Ein stabiler Gang für humanoide, Fußball spielende Roboter

A stable gait for humanoid, soccer playing robots

Johannes Kulick

Masterarbeit an der
Freien Universität Berlin,
Institut für Informatik

Gutachter:
Prof. Dr. Raúl Rojas
Dr. Daniel Göhring

Berlin, 19. Oktober 2011

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Berlin, den 19. Oktober 2011

Zusammenfassung

Im RoboCup Soccer Wettbewerb treten humanoide Roboter beim Fußball gegeneinander an. Dabei ist die einzige erlaubte Fortbewegungsmethode der zweibeinige Gang. Diesem kommt daher eine wichtige Bedeutung zu.

In dieser Arbeit wurde für die neu konstruierten Roboter des Teams FUmanoid ein Gang entwickelt, der es den Robotern erlaubt, schnell in jede Richtung zu gehen. Dafür wurde die inverse Kinematik der Roboter gelöst und eine allgemeine inverse Kinematik für Parallelogramm-Mechaniken entworfen. Zur Gangerzeugung wurde durch Analyse von Passive Dynamic Walkern und mittels Zero Moment Point-basierten Trajektorienoptimierungen ein Laufsystem entwickelt. Darin integriert ist auch ein dynamischer Schuss. Methoden der Regelungstechnik kamen zum Einsatz, um des System weiter zu stabilisieren.

Das System wurde 2011 beim Robot Soccer WorldCup in der Liga der humanoiden KidSize Roboter vom Team FUmanoids eingesetzt, die den vierten Platz belegen konnten.

Abstract

The only permited locomotion in the Robot Soccer World Cup competition is the biped walk. It therefore is crucial for every team to have a stable and fast walking system.

In this masters thesis a walking system for the new constructed FUmanoid robots is developed, which enables them to move in every direction. For this purpose the inverse kinematics problem was solved and a generic inverse kinetic for parallelogram-mechanics is introduced. The actual gait is based on an analysis of Passive Dynamic Walkers and Zero Moment Point based trajectory optimizations. The system is also capable of kicking the ball during walk. Control theroy methods were used for further stabilizing.

The developed gait was used by the Team FUmanoids in the RoboCup 2011 competition, where they won the fourth place.

Inhaltsverzeichnis

1 Einleitung

Der zweibeinige, aufrechte Gang galt und gilt vielen als das Alleinstellungsmerkmal des Menschen gegenüber allen anderen Lebewesen auf der Erde. Schon früh wurde daher der Wunsch nach zweibeinigen Robotern oder Automaten laut. Bereits in der jungen Renaissance zeichnete Leonardo da Vinci Pläne für einen Roboter, dessen Rekonstruktion tatsächlich funktioniert (siehe Abb. 1.1(a)). Solche Maschinen regten die Fantasie der Menschen in allen Zeiten an, zum Beispiel in dem berühmten Film 'Metropolis' von Fritz Lang aus dem Jahr 1927 (siehe Abb. 1.1(b)).

Es dauerte aber bis in die 70er Jahre des 20. Jahrhunderts, bis solche zweibeinigen Roboter nicht nur wie da Vincis Modell einfache mechanisch vorgegebene Armbewegungen abfahren konnten oder sogar nur reine Fiktion waren.

Der erste humanoide Roboter, der gehen konnte, war der 1973 an der Waseda Universität in Japan entwickelte WABOT-1 (siehe Abb. 1.1(c)). Auf ihn folgten viele weitere Roboter, die auf zwei Beinen gehen konnten, bis hin zu den neueren Entwicklungen wie ASIMO, Nao oder HRP4 (siehe Abb. 1.1(d), 1.1(e), 1.1(f)).

Heute sind humanoide Roboter verschiedener Größe und Ausrichtungen in Forschung und Lehre überall auf der Welt im Einsatz. Seit 2004 werden zweibeinige Roboter auch bei Wettbewerben der Robot World Cup Initiative, kurz RoboCup, eingesetzt.

1.1 RoboCup

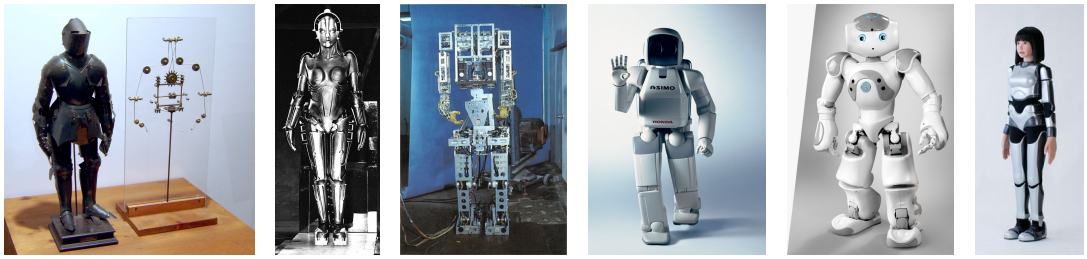
Nachdem sich in den frühen 90er Jahren des 20. Jahrhunderts abzeichnete, dass Schach als Referenz für künstliche Intelligenz zu einfach werden würde, begannen Forscher weltweit neue Felder aufzutun. Alan Mackworth schlug dabei 1993 als erster vor, Roboter Fußball spielen zu lassen [26].

Auch in Japan wurde Fußball als lohnendes Forschungsfeld für die Robotik erkannt, da es viele Felder vereint. Reichte beim Schach noch reines Expertenwissen gepaart mit Lernalgorithmen zur Optimierung aus, um auf Großmeisterniveau zu spielen, so sind beim Fußball Interaktion mit einem dynamischen Umfeld, die Verarbeitung von Sensordaten und die Kommunikation mit Mitspielern zusätzlich von Bedeutung.

Ebenfalls 1993 wurde daher in Japan die Robot J-League gegründet, eine Liga Fußball spielender Roboter. Das internationale Interesse an dieser Liga war aber so groß, dass der Fokus auf Japan bald fallen gelassen und die Liga in Robot World Cup umbenannt wurde. Nach zahlreichen technischen und organisatorischen Vorarbeiten wurde 1997 der erste offizielle RoboCup abgehalten.

Bis heute findet jedes Jahr die Weltmeisterschaft im RoboCup an wechselnden Orten statt. Das offizielle, ambitionierte Ziel ist es, den amtierenden FIFA Fußballweltmeister 2050 besiegen zu können.

Um den verschiedenen Teilbereichen der Forschung gerecht zu werden, wurden unterschiedliche Ligen gegründet. Ein wichtiges Unterscheidungsmerkmal von Ligen ist dabei, ob eine Liga in einer



(a) (b) (c) (d) (e) (f)

Abb. 1.1: Entwicklung humanoider Roboter: (a) Zweibeiniger Automat von Leonardo da Vinci (1495) (b) Roboter Maria aus Metropolis (1927) (c) WABOT-1 der Universität von Waseda (1973) (d) Hondas ASIMO (2000) (e) Aldebarans Nao (2006) (f) HRP-4C von AIST (2009). Bilder vom Hersteller des jeweiligen Roboters, bis auf (a) Quelle: http://de.wikipedia.org, (b) Metropolis (Film)

simulierten oder in einer realen Umgebung spielt. Die simulierten Ligen umfassen 2011 die beiden Unterligen 2D und 3D.

Auf der anderen Seite gibt es die Ligen, die in realen Umgebungen spielen und daher auch reale Roboter als Agenten einsetzen müssen. Hierbei gibt es zum einen die Small Size League und die Middle Size League, in denen Roboter auf Rädern zum Einsatz kommen. Zum anderen gibt es die Standard Platform League (SPL) und seit 2004 die Humanoid League, bei der zweibeinige Roboter eingesetzt werden. In der Vorgängerliga der SPL, der Four Legged League, wurde bis 2008 der vierbeinige Roboter Aibo der Firma Sony eingesetzt. Seitdem darf in der SPL ausschließlich der Nao Roboter der Firma Aldebaran genutzt werden. Die in der Humanoid League eingesetzten Roboter werden zusätzlich selbst konstruiert. Dabei gibt es drei Größen: die Kid Size (30-60 cm), die Teen Size (100-120 cm) und die Adult Size (größer als 130 cm).

Zusätzlich zu den Fußballligen wurden die beiden Wettbewerbe Rescue und @Home eingeführt. Bei der Rescue Liga treten Roboter an, die Einsätze in Katastrophensituationen durchführen müssen, bei @Home sind typische Aufgaben aus dem Alltag zu meistern. Zur Förderung der Lehre gibt es diese Wettbewerbe in vereinfachter Form auch als Junior Wettbewerbe, bei denen die Roboter von Schülern gebaut und programmiert werden.

Das Team FUmanoids, in dessen Rahmen diese Arbeit entstand, spielt in der humanoiden Kid Size Liga.

1.2 Humanoide Kid Size Liga

In der Kid Size Liga spielen die kleinsten Humanoiden des RoboCups. Ihre Größe beträgt zwischen 30 und 60 cm. Ihre Körpergröße definiert die maximalen Ausmaße der restlichen Körperteile, so dass ein möglichst menschenähnliches Aussehen erreicht wird.

Auch in der Sensorik wird eine möglichst große Ähnlichkeit zu Menschen angestrebt, daher sind nur passive Sensoren erlaubt. Dazu zählen unter anderem Drucksensoren oder Lage- und

Beschleunigungssensoren. Jegliche aktive Sensoren, wie beispielsweise Ultraschall- oder Lasersensoren, sind verboten.

Das Kamerasystem der Roboter soll ebenfalls dem menschliche Vorbild nahekommen: Es dürfen maximal zwei Kameras mit signifikantem Überschneidungswinkel und einem gesamten Blickwinkel nicht größer als 180° eingesetzt werden.

Gespielt wird 3 gegen 3 in zwei Halbzeiten zu je 10 min mit einer Pause von 5 min. Während des Spiels müssen alle Roboter völlig autonom agieren. Sie spielen auf einem 6×4 m großen, grünen Feld. Alle Objekte auf dem Feld sind zur Erleichterung farblich markiert. So ist der Ball orange, die Tore sind gelb bzw. blau und die Teams haben Markierungen in den Farben cyan und magenta. Das Feld besteht aus Teppichboden, der möglichst eben verlegt ist. Die weiteren Regeln basieren auf den FIFA Fußballregeln und wurden dem Entwicklungsstand der Roboter angepasst. Sie können unter [39] nachgelesen werden.

Während des Spiels ist die einzig erlaubte Fortbewegungsmethode der zweibeinige Gang. Ihm kommt in dieser Liga also eine herausragende Bedeutung zu. Wann immer der Ball gespielt werden soll, muss er gehend erreicht werden. Jeder Zweikampf passiert aus dem Gang heraus und keine Torchance kann ohne einen stabilen Gang erarbeitet werden. Den entscheidenen Unterschied zwischen zwei Teams in der humanoiden Liga macht nicht selten die Fähigkeit aus, besser oder schlechter gehen zu können.

Aus diesem Grund war es nötig für die neu konstruierten Roboter des Teams FUmanoids einen stabilen, schnellen und flexiblen Gang zu entwerfen, der die Fortbewegung in alle Richtungen erlaubt. Ein solcher Gang soll in dieser Arbeit entwickelt werden.

1.3 Aufbau der Arbeit

In Kapitel 2 werden die mathematischen Grundlagen wie Regelungstechnik und Kinematik besprochen und erläutert. Sie dienen im Weiteren als theoretisches Grundgerüst um die in der Arbeit gemachten Entwicklungen zu beschreiben.

Im darauf folgenden Kapitel wird ein Abriss über Entwicklungen bei humanoiden Robotern in Bezug auf ihren Gang gegeben. Dabei werden die Ursprünge und Grundbegriffe in der Laufforschung ebenso erläutert wie Einblicke in die aktuelle Forschung gegeben. Ein besonderes Augenmerk wird auf die beiden Gangtheorien Zero Moment Point und Passive Dynamic Walking gelegt. Auch wird Einblick in die Entwicklung im RoboCup Umfeld gegeben.

In Kapitel 4 wird die FUmanoid-Plattform, auf welcher der Gang entwickelt wird, vorgestellt. Hierbei wird auch auf die Entwicklung eines Drucksensormoduls eingegangen.

Das Kapitel 5 beschreibt die Lösung der inversen Kinematik für die Laufplattform auf zwei unterschiedliche Arten, zum einen eine geometrische, zum anderen auch eine approximative Lösung. Dabei ist insbesondere die parallelmechanische Besonderheit der FUmanoid-Roboter zu beachten.

In Kapitel 6 folgt die eigentliche Entwicklung des Laufsystems. Hier werden sämtliche Algorithmen und mechanische Methoden, die zum Einsatz kommen, beschrieben. In Kapitel 7 werden dann zusätzliche Bewegungen, die aus dem Laufen heraus ausgeführt werden können, vorgestellt und ihre Umsetzung erläutert.

In den letzten beiden Kapiteln werden einige Experimente beschrieben und ausgewertet, sowie ein Ausblick auf mögliche zukünftige Arbeiten gegeben.

2 Grundlagen

2.1 Regelungstechnik

Als Regelungstechnik wird das Teilgebiet der Ingenieurwissenschaften bezeichnet, das mittels Messen und Steuern, dem sogenannten Regeln, versucht, automatisierte Prozesse in gewünschte Zustände zu versetzen. Beispiele hierfür sind die Regelung von pH-Werten in chemischen Prozessen, Druck in Systemen oder Temperatur von Umgebungen. Im Bereich der Robotik treten Regler in vielen Bereichen auf. Der häufigste ist dabei der Positionsregler, der die Position eines Endeffektors oder auch nur eines einzelnen Motors regelt.

Ein *Regler* ist ein System, welches eine *Regelgröße* von einem Ist-Wert auf einen Soll-Wert regelt. Beispielsweise soll die Temperatur in einem Raum stets auf 21℃ geregelt werden. Dazu wird die Regelgröße (Raumtemperatur) gemessen und die *Regelabweichung* $e(t)$ vom Soll-Wert ermittelt. Über ein *Stellglied* (die Heizung) wird nun eine Anpassung der Regelgröße vorgenommen, die sogenannte *Stellgröße* $u(t)$ (Vorlauftemperatur, Ventilöffnung). Nun wird wieder die Regelgröße gemessen. Durch *Störgrößen* (kaltes Wetter, offenes Fenster), aber auch durch Verzögerungen in der Regelung durch indirekte Regelung (die Heizung wird erwärmt und muss ihrerseits den Raum erwärmen) können weiterhin Abweichungen vom Soll-Wert in der *Regelstrecke* vorherrschen und es muss weiter geregelt werden. Es liegt ein *Regelkreis*, auch Closed Loop genannt, vor (Abb. 2.1). Der offene Steuerungskreis, Open Loop, verzichtet dagegen auf die Rückkoppelung von Sensordaten.

Regler können unterschiedlich auf Regelabweichungen reagieren. Die häufigsten Regler sind PID-Regler. Sie bestehen aus einem proportionalen Anteil, einem integrativen Anteil und einem differentialen Anteil. Indem einer oder zwei dieser Anteile weggelassen werden, entstehen die ebenso häufig eingesetzten PI-, PD-, P- und I-Regler. Beschreibungen dieser Regler finden sich in vielen Lehrbüchern zum Thema Regelungstechnik, z.B. [25, 51].

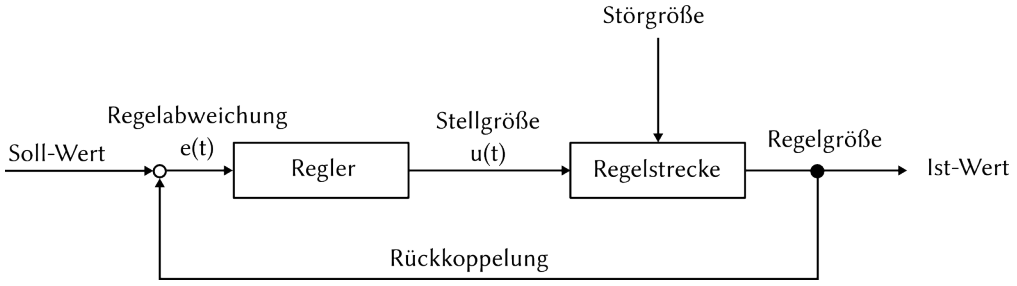


Abb. 2.1: Ein Regelkreis: Mittels Rückführung von Messdaten kann ein Regler auf Störungen in der Regelstrecke reagieren.

2.1.1 P-Anteil

Der einfachste Regler ist ein P-Regler (Proportional-Regler). Er besteht ausschließlich aus einem P-Anteil. Ein solcher Regler vergrößert die Stellgröße proportional zur Regelabweichung. Seine Stellgröße ist also

$$u(t) = K_P e(t).$$

Hierbei ist K_P die Proportionalteilkonstante, die den Verstärkungsfaktor des P-Anteils bestimmt. Mit einem solchen Regler werden Regelabweichungen schnell abgebaut. Er neigt jedoch auch leicht zu Übersteuerungen, da er die Dynamik des Systems ignoriert. So kann z.B. ein Motor nicht abrupt abgestoppt werden und eine Heizung reagiert nicht sofort. P-Regler neigen daher dazu, zu oszillieren, bis sie den Soll-Wert erreichen. Trotzdem kann ein einfacher P-Regler in vielen Fällen schon nützlich sein. Außerdem enthält nahezu jeder Regler einen P-Anteil.

2.1.2 I-Anteil

Ein I-Regler (Integral-Regler) integriert Regelabweichungen über die Zeit auf, verändert so also kontinuierlich die Stellgröße. Die Stellgröße eines I-Reglers ist

$$u(t) = \frac{1}{T_N} \int_0^t e(\tau) d\tau,$$

wobei T_N die Nachstellzeit ist, also die Zeit, bis ein Integral-Regler die Stellgröße so verändert hat, dass der Soll-Wert erreicht ist. Ein Integral-Anteil wird meist dazu eingesetzt Regelfehler auszugleichen. Wenn also beispielsweise ein Stellglied nicht wie gewünscht arbeitet, sondern konstant von der gewünschten Stellgröße abweicht, kann ein I-Anteil helfen diesen konstanten Fehler über die Zeit auszugleichen. Im zeitdiskreten Fall wird hierbei aus dem Integral eine Summe.

2.1.3 D-Anteil

Ein D-Glied (Differential-Glied) in einem Regler reagiert im Gegensatz zu den anderen beiden Anteilen nicht auf die Größe der Regelabweichung, sondern auf deren Änderung. Die Stellgröße ist dabei

$$u(t) = T_D \frac{de(t)}{dt},$$

wobei T_D die Vorhaltzeit ist. Dies bezeichnet die Zeit, die ein P-Regler bei konstanter Änderungsgeschwindigkeit der Regelgröße bräuchte, um die gleiche Änderung zu bewirken, die der D-Anteil eines PD-Reglers ohne Zeitverzögerung bewirkt. D-Anteile werden zur Dämpfung von Reglern eingesetzt. Ist die Änderung der Regelabweichung sehr groß, steuern sie gegen, damit nicht übersteuert wird. Zum anderen sprechen sie früher auf Änderungen an, da sie den Gesamtwert der Regelabweichung ignorieren. Technisch ist ein reines D-Glied nicht realisierbar, da die Änderung der Regelabweichung nicht in einem Punkt gemessen werden kann. Stattdessen wird im zeitdiskreten Fall, wie er meist in der Robotik vorliegt, die Änderung seit der letzten Messung als echte Änderung angenommen. Liegt die Samplefrequenz deutlich über der zu erwartenden Frequenz von Schwankungen in der Regelgröße, ist dies unproblematisch.

2.1.4 PID- und PD-Regler

Als Kombination der oben erwähnten Anteile können nun Regler konstruieren werden. Dabei sind die häufigsten Kombinationen der PID- und der PD-Regler. Sie bestehen aus den jeweiligen im Namen enthaltenen Anteilen. Auch einfache P- und I-Regler sind in der Praxis anzutreffen. Für einen üblichen PID-Regler ergibt sich damit die Differentialfunktion der Stellgröße

$$u(t) = K_P e(t) + \frac{1}{T_N} \int_0^t e(\tau) d\tau + T_D \frac{de(t)}{dt}.$$

Häufig werden die Nachstell- und Vorhaltzeit auch mit Reglerkonstanten ersetzt, dann gilt

$$\begin{aligned} K_I &= \frac{1}{T_N} \\ K_D &= T_D \\ u(t) &= K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt}. \end{aligned}$$

Um diese Konstanten zu wählen, gibt es verschiedene mathematische Methoden, die aufwendig zu berechnen sind. Deswegen werden viele PID Regler manuell eingestellt. Dafür werden Experimente mit der Regelstrecke durchgeführt, bis zufriedenstellende Reglerkonstanten gefunden werden. Dabei gibt es unterschiedliche Anforderungen an einen Regler. Je nach Anwendungen können Übersteuerungen nicht akzeptiert werden oder sie sind für ein schnelles Einschwingen auf den Soll-Wert vertretbar. Auch die Geschwindigkeit, mit der ein Regler auf Regelabweichungen reagiert, ist von Belang. Alle in dieser Arbeit eingesetzten Regler wurden im manuellen Verfahren eingestellt, deshalb wird auf die Beschreibung mathematischer Verfahren verzichtet.

2.2 Kinematik

Kinematik ist die Lehre der Bewegung von Körpern unter Vernachlässigung der Kräfte. Im Kontext der Robotik beschäftigt sich die Kinematik mit der Berechnung der Lage von Teilen des Roboters in Abhängigkeit der Stellung der Gelenke. Dabei gibt es zwei wichtige Fragen. Die erste ist die der Vorwärts-Kinematik oder auch direkten Kinematik und lautet:

> Gegeben alle Gelenkstellungen, wo befindet sich ein gegebener Punkt des Roboters (Endeffektor) im Raum?

Die zweite Frage ist die der Rückwärts- bzw. inversen Kinematik. Sie lautet:

> Gegeben die Position eines Endeffektors, wie müssen die Gelenke gestellt werden, um diese Position zu erreichen?

Von diesen Fragen ist besonders die zweite essentiell zur Gangerzeugung, da durch ihre Antwort Trajektorien im euklidischen Raum in Trajektorien im Gelenkraum umgerechnet werden können. Aber auch die Berechnung der Vorwärts-Kinematik ist wichtig. Beispielsweise wird der Schwerpunkt des Roboters berechnet, um mit ihm Optimierungen der Laufbewegung durchführen zu können.

2.2.1 Homogene Koordinaten

Um Positionen, Translationen und Rotationen darzustellen, werden die sogenannten Homogenen Koordinaten genutzt. Sie erlauben eine kompakte Darstellung und Speicherung komplexer Transformationen in einer 4×4 Matrix [54].

Einzelne Rotationen um eine Koordinatensystem-Achse können als homogene Rotationsmatrix in Abhängigkeit zum Winkel ϑ dargestellt werden. So sind die Rotationen um die X-, Y- und Z-Achse:

$$R_x(\vartheta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\vartheta) & -\sin(\vartheta) & 0 \\ 0 & \sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_y(\vartheta) = \begin{pmatrix} \cos(\vartheta) & 0 & \sin(\vartheta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\vartheta) & 1 & \cos(\vartheta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\vartheta) = \begin{pmatrix} \cos(\vartheta) & -\sin(\vartheta) & 0 & 0 \\ \sin(\vartheta) & \cos(\vartheta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Die Translation um den Vektor $t = \begin{pmatrix} t_x & t_y & t_z \end{pmatrix}^T$ ist:

$$T(t) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Weitere mögliche Transformationen sind Skalierung, perspektivische Verzerrung und orthogonale Projektion. Mit diesen grundlegenden Transformationen können nun komplexere Transformationen erstellt werden. Die Notation als Transformationsmatrizen hat zur Folge, dass nacheinander ausgeführte Transformationen als Matrixmultiplikation aufgefasst werden können. Eine Transformation T, bei der zuerst eine Verschiebung T_1, dann eine Rotation R und dann eine Verschiebung T_2 ausgeführt wird, ist also:

$$T = T_2 \cdot R \cdot T_1$$

Durch diese Nacheinanderausführung von verschiedenen Transformationen können nun beliebig komplexe Transformationen als 4×4 Matrix gespeichert werden.

2.2.2 Direkte Kinematik

In vielen Situationen wird die aktuelle Position eines Endeffektors als Feedback in Regelkreisen benötigt. Die Sensorik der Aktuatoren bietet aber nur die Stellung des Aktuators als Messgröße. Es ist also notwendig aus der Stellung der Aktuatoren die Position eins Endeffektors zu berechnen.

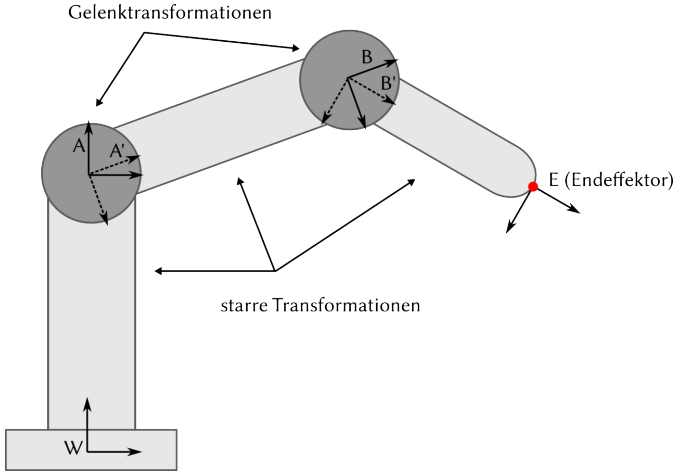


Abb. 2.2: Kinematische Kette in einem einfachen Roboterarm mit zwei Drehgelenken. Zu sehen sind die Koordinatensysteme der einzelnen Körper.

Übliche Aktuatoren sind Drehgelenke oder Translationsgelenke. Sie führen eine Drehung oder eine Verschiebung durch.

Zusätzlich zu den Aktuatoren gibt es noch starre Körper in einem Roboter. Auch diese müssen bei der Berechnung der Endeffektorposition beachtet werden.

Um eine Endeffektorposition bestimmen zu können, wird zuerst ein Ursprungskoordinatensystem W als Referenz gebraucht. Dieses kann frei im Raum gewählt werden, steht jedoch in einem festen Bezug zu einem definierten Punkt im Roboter. Dieser Ursprung wird zum Beispiel im Fuß festgelegt.

