$$

Es wird das Minimum der quadratischen Approximation unter Berücksichtigung des Trust-Region Radius Δ^k bestimmt (Formel (2-34))

$$\min_{h \in \mathbb{R}^n} m^k(h) = f(x^k) + \nabla f(x^k)^T h + \frac{1}{2} h^T \nabla^2 f(x^k) h \quad \forall \|h\| \leq \Delta^k \quad (2-34)$$

Aus der Minimierung des Unterproblems ergibt sich die Schrittweite h^k (Formel (2-35)).

$$h^k \approx \text{Argmin } m^k(h) \quad \forall \|h\| \leq \Delta^k \quad (2-35)$$

Um eine Bewertung der Lösung aus der bestimmten Schrittweite zu erhalten, wird ein Quotient eingeführt. Dieser Quotient gibt die Güte des lokalen Modells an, in dem der erwartete und der tatsächliche Zielfunktionswert betrachtet wird (Formel (2-36)). [NoWr2006b, Ste2018, SuYu2006]

$$\rho^k = \frac{f(x^k) - f(x^k + h^k)}{m^k(0) - m^k(h^k)} \quad (2-36)$$

Wird der Quotient negativ wird der $f(x^k + h^k) > f(x^k)$ und der Schritt abgelehnt. In diesem Fall wird eine neue Iteration mit einem kleineren Radius Δ^k durchgeführt. Sollte der Quotient eins oder nah an dem Wert eins liegen, existiert eine gute Übereinstimmung zwischen dem Unterproblem m^k und der Zielfunktion $f(x^k)$ und der Radius kann im nachfolgenden Schritt vergrößert werden. Wenn sich h^k in einem festen Akzeptanzniveau befindet, ergibt sich der nächste Punkt wie folgt (Formel (2-37) und (2-38)) [NoWr2006b, Ste2018, SuYu2006]

$$x^{k+1} = x^k - h^k \quad (2-37)$$

$$m^{k+1}(h) = f(x^{k+1}) + \nabla f(x^{k+1})^T h + \frac{1}{2} h^T \nabla^2 f(x^{k+1}) h \quad (2-38)$$

Näherungsweise kann auch der Cauchy-Punkt verwendet werden, um die Schrittweite h_c^k zu bestimmen (Formel (2-39)). Durch dieses Verfahren wird das Modell in Richtung des steilsten Abstiegs minimiert. Eine detaillierte Betrachtung des Verfahrens kann in den nachfolgenden Quellen entnommen werden. [NoWr2006b, Ste2018, SuYu2006]

$$h_c^k = -\alpha_c^k \frac{\nabla f(x^k)}{\|\nabla f(x^k)\|} \quad (2-39)$$

2.6 Hand-Auge-Kalibration mittels 3D Sensors

In der Hand-Auge-Kalibration wird eine Transformationsmatrix zwischen dem Roboter- und dem Sensorkoordinatensystem bestimmt. Die Hand bezeichnet den Flansch oder den verwendeten Endeffektor. Das Auge bei der Kalibrierung ist der verbaute Sensor.

In den nachfolgenden Unterkapiteln werden zwei Kalibrationsansätze des aktuellen Stands der Technik erläutert. Im ersten Ansatz (Kapitel 2.6.1) wird die Hand-Auge-Kalibration von Jianfeng Li, ming Chen, Xuebi Jin, Yu Chen, Zhonghua Ou und Qin Tang detailliert betrachtet [LCJ+2011]. Das abschließende Unterkapitel zeigt einen zweiten Kalibrationsansatz der Hand-Auge-Kalibration von Min Young Kim, Hyung Suck Cho und Jae Hoon Kim auf [KCK2001].

2.6.1 Hand-Auge-Kalibration nach Jianfeng Li, Ming Chen

Der betrachtete wissenschaftliche Artikel behandelt eine Hand-Auge-Kalibration zwischen einem 3D-Sensor und dem Endeffektor des Roboters. In dieser Strategie werden durch zwei translatorische Verfahrenswege des Roboters die Rotationsbereiche bestimmt und durch drei Rotationsbewegungen des Roboters der Translationsbereich ermittelt. Die Vorteile dieser Strategie sind, dass kein Kalibrationsgitter und keine Messung der dazugehörigen Punkte benötigt werden. Ein weiterer Vorteil ist es, dass die Kamerakalibrierung nur einmal vorgenommen wird, wodurch der Kalibrierungsfehler minimiert wird. Der Ansatz nutzt die Unterschiede in den verschiedenen Roboterstellungen in der Berechnung, wodurch der Kalibrierungsfehler weiter minimiert wird. In der nachfolgenden Abbildung 2-13 wird eine schematische Darstellung des physikalischen Modells abgebildet. [LCJ+2011]

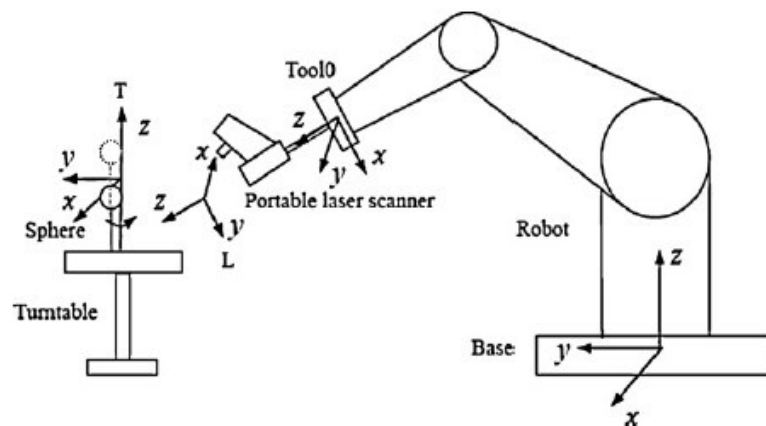


Abbildung 2-12: Schematische Abbildung des verwendeten physikalischen Systems zur Kalibration [LCJ+2011]

In diesem Ansatz wird von der allgemeinen Lösung des Gleichungssystem (Formel (2-43)) ausgegangen. Aufgrund der Verwendung einer Kalibrierkugel mit bekanntem Durchmesser und bekannter Lage im Raum kann die Kalibration in die nachfolgenden Schritte unterteilt werden. Im ersten Schritt wird die Rotationsmatrix bestimmt und im zweiten Schritt der Translationsvektor. [LCJ+2011]

Zur Bestimmung der unbekanntenen Rotationsmatrix R_s wird ein Punkt relativ zur Basis des Roboters angenommen, dessen Koordinaten bekannt sind. Aufgrund dieser Informationen kann die nachfolgende Gleichung aufgestellt werden (Formel (2-40)). Durch das zweifache translatorische Verfahren des Roboters kann die nachfolgende Gleichung definiert werden, wobei der gemessene Vektor vom Sensor zum Punkt mit X_l und der Vektor des definierten Punkts zum Basiskoordinatensystem mit X_b angegeben wird. Aufgrund der reinen translatorischen Bewegung entlang einer Achse bleiben die Rotationsmatrizen $R_{01} = R_{02}$ identisch. Mittels dieser Eigenschaft lässt sich die Darstellung wie folgt vereinfachen (Formel (2-41)). Wird dieses einige Male wiederholt, kann R_s mit einer nichtlinearen Least-Square-Methode gelöst werden. [LCJ+2011]

$$\begin{pmatrix} X_b \\ 1 \end{pmatrix} = \begin{pmatrix} R_0 & T_0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} R_s & T_s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X_l \\ 1 \end{pmatrix} \quad (2-40)$$

$$R_0 * R_s * (X_{l2} - X_{l1}) = T_{02} - T_{01} \quad (2-41)$$

Die im zweiten Schritt verwendete Methode zur Bestimmung des unbekanntenen Translationsvektors T_s wird durch die nachstehende Gleichung beschrieben. In dem die oben gewonnene Rotationsmatrix verwendet wird und der Roboter nur rotatorisch verfahren wird, kann durch die Beziehung des virtuellen Kugelmittelpunktes X_v und dem realen Kugelmittelpunkt der Translationsvektor bestimmt werden (Formel (2-42)). Wird dieses einige Male wiederholt, kann R_s mit einer nichtlinearen Least-Square-Methode gelöst werden. [LCJ+2011]

$$X_a = X_v + R_0 * T_s \quad (2-42)$$

2.6.2 Hand-Auge-Kalibration nach Min Yung Kim, Hyung Suck Cho

Randbedingungen dieser Methode sind, dass zu jeder Zeit die Koordinaten des Kalibrierkörpers im Weltsystem bekannt sind und dass der gemessene Punkt am Kalibrierkörper in Sensorkoordinaten vorliegt. Aufgrund der angegebenen Randbedingungen des Artikels ergibt sich ein Gleichungssystem in der nachfolgenden Art (Formel (2-43)). Dieses System wird durch eine bekannte relative Bewegung des Roboters gelöst. Aufgrund der bekannten Bewegung und dessen Informationen ergibt sich ein weiteres Gleichungssystem der Art (Formel (2-43)). Die entstandenen Gleichungssysteme lassen sich durch eine Umformung zusammenführen und mit einer Least-Square-Methode lösen. [KCK2001]

$$AX = BX \quad (2-43)$$

Um die betrachtete Gleichung zu lösen, wird in der nachfolgenden Abbildung 2-13 das betrachtete physikalische System abgebildet. Die homogenen Koordinaten X_S stellen die gemessene Strecke zwischen dem Sensorkoordinatensystem und dem Kalibrierkörper dar. Die Transformationsmatrix T_R^W und die homogenen Koordinaten X_W sind ausreichend genau bekannt. Die relative Roboterbewegung ΔT ist durch die beiden Roboterpositionen und deren Transformationsmatrizen T_R bekannt. Die beiden Roboterpositionen werden durch die Indizes P und Q gekennzeichnet. [KCK2001]

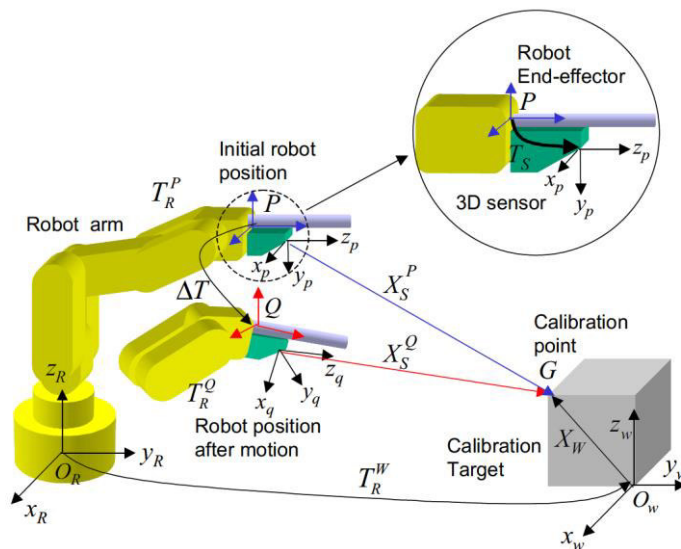


Abbildung 2-13: Relative Roboterbewegung für die Hand-Auge-Kalibration nach [KCK2001]

Das betrachtete physikalische System (Abbildung 2-13) wird in die Form der Gleichung (2-43) gebracht. Die unbekannte homogene Transformationsmatrix T_S wird für X substituiert. An der Stelle der homogenen Transformationsmatrix A wird die relative Roboterbewegung ΔT eingesetzt. Die homogene Transformationsmatrix B wird durch die Multiplikation der homogenen Koordinaten $(X_S^P X_W^T)(X_S^Q X_W^T)^{-1}$ substituiert. Das so entstandene Gleichungssystem (Formel (2-44)) ist unabhängig von der homogenen Transformationsmatrix T_R^W . [KCK2001]

$$\Delta T T_S = T_S (X_S^P X_W^T)(X_S^Q X_W^T)^{-1} \quad (2-44)$$

Zum Lösen der Gleichung (2-44) wird die unbekannte homogene Transformationsmatrix T_S in ein 12×1 Vektor u umgeformt. Dabei werden die homogenen Transformationsmatrizen, die Messungen und die relativen Roboterbewegungen in der neuen Form berücksichtigt. [KCK2001]

Die Gleichung kann so in die unbekanntes Komponenten und die konstanten Werte zerlegt werden und es ergibt sich das nachfolgende lineare System (Formel(2-45)). Dabei besitzt die Matrix D Form 12×12 und der Vektor F die Form 12×1 in denen die konstanten Werte angeordnet sind. (s. Anhang A) [KCK2001]

$$DU = F \quad (2-45)$$

Aus den zwei relativen Bewegungen des Roboters ergibt sich das nachfolgende Gleichungssystem, welches mittels der Least-Square-Method gelöst werden kann (Formel (2-46)). [KCK2001]

$$\begin{bmatrix} D_1 \\ D_2 \end{bmatrix} U = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \quad (2-46)$$

Eine detaillierte Beschreibung des Vorgehens kann der nachfolgenden Quellen entnommen werden. [KCK2001]

3 Methodische Betrachtung

In der nachfolgenden Arbeit wird die Umsetzung und die Implementierung eines Optimierungsalgorithmus in der Robotertechnik aufgezeigt.

Die bisher entwickelten mathematischen Modelle zur Kalibrierung von Robotersystemen werden in wissenschaftlichen Artikeln und Büchern wie [GaTa2011, KCK2001, LCJ+2011] erläutert. In der detaillierteren Betrachtung der Optimierungsstrategien in der Hand-Auge-Kalibration ergeben sich unterschiedliche Umsetzungsmöglichkeiten. Im Wesentlichen unterscheiden sich die methodischen Vorgehensweisen in den verwendeten mathematischen Modellen und in deren Lösungen. In den veröffentlichten Artikeln und den exemplarisch erläuterten Vorgehensweisen im Stand der Technik (s. Kapitel 2.6.1 und Kapitel 2.6.2) werden die verwendeten Algorithmen zur Lösung der unterschiedlichen unter- und überbestimmten Systeme nicht detailliert betrachtet. Um diese abzubilden, wird die im folgenden aufgezeigte Vorgehensweise verwendet (Tabelle 3-1).

Im Stand der Technik werden die benötigten mathematischen Grundlagen sowie die numerische Umsetzung der Roboterbewegungen erläutert. Des Weiteren werden die aktuellen Verfahren zur Kalibrierung des Hand-Auge-Systems an Robotern aufgezeigt. Bei der Recherche der einzelnen Kalibriermethoden werden die Optimierungsalgorithmen nicht detailliert erläutert. Um diese Information darzustellen, wird in den nachfolgenden Schritten eine Methode zur Analyse des physikalischen Systems und dessen Umsetzung in einen Optimierungsalgorithmus beschrieben.

Nach der Ermittlung des mathematischen Modells wird eine Analyse dessen vorgenommen. Aus der vorgenommenen Analyse ergeben sich die zur Verwendung möglichen Algorithmen. Aufgrund einer Bewertung der relevantesten Algorithmen werden zwei Minimierungsalgorithmen herangezogen und umgesetzt.

Aufgrund der numerischen Umsetzung der Minimierungsalgorithmen wird eine Erprobung zur Identifikation von systematischen Fehlern benötigt. Diese Erprobung wird durch die zwei aufeinander folgenden Abläufe realisiert. Im ersten Schritt wird eine analytische Erprobung mit gleichzeitiger visueller Ausgabe des Ergebnisses durchgeführt. Aus diesem Vorgehen können systematische Fehler in der numerischen Umsetzung erkannt werden, in dem sie mit einer analytischen Lösung verglichen werden. Mit einer zusätzlichen Simulation des verwendeten realen mathematischen Modells, mit zufällig gewählten Inputargumenten, wird eine weitere Überprüfung des Optimierungsalgorithmus vorgenommen.

Die methodische Vorgehensweise schließt mit einer Auswertung der gewonnenen Ergebnisse aus den Erprobungen und dem Vergleich der beiden ausgewählten Optimierungsalgorithmen ab.

Tabelle 3-1: Strategie zur Bestimmung geeigneter Optimierungsalgorithmen

| Grundlagenbetrachtung | |
|--|---|
| Stand der Technik | |
| Grundlagen der Robotertechnik | Grundlage der Optimierung |
| <ul style="list-style-type: none"> • Aufbau • Handhabung • Interne Berechnung | <ul style="list-style-type: none"> • Grundlagen der Optimierung • Mathematische Grundlagen • Minimierungsalgorithmen |
| Methode zur Identifizierung und Lösung des mathematischen Modells | |
| Betrachtung des physikalischen Systems | |
| Identifizieren des mathematischen Modells | |
| Identifizieren der benötigten Randbedingungen | Versuchsstrategie |
| Auswahl der Minimierungsalgorithmen | Versuchsaufbau und Messstrategie |
| Umsetzungen der Algorithmen | |
| Analytische Umsetzung der ausgewählten Optimierungsalgorithmen | |
| Erprobung an definierter Funktion | Visuelle Darstellung der erprobten Funktionswerte |
| Implementierung und Umsetzung | |
| Implementierung der Einzelfunktionen | Erprobung der Einzelfunktionen und Bewertung |
| Zusammensetzen der Einzelfunktionen zum definierten Optimierungsalgorithmen | |
| Simulation der Algorithmen mittels zufälligen Inputargumenten am realen mathematischen Modell | |
| Lösen des mathematischen Modells mittels der Optimierungsalgorithmen | |
| Aufnahme realer Messdaten | |
| Umwandlung der Messdaten in eine Input-Datei | |
| Lösen mittels Optimierungsalgorithmus „A“ | Lösen mittels Optimierungsalgorithmus „B“ |
| Auswertung der Ergebnisse | |
| Zusammenfassung und Ausblick | |

4 Physikalisches und mathematisches Modell

In diesem Kapitel wird eine Analyse bekannter physikalischer Modelle und deren Umsetzung in ein mathematisches Modell durchgeführt. Anhand der gewonnenen Erkenntnisse wird ein physikalisches System mit zugehörigem mathematischem Modell zur Hand-Auge-Kalibration entwickelt.

4.1 Analyse bekannter physikalischer Systeme und mathematischer Modelle

Im Nachfolgenden werden bekannte mathematische Modelle zur Hand-Auge-Kalibration analysiert. Dies ermöglicht eine effiziente Entwicklung des mathematischen Modells für das physikalische System, welches in dieser Arbeit untersucht wird. In der nachfolgenden Betrachtung werden die unterschiedlichen physikalischen Systeme analysiert. Dabei wird auf die einzelnen Systemkomponenten, die Kalibrierkörper und die mathematischen Modelle eingegangen.

Der erste Kalibrieransatz der Hand-Auge-Kalibration von Min Young Kim, Hyung Suck Cho und Jae Hoon Kim [KCK2001] besteht aus den physikalischen Systembauteilen: Roboter, Endeffektor, Sensor und dem kubischen Kalibrierkörper (s. Kapitel 2.6.2). Das sich aus diesem System ergebene mathematische Modell hat die allgemeine Form $AX = XB$. Eine Lösung des Modells wird über das Vermessen der Kalibrierkörperecken und der bekannten relativen Roboterbewegung ermöglicht. Bei der Lösung des Gleichungssystems wird eine Umformung auf ein lineares Gleichungssystem der Form $D_i U = F_i$ durchgeführt. Zur Lösung des mathematischen Modells wird eine Least-Squares-Method verwendet (s. Kapitel 2.6.2). [KCK2001]

In dem zweiten Kalibrieransatz der Hand-Auge-Kalibration von Jianfeng Li, ming Chen, Xuebi Jin, Yu Chen, Zhonghua Ou und Qin Tang [LCJ+2011] besteht das physikalische System aus den Systembauteilen: Roboter, Endeffektor, Sensor, sphärische Kalibrierkörper und einem Positionierer (s. Kapitel 2.6.1). Das sich allgemein ergebene mathematische Modell hat die Form $AX = XB$. Die Lösung des Modells wird im Vergleich zum vorherigen Ansatz nach [KCK2001] in zwei Schritten umgesetzt. Im ersten Schritt wird durch eine reine translatorische Bewegung des Roboters die Rotationsmatrix bestimmt. Aufgrund der bekannten Rotationsmatrix kann durch eine reine Rotation des Roboters die Translationsmatrix bestimmt werden. Die jeweilige Lösung der einzelnen Gleichungssysteme wird mittels einer Least-Square-Method ermittelt (s. Kapitel 2.6.1). [LCJ+2011]

Eine weitere Möglichkeit zur Hand-Auge-Kalibration zwischen einem Roboter und einem Sensor weicht von den zuvor betrachteten Kalibrieransätzen ab. In diesem Ansatz wird ein Referenzsystem zum bestehenden Robotersystem eingeführt. Dieses System ist ein Lasertracker mit drei einzelnen Tripelspiegeln. Aufgrund der definierten Montage der drei Tripelspiegel an dem Sensor kann mittels Lasertracker und der bekannten Roboterbewegung die Position des Sensors am Roboter bestimmt werden. [WRQ2001]

In der nachfolgenden Tabelle 4-1 werden die betrachteten Kalibrieransätze hinsichtlich des Aufwands, der Komplexität des mathematischen Modells, des Kalibrierkörpers und der Kosten gegenübergestellt. Dabei werden die niedrigen Kosten sowie der niedrige Aufwand mit einem positiven Effekt verbunden. Die quantitative Bewertung geht von sehr ungünstig (1) bis sehr gut (10).

Tabelle 4-1: Analysen Bewertung der einzelnen Kalibrieransätze

| Kalibrieransatz | Bewertung | Bewertung |
|--|---|-----------|
| Kubischer Kalibrierkörper [KCK2001] | - Aufwändige Kalibrierkörperfertigung und Positionsbestimmung in Roboterkoordinaten, sensibles System | 3 |
| | - Hoher Aufwand der Kalibrierkörper Positionsbestimmung | 3 |
| | - Niedriger Aufwand in dem Erzeugen der Inputargumente | 8 |
| | - Niedriger Aufwand zum Lösen des mathematischen Modells | 8 |
| - Summe | | 22/50 |
| Sphärischer Kalibrierkörper [LCJ+2011] | - Niedrige Anschaffungskosten für den sphärischen Kalibrierkörper, robustes System | 9 |
| | - Niedriger Aufwand der Kalibrierkörper Positionsbestimmung | 8 |
| | - Mittlerer Aufwand in der Generierung der Inputargumente | 6 |
| | - Hoher Aufwand zum Lösen des mathematischen Modells aufgrund der getrennten Berechnung von Translations- und Rotationsmatrizen | 3 |
| - Summe | | 26/50 |

| | | |
|--------------|---|-------|
| Lasertracker | - Hohe Anschaffungskosten des Lasertrackers und der Tripel- spiegel, sensibles System | 3 |
| | - Mittlerer Aufwand der Kalibrierkörper Positionsbestimmung (Tripelspiegel) | 5 |
| | - Hoher Aufwand zur Generierung der Inputargumente | 3 |
| | - Hoher Aufwand zum Lösen des mathematischen Modells, Komplexes mathematisches Modell (Minimierung der sys- tematischen Fehler im Modell) | 5 |
| - Summe | | 16/50 |

Das nachfolgende physikalische Modell wird auf der Grundlage der gewonnenen Ergebnisse aus diesem Kapitel entwickelt. Um die Kosten im physikalischen System zu reduzieren, wird ein sphärischer Kalibrierkörper auf einem 2-Achs-Positionierer verwendet (s. Kapitel 4.2). Zur effizienten Lösung des mathematischen Modells wird auf eine zweistufige Lösung hinsichtlich der Rotations- und Translationsmatrix verzichtet (s. Kapitel 4.4).

4.2 Physikalisches System

Das in dieser Arbeit untersuchte physikalische System wird in der nachfolgenden Abbildung 4-1 schematisch dargestellt. In dem betrachteten System wird ein 6-Achs-Industrieroboter KUKA KR 60 HA der Firma KUKA, einen 2-Achs-Positionierer (KUKA KP1-V), ein Laserschweißkopf der Firma Trumpf und ein Sensor scanCONTROL 2910-100 der Firma Micro-Epsilon verwendet (s. Anhang B1 und B2). Die Wiederholgenauigkeit des verwendeten Roboters KUKA KR 60 HA beträgt laut Herstellerangaben $\pm 0,05$ mm [KUR2005].

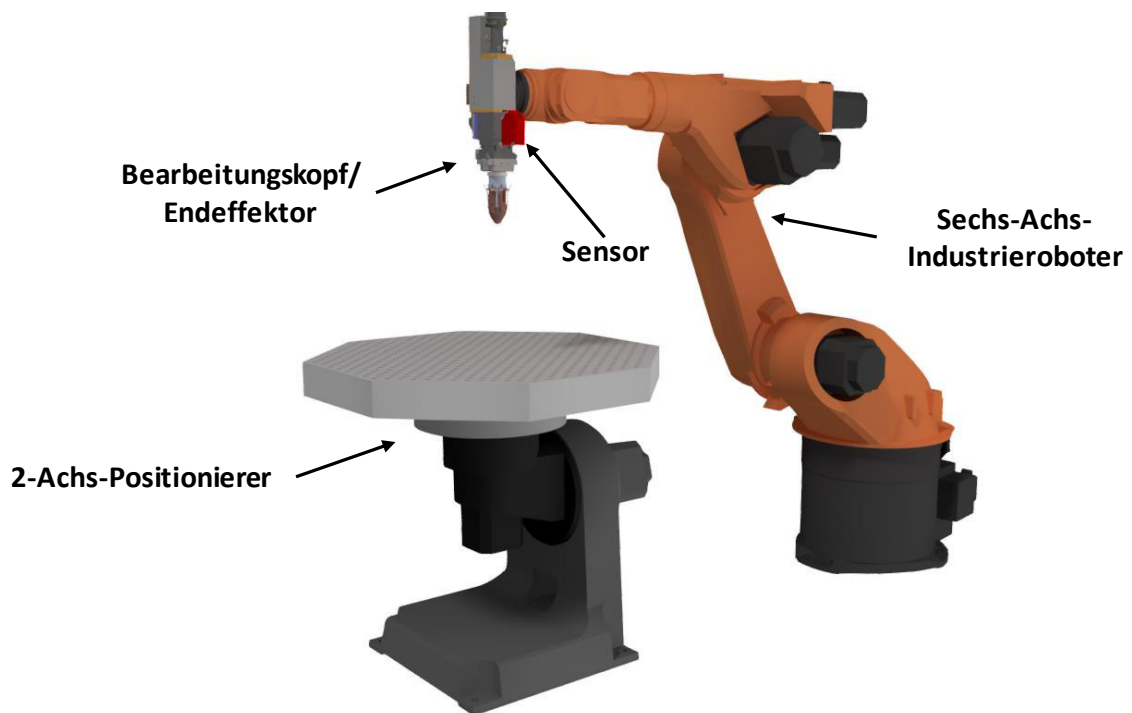


Abbildung 4-1: Reales System bestehend aus Industrieroboter, 2-A-Positionierer, Endeffektor und Sensor

Der für die Arbeit verwendete Sensor wird mittels einer Aufhängung an dem Schweißkopf angebracht. Die Ausrichtung des Sensors ist durch die verwendete Aufhängung fest vorgegeben. Im Idealfall wird ein paralleler Verlauf der Laserline zu der z-Achse des TCP-Koordinatensystems erzielt. (Abbildung 4-2)

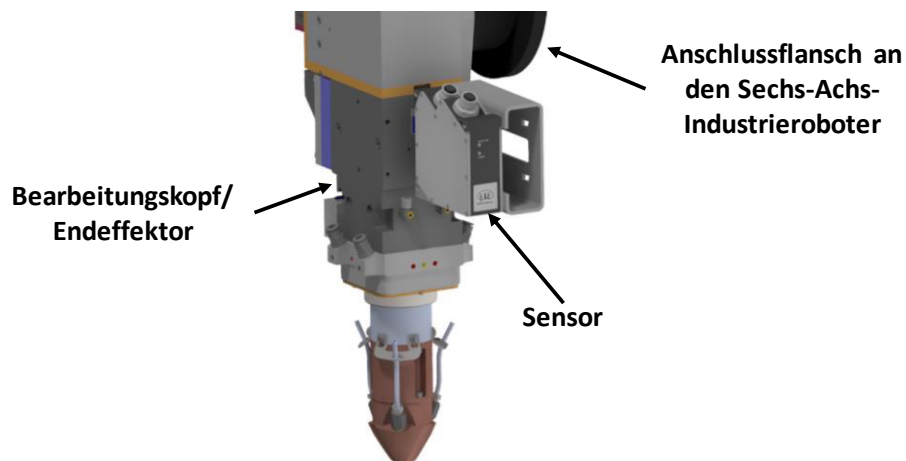


Abbildung 4-2: Detaillierte Betrachtung der Sensoranbringung am Endeffektor

4.3 Kalibration im physikalischen System

In dem betrachteten System ergeben sich unbekannte und ungenaue Positionsdaten der einzelnen Systembauteile zueinander. Um eine hohe Genauigkeit in dem System zu erreichen und eine ausreichende Absolutgenauigkeit und Wiederholgenauigkeit zu erhalten, sind Kalibrationen der einzelnen Systembauteile zueinander sowie im Systembauteil selbst nötig.

