INVERSE OBERKÖRPER-KINEMATIK FÜR DIE HUMANOIDE ROBOTERPLATFORM NAO

STEPHAN BISCHOFF

Informatik, Mathematik und Naturwissenschaften
Hochschule für Technik, Wirtschaft und Kultur

April 2016

Betreuender Professor: Prof. Dr. rer. nat. Klaus Bastian
Betreuer Nao-Team: M. Sc. Thomas Reinhardt

SELBSTSTÄNDIGKEITSERKLÄRUNG

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Leipzig, April 2016

Stephan Bischoff

INHALTSVERZEICHNIS

1

EINLEITUNG

Um mit seiner Umwelt interagieren zu können besitzen Roboter über Arme aus mit Gliedern verbundenen Gelenken, sogenannte kinematische Ketten. Am Ende des Armes sind Werkzeuge angebracht, die verschiedene Aufgaben erfüllen sollen. Notwendige Voraussetzung für das Erfüllen dieser Aufgaben ist, dass sich das Werkzeug an der richtigen Position befindet. Die Positionierung des Werkzeuges am Ende eines Armes erfordert, dass die Winkel der Gelenke so gestellt werden, dass sich das Werkzeug aufgrund der Struktur des Armes an die gewünschte Position bewegt. Für die Berechnung der Gelenkwinkel mit gegebener Struktur des Armes und dem Zielpunkt wird die inverse Kinematik verwendet.

1.1 MOTIVATION

Das NAO-Team der HTWK programmiert humanoide Roboter des Herstellers Aldebaran, um im Rahmen der Standard Platform League des RoboCups an Fußballwettkämpfen teilzunehmen. Die Fußballspiele werden von den Robotern des Typs NAO autonom durchgeführt. Aufgrund des "Embedded"-Charakters und der einhergehenden knappen Systemressourcen wird sich bei der Erstellung der Software auf das Notwendigste beschränkt. So besitzt die Software des NAO-Teams der HTWK zwar über eine inverse Kinematik, jedoch wurde diese auf den Unterkörper beschränkt, um die üblichen Aktivitäten während eines Fußballspiels durchzuführen (Laufen, Kicken, Balllancieren, usw.). Bewegungen wie das Aufstehen nach einem Sturz, oder das Halten eines Balles in der Rolle des Torwarts wurden manuell konstruiert. Dies liefert zwar eine vorhersehbare und zuverlässige Bewegung während eines Fußballspiels, doch fehlen dynamischere autonome Möglichkeiten. Gerade mit steigender Popularität und dem bevorstehenden RoboCup in Leipzig rücken Präsentationen der Roboter und ihrer Möglichkeiten außerhalb des RoboCups und damit dem Roboterfußball in ein stärkeres Licht. Eine inverse Kinematik, die Gelenkwinkelberechnungen für den gesamten Roboter durchführen kann, wäre im Angesicht des steigenden Interesses also wünschenswert.

1.2 Zielsetzung und Abgrenzung

Ziel dieser Arbeit ist die Beschreibung mehrerer Verfahren zur Lösung des Problems der inversen Kinematik und der zugrunde liegenden Datenstruktur der kinematischen Kette des NAO-Oberkörpers. Weiter sollen die beschriebenen Verfahren mit Bezug auf die Anforderungen des NAO-Teams der HTWK bewertet werden. Die entscheidenden Kriterien hierfür sind Berechnungszeit, Flexibilität und Stabilität.

1.3 Aufbau der Arbeit

Im Kapitel 2 GRUNDLAGEN werden die Grundlagen der vorwärts (oder forward) und inversen Kinematik beschrieben. Für diese Arbeit relevante Aussagen über die Hardware des NAO werden in Kapitel 2.1 *Der* NAO getroffen. Die für das Verständnis wichtigen Kapitel 2.2.1 *Koordinatentransformation* und 2.2.2 *Forward Kinematik* beschreiben die Darstellung der Struktur des NAO. In Kapitel 2.2.3 *Inverse Kinematik* wird kurz beschrieben, was die inverse Kinematik leistet und welche Ansätze verfolgt werden können.

Das Kapitel 3 INVERSE OBERKÖRPER-KINEMATIK definiert die Datenstruktur der kinematischen Kette, auf der die inverse Kinematik arbeitet (3.1 *Kinematische Ketten des* NAO) und stellt drei Verfahren vor (3.2 *Transponierte und Pseudoinverse Jacobi-Matrix*, 3.3 *Analytisches Verfahren* und 3.4 *FABRIK*) und vergleicht diese in 3.5 *Vergleich*.

Kapitel 4 schließt die Arbeit mit der Argumentation über die geeignetste Wahl und dem Ausblick auf die mögliche Zukunft der inversen Kinematik im NAO-Team der HTWK ab.

2

GRUNDLAGEN

In diesem Kapitel werden die Grundlagen für die Verfahren der inversen Kinematik behandelt. Es folgt eine kurze Beschreibung über die zur Verfügung stehende Hardware des NAO gefolgt von der allgemeinen Beschreibung der Kinematik. Für das Verständnis notwendige Grundlagen der Kinematik sind Koordinatentransformationen, welche das mathematische Werkzeug für die Beschreibung kinematischer Ketten bilden. Anschließend wird die forward Kinematik beschrieben, da diese als Ausgangspunkt für viele Verfahren der inversen Kinematik, zumindest aber als Werkzeug zum Prüfen der Korrektheit der über die inverse Kinematik gefundenen Lösungen, dient. Abschließend wird erklärt, was die inverse Kinematik ist und grob, welche Arten an Algorithmen zu unterscheiden sind.

2.1 DER NAO

Der NAO ist ein von Aldebaran entwickelter humanoider Roboter der über 21 Gelenke (davon 12 im Oberkörper) an für Humanoide typischen Positionen verfügt. Über diese Gelenke kann sich der NAO ähnlich dem Menschen bewegen. Darunter zählt unter Anderem die Fortbewegung mit zwei Beinen, oder das Greifen mit einer dem Menschen nachempfunden Hand. Für Berechnungen steht dem NAO in der Version 5 eine 1,6 GHz Intel ATOM Z530 CPU und 1 GB Arbeitsspeicher zur Verfügung. Für die Ansteuerung der Hardware ist ein Microcontroller integriert mit dem kommuniziert werden kann. An den Microcontroller gesendete Daten beinhalten die Sollwerte für die Gelenkwinkel. Vom Microcontroller empfangene Daten sind unter Anderem die aktuellen Werte der Gelenkwinkel, Gyroscope, Accelerometer usw.

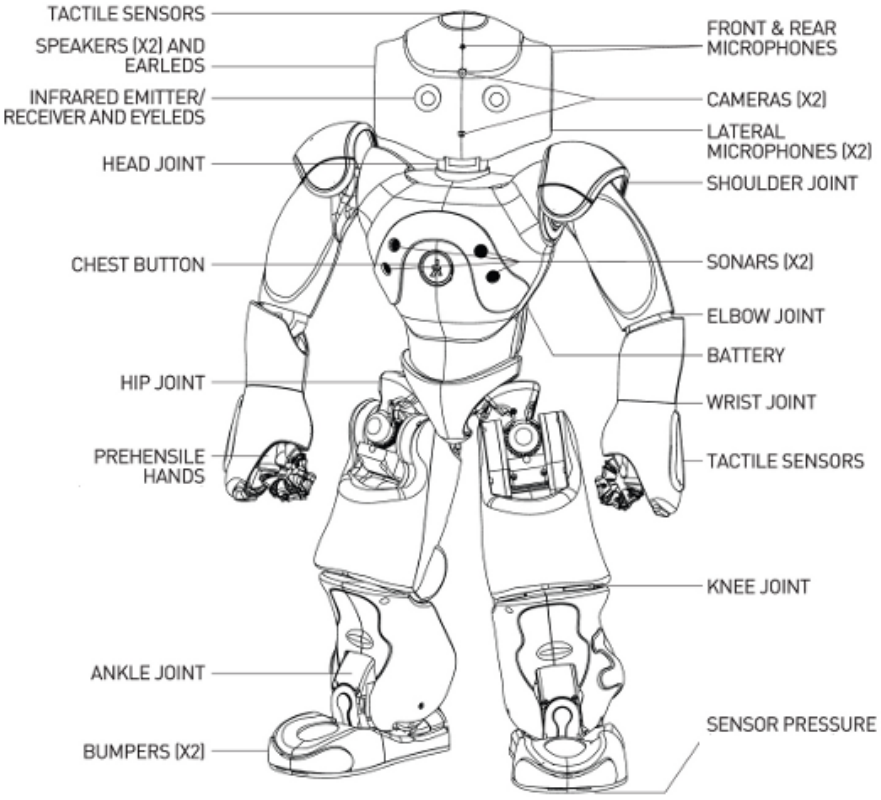

Abbildung 1: Der NAO © *Aldebaran-Robotics*

2.2 KINEMATIK

Die Kinematik ist die Lehre der Bewegung von Körpern und Punkten im Raum. Jegliche Kräfte, die auf den Körper einwirken, werden bei der Betrachtung ignoriert. Gegenstand der Betrachtungen sind Körper, die sich im Bezug zueinander bewegen. Die Bewegung kann dabei sowohl Translation, als auch Rotation darstellen und beschleunigt sein. Sind die die betrachteten Körper über ein Gelenk oder eine starre Verbindung gekoppelt, beschreibt die Kinematik das Zusammenwirken dieses Mehrkörpersystems.

