Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Julia Dubcova

Steuerung eines 5-DOF-Handhabungsroboters in Arbeitsraumkoordinaten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Julia Dubcova

Steuerung eines 5-DOF-Handhabungsroboters in Arbeitsraumkoordinaten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.Ing. Andreas Meisel
Zweitgutachter : Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 2. Dezember 2010

Julia Dubcova

Thema der Bachelorarbeit

Steuerung eines 5-DOF-Handhabungsroboters in Arbeitsraumkoordinaten

Stichworte

Kinematik, inverse Kinematik, Vorwärtskinematik, Assistenzroboter, Handhabungsroboter, Roboter, Roboterarm, Denavit-Hartenberg Notation, RPPPR, Teach-in, SCHUNK-Arm

Kurzzusammenfassung

In der vorliegenden Arbeit wurde eine Steuerungsapplikation für einen 5-DOF-Handhabungsroboter entwickelt, die mit Hilfe der inversen Kinematik den Roboterarm an die Position bringt, die nur durch den Abstand, die Höhe und den Gesamtneigungswinkel des Roboterarms beschrieben wird. Die Berechnung der Gelenkwinkel erfolgt in Zylinderkoordinaten.

Es wurde auch eine Teach-in Applikation entwickelt, die einen bestimmten Ablauf der Bewegungen merken und wiederholen kann. Der Roboter fährt verschiedene Positionen manuell gesteuert an, die Positionen werden von dem Roboter erfasst, gespeichert und später wird die komplette Bahn durchgefahren.

Julia Dubcova

Title of the paper

Control system of a 5-DOF-Robot in workspace coordinates

Keywords

kinematics, inverse kinematics, forward kinematics, assistant robot, robot, robotic arm, Denavit-Hartenberg notation, RPPPR, Teach-in, SCHUNK arm

Abstract

In this paper, a control system for a 5-DOF-robot is developed, which brings the robot arm to a specified position, that is only defined by distance, hight and angle of the robot arm. To accomplish that goal, inverse kinematics are applied. The calculations were made in cylindrical coordinates.

Additionally, a teach-in application was developed, that can remember and repeat certain orders of movements. The robot approaches different positions controlled manually, these positions are recognized and saved. Later, the whole path will be approached.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

In der vorliegenden Arbeit wird eine Steuerungsapplikation für einen 5-DOF-Handhabungsroboter entwickelt, die mit Hilfe der inversen Kinematik den Roboterarm an die Position bringt, die nur durch den Abstand, die Höhe und den Gesamtneigungswinkel des Roboterarms beschrieben wird. Die Berechnung erfolgt in Zylinderkoordinaten. Später wird der Handhabungsroboter in dem Living Place von der Hochschule für Angewandte Wissenschaften Hamburg fahren und die Gegenstände bis zu 3 kg transportieren können.

Weiter wird eine Teach-in Applikation entwickelt, die einen bestimmten Ablauf der Bewegungen merken und wiederholen kann. Der Roboter wird verschiedene Positionen manuell gesteuert angefahren, die Positionen werden von dem Roboter erfasst, gespeichert und später wird die komplette Bahn durchgefahren.

1.1 Anwendungsgebiete

Der Bereich Robotik gewinnt immer mehr an Bedeutung in unserem Leben. Man unterscheidet drei Typen von Robotern: autonome Fahrzeuge, Handhabungsroboter und mobile Handhabungsroboter. Bei den autonomen Fahrzeugen ist das Zusammenspiel zwischen Software und Hardware zwingend notwendig. Das Programm übernimmt die komplette Hardwaresteuerung, wie Lenken, Bremsen, Schalten, Beschleunigen usw. Das primäre Ziel von den autonomen Fahrzeugen ist mehr Sicherheit in den Personennahverkehr zu bringen. Auch in der Landwirtschaft, beispielsweise fürs Pflügen oder Säen der Felder werde sie oft eingesetzt. Im Militärwesen, etwa bei der Bombenentschärfung oder in gefährlichen Gebieten, sind weitere Anwendungsgebiete.(Bertsch, 2006)

Ein Handhabungsroboter verfügt über einen Greifarm und besitzt hohe Präzision sowie Belastbarkeit und fährt mit einer hohen Geschwindigkeit. Solche Roboter werden oft in der Automobilindustrie oder am Fließband eingesetzt. An Fertigungsstraßen werden sie bei der Produktion von Gütern verwendet. Aufgrund der festen Bindung an ihren Platz sind ihre Möglichkeiten eingeschränkt.

Mobile Handhabungsroboter werden auch als Assistenzroboter eingesetzt, mit dem Ziel in Bereichen zu arbeiten, in dem es für Menschen zu gefährlich ist. Dies betrifft zum Beispiel

Labore, in denen giftige Gase austreten können oder Viren bzw. Radioaktivität vorkommen kann.

Durch die Überalterung der Gesellschaft in den Industrieländern entsteht für Assistenzroboter ein neues Tätigkeitsfeld. Die Assistenzroboter sollen Menschenbegleiter in unserem

Abbildung 1.1: Der Roboterarm

alltäglichen Leben werden. Sie sollen uns im Service-, Pflege- und Entertainmentbereich unterstützen. Die Serviceroboter sind primär in den Haushalten zu finden, wie zum Beispiel der Staubsauger-, Rasenmäher- oder Spielzeugroboter. In Japan werden aufgrund mangelnden Personals immer mehr Pflegeroboter eingesetzt. Die Pflegeroboter werden in Krankenhäusern, Alters- und Pflegeheimen als Mitarbeiter integriert. Sie können zum Beispiel den Patientenzustand überwachen, physiotherapeutische Übungen durchführen, dem Patienten beim Aufstehen helfen und vieles mehr.

Im Entertainmentbereich werden die Roboter auch immer populärer. Ein ”Asimo” Roboter dirigiert sogar ein Symphonieorchester in der USA. In Osaka(Japan) spielen die Roboter gleichberechtigt in einem Theaterstück. Hiroshi Kobayashi von der Universität Tokio hat eine Lehrerin aus Metall und Kabeln erfunden, die mittlerweile auch in einer Grundschule in

Japan eingesetzt wird. Die vollautomatische Pädagogin kann die Anwesenheit prüfen, Aufgaben stellen, sie kann verschiedene Emotionen zeigen und auf verschiedene Situationen entsprechend reagieren.(Keil, 2010, Wissenschaft) Unser Assistenzroboter (Scitos G5, Abb. 1.1) kann auch in den Fabriken eingesetzt werden, um die Menschen von der monotonen Arbeit, wie zum Beispiel die permanente Ablesung der Messwerte von Schadstoff- und Sauerstoffgehalt in der Luft zu erlösen und Personalkosten zu sparen. Da der Scitos G5 über eine mobile Roboterplattform verfügt, kabelfrei und autonom ist, sowie über 8 Stunden Akkulaufzeit besitzt, sind weitere Einsatzgebiete möglich, wie Getränkeausschank oder Messen-, Museen- und Tierparkführungen. Die Anzahl der Roboter nimmt permanent zu. Zwar hat die Wirtschaftskrise den Bereich auch sehr angeschlagen, aber die Wirtschaft erholt sich schnell und zum Jahr 2013 werden wieder weltweit jährliche Verkäufe von annähernd 100.000 Industrierobotern erwartet. (World of Robotics, 2010, S.1)

1.2 Anwendungsszenario Living Place

Living Place ist ein Projekt der HAW, das sich mit Planung, Konzipierung und Realisierung der intelligenten Umgebung beschäftigt.(Rahimi und Vogt (2009)) Der Scitos G5 wird mit Hilfe der Spracherkennung mit Menschen im Living Place interagieren können. Später soll er die Rolle des Menschenbegleiters übernehmen und sich in der intelligenten Umgebung zurecht finden können. Geplant ist auch, eine Kamera auf den Roboterarm für die Objekterkennung zu installieren.

Jeder von uns kennt die Situation, wenn man aus dem Haus geht und das Gefühl hat, einen Gegenstand vergessen zu haben. Mit Hilfe von Scitos G5 wird es ermöglicht, den Roboter anzurufen und mit seiner Hilfe nachzuschauen, ob das Objekt wirklich zuhause vergessen wurde. Falls das beispielsweise ein Dokument wäre, könnte der Scitos G5 mit Hilfe der Kamera das Dokument abfotografieren und es per E-Mail an den Eigentümer senden.

Die andere allseits bekannte Situation ist, wenn man gerade fest in einem Thema vertieft ist und man nicht von dem Platz weggehen kann, um den Gedanken nicht zu verlieren, man aber gerade Durst bekommt oder einen Gegenstand aus einem anderen Raum benötigt und keiner da ist, der es herbringen könnte. In diesem Fall wäre Scitos G5 die perfekte Lösung. Mit Hilfe des Roboterarms und der später drauf installierten Kamera kann der Scitos G5 das Objekt erkennen, heben, drehen, kippen und transportieren. Um all diese Bewegungen zu ermöglichen, braucht man die inverse Kinematik. Mit Hilfe der Kamera kann man zwar das Objekt finden und seine feste Position bestimmen, aber man muss den Roboterarm in die richtige Position bringen, um das Objekt greifen zu können.

Vorstellbar ist, dass man vergisst, sich um seine Pflanzen zu kümmern. Auch ist ein längerer Urlaub denkbar, in dem die Blumen eingehen würden, wenn sie nicht gegossen werden.

Der Scitos G5 wäre in der Lage, die Pflanzen regelmäßig mit Wasser zu versorgen. Auch Haustiere könnte er füttern und so einen sorgenfreien Urlaub gewährleisten.

Schließlich ist der Aspekt der Vereinsamung der Gesellschaft zu betrachten. Viele Menschen haben eine Tierallergie und können deswegen keine Haustiere besitzen. Durch die Bewegungen in der Wohnung vermittelt Scitos G5 das Gefühl, nicht allein im Raum zu sein. Seine Spracherkennungs- und Sprachanalyseneigenschaften werden ein Vertrauensgefühl verbreiteten. Zwar kann der Roboter die menschlich Wärme nicht ersetzen, aber er kann unterhaltsam und hilfreich im Haushalt sein.

1.3 Schwerpunkte und Projektziele

Es ist oft so, dass man nur die Endstellung des Endeffektors[1] kennt. Für die Steuerung des Roboterarms benötigt man aber alle Gelenkwinkel und deshalb kommt hier die Inverse Kinematik zur Hilfe. Das Ziel des Projektes ist es, einen Lösungsweg zu finden, der möglichst fehlerfrei die Berechnung der Gelenkwinkel des Roboterarms für die neue Position des Endeffektors mit Hilfe des Ansatzes, der auf Denavit-Hartenberg Notation(siehe Kapitel 2.3 Denavit-Hartenberg Notation) basiert, ermöglicht.

Eine weitere Aufgabe ist es, eine Steuerungsapplikation zu schreiben, die die manuelle Steuerung des Arms, sowie die Eingabe der neuen Position des Endeffektors und die Berechnung der neuen Winkel ermöglicht und eine Teach-in Applikation beinhaltet. Dies wird mit der vorgegebenen Qt Software gemacht, die auf C++ basiert und zahlreiche Gestaltungsmöglichkeiten für die Anwendungsoberfläche bietet. Die Software muss stabil und fehlerfrei laufen, sowie zur Vorführung für die Besucher in der Living Place dienen.

[1] das letzte Element einer kinematischen Kette, hier der Greifer

2 Vorwärtskinematik

Der Begriff ”Roboterkinematik” bedeutet ”Bewegungslehre von Punkten und Körpern im Raum”, dabei spielen dynamische Ursachen wie Masse oder Kraft keine Rolle. Es handelt sich um die geometrische Beschreibung von Bewegungen der Gelenkkörper, welche aus einzelnen Teilen bestehen, die durch Gelenke verbunden sind. Die Roboterkinematik unterteilt sich in inverse Kinematik und Vorwärtskinematik. Bei der Vorwärtskinematik handelt es sich um die Berechnung der Haltung, die aus Gelenkstellung und der Gelenkgeometrie resultiert. (Weiß, 2002, S.3)

2.1 Gelenke und Freiheitsgrade

Abbildung 2.1: Die Gelenkarten des SCHUNK-Arms

Aus anatomischer Sicht verbindet ein Gelenk zwei oder mehrere Knochen. Es dient dazu, eine Bewegung zu ermöglichen und die Kräfte zu übertragen. In der Technik sieht es ähnlich

aus. Hier verbindet ein Gelenk zwei Objekte, die sich relativ zueinander bewegen. Die Bewegungen der meisten Gelenke sind auf bestimmte Freiheitsgrade beschränkt, zum Beispiel auf die Torsionsachse, die Knickachse oder das Schubgelenk. Um die Gelenkbewegung zu beschreiben, hat man Freiheitsgrade, "degree of freedom"(DOF) eingeführt. Die Freiheitsgrade des Gelenkes beschreiben die Anzahl der möglichen Bewegungsrichtungen und die Gelenkart. Man unterscheidet zwei Gelenkarten: Rotation und Translation.

Rotation ist eine Bewegung des Gelenkes auf einer kreisförmigen Bahn. Die Module bei dem SCHUNK-Arm sind von 20 bis 25 durchnummeriert. Jedes Modul verfügt über einen Schrittmotor, der sich an der Verbindungsstelle befindet. Diese Verbindungsstellen nennt man Gelenk. Das 20. und das 24. Gelenk des Roboterarms sind typische Rotationsgelenke bzw. Drehgelenke, die sich nur um eine feste Achse drehen können. Die Gelenke 21 bis 23 sind die sogenannten reinen Translationsgelenke bzw. Knickgelenke, welche sich nur linear fortbewegen können. Das Modul 25 ist der Greifer und ist im obigen Sinne kein Gelenk, da es kein Nachfolgermodul gibt.

