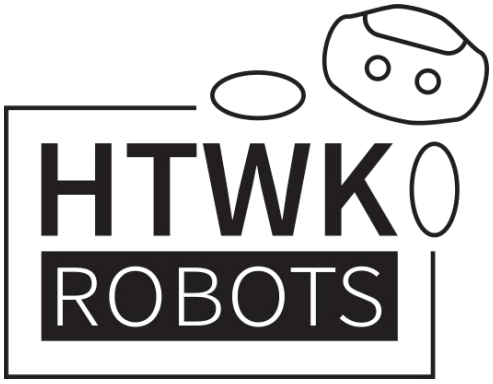
Hochschule für Technik,
Wirtschaft und Kultur Leipzig



Bachelorarbeit
zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

im Bachelorstudiengang Informatik
der Fakultät Informatik und Medien

Stabilisierung von humanoiden Robotern mithilfe dynamischer Korrekturen durch Vorwärtskinematik

vorgelegt von
Eric Behrendt

Leipzig, den 21. November 2024

Erstprüfer: Prof. Dr. rer. nat. Mario Hlawitschka
Zweitprüfer: M.Sc. Tobias Jagla

Inhaltsverzeichnis

1 Einleitung

Mobile Robotik und Künstliche Intelligenz ermöglichen neue spannende Möglichkeiten für den Einsatz autonomer Systeme in unserer Umwelt. Mobile Roboter haben das Potenzial, sich selbstständig in verschiedenen Umgebungen zu bewegen und können dabei auf dynamische Veränderungen reagieren und komplexe Aufgaben lösen. Dies ist in Bereichen wie der Logistik, Landwirtschaft, dem Gesundheitswesen und noch vielen weiteren relevant [1].

1.1 Motivation

Roboterfußball bietet eine hervorragende Plattform für Forschungen in den Bereichen Künstliche Intelligenz und mobile Robotik. Beim RoboCup treten deshalb jedes Jahr Teams aus aller Welt in dieser Disziplin gegeneinander an und tauschen sich über Fortschritte aus. Um zu gewinnen, müssen viele verschiedene Teilbereiche gut zusammen funktionieren und interagieren, wie beispielsweise die Bildverarbeitung, Strategie und Motorik. Dabei ist es besonders wichtig, einen stabilen Lauf zu haben, damit keine Chancen verschenkt werden, weil ein Roboter hinfällt. Gelegentliches Hinfallen ist aufgrund der komplexen Umgebung jedoch nicht vermeidbar. Deshalb ist es ebenfalls wichtig, schnell und zuverlässig wieder aufzustehen. Jede Sekunde, die mit Aufstehen verbracht wird, bringt dem Gegner einen Vorteil, da der gefallene Roboter nicht aktiv in das Spielgeschehen eingreifen kann. Die Aufstehbewegung aus der Rückenlage dauert bei den HTWK-Robots ca. 6.6 s, was bei einer maximalen Laufgeschwindigkeit von ca. $30\,\frac{\mathrm{cm}}{\mathrm{s}}$ einen Unterschied von knapp zwei Metern bedeuten kann. Wenn die Aufstehbewegung zusätzlich instabil ist, kann das dazu führen, dass der Roboter in der Bewegung wieder umkippt und von vorne beginnen muss. Das führt zu weiterem Zeitverlust. Auch kann häufiges Stürzen zu beschädigten Gelenken bei den Robotern führen. Dies ist neben dem finanziellen Schaden auch im Spiel ein Problem, da unter Umständen Spiele in Unterzahl gespielt werden müssen, wenn sich mehrere Roboter in der Reparatur befinden. Somit führen eine langsame und instabile Aufstehbewegung zu erheblichen Nachteilen in Spielen.
Dies war besonders auf dem letzten RoboCup 2024 in Eindhoven spürbar. Dort sind die

HTWK-Robots 604-mal hingefallen, was fast das Vierfache der amtierenden Weltmeister B-Human ist [2]. Dabei wurden rund 46 % der Aufstehversuche bei den HTWK-Robots wieder abgebrochen. Des Weiteren benötigt das Aufstehen des auf dem Rücken liegenden HTWK-Roboters im Vergleich zu B-Human ca. 80 % mehr Zeit [3].
Es ist also erkennbar, dass beim Aufstehen der HTWK-Robots in vielen Bereichen noch Verbesserungspotenzial besteht.

1.2 Ziele und Abgrenzung

In dieser Arbeit soll ein Verfahren entwickelt werden, das die Aufstehbewegungen stabilisiert und unempfindlicher gegenüber Störfaktoren macht. Dazu soll die aktuelle Position des Roboters während der Bewegung analysiert werden, um Probleme zu erkennen. Diese Probleme sollen anschließend ausgeglichen beziehungsweise korrigiert werden.
Außerdem soll es möglich sein, dieses Verfahren einfach und schnell auf andere Bewegungen zu übertragen. Diese Arbeit konzentriert sich auf die Aufstehbewegung aus der Rückenlage, da es dort bei den HTWK-Robots das größte Verbesserungspotenzial gibt.
Es ist nicht das Ziel dieser Arbeit, eine komplett neue Aufstehbewegung zu entwickeln.

1.3 Aufbau

In Kapitel 2 werden grundlegende Informationen zum RoboCup, dessen Ligen und dem in dieser Arbeit verwendeten humanoiden Roboter NAO erläutert. Darüber hinaus werden wichtige Begriffe erklärt und es wird ein Einblick in die aktuelle Forschungslage des Themas gegeben. Kapitel 3 behandelt die aktuelle Umsetzung der Aufstehbewegungen im HTWK-Code. Außerdem werden die daraus resultierenden Probleme abgeleitet. Anschließend geht es in Kapitel 4 um die Vorstellung und Implementierung des Ansatzes dieser Arbeit. Eine Evaluation, in der verschiedene Situationen betrachtet werden, um zu beurteilen, wie gut der Ansatz funktioniert, folgt in Kapitel 5. Zuletzt werden in Kapitel 6 die Ergebnisse der Arbeit zusammengefasst und ein Ausblick auf mögliche Weiterentwicklungen gegeben.

2 Grundlagen

Dieses Kapitel bietet eine Einführung in die wesentlichen Begriffe und Konzepte, die für diese Arbeit relevant sind. Dafür werden zunächst der RoboCup und die *Standard Platform League* (SPL) vorgestellt. Anschließend werden wichtige Begriffe erläutert, welche in der weiteren Arbeit genutzt werden um das entwickelte Verfahren zu beschreiben und umzusetzen. Zuletzt wird ein Überblick über verwandte Arbeiten gegeben, um verschiedene Ansätze zur Verbesserung der Balance humanoider Roboter aufzuzeigen.

2.1 RoboCup

Der RoboCup ist ein seit 1997 jährlich von der RoboCup Federation ausgerichteter Wettbewerb, in dem Teams aus aller Welt in verschiedenen Ligen und Disziplinen gegeneinander antreten. Das Ziel ist dabei, die Forschung in den Bereichen Künstliche Intelligenz und Robotik zu fördern. Innerhalb des RoboCups gibt es verschiedene Disziplinen. So wird bei *RoboCup@Home* an Robotern für den Alltagsgebrauch geforscht, bei *RoboCupRescue* an Robotern für Rettungsmissionen in schwierigem Gelände und bei *RoboCupIndustrial* an Robotern für Industrie und Arbeit. Die größte Disziplin des RoboCups ist allerdings der Roboterfußball.
Nachdem 1997 *Deep Blue* als erster Computer den menschlichen Schachweltmeister besiegt hatte, wurde Fußball als neues Standardproblem erkannt. Im Gegensatz zu Schach muss im Fußball auf ein dynamisches Umfeld mit unvollständigen Informationen in Echtzeit reagiert werden. Dies stellt komplett neue Probleme und Forschungsansätze dar, für die Lösungen entwickelt werden müssen. Das große Ziel des RoboCups ist es, dass bis zur Mitte des 21. Jahrhunderts ein Team aus völlig autonomen humanoiden Robotern ein Fußballspiel nach den offiziellen Regeln der FIFA gegen den Sieger der letzten Weltmeisterschaft gewinnt. Dazu gibt es innerhalb dieser Roboterfußballdisziplin mehrere Ligen, in denen verschiedene Forschungsbereiche betont werden. Die für diese Arbeit relevante Liga ist die *Standard Platform League* (SPL), an der auch das HTWK-Robots-Team teilnimmt [4, 5].

2.1.1 Standard Platform League

In der SPL spielen zwei Teams mit jeweils sieben Robotern auf einem 9 m × 6 m Kunstrasenfeld gegeneinander. Die Roboter werden an den Spielfeldrand gesetzt und müssen vom Einlaufen bis zum Schlusspfiff komplett autonom agieren, wobei sie lediglich untereinander kommunizieren können. Ein Spiel dauert zwei Halbzeiten à 10 Minuten. Bei einem Unentschieden in K.o.-Runden-Spielen wird der Gewinner durch ein Elfmeterschießen bestimmt. Außerdem gibt es Zeitstrafen bei verschiedenen Regelverstößen, wie zum Beispiel einem Stoß gegen einen gegnerischen Roboter. Auch können die Roboter eine Zeitstrafe bekommen, wenn sie es nicht schaffen aufzustehen. Ein Roboter wird als nicht fähig aufzustehen gewertet, wenn er unbehindert zweimal scheitert, beziehungsweise dreimal, falls er behindert wurde. Wenn das der Fall ist, wird der Roboter für 45 Sekunden an den Rand gestellt.
Als Standard Plattform verwendet die Liga momentan den Roboter *NAO* von der Firma SoftBank Robotics, welcher in Abbildung 2.1 zu sehen ist. Da an den Robotern keine Verbesserungen vorgenommen werden dürfen, kann kein Vorteil durch hochwertigere Hardware entstehen. Allein die Software ist für den Erfolg eines Teams verantwortlich.
Um die Innovationen und Entwicklungen innerhalb der Liga voranzutreiben, werden die Regeln jährlich angepasst. Dadurch entstehen stets neue Herausforderungen, an denen die Teams arbeiten können [6].

2.1.2 NAO-Roboter

Der momentan in der SPL benutzte NAO ist bereits die sechste Version dieses Roboters. Er hat in etwa eine Höhe von 37.4 cm und wiegt circa 5.6 kg.
Um sich auf einem Fußballfeld zu orientieren, hat der NAO verschiedene Sensoren. Zu diesen gehören mehrere Berührungssensoren, vier Ultraschallsensoren, vier Mikrofone, zwei Kameras und eine Inertialeinheit (inertial measurement unit, IMU) bestehend aus Beschleunigungssensor und Gyroskop. Die Werte der IMU werden relativ zur Gravitation gemessen. Zur Verarbeitung aller Daten und Programme besitzt der NAO einen *Atom E2845* 1.91 GHz Prozessor und 4 GB DDR3-Speicher.
Die Motoren der 25 Freiheitsgrade (Degrees of Freedom, DOF) des Roboters können mit einer Frequenz von 83 Hz angesteuert werden. Bei der Ansteuerung können Werte für die Winkelpositionen und Festigkeit der Motoren übergeben werden. Die Festigkeit hat einen Wert zwischen Null und Eins, wobei das Gelenk bei null frei bewegbar ist, während bei einem Wert von Eins das volle Drehmoment des Motors genutzt wird, um den spezifizier-

ten Winkel zu erreichen und zu halten. In Abbildung 2.1 ist der Roboter mit allen DOF zu sehen [7].

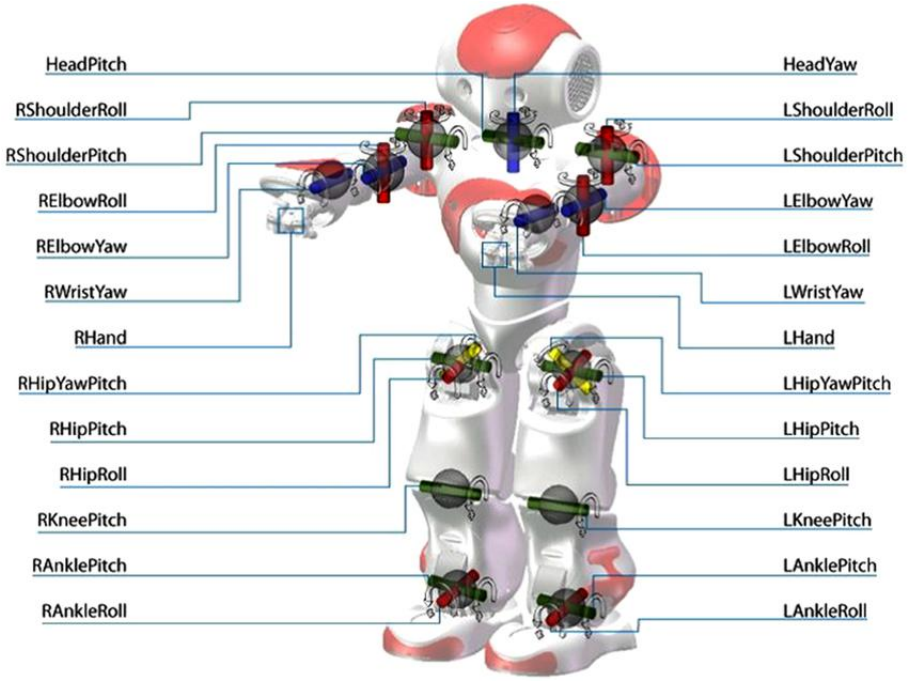


Abbildung 2.1: Abbildung des NAO v5. Die Gelenkanordnung und Bezeichnungen stimmen mit dem NAO v6 überein.[1]

2.2 Technologien und Begriffserklärungen

Nachdem die Grundlagen des RoboCups erklärt wurden, müssen nun noch wichtige Begriffe und Technologien eingeführt werden. Diese werden in der weiteren Arbeit zur Stabilisierung und Bewegungssteuerung des Roboters genutzt.

2.2.1 Vorwärtskinematik

Das Ziel der Vorwärtskinematik ist es, die Position und Orientierung eines Endeffektors zu berechnen. Ein Endeffektor ist das letzte Glied in einer kinematischen Kette, die den strukturellen Zusammenhang der Glieder und Gelenke beschreibt. Für die Berechnung werden die Winkel aller Gelenke genutzt, die zur entsprechenden kinematischen Kette gehören. Zusätzlich werden die Dimensionen der Glieder, die die Gelenke verbinden, sowie eine Basis benötigt. Die Basis ist die Startposition der Berechnungen, und die resultierende Position und Orientierung des Endeffektors sind relativ zu dieser Basis [8].

[1] Quelle: http://doc.aldebaran.com/2-1/_images/hardware_jointname.jpg, Zugriff: 11.11.2024, Rechteinhaber: Aldebaran - SAS (Limited Company)

2.2.2 Quaternionen

Quaternionen sind eine Erweiterung der komplexen Zahlen und können benutzt werden, um Rotationen im dreidimensionalen Raum zu beschreiben. Quaternionen sind folgendermaßen definiert:

$$\mathbb{H} \widehat{=} \left\{ q_w + \hat{i}q_x + \hat{j}q_y + \hat{k}q_z : q_w, q_x, q_y, q_z \in \mathbb{R} \right\}, \tag{2.1}$$

wobei gilt:

$$\hat{i}^2 = \hat{j}^2 = \hat{k}^2 = \hat{i}\hat{j}\hat{k} = -1 \tag{2.2}$$

und:

$$\hat{i}\hat{j} = \hat{k}, \quad \hat{j}\hat{k} = \hat{i}, \quad \hat{k}\hat{i} = \hat{j}. \tag{2.3}$$

Um eine Rotation um eine Achse, beschrieben durch den Einheitsvektor $\boldsymbol{u}$, mit einem Winkel ϕ als Quaternion darzustellen, kann folgende Formel genutzt werden:

$$r = \cos(\phi/2) + \sin(\phi/2)(\hat{i}\,\boldsymbol{u}_x + \hat{j}\,\boldsymbol{u}_y + \hat{k}\,\boldsymbol{u}_z). \tag{2.4}$$

Diese Formel liefert eine Einheitsquaternion. Das ist wichtig, da nur eine Einheitsquaternion eine 3D-Rotation darstellen kann. Eine Einheitsquaternionen ist durch folgende Eigenschaft bestimmt:

$$\|q\| = 1 \tag{2.5}$$

wobei:

$$\|q\| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} \tag{2.6}$$

Um aufeinanderfolgende Rotationen darzustellen, können die Quaternionen der einzelnen Rotationen miteinander multipliziert werden. Die Multiplikation bei Quaternionen ist nicht kommutativ und somit ist die Reihenfolge der Multiplikation relevant. Dabei bildet die zuerst ausgeführte Rotation den ersten Faktor, da sie die zweite Rotation beeinflusst. Die Multiplikation an sich ist dann folgendermaßen definiert:

$$qq' = (q_w + \hat{i}q_x + \hat{j}q_y + \hat{k}q_z)(q'_w + \hat{i}q'_x + \hat{j}q'_y + \hat{k}q'_z) \tag{2.7}$$

Die resultierende Quaternion repräsentiert eine Rotation, bei welcher ein Objekt zuerst um q und anschließend um q' rotiert wurde.
Um eine Rotation in die entgegengesetzte Richtung einer Quaternion q durchzuführen, kann die konjugierte Quaternion q^* genutzt werden. Die Konjugation ist dabei so definiert:

$$q^* = \mathrm{Re}(q) - \mathrm{Im}(q) \tag{2.8}$$

Hier steht $\mathrm{Re}(q)$ für den Realen Anteil von q und $\mathrm{Im}(q)$ für den imaginären Anteil.
Um einen beliebigen Vektor $\boldsymbol{v}$ um eine Achse $\boldsymbol{u}$ mit einem Winkel ϕ zu rotieren, muss $\boldsymbol{v}$ zunächst in eine pure Quaternion umgewandelt werden. Eine pure Quaternion ist eine, bei der der reale Anteil gleich Null ist. Als imaginärer Teil wird $\boldsymbol{v}$ genommen. Mit dieser puren Quaternion kann die Rotation folgendermaßen berechnet werden.

$$\boldsymbol{p} = \mathrm{Im}(rvr^*) \tag{2.9}$$

Der erhaltene Vektor $\boldsymbol{p}$ ist der rotierte Ortsvektor des ursprünglichen Vektors $\boldsymbol{v}$ [9, 10].

