Industrieroboter - Vorwärtskoordinatentransformation

Durch welche mathematischen Methoden lässt sich der Werkzeugmittelpunkt eines Industrieroboters aus dessen Gelenkswerten berechnen?

Vorwissenschaftliche Arbeit

Schuljahr 2014/15

vorgelegt von

Alexander Prutsch

eingereicht am BG/BRG Leibnitz

bei Mag. Christa Strohmeier

Abgabedatum: 27.02.2015

Abstract

In dieser vorwissenschaftlichen Arbeit wird eine mathematische Problemstellung bei Industrierobotern behandelt: Bei der Vorwärtskoordinatentransformation, auch direkte Kinematik genannt, soll die Position und die Orientierung des Werkzeugmittelpunktes (= TCP) aus den Gelenkswerten eines Industrieroboters errechnet werden. Es wird in dieser VWA die für die direkte Kinematik angewandte Denavit-Hartenberg-Transformation erklärt und die Transformationsmatrizen werden auch hergeleitet. Als Hinleitung zu diesem Thema werden ebenfalls Grundlagen des Themas Industrieroboter, wie die Geschichte, die Anwendung und die Programmierung von Industrierobotern, sowie auch die verschiedenen Typen der Kinematik behandelt. Zusätzlich habe ich zwei Mal das Werk der Firma Magna Steyr in Graz besucht. Dort habe ich durch ein Interview mit dem Leiter der Wartung der Industrieroboter in einem Bereich der Fertigung einen Einblick in die praktische Anwendung von Industrierobotern bekommen können. Außerdem wurden mir auch Daten eines Industrieroboters bereitgestellt, die ich für Beispielberechnungen genutzt habe. Der Großteil der Arbeit beruht auf Literaturrecherche, großteils in Büchern und teilweise auch im Internet. Das Interview bei der Firma Magna, die Beispielberechnungen, sowie die Erstellung eines Skripts zur Berechnung des TCP bei Industrierobotern ergeben den produktiven Teil meiner Arbeit.

Vorwort

Ich möchte mich an dieser Stelle sehr herzlich bei meinen beiden Ansprechpartnern bei der Firma Magna Steyr, Herrn Wolfgang Bartl und Herrn Andreas Huber, für ihre Hilfsbereitschaft und Unterstützung beim Verfassen dieser Arbeit, bedanken!

Ganz besonders bedanken möchte ich mich bei Frau Professor Mag. Christa Strohmeier für die Betreuung dieser Arbeit!

Lichendorf, am 20. Februar 2015 Alexander Prutsch

Inhaltsverzeichnis

1 Einleitung

Ich beschäftige mich in dieser Arbeit mit einem Thema, welches mich schon seit meiner Kindheit fasziniert: Roboter. Schon als Kind baute und programmierte ich mit diversen Baukästen Roboter und eignete mir Wissen über diese an. Für mich macht den Reiz an ihnen die Kombination der ausgefeilten Konstruktion mit der durch die Programmierung erschaffenen Künstlichen Intelligenz aus.

Meinen Fokus habe ich speziell auf Industrieroboter gelegt, also Roboter die in Fabriken zur Fertigung eingesetzt werden. Im Zuge dieser Arbeit möchte ich erforschen, durch welche mathematischen Methoden es möglich ist, den Werkzeugmittelpunkt (= TCP) aus den Gelenkswerten eines Industrieroboters zu berechnen. Meine Forschungsfrage habe ich daher folgendermaßen gewählt:

"Durch welche mathematischen Methoden lässt sich der Werkzeugmittelpunkt eines Industrieroboters aus dessen Gelenkswerten berechnen?"

Aufgrund dieser Schwerpunktsetzung auf mathematische Berechnungen lässt sich die Arbeit zum Großteil der Mathematik, aber auch der Physik und der Informatik zurechnen.

Ich musste bei der Wahl dieses Themas ein wenig Mut aufbringen, da ich auf keine Technische Schule gehe und ich somit keine Information zu den fachtechnischen Aspekten meines Themas im Unterricht vermittelt bekommen habe. Auch das Thema Matrizenrechnung war für mich vor dieser Arbeit komplettes Neuland und ich musste mir das Wissen darüber weitgehend im Selbststudium zu Hause aneignen.

Bereits im Vorfeld beschäftigte ich mich im Zuge eines Vorbereitungskurses auf die vorwissenschaftlichen Arbeit mit dem Thema Industrieroboter, ich legte den Fokus bei dieser Arbeit auf die verschiedenen Programmierungsmethoden von Industrierobotern. Dies bot mir eine ausgezeichnete Gelegenheit mich zum ersten Mal konkret mit meinem Thema zu beschäftigen und die Erkenntnisse aus dieser Arbeit habe ich auch in meine Vorwissenschaftliche Arbeit einfließen lassen.

Den Aufbau der Arbeit habe ich so gewählt, dass ich zur Hinleitung zunächst Geschichte, sowie Begriffsherkunft und Definition behandle. Auch möchte ich kurz ansprechen, welche verschiedenen kinematischen Aufbauarten es gibt, bevor ich auf die Berechnung des TCP eingehe. Außerdem habe ich ein Skript für das Computerprogramm MATLAB entwickelt, mit dem sich eine Berechnung des TCP an einem beliebigen Industrieroboter durchführen lässt. Fortführend werde ich auch

noch auf die Anwendungen von Industrierobotern und die verschiedenen Programmierungsmethoden eingehen. Zum Abschluss werde ich von meinen beiden Werksbesuchen bei der Firma Magna Steyr berichten. Bei meinem ersten Besuch habe ich durch einen Werksrundgang einen praktischen Einblick in das Thema Industrieroboter bekommen können und ich habe außerdem auch ein Interview mit einem Mitarbeiter der Firma Magna in Graz durchgeführt. Als ich das zweite Mal gegen Ende des Schreibprozesses das Magna Werk besuchte, konnte ich einen Blick auf die Arbeit eines Industrieroboterprogrammierers werfen und außerdem wurden mir auch reale Gelenkswerte und Daten eines Industrieroboters, welche ich für meine Beispielberechnungen nutzte, bereitgestellt.

Mein Wissen habe ich hauptsächlich aus Bücher bezogen, allerdings nutze ich auch einige Webseiten, hauptsächlich von Universitäten, als Quellen. Meine beiden Werksbesuche, sowie die Erstellung des Skripts zur TCP-Berechnung und die Beispielberechnungen sind produktive Teile an meiner Arbeit.

2 Geschichte, Begriffsherkunft und Definition

Der Begriff Roboter stammt nicht wie viele andere Begriffe aus dem Lateinischen, Griechischen oder auch Englischen, sondern er stammt aus dem Tschechischen. Das Wort "robota" bedeutet im Tschechischen "arbeiten" (vgl. Weber 2007: 14).

Das Wort Roboter als Begriff für Maschinen wurde erstmals vom tschechischen Schriftsteller Karl Čapek verwendet. Der Urheber des Wortes ist allerdings sein Bruder Josef Čapek. Karl Čapek verwendet in seinem Bühnenstück "Rossums Universal Robot" den Namen "Robot" für Maschinen, die schwere Arbeiten verrichten, aber mit der Zeit zu rebellieren beginnen und dann die Menschheit vernichten (vgl. Wikipedia 2013 s. v. *R. U. R*).

Bereits im 15. oder 16. Jahrhundert wurden erste roboterähnliche Maschinen von Leonardo da Vinci gebaut. Um 1700 wurden bereits musikspielende Puppen und mechanische Uhren gebaut, welche ebenfalls als Vorgänger heutiger Roboter angesehen werden (vgl. Stark 2009: 14).

Der Grundstein für Roboter im industriellen Einsatz wurde 1954 von G. C. Devol mit dem Patent No. 2988237 "Programmed Article Transfer" gelegt. Fast zeitgleich reichte der britische Erfinder C. W. Kenward 1954 auch ein Patent für ein zweiarmiges Robotergerät ein. Die ersten Industrieroboter wurden dann 1960 von der amerikanischen Firma "Unimation" hergestellt. Diese hydraulisch angetriebenen Industrieroboter kamen bei General Motors zum Einsatz (vgl. Hesse, Malisa, Almansa, Ambrosch, Graf, Hieger, Trenker, Kubinger, Wagner 2010: 32).

In den 1970er Jahren wurden Mikroprozessoren entwickelt, welche für die Steuerung von Industrierobotern verwendet wurden. Ab dieser Zeit wurden auch Industrieroboter erstmals in höheren Stückzahlen als Roboter in der Automobilindustrie für Handhabungs- und Schweißarbeiten eingesetzt. In den 1980er Jahren erlebte dann die Entwicklung von Mikroprozessoren einen Boom und so wurden die Roboter wesentlich schneller und genauer, womit auch deren Verbreitung in der Industrie stieg. Erst ab den 90er Jahren wurden Industrieroboter auch erstmals in größerem Umfang in anderen Branchen außerhalb der Automobilfertigung eingesetzt. Die jüngsten Entwicklungen seit der Jahrtausendwende sind robotergesteuerte künstliche Glieder, Mikroroboter und kooperative Roboter, die untereinander kommunizieren können (vgl. Stark 2009: 14).

Es gibt verschiedene Definitionen eines Industrieroboters, im Deutschen Sprachraum ist die VDI-Richtlinie 2860 (Verein Deutscher Ingenieure) die gängigste Definition:

> Industrieroboter sind universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolge und Wegen bzw. Winkel frei programmierbar (d. h. ohne mechanischen Eingriff vorzugeben bzw. änderbar) und gegebenenfalls sensorgeführt sind. Sie sind mit Greifern, Werkzeugen oder anderen Fertigungsmitteln ausrüstbar und können Handhabe- oder andere Fertigungsaufgaben ausführen (Weber 2007: 15).

3 Kinematische Typen

Der kinematische Typ beschreibt die geometrische Form des Arbeitsraumes eines Industrieroboters und ist eine Möglichkeit um Industrieroboter in verschiedene Kategorien einzuteilen. Um den Typ der Kinematik bei einem Industrieroboter zu bestimmen werden die ersten drei Achsen, der in der Regel sechs Achsen eines Industrieroboters, betrachtet. Es kann sein, dass der Arbeitsraum eines Industrieroboters durch eine siebente oder achte Achse (zum Beispiel hin und her fahren auf Schienen) erweitert wird. Natürlich gibt es auch Roboter mit weniger als sechs Achsen, wenn eine so große Beweglichkeit nicht gebraucht wird, da diese einfacher zu handhaben und billiger sind (vgl. Kreuzer, Lugtenburg, Meißner, Truckenbrodt 1994: 11).

Diese ersten drei Gelenke eines Industrieroboters mit sechs Achsen sind für die sogenannten Makrobewegungen zuständig, die die Position des Roboterwerkzeugs bestimmen. Die anderen drei Gelenke führen sogenannte Mikrobewegung, die für die richtige Orientierung des Roboterwerkzeugs sorgen, durch. Die unterschiedlichen kinematischen Typen hängen nun davon ab, ob es sich bei den ersten drei Gelenken um Schubgelenke oder Drehgelenke handelt (vgl. ebd.).

Wenn es sich um drei Schubgelenke handelt, besitzt der Roboter eine Kartesische Kinematik und sein Bewegungsraum ist rechteckig. Vorteile dieses Typs sind die preiswerte Anschaffung, die steife und einfache Bauart, sowie das simple Steuerkonzept (vgl. Schmid, Kaufmann, Pflug, Strobel, Baur 2013: 307).

Wenn das erste Gelenk ein Drehgelenk ist und die anderen beiden Schubgelenke, handelt es sich um eine Zylindrische Kinematik, bei der der Bewegungsraum dem Namen entsprechend zylindrisch ist (vgl. ebd.: 308).

Ein Roboter hat eine Sphärische Kinematik (Hohlkugel), wenn die ersten beiden Gelenke Drehgelenke sind und das dritte ein Schubgelenk. Diese Roboter zeichnen

sich ebenfalls besonders durch ihre steife Bauart aus (vgl. Kreuzer, Lugtenburg, Meißner, Truckenbrodt 1994: 12f.).

Der am weitverbreitetste Typ ist die Drehkinematik. Bei einer Drehkinematik sind die ersten Dreigelenke des Roboters Drehgelenke. Dem Roboter ist es möglich nahezu jeden Punkt rund um ihn zu erreichen. Die optimale Orientierungsmöglichkeit des Greifers, vor allem wenn es sich auch bei den anderen drei Gelenken um Drehgelenke handelt, sowie der sehr große Verstellbereich sind die größten Vorteile von Knickarmrobotern (vgl. ebd.).

4 Mathematische Methoden zur Koordinatentransformation

Koordinatentransformationen werden verwendet, um die in einem Koordinatensystem bekannten Koordinaten eines Punkts oder eines Vektors für ein anderes Bezugsystem zu berechnen. Dabei wird mit der Vektor- und Matrizenrechnung gearbeitet. Eine Matrix ist eine rechteckige Anordnung von Zahlen, die sich als Mehrfachvektor interpretieren lässt. So kann man durch eine 3x3 Matrix die Orientierung eines kartesischen Koordinatensystems definieren, wobei jede Spalte den Richtungsvektor einer der drei Achsen angibt. Diese Gegebenheit ist essentiell bei Koordinatentransformationen (vgl. Stark 2009: 50).