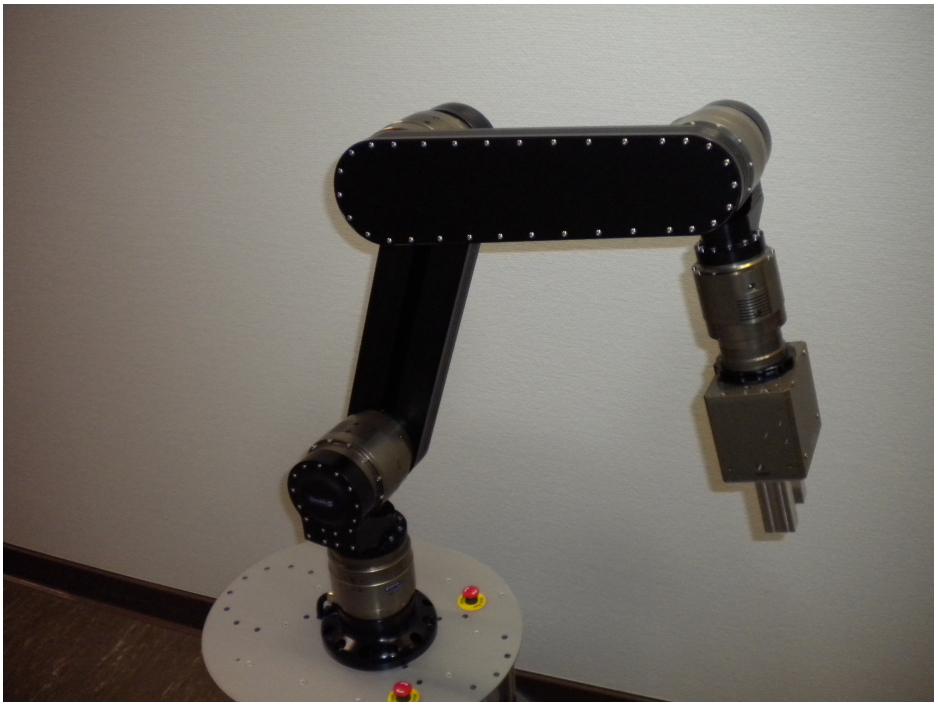


Abbildung 2.2: Der SCHUNK-Arm

Der Freiheitsgrad eines Roboterarms ergibt sich aus der Summe der Gelenke. Wie man auf der Abbildung 2.1 sieht, stehen die Motorachsen bei dem SCHUNK-Arm orthogonal zueinander, dadurch werden Rotations- und Translationsgelenke realisiert. Das untere Gelenk ist ein Rotationsgelenk (Modul 20) und das obere ein Translationsgelenk(Modul 21). Bei dem

SCHUNK-Arm (Abb. 2.2) hat jedes Gelenk nur einen Freiheitsgrad, somit verfügt der Arm über 5 Freiheitsgrade: drei der Translation und zwei der Rotation.

Das besondere an diesem Roboterarm ist, dass die Drehrichtung eines Moduls immer entgegengesetzt der Drehrichtung des vorherigen Moduls ist. Bewegt sich zum Beispiel das Modul 21 bei einem positiven Wert von der Nullstellung nach rechts, so bewegt sich bei dem gleichen Wert das Modul 22 nach links und das nächste Modul würde sich wieder nach rechts bewegen.

2.2 Direkte Herleitung

Ein Roboterarm kann beliebig viele Gelenke haben. Der Schunk-Roboterarm ist ein RPPPR-Roboter. Ein RPPPR-Roboterarm besteht aus zwei Rotationsgelenken (R=Roll) und drei Knickgelenken (P=Pitch). RPPPR steht für die Anordnung der Knick- und Rotationsgelenke.

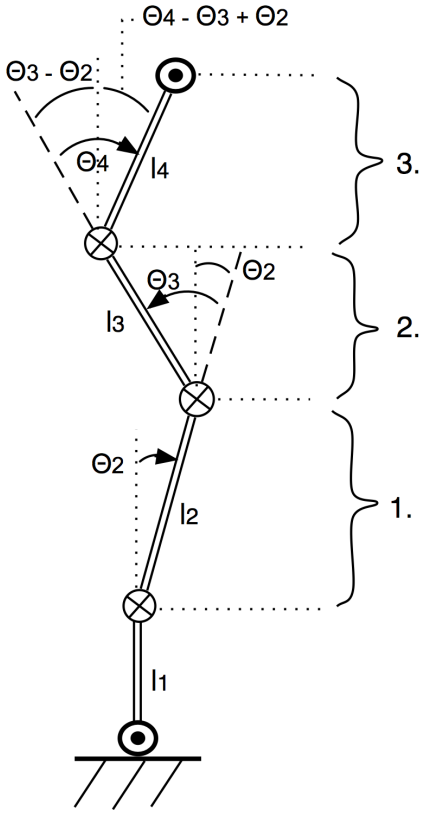


Abbildung 2.3:

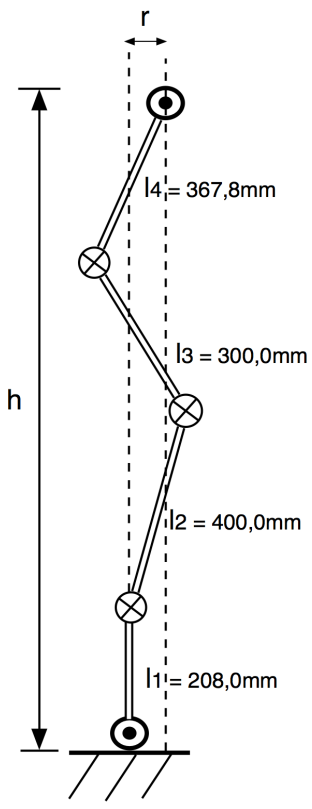


Abbildung 2.4:

Auf der Abbildung 2.3 und 2.4 kann man eine schematische Darstellung des Roboterarms in Zylinderkoordinaten sehen. l beschreibt die Länge eines Moduls, so ist l_1 die Länge des Moduls 20, l_2 ist die Länge des Moduls 21 usw. θ_2, θ_3 und θ_4 entsprechen den Gelenkwinkeln von den Modulen 21 bis 23 (Knickgelenke). Für die inverse Kinematik benötigt man den Radius r, die Höhe h und den Gesamtneigungswinkel des Roboters ϕ_{neig} für die Berechnung der Gelenkwinkel. Zuerst muss man die Parameter direkt herleiten, d.h. man geht davon aus, dass alle Gelenkwinkel bekannt sind. Danach wird die Höhe und der Radius für einzelne Gelenke berechnet. Hier ist ein Bespiel zur Berechnung der Höhe und des Radius für den ersten Abschnitt (Abb. 2.3):

$$h_1 = l_2 \cdot cos(\theta_2) \tag{2.1}$$

$$r_1 = l_2 \cdot sin(\theta_2) \tag{2.2}$$

Im zweiten und im dritten Abschnitte ist die abwechselnde Neigungsrichtung der Gelenke zu berücksichtigen. In folgenden Gleichungen sieht man die Gesamthöhe, den Gesamtradius und den Gesamtneigungswinkel. Da das erste Gelenk ein Rotationsgelenk ist, ist der Radius gleich 0 und die Höhe gleich l_1.

$$h = l_1 + l_2 cos(\theta_2) + l_3 cos(\theta_3 - \theta_2) + l_4 cos(\theta_4 - \theta_3 + \theta_2) \tag{2.3}$$

$$r = l_2 sin(\theta_2) - l_3 sin(\theta_3 - \theta_2) + l_4 sin(\theta_4 - \theta_3 + \theta_2) \tag{2.4}$$

$$\phi_{neig} = \theta_2 - \theta_3 + \theta_4 \tag{2.5}$$

2.3 Denavit-Hartenberg Notation

J. Denavit und R.S.Hartenberg haben eine Notation eingeführt, die eine Möglichkeit bietet, die Position der Gelenke, die eine kinematische Kette bilden, zu beschreiben. Dabei dürfen diese Gelenke nur einen Freiheitsgrad aufweisen. Falls ein Gelenk mehrere Beweglichkeiten hat, muss es durch eine Folge von Gelenken mit einem Freiheitsgrad ersetzt werden. Dieses Verfahren erleichtert vor allem die Berechnung der Vorwärtskinematik, besonders im Bereich der Robotik. (Prof.Dr.-Ing. A. Jahr, 2001, S.1f)

2.3.1 Vorwärtskinematik mit Denavit-Hartenberg Notation

Mit Hilfe der Denavit-Hartenberg Notation (DH-Notation) kann, ausgehend von dem lokalen Koordinatensystem $KS(i)$, die Lage und die Orientierung des nachfolgenden Koordinatensystems $i+1$ unter Verwendung der in Abb. 2.5 dargestellten Parameter bestimmt werden. (Jackel u. a., 2006, S.40 ff) Der Ursprung des $KS(i)$ liegt im gemeinsamen Schnittpunkt der

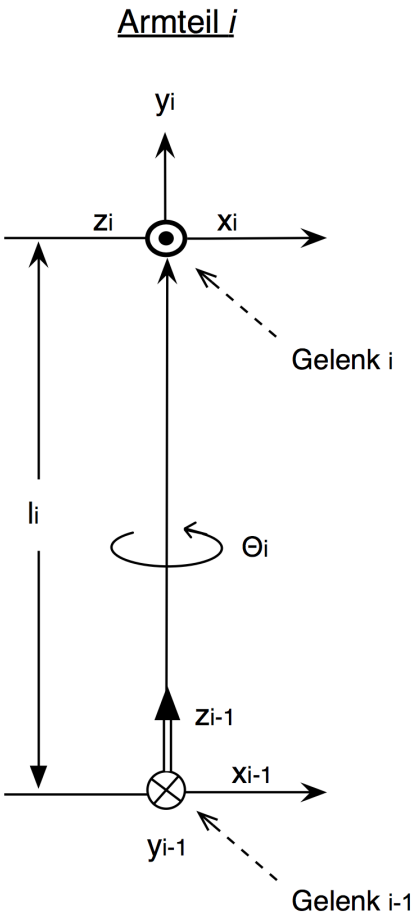


Abbildung 2.5: Beispiel Armteil i

Normalen von den Gelenke i und $i+1$. Auf der Abbildung 2.5 kann man einen Armteil (A_i) sehen, der die Lage des $KS(i)$ in Koordinaten des $KS(i-1)$ beschreibt.

Dabei muss berücksichtigt werden, dass

- die Koordinatensysteme fest in den Bewegungsachsen liegen
- die Motorachse immer als z_i - Achse genommen wird
- die x_i -Achse auf der z_{i-1} steht
- die z_i- Achse mit Hilfe des Rechtssystems[1] bestimmt wird.

Die Parameter θ_i, a_i, d_i und α_i haben folgende Bedeutung:

[1] Als Rechtssystem oder auch rechtshändiges System wird in Mathematik und Physik ein System bezeichnet, dass aus drei räumlichen Vektoren besteht, deren Spatprodukt positiv ist.

- θ_i: motorisierte Rotation um die z_{i-1}- Achse.
- a_i: z- Achsenabstand, hier auf der Abbildung 2.5 a = 0 (schneidende Gelenkachsen). Bei den senkrechten Gelenken ist $a = +/ - \pi/2$.
- d_i: Distanz zwischen zwei benachbarten Gelenkverbindungen in Richtung der z_{i+1}-Achse.
  1. wenn sich die z-Achsen schneiden, ist der Abstand der Schnittpunkt zum Koordinatensystem $i - 1$.
  2. wenn die z-Achsen parallel zueinander sind, ist der Abstand 0.
  3. wenn die z-Achsen ineinander gehen, ist der Abstand gleich $KS(i-1) \rightarrow KS(i)$.
- α_i: Verdrehung der z-Achsen $z_{i-1} \rightarrow z_i$ aus x_i-Richtung gesehen, bei diesem Beispiel beträgt α_i 90°.

Die Transformation des KS_i nach KS_{i+1} erfolgt mit der Matrix $DH_{i,i+1}$, die durch Multiplikation der Transformationsmatrizen $Rot_z(\theta_i)$, $Trans_z(d_i)$, $Trans_x(a_i)$ und $Rot_x(\alpha_i)$ erzeugt wird. (Jackel u. a., 2006, S.40ff) Die Translationsmatrizen kann man auch in eine Matrix zusammenfassen. Es gilt also

$$DH_{i,i+1} = ROT(z, \theta_i) * TRANS(a_i, 0, d_i) * ROT(x, \alpha_i) \tag{2.6}$$

mit

$$ROT(z, \theta_i) = \begin{vmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & 0 \\ sin(\theta_i) & cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix},$$

$$TRANS(a_i, 0, d_i) = \begin{vmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{vmatrix},$$

$$ROT(x, \alpha_i) = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha_i) & -sin(\alpha_i) & 0 \\ 0 & sin(\alpha_i) & cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}.$$

Durch Ausmultiplizieren der Matrizen erhält man eine $DH_{i,1+1}$-Matrix, die wie folgt aussieht

$$DH_{i-1,i} = \begin{vmatrix} cos(\theta_i) & -\sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & a_i cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & a_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{vmatrix}.$$

und die Inverse dazu:

$$DH_{i,i+1} = \begin{vmatrix} cos(\theta_i) & sin(\theta_i) & 0 & a_i \\ -sin(\theta_i)cos(\alpha_i) & cos(\theta_i)cos(\alpha_i) & sin(\alpha_i) & d_i sin(\alpha_i) \\ sin(\theta_i)sin(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & cos(\alpha_i) & -d_i cos(\alpha_i) \\ 0 & 0 & 0 & 1 \end{vmatrix}.$$

Da es sich um homogene Transformationen handelt, werden solche Matrizen auch als A-Matrix bezeichnet. (Siciliano und Khatib, 2008)

Die Denavit-Hartenberg Notation ist nicht immer eindeutig. Bei den parallelen Gelenkachsen gibt es viele Normalen, was die Bestimmung von d_i unmöglich macht. Die Lösung des Problems ist eine zufällige Festlegung von d_i, zum Beispiel $d_i = 0$.

Die Festlegung des Basiskoordinatensystems KS_0, bzw. die des letztes Gliedes KS_n fordert auch eine Sonderregelung, da es keine Vorgänger- bzw. Nachfolgerkoordinaten für diese gibt. Die Regel besagt, dass die z-Achse in die Gelenkachsenrichtung zeigen muss, d.h. x- und y-Achsen können willkürlich gewählt werden, was die Beschreibung des Endeffektors erschwert. (Heimann u. a., 2007, S.201)

2.3.2 Inverse Kinematik mit Denavit-Hartenberg Notation

Bei der inversen Kinematik wird aus einer gegebenen Lage des TCP's[2] die Winkelposition der einzelnen Gelenke bestimmt. Zur Lösung des Problems bieten sich drei Lösungsvarianten an.