2.2.3 PID-Controller

Der Proportional-Integral-Derivative-Controller ist ein Regler, mit dem Korrekturen berechnet werden können, um einen Sollwert auch unter Störungen zu erreichen. Die Korrektur setzt sich aus einem proportionalen (P), einem integralen (I) und einem differentiellen (D) Anteil zusammen:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right) \tag{2.10}$$

Dabei beschreibt e den vorzeichenbehafteten Fehler zum Zeitpunkt t, K den Verstärkungsfaktor des P-Anteils, T_i die Zeitkonstante des I-Anteils und T_d die Zeitkonstante des D-Anteils. Der P-Anteil regelt also dem aktuellen Fehler entgegen. Der I-Anteil nutzt den kumulierten Gesamtfehler, sodass dieser Anteil wächst oder schrumpft, umso länger der kontrollierte Wert vom Sollwert abweicht. Zuletzt beschreibt der D-Anteil den Anstieg der Fehlerfunktion. Somit wirkt dieser dem D- und I-Anteil entgegen um eine Überkorrektur zu verhindern.
Gelegentlich wird statt K/T_i für den I-Anteil und KT_d für die D-Anteil jeweils ein eigener Verstärkungsfaktor genommen: K_i und K_d. Dadurch lässt sich die Gewichtung der drei Faktoren unabhängig voneinander einstellen. Damit ergibt sich:

$$u(t) = Ke(t) + K_i\int_0^t e(\tau)\,d\tau + K_d\frac{de(t)}{dt} \tag{2.11}$$

Eine Abwandlung des PID-Controllers ist der PI-Controller, bei dem lediglich der D-Anteil entfällt. Diese Form des Controllers wird auch in dieser Arbeit verwendet [11].

2.3 Verwandte Arbeiten

Nach einem Sturz wieder aufstehen zu können, ist in dynamischen Umgebungen, wie dem Roboterfußball, von grundlegender Bedeutung. Ein wichtiger Bestandteil dabei ist die Balance, um zu verhindern, dass der Roboter hinfällt oder um Bewegungen wie das Aufstehen stabil durchführen zu können. Aus diesen Gründen wird bereits seit vielen Jahren an der Balance von Humanoiden Robotern geforscht. Über die Zeit wurden verschiedene Ansätze und Strategien entwickelt, um dieses Problem zu lösen.
Um zuverlässig balancieren zu können, werden Metriken benötigt, nach denen beurteilt werden kann, ob ein Roboter in einer bestimmten Position aufrecht stehen bleibt oder hinfallen wird. Daran wird seit den frühen Arbeiten von Vukobratović und Juričić (1968) zur Balance [12] geforscht. Zwei der wichtigsten Metriken sind der Zero-Moment-Point (ZMP) und das Center of Pressure (CoP) [13].
Der ZMP, ist der Punkt innerhalb der Fuß-Ebene des Roboters, in der sich alle horizontalen Momente, die auf den Roboter einwirken, ausgleichen. Solange sich dieser Punkt innerhalb des Auflagepolygons des Roboters befindet, bleibt dieser aufrecht stehen. Das Auflagepolygon ist dabei durch die konvexe Hülle der Auflageflächen des Roboters beschrieben [14]. Auch beim RoboCup wird der ZMP von einigen Teams zur Stabilisierung von Bewegungen wie Schüssen oder beim Aufstehen genutzt [15, 16].
Das CoP hingegen beschreibt den Punkt, an dem die ground reaction force (GRF) einwirkt. Die GRF ist die Kraft, die ausgehend vom Boden dem Roboter entgegenwirkt. Solange der Roboter balanciert ist, stimmen der ZMP und das CoP überein [14].
Da es aufwendig ist, die kompletten dynamischen Prozesse eines Roboters zu berechnen, wird häufig ein dreidimensionales lineares invertiertes Pendel (LIP) mit einem konzentrierten Center of Mass (CoM) an dessen Spitze verwendet [17, 18, 19].
In dieser wird weder ein ZMP noch ein CoP benutzt, um den Roboter zu balancieren. Die Aufstehbewegung besteht bereits als eine statische im HTWK-Framework und sollte somit während des gesamten Verlaufs balanciert sein, sofern der Schwung des Roboters nicht mit beachtet wird. Um den Schwung mit einzubeziehen, könnte ein ZMP genutzt werden. Es sollte jedoch ausreichen, den Schwung zu korrigieren, indem die Winkelgeschwindigkeit des Roboters kontrolliert wird. Es wird auch kein LIP genutzt, weil sich diese Arbeit auf den Oberkörper konzentriert. Dieser hat durch die erhöhte Position einen stärkeren Einfluss auf das Moment des Roboters.
Um die genannten Metriken auszuwerten und um zu bestimmen, welche Maßnahmen zur Stabilisierung getroffen werden sollten, werden Balance Controller (BC) genutzt. Auch von diesen gibt es verschiedene Varianten. Mithilfe des Model Predictive Control (MPC) kann der Pfad des CoM so geplant werden, dass gewünschte Beschränkungen, wie z.B. dass der ZMP im Auflagepolygon bleibt, erfüllt sind. Für die Vorhersagen können verschie-

dene dynamische Modelle, wie ein LIP, aber auch vollständige dynamische Berechnungen genutzt werden. MPCs sind jedoch rechenaufwändig, was bei Onboard-Berechnungen, wie beim NAO, ein Problem sein kann [20]. Dennoch werden bei einigen Robotern MPCs erfolgreich zur Planung von Schritten beim Laufen verwendet [18, 21].
Ein anderer Ansatz basiert auf dem Drehimpuls des Roboters. Es wurde gezeigt, dass der Drehimpuls ein wichtiger Faktor für die Balancierung von Robotern sein kann. Der Drehimpuls um den CoM wird als Centroidal Angular Momentum (CAM) bezeichnet und ist dabei besonders wichtig [22]. Auch hier werden für die Modellierung sowohl LIPs [23, 24] als auch vollständige Dynamiken benutzt [25, 26, 27].
Ein verwandter Ansatz ist die Nutzung der Winkelgeschwindigkeit und des Körperwinkels des Roboters. Dadurch kann auf Abweichungen in der Position und auf externe Störungen reagiert werden. In [28] wird ein solcher Ansatz für den Roboter EROS beschrieben. Dort soll der BC zum einen ermöglichen Positionen zu halten und zum anderen Oszillationen und Stöße auszugleichen, während der Roboter läuft. Dafür wird der BC in zwei Teile geteilt: Pose Control und Angular Velocity Control. Im Pose Control wird der Winkel des Körpers genutzt. Falls dieser vom Zielwinkel abweicht, werden die Arme in die entgegengesetzte Richtung geschwenkt um dem abweichenden Moment entgegen zu wirken. Für die Bestimmung der Gelenkwinkel der Arme wird ein PID-Controller genutzt. Da sich der Oberkörper beim Laufen nicht bewegen soll, wird beim Angular Velocity Control ein Zielwert von Null angenommen. Wenn der tatsächliche Wert durch Stopps oder externe Einflüsse abweicht, regelt der Controller gegen.
In dieser Arbeit werden ebenfalls die Winkel und Winkelgeschwindigkeiten des Oberkörpers genutzt, um den Roboter zu balancieren. Dabei werden nur die *pitch*- und *roll*-Winkel mit einbezogen, da die *yaw* Orientierung für das Aufstehen keine Rolle spielt. Im Gegensatz zu [28] werden dafür jedoch statt eines PID-Reglers vier PI-Regler und nur die Beine zur Korrektur genutzt.
Zuletzt, wenn entschieden wurde wann und wie stark korrigiert wird, muss der BC noch entscheiden, mit welchen Gelenken diese Korrektur stattfinden soll. Bei Untersuchungen in der Biomechanik wurde festgestellt, dass Menschen unter anderem zwei wichtige Strategien, *Ankle Strategy* und *Hip Strategy*, nutzen, um bei Störungen die Balance zu halten [29]. Bei der *Ankle Strategy* werden ausschließlich die Fußgelenke zur Korrektur genutzt. Das funktioniert bei kleinen Abweichungen gut, da dadurch der CoP verschoben wird, was eine Korrektur des Körpers in die entgegengesetzte Richtung zur Folge hat. Wenn größere Korrekturen durchgeführt werden müssen, wird die benötigte Kraft durch den langen Hebel zu groß um mit den Fußgelenken zu korrigieren. In diesen Fällen kann die *Hip Strategy* genutzt werden. Dabei werden nur die Hüftgelenke genutzt, da durch die Rotation um die Hüfte der COM des Roboters schnell wieder über das Auflagepolygon gebracht werken kann [30].

Ein anderer Ansatz ist der Whole-Body-Control (WBC), wobei der gesamte Körper mit einbezogen wird, um den Roboter zu stabilisieren. Dadurch wird die Stabilisierung flexibler, da beispielsweise zusätzlich die Arme genutzt werden können, um den Roboter besser zu balancieren [31].
In dieser Arbeit wird hauptsächlich eine Kombination aus *Ankle* und *Hip Strategy* genutzt. Zusätzlich werden aber auch noch in bestimmten Situationen die Kniegelenke des Roboters verwendet, um die Last auf den anderen Gelenken zu reduzieren.

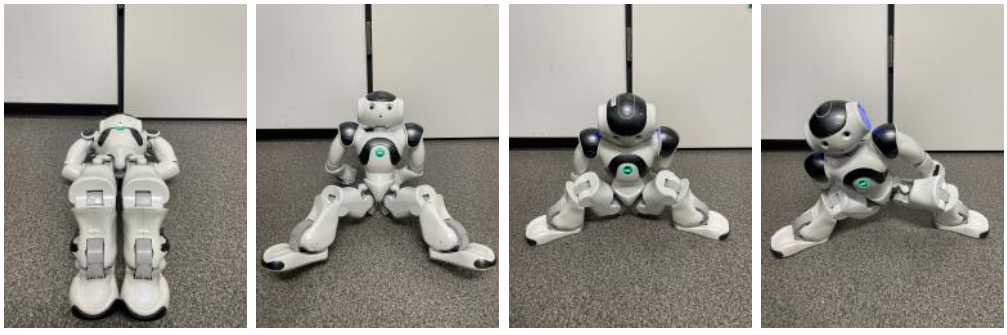
3 Aufstehbewegungen im HTWK-Framework

Bevor das in dieser Arbeit entwickelte Verfahren beschrieben werden kann, ist es wichtig zu verstehen, wie die Aufstehbewegungen aktuell im HTWK-Framework funktionieren. Daraus können Probleme abgeleitet werden, welche als Orientierung genutzt werden, um das Verfahren bewerten zu können.

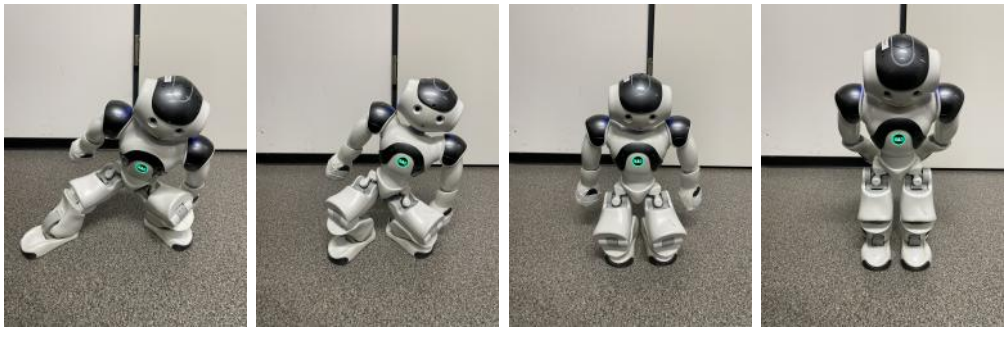
3.1 Aktuelle Implementation

Bei einer Bewegung werden aller 12 ms Zielwinkel an alle Gelenke gesendet. Das ergibt sich aus der in Abschnitt 2.1.2 erwähnten Ansteuerfrequenz von 83 Hz. Jeder dieser Zeitabschnitte wird hier als *Frame* bezeichnet. Dabei werden alle Frames aus einer Datei geladen, die die abgespeicherte Bewegung enthält. Die Bewegung beginnt bei Frame Null und endet, wenn alle Frames aus der Datei abgespielt wurden. Die Gelenkwinkel im HTWK-Framework sind somit zeitabhängig und beschreiben eine statische, vordefinierte Bewegung. Es gibt allerdings auch noch andere Ansätze, bei denen die Gelenkwinkel nicht nur allein von der Zeit abhängen, sondern durch eine mehrdimensionale Funktion bestimmt werden, bei der sowohl die Zeit als auch die aktuellen Gelenkwinkel eine Rolle spielen [32]. Die Ausgangsposition einer jeden Aufstehbewegung ist ein liegender Roboter und die Zielposition ein stehender. Der Weg vom Anfang zum Ziel, die eigentliche Bewegung, muss manuell mithilfe von Zwischenpositionen beschrieben werden. Im HTWK-Framework wird dafür ein Tool, der *MotionEditor*, genutzt. Darin können die Gelenke des Roboters fest oder locker gestellt werden. Während ein Gelenk locker ist, kann es frei bewegt werden. So kann ein Roboter in verschiedene Haltungen gebracht werden, und wenn eine gewünschte Position eingenommen ist, kann diese abgespeichert werden. Die so abgespeicherten Gelenkwinkel werden als *Keyframe* bezeichnet. Zu jedem Keyframe wird außerdem eine Frame-Nummer abgespeichert, welche definiert, zu welchem Zeitpunkt in der Bewegung der Keyframe erreicht werden soll. Mehrere Keyframes zusammen definieren durch ihre

Reihenfolge eine Bewegung. Für jeden Zeitpunkt zwischen den Keyframes werden die Gelenkwinkel interpoliert, indem zwischen dem Anfangswert und dem Zielwert ein linearer Verlauf angenommen wird. Diese Herangehensweise ist in der Robotik üblich und wird unter anderem für Aufstehbewegungen oder Schüsse genutzt [16, 33].
Im HTWK-Framework gibt es im Moment drei verschiedene Aufstehbewegungen. Es gibt eine Bewegung für den Fall, dass der Roboter nach vorne umgekippt ist, und zwei für den Fall, dass er nach hinten gefallen ist. Die Bewegungen heißen *getup_front*, *getup_back* und *getup_back_slow* und dauern 3.5s, 3s beziehungsweise 6.6s. Die Abläufe der beiden aus der Rückenlage beginnenden Bewegungen sind in den Abbildungen 3.1 und 3.2 dargestellt.
Falls eine Bewegung nicht korrekt ausgeführt wird, sollte diese abgebrochen und neu gestartet werden. Dafür geht der Roboter zunächst in die Hocke, um die Falldistanz zu verringern, und setzt die Festigkeit verschiedener Gelenke auf null, um die Last auf der Hardware zu minimieren. Dies schont den Roboter und sorgt dafür, dass keine weitere Zeit durch einen fehlgeschlagenen Versuch verloren wird. Für das Abbrechen wurden aus vergangenen Daten Durchschnittswerte für den Torso *pitch*- und *roll*-Winkel berechnet. Wenn der Roboter während der Bewegung zu stark von diesen Werten abweicht, deutet das darauf hin, dass etwas bei der Bewegung nicht in Ordnung ist, und der Vorgang wird abgebrochen und neugestartet.
Aktuell wird für einen nach hinten gefallenen Roboter nur die *getup_back_slow*-Bewegung genutzt, da *getup_back* zu instabil ist und zu häufigen Abbrüchen führt.



