Philipp Simon Eisele, B.Sc.

Konzept eines kollaborativen Gelenkarmroboters mit Arduino-Steuerung

Masterarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Individuelles Masterstudium

Wirtschaftsingenieurwesen- Maschinenbau

eingereicht an der

Technischen Universität Graz

Betreuer

Univ.-Prof. Dipl.-Ing. Dr.techn. Franz Haas

Institut für Fertigungstechnik

Graz, Mai 2018

Kurzfassung

Die Digitalisierung eröffnet neue Möglichkeiten in einer flexiblen Fertigung. Durch den Einsatz von kollaborativen Robotern (engl. collaborativ robot Abk.: Cobot) können auch Aufgaben automatisiert werden, die früher aufgrund von Komplexität oder notwendiger Flexibilität nicht wirtschaftlich automatisiert werden konnten. Die Vorteile von Mensch und Maschine werden durch den Einsatz von Cobots bei der Bearbeitung eines Werkstücks kombiniert. Kollaborative Roboter unterscheiden sich von herkömmlichen Industrierobotern durch ihre kompaktere und leichte Bauart sowie ihrer Wahrnehmung der Umgebung durch Sensoren. Dank der Sensitivität des Cobots ist eine Zusammenarbeit von Mensch und Roboter im selben Arbeitsraum möglich. Dadurch können Tätigkeiten, die für den Roboter zu komplex und für den Menschen auf Grund von Umwelteinflüssen oder anderen Faktoren nicht oder nur schwer möglich sind, effizient ausgeführt werden. Damit ist ein Einsatz in einer flexiblen Fertigung realisierbar, da eine abgeschirmte Roboterzelle und die Programmierung der Umgebungsgegebenheiten entfallen.
Im Zuge der Arbeit wurde im Anschluss an eine Literaturrecherche nach den Anforderungen und Auflagen eines Cobots ein sechsachsiger Gelenkarmroboter entwickelt und konstruiert. Hierbei soll für die Übersetzung des Motormomentes das Wellgetriebe des Instituts für Fertigungstechnik (Abk.: IFT) verwendet werden. Die weiteren Komponenten wurden mit Hinblick auf den späteren Einsatz in der Forschung ausgewählt. Es wurden keine herstellergebundenen Komponenten verwendet und die Software wurde selbst erstellt. Auf Basis der Konstruktion wurden die kinematischen Beziehungen des Roboters ermittelt. Um eine Steuerung mittels MATLAB zu realisieren wurde die direkte und inverse Positionierung zuerst errechnet und anschließend in Programmcode umgesetzt. Die Berechnung von Kreisen und Linien wurde integriert, um für die Bahnerstellung genutzt werden zu können. Als Schnittstelle zwischen PC und Motoren wurde ein Arduino Uno verwendet, dieser steuert die Motoren des Roboters über einen Motortreiber an und verarbeitet die Sensorwerte. Für die Validierung der Funktionsweise des Gesamtsystems wurde es an einer Konfiguration mit drei Schrittmotoren erfolgreich getestet.

Abstract

Digitalization opens up new possibilities in flexible production. By using collaborative robots (abbr.: cobot) it is now also possible to automate tasks that previously could not be automated economically due to complexity or necessary flexibility. The advantages of man and machine are combined by the use of Cobots when handling a workpiece. Collaborative robots differ from traditional industrial robots in their more compact and lighter design and perception of the environment by sensors. Thanks to the sensitivity of the Cobot, it is possible to collaborate between humans and robots in the same workspace. As a result, activities that are too complex for the robot or too difficult or impossible to do for humans can be carried out efficiently. This makes usage in flexible production feasible because shielded robot cells omitted.
In the course of the work, a six-axis articulated robot was developed and constructed following a literature search according to the requirements and obligations of a Cobot. Here, the gearbox of the Institute of Production Engineering (abbr.: IFT) should be used for the transmission of the engine torque. The other components were selected for future use in research. No manufacturer-bound components were used and the software is self developed. Based on the design, the kinematic relationships of the robot were determined. In order to realize a control via MATLAB, the direct and inverse positioning was first calculated and then converted to program code. The calculation of circles and lines has been integrated in order to be used for the path creation. An Arduino Uno was used as interface between PC and engines, this controls the motors of the robot via a motor driver and processes the sensor values. For the validation of the functioning of the whole system, it was successfully tested on a configuration with three stepper motors.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Graz, den 04. Mai 2018

Philipp Simon Eisele, B.Sc.

Gender Erklärung

Aus Gründen der besseren Lesbarkeit wird in dieser Masterarbeit die Sprachform des generischen Maskulinums angewendet. Es wird an dieser Stelle darauf hingewiesen, dass die ausschließliche Verwendung der männlichen Form geschlechtsunabhängig verstanden werden soll.

Inhaltsverzeichnis

Nomenclature

CAD	computer-aided design
APAS	Automated produktion assistant
Cobot	collaborative robot
CP	continuous path
DH	Denavit-Hartenberg
ESD	electrostatic discharge
GUI	graphical user interface
IFT	Institut für Fertigungstechnik
LBR	Leichtbauroboter
PTP	Point-to-Point
TCP	Tool Center Point
WCP	Wrist Center Point

1 Einleitung

Bisher galt der Grundsatz, dass die Vorteile eines Industrieroboters in der Ausführung der immer selben Aufgabe liegen. Hersteller von Massenware haben daher schon lange Industrieroboter in ihrer Fertigung im Einsatz. Für den Betrieb eines Industrieroboters werden trennende Schutzeinrichtungen benötigt, um Mitarbeiter gegen Verletzungen durch den Roboter zu schützen. Dadurch ist ein flexibler Einsatz nur schwer möglich, da die Sicherheitseinrichtungen an den Bearbeitungsprozess angepasst werden müssen. Für klein- und mittelständischen Unternehmen, die hauptsächlich kundenspezifisch fertigen, kam daher der Einsatz eines herkömmlichen Industrieroboters nicht in Frage. Mit Robotern, die ihre Umgebung selbstständig wahrnehmen können und ihre Bewegungen den Umständen anpassen können, werden die starren Sicherheitseinrichtungen obsolet. Bereits im Jahre 1996 entwickelten James Edward Colgate und Michael A. Peshkin den ersten Cobot für General Motors[1]. In ihrem im Jahre 1999 veröffentlichten Patent definieren sie einen Cobot als:

„an apparatus and method for direct physical interaction between a person and a general purpose manipulator controlled by a computer“[2]

Mit der steigenden Rechenleistung von Computern und der besseren Vernetzbarkeit von Maschinen im Zuge der Digitalisierung sind die Kosten für einen Cobot in den letzten Jahren gefallen und das Angebot der Hersteller an verschiedenen Modellen gestiegen.
Um zusammen mit dem Menschen im selben Arbeitsraum arbeiten zu können, muss der Cobot den Ansprüchen der Normen ISO 10218 Teil 1 und 2 sowie ISO/TS 15066 genügen.[3] Diese Normen schließen auch die angeschlossenen Werkzeuge sowie die bewegten Bauteile in die Risikobewertung mit ein. Im Rahmen der Kollaboration assistiert der Roboter dem Menschen. Kognitive und sensorische Fähigkeiten sowie die Möglichkeit des schnellen Lernens des Menschen werden mit den Vorteilen wie Wiederholungsgenauigkeit und Kraft eines Industrieroboters kombiniert.[4] Der Mensch kann dem Roboter schwierige Entscheidungen abnehmen und in unbekannten Situationen unterstützen. Im Gegenzug reduziert der Roboter den Anteil an

[1] Teresko 2004
[2] *James Edward Colgate 1999*
[3] ISO10218-1 2012; ISO10218-2 2011; ISO/TS15066 2016
[4] Weber 2017, S. 27

monotoner oder körperlich belastender Arbeit sowie die Arbeit in gefährlichen Umgebungen. Diese Symbiose erfordert jedoch auch ein hohes Maß an Sicherheit, um den Bedienenden vor Verletzungen durch den Roboter und dessen Werkzeugen zu schützen.[5]

Diese neue Technologie beginnt sich gerade neben großen Industrieunternehmen auch in klein- und mittelständischen Unternehmen zu etablieren. Die verschiedenen Hersteller von Cobots versuchen durch einfache Programmierung und schnelle Inbetriebnahme ihrer Produkte bei den Kunden die Bereitschaft zum Einsatz eines Cobots zu schaffen. Bisher sind für den Einsatz eines Roboters spezielle Roboterkenntnisse und das Beherrschen der herstellerspezifischen Programmiersprache erforderlich. Das macht die Inbetriebnahme zeit- und kostenintensiv und den Einsatz erst bei hohen Stückzahlen und gleichbleibenden Aufgaben rentabel. Der Wunsch der Verbraucher nach individuellen Produkten hat die sogenannten Losgröße „Eins" zum Ziel. Um Robotersysteme auch für kleine Losgrößen wirtschaftlich betreiben zu können, setzen immer mehr Unternehmen auf Cobots.

Das Wachstum des Marktes von kollaborierenden Robotern wird weltweit zwischen 2016 und 2019 mit 13% vorausgesagt. Den größten Zuwachs wird dabei China verzeichnen, wo die Roboterdichte zurzeit bei 36 Einheiten pro 10.000 Mitarbeiter liegt. In Deutschland sind es zum Stand 2015 insgesamt 292 Roboter und in Korea 478 Einheiten pro 10.000 Mitarbeiter.[6] Damit ist der größte Zuwachs in China zu erwarten und wird mit mehr als 25% vorhergesagt. In Europa wird ein Wachstum von sechs Prozent vorhergesagt, wobei die Hauptkriterien für eine Anschaffung die einfache Bedienung und flexible Einsatzfähigkeit sind. Mit der Industrie 4.0 als treibenden Faktor wird der Einsatz von kollaborierenden Robotern in sogenannten Smart Factories in Europa und den USA dazu beitragen, dass Herstellungskosten sinken, die Qualität der Produkte steigt und kürzere Produktlebenszyklen realisiert werden können. Dem Wunsch der Kunden nach Individualität und die damit verbundene Variantenvielfalt der Produkte kann durch den Einsatz der flexiblen Robotereinheiten entsprochen werden[7]. Für diesen Zukunftsmarkt wird mit dieser Arbeit eine Grundlage geschaffen, um am IFT die weitere Forschung zum Thema kollaborative Robotik intensiv voranzutreiben.

[5] ISO10218-1 2012; ISO10218-2 2011; ISO/TS15066 2016

[6] International Federation Of Robotics 2016

[7] ebd.

1.1 Ausgangssituation und Ziel der Arbeit

Für die Lehrveranstaltung und Übung Industrieroboter besitzt das IFT als Beispiel für einen herkömmlichen Industrieroboter den ABB RB-120 (s. Abb.: 1.1) und für die Veranschaulichung der Eigenschaften eines kollaborativen Roboters den UR5 (s. Abb.: 1.2).

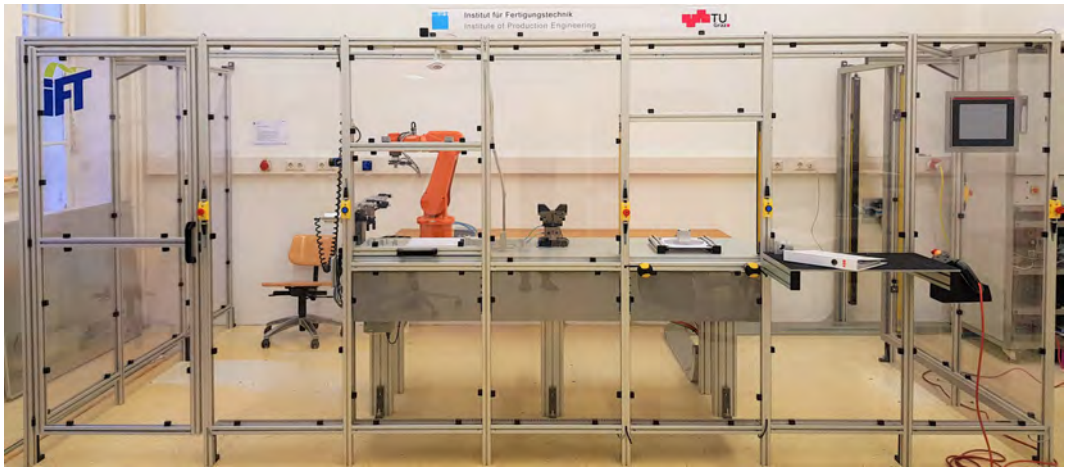

Abbildung 1.1: Roboterzelle mit ABB IRB-120

Da diese Systeme jedoch nicht quelloffen sind, ist ein detaillierter Einblick für die Studierende in die Funktionsweise der Roboter schwer bis gar nicht möglich. Die herstellerspezifische Software bietet zwar eine schnellen und einfache Möglichkeit den Roboter zu programmieren, lässt aber keinen Einblick in die Prozesse im Hintergrund zu. Auch der Aufbau des Roboters ist von außen schwer erkennbar, da eine Öffnung des Roboters mit dem Erlöschen der Garantieansprüche einhergeht.

Abbildung 1.2: UR-5 Cobot [8]

[8] Universal Robots A/S 2009

Im Rahmen eines studentischen Projektes ist bereits ein Getriebe für die Verwendung im Roboter entwickelt worden. In seiner Ausführung als Wellgetriebe ermöglicht dieses eine hohe Übersetzung von 50 in kompakter Bauform (s. Abb.: 1.3). Der Antrieb der Drehachsen soll mittels dieses Getriebes realisiert werden. Für die Montage sind bereits Gewindebohrungen am äußeren Durchmesser vorhanden. Der verwendete Motor lässt sich im Rahmen der Getriebeabmessungen den Bedürfnissen für die Verwendung im Roboter anpassen. Derzeit wird ein bürstenloser Gleichstrommotor in Außenläufer-Ausführung eingesetzt, welcher fest im Gehäuse verbaut ist.

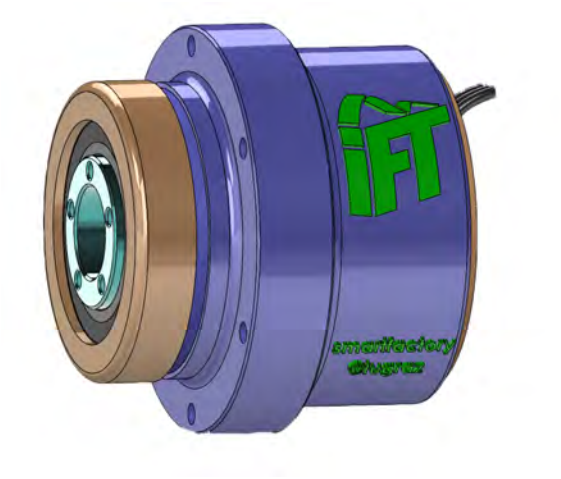

Abbildung 1.3: Wellgetriebe des IFT

Ziel der Arbeit ist es zunächst, eine Literaturrecherche über die aktuelle Marktsituation, die Voraussetzungen für den Betrieb eines Cobots und die mathematischen Grundlagen für die Berechnung der kinematischen Struktur durchzuführen.
Auf Basis der Erkenntnisse aus der Literaturrecherche ist ein Cobot zunächst zu konstruieren und anschließend eine Steuerung zu entwickeln, die eine Interaktion mit dem Roboters ermöglicht. Für den Betrieb als kollaborativen Roboter müssen Sicherheitsvorrichtungen verbaut werden, um eine Verletzung des Bedienenden zu verhindern und einen sicheren Betrieb ohne Schutzvorrichtung zu gewährleisten. Hierbei gilt es, eine Plattform zu schaffen, die zu einem späteren Zeitpunkt um zusätzliche Sensoren oder Aktoren ergänzt werden kann, um damit die Einsatzmöglichkeiten des Roboters zu erweitern. Um dieses Ziel kostengünstig zu erreichen, sind für den Bau des Roboters Standardkomponenten zu verwenden. Die Steuereinheit ist selbst zu entwerfen, um spätere Adaptionen zu ermöglichen.

1.2 Aufbau der Arbeit

Diese Masterarbeit gliedert sich in fünf Teile und beginnt mit einer Darstellung des aktuellen Standes der Technik. Dabei wird von der allgemeinen Definition von Industrierobotern ergänzend der Fokus auf kollaborative Roboter und deren Entwicklung im Laufe der Digitalisierung gelegt (vgl. Kapitel 2.1). Um im weiteren Verlauf eine Entscheidung über die Bauart und die zu implementierenden Komponenten treffen zu können, wird in Kapitel 2.2 ein Überblick über die verschiedenen Eigenschaften und Bauarten von kollaborativen Robotern gegeben, die zurzeit am Markt verfügbar sind, und es werden deren Einsatzgebiete dargestellt. Das Thema der Sicherheit beim Betrieb von Robotern zusammen mit Menschen im selben Arbeitsraum wird im darauffolgenden Kapitel 2.3 betrachtet. Die Vorteile, die der Einsatz eines Industrieroboters bietet, soll nicht mit einem erhöhten Verletzungsrisiko einhergehen. Daher müssen die Fähigkeiten des Roboters teilweise eingeschränkt werden, um einen sicheren Betrieb zu gewährleisten. Hierbei sind auch geltende Normen und internationale Richtlinien zu beachten, die den Einsatz von kollaborativen Robotern regeln. Um hierbei einen guten Kompromiss zwischen Sicherheit und Einsatzfähigkeiten zu finden, wird eine Basis geschaffen, die in Folge die bei der Entwicklung der Komponenten des Roboters und dessen Steuereinheit zu erfüllenden Bestimmungen aufzeigt. Als ein Teil der Aufgabenstellung ist ein bereits vorhandenes Wellgetriebe zu verbauen, was in Kapitel 2.4 vorgestellt wird. Dabei werden die Vorteile und Einsatzmöglichkeiten von Wellgetrieben dargestellt was zu einem besseren Verständnis für die Wahl dieser Konfiguration führen soll.
Als letzten Abschnitt des zweiten Kapitels werden die unterschiedlichen Möglichkeiten der Programmierung vorgestellt. Dabei wird zwischen online und offline Programmierungsarten unterschieden. Bei der Online-Programmierung wird der Bewegungsablauf direkt am Roboter einprogrammiert, wohingegen bei der Offline-Programmierung die Robotereinheit nicht benötigt wird.
Im Kapitel 3 werden die mathematischen Grundlagen für die Berechnung der Kinematik dargestellt. Die Beschreibung von Punkten im Raum sowie die Transformation verschiedener Koordinatensysteme sind für die spätere Berechnung der Gelenkwinkel notwendig. Dabei gilt es im weiteren Verlauf zwischen einer direkten und einer inversen Kinematik zu unterscheiden. Im direkten Fall sind die Gelenkwinkel gegeben und es muss die Position und Orientierung des Endeffektors berechnet werden. Im inversen Fall müssen nach einer gegebener Position des Endeffektors die Gelenkwinkel ermittelt werden. Um den Bewegungsverlauf des Roboters festzulegen, werden im Abschnitt 3.5 die verschiedenen Möglichkeiten zur Bewegungssteuerung dargestellt. Für die Erstellung der dynamischen Modells werden unter Abschnitt 3.6 zwei Verfahren der Berechnung vorgestellt.
Das vierte Kapitel widmet sich dem Entwurf des kollaborativen Roboters. Dies-

bezüglich werden unter Abschnitt 4.1 zunächst die Anforderungen an den Cobot dargestellt. Auf Basis dessen werden in Folge unter Abschnitt 4.2 die zu verwendenden Komponenten für den Antrieb sowie Elektronik und Software ausgewählt und vorgestellt. Um später günstige Weiterentwicklungen zu ermöglichen ist es sinnvoll, sich auf die Verwendung von systemoffenen Komponenten zu beschränken. Mit diesen Komponenten wurden die einzelnen Segmente des Roboters konstruiert. Im Abschnitt 4.3 werden die Segmente Schulter, Oberarm, Unterarm und Handgelenk vorgestellt. Diese wurden computerunterstützt entworfen (engl.: Computer-Aided Design Abk.: CAD) und konstruiert. Dabei wurden die zuvor genannten Anforderungen umgesetzt. Die Konstruktion liefert die Randbedingungen für die folgende Berechnung der Kinematik. Unter 4.4 wird mit der Vorstellung des Denavit-Hartenberg Verfahrens begonnen. Dieses wird für die Erstellung der Transformationsmatrizen des zuvor konstruierten Roboters benötigt. Mittels der Transformationsmatrizen lässt sich die Position des Endeffektors in der Vorwärtstransformation durch Eingabe der bekannten Gelenkwinkel berechnen. Um von einer bekannten Position auf die korrespondierenden Gelenkwinkel zu schließen, ist eine inverse Berechnung notwendig. Die gewählte Konfiguration des Roboters lässt hierbei eine analytische Lösung zu, welche unter Einführung von Randbedingungen zu einer spezifischen Lösung führt. Die Steuerung des Roboters steuert mit den zuvor berechneten Matrizen und Gleichungssystemen die Motoren an. Am Ende von Kapitel 4 werden die verbauten Sicherheitseinrichtungen vorgestellt und der Testaufbau für die Validierung der Steuerung vorgestellt.
Mit einer Zusammenfassung der Ergebnisse und einem Ausblick mit Anregungen für weiterführende Betrachtungen schließt die Masterarbeit.

2 Stand der Technik

Dieses Kapitel zeigt im ersten Abschnitt die Eigenschaften kollaborativer Roboter sowie deren Vorteile gegenüber konventionellen Robotern auf. Dabei werden einige aktuelle Modelle unterschiedlicher Bauarten vorgestellt.
Für den Einsatz von Cobots relevante Sicherheitsbestimmungen und die Möglichkeiten der Umsetzung werden in Abschnitt 2.3 dargelegt. Um das Moment der Motoren in kompakter Bauform übersetzen zu können, wird das zu verwendende IFT-Wellgetriebe und dessen Funktionsweise erklärt. Die Möglichkeiten bei der Programmierung eines Industrieroboters beschießen in Abschnitt 2.5 dieses Kapitel.

2.1 Kollaborative Roboter

Der herkömmliche Industrieroboter wird definiert als „Arbeitsmaschine, die zur selbsttätigen Handhabung von Objekten mit zweckdienlichen Werkzeugen ausgerüstet, in mehreren Bewegungsachsen hinsichtlich Orientierung, Position sowie Arbeitsablauf programmierbar ist.“ [1]
Die Beziehung zwischen Mensch und Roboter unterscheidet sich jedoch je nach Interaktionsgrad. In Abbildung 2.1 sind die verschiedenen Arten der Mensch-Roboter Kooperationen dargestellt.

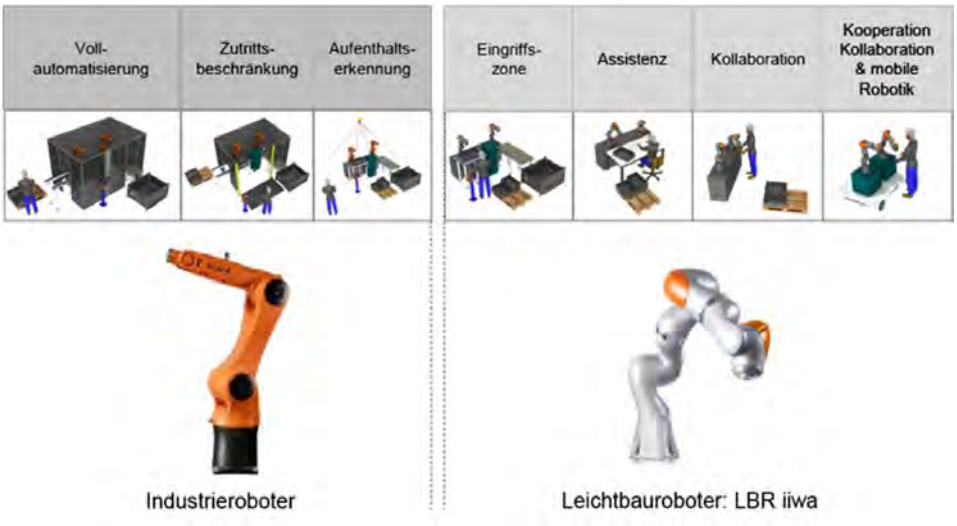

Abbildung 2.1: Stufen der Mensch–Roboter Kooperation [2]

[1] K.-H. Grote 2011, S. T97

Bei der vollautomatisierten Fertigung in einer abgeschlossenen Zelle ist der Zutritt für Menschen während des Betriebs nicht möglich. Hier kann der Roboter sein volles Potenzial ausschöpfen und mit maximalen Geschwindigkeiten und Beschleunigungen arbeiten, da die Verletzungsverhütung durch den abgeschirmten Betrieb sichergestellt ist. Der Bedienende des Roboters kann nur über einen Informationsaustausch mittels Computer mit dem Roboter interagieren. Ein direkter Kontakt ist nur bei Wartungs-/Einrichtungsarbeiten und unter besonderen Auflagen möglich.
Bei einer Roboterzelle mit Zutrittsbeschränkung kann der Arbeitsraum des Roboters während der Bearbeitung betreten werden. Durch Lichtvorhänge, Türen oder anderen Sensoren wird das Betreten registriert und löst einen Not-Stopp des Systems aus. Erst nach dem Verlassen des Arbeitsbereiches und einer Quittierung ist eine Fortsetzung des Betriebes möglich. So können zum Beispiel Werkstücke eingelegt oder entnommen und der Betrieb anschließend fortgesetzt werden.
Darüber hinaus gibt es die Möglichkeit, dass durch eine Aufenthaltserkennung ein Detektieren des Menschen stattfindet und die Geschwindigkeit des Roboters dem Abstand des Menschen zum Roboters bis hin zum Stillstand angepasst wird. So wird nicht unmittelbar ein Sicherheitsstopp ausgelöst und der Betrieb geht nach Verlassen des Menschen regulär weiter. Aber auch hier ist eine direkte Kollaboration zwischen Mensch und Maschine nicht möglich, denn ein direktes Interagieren mit dem Roboter wird durch die Steuerung unterbunden, um Verletzungen zu vermeiden.
Erst mit dem Einsatz von Leichtbaurobotern ist die Gefahr von Verletzungen unter Einsatz spezieller Sicherheitsvorkehrungen so gering, dass eine Interaktion mit dem Menschen möglich ist. Das geringere Gewicht des Roboters und die geringere Geschwindigkeiten reduzieren die wirkenden Kräfte. Zusammen mit konstruktiven Maßnahmen, wie abgerundete Kanten oder Polsterung wird das Verletzungsrisiko reduziert. Es ist eine ständige Präsenz des Menschen im Arbeitsraum des Roboters möglich. Der Roboter ist mit Sensoren ausgestattet, um die Folgen einer Kollision so gering wie möglich zu halten und im besten Fall ganz zu verhindern. Bei der Risikobewertung solcher Systeme ist aber nicht nur der Roboter, sondern auch das eingesetzte Werkzeug zu berücksichtigen. Bei der Verwendung von gefährlichen Werkzeugen kann es nötig sein, auch diese Roboter in Käfigen zu betreiben.
Die Einsatzmöglichkeiten von Leichtbaurobotern gehen von der Koexistenz zwischen Mensch und Roboter im selben Arbeitsraum, bis hin zur Kollaboration.
Erst wenn Roboter und Mensch gleichzeitig am selben Werkstück arbeiten, spricht man von einem kollaborativen Roboter. In diesem Szenario teilen sich Mensch und Roboter die Arbeit, wobei sich jeder auf seine Stärken konzentriert. Der Mensch bereichert den Arbeitsvorgang durch seine kognitiven und kreativen Fähigkeiten, der Roboter auf der anderen Seite steuert seine Kraft, Ausdauer und Präzision bei.

[2]Reichenbach 2013

Somit ist eine hohe Qualität der Produkte gewährleistet und der Mensch wird bei anstrengender und monotoner Arbeit entlastet.[3]

2.2 Verschiedene Bauarten und ihre Eigenschaften

In diesem Abschnitt werden einige auf dem Markt befindliche, Bauformen von kollaborativen Robotern vorgestellt. Es besteht kein Anspruch auf Vollständigkeit, jedoch werden die gängigsten Varianten abgedeckt, um einen Überblick über die aktuellen Möglichkeiten von kollaborativen Robotern zu bieten. Es werden jeweils die Besonderheiten der einzelnen Roboter und ihre Einsatzmöglichkeiten aufgezeigt, allerdings ohne detaillierte Betrachtung der technischen Daten.

2.2.1 7-Achs Gelenkarmroboter

Mit dem Leichtbauroboter (Abk.: LBR) IIWA 7 R800 (vgl. Abbildung 2.2) hat die Firma KUKA AG einen 7-Achs Knickarmroboter auf dem Markt, der für den kollaborativen Einsatz konstruiert wurde. Es sind keine Kanten am Gerät vorhanden, die zu Verletzungen führen könnten.

Abbildung 2.2: Kuka LBR IIWA[4]

Antriebe und Leitungen sind im Inneren der Robotereinheit verbaut, um ein Hängenbleiben zu verhindern. Die einzelnen Gelenkmodule sind aus Aluminium gefertigt und können ein Gewicht von 7 kg tragen. Durch sein geringes Eigengewicht von 23,9 kg und die geringe Beschleunigung der Motoren ergeben sich geringe Wirkkräfte. Die verbaute Kraftsensorik ermöglicht zum einen einen sicheren Betrieb in

[3] Alfons Botthof 2014, S. 59 ff
[4] KUKA Roboter GmbH 2016

Gegenwart von Menschen und zum anderen auch die Automatisierung von Montagearbeiten, bei denen „nach Gefühl" gearbeitet werden muss.[5] Für den Einsatz neben oder mit dem Menschen sind aber neben physischen Verletzungen auch psychische Verletzung durch Lärmemission zu vermeiden.[6] Mit einem Schalldruck von 75 dB unterschreitet der IIWA 7 R800 den Grenzwert von 85 dB, der in der Verordnung über Schutz der Arbeitnehmer/innen vor der Gefährdung durch Lärm und Vibrationen[7] festgesetzt ist und damit ist er für den Einsatz im kollaborativen Betrieb zugelassen.[8] Mit einer IP54 Zertifizierung[9] ist er gegen Staub und Spritzwasser geschützt und kann mit einer Positionswiederholungsgenauigkeit von $\pm 0,1mm$ innerhalb eines Arbeitsvolumen von $1,7\,m^3$agieren.[10] Er kann sowohl auf dem Boden als auch an Wand und Decke montiert werden.[11]

2.2.2 6-Achs Gelenkarmroboter

Mit einer Achse weniger bietet der in Abbildung 2.3 dargestellte UR5 von Universal Robots ähnliche Möglichkeiten wie der zuvor beschriebene KUKA Roboter. Die Traglast beträgt 5 kg bei einem Eigengewicht von 18,4 kg.