4.1 Matrizen

Wie bereits erwähnt, versteht man unter einer Matrix ein rechteckiges Zahlenschema. Eine (m x n) Matrix besteht aus m x n Elementen und hat m Zeilen und n Spalten. Die Elemente werden dabei als a_{ij} bezeichnet. Alle Elemente aus einer Zeile i bilden den i-ten Zeilenvektor von A - alle Elemente aus der Spalte j bilden den j-ten Spaltenvektor von A (vgl. Kirchgessner, Schreck 2013: 55).

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

Abbildung 1

Die wichtigste Rechenoperation bei Koordinatentransformationen ist die Matrizenmultiplikation. Diese ist nicht kommutativ, das heißt die Reihenfolge der

Faktoren darf nicht getauscht werden, da die zweite Matrix bei einer Matrizenmultiplikation "gedreht" wird und somit immer eine Zeile der ersten Matrize mit einer Spalte der zweiten multipliziert wird. Daher ist es auch eine Voraussetzung für eine Matrizenmultiplikation, dass die Anzahl der Spalten der ersten Matrix ident ist mit der Anzahl der Zeilen der zweiten Matrix. Für die Multiplikation einer Matrix A mit der Matrix B gilt beispielsweise (vgl. ebd. 58f.):

$$A * B = C$$

$$A = \begin{pmatrix} 2 & 8 & -9 \\ 3 & 4 & 7 \end{pmatrix} \qquad B = \begin{pmatrix} 4 & 3 \\ 5 & -2 \\ 1 & 6 \end{pmatrix}$$

$$C = \begin{pmatrix} 2 & 8 & -9 \\ 3 & 4 & 7 \end{pmatrix} * \begin{pmatrix} 4 & 3 \\ 5 & -2 \\ 1 & 6 \end{pmatrix} = \begin{pmatrix} 2*4+8*5-9*1 & 2*3-8*2-9*6 \\ 3*4+4*5+7*1 & 3*3-4*2+7*6 \end{pmatrix} = \begin{pmatrix} 39 & -64 \\ 39 & 43 \end{pmatrix}$$

Für die geometrischen Transformationen Rotation, Skalierung, Projektion sowie Translation gibt es spezielle Transformations-Matrizen, mit denen sich die Koordinaten eines Punktes nach einer solchen Operation berechnen lassen. Für eine Koordinatentransformation zur Berechnung des TCP bei einem Industrieroboter wird eine Kombination von Translations- und Rotationsmatrizen gebraucht (vgl. Kreuzer, Lugtenburg, Meißner, Truckenbrodt 1994: 40f.).

4.2 Herleitung Rotationsmatrizen

4.2.1 2D-Rotationsmatrix

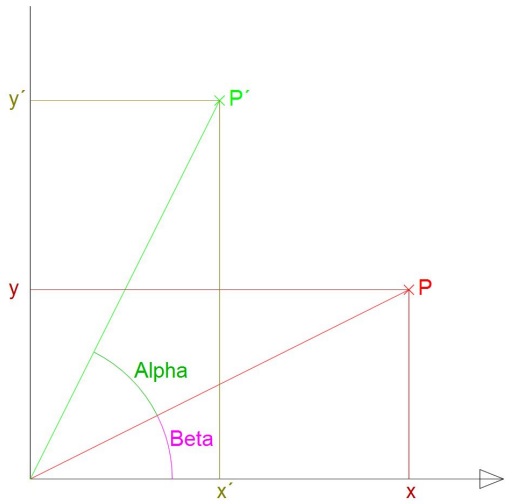


Abbildung 2

Vor der Herleitung der 3D-Rotationsmatrizen, wie sie bei einer Koordinatentransformation im Raum gebraucht werden, will ich zuerst die Herleitung

der 2D-Rotationsmatrix besprechen. Der Punkt P mit den Koordinaten x und y stellt dabei den Ausgangspunkt dar. Dieser Punkt wird dann um den Winkel Alpha α und mit der Drehachse durch den Ursprung so gedreht, dass der Punkt P´, mit den Koordinaten x´ und y´ entsteht (vgl. Wiley Information Services 2014).

Wenn man sich einen Einheitskreis im Ursprung des Koordinatensystems vorstellt, so gilt für x=r*cos(β) und x´=r*cos(α+β), sowie y=r*sin(β) und y´=r*sin(α+β). Durch Anwendung der Additionstheoreme für cos(α+β) und sin(α+β) lassen sich die Koordinaten des Punktes P´ nach einer Drehung um α aus den ursprünglichen Koordinaten berechnen (vgl. ebd.).

$$Additionstheorem:\ cos(\alpha+\beta) = cos\ (\alpha)\cdot cos\ (\beta) - sin\ (\alpha)\cdot sin\ (\beta)$$

$$=> x' = r*cos(\alpha)\cdot cos(\beta) - r*sin(\alpha)\cdot sin(\beta)$$

$$x' = x*cos(\alpha) - y*sin(\alpha) \quad (da \quad r*cos(\beta) = x\ und\ r*sin(\beta) = y)$$

$$Additionstheorem: sin(\alpha+\beta) = sin(\alpha)\cdot cos(\beta) + cos(\alpha)\cdot sin(\beta)$$

$$=> y' = r*sin(\alpha)*cos(\beta) + r*cos(\alpha)*sin\ (\beta)$$

$$y' = x*sin(\alpha) + y*cos(\alpha) \quad (\ da \quad r*cos(\beta) = x\ und\ r*sin(\beta) = y\ )$$

Diese Multiplikationen in Matrizenform ausgedrückt ergibt die 2x2 2D-Rotationsmatrix, mit der man die Koordinaten eines Punktes nach einer Drehung berechnen kann, in dem man den Ausgangspunkt als 2. Faktor an die Matrix multipliziert (vgl. ebd.).

$$P': 2D - Rotationsmatrix \begin{pmatrix} cos\ (\alpha) & -sin\ (\alpha) \\ sin\ (\alpha) & cos\ (\alpha) \end{pmatrix} * P \begin{pmatrix} px \\ py \end{pmatrix}$$

Eine weitere Möglichkeit, diese Rotationsmatrix zu interpretieren ist, dass der 1. Spaltenvektor der Rotationsmatrix die x-Achse des gedrehten Koordinatensystems angibt und der 2. Spaltenvektor die y-Achse. Man muss also den Ausgangspunkt mit den Achsvektoren des gedrehten Koordinatensystems multiplizieren, um den gedrehten Punkt P zu erhalten. Diese Erkenntnis vereinfacht die Herleitung der 3D-Rotationsmatrix sehr (vgl. Kirchgessner, Schreck 2014: 131).

4.2.2 3D-Rotationsmatrix

Eine Drehung im Raum um die x- y- oder z-Achse lässt sich durch eine 3x3 Matrix beschreiben. Man kann für eine Drehung um jede der Achsen eine Rotationsmatrix

aufstellen. Um nach einer Rotation den neuen Punkt P´ zu erhalten, muss wie in 2D der bestehende Punkt P mit der Rotationsmatrix multipliziert werden (vgl. Kirchgessner, Schreck 2014: 129f.).

$$Drehung\ um\ x-Achse: \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,(\alpha) & -sin\,(\alpha) \\ 0 & sin\,(\alpha) & cos\,(\alpha) \end{pmatrix}$$

$$Drehung\ um\ y-Achse: \begin{pmatrix} cos\,(\alpha) & 0 & sin\,(\alpha) \\ 0 & 1 & 0 \\ -sin\,(\alpha) & 0 & cos\,(\alpha) \end{pmatrix}$$

$$Drehung\ um\ z-Achse: \begin{pmatrix} cos\,(\alpha) & -sin\,(\alpha) & 0 \\ sin\,(\alpha) & cos\,(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Diese Matrizen lassen sich, hier am Beispiel der Matrix für eine Drehung um die z-Achse, folgendermaßen herleiten: Wenn man auf die Achsen x und y einen Einheitsvektor legt, haben diese die Koordinaten e_x=(1|0|0) und e_y=(0|1|0). Die Koordinaten der beiden Einheitsvektoren e_x´ und e_y´ nach der Drehung lauten e_x´=(cos(α)|sin(α)|0) und e_y´=(-sin(α)|cos(α)|0). Der Einheitsvektor auf der z-Achse lautet beide Male e_z=e_z´=(0|0|1). Somit sind nun auch die Achsen des gedrehten Koordinatensystems bekannt und dies lässt sich in einer 3x3 Matrix darstellen. Diese Matrix ist zugleich, wie bei der Herleitung der 2D-Rotationsmatrix festgestellt wurde, auch die fertige Rotationsmatrix - in diesem Fall für eine Rotation um die x-Achse im dreidimensionalen Raum. Wenn die Drehung um die x- oder y-Achse erfolgt, verändern sich dementsprechend die Werte der Vektoren, und es bleibt natürlich auch nicht der Vektor der z-Achse konstant, sondern der der x- beziehungsweise y-Achse. Diese Vektoren in einer 3x3 Matrize festgehalten, ergeben dann wiederum die Rotationsmatrix für eine Drehung um die x-Achse oder die y-Achse (vgl. ebd.).

4.3 Herleitung Translationsmatrizen

Wie erwähnt wird bei einer Rotation der Punkt P mit der Rotationsmatrix multipliziert, um den gedrehten Punkt P´ zu erhalten. Bei einer Verschiebung muss zum Punkt P nur der Verschiebevektor v addiert werden, um die Koordinaten des verschobenen Punktes P´ zu erhalten (vgl. Kirchgessner, Schreck 2014: 40).

$$P' = \begin{pmatrix} xp \\ yp \\ zp \end{pmatrix} + \begin{pmatrix} xv \\ yv \\ zv \end{pmatrix}$$

4.4 Homogene Transformationsmatrizen

Die Berechnung einer Translation mit Hilfe der Addition von 3x1 Matrizen hat allerdings den Nachteil, dass mehrere Schritte, vor allem in Kombination mit der Rotation, welche durch Multiplikation von 3x3 Matrizen berechnet wird, kompliziert ist sind, da unterschiedliche Matrizengrößen vorliegen. Um dieses Problem zu lösen, werden homogene Matrizen genutzt: Bei homogenen Matrizen wird der Raum um eine weitere Dimension erweitert. Mit diesen 4x4 Matrizen lassen sich neben Rotationen und Translationen weitere Transformationen, wie Skalierungen, berechnen. Auch die Kombination von mehreren Schritten und mehreren Arten von Transformationen funktioniert problemlos (vgl. Thaden 2001).

Homogene Matrizen können bis zu vier Submatrizen enthalten, wobei für eine Koordinatentransformation bei Industrierobotern eigentlich nur zwei davon wirklich relevant sind. Links oben befindet sich eine 3x3 Matrix, mit deren Hilfe sich eine Rotation oder auch eine Skalierung, eine Scherung oder eine Spiegelung beschreiben lässt und in der letzten Spalte befindet sich eine 3x1 Submatrix mit der sich eine Translation beschreiben lässt. Links befindet sich in der letzten Zeile eine 1x3 Submatrix mit der sich eine Perspektivtransformation beschreiben lässt und das Element a_{44} ist ein Skalierungsfaktor. Da bei einer Koordinatentransformation in der Robotik eigentlich nur mit Rotationen und Translationen gearbeitet wird, lautet die vierte Zeile einer homogenen Matrix in der Robotik meistens (0,0,0,1) (vgl. Suchý 2010).

$$T = \left[\begin{array}{c|c} R_{3\times3} & \vec{p}_{3\times1} \\ \hline \vec{f}_{1\times3} & w_{1\times1} \end{array}\right] = \left[\begin{array}{c|c} \text{Rotations-} & \text{Positions-} \\ \text{matrix} & \text{vektor} \\ \hline \text{Perspektiv-} & \text{Skalierungs-} \\ \text{transformation} & \text{faktor} \end{array}\right]$$

Abbildung 3: Schema einer homogenen Transformationsmatrix

Basierend auf dieser Grundlage lässt sich nun eine Translation im Raum durch eine Multiplikation einer homogenen Matrix (mit dem Verschiebevektor in der 3x1 Submatrix) mit dem Ausgangspunkt berechnen. Man erhält für eine Translation auf einer bestimmten Achse um die Länge l folgende Matrizen (vgl. Thaden 2001):

$$Verschiebung\ auf\ x-Achse: \begin{pmatrix} 1 & 0 & 0 & l \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Verschiebung\ auf\ y-Achse: \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Verschiebung\ auf\ z-Achse: \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Allerdings müssen nun auch die Rotationsmatrizen aus Kapitel 4.2.2 noch durch Einsetzen dieser in die 3x3 Submatrix zu homogenen 4x4 Matrizen erweitert werden. Die Drehwinkel sind dabei immer positiv im Gegenuhrzeigersinn, da mit Rechtskoordinatensystemen gearbeitet wird. Folgende homogene Matrizen gelten für eine Rotation um den Winkel α um die x-, y- oder z-Achse (vgl. Suchý 2010):

$$Drehung\ um\ x-Achse: \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\,(\alpha) & -sin\,(\alpha) & 0 \\ 0 & sin\,(\alpha) & cos\,(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Drehung\ um\ y-Achse: \begin{pmatrix} cos\,(\alpha) & 0 & -sin\,(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ sin\,(\alpha) & 0 & cos\,(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Drehung\ um\ z-Achse: \begin{pmatrix} cos\,(\alpha) & -sin\,(\alpha) & 0 & 0 \\ sin\,(\alpha) & cos\,(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

5 Vorwärtskoordinatentransformation

Die mathematische Berechnung des Tool Center Point (=TCP) aus den Gelenkswerten eines Industrieroboters, also den Drehwinkeln beziehungsweise den Schublängen der einzelnen Gelenke, in Bezug auf ein Basiskoordinatensystem wird Vorwärtskoordinatentransformation oder direkte Kinematik genannt. Das Gegenstück dazu ist die Rückwärtskoordinatentransformation, bei der die Gelenkswerte berechnet werden. Bei Rückwärtstransformationen gibt es oftmals mehrere Lösungen für eine gewisse Stellung des TCP. Bei der Lösung dieses mathematischen Problems der direkten Kinematik werden Transformationsmatrizen angewand. Der Kern liegt dabei in der Denavit-Hartenberg-Transformationsmatrix, mit deren Hilfe es möglich ist, Koordinatensysteme, welche für jedes Gelenk und auch für den TCP aufgestellt werden müssen, in das nächstfolgende überzuführen.

