Prototypische Implementierung einer
Handhabungsaufgabe für ein kollaboratives
Laborumfeld mit einem SCARA-Roboter

Simon Rauch
7. Semester, MFB7G

Abgegeben am 01.02.2017
Kurzzusammenfassung


Prototypic implementation of a handling task in a collaborative laboratory environment with a SCARA robot
Abstract

A robot with SCARA kinematics (SCARA: Selective Compliance Assembly Robot Arm) was built to demonstrate the technology know-how of SENSODRIVE GmbH. This robot is equipped with torque sensors that allow a high performance torque control. Torque based effects can be simulated by a torque control, for example virtual friction compensation or haptic effects.

In the course of this thesis, the demonstrator will be updated by a practical application, which will allow a human-robot collaboration. An application from the laboratory environment is selected for this purpose. A sample preparation in the single vessel handling will be realized, including picking, shaking and placing the sample. For the purpose of demonstrating the application field of torque-controlled drives, the application is monitored by a collision detection. This enables a collaborative work of human and robot without separating protective devices. In order to further improve this collaboration, the robot is only moved linearly. The movement sequences are then more predictable for the user. In addition, a new *Teach In* tool will be developed. By guiding the robot with the hand, the user can prescribe and store the target points for the robot.
Inhalt

1 Einleitung .................................................................................................................. 8
  1.1 Problemstellung .................................................................................................. 8
  1.2 Ziel und Gliederung der Arbeit ....................................................................... 9
  1.3 Technologiedemonstrator SCARA-Roboter .................................................. 11
    1.3.1 Mechanischer Aufbau ............................................................................... 11
    1.3.2 Entwicklungsumgebung des Technologiedemonstrators .................... 12
    1.3.3 Bisherige Funktionen des Technologiedemonstrators ....................... 13

2 Kinematik eines SCARA-Roboters ..................................................................... 14
  2.1 Vorwärtskinematik ............................................................................................ 14
    2.1.1 Die Denavit-Hartenberg-Konvention ..................................................... 16
    2.1.2 DH-Parameter des SCARA-Roboters ..................................................... 19
  2.2 Inverse Kinematik ............................................................................................ 21

3 Bahnsteuerung ...................................................................................................... 25
  3.1 Bewegungsarten und Interpolation .................................................................. 25
    3.1.1 PTP-Steuerung ....................................................................................... 25
    3.1.2 CP-Steuerung ....................................................................................... 26
  3.2 Rampenprofil zur Interpolation ......................................................................... 28
  3.3 Sinoidenprofil zur Interpolation ........................................................................ 33
  3.4 Linearinterpolation ............................................................................................ 36
  3.5 Durchfahren von Zwischenpunkten ohne Anhalten ....................................... 39

4 Entwicklung einer flexiblen Teach In Funktion .............................................. 41
  4.1 Notwendigkeit einer flexiblen Teach In Funktion .......................................... 41
  4.2 Positionen speichern ......................................................................................... 42
    4.2.1 Vorüberlegungen ...................................................................................... 42
    4.2.2 Umsetzung ............................................................................................... 44
  4.3 Berechnung aller Positionen .............................................................................. 46
    4.3.1 Rackpositionen ......................................................................................... 46
    4.3.2 Zentrifugenpositionen ................................................................................ 47
  4.4 Positionen sortieren und bereitstellen ............................................................ 48

5 Programmablauf .................................................................................................. 51
  5.1 Steuerung des Greifers .................................................................................... 51
    5.1.1 Referenzfahrt ........................................................................................... 51
    5.1.2 Modus für das Greifen und Absetzen .................................................... 53
    5.1.3 Modus für das Teach In ........................................................................... 54
  5.2 Ablaufsteuerung ................................................................................................ 57
  5.3 Kollisionserkennung ......................................................................................... 59

6 Zusammenfassung .................................................................................................. 61

7 Ausblick .................................................................................................................. 63
Abbildungen
Abbildung 1 – Achsen am SCARA-Roboter ................................................................. 11
Abbildung 2 – Technologiedemonstrator SCARA-Roboter ........................................ 12
Abbildung 3 – Vereinfachte Entwicklungsumgebung Technologiedemonstrator .......... 13
Abbildung 5 – Koordinatensysteme am SCARA-Roboter ........................................ 20
Abbildung 6 – Koordinatensysteme am Technologiedemonstrator ........................... 21
Abbildung 7 – Mögliche Konfigurationen eines SCARA-Roboters .............................. 22
Abbildung 8 – Skizze zur Berechnung der inversen Kinematik .................................. 22
Abbildung 9 – Wahl der Konfiguration anhand des Arbeitsraumes ............................ 23
Abbildung 10 – Dynamisches Umkonfigurieren am SCARA-Roboter ......................... 24
Abbildung 11 – Arbeitsraumverlust bei fester Konfiguration ..................................... 25
Abbildung 12 – Bahnvergleich bei verschiedenen Steuerungsarten [20] ....................... 27
Abbildung 13 – Arbeitsplatzanordnung der Handhabungsaufgabe .............................. 27
Abbildung 14 – Rampenförmiges Geschwindigkeitsprofil ....................................... 28
Abbildung 15 – Geschwindigkeitsprofil für kleine Strecken ...................................... 29
Abbildung 16 – Bewegungsgrößen bei Interpolation mit Rampenprofil ..................... 31
Abbildung 17 – Beschleunigung und Ruck bei Rampeninterpolation .......................... 32
Abbildung 18 – Beschleunigung und Ruck bei Sinoideninterpolation ......................... 33
Abbildung 19 – Bewegungsgrößen bei Interpolation mit Sinoidenprofil ..................... 35
Abbildung 20 – Programmausschnitt Berechnung von Zielwinkel für die Interpolation 37
Abbildung 21 – Flussdiagramm Berechnung von Zwischenpunkten ............................ 38
Abbildung 22 – Geschwindigkeitsprofil bei Linearinterpolation ................................. 39
Abbildung 23 – Programmausschnitt auslösende Ereignisse zur Zwischenpunktberechnung .............................................................. 40
Abbildung 24 – Reihenfolge beim Teach In ................................................................. 44
Abbildung 25 – Ein- und Ausgänge der Funktion savePos ........................................... 45
Abbildung 26 – Flussdiagramm Abspeichern von kartesischen Koordinaten ............... 46
Abbildung 27 – Skizze zur Berechnung der Rackpunkte .......................................... 47
Abbildung 28 – Berechnungsergebnis von geteachten Punkten ............................... 48
Abbildung 29 – Ergebnis der Funktion savePos mit zwei Racks ............................... 48
Abbildung 30 – Überlappung zweier Probenfläschchen in Greifer und Rack ............... 49
Abbildung 31 – Sortierschema zur Kollisionsvermeidung ......................................... 50
Abbildung 32 – Zielvektoren für die Ablaufsteuerung ................................................. 50
Abbildung 33 – Flussdiagramm systematische Funktion des Greifers ....................... 51
Abbildung 34 – Schematischer Aufbau Greifer .......................................................... 52
Abbildung 35 – Programmausschnitt Referenzfahrt Greifer ..................................... 53
Abbildung 36 – Flussdiagramm Funktionsablauf Greifen/Absetzen ........................................... 54
Abbildung 37 – Flussdiagramm Greifermodus für das Teach In ............................................. 55
Abbildung 38 – Programmausschnitt Greifer .............................................................................. 56
Abbildung 39 – Ein- und Ausgänge der Ablaufsteuerung ......................................................... 57
Abbildung 40 – Programmausschnitt Ablaufsteuerung ............................................................... 58

Tabellen
Tabelle 1 – DH-Parameter des SCARA-Roboters ..................................................................... 20
Tabelle 2 – Mögliche Ergebnisse der Funktion savePos ......................................................... 45

Abkürzungen

CAD: Computer-Aided Design
CAN: Controller Area Network
CP: Continuous Path
DH: Denavit-Hartenberg
EGH: Einzelgefäß-Handling
HTS: High Throughput Screening
MRK: Mensch-Roboter-Kollaboration
PTP: Point-to-Point
SCARA: Selective Compliance Assembly Robot Arm
TCP: Tool Center Point
TCP/IP: Transmission Control Protocol/ Internet Protocol

Numerische Werte

\( x, X \): Skalare
\( x \): Vektoren
\( X \): Matrizen

Indizierung

\( x_i \): Eigenschaft \( x \) des Glieds \( i \)
\( X^j_i \): Transformationsmatrix von Glied \( j \) zum Glied \( i \)
\( X_{ij} \): Element der Matrix \( X \)
\( x^j_i \): Vektor von Glied \( j \) zum Glied \( i \)
1 Einleitung

1.1 Problemstellung

In einem globalisierten und sich rasch verändernden Marktumfeld müssen sich Industriebetriebe durch eine hohe Variantenvielfalt, häufige Produktwechsel und kleine Losgrößen anpassen [3]. Der Trend geht weg von der Massenproduktion bis hin zum gegenüberliegenden Extrem, das auch als „Mass Customization“ bezeichnet wird [12].


Um die Akzeptanz der MRK durch den Werker zu erreichen, muss der Aspekt der Sicherheit im Vordergrund stehen und die Entwicklung nach dem individuellen Sicherheitsempfinden des Menschen erfolgen. Das System sollte auf die individuellen menschlichen Eigenschaften des Werkers abgestimmt werden können [17].
Nicht nur im industriellen Bereich, auch in Laboren der Life Science, ein interdisziplinäres Gebiet aus Biologie, Chemie, Lebensmitteltechnologie und Pharmazie, wächst der Wunsch nach flexiblen Roboterplattformen [8]. Die bisherige Automatisierung beschränkt sich auf hochspezialisierte Prozesse wie z.B. das Pipettieren, charakterisiert durch starre Konfigurationen, Beschränkung auf ein Gefäßformat mit multiparallelen Probenhandling und teure Automaten [8, 10]. So werden im High Throughput Screening (HTS), zum Beispiel bei der Medikamentenentwicklung, rund 10.000 Proben täglich bearbeitet [14, 18]. Im Bereich der Forschung und in kleinen bis mittleren Laboren wäre eine Teilautomatisierung mit flexiblen Roboterplattformen im Einzelgefäßen-Handling (EGH) interessant. Der Einsatz verschiedener Gefäßformate mit einem Volumenbereich von 1ml – 500ml kennzeichnet das EGH [8].