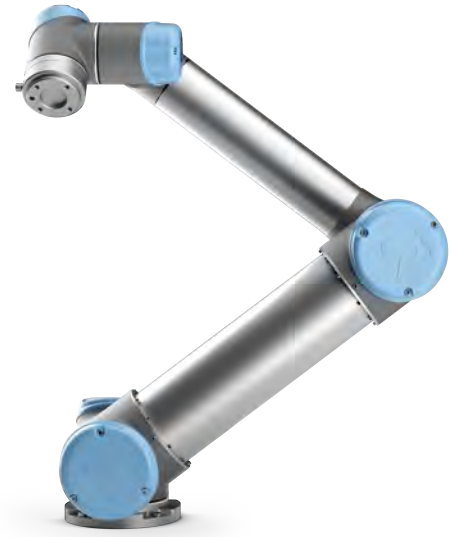

Abbildung 2.3: UR5 [12]

[5] KUKA Roboter GmbH 2016
[6] AschG o.D., §65
[7] VOLV-Lärm o.D., §3.1
[8] ISO10218-1 2012
[9] ISO20653 2013
[10] ISO9283 1998
[11] KUKA Roboter GmbH 2016
[12] Universal Robots A/S 2009

Das Chassis des UR5 ist aus Aluminium und PP-Plastik und frei von Kanten und außenliegenden Verkabelungen. Alle Achsen sind 360 Grad drehbar die den 120°-175° drehbaren Achsen des KUKA Roboters gegenüberstehen und erreichen eine Wiederanfahrgenauigkeit von 0.1 mm. Eine Kollision mit Objekten detektiert der UR5 ähnlich dem iiwa 7 kraft-sensitiv ab 50 N. Staub und Spritzwasserdichtigkeit wird nach IP54 gewährleistet. Die vom Hersteller versprochene einfache Bedienung wird wie beim Model von KUKA über ein Touchpanel unter Verwendung der herstellerspezifischen Software vorgenommen.[13]

2.2.3 Zweiarmroboter

Eine Kombination aus zwei Roboterarmen kommt bei dem Model YuMi IRB 1400 von ABB (siehe Abbildung 2.4) zum Einsatz. Bei einer maximalen Traglast von 0,5 kg pro Arm und einem Eigengewicht von 38 kg sowie einer IP30 Zertifizierung ist dieser Roboter nicht für feuchte Umgebungen und schwere Lasten geeignet. Der Vorteil liegt in der Genauigkeit des Roboters die mit einer Wiederanfahrgenauigkeit von 0.02 mm angegeben wird und dem Schutz gegen Elektrostatische Entladungen (engl.: electrostatic discharge Abk.: ESD).[14] Damit ist er für die Handhabung kleiner, sensibler Elektronikbauteile und deren exakte Positionierung geeignet.

Abbildung 2.4: ABB YuMi IRB 1400 [15]

Um die Sicherheit der Mitarbeiter zu gewährleisten, ist bei der Konstruktion darauf geachtet worden, Kanten zu vermeiden und die Leitungen innerhalb des Roboters zu verlegen. Zusätzlich können die Arme gepolstert werden und bieten damit zusätzlichen Schutz. Aufgrund dieser Maßnahmen und der geringen Traglast ist eine Geschwindigkeit von bis zu 1500 $\frac{mm}{sec}$ möglich.[16]

[13] Universal Robots A/S 2009
[14] DIN61340 2015
[15] ABB AG 2015
[16] ebd.

2.2.4 Mobiler Gelenkarmroboter

Als Beispiel für eine mobile Robotereinheit wird im Folgendem der flexible Produktionsassistent von Bosch (engl.: automated produktion assistant Abk.: APAS) vorgestellt. Er zeichnet sich durch einen modularen Aufbau und seine Flexibilität aus. Er besteht aus einer Plattform (APAS Base) und verschiedenen Aufbauten. Als Aufsatzmodule sind ein Gelenkarm (APAS assistant), eine Prüfvorrichtung (APAS inspector) sowie eine Fügevorrichtung (APAS flexpress) verfügbar. Weitere Module sind nach Kundenwünschen realisierbar. Für einen Vergleich wird im Weiteren der APAS assistant mobile (siehe Abbildung 2.5) betrachtet. Seine Traglast beträgt 7 kg bei mit einer Wiederholungsgenauigkeit von $\pm 0,03\,mm$ und einer Geschwindigkeit von $0,5\,\frac{m}{s}$ im kollaborativen Betrieb und bis zu $2,3\,\frac{m}{s}$ im fern-überwachten Betrieb.[17] Für die Positionierung ist ein Bildverarbeitungssystem verbaut, das aus 2D und 3D Kameras besteht und sich am Greifer befindet. Für die Verletzungsverhütung ist der Greifarm mit einer Sensorhaut überzogen, die Annäherungen von Menschen detektiert und den Roboter zum Stehen bringt, noch bevor es zur Kollision kommt.

Abbildung 2.5: APAS assistant mobile[18]

Die APAS Plattform ermöglicht einen schnellen und einfachen Ortswechsel der Einheit. Die Schnellfixierrollen an der mobilen Plattform gewährleisten zum einen eine einfach Verbringung der Einheit an den Einsatzort und zum anderen einen sicheren Stand. Die Steuerungskomponenten befinden sich im Inneren der Plattform, sodass für den Betrieb nur ein Strom und Datenanschluss notwendig ist und der Stellplatz für die Steuerung entfällt.[19]

[17] Robert Bosch GmbH 2015
[18] Robert Bosch Manufacturing Solutions GmbH 2017
[19] ebd.

2.3 Sicherheitsbestimmungen für den Einsatz von kollaborativen Robotern

Für den sicheren Betrieb von kollaborierenden Robotern im Fertigungsprozess sind besondere Sicherheitsbestimmungen einzuhalten. Die ISO 10218 regelt die Sicherheitsanforderungen an Robotern im Allgemeinen und die ISO/TS 15066 die Anforderungen an kollaborative Roboter im Speziellen. Für eine Risikobeurteilung des Robotersystems müssen die möglichen Gefahren zunächst identifiziert werden. Dafür bietet der Anhang A der ISO 10218-2 eine Übersicht über signifikante Gefährdungen, die sich aus der allgemeinen Maschinen Sicherheitsnorm ISO 12100 ableitet.[20] Dabei sind roboterbezogene, mit dem Robotersystem verbundene und anwendungsbezogene Gefährdungen einzuschließen.[21] Zusätzlich sind auch die Gefahren, die durch die Art der Endanwendung des Roboters auftreten, zu identifizieren. Diese sind in einem ersten Schritt „durch inhärente sichere Konstruktion oder deren Minderung durch Substitution"[22] zu beseitigen. Sollte dies nicht möglich sein, sind Schutzmaßnahmen, welche „das Erreichen einer Gefährdung durch Personen verhindert oder die Gefährdung beherrschen, indem sie in einen sicheren Zustand gebracht werden"[23] wie Geschwindigkeits-, Kraftbegrenzungen oder Anhalten vorzusehen. Dabei sind die in Tabelle A.2 der ISO/TS 15066 beschriebenen biomechanischen Grenzwerte zu beachten, um Verletzungen im Falle einer Kollision zu vermeiden. Dies kann auch durch zusätzliche angebrachte Schutzverkleidungen realisiert werden. Diese können die Kollisionsenergie abbauen und schützen den Bedienenden vor Verletzungen. Besteht weiterhin ein Restrisiko oder ist die Gefährdung durch die voranstehenden Maßnahmen nicht beseitigt worden, ist durch „Benutzerinformation, Schulung, Schilder, persönliche Schutzausrüstung usw."[24] auf die Gefahr hinzuweisen und so ein Bewusstsein beim Bedienenden des Roboters für die Gefahren zu schaffen.
Bei der Gestaltung des Roboters ist darauf zu achten, dass aktuelle Eigenschaften der Sicherheitseinstellungen vom Bedienpersonal einsehbar sind. Um im Betrieb des Roboters die Möglichkeit zu haben die Roboterbewegung jederzeit zu stoppen sind Not-Halt Einrichtungen zu installieren, die der ISO 13850 entsprechen.[25] Fluchtwege aus dem Kollaborationsraum sind hindernisfrei zu halten, um ein Verlassen des Gefahrenbereichs zu ermöglichen. Auch muss ersichtlich sein, wann sich der Roboter im kollaborierenden Betrieb befindet. Dies kann zum Beispiel durch eine visuelle Anzeige realisiert werden.[26] Für den Einrichtbetrieb ist ein manueller Modus mit re-

[20] ISO12100 2010
[21] ISO/TS15066 2016
[22] ebd., S. 4.3.4
[23] ebd., S. 4.3.4
[24] ebd., S. 4.3.4
[25] ISO13850 2015
[26] ISO10218-1 2012

duzierter Geschwindigkeit von nicht mehr als 250 $\frac{mm}{s}$ vorzusehen. Dabei ist darauf zu achten, dass in der Nähe von Singularitäten keine unerwartete Geschwindigkeitssteigerung stattfindet die für den Bedienenden überraschend eintritt und über die zulässige Geschwindigkeit von 250 $\frac{mm}{s}$ hinausgeht. Der Roboter muss vor dem Anfahren der Singularität mit gefährdenden Bewegungen einen Sicherheitsstopp auslösen.[27]
Sind alle relevanten europäischen Richtlinien eingehalten worden, kann der Hersteller dies mit der EU-Konformitätserklärung bestätigen und das CE-Kennzeichen anbringen. Dies ist jedoch kein Prüfsiegel einer unabhängigen Stelle sondern lediglich die Bestätigung des Herstellers, dass alle anzuwendenden EU-Richtlinien in Bezug auf das Produkt eingehalten wurden.[28]
Neben den internationalen Richtlinien sind auch länderspezifische Regulierungen hinsichtlich der Arbeitssicherheit zu berücksichtigen. In Österreich müssen beispielsweise die Grenzwerte bei Lärm und Vibrationen zum Schutz der Arbeitnehmer eingehalten werden.[29]

[27] ISO10218-1 2012
[28] Haas 2016
[29] VOLV-Lärm o.D.

2.4 Funktionsweise des IFT-Wellgetriebes

Die besonderen Eigenschaften des Wellgetriebes (siehe Abbildung 2.6) liegen in der Übertragung von hohen Drehmomenten bei sehr kompakter Bauform unter der Verwendung von wenigen Teilen.

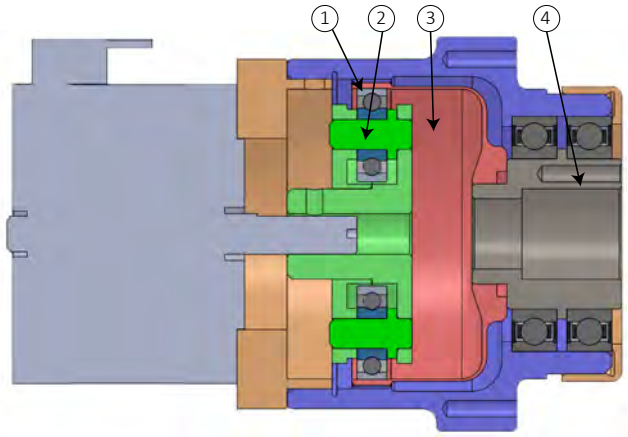

Abbildung 2.6: Wellgetriebe des IFT

Die nachstehende Abbildung 2.7 zeigt das Funktionsprinzip des Wellgetriebes. Die Bezeichnung Flexspline beschreibt ein hohles flexibles Bauteil (vgl. Abb.: 1.3 Nr. 1) mit Außenverzahnung, das auf einer elliptischen Scheibe mit konzentrischer Nabe, dem sogenannten Wave Generator (vgl. Abb.: 1.3 Nr. 2) gelagert ist. Der Antrieb erfolgt an der Nabe des Wave Generators mittels eines Motors und versetzt diesen in Rotation.

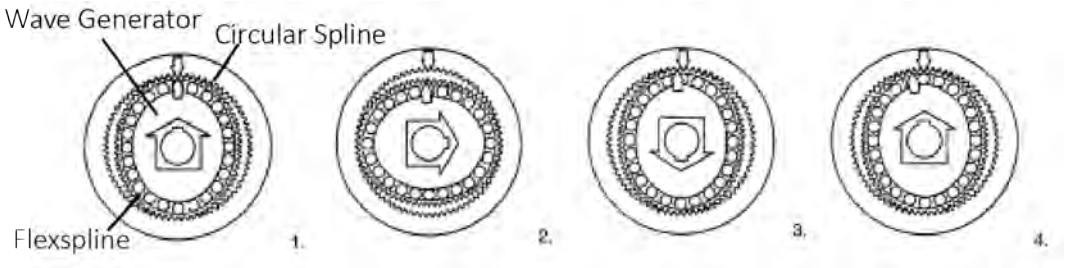

Abbildung 2.7: Funktionsprinzip des Wellgetriebes [30]

Durch die Rotation des Wave Generators verformt sich der Flexspline und greift in den runden innen-verzahnten Circular Spline (vgl. Abb.: 1.3 Nr. 3) mit jeweils einem Zahn auf zwei gegenüberliegenden Seiten ein. Der Flexspline besitzt zwei Zähne weniger als der Circular Spline und bewirkt damit bei einer halben Umdrehung eine Verschiebung des Circular Splines um einen Zahn. Durch das Zahnverhältnis von

[30] Slatter 2002

Flexspline und Circular Spline wird das Übersetzungsverhältnis beschrieben. Mit dieser Bauart sind Übersetzungsverhältnisse von 30:1 bis 320:1 bei einem Geriebe-Außendurchmesser von 20 bis 30 mm möglich.[31] Aus der hohen Übersetzung resultieren folgende Eigenschaften:[32]

- hohe Positioniergenauigkeit
- hohe Drehmomentkapazität
- hohe Torsionssteifigkeit
- guter Wirkungsgrad von bis zu 85%
- Selbsthemmung
- lange Lebensdauer

Dank der Eigenkonstruktion des Getriebes am IFT sind Anpassungen an unterschiedliche Anwendungsfälle schnell umsetzbar. Anschlussmöglichkeiten und Abmessungen sowie Übersetzungen lassen sich anpassen.

[31] Slatter 2002
[32] ebd.

2.5 Programmiermöglichkeiten

Für eine Interaktion zwischen Mensch und Maschine bietet die Programmierung des Roboters die Möglichkeit, Bewegungen und Abläufe durch geeigneter Befehle zu veranlassen. Dies kann entweder online oder offline erfolgen. Im Folgenden werden die Unterschiede und Möglichkeiten der zwei Varianten dargestellt.

2.5.1 Online-Programmierung

Diese Art der Programmierung findet direkt am Roboter statt und wird daher auch teilweise „direkte Programmierung" genannt. Der Roboter wird manuell in die gewünschte Position oder entlang der gewünschten Bahn geführt und speichert die Zielpunkte beziehungsweise den Bahnverlauf ab. Die Kommunikation mit dem Roboter während des Programmierer geschieht über ein Handprogrammiergerät, das Eingaben ermöglicht. Entweder werden die Motoren des Roboters angesteuert und bewegen diesen in der gewünschten Weise oder der Programmierende bringt den Roboter bei abgeschalteten Motoren durch seine Muskelkraft in Position. Bei beiden Varianten werden beim Abspeichern der aktuellen Position die aktuellen Winkelstellungen der Motoren gespeichert.[33]

2.5.1.1 Teach-In Verfahren

Durch das portable Handprogrammiergerät wird der gewünschte Punkt angefahren und die Winkelstellungen der einzelnen Gelenke abgespeichert. Damit ist gewährleistet, dass der Roboter exakt diese Position in der geforderten Orientierung wieder anfahren kann. Der Weg zwischen den einzelnen eingespeicherten Punkten wird wie in Kapitel 3.5 errechnet.[34]

2.5.1.2 Lead-Trough Verfahren

Lassen sich die Bewegungen des Roboters aufgrund ihrer geforderten Komplexität nur schwer durch das Anfahren einzelner Punkte bestimmen, bietet das Lead-Through Verfahren eine Alternative zum Teach-In Verfahren. Meistens wird dieses Verfahren für kontinuierliche Pfade genutzt, wie sie zum Beispiel beim Lackieren oder Schweißen vorkommen.[35] Dabei wird der Effektor vom Programmierenden händisch entlang der geforderten Bahn geführt. In definierten, kurzen Zeitintervallen werden die Gelenkstellungen von der Steuerung abgespeichert. Es kann auch die Geschwindigkeit mit abgespeichert werden, um Prozessparameter beim Lackieren oder Schweißen einzuhalten.

[33] Nof 1999, S.337
[34] Weber 2017, S.107
[35] Nof 1999, S.348

2.5.1.3 Master-Slave Verfahren

Dieses Verfahren überträgt die Bewegungen einer Master-Einheit auf eine Slave-Einheit. Eingesetzt wird dieses Verfahren hauptsächlich bei der Handhabung in gefährlichen Umgebungen. Die Slave-Einheit befindet sich örtlich von der Master-Einheit entfernt und ist mit dieser über eine Steuerung verbunden. Der Bediener gibt eine direkte Bewegungsvorgabe an der Master-Einheit die zeitgleich von der Slave-Einheit ausgeführt wird. Dafür ist Sichtkontakt notwendig und ein Speichern der Abläufe ist nicht notwendig. Daher zählt dieses Verfahren streng genommen nicht zu den Programmierverfahren, da die Sequenzen nicht wiederholt werden.[36]

2.5.2 Offline Programmierung

Bei der Offline Programmierung wird das Programm ganz oder teilweise entwickelt, ohne dass der Roboter verwendet werden muss. Dies ermöglicht eine höhere produktive Auslastung des Roboters in der Produktion, da Standzeiten für die Programmierung entfallen. Darüber hinaus bietet dieses Verfahren die Möglichkeiten einer präziseren Positionierung, die Einbeziehung von Sensoren und eine sichere Umgebung für den Programmierenden. Die programmierten Abläufe können vor der Übertragung auf den Roboter im Computer simuliert werden, um eventuelle Fehler im Programm frühzeitig zu erkennen.[37]

2.5.2.1 Textbasierte Programmierung

Bei der textbasierten Programmierung lässt sich der Roboter mittels vorgegebener Befehle programmieren. Diese können in der einfachsten Variante in einem Texteditor in der Reihenfolge ihrer Abarbeitung geschrieben und mit den erforderlichen Parametern versehen werden. Um allerdings Fehler in der Syntax und Semantik zu vermeiden, kann auf spezielle Editoren der Hersteller zurück gegriffen werden. Diese erkennen Fehler und bieten darüber hinaus auch unterstützende Funktionen, um die Programmierung zu vereinfachen.

2.5.2.2 CAD-basierte Programmierung

Durch die Abbildung der kompletten Roboterzelle bietet die CAD-basierte Programmierung eine Möglichkeit das Programm in der virtuellen Umgebung zu simulieren. Dabei können Kollisionen mit der Zelle verhindert werden und die Ausführung des Programms kann im Vorfeld simuliert werden. Dadurch reduziert sich die Nacharbeit direkt am Roboter und damit dessen Stehzeit. Die Visualisierung erleichtert es dem Programmierer eine Vorstellung seiner Arbeit zu erhalten.[38]

[36] Weber 2017, S.110
[37] Nof 1999, S.353
[38] Hesse und Malisa 2016, S.250

3 Kinematisches Grundlagen

In diesem Kapitel soll die Grundlage geschaffen werden, um eine Robotersteuerung zu realisieren. Dafür werden zunächst die möglichen Lagebeschreibungen eines Punktes im Raum und die vollständige Beschreibung der Lage des Roboterarms erklärt. Dabei werden die Verknüpfungen zwischen den einzelnen Koordinatensystemen der verschiedener Komponenten des Roboters mithilfe von Transformationsmatrizen beschrieben. Für die spätere Verwendung ist die Einführung von Welt-, Greifer-, Werkstück und anderen Koordinatensystemen sinnvoll. Mithilfe von Transformationen sollen diese ineinander übergeführt werden. Im Weiteren werden Möglichkeiten der Interpolation von Bewegungen des Roboters dargestellt und die damit verbundene Programmierung der Bewegungsabläufe.

3.1 Lagebeschreibung

Zur Lagebeschreibung werden kartesische Koordinatensysteme verwendet und die Gültigkeit der Rechtsschraubenregel vorausgesetzt. Damit lässt sich durch Vektoren jeder Punkt im Raum beschreiben. Solange die Vektoren im selben Koordinatensystem beschrieben sind, lassen sie sich nach den Regeln der Vektorenrechnung miteinander durch Operatoren verknüpfen.
Dabei ist zwischen freien und Ortsvektoren zu unterscheiden. Um physikalische Größen mit einem Betrag und einer Richtung zu beschreiben werden freie Vektoren verwendet. Die Länge des Vektors beschreibt dabei den Betrag der Größe und die Orientierung beschreibt die Richtung. Diese Vektoren können „frei" im Raum verschoben werden ohne dabei Richtung oder Betrag zu ändern. Ein Ortsvektor beginnt immer im Ursprung des Koordinatensystems und kann jeden beliebigen Raumpunkt beschreiben.
Der in Abbildung 3.1 dargestellte Ortsvektor $\vec{r}_P$ beschreibt die Lage des Punktes P im Raum eindeutig. Für die Entwicklung einer Kinematik der Robotersteuerung werden Ortsvektoren für die Beschreibung der Lage der verschiedenen Koordinatensysteme zueinander beschrieben.[1]

[1] Weber 2017, S.33

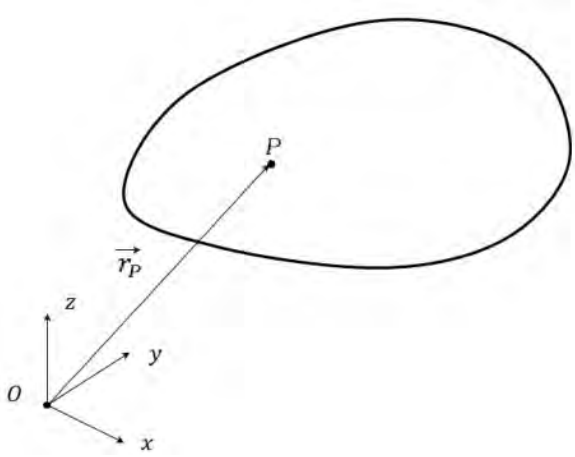

Abbildung 3.1: Beschreibung der Lage im Raum

3.2 Transformation der Koordinatensysteme

Bei einem Roboter besitzt jeder Antrieb sein eigenes Koordinatensystem. Um den Endeffektor frei im Raum zu bewegen, muss diese Bewegung durch eine Transformation auf die verschiedenen Antriebe umgerechnet werden. In Abbildung 3.2 sind zwei Koordinatensysteme dargestellt. Um einen Vektor des Bezugsystem {B} mit einem des Referenzsystems {R} kombinieren zu können, ist dieser zunächst auf {R} umzuformen. Dafür stehen ein Vektor vom Ursprung von {R} zu {B} und eine Drehmatrix zur Verfügung. Mit Hilfe dieser beiden Komponenten lässt sich der Vektor aus {B} eindeutig im {R} System ausdrücken.

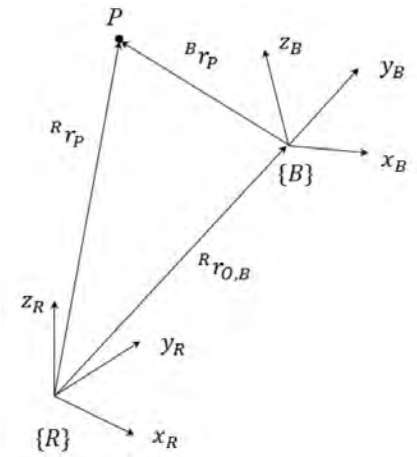

Abbildung 3.2: kartesisches Koordinatensystem eines starren Körpers [2]

Sollten beide Koordinatensysteme die gleiche Orientierung aufweisen ist eine Drehmatrix nicht zu berücksichtigen. Der Ortsvektor ${}^{R}\vec{r}_{O,B} = \left\{ {}^{R}x \;\middle|\; {}^{R}y \;\middle|\; {}^{R}z \right\}^{T}$ be-

[2] Wloka 1992, S.73

schreibt dabei die Lage des Ursprungs von {B} im {R} System. Damit lässt sich durch Addition mit ${}^{B}\vec{r}_{P}$ der Punkt P im {R} System beschreiben und entspricht damit ${}^{R}\vec{r}_{P}$. Sollte die Orientierung sich jedoch wie in Abbildung 3.2 unterscheiden wird dies in einer Drehmatrix dargestellt. Diese besteht aus einer Kombination von den drei nachstehend aufgeführten Drehmatrizen der einzelnen Raumrichtungen. Sollte das Koordinatensystem um mehr als eine Raumachse verdreht sein, werden die Matrizen durch eine Multiplikation zu einer Gesamtdrehmatrix zusammengefasst.

$$R_x(\alpha) = \begin{Bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{Bmatrix} \tag{3.1}$$

$$R_y(\beta) = \begin{Bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{Bmatrix} \tag{3.2}$$

$$R_z(\gamma) = \begin{Bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{Bmatrix} \tag{3.3}$$

Dabei beschreibt der Winkel α die Verdrehung des Bezugssystems gegenüber des Referenzsystems um die x-Achse, analog β und γ Drehungen um y-und z-Achse. Die Transformation von {B} nach {R} bildet einen Vektor aus {B} in {R} ab und wird mit ${}^{R}_{B}\vec{T}$ beschrieben. Durch eine Multiplikation mit den Drehmatrizen und einer anschließenden Addition mit ${}^{R}\vec{r}_{O,B}$ lässt sich ${}^{B}\vec{r}_{P}$ im {R} System darstellen und entspricht damit ${}^{B}\vec{r}_{P}$. Mit der Gesamttransformations-Matrix lassen sich die Verschiebungen und Verdrehungen zusammenfassen.[3]

$${}^{R}_{B}\vec{T} = \begin{Bmatrix} b_{11} & b_{12} & b_{13} & r_{O,B1} \\ b_{21} & b_{22} & b_{23} & r_{O,B2} \\ b_{31} & b_{32} & b_{33} & r_{O,B3} \\ 0 & 0 & 0 & 1 \end{Bmatrix} \tag{3.4}$$

Dabei beschreiben b_{ij} die Orientierung des Koordinatensystems und $r_{O,Bi}$ dessen Lage. Für die inverse Transformation von {R} nach {B} ${}^{B}_{R}\vec{T}$ wird die Matrix invertiert. Da bei einem Roboter die geforderte Bewegung für den Endeffektor angegeben wird und sich alle anderen Bewegungen der verschiedenen Achsen daraus ableiten, spricht man von einer inversen Kinematik.[4]

[3] Weber 2017, S.58
[4] Haas 2016, S.41

3.3 Direkte Kinematik

Bei der Vorwärtstransformation ist die Stellung der einzelnen Koordinatensysteme zueinander von vornherein festgelegt und es gilt die Position des Endeffektors zu berechnen. Hierfür kann die Denavit-Hartenberg-Transformation genutzt werden. Diese beschreibt durch zwei Translationen und zwei Rotationen eine Transformations-Matrix. Die Parameter werden für jedes Gelenk definiert. Dabei beschreibt φ den Winkel um die z_{n-1}- Achse, d den Abstand entlang der z_{n-1}- Achse, a den Abstand entlang der x_{n-1}- Achse und α den Winkel um x_n. Ausgang ist eine definierte Nullstellung des Roboters. Damit lassen sich anschließend mit den Matrizen $\vec{\vec{R}}_{z_{i-1}}$die Rotation um die z_{i-1}Achse und $\vec{\vec{T}}_{z_{i-1}}$die Translation entlang der z_{i-1}Achse beschreiben. Dies gilt analog für Rotation und Translation um die x_i-Achse.

$$\vec{\vec{R}}_{z_{i-1}}(\varphi_i) = \begin{Bmatrix} cos(\varphi_i) & -sin(\varphi_i) & 0 & 0 \\ sin(\varphi_i) & cos(\varphi_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix} \tag{3.5}$$

$$\vec{\vec{T}}_{z_{i-1}}(d_i) = \begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{Bmatrix} \tag{3.6}$$

$$\vec{\vec{T}}_{x_i}(a_i) = \begin{Bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix} \tag{3.7}$$

$$\vec{\vec{R}}_{x_i}(\alpha_i) = \begin{Bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sin(\alpha_i) & 0 \\ 0 & sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{Bmatrix} \tag{3.8}$$

Dabei befindet sich das Koordinatensystem i näher am Endeffektor als das Koordinatensystem $i-1$. Es wird außerdem ein Weltkoordinatensystem K_0 festgelegt. Dieses ist feststehend und befindet sich in der Basis des Roboters. Die anderen Koordinatensysteme befinden sich in den Gelenken. Die Denavit-Hartenberg-Matrix

${}^{n-1}_{n}\vec{\vec{T}}$ ist eine Kombination aus den vier Matrizen 3.5-3.8.

$$
\begin{aligned}
{}^{n-1}_{n}\vec{\vec{T}}(\varphi_i, d_i) &= \vec{\vec{R}}_{z_{i-1}}(\varphi_i) \cdot \vec{\vec{T}}_{z_{i-1}}(d_i) \cdot \vec{\vec{T}}_{x_i}(a_i) \cdot \vec{\vec{R}}_{x_i}(\alpha_i) \\
&= \begin{Bmatrix}
cos(\varphi_i) & -sin(\varphi_i) \cdot cos(\alpha_i) & sin(\varphi_i) \cdot sin(\alpha_i) & a_i \cdot cos(\varphi_i) \\
sin(\varphi_i) & cos(\varphi_i) \cdot cos(\alpha_i) & -cos(\varphi_i) \cdot sin(\alpha_i) & a_i \cdot sin(\varphi_i) \\
0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\
0 & 0 & 0 & 1
\end{Bmatrix}
\end{aligned}
\tag{3.9}
$$

Durch die Multiplikation der Matrizen in der Reihenfolge ihrer Entfernung vom Endeffektor berechnet sich die Lage des Tool Center Points (Abk.: TCP).[5]

$$
\vec{\vec{X}}_{TCP}(\vec{\varphi}, \vec{d}) = \prod_{i=1}^{n} {}^{n-1}_{n}\vec{\vec{T}}(\varphi_i, d_i) \tag{3.10}
$$

[5] Haas 2016, S.51

3.4 Inverse Kinematik

Bei der Rückwärtstransformation ist die Stellung der einzelnen Koordinatensysteme zueinander nicht festgelegt. Die Lage des Endeffektors $\vec{X} = \begin{Bmatrix} x & y & z & \phi & \theta & \psi \end{Bmatrix}^T$ ist bekannt und soll durch die Gelenkstellung $\vec{q} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6,)$ erreicht werden. Da der gleiche Punkt oft durch mehrere verschiedenen Gelenkstellungskombinationen erreicht werden kann (vgl. Abbildung 3.3), können sich Mehrdeutigkeiten (Singularitäten) ergeben.