Entlang der kinematischen Kette des Roboters, der Aneinanderreihung der starren Körper und Gelenke, werden nun Verschiebungen und Drehungen berechnet. Jedem Körper und Gelenk wird dabei ein zu ihm festes Koordinatensystem zugeordnet. Die Transformationsmatrix einer starren Transformation von einem Körper A zum Körper B ist unabhängig von jeglicher Stellgröße und wird mit T_{AB} betitelt. Sie entspricht dabei einer Verschiebung des Koordinatensystems, bei gekrümmten oder gewinkelten Körpern möglicherweise auch einer Drehung.

Die Transformation in Gelenken ist abhängig von dem Winkel des Drehgelenks oder der Auszugslänge des Translationsgelenks, der Stellgröße q. Die dazugehörige Transformationsmatrix in einem Gelenk B wird $T_{BB'}(q)$ genannt. In einem Rotationsgelenk entspricht sie einer Drehung, bei einem Translationsgelenk einer Verschiebung.

Um die Transformationsmatrix vom Ursprung zu einem Endeffektor E zu berechnen, müssen nun alle Transformationsmatrizen der kinematischen Kette multipliziert werden. Zu dem in Abbildung 2.2 dargestellten Roboterarm ist diese Berechnung:

$$T_{WE}(q_1, q_2) = T_{WA} \cdot T_{AA'}(q_1) \cdot T_{A'B} \cdot T_{BB'}(q_2) \cdot T_{B'E}$$

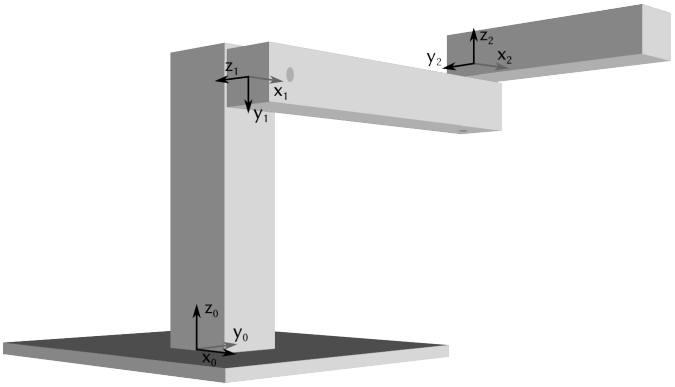


Abb. 2.3: Die Koordinatensysteme nach der Denavit-Hartenberg-Konvention

Denavit-Hartenberg-Konvention

Da die Koordinatensysteme beliebig an den Körpern positioniert werden können, sind die Transformationsmatrizen nicht eindeutig. Zur Übersichtlichkeit und zur einfacheren Konstruktion ist es aber hilfreich einen Standard zu definieren. Einen solchen Standard schlugen Jaques Denavit und Richard S. Hartenberg 1955 vor [11]. Er wird nach ihnen Denavit-Hartenberg-Konvention genannt.

Die Denavit-Hartenberg-Konvention (DH-Konvention) beschreibt jede Transformation zwischen zwei Körpern als Kombination von vier Transformationen, die jeweils von einem Parameter bestimmt werden. Um diese minimale Darstellung zu erreichen, wird folgednes vorrausgesetzt:

Entlang einer kinematischen Kette mit $n + 1$ Körpern und n Gelenken gilt:

1. z_i, die z-Achse des i-ten Koordinatensystems, ist die Drehachse des i-ten Gelenks.
2. x_i ist parallel zu $z_{i-1} \times z_i$.
3. y_i ergibt mit x_i und z_i zusammen ein rechtshändiges Koordinatensystem.

Sind diese Vorraussetzungen erfüllt (siehe Abb. 2.3), kann eine Transformation von einem in das nächste Koordinatensystem mit den folgenden vier Transformationen durchgeführt werden:

1. eine Rotation um die Gelenkachse z_{i-1} mit einem Winkel ϑ_i als Parameter (Gelenkwinkel)

$$R_{z_{i-1}}(\vartheta_i) = \begin{pmatrix} \cos(\vartheta_i) & -\sin(\vartheta_i) & 0 & 0 \\ \sin(\vartheta_i) & \cos(\vartheta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2. eine Verschiebung entlang der Gelenkachse z_{i-1} mit der Distanz zum nächsten Gelenk d_i als Parameter

$$T(0, 0, d_i) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. eine Verschiebung entlang der x_i-Achse, orthogonal zu den beiden Drehachsen z_{i-1} und z_i mit dem Abstand r_i zum nächsten Gelenk

$$T(r_i, 0, 0) = \begin{pmatrix} 1 & 0 & 0 & r_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4. eine Rotation um die x_i-Achse um den Winkel α_i, so dass die Lage des nächsten Gelenks erreicht wird

$$R_{x_i}(\alpha_i) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Aus diesen vier Transformationen ergibt sich die Gesamttransformation

$$\begin{aligned} T_i &= R_{z_{i.1}}(\vartheta_i) \cdot T(0, 0, d_i) \cdot T(r_i, 0, 0) \cdot R_{x_i}(\alpha_i) \\ &= \begin{pmatrix} \cos(\vartheta_i) & -\sin(\vartheta_i) \cdot \cos(\alpha_i) & \sin(\vartheta_i) \cdot \sin(\alpha_i) & r_i \cdot \cos(\vartheta_i) \\ \sin(\vartheta_i) & \cos(\vartheta_i) \cdot \cos(\alpha_i) & -\cos(\vartheta_i) \cdot \sin(\alpha_i) & r_i \cdot \sin(\vartheta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}. \end{aligned}$$

Es ergeben sich pro Transformation die vier Denavit-Hartenberg-Parameter ϑ_i, d_i, r_i und α_i. Dabei sind ϑ_i bzw. d_i bei aktuierten Gelenken abhängig von der Gelenkstellung, r_i und α_i sind stets fix. Eine kinematische Kette kann als Tabelle dieser vier Parameter beschrieben werden. Aus diesen Parametern lassen sich die Transformationsmatrizen erstellen und die direkte Kinematik lösen.

2.2.3 Inverse Kinematik

Während die direkte Kinematik für jede kinematische Kette analytisch gelöst werden kann, ist das inverse Problem nicht allgemein lösbar. Es ist leicht zu erkennen, dass mehrere oder auch keine Lösung existieren können (Abb. 2.4). Bei der Berechnung muss eine Lösung gewählt werden.

Für bestimmte Konfigurationen von kinematischen Ketten ist es möglich, die inverse Kinematik analytisch zu lösen. So hat Pieper in [37] gezeigt, dass Ketten von sechs Drehgelenken, von denen sich drei aufeinanderfolgende Drehachsen in einem Punkt schneiden, immer lösbar sind. Industriell genutzte Roboter nutzen oft eine solche Konfiguration, um die inverse Kinematik leicht lösen zu können.

Bei einigen Robotern ist es auch möglich, eine geometrische Lösung der inversen Kinematik zu finden. Ein solcher Ansatz hat den Vorteil, dass die Berechnung sehr effizient erfolgen kann. Nachteilig ist, dass ein geometrischer Ansatz abhängig vom eingesetzten Roboter ist [54]. Auch in einem Roboter kann jeweils nur ein Endeffektor gleichzeitig auf diese Weise modelliert werden.

Allgemeiner sind numerische Verfahren, die Lösungen approximieren und für alle Roboter eingesetzt werden können. Ihr Nachteil ist, dass sie oft sehr rechenaufwändig sind und nicht zur

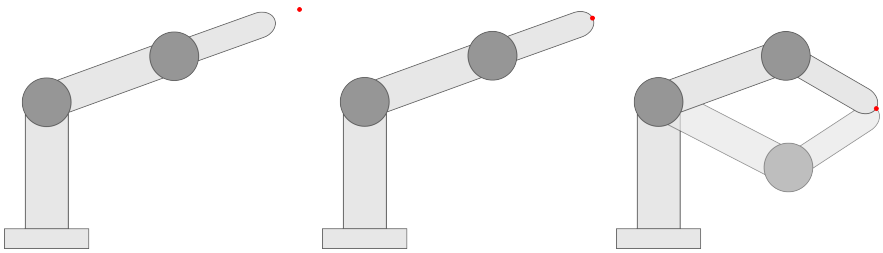


Abb. 2.4: Bei der inversen Kinematik können keine, eine oder mehrere Lösungen möglich sein. Der rote Punkt ist das zu erreichende Ziel.

Laufzeit des Roboters berechnet werden können. Zur Vorberechnung und Optimierung von Bewegungen sind sie aber sehr hilfreich. Ein häufig angewandtes Verfahren nutzt hierzu die Jacobi-Matrix [54].

Sei $\phi(q)$ die kinematische Abbildung von den Gelenkstellungen q in den euklidischen Raum, so ist die Jakobi-Matrix J definiert als

$$J(q) = \frac{\partial}{\partial q}\phi(q).$$

Dies ist die Ableitung der kinematischen Abbildung nach allen Gelenkstellungen. Vorrausgesetzt, die kinematische Kette besteht ausschließlich aus Drehgelenken, so ist die Ableitung nach einem Gelenkwinkel genau senkrecht zur Drehachse und der Strecke von der Drehachse zum Endeffektor (Abb. 2.5), da der Endeffektor sich auf einer Kreisbahn um das Gelenk bewegt [35]. Es gilt, wie dort gezeigt:

$$J(q) = \begin{pmatrix} [z_1 \times (p_{eff} - p_1)] & [z_2 \times (p_{eff} - p_2)] & \cdots & [z_n \times (p_{eff} - p_n)] \end{pmatrix} q.$$

Die Bewegung der Endeffektoren kann nun durch

$$p'_{eff} = J(q) \cdot \dot{q}$$

leicht linear approximiert werden, wobei $\dot{q}$ die Gelenkwinkel zeitlich differenziert sind und p'_{eff} die Position des Endeffektors nach einem Zeitschritt ist. Um eine bestimmte Position anfahren zu können, wird die dafür notwendige Änderung der Gelenkwinkel benötigt. Diese entspricht $\dot{q}$. Um den Endeffektor also zur Position p_g zu bewegen, berechnet man

$$\dot{q} = J(q)^{-1} \cdot p_g.$$

Da J nicht unbedingt invertierbar ist, kann auch die Pseudoinverse genutzt werden. Das Invertieren von Matrizen ist im Allgemeinen aber rechenaufwändig, weshalb dieses Verfahren nur

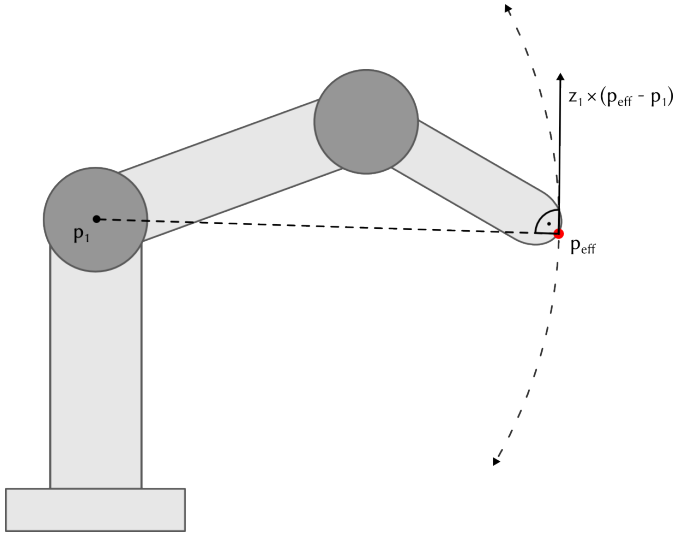


Abb. 2.5: Die Jakobi-Matrix bietet eine lineare Approximation der inversen Kinematik. Die gestrichelte Linie zeigt die echte Bewegung des Arms um das erste Gelenk, der durchgezogene Pfeil ist die lineare Approximation.

auf leistungsfähigen Rechnern in Echtzeit ausgeführt werden kann. Auf Robotern mit geringer Rechenleistung kann eine solche inverse Kinematik nicht zur Laufzeit berechnet werden. Diese Approximation ist außerdem nicht für größere Positionsänderungen geeignet. Die Annahme der Linearität führt schon früh zu großen Abweichungen gegenüber den Kreisbögen der Gelenke. Für kleine, iterativ angewandte Positionsänderungen reicht die Genauigkeit aber im Allgemeinen aus.

Neben der Position eines Endeffektors lässt sich mit Hilfe der Jakobi-Matrix auch seine Ausrichtung vorgeben. Dazu muss die Ableitung der Ausrichtung betrachtet werden. Sie ist orthogonal zur Drehachse und zur alten Ausrichtung. Die Jakobi-Matrix lautet also

$$J(q) = \begin{pmatrix} [p_{eff} \times p_1] & [p_{eff} \times p_2] & \cdots & [p_{eff} \times p_n] \end{pmatrix} q.$$

2.2.4 Kinetik

In vielen Anwendungen wird nicht nur die Position oder die Ausrichtung eines Endeffektors benötigt, sondern auch der Masseschwerpunkt des Roboters. Insbesondere bei autonomen, mobilen Robotern, die ihr Gleichgewicht halten müssen, ist die Position des Schwerpunkts ein wichtiges Kriterium, nach dem geregelt werden kann. Der Schwerpunkt stellt jedoch keinen normalen Endeffektor dar, da er nicht in festem Bezug zu einem speziellen Punkt im Körper des Roboters steht. Er kann sich in ihm bewegen oder sogar außerhalb des Körpers liegen.

Die Position des Gesamtschwerpunkts lässt sich leicht aus den Schwerpunkten der einzelnen Körper berechnen. Diese können vorberechnet werden und ändern sich nicht, da die Körper sich nicht ändern. Der Gesamtschwerpunkt G eines Körpers mit der Masse m ist zum Ursprung O_1:

$$O_1G = \frac{1}{m} \sum_{i=1}^{n} O_1G_i \cdot m_i.$$

Wobei G_i die Position des Schwerpunkts des i-ten Köpers ist und m_i seine Masse. G_i lässt sich mit Hilfe der direkten Kinematik bestimmen. Die direkte Kinetik ist also leicht zu berechnen. Da das i-te Gelenk alle Körper ab dem i-ten und damit auch ihre gemeinsamen Schwerpunkte bewegt, wird in [5] der augmentierte Schwerpunkt eingeführt. Dieser ist der Schwerpunkt des Teilkörpers, der von dem Gelenk j getragen wird, und berechnet sich analog zum Gesamtschwerpunkt im Ursprung des j-ten Körpers als

$$O_jG_{aj} = \frac{1}{m_{aj}} \sum_{i=j}^{n} O_jG_i \cdot m_i.$$

Mit diesen augmentierten Schwerpunkten kann die Jakobi-Matrix für den Gesamtschwerpunkt berechnet werden [5]:

$$v_i = \sum_{j=i}^{n} \left[\frac{m_{aj}}{m} \left(z_j \times O_jG_{aj} \right) \right]$$
$$J_G(q) = \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix} q.$$

Diese ist nach den Schwerpunkten gewichtet und kann zur linearen Approximation der inversen Kinetik herangezogen werden.

3 Verwandte Arbeiten

3.1 Grundbegriffe

Ein zweibeiniger Gang ist eine sich rhythmisch wiederholende Bewegung, die aus unterschiedlichen Phasen besteht. Als Erstes kann unterschieden werden, ob sich gerade ein oder zwei Füße auf dem Boden befinden. Diese Phasen heißen Single-Support-Phase, wenn ein Fuß den Boden berührt, beziehungsweise Double-Support-Phase, wenn beide Füße den Boden berühren (siehe Abb. 3.1). Weiterhin wird pro Fuß die Schwung- und die Standphase unterschieden. Befindet sich der Fuß gerade in der Luft, ist er in der Schwungphase, befindet er sich auf dem Boden, so ist er in der Standphase.

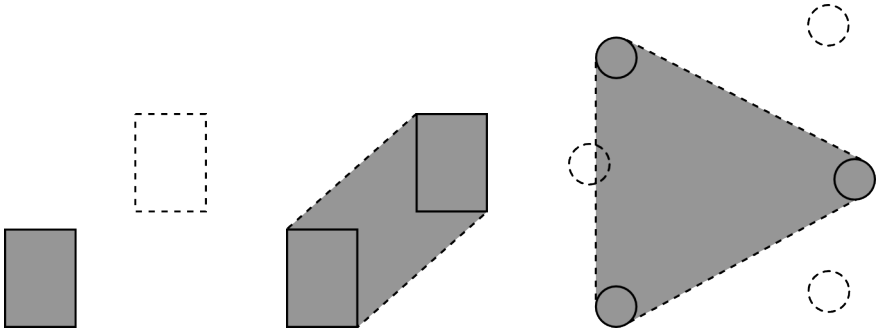
Die einfachste Art, einen Gang zu stabilisieren, ist die statische Stabilität. Ein Gang wird statisch stabil genannt, wenn der Roboter in jeder Phase angehalten werden kann, ohne umzufallen. Um einen solchen Gang zu realisieren, muss der Masseschwerpunkt des Roboters stets über dem Supportpolygon gehalten werden. Das Supportpolygon ist die konvexe Hülle um alle Verbindungspunkte des Roboters mit dem Untergrund (siehe Abb. 3.1). Solche Gangarten sind für zweibeinige Roboter sehr unnatürlich und für hohes Tempo schwer umzusetzen, da bis zum Aufsetzen des Fußes in der Schwungphase der Schwerpunkt des Roboters über dem Standfuß gehalten werden muss. Für Roboter mit vier oder mehr Füßen ist dieser Gang aber ein häufig und erfolgreich eingesetzter Gang [22].

Von dynamischer Stabilität hingegen wird immer gesprochen, wenn der Roboter während des Gangs nicht hinfällt. Es ist jedoch nicht garantiert, dass er das Gleichgewicht bei einem abrupten Ende des Bewegungsablaufes hält. Es können verschiedene Kriterien genutzt werden, um eine dynamische Stabilität zu gewährleisten. Das bekannteste von ihnen ist der Zero Moment Point (ZMP).

3.2 Zero Moment Point

Der Zero Moment Point (ZMP) wurde bereits in den frühen 1970ern von Vokobratović als Stabilitätskriterium vorgeschlagen [20]. Schon der erste humanoide Roboter mit einem dynamisch blancierten Gang, WABOT-1, nutzte den ZMP [57]. Auch heute noch verwenden ihn viele bekannte humanoide Roboter zur Balanceregelung, wie beispielsweise Asimo [18] und die HRP-Reihe [21].

Der Fuß eines Roboters kann als unaktuiertes Gelenk zwischen Untergrund und Roboter gesehen werden. Während des Gangs soll immer der gesamte Fuß in Verbindung mit dem Untergrund sein und nicht nur eine Kante. Sobald nur noch eine Kante in Verbindung mit dem Untergrund ist, beginnt der Roboter über die Kante zu fallen und kann, da sie nicht aktuiert ist, keine Kraft aufbringen, um diesem Fall entgegen zu wirken. Mittels des Zero Moment Points wird versucht eine solche Situation zu verhindern. Die Beschreibung folgt im wesentlichen [53].



(a) Single-Support-Phase

(b) Double-Support-Phase

(c) Multi-Support-Phase eines Hexapods

Abb. 3.1: Das Supportpolygon ist die konvexe Hülle um alle Verbindungspunkte des Roboters mit dem Grund, hier grau markiert. Füße, die in der Luft sind, sind gestrichelt dargestellt.

Zur Analyse der Dynamik wird der Einfluss auf den Roboter als Kraft F_A und Drehmoment M_A vereinfacht (Abb. 3.2). Die Gegenkraft besteht dann aus der Kraft R und dem Drehmoment M, welche im Punkt P wirken. Die horizontalen Komponenten der Kraft R, R_x und R_y, und die vertikale Komponente M_z des Drehmoments M repräsentieren die Reibungskräfte, welche die horizontalen Komponenten von F_A und die vertikale Komponente von M_A kompensieren. Dafür wird die Annahme getroffen, dass die Verbindung zwischen Fuß und Untergrund genügend Reibung aufweist, so dass der Roboter nicht rutscht. Die Kraft R_z repräsentiert die Kraft, die vertikale Kräfte balanciert.

Es müssen also noch die horizontalen Drehmomente M_x und M_y balanciert werden. Liegt P, also der Punkt, in dem die Gegenkraft R wirkt, innerhalb des Supportpolygons, so gibt es an diesem Punkt keine induzierten Drehmomente M_x und M_y. Sie müssen also auch nicht kompensiert werden. Wenn das Supportpolygon allerdings zu klein ist, wirkt die Gegenkraft R an einer Kante des Fußes und induziert Drehmomente, so dass der Roboter über diese Kante fällt. Liegt P innerhalb des Supportpolygons, heißt er Zero Moment Point (Null-Drehmoment-Punkt), da an ihm keine Drehmomente wirken. Dies macht auch deutlich, dass der Zero Moment Point nur innerhalb des Supportpolygons existieren kann. Läge er außerhalb, würde die Kraft R so wirken, dass Drehmomente entstünden.

Wenn als Stabilitätskriterium angenommen wird, dass der Roboter immer mit dem gesamten Fuß auf dem Boden stehen soll, so muss der ZMP stets innerhalb des Supportpolygons des Roboters existieren.

Mittels eines genauen Modells der Bewegung des Roboters könnte der Zero Moment Point berechnet werden. Dies ist aber sehr rechenaufwendig und erfodert außerdem eine sehr genaue Messung der echten Bewegung. In [45] wurde jedoch gezeigt, dass der Druckmittelpunkt gleich dem Zero Moment Point ist, wenn dieser existiert. Es genügt also, mittels Drucksensoren den Druckmittelpunkt und darüber den ZMP zu bestimmen. Mittels Regelung kann dann versucht werden, den Roboter so zu bewegen, dass der ZMP stets existiert.

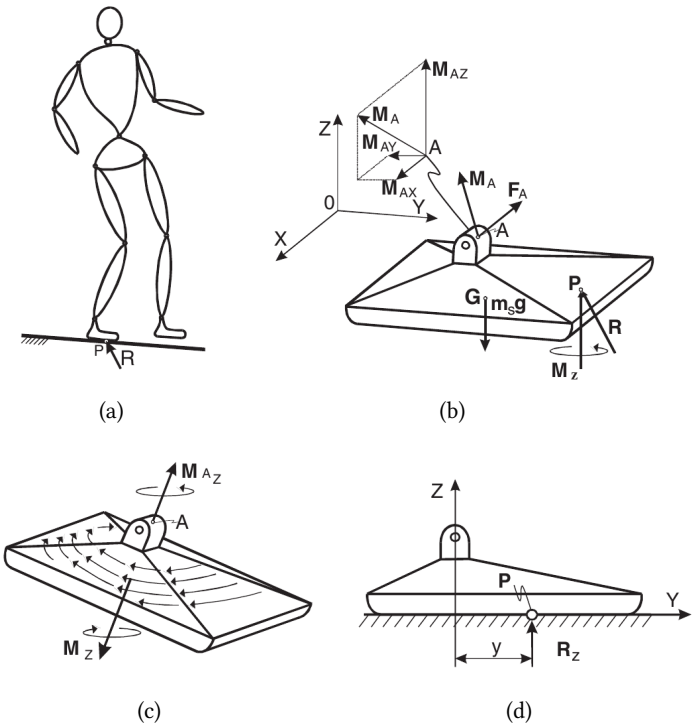


Abb. 3.2: Das mechanische Modell eines Roboters und die Kräfte, die auf ihn wirken. Abbildung aus [53].

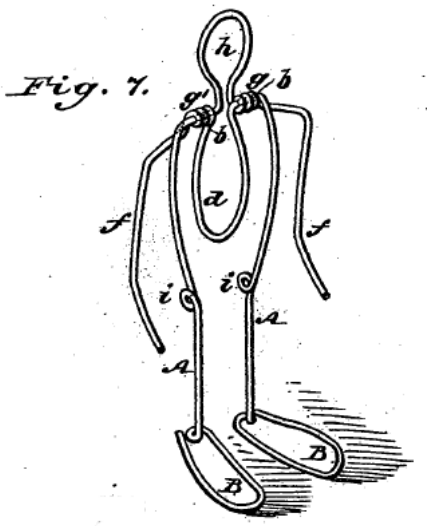


Abb. 3.3: Einfache Lauf-Spielzeuge sind schon lange bekannt. Das Konzept des Passive Dynamic Walkers basiert auf dem selben Prinzip. Abbildung aus [12].

3.3 Passive Dynamic Walker

Ein gänzlich anderer Ansatz zur Analyse von zweibeinigen Gängen ist das Konzept des Passive Dynamic Walkers (passiver dynamischer Gang). Es basiert auf der Beobachtung alter Spielzeuge, die ohne jeden Motor in der Lage sind Schrägen herabzugehen (Abb. 3.3). Schon im 19. Jahrhundert patentierte Fallis solche Spielzeuge [12].

Eine ausführliche Untersuchung des Prinzips des Passive Dynamic Walkers fand in den späten Achtzigern und frühen Neunzigern des 20. Jahrhunderts durch Tad McGeer statt [28–30]. Er beschreibt die mathematischen Modelle für eine solche Laufmaschine mit und ohne Knie. Weitergeführt wurde diese Modellierung mit dem Simplest Walker Modell von Garcia et al. [13]. Hier wird versucht, das Modell so weit wie möglich zu vereinfachen, um dann die Dynamik mathematisch beschreiben zu können.

