Laufen humanoider Roboter auf regelungstechnischer Basis mit Echtzeitmodifikation der Fußpositionen

Dissertation

zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von
Oliver Urbann

Dortmund
2017

Tag der mündlichen Prüfung:
30.1.2017

Dekan:
Prof. Dr.-Ing. Gernot A. Fink

Gutachter:
Prof. Dr.-Ing. Uwe Schwiegelshohn, TU Dortmund
Prof. Dr. Ubbo Visser, University of Miami

Inhaltsverzeichnis

KAPITEL 1

Einleitung

Die Industrialisierung war ein wesentlicher Meilenstein in der Geschichte der wirtschaftlichen Entwicklung. Sie wurde wesentlich vorangetrieben von der Erfindung verschiedener Maschinen, die dem Menschen simple aber körperlich schwere Arbeit abnehmen können, beispielsweise der Dampfmaschine. Technologischer Fortschritt ist so bis heute ein wesentlicher Faktor für Wirtschaftswachstum, wenn auch ein umstrittener, da Maschinen und Computer für Menschen häufig den Arbeitsplatzverlust bedeuten. Schon Anfang des 20. Jahrhunderts gab es daher die Vorstellung von Maschinen, die dem Menschen nicht nur Arbeit abnehmen, sondern ihn vollständig ersetzen können. Der Schriftsteller Karel Čapek zeichnete 1921 die Vision von derartigen Maschinen, die er (im tschechischen Original) „robota", zu Deutsch „Frondienst" oder „Zwangsarbeit" nannte. In seinem Drama, welches in dreißig Sprachen übersetzt wurde, sind die Roboter dem Menschen sehr ähnlich. Heute jedoch steht der Begriff vielmehr für eine Kategorie von Maschinen, die vielfach Anwendung in der Industrie finden, im Allgemeinen aber nicht mit einem Menschen vergleichbar sind. Dennoch verbleibt die Vision, dass menschenähnliche Roboter, auch „humanoide Roboter" genannt, eines Tages mindestens alle Tätigkeiten verrichten können, zu denen auch ein Mensch fähig ist.

Die menschliche Form hat für Roboter wesentliche Vorteile. Gebäude, Räume, Maschinen usw. sind für den Menschen gebaut und daher auf ihn angepasst. Das betrifft vor allem die Größe: Türen, Räume und Möbel sind entsprechend dimensioniert. Auch Gegenstände sind passend konstruiert, beispielsweise für die Benutzung mit der menschlichen Hand. Ein weiterer Punkt ist vor allem für gehbehinderte Menschen ein Grund zur Ärgernis: Stufen oder andere Hindernisse am Boden. Wenn auch im öffentlichen Raum immer seltener anzufinden, kann insbesondere im industriellen und privaten Bereich nicht ausgeschlossen werden, dass die Fortbewegung auf Rädern schwierig bis unmöglich ist. Zudem ist es auch der Mensch,

der sich im Laufe der Evolution der Natur angepasst hat, man betrachte beispielsweise die Jagd im Wald.
Die Vorstellung von menschenähnlichen Maschinen bekommt somit eine objektive Begründung. Für Menschen geschaffene Umgebungen müssen nicht an Maschinen angepasst werden, was nicht nur finanziell aufwändig bis unmöglich wäre, sondern möglicherweise auch praktische und ästhetische Nachteile mit sich bringen würde.

1.1 Zielsetzung

Diese Arbeit beschäftigt sich mit einem wesentlichen Aspekt von humanoiden Robotern, der Fortbewegung auf zwei Beinen. Nach wie vor existieren keine Roboter bzw. Algorithmen, die einen derart stabilen Lauf ermöglichen würden, so dass ein Produktiveinsatz denkbar wäre. Wie auch in Kapitel 2, „Stand der Forschung“ deutlich wird, ist der Grund dafür nicht das mangelnde Forschungsinteresse. Es existiert eine Vielzahl an humanoiden Robotern in verschiedenen Größen und eine große Zahl an Algorithmen, auf die in Kapitel 2 näher eingegangen wird. Allerdings sind im Vergleich zu industriell genutzten Roboterarmen die Anforderungen an laufende Roboter höher. Die Beine müssen gleichzeitig leicht und stabil genug sein, um das gesamte Gewicht des Roboters tragen zu können. Sie müssen, während sie keinen Kontakt zum Boden haben, genau so gut geregelt werden wie in dem Fall, wo sie das gesamte Gewicht des Roboters tragen. Kleine Abweichungen können zu verschiedenen Fehlern führen, die entweder von vornherein vermieden oder nach dem Eintreten ausbalanciert werden müssen.
Ebenso können größere Störungen auftreten, beispielsweise verursacht durch Stöße anderer Roboter oder Menschen. Während es für kleinere Störungen möglicherweise ausreicht, die Oberkörper- oder Armbewegung anzupassen, ist es notwendig bei größeren Störungen die gewünschten Fußpositionen anzupassen, was beim Menschen als Ausfallschritt bezeichnet wird.
Ziel dieser Arbeit ist daher die Entwicklung eines Laufalgorithmus für reale Roboter, der

1. von Grund auf speziell auf die existierenden Schwächen von realen Robotern eingeht und

2. in realen und dynamischen (sich verändernden) Umgebungen einsetzbar ist.

Reale Roboter und Umgebungen weichen im Allgemeinen vom Modell ab und führen daher zu unvorhergesehenen Störungen. Zudem sind in realen, dynamischen Umgebungen Kollisionen mit Robotern, Gegenständen oder Menschen kaum zu vermeiden. Weitere wichtige Ziele sind daher, dass der Lauf

3. kleinere Störungen ausbalancieren kann und

4. zudem mit größeren Störungen umgehen kann, für die Ausfallschritte notwendig sind.

Die Sinnhaftigkeit des dritten Ziels wird im nächsten Abschnitt näher begründet. Dort werden die drei Punkte zudem zu Anforderungen an den Laufalgorithmus führen, welche sich auch in der Gliederung dieser Arbeit widerspiegeln, die im darauf folgenden Abschnitt dargestellt wird.

1.2 Anforderungen

Aus den allgemeinen Zielen dieser Arbeit folgen einige Forderungen, die an den Laufalgorithmus zu stellen sind. Diese sind als Eckpunkte für neue Entwicklungen wesentlich, müssen zudem jedoch auch bei der Auswahl von bestehenden Konzepten und Verfahren beachtet werden.
Aus dem Ziel, dass der Laufalgorithmus auf die Schwächen realer Roboter eingehen muss, folgen einige Anforderungen, die zunächst auch ohne Betrachtung einer Sensorkontrolle gelten müssen:

Echtzeitfähigkeit ist eine der wichtigsten Anforderungen. Echtzeitfähigkeit bedeutet, dass der Algorithmus in jedem Fall ein Ergebnis liefern muss, bevor eine bestimmte Zeit abgelaufen ist. In dieser Zeitspanne wird aus den Sensordaten, welche mit ausreichend hoher Frequenz vom Roboter zur Verfügung gestellt werden, eine Konfiguration berechnet, die beispielsweise als Gelenkwinkel an die Servos des Roboters weitergeleitet werden. Während des nächsten Rechenschrittes (von Sensordaten bis Gelenkwinkel) stellt diese Konfiguration die Sollwinkel für die Gelenke dar, welche daher rechtzeitig an die Servos weitergeleitet worden sein müssen, da zunächst undefiniert ist, wie sich die Servos für den Fall verhalten, sollten die Winkel nicht rechtzeitig weitergeleitet worden sein. Das kann dazu führen, dass die Gelenke kein Drehmoment mehr ausüben und der Roboter fällt. Es handelt sich somit um eine harte Echtzeitbedingung. Üblicherweise werden Laufalgorithmen mit einer Frequenz von mindestens 100 Hz [25, 28] betrieben, was bedeutet, dass der Algorithmus, im Falle von genau 100 Hz, nicht mehr als 10 ms benötigen darf. Diese Zeit steht allerdings nicht vollständig nur für die Berechnung des Laufs zur Verfügung. In der Praxis werden weitere Algorithmen parallel zum Lauf ausgeführt, beispielsweise die Verarbeitung von Kamerabildern oder anderen Sensordaten, die von der Pfadplanung verwendet werden. Es ist daher wünschenswert, möglichst weit unter der oberen Schranke zu liegen.

Onlinefähigkeit gibt an, dass der Algorithmus den Lauf nicht vollständig vorherplant, sondern ihn während der Ausführung anhand der aktuellen Geschwindigkeit oder Richtung bestimmt. Diese externe Eingabe kann beispielsweise von Verhaltensalgorithmen während des Laufs bestimmt und verändert werden und steht somit nicht vollständig vor

Beginn des Laufs zur Verfügung, weshalb man hier beim Lauf von einem „Online"-Algorithmus spricht. Das ist insbesondere in dynamischen Umgebungen wichtig, um beispielsweise Hindernissen auszuweichen oder die Zielposition anpassen zu können.

Stabilität ist eine ebenso grundlegende Anforderung. Es gibt verschiedene Definitionen dieses Begriffs, intuitiv steht er synonym für Vermeidung von Stürzen. Da dies jedoch in den oben aufgeführten Situationen nicht immer vermeidbar ist, wird hier nur eine Verbesserung der Stabilität gefordert. Eine sinnvolle Möglichkeit, einen Roboter als stabiler zu bezeichnen ist, wenn der Oberkörper eine geringere Abweichung von der Sollorientierung zeigt, da sich oft Stürze zuvor durch eine Schwankung ankündigen.

Winkelbasierte Kontrolle bedeutet, dass die Gelenke des Roboters als Eingabe Gelenkwinkelpositionen erwarten. Das gilt für einen großen Teil der aktuell im Handel befindlichen Roboter, weshalb diese Arbeit sich auf die winkelbasierte Kontrolle konzentriert. Für hochpreisige, derzeit noch nicht käuflich erwerbbare Roboter, deren Gelenke Drehmomente verarbeiten können [38, 2], ließen sich die Algorithmen dieser Arbeit um eine inverse Dynamik erweitern, so dass aus den Winkelpositionen Drehmomente berechnet werden können [43, Kapitel 2.5.1].

Aus diesen Kriterien folgt aber noch kein Lauf, bei dem man von einer Fortbewegung sprechen kann. Es wird daher eine Referenz benötigt, um festlegen zu können, in welche Richtung oder mit welcher Geschwindigkeit sich der Roboter bewegt. Naheliegend ist die direkte Angabe der Geschwindigkeit $\vec{v} = (v_x, v_y, \dot{\varphi})$, möglicherweise aber auch die Angabe der gewünschten Fußpositionen am Boden, für den Fall, dass sich beispielsweise Hindernisse am Boden befinden, auf die er nicht treten soll, oder der Roboter exakt eine bestimmte Position vor einem Gegenstand erreichen muss. Da es sich hier um eine externe Eingabe handelt, welche sich während des Laufs ändern kann, aber nicht vorher vollständig bekannt ist, spricht man, wie oben bereits erwähnt, von einem Online-Algorithmus.
Die Wahl dieser Referenz muss vor allem das obige Stabilitätskriterium erfüllen. Da dies für einen realen Roboter gelten muss, sind Ungenauigkeiten des Roboters mit einzukalkulieren, die vom Modell nicht abgedeckt werden. Während der physische Roboter aufgrund seiner Beinlänge und des maximalen Drehmoments der Motoren bereits eine maximale Geschwindigkeit vorgibt, ist es sinnvoll einen kleineren Wert für die maximale Geschwindigkeit auszuwählen, um beispielsweise den Verschleiß und die Hitzeentwicklung zu begrenzen. Aber auch zur Steigerung der Stabilität ist es sinnvoll, da größere Schrittweiten auch mehr Instabilitäten verursachen. Nach obiger Definition von Stabilität (geringe Schwankung) ist es zudem sinnvoll den Roboter möglichst gering zu beschleunigen, sofern es nicht notwendig ist, da wechselnde Beschleunigungen zu einer verstärkten Oszillation führen können. Folgende Anforderung ist daher ebenfalls zu stellen:

Freie Referenzwahl mit manuell festlegbaren Eigenschaften, die nicht durch die technischen oder mathematischen Eigenschaften der Teilalgorithmen beschränkt werden, und damit primär das Stabilitätskriterium erfüllen.

Auf diese Weise lässt sich der Lauf bereits durch die Wahl der Referenz so beeinflussen, dass der Oberkörper möglichst wenig beschleunigt wird, um so Schwankungen zu vermeiden und damit den Lauf zu stabilisieren. Weitere mögliche Definitionen des Begriffs Stabilität werden auch in Kapitel 2, „Stand der Forschung“ eine wesentliche Rolle spielen und dort um mathematische Definitionen erweitert.
Die Ziele dieser Arbeit umfassen auch die Reaktion auf unvorhersehbare aber messbare Instabilitäten. Eine typische beobachtbare Variante ist die oben erwähnte Schwankung des Oberkörpers, die sich oft nicht vollständig vermeiden lässt, sich aber derart verstärken kann, dass der Roboter stürzt. Dieses Beispiel fällt in die Kategorie der leichteren Störungen, die nicht umgehend zu einem Fall führen. Ein sofortiger Sturz liegt dann vor, wenn eine Störung, beispielsweise ein Stoß von anderen Robotern oder Menschen, eine Rotation des Oberkörpers um die x- oder y-Achse hervorruft, die nicht ihre Richtung ändert bevor der Roboter gestürzt ist.
Üblicherweise gibt eine vorgegebene Referenz auch den nächsten Schritt vor, da beispielsweise eine gewünschte Geschwindigkeit bei gegebener Schrittdauer zu einer bestimmten Schrittweite führt. Je nach verwendetem Algorithmus kann es notwendig sein, mehr Schritte als nur den nächsten zu planen, was in Unterabschnitt 2.2.1, „Grundlagen zur Bewegungssteuerung“ näher erläutert wird. Zur Stabilisierung von drohenden sofortigen Stürzen ist es unter Umständen notwendig, die vorgeplanten Positionen zu modifizieren, was jedoch nur unter bestimmten Bedingungen möglich ist:

Bedingte Ausfallschritte bedeuten anders formuliert, dass die vorgeplanten Fußpositionen modifiziert werden, um den Roboter zu stabilisieren. Eine Umplanung ist aber zu vermeiden, wenn, wie oben bereits erwähnt, nur bestimmte Fußpositionen möglich sind. Es ist aber auch möglich, dass Modifikationen nur zu bestimmten Zeitpunkten nicht durchführbar sind. Ist eine Modifikation nicht möglich, ist es sinnvoll, den Roboter anderweitig zu stabilisieren. Das begründet das dritte Ziel dieser Arbeit, kleinere Störungen zu balancieren ohne die Schritte zu modifizieren.

Definierbare Startzeitpunkte für die Ausfallschritte sind notwendig, da nicht zu jedem Zeitpunkt jede Form der Modifikation anwendbar ist, möglicherweise jedoch später. Beispielsweise können Roboter aus mechanischen Gründen die Beine kaum kreuzen. Außerdem definiert die Beinlänge einen Radius um den Roboter, bei dem der Fuß noch den Boden berühren kann, so dass der Ausfallschritt möglicherweise später eingeplant werden muss, um diese Bedingung nicht zu verletzen.

Die Arbeit wird sich eng an den Zielen und Anforderungen orientieren, was bei der Diskussion des Standes der Forschung und der Auswahl geeigneter existierender Methoden in Kapitel 2 eine große Rolle spielen wird. Die Ziele sind ebenfalls in der Gliederung der Arbeit erkennbar, aufgeführt im nächsten Abschnitt.

1.3 Überblick

Abbildung 1.1 zeigt einen Überblick über den Aufbau und den logischen Zusammenhang der Arbeit. Das nächste Kapitel vergleicht anhand der gestellten Ziele und Anforderungen verschiedene Robotermodelle und darauf aufbauende Algorithmen. Als besonders geeignet zeigt sich das „3D Linear Inverted Pendulum Mode“, welches in diesem Kapitel näher vorgestellt wird. Darauf aufbauend sind die Regler „Model Predictive Control“ (MPC) bzw. „Preview Control“ (PC) die erfolgversprechendsten Ansätze, welche in Kapitel 3 näher erläutert werden, wobei die hier vorgestellten Varianten Anpassungen für die Zwecke dieser Arbeit enthalten. Anhang A zeigt zudem eine Herleitung der MPC-Zielfunktion, welche bisher nicht in der Literatur zu finden ist, und Anhang B, wie die dazu notwendigen Matrizen hergeleitet werden können. Wie in der Abbildung zu sehen, orientieren sich die folgenden Kapitel an den beiden Reglern und behandeln beide in den folgenden Kapiteln parallel.

Kapitel 4 geht auf die Rückführung von Sensordaten ein, wobei zunächst die Eignung verschiedener Sensoren für verschiedene Störungen und dessen Behandlungen besprochen werden. Daraufhin wird ein Beobachter entworfen, der die Sensordaten für die Regler nutzbar macht. Im Falle des PC wird im Anschluss die Möglichkeit von Ausfallschritten, oder anders ausgedrückt, die Modifikation von Schritten, hinzugefügt, wobei diese Möglichkeit in MPC bereits enthalten ist.

Die Regler mit der Verarbeitung von Sensordaten stellen nur einen Teil eines vollständigen Laufalgorithmus dar und müssen um einige notwendige Module erweitert werden, beispielsweise der Schrittplanung und der inversen Kinematik. Kapitel 5 geht auf diese Dinge insbesondere für PC ein, wobei die Module für MPC prinzipiell nahezu gleich anwendbar wären. Da jedoch der MPC-Ansatz mächtiger ist, würde dieser nur einen Teil der Module benötigen. Die beiden Verfahren werden anschließend in Kapitel 6 miteinander in verschiedenen Experimenten verglichen. Es beginnt mit dem Vergleich der Laufzeit, welche bei MPC unter den gegebenen Voraussetzungen zu hoch ist. Neben anderen Nachteilen führt das letztendlich zu dem Schluss, dass MPC auf vielen existierenden Robotern nicht echtzeitfähig ist, weswegen die Evaluation nur für den PC-Ansatz fortgesetzt wird. Abschließend wird ein Fazit gezogen und diskutiert, inwiefern die Ergebnisse zu künftigen Projekten führen können.

1.4 Beitrag

Diese Arbeit konzentriert sich auf die Stabilisierung mit Hilfe von Sensordaten. Als Grundlage dienen die Verfahren von Kajita et al. [24, 25] und Diedam et al. [11], wobei ersteres leicht umformuliert angewendet wird und sich an der ursprünglichen Variante von Katayama et al. [26] orientiert. Alle dargestellten Herleitungen zu diesen Verfahren sind zuvor nicht von anderen Autoren veröffentlicht worden, teilweise auch nicht die Ergebnisse und werden daher in dieser Arbeit hergeleitet. Hier nicht gezeigte Herleitungen sind in der jeweils angegebenen Literatur zu finden.

Die darauf aufbauenden Verfahren zur Sensorkontrolle, nämlich der Beobachter und die Ausfallschritte, sind der zentrale Beitrag dieser Arbeit. Insbesondere sind alle hier gezeigten Herleitungen für diese Arbeit erstellt worden, wobei die grundsätzlichen Formen eines Zustandsraummodells und eines Beobachters allgemein in der Regelungstechnik bekannt sind.

Während in der Forschung unter Laborbedingungen die Stabilität des Verfahrens von Kajita et al. [24, 25] bereits 2003 und 2006 gezeigt wurde, zeigt diese Arbeit die Anwendbarkeit in einem Umfeld mit deutlich höheren Anforderungen. Ein Beitrag sind daher zudem alle Algorithmen in Kapitel 5, welche mit dem Ziel entwickelt wurden einen Lauf zu realisieren, der auch außerhalb einer fest definierten Laborumgebung die gestellten Ziele erreicht.

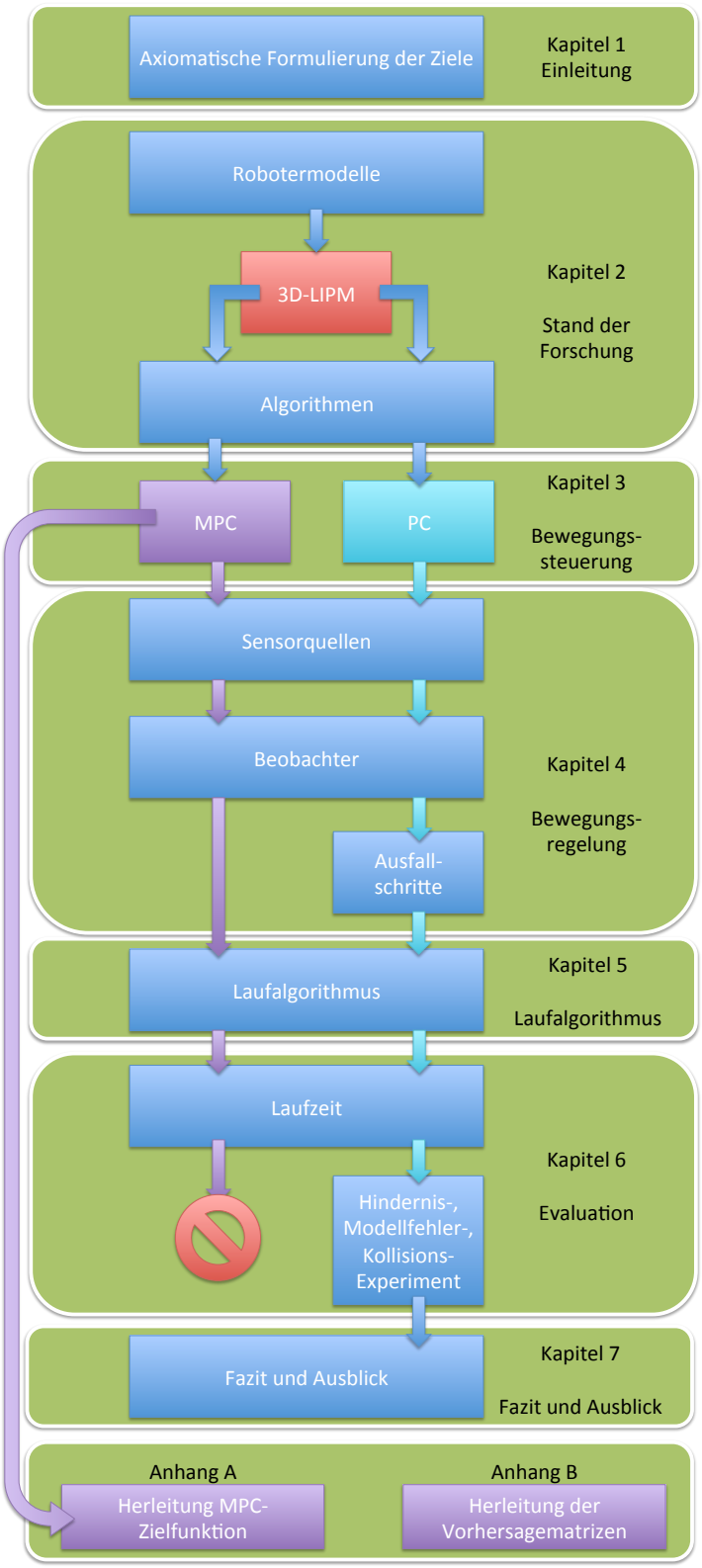

Abbildung 1.1: Überblick über den Aufbau der Arbeit.

KAPITEL 2

Stand der Forschung

Zu Beginn der Entwicklung steht ein Überblick über bereits vorhandene Verfahren, welche möglichst weitgehend die Anforderungen aus Abschnitt 1.2 erfüllen.

Abschnitt 2.1, „Robotermodelle“ gibt zunächst einen Überblick über die wesentlichen Arbeiten im Bereich Stabilitätskriterien und Robotermodellen und verdeutlicht zudem Vor- und Nachteile der jeweiligen Ansätze in Hinblick auf die Anforderungen. Anschließend vergleicht Abschnitt 2.2 existierende Algorithmen und wählt zwei vielversprechende Ansätze aus, die in den folgenden Kapiteln weiterentwickelt und evaluiert werden.

2.1 Robotermodelle

Forschung im Bereich zweibeiniges Laufen, die sich mit der Dynamik und Laufmustererzeugung beschäftigt, begann bereits in der ersten Hälfte des 20. Jahrhunderts [4] mit der Motivation, gehbehinderten Menschen zu helfen. Bevor Miomir Vukobratović 1973 ein Exoskelett zu diesem Zweck präsentierte [51], definierte er wenige Jahre zuvor den Begriff Körperstabilität [53] und entwickelte 1969 ein Stabilitätskriterium [54], das er später „Zero Moment Point“ (ZMP) nannte [51]. Er basiert auf der Festkörperdynamik eines Roboters, im englischen „Rigid Body Dynamics“ (RBD). Es existieren verschiedene Stabilitätskriterien, die ebenfalls ZMP genannt werden, allerdings teilweise auf anderen, simpleren Modellen basieren. Verschiedene Modelle und die darauf basierenden Stabilitätskriterien sind als Hierarchie ihrer Abstammung in Abbildung 2.1 dargestellt und werden in diesem Kapitel näher erläutert. Zunächst jedoch beschäftigt sich der folgende Abschnitt näher mit der ursprünglichen Definition des ZMP.

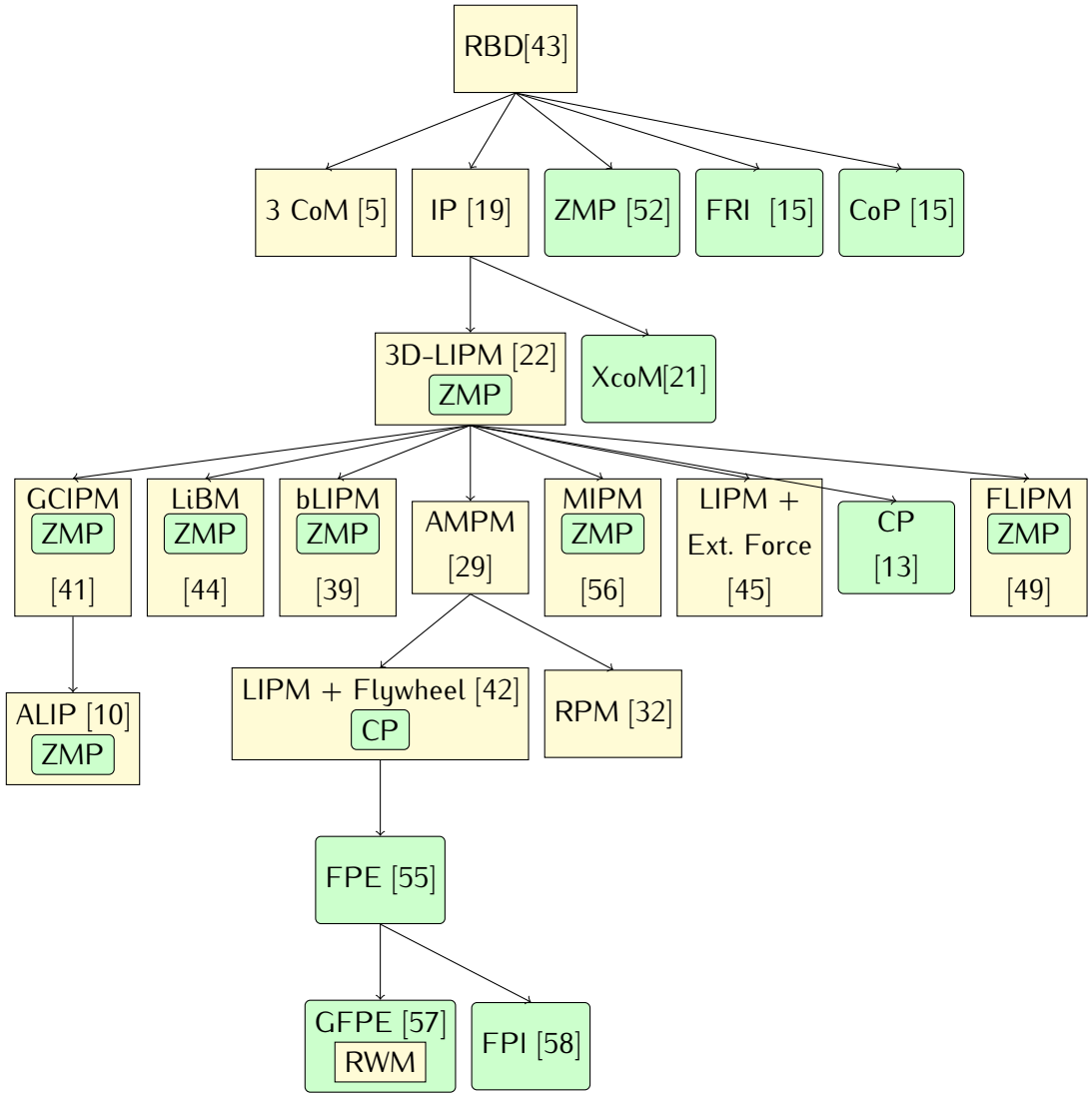

Abbildung 2.1: Hierarchie zur Abstammung verschiedener Robotermodelle (gelb) und darauf basierenden Punkten (grün). Häufig sind Modelle und die Punkte, die man mit ihnen berechnen kann, eng verknüpft und werden zusammen in der jeweiligen Arbeit behandelt. Eine klare Trennung ist daher oft nicht möglich. Die Farben und Bezeichnungen folgenden daher den Schwerpunkten, die von den Autoren der Arbeiten selbst gelegt werden.

2.1.1 Zero Moment Point mittels Festkörperdynamik

Vukobratovićs Definition von Körperstabilität ist ähnlich zu der hier getroffenen Definition. Ein System ist stabil, wenn eine geschlossene Region R existiert, die die ungestörte Trajektorie der drei räumlichen Winkel und der Höhe umschließt, so dass im Falle einer Störung die Trajektorie im zeitlich unendlichen in diese Region zurückkehrt. Ein schwankender Roboter ist demnach stabil, solange er in die Ausgangsposition zurück kehrt.

Mit dem ZMP lässt sich hingegen die Frage nach Stabilität bereits beantworten, bevor der Roboter in seine Ausgangslage zurückgekehrt oder gestürzt ist, allerdings ist der Stabilitätsbegriff des ZMP nicht äquivalent zu der Definition von Körperstabilität. Der ZMP ist der Angriffspunkt der Gegenkraft vom Boden auf den Roboter, bei dem die Drehmomente um die Transversalachse[1] (bei Roboter auch y-Achse genannt) und Sagittalachse[2] (bei Roboter auch x-Achse genannt) , erzeugt durch die Gegenkraft und dem Gegendrehmoment vom Boden auf den Roboter, gleich Null sind [52]. Der Roboter ist dann stabil, wenn der ZMP innerhalb des Stützpolygons liegt. Da der ZMP der Angriffspunkt der Gegenkraft ist, kann er nicht außerhalb des Stützpolygons liegen. Rein rechnerisch wäre das möglich, wenn man die beschränkende Fußgröße außer acht lässt. Dafür wird allerdings der Begriff „Foot Rotation Indicator“ (FRI) vorgeschlagen [15], wobei Vukobratović dafür den Begriff „Fictious ZMP“ (fZMP) verwendet.

Ein ähnlicher Punkt ist der „Center of Pressure“ (CoP) [15], wo die Gegenkraft des Bodens auf den Roboter wirkt, aber keine Forderungen an die Drehmomente gestellt werden. Er wird vielmehr als gewichtetes Mittel der Positionen berechnet, an denen die Kräfte greifen. Er ist daher mit Kraftsensoren messbar, was in Abschnitt 4.1 näher erläutert wird. Es lässt sich zeigen, dass im Falle der Existenz des ZMP, er mit dem CoP übereinstimmt [15], ist allerdings im allgemeinen nicht mit dem ZMP gleichzusetzen, da der CoP auch existiert, wenn der Roboter kippt. Daraus entsteht die Problematik, dass dem CoP nicht zu entnehmen ist, ob der Roboter kippt oder nicht. Im ersteren Fall liegt er an der Kante des Fußes, um die der Roboter kippt, dennoch kann er auch im letzteren Fall dort liegen.

Während die Messung des CoP einfach und schnell durchzuführen ist, gilt das nicht für die Berechnung von ZMP oder FRI. Da beide auf dem RBD basieren, müssen dazu Algorithmen wie der Newton-Euler-Algorithmus verwendet werden [43, Kapitel 2.5.1]. Der Zeitbedarf für die Berechnung ist allerdings hoch und für gewöhnlich nicht mit der vorgegebenen Echtzeitbedingung vereinbar. Dies motivierte viele Forscher Abstraktionen zu entwickeln, die nicht mehr alle Schwerpunkte und Trägheitsinformationen des Roboters beinhalten. Der nächste Abschnitt stellt das gebräuchlichste Modell vor.

[1] Die Transversalachse bezeichnet bei Humanoiden die Achse von rechts nach links, in der Robotik auch als y-Achse bezeichnet.

[2] Die Sagittalachse bezeichnet die Achse bei Humanoiden von hinten nach vorne, in der Robotik auch als x-Achse bezeichnet.

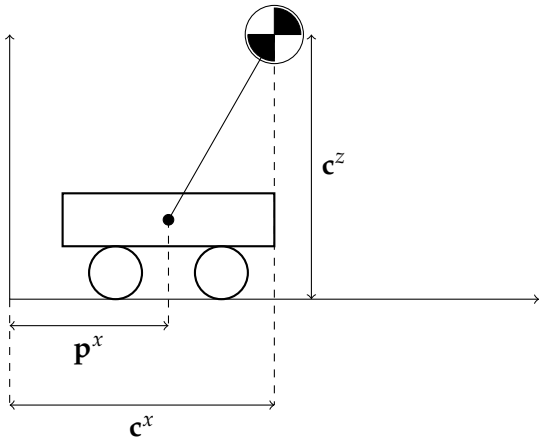

Abbildung 2.2: Lineares Modell zur Bestimmung des ZMP. Die Masse wird durch den Wagen so beschleunigt, dass die Höhe des Schwerpunktes konstant ist. Der ZMP $\mathbf{p}^x$ stimmt daher immer mit dem Stützpunkt überein.