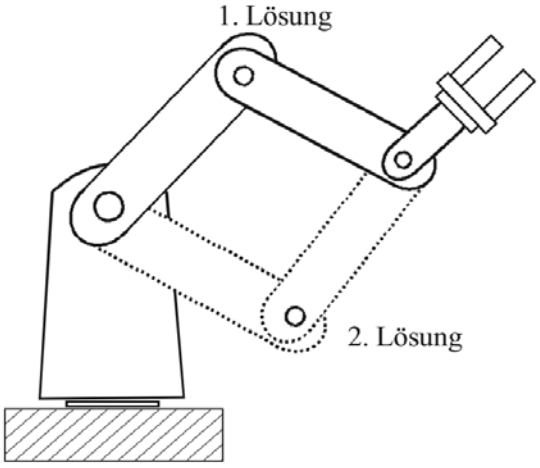

Abbildung 3.3: Beispiel für Mehrdeutigkeit [6]

Bei einer Singularität (vgl. Abbildung 3.4) besitzen zwei Achsen die selbe Orientierung, wodurch das System einen Freiheitsgrad verliert und singulär wird.

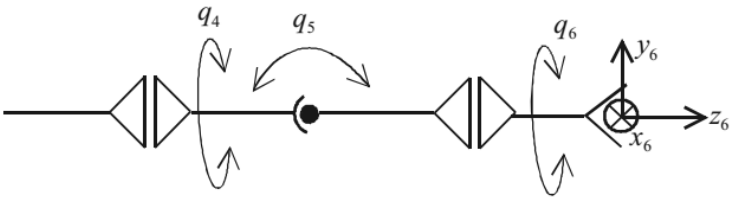

Abbildung 3.4: Beispiel für eine Singulariät [7]

Mittels analytischen, numerischen oder Näherungsverfahren lassen sich die Achskoordinaten ermitteln. Dabei ist zu beachten, dass nicht alle Stellungen der Gelenke zulässig sind. Es kann zu Kollisionen mit dem Roboter selber oder seiner Umgebung kommen oder die Mechanik des Roboters lässt gewisse Winkel nicht zu.

[6] Weber 2017, S.59
[7] ebd., S.59

3.4.1 Ansatz von Paul

Um die inverse Kinematik mit dem Ansatz von Paul zu lösen werden zunächst die Matrizen ${}^{n-1}_{n}\vec{T}$ aufgestellt um 3.10 zu erreichen. Da bei herkömmlichen 6-Achsrobotern meist die Verschiebung des TCP innerhalb des Weltkoordinatensystems benötigt wird, soll dieser Fall näher betrachtet werden. Durch elementweises Gleichsetzen von 3.10 erhält man für sechs Achsen 12 nichttriviale Gleichungen:

$$ {}^{0}_{6}\vec{T} = \vec{X}_{TCP} $$

wird

$$
\begin{aligned}
{}^{0}_{6}\vec{T} &= {}^{0}_{1}\vec{T} \cdot {}^{1}_{2}\vec{T} \cdot {}^{2}_{3}\vec{T} \cdot {}^{3}_{4}\vec{T} \cdot {}^{4}_{5}\vec{T} \cdot {}^{5}_{6}\vec{T} \\
{}^{1}_{0}\vec{T} \cdot {}^{0}_{6}\vec{T} &= {}^{1}_{2}\vec{T} \cdot {}^{2}_{3}\vec{T} \cdot {}^{3}_{4}\vec{T} \cdot {}^{4}_{5}\vec{T} \cdot {}^{5}_{6}\vec{T} \\
{}^{2}_{1}\vec{T} \cdot {}^{1}_{0}\vec{T} \cdot {}^{0}_{6}\vec{T} &= {}^{2}_{3}\vec{T} \cdot {}^{3}_{4}\vec{T} \cdot {}^{4}_{5}\vec{T} \cdot {}^{5}_{6}\vec{T} \\
{}^{3}_{2}\vec{T} \cdot {}^{2}_{1}\vec{T} \cdot {}^{1}_{0}\vec{T} \cdot {}^{0}_{6}\vec{T} &= {}^{3}_{4}\vec{T} \cdot {}^{4}_{5}\vec{T} \cdot {}^{5}_{6}\vec{T} \\
{}^{4}_{3}\vec{T} \cdot {}^{3}_{2}\vec{T} \cdot {}^{2}_{1}\vec{T} \cdot {}^{1}_{0}\vec{T} \cdot {}^{0}_{6}\vec{T} &= {}^{4}_{5}\vec{T} \cdot {}^{5}_{6}\vec{T} \\
{}^{5}_{4}\vec{T} \cdot {}^{4}_{3}\vec{T} \cdot {}^{3}_{2}\vec{T} \cdot {}^{2}_{1}\vec{T} \cdot {}^{1}_{0}\vec{T} \cdot {}^{0}_{6}\vec{T} &= {}^{5}_{6}\vec{T}
\end{aligned}
$$

Aus den entstandenen Gleichungen wird eine Gleichung oder eine Kombination aus Gleichungen gewählt, die sich nach einer Gelenkvariablen auflösen lässt. Damit vereinfacht sich das Gleichungssystem und so lassen sich sukzessiv alle Werte errechnen. Dabei ist eine Verwendung von *arccos* und *arcsin* aufgrund ihres periodischen Verlaufs zu vermeiden, um Mehrdeutigkeiten zu verhindern. Als Alternative ist der *arctan* zu verwenden.[8]

3.4.2 Numerische Variante

Bei der numerischen Variante ergeben die momentanen Werte der Gelenkparameter die aktuelle Position. Und solange der Abstand $\overrightarrow{e}$ von der gewünschten Position $\vec{t}$ von der tatsächlichen Position $\vec{s} = \vec{s}(\theta)$ zu groß ist wird ein $\triangle\Theta$ berechnet welches $t \approx s(\Theta + \triangle\Theta)$ liefert. Um $\triangle\Theta$ zu berechnen ergeben sich verschiedene iterative Möglichkeiten. Ist die Funktion $\overrightarrow{s}$ linear und die Jakobi-Matrix quadratisch und nicht singulär kann $\triangle\Theta$ über

$$
\begin{aligned}
\triangle\Theta &= J^{-1} \cdot \vec{e} \\
\overrightarrow{e} &= \frac{\partial s}{\partial \theta} \cdot \Delta\theta \\
&= J(\theta) \cdot \Delta\theta
\end{aligned}
$$

[8] Jörg Wollnack 2007, S. 3-16

berechnet werden.[9]
Bei Verwendung des *Jacobian-Transpose-Verfahrens* wird $\triangle\Theta$ aus der Multiplikation eines Dämpfungsfaktors α mit der transponierten Jakobi-Matrix und dem Abstand $\overrightarrow{e}$ von $\overrightarrow{s}$ und $\overrightarrow{t}$.

$$\triangle\Theta = \alpha \cdot J^T \cdot \vec{e}$$

Dieses Verfahren führt zu gleichmäßigen und glatten Bewegungen, konvergiert aber sehr langsam gegen die gewünschte Zielposition.[10]
Anstelle der Inversen der Jakobi-Matrix ist die *Pseudo-Inverse* auch für singuläre und nicht-quadratische Matrizen definiert.

$$\triangle\Theta = \alpha \cdot J^\dagger \cdot \vec{e}$$

Dabei kann es in der Nähe von Singularitäten zu Stabilitätsproblemen kommen. Durch den Verlust des vollen Ranges der Jakobi-Matrix treten Singularitäten auf. Direkt in der Singularität ist die Pseudo-Inverse definiert. Allerdings treten in der Nähe von Singularitäten bei kleinen Änderungen von $\overrightarrow{e}$ große Änderungen von $\triangle\Theta$ auf.[11]
Dies kann mit der *Damped-Least-Squares* Methode vermieden werden. Hierbei berechnet sich $\triangle\Theta$ durch:

$$\triangle\Theta = \underset{\triangle\Theta}{min} = \|J\triangle\Theta - \overrightarrow{e}\|^2 + \lambda^2 \|\triangle\Theta\|^2$$

Die Dämpfungskonstante λ soll die Stabilität in der Nähe von Singularitäten gewährleisten. Wird λ allerdings zu groß gewählt, kommt es zu einer langsamen Konvergenzgeschwindigkeit. Diese Methode ist stabil aber sehr rechenaufwendig, da nach jedem Schritt die Jakobi-Matrix erneut berechnet werden muss.[12]
Bei der Anwendung des *Cyclic Coordinaten Desent* Verfahrens wird durch die Betrachtung aller Gelenke versucht, den Zielfunktionswert hinreichend genau zu erreichen. Dabei werden in einem Iterationsschritt alle Gelenke nacheinander verändert, um den Abstand $\overrightarrow{e}$ in jedem Schritt zu minimieren. Da dies ein heuristisches Verfahren ist, kann keine Aussage über die Konvergenzgeschwindigkeit getroffen werden. Diese Methode ist auch in der Nähe von Singularitäten stabil, da immer nur ein Wert geändert wird. Im Ergebnis führt diese Methode zu ruckartigen Bewegungen.[13]
Als letzte Variante wird kurz die *Lagrange-Multiplikation* vorgestellt. Hierbei wird aus der Lagrangefunktion ein Optimierungsproblem, in diesem Fall die Suche nach

[9] Steidl 2011, S.6
[10] ebd., S.7
[11] ebd., S.8
[12] ebd., S.8
[13] ebd., S.9

einem Extrema einer Funktion unter Berücksichtigung von Nebenbedingungen. Als Nebenbedingung gilt bei der Suche nach den Gelenkwinkeln:

$$J\triangle\Theta = \overrightarrow{e}$$

Duch das Hinzufügen von Nebenbedingungen werden die einzelnen Gleichungen linear abhängig und reduzieren sich auf ein Problem ohne Nebenbedingungen, das beispielweise mit dem Gradientenverfahren gelöst werden kann.[14]

3.4.3 Analytisches Verfahren

Hiemit lassen sich einfache Systeme gut lösen. Unter Zuhilfenahme von zum Beispiel dem Kosinussatzes lassen sich die Gelenkparameter θ ermitteln.

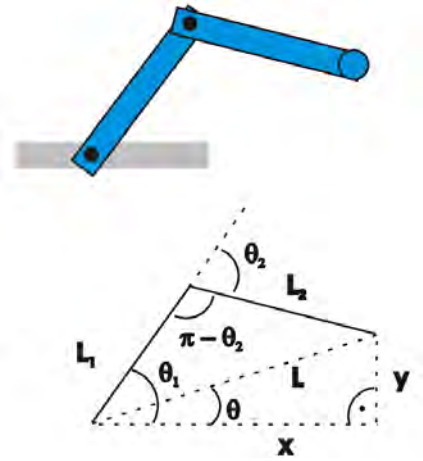

Abbildung 3.5: Arm mit zwei Gelenken[15]

Daraus lassen sich die Winkel wie folgt berechnen:

$$\theta_1 = cos^{-1} \cdot \left(- \frac{-L_2^2 + L_1^2 + L^2}{2L_1 L} \right) + \theta + 2\pi n \quad für\, n\epsilon\mathbb{Z}$$

$$\theta_2 = cos^{-1} \cdot \left(- \frac{-L^2 + L_1^2 + L_2^2}{2L_1 L_2} \right) - \pi n \qquad für\, n\epsilon\mathbb{Z}$$

$$L = \sqrt{x^2 + y^2}$$

$$\theta = cos^{-1} \frac{x}{L}$$

Diese Methode ist allerdings auf einfache Systeme beschränkt und wird mit zunehmenden Freiheitsgraden unpraktikabel. Wenn sich die Komplexität allerdings durch Randbedingungen reduzieren lässt, ist eine analytische Lösung in einigen Fällen möglich.

[14] Arens u. a. 2015, S.1311
[15] Steidl 2011, S.6

3.5 Interpolation von Bewegungen

Um die berechneten Punkte im Raum zu erreichen, muss ein Weg zwischen der aktuellen Position und der gewünschten interpoliert werden. Es werden die zwei gängigsten Arten der Steuerung vorgestellt.

3.5.1 Punktsteuerung

Um den Zielpunkt zu erreichen werden bei der Punktsteuerung (engl. Point to Point, Abk.: PTP) die Zwischenpunkte in den Achskoordinaten berechnet, dies zieht eine Bewegung entlang einer beliebigen Bahn nach sich. Sie ist zum Anfahren einer Position geeignet wenn der Weg dahin nicht von Belangen ist. Bei einer nicht synchronisieren Steuerung fahren alle Achsen ihre Endstellung ohne Rücksicht auf die anderen Achsen an und beenden damit ihre Bewegung zu unterschiedlichen Zeiten. Dies wird bei einer synchronisierten Steuerung (vgl. Abb. 3.6) durch einen abgestimmten Ablauf durch die individuelle Anpassung von Beschleunigungs- und Geschwindigkeitswerten verhindert.

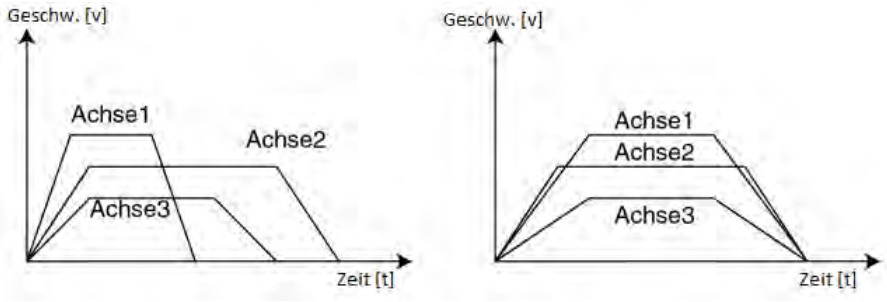

Abbildung 3.6: Links: Asynchrone PTP-Steuerung, Rechts: Synchrone PTP-Steuerung [16]

3.5.2 Bahnsteuerungen

Bei vielen Anwendungen ist es aber notwendig, einen definierten Weg zurückzulegen. Meistens ist gefordert, dass sich der Endeffektor auf einer festgelegten Bahn bewegt (continuous path Abk.: CP) Dafür folgen die Beschleunigungs- und Geschwindigkeitsprofile der Gelenke keinen gleichmäßigen Mustern mehr sondern richten sich nach dem geforderten Verlauf des Endeffektors.[17]

Linearinterpolation Um das Werkzeug mit dem TCP entlang einer Geraden zu verschieben, wird die Wegstrecke s_{AB} über:

[16] Wüst 2014, S.22
[17] ebd., S.19

$$s_{AB} = |p_A - p_B| = \sqrt{(p_{B,x} - p_{A,x})^2 + (p_{B,y} - p_{A,y})^2 + (p_{B,z} - p_{A,z})^2} \tag{3.11}$$

berechnet. Diese beschreibt, wie in Abbildung 3.7 dargestellt den Weg vom Punkt A zum Zeitpunkt $t = t_A = 0$ Punkt B zur gewünschten Endzeit $t_e = t_B$.

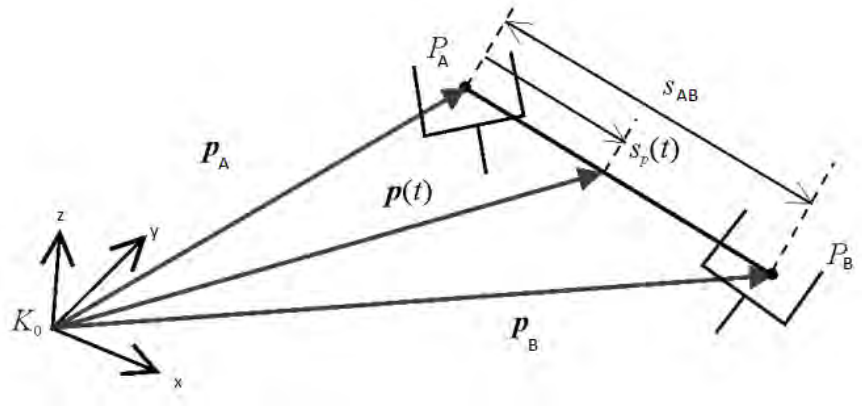

Abbildung 3.7: Berechnung der Wegstrecke zwischen A und B [18]

Mit $p(t)$ können die Zwischenwerte des Weges wie folgt berechnet werden:

$$p(t) = p_A + s_p(t) \cdot \frac{p_B - p_A}{s_{AB}} \tag{3.12}$$

Über $s_p(t)$ wird interpoliert mit der Bedingung, dass am Anfang und am Ende die Geschwindigkeit null ist.[19]

$$\begin{aligned} s_p(0) &= \dot{s}_p(0) = v_p(0) = 0 \\ s_p(t_e) &= \dot{s}_p(t_e) = v_p(t_e) = 0 \end{aligned}$$

Zirkularinterpolation Neben der Linearinterpolation für die Beschreibung von Geraden ist für eine numerische Steuerung die Zirkularinterpolation für die Beschreibung von Kreisbögen eine weitere Möglichkeit den Weg des Endeffektors zu beschreiben. Durch die Angabe von drei Punkten kann ein Kreisbogen festgelegt werden (siehe Abb.: 3.8). Der Startpunkt (Punkt 1) wird entweder gesetzt oder der Zielpunkt der vorherigen Bahn wird als Startpunkt übernommen. Darüber hinaus sind ein Zielpunkt (Punkt 3) und ein Hilfspunkt (Punkt 2) notwendig, um den Kreisbogen zu beschreiben. Um die gesamte Bewegung vollständig zu beschreiben, sind

[18] Weber 2017, S.86
[19] ebd., S.87

zusätzlich Angaben über Beschleunigung, Geschwindigkeit und Orientierungsänderung des TCP notwendig.

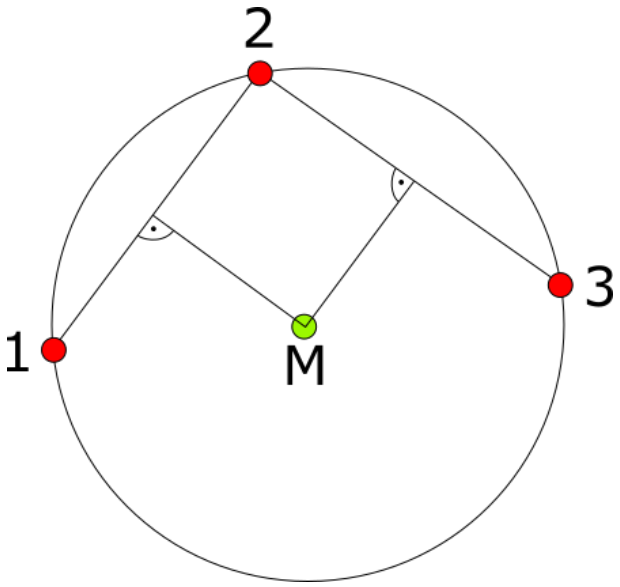

Abbildung 3.8: Kreis mittels 3 Punkten

Den Mittelpunkt (M) des Kreises und damit auch der Radius des Kreises lässt sich durch den Schnittpunkt der Normalen der Verbindungen zwischen den Punkten 1-3 ermitteln.

3.5.3 Überschleifen

Beim Anfahren der einzelnen Punkte würde der Roboter beim Erreichen eines Zwischenpunktes bis zum Stillstand abbremsen und für den nächsten Punkt erneut beschleunigen. Dies führt zu einem ruckartigen Ablauf und ist in den meisten Fällen nicht gewünscht. Um dem entgegen zu wirken, gibt es zwei Lösungen des Überschleifens. Beim **Positionsüberschleifen** wird das nächste Bahnsegment angefahren sobald ein Mindestabstand zwischen TCP und Zielpunkt unterschritten wurde. Mit der zweiten Methode, dem **Geschwindigkeitsüberschleifen**, wird nach dem Unterschreiten einer Mindestgeschwindigkeit das nächste Bahnsegmet abgefahren. Durch diese Methoden kommt der Effektor nie zum Stillstand und durchfährt die vorgegeben Strecke kontinuierlich. Diese Arten des Überschleifens werden für die PTP und CP-Steuerung gleichermaßen angewandt.[20]

[20] Weber 2017, S.96

3.6 Dynamisches Modell

Dieses Kapitel vermittelt die Herleitung des dynamischen Robotermodells und zeigt den Zusammenhang zwischen Position, Geschwindigkeit und Beschleunigung der einzelnen Glieder mit den erforderlichen Antriebskräften und Momenten.

3.6.1 Lagrange Verfahren

Die Bewegungsgleichungen des Roboters lassen sich mit dem Lagrange Verfahren durch n gekoppelte nichtlineare Differentialgleichungen zweiter Ordnung aufstellen. Dieses Verfahren leitet sich aus dem Hamiltonschen Prinzip ab. Durch ein Extremum des Wirkungsfunktionals wird der Zustand des konservativen physikalischen Systems festgelegt.[21]

$$T(x) = \int_{t_1}^{t_2} \mathcal{L}(x(t), \dot{x}(t), t)dt \tag{3.13}$$

In Gleichung 3.13 ist das Wirkungsfunktional zwischen den Zeitpunkten $t_1$und t_2 stationär, wobei $\mathcal{L}$ die Lagrange-Funktion (vgl. Formel 3.14) bei Starrkörpersystemen als Differenz von kinetischer (K) und potentieller (P) Energie bezeichnet wird.

$$\mathcal{L}(x, \dot{x}, t) = K(x, \dot{x}, t) - P(x, \dot{x}, t) \tag{3.14}$$

Daraus lassen sich die Lagrange Gleichungen (3.15) durch Differentiation von (3.14) nach der Zeit und der generalisierten Koordinate q_i aufstellen. Dabei bezeichnet n die Anzahl der Freiheitsgrade des Systems.

$$\tau_i = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \frac{d}{dt}\frac{\partial K}{\partial \dot{q}_i} - \frac{\partial K}{\partial q_i} + \frac{\partial P}{\partial q_i} \quad i = 1, .., n \tag{3.15}$$

Um die Bewegungsgleichungen aufstellen zu können, müssen die Massen m_i, deren Verteilung $I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz}$ und die Lage der Schwerpunkte $\vec{r}_i$ bekannt sein. Dafür wird die Trägheitsmatrix (3.16) mit den Trägheitsmomenten auf der Diagonalen aufgetragen.

$$\vec{\vec{\mathcal{I}}} = \begin{Bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yx} \\ -I_{xz} & -I_{yx} & I_{zz} \end{Bmatrix} \tag{3.16}$$

Diese beschreibt die Massenverteilung eines Gliedes im Referenzkoordinatensystems. Daraus wird die zeitinvariante Massenmatrix (3.17) für das i-te Glied erstellt.

[21] Arens u. a. 2015, S.1321

$$\vec{\vec{M}}_i = \begin{Bmatrix} m_i & 0 & 0 & 0 & 0 & 0 \\ 0 & m_i & 0 & 0 & 0 & 0 \\ 0 & 0 & m_i & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & -I_{xy} & -I_{xz} \\ 0 & 0 & 0 & -I_{xy} & I_{yy} & -I_{yx} \\ 0 & 0 & 0 & -I_{xz} & -I_{yx} & I_{zz} \end{Bmatrix} \tag{3.17}$$

Damit lässt sich die kinetische Energie eines Gliedes berechnen und durch Aufsummieren die gesamte kinetische Energie des Roboters berechnen. Diese setzt sich aus der Translationsenergie bezogen auf die Schwerpunktsgeschwindigkeit und der Rotationsenergie um den Schwerpunkt zusammen. Allgemein ist die kinetische Energie beschrieben als $K = \underbrace{\frac{1}{2} \cdot m \cdot v_s^2}_{Translation} + \underbrace{\frac{1}{2} \cdot I \cdot \omega^2}_{Rotation}$.[22] Übertragen auf das Robotersystem mit mehreren Freiheitsgraden ergibt sich nachstehende Formel (3.18) mit J_i als i-te partielle Jakobi-Marix $(6 \times n)$, ${}^i_0T$ als Transformationsmatrix (siehe 3.4), Q als Gelenkvariablen,$\dot{Q}$ als Gelenkgeschwindigkeit und $\vec{r}_i$ als Positionsvektor des Schwerpunktes des i-ten Gliedes im i-ten Koordinatensystem.

$$K = \frac{1}{2} \cdot \underbrace{\left\{ \sum_{i=1}^{n} \left(\vec{J}_i \left[{}^i_0\vec{T} \cdot \begin{Bmatrix} 1 & 0 & 0 & r_{x,i} \\ 0 & 1 & 0 & r_{y,i} \\ 0 & 0 & 1 & r_{z,i} \\ 0 & 0 & 0 & 1 \end{Bmatrix} \right] \right)^T \cdot \vec{\vec{M}}_i \cdot \left(\vec{J}_i \left[{}^i_0\vec{T} \cdot \begin{Bmatrix} 1 & 0 & 0 & r_{x,i} \\ 0 & 1 & 0 & r_{y,i} \\ 0 & 0 & 1 & r_{z,i} \\ 0 & 0 & 0 & 1 \end{Bmatrix} \right] \right) \right\}}_{\vec{\vec{M}}(Q)} \cdot \dot{Q}^2 \tag{3.18}$$

Die potentielle Energie ist durch die Lage einer Masse im konstanten Schwerefeld bestimmt und lautet allgemein $P = m \cdot g \cdot h$. Angewandt auf das System des Roboters ergibt sie sich aus

$$P = \sum_{i=1}^{n} -m_i \cdot (g_x, g_y, g_z, 0) \cdot {}^i_0\vec{\vec{T}} \cdot \begin{pmatrix} r_{x,i} & r_{y,i} & r_{z,i} & 1 \end{pmatrix}^T \tag{3.19}$$

die potentielle Energie des Gesamtsystems. Die Fallbeschleunigung in normalen Richtung beträgt auf Meereshöhe $\vec{g} = 9,80665 \frac{m}{s^2}$ und wirkt bei einem Roboter, der auf einer horizontalen Plattform montiert ist, nur in negative z-Richtung.

[22] Honerkamp und Römer 1993, S.84

Mit (3.18) und (3.19) ergibt sich für (3.15):

$$\tau_i = \vec{\vec{M}}(Q) \cdot \ddot{Q} + \underbrace{\left(\sum_{i=1}^{n} \frac{\partial \vec{\vec{M}}(Q)}{\partial q_i} \dot{q}_i \right) \cdot \dot{Q} - \frac{1}{2} \begin{pmatrix} \dot{Q}^T \frac{\vec{\vec{M}}(Q)}{\partial q_1} \cdot \dot{Q} \\ \vdots \\ \dot{Q}^T \frac{\vec{\vec{M}}(Q)}{\partial q_n} \cdot \dot{Q} \end{pmatrix}}_{\vec{\vec{C}}(Q,\dot{Q}) \cdot \dot{Q}} + \underbrace{\begin{pmatrix} \frac{P}{\partial q_1} \\ \vdots \\ \frac{P}{\partial q_n} \end{pmatrix}}_{\vec{G}(Q)} \tag{3.20}$$

Mit $M(Q)$ wird die Massenmatrix, mit $C(Q, \dot{Q})$ die Matrix der Zentrifugal- und Corioliskräfte und $G(Q)$ beschreibt den Gravitationsvektor.

3.6.2 Newton-Euler Verfahren

Auf Basis der drei Newton'schen Axiome[23] werden die Bewegungen von Körpern unter dem Einfluss von einwirkende Kräfte bestimmt.

3.6.2.1 Basis zum Endeffektor

Winkelgeschwindigkeit $\vec{\omega}_i$ und Beschleunigung $\vec{\dot{\omega}}_i$ werden iterativ ausgehend von der Basis (vgl.: Abb. 3.9) bis zum Effektor bestimmt.