In der Kalibration der einzelnen Systembauteile zueinander werden die relativen Positionen und Orientierungen bestimmt. Im Allgemeinen wird die Kalibrationen des betrachteten Robotersystems gemäß der nachfolgenden Abbildung 4-3 aufgeteilt. Die Kalibrationen des Roboters (Aktuatorbasis-KS zum Flansch-KS) und des Sensors werden als interne Kalibrierung bezeichnet. Diese internen Kalibrierungen der einzelnen Systembauteile werden von den Herstellern durchgeführt. Bei der Kalibration des Roboters zum Endeffektor werden von den Herstellern bereits fertige Lösungen angeboten, welche in industrielle Betrieben Anwendung finden. Ist eine Kalibration zwischen Roboter und Endeffektor vorhanden, kann eine Hand-Auge-Kalibration zwischen dem Endeffektor und dem Sensor vorgenommen werden. Sollen an einem Roboter abwechselnd mehrere Bearbeitungsköpfe geführt werden, muss bei jedem Wechsel eine erneute Hand-TCP-Kalibration durchgeführt werden. Sollte dieses nicht möglich sein, kann eine zweite Hand-Auge-Kalibration zwischen dem Flansch-Koordinatensystem und dem Sensor durchgeführt werden. [KCK2001, LCJ+2011]

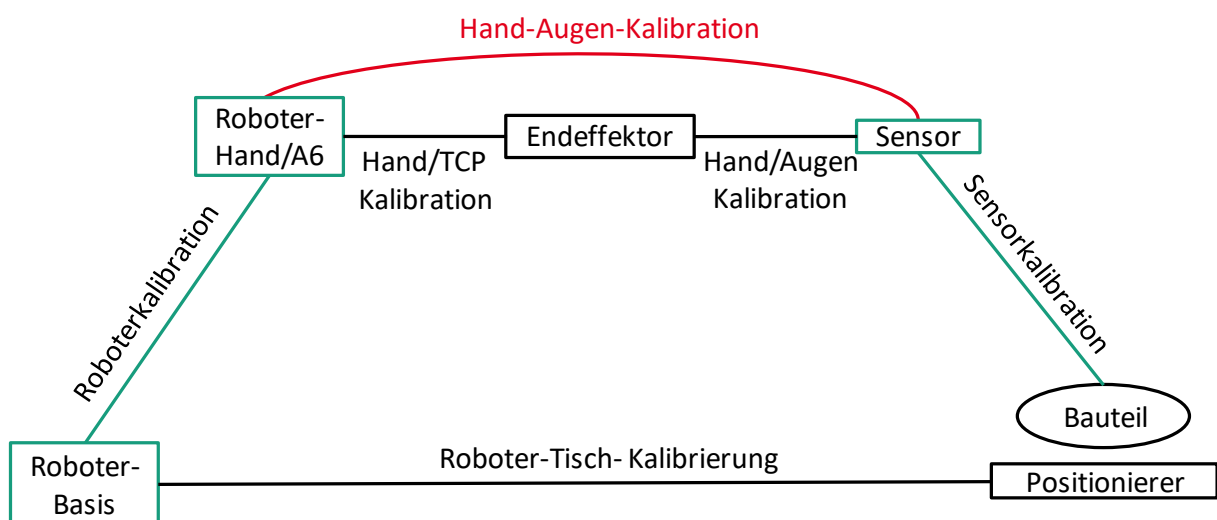


Abbildung 4-3: Schematische Darstellung der Kalibrationen am betrachteten Robotersystem

Durch diese Kalibrationen kann mittels des Sensors und einem Kalibrierkörper die Roboter-Tisch-Kalibrierung durchgeführt werden [LCJ+2011]. Durch das Fehlen einer Kalibration in dem Kreislauf ergeben sich bei dem im nachfolgenden Kapitel 4.4 betrachteten mathematischen Modell systematische Fehler. Diese Fehler können in der Hand-Auge-Kalibrierung einen wiederholten Einfluss haben, wodurch die Kalibration einer verringerten Genauigkeit aufweist.

4.4 Mathematisches Modell

In dem nachfolgenden Abschnitt wird auf die Umsetzung des mathematischen Modells eingegangen. Die Umsetzung stützt sich dabei auf die im Stand der Technik angeführten Informationen zum Lösen mathematischer Modelle und dem entwickelten physikalischen System aus dem Kapitel 4.2. Im abschließenden Unterkapitel wird die Strategie zur Lösung des mathematischen Modells erläutert.

4.4.1 Entwickeltes mathematisches Modell

Das in dieser Arbeit untersuchte mathematische Modell wird auf der Grundlage des im Kapitel 4.2 eingeführten physikalischen Systems entwickelt. Aufgrund der Informationen aus den im Kapitel 4.1 beschriebenen mathematischen Modellen wird eine Anpassung des betrachteten physikalischen Systems benötigt. Das im Nachfolgenden entwickelte mathematische Modell stellt eine Hand-Auge-Kalibration zwischen dem Flansch an der letzten Achse (A6) des Roboters und dem verwendeten Triangulationssensors dar.

Die Anpassung des physikalischen Systems besteht in der zusätzlichen Einbringung eines Kalibrierkörpers. Aufgrund des Kalibrierkörpers wird ein definierter Zusammenhang zwischen dem Sensor und dem Roboter gewährleistet. Für eine Überführung des physikalischen Systems in ein mathematisches Modell wird eine detaillierte Betrachtung der verwendeten Systembauteile benötigt. In der nachfolgenden Abbildung 4-4 wird das globale physikalische System dargestellt.

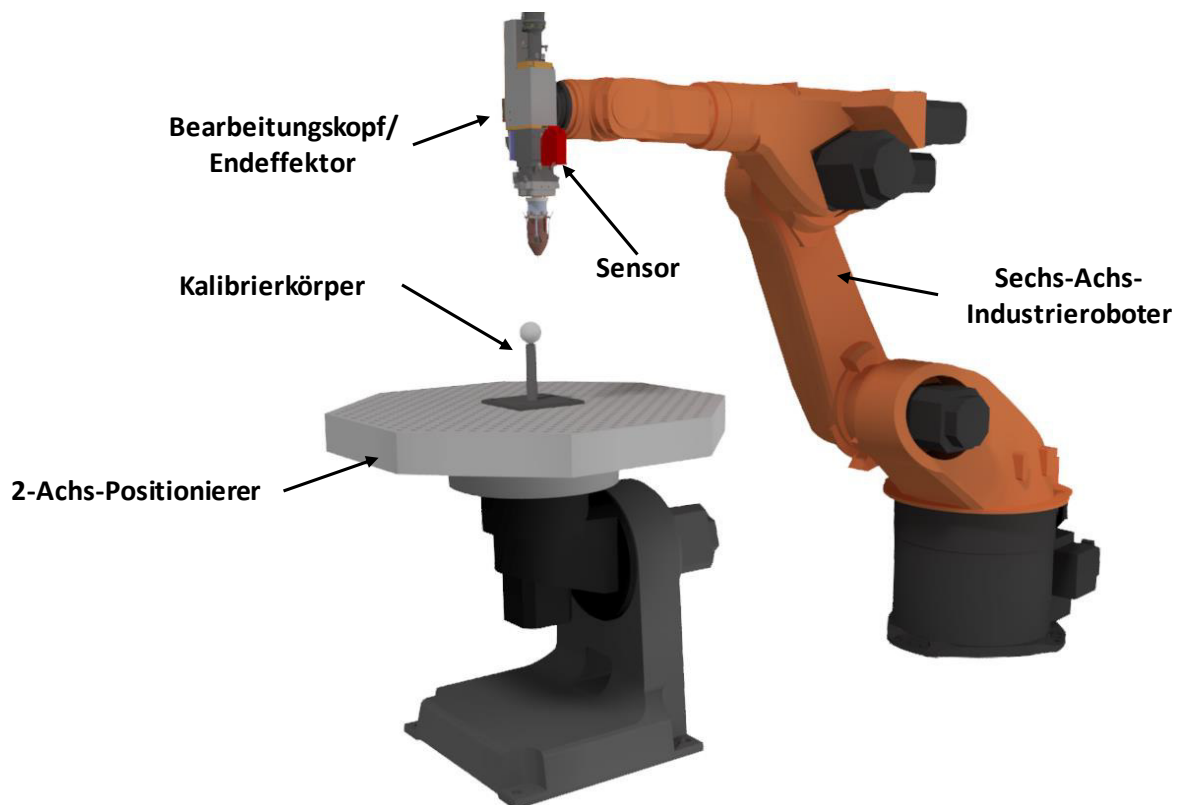


Abbildung 4-4: Angepasstes physikalisches System, Grundlage des mathematischen Modells

Das mathematische Modell wird anhand der definierten Koordinatensysteme der einzelnen Systembauteile entwickelt. Durch die Verwendung der einzelnen Koordinatensysteme wird eine eindeutige und definierte Lage der einzelnen Systeme gewährleistet. Die Beschreibung zwischen den Positionen der einzelnen Koordinatensysteme wird durch die Formulierung der homogenen Transformationsmatrizen umgesetzt (s Kapitel 2.1.2).

Die für die Entwicklung des mathematischen Modells benötigten Koordinatensysteme werden in der nachfolgenden Abbildung 4-5 schematisch dargestellt. Aufgrund der internen Roboterkalibrierung ist eine Ermittlung der Transformationsmatrix T_A^F zwischen der Aktuatorbasis und dem Flansch des Roboters möglich. Eine Identifizierung der Transformationsmatrix T_P^A zwischen der Aktuatorbasis und der Grundplatte des Kalibrierkörpers ist nur bedingt möglich. Die Ermittlung der Lage des Mittelpunkts P' der Kugel im Sensor- und Kalibrierkörperkoordinatensystem ist durch die interne Kalibrierung des Sensors und der Vermessung des Kalibrierkörpers in einer 3D-Koordinatenmessmaschine X0 107 3D der Firma Wenzel mit einer Genauigkeit vom $1,7 \mu m$ ausreichend genau bestimmbar (s. Anhang C13). Das Bestimmen der Transformationsmatrix T_F^S zwischen dem Roboterflansch und dem Sensorkoordinatensystem ist nur durch die Konstruktionsdaten in einer CAD-Software möglich. Die so ermittelte Transformationsmatrix T_F^S ist aufgrund von Fertigungstoleranzen und Montagetoleranzen nicht ausreichend genau und stellt die zu ermittelnde Unbekannte dar.

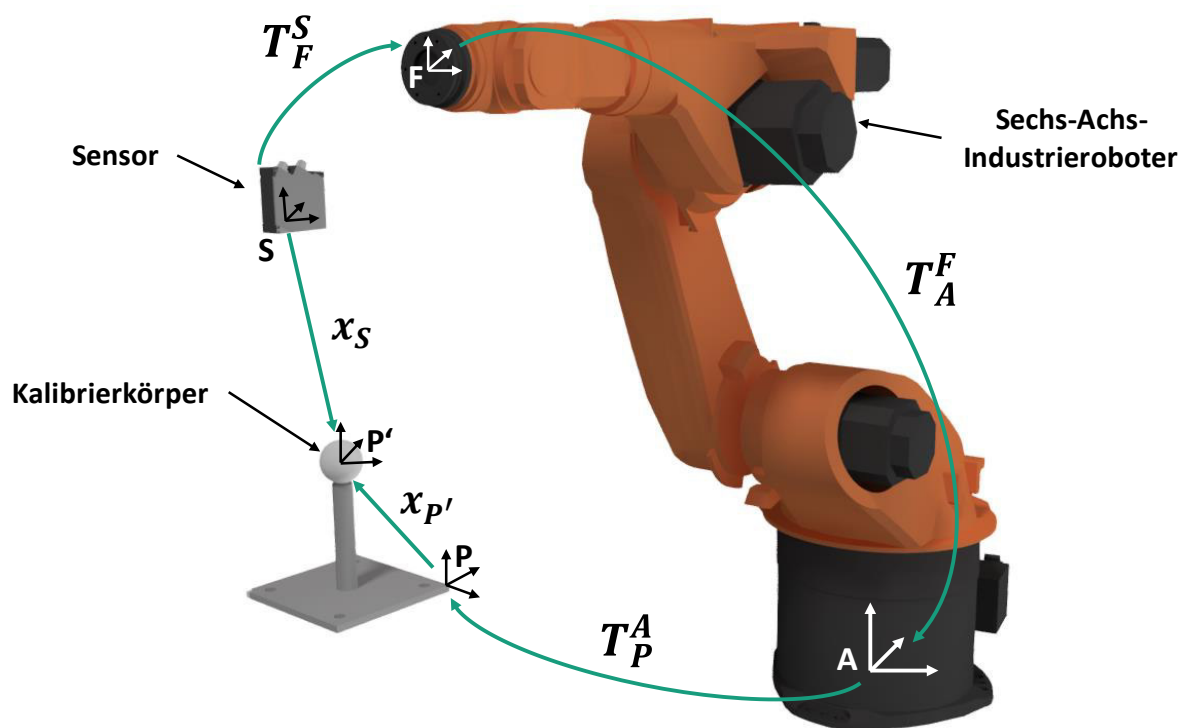


Abbildung 4-5: Schematische Darstellung des entwickelten mathematischen Modells

Das entwickelte mathematische Modell zur Hand-Auge-Kalibration wird in der nachfolgenden Formel (4-1) abgebildet. Die Gleichung besteht aus drei Transformationsmatrizen und zwei Vektoren. Mit Hilfe des entwickelten mathematischen Modells kann eine Hand-Auge-Kalibration zwischen dem Roboterflansch und dem Sensor realisiert werden.

$$T_P^A * x_{P'} = T_A^F * T_F^S * x_S \quad (4-1)$$

In der Gleichung (4-1) werden mathematisch beschreibbare Wege von der Aktuatorbasis zum Mittelpunkt der Kalibrierkugel beschrieben. Auf der linken Seite der Gleichung wird der Weg von der Aktuatorbasis über den Kalibrierkörper beschrieben. Dabei wird die homogene Transformationsmatrix T_P^A mit dem Vektor $x_{P'}$ von der Grundplattenecke bis zum Kugelmitelpunkt multipliziert. Auf der rechten Seite der Gleichung wird der zweite mögliche Weg über den Flansch des Roboters zum Kalibrierkörpermittelpunkt beschrieben. Auf der Seite der Gleichung werden die Transformationsmatrizen von der Aktuatorbasis zum Flansch (T_A^F) und vom Flansch zum Sensor (T_F^S) mit dem Vektor x_S vom Sensor zum Kalibrierkugelmittelpunkt multipliziert. Theoretisch betrachtet wird durch beide Wege der gleiche Punkt ausgehend von der Aktuatorbasis beschrieben.

Die Gleichung (4-1) lässt sich in dieser Form nicht optimal minimieren, um die unbekanntenen homogenen Transformationsmatrizen zu ermitteln. Durch ein Umformen der Gleichung in die nachfolgende Anordnung wird eine optimale Optimierung ermöglicht (Gleichung (4-2)). Aufgrund dieser Umformung ergibt sich auf der rechten Seite der Idealwert Null.

$$0 = (T_A^F * T_F^S * x_S) - (T_P^A * x_{P'}) \quad (4-2)$$

4.4.2 Lösung des mathematischen Modells

Die Lösung des mathematischen Modells in der Form der Gleichung (4-2) wird mittels eines Minimierungsalgorithmus umgesetzt. Die Auswahl eines geeigneten Minimierungsalgorithmus wird in dem nachfolgenden Kapitel 5 erläutert.

In der Gleichung (4-2) werden drei homogene Transformationsmatrizen und zwei Vektoren definiert. Die Vektoren $x_{P'}$ und x_S werden durch das Vermessen des Kalibrierkörpers und der gemessenen Sensordaten ermittelt. Die homogene Transformationsmatrix T_A^F ergibt sich aus der Istposition des Roboters. Die Genauigkeiten der so gewonnenen Werte sind aufgrund der vorherigen Kalibrierungen des Roboters und des Sensors sowie des Vermessens des gefertigten Kalibrierkörpers in einer 3D-Koordinatenmessmaschine ausreichend genau (s. Anhang C12 und Kapitel 6.4.1). Die Ermittlung der homogenen Transformationsmatrix T_P^A vom der Aktuatorbasis zur Grundplatte des Kalibrierkörpers wird durch den Roboter bestimmt.

In der theoretischen Betrachtung kann das mathematische Modell hinsichtlich der letzten Unbekannten T_F^S mittels eines Minimierungsalgorithmus gelöst werden.

Aufgrund der numerischen Lösung des mathematischen Modells ergeben sich zwangsläufig Rundungsungenauigkeiten. Diese Ungenauigkeiten resultieren aus der Darstellung der Werte in binäre Gleitkommazahlen nach der Norm IEEE754 in Computern. Aufgrund dieses Phänomens wird ein Residuum ε definiert. Dieses Residuum wird zusätzlich als Abbruchkriterium des Algorithmus definiert und stellt somit eine standardisierte Genauigkeit der Ergebnisse dar (Gleichung (4-3)).

$$\varepsilon = (T_A^F * T_F^S * x_S) - (T_P^A * x_{P'}) \quad (4-3)$$

Das zu minimierende mathematische Modell in der Form der Gleichung (4-3) besteht aus einer Unbekannten homogenen Transformationsmatrix T_F^S . Aufgrund der Umrechnung von Posen in Transformationsmatrizen ergeben sich sechs zu bestimmende Unbekannte, um die Gleichung hinreichend genau zu lösen.

Das Lösen des mathematischen Modells (Gleichung (4-3)) wird aufgrund der Positionsbestimmung des Kalibrierkörpers in zwei Schritten durchgeführt. Bei der Bestimmung der Position des Kalibrierkörpers ergeben sich unzureichende Genauigkeiten. Des Weiteren geht der systematische Fehler der internen Roboterkalibration ein weiteres Mal mit in die Berechnung ein. Um den so eingebrachten systematischen Fehler wieder zu eliminieren, wird ein weiterer Durchlauf des Minimierungsalgorithmus benötigt. Durch die unzureichend genaue Bestimmung des Kalibrierkörperposition ergeben sich sechs weitere Unbekannte T_P^A .

Aufgrund der angeführten Ungenauigkeiten und den zwölf Unbekannten ergibt sich ein vierfach unterbestimmtes Gleichungssystem. Dieses resultiert aus den drei gewonnenen Informationen pro Messung. Die Anzahl an Messungen für die Lösung des unterbestimmten Gleichungssystems und deren Vermessungsstrategie wird im Kapitel 6.4 erläutert.

In einer zwei stufigen Optimierungsstrategie wird im ersten Schritt die homogene Transformationsmatrix T_F^S zwischen dem Sensor und dem Flansch bestimmt. Zur Steigerung der Genauigkeit kann im zweiten Schritt über die homogenen Transformationsmatrizen T_F^S und T_P^A minimiert. Ausgangspunkt sind dabei die gewonnenen Ergebnisse aus der Minimierung des ersten Schritts. In der nachfolgenden Tabelle 4-2 wird die Optimierungsstrategie der zwei stufigen Optimierung detailliert betrachtet.

Tabelle 4-2: Optimierungsstrategie des mathematischen Modells

| Optimierungsstrategie | |
|-----------------------|--|
| Schritt | Besonderheiten |
| 1 | <ul style="list-style-type: none">- Sechs Unbekannte aus der homogenen Transformationsmatrix T_F^S- Alle weiteren Transformationsmatrizen und Vektoren werden als ausreichend bekannt betrachtet |
| 2 | <ul style="list-style-type: none">- Zwölf Unbekannte aus den homogenen Transformationsmatrizen T_F^S und T_P^A- Alle weiteren Transformationsmatrizen und Vektoren werden als ausreichend bekannt betrachtet |

5 Analyse aktueller Lösungsalgorithmen

In diesem Kapitel werden unterschiedliche Lösungsalgorithmen betrachtet und analysiert. Die Analyse wird hinsichtlich ihrer Vor- und Nachteile in Bezug auf die Lösung von Minimierungsproblemen durchgeführt. Das mathematische Modell aus dem Kapitel 4.4 stellt das Minimierungsproblem dar. Abschließend wird ein geeigneter Algorithmus ausgewählt.

5.1 Potentialanalyse der Lösungsalgorithmen

Aufgrund der großen Auswahl und Varianten der Minimierungsalgorithmen wird im nachstehenden Abschnitt eine Potentialanalyse der wichtigsten Gruppen vorgenommen. Um einen optimalen Minimierungsalgorithmus zu ermitteln, wird die Analyse des mathematischen Modells bezüglich der Randbedingungen und Charakteristiken benötigt (s. Kapitel 4.4). Das betrachtete mathematische Modell stellt ein unterbestimmtes unrestringiertes Optimierungsproblem dar.

Die Einteilung der Minimierungsalgorithmen und deren relevantesten Lösungsalgorithmen wird in der nachfolgenden Tabelle 5-1 angeführt. Dabei wird in die Gruppen des Gradienten-Verfahrens (G-Verfahren), des Trust-Region-Verfahrens (TR-Verfahren), des Verfahrens der kleinsten Quadrate (KQ-Verfahren) und des neuronalen Netzes (NN) unterschieden.

In dem Gradienten-Verfahren wird die Bestimmung des nächsten Schrittes in zwei Abläufe unterteilt. Im ersten Schritt wird eine allgemeine Abstiegsrichtung bestimmt, nachfolgend wird dann die Schrittweite auf einer Linie, in welcher die Bedingung $f(x^k + \alpha h^k) < f^k$ erfüllt ist, bestimmt. [NoWr2006b, PLB2012b, SuYu2006]

Das Trust-Region-Verfahren bestimmt und bewertet die Suchrichtung und die Schrittweite gemeinsam. Dabei wird zuerst der Suchradius bestimmt und im Nachfolgenden die Abstiegsrichtung in dem vertrauenswürdigen Suchradius (s. Kapitel 2.5.4). [NoWr2006b, PLB2012b, SuYu2006]

Bei dem Verfahren der kleinsten Quadrate wird eine Problemstellung durch optimale Parameterwahl gelöst. Dabei wird das Minimum der gesuchten Zielfunktion durch eine quadratische Lösung der Residuen gelöst. Die Lösung des kleinsten quadratischen Problems wird durch die sukzessive Veränderung der gesuchten Parameter umgesetzt. [NoWr2006b, PLB2012b, SuYu2006]

Das Verfahren der neuronalen Netze ermöglicht eine Lösung des mathematischen Modells über ein vorheriges trainiertes Netz mit mehreren Ebenen. Aufgrund der verwendeten Struktur aus Eingangsschicht zur Aufnahme der Inputparameter, der versteckten Ebenen zur Verarbeitung der zugeführten Ergebnisse der Schichten davor und der Ausgangsschicht für die Ausgabe des gesuchten Ergebnisses. [BeKo1994, Rim2017]

Tabelle 5-1: Einteilung der Minimierungsalgorithmen und deren Lösungsalgorithmen der Potentialanalyse

| Verfahren / Gruppen | Lösungsalgorithmen / Besonderheiten |
|----------------------------|---|
| G-Verfahren | <ul style="list-style-type: none"> - Abstiegsrichtungsbestimmung nach: <ul style="list-style-type: none"> ○ Steilster Abstieg $\rightarrow B = I; h = -\nabla f(\bar{x})$ ○ Newton-Richtung $\rightarrow B = \nabla^2 f(\bar{x})$ ○ Modifizierter Newton $\rightarrow B = [\nabla^2 f(\bar{x}) + \lambda I] > 0$ ○ Quasi Newton \rightarrow Approximation von $\nabla^2 f(\bar{x})$ ○ Wolfe-Bedingungen (Armijo- und Krümmungsbedingung) ○ Konjugierte Gradienten |
| TR-Verfahren | <ul style="list-style-type: none"> - Lösung des Trust-Region Unterproblems <ul style="list-style-type: none"> ○ Newton $\rightarrow B = \nabla^2 f(\bar{x})$ ○ Quasi Newton \rightarrow Approximation von $\nabla^2 f(\bar{x})$ ○ Dogleg-Methode - Schrittweite über Chauchy-Punkt Bedingung und Fortschrittsfaktor |
| KQ-Verfahren | <ul style="list-style-type: none"> - Iterative Lösung <ul style="list-style-type: none"> ○ Gauß-Newton ○ Levenberg-Marquardt |
| NN-Verfahren | <ul style="list-style-type: none"> - Eigene Struktur durch hierarchische Ebenen, kein klassischer Lösungsalgorithmus vorhanden |

5.2 Vergleich der Lösungsalgorithmen

Die zuvor durchgeführte Potentialanalyse lässt sich ableiten, dass es vier Verfahrensgruppen mit den entsprechenden Lösungsalgorithmen gibt. Auf dieser Grundlage wird in diesem Abschnitt ein Vergleich hinsichtlich der Vor- und Nachteile der einzelnen Gruppen sowie der Lösungsalgorithmen durchgeführt.

In der detaillierten Betrachtung des Gradienten-Verfahrens ergibt sich eine schnelle und einfache numerische Umsetzung. Dem gegenüber steht ein schlechtes Konvergenzverhalten bei komplexen und umfangreichen Zielfunktionen. Die numerische Umsetzung des Trust-Region-Verfahrens ist im Vergleich zum G-Verfahren aufwändiger und benötigt eine weitere Minimierung eines Unterproblems zum Lösen des Optimierungsproblems. Dieses macht die Berechnung aufwändiger. Dem gegenüber steht das gute Konvergenzverhalten aufgrund des verwendeten Vertrauensbereichs. Die Lösung der kleinsten Quadrate besitzt aufgrund der Minimierung der Residuumsfunktion und der benötigten Optimalitätsbedingung der ersten Ordnung eine numerisch stabile Lösung. Des Weiteren wird keine rechenintensive Bildung der Hessematrix benötigt. Nachteil ist, dass der Startparametersatz bereits dicht am Minimum liegen muss.

In der Betrachtung der ersten drei Gruppen und deren Lösungsalgorithmen ergibt sich eine Schnittmenge von einzelnen Algorithmen. Diese Schnittmenge beruht auf dem grundlegenden Prinzip des Newton-Verfahrens (N-Verfahrens) (s. Kapitel 2.5.2). Ein weiterer Vorteil dieser Lösungsalgorithmen liegt in der klassischen Struktur des Optimierungsalgorithmus sowie der leichten Anpassung und Erweiterung der einzelnen Algorithmen auf weitere Problemstellungen.

Die vierte Verfahrensgruppe basiert auf einer gänzlich abweichenden Struktur. In dieser Struktur wird eine Schicht- / Ebenenstruktur verwendet. Diese Struktur benötigt einen großen Aufwand in der Ermittlung der einzelnen Schichten zwischen der Eingangs- und Ausgabeschicht. Des Weiteren wird zur Ermittlung des Ergebnisses eine aufwändige Trainingsphase des neuronalen Netzes benötigt. Ein weiterer Nachteil ist die aufwändige und kostenintensive Ermittlung ausreichender Inputparameter für die Trainingsphase. Ein Vorteil des Optimierungsalgorithmus liegt in dem Zusammenhang zwischen der Genauigkeitserhöhung und der Anzahl der verwendeten Trainingsinputparameter.

In der nachfolgenden Tabelle 5-2 wird eine Auflistung der Vor- und Nachteile der einzelnen Verfahrensgruppen sowie eine Bewertung bezüglich des in dieser Arbeit verwendeten mathematischen Modells aufgezeigt. Unter der Bewertung der numerischen Umsetzung wird der Aufwand der allgemeinen Umsetzung in einer Programmiersprache verstanden. Die Stabilität bildet das ungewollte numerische Aufschwingen beim Suchen des Minimums sowie die Empfindlichkeit in der Änderung des Startparameters ab. Die Anpassung an neue oder komplexere mathematische Modelle wird in der letzten Spalte der Bewertung abgebildet.