Die meisten Roboter, die auf den Prinzipien des Passive Dynamic Walking basieren, verzichten ganz oder zum Großteil auf Motoren. Diese Maschinen gehen durch ihre systeminhärente Stabilität und die Gravitation als Kraft. Als Modell für einen solchen Roboter kann, wie in [28] beschrieben, ein sogenanntes synthetisches Rad dienen. Durch seine runde Form ist ein Rad in der Lage, eine Schräge ohne weiteren Einfluss herunter zu rollen. Auch wenn die Felge des Rades entfernt wird, rollt das Rad allein auf den Speichen, vorausgesetzt der Winkel zwischen den Speichen ist klein genug und die Steigung groß genug. Jede dieser Speichen kann auch als Bein einer Laufmaschine interpretiert werden. Wenn sich die Speichen frei um die Nabe des Rades drehen können, so reichen zwei Speichen aus, um das kontinuierliche Rad zu simulieren. Dafür wird zunächst ignoriert, dass die schwingende Speiche bei ihrem Weg nach vorne eine Bodenkollision haben würde. Werden wieder Teilausschnitte der Felge an die Speichen angefügt, simuliert dieses

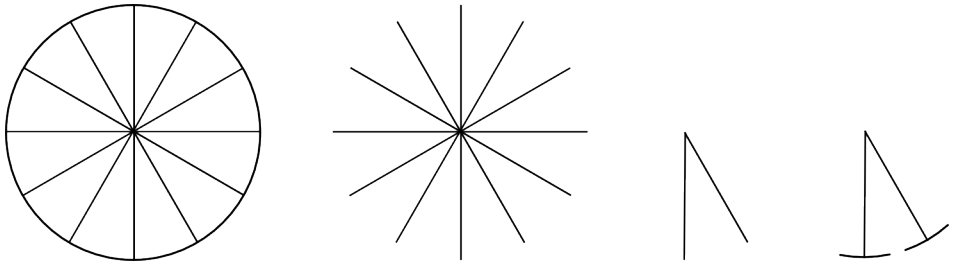


Abb. 3.4: Von einem Rad zum Zweibeiner. Wird ein Großteil der Felge und alle bis auf zwei Speichen, die sich frei um die Nabe drehen können, entfernt, entsteht aus einem Rad ein *synthetisches Rad*, sehr ähnlich einem Zweibeiner.

mit zwei Speichen und je einem Felgenteil ausgestattete Gerät ein Rad. Es ist ein synthetisches Rad (Abb. 3.4).

Diese synthetischen Räder dienen als Modell für eine zweibeinige Laufmaschine. Ein Problem bei der Realisierung ist allerdings die Vernachlässigung der Bodenkollision. Bei einem echten Roboter treten solche Kollisionen unvermeidlich auf. Viele Modelle, so auch Fallis Spielzeug, haben dafür schräg angebrachte Füße, die zu einem oszillierenden Schwingen von links nach rechts führen. Dadurch können die Beine trotz gleicher Länge frei schwingen. McGeers erstes Modell erhielt hingegen Translationsgelenke, um die Beine während der Schwungphase aktiv zu verkürzen [30]. Kurz darauf führte er aber Kniegelenke ein, die erstaunlicherweise ebenfalls ohne Aktuatoren auskommen und nur einen mechanischen Stopp erfordern, so dass sie nicht überstreckt werden können [29].

Tatsächlich können solche Roboter ohne zusätzliche Energie rein durch die Gravitationskräfte Schrägen hinablaufen, so wie ein Rad sie herunter rollen würde, und sind stabil. Auf Ebenen oder bergauf sind zusätzliche Aktuatoren erforderlich. Aber auch hier ist die Art des Gehens sehr energieeffizient, wie bei einem Experiment im Mai 2011 festgestellt werden konnte: Der Roboter *Cornell Ranger* lief eine Strecke von 65 km in 30 Stunden mit einer Akkuladung von 493 Wh, ohne einmal hinzufallen. Damit erreichte er Transportkosten (Cost of Transport, angegeben in Energie pro Last und zurückgelegter Distanz) von $0.28\frac{Wh}{kg \cdot km}$, was nur wenig teurer ist als die durchschnittlichen Transportkosten eines Menschen ($0.2\frac{Wh}{kg \cdot km}$). Transportkosten anderer Roboter, beispielsweise von ASIMO mit $2\frac{Wh}{kg \cdot km}$, sind deutlich höher [42].

Ein großer Vorteil von solchen Laufmaschinen ist, dass keinerlei Trajektorien generiert oder optimiert werden müssen. Diese entstehen stattdessen aus den mechanischen Eigenschaften des Roboters wie Länge der Beine, Kniestopp und Gewichtsverteilung und der Steigung, die ein solcher Roboter herabgeht. Diese Steigung ist proportional zur Energie, die der Roboter verbraucht. Sie kann leicht durch Aktuierung ersetzt werden.

3.4 Central Pattern Generator

Von Organismen inspiriert ist das Konzept des Central Pattern Generators (CPG, auch Zentraler Mustergenerator). In der Natur ist ein CPG ein Teil des zentralen Nervensystems, der ohne Einga-

be und Feedback in der Lage ist, rhythmische Muster zu generieren [15]. Solche CPGs wurden in vielen Tieren nachgewiesen [10]. Die Ausgaben der CPGs werden genutzt, um sich wiederholende Bewegungen zu steuern. Darunter fallen beispielsweise Schwimmen, Atmung, Herzschlag und auch die Bewegung der Beine beim Gehen und Laufen [15]. CPGs arbeiten unabhängig von der bewussten Steuerung. Die Muster, die sie generieren, sind vorprogrammiert und werden abgespielt. Dadurch können rhythmische Tätigkeiten wie Laufen, Atmen etc. ausgeführt werden, ohne diese aktiv zu steuern. Trotzdem müssen diese Muster durch sensorisches Feedback angepasst werden, damit auf Unregelmäßigkeiten und Störungen reagiert werden kann [1].

Seit einiger Zeit werden CPGs auch vermehrt für die Fortbewegung von Robotern eingesetzt. Dabei geht der Einsatz von Schwimmen und Kriechen [1, 8] bis zu zweibeiniger Fortbewegung [38, 52]. Mit unterschiedlichen Arten von Oszillatoren werden dabei CPGs nachgebildet. Die Ausgaben dieser künstlichen CPGs werden dann direkt an die Motoren gegeben. Da die Parameterfindung solcher künstlicher Neurone schwer ist, werden häufig adaptive Systeme eingesetzt, die offline oder online die Parameter der CPGs lernen [8, 38]. Dabei dienen CPGs nicht der Trajektorienfindung eines Endeffektors, sondern als Ersatz für klassische Regler wie PD- oder PID-Regler. Sie finden also Trajektorien für einzelene Aktuatoren.

Experimente mit CPG gesteuerten Laufmaschinen, die Sensorfeedback nutzen, um Trajektorien anzupassen, legen nahe, dass ein solcher Ansatz robuster gegen Störungen und veränderte Untergründe ist als klassische Ansätze der Regelungstechnik [27, 52]. Insgesamt werden CPGs jedoch zur Zeit hauptsächlich in Simulationen genutzt, tatsächliche Implementierungen für einen Gang sind selten.

3.5 Laufen im RoboCup

Da Laufen im RoboCup-Umfeld ein wichtiges Thema ist, sind auch hier viele Forschungsarbeiten veröffentlicht worden. Die meisten Laufsysteme basieren auf einem der oben vorgestellten Ansätze.

Begonnen wurde die Laufforschung im RoboCup mit den Aibo-Robotern, die in der Four-Legged League (Vierbeinige Liga) eingesetzt wurden. Dadurch, dass der Roboter vierbeinig ist, ist die Stabilität deutlich einfacher zu erreichen, als bei zweibeinigen Robotern. Es wurde daher viel Forschungsaufwand in Geschwindigkeit gesetzt. Dabei setzten sich vor allem evolutionäre Lernmethoden durch [6, 19, 43]. Aber auch Reinforcement Learning wurde eingesetzt, um das Laufen der Roboter zu verbessern [24, 44].

Nachdem in der SPL auf den Nao Roboter umgestellt wurde, fällt der Fokus immer mehr auch auf den zweibeinigen Gang. Dabei werden vor allem ZMP-basierte Methoden eingesetzt. Die amtierenden Weltmeister und der Vizeweltmeister, B-Human aus Bremen und die Nao Devils aus Dortmund, setzen beide auf diese Art der Trajektoriengenerierung [9, 14].

In der KidSize und TeenSize Liga gibt es ebenfalls viele Lösungen zur Gangerzeugung. Das Team NimbRo der Universität Bonn nutzt einen CPG-Ansatz, wobei die Muster jedoch nicht direkt Motoren ansteuern, sondern verschiedene Eigenschaften wie zum Beispiel die Länge des Beins oder den Winkel der Hüfte [3, 46]. Damit sind die NimbRo KidSize Roboter in der Lage, 25 cm/s zurück zu legen. Ein weiteres Forschungsgebiet der Bonner ist die Schrittplanung. Mit Hilfe eines Neuronalen Netzes können sie für unterschiedliche Stellungen zu einem Zielpunkt Schritte im

voraus planen. Dafür haben sie eine Datenbank von verschiedenen Positionen aufgebaut, für die optimale Wege bekannt sind. Das Neuronale Netz liefert dann die Interpolation zwischen den bekannten Punkten [47].

Die schnellsten humanoiden Roboter im RoboCup stammen aus Darmstadt [16]. Das Team der Darmstadt Dribblers nutzt Roboter des Typs DD2008. Diese basieren auf dem Robotertyp HR18, welche vom Hajime Research Institute in Japan gebaut werden. Ihr grundlegender Gang basiert auf mittels ZMP-optimierten Trajektorien, die zusätzlich mit Gyroskopdaten stabilisiert werden [49]. Außerdem werden statistische Optimierungsmethoden angewandt, um die Stabilität zu erhöhen [16].

4 Die Plattform

4.1 Mechanik und Kinematik

Die Roboter des Teams FUmanoids sind 60 cm große, humanoide Roboter. Sie haben 21 Freiheitsgrade, davon 7 in jedem Bein, 3 in jedem Arm und einen Freiheitsgrad, um den Kopf nach links und rechts zu drehen. Mit einem Gewicht von 4.8 kg sind sie relativ schwer. Ihr Schwerpunkt liegt bei 35 cm und damit verhältnismäßig weit oben. In Abbildung 4.1 ist ein Roboter des Teams FUmanoids als Konstruktionszeichnung dargestellt.

Eine Besonderheit stellt die Parallelogramm-Mechanik[1] dar. Die Freiheitsgrade in den Knien ändern den Winkel eines Parallelogramms. Dadurch bleiben die oberen und unteren Ebenen stets parallel (Abb. 4.2). Werden nur diese Parallelogramm-Motoren genutzt, bleiben die Fußflächen immer parallel zum Boden und insbesondere auch beide Füße parallel zueinander. Besonders für die wichtige Auftrittsphase beim Gang ist dies von Vorteil. Wie in Kapitel 3 näher beschrieben wurde, ist es wichtig, stets den gesamten Fuß auf dem Boden zu behalten. Während der Auftrittsphase ist dies sehr schwierig. Regelungenauigkeiten führen zu schräg aufkommenden Füßen und damit zu Instabilitäten. Da der schwingende Fuß aber mechanisch parallel zum stehenden Fuß – und damit zum Boden – gehalten wird, sind Regelungenauigkeiten nicht von Bedeutung. Die Auftrittsphase kann also immer im höchsten Maße parallel zum Boden erfolgen.

Aktuiert wird der Roboter durch Motoren der Dynamixel-Reihe der Firma Robotis. Bei dieser Reihe handelt es sich um integrierte Servomotoren, also Motoren, die bereits mit einer Regelung und einem Getriebe versehen sind. Robotis vertreibt Servomotoren unterschiedlicher Größe und Stärke. Die eingesetzten Motoren können über einen seriellen RS-485 Bus angesprochen werden. Sie haben einen ansteuerbaren Bereich von 300° bei einer ansteuerbaren Auflösung von 1024. Die Motoren haben werkseitig ein Potentiometer als Positionssensor. In einem Teil der Motoren wurde dieses durch ein verschleißfreies Magnetsensorsystem ausgetauscht. Diese Modifikiation ist für den Nutzer vollkommen transparent. In den Beinen der FUmanoid-Roboter werden Dynamixel RX-64-Motoren eingesetzt, in den Armen sowie für den Kopf die etwas schwächeren Dynamixel RX-28. In Tabelle 4.1 sind die technischen Daten der Motoren aufgeführt.

Die Motoren sind in einer seriellen kinematischen Kette angeordnet. Für die Drehachsen wird die Namenskonvention aus der Raum- und Luftfahrt übernommen. Der Winkel bei Drehungen um die X-Achse heißt Roll-Winkel, der um die Y-Achse Pitch-Winkel und der um die Z-Achse Yaw-Winkel. Wobei die X-Achse nach vorne, die Y-Achse nach links und die Z-Achse nach oben zeigt.

[1] Der für diese Art von Mechanik häufig benutzte Begriff *Parallelkinematik* ist irreführend. Er impliziert eine kinematische Kette mit parallel agierenden Aktuatoren und wird auch in diesem Zusammenhang genutzt [55]. Die Kinematik der FUmanoid Roboter ist aber weiterhin seriell, nur werden Parallelogramme geschert und damit Ebenen parallel gehalten.

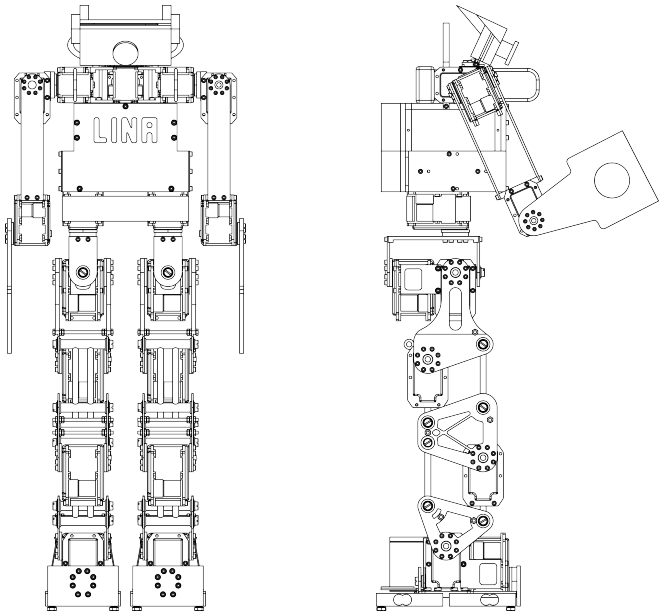
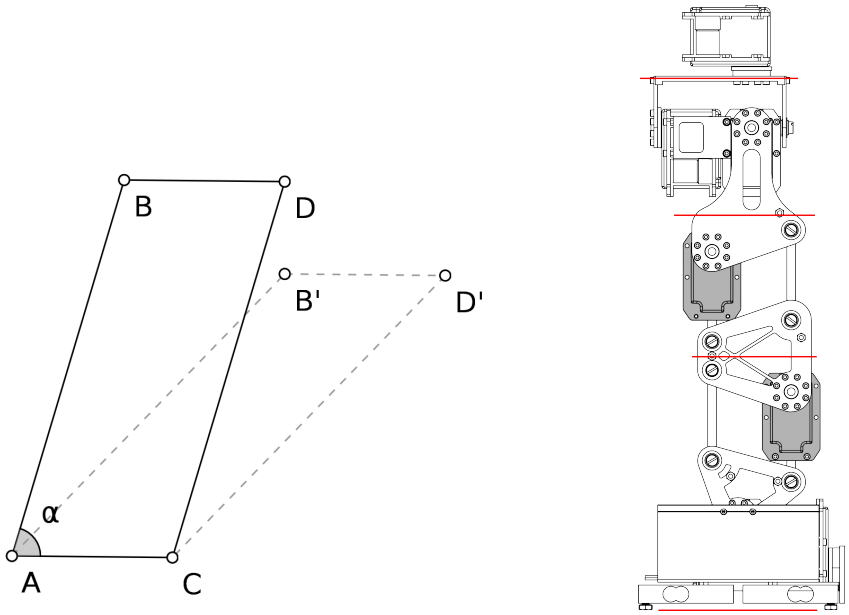


Abb. 4.1: Die Roboter des Teams FUmanoid am Beispiel Lina, technische Zeichnung von Jan Streckenbach und Moritz Fröhlich.

Tab. 4.1: Technische Daten der Dynamixel Servomotoren [40, 41]

Motortyp	**Geschwindigkeit**	**Drehmoment**	**Gewicht**	**Übersetzung**	**Spannung**
RX-64	5.5 − 6.6 $\frac{rad}{s}$	5.0 − 7.5 Nm	125 g	1:200	12V − 18.5V
RX-28	6.2 − 8.3 $\frac{rad}{s}$	2.8 − 3.7 Nm	72 g	1:193	12V − 18.5V



(a) (b)

Abb. 4.2: Wird in einem Parallelogramm ein Winkel verändert, bleiben die gegenüberliegenden Seiten parallel. Dies wird sich im Bein der FUmanoid-Roboter zu Nutze gemacht. Werden nur die grau markierten Motoren zur Veränderung des Pitchwinkels benutzt, bleiben die roten Ebenen stets parallel. Technische Zeichnung von Jan Streckenbach und Moritz Fröhlich.

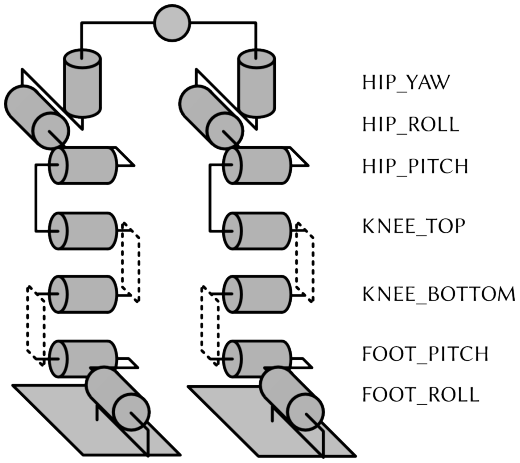


Abb. 4.3: Schematische Darstellung der Beine des Roboters als kinematische Kette. Die gestrichelten Parallelogramme symbolisieren die Parallelogramm-Mechanik. Die Motoren sind als Zylinder entsprechend ihrer Drehachse dargestellt.

Die einzigen Yaw-Achsen des Roboters befinden sich in den Hüften und sind die erste Achse der kinematischen Kette des Beines. Danach folgt eine Roll-Achse und eine Pitch-Achse. Diese drei Motoren bilden zusammen die Hüfte des Roboters. Im Bein sind zwei weitere Pitch-Achsen, die jeweils eine Parallelogramm-Mechanik betätigen. Im Fuß befindet sich eine weitere Pitch-Achse und eine Roll-Achse (Abb. 4.1).

Die Arme besitzen zwei Schultergelenke, eines für die Pitch-, eines für die Roll-Bewegung, und ein Ellbogengelenk mit Pitch-Achse. Im Körper sind das Hauptmodul und ein Mikroprozessorboard zur Ansteuerung der Motoren untergebracht. Sie werden in Abschnitt 4.3 näher beschrieben werden.

4.2 Sensorik

Die Roboter besitzen neben den Positionssensoren der Motoren drei weitere Arten von Sensoren. Der wichtigste Sensor ist eine USB Kamera der Firma Logitech. Sie ist im Kopf untergebracht und versorgt das Vision-Modul der Software mit Bildinformationen. Eine 180° Fischaugenlinse bietet dabei stets einen guten Überblick. Der zweite Sensor, der im Körper des Roboters untergebracht ist, ist eine Inertia Messeinheit, die Beschleunigungen in X-, Y- und Z-Achsenrichtung sowie Rotationen um die Roll- und Pitch-Achse messen kann.

Für das Laufen besonders wichtig ist die Kontrolle des Fuß-Boden-Kontakts. Dafür wurde im Rahmen dieser Arbeit ein Drucksensormodul entworfen. Basierend auf Dehnmessstreifen wird dabei der Druck, der auf den Füßen lastet, gemessen. Daraus können Rückschlüsse auf die Bewegungen des Roboters gezogen werden.

Der Fuß besteht neben den beiden Motoren im Wesentlichen aus den Wägezellen (Abb. 4.4). Diese bestehen aus einem Doppelbiegebalken-Federkörper aus Aluminium, auf dem Dehnungs-

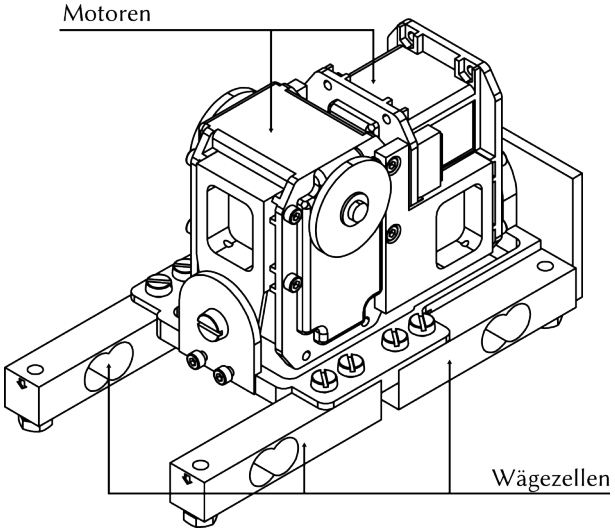


Abb. 4.4: Der Fuß der Roboter beinhaltet vier Wägezellen, die mit Hilfe von Messstreifen den Druck, der auf den Füßen lastet, messen können. Technishce Zeichnung von Jan Streckenbach und Moritz Fröhlich.

messstreifen aufgebracht sind. Diese bilden eine Wheatstonesche Messbrücke[2]. Durch die Verformung der Federkörper verändert sich der Widerstand der Dehnungsmessstreifen und damit der Spannungsabfall bei konstanter Eingangsspannung. Diese Spannungsänderung wird gemessen und darüber der Druck bestimmt, der auf den Sensoren lastet.

4.2.1 Drucksensoren

Die Wägezellen im Fuß des Roboters entstammen handelsüblichen Digitalwaagen. Sie haben einen Wiegebereich bis maximal 5 kg. Da die Spannungsänderungen der Messstreifen sehr klein sind, ist eine Verstärkung notwendig. Zu diesem Zweck kommt ein Operationsverstärker (OP) der Firma National Semiconductors zum Einsatz. Dieser ist ein Rail-to-Rail OP. Er kann also bis zur Versorgungsspannung verstärken. Dies hat den Vorteil, dass die gesamte Auflösung der Analog-Digital-Wandler genutzt werden kann. Er wird in einer klassischen Differenzverstärkerschaltung betrieben (Abb. 4.5). Dabei wird der Widerstand R_2 zur Einstellung des Verstärkungsfaktors genutzt. Der Kondensator C_1 dient zur Tiefpassfilterung des Signals.

Besondere Aufmerksamkeit musste auf Grund von Platzbeschränkungen der Größe der Schaltung geschenkt werden. Da vier Wägezellen pro Fuß zum Einsatz kommen, konnte nicht eine typische Messverstärkerschaltung mit drei Operationsverstärkern eingesetzt werden. Es musste stattdessen eine vereinfachte Schaltung verbaut werden. Mit der eingesetzten Schaltung sind nur drei SMD-Bauteile pro Wägezelle und ein IC für alle vier OPs von Nöten.

[2]Eine Wheatstonesche Messbrücke ist eine Schaltung, mit der auch kleine Widerstände über den Spannungsabfall gemessen werden können. Nähere Erläuterungen sind z.B. in [50] zu finden.

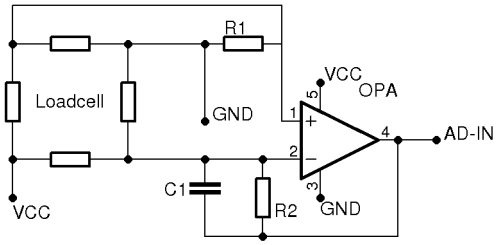


Abb. 4.5: Die Wägezellen werden mit einem Operationsverstärker in Differenzverstärkerschaltung betrieben. Wegen ihrer hochohmigen Eingänge kann dabei ein Einstellungswiderstand eingespart werden.

Die verstärkten Signale werden in einen Mikrocontroller gespeist. Zum Einsatz kam der Atmel ATmega8. Dieser Prozessor ist ein 8-Bit RISC Prozessor, der schon bei anderen Sensorboards des Teams FUmanoid zum Einsatz kam [31]. Er wird auch in den Dynamixel-Motoren genutzt.

Der Mikrocontroller hat die Aufgabe, die analogen Werte in ein digitales Format zu wandeln und dann in das Dynamixel-Protokoll einzubetten. So können die Fußsensoren über den gleichen Bus angesteuert werden wie die Motoren und die Inertia Messeinheit. Der ATmega8 hat zu diesem Zweck acht 10-Bit AD-Wandler und ein serielles Interface zur Ausgabe. Die Software anderer Sensorboards konnte zu diesem Zweck weiter genutzt werden und musste nur geringfügig erweitert werden.

Da die Wägezellen aus Waagen ausgebaut wurden, war nicht ersichtlich, um welches konkrete Fabrikat es sich handelt. Folglich musste ihre Linearität geprüft werden. Außerdem musste geprüft werden, ob die Ausgabe verschiedener Wägezellen gleich ist. Dazu wurden die Wägezellen einzeln mit definierten Gewichten belastet und eine Messreihe erstellt (Abb. 4.6). Dabei wurde festgestellt, dass die Wägezellen einen proportionalen Spannungsanstieg zur Gewichtsbelastung aufweisen. Diese Spannungen sind jedoch um einen wägezellenspezifischen Wert verschoben. So gehen manche Wägezellen nach der Verstärkung schon deutlich vor der Maximalbelastung von 5 kg in Saturierung. Da aber vier Wägezellen pro Fuß bei einem Gesamtgewicht des Roboters von 4.8 kg eingesetzt werden, verteilt sich der Druck ausreichend. Auch die hochohmigste Wägezelle geht erst in die Satuierung, wenn sich der Roboter bereits im Fallen befindet.

Problematischer gestaltet sich die Anbringung der Sensoren. Da auf einem Teppich gespielt wird, sinkt der Fuß des Roboters einige Millimeter ein. So kann nicht immer sichergestellt werden, dass die Druckbelastung wirklich auf den vorgesehenen Punkten wirkt. Dadurch ergeben sich zum Teil erhebliche Abweichungen, die nur durch Änderungen in der Konstruktion vermieden werden können. Viele Stabilisierungsansätze erfahren dadurch deutliche Einbußen.