(A) $t = 0.54s$ (B) $t = 1.02s$ (C) $t = 1.2s$ (D) $t = 2.22s$



(E) $t = 3.52s$ (F) $t = 4.88s$ (G) $t = 6.07s$ (H) $t = 6.67s$

Abbildung 3.1: Zeitlicher Ablauf von *getup_back_slow*

(A) $t = 0.37s$ (B) $t = 0.77s$ (C) $t = 0.98s$ (D) $t = 1.2s$

(E) $t = 1.46s$ (F) $t = 2.35s$ (G) $t = 2.92s$

ABBILDUNG 3.2: Zeitlicher Ablauf von *getup_back*

3.2 Probleme

Aus der beschriebenen aktuellen Implementierung ergeben sich verschiedene Probleme. Diese werden in diesem Kapitel behandelt. Ziel ist es, diese Probleme detailliert zu beschreiben, um im weiteren Verlauf der Arbeit gezielte Lösungsansätze zur Verbesserung der Stabilität und Zuverlässigkeit des Roboters zu entwickeln.

3.2.1 Schwung

Das Hauptproblem bei *getup_back* ist, dass aufgrund der hohen Geschwindigkeit zu viel Schwung im System entsteht, was den Roboter umkippen lässt. Dies lässt sich nicht komplett durch eine reine Anpassung der statischen Bewegung ausgleichen, da der Schwung nicht bei jedem Versuch gleich ist. Wie stark der Roboter schwingt, hängt von verschiedenen Faktoren ab.
Einer dieser Faktoren sind die unterschiedlichen Zustände der Roboter. Ältere Roboter, die bereits mehrmals defekt waren und repariert wurden, haben in ihren Gelenken mehr Spiel. Dadurch können die Gelenkwinkel selbst bei maximaler Festigkeit nicht mehr durchgängig gehalten werden. Die herstellereigene Software wirkt dieser Abweichung entgegen und versucht, den Zielwinkel zu erreichen. Jedoch kann die Korrektur mehrere Sekunden

dauern, was für die Zwecke dieser Arbeit zu langsam ist [16]. Das führt dazu, dass die Roboter unterschiedliche Anforderungen an die Bewegung stellen, um sie erfolgreich ausführen zu können.
Weitere Faktoren, welche den Schwung beeinflussen, entstehen durch äußere Umstände. Zum einen können Roboter von anderen Robotern angestoßen werden, was mehr Energie in das System bringt und für verstärkte Schwingungen sorgt. Zum anderen variiert die Qualität und Beschaffenheit des Kunstrasens sowie die Neigung des Bodens zwischen verschiedenen Veranstaltungsorten, was ebenfalls die Bewegung beeinflusst.

3.2.2 Gelenkwinkel werden nicht erreicht

Ein weiteres Problem kann sein, dass die von der Bewegungssteuerung vorgegebenen Gelenkwinkel nicht erreicht werden. Dafür gibt es verschiedene Ursachen: Einerseits kann das unterschiedliche Spiel in den Gelenken der Roboter dafür verantwortlich sein, andererseits kann etwas Externes, wie ein Ball, die Bewegung physisch blockieren. Auch das eigene Gewicht des Roboters kann ein Grund sein. Die Gelenkmotoren können nur eine bestimmte Kraft aufbringen. Wenn der Roboter durch Schwung oder aus anderen Gründen mehr Kraft auf ein Gelenk ausübt, als vorgesehen ist, kann es vorkommen, dass das Gelenk die gewünschte Position nicht mehr erreicht [16]. Dies führt zu einer Verlagerung des Schwerpunktes des Roboters, was zum Umkippen während der Bewegung führen kann.
Zusätzlich wird die Festigkeit der Motoren automatisch reduziert, wenn diese zu heiß werden. Dadurch ist es möglich, dass ein Roboter nicht mehr die benötigte Kraft aufbringen kann, um die Bewegung korrekt auszuführen.

3.2.3 Statische Bewegung

Wie bereits in Kapitel 3.1 erwähnt, ist der aktuelle Bewegungsablauf statisch. Das heißt, die Bewegung wird jedes Mal gleich abgespielt, unabhängig von allen Umständen. Dadurch können die in Kapitel 3.2.1 und 3.2.2 beschriebenen Probleme bzw. Einflussfaktoren auf die Aufstehbewegung nicht berücksichtigt und in den Bewegungsablauf einbezogen werden.
Dies führt dazu, dass die Bewegung nur unter bestimmten Bedingungen oder auf bestimmten Robotern gut funktioniert. Deshalb ist die statische Bewegung nicht für einen allgemeinen, zuverlässigen Einsatz im Roboterfußball geeignet.

4 Implementierung

Die in diesem Kapitel beschriebenen Verfahren wurden entwickelt, um die in Kapitel 3.2 erwähnten Probleme zu beheben.
Wie bereits in Kapitel 2.3 erwähnt, wird dafür ein Ansatz gewählt, bei dem die *pitch*- und *roll*-Winkel und -Winkelgeschwindigkeiten berücksichtigt werden. Diese Methode eignet sich gut für die angesprochenen Probleme, da über die gemessenen Winkel direkt erkannt wird, wann ein Roboter in einer falschen Position ist.
Die gemessenen Winkel und Winkelgeschwindigkeiten beziehen sich immer auf den Oberkörper des Roboters. Die Korrektur beginnt erst, wenn der Roboter auf beiden Beinen steht. So können Abweichungen erkannt werden, wenn der Roboter zu stark in eine Richtung schwingt oder wenn Gelenkwinkel nicht erreicht werden, da sich dadurch der Oberkörper verlagert.
Um die Position des Roboters korrigieren zu können, müssen die Zielwerte für jeden Zeitpunkt bekannt sein. Diese Werte werden mithilfe eines kinematischen Modells des NAO und Vorwärtskinematik berechnet. Da noch keine Vorwärtskinematik im HTWK-Framework implementiert ist, wird diese im nachfolgenden Kapitel eingeführt. Die berechneten Winkel und Winkelgeschwindigkeiten können anschließend mit in dem Motion-File der Bewegung abgespeichert werden. Somit können diese zusammen mit den Gelenk-Zielwinkeln geladen werden, wenn der Roboter aufstehen muss. Während der Bewegung werden die berechneten Werte kontinuierlich mit den von der IMU gemessenen Werten verglichen.
Um Korrekturen in *pitch*- und *roll*-Richtung zu berechnen, werden vier PI-Regler genutzt. Jeweils zwei für die *pitch*- und zwei für die *roll*-Werte. Dabei kontrolliert jeweils einer die Winkelabweichungen und einer die Abweichungen der Winkelgeschwindigkeiten. Die einzelnen Korrekturwerte werden summiert, um jeweils einen einzelnen Korrekturwinkel für *pitch* und *roll* zu erhalten. Zuletzt wird entschieden, wie die Korrekturwinkel auf die Gelenke verteilt werden. Diese Korrekturen werden dann auf die ursprünglichen Gelenkwinkel der Bewegung addiert.

4.1 Kinematische Modellierung und Berechnungsmethoden

Um die Erwartungswerte für die Oberkörperwinkel des Roboters zu berechnen, wird ein kinematisches Modell gebraucht. Dies wird in diesem Kapitel eingeführt. Weiterhin wird überprüft, ob die aktuellen Aufstehbewegungen der HTWK-Robots alle Anforderungen an dieses Modell erfüllen. Wenn nötig werden die Bewegungen angepasst.

4.1.1 Berechnung der Oberkörperwinkel

Um die *pitch*- und *roll*-Winkel des Torsos zu berechnen, wird die Orientierung dessen benötigt. Diese hängt wiederum von der Orientierung der Beine ab. Über die in Kapitel 2.2.1 beschriebene Vorwärtskinematik können diese Orientierungen berechnet werden. Für die Umsetzung stehen mehrere Methoden zur Verfügung. Häufig werden dafür beim NAO Rotationsmatrizen genutzt [34, 35]. Eine Alternative, um 3D-Rotationen zu berechnen, sind Quaternionen. Diese haben im Vergleich zu Rotationsmatrizen einen geringeren Rechenaufwand bei der Berechnung von kombinierten Rotationen. Dies ist insbesondere bei Echtzeitanwendungen ein Vorteil.
Wie bereits in Kapitel 2.2.2 erklärt, können kombinierte Rotationen durch Quaternionen einfach berechnet und ausgedrückt werden. Die korrekte Reihenfolge der Multiplikationen ist dabei entscheidend, da die Orientierung eines Gelenks von den Stellungen der Gelenke abhängt, die vor diesem in der kinematischen Kette liegen. Die Beine des NAO bilden zwei unabhängige kinematische Ketten. Die Reihenfolge der Gelenke ist jedoch identisch und kann in Tabelle 4.1 für beide Beine eingesehen werden.

linkes Bein	**rechtes Bein**	Abkürzung
LHipYawPitch (lhyp)	RHipYawPitch (rhyp)	hyp
LHipRoll (lhr)	RHipRoll (rhr)	hr
LHipPitch (lhp)	RHipPitch (rhp)	hp
LKneePitch (lkp)	RKneePitch (rkp)	kp
LAnklePitch (lap)	RAnklePitch (rap)	ap
LAnkleRoll (lar)	RAnkleRoll (rar)	ar

Tabelle 4.1: In dieser Tabelle sind die kinematischen Ketten der beiden Beine des NAO v6 dargestellt. Die Reihenfolge wird durch den Startpunkt der Betrachtung bestimmt. Ausgehend vom Torso ist die Reihenfolge von oben nach unten. Ausgehend von den Füßen von unten nach oben.

Als Basis für die Berechnungen werden die Füße gewählt, da dadurch die resultierenden Quaternionen die Orientierungen der beiden Hüftenden darstellen. Diese können dann kombiniert werden, um die Orientierung des Torsos zu bestimmen. Dabei wird angenommen, dass die Füße stets eben auf dem Boden stehen. Dies vereinfacht später die Bestimmung der Winkel des Oberkörpers, da die Ebene auf der der Roboter steht, die x-y-Ebene, dann bereits bekannt ist. Die Winkel werden dann zur Normale dieser Ebene, also der z-Achse, berechnet. Würde der Torso als Basis genommen werden, müsste die Ebene auf der der Roboter steht zusätzlich berechnet werden.
Bevor die Quaternionen der Beine berechnet werden können, muss allerdings noch die *yaw*-Rotation des jeweiligen Beines beachtet werden. Wie beispielsweise in Abbildung 3.1f zu sehen ist, sind die Beine stets zum Torso hin rotiert. Um diese Rotation darzustellen, muss die gesamte *yaw*-Rotation des jeweiligen Beines zu Beginn der Berechnung an den Füßen angesetzt werden. Somit ergibt sich für die Quaternionen der Beine folgende Formel:

$$q_{leg} = q_y \prod_{n=1}^{6} q_n \tag{4.1}$$

Hier ist q_y die Quaternion für die *yaw*-Rotation des jeweiligen Beines. Die restlichen Quaternionen gehören zu den jeweils involvierten Bein-Gelenken. Diese können in Tabelle 4.1 eingesehen werden. Da die Berechnung bei den Füßen beginnen soll, beschreibt q_1 die Quaternion des jeweiligen *AnkleRoll*-Gelenks. Die restlichen Quaternionen q_n folgen dann der kinematischen Kette. Die Quaternionen der Gelenke können jeweils mit der Formel 2.4 erstellt werden. Für $\boldsymbol{u}$ wird die Rotationsachse des Gelenks eingesetzt und für ϕ der aktuelle Rotationswinkel dieses Gelenks. Um q_y zu berechnen, welche die gesamte *yaw*-Rotation des Beines darstellen soll, muss zunächst die Quaternion des Beines ohne q_y berechnet werden. Anschließend kann ein Vektor der Form $(0, x, 0)^T, x > 0$ mithilfe der Formel 2.9 und der entstandenen Quaternion rotiert werden. Die Form des Vektors wurde so gewählt, um die ursprüngliche Ausrichtung der Hüftachse zu repräsentieren. Mit dem resultierenden Punkt p, kann dann der *yaw*-Winkel mit folgender Formel berechnet werden:

$$yaw = \arctan\left(\frac{p_x}{p_y}\right) \tag{4.2}$$

Der *yaw*-Winkel entspricht dem Winkel zwischen dem Ortsvektor des Punktes p und der y-Achse. Dieser Winkel wird benutzt um q_y zu berechnen. Damit wird garantiert, dass die Achse der Hüfte des Roboters stets parallel zur y-Achse ist, was die Berechnungen der Oberkörperwinkel später vereinfacht.
Nachdem die beiden Quaternionen der Beine mithilfe der Formel 4.1 berechnet wurden, können diese kombiniert werden, um die Quaternion des Torsos zu erhalten. Da die Rotationen der Beine jedoch unabhängig voneinander sind, reicht es hier nicht eine Multi-

plikation durchzuführen. Stattdessen kann hier eine *spherical linear interpolation* (slerp) durchgeführt werden. Die Formel sieht dafür folgendermaßen aus:

$$\text{slerp}(q_1, q_2, t) = \frac{q_1 \sin((1-t)\theta) + q_2 \sin(t\theta)}{\sin(\theta)} \tag{4.3}$$

mit:

$$\theta = \arccos(q_1 \cdot q_2) \tag{4.4}$$

Durch diese Formel wird zwischen den beiden Quaternionen q_1 und q_2 auf einem Kreisbogen interpoliert, wobei t den Abschnitt angibt. Mit einem Wert von $t = 0.5$ wird die Quaternion ermittelt, welche exakt in der Mitte des Kreisbogens zwischen q_1 und q_2 liegt [36]. Diese Quaternion kann für den Torso genutzt werden. Somit ergibt sich für die Torso-Quaternion:

$$q_t = \text{slerp}(q_l, q_r, 0.5) \tag{4.5}$$

Diese Quaternion kann nun, ähnlich zur Berechnung von q_y, genutzt werden um die *pitch*- und *roll*-Winkel des Torsos zu berechnen. Diesmal wird jedoch ein Vektor der Form $(0, 0, x)^T, x > 0$ für die Formel 2.9 benötigt, um die ursprüngliche Orientierung des Torsos darzustellen. Mit dem erhaltenen Punkt p können die *pitch*- und *roll*-Winkel mit folgenden Formeln berechnet werden:

$$pitch = \arctan\left(\frac{p_x}{p_z}\right) \tag{4.6}$$

$$roll = \arctan\left(\frac{p_y}{p_z}\right) \tag{4.7}$$

Neben den Winkeln, werden noch die Winkelgeschwindigkeiten benötigt. Diese können über die Werte der Winkel berechnet werden:

$$\omega = \frac{\theta_n - \theta_{n-1}}{0.012\,\text{s}} \tag{4.8}$$

θ ist hier entweder der *pitch*- oder *roll*-Winkel bei Frame n. Die 0.012 s ergeben sich daraus, dass ein Frame 12 ms lang ist.
Diese Berechnungen werden für jeden Frame der Bewegung durchgeführt. Die dadurch erhaltenen *pitch*- und *roll*-Winkel und -Winkelgeschwindigkeiten dienen als Erwartungswerte. Diese werden dann während des Aufstehens benutzt, um Abweichungen zu erkennen und Korrekturen durchzuführen.

