

Vierbeiniges Laufen

Modellierung und Optimierung von Roboterbewegungen

Diplomarbeit

Uwe Düffert

22. Juli 2004



Lehr- und Forschungsgebiet Künstliche Intelligenz
Institut für Informatik
Humboldt-Universität zu Berlin

Gutachter: Prof. Dr. sc. Hans-Dieter Burkhard, Dipl.-Phys. Jan Hoffmann

Inhaltsverzeichnis

| | | |
|----------|---|-----------|
| 1 | Einleitung | 1 |
| 1.1 | RoboCup | 1 |
| 1.2 | Roboter und Spielfeld | 1 |
| 1.3 | Motivation | 3 |
| 1.4 | Ziel | 3 |
| 1.5 | Integration | 4 |
| 1.6 | Begriffe | 4 |
| 2 | Modellierung | 7 |
| 2.1 | Ziel | 7 |
| 2.1.1 | Laufeigenschaften | 7 |
| 2.1.2 | Linearität der Ansteuerung | 8 |
| 2.2 | Kinematik | 8 |
| 2.2.1 | Vorwärtskinematik | 9 |
| 2.2.2 | Inverse Kinematik | 10 |
| 2.2.3 | Anatomiebedingte Korrektur | 12 |
| 2.3 | Ein einfaches Laufmodell | 13 |
| 2.3.1 | Stehen | 14 |
| 2.3.2 | Schwerpunktbewegung | 14 |
| 2.3.3 | Schrittberechnung | 15 |
| 2.3.4 | Statisch vs. dynamisch stabiles Laufen | 15 |
| 2.3.5 | Parameter | 16 |
| 2.4 | Erweitertes Laufmodell | 17 |
| 2.4.1 | Verbesserungen | 17 |
| 2.4.2 | Rädermodell | 18 |
| 2.4.3 | Parameter | 20 |
| 2.4.4 | Parametersatzauswahl und -interpolation | 22 |
| 2.4.5 | Parameterbestimmung | 24 |
| 3 | Optimierung | 25 |
| 3.1 | Optimierungsmethoden | 25 |
| 3.1.1 | Berechenbarkeit eines optimalen Laufmusters | 25 |
| 3.1.2 | Von der Natur inspirierte Laufmuster | 25 |
| 3.1.3 | Vorgehensweise | 26 |
| 3.2 | Lokalisierung | 27 |
| 3.2.1 | Orientierungshilfe | 28 |
| 3.2.2 | Positionsberechnung | 30 |
| 3.2.3 | Glättung | 31 |

| | | |
|----------|---|-----------|
| 3.2.4 | Positionsgenauigkeit | 32 |
| 3.3 | Omnidirektionale Optimierung | 33 |
| 3.3.1 | Der Parcours | 33 |
| 3.3.2 | Fitnessfunktion und Gütemaß | 37 |
| 3.3.3 | Optimierungsvoraussetzungen | 38 |
| 3.3.4 | Lernmethode | 41 |
| 3.3.5 | Kreuzung, Mutation und Population | 42 |
| 3.3.6 | Ergebnisse der Evolution | 43 |
| 3.4 | Vermessung verschiedener Parametersätze | 45 |
| 3.4.1 | Geschwindigkeitsmessung | 46 |
| 3.4.2 | Drehvermessung | 47 |
| 3.4.3 | Parametersatzvermessung | 47 |
| 3.5 | Kalibrierung | 52 |
| 3.5.1 | Kalibrierungsergebnis | 54 |
| 3.6 | Optimierungsergebnis | 56 |
| 4 | Zusammenfassung und Ausblick | 59 |
| A | Literaturverzeichnis | 63 |
| B | Danksagung | 67 |

1 Einleitung

Im Rahmen dieser Arbeit wird einem handelsüblichen vierbeinigen Roboter das Laufen beigebracht. Für den Einsatz autonomer Roboter in menschlicher Umgebung gilt die Fortbewegung dieser Roboter mit Beinen als Grundvoraussetzung für ihre Akzeptanz und Nützlichkeit. Um laufen zu können, müssen solche Roboter auf der einen Seite geeignet konstruiert sein, auf der anderen Seite folgt aus gegebener Konstruktion noch lange nicht, wie damit ein möglichst gutes Laufen aussieht. Deshalb wird hier an einem konkreten Beispiel der