2.2.1 *Koordinatentransformation*

Um eine kinematische Kette möglichst einfach zu beschreiben bietet sich die Anwendung lokaler Koordinatensysteme pro Kettenglied an. Durch diese lokalen Koordinatensysteme werden für das aktuelle Kettenglied alle vorhergehenden und nachfolgenden Parameter irrelevant. Beschrieben wird nur, welchen Abstand das nächste Gelenk zum vorherigen hat und welchen Winkel es einnimmt. Dadurch wird

die Beschreibung nicht nur anschaulicher, sondern auch leichter berechenbar.

Da nachher die tatsächliche Lage der Kette im $\mathbb{R}^3$ nicht unerheblich ist, müssen die Koordinatensysteme ineinander überführbar sein. Hierfür werden Koordinatentransformationen genutzt, die im Falle der Aufgabenstellung Rotationen und Translationen umrechnen müssen.

Für Koordinatensysteme, die um einen Winkel θ um eine Achse des Referenzsystems gedreht sind, lassen sich folgende Rotationsmatrizen verwenden [GPSo6]:

$$\mathbf{R}_{x,\theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \tag{2.2.1.1}$$

$$\mathbf{R}_{y,\theta} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{2.2.1.2}$$

$$\mathbf{R}_{z,\theta} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2.1.3}$$

Soll ein Vektor $\vec{p}$ vom System 1 in das System 0 abgebildet werden, wird der Vektor aus dem System 1 mit der entsprechenden Rotationsmatrix multipliziert:

$$\vec{p}_0 = \mathbf{R}_0^1 \vec{p}_1$$

$\vec{p}_1$ ist dabei der Vektor $\vec{p}$ im Referenzsystem 1 und entsprechend $\vec{p}_0$ der resultierende Vektor im System 0. Die Matrix $\mathbf{R}_0^1$ bezeichnet die Matrix, die den Vektor $\vec{p}_1$ in den Vektor $\vec{p}_0$ überführt. Allgemein steht das Superscript einer Transformationsmatrix für das Quellsystem und das Subscript für das Zielsystem.

Ist nun eine Kette an Referenzsystemen gegeben, kann ein Vektor $\vec{p}_n$ im Referenzsystem n in das System 0 durch „*in-order*“ Multiplikation der Rotationsmatrizen überführt werden:

$$\vec{p}_0 = \left(\prod_{i=0}^{n-1} \mathbf{R}_i^{i+1} \right) \vec{p}_n \tag{2.2.1.4}$$

Zu beachten ist, dass die Winkel für die Rotationsmatizen jeweils die Winkel zwischen den jeweiligen Koordinatensystemen sind und relativ zum Zielsystem angegeben sind. Die Notwendigkeit einer Reihenfolge ist durch die Nichtkommutativität des Matrixprodukts gegeben ($\mathbf{N} \cdot \mathbf{M} \neq \mathbf{M} \cdot \mathbf{N}$). Eine andere Möglichkeit wäre, alle Winkel im Bezug auf das Referenzsystem 0 anzugeben. Dadurch ergibt sich allerdings eine umgekehrte Reihenfolge bei der Multiplikation einer Kette.

Aufgrund der Aufgabenstellung wird schnell klar, dass die Beschreibung der Rotation um den Koordinatenursprung noch nicht ausreicht. Bedingt durch die Geometrie des Roboters wird eine Möglichkeit benötigt, Rotationspunkte innerhalb des $\mathbb{R}^3$ anzugeben. Für die Lösung dieses Problems können homogene Transformationen genutzt werden [Wol11, 2-1]:

$$\mathbf{H} = \begin{pmatrix} \mathbf{R} & \vec{d} \\ 0 & 1 \end{pmatrix} \tag{2.2.1.5}$$

Mit dem Vektor $\vec{d}$ wird das „*displacement*“, also die Translation, angegeben. $\mathbf{R}$ bezeichnet weiterhin die Rotation. Hier wird allerdings statt um $\vec{o}$ um die von $\vec{d}$ angezeigte Position rotiert. Sowohl der Winkel θ als auch der Vektor $\vec{d}$ werden im Bezug auf das Zielsystem angegeben. Um die homogene Transformation nutzen zu können, muss jeder Positionsvektor um eine Komponente erweitert werden (homogene Koordinaten):

$$\vec{p}_{neu} = \begin{pmatrix} \vec{p}_{alt} \\ 1 \end{pmatrix}$$

Ein Beispiel mit $\mathbf{R}_{z,\pi/2}$ und $\vec{d} = (0{,}5{,}0)^T$ auf den Vektor $\vec{p} = (1{,}0{,}0)^T$:

$$\vec{p}_0 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 6 \\ 0 \\ 1 \end{pmatrix}$$

Um die Richtungen der Rotationstransformation umzukehren muss die inverse Rotationsmatrix für die entsprechende Achse verwendet werden. Da es sich bei den Rotationsmatrizen um orthogonale Matrizen handelt und daher $\mathbf{R}\mathbf{R}^T = \mathbf{E}$, mit der Einheitsmatrix $\mathbf{E}$, gilt, ist die Inverse gleich der Transponierten [Wol11, S. 2-2]:

$$\mathbf{R}^{-1} = \mathbf{R}^T \tag{2.2.1.6}$$

Die inverse homogene Matrix lässt sich dann mit

$$\mathbf{H}^{-1} = \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\vec{d} \\ 0 & 1 \end{pmatrix} \tag{2.2.1.7}$$

berechnen.

2.2.2 *Forward Kinematics*

Sind alle Gelenkparameter einer Folge von Gliedern (Kette) gegeben, so kann die Position des *End-Effektors*, durch Anwendung einer Folge von Transformationen, eindeutig berechnet werden. Dies wird als Vorwärtsrechnung oder *Forward Kinematics* bezeichnet.

Analog zu den im Kapitel 2.2.1 getroffenen Notationen bezüglich der Transformationen, wird auch hier das zugehörige Bezugssystem im Subskipt angegeben. Dass dem n-ten Arm zugeordnete Koordinatensystem wird entsprechend mit S_n bezeichnet. Der n-te Arm ist dabei die Verbindung zwischen dem n-ten und dem $(n+1)$-ten Gelenk.

Aufgrund der sich ergebenden, einfachen Transformationsmatrizen für die späteren Berechnung, hat sich für die Beschreibung der Gelenkketten die Festlegungen nach Denavit und Hartenberg als üblich ergeben. Grundsetzlich geht es um die Lage und Ausrichtung der lokalen Koordinatensysteme der Gelenke [SB96][DH55]:

1 S_0 ist das, in der Regel ortsfeste, Basiskoordinatensystem.

2 Die z_n-Achse wird entlang der Bewegungsachse des $(n+1)$-ten Gelenks gelegt.

3 Die x_n-Achse ist normal zur z_{n-1}-Achse und zeigt von ihr weg.

4 Die y_n-Achse wird so festgelegt, dass sich ein Rechtssystem ergibt.

Da sich die Punkte 3 und 4 auf ein Vorgängersystem beziehen, kann das Basiskoordinatensystem fast frei ausgerichtet werden, sollange Punkt 2 nicht verletzt wird. Ähnliches gilt für die Ausrichtung des End-Effektors, der kein nachvolgendes Gelenk hat und daher nicht nach Punkt 2 gerichtet werden kann.