In den meisten Fällen genügt als Sensorausgabe, ob ein Fuß auf dem Boden ist oder nicht. Da die Messleitungen leicht zerstörbar sind, kann es während einer Spielsituation immer zu Ausfällen einzelner Wägezellen kommen. Ein einfacher Schwellenwert würde dann falsche Resultate liefern. Um das System robuster gegen solche Ausfälle zu gestalten, wird eine exponentielle Glättung genutzt. Mit ihr wird der Schwellenwert S_n zum Zeitpunkt n bestimmt. Dabei kommt ein Parameter ϑ zum Einsatz, mit dem eingestellt werden kann, wie schnell sich der Schwellenwert anpasst. Je

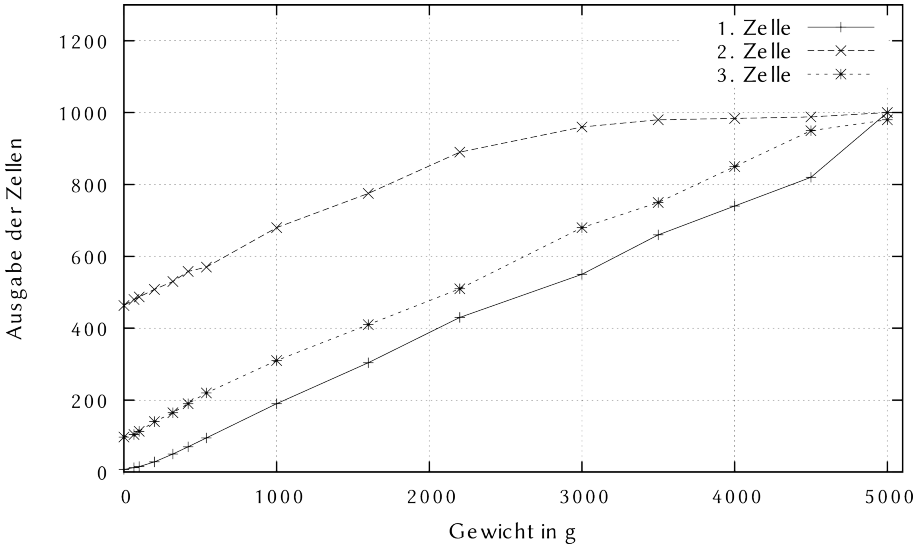


Abb. 4.6: Verschiedene Wägezellen haben stets einen linearen Anstieg proportional zur Belastung. Nur sind sie um einen spezifischen Wert verschoben. Dadurch gehen manche Zellen zu früh in die Saturation (hier die 2. Zelle). Da aber so hohe Gewichte nie in einer relevanten Situation auf einer Wägezelle lasten, wird das Problem nicht weiter betrachtet.

kleiner ϑ ist, desto schneller ändert sich S_n. Seien M_1 – M_4 die gemessenen Druckwerte in einem Fuß, so gilt

$$S_n = \vartheta \cdot S_{n-1} + (1 - \vartheta) \cdot (M_1 + M_2 + M_3 + M_4).$$

Mit dieser rekursiven Formel passt sich der Schwellenwert automatisch Änderungen an. Da während eines Ganges die Füße abwechselnd auf dem Boden und in der Luft sind, ergibt sich ein Mittelwert, der gut zwischen beiden Situationen separieren kann. Um zu hohe oder zu niedrige Schwellenwerte zu vermeiden, wird S_n nicht aktualisiert, wenn beide Füße auf dem Boden sind oder wenn kein Fuß den Boden berührt (z.B. wenn der Roboter gerade getragen wird). Dann gilt $S_n = S_n - 1$. Als Startwert wird der Mittelwert zwischen einem Fuß in der Luft und einem Fuß auf dem Boden gewählt. Dieser Startwert wird manuell vor einem Spiel ermittelt.

4.3 Software und Rechnerplattform

Die Roboter des Teams FUmanoid haben als zentrale Recheneinheit einen ARM Cortex A8 Prozessor mit 1 GHz Taktung auf einem IGEP Board mit 512 MB RAM. Auf diesem Board sind alle essentiellen Schnittstellen vorhanden: Ethernet, IEEE 802.11 b/g, USB und RS232. Zusätzlich sind weitere Schnittstellen für andere mögliche Einsätze vorhanden. Eine Micro-SD-Karte kann direkt genutzt werden und beinhaltet die Software.

Daneben kommt ein Mikroprozessorboard zum Einsatz, das die Motoransteuerung übernimmt. Hier wird ein 72 MHz ARM Cortex M3 eingesetzt. Er empfängt die Motordaten des Hauptrechners

über eine RS232-Verbindung und fungiert als Proxy zum Motorbus. In späteren Versionen sollen auf den Mikrocontroller auch viele Teile der Bewegungssteuerung ausgelagert werden.

Auf dem Hauptprozessor läuft ein OpenEmbedded Linux-System, welches den Zugang zur Hardware abstrahiert. Darauf läuft die FUmanoid-Software. Sie ist eine Selbstentwicklung des Teams FUmanoid und steuert alle Bereiche des Roboters.

Bewegungen werden in diesem System in statische und dynamische Beqwegungen unterteilt. Statische Bewegungen sind auf Keyframetechnik basierende, abgespeicherte Bewegungen, die immer wieder gleichförmig abgespielt werden. Diese Bewegungen haben ein breites Einsatzfeld. Von einfachen Unterhaltungsbewegungen wie Winken, Applaudieren oder Jubeln über Bewegungen in technischen Wettbewerben, beispielsweise das Greifen eines Balls, bis hin zu komplexen spielkritischen Bewegungen wie Aufstehen und Schießen werden statische Motions eingesetzt.

Dynamische Motions hingegen sind Bewegungen, die geregelt werden, also in ihrer Ausführung von Sensorfeedback abhängen. Zur Zeit gibt es in der FUmanoid Software ausschließlich das Laufsystem, den *Walker*, als dynamische Motion. Er ist zusätzlich noch parametrisiert, um dem Roboter zu ermöglichen, in verschiedene Richtungen zu gehen oder die Geschwindigkeit anzupassen.

5 Lösung der Inversen Kinematik

Trajektorien zur Gangerzeugung im Raum der Motorwinkel zu generieren ist sehr mühsam, zeitaufwendig und fehleranfällig. Intuitiver können solche Trajektorien im euklidischen Raum mittels kartesischer Koordinaten für die Endeffektoren angelegt werden. Dazu ist es nötig, die inverse Kinematik des Roboters zu lösen (vgl. Abschnitt 2.2). Dies wird auf zwei Arten gemacht. Zum einen wird eine geometrische Lösung genutzt. Sie lässt sich sehr schnell berechnen und kann deshalb in Echtzeit auf dem Roboter ausgeführt werden. Trajektorien können auf dem Roboter in kartesischen Koordinaten gespeichert bzw. erzeugt und zur Laufzeit in Gelenkwinkel umgerechnet werden. Die geometrische inverse Kinematik umfasst allerdings nur die Füße als Endeffektoren. Zur Optimierung des Ganges wird aber auch der Masseschwerpunkt als Endeffektor betrachtet. Um diesen integrieren zu können, wird eine inverse Kinematik auf Basis der Jakobi-Matrix erstellt. Sie wird nur zur Vorberechnung von Trajektorien eingesetzt.

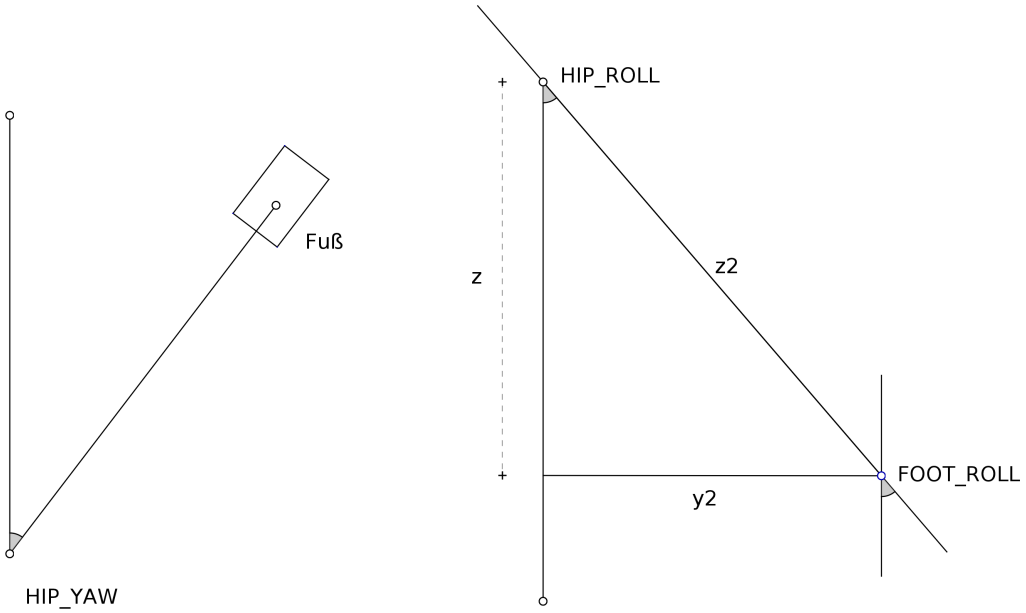
5.1 Geometrische Lösung

Auf Basis der Lösung der inversen Kinematik der Roboter der Darmstadt Dribblers [48] und der alten FUmanoid-Modelle [36] wurde eine geometrische Lösung der inversen Kinematik der Beine entwickelt. Dabei musste die Parallelogramm-Mechanik mit einbezogen werden. Diese führt dazu, dass das Knie des Roboters immer im rechten Winkel zum Oberkörper steht. Die inverse Kinematik berechnet aus der gewünschten Position des Fußes und seiner Orientierung die Gelenkwinkel des Beines:

$$ik \begin{pmatrix} x \\ y \\ z \\ yaw \end{pmatrix} = \begin{pmatrix} HIP\_YAW \\ HIP\_ROLL \\ KNEE\_TOP \\ KNEE\_BOTTOM \\ FOOT\_ROLL \end{pmatrix}.$$

Dabei dient die Hüfte als Ursprung. Zur Vereinfachung wächst die z-Achse nach unten. Die beiden Gelenke HIP_PITCH und $FOOT\_PITCH$ bleiben stets gleich. So wird gewährleistet, dass die Bewegung der Füße nur über Parallelogramm-Mechaniken gesteuert wird und die Füße immer parallel zum BOden gehalten werden. Zur Berechnung wird schrittweise durch die drei Ebenen gegangen: Transversalebene (Blick von oben), Frontalebene (Blick von vorne) und Sagittalebene (Blick von der Seite). In jeder dieser Ebenen können einzelne Gelenke berechnet werden. Zuerst erfolgt die Berechnung des HIP_YAW Gelenks in der Transversalebene. Dieses ist das einzige Gelenk, das die Ausrichtung des Beines beeinflusst. Es wird daher einfach auf den gewünschten Winkel gesetzt (Abb. 5.1(a)):

$$HIP\_YAW = yaw.$$



(a) Ansicht der Transversalebene. Das HIP_YAW-Gelenk ist das einzige Gelenk, welches um die Yaw-Achse dreht.

(b) Ansicht der Frontalebene. $FOOT\_ROLL$ hält den Fuß parallel zum Oberkörper.

Abb. 5.1: Die ersten beiden Berechnungsschritte können in der Transversalebene und der Frontalebene visualisiert werden.

Durch diese Drehung müssen die x- und y-Koordinaten für die weiteren Berechnungen angepasst werden. Eine Bewegung in x-Richtung wird nun nämlich nicht mehr ausschließlich durch die Pitch-Motoren durchgeführt, sondern auch von Roll-Motoren. Die Koordinaten werden entsprechend des Winkels rotiert:

$$x_2 = x \cdot \cos(yaw) + y \cdot \sin(yaw)$$
$$y_2 = x \cdot \sin(yaw) + y \cdot \cos(yaw).$$

Im nächsten Schritt werden die beiden Roll-Motorwinkel berechnet. Da der Fuß stets parallel zum Boden stehen soll, sind der Hüft- und der Fußwinkel genau entgegengesetzt zueinander. Sie werden mit Hilfe von y_2 wie folgt berechnet (siehe Abb. 5.1(b)):

$$HIP\_ROLL = \tan^{-1}\left(\frac{y_2}{z}\right)$$
$$FOOT\_ROLL = -HIP\_ROLL.$$

Dies ist möglich, da stets $z > 0$ gilt. Andernfalls wäre der Fuß *im* Oberkörper. Falls das Bein in der Frontalebene schräg steht – und damit bereits in der Höhe verkürzt ist – , muss es wieder verlängert werden, damit der Fuß z erreicht. Dafür muss im nächsten Berechnungsschritt z angepasst

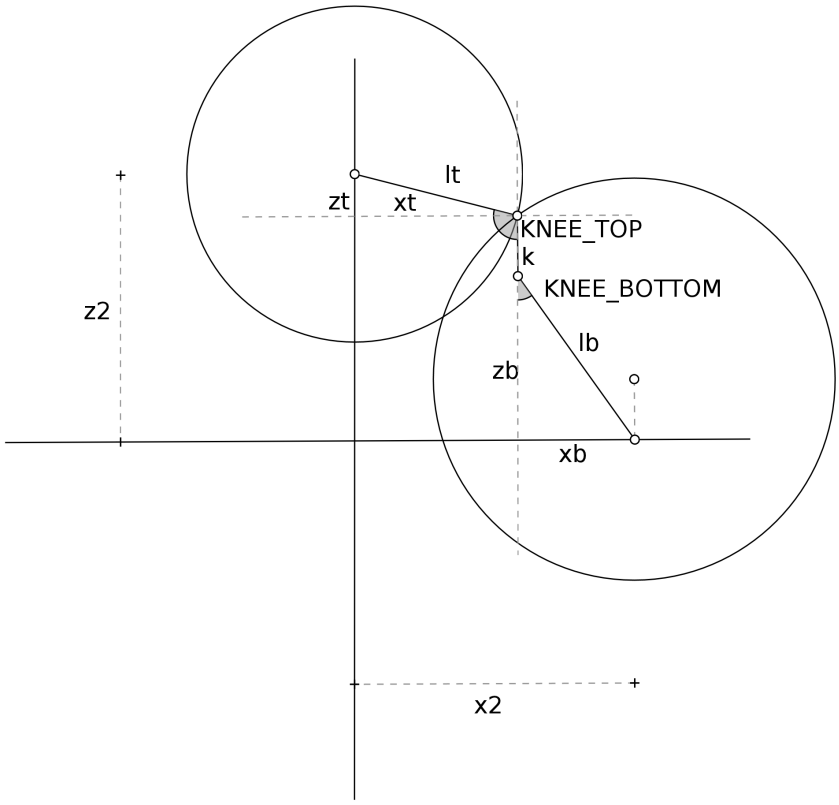


Abb. 5.2: Ansicht der Sagittalebene, die beiden Schnittpunkte der Kreise bestimmen die Position des Knies.

werden:

$$z_2 = \sqrt{{y_2}^2 + z^2}.$$

Im letzten Schritt werden die Pitch-Motorwinkel bestimmt. Da hier zwei Motoren zum Einsatz kommen, gibt es in den meisten Fällen zwei mögliche Lösungen. Zuerst müssen die beiden möglichen Kniepositionen berechnet werden. Dies geschieht, wie in Abbildung 5.2 gezeigt, durch die Berechnung der Schnittpunkte zweier Kreise. Diese haben ihre Mittelpunkte einmal in der oberen Drehachse des Oberschenkels sowie in der um die Länge k des Knies nach oben verschobenen Drehachse des Unterschenkels. Dadurch kann das feste Kniestück für die Berechnung ignoriert werden, da es ein konstanter Offset im rechten Winkel zum Oberkörper ist.

Der Schnittpunkt dieser beiden Kreise lässt sich mit Hilfe quadratischer Gleichungen berechnen. Der Mittelpunkt des oberen Kreises liegt stets in $(0; 0)$, da er sich im Hüftgelenk befindet. Die Endeffektorposition ist in der Sagittalebene $(x_2; z_2)$. Damit ist der Mittelpunkt des unteren Kreises bei $(x_2; z_2 - k)$. Die Längen des Ober- bzw. Unterschenkels sind konstant l_t bzw. l_b. Durch Einsetzen in die Kreisgleichungen und Abziehen der beiden Kreisgleichungen voneinander ergibt

sich die lineare Funktion

$$y = mx + b$$

mit

$$m = \frac{-x_2}{z_2 - k}$$
$$b = \frac{-{l_b}^2 + {l_t}^2 + (z_2 - k)^2 + {x_2}^2}{2 \cdot (z_2 - k)}.$$

Diese Gerade geht durch die beiden Schnittpunkte der Kreise. Wird sie mit der Kreisgleichung eines der beiden Kreise gleichgesetzt, ergibt sich eine quadratische Gleichung, deren beide Lösungen x_S die Schnittpunkte der Kreise sind. Diese kann z.B. mittels der pq-Formel berechnet werden, wobei

$$p = \frac{2 \cdot m \cdot b}{1 + m^2}$$
$$q = \frac{b^2 - l_t}{1 + m^2}$$
$$x_S = \pm\frac{p}{2} + \sqrt{\frac{p^2}{4} - q}.$$

Da nur eine Lösung notwendig ist, wird eine der beiden Lösungen ignoriert. Um eine menschenähnlichere Bewegung zu erzeugen, soll das Kniegelenk möglichst nicht überstreckt werden. Dies wird erreicht, indem stets das x_S gewählt wird, welches größer ist. So wird eher der Oberschenkel angehoben, als dass der Unterschenkel nach vorne gestreckt wird.

Um y_S zu berechnen, wird x_S in die lineare Gleichung eingesetzt:

$$y_S = m \cdot x_S + b.$$

Damit ergibt sich die Knieposition und darüber die beiden rechtwinkligen Dreiecke (Abb. 5.2), mit deren Hilfe die beiden Gelenkwinkel berechnet werden können. Die Seitenlängen der Dreiecke berechnen sich als

$$x_t = x_S$$
$$y_t = y_S$$
$$x_b = x - x_S$$
$$y_b = y - y_S - k.$$

Daraus errechnen sich die beiden Winkel mit

$$KNEE\_TOP = 90° + \cos^{-1}\left(\frac{x_t}{l_t}\right)$$
$$KNEE\_BOTTOM = \sin^{-1}\left(\frac{x_b}{l_b}\right).$$

So sind alle fünf Gelenke, die bewegt werden sollen, eindeutig berechenbar und die inverse Kinematik ist lösbar.

5.2 Lösung mit Jakobi-Matrix

Als zweite Lösung zur Offline-Generierung von Trajektorien zur Schwerpunktbewegung wurde ein Ansatz mit Jakobi-Matrix gewählt. Dazu wurde zuerst die Vorwärtskinematik mit Hilfe der Denavit-Hartenberg-Konvention gelöst. Dabei muss die Parallelogramm-Mechanik mit ihren Besonderheiten integriert werden. Für die Berechnung der Vorwärtskinematik ist dies einfach lösbar, indem jedes Parallelogramm-Gelenk als zwei Gelenke aufgefasst wird, bei dem das zweite Gelenk den invertierten Winkel des ersten Gelenks einnimmt.

Bei der Jakobi-Matrix hingegen, welche die Ableitung der Transformationen darstellt (vgl. 2.2), kann diese zusätzliche Einschränkung nicht einfach hinzugefügt werden. Stattdessen muss die Ableitung nach den Parallelogramm-Gelenkwinkeln angepasst werden. Während normale Gelenke alle getragenen Körper um ihre Achse drehen, drehen sich bei Parallelogramm-Gelenken nur die Körper des Parallelogramms. Alle Körper danach verschieben sich parallel dazu. Die Ableitung lautet also nicht mehr

$$\frac{\partial}{\partial q_i} = z_i \times (p_{eff} - p_i)$$

sondern

$$\frac{\partial}{\partial q_i} = z_i \times (p_{i+1} - p_i).$$

Dabei ist p_{i+1} das Ende des Parallelogramms. Es bewegen sich alle Objekte nach dem Parallelogramm gleich (Abb. 5.3). Auch die Ausrichtung der Objekte bleibt offensichtlich gleich. Die Ableitung der Ausrichtung ist daher stets 0. Der Masseschwerpunkt hingegen ändert sich. Daher ändert sich auch seine Jakobi-Matrix. Ihre Einträge werden zu

$$v_i = \sum_{j=i}^{n} \left[\frac{m_{aj}}{m} d(j) \right]$$

mit

$$d(i) = \begin{cases} z_i \times O_i G_{ai} & \text{falls Gelenk i normales Gelenk} \\ z_i \times O_i p_{i+1} & \text{falls Gelenk i Parallelogramm-Gelenk .} \end{cases}$$

Mit Hilfe dieser Jakobi-Matrizen ist es nun möglich, mehrere Aufgaben zur Bewegung von Endeffektoren mit einer Matrix zu behandeln. Dafür werden die verschiedenen Jakobi-Matrizen zu einer großen Jakobi-Matrix zusammen gefügt:

$$J = \begin{pmatrix} J_1 \\ J_2 \\ \dots \\ J_n \end{pmatrix}.$$

Von dieser kann nun die Pseudoinverse gebildet werden. Sie wird zur Lösung aller Aufgaben in einem Schritt genutzt. Dazu wird ein Aufgabenvektor P gebildet, der alle Zielpositionen der Endeffektoren beinhaltet. Die Lösung der inversen Kinematik lautet dann:

$$\dot{q} = J(q)^{-1} \cdot P.$$

Es handelt sich auch hierbei um eine lineare Approximation. Größere Sprünge können also nicht in einem Schritt berechnet werden.

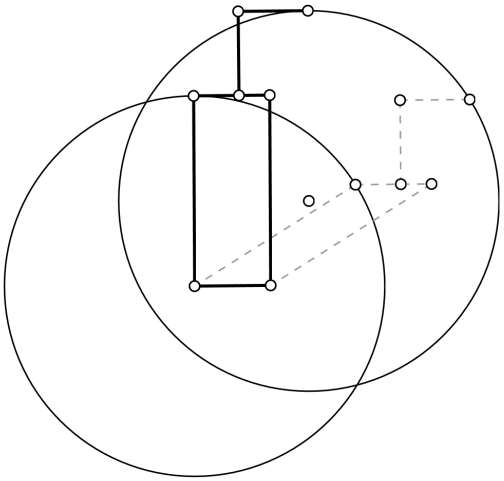


Abb. 5.3: Wird ein Parallelogramm geschert, so drehen sich alle Objekte, die daran befestigt sind, auf einem Kreis des gleichen Radius. Nur der Mittelpunkt ist verschoben. Alle Objekte nach dem Parallelogramm bewegen sich also parallel.

6 Das Laufsystem

6.1 Aufbau

6.1.1 Anforderungen

Um in einem RoboCup-Wettbewerb erfolgreich zu sein, kommt dem Laufen eine wichtige Rolle zu. In den letzten Jahren zeigte sich, dass vor allem solche Teams Siege erringen konnten, deren Roboter stabil und schnell laufen. Die wichtigste Anforderung stellt also die Stabilität des Laufens dar. Zwar sind die FUmanoid-Roboter in der Lage autonom aufzustehen, dabei geht jedoch wertvolle Spielzeit verloren, in der gegnerische Spieler den Ball erobern können. Stürze sollten also so selten wie möglich vorkommen und, wann immer es geht, durch Stabilisierungsmethoden verhindert werden.

Ähnlich wichtig ist auch die Geschwindigkeit der Roboter. Sie entscheidet darüber, ob der eigene Spieler vor dem gegnerischen Spieler am Ball ist und so das Spiel unter Kontrolle gebracht beziehungsweise gehalten werden kann. Auch im Zweikampf ist eine hohe Geschwindigkeit von Vorteil, da der Roboter so mit dem Ball Gegnern entkommen kann.

In solchen Situationen ist auch die Flexibilität des Ganges wichtig. Während es auf einem freien Feld für den Roboter leicht möglich ist, sich auszurichten und dann vorwärts zu gehen, muss er im direkten Gegenspiel in der Lage sein, sich in alle Richtungen zu bewegen. Dieser sogenannte *Omniwalk* hilft dabei, sich mit dem Ball stets zum gegnerischen Tor auszurichten und Gegner mit Blick auf dieses zu umspielen. Für dieses Verhalten ist auch das sogenannte *Orthodribbeln* wichtig. Der Roboter bewegt sich dabei mit gegeneinander nach vorne beziehungsweise hinten verschobenen Füßen, um so den Ball seitwärts vor sich her zu treiben.

Da durch den hohen Schwerpunkt der Roboter ein Schuss mittels statischer Motion, wie er bei den früheren Modellen erfolgreich eingesetzt wurde, nur schwer zu stabilisieren ist, wird weithin gefordert, dass aus dem Laufen heraus ein kurzer, schnell auszuführender Schuss möglich sein soll.

Neben diesen durch das Spiel bedingten Anforderungen existieren auch technische Anforderungen und Anforderungen an die Dokumentation und Portierbarkeit. Um stabil laufen zu können, müssen die Bewegungen sehr glatt sein und Endeffektorpositionen dürfen nicht ruckartig springen. Dafür sind mindestens 80 Positionsberechnungen pro Sekunde erforderlich. Daher muss die Berechnung der Trajektorien und der inversen Kinematik auf der vorhandenen Hardware in weniger als 5 ms berechnet werden. So ist gesichert, dass auch die anderen Robotermodule wie Bildverarbeitung und Verhalten genügend Rechenzeit zur Verfügung haben.

Aufgrund der hohen Fluktuation in einem studentischen Team wie den FUmanoids ist es notwendig, dass der Quelltext und die Konzepte des Walkermoduls gut dokumentiert und wartbar gehalten werden. Außerdem sollte der Quelltext möglichst portierbar sein, da laufend Veränderungen an der Plattform vorgenommen werden. So sind für die nächste Saison bereits neue Beine geplant und der Umstieg auf eine schnellere Prozessoreinheit ist auch absehbar. Um in solchen Fäl-

len nicht das Modul komplett neu implementieren zu müssen, sollten alle hardwarespezifischen Parameter leicht angepasst werden können.

6.1.2 Konzeption

Das grundlegende Konzept des Laufmoduls ist es, die Energieeffizienz eines Passive Dynamic Walkers auch für den RoboCup nutzen zu können. Da die Dynamixel-Motoren auch im ausgeschalteten Zustand eine zu große Bremswirkung haben und die Beine der Roboter recht schwer ausfallen, um genügend Kraft zum Schießen zu haben, ist es nicht möglich, mit diesem Robotermodell direkt einen Passive Dynamic Walker zu verwirklichen.