Voraussetzung zur Anwendung der Denavit-Hartenberg-Transformationsmatrix ist allerdings, dass die Koordinatensysteme für die einzelnen Gelenke nach den Regeln der Denavit-Hartenberg-Konvention aufgestellt wurden, da nur dann eine Überführung funktioniert (vgl Weber 2007: 56).

5.1 Denavit-Hartenberg-Konvention

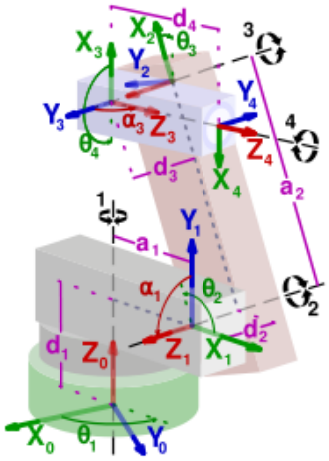


Abbildung 4: Schemenhafte Darstellung eines Roboters mit vier Drehgelenken - Koordinatensysteme nach der DH-Konvention

Für die Berechnung des TCP ist es wie erwähnt zunächst notwendig, dass an jedes Gelenk und auch an den TCP ein Koordinatensystem angelegt wird. Die Regeln, nach denen diese Koordinatensysteme erstellt werden müssen, sind in der Denavit-Hartenberg-Konvention festgehalten. Die Koordinatensysteme werden als K_i für jedes Armteil i (i=0,1,..n) bei einem Roboter mit n Armteilen bezeichnet. Für einen Industrieroboter mit sechs Gelenken müssen folglich 7 Koordinatensysteme aufgestellt werden (vgl. Weber 2007: 46).

Der Ursprung des Basis- oder auch Weltkoordinatensystem K_0 muss sich auf der 1. Gelenksachse befinden, vorzugsweise nahe dem ersten Armteil. Die z-Achse des Koordinatensystems muss entlang der 1. Gelenksachse verlaufen. Die x- und die y-Achse sind unter der Bedingung eines Rechtssystems frei wählbar (vgl. ebd.).

Bei der Festlegung der Koordinatensysteme K_i lässt sich allgemein feststellen, dass der Ursprung von K_i immer auf der Gelenksachse i+1 liegt. Wenn sich die Achsen i und i+1 schneiden, ist der Ursprung in diesem Schnittpunkt festzulegen. Wenn i und i+1 parallel sind, wird zunächst das Koordinatensystem K_{i+1} festgelegt und K_i wird dann auf der Gelenkachse i+1 so festgelegt, dass der Abstand zwischen K_i und K_{i+1}

minimal ist. Wenn allerdings die Achse i und i+1 sich weder schneiden, noch parallel sind, wird der Ursprung von K_i auf den Schnittpunkt einer gemeinsamen Normale von i und i+1 mit der Gelenkachse i+1 gelegt (vgl. ebd.: 46f.).

Die z_i-Achse verläuft immer entlang der Gelenkachse i+1, wobei hierbei zu bedenken ist, dass dies in zwei Richtungen erfolgen kann. Für die Festlegung der x_i-Achse gilt, dass sie, wenn sich die Achsen z_{i-1} und z_i schneiden, parallel zum Kreuzprodukt von z_{i-1} und z_i verläuft, wobei auch hier zwei Möglichkeiten vorhanden sind. Schneiden sie sich nicht, so verläuft die x_i-Achse entlang der gemeinsamen Normalen der Achsen i und i+1. Die Richtung der Achse ist dabei von Achse i zur Gelenksachse i+1. Die y-Achse wird dabei so ergänzt, dass ein Rechtssystem entsteht (vgl. ebd.: 47).

Das Koordinatensystem K_n ist allgemein so zu wählen, dass es möglich ist, dass das Koordinatensystem K_{n-1} durch eine Denavit-Hartenberg-Transformation in das Koordinatensystem K_n übergeführt werden kann. Der Ursprung von K_n liegt idealerweise im TCP. Eine Möglichkeit der Festlegung ist, dass die z_n Achse in Richtung der z_{n-1} Achse durch den TCP gelegt wird, die x_n Achse senkrecht auf die z_{n-1} Achse, so dass sie in Richtung der z_n-Achse zeigt. Liegen z_n und z_{n-1} auf einer Linie dann wird x_n wie x_{n-1} gewählt (vgl. ebd.).

5.2 Denavit-Hartenberg-Transformation

Aufgrund der Festlegung der Koordinatensysteme nach der DH-Konvention ist es nun möglich, ein Koordinatensystem K_i in das Koordinatensystem K_{i+1} mit Hilfe zweier Drehungen, sowie zweier Translationen überzuführen. Für die Überführung werden vier Parameter benötigt, die sogenannten Denavit-Hartenberg-Parameter. Für jeden Industrierobotertyp und für jedes Gelenk eines Roboters müssen drei von diesen zuerst bestimmt werden, die sogenannten konstruktiven Parameter. Der vierte Parameter ist der variable Gelenkswert. Bei Drehgelenken ist dieser der Drehwinkel, bei Schubgelenken die Schublänge und somit ist es der Parameter, der sich bei einer Bewegung des Gelenkes verändert. Abhängig vom Aufbau des Industrieroboters ist es aber teilweise nötig, auch zum Gelenkswert einen fixen Winkel oder eine fixe Schublänge dazuzurechnen, um eine Überführung der Koordinatensysteme zu ermöglichen. Diese Überführung eines Koordinatensystems in ein anderes nach den Regeln der Denavit-Hartenberg-Konvention wird in der Denavit-Hartenberg-Transformationsmatrix beschrieben (vgl. Kreuzer, Lugtenburg, Meißner, Truckenbrodt 1994: 40f.).

5.2.1 Denavit-Hartenberg-Transformationsmatrix

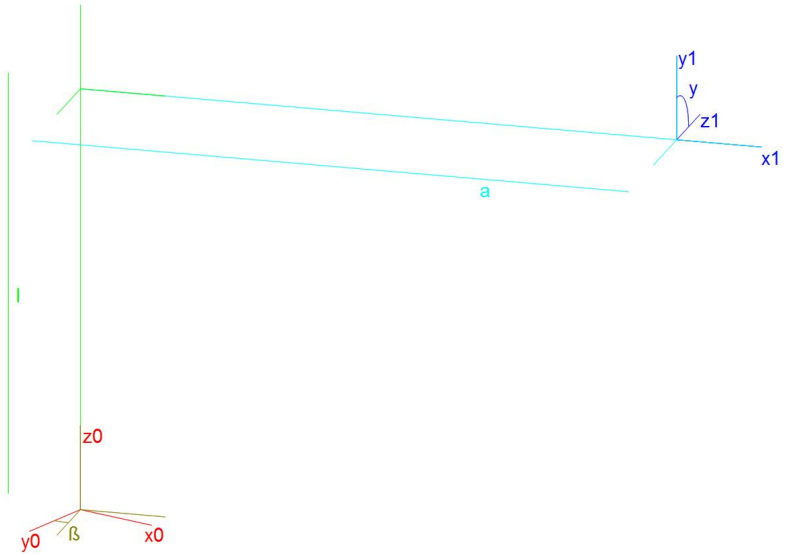


Abbildung 5: Beispiel für Überführung zweier Koordinatensysteme nach der DH-Konvention mit Hilfe zweier Rotationen und Translationen

Wie bereits erwähnt, muss die Denavit-Hartenberg-Transformationsmatrix sowohl zwei Drehungen, eine um die x- und eine um die z-Achse, als auch zwei Translationen beinhalten. Um die Denavit-Hartenberg-Transformationsmatrix zu erhalten, muss man eine Multiplikation der Matrizen, die die einzelnen Teilschritte beschreibt, durchführen. Die Reihenfolge der Matrizen bei dieser Multiplikation muss dabei der Reihenfolge der einzelnen Teilschritte entsprechen, da die Matrizenmultiplikation wie erwähnt nicht kommutativ ist. Bei der Überführung eines Koordinatensystems nach der Denavit-Hartenberg-Konvention in ein anderes lautet diese folgendermaßen (siehe Abbildung 5): Zuerst wird das Koordinatensystem um die z_{i-1}-Achse um den Winkel β_i (bei Drehgelenken der Gelenkswinkel) gedreht. Danach wird es um die Armteillänge l_i (bei Schubgelenken die Schublänge) entlang der z_{i-1}-Achse verschoben, bevor es um die Länge a_i auf der x_i-Achse verschoben wird. Um die Überführung zu vervollständigen, muss abschließend noch eine Drehung um die x_i Achse um den Verwindungswinkel γ_i durchgeführt werden. Die vier Matrizen, die die Teilschritte beschreiben, lauten mitsamt Variablennamen wie auf der Abbildung 5 (vgl. Härtig 2004):

$$Rot(z_{i-1}, \beta_i) = \begin{pmatrix} cos\,(\beta_i) & -sin\,(\beta_i) & 0 & 0 \\ sin\,(\beta_i) & cos\,(\beta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Trans(z_{i-1},l_i)=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&l_i\\0&0&0&1\end{pmatrix}$$

$$Trans(x_i,\ a_i)=\begin{pmatrix}1&0&0&a_i\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$$

$$Rot(x_i,\gamma_i)=\begin{pmatrix}1&0&0&0\\0&cos\,(\gamma_i)&-sin\,(\gamma_i)&0\\0&sin\,(\gamma_i)&cos\,(\gamma_i)&0\\0&0&0&1\end{pmatrix}$$

Die ausmulitplizierte Matrix ${}^{i-1}T_i$, die Denavit-Hartenberg-Transformationsmatrix, mit der sich die Überführung eines Koordinatensystems berechnen lässt, lautet (vgl. ebd.):

$${}^{i-1}T_i = Rot(z_{i-1},\beta_i)*Trans(z_{i-1},l_i)*Trans(x_i,a_i)*Rot(x_i,\gamma_i)$$

$${}^{i-1}T_i=\begin{pmatrix}cos\,(\beta_i)&-sin(\beta_i)*cos\,(\gamma_i)&sin(\beta_i)*sin\,(\gamma_i)&a_i*cos\,(\beta_i)\\sin\,(\beta_i)&cos(\beta_i)*cos\,(\gamma_i)&-cos(\beta_i)*sin\,(\gamma_i)&a_i*sin\,(\beta_i)\\0&sin\,(\gamma_i)&cos\,(\gamma_i)&l_i\\0&0&0&1\end{pmatrix}$$

5.3 Berechnung des TCP mit Hilfe der DH-Transformationsmatrix

Mit Hilfe der Denavit-Hartenberg-Transformationsmatrix lässt sich nun eine Berechnung des TCP durchführen. Eine Denavit-Hartenberg-Transformationsmatrix beschreibt allerdings immer nur eine Überführung. Um eine Berechnung bei einem Industrieroboter, der mehrere Gelenke besitzt, durchzuführen, müssen die Denavit-Hartenberg-Matrizen für jede Überführung, also für jedes Gelenk, miteinander multipliziert werden. Bei einem Roboter mit sechs Gelenke würde das folgendermaßen aussehen (vgl. Weber 2007: 56):

$${}^0T_6={}^0T^1*{}^1T^2*{}^2T^3*{}^3T^4*{}^4T^5*{}^5T^6$$

$$\begin{pmatrix}xx&yx&zx&x\\xy&yy&zy&y\\xz&yz&zz&z\\0&0&0&1\end{pmatrix}$$

Das Ergebnis, eine 4x4 Matrix, ist dabei nach den Regeln der Homogenen Matrizen folgendermaßen zu interpretieren: Die Werte x, y und z geben die Position des TCP an, also den Verschiebevektor vom Ursprungskoordinatensystem in den TCP. Die

anderen Werte geben die Richtungsvektoren des Koordinatensystems im TCP an: xx, xy und xz ergeben den Richtungsvektor der x-Achse, yx, yy und yz ergeben den der y-Achse und zx, zy und zz den Richtungsvektor der z-Achse. Bei allen drei Vektoren handelt es sich um Einheitsvektoren (vgl. Kreuzer, Lugtenburg, Meißner, Truckenbrodt 1994: 44).

5.4 Beispielberechnungen Reis RV 6

In weiterer Folge werde ich ein Beispiel für die Vorwärtskoordinatentransformation durchrechnen und so die Anwendung der Denavit-Hartenberg-Transformation an einem Industrieroboter des Typ Reis RV 6 zeigen.