2.1.2 Invertiertes Pendel

Die Problematik des Rechenaufwandes zur Bestimmung des ZMP führte bereits 1977 zu einer Abstraktion der Festkörperdynamik, die sich auf einen Schwerpunkt beschränkt und von einem masselosen Stab gestützt wird, dem invertierten Pendel (IP) [19]. Der große Vorteil des IP basiert auf der Tatsache, dass der fZMP sich genau an dem Punkt befinden muss, wo der Stab am Boden aufsetzt, damit die Masse weder an Höhe gewinnt noch verliert. Dieses Modell hat allerdings noch einen Nachteil, es ist nicht linear.
Den eigentlichen Durchbruch in der Laufforschung stellt daher das dreidimensionale lineare invertierte Pendelmodell (3D-LIPM) [22] dar, welches eine feste Schwerpunkthöhe $\mathbf{c}^z$ fordert. In Abbildung 2.2 ist die Masse über einen Stab mit einer Möglichkeit versehen, sie entlang der x-Achse zu beschleunigen. Die Länge des Stabes ist variabel, nicht aber die Höhe des Schwerpunktes. Daher stimmt der ZMP $\mathbf{p}^x$ per Definition mit dem Aufsatzpunkt des Pendels überein. Ist das Pendel nicht senkrecht, muss die Masse daraus folgend beschleunigt werden. Aus den Gleichungen für ein IP lässt sich mit diesen Annahmen folgende Gleichung herleiten [22]:

$$\mathbf{p}^x = \mathbf{c}^x - \frac{\mathbf{c}^z}{g}\ddot{\mathbf{c}}^x, \tag{2.1}$$

wobei $\mathbf{c}^x$ die Position des Schwerpunktes entlang der x-Achse ist und g die Gravitationskonstante.
Mit Hilfe dieses simplen Modells lässt sich durch Umstellen von Gleichung 2.1 der Zusammenhang zwischen Beschleunigung und ZMP intuitiv verständlich machen:

$$\mathbf{p}^x = \mathbf{c}^x - \frac{\mathbf{c}^z}{g}\ddot{\mathbf{c}}^x \Rightarrow |\mathbf{c}^x - \mathbf{p}^x| = \frac{\mathbf{c}^z}{g}|\ddot{\mathbf{c}}^x|. \tag{2.2}$$

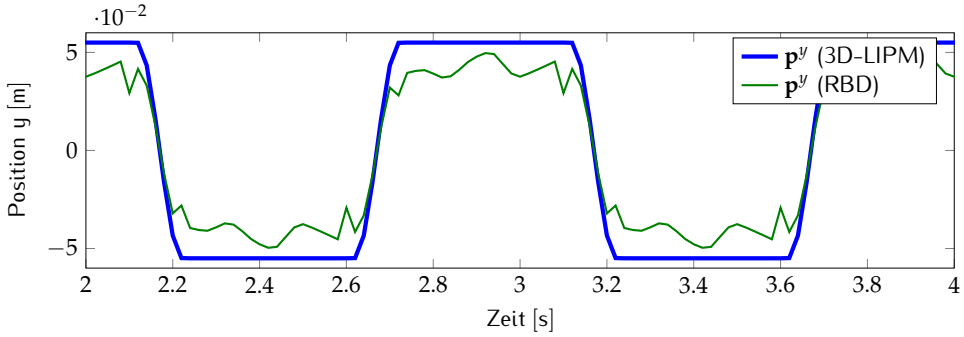

Abbildung 2.3: Schwerpunktverlauf und Vergleich zwei verschiedener Modelle eines zweibeinigen Laufs.

Da $\frac{\mathbf{c}^z}{g}$ konstant ist, hängt der Abstand vom ZMP zum Schwerpunkt entlang der x-Achse nur von der aktuellen Beschleunigung ab. Dies gilt allerdings nur für dieses Modell, im Allgemeinen nicht für ein komplexeres Modell wie der Festkörperdynamik.

Abbildung 2.3 verdeutlicht den Unterschied zwischen den ZMPs berechnet mittels RBD und 3D-LIPM bei einem beispielhaften Roboter. Der dargestellte Lauf ist laut dem ZMP des 3D-LIPM nahezu optimal, nimmt man an, dass die Fußmitte bei $\pm 5.5\,\mathrm{cm}$ liegt. Eine Bewertung anhand der RBD ergibt eine sichtbar andere Aussage. Dies motiviert, das invertierte Pendelmodell in verschiedener Weise zu erweitern, was Gegenstand des nächsten Abschnittes ist.

2.1.3 Erweiterungen des invertierten Pendels

Es existieren verschiedene Erweiterungen des 3D-LIPM, die unterschiedliche Nachteile ausgleichen sollen. Eine naheliegende Kritik ist die starke Abstraktion auf nur einen Schwerpunkt. Vor allem die Beine können die Lösung ungenau machen, da sie relativ schwer sind und vor allem das Schwungbein starken Beschleunigungen unterliegt. Ein Vorschlag ist daher das Modell „Gravity Compensated Inverted Pendulum Model“ (GCIPM) [41], welches für das Schwungbein einen weiteren Schwerpunkt vorsieht. Ein Wechsel des Standbeines bedeutet für dieses Modell eine größere Änderung. Ein weiterer Vorschlag ist daher, für jedes Bein einen Schwerpunkt vorzusehen [5].

Auch andere Forscher sehen in dem Ansatz, nur ein Bein zu modellieren, einen Nachteil. Das „Biped Linear Inverted Pendulum Model“ (bLIPM) [39] sieht zwar auch nur einen Schwerpunkt vor, modelliert jedoch explizit die Double-Support-Phase, in der beide Beine Kraft auf den Schwerpunkt ausüben. Daher ist es möglich, dass die Stützpunkte des Doppelpendels

mit den tatsächlichen Fußpositionen übereinstimmen. Der Stützpunkt des 3D-LIPM muss sich hingegen während des Schrittes ändern[3].

Auch das „Linear Biped Model“ (LiBM) [44] geht davon aus, dass der modellierte Stützpunkt für das Pendel mit der wirklichen Fußposition übereinstimmt, und bestimmt daher auf ähnliche Weise wie das bLIPM eine separate Dynamik für die Double-Support-Phase. Einen noch variableren Ansatz verfolgt das „Moving Inverted Pendulum Model“ (MIPM) [56], welches speziell gebogene Fußsohlen modelliert, über die sich der Läufer abrollt. Beide Modelle beschränken sich wie das bLIPM dabei auf einen einzelnen Schwerpunkt.

Das „Augmented Linear Inverted Pendulum Model“ (ALIP) [10] versteht sich als eine weitere Verallgemeinerung des GCIPM, indem eine allgemeine Funktion F zur Gleichung des 3D-LIPM addiert wird. Die Funktion F muss eine spezielle Form haben, so dass Gleichung 2.1 weiterhin eine lineare Differentialgleichung zweiter Ordnung ist, wodurch sie mit Standardmethoden gelöst werden kann.

Die in diesem Abschnitt erwähnten Erweiterungen haben das Ziel gemein, die Dynamik genauer abschätzen zu können als das 3D-LIPM. Es gibt zudem eine Gruppe weiterer Modelle, die nicht mit dem ZMP verknüpft sind, sondern mit dem „Capture Point“ (CP) [42], der sich mit Hilfe einer speziellen Art von Erweiterungen des IP ergibt. Diese Erweiterungen und verschiedene Varianten des CP sind Thema des nächsten Abschnittes.

2.1.4 Capture Point

Der ZMP gibt Auskunft darüber, ob der Lauf aktuell stabil ist oder nicht. Falls er das nicht ist, bleibt aber offen, wie genau die Störung aussieht oder wie sie zu behandeln ist. Der FRI bzw. fZMP gibt zumindest eine Bewertung, wie stark der Roboter derzeit kippt, allerdings lässt sich auch damit nicht ohne Weiteres eine Behandlungsmöglichkeit berechnen.

Untersuchungen am Menschen zeigen verschiedene Methoden, die der Mensch zum Ausgleich von Störungen anwendet [31]. Es wird gefolgert, dass die „Hüftstrategie“, eine Bewegung von Knöchel und Hüfte im Verhältnis 1:3, die vom Menschen bevorzugte Variante ist. Dies motiviert alternative Erweiterungen des 3D-LIPM, die diese Strategie adäquat abbilden können. Das „Angular Momentum Inducing Inverted Pendulum Model“ (AMPM) [29] erweitert das 3D-LIPM um die Möglichkeit eines Drehmoments um den Schwerpunkt. Auf derselben Idee basiert auch das „Linear Inverted Pendulum Plus Flywheel Model“ und erlaubt die Berechnung des „Capture Points“ (CP) [42]. Der CP ist der Punkt am Boden, der als Referenz für den Ort des CoP dient, damit der Roboter direkt darüber zum Stehen kommt. Wie auch bei der Hüftstrategie des Menschen ist das Ziel demnach ein statischer Stand.

[3] Es sei hier angemerkt, dass die Änderung des Stützpunktes bei den invertierten Pendelmodellen kein Nachteil darstellt, da Gleichung 2.1 für jeden Zeitpunkt neu aufgestellt werden kann. Wie in Abschnitt 2.2 dargestellt ist das sogar ein Vorteil.

Der „Foot Placement Estimator“ (FPE) [58] ist ein ähnlicher Punkt mit dem Vorteil, dass er nicht wie der CP seine Position im Laufe der Zeit ändert. Wie auch beim invertierten Pendel kann man die Abstraktion auf einen Schwerpunkt kritisieren, was zu dem Vorschlag des „Foot Placement Indicator“ (FPI) [58] führt, der den FPE auf beliebig viele Schwerpunkte und Trägheitstensoren erweitert. Auch eine Erweiterung auf unebenem Boden existiert unter dem Begriff „Generalized Foot Placement Estimator“ (GFPE) [57], der sich allerdings auf einen Schwerpunkt beschränkt.
Auch wenn der CP und seine verschiedenen Varianten speziell zum Stoppen der Bewegung ausgelegt sind, gibt es Ansätze für Laufalgorithmen, die zusammen mit Laufalgorithmen der anderen hier präsentierten Modelle im nächsten Abschnitt vorgestellt werden.

2.2 Algorithmen

Im vorherigen Abschnitt wurden einige Modelle vorgestellt, mit denen sich verschiedene Punkte berechnen lassen, wie den ZMP und den CP. Sie bieten die Möglichkeit, Laufalgorithmen zu entwerfen, die die Dynamik des Roboters beachten und daher nach gängigen Kriterien wie dem ZMP stabil sind, wobei es auch möglich ist einen Lauf zu entwerfen, der nicht auf physikalischen Gesetzen beruht.
Der folgende Abschnitt zeigt einen Überblick über verschiedene Verfahren mit und ohne physikalischem Hintergrund und deren Vor- und Nachteile, zunächst ohne die Rückführung von Sensordaten. Unterabschnitt 2.2.3 stellt danach einige existierende Möglichkeiten vor, diese Laufsteuerungen um Sensorkontrollen zu erweitern, um auch auf unvorhergesehene Ereignisse reagieren zu können.

2.2.1 Grundlagen zur Bewegungssteuerung

Die Entwicklung eines Laufalgorithmus wird durch die Entscheidung, ob er auf physikalischen Prinzipien beruht, wesentlich beeinflusst. Laufbewegungen ohne physikalische Berechnung benötigen während des Laufs kaum Rechenzeit. Die Bewegungen müssen allerdings zuvor optimiert werden, was zu Mustern führt, die im Allgemeinen nur für den verwendeten Roboter und Untergrund optimal sind. Unter der Verwendung von physikalischen Gesetzen ist es hingegen möglich, viele Werte, die ansonsten blind optimiert werden müssen, durch Berechnungen zu ersetzen, wodurch die Läufe allgemeingültiger sind, solange der Roboter und die Umgebung nicht wesentlich vom physikalischen Modell abweichen.
Das ist nicht nur vorteilhaft wenn der Untergrund sich ändert (beispielsweise die Stärke des Teppichs) sondern auch um bei der Wahl der Laufrichtung und Geschwindigkeit flexibler zu sein. Man spricht von einem „omnidirektionalen Lauf“, wenn Vorwärts-, Seitwärts- und Rotationsgeschwindigkeit in beliebigen Kombinationen vorgegeben werden können. Können diese Werte (oder einzelne) während des Laufs verändert werden, spricht man von einem „Online“-

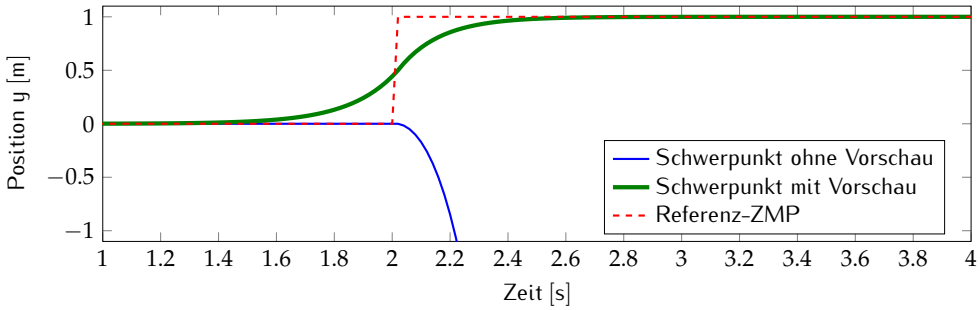

Abbildung 2.4: Berechnung des Schwerpunktverlaufs mittels 3D-LIPM mit und ohne Vorschau des ZMP.

Algorithmus (für Details siehe Abschnitt 1.2, „Anforderungen“, Punkt „Onlinefähigkeit“'). In vielen Laufalgorithmen werden diese Werte zunächst in andere Referenzwerte umgewandelt, meist in Soll-Fußpositionen. Diese können dazu benutzt werden, eine Soll-Trajektorie des ZMP zu erzeugen nach der Idee, dass der Roboter stabil laufen wird, solange der tatsächliche ZMP mit dem gewünschten übereinstimmt.

Abhängig von der gewählten physikalischen Grundidee des Algorithmus kann diese Referenz-Trajektorie auf bestimmte Formen beschränkt sein. Häufig ist diese fest, stimmt demnach während eines Schrittes mit der Position des Fußes modellbedingt überein. Andere Verfahren verlangen eine Spline-, Fourier- oder lineare Kurve. Besser sind Algorithmen, die eine freie Wahl erlauben, da das Stützpolygon weitgehend ausgenutzt werden kann, um Vorüberlegungen einfließen zu lassen, die den Lauf stabilisieren können, wie in Abschnitt 1.2 (Punkt „Freie Referenzwahl“') und 5.2 erläutert. Im besten Fall werden nur Bedingungen an den ZMP-Verlauf gestellt, beispielsweise, dass er sich innerhalb bestimmter Grenzen befinden soll. Die ZMP-Trajektorie wird daraufhin automatisch anhand dieser Beschränkungen und eines Optimalitätskriteriums bestimmt.

Wird ein Algorithmus genutzt, der eine freie ZMP-Referenz erlaubt, wird grundsätzlich eine Vorschau der Referenz benötigt, wie in Abbildung 2.4 verdeutlicht. Bei Sekunde 2 wechselt der Referenz-ZMP von $0\,\mathrm{m}$ auf $1\,\mathrm{m}$ (die positive Richtung der y-Achse zeigt nach links vom Roboter aus gesehen), beispielsweise um das Gewicht von beiden Füßen des Roboters auf den linken Fuß zu verlagern und anschließend so stehen zu bleiben.

Verwendet man keine Vorschau, stimmt die Position des Schwerpunktes mit der gewünschten Position des ZMP bis zum Wechsel überein, der Roboter steht also ruhend. Dieses Verhalten lässt sich anhand von Gleichung 2.1 erklären, welche dargestellt für die y-Achse lautet: $\mathbf{p}^y = \mathbf{c}^y - \frac{\mathbf{c}^z}{g}\ddot{\mathbf{c}}^y$. In der Gleichung ist $\frac{\mathbf{c}^z}{g}$ konstant, so dass die Beschleunigung $\ddot{\mathbf{c}}^y$ des Schwerpunktes bestimmt, inwiefern der ZMP $\mathbf{p}^y$ von der Position $\mathbf{c}^y$ des Schwerpunktes abweicht. Beträgt die Beschleunigung 0, so ergibt sich demnach die Gleichung $\mathbf{p}^y = \mathbf{c}^y$. Bis Sekunde 2 ist das so auch erwünscht, denn ein stehender Roboter, dessen ZMP konstant unter dem Oberkörper liegt, soll auch ohne Bewegung stehen. Die Änderung des gewünschten ZMP von $0\,\mathrm{m}$ auf

1 m verursacht jedoch, dass das linke Bein den Oberkörper nun immer weiter nach rechts beschleunigen muss, was nach kurzer Zeit nicht mehr möglich ist und zum Sturz führt. Diese Bewegung ist ebenfalls eine Folge von Gleichung 2.1, denn eine negative Beschleunigung führt zur Addition eines positiven Wertes zur Position des Schwerpunktes um den ZMP zu bestimmen. Es sind also immer stärkere Beschleunigungen in negativer Richtung notwendig, um den ZMP in positive Richtung relativ zur aktuellen Schwerpunktposition zu verschieben. Ist eine freie Wahl des Referenz-ZMP erlaubt, existieren im Allgemeinen also Fälle, in denen es notwendig ist zuvor zu wissen, wie der ZMP in Zukunft verlaufen soll, um bereits vor der Bewegung des ZMP eine Bewegung des Schwerpunktes veranlassen zu können. So lässt sich mit einer stärker werdenden Beschleunigung der Schwerpunkt bis Sekunde 2 in Richtung des zukünftigen Referenz-ZMP verschieben. Dabei ist Gleichung 2.1 erfüllt, wenn auch in der Praxis unendlich kleine Beschleunigungen nicht möglich sind und daher leichte Abweichungen von der Referenz zu erwarten sind. Beim Wechsel bei Sekunde 2 durchkreuzt der Schwerpunktverlauf den Verlauf des ZMP, wie in Abbildung 2.4 zu erkennen, bleibt jedoch nicht kurzzeitig stehen, da der Wechsel von 0 m auf 1 m sprunghaft statt findet. In diesem Moment muss die Beschleunigung das Vorzeichen wechseln, damit der ZMP nun in positive Richtung von der Schwerpunktposition abweicht. Der Schwerpunkt wird durch die nun stattfindende negative Beschleunigung kontinuierlich abgebremst, bis er schließlich wieder über dem Referenz-ZMP zum Stehen kommt.

Die Form der Referenz und das physikalische Grundprinzip lassen häufig keine analytische Lösung zu. Speziell im Falle von harten Beschränkungen bezüglich der Referenztrajektorie sind numerische, iterative Optimierungen unumgänglich. Die Echtzeitfähigkeit ist in solchen Fällen nicht selbstverständlich. Verschiedene Lösungsverfahren, eingebettet in Laufalgorithmen, sind Thema des nächsten Abschnittes.

2.2.2 Laufalgorithmen

Tabelle 2.1 zeigt eine Auswahl der wichtigsten existierenden Verfahren im Vergleich. Eine wesentliche Eigenschaft der Algorithmen ist die Form der Referenz. Möglich sind frei wählbare Referenzverläufe („Frei“), lineare Verläufe („Linear“), zwingend konstante Positionen während eines Schrittes („Fest“), Spline-Funktionen („Spline“), auf Basis von Fourierreihen („Fourier“) und Verläufe, die nicht vorgegeben werden, sondern vom Algorithmus automatisch optimal gewählt werden („Optimal“). Es sind auch zwei Vertreter der Kategorie „Feste Trajektorien“ aufgeführt [28, 18], welche ohne physikalische Prinzipien arbeiten. Sie bestehen aus fest gewählten Bewegungsabläufen, wie Vorwärts- und Seitwärtsschritten, sowie aus Schritten zum Laufen von Kurven. Sie können beliebig aneinandergereiht werden, was während des Laufs geschieht, so dass sie omnidirektional und onlinefähig sind. Jedoch ist die Wahl der Bewegungsabläufe ohne physikalische Prinzipien zeitintensiv und führt zu Bewegungen, die optimiert sind für bestimmte Untergründe und für bestimmte Roboter. Ein anderer Untergrund

Stichwort	Quelle	Jahr	Modell	Analytisch	Online	Referenz
Optimal Gradient	[37]	1999	RBD	✗	✓	Frei
Virtual Torque	[7]	2012	GCIPM	✓	✓	Fest
Spline Collocation	[5]	2007	3 CoM	✗	✓	Linear
Impulsbasiert	[41]	1998	GCIPM	✓	✗	Fest
Convolution Sum	[27]	2007	3D-LIPM	✓	✓	Frei
Bewegungsgleichung	[10]	2010	ALIP	✗	✗	Fest
Energy Conserving Orbit	[23]	1992	3D-LIPM	✓	✗	Fest
Impulsbasiert	[56]	2006	MIPM	✓	✓	Fest
Abschnittsweise Planung	[17, 36]	2006	3D-LIPM	✓	✓	Spline
MPC	[20, 12]	2010	3D-LIPM	✗	✓	Optimal
PR-MPC	[45]	2010	1 CoM	✗	✓	Optimal
CPT, CPS	[13]	2011	3D-LIPM	✓	✓	Fest
Orbitalenergie	[39]	2011	bLIPM	✓	✓	Fest
Differentialgleichung	[30]	2003	AMPM	✓	✗	Fest
Orbital Energie Control	[44]	2009	LiBM	✓	✓	Fest
MPC	[3]	2002	RBD	✗	✓	Optimal
Feste Trajektorien	[28]	2006	✗	-	✓	-
Feste Trajektorien	[18]	2006	✗	-	✓	-
Preview Control	[24, 25]	2006	3D-LIPM	✓	✓	Frei
Abschnittsweise Planung	[40]	2010	3D-LIPM	✓	✓	Fourier

Tabelle 2.1: Überblick über verschiedene Laufalgorithmen und ihre Eigenschaften. Die Art und Weise, wie eine Lösung bestimmt wird, wird mit einem charakterisierenden Stichwort umschrieben. Weitere Eigenschaften sind die Lösbarkeit in analytischer Form, die Form der Referenz und ob der Lauf „online“ berechnet wird.

oder die Anwendung auf einem anderen Roboter kann bereits zu einem instabilen Lauf führen. Zudem ist ohne physikalisches Modell die Rückführung von gemessenen physikalischen Größen problematisch. Diese Ansätze werden hier daher nicht weiter betrachtet.

Unter den auf Physik basierenden Ansätzen finden sich Laufalgorithmen für viele der in Abbildung 2.1 aufgeführten Modelle. Einer der ältesten Ansätze ist zusammen mit dem 3D-LIPM vorgeschlagen worden [23] und kann mit dem Stichwort „Energy Conserving Orbit" beschrieben werden. Hier wird die Energie des Systems, bestehend aus kinetischer und potentieller Energie, als konstant angenommen und daraus eine Schwerpunktbewegung für einen Schritt berechnet. Die Lösung kann daher analytisch bestimmt werden, ist aber nicht onlinefähig, da bei einem Übergang von einem Schritt zum nächsten die Energie erhalten bleibt und damit auch den nächsten Schritt fest definiert. Zudem wird angenommen, dass der Kontaktpunkt des Pendels mit dem Boden auch gleichzeitig die Position des Fußes ist, was gleichbedeutend mit einem konstanten Referenz-ZMP während eines Schrittes ist. Ebenfalls über Energie werden die Laufalgorithmen für LiBM [44] und bLIPM [39] hergeleitet, allerdings ohne die Forderung, dass die Energie im System konstant bleibt. Änderungen in der Energie sind in der Double-Support-Phase daher möglich um die Geschwindigkeit zu ändern und um Ausfallschritte auszuführen, sollte die Energie durch externe Einflüsse verändert werden.

Eine weitere Gruppe von Lösungsmöglichkeiten leitet Differentialgleichungen her, die per Exponentialansatz lösbar sind. Die Differentialgleichungen können auf verschiedene Weisen aufgestellt werden. Möglich ist der Ansatz über die Bewegungsgleichung eines inversen Pendels, erweitert um die Besonderheiten der jeweiligen Modelle [10, 30], siehe Abschnitt 2.1.3 bzw. 2.1.4. Auch eine impulsbasierte Herleitung ist möglich, wie von Yeon et al. [56] und Park et al. [41] vorgeschlagen, wobei für letzteren Ansatz auch eine verbesserte Version existiert, die onlinefähig ist und in einer Simulation auch rennen[4] ermöglicht [7]. Erreicht wird das durch eine Differenz zwischen dem ZMP und dem Berührungspunkt des Pendels mit dem Boden. Da der ZMP der Punkt am Boden ist, bei dem keine Drehmomente auf den Roboter um die x- und y-Achse übertragen werden, wird bei einer Abweichung des Berührungspunktes vom ZMP ein Drehmoment um wenigstens eine dieser Achsen ausgeübt. Dieses Drehmoment wird von den Autoren „Virtual Torque" genannt und wird hier genutzt, um den Roboter in eine gewünschte Richtung zu beschleunigen. Handelt es sich dabei beispielsweise um ein Drehmoment um die y-Achse, so resultiert dies in eine Beschleunigung des Oberkörpers in Richtung der x-Achse.

Eine ähnliche bekannte Methode um onlinefähige Algorithmen zu erhalten ist die gleichzeitige Planung von Schwerpunkttrajektorie und ZMP für die nächste Laufphase. Es muss sichergestellt werden, dass die Anfangs- und Endbedingungen (Geschwindigkeit, Position usw.) für diese Phase eingehalten werden. Ansonsten kann der ZMP-Verlauf beliebig geplant werden, wobei verschiedene Algorithmen unterschiedliche Einschränkungen beim

[4] Als Rennen bezeichnet man einen Lauf, bei dem beide Füße für einen Moment keinen Kontakt zum Boden haben.

ZMP-Verlauf aufweisen. Bei den Vorschlägen einer Forschergruppe des AIST (Japan) muss der ZMP als Spline-Kurve definiert sein [17, 36], wohingegen Park et al. eine Fourierreihe vorschlagen [40]. Ein Ansatz vorgeschlagen von Buschmann et al. [5] verwendet als Referenz Drehmomente an den Gelenken, die Abschnittsweise linear sein müssen.

Wie bereits oben erwähnt, wäre es wünschenswert den Verlauf frei wählen zu können. Nagasaka et al. [37] schlagen ein Verfahren vor, dass onlinefähig ist und mit dem Mehrkörpermodell des Roboters arbeitet. Das realitätsnähere Modell macht allerdings eine Optimierung notwendig, wobei die optimale Gradientenmethode vorgeschlagen wird. Ebenfalls onlinefähig bei freier Referenz ist der Vorschlag von Kajita et al. einen Regler zu verwenden, dessen Stellgröße die Ableitung des ZMP ist und als Führungsgröße der Referenz-ZMP dient, so dass die numerische Optimierung entfällt [24, 25]. Ein gängiger Begriff für den Regler ist „Preview-Controller" (PC), da er die oben erklärte Vorschau des ZMP benötigt. Ein Vorschlag mit nahezu gleichen Eigenschaften von Kim ist die Faltung der Impulsantwort des Systems mit einem ZMP-Impuls [27], Stichwort Convolution Sum (CS).

Der erwähnte PC ist eine frühe Form von Regler, der später seine Verallgemeinerung im „Model Predictive Control" (MPC) fand. Damit sind Regler gemeint, die ein optimales Ausgangssignal anhand einer Gütefunktion bestimmen, die ähnlich zum PC das Ausgangssignal im Vergleich zu einer Vorschau der Referenz bewertet. MPC-Systeme können wie auch PC-Systeme geschlossen gelöst werden (wobei bestimmte Matrizen zuvor numerisch bestimmt werden müssen). Im Gegensatz zu PC ist es bei MPC möglich, neben der Gütefunktion Ungleichungen als Beschränkungen zu formulieren. In diesem Fall lässt sich das System nicht mehr geschlossen lösen, sondern es bedarf numerischer Algorithmen [6]. Dazu wird beispielsweise ein Optimierungsproblem vom Typ „Quadratische Programmierung" formuliert, zur dessen Lösung Bibliotheken existieren, beispielsweise qpOASIS [14][5].

MPC kann auf verschiedene Modelle angewendet werden. Azevedi et al. verwenden RBD, was allerdings mit den erwartbaren Geschwindigkeitseinbußen einhergeht [3]. Die Forschung beschäftigt sich daher auch hier mit vereinfachten Modellen. Beim Vorschlag von Herdt et al. [20] wird MPC auf Basis von 3D-LIPM angewendet. Die Autoren geben eine durchschnittliche Zeit von 1 ms zur Lösung des quadratischen Programms an [20, 12], zeigen allerdings die Anwendbarkeit der Ausfallschritte nur in einer Simulation. Ein weiterer Vorschlag, bekannt als „Push Recovery Model Predictive Control" (PR-MPC) [45], integriert Ausfallschritte, wobei aufgrund des numerischen Lösungsverfahrens die Echtzeitfähigkeit nicht selbstverständlich ist. Diese Frage lassen die Autoren jedoch offen. Der Hersteller des humanoiden Roboters „Nao", Aldebaran-Robotics, nutzt das gleiche Verfahren in seinem Lauf, löst die Optimierung allerdings nicht in jedem Zeitschritt [16], was daher nur für Laufsteuerungen in Frage kommt.

Die Idee, dass der Lauf nicht durch den Referenz-ZMP definiert wird, sondern durch die Fußpositionen, lässt sich auch auf eine weitere, neue Weise umsetzen. Der Capture Point

[5] http://www.qpoases.org

(siehe Unterabschnitt 2.1.4) kann wie der ZMP auch invers benutzt werden, indem der gewünschte Ort festgelegt wird, woraus dann eine ZMP Kurve erzeugt wird, die genau zu dieser CP-Position führt. Ott et al. zeigen und vergleichen dazu zwei Verfahren [13], die allerdings noch nicht in der Lage sind die geplanten Schrittpositionen zu modifizieren.

2.2.3 Laufen mit Sensorkontrolle

Einige der im vorherigen Abschnitt vorgestellten Algorithmen verfügen bereits über die Möglichkeit, mit Hilfe von Sensordaten den Lauf zu stabilisieren, beispielsweise indem mit Zuständen von physikalischen Systemen gearbeitet wird. Häufig gehen die Ansätze davon aus, dass der Zustand dem tatsächlichen Zustand des physischen Systems entspricht, ohne genauer darauf einzugehen, wie dies gewährleistet wird. Eine Ausnahme bildet der bereits oben erwähnte Ansatz PR-MPC, der dem ebenfalls vorgestellten MPC sehr ähnlich ist. Die Autoren zeigen hier den Einsatz der „Inertial Measurement Unit“ (IMU) [6] und von Gelenkwinkelsensoren, um den Zustand des Modells zu aktualisieren. Wie auch bei MPC folgen daraus gegebenenfalls geänderte Referenz-ZMP-Trajektorien und modifizierte Sollfußpositionen.
Auf einen anderen Ansatz in Tabelle 2.1 wird in Abschnitt 2.2.1 nicht näher eingegangen, da er auf festen Trajektorien basiert [28]. Er ist allerdings hier von Bedeutung, da auf einige Sensorkontrollen eingegangen wird. Beispielsweise wird speziell mittels Flexibilität von bestimmten Gelenken anhand gemessener Gelenkwinkel gedämpft und der Lauf verzögert, sollte der Fuß nicht zum geplanten Zeitpunkt aufsetzen. Ausfallschritte sind in diesem Ansatz allerdings nicht vorgesehen.
Häufig setzen Ansätze zu Sensorkontrollen auf physikalischen Modellen und Algorithmen auf. Eine interessante Möglichkeit wird von Meriçli et al. vorgestellt [33], wo nur das Modell auf physikalischen Prinzipien beruht, nicht aber die Sensorkontrolle. Es wird stattdessen der Roboter von einem Menschen gesteuert, der durch Betrachtung des Laufs Korrekturen vornimmt, die der Roboter dann in Abhängigkeit von den gemessenen Werten lernt und später selbstständig vornehmen kann. Aufgrund der schwierigen Steuerung durch den Menschen ist das allerdings bedingt erfolgreich. Ebenfalls nur bedingt erfolgreich ist der Versuch die Orbitalenergie zu kontrollieren, indem der Oberkörperwinkel genutzt wird, um die tatsächliche Energie zu bestimmen und entsprechend die Bewegung zu verlangsamen oder zu beschleunigen [1]. Aufgrund der schlechten Sensoren in der verwendeten Roboterplattform funktioniert das nur in seitlicher Richtung und nur als Entscheidung, ob der kippende Roboter mit der Fortsetzung des Laufs warten soll. Morisawa et al. schlagen ebenfalls eine Erweiterung ihrer vorherigen Arbeiten vor, um die mittels des analytischen Ansatzes erzeugten Referenz-Trajektorien zu modifizieren [34, 35]. Dazu werden die Daten der IMU, Gelenkwinkelsensoren und Kraftsensoren in den Füßen verwendet, um den Zustand des Modells zu aktualisieren.

[6] Die „Inertial Measurement Unit“ ist eine Kombination aus Beschleunigungssensor und Gyroskop zur Schätzung der Lage und Position im Raum.

In beiden Ansätzen wird allerdings, wie auch beim MPC, eine Zielfunktion formuliert, die iterativ minimiert werden muss.
Wie schon bei den vorgestellten Algorithmen zur Laufsteuerung und -regelung deutlich wurde, gibt es kaum Ansätze, die alle für diese Arbeit formulierten Anforderungen erfüllen, was im folgenden Abschnitt weiter diskutiert wird.

2.3 Diskussion

Aus der zuvor vorgestellten Vielzahl von Modellen und Laufsteuerungen kommen keine in Frage, die keinen variablen ZMP zulassen, da dieser laut den Anforderungen enorme Vorteile bietet. In Tabelle 2.1 kommen daher nur die Ansätze „Optimal Gradient“, „Convolution Sum“ (CS), und „Preview Controller“ (PC) in Frage, da die anderen den Referenz-ZMP unnötig einschränken. Die Ansätze „MPC“ und „PR-MPC“ nutzen keinen Referenz-ZMP, sondern legen ihn automatisch anhand der Referenz-Fußpositionen fest, was bei geeigneter Wahl der harten Beschränkungen des Optimierungsproblems sogar ein Vorteil gegenüber der manuellen ZMP-Wahl sein kann.
Des Weiteren muss der Algorithmus laut den Anforderungen echtzeitfähig sein, wodurch Algorithmen mit iterativen Lösungsverfahren nur bedingt in Frage kommen. Herdt et al. [20] und Dimitrov et al. [12] stellen allerdings in ihren Arbeiten fest, dass die Lösung des Optimierungsproblems im Mittel nur 1 ms benötigt, was demnach als echtzeitfähig einzustufen wäre. Diese Ergebnisse lassen sich bei einer Frequenz von 10 Hz nachvollziehen, wobei hier mindestens 100 Hz gefordert werden. Abschnitt 6.2 beschäftigt sich mit den daraus folgenden negativen Auswirkungen.
Neben MPC sind auch die Ansätze CS und PC vielversprechend, da sie alle gewünschten Kriterien erfüllen. Sie sind anzunehmen als

- echtzeitfähig,
- onlinefähig, da die benötigte Vorschau des Referenz-ZMP bzw. der Schritte endlich ist und damit nach der Vorschau anders gewählt werden kann, ohne den Lauf neu zu starten,
- winkelbasiert, da die Schwerpunktpositionen per inverser Kinematik in Winkel umgerechnet werden können.

In der Forschung findet CS keine weite Verbreitung, was damit erklärt werden kann, dass es vergleichbare Eigenschaften besitzt, jedoch später vorgestellt wurde. Es werden daher die beiden Ansätze MPC und PC weiter verfolgt. Sie sind allerdings um eine Sensorkontrolle und PC insbesondere um Ausfallschritte zu erweitern.
Die genannten Eigenschaften müssen zudem beim PC wie auch MPC experimentell verifiziert werden, da die genauen Umstände, unter denen die Ergebnisse in der Literatur zustande

kamen, unklar sind. Zudem wurde der MPC bisher nicht auf einem realen Roboter gezeigt, so dass seine Eignung in Frage zu stellen ist.

KAPITEL 3

Bewegungssteuerung

Ziel dieses Kapitels ist, die im vorherigen Abschnitt ausgewählten Laufalgorithmen genauer vorzustellen. Sowohl „Preview Control“ (PC) als auch „Model Predictive Control“ (MPC) arbeiten mittels eines Zustandes des Roboters und sind daher grundsätzlich in der Lage, Sensordaten, die den Zustand beeinflussen, zu verarbeiten. Die Art und Weise, wie Sensordaten in den Zustand einfließen, sind allerdings nicht Gegenstand dieses Kapitels. Üblicherweise wird in diesem Fall, wo keine Sensordaten verwendet werden, nicht nur der eigentliche Regler ausgeführt, sondern es wird auch mitberechnet, wie das System sich im Idealfall verhält [24, 25]. Dazu wird auf den aktuellen Zustand zum Zeitpunkt t die Ausgabe des Reglers angewendet und per Systemmodell der voraussichtliche Zustand zum nächsten, künftigen Zeitpunkt $t + 1$ bestimmt. Diese Rechnung ist mit einer Simulation vergleichbar, die allerdings den Roboter mit dem Modell simuliert, welches auch vom Regler verwendet wird. Es handelt sich also um eine Laufsteuerung.