4.1.2 Überprüfung der Modellannahmen

In Kapitel 4.1.1 wurde die Annahme beschrieben, dass die Füße des Roboters stets eben auf dem Boden stehen, sobald dieser eine stehende Position angenommen hat. Jedoch muss noch überprüft werden, ob diese Annahme für die vorliegenden Bewegungen, *getup_back_slow* und *getup_back*, zutrifft. Falls das nicht der Fall ist, muss dies noch korrigiert werden, damit das in Kapitel 4.1.1 beschriebene Verfahren auch angewendet werden kann.
Um diese Annahme zu überprüfen, kann die Position eines Fußgelenks relativ zum anderen Fußgelenk berechnet werden. Wenn die Annahme zutrifft, befinden sich beide Fußgelenke auf derselben Höhe. Falls beide Fußgelenke auf derselben Höhe sind, muss noch überprüft werden, ob der Richtungsvektor vom Fußgelenk zum Fuß orthogonal zur Standebene des Roboters ist. Wenn das nicht der Fall ist, muss auch das noch angepasst werden.
Um diese Berechnungen durchzuführen, kann wieder die Vorwärtskinematik benutzt werden. Da diesmal aber nicht nur die Orientierung eines Endeffektors, sondern auch dessen Position berechnet werden soll, werden zusätzlich die Dimensionen der verbindenden Glieder benötigt. Diese sind in Abbildung 4.1 dargestellt.

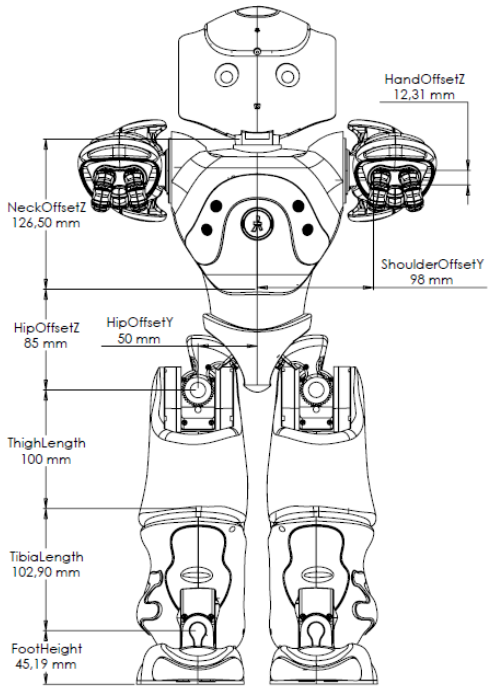


Abbildung 4.1: Dimensionen des NAO v6.[1]

[1] Quelle: http://doc.aldebaran.com/2-8/_images/hardware_lengthfront_3.3.png, Zugriff: 11.11.2024, Rechteinhaber: Aldebaran - SAS (Limited Company)

Als Startpunkt wird das linke Fußgelenk genommen. Mithilfe der Formel 2.9 kann iterativ die Position des rechten Fußgelenks berechnet werden:

$$p_{rAnkle} = \sum_{n=1}^{5} \mathrm{Im}(q_n v_n q_n^*) \tag{4.9}$$

wobei:

$$q_1 = q_{lar} q_{lap}, \quad q_2 = q_1 q_{lkp}, \quad q_3 = q_2 q_{lhp} q_{lhr} q_{lhyp}, \quad q_4 = q_3 q_{rhyp} q_{rhr} q_{rhp}, \quad q_5 = q_4 q_{rkp} \tag{4.10}$$

und:

$$v_1 = \begin{pmatrix} 0 \\ 0 \\ 0.1029\,\mathrm{m} \end{pmatrix}, \quad v_2 = \begin{pmatrix} 0 \\ 0 \\ 0.1\,\mathrm{m} \end{pmatrix}, \quad v_3 = \begin{pmatrix} 0 \\ 0.1\,\mathrm{m} \\ 0 \end{pmatrix}, \quad v_4 = -v_2, \quad v_5 = -v_1. \tag{4.11}$$

q_1, q_3 und q_4 ergeben sich aus mehreren Gelenken. Das liegt daran, dass dies kombinierte Gelenke sind, das heißt, dass zwei oder mehr Gelenke denselben Rotationsmittelpunkt teilen. Auch hier muss bei der Multiplikation auf die durch die kinematische Kette vorgegebene Reihenfolge geachtet werden.
Falls beide Fußgelenke auf derselben Höhe sind, ergibt sich bei p_{rAnkle} ein z-Wert von Null. Es werden allerdings auch noch Werte akzeptiert, welche bis zu 0.1 mm abweichen, da es immer zu numerischen Fehlern kommen kann oder die Bewegung an sich nicht perfekt ist. Solche kleinen Abweichungen haben jedoch keinen relevanten Einfluss auf die tatsächliche Bewegung. Diese Berechnung muss für jeden Frame der Aufstehbewegungen durchgeführt werden.
Dabei ist aufgefallen, dass keine der beiden Bewegungen die Annahme erfüllt und die Abweichungen bis zu 2.5 cm betragen. Dementsprechend müssen diese Bewegungen angepasst werden, damit das Verfahren aus Kapitel 4.1.1 korrekt angewandt werden kann.

4.1.3 Anpassung der Aufstehbewegung

Bei der Korrektur ist es das Ziel, die Position des Roboters so zu verändern, dass dieser stets eben auf dem Boden steht. Dafür wird die Position, p_{rAnkle}, des rechten Fußgelenks benötigt, welche im letzten Kapitel berechnet wurde. Da das linke Fußgelenk als Basis gewählt wurde, entspricht p_{rAnkle} gleichzeitig dem Richtungsvektor vom linken zum rechten Fußgelenk. Um den Ziel-Richtungsvektor, p_{soll}, zu erhalten, können die x- und y-Koordinaten von p_{rAnkle} übernommen und die z-Koordinate auf Null gesetzt werden. Dadurch wird die Richtung beibehalten und nur die Höhe verändert.

Um eine Quaternion für die benötigte Rotation zur Korrektur zu erstellen, wird der Winkel zwischen p_{rAnkle} und p_{soll} genutzt. Als Rotationsachse dient das Kreuzprodukt dieser beiden Vektoren. Mit diesen Elementen lässt sich mit der Gleichung 2.4 die Quaternion, q_c, erstellen, welche die benötigte Rotation darstellt.
Um die Position des Roboters zu korrigieren, können die *pitch*- und *roll*-Winkel des linken Fußgelenks verändert werden. Die dafür benötigten Winkel können aus q_c abgeleitet werden, indem die Quaternion in Tait-Bryan-Winkel umgewandelt wird.
Tait-Bryan-Winkel beschreiben eine Rotation durch drei aufeinanderfolgende Rotationen um drei verschiedene Achsen. Bei der Umwandlung von q_c muss die Rotationsreihenfolge der Tait-Bryan-Winkel beachtet werden. Je nachdem, in welcher Reihenfolge die Rotationen ausgeführt werden, werden die Winkel unterschiedlich aus der Quaternion berechnet. Hier wird die Reihenfolge *roll-pitch-yaw* benötigt, was durch die kinematische Kette des Beines vorgegeben wird. Die *yaw*-Rotation kann in diesem Fall ignoriert werden, da das Fußgelenk keine *yaw*-Rotationsachse hat. Die Formeln für *pitch*- und *roll*-Winkel sind folgende [10]:

$$pitch = -\arcsin(2q_xq_z - 2q_wq_y) \tag{4.12}$$

$$roll = \arctan2(2q_yq_z + 2q_wq_x, q_z^2 - q_y^2 - q_x^2 + q_w^2) \tag{4.13}$$

Die Ergebnisse dieser Berechnungen können auf die aktuellen *LAnklePitch*- und *LAnkleRoll*-Winkel addiert werden. Allerdings ist dadurch noch nicht garantiert, dass das rechte Fußgelenk auf derselben Höhe ist wie das linke. Das liegt an der vereinfachten Berechnung, bei der nur der Vektor vom linken zum rechten Fußgelenk genommen und der restliche Roboter ignoriert wird. Jedoch wurde durch die Berechnung sichergestellt, dass sich die Korrekturwinkel den tatsächlich benötigten Winkeln annähern. Deshalb müssen p_{rAnkle} und der Winkel zu p_{soll} mit den neuen Werten neu berechnet werden. Von da an muss das Verfahren so lange wiederholt und die Korrekturwinkel aufaddiert werden, bis eine definierte Genauigkeit erreicht wird. Wie bereits in Kapitel 4.1.2 beschrieben, wird hier so lange korrigiert, bis die Höhe von p_{rAnkle} maximal 0.1 mm von der Höhe des linken Fußgelenks abweicht.
Durch die Korrektur am linken Fußgelenk hat sich auch die Orientierung des rechten Fußgelenks verändert. Dadurch müssen die Winkel der rechten *pitch*- und *roll* Fußgelenke so neu berechnet werden, dass der Fuß eben auf dem Boden ist. Die Position des Fußes wird folgenderMaßen berechnet:

$$p_{rFoot} = p_{rAnkle} + \mathrm{Im}(q_6v_6q_6^*) \tag{4.14}$$

mit einer Erweiterung für 4.10 und 4.11:

$$q_6 = q_5 q_{rap} q_{rar}, \quad v_6 = \begin{pmatrix} 0 \\ 0 \\ -0.045\,19\,\mathrm{m} \end{pmatrix} \tag{4.15}$$

Der Fuß ist Eben auf dem Boden, wenn der Richtungsvektor, p_r, vom rechten Fußgelenk zum rechten Fuß gerade nach unten zeigt. Er muss also der Form von p_{rSoll} entsprechen: $(0, 0, x)^T, x < 0$. Um herauszufinden, wie stark korrigiert werden muss, wird der Winkel zwischen p_r und p_{rSoll} benötigt. Die Rotationsachse ergibt sich, wie bei dem linken Fußgelenk, aus dem Kreuzprodukt der beiden Vektoren. Diese können jedoch noch nicht verwendet werden, da sie noch in globalen Koordinaten vorliegen. Die Vektoren werden aber in den lokalen Koordinaten des rechten Fußgelenks benötigt, da die Winkel dieses Gelenks auch korrigiert werden sollen. Bei dem linken Fußgelenk war dies nicht nötig, da dessen lokales Koordinatensystem dem globalen entspricht. Um die Koordinaten umzuwandeln, kann folgende Formel genutzt werden:

$$p_{local} = \mathrm{Im}(q^* v_g q) \tag{4.16}$$

v_g ist hier ein Vektor in globalen Koordinaten und q die Quaternion, in deren relatives Koordinatensystem übersetzt werden soll. Durch diese Berechnung bleiben die Richtungen der Vektoren im globalen Koordinatensystem gleich, werden aber in den Koordinaten des lokalen Systems ausgedrückt.
Nachdem die beiden Vektoren übersetzt wurden kann die Quaternion für die Korrektur mit der Formel 2.4 erstellt werden. Dafür wird wie erwähnt der Winkel zwischen den Vektoren und deren Kreuzprodukt als Rotationsachse genutzt. Die erhaltene Quaternion muss wie im vorherigen Abschnitt in Tait-Bryan-Winkel umgewandelt werden, um die einzelnen Winkel für die Korrektur zu erhalten. Diesmal muss jedoch eine andere Reihenfolge, nämlich *yaw-pitch-roll*, genutzt werden, um der kinematischen Kette gerecht zu werden. Damit ergibt sich für die *pitch*- und *roll*-Winkel [10]:

$$pitch = \arcsin(2q_x q_z + 2q_w q_y) \tag{4.17}$$

$$roll = \arctan2(-2q_y q_z + 2q_w q_x, q_z^2 - q_y^2 - q_x^2 + q_w^2) \tag{4.18}$$

Diese Winkel können auf die aktuellen *RAnklePitch*- und *RAnkleRoll*-Winkel addiert werden.

Nach diesen beiden Korrekturen ist es garantiert, dass der Roboter im aktuellen Frame mit beiden Beinen eben auf dem Boden steht. Dieses Verfahren muss bei jedem Frame angewandt werden, in welchem der Roboter nicht eben auf dem Boden steht.

4.2 Gelenkauswahl

In Kapitel 4.1.2 wurde überprüft, ob die Bewegung den Anforderungen entspricht. Gegebenenfalls wurde sie in Kapitel 4.1.3 angepasst, und in Kapitel 4.1.1 wurden Zielwerte für die Torso-Winkel bestimmt. Nun fehlt noch die Auswahl der Gelenke, mit welchen die Korrekturen ausgeführt werden sollen. Wie die Korrekturwinkel berechnet werden, wird in Kapitel 4.3 beschrieben.
Wie bereits in Kapitel 2.3 erwähnt, wird zur Auswahl der Gelenke eine Mischung aus *Ankle* und *Hip Strategy* genutzt. Bei der *pitch*-Korrektur werden zusätzlich noch die Kniegelenke genutzt. Standardmäßig werden die Korrekturen gleichmäßig auf alle relevanten Gelenke verteilt. Das Kniegelenk kommt allerdings nur zum Einsatz, wenn die Beine weniger als 24° von der neutralen Position weggedreht sind, um zu garantieren, dass die Korrektur hauptsächlich in *pitch*-Richtung geschieht. Bei der Korrektur mit der Hüfte wird das *HipYawPitch*-Gelenk nicht mitbenutzt, da hier immer zusätzlich eine *yaw*-Rotation entsteht. Diese Gelenkauswahl soll dafür sorgen, dass sowohl auf kleine als auch große Veränderungen reagiert werden kann. Außerdem soll durch die Verteilung die Last auf den einzelnen Gelenken minimiert werden. Weiterhin eignet sich diese Gelenkauswahl gut, um die Haltung des Oberkörpers zu korrigieren, da sich die gemessenen Abweichungen der Oberkörperwinkel auf die Gelenke der Beine übertragen lassen. Außerdem hat eine Änderung der Winkel bei den Beingelenken einen direkten Einfluss auf die Winkel des Torsos. Die Korrekturwinkel können jedoch in der Regel nicht direkt auf die Gelenke addiert werden, weil, je nachdem, wie die Beine orientiert sind, die lokalen Koordinatensysteme der Gelenke nicht mit dem globalen Koordinatensystem übereinstimmen. Dementsprechend müssen die jeweiligen *AnkleRoll*- und *AnklePitch*-Gelenke und *HipRoll*- und *HipPitch*-Gelenke so kombiniert rotiert werden, dass eine reine Rotation um die gewünschte Achse im globalen Koordinatensystem entsteht. Hierfür können erneut Quaternionen verwendet werden.
Zunächst muss die gewünschte Rotationsachse, für *pitch* $(0, 1, 0)^T$ und für *roll* $(1, 0, 0)^T$, in das relative Koordinatensystem der jeweiligen Gelenke gebracht werden. Dafür kann die Formel 4.16 genutzt werden. Für q muss die Quaternion eingesetzt werden, welche die aktuelle Orientierung des relativen Koordinatensystems an den Gelenken beschreibt. Dies ergibt sich aus der Multiplikation aller Quaternionen, welche vor dem Gelenk in der

Kinematischen Kette liegen. Bei den kombinierten Gelenken muss noch das erste Teilgelenk mit einbezogen werden, da es ebenfalls die Orientierung des kombinierten Gelenks beeinflusst. Somit ergibt sich für die Orientierungen der verschiedenen Bereiche:

$$o_{ankle} = q_y q_{ar} \tag{4.19}$$

$$o_{knee} = q_y q_{ar} q_{ap} \tag{4.20}$$

$$o_{hip} = q_y q_{ar} q_{ap} q_{kp} q_{hp} \tag{4.21}$$

Nachdem die jeweilige Rotationsachse in das gewünschte lokale Koordinatensystem übersetzt wurde, kann diese genutzt werden, um mit der Formel 2.4 eine Quaternion zu erzeugen. Als Winkel, ϕ, wird der Anteil des Korrekturwinkels genommen, welcher mit dem jeweiligen Gelenk abgedeckt werden soll. Die resultierende Quaternion kann anschließend mit den Formeln 4.12 und 4.13 in Tait-Bryan-Winkel aufgeteilt werden. Diese Winkel können dann direkt auf die Gelenke addiert werden, um den Oberkörper um den Winkel ϕ um die spezifizierte globale Rotationsachse zu rotieren.