Die Lage der Koordinatensysteme kann nach Denavit und Hartenberg durch 4 Parameter beschrieben werden (siehe Abbildung 2):

- a_n kürzeste Entfernung zwischen der z_{n-1}-Achse und der z_n-Achse
- α_n Winkel zwischen z_{n-1}-Achse und z_n-Achse um die x_n-Achse
- d_n Entfernung (entlang der z_{n-1}-Achse) vom Ursprung O_{n-1} bis zum Schnittpunkt der z_{n-1}-Achse mit der x_n-Achse
- θ_n der Gelenkwinkel um die z_{n-1}-Achse

Abgesehen von θ_n sind alle Parameter abhängig von der Struktur des Gliedes und daher konstant. Der Winkel θ_n zeigt die Stellung des Gliedes an und kann verändert werden. Für jeden der vier Parameter wird auf ein Koordinatensystem S_{n-1} eine Transformation angewandt um es in das System S_n zu überführen. Wie in Kapitel 2.2.1 beschrieben, muss die Reihenfolge der Transformationen eingehalten

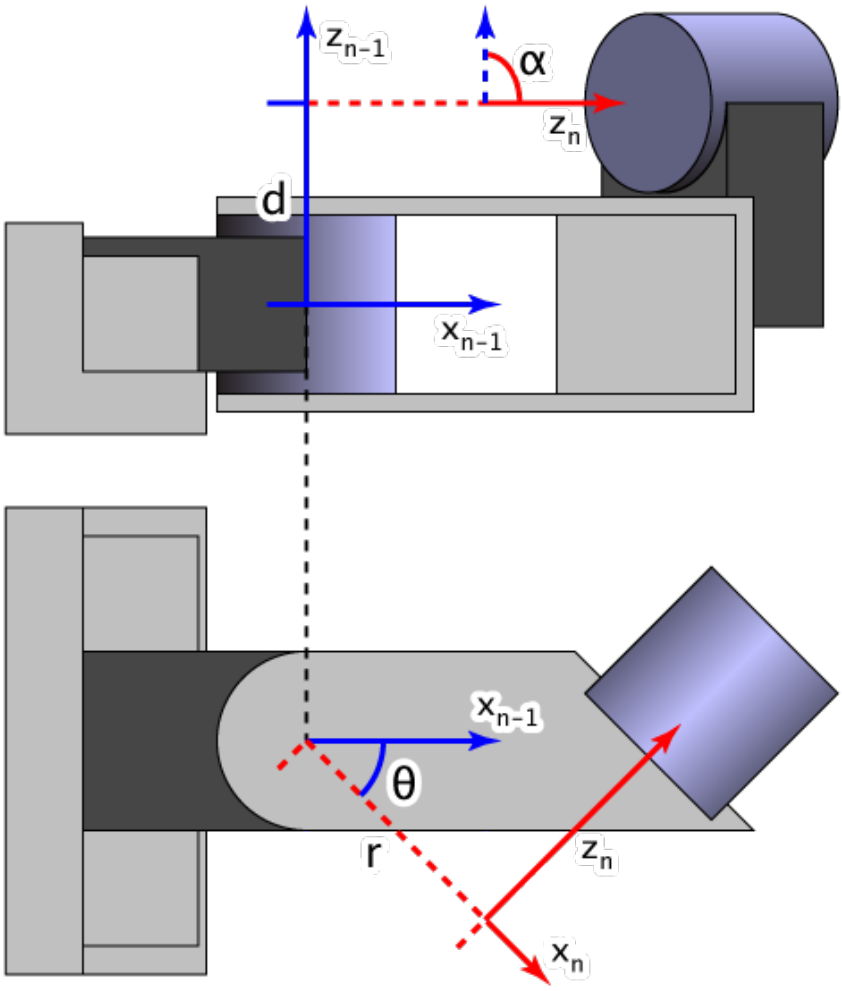

Abbildung 2: Beispiel der Denavit-Hartenberg-Parameter [TT09]

werden. Im Falle von Denavit-Hartenberg ist das eine Translation entlang der z_{n-1}-Achse um d_n:

$$\vec{d}_n = \begin{pmatrix} 0 \\ 0 \\ d_n \end{pmatrix} \quad \rightarrow \quad T_{\vec{d}_n} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2.2.1}$$

Eine Drehung um die z_{n-1}-Achse um θ_n:

$$\mathbf{R}_{z,\theta_n} = \begin{pmatrix} \cos\theta_n & -\sin\theta_n & 0 & 0 \\ \sin\theta_n & \cos\theta_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2.2.2}$$

Eine Translation entlang der neuen x-Achse um a_n:

$$\vec{a}_n = \begin{pmatrix} a_n \\ 0 \\ 0 \end{pmatrix} \quad \rightarrow \quad \mathbf{T}_{\vec{a}_n} = \begin{pmatrix} 1 & 0 & 0 & a_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2.2.3}$$

Sowie einer Drehung um die neue x-Achse um α_n:

$$\mathbf{R}_{x,\alpha_n} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_n & -\sin\alpha_n & 0 \\ 0 & \sin\alpha_n & \cos\alpha_n & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2.2.4}$$

Die DH-Transformationsmatrix ergibt sich dann durch

$$\mathbf{A}^n_{n-1} = \mathbf{T}_{\vec{d}_n} \cdot \mathbf{R}_{z,\theta_n} \cdot \mathbf{T}_{\vec{a}_n} \cdot \mathbf{R}_{x,\alpha_n} \tag{2.2.2.5}$$

Ausmultipliziert:

$$\mathbf{A}^n_{n-1} = \begin{pmatrix} \cos\theta_n & -\cos\alpha_n \sin\theta_n & \sin\alpha_n \sin\theta_n & a_n \cos\theta_n \\ \sin\theta_n & \cos\alpha_n \cos\theta_n & -\sin\alpha_n \cos\theta_n & a_n \sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2.2.6}$$

mit der Inversen

$$\left(\mathbf{A}^n_{n-1}\right)^{-1} = \mathbf{A}^{n-1}_n = \begin{pmatrix} \cos\theta_n & \sin\theta_n & 0 & -a_n \\ -\cos\alpha_n \sin\theta_n & \cos\alpha_n \cos\theta_n & \sin\alpha_n & -d_n \sin\alpha_n \\ \sin\alpha_n \sin\theta_n & -\sin\alpha_n \cos\theta_n & \cos\alpha_n & -d_n \cos\alpha_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2.2.7}$$

Wie in Formel 2.2.1.4 beschrieben, lassen sich durch die Multiplikation mehrerer Transformationsmatrizen mehrgliedrige Ketten berechnen:

$$\mathbf{A}^n_0 = \mathbf{A}^1_0 \cdot \mathbf{A}^2_1 \cdot \mathbf{A}^3_2 \cdot \ldots \cdot \mathbf{A}^n_{n-1} \tag{2.2.2.8}$$

Durch Multiplikation mit dem lokalen Koordinatenursprung (also dem Nullvektor $\vec{o}_n$) kann die Position des End-Effektors relativ zum Koordinatensystem S_0 berechnet werden:

$$\vec{p}_0 = \mathbf{A}^n_0 \cdot \vec{o}_n \tag{2.2.2.9}$$

Da die Winkel θ die einzigen Variablen sind, ist die forward Kinematik eine Funktion über die Winkel der einzelnen Glieder:

$$\vec{p} = f(\theta_0,\theta_1,\ldots,\theta_n) = \mathbf{A}^n_0(\theta_0,\theta_1,\ldots,\theta_n) \cdot \vec{o}_n \tag{2.2.2.10}$$

2.2.3 *Inverse Kinematic*

Im Gegensatz zur forward Kinematik sind bei der inversen Kinematik die Winkel anhand gegebener Position und Ausrichtung des End-Effektors zu berechnen. Abhängig von der Anzahl der Freiheitsgrade sind auch die Lösungen oft nicht eindeutig.

Für die Berechnung der inversen Kinematik gibt es kein Verfahren, was allgemein gültig angewandt werden könnte. Stattdessen werden verschiedene Ansätze verfolgt, die auf die spezifischen Anforderungen der Anwendung Rücksicht nehmen.

- Beim *analytischer Ansatz* werden die Denavit-Hartenberg-Transformationsmatrizen der einzelnen Glieder invertiert und mit der den End-Effektor beschriebenden homogenen Matrix gleichgesetzt. Das resultierende Gleichungssystem kann dann nach den Winkeln aufgelöst werden.

- Mittels *iterativer Verfahren* werden die Winkel der Gelenke über mehrere gleichartige Schritte angenähert.

Ausschlaggebend für die Wahl des Ansatzes ist die Komplexität der kinematischen Kette und die zur Verfügung stehende Rechenzeit. Nachfolgend wird auf die drei verschiedenen Ansätze näher eingegeangen.

3

INVERSE OBERKÖRPER-KINEMATIK

In diesem Kapitel werden die kinematischen Ketten definiert und die Forward Kinematik aufgestellt auf deren Basis die drei nachfolgenden Verfahren aufbauen. Das erste vorgestellte Verfahren dient als Präsentation eines allgemeingültigen Standards, während das analytische Verfahren sehr speziell auf die Struktur der kinematischen Ketten zugeschnitten ist. Abschließend wird mit dem FABRIK ein sehr flexibler iterativer Algorithmus präsentiert.

3.1 KINEMATISCHE KETTEN DES NAO

Die kinematische Kette des NAO's – in dieser Arbeit speziell des Oberkörpers – ist die Datenstruktur, auf der die Algorithmen der inversen Kinematik arbeiten. Wie in Abbildung 3 dargestellt, besitzt der Oberkörper des NAO über drei End-Effektoren: Kopf, sowie linke und rechte Hand.
Der Kopf wird in dieser Arbeit ignoriert, da die Ansteuerung über direkte Winkelvorgaben sinnvoller erscheint.