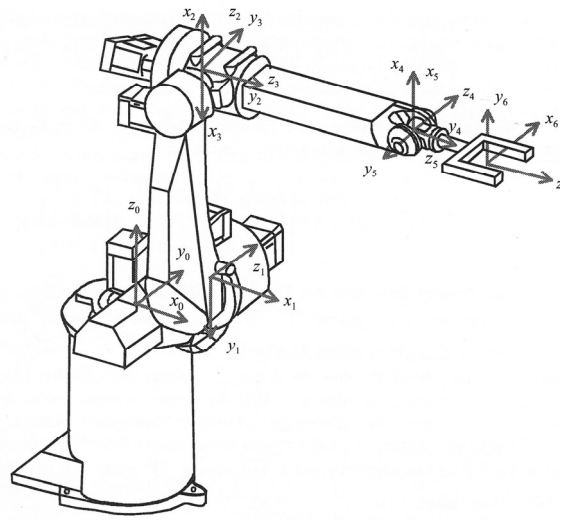
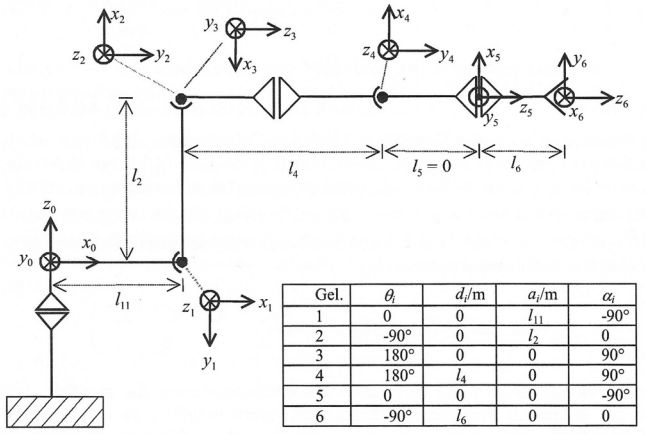


Abbildung 6



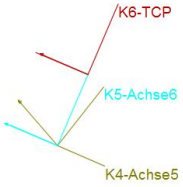
Gel.	θ_i	d_i/m	a_i/m	α_i
1	0	0	l_{11}	-90°
2	-90°	0	l_2	0
3	180°	0	0	90°
4	180°	l_4	0	90°
5	0	0	0	-90°
6	-90°	l_6	0	0

Abbildung 7

In der Abbildung 6 ist eine Skizze des Roboters samt angelegter Koordinatensysteme nach der Denavit-Hartenberg-Konvention zu sehen. Eine schematische Darstellung, sowie die Denavit-Hartenberg-Parameter des Industrieroboters sind in der Abbildung 7 erkennbar (Θ≙β; d≙l; a≙a; α≙γ). Zu den Längen der einzelnen Armteile sind allerdings keine Daten vorhanden und daher habe ich diese auf l11=0,3m; l2=0,85m; l4=0,7m und l6=0,2m geschätzt.

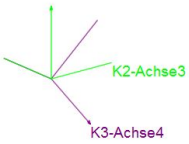
Der TCP des Roboters liegt bei der auf den Abbildungen 6 und 7 dargestellten Position im Punkt TCP1(1.2, 0, 0.85) und mit der Orientierung (Richtungsvektoren des Koordinatensystem) X(0, 1, 0); Y(0, 0, 1); Z(1, 0, 0). Nun wird die erste Achse des Roboters um 90 Grad gedreht, die dritte Achse um -40 und die fünfte Achse um -20 Grad. Die neue Position und die neue Orientierung des TCP2 sollen errechnet werden.

Zunächst muss nun für jede Überführung eines Koordinatensystems K_i in das darauffolgende K_{i+1} eine Denavit-Hartenberg-Transformationsmatrix aufgestellt werden, welche dann mit einander multipliziert werden um die neue Position und Orientierung des TCP zu bekommen:



$$ {}^{0}T_{6} = {}^{0}T^{1} * {}^{1}T^{2} * {}^{2}T^{3} * {}^{3}T^{4} * {}^{4}T^{5} * {}^{5}T^{6} $$

$$ = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -0{,}866 & 0{,}5 & 0{,}9362 \\ 0 & 0{,}5 & 0{,}866 & 1{,}4732 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$



Aus dieser Matrix lässt sich ablesen, dass der TCP nach den Drehungen nun auf dem Punkt TCP2(0, 0.9362, 1.4732) liegt. Außerdem hat das Koordinatensystem des TCP2 folgende Orientierung, festgelegt durch die Vektoren, die die Achsen des Koordinatensystems bilden: x-Achse (-1, 0, 0), y-Achse (0, -0.866, 0.5) und z-Achse (0, 0.5, 0.866). Dieses Ergebnis stimmt auch mit meinem Ergebnis aus einem CAD Programm überein.

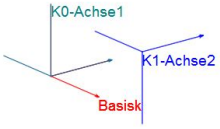


Abbildung 8

5.5 Beispielberechnungen KUKA KR210 2700 Prime

Mir wurden von der Firma Magna bei meinem zweiten Besuch im Werk in Graz die Maße des KUKA KR210 2700 Prime Industrieroboters zur Verfügung gestellt (Abbildung 9). Dieser ist aktuell der modernste Industrieroboter, der in der Firma eingesetzt wird und er repräsentiert den aktuellen Stand der Technik im Bereich Industrieroboter. Der Reis RV 6, der für die vorige Beispielberechnung genutzt wurde, ist dagegen schon rund 15 Jahre alt. Außerdem wurden mir auch die Achswerte bei einigen zufälligen Stellungen des Industrieroboters zur Verfügung gestellt (siehe Tabelle 1), sowie die Position und die Orientierung des TCP bei diesen Stellungen (siehe Tabellen 3 und 4). Diese dienten mir bei meinen Berechnungen als Kontrolle. Die Orientierung des TCP ist nicht durch die

Einheitsvektoren angegeben, sondern durch die vor allem in der Fahrzeugtechnik gebrauchten Roll-Nick-Gier-Winkel, mit deren Hilfe sich die Orientierung eines Koordinatensystems im Raum beschreiben lässt. Der Gierwinkel α gibt, wie man sich leicht anhand eines Flugzeug vorstellen kann, den Drehwinkel um die z-Achse an, der Nickwinkel β die Drehung um die γ-Achse und der Rollwinkel die Drehung um die x-Achse. Nach der Reihenfolge der Rotationen werden diese Winkel auch Z-Y-X-Euler Winkel genannt (vgl. Wikipedia 2015 s. v. Roll-Nick-Gier-Winkel).

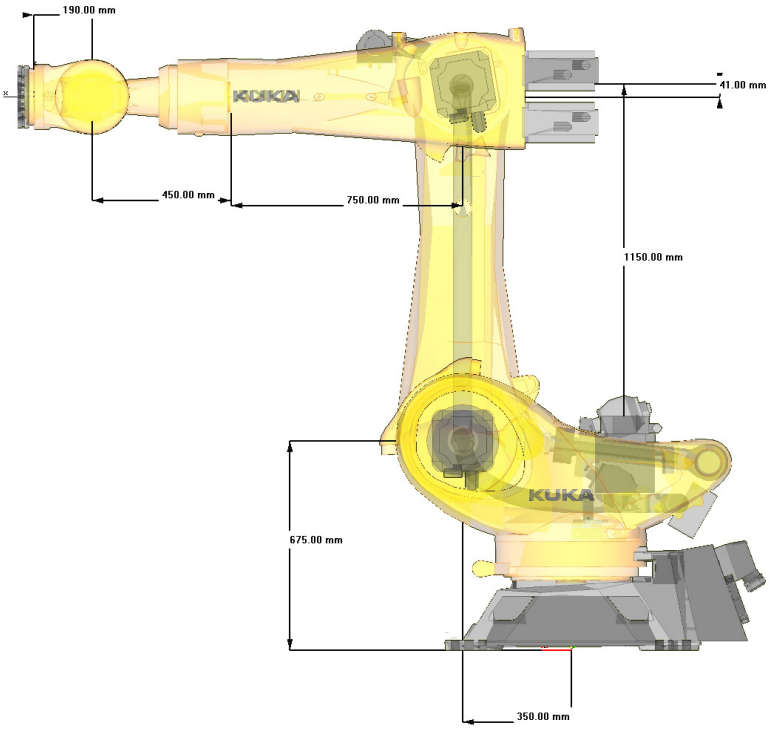


Abbildung 9: Maße des Kuka KR 210 2700 Prime

	J1	J2	J3	J4	J5	J6
via	66,146	-18,726	53,908	-178,344	-110,097	117,526
via1	-29,692	-71,408	130,028	-76,245	-81,748	30,383
via2	45,111	-117,828	-85,299	-263,575	-75,317	66,044
via3	45,111	-85,402	81,136	-267,185	-74,244	79,737

Tabelle 1: Drehwinkel für jedes der sechs Gelenke bei den vier Stellungen

Vor der eigentlichen Berechnung des TCP musste ich zunächst für jedes Gelenk und auch für den TCP Koordinatensysteme aufstellen und dann die Denavit-Hartenberg-Parameter für den KUKA Roboter bestimmen (siehe Abbildung 10). Dies stellte sich schwieriger als angenommen dar: Ich musste einerseits sehr genau auf die Regeln der Denavit-Hartenberg Konvention achten und andererseits auch sehr auf die Drehrichtungen der Verwindungswinkel und der Drehwinkel der Gelenke, ich musste

zunächst erst recherchieren, welche Drehrichtung bei den Gelenken überhaupt als positive Drehrichtung angesehen wird. Auffällig ist außerdem, dass die erste Achse invertiert ist: Ein positiver Drehwinkel bedeutet eine Drehung im Uhrzeigensinn und nicht gegen den Uhrzeigersinn, wie es bei Rechtskoordinatensystemen üblich ist.

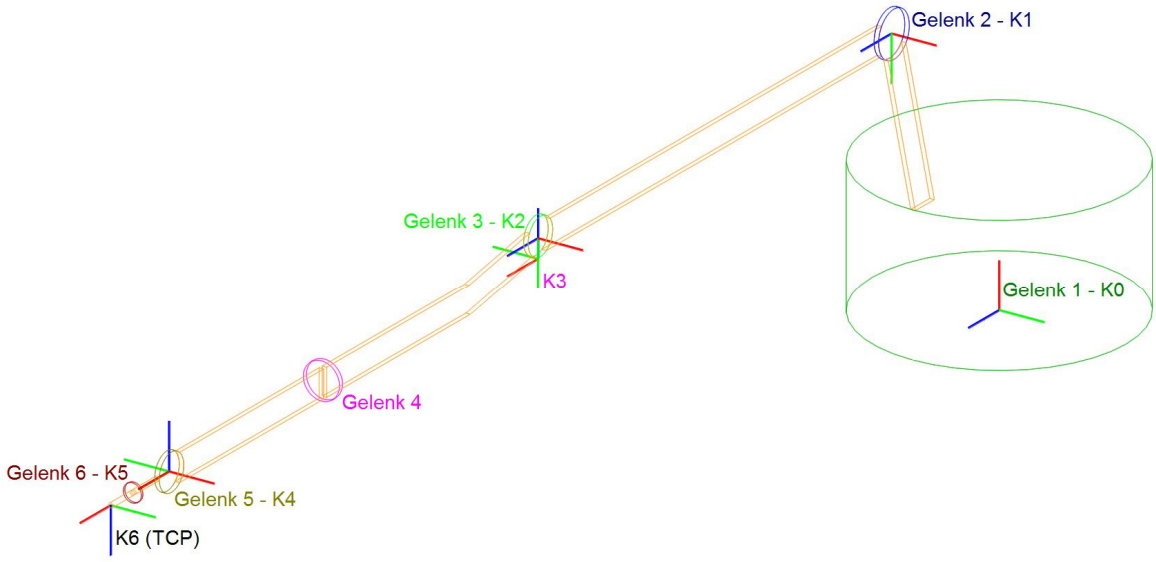


Abbildung 10: Koordinatensysteme nach der DH-Konvention am KUKA KR270
z-Achsen: Rot x-Achsen: Blau y-Achsen: Grün

Gelenk	Beta	l	a	Gamma
1	-D1	675	350	-90
2	D2	0	1150	0
3	D3-90	0	-41	-90
4	-D4	1200	0	90
5	D5	0	0	-90
6	180-D6	190	0	0

Tabelle 2: DH-Parameter für den KUKA KR210 2700 Prime

Zur Berechnung verwendete ich mein selbst entwickeltes MATLAB-Skript um mir Zeit zu ersparen (siehe Kapitel 6). Ich musste jeweils immer nur die konstruktiven Parameter und den Drehwinkel für jedes der sechs Gelenke eingeben. Meine Berechnungsergebnisse für die Position des TCP finden sich in der Tabelle 3. Es ist dabei eine kleine Differenz zu den Vergleichswerten erkennbar, die daher stammt, dass die Winkel nur auf drei Kommastellen angegeben sind und es somit bei Verschiebelängen von über einem Meter zu Abweichungen kommt.