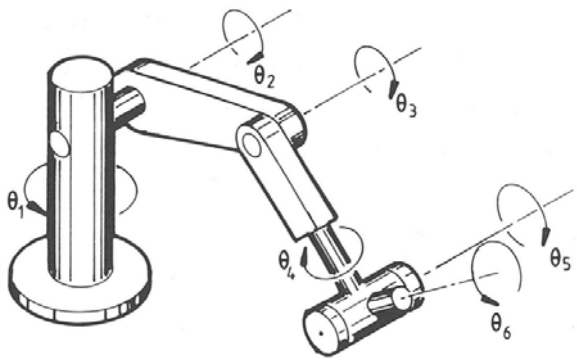

Abbildung 3.9: Sechs Drehgelenke eines vertikalen Knickarmroboters [24]

Da nur Drehgelenke betrachtet werden, setzt sich die Gelenkwinkelgeschwindigkeit ω_{i+1} des $i+1$ Arms zusammen aus:

$$\vec{\dot{\omega}}_{i+1} = \omega_i + z_i \cdot \dot{\Theta}_{i+1} \tag{3.21}$$

[23] Honerkamp und Römer 1993, S.7
[24] McCloy 1989

Dabei beschreibt z_i den Versatz der Achse $i+1$ zur Achse i in Richtung der $i+1$ Gelenkachse und $\dot{\Theta}_{i+1}$die Winkelgeschwindigkeit[25].
Die Winkelbeschleunigung $\dot{\omega}_{i+1}$entspringt der Differenziation 3.21 und lautet:

$$\vec{\dot{\omega}}_{i+1} = \dot{\omega}_i + z_i \cdot \ddot{\Theta}_{i+1} + \omega_i \times z_i \cdot \dot{\Theta}_{i+1} \tag{3.22}$$

Analog zu dem Zusammenhang zwischen Winkel, Winkelgeschwindigkeit und Winkelbeschleunigungen gilt es auch für die Lage, Geschwindigkeit und Beschleunigung des Schwerpunktes im globalen Koordinatensystems (siehe Abbildung 3.10).

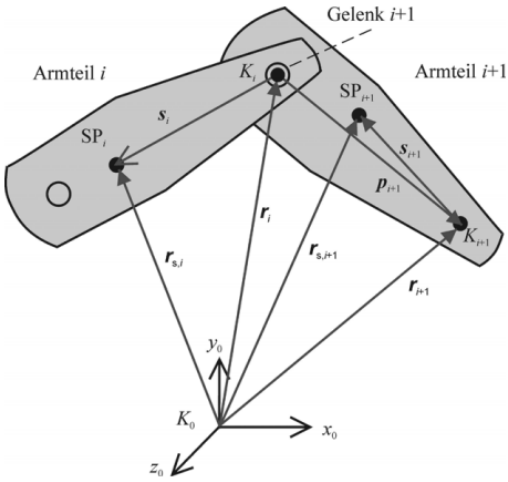

Abbildung 3.10: kinematische Beziehung zwischen zwei Armteilen [26]

Die Lage des Koordinatenursprungs i ist dabei mit $\vec{r}_i$ und die relative Lage des Ursprungs $i+1$ davon wird mit $\vec{p}_{i+1}$ bezeichnet. Mit

$$\vec{r}_{i+1} = \vec{r}_i + \vec{p}_{i+1} \tag{3.23}$$

ergibt sich die absolute Lage von K_{i+1}im Koordinatensystem K_0
Die Geschwindigkeit und Beschleunigung von K_{i+1} ergibt sich aus:

[25] Weber 2017, S.133
[26] ebd., S.133

$$\vec{v}_{i+1} = \frac{d}{dt}\vec{r}_{i+1} = \frac{d}{dt}\vec{r}_i + \frac{d^{(i+1)}}{dt}\vec{p}_{i+1} = \vec{v}_i + \frac{d^{(i+1)}}{dt}\vec{p}_{i+1} + \vec{\omega}_{i+1} \times \vec{p}_{i+1} \tag{3.24}$$

$$\begin{aligned}\dot{\vec{v}}_{i+1} = \frac{d}{dt}\vec{v}_{i+1} &= \frac{d}{dt}\vec{v}_i + \left(\frac{d^{2,(i+1)}}{dt^2}\vec{p}_{i+1} + \vec{\omega}_{i+1} \times \frac{d^{(i+1)}}{dt}\vec{p}_{i+1} + \vec{\omega}_{i+1} \times \frac{d^{(i+1)}}{dt}\vec{p}_{i+1}\right) \\ &+ \dot{\vec{\omega}}_{i+1} \times \vec{p}_{i+1} + \dot{\vec{\omega}}_{i+1} \times (\vec{\omega}_{i+1} \times \vec{p}_{i+1}) \\ &= \dot{\vec{v}}_i + \dot{\vec{\omega}}_{i+1} \times \vec{p}_{i+1} + (\vec{\omega}_{i+1} \times \vec{p}_{i+1}) \\ &+ \left(\frac{d^{2,(i+1)}}{dt^2}\vec{p}_{i+1} + 2 \cdot \vec{\omega}_{i+1} \times \frac{d^{(i+1)}}{dt}\vec{p}_{i+1}\right)\end{aligned} \tag{3.25}$$

Für die Berechnung der Lineargeschwindigkeit und Linearbeschleunigung des Schwerpunktes des $i+1$ ten Gliedes kann der Vektor $r_{s,i+1}$ (vgl. 3.10) genutzt werden.[27]

$$\vec{v}_{s,i+1} = \frac{d}{dt}\left(\vec{r}_{i+1} + \vec{s}_{i+1}\right) = \vec{v}_{i+1} + \frac{d^{(i+1)}}{dt}\vec{s}_{i+1} + \vec{\omega}_{i+1} \times \vec{s}_{i+1} \tag{3.26}$$

$$= \vec{v}_{i+1} + \vec{\omega}_{i+1} \times \vec{s}_{i+1} \tag{3.27}$$

$$\begin{aligned}\dot{\vec{v}}_{s,i+1} &= \dot{\vec{v}}_{i+1} + \frac{d}{dt}\left(\vec{\omega}_{i+1} \times \vec{s}_{i+1}\right) \\ &= \dot{\vec{v}}_{i+1} + \frac{d}{dt}\vec{\omega}_{i+1} \times \vec{s}_{i+1} + \vec{\omega}_{i+1} \times \frac{d}{dt}\vec{s}_{i+1} \\ &= \dot{\vec{v}}_{i+1} + \dot{\vec{\omega}}_{i+1} \times \vec{s}_{i+1} + \vec{\omega}_{i+1} \times \left(\frac{d^{(i+1)}}{dt}\vec{s}_{i+1} + \vec{\omega}_{i+1} \times \vec{s}_{i+1}\right) \\ &= \dot{\vec{v}}_{i+1} + \dot{\vec{\omega}}_{i+1} \times \vec{s}_{i+1} + \vec{\omega}_{i+1} \times (\vec{\omega}_{i+1} \times \vec{s}_{i+1})\end{aligned} \tag{3.28}$$

Mittels des Impulssatzes[28] lassen sich die Kräfte $\mathcal{F}_i$ bestimmen wenn die Massen m_i der Glieder bekannt sind .

$$\mathcal{F}_i = m_i \cdot \dot{\vec{v}}_i \tag{3.29}$$

Mit der Multiplikation der Trägheitsmatrix $\mathcal{I}_{s,i}$ bezüglich des Schwerpunktes von i (3.16) mit der Winkelrotation $\vec{\omega}_i$ lässt sich der Drehimpuls und damit nach Differentiation auch das Drehmoment $\mathcal{M}_i$, welches im Schwerpunkt s vom i-ten Glied wirkt, berechnen. Dieser Zusammenhang ergibt sich aus dem Drallsatz[29]. Dabei ist darauf zu achten das die Trägheitsmatrix des Gliedes im Basiskoordinatensystem ausgedrückt ist.

$$\mathcal{M}_i = \mathcal{I}_{s,i} \cdot \dot{\vec{\omega}}_i - \vec{\omega}_i \times (\mathcal{I}_{s,i} \cdot \vec{\omega}_i) \tag{3.30}$$

[27] Weber 2017, S.134
[28] Honerkamp und Römer 1993, S.27
[29] ebd., S.24

3.6.2.2 Endeffektor zur Basis

Um von dem i-ten Glied auf das vorherige zu schließen, wird bei der Rückwärtsrechnung beim Endeffektor gestartet und sich bis zur Basis vorgearbeitet. Dabei werden die Kräften und deren Angriffspunkt am Armgelenk bestimmt

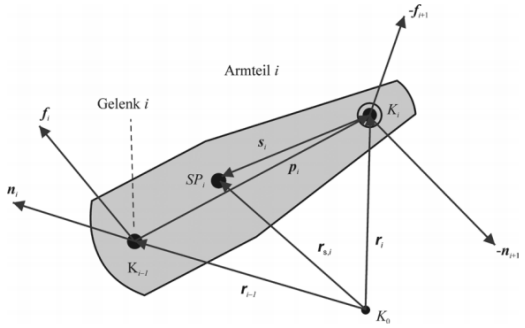

Abbildung 3.11: Kräfte und Moment an einem freigeschnittenen Armteil[30]

Um die Kraft $\vec{f_i}$ und das Moment $\vec{n}_i$ welches vom Gelenk i auf das Gelenk $i-1$ einwirkt zu berechnen. Die Lasten bei $i+1$werden Anfangs mit $\vec{0}$ angenommen. Damit ergibit sich:

$$\vec{f_i} = \vec{f}_{i+1} + \mathcal{F}_i \tag{3.31}$$

$$\vec{n_i} = \vec{n}_{i+1} + \mathcal{M}_i + \vec{p_i} \times \vec{f_i} + \vec{r_i} \times \mathcal{F}_i \tag{3.32}$$

Für den Vektor τ_i aus dem Newton-Euler Verfahren ergibt sich für Drehgelenke lediglich die Komponente von $\vec{n}_i$ in z_{i-1} Richtung:

$$\tau_i = \vec{n}_i^T \cdot z_{i-1} \tag{3.33}$$

[30] Weber 2017, S.136

4 Entwurf des Konzeptes

Dieses Kapitel soll nach einer Klärung der Anforderungen und dem Aufzeigen verschiedener Realisierungsmöglichkeiten zu einem umsetzbaren Konzept führen. Dabei ist das Ziel, Standardkomponenten einzusetzen und nur in Ausnahmefällen auf eine Eigenfertigung zu setzen. Durch die Vorgaben der Aufgabenstellung ist bereits die Verwendung des IFT-Wellgetriebes gesetzt.

4.1 Anforderungen

Für die Anwendung als kollaborativer Roboter steht die Sicherheit des Bedienenden an oberster Stelle. Durch das Wahrnehmen der Umgebung mittels Sensoren soll es dem Roboter möglich sein, Kollisionen und damit Verletzungen zu vermeiden. Darüber hinaus werden im Rahmen der Möglichkeiten die „Gestaltungsanforderungen und Schutzmaßnahmen" der ISO 10218-1 berücksichtigt. Diese schreibt zum Beispiel vor, dass die Gefährdung durch bewegliche Teile wie Motoren und Getrieben nach Möglichkeit durch Abdeckungen auszuschließen sind. Desweiteren darf ein Energieausfall oder eine Schwankung nicht zu Gefährdungen führen. Um im Notfall zu einem sichern Halt zu kommen, von dem keine weitere Gefährdung ausgeht, ist ein Not Halt vorzusehen, der zu jeder Zeit zur Verfügung steht. Durch eine Zustandsanzeige muss der aktuelle Zustand des Roboters ersichtlich sein.[1]
Um eine gewisse Mobilität des Roboters zu ermöglichen, soll eine einfache De-/Montage, sowie eine schnellen Inbetriebnahme möglich sein. Um bei der Steuerung nicht von Drittanbietern abhängig zu sein, ist eine eigene Ausführung vorzusehen. Diese soll damit die Möglichkeit einer einfachen Erweiterung bieten und flexibel an die Anforderungen angepasst werden können.

[1] ISO10218-1 2012, S.12 ff.

4.2 verwendete Komponenten

Sowohl für die Konstruktion des Roboters als auch für dessen Steuerung bieten sich eine Reihe von Möglichkeiten der Umsetzungen. In Kapitel 2.2 sind einige gängige Bauarten von kollaborativen Robotern anhand des aktuellen Angebotes am Markt vorgestellt worden. Dabei hat sich die Konfiguration mit sechs Achsen als am verbreitetsten herausgestellt, da sie alle Orientierungsmöglichkeiten im Raum abdeckt und keine redundanten Achsen besitzt. Da der Komplexitätsgrad der Kinematik mit jeder weiteren Achse zunimmt, wurde sich für eine sechsachsige Konfiguration entschieden.

4.2.1 Antrieb

Für einen sechsachsigen Roboter sind sechs nicht redundante Getriebefreiheitsgrade nötig.[2] Diese können sowohl rotatorisch als auch translatorisch ausgeführt sein. Dabei positionieren die ersten drei Achsen (beginnend im Fuß) den Endeffektor und bilden die Hauptachsen. Für die Orientierung des Endeffektors sind die Achsen vier bis sechs zuständig und werden als Handachsen bezeichnet. Diese tragen nur marginal zur Positionierung bei. Aus der Anordnung und Art der Hauptachsen ergibt sich die Bauart des Roboters. Charakteristisch ist dabei die Verkettung von linearen- und rotatorischen Achsen. Für einen Knickarmroboter ist die serielle Verknüpfung der drei Hauptachsen mit rotatorischen Gelenken vorgesehen (RRR-Bauform).[3] Damit ergibt sich ein kugelförmiger Arbeitsraum, der nur durch eventuelle Winkeleinschränkungen der Gelenke beschränkt ist.

Abbildung 4.1: Drehmomenten Kurve des QSH6018-45-28-110 bei 30V/3A[4]

[2] K.-H. Grote 2011, S.1601
[3] ebd., S.1603

Angetrieben werden die Achsen durch die Kombination eines Schrittmotors mit dem IFT-Wellgetriebe. Nachdem das IFT-Getriebe zunächst nur für die Verwendung mit einem kleinen Gleichstrommotor ausgelegt war musste dieses so angepasst werden um auch mit Schrittmotoren, die das nötige Drehmoment bieten, betrieben werden zu können. Durch die festgelegte Schrittweite ist eine exakte Positionierung beziehungsweise Drehung möglich. Für die Auswahl des passenden Motors wurden die Kriterien Drehzahl, Drehmomenten-Verlauf und Haltemoment betrachtet. Durch das IFT-Wellgetriebe ist zusätzlich auf die Anschlussmaße zu achten. Unter Berücksichtigung all dieser Faktoren wurde der QSH6018-45-28-110 Motor der Firma Trinamic gewählt, dessen Drehmomentverlauf der Abbildung 4.1 entnommen werden kann.

4.2.2 Elektronik

Für die Stromversorgung wurde ein 24 V/DC Netzteil mit 10 A der Firma PULS gewählt. Für einen sicheren Betrieb werden zwei Netzteile parallel geschaltet, so dass sich ein Gesamt-Strom von 20 A ergibt. Die Motoren sind für 2.8 A ausgelegt und können damit in dieser Konfiguration betrieben werden. Der Vorteil bei dieser Anordnung ist zudem, dass im Falle eines Ausfalls einer Stromquelle der Strom der zweiten ausreicht, um die Motoren mit reduzierter Geschwindigkeit zu einem sicheren Halt zu bringen.

Abbildung 4.2: PULS Dimension CS10.244[5]

Für die Steuerung der Motoren ist ein Mikrocontroller mit mindestens 6 Pulsweitenmodulations Ausgängen erforderlich. Die Frequenz und Periodendauer der Pulse werden durch einen Motortreiber in die Ansteuerung der vier Spulen des Schrittmotors umgesetzt. Als preiswerter Mikrocontroller wird ein Arduino Uno verwendet, welcher diese Anforderungen erfüllt.

[4]TRINAMIC Motion Control GmbH & Co. KG 2011, S.7
[5]PULS Österreich 2017

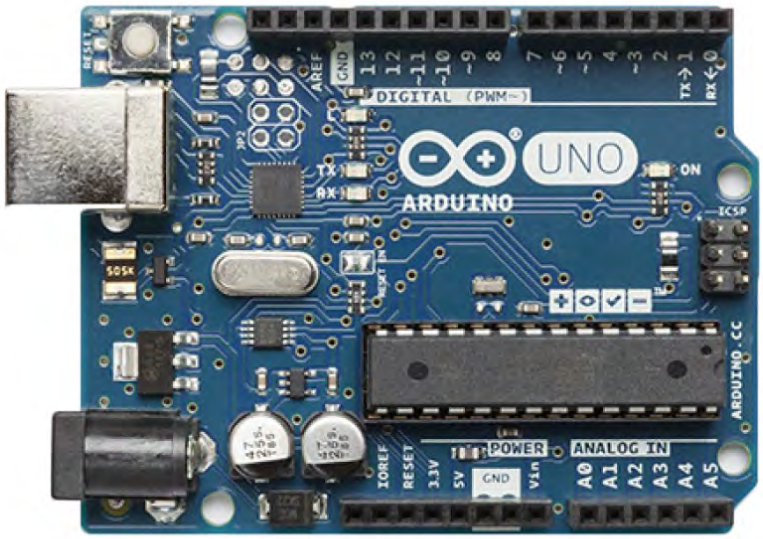

Abbildung 4.3: Arduino UNO[6]

Der Schrittmotortreiber hat die Aufgabe aus den Steuersignalen des Mikrocontrollers die Spannung aus der Spannungsquelle auf die Wicklungen zu schalten, um so den Motor in Rotation zu versetzen. Er fundiert damit als Verstärker, da die 5V des Mikrocontrollers nicht ausreichen, um den Motor mit ausreichender Spannung zu versorgen. Dabei wird die Spannungsquelle sowie der Motor und Mikrocontroller am Motortreiber angeschlossen (s. Abbildung 4.4).

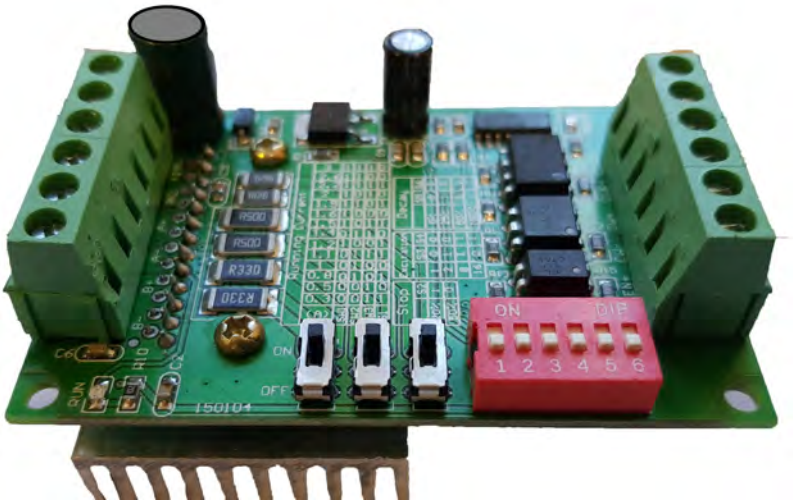

Abbildung 4.4: TB6560 Motor Treiber

[6] Arduino 2017

Um den Mikrocontroller vor den hohen Spannungen der Spannungsquelle (Abb. 4.5 Nr. 4) zu schützen sind die Steuereingänge (Abb. 4.5 Nr. 6 und 7) des Treibers durch Optokoppler (Abb. 4.5 Nr. 1) galvanisch getrennt. Gewählt wurde der TB6560 Treiber (Abb. 4.5 Nr. 3) da dieser Ströme bis zu 3 A ermöglicht und damit für die maximalen 2,8 A der Motoren geeignet ist.
Zusätzliche Einstellmöglichkeiten direkt auf der Treiberplatine (Abb. 4.5 Nr. 2) ermöglichen es, den Haltestrom, Mikroschritte und die Art der Abschaltung zu konfigurieren.

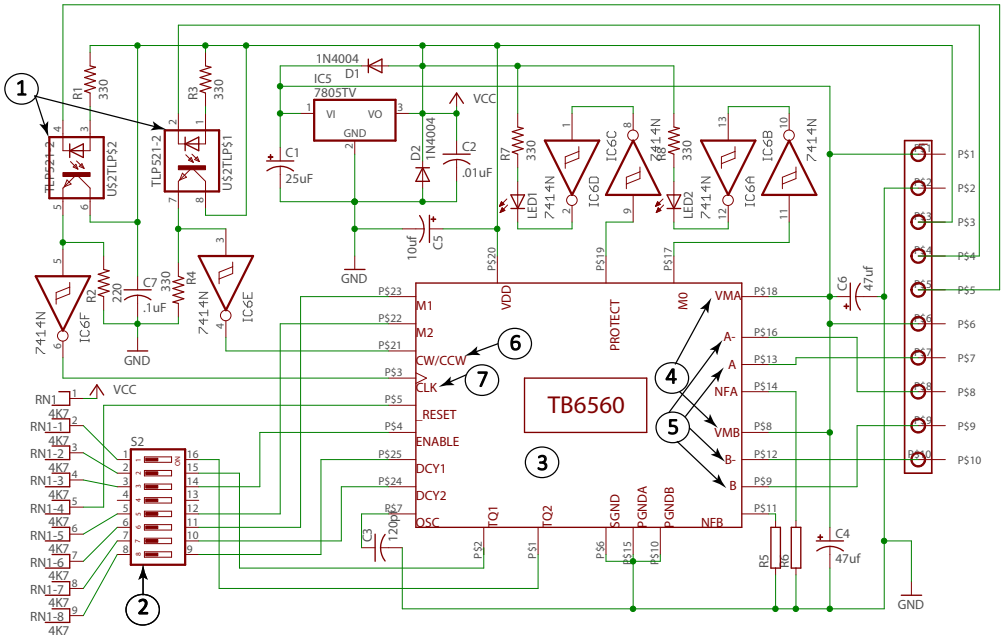

Abbildung 4.5: Schaltplan TB 6560

Der Haltestrom legt fest, wie viel Strom für das Halten der Position zur Verfügung steht. Als Auswahl bietet der Treiber ein Fünftel beziehungsweise die Hälfte des Nennstroms. Mikroschritte erlauben eine Unterteilung der einzelnen Schritte des Motors in kleinere Schritte. Eine Unterteilung in ganze, halbe, achtel und sechzentel Schritte ist möglich. Somit ist eine exaktere Positionierung des Motors möglich.
Aus diesen Komponenten lässt sich nun folgender Aufbau (siehe Abbildung 4.6) realisieren.

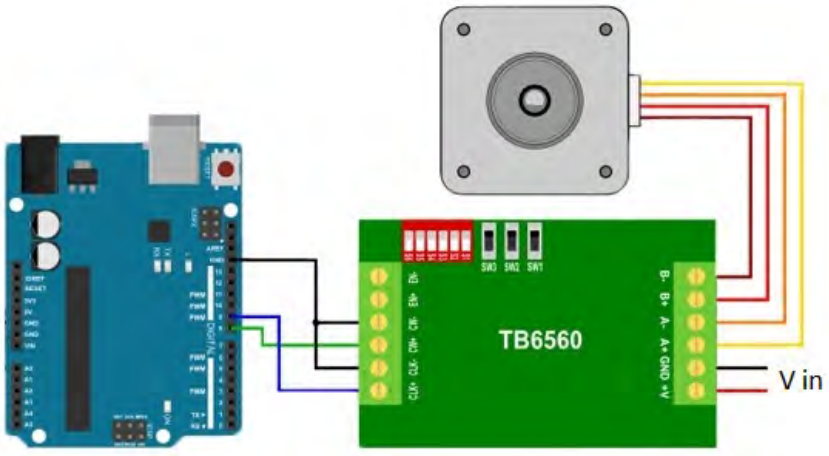

Abbildung 4.6: Konfiguration

4.2.3 Software

Für die Interaktion zwischen Bediener und Roboter wird die Arduino Schnittstelle von MATLAB genutzt. Diese bietet die Mögichkeit direkt aus dem MATLAB Workspace die Funktionen des Arduinos abzurufen. Auch existiert für MATLAB bereits die Robotics System Toolbox von Peter Corke.[7] Diese bietet Funktionen für die Simulation von Robotern und wurde für diese Arbeit verwendet. Mit diesen zur Verfügung stehenden Mitteln wurde eine Software erstellt, die mittels einer grafischen Benutzerschnittstelle die Möglichkeit bietet, den Roboter zu steuern und dabei auch die Sensordaten des Roboters zu verarbeiten. Eine Eingabe des Gelenkwinkels führt zu einer Vorwärtstransformation und gibt zum einen die Position des Endeffektors im Arbeitsraum aus und steuert die Motoren entweder hintereinander oder gleichzeitig an um diese Position einzunehmen. Im anderen Fall sind die Gelenkwinkel unbekannt und müssen nach Eingabe der gewünschten Position im Raum errechnet werden. Dies geschieht durch eine Rücktransformation der Transformationsmatrizen und führt zu einer Ausgabe der erforderlichen Gelenkwinkel und deren Umsetzung durch den Arduino. Als Kontrollinstanz wurde eine Visualisierung der Bewegung des Roboters in die Benutzerschnittstelle integriert. Diese ermöglicht es, mögliche Gefahren und Kollisionen im Vorfeld zu erkennen. Die Visualisierung kann je nach Einsatzort durch weitere Objekte im Arbeitsraum ergänzt werden.

[7] Corke 2017

4.3 Konstruktion des Roboterarms

Bei der Konstruktion des Roboters wurden alle Komponenten in die Arme integriert um ein geschlossenes Design zu erreichen und das Verletzungspotential so gering wie möglich zu halten. Der Roboter unterteilt sich in vier Elemente, die im Folgenden vorgestellt werden. Die Konstruktion ist dieser Arbeit in digitaler Form angehängt.

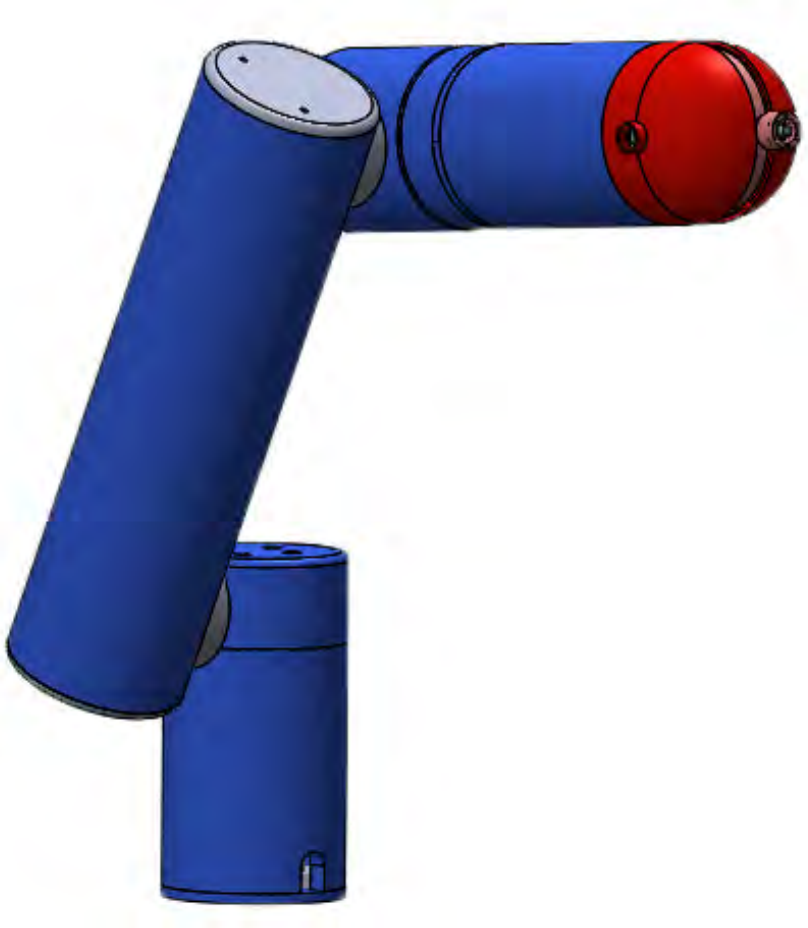

Abbildung 4.7: Roboter

4.3.1 Schulter

Die Schulter-Einheit (siehe Abbildung 4.8) beherbergt eine Drehachse um die vertikale Achse. Der Antrieb der Drehachse erfolgt über den Motor QSH6018-45-28-110 in Kombination mit dem IFT-Wellgetriebe. Der Abtrieb des Getriebes ist als Keilwelle ausgelegt und überträgt das Drehmoment formschlüssig auf die hohle Welle_1 (vgl. Abb. 4.8). Diese ist mit einem Radiallager und einem doppeltwirkenden Axiallager im Topf gelagert, welcher auf den Boden geschraubt ist. Der Lagerdeckel beherbergt das Radiallager, welches lediglich auf der Hohlwelle gesichert ist und spannt das Axiallager vor. Die Kraftübertragung von der Welle_1 auf den Deckel erfolgt über eine Schraubverbindung und bedingt dadurch die Drehung des Deckels. Die Verbindung mit der Oberarm-Einheit erfolgt über eine fest am Deckel montierten hohlen Welle_2. Durch die Nuten in der Welle_2 und Welle_1 werden die Kabel geführt. Damit die Kabel aus der Welle_2 in die Welle_1 geführt werden können, wird die Welle_1 durch Abstandshalter mit dem Deckel verbunden. Die Kabel werden über die Kabeldurchführung nach Außen geführt und sind mit der Steuereinheit verbunden.