4.2.1 Berechnung der Gelenk-Gewichte

Ein Problem bei der Korrektur kann sein, dass ein Gelenk die erforderliche Rotation nicht ausführen kann. Das kann zum einen daran liegen, dass durch die Korrektur die maximale Auslenkung eines Gelenks überschritten wird oder dass es zu Kollisionen zwischen den Körperteilen des Roboters kommt. Die Maximalwinkel der für die Korrektur relevanten Gelenke können in Tabelle 4.2 eingesehen werden. Bei den Fußgelenken gibt es zusätzliche Einschränkungen, welche der Kollisionsvermeidung dienen und in der Tabelle 4.3 zu sehen sind. Es muss also bestimmt werden, welchen maximalen Anteil der erforderlichen Gesamtkorrektur jedes einzelne Gelenk leisten kann, um festzulegen, wie die Korrektur zwischen den Gelenken aufgeteilt werden soll.
Um das herauszufinden, wird eine binäre Suche genutzt. Die binäre Suche ermöglicht es, effizient den maximalen Korrekturanteil zu finden, den ein Gelenk ausführen kann, ohne seine Grenzen zu überschreiten. Dafür wird zu Beginn angenommen, dass ein gewähltes Gelenk die Hälfte des kompletten Korrekturwinkels umsetzen muss. Anschließend werden die Berechnungen aus Kapitel 4.2 für dieses ausgewählte Gelenk durchgeführt, um herauszufinden, was die aus dem Korrekturwinkel resultierenden Gelenkwinkel wären. Wenn sich die resultierenden Gelenkwinkel nicht innerhalb der erlaubten Grenzen befinden, wird das Verfahren mit einem durch die binäre Suche angepassten Korrekturwinkel wiederholt. Dies wird maximal zehnmal durchgeführt. Dadurch ist sichergestellt, dass eine gute Annäherung an den durch das Gelenk tatsächlich realisierbaren maximalen Korrekturwinkel

gefunden wird.
Diese Berechnung wird für jedes an der Korrektur beteiligte Gelenk durchgeführt. Die dadurch erhalten maximalen Gewichte, $weight_{jointMax}$, geben an, wie viel Prozent der gesamten Korrektur von diesem Gelenk übernommen werden könnte. Um schlussendlich zu berechnen, wie stark jedes Gelenk tatsächlich korrigieren muss, wird folgende Formel genutzt:

$$weight_{joint} = \min\left(weight_{jointMax}, \frac{weight_{jointMax}}{weight_{total}}\right) \tag{4.22}$$

mit:

$$weight_{total} = \sum_{n=0}^{x} weight_{jointMax,n} \tag{4.23}$$

Das in Formel 4.23 berechnete $weight_{total}$ ist die Summe der $weight_{jointMax}$ aller Gelenke, welche zur Korrektur verwendet werden. Mit der Formel 4.22 kann für jedes Gelenk das jeweils finale Gewicht berechnet werden. Dadurch wird garantiert, dass kein Gelenk einen Winkel ansteuert, der physisch nicht erreichbar ist und trotzdem, wenn möglich, die Korrektur vollständig umgesetzt wird.

Gelenk	**Bereich in Grad**	
	links	**rechts**
HipRoll	-21.74 – 45.29	-45.29 – 21.74
HipPitch	-88.00 – 27.73	-88.00 – 27.73
KneePitch	-5.29 – 121.04	-5.90 – 121.47
AnklePitch	-68.15 – 52.86	-67.97 – 53.40
AnkleRoll	-22.79 – 44.06	-44.06 – 22.80

Tabelle 4.2: Minimale und maximale Winkel der Beingelenke des NAO v6. Einschränkungen für die *AnkleRoll*-Gelenke sind in Tabelle 4.3 abgebildet.[1]

pitch-Winkel in Grad	**roll-Winkel-Bereich in Grad**
-68.15	-2.86 – 4.30
-48.13	-10.31 – 9.74
-40.11	-22.80 – 12.61
-25.78	-22.80 – 44.06
5.73	-22.80 – 44.06
20.05	-22.80 – 31.54
52.87	0.00 – 2.86

Tabelle 4.3: Minimale und maximale Winkel des *LAnkleRoll*-Gelenks in Abhängigkeit des Winkels des *LAnklePitch*-Gelenks. Dies dient der Kollisionsvermeidung der Glieder. Für das rechte Bein sind die *pitch*-Winkel identisch und die *roll*-Winkel negiert.[1]

[1] Quelle: http://doc.aldebaran.com/2-8/family/nao_technical/joints_naov6.html, Zugriff: 11.11.2024

4.3 Berechnung des Korrekturwinkels

Um die tatsächliche Balancierung des Roboters durchzuführen, müssen noch die Korrekturwinkel bestimmt werden. Dafür wird, wie in Kapitel 4 beschrieben, die aktuelle Position des Roboters mit der Sollposition verglichen, welche in Kapitel 4.1.1 berechnet wurde. Dabei wird jedoch nicht die Sollposition des aktuellen Frames genutzt, sondern die Winkel des Frames, der bereits 4 Frames in der Vergangenheit liegt. Bei Frame 104, wird also die berechneten Sollwinkel von Frame 100 als Vergleich herangezogen. Diese Vorgehensweise ist nötig, da es zu Verzögerungen kommt, wenn die Gelenkwinkel an die Motoren übertragen, umgesetzt und die Veränderungen anschließend gemessen werden [15].
Um zu berechnen wie stark der Roboter korrigieren soll, werden wie in Kapitel 4 erwähnt vier PI-Regler genutzt. Dabei wurde bewusst gegen PID-Regler entschieden. Das Problem bei PID-Reglern ist in diesem Fall der D-Anteil, da er bei der Eingabe eines Torso-Winkels der Winkelgeschwindigkeit entspricht. Somit würde der D-Anteil stets die Bewegung verstärken oder dieser entgegenwirken, unabhängig davon, ob der Oberkörper dem Verlauf der Sollwinkel folgt oder nicht.
Durch die Aufteilung des PID-Reglers in zwei PI-Regler kann dieses Problem umgangen werden. Denn somit sind die Korrekturen, welche sich durch die Abweichung der Torso-Winkel ergeben unabhängig von der Winkelgeschwindigkeit. Um die Abweichungen der Winkelgeschwindigkeit zu kontrollieren, gibt es dann einen zweiten PI-Regler, welcher einen eigenen Sollwert erhält und somit Abweichungen der Winkelgeschwindigkeit entgegenwirkt. Zusätzlich erhält die Korrektur der Winkelgeschwindigkeit durch den eigenen Regler eine weitere Komponente, da ein eigener I-Anteil vorhanden ist.
Nachdem die Regler definiert sind, müssen diese noch eingestellt werden, indem sinnvolle Werte für K und K_i gefunden werden. Dafür wurde ein Roboter in durchschnittlichem Zustand genutzt. Dieser wurde auf einem Linoleumboden aus der Rückenlage mit der Bewegung *getup_back_slow* zum Aufstehen gebracht. In jedem Versuch wurde der Roboter dreimal aufstehen gelassen, um Durchschnittswerte zu bilden. Gemessen wurden die *pitch*- und *roll*-Winkel des Oberkörpers. Zunächst wurde dies ohne Korrektur durchgeführt, um eine Grundlage zu haben, auf der die zukünftigen Versuche bewertet werden können. Zu Beginn wurden alle K und K_i Werte auf Null gestellt.
Zuerst wurden die Regler für die *pitch*-Korrektur angepasst, da in dieser Richtung der meiste Schwung ist. Außerdem ist die Standfläche des Roboters in diese Richtung schmal, was ein robustes balancieren in diese Richtung besonders wichtig macht. Begonnen wurde mit der Einstellung des Proportionalwertes K des Winkelgeschwindigkeits-Reglers. Der Winkelgeschwindigkeits-Regler wurde vor dem Winkel-Regler eingestellt, da dieser vor allem starken Schwankungen entgegenwirken soll. Als Startwert wurde ein geringer Wert

von 0.01 angenommen. Dieser wurde dann schrittweise über eine binäre Suche angepasst, um den optimalen Wert zu finden. Zwischen jeder Anpassung wurde ein Versuch durchgeführt. Dies wurde auch für den K-Wert und anschließend auch für den Winkel-Regler wiederholt.
Das Ergebnis ist in Diagramm 4.2 ersichtlich. Es lässt sich erkennen, dass die Abweichung teilweise noch ähnlich groß ist, jedoch vor allem die Schwankungen deutlich geringer ausfallen. Dieselbe Prozedur wurde anschließend für die *roll*-Regler durchgeführt. Dabei blieb die *pitch*-Korrektur aktiv, um sicherzustellen, dass die gefundenen Werte kompatibel sind. Zudem wurde die Minimierung der *pitch*-Abweichung priorisiert, da diese Richtung aufgrund der schmalen Standfläche kritisch für die Stabilität ist. Durch die bessere Grundstabilität bei den *roll*-Winkeln hat die Korrektur keinen so großen Einfluss wie beim *pitch*, dennoch lässt sich im Diagramm 4.3 eine leichte Verbesserung erkennen.

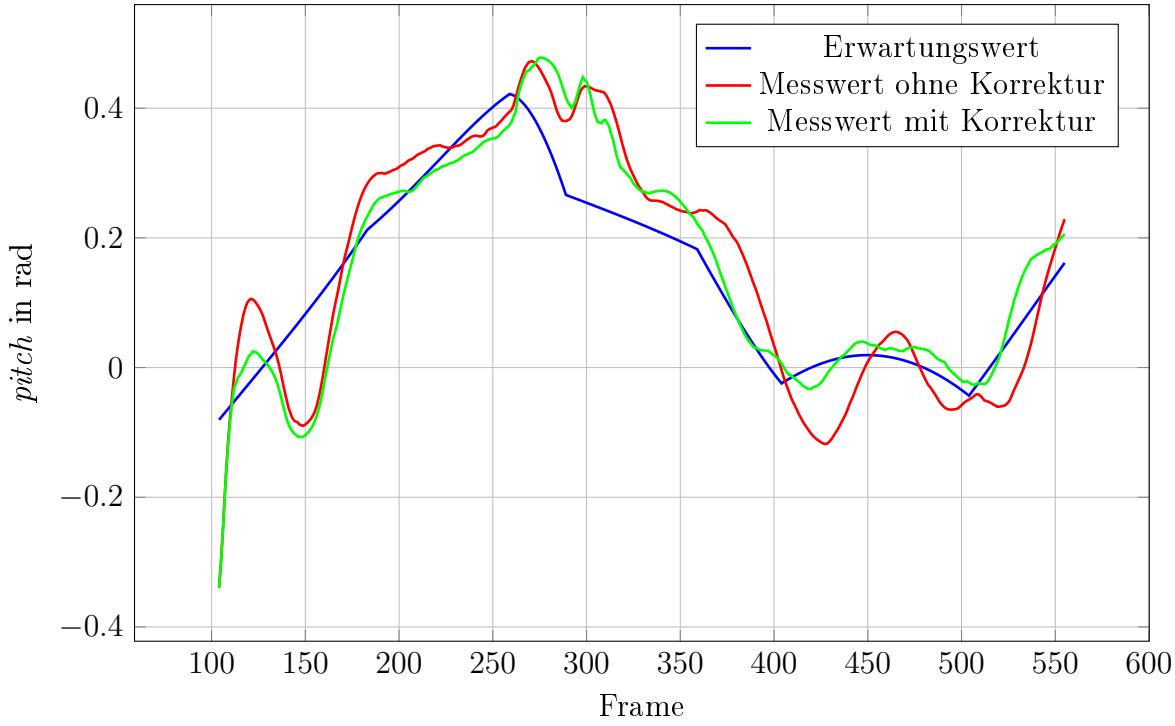


Abbildung 4.2: Messwerte der Torso-*pitch*-Winkel mit und ohne *pitch*-Korrektur im Vergleich zum Erwartungswert.

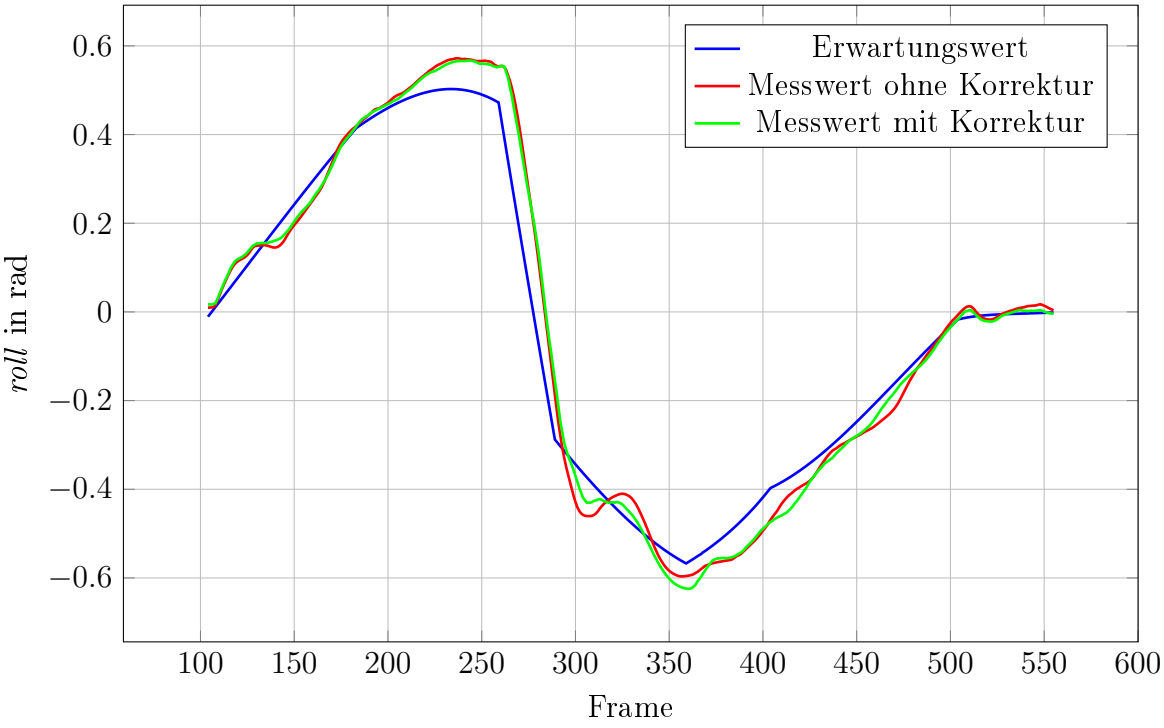


Abbildung 4.3: Messwerte der Torso-*roll*-Winkel mit und ohne *pitch*- und *roll*-Korrektur im Vergleich zum Erwartungswert.

5 Evaluation

In dieser Evaluation wird überprüft, ob die in Kapitel 1.2 definierten Ziele dieser Arbeit erreicht wurden. Dafür wurde für jedes der Ziele und der in Kapitel 3.2 genannten Probleme eine Versuchsreihe durchgeführt. In jeder der Versuchsreihen wurden dieselben drei Roboter genutzt. Dabei wurden diese so ausgewählt, dass deren Zustand unterschiedlich ist, um die Erkenntnisse auf möglichst viele Roboter übertragen zu können. Jeder der Roboter musste fünfmal aus der Rückenlage aufstehen. Alle Versuche fanden, wenn nicht anders erwähnt, auf einem ebenen Linoleumboden statt.
In die Auswertung der Versuche fließt zum einen die Anzahl der fehlgeschlagenen Aufstehversuche ein. Zum anderen wird die Schwankung in *pitch*-Richtung beachtet, da diese, wie bereits erwähnt, ein Hauptverursacher der Instabilität ist.

5.1 Stabilisierung

Zunächst sollte überprüft werden, ob die Schwankungen der Roboter beim Aufstehen, durch das entwickelte Verfahren verringert werden. Da das Verfahren auf Grundlage der *getup_back_slow*-Bewegung entworfen wurde, wurde diese Bewegung auch beim ersten Test verwendet.
Dabei hat sich gezeigt, dass alle Roboter, unabhängig vom Zustand, sowohl mit als auch ohne Korrekturen in der Lage waren aufzustehen. Bei keinem der Versuche ist ein Roboter gefallen. Die durchschnittlichen *pitch*-Werte der Roboter ohne Korrektur sind in Diagramm 5.1 zu sehen. In Diagramm 5.2 sind die Werte mit Korrektur abgebildet. Bei einem Vergleich der Graphen fällt auf, dass die Schwankungen durch die Korrektur durchweg reduziert wurden und sich die Werte näher an den Erwartungen befinden. Vor allem die starken Schwankungen zwischen Frame 100 und 200, welche durch das schnelle Aufrichten aus der Rückenlage entstehen, sind deutlich schwächer. Aber auch die Schwankungen zwischen Frame 400 und 500 sind nicht mehr so intensiv. Weiterhin sind die Unterschiede zwischen den Robotern nicht mehr so stark. Roboter 2 und 3 verhalten sich sogar nahezu identisch, lediglich Roboter 1 weicht ein wenig von ihnen ab.