Tabelle 5-2: Bewertung der Lösungsalgorithmen der Potentialanalyse

| Verfahren | Lösungsalgorithmus | Bewertung | | | |
|-------------------|---------------------|----------------------|-------------|--|-----------------------------|
| | | Numerische Umsetzung | Stabilität | Anpassung an weitere mathematische Modelle | Genauigkeit des Ergebnisses |
| Gradienten | Steilster Abstieg | Sehr gut | Schlecht | Gut | Gut |
| | Newton | Sehr gut | Ausreichend | Gut | Gut |
| | Wolfe-Bedingungen | Sehr aufwändig | Gut | Aufwändig | Sehr gut |
| Trust-Region | Newton | Aufwändig | Sehr gut | Aufwändig | Sehr gut |
| | Dogleg | Aufwändig | Sehr gut | Aufwändig | Gut |
| Kleinste Quadrate | Gauß-Newton | Sehr gut | Sehr gut | Sehr gut | Sehr gut |
| | Levenberg-Marquardt | Sehr gut | Sehr gut | Sehr gut | Sehr gut |
| Neuronale Netze | Mehrere Ebenen | Aufwändig | Sehr gut | Aufwändig | Gut |

Bewertung in vier Ebenen: schlecht / sehr aufwändig, aufwändig / ausreichend, gut und sehr gut.

Auf der Grundlage der Bewertung der einzelnen Lösungsalgorithmen für das im Kapitel 4.4 entwickelten mathematischen Modells wird sich für die nachfolgenden Lösungsalgorithmen entschieden. Bei der Entscheidungsfindung der beiden finalen Lösungsalgorithmen wird ein besonderes Augenmerk auf die Anpassung neuer mathematischer Modelle, die Nachvollziehbarkeit des Lösungswegs der einzelnen Iterationen und die Stabilität der numerischen Berechnung gelegt.

Trust-Region Verfahren

Das Trust-Region-Verfahren wird aufgrund der Modifizierung des Newtons mit dem Vertrauensbereich und der damit einhergehenden Steigerung der Genauigkeit, Stabilität und Minimierung der Iterationen bis ins gesuchte Minimum verwendet. Des Weiteren kann eine hohe Güte in den Ergebnissen erzeugt werden.

Gauß Newton-Verfahren

Das Gauß Newton-Verfahren wird aufgrund seiner einfachen Modifizierbarkeit an neue mathematische Modelle sowie seiner sehr guten Interpretation der einzelnen Iterationen verwendet. Zudem lassen sich mit kleinen Anpassungen weiter numerische Stabilisationen und besondere Eigenschaften in der Minimierung umsetzen.

6 Umsetzung und Implementierung

In den nachfolgenden Kapiteln wird die numerische Umsetzung der ausgewählten Algorithmen und der zugehörigen Teilberechnungen erläutert. Im letzten Abschnitt wird die experimentelle Datenaufnahme betrachtet. Die Datenaufnahme gliedert sich in den Versuchsaufbau mit dem entwickelten Kalibrierkörper und der Entwicklung einer geeigneten Messstrategie.

6.1 Integration und Lösung in MATLAB

Die Struktur des entwickelten Algorithmus besteht aus mehreren Unterfunktionen. Aufgrund der gewählten Struktur aus einzelnen Unterfunktionen wird ein übersichtlicheres, nachvollziehbares und änderungsfreundlicheres Skript ermöglicht. Die allgemeine Struktur des Optimierungsskriptes gliedert sich in drei Bereiche. In dem ersten Bereich befindet sich ein Kommentarblock, in welchem der Ersteller und wichtige Informationen zum Optimierungsskript oder zur Unterfunktion enthalten sind. In diesen Informationen wird auf die Grenzen des Skriptes sowie die benötigten Inputparameter eingegangen. In dem zweiten Abschnitt werden die für die Berechnung benötigten globalen Variablen für die verwendeten Funktionen definiert. Des Weiteren werden in diesem Abschnitt alle geöffneten Fenster geschlossen und alle definierten Variablen gelöscht, um einen sicheren und stabilen Ablauf des Optimierungsskriptes zu gewährleisten. Im letzten Abschnitt werden die zum Durchlaufen des Optimierungsalgorithmus benötigten Schleifen und Unterfunktionen definiert. Eine schematische Darstellung der einzelnen Bereiche wird in der nachfolgenden Tabelle 6-1 gezeigt.

Tabelle 6-1: Schematische Struktur des entwickelten Optimierungsskriptes mit den zugehörigen Bereichen

| Bereich „1“ |
|---|
| <pre> %% ***** % Ersteller: Christoph Scholl % Datum: 01.05.2019 % Arbeitsbereich: Masterthese % Inhalt: Schematische Darstellung einer Optimierungsskriptes mit den drei % Bereichen % Besonderheiten: Nur eine schematische Darstellung %% ***** </pre> |
| Bereich „2“ |
| <pre> %% ***** clear all %Löschen aller Daten close all %Schließen aller Fenster clc %Alles säubern format longG %Ausgabeformat % Globale Daten des Optimierungsalgorithmus [P1, P2, P3, XS, XP] = IMPORT_Daten(); MaxIter = 250; mse_old = [0; 0; 0; 0]; tol = 1.e-20; %% ***** </pre> |

Bereich „3“

```
%% Berechnung *****
for k = 1:MaxIter
    % -----
    % Funktion "1"
    % -----

    [X,Y,Z] = Func_1(P1,P2,P3,XS,XP);

    % -----
    % Funktion "2"
    % -----

    [Alpha,Betta,Gamma,err] = Func_2(X,Y,Z);

    % -----
    % Erzeugung des Abbruchkriteriums bei Unterschreitung der Toleranz
    % -----
    if (abs(err) <= tol)
        break
    end
end
end
```

Aufgrund dieser entwickelten Struktur können Anpassungen und Optimierungen in den einzelnen Unterfunktionen vorgenommen werden, ohne den bestehenden Optimierungsalgorithmus zu verändern. Um einen weiteren Optimierungsalgorithmus auf die bestehende Struktur anzuwenden, wird nur der Austausch des dritten Bereiches mit dem neuen Algorithmus benötigt. Aufgrund der zur Ausführung der Funktionen benötigten Inputparameter und der definierten Outputparameter werden nur die benötigten Variablen im Workspace hinterlegt. Diese Eigenschaft ermöglicht die genaue Definition der auszugebenden Variablen und reduziert damit den benötigten Speicherplatz. Die angeführte Struktur für den Lösungsalgorithmus wird in den Unterfunktionen gleichwertig umgesetzt.

6.1.1 Import und Datenstruktur

Um die gemessenen Inputparameter in eine Struktur zu bringen, wird die nachfolgende Funktion aufgestellt. In dieser Funktion werden alle Daten, die als Inputparameter des Minimierungsalgorithmus benötigt werden, strukturiert und mit einer fest definierten Variable versehen.

Die pro Messungen i gewonnenen Roboterposen $p_{1,i}$ und gemessenen Vektoren $x_{s,i}$ werden in der Funktion zu einer Matrix zusammengeführt. In der Matrix $P1$ werden die einzelnen Posen als Spaltenvektor hintereinander strukturiert (Formel (6-1)). Aufgrund der $[6 \times 1]$ Größe der Posen ergibt sich eine Matrixgröße von $[6 \times i]$. Die zugehörigen Vektoren $x_{s,i}$ werden als Spaltenvektoren in einer Matrix XS hintereinander angeordnet (Formel (6-2)). In der Betrachtung der Vektorgröße von $[4 \times 1]$ ergibt sich eine Matrixgröße von $[4 \times i]$.

$$P1 = [p_{1,1} \quad p_{1,2} \quad \dots \quad p_{1,i}] = \begin{bmatrix} x_{1,1} & \dots & x_{1,i} \\ y_{1,1} & \dots & y_{1,i} \\ z_{1,1} & \dots & z_{1,i} \\ A_{1,1} & \dots & A_{1,i} \\ B_{1,1} & \dots & B_{1,i} \\ C_{1,1} & \dots & C_{1,i} \end{bmatrix} \quad (6-1)$$

$$XS = [x_{s,1} \quad x_{s,2} \quad \dots \quad x_{s,i}] = \begin{bmatrix} x_{s,1} & \dots & x_{s,i} \\ y_{s,1} & \dots & y_{s,i} \\ z_{s,1} & \dots & z_{s,i} \\ 1 & \dots & 1 \end{bmatrix} \quad (6-2)$$

Die Startpose für die homogene Transformationsmatrix T_S^F wird in der Variablen $P2$ definiert und besitzt die Größe $[6 \times 1]$. In der Betrachtung der linken Seite der Gleichung (4-1) ergibt sich eine Pose ($P3$) für die homogene Transformationsmatrix T_P^A und ein Vektor X_P' . Die Größe der Pose ist $[6 \times 1]$ und die des Vektors ist $[4 \times 1]$ (s. Kapitel 2.1.2).

Zur Initialisierung der Funktion werden keine zusätzlichen Parameter benötigt. Das entwickelte MATLAB-Skript zum Importieren der Messdaten in den Minimierungsalgorithmus ist im Anhang C1 hinterlegt.

6.1.2 Umrechnung von Posen in homogene Transformationsmatrizen

Im Nachfolgenden wird auf die Umsetzung der im Stand der Technik beschriebenen Umrechnung von Posen in homogene Transformationsmatrizen eingegangen. Das entwickelte Skript ist in einer Funktion geschrieben. Zur Initialisierung der Funktion zur Umrechnung einer Pose in eine homogene Transformationsmatrix wird ein Zeilenvektor / Pose p mit sechs Einträgen benötigt. Diese sechs Inputparameter stellen die klassischen Poseneinträge dar. Die ersten drei Werte stellen die Translation in x-, y- und z-Achse dar. An der vierten Stelle wird der Rotationswinkel A/α um die z-Achse, an der fünften Stelle wird der Rotationswinkel B/β um die y-Achse und an der letzten Stelle wird der Rotationswinkel C/γ um die x-Achse abgebildet. Da die Multiplikation der Rotationsmatrizen nicht kommutativ ist wird die RPY-Definition verwendet (s. Kapitel 2.1.2).

Die allgemeine Umrechnung einer Pose in eine homogene Transformationsmatrix wird durch die Multiplikation der einzelnen Translationsmatrizen und Rotationsmatrizen umgesetzt (Formel(6-3)). Das zur Umsetzung entwickelte MATLAB-Skript zum Umrechnen einer Pose in eine homogene Transformationsmatrix wird im Anhang C2 hinterlegt.

$$H_T = T * R = T_z(t_z) * T_y(t_y) * T_x(t_x) * T_{Rz}(C) * T_{Ry}(B) * T_{Rx}(A) \quad (6-3)$$

$$T_x(t_x) = \begin{bmatrix} E & (t_x \ 0 \ 0)^T \\ 0 & 1 \end{bmatrix} \quad (6-4)$$

$$T_y(t_y) = \begin{bmatrix} E & (0 \ t_y \ 0)^T \\ 0 & 1 \end{bmatrix} \quad (6-5)$$

$$T_z(t_z) = \begin{bmatrix} E & (0 \ 0 \ t_z)^T \\ 0 & 1 \end{bmatrix} \quad (6-6)$$

$$T_{Rx}(C) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos C & -\sin C & 0 \\ 0 & \sin C & \cos C & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6-7)$$

$$T_{Ry}(B) = \begin{bmatrix} \cos B & 0 & \sin B & 0 \\ 0 & 1 & 0 & 0 \\ -\sin B & 0 & \cos B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6-8)$$

$$T_{Rz}(A) = \begin{bmatrix} \cos A & -\sin A & 0 & 0 \\ \sin A & \cos A & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6-9)$$

6.1.3 Berechnung des Residuums

In bestimmten Minimierungsalgorithmen wird zur Einschätzung der Lösung eine Umschreibung des Problems benötigt. Dieses wird in einem Gauß-Newton-Minimierungsalgorithmus durch die Summe der kleinsten Quadrate umgesetzt. Im Nachfolgenden wird die in der Arbeit verwendete Form der nichtlinearen kleinsten Quadrate betrachtet und numerisch umgesetzt. Zu jeder Messung i wird ein Residuum ε_i berechnet. Aufgrund des verwendeten mathematischen Modells ergibt sich ein Vektor der Größe [4x1] (Formel (6-10)). Aus den einzelnen Residuen werden die nichtlinearen kleinsten Quadrate $\bar{\varepsilon}$ gebildet (Formel (6-11)). Als Ausgabevariable wird der Residuumsvektor ε aus allen Residuen mit der Größe [(4*)x1] und der Vektor $\bar{\varepsilon}$ mit der Größe [4x1] definiert.

$$\varepsilon = (T_A^F * T_F^S * x_S) - (T_P^A * x_{P'}) \quad (6-10)$$

$$\bar{\varepsilon} = \frac{1}{2} \sum_{i=1}^n \varepsilon_i^2 \quad (6-11)$$

Zur Initialisierung der Funktion zur Berechnung der Residuen und den nichtlinearen kleinsten Quadraten werden die Parameter $[P1, P2, P3, XS, XP]$ aus der Importfunktion benötigt (s. Kapitel 6.1.1). Das zur Umsetzung entwickelte MATLAB-Skript zur Berechnung des Residuums ist im Anhang C3 hinterlegt.

6.1.4 Berechnung der Jacobi-Matrix

Zur Bestimmung der Schrittweite wird in ausgewählten Minimierungsalgorithmen die Jacobi-Matrix des mathematischen Modells benötigt. In der Definition der Jacobi-Matrix werden die einzelnen partiellen Ableitungen der Zielfunktion benötigt (Formel (6-12) und (6-13)).

$$F = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_i \end{pmatrix} \quad (6-12)$$

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \quad (6-13)$$

Die numerische Umsetzung der Jacobi-Matrix wird mittels der Finite Differenzen Methode (FDM) realisiert. In der numerischen Umsetzung wird die Zielfunktion in eine Differentialgleichung mit Grenzwertbetrachtung überführt (Formel (6-14)). Daraus lassen sich verschiedene Differenz-Methoden ableiten. Im Nachfolgenden wird die Vorwärts-Differenzen Methode detaillierter betrachtet. Zur Umsetzung der Differenzen Methode wird eine Definition einer Schrittweite h benötigt. Aufgrund dieser Schrittweite kann eine partielle Ableitung der Zielfunktion bestimmt werden. Bei der Umsetzung wird der Funktionswert $f(x+h)$ von dem Funktionswert $f(x)$ subtrahiert und durch die Schrittweite h dividiert (Formel (6-15)).

$$\frac{\partial f}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \quad (6-14)$$

$$\frac{\partial f}{\partial x} = \frac{f(x+h) - f(x)}{h} \quad (6-15)$$

Aus jeder gemessenen Roboterposition i ergibt sich eine Jacobi-Matrix J_i mit der Größe $[4 \times 6]$. Für die Ausgabeparameter der Funktion werden die Jacobi-Matrizen untereinander in einem Vektor angeordnet. Der so entstehende Jacobi-Vektor besitzt eine Größe von $[(4 \cdot i) \times 6]$.

Zur Initialisierung der Funktion zur Berechnung der Jacobi-Matrix werden die Parameter $[P1, P2, P3, XS, XP]$ aus der Importfunktion benötigt (s. Kapitel 6.1.1). Das zur Umsetzung entwickelte MATLAB-Skript zur Berechnung der Jacobi-Matrizen ist im Anhang C4 hinterlegt.

6.1.5 Hand-Auge-Kalibrationsalgorithmus mittels Gauß-Newton-Verfahren

Die in dieser Arbeit entwickelte Struktur des Gauß-Newton-Verfahrens basiert auf der allgemeinen Gauß-Newton Struktur (s. Kapitel 2.5.3). Die entwickelte Struktur wird aufgrund der spezifischen Eigenschaft des mathematischen Modells geringfügig modifiziert. Diese Modifikation basiert auf der Eigenschaft der Umrechnung von Posen in homogene Transformationsmatrizen.

Die Anpassung der Struktur basiert auf der Übergabe der Inputparameter. Diese werden als einzelne Posen $(p_{1,i}, p_2^k, p_3,)$ und Vektoren $(x_{S_i}, x_{P'})$ der jeweiligen Messungen (i) übergeben. Die Umrechnung der Posen in äquivalente homogene Transformationsmatrizen wird in der Schleife des Gauß-Newtons durchgeführt. Durch diese Übergabe kann eine Pose mit sechs Unbekannten minimiert werden, da bei jedem weiteren Durchlaufen der Schleife ein neues Gleichungssystem erzeugt wird. Diese Art der Modifizierung ist auf die Minimierung der gesuchten Parameter zurückzuführen. Des Weiteren wird eine Mehrdeutigkeit in einer theoretisch bestimmten finalen homogenen Transformationsmatrix vermieden.

Die Umsetzung des modifizierten Gauß-Newton-Algorithmus ergibt die nachfolgende Struktur zur Minimierung des mathematischen Modells (s. Kapitel 4.4) (s. Gleichung (4-3)). Der umgesetzte Minimierungsalgorithmus mit dem entwickelten Gauß-Newton-Algorithmus ist im Anhang C5 hinterlegt. Die in dem Minimierungsalgorithmus benötigten Unterfunktionen werden in den Kapiteln 6.1.1 bis 6.1.4 erläutert.

Um in einem zweiten Schritt der Optimierung die Pose p_3 zusätzlich zur davor ermittelten Pose p_2 umzusetzen, wird das untenstehende Skript im Allgemeinen um eine weitere Zeile und um ein weiteres Abbruchkriterium erweitert. Diese Erweiterung bewirkt das Optimieren eines mathematischen Modells mit zwei Posen und 12 unbekannt Variablen.

Algorithmus: Modifiziertes Gauß-Newton-Verfahren

Input: $p_{1,i}; p_2; p_3; x_{S_i}; x_{P'}; j_{max}; tol; \epsilon_{old}; i = \text{Anzahl Messungen}$

for $k = 1 : j_{max}$

$$\begin{aligned} \mathbf{x} &= [p_{1,i}, p_2^k, p_3, x_{S_i}, x_{P'}]^T \\ p_2^{k+1} &= p_2^k + \mathbf{d}^k \\ \epsilon &= \frac{1}{2} \sum_{i=1}^{i_{max}} F_i(\mathbf{x}^k)^2 = \frac{1}{2} \sum_{i=1}^{i_{max}} \epsilon_i^2 \\ \mathbf{d}^k &= - \left(J(\mathbf{x}^k)^T J(\mathbf{x}^k) \right)^{-1} J(\mathbf{x}^k) F(\mathbf{x}^k) \end{aligned}$$

if $\|\epsilon - \epsilon_{old}\| \leq tol$
 brake

end

$\epsilon_{old} = \epsilon$

end

6.1.6 Hand-Auge-Kalibrationsalgorithmus mittels Trust-Region-Verfahren

Für die Erstellung des Skriptes für das Trust-Region_Verfahren wird auf die Toolbox der Software MATLAB zurückgegriffen. In der Toolbox befinden sich zwei Abwandlungen des Trust-Region-Verfahrens. Dieses wird in das klassische Trust-Region und das Trust-Region-Dogleg unterteilt.

Zur Minimierung des in der Arbeit entwickelten mathematischen Modells wird das klassische Trust-Region-Verfahren von MATLAB verwendet (s. Kapitel 2.5.4). Dabei wird die zu minimierende Zielfunktion in einer abweichenden Form der zuvor verwendeten Struktur des Gauß-Netzwerks verwendet. Das mathematische Modell wird nicht in Form von Posen $(p_{1,i}, p_2^k, p_3,)$ als Inputparameter definiert, sondern als deren äquivalenten homogenen Transformationsmatrizen (T_A^F, T_P^A, T_F^S) .

Die in der Software MATLAB hinterlegte Struktur zum Minimieren des mathematischen Modells wird in der nachfolgenden Struktur schematisch abgebildet. [PLB2012b, Ste2018] Dabei wird das zu minimierende Problem in einem Subproblem approximiert unter gleichzeitiger Berücksichtigung des Vertrauensbereiches δ . Anhand der berechneten Güte der Approximation wird der Trust-Region-Radius (t^{k+1}) des nächsten Schrittes bestimmt. Dieses Vorgehen wird solange wiederholt, bis das Abbruchkriterium erfüllt ist.

Die Ansteuerung der MATLAB-Optimierungs-Toolbox zur Ausführung des implementierten Trust-Region-Verfahrens zur Minimierung des in dieser Arbeit entwickelten mathematischen Modells ist im Anhang C6 hinterlegt.

Algorithmus: Schematisches Trust-Region-Verfahren / MATLAB

Input: $p_1, i; p_2; p_3; x_{S_i}; x_{P_i}; tol; \epsilon_{old}; i = \text{Anzahl Messungen}; \delta_{trust}^0, k = 0$
 $\check{t}; t^0 \text{ (Radius)}; \eta \in [0, \frac{1}{4})$

while $\|\nabla f(x^k)\| > tol$

$$\min f(x^k + \delta x) = m^k(\delta x) := \frac{1}{2} \delta x^T \nabla^2 f(x^k) \delta x + \nabla f(x^k)^T \delta x + f(x^k)$$

$$r^k = \frac{f(x^k) - f(x^k + \delta^k)}{m^k(0) - m^k(\delta^k)}$$

if $r^k < \frac{1}{4}$

$$t^{k+1} = \frac{1}{4} \|\delta^k\|_2$$

else

if $r^k > \frac{3}{4}$ & $\|\delta^k\|_2 = t^k$

$$t^{k+1} = \min\{2t^k, \check{t}\}$$

else

$$t^{k+1} = t^k$$

end

end

if $r^k > \eta$

$$x^{k+1} = x^k + \delta^k$$

else

$$x^{k+1} = x^k$$

end

$$k = k + 1$$

end

6.2 Theoretische Betrachtung einer definierten Potenzfunktion

Für eine Visualisierung der Ergebnisse des Minimierungsalgorithmus und zur Verifizierung der richtigen Umsetzung wird eine Potenzfunktion eingeführt (Formel (6-16)). Aufgrund der visuellen Darstellbarkeit hängt die Funktion von zwei Variablen (x_1, x_2) ab. Diese Abhängigkeit ermöglicht eine dreidimensionale Abbildung der Funktion.

$$f(x) = \frac{-100}{(x_1 - 1)^2 + (x_2 - 1)^2 + 1} - \frac{200}{(x_1 + 1)^2 + (x_2 + 2)^2 + 1} \quad (6-16)$$

Die Funktion besitzt ein lokales und ein globales Minimum, welches durch einen Sattelpunkt miteinander verbunden sind. Dieser Sattelpunkt liegt niedriger als das normale Niveau der Funktion um die Minima herum (Abbildung 6-1). In der detaillierteren Betrachtung der Funktion ergeben sind drei signifikante Stellen, an denen ein Minimierungsalgorithmus eine Extremstelle erkennen kann. Durch die sich ergebene horizontale Tangente wird in den beiden Minima und dem Sattelpunkt eine Extremstelle ermittelt. Bei einer reinen Verwendung des Gradienten kann aufgrund der fehlenden Information kein Unterschied zwischen dem Sattelpunkt und den Minima erkannt werden.

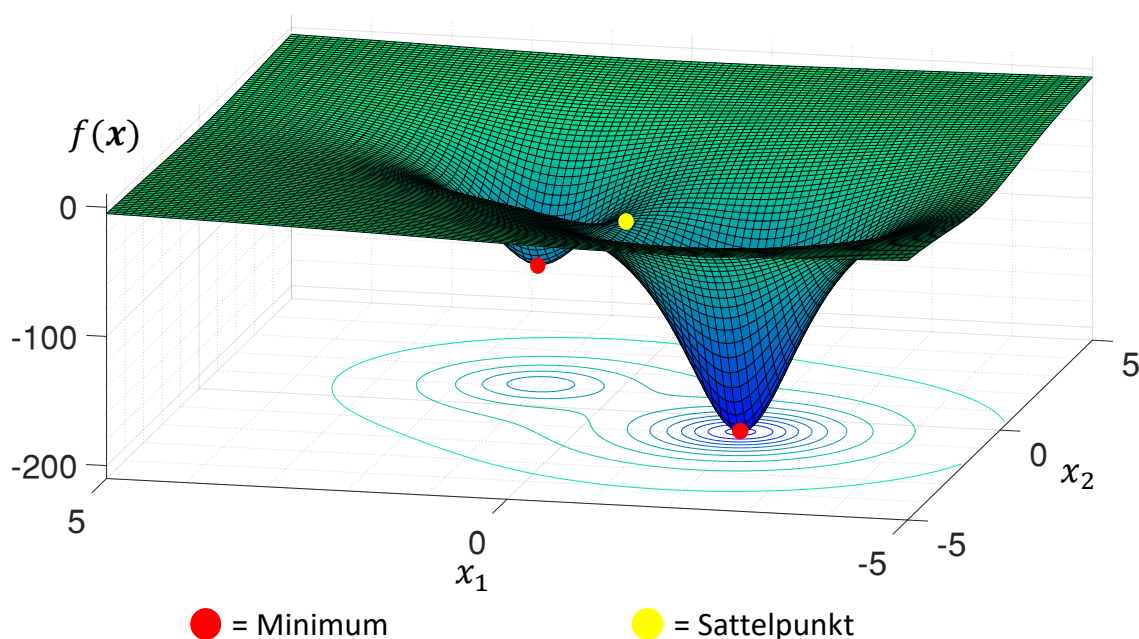


Abbildung 6-1: Dreidimensionale Abbildung der Potenzfunktion mit allen Extremstellen

Mit der Definition der Hessematrix kann eine weitere Information über die Extremstelle gewonnen werden. Dabei werden die Information der zweiten partiellen Ableitung der Funktion in der Hessematrix anhand der Definitheit betrachtet. Unter der Betrachtung wird ein notwendiges und ein hinreichendes Optimalitätskriterium definiert (Kapitel 2.4.3). Aufgrund dieser Kriterien kann eine definierte Aussage über das gefundene Minimum getroffen werden. Der sich aus der Funktion ergebene Gradient wird in der nachfolgenden Formel (6-17) abgebildet. Eine Abbildung der Hessematrix ist im Anhang C7 hinterlegt.

$$\nabla f(\mathbf{x}) = \left[\begin{array}{c} \frac{200(x_1 - 1)}{((x_1 - 1)^2 + (x_2 - 1)^2 + 1)^2} + \frac{400(x_1 + 1)}{((x_1 + 1)^2 + (x_2 + 2)^2 + 1)^2} \\ \frac{200(x_2 - 1)}{((x_1 - 1)^2 + (x_2 - 1)^2 + 1)^2} + \frac{400(x_2 + 2)}{((x_1 + 1)^2 + (x_2 + 2)^2 + 1)^2} \end{array} \right] \quad (6-17)$$

Aus der allgemeinen Definition des Newton-Algorithmus ergibt sich die nachfolgende Struktur zur Minimierung der Zielfunktion (Formel (6-16)). Dabei wird der zuvor definierte Gradient (Formel (6-17)) sowie die Hessematrix verwendet.