Es existiert ein weiterer Grund, der die Bestimmung eines solchen Idealzustands auch bei der Rückführung von Sensordaten notwendig macht. Die Ausgabe des Reglers bei PC und MPC ist in der Literatur die Ableitung der Komponente des Zustandsvektors mit dem höchsten Index [25, 11]. Eines der Ziele dieser Arbeit ist jedoch die Ansteuerung der Gelenke per Gelenkwinkel, so dass die Ausgabe des Reglers nicht direkt verwendet werden kann. Durch die Bestimmung des idealen Zustands zum Zeitpunkt $t + 1$ steht jedoch ein anwendbarer Wert zur Verfügung. Bei 3D-LIPM ist das die Position des Schwerpunktes als Teil des Zustands, welcher per inverser Kinematik in Gelenkwinkel überführt werden kann. Die Details sind Thema von Kapitel 5.

Dieses Kapitel orientiert sich an den Arbeiten von Kajita et al. [24, 25] bzw. Diedam et al. [11]. Beweise und Herleitungen der jeweiligen Gleichungen sind diesen Arbeiten zu entnehmen, wobei die Arbeit von Kajita et al. auf den regelungstechnischen Grundlagen von Katayama et

al. [26] basiert, so dass Teile der Beweise dort zu finden sind. Beweise und ihre Ergebnisse, die nicht veröffentlicht wurden, sind in dieser Arbeit gegeben.
In dieser Arbeit wird zur Symbolisierung von Geschwindigkeit und Beschleunigung die Newton'sche Notation der Ableitung verwendet. Ein Index bei einer kontinuierlich dargestellten Variable bedeutet, dass diese einen diskreten Wert zu einem bestimmten Zeitpunkt darstellt. Insbesondere MPC und PC finden hier als diskrete Verfahren Anwendung, so dass es sich grundsätzlich um diskrete Funktionen handelt. Es sei erwähnt, dass Sensorwerte diskret abgetastet und mit einer Frequenz von 100 Hz vom Roboter zur Verfügung gestellt werden.
Zunächst handelt dieses Kapitel vom Model Predictive Control Algorithmus, der bereits in der Lage wäre Ausfallschritte auszuführen, obwohl dies ohne Sensordaten nicht geschieht. Im anschließenden Abschnitt wird der Preview Controller vorgestellt, der um die Möglichkeit von Ausfallschritten in Abschnitt 4.3 erweitert wird.

3.1 Bewegungssteuerung mittels Model Predictive Control

Model Predictive Control ist eine Regelungsmethode, die in der Industrie eine breite Anwendung findet [6]. Gründe dafür sind

- eine einfache, intuitive Handhabung, so dass auch ohne tiefere regelungstechnische Kenntnisse gute Ergebnisse erzielt werden können,
- die Möglichkeit auch mit multivariablen Regelstrecken umgehen zu können,
- die Kompensation von Totzeiten,
- die allgemeine Form, so dass sie auf eine Vielzahl von sehr unterschiedlichen Problemen angewendet werden kann,
- die Nutzung eines Systemmodells,
- die Anwendung einer Vorschau der Länge N, welche für viele Einsatzzwecke benötigt wird, beispielsweise wie in Abschnitt 2.2.1 beschrieben,

um nur einige zu nennen. Bei MPC ist es möglich, neben der Zielfunktion auch harte Bedingungen zu formulieren, was insbesondere für humanoides Laufen sinnvoll genutzt werden kann. So lassen sich Schrittmodifikationen auf physisch realisierbare Regionen beschränken. Der ZMP lässt sich mittels harter Beschränkungen auf die Stützpolygone der verschiedenen Zeitpunkte beschränken, so dass neben der Laufrichtung auch ein stabiler Lauf gewährleistet wird. So kann der Regler den ZMP flexibel innerhalb des Stützpolygons wählen und falls nötig auch das Stützpolygon durch Modifikation der Schritte ändern, was insbesondere im Störungsfall nützlich sein kann.

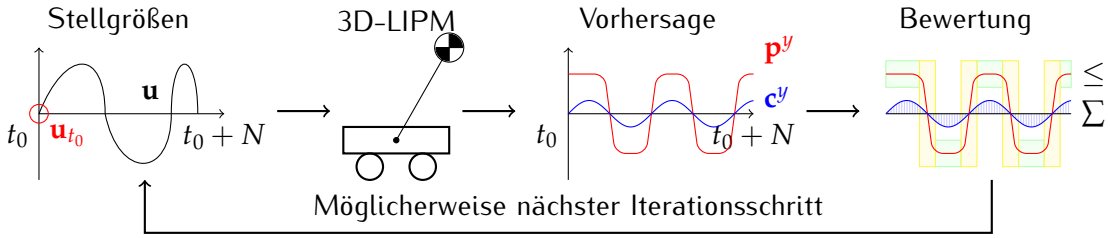

Abbildung 3.1: Vereinfachte Darstellung der Idee zur Erzeugung von Stellgrößen $\mathbf{u}$ nach MPC.

Abbildung 3.1 zeigt die Idee im Detail, wie man sich die Erzeugung einer Stellgröße für den aktuellen Zeitpunkt vorstellen kann. Dieser Ablauf muss für jeden Zeitschritt durchgeführt werden:

1. Zunächst wird ein Satz von Stellgrößen für den aktuellen und $N-1$ zukünftige Zeitpunkte erzeugt, wobei nur der erste tatsächlich angewendet werden würde. Die restlichen Stellgrößen dienen nur zur Erzeugung der Vorhersage, wie sich das System in Zukunft verhalten würde, wenn man diese Stellgrößen anwendet. Bei dem von Diedam et al. [11] vorgeschlagenen Algorithmus handelt es sich bei den Stellgrößen um die erste Ableitung der Beschleunigung und die Fußpositionen.

2. Mit Hilfe des 3D-LIPM, wie auch von Diedam et al. vorgeschlagen, wird aus diesen Stellgrößen eine Vorhersage des ZMP- und Schwerpunktverlaufs erzeugt. Vereinfacht dargestellt wird hier Schritt für Schritt die Stellgröße auf das System angewendet, um dessen Zustand zu den Zeitpunkten $t_0+1, t_0+2, \ldots$ zu bestimmen.

3. Die Vorhersage wird bewertet. Dazu müssen zuvor Ungleichheitsbedingungen aufgestellt und eine Kostenfunktion definiert werden, welche zu minimieren ist. Als Beispiel ist hier ein Summand der Zielfunktion die Summe der Differenzen zwischen der vorhergesagten CoM-Position und der gewünschten CoM-Position zu den Zeitpunkten $t_0+1, \ldots, t_0+N$, wodurch der Schwerpunkt der Laufrichtung folgt.

   Zudem ist ein wichtiger Teil der Ungleichheitsbedingung, dass der ZMP nicht das Stützpolygon verlassen darf. In Abbildung 3.1 sind das während der Double-Support-Phase die gelben Flächen und während der Single-Support-Phase die grünen.

4. Sind die Bedingungen erfüllt und ist bezüglich der Kostenfunktion ein lokales Minimum erreicht, so wird die Stellgröße des aktuellen Zeitpunktes tatsächlich angewendet. Wenn nicht, beginnt der Vorgang wieder bei 1.

Dieser Ablauf dient nur dem Verständnis. Die tatsächliche Regelung kann auf verschiedene Weisen erfolgen. Da für humanoides Laufen von Robotern mit Hilfe des 3D-LIPM die Darstellung als Zustandsraum nahe liegt und zudem Ungleichheitsbedingungen benötigt werden, um den ZMP auf das Stützpolygon zu beschränken, ist eine iterative Optimierung notwendig.

Diedam et al. [11] schlagen dazu die quadratische Programmierung (QP) [6] vor. So wird ein Optimierungsproblem bezeichnet, bei dem die Zielfunktion und die Nebenbedingungen eine bestimmte Form aufweisen. Im nächsten Abschnitt wird zunächst die Zielfunktion aufgestellt, worauf im Anschluss die Ungleichheitsbedingungen formuliert werden.

3.1.1 Formulierung der Zielfunktion

Ziel dieses Abschnittes ist eine Zielfunktion in einer für ein quadratisches Programm geeigneten Form:

$$\min_{\mathbf{u}_k} \frac{1}{2}\mathbf{u}_k^T \mathbf{Q}\mathbf{u}_k + \mathbf{b}_k^T \mathbf{u}_k \tag{3.1}$$

wobei $\mathbf{Q}$ eine symmetrische Matrix ist, $\mathbf{b}_k^T$ ein reeller Vektor und $\mathbf{u}_k = \left(\ddot{\mathbf{X}}_k, \mathbf{F}_k\right)^T$ die Stellgröße zum Zeitpunkt k ist. Sie besteht aus der aktuellen und $N-1$ künftigen Änderungen der CoM-Beschleunigung $\ddot{\mathbf{X}}_k = \left(\dddot{\mathbf{c}}_k^x, \ldots, \dddot{\mathbf{c}}_{k+N-1}^x\right)^T$ und, da auch die Fußpositionen optimiert werden sollen, zudem aus M Fußpositionen $\mathbf{F}_k = \left(\mathbf{f}_k^x, \ldots, \mathbf{f}_{k+M-1}^x\right)^T$, wobei eine Fußposition für die Dauer eines Schrittes gültig ist. Die hochgestellten x bezeichnen die x-Achse, die hier für alle Gleichungen und Herleitungen verwendet wird, wobei diese auch analog für die y-Achse gelten und anwendbar sind. Zudem bedeuten Punkte über einer Variablen, dass es sich hierbei um die entsprechende Geschwindigkeit bzw. Beschleunigung handelt, jedoch nicht, dass es sich um einen zeitlich kontinuierlich differenzierten Wert handelt.
Im Folgenden wird die Matrix $\mathbf{Q}$ sowie der Vektor $\mathbf{b}_k^T$ hergeleitet, so dass die folgenden Ziele erreicht werden:

- Der CoM soll einem Pfad folgen. Das lässt sich in der Zielfunktion ausdrücken als die Differenz zwischen einer Referenz-CoM-Trajektorie, abgeleitet aus den Referenz-Fußpositionen, und der tatsächlichen.
- Die Fußpositionen sollen ebenfalls minimal von der Referenz $\mathbf{F}_k^{ref} = \left(\mathbf{f}_k^{ref,x}, \ldots, \mathbf{f}_{k+M-1}^{ref,x}\right)^T$ abweichen, so dass eine Modifikation nur dann vorgenommen wird, wenn der Roboter ansonsten instabil werden würde.
- Die CoM-Geschwindigkeit soll so gering wie möglich sein, solange der CoM der Referenz folgt.
- Die Änderungen der Beschleunigungen sollen minimal sein, da variierende Kräfte und Drehmomente den Lauf destabilisieren können.

Zur Herleitung wird zunächst eine Beschreibung des Systems benötigt. Die Dynamik, basierend auf dem 3D-LIPM, lässt sich mit einem Zustandsraummodell beschreiben, welches sich hier von anderen Arbeiten, wie zum Beispiel von Kajita et al. [24], kaum unterscheidet:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\dddot{\mathbf{c}}_k^x, \tag{3.2}$$

wobei $\mathbf{A}$ die Systemmatrix und $\mathbf{B}$ der Eingangsvektor ist. Diese Gleichung enthält nicht die Fußpositionen, ist aber ausreichend für die folgenden Herleitungen. Als Zustand wird hier die Position des CoM sowie dessen Geschwindigkeit und Beschleunigung genutzt:

$$\mathbf{x}_k = (\mathbf{c}_k^x, \dot{\mathbf{c}}_k^x, \ddot{\mathbf{c}}_k^x)^T, \tag{3.3}$$

so dass für $\mathbf{A}$ und $\mathbf{B}$ gilt:

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \frac{\Delta t^3}{6} \\ \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}. \tag{3.4}$$

Die Definitionen des Zustandes, des Zustandsraummodells und der dafür verwendeten Matrizen sind nur für diesen Abschnitt gültig, da für den MPC-Ansatz der Zustand die Beschleunigung des Schwerpunktes enthält. Sie werden in den folgenden Abschnitten neu definiert. Aus dieser Systembeschreibung gilt es nun die Vorhersage für den Zustand zu den Zeitpunkten $k+1, \ldots, k+N$ herzuleiten, indem Gleichung 3.2 mehrfach rekursiv angewendet wird, so dass sich folgende Gleichungen für die Vorhersage der CoM-Positionen $\mathbf{X}_{k+1}$ ergeben (siehe Anhang B):

$$\mathbf{X}_{k+1} = \mathbf{A_P}\mathbf{x}_k + \mathbf{B_P}\dddot{\mathbf{X}}_k, \tag{3.5}$$

$$\mathbf{A_P} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ & \vdots & \\ 1 & N\Delta t & \frac{(N\Delta t)^2}{2} \end{bmatrix}, \tag{3.6}$$

$$\mathbf{B_P} = \begin{bmatrix} \frac{1}{6}\Delta t^3 & 0 & \ldots & 0 \\ \left(1+\frac{1}{6}\right)\Delta t^3 & \frac{1}{6}\Delta t^3 & \ldots & 0 \\ \left(3+\frac{1}{6}\right)\Delta t^3 & \left(1+\frac{1}{6}\right)\Delta t^3 & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ \left(\frac{(N-1)N}{2}+\frac{1}{6}\right)\Delta t^3 & \left(\frac{(N-2)(N-1)}{2}+\frac{1}{6}\right)\Delta t^3 & \ldots & \frac{1}{6}\Delta t^3 \end{bmatrix}, \tag{3.7}$$

wobei $\mathbf{X}_{k+1}$ ein Vektor mit den zukünftigen CoM-Positionen (entlang der x-Achse) ist:

$$\mathbf{X}_{k+1} = \begin{bmatrix} \mathbf{c}_{k+1}^x \\ \vdots \\ \mathbf{c}_{k+N}^x \end{bmatrix}. \tag{3.8}$$

Da auch der Zustand des Systems $\mathbf{x}$ die CoM-Position enthält, ähnelt seine Bezeichnung der dieses Vektors, ist aber nicht mit diesem zu verwechseln, da $\mathbf{X}_{k+1}$ neben der CoM-Position zum Zeitpunkt $k+i$ auch die $N-1$ zukünftigen enthält, jedoch nicht die Geschwindigkeiten, welche in einem anderen Vektor $\dot{\mathbf{X}}_{k+1}$ zusammengefasst werden:

$$\dot{\mathbf{X}}_{k+1} = \mathbf{A_S}\mathbf{x}_k + \mathbf{B_S}\dddot{\mathbf{X}}_k, \tag{3.9}$$

$$\mathbf{A_S} = \begin{bmatrix} 0 & 1 & \Delta t \\ & \vdots & \\ 0 & 1 & N\Delta t \end{bmatrix}, \tag{3.10}$$

$$\mathbf{B_S} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & \dots & 0 \\ \left(1+\frac{1}{2}\right)\Delta t^2 & \frac{1}{2}\Delta t^2 & \dots & 0 \\ \left(2+\frac{1}{2}\right)\Delta t^2 & \left(1+\frac{1}{2}\right)\Delta t^2 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ \left(N-1+\frac{1}{2}\right)\Delta t^2 & \left(N-2+\frac{1}{2}\right)\Delta t^2 & \dots & \frac{1}{2}\Delta t^2 \end{bmatrix}, \tag{3.11}$$

mit

$$\dot{\mathbf{X}}_{k+1} = \begin{bmatrix} \dot{\mathbf{c}}^x_{k+1} \\ \vdots \\ \dot{\mathbf{c}}^x_{k+N} \end{bmatrix}. \tag{3.12}$$

An dieser Stelle sei auch die Vorhersage der ZMP-Positionen $\mathbf{P}_{k+1}$ gezeigt, welche allerdings nicht für die Zielfunktion, sondern erst für die Ungleichheitsbedingungen benötigt werden:

$$\mathbf{P}_{k+1} = \mathbf{A_Z}\mathbf{x}_k + \mathbf{B_Z}\dddot{\mathbf{X}}_k, \tag{3.13}$$

$$\mathbf{A_Z} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} - \frac{\mathbf{c}^z}{g} \\ & \vdots & \\ 1 & N\Delta t & \frac{(N\Delta t)^2}{2} - \frac{\mathbf{c}^z}{g} \end{bmatrix}, \tag{3.14}$$

$$\mathbf{B_Z} = \begin{bmatrix} \frac{1}{6}\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t & 0 & \dots & 0 \\ (1+\frac{1}{6})\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t & \frac{1}{6}\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t & \dots & 0 \\ (3+\frac{1}{6})\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t & (1+\frac{1}{6})\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ \left(\frac{(N-1)N}{2}+\frac{1}{6}\right)\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t & \left(\frac{(N-2)(N-1)}{2}+\frac{1}{6}\right)\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t & \dots & \frac{1}{6}\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t \end{bmatrix}. \tag{3.15}$$

Es sind nun alle nötigen Matrizen vorhanden, um die folgende Zielfunktion aufzustellen. Sie ist jedoch nur als Zwischenschritt anzusehen, da sie noch nicht in der Form vorliegt, die für ein quadratisches Programm benötigt wird:

$$\min_{\mathbf{u}_k} \frac{1}{2}|\dddot{\mathbf{X}}_k|^2 + \frac{\alpha}{2}|\dot{\mathbf{X}}_{k+1}|^2 + \frac{\beta}{2}|\mathbf{X}_{k+1} - \mathbf{X}^{ref}_{k+1}|^2 + \frac{\gamma}{2}|\mathbf{F}_k - \mathbf{F}^{ref}_k|^2. \tag{3.16}$$

Als eine Vereinfachung bei der Erstellung der Referenzen kann die CoM-Referenz $\mathbf{X}_{k+1}^{ref}$ in der Mitte der jeweiligen Stützpolygone gewählt werden, so dass diese in direktem Zusammenhang mit der gewählten Fußpositionsreferenz $\mathbf{F}_k^{ref}$ stehen[1]:

$$\mathbf{X}_{k+1}^{ref} = \mathbf{U}_{k+1}\mathbf{F}_k^{ref}, \tag{3.17}$$

wobei Matrix $\mathbf{U}_{k+1}$ in jeder Zeile einem Zeitpunkt eine Fußreferenzposition zuordnet:

$$\mathbf{U}_{k+1} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}}_{M} \begin{matrix} \left.\vphantom{\begin{matrix} 1 \\ \vdots \\ 1 \end{matrix}}\right\} n_1 \\ \left.\vphantom{\begin{matrix} 1 \\ \vdots \\ 1 \end{matrix}}\right\} n_2 \\ \left.\vphantom{\begin{matrix} 1 \\ \vdots \\ 1 \end{matrix}}\right\} N - n_1 - n_2 \end{matrix} \tag{3.18}$$

Dies ist notwendig, da in $\mathbf{F}_k^{ref}$ jeder Schritt nur einen Eintrag hat, egal wie viele Zeitschritte er dauern soll, $\mathbf{X}_{k+1}^{ref}$ jedoch für jeden Zeitschritt einen Eintrag für die Referenzposition benötigt. Bei $\mathbf{X}_{k+1}^{ref}$ handelt es sich demnach um die Referenzpositionen für die Füße, jedoch dargestellt als Vektor mit einer Referenzposition pro Zeitschritt.
Welche Fußreferenzposition einem Zeitpunkt zugeordnet wird ist frei wählbar. Die hier dargestellte Matrix $\mathbf{U}_{k+1}$ stellt die hier angewendete Möglichkeit dar. Die Länge der einzelnen Schritte (in Zeitschritten) ist fest gewählt und hier mit n_2 bezeichnet. Der Inhalt der Matrix rückt in jedem Zeitschritt um eine Zeile nach oben und wird mit der Zeile für die Zuordnung zum Zeitpunkt N aufgefüllt. Daher ist die Anzahl n_1 der verbleibenden Zeitschritte im aktuellen Schritt höchstens so groß wie n_2, wobei alle zusammen N Zeilen ergeben müssen.
Die Zielfunktion 3.16 kann nun durch Einsetzen der Gleichungen 3.5, 3.9 und nachfolgender Vereinfachung in die gewünschte Form von Gleichung 3.1 überführt werden. Somit gilt für $\mathbf{Q}$ und $\mathbf{b}_k^T$:

$$\mathbf{Q} = \begin{bmatrix} \mathbf{I} + \alpha {\mathbf{B_S}}^T \mathbf{B_S} + \beta {\mathbf{B_P}}^T \mathbf{B_P} & 0 \\ 0 & \gamma \mathbf{I} \end{bmatrix}, \tag{3.19}$$

$$\mathbf{b}_k = \begin{bmatrix} \alpha {\mathbf{x}_k}^T {\mathbf{A_S}}^T \mathbf{B_S} + \beta {\mathbf{x}_k}^T {\mathbf{A_P}}^T \mathbf{B_P} - \beta {\mathbf{F}_k^{ref}}^T \mathbf{U}_{k+1}^T \mathbf{B_P} \\ -\gamma \mathbf{F}_k^{ref} \end{bmatrix}. \tag{3.20}$$

[1] Zur Vereinfachung wird hier keine Double-Support-Phase durchgeführt.

Dieses Ergebnis stimmt mit den Ergebnissen von Diedam et al. [11] überein. Die genaue Herleitung ist dort jedoch nicht zu entnehmen und befindet sich in Anhang A.
Die Zielfunktion umfasst wichtige Aspekte, wie die Verfolgung eines Pfades und die Einhaltung der vorgegebenen Fußpositionen, sofern möglich. Es gibt allerdings weitere wichtige Punkte, die durch Ungleichheitsbedingungen besser gelöst werden können, welche Gegenstand des nächsten Abschnittes sind.

3.1.2 Formulierung der Ungleichheitsbedingungen

Häufig werden die Komponenten der Zielfunktion als weiche Bedingungen bezeichnet, da man hier Ziele verfolgt, die nicht strikt eingehalten werden müssen. Insbesondere beim humanoiden Laufen gibt es allerdings Bedingungen, die keinesfalls verletzt werden dürfen, und daher als eine Gleichheits- oder Ungleichheitsbedingung zu formulieren sind:

- Der ZMP muss innerhalb des Stützpolygons liegen, welches von der aktuellen Laufphase und den gewählten (und möglicherweise modifizierten) Fußpositionen abhängt.
- Die Fußposition des aktuellen Schrittes darf nicht modifiziert werden, da der Fuß Kontakt zum Boden hat. Die Modifikation der zukünftigen Schritte muss auf ein physisch realisierbares Maß begrenzt werden.

Im Folgenden sind diese Beschränkungen als Ungleichheitsbedingungen zu formulieren, welche in folgender Form vorliegen müssen:

$$\mathbf{h}_k^l \leq \quad \mathbf{u}_k \quad \leq \mathbf{h}_k^u, \tag{3.21}$$

$$\mathbf{h}_k^{A,l} \leq \quad \mathbf{D}_k\mathbf{u}_k \quad \leq \mathbf{h}_k^{A,u}. \tag{3.22}$$

Die Vektoren $\mathbf{h}_k^l, \mathbf{h}_k^u, \mathbf{h}_k^{A,l}, \mathbf{h}_k^{A,u}$ und die Matrix $\mathbf{D}_k$ können beliebig gewählt werden, wobei die obige Darstellung bedeutet, dass die Vektoren zeilenweise miteinander verglichen werden.

ZMP-Bedingung

Für die ZMP-Bedingung muss zunächst der ZMP vorhergesagt werden, wie in Gleichung 3.13 dargestellt. Der so berechnete ZMP ist im Weltkoordinatensystem und muss in das Koordinatensystem des Standfußes konvertiert werden, indem der jeweilige Standfuß subtrahiert wird, siehe Gleichung 3.17. Es ergibt sich der ZMP im Fußkoordinatensystem, welcher nicht größer bzw. kleiner sein darf als die Distanz d^Z der vorderen bzw. hinteren Fußkante zum Ursprung. Es ergibt sich folgende Ungleichung:

$$-d^Z \leq \mathbf{P}_{k+1} - \mathbf{U}_{k+1}\mathbf{F}_k \leq d^Z \tag{3.23}$$

$$\Leftrightarrow \quad -d^Z \leq \mathbf{A_Z}\mathbf{x}_k + \mathbf{B_Z}\dddot{\mathbf{X}}_k - \mathbf{U}_{k+1}\mathbf{F}_k \leq d^Z \tag{3.24}$$

Diese Ungleichung ist in Abbildung 3.1 dargestellt als gelbe bzw. grüne Flächen, die das Stützpolygon darstellen und als erlaubte Fläche für den vorhergesagten ZMP (rote Linie) dienen.
Dieses Ergebnis lässt sich wie folgt in die Form von Ungleichung 3.22 wandeln:

$$\mathbf{D}_k = [\mathbf{B_Z}, -\mathbf{U}_{k+1}], \tag{3.25}$$
$$\mathbf{h}_k^{A,l} = -d^Z - \mathbf{A_Z}\mathbf{x}_k, \tag{3.26}$$
$$\mathbf{h}_k^{A,u} = d^Z - \mathbf{A_Z}\mathbf{x}_k. \tag{3.27}$$

Die Dimension von $\mathbf{D}_k$ ist somit $N \times N + M$.

Fußpositionsbedingung

Verschiedene Gründe limitieren die physischen Möglichkeiten des Roboters bei der Ausführung von Schritten, beispielsweise das maximale Drehmoment und die maximale Geschwindigkeit der Motoren, die maximale Länge des Beines usw. Diese Tatsache muss auch bei der Modifikation der geplanten Schritte beachtet werden.
Diedam et al. schlagen vor [11], die Position eines Fußes relativ zum anderen zu beschränken. Zudem wird die Geschwindigkeit begrenzt, die für den Schwungfuß im aktuellen Schritt nötig wäre, um die modifizierte Zielposition zu erreichen.
Im Gegensatz dazu wird hier die maximale Modifikation begrenzt, was die Geschwindigkeit des Schwungfußes im aktuellen Schritt, aber auch aller künftigen beschränkt. Zudem können so gleichzeitig Selbstkollisionen oder physisch nicht erreichbare Positionen vermieden werden ohne zusätzliche Ungleichheitsbedingungen aufzustellen.
Zur Beschränkung der maximalen Modifikation stehen alle nötigen Werte in $\mathbf{u}_k$ zur Verfügung, so dass die Beschränkung in Form von Gleichung 3.21 aufgestellt werden kann mit:

$$\mathbf{h}_k^l = \left(-\infty, \ldots, -\infty, \mathbf{F}_k^{ref} + \mathbf{d}_k^{F,l}\right)^T, \tag{3.28}$$
$$\mathbf{h}_k^u = \left(\infty, \ldots, \infty, \mathbf{F}_k^{ref} + \mathbf{d}_k^{F,u}\right)^T. \tag{3.29}$$

Die einzelnen Vektorelemente ergeben sich wie folgt. Wie in Unterabschnitt 3.1.1 beschrieben stehen die Elemente $1, \ldots, N$ von $\mathbf{u}_k$ für die geplanten Beschleunigungen, welche hier nicht beschränkt werden sollen. Die entsprechenden Elemente in $\mathbf{h}_k^l$ und $\mathbf{h}_k^u$ sind daher mit einem möglichst kleinen bzw. großen Wert zu belegen, hier als $-\infty$ bzw. ∞ beschrieben und implementiert mit den kleinsten bzw. größten darstellbaren Zahlen.
Des Weiteren gilt:

$$\mathbf{d}_k^{F,u} = \left(-\mathbf{f}_k^{ref,x} + \mathbf{f}_k^x + \epsilon, d_{k+1}^{F,u}, \ldots, d_{k+M-1}^{F,u}\right)^T, \tag{3.30}$$
$$\mathbf{d}_{k+1}^{F,l} = \left(-\mathbf{f}_k^{ref,x} + \mathbf{f}_k^x - \epsilon, d_{k+1}^{F,l}, \ldots, d_{k+M-1}^{F,l}\right)^T, \tag{3.31}$$

für die Vektoren der maximal bzw. minimal erlaubten Abweichung von der Referenz. Die Werte sind primär abhängig davon, ob sich aufgrund der Möglichkeit von Kollisionen der beiden Füße andere mögliche Modifikationen ergeben. So ist es entlang der y-Achse kaum möglich, dass die Beine sich kreuzen. Entlang der x-Achse dienen die Beschränkungen der Vermeidung von unerreichbaren Positionen.

Für den aktuellen Schritt gilt jedoch, dass er nicht weiter modifiziert werden darf, da er aktuell den Roboter stützt. Daher gilt hier, dass er höchstens um ein $\epsilon > 0$ (hier $0.1\,\mathrm{mm}$) von der zuletzt geplanten Position abweichen darf. Dieser Wert wird einmal in dem Moment bestimmt, wo er zum Standfuß wurde.

3.2 Bewegungssteuerung mittels Preview Control

In seinen Eigenschaften ist Preview Control (PC) dem Model Predictive Control (MPC) ähnlich. Das zu regelnde System wird mit einem Zustandsraummodell beschrieben und die Ziele mit Hilfe einer Zielfunktion definiert. Zudem wird beim PC eine Vorschau des gewünschten Verlaufs der Regelgröße verwendet. Mathematisch gesehen basiert PC allerdings auf den linear-quadratischen Reglern, weshalb die Art und Weise, wie der Regler den optimalen Verlauf der Stellgröße bestimmt, sich wesentlich von MPC unterscheidet. Es lassen sich keine Beschränkungen definieren, so dass der gewünschte Verlauf des ZMP im Vorfeld anhand der gewünschten Fußpositionen festzulegen ist. Dieses Vorgehen hat wesentliche Vorteile, da durch harte Bedingungen Lösungen nicht immer möglich sind und eine iterative Suche auch fehlschlagen kann, selbst wenn eine Lösung existiert, da aufgrund der beschränkten Zeit nicht alle Möglichkeiten betrachtet werden können. Da der PC Ansatz ohne solche Bedingungen auskommt, wird eine Lösung immer gefunden, was bereits vor der Implementierung auch als potentiell Echtzeitfähig eingestuft werden kann.

In diesem Abschnitt wird eine Version der Laufsteuerung beschrieben, wie sie von Kajita et al. vorgeschlagen wird [25], wobei der dazu verwendete Regler von Katayama et al. [26] stammt. Die Herleitungen und Beweise sind in den jeweiligen Arbeiten zu finden.

Aus Gründen der Übersichtlichkeit werden die Bezeichnungen aus dem vorherigen Abschnitt hier neu definiert und nicht gesondert gekennzeichnet. Wie auch im vorherigen Abschnitt bezeichnen hochgestellte x die x-Achse, wenn auch das Folgende auf die y-Achse angewendet werden kann. Punkte über den Variablen stellen die entsprechenden Geschwindigkeiten bzw. Beschleunigungen als diskrete Werte dar und besagen nicht, dass es sich um zeitlich kontinuierlich differenzierte Werte handelt.

Für PC wird das Zustandsraummodell minimal anders definiert als in Gleichung 3.2:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u_k, \tag{3.32}$$

mit $u_k = \dot{\mathbf{p}}_k^x$. Der Zustand für das Zustandsraummodell besteht aus der Position und Geschwindigkeit des Schwerpunktes und der Position des ZMP:

$$\mathbf{x}_k = (\mathbf{c}_k^x, \dot{\mathbf{c}}_k^x, \mathbf{p}_k^x)^T, \tag{3.33}$$

so dass bezüglich Gleichung 3.32 für $\mathbf{A}$ und $\mathbf{B}$ gilt:

$$\mathbf{A} = \begin{bmatrix} 1 & \Delta t & 0 \\ \frac{g}{\mathbf{c}^z}\Delta t & 1 & -\frac{g}{\mathbf{c}^z}\Delta t \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \Delta t \end{bmatrix}. \tag{3.34}$$

Als Regelgröße wird der ZMP gewählt:

$$y_k = \mathbf{C}\mathbf{x}_k, \tag{3.35}$$
$$\mathbf{C} = (0,0,1), \tag{3.36}$$

weswegen der ZMP auch die Referenz ist, zu der der tatsächliche ZMP eine möglichst geringe Differenz haben soll. Das spiegelt sich in der Zielfunktion wieder, die vom Regler zu minimieren ist:

$$J = \sum_{j=0}^{\infty} \left\{ Q_e \left[\mathbf{p}_j^x - p_j^{ref} \right]^2 + \Delta \mathbf{x}_j^T Q_x \Delta \mathbf{x}_j + R \Delta u_j^2 \right\}, \tag{3.37}$$

wobei Q_e, Q_x und R plattformabhängige Parameter sind und händisch gewählt werden. Delta (Δ) symbolisiert hier die Änderung der Größe im Vergleich zum letzten Zeitpunkt. Der erste Summand minimiert die Differenz zwischen der ZMP-Referenz p_j^{ref} und dem geschätzten ZMP, der dritte die Änderungen in der Stellgröße. Der zweite Summand wird von Kajita et al. [25] nicht vorgeschlagen und orientiert sich an der ursprünglichen Variante von Katayama et al. [26]. Wie bereits in Abschnitt 1.2, „Anforderungen“ diskutiert, sind zur Minimierung von Oberkörperschwankungen niedrige Geschwindigkeiten und Beschleunigungen erstrebenswert, was durch diesen Summanden in Bezug auf den Schwerpunkt realisiert wird. Die Stellgröße, welche die Zielfunktion minimiert, ist für den Zeitpunkt k gegeben als:

$$u_k = -G_I \sum_{i=0}^{k} \left[\mathbf{C}\mathbf{x}_i - p_i^{ref} \right] - \mathbf{G}_x \mathbf{x}_k - \sum_{j=1}^{N} G_{d,j} p_{k+j}^{ref}, \tag{3.38}$$

wobei die Werte G_I , $\mathbf{G}_x$ und $G_{d,j}$ wie folgt zu wählen sind [26]:

$$G_I = \left[R + \tilde{\mathbf{B}}^T \mathbf{P} \tilde{\mathbf{B}}\right]^{-1} \tilde{\mathbf{B}}^T \mathbf{P} \mathbf{I}, \tag{3.39}$$

$$\mathbf{G}_x = \left[R + \tilde{\mathbf{B}}^T \mathbf{P} \tilde{\mathbf{B}}\right]^{-1} \tilde{\mathbf{B}}^T \mathbf{P} \mathbf{F}, \tag{3.40}$$

$$G_{d,j} = -\left[R + \tilde{\mathbf{B}}^T \mathbf{P} \tilde{\mathbf{B}}\right]^{-1} \tilde{\mathbf{B}}^T \left[\tilde{\mathbf{A}}_c^T\right]^j \mathbf{P} \mathbf{I}, j \in \mathbb{N}, \tag{3.41}$$

$$\tilde{\mathbf{B}} = \begin{bmatrix} \mathbf{CB} \\ \mathbf{B} \end{bmatrix}, \mathbf{I} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \mathbf{F} = \begin{bmatrix} \mathbf{CA} \\ \mathbf{A} \end{bmatrix}, \tag{3.42}$$

$$\mathbf{Q} = \begin{bmatrix} Q_e & 0 \\ 0 & \mathbf{C}^T Q_x \mathbf{C} \end{bmatrix}, \tilde{\mathbf{A}} = [\mathbf{I}, \mathbf{F}], \tag{3.43}$$

$$\tilde{\mathbf{A}}_c = \tilde{\mathbf{A}} - \tilde{\mathbf{B}} \left[R + \tilde{\mathbf{B}}^T \mathbf{P} \tilde{\mathbf{B}}\right]^{-1} \tilde{\mathbf{B}}^T \mathbf{P} \tilde{\mathbf{A}}. \tag{3.44}$$

Dabei ist $\mathbf{P}$ die Lösung der zeitdiskreten Matrix-Riccati-Gleichung:

$$\mathbf{P} = \tilde{\mathbf{A}}^T \mathbf{P} \tilde{\mathbf{A}} - \tilde{\mathbf{A}}^T \mathbf{P} \tilde{\mathbf{B}} \left[R + \tilde{\mathbf{B}}^T \mathbf{P} \tilde{\mathbf{B}}\right]^{-1} \tilde{\mathbf{B}}^T \mathbf{P} \tilde{\mathbf{A}} + \mathbf{Q}. \tag{3.45}$$

Es ist zu erkennen, dass der Regler (Gleichung 3.38) einen Integralanteil (erster Summand), Proportionalanteil (zweiter Summand) und einen Vorschauanteil (dritter Summand) besitzt, wobei ersteres von Kajita et al. ebenfalls nicht vorgeschlagen wird, aber speziell im Hinblick auf die Rückführung von Sensordaten von Vorteil ist, da insbesondere bei langsamen Bewegungen wie Stehen auf einem Bein so auch kleinere gemessene Positionsabweichungen im Schwerpunkt auf längere Sicht ausgeglichen werden.
Des Weiteren ist zu diskutieren, dass die Summe, mit der die Vorschau im Regler einfließt, endlich ist, während die Zielfunktion eine unendliche Vorschau betrachtet. Wie Abbildung 3.2 zeigt, ist eine unendliche Summe auch nicht notwendig. Während der Fehler im tatsächlichen ZMP im Vergleich zum gewünschten ZMP bei $N = 20$ und $N = 40$ noch deutlich sichtbar ist, reicht bereits $N = 60$ aus um kaum noch wahrnehmbare Abweichungen zu ermöglichen. Die Lösungen der Gleichungen 3.39 bis 3.45 können vor dem Lauf bestimmt werden, so dass währenddessen keine iterativ bestimmten Lösungen gefunden werden müssen.