1. Algebraische Methode
2. Geometrische Methode
3. Numerische Methode

Algebraische Methode: Beim algebraischen Lösungsansatz wird durch sukzessive Invertierung der DH Transformationsmatrizen die Gelenkwinkelvektorkomponente berechnet.

$$DH_{TCP}(\theta) = DH_1(\theta_1) \cdot DH_2(\theta_2) \cdot \ldots \cdot DH_n(\theta_n) \tag{2.7}$$

Geometrische Methode: Hierbei ist DH_{TCP} eine homogene Matrix, die die Position des Endeffektors beschreibt. Die geometrische Methode basiert auf dem Wissen über die Geometrie des Roboters. Hier wird zum Beispiel versucht, mit Hilfe des Sinussatzes und des Kosinussatzes den Gelenkwinkelvektor zu bestimmen. Die ersten beiden Methoden werden auch als analytisches Lösungsverfahren, das nur für einfache Beispiele geeignet ist, bezeichnet.

Nummerische Methode: Beim numerischen Lösungsverfahren wird iterativ versucht, den Gelenkwinkelvektor zu finden. Es werden Näherungsverfahren angewendet, wie zum Beispiel das Newton-Verfahren oder die Taylor-Entwicklung. Bei fünf Gelenken sind die analytischen Ansätze nicht mehr möglich, daher wird hier die numerische Lösung verwendet. (Deutsche Enzyklopädie, 2005a)

2.4 Bewegungen und Beschränkungen des RPPPR Roboters

Es gibt verschiedene Koordinatensysteme, das bekannteste davon ist das kartesische Koordinatensystem. Dieses System ist nur für wenige Robotersteuerungen geeignet, da sich der Roboter in einer fremden Umgebung immer neu orientieren und den Ursprungspunkt neu festlegen müsste. Daher spielt es hier keine essentielle Rolle. Folglich wurde für die Beschreibung der Positionen der Gelenke des SCHUNK-Arms Zylinderkoordinaten ausgewählt. Zylinderkoordinaten sind im wesentlichen ebene Polarkoordinaten(Kreiskoordinatensystem), die um die Höhe eines Punktes ergänzt werden. Der Unterschied der Polarkoordinaten zum

[2] Tool Center Point = Endeffektor

kartesischen Koordinatensystem besteht darin, dass jeder Punkt auf einer Ebene durch einen Winkel und einen Abstand beschrieben werden kann.

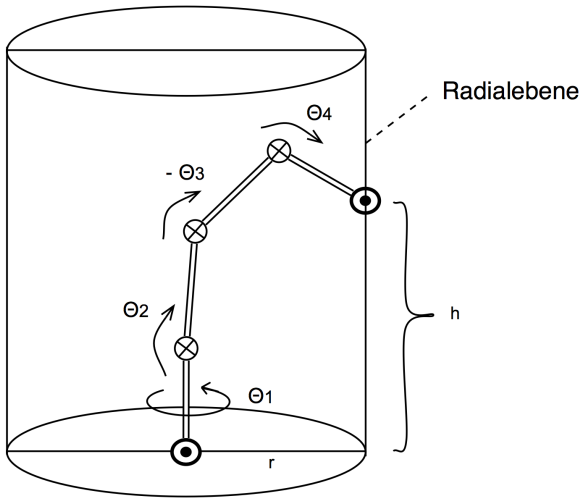


Abbildung 2.6: 5-DOF-Roboterarm in der Radialebene

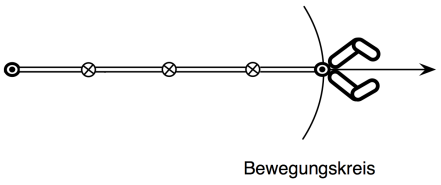


Abbildung 2.7:

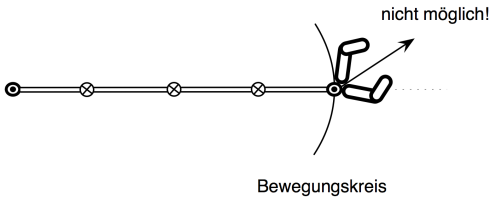


Abbildung 2.8:

Mit den Achsen θ_1, θ_2, θ_3 und θ_4 kann der Endeffektor auf jeden Punkt des Bewegungszylinders gebracht werden, dabei kann die Hand in der gegebenen Radialebene um den Winkel ϕ_{neig} verschwenkt werden. Wie man auf der Abbildung 2.7 sieht, kann der Greifer mit θ_5 um die Unterarmachse verdreht werden. Eine Verschwenkung aus der Radialachse ist aber nicht möglich (siehe Abb. 2.8).

2.5 Konsequenzen

Sofern ein Objekt von oben gegriffen (Abb. 2.9) werden kann, gibt es keine Einschränkungen bezüglich der Objektlage. Mit θ_5 wird die Greiforientierung eingestellt. Falls nicht senkrecht gegriffen wird, so sind nur solche Objekte handhabbar, die senkrecht aus jeder Richtung gegriffen werden können, zum Beispiel ein Trinkglas oder eine Flasche. Dazu muss man noch beachten, dass der Greifer sich maximal nur bis auf 68 mm öffnen lässt, was für den Roboterarm bedeutet, dass die Objekte auch nicht breiter sein dürfen.

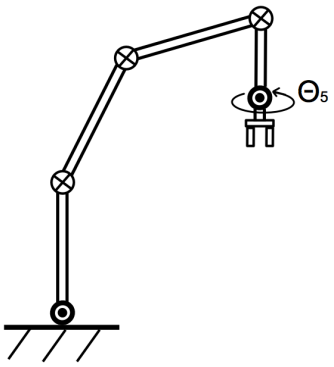


Abbildung 2.9: Greifen von oben

3 Inverse Kinematik

Im Bereich Robotik beschäftigt sich die inverse Kinematik, auch Rückwärtskinematik genannt, mit der exakten Positionierung von Greifarmen. Die inverse Kinematik ermöglicht die Bestimmung der Gelenkwinkel der Armelemente anhand der genauen Lage des TCP's. Die Berechnung der Pose basiert auf mathematischen Gleichungssystemen, die zur Simulation der Gelenkstellung dienen. Dieses Problem ist nicht einfach zu lösen, da die Lage des Endpunktes in unserem Fall auf verschiedene Weise durch drei Winkel beschrieben werden kann. Es führt zu komplizierten Umkehrfunktionen der trigonometrischen Formeln, die nur mit Hilfe von Näherungsverfahren zu bestimmen sind.

3.1 Anwendungsbeispiele

Bei den Industrierobotern ist es öfters so, dass man alles manuell programmieren muss, was bei den Assistenzrobotern nicht erwünscht ist. Der autonome Assistenzroboter sollte selbst in der Lage sein, den Weg zu einem Objekt bestimmen zu können. Dabei spielt die inverse Kinematik eine essentielle Rolle.

Kaum jemand macht sich Gedanken darüber, wie komplex der menschliche Gang ist. Bei jedem Schritt und jeder Bewegung des Körpers muss der Körperschwerpunkt so verlagert werden, dass unser Gleichgewicht gehalten wird. Unser Gehirn steuert in unserem Unterbewusstsein das Ganze bis ins kleinste Detail. Die wirkliche Schwierigkeit erscheint uns erst dann, wenn man den Gang nachzusimulieren und nachzubauen. Bei der Simulation wird der menschliche Gang zuerst vereinfacht, in dem man davon ausgeht, dass sich mit der Bewegung des Fußes, auch der Oberschenkel und der Unterschenkel mitbewegen. So wird der Schwerpunkt während des Ganges immer neu berechnet und verlagert. Dabei wird die Position des Fußes für den nächsten Schritt angegeben und es werden die Winkel für die anderen Glieder der Kette berechnet, so dass sich bei der Positionierung des Fußes die anderen Glieder des Beines entsprechend den Freiheitsgrade anpassen.

Ein anderes Beispiel ist der menschliche Arm. Wenn man die Hand zu drehen versucht, ziehen der Unterarm und der Oberarm automatisch nach. Genauso muss es bei dem Roboterarm passieren und hier kommt die inverse Kinematik zur Hilfe, die sich damit beschäftigt, aus der Position des Endeffektors die Winkel der Gelenke zu berechnen. Außerdem ist es

relevant, dass man nicht die gleiche Rückwärtskinematik bei verschiedenen Robotern anwenden kann. (Schmiedecke, 2010) Darum wird hier in erster Linie die inverse Kinematik für den Schunk-Roboterarm untersucht.

3.2 Ansteuerung des RPPPR-Roboters

Am besten beschreibt man den RPPPR-Roboterarm in Zylinderkoordinaten. Bei den Zylinderkoordinaten handelt es sich um eine Beschreibung des TCP's im Raum durch Radius, Höhe und Neigungswinkel. Es gibt zwei Methoden, den RPPPR-Roboterarm zu steuern: durch die Angabe der Gelenkwinkel oder durch die Angabe der Position des Endeffektors und die daraus folgende Berechnung der Gelenkwinkel.

Bei der Rückwärtskinematik sind Radius r, Höhe h und Gesamtneigungswinkel phi_{neig} schon vorgegeben und werden mit folgenden Formeln durch die Gelenkwinkel beschrieben.

$$r = l_2 sin(\theta_2) - l_3 sin(\theta_3 - \theta_2) + l_4 sin(\theta_4 - \theta_3 + \theta_2) \tag{3.1}$$

$$h = l_1 + l_2 cos(\theta_2) + l_3(\theta_3 - \theta_2) + l_4 sin(\theta_4 - \theta_3 + \theta_2) \tag{3.2}$$

$$phi_{neig} = \theta_2 - \theta_3 + \theta_4 \tag{3.3}$$

θ_2, θ_3 und θ_4 sind zu bestimmen. Die Gleichungen sind nur mit nicht linearen Gleichungssystemen zu lösen, durch einfaches Umstellen sind die Gleichungen nicht lösbar. Im nächsten Kapitel wird die Lösung der Gleichungen noch ausführlich beschrieben.

3.3 Probleme der inversen Kinematik

Es gibt keine einheitliche Lösung des Inverse Kinematik-Problems. Bei manchen Verfahren wächst die Laufzeit so stark, dass sie nicht brauchbar sind. Bei anderen Verfahren kann es dazu kommen, dass die Lösung nicht konvergiert. Die Inverse Kinematik stößt auch immer dann an ihre Grenzen, wenn die kinematische Kette aus fünf und mehr Freiheitsgraden besteht. Da die Gleichung unterbestimmt ist , wird die numerische Methode verwendet. Dabei wird die Laufzeit so groß, dass eine flüssige Animation der Bewegung nicht mehr gegeben ist. Außerdem gibt es oft mehrere "richtige" Lösungen, da verschiedene Konfigurationen der Gelenke zur gewünschten Position des Endeffektors führen können. Es muss also aus mehreren Möglichkeiten für die Armpose eine sinnvolle Gelenkstellung des Arms ausgewählt werden. Allerdings kann es auch passieren, dass sich die Gelenkstellung im unerlaubten Bereich befindet und damit die Lösung unbrauchbar ist. Ein anderes Problem ist, dass das Verfahren nur nach einer Lösung sucht, obwohl es mehrere Lösungen gibt. Beispielsweise

wird beim Newton Verfahren mit der Householder-Transformation[1] immer nur die Nullstelle ausgerechnet, die sich näher an die Stützstelle befindet. Dabei kann die Lösung suboptimal oder sogar falsch sein, da sich ein oder mehrere Gelenke im unerlaubten Definitionsbereich befinden können. (Philippsen, 1998, Kap.8)

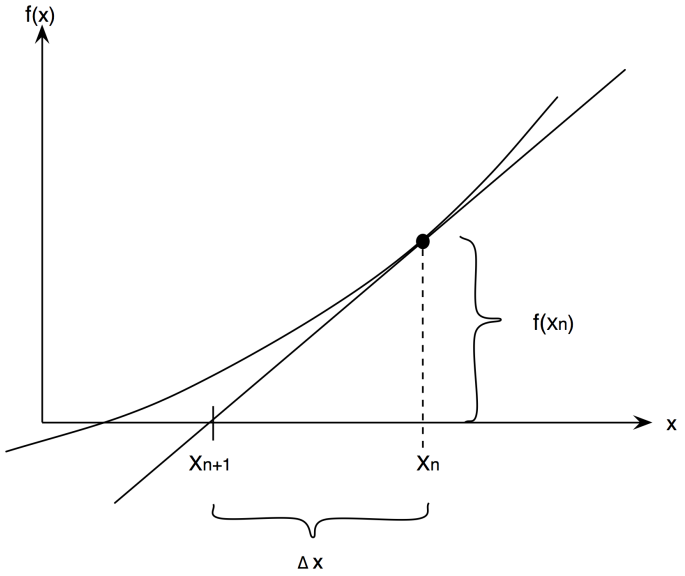
3.4 Entwicklung einer Lösung mit Hilfe des Newton Verfahren

Im Gegensatz zu der direkten Kinematik ist die Position des Endeffektors und der Gesamtneigungswinkel ϕ_{neig} in der inversen Kinematik schon vorgegeben. Um eine Bewegung zu erzeugen, müssen θ_2, θ_3 und θ_4 berechnet und verändert werden. Die Rotationsgelenke werden zuerst nicht betrachtet. Hier wird ein iterativer Lösungsansatz untersucht, der mit Hilfe des Newton Verfahrens gelöst wird.