Auch im Laborbereich gilt es, nicht alle technisch möglichen, sondern nur sinnvolle Prozesse zu automatisieren und den Menschen miteinzubeziehen. Ein interessanter Bereich hierfür ist die Probenvorbereitung, die unter anderem das Temperieren, Druckbeaufschlagen und Durchmischen von Proben beinhaltet [8]. Werden solche Arbeitsschritte teilautomatisiert, werden Kapazitäten der Laborarbeiter für andere Tätigkeiten frei.


1.2 Ziel und Gliederung der Arbeit


Einleitung

Ziel ist es, durch sicherheitstechnische und ergonomische Erkenntnisse nach Thiemermann [17] auf trennende Schutzeinrichtungen zu verzichten und so eine MRK zu ermöglichen.


Um zu zeigen, dass die Technologie geeignet ist, flexible Aufgaben zu bewältigen, wird eine eigene Teach In Funktion entwickelt. Das Teach In soll intuitiv, individuell und ohne Programmierung erfolgen. Dies wird durch das Handführen des Roboters als kraftgeführte Bewegung [19] realisiert.

1.3 Technologiedemonstrator SCARA-Roboter

1.3.1 Mechanischer Aufbau


Der Großteil der auf dem Markt verfügbaren SCARA-Roboter ist so aufgebaut, dass die Aktuatoren für die Achse 1 und 2 auf den jeweiligen Achsen montiert sind. Dies ermöglicht zwar eine einfache Kraftübertragung von Aktuator auf das Glied, erhöht aber für die Achse 2 auch die Trägheit. Die höhere Trägheit der Achse 2 wirkt sich negativ auf das dynamische Verhalten des Roboters aus. Aus diesem Grund wurde bei dem Technologiedemonstrator der Firma SENSODRIVE GmbH der Aktuator für die Achse 2

Zur Erfüllung einer *Pick and Place* Aufgabe ist der Technologiedemonstrator mit einem elektrischen Parallelgreifer als Endeffektor ausgestattet. Da nur rotationssymmetrische Teile gehandhabt werden sollen, wurde auf eine vierte Achse zur Änderung der Orientierung des TCP verzichtet.


### 1.3.2 Entwicklungsumgebung des Technologiedemonstrators

Die Entwicklung einer Anwendung für den Technologiedemonstrator erfolgt unter MATLAB/ Simulink aus dem Hause *The MathWorks*. Um eine Echtzeitanwendung zu realisieren, wird die Toolbox *Simulink Real-Time* und ein Industrierechner (*xPC-Target*)


1.3.3 Bisherige Funktionen des Technologiedemonstrators

Um den Technologiedemonstrator der Öffentlichkeit vorzustellen, wurden bereits einige Grundfunktionen implementiert, die im Folgenden kurz vorgestellt werden.

Durch eine Drehmomentregelung in den Achsen lassen sich zwei Funktionen zeigen. Zum Beispiel kann eine Nullmomentregelung realisiert werden, die eine gefühlte reibungsfreie Bewegung des Roboters ermöglicht. Dabei wird die Führungsgröße gleich Null gewählt, um sämtlichen Momenten auf der Abtriebsseite des Aktuators auszuweichen. Liegt am Abtrieb des Motors ein Moment an, wird antriebsseitig ein Drehmoment gestellt, welches dazu führt, dass der Motor der Bewegung des abtriebsseitigen Moments folgt. So wird auch das Reibmoment des Getriebes virtuell vom Motor kompensiert, da das abtriebsseitige Moment vereinfacht durch die Summe aus Getriebereibmoment und angelegtem Moment beschrieben werden kann. Um die virtuelle Kompensation der

Eine weitere Anwendung, die auf der Drehmomentregelung beruht, ist die Aufschaltung verschiedener Effekte zum Verhalten des Roboters. Es ist möglich, virtuell zum Beispiel Federn, Dämpfer, Reibung und Trägheit zu simulieren und so das Verhalten des Roboters zu beeinflussen. Mit einem virtuellen Feder-Dämpfer-Verhalten werden zum Beispiel Software-Endanschläge für die Achsen 1 und 2 realisiert.


2 Kinematik eines SCARA-Roboters

2.1 Vorwärtskinematik


Die Beschreibung der Vorwärtskinematik kann für Roboter mit wenigen Gliedern noch auf einfache Art und Weise durch elementargeometrische Zusammenhänge erfolgen. Mit steigender Gelenkzahl \( n \) nimmt die Komplexität jedoch stark zu und es müssen allgemeine Formulierungen gefunden werden.

Betrachtet man einen Roboter mit \( n \) Gelenken und der Voraussetzung, dass jedes Gelenk mit zwei Gliedern verbunden ist, ergeben sich \( n + 1 \) Glieder. Gezählt werden Gelenke von 1 bis \( n \) und Glieder von 0 bis \( n \), wobei das nullte Glied die Basis des Roboters und das \( n \)-te Glied den Endeffektor darstellt. Nach dieser Konvention ist das Gelenk \( i \) mit dem Glied \( i - 1 \) und dem Glied \( i \) so verbunden, dass durch die Betätigung des Gelenks \( i \) das Glied \( i \) bewegt wird. Das Glied 0, die Basis des Roboters, ist ortsfest wird nicht bewegt, wenn eines der Gelenke betätigt wird. Die Stellung von zwei Gliedern \( i \) und \( i - 1 \) zueinander kann durch die homogene Transformationsmatrix

\[
A_i = A_i(q_i)
\]

(1)

der Form

\[
A_i = \begin{bmatrix}
R_{i}^{i-1} & q_{i}^{i-1} \\
0 & 1
\end{bmatrix}
\]

(2)

beschrieben werden. Dabei ist \( R \) eine 3x3 Matrix, die den Rotationsanteil angibt, \( o \) der Vektor zwischen dem Glied \( i - 1 \) und \( i \) und \( q_i \) die verallgemeinerte Gelenkvariable darstellt. Es gilt:

\[
q_i = \begin{cases}
\theta_i, & \text{wenn das Gelenk } i \text{ ein Drehgelenk ist.} \\
d_i, & \text{wenn das Gelenk } i \text{ ein Schubgelenk ist.}
\end{cases}
\]

(3)
Aus der homogenen Transformationsmatrix (1) ergibt sich nun die Möglichkeit, die Lage des Gliedes $j$ in Bezug zu Glied $i$ einer kinematischen Kette zu beschreiben. Dabei gilt die Transformationsmatrix

$$ T_j^i = A_{i+1} \cdot A_{i+2} \ldots A_{j-1} \cdot A_{j} \quad (4) $$

### 2.1.1 Die Denavit-Hartenberg-Konvention

Um die homogene Transformationsmatrix zu ermitteln, hat sich die Methode nach Denavit-Hartenberg etabliert [7, 15, 16], die sogenannte Denavit-Hartenberg-Konvention (DH-Konvention). Nach dieser Konvention lässt sich jede homogene Transformation $A_i$ als Produkt aus zwei Rotationen und zwei Translationen darstellen (Rechenweg und homogene Matrizen im Anhang A):

$$ A_i = Rot_{z, \theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,a_i} \quad (5) $$

Durch die Anwendung der DH-Konvention sind zur Beschreibung der homogenen Transformationsmatrix nur vier Parameter (im Folgenden DH-Parameter) und nicht sechs wie im allgemeinen Fall nötig. Die vier DH-Parameter repräsentieren die geometrischen Eigenschaften des jeweiligen Robotergliedes wie folgt:

- $a_i$: Länge des Gliedes $i$
- $a_i$: Verdrehung des Gliedes $i$
- $d_i$: Hub des Gelenkes $i$
- $\theta_i$: Winkel des Gelenkes $i$

Dabei sind $a_i$ und $a_i$ mechanische Konstanten des jeweiligen Gliedes und $d_i$ und $\theta_i$ die Gelenkvariablen. Abbildung 4 zeigt die geometrische Interpretation der DH-Parameter.

Dass auf zwei Parameter verzichtet werden kann, ermöglicht das Vorgehen zu jedem Glied $i$ ein festes Koordinatensystem $K_i$ zu definieren, welches folgende zwei Annahmen erfüllt:

- DH1: Die Achse $x_i$ steht senkrecht auf der Achse $z_{i-1}$
- DH2: Die Achse $x_i$ schneidet die Achse $z_{i-1}$

Den mathematischen Beweis und eine weitreichendere Erklärung bietet zum Beispiel „Spong et al“ [16]. Im Folgenden werden die wesentlichen Schritte zur Ermittlung der DH-Parameter erläutert [7, 15, 16]:

Joint = Gelenk
Link = Glied

Nummerierung der Glieder

Die Nummerierung der Glieder, von der Basis Glied 0 zum Endeffektor Glied n, wie in Kap. 2.1 beschrieben, ermöglicht ein strukturiertes Vorgehen und die korrekte Benennung der Koordinatensysteme.