Algorithmus: lokales Newton-Verfahren

Input: $x_0; i_{max}$

while $\|f(\mathbf{x}^k)\| > \varepsilon \& < i_{max}$

$$\begin{aligned} \nabla^2 f(\mathbf{x}^k) \mathbf{s}^k &= -\nabla f(\mathbf{x}^k) \\ \mathbf{x}^{k+1} &= \mathbf{x}^k + \mathbf{s}^k \end{aligned}$$

end

Mittels des dargestellten Newton-Algorithmus werden in der nachfolgenden Tabelle die Funktionswerte der Zielfunktion an den einzelnen Extremstellen abgebildet. In der Abbildung 6-2 werden die Höhenlinien der Zielfunktion und die Extremstellen dargestellt. Im Anhang C8 ist das entwickelte Newtonskript hinterlegt. Die zur Ausführung des Newtonskripts benötigten Funktionen des Gradienten und der Hessematrix werden im Anhang C9 abgebildet.

| | Numerische Lösung | | |
|------------------|-------------------|--------|--------|
| | $f(x)$ | x_1 | x_2 |
| Sattelpunkt | -67,484 | 0,105 | -0,205 |
| Lokales Minimum | -114,425 | 0,979 | 0,968 |
| Globales Minimum | -207,159 | -0,995 | -1,992 |

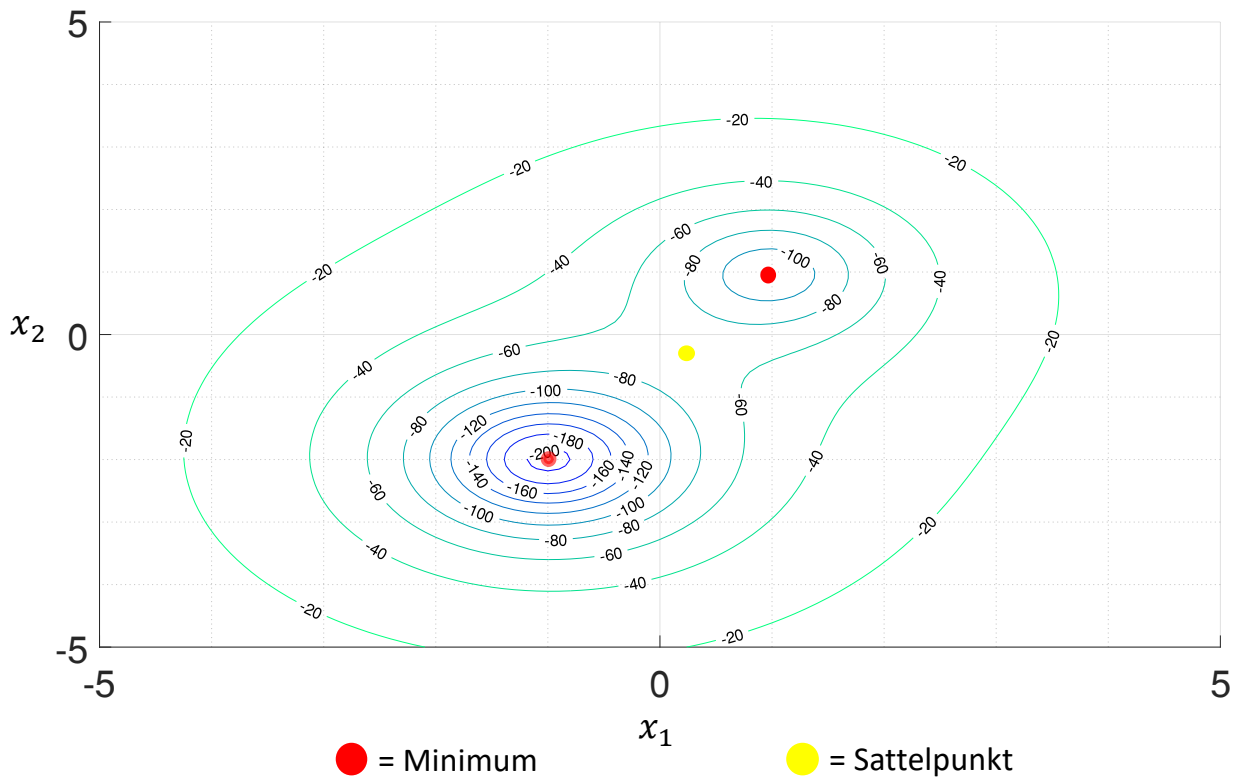


Abbildung 6-2: Höhenlinien der Potenzfunktion mit allen Extremstellen

6.3 Strategie zur Ermittlung von Simulations-Inputargumenten

Um die entwickelten Minimierungsalgorithmen zu testen und die Interpretation des gefundenen Minimus zu bewerten, wird im Nachfolgenden eine Strategie zur Ermittlung geeigneter Inputparameter beschrieben.

Die Überprüfung des entwickelten Minimierungsalgorithmus ist durch die gewonnenen Inputparameter aus dem realen System nicht ausreichend möglich. Aufgrund der Messunsicherheiten und dem Verwenden eines realen Systems wird eine Einschätzung und Interpretation des Minimierungsergebnisses erschwert. Des Weiteren können bei der Verwendung der realen Inputargumente systematische Fehler in der Minimierung nur mühevoll erkannt werden. Diese Problematiken lassen keine genaue Aussage zur korrekten Umsetzung des Minimierungsalgorithmus zu. Um dieses zu verhindern, wird ein System definiert, welches eine nachvollziehbare Manipulation der einzelnen Koordinatensysteme gewährleistet.

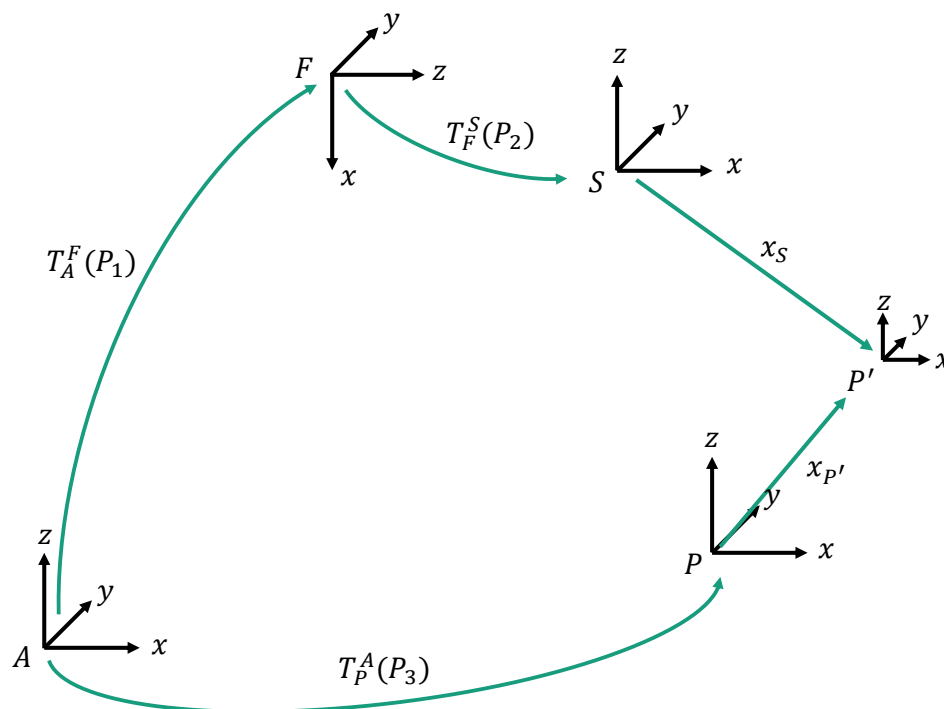


Abbildung 6-3: Ideales System mit einer Drehung um 90° um die y-Achse (A)

Das entwickelte System zur eindeutigen und idealen Manipulation der einzelnen Koordinatensysteme stützt sich auf die Struktur des realen mathematischen Modells. Dabei unterscheidet sich das ideale System nur in der Ausrichtung der einzelnen Koordinatensysteme zu dem realen System (Abbildung 6-3). In dem definierten System sind alle Transformationsmatrizen / Posen und Vektoren bekannt. Die zu minimierende Transformationsmatrix T_F^S ist in allen Manipulationen identisch. Zur Ermittlung der einzelnen Inputargumente wird das Koordinatensystem des Flansches um die Achsen der Aktuatorbasis gedreht (Formel (6-18) bis (6-20)). Bei den Drehungen des Koordinatensystems ergibt sich für jede Drehung eine neue homogene Transformationsmatrix $T_{A,i}^F$ und ein zugehöriger Vektor $x_{S,i}$. Die Transformationsmatrix T_P^A und der Vektor $x_{P'}$ bleiben bei den einzelnen Manipulationen konstant.

Drehung um die z-Achse

$$\alpha = 22,5 * n \quad \text{mit } \{n \in \mathbb{Z} \mid 1 \leq n \leq 4\} \quad (6-18)$$

Drehung um die y-Achse

$$\beta = 22,5 * n \quad \text{mit } \{n \in \mathbb{Z} \mid 0 \leq n \leq 4\} \quad (6-19)$$

Drehung um die x-Achse

$$\gamma = 22,5 * n \quad \text{mit } \{n \in \mathbb{Z} \mid 1 \leq n \leq 4\} \quad (6-20)$$

Die so definierten Drehungen um die einzelnen Koordinatenachsen der Aktuatorbasis ergeben die nachfolgenden Posen, welche als Inputparameter des Minimierungsalgorithmus verwendet werden. Aufgrund der einzelnen Drehungen des Koordinatensystems des Flansches ergeben sich zu jeder definierten Drehung jeweils eine neue Pose $P_{1,i}$ und ein neuer Vektor $x_{s,i}$. Die Transformationsmatrizen / Posen der T_F^S / P_2 und T_P^A / P_3 sowie der Vektor $x_{p'}$ bleiben unverändert. In der nachfolgenden Tabelle 6-2 werden die Posen P_2 , P_3 und der Vektor $x_{p'}$ sowie die vier beispielhaften Rotationen mit den zugehörigen Posen $p_{1,i}$ und Vektoren $x_{s,i}$ dargestellt. Eine detaillierte Auflistung aller Inputparameter für die simulative Überprüfung der erstellten Minimierungsalgorithmen werden im Anhang C10 hinterlegt.

Tabelle 6-2: Posen/Vektoren des idealen Systems für die Simulations-Inputargumente

| Konstante Posen und Vektoren des idealen Systems | |
|---|--|
| $P_2 = \begin{pmatrix} 5 \\ 0 \\ 5 \\ 0 \\ -90 \\ 0 \end{pmatrix}; P_3 = \begin{pmatrix} 25 \\ -5 \\ 5 \\ 0 \\ 0 \\ 0 \end{pmatrix}; x_{p'} = \begin{pmatrix} 5 \\ 5 \\ 5 \\ 1 \end{pmatrix}$ | |
| Drehung um die z-Achse mit $\alpha, \beta, \gamma = 0^\circ$ | Drehung um die z-Achse mit $\alpha = 90^\circ$ |
| $p_{1,0} = \begin{pmatrix} 20 \\ 5 \\ 20 \\ 0 \\ 0 \\ 0 \end{pmatrix}; x_{s,0} = \begin{pmatrix} -15 \\ 0 \\ -5 \\ 1 \end{pmatrix}$ | $p_{1,\alpha 90} = \begin{pmatrix} 20 \\ 5 \\ 20 \\ 90 \\ 0 \\ 0 \end{pmatrix}; x_{s,90} = \begin{pmatrix} -15 \\ -10 \\ 5 \\ 1 \end{pmatrix}$ |
| Drehung um die y-Achse mit $\beta = 90^\circ$ | Drehung um die x-Achse mit $\gamma = 90^\circ$ |
| $p_{1,\beta 90} = \begin{pmatrix} 20 \\ 5 \\ 20 \\ 0 \\ 90 \\ 0 \end{pmatrix}; x_{s,90} = \begin{pmatrix} 5 \\ 0 \\ -5 \\ 1 \end{pmatrix}$ | $p_{1,\gamma 90} = \begin{pmatrix} 20 \\ 5 \\ 20 \\ 0 \\ 0 \\ 90 \end{pmatrix}; x_{s,90} = \begin{pmatrix} -5 \\ -10 \\ -5 \\ 1 \end{pmatrix}$ |

6.4 Experimentelle Datenaufnahme

In den nachfolgenden Unterkapiteln werden die Randbedingungen zur Aufnahme der benötigten Inputargumente aufgezeigt. Aufgrund der gewonnenen Inputargumente kann das mathematische Modell gelöst werden. Die zur Lösung benötigte Strategie, die Durchführung und der Kalibrierkörper werden im Nachfolgenden detailliert erläutert.

6.4.1 Sphärischer Kalibrierkörper

Für die Lösung des mathematischen Modells in der Form der Gleichung (4-1) bis (4-3) wird ein Kalibrierkörper benötigt. Dieser ermöglicht eine Zusammenbringung der beiden Rechenwege. Aus den definierten Dimensionen und der Position des Kalibrierkörpers wird der Vektor $x_{p'}$ festgelegt und der Vektor x_s durch die Sensorinformationen ermittelt. Diese beiden Informationen sind essenziell zur Lösung des mathematischen Modells. In der nachfolgenden Abbildung 6-4 wird der Kalibrierkörper schematisch dargestellt.

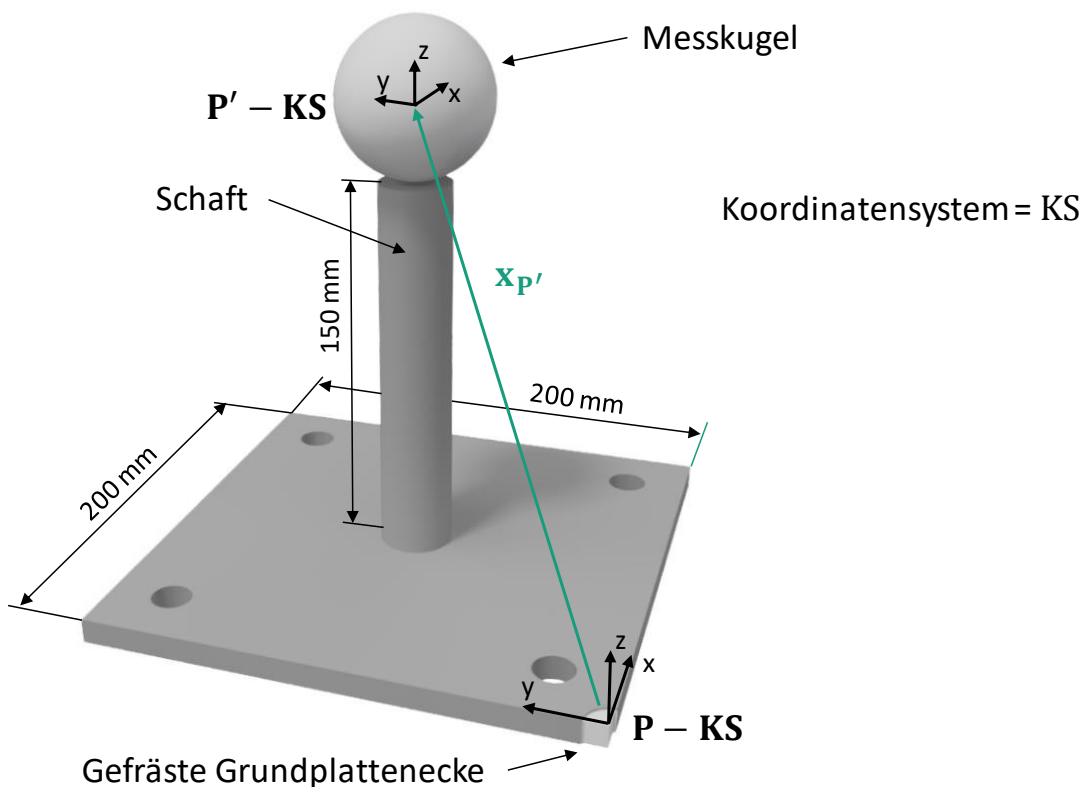


Abbildung 6-4: Schematische Darstellung des Kalibrierkörpers

Der entwickelte Kalibrierkörper besteht aus einer Grundplatte, einem Schaft und einer Messkugel der Firma Saphirwerk AG. Um eine möglichst genaue Positionierung der Messkugel im Raum zu erlangen, wurde eine Grundplatte (200 mm x 200 mm) aus dem Werkstoff S355JR mit einer Dicke von 10 mm verwendet. Eine Ecke der Grundplatte wird auf drei Seiten um einen Millimeter abgefräst, um eine rechtwinklige und ebene Ecke zu erhalten. Für die Positionierung der Grundplatte auf dem 2-Achs-Positionierer wurde das definierte Bohrbild der Bearbeitungsplatte übernommen. Der Schaft ist aus dem gleichen Material wie die

Grundplatte gefertigt und hat einen Durchmesser von 30 mm. Für eine ideale Verbindung zwischen der Grundplatte und der Messkugel sind an beiden Enden M8 Gewinde vorgesehen. Die verwendete Kalibrierkugel mit der Bezeichnung TW DK60.0 GE_M8 besitzt einen Durchmesser von 59,94507 mm. Das Messzertifikat der Kalibration der Messkugel ist im Anhang C11 hinterlegt.

Die Verbindung zwischen der Grundplatte und dem Schaft wird durch ein M8 Innengewinde in der Grundplatte und dem M8 Gewindezapfen umgesetzt. Um ein Lösen oder Verstellen der Verbindung zu vermeiden, wird eine Kontermutter am durchgeschraubten Ende des M8 Gewindezapfens montiert. Am oberen Ende des Schaftes wird die Messkugel auf das M8 Gewinde geschraubt. Eine feste Montage auf der Demmeler-Bearbeitungsplatte, welche sich auf dem 2-Achs-Positionierer befindet, wird mithilfe der von Demmeltersystem vorgesehenen Anschlag / Absteckbolzen und des übernommenen Lochbildes realisiert (Abbildung 6-5).

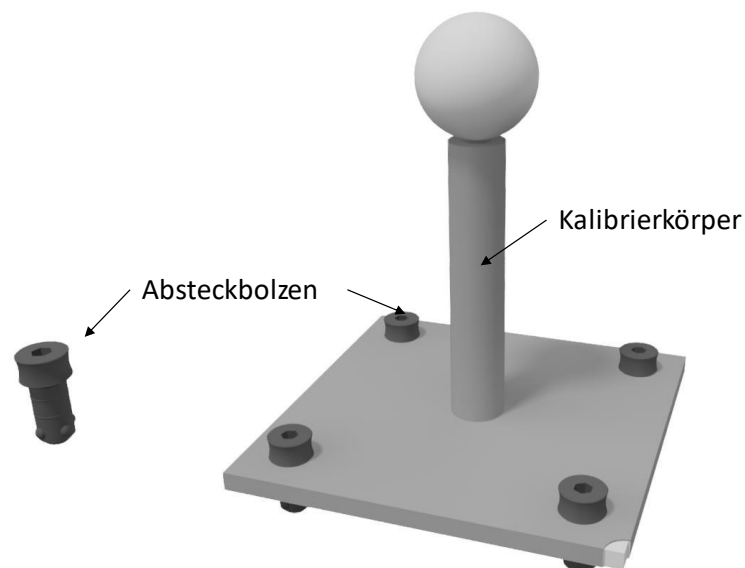


Abbildung 6-5: Befestigung des Kalibrierkörpers auf dem 2-Achs-Positionierer

Aufgrund von Fertigungs- und Montagetoleranzen wird eine Vermessung des montierten Kalibrierkörpers benötigt. Durch die Vermessung des Kalibrierkörpers kann eine ausreichend genaue Lageposition der Messkugel zur Grundplatte ermittelt werden. Die Vermessung des Kalibrierkörpers wird auf einer 3D-Koordinatenmessmaschine X0 107 3D der Firma Wenzel durchgeführt. Aufgrund der verwendeten Messmaschine kann eine Genauigkeit von bis zu $1,7 \mu\text{m}$ erreicht werden (s. Anhang C12). Um die Lage der Messkugel im Raum zu bestimmen, wird eine taktile Messung vorgenommen. Zur Bestimmung des Vektors $x_{p'}$ wird die gefräste Ecke an der Grundplatte taktile vermessen. Durch die so definierten drei Ebenen wird durch deren Schnittpunkt die Lage der Grundplattenecke bestimmt. Anhand der gewonnenen Lagepositionen der Grundplattenecke und dem Mittelpunkt der Messkugel im Raum wird der Vektor $x_{p'}$ definiert. Der Wert des Vektors $x_{p'}$ wird in der nachfolgenden Tabelle 6-3 dargestellt. Des Weiteren werden die theoretischen Werte mit den realen Messergebnissen (s. Anhang C13) und deren Differenz abgebildet. Die angegebenen Winkel und Längen beziehen sich auf das Grundplatten-Koordinatensystem. (Abbildung 6-4)

Tabelle 6-3: Vergleich der theoretischen und realen Kalibrierkörperabmessungen und Positionen der KS

| | Lage Messkugel-KS $(x_{p'} \ y_{p'} \ z_{p'})^T$ | Vektorbetrag [mm] $\ x_{p'}\ $ | Winkel zum Grundplatten-KS [°] | Durchmesser der Kalibrierkugel [mm] |
|---------------|--|-----------------------------------|---|---|
| CAD-Modell | $\begin{pmatrix} 100,00 \\ 100,00 \\ 181,00 \end{pmatrix}$ | 229,6976 | $\begin{pmatrix} 64,1921 \\ 64,1921 \\ 38,0017 \end{pmatrix}$ | 60,00 |
| Reales-Modell | $\begin{pmatrix} 98,5415 \\ 99,9165 \\ 181,6006 \end{pmatrix}$ | 229,5049 | $\begin{pmatrix} 64,5727 \\ 64,1920 \\ 37,6956 \end{pmatrix}$ | 60,5541 |
| Differenz | $\begin{pmatrix} 1,4585 \\ 0,0835 \\ -0,6006 \end{pmatrix}$ | 0,1927 | $\begin{pmatrix} -0,3806 \\ 0,0010 \\ 0,3062 \end{pmatrix}$ | 0,5541 |

6.4.2 Vermessungsstrategie

Im Nachfolgenden wird die in dieser Arbeit entwickelte Vermessungsstrategie erläutert. Aufgrund dieser Strategie werden unterschiedliche und unabhängige Inputargumente erzeugt. Die Qualität der Inputargumente ist zum Lösen des mathematischen Modells ausschlaggebend.

Die zur Lösung des mathematischen Modells benötigten Inputparameter werden durch das Vermessen der Kalibrierkugel mittels des Triangulationssensors erzeugt. Um das mathematische Modell zu lösen, werden mehrere Messungen benötigt (s. Kapitel 4.4.2). Um in den erzeugten Inputparametern keine doppelten Informationen einzubringen und damit Singularitäten im Lösungsalgorithmus zu erzeugen, wird eine Vermessungsstrategie mit möglichst unterschiedlichen Achsstellungen des Roboters entwickelt.

Um eine visuelle Darstellung der Vermessungsstrategie zu erhalten, wird die Ausrichtung des Koordinatensystems der Aktuatorbasis in die Kalibrierkugel übertragen. Die Beschreibung der Vermessungsstrategie bezieht sich auf das Koordinatensystem der Kalibrierkugel. Aufgrund der übertragenen Orientierung des Koordinatensystems in die Kalibrierkugel und der Lage der beiden Systemkomponenten zueinander, wird eine Übertragbarkeit auf andere Robotermodelle und Anordnungen ermöglicht. Der Blickwinkel zur Visualisierung der Vermessungsstrategie wird vom Roboter zur Kalibrierkugel definiert.

Aufgrund der maximalen Verstellung der einzelnen Roboterachsen und der Mindestanzahl an Messungen wird der Sensor / die Messlinien um jeweils $22,5^\circ$ gedreht. Diese Drehungen werden um die x - und y -Achse sowie um die x^* - und y^* -Achse des um α gedrehten Koordinatensystems ausgeführt. Mittels des positiven Drehsinns wird ein Definitionsbereich der Drehungen um die x, y, x^* und y^* - Achsen von -90 bis $+90^\circ$ festgelegt. Die Drehung um die z -Koordinatenachse wird einmal um den Winkel $\alpha = 45^\circ$ durchgeführt, wodurch dann das gedrehte Koordinatensystem erzeugt wird (x^* - und y^*) (Abbildung 6-6). Aus der Definition ergeben sich pro Achse neun unabhängige Messungen. Im Gesamten betrachtet kann mittels dieser Definition eine maximale Anzahl von 36 unabhängigen Messungen mit maximalen Verstellwinkeln der einzelnen Roboterachsen erzeugt werden (s. Definition (6-21) bis (6-25)). Jede Messung ergibt drei neue Inputargumente (x, y, z -Koordinate des Mittelpunktes der Messkugel) für den Lösungsalgorithmus.

$$\alpha = 45^\circ \tag{6-21}$$

$$\beta = 22,5 * n \quad \text{mit } \{n \in \mathbb{Z} \mid -4 \leq n \leq 4\} \tag{6-22}$$

$$\beta^* = 22,5 * n \quad \text{mit } \{n \in \mathbb{Z} \mid -4 \leq n \leq 4\} \tag{6-23}$$

$$\gamma = 22,5 * n \quad \text{mit } \{n \in \mathbb{Z} \mid -4 \leq n \leq 4\} \tag{6-24}$$

$$\gamma^* = 22,5 * n \quad \text{mit } \{n \in \mathbb{Z} \mid -4 \leq n \leq 4\} \tag{6-25}$$

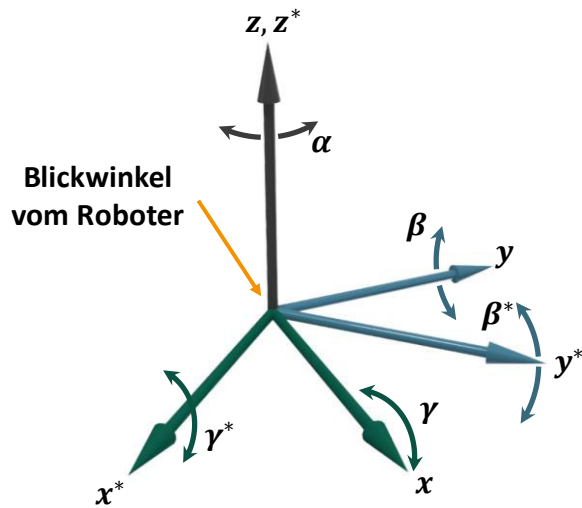


Abbildung 6-6: Definierte Messwinkel und Koordinatensysteme für die Vermessungsstrategie

In der nachfolgenden Abbildung 6-7 wird eine schematische Darstellung einer Messung, die sich um die x -Achse mit dem Winkel $\gamma = 45^\circ$ gedreht ist, aufgezeigt.

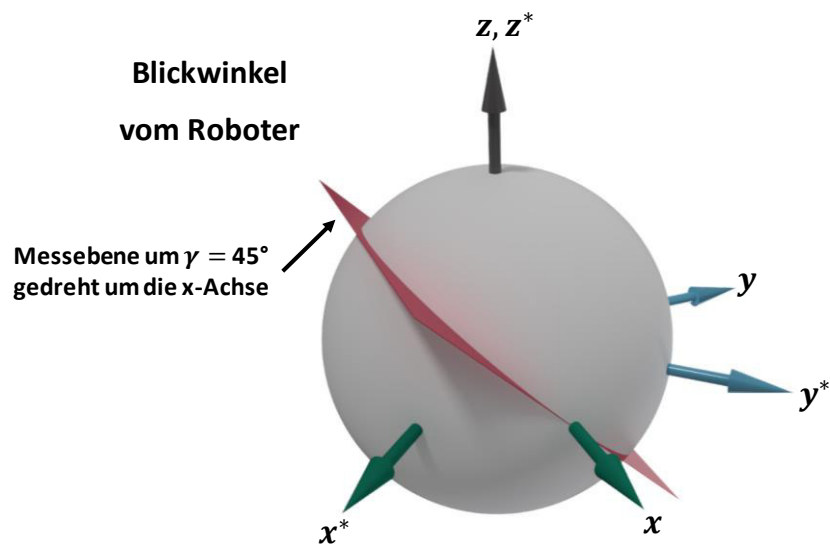


Abbildung 6-7: Schematische Darstellung einer Messung, die um 45° um die x -Achse gedreht ist

6.4.3 Aufnahme der benötigten Inputargumente

Die Aufnahme der einzelnen Messungen aus der zuvor erläuterten Vermessungsstrategie (s. Kapitel 6.4.2) wird anhand des physikalischen Modells (s. Kapitel 4.2) durchgeführt. Im nachfolgenden Kapitel wird auf die Erzeugung einzelner Messwerte und Transformationsmatrizen eingegangen. Die so gewonnenen Daten aus dem Kalibrierkörper gelten nur für den einen Messablauf, nach jedem erneuten Zusammenbau und / oder Montage auf dem 2-Achs-Positionierer, müssen die Vektoren und Posen bezüglich des Kalibrierkörpers neu ermittelt werden.