Daher werden die Analysen der verschiedenen Passive Dynamic Walker Systeme genutzt, um eine Trajektoriengenerierung zu entwerfen, die ein ähnliches Verhalten wie passive Laufmaschinen aufweist. Trotzdem sind alle Bewegungen vollständig aktuiert. Dies hilft auch, im Spiel unvorhersehbare Bewegungen zu vermeiden.

Da Passive Dynamic Walker nur in der Lage sind, vorwärts zu gehen, beim Fußballspielen aber auch seitliche Bewegungen und Drehungen notwendig sind, werden weitere Methoden benötigte, um diese Bewegungen ausführen zu können. Die Trajektorien dafür werden mit Hilfe des ZMP Kriteriums voroptimiert. Neben diesen voroptimierten Trajektorien kommen zusätzliche Stabilisierungsregler zum Einsatz.

Die verschiedenen Trajektoriengeneratoren werden dann kombiniert, um in jede Richtung gehen zu können. Dafür sind sie auch parametrisierbar. Diese Parameter sind direkt linear abhängig zur Schrittweite. So kann die Geschwindigkeit des Roboters leicht gesteuert werden.

6.1.3 Architektur

Der Walker der FUmanoids hat eine einfache Struktur, um die Echtzeitanforderungen realisieren zu können. Er läuft in einem Thread und berechnet zyklisch die Positionsdaten der beiden Beine und Arme. Für die zeitliche Regulierung kommt dazu je ein Timer pro Bein zum Einsatz. Diese laufen im Intervall von $[0; 360]$ Ticks. Diese Taktung wurde gewählt, da die Oszillatoren, welche die Trajektorien generieren, größtenteils auf trigonometrischen Funktionen basieren und so genau ein Zyklus abgeschlossen ist. Die Länge eines Ticks kann über einen Parameter eingestellt werden. Dabei gilt, dass 360 Ticks zwei Schritten entsprechen, einer mit dem linken Bein, einer mit dem rechten. Ein Tick entspricht bei n Schritten pro Sekunde $\frac{1000}{180 \cdot n} ms$. Ein *Frame*, eine Berechnung aller Endeffektorpositionen, hat dabei keine vorgegebene Länge von Ticks oder ms. Es wird nur die Geschwindigkeit von mindestens 80 Frames pro Sekunde gefordert. Trotzdem wird intern ein Frame als atomar angenommen. Die Zeit wird also nur zu Beginn des Frames gemessen und danach der gespeicherte Wert genutzt. Dies verhindert, dass mehrere Messungen leicht unterschiedliche Zeiten ergeben und Positionen, die gesetzt werden, daher nicht mehr zueinander passen.

Ebenfalls frameweise werden die Sensoren gelesen. Für das Laufen relevant sind dabei hauptsächlich die Drucksensoren (siehe Abschnitt 4.2.1). Mittels Inertialeinheit wird zusätzlich festgestellt, ob der Walker stürzt. Ist dies der Fall, so wird eine statische Abrollbewegung – abhängig von der Sturzrichtung – abgespielt, die dafür sorgt, dass der Roboter in einer definierten Position liegenbleibt und die statische Aufstehmotion abspielen kann.

Die Drucksensordaten werden dann dazu genutzt zu erkennen, ob ein Fußaufprall vorgekommen ist. Diese Information führt zu einem regelmäßigen Schrittwechsel. Tritt der Aufprall in normalen Abständen auf, so wird der nächste Schritt gestartet, indem der Timer des jeweils anderen Beins zurückgesetzt wird. Tritt ein Aufprall zu früh auf, so wartet der Roboter mit dem nächsten Schritt, um wieder in die übliche Frequenz zurückfallen zu können. Tritt der Aufprall zu spät ein, so werden entsprechende Stabilisierungsmaßnahmen gestartet. Dieses Timing ist zentral. Es führt dazu, dass eine gleichmäßige Schrittfrequenz eingehalten wird. Dadurch ist der Roboter in der Lage, sich im Laufen zu stabilisieren (siehe Abschnitt 6.2.1).

Neben den Sensordaten kommt die zweite relevante Information vom Verhaltensmodul. Dieses sagt ihm, in welche Richtung und wie schnell er zu gehen hat. Diese Daten werden nur nach jedem Schritt nach einer entsprechenden Vorverarbeitung und Filterung gespeichert. So ist garantiert, dass ein Schritt wie zu seinem Beginn geplant zu Ende geführt wird und keine Richtungsänderung der Beine während der eigentlichen Bewegung auftritt.

Nach dieser Filterung werden die Daten an die Oszillatoren, welche die Trajektorien generieren, weitergegeben. Die Werte dieser Oszillatoren werden zum einen als Positionswerte für die Endeffektoren genutzt, zum anderen auch für die Bewegung von Motoren, die sich zusammen mit bestimmten Endeffektoren bewegen, um so beispielsweise den Armschwung zu erzeugen.

Die Positionswerte werden dann an die inverse Kinematik geleitet, welche die Motorenwerte der Beine berechnet. Diese Werte werden dann geschlossen dem Motorbus-Modul übergeben, welches sie an die Motoren schickt. Damit ist ein Frame des Walkers beendet. Ein neues beginnt wieder mit der Zeitmessung. Wird dem Walker das Signal zum Beenden geschickt, so wird noch ein voller Schritt zu Ende geführt, damit ein stabiler Abschluss gewährleistet ist. In Abbildung 6.1 ist der Ablauf in einem Flussdiagramm dargestellt.

6.2 Timing

6.2.1 Schrittfrequenz

Die zeitliche Regulierung der Schrittabfolge ist zentral für die Stabilität des Roboters beim Gehen. Um eine stabile und schnelle Schrittfrequenz zu finden, betrachten wir zuerst das synthetische Rad von McGeer [28]. Er beschreibt dort den Zusammenhang zwischen der Schrittperiode τ, der Schrittlänge und der Geschwindigkeit. Bei einem Pendel ist die Frequenz der Schwingung nicht von der Masse und des zurückgelegten Winkels des Pendels abhängig, sondern ausschließlich von seiner Länge. Da McGeers Laufmaschine ein frei schwingendes Bein hat, kann die Dynamik eines Pendels angenommen werden. Bei einer Beinlänge von 0.5 m wurde von ihm eine Schrittperiode von $\tau = 2.8$ gemessen. Dabei ist τ dimensionslos und wird in $\sqrt{l/g}$ Einheiten angegeben, wobei l die Länge des Beins und g die Erdbeschleunigung ist. So ist τ unabhängig von der Länge des Beins. Damit können Schrittperioden von Beinen unterschiedlicher Länge verglichen werden. Schrittperiode und Frequenz stehen in umgekehrt proportionalen Verhältnis zueinander:

$$f = \frac{1}{\tau \cdot \sqrt{l/g}}.$$

Damit ergibt sich für McGeers Passive Walker eine Schrittfrequenz von ca. 1.54 Hz. Menschen haben bei höheren Geschwindigkeiten eine Schrittfrequenz von ca. 2.0 Hz [4]. Bei einer durch-

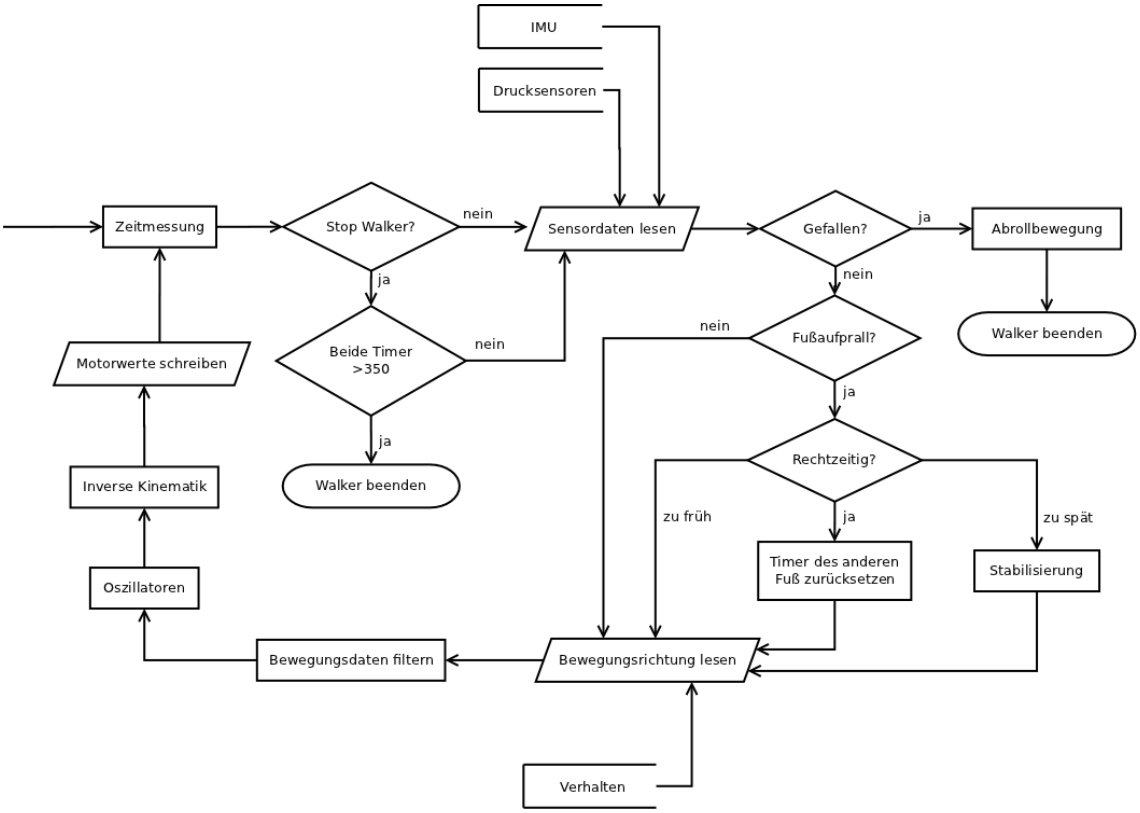


Abb. 6.1: Ein Flussdiagramm des Walkerablaufs

schnittlichen Beinlänge von 0.90 m ergibt das eine deutlich kleinere Schrittperiode von $\tau = 1.65$. Sie können ihre Schrittfrequenz aber auch auf bis zu 2.5 Hz erhöhen, um schneller zu gehen. Dies entspricht einer Schrittperiode von sogar nur $\tau = 1.25$.

Aus diesen Werten können nun Schrittfrequenzen für die FUmanoid-Roboter hergeleitet werden. Für Roboter mit 35 cm langen Beinen ergeben sich Schrittfrequenzen zwischen 1.89 Hz und 4.01 Hz. In diesem Rahmen und leicht darüber hinaus wurden verschiedene Schrittfrequenzen getestet. Um einen möglichst schnellen und trotzdem gleichmäßigen Gang zu gewährleisten, bewährte sich dabei eine Schrittfrequenz am oberen Rand der berechneten Werte. Die Schrittfrequenz kann über einen entsprechenden Parameter eingestellt werden.

6.2.2 Schrittsynchronisierung

Die im vorigen Abschnitt beschriebene Schrittfrequenz kann durch viele Umstände gestört werden. Dabei gibt es zwei unterschiedliche Szenarios. Einerseits kann ein Fuß zu früh, andererseits zu spät den Boden berühren. Die Schritte müssen darum mit der Körperdynamik synchronisiert werden. Eine solche Schrittsynchronisierung führt zu deutlichen Verbesserungen der Stabilität, wie von Moballegh gezeigt wurde [31].

Trifft ein Fuß zu früh auf den Boden auf, so ist es sehr wahrscheinlich, dass der Roboter nach vorne kippt. Dies geschieht, weil zu viel kinetische Energie im System ist. Hierzu wird das Energieverhältnis im Roboter untersucht.

Im einfachsten Walker Modell, dem *Simplest Walker* von Garcia et al. [13], besteht ein Roboter aus einer Punktmasse in der Hüfte und masselosen Beinen, die einen punktförmigen Kontaktpunkt mit dem Untergrund haben. Es wird vereinfachend angenommen, dass während einer normalen Schwungphase keine Energie an die Umwelt übertragen wird. Reibung, Luftwiderstand und ähnliches wird also vernachlässigt. In diesem Fall besteht das System nach der klassischen Mechanik aus der Summe der potentiellen und der kinetischen Energie. Sei m die Masse, h die Höhe über dem Untergrund, g die Erdbeschleunigung und v die Geschwindigkeit, so ist die Energie des Walkers beschrieben (vgl. [31]) mit

$$E = mgh + \frac{1}{2}mv^2.$$

Bei einem durchgestreckten Bein, wie beim Simplest Walker, wandert die Energie immer zwischen potentieller Energie und kinetischer Energie. Zuerst steigt die Masse in der Höhe, dabei verlangsamt sie sich. Danach sinkt sie wieder in der Höhe und beschleunigt. Dabei geht keine Energie an die Umgebung verloren. Die einzige Energieabgabe tritt beim Aufprall des Fußes auf [31]. Die Geschwindigkeit v zum Zeitpunkt des Fußaufpralls kann in zwei Komponenten aufgeteilt werden, zum einen die Geschwindigkeit v_l entlang des Beines, zum anderen die Geschwindigkeit v_t orthogonal dazu. Diese Geschwindigkeit befindet sich auf einer Tangente zum Kreisbogen, den die Punktmasse im nächsten Schritt beschreiben wird. Die Energie aus v_l wird durch den unelastischen Aufprall komplett an den Untergrund abgegeben. Daraus ergibt sich folgende Energie:

$$E_A = mgh_A + \frac{1}{2}mv_t^2.$$

Dabei berechnet sich $v_t = v \cdot \cos\alpha$ mit α als dem Winkel zwischen beiden Beinen. Um nun laufen zu können, muss durch geeignete Aktuation Energie ins System gepumpt werden.

Trifft der Fuß zu früh auf dem Boden auf, so ist das System gestört. Bei einer solchen Störung kann nicht mehr davon ausgegangen werden, dass beide Beine gleich lang sind. Dies hat zwei Implikationen. Zum Ersten ist anzunehmen, dass das Bein relativ steil auftrifft. Das Bein wird nur bis zu einem definierten Grad nach vorne bewegt. Da es zu früh auf dem Boden auftritt, hat es diesen Winkel noch nicht erreicht. Außerdem ist es noch nicht gänzlich gestreckt. Der Roboter muss also stärker um seinen Standfuß rotiert sein, so dass der Winkel zwischen auftreffendem Fuß und Boden nochmals steiler ist. Zum Zweiten ist die Höhe der Masse nun niedriger als bei einem normalen Schritt, da das Bein kürzer ist. Dies vermindert die potentielle Energie. Diese Energie muss also in kinetische Energie umgewandelt worden sein.

Des Weiteren ist der Winkel β zwischen Untergrund und Bein größer. Auf die Energie hat aber nur der Winkel α einen Einfluss (vgl. Abbildung 6.2). Der Drehradius des Beines wird aber deutlich kleiner. Damit wird das Trägheitsmoment um den Aufsetzpunkt kleiner und mit ihm die Rotationsgeschwindigkeit um den Fuß größer. Beim sofortigen Start eines neuen Schrittes würde der Roboter leicht nach vorne fallen, da dieser Schritt schneller ausgeführt werden müsste. Um dem entgegenzuwirken, muss ein Teil der kinetischen Energie wieder in potentielle Energie umgewandelt werden. Dazu wird das Bein verlängert, welches zu früh auf den Boden gekommen ist, ohne einen neuen Schritt zu starten. Damit wird der alte Radius wieder hergestellt. Ferner wird

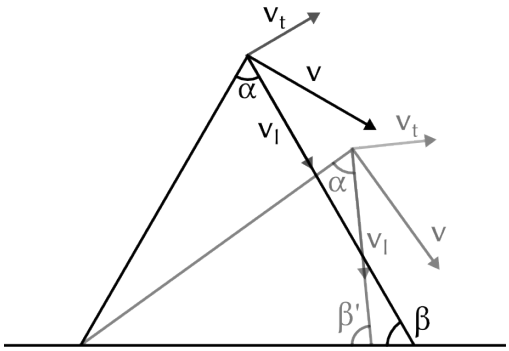


Abb. 6.2: Die Längen der Geschwindigkeitsvektoren v_l und v_t sind unabhängig von der Länge des aufsetzenden Beines. Damit ist die Energie, die der nächste Schritt zur Verfügung hat, unabhängig von der Länge des aufsetzenden Beines. Der Rotationsradius ist jedoch kleiner, die Geschwindigkeit der Rotation steigt also. Es kommt leichter zu einem Fall.

versucht, β wieder zu erreichen, so dass wieder bei einem normalen Schritt begonnen werden kann. Die zusätzlich in das System gebrachte Energie wird so kompensiert.

Dies gilt auch für Energie, die durch Stöße oder Ähnliches zusätzlich eingebracht wird. Der Schritt wird also normal zu Ende geführt, dabei verlängert sich das neue Standbein auf die ursprüngliche Größe und der Fuß-Boden-Winkel wird auf ein normales Maß gebracht. Erst wenn dieser Vorgang beendet ist, wird der neue Schritt gestartet.

Anders verhält es sich, wenn der Fuß zu spät aufkommt. Dies geschieht, wenn kinetische Energie durch Stöße in die Umgebung abgegeben wird. Dies kann nicht durch eine Anpassung der Standbeinlänge kompensiert werden. Wird nun ein neuer Schritt gestartet, bevor der schwingende Fuß den Boden berührt, beginnt der gesamte Roboter abzusinken, da der Standfuß verkürzt wird. Die potentielle Energie, die dabei in kinetische Energie umgewandelt wird, wirkt genau im rechten Winkel zum Untergrund und damit zum Großteil direkt in den Untergrund oder aber einem weiteren Schritt entgegen. Stattdessen muss versucht werden die Masse des Roboters wieder zu beschleunigen. Dies kann durch einfaches Bewegen des Fußgelenks erfolgen, bei stärkeren Störungen sollte aber auch das Hüftgelenk mit einbezogen werden (siehe Abschnitt 6.5.3).

6.3 Trajektoriengenerierung

Zum Generieren der Trajektorien für den Gang wurden zwei verschiedene Methoden angewandt, um damit Oszillatoren zu optimieren, die Endeffektortrajektorien erzeugen. Dabei wurden zum einen Erkenntnisse aus der Analyse von Passive Dynamic Walkern genutzt, zum anderen ZMP-basierte Optimierungsverfahren eingesetzt.

6.3.1 Vorwärtsbewegung mit Passive Dynamic Walking

Die Vorwärtsbewegung des Roboters ist der Bewegung eines Passive Dynamik Walkers nachempfunden. Das Bein einer solchen Laufmaschine verhält sich in seiner Schwungphase wie ein freischwingendes Pendel. Die Phase eines solchen Pendels lässt sich durch eine sinusoide Funktion

beschreiben [17]:

$$\theta(t) = \hat{\theta} \cdot \sin\left(f \cdot t + \phi_0\right).$$

Dabei ist $\hat{\theta}$ die Winkelamplitude, also der maximale Winkelausschlag des Beines. Die Nullphase ϕ_0 ist $-90°$, da das Bein stets hinten beginnt zu schwingen. Die Schrittfrequenz f kann ignoriert werden, da sie bereits in der Skalierung von t integriert ist (siehe 6.2.1). Daraus lässt sich die x-Position des Fußes in der Schwungphase berechnen:

$$x(t) = l \cdot \sin\left(\hat{\theta} \cdot \sin(t + \phi_0)\right).$$

Als Parameter kann nun der Schrittwinkel $\hat{\theta}$ gewählt werden. Eine intuitivere Größe ist aber die Schrittlänge s. Aus dieser Größe kann der Schrittwinkel berechnet werden:

$$\begin{aligned} & s = 2l \cdot \sin\left(\hat{\theta}\right) \\ \Leftrightarrow \qquad & \hat{\theta} = \sin^{-1}\left(\frac{s}{2l}\right). \end{aligned}$$

Um diese Funktion zu approximieren wird die Kleinwinkelnäherung genutzt. Diese besagt, dass für kleine Winkel der Sinus des Winkels gleich dem Winkel (im Bogenmaß) ist. Damit ergibt sich als Approximation eine einfache Sinusfunktion mit der maximalen Schrittlänge als Amplitude:

$$x_{App}(t) = \frac{s}{2} \cdot \sin(t + \phi_0).$$

Die maximale Abweichung dieser Approximation liegt im relevanten Bereich bis $\hat{\theta} = 60°$ unter 10% (vgl. Abb. 6.3(b)). Dies ist insbesondere vertretbar, da der Bereich der maximalen Abweichung in der Mitte des Schrittes und nicht am Beginn oder Ende liegt. Die Schrittlänge wird also nicht beeinflusst.

Während der Standphase des Beines verhält sich der Passive Dynamic Walker ähnlich einem Rad (vgl. Abschnitt 3.3). Ein rollendes Rad, welches nicht beschleunigt oder gebremst wird, dreht sich mit gleichbleibender Winkelgeschwindigkeit um seine Nabe. Die Änderung des Winkels ist also konstant, der Winkel verändert sich linear. Um dieses Verhalten nachzubilden, wird der Winkel θ im Zeitintervall der Standphase $[180, 360]$ linear verändert:

$$\theta(t) = 3 \cdot \hat{\theta} - \frac{2 \cdot \hat{\theta}}{180} t.$$

Wieder ist die Position des Beines mittels trigonometrischer Funktionen zu berechnen:

$$x(t) = l \cdot \sin(\theta(t)).$$

Diese Funktion kann im gegebenen Intervall von $[180, 360]$ wiederum mit einer linearen Funktion approximiert werden:

$$x_{App}(t) = \frac{3s}{2} - \frac{s}{180} t.$$

Auch hier ist die Abweichung im relevanten Bereich unter 10%. Insgesamt führen die Approximationen zwar nicht zu einem erheblichen Geschwindigkeitsgewinn, können aber je nach eingesetzter Recheneinheit wichtig für die Umsetzbarkeit der Trajektoriengeneration sein.

Bisher unbeachtet geblieben ist der Abstand zwischen Fuß und Boden. Er verhindert eine Kollision des schwingenden Fußes mit dem Boden. Zur einfachen Analyse wird dieser Abstand bei vielen Passive Dynamic Walking Modellen vernachlässigt. Stattdessen wird die nicht-physikalische Annahme getroffen, dass während des Schwingens keine Kollision mit dem Boden auftreten kann, das Bein also durch den Boden schwingen kann [13, 28]. Bei der Realisierung von Laufmaschinen mussten dann aber Möglichkeiten zur Beinverkürzung geschaffen werden. Dazu können Teleskopgelenke dienen [28]. Als energieeffizienter stellen sich aber Knie heraus, die sogar komplett passiv betrieben werden können [29].

Bei einem solchen Aufbau beschreibt der Abstand zwischen Fuß und Boden eine doppelte Kurve. Nachdem sich der Fuß zu Beginn des Schrittes vom Boden entfernt, kommt er ihm in der Mitte des Schrittes nahe, um dann noch einmal Abstand zu gewinnen, bevor er schließlich wieder aufkommt (Abb. 6.3(d)). Zwar nutzt diese Kurve optimal die Energie aus, es kann aber leicht zu Kollisionen zwischen Fuß und Boden kommen. Da solche Kollisionen unweigerlich zu Stürzen führen, sollten sie möglichst vermieden werden. In [32] wird gezeigt, dass der Abstand zwischen Fuß und Boden ab einem gewissen Minimum keinen Einfluß auf die besonders kritische seitliche Stabilität hat, eine Erhöhung des Bodenabstands ist also nicht problematisch.

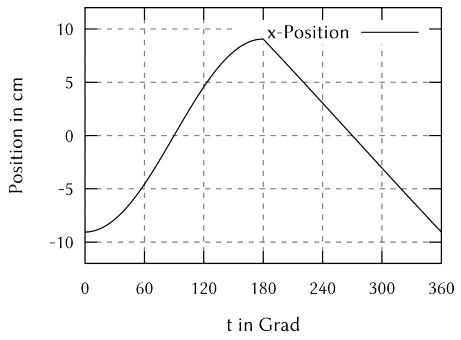
Um auch eine rechnerisch möglichst einfache Lösung zu finden, wird der Fuß einfach auf einer Sinuskurve angehoben. Während der Standphase wird zuerst die z-Position des Fußes sogar noch erhöht, um stets das durchgestreckte Bein nutzen zu können. Dies ist besonders wünschenswert, da ein geknicktes Knie einen erhöhten Energieaufwand bedeutet. Sei h die gewünschte Schritthöhe, so errechnet sich die z-Position als

$$z(t) = \begin{cases} l \cdot \cos(\theta(t)) & \text{falls } 0^\circ \leq t \leq 180^\circ \\ l \cdot \cos(\theta(t)) - h \cdot \sin(t) & \text{falls } 180^\circ < t \leq 360^\circ \end{cases} .$$

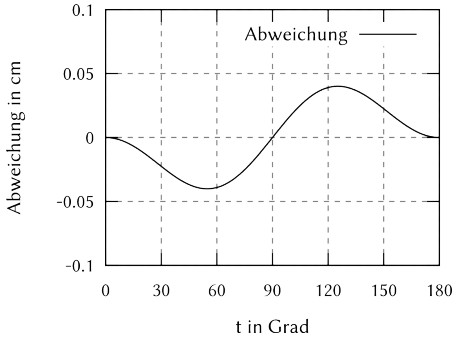
Das ständige Anheben und Absenken des Schwerpunkts kann zu Instabilitäten führen. Es ist daher auch möglich, den Schwerpunkt auf der gleichen Höhe zu halten. Dies führt allerdings zu einem energetisch etwas ineffizienteren Gang.