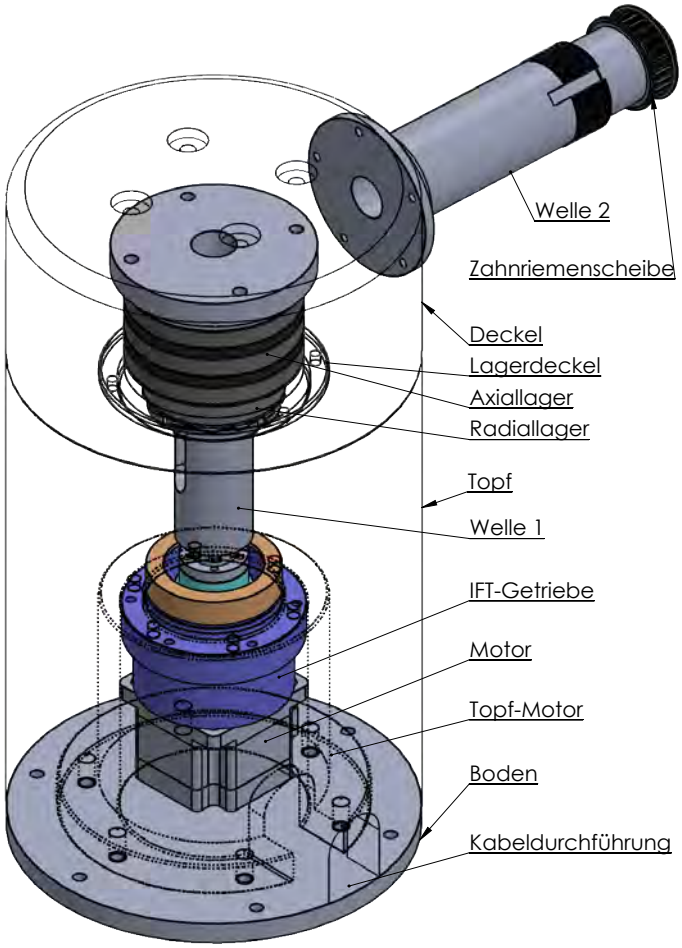

Abbildung 4.8: Schulter

4.3.2 Oberarm

Der Oberarm treibt die Drehachsen 2 und 3 an. Dafür sind zwei Motoren mit IFT-Getriebe auf der Verbindungsplatte montiert. Über zwei Zahnriemen sind die Abtriebe der Getriebe mit den Wellen von Schulter und Unterarm verbunden. Die Lagerung der Wellen 2 und 3 erfolgt über ein zweireihiges Schrägkugellager. Dieses ist mit je 12 Schrauben mit der Verbindungsplatte verbunden und über den Lagerdeckel vorgespannt. Durch die Kabeldurchführung werden die Zuleitungen zu den Motoren geführt. Zusammen mit den Zuleitungen zum Unterarm werden diese durch die Welle 2 zur Schulter geführt. Die Außenhülle schirmt die verbauten Komponenten ab.

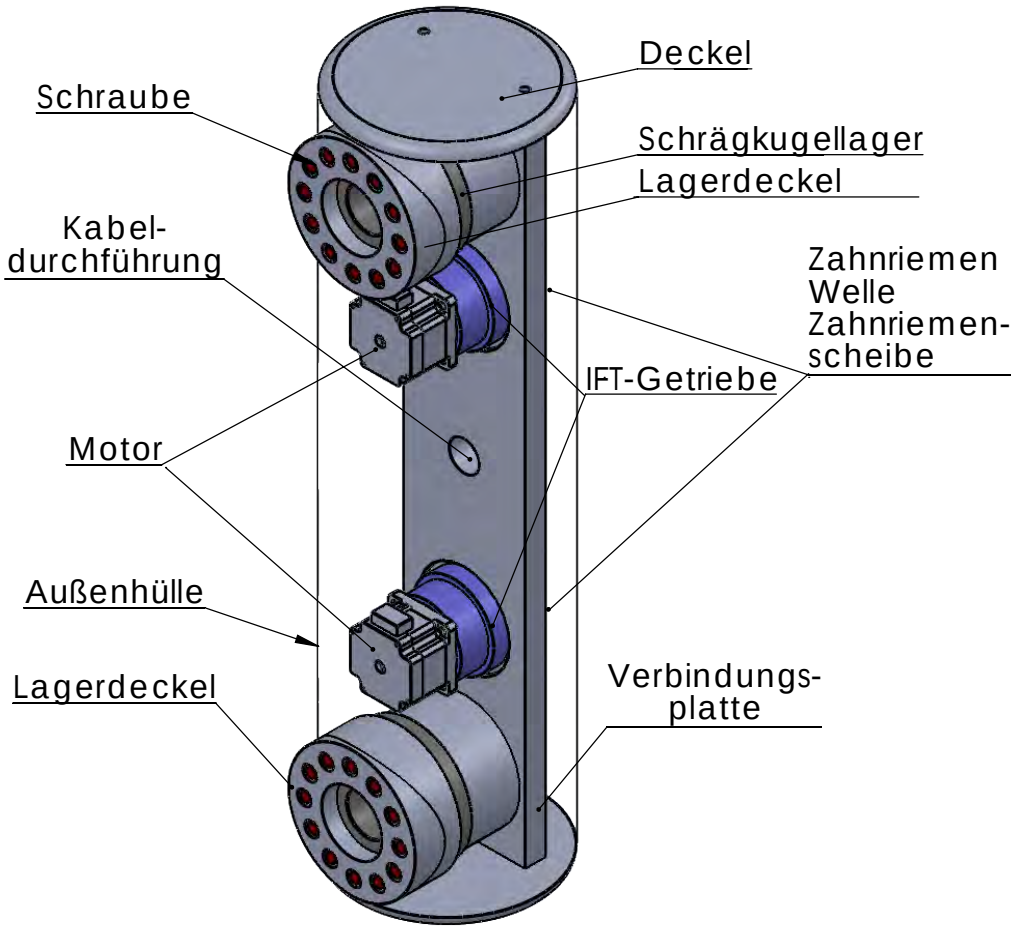

Abbildung 4.9: Oberarm

4.3.3 Unterarm

In diesem Teil werden die Drehachsen 4, 5 und 6 angetrieben. Die Welle 3 wird über den Zahnriemen aus dem Oberarm angetrieben. Die hohle Welle 4 wird durch eine am Motor montierte Keilwelle angetrieben und besitzt am anderen Ende eine Scheibe vom Durchmessers des Unterarms. Der Motor der Welle 4 ist im Motortopf montiert, welcher mit dem Untertopf verschraubt ist. Um die Kabel des Motors abführen zu können, sind im Motortopf Aussparungen vorgesehen worden. Die Welle 4 des oberen Teils des Unterarmes ist durch ein zweireihiges Schrägkugellager vom unteren Teil entkoppelt. Für die axiale Fixierung des oberen Teils wird der Innenring des zweireihigen Schrägkugellagers mit einer Nutmutter und Passfeder und der Außenring mit einem Deckel vorgespannt (s. Abb. 4.10) . Die damit erreichte O-Lagerung kann das Kippmoment durch die außerhalb des Lagers auftretenden Kräfte aufnehmen. Auf der Scheibe der Welle 4 ist ein S-förmiges Blech geschweißt, welches die zwei Motoren mit dem IFT-Getriebe aufnimmt. Die Zuleitungen dieser Motoren erfolgten durch die hohle Welle 4, welche am unteren Ende eine seitliche Bohrung für die Kabeldurchführung aufweist. Die Abtriebswellen der Getriebe leiten das Moment über Zahnriemen an das Differenzial im Handgelenk.

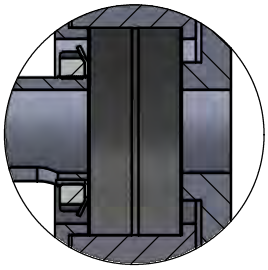

Abbildung 4.10: Lagerung

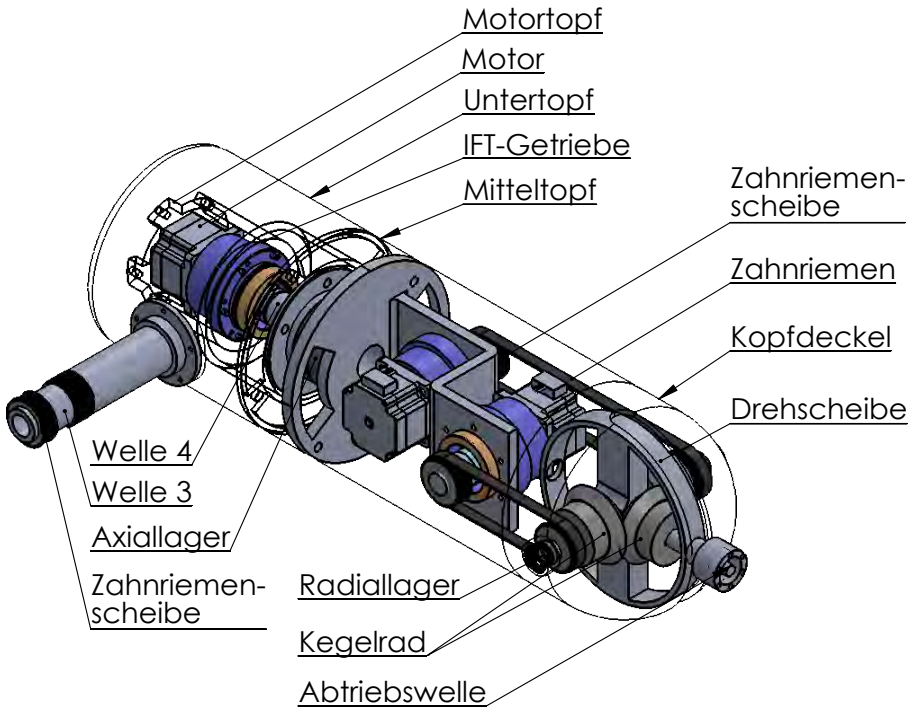

Abbildung 4.11: Unterarm

4.3.4 Handgelenk

Am Ende der kinematischen Kette befindet sich der Endeffektor. Dieser interagiert mit der Umwelt und befindet sich am Ende des Handgelenks.

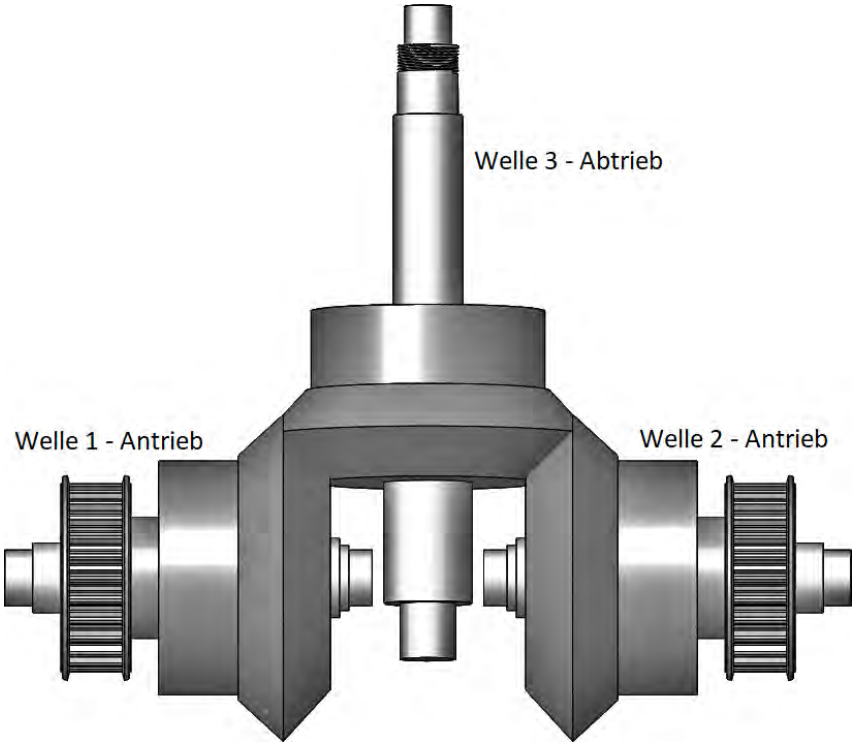

Abbildung 4.12: Differenzial

Der mechanische Aufbau des Handgelenks ist in Abbildung 4.12 dargestellt. Durch die Verwendung eines Differentials mit Riemenantrieb lassen sich zum einen die Motoren im Abstand zum Endeffektor montieren und zum anderen zusätzlich Bauraum einsparen. Ohne ein Differential müssten die Motoren gekreuzt eingebaut werden, um sowohl eine Drehung um die vertikal abgebildete Abtriebswelle als auch um die horizontale Welle zu ermöglichen.
Durch den Gleichlauf der beiden Antriebswellen wird eine Drehung um die horizontale Achse hervorgerufen. Bei gleicher gegenläufiger Momentenbeaufschlagung wird ausschließlich die Abtriebswelle angetrieben. Bei unterschiedlicher Beaufschlagung der Antriebswellen können auch Mischformen aus Drehung um die horizontale Achse und Drehung der Abtriebsachse entstehen.

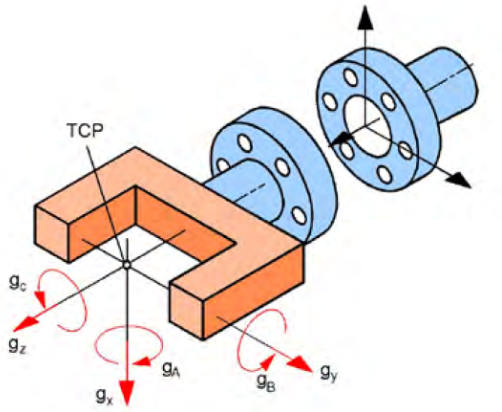

Abbildung 4.13: Endeffektor an Flansch [8]

An der Abtriebswelle können verschiedene Endeffektoren angeschlossen werden. Der TCP ist, wie in Abbildung 4.13 durch die Geometrie des angeschlossenen Endeffektors gegeben und ermöglicht mit der in Kapitel 4.4.1 beschriebenen Methode die Aufstellung der Transformationsmatrix.

[8] Hesse und Malisa 2016, S. 121

4.4 Kinematik des Roboters

Um den Roboter mathematisch darstellen zu können und Bewegungen zu berechnen, muss ein kinematisches Modell des Roboters erstellt werden. Das Denavit-Hartenberg Verfahren bietet sich bei sechsachsigen Gelenkarmrobotern mit ausschließlich rotierenden Achsen an. Damit lassen sich die benötigten Transformationsmatrizen aufstellen.

4.4.1 Denavit-Hartenberg Verfahren

Für die Beschreibung der kinematischen Kette des Entworfenen Roboters wird das Denavit-Hartenberg (Abk.: DH)Verfahren angewendet. Dafür wird jedes Glied der Kette mit einem körperfesten Koordinatensystem versehen und die Relationen der einzelnen Koordinatensysteme zueinander durch vier Parameter ausgedrückt. [9] Bei der Festlegung der n Koordinatensysteme sind vier Regeln (s. Tabelle4.1) zu befolgen.

z_n- Achse entspricht der Drehachse des Gelenks
x_n- Achse ist senkrecht zu z_n und z_{n-1}
y_n - Achse folgt der rechten Hand Regel
x_n- Achse muss sich mit z_{n-1}- Achse schneiden

Tabelle 4.1: Regeln für das Erstellen von Koordinatensysteme nach dem DH-Prinzip

Die vier DH-Parameter beschreiben die nachstehend aufgeführten Parameter [10].

θ_n	Rotation des n-Koordinatensystems um die z_{n-1}Achse
α_n	Winkel zwischen der z_{n-1}und z_n Achse um die x_n-Achse
d	Versatz des n zum $n-1$ Koordinatensystems entlang der z_{n-1}-Achse
a	Versatz des n zum $n-1$ Koordinatensystems entlang der x_n-Achse

Tabelle 4.2: Definition der DH-Parameter

Das Koordinatensystem der Basis kann unter Berücksichtigung der ersten Regel beliebig gewählt werden. Die gewählte Konfiguration ist in Abbildung 4.14 links ersichtlich.

[9] Gerke 2014, S. 180
[10] Hesse und Malisa 2016, S.203

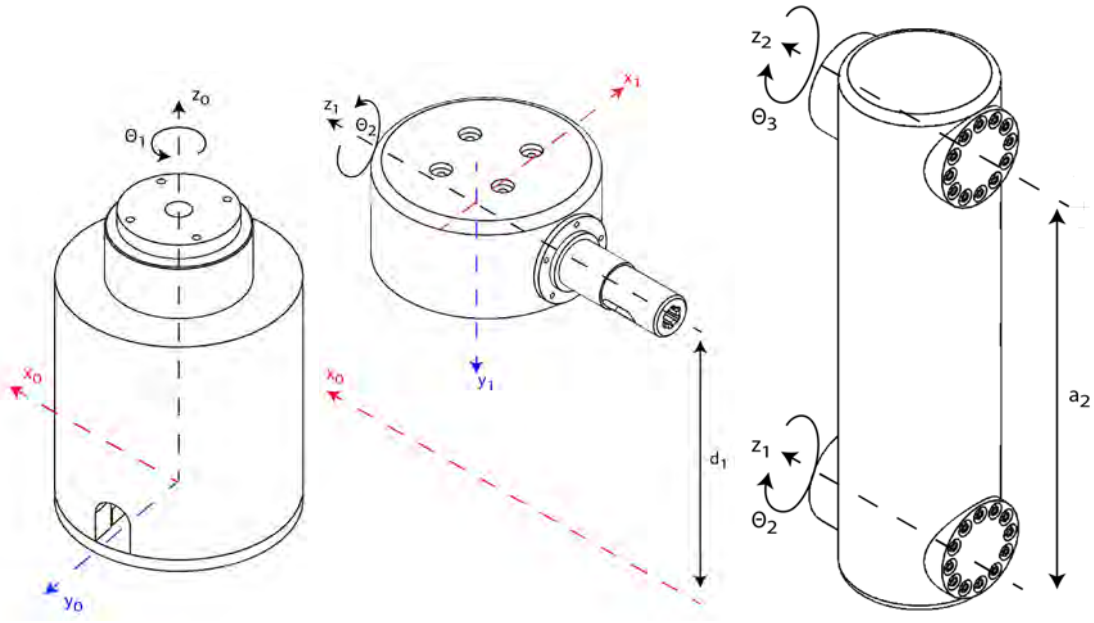

Abbildung 4.14: Koordinatensysteme: links: Basis | mitte: Schulter | rechts: Oberarm

Für das Schultergelenk (siehe Abbildung 4.14 Mitte) ergibt sich die z_1 Achse aus der Drehachse des Schultergelenks. Die x_1 Achse liegt senkrecht dazu und schneidet die Achse z_0. Damit ergibt sich aus der rechten Hand Regel die y_1 Achse. Die DH-Parameter ergeben sich zum einen aus dem Versatz entlang der z_0- Achse mit $d_1 = 240\,\text{mm}$ und der Rotation um die z_0- Achse um $-90°$. Da der Ursprung des Schulter-Koordinatensystems nur in der z_0- Richtung verschoben ist, ist der Versatz in x_1-Richtung gleich null. Für die DH-Parameter folgen daraus folgende Werte:

θ	d	a	α
θ_1	240 mm	0 mm	$-\frac{\pi}{2}$

Im Weiteren Verlauf folgt der Oberarm des Roboters dessen Drehachse mit z_2 abgebildet wird. Die x_2- Achse wird so gewählt, dass sie sich mit der z_1- Achse im 90° Winkel schneidet und damit senkrecht auf der z_2 und z_1 Achse steht. Die y_2- Achse ergibt sich aus der rechten Hand Regel. Um die Drehung des Koordinatensystems um die z_2 Achse gegenüber der z_1 Achse abzubilden wird θ_2 mit dem Verdrehwinkel 90°addiert. Der Versatz in x_2 Richtung beträgt $a_2 = 530\text{mm}$. Darüber hinaus existiert keine Rotation beziehungsweise kein Versatz gegenüber dem vorangegangenen Koordinatensystems. Damit lauten die DH-Parameter des Oberarms:

θ	d	a	α
$\theta_2 + 90°$	0 mm	530 mm	0

Der Ursprung des Koordinatensystems $n = 3$ fällt mit dem des Oberarms zusammen und ist lediglich um die x_3- Achse mit 90° verdreht. Damit ergeben sich die DH-Parameter:

θ	d	a	α
θ_3	0 mm	0 mm	$\frac{\pi}{2}$

Für das Handgelenk (siehe Abbildung 4.16) befindet sich die z_4- Achse im Abstand von $d_4 = 440$mm entlang der z_3-Achse zusätzlich ist das Koordinatensystem um $-90°$ um die x_4- Achse gedreht um die Bedingung nach der senkrechten Richtung der x_4 - Achse in Bezug auf z_3- und z_4-Achse zu erfüllen. Die zugehörigen DH-Parameter lauten daher:

θ	d	a	α
θ_4	440 mm	0 mm	$-\frac{\pi}{2}$

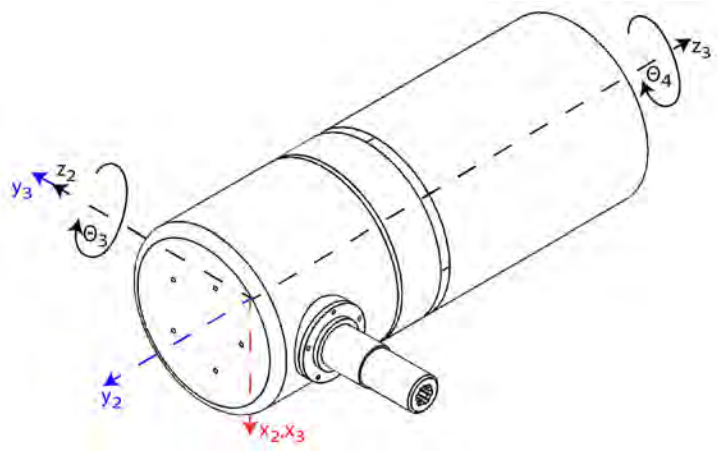

Abbildung 4.15: Koordinatensystem Unterarm

Für die Rotation der Abtriebswelle bildet die z_5- Achse die Drehachse. Da der Ursprung mit dem vorangegangenen System zusammenfällt, bedarf es lediglich einer Drehung von 90° , um die x_5- Achsem, damit die Regeln aus Tabelle 4.1 erfüllt sind. Folglich lauten die DH-Parameter:

θ	d	a	α
θ_5	0 mm	0 mm	$\frac{\pi}{2}$

Die Orientierung des Koordinatensystems am Endeffektor kann frei gewählt werden und wird daher wie im vorherigen gewählt. Lediglich der Versatz $d_6 = 123$mm entlang der z_5- Achse wird mit den DH-Parametern abgebildet:

θ	d	a	α
θ_6	123 mm	0 mm	0

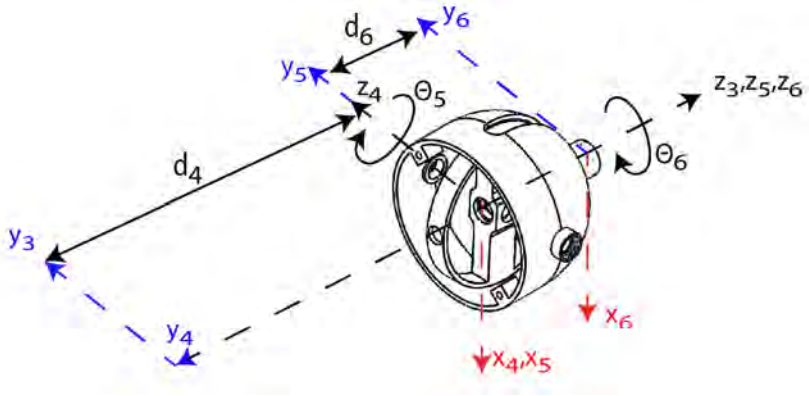

Abbildung 4.16: Koordinatensystem Handgelenk

Zusammengefasst ergibt sich Tabelle 4.3:

Gelenk i	Θ_i	d_i	a_i	α_i
1	Θ_1	$240mm$	0	$-\frac{\pi}{2}$
2	$\Theta_2 + \frac{\pi}{2}$	0	$530mm$	0
3	Θ_3	0	0	$\frac{\pi}{2}$
4	Θ_4	$440mm$	0	$-\frac{\pi}{2}$
5	Θ_5	0	0	$\frac{\pi}{2}$
6	Θ_6	$123mm$	0	0

Tabelle 4.3: Denavit-Hartenberg Parameter

Mit MATLAB lässt sich folgendes Modell erstellen:

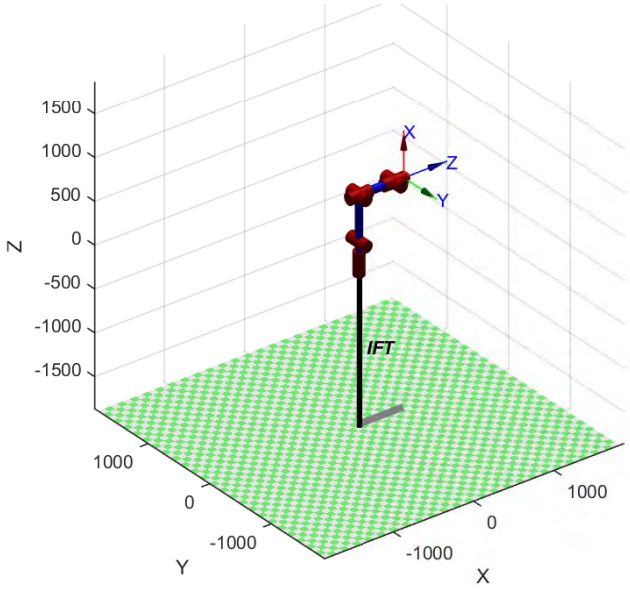

Abbildung 4.17: Roboter mittels DH-Parameter simuliert

Die Transformationsmatrix lässt sich damit durch folgende Matrix beschreiben.

$$\overset{\mathrm{i-1}}{\mathrm{i}}\mathrm{T} = \mathrm{Rot(ez,}\theta_i) \cdot Trans(ez, d_i) \cdot Trans(ex, a_i) \cdot Rot(ex, \alpha_i) \tag{4.1}$$

Für den allgemeinen Fall ergiebt sich aus 4.1

$${}^{i-1}_{i}T = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) \cdot cos(\alpha_i) & sin(\theta_i) \cdot sin(\alpha_i) & a_i \cdot cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i) \cdot cos(\alpha_i) & -cos(\theta_i) \cdot sin(\alpha_i) & a_i \cdot sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Für die einzelnen Gelenke ergeben sich damit nachstehende Matrizen.

$${}^{0}_{1}T = \begin{bmatrix} cos(\theta_1) & 0 & -sin(\theta_1) & a_1 \cdot cos(\theta_1) \\ sin(\theta_1) & 0 & cos(\theta_1) & a_1 \cdot sin(\theta_1) \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^{1}_{2}T = \begin{bmatrix} cos(\Theta_2 + \frac{\pi}{2}) & -sin(\Theta_2 + \frac{\pi}{2}) & 0 & a_2 \cdot cos(\Theta_2 + \frac{\pi}{2}) \\ sin(\Theta_2 + \frac{\pi}{2}) & cos(\Theta_2 + \frac{\pi}{2}) & 0 & a_2 \cdot sin(\Theta_2 + \frac{\pi}{2}) \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -sin(\theta_2) & -cos(\Theta_2) & 0 & -a_2 \cdot sin(\Theta_2) \\ cos(\theta_2) & -sin(\Theta_2) & 0 & a_2 \cdot cos(\Theta_2) \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^{2}_{3}T = \begin{bmatrix} cos(\theta_3) & 0 & sin(\theta_3) & a_3 \cdot cos(\theta_3) \\ sin(\theta_3) & 0 & -cos(\theta_3) & a_3 \cdot sin(\theta_3) \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^{3}_{4}T = \begin{bmatrix} cos(\theta_4) & 0 & -sin(\theta_4) & a_4 \cdot cos(\theta_4) \\ sin(\theta_4) & 0 & cos(\theta_4) & a_4 \cdot sin(\theta_4) \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$
{}^{4}_{5}T = \begin{bmatrix} cos(\theta_5) & 0 & sin(\theta_5) & a_5 \cdot cos(\theta_5) \\ sin(\theta_5) & 0 & -cos(\theta_5) & a_5 \cdot cos(\theta_5) \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{5}_{6}T = \begin{bmatrix} cos(\theta_6) & -sin(\theta_6) & 0 & a_6 \cdot cos(\theta_6) \\ sin(\theta_6) & cos(\theta_6) & 0 & a_6 \cdot cos(\theta_6) \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Um die Orientierung des Endeffektors gegenüber der Basis abbilden zu können wird durch Multiplikation der einzelnen Transformationsmatrizen die Gesamttransformationsmatrix berechnet.

$$
\begin{aligned}
{}^{0}_{6}T &= \prod_{i=1}^{6} {}^{i-1}_{i}T \\
&= \underbrace{{}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T}_{\text{Position}} \cdot \underbrace{{}^{3}_{4}T \cdot {}^{4}_{5}T \cdot {}^{5}_{6}T}_{\text{Orientierung}}
\end{aligned} \tag{4.2}
$$

Wobei die Matrizen ${}^{0}_{1}T$, ${}^{1}_{2}T$ und ${}^{2}_{3}T$ für die Positionierung des Armes zuständig sind und ${}^{3}_{4}T$, ${}^{4}_{5}T$ und ${}^{5}_{6}T$ für die Orientierung des Endeffektors verantwortlich sind.