Transformationsmatrix T_P^A

Der Kalibrierkörper (s. Kapitel 6.4.1) wird auf dem 2-Achs-Positionierer mittels der Absteckbolzen montiert. Nach der Montage des Kalibrierkörpers auf dem 2-Achs-Positionierer wird unter Zuhilfenahme des Roboters die Position der bearbeiteten Kalibriergrundplattenecke ermittelt.

Die Ermittlung der Grundplattenecke kann unter Berücksichtigung des bekannten und im Roboter hinterlegten kalibrierten TCP des Bearbeitungskopfes durchgeführt werden. Der verwendete Bearbeitungskopf besitzt den TCP 17 mm vor dem Austrittsbereich des Laserstrahls. Durch das reine translatorische Verfahren des Bearbeitungskopfes kann die bearbeitete Grundplattenecke mit dem TCP angefahren werden. Aus der Robotersteuerung lässt sich dann indirekt die homogene Transformationsmatrix in der Form der Pose herauslesen. Mittels des in Kapitel 6.1.2 entwickelten MATLAB-Skriptes zur Umrechnung von Posen in homogene Transformationsmatrizen kann eine ausreichend genaue Bestimmung der Transformationsmatrix T_P^A ermittelt werden. In der nachstehenden Abbildung 6-8 wird die schematische Ermittlung der Überlagerung des TCP zur bearbeiteten Kalibriergrundplattenecke aufgezeigt.

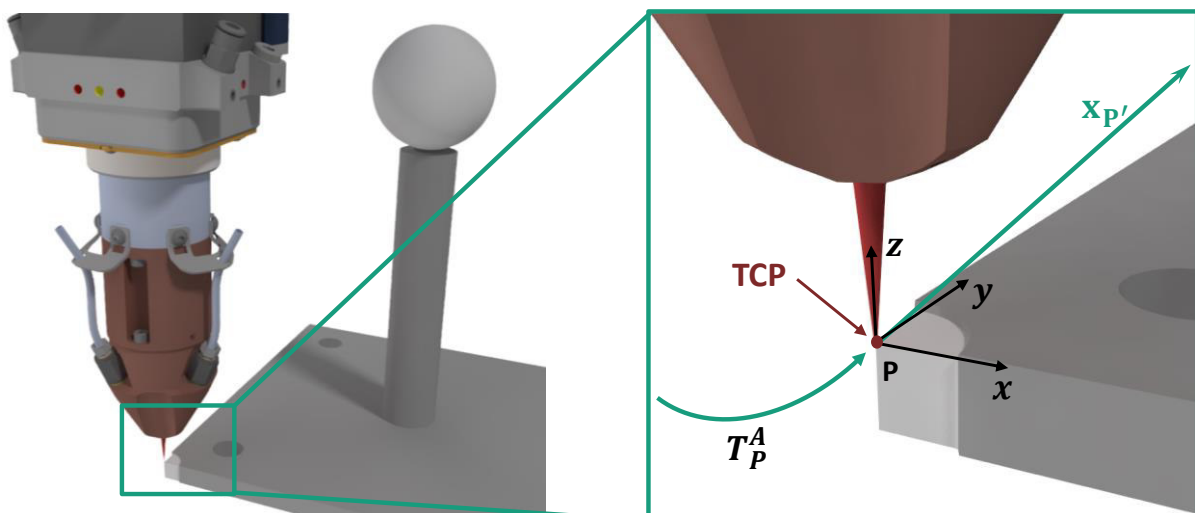


Abbildung 6-8: Schematische Darstellung der Positionsermittlung des Kalibrierkörpers und der Matrix T_P^A

In der nachfolgenden Tabelle 6-4 wird die ermittelte Pose P_{3p} und die sich daraus ergebene homogene Transformationsmatrix T_P^A dargestellt.

Tabelle 6-4: Werte des Inputparameters T_p^A ; Pose und homogene Transformationsmatrix

| Pose p_{3p} vom der Kalibrierkörpergrundplattenecke | |
|---|--|
| $p_{3p} = \begin{pmatrix} x_p \\ y_p \\ z_p \\ A_p \\ B_p \\ C_p \end{pmatrix} = \begin{pmatrix} 1430,90 \\ -698,77 \\ 937,43 \\ -2,78 \\ -0,02 \\ -0,15 \end{pmatrix}$ | |
| Homogene Transformationsmatrix T_p^A | |
| $T_p^A = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0,9988 & 0,0485 & -0,0002 & 1430,9000 \\ -0,0485 & 0,9988 & 0,0026 & -698,7700 \\ 0,0003 & -0,0026 & 0,9999 & 937,4300 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ | |

Vektor $x_{p'}$

Der Vektor $x_{p'}$ wird durch das Vermessen des montierten Kalibrierkörpers in einer 3D-Koordinatenmessmaschine X0 107 3D der Firma Wenzel ermittelt. Die Dimension des Kalibrierkörpers und die Vermessung wird im Kapitel 6.4.1 detailliert erläutert. In der nachfolgenden Tabelle 6-5 werden die Werte des Vektors $x_{p'}$ abgebildet.

Tabelle 6-5: Werte des Inputparameters $x_{p'}$; Vektor

| Bezeichnung | Werte [mm] |
|---|--|
| $\begin{pmatrix} x_{p'x} \\ x_{p'y} \\ x_{p'z} \end{pmatrix}$ | $\begin{pmatrix} 98,5415 \\ 99,9165 \\ 181,6006 \end{pmatrix}$ |
| $ x_{p'} $ | 229,5049 |

Transformationsmatrix T_A^F

Die Bestimmung der homogenen Transformationsmatrix T_A^F wird in jeder definierten Roboterstellung der Vermessungsstrategie aus der Robotersteuerung ausgelesen (s. Kapitel 6.4.2). Die ausgelesene Stellung des Flansch-Koordinatensystems wird als Pose ausgegeben. Diese Posen werden mit dem entwickelten MATLAB-Skript aus dem Kapitel 6.1.2 in eine äquivalente homogene Transformationsmatrix umgerechnet. Im Nachfolgenden werden vier beispielhafte Posen und deren Transformationsmatrizen bei einer positiven Rotation von 45° um die x-, y-,

x^* und y^* -Achse abgebildet. Eine ausführliche Auflistung aller Posen und Transformationsmatrizen erfolgt im Anhang C14.

Tabelle 6-6: Inputargumente der homogenen Transformationsmatrix T_A^F um 45° gedreht

| Pose p_{1x45} und homogene Transformationsmatrix T_{Ax45}^F | |
|---|--|
| $p_{1x45} = \begin{pmatrix} x_{x45} \\ y_{x45} \\ z_{x45} \\ A_{x45} \\ B_{x45} \\ C_{x45} \end{pmatrix} = \begin{pmatrix} 1379,64 \\ -380,04 \\ 1483,81 \\ -90,17 \\ 45,03 \\ -90,18 \end{pmatrix}; T_{Ax45}^F = \begin{pmatrix} -0.0020 & -0.0010 & 0.9999 & 1379.64 \\ -0.7067 & 0.7074 & -0.0007 & -380.04 \\ -0.7074 & -0.7067 & -0.0022 & 1483.81 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ | |
| Pose P_{1y45} und homogene Transformationsmatrix T_{Ay45}^F | |
| $p_{1y45} = \begin{pmatrix} x_{y45} \\ y_{y45} \\ z_{y45} \\ A_{y45} \\ B_{y45} \\ C_{y45} \end{pmatrix} = \begin{pmatrix} 1171.58 \\ -471.20 \\ 1367.76 \\ 0.14 \\ 45.06 \\ 90.15 \end{pmatrix}; T_{Ay45}^F = \begin{pmatrix} 0.7063 & 0.7078 & 0.0005 & 1171.58 \\ 0.0017 & -0.0008 & -0.9999 & -471.20 \\ -0.7078 & 0.7063 & -0.0018 & 1367.76 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ | |
| Pose P_{1x^*45} und homogene Transformationsmatrix $T_{Ax^*45}^F$ | |
| $p_{1x^*45} = \begin{pmatrix} x_{x^*45} \\ y_{x^*45} \\ z_{x^*45} \\ A_{x^*45} \\ B_{x^*45} \\ C_{x^*45} \end{pmatrix} = \begin{pmatrix} 1591.03 \\ -341.92 \\ 1493.79 \\ -135.13 \\ 44.99 \\ -90.20 \end{pmatrix}; T_{Ax^*45}^F = \begin{pmatrix} -0.5013 & 0.4985 & 0.7072 & 1591.03 \\ -0.4988 & 0.5012 & -0.7069 & -341.92 \\ -0.7069 & -0.7072 & -0.0025 & 1493.79 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ | |
| Pose P_{1y^*45} und homogene Transformationsmatrix $T_{Ay^*45}^F$ | |
| $p_{1y^*45} = \begin{pmatrix} x_{y^*45} \\ y_{y^*45} \\ z_{y^*45} \\ A_{y^*45} \\ B_{y^*45} \\ C_{y^*45} \end{pmatrix} = \begin{pmatrix} 1284.81 \\ -532.63 \\ 1494.81 \\ -45.10 \\ 45.06 \\ -90.12 \end{pmatrix}; T_{Ay^*45}^F = \begin{pmatrix} 0.4986 & -0.5012 & 0.7072 & 1284.81 \\ -0.5003 & 0.4997 & 0.7070 & -532.63 \\ -0.7078 & -0.7063 & -0.0016 & 1494.81 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ | |

Transformationsmatrix T_F^S

Die homogene Transformationsmatrix T_F^S wird als Startparameter der gesuchten homogenen Transformationsmatrix eingesetzt. Zur Bestimmung der Pose werden die CAD-Dateien herangezogen. Die Orientierung der Koordinatensysteme am Flansch und am Sensor werden den Dokumentationen entnommen [KUR2005, MIC2019].

In der nachfolgenden Abbildung 6-9 wird die Anordnung der Koordinatensysteme schematisch dargestellt. Das Sensor-Koordinatensystem liegt um den Winkel $\beta = 90^\circ$ um die y-Achse gedreht und um den Winkel $\alpha = 180^\circ$ um die z-Achse gedreht zum Flansch-Koordinatensystem.

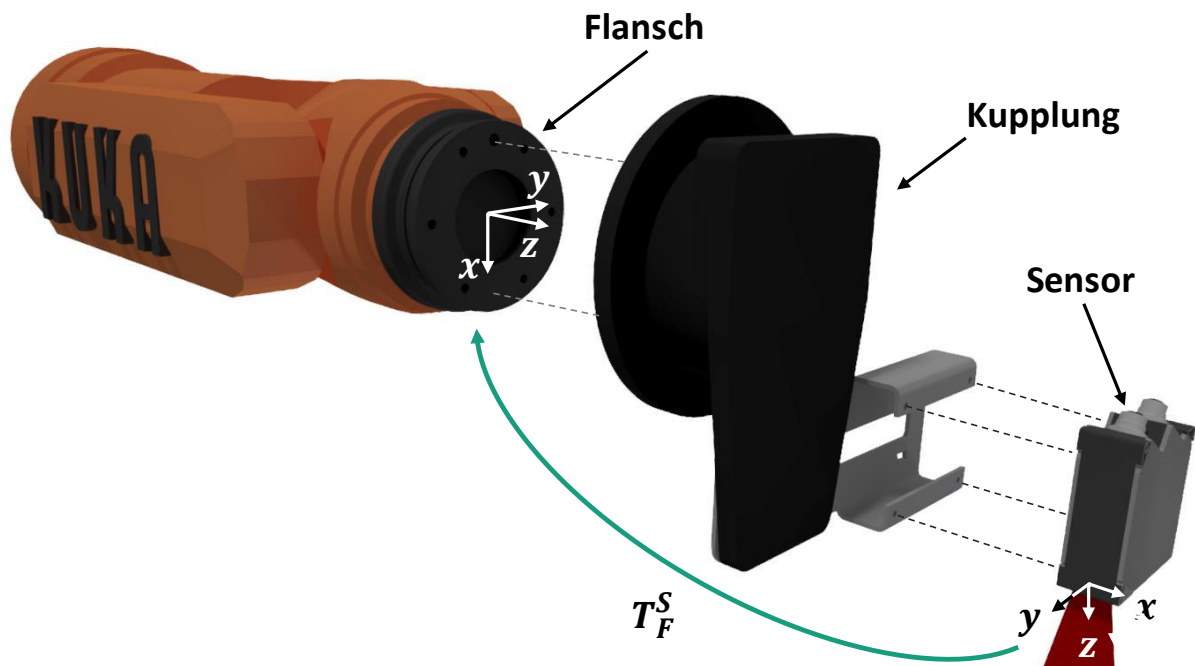


Abbildung 6-9: Koordinatensystem Flansch und Sensor zur Identifizierung des Startparameters T_F^S

In der nachfolgenden Tabelle 6-7 wird die aus den CAD Daten ausgemessene Pose $p_{2_{SF}}$ und die dazugehörige homogene Transformationsmatrix angeführt. Diese Pose wird als Startparameter der Optimierung verwendet.

Tabelle 6-7: Start-Pose $p_{2_{SF}}$ und homogene Transformationsmatrix T_F^S

| Start-Pose $p_{2_{SF}}$ | |
|---|--|
| $p_{2_{SF}} = \begin{pmatrix} x_{2_{SF}} \\ y_{2_{SF}} \\ z_{2_{SF}} \\ A_{2_{SF}} \\ B_{2_{SF}} \\ C_{2_{SF}} \end{pmatrix} = \begin{pmatrix} 156,99 \\ 87,55 \\ 115 \\ 180 \\ -90 \\ 0 \end{pmatrix}$ | |
| homogene Transformationsmatrix T_F^S | |
| $T_F^S = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 156,99 \\ 0 & -1 & 0 & 87,55 \\ 1 & 0 & 0 & 115,00 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ | |

Vektor x_S

Die Bestimmung des Vektors x_S wird mittels des Triangulationssensors vorgenommen. Aufgrund der Vermessungsstrategie der Kalibrierkugel ergibt sich zu jeder Pose des Flansches ein gemessener Vektor x_{S_i} (s. Kapitel 6.4.2). Aufgrund des verwendeten Laser-Linien-Triangulationssensors ergibt sich pro Messung ein Datensatz mit x- und z-Koordinaten der einzelnen Messpunkte auf der Laserlinie. Eine weitere Information über die y-Koordinate der einzelnen Messpunkte kann nicht direkt aus den Messdaten gewonnen werden.

In der nachfolgenden Abbildung 6-10 wird eine schematische Darstellung einer Messung der Kalibrierkugel aufgezeigt. Aufgrund der nicht eindeutigen Lage des Sensors am Bearbeitungskopf kann keine exakte Messung durch den Mittelpunkt der Kalibrierkugel gewährleistet werden.

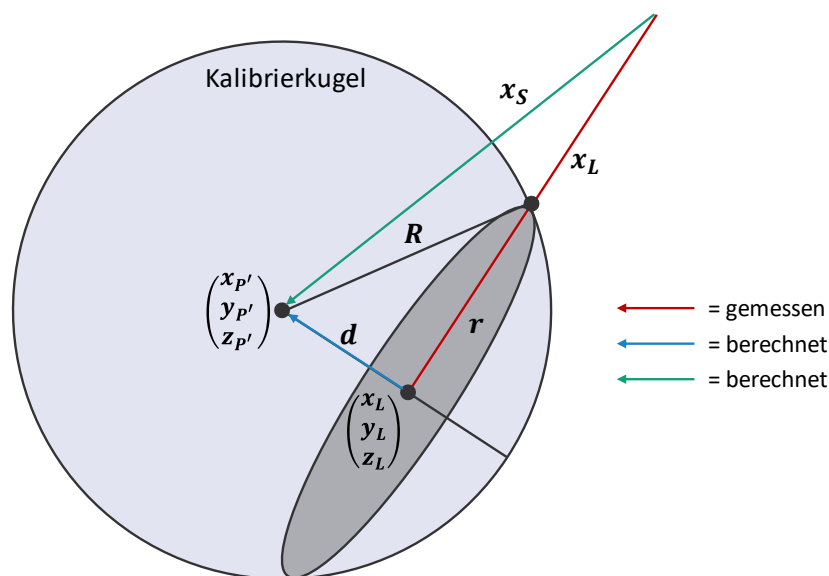


Abbildung 6-10: Schematische Darstellung der Bestimmung des Vektors x_S

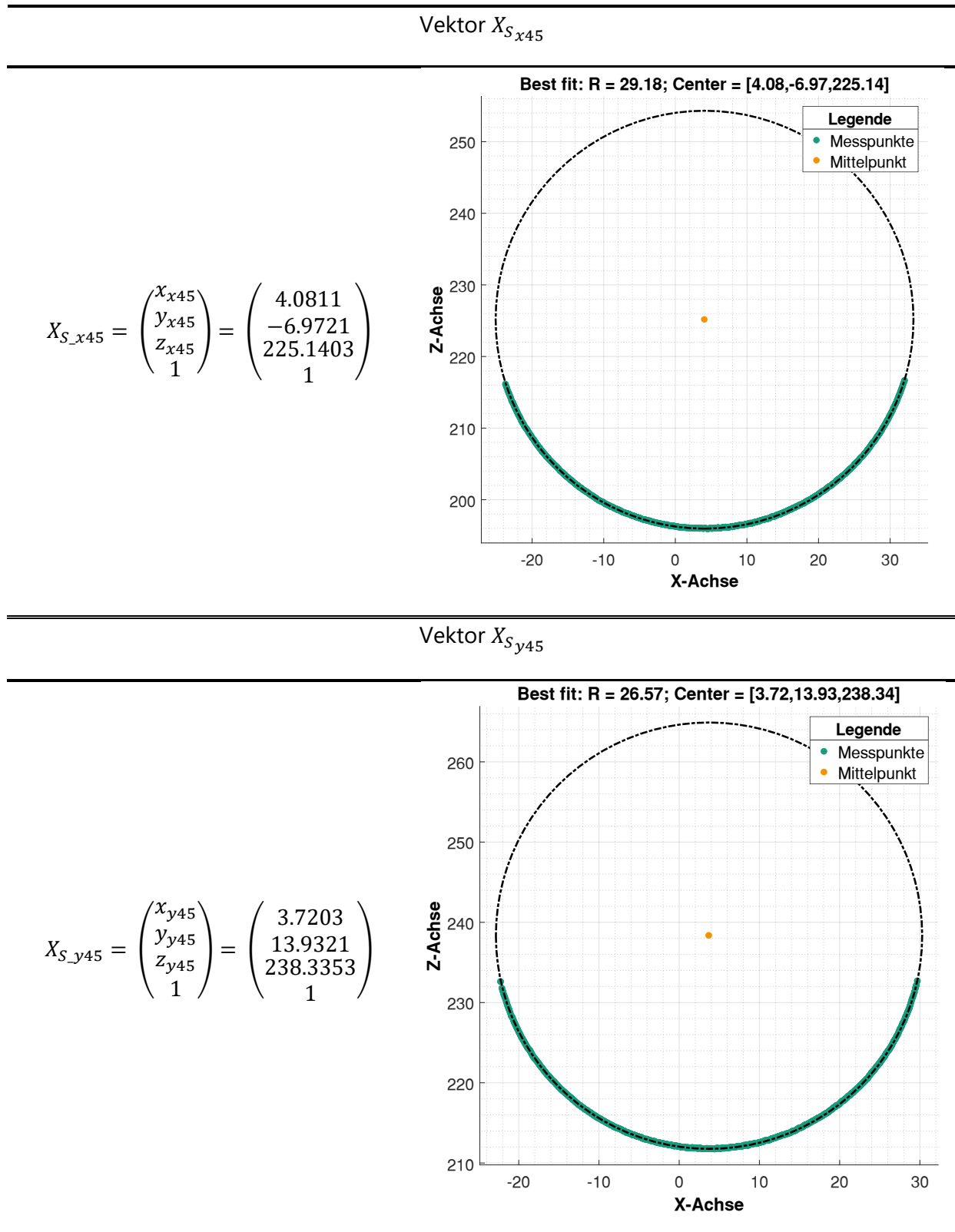
Zur Ermittlung der fehlenden Vektorkomponente wird die nachfolgende Gleichung (6-26) herangezogen. Um den Mittelpunkt der Kalibrierkugel in Sensorkoordinaten zu berechnen, wird aus den Messpunkten des Sensors ein Kreis interpoliert. Dieser Kreis hat einen Radius r , der von der Lage der Messlinie vom Mittelpunkt der Messkugel abhängig ist. Über diese Beziehung kann die nachfolgende Gleichung (6-26) ermittelt werden. Aufgrund der rechtwinkligen Abhängigkeit des Abstandsvektors x_L zum Mittelpunkt der Messkugel zur berechneten Kreisoberfläche kann mittels des Satzes des Pythagoras (Gleichung (6-27)) der Wert d bestimmt werden. [GaTa2011]

$$X_S = \begin{pmatrix} x_{P'} \\ y_{P'} \\ z_{P'} \end{pmatrix} = \begin{pmatrix} x_L \\ y_L \\ z_L \end{pmatrix} + \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} * d \quad (6-26)$$

$$d = \sqrt{(R^2 - r^2)} \quad (6-27)$$

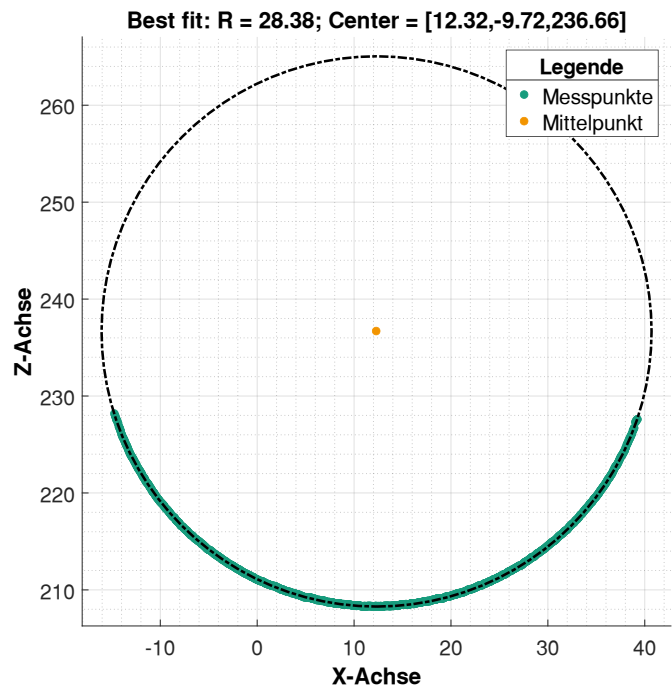
Das entwickelte Skript zur numerischen Umsetzung der Mittelpunktberechnung und der Bestimmung des Vektors x_s wird im Anhang C15 und C16 hinterlegt. In der nachfolgenden Tabelle 6-8 werden vier beispielhafte Vektoren x_s bei einer positiven Rotation von 45° um die x -, y -, x^* und y^* -Achse abgebildet. Eine ausführliche Auflistung aller Vektoren wird im ANHANG C17 gezeigt.

Tabelle 6-8: Inputargumente des Vektors x_s um jeweils 45° gedreht

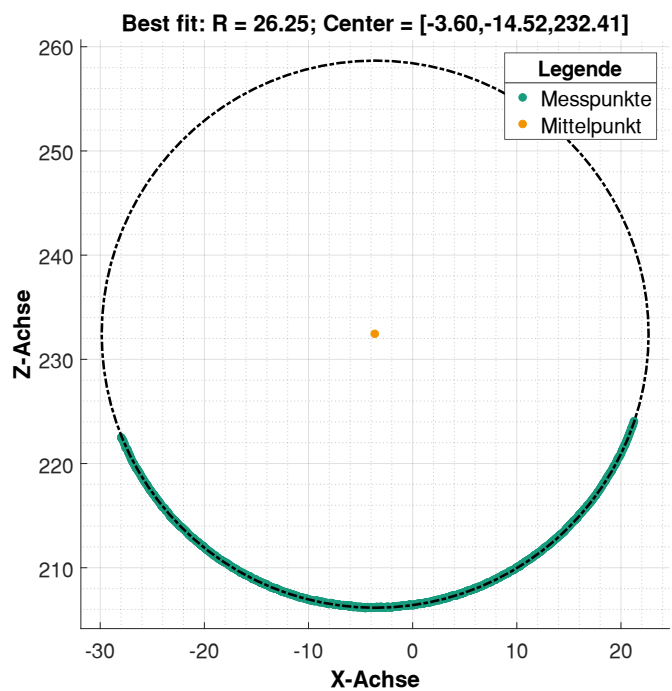


Vektor $X_{S_x^{*45}}$

$$X_{S_x^{*45}} = \begin{pmatrix} x_{x^{*45}} \\ y_{x^{*45}} \\ z_{x^{*45}} \\ 1 \end{pmatrix} = \begin{pmatrix} 12.3202 \\ -9.7175 \\ 236.6635 \\ 1 \end{pmatrix}$$

Vektor $X_{S_y^{*45}}$

$$X_{S_y^{*45}} = \begin{pmatrix} x_{y^{*45}} \\ y_{y^{*45}} \\ z_{y^{*45}} \\ 1 \end{pmatrix} = \begin{pmatrix} -3.6036 \\ -14.5156 \\ 232.4136 \\ 1 \end{pmatrix}$$



7 Ergebnisse und Diskussion

Im folgenden Kapitel erfolgt die Auswertung der Minimierungsalgorithmen hinsichtlich ihrer Stabilität und des zeitlichen Rechenaufwands. Am Ende des Kapitels wird eine Analyse und Interpretation der gewonnenen Ergebnisse durchgeführt.

7.1 Gegenüberstellung der Minimierungsalgorithmen anhand theoretischer Modelle

Die numerische Umsetzung der ausgewählten Lösungsalgorithmen in der Software MATALB muss hinsichtlich eines fehlerfreien Ablaufes überprüft werden. Diese Überprüfung wird anhand von zwei bekannten theoretischen Modellen durchgeführt. Im ersten Modell wird eine bekannte Potenzfunktion (s. Kapitel 6.2) mit zwei Variablen betrachtet. Das zweite Modell besitzt die gleiche mathematische Form wie das reale mathematische Modell. Der Unterschied zum realen mathematischen Modell liegt in den Lagen und Orientierungen der einzelnen Koordinatensysteme zueinander. In beiden Modellen sind die Lösungen des gesuchten Minimums bekannt.

Aufgrund der bekannten Minima kann eine Aussage über die Qualität und die Genauigkeit der Minimierungsalgorithmen getroffen werden. Des Weiteren können durch die Erprobung der Minimierungsalgorithmen an bekannten Modellen systematische Fehler in der numerischen Umsetzung aufgezeigt werden. In den nachfolgenden Kapiteln 7.1.1 und 7.1.2 wird die Auswertung der Minimierungsalgorithmen erläutert.

7.1.1 Vergleich der Minimierungsalgorithmen anhand einer Potenzfunktion

In diesem Unterkapitel werden die Lösungsalgorithmen mittels einer festgelegten Potenzfunktion (s. Kapitel 6.2) (Formel (6-16)) getestet. Des Weiteren kann durch eine grafische Darstellung der Potenzfunktion eine visuelle Überprüfung der Ergebnisse vorgenommen werden.

Mit den bekannten Minima und dem Sattelpunkt der Funktion kann eine Aussage über die Genauigkeit, die Anzahl der benötigten Iterationen und die Stabilität getroffen werden. Die Stabilität wird anhand unterschiedlicher Startparametersätze untersucht. Dabei werden die verschiedenen Startparametersätze sukzessive mit einer Schrittweite von 0,5 immer weiter vom gesuchten Minimum entfernt angeordnet.

In der nachfolgenden Tabelle 7-1 werden die gewonnenen Ergebnisse der Minimierung der Potenzfunktion mittels der ausgewählten Lösungsalgorithmen dargestellt. Die verwendeten Startparameter $x_{0,max}$ bilden beispielhaft die weitest mögliche Entfernung vom globalen Minimum ab, bei dem der jeweilige Minimierungsalgorithmus noch das Minimum ermitteln kann. Bei den verwendeten Lösungsalgorithmen handelt es sich um den numerisch umgesetzten Newton und Gauß-Newton sowie den von MATLAB implementierten Trust-Region-Verfahren. Für den visuellen Vergleich der gewonnenen Koordinaten x^* der Minima wird in der Abbildung 6-2 ein Höhenliniengraph mit angegebenen idealen Minima dargestellt.