6.3.2 Seitwärts- und Drehbewegung mit Zero Moment Point

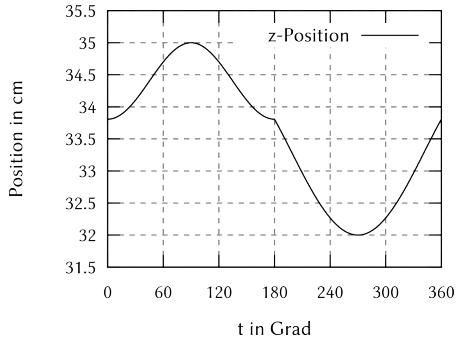
Für Seitwärtsbewegungen ist bisher kein rein passives Laufmodell bekannt. Es ist also nicht ohne weiteres möglich, Trajektorien einer passiven Laufmaschine zu übernehmen. Stattdessen wird eine Modelloptimierung anhand des ZMP durchgeführt. Um einen stabilen Gang zu gewährleisten, muss der ZMP stets im Supportpolygon gehalten werden (vgl. Abschnitt 3.2). Als erstes werden also geeignete ZMP-Trajektorien festgelegt (Abb. 6.4(b)). Der ZMP ist jedoch nicht mittels inverser Kinematik direkt steuerbar. Wie in Abschnitt 2.2.4 gezeigt, ist es aber möglich, den Masseschwerpunkt (CoM, Center of Mass) des Roboters mit inverser Kinetik zu steuern. Wenn sich der Roboter nicht bewegt, so koinzidieren CoM und ZMP. In der Bewegung hat aber die Dynamik des Roboters Einfluss auf die Position des ZMP (beziehungsweise über seine Existenz). Die gesamte Dynamik eines Roboters ist schwer zu modellieren. Daher wird ein vereinfachtes Modell, welches Kajita et al. in [21] beschreiben, genutzt. Bei dem sogenannten 3D-LIPM (3-Dimensional Linear Inverse Pendulum Mode) wird der Roboter als invertiertes Pendel modelliert, das stets die gleiche Höhe hat. Dieses Modell hat eine sehr einfache Dynamik, so dass die Koordinaten des ZMP aus den



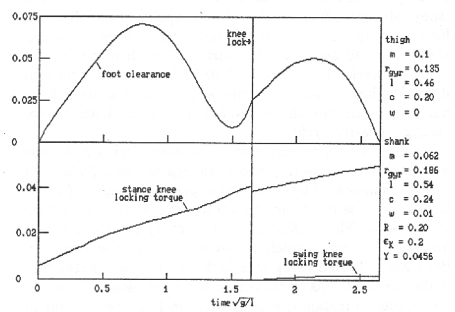
(a) x-Position des Fußes bei einem Schritt mit einem Schrittwinkel von 15°



(b) Abweichung der Approximation bei einem Schrittwinkel von 15°



(c) z-Position des Fußes bei einem Schritt mit einem Schrittwinkel von 15°



(d) Bodenabstand des Fußes eines Passive Dynamic Walkers (aus [29])

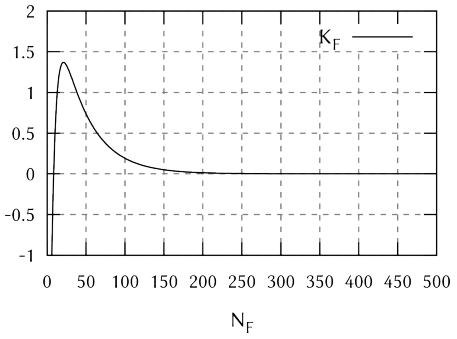
Abb. 6.3: Positionen des Fußes bei der Vorwärtsbewegung

Beschleunigungen leicht zu errechnen sind [21]. Sei z_c die Höhe des Pendels, y und x die Koordinaten des Pendels, $\ddot{y}$ und $\ddot{x}$ die Beschleunigungen des Pendels und g die Erdbeschleunigung, so liegt der ZMP bei

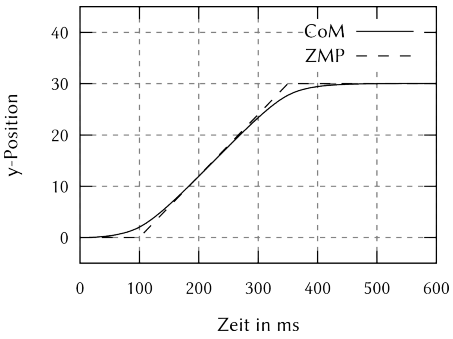
$$p_x = y - \frac{z_c}{g}\ddot{y},$$
$$p_y = x - \frac{z_c}{g}\ddot{x}.$$

Es ist also mittels zweifacher zeitlicher Differenzierung möglich, aus der Bewegung des CoM den ZMP zu berechnen. Zur Bewegungserzeugung ist das Problem aber invers. Aus einer vorgegebenen ZMP Trajektorie soll die Trajektorie des CoM berechnet werden. Diese kann dann mit Hilfe der inversen Kinetik zur Berechnung von Gelenktrajektorien genutzt werden.

Wie in Abb. 6.4(b) zu sehen, muss das CoM bewegt werden, bevor der ZMP reagiert. Bei einer gegebenen ZMP-Trajektorie muss das CoM also bewegt werden, bevor die Trajektorie eine Änderung erfährt. Es ist also notwendig, zukünftige Informationen mit einzubeziehen. Für solche Probleme wurden Preview Controller entworfen [23]. Diese Regler nutzen vorhersehbare Daten, um bereits frühzeitig in den Regelprozess einzugreifen.



(a) Der Reglerparameter K_F als Funktion der Vorhersehlänge



(b) Mittels Preview Controller bestimmte CoM Trajektorie

Abb. 6.4: Der Preview Controller kann eingesetzt werden, um aus einer ZMP-Trajektorie eine CoM-Trajektorie zu generieren.

Neben einem P- und einem I-Teil gibt es bei einem solchen Regler auch einen Teil, der die vorhergesagten Teile integriert. Ein solcher Regler ist im Zeitdiskreten gegeben durch

$$u(t) = K_P e(t) + K_I \sum_{\tau=0}^{t} e(\tau) + \sum_{j=1}^{N_F} K_F(j) p_{ref}(j+t).$$

Dabei ist $p_{ref}(x)$ der vorhergesehene Wert zur Zeit x. Der Faktor $K_F(x)$ erfüllt die Funktion einer Reglerkonstanten, ist jedoch eine Funktion der Zeit. So werden vorhergesagte Werte in unterschiedlicher zeitlicher Entfernung verschieden gewichtet. N_F ist der Zeithorizont, der vorhergesagt werden soll.

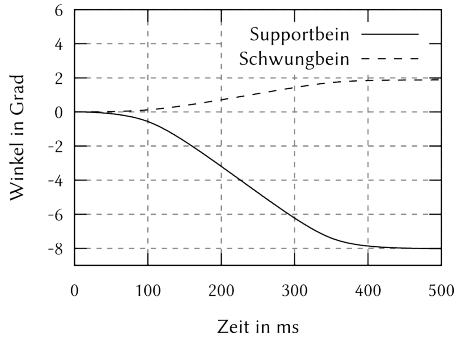
Mit einem solchen Regler kann nun die CoM Position bestimmt werden, sodass der ZMP der vorgegebenen Trajektorie folgt. Die Parameter K_I, K_P und K_F werden wie in [23] beschrieben berechnet. Der Verlauf von $K_F(x)$ ist in Abb. 6.4(a) zu sehen.

Als ZMP-Trajektorie wird eine einfache Stufenfunktion gewählt, wie sie auch in [21] genutzt wird (Abb. 6.4(b)). Diese überführt den ZMP während der Doppelsupportphase möglichst schnell von einem zum anderen Fuß. Als weiteres Kriterium wird eine lineare Überführung des Fußes von seiner bisherigen Position zur Endposition hinzugefügt. Die z-Position während dieser Phase folgt der gleichen Kurve wie bei der Vorwärtsbewegung, um den Gang möglichst leicht in alle Richtungen interpolierbar zu machen.

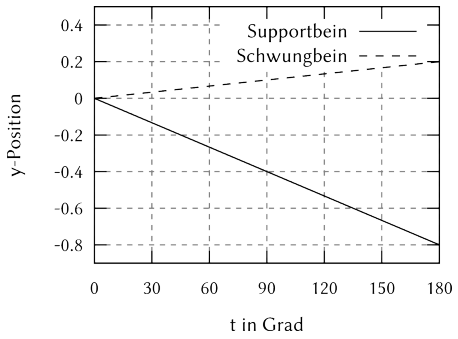
Mit Hilfe der inversen Kinetik (vgl. Abschnitt 2.2.4) werden nun Trajektorien für die Motoren generiert. Besonders interessant sind dabei die HIP_ROLL Gelenke, da sie als einzige die seitliche Bewegung deutlich beeinflussen. Auffällig dabei ist die asymmetrische Auslenkung der beiden Beine (Abb. 6.5(a)). Das Supportbein bewegt sich also stärker als das Schwungbein.

Die Kurven der beiden Winkel γ_{Swing} und $\gamma_{Support}$ werden mit den trigonometrischen Funktionen

$$y_{Support} = l \cdot \sin(\gamma_{Support})$$
$$y_{Swing} = l \cdot \sin(\gamma_{Swing})$$



(a) Die Winkel der beiden Hüftgelenke bewegen sich asymmetrisch beim Seitwärtsgang.



(b) Die daraus resultierenden Kurven sind linear approximierte Positionen der Füße.

Abb. 6.5: Die Hüftwinkel bei der Seitwärtsbewegung

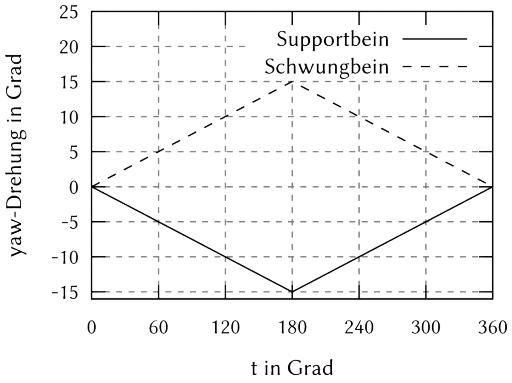


Abb. 6.6: Die Trajektorien der Yaw-Motoren bei der Drehung sind gleichmäßig. Das Standbein dreht dabei in die entgegengesetzte Richtung des Schwungbeins.

in Positionskurven der Füße überführt. Diese werden im Laufsystem linear approximiert (Abb. 6.5(b)). Dabei wird vereinfachend angenommen, dass das Schwungbein einen Anteil von $\frac{1}{4}$ an der Schrittweite hat. Diese Approximation hat zur Folge, dass die Schritte auch seitlich mit der Schrittweite parametrisiert werden können.

Die letzte Bewegung, die noch zu einem kompletten Omniwalk fehlt, ist die Drehbewegung. Auch hier gibt es keine passiven Systeme, die als Vorlage dienen könnten. Deshalb wurde hier ebenfalls mittels inverser Kinematik und Kinetik eine Trajektorienoptimierung durchgeführt. Die z-Bewegung des Beins entspricht wieder der z-Bewegung bei der Vorwärtsbewegung. Die sich ergebenen Kurven werden ebenfalls linear approximiert und führen zu mit dem Drehwinkel parametrisierbaren Kurven, die leicht in das bestehende System eingebunden werden können (Abb. 6.6).

6.3.3 Zusammenführung zum Omniwalk

Mit Hilfe dieser Trajektorien ist es möglich, einzelne Schritte nach Vorne, zur Seite oder mit Drehung durchzuführen. Alle diese Werte sind mit Schrittweite oder Drehwinkel parametrisierbar. Diese Trajektorien können jetzt kombiniert werden, um einen Omniwalk zu erzeugen. Dabei wird keine spezielle Anpassung der Trajektorien vorgenommen, wenn x-, y- und Drehwerte kombiniert werden. Dieser Ansatz wird auch von Behnke et al. [3] verfolgt und führt dort zu einem stabilen Omniwalk. Es entstehen so Interpolationen für alle kombinierten Richtungen.

6.4 Filterung

Der Walker bekommt die Richtung, in die er gehen soll, vom Verhaltensmodul angegeben. Dabei müssen verschiedene Filterstufen durchlaufen werden, um die Bewegung des Roboters zu stabilisieren. Zum einen müssen starke Richtungsänderungen geglättet werden, damit Dynamiksprünge abgefangen werden können. Zum anderen sollte die Richtung, in die gelaufen wird, geglättet werden, um ständig schwankende Bewegungen zu vermeiden. Diese entstehen durch verrauschte Daten, durch die z.B. der Ball immer leicht abweichend gesehen wird. Zuletzt sollten diese Bewegungswerte dann begrenzt werden, damit der Roboter höchstens mit der Geschwindigkeit läuft, mit der er sich noch stabil fortbewegen kann.

6.4.1 Eingangsfilterung

Zu Beginn der Eingabeverarbeitung werden die Bewegungsrichtungen vom Verhalten auf die Einheiten des Walkers umgerechnet. Dies ist eine simple lineare Transformation. Mit diesen Werten werden alle weiteren Berechnungen durchgeführt.

Da die Daten des Verhaltens viel Rauschen beinhalten, sollen die Daten geglättet werden. Dies soll möglichst effizient geschehen, da das Verhalten mehrmals pro Sekunde, unter optimalen Bedingungen 20 mal, neue Daten schickt.

Die einfachste Möglichkeit, die Eingangsdaten d_i zum Zeitpunkt t zu filtern, ist eine gleitende Mittelwertbildung der letzten n Datensätze:

$$x_t = \frac{1}{n} \sum_{i=1}^{n} d_{t-i}.$$

Über die Größe von n lässt sich die Trägheit dieses Filters einstellen. Je größer n ist, umso stärker filtert die Methode. Im Extremfall $n = 1$ filtert dieses Filter gar nicht. Ist n so groß wie alle bisherigen eingegangenen Daten, werden die Daten maximal gefiltert, so dass der Roboter in die mittlere Richtung aller Befehle geht. Dieses Filter bewertet die letzten n Datensätze gleich stark. Dies hat zur Folge, dass Ausreißer besonders ins Gewicht fallen. Eine exponentielle Glättung verhindert dies:

$$x_t = \alpha \cdot x_{t-1} + (1 - \alpha) \cdot d_t.$$

Dies integriert mit x_{t-1} sämtliche vorherigen Messungen. Je weiter sie jedoch in der Vergangenheit liegen, umso schwächer tragen sie zu dem geglätteten Wert bei. Über α kann hier die

Abb. 6.7: Verschiedene Filter zur Filterung der Bewegungsdaten vom Verhalten. Zu sehen sind typische Verhaltensdaten, die mit unterschiedlichen Filtern geglättet werden.

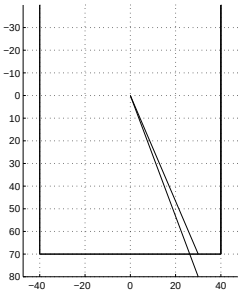
Stärke der Glättung eingestellt werden. Je größer α ist, desto stärker trägt der neue Datenpunkt bei und desto schwächer ist die Glättung. Das Filter hat außerdem den Vorteil, dass es unabhängig von der Trägheit für jeden Schritt konstant viel Rechenzeit benötigt.

Auch komplexere Filter wie das Kalman-Filter können in Betracht gezogen werden. Wie in Abb. 6.7 zu sehen, sind die Unterschiede zwischen exponentieller Glättung und Kalman-Filter aber sehr gering. Dafür ist die Komplexität in der Implementierung und auch Berechnung bei einem Kalman-Filter deutlich höher. Aus diesem Grund wird die exponentielle Glättung eingesetzt.

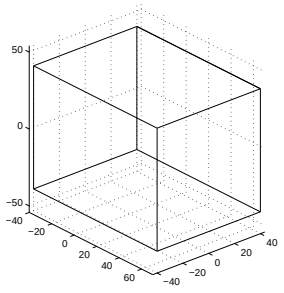
6.4.2 Geschwindigkeitsbegrenzung

Nachdem die Eingangsdaten gefiltert werden, ist es möglich, dass die gewünschten Geschwindigkeiten über den stabilen Maximalgeschwindigkeiten des Roboters liegen. Sie sind aber nicht für jede Richtung gleich. Um markante, leicht zu testende Punkte werden daher Geschwindigkeitsgrenzen gelegt. Überschreiten die geforderten Geschwindigkeiten diese Grenzen, so können sie an diesen abgeschnitten werden. Werden die Geschwindigkeiten als dreidimensionale Werte (Vorwärts, Seitwärts, Drehung) betrachtet, so muss darauf geachtet werden, dass die Winkel von gekürztem und Originalvektor gleich sind. Ist dies nicht der Fall, bewegt sich der Roboter bei großen Geschwindigkeiten nicht in die gewünschte Richtung, sondern weicht von der Richtung ab (siehe Abb. 6.8(a)).

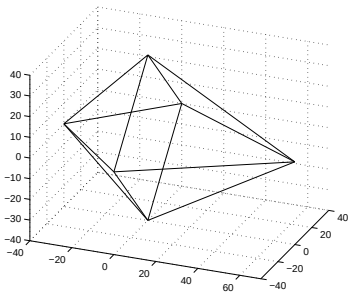
Die einfachste Form von Geschwindigkeitsgrenzen ist ein Quader (Abb. 6.8(b)). Hierbei ist jedoch problematisch, dass die kombinierten Höchstgeschwindigkeiten betrachtet werden müssen. Ansonsten würden bei diagonalen Richtungen Instabilitäten auftreten. Dadurch wird aber die



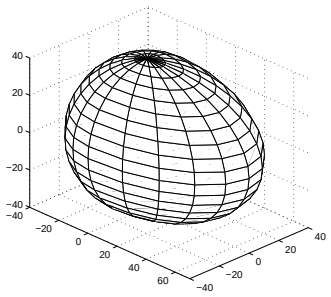
(a) Wird lediglich die zu große Geschwindigkeit gekürzt, ändert sich der Winkel der Bewegungsrichtung.



(b) Ein Quader



(c) Ein Polyeder



(d) Zwei Halbellipsoide

Abb. 6.8: Verschiedene Geschwindigkeitsbegrenzungen mit gleichen Maximalgeschwindigkeiten

Höchstgeschwindigkeit beim einfachen Vorwärtsgehen stärker beschränkt als nötig. Dieses Verfahren wurde bis 2010 eingesetzt. Dabei wurde deutlich, dass die erwähnten Beschränkungen die Geschwindigkeit der Roboter zu stark drosseln. Es soll daher nun abgelöst werden.

Eine bessere Approximation ist es, einen Oktaeder um die Maximalgeschwindigkeiten zu legen (Abb. 6.8(c)). So können die echten Höchstgeschwindigkeiten nach vorne, zur Seite und in der Drehung genutzt werden, um die Geschwindigkeiten zu begrenzen und die experimentell ermittelten Höchstgeschwindigkeiten können tatsächlich erreicht werden. Bei diagonalen Bewegungen werden aber nach wie vor stabile Bereiche nicht genutzt.

Lineare Beschränkungen erweisen sich als nicht hinreichend für die Modellierung von Geschwindigkeitsgrenzen. Aus diesem Grund werden nicht-lineare Grenzen in Betracht gezogen. Für den Fall, dass Vorwärts-, Seitwärts- und Drehhöchstgeschwindigkeiten gleich sind, kann angenommen werden, dass die Höchstgeschwindigkeit in jede beliebige Richtung gleich groß ist. Eine sinnvolle Begrenzung wäre in diesem Fall eine Kugel. Wird eine solche Kugel in eine Richtung gestreckt, ergibt sich ein Ellipsoid. Als Geschwindigkeitsbegrenzung werden daher zwei Halbellipsoide genutzt (Abb. 6.8(d)), da die maximale Vorwärtsgeschwindigkeit nicht gleich der maximalen Rückwärtsgeschwindigkeit ist.

6.5 Stabilisierung

Mit der beschriebenen Trajektoriengenerierung können die FUmanoid-Roboter bereits laufen. Besonders die Schrittsynchronisierung führt schon zu einer relativ guten Stabilität. In vielen Fällen ist die Stabilität aber noch unzureichend. Deshalb werden zusätzliche Stabilisierungsmethode angewandt, um einen stabileren Gang zu realisieren. Dabei gibt es zwei Gründe für Instabilitäten. Systemimmanente Gründe sind beispielsweise Rutschen auf dem Untergrund, welches je nach Untergrund nicht komplett ausgeschlossen werden kann, aber auch Rauschen bei der Motorkontrolle. Äußere Gründe sind Störungen, die besonders in Spielsituationen auftreten können: Stöße durch andere Roboter oder Kollisionen mit Gegenständen auf dem Spielfeld, zum Beispiel Torpfosten.

Die systemimmanenten Gründe beeinflussen den normalen Gang ohne Einflussnahme anderer Roboter. Sie sind daher besonders störend und sollten möglichst komplett ausgeglichen werden. Äußere Gründe sollten bis zu einem gewissen Grad die Stabilität der Roboter nicht gefährden, können aber nicht zur Gänze abgefangen werden.

6.5.1 Fußstellung

Besonders wichtig für die Stabilität der Roboter ist die Stellung der Füße während des Laufens. Die meisten Passive Dynamic Walker haben ellipsoide Füße. Sie dienen dazu, dass der Roboter gut um sie als unaktuiertes Gelenk schwingen kann. Damit ist es jedoch schwer stabil zu stehen. Da aber beispielsweise für Schüsse oder das autonome Aufstehen ein stabiler Stand wichtig ist, bestehen die Füße der FUmanoid-Roboter aus Auflagepunkten, die in einer Ebene angebracht sind. Durch die Parallelogramm-Mechanik kann der Roboter über das darüber liegende Gelenk schwingen, da der Fuß stets parallel zum Untergrund ist.

Zusätzlich muss eine Laufmaschine auch nach links und rechts schwingen, damit sie nicht zur Seite kippt. Ein typischer Ansatz ist dabei eine sinusförmige Schwingung der Hüfte von links nach rechts [48]. Optimierungen mittels ZMP legen eine solche Schwingung nahe. Eine andere Strategie ist es, die Füße leicht schräg zu stellen. Damit wird bewirkt, dass die Maschine von sich aus leicht zu schwingen beginnt und somit den Schwerpunkt besser über den Füßen hält. Dies wird schon bei Fallis Laufspielzeug genutzt. Er benötigt dieses Verfahren auch, um Abstand zwischen Fuß und Boden zu erhalten, da keine Kniegelenke vorhanden sind. Es führt aber auch zu einer systemimmanenten seitlichen Stabilisierung. Auch der Roboter von Collins et al. nutzt diese Strategie, um eine seitliche Stabilisierung zu erreichen [7].

Dieser Effekt wird für die FUmanoid-Roboter genutzt und die Füße werden leicht v-förmig schräg gestellt. Dadurch ergibt sich die leichte Schwingung. Sinusförmige Hüftbewegungen wurden ebenfalls getestet, wegen schlechter Ergebnisse aber verworfen.

Ferner ist wichtig, wie breit die Beine gestellt werden. Dies wird über den Winkel α zwischen den Beinen eingestellt. Dieser beeinflusst direkt den Winkel β zwischen Fuß und Boden, da die Beine im Stand ein gleichschenkliges Dreieck bilden. Es gilt daher $\beta = \frac{\pi-\alpha}{2}$. Diese Einstellung hat zwei Wirkungen. Zum Einen wird mit größerem α das Support-Polygon im Stand vergrößert. Auch beim Auftritt des Fußes ist das Support-Polygon vergrößert. Dies erhöht die Stabilität.

Zum Anderen verringert sich die Energie des Systems, wie in [32] gezeigt wird. Sind die Füße also zu weit auseinander, kann der Roboter nicht genügend Energie aufbringen, um einen stabilen Gang zu gehen. Sind die Beine zu nah beieinander, ist zu viel Energie im System, so dass der

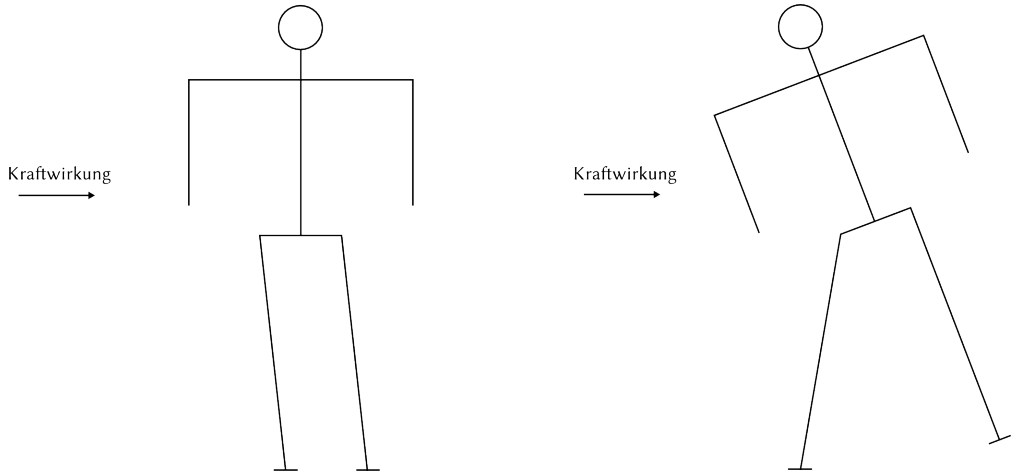


Abb. 6.9: Bei der Knöchelstrategie (links) wird mittels der Knöchelgelenke versucht, auftretenden Kräften entgegen zu wirken. Die Hüftstrategie (rechts) nutzt dafür sowohl Knöchel- als auch Hüftgelenke, um das CoM wieder über das Supportpolygon zu schieben.

Roboter leicht fällt. Ein weiteres Problem bei zu weit gestellten Beinen tritt beim Spielen auf. Der Ball hat eine Breite von 60mm. Sind die Beine deutlich breiter gestellt, rollt der Ball leicht zwischen den Beinen hindurch. Das bedeutet entweder einen Ballverlust oder führt als Halten des Balls zu einer Strafe, da sich der Ball maximal eine Sekunde innerhalb der konvexen Hülle des Roboters befinden darf. Der Arbeitspunkt wird wie von Moballegh et al. erläutert berechnet und so β festgelegt [32].