3.4.1 Einführung und Herleitung

Das Newton Verfahren ist ein Standardverfahren zur numerischen Lösung von nichtlinearen Gleichungen und Gleichungssystemen in der Mathematik. Die Idee des Newtonschen Näherungsverfahrens besteht darin, die Funktion in einem Ausgangspunkt zu linearisieren, beziehungsweise ihre Tangente zu bestimmen, und die Nullstelle der Tangente als bessere Näherung der Nullstelle der Funktion zu verwenden. Wenn man von einer bestimmten Stelle aus die nächste Nullstelle bestimmen möchte, so schaut man, wo die Tangente an den Graphen bei dieser Stelle eine Nullstelle hat, und verwendet dann diese Nullstelle als nächste Näherung. Zur Berechnung des Schnittpunktes der Tangente mit der x-Achse braucht man den Funktionswert und die Steigung der Tangente. Diese werden mit Hilfe der ersten Ableitung der Funktion bestimmt, die auch numerisch gut angenähert werden kann. (Brünner, 2003)

[1] Die Householder Transformation wird zur numerischen Berechnung der QR-Zerlegung für das Lösen der nicht linearen Gleichungssysteme verwendet



$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3.4}$$

$$f'(x_n) = -\frac{f(x_n)}{x_n - x_{n+1}} \tag{3.5}$$

$$f'(x_n) + \triangle x = -f(x_n) \tag{3.6}$$

Für unseren Roboterarm muss zuerst das Nullstellenproblem für das Newton Verfahren formuliert werden.

$$(f_1) : l_2 \cdot sin(\theta_2) + l_3 \cdot sin(\theta_3 - \theta_2) + l_4 \cdot sin(\theta_2 - \theta_3 + \theta_4) - r = 0 \tag{3.7}$$

$$(f_2) : l_2 \cdot cos(\theta_2) - l_3 \cdot cos(\theta_3 - \theta_2) + l_4 \cdot cos(\theta_2 - \theta_3 + \theta_4) + l_1 - h = 0 \tag{3.8}$$

$$(f_3) : \theta_2 - \theta_3 + \theta_4 - \phi_{neig} = 0 \tag{3.9}$$

Für das Newton-Verfahren müssen die Funktionen differenzierbar sein. Darum werden als nächstes alle drei Funktionen nach θ_2, θ_3 und θ_4 abgeleitet.

$$f_1'(\theta_2) = l_2 cos(\theta_2) - l_3 cos(\theta_3 - \theta_2) + l_4 cos(\theta_2 - \theta_3 + \theta_4) \tag{3.10}$$

$$f_1'(\theta_3) = l_3 cos(\theta_3 - \theta_2) - l_4 cos(\theta_2 - \theta_3 + \theta_4) \tag{3.11}$$

$$f_1'(\theta_4) = l_4 cos(\theta_2 - \theta_3 + \theta_4) \tag{3.12}$$

$$f_2'(\theta_2) = -l_2 sin(\theta_2) - l_3 sin(\theta_3 - \theta_2) - l_4 sin(\theta_2 - \theta_3 + \theta_4) \tag{3.13}$$

$$f_2'(\theta_3) = l_3 sin(\theta_3 - \theta_2) + l_4 sin(\theta_2 - \theta_3 + \theta_4) \tag{3.14}$$

$$f_2'(\theta_4) = -l_4 sin(\theta_2 - \theta_3 + \theta_4) \tag{3.15}$$

$$f_3'(\theta_2) = 1 \tag{3.16}$$

$$f_3'(\theta_3) = -1 \tag{3.17}$$

$$f_3'(\theta_4) = 1 \tag{3.18}$$

Danach wird die Jacobi Matrix aufgestellt. Die Jacobi Matrix ist eine Matrix zur näherungsweisen Berechnung mehrdimensionaler Funktionen in der Mathematik. Die m $\times$ n -Matrix beinhaltet sämtliche erste partielle Ableitungen einer differenzierbaren Funktion. (Deutsche Enzyklopädie, 2005b)

$$\underline{J_n} = \begin{pmatrix} \frac{df_1}{d\theta_2} & \frac{df_1}{d\theta_3} & \frac{df_1}{d\theta_4} \\ \frac{df_2}{d\theta_2} & \frac{df_2}{d\theta_3} & \frac{df_2}{d\theta_4} \\ \frac{df_3}{d\theta_2} & \frac{df_3}{d\theta_3} & \frac{df_3}{d\theta_4} \end{pmatrix}$$

Und die daraus resultierende Gleichung sieht folgendermaßen aus.

$$\underline{J}(\theta_{2n}, \theta_{3n}, \theta_{4n}) \cdot \begin{pmatrix} \triangle\theta_2 \\ \triangle\theta_3 \\ \triangle\theta_4 \end{pmatrix} = - \begin{pmatrix} f_1(\theta_{2n}, \theta_{3n}, \theta_{4n}) \\ f_2(\theta_{2n}, \theta_{3n}, \theta_{4n}) \\ f_3(\theta_{2n}, \theta_{3n}, \theta_{4n}) \end{pmatrix}$$

mit

$$\theta_{2n+1} = \theta_{2n} + \triangle\theta_2 \tag{3.19}$$

$$\theta_{3n+1} = \theta_{3n} + \triangle\theta_3 \tag{3.20}$$

$$\theta_{4n+1} = \theta_{4n} + \triangle\theta_4 \tag{3.21}$$

Die Winkel in $f(\theta_{2n}, \theta_{3n}, \theta_{4n})$ sind die aktuellen Winkelpositionen der Gelenke sowie die Winkel in der Jacobi Matrix. Die θ_2, θ_3 und θ_4 sind die neuen Winkel, die bestimmt werden müssen.

3.4.2 Simulation und Analyse des Verfahrens

Bei der Matlab-Simulation wird der Roboterarm mit Hilfe des Robotics Toolbox und folgender Matrix, die aus DH-Parametern α_i, a_i, θ_i und d_i besteht, beschrieben.

$$A = \begin{vmatrix} \pi/2 & 0 & 0 & l_1 \\ \pi & l_2 & 0 & 0 \\ -\pi & l_3 & 0 & 0 \\ \pi & l_4 & 0 & 0 \\ -\pi/2 & 0 & 0 & 0 \end{vmatrix}$$

l_1,l_2, l_3 und l_4 beschreiben die Längen der Module 20 bis 25, dabei wurde die Länge der Module 24 und 25 in l_4 zusammengefasst. Die DH-Matrix wird von Matlab selbst generiert. Man muss aber beachten, dass in Matlab die Nullstellung des RPPPR Roboters (Abb. 3.1) nicht gleich der Nullstellung des SCHUNK-Roboterarms (Abb. 3.2) ist.

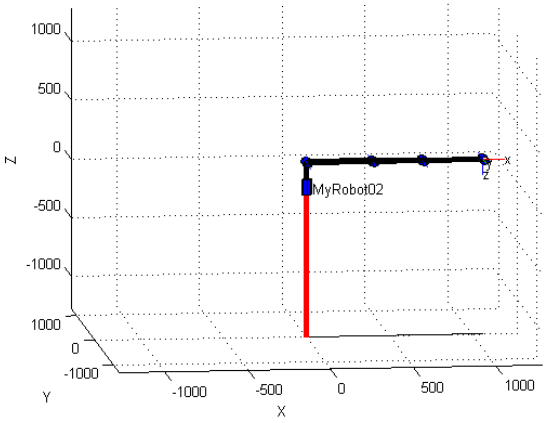


Abbildung 3.1:

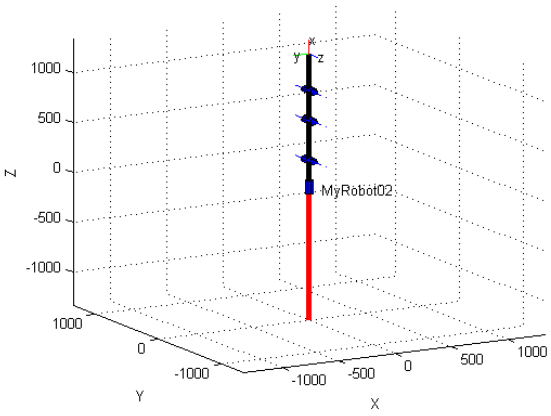


Abbildung 3.2:

Darum benötigt man bei der Matlab-Simulation zwei Methoden zur Umwandlung der Parameter von Steuerungswerten zur inversen Kinematik. Einmal vor der Berechnung der neuen Winkel und einmal nach der Berechnung wieder zurück . Dabei wird 90° zu θ_2 addiert/subtrahiert und bei θ_3 das Vorzeichen umgekehrt. Dadurch erhält man bei der Simulation die Nullstellung des RPPPR-Roboterarms wie bei dem SCHUNK-Roboterarm (Abb. 3.2). Für die Lösung der nichtlinearen Gleichungssysteme wurde bei Matlab eine fertige Funktion verwendet: linsolve(). Diese verwendet als Parameter die Jacobi Matrix und den Vektor mit den Funktionen $f(\theta_{2n}, \theta_{3n}, \theta_{4n})$.

Die Ergebnisse der Simulation haben ergeben, dass bei der Bahnberechnung das Newton-Verfahren sehr stabil und zuverlässig ist, solange die Schritte nicht zu groß sind. Bei größeren Abständen zwischen der aktuellen und der nächsten Position wird oft keine Lösung gefunden. Da es sich nur um ein Annäherungsverfahren handelt, es also mehrere Lösungen gibt, passierte es oft bei größeren Abständen, dass der Arm plötzlich nach unten kippte. Dabei stimmten alle Werte (Radius, Höhe, Neigungswinkel). Das erklärt sich dadurch, dass es mehrere Nullstellen gibt und man die nächste Position zu weit von der aktuellen Position wählt. So wurde nicht die Annäherung an die nächstliegende Nullstelle gesucht, sondern zu einer anderen.

4 Teach-in Anwendung

4.1 Einleitung in die Teach-in

Die Teach-in Applikation ist eine Anwendung, die das Erlernen von Abläufen ermöglicht. In der Robotik bedeutet dies, dass der Roboter mit Hilfe der manuellen Steuerung an eine gewünschte Position geführt und jede dieser Positionen gespeichert wird. Das Teach-in wird so lange durchgeführt, bis ein gesamter Arbeitszyklus einmal durchlaufen ist. Am Ende ist der Roboterarm in der Lage, alle gespeicherten Positionen autonom nachzufahren und den Ablauf beliebig oft mit wählbarer Geschwindigkeit und Beschleunigung zu wiederholen. (Hesse, 1998) Oft werden Roboter mit Kameras oder anderen Sensoren ausgerüstet. Dann kann die Ausführung entsprechend der Kameradaten oder der Sensordaten variieren.

4.2 Mögliche Realisierungsansätze

Es gibt bildgestützte und positionsbasierte Teach-in-Anwendungen. Das Ziel des bildgestützten Teach-in ist das Bewegen des Endeffektors zum Objekt, was mit Hilfe der Analyse der Bilder, welche die Roboterkamera liefert, ermöglicht wird. Dabei wird über die Bildverarbeitung das Objekt detektiert und lokalisiert. Die Lokalisierung des Objektes alleine sagt noch nichts über die Hindernisse aus, die auf dem Weg des Greifers zum Gegenstand liegen können. Soll der Roboterarm beim Greifen des Zielobjektes Kollisionen vermeiden, müssen Zielführung, Zielverfolgung und Kollisionsvermeidung aktiviert und koordiniert werden. (Kraiss, 2005)

Da aber der SCHUNK-Arm noch keine Kamera zur Verfügung hat, muss hier ein positionsbasiertes Teach-in realisiert werden. Das positionsbasierte Teach-in verwendet keine Hinderniserkennung. Der Roboterarm wird manuell an verschiedene Positionen des Bewegungsablaufs gesteuert, der später wiederholt werden soll und die Positionen werden in einer Datenbank abgespeichert. Nach dem Beenden des Bewegungsablaufs kann der Arm alle Positionen autonom nachfahren. Die gespeicherten Gelenkpositionen werden einzeln nacheinander aus der Datenbank gelesen und ausgeführt.

4.3 Anwendungsgebiete

Eingesetzt werden positionsbasierte ”geteachte” Industrieroboter in vielen Bereichen der Fertigung, beispielsweise als Handhabungseinrichtung zum Stapeln (Stapelrobopter), Verpacken und Montieren. Sie können auch Teile entnehmen oder als Schweißroboter zum Bahnschweißen, Laserstrahlschweißen oder Bolzenschweißen eingesetzt werden.

Ein weiteres Anwendungsgebiet von solchen Robotern ist in der Messtechnik – Messen und Testen von verschiedenen Materialien, Gasen, Temperaturen usw. In der Automobilindustrie werden ”geteachte” Roboter auch zum Lackieren, Schleifen oder zum Polieren verwendet.

Die Industrieroboter, die mit dem bildgestützten Teach-in arbeiten, werden zur Unterscheidung von mehreren Farben eingesetzt, beispielsweise bei der Sortierung und in der Qualitätsüberwachung bei den Prozessen, wo die Farbe als Hauptkriterium gilt. Auch bei der Objekterkennung kann bildgestütztes Teach-in verwendet werden. So könnte der Roboter anhand der erlernten Bildparameter des Objektes, dieses in einer Menge von anderen Objekten unterscheiden und gegebenenfalls die Klassifizierung der Gesamtmenge durchführen. Zum Beispiel könnte ein derartiger Roboter in großen Lagern die Ware bildbasierend sortieren.

4.4 Anforderungen an die Teach-in-Anwendung

Die Hauptanforderungen sind, dass das gesamte Steuerungsprogramm nach Vorgabe in Qt (C++) geschrieben werden soll. Der Roboterarm soll alle Positionen nacheinander nachfahren. Es ist außerdem wichtig, dass jede Trajektorie komplett in einer Textdatei abgespeichert werden kann, so dass die Datei zu jeder Zeit abrufbar ist. Der Dateiname darf nur Buchstaben, Zahlen, ’ _ ’ und ’ . ’ enthalten. Es soll auch möglich sein, die Positionen über manuelle Steuerung, sowie über die ’Inverse Kinematik’ zu speichern. Die Positionen sollen in einer Tabelle abgespeichert werden.

Die Tabelle soll folgende Funktionen haben:

- sie soll die Winkelpositionen aller Gelenke enthalten, dabei sollen die Winkel in Grad angezeigt werden
- die Werte sollen durch den Benutzer veränderbar sein
- leeren der Tabelle soll möglich sein
- löschen einer ausgewählten Position soll ermöglicht werden
- speichern der Tabelle in eine Datei

- laden der Tabelle mit Werten aus einer Datei

Die Teach-in-Steuerung soll über einen ”Start”-Knopf verfügen, der den Ablauf der Trajektrorie startet und ihn in einer Endlosschleife wiederholt, bis man den ”Stopp”-Knopf drückt. Während des Ablaufs soll die GUI funktionsfähig und ansprechbar bleiben.