Aufgrund der Ergebnisse in der Tabelle 7-1 lässt sich eine Bewertung der Lösungsalgorithmen durchführen. Die Bewertung der Stabilität erfolgt anhand der Startparameter. Aus dieser Betrachtung heraus ist das Trust-Region-Verfahren der stabilste Lösungsalgorithmus. Das Newton-Verfahren weißt mit den maximalen Startparametern von ca. 0,25 vom Minimum entfernt die schlechteste Stabilität auf. In der detaillierten Betrachtung der Iterationen der beiden stabilsten Lösungsalgorithmen ergeben sich nur minimale Unterschiede. Bei dem Trust-Region-Verfahren ergibt sich die geringste Anzahl an benötigten Iterationen. Des Weiteren hat dieser Lösungsalgorithmus den am weitesten von Minimum entfernten Startparameter. Dieses Phänomen liegt in der Eigenschaft der dynamischen Anpassung des Trust-Region-Bereiches, welcher in jedem Iterationsschritt neu bewertet und verändert wird. Diese Unterschiede sind unter der Betrachtung der Entfernung der Startparameter vom Minimum nicht signifikant. Die Abweichung μ der ermittelten Ergebnisse zu den idealen Ergebnissen ist in allen betrachteten Lösungsalgorithmen nahezu identisch.

Ein Sattelpunkt konnte mit keinem der verwendeten Minimierungsalgorithmen gefunden werden. Das liegt an der speziellen Eigenschaft des Sattelpunktes (s. Kapitel 2.3 und 2.4). Um einen Sattelpunkt zu identifizieren, muss exakt der Punkt getroffen werden, bei dem sich eine horizontale Tangentenebene ausbildet. Ist dies nicht der Fall, so läuft der Minimierungsalgorithmus in das nächste Minimum.

Tabelle 7-1: Ergebnisse der Minimierungsalgorithmen an einer definierten Potenzfunktion

| | Ergebnisse $(x_1 \ x_2)^T$ | | | |
|-------------------------------------|---|---|---|---|
| | Ideale Lösung | Newton / Startparameter x_{0max} | Gauß-Newton / Startparameter x_{0max} | Trust-Region / Startparameter x_{0max} |
| Lokales Minimum | $x^* = \begin{pmatrix} 0,98 \\ 0,97 \end{pmatrix}$ | $x_N^* = \begin{pmatrix} 0,9789 \\ 0,9684 \end{pmatrix} / x_{0max} = \begin{pmatrix} 1,25 \\ 1,25 \end{pmatrix}$ | $x_{GN}^* = \begin{pmatrix} 0,979 \\ 0,968 \end{pmatrix} / x_{0max} = \begin{pmatrix} 4,3 \\ 4,3 \end{pmatrix}$ | $x_{TR}^* = \begin{pmatrix} 0,979 \\ 0,968 \end{pmatrix} / x_{0max} = \begin{pmatrix} 4,5 \\ 4,5 \end{pmatrix}$ |
| Globales Minimum | $x^* = \begin{pmatrix} -0,99 \\ -1,99 \end{pmatrix}$ | $x_N^* = \begin{pmatrix} -0,9948 \\ -1,9922 \end{pmatrix} / x_{0max} = \begin{pmatrix} -1,25 \\ -2,25 \end{pmatrix}$ | $x_{GN}^* = \begin{pmatrix} -0,995 \\ -1,992 \end{pmatrix} / x_{0max} = \begin{pmatrix} 35 \\ 35 \end{pmatrix}$ | $x_{TR}^* = \begin{pmatrix} -0,995 \\ -1,992 \end{pmatrix} / x_{0max} = \begin{pmatrix} 45 \\ 45 \end{pmatrix}$ |
| Sattelpunkt | $x^* = \begin{pmatrix} 0,105 \\ -0,205 \end{pmatrix}$ | Keine Lösung | Keine Lösung | Keine Lösung |
| Iteration i | – | $i_{Lokal} = 9 / i_{global} = 7$ | $i_{Lokal} = 7 / i_{global} = 10$ | $i_{Lokal} = 6 / i_{global} = 7$ |
| Abweichung zur idealen Lösung μ | – | $\mu_{N_{lokal}} = \begin{pmatrix} 1,1e^{-3} \\ 1,6e^{-3} \end{pmatrix}$ $\mu_{N_{global}} = \begin{pmatrix} 4,8e^{-3} \\ 2,2e^{-3} \end{pmatrix}$ | $\mu_{GN_{lokal}} = \begin{pmatrix} 1e^{-3} \\ 2e^{-3} \end{pmatrix}$ $\mu_{GN_{global}} = \begin{pmatrix} 5e^{-3} \\ 2e^{-3} \end{pmatrix}$ | $\mu_{TR_{lokal}} = \begin{pmatrix} 1e^{-3} \\ 2e^{-3} \end{pmatrix}$ $\mu_{TR_{global}} = \begin{pmatrix} 5e^{-3} \\ 2e^{-3} \end{pmatrix}$ |
| Stabilität | – | Schlecht | Gut | Sehr gut |

Bewertung der Stabilität in drei Ebenen: schlecht / gut / sehr gut

7.1.2 Vergleich der Minimierungsalgorithmen auf Grundlage des Simulationsmodells

In diesem Unterkapitel werden die Lösungsalgorithmen anhand eines Simulationsmodells (s. Kapitel 6.3) getestet. Das beschriebene Simulationsmodell besitzt die gleiche Struktur wie das reale mathematische Modell. Der Unterschied zwischen dem Simulationsmodell und dem realen mathematischen Modell liegt in der Lage und Orientierung der einzelnen Koordinatensysteme zueinander (s. Kapitel 6.3). Aufgrund der bekannten Transformationsmatrizen kann eine Aussage über die Genauigkeit, die Anzahl der benötigten Iterationen und die Stabilität getroffen werden.

In dem betrachteten Simulationsmodell sind alle homogenen Transformationsmatrizen und Vektoren bekannt. Dies hat zur Folge, dass sich die zu minimierende Pose p_2 zwischen dem Flansch- und dem Sensor-Koordinatensystem bereits im Minimum befindet. Aufgrund dieser Eigenschaft wird eine manipulierte Startpose $p_{2,Start}$ definiert. Durch diese Manipulation kann eine Minimierung des Simulationsmodells erzwungen werden. Mittels der unterschiedlichen Startposen kann eine Abschätzung der Stabilität der einzelnen Lösungsalgorithmen erzielt werden. Die verwendeten Startposen werden in der nachfolgenden Tabelle 7-2 angegeben.

Tabelle 7-2: Startposen $p_{2,i}$ des Simulationsmodells

| Nr. | Manipulation | Startpose $p_{2,i}$ |
|-----|---|--|
| 0 | Originale Pose | $p_{2,Min} = (5 \ 0 \ 5 \ 0 \ -90 \ 0)^T$ |
| 1 | Verschiebung um die x-Achse | $p_{2,x-trans} (75 \ 0 \ 5 \ 0 \ -90 \ 0)^T$ |
| 2 | Verschiebung um die y-Achse | $p_{2,y-trans} (5 \ 60 \ 5 \ 0 \ -90 \ 0)^T$ |
| 3 | Verschiebung um die z-Achse | $p_{2,z-trans} (5 \ 0 \ 45 \ 0 \ -90 \ 0)^T$ |
| 4 | Drehung um die z-Achse | $p_{2,x-rot} (5 \ 0 \ 5 \ 0,001 \ -90 \ 0)^T$ |
| 5 | Drehung um die y-Achse | $p_{2,y-rot} (5 \ 0 \ 5 \ 0 \ -90,1 \ 0)^T$ |
| 6 | Drehung um die x-Achse | $p_{2,z-rot} (5 \ 0 \ 5 \ 0 \ -90 \ 0,001)^T$ |
| 7 | Verschiebung in allen Achsen | $p_{2,trans} (75 \ 75 \ 55 \ 0 \ -90 \ 0)^T$ |
| 8 | Drehung um alle Achsen | $p_{2,rot} (5 \ 0 \ 5 \ 0,005 \ -90,1 \ 0,005)^T$ |
| 9 | Kombination aus Drehung und Translation | $p_{2,rot-trans} (75 \ 75 \ 55 \ 0,01 \ -90,1 \ 0,01)^T$ |

In der nachfolgenden Tabelle 7-3 werden die gewonnenen Ergebnisse der Minimierung des Simulationsmodells der Lösungsalgorithmen aufgezeigt. Bei den verwendeten Algorithmen handelt es sich um den numerisch umgesetzten Gauß-Newton und das von MATLAB implementierte Trust-Region-Verfahren. Für eine Bewertung der Ergebnisse wird die minimale

Pose $p_{2_{Min}}$ verwendet. Diese Pose stellt das theoretische Minimum des Simulationsmodells dar.

Aufgrund der Ergebnisse in der Tabelle 7-3 ergibt sich eine Bewertung der Lösungsalgorithmen in Bezug auf das verwendete Simulationsmodell. Die Bewertung der Stabilität wird anhand der Startparameter durchgeführt. Aus dieser Betrachtung heraus ist das Gauß-Newton-Verfahren stabiler als das Trust-Region-Verfahren. In der detaillierten Betrachtung der Iterationen, der beiden stabilsten Lösungsalgorithmen, ergeben sich nur minimale Unterschiede. Bei dem Trust-Region-Verfahren ergibt sich bei der Minimierung des mathematischen Modells eine geringfügig größere Anzahl an Iterationen. Des Weiteren wird aufgrund der Entfernung des Startparameters kein Minimum gefunden. Dieses Problem ist im Gauß-Newton-Verfahren nicht zu beobachten. Der Unterschied in den Iterationen liegt in der dynamischen Anpassung des Trust-Region-Bereiches in jedem Iterationsschritt. Des Weiteren sind die Abweichungen μ der Ergebnisposen zum Simulationsminimum nicht signifikant.

Der Minimierungsalgorithmus des Gauß-Newton-Verfahrens stellt sich im Vergleich zu dem Trust-Region-Verfahren als stabiler und schneller heraus. Aufgrund dieser Vorteile wird das Gauß-Newton-Verfahren zur finalen Simulation des realen mathematischen Modells herangezogen.

Tabelle 7-3: Ergebnisse der finalen Minimierungsalgorithmen am Simulationsmodell

| Ergebnisse | | | | | | |
|-------------------|---|--------------------|--|---|--------------------|--|
| Pose | Gauß-Newton-Verfahren | | | Trust-Region-Verfahren | | |
| | p_{2GN}^* | Iteration i_{GN} | | p_{2TR}^* | Iteration i_{TR} | |
| p_{2Min} | $(5 \ 1,8e^{-16} \ 5 \ 3,2e^{-15} \ -90 \ -3,2e^{-15})^T$ | 1 | | $(5 \ 2,3e^{-12} \ 5 \ 2,4e^{-13} \ -90 \ -1,2e^{-10})^T$ | 1 | |
| $p_{2x-trans}$ | $(5 \ 9,3e^{-16} \ 5 \ 5,8e^{-7} \ -90 \ -5,8e^{-7})^T$ | 4 | | $(5 \ 3,4e^{-10} \ 5 \ 7,5e^{-9} \ -90 \ -4,3e^{-6})^T$ | 6 | |
| $p_{2y-trans}$ | $(5 \ 9,7e^{-16} \ 5 \ -5,69e^{-7} \ -90 \ 5,69e^{-7})^T$ | 4 | | $(5 \ 9,3e^{-11} \ 5 \ 7,3e^{-6} \ -90 \ -2,0e^{-6})^T$ | 6 | |
| $p_{2z-trans}$ | $(5 \ 8,2e^{-17} \ 5 \ -5,1e^{-7} \ -90 \ 5,1e^{-7})^T$ | 4 | | $(5 \ 1,4e^{-9} \ 5 \ 3,3e^{-4} \ -90 \ 4,5e^{-5})^T$ | 5 | |
| $p_{2x-rota}$ | $(5 \ 9,3e^{-17} \ 5 \ 1,2e^{-4} \ -90 \ -1,2e^{-4})^T$ | 4 | | $(- \ - \ - \ - \ - \ -)^T$ | - | |
| $p_{2y-rota}$ | $(5 \ 6e^{-16} \ 5 \ -6e^{-4} \ -90 \ 6e^{-4})^T$ | 4 | | $(5 \ 3e^{-10} \ 5 \ 6,1e^{-3} \ -90,004 \ 1,3e^{-3})^T$ | 9 | |
| $p_{2z-rota}$ | $(5 \ 7,9e^{-17} \ 5 \ 4e^{-4} \ -90 \ 4e^{-4})^T$ | 4 | | $(- \ - \ - \ - \ - \ -)^T$ | - | |
| p_{2trans} | $(5 \ -2,4e^{-16} \ 5 \ 1,3e^{-6} \ -90 \ -1,3e^{-6})^T$ | 4 | | $(5 \ 2,7e^{-9} \ 5 \ 4,6e^{-8} \ -90 \ 3e^{-6})^T$ | 6 | |
| p_{2rota} | $(5 \ 1,6e^{-16} \ 5 \ -3,7e^{-4} \ -90 \ 3,7e^{-4})^T$ | 4 | | $(5 \ 4,8e^{-9} \ 5 \ 1,2e^{-2} \ -90,01 \ 1,4e^{-2})^{T*}$ | 12* | |
| $p_{2rota-trans}$ | $(5 \ 1,1e^{-15} \ 5 \ -1,5e^{-4} \ -90 \ 1,5e^{-4})^T$ | 4 | | $(- \ - \ - \ - \ - \ -)^T$ | - | |
| Stabilität | Sehr gut | | | Gut | | |

Bewertung der Stabilität in drei Ebenen: schlecht / gut / sehr gut

(* Angepasste Rotation auf 0,001 statt 0,005)

7.2 Minimierung des realen mathematischen Modells

In diesem Unterkapitel wird die Minimierung des realen mathematischen Modells durchgeführt. Aufgrund der gewonnenen Erkenntnisse aus dem Kapitel 7.1.2 wird der Lösungsalgorithmus Gauß-Newton verwendet.

Die angewandte Minimierung des mathematischen Modells besitzt eine aus den CAD-Daten ausgemessene Startpose p_{2Start} , eine unveränderlichen Pose p_3 sowie einen Vektor x_p , (s. Kapitel 6.4.3). Zudem werden die ermittelten Posen p_{1_i} und Vektoren x_{s_i} aus den 31 Messungen als Inputargumente verwendet (s. Kapitel 6.4.3). Aus diesen Messungen wird das benötigte überbestimmte Gleichungssystem erzeugt. Das für die Minimierung entwickelte Gauß-Newton-Verfahren wird im Kapitel 6.1.5 detailliert erläutert.

In der nachfolgenden Tabelle wird das Ergebnis der Minimierung des mathematischen Modells angegeben. Die Minimierung findet nach 32 Iterationen das gesuchte Minimum. Dabei wird die Schrittweite der Finiten Differenzen Methode mit $1e^{-5}$ und die Toleranz für das Abbruchkriterium mit $1e^{-5}$ festgelegt.

Tabelle 7-4: Ergebnisse der Minimierung des realen mathematischen Modells

| Gauß-Newton-Verfahren | |
|---|---|
| Startpose / Transformationsmatrix | Ergebnispose / Transformationsmatrix |
| $p_{2Start} = \begin{pmatrix} x_{2Start} \\ y_{2Start} \\ z_{2Start} \\ A_{2Start} \\ B_{2Start} \\ C_{2Start} \end{pmatrix} = \begin{pmatrix} 156,99 \\ 87,55 \\ 115 \\ 180 \\ -90 \\ 0 \end{pmatrix}$ | $p_{2End} = \begin{pmatrix} x_{2End} \\ y_{2End} \\ z_{2End} \\ A_{2End} \\ B_{2End} \\ C_{2End} \end{pmatrix} = \begin{pmatrix} 189,69 \\ 99,71 \\ 130,36 \\ -181,40 \\ -89,24 \\ 0,155 \end{pmatrix}$ |
| $T_{FStart}^S = \begin{pmatrix} 0 & 0 & 1 & 156,99 \\ 0 & -1 & 0 & 87,55 \\ 1 & 0 & 0 & 115,00 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ | $T_{FEnd}^S = \begin{pmatrix} -0,013 & -0,021 & 0,999 & 189,69 \\ 0 & -0,999 & -0,021 & 99,71 \\ 0,999 & 0 & 0,013 & 130,36 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ |

7.3 Zusammenfassung und Fazit der gewonnenen Ergebnisse

Auf Grundlage der Ergebnisauswertung kann ein Lösungsalgorithmus ausgewählt werden. Durch die Gegenüberstellung der Überprüfung mittels einer Potenzfunktion und eines Simulationsmodells kann eine Aussage über die Abweichung der Ergebnisse, der Stabilität und des Rechenaufwandes getroffen werden. Unter der Berücksichtigung dieser Faktoren wird zur Minimierung des realen mathematischen Modells das Gauß-Newton-Verfahren gewählt.

Aus der Minimierung des mathematischen Modells wird die unbekannte Pose P_2 ermittelt (s. Tabelle 7-4). Die Validierung des gewonnenen Ergebnisses wird durch die Lösung des realen mathematischen Modells mittels einer beliebigen Roboterposition umgesetzt. Zur Lösung des mathematischen Modells werden alle bekannten homogenen Transformationsmatrizen der zu betrachtenden Roboterposition verwendet. Des Weiteren wird die aus der Minimierung gewonnene homogene Transformationsmatrix (s. Tabelle 7-4) mit eingebunden. Die Ermittlung des finalen x_s Vektors wird durch das Fitten einer Kugel aus den Sensormessdaten gewonnen. Durch die so bestimmten Transformationsmatrizen kann das reale mathematische Modell berechnet werden. Bei einer idealen Berechnung wird die Gleichung (4-3) den Nullvektor ergeben. Dies ist aufgrund der durchgeführten numerischen Berechnungen, der systematischen Roboterfehler sowie des Messrauschens nicht möglich. Die sich ergebene Differenz ε wird in der nachfolgenden Tabelle 7-5 angeführt.

Tabelle 7-5: Ergebnis und Bewertung der minimierten Pose p_2

| | Radius | Durchmesser |
|----------------------------|--|-------------|
| Reales Modell | 30,2770 | 60,5541 |
| Berechnetes Modell | 30,1978 | 60,39561 |
| Abweichungen ε | $(0,24 \quad -0,67 \quad -0,11 \quad 0)^T$ | |

Die Ergebnisse aus der Berechnung des vollständig bekannten mathematischen Modells weichen um 0,1 mm - 0,6 mm von der idealen Lösung ab. Diese Abweichungen resultieren aus den numerischen Berechnungen, der systematischen Roboterfehler sowie des Messrauschens. Des Weiteren ist die Abweichung der bestimmten Koordinaten signifikant von den Messwerten zur Bestimmung des Kugelmittelpunkts abhängig.

8 Zusammenfassung und Ausblick

Die Zielsetzung der vorliegenden Arbeit bestand in der Optimierung der Kalibration von Roboter-Sensor-Systemen. Im Fokus der Betrachtung stand die Analyse bekannter Systeme und die Entwicklung eines physikalischen und mathematischen Modells zur Kalibrierung eines Roboter-Sensor-Systems. Die numerische Umsetzung der Minimierung des mathematischen Modells wurde in der Software MATLAB umgesetzt. Die Validierung der gewonnenen Ergebnisse wurde anhand theoretischer Modelle und dem Vermessen einer sphärischen Kugel durchgeführt.

Auf Grundlage der bereits bestehenden Kenntnisse aus dem aktuellen Stand der Technik wurde ein physikalisches Modell mit den benötigten Randbedingungen und Dimensionen des zu untersuchenden Robotersystems entworfen. Des Weiteren wurde das physikalische Modell so weit abstrahiert, dass nur die signifikanten Systembauteile berücksichtigt wurden. Um eine möglichst genaue Übertragung des Robotersystems in ein mathematisches Modell zu erhalten, wurden ein kalibrierter Roboter sowie ein kalibrierter Sensor verwendet. Die Umsetzung des physikalischen Modells in ein mathematisches Modell wurde auf der Grundlage der Roboterposen und deren äquivalenten homogenen Transformationsmatrizen durchgeführt.

Das mathematische Modell wurde unter Berücksichtigung von bekannten physikalischen Systemen und deren mathematischer Umsetzung entwickelt. Die Überprüfung der Lösungsalgorithmen erfolgte anhand theoretischer Modelle. Dazu wurden eine Potenzfunktion und ein Simulationsmodell herangezogen.

Das in dieser Arbeit entwickelte mathematische Modell kann für die Kalibrierung eines Roboter-Sensor-Systems herangezogen werden. Diese Kalibrierung kann zwischen verschiedenen Koordinatensystemen am Roboter zum Sensor durchgeführt werden. Dadurch kann eine schnellere und effizientere Umrüstung des Roboters vorgenommen werden. Des Weiteren können mit dem entwickelten Minimierungsalgorithmus auch strukturell andersartige mathematische Modelle optimiert werden. Die entwickelte numerische Struktur des Algorithmus und der Unterfunktionen lassen eine einfache und schnelle Anpassung an neuen Unterfunktionen und mathematischen Modellen zu. Somit kann eine sequenzielle Verbesserung oder Optimierung einzelner numerischer Berechnungen umgesetzt werden, ohne den bestehenden Minimierungsalgorithmus zu beeinträchtigen.

Durch diese Kalibrierung kann eine genaue Aussage über die Position des Sensors an dem Roboter getroffen werden. Dies ermöglicht die Erschließung einer Vielzahl von neuen Anwendungsfeldern. Auf der Grundlage der bekannten Position des Sensors kann ein geschlossener Regelkreis in die Robotersteuerung integriert werden. Auf dieser Grundlage lässt sich eine Höhenregelung in der additiven Fertigung umsetzen. Eine Ermittlung von Unregelmäßigkeiten in der aufgebauten Struktur sowie Formabweichungen können zusätzlich umgesetzt werden. Mittels der bekannten Position des Sensors am Roboter können Bauteile und

zu bearbeitende Flächen im Aktuatorbasis-Koordinatensystem ermittelt werden und eine Anpassung der Fluglinien vorgenommen werden. Eine weitere Optimierung der Position des 2-Achs-Positionierers kann mittels der bestehenden Kalibration des Sensors und dem entwickelten Minimierungsalgorithmus umgesetzt werden.

9 Literaturverzeichnis

- [Bec2014] Beck, A.: Introduction to nonlinear optimization. Theory, algorithms, and applications with MATLAB. SIAM Society for Industrial and Applied Mathematics, Philadelphia, Pa., 2014.
- [BeKo1994] Berns, K.; Kolb, T.: Neuronale Netze für technische Anwendungen. Springer, Berlin, Heidelberg, 1994.
- [Bey2004] Beyer, L.: Genauigkeitssteigerung von Industrierobotern Insbesondere mit Parallelkinematik. Dissertation, Hamburg, 2004.
- [DIN2012] DIN EN ISO 8373: Roboter und Robotikgeräte - Wörterbuch, 2012.
- [GaTa2011] Gan, Z.; Tang, Q.: Visual Sensing and its Applications. Integration of Laser Sensors to Industrial Robots. Zhejiang University Press Hangzhou and Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg, 2011.
- [Gri2013] Gritzmann, P.: Grundlagen der Mathematischen Optimierung. Diskrete Strukturen, Komplexitätstheorie, Konvexitätstheorie, Lineare Optimierung, Simplex-Algorithmus, Dualität. Springer, Wiesbaden, 2013.
- [HeMa2016] Hesse, S.; Malisa, V. Hrsg.: Taschenbuch Robotik, Montage, Handhabung. Mit 34 Tabellen. Fachbuchverlag Leipzig im Carl Hanser Verlag, München, 2016.
- [HLN2012] Hertzberg, J.; Lingemann, K.; Nüchter, A.: Mobile Roboter. Eine Einführung aus Sicht der Informatik. Springer Vieweg, Berlin, Heidelberg, 2012.
- [Kab1996] Kaballo, W.: Einführung in die Analysis. Spektrum Akad. Verl., Heidelberg, 1996.
- [KCK2001] Kim, M.-Y.; Cho, H.; Kim, J.H. Hrsg.: Hand/eye calibration of a robot arm with a 3D visual sensor. International Society for Optics and Photonics, 2001.
- [Kön1995] Königsberger, K.: Analysis 1. Springer, Berlin, Heidelberg, 1995.
- [KUR2005] KU Roboter GmbH: Betriebsanleitung KUKA Spez_KR_30_60_HA_en, 2005.
- [LCJ+2011] Li, J. et al.: Calibration of a multiple axes 3-D laser scanning system consisting of robot, portable laser scanner and turntable. In Optik, 2011, 122; S. 324–329.
- [Mai2016] Maier, H.: Grundlagen der Robotik. VDE Verlag GmbH, Berlin, Offenbach, 2016.
- [Mes2015] Messac, A.: Optimization in practice with MATLAB® for engineering students and professionals. Cambridge Univ. Press, New York, NY, 2015.
- [MIC2019] MICRO-EPSILON: Betriebsanleitung scanCONTROL29XX, Königbacher Str. 15 · 94496 Ortenburg / Deutschland, 2019.
- [NoWr2006a] Nocedal, J.; Wright, S. J.: Numerical Optimization. Springer Science+Business Media LLC, New York, NY, 2006.
- [NoWr2006b] Nocedal, J.; Wright, S.: Numerical optimization. Springer Science & Business Media, 2006.

- [Pie2017] Pieper, M.: Mathematische Optimierung. Eine Einführung in die kontinuierliche Optimierung mit Beispielen, 2017.
- [PLB2012a] Papageorgiou, M.; Leibold, M.; Buss, M.: Optimierung. Statische, dynamische, stochastische Verfahren für die Anwendung. Springer, Berlin, Heidelberg, 2012.
- [PLB2012b] Papageorgiou, M.; Leibold, M.; Buss, M.: Optimierung. Statische, dynamische, stochastische Verfahren für die Anwendung. Springer, Berlin, Heidelberg, 2012.
- [Rim2017] Rimscha, M. v.: Algorithmen kompakt und verständlich. Lösungsstrategien am Computer. Springer Vieweg, Wiesbaden, 2017.
- [Sch1993] Schröder, K.: Identifikation von Kalibrationsparametern kinematischer Ketten. Hanser, 1993.
- [Ste2018] Stein, O.: Grundzüge der Nichtlinearen Optimierung. Springer Spektrum, Berlin, Heidelberg, 2018.
- [SuYu2006] Sun, W.; Yuan, Y.-X.: Optimization Theory and Methods. Nonlinear Programming. Springer Science+Business Media LLC, Boston, MA, 2006.
- [Ven2009] Venkataraman, P.: Applied optimization with MATLAB programming. Wiley, Hoboken, NJ, 2009.
- [Web2017] Weber, W.: Industrieroboter. Methoden der Steuerung und Regelung. Fachbuchverlag Leipzig im Carl Hanser Verlag, München, 2017.
- [Wol2016] Wolff, M.: Sensor-Technologien. Band 1: Position, Entfernung, Verschiebung, Schichtdicke. De Gruyter Oldenbourg, Berlin, 2016.
- [WRQ2001] Wollnack, J.; Rall, K.; Quellmalz, W.: Flexibilisierte, automatisierte Großbauteilmontage, 2001.