3.3 Zusammenfassung

Dieses Kapitel zeigt zwei regelungstechnische Lösungen zu einem gewünschten Lauf eine Oberkörperbewegung zu bestimmen, die nach dem 3D-LIPM stabil ist. Beide gezeigten Regler arbeiten weitgehend wie von Diedam et al. [11] (MPC) bzw. von Kajita et al. vorgeschlagen. Unterschiede finden sich bei MPC in der Beschränkung der möglichen Fußpositionsmodifikation, welche hier vereinfacht umgesetzt wird und bei PC in der Zielfunktion und dem Regler,

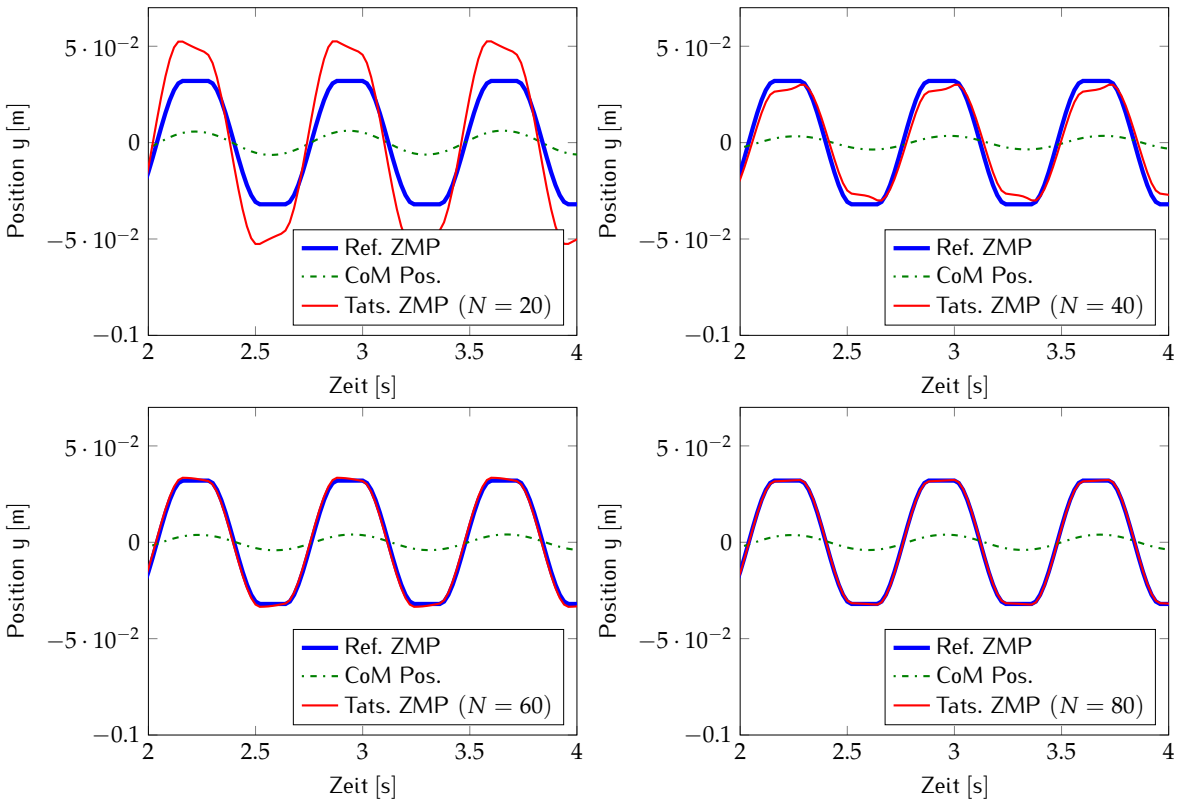

Abbildung 3.2: Übereinstimmung von Referenz-ZMP und tatsächlichem ZMP bei verschieden groß gewählter Länge der Vorschau.

welcher um einen Integralteil erweitert wurde, wie auch ursprünglich von Katayama et al. vorgeschlagen.
Während PC mit deutlich weniger Gleichungen auskommt, ist der MPC-Ansatz bereits jetzt in der Lage, Ausfallschritte zu planen. Diese Möglichkeit wird zum PC-Ansatz in Kapitel 4.3 hinzugefügt. In der Evaluation wird insbesondere der Nachteil des MPC der höheren Laufzeit aufgrund der Lösung des quadratischen Programms untersucht. Es wird sich zeigen, dass der Ansatz nicht echtzeitfähig nach Definition dieser Arbeit ist.
Zunächst wird im folgenden Kapitel eine Möglichkeit vorgestellt, den Zustand eines Systems mit Hilfe von Sensoren zu messen, ohne dass für jede einzelne Komponente des Zustandes geeignete Sensoren vorhanden sein müssen.

KAPITEL 4

Bewegungsregelung

Die Rückführung von Sensordaten stellt in vielerlei Hinsicht eine Notwendigkeit dar, um einen praxistauglichen Lauf zu entwickeln. Während diese Tatsache unbestritten ist, zeigt Kapitel 2, „Stand der Forschung“, dass die Forschung in diesem Bereich noch unterentwickelt ist. Speziell im Bereich Laufen mittels Regler sind die Methoden zwar auf Rückführung von Sensordaten vorbereitet, die tatsächliche Art und Weise, wie dies nachweisbar zu Verbesserungen führt, ist allerdings noch nicht behandelt worden.

Dieses Kapitel beschäftigt sich daher mit Sensorkontrolle auf Basis eines Beobachters und der Erweiterung des PC-Ansatzes durch Ausfallschritte.

4.1 Sensorquellen

Humanoide Roboter haben üblicherweise eine Vielzahl von Sensoren, wobei hier als Beispiel der humanoide Roboter „Nao“ von Aldebaran Robotics dient. Mit vergleichbarer Sensorik ausgestattet sind der „DARwIn-OP“ von Robotis und der humanoide Laufroboter des Deutschen Luft- und Raumfahrtzentrums [38].

Eine der wichtigsten Sensortypen ist die Inertial Measurement Unit (IMU). Sie besteht für gewöhnlich aus einem Beschleunigungssensor und einem Gyroskop, welche mittels Sensorfusion für die Orientierungsmessung genutzt werden. Daneben werden zur Unterstützung von Laufalgorithmen Force Sensing Resistor (FSR) in den Füßen angebracht. Sie können die Kraft an einem bestimmten Punkt messen, die der Boden auf den Roboter ausübt, woraus der CoP berechnet werden kann, wie weiter unten beschrieben. Zudem ist es möglich, die aktuelle Winkelposition der Gelenke zu messen, wobei die dazu verwendeten Sensoren un-

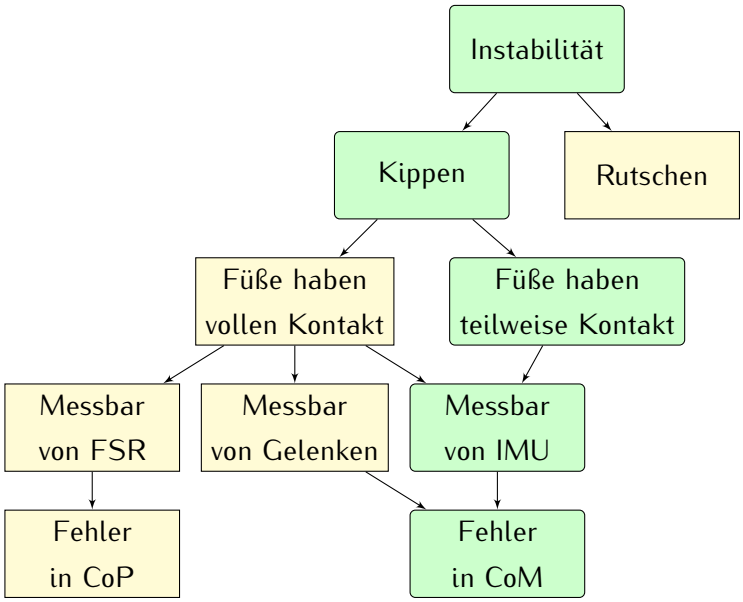

Abbildung 4.1: Übersicht über verschiedene Typen von Instabilitäten und angemessenen Messmethoden. Grün kennzeichnet den Weg zur notwendigen Messung für Ausfallschritte.

terschiedlich sein können, wie beispielsweise Potentiometer oder, eher unüblich, wie beim Nao durch Magnetic Rotary Encoder (MRE).

Von Interesse sind hier die messbaren Komponenten der Zustandsvektoren aus Kapitel 3. Abbildung 4.1 zeigt, welche Sensoren dazu in verschiedenen Situationen sinnvoll eingesetzt werden können. Diese Arbeit beschäftigt sich, wie in der Einleitung unter den Zielen (Stabilität) definiert, mit kippenden Robotern. Aufgrund von Ungenauigkeiten in den Gelenken und flexiblen Bauteilen des Roboters kann der Oberkörper gekippt sein, obwohl die gesamte Sohle Kontakt zum Boden hat. Für diesen Fall eignen sich primär FSR, mit denen sich der CoP messen lässt. Auch der Beschleunigungssensor kann genutzt werden, indem mittels 3D-LIPM aus der Beschleunigung der ZMP bestimmt wird. Jedoch wird dazu zusätzlich die gemessene Position des Schwerpunktes benötigt, so dass diese Möglichkeit hier keine Priorität hat. Eine weitere Messmöglichkeit bei vollem Kontakt zum Boden ist die Orientierung des Oberkörpers gemessen mittels der Gelenkwinkel und direkter Kinematik, da in diesem Fall der Winkel zwischen Boden und Fuß bekannt ist. Dies hat den Vorteil gegenüber einer Messung mittels Beschleunigungssensoren, dass kein Integrieren notwendig ist und gemessene Gelenkwinkel wenig verrauscht sind.

Ausfallschritte sind für den Fall vorgesehen, dass eine größere Störung vorliegt, in dem die Füße keinen vollständigen Kontakt mehr haben. Ein verbreiteter Sensor, der in der Lage ist in dieser Situation den Fehler prinzipiell korrekt zu messen, ist die IMU, wobei die Ausgabe, die Orientierung im Raum, in einen translatorischen Fehler in der Schwerpunktposition umgerechnet werden kann.

Es werden daher der CoP und die CoM-Position verwendet, um den Zustand des Roboters zu messen, was im folgenden Abschnitt durch einen Beobachter realisiert wird. Im darauf folgenden Abschnitt wird der durch den Beobachter geschätzte CoM-Fehler verwendet, um Modifikationen für die Fußpositionen zu bestimmen. Zunächst wird hier jedoch dargestellt, wie die Messungen in eine CoM- und ZMP-Position umgerechnet werden.

Schwerpunktmessung mittels Gelenkwinkel

Anhand der gemessenen Gelenkwinkel kann die Position des Gesamtschwerpunktes berechnet werden, indem zunächst die Schwerpunktpositionen in den Koordinatensystemen der jeweiligen Körper in das Fußkoordinatensystem transferiert werden, anschließend darüber die Summe gebildet und durch die Gesamtmasse geteilt wird. Der so berechnete Gesamtschwerpunkt im Fußkoordinatensystem kann anschließend in das Weltkoordinatensystem transferiert werden. Es ergibt sich die Gleichung:

$$c_k^{snr} = \mathbf{S}_{x,y} \left[\mathbf{T}_s^w (k) \left(\frac{1}{\sum_l m_l} \sum_i \mathbf{T}_{O_i}^s (\boldsymbol{\varphi}_k) \, \mathbf{c}^i m_i \right) \right]. \tag{4.1}$$

Je nachdem, ob der Beobachter für die x- oder y-Achse entworfen wird, gilt $\mathbf{S}_{x,y} = (1,0,0)$ oder $\mathbf{S}_{x,y} = (0,1,0)$. Des Weiteren gilt:

- s ist das Koordinatensystem des Standfußes,
- $\mathbf{c}^i$ ist die Position des Schwerpunktes von Körper i in dessen Koordinatensystem,
- w ist das Weltkoordinatensystem,
- O_i ist das Koordinatensystem von Körper i,
- m_i ist die Masse von Körper i,
- $\mathbf{T}_j^i$ ist die homogene Transformationsmatrix als Funktion der Winkel vom Koordinatensystem des Körpers j nach i und
- $\boldsymbol{\varphi}_k$ ist der Vektor aller gemessenen Gelenkwinkel zum Zeitpunkt k.

Wie an Gleichung 4.1 zu sehen kommen zwei Transformationsmatrizen zum Einsatz, die auf verschiedene Art bestimmt werden. Zur Transformation von einem Roboterkörper zu einem anderen können die gemessenen Gelenkwinkel genutzt werden, so dass $\mathbf{T}_j^i (\boldsymbol{\varphi}_k)$ eine Funktion von ihnen ist.
Die Matrix $\mathbf{T}_s^w$ könnte ebenfalls gemessen werden, indem zunächst die Orientierung und Position im Raum mittels IMU geschätzt wird, um danach vom Oberkörper ausgehend die Position des Fußes im Raum mittels Gelenkwinkel zu bestimmen. Jedoch wäre in diesem

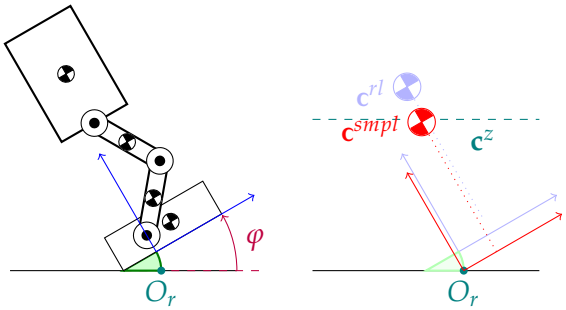

Abbildung 4.2: Näherungsweise Bestimmung der Schwerpunktposition $\mathbf{c}^{smpl}$ anhand der geschätzten Schwerpunktposition $\mathbf{c}^{rl}$, korrigiert durch die per IMU gemessene Orientierung φ.

Fall die IMU notwendig, wodurch die Vorteile der CoM-Messung ausschließlich mittels Gelenkwinkel verloren gehen würden. In Experimenten führt eine Bestimmung durch gemessene Werte zu einem sofortigen starken Zittern, so dass eine zu starke Filterung notwendig wäre, um diesen Weg nutzen zu können.
Eine andere Methode setzt daher auf nicht gemessene Werte. Für die vorgestellten Regler werden Referenz-Fußpositionen bestimmt (bei PC zur Bestimmung des Referenz-ZMP), dessen Positionen und Orientierungen im Weltkoordinatensystem vorliegen, so dass $\mathbf{T}_s^w$ sich aus diesen Sollwerten ergibt.

Schwerpunktmessung mittels IMU

Ausfallschritte sind insbesondere für Störungen prädestiniert, die den Roboter sofort zu Fall bringen können. Für sie müssen daher Daten vorliegen, die auch bei schweren Störungen noch sinnvoll sind. Die Annahme für den vorherigen Abschnitt, dass die Füße vollen Kontakt haben, ist daher nicht gültig. Trotz der Nachteile wird daher die IMU verwendet, um mittels eines vereinfachten Modells den CoM zu schätzen, welches in Abbildung 4.2 dargestellt wird. In der Abbildung links kippt der Roboter um die hintere Kante des Fußes um einen Winkel φ, so dass das Koordinatensystem O_r des Fußes um diesen Punkt rotiert werden muss. Da es ein rechenaufwändiger Vorgang wäre, den Punkt der Rotation bei einem beliebig geformten Fuß zu berechnen, wird die Rotation hier vereinfachend um den Ursprung des Koordinatensystems des Fußes durchgeführt, wie rechts in Abbildung 4.2 gezeigt. In diesem Beispiel wird die tatsächliche CoM-Position $\mathbf{c}^{rl}$ approximiert durch $\mathbf{c}^{smpl}$. Eine weitere Vereinfachung ergibt sich dadurch, dass im 3D-LIPM eine feste CoM-Höhe $\mathbf{c}^z$ angenommen wird, so dass die Höhe von $\mathbf{c}^{smpl}$ fest auf $\mathbf{c}^z$ gesetzt wird, in Abbildung 4.2 als gestrichelte Linie dargestellt.

ZMP-Messung mittels Beschleunigungssensor

Die Messung des ZMP mittels eines Beschleunigungssensors ist durch Anwendung der Gleichung 2.1 möglich [9]. Es muss dazu allerdings ebenfalls die Position des Schwerpunktes

bekannt sein, so dass, wenn man diese Gleichung für das PC-System einsetzt, die dritte Komponente (der ZMP) von der Schätzung der ersten Komponente (die CoM-Position) des Zustandsvektors abhängt. Zudem muss die Orientierung des Beschleunigungssensors bedacht werden, da für den Fall, dass die Achsen des Beschleunigungssensors nicht mit den Achsen des Weltkoordinatensystems übereinstimmen, Beschleunigungen aufgrund der Gravitation gemessen werden, die zu verfälschten Ergebnissen führen. Es ist daher mit Hilfe der gemessenen Orientierung des Oberkörpers der gemessene Beschleunigungsvektor in entgegengesetzter Richtung zu drehen, so dass die Gravitation mit der z-Achse des Sensorkoordinatensystems übereinstimmt. Als Vorteil der Messmethode mittels Beschleunigungssensor ist zu erwähnen, dass so der fZMP berechnet wird, der auch außerhalb des Stützpolygons liegen kann.

ZMP-Messung mittels FSR

Die CoP-Messung mittels FSR ist ein gewichtetes Mittel der Sensorpositionen im Fußkoordinatensystem, bei dem anhand der gemessenen Kraft gewichtet wird:

$$p_k^{snr} = \frac{\sum_i f_k^i \mathbf{r}_k^i}{\sum_i f_k^i}, \tag{4.2}$$

wobei f_k^i die gemessene Kraft und $\mathbf{r}_k^i$ die Position von Sensor i im Weltkoordinatensystem zum Zeitpunkt k ist. Die Position $\mathbf{r}_k^i$ wird analog zu $\mathbf{T}_s^w$ durch die Soll-Positionen der Füße bestimmt. Anschließend wird vereinfachend davon ausgegangen, das eine Situation vorliegt, in der der ZMP existiert und dieser daher mit dem hier gemessenen CoP übereinstimmt.

4.2 Beobachter

Im vorherigen Kapitel wurden zwei Regler vorgestellt, die auf dem Zustandsraummodell basieren. Wie der Name schon sagt, arbeiten sie mit einem Zustand, in dem sich das System befindet. Dieser ist im besten Fall gemessen, wobei es auch möglich ist, dass nicht für alle Komponenten des Zustandes Sensoren vorhanden sind, was hier aufgrund der fehlenden Messmöglichkeit der CoM-Geschwindigkeit der Fall ist. In solchen Fällen werden Beobachter eingesetzt, um die nicht messbaren Komponenten zu schätzen. Dies ist allerdings nur möglich, wenn das System beobachtbar ist, wofür verschiedene Kriterien zum Nachweis verwendet werden können.
Neben diesem Anwendungszweck hat ein Beobachter für den Einsatz beim humanoiden Laufen einen weiteren Vorteil. Die teilweise durch die Erschütterungen stark verrauschten Daten werden gefiltert, so dass beispielsweise aus dem Aufsetzen des Fußes auf dem Boden keine ruckartigen ZMP-Verläufe folgen. Das gleiche gilt allerdings auch für hochfrequente Störungen, auf die der Lauf reagieren muss.

Der nachfolgend gezeigte Beobachter kann prinzipiell für beide im vorigen Kapitel vorgestellten Regler verwendet werden. Im Folgenden wird er zur Schätzung des PC-Zustandes gezeigt, wobei eine Schätzung für MPC analog verlaufen würde. Daneben gilt auch hier, dass alle Gleichungen für die x-Achse angegeben werden, wobei das Verfahren auch analog für die y-Achse verwendet werden kann.
Es lässt sich demnach folgende Matrix aufstellen [50], die die messbaren Komponenten des Zustandes selektiert:

$$\mathbf{C}_m = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.3}$$

so dass für den Vektor $\hat{\mathbf{y}}$ der messbaren Größen gilt:

$$\hat{\mathbf{y}}_k = \mathbf{C}_m \mathbf{x}_k \hat{=} \begin{bmatrix} c_k^{snr} \\ p_k^{snr} \end{bmatrix}. \tag{4.4}$$

Das Zustandsraummodell (Gleichung 3.32) wird nun um eine Komponente $\mathbf{e}_k := \mathbf{L}\left[\hat{\mathbf{y}}_k - \mathbf{C}_m \hat{\mathbf{x}}_k\right]$ erweitert, die die Messungen in den Zustand einfließen lässt:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}\hat{\mathbf{x}}_k + \mathbf{L}\left[\hat{\mathbf{y}}_k - \mathbf{C}_m \hat{\mathbf{x}}_k\right] + \mathbf{B}\hat{u}_k. \tag{4.5}$$

Zur Verdeutlichung, dass es sich nun um einen geschätzten Zustand handelt (während zuvor der Zustand im Idealfall betrachtet wurde), wird dieser hier mit $\hat{\mathbf{x}}_k = \left(\hat{\mathbf{c}}_k^x, \hat{\dot{\mathbf{c}}}_k^x, \hat{\boldsymbol{\rho}}_k^x\right)^T$ bezeichnet, wobei die Komponenten ansonsten die gleiche Bedeutung haben. Gleiches gilt für den Regler, da dieser nun auf dem geschätzten Zustand arbeitet. Im Falle eines Beobachters für PC ist der Regler dann gegeben als:

$$\hat{u}_k = -G_I \sum_{i=0}^{k} \left[\mathbf{C}\hat{\mathbf{x}}_i - \hat{p}_i^{ref}\right] - \mathbf{G}_x \hat{\mathbf{x}}_k - \sum_{j=1}^{N} G_{d,j} \hat{p}_{k+j}^{ref}. \tag{4.6}$$

Wenn auch für die Anwendung eines Beobachters nicht notwendig, wird hier auch der Referenz-ZMP von dem Fall der Steuerung unterschieden und durch $\hat{p}_i^{ref}$ dargestellt, was im nächsten Abschnitt zum Thema Ausfallschritte von Nutzen sein wird. Ohne Ausfallschritte stimmt er mit p^{ref} überein.
Die Matrix $\mathbf{L}$ kann mit gängigen Methoden zur Entwicklung eines LQ-Reglers ermittelt werden, indem dazu zwei neue Wichtungsmatrizen und $\mathbf{A}^T$ sowie $\mathbf{C}_m^T$ eingesetzt werden [8].

4.2.1 Beispiel. Abbildung 4.3 zeigt zwei beispielhafte Störungen und die Reaktion des Reglers. Links handelt es sich dabei um eine Störung in der CoM-Position von 0.005 m bei 1.8 s. Die ZMP-Kurve zeigt diesen Fehler nahezu nicht. Die zu sehende Abweichung von der Referenz ist die Reaktion des Reglers, durch die der CoM zurück in Richtung der ursprünglichen

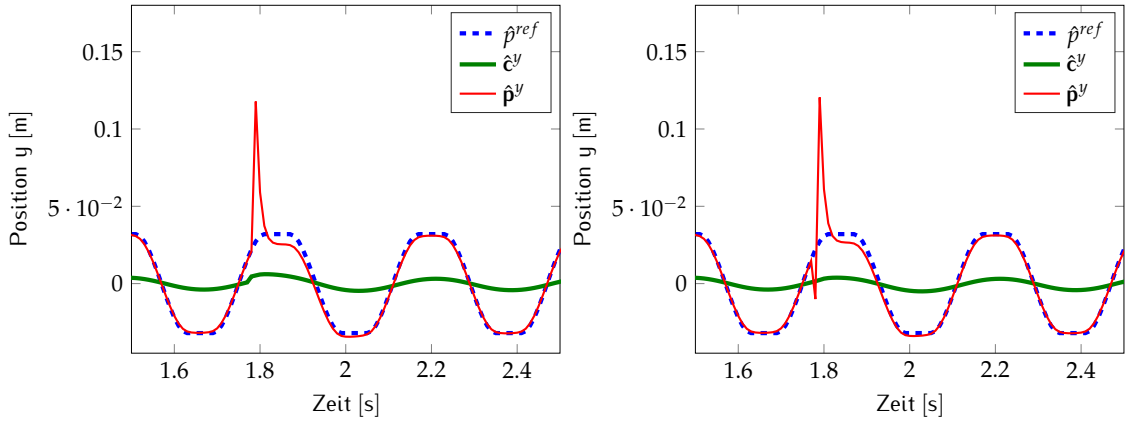

Abbildung 4.3: Beispiele für gemessene Störungen. Links eine Störung der Schwerpunktposition von 0.005 m bei 1.8 s, zu erkennen am Knick der grünen Kurve. Diese führt zu einer Reaktion des Reglers mit der Intention, die Abweichung zu korrigieren und den Schwerpunktverlauf auf seine ursprüngliche Bahn zurück zu leiten, wodurch die rote Kurve eine Spitze kurze Zeit später aufweist. Rechts ist eine Störung im gemessenen ZMP von −0.05 m ebenfalls bei 1.8 s in der roten Kurve an der Spitze in negativer Richtung zu erkennen. Auch hier führt die Reaktion des Reglers zu einer ähnlichen Spitze im ZMP in positive Richtung kurze Zeit später.

Bahn beschleunigt wird. Darauf folgend ist eine kleinere Abweichung des ZMP in die entgegengesetzte Richtung zu sehen, um den CoM abzubremsen, bis er die für den Referenz-ZMP ideale Bahn wieder erreicht hat. Auf der rechten Seite verursacht eine ZMP-Störung eine vergleichbare Reaktion, welche damit zu erklären ist, dass der ZMP-Fehler den CoM in positive Richtung beschleunigt, so dass ein ähnlicher Fehler entsteht wie links.

Dieses Beispiel dient auch als Motivation für die Ausfallschritte, welche im nächsten Kapitel vorgestellt werden.

4.3 Ausfallschritte

Das Beispiel des vorigen Abschnittes zeigt, wie eine gemessene Störung von dem PC-Regler ausgeglichen wird. Die plötzliche Beschleunigung in Richtung der optimalen Bahn führt zwar in dem Beispiel zu einer Stabilisierung. Sie zeigt allerdings auch den Grund, warum Roboter oder auch Menschen fallen: Es ist nicht die Störung selbst, sondern eine nicht angemessene Reaktion darauf, sei es, weil sie nicht korrekt bestimmt oder ausgeführt wurde oder weil eine angemessene Reaktion physisch nicht möglich ist. Möglich, aber nicht angemessen wäre beispielsweise eine ausbleibende Reaktion, oder eine Korrektur, die jedoch zu stark ist und daher selbst zu einem Kippen führt, wenn nämlich der ZMP dabei das Stützpolygon verlässt. Es ist klar, dass manche Störungen nicht auf zuvor gezeigter Weise korrigiert werden können,

ohne dass der ZMP das Stützpolygon verlassen würde. Die einzig verbleibende Möglichkeit ist das Stützpolygon selbst anzupassen, was bedeutet, die geplante Schrittfolge zu ändern.

Eine naheliegende Idee der Änderung, die Fußposition dem in Abbildung 4.3 dargestellten ZMP anzupassen, ist jedoch aus zweierlei Gründen nicht realisierbar. Erstens müsste die Anpassung sofort statt finden und zweitens ist die Störung nur kurzzeitig, so dass nach 10 ms die Anpassung wieder rückgängig gemacht werden müsste.

Idealerweise sollte daher die Abweichung durch den Regler vermieden werden. Dazu muss zunächst geklärt werden, wann genau der Regler auf die gemessene Störung reagiert. Angenommen, eine Störung tritt zum Zeitpunkt T auf und **nur** zu diesem Zeitpunkt. Dann ist an der Beobachter-Gleichung 4.5 zu erkennen, dass die Störung $\mathbf{e}_T$ den Zustandsvektor zum Zeitpunkt $T+1$ beeinflusst, welcher die Stellgröße $\hat{u}_{T+1}$ (siehe Gleichung 4.6) aufgrund dessen beeinflusst und verändert. Diese Stellgröße fließt nun in $\hat{\mathbf{x}}_{T+2}$ ein, was als Ausschlag in Abbildung 4.3 sichtbar wird.

Diese Störung bzw. Reaktion soll nun vermieden werden. Vermeiden heißt, dass der geschätzte ZMP im Zustand $\hat{\mathbf{x}}_{T+2}$ trotz gemessenen Fehlers zum Zeitpunkt T keine Abweichungen aufgrund der Störungen aufweisen soll, als wäre die Störung nicht aufgetreten ($\mathbf{e}_T = 0$). Das ist beispielsweise der Fall, wenn es keinen Rückfluss der Sensordaten gibt, also im Steuerungsfall. Dies führt zu der zentralen Forderung dieses Kapitels, welche der Ausgangspunkt zur Herleitung der Fußpositionsmodifikation ist:

$$\mathbf{C}\hat{\mathbf{x}}_{T+2} = \mathbf{C}\mathbf{x}_{T+2}. \tag{4.7}$$

In dieser Form dient die Forderung zur Vermeidung von ZMP-Abweichungen, deren Ursache in gemessenen CoM-Störungen liegt, wie zuvor in der Einleitung erläutert. Es ist auch eine Erweiterung der Forderung möglich, um auch gemessene ZMP-Fehler weitgehend auszugleichen [46]. In dieser praxisorientierten Arbeit sind, wie in Abschnitt 4.1 erwähnt, Ausfallschritte nur für größere Störungen vorgesehen, bei denen der Roboter bereits um eine Kante des Fußes kippt. In diesem Fall ist der ZMP nicht mehr definiert und eine Messung ist daher höchsten näherungsweise sinnvoll. Auf einen Ausgleich von Abweichungen im geschätzten ZMP durch gemessene ZMP-Fehler wird daher verzichtet.

Eine weitere grundlegende Idee ist nun, dass das gestörte System nicht zu seinem ursprünglichen Verlauf zurückkehren muss. Bei einer Verschiebung des Schwerpunktes kann diese auch akzeptiert werden, indem die Sollwerte um die Größe dieser Verschiebung ebenfalls verschoben werden. Aus regelungstechnischer Sicht bedeutet das, dass der Referenz-ZMP modifiziert wird, anstatt den CoM auf die ursprüngliche Bahn zurück zu lenken. Dazu wird nun $\hat{p}^{ref}$ wie folgt definiert:

$$\hat{p}_k^{ref} = p_k^{ref} + \hat{p}_k^m, \tag{4.8}$$

$$\forall k < T + t_0 : \hat{p}_k^m = 0, \tag{4.9}$$

$$\forall k \geq T + t_0 : \hat{p}_k^m = m, \tag{4.10}$$

wobei $\hat{p}_k^m$ die Modifikation der Referenz ist. Sie ist ohne Störung 0, was für alle Zeitpunkte vor T gilt. Nach der Störung ist sie zunächst vor $T + t_0$ für ein bedingt wählbares t_0 ebenfalls 0, dann konstant m. Damit lässt sich nun berechnen, wie der Referenz-ZMP modifiziert werden muss, um den vom Regler im ZMP induzierten Fehler zu vermeiden. Inwiefern t_0 gewählt werden kann, ist Teil der folgenden Herleitung [47].