Kennzeichnung der Gelenkachsen und Ermittlung der $z$ – Achsen

Die Gelenkachsen sind die Bewegungsebenen der jeweiligen Gelenke. Bei einem Rotationsgelenk erfolgt die Bewegung um die Gelenkachse und bei einem Schubbewegungsebenen des Roboters. Es gilt die Vergabe der $z$ – Achsen so zu wählen, dass die Achse $z_i$ auf der Bewegungsebene des Gliedes $i + 1$ liegt. Also liegt die Achse $z_0$ auf der Bewegungsebene des ersten Gliedes. Die Richtung der $z$ – Achsen ist frei wählbar.
Festlegung des Koordinatensystems $K_0$

Es müssen noch die $x$- und $y$-Achse des Koordinatensystems $K_0$ bestimmt werden. Die Lage der Achsen $x_0$ und $y_0$ sind beliebig unter der Einschränkung, dass sie mit der Achse $z_0$ ein Rechtssystem bilden müssen. Es empfiehlt sich aber die $x$-Achse so zu legen, dass sie die Achse $z_1$ schneidet. Dies führt zu einer vereinfachten Transformationsgleichung.

Festlegung der Koordinatensysteme $K_1$ bis $K_{n-1}$

Zur Festlegung der Koordinatensysteme $K_1$ bis $K_n$ müssen drei Fälle unterschieden werden:

1. Die Achsen $z_{i-1}$ und $z_i$ sind nicht parallel und schneiden sich nicht

   Es wird die kürzeste Verbindung zwischen den Achsen $z_{i-1}$ und $z_i$ (gemeinsame Normale) gesucht. Der Ursprung des Koordinatensystems $K_i$ wird in den Schnittpunkt der Normalen mit der Achse $z_i$ gelegt. Die Achse $x_i$ wird entlang der gemeinsamen Normalen in Richtung der Achse $z_{i+1}$ gelegt, also zum nächsten Glied weisend. Die $y_i$-Achse wird so ergänzt, dass sich ein Rechtssystem ergibt.

2. Die Achsen $z_{i-1}$ und $z_i$ schneiden sich

   Der Ursprung des Koordinatensystems $K_i$ wird in den Schnittpunkt der Achsen $z_{i-1}$ und $z_i$ gelegt. Die Achse $x_i$ wird senkrecht auf die Achsen $z_{i-1}$ und $z_i$ gelegt, so dass sie in Richtung $z_{i+1}$ weist. Die $y_i$-Achse wird so ergänzt, dass sich ein Rechtssystem ergibt.

3. Die Achsen $z_{i-1}$ und $z_i$ verlaufen parallel

   In diesem Fall gibt es unendlich viele gemeinsame Normalen. Es wird entweder die kürzeste Verbindung zu $K_{i-1}$ oder zu $K_{i+1}$ gesucht. Der Schnittpunkt dieser Verbindung mit der Achse $z_i$ ist der Ursprung des Koordinatensystems $K_i$. Die Achse $x_i$ wird wie im Fall 1 entlang der gemeinsamen Normalen in Richtung der Achse $z_{i+1}$ gelegt. Auch die $y_i$-Achse wird wieder so ergänzt, dass sich ein Rechtssystem ergibt.
Festlegung des Koordinatensystems $K_n$ des Endeffektors

Da hier die kinematische Kette zu Ende ist, ergeben sich mehr Freiheiten. Damit die Transformationsregeln gültig bleiben, muss die Achse $z_n$ auf der Bewegungssachse des Endeffektors liegen und die Achse $x_n$ senkrecht auf der Achse $z_{n-1}$. Die Orientierungen beider Achsen sind beliebig. Es aber kann sinnvoll sein, die Orientierung der $z_n - Achse$ so zu wählen, dass sie in Annäherungsrichtung des Endeffektors zeigt. Die $y_n - Achse$ muss wieder so gelegt werden, dass sich ein Rechtssystem ergibt.

Sind alle Koordinatensysteme $K_1$ bis $K_n$ vergeben, können die DH-Parameter direkt abgelesen werden (Abbildung 4).

\[
\begin{align*}
 a_i & : \text{Abstand zwischen } z_{i-1} \text{ und } z_i \text{ entlang der } x_i - Achse \\
 \alpha_i & : \text{Rotation der } z_{i-1} - Achse \text{ um } x_i, \text{ um parallel zu } z_i \text{ zu werden} \\
 d_i & : \text{Abstand zwischen } x_{i-1} \text{ und } x_i \text{ entlang der } z_i - Achse \\
 \theta_i & : \text{Rotation der } x_{i-1} - Achse \text{ um } z_{i-1} \text{, um parallel zu } x_i \text{ zu werden}
\end{align*}
\]

2.1.2 DH-Parameter des SCARA-Roboters

Um die Lage des Endeffektors im kartesischen Raum zu beschreiben, muss die Transformationsmatrix $T_0^4$ nach Gleichung (4) aufgestellt werden. Sie gibt die Lage des Endeffektors bezüglich der starren Basis des Roboters an. Die Basis des Roboters wird auch als Ursprung des Weltkoordinatensystems der Demonstrationsanwendung verwendet. Die $xy$ - Ebene fällt mit der Tischplatte des Aufbaus zusammen (Abbildung 6).

Es muss also nur noch die Transformationsmatrix $T^3_0$ aufgestellt werden. Um die DH-Parameter zu identifizieren, wird ein schematischer Aufbau des SCARA-Roboters skizziert und die Koordinatensysteme nach der Vorgehensweise beschrieben in Kap. 2.1.1 vergeben (Abbildung 5).

Abbildung 5 – Koordinatensysteme am SCARA-Roboter

Die ermittelten DH-Parameter lassen sich nun für jedes Glied angeben. Tabelle 1 zeigt die DH-Parameter für die Glieder 1 bis 3.

<table>
<thead>
<tr>
<th>Glied</th>
<th>$a_i$</th>
<th>$a_i$</th>
<th>$\theta_i$</th>
<th>$d_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$l_1$</td>
<td>$\theta_1^*$</td>
<td>$d_1$</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>$l_2$</td>
<td>$\theta_2^*$</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>$\pi$</td>
<td>0</td>
<td>0</td>
<td>$d_3^*$</td>
</tr>
</tbody>
</table>

Tabelle 1 – DH-Parameter des SCARA-Roboters

Die Parameter $l_1$ und $l_2$ repräsentieren die Längen der Glieder 1 und 2, während $d_1$ die Höhe des Gliedes 1 über der Tischplatte beschreibt. Diese drei Parameter sind geometrische Konstanten mit

\[
l_1 = 0,24972 \, m
\]
\[
l_2 = 0,24310 \, m
\]
\[
d_1 = 0,2 \, m
\]

Die Parameter $\theta_1$, $\theta_2$ und $d_3$ sind die Gelenkvariablen und sind mit einem * markiert.

Nun lässt sich die Gesamttransformationsmatrix nach Gleichung (4) wie folgt darstellen:
Die Berechnung der Vorwärtskinematik erfolgt in zwei Schritten. Erst werden die homogenen Transformationsmatrizen $A_i$ nach Gleichung (5) gebildet und anschließend nach Gleichung (6) aufmultipliziert. Vom Ergebnis, der 4x4 Gesamttransformationsmatrix $T_{30}$, werden nur die Elemente $T_{14}$, $T_{24}$ und $T_{34}$ verwendet, die den Ortsvektor $o_3^0$ und damit die kartesische Position des TCP beschreiben (vgl. Gleichung (7)).

\[
T_{30} = A_1 \cdot A_2 \cdot A_3 \quad (6)
\]

\[
T_{30}^0 = \begin{bmatrix} R_3^0 & o_3^0 \\ 0 & 1 \end{bmatrix} \quad (7)
\]


Abbildung 6 – Koordinatensysteme am Technologiedemonstrator

### 2.2 Inverse Kinematik

Wie in Kap. 2.1 erwähnt, beschäftigt sich die inverse Kinematik damit, die notwendigen Gelenkstellungen zu berechnen, um einen bestimmten Punkt im Raum zu erreichen. Gewissermaßen beschreibt die inverse Kinematik die Umkehrfunktion der Vorwärtskinematik. Hierbei tritt das Problem auf, dass die Umkehrfunktion der Winkelfunktionen...

Abbildung 7 – Mögliche Konfigurationen eines SCARA-Roboters

In Anlehnung eines SCARA-Roboters an den menschliche Arm werden die Konfigurationen auch als ① Ellenbogen links und ② Ellenbogen rechts bezeichnet.

Abbildung 8 – Skizze zur Berechnung der inversen Kinematik

Die mathematischen Ausdrücke der Gelenkwinkel, um einen Punkt P in der Ebene zu erreichen, ergeben sich nach Abbildung 8 zu
\[ \theta_2 = \arctan2\left( \pm \sqrt{1 - D^2}, D \right) \] (8)

mit

\[ D = \cos(\theta_2) = \frac{x_k^2 + y_k^2 - l_1^2 - l_2^2}{2l_1l_2} \] (9)

und

\[ \theta_1 = \varphi - (\varphi - \theta_1) = \arctan2(y_k, x_k) - \arctan2(l_2 \sin(\theta_2), l_1 + l_2 \cos(\theta_2)) \] (10)

Es wird der in der Robotik übliche Arkustangens mit zwei Argumenten [16] zur Berechnung der Winkel verwendet, um den Winkel im richtigen Quadranten zu ermitteln (Anhang B). Im Anhang B ist außerdem die Herleitung der Formeln (8),(9) und (10) ausführlich beschrieben.


Abbildung 9 – Wahl der Konfiguration anhand des Arbeitsraumes

(1) Linke Halbebene, Konfiguration Ellenbogen rechts

(2) Rechte Halbebene, Konfiguration Ellenbogen links

Das Problem des dynamischen Umkonfigurerens beim SCARA-Roboter ist, dass die Anforderung der linearen Interpolation verletzt wird. Mit zwei Rotationsgelenken ist es

Abbildung 10 – Dynamisches Umkonfigurieren am SCARA-Roboter

3 Bahnsteuerung

3.1 Bewegungsarten und Interpolation

Die Kinematik eines Roboters befasst sich mit der statischen Beschreibung seiner Armglieder. Mit dem Bewegungsablauf als solchen befasst sich die Bahnsteuerung. Es wird nicht nur der Zielpunkt beschrieben, sondern auch wie er erreicht wird. Es werden Profile der Beschleunigung, Geschwindigkeit und des Wegs berechnet, um von einem Start- zu einem Zielpunkt zu gelangen.