VI Anhang

A Besetzung der D-Matrix, des U- und F-Vektors nach [KCK2001]

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \\ u_5 & u_6 & u_7 & u_8 \\ u_9 & u_{10} & u_{11} & u_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$U = [u_1 \quad u_2 \quad \dots \quad u_{12}]^T,$$

$$D = \begin{bmatrix} a_{11} - b_{11} & -b_{21} & -b_{31} & -b_{41} & a_{12} & 0 & 0 & 0 & a_{13} & 0 & 0 & 0 \\ -b_{12} & a_{11} - b_{22} & -b_{32} & -b_{42} & 0 & a_{12} & 0 & 0 & 0 & a_{13} & 0 & 0 \\ -b_{13} & -b_{23} & a_{11} - b_{33} & -b_{43} & 0 & 0 & a_{12} & 0 & 0 & 0 & a_{13} & 0 \\ -b_{14} & -b_{24} & -b_{34} & a_{11} - b_{44} & 0 & 0 & 0 & a_{12} & 0 & 0 & 0 & a_{13} \\ a_{21} & 0 & 0 & 0 & a_{22} - b_{11} & & & & a_{23} & 0 & 0 & 0 \\ 0 & a_{21} & 0 & 0 & -b_{12} & \ddots & & & 0 & a_{23} & 0 & 0 \\ 0 & 0 & a_{21} & 0 & & & & & 0 & 0 & a_{23} & 0 \\ 0 & 0 & 0 & a_{21} & & & & & 0 & 0 & 0 & a_{23} \\ a_{31} & 0 & 0 & 0 & a_{32} & 0 & 0 & 0 & a_{33} - b_{11} & & & \\ 0 & a_{31} & 0 & 0 & 0 & a_{32} & 0 & 0 & -b_{12} & \ddots & & \\ 0 & 0 & a_{31} & 0 & 0 & 0 & a_{32} & 0 & & & & \\ 0 & 0 & 0 & a_{31} & 0 & 0 & 0 & a_{32} & & & & a_{33} - b_{44} \end{bmatrix},$$

$$F = [0 \quad 0 \quad 0 \quad -a_{14} \quad 0 \quad 0 \quad 0 \quad -a_{24} \quad 0 \quad 0 \quad 0 \quad -a_{34}]^T.$$

B Anhang aus Kapitel 4**B1 Sensor scanCONTROL 2910-100 Technische Daten [MIC2019]**

| Typ | scanCONTROL | | |
|---|---|---------|----------|
| | 29xx-25 | 29xx-50 | 29xx-100 |
| Messbereich Z-Achse | 25 mm | 50 mm | 100 mm |
| Messbereichsanfang | 53,5 mm | 70 mm | 190 mm |
| Messbereichsende | 78,5 mm | 120 mm | 290 mm |
| Messbereichsanfang, erweitert, ca. | 53 mm | 65 mm | 125 mm |
| Messbereichsende, erweitert, ca. | 79 mm | 125 mm | 390 mm |
| Linienlänge MBM (X-Achse) | 25 mm | 50 mm | 100 mm |
| Linearität ¹ | ± 0,16 % d.M. (3 σ) | | |
| Auflösung X-Achse | 1280 Punkte/Profil | | |
| Profilfrequenz (abhängig vom Sensortyp) | 200 - 2000 Hz | | |
| Lichtquelle Laser | Halbleiterlaser ca. 658 nm, 20 °... 25 ° Öffnungswinkel, Laserklasse 2M: Leistung 8 mW, reduziert 2 mW Halbleiterlaser ca. 658 nm, 20 °... 25 ° Öffnungswinkel, Laserklasse 3B: Leistung 20 mW, reduziert 8 mW | | |
| Schutzgrad (DIN EN 60529) | IP 65 | | |
| Betriebstemperatur | 0 °C bis +45 °C | | |
| Lagertemperatur | -20 °C bis 70 °C | | |
| Ausgänge/Eingänge | Ethernet, Laser on/off (optional), 1x RS422 programmierbar (halbduplex), 3 Schalteingänge programmierbar HTL/TTL, alle Ein-/Ausgänge galvanisch getrennt | | |
| Anzeigen | 1x state / 1x laser on | | |

| Typ | scanCONTROL | | |
|--|---|---------|----------|
| | 29xx-25 | 29xx-50 | 29xx-100 |
| Versorgung | 11 ... 30 VDC, 500 mA IEEE 802.3af Power over Ethernet, Class 2 | | |
| Elektromagnetische Verträglichkeit (EMV) | gemäß: EN 61326-1: 2006-10 DIN EN 55011: 2007-11 (Gruppe 1, Klasse B) EN 61000-6-2: 2006-03 | | |

C Anhang aus Kapitel 6**C1 Entwickelte Funktion zum Importieren der Messdaten**

```
%%*****  
% Ersteller: Christoph Scholl  
% Datum: 01.06.2019  
% Inhalt: Funktion zum Importieren aller Dateien die für die Optimierung  
der  
%       Hand-Auge-Kalibration der Masterthese Christoph Scholl benötigt  
%       werden.  
%  
% Funktion: [P1, P2, P3, XS, XP] = IMPORT_MASTERTHESE()  
%%*****  
  
function [P1, P2, P3, XS, XP] = IMPORT_MASTERTHESE()  
  
% Import der Posen für die homogene Transformationsmatrix T_FA(p1)  
p0 = importdata('Messdaten\Posen\Pose_1_000.txt');  
p1 = importdata('Messdaten\Posen\Pose_1_001.txt');  
p2 = importdata('Messdaten\Posen\Pose_1_002.txt');  
p3 = importdata('Messdaten\Posen\Pose_1_003.txt');  
p4 = importdata('Messdaten\Posen\Pose_1_004.txt');  
p5 = importdata('Messdaten\Posen\Pose_1_005.txt');  
p6 = importdata('Messdaten\Posen\Pose_1_006.txt');  
p7 = importdata('Messdaten\Posen\Pose_1_007.txt');  
p8 = importdata('Messdaten\Posen\Pose_1_008.txt');  
p9 = importdata('Messdaten\Posen\Pose_1_009.txt');  
p10 = importdata('Messdaten\Posen\Pose_1_010.txt');  
p11 = importdata('Messdaten\Posen\Pose_1_011.txt');  
p12 = importdata('Messdaten\Posen\Pose_1_012.txt');  
% Zusammensetzen der Posen in einer [6xi] Matrix  
P1 = [p0 p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12];  
  
% Import der Posen für die homogene Transformationsmatrix T_SF(p2)  
P2 = importdata('Messdaten\Posen\Pose_2_Manipuliert.txt');  
  
% Import der Posen für die homogene Transformationsmatrix T_AP(p3)  
P3 = importdata('Messdaten\Posen\Pose_3.txt');  
  
% Import der Vektoren der einzelnen Messungen des Sensors XS  
XS0 = importdata('Messdaten\Vektoren\Vektor_XS_000.txt');  
XS1 = importdata('Messdaten\Vektoren\Vektor_XS_001.txt');  
XS2 = importdata('Messdaten\Vektoren\Vektor_XS_002.txt');  
XS3 = importdata('Messdaten\Vektoren\Vektor_XS_003.txt');  
XS4 = importdata('Messdaten\Vektoren\Vektor_XS_004.txt');  
XS5 = importdata('Messdaten\Vektoren\Vektor_XS_005.txt');  
XS6 = importdata('Messdaten\Vektoren\Vektor_XS_006.txt');  
XS7 = importdata('Messdaten\Vektoren\Vektor_XS_007.txt');  
XS8 = importdata('Messdaten\Vektoren\Vektor_XS_008.txt');  
XS9 = importdata('Messdaten\Vektoren\Vektor_XS_009.txt');  
XS10 = importdata('Messdaten\Vektoren\Vektor_XS_010.txt');  
XS11 = importdata('Messdaten\Vektoren\Vektor_XS_011.txt');  
XS12 = importdata('Messdaten\Vektoren\Vektor_XS_012.txt');  
  
% Zusammensetzen der Vektoren in einer [4xi] Matrix  
XS = [XS0 XS1 XS2 XS3 XS4 XS5 XS6 XS7 XS8 XS9 XS10 XS11 XS12];  
  
% Import des Vektors des Kalibrierköreppers XP  
XP = importdata('Messdaten\Vektoren\Vektor_XP.txt');  
  
end
```

C2 Skript zur Umrechnung von Posen in h.Transformationsmatrizen

```

%% *****
% Ersteller: Christoph Scholl
% Datum: 01.06.2019
% Inhalt: Umrechnung einer Pose in eine homogen Transformationsmatrix.
%         Eingabewert ist die Pose in [x,y,z,A,B,C] A=Alpha, B=Betta,
%         C=Gamma. Drehungen der Winkel nach den KUKA-Definitionen A um
%         die Z-Achse, B um die Y-Achse, C um die X-Achse in Radiant
% Funktion: [T] = TransFormMatrixRadiant (p)
%% *****

function [T] = TransFormMatrixRadiant (p)

% Roll-Pitch-Yaw Translations-Matrizen
E = eye(3);           % Einheitsmatrix 3x3
a = [0 0 0 1];       % Vektor zum erstellen der vollen Homogenen Matrix
b = [0;0;0];         % Vektor zum erstellen der vollen Homogenen Matrix

alpha = p(4)*pi/180; % Umrechnung in Radiant /Rotation um Z-Achse / A
beta  = p(5)*pi/180; % Umrechnung in Radiant /Rotation um Y-Achse / B
gamma = p(6)*pi/180; % Umrechnung in Radiant /Rotation um X-Achse / C

tx = [p(1);0;0];     % Translation in X-Richtung
ty = [0;p(2);0];     % Translation in Y-Richtung
tz = [0;0;p(3)];     % Translation in Z-Richtung

TX = [horzcat(E, tx)]; % Zusammenstellen der Matrix Horizontal
TY = [horzcat(E, ty)]; % Zusammenstellen der Matrix Horizontal
TZ = [horzcat(E, tz)]; % Zusammenstellen der Matrix Horizontal

Tx = vertcat(TX, a);  % Zusammenstellen der Matrix Vertikal
Ty = vertcat(TY, a);  % Zusammenstellen der Matrix Vertikal
Tz = vertcat(TZ, a);  % Zusammenstellen der Matrix Vertikal

% Roll-Pitch-Yaw Rotations-Matrizen

% Rotation um die x-Achse
rx = [1 0 0; 0 cos(gamma) -sin(gamma); 0 sin(gamma) cos(gamma)];

% Rotation um die y-Achse
ry = [cos(beta) 0 sin(beta); 0 1 0; -sin(beta) 0 cos(beta)];

% Rotation um die z-Achse
rz = [ cos(alpha) -sin(alpha) 0; sin(alpha) cos(alpha) 0; 0 0 1];

RX = [horzcat(rx, b)]; % Zusammenstellen der Matrix Horizontal
RY = [horzcat(ry, b)]; % Zusammenstellen der Matrix Horizontal
RZ = [horzcat(rz, b)]; % Zusammenstellen der Matrix Horizontal

Trx = vertcat(RX, a);  % Zusammenstellen der Matrix Vertikal
Try = vertcat(RY, a);  % Zusammenstellen der Matrix Vertikal
Trz = vertcat(RZ, a);  % Zusammenstellen der Matrix Vertikal

%% Transformationsmatrix
T = Tz*Ty*Tx*Trz*Try*Trx; %!!!! Diese Operation ist nicht kommutativ!!!!

```

C3 Entwickelte Funktion zur Berechnung der Differenz/Epsilon

```

%% *****
% Ersteller: Christoph Scholl
% Datum: 01.06.2019
% Inhalt: Funktion zur Bestimmung der Differenz/Epsilon des mathematischen
%         Modells. Dabei werden die Epsilon_i der einzelnen Parameter und
%         der Epsilon_Vektor mit allen Werten bestimmt
%
% Funktion: [Epsilon, Epsilon_Vektor] = EPSILON(P1,P2,P3,XS,XP)
%% *****

function [Epsilon_Vektor,mse] = EPSILON(P1,P2,P3,XS,XP)
% Definitionen der benötigten Variablen
n = 13; % Anzahl der Messungen/Inputargumente

% Berechnung der konstanten homogenen Transformationsmatrizen und Vektoren
T_SF = TransFormMatrixRadiant (P2(:,1));
T_AP = TransFormMatrixRadiant (P3(:,1));
xp = XP;

% Definitionen der benötigten Variablen
for i = 1:n

    % Erzeugen der Transformationsmatrizen T_FA_i und Vektoren xs_i
    eval( [['T_FA_' int2str(i)] '= TransFormMatrixRadiant (P1(:,i))' ] );
    eval( [['xs_' int2str(i)] '= (XS(:,i))' ] );

end

% Erzeugen der Differenz/Epsilon der einzelnen Gleichungssysteme
eval( [['Epsilon_' int2str(1)] '= (T_FA_1*T_SF*xs_1) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(2)] '= (T_FA_2*T_SF*xs_2) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(3)] '= (T_FA_3*T_SF*xs_3) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(4)] '= (T_FA_4*T_SF*xs_4) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(5)] '= (T_FA_5*T_SF*xs_5) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(6)] '= (T_FA_6*T_SF*xs_6) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(7)] '= (T_FA_7*T_SF*xs_7) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(8)] '= (T_FA_8*T_SF*xs_8) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(9)] '= (T_FA_9*T_SF*xs_9) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(10)] '= (T_FA_10*T_SF*xs_10) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(11)] '= (T_FA_11*T_SF*xs_11) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(12)] '= (T_FA_12*T_SF*xs_12) - (T_AP*xp)' ] );
eval( [['Epsilon_' int2str(13)] '= (T_FA_13*T_SF*xs_13) - (T_AP*xp)' ] );

% Erzeugen des Differenz/Epsilon-Vektors
Epsilon_Vektor = [Epsi-
lon_1;Epsilon_2;Epsilon_3;Epsilon_4;Epsilon_5;...
Epsilon_6;Epsilon_7;Epsilon_8;Epsilon_9;Epsilon_10;Epsilon_11;...
Epsilon_12;Epsilon_13];

% Erzeugen der euklidische Norm
mse = (1/(2*n))*(Epsilon_1+Epsilon_2+Epsilon_3+Epsilon_4+Epsilon_5+...
Epsi-
lon_6+Epsilon_7+Epsilon_8+Epsilon_9+Epsilon_10+Epsilon_11+...
Epsilon_12+Epsilon_13).^2;

end

```


C4 Entwickelte Funktion zur Ermittlung der Jacobimatrix

```

%%
*****
% Ersteller: Christoph Scholl
% Datum: 01.06.2019
% Inhalt: Erstellen der Jacobimatrix mittels der finite
%         Vorwärts-Differenzen Methode (FDM)
%
% Funktion: [Jges] = JACOBIAN (P1,P2,P3,XS,XP)
%%
*****

function [Jges] = JACOBIAN (P1,P2,P3,XS,XP)
% Definitionen der benötigten Variablen
% Berechnung der konstanten homogenen Transformationsmatrizen und Vektoren
T_AP = TransFormMatrixRadiant (P3);
xp = XP;

p =13; % Anzahl der Messungen/Inputargumente

for i = 1:p
    % Erzeugen der Variablen und deren INPUT
    eval( [['T_FA_' int2str(i)] '= TransFormMatrixRadiant (P1(:,i))' ] );
    eval( [['xs_' int2str(i)] '= (XS(:,i))' ] );
end

% Erzeugen der Jacobi-Matrizen
eps = 5.e-7; % Schrittweite
k = length(P2(:,1)); %Länge der Posen
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransFormMatrixRadiant (P2);
    T_SF_EPS = TransFormMatrixRadiant (P2_EPS);

    J1(:,i) = (((T_FA_1*T_SF_EPS*xs_1)-(T_AP*xp))-((T_FA_1*T_SF*xs_1)-
(T_AP*xp)))/eps;
end

for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransFormMatrixRadiant (P2);
    T_SF_EPS = TransFormMatrixRadiant (P2_EPS);

    J2(:,i) = (((T_FA_2*T_SF_EPS*xs_2)-(T_AP*xp))-((T_FA_2*T_SF*xs_2)-
(T_AP*xp)))/eps;
end

for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransFormMatrixRadiant (P2);
    T_SF_EPS = TransFormMatrixRadiant (P2_EPS);

    J3(:,i) = (((T_FA_3*T_SF_EPS*xs_3)-(T_AP*xp))-((T_FA_3*T_SF*xs_3)-
(T_AP*xp)))/eps;
end

```

```

for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J4(:,i) = (((T_FA_4*T_SF_EPS*xs_4)-(T_AP*xp))-((T_FA_4*T_SF*xs_4)-(T_AP*xp)))/eps;
end
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J5(:,i) = (((T_FA_5*T_SF_EPS*xs_5)-(T_AP*xp))-((T_FA_5*T_SF*xs_5)-(T_AP*xp)))/eps;
end
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J6(:,i) = (((T_FA_6*T_SF_EPS*xs_6)-(T_AP*xp))-((T_FA_6*T_SF*xs_6)-(T_AP*xp)))/eps;
end
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J7(:,i) = (((T_FA_7*T_SF_EPS*xs_7)-(T_AP*xp))-((T_FA_7*T_SF*xs_7)-(T_AP*xp)))/eps;
end
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J8(:,i) = (((T_FA_8*T_SF_EPS*xs_8)-(T_AP*xp))-((T_FA_8*T_SF*xs_8)-(T_AP*xp)))/eps;
end
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J9(:,i) = (((T_FA_9*T_SF_EPS*xs_9)-(T_AP*xp))-((T_FA_9*T_SF*xs_9)-(T_AP*xp)))/eps;
end

```

```
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J10(:,i) = (((T_FA_10*T_SF_EPS*xs_10)-(T_AP*xp))-
    ((T_FA_10*T_SF*xs_10)-(T_AP*xp)))/eps;
end
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J11(:,i) = (((T_FA_11*T_SF_EPS*xs_11)-(T_AP*xp))-
    ((T_FA_11*T_SF*xs_11)-(T_AP*xp)))/eps;
end
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J12(:,i) = (((T_FA_12*T_SF_EPS*xs_12)-(T_AP*xp))-
    ((T_FA_12*T_SF*xs_12)-(T_AP*xp)))/eps;
end
for i = 1:k
    P2_EPS = P2;
    P2_EPS(i) = P2(i)+eps;

    T_SF = TransForMatrixRadiant (P2);
    T_SF_EPS = TransForMatrixRadiant (P2_EPS);

    J13(:,i) = (((T_FA_13*T_SF_EPS*xs_13)-(T_AP*xp))-
    ((T_FA_13*T_SF*xs_13)-(T_AP*xp)))/eps;
end

% Zusammensetzen des Jacobi-Vektors
Jges = [J1;J2;J3;J4;J5;J6;J7;J8;J9;J10;J11;J12;J13];

end
```

C5 Entwickeltes Skript des Gauß-Newton-Minimierungsalgorithmus

```

%*****
% Ersteller: Christoph Scholl
% Datum: 01.05.2019
% Inhalt: Hauptskript zur Durchführung der Minimierung des mathematischen
% Modells aus der Masterthese Christoph Scholl: Zur Hand-Auge-Kalibration.
%*****

clear all % Workspace leeren
close all % Alles Fenster schließen
clc % Command_Window leeren
format longG % Ausgabe Format

%% Input Data
% Importieren der Posen für die homogene Transformationsmatrizen und der
% Vektoren zum erstellen und lösen des mathematischen Modells der Hand-Auge
% Kalibration
[P1, P2, P3, XS, XP] = IMPORT_MASTERTHESE();
MaxIter = 250; % Maximale Iterationen des Algorithmus
mse_old = [0; 0; 0; 0]; % Definition des Parameters
tol = 1.e-20; % Toleranz

%% Berechnung
for k = 1:MaxIter
% -----
% Aufstellen der Residuumsfunktion & Berechnung der Residuumsquadrate
% -----

[Epsilon_Vektor, mse] = EPSILON(P1, P2, P3, XS, XP);

% -----
% Bildung der Jacobi-Matrix und ihrer Transponierten
% -----

[Jges] = JACOBIAN (P1, P2, P3, XS, XP);

% -----
% Berechnung der Parameter über die Matrixgleichung
% -----

% Schrittweite
delta_vektor = (Jges'*Jges)\(Jges'*Epsilon_Vektor)
% Nächster Schritte
P2 = P2 - (delta_vektor)

% -----
% Erzeugung des Abbruchkriteriums bei Unterschreitung der Tolleranz
% -----

err = mse - mse_old % Fehlerberechnung
if (abs(err) <= tol)
break
end
mse_old = mse;

end

```

C6 Ansteuerung der MATLAB-Toolbox des realen mathematischen Modells

| Problem Setup and Results | Options |
|---|---|
| Solver: <input type="text" value="fminunc - Unconstrained nonlinear minimization"/> | Stopping criteria |
| Algorithm: <input type="text" value="Trust region"/> | Max iterations: <input checked="" type="radio"/> Use default: 400 |
| Problem | <input type="radio"/> Specify: <input type="text"/> |
| Objective function: <input type="text" value="@MinRealModell"/> | Max function evaluations: <input checked="" type="radio"/> Use default: 100*numberOfVariables |
| Derivatives: <input type="text" value="Approximated by solver"/> | <input type="radio"/> Specify: <input type="text"/> |
| Start point: <input type="text" value="P2"/> | X tolerance: <input checked="" type="radio"/> Use default: 1e-6 |
| Run solver and view results | <input type="radio"/> Specify: <input type="text"/> |
| <input type="button" value="Start"/> <input type="button" value="Pause"/> <input type="button" value="Stop"/> | Function tolerance: <input checked="" type="radio"/> Use default: 1e-6 |
| Current iteration: <input type="text"/> | <input type="radio"/> Specify: <input type="text"/> |
| <input type="button" value="Clear Results"/> | Unboundedness threshold: <input checked="" type="radio"/> Use default: -1e20 |
| | <input type="radio"/> Specify: <input type="text"/> |
| | Function value check |
| | <input type="checkbox"/> Error if user-supplied function returns Inf, NaN or complex |
| | User-supplied derivatives |
| | <input type="checkbox"/> Validate user-supplied derivatives |
| | Hessian sparsity pattern: <input checked="" type="radio"/> Use default: sparse(ones(numberOfVariables)) |
| | <input type="radio"/> Specify: <input type="text"/> |
| | Hessian multiply function: <input checked="" type="radio"/> Use default: No multiply function |
| | <input type="radio"/> Specify: <input type="text"/> |
| | Approximated derivatives |
| | Finite differences $f(x + r*x) - f(x)$ |
| | Type: <input type="text" value="forward differences"/> |
| | Relative perturbation vector r: <input checked="" type="radio"/> Use default: sqrt(eps)*ones(numberOfVariables,1) |
| | <input type="radio"/> Specify: <input type="text"/> |
| | Minimum perturbation r*x : <input checked="" type="radio"/> Use default: 0 |
| | <input type="radio"/> Specify: <input type="text"/> |
| | Maximum perturbation r*x : <input checked="" type="radio"/> Use default: Inf |
| | <input type="radio"/> Specify: <input type="text"/> |
| Final point: <input type="text"/> | |

| Problem Setup and Results | Options |
|---|---|
| Solver: fminunc - Unconstrained nonlinear minimization | Relative perturbation vector r: <input checked="" type="radio"/> Use default: $\sqrt{\text{eps}} \cdot \text{ones}(\text{numberOfVariables}, 1)$ <input type="radio"/> Specify: <input type="text"/> |
| Algorithm: Trust region | Minimum perturbation r*x : <input checked="" type="radio"/> Use default: 0 <input type="radio"/> Specify: <input type="text"/> |
| Problem | Maximum perturbation r*x : <input checked="" type="radio"/> Use default: Inf <input type="radio"/> Specify: <input type="text"/> |
| Objective function: @MinRealModell | Hessian update: BFGS |
| Derivatives: Approximated by solver | Algorithm settings |
| Start point: P2 | Subspace forming method: <input type="radio"/> Cholesky factorization <input checked="" type="radio"/> Preconditioned CG Preconditioner bandwidth: <input type="text" value="0"/> |
| Run solver and view results | Typical X values: <input checked="" type="radio"/> Use default: $\text{ones}(\text{numberOfVariables}, 1)$ <input type="radio"/> Specify: <input type="text"/> |
| <input type="button" value="Start"/> <input type="button" value="Pause"/> <input type="button" value="Stop"/> | Inner iteration stopping criteria |
| Current iteration: <input type="text"/> | Preconditioned conjugate gradient: Maximum iterations: <input checked="" type="radio"/> Use default: $\max(1, \text{floor}(\text{numberOfVariables}/2))$ <input type="radio"/> Specify: <input type="text"/> |
| <input type="button" value="Clear Results"/> | Tolerance: <input checked="" type="radio"/> Use default: 0.1 <input type="radio"/> Specify: <input type="text"/> |
| <div style="border: 1px solid gray; height: 300px; width: 100%;"></div> | Plot functions <input type="checkbox"/> Current point <input type="checkbox"/> Function count <input type="checkbox"/> Function value <input type="checkbox"/> Current step <input type="checkbox"/> First order optimality <input type="checkbox"/> Custom function: <input type="text"/> |
| | Output function <input type="checkbox"/> Custom function: <input type="text"/> |
| | Display to command window Level of display: <input type="text" value="off"/> |
| | <input type="checkbox"/> Show diagnostics |
| Final point: <input type="text"/> | |

C7 Hessematrix der Potenzfunktion, $\nabla^2 f(x)$

$$\nabla^2 f(x) = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}$$

$$H_{11} = \frac{-800(x_1 - 1)^2}{((x_1 - 1)^2 + (x_2 - 1)^2 + 1)^3} + \frac{200}{((x_1 - 1)^2 + (x_2 - 1)^2 + 1)^2} + \frac{400}{((x_1 + 1)^2 + (x_2 + 2)^2 + 1)^2} - \frac{-1600(x_1 - 1)^2}{((x_1 + 1)^2 + (x_2 + 2)^2 + 1)^3}$$

$$H_{12} = \frac{-800(x_1 - 1)(x_2 - 1)}{((x_1 - 1)^2 + (x_2 - 1)^2 + 1)^3} - \frac{1600(x_1 + 1)(x_2 + 2)}{((x_1 + 1)^2 + (x_2 + 2)^2 + 1)^3}$$

$$H_{21} = \frac{-800(x_1 - 1)(x_2 - 1)}{((x_1 - 1)^2 + (x_2 - 1)^2 + 1)^3} - \frac{1600(x_1 + 1)(x_2 + 2)}{((x_1 + 1)^2 + (x_2 + 2)^2 + 1)^3}$$

$$H_{22} = \frac{-800(x_1 - 1)^2}{((x_1 - 1)^2 + (x_2 - 1)^2 + 1)^3} + \frac{200}{((x_1 - 1)^2 + (x_2 - 1)^2 + 1)^2} + \frac{400}{((x_1 + 1)^2 + (x_2 + 2)^2 + 1)^2} - \frac{-1600(x_1 - 1)^2}{((x_1 + 1)^2 + (x_2 + 2)^2 + 1)^3}$$

C8 Entwickler Newton-Algorithmus zur Minimierung der Potenzfunktion

```
%% *****  
% Ersteller: Christoph Scholl  
% Datum: 01.05.2019  
% Inhalt: Newton zur Minimierung der definierten Potenzfunktion  
%% *****  
clear all            % Workspace leeren  
close all           % Alles Fenster schließen  
clc                 % Command_Window leeren  
format longG        % Ausgabe Format  
  
%% Input Data  
MaxIter = 250;            % Maximale Iterationen des Algorithmus  
x = [-1.2;-2.3];  
n=0;                 % Iterationszähler initialisieren  
eps=1;                % Initialisieren des Parameters eps  
  
while eps>1e-10&n<1000  
    G = GradMinFunc_001(x);            % Gradient der Zielfunktion  
    H = HesseMinFunc_001(x);         % Hessematrix der Zielfunktion  
    s = H\G;                         % Schrittweite  
    y=x-s;                            % Neuer Schritte  
    x=y;                              % Update x  
    n=n+1;                            % Zähler 1  
    eps=abs(G(1))+abs(G(2));         % Abbruch Kriterium  
end  
  
n, x, eps                             % Ausgabe aller Werte
```


C9 Entwickelte Funktion / Implementierung der Potenzfunktion

```

%% *****
% Ersteller: Christoph Scholl
% Datum: 01.06.2019
% Inhalt: Zielfunktion/Potenzfunktion
%% *****

function Z=MinFunc_001(x)
Z=(-(100)./((x(1)-1).^2+(x(2)-1).^2)...
    +1))-((200)./((x(1)+1).^2+(x(2)+2).^2+1));
end

```

```

%% *****
% Ersteller: Christoph Scholl
% Datum: 01.06.2019
% Inhalt: Gradient der Zielfunktion (Potenzfunktion)
%% *****

function G=GradMinFunc_001(x)

G = [((200*(x(1)-1))./((x(1)-1).^2+(x(2)-
1).^2+1).^2))+((400*(x(1)+1))./...
    ((x(1)+1).^2+(x(2)+2).^2+1).^2));...
    ((200*(x(2)-1))./((x(1)-1).^2+(x(2)-
1).^2+1).^2))+((400*(x(2)+2))./...
    ((x(1)+1).^2+(x(2)+2).^2+1).^2)];
end