4.4.2 Vorwärts Kinematik

Mit den ermittelten DH-Parametern ergibt sich für die Position des Armes im Raum folgende Matrix (vgl. 4.2)

$$
\begin{aligned}
{}^{0}_{3}T &= {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \\
&= \begin{bmatrix} -cos(\theta_1) \cdot sin(\theta_2) \cdot cos(\theta_3) - cos(\theta_1) \cdot cos(\theta_2) \cdot sin(\theta_3) & -sin(\theta_1) \\ -sin(\theta_1) \cdot sin(\theta_2) \cdot cos(\theta_3) - sin(\theta_1) \cdot cos(\theta_2) \cdot sin(\theta_3) & cos(\theta_1) \\ sin(\theta_3) \cdot sin(\theta_2) - cos(\theta_3) \cdot cos\theta_2) & 0 \\ 0 & 0 \end{bmatrix} \\
&\quad \begin{matrix} -cos(\theta_1) \cdot sin(\theta_2) \cdot sin(\theta_3) + cos(\theta_1) \cdot cos(\theta_2) \cdot cos(\theta_3) \\ -sin(\theta_1) \cdot sin(\theta_2) \cdot sin(\theta_3) + sin(\theta_1) \cdot cos(\theta_2) \cdot cos(\theta_3) \\ -cos(\theta_2) \cdot sin(\theta_3) - sin(\theta_2) \cdot cos(\theta_3) \\ 0 \end{matrix} \\
&\quad \begin{bmatrix} -cos(\theta_1) \cdot a_2 \cdot sin(\theta_2) \\ -sin(\theta_1) \cdot a_2 \cdot sin(\theta_2) \\ d_1 - a_2 \cdot cos(\theta_2) \\ 1 \end{bmatrix}
\end{aligned} \tag{4.3}
$$

Für die Orientierung des Endeffektors sind ${}^3_4T$, ${}^4_5T$ und ${}^5_6T$ verantwortlich und bilden gemeinsam ${}^3_6T$

$$
{}^3_6T = {}^3_4T \cdot {}^4_5T \cdot {}^5_6T = \begin{bmatrix}
cos(\theta_4) \cdot cos(\theta_5) \cdot cos(\theta_6) - sin(\theta_4) \cdot sin(\theta_6) & -cos(\theta_4) \cdot cos(\theta_5) \cdot sin(\theta_6) - sin(\theta_4) \cdot cos(\theta_6) & cos(\theta_4) \cdot sin(\theta_5) & cos(\theta_4) \cdot sin(\theta_5) \cdot d_6 \\
sin(\theta_4) \cdot cos(\theta_5) \cdot cos(\theta_6) + cos(\theta_4) \cdot sin(\theta_6) & -sin(\theta_4) \cdot cos(\theta_5) \cdot sin(\theta_6) + cos(\theta_4) \cdot cos(\theta_6) & sin(\theta_4) \cdot sin(\theta_5) & sin(\theta_4) \cdot sin(\theta_5) \cdot d_6 \\
-sin(\theta_5) \cdot cos(\theta_6) & sin(\theta_5) \cdot sin(\theta_6) & cos(\theta_5) & cos(\theta_5) \cdot d_6 + d_4 \\
0 & 0 & 0 & 1
\end{bmatrix} \tag{4.4}
$$

Durch Einsetzen der Winkel $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6$ ergibt sich die Position $[X \mid Y \mid Z]$ des Endeffektors im Raum aus der Gesamtmatrix

$$
{}^0_6T = {}^0_3T \cdot {}^3_6T = \begin{bmatrix}
u_x & v_x & w_x & p_x \\
u_y & v_y & w_y & p_y \\
u_z & v_z & w_z & p_z \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

4.4.3 Inverse Kinematik

Durch die gewählte Roboter Konfiguration (s. Abb.: 4.18) entstehen zusätzliche Randbedingungen, die eine analytische Lösung der inversen Kinematik ermöglichen. Durch das Schneiden der letzten 3 Achsen des Handgelenks in einem Punkt. Dies bietet uns die Möglichkeit, die Inverse der Kinematik analytisch zu lösen.

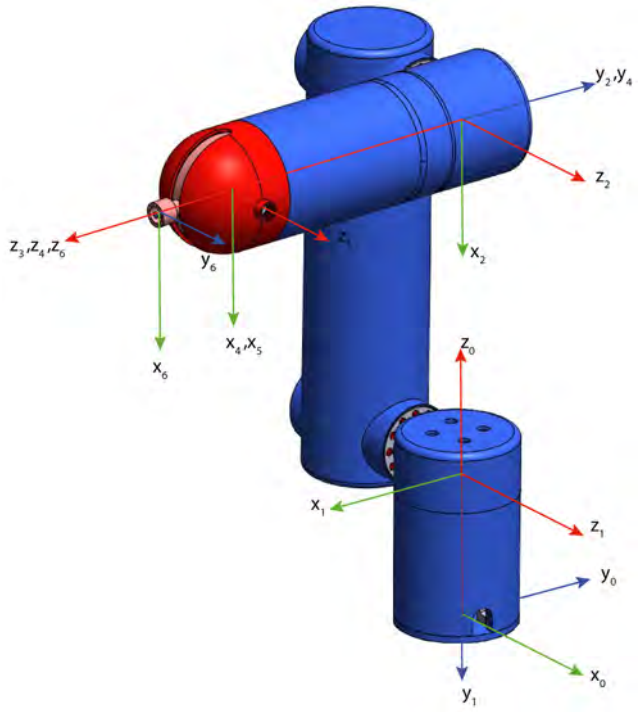

Abbildung 4.18: Koordinatensysteme der Gelenke

Dabei wird der Schnittpunkt der drei Handgelenkachsen als „Wrist Center Point" (Abk.:) bezeichnet. Die Position dieses WCP gilt es im ersten Schritt zu berechnen. Sie ist durch den Vektor $q_0 = [q_x, q_y, q_z, 1]^T$ beschrieben. In Bezug auf den TCP gilt $q_6 = [0, 0, -d_6, 1]^T$. Die Transformation von q_6 zu q_0 lautet damit:

$$q_0 = {}^0_6T \cdot q_6 = \begin{bmatrix} u_x & v_x & w_x & p_x \\ u_y & v_y & w_y & p_y \\ u_z & v_z & w_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix} \tag{4.5}$$

$$q_0 = \begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x - d_6 \cdot w_x \\ p_y - d_6 \cdot w_y \\ p_z - d_6 \cdot w_z \\ 1 \end{bmatrix} \tag{4.6}$$

Dies stellt die WCP-Position in Bezug auf das Aktuator-Koordinatensystem für eine

TCP-Position dar. Für die Position des WCP gilt in Bezug auf das dritte Gelenk $q_3 = [0,\ 0,\ d_4,\ 1]^T$. Die Transformation von q_3 zu q_0 lautet mit den Vereinfachungen

$$\begin{aligned} C_i &= cos(\theta_i) \\ S_i &= sin(\theta_i) \\ C_{ij} &= cos(\theta_i) \cdot cos(\theta_j) - sin(\theta_i) \cdot sin(\theta_j) = cos(\theta_i + \theta_j) \\ S_{ij} &= sin(\theta_i) \cdot cos(\theta_j) + sin(\theta_i) \cdot cos(\theta_j) = sin(\theta_i + \theta_j) \end{aligned}$$

damit:

$$q_0 = {}^0_3T \cdot q_3 = \begin{bmatrix} p_x - d_6 \cdot w_x \\ p_y - d_6 \cdot w_y \\ p_z - d_6 \cdot w_z \\ 1 \end{bmatrix} = \begin{bmatrix} -C_1 \cdot S_{23} & -S_1 & C_1 \cdot C_{23} & -C_1 \cdot S_2 \cdot a_2 \\ -S_1 \cdot S_{23} & C_1 & S_1 \cdot C_{23} & -S_2 \cdot S_1 \cdot a_2 \\ -C_{23} & 0 & -S_{23} & d_1 - C_2 \cdot a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ d_4 \\ 1 \end{bmatrix} \tag{4.7}$$

Nun kann durch die Invertierung von ${}^1_0T$:

$$\begin{aligned} ({}^0_1T)^{-1} \cdot q_0 &= ({}^1_0T)^{-1} \cdot {}^3_0T \cdot q_3 \\ &= ({}^1_0T)^{-1} \cdot {}^1_0T \cdot {}^2_1T \cdot {}^3_2T \cdot q_3 \\ &= {}^2_1T \cdot {}^3_2T \cdot q_3 \end{aligned} \tag{4.8}$$

$$\begin{bmatrix} C_1 & S_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix} = \begin{bmatrix} -S_{23} & 0 & C_{23} & -a_2 \cdot S_2 \\ C_{23} & 0 & S_{23} & a_2 \cdot C_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ d_4 \\ 1 \end{bmatrix} \tag{4.9}$$

ein Gleichungssystem mit drei unabhängigen Gleichungen aufstellen, um die ersten drei Variablen zu lösen.

$$q_x \cdot C_1 + S_1 \cdot q_y = C_{23} \cdot d_4 - a_2 \cdot S_2 \tag{4.10}$$

$$q_z + d_1 = S_{23} \cdot d_4 + a_2 \cdot C_2 \tag{4.11}$$

$$-q_x \cdot S_1 + q_y \cdot C_1 = 0 \tag{4.12}$$

mit (4.12) lässt sich der Winkel von θ_1 berechnen:

$$\theta_1 = arctan2(q_y, q_x) \tag{4.13}$$

Wobei sich die Variablen q_x und q_y aus (4.6) ergeben.

Durch die Quadratur von (4.10)-(4.12) ergibt sich durch die Addition aller drei Gleichungen:

$$q_x^2 + q_y^2 + q_z^2 + 2 \cdot q_z \cdot d_1 + d_1^2 = d_4^2 + a_2^2 + 2 \cdot a_2 \cdot d_4 \cdot (S_{23} \cdot C_2 - C_{23} \cdot S_2) \quad (4.14)$$

Mit dem Additionstheorem

$$S_{23} \cdot C_2 - C_{23} \cdot S_2 = sin(\theta_2 + \theta_3 - \theta_2) = sin(\theta_3) = S_3$$

folgt aus (4.14) :

$$q_x^2 + q_y^2 + q_z^2 + 2 \cdot q_z \cdot d_1 + d_1^2 = d_4^2 + a_2^2 + 2 \cdot a_2 \cdot d_4 \cdot S_3 \quad (4.15)$$

Für die weitere Rechnung wird (4.15) vereinfacht:

$$\kappa_1 = 2 \cdot a_2 \cdot d_4$$
$$\kappa_2 = q_x^2 + q_y^2 + q_z^2 + 2 \cdot q_z \cdot d_1 + d_1^2 - (d_4^2 + a_2^2)$$
$$\kappa_2 = \kappa_1 \cdot S_3$$

Damit lässt sich mit der trigonometrischen Beziehung:

$$sin(x) = \frac{2 \cdot tan\frac{X}{2}}{1 + tan^2\frac{X}{2}}$$

S_3 ersetzen durch $S_3 = \frac{2t_3}{1+t_3^2}$ wobei $t_3 = tan\frac{\theta_3}{2}$ist. Eingesetzt in (4.15) folgt die quadratische Gleichung:

$$\kappa_2 \cdot t_3^2 - 2 \cdot \kappa_1 \cdot t_3 + \kappa_2 = 0 \quad (4.16)$$

Die durch die pq-Formel gelöst werden kann:

$$t_{3;1,2} = \frac{\kappa_1 \pm \sqrt{\kappa_1^2 - \kappa_2^2}}{\kappa_2} \quad (4.17)$$

Und damit lässt sich nun θ_3 berechnen:

$$\theta_{3;1,2} = 2arctan(t_{3;1,2}) \quad (4.18)$$

Die zwei unterschiedlichen Lösungen sind je nach Konfiguration des Roboters zu wählen. Entweder ist der Ellbogen unten oder oben. Man bezeichnet diese Lösung auch als rechts- und linksarmige Lösung des inversen Problems.[11]
Der letzte Winkel des Armes θ_2 lässt sich aus (4.10) und (4.11) bestimmen.

[11] Jörg Wollnack 2007, S.3-19

Mit den folgenden Zusammenfassungen

$$\mu_1 = a_2 + d_4 \cdot S_3$$
$$\mu_2 = -d_4 \cdot C_3$$
$$\nu_1 = d_4 \cdot C_3$$
$$\nu_2 = a_2 + d_4 \cdot S_3$$

lässt sich ein lineares Gleichungssystem bilden:

$$\mu_1 \cdot C_2 + \nu_1 \cdot S_2 = \gamma_1$$
$$\mu_1 \cdot C_2 + \nu_2 \cdot S_2 = \gamma_2$$

Mit der Cramerschen Regel [12] erhält man die Lösungen für

$$S_2 = \frac{\gamma_1 \cdot \mu_2 - \gamma_2 \cdot \mu_1}{\nu_1 \cdot \gamma_2 - \nu_2 \cdot \gamma_1}$$
$$C_2 = \frac{\gamma_1 \cdot \nu_2 - \gamma_2 \cdot \nu_1}{\nu_2 \cdot \mu_1 - \nu_1 \cdot \mu_2}$$

Die Werte für γ_1 und γ_2 lassen sich dabei über den bereits berechneten Winkel $\theta_1$ermitteln.

$$\begin{aligned}\gamma_1 &= \big(a_2 + d_4 \cdot sin(\theta_3)\big) \cdot cos(\theta_2) + d_4 \cdot cos(\theta_3) \cdot sin(\theta_2) \\ &= a_2 \cdot cos(\theta_2) + d_4 \cdot sin(\theta_3) \cdot cos(\theta_2) + d_4 \cdot sin(\theta_2) \cdot cos(\theta_3) \\ &= a_2 \cdot cos(\theta_2) + d_4 \cdot \underbrace{\Big(sin(\theta_3) \cdot cos(\theta_2) + sin(\theta_2) \cdot cos(\theta_3)\Big)}_{sin(\theta_2+\theta_3)}\end{aligned}$$

Verglichen mit Formel 4.10 folgt:

$$\gamma_1 = q_x \cdot cos(\theta_1) - sin(\theta_1) \cdot q_y$$

$$\begin{aligned}\gamma_2 &= -d_4 \cdot cos(\theta_3) \cdot cos(\theta_2) + \Big(a_2 + d_4 \cdot sin(\theta_3)\Big) \cdot sin(\theta_2) \\ &= -d_4 \cdot cos(\theta_2) \cdot cos(\theta_3) + a_2 \cdot sin(\theta_2) + d_4 \cdot sin(\theta_3) \cdot sin(\theta_2) \\ &= -d_4 \cdot \underbrace{\Big(cos(\theta_2) \cdot cos(\theta_3) - sin(\theta_3) \cdot sin(\theta_2)\Big)}_{cos(\theta_2 \cdot \theta_3)} + a_2 \cdot sin(\theta_2)\end{aligned}$$

Verglichen mit Formel 4.11 folgt:

$$\gamma_2 = q_z + d_1$$

[12] Dahmen und Reusken 2008, S. 64

Damit kann der Winkel θ_2 berechnet werden:

$$\theta_2 = arctan2(S_2, C_2) \tag{4.19}$$

Auch für diesen Winkel existieren wieder zwei Lösungen. Damit sind sämtliche Winkel der ersten drei Achsen bestimmt.

Für die Berechnung der Handgelenkwinkel ist die Orientierung des TCP entscheidend. Diese lässt sich aus (4.4) bestimmen. Da die Position des TCP schon bekannt ist und die verbleibenden Winkel nur für die Orientierung verantwortlich sind, werden nur die Rotationsmatrizen ${}^{0}_{6}R$ betrachtet die sich aus ${}^{0}_{3}R$ und ${}^{3}_{6}R$ zusammen setzt.

$$\begin{aligned}
{}^{3}_{6}R = &\begin{bmatrix}
cos(\theta_4)\cdot cos(\theta_5)\cdot cos(\theta_6) - sin(\theta_4)\cdot sin(\theta_6) \\
sin(\theta_4)\cdot cos(\theta_5)\cdot cos(\theta_6) + cos(\theta_4)\cdot sin(\theta_6) \\
-sin(\theta_5)\cdot cos(\theta_6)
\end{bmatrix} \\
&\begin{bmatrix}
-cos(\theta_4)\cdot cos(\theta_5)\cdot sin(\theta_6) - sin(\theta_4)\cdot cos(\theta_6) \\
-sin(\theta_4)\cdot cos(\theta_5)\cdot sin(\theta_6) + cos(\theta_4)\cdot cos(\theta_6) \\
sin(\theta_5)\cdot sin(\theta_6)
\end{bmatrix} \\
&\begin{bmatrix}
cos(\theta_4)\cdot sin(\theta_5) \\
sin(\theta_4)\cdot sin(\theta_5) \\
cos(\theta_5)
\end{bmatrix} \\
= &\begin{bmatrix}
C_4\cdot C_5\cdot C_6 - S_4\cdot S_6 & -C_4\cdot C_5\cdot S_6 - S_4\cdot C_6 & C_4\cdot S_5 \\
S_4\cdot C_5\cdot C_6 + C_4\cdot S_6 & -S_4\cdot C_5\cdot S_6 + C_4\cdot C_6 & S_4\cdot S_5 \\
-S_5\cdot C_6 & S_5\cdot S_6 & C_5
\end{bmatrix}
\end{aligned} \tag{4.20}$$

$$\begin{aligned}
{}^{0}_{3}R = &\begin{bmatrix}
cos(\theta_1)\cdot cos(\theta_2)\cdot cos(\theta_3) + (cos(\theta_1) - sin(\theta_2))\cdot sin(\theta_3) \\
sin(\theta_1)\cdot cos(\theta_2)\cdot cos(\theta_3) - sin(\theta_1)\cdot sin(\theta_2)\cdot sin(\theta_3) \\
-sin(\theta_2)\cdot cos(\theta_3) + (-cos(\theta_2)\cdot sin(\theta_3))
\end{bmatrix} \\
&\begin{bmatrix}
-sin(\theta_1) & cos(\theta_1)\cdot cos(\theta_2)\cdot sin(\theta_3) + cos(\theta_1)\cdot sin(\theta_2)\cdot cos(\theta_3) \\
cos(\theta_1) & sin(\theta_1)\cdot cos(\theta_2)\cdot sin(\theta_3) + sin(\theta_1)\cdot sin(\theta_2)\cdot cos(\theta_3) \\
0 & -sin(\theta_2)\cdot sin(\theta_3) + cos(\theta_2)\cdot cos(\theta_3)
\end{bmatrix} \\
= &\begin{bmatrix}
C_1\cdot C_{23} & -S_1 & C_1\cdot S_{23} \\
S_1\cdot C_{23} & C_1 & S_1\cdot S_{23} \\
-S_{23} & 0 & C_{23}
\end{bmatrix}
\end{aligned} \tag{4.21}$$

Die Inverse von ${}^0_3R$ ist gleich ihrer Transponierten:

$$({}^0_3R)^{-1} = \begin{bmatrix} C_1 \cdot C_{23} & S_1 \cdot C_{23} & -S_{23} \\ -S_1 & C_1 & 0 \\ C_1 \cdot S_{23} & S_1 \cdot S_{23} & C_{23} \end{bmatrix}$$

und ergibt mit ${}^3_6R$ die Gleichung:

$$\begin{aligned} &\begin{bmatrix} C_4 \cdot C_5 \cdot C_6 - S_4 \cdot S_6 & -C_4 \cdot C_5 \cdot S_6 - S_4 \cdot C_6 & C_4 \cdot S_5 \\ S_4 \cdot C_5 \cdot C_6 + C_4 \cdot S_6 & -S_4 \cdot C_5 \cdot S_6 + C_4 \cdot C_6 & S_4 \cdot S_5 \\ -S_5 \cdot C_6 & S_5 \cdot S_6 & C_5 \end{bmatrix} \\ &= \begin{bmatrix} C_1 \cdot C_{23} & S_1 \cdot C_{23} & -S_{23} \\ -S_1 & C_1 & 0 \\ C_1 \cdot S_{23} & S_1 \cdot S_{23} & C_{23} \end{bmatrix} \cdot \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \end{aligned} \tag{4.22}$$

Damit lässt sich der Winkel θ_5 über r_{33} berechnen. Denn laut der Gleichung (4.22) ist $C_5 = r_{33}$ und damit beschreibt

$$\theta_{5;1,2} = \pm arccos(r_{33}) \tag{4.23}$$

den Winkel des fünften Gelenks. Aus r_{13} und r_{23} ergibt sich die Beziehung

$$\begin{aligned} C_4 &= \frac{r_{13}}{S_5} \\ S_4 &= \frac{r_{23}}{S_5} \end{aligned}$$

und damit:

$$\theta_4 = arctan2\left(\frac{r_{23}}{S_5}, \frac{r_{13}}{S_5}\right) \tag{4.24}$$

Für θ_6 gilt dann analog:

$$\theta_6 = arctan2\left(\frac{r_{32}}{S_5}, \frac{r_{31}}{S_5}\right) \tag{4.25}$$

Für beide Winkel existieren wieder zwei Lösungen. Für das ganze System existieren somit vier unterschiedliche Lösungen. Diese verschiedenen Möglichkeiten können genutzt werden, um Hindernisse oder Gelenkbeschränkungen zu „umfahren".

4.5 Steuerung

Um mit dem Roboter interagieren zu können, wurde eine Software mit MATLAB entworfen. Dabei wurde sowohl die Vorwärtstransformation als auch die Inverse Berechnung der Gelenkwinkel umgesetzt. Der kommentierte Code ist dieser Arbeit im Anhang A angehängt. Sämtliche Dateien sind dem digitalem Datenträger zu entnehmen.
Die Software ermöglicht es den Roboter manuell oder durch die Eingabe der Zielkoordinaten zur gewünschten Position zu bewegen. Dabei werden die eingegebenen oder berechneten Gelenkwinkel in ein Steuersignal für den Arduino umgerechnet und übertragen.

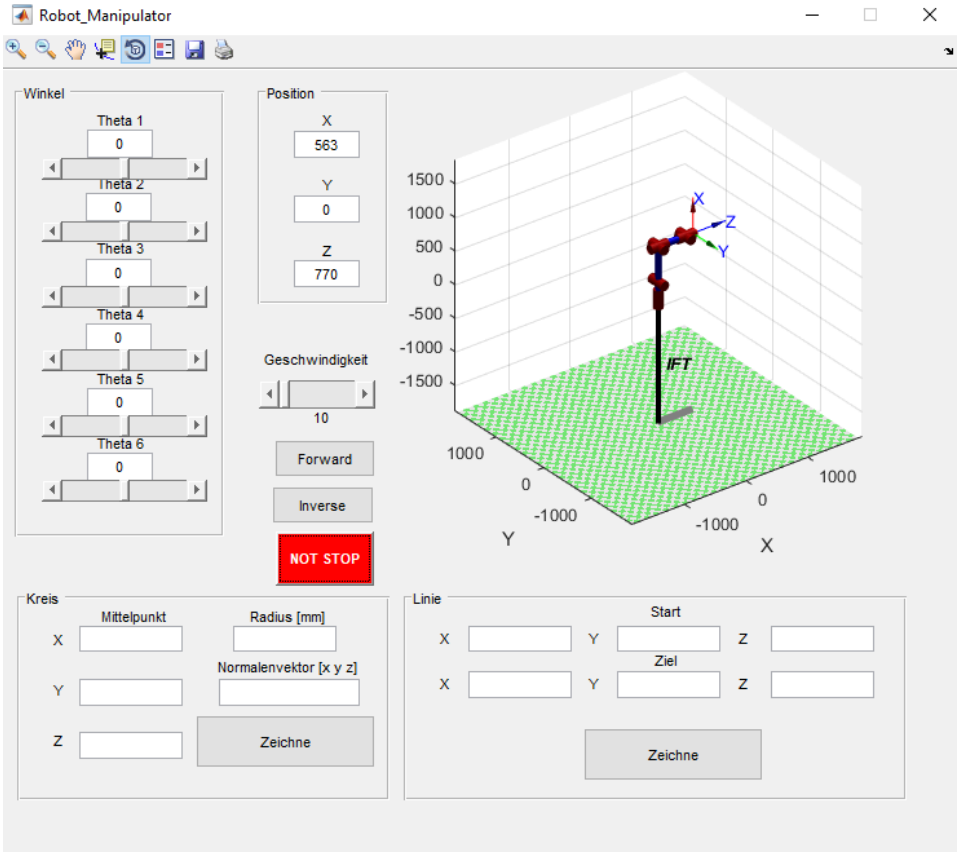

Abbildung 4.19: GUI in MATLAB

Die Grafische Benutzeroberfläche (engl. graphical user interface Abk.: GUI) bietet die Möglichkeit, im Abschnitt „Winkel" jede Achse einzeln zu verstellen. Dies ist entweder durch die direkte Eingabe im Textfeld möglich oder mittels Schieberegler. Nach der Eingabe der Werte wird automatisch die Animation angepasst und ein Befehl an den Arduino übertragen. Die Geschwindigkeit der Motoren lässt sich über einen Schieberegler von 10%-100% einstellen. Im inversen Fall werden die Koordinaten des TCP in die Felder im Abschnitt „Position" eingegeben. Durch die

Schaltfläche „Inverse” werden die Gelenkwinkel berechnet und die Übertragung an den Arduino gestartet. Für das Verfahren des TCP in einer Kreisbahn können unter dem Abschnitt „Kreis” die erforderlichen Parameter eingegeben werden. Diese sind neben dem Mittelpunkt und dem Radius auch der Normalenvektor auf die Kreisfläche. Nach dem Betätigen der Schaltfläche „Zeichne” wird die Kreisbahn in einzelne Punkte unterteilt und die benötigten Gelenkwinkel mittels inverser Kinematik berechnet. Sobald die Gelenkwinkel für alle Punkte erstellt wurden, werden diese zusammen mit der gewählten Geschwindigkeit an den Arduino übertragen. Für eine Linie muss unter dem Abschnitt „Linie” der Start- und Zielpunkt angegeben werden. Daraus werden wieder einzelne Punkte entlang der Linie berechnet und die dazugehörigen Gelenkwinkel invers berechnet.
In allen Fällen lässt sich die Bewegung des Roboters durch das Betätigen des Not-Stopp Schalters auf dem Steckboard oder der Schaltfläche „NOT STOPP” stoppen. Die Motoren werden dabei nicht spannungsfrei geschaltet sondern werden weiterhin für das Halten der Position mit Strom versorgt.

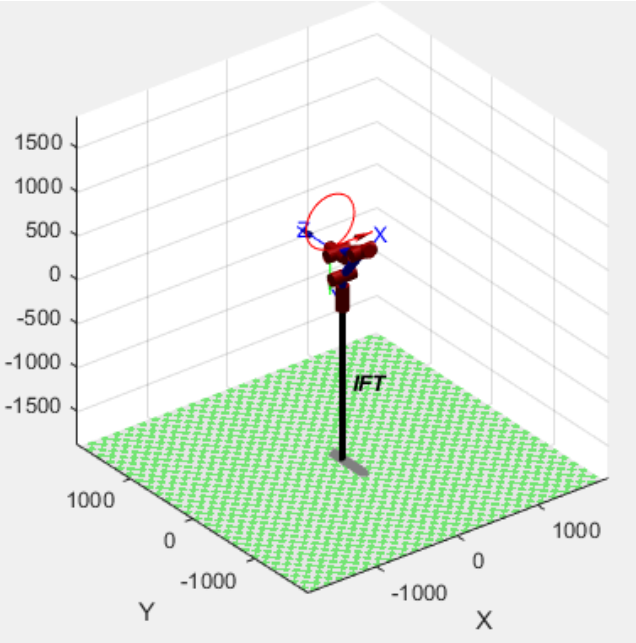

Abbildung 4.20: Darstellung der Bahn des TCP

Bei der Eingabe einer Bahn über den Abschnitt „Kreis” oder „Linie” wird vor der Ausführung des Programms durch den Arduino zunächst, wie in Abbildung 4.20, der Weg des Endeffektors visualisiert. Dadurch ist es möglich, etwaige Kollisionen im Vorfeld zu erkennen und das Programm über die Nothalt Funktion zu stoppen. Nach dem Ausführen der Bewegung werden die aktuellen Gelenkwinkel der Motoren in die Felder im Bereich „Winkel” geschrieben und die Positionswerte aktualisiert.

4.6 Sicherheitseinrichtungen

Die Eingänge des Arduino ermöglichen den Anschluss von unterschiedlichen Sensoren. Digitale Signale von Schaltvorrichtungen, wie Lichtschranken oder Schalter können über die digitalen Eingänge des Arduinos (vgl. Abb.: 4.21) angeschlossen werden. Für analoge Sensoren, wie Ultraschallsensoren oder Dehnmessstreifen, stehen fünf analoge Eingänge zur Verfügung.

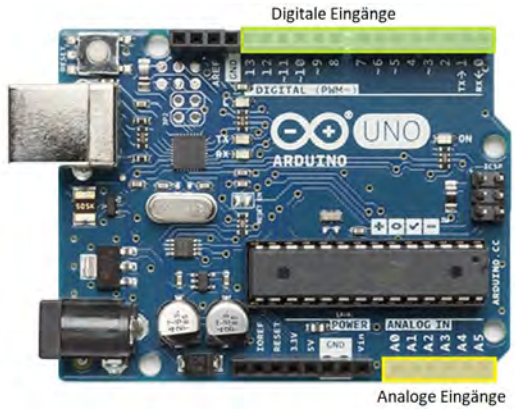

Abbildung 4.21: Eingänge Arduino

Um im Notfall den Roboter zu einem sicheren Stillstand zu bringen, wurde eine Nothalt-Funktion eingebaut. Diese stoppt die Motoren und hält sie in der letzten Position. Durch den Motortreiber werden die Motoren weiter mit Strom versorgt, um die Last zu halten und eine Verletzung durch ein Kollidieren des Roboters zu verhindern.

Abbildung 4.22: Ultraschallsensor

Durch die gewählte Konstruktion befinden sich an der Außenseite des Roboters keine scharfen oder spitzen Kanten. Die Drehachsen und deren Antrieb wurden im Inneren des Roboters verbaut um auch hier eine Interaktion zu vermeiden.
Neben diesen konstruktiven Maßnahmen bestimmt ein Ultraschallsensor (s. Abb.: 4.22) den Abstand zu Personen im Arbeitsbereich des Roboters und drosselt die Geschwindigkeit der Gelenke je nach Abstand zum Menschen bis hin zum Stillstand. Über Leuchtdioden wird der aktuelle Zustand des Roboters für den Bediener angezeigt. So kann zwischen automatik- und kollaborativen-Betrieb unterschieden werden. Fehler im Betrieb eingetretene Notabschaltungen lassen sich ebenfalls damit auch visualisieren.