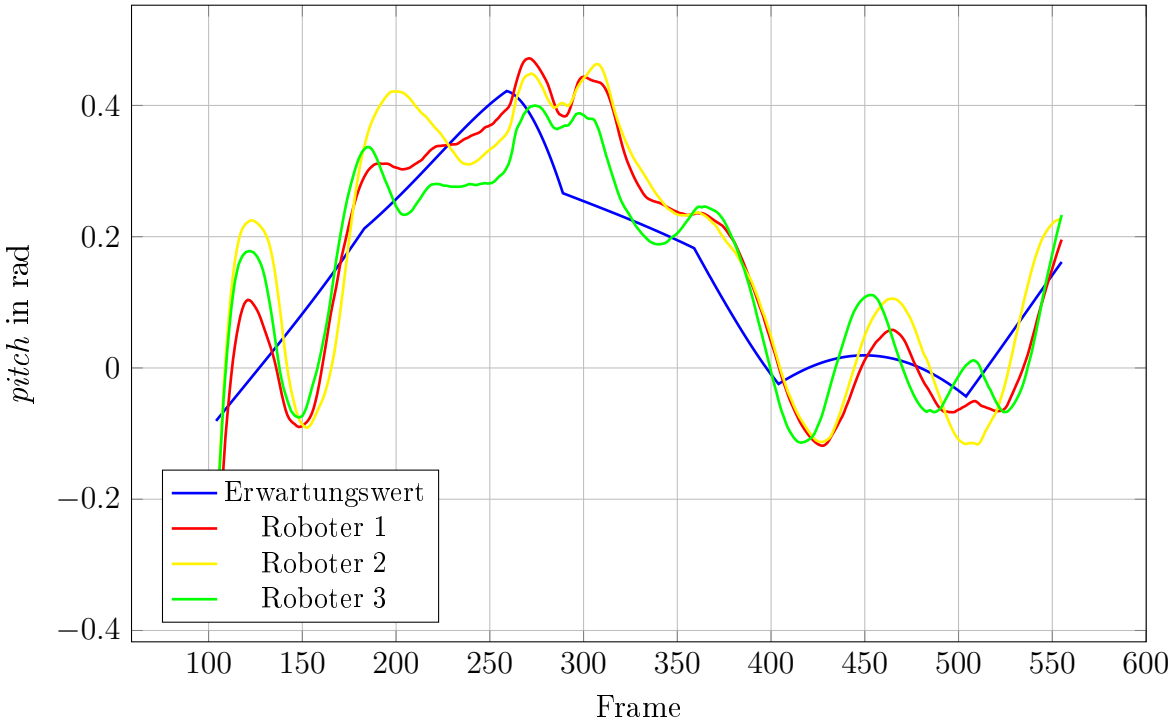


Abbildung 5.1: Durchschnittliche Torso-*pitch*-Winkel der Testroboter beim Aufstehen mit der *getup_back_slow*-Bewegung ohne Korrektur.

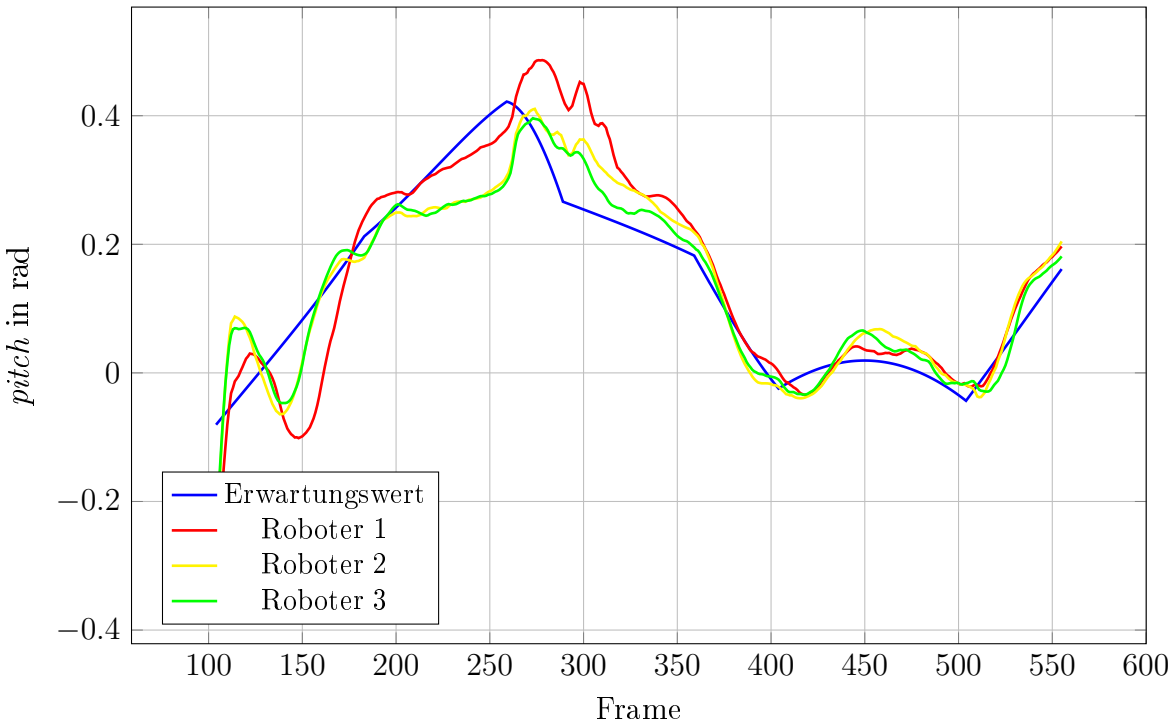


Abbildung 5.2: Durchschnittliche Torso-*pitch*-Winkel der Testroboter beim Aufstehen mit der *getup_back_slow*-Bewegung mit Korrektur.

Dass sich Roboter 1 von den anderen beiden unterscheidet, liegt daran, dass dieser Roboter der älteste ist. Durch die damit verbundene Abnutzung hat dieser das größte Spiel in seinen Gelenken und kann deshalb leichter in die Richtung der größten Beschleunigung kippen. Außerdem fällt auf, dass Roboter 2 und 3 es nicht schaffen, die Erwartungswerte zwischen Frame 200 und 250 zu erreichen. Roboter 2 schneidet an dieser Stelle besser ab, liegt aber anschließend zwischen Frame 275 und 325 deutlich über der Erwartung.
Insgesamt hat sich die durchschnittliche Abweichung vom Erwartungswert bei allen Robotern verbessert. Bei Roboter 1 von 0.045 rad auf 0.027 rad, bei Roboter 2 von 0.063 rad auf 0.011 rad und bei Roboter 3 von 0.027 rad auf 0.01 rad.

5.1.1 Kompensation für inaktive Gelenke

Nachdem in Test eins gezeigt wurde, dass das entwickelte Verfahren die Bewegung im Allgemeinen stabilisiert, sollte im zweiten Test überprüft werden, wie gut Gelenke kompensiert werden können, welche nicht ihren Sollwinkel erreichen. Dafür wurde das *HipYawPitch*-Gelenk zwischen Frame 400 und 500 festgestellt. Das Gelenk wurde gewählt, da dieses bei Überbelastung am ehesten anfängt zu hängen und die Sollwinkel nicht mehr erreicht [16]. Der Zeitbereich wurde so gewählt, dass der Roboter das Gelenk benötigt, um stabil zu bleiben.

Roboter	**Resultat**	
	ohne Korrektur	**mit Korrektur**
Roboter 1	2/5	5/5
Roboter 2	0/5	5/5
Roboter 3	0/5	5/5

Tabelle 5.1: Erfolgsraten aus Test zwei beim Aufstehen mit inaktivem *HipYawPitch*-Gelenk.

Die Resultate des zweiten Tests sind in Tabelle 5.1 einzusehen. Ohne Korrektur ist es nur Roboter 1 zweimal gelungen erfolgreich aufzustehen. Die anderen Roboter haben es ohne Korrektur gar nicht geschafft. Durch die Korrektur konnte die Erfolgsrate auf 100 % gesteigert werden. Dies zeigt deutlich, dass die Korrektur über die Torso-Winkel und Winkelgeschwindigkeiten gut dazu genutzt werden kann, hängende Gelenke auszugleichen. In Diagramm 5.3 wird der Verlauf der *pitch*-Winkel aus jeweils einem Versuch mit und ohne Korrektur verglichen. Dort lässt sich erkennen, dass die Korrektur zwar dabei hilft die Bewegung erfolgreich durchzuführen, jedoch das hängende Gelenk nicht vollständig ausgleichen kann. Die Abweichung von dem Erwartungswert ist immer noch groß. Der Roboter überschreitet aber nicht den Kipppunkt. Ohne Korrektur wird der Kipppunkt

überschritten. Dadurch fällt der Roboter um, nachdem er sein *HipYawPitch*-Gelenk wieder bewegen kann und kurzzeitig beginnt seine *pitch*-Winkel zu verbessern.

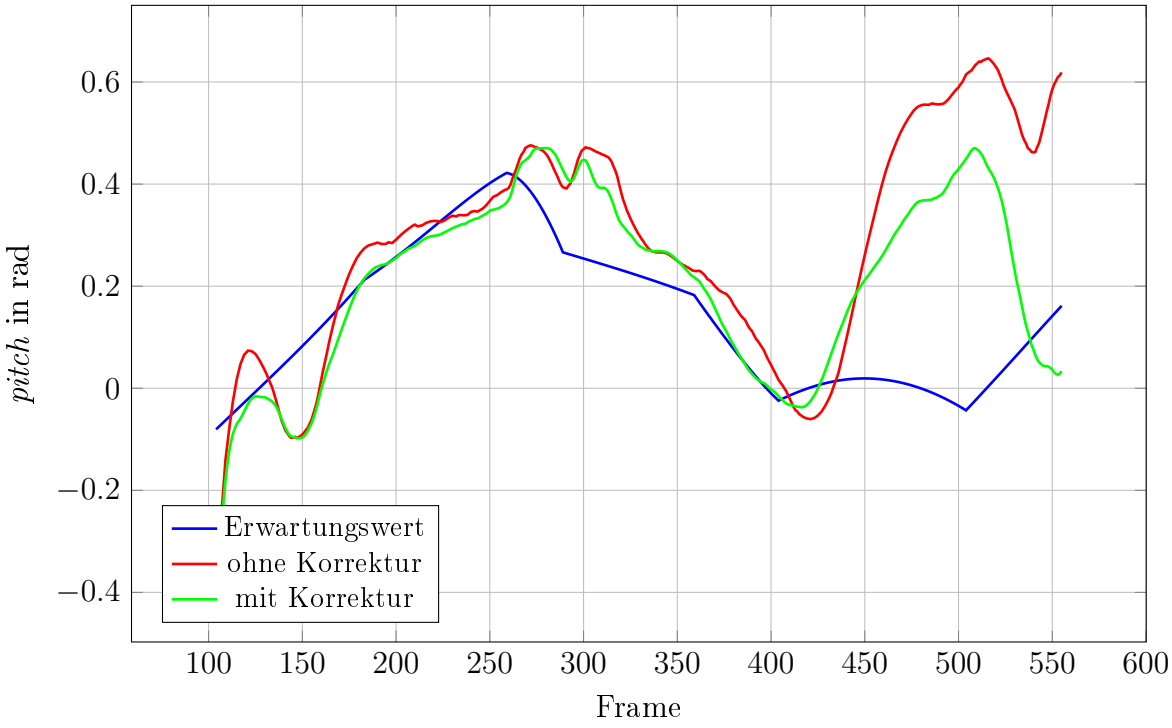


Abbildung 5.3: Torso-*pitch*-Winkel zweier Versuche beim Aufstehen mit inaktivem *HipYawPitch*-Gelenk. Ein Versuch mit und einer ohne Korrektur. Beide Versuche wurden mit Roboter 1 durchgeführt.

5.2 Übertragbarkeit

In den ersten beiden Tests wurde untersucht, ob das entwickelte Verfahren die in Kapitel 3.2 beschriebenen Probleme behebt. Nun sollte in Test drei untersucht werden, ob auch das Ziel der Übertragbarkeit erfüllt wird. Dafür wurde das Verfahren auf die Bewegung *getup_back* angewandt, wobei die Bewegung wie in Kapitel 4.1.3 angepasst und wie in Kapitel 4.1.1 die Sollwinkel berechnet werden mussten. Um die Übertragung möglichst einfach zu halten, wurden die Werte der PI-Regler nicht angepasst.
Bei allen Versuchen haben es die Roboter geschafft erfolgreich aufzustehen. Die durchschnittlichen Werte der Versuche sind in den Diagrammen 5.4 und 5.5 dargestellt. Es lässt sich auch hier erkennen, dass Roboter 1 sich aufgrund seines Alters ohne Korrektur etwas anders verhält. Mit der Korrektur verhalten sich alle Roboter ähnlicher, wobei die Abweichungen zwischen den Robotern jedoch höher sind, als bei *getup_back_slow*. Das liegt zum einen daran, dass diese Bewegung generell schneller abläuft und somit weniger Zeit zur Detektion von Abweichungen und Korrektur bleibt. Zum anderen gibt es hier zu

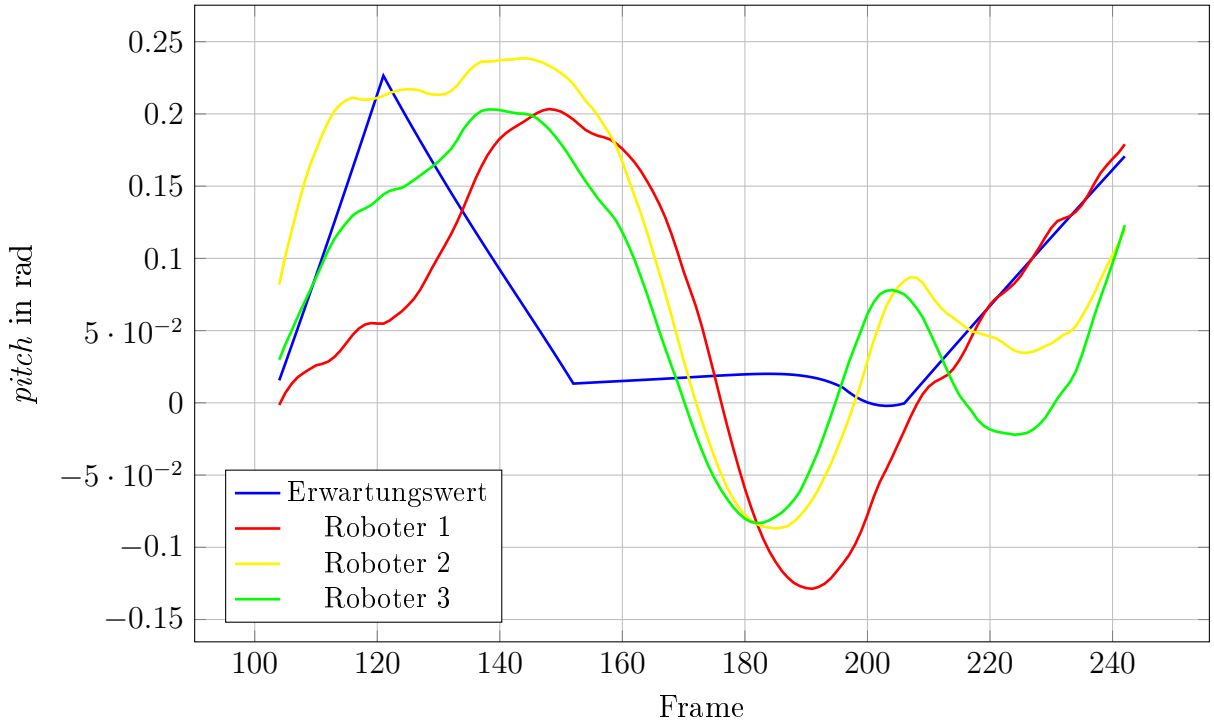


Abbildung 5.4: Durchschnittliche Torso-*pitch*-Winkel der Testroboter beim Aufstehen mit der *getup_back*-Bewegung ohne Korrektur.

Beginn bei Frame 120 eine abrupte Richtungsänderung, welche sich aufgrund der Trägheit des Roboters nicht so umsetzen lässt.
Dennoch hat sich durch die Korrektur die durchschnittliche Abweichung bei Roboter 1 von 0.031 rad auf 0.007 rad, bei Roboter 2 von 0.053 rad auf 0.04 rad und bei Roboter 3 von 0.02 rad auf 0.014 rad reduziert. Diese Verbesserung ist zwar nicht so gut wie in Test 1, aber dennoch wird durch das Verfahren die Aufstehbewegung stabiler und planbarer.