In Tabelle 1 sind die Maße der linken Seite des NAO-Oberkörpers, wie in [AR12, H25 – Links] angegeben, aufgeführt. Sowohl Schulter-,

Quelle	Ziel	x (mm)	y (mm)	z (mm)
Torso	Schulter l (Pitch)	0.00	98.00	100.00
Schulter l (Pitch)	Schulter l (Roll)	0.00	0.00	0.00
Schulter l (Roll)	Ellbogen l (Yaw)	105.00	15.00	0.00
Ellbogen l (Yaw)	Ellbogen l (Roll)	0.00	0.00	0.00
Ellbogen l (Roll)	Handgelenk l (Yaw)	55.95	0.00	0.00

Tabelle 1: Maße der linken Oberkörperseite.

als auch Ellbogen-Gelenke besitzen über jeweils zwei Rotationsachsen: Roll und Yaw. Das Handgelenk wird in dieser Arbeit ignoriert, da die Handgelenke der NAO's des Nao-Teams der HTWK für mehr Stabilität fixiert sind. Die Rotationsreihenfolge lässt sich aus Tabelle 1 ablesen. *Roll*, *pitch* und *yaw* bezeichnen die Winkel um folgende Achsen:

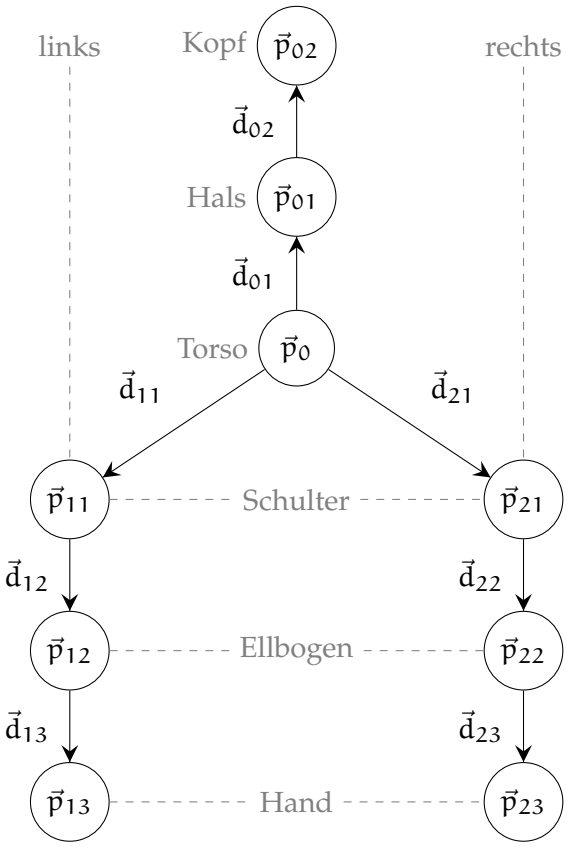

Abbildung 3: Kinematischer Baum des NAO-Oberkörpers. Die Vektoren $\vec{p}_x$ sind die Positionsvektoren relativ zu $\vec{p}_0$ und somit zum globalen Koordinatensystem. Die Vektoren $\vec{d}_x$ bezeichnen die Positionsvektoren relativ zum in der Kette vorhergehenden Knoten.

- Roll: Rotation um die x-Achse
- Pitch: Rotation um die y-Achse
- Yaw: Rotation um die z-Achse

Aldebaran-Robotics hat für Längen > 0 die in Tabelle 2 angegebenen Namen festgelegt.

Ausgehend vom Torsormittelpunkt zeigt die x-Achse in Richtung Front des Nao, die y-Achse wird Richtung linke Seite positiv und die z-Achse zeigt nach oben (siehe Abbildung 4). Die Maße für die rechte Seite sind an der x-z-Ebene gespiegelt.

Obwohl der Torso die Wurzel des kinematischen Baumes ist, dient er nur als Referenzpunkt für die Schultern, die die tatsächlichen Basen für die ihnen nachfolgenden Ketten bilden.

Für die vier Gelenke pro Kette sind die Denavit-Hartenberg-Parameter in Tabelle 3 wie in [KOL] angegeben aufgeführt. Wie in Kapitel 2.2.2 beschrieben, ergeben sich die α's aus der Struktur, speziell der Rotation der Rotationsachse des Gelenks um die x-Achse, des Gelenks so,

Name	Länge in mm
ShoulderOffsetY	98.00
ElbowOffsetY	15.00
UpperArmLength	105.00
LowerArmLength	55.98
ShoulderOffsetZ	100.00
HandOffsetX	57.75
HandOffsetZ	12.31

Tabelle 2: Benannte Maße der Arme

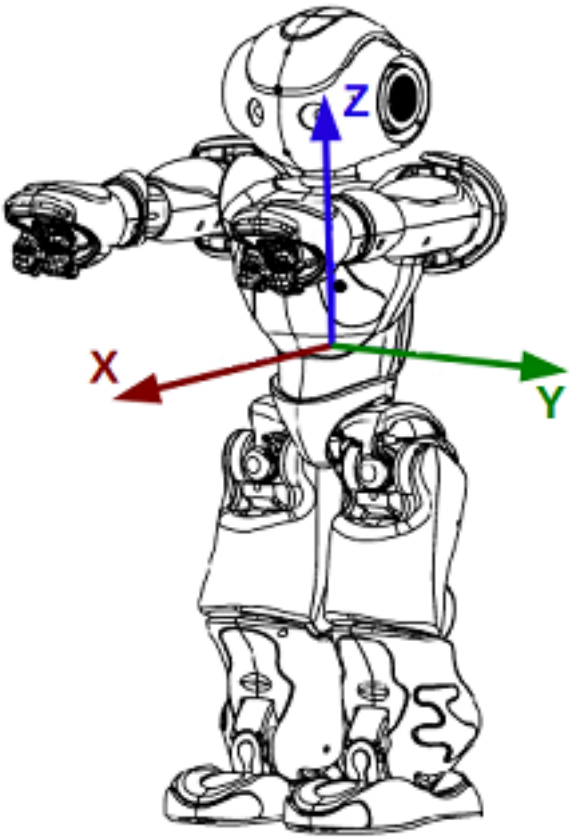

Abbildung 4: Achsen des NAO © *Aldebaran-Robotics*

dass die Rotationsachse des Gelenks der lokalen z-Achse entspricht und den Festlegungen nach Denavit und Hartenberg entspricht. Da sich das erste Schultergelenk (Pitch) um die globale y-Achse dreht, muss eine Rotation um die globale x-Achse um $-\pi/2$ vorgenommen werden. Die neue y-Achse zeigt daraufhin in die globale $-z$-Richtung. Aufgrund der Ausrichtung der Rotationsachse des zweiten Schultergelenks (Roll) entlang der globalen z-Achse, muss mit einem

Gelenk	a	α	d	θ
Schulter l (Pitch)	0	$-\frac{\pi}{2}$	0	θ_1
Schulter l (Roll)	0	$\frac{\pi}{2}$	0	$\theta_2 - \frac{\pi}{2}$
Ellbogen l (Yaw)	0	$-\frac{\pi}{2}$	UpperArmLength	$-\theta_3$
Ellbogen l (Roll)	0	$\frac{\pi}{2}$	0	θ_4

Tabelle 3: Denavit-Hartenberg-Parameter des linken Arms.

α-Winkel von $\pi/2$ um die lokale (und globale) x-Achse das aktuelle Koordinatensystem zurückgedreht werden. Für das Koordinatensystem des ersten Ellbogengelenks (Yaw) wäre die y-Achse die optimale Achse für eine Rotation zur Ausrichtung der lokalen z-Achse des Gelenks. Der Denavit-Hartenberg-Parameter α bezieht sich jedoch auf eine Rotation um die x-Achse. Es wird daher eine Rotation um die z-Achse des vorhergehenden zweiten Schultergelenks (Roll) um $-\pi/2$ vorgenommen um die x-Achse für den Ellbogen auszurichten ($\theta_2 - \pi/2$). Die x-Achse zeigt daraufhin in die globale $-$y-Richtung. Mit einem α von $-\pi/2$ wird dann die z-Achse auf die Rotationsachse des ersten Ellbogengelenks (Yaw) gelegt. Da sich der Ellbogen am Ende des Oberarms befindet, muss noch um UpperArmLength verschoben werden (DH-Parameter d). Für das zweite Ellbogengelenk (Roll) reicht ein α von $\pi/2$.