4.3.1 Satz. *Gegeben sei ein impulsartiger Fehler $\boldsymbol{e}_t$ mit $\boldsymbol{e}_t = 0$ für $t \neq T$. Dann existiert eine Abbildung $S : \{2, \ldots, N+1\} \times \mathbb{R}^3 \to \mathbb{R}, (t_0, \boldsymbol{e}) \mapsto m$, welche in geschlossener Form bestimmt werden kann, so dass $\boldsymbol{C}\hat{\mathbf{x}}_{T+2} = \boldsymbol{C}\mathbf{x}_{T+2}$ gilt.*

Beweis. Um S zu bestimmen, muss der Einfluss des Fehlers $\mathbf{e}_T$ auf den Zustandsvektor $\hat{\mathbf{x}}_{T+2}$ näher betrachtet werden, indem Gleichungen 4.5 und 4.6 rekursiv auf 4.7 angewendet werden:

$$\mathbf{C}\hat{\mathbf{x}}_{T+2} = \mathbf{C}\left(\mathbf{A}\left(\mathbf{A}\hat{\mathbf{x}}_T + \mathbf{e}_T + \mathbf{B}\hat{u}_T\right) + \mathbf{e}_{T+1} + \mathbf{B}\hat{u}_{T+1}\right) \tag{4.11}$$

Zur besseren Lesbarkeit werden die drei Summanden des Reglers separat betrachtet:

$$\hat{u}_{T+1} = \widehat{I}_{T+1} + \widehat{X}_{T+1} + \widehat{D}_{T+1}, \tag{4.12}$$

$$\widehat{I}_{T+1} = -G_I \sum_{i=0}^{T+1} \left[\mathbf{C}\left(\mathbf{A}\hat{\mathbf{x}}_{i-1} + \mathbf{e}_{i-1} + \mathbf{B}\hat{u}_{i-1}\right) - \hat{p}_i^{ref}\right], \tag{4.13}$$

$$\widehat{X}_{T+1} = -\mathbf{G}_x\left(\mathbf{A}\hat{\mathbf{x}}_T + \mathbf{e}_T + \mathbf{B}\hat{u}_T\right), \tag{4.14}$$

$$\widehat{D}_{T+1} = -\sum_{j=1}^{N} G_{d,j}\hat{p}_{T+1+j}^{ref}. \tag{4.15}$$

Da Gleichung 4.13 auf negative Indizes zugreift, ist festzulegen, dass $\hat{\mathbf{x}}_k = (0,0,0)^T$, $\mathbf{e}_k = (0,0,0)^T$ und $\hat{u}_k = 0$ für $k < 0$ gilt.
Das Ziel der folgenden Umformungen und Vereinfachungen ist es, alle Komponenten zu separieren, die keinen Faktor des Regelungsfalls enthalten, also $\hat{\mathbf{x}}, \hat{u}, \hat{p}^m$ oder $\mathbf{e}$. Gleichung 4.15 kann mit Hilfe von Gleichung 4.8 vereinfacht werden:

$$\widehat{D}_{T+1} = -\sum_{j=1}^{N} G_{d,j}p_{T+1+j}^{ref} - \sum_{j=1}^{N} G_{d,j}\hat{p}_{T+1+j}^m, \tag{4.16}$$

$$= -\sum_{j=1}^{N} G_{d,j}p_{T+1+j}^{ref} - m\sum_{j=t_0-1}^{N} G_{d,j}. \tag{4.17}$$

Aus dem zweiten Summanden von Gleichung 4.16 lässt sich bereits entnehmen, ab welchem Zeitpunkt die Modifikation frühestens starten kann. Die Summe beginnt bei 1, daher wird hier die Modifikation der Referenz ab $T+1+1$ verwendet, was bei Darstellung als $\hat{p}^m_{T+t_0}$ bedeutet, dass $t_0 \geq 2$ gelten muss. Entsprechend lässt sich die zweite Summe in Gleichung 4.17 darstellen wie gezeigt.
Gleichungen 4.13, 4.14 und 4.11 werden mit demselben Ziel, die von den Sensordaten beeinflussten Größen zu separieren, zunächst ausmultipliziert:

$$\widehat{I}_{T+1} = -G_I \sum_{i=0}^{T+1} \mathbf{CA}\hat{\mathbf{x}}_{i-1} - G_I \sum_{i=0}^{T+1} \mathbf{Ce}_{i-1} - G_I \sum_{i=0}^{T+1} \mathbf{CB}\hat{u}_{i-1} - G_I \sum_{i=0}^{T+1} \hat{p}_i^{ref}, \quad (4.18)$$

$$\widehat{X}_{T+1} = -\mathbf{G}_x\mathbf{A}\hat{\mathbf{x}}_T - \mathbf{G}_x\mathbf{e}_T - \mathbf{G}_x\mathbf{B}\hat{u}_T, \quad (4.19)$$

$$\mathbf{C}\hat{\mathbf{x}}_{T+2} = \mathbf{CA}^2\hat{\mathbf{x}}_T + \mathbf{CAe}_T + \mathbf{CAB}\hat{u}_T + \mathbf{Ce}_{T+1} + \mathbf{CB}\hat{u}_{T+1}. \quad (4.20)$$

Zur Vereinfachung können nun folgende Gleichungen angewendet werden, die aus der Voraussetzung folgen, dass es sich um einen impulsartigen Fehler zum Zeitpunkt T handelt:

$$\forall k \leq T : \hat{\mathbf{x}}_k = \mathbf{x}_k, \hat{u}_T = u_T \quad (4.21)$$

$$\forall k \neq T : \mathbf{e}_k = 0, \quad (4.22)$$

$$\forall k < T + t_0, t_0 \geq 2 : \hat{p}_k^{ref} = p_k^{ref}. \quad (4.23)$$

Diese angewendet auf Gleichungen 4.18, 4.19 und 4.20 ergibt:

$$\widehat{I}_{T+1} = -G_I \sum_{i=0}^{T+1} \mathbf{CA}\mathbf{x}_{i-1} - G_I\mathbf{Ce}_T - G_I \sum_{i=0}^{T+1} \mathbf{CB}u_{i-1} - G_I \sum_{i=0}^{T+1} p_i^{ref}, \quad (4.24)$$

$$\widehat{X}_{T+1} = -\mathbf{G}_x\mathbf{A}\mathbf{x}_T - \mathbf{G}_x\mathbf{e}_T - \mathbf{G}_x\mathbf{B}u_T, \quad (4.25)$$

$$\mathbf{C}\hat{\mathbf{x}}_{T+2} = \mathbf{CA}^2\mathbf{x}_T + \mathbf{CAe}_T + \mathbf{CAB}u_T + \mathbf{CB}\hat{u}_{T+1}. \quad (4.26)$$

Nun wird die Forderung aus Gleichung 4.7 angewendet, welche umgestellt bedeutet, dass die Differenz der ZMP 0 sein soll. Bei der Bildung der Differenz entfallen alle Summanden, die in der Steuerung und in der Regelung gleich sind:

$$0 = \mathbf{C}\hat{\mathbf{x}}_{T+2} - \mathbf{C}\mathbf{x}_{T+2}, \quad (4.27)$$

$$= \mathbf{CAe}_T + \mathbf{CB}\left(-G_I\mathbf{Ce}_T - \mathbf{G}_x\mathbf{e}_T - m\sum_{j=t}^{N} G_{d,j}\right) \Leftrightarrow \quad (4.28)$$

$$\mathbf{CB}m\sum_{j=t_0-1}^{N} G_{d,j} = \mathbf{CAe}_T + \mathbf{CB}\left(-G_I\mathbf{Ce}_T - \mathbf{G}_x\mathbf{e}_T\right) \Leftrightarrow \quad (4.29)$$

$$m = \underbrace{\left(\mathbf{CB}\sum_{j=t_0-1}^{N} G_{d,j}\right)^{-1}\mathbf{C}\left(\mathbf{A} - \mathbf{B}G_I\mathbf{C} - \mathbf{BG}_x\right)}_{=:\mathbf{G}_{e,t_0}}\mathbf{e}_T. \quad (4.30)$$

Die Invertierbarkeit in Gleichung 4.30 ist gegeben, da es sich bei der Summe um skalare Werte handelt, die Dimension von $\mathbf{C}$ 1×3 und von $\mathbf{B}$ 3×1 ist, und es sich bei dem Produkt damit ebenfalls um ein Skalar handelt. Funktion S kann nun wie folgt definiert werden:

$$S(t_0, \mathbf{e}) = \mathbf{G}_{e,t_0}\mathbf{e}, \qquad t_0 \in 2, \ldots, N+1 \tag{4.31}$$

□

Die Matrizen $\mathbf{G}_{e,t_0}$ können für alle möglichen t_0 bestimmt und in eine Tabelle geschrieben werden, bevor der Lauf startet, und während des Laufs aus der Tabelle entnommen werden. Zum besseren Verständnis sei noch erwähnt, dass der Index $T + t_0$, ab dem der Referenz-ZMP modifiziert wird, nicht damit zusammenhängt, wann modifiziert wird. Dies hat online zu geschehen, wenn der gemessene Fehler sich auf den Zustand auswirkt und damit die Ausgabe des Reglers beeinflusst. Da dies beim Zustand $T + 1$ der Fall ist, muss die Modifikation direkt nach der Berechnung von $\hat{\mathbf{x}}_{T+1}$ und vor $\hat{u}_{T+1}$ durchgeführt werden.
Für die praktische Anwendung ist dies jedoch noch nicht ausreichend, da nicht davon auszugehen ist, dass nur zu einem Zeitpunkt ein Fehler gemessen wird. Idealerweise werden auf diese Art alle auftretenden Fehler ausgeglichen. Die Forderung nach einem impulsartigen Fehler wird daher aufgegeben und zudem nicht mehr gefordert, dass die Modifikation dazu führt, dass sich das System trotz Fehler wie im Steuerungsfall verhält, bei dem $\mathbf{e}_k = 0, \forall k$ gilt.
Es ist zu beachten, dass eine vollständig separate Betrachtung der Modifikationen nicht möglich ist, da die Forderung aus Gleichung 4.7 nur für den Zeitpunkt $T + 2$ gilt, so dass kleinere Abweichungen zu anderen Zeitpunkten zu erwarten sind. Diese wirken sich auch auf den nun zu korrigierenden Zeitpunkt aus, während diese Korrektur auch die folgenden bereits durchgeführten Korrekturen betrifft. Die Größe des Einflusses wird im Folgenden genauer diskutiert.

4.3.2 Beispiel. Abbildung 4.4 zeigt einen Beispiellauf, bei dem drei Störungen gemessen werden. Wie zu sehen ist, werden die Störungen mittels der Ausfallschritte derart balanciert, dass der ZMP-Verlauf mit dem ZMP-Verlauf im ungestörten Fall vergleichbar ist.

Fehlerabschätzung

Abbildung 4.5 zeigt die Differenzen zwischen den Referenz-ZMP und den geschätzten ZMP mit und ohne Modifikation. Allgemein ist zu erwarten, dass ein geschätzter ZMP nicht exakt der Referenz folgt, da die durch den Regler zu minimierende Funktion (siehe Gleichung 3.37) neben der Differenz zwischen Referenz-ZMP und geschätztem ZMP weitere Summanden enthält. Es ist zu erkennen, dass zu den Zeitpunkten, an denen der Referenz-ZMP modifiziert wird, eine größere Abweichung im Vergleich zum Steuerungsfall entsteht, da sich die Referenz hier sprunghaft ändert, jedoch laut Zielfunktion die Änderung im ZMP ebenfalls minimiert werden soll, so dass der modifizierten Referenz nur verzögert gefolgt werden kann. Das betrifft

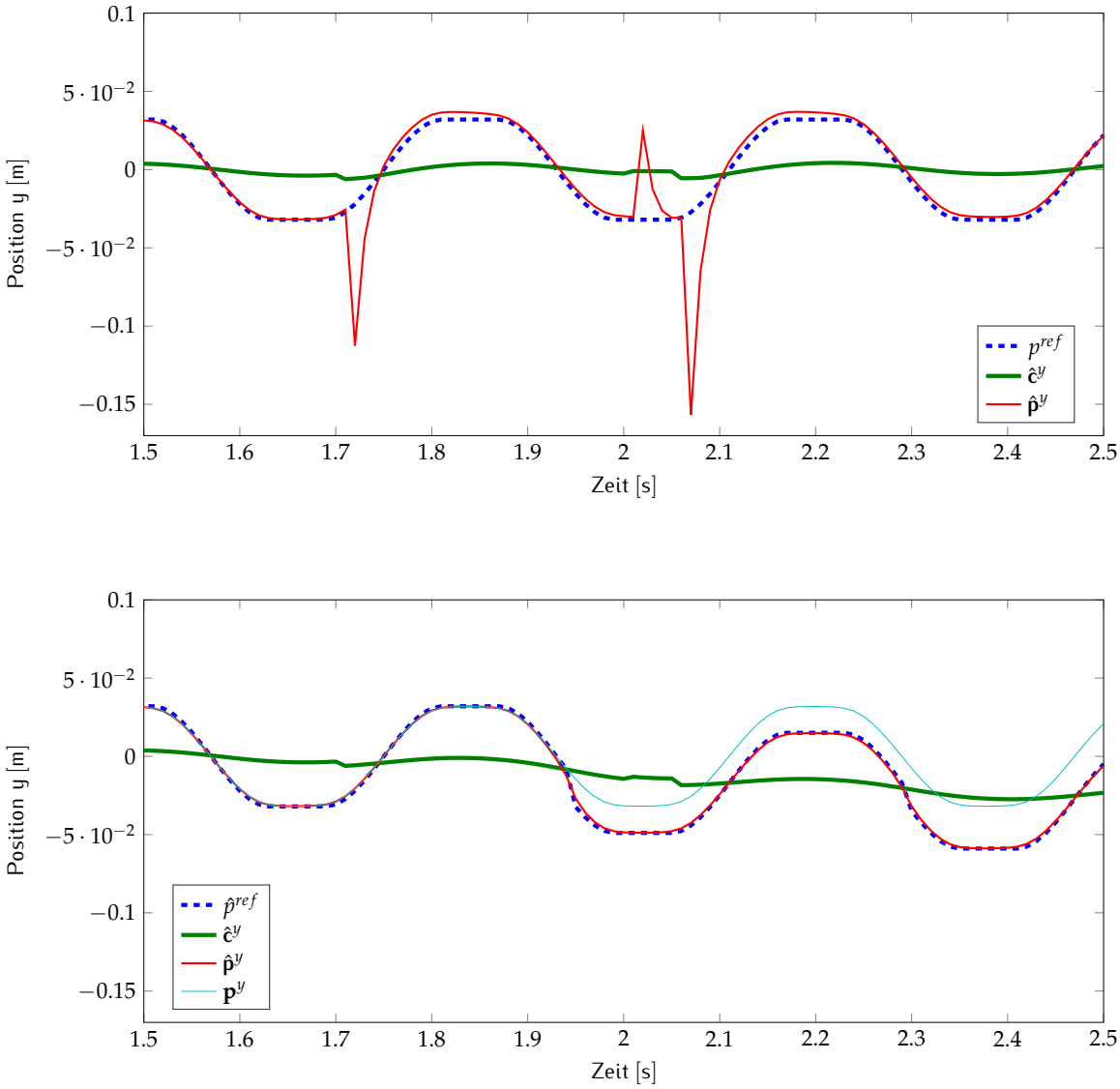

Abbildung 4.4: Beispiele für gemessene Störungen in der Schwerpunktposition von $-0.005\,\mathrm{m}$ bei $1.7\,\mathrm{s}$, $0.003\,\mathrm{m}$ bei $2\,\mathrm{s}$ und $-0.007\,\mathrm{m}$ bei $2.05\,\mathrm{s}$. Im oberen Diagramm balanciert ohne Modifikation des Referenz-ZMP, unten mit Modifikationen bei $1.95\,\mathrm{s}$ und $2.3\,\mathrm{s}$. Der Unterschied des modifizierten Referenz-ZMP zur unmodifizierten Referenz verstärkt sich sichtbar nach der zweiten Modifikation, siehe Sekunde 2.4 im Vergleich zu Sekunde 2.

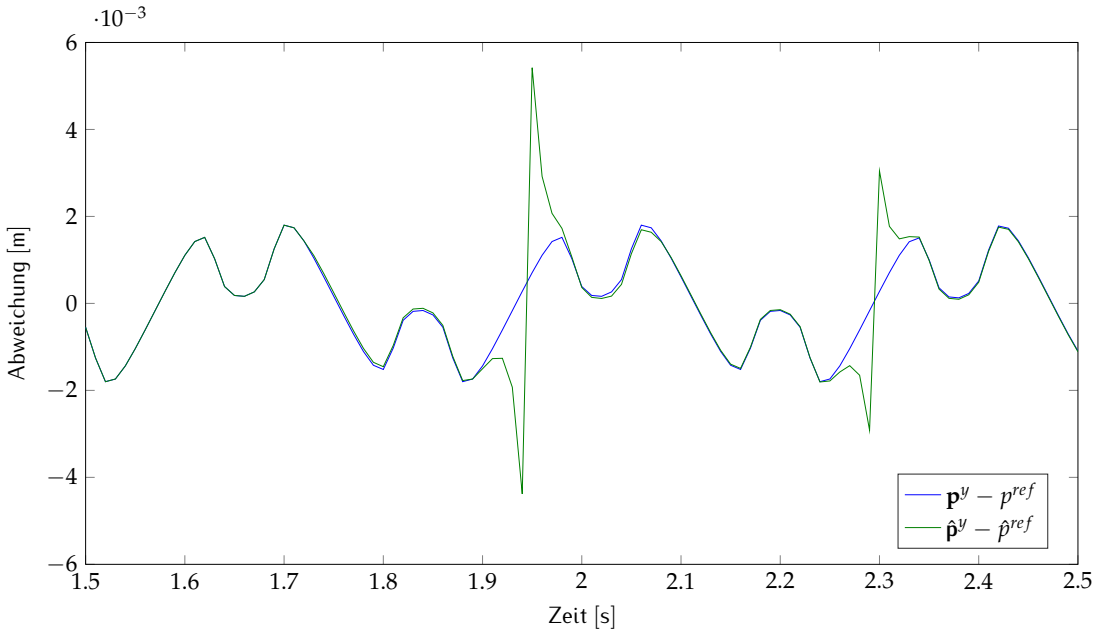

Abbildung 4.5: Differenz der ZMP-Referenz zum geschätzten ZMP ohne und mit Modifikation des zukünftigen Referenz-ZMP, berechnet anhand des in Abbildung 4.4 gezeigten Beispiels.

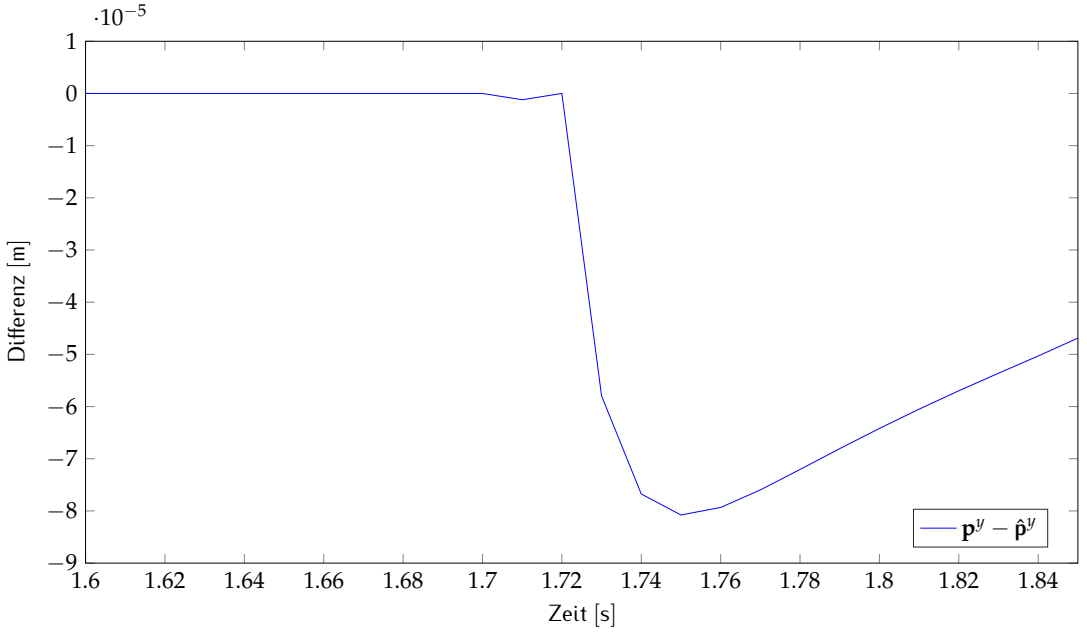

Abbildung 4.6: Differenz zwischen den geschätzten ZMP eines Systems ohne und mit einer Störung bei 1.7 s, berechnet anhand des in Abbildung 4.4 gezeigten Beispiels.

aufgrund der Vorschau auch den Zeitraum vor der sprunghaften ZMP Änderung (ab $T+3$), da hier die Referenz innerhalb der Vorschau bereits vom ungestörten System abweicht und der Regler sich daher anders verhält.
Abbildung 4.6 zeigt den Unterschied der geschätzten ZMP im modifizierten und ungestörten Fall anhand des obigen Beispiels. Die Differenz ab $T+3$ (hier 1.73 s) wird hier deutlich, während die geschätzten ZMP zum Zeitpunkt $T+2$ (hier 1.72 s) wie nach Gleichung 4.7 gewünscht keine Differenz aufweisen. Bei $T+1$ (hier 1.71 s) ist jedoch ebenfalls ein Fehler von $-0.0012\,\mathrm{mm}$ zu erkennen, welcher damit zu erklären ist, dass aufgrund der Störung bei T der gesamte Zustand des Systems beeinflusst wird und damit nicht mehr mit dem ungestörten Zustand übereinstimmt.

4.4 Zusammenfassung

Die Rückführung von Sensordaten zur Stabilisierung eines Laufs ist wesentlich für den Einsatz auf realen Robotern. In diesem Abschnitt werden mögliche Sensoren für verschiedenartige Störungen diskutiert und kategorisiert in leichtere und schwerere Störungen. Allgemein werden aus den Sensordaten die Positionen von ZMP und CoM errechnet, die vom Beobachter genutzt werden, um den Zustand des Roboters zu schätzen. Bei schwereren Störungen, durch die der Roboter kippt, wird die CoM-Messung zur Durchführung von Ausfallschritten genutzt, welche Reaktionen auf Störungen vermeiden, die den Roboter zu Fall bringen können. Das folgende Kapitel wird zeigen, wie die Ergebnisse dieses und des vorherigen Kapitels zu einem vollständigen Laufalgorithmus beitragen.

KAPITEL 5

Laufalgorithmus

Die in den Kapiteln zuvor vorgestellten Regler und die Behandlung von Sensordaten stellen den zentralen Punkt für einen Laufalgorithmus dar. Jedoch sind Referenz-Fußpositionen bzw. der Referenz-ZMP eine unintuitive Möglichkeit, den Lauf eines Roboters zu definieren. Die gewünschte Geschwindigkeit ist hier ein weit zugänglicherer Wert. Auch die Regelgröße, die Änderung des tatsächlichen ZMP, ist nur indirekt auf den Roboter anwendbar, da die Bewegung von Robotern häufig durch Winkel der Gelenke definiert wird, so dass weitere Zwischenschritte durchgeführt werden müssen.
Gegenstand dieses Kapitels ist daher ein Laufalgorithmus, der wie in Abbildung 5.1 dargestellt aufgebaut ist und als Eingabe die gewünschte Geschwindigkeit erhält. Es wird hier der Algorithmus vorgestellt, der auf den PC mit separaten Modulen für Referenz-Erzeugung und Ausfallschritten basiert. Prinzipiell wäre ein Algorithmus für MPC ähnlich, würde allerdings die Komponenten „ZMP Generation“, „ZMP/IP Controller“ und Teile des „Swing Leg Controllers“ nicht benötigen, da diese implizit in MPC enthalten sind. Die einzelnen Komponenten werden im Folgenden näher erläutert.

5.1 Pattern Generator

Die Eingabe des Laufalgorithmus ist die gewünschte durchschnittliche Geschwindigkeit entlang der Sagittal- und Lateralachse, sowie die Rotationsgeschwindigkeit. Daraus werden zweidimensionale Fußpositionen im Weltkoordinatensystem ${}^{2W}\mathbf{F}$ erzeugt, die man sich wie Fußstapfen im Schnee vorstellen kann, in die der Roboter seine Füße platzieren soll. Jede erzeugte Position ist einer von vier möglichen Phasen zugeordnet. Bei zwei der vier Phasen stützen beide Füße den Roboter, bei den beiden anderen nur der linke bzw. der rechte

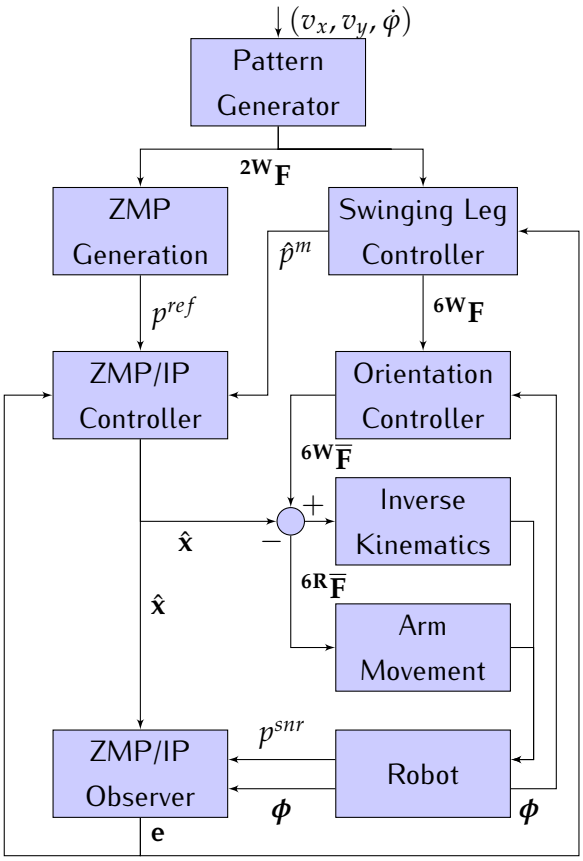

Abbildung 5.1: Übersicht zur (vereinfachten) Struktur des Laufalgorithmus. Mit $\mathbf{F}$ werden die Fußpositionen bezeichnet, welche im Weltkoordinatensystem (W) oder Roboterkoordinatensystem (R) angegeben werden können. Indizes 2 und 6 bezeichnen die Dimension der Positionen (translatorisch und im letzten Fall auch rotatorisch) und $\overline{\mathbf{F}}$ eine zusätzliche Behandlung von fehlerhaften Oberkörperorientierungen. Die gemessene Orientierung wird mit $\boldsymbol{\phi}$ bezeichnet und der gemessene ZMP mit p^{snr}.

Fuß. Die vier Phasen werden nacheinander durchgeführt, wobei auf eine Single-Support-Phase eine Double-Support-Phase folgt. Die Distanz zwischen den beiden Füßen ergibt sich durch die Geschwindigkeit und die fest vorgegebene Zeit $2\,(T_{ss} + T_{ds})$, in der alle 4 Phasen ausgeführt werden, wobei T_{ss} die Dauer einer Single-Support-Phase und T_{ds} die Dauer einer Double-Support-Phase ist.

Die gewünschte Geschwindigkeit wird nicht sofort erreicht, sondern wird durch eine einstellbare Beschleunigung begrenzt. Stoppen ist jedoch jederzeit ohne derartige Beschränkungen möglich, sobald ein Schritt neu geplant wird, bei dem beide Füße nebeneinander aufgesetzt werden können.

5.2 ZMP Generation

Die Erzeugung des ZMP erscheint zunächst simpel, da er sich theoretisch nur innerhalb des Stützpolygons befinden muss. Für reale Anwendungen wird häufig argumentiert, er sollte sich in der Mitte des Polygons befinden [13], um robust gegen Störungen zu sein. In der Praxis zeigt sich jedoch der Nachteil einer konstanten Position. Gleichung 2.2 besagt, dass ein größerer Abstand zwischen ZMP und CoM zu größeren Beschleunigungen führt. Falls der Abstand zudem nicht konstant ist, können die wechselnden Beschleunigungen zu einem Aufschwingen und letztendlich zum Fall führen. Es wird hier daher der ZMP mit dem Ziel gewählt die Beschleunigungen zu minimieren.

In den bisherigen Abschnitten stellte der Index 0 des Referenz-ZMP den aktuellen Zeitpunkt dar. In den folgenden Abschnitten wird gezeigt, wie er für jeweils einen gesamten Schritt (eine Double-Support-Phase und eine Single-Support-Phase) erzeugt wird, woraus sich anschließend die benötigte Referenz zusammenstellen lässt. Im Folgenden sei nun angenommen, dass der aktuelle Zeitpunkt $k = 0$ der Beginn einer Single-Support-Phase ist. Der Referenz-ZMP $\mathbf{p}_k^{ref}$ wird in diesem Kapitel in Weltkoordinaten dargestellt und ist daher zweidimensional. Er wird in den nächsten beiden Abschnitten für die x und y-Achse des Roboters separat bestimmt:

$$\mathbf{p}_k^{ref} = \begin{pmatrix} p_k^{ref,x} \\ p_k^{ref,y} \end{pmatrix}.$$

Zunächst wird hier die Erzeugung für einen Lauf parallel zur x-Achse gezeigt ohne Rotationen. Diese werden nachträglich hinzugefügt, wie in Abbildung 5.3 gezeigt, indem der ZMP-Verlauf um den Ursprung des jeweiligen Fußes gedreht wird, der aktuell den Roboter stützt. Falls beide Füße den Roboter stützen ist es der Fuß der letzten Phase. Der zu rotierende Winkel ist dabei der Winkel des Oberkörpers im Raum.

Saggitalachse

Für die x-Achse lässt sich $p_k^{ref,x}$ so wählen, dass er sich, wie auch der CoM, konstant mit v_x bewegt. Für die Geschwindigkeit $\dot{p}_{x,ss}^{ref}$ des ZMP für die x-Achse im Falle einer Single-Support-Phase gilt zunächst:

$$\dot{p}_{x,ss}^{ref} = \begin{cases} v_x, & \text{falls } v_x \leq \frac{l_f}{T_{ss}} \\ \frac{l_f}{T_{ss}}, & \text{andernfalls} \end{cases},$$

wobei l_f die Länge des Stützfußes ist. Der zweite Fall begrenzt den Bewegungsraum des ZMP auf die Länge des Fußes, falls die Geschwindigkeit zu hoch ist. In diesem Fall gilt für die ZMP-Geschwindigkeit in der Double-Support-Phase:

$$\dot{p}_{x,ds}^{ref} = \frac{(T_{ss} + T_{ds})v_x - T_{ss}\dot{p}_{x,ss}^{ref}}{T_{ds}}.$$

Ist die Geschwindigkeit nicht zu hoch, verläuft der ZMP auch hier mit konstanter Geschwindigkeit v_x.
Aus den ZMP-Geschwindigkeiten lässt sich mittels explizitem Euler-Verfahren die Position zum Zeitpunkt k bestimmen, indem zunächst die Single-Support-Phase und getrennt davon die Double-Support-Phase betrachtet wird:

$$p_k^{ref,x} = \begin{cases} {}^{2W}\mathbf{F}_{SS}^x + \dot{p}_{x,ss}^{ref} \cdot \Delta t \cdot k, \text{falls } k = 0, \ldots, \frac{T_{ss}}{\Delta t} - 1 \\ {}^{2W}\mathbf{F}_{SS}^x + \dot{p}_{x,ss}^{ref} \cdot T_{ss} + \dot{p}_{x,ds}^{ref} \cdot \Delta t \cdot \left(k - \frac{T_{ss}}{\Delta t}\right), \text{falls } k = \frac{T_{ss}}{\Delta t}, \ldots, \frac{T_{ss}+T_{ds}}{\Delta t} - 1 \end{cases}.$$

Dabei ist ${}^{2W}\mathbf{F}_{SS}^x$ die x-Komponente der zweidimensionalen Position der hinteren Kante des Fußes während der Single-Support-Phase.

Lateralachse

Entlang der y-Achse ist eine Beschleunigung unvermeidbar, da diese benötigt wird, um die dynamische Stabilität herzustellen. Dennoch ist es sinnvoll, den Referenz-ZMP so zu wählen, dass unnötige Beschleunigungen vermieden werden können. Entlang dieser Achse ähnelt der Verlauf des CoM einer Sinus-Kurve, falls der Referenz-ZMP fest in der Fußmitte gewählt wird. Daraus lässt sich jedoch nicht ableiten, dass ein Sinus-Verlauf des Referenz-ZMP zu einer maximalen Stabilität führt. Es ist experimentell festzulegen, welcher Verlauf des Referenz-ZMP zu geringen Beschleunigungen führt, der zudem innerhalb des Stützpolygons liegt und auch bei Anwendung auf einem physischen Roboter zu der gewünschten Stabilität führt. Eine flexible Art und Weise eine Referenz zu wählen sind kubische Bézier-Kurven. Alternativ kämen auch B-Spline-Kurven in Frage, welche jedoch zur effizienten Berechnung in Bézier-Kurven zerlegt werden, wobei die höhere Komplexität an dieser Stelle nicht notwendig ist.

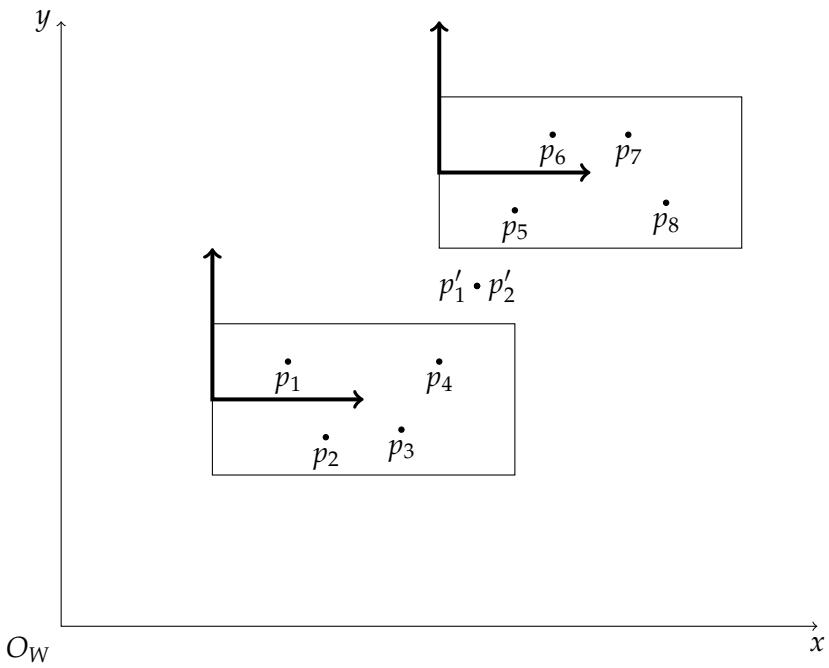

Abbildung 5.2: Definition des Kontrollpolygons zur Erzeugung des Referenz-ZMP mittels Bézier-Kurve entlang der Lateralachse. Die Punkte p'_1 und p'_2 stimmen überein und liegen in der Mitte zwischen den beiden Füßen. Die Abbildung zeigt zur besseren Lesbarkeit die Punkte mit zwei Dimensionen, wobei die x-Komponente automatisch durch die Geschwindigkeit festgelegt wird und nur die y-Komponente wählbar ist. Auch die Bézier-Kurve wird eindimensional berechnet.

Während der Single-Support-Phase besteht das Kontrollpolygon aus vier eindimensionalen Punkten. Abbildung 5.2 zeigt ein Beispiel, in dem die Dimension x zur besseren Visualisierung mit eingezeichnet ist. Die y-Koordinaten des Kontrollpolygons im Falle der Single-Support-Phase werden im lokalen Fußkoordinatensystem definiert. Bei den Punkten des Kontrollpolygons handelt es sich bei der Single-Support-Phase um p_1,p_2,p_3 und p_4 bzw. p_5,p_6,p_7 und p_8 und können frei gewählt werden. Während der Double-Support-Phase wird ebenfalls eine Menge P_{ds} mit vier Kontrollpunkten definiert:

$$P_{ds} = \left(p_4, p'_1, p'_2, p_5\right).$$

Bei den Punkten p_4 und p_5 handelt es sich jeweils um den letzten aus dem vorherigen Schritt und dem ersten innerhalb des aktuell aufgesetzten Fußes. Die verbleibenden werden aus den y-Positionen des rechten bzw. linken Fußes berechnet:

$$p'_1 = p'_2 = \frac{{}^{2W}\mathbf{F}_L^y - {}^{2W}\mathbf{F}_R^y}{2} + {}^{2W}\mathbf{F}_R^y.$$

Nachdem die Punkte des Kontrollpolygons festgelegt wurden, wird die Bézier-Kurve für die jeweilige Phase berechnet und $p_k^{ref,y}$ für $k = 0, \ldots, \frac{T_{ss}}{\Delta t} - 1$ bzw. $k = \frac{T_{ss}}{\Delta t}, \ldots, \frac{T_{ss}+T_{ds}}{\Delta t} - 1$ zugewiesen.

5.3 ZMP/IP Controller und Observer

Der Referenz-ZMP ist die Eingabe für den ZMP/IP Controller. Aus diesem und dem aktuellen Zustand wird, wie in Abschnitt 3.2 beschrieben, die Reglerausgabe berechnet. Wie bereits erwähnt eignet sich diese jedoch nicht als Eingabe für den Roboter. Um die notwendigen Gelenkwinkel bestimmen zu können, wird die Zustandsgleichung 4.5 mit den Matrizen 3.34 berechnet. Die Ausgabe des ZMP/IP Controller ist daher der Zustand, woraus die CoM-Position später verwendet wird, um Fußpositionen im Roboterkoordinatensystem zu bestimmen, welche letztendlich zur Bestimmung der Winkel dienen.
Der aktuelle Zustand wird auch vom Beobachter verwendet, der, wie in Gleichung 4.5 dargestellt, den aktuellen Fehler **e** berechnet. Da bereits der ZMP/IP Controller den aktuellen Zustand berechnet, wird dies nicht im Beobachter-Modul durchgeführt, so dass dort die Ausgabe nur der Fehler ist, welcher vom ZMP/IP Controller zur Bestimmung des aktuellen Zustandes verwendet wird.