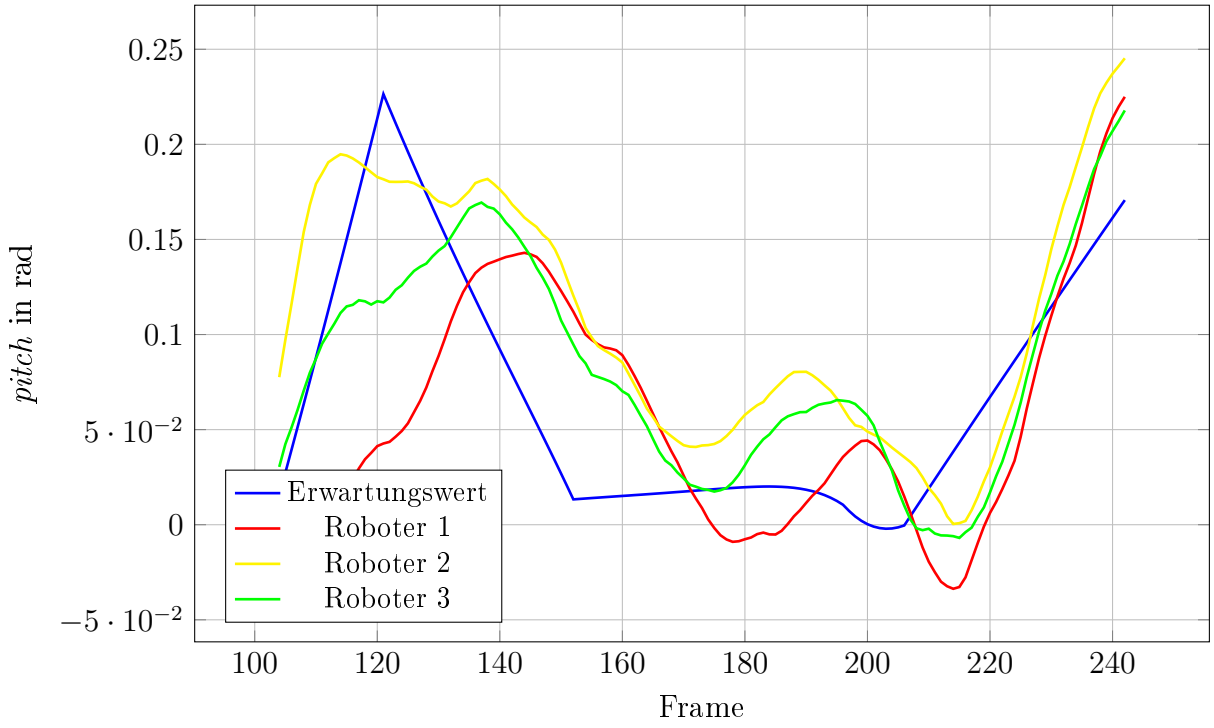


ABBILDUNG 5.5: Durchschnittliche Torso-*pitch*-Winkel der Testroboter beim Aufstehen mit der *getup_back*-Bewegung mit Korrektur.

5.3 Einfluss äußerer Störfaktoren

Ein weiterer Faktor, der getestet werden muss, sind äußere Einflüsse. In einem Fußballspiel kann es zu ungewollten Stößen zwischen Robotern kommen. Daher sollte getestet werden, ob das entwickelte Verfahren solchen Störfaktoren widerstehen kann.
Um diese Stöße zu simulieren, wurde das gleiche Verfahren wie bei [16] verwendet. Dafür wurde eine Flasche mit einem Faden an einem Tor aufgehangen und während des Aufstehens gegen den Roboter geschwungen. Bei [16] wurde ein Gewicht von 520 g gewählt. Aufgrund des höheren Gewichts des NAO v6 wurde das Gewicht der Flasche auf 540 g erhöht. Der Aufbau ist in Abbildung 5.6 zu sehen.
Bei jedem Versuch wurde ein Roboter auf den Rücken gelegt. Von dort ist dieser aufgestanden. Während des Aufstehens wurde die Flasche, aus einer Auslenkung von circa 80°, fallengelassen, sodass diese gegen den Torso des Roboters schwang.
In Tabelle 5.2 sind die Ergebnisse des Tests zu sehen. Daraus lässt sich erkennen, dass auch hier die Korrekturen das Aufstehverfahren deutlich resilienter macht. Lediglich bei einem Versuch mit Korrektur ist das Aufstehen fehlgeschlagen. Das lag in diesem Fall daran, dass die Flasche den Roboter in dem Moment getroffen hat, als dieser in der ersten Schwungphase sein Maximum erreicht hatte. Dadurch wurde der Roboter weiter nach

Roboter	Resultat	
	ohne Korrektur	mit Korrektur
Roboter 1	2/5	5/5
Roboter 2	3/5	5/5
Roboter 3	2/5	4/5

Tabelle 5.2: Erfolgsraten des Stoßtests.

vorne beschleunigt, bevor dieser nach hinten korrigieren konnte. In den meisten Fällen in denen es der Roboter ohne Korrektur geschafft hat erfolgreich aufzustehen, traf die Flasche den Roboter in einem Moment als dieser seinen Torso nach hinten bewegte.

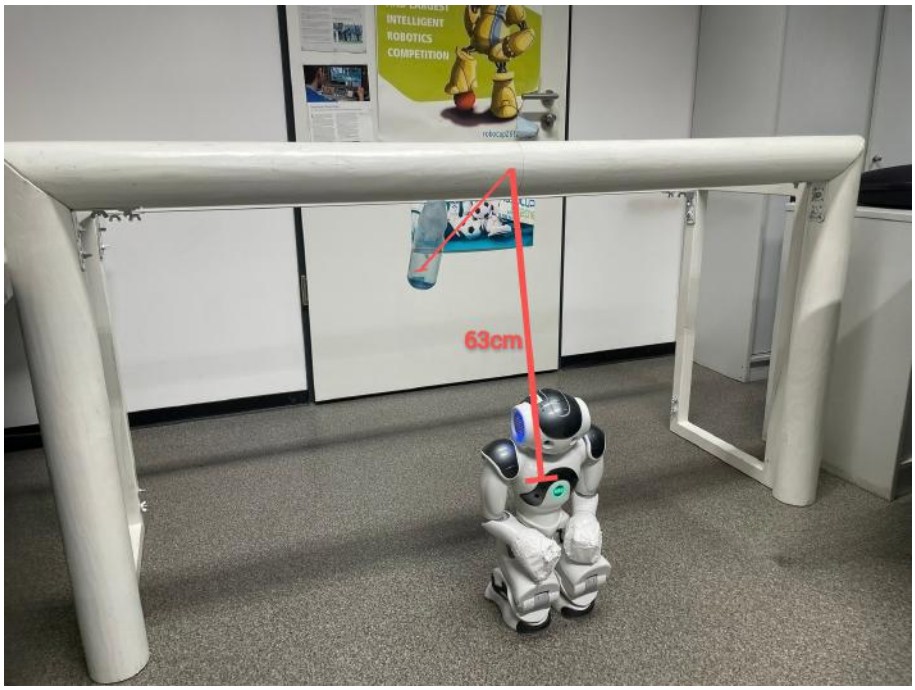


Abbildung 5.6: Aufbau des Stoßtests

In Diagramm 5.7 sind zwei Versuche desselben Roboters zu sehen. In diesen wurde der Roboter zu einem ähnlichen Zeitpunkt, einmal mit und einmal ohne Korrektur getroffen. Der Stoß erfolgte so, dass in die aktuelle Bewegungsrichtung des Roboters beschleunigt wurde. Bei dem Versuch mit Korrektur erfolgte der Stoß circa in Frame 120, bei dem ohne circa in Frame 125. Die Zeitpunkte lassen sich an der deutlichen Winkeländerung in Bewegungsrichtung erkennen. Wie gut zu sehen ist, konnte es der Roboter durch die Korrektur vermeiden den Kipppunkt zu überschreiten, während der Roboter ohne Korrektur hinfiel. Das Absinken des *pitch*-Winkels zum Ende des Versuchs ohne Korrektur, liegt daran, dass der Roboter aufgefangen wurde, bevor dieser auf dem Boden auftraf, um Schaden zu vermeiden.
Insgesamt lässt sich aus dem Test schließen, dass die Korrektur das Aufstehen auch bei Stößen stabilisiert. Es ist aber auch zu sehen, dass nicht alle Stöße abgefangen werden

können. Der Zeitpunkt des Stoßes ist als ein weiterer Faktor für den Erfolg des Aufstehens relevant, da es spezielle Zeitbereiche gibt, in denen die Korrektur schwer möglich ist.

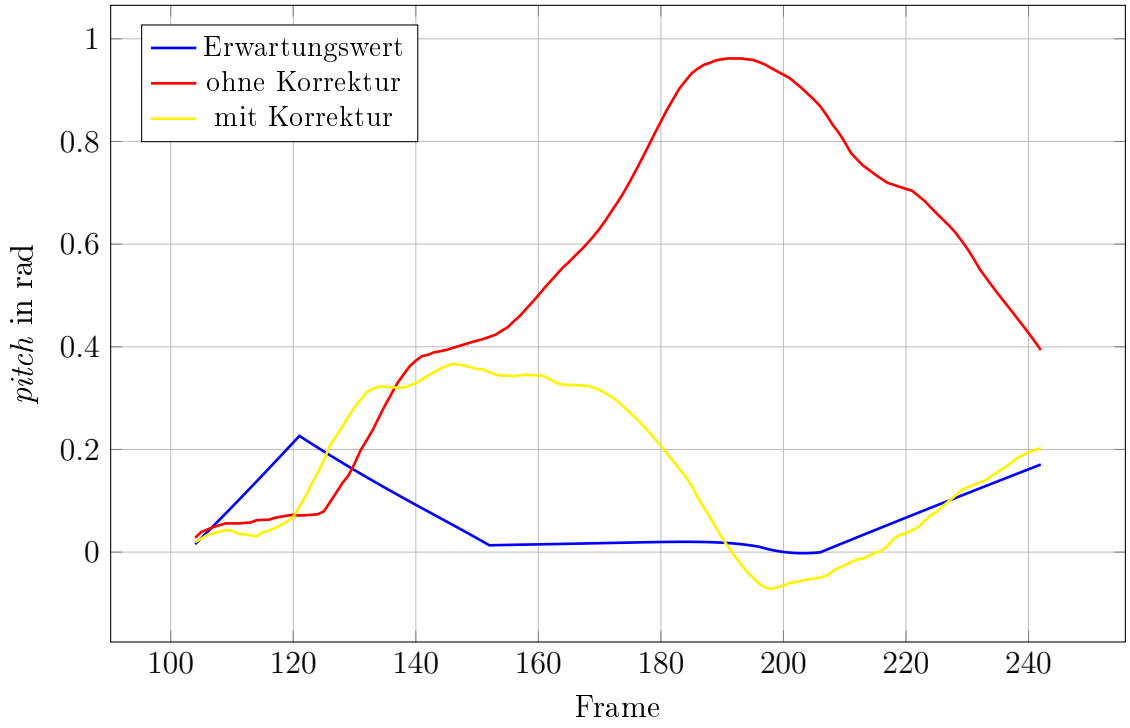


Abbildung 5.7: Torso-*pitch*-Winkel zweier Versuche beim Stoßtest. Ein Versuch mit und einer ohne Korrektur. Beide Versuche wurden mit Roboter 1 durchgeführt.

5.4 Einfluss des Untergrunds

Zuletzt wurde noch der Einfluss des Untergrundes getestet. Die bisherigen Tests wurden auf einem Linoleumboden durchgeführt. Dieser eignet sich gut, um das generelle Prinzip des Korrekturverfahrens zu überprüfen, da er, im Vergleich zu Kunstrasen, stabil und unnachgiebig ist und somit äußere Einflussfaktoren eliminiert. Während eines Spiels stehen und laufen die Roboter jedoch auf Kunstrasen. Dadurch, dass dieser nachgiebiger ist, wird die Stabilität negativ beeinflusst. Um zu überprüfen, wie gut die in dieser Arbeit entwickelte Korrektur ist, mussten die Roboter auf Kunstrasen aufstehen. Dafür wurde die *getup_back*-Bewegung genutzt, da diese auf Kunstrasen, wie in Kapitel 4 erwähnt, bisher nicht zuverlässig funktioniert hat.
Die Ergebnisse sind in Tabelle 5.3 zusammengefasst. Überraschenderweise hat es der älteste Roboter, Roboter 1, auch ohne Korrektur geschafft alle fünf Versuche aufzustehen. Allerdings hat der Roboter während aller Versuche stark geschwankt und es nur gerade so geschafft. Das lag vor allem daran, dass sich durch den weicheren Boden die Schwankungen verlängert und verschoben haben.

Roboter	Resultat	
	ohne Korrektur	mit Korrektur
Roboter 1	5/5	5/5
Roboter 2	2/5	5/5
Roboter 3	1/5	5/5

TABELLE 5.3: Erfolgsraten aus Test vier beim Aufstehen mit der *getup_back*-Bewegung auf Kunstrasen.

In Diagramm 5.8 sind die Durchschnittswerte auf Rasen- und Linoleumboden von Roboter 1 gegenübergestellt. Dass es der älteste Roboter geschafft hat, scheint daran zu liegen, dass dieser die stehende Position mit weniger Schwung erreicht. Dadurch kippt dieser nicht so schnell nach vorne. Die anderen Roboter sind in der Regel nach vorne gekippt, wenn sie umgefallen sind. Durch den weicheren Boden, drücken sich die Kanten der Füße in den Boden, wodurch der Roboter schneller den Kipppunkt erreicht.

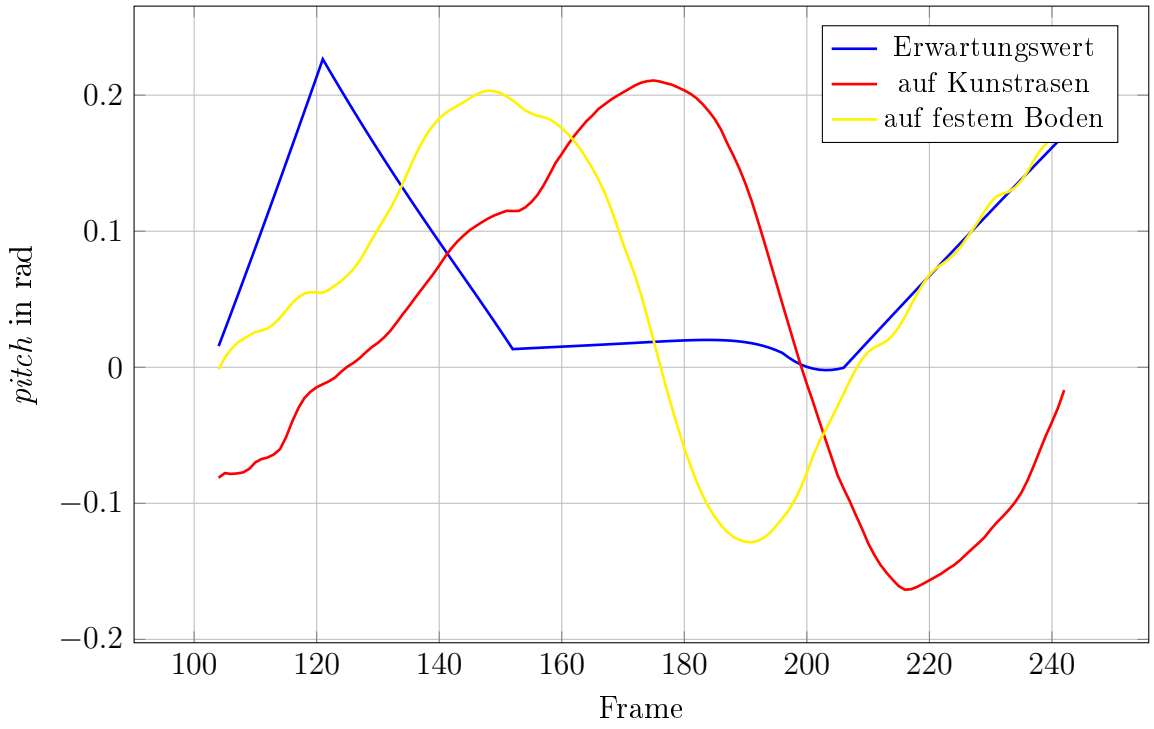


ABBILDUNG 5.8: Torso-*pitch*-Winkel zweier Versuche beim Aufstehen mit der *getup_back*-Bewegung ohne Korrektur. Ein Versuch auf Kunstrasen und einer auf Linoleumboden. Beide Versuche wurden mit Roboter 1 durchgeführt.