6.5.2 Entgegengesetzter Armschwung

Durch den Schwung der Beine beginnen die Roboter sich während des Ganges um den Standfuß zu drehen. Es wird also eine Yaw-Drehung in das System gebracht. Grund ist das Drehmoment, das auf das Standbein durch die Energie des nach vorne schwingenden Beines einwirkt. Eine einfache, aber erfolgreiche Strategie ist es, die Arme des Roboters entgegengesetzt zu den Beinen schwingen zu lassen. Sie erzeugen dadurch ein Drehmoment, welches dem durch die Beine induzierten Drehmoment entgegen wirkt. Diese Strategie verfolgten beispielsweise Collins et al. [7] erfolgreich bei ihrem Passive Dynamic Walker. Gegen die unerwünschte Drehbewegung werden auch bei den FUmanoid-Robotern die Arme genutzt und entgegengesetzt zum Bein geschwungen.

6.5.3 Knöchel- und Hüftstrategie

Um externe Störungen zu kompensieren, werden weitere Verfahren benötigt. Am Menschen wurden von Atkenson und Stephens dabei verschiedene Balancestrategien beobachtet. Typische Strategien sind die Knöchel- und die Hüftstrategie [2]. Bei der Knöchelstrategie wird durch das Knöchelgelenk Druck auf den Boden ausgeübt, um Störungen entgegen zu wirken. Bei der Hüftstrategie wird die Hüfte geknickt, um das CoM wieder über die Füße schieben zu können (Abb. 6.9). Diese Strategien wurden auch erfolgreich auf Robotern getestet [34].

Meist werden solche Strategien genutzt, um die Stabilisierung von Robotern in der Sagittalebene zu verbessern. Diese Stabilität ist aber durch die verhältnismäßig langen Füße bei den FUmanoid-Robotern relativ hoch. Stürze erfolgen häufiger zur Seite. Die Stategien wurden daher auf die Coronalebene übertragen.

Um solche Strategien einsetzen zu können, ist es nötig, äußere Störungen wahrzunehmen. Eine einfache Möglichkeit ist es, Beschleunigungssensoren zu nutzen. Der in dem FUmanoid-Roboter verbaute Beschleunigungssensor ist jedoch zu ungenau, um für diesen Zweck sinnvoll eingesetzt zu werden. Stattdessen werden die Drucksensoren in den Füßen genutzt. So können ungewollte Schwankungen nach links bzw. rechts früh bemerkt und Stabilisierungsmethoden gestartet werden. Dafür werden die vier Druckwerte p_{bl}, p_{br}, p_{fl}, p_{fr} des Standfußes gemessen und die Differenz

$$d = (p_{bl} + p_{fl}) - (p_{br} + p_{fr})$$

betrachtet. Diese Differenz wird als Eingabe für einen PD-Regler genutzt. Bis zu einem Schwellenwert θ wird so das $FOOT\_ROLL$ Gelenk bewegt. Dieses übt Druck auf den Boden aus, um den Roboter stabil zu halten. In diesem Fall verhält sich der Roboter ungefähr wie ein invertiertes Pendel. Sind die extern auf ihn einwirkenden Kräfte klein, so reicht das Knöchelgelenk alleine aus, um die Balance zu halten. Wird der Schwellenwert jedoch überschritten, so wird zusätzlich das HIP_ROLL Gelenk bewegt, allerdings in die entgegengesetzte Richtung. Dadurch knickt der Roboter in der Hüfte ein und hebt das Schwungbein. Durch diese Bewegung wird das Schwungbein auch nach außen geführt und hilft so beim Halten der Balance (Abb. 6.9). Dabei ist der Winkel der Hüfte deutlich größer als der Winkel des Knöchels. In mehreren Experimenten wurde ein Verhältnis von 1:4 festgelegt. Durch die Aggressivität des PD-Reglers kann zusätzlich noch eingestellt werden, wie stark die Bewegung ausgeführt werden soll. Ist sie zu stark, greift sie auch schon bei normalen Schwankungen des Gehens und beeinflusst das Laufen. Ist sie zu schwach, werden Stürze nicht mehr effektiv verhindert.

Durch die Kombination aus Ausgleichsbewegung und Schrittsynchronisierung wird auch der Schrittzyklus unterbrochen, so dass ein neuer Schritt erst gestartet wird, wenn der Fuß wieder den Boden berührt. In diesem Zustand ist die Stabilität des Roboters meistens wieder gewährleistet. Die Hüftstrategie führt also nicht implizit zu Störungen.

6.5.4 Verringerung von Schrägstellungen

Durch Fertigungstoleranzen sind die Roboter oft nicht völlig identisch, sondern leicht verschieden. Dadurch kann es passieren, dass Roboter trotz guter Kalibrierung schief stehen, da beispielsweise die Beine einige Millimeter unterschiedlich lang sind. Solche Probleme führen dazu, dass der Schwerpunkt des Roboters nicht genau mittig liegt, sondern leicht nach links oder rechts verschoben ist. Um diese Schrägstellung während des Laufens zu korrigieren, wird ein adaptives System eingesetzt.

Eine Schrägstellung des Roboters führt dazu, dass ein Fuß höher ist. Dieser Fuß trifft später auf den Boden auf. Es kommt also zu verspäteten Fuß-Boden-Kontakten. Diese werden registriert. Die Verspätung wird als Eingabe für einen I-Regler genutzt. Dieser I-Regler schiebt den Oberkörper des Roboters mittels inverser Kinematik in Richtung des Fußes, der zu spät auf den Boden aufgekommen ist. Dies ist annähernd eine Verschiebung des Schwerpunkts des Roboters. Er wird

dadurch nach und nach wieder in die Mitte des Roboters geschoben und die Fuß-Boden-Kontakte treten wieder zur gewünschten Zeit ein.

7 Bewegungen aus dem Laufen

Zusätzlich zum normalen Laufen sollen auch einige spezielle, geregelte Bewegungen aus dem Laufen heraus ausgeführt werden. Besonders wichtig ist dabei ein Schuss, da ein statischer Schuss, wie er in den letzten Jahren zum Einsatz kam, nicht stabilisiert werden konnte. Zusätzlich sollen die Roboter in der Lage sein, größere seitliche Schritte direkt auszuführen, um sich so hinter dem Ball besser positionieren zu können. Diese Schritte sollten parametrisiert sein. Daraus ist leicht ein Interface zu entwickeln, welches echte Schrittplanung zulässt.

7.1 Dynamischer Schuss

Eine wichtige Fähigkeit beim Fußballspielen ist das Schießen. Bei den vorangegangenen Modellen wurde dabei beim Team FUmanoids ein statischer Schuss eingesetzt. Bei diesem muss sich der Roboter zuerst ausrichten. Danach wird der Walker gestoppt, der Roboter geht in seine Grundstellung und führt dann die Schussbewegung aus. Diese ist nicht geregelt. Zwar ist es so möglich, weite Schüsse zu realisieren, diese sind aber nicht dynamisch stabilisiert. Treten also während des Schusses Störungen auf, ist ein Sturz meist unvermeidlich.

Ein anderer Ansatz ist ein parametrisierter, dynamischer Schuss. Dieser wurde beispielsweise von Otte [36], Xu und Mellmann [56] und Müller et al. [33] umgesetzt. Dabei kann je nach System der Schuss in Weite und Richtung [36] parametrisiert werden oder aber auch während des Schusses noch angepasst werden [33, 56]. Solche Online-Anpassungen können insbesondere auch der Stabilisierung eines Schusses dienen. Trotzdem ist es noch nötig den Walker zu beenden, um den Schuss zu starten. Menschliche Fußballer hingegen führen Schüsse fast ausschließlich aus dem Laufen heraus aus. Ein solcher Schuss aus dem Laufen wurde in [36] vorgestellt.

Da ein statischer Schuss mit der gegebenen Hardware nicht hinreichend stabilisierbar ist, wird ein Schuss aus dem Laufen präferiert. Dabei ist die Hauptanforderung nicht, einen besonders starken Schuss zu entwickeln, sondern einen stabilen, möglichst schnell ausführbaren Schuss zur Verfügung zu stellen. Mit einem solchen Schuss ist es möglich den Ball in Bewegung zu halten und so ein schnelles Spiel zu forcieren. Teams, die sich erst am Ball ausrichten müssen, können so Probleme bereitet werden.

Nach mehreren Versuchen, einen Schuss mit dem HIP_PITCH Gelenk zu realisieren, wird nun die stabilere Parallelogramm-Mechanik genutzt. Dafür wird die Schrittweite auf einen Maximalwert erhöht, ohne dass die Filterung eingreift. So werden die Beine schnell nach vorne bewegt und alle Stabilisierungen können unverändert aktiviert bleiben.

Problematisch dabei ist, dass der Roboter sich immer noch sehr genau am Ball positionieren muss. Grund dafür ist, dass auch ein maximaler Schritt nach vorne keine große Reichweite überbrückt. Mit der gegebenen Hardware ist die maximale Schrittweite ca. 10 cm. Auch ist der Schuss relativ schwach, da kein Schwung geholt wird und das Bein nicht hinreichend beschleunigt wer-

den kann. Um dieses Problem zu beheben, werden statt einem Schritt zwei Schritte mit der größeren Schrittweite durchgeführt. Dabei wird zuerst der nicht schießende Fuß nach vorne bewegt. So erreicht man, dass der Schussfuß eine maximale Schwungphase hat. Außerdem tritt der Fuß, der nicht schießt, neben den Ball, so dass die Position, während der Fuß den Ball trifft, nahe am Nulldurchgang ist. Dort ist die höchste Geschwindigkeit zu erwarten, da auch ein Pendel im Nulldurchgang die höchste Geschwindigkeit hat. Hinzu kommt, dass der Fuß den Ball noch eine gewisse Strecke führen kann. Auch das Problem der Positionierung ist kleiner, da der Vorschritt den Roboter bereits näher an den Ball bringt.

Ein weiterer Vorteil dieses Schusses ist, dass der Roboter die ganze Zeit im Laufmodus bleibt und der Walker nicht neu gestartet werden muss. Es werden also wichtige Sekunden gewonnen.

7.2 Positionierung und Schrittplanungs-Interface

Um die Positionierung am Ball schneller gestalten zu können, ist ein Positionierungsschritt sinnvoll. Dabei wird der Roboter nicht über gefilterte und geregelte Eingabewerte langsam dem Ball angenähert, sondern es wird ein größerer platzierter Schritt durchgeführt, wenn der Roboter nahe genug am Ball ist. Dieser ist vom Verhalten gesteuert.

Aufgrund der Parametrisierung der Schritte mit ihrer Schrittweite ist ein solcher Platzierungsschritt einfach umzusetzen. Das Verhalten sendet die entsprechenden Schrittdaten und diese werden unverändert ausgeführt. Um danach tatsächlich die vom Verhalten gewünschten Position zu erreichen, wird die Geschwindigkeit nach dem Schritt auf Null gesetzt. Der Schritt wird dabei, um größtmöglichen Bewegungsfreiraum zu schaffen, immer mit dem Fuß, der dem Ziel näher ist, durchgeführt. So wird auch ein instabiles Überkreuzen der Beine vermieden.

Aus dieser Ansteuerung des Walkers kann ohne großen Aufwand ein Schrittplanungs-Interface gebaut werden. Bei einem solchen System wird die komplette Kontrolle über die Positionierung an eine externe Instanz abgegeben. Der Walker führt in diesem Modus nur noch die Schritte aus, die er als Eingabe bekommt. Die Filterung der Eingabewerte ist in diesem Fall abgeschaltet. Außerdem sollten in diesem Modus nicht konstant Werte an den Walker geschickt werden, sondern explizit Werte für den nächsten Schritt. Jeder beendete Schritt wird vom Walker mittels eines Events an alle angemeldeten Instanzen gemeldet, so dass die Daten für den nächsten Schritt rechtzeitig übergeben werden können.

Mit diesem Modus ist es möglich einen externen Schrittplanungsalgorithmus mit dem Walker zu verwenden, wie zum Beispiel in [47] vorgestellt. Damit sind sehr genaue Annäherungen an den Ball oder an bestimmte Positionen, zum Beispiel den Kreis vor einem Anstoss oder im Tor als Torwart, möglich.

7.3 Orthodribbeln

Um den Ball im Zweikampf von einem anderen Roboter seitlich weg zu bewegen, nutzen die FUmanoid-Roboter das Prinzip des Orthodribbelns. Dabei wird der eine Fuß nach vorne, der andere nach hinten verschoben. Nun kann mit dem vorderen Fuß der Ball seitwärts bewegt werden.

Mit Hilfe der inversen Kinematik ist das Orthodribbeln leicht umzusetzen. Dabei wird während des Laufens ein Offset auf die x-Koordinate beider Füße gesetzt, negativ in Laufrichtung, positiv

in der entgegengesetzten Richtung. Aufgrund dieser Verschiebung steigt jedoch die Entfernung der beiden Beine zueinander, wodurch der Ball leicht zwischen den Beinen hindurch rollen kann. Um dies zu verhindern wird der α-Winkel verkleinert.

Durch diese Anpassung sinkt aber die Stabilität des Roboters. Um dem entgegenzuwirken, wird ein P-Regler genutzt, um den Oberkörper leicht nach vorne zu beugen. Dies verschiebt den Schwerpunkt des Roboters über den vorderen Fuß und hilft so, den Roboter wieder zu stabilisieren. Trotzdem müssen zusätzlich die Höchstgeschwindigkeiten nach unten korrigiert werden, wenn Orthodribbeln aktiv ist.

8 Ergebnisse

Um das allgemeine Laufverhalten und die eingesetzten Methoden zu testen, wurden verschiedene Versuche durchgeführt. Bei diesen wurde Wert darauf gelegt, dass sie wiederholbar sind und so auch mit Ergebnissen anderer Laufmaschinen verglichen werden können. Weiterhin wurde die Laufmaschine mit den beschriebenen Verfahren beim RoboCup 2011 in Istanbul erfolgreich eingesetzt. Die FUmanoids erreichten bei diesem Turnier den vierten Platz von 22 angetretenen internationalen Teams. Sie mussten sich einzig den Teams der Darmstadt Dribblers und CIT Brains geschlagen geben. Gegen den amtierenden Weltmeister DARwIn sind sie nicht angetreten. Besonders der Schuss im Laufen und die dadurch verbundene Umstellung der Strategie führte zu diesem guten Ergebniss. Wie erwartet konnte der kurze Schuss die Taktik vieler Teams durchkreuzen. Auch gegen die Zweit- und Drittplatzierten konnten sich die FUmanoids behaupten, wenn es auch nicht zu einem Sieg reichte. [3]

8.1 Trajektorienüberprüfung mittels Video

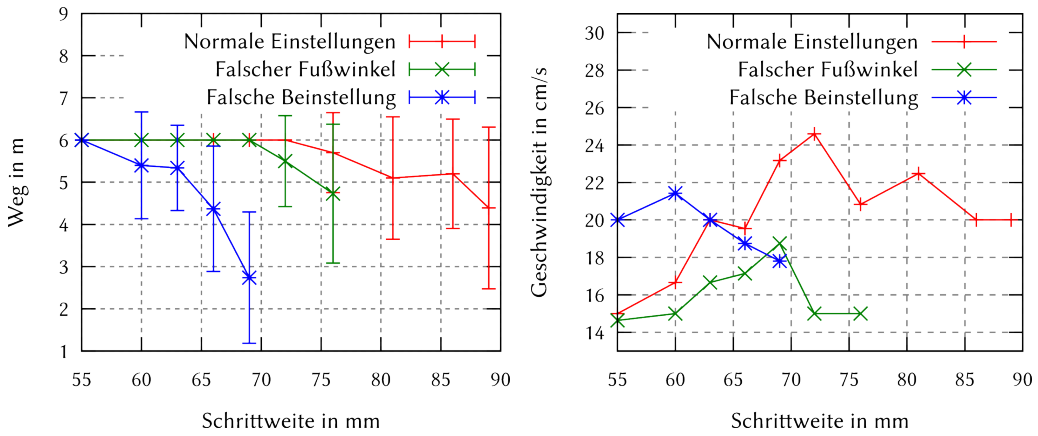
Zunächst wurden die Trajektorien getestet, indem das Laufverhalten und die Bewegungen der Roboter beobachtet wurden. Eine genauere Analyse der Trajektorien wurde mit Hilfe von Videoaufnahmen vorgenommen. So konnten Abweichung der gewünschten und der tatsächlichen Trajektorien in der Verlangsamung gut erkannt werden. Die finalen Trajektorien des Vorwärtsgangs sind in Abb. 8.3 zu sehen. Gut zu erkennen ist die Vorwärtsbewegung des Schwungfußes in Kombination mit der Rückwärtsbewegung des Standfußes. In Abb. 8.4 ist die asymmetrische Ausrichtung des Roboters in der Seitwärtsbewegung zu sehen.

8.2 Lauftests

Der nächste durchgeführte Test war ein allgemeiner Lauftest. Dabei wurde der Roboter in der Mitte des Tores, zum gegnerischen Tor ausgerichtet, gestartet. Er ging gerade vorwärts, ohne Hindernissen oder anderen Störungen ausgesetzt zu sein. Es wurden zwei Werte gemessen. Zum einen die zurückgelegte Strecke, wobei das Maximum hier 6m darstellt. Dies entspricht dem gesamten Feld. Der Normalfall sollte dabei sein, dass der Roboter diese Strecke ohne Sturz und möglichst gerade durchläuft. Bei einem Sturz oder beim seitlichen Verlassen des Feldes aufgrund von Drift, wird die Strecke bis zum Sturz bzw. Austritt aus dem Feld gemessen. Der zweite gemessene Wert ist die Dauer, die der Roboter für diese Strecke benötigt.

Um den Einfluss von der richtigen Fuß- und Beinstellung zu testen, wurden diese jeweils in der Stellung des Originalwalkers und in der Grundstellung getestet. Grundstellung bedeutet dabei für

[3] Videos der Spiele sind auf der Homepage der FUmanoids http://www.fumanoids.de zu finden



(a) Stabilität (b) Geschwindigkeit

Abb. 8.1: Die Stabilität und Geschwindigkeit der FUmanoid Roboter bei unterschiedlichen Schrittweiten

den Fußwinkel parallel zum Untergrund zu sein und für die Beinstellung, dass beide Beine parallel zueinander stehen.

Für jede Kombination aus Einstellungen und Geschwindigkeit wurden zehn Testläufe mit insgesamt 220 Läufe durchgeführt. Die Ergebnisse sind in Abb. 8.1(a) dargestellt. Zu sehen ist die durchschnittlich zurückgelegte Strecke mit jeweils einer Standardabweichung. Ebenfalls zu erkennen ist, dass ein stabiler Gang bei einer Schrittweite von 72 mm noch möglich ist. Bei dieser Schrittweite erreicht der Roboter auch seine Höchstgeschwindigkeit von ca. 25 cm/s. Das ist nah an der errechneten Geschwindigkeit, die bei einer Schrittfrequenz von 4 Hz und einer Schrittweite von 72 mm bei 28.8 cm/s liegt. Die Geschwindigkeiten mit den verschiedenen Einstellungen sind in Abb. 8.1(b) abgebildet.

8.3 Stoßtests

Ein weiterer Test war ein Stoßtest. Dabei wurde besonders die Hüftstrategie zur Stoßstabilisierung untersucht. Für diesen Test wurde ein Pendel von 1.5 m Länge mit einem Pendelgewicht von 250 g gegen den Roboter geschwungen. Um die dabei auftretende Energie zu variieren, wurde die initiale Auslenkung des Pendels zwischen 41° und 90° verändert. Die dabei auftretenden Energien sind in Tab. 8.1 aufgeführt. Für jede Auslenkung wurden zehn Testdurchläufe gemacht, so dass insgesamt 80 Stöße auf den Roboter einwirkten. Dabei wurde die Hüftstabilisierung bei der Hälfte der Stöße deaktiviert und die Ergebnisse verglichen. In Abb. 8.2 ist zu sehen, wie viele Stöße der Roboter ohne zu stürzen überstehen konnte.

Tab. 8.1: Energie und Geschwindigkeit der verschiedenen Stöße gegen den Roboter

Pendelauslenkung	**Stoßenergie**	**Stoßgeschwindigkeit**
41°	1.0 J	10.0 km/h
53°	1.5 J	12.5 km/h
69°	2.5 J	16.0 km/h
90°	3.5 J	19.0 km/h

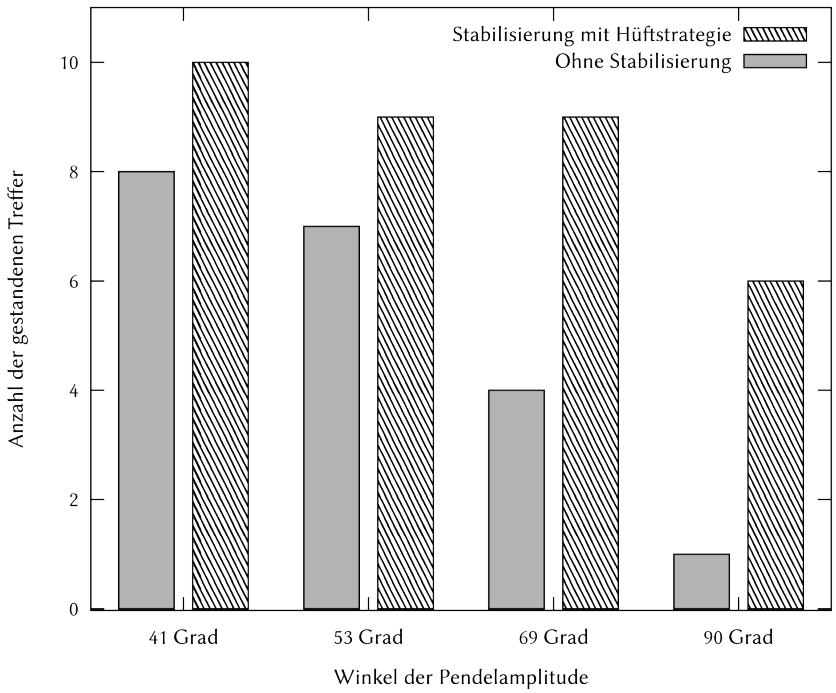


Abb. 8.2: Stabilität der Roboter bei seitlichen Stößen

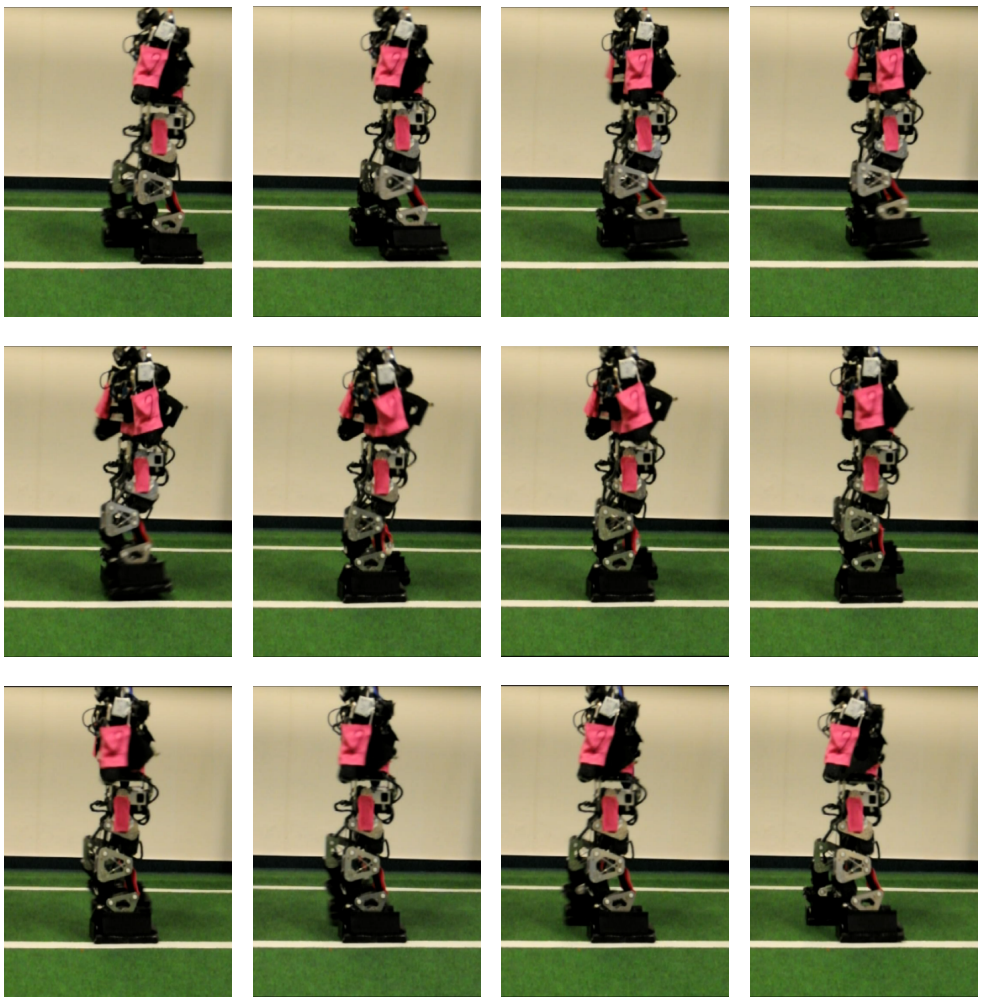


Abb. 8.3: Der Roboter bei einem Doppelschritt vorwärts in der Sagittalebene gesehen. Zwischen je zwei Bildern liegen ca. 40 ms.