Zusätzlich zu den Denavit-Hartenberg-Parametern muss das Offset der Schulter formuliert werden. Hierfür reicht eine reine Translationsmatrix:

$$\mathbf{T}^0_{\text{Basis}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mit $d_y = \text{ShoulderOffsetY} + \text{ElbowOffsetY}$ und $d_z = \text{ShoulderOffsetZ}$ liegt der Koordinatenursprung der Basis in der Mitte der y-z-Ebene des Armes ohne Offset entlang der x-Achse. Für den Endeffektor am Ende der Hand müssen schließlich noch eine Rotation z-Achse um $\pi/2$ ($\mathbf{R}_{z,\pi/2}$), sowie eine Translation entlang der x-Achse um HandOffsetX + LowerArmLength (d_x) durchgeführt werden:

$$\mathbf{T}_{\text{End}} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Die Transformationsmatrix für den linken Arm lässt sich somit wie in Gleichung 3.1.0.1 aufstellen.

$$\mathbf{A} = \mathbf{T}^0_{\text{Basis}} \mathbf{A}^1_0 \mathbf{A}^2_1 \mathbf{A}^3_2 \mathbf{A}^4_3 \mathbf{A}^5_4 \mathbf{R}_{z,\pi/2} \mathbf{T}_{\text{End}} \tag{3.1.0.1}$$

Aufgrund der genannten Spiegelsymmetrie des Oberkörpers an der x-z-Ebene sind die meisten Parameter der kinematischen Kette des rechten Armes der des linken Arms identisch. Unterschiede gibt es nur bei Parametern, die sich auf die y-Achse beziehen. Die Denavit-Hartenberg-Parameter des rechten Armes sind in Tabelle 4 aufgeführt. Da ab der Zusatzrotation mit θ_2 die z-Achse in die nega-

Gelenk	a	α	d	θ
Schulter r (Pitch)	0	$-\frac{\pi}{2}$	0	θ_1
Schulter r (Roll)	0	$\frac{\pi}{2}$	0	$\theta_2 + \frac{\pi}{2}$
Ellbogen r (Yaw)	0	$-\frac{\pi}{2}$	$-$UpperArmLength	θ_3
Ellbogen r (Roll)	0	$\frac{\pi}{2}$	0	θ_4

Tabelle 4: Denavit-Hartenberg-Parameter des rechten Arms.

tive globale x-Richtung, statt der positiven wie im linken Arm, zeigt, sind die d-Parameter negativ. Auch die Translationen und Rotationen müssen angepasst werden:

$$\mathbf{T}^0_{\text{Base}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

mit $d_y = -\text{ShoulderOffsetY} - \text{EllbowOffsetY}$. d_z bleibt von der Spiegelsymmetrie unberührt und ist gleich ShoulderOffsetZ. Für die Anpassung für den End-Effektor ist wieder eine Rotation um die z-Achse von $-\pi/2$ notwendig. Die Translation entlang der x-Achse ist gleich der des linken Armes. Schließlich muss noch eine Rotation um die z-Achse von $-\pi$ vorgenommen werden, da die z-Achse invertiert ist. Für den rechten Arm lässt sich dann analog zum linken Arm die in Gleichung 3.1.0.2 gezeigte Transformationsmatrix aufstellen.

$$\mathbf{A} = \mathbf{T}^0_{\text{Basis}} \mathbf{A}^1_0 \mathbf{A}^2_1 \mathbf{A}^3_2 \mathbf{A}^4_3 \mathbf{A}^5_4 \mathbf{R}_{z,-\pi/2} \mathbf{T}_{\text{End}} \mathbf{R}_{z,-\pi} \tag{3.1.0.2}$$

Die aufgestellten Transformationsmatizen stellen die Datenstruktur für viele Verfahren der Inversen Kinematik dar und werden in den nächsten beiden Kapiteln als Grundlage verwendet.

3.2 TRANSPONIERTE UND PSEUDOINVERSE JACOBI-MATRIX

Die Jacobi-Matrix einer Funktion $f : \mathbb{R}^n \to \mathbb{R}^m$ ist die $m \times n$-Matrix aller partiellen Ableitungen dieser Funktion. Sie ist eine lineare Approximation einer differenzierbaren Funktion in der Nähe eines gegebenen Punktes. Angewand auf die Funktion der forward Kinematik wird mit der Jacobi-Matrix eine Positionsänderung des End-Effektors

aufgrund von Winkeländerungen der Gelenke berechnet (Gleichung 3.2.0.2). Ein Parameter des Verfahrens ist daher der Fehlervektor $\vec{e}$ der sich aus der aktuellen Position $\vec{r}$ des End-Effektors und der Zielposition $\vec{t}$ errechnet: $\vec{e} = \vec{t} - \vec{r}$. Ziel des Verfahrens ist es $\vec{e}$ zu minimieren.

$$\mathbf{J}(\theta) = \frac{\partial f}{\partial \theta} \tag{3.2.0.1}$$

$$\Delta\vec{r} = \mathbf{J}(\theta)\Delta\theta \tag{3.2.0.2}$$

Aufgrund der Linearisierung durch die Jacobi-Matrix lässt sich die in Gleichung 3.2.0.2 aufgestellte Approximation der forward Kinematik schneller berechnen, als die im vorhergehenden Kapitel aufgestellten Transformationsmatrizen in Gleichung 3.1.0.1 und 3.1.0.2, wenn auch nicht so präzise. Um nun die Winkeländerungen $\Delta\theta$ für $\vec{r} + \Delta\vec{r} = \vec{t}$, also $\Delta\vec{r} = \vec{e}$ zu finden, wird Gleichung 3.2.0.2 nach $\Delta\theta$ umgestellt:

$$\mathbf{J}(\theta)^{-1}\vec{e} = \Delta\theta \tag{3.2.0.3}$$

Da die Linearisierung zwar das Problem vereinfacht, aber dadurch Ungenauigkeiten einführt, muss nach einer Berechnung von $\Delta\theta$ die genaue Position des End-Effektors über die nicht-linearen Transformationsmatrizen 3.1.0.1 und 3.1.0.2 bestimmt werden und ein neuer Fehlervektor $\vec{e}$ aufgestellt werden. Ist $\vec{e}$ außerhalb einer Entfernungstoleranz, müssen erneut $\Delta\theta$'s berechnet werden. Audgrund dieser notwendigen Repetition ist das Lösen der inversen Kinematik über die Jacobi-Matrix als iteratives Verfahren klassifiziert. Abgesehen von der genannten Toleranz für die Positionsabweichung ist eine zweite Abbruchbedingung der Iteration gegeben durch eine fehlende Änderung in der Position des End-Effektors.

Die Jacobi-Matrix lässt sich nach [Nil09] mit Gleichung 3.2.0.4 berechnen.

$$\frac{\partial f}{\partial \theta_i} = \vec{a}_i \times \vec{q}_i \tag{3.2.0.4}$$

$\vec{a}_i$ bezeichnet den Einheitsvektor in Richtung der Rotationsachse des i-ten Gelenks und $\vec{q}_i$ bezeichnet den Vektor von der Position des Gelenks bis zum End-Effektor (Abbildung 5). Wie in Abbildung 6 dargestellt nähern sich die Vektoren der Differentiale einer Geraden an, wenn sie sich nahe einer Singularität befinden. Dies führt zu großen Änderungen in den berechneten Winkeln und dementsprechend zu sprunghaftem Verhalten. Eine Möglichkeit der Vermeidung dieses Problems ist das Prüfen auf Erreichbarkeit vor der tatsächlichen Berechnung der Differentiale. Unter Umständen ist es sinnvoll alle Punkte außerhalb der Reichweite der kinematischen Kette an den Rand der Reichweite zu mappen und so die Singularität zu eliminieren.

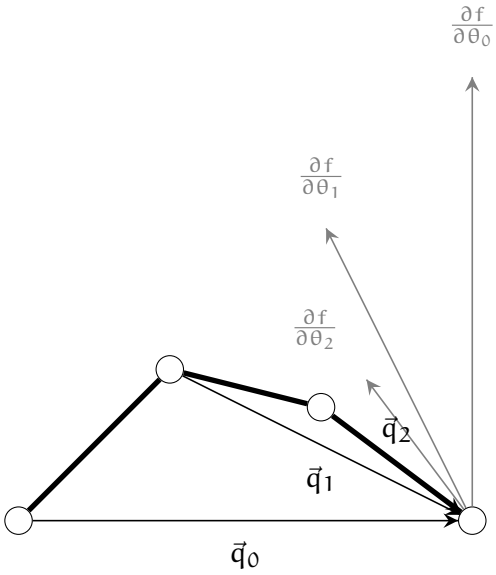

Abbildung 5: Veranschaulichung der Jacobi-Matrix. $\vec{a}$ zeigt für alle Gelenke in die positive z-Richtung (aus der Seite heraus).

Im Fall des NAO-Oberkörpers gibt es für jede Hand 4 Gelenke. Die für diese kinematische Kette notwendige Jacobi-Matrix ist daher eine 3×4 Matrix und somit nicht quadratisch und entsprechend nicht invertierbar. Im Folgenden werden zwei Methoden beschrieben, die eine Alternative zur Inversen darstellen.