Grundsätzlich kann man die Bewegung von Robotern in zwei Arten einteilen, die Punkt-zu-Punkt-Steuerung (PTP, point-to-point) und die stetige Bahnsteuerung (CP, continuous path) [16, 20].

3.1.1 PTP-Steuerung


Im einfachsten Fall werden alle Achsen gleichzeitig gestartet und laufen mit maximaler Geschwindigkeit. Dies führt im allgemeinen Fall dazu, dass die Bewegungsdau-

Um diese negativen Effekte zu verringern, wurde die synchrone PTP-Steuerung entwickelt, bei der alle Achsen gleichzeitig starten und stoppen. Dazu wird für jedes Gelenk aus der Fahrstrecke und der maximalen Geschwindigkeit die Bewegungsdauer errechnet. Die Achse mit der größten Bewegungsdauer wird als Leitachse bestimmt und die Geschwindigkeiten aller anderen Achsen so skaliert, dass die Bewegungsdauern der der Leitachse entsprechen. Von einer vollsynchrone PTP-Steuerung spricht man, wenn auch die Beschleunigungsphasen auf die Leitachse abgestimmt werden.

Die PTP-Steuerung sollte vor allem für Anwendungen, bei denen ein großer Arbeitsraum zur Verfügung steht und die genaue Bahn des Roboters von untergeordneter Bedeutung ist, verwendet werden. Für zeitkritische Anwendungen ist die PTP-Steuerung auch sehr gut geeignet, da hier im Vergleich zu anderen Interpolationsarten die Achsen am längsten mit maximaler Geschwindigkeit betrieben werden und so die kürzesten Zeiten erreicht werden. Ein Beispiel wäre eine Bestückungsaufgabe einer Maschine oder eines Förderbandes.

### 3.1.2 CP-Steuerung

3.2 Rampenprofil zur Interpolation

Zur Interpolation eines Bewegungsablaufes zwischen zwei Punkten werden üblicherweise Geschwindigkeits- oder Beschleunigungsprofile als Vorgabe verwendet. Eine reine Interpolation der Gelenkvariable \( q_i \) nach Gleichung (3) kann durch die Aufteilung der Strecke in gleichmäßige Intervalle zu sehr hohen Beschleunigungen und somit zu hohen Kräften führen [20].

**Beispiel:** Aus dem Stillstand einer Linearachse wird im ersten Intervall eine Strecke \( s = 5mm \) in \( 5ms \) durchfahren. Die Beschleunigung beträgt somit:

\[
a = \frac{2s}{t^2} = \frac{2 \cdot 5mm}{(5 \cdot 10^{-3} s)^2} = 400 \frac{m}{s^2}
\]

Durch die Wahl eines Geschwindigkeitsprofils lassen sich diese hohen Beschleunigungen vermeiden. Bei einem rampenförmigen Geschwindigkeitsprofil wird die Geschwindigkeit während einer Beschleunigungszeit \( t_{acc} \) durch eine konstante Beschleunigung \( \ddot{q} \) gleichmäßig auf einen Maximalwert \( q_{max} = \ddot{q} t_{acc} \) gesteigert. Bei Erreichen der maximalen Geschwindigkeit wird die Beschleunigung auf null gesetzt und ab dem Zeitpunkt \( t = t_{dec} \) wird mit einer konstanten Beschleunigung auf \( \ddot{q} = 0 \) abgebremst (Abbildung 14).

![Abbildung 14 – Rampenförmiges Geschwindigkeitsprofil](image)

Um nun zu jedem Zeitpunkt \( t \) eine Aussage über Beschleunigung, Geschwindigkeit und Weg treffen zu können, müssen jeweils geschlossene Ausdrücke gefunden und markante Zeitpunkte definiert werden. Benötigt werden die Beschleunigungs-, Brems- und Gesamtzeit, um die zu interpolierende Strecke zu durchfahren. Für ein symmetrisches rampenförmiges Geschwindigkeitsprofil gilt:
\[ t_{acc} = t_{dec} \quad (11) \]
\[ t_{acc} = \frac{q_{max}}{q_{max}} \quad (12) \]
\[ t_{way} = \frac{q_{way}}{q_{max}} + t_{acc} \quad (13) \]
\[ t_{dec} = t_{way} - t_{acc} \quad (14) \]

Bevor die Bewegungsgleichungen angegeben werden, muss noch ein Sonderfall berücksichtigt werden. Ist die zu interpolierende Strecke \( q_{way} \) sehr klein, kann es sein, dass es keinen Bereich mit konstanter Geschwindigkeit gibt und auch nicht auf die Geschwindigkeit \( \dot{q}_{max} \) beschleunigt werden kann. In diesem Fall ergibt sich ein Geschwindigkeitsprofil wie in Abbildung 15 und es muss die Geschwindigkeit \( \dot{q}_{max} \) auf einen Wert \( \dot{q}_{max, skal} \) korrigiert werden.

\[ \dot{q}_{max, skal} = \sqrt{q_{way} \dot{q}_{max}} \quad (15) \]

Abbildung 15 – Geschwindigkeitsprofil für kleine Strecken

Da nun alle relevanten Zeitpunkte bekannt sind, können die Bewegungsgleichungen aufgestellt werden. Für die Beschleunigung wird keine geschlossene Form, sondern eine abschnittsweise Definition angegeben.

\[ \ddot{q}(t) = \begin{cases} 
\dot{q}_{max} & \text{für } t < t_{acc} \\
0 & \text{für } t_{acc} \leq t \leq t_{dec} \\
-\dot{q}_{max} & \text{für } t > t_{dec}
\end{cases} \quad (16) \]

Die Ausdrücke für die Geschwindigkeit und den Weg ergeben sich durch Integration über die jeweiligen Zeitabschnitte.
Für $t < t_{acc}$

\[ \dot{q}(t) = \ddot{q}_{\text{max}} t \]
\[ q(t) = \frac{1}{2} \ddot{q}_{\text{max}} t^2 \]  

(17)

(18)

Für $t_{acc} \leq t \leq t_{dec}$

\[ \dot{q}(t) = \ddot{q}_{\text{max}} \]
\[ q(t) = \ddot{q}_{\text{max}} t - \frac{\ddot{q}_{\text{max}}}{2} t^2 \]  

(19)

(20)

Und für $t > t_{dec}$

\[ \dot{q}(t) = \ddot{q}_{\text{max}} - \ddot{q}_{\text{max}} (t - t_{dec}) \]
\[ q(t) = \ddot{q}_{\text{max}} (t - t_{dec}) - \frac{1}{2} \ddot{q}_{\text{max}} (t_{\text{way}} - t)^2 \]  

(21)

(22)

Die zeitlichen Verläufe für die Beschleunigung, Geschwindigkeit und den Weg bei einer Interpolation mit rampenförmigem Geschwindigkeitsprofil sind in Abbildung 16 dargestellt.
Ein großer Vorteil dieser Interpolationsmethode ist es, dass durch die konstante Beschleunigung die Maximalgeschwindigkeit $\dot{q}_{max}$ sehr schnell erreicht wird und die Bewegung am längsten mit Maximalgeschwindigkeit abläuft. Dadurch werden, verglichen mit anderen Interpolationsmethoden, die kürzesten Zeiten erreicht.

Das sprungförmige Auftreten der Beschleunigung, ausgelöst durch das rampenförmige Geschwindigkeitsprofil, führt aber auch zu Problemen. Betrachtet man die zeitliche Änderung der Beschleunigung, ergeben sich mehrere Dirac-Impulse (Abbildung 17). Um Aussagen über dieses Phänomen treffen zu können, wird eine neue Größe $\ddot{q}$ eingeführt und als Ruck (engl. jerk) bezeichnet [16, 20]. Dass der Ruck bei der Interpolation mit rampenförmigem Geschwindigkeitsprofil aus mehreren Dirac-Impulsen besteht,
kann negative Auswirkungen auf das Robotersystem haben. Die mechanische Struktur wird stark durch die breitbandige Frequenzanregung belastet. Speziell für eine MRK gilt aber das Problem, dass das Anfahren und Abbremsen des Roboters durch die sprungför-
migen Änderungen der Beschleunigung sehr abrupt ist. Um eine Akzeptanz des Robo-
ters durch den Menschen zu erreichen, wäre es besser, wenn die Beschleunigung mit einer Steigung gleich Null ansteigt und durch sanfte Übergänge geprägt ist.

Abbildung 17 – Beschleunigung und Ruck bei Rampeninterpolation
3.3 Sinoidenprofil zur Interpolation

Eine Möglichkeit ein Beschleunigungsprofil zu generieren, welches stetig ansteigt und so auch zu einem stetigen Ruck führt, ist ein, sinoides Profil folgender Form vorzugeben [16, 20]:

\[ \ddot{q}(t) = \ddot{q}_{\text{max}} \sin^2 \left( \pi \frac{t}{t_{\text{acc}}} \right) \]  

(23)

Die Änderung der Beschleunigung ist stetig und weich, die Bewegung wird flüssig und der Ruck begrenzt (Abbildung 18).