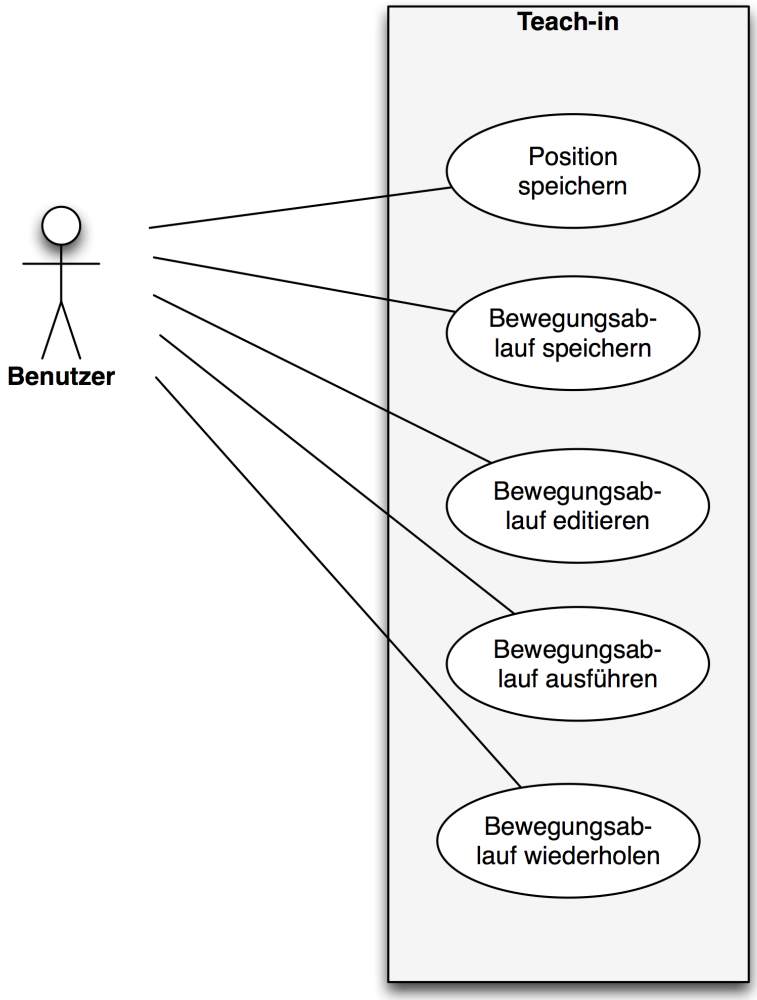


Abbildung 4.1: Use-Case-Diagramm für das Teach-in

5 Realisierung der Steuerungssoftware

Nach der Vorgabe wird die GUI in Qt entwickelt, hierfür wird Qt Designer verwendet. Qt Designer ist eine OpenSource-Software, die ausreichende Möglichkeiten bietet, die GUI nach eigenen Vorstellungen zu gestalten. Um die GUI-Elemente mit Leben zu füllen, kann man die Entwicklungsumgebung Qt Creator oder Eclipse mit dem Qt-Plugin benutzen. Die GUI wird als ein Tabwidget aufgebaut. Es gibt insgesamt drei Tabs: 'Inverse Kinematik', 'Manuelle Steuerung' und 'Teach-in'. In der Abbildung 5.1 sieht man den Klassenaufbau der Steuerungssoftware. Die Klasse **Main** startet die Initialisierung des Roboters und die Applikation, außerdem wird ein Objekt von der Klasse **Mainwindow** erstellt. Die Klasse **Mainwindow** ist für die GUI-Funktionalität zuständig. Hier wird die GUI erstellt und die GUI-Objekte mit Leben gefüllt. Bei dem Entwerfen der GUI wurden alle Buttons mit Signale und Slots versehen. Signale beschreiben, bei welchem Knopfzustand etwas passieren soll, zum Beispiel, wenn der Knopf gedrückt ist. Die Slots sind die Methoden, die auf das Signal reagieren. Diese Methoden werden als **public slots** in **Mainwindow.h** deklariert und in **Mainwindow.C** implementiert. Beim Erstellen des Objektes von der Klasse **Mainwindow** wird ein Thread (**GuiUpdate_thread**) gestartet, der für die Aktualisierung der aktuellen Gelenkwinkelpositionen und die Berechnung der Höhe, des Radius und des Gesamtneigungswinkels im Tab 'Manuelle Steuerung' zuständig ist. Der Rest der Klassen wird von den Methoden der einzelnen Tabs für bestimmte Signale benötigt.

5.1 Tab 'Inverse Kinematik'

5.1.1 Implementierung

Für die Implementierung der inversen Kinematik gibt es ein extra Tab 'Inverse Kinematik' (Abb. 5.2). Als Eingabemöglichkeiten stehen Textfelder für die Eingabe der Höhe, des Radius und des Gesamtwinkels zu Verfügung. Mit dem 'Refresh'-Button ist es möglich, sich die aktuellen Werte (r, h, ϕ_{neig}) anzeigen zu lassen, sie werden aus den aktuellen Gelenkpositionen berechnet. In die Textfelder werden auch die neuen Parameter eingegeben und nach Knopfdruck auf 'Calculate' werden in den unteren Textfeldern die neu ausgerechneten Gelenkwinkel für die Module 21, 22 und 23 dargestellt. Für die anderen Module werden

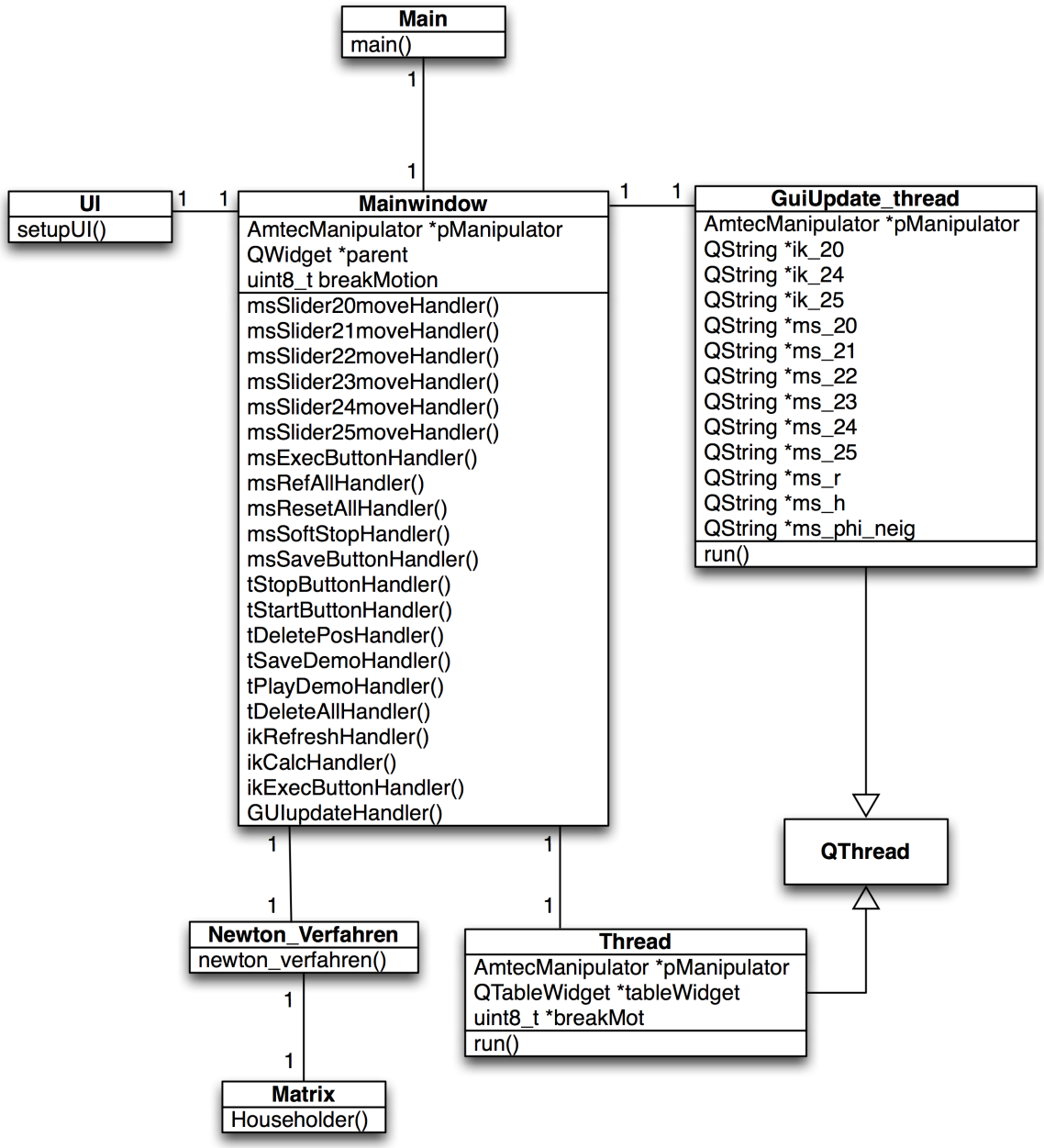


Abbildung 5.1: Klassendiagramm für die gesamte Steuerungssoftware

die Werte der aktuellen Position übernommen. Wie man in folgendem Klassendiagramm (Abb.5.3) sehen kann, erfolgt der Hauptaufruf aus der **Mainwindow** - Klasse. Die Berechnung der neuen Winkel wird in der Klasse **Newton_verfahren** implementiert. Für die Lösung der nichtlinearen Gleichungssysteme wurden Householder-Transformationen aus der Klasse **Matrix** eingesetzt. Das Householder-Verfahren erlaubt es, eine symmetrische Matrix n$\times$n A

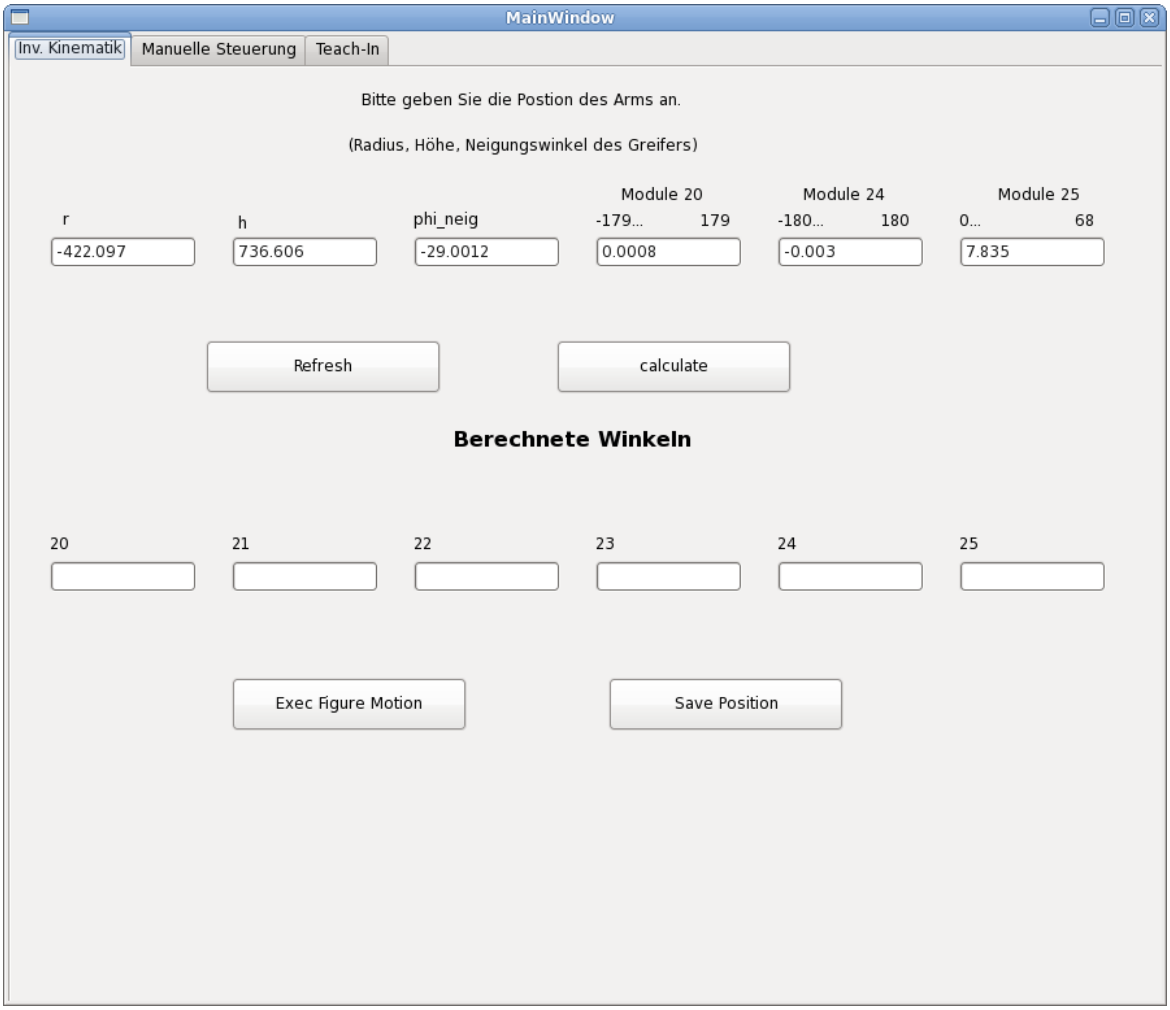


Abbildung 5.2: 'Inverse Kinematik' - Tab Screenshot

$= (a_{i,j})$ in eine Tridiagonalmatrix[1] umzuwandeln und damit das nicht lineare Gleichungssystem lösbar zu machen. (Malek, 2010)

5.1.2 Tests

Die inverse Kinematik funktioniert stabil, solange der Abstand zwischen der aktuellen Position und der nächsten Position nicht zu groß ist. Bei konstanter Höhe und konstantem Gesamtneigungswinkel ist eine Änderung des Radius von bis zu 50cm möglich. Bei konstantem

[1] Eine quadratische Matrix, die nur in der Diagonalen und in den beiden ersten Nebendiagonalen Einträge ungleich Null hat.