3.2.1 *Transponierte Jacobi-Matrix*

Eine sehr einfache Möglichkeit den Umstand der Nicht-Invertierbarkeit der 3×4 Jacobi-Matrix zu umgehen, ist die Verwendung der Transponierten anstatt der Inversen:

$$\Delta\theta = \alpha J^T \vec{e} \tag{3.2.1.1}$$

Der Beweis wie in [Buso4] beschrieben:

SATZ 1:
Für alle J *und* $\vec{e}$ *gilt* $\langle JJ^T\vec{e}, \vec{e} \rangle \geqslant 0$

BEWEIS 1:
$\langle JJ^T\vec{e}, \vec{e} \rangle = \langle J^T\vec{e}, J^T\vec{e} \rangle = \| J^T\vec{e} \|^2 \geqslant 0.$ □

Die Richtung der Positionsänderung durch eine Winkeländerung ist demnach korrekt. Gleiches gilt nicht für die Längen der Vektoren. Diese werden mit einem sinnvoll gewählten α skaliert. [Buso4] schlägt eine Berechnung des α's wie in Gleichung 3.2.1.2 dargestellt vor.

$$\alpha = \frac{\langle \vec{e}, JJ^T\vec{e} \rangle}{\langle JJ^T\vec{e}, JJ^T\vec{e} \rangle} \tag{3.2.1.2}$$

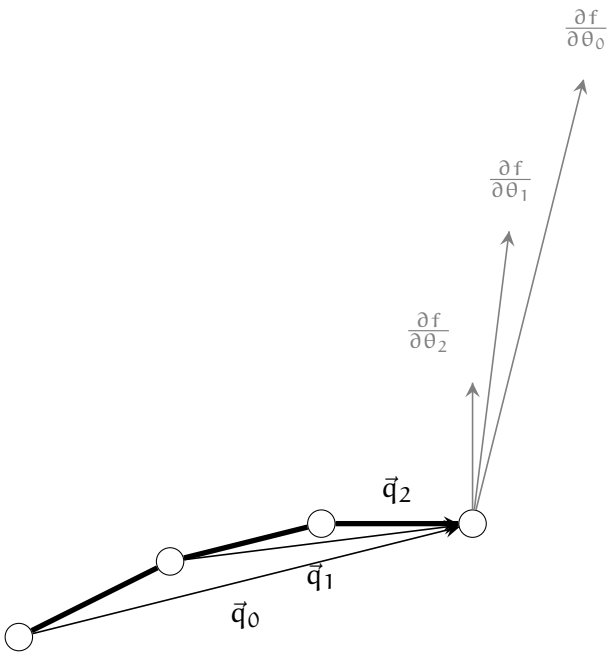

Abbildung 6: Veranschaulichung der Jacobi-Matrix in der Nähe von Singularitäten. $\vec{a}$ zeigt für alle Gelenke in die positive z-Richtung (aus der Seite heraus).

Begründet wird diese Wahl des α's mit der Annahme, dass die Änderung der End-Effektorpossition genau $\alpha \mathbf{J}\mathbf{J}^T \vec{e}$ ist und das α so gewählt wird, dass $\alpha \mathbf{J}\mathbf{J}^T \vec{e}$ möglichst nahe an $\vec{e}$ liegt.

3.2.2 *Pseudoinverse Jacobi-Matrix*

Die Pseudoinverse $\mathbf{J}^\dagger$, oder auch Moore-Penrose Inverse, ist eine algemeinere Form der Inversen, die auch auf nicht-quadratische Matrizen angewand werden kann. Wird sie wie in Gleichung 3.2.2.1 zur Lösung linearer Gleichungssysteme genutzt, liefert sie eine Lösung für $\mathbf{J}\Delta\theta = \vec{e}$ die im Sinne der Methode der kleinsten Quadrate möglichst nahe an der Zielposition liegt [Buso4], also die Differenz $\mathbf{J}\Delta\theta - \vec{e}$ minimiert.

$$\Delta\theta = \mathbf{J}^\dagger \vec{e} \tag{3.2.2.1}$$

Der Algorithmus für die Pseudoinverse ist in [Nilo9] wie folgt beschrieben.

Von Gleichung 3.2.0.1 wird Gleichung 3.2.2.2 abgeleitet.

$$\mathbf{J}^T\mathbf{J}\Delta\theta = \mathbf{J}^T\vec{e} \tag{3.2.2.2}$$

und nach $\Delta\theta$ aufgelöst:

$$\Delta\theta = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\vec{e} \tag{3.2.2.3}$$

$\mathbf{J}^T\mathbf{J}$ ist im Falle des NAO-Arms eine 5×5 Matrix, die invertiert werden muss.

3.3 ANALYTISCHES VERFAHREN

Beim analytischen Verfahren werden die Winkel der Gelenke direkt berechnet. In [KOL] wird der Prozess der Analyse für den NAO in der Version 3.3 anhand von 7 Schritten beschrieben:

1. Aufstellen der numerischen Transformationsmatrix zum Zielpunkt
2. Aufstellen der symbolischen Transformationsmatrix durch die Kette
3. Gleichsetzen der beiden Matrizen
4. Vereinfachen durch Manipulation beider Seiten
5. Über Geo- und Trigonometrie versuchen, Winkel zu ermitteln
6. Das Gleichungssystem für die verbleibenden Winkel auflösen
7. Mögliche Lösungen mit der Forward Kinematic überprüfen

Das Aufstellen der numerischen Transformationsmatrix besteht darin, dass zwei Vektoren $\vec{t}$ und $\vec{r}$ gegeben sind die den Zielpunkt beschreiben. $\vec{t}$ bezeichnet dabei die Zielposition und $\vec{r}$ die Zielausrichtung. Die Transformationsmatrix wird dann berechnet aus

$$\mathbf{A} = \mathbf{T}_{\vec{t}}\mathbf{R}_{z,r_z}\mathbf{R}_{y,r_y}\mathbf{R}_{x,r_x} \tag{3.3.0.1}$$

Die von [KOL] erarbeitete Lösung der Inversen Kinematik des linken Armes ist nachfolgend aufgeführt:

$$\theta_4 = -\left(\pi - \arccos\left(\frac{l_3^2 + l_4^2 - \sqrt{(s_x - \mathbf{A}_{(1,4)})^2 + (s_y - \mathbf{A}_{(2,4)})^2 + (s_z - \mathbf{A}_{(3,4)})^2}^2}{2 l_3 l_4}\right)\right) \tag{3.3.0.2}$$

$$\theta_2 = \pm \arccos\left(\frac{\mathbf{A}_{(2,4)} - l_1 - \left(\frac{l_4 \sin\theta_4 \mathbf{A}_{(2,2)}}{\cos\theta_4}\right)}{l_3 + l_4\cos\theta_4 + l_4\frac{\sin^2\theta_4}{\cos\theta_4}}\right) + \frac{\pi}{2} \tag{3.3.0.3}$$

$$\theta_3 = -\arcsin\left(\frac{\mathbf{A}_{(2,3)}}{\sin(\theta_2 - \frac{\pi}{2})}\right) \tag{3.3.0.4}$$

$$\theta_3 = -\left(\pi - \arcsin\left(\frac{\mathbf{A}_{(2,3)}}{\sin(\theta_2 - \frac{\pi}{2})}\right)\right) \tag{3.3.0.5}$$

Wenn $\theta_3 \neq |\pi/2|$:

$$\theta_1 = \pm \arccos\left(\frac{\mathbf{A}_{(3,3)} + \frac{\mathbf{A}_{(1,3)}\sin(-\theta_3)\cos(\theta_2 - \frac{\pi}{2})}{\cos(-\theta_3)}}{\cos(-\theta_3) + \frac{\cos^2(\theta_2 - \frac{\pi}{2})\sin^2(-\theta_3)}{\cos(-\theta_3)}}\right) \tag{3.3.0.6}$$

Wenn $\theta_3 = |\pi/2|$ und $\theta_2 \neq 0$:

$$\theta_1 = \pm \arccos\left(\frac{\mathbf{A}_{(1,3)}}{\cos(\theta_2 - \frac{\pi}{2})\sin(-\theta_3)}\right) \tag{3.3.0.7}$$

Wenn $\theta_3 = |\pi/2|$ und $\theta_2 = 0$:

$$\widehat{\mathbf{A}} = \mathbf{A}(\mathbf{T}_4^{\text{End}})^{-1}$$

$$\theta_1 = \pm \arccos\left(\frac{\widehat{\mathbf{A}}_{(1,4)}}{l_3}\right) \tag{3.3.0.8}$$

mit

s_x = x-Komponente des Vektors von der Schulter zur Zielposition
s_y = y-Komponente des Vektors von der Schulter zur Zielposition
s_z = z-Komponente des Vektors von der Schulter zur Zielposition
l_1 = ShoulderOffsetY + EllbowOffsetY
l_2 = ShoulderOffsetZ
l_3 = UpperArmLength
l_4 = HandOffsetX + LowerArmLength
$\mathbf{A}$ = siehe Gleichung 3.3.0.1
$\mathbf{A}_{(i,j)}$ = Element (i,j) der Matrix $\mathbf{A}$