Abbildung 18 – Beschleunigung und Ruck bei Sinoideninterpolation
Zur Berechnung der Bewegungsgrößen wird wieder der Sonderfall einer kurzen Strecke berücksichtigt. Wird die maximale Geschwindigkeit \( \dot{q}_{\text{max}} \) nicht erreicht, wird auf eine Geschwindigkeit \( \dot{q}_{\text{max, skal}} \) nach folgender Gleichung skaliert.

\[
\dot{q}_{\text{max, skal}} = \sqrt{\frac{\dot{q}_{\text{way}} \dot{q}_{\text{max}}}{2}}
\]  

(24)

Zur Berechnung der relevanten Zeiten gilt ähnlich zu Kap. 3.2.

\[
t_{\text{acc}} = t_{\text{dec}}
\]

(25)

\[
t_{\text{acc}} = \frac{2\dot{q}_{\text{max}}}{\dot{q}_{\text{max}}}
\]

(26)

\[
t_{\text{way}} = \dot{q}_{\text{way}} + t_{\text{acc}}
\]

(27)

\[
t_{\text{dec}} = t_{\text{way}} - t_{\text{acc}}
\]

(28)

Die Bewegungsgleichungen werden analog zu Kap. 3.2 durch Integration über die Zeitabschnitte gebildet und ergeben sich zu.

Für \( t < t_{\text{acc}} \)

\[
\ddot{q}(t) = \dot{q}_{\text{max}} \sin^2 \left( \pi \frac{t}{t_{\text{acc}}} \right)
\]

\[
\dot{q}(t) = \dot{q}_{\text{max}} \left( \frac{t}{2} - \frac{t_{\text{acc}}}{4\pi} \sin \left( 2\pi \frac{t}{t_{\text{acc}}} \right) \right)
\]

\[
q(t) = \dot{q}_{\text{max}} \left( \frac{t^2}{4} + \frac{t_{\text{acc}}^2}{8\pi^2} \cos \left( 2\pi \frac{t}{t_{\text{acc}}} - 1 \right) \right)
\]

(29) (30) (31)

Für \( t_{\text{acc}} \leq t \leq t_{\text{dec}} \)

\[
\ddot{q}(t) = 0
\]

\[
\dot{q}(t) = \dot{q}_{\text{max}}
\]

\[
q(t) = \dot{q}_{\text{max}} \left( t - \frac{t_{\text{acc}}}{2} \right)
\]

(32) (33) (34)

Und für \( t > t_{\text{dec}} \)

\[
\ddot{q}(t) = -\dot{q}_{\text{max}} \sin^2 \left( \pi \frac{t - t_{\text{dec}}}{t_{\text{acc}}} \right)
\]

\[
\dot{q}(t) = \dot{q}_{\text{max}} - \dot{q}_{\text{max}} \left( \frac{t - t_{\text{dec}}}{2} - \frac{t_{\text{acc}}}{4\pi} \sin \left( 2\pi \frac{t - t_{\text{dec}}}{t_{\text{acc}}} \right) \right)
\]

\[
q(t) = \dot{q}_{\text{max}} (t - t_{\text{acc}}) - \dot{q}_{\text{max}} \left( \frac{(t - t_{\text{dec}})^2}{4} + \frac{t_{\text{acc}}^2}{8\pi^2} \cos \left( 2\pi \frac{t - t_{\text{dec}}}{t_{\text{acc}}} - 1 \right) - \frac{t_{\text{acc}}^2}{4} \right)
\]

(35) (36) (37)
Die Ergebnisse einer Interpolation mit sinoidem Beschleunigungsprofil ist in Abbildung 19 dargestellt. Erkennbar ist ein weicher und gleichmäßiger Verlauf der Bewegungsgrößen, was zu einem sanften Fahrverhalten des Roboters führt. Deshalb ist die Interpolation mit sinoidem Beschleunigungsprofil besonders für eine MRK geeignet und wird für die Handhabungsaufgabe verwendet. Um dem individuellen Sicherheitsempfinden des Menschen gerecht zu werden, ist auch der Parameter $\dot{q}_{\text{max}}$ zur Laufzeit änderbar. So kann sich der Werker die Geschwindigkeit des Roboters auf seinen Idealwert einstellen. Damit keine Unter- oder Überforderung auftritt, ist dies besonders wichtig [9].

Abbildung 19 – Bewegungsgrößen bei Interpolation mit Sinoidenprofil
3.4 Linearinterpolation

Für eine MRK ist es von entscheidender Bedeutung, dass die Bahn des Roboters so verläuft, wie ein Mensch es erwartet. Das bedeutet, dass die Bahn zwischen zwei Punkten eine Gerade beschreibt.

Abbildung 20 – Programmausschnitt Berechnung von Zielwinkel für die Interpolation
Für den Trajektoriengenerator aus Abbildung 20 wird eine Schrittweite (delta_way, Inputport 4) vorgegeben. Diese Schrittweite ist die Strecke, zwischen der die PTP-Interpolation stattfindet. Für die Handhabungsaufgabe lieferte ein Wert von delta_way = 5mm gute Ergebnisse. Aus der Schrittweite, den aktuellen und den Zielkoordinaten (target_coordinate, Inputport 3) werden nach dem ersten und zweiten Strahlensatz Weginkremente dx und dy berechnet (siehe Anhang C).

Sobald sich die Zielkoordinaten ändern oder der TCP an den Zwischenpunkt annähert (vgl. Abbildung 23), werden die Weginkremente auf die aktuellen kartesischen Koordinaten addiert. Die so berechneten Koordinaten beschreiben den neuen Zwischenpunkt, der mit der PTP-Interpolation anfahren wird. Abbildung 21 zeigt den Programmablauf für die Berechnung der Zwischenpunkte als Flussdiagram.

Abbildung 21 – Flussdiagramm Berechnung von Zwischenpunkten
3.5 Durchfahren von Zwischenpunkten ohne Anhalten

Würde zwischen den in Kap. 3.4 berechneten Zwischenpunkten einfach die PTP-Interpolation angewendet werden, ergäbe sich ein Geschwindigkeitsprofil wie in Abbildung 22 a) dargestellt. Zwischen zwei Punkten gäbe es immer eine Beschleunigungs- und Bremsphase und an jedem berechneten Zwischenpunkt wäre die Geschwindigkeit \( \dot{q} = 0 \). Dieses Verhalten ist unerwünscht, da es keine gleichmäßige Bewegung erzeugt.

Um zu vermeiden, dass vor dem Erreichen des Zielpunktes abgebremst wird, kommt ein Verfahren zum Einsatz, das *Positionsüberschleifen* genannt wird [20]. Dem Interpolator werden neue Zielkoordinaten übergeben, bevor die aktuellen Zielkoordinaten erreicht sind. Zusätzlich wird die aktuelle Geschwindigkeit \( \dot{q} \) für die Folgeinterpolation berücksichtigt. Das Ergebnis ist eine flüssige Bewegung zwischen Start- und Zielpunkt, ohne dass an Zwischenpunkten angehalten wird (Abbildung 22 b)).

![Abbildung 22 – Geschwindigkeitsprofil bei Linearinterpolation](image)

a) Ohne *Positionsüberschleifen*

b) Mit *Positionsüberschleifen*

Das *Positionsüberschleifen* entspricht dem Annähern des TCP an den nächsten Zwischenpunkt, das in Kap. 3.4 beschrieben wird. Wurde die Hälfte der Wegstrecke zum nächsten Zwischenpunkt zurückgelegt, wird ein neuer Zwischenpunkt berechnet. Der Programmausschnitt in Abbildung 23 zeigt alle auslösenden Ereignisse zur Berechnung des nächsten Zwischenpunktes.
Abbildung 23 – Programmausschnitt auslösende Ereignisse zur Zwischenpunktberechnung
4 Entwicklung einer flexiblen Teach In Funktion

4.1 Notwendigkeit einer flexiblen Teach In Funktion

Die Programmierung eines Roboters kann grundsätzlich auf zwei Arten erfolgen, durch die Online- oder die Offline-Programmierung [13, 20]. Bei der Offline-Programmierung wird zuerst ohne den Roboter gearbeitet und dann das fertige Programm auf die Robotersteuerung kopiert. Für eine erfolgreiche Offline-Programmierung ist es notwendig, genaue geometrische Daten über die Aufgabe und alle beteiligten Komponenten zu besitzen. Für eine MRK im Laborbereich, die flexibel sein und ohne aufwendige Gerätschaften auskommen soll, ist das ungeeignet. Hingegen werden bei der Online-Programmierung der eingeschaltete Roboter und seine Steuerung als Hilfsmittel verwendet. Es werden drei Arten der Online-Programmierung unterschieden:

- Teach-In-Programmierung
- Master-Slave-Programmierung
- Folgeprogrammierung

Entwicklung einer flexiblen Teach In Funktion


Für die Handhabungsaufgabe soll der SCARA-Roboter ohne Programmierkenntnisse auf möglichst einfache Art und Weise programmiert werden. Da es hierfür keine verfügbare Lösung gibt, wird eine flexible Teach In Funktion entwickelt, die auf der Folgeprogrammierung durch eine kraftgeführte Bewegung beruht. Es wird aber nicht die gezeigte Bahn, sondern einzelne Punkte abspeichert und für die Ablaufsteuerung aufbereitet. Außerdem sollen nicht alle, sondern möglichst wenig Punkte geteacht (aus dem Englischen „to teach“, eingedeutschter Begriff in der Robotik) werden.


\[
\tau = J \ddot{\phi} + d \dot{\phi}
\]

Über die Parameter \( J \) und \( d \) lässt sich das Wunschverhalten individuell auf den Bediener anpassen. Um die Leistungsfähigkeit der Drehmomentregelung zu unterstreichen, bietet es sich allerdings an, die beiden Parameter \( J = d = 0 \) zu setzen.

4.2 Positionen speichern

4.2.1 Vorüberlegungen

Um das Abspeichern der geteachten Punkte ohne zusätzliches Material, wie zum Beispiel ein Anzeige- oder Eingabebereich, durchführen zu können, muss eine feste Reihenfolge für den Ablauf des Teach In vorgegeben werden. Für die Echtzeitfähigkeit der Funktion
müssen außerdem alle Variablen zum Zeitpunkt des Kompilierens vollständig definiert sein. Es müssen also ein Dummywert zur Initialisierung und die Dimension der Variable vorgegeben werden.