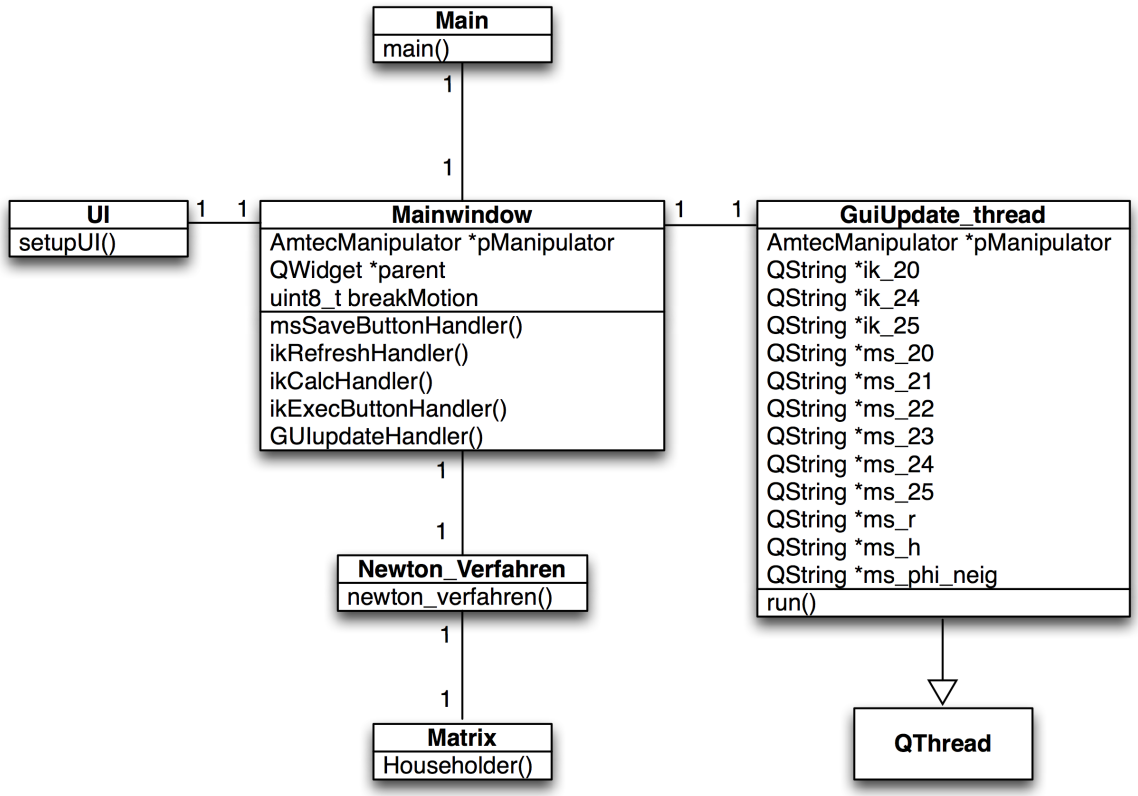


Abbildung 5.3: Klassendiagrammausschnitt für die Inverse Kinematik

Radius und konstantem Neigungswinkel ist eine Veränderung der Höhe von bis zu 30cm erlaubt. Wenn die Höhe und der Radius konstant sind, kann der Gesamtneigungswinkel bis zu 50° verändert werden. Empfohlen ist eine Änderung der einzelnen Parameter von bis zu 20cm. Man muss beachten, dass der Radius auch negativ werden kann, da der Wertebereich von π bis $-\pi$ geht und damit der Greifer in beliebige Positionen der Zylinderkoordinaten gebracht werden kann.

Da die berechneten Gelenkwinkel im Bereich von 0° bis 360° liegen, erfolgt anstelle der ausschließlichen Überprüfung der Winkel gleichzeitig auch eine Anpassung der Werte im Bereich von π bis $-\pi$. Das geschieht über die Funktion

$$\theta_i = atan2(sin(\theta_i), cos(\theta_i)) \tag{5.1}$$

Falls die berechneten Winkel nicht im erlaubten Wertebereich liegen, erscheint eine Warnung in der GUI, damit sichergestellt wird, dass der Arm keine unerlaubten Posen annimmt.

5.2 Tab ’Manuelle Steuerung’

5.2.1 Implementierung

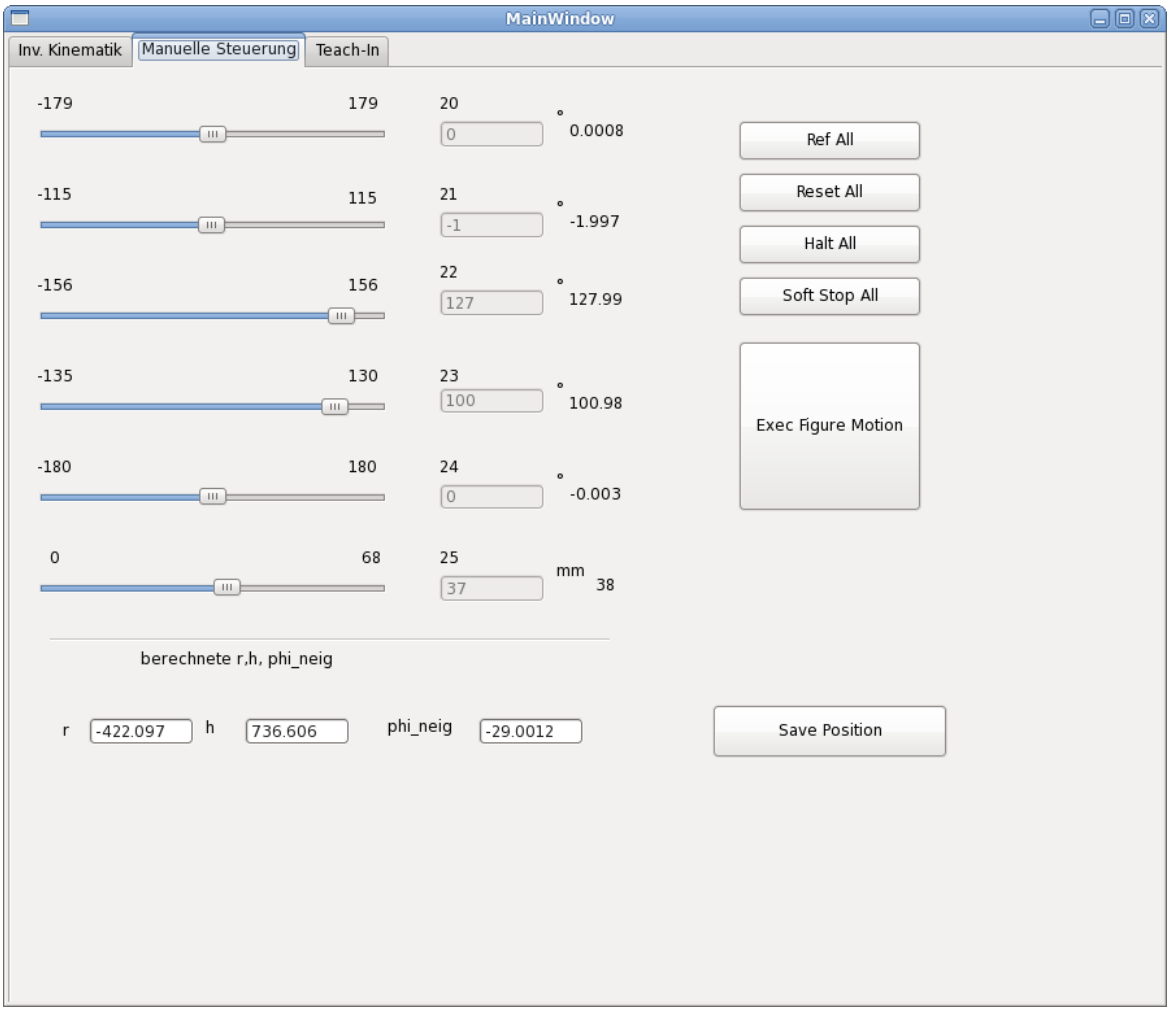


Abbildung 5.4: Tab ’Manuelle Steuerung’ - Screenshot

Der Tab ’Manuelle Steuerung’ (Abb. 5.6) ist mit horizontalen Slidern für jedes Modul ausgestattet, so dass jeder beliebige Wert in Grad aus dem Wertebereich des entsprechenden Moduls einstellbar ist. Außerdem ist der Tab mit Buttons ausgestattet, die folgendes ermöglichen: alle Module referenzieren (’Ref All’), Zurücksetzen aller Module (’Reset All’), Notaus (’Halt All’). Bei Notaus wird bei jedem Modul ein Errorflag gesetzt und die Module können erst nach einem Zurücksetzen wieder in Bewegung genommen werden. Es gibt außerdem ein ’Soft Stop All’ Knopf, der auch zum Anhalten des Roboterarms dient, aber hier werden

keine Errorflags bei den Modulen gesetzt. Im Gegensatz zu 'Halt All' wird die Bewegung bis zum Ende ausgeführt und erst danach bleibt der Roboterarm stehen. Der 'Exec Figure Motion' Knopf ist zum Ausführen der manuell in der GUI eingestellten Winkel. Im separatem Thread (**GuiUpdate_thread**) läuft die Aktualisierung der aktuellen Gelenkwinkelpositionen, so dass, wenn man beispielsweise im Tab 'Inverse Kinematik' die Stellung des Roboters verändert, man im Tab 'Manuelle Steuerung' die aktuellen Werte sehen kann (Abb. 5.5). Die MetraLabs-Bibliothek benutzt für alle Gelenkpositionen Bogenmaß, da aber der normale Benutzer sich besser eine Position in Grad vorstellen kann, wird in der gesamten GUI mit Grad gearbeitet.

Eines der wichtigsten Konzepte von Qt besagt, dass man die Hardwaresteuerung und die GUI-Steuerung in den Nebenthreads immer von einander trennen muss, darum werden im **GUIupdate_thread** nur die Modulwerte aus dem CAN-Bus abgelesen. Daher bekommt der **GuiUpdate_thread** als Parameter für die Threadsynchronisation einen Zeiger auf die QString Variablen, die nur in dem Thread mit den aktuellen Gelenkwinkeln beschrieben werden können. Danach wird ein Signal **GUIupdate** an den Main-Thread geschickt. Zur Verarbeitung des Signals gibt es einen Slot in der **Mainwindow** - Klasse, der die GUI im jeweiligen Tab mit den aktuellen Werten der Gelenkwinkelposition des Robotersarms aktualisiert. In dem **GuiUpdate_thread** wird auch der neue Radius, die neue Höhe und der neue Gesamtneigungswinkel berechnet und in dem Tab 'Manuelle Steuerung' angezeigt.

5.2.2 Tests

Bei der manuellen Steuerung kann der Benutzer keine unerlaubten Gelenkwinkel angeben, da mit den horizontalen Slidern für jedes Modul der erlaubte Wertebereich beschränkt ist.

Die aktuelle Gelenkwinkelposition, sowie der aktuelle Radius, die Höhe und der Gesamtneigungswinkel im Tab 'Manuelle Steuerung' werden jede Sekunde aktualisiert.

Wenn man die Gelenkwinkel verändert und den 'Exec Figure Motion' Knopf betätigt, bewegen sich die Gelenke im synchronen Modus, d.h. sie fangen gleichzeitig an, sich zu bewegen. Allerdings beenden sie die Bewegung zu unterschiedlichen Zeitpunkten. Während der Bewegung des Arms ist es möglich, die Werte zu verändern; der Roboterarm übernimmt sofort die neue Werte für die Bewegung.

Wenn der Notaus-Knopf gedrückt ist, wird das Programm zwar gestartet, aber es werden nicht alle Module gefunden. Es erscheint eine Meldung im Tab 'Manuelle Steuerung', dass der Notaus-Knopf gedrückt wurde und das Programm neu gestartet werden muss.

Falls man den Notaus-Knopf während des Betriebs drückt, wird ein Errorflag gesetzt, der Roboter wird komplett angehalten. Solange man den Notaus-Knopf nicht wieder ausschaltet

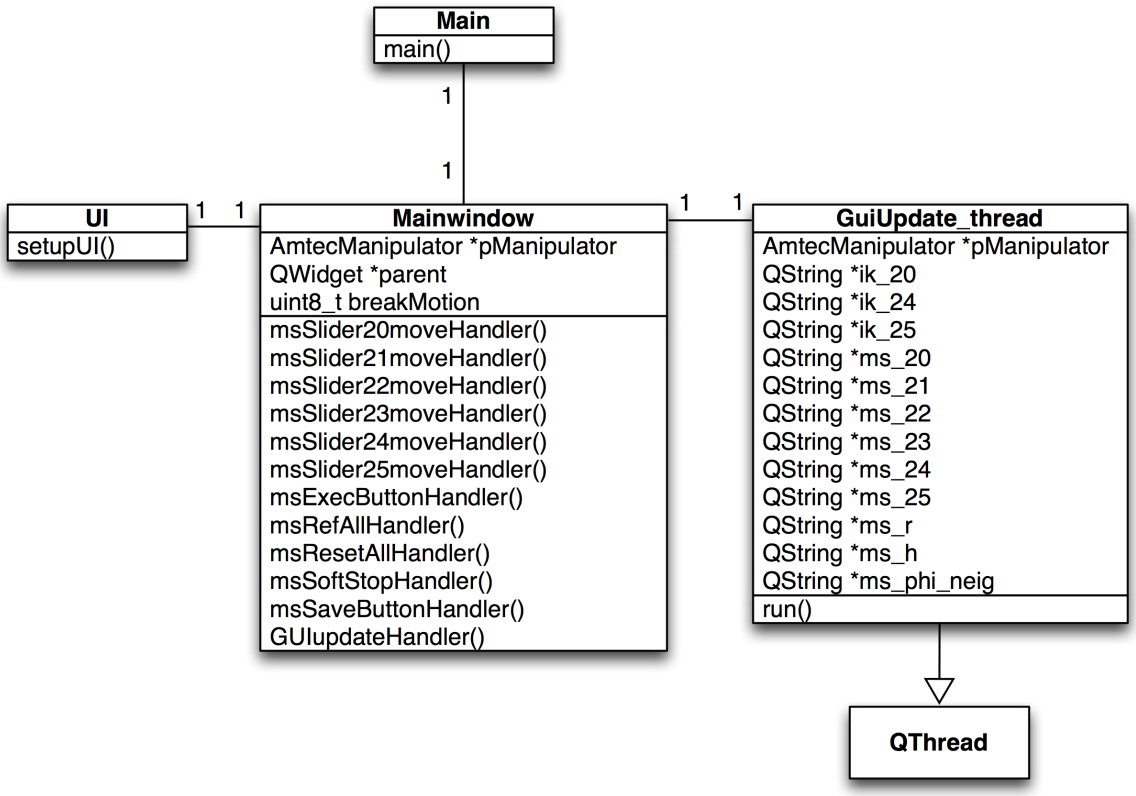


Abbildung 5.5: Klassendiagrammausschnitt zur Manuellen Steuerung

und den ’Reset All’-Knopf drückt, bleiben die Module inaktiv. In der GUI erscheint die Meldung, dass der Notaus-Knopf betätigt wurde. Es wird unterschieden, ob der Notaus-Knopf oder ’Halt All’ gedrückt wurde. Bei ’Halt All’ erscheint auch eine entsprechende Fehlermeldung, bei der der Anwender darauf hingewiesen wird, den ’Reset All’ zu drücken, um die Module wieder in Betrieb zu nehmen.