	Gegeben			Berechnet		
TCP	X	Y	Z	X	Y	Z
via	883,352	-2013,865	182,64	901,22	-2050,9	211,12
via1	1306,705	479,506	719,162	1270,9	514,44	719,16
via2	-805,29	1135,303	1258,391	-829,61	1091,6	1258,4
via3	1368,669	-1047,068	1863,214	1324,9	-1071,3	1864,4

Tabelle 3: Gegebene Koordinaten des TCP und Ergebnisse meiner Berechnungen

Bei meinen Berechnungen über das MATLAB-Skript erhielt ich als Ergebnis für die Orientierung des TCP die Einheitsvektoren des Koordinatensystems im TCP und nicht die Roll-Nick-Gier-Winkel mit denen das Koordinatensystem gedreht wurde, wie in den von Magna bereitgestellten Vergleichswerten. Um meine Berechnungen aber trotzdem überprüfen zu können, habe ich aus meinen Einheitsvektoren auch die Winkel berechnet. Dabei musste ich zunächst folgende Formeln anwenden um aus der Homogenen Matrize, die den TCP beschreibt, zunächst die Winkel im Bogenmaß und dann in weiterer Folge auch in Grad zu berechnen (vgl. Wikipedia 2015 s. v. Roll-Nick-Gier-Winkel):

$$\alpha = atan2(a_{21}, a_{11})$$

$$\beta = atan2(-a_{31}, \sqrt{a_{11}^2 + a_{21}^2})$$

$$\gamma = atan2(a_{32}, a_{33})$$

$$f\ddot{u}r\ \beta = +\frac{\pi}{2}:\ \ \alpha = 0 \quad \gamma = atan2(a_{12}, a_{22})$$

$$f\ddot{u}r\ \beta = -\frac{\pi}{2}:\ \ \alpha = 0 \quad \gamma = -atan2(a_{12}, a_{22})$$

Elemente a entnommen aus der homogenen Matrix, die den TCP beschreibt

Wie in folgender Tabelle 4 erkennbar, unterscheiden sich meine errechneten Werte nur marginal von den gegeben Vergleichswerten.

	Gegebene			Berechnet		
TCP	RX	RY	RZ	RX	RY	RZ
via	-127,58	-21,03	41,184	-127,58	-19,861	41,184
via1	44,309	90	0	44,309	90	0
via2	-60,888	90	0	-60,888	89,999	0
via3	-105,999	-84,393	-45,111	-106	-84,403	-45,109

Tabelle 4: Gegebene Werte und berechnete Werte (Orientierung des TCP)

6 Entwicklung eines MATLAB-Skript zur TCP-Berechnung

Als weiteren produktiven Teil meiner VWA habe ich ein Skript entwickelt, mit dem sich die Berechnung des TCP aus den Gelenkswerten an einem beliebigen Industrieroboter am Computer durchführen lässt. Um diese Idee zu realisieren, habe ich mich für das Programm MATLAB entschieden, mit welchem ich bereits im Vorfeld ein wenig vertraut war. Der Quellcode zu diesem MATLAB Skript befindet sich im Anhang.

Mein erster Versuch war so programmiert, dass zunächst über ein Dialogfenster am Beginn die Anzahl der Gelenke des Roboters abgefragt wird (siehe Abbildung 11). In weiterer Folge werden dann, durch eine for-Schleife für jedes Gelenk wiederholt, über ein Dialogfenster die Denavit-Hartenberg-Parameter eines Gelenks abgefragt. Diese werden dann in Denavit-Hartenberg-Transformationmatrizen eingesetzt, welche dann vom Programm multipliziert werden. Am Ende des Skript wird die Position und die Orientierung des TCP in Form einer homogenen Matrize im MATLAB Hauptfenster ausgegeben (siehe Abbildung 12).

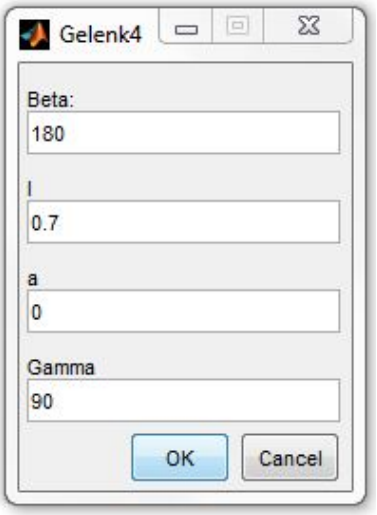


Abbildung 11

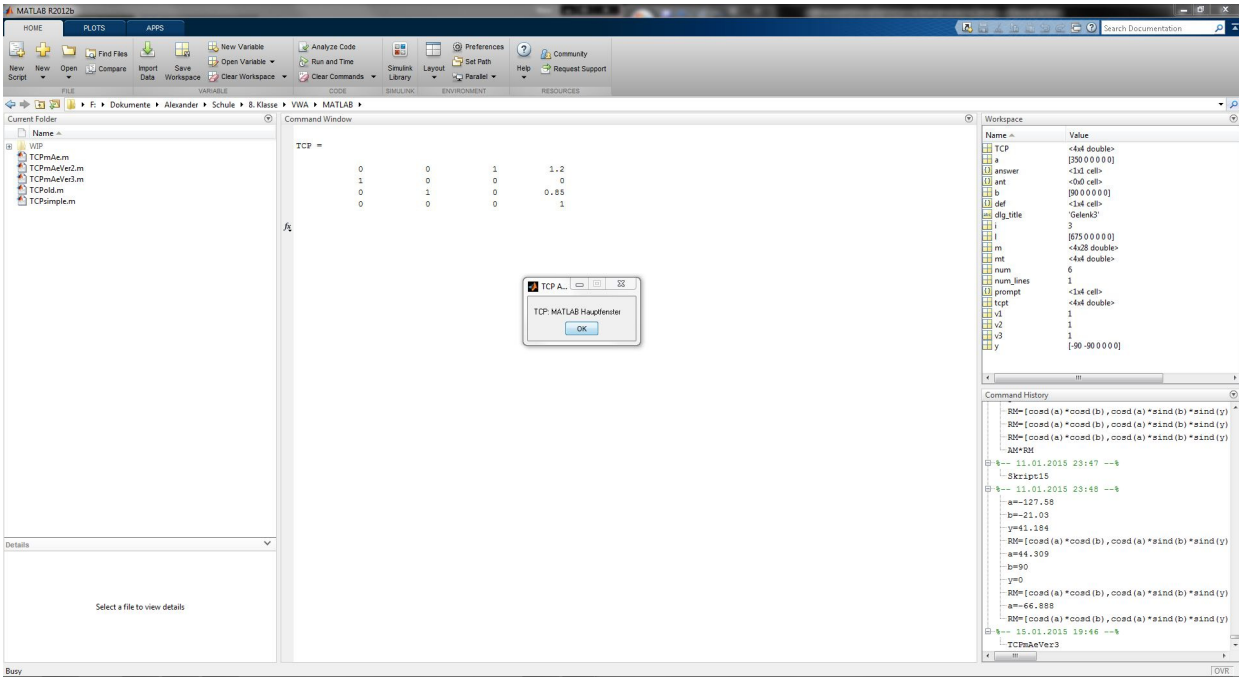


Abbildung 12

Danach wollte ich dieses Skript so erweitern, dass nach der ersten Berechnung des TCP optional weitere Berechnungen am gleichen Industrieroboter (ohne eine erneute Eingabe der konstruktiven Denavit-Hartenberg-Parameter) oder an einem anderen Modell möglich sind. Um diese Idee zu realisieren, musste ich meinen ursprünglichen Code fast zur Gänze verwerfen, da es MATLAB nicht erlaubt, einen Variablennamen zu erstellen, der aus einem Buchstaben und einem variablen Index besteht, so dass ich zum Beispiel den Drehwinkel Beta über eine for-Schleife für das erste Gelenk als Variable b1, für das zweite als b2, definieren hätte können. Ich musste mir daher eine andere Möglichkeit überlegen, wie ich nach einer ersten TCP Berechnung nur auf eine bestimmte Variable zugreifen kann und deren Wert verändern kann, ohne dass die anderen Variablen verändert werden und neu eingegeben werden müssen. Dieses Problem habe ich dadurch gelöst, dass ich für jeden der vier Parameter einen Array erstellt habe, in dem der i-te Wert für den Wert des i-ten Gelenks steht. Dadurch war es mir in weiterer Folge möglich, nur einen Wert auszutauschen, ohne dass die anderen davon betroffen sind und somit bleibt es dem Nutzer bei einer Berechnung am selben Industrieroboter erspart, alle konstruktiven Parameter wieder einzugeben. Außerdem muss der Nutzer des Programms die Gelenkswerte der Gelenke, die sich nicht verändern, auch nicht erneut eingeben. Die Eingabe der neuen Werte habe ich nach einigen Überlegungen so gestaltet, dass man den neuen absoluten Drehwinkel eingibt und nicht den Drehwinkel relativ zum vorher eingegeben Wert, da mir diese Variante als praktischer und logischer erschien.

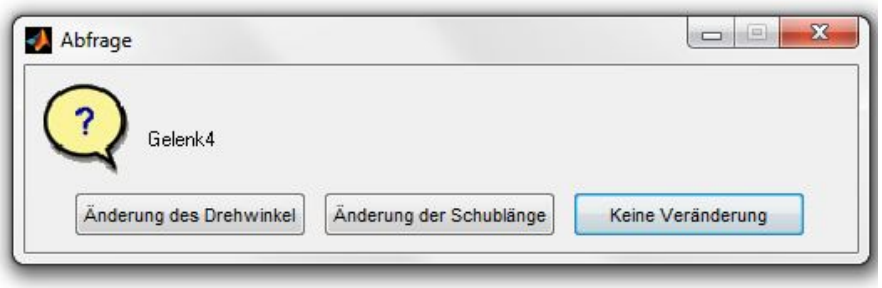


Abbildung 13

Ich habe das Skript am Ende dann um ein Dialogfeld ergänzt über welches abgefragt wird, ob man eine weitere Berechnung am selben Roboter, oder an einem anderen durchführen will, oder ob man das Programm beenden will (siehe Abbildung 13). Allerdings stellte sich mir dann das Problem, dass MATLAB keinen "goto-Befehl" (Sprung zu einer gewissen Zeile des Programms) beherrscht, welchen ich immer als sehr praktisch bei der Programmierung mit GW-Basic empfunden habe. Ich musste daher das Wiederhohlen des Skripts über eine while-Funktion durchführen und das

Ausführen der unterschiedlichen Skriptteile von if-Abfragen abhängig machen. Diese if-Abfragen sind abhängig von Variablen, die je nach gewählter Funktion nach einem Durchlauf des Programms auf 0 oder 1 gesetzt wurden. Je nach dem ob die Variable nun den Wert 0 oder 1 hat wird der Teil des Programms in dieser if-Abfrage ausgeführt oder auch nicht.

Ich habe es zuerst umständlich gefunden mit Arrays zu arbeiten, da ich keine Erfahrung in der Array-Programmierung hatte und ich mich daher neu in das Thema einlesen musste. Die Verwendung von if-Abfragen anstatt dem einfachen Befehl "goto" habe ich zuerst als komplizierter empfunden. Allerdings habe ich bei meinen Recherchen im Internet in Erfahrung gebracht, dass diese Variante der Programmierung wesentlich professioneller ist und das der goto-Befehl nur selten in Programmen gebraucht wird, da es durch ihn schwierig wird einen komplexen Programmcode zu lesen, da man immer die Sprünge beachten muss (vgl. MATLAB Central 2014).

Ich habe meinen Programmcode dann außerdem noch um Kommentare ergänzt, die einem Außenstehenden grob beschreiben, wie der Programmcode funktioniert. Außerdem habe ich auch noch Veränderungen an den Dialogfeldern vorgenommen, die das Skript leicht verständlich und auch praktisch zu bedienen machen. Zusätzlich zu der Textausgabe des TCP habe ich auch noch eine grafische Ausgabe angefügt, bei der die Stellung des Industrieroboters schemenhaft dargestellt wird. Wenn man eine weitere Berechnung durchführt, lässt sich anhand der Grafik zusätzlich auch noch, wie in Abbildung 14 erkennbar, die Veränderung der Stellung des Industrieroboters zur vorigen erkennen.

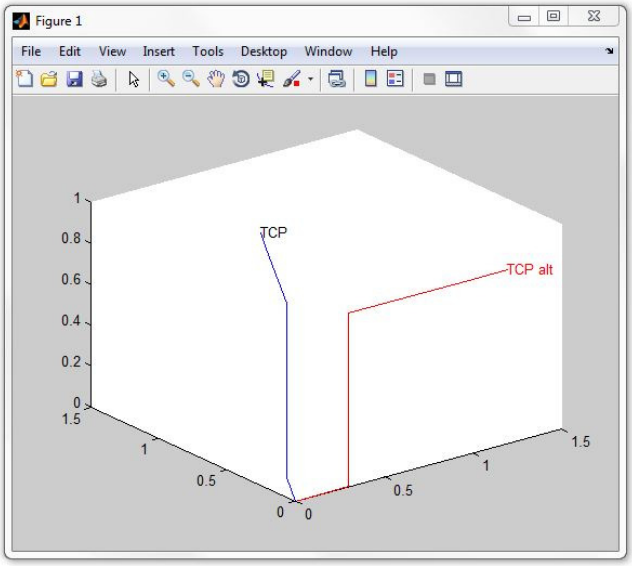


Abbildung 14

7 Anwendungen von Industrierobotern

Man unterscheidet bei Industrierobotern je nach Anwendungsgebiet zwischen Handhabungsrobotern (Manipulationen), Produktionsrobotern (Bearbeiten) und Universalrobotern. Universalroboter können sowohl Handhabungs- als auch Produktionsaufgaben übernehmen. Es werden auch Roboter eingesetzt, die mehrere verschiedene Aufsätze verwenden und diese auch selbstständig während des Arbeitsprozesses wechseln können (vgl. Langmann, Garcia, Große, Haehnel, Hansen, Jacques, Schorn 2010: 466).