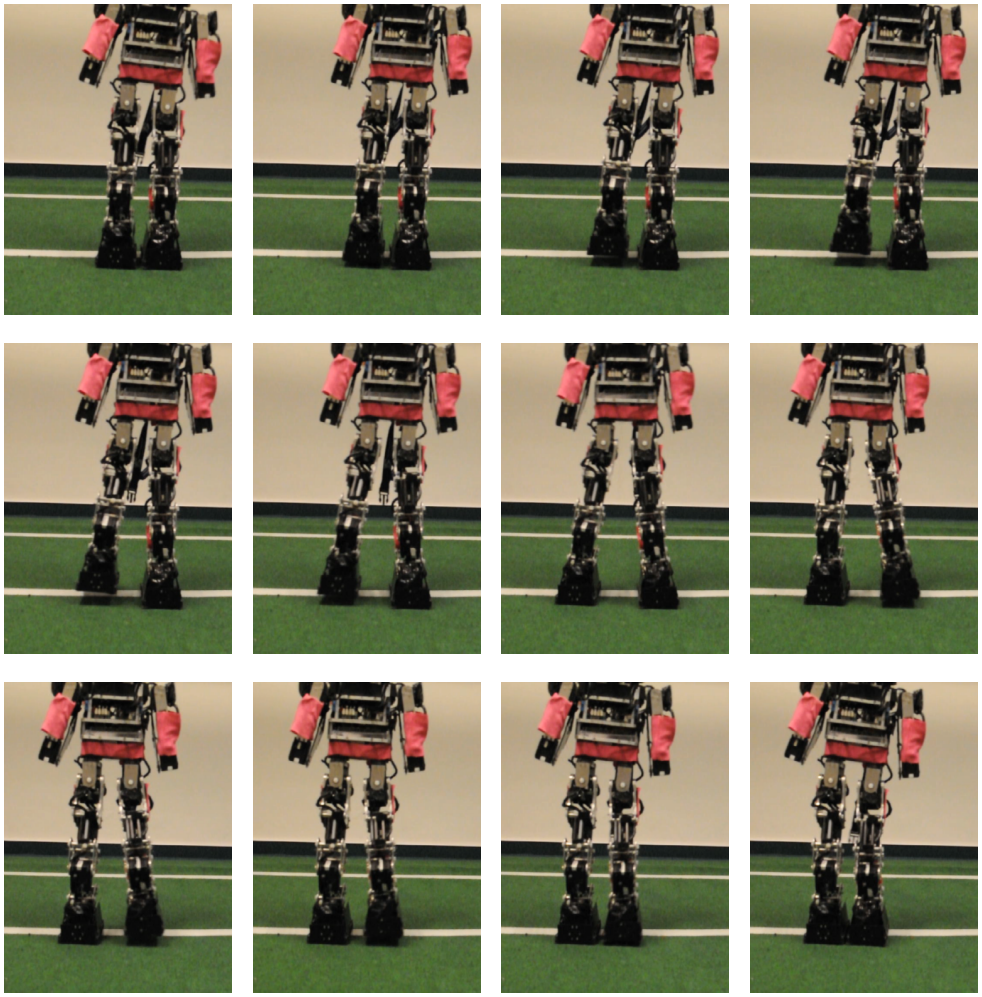


Abb. 8.4: Der Roboter bei einem Seitwärtsschritt. Gut zu sehen ist die asymmetrische Beinstellung in der Mitte des Schritts. Zwischen den Bildern liegen je ca. 40 ms.

9 Diskussion

9.1 Stabilität und Geschwindigkeit

Die wichtigsten Kriterien zur Begutachtung des vorgestellten Laufansatzes sind die Stabilität und die Geschwindigkeit. Sie wurden in den im vorherigen Kapitel beschriebenen Experimenten getestet. In diesen konnte die Stabilität bis zu einer Schrittweite von 72 mm überzeugen. Über 60 m hinweg stürzte der Roboter kein einziges Mal. Dabei konnte er eine Geschwindigkeit von 24.7 cm/s erreichen und liegt damit im oberen Mittelfeld vergleichbarer Roboter (siehe Tab. 9.1).

Interessant ist die Beobachtung, dass der Roboter langsamer wird, wenn die Schrittgröße weiter erhöht wird. Dies liegt zum einen daran, dass die größere Schwungkraft nicht mehr durch den Armgegenschwung kompensiert werden kann. Dadurch entsteht eine Drehung um die Yaw-Achse des Roboters. So benötigt er für die zurückgelegte Strecke mehr Zeit, da sie nun mehr als 6 m beträgt. Ferner entstehen leichter Instabilitäten, die von der Hüftstabilisierung ausgeglichen werden müssen und Zeit kosten. Diese These wird davon unterstützt, dass ab dieser Schrittweite erste Stürze zu verzeichnen sind.

Zum anderen kippt der Roboter nach vorne, wenn der Schwungfuß so weit nach vorne geführt wird, dass das CoM nicht mehr über dem Standfuß liegt. In diesem Fall wird der Bodenkontakt schon früher hergestellt als die eingestellte. Ein solcher Schritt wird durch die Schrittsynchronisierung nicht zu Ende geführt. Eine Geschiwndigkeitssteigerung bleibt aus.

Die richtigen Einstellungen der Bein- und Fußstellung haben einen deutlichen Einfluss auf die Stabilität des Roboters. Besonders falsch zueinander gestellte Beine lassen den Roboter schnell instabil werden. Er kippt dann leicht nach links oder rechts. Mit nah zueinander gestellten Beinen ist dafür die Geschwindigkeit höher. Dies liegt an der höheren Energie, die der Roboter zur Verfügung hat und mit der er besser die Schrittweite erreichen kann. Die Instabilitäten, die sich ergeben, verringern aber die Geschwindigkeit so sehr, dass der Gang mit falschem β-Winkel nicht

Tab. 9.1: Geschwindigkeiten verschiedener Roboter. Daten aus [3, 14, 49]

Team/Roboter	Größe	Gewicht	Geschwindigkeit
Darmstadt Dribblers/DD2008	57.5 cm	3.3 kg	40 cm/s
B-Human/Nao	58 cm	4.3 kg	28 cm/s
Nimbro/Eigenbau	60 cm	2.9 kg	25 cm/s
FUmanoids/Eigenbau	60 cm	4.8 kg	24.7 cm/s
Team DARwIn/DARwIn-OP	45.5 cm	2.8 kg	24 cm/s
NUbots/Nao	58 cm	4.3 kg	14 cm/s
Aldebaran/Nao	58 cm	4.3 kg	10 cm/s

sinnvoll eingesetzt werden kann. Mit weiter auseinander gestellten Füßen ist dagegen die Gefahr, dass der Ball durch die Beine rollt, zu groß.

Der Fußwinkel hat auch auf die Stabilität einen gewissen Einfluss, besonders macht sich die richtige Einstellung hier aber bei der Geschwindigkeit bemerkbar. Durch das periodische Pendeln des Roboters ist der Fußabstand erhöht, so dass die Schrittweite tatsächlich gut erreicht werden kann. Dieser passive Effekt zeigt auch, dass eine gute Grundkonstruktion und -einstellung eines Roboters oft die größeren Effekte erzielt, als durch Aktuatoren erreicht werden können.

Die Stabilisierung mit Hüftstrategie liefert ebenfalls gute Werte. Während der Roboter ohne Stabilisierung nur noch 10 % Stoß mit 3.5 J Energie übersteht, schafft er mit Stabilisierung noch 60 % dieser Stöße. Die Stabilisierung unterbricht auch nicht direkt den Walker, so dass zwar durch die Schrittsynchronisierung der nächste Schritt verzögert wird, der Gang aber danach sofort weitergeführt werden kann. Im Zweikampf konnte der Roboter sich auf diese Weise oft durchsetzen.

Problematisch hingegen stellt sich bei dem Walker der Start dar. Eine Verbesserung wurde zwar durch langsames Anlaufen erreicht. Dabei wird die Höhe des Schritts langsam erhöht. Außerdem kann der Roboter in dieser Phase noch nicht in eine Richtung gehen, bis die Schritte die endgültige Höhe erreicht haben. So soll die Synchronisierung der Schritte mit der Körperdynamik erreicht werden. Dennoch können zu Beginn des Gangs häufig Stürze beobachten werden. Dabei beginnt der Roboter sich aufzuschaukeln und stürzt zuletzt seitwärts. Hier ist eine genauere Einstellung der Schrittsynchronisierung notwendig.

Ein weiteres Stabilitätsproblem ist das komplizierte Zusammenspiel zwischen Verhalten und Walker. Die Filterung ist zwar ein gutes Mittel, die vom Verhaltensmodul erzeugten Wege so anzupassen, dass der Roboter ihnen stabiler folgen kann, trotzdem sind rasche Richtungsänderungen, die zu Instabilitäten führen, noch möglich. Alternativ könnte die Filterung stärker eingestellt werden. Dann entstünden aber deutlich trägere Richtungsänderungen, so dass ein agiles Spiel schwer umzusetzen wäre.

Die Geschwindigkeit hingegen ist mit dem gegebenen Gewicht angemessen. Zwar ist die mögliche Höchstgeschwindigkeit anderer Roboter größer, während des RoboCups war dies jedoch kein entscheidener Grund für eine der beiden Niederlagen. Mit einer leichteren Konstruktion sollte ein schnellerer Gang möglich sein.

9.2 Portierbarkeit

Das Laufsystem wurde ohne größere Änderungen auf eine aktualisierte Version der FUmanoid-Roboter übertragen. Dafür wurde für diese Roboter eine angepasste geometrische Kinematik entwickelt. Da sich die Kinematik aber hauptsächlich in der Länge der Teile und der Drehrichtung der Motoren unterscheidet, waren auch hier keine größeren Änderungen notwendig. Innerhalb eines Tages war es möglich, die Roboter zum Laufen zu bringen. Zwar sind sie in dieser kurzen Zeit noch nicht fähig stabil zu laufen, aber sie können sich ohne Anpassung des Walkers fortbewegen. Auch wurden die Optimierungen nicht für die neuen Gewichtsverteilungen berechnet, so dass gewisse Anpassung noch vorgenommen werden müssen.

Es ist gelungen ein leicht auf neue Hardware zu portierendes System zu entwickeln. Einzig die inverse Kinematik muss neu gelöst werden, da diese hardwarespezifisch ist. Durch Kapselung des entsprechenden Moduls ist der Austausch aber ohne großen Aufwand möglich. Die einzige An-

forderung, die an die Hardware gestellt wird, ist, dass in beiden Füßen Drucksensoren vorhanden sind.

Allerdings basieren viele Annahmen darauf, dass die eingesetzten Roboter mit einer Parallelogramm-Mechanik arbeiten. Die dadurch garantierte Parallelität der Füße müsste also bei der Lösung der inversen Kinematik beachtet werden. Regelungenauigkeit könnten aber trotzdem zu Problemen führen, die bei den FUmanoid-Robotern wegen der Parallelogramm-Mechanik nicht aufgetreten sind.

9.3 Zukünftige Arbeiten

Ein wichtiger Punkt für die bessere Nutzung des Walkers ist die Entwicklung eines Schrittplaners. Durch diesen kann eine deutlich bessere Stabilität erreicht werden. Zur Zeit bekommt der Walker die Richtung, in die er gehen soll, jedoch keine Fußpositionen. Er errechnet aus diesen selbstständig die Position für den nächsten Schritt. Der Walker hat jedoch keinerlei Information darüber, woraus diese Richtungsdaten gewonnen wurden. Daher können spontane, starke Richtungsänderungen und kontiniuierliche Schwankungen nur mittels Filterung geglättet werden. Da hier die meisten Probleme bestehen, sollte die Schnittstelle zwischen Verhalten und Walker zumindest angepasst und weiter verbessert werden. Eine bessere Alternative wäre jedoch, einem externen Modul mit Informationen über Gründe von Richtungsänderungen die Entscheidungsgewalt über die Schrittplanung zu überlassen. So könnten kleine Schwankungen gefiltert, abrupte Richtungsänderung, die begründet sind, jedoch durchgeführt werden. Außerdem könnte eine Annäherung an den Ball sinnvoller durchgeführt werden, so dass ein Schuss schneller möglich ist.

Viele Parameter des Systems, gerade für Stabilisierungsregler, wurden bisher manuell eingestellt. Mit Hilfe von maschinellen Lernverfahren oder anderen Optimierungsverfahren sollte es möglich sein, das entstandene System nochmals hinsichtlich Geschwindigkeit und Stabilität zu verbessern. Experimente von Hemker et al. legen nahe, dass ein signifikanter Geschwindigkeitsgewinn bei gleichbleibender Stabilität möglich ist [16]. Auch könnte die Portierbarkeit erhöht werden, da neue Parametersätze nicht aufwendig von Hand erstellt werden müssten, sondern von einem Lernverfahren schnell auf neue Hardware angepasst werden könnten.

Eine weitere Verbesserung des Systems wäre es, die inverse Kinematik auf das Motorboard auszulagern. So müsste das Programm auf dem Hauptrechner nur Positionen für die Endeffektoren berechnen und die Berechnung der inversen Kinematik könnte entfallen. Dadurch könnte die Taktung des Walkers gegebenenfalls erhöht werden. Die inverse Kinematik ist ohnehin hardwarespezifisch und mit relativ wenig Rechenleistung berechenbar, so dass ein Verbleiben auf der Hauptplatine keinen Vorteil darstellt. Eine komplette Migrierung des Laufsystems auf das Motorboard erscheint nicht sinnvoll, da die Rechenleistung des IGEP oder eines zukünftigen Hauptboards deutlich stärker ist als die des Microcontrollers auf dem Motorboard. Gerade das Fehlen einer Fließkommaeinheit würde die Leistung des Walkers stark mindern.

Mit Hilfe der vorgestellten Trajektorienoptimierung mit ZMP könnten außerdem Bewegungen erstellt werden, die dann halb-dynamisch abgespielt werden. So könnte mit Hilfe der Drucksensoren der ZMP verfolgt und über Regler eingegriffen werden, wenn die tatsächliche ZMP-Trajektorie von der gewünschten abweicht.

9.4 Fazit

In der vorliegenden Arbeit wurde ein Laufsystem für die Roboter des Teams FUmanoid entwickelt. Dieses basiert auf der Analyse von Passive Dynamic Walkern. Die nicht passiv möglichen Bewegungen werden mit Hilfe von ZMP-Optimierung erstellt. Der so enstandene Gang erreicht mit 24.7 cm/s eine gute Geschwindigkeit bei ausreichender Stabilität. Die eingesetzten Methoden sind dabei nicht rechenintensiv und können online parametrisiert und berechnet werden. So ist es mit dem vorgestellten Gang möglich, in jegliche Richtung zu laufen. In den Gang integriert wurde ein kurzer, schneller Schuss, der das Laufen nicht unterbricht und daher häufig und ohne Verzögerung ausgeführt werden kann.

Um den Gang und den Schuss realisieren zu können wurde die inverse Kinematik auf zwei Arten gelöst. Zum einen geometrisch, da diese Art sehr schnell und einfach zu berechnen ist und daher auch online eingesetzt werden kann. Zum anderen wurde eine Lösung auf Basis der Jakobi-Matrix vorgestellt. Dafür wurde eine allgemeine Lösung für parallelmechanische kinematische Ketten entwickelt.

Um Stabilisierungsmethoden integrieren zu können, wurde ein Drucksensormodul entwickelt, welches mit Hilfe von Wägezellen die Druckverteilung im Fuß des Roboters messen kann. Damit wurden verschiedene Stabilisierungsregler integriert.

Insgesamt ist ein lauffähiges System entstanden, welches im direkten Vergleich mit anderen Robotern gut abgeschnitten hat. Beim RoboCup 2011 in Istanbul wurde mit dem System der 4. Platz bei der Weltmeisterschaft der humanoiden KidSize Roboter gewonnen. Dies war auch dem vorgestellten Laufsystem zu verdanken.

Literatur

[1] P. Arena. "The Central Pattern Generator: a paradigm for artifical locomotion". In: *Soft Computing* 4 (2000), S. 251–265.

[2] C. G. Atkeson und B. Stephens. "Multiple Balance Strategies From One Optimization Criterion". In: *Proceedings of the International Conference on Humanoid Robots*. 2007, S. 57–64.

[3] S. Behnke, M. Schreiber, J. Stückler, R. Renner und H. Strasdat. "See, walk, and kick: Humanoid robots start to play soccer". In: *Proceedings of the International Conference on Humanoid Robots*. 2006, S. 497–503.

[4] J. E. A. Bertram und A. Ruina. "Multiple Walking Speed frequency Relations are Predicted by Constrained Optimization". In: *Journal of theoretical Biology* 209 (2001), S. 445–453.

[5] R. Boulic, R. Mas und D. Thalmann. "A Robust Approach for the Center of Mass Position Control with Inverse Kinetics". In: *Journal of Computers and Graphics* 20.5 (1996), S. 443–452.

[6] S. Chernova und M. Veloso. "An evolutionary approach to gait learning for four-legged robots". In: *Proceedings of the International Conference on Intelligent Robots and Systems*. 2004, S. 2562–2567.

[7] S. H. Collins, M. Wisse und A. Ruina. "A Three-Dimensional Passive-Dynamic Walking Robot with Two Legs and Knees". In: *International Journal of Robotics Research* 20.7 (2001), S. 607–615.

[8] A. Crespi und A. J. Ijspeert. "AmphiBot II: An Amphibious Snake Robot that Crawls and Swims using a Central Pattern Generator". In: *Proceedings of the International Conference on Climbing and Walking Robots*. 2006, S. 19–27.

[9] S. Czarnetzki und S. K. und Oliver Urbann. "Applying Dynamic Walking Control for Biped Robots". In: *RoboCup 2009: Robot Soccer World Cup XIII*. Hrsg. von J. Baltes, M. G. Lagoudakis, T. Naruse und S. Shiry. Springer, 2010, S. 58–68.

[10] F. Delcomyn. "Neural basis of rhythmic behavior in animals". In: *Science* 210.4469 (1980), S. 492–498.

[11] J. Denavit und R. S. Hartenberg. "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices". In: *Journal of Applied Mechanics* 22 (1955), S. 215–221.

[12] G. Fallis. *Walking toy*. US Patent No. 376,588. 1888.

[13] M. Garcia, A. Chatterjee, A. Ruina und M. Coleman. "Simplest Walking Model: Stability, Complexity, and Scaling". In: *Journal of Biomechanical Engineering* 120 (1998), S. 281–288.

[14] C. Graf und T. Röfer. “A Closed-loop 3D-LIPM Gait for the RoboCup Standard Platform League Humanoid”. In: *Proceedings of the Workshop on Humanoid Soccer Robots at the International Conference on Humanoid Robots*. 2010.

[15] S. Grillner und P. Wallén. “Central Pattern Generators for Locomotion, with Special Reference to Vertebrates”. In: *Annual Review of Neuroscience* 8 (1985), S. 233–261.

[16] T. Hemker, H. Sakamoto, M. Stelzer und O. von Stryk. “Efficient walking speed optimization of a humanoid robot”. In: *International Journal of Robotics Research* 28.2 (2009), S. 303–314.

[17] E. Hering, R. Martin und M. Stohrer. *Physik für Ingenieure*. Springer, 2007.

[18] K. Hirai, M. Hirose, Y. Haikawa und T. Takenaka. “The development of Honda humanoid robot”. In: *Proceedings of the International Conference on Robotics and Automation*. Bd. 2. 1998, S. 1321–1326.

[19] G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto und O. Hanagata. “Autonomous evolution of gaits with the sony quadruped robot”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. 1999, S. 1297–1304.

[20] D. Juricić und M. Vukobratović. “Mathematical Modeling of Biped Walking Systems”. In: *ASME Publication 72-WA/BHF-13* (1972).

[21] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara und K. H. K. Yokoi. “Biped walking pattern generation by using preview control of zero-moment point”. In: *Proceedings of the International Conference on Robotics and Automation*. 2003, S. 1620–1626.

[22] D. Kar. “Design of Statically Stable Walking Robot: A Review”. In: *Journal of Robotic Systems* 20 (2003), S. 671–686.

[23] T. Katayama, T. Ohki, T. Inoue und T. Kato. “Design of an optimal controller for a discrete-time system subject to previewable demand”. In: *International Journal of Control* 41.3 (1985), S. 677–699.

[24] N. Kohl und P. Stone. “Policy gradient reinforcement learning for fast quadrupedal locomotion”. In: *Proceedings of the International Conference on Robotics and Automation*. 2004, S. 2619–2624.

[25] J. Lunze. *Regelungstechnik 1*. Springer, 2010.

[26] A. K. Mackworth. “On Seeing Robots”. In: *Computer Vision: Systems, Theory and Applications*. Hrsg. von A. Basu und X. Li. Singapore: World Scientific Press, 1993, S. 1–13.

[27] T. Matsubara, J. Morimoto, J. Nakanishi, M.-a. Sato und K. Doya. “Learning CPG-based biped locomotion with a policy gradient method”. In: *Journal on Robotics and Autonomous Systems* 54 (2006), S. 911–920.

[28] T. McGeer. “Passive Dynamic Walking”. In: *International Journal of Robotics Research* 9.2 (1990), S. 62–82.

[29] T. McGeer. “Passive Walking with Knees”. In: *Proceedings of IEEE Robotics and Automation Conference*. 1990, S. 1640–1645.

[30] T. McGeer. *Stability and control of two-dimensional biped walking*. Techn. Ber. CSS-IS TR 88-01. Simon Fraser University, Centre for System Science, 1988.

[31] H. Moballegh. “Development of an Autonomous Humanoid Robot Team”. Diss. Freie Universität Berlin, 2011 (zu erscheinen).

[32] H. Moballegh, M. Mohajer und R. Rojas. “Increasing foot clearance in biped walking: Independence of body vibration amplitude from foot clearance”. In: *RoboCup 2008: Robot Soccer World Cup XII*. Hrsg. von A. W. C. Z. L. Iocchi H. Matsubara. 2008, S. 157–165.

[33] J. Müller, T. Laue und T. Röfer. “Kicking a Ball – Modeling Complex Dynamic Motions for Humanoid Robots”. In: *RoboCup 2010: Robot Soccer World Cup XIV*. Hrsg. von J. R. del Solar, E. Chown und P. Ploeger. 2010, S. 109–120.

[34] D. N. Nenchev und A. Nishio. “Experimental Validation of Ankle and Hip Strategies for Balance Recovery with a Biped Subjected to an Impact”. In: *Proceedings of the International Conference on Intelligent Robots and Systems*. 2007, S. 4035–4040.

[35] D. E. Orin und W. W. Schrader. “Efficient Computation of the Jacobian for Robot Manipulators”. In: *International Journal of Robotics Research* 3.4 (1984), S. 66–75.

[36] S. Otte. *Entwicklung eines dynamischen Schusses für humanoide Fußballroboter*. Bachelorarbeit, Freie Universität Berlin. 2010.

[37] D. Pieper. “The Kinematics of Manipulators Under Computer Control”. Diss. Stanford University., 1968.

[38] L. Righetti und A. J. Ijspeert. “Programmable Central Pattern Generators: an application to biped locomotion control”. In: *Proceedings of IEEE International Conference on Robotics and Automation*. 2006, S. 1585–1590.

[39] RoboCup Humanoid League Technical Committee. *RoboCup Soccer Humanoid League Rules 2011*. 2011.

[40] Robotis Co. Ltd. *User’s Manual Dynamixel RX-28*. v1.10.

[41] Robotis Co. Ltd. *User’s Manual Dynamixel RX-64*. v1.10.

[42] A. Ruina. *Cornell Ranger 2011*. 2011. URL: `http://ruina.tam.cornell.edu/research/topics/locomotion\_and\_robotics/ranger/Ranger2011/index.html`.

[43] T. Röfer. “Evolutionary Gait-Optimization Using a Fitness Function based on Proprioception”. In: *RoboCup 2004: Robot Soccer World Cup VIII*. Hrsg. von D. Nardi, M. Riedmiller, C. Sammut und J. Santos-Victor. 2005, S. 310–322.

[44] M. Saggar, T. D’Silva, N. Kohl und P. Stone. “Autonomous Lerning of Stable Quadruped Locomotion”. In: *RoboCup 2006: Robot Soccer World Cup X*. Hrsg. von G. Lakemeyer, E. Sklar, D. Sorenti und T. Takahashi. 2006, S. 98–109.

[45] P. Sardain und G. Bessonnet. “Forces acting on a biped robot. Center of pressure-zero moment point”. In: *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEETransactions on* 34.5 (2004), S. 630 –637.

[46] A. Schmitz, M. Missura und S. Behnke. “Learning Footstep Prediction from Motion Capture”. In: *RoboCup 2010: Robot Soccer World Cup XIV*. Hrsg. von J. R. del Solar, E. Chown und P. Ploeger. 2010, S. 97–108.

[47] A. Schmitz, M. Missura und S. Behnke. “Real-Time Trajectory Generation by Offline Footstep Planning for a Humanoid Soccer Robot”. In: *RoboCup 2011: Robot Soccer World Cup XV*. Hrsg. von T. Röfer, N. M. Mayer, J. Savage und U. Saranlı. 2011.

[48] D. Scholz. *Modulare und plattformunabhängige dynamische Bewegungserzeugung für autonome mobile humanoide Roboter*. Diplomarbeit, Technische Universität Darmstadt. 2008.

[49] D. Scholz, M. Friedmann und O. von Stryk. “Fast, Robust and Versatile Humanoid Robot Locomotion with Minimal Sensor Input”. In: *Proceedings of the Workshop on Humanoid Soccer Robots at the International Conference on Humanoid Robots*. 2010.

[50] E. Schrüfer. *Elektrische Messtechnik: Messung elektrischer und nichtelektrischer Größen*. Hanser Verlag, 2007, S. 215–216.

[51] G. Schulz. *Regelungstechnik 1*. Oldenbourg, 2004.

[52] G. Taga, Y. Yamaguchi und H. Shimizu. “Self-organized control of bipedal locomotion by neural oscillators in unpredictable enivronment”. In: *Biological Cybernetics* 65 (1991), S. 147–159.

[53] M. Vukobratovic und B. Borovac. “Zero-Moment Point - Thirty Five Years of its Life.” In: *International Journal of Humanoid Robotics* (2004), S. 157–173.

[54] K. Waldron und O. Khatib. “Kinematics”. In: *Handbook of Robotics*. Hrsg. von B. Siciliano und O. Khatib. Springer, 2008, S. 9–31.

[55] M. Weck und D. Staimer. “Parallel Kinematic Machine Tools – Current State and Future Potentials”. In: *CIRP Annals - Manufacturing Technology* 51.2 (2002), S. 671–683.

[56] Y. Xu und H. Mellmann. “Adaptive Motion Control: Dynamic Kick for a Humanoid Robot,” in: *KI 2010: Advances in Artificial Intelligence*. Springer, 2010, S. 392–399.

[57] J.-I. Yamaguchi, A. Takanishi und I. Kato. “Development of a bipedwalking robot compensating for three-axis moment by trunk motion”. In: *Proceedings of the International Conference on Intelligent Robots and Systems*. Bd. 1. 1993, S. 561 –566.

Abbildungsverzeichnis