5.3 Tab ’Teach-in’

5.3.1 Implementierung

In dem UML-Diagramm (Abb.5.7) kann man sehen, dass die Steuerung der GUI Elemente des Teach-In-Tabs in **Mainwindow.C** statt findet. Mit dem Knopf ’Save as Demo’ wird zuerst überprüft, ob in dem rechts stehenden Feld ein Dateiname eingegeben wurde. Falls der Dateiname nicht die Endung .txt hat, wird sie hinzugefügt. Danach wird die komplette Tabelle in diese Datei gespeichert und im Ordner ”demo/” abgelegt. Mit dem Knopfdruck auf

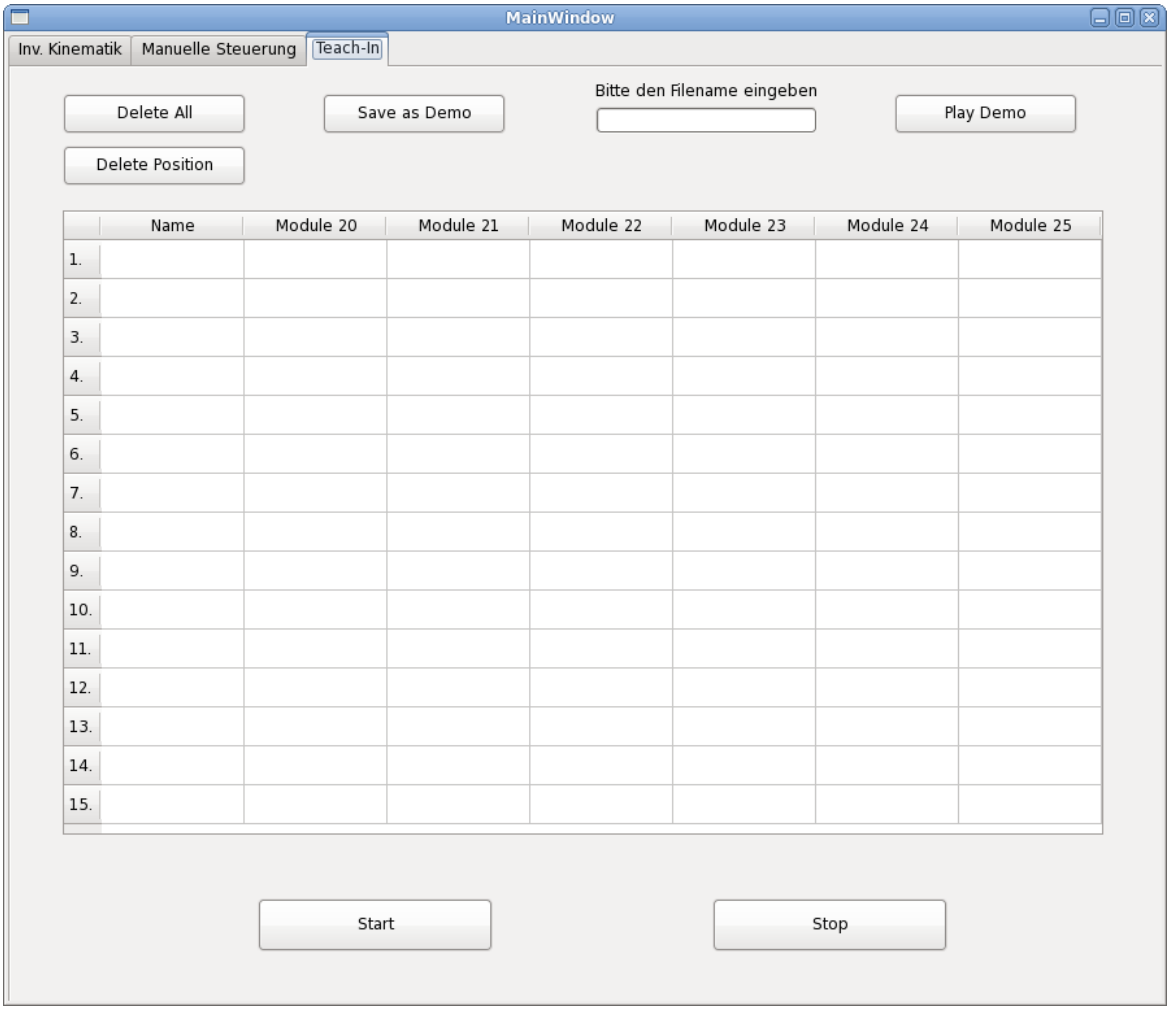


Abbildung 5.6: Tab ’Teach-in’ - Screenshot

’Play Demo’ wird zuerst das Feld mit dem Dateinamen überprüft. Hier wird darauf geachtet, ob es leer ist und ob der Dateiname vollständig mit der .txt Endung angegeben wurde. Im Anschluss wird nach der Datei im Ordner ”demo/” gesucht. Wenn die Datei existiert, wird daraus die Tabelle geladen. Mit dem ’Start’- Knopf wird ein Thread (**Thread.C**) gestartet, der für das zeilenweise Auslesen der Werte aus der Tabelle und für die synchrone Ausführung der Befehle durch den Roboterarm zuständig ist. Mit Hilfe des Threads bleibt die GUI während der Bewegung des Roboters ansprechbar und kann auf die Eingaben des Benutzers reagieren.

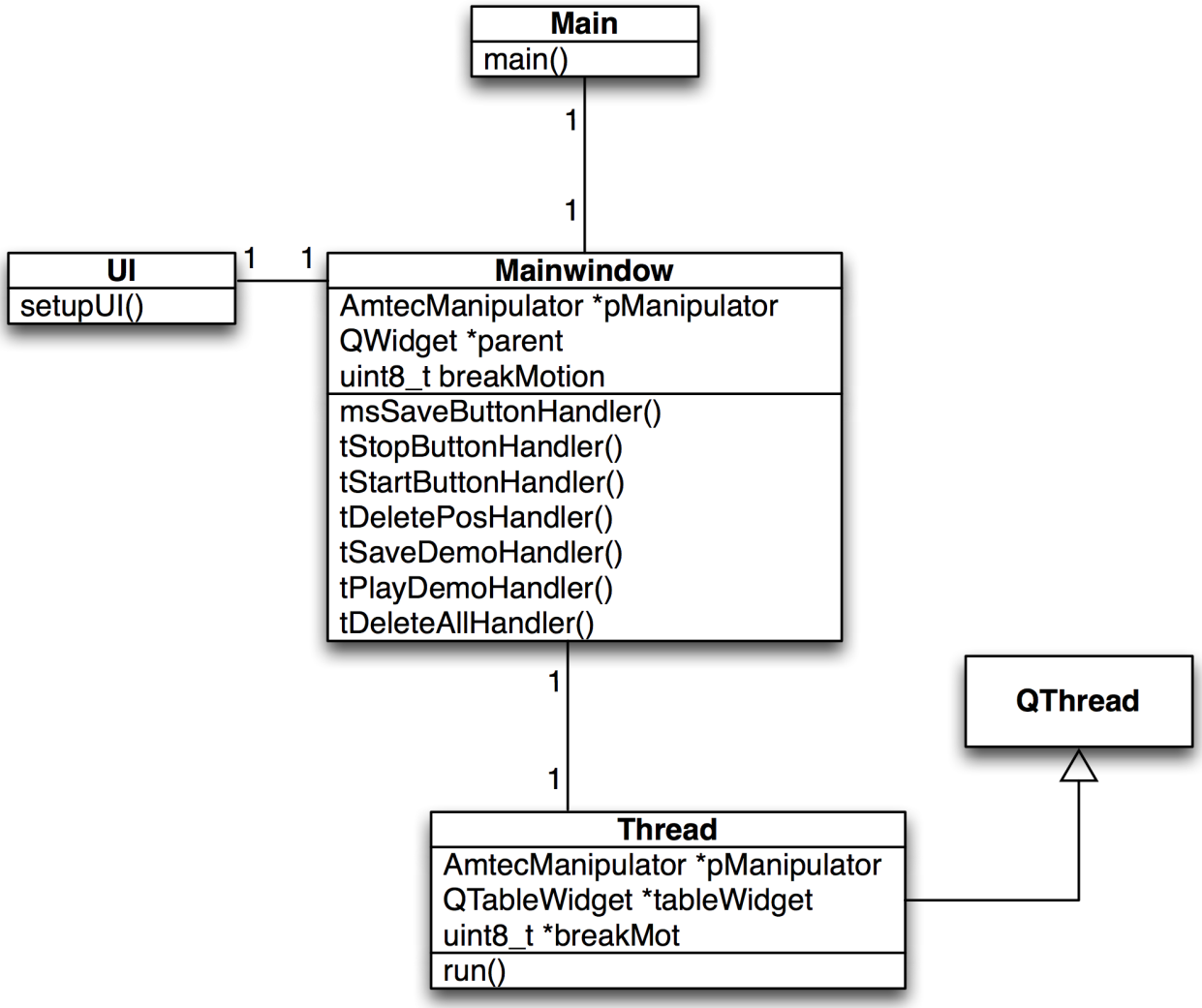


Abbildung 5.7: Klassendiagrammsausschnitt für die Teach-in

5.3.2 Tests

Zuerst wird getestet, ob die Dateien mit anderen Endungen, also nicht .txt, gespeichert und wieder ausgelesen werden können. Das Ergebnis des Tests ist, dass der Parser nur Textdateien erlaubt, die auch den Anforderungen entsprechen. Außerdem können beim Dateinamen nur Zahlen, Groß- und Kleinbuchstaben, sowie '_' und ein '.' für die Endung verwendet werden. Als nächstes wird getestet, ob die gespeicherte Tabelle verändert und anschließend wieder abgespeichert werden kann. Da vor dem Abspeichern immer überprüft wird, ob die Datei schon existiert und in diesem Fall zuerst gelöscht und danach wieder angelegt wird, entsteht beim Verändern der Datei keine Dateninkonsistenz. Beim Auslesen von leeren Da-

teien oder Dateien mit fehlerhaftem Inhalt wird eine Meldung eingeblendet, dass die Datei nicht lesbar oder leer ist. Es erscheint auch eine Fehlermeldung, wenn die Datei nicht existiert.

Wenn man den 'Start'-Knopf betätigt, fängt der Arm sofort an, sich zu bewegen. Die gesamte Bahntrajektorie wird aus der Tabelle ausgelesen und in einer Endlosschleife wiederholt, allerdings muss man beim Anhalten des Roboters beachten, dass dies wie eine 'Halt All'-Funktion wirkt. Der Arm bleibt sofort stehen und das Errorflag wird gesetzt. Um den Arm wieder in Bewegung nehmen zu können, muss man 'Reset All' im Tab 'Manuelle Steuerung' drücken. Die Ausführung der Bewegung verläuft in einem Thread und man muss beachten, dass für eine abgeschlossene Bewegung des Roboterarms im Thread immer überprüft wird, ob der Arm noch in Bewegung ist bevor die nächste Zeile ausgelesen wird. Qt bietet an der Stelle keine Threads-Liste an und darum wird hier eine **volatile** Variable benutzt, um den Thread zu terminieren. Die Überprüfung, ob der Arm in Bewegung ist, kann im schlimmsten Fall bis zu 5 Sekunden dauern, da die neue Abfrage der volatile Variable über die Funktion getModules() aus der Bibliothek von Metralabs immer ein sleep() von einer Sekunde benötigt, um die Daten vom CAN-Bus vollständig abzulesen. Darum ist es empfehlenswert, den Knopf 'Ref All' erst nach 5 Sekunden zu betätigen, um unerwünschte und unvorhersehbare Bewegungen des Arms zu vermeiden.

6 Evaluierung der Gesamtergebnisse

6.1 Echtzeitfähigkeit

Die Echtzeitfähigkeit ist die Fähigkeit eines Systems, auf Ereignisse aus der Systemumwelt innerhalb definierter Zeitschranken zu reagieren und bestimmte Berechnungen in einer festgelegten Zeit abzuschließen. Wird die Zeitanforderung nicht eingehalten, sind die Ergebnisse damit unbrauchbar.

Die Scitos G5-Platform ist mit einem embedded PC ausgestattet. Das Betriebssystem, Linux Fedora 12, ist aber keine RTOS (Real Time Operating System), d.h. eine Berechnung ist immer korrekt, egal wie lange sie dauert. Es gibt keine Zeitschranken und die Zeit beeinträchtigt nicht die Richtigkeit der Ergebnisse. Die Interrupts des Betriebssystems können unter Umständen bevorzugt behandelt werden. Es ist ein Event-triggered System, das heißt das System reagiert auf externe Ereignisse, beispielsweise wird ein Knopf gedrückt oder die Zeitscheibe ist abgelaufen. Für das Steuerungsprogramm des Roboterarms bedeutet dies, dass das zeitliche Verhalten teilweise schwer vorhersagbar ist, allerdings ist eine schnelle Reaktion auf externe Ereignisse gegeben. Scheduling ist stets dynamisch. (Korf, 2009, F.22) Das System ist also nicht echtzeitfähig, was unter Umständen zu Problemen führen könnte.

6.2 Fehleranfälligkeit

Das Ziel der graphischen Oberfläche ist es, diese möglichst überschaubar für den Anwender zu gestalten, damit der Roboterarm fehlerfrei gesteuert werden kann. In dem Tab 'Manuelle Steuerung' wurde mit Hilfe der horizontalen Slider sichergestellt, dass der Benutzer keine ungültigen Parameter für die Gelenkwinkel eingeben kann und in dem Tab 'Inverse Kinematik' werden alle neuen Gelenkwinkel zuerst überprüft, ob sie im erlaubten Wertebereich liegen, bevor eine Bewegung ausgeführt wird. Dadurch ist gewährleistet, dass der Arm keine ungültigen Werte bekommt, aber es ist damit nicht garantiert, dass der Roboterarm eine unerlaubte Stellung annimmt, beziehungsweise der Arm gegen die Platform oder gegen ein anderes Hindernis fährt.

Für das Teach-in wird immer nur die aktuelle Position des SCHUNK-Arms gespeichert. Die einzige Schwachstelle ist die Tabelle, weil die Werte editierbar sind und das bedeutet, dass der Anwender die Werte unbrauchbar machen könnte, bzw. die Werte so verändern könnte, dass sie nicht im erlaubten Wertebereich liegen. Da aber auch die Hardware eine interne Parameterüberprüfung macht, wird die Position einfach übersprungen und die nächste Zeile der Tabelle wird ausgeführt, solange die Winkel im erlaubten Bereich liegen.

Bei einer hoch eingestellten Geschwindigkeit der Module bewegt sich der Roboterarm sehr ruckartig, was die Verschiebung der gesamten Plattform verursacht und damit auch die Ungenauigkeit der Gelenkpositionen des Arms. Besonders ist die Ausführung von komplexeren Trajektorien fehleranfällig, da sich die Plattform manchmal bis zu 10cm von der Anfangsstellung verschiebt und damit die Genauigkeit der Position des Endeffektors sehr beeinträchtigt. Darum wird empfohlen, die Geschwindigkeit der Module möglichst im unteren bis mittleren Bereich zu halten.