Durch die Korrektur konnten die Roboter zwar alle aufstehen, jedoch haben diese alle angefangen zu zittern. Ursache hierfür war die Korrektur mit den Füßen. Während der Roboter die Beine aus der gespreizten Position zusammengezogen hat, haben sich diese in den Boden gegraben. Wenn der Roboter beispielsweise weit nach vorne gelehnt war, wurden die Fußspitzen nach unten bewegt um die Position auszugleichen. Wenn der Roboter die Beine nicht bewegt oder wenn der Boden fest ist, ist das kein Problem. Beim

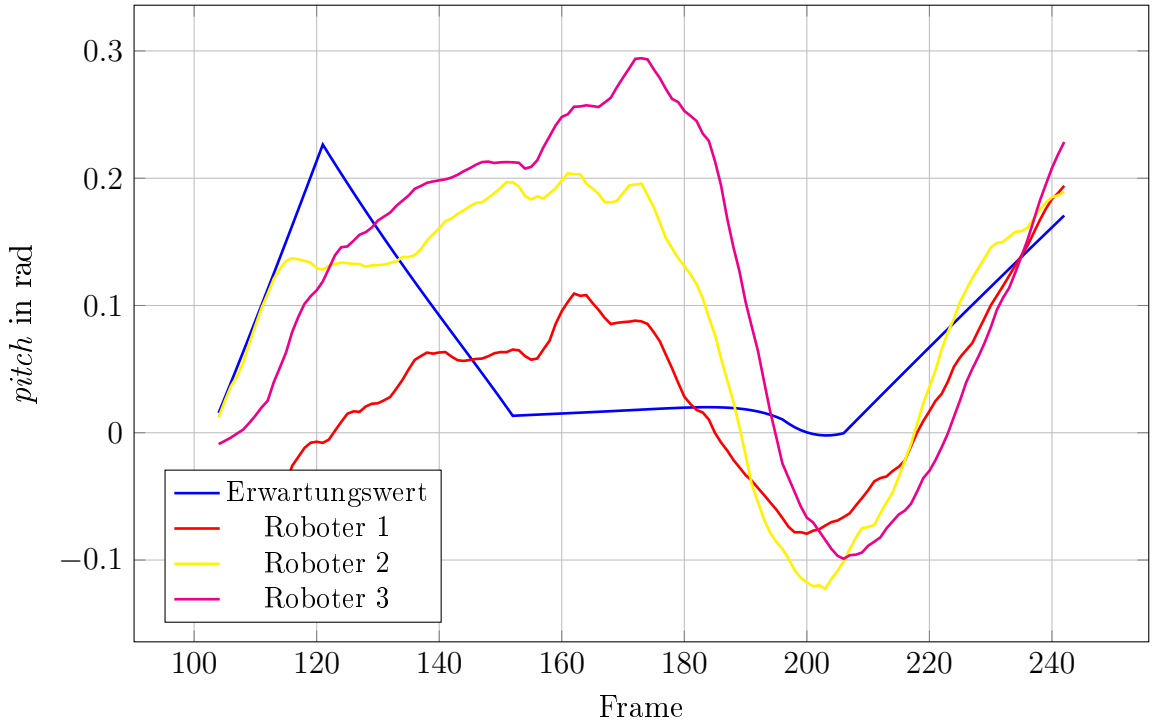


Abbildung 5.9: Torso-*pitch*-Winkel der Testroboter bei jeweils einem Versuch beim Aufstehen mit Korrektur auf Kunstrasen

Kunstrasen bleiben die Füße jedoch durch die Korrektur vermehrt im Kunstrasen hängen, was das Zittern verursacht. Dieses Zittern ist auch in den *pitch*-Winkeln der Roboter zu sehen. In Diagramm 5.9 ist jeweils ein Versuch von jedem Roboter dargestellt, in dem das Zittern gut erkennbar ist. Das Zittern tritt hier vor allem zwischen den Frames 120 und 180 auf, da der Roboter in dieser Zeit seine Beine zusammenzieht.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Verfahren zur Stabilisierung der Aufstehbewegungen humanoider Roboter entwickelt und erprobt. Ziel war es, die bestehenden Bewegungsabläufe der HTWK-Robots zu verbessern, indem Schwankungen während des Aufstehens reduziert und äußere Störungen ausgeglichen werden. Das entwickelte System nutzt einen hybriden Ansatz aus *Ankle* und *Hip Strategy* in Kombination mit vier PI-Reglern zur Anpassung von *pitch*- und *roll*-Winkeln des Oberkörpers. Dafür wurden kinematische Berechnungen mithilfe von Quaternionen implementiert, um die theoretischen Zielwinkel der Bewegung zu bestimmen. Weiterhin wurden die Bewegungen angepasst, sodass diese alle Voraussetzungen für das Verfahren erfüllen. Außerdem wurde ein System implementiert um die Auswahl der Gelenke zu treffen und die Verteilung der Korrekturen zwischen diesen zu gewichten.
Die Evaluation des entwickelten Stabilisierungssystems für die Aufstehbewegungen umfasste mehrere Tests. Diese dienten dazu, das Verhalten des Systems unter verschiedenen Bedingungen zu analysieren. Dabei hat sich gezeigt, dass das Verfahren das Aufstehen in allen untersuchten Situationen robuster machte. So wurde unter anderem gezeigt, dass sowohl inaktive Gelenke ausgeglichen als auch Stöße abgefedert werden können. Außerdem ließ sich das Verfahren einfach auf eine andere Aufstehbewegung übertragen. Insgesamt über alle Tests hinweg ließ sich die Erfolgsrate des Aufstehens von 62.67 % ohne Korrekturen auf 98.67 % mit Korrekturen steigern.

Auch wenn das hier entwickelte Verfahren bereits gute Ergebnisse liefert, bieten sich für zukünftige Arbeiten mehrere Erweiterungsmöglichkeiten. Zum einen funktioniert das Aufstehen auf dem Rasen noch nicht ausreichend gut. Das Zittern, was durch das Hängenbleiben am Rasen ausgelöst wird, könnte durch eine Anpassung der Gelenk-Gewichtung verbessert werden. Eine weitere Verbesserungsmöglichkeit bietet die Einbindung der Gelenk-Temperaturen in die Gelenk-Gewichtung. Dadurch könnten zu heiße, und damit weniger leistungsfähigere Gelenke, entlastet werden. Ebenso könnte der Ansatz auf andere Bewegungsabläufe im RoboCup wie das Schießen angewendet werden. Auch ist die Einbindung weiterer Sensordaten oder die Berücksichtigung der aktuellen Beschleunigung des Roboters denkbar. Zuletzt fehlt noch ein automatisiertes Abbrechen des Aufstehens. Das

Korrekturverfahren lässt sich zwar auf andere Aufstehbewegungen übertragen, jedoch ist das Abbrechen der Bewegung bei einem Fehlschlag noch abhängig von langwierig gesammelten Realdaten. Dort müsste ein automatisierter Prozess entwickelt werden, welcher entscheidet, wann ein Roboter einen Aufstehversuch abbrechen muss.

Literaturverzeichnis

[1] R. Raj and A. Kos, "A Comprehensive Study of Mobile Robot: History, Developments, Applications, and Future Research Perspectives," *Applied Sciences*, vol. 12, no. 14, p. 6951, Jul. 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/14/6951

[2] B-Human, "Robocup_2024_Falls," 2024, abgerufen am: 07.10.2024. [Online]. Available: https://b-human.informatik.uni-bremen.de/public/Statistics/2024/RoboCup_2024_Falls.ods

[3] HTWK-Robots, "RoboCup 2024 - HTWK Robots vs. B-Human (Standard Platform League, Group Phase Round 1)," 2024, abgerufen am: 07.10.2024. [Online]. Available: https://www.youtube.com/watch?v=AhgxBXd_R44

[4] "A Brief History of RoboCup," abgerufen am: 07.10.2024. [Online]. Available: https://www.robocup.org/a_brief_history_of_robocup

[5] "Objective," abgerufen am: 07.10.2024. [Online]. Available: https://www.robocup.org/objective

[6] R. T. Committee, "Robocup Standard Platform League (NAO) Rule Book," 2024, abgerufen am: 07.10.2024. [Online]. Available: https://spl.robocup.org/wp-content/uploads/SPL-Rules-master.pdf

[7] S. Robotics, "NAO - Developer Guide ," abgerufen am: 07.10.2024. [Online]. Available: http://doc.aldebaran.com/2-8/family/nao_technical/index_naov6.html

[8] S. Kucuk and Z. Bingul, *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher London, UK, 2006.

[9] B. Vilhena Adorno, "Robot Kinematic Modeling and Control Based on Dual Quaternion Algebra — Part I: Fundamentals." Feb. 2017, working paper or preprint. [Online]. Available: https://hal.science/hal-01478225

[10] J. Diebel *et al.*, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[11] A. Visioli, *Practical PID control*. Springer Science & Business Media, 2006.

[12] M. Vukobratović and D. Juričić, "Contribution to the Synthesis of Biped Gait," *IFAC Proceedings Volumes*, vol. 2, no. 4, pp. 469–478, Sep. 1968. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1474667017688918

[13] S. Zhao, Z. Jiang, Y. Li, J. Xu, and C. Wang, "Key Components and Future Development Analysis of Humanoid Robots," in *2024 8th International Conference on Robotics, Control and Automation (ICRCA)*. Shanghai, China: IEEE, Jan. 2024, pp. 140–147. [Online]. Available: https://ieeexplore.ieee.org/document/10649218/

[14] P. Sardain and G. Bessonnet, "Forces Acting on a Biped Robot. Center of Pressure—Zero Moment Point," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 34, no. 5, pp. 630–637, Sep. 2004. [Online]. Available: http://ieeexplore.ieee.org/document/1325327/

[15] A. Böckmann and T. Laue, "Kick Motions for the NAO Robot Using Dynamic Movement Primitives," in *RoboCup 2016: Robot World Cup XX*, S. Behnke, R. Sheh, S. Sarıel, and D. D. Lee, Eds. Cham: Springer International Publishing, 2017, vol. 9776, pp. 33–44, series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-319-68792-6_3

[16] P. Reichenberg, "Entwicklung eines dynamischen Aufstehens unter Vermeidung inkorrekter Bewegungszustände," 2020, abgerufen am: 09.10.2024. [Online]. Available: https://b-human.de/downloads/theses/bachelor-thesis-philip.pdf

[17] S. Shamsuddin, L. I. Ismail, H. Yussof, N. Ismarrubie Zahari, S. Bahari, H. Hashim, and A. Jaffar, "Humanoid robot NAO: Review of control and motion exploration," in *2011 IEEE International Conference on Control System, Computing and Engineering*. Penang, Malaysia: IEEE, Nov. 2011, pp. 511–516. [Online]. Available: http://ieeexplore.ieee.org/document/6190579/

[18] A. K. Kashyap and D. R. Parhi, "Optimization of stability of humanoid robot NAO using ant colony optimization tuned MPC controller for uneven path," *Soft Computing*, vol. 25, no. 7, pp. 5131–5150, Apr. 2021. [Online]. Available: http://link.springer.com/10.1007/s00500-020-05515-1

[19] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Resolved momentum control: humanoid motion planning based on

the linear and angular momentum," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 2. Las Vegas, NV, USA: IEEE, 2003, pp. 1644–1650. [Online]. Available: http://ieeexplore.ieee.org/document/1248880/

[20] S. Katayama, M. Murooka, and Y. Tazaki, "Model predictive control of legged and humanoid robots: models and algorithms," *Advanced Robotics*, vol. 37, no. 5, pp. 298–315, Mar. 2023. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/01691864.2023.2168134

[21] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo, "MPC for Humanoid Gait Generation: Stability and Feasibility," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1171–1188, Aug. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8955951/

[22] R. Schuller, G. Mesesan, J. Englsberger, J. Lee, and C. Ott, "Online Centroidal Angular Momentum Reference Generation and Motion Optimization for Humanoid Push Recovery," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5689–5696, Jul. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9435940/

[23] B. Stephens, "Humanoid push recovery," in *2007 7th IEEE-RAS International Conference on Humanoid Robots*. Pittsburgh, PA, USA: IEEE, Nov. 2007, pp. 589–595. [Online]. Available: http://ieeexplore.ieee.org/document/4813931/

[24] Y. Gong and J. Grizzle, "One-Step Ahead Prediction of Angular Momentum about the Contact Point for Control of Bipedal Locomotion: Validation in a LIP-inspired Controller," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 2832–2838. [Online]. Available: https://ieeexplore.ieee.org/document/9560821/

[25] S.-H. Lee and A. Goswami, "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground," *Autonomous Robots*, vol. 33, no. 4, pp. 399–414, Nov. 2012. [Online]. Available: http://link.springer.com/10.1007/s10514-012-9294-z

[26] G. Wiedebach, S. Bertrand, T. Wu, L. Fiorio, S. McCrory, R. Griffin, F. Nori, and J. Pratt, "Walking on partial footholds including line contacts with the humanoid robot atlas," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. Cancun, Mexico: IEEE, Nov. 2016, pp. 1312–1319. [Online]. Available: http://ieeexplore.ieee.org/document/7803439/

[27] R. Hinata and D. N. Nenchev, "Balance Stabilization with Angular Momentum Damping Derived from the Reaction Null-Space," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. Beijing, China: IEEE, Nov. 2018, pp. 188–195. [Online]. Available: https://ieeexplore.ieee.org/document/8624933/

[28] A. A. Saputra, I. A. Sulistijono, A. S. Khalilullah, T. Takeda, and N. Kubota, "Combining pose control and angular velocity control for motion balance of humanoid robot soccer EROS," in *2014 IEEE Symposium on Robotic Intelligence in Informationally Structured Space (RiiSS)*. Orlando, FL, USA: IEEE, Dec. 2014, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/7009164/

[29] A. Kuo, "An optimal control model for analyzing human postural balance," *IEEE Transactions on Biomedical Engineering*, vol. 42, no. 1, pp. 87–101, Jan. 1995. [Online]. Available: http://ieeexplore.ieee.org/document/362914/

[30] B. Stephens, "Integral control of humanoid balance," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. San Diego, CA, USA: IEEE, Oct. 2007, pp. 4020–4027. [Online]. Available: http://ieeexplore.ieee.org/document/4399407/

[31] A. K. Kashyap, D. R. Parhi, and S. Kumar, "Dynamic Stabilization of NAO Humanoid Robot Based on Whole-Body Control with Simulated Annealing," *International Journal of Humanoid Robotics*, vol. 17, no. 03, p. 2050014, Jun. 2020. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/S0219843620500140

[32] K.-N.-K. Nguyen, Y. Kojio, S. Noda, F. Sugai, K. Kojima, Y. Kakiuchi, K. Okada, and M. Inaba, "Dynamic Fall Recovery Motion Generation on Biped Robot With Shell Protector," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6741–6748, Oct. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9472986/

[33] M. Depinet, P. MacAlpine, and P. Stone, "Keyframe Sampling, Optimization, and Behavior Integration: Towards Long-Distance Kicking in the RoboCup 3D Simulation League," in *RoboCup 2014: Robot World Cup XVIII*, R. A. C. Bianchi, H. L. Akin, S. Ramamoorthy, and K. Sugiura, Eds. Cham: Springer International Publishing, 2015, vol. 8992, pp. 571–582, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-18615-3_47

[34] C. Sahu, D. R. Parhi, P. B. Kumar, M. K. Muni, A. Chhotray, and K. K. Pandey, "Humanoid NAO: A Kinematic Encounter," *Robotica*, vol. 39, no. 11, pp. 1997–2007,

Nov. 2021. [Online]. Available: https://www.cambridge.org/core/product/identifier/S0263574721000096/type/journal_article

[35] N. Kofinas, E. Orfanoudakis, and M. G. Lagoudakis, "Complete Analytical Forward and Inverse Kinematics for the NAO Humanoid Robot," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 2, pp. 251–264, Feb. 2015. [Online]. Available: http://link.springer.com/10.1007/s10846-013-0015-4

[36] V. E. Kremer, "Quaternions and slerp," in *Embots. dfki. de/doc/seminar ca/Kremer Quaternions. pdf*, 2008.

Abbildungsverzeichnis

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Graduierungsarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommene Stellen sind als solche einzeln kenntlich gemacht.

Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht worden.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

.....
Eric Behrendt
Leipzig, 21. November 2024