5.4 Swinging Leg Controller

Wie in Abschnitt 5.1 beschrieben stehen nach der Ausführung der zuvor beschriebenen Module nur zweidimensionale Fußpositionen zur Verfügung. Um den Fuß von einem Punkt am Boden zum nächsten zu bewegen, wird der Swinging Leg Controller benötigt, welcher eine

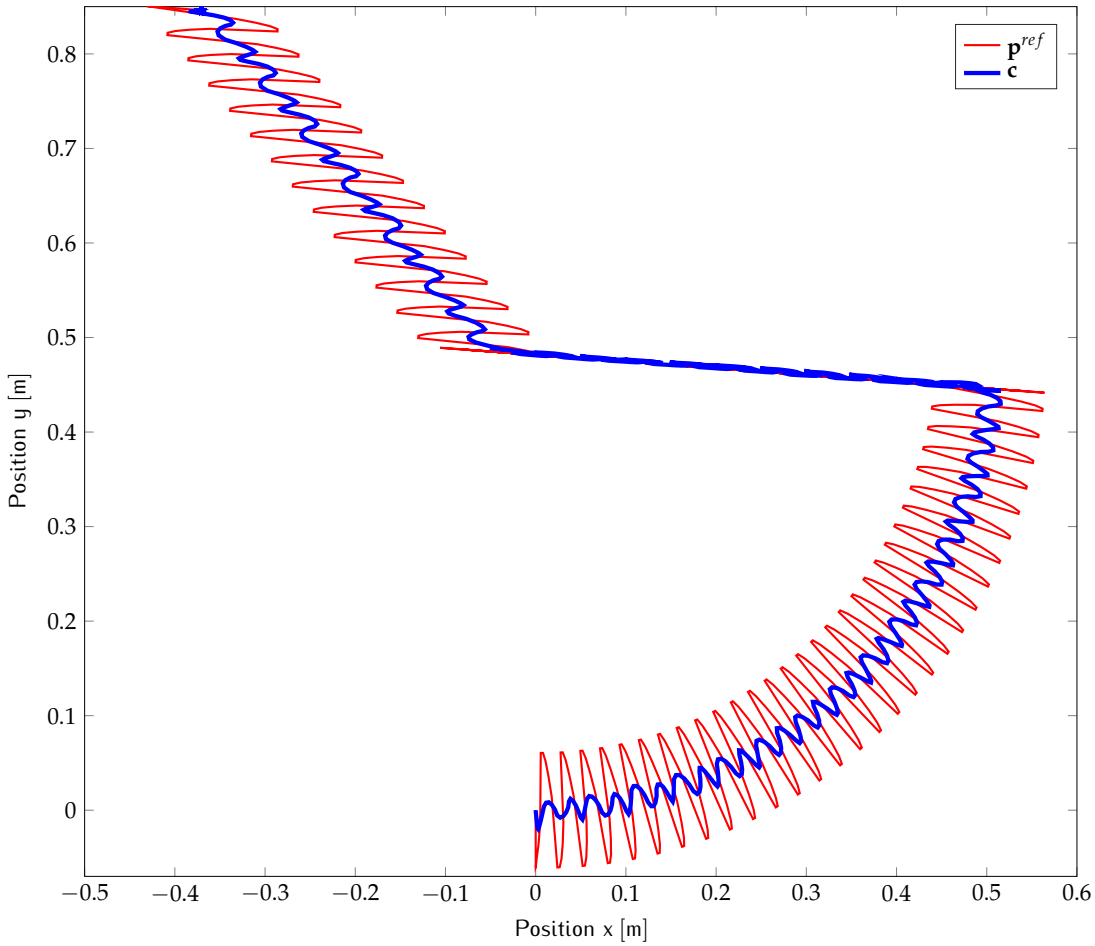

Abbildung 5.3: Beispiel für einen Referenz-ZMP Verlauf ($\mathbf{p}^{ref}$) und daraus resultierendem CoM Verlauf ($\mathbf{c}$) bei einem omnidirektionalen Lauf.

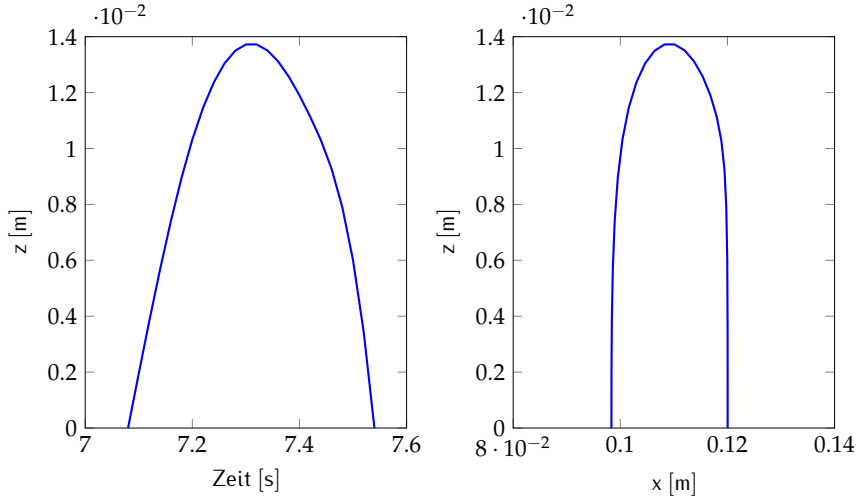

Abbildung 5.4: Beispiel für eine Fußbewegung während eines Schrittes, links als Höhe über Zeit, rechts als Höhe über Vorwärtsbewegung.

fließende Bewegung berechnet, die auch die Höhe über dem Boden enthält. Die Bewegung ist zu komplex für Bézier-Kurven mit nur 4 Kontrollpunkten, daher werden B-Spline-Kurven verwendet, wobei der Grad und die Anzahl der Kontrollpunkte wählbar ist. Auch die Positionen der Kontrollpunkte entlang der Geraden vom Start- zum Zielpunkt, sowie die Höhe über dem Boden kann frei eingestellt werden. Vorzugsweise werden die Punkte so gewählt, dass der Fuß am Ende des Schrittes weitgehend ohne Vorwärtsbewegung abgesenkt wird, für den Fall, dass er zu früh den Boden berührt. Abbildung 5.4 zeigt ein Beispiel.
Zudem ist der Swinging Leg Controller verantwortlich für die Bestimmung der Ausfallschritte, da hier die detaillierten Informationen zu den Schritten zur Verfügung stehen bzw. bestimmt werden. Die Modifikation wird wie in Gleichung 4.31 beschrieben berechnet, direkt in die Planung der Schritte einbezogen und an den ZMP/IP Controller weitergereicht. Die Modifikation ist jedoch nicht immer möglich. Sie führt möglicherweise zu einem physisch nicht ausführbaren Schritt, beispielsweise wenn sich die Beine kreuzen oder sie zu kurz sind, falls die Modifikation zu groß wird. Das geschieht im Normalfall nicht innerhalb eines Zeitschrittes, in dem eine einzelne Modifikation eher klein ist, sondern kann in der Summe über einen ganzen Schritt geschehen. Es wird daher in jedem Zeitschritt einzeln geprüft, ob die berechnete Modifikation anwendbar ist. Ist sie es nicht, so wird sie nicht eingeplant und der Fehler durch den Regler ausgeglichen. Abbildung 5.5 zeigt den Entscheidungsprozess.

5.5 Orientation Controller

Der Orientation Controller hat zwei Aufgaben. Zum einen soll er unerwünschte Kontakte des Schwungfußes mit dem Boden vermeiden, der im Falle eines kippenden Roboters wahrscheinlich ist. Zum anderen werden hohe Kräfte und Drehmomente benötigt, um einen gekippten Roboter wieder aufzurichten, was durch den Orientation Controller abgemildert werden soll. Beides zusammen geschieht in zwei Phasen. Wie in Abbildung 5.6 zu sehen ist, wird der Schwungfuß (hier der rechte mit der Position ${}^{6W}\mathbf{F}_R$) um den Koordinatensystemursprung des Standfußes (hier ${}^{6W}\mathbf{F}_L$) gedreht, so dass die Höhe über dem Boden der geplanten entspricht. Zusätzlich wird der Fuß um $-\varphi_y$ gedreht, so dass er parallel zum Boden ist, wobei beide Anpassungen nur durchgeführt werden, wenn der Fuß sich nicht am Boden befindet. Sobald er den Boden planmäßig berührt, beginnt die Double-Support-Phase und damit die zweite Phase des Orientation Controller, in der zunächst der Oberkörperwinkel um die y-Achse gespeichert wird, wie er zu Beginn dieser Phase gemessen wird, hier bezeichnet als φ_f. In der aktuellen Double-Support-Phase und der folgenden Single-Support-Phase richten die Standfüße den Oberkörper dann langsam auf. Die Anpassung geschieht wie zuvor, jedoch mit dem Winkel $\varphi_f \cdot \lambda_1^{t_s}$, wobei λ_1 ein Faktor < 1 und $t_s \in \mathbb{N}_0$ der Zeitindex innerhalb des aktuellen Schrittes ist (Double-Support-Phase zuzüglich folgende Single-Support-Phase).
Bei größeren Kippwinkeln nimmt auch die Höhe des CoM über dem Boden ab, so dass er anschließend wieder in seine gewünschte Höhe gehoben werden muss. Neben der Wie-

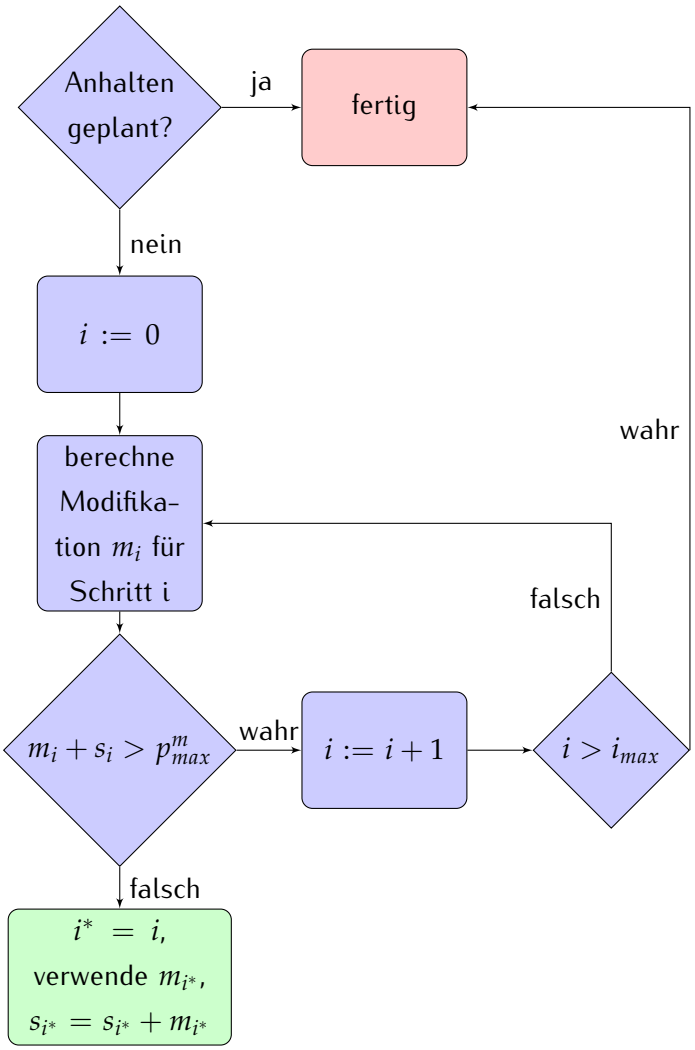

Abbildung 5.5: Entscheidungsprozedur für einen Ausfallschritt, der den Fehler, aufgetreten im aktuellen Zeitschritt, ausgleichen soll. Hier steht i^* für den Schritt (Fußposition), in dem der Ausfallschritt eingeplant werden soll, wobei 0 der nächste Schritt ist. Es wird zunächst eine Modifikation der Größe m_i berechnet. Ist die Summe aus aktueller Modifikation und der Summe s_i aller bisherigen Modifikationen für Schritt i größer als ein p^m_{max}, wird Schritt i nicht weiter modifiziert. Die Schranke i_{max} bewirkt, dass eine Modifikation nicht zu weit in die Zukunft hinausgezögert wird, was zu deutlich vergrößerten Ausfallschritten führen würde. Wurde ein möglicher Schritt i^* für die Modifikation gefunden, wird auf die Summe s_{i^*} für diesen Schritt die zusätzliche Modifikation addiert.

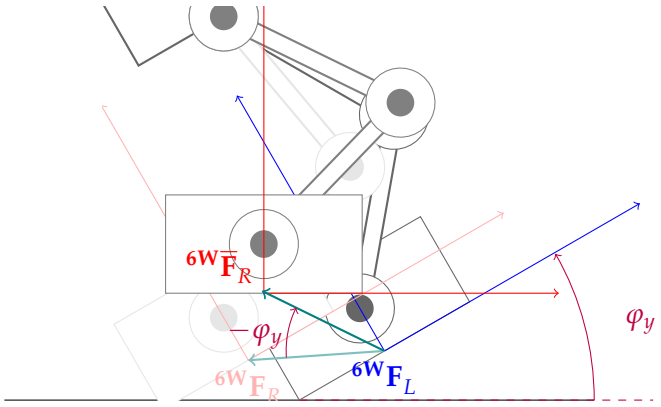

Abbildung 5.6: In dieser Beispielabbildung wird der Schwungfuß mit der Position ${}^{6W}\mathbf{F}_R$ um den Standfuß mit dem Ursprung bei ${}^{6W}\mathbf{F}_L$ rotiert um ungewollte Kontakte mit dem Boden zu vermeiden. Die Position des Fußes vor der Anpassung an die Orientierung ist hier in Grau angedeutet, das Resultat der Drehung mit ${}^{6W}\overline{\mathbf{F}}_R$ bezeichnet. Zusätzlich wird der Schwungfuß um die y-Achse seines Koordinatensystems um $-\varphi_y$ rotiert, um Parallelität mit dem Boden zu erreichen.

derherstellung der senkrechten Haltung ist auch diese Korrektur mit erhöhten Kräften und Drehmomenten verbunden. Ähnlich wie oben wird hier daher das Aufrichten unterstützt, indem für kurze Zeit ein niedrigerer Schwerpunkt akzeptiert wird. Der Orientation Controller ersetzt daher die ursprüngliche Schwerpunkthöhe $\mathbf{c}^z$ durch $\mathbf{c}^z - \varphi_f \cdot \lambda_2$, wobei λ_2 bestimmt, wie stark die Höhe in Abhängigkeit von der Orientierung abnimmt.
Die Werte für λ_1 und λ_2 werden experimentell bestimmt. Höhere Werte führen zu einer schnelleren Wiederherstellung der gewünschten Orientierung und Höhe, können jedoch zu hohe Anforderungen an die Motoren stellen.

5.6 Koordinatensysteme und inverse Kinematik

Bis zu diesem Punkt liegen alle Koordinaten im Weltkoordinatensystem vor. Bevor die inverse Kinematik angewendet wird, müssen sie, wie in Abbildung 5.7 dargestellt, in das Roboterkoordinatensystem konvertiert werden. Durch den Regler ist die Position des CoM $\mathbf{c}^x$ im Weltkoordinatensystem gegeben. Zudem kann der aktuelle Schwerpunkt des Roboters $\mathbf{c}_b^x$ (im Roboterkoordinatensystem) bestimmt werden, indem eine direkte Kinematik mit den einzelnen Schwerpunkten der Bauteile angewendet wird. Wie in der Grafik zu sehen ist, lässt sich der Vektor ${}^{6R}\overline{\mathbf{F}}$ aus $\mathbf{c}_b^x$, $\mathbf{c}^x$ und ${}^{6W}\overline{\mathbf{F}}$ berechnen, wenn angenommen wird, dass es sich bei $\mathbf{c}^x$ und $\mathbf{c}_b^x$ wie gewünscht um die gleichen Punkte im Raum handelt:

$$ {}^{6R}\overline{\mathbf{F}} = \mathbf{c}_b^x - \mathbf{c}^x + {}^{6W}\overline{\mathbf{F}}. \tag{5.1} $$

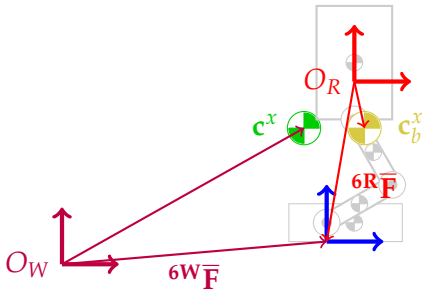

Abbildung 5.7: Aus den Fußpositionen im Weltkoordinatensystem O_W können die Fußpositionen im Roboterkoordinatensystem O_R berechnet werden, indem angenommen wird, dass die tatsächliche Position des Schwerpunktes im Roboterkoordinatensystem mit der gewünschten übereinstimmt.

Es sei nochmals erwähnt, dass auch wenn es sich um den gleichen Ort im Raum handelt, sich $\mathbf{c}^x$ und $\mathbf{c}_b^x$ dennoch unterscheiden, da sie in unterschiedlichen Koordinatensystem ausgedrückt werden.
An dieser Stelle wird eine Schwierigkeit in der Berechnung von $\mathbf{c}_b^x$ klar. Es wird eine direkte Kinematik zur Bestimmung von $\mathbf{c}_b^x$ angewendet, welche die Gelenkwinkel benötigt, die allerdings erst nach der Berechnung von Gleichung 5.1 und der inversen Kinematik zur Verfügung stehen. Ein iterativer Prozess wäre demnach nötig, um $\mathbf{c}_b^x$ korrekt zu bestimmen. Zur Vermeidung dieses zeitintensiven Verfahrens werden hier daher die gemessenen Gelenkwinkel aus dem vorherigen Zeitschritt verwendet.
Anschließend kann zur Berechnung der Winkel der Beine die inverse Kinematik ausgeführt werden. Es existiert jedoch keine für die Arme, so dass die Bewegung der Arme nach der inversen Kinematik im Winkelraum definiert wird. Die menschliche Bewegung wird hier zum Vorbild genommen. Bewegt sich ein Fuß in positive x-Richtung, bewegt sich der Arm der gegenüberliegenden Seite ebenfalls proportional nach vorne. Damit werden Drehmomente um die z-Achse ausgeglichen, so dass weniger ungewollte Rotationen entstehen. Anschließend werden die Winkel zum Roboter gesendet.

5.7 Zusammenfassung und Diskussion

Dieses Kapitel zeigt die Vervollständigung des zuvor dargestellten Reglers zum vollständigen Laufalgorithmus. Beginnend mit der gewünschten Geschwindigkeit werden Referenzfußpositionen erzeugt, mit deren Hilfe der Referenz-ZMP bestimmt wird. Dieser wird vom Regler genutzt, um die Referenz-CoM-Position zu berechnen, die zur Konvertierung der Fußpositionen in Weltkoordinaten in das Roboterkoordinatensystem verwendet wird. Dazu wird der CoM im Roboterkoordinatensystem genutzt, der aus den letzten gemessenen Gelenkwinkeln mittels direkter Kinematik berechnet wird. Nach der inversen Kinematik können die Gelenkwinkel an den Roboter geschickt werden.

Dieser Ablauf entspricht nicht in allen Teilen dem, wie es in einem einfachen Regelkreis erwartet wird. Es gibt neben der eigentlichen Rückführung der Sensordaten zwei weitere Rückkopplungen, die unerwünscht aber derzeit nicht vermeidbar sind.
Zum einen stellt die Berechnung $\mathbf{c}_b^x$ einen weiteren geschlossenen Kreis dar. Die gemessenen Winkel fließen in die Berechnung der nächsten Sollwinkel ohne Gewichtung ein. Da sich die Position des CoM im Roboterkoordinatensystem im Vergleich zu dem im Weltkoordinatensystem allerdings nur wenig verändert, stellt dies in der Praxis keine Schwierigkeit dar.
Jedoch gibt es eine weitere Rückkopplung, die sich aufgrund der Tatsache ergibt, dass der CoM (erste Komponente des Zustandes) zur Ansteuerung des Roboters genutzt werden muss. Wie in Gleichung 4.5 zu sehen ist, wird die Differenz zwischen geschätztem und tatsächlichem Zustand, skaliert um den Vektor **L**, zum Zustand addiert, aus dem die CoM Position für die aktuellen Sollwinkel stammt. Dadurch ist bei der Wahl von **L** auch zu berücksichtigen, dass diese Rückkopplung nicht zu einer Instabilität im System führt. Die Evaluation, dargestellt im nächsten Kapitel, wird jedoch zeigen, dass der Beobachter nachweislich das System stabilisiert und erläutert zudem wie genau dies geschieht.

KAPITEL 6

Evaluation

In diesem Kapitel werden die vorgeschlagenen Verfahren auf ihre Eignung und dem Verbesserungspotential bezüglich der geforderten Ziele untersucht. Durch die Formulierung von geeigneten Anforderungen und entsprechende Wahl des grundlegenden Bewegungssteuerungsverfahrens ist bereits die Erfüllung einiger Punkte sichergestellt. Wie in Kapitel 3 dargestellt erlauben MPC und PC durch die Bestimmung des Schwerpunktverlaufs mit anschließender inversen Kinematik (Abschnitt 5.6) die winkelbasierte Kontrolle, die freie Referenzwahl (Referenz-ZMP p_k^{ref} bzw. Referenz-Fußpositionen $\mathbf{F}_k^{ref}$ als Vorschau) und sind auch onlinefähig, da die Vorschau nur eine begrenzte Länge hat und anschließend Richtung und Geschwindigkeit neu gewählt werden können. Die Ausfallschritte sind im Fall PC wie auch MPC bedingt und können für verschiedene Zeitpunkte eingeplant werden. Es bleibt zu klären, inwiefern die für einen realen Roboter notwendige Echtzeitfähigkeit gegeben ist, und ob die vorgeschlagenen Verfahren tatsächlich eine Verbesserung bezüglich der Stabilität darstellen.

Die Experimente sind so gewählt, dass der Vorteil eines Verfahrens im Vergleich zu dem Fall gezeigt wird, in dem es nicht genutzt wird. So lässt sich der Vorteil des Beobachters separat vom Vorteil der Ausfallschritte bewerten. Zudem werden verschiedene Sensoren untersucht, da zwar der ZMP anhand der Daten verschiedener Sensoren berechnet werden kann, jedoch sich nicht jeder Sensor gleichermaßen eignet.

Eine typische Instabilität von realen Robotern sind unerwünschte Schwingungen des Oberkörpers. Diese treten bei der gewählten Laufsteuerung entlang der Saggitalachse auf und können sich verstärken bis der Roboter fällt. Das Hindernisexperiment [50] hat zum Zweck, eine derartige Schwingung zu provozieren, so dass der Roboter in miteinander vergleichbaren Situationen fällt oder sich stabilisieren kann, je nachdem, ob Sensordaten in den Zustand einfließen oder nicht. Je nach Stärke des Einflusses sind hier auch Ausfallschritte notwendig.

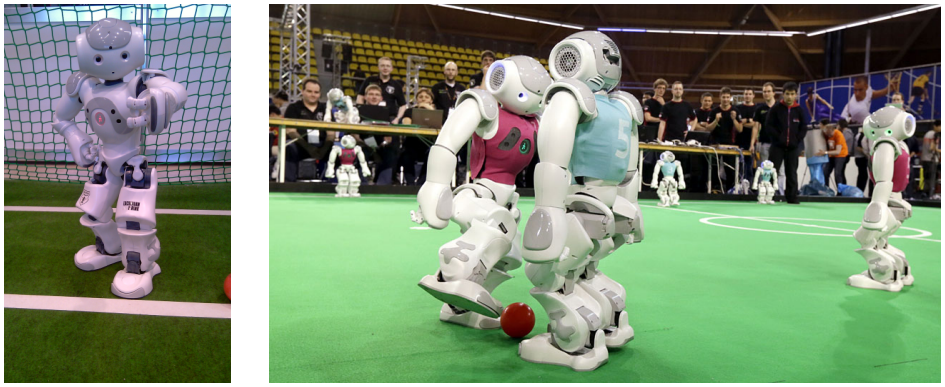

Abbildung 6.1: Nao von Aldebaran Robotics beim Fußballspiel.

Ein Grund für diese Art von Störungen sind fehlerhafte Annahmen im Modell. Das Modellfehlerexperiment [8] erzeugt einen kontrollierten Fehler, indem das Modell vom Roboter in der Simulation absichtlich verfälscht wird.

Des Weiteren sind Kollisionen ein typischer Grund für Störungen, die ebenfalls unterschiedlich stark sein können und daher auch in verschiedenen Stufen ausgeglichen werden müssen. Sie werden hier durch Kollision mit einem ca. 1 kg schweren Pendel evaluiert, bzw. in einer Simulation durch eine zuverlässig definierbare Kraft.

Nach einer Einführung in die genutzte Roboter-Plattform wird zunächst jedoch untersucht, inwiefern die Laufzeiten der Algorithmen bereits einen Einsatz unmöglich machen.

6.1 Roboter-Plattform

Während die bisherigen Kapitel unabhängig von einer Roboter-Plattform sind, wenn auch in einigen Fällen auf die Besonderheiten von kostengünstigen Robotern eingegangen wird, wird hier für die Evaluation der „Nao“ von Aldebaran genutzt, siehe Abbildung 6.1.

Seine Höhe beträgt maximal 573.2 mm und sein Gewicht (inkl. Akku) 5.4 kg. Er besitzt, je nach Version, 25 Freiheitsgrade, wobei jedoch die Beine nur 11 besitzen. Diese Einschränkung befindet sich im Hüftgelenk, das ein Gelenk für beide Beine gemeinsam für die Rotation um die z-Achse besitzt. Das bedeutet, dass der Rotationswinkel der Füße um die z-Achse nicht unabhängig gewählt werden kann. Das hat in den folgenden Experimenten jedoch keine Auswirkungen, da keine Experimente beim Laufen von Kurven stattfinden. Dagegen wichtig ist das Material der Getriebe und Verbindungen, die aus Kunststoffen bestehen, laut Hersteller ABS-PC, PA-66 und XCF-30. Die Auswirkungen auf den Lauf sind für den Anwender leicht ersichtlich: Spiel in den Getrieben und Flexibilitäten, die auch in Experimenten einen nachvollziehbaren Einfluss haben [48].

Der Rechner des Naos besteht aus einem Atom Z530 mit 1.6 GHz und einem 1 GB RAM. Verglichen mit aktuellen PCs oder Mobiltelefonen ist daher die Leistung begrenzt, was ein wichtiger Punkt bei der Rechenzeitanalyse im nächsten Abschnitt ist, da trotz geringerer Leistung die Lösung innerhalb von 10 ms zur Verfügung stehen muss, wobei es wünschenswert ist, nebenbei noch weitere Algorithmen ausführen zu können. Laut Hersteller ist das installierte Linux-Betriebssystem echtzeitfähig, was jedoch nicht ohne weiteres für die Algorithmen dieser Arbeit genutzt werden kann.
Der Nao bietet verschiedene Sensoren, die bereits in Kapitel 4 angesprochen wurden. Die Gelenkwinkel werden durch Magnetic Rotary Encoder (MRE) gemessen, wobei sie so angebracht sind, dass sie den Winkel zwischen den beiden verbundenen Bauteilen inkl. Getriebe messen und damit dessen wichtigen Einfluss beinhalten. Des Weiteren besitzt er eine IMU im Oberkörper und vier Force Sensing Resistor (FSR) je Fuß. Die Sensordaten können 100 mal je Sekunde ausgelesen werden, wobei der Hersteller angibt, dass sie intern mit einer höheren Frequenz ausgelesen und gefiltert werden. Insbesondere wird bereits eine Oberkörperorientierung zur Verfügung gestellt, so dass hier keine eigene Filterung notwendig ist. Das gilt auch für den CoP, der ebenfalls bereits vom Hersteller zur Verfügung gestellt wird, jedoch aufgrund von Fehlern in der Berechnung des Herstellers nicht nutzbar ist, so dass hier die Berechnung wie in Abschnitt 4.1 beschrieben durchgeführt wird.

6.2 Rechenzeitanalyse

Eine wichtige Anforderung ist die Echtzeitfähigkeit, welche hier definiert wird als eine Laufzeit von $<< 10\,\text{ms}$, wobei in dieser Zeit auch eine Lösung garantiert vorliegen muss. In diesem Experiment wird die Laufzeit der C++-Implementierungen auf einem Rechner mit einem Core i5 1.8 GHz Prozessor verglichen.
Bei MPC ist der rechnerisch aufwändige Teil die Lösung des QP-Problems, was die Modifikation der Schritte und die Bestimmung der CoM-Trajektorie beinhaltet. Zur Lösung wird die von Diedam et al. [11] und von Herdt et al. [20] vorgeschlagene Bibliothek qpOASIS[1] verwendet, wobei beide Arbeiten eine durchschnittliche Laufzeit von weniger als 1 ms für die Lösung eines Zeitschrittes angeben. Dies kann in den hier durchgeführten Experimenten bestätigt werden. Allerdings berechnen die Autoren die Laufbewegung nur mit einer Frequenz von 10 Hz, was weit unter den hier gestellten Anforderungen liegt. Bei dieser Frequenz ist im schlechtesten Fall die Reaktion auf eine Störung um 100 ms verzögert, was bei 4 Schritten pro Sekunde bereits nahezu ein halber Schritt ist. Das Ergebnis des gleichen Algorithmus ausgeführt mit 100 Hz ist in Abbildung 6.2 zu sehen. Bei den meisten Zeitpunkten ist der Zeitbedarf zur Lösung nach wie vor gering und liegt bei ca. 2 ms. Der Unterschied zu 1 ms lässt sich damit erklären, dass bei steigender Frequenz von 10 Hz auf 100 Hz auch die

[1] http://set.kuleuven.be/optec/Software/qpOASES-OPTEC

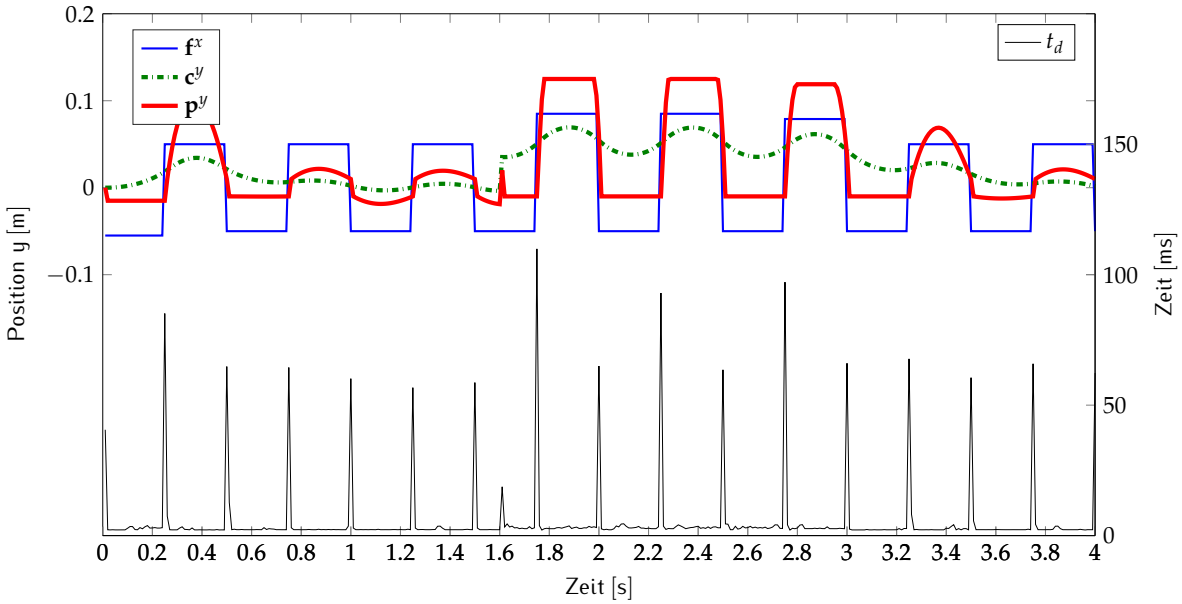

Abbildung 6.2: Ein Lauf erzeugt mittels MPC, seitlich gestört bei 1.6 s. Dazu der Zeitbedarf t_d zur Lösung des QP-Problems.

Vorschau pro Sekunde die zehnfache Menge an Punkten aufweist, aber nicht kürzer wird, so dass der Aufwand der Optimierung entsprechend steigt. Besonders groß wird der Zeitbedarf zu den Zeitpunkten, wo die neue Lösung nicht mehr in der Nähe der alten liegt und entsprechend mehr Iterationen benötigt werden. Das ist bei einem Wechsel des Standfußes wie auch bei unvorhergesehenen Störungen der Fall. Hier kann der Zeitbedarf bei über 100 ms liegen.

Mit der Länge der Vorschau und den wählbaren Gewichtungen hängt auch eine weitere Schwierigkeit zusammen, die Garantie der Lösung. Die Länge der Vorschau muss ausreichend weit in die Zukunft reichen, da veränderte Gegebenheiten nicht nur zu höheren Laufzeiten sondern auch zum Scheitern des Optimierers führen können. Zudem ist die Wahl der Gewichtungen eine bekannte Schwierigkeit von MPC, da von Ihnen nicht nur „das Aussehen“ der Lösung sondern auch die Auffindbarkeit irgendeiner gültigen Lösung abhängt, was bei der manuellen Wahl der Gewichtungen in dieser Evaluation auch festzustellen ist.

Abbildung 6.3 zeigt zum Vergleich einen Lauf mittels PC. Bevor der Lauf startet ist der Zeitbedarf t_d der Module zur Bestimmung des Referenz-ZMP, der Ausfallschritte und der CoM-Bewegung sehr gering, da je Modul nur überprüft wird, ob der Lauf gestartet ist und bis zu Sekunde 4 nichts weiter ausgeführt wird. Es ist zu erkennen, dass aufgrund des Multitaskings des nicht echtzeitfähigen Betriebssystems der Zeitbedarf variiert und einzelne Ausreißer aufweist.