7 Zusammenfassung und Ausblick

7.1 Offene Probleme

Der Roboterarm ist in der Lage, die aktuelle Position der Gelenke mit einer Genauigkeit von bis zu 0.5° zu speichern und beim Wiedereinschalten des Roboterarms die Position halten. Das Problem, was dabei auftritt ist, dass beim Neustarten der Steuerungssoftware die Werte gerundet werden und es so oft zu einer Verschiebung der Position der Module von jeweils bis zu 1° kommt. Es ist noch zu untersuchen, wie das Problem behoben werden kann.

Bei der inversen Kinematik funktioniert momentan nur die Berechnung der Gelenkwinkel für die nächste Position des Endeffektors, wenn diese maximal 30cm von der aktuellen Position entfernt ist. Für eine größere Entfernung der beiden Positionen müsste man die Entfernung in kleinere Abschnitte unterteilen und dabei sollte sich der Arm fließend von der Position A nach B bewegen können. Momentan hält der Arm bei dem Teach-in nach jeder Position komplett an, die Gelenkmotoren bekommen keinen Strom und die Magnetbremsen greifen. Bei komplexeren Trajektorien ist es aber erwünscht, dass der Arm sich fließend und ohne anzuhalten bewegt, d.h. dass die Gelenkmotoren die ganze Zeit über Strom bekommen und keine unnötigen Pausen durch die magnetische Bremsenverriegelung und -entriegelung entstehen. Weiterhin könnte man die Bewegungen bei der Bahnplanung auch auf eine Kreisbahn bringen, um die Bewegungen des Armes natürlicher aussehen zu lassen.

7.2 Ideen zur Verbesserung des Systems

Das Steuerungssystem ist nur der erste Schritt zu einer flexiblen Robotersteuerung. Der Assistenzroboter soll in Zukunft in einer Wohnung fahren und mit Menschen interagieren können. Der Roboterarm kann zwar jetzt schon komplexe Trajektorien nachfahren, aber es fehlt eine richtige Bahnplanung. Um zum Beispiel ein Tablett halten zu können, ist es notwendig, dass der Arm alle Module gleichzeitig aber mit verschiedenen Geschwindigkeiten bewegt. Momentan fahren die Gelenke im $MOTION\_FRAMP\_MODE$, d.h. die Gelenke bewegen sich mit der eingestellten Beschleunigung und Geschwindigkeit. Dabei kommen die Module zu unterschiedlichen Zeitpunkten bei der Endposition an. Die Bahnplanung fordert

aber, dass die Module gleichzeitig ankommen, so könnte man ein Glas Wasser bewegen, ohne das Wasser zu verschütten.

Abbildung 7.1: der Greifer

Der Greifer (Abb. 7.1) erkennt noch nicht, wie stark er den Gegenstand drückt, dabei hat er 200N Kraft und könnte ein Glas zerdrücken und damit die Leute in der Umgebung verletzen; es fehlen dem Greifer Drucksensoren. Als Beispiel würden sich resistive Folien-Drucksensoren anbieten. Diese Sensoren erzeugen eine große Widerstandsänderung und lassen sich unkompliziert in einer Spannungsteilerschaltung anschließen. Diese Sensoren basieren auf polymeren Halbleiterschichten, deren Übergangswiderstand druckabhängig ist.

Zurzeit hat der Roboterarm einen festen Greifer, der in seiner Funktion nur eingeschränkt arbeiten kann, da der Durchmesser zwischen den Greiffingern nur 68mm beträgt und daher nur kleine Objekte gegriffen werden können. Außerdem sind die Greiffinger parallel zueinander, was die Greiffläche enorm reduziert. Darum wäre es wünschenswert, neue Greifer in verschiedener Form herzustellen, z.B. mit einem Öffnungswinkel oder mit mehreren Fingern. Ideal wäre eine 3-Finger-Greifhand, die präzises, kraftgeregeltes Greifen von schwierig zu greifenden Objekten ermöglicht. (SCHUNK GmbH, 2010) Eine weitere Verbesserungsmöglichkeit wäre, den Greifer einfach austauschbar zu machen, damit bei Bedarf auf ein anderes System gewechselt werde kann.

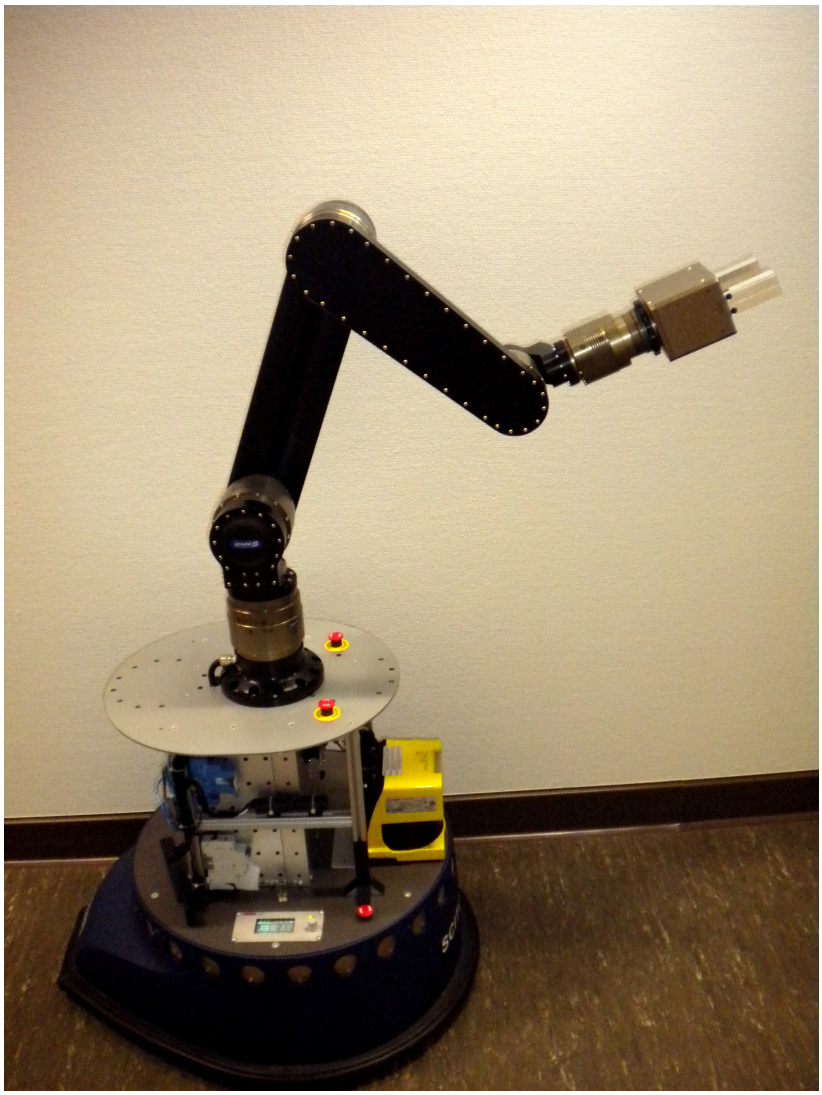


Abbildung 7.2: Scitos G5

Der Roboterarm ist gegenwärtig noch ”blind”, d.h. er ist alleine nicht in der Lage, Gegenstände zu erkennen. Es sollte daher später eine Kamera auf den Greifer montiert werden, die mit einer Software versehen ist, um Gegenstände zu erkennen und Kollisionen mit Hindernissen und Gegenständen im Raum zu vermeiden. Nicht jeder Gegenstand ist einfach zu greifen, darum muss außer der Gegenstandserkennung auch noch eine Entscheidung getroffen werden, wie man das Objekt am besten greifen soll.

Es wird gerade auch an einer Sprachsteuerung gearbeitet, die den gesamten Roboter steuern soll. Für die Interaktion mit Menschen in einem Wohnraum soll der MetraLabs Roboter auf die Sprachkommandos hören und entsprechend reagieren. Dabei soll er nur dann aktiv werden, wenn der Roboter angesprochen wird und damit das so genannten Cocktailparty-Problem vermeiden. Das Cocktailparty-Problem ist die Extraktion von relevanten Informationen aus einer Menge von Umgebungsreizen(Signalen). Für einen normalen Menschen mit gutem Gehör stellt die Unterhaltung auf einer Cocktailparty keine Problem dar, das Gehirn kann die Störgeräusche eliminieren und die wichtigen Informationen herausfiltern. Für einen Roboter hingegen ist das eine richtige Herausforderung, er muss in der Lage sein, die Sprachkommandos, die an ihn gerichtet sind, von einer normalen Unterhaltung zwischen mehreren Personen unterscheiden zu können. (Ziehe und Müller, 2002) Die Fahrplattform des Scitos G5 ist zwar schon mit vielen Sensoren für die Raumorientierung ausgestattet, aber es fehlt noch eine Software für das Mapping des Raumes, um feste Gegenstände, wie Wände und Türen von mobilen Gegenständen, wie beispielsweise Möbelstücke und Personen, unterscheiden zu können. Auch die gesamte Fahrweise und Orientierung im Wohnraum sollte noch entwickelt werden.

Literaturverzeichnis

[Bertsch 2006] BERTSCH, Patrick: *Autonome Fahrzeuge*. 2006. – URL http://www.informatik.uni-ulm.de/ki/Edu/Proseminare/KI/SS06/Ausarbeitungen/07-Bertsch.pdf. – Zuletzt abgerufen am: 26.11.2010

[Brünner 2003] BRÜNNER, Arndt: *Das Newton-Algorithmus zur Approximation von Nullstellen*. 2003. – URL http://www.arndt-bruenner.de/mathe/java/newton.htm. – Zuletzt abgerufen am: 16.11.2010

[Deutsche Enzyklopädie 2005a] DEUTSCHE ENZYKLOPÄDIE: *Inverse Kinematik*. 2005. – URL http://www.calsky.com/lexikon/de/txt/i/in/inverse_kinematik.php. – Zuletzt abgerufen am: 16.11.2010

[Deutsche Enzyklopädie 2005b] DEUTSCHE ENZYKLOPÄDIE: *Jacobi-Matrix*. 2005. – URL http://www.calsky.com/lexikon/de/txt/j/ja/jacobi_matrix.php. – Zuletzt abgerufen am: 17.11.2010

[Heimann u. a. 2007] HEIMANN, Bodo ; GERTH, Wilfried ; POPP, Karl: *Mechatronik*. Carl Hanser Verlag, 2007. – ISBN 3-446-40599-2

[Hesse 1998] HESSE, Stefan: *Industrieroboterpraxis*. Hanser Fachbuchverlag, 1998. – ISBN 978-3-528-06887-5

[Jackel u. a. 2006] JACKEL, Dietmar ; NEUNREITHER, Stephan ; WAGNER, Friedrich: *Methoden der Computeranimation*. Springer, 2006. – ISBN 13978-3-540-26114-8

[Keil 2010] KEIL, Lars-Broder: Japan setzt erstmals Roboter als Lehrer ein. In: *Berliner Morgenpost* (2010), März, Nr. 6.03, S. 32–35. – URL http://www.morgenpost.de/web-wissen/article1049313/Japan_setzt_erstmals_Roboter_als_Lehrer_ein.html

[Korf 2009] KORF, Franz: *Folien zur Vorlesung System- und Echtzeitprogrammierung (Teil 1)*. 2009

[Kraiss 2005] KRAISS, Karl-Friedrich: *Bildgestütztes Teach-In eines mobilen Manipulators in einer virtuellen Umgebung*. 2005. – URL http://darwin.bth.rwth-aachen.de/opus3/volltexte/2005/1017/. – Zuletzt abgerufen am: 17.11.2010

[Malek 2010] MALEK, Massoud: *Householder Algorithm*. 2010. – URL http://www.mcs.csueastbay.edu/~malek/Class/Housholder.pdf. – Zuletzt abgerufen am: 30.11.2010

[Philippsen 1998] PHILIPPSEN, Nils: *Virtuelle Realität*. 1998. – URL http://www.it.hs-esslingen.de/~schmidt/vorlesungen/vr/seminar/ws9899/kinmod-invkinem.html. – Zuletzt abgerufen am: 16.11.2010

[Prof.Dr.-Ing. A. Jahr 2001] PROF.DR.-ING. A. JAHR: *Die Hartenberg-Denavit-Notation*. 2001. – URL http://tww.fh-duesseldorf.de/jahr/Mechanik-Vorlesung/Hartenberg-Denavit-Notation.pdf

[Rahimi und Vogt 2009] RAHIMI, Mohammadali ; VOGT, Matthias: *Vorbereitende Arbeiten zum Living Place, Hamburg*. 2009. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2009-proj/rahimi-vogt.pdf

[Schmiedecke 2010] SCHMIEDECKE, Christoph: *Entwicklung einer allgemeinen dynamischen inversen Kinematik*. 2010. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master09-10-aw1/schmiedecke/bericht.pdf

[SCHUNK GmbH 2010] SCHUNK GMBH: *Greifhand*. 2010. – URL http://www.de.schunk.com/schunk/schunk_websites/products/products_level_3/product_level3.html?product_level_3=7261&product_level_2=250&product_level_1=244&country=DEU&lngCode=DE&lngCode2=DE. – Zuletzt abgerufen am: 17.11.2010

[Siciliano und Khatib 2008] SICILIANO, Bruno ; KHATIB, Oussama: *Handbook of Robotics*. Springer Verlag, 2008. – ISBN 978-3-540-23957-4

[Weiß 2002] WEISS, Claudia: *Inverse Kinematik*. 2002. – URL http://animalrace.uni-ulm.de/lehre/courses/ss02/Computergrafik/ClaudiaWeiss.pdf

[World of Robotics 2010] WORLD OF ROBOTICS: *Robotikindustrie ist zurück auf dem Wachstumspfad*. 2010. – URL http://www.worldrobotics.org/downloads/2010-06-09_PI_IFR_Automatica_deutsch.pdf. – Zuletzt abgerufen am: 04.10.2010

[Ziehe und Müller 2002] ZIEHE, Andreas ; MÜLLER, Klaus-Robert: *Das Cocktailparty-Problem: Neue Verfahren zur Signalquellentrennung*. 2002. – URL http://ddi.cs.uni-potsdam.de/HyFISCH/Spitzenforschung/Mueller.htm. – Zuletzt abgerufen am: 17.11.2010

Index

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 2. Dezember 2010	
Ort, Datum	Unterschrift