```

```

%% *****
% Ersteller: Christoph Scholl
% Datum: 01.06.2019
% Inhalt: Hessematrix der Zielfunktion (Potenzfunktion)
%% *****

function H =HesseMinFunc_001(x)
hxx =(-(800*(x(1)-1)^2)./((x(1)-1).^2+(x(2)-1).^2+1).^3))+((200)./...
    ((x(1)-1).^2+(x(2)-
1).^2+1).^2))+((400)./((x(1)+1).^2+(x(2)+2).^2+1)...
    .^2))-((1600*(x(1)+1).^2)./((x(1)+1).^2+(x(2)+2).^2+1).^3));
hxy = (-800*(x(1)-1)*(x(2)-1)./((x(1)-1).^2+(x(2)-1).^2+1).^3)...
    -(1600*(x(1)+1)*(x(2)+2)./((x(1)+1).^2+(x(2)+2).^2+1).^3));
hyx = (-800*(x(1)-1)*(x(2)-1)./((x(1)-1).^2+(x(2)-1).^2+1).^3)...
    -(1600*(x(1)+1)*(x(2)+2)./((x(1)+1).^2+(x(2)+2).^2+1).^3));
hyy =(-(800*(x(2)-1)^2)./((x(1)-1).^2+(x(2)-1).^2+1).^3))+((200)./...
    ((x(1)-1).^2+(x(2)-
1).^2+1).^2))+((400)./((x(1)+1).^2+(x(2)+2).^2+1)...
    .^2))-((1600*(x(2)+2).^2)./((x(1)+1).^2+(x(2)+2).^2+1).^3));

H = [hxx hxy; hyx hyy];
end

```

C10 Simulationsposen und InputargumenteDrehung um die z-Achse mit dem Winkel α

| Pose | 0° | 22,5° | 45° | 67,5° | 90° |
|---------------|----|----------------------------------|--------------------------------|----------------------------------|--------------------------------|
| $P_{1\alpha}$ | - | $[20 \ 5 \ 10 \ 22,5 \ 0 \ 0]^T$ | $[20 \ 5 \ 10 \ 45 \ 0 \ 0]^T$ | $[20 \ 5 \ 10 \ 67,5 \ 0 \ 0]^T$ | $[20 \ 5 \ 10 \ 90 \ 0 \ 0]^T$ |

Drehung um die y-Achse mit dem Winkel β

| | | | | | |
|--------------|-------------------------------|----------------------------------|--------------------------------|----------------------------------|--------------------------------|
| $P_{1\beta}$ | $[20 \ 5 \ 10 \ 0 \ 0 \ 0]^T$ | $[20 \ 5 \ 10 \ 0 \ 22,5 \ 0]^T$ | $[20 \ 5 \ 10 \ 0 \ 45 \ 0]^T$ | $[20 \ 5 \ 10 \ 0 \ 67,5 \ 0]^T$ | $[20 \ 5 \ 10 \ 0 \ 90 \ 0]^T$ |
|--------------|-------------------------------|----------------------------------|--------------------------------|----------------------------------|--------------------------------|

Drehung um die x-Achse mit dem Winkel γ

| | | | | | |
|---------------|---|----------------------------------|--------------------------------|----------------------------------|--------------------------------|
| $P_{1\gamma}$ | - | $[20 \ 5 \ 10 \ 0 \ 0 \ 22,5]^T$ | $[20 \ 5 \ 10 \ 0 \ 0 \ 45]^T$ | $[20 \ 5 \ 10 \ 0 \ 0 \ 67,5]^T$ | $[20 \ 5 \ 10 \ 0 \ 0 \ 90]^T$ |
|---------------|---|----------------------------------|--------------------------------|----------------------------------|--------------------------------|

Drehung um die z-Achse mit dem Winkel α

| Vektor | 0° | 22,5° | 45° | 67,5° | 90° |
|---------------|----|-------------------------------|-------------------------------|------------------------------|-------------------------|
| $x_{s\alpha}$ | - | $[-15 \ -3,82 \ -4,23 \ 1]^T$ | $[-15 \ -7,07 \ -2,07 \ 1]^T$ | $[-15 \ -9,23 \ 1,17 \ 1]^T$ | $[-15 \ -10 \ 5 \ 1]^T$ |

Drehung um die y-Achse mit dem Winkel β

| | | | | | |
|--------------|------------------------|------------------------------|--------------------------|----------------------------|----------------------|
| $x_{s\beta}$ | $[-15 \ 0 \ -5 \ 1]^T$ | $[-10,41 \ 0 \ -8,06 \ 1]^T$ | $[-5 \ 0 \ -9,14 \ 1]^T$ | $[0,41 \ 0 \ -8,06 \ 1]^T$ | $[5 \ 0 \ -5 \ 1]^T$ |
|--------------|------------------------|------------------------------|--------------------------|----------------------------|----------------------|

Drehung um die x-Achse mit dem Winkel γ

| | | | | | |
|---------------|---|-------------------------------|-------------------------------|------------------------------|-------------------------|
| $x_{s\gamma}$ | - | $[-14,23 \ -3,82 \ -5 \ 1]^T$ | $[-12,07 \ -7,07 \ -5 \ 1]^T$ | $[-8,82 \ -9,23 \ -5 \ 1]^T$ | $[-5 \ -10 \ -5 \ 1]^T$ |
|---------------|---|-------------------------------|-------------------------------|------------------------------|-------------------------|

| P_2 | P_3 | x'_p |
|------------------------------|-------------------------------|---------------------|
| $[5 \ 0 \ 5 \ 0 \ 90 \ 0]^T$ | $[25 \ -5 \ 5 \ 0 \ 0 \ 0]^T$ | $[5 \ 5 \ 5 \ 1]^T$ |

C11 Zertifikat der Kalibrierkugel TW DK60.0 GE_M8

saphirwerk
swiss precision in ceramics

Calibration Laboratory accredited
by Swiss Accreditation Service
SCS No 0073
Accreditation to ISO/IEC 17025



Certificate of Calibration

| | | | |
|-------------------------|---|---------------------|----------|
| Serial No. | D-11507 | Type of calibration | Standard |
| Origin / old Serial No. | | | |
| Object | 508796 Kalibrierkugel TW DK60.0 GE_M8 | | |
| Material | TW | | |
| Nominal diameter [mm] | 60 | | |
| Order / Number | Calibration of diameter and roundness deviation | 1093299 | |
| Fabricant | Saphirwerk AG, Switzerland | | |

Measures

| | | | | |
|-----------------------------------|---------------------|----------------------------------|--------------------|--|
| Diameter | 59.94507 mm | Measurement uncertainty U_D | 0.00030 mm | |
| Roundness deviation equatorial | 0.264 μm | Measurement uncertainty $URON_L$ | 0.04 μm | |
| Roundness deviation, other planes | 0.258 μm | 0.236 μm | | |

References

| | |
|-----------------|--------------------|
| Temperature | 20° ± 0.5°C |
| Roundness Probe | Ruby ball Ø 2.0 mm |
| Filter | 1-50 W/U, Gaussian |
| Reference ball | N-55-1-KU-01 |

Measurement procedure and measurement conditions:

The measurement of diameter is made on a length measurement machine using mechanical probing. The indicated value was determined in comparison with the reference ball specified above.

The roundness deviation (RDM) was measured according to ISO 12181. It is defined as the peak to valley deviation from the least squares (LSC) circle fitted to the measured profile.

Uncertainty of measurement:

The reported uncertainty of measurement is stated as the combined standard uncertainty multiplied by the coverage factor $k = 2$, which for a normal distribution corresponds to a coverage probability of approximately 95%.

The measurement uncertainty contains contributions originating from the measurement standard, from the calibration method, from the environmental conditions and from the object being calibrated. The long-term characteristic of the product is not included.

Note
















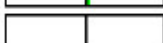


The reported data refer to the product as supplied.

| | | | |
|----------------------|------------|------------------------|-----------------|
| Date | 21.11.2018 | City | 2555 Brügg-Biel |
| For the measurements | | Responsible of the lab | |

C12 Abnahme-Protokoll Wenzel, 3D-Koordinatenmessmaschine X0 107 3D

| KMG-Abnahme-Protokoll | | WENZEL® <small>The company of a</small> |
|--|--|--|
| Auftrag | <u>Komm.-Nr.:</u> SA 36786 Wartung 2019 <u>Protokoll Nr.:</u> 2017-HW-0027_1 | |
| Kunde | Laser Zentrum Nord GmbH Messraum Am Schleusengraben 14 21079 Hamburg | <u>Ansprechpartner:</u> Peter Lindecke <u>Gerätestandort:</u> Messraum <u>Klimatisierung:</u> ja |
| Messgerät | <u>Masch.-Nr.:</u> 116957 <u>Masch.-Typ:</u> LH 87 / CNC <u>Baujahr:</u> 2011 <u>Messbereich:</u> X-Achse: 800 mm Y-Achse: 1000 mm Z-Achse: 700 mm | <u>Tasteinrichtung:</u> Tastkopf: PH 10M Verlängerung: 0 mm Tastsystem-Typ: SP25 M Tastsystem-Nr.: 51C543 Tasterdurchmesser: 6,0 mm Tasterlänge: 30 mm |
| <u>Grenzwert Längenmessabweichung:</u> $MPE_E = 1,7 \mu\text{m} + (L / 350,0 \text{ mm}) \mu\text{m}$ ohne Maximalwertbegrenzung (gültig im gesamten Messbereich) | | |
| Prüfung | <u>Richtlinie:</u> ISO 10360-2, -4 <u>Prüfmittel:</u> Die für die Abnahmemessungen verwendeten Prüfmittel unterliegen alle der regelmäßigen Prüfmittelüberwachung. Ihre Identifikationsnummern sind auf den folgenden Seiten angegeben sowie Kopien der Deckblätter der Kalibrierscheine zum Nachweis des ordnungsgemäßen Anschlusses an nationale Längennormale als Anlagen beigelegt. | <u>Prüfer:</u> M.Surmann |
| <u>Prüfaufgaben:</u> - Antastabweichung 3D - Achsparallele Längenmessung - Raumdiagonale Längenmessung - Scanning-Antastabweichung | | <u>Prüferteil:</u> geprüft mit Kugelnormale i.O. geprüft mit Parallelendmaßen i.O. geprüft mit Parallelendmaßen i.O. geprüft mit Kugelnormale i.O. |
| <u>Gesamterteil:</u> Das o.a. Koordinatenmeßgerät hat unter den vorliegenden Umgebungsbedingungen die spezifizierte Genauigkeit bei allen Prüfaufgaben eingehalten. | | |
| <u>Ergänzende Bemerkungen:</u> - keine - | | |
| Testat | Der WENZEL PRÄZISION GmbH wird hiermit die Erfüllung Ihrer Leistungen bescheinigt. | |
| | Ort: <u>Hamburg</u> | Datum: <u>10.04.19</u> |
| | _____ Kunde | _____ WENZEL PRÄZISION GmbH |

C13 Messprotokoll 3D-Koordinatenmessmaschine X0 107 3D, Kalibrierkörper

| ID | Merkmalstyp | | | | | Wirklänge | | Grafik |
|--|------------------------------------|---------|--------|---------|-----------|-----------|------|---|
| | Nennwert | ISO 286 | OTol | UTol | Istwert | Abw | %Abw | |
|  1 | Position [x y z] | | | | | | | |
| x | -103.0000 | | 0.2000 | -0.2000 | -102.9614 | 0.0386 | 19% |  |
| y | -440.6000 | | 0.2000 | -0.2000 | -440.6013 | -0.0013 | -1% |  |
| z | 66.7000 | | 0.2000 | -0.2000 | 66.7128 | 0.0128 | 6% |  |
|  3 | Durchmesser | | | | | | | |
| ∅ | 60.6000 | | 0.2000 | -0.2000 | 60.5541 | -0.0459 | -23% |  |
|  4 | Formabweichung | | | | | | | |
| R | 0.0000 | | 0.2000 | | 0.0674 | 0.0674 | 34% |  |
|  2 | Position [x y z] | | | | | | | |
| x | -4.4000 | | 0.2000 | -0.2000 | -4.4199 | -0.0199 | -10% |  |
| y | -540.5000 | | 0.2000 | -0.2000 | -540.5178 | -0.0178 | -9% |  |
| z | -114.9000 | | 0.2000 | -0.2000 | -114.8878 | 0.0122 | 6% |  |
|  5 | Abstand Punkt-Punkt [x y z] | | | | | | | |
| x | 98.5000 | | 0.2000 | -0.2000 | 98.5415 | 0.0415 | 21% |  |
| y | 99.9000 | | 0.2000 | -0.2000 | 99.9165 | 0.0165 | 8% |  |
| z | 181.6000 | | 0.2000 | -0.2000 | 181.6006 | 0.0006 | 0% |  |
|  6 | Abstand Punkt-Gerade | | | | | | | |
| d | 0.0000 | | 0.2000 | -0.2000 | 0.0000 | 0.0000 | 0% |  |

C14 Posen der gemessenen Inputparameter T_F^A Pose bei einer Drehung um die x-Achse mit dem Winkel γ

| γ | 0° | 22,5° | 45° | 67,7° | 90° | 112,5° | 135° | 157,5° | 180° |
|-----------------|---|--|--|--|--|--|--|---|--|
| $P_{1x,\gamma}$ | $\begin{pmatrix} 1397,69 \\ -183,84 \\ 1223,09 \\ -90,04 \\ 0,02 \\ -90,16 \end{pmatrix}$ | $\begin{pmatrix} 1397,56 \\ -248,24 \\ 1379,10 \\ -90,10 \\ 22,53 \\ -90,16 \end{pmatrix}$ | $\begin{pmatrix} 1379,64 \\ -380,04 \\ 1483,82 \\ -90,17 \\ 45,03 \\ -90,18 \end{pmatrix}$ | $\begin{pmatrix} 1401,96 \\ -532,52 \\ 1555,41 \\ -90,31 \\ 67,52 \\ -90,30 \end{pmatrix}$ | $\begin{pmatrix} 1402,14 \\ -678,57 \\ 1546,95 \\ 172,24 \\ 89,87 \\ 172,26 \end{pmatrix}$ | $\begin{pmatrix} 1398,03 \\ -841,25 \\ 1479,84 \\ 90,32 \\ 67,48 \\ 90,37 \end{pmatrix}$ | $\begin{pmatrix} 1389,07 \\ -965,49 \\ 1371,51 \\ 90,14 \\ 44,98 \\ 90,21 \end{pmatrix}$ | $\begin{pmatrix} 1398,09 \\ -1026,85 \\ 1213,46 \\ 90,06 \\ 22,47 \\ 90,16 \end{pmatrix}$ | $\begin{pmatrix} - \\ - \\ - \\ - \\ - \\ - \end{pmatrix}$ |

Pose bei einer Drehung um die y-Achse mit dem Winkel β

| β | 0° | 22,5° | 45° | 67,7° | 90° | 112,5° | 135° | 157,5° | 180° |
|----------------|--|--|---|--|---|--|--|--|--|
| $P_{1y,\beta}$ | $\begin{pmatrix} - \\ - \\ - \\ - \\ - \\ - \end{pmatrix}$ | $\begin{pmatrix} 1076,09 \\ -470,88 \\ 1224,03 \\ 0,07 \\ 22,5 \\ 90,12 \end{pmatrix}$ | $\begin{pmatrix} 1171,59 \\ -471,21 \\ 1367,76 \\ 0,14 \\ 45,06 \\ 90,15 \end{pmatrix}$ | $\begin{pmatrix} 1288 \\ -471,28 \\ 1484,41 \\ 0,27 \\ 67,57 \\ 90,26 \end{pmatrix}$ | $\begin{pmatrix} 1445,82 \\ -471,36 \\ 1546,20 \\ 125,19 \\ 89,88 \\ -143,82 \end{pmatrix}$ | $\begin{pmatrix} 1600,89 \\ -471,36 \\ 1551,82 \\ 179,75 \\ 67,41 \\ -90,28 \end{pmatrix}$ | $\begin{pmatrix} 1759,17 \\ -471,31 \\ 1492,73 \\ 179,89 \\ 44,91 \\ -90,15 \end{pmatrix}$ | $\begin{pmatrix} 1875,75 \\ -471,24 \\ 1376,73 \\ 179,95 \\ 22,40 \\ -90,11 \end{pmatrix}$ | $\begin{pmatrix} 1905,84 \\ -471,61 \\ 1223,41 \\ -180 \\ -0,13 \\ -90,12 \end{pmatrix}$ |

Pose bei einer Drehung um die x^* -Achse mit dem Winkel γ^*

| γ^* | 0° | 22,5° | 45° | 67,7° | 90° | 112,5° | 135° | 157,5° | 180° |
|---------------------|--|---|---|---|---|--|--|---|--|
| P_{1x^*,γ^*} | $\begin{pmatrix} 1707,76 \\ -232,81 \\ 1225,91 \\ -135 \\ -0,03 \\ -90,16 \end{pmatrix}$ | $\begin{pmatrix} 1683,53 \\ -256,89 \\ 1381,34 \\ -135,05 \\ 22,48 \\ -90,16 \end{pmatrix}$ | $\begin{pmatrix} 1591,03 \\ -341,92 \\ 1493,79 \\ -135,14 \\ 44,99 \\ -90,21 \end{pmatrix}$ | $\begin{pmatrix} 1481,88 \\ -450,96 \\ 1558,75 \\ -135,31 \\ 67,49 \\ -90,36 \end{pmatrix}$ | $\begin{pmatrix} 1374,89 \\ -557,86 \\ 1551,55 \\ 141,46 \\ 89,87 \\ 173,556 \end{pmatrix}$ | $\begin{pmatrix} 1250,60 \\ -690,15 \\ 1481,44 \\ 45,38 \\ 67,52 \\ 90,37 \end{pmatrix}$ | $\begin{pmatrix} 1168,17 \\ -771,90 \\ 1349,32 \\ 45,18 \\ 45,02 \\ 90,02 \end{pmatrix}$ | $\begin{pmatrix} 1100,75 \\ -839,4 \\ 1205,51 \\ 45,11 \\ 22,52 \\ 90,02 \end{pmatrix}$ | $\begin{pmatrix} - \\ - \\ - \\ - \\ - \\ - \end{pmatrix}$ |

Pose bei einer Drehung um die y^* -Achse mit dem Winkel β^*

| β^* | 0° | 22,5° | 45° | 67,7° | 90° | 112,5° | 135° | 157,5° | 180° |
|--------------------|---|---|---|---|---|---|---|--|--|
| P_{1y^*,β^*} | $\begin{pmatrix} 1141,27 \\ -388,66 \\ 1223,56 \\ -45,03 \\ 0,05 \\ -90,10 \end{pmatrix}$ | $\begin{pmatrix} 1196,82 \\ -444,44 \\ 1378,26 \\ -45,06 \\ 22,5 \\ -90,11 \end{pmatrix}$ | $\begin{pmatrix} 1284,82 \\ -532,63 \\ 1494,82 \\ -45,1 \\ 45,06 \\ -90,13 \end{pmatrix}$ | $\begin{pmatrix} 1393,95 \\ -641,7 \\ 1556,43 \\ -45,25 \\ 67,57 \\ -90,25 \end{pmatrix}$ | $\begin{pmatrix} 1513,68 \\ -761,16 \\ 1533,08 \\ -170,19 \\ 89,86 \\ 144,82 \end{pmatrix}$ | $\begin{pmatrix} 1672,11 \\ -874,49 \\ 1481,30 \\ 135,27 \\ 67,40 \\ 90,29 \end{pmatrix}$ | $\begin{pmatrix} 1715,30 \\ -962,19 \\ 1400,83 \\ 135,13 \\ 44,88 \\ 90,18 \end{pmatrix}$ | $\begin{pmatrix} - \\ - \\ - \\ - \\ - \\ - \end{pmatrix}$ | $\begin{pmatrix} - \\ - \\ - \\ - \\ - \\ - \end{pmatrix}$ |

C15 Funktion zur Bestimmung der Mittelpunktkoordinaten einer Kugel

```
%% *****  
% Ersteller: Christoph Scholl  
% Datum: 01.06.2019  
% Inhalt: Function zum Fitten eines Kreises durch gegebenen Punkte  
%  
% Funktion: [xFit, zFit ,rFit] = CircleFit (x,z)  
% a = der Koeffizient;  
% Kreisfunktion => x^2+y^2+a(1)*x+a(2)*y+a(3)=0  
%%*****  
  
function [xFit, zFit ,rFit] = CircleFit (x,z)  
  
% Verwenden aller X und Z Werte  
x=x(:);  
z=z(:);  
  
% Koeffizient  
a=[x z ones(size(x))]\[-(x.^2+z.^2)];  
  
% Berechnung des Mittelpunktes und des Radius  
xFit = -.5*a(1);  
zFit = -.5*a(2);  
rFit = sqrt((a(1)^2+a(2)^2)/4-a(3));
```

C16 Entwickeltes Skript zur Ermittlung des Kugelmittelpunktes und der graphischen Darstellung des Kreises

```

%% *****
% Ersteller: Christoph Scholl
% Datum: 01.06.2019
% Inhalt: Fit Kugel aus Messungen der des Triangulationssensors
%         Verwendeter Sensor: Der Firma MicroEpsilon;
%         Modell: scan CONTROL 29xx-100
%% *****
clear all      % Workspace leeren
close all     % Alles Fenster schließen
clc           % Command Window leeren
format longG % Ausgabe Format
%% Importdate

p1 = importda-
ta('E:\01_Masterthese_Roboterzelle_HandAuge\03_Experiment\02_FINALE_Messdat
en_TCP_und_FLANSCH\01_Drehung_X_Achse\Experimentelle_Aufnahmen_Drehung_X_Ac
hse\Drehung_X_Achse_45_00001.txt');      % Importieren der Messwerte ei-
ner Messung mit 8 Messpunkten
p2 = importda-
ta('E:\01_Masterthese_Roboterzelle_HandAuge\03_Experiment\02_FINALE_Messdat
en_TCP_und_FLANSCH\01_Drehung_X_Achse\Experimentelle_Aufnahmen_Drehung_X_Ac
hse\Drehung_X_Achse_45_00002.txt');      % Importieren der Messwerte ei-
ner Messung mit 8 Messpunkten

% Austellen der Wertepaare aus den Mittelwerten

x = [p1(:,1);p2(:,1)];
z = [p1(:,2);p2(:,2)];

%% Aufrufen der Funktion circlFit

[xFit, zFit ,rFit] = CircleFit (x,z)

%% Koordinaten der Kugel

Xs = [xFit; -sqrt(30^2-rFit^2); zFit ]

%% Plotten
set(0,'DefaultTextFontName','Times New Roman');
figure('Name','Kreis gefittet aus den Musspunkten','NumberTitle','off')

hold on
% Plot der Messwerte
A = plot(x,z);
    A(1).LineStyle = 'none';
    A(1).Marker = 'o';
    A(1).MarkerSize =6;
    A(1).MarkerEdgeColor ='[0.0902 0.6118 0.4902]';
    A(1).MarkerFaceColor ='[0.0902 0.6118 0.4902]';

% Plot vom gefitteten Kries
B = rectangle('position',[xFit-rFit,zFit-
rFit,rFit*2,rFit*2],'curvature',[1,1]);
    B(1).LineStyle = '-.';
    B(1).LineWidth = 2 ;

```

```
% Plot vom Mittelpunkt
C = plot(xFit,zFit,'g. ');
    C(1).LineStyle = 'none';
    C(1).Marker = 'o';
    C(1).MarkerSize =6;
    C(1).MarkerEdgeColor = '[.94901 .58039 0]';
    C(1).MarkerFaceColor = '[.94901 .58039 0]';

gg = legend('Messpunkte','Mittelpunkt '); % Legendeneinträge
gg.FontSize = 16; % Schriftgröße der Legendeneinträge
title(gg, 'Legende', 'FontSize',16) % Titel der Legen der

axis equal
grid on
grid minor

title(sprintf('Best fit: R = %0.2f; Center =
[%0.2f,%0.2f,%0.2f]',rFit,Xs(1),Xs(2),Xs(3))) % Titel des Plots
% Achsenbeschriftung
xlabel('X-Achse','FontSize',16,'FontWeight','bold','Color','k')
ylabel('Z-Achse','FontSize',16,'FontWeight','bold','Color','k')
% Achsenbereiche
xlim([xFit-rFit-2,xFit+rFit+2])
ylim([zFit-rFit-2,zFit+rFit+2])
% Schriftgröße der Achsenbeschriftungen
set(gca,'FontSize',16)
hold off
```

C17 Messungen des Inputparameters des gemessenen Vektors X_S Vektor bei einer Drehung um die x-Achse mit dem Winkel γ

| γ | 0° | 22,5° | 45° | 67,7° | 90° | 112,5° | 135° | 157,5° | 180° |
|---------------|---|---|---|--|--|--|---|---|--|
| $x_{s\gamma}$ | $\begin{pmatrix} 3,386 \\ -9,170 \\ 230,275 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 3,705 \\ -10,163 \\ 2237,395 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 4,081 \\ -6,972 \\ 225,140 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} -0,311 \\ -1,169 \\ 242,678 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} -0,001 \\ 14,668 \\ 236,955 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 4,049 \\ 8,235 \\ 234,279 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 3,809 \\ 11,780 \\ 245,602 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 3,807 \\ 14,373 \\ 239,657 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} - \\ - \\ - \\ 1 \end{pmatrix}$ |

Vektor bei einer Drehung um die y-Achse mit dem Winkel β

| β | 0° | 22,5° | 45° | 67,7° | 90° | 112,5° | 135° | 157,5° | 180° |
|--------------|--|---|---|--|--|---|---|--|---|
| $x_{s\beta}$ | $\begin{pmatrix} - \\ - \\ - \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 3,343 \\ 13,812 \\ 269,069 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 3,720 \\ 13,932 \\ 238,335 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 3,464 \\ 8,958 \\ 239,055 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 3,260 \\ 9,838 \\ 236,583 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 3,114 \\ -5,684 \\ 237,130 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 2,873 \\ -6,973 \\ 237,396 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 2,977 \\ -10,481 \\ 230,240 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 3,966 \\ -9,962 \\ 189,154 \\ 1 \end{pmatrix}$ |

Vektor bei einer Drehung um die x^* -Achse mit dem Winkel γ^*

| γ^* | 0° | 22,5° | 45° | 67,7° | 90° | 112,5° | 135° | 157,5° | 180° |
|-----------------|---|--|---|--|--|--|--|--|--|
| $x_{s\gamma^*}$ | $\begin{pmatrix} 7,231 \\ -12,428 \\ 199,93 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 6,692 \\ -12,515 \\ 238,281 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 12,320 \\ -9,715 \\ 236,66 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 12,410 \\ -7,157 \\ 244,248 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 12,941 \\ 14,582 \\ 241,866 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 7,406 \\ -10,238 \\ 243,614 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 7,465 \\ -12,466 \\ 238,344 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} 6,627 \\ -10,370 \\ 278,886 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} - \\ - \\ - \\ 1 \end{pmatrix}$ |

Vektor bei einer Drehung um die y^* -Achse mit dem Winkel β^*

| β^* | 0° | 22,5° | 45° | 67,7° | 90° | 112,5° | 135° | 157,5° | 180° |
|----------------|--|---|--|---|--|---|--|--|--|
| $x_{s\beta^*}$ | $\begin{pmatrix} -4,358 \\ -9,159 \\ 236,772 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} -3,724 \\ -11,993 \\ 229,873 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} -3,603 \\ -14,515 \\ 232,41 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} -3,504 \\ -11,574 \\ 239,156 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} -3,201 \\ -9,649 \\ 223,055 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} -3,625 \\ -12,111 \\ 244,074 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} -4,329 \\ 16,815 \\ 281,644 \\ 1 \end{pmatrix}$ | $\begin{pmatrix} - \\ - \\ - \\ 1 \end{pmatrix}$ | $\begin{pmatrix} - \\ - \\ - \\ 1 \end{pmatrix}$ |

D Anhang CD

Auf der CD befinden sich im Ordner „Anhang_zur_Masterthese_von_Christoph_Scholl“ Rohdaten der Messungen, Daten mit Tabellen und Bildern, die zur Strategie und Transparenz der geschriebenen Masterthese beitragen. Diese Variante wird aus den Gründen der Darstellbarkeit gewählt. Die Tabellen, Bilder und Rohdaten auf der CD, würden auf einer Dina A4 Seite unübersichtlich oder nicht erkennbar werden.