4.7 Validierung

Die entworfene Steuerung wurde anhand des in der Abbildung 4.23 dargestellten Aufbaus getestet. Dabei wurden die Motortreiber TB6560 mit der Spannungsquelle, je einem Motor und dem Arduino verbunden. Mittels USB wird der Arduino mit dem PC verbunden. Für die Stromversorgung des Arduinos können optional 5V angelegt werden. Die Spannung des USB Anschlusses reicht jedoch für die meisten Anwendungen aus. Um auch im Testbetrieb die Sicherheit gewährleisten zu können, wurde ein Nothalt-Schalter integriert.

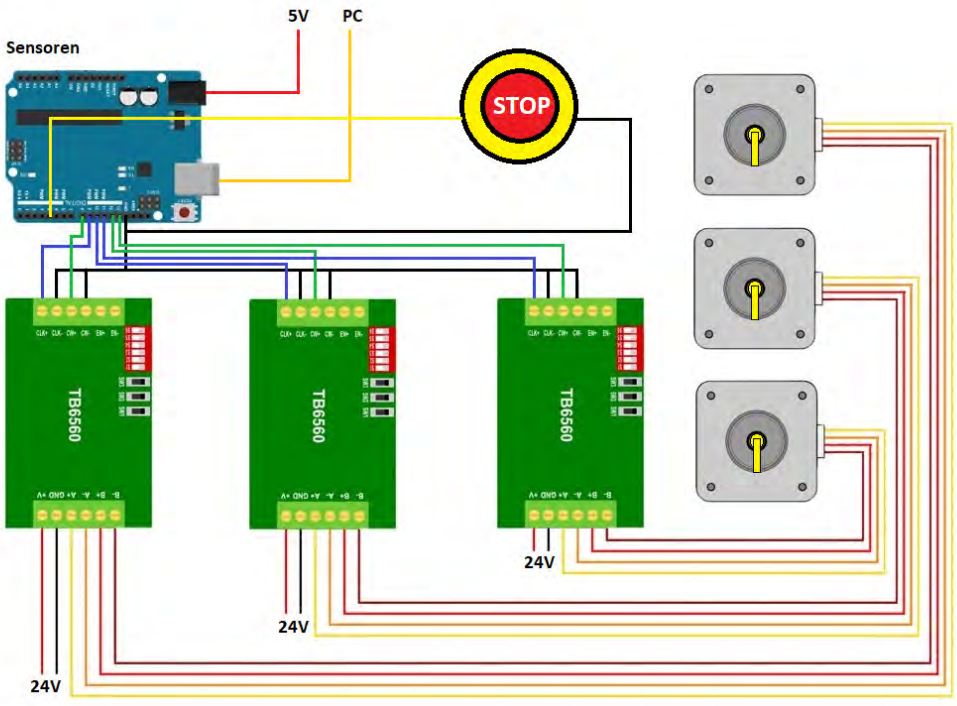

Abbildung 4.23: Testaufbau

Um den Wert des tatsächlich gedrehten Winkels der Motoren besser ablesen zu können, wurden gelbe Fähnchen an den Abtriebswellen montiert. Um auf den späteren Einsatz mit Wellgetriebe zurückschließen zu können wurden die Umdrehungen um die Übersetzungsstufen von 50 reduziert.
Dieser Testaufbau hat gezeigt, dass die Software mit der Berechnung der Winkel und die Kommunikation mit dem Arduino und in Folge auch mit den Motortreibern funktioniert.

5 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war die Entwicklung eines kollaborativen Roboter angefangen bei der Konstruktion des Roboters bis hin zur Entwicklung der Software. Damit sollte die Grundlage für weitere Forschung am IFT zum Thema kollaborativer Robotik geschaffen werden. Daher ergab sich die Forderung nach der Verwendung von „offenen" Komponenten, die frei konfigurierbar sind.

5.1 Zusammenfassung der Ergebnisse

Ein Gelenkarm-Roboter, der für den kollaborativen Einsatz geeignet ist, wurde konstruiert. Sämtliche Kabel und Aktoren sind dabei in das Gehäuse integriert, um Verletzungen vorzubeugen. Sämtliche Achsen werden durch die Kombination eines Schrittmotors und dem IFT-Wellgetriebe angetrieben. Der Arbeitsraum des Gelenkarmes beschränkt sich auf eine Halbkugel vom Radius 1333 mm. Damit eine analytische Lösung des inversen Problems möglich ist schneiden sich die drei Achsen der Hand in einem Punkt. Dies erlaubt die getrennte Berechnung der Achsen eins bis drei von den Achsen vier bis sechs.
Auf Basis des konstruierten Roboters wurden die Werte der erforderlichen Denavit-Hartenberg Parameter ermittelt. Damit lässt sich die Transformation zwischen den einzelnen Gelenken beschreiben. Die aktuelle Position des TCP lässt sich durch das Einfügen der aktuellen Gelenkwinkel in die erstellte Matrix der Vorwärtstransformation ermitteln.
Für den umgekehrten Fall wurde das inverse Problem der Gelenkwinkelberechnung analytisch gelöst. Durch geometrische Beziehungen wurden die einzelnen Gelenkwinkel schrittweise berechnet. Am Ende stehen vier unterschiedliche Lösungen des Problems zur Verfügung, wenn die Orientierung der dritten Achse festgelegt wird.
Um eine höhere Spannung als die vom Arduino berreitgestellten 5 V für die Motoren zu nutzen wird ein Motortreiber verwendet. Dieser schaltet die Spannung einer externen Spannungsquelle den Signalen des Arduinos entsprechend auf die Wicklungen der Schrittmotoren.
Die Software wurde mit Matlab entwickelt und berechnet mit den vorher aufgestellten Matrizen und dem Gleichungssystem die erforderlichen Schritten der Schrittmotoren. Die erstellte grafische Benutzeroberfläche bietet verschiedene Eingabemöglichkeiten, um mit dem Roboter zu interagieren. Ein Ultraschallsensor ermittelt die

Entfernung zu Personen in der Umgebung und reguliert die Geschwindigkeit. Der verbaute Not-Halt Schalter bietet die Möglichkeit, das Programm jederzeit zu unterbrechen. Dabei werden die Motoren weiterhin mit Strom versorgt, um die Last weiterhin tragen zu können.
Die Software wurde erfolgreich mit einer Konfiguration von drei Motoren inklusive Motortreiber getestet.

5.2 Ausblick

Um den in dieser Arbeit konzipierten Cobot weiter zu entwickeln sind weitere Folgearbeiten möglich. Eine Auswahl von möglichen Erweiterungen des Systems sind nachfolgend aufgelistet.

- Zur Zeit werden für die Ansteuerung der Motoren pro Motor ein **Motortreiber** eingesetzt. Durch die Integration von sechs Treibern zu einem kann ein Spannungseingang alle Motoren versorgen und die Einstellung für Haltestrom und Schrittteilung global vorgenommen werden. Eine Kühlung des integrierten Motortreibers ist dabei vorzusehen.

- Die genutzte Programmierumgebung von MATLAB bietet zwar den Vorteil der integrierten Bibliotheken für den Arduino und die Visualisierung, jedoch kommt es teilweise zu langen Rechenzeiten. Für eine **Echtzeitfähigkeit** des Systems wäre eine Umsetzung in einer Hochsprache wie zum Beispiel C++ sinnvoll. Damit sollte es möglich sein, die Software genauer auf die geforderten Aufgaben anzupassen und dadurch effizienter zu gestalten.

- Die Hinderniserkennung wird zur Zeit durch einen Ultraschallsensor realisiert. Bei Hindernissen auf der Bahn des Roboters wird zunächst die Geschwindigkeit gedrosselt und bei anhaltender Behinderung ein Not-Halt eingeleitet. Um in beschränkten Arbeitsräumen arbeiten zu können, sollten bekannte Hindernisse in der **Bahnplanung** berücksichtigt werden. Darüber hinaus wäre eine dynamische Bahnplanung ebenfalls denkbar. Diese wäre in der Lage, Hindernisse zu erkennen und eine alternative Route zu nehmen, um an den Zielpunkt zu gelangen.

- Für den Einsatz in einer vernetzten Produktion ist eine **Kommunikation** mit der Zentraleinheit und anderen Maschinen notwendig. Eine zu entwickelnde Schnittstelle innerhalb der Software kann den Zustand des Roboters bereitstellen und Befehle von außen entgegennehmen. Es sind verschiedene Möglichkeiten der Datenübertragung denkbar. Eine kabellose Übertragungsmöglichkeit ist dabei für den mobilen Einsatz innerhalb der Produktion sinnvoll.

- Um die **Sensitivität** des Roboters weiter zu steigern können Dehnmessstreifen an den Gelenken eingebrachte Kräfte messen. Damit können Kollisionen des Roboters schnell erkannt werden und schwere Beschädigungen verhindert werden. Für die Zusammenarbeit mit dem Menschen können damit schon leichte Berührungen erkannt werden. Die analogen Eingänge des Arduinos ermöglichen eine einfache Integration in die Software.

- Für eine **online Programmierung** sind in den Gelenken absolute Winkelgeber vorzusehen. Damit lassen sich durch manuelles Führen des Roboters die gewünschten Positionen und Bahnverläufe abspeichern. Diese Informationen können für die Erstellung des Programms genutzt werden. Einfache Abläufe lassen sich so schnell und ohne Programmierkenntnisse abspeichern. Die aktuelle Position wird bei den meisten Winkelgeber digital übertragen und ermöglichen damit die Verarbeitung über die digitalen Eingänge es Arduinos.

Literatur

AschG (o.D.). *ArbeitnehmerInnenschutzgesetz*. Gesetz. Republik Österreich.

ABB AG (2015). *YuMi 1400 Overview*. URL: https://robo-hunter.com/uploads/files/554616d422f1e.pdf.

Alfons Botthof, Ernst Andreas Hartmann (2014). *Zukunft der Arbeit in Industrie 4.0*. Springer Vieweg. 172 S. ISBN: 3662459140. URL: %5Curl%7Bhttp://www.ebook.de/de/product/23529707/zukunft_der_arbeit_in_industrie_4_0.html%7D.

Arduino (2017). *Arduino UNO*. URL: https://store.arduino.cc/arduino-uno-smd-rev3.

Arens, Tilo u. a. (2015). *Mathematik*. Springer-Verlag GmbH. ISBN: 3642449182. URL: http://www.ebook.de/de/product/24197721/tilo_arens_frank_hettlich_christian_karpfinger_ulrich_kockelkorn_klaus_lichtenegger_mathematik.html.

Corke, Peter (2017). *Robotics, Vision and Control*. Springer.

Dahmen, Wolfgang und Arnold Reusken (2008). *Numerik für Ingenieure und Naturwissenschaftler*. Springer Berlin Heidelberg. URL: http://www.ebook.de/de/product/25408475/wolfgang_dahmen_arnold_reusken_numerik_fuer_ingenieure_und_naturwissenschaftler.html.

DIN61340 (2015). *Elektrostatik - Teil 5-1: Schutz von elektronischen Bauelementen gegen elektrostatische Phänomene - Allgemeine Anforderungen*. Norm. DIN.

Gerke, Wolfgang (2014). *Technische Assistenzsysteme*. Gruyter, de Oldenbourg. ISBN: 3110343703. URL: %5Curl%7Bhttp://www.ebook.de/de/product/21736803/wolfgang_gerke_technische_assistenzsysteme.html%7D.

Haas, Univ.-Prof. Dipl.-Ing. Dr.techn. Franz (2016). "Skript: Industrieroboter".

Hesse, Stefan und Viktorio Malisa (2016). *Taschenbuch Robotik - Montage - Handhabung*. Hanser Fachbuchverlag. 614 S. ISBN: 3446443657. URL: %5Curl%7Bhttp://www.ebook.de/de/product/25486160/stefan_hesse_viktorio_malisa_taschenbuch_robotik_montage_handhabung.html%7D.

Honerkamp, Josef und Hartmann Römer (1993). *Klassische Theoretische Physik: Eine Einführung (Springer-Lehrbuch) (German Edition)*. Springer. ISBN: 3-540-55901-9. URL: https://www.amazon.com/Klassische-Theoretische-Physik-Einf%C3%BChrung-Springer-Lehrbuch/dp/3540559019?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3540559019.

International Federation Of Robotics (2016). *Robotics-World-Survey*. URL: http://www.focusonfof.eu/downloads/news/robotics-world-survey.pdf.

ISO10218-1 (2012). *Industrieroboter - Sicherheitsanforderungen - Teil 1: Roboter*. Norm. ISO.

ISO10218-2 (2011). *Industrieroboter - Sicherheitsanforderungen - Teil 2: Robotersysteme und Integration*. Norm. ISO.

ISO12100 (2010). *Sicherheit von Maschinen - Allgemeine GestaltungsleitsÃ€tze - Risikobeurteilung und Risikominderung*. Norm. ISO.

ISO13850 (2015). *Sicherheit von Maschinen - Not-Halt*. Norm. ISO.

ISO20653 (2013). *Straßenfahrzeuge - Schutzarten (IP-Code) - Schutz gegen fremde Objekte, Wasser und Kontakt - Elektrische Ausrüstungen*. Norm. ISO.

ISO9283 (1998). *Industrieroboter - Leistungskenngrößen und zugehörige Prüfmethoden*. Norm. ISO.

ISO/TS15066 (2016). *Roboter und Robotikgeräte - Kollaborierende Roboter*. Norm. ISO.

James Edward Colgate, Michael A. Peshkin (1999). "Cobots". Englisch. US 5952796 A.

Jörg Wollnack, Dr.-Ing. habil (2007). "Skript: Robotik".

K.-H. Grote, J. Feldhusen (2011). *Dubbel: Taschenbuch für den Maschinenbau (German Edition)*. Springer. ISBN: 978-3-642-17305-9.

KUKA Roboter GmbH (2016). *LBR iiwa Spezifikation*. URL: https://www.kuka.com/-/media/kuka-downloads/imported/48ec812b1b2947898ac2598aff70abc0/spez_lbr_iiwa_de.pdf.

McCloy, D. M. J. Harris D. (1989). *Robotertechnik - Einführung*. VCH Verlagsgesellschaft Weinheim. ISBN: 3527269177. URL: https://www.amazon.com/Robotertechnik-Einf%C3%BChrung-D-Harris-McCloy/dp/3527269177?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3527269177.

Nof, Shimon Y., Hrsg. (1999). *Handbook of Industrial Robotics*. Wiley. ISBN: 0-471-17783-0. URL: https://www.amazon.com/Handbook-Industrial-Robotics-Shimon-Nof/dp/0471177830?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0471177830.

PULS Österreich (2017). *PULS DIMENSION CS10.244*.

Reichenbach, Dr. Matthias (2013). "Merkmale der Mensch-Roboter-Kooperation im produktiven Umfeld". In: *16. IFF-Wissenschaftstage*.

Robert Bosch GmbH (2015). *APAS assistant Flexibler Produktionsassistent*. http://www.faude.de/fileadmin/user_upload/APASassistant_DE_Digital.pdf. URL: http://www.faude.de/fileadmin/user_upload/APASassistant_DE_Digital.pdf.

Robert Bosch Manufacturing Solutions GmbH (2017). *APAS-Broschüre*.

Slatter, Rolf (2002). "Leichtbaugetriebe für Roboter in der Raumfahrt". In: *Antriebstechnik*.

Steidl, Daniela (2011). *Inverse Kinematik*. TU München.

Teresko, John (2004). "Here Come the Cobots!" In: *IndustryWeek*.

TRINAMIC Motion Control GmbH & Co. KG (2011). *QMOT QSH6018 MANUAL*. Hamburg.

Universal Robots A/S (2009). *UR5 User Manual*. (Accessed on 06/29/2017). URL: https://www.universal-robots.com/media/8704/ur5_user_manual_gb.pdf.

VOLV-Lärm (o.D.). *Verordnung über Schutz der Arbeitnehmer/innen vor der Gefährdung durch Lärm und Vibrationen*. Verordnung. Republik Österreich.

Weber, Wolfgang (2017). *Industrieroboter*. Hanser Fachbuchverlag. 242 S. ISBN: 3446433554. URL: %5Curl%7Bhttp://www.ebook.de/de/product/22668743/wolfgang_weber_industrieroboter.html%7D.

Wloka, Dieter (1992). *Robotersysteme 1: Technische Grundlagen (German Edition)*. Springer. ISBN: 3-540-54739-8. URL: %5Curl%7Bhttps://www.amazon.com/Robotersysteme-1-Technische-Grundlagen-German/dp/3540547398?SubscriptionId=0JYN1NVW651KCA56C102&tag=techkie-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=3540547398%7D.

Wüst, Prof. Dr. Klaus (2014). "Skript: Grundlagen der Robotik".