Die kinematische Kette des rechten Armes ist aufgrund der Symetrie fast identisch zur Kette des linken Arms. Die Lösung der Inversen Kinematik nach [KOL]:

$$\mathbf{A}' = \mathbf{A}_{\text{rhandUnrotated}} = \mathbf{A}(\mathbf{R}_{z,-\pi})^{-1} \tag{3.3.0.9}$$

Die letze Rotation in der Matrix des rechten Armes (3.1.0.2) wird weggelassen um das Problem zu vereinfachen.

$$\theta_4 = \pi - \arccos\left(\frac{l_3^2 + l_4^2 - \sqrt{(s_x - \mathbf{A}'_{(1,4)})^2 + (s_y - \mathbf{A}'_{(2,4)})^2 + (s_z - \mathbf{A}'_{(3,4)})^2}^2}{2 l_3 l_4}\right) \tag{3.3.0.10}$$

$$\theta_2 = \pm \arccos\left(\frac{-\mathbf{A}'_{(2,4)} - l_1 - \left(\frac{l_4 \sin\theta_4 \mathbf{A}'_{(2,2)}}{\cos\theta_4}\right)}{l_3 + l_4\cos\theta_4 + l_4\frac{\sin^2\theta_4}{\cos\theta_4}}\right) - \frac{\pi}{2} \tag{3.3.0.11}$$

$$\theta_3 = -\arcsin\left(\frac{\mathbf{A}'_{(2,3)}}{\sin(\theta_2 + \frac{\pi}{2})}\right) \tag{3.3.0.12}$$

$$\theta_3 = \pi - \arcsin\left(\frac{\mathbf{A}'_{(2,3)}}{\sin(\theta_2 + \frac{\pi}{2})}\right) \tag{3.3.0.13}$$

Wenn $\theta_3 \neq |\pi/2|$:

$$\theta_1 = \pm \arccos\left(\frac{\mathbf{A}'_{(3,3)} + \frac{\mathbf{A}'_{(1,3)} \sin(\theta_3)\cos(\theta_2+\frac{\pi}{2})}{\cos(\theta_3)}}{\cos(\theta_3) + \frac{\cos^2(\theta_2+\frac{\pi}{2})\sin^2(\theta_3)}{\cos(\theta_3)}}\right) \tag{3.3.0.14}$$

Wenn $\theta_3 = |\pi/2|$ und $\theta_2 \neq 0$:

$$\theta_1 = \pm \arccos\left(\frac{\mathbf{A}'_{(1,3)}}{\cos(\theta_2 + \frac{\pi}{2})\sin(\theta_3)}\right) \tag{3.3.0.15}$$

Wenn $\theta_3 = |\pi/2|$ und $\theta_2 = 0$:

$$\widehat{\mathbf{A}} = \mathbf{A}'(\mathbf{T}_4^{\text{End}})^{-1}$$

$$\theta_1 = \pm \arccos\left(\frac{\widehat{\mathbf{A}}_{(1,4)}}{l_3}\right) \tag{3.3.0.16}$$

Die Symbole haben die gleiche Bedeutung wie bei der Inversen Kinematik des linken Arms.

3.4 FABRIK

Forward and Backward Reaching Inverse Kinematics (FABRIK) nach [AL11] ist ein iterativer Algorithmus, der durch das Berechnen von Punkten auf Linien die Gelenkpositionen der kinematischen Kette approximiert. Nachdem die Position des End-Effektors hinreichend genau bestimmt wurde, können die Winkel der Gelenke über die Forward Kinematik zurückgerechnet werden.

FABRIK besteht aus zwei Teilen: dem Forward Reaching und dem Backward Reaching. Beim Forward Reaching (Abbildung 7 grün) wird sofort der End-Effektor auf die Zielposition gesetzt und dann die Positionen in der Kette rückwärts angepasst. Die jeweils neue Position für das ausgewählte Gelenk liegt auf der Geraden, die durch das vorherig berechnete Gelenk und das ausgewählte Gelenk gelegt wird. Auf dieser Geraden wird das Gelenk dann so positioniert, dass die Entfernung zum vorher berechneten Gelenk der jeweiligen Gliedlänge entspricht. Die neu berechnete Position dient dann als Berechnungsgrundlage für das nächste Gelenk.

Nach dem Forward Reaching ist die im Raum fixierte Basis der Kette (p_0) verändert und muss auf die ursprüngliche Position zurückgelegt werden. Für diese Aufgabe wird beim Backward Reaching (Abbildung 7 blau) die Richtung des Forward Reachings umgekehrt, die

Basis also als End-Effektor betrachtet und als Ziel die ursprüngliche Position der Basis gesetzt. Der restlichte Ablauf ist dem des Forward Reachings identisch. Nach dem Backward Reaching ist dann die Basis wieder auf der ursprünglichen Position, der End-Effektor ist jedoch gegenüber der Zeilposition um eine Abweichung verschoben. Sowohl Forward als auch Backward Reaching werden so oft ausgeführt, bis die Abweichung vom Zielpunkt hinreichend klein ist.

Da bei der Ermittlung der Gelenkpositionen keine komplizierten Berechnungen auftreten, oder Matrixoperationen durchgeführt werden, ist ein Interationsschritt verhältnismäßig schnell und robust gegenüber Fehlern. So treten während der Iteration keine Singularitäten auf. Zusätzlich ist die gelieferte Lösung konkret und die Pose wirkt natürlich.

Die Einhaltung der Winkelbegrenzung muss während jedem Iterationsschritt sicher gestellt werden. Wird eine Gelenkposition berechnet, muss dessen Winkel überprüft und gegebenenfalls korrigiert werden. Der Winkel des Gelenkes lässt sich über die aus der Vektorrechnung bekannten Gleichung (3.4.0.1) berechnen.

$$\cos\theta_i = \frac{\vec{d}_i \cdot \vec{d}_{i+1}}{|\vec{d}_i| \cdot |\vec{d}_{d+1}|} \tag{3.4.0.1}$$

$\vec{d}_x$ bezeichnet die jeweilige Gliedlänge für das Gelenk p_x. Nach der Korrekturrotation wird die Iteration normal fortgesetzt.

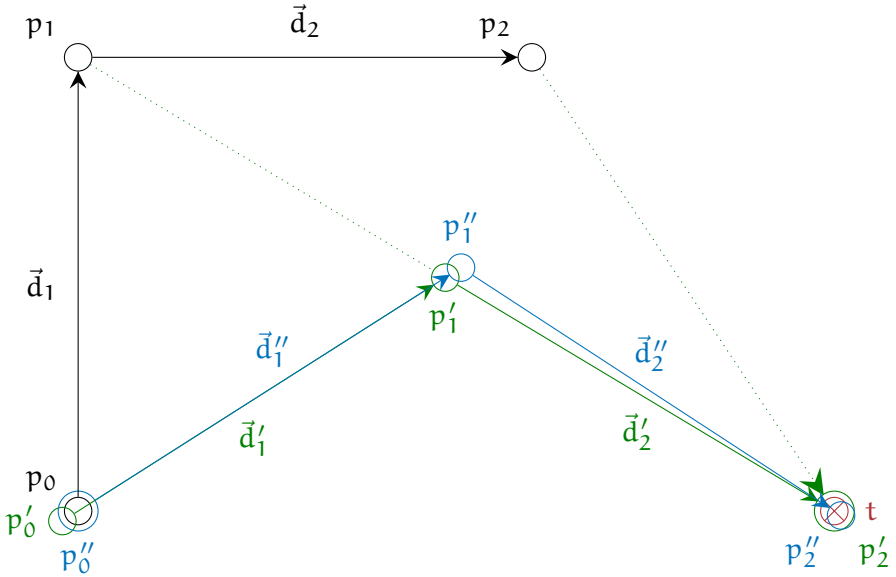

Abbildung 7: Beispiel eines Iterationsschrittes der Forward and Backward Reaching Inverse Kinematics (FABRIK). p_0, p_1 und p_2 sind Gelenke eines Armes. p_0 bezeichnet das räumlich fixierte Basisgelenk. p_2 ist der End-Effektor. Forward Reaching in grün, Backward Reaching in blau. Die Zielposition t ist rot.