Der Aufgabenbereich Manipulation umfasst hauptsächlich das Verpacken, Palettieren und Weitertransportieren von Teilen. Der Aufgabenbereich Bearbeitung ist vielschichtiger und hängt natürlich stark von der Art des Werkstückes ab. Die Aufgaben reichen hierbei von verschiedenen Schweißarbeiten, Kleben und Schrauben, über Fräsen und Schneiden bis hin zur Montage von Baugruppen und der Bestückung von Leiterplatten. Einer der Industriezweige, der Industrieroboter am intensivsten nutzt, ist die Automobilindustrie, da dort eine große Anzahl von einem gleichen Produkt produziert wird. Die Hauptaufgabe der Industrieroboter ist dabei das Schweißen und der Weitertransport von Teilen (vgl. Suchý 2010).

Industrieroboter lassen sich außerdem in drei verschiedene Klassen der Automatisierung einteilen, welche jeweils für unterschiedliche Anwendungsbereiche geeignet sind. In die erste Klasse, der Festen Automatisierung, fallen Roboter, die Werkstücke in sehr großen Serien ohne große Variabilität produzieren können. Diese Roboter werden nur selten bis gar nicht umprogrammiert, wie zum Beispiel bei einer Flaschenfüllmaschine. Wenn ein Roboter für die Produktion in mittleren Serien gedacht ist und eventuell auch eine hohe Variabilität der Produkte vorgesehen ist, zählt er zur zweiten Klasse, der programmierbaren Automatisierung. Roboter, die sich schnell umprogrammieren lassen und sich daher für die Produktion von kleinen Serien mit sehr hoher Variabilität eignen, gehören zur dritten Klasse, der Flexiblen Automatisierung (vgl. ebd.).

Um den idealen Roboter für ein gewissen Anwendungsgebiet zu finden, sind der kinematische Aufbau, die Belastbarkeit des Roboters, die Geschwindigkeit, die Genauigkeit, sowie die Wiederhohlgenauigkeit und die Anschaffungskosten von Bedeutung (vgl. Kreuzer, Lugtenburg, Meißner, Truckenbrodt 1994: 7f.).

8 Programmierung von Industrierobotern

Man unterscheidet bei der Programmierung zwischen Online-Programmierung und Offline-Programmierung. Bei einer Online-Programmierung (oder auch direkte Programmierung) wird die Programmierung direkt am Einsatzort des Industrieroboters vorgenommen. Eine Offline-Programmierung findet am Computer und nicht direkt am Roboter statt. Dabei wird das Programm in, speziell für die Roboterprogrammierung von Industrierobotern ausgelegten, Programmiersprachen entwickelt. Das Programm wird in weiterer Folge meistens auch in einem CAD-Programm getestet (vgl. Weber 2007: 106).

Die Vorteile einer Online-Programmierung liegen vor allem darin, dass keine besonderen Kenntnisse in einer Programmiersprache nötig sind, sondern es reicht, wenn sich der Programmierer gut im Anwendungsbereich des Roboters auskennt. Im Gegensatz dazu setzt eine Offline-Programmierung neben den Kenntnissen im Aufgabenbereich des Industrieroboters natürlich auch gewisse Programmierkenntnisse voraus. Ein Nachteil der Offline-Programmierung ist allerdings, dass Hindernisse in der Arbeitszelle, Abweichungen von Werkstücken und Werkzeugen leicht übersehen werden können. Daher werden in der Praxis die Programme meistens zuerst "offline" entwickelt und dann in einem CAD-Programm und am echten Roboter auf Fehler überprüft und durch Online-Programmier-Methoden nachjustiert (siehe Kapitel 8.3) (vgl. ebd.: 106ff.).

8.1 Online-Programmierung

8.1.1 Teach-In-Programmierung

Beim Teach-In Verfahren wird mit Hilfe eines sogenannten Programmierhandgerät die Position und die Orientierung des TCP von einem Arbeiter durch Bewegen von einzelnen oder mehreren Gelenken in die gewünschte Lage gebracht, die Werte werden dann abgespeichert und der Arbeiter kann dann den nächsten Punkt "teachen". Es ergibt sich dann eine Kette von Punkten, die der Industrieroboter der Reihe nach abfährt (vgl. Schmid, Kaufmann, Pflug, Strobel, Baur 2013: 312f.).

8.1.2 Play-Back-Programmierung

Bei einer Play-Back-Programmierung oder auch Folgeprogrammierung wird der Effektor von einem Arbeiter manuell entlang der Bahn, auf der er sich später bewegen soll, geführt. In vorgegebenen Zeitabständen werden dabei die Gelenkskoordinaten als Soll-Werte für den späteren Betrieb abgespeichert. Diese Programmierart ist nur bei speziell dafür konstruierten Robotern möglich, da viele Roboter zu schwer sind, um diese mit reiner Muskelkraft zu bewegen. Bei Robotern, die nicht in Leichtbauweise konstruiert sind, muss die Steuerung, um eine Play-Back-Programmierung zu ermöglichen, während des Programmierprozesses so eingestellt sein, dass der Antrieb die Bewegungen des Programmierers unterstützt, so dass Gegenkräfte, wie Reibung, Gravitation und Massenträgheit, verringert werden. Das häufigste Einsatzgebiet von Play-Back-programmierten Robotern ist das Lackieren von komplexen Freiformflächen. Das Lackieren von beispielsweise Autoteilen würde sehr komplizierte und schwer programmierbare analytische Bahnen benötigen. Es ist viel einfacher, wenn ein professioneller Lackierer den Lackierroboter in der richtigen Geschwindigkeit auf der richtigen Bahn bewegt und der Roboter dann diese Bahn immer wieder wiederholt (vgl. Weber 2007: 108; 110).

8.1.3 Master-Slave-Programmierung

Die Master-Slave-Programmierung ist ähnlich der Play-Back-Programmierung. Allerdings wird nicht der Roboterarm selbst bewegt, sondern die Programmierung wird an einem sogenannten Bedienarm (=Master Arm) vorgenommen. Der Bedienarm sollte bestenfalls kinematisch sehr ähnlich aufgebaut sein wie der Arbeitsarm (=Slave-Arm), damit die Gelenkskoordinaten des Bedienarms als die des Arbeitsarms verwendet werden können. Wenn der Bedienarm eine andere kinematische Struktur aufweist als der Arbeitsarm, werden dessen Weltkoordinaten registriert und es müssen dann anhand dieser die Gelenkskoordinaten für den Slave-Arm berechnet werden. Zur Erhöhung der Genauigkeit können die Bewegungen des Bedieners auch skaliert werden (vgl. Weber 2007: 108; 110).

Auch in der Telerobotik werden Master-Slave-Systeme verwendet. Die Bewegungen des Masterarms werden ohne Zeitverzögerung sofort vom Slavearm ausgeführt. Dieses Verfahren wird meistens dann angewendet, wenn der Aufenthalt in der Nähe des Arbeitsarm entweder gefährlich oder nur schwer zu bewerkstelligen ist. Die häufigsten Einsatzgebiete dafür sind die Medizintechnik und die Weltraumtechnik. In

der Medizintechnik ist es außerdem auch von Vorteil, dass die Bewegungen skaliert werden: Die Bewegungen des Chirurgen werden präzisiert, was bei besonders heiklen Operation sehr wichtig ist (vgl. ebd.).

8.2 Offline-Programmierung

8.2.1 Textuell

Bei der textuellen Programmierung von Industrierobotern werden entsprechende Bewegungsbefehle von problemorientierten, für die Programmierung von Industrierobotern ausgelegten, Programmiersprachen verwendet, um die Bewegungsabfolge vorzugeben. Es ist zwar prinzipiell möglich, Programme in einem normalen Texteditor zu schreiben, allerdings erkennt dieser nicht Schreib- und Syntaxfehler, welche bei der Programmerstellung passieren können. Um die Programmerstellung zu beschleunigen werden von den Roboterherstellern Programme bereitgestellt, die den Programmierer dabei unterstützen. In diesen textuellen Programmiersystemen ist es meistens möglich, Befehle mittels verschiedenen Befehlsflächen, die man mit der Maus anklickt, in einem Dialog zu generieren. Außerdem ist in diesen Programmiersystemen meistens eine automatische Prüfung eingebaut, die eine Überprüfung auf Syntax und Semantikfehler durchführt (vgl. Weber 2007: 112).

Die Programme werden eigentlich immer in ein Haupt- und mehrere Unterprogramme gegliedert: Im Hauptprogramm werden alle allgemein gültigen Funktionen definiert, wie die maximale Geschwindigkeit, die maximale Beschleunigung, das Bezugskoordinatensystem oder die Startposition. Die einzelnen Arbeitsaufgaben mit den dazugehörigen Positionsdaten werden in den Unterprogrammen beschrieben (vgl. Schmid, Kaufmann, Pflug, Strobel, Baur 2013: 313).

8.2.2 CAD-basierte Programmierung

CAD-basierte Programmierungen sollen vor allem den Aufwand an Nachjustierungen, die am Roboter vorgenommen werden müssen, weil der Programmierer entweder gewisse Hindernisse in der Arbeitszelle oder auch andere geometrische Gegebenheiten nicht vollständig berücksichtigt hat, verringern. In einem CAD-Programm wird die gesamte Arbeitszelle, welche aus dem Roboter

selbst, der Peripherie sowie dem Werkstück besteht, nachgebildet (siehe Abbildung 16). Der Programmierer kann dann die gewünschte Zielstellung in der CAD-Umgebung anfahren und führt somit eine Art virtuelles Teach-In durch. Ein weiteres, sehr häufig genütztes Anwendungsgebiet von solcher Software ist die Fehlerüberprüfung von textuell geschrieben Programmen: Man lässt den virtuellen Roboter das Programm ausführen und es lassen sich schnell Fehler oder Kollisionen erkennen (vgl. Weber 2007: 112f.).

8.3 Programmierung in der Praxis

Bei meinem Besuch bei Magna Steyr konnte ich auch in Erfahrung bringen, wie die Programmierung von Industrierobotern in der Praxis abläuft: Grundsätzlich wird das Programm für den Industrieroboter in einer textuellen Programmiersprache geschrieben, allerdings werden dann in weiterer Folge am Programm meistens noch Abänderungen, etwa bei Kollisionen, mit Hilfe des Teach-In-Verfahrens direkt am Roboter vorgenommen. Der Programmierer verwendet dafür ein Programmierhandgerät mit dem es ihm möglich ist einzelne Punkte des Programmes zu verändern. Es ist auch möglich, mit diesem Handgerät jedes Gelenk des Roboters extra zu bewegen. Vor dem Test am realen Roboter werden die Programme oft auch in CAD-Programmen simuliert: Die gesamte Arbeitszelle samt Roboter und Werksstück werden als CAD-Modell erstellt und es lassen sich so schnell Fehler erkennen, wenn man den virtuellen Roboter das Programm ausführen lässt. Sehr interessant ist auch, dass sich in jeder Arbeitszelle eine Scheibe mit einem Loch befindet, in welches die Werkzeugspitzte des Industrieroboters genau hineinpasst. Mit Hilfe dieses Loches, dessen Position im Raum genau bekannt ist kann der Roboter kalibriert werden (vgl. Interview Anhang).

9 Werksbesuche Magna Steyr

9.1 1. Werksbesuch

Zu Beginn meiner Arbeit besuchte ich das Werk Graz Liebenau der Firma Magna Steyr. Ich konnte dort bei einer Werksführung Schritt für Schritt mit verfolgen, wie der Rohbau der in Graz produzierten Mini Modelle mit Hilfe von rund 180 Industrierobotern (Entspricht einem Automatisierungsgrad von 99%) abläuft. Die Roboter werden großteils für Widerstand- und Bolzenschweißen eingesetzt, aber

auch für den Weitertransport zwischen den einzelnen Stationen und das Auftragen von Kleber. Kurz nach meinem Besuch im Sommer 2014 sollten außerdem auch rund 40 Industrieroboter durch neue ersetzt werden, welche eine höhere Effizienz liefern sollten. Diese Roboter wurden bereits angeliefert und ich konnte in der Roboterwerkstand mitverfolgen, wie diese Roboter zusammengesetzt und in Betrieb genommen wurden (vgl. Interview Anhang).