Für die in Kap. 1.2 beschriebene Funktion wird ein Modell einer Zentrifuge mit \( N_{cent} = 12 \) Probenpositionen verwendet. Bei einer Anzahl von \( N_{rack} = 4 \) Probenpositionen pro Rack können also Proben von maximal 3 Racks entnommen, durchmischt und damit die Zentrifuge bestückt werden. Um eine minimale Anzahl an Teachpunkten zu erreichen, wird pro Probenträger (Zentrifuge, Racks) nur die Anzahl an Punkten geteacht, die ausreichend ist, um alle anderen Punkte zu berechnen. Bei der Zentrifuge sind das \( N_{teach,cent} = 3 \) und bei den Racks \( N_{teach,rack} = 2 \). Außerdem muss noch eine Position \( Pos_{shake} \), an der das Durchmischen der Proben stattfindet, definiert werden. Es ergibt sich also eine maximale Anzahl von Teachpunkten nach:

\[
N_{teach,max} = N_{teach,cent} + 3N_{teach,rack} + Pos_{shake} = 10
\]  

(39)

Die geteachten Punkte werden komponentenweise gespeichert. Das heißt, es werden zwei Arrays, eines für die x-Koordinaten und eines für die y-Koordinaten, mit je 10 Elementen benötigt. Initialisiert werden sie mit den Dummywerten Dummy = -1. Dieser Wert liegt außerhalb des Arbeitsraumes und kann so zweifelsfrei als Dummy identifiziert werden.

Um aus den Elementen der Arrays in den folgenden Funktionen die richtigen Werte zu entnehmen, wird eine feste Reihenfolge vorgegeben, wie das Teach In ablaufen soll. Zuerst werden die Diagonalen der Racks geteacht, danach die Position des Durchmischiens und zuletzt drei Positionen auf der Zentrifuge (Abbildung 24). Es können ein, zwei oder drei Racks geteacht werden, um zu zeigen, dass die Teach In Funktion flexibel genutzt werden kann. Das Abspeichern der Punkte wird automatisch beendet, wenn die maximale Anzahl von \( N_{teach,max} = 10 \) Punkten geteacht wurde. Wenn weniger als drei Racks verwendet werden, kann das Teach In auch vorzeitig beendet werden. Dazu wird der letzte Punkt zweimal geteacht. Das Vorgehen, um die Proben-positionen exakt zu treffen, wird in Kap. 5.1.3 erläutert.
4.2.2 Umsetzung


Es wird eine Funktion mit drei Eingängen und zwei Ausgängen geschrieben (Abbildung 25). Eingänge der Funktion sind die aktuellen kartesischen Koordinaten als 2x1 Vektor, der Speicherbefehl und ein Resetbefehl. Die gespeicherten Werte für die x- und y-Koordinaten werden je als 10x1 Vektor ausgegeben.
Abbildung 25 – Ein- und Ausgänge der Funktion savePos

In der Funktion savePos (Abbildung 25) sollen die aktuellen kartesischen Koordinaten (act_Pos_cartesian, Inputport 1) immer dann in einem Work vector abgespeichert werden, wenn am Eingang save_Pos (Inputport 2) eine steigende Flanke anliegt. Die Ausgänge (Outputport 1 und 2) geben solange einen mit dem Wert Dummy = −1 beschriebenen Vektor aus, bis das Teach In beendet ist, erst dann werden die im Work vector gespeicherten Werte ausgegeben. Soll ein neuer Satz Punkte abgespeichert werden oder wurde ein Fehler beim Teach In gemacht, können die Ausgänge und Work vectors über den Eingang reset (Inputport 3) auf den Dummywert zurückgesetzt werden.

Die Programmstruktur der Funktion savePos ist in Abbildung 26 als Flussdiagramm dargestellt. Mögliche Ergebnisse der Funktion sind in Tabelle 2 aufgelistet.

<table>
<thead>
<tr>
<th>Initialwert</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ein Rack</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Zwei Racks</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Drei Racks</th>
<th>x</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabelle 2 – Mögliche Ergebnisse der Funktion savePos
4.3 Berechnung aller Positionen

4.3.1 Rackpositionen

Wie in Kap. 4.2.1 beschrieben, sollen so wenig Punkte wie möglich geteacht werden. Um bei den quadratischen Racks mit vier Probenpositionen alle Punkte definieren zu können, reicht es, zwei Punkte $T_1$ und $T_2$ auf einer Diagonalen zu teachen. Die beiden übrigen Punkte $P_1$ und $P_2$ können dann nach Abbildung 27 berechnet werden. Es gilt:

$$P_1 = M + \lambda u$$
$$P_2 = M - \lambda u$$

Die Berechnungen von $\lambda$ und $u$ sind in Anhang D dargestellt.
4.3.2 Zentrifugenpositionen

Um die kreisförmig auf dem Zentrifugenmodell angeordneten Positionen zu berechnen, genügt es drei beliebige Positionen abzuspeichern. Aus diesen drei Punkten können alle übrigen Positionen auf dem Zentrifugenmodell berechnet werden.


$$R(\alpha) = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \tag{41}$$

um den Winkel $\alpha$ rotiert. Für $\alpha$ gilt

$$\alpha_i = i \cdot \frac{2\pi}{N_{cent}}, \text{mit } i = [1; 11] \tag{42}$$

Da die Punkte um den Ursprung des Koordinatensystems rotiert wurden, müssen sie nun noch um den Ortsvektor des Mittelpunkts $\mathbf{m}$ des Umkreises verschoben werden. Damit gilt für jeden Punkt $P_i$ auf dem Zentrifugenmodell

$$P_i = \mathbf{m} + R(\alpha_i)\mathbf{pm} \tag{43}$$

Die berechneten Ergebnisse der Rackpositionen aus Gleichung (40) und der Zentrifugenpositionen aus Gleichung (43) sind in Abbildung 28 dargestellt.
4.4 Positionen sortieren und bereitstellen

Das Ergebnis aus der Funktion savePos muss für die Berechnung der übrigen Positionen und zur Weiterverwendung in der Ablaufsteuerung noch aufbereitet werden. Die Ausgangsvektoren der Funktion savePos haben durch die vorgegebene Reihenfolge eine Form, wie in Abbildung 29 am Beispiel von zwei Racks gezeigt.

Zur Berechnung der übrigen Punkte, müssen die entsprechenden Elemente der Vektoren entnommen werden. Zur Identifizierung der benötigten Elemente wird die Anzahl der geteichten Punkte verwendet. Dazu wird zuerst überprüft, ob alle 10 möglichen Punkte geteacht wurden. Wenn weniger als 10 Punkte geteacht wurden, ist das letzte Element des Vektors gleich Dummy = −1. Ist dies der Fall, wird das erste Element der Vektoren gesucht, welches dem Wert Dummy = −1 entspricht. Von diesem oder dem letzten Element, im Falle von 10 geteachten Punkten, werden die letzten drei Elemente zur


![Abbildung 30 – Überlappung zweier Probenfläschchen in Greifer und Rack](image)

Um zu verhindern, dass es zu Kollisionen zwischen zwei Probenfläschchen kommt, muss bei der Bahnplanung sichergestellt werden, dass sich zwischen der Position, an
Entwicklung einer flexiblen Teach In Funktion

Abbildung 31 – Sortierschema zur Kollisionsvermeidung

Mit der nun festgelegten Reihenfolge der Punkte muss noch ein Vektor aus Zielkoordinaten für die Ablaufsteuerung erzeugt werden. Es sollen abwechselnd Rack- und Zentrifugenkoordinaten und als letztes Element die Position des Durchmischens übergeben werden (Abbildung 32). Werden nicht alle Racks verwendet wird der Wert Dummy = −1 übergeben.

<table>
<thead>
<tr>
<th>$X_{Rack1}$</th>
<th>$X_{cent1}$</th>
<th>$X_{Rack2}$</th>
<th>$X_{cent2}$</th>
<th>$X_{Rack3}$</th>
<th>$X_{cent3}$</th>
<th>$\ldots$</th>
<th>$X_{Rack12}$</th>
<th>$X_{cent12}$</th>
<th>$X_{shake}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_{Rack1}$</td>
<td>$Y_{cent1}$</td>
<td>$Y_{Rack2}$</td>
<td>$Y_{cent2}$</td>
<td>$Y_{Rack3}$</td>
<td>$Y_{cent3}$</td>
<td>$\ldots$</td>
<td>$Y_{Rack12}$</td>
<td>$Y_{cent12}$</td>
<td>$Y_{shake}$</td>
</tr>
</tbody>
</table>

Abbildung 32 – Zielvektoren für die Ablaufsteuerung
5 Programmablauf

5.1 Steuerung des Greifers

5.1.1 Referenzfahrt


![Flussdiagramm systematische Funktion des Greifers](image)

Eine Referenzfahrt ist nötig, da der Linearstellantrieb nur über ein inkrementelles Wegmesssystem verfügt. Für jeden Systemneustart und Fehler muss die $z$-Position des Greifers referenziert werden. Für die Referenzfahrt steht eine Gabellichtschranke an der Führungsschiene des Linearstellantriebes und eine Referenzmarke am elektrischen

Abbildung 34 – Schematischer Aufbau Greifer

5.1.2 Modus für das Greifen und Absetzen

Ist die Referenzfahrt durchgeführt, wechselt die Greifersteuerung in einen Default State. Nun kann die Ablaufsteuerung auf den Greifer zugreifen. Für eine möglichst einfache Gestaltung der Ablaufsteuerung wird eine Funktion entwickelt, die das Greifen und Absetzen übernimmt. Diese Funktion führt je nach übergebenem Parameter das Greifen
oder das Absetzen der Probe komplett aus. Das Funktionsschema ist in Abbildung 36 dargestellt.