Algorithm 1 FABRIK nach [AL11]

```
1: ▷ Parameter:
2: ▷ p_i ist die Liste der Gelenkpositionen mit i = 1,...,n
3: ▷ t ist der Zielpunkt
4: ▷ d ist der Gelenkabstand d_i = |p_{i+1} − p_i| mit i = 1,...,n
5: function FABRIK(p, d, t)
6:     ▷ prüfe, ob der Zielpunkt in Reichweite ist
7:     if |p_1 − t| > d_1 + d_2 + ... + d_{n−1} then
8:         ▷ Ziel ist nicht in Reichweite
9:         for i = 1,...,n − 1 do
10:            ▷ berechne die Entfernung r_i zwischen dem Ziel t
                 und der Gelenkposition p_i
11:            r_i ← |t − p_i|
12:            λ_i ← d_i/r_i
13:            ▷ berechne neue Gelenkposition p_i
14:            p_{i+1} ← (1 − λ_i)p_i + λ_i t
15:        end for
16:    else
17:        ▷ Ziel ist in Reichweite
18:        b ← p_1
19:        ▷ prüfe, ob die Entfernung zwischen p_n und t größer als
             die gegebene Toleranz tol ist
20:        dif_A ← |p_n − t|
21:        while dif_A > tol do
22:            ▷ Teil 1: Forward Reaching
23:            ▷ setze p_n als neues Ziel
24:            p_n ← t
25:            for i = n − 1,...,1 do
26:                ▷ berechne Entfernung r_i zwischen der neuen Ge-
                     lenkposition p_{i+1} und p_i
27:                r_i ← |p_{i+1} − p_i|
28:                λ_i ← d_i/r_i
29:                ▷ Berechne die neue Gelenkposition p_i
30:                p_i = (1 − λ_i)p_{i+1} + λ_i p_i
31:            end for
32:            ▷ Teil 2: Backward Reaching
33:            ▷ setze die Basis p_1 zurück
34:            p_1 ← b
35:            for i = 1,...,n-1 do
36:                ▷ berechne Entfernung r_i zwischen der neuen Ge-
                     lenkposition p_i und p_{i+1}
37:                r_i ← |p_{i+1} − p_i|
38:                λ_i ← d_i/r_i
39:                ▷ Berechne die neue Gelenkposition p_i
```

```
            p_i = (1 − λ_i)p_i + λ_i p_{i+1}
        end for
        dif_A ← |p_n − t|
    end while
  end if
  return p
end function
```

3.5 VERGLEICH

In diesem Abschnitt sollen die drei vorgestellten Methoden anhand der Kategorien *Berechnungszeit*, *Flexibilität* und *Stabilität* verglichen werden. Dabei ist die *Berechnungszeit* die Zeit, die die Methode zum Finden der Gelenkwinkel benötigt. Die *Flexibilität* sagt aus, wie gut sich die Methode auf eine andere kinematische Kette erweitern lässt und wie gut sie mit mehreren End-Effektoren gleichzeitig zurecht kommt. Bei der *Stabilität* geht es darum, eine Aussage über das Konvergenzverhalten an kritischen Positionen zu machen. Kritische Positionen sind in diesem Fall die Ränder der Reichweite der kinematischen Kette.

- Die analytische Lösung des Problems inverse Kinematik bietet aufgrund der direkten Berechnung die kleinste Berechnungszeit und sehr hohe Stabilität, ist aber durch die manuelle Ableitung der Gleichungen für die Gelenkwinkel sehr unflexibel.

- FABRIK ist nach [AL11] der deutlich schnellere von den hier vorgestellten iterativen Methoden, aber aufgrund der notwendigen Iterationen langsamer als die direkte analytische Lösung. Abgesehen von der überdurchschnittliche kleinen Berechnungszeit ist der Hauptvorteil des FABRIK die sehr hohe Flexibilität.

- Die Berechnungen über die Jacobi-Matrix sind im Vergleich zum FABRIK sehr langsam. [AL11] gibt eine 1000 mal längere Berechnungszeit an. Dies ist primär der hohen Anzahl an Iterationen geschuldet, die mit der Linearisierung zustande kommen. Zusätzlich zur langen Berechnungszeit leiden die Berechnungen über die Jacobi-Matrix, aufgrund der Verwendung von Inversen, unter Stabilitätsproblemen in der Nähe von Singularitäten. Im Gegensatz zur analytischen Methode ist die Jacobi-Matrix aber sehr flexibel.

4

AUSBLICK UND AUSWERTUNG

4.1 AUSWERTUNG

Für eine Anwendung der inversen Oberkörper-Kinematik im Rahmen des Roboterfußballs müssen und können Abstriche in der Flexibilität der Methode vorgenommen werden, um Ressourcen für wichtigere Berechnungen frei zu lassen. Sollte eine inverse Oberkörper-Kinematik für den Fußball implementiert werden, wäre die von [KOL] aufgestellte analytische Methode die beste Wahl um einfache Aufgaben zu erfüllen.
Für die Öffentlichkeitsarbeit könnte sich aber eine flexibelere Methode besser eignen, da mit ihr einfacher mehr Bewegungen konstruiert werden können. Der von [AL11] beschriebene FABRIK Algorithmus ist der Jacobi-Matrix mindestens ebenbürtig und käme daher für eine Implementierung eher in Frage.
Der FABRIK wäre auch der beste Kandidat für eine Erweiterung auf den Unterkörper zu einer kompletten inversen Kinematik des NAO's. Dies würde die Möglichkeit schaffen, kinematische Ketten über mehrere Gliedmaßen zu erstrecken. Einsatzgebiete dafür wären lange Schritte oder das Strecken zu einem Punkt mit dem ganzen Körper.
Alternativ lässt sich zu Gunsten der Flexibilität auch die in [KOL] zur Verfügung gestellte komplette analytische Lösung der inversen Kinematik implementieren. Hier wäre das Bezugskoordinatensystem mit dem Oberkörper aber festgelegt und kann nicht auf ein anderes Gelenk verschoben werden.

4.2 AUSBLICK

Abgesehen von den hier vorgestellten drei Methoden zur Berechnung der Gelenkwinkel gibt es natürlich noch viele andere. Zum Teil sind es Methoden die auf der Jacobi-Matrix aufbauen oder diese erweitern. Vertreter dieser Klasse sind die *"Jacobian (Selective) Damped Least Squares ((S)DLS)"* und *"Jacobian of the Singular Value Decomposition (SVD)"*. Das *"Follow the Leader (FTL)"* und der *"Cyclic Coordinate Descent (CCD)"* sind Algorithmen die dem Ziehen an einem Seil nachempfunden sind, der End-Effektor also in Richtung Zielposition gezogen wird. Diese Algorithmen sind aber entweder Variationen der hier vorgestellten Verfahren, oder sind den Vorgestellten unterlegen. Der nächste Schritt wäre die Implementierung der gewählten Algortihmen und die Ausweitung auf die gesamte Kinematik des NAO und der Entwurf einer Schnittstelle zur Echtzeitsteuerung.

LITERATURVERZEICHNIS

[AL11] Aristidou, Andreas ; Lasenby, Joan ; Department of Engineering University of Cambridge (Hrsg.): *FABRIK: A fast, iterative solver for the Inverse Kinematics problem*. Cambridge CB2 1PZ, UK: Department of Engineering University of Cambridge, 2011

[AR12] Aldebaran-Robotics: *Aldebaran Documentation*. http://doc.aldebaran.com/2-1/home_nao.html. Version: 2006-2012. – [Online, letzter Besuch März 2016]

[Bus04] Buss, Samuel R. ; Department of Mathematics, University of California, San Diego (Hrsg.): *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods*. 9500 Gilman Dr, La Jolla, CA 92093, United States: Department of Mathematics, University of California, San Diego, 2004

[DH55] Denavit, J. ; Hartenberg, R.S.: *A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices*. ASME J. of Appl. Mechanics, 1955

[GPS06] Goldstein, Herbert ; Poole, Charles P. ; Safko, John L.: *Klassische Mechanik*. 3. Auflage. Wiley-VCH, 2006. – ISBN 3527405895

[KOL] Kofinas, Nikos ; Orfanoudakis, Emmanouil ; Lagoudakis, Michail G. ; Intelligent Systems Laboratory, Department of Electronic and Computer Engineering, Technical University of Crete (Hrsg.): *Complete Analytical Inverse Kinematics for NAO*. Chania, Crete 73100, Greece: Intelligent Systems Laboratory, Department of Electronic and Computer Engineering, Technical University of Crete

[Nil09] Nilsson, Rickard: *Inverse Kinematics*, Luleå University of Technology, Diplomarbeit, 2009

[SB96] Siegert, Hans-Jürgen ; Bocionek, Siegfried: *Robotik: Programmierung intelligenter Roboter*. Springer-Verlag Berlin Heidelber, 1996. – ISBN 3–540–60665–3

[TT09] Tira-Thompson, Ethan: *Sample Denavit-Hartenberg Diagram*. https://commons.wikimedia.org/wiki/File:Sample_Denavit-Hartenberg_Diagram.svg. Version: 2009. – [Online, letzter Besuch März 2016]

[Wol11] Wollnack, Jörg: *Vorlesungsskript Robotik*. Technische Universität Hamburg-Harburg, 2011