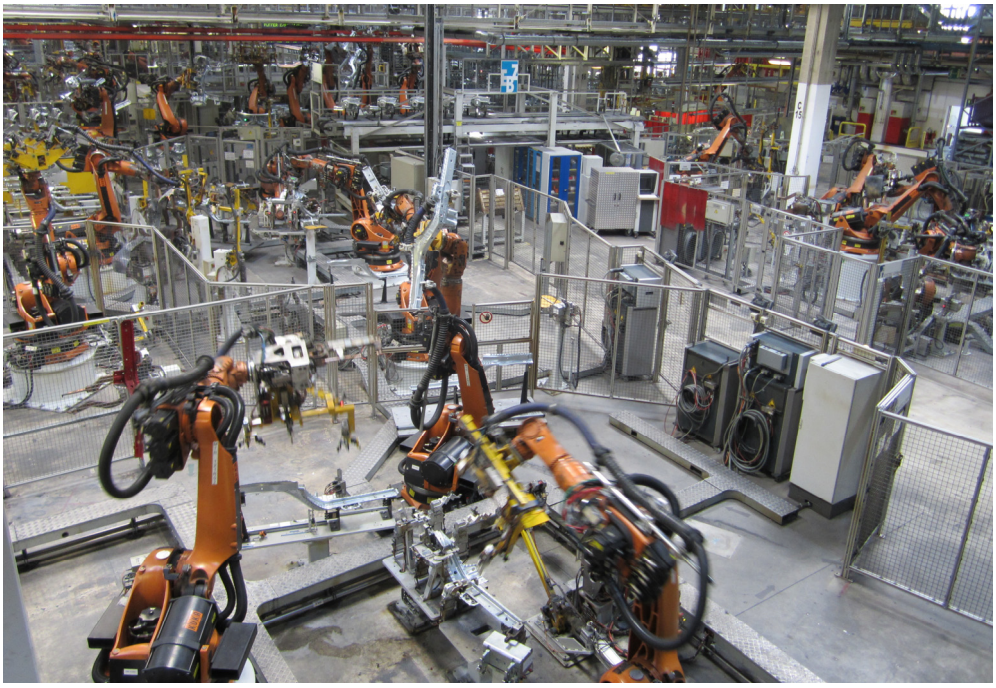


Abbildung 15: Produktionshalle Magna Steyr

Ich habe außerdem in Erfahrung gebracht, dass die direkte Kinematik für Anwender von Industrierobotern eigentlich keine wichtige Rolle spielt, wenn überhaupt nur gering für die Mitarbeiter, die für die Offline-Programmierung zuständig sind. Die Vorwärtskoordinatentransformation spielt allerdings bei der Roboterentwicklung eine große Rolle. Roboterfirmen wie etwa KUKA und Reis entwickeln nämlich Applikationen, in denen dann die Offline-Programmierer die Programme in speziellen Programmiersprachen schreiben. Bei der Programmierung dieser Applikationen ist die direkte Kinematik unumgänglich. Zusammenfassend kann man also sagen, dass die Vorwärtskoordinatentransformation für Roboteranwender keine Rolle spielt, da es ein steuerungsinterner Prozess ist, somit aber für die Roboterentwickler sehr wichtig ist (vgl. Interview Anhang).

9.2 2. Werksbesuch

Im Dezember 2014 besuchte ich noch einmal das Magna Werk in Graz. Zu diesem Zeitpunkt war ich mit dem Schreiben der Arbeit bereits sehr weit fortgeschritten. Ich konnte dort mit einem Offlineprogammierer einige kleine Detailfragen an meiner Arbeit besprechen und außerdem konnte ich mir auch ansehen, wie die Programmierung eines Industrieroboters abläuft: Die gesamte Arbeitszelle wird in ein CAD-Programm visualisiert und die geschriebenen Programme werden dann in dieser CAD-Umgebung simuliert, um etwaige Kollisionen oder andere Probleme zu finden. Außerdem wurden mir die Maße eines KUKA KR210 2700 Prime 6-Achse Industrieroboters zu Verfügung gestellt, welcher den aktuellen Stand der Technik repräsentiert. Neben den Maßen wurden mir auch die Gelenkswerte, sowie die dazu passende Lage und Orientierung des TCP zu vier beliebigen Stellungen des Roboters angegeben. Diese Werte habe ich dann auch für Beispielberechnungen genutzt (Kapitel 5.5). Außerdem wurde mir auch ein kurzes Video zur Verfügung gestellt, in dem die Simulation des Einsatzes eines Industrieroboters in einem CAD-Programm zu sehen ist (siehe Abbildung 16).

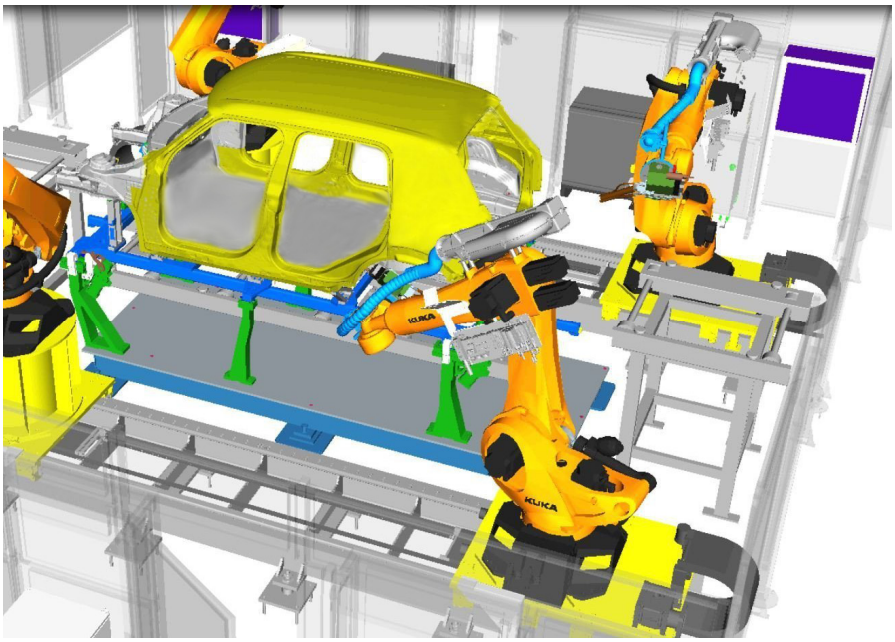


Abbildung 16: Simulation eines Industrieroboters in einem CAD-Programm (Screenshot aus Video)

10 Schluss

Das Schreiben dieser Arbeit hat mir sehr viel Spaß bereitet. Ich muss zugeben, am Anfang des Schreibvorganges war mir etwas mulmig zu Mute, ob ich mich mit meiner Themenwahl nicht doch ein wenig übernommen habe, da ich quasi kein Vorwissen zum Thema Industrieroboter oder Matrizenrechnung hatte. Ich musste mich daher sukzessive durch zahlreiche Quellen arbeiten, um meine Arbeit zu verfassen.

Die Berechnung des TCP aus den Gelenkswerten funktioniert über die Denavit-Hartenberg-Transformation und ich war einerseits überrascht, wie einfach eine Berechnung des TCP durch das Multiplizieren mehrerer Denavit-Hartenberg-Transformationsmatrizen möglich ist. Andererseits ist die Herleitung dieser wiederum recht komplex und sehr spannend, vor allem da sie im Endeffekt doch nur auf simple mathematische Methoden, wie die Sinus- und Cosinus-Funktion, und auch auf der Vektorenaddition, beruht. Bevor eine TCP Berechnung erfolgen kann, muss zunächst an jedes Gelenk ein Koordinatensystem nach der Denavit-Hartenberg-Konvention aufgestellt werden und die drei konstruktiven Denavit-Hartenberg-Parameter müssen für jedes Gelenk eines Industrieroboters ermittelt werden. Durch die Erfahrungen aus meinen Beispielberechnungen kann ich sagen, dass diese Aufgabe sehr viel Konzentration und Genauigkeit erfordert.

Zu Beginn meiner Arbeit war ich der Ansicht, dass es zur Vorwärtskoordinatentransformation eher wenig Literatur gibt. Allerdings wurde ich im Laufe meiner Arbeit immer mehr und mehr vom Gegenteil überzeugt: Ich hatte nie das Problem, dass mir Unterlagen zu einem gewissen Thema fehlten, ich musste mich nur immer neu in die Quelle einlesen, da meistens unterschiedliche Variablen und Schreibweisen verwendet wurden.

Aus meinen beiden Werksbesuchen habe ich mitgenommen, dass Koordinatentransformationen für Firmen, die Industrieroboter anwenden eigentlich keine wirkliche Bedeutung hat, sondern nur für Hersteller von Industrierobotern, womit ich eigentlich nicht gerechnet habe. Eher wird von einem Anwender das Gegenstück zur direkten Kinematik, die Indirekte Kinematik, die Frage, wie sich die Gelenkswerte ändern müssen, um den TCP von einer bestimmten Position in eine andere zu bringen, benötigt. Diese Berechnungen stellen sehr komplexe mathematische Aufgaben dar. Außerdem bin ich auch auf weitere spannende mathematische Aspekte bei Industrierobotern gestoßen, wie Geschwindigkeits- und Dynamikberechnungen. Sehr interessant sind auch die verschiedenen Arten der

Bahnsteuerung, welche wiederum verschiedene Berechnungsmethoden als Grundlage haben.

Mir war es zwar bereits im Vorfeld bewusst, dass Industrieroboter eine sehr wichtige Rolle in manchen Branchen übernehmen und dass nur mehr sehr wenige Aufgaben manuell getätigt werden, allerdings hat mich der Automatisierungsgrad von 99% bei der Fertigung der Rohkarosse eines "Minis" im Magna Werk doch sehr überrascht.

Neben den Werksbesuchen hat mir besonders die Arbeit am Skript zur TCP-Berechnung sehr viel Spaß gemacht, da ich mir dadurch einerseits die Berechnung der Beispiele erleichtert habe und es mir andererseits auch Spaß machte, meine Erkenntnisse in eine praktische Anwendung einfließen zu lassen, auch wenn ich sehr viel Recherche zu MATLAB-Befehlen durchführen musste, um mein Programm zu realisieren. Auch die Beispielberechnungen fand ich sehr spannend, allerdings waren die Berechnungen am KUKA KR270 2700 Prime sehr zeitaufwendig und schwierig, da ich einige Anläufe gebraucht habe, um die Denavit-Hartenberg-Parameter richtig aufzustellen. Es erforderte außerdem große Geduld und Konzentration, die Denavit-Hartenberg-Parameter wieder und wieder in den Computer einzugeben.

Trotz dieser kleinen Schwierigkeiten ziehe ich ein sehr positives Resümee aus dieser Arbeit. Ich habe sehr viele neues Wissen sammeln können und außerdem war auch das Verfassen dieser Arbeit eine sehr spannende Erfahrung, da ich zum ersten Mal mich über einen so langen Zeitraum und in einer solchen Intensität mit einem Thema befasst habe.

11 Literaturverzeichnis

Hesse, Stefan; Malisa, Viktorio; Almansa, Ana; Ambrosch, Roland; Graf, Birgitt; Hieger, Christof; Trenker, Markus; Kubinger, Wilfried; Wagner, Erwin (2010). *Taschenbuch Robotik - Montage - Handhabung*. München: Carl Hanser Verlag.

Kirchgessner, Karsten; Schreck, Marco (2013). *Vektor- und Matrizenrechnung für Dummies.* Weinheim: Wiley-VCH Verlag.

Kreuzer, Edwin; Lugtenburg, Jan-Bernd; Meißner, Hans-Georg; Truckenbrodt, Andreas (1994). *Industrieroboter: Technik, Berechnung und anwendungsorientierte Auslegung*. Heidelberg: Springer Verlag.

Langmann, Reinhard; Garcia, Carlos Alberto Reyes; Große, Norbert; Haehnel Hartmut; Hansen, Ralph; Jacques, Harald; Schorn, Wolfgang (2010). *Taschenbuch der Automatisierungstechnik*. München: Carl Hanser Verlag.

Schmid, Dietmar; Kaufmann, Hans; Pflug, Alexander; Strobel, Peter; Jürgen, Baur (2013). *Automatisierungstechnik mit Informatik und Telekommunikation: Grundlagen, Komponenten und Systeme*. Haan: Europa Lehrmittel.

Stark, Georg (2009). *Robotik mit MATLAB*. Leipzig: Carl Hanser Fachbuchverlag.

Weber, Wolfgang (2007). *Industrieroboter: Methoden zur Steuerung und Regelung*. München: Carl Hanser Verlag.

12 Internetverzeichnis

Härtig, Janko (2004). "Praktikumsarbeit: Roboterkinematiken." [Online im Internet]. http://www.easy-rob.com/uploads/media/bps-inverse-kinematik-jhaertig.pdf. [2014-10-05].

MATLAB Central (2013). " Is there command to go line I want?.'goto'." [Online im Internet]. http://www.mathworks.com/matlabcentral/answers/123694-is-there-command-to-go-line-i-want-goto. [2015-01-02].

Suchý, Jozef (2010). "Vorlesung Grundlagen der Robotik." [Online im Internet]. https://www-user.tu-chemnitz.de/~krdav/uploads/Robotik/GrdlRobotik100726.pdf. [2014-08-06].

Thaden, Uwe (2001). "Darstellung der Translation mit homogenen Koordinaten." [Online im Internet]. http://olli.informatik.uni-oldenburg.de/Grafiti3/grafiti/flow7/page4.html#Ref_ID120. [2014-08-06].

Wikipedia (2013). „R.U.R..“ [Online im Internet]. http://de.wikipedia.org/wiki/R.U.R.. [2014-11-04].

Wikipedia (2015). „Roll-Nick-Gier-Winkel.“ [Online im Internet]. http://de.wikipedia.org/wiki/Roll-Nick-Gier-Winkel. [2015-01-21].