Der Lauf mit einer Zielgeschwindigkeit von $v_x = 10\frac{\text{cm}}{\text{s}}$ startet bei ca. 4 s und wird zunächst beschleunigt, bis die gewünschte Geschwindigkeit erreicht wird, siehe Abschnitt 5.1. In den

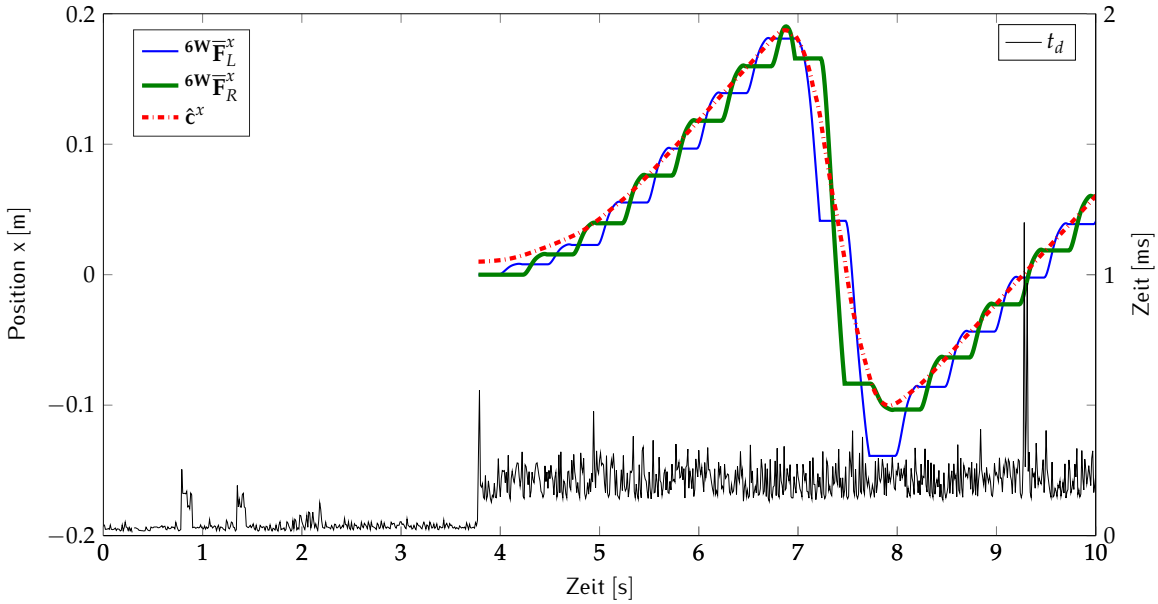

Abbildung 6.3: Lauf bei $v_x = 10\frac{\text{cm}}{\text{s}}$ mit Beschleunigungsphase und einem Stoß von vorne mit 2 Ns bei ≈ 6.5 s, der mittels Ausfallschritte balanciert wird. Dazu der Zeitbedarf t_d der Module zur Bestimmung des Referenz-ZMP, der Ausfallschritte und der CoM-Bewegung.

Abschnitten, wo keine Änderung der Fußposition zu erkennen ist, befindet sich der Fuß am Boden. Ansonsten ist die Bewegung in der Luft zu sehen.
Bei ca. 6.5 s wird der Lauf durch einen Stoß der Stärke 2 Ns gestört, welcher durch Modifikation der folgenden fünf Schritte ausbalanciert wird. Der Zeitbedarf der Lösung liegt auch hier dauerhaft unter einer halben Millisekunde. Auch hier sind die einzelnen Ausreißer aufgrund des Multitaskings zu erkennen, jedoch ist keine Systematik erkennbar, dass bei einem neuen Schritt oder einer Modifikation die Berechnungsdauer steigt.
Messungen auf einem Nao ergeben, dass hier der Laufalgorithmus ca. 4 ms für eine Lösung benötigt. Das zeigt die schwache Leistung des verbauten Atom-Prozessors, ist aber ebenfalls ausreichend schnell.

6.3 Hindernisexperiment

Eines der wesentlichen Ziele ist der Ausgleich von kleineren Störungen, wie beispielsweise die saggitale Schwingung des Oberkörpers. Es wird daher ein Experiment benötigt, in dem auf kontrollier- und nachvollziehbare Weise eine Oberkörperschwingung erzeugt und ausgeglichen wird. Eine offensichtliche Verbesserung tritt ein, wenn der Roboter aufgrund der Störung fällt, während dies mit dem jeweiligen Verfahren nicht der Fall ist.
Im einfachsten Fall wäre der Lauf ohne Nutzung von Sensoren an sich instabil, was jedoch nicht immer der Fall ist. Es ist daher notwendig, eine Schwingung zu provozieren, was mit einer Unebenheit auf dem Boden auf kontrollierte Art erreicht werden kann.

	Simulation	Realer Roboter		
	Beobachter	Beobachter		Ausfallschritte
	Beschl.	Beschl.	FSR	FSR/Beschl./Gyroskop
Mit Beobachter bzw. Ausfallschritte	10/50	5/10	0/10	Siehe Text
Ohne Beobachter bzw. Ausfallschritte	48/50	10/10	10/10	10/10

Tabelle 6.1: Anzahl der Iterationen, in denen der Roboter hingefallen ist, und die Anzahl der insgesamt durchgeführten Iterationen bei den Hindernisexperimenten. Es ist zu beachten, dass die Experimente für Simulation, Beobachter mit realem Roboter und Ausfallschritte unterschiedlich aufgebaut und daher nicht miteinander vergleichbar sind.

Da, wie in Kapitel 4 besprochen, zur Messung von kleineren Störungen der ZMP aus verschiedenen Quellen geeignet ist, wird hier ebenso untersucht, welche Art Sensor sich am besten eignet.

Beobachter in Simulation

Als unbekannte Unebenheit wird in der Simulation eine Erhöhung von einem Zentimeter gewählt, wie in Abbildung 6.4 dargestellt. Der Roboter läuft vorwärts mit einer Geschwindigkeit von $v_x = 20\frac{\text{mm}}{\text{s}}$, so dass die Schrittweite zu klein ist, um mit einem Schritt vollständig auf der Erhöhung aufzusetzen. Diese Störung wird durch einen Beobachter anhand des gemessenen ZMP in den Zustand zurückgeführt. In einer Simulation sind FSR nur unzureichend zu simulieren. Die Werte sind aufgrund der Art, wie Kollisionen detektiert und behandelt werden, stark verrauscht und sind daher als alleinige Quelle zur Stabilisierung in einer Simulation ungeeignet. Es wird daher hier ausschließlich der Beschleunigungssensor zur Bestimmung des ZMP eingesetzt. Aus zwei Gründen wird zudem angenommen, dass der CoM nicht vom Soll abweicht. Zum einen werden keine Ausfallschritte ausgeführt und zum anderen weichen die Gelenkwinkel im Simulator zwar unter Umständen vom Soll ab, allerdings werden hier keine Ungenauigkeiten physischer Roboter simuliert, wie Toleranzen in den Getrieben, Flexibilitäten oder ähnliches [48], so dass die Abweichungen nicht mit denen eines physischen Roboters vergleichbar sind.
Der Roboter startet zudem mit zufällig gewählten und daher jeweils leicht unterschiedlichen Startpositionen, so dass der Fuß an unterschiedlichen Stellen die Kante berührt. Das führt zu unterschiedlichen Oberkörperwinkeln, was anhand von Abbildung 6.4 verdeutlicht wird. Nach wenigen Schritten ist er so weit auf das Hindernis gelaufen, dass er nach vorne kippt und nun auf dem Hindernis läuft. Diese Situation verursacht eine Störung, welche in den meisten Fällen zum Fall führt. Jeweils 50 Durchläufe mit Beobachter und 50 ohne Beobachter

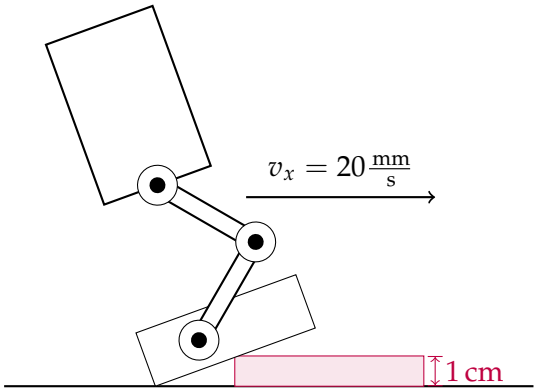

Abbildung 6.4: Schematische Darstellung des Hindernisexperiments in der Simulation.

zeigen den klaren Vorteil in Tabelle 6.1. Ohne Einsatz des Beobachters fällt der Roboter in 96% der Fälle, mit Beobachter nur in 20%.

Beobachter bei realem Roboter

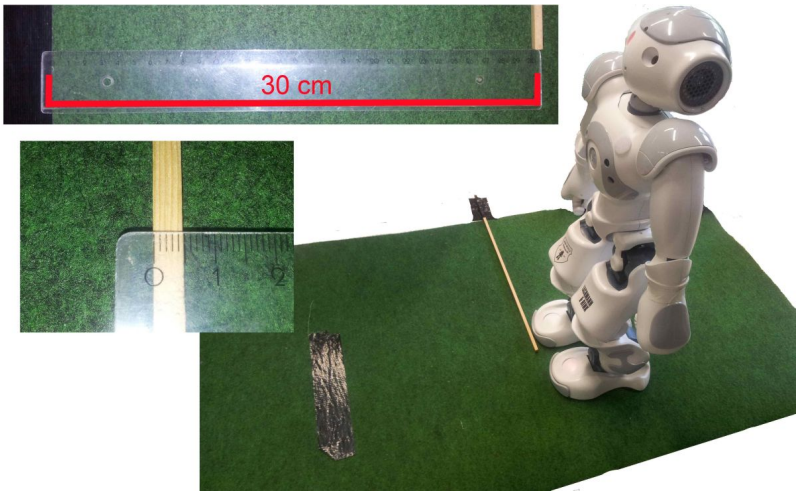

Abbildung 6.5: Aufbau des Hindernisexperiments beim realen Roboter [50]. Ein Balken mit einer Kantenlänge von 5 mm ist der Startpunkt für den Roboter. Der Roboter läuft mit einem Fuß über den Balken, wodurch eine saggitale Schwingung induziert wird. Der Test gilt als erfolgreich, wenn eine Distanz von 30 cm ohne zu fallen zurück gelegt wird.

Der Aufbau des Experiments mit realem Roboter ist ähnlich. Anstatt einer Stufe wird hier ein Balken mit 5 mm Kantenlänge verwendet, was der halben Höhe entspricht, aber dennoch eine zusätzliche Instabilität bedeutet, da der Roboter zusätzlich nach vorne kippt. Dem realen Roboter ist es nicht möglich, mit beiden Füßen das Hindernis zu besteigen, da der Winkel des Oberkörpers zu groß wird, so dass ein rückwärtiger Fall unvermeidbar ist. Das wird später von Interesse sein, jedoch ist zunächst nur eine leichtere Störung das Ziel. Daher läuft der

Roboter nur mit einem Fuß über das Hindernis, was ausreicht, um eine saggitale Schwingung zu provozieren. Er muss danach noch eine Strecke von 30 cm bis zur schwarzen Markierung (siehe Abbildung 6.5) zurücklegen, bevor der Lauf als stabil gewertet wird.

In diesem Experiment wird auch die CoM-Position gemessen, indem eine direkte Kinematik die gemessenen Gelenkwinkel anwendet. Diese Wahl ist hier sinnvoll, da es um kleinere Abweichungen geht, die zu einem großen Teil aus den Ungenauigkeiten der Gelenke herrühren [48], wie auch weiter unten dargelegt wird.

Auf dem realen Roboter liefern FSR wie auch der Beschleunigungssensor Daten, die sich zur Berechnung des ZMP eignen. Es werden daher Experimente ohne Beobachter, mit Messungen vom Beschleunigungssensor und mit Messungen der FSR-Sensoren durchgeführt. Tabelle 6.1 zeigt, dass in keinem Fall der Roboter ohne einen Beobachter die 30 cm zurücklegt. In der Hälfte der Fälle gelingt es mit Daten des Beschleunigungssensors und in allen Fällen, wenn der ZMP aus den FSR-Daten bestimmt wird. Aufgrund dieses klaren Unterschieds und da erwartet werden kann, dass sich dies so fortsetzt, sind hier 10 Iterationen ausreichend.

Das schlechte Abschneiden der Beschleunigungssensoren lässt sich mit starken Störungen in den Daten erklären. Die Beschleunigungssensoren reagieren empfindlich auf alle Erschütterungen des Roboters, die vor allem beim Aufsetzen der Füße auftreten.

Abbildung 6.6 zeigt, wie die Stabilisierung im Falle der Messung mittels FSR erreicht wird. Subjektiv betrachtet scheinen beide Fälle ähnlich instabil zu sein. Allerdings ist die Distanz zwischen der Referenz und dem gemessenen CoP keine geeignete Größe, um darüber zu entscheiden, wie instabil ein Lauf ist, da der CoP auf das Stützpolygon begrenzt ist. Aufgrund dieser Tatsache lässt sich dennoch aus der Abbildung entnehmen, zu welchen Zeitpunkten per ZMP-Definition der Lauf instabil ist, nämlich dann, wenn er den Rand des Stützpolygons erreicht. In diesem Fall ist über einen längeren Zeitraum keine Änderung in der Position festzustellen. Es ist zu erkennen, dass im Fall ohne Nutzung der Sensoren diese Situationen deutlich häufiger bzw. länger auftreten. Zudem ist zu erkennen, dass der CoM-Verlauf dem gemessenen CoP folgt. Beispielsweise weicht bei 1.5 s in Abbildung 6.6 der CoP in negative Richtung ab, woraufhin der Schwerpunkt ebenfalls in diese Richtung differiert. Aus der Gleichung 2.2 zum 3D-LIPM ergibt sich, dass eine geringere Distanz zwischen ZMP und CoM eine geringere Beschleunigung bedeutet. Die Dämpfung der Schwingung stabilisiert demnach den Lauf.

Es ist außerdem in Abbildung 6.6 zu erkennen, dass häufig der Roboter schwankt ohne über die vordere oder hintere Fußkannte zu kippen (was anhand eines konstanten ZMP für kurze Zeit zu erkennen wäre), was bestätigt, dass ein großer Teil der Fehler auf die Gelenke zurückzuführen ist.

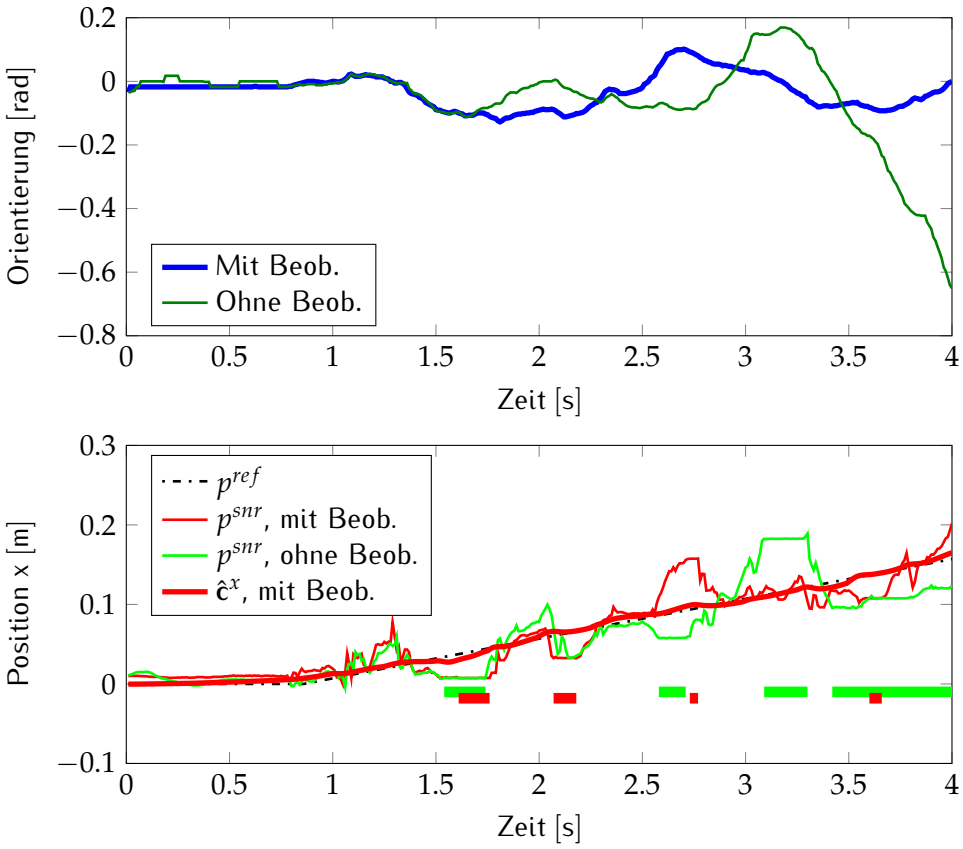

Abbildung 6.6: Laufstabilisierung mit Sensordatenrückführung. Der Roboter tritt zu Beginn des Laufs auf einen Balken, der eine Schwingung des Roboters anregt, welche schließlich zum Fall führt. Zu den Zeitpunkten, wo der Roboter um eine Kante des Fußes kippt, wird ein über mehrere Zeitschritte gleichbleibender ZMP gemessen, in der unteren Abbildung zur Verdeutlichung mit einem farbigen Balken markiert. Die obere Abbildung zeigt die dazugehörige gemessene Oberkörperorientierung.

Schrittmodifikation bei realem Roboter

Im vorherigen Experiment läuft der Roboter nur mit einem Fuß über den Balken, da der Winkel des Oberkörpers sonst derart groß werden würde, dass er schließlich fällt. In diesem Experiment ist das nun gewünscht, denn die Modifikation der Schritte soll nun dazu führen, dass der Roboter aufgrund des steigenden Winkels die Schritte modifiziert, um den Sturz zu vermeiden. In diesem Experiment wird $\lambda_1 = 0.7$ und $\lambda_2 = 0.1$ gewählt, siehe Abschnitt 5.5. Sobald er den Balken betritt, steigt die rückwärtige Neigung des Oberkörpers kontinuierlich an. Ohne die Schritte zu modifizieren wird sie in allen Fällen (10) so groß, dass der Roboter fällt. Ist die Schrittmodifikation aktiviert, werden aufgrund der rückwärtigen Neigung die Schritte in die selbe Richtung modifiziert, so dass er rückwärts läuft. Dadurch verringert sich die Neigung und auch die Modifikation, so dass er kurzzeitig wieder vorwärts auf den Balken läuft. Dies würde sich nun unbegrenzt wiederholen. Jedoch ändert der Roboter ungewollt

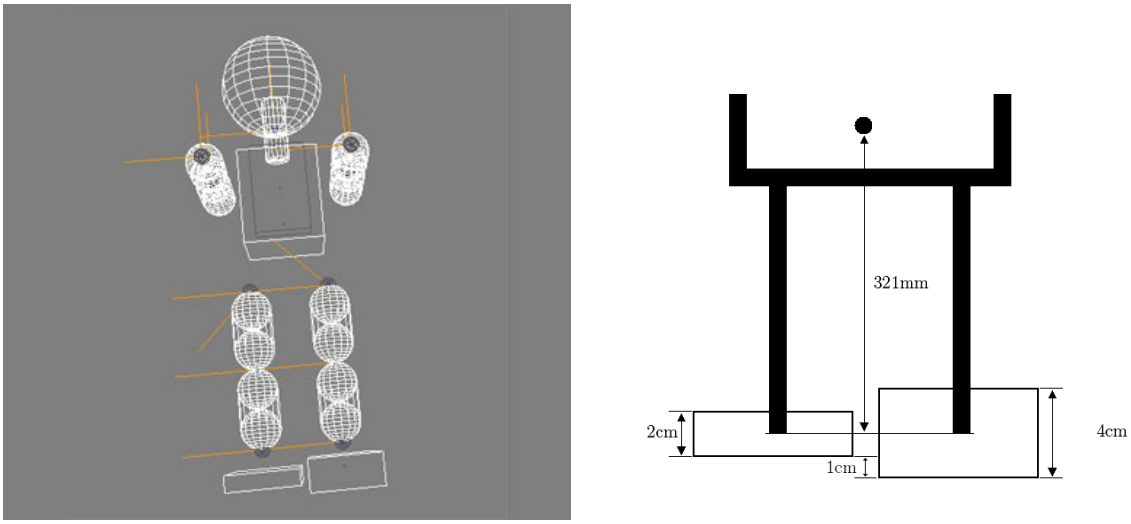

Abbildung 6.7: Änderung für das Modellfehlerexperiment. Die Höhe des linken Fußes wird verdoppelt ohne die entsprechende Anpassung im Modell der Kinematik vorzunehmen.

zusätzlich langsam seine Laufrichtung. Ist die Laufrichtung des Roboters nicht mehr senkrecht zum Balken, steigt der Oberkörperwinkel weniger stark an. Das führt dazu, dass er es nach einigen Versuchen doch schafft, den Balken zu passieren. Da diese Änderung der Laufrichtung zwar reproduzierbar aber ungewollt und zufällig ist, lässt sich in diesem Experiment der Erfolg nicht quantifizieren. Dennoch lässt sich feststellen, dass sich hier ein Sturz durch Modifikation der Schritte zuverlässig vermeiden lässt.

6.4 Modellfehlerexperiment

Eine typische Quelle für Störungen aus dem vorherigen Abschnitt sind Modellannahmen, die stark von der Realität abweichen. Dazu zählen nicht nur Abweichungen vom 3D-LIPM, sondern auch Abweichungen im Modell der Kinematik und fehlerhaft umgesetzte Gelenkwinkel. In einer gewöhnlichen Simulation treten die beiden letzteren Punkte nicht auf, da der Roboter wie von der Kinematik angenommen modelliert werden kann und Gelenkwinkel zumeist exakt umgesetzt werden (siehe oben).

Ziel dieses Experiments ist daher, in einer Simulation eine Abweichung vom Modell zu implementieren um festzustellen, ob die Sensorrückführung den Fehler ausgleichen kann. Dazu wird wie in Abbildung 6.7 zu sehen die Höhe des linken Fußes verdoppelt, so dass der Roboter links um 1 cm erhöht steht. Dies verursacht eine Störung im Lauf, die in allen 10 Versuchen nach kurzer Zeit zum Sturz führt, wie in Tabelle 6.2 dargestellt. Mit Rückführung von Sensordaten war auch nach einer Minute noch kein Sturz aufgetreten.

Durchgang	1	2	3	4	5
Ohne Sensoren	6.98 s	6.96 s	7.16 s	10.34 s	8.26 s
Mit Sensoren	> 60 s	> 60 s	> 60 s	> 60 s	> 60 s
Durchgang	6	7	8	9	10
Ohne Sensoren	9.32 s	8.36 s	6.98 s	8.34 s	6.98 s
Mit Sensoren	> 60 s	> 60 s	> 60 s	> 60 s	> 60 s

Tabelle 6.2: Laufzeit des Roboters bis zum Sturz mit und ohne Rückführung von Sensordaten. Im letzteren Fall stürzte der Roboter nach spätestens 11 Sekunden, während mit Sensordatenrückführung auch nach einer Minute der Roboter nicht fiel.

6.5 Kollisionsexperiment

Dieses Experiment dient primär zur Evaluation der Schrittmodifikation, indem eine Störung verursacht wird, die größer ist als in den vorherigen Experimenten, nur kurzzeitig auftritt und stark genug ist, um den Roboter sofort zu Fall zu bringen[2]. Derartige Störungen können nicht durch Dämpfung ausgeglichen werden, da keine steigende Amplitude einer Schwingung die Ursache ist. Da bei größeren Störungen der Roboter um eine Kante der Füße rotiert, wird in diesem Experiment die IMU verwendet, um die CoM-Position zu messen. Wie zuvor festgestellt, sind die FSR auf einem realen Roboter besser zur Messung des CoP geeignet als Beschleunigungssensoren. Daher werden die folgenden Experimente grundsätzlich mit dem ZMP aus FSR-Messungen durchgeführt. Dies führt in der Simulation zu keinem Nachteil, da die ZMP-Messungen für die Berechnung der Modifikation eine untergeordnete Rolle spielen. Zudem wird in diesem Experiment $\lambda_1 = 0.7$ und $\lambda_2 = 0.1$ gewählt, siehe Abschnitt 5.5.

Simulation

In der Simulation lässt sich ein vorher festgelegter Impuls in einer reproduzierbaren Situation auf den Roboter einwirken. Ein bei $\approx 6.5\,\mathrm{s}$ ausgeübter Impuls von $2\,\mathrm{Ns}$ ist ausreichend, um einen Roboter, der mit einer Geschwindigkeit von $v_x = 10\frac{\mathrm{cm}}{\mathrm{s}}$ läuft, sofort zu Fall zu bringen. Das Experiment, zu sehen in Abbildung 6.8, zeigt, dass die Ausfallschritte auch hier in der Lage sind, den Sturz zuverlässig zu verhindern.

Realer Roboter

Die Situation bei einem realen Roboter ist deutlich komplexer. Zunächst muss ein Aufbau gefunden werden, bei dem der Impuls ebenso zuverlässig kontrolliert ausgeübt wird. Es wird

[2] Unter einer Störung, die zum sofortigen Fall führt, wird hier eine Störung verstanden, die eine Rotation des Oberkörpers um die x- oder y-Achse verursacht, die zum Fall führt, ohne dass die Rotation vorher ihre Richtung ändert.

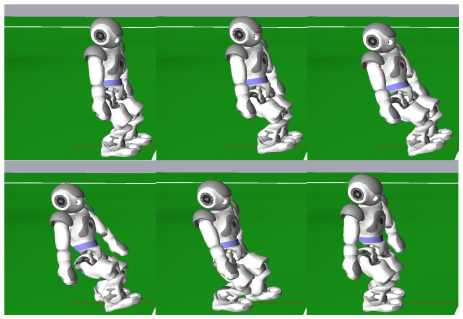

Abbildung 6.8: Während des simulierten Laufs wird der Roboter von einer Kraft von 2 Ns nach hinten gestoßen. Der Roboter kann diese Störung nur mit Hilfe der Schrittmodifikation ausbalancieren.

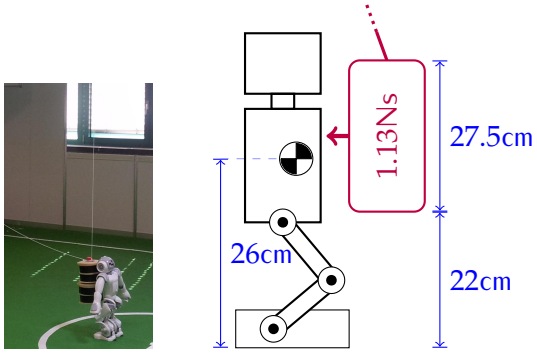

Abbildung 6.9: Aufbau des Pendelexperiments.

dazu ein an der Decke befestigtes Pendel verwendet, welches einen Impuls von 1.13 Ns ausübt, wie in Abbildung 6.9 zu sehen. Um die Streuung weiter zu verringern, läuft der Roboter auf der Stelle. Das Experiment wurde insgesamt 60 mal durchgeführt, 30 Wiederholungen mit Modifikation und 30 ohne. Es zeigt sich, dass trotz der getroffenen Maßnahmen der Roboter unterschiedlich stark getroffen wird und in 11 der 30 Wiederholungen sofort stürzt, falls keine Modifikation statt findet. Werden die Schritte modifiziert, stürzt er in keinem Fall sofort. Jedoch führen die Ausfallschritte zu einer Bewegung auf dem Feld, die teilweise zu einer weiteren Kollision mit dem Pendel führt (durch Modifikationen, durch die sich der Roboter in Richtung Pendel bewegt) und teilweise auch an Übersteuern erinnern. Dies kann zu einem Aufschwingen führen, welches, zusammen mit den zusätzlichen Kollisionen, in 5 Fällen zu einem Sturz führte.

Abbildung 6.10 zeigt den Moment des Stoßes und danach. Es ist zu erkennen, dass Modifikationen von ca. $-1.2\,\mathrm{cm}$ vor dem Stoß bei ca. 2.2 s vorgenommen werden, die mit den Ungenauigkeiten des Roboters erklärt werden können. Im ersten Schritt nach der Störung wird eine deutlich größere Modifikation von ca. $-8.4\,\mathrm{cm}$ vorgenommen und im zweiten Schritt nach der Störung von $-5.2\,\mathrm{cm}$, die zu einem gemessenen CoP-Verlauf führen, der trotz der

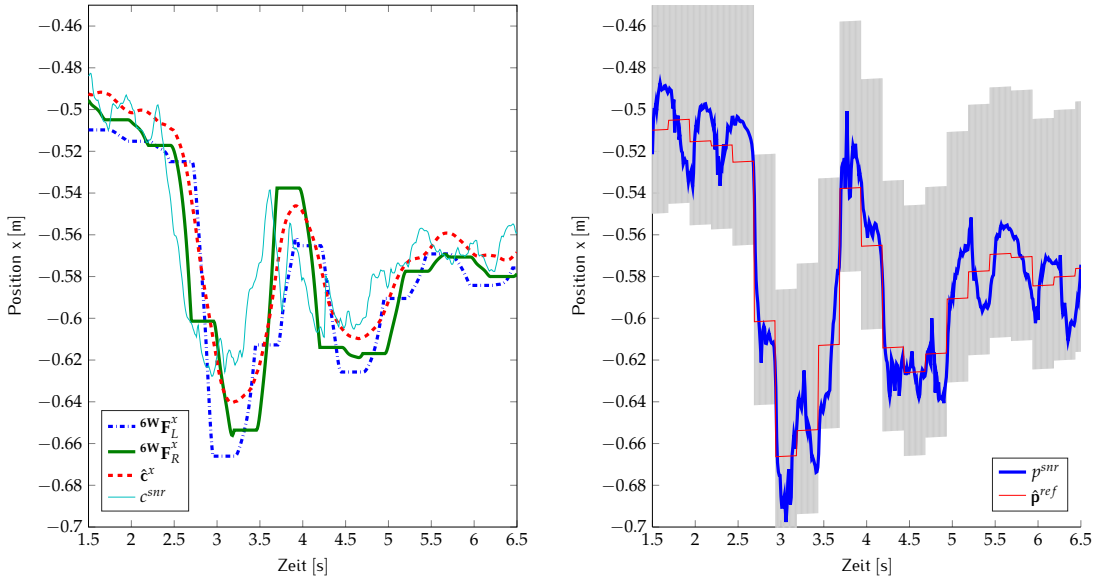

Abbildung 6.10: Stoß des Pendels und nachfolgende Stabilisierung mittels Schrittmodifikation. Links sind die Positionen der Füße entlang der x-Achse dargestellt (${}^{6W}\mathbf{F}_L^x$ bzw. ${}^{6W}\mathbf{F}_R^x$), rechts der gemessene CoP, grau hinterlegt zur Darstellung des jeweils aktuellen Stützpolygons.

großen Störung innerhalb des Stützpolygons liegt. Der Roboter ist daher aufgrund der korrekt geplanten Modifikationen stabil.

6.6 Fazit

Die Experimente zeigen, dass die gestellten Anforderungen erfüllt werden und geben zudem wichtige Erkenntnisse über die Eignung verschiedener Sensortypen und den Grund des stabilisierenden Effekts. Während sich in der Simulation die Beschleunigungssensoren aufgrund des negativen Einflusses der FSR auf die Laufstabilität als besser geeignet erweisen, sind sie auf dem realen Roboter weniger geeignet, da Rauschen und unerwünschte Effekte die Messungen ungenau machen. Hier ist eine Fusion aus CoM-Position, gemessen mittels Gelenkwinkel, und CoP, gemessen mittels FSR, die beste Wahl. Bei größeren Störungen, die zum sofortigen Fall führen können, zeigen sich Ausfallschritte als zusätzliches Mittel zur Stabilisierung als geeignet, jedoch sollte der Schwerpunkt dann mittels IMU statt durch Gelenkwinkelmessungen bestimmt werden, da letztere kein Kippen des Roboters feststellen können.

Zudem zeigen die Experimente den Grund für die Stabilisierung. Während intuitiv häufig angenommen wird, dass ein kippender Roboter durch „Gegensteuern“ stabilisiert werden kann, kann hier gezeigt werden, dass eine Dämpfung der Schwingung den gewünschten Effekt

zeigt. Größere Störungen hingegen können nur durch Modifikation der Schritte ausgeglichen werden, so dass der tatsächliche ZMP dadurch innerhalb des Stützpolygons bleibt.

KAPITEL 7

Fazit und Ausblick

Humanoides Laufen ist nicht nur aufgrund der begrenzten Rechenkapazität eine große Herausforderung. Auch wenn sie unbegrenzt wäre, würde die Wissenschaft noch Jahre brauchen, um ein genaues Modell von Robotern zu entwickeln. Viele physikalische Aspekte werden derzeit intensiv erforscht, wie beispielsweise die Kontrolle von Robotern mit flexiblen Elementen und realistischen Getrieben [49]. Selbst Simulationen mit derartigen Fähigkeiten sind noch in der Erforschung [48].

Jedoch selbst wenn ein solches Modell zur Verfügung stehen würde, gäbe es unvorhersehbare Dinge, wie Stöße von externen Aktoren oder auch unvorhersehbare Bodenbeschaffenheit. Um all diese Aspekte zu berücksichtigen, werden in der Einleitung Ziele formuliert, die ein stabiles Laufen eines realen Roboters in unbekannter Umgebung ermöglichen sollen, bei kleineren wie auch größeren Störungen.

Zwei vielversprechende Ansätze werden daraufhin implementiert beziehungsweise erweitert: Ein Laufalgorithmus basierend auf Model Predictive Control (MPC), welcher bereits die Modifikation anhand von Sensordaten beinhaltet, und ein Algorithmus basierend auf Preview Control, welches um ein Verfahren ergänzt wird, das die Fußpositionen modifiziert, und als ZMP/IP Controller bezeichnet wird. Während sich der MPC-Ansatz als zu rechenintensiv für hohe Regelfrequenzen erweist, ist der ZMP/IP Controller nachweislich in der Lage, kleinere Störungen, insbesondere Schwingungen des Oberkörpers, durch Dämpfung auszugleichen. Auch die Modifikation der Schritte bei größeren Störungen ist erfolgreich, wie der Stoß von einem Pendel, aber auch bei unmodellierten Unebenheiten des Bodens. Die Laufzeit ist dabei gering, so dass auch andere Algorithmen, die zum Betrieb eines Roboters notwendig sind, beispielsweise die Wahrnehmung, genug Rechenleistung zur Verfügung haben. Zudem sind die Lösungen garantiert, es müssen also keine unlösbaren Situationen befürchtet werden.

Zusammen erweist sich der ZMP/IP Controller daher nicht nur als die geeignetere Variante, sondern als die einzig anwendbare. Im Jahr 2010 war bereits eine Vorgängerversion in der Lage, den dokumentierten Geschwindigkeitsrekord von $445\frac{\text{mm}}{\text{s}}$ aufzustellen [50]. Allerdings zeigt sich in der Praxis, dass ein optimaler Lauf aufgrund der extremen Abstraktion des Modells nicht immer dann gegeben ist, wenn es laut Modell der Fall sein sollte. Der Algorithmus kann mit Hilfe von über 100 Werten parametrisiert werden, wovon die wesentlichen hier vorgestellt wurden. Darunter die Form des Referenz-ZMP, die Schritthöhe, die Schrittfrequenz usw. Einige Ungenauigkeiten des Modells können mit Hilfe einer geschickten Wahl dieser Werte ausgeglichen werden. So zeigt sich, dass die Servos des Naos nicht schnell genug die notwendigen Drehmomente erzeugen, um einen Standbeinwechsel wie geplant durchzuführen. Es hat sich daher bei den meisten Nutzern des Nao durchgesetzt, eine hohe Schrittfrequenz zu wählen, so dass die Amplitude der seitlichen Schwingung klein gehalten werden kann. Zudem ist die Schritthöhe ein wesentlicher Faktor. Ist sie zu gering, ist der Schwungfuß nicht in der Lage vom Boden abzuheben. Auch die Möglichkeit, den Referenz-ZMP frei wählen zu können, erweist sich in der Praxis als unverzichtbar. Referenzverläufe, wie ein konstanter ZMP in der Fußmitte, führen zu derart starken Schwingungen, dass allein dadurch ein Lauf unmöglich sein kann.

Außerdem zeigt die Diskussion in Abschnitt 5.7, dass die Schwerpunkt-Position bzw. die Gelenkwinkel als Eingabe für den Roboter ungeeignet sind, um derart hochentwickelte Algorithmen zu verwenden. Eine Regelung mittels gewünschtem Drehmoment könnte hilfreich sein, um die zuvor diskutierten Schwierigkeiten (unzureichende Drehmomente) bei der seitlichen Schwingung zu vermeiden. Auch wäre es dann denkbar, die üblichen Regelungsschleifen, die direkt an den Servos arbeiten, durch eine hochfrequente globale Schleife zu ersetzen, die auf Basis von PC oder MPC arbeitet und ein Modell des gesamten Roboters beinhaltet. Dadurch können Ungenauigkeiten an einem Gelenk durch Anpassung der gesamten Bewegung ausgeglichen werden.