Abbildungsverzeichnis

Anhang

A. Steuerung

```
10 function varargout = Robot_Manipulator(varargin)
11 %% ROBOT_MANIPULATOR MATLAB code for Robot_Manipulator.fig
12 %
13 % GUI Initialisierung
14 gui_Singleton = 1;
15 gui_State = struct('gui_Name',       mfilename, ...
16                    'gui_Singleton',  gui_Singleton, ...
17                    'gui_OpeningFcn', @Robot_Manipulator_OpeningFcn, ...
18                    'gui_OutputFcn',  @Robot_Manipulator_OutputFcn, ...
19                    'gui_LayoutFcn',  [] , ...
20                    'gui_Callback',   []);
21 if nargin && ischar(varargin{1})
22     gui_State.gui_Callback = str2func(varargin{1});
23 end
24
25 if nargout
26     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
27 else
28     gui_mainfcn(gui_State, varargin{:});
29 end
30 % End GUI Initialisierung
31
32
33 %% Wird ausgeführt kurz vor dem Erscheinen der GUI
34 function Robot_Manipulator_OpeningFcn(hObject, eventdata, handles, varargin)
35 handles.output = hObject;
36 guidata(hObject, handles);
37
38 % Einführen der globalen Variabeln
39 global Robot            %speichert die Roboterkonfiguration
40 global Step             %Schrittweite der Motoren
41 global Theta1OLD        %Speichert den aktuellen Winkel 1
42 global Theta2OLD        %Speichert den aktuellen Winkel 2
43 global Theta3OLD        %Speichert den aktuellen Winkel 3
44 global Theta4OLD        %Speichert den aktuellen Winkel 4
45 global Theta5OLD        %Speichert den aktuellen Winkel 5
46 global Theta6OLD        %Speichert den aktuellen Winkel 6
47 global Hz               %Frequenz (Geschwindigkeit) der Motoren
48 global Freigabe2        %Not-Stopp Signal des GUI Buttons
49
50 %Geschwindigkeit als Frequenz
51 Hz = str2double(handles.textSpeed.String);
52
53 %Anfangswinkel auf 0°
54 Theta1OLD=0;
55 Theta2OLD=0;
```

```
Theta3OLD=0;
Theta4OLD=0;
Theta5OLD=0;
Theta6OLD=0;
Freigabe2=1;

%Schrittweite der Motoren
Step=1.8;

% DH-Parameter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%|theta| d | a | alpha |Offset|%
%------------------------------%
%|  1  |240| 0 | -90°  |   0  |%
%|  2  | 0 |530|   0°  |  90  |%
%|  3  | 0 | 0 |  90°  |   0  |%
%|  4  |440| 0 | -90°  |   0  |%
%|  5  | 0 | 0 |  90°  |   0  |%
%|  6  |123| 0 |   0°  |   0  |%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%DH-Parameter Roboter ungleich 0
d_1 = 240;
a_2 = 530;
d_4 = 440;
d_6 = 123;

% Erstellen des Roboter-Objekt
L (1) = Link([0 d_1 0  pi/2 0 0]);
L (2) = Link([ 0 0 a_2 0 0 pi/2]);
L (3) = Link([0 0 0 pi/2 0 0]);
L (4) = Link([ 0 d_4 0 -pi/2 0 0]);
L (5) = Link([ 0 0 0 pi/2 0 0]);
L (6) = Link([0 d_6  0 0 0 0]);

%Definiert Roboter nach DH
Robot = SerialLink(L);
Robot.name = 'IFT';
Robot.plot([0 0 0 0 0 0]);


%% Not-Stopp Wert setzten für GUI Button
function setGlobalFreigabe2(var)
global Freigabe2
Freigabe2 = var;


%% Setzt Freigabe2 auf 0 wenn GUI Not-Stopp Schaltfläche gedrückt.
function NotStop_Callback(hObject, eventdata, handles)
setGlobalFreigabe2(0);
 h = msgbox('NOT HALT', 'NOTHALT','Warn');
           playTone(a,'D3',0,0);
           writePWMDutyCycle(a,'D9',0);
           writePWMDutyCycle(a,'D10',0);
           writePWMDutyCycle(a,'D11',0);
```

```
111
112
113 %% Ausgabe in Kommandozeile
114 function varargout = Robot_Manipulator_OutputFcn(hObject, eventdata, handles)
115 varargout{1} = handles.output;
116
117
118 %% Wird ausgeführt wenn der INVERSE Button gedrückt wird
119 function Inverse_Callback(hObject, eventdata, handles)
120
121 %Initialisiert die benötigten Global Werte
122 global Step
123 global Theta1OLD
124 global Theta2OLD
125 global Theta3OLD
126 global Robot
127
128 % Liest die X, Y, Z Koordinaten aus der GUI ein
129 PX = str2double(handles.Pos_X.String);
130 PY = str2double(handles.Pos_Y.String);
131 PZ = str2double(handles.Pos_Z.String);
132
133 % Erstellt die Transformationsmatrix aus den angegebenen Werten
134 T =     [1         0         0      PX
135          0         1         0      PY
136          0         0         1      PZ
137          0         0         0      1]  ;
138
139 J = Robot.ikine(T);
140
141 % Rundet die ThetaX und stellt sie im deg Format dar
142 Theta1=round(J(1)*180/pi);
143 Theta2=round(J(2)*180/pi);
144 Theta3=round(J(3)*180/pi);
145 Theta4=round(J(4)*180/pi);
146 Theta5=round(J(5)*180/pi);
147 Theta6=round(J(6)*180/pi);
148
149 % Stellt die Slider auf die ThetaX Werte
150 handles.slider1.Value = Theta1;
151 handles.slider2.Value = Theta2;
152 handles.slider3.Value = Theta3;
153 handles.slider4.Value = Theta4;
154 handles.slider5.Value = Theta5;
155 handles.slider6.Value = Theta6;
156
157 % Schreibt die Werte in die ThetaX Felder
158 handles.Theta_1.String = Theta1;
159 handles.Theta_2.String = Theta2;
160 handles.Theta_3.String = Theta3;
161 handles.Theta_4.String = Theta4;
162 handles.Theta_5.String = Theta5;
163 handles.Theta_6.String = Theta6;
164
165 % Plottet den Roboter in der GUI mit den Winkeln aus J
```

```
166 Robot.plot(J);
167
168 % Berechnet die Schritte für den Motor
169  x = (Theta1-Theta1OLD)/Step;
170  y = (Theta2-Theta2OLD)/Step;
171  z = (Theta3-Theta3OLD)/Step;
172
173 % Sichert die Gelenkswerte um im nächsten Schritt die Differenz berechnen
174 % zu können
175  Theta1OLD=Theta1;
176  Theta2OLD=Theta2;
177  Theta3OLD=Theta3;
178
179 % Steuert den Motor an
180 DrehMotor(x,y,z);
181
182
183 %% Wird ausgeführt wenn der VORWÄRTS Button gedrückt wird
184 function forward_Callback(hObject, eventdata, handles)
185
186 % Initialisiert die benötigten Global Werte
187 global Robot
188 global Theta1OLD
189 global Theta2OLD
190 global Theta3OLD
191 global Step
192
193 % Liest die Gelenkswinkel aus der GUI ein
194 Th_1 = str2double(handles.Theta_1.String)*pi/180;
195 Th_2 = str2double(handles.Theta_2.String)*pi/180;
196 Th_3 = str2double(handles.Theta_3.String)*pi/180;
197 Th_4 = str2double(handles.Theta_4.String)*pi/180;
198 Th_5 = str2double(handles.Theta_5.String)*pi/180;
199 Th_6 = str2double(handles.Theta_6.String)*pi/180;
200
201 % Stellt den Roboter mit den Gelenkwinkel in der GUI dar
202 Robot.plot([Th_1 Th_2 Th_3 Th_4 Th_5 Th_6]);
203
204 % Berechnet die Transformationsmatrix
205 T = Robot.fkine([Th_1 Th_2 Th_3 Th_4 Th_5 Th_6]*180/pi);
206
207 % Liest die Koordinaten des Endeffektors aus der Transformationsmatrix aus
208 handles.Pos_X.String = num2str(floor(T.t(1)));
209 handles.Pos_Y.String = num2str(floor(T.t(2)));
210 handles.Pos_Z.String = num2str(floor(T.t(3)));
211
212 % Berechnet die Schritte für den Motor
213  x = ((Th_1*180/pi)-Theta1OLD)/Step;
214  Theta1OLD=Th_1*180/pi;
215  y = ((Th_2*180/pi)-Theta2OLD)/Step;
216  Theta2OLD=Th_2*180/pi;
217  z = ((Th_3*180/pi)-Theta3OLD)/Step;
218  Theta3OLD=Th_3*180/pi;
219
220 % Steuert den Motor an
```

```
221 DrehMotor(x,y,z)
222
223
224 %% Motoren Anschlüsse Arduino
225 % --- CW  --- Richtung
226 % --- CLK --- Schritte
227 % Motor 1 CW Anschluss: D11
228 %         CLK Anschluss: D3
229 % Motor 2 CW Anschluss: D9
230 %         CLK Anschluss: D6
231 % Motor 3 CW Anschluss: D10
232 %         CLK Anschluss: D5
233 %% Sensoren Anschlüsse Arduino
234 % --- Ultraschall: A1
235 %
236 %% Dreht die Motoren nacheinander
237 function DrehMotor(x,y,z)
238 % Initialisiert die benötigten Global Werte
239 global Hz
240 global Freigabe2
241
242 % Initialisiert den Arduiona an COM 3
243 a = arduino('com3','uno');
244
245 %Initialisiert Not Aus
246 configurePin(a, 'D12', 'pullup');
247 Freigabe = readDigitalPin(a,'D12');
248
249 % Kompensiert den Fehler des Motors
250 f = 25.6*2;
251
252 % Steuert die Achsen seriell an
253 Achse1(a,f,x,Hz,Freigabe,Freigabe2)
254 Achse2(a,f,y,Hz,Freigabe,Freigabe2)
255 Achse3(a,f,z,Hz,Freigabe,Freigabe2)
256
257 % Löscht das Arduino Objekt
258 clear a;
259
260
261 %% Steuert nur Achse 1
262 function Achse1(a,f,x,Hz,Freigabe,Freigabe2)
263 % Bestimmt die Anzahl der Einträge in Vektor x und damit die Schritte
264 n = size(x,1) ;
265
266 %Entfernung von Ultraschallsensor
267 US = readVoltage(a, 'A1');
268
269 for i=1:n
270     double xi;
271     xi=x(i);
272     double x;
273     x = x(i);
274
275     % Anzahl der Schritte
```

```
276     xi=abs(xi);
277     % Korrektur mit Faktor f
278     xi=xi/f;
279
280         % Überprüft ob NOT-AUS gedrückt und Position schon erreicht
281         while xi > 0 && Freigabe == 1  && Freigabe2 == 1 && US > 3
282            % Bestimmt die Drehrichtung
283                 if x<0
284                 % rückwärts
285                     writePWMDutyCycle(a,'D11', 0);
286                 else
287                 % vorwärts
288                     writePWMDutyCycle(a,'D11',1);
289                 end
290
291              % Steuert Motor mit vorgegebener Frequenz
292               playTone(a,'D3',Hz,0.1);
293               pause(0.025);
294               Freigabe = readDigitalPin(a,'D12');
295
296              %Abstandsmessung
297               US = readVoltage(a,'A1');
298              if US < 3
299                  %Stoppt die Motoren
300                   writePWMDutyCycle(a,'D9',0);
301                   writePWMDutyCycle(a,'D10',0);
302                   writePWMDutyCycle(a,'D11',0);
303                  % Interaktionsfenster
304                   choice = questdlg('Objekt in Arbeitsbereich! Wenn Fehler behoben↙
bitte "Fortsetzten"', ...
305                  'Warnung', ...
306                  'Fortsetzen','Abbruch');
307                  % Antwort
308                  switch choice
309                      case 'Fortsetzen'
310                       disp([choice'Wird fortgesetzt'])
311                       US = 5;%Fehler wird zurückgesetzt
312                      case 'Abbruch'
313                       disp([choice'Abbruch durch Benutzer'])
314                       Freigabe = 0;%Not-Stop wird ausgelöst
315                  end
316              end
317
318               xi = xi - 1;
319          end
320
321          % Not-Stopp
322         if Freigabe==0 || Freigabe2==0
323              h = msgbox('NOT HALT', 'NOTHALT','Warn');
324              writePWMDutyCycle(a,'D9',0);
325              writePWMDutyCycle(a,'D10',0);
326              writePWMDutyCycle(a,'D11',0);
327         end
328 end
329     % Schaltet die Motoren aus
```

```
330     playTone(a,'D3',0,0);
331     writePWMDutyCycle(a, 'D11',0);
332
333
334     %% Steuert nur Achse 2
335 function Achse2(a,f,y,Hz,Freigabe,Freigabe2)
336 % Bestimmt die Anzahl der Einträge in Vektor x und damit die Schritte
337 n = size(y,1);
338
339 %Entfernung von Ultraschallsensor
340 US = readVoltage(a, 'A1');
341
342 for i=1:n
343     double yi;
344     yi=y(i);
345     double y;
346     y=y(i);
347
348
349     % Anzahl der Schritte
350     yi=abs(yi);
351     % Korrektur mit Faktor f
352     yi=yi/f;
353
354         % Überprüft ob NOT-AUS gedrückt und Position schon erreicht
355         while yi > 0 && Freigabe == 1 && Freigabe2 == 1 && US > 3
356
357             % Bestimmt die Drehrichtung
358             if y<0
359                 % rückwärts
360                  writePWMDutyCycle(a,'D9', 0);
361             else
362                 % vorwärts
363                  writePWMDutyCycle(a,'D9',1);
364             end
365
366             % Steuert Motor mit vorgegebener Frequenz
367              playTone(a,'D6',Hz,0.1);
368              pause(0.025);
369              Freigabe = readDigitalPin(a,'D12');
370
371             %Abstandsmessung
372              US = readVoltage(a,'A1');
373             if US < 3
374                 %Stoppt die Motoren
375                  writePWMDutyCycle(a,'D9',0);
376                  writePWMDutyCycle(a,'D10',0);
377                  writePWMDutyCycle(a,'D11',0);
378                 % Interaktionsfenster
379                  choice = questdlg('Objekt in Arbeitsbereich! Wenn Fehler behoben↙
bitte "Fortsetzten"', ...
380                 'Warnung', ...
381                 'Fortsetzen','Abbruch');
382                 % Antwort
383                 switch choice
```

```
384                     case 'Fortsetzen'
385                      disp([choice'Wird fortgesetzt'])
386                      US = 5;%Fehler wird zurückgesetzt
387                     case 'Abbruch'
388                      disp([choice'Abbruch durch Benutzer'])
389                      Freigabe = 0;%Not-Stop wird ausgelöst
390                 end
391             end
392          yi = yi - 1;
393          end
394
395        %Not-Stopp
396        if Freigabe==0 || Freigabe2==0
397            h = msgbox('NOT HALT', 'NOTHALT','Warn');
398            writePWMDutyCycle(a,'D9',0);
399            writePWMDutyCycle(a,'D10',0);
400            writePWMDutyCycle(a,'D11',0);
401        end
402
403 end
404 % Schaltet den Motor aus
405 playTone(a,'D6',0,0);
406 writePWMDutyCycle(a,'D9',0);
407
408
409 %% Steuert nur Achse 3
410 function Achse3(a,f,z,Hz,Freigabe, Freigabe2)
411 % Bestimmt die Anzahl der Einträge in Vektor x und damit die Schritte
412 n = size(z,1);
413
414 %Entfernung von Ultraschallsensor
415 US = readVoltage(a,'A1');
416
417 for i=1:n
418     double zi;
419     zi=z(i);
420     double z;
421     z=z(i);
422
423     % Anzahl der Schritte
424     zi=abs(zi);
425     % Korrektur mit Faktor f
426     zi=zi/f;
427
428     % Überprüft ob NOT-AUS gedrückt und Position schon erreicht
429     while zi > 0 && Freigabe == 1 && Freigabe2 == 1 && US > 3
430          % Bestimmt die Drehrichtung
431         if z<0
432             % rückwärts
433              writePWMDutyCycle(a,'D10', 0);
434         else
435             % vorwärts
436              writePWMDutyCycle(a,'D10',1);
437         end
438
```

```
439         % Steuert Motor mit vorgegebener Frequenz
440         playTone(a,'D5',Hz,0.1);
441         pause(0.025);
442         Freigabe = readDigitalPin(a,'D12');
443 
444         %Abstandsmessung
445         US = readVoltage(a,'A1');
446             if US < 3
447                 %Stoppt die Motoren
448                 writePWMDutyCycle(a,'D9',0);
449                 writePWMDutyCycle(a,'D10',0);
450                 writePWMDutyCycle(a,'D11',0);
451                 % Interaktionsfenster
452                 choice = questdlg('Objekt in Arbeitsbereich! Wenn Fehler behoben
bitte "Fortsetzten"', ...
453                 'Warnung', ...
454                 'Fortsetzen','Abbruch');
455                 % Antwort
456                 switch choice
457                     case 'Fortsetzen'
458                     disp([choice'Wird fortgesetzt'])
459                     US = 5;%Fehler wird zurückgesetzt
460                     case 'Abbruch'
461                     disp([choice'Abbruch durch Benutzer'])
462                     Freigabe = 0;%Not-Stop wird ausgelöst
463                 end
464             end
465         zi = zi - 1;
466     end
467 
468     % Not-Stopp
469     if Freigabe==0 || Freigabe2==0
470             h = msgbox('NOT HALT', 'NOTHALT','Warn');
471             writePWMDutyCycle(a,'D9',0);
472             writePWMDutyCycle(a,'D10',0);
473             writePWMDutyCycle(a,'D11',0);
474     end
475 end
476 % Schaltet den Motor aus
477 playTone(a,'D5',0,0);
478 writePWMDutyCycle(a,'D10',0);
479 
480 
481 %% Steuert alle Motoren simultan an
482 function SimultanMotor (x,y,z)
483     global Hz
484 
485     %Initialisiert den Arduino an COM-Port
486     a = arduino('com3','uno');
487     f = 25.6*2;  % Fehler Motor
488 
489     % Initialisiert Not-Stopp Knopf an Arduino
490     configurePin(a,'D12','pullup');
491     Freigabe = readDigitalPin(a,'D12');
492 
```

```
493     % Legt Schritte für Motoren fest
494         n1 = size(x,1);
495         n2 = size(y,1);
496         n3 = size(z,1);
497         n= max([n1 n2 n3]);
498 
499 for i=1:n
500     %Variablen
501     double xi;
502     double yi;
503     double zi;
504     xi=x(i);
505     yi=y(i);
506     zi=z(i);
507     double x;
508     double y;
509     double z;
510     x=x(i);
511     y=y(i);
512     z=z(i);
513 
514     xi=abs(xi);% Schritte Achse 1
515     xi=xi/f;
516     yi=abs(yi);% Schritte Achse 2
517     yi=yi/f;
518     zi=abs(zi);% Schritte Achse 3
519     zi=zi/f;
520 
521     % Achse-1 ansteuern wenn kein Not Stopp und kein Hindernis
522     while xi > 0 && Freigabe == 1 && Freigabe2 == 1 && US > 3
523         %Drehrichtung
524         if x<0
525             writePWMDutyCycle(a,'D11', 0); %rückwärts = 0
526         else
527             writePWMDutyCycle(a,'D11', 1); %vorwärts = 1
528         end
529 
530         %Motor Ansteuern mit vorgegebener Geschwindigkeit
531         playTone(a,'D3',Hz,0.1);
532         pause(0.025);
533         Freigabe = readDigitalPin(a,'D12');
534 
535         %Abstandsmessung
536         US = readVoltage(a,'A1');
537         if US < 3
538             %Stoppt die Motoren
539             writePWMDutyCycle(a,'D9',0);
540             writePWMDutyCycle(a,'D10',0);
541             writePWMDutyCycle(a,'D11',0);
542 
543             % Interaktionsfenster
544             choice = questdlg(...
545             'Objekt in Arbeitsbereich! Wenn Fehler behoben bitte "Fortsetzten",'↙
...
546             'Warnung','Fortsetzen','Abbruch');
```

```
547             % Antwort
548             switch choice
549                 case 'Fortsetzen'
550                  disp([choice'Wird fortgesetzt'])
551                  US = 5;%Fehler wird zurückgesetzt
552                 case 'Abbruch'
553                  disp([choice'Abbruch durch Benutzer'])
554                  Freigabe = 0;%Not-Stop wird ausgelöst
555             end
556         end
557       xi = xi - 1;
558     end
559
560     % Achse-2 ansteuern wenn kein Not Stopp und kein Hindernis
561     while yi > 0 && Freigabe == 1 && Freigabe2 == 1 && US > 3
562         %Drehrichtung
563         if y<0
564         writePWMDutyCycle(a,'D9', 0);
565         else
566         writePWMDutyCycle(a,'D9', 1);
567         end
568
569         %Motor Ansteuern mit vorgegebener Geschwindigkeit
570         playTone(a,'D6',Hz,0.1);
571         pause(0.025);
572         Freigabe = readDigitalPin(a,'D12');
573
574         %Abstandsmessung
575         US = readVoltage(a,'A1');
576             if US < 3
577                 %Stoppt die Motoren
578                  writePWMDutyCycle(a,'D9',0);
579                  writePWMDutyCycle(a,'D10',0);
580                  writePWMDutyCycle(a,'D11',0);
581                 % Interaktionsfenster
582                  choice = questdlg(...
583                 'Objekt in Arbeitsbereich! Wenn Fehler behoben bitte↙
"Fortsetzten"', ...
584                 'Warnung','Fortsetzen','Abbruch');
585                 % Antwort
586                 switch choice
587                     case 'Fortsetzen'
588                      disp([choice'Wird fortgesetzt'])
589                      US = 5;%Fehler wird zurückgesetzt
590                     case 'Abbruch'
591                      disp([choice'Abbruch durch Benutzer'])
592                      Freigabe = 0;%Not-Stop wird ausgelöst
593                 end
594             end
595     yi = yi - 1;
596     end
597
598     % Achse-3 ansteuern wenn kein Not Stopp und kein Hindernis
599     while zi > 0 && Freigabe == 1 && Freigabe2 == 1 && US > 3
600         %Drehrichtung
```

```
601         if z<0
602             writePWMDutyCycle(a,'D10', 0);
603         else
604             writePWMDutyCycle(a,'D10', 1);
605         end
606         %Motor Ansteuern mit vorgegebener Geschwindigkeit
607         playTone(a,'D5',Hz,0.1);
608         pause(0.025);
609         Freigabe = readDigitalPin(a,'D12');
610 
611         %Abstandsmessung
612         US = readVoltage(a,'A1');
613         if US < 3
614            %Stoppt die Motoren
615             writePWMDutyCycle(a,'D9',0);
616             writePWMDutyCycle(a,'D10',0);
617             writePWMDutyCycle(a,'D11',0);
618             % Interaktionsfenster
619             choice = questdlg('Objekt in Arbeitsbereich! Wenn Fehler behoben
bitte "Fortsetzten"', ...
620             'Warnung', ...
621             'Fortsetzen','Abbruch');
622             % Antwort
623             switch choice
624                   case 'Fortsetzen'
625                    disp([choice'Wird fortgesetzt'])
626                    US = 5;%Fehler wird zurückgesetzt
627                   case 'Abbruch'
628                    disp([choice'Abbruch durch Benutzer'])
629                    Freigabe = 0;%Not-Stop wird ausgelöst
630             end
631         end
632     zi = zi - 1;
633     end
634 
635     % Not-Stopp
636     if Freigabe == 0 || Freigabe2 == 0
637         h = msgbox('NOT HALT', 'NOTHALT','Warn');
638            writePWMDutyCycle(a,'D9',0);
639            writePWMDutyCycle(a,'D10',0);
640            writePWMDutyCycle(a,'D11',0);
641     end
642 
643     % Stoppt die Motoren wenn alle Schritte ausgeführt wurden
644     if n1==n
645         playTone(a,'D3',0,0);
646         elseif n2==n
647         playTone(a,'D6',0,0);
648         elseif n3==n
649         playTone(a,'D5',0,0);
650         else
651     end
652 
653 end
654     writePWMDutyCycle(a,'D11',0);
```

```
655     writePWMDutyCycle(a, 'D9',0);
656     writePWMDutyCycle(a, 'D10',0);
657 clear a;
658
659
660 %% Zeichnet einen Kreis
661 function zeichneKreis_Callback(hObject, eventdata, handles)
662 % Kreisparameter
663 global Robot
664 global Theta1OLD
665 global Theta2OLD
666 global Theta3OLD
667 global Step
668
669 radius = str2double(handles.radiusKreis.String);
670
671 % Unterteilung des Kreises in X Punkte
672 n=25;
673
674 % Mittelpunkt des Kreises
675 x=str2double(handles.xKreis.String);
676 y=str2double(handles.yKreis.String);
677 z=str2double(handles.zKreis.String);
678 INI = transl(x,y,z);
679
680 % Erzeugen der Kreisbahn
681 Ebene = str2num(handles.nVektor.String);
682 xRadius= Ebene(1)*radius;
683 yRadius= Ebene(2)*radius;
684 zRadius= Ebene(3)*radius;
685 center=[x y z];
686 normal = [Ebene(1) Ebene(2) Ebene(3)];
687
688 % Darstellung Kreis
689 theta=0:0.01:2*pi; %Punkte des Kreises von 0-2pi in 0.1 Schritten
690 v=null(normal);
691 points=repmat(center',1,size(theta,2))+radius*(v(:,1)*cos(theta)+v(:,2)...
692         *sin(theta));
693 plot3(points(1,:),points(2,:),points(3,:),'r-');
694
695  for i=1:n
696      if Ebene(1)== 1 && Ebene(2)==1
697      T1(:,:,i)= INI*(Ebene(1)*trotz(-pi/2)*Ebene(2)*trotx(-pi/2))*(Ebene(1)*troty
(2*pi*i/n)*Ebene(2)*trotz(2*pi*i/n))*transl(xRadius, yRadius, zRadius);
698      elseif Ebene(1)== 1 && Ebene(3)==1
699      T1(:,:,i)= INI*(Ebene(1)*trotz(-pi/2)*Ebene(3)*trotx(-pi/2))*(Ebene(1)*troty
(2*pi*i/n)*Ebene(3)*trotz(2*pi*i/n))*transl(xRadius, yRadius, zRadius);
700      elseif Ebene(2)== 1 && Ebene(3)==1
701      T1(:,:,i)= INI*(Ebene(2)*trotz(-pi/2)*Ebene(3)*trotx(-pi/2))*(Ebene(2)*troty
(2*pi*i/n)*Ebene(3)*trotz(2*pi*i/n))*transl(xRadius, yRadius, zRadius);
702      else
703      T1(:,:,i)= INI*(Ebene(1)*trotz(-pi/2)+Ebene(2)*trotx(-pi/2)+Ebene(3)*troty(-
pi/2))*(Ebene(1)*troty(2*pi*i/n)+Ebene(2)*trotz(2*pi*i/n)+Ebene(3)*trotx(2*pi*i/n))
*transl(xRadius, yRadius, zRadius);
704      end
```

```
705  end
706 Q1= Robot.ikine(T1);
707
708 % Bewegt den Roboter um Kreis zu erzeugen
709 Robot.plot(Q1);
710
711 % Rundet die ThetaX und stellt sie im deg Format dar
712 Theta1=round(Q1(:,1)*180/pi);
713 Theta2=round(Q1(:,2)*180/pi);
714 Theta3=round(Q1(:,3)*180/pi);
715 Theta4=round(Q1(:,4)*180/pi);
716 Theta5=round(Q1(:,5)*180/pi);
717 Theta6=round(Q1(:,6)*180/pi);
718
719 % Stellt die Slider auf die ThetaX Werte
720 handles.slider1.Value = Theta1(n);
721 handles.slider2.Value = Theta2(n);
722 handles.slider3.Value = Theta3(n);
723 handles.slider4.Value = Theta4(n);
724 handles.slider5.Value = Theta5(n);
725 handles.slider6.Value = Theta6(n);
726
727 % Schreibt die Werte in die ThetaX Felder
728 handles.Theta_1.String = Theta1(n);
729 handles.Theta_2.String = Theta2(n);
730 handles.Theta_3.String = Theta3(n);
731 handles.Theta_4.String = Theta4(n);
732 handles.Theta_5.String = Theta5(n);
733 handles.Theta_6.String = Theta6(n);
734
735 % Anzahl der Schritte wird berechnet
736  x = (Theta1-Theta1OLD)/Step;
737  y = (Theta2-Theta2OLD)/Step;
738  z = (Theta3-Theta3OLD)/Step;
739
740  % AlterWinkel wird auf den Wert des aktuelle Winkels gesetzt
741  Theta1OLD=Theta1(n);
742  Theta2OLD=Theta2(n);
743  Theta3OLD=Theta3(n);
744
745  % Steuert die Motoren an
746 SimultanMotor(x,y,z);
747
748
749 %% Zeichnet eine Linie
750 function zeichneLinie_Callback(hObject, eventdata, handles)
751 %% Linien-Parameter
752 global Robot
753 global Step
754 global Theta1OLD
755 global Theta2OLD
756 global Theta3OLD
757
758 % Linie zeichnen
759 xStart=str2double(handles.xStartL.String);
```

```
760 yStart=str2double(handles.yStartL.String);
761 zStart=str2double(handles.zStartL.String);
762
763 xZiel=str2double(handles.xZielL.String);
764 yZiel=str2double(handles.yZielL.String);
765 zZiel=str2double(handles.zZielL.String);
766
767 t=[0:0.1:1];
768
769 StartPoint=transl([xStart,yStart,zStart]);
770 EndPoint=transl([xZiel,yZiel,zZiel]);
771
772 Linie=round(ctraj(StartPoint,EndPoint,length(t)));
773
774 path=[xStart yStart zStart
775         xZiel yZiel zZiel];
776
777 J0 = Robot.ikine(StartPoint);
778 plot3(path(:,1),path(:,2),path(:,3),'color','b','lineWidth',1);
779 Robot.plot(J0)
780
781 J1 = Robot.ikine(Linie);
782
783 Robot.plot(J1);
784
785 % Rundet die ThetaX und stellt sie im deg Format dar
786 Theta1=round(J1(:,1)*180/pi);
787 Theta2=round(J1(:,2)*180/pi);
788 Theta3=round(J1(:,3)*180/pi);
789 Theta4=round(J1(:,4)*180/pi);
790 Theta5=round(J1(:,5)*180/pi);
791 Theta6=round(J1(:,6)*180/pi);
792
793 % Stellt die Slider auf die ThetaX Werte
794 handles.slider1.Value = Theta1(6);
795 handles.slider2.Value = Theta2(6);
796 handles.slider3.Value = Theta3(6);
797 handles.slider4.Value = Theta4(6);
798 handles.slider5.Value = Theta5(6);
799 handles.slider6.Value = Theta6(6);
800
801 % Schreibt die Werte in die ThetaX Felder
802 handles.Theta_1.String = Theta1(6);
803 handles.Theta_2.String = Theta2(6);
804 handles.Theta_3.String = Theta3(6);
805 handles.Theta_4.String = Theta4(6);
806 handles.Theta_5.String = Theta5(6);
807 handles.Theta_6.String = Theta6(6);
808
809 % Anzahl der Schritte wird berechnet
810  x = (Theta1-Theta1OLD)/Step;
811  y = (Theta2-Theta2OLD)/Step;
812  z = (Theta3-Theta3OLD)/Step;
813
814  % AlterWinkel wird auf den Wert des aktuelle Winkels gesetzt
```

```
 815  Theta1OLD=Theta1(6);
 816  Theta2OLD=Theta2(6);
 817  Theta3OLD=Theta3(6);
 818
 819  % Steuert Motoren an
 820 SimultanMotor(x,y,z);
 821
 822 %% Setzt den Wert von Hz auf die Geschwindigkeit
 823 function speed_Callback(hObject, eventdata, handles)
 824 global Hz
 825 var = hObject.Value;
 826 Hz = hObject.Value;
 827 var=var/10;
 828 var = num2str(var);
 829 set(handles.textSpeed, 'String', var);
 830
 831 function speed_CreateFcn(hObject, eventdata, handles)
 832 if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 833     set(hObject,'BackgroundColor',[.9 .9 .9]);
 834 end
 835
 836
 837 %% Durch Eingaben von Theta_X wird eine Vorwärtstransformation ausgeführt
 838 function Theta_1_Callback(hObject, eventdata, handles)
 839 val =str2double(get(handles.Theta_1,'String'));
 840 set(handles.slider1, 'Value', val);
 841 forward_Callback (hObject, eventdata, handles);
 842
 843 function Theta_1_CreateFcn(hObject, eventdata, handles)
 844 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 845     set(hObject,'BackgroundColor','white');
 846 end
 847
 848 function Theta_2_Callback(hObject, eventdata, handles)
 849 val =str2double(get(handles.Theta_2,'String'));
 850 set(handles.slider2, 'Value', val);
 851 forward_Callback (hObject, eventdata, handles);
 852
 853 function Theta_2_CreateFcn(hObject, eventdata, handles)
 854 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 855     set(hObject,'BackgroundColor','white');
 856 end
 857
 858 function Theta_3_Callback(hObject, eventdata, handles)
 859 val =str2double(get(handles.Theta_3,'String'));
 860 set(handles.slider3, 'Value', val);
 861 forward_Callback (hObject, eventdata, handles);
 862
 863 function Theta_3_CreateFcn(hObject, eventdata, handles)
 864 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 865     set(hObject,'BackgroundColor','white');
```

```
866 end
867
868 function Theta_4_Callback(hObject, eventdata, handles)
869 val =str2double(get(handles.Theta_4,'String'));
870 set(handles.slider4, 'Value', val);
871 forward_Callback (hObject, eventdata, handles);
872
873 function Theta_4_CreateFcn(hObject, eventdata, handles)
874 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
875     set(hObject,'BackgroundColor','white');
876 end
877
878 function Theta_5_Callback(hObject, eventdata, handles)
879 val =str2double(get(handles.Theta_5,'String'));
880 set(handles.slider5, 'Value', val);
881 forward_Callback (hObject, eventdata, handles);
882
883 function Theta_5_CreateFcn(hObject, eventdata, handles)
884 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
885     set(hObject,'BackgroundColor','white');
886 end
887
888 function Theta_6_Callback(hObject, eventdata, handles)
889 val =str2double(get(handles.Theta_6,'String'));
890 set(handles.slider6, 'Value', val);
891 forward_Callback (hObject, eventdata, handles);
892
893 function Theta_6_CreateFcn(hObject, eventdata, handles)
894 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
895     set(hObject,'BackgroundColor','white');
896 end
897
898
899 %% Durch Verschieben der Slider wird eine Vorwärtstransformation ausgeführt
900 function slider1_Callback(hObject, eventdata, handles)
901     var = hObject.Value;
902     var = num2str(var);
903     set(handles.Theta_1, 'String', var);
904     forward_Callback (hObject, eventdata, handles);
905
906 function slider1_CreateFcn(hObject, eventdata, handles)
907 if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
908     set(hObject,'BackgroundColor',[.9 .9 .9]);
909 end
910
911 function slider2_Callback(hObject, eventdata, handles)
912     var = hObject.Value;
913     var = num2str(var);
914     set(handles.Theta_2, 'String', var);
915     forward_Callback (hObject, eventdata, handles);
916
```

```
 917 function slider2_CreateFcn(hObject, eventdata, handles)
 918 if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 919     set(hObject,'BackgroundColor',[.9 .9 .9]);
 920 end
 921
 922 function slider3_Callback(hObject, eventdata, handles)
 923     var = hObject.Value;
 924     var = num2str(var);
 925     set(handles.Theta_3, 'String', var);
 926     forward_Callback (hObject, eventdata, handles);
 927
 928 function slider3_CreateFcn(hObject, eventdata, handles)
 929 if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 930     set(hObject,'BackgroundColor',[.9 .9 .9]);
 931 end
 932
 933 function slider4_Callback(hObject, eventdata, handles)
 934 var = hObject.Value;
 935 var = num2str(var);
 936 set(handles.Theta_4, 'String', var);
 937 forward_Callback (hObject, eventdata, handles);
 938
 939 function slider4_CreateFcn(hObject, eventdata, handles)
 940 if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 941     set(hObject,'BackgroundColor',[.9 .9 .9]);
 942 end
 943
 944 function slider5_Callback(hObject, eventdata, handles)
 945 var = hObject.Value;
 946 var = num2str(var);
 947 set(handles.Theta_5, 'String', var);
 948 forward_Callback (hObject, eventdata, handles);
 949
 950 function slider5_CreateFcn(hObject, eventdata, handles)
 951 if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 952     set(hObject,'BackgroundColor',[.9 .9 .9]);
 953 end
 954
 955 function slider6_Callback(hObject, eventdata, handles)
 956 var = hObject.Value;
 957 var = num2str(var);
 958 set(handles.Theta_6, 'String', var);
 959 forward_Callback (hObject, eventdata, handles);
 960
 961 function slider6_CreateFcn(hObject, eventdata, handles)
 962 if isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 963     set(hObject,'BackgroundColor',[.9 .9 .9]);
 964 end
 965
 966
```

```
 967 %% Erstellt die Felder für die Inverse
 968 function Pos_Z_CreateFcn(hObject, eventdata, handles)
 969 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 970     set(hObject,'BackgroundColor','white');
 971 end
 972
 973 function Pos_Y_CreateFcn(hObject, eventdata, handles)
 974 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 975     set(hObject,'BackgroundColor','white');
 976 end
 977
 978 function Pos_X_CreateFcn(hObject, eventdata, handles)
 979 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 980     set(hObject,'BackgroundColor','white');
 981 end
 982
 983 function Pos_X_Callback (hObject, eventdata, handles)
 984 function Pos_Y_Callback (hObject, eventdata, handles)
 985 function Pos_Z_Callback (hObject, eventdata, handles)
 986
 987
 988 %% Erstellt die Felder für den Kreis
 989 function xKreis_CreateFcn(hObject, eventdata, handles)
 990 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 991     set(hObject,'BackgroundColor','white');
 992 end
 993
 994 function yKreis_CreateFcn(hObject, eventdata, handles)
 995 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
 996     set(hObject,'BackgroundColor','white');
 997 end
 998
 999 function zKreis_CreateFcn(hObject, eventdata, handles)
1000 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
1001     set(hObject,'BackgroundColor','white');
1002 end
1003
1004 function radiusKreis_CreateFcn(hObject, eventdata, handles)
1005 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
1006     set(hObject,'BackgroundColor','white');
1007 end
1008
1009 function nVektor_CreateFcn(hObject, eventdata, handles)
1010 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
1011     set(hObject,'BackgroundColor','white');
1012 end
1013
```

```
1014
1015 %% Erstellt die Felder für Linie
1016 function zStartL_Callback(hObject, eventdata, handles)
1017 function zStartL_CreateFcn(hObject, eventdata, handles)
1018
1019 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
1020     set(hObject,'BackgroundColor','white');
1021 end
1022
1023 function yStartL_Callback(hObject, eventdata, handles)
1024
1025 function yStartL_CreateFcn(hObject, eventdata, handles)
1026 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
1027     set(hObject,'BackgroundColor','white');
1028 end
1029
1030 function xStartL_Callback(hObject, eventdata, handles)
1031
1032 function xStartL_CreateFcn(hObject, eventdata, handles)
1033 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
1034     set(hObject,'BackgroundColor','white');
1035 end
1036
1037 function xZielL_Callback(hObject, eventdata, handles)
1038
1039 function xZielL_CreateFcn(hObject, eventdata, handles)
1040 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
1041     set(hObject,'BackgroundColor','white');
1042 end
1043
1044 function yZielL_Callback(hObject, eventdata, handles)
1045
1046 function yZielL_CreateFcn(hObject, eventdata, handles)
1047 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
1048     set(hObject,'BackgroundColor','white');
1049 end
1050
1051 function zZielL_Callback(hObject, eventdata, handles)
1052
1053 function zZielL_CreateFcn(hObject, eventdata, handles)
1054 if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
1055     set(hObject,'BackgroundColor','white');
1056 end
1057
```

```
%%Inverse Kinematik (PX,PY,PZ)

PX = 563;
PY = 0;
PZ = 770;

%Erstellt die Transformationsmatrix aus den angegebenen Werten
T =     [1
         0
         0
         0

%Berechnet die Gelenkswinkel
%DH Parameter für die Berrechnung
                a2 = 530;
                d1 = 24
                d4 = 44
                d6 = 12
               qx=PX-d6*T(1,3
               qy=PY-d6*T(2,3
               qz=PZ-d6*T(3,3

                theta = zeros(6);
      %Theta 1
                theta(1,:) = atan2(qy,qx);

      %Theta 3
                  k1=2*a2*d
                  k2=qx^2+qy^2+qz^2+2*qz*d1+d1^2-(d4^2+a2^2

                  t1=(k1+sqrt(k1^2-k2^2))/
                  t2=(k1-sqrt(k1^2-k2^2))/

                  theta(3,1:3)=real(2*atan(t1
                  theta(3,4:6)=real(2*atan(t2


      %Theta 2
                   u1=a2+d4*sin(theta(3)
                   u2=-d4*cos(theta(3)
                   v1=d4*cos(theta(3)
                   v2=a2+d4*sin(theta(3)
                   y1=qx*cos(theta(1))-sin(theta(1))*q
                   y2=qz+d
                   S2=round((y1*u2-y2*u1)/(v1+y2-v2*y1)
                   C2=round((y1*v2-y2*v1)/(v2*u1-v1*u2)

                   theta(2,1:3)=atan2(S2,C2)-pi/
                   theta(2,4:6)=-atan2(S2,C2)+pi/
```

```
50
51      %Theta 4 5 6
52                    R30=[ cos(theta(1))*cos(theta(2)+theta(3)) sin(theta(1)*cos
(theta(2)+theta(3))) -sin(theta(2)+theta(3))
53                          -sin(theta(1)) cos(theta(1)) 0
54                          cos(theta(1))*sin(theta(2)+theta(3)) sin(theta(1)*sin
(theta(2)+theta(3))) cos(theta(2)+theta(3))];
55                    R06=R30*T(1:3,1:3);
56                    theta(5,1:3)=acos(R06(9));
57                    theta(5,4:6)=-acos(R06(9));
58                    theta(4,:)=atan2((R06(8)/sin(theta(5))),(R06(7)/sin(theta
(5))));
59                    theta(6,:)=atan2((R06(6)/sin(theta(5))),(R06(3)/sin(theta
(5))));
60
61 %Matrix "Winkel" enthält die 6 Möglichen Winkelkombinationen. Durch
62 %Bauliche gegebenheiten können hier Einschränkungen vorgenommen werden.
63 %Diese sind noch unbekannt und daher wird der erste Vektor der Matrix
64 %gewählt. Q1 enthält die Winkel der ersten Spalte der Matrix "Winkel"
65
66 Winkel=theta*180/pi;
67 Q1=theta(:,1)
68
```