Wiley Information Services, Chemgapedia (2014). "Drehung von Vektoren in 2D- und 3D-Raum." [Online im Internet]. http://www.chemgapedia.de/vsengine/vlu/vsc/de/ma/1/mc/ma_11/ma_11_03/ma_11_03_02.vlu.html. [2014-12-05].

13 Abbildungs- und Tabellenverzeichnis

14 Anhang

14.1 Interview Magna

Alexander Prutsch (AP): Welche Aufgabenbereiche umfasst ihre Tätigkeit bei der Firma Magna?

Wolfang Bartl (WB): Ich bin für die Instandhaltung im Bereich "Rohbau Mini" zuständig. Rohbau heißt, die Rohkarosserie wird hier gefertigt. Wir machen zwei verschiedene Modelle: den Mini Paceman und den Mini Countryman mit verschiedenen Ausprägungen (Volldach, Panoramadach). Der Bereich Instandhaltung im Rohbau umfasst die Hauptbereiche Instandhaltung Elektrik, Instandhaltung der Anlagen und der Mechanik und umfasst die Instandhaltung der Schweißzangen. Insgesamt hat die Instandhaltung aktuell 66 Mitarbeiter, der Automatisierungsgrad beträgt beim Rohbau Mini 99%, und wir arbeiten in einem Dreischichtbetrieb. Wir produzieren aktuell 540 Autos am Tag. Schichtbeginn ist am Sonntag um 21 Uhr und die Schicht endet am Freitag um 23 Uhr. Am Wochenende findet dann Instandhaltung und Wartung statt. Instandhaltung heißt natürlich nicht nur Instandhaltung der bestehenden Anlage, sondern auch Mitwirken bei Planungstätigkeiten, kleine Umbauten selber durchführen, was eine der Hauptaufgaben im Instandhaltungsbereich ist. Beim Thema Instandhaltung unterscheiden wir zwischen Leuten, die Spezialisten sind die den Roboter reparieren und Leuten, die den Roboter nur bedienen. Beim Reparieren unterscheiden wir wieder zwischen elektrischem Bereich, mechanischen Bereich und Werkzeug: Punktschweißzange, Bolzenschweißgerät oder auch ein Klebeequipment.

AP: Zu den Werkzeugen: Für welche Aufgaben, neben Schweißen, werden hier in der Firma Roboter eingesetzt?

WB: Die vorwiegenden Aufgaben sind das Widerstandsschweißen, Bolzenschweißen (kleine Verkupferte Stahlbolzen, die auf das Blech aufgeschweißt werden und diese dienen dazu, einen Teppich oder einen Kabelbaum zu befestigen), Kleben (Es werden doch einige Meter Klebernaht aufgeführt; Wir unterscheiden drei verschiedene Klebersorten: Festigkeitskleber, Strukturkleber und Dichtigkeitskleber) und es gibt natürlich auch die Aufgabe Handling, dass heißt der Roboter nimmt ein Bauteil aus einer Vorrichtung und legt diesen in die nächste Vorrichtung oder er nimmt ihn und führt ihn einer stationären Punktschweißzange zu. Im Moment werden einige unsere Roboter ausgetauscht. Das neue Modell, der KUKA KR210 2700

Prime, ist das modernste Modell der Firma und repräsentiert den aktuellen Stand der Technik. Er ist genauer und erlaubt höhere Taktzeiten als das Vorgängermodell.

AP: Vom Aufbau her werden also zum Großteil Knickarmroboter eingesetzt?

WB: Es sind Sechsachsroboter, vorwiegend der Firma KUKA, einem deutschen Lieferanten mit Sitz in Augsburg. Die Gewichtsklassen der Roboter, die wir einsetzen, sind im Bereich rund um 210kg, die größten, die wir einsetzen, liegen bei 360kg Tragkraft - wir setzen Roboter mit 150kg, 180kg, 210kg und 350/360kg Tragkraft ein. Kleinere Roboter, wie 16kg Maschinen für MAG-Schweißen, sind bei uns früher eingesetzt worden, werden jetzt aber vorwiegend in der Lackierabteilung als Lackierroboter verwendet.

AP: Zur Roboterprogrammierung: Ich habe im Vorfeld in einem Vorbereitungskurs eine Arbeit über die verschiedenen Programmierarten geschrieben. Wie wird die Programmierung hier vorgenommen?

WB: Wir machen mittlerweile auch sehr viel in Richtung Offline Programmierung. Wir verwenden dazu ein Simulationsprogramm der Firma Siemens, dort werden, wenn eine neue Station kommt, die Pfade und die Punkte ermittelt und natürlich auch die Kollisionen untereinander. Das wird alles Offline vorbereitet, wie auch die Sicherheit des Roboters, dann eingespielt in den Roboter und dann entsprechend korrigiert und ergänzt: Die Punkte werden fein justiert (fein geteacht) und auch Abläufe zueinander sowie Anlagensteuerung werden geprüft. Auch müssen die Roboter mit Hilfe eines Loches in einer Scheibe, dessen Position bekannt ist, kalibriert werden.

AP: Meine letzte Frage wäre, in wie fern sich ein Programmierer mit der Vorwärtskoordinatentransformation auseinandersetzen muss oder ob diese schon in die Applikationen integriert ist?

WB: Leider kann ich das nicht beantworten, da ich kein Experte im Thema Offline Programmierung bin. Allerdings können wir das nachher den Experten von der Offline Programmierung fragen.

Andreas Huber (Offline Programmier): In meinem normalen Berufsalltag brauche ich die Vorwärtskoordinatentransformation nur selten. Diese ist eher für die Entwickler und Hersteller von Industrierobotern, in unserem Falle die Firma KUKA, wichtig, aber eher weniger für den Anwender eines Industrieroboters. Die direkte Kinematik ist in den Programmen zur Programmierung bereits integriert.

(Per Tonaufnahme aufgezeichnet - sprachlich adaptiert)

14.2 Programmcode MATLAB-Skript

```
%Berechnung des TCP aus den Gelenkswerten - Alexander Prutsch

%Info Messageboxen und Clear
uiwait(msgbox('---Berechnung des TCP bei Industrierobotern mit grafischer
Darstellung--- ----------Komma als Punkt eingeben - Werte in Grad &
Zentimeter----------','INFO'));
uiwait(msgbox('Beta=Drehwinkel l=Armteillänge a=Verschiebelänge
Gamma=Verwindungswinkel','INFO'));
clear all
close all
clc
v1=1;
v2=1;


while v1>0      %Schleife für mehrere Berechnungen
if v2>0         %Abfrage Anzahl der Armteile (nur bei '1. Berechnung' oder
'weitere Berechnung anderer Roboter')
    prompt = {'Anzahl der Armteile des Roboters'};
    dlg_title = 'Eingabe';
    num_lines = 1;
    def = {'6'};
    answer = inputdlg(prompt,dlg_title,num_lines,def);
    num = str2num(answer{:});
    %Platzhalter Arrays
    b=zeros(1,num);
    l=zeros(1,num);
    a=zeros(1,num);
    y=zeros(1,num);
    m=zeros(4,4+num*4);
    m(1:4,1:4)=[1,0,0,0;0,1,0,0;0,0,1,0;0,0,0,1];
    v2=1;
end


if v2>0
    for i=1:num         %Schleife fuer Berechnung aller Gelenke ('1.
Berechnung' oder 'weitere Berechnung anderer Roboter')

    %Abfrage Gelenkswerte
    prompt = {'Beta','l','a','Gamma'};
    dlg_title = sprintf(['Gelenk' num2str(i)]);
    num_lines = 1;
    def={'0', '0', '0', '0'};
    ant = inputdlg(prompt,dlg_title,num_lines,def);

    %Werte->Array
    b(1,i)=str2num(ant{1});
    l(1,i)=str2num(ant{2});
    a(1,i)=str2num(ant{3});
    y(1,i)=str2num(ant{4});

    %Position des vorigen Koordinatensystem
    tcpt=m(1:4,i*4-3:i*4);

    %Tansformationsmatrix
    mt=[cosd(b(1,i)),-
sind(b(1,i))*cosd(y(1,i)),sind(b(1,i))*sind(y(1,i)),a(1,i)*cosd(b(1,i));sin
d(b(1,i)),cosd(b(1,i))*cosd(y(1,i)),-
```

```
cosd(b(1,i))*sind(y(1,i)),a(1,i)*sind(b(1,i));0,sind(y(1,i)),cosd(y(1,i)),l
(1,i);0,0,0,1];

    %Position nach Überführung
    TCP=tcpt*mt;

    %Postion->Array
    m(1:4,i*4+1:i*4+4)=TCP;
    TCP;
    end
else
    for i=1:num         %Schleife fuer Abfrage und Berechnung aller Gelenke
('weitere Berechnung gleicher Roboter')
    choice = questdlg(['Gelenk' num2str(i)], 'Abfrage', 'Änderung des
Drehwinkel','Änderung der Schublänge','Keine Veränderung', 'Keine
Veränderung');
    switch choice
        case 'Änderung des Drehwinkel'
        prompt = {'Drehwinkel'};
        dlg_title = 'Eingabe';
        num_lines = 1;
        def = {'0'};
        answer = inputdlg(prompt,dlg_title,num_lines,def);
        b(1,i) = str2num(answer{:});
        tcpt=m(1:4,i*4-3:i*4);
        mt=[cosd(b(1,i)),-
sind(b(1,i))*cosd(y(1,i)),sind(b(1,i))*sind(y(1,i)),a(1,i)*cosd(b(1,i));sin
d(b(1,i)),cosd(b(1,i))*cosd(y(1,i)),-
cosd(b(1,i))*sind(y(1,i)),a(1,i)*sind(b(1,i));0,sind(y(1,i)),cosd(y(1,i)),l
(1,i);0,0,0,1];
        TCP=tcpt*mt;
        m(1:4,i*4+1:i*4+4)=TCP;
        case 'Änderung der Schublänge'
        prompt = {'Schublänge'};
        dlg_title = 'Eingabe';
        num_lines = 1;
        def = {'0'};
        answer = inputdlg(prompt,dlg_title,num_lines,def);
        a(1,i) = str2num(answer{:});
        tcpt=m(1:4,i*4-3:i*4);
        mt=[cosd(b(1,i)),-
sind(b(1,i))*cosd(y(1,i)),sind(b(1,i))*sind(y(1,i)),a(1,i)*cosd(b(1,i));sin
d(b(1,i)),cosd(b(1,i))*cosd(y(1,i)),-
cosd(b(1,i))*sind(y(1,i)),a(1,i)*sind(b(1,i));0,sind(y(1,i)),cosd(y(1,i)),l
(1,i);0,0,0,1];
        TCP=tcpt*mt;
        m(1:4,i*4+1:i*4+4)=TCP;
        case 'Keine Veränderung'
        tcpt=m(1:4,i*4-3:i*4);
        mt=[cosd(b(1,i)),-
sind(b(1,i))*cosd(y(1,i)),sind(b(1,i))*sind(y(1,i)),a(1,i)*cosd(b(1,i));sin
d(b(1,i)),cosd(b(1,i))*cosd(y(1,i)),-
cosd(b(1,i))*sind(y(1,i)),a(1,i)*sind(b(1,i));0,sind(y(1,i)),cosd(y(1,i)),l
(1,i);0,0,0,1];
        TCP=tcpt*mt;
        m(1:4,i*4+1:i*4+4)=TCP;
    end
    end
end


%Ausgabe Text
TCP
```

```
uiwait(msgbox('TCP: MATLAB Hauptfenster', 'TCP Ausgabe'))


%Ausgabe Grafik
if v2<1          %Alte Werte bei 2. Berechnung
    kxa=kx;
    kya=ky;
    kza=kz;
    ktxa=ktx;
    ktya=kty;
    ktza=ktz;
end


%Koordinaten bekommen
e=length(num);
kx=zeros(1:e,1);
ky=zeros(1:e,1);
kz=zeros(1:e,1);
for i=1:num
    kx(i+1,1)=m(1,4*i+4);
    ky(i+1,1)=m(2,4*i+4);
    kz(i+1,1)=m(3,4*i+4);
end
%Zeichnen
if v2>0          %1. Berechnung
    ktx=kx(i+1,1);
    kty=ky(i+1,1);
    ktz=kz(i+1,1);
    f=plot3(kx,ky,kz);
    xlabel('X');
    ylabel('Y');
    zlabel('Z');
    movegui(f,[-100,350]);
    text(ktx,kty,ktz,'TCP');
else             %Weitere Berechnung
    ktx=kx(i+1,1);
    kty=ky(i+1,1);
    ktz=kz(i+1,1);
    f=plot3(kx,ky,kz,'b',kxa,kya,kza,'r');
    movegui(f,[-100,350]);
    text(ktx,kty,ktz,'TCP');
    text(ktxa,ktya,ktza,'TCP alt','Color','r') ;
end


%Abrage nach weiter Berechnung
choice = questdlg('Weitere Berechnung?', 'Abfrage', 'Ja - gleicher
Roboter','Ja - anderer Roboter','Nein','Nein');
switch choice
    case 'Ja - gleicher Roboter'
        v2=0;
    case 'Ja - anderer Roboter'
        v2=1;
    case 'Nein'
        disp('Ende des Programs')
        uiwait(msgbox('Ende des Programms'))
        v1=0;
end
end
```