ANHANG A

Herleitung der MPC-Zielfunktion

In diesem Abschnitt wird die Herleitung der Zielfunktion in Form von Gleichung 3.1 dargestellt. Dazu werden die einzelnen Summanden von Gleichung 3.16 umgeformt und $\mathbf{Q}$ oder $\mathbf{b}_k^T$ zugeordnet. Die Zuordnung richtet sich nach dem Vorkommen von $\dddot{\mathbf{X}}_k$ bzw. $\mathbf{F}_k$, wobei ein Term $\mathbf{Q}$ zugeordnet wird, wenn einer der beiden Faktoren voran- und nachgestellt vorkommt und $\mathbf{b}_k^T$, falls er nur nachgestellt vorkommt.

Der Summand $\frac{1}{2}|\ddot{\mathbf{X}}_k|^2$ kann in Gleichung 3.1 durch die Einheitsmatrix erreicht werden, wie in Gleichung 3.19 zu sehen ist. Der Summand $\frac{\alpha}{2}|\dot{\mathbf{X}}_{k+1}|^2$ muss zunächst umgeformt werden:

$$
\begin{aligned}
\frac{\alpha}{2}|\dot{\mathbf{X}}_{k+1}|^2 &= \frac{\alpha}{2}\dot{\mathbf{X}}_{k+1}^T\dot{\mathbf{X}}_{k+1} = \frac{\alpha}{2}\left(\mathbf{A_S}\mathbf{x}_k + \mathbf{B_S}\dddot{\mathbf{X}}_k\right)^T\left(\mathbf{A_S}\mathbf{x}_k + \mathbf{B_S}\dddot{\mathbf{X}}_k\right) = \\
&= \frac{\alpha}{2}\left(\mathbf{x}_k^T{\mathbf{A_S}}^T + \dddot{\mathbf{X}}_k^T{\mathbf{B_S}}^T\right)\left(\mathbf{A_S}\mathbf{x}_k + \mathbf{B_S}\dddot{\mathbf{X}}_k\right) = \\
&= \frac{\alpha}{2}\left(\mathbf{x}_k^T{\mathbf{A_S}}^T\mathbf{A_S}\mathbf{x}_k + \mathbf{x}_k^T{\mathbf{A_S}}^T\mathbf{B_S}\dddot{\mathbf{X}}_k + \dddot{\mathbf{X}}_k^T{\mathbf{B_S}}^T\mathbf{A_S}\mathbf{x}_k + \dddot{\mathbf{X}}_k^T{\mathbf{B_S}}^T\mathbf{B_S}\dddot{\mathbf{X}}_k\right) \\
&= \alpha\left(\underbrace{\frac{1}{2}\mathbf{x}_k^T{\mathbf{A_S}}^T\mathbf{A_S}\mathbf{x}_k}_{\text{konstant}}\underbrace{+\dddot{\mathbf{X}}_k^T{\mathbf{B_S}}^T\mathbf{A_S}\mathbf{x}_k}_{\rightarrow\mathbf{b}_k^T} + \underbrace{\frac{1}{2}\dddot{\mathbf{X}}_k^T{\mathbf{B_S}}^T\mathbf{B_S}\dddot{\mathbf{X}}_k}_{\rightarrow\mathbf{Q}}\right).
\end{aligned}
\tag{A.1}
$$

Es ist zu erkennen, dass aufgrund der $\dddot{\mathbf{X}}_k$ der zweite Summand $\mathbf{b}_k^T$ und der dritte $\mathbf{Q}$ zugeordnet werden kann. Der erste Summand ist konstant und kann daher vernachlässigt werden.

Der Summand $\frac{\beta}{2}|\mathbf{X}_{k+1} - \mathbf{X}_{k+1}^{ref}|^2$ wird zunächst auf vergleichbare Weise umgeformt und die daraus folgenden Summanden wie oben entsprechend zugeordnet, wenn sie nicht konstant sind und daher entfallen:

$$\frac{\beta}{2}|\mathbf{X}_{k+1}-\mathbf{X}_{k+1}^{ref}|^2=\frac{\beta}{2}\left(\left(\mathbf{X}_{k+1}-\mathbf{X}_{k+1}^{ref}\right)^T\left(\mathbf{X}_{k+1}-\mathbf{X}_{k+1}^{ref}\right)\right)= \tag{A.2}$$

$$=\frac{\beta}{2}\left(\left(\mathbf{A_P}\mathbf{x}_k+\mathbf{B_P}\dddot{\mathbf{X}}_k-\mathbf{X}_{k+1}^{ref}\right)^T\left(\mathbf{A_P}\mathbf{x}_k+\mathbf{B_P}\dddot{\mathbf{X}}_k-\mathbf{X}_{k+1}^{ref}\right)\right)= \tag{A.3}$$

$$=\frac{\beta}{2}\left(\left(\mathbf{x}_k^T\mathbf{A_P}^T+\dddot{\mathbf{X}}_k^T\mathbf{B_P}^T-\mathbf{X}_{k+1}^{ref\ T}\right)\left(\mathbf{A_P}\mathbf{x}_k+\mathbf{B_P}\dddot{\mathbf{X}}_k-\mathbf{X}_{k+1}^{ref}\right)\right)= \tag{A.4}$$

$$=\underbrace{\frac{\beta}{2}\mathbf{x}_k^T\mathbf{A_P}^T\mathbf{A_P}\mathbf{x}_k}_{\text{konstant}}\underbrace{+\beta\mathbf{x}_k^T\mathbf{A_P}^T\mathbf{B_P}\dddot{\mathbf{X}}_k}_{\rightarrow\mathbf{b}_k^T}\underbrace{-\beta\mathbf{x}_k^T\mathbf{A_P}^T\mathbf{X}_{k+1}^{ref}}_{\text{konstant}}\underbrace{+\frac{\beta}{2}a\dddot{\mathbf{X}}_k^T\mathbf{B_P}^T\mathbf{B_P}\dddot{\mathbf{X}}_k}_{\rightarrow\mathbf{Q}}-$$
$$\underbrace{-\frac{\beta}{2}\mathbf{X}_{k+1}^{ref\ T}\mathbf{A_P}\mathbf{x}_k}_{\text{konstant}}\underbrace{-\beta\mathbf{X}_{k+1}^{ref\ T}\mathbf{B_P}\dddot{\mathbf{X}}_k}_{\rightarrow\mathbf{b}_k^T}\underbrace{+\frac{\beta}{2}\mathbf{X}_{k+1}^{ref\ T}\mathbf{X}_{k+1}^{ref}}_{\text{konstant}}$$

Der letzte Summand der Zielfunktion 3.16 betrifft die Modifikation der Fußpositionen:

$$\frac{\gamma}{2}|\mathbf{F}_k-\mathbf{F}_k^{ref}|^2=\frac{\gamma}{2}\left(\mathbf{F}_k-\mathbf{F}_k^{ref}\right)^T\left(\mathbf{F}_k-\mathbf{F}_k^{ref}\right)= \tag{A.5}$$

$$=\frac{\gamma}{2}\left(\mathbf{F}_k{}^T-\mathbf{F}_k^{ref\,T}\right)\left(\mathbf{F}_k-\mathbf{F}_k^{ref}\right)= \tag{A.6}$$

$$=\gamma\left(\underbrace{-\mathbf{F}_k^{ref\,T}\mathbf{F}_k}_{\rightarrow\mathbf{b}_k^T}\underbrace{+\frac{1}{2}\mathbf{F}_k^{ref\,T}\mathbf{F}_k^{ref}}_{\text{konstant}}\underbrace{+\frac{1}{2}\mathbf{F}_k{}^T\mathbf{F}_k}_{\rightarrow\mathbf{Q}}\right). \tag{A.7}$$

Zusammegefasst ergeben sich somit die Gleichungen 3.19 und 3.20.

ANHANG B

Herleitung der Vorhersagematrizen von Model Predictive Control

Im Folgenden werden die Vorhersagematrizen der Gleichungen 3.6, 3.7, 3.10, 3.11, 3.14 und 3.15 hergeleitet. Die beiden ersteren sind Teil von Gleichung 3.5, der Vorhersage der CoM-Positionen $\mathbf{X}_{k+1}$ basierend auf dem aktuellen Zustand und den Änderungen der Beschleunigungen des Schwerpunktes: $\mathbf{X}_{k+1} = \mathbf{A_P}\mathbf{x}_k + \mathbf{B_P}\dddot{\mathbf{X}}_k$.
Die Matrizen der Gleichungen 3.10 und 3.11 finden sich in Gleichung 3.9, die ebenfalls eine Vorhersage basierend auf den gleichen gegebenen Größen darstellt, jedoch für die CoM-Geschwindigkeiten: $\dot{\mathbf{X}}_{k+1} = \mathbf{A_S}\mathbf{x}_k + \mathbf{B_S}\dddot{\mathbf{X}}_k$. Die Herleitung lässt sich daher für beide gemeinsam zeigen.
Die erste Komponente der Vektoren $\mathbf{X}_{k+1}$ und $\dot{\mathbf{X}}_{k+1}$ ist $\mathbf{c}^x_{k+1}$ bzw. $\dot{\mathbf{c}}^x_{k+1}$, die auch Teil des Zustandes $\mathbf{x}_{k+1} = \left(\mathbf{c}^x_{k+1}, \dot{\mathbf{c}}^x_{k+1}, \ddot{\mathbf{c}}^x_{k+1}\right)$ sind. Der Ausgangspunkt ist demnach das Zustandsraummodell, Gleichung 3.2: $\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\dddot{\mathbf{c}}^x_k = \left(\mathbf{c}^x_{k+1}, \dot{\mathbf{c}}^x_{k+1}, \ddot{\mathbf{c}}^x_{k+1}\right)$. Daraus folgt bereits die erste Zeile von $\mathbf{A_P}$: $\left(1, \Delta t, \frac{\Delta t^2}{2}\right)$ und die erste Zeile von $\mathbf{A_S}$: $(0, 1, \Delta t)$, was der ersten bzw. zweiten Zeile von $\mathbf{A}$ entspricht. Da für $\mathbf{c}^x_{k+1}$ bzw. $\dot{\mathbf{c}}^x_{k+1}$ nur die erste Komponente von $\dddot{\mathbf{X}}_k$ multipliziert mit $\mathbf{B}$ benötigt wird, lautet die erste Zeile von $\mathbf{B_P}$ und $\mathbf{B_S}$ $\left(\frac{\Delta t^3}{6}, 0, \ldots, 0\right)$ bzw. $\left(\frac{\Delta t^2}{2}, 0, \ldots, 0\right)$.
Die zweite Zeile der Vektoren $\mathbf{X}_{k+1}$ und $\dot{\mathbf{X}}_{k+1}$ beinhalten $\mathbf{c}^x_{k+2}$ bzw. $\dot{\mathbf{c}}^x_{k+2}$, so dass hier der Zustand $\mathbf{x}_{k+2}$ zum Zeitpunkt $k+2$ von Interesse ist. Dieser lässt sich auch basierend auf dem Zustand $\mathbf{x}_k$ zum Zeitpunkt k durch rekursives Einsetzen von 3.2 darstellen:

$$\begin{aligned}
\mathbf{x}_{k+2} &= \mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\dddot{\mathbf{c}}^x_{k+1} && \text{(B.1)}\\
&= \mathbf{A}\left(\mathbf{A}\mathbf{x}_k + \mathbf{B}\dddot{\mathbf{c}}^x_k\right) + \mathbf{B}\dddot{\mathbf{c}}^x_{k+1} && \text{(B.2)}\\
&= \mathbf{A}^2\mathbf{x}_k + \mathbf{A}\mathbf{B}\dddot{\mathbf{c}}^x_k + \mathbf{B}\dddot{\mathbf{c}}^x_{k+1}. && \text{(B.3)}
\end{aligned}$$

Daraus ließe sich bereits die zweite Zeile der Vorhersagematrizen ableiten, jedoch ist es sinnvoll zunächst eine Systematik offen zu legen:

$$\begin{aligned}
\mathbf{x}_{k+3} &= \mathbf{A}\mathbf{x}_{k+2} + \mathbf{B}\ddot{\mathbf{c}}^{x}_{k+1} && \text{(B.4)}\\
&= \mathbf{A}\left(\mathbf{A}^2\mathbf{x}_k + \mathbf{A}\mathbf{B}\ddot{\mathbf{c}}^{x}_{k} + \mathbf{B}\ddot{\mathbf{c}}^{x}_{k+1}\right) + \mathbf{B}\ddot{\mathbf{c}}_{k+2} && \text{(B.5)}\\
&= \mathbf{A}^3\mathbf{x}_k + \mathbf{A}^2\mathbf{B}\ddot{\mathbf{c}}^{x}_{k} + \mathbf{A}\mathbf{B}\ddot{\mathbf{c}}^{x}_{k+1} + \mathbf{B}\ddot{\mathbf{c}}_{k+2}. && \text{(B.6)}
\end{aligned}$$

Allgemein gilt demnach für den Zustand zu einen Zeitpunkt $k+i$:

$$\mathbf{x}_{k+i} = \mathbf{A}^i\mathbf{x}_k + \mathbf{A}^{i-1}\mathbf{B}\ddot{\mathbf{c}}^{x}_{k} + \cdots + \mathbf{A}^0\mathbf{B}\ddot{\mathbf{c}}_{k+i-1}. \quad \text{(B.7)}$$

Daraus folgt, dass zur Bestimmung der Zeilen der Vorhersagematrizen die Matrizen $\mathbf{A}^j$ und $\mathbf{A}^j\mathbf{B}$ benötigt werden (Beweis per Induktion):

$$\mathbf{A}^2 = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}^2 = \begin{bmatrix} 1 & 2\Delta t & \frac{4\Delta t^2}{2} \\ 0 & 1 & 2\Delta t \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{(B.8)}$$

$$\mathbf{A}^j = \begin{bmatrix} 1 & j\Delta t & \frac{(j\Delta t)^2}{2} \\ 0 & 1 & j\Delta t \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{(B.9)}$$

$$\begin{aligned}
\mathbf{A}^{j+1} &= \begin{bmatrix} 1 & j\Delta t & \frac{(j\Delta t)^2}{2} \\ 0 & 1 & j\Delta t \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} && \text{(B.10)}\\
&= \begin{bmatrix} 1 & \Delta t + j\Delta t & \frac{\Delta t^2}{2} + j\Delta t\Delta t + \frac{(j\Delta t)^2}{2} \\ 0 & 1 & \Delta t + j\Delta t \\ 0 & 0 & 1 \end{bmatrix}.
\end{aligned}$$

Die obere rechte Komponente lässt sich wie folgt auflösen:

$$\begin{aligned}
\frac{\Delta t^2}{2} + j\Delta t\Delta t + \frac{(j\Delta t)^2}{2} &= \left(j + \frac{1}{2} + \frac{1}{2}j^2\right)\Delta t^2 && \text{(B.11)}\\
&= \frac{1}{2}\left(j^2 + 2j + 1\right)\Delta t^2\\
&= \frac{\left((j+1)\,\Delta t\right)^2}{2},
\end{aligned}$$

womit der Induktionsschritt gezeigt wäre. Des weiteren gilt:

$$\mathbf{A}^j\mathbf{B} = \begin{bmatrix} \left(\frac{j(j+1)}{2} + \frac{1}{6}\right)\Delta t^3 \\ \left(j + \frac{1}{2}\right)\Delta t^2 \\ \Delta t \end{bmatrix}. \quad \text{(B.12)}$$

Gleichung B.7 ließe sich auch als drei separate Zeilen darstellen für Position, Geschwindigkeit und Beschleunigung. Für die Vorhersage der CoM-Position zum Zeitpunkt $k+i$ ist die erste Zeile relevant, und daher auch die ersten Zeilen von $\mathbf{A}^i$ und $\mathbf{A}^j\mathbf{B}$, welche den Matrizen $\mathbf{A_P}$ und $\mathbf{B_P}$ zugeordnet werden müssen. Dies gilt analog für die Vorhersage der Geschwindigkeit.

Aus Gleichung B.7 lässt sich auch die Dimension der Vorhersagematrizen entnehmen. Zunächst hat der Zustandsvektor die Länge 3. Des Weiteren wird für die erste Zeile von $\mathbf{A_P}$ von $\mathbf{A}^j$ die erste Zeile mit $j = 1$ verwendet, da die erste Vorhersage für den Zeitpunkt $k+1$ benötigt wird. Da die Länge der Vorhersage N ist, ist der letzte Zeitpunkt $k+N$. Die Dimension von $\mathbf{A_P}$ ist daher $N \times 3$. Es folgt Gleichung 3.6.

Die Zeilen der Matrix $\mathbf{B_P}$ bestehen, wie der Gleichung B.7 zu entnehmen ist, aus der ersten Komponente von $\mathbf{A}^j\mathbf{B}$ für $j = i-1, \ldots, 0$, alle weiteren sind 0 bis die Länge von $\dddot{\mathbf{X}}_k$ erreicht ist. Zudem ist auch die Zahl der Zeilen wie oben N und damit die Dimension $N \times N$. Es folgt Gleichung 3.7. Analog lassen sich auch die Gleichungen 3.10 und 3.11 herleiten.

Die Herleitung zur Bestimmung der ZMP-Vorhersage per Gleichung 3.13 ist nur teilweise zur obigen Herleitung ähnlich, da sie nicht auf dem Zustandsraummodell basiert, sondern auf dem zugrunde liegenden Modell 3D-LIPM (Gleichung 2.1), hier als Vektor $\mathbf{m} = \left(1, 0, -\frac{\mathbf{c}^z}{g}\right)$ multipliziert mit dem Zustand:

$$
\begin{aligned}
\mathbf{p}^x_{k+1} &= \mathbf{m}\mathbf{x}_{k+1} && \text{(B.13)}\\
&= \mathbf{m}\left(\mathbf{A}\mathbf{x}_k + \mathbf{B}\dddot{\mathbf{c}}^x_k\right) && \text{(B.14)}\\
&= \mathbf{m}\mathbf{A}\mathbf{x}_k + \mathbf{m}\mathbf{B}\dddot{\mathbf{c}}^x_k && \text{(B.15)}\\
&= \overline{\mathbf{A}}_1\mathbf{x}_k + \overline{\mathbf{B}}_0\dddot{\mathbf{c}}^x_k, && \text{(B.16)}\\
\mathbf{p}^x_{k+2} &= \overline{\mathbf{A}}_1\mathbf{x}_{k+1} + \overline{\mathbf{B}}_0\dddot{\mathbf{c}}^x_{k+1} && \text{(B.17)}\\
&= \overline{\mathbf{A}}_1\left(\mathbf{A}\mathbf{x}_k + \mathbf{B}\dddot{\mathbf{c}}^x_k\right) + \overline{\mathbf{B}}_0\dddot{\mathbf{c}}^x_{k+1} && \text{(B.18)}\\
&= \overline{\mathbf{A}}_2\mathbf{x}_k + \overline{\mathbf{B}}_1\dddot{\mathbf{c}}^x_k + \overline{\mathbf{B}}_0\dddot{\mathbf{c}}^x_{k+1}, && \text{(B.19)}\\
\mathbf{p}^x_{k+i} &= \overline{\mathbf{A}}_i\mathbf{x}_k + \overline{\mathbf{B}}_{i-1}\dddot{\mathbf{c}}^x_k + \overline{\mathbf{B}}_{i-2}\dddot{\mathbf{c}}^x_{k+1} + \cdots + \overline{\mathbf{B}}_0\dddot{\mathbf{c}}^x_{k+i-1}. && \text{(B.20)}
\end{aligned}
$$

Die Matrizen $\overline{\mathbf{A}}_j$ und $\overline{\mathbf{B}}_j$ lassen sich folgendermaßen bestimmen:

$$
\begin{aligned}
\overline{\mathbf{A}}_j = \mathbf{m}\mathbf{A}^j &= \left(1, j\Delta t, \frac{(j\Delta t)^2}{2} - \frac{\mathbf{c}^z}{g}\right), && \text{(B.21)}\\
\overline{\mathbf{B}}_j = \overline{\mathbf{A}}_j\mathbf{B} &= \frac{1}{6}\Delta t^3 + \frac{j}{2}\Delta t^3 + \frac{j^2}{2}\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t && \text{(B.22)}\\
&= \frac{3j^2+3j+1}{6}\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t && \text{(B.23)}\\
&= \left(\frac{j\,(j+1)}{2} + \frac{1}{6}\right)\Delta t^3 - \frac{\mathbf{c}^z}{g}\Delta t. && \text{(B.24)}
\end{aligned}
$$

Daraus lassen sich die Matrizen $\mathbf{A_Z}$ und $\mathbf{B_Z}$ unter Beachtung der Indizes mit Hilfe von Gleichung B.20 erstellen. Die Dimension ist hier $N \times 3$ bzw. $N \times N$.

ANHANG C

Publikationen

Verschiedene Teile dieser Arbeit sind in den folgenden Publikationen veröffentlicht worden (in chronologischer Reihenfolge):

- Czarnetzki, Stefan, Sören Kerner und Oliver Urbann: *Observer-based dynamic walking control for biped robots*. Robotics and Autonomous Systems, 57(8):839–845, 2009[1]
- Czarnetzki, Stefan, Sören Kerner und Oliver Urbann: *Applying dynamic walking control for biped robots*. In: *RoboCup 2009: Robot Soccer World Cup XIII*, Seiten 69–80. Springer Berlin Heidelberg, 2010[1]
- Urbann, Oliver, Sören Kerner und Stefan Tasse: *Rigid and Soft Body Simulation Featuring Realistic Walk Behaviour*. In: *RoboCup 2011: Robot Soccer World Cup XV*, Band 7416 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 126–136. Springer Berlin Heidelberg, 2012
- Urbann, Oliver und Stefan Tasse: *Observer based biped walking control, a sensor fusion approach*. Autonomous Robots, 35(1):37–49, 2013
- Urbann, Oliver und Matthias Hofmann: *Modification of Foot Placement for Balancing Using a Preview Controller Based Humanoid Walking Algorithm*. In: *RoboCup 2013: Robot World Cup XVII*, Band 8371 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 420–431. Springer Berlin Heidelberg, 2014
- Urbann, Oliver und Matthias Hofmann: *A Reactive Stepping Algorithm Based on Preview Controller with Observer for Biped Robots*. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 2016, akzeptiert

[1] Autoren in alphabetischer Reihenfolge.

Abbildungsverzeichnis

Literaturverzeichnis

[1] Alcaraz-Jiménez, JuanJosé, Marcell Missura, Humberto Martínez-Barberá und Sven Behnke: *Lateral Disturbance Rejection for the Nao Robot*. In: Chen, Xiaoping, Peter Stone, LuisEnrique Sucar und Tijn van der Zant (Herausgeber): *RoboCup 2012: Robot Soccer World Cup XVI*, Band 7500 der Reihe *Lecture Notes in Computer Science*, Seiten 1–12. Springer Berlin Heidelberg, 2013.

[2] Anderson, Stuart O und Jessica K Hodgins: *Adaptive torque-based control of a humanoid robot on an unstable platform*. In: *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, Seiten 511–517. IEEE, 2010.

[3] Azevedo, Christine, Philippe Poignet und Bernard Espiau: *Moving Horizon Control for Biped Robots without Reference Trajectory.* In: *ICRA*, Seiten 2762–2767. IEEE, 2002.

[4] Bernstein, N. A.: *Investigations in Biodynamics of Locomotion*, Band 1. WIEM, Moscow, 1935.

[5] Buschmann, T., S. Lohmeier, M. Bachmayer, H. Ulbrich und F. Pfeiffer: *A collocation method for real-time walking pattern generation*. In: *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, Seiten 1–6, Nov 2007.

[6] Camacho, Eduardo F, Carlos Bordons, Eduardo F Camacho und Carlos Bordons: *Model predictive control*, Band 2. Springer London, 2004.

[7] Cho, Jae Uk, Je Sung Yeon und Jong Hyeon Park: *Stable running velocity change of biped robot based on virtual torque*. In: *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, Seiten 301–306, 2012.

[8] Czarnetzki, Stefan, Sören Kerner und Oliver Urbann: *Observer-based dynamic walking control for biped robots*. Robotics and Autonomous Systems, 57(8):839–845, 2009.

[9] Czarnetzki, Stefan, Sören Kerner und Oliver Urbann: *Applying dynamic walking control for biped robots*. In: *RoboCup 2009: Robot Soccer World Cup XIII*, Seiten 69–80. Springer Berlin Heidelberg, 2010.

[10] Dau, Huan, Chee-Meng Chew und Aun-Neow Poo: *Proposal of Augmented Linear Inverted Pendulum model for bipedal gait planning*. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Seiten 172–177. IEEE, 2010.

[11] Diedam, Holger, Dimitar Dimitrov, Pierre-Brice Wieber, Katja Mombaur und Moritz Diehl: *Online walking gait generation with adaptive foot positioning through Linear Model Predictive control*, Band 24, Seiten 1121–1126. IEEE, 2008.

[12] Dimitrov, Dimitar, Antonio Paolillo und Pierre-Brice Wieber: *Walking motion generation with online foot position adaptation based on l1- and l∞-norm penalty formulations*. In: *IEEE International Conference on Robotics & Automation*, Shanghai, China, 2011.

[13] Englsberger, J., C. Ott, M.A. Roa, A. Albu-Schaffer und G. Hirzinger: *Bipedal walking control based on Capture Point dynamics*. In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Seiten 4420 –4427, Sept. 2011.

[14] Ferreau, H.J., H.G. Bock und M. Diehl: *An online active set strategy to overcome the limitations of explicit MPC*. International Journal of Robust and Nonlinear Control, 18(8):816–830, 2008.

[15] Goswami, Ambarish: *Foot Rotation Indicator (FRI) point: A New Gait Planning Tool to Evaluate Postural Stability of Biped Robots*. In: *IEEE International Conference on Robotics and Automation*, Seiten 47–52, 1999.

[16] Gouaillier, D., C. Collette und C. Kilner: *Omni-directional closed-loop walk for Nao*. In: *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, Seiten 448–454, 2010.

[17] Harada, Kensuke, Shuuji Kajita, Kenji Kaneko und Hirohisa Hirukawa: *An analytical method for real-time gait planning for humanoid robots*. International Journal of Humanoid Robotics, 3(01):1–19, 2006.

[18] Hebbel, Matthias, Ralf Kosse und Walter Nisticò: *Modeling and Learning Walking Gaits of Biped Robots*. In: *Proceedings of the Workshop on Humanoid Soccer Robots of the IEEE-RAS International Conference on Humanoid Robots*, Seiten 40–48, December 2006.

[19] Hemami, H und CL Golliday: *The inverted pendulum and biped stability*. Mathematical Biosciences, 34(1):95–110, 1977.

[20] Herdt, Andrei, Holger Diedam, Pierre-Brice Wieber, Dimitar Dimitrov, Katja Mombaur und Moritz Diehl: *Online Walking Motion Generation with Automatic Footstep Placement*. Advanced Robotics, 24(5-6):719–737, 2010.

[21] Hof, AL, MGJ Gazendam und WE Sinke: *The condition for dynamic stability*. Journal of biomechanics, 38(1):1–8, 2005.

[22] Kajita, S., F. Kanehiro, K. Kaneko, K. Yokoi und H. Hirukawa: *The 3D linear inverted pendulum mode: a simple modeling for a biped walking pattern generation*. In: *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, Band 1, Seiten 239 –246, 2001.

[23] Kajita, S., Tomio Yamaura und A. Kobayashi: *Dynamic walking control of a biped robot along a potential energy conserving orbit*. Robotics and Automation, IEEE Transactions on, 8(4):431–438, 1992.

[24] Kajita, Shuuji, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi und Hirohisa Hirukawa: *Biped walking pattern generation by using preview control of zero-moment point*. In: *ICRA*, Seiten 1620–1626. IEEE, 2003.

[25] Kajita, Shuuji, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kazuhito Yokoi und Hirohisa Hirukawa: *Biped Walking Pattern Generator allowing Auxiliary ZMP Control*. In: *IROS*, Seiten 2993–2999. IEEE, 2006.

[26] Katayama, Tohru, Takahira Ohki, Toshio Inoue und Tomoyuki Kato: *Design of an optimal controller for a discrete-time system subject to previewable demand*. International Journal of Control, 41(3):677 – 699, 1985.

[27] Kim, Jung-Hoon: *Walking pattern generation of a biped walking robot using convolution sum*. In: *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, Seiten 539 –544, Dez. 2007.

[28] Kim, Jung-Yup, Ill-Woo Park und Jun-Ho Oh: *Experimental Realization of Dynamic Walking of Biped Humanoid Robot KHR-2 using ZMP Feedback and Inertial Measurement*. Advanced Robotics, 20(6):707 – 736, Juni 2006.

[29] Komura, Taku, Akinori Nagano, Howard Leung und Yoshihisa Shinagawa: *Simulating pathological gait using the enhanced linear inverted pendulum model*. Biomedical Engineering, IEEE Transactions on, 52(9):1502–1513, 2005.

[30] Kudoh, S. und T. Komura: *C2 continuous gait-pattern generation for biped robots*. In: *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, Band 2, Seiten 1135 – 1140, Okt. 2003.

[31] Kuo, Arthur D und Felix E Zajac: *Human standing posture: multi-joint movement strategies based on biomechanical constraints.* Progress in brain research, 97:349–358, 1992.

[32] Lee, Sung-Hee und A. Goswami: *Reaction Mass Pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots*. In: *Robotics and Automation, 2007 IEEE International Conference on*, Seiten 4667 –4672, April 2007.

[33] Meriçli, Çetin und Manuela Veloso: *Biped walk learning through playback and corrective demonstration*. In: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, Seiten 1594–1599, 2010.

[34] Morisawa, M., K. Harada, S. Kajita, K. Kaneko, J. Sola, E. Yoshida, N. Mansard, K. Yokoi und J.-P. Laumond: *Reactive stepping to prevent falling for humanoids*. In: *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, Seiten 528 –534, Dez. 2009.

[35] Morisawa, M., F. Kanehiro, K. Kaneko, N. Mansard, J. Sola, E. Yoshida, K. Yokoi und J. Laumond: *Combining suppression of the disturbance and reactive stepping for recovering balance*. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Seiten 3150 –3156, Okt. 2010.

[36] Morisawa, Mitsuharu, Kensuke Harada, Shuuji Kajita, Shinichiro Nakaoka, Kiyoshi Fujiwara, Fumio Kanehiro, Kenji Kaneko und Hirohisa Hirukawa: *Experimentation of Humanoid Walking Allowing Immediate Modification of Foot Place Based on Analytical Solution*. Proceedings 2007 IEEE International Conference on Robotics and Automation, (April):3989–3994, April 2007.

[37] Nagasaka, K., H. Inoue und M. Inaba: *Dynamic walking pattern generation for a humanoid robot based on optimal gradient method*. In: *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, Band 6, Seiten 908–913, 1999.

[38] Ott, Christian, C Baumgartner, Johannes Mayr, Matthias Fuchs, Robert Burger, Dongheui Lee, Oliver Eiberger, A Albu-Schaffer, Markus Grebenstein und Gerd Hirzinger: *Development of a biped robot with torque controlled joints*. In: *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, Seiten 167–173. IEEE, 2010.

[39] Parietti, Federico und Hartmut Geyer: *Reactive balance control in walking based on a bipedal linear inverted pendulum model*. In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, Seiten 5442–5447. IEEE, 2011.

[40] Park, Ill-Woo und Jung-Yup Kim: *Fourier series-based walking pattern generation for a biped humanoid robot*. In: *Humanoids*, Seiten 461–467, 2010.

[41] Park, Jong H und Kyong D Kim: *Biped robot walking using gravity-compensated inverted pendulum mode and computed torque control*. In: *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, Band 4, Seiten 3528–3533. IEEE, 1998.

[42] Pratt, Jerry E., John Carff, Sergey V. Drakunov und Ambarish Goswami: *Capture Point: A Step toward Humanoid Push Recovery*. In: *Humanoids*, Seiten 200–207. IEEE, 2006.

[43] Siciliano, Bruno und Oussama Khatib (Herausgeber): *Handbook of Robotics*. Springer, Berlin, Heidelberg, 2008.

[44] Stephens, Benjamin J. und Christopher G. Atkeson: *Modeling and control of periodic humanoid balance using the Linear Biped Model.* In: *Humanoids*, Seiten 379–384. IEEE, 2009.

[45] Stephens, B.J. und C.G. Atkeson: *Push Recovery by stepping for humanoid robots with force controlled joints*. In: *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, Seiten 52 –59, Dez. 2010.

[46] Urbann, Oliver und Matthias Hofmann: *Modification of Foot Placement for Balancing Using a Preview Controller Based Humanoid Walking Algorithm*. In: *RoboCup 2013: Robot World Cup XVII*, Band 8371 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 420–431. Springer Berlin Heidelberg, 2014.

[47] Urbann, Oliver und Matthias Hofmann: *A Reactive Stepping Algorithm Based on Preview Controller with Observer for Biped Robots*. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 2016, akzeptiert.

[48] Urbann, Oliver, Sören Kerner und Stefan Tasse: *Rigid and Soft Body Simulation Featuring Realistic Walk Behaviour*. In: *RoboCup 2011: Robot Soccer World Cup XV*, Band 7416 der Reihe *Lecture Notes in Artificial Intelligence*, Seiten 126–136. Springer Berlin Heidelberg, 2012.

[49] Urbann, Oliver, Ingmar Schwarz und Matthias Hofmann: *Flexible Linear Inverted Pendulum Model for Cost-Effective Biped Robots*. In: *Humanoid Robots (Humanoids), 2015 15th IEEE-RAS International Conference on*, Seiten 128–131, Nov 2015.

[50] Urbann, Oliver und Stefan Tasse: *Observer based biped walking control, a sensor fusion approach*. Autonomous Robots, 35(1):37–49, 2013.

[51] Vukobratovic, M.: *How to Control Artificial Anthropomorphic Systems*. Systems, Man and Cybernetics, IEEE Transactions on, SMC-3(5):497–507, 1973.

[52] Vukobratović, M und B Borovac: *Zero-moment Point – Thirty Five Years of its life*. International Journal of Humanoid Robotics, 1(1):157–173, 2004.

[53] Vukobratovic, M., A.A. Frank und D. Juricic: *On the Stability of Biped Locomotion*. Biomedical Engineering, IEEE Transactions on, BME-17(1):25–36, 1970.

[54] Vukobratovic, Miomir und Davor Juricic: *Contribution to the Synthesis of Biped Gait*. Biomedical Engineering, IEEE Transactions on, BME-16(1):1–6, Jan 1969.

[55] Wight, Derek L, Eric G Kubica und David WL Wang: *Introduction of the foot placement estimator: A dynamic measure of balance for bipedal robotics*. Journal of Computational and Nonlinear Dynamics, 3(1), 2008.

[56] Yeon, Je Sung, Ohung Kwon und Jong Hyeon Park: *Trajectory generation and dynamic control of planar biped robots with curved soles*. Journal of mechanical science and technology, 20(5):602–611, 2006.

[57] Yun, Seung-Kook und A. Goswami: *Momentum-based reactive stepping controller on level and non-level ground for humanoid robot push recovery*. In: *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Seiten 3943 –3950, Sept. 2011.

[58] Zutven, P. van, D. Kostic und H. Nijmeijer: *Foot placement for planar bipeds with point feet*. In: *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, Seiten 983 –988, Mai 2012.