![Flussdiagramm Funktionsablauf Greifen/Absetzen](attachment:diagram.png)

**Abbildung 36 – Flussdiagramm Funktionsablauf Greifen/Absetzen**

### 5.1.3 Modus für das Teach In
Um für die *Teach In* Funktion aus Kap. 4 die genaue Position eines Probenfläschchens bereitzustellen, wird der Greifer verwendet. Der Bediener soll mit dem Greifer die Position eines Probenfläschchens anfahren und es greifen. Wenn die Probe gegriffen ist, wird der Speicherbefehl gegeben. Um diese manuelle Operation möglichst komfortabel zu gestalten, wird ein Taster als Eingabegerät am Greifer montiert.

Mit diesem Taster soll der Bediener alle Funktionen für das *Teach In* durchführen können. Der Greifermodus für das *Teach In* folgt dem Schema aus Abbildung 37.
Nach dem Funktionsschema aus Abbildung 37 muss der Bediener also den Taster solange gedrückt halten, bis die Position der Probe erreicht ist. Dann wird die Probe durch einen erneuten Tastendruck gegriffen und anschließend die Position gespeichert. Wird in der Abwärtsbewegung der Taster losgelassen, fährt der Greifer auf den maximalen Hub.


Die gesamte Greifersteuerung, aufgeteilt in die Referenzfahrt und die Greifermodi, ist in Abbildung 38 dargestellt.
Abbildung 38 – Programmausschnitt Greifer
5.2 Ablaufsteuerung

Um die Gesamtfunktion der Handhabungsaufgabe (Kap. 1.2) zu erfüllen, müssen folgende Teilfunktionen ausgeführt werden:

- Greifen der Probe
- Durchmischen der Probe
- Absetzen der Probe

Es wird eine Ablaufsteuerung entwickelt, die alle Befehle zum richtigen Zeitpunkt gibt, um diese Teilfunktionen zu erfüllen. Diese Befehle sind:

- die Ausgabe von Zielkoordinaten für die Bahnsteuerung
- der Greiferbefehl zum Greifen oder Absetzen
- die Wahl des Roboterzustandes

Der Ablaufsteuerung (Abbildung 39) werden dabei die aktuelle kartesische Position (Inputport 1 und 2), die gespeicherten Punkte aus der Teach In Funktion (Inputport 3 und 4) und der Start- (Inputport 5) und Resetbefehl (Inputport 6) für den Funktionsablauf übergeben, außerdem der Zustand des Greifers (Inputport 7 und 8) und der Zustand der Achsen 1 und 2 (Inputport 9 und 10).

Abbildung 39 – Ein- und Ausgänge der Ablaufsteuerung

Die Koordinaten der gespeicherten Punkte aus der Teach In Funktion sind durch die Sortierung aus Abbildung 32 schon so vorbereitet, dass sie in der Ablaufsteuerung der

Abbildung 40 zeigt den Programmausschnitt der Ablaufsteuerung der Handhabungsaufgabe. Der Default State ist der Zustand Home_position, in dem der Zähler \( k \) der Ablaufsteuerung auf den Wert \( k = 1 \) gesetzt und eine Startposition übergeben wird. Diese Position wird zu Beginn und am Ende der Handhabungsaufgabe anfahren und durch eine steife Positionsregelung fixiert. Mit einem Startsignal, das durch Tastendruck auf den Taster aus Kap. 5.1.3 erzeugt wird, beginnt die Ablaufsteuerung.
Den Zielkoordinaten wird im Zustand \textit{goto\_target\_position} das \textit{k-te} Element der Punkte aus der \textit{Teach In} Funktion übergeben. Dieser Zielpunkt wird von der Bahnsteuerung angefahren. Zur Bestätigung, dass der Zielpunkt sicher erreicht wurde, muss sich der SCARA-Roboter für eine gewisse Zeitspanne innerhalb eines Kreises mit Radius \(r = 0.5\, \text{mm}\) um den Zielpunkt befinden. Ist dies der Fall, wird der SCARA-Roboter wieder durch die steife Positionsregelung fixiert. Durch die Fixierung wird sichergestellt, dass es keine Positionsabweichungen während der Greifoperation gibt.

Je nach Wert des Zählers \(k\) wird die Greiferoperation – Greifen oder Absetzen – ausgeführt. Für den Fall, dass der Wert von \(k\) ungerade ist, wird eine Probe gegriffen und die Position des Durchmischens angefahren. Das Durchmischen der Probe erfolgt durch eine PTP-interpolierte Bewegung zwischen zwei Punkten mit erhöhter Beschleunigung. Für gerade Werte von \(k\) wird die Probe in der Zentrifuge abgesetzt. Nach dem Durchmischen oder Absetzen der Probe kann die nächste Position angefahren werden. Dazu wird der Wert von \(k\) inkrementiert und die Ablaufsteuerung in den Zustand \textit{goto\_target\_position} gewechselt.

Wurde die letzte Probe in der Zentrifuge platziert, wird wieder die Startposition angefahren und der SCARA-Roboter fixiert. Die letzte Probe wird durch einen Wert \(k = 24\) oder \(k + 1 = -1\) erkannt. Für \(k = 24\) wurde das Zentrifugenmodell voll bestückt, während \(k + 1 = -1\) bedeutet, dass nicht alle Racks verwendet wurden und nun Dummywerte folgen.

### 5.3 Kollisionserkennung

Um die Sicherheit des Menschen in der MRK zu gewährleisten, ist eine Kollisionserken-
nung von entscheidender Bedeutung. Für den Technologiedemonstrator wird eine Kol-
losionserkennung mit verschiedenen Reaktionen entwickelt. Wird eine Kollision durch
ein unzulässig hohes Drehmoment detektiert, soll eine Reaktion darauf erfolgen. Mögli-
che Reaktionen auf eine Kollision sind:

- Sofortiges Anhalten und sicheres Halten der Position
- Wechsel in den freien Modus, um nachgiebig zu sein
- Sofortiges Anhalten und Starten einer Ausweichfahrt

Für eine MRK ist das Durchführen einer Ausweichfahrt unter Umständen nicht ge-
eignet. Da von den Drehmomentsensoren zwar eine Kollision erkannt wird, aber nicht

Um im Falle einer Kollision das Sicherheitsrisiko so gering wie möglich zu halten, müssen Grenzwerte für die Stoß- und Quetschkräfte eingehalten werden [17]. Da es sich bei dem Technologiedemonstrator allerdings nicht um ein kommerzielles Produkt handelt, sondern um einen Prototyp, mit dem das Leistungsvermögen der Produkte gezeigt werden soll, werden die Kollisionsmomente nicht auf festgelegte Grenzwerte ausgelegt. Die Kollisionsmomente werden so ausgelegt, dass die gefühlte Kollisionskraft so gering wie möglich ist.

Die bestehende Kollisionserkennung des Technologiedemonstrators (Kap. 1.3.3) erkennt eine Kollision, wenn ein definiertes Moment am Abtrieb des Motors überschritten wird. Da für die Handhabungsaufgabe eine Linearinterpolation zum Einsatz kommt, reicht es nicht mehr, nur ein maximales Moment als Kollisionsgrund zu verwenden. Bei der Linearinterpolation treten durch die kurzen Interpolationsstrecken hohe Beschleunigungen in kurzen Zeiträumen auf. Folglich müssen die Aktuatoren der Achsen auch hohe Momente liefern. Es muss also entweder ein hohes Kollisionsmoment definiert oder eine alternative Lösung gefunden werden. Um ein Kollisionsmoment, kleiner als das Spitzenmoment der Beschleunigung detektieren zu können, wird eine Momentüberwachung implementiert. Diese Momentüberwachung prüft, ob ein Moment größer als ein definierter Grenzwert über einen bestimmten Zeitraum hinweg anliegt. So werden die Spitzenmomente ignoriert und gefühlt ein sehr geringes Kollisionsmoment erreicht.

Kollision die Nullmomentregelung (vgl. Kap. 1.3.3) aktiviert und so ein nachgiebiges Verhalten des Roboters erzeugt. Dadurch kann der Bediener direkt in den Programmablauf eingreifen und den Roboter durch eine handgeführte Bewegung beeinflussen. Soll der Programmablauf fortgesetzt werden, muss der Taster aus Kap. 5.1.3 gedrückt werden.

6 Zusammenfassung


Die Demonstrationsanwendung, die Gegenstand dieser Arbeit war, zeigt bei Tests durch Kollegen gute Ergebnisse. Es ist eine geschlossene Demonstration möglich.
7 Ausblick

Die Anforderungen im Rahmen der Aufgabenstellung sind soweit erfüllt, dass eine geschlossene Demonstrationsanwendung möglich ist. Um eine bessere Bedienbarkeit und Präsentation des Technologiedemonstrators zu ermöglichen, ergeben sich einige Ansatzpunkte.

Durch eine graphische Benutzeroberfläche wäre ein bequemer Wechsel zwischen den Modi des Roboters möglich. Vor allem für die Teach In Funktion ergäbe sich die Möglichkeit den gesamten Vorgang durch Anweisungen anzuleiten. Auch könnten die gespeicherten Punkte sofort angezeigt werden, dadurch ließe sich der Vorgang visuell überwachen und Fehler direkt erkennen. Für die Geschwindigkeitsskalierung der Linearinterpolation könnte zusätzlich ein Schieberegler dargestellt werden, mit dem der Skalierungsfaktor gewählt werden kann.


Teile der Arbeit können auch für Folgeprojekte verwendet werden. Im Besonderen kann die Teach In Funktion in Kombination mit der Drehmoment-Sensorik für ein Master-Slave System eingesetzt werden. Während das Teach In am Master durchgeführt wird, ist eine Rückkopplung der Drehmomente vom Slave möglich. So kann eine mögliche Kollision des Slaves vom Bediener wahrgenommen werden.
Anhang

Anhang A

Im Gegensatz zur physikalischen Beschreibung von Translationen und Rotationen im Raum ermöglicht die Beschreibung der Bewegungen mit homogenen Matrizen beides, sowohl die Änderung der Orientierung als auch des Ortes in einer Matrix zu beschreiben [15, 16, 20]. Dazu werden die physikalischen Vektoren aus dem \( \mathbb{R}^3 \) in den \( \mathbb{R}^4 \) übertragen. So wird ein Punkt \( P \) in homogenen Koordinaten wie folgt dargestellt:

\[
\begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix} \Rightarrow \begin{pmatrix}
  x \\
  y \\
  z \\
  1
\end{pmatrix}
\]

\( \mathbb{R}^3 \rightarrow \mathbb{R}^4 \)

Rotationen und Translationen werden durch 4x4 Matrizen beschrieben, bei denen die vierte Zeile immer \((0 \ 0 \ 0 \ 1)\) heißt. Transformationen werden wie folgt dargestellt:

Translation um den Vektor \((x\ y\ z)^T\)

\[
\text{Trans}(x, y, z) = \begin{pmatrix}
  1 & 0 & 0 & x \\
  0 & 1 & 0 & y \\
  0 & 0 & 1 & z \\
  0 & 0 & 0 & 1
\end{pmatrix}
\]

Rotation um die x-Achse um den Winkel \(\theta\)

\[
\text{Rot}(x, \theta) = \begin{pmatrix}
  1 & 0 & 0 & 0 \\
  0 & \cos(\theta) & -\sin(\theta) & 0 \\
  0 & \sin(\theta) & \cos(\theta) & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
\]

Rotation um die y-Achse um den Winkel \(\theta\)

\[
\text{Rot}(y, \theta) = \begin{pmatrix}
  \cos(\theta) & 0 & \sin(\theta) & 0 \\
  0 & 1 & 0 & 0 \\
  -\sin(\theta) & 0 & \cos(\theta) & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
\]

Rotation um die z-Achse um den Winkel \(\theta\)

\[
\text{Rot}(z, \theta) = \begin{pmatrix}
  \cos(\theta) & -\sin(\theta) & 0 & 0 \\
  \sin(\theta) & \cos(\theta) & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{pmatrix}
\]

Folgen mehrere Transformationen aufeinander, werden diese aufmultipliziert. Dadurch kann die in Kapitel 2 angesprochene homogene Transformationsmatrix \( A_i \) durch zwei Rotationen und zwei Translationen wie folgt beschrieben werden:
\[ A_i = \text{Rot}_{z, \theta_i} \text{Trans}_{z, d_i} \text{Trans}_{x, a_i} \text{Rot}_{x, \alpha_i} \]

\[
\begin{pmatrix}
\cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\
\sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & a_i \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & d_i \\
0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\
0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
\cos(\theta_i) & -\sin(\theta_i) & \sin(\theta_i) \cos(\alpha_i) & a_i \cos(\theta_i) \\
\sin(\theta_i) & \cos(\theta_i) \cos(\alpha_i) & \sin(\theta_i) \sin(\alpha_i) & a_i \sin(\alpha_i) \\
0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\
0 & 0 & 0 & 1
\end{pmatrix}
\]
Anhang B
Herleitung der inversen Kinematik

Es gilt

\[ r = \sqrt{x_k^2 + y_k^2} \tag{1} \]

und der Kosinussatz:

\[ r = \sqrt{l_1^2 + l_2^2 - 2l_1 l_2 \cos(\theta_2')} \]

mit \( \theta_2' = \pi - \theta_2 \) folgt:

\[ \cos(\theta_2') = \cos(\pi - \theta_2) = \cos(\pi) \cos(\theta_2) + \sin(\pi) \sin(\theta_2) = -1 \cdot \cos(\theta_2) = -\cos(\theta_2) \]

\[ \Rightarrow \quad r = \sqrt{l_1^2 + l_2^2 + 2l_1 l_2 \cos(\theta_2)} \tag{2} \]

Aus (1) und (2) folgt:

\[ D = \cos(\theta_2) = \frac{x_k^2 + y_k^2 - l_1^2 - l_2^2}{2l_1 l_2} \]

Die Berechnung von \( \theta_2 \) erfolgt mit dem Tangens

\[ \tan(\theta_2) = \frac{\sin(\theta_2)}{\cos(\theta_2)} \]

mit

\[ 1 = \sqrt{\sin^2(\theta_2) + \cos^2(\theta_2)} \Rightarrow \sin(\theta_2) = \sqrt{1 - \cos^2(\theta_2)} \]

folgt
\[ \tan(\theta_2) = \frac{\sqrt{1 - \cos^2(\theta_2)}}{\cos(\theta_2)} = \frac{\sqrt{1 - D^2}}{D} \]

**Berechnung von \( \theta_2 \) mit dem Arkustangens mit zwei Argumenten**

\[ \theta_2 = \arctan2 \left( \pm \sqrt{1 - D^2}, D \right) \]

**Arkustangens mit zwei Argumenten:**

Der Arkustangens mit zwei Argumenten ist eine Funktion, die in vielen Programmiersprachen die Möglichkeit bietet, den Winkel im korrekten Quadranten zu ermitteln, und ist am Funktionswert \( \pm \frac{\pi}{2} \) umkehrbar.

**Definition:**

\[
\arctan2(y, x) = \begin{cases} 
\arctan \frac{y}{x} & \text{für } x > 0 \\
\arctan \frac{y}{x} + \pi & \text{für } x < 0, y \geq 0 \\
\arctan \frac{y}{x} - \pi & \text{für } x < 0, y < 0 \\
\frac{\pi}{2} & \text{für } x = 0, y > 0 \\
-\frac{\pi}{2} & \text{für } x = 0, y < 0 \\
0 & \text{für } x = 0, y = 0 
\end{cases}
\]

**Wertebereich:**

\[ -\pi < \arctan2(y, x) \leq \pi \]

**Berechnung von \( \theta_1 \)**

\[ \theta_1 = \varphi - (\varphi - \theta_1) \]

mit

\[ \varphi = \arctan2(y_k, x_k) \]

und

\[ \varphi - \theta_1 = \arctan2(l_2 \sin(\theta_2), l_1 + l_2 \cos(\theta_2)) \]

folgt

\[ \theta_1 = \varphi - (\varphi - \theta_1) = \arctan2(y_k, x_k) - \arctan2(l_2 \sin(\theta_2), l_1 + l_2 \cos(\theta_2)) \]
Berechnung der Weginkrete dx und dy nach dem ersten und zweiten Strahlensatz.

Es gilt:

\[ \frac{dx}{\Delta x} = \frac{dy}{\Delta y} = \frac{\text{delta}_\text{way}}{s} \]

daraus folgt

\[ dx = \frac{\text{delta}_\text{way} \cdot \Delta x}{s} \]

und

\[ dy = \frac{\text{delta}_\text{way} \cdot \Delta y}{s} \]
Zu den geteichten Punkten $T_1$ und $T_2$ wird der Mittelpunkt $M$ gebildet. Senkrecht auf die Strecke $T_1T_2$ durch den Punkt $M$ wird dann eine Gerade $g$ mit

$$g = M + \lambda \vec{u}$$

gelegt. Die gesuchten Punkte $P_1$ und $P_2$ sind Elemente dieser Geraden. Der Richtungsvektor $\vec{u}$ der Geraden $g$ kann über das Skalarprodukt

$$0 = \vec{u} \cdot \overrightarrow{T_1M}$$

gebildet werden. Es ergibt sich $\vec{u}$ zu

$$\vec{u} = \begin{pmatrix} - (m_y - t_{1y}) \\ m_x - t_{1x} \end{pmatrix}$$

Um die Punkte $P_1$ und $P_2$ bestimmen zu können, muss nun noch der Skalierungs faktor $\lambda$ berechnet werden. Durch die Symmetrie des Racks gilt

$$|\overrightarrow{T_1M}| = |MP_3|$$

mit

$$MP_3 = \overrightarrow{p_3} - \overrightarrow{m} = \overrightarrow{m} + \lambda \vec{u} - \overrightarrow{m} = \lambda \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

Für $\lambda$ gilt somit

$$\lambda = \frac{(m_x - t_{1x})^2 + (m_y - t_{1y})^2}{\sqrt{(t_{1y} - m_y)^2 + (m_x - t_{1x})^2}}$$
Literaturverzeichnis


[10] LABO, Marktübersicht Pipettier-Roboter, LABO Marktübersichten 08/2013,


Danksagung


Bei Herrn Prof. Dr.-Ing. Klaus Webers, der diese Bachelorarbeit betreut hat, möchte ich mich ganz herzlich bedanken.

Herrn Prof. Dr.-Ing. Alexander Steinkogler danke ich für die Übernahme des Amts als Zweitkorrektor.

Mein besonderer Dank gilt Herrn Dipl.-Ing. (FH) Sebastian Wenleder für die Betreuung dieser Arbeit. Seine fachliche Kompetenz und der Freiraum, den er mir zur Verfügung gestellt hat, haben sehr zum Gelingen dieser Arbeit beigetragen.


Abschließend möchte ich meiner Freundin Martina für ihre Geduld und das Korrekturlesen dieser Arbeit bedanken.
Selbstständigkeitserklärung

Hiermit erkläre ich gemäß der Rahmenprüfungsordnung der Hochschule München, dass ich die vorliegende Arbeit mit dem Titel

Prototypische Implementierung einer Handhabungsaufgabe für ein kollaboratives Laborumfeld mit einem SCARA-Roboter

Selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt, sowie wörtliche und sinn-gemäße Zitate als solche gekennzeichnet habe.

Freising, __________________ _________________

(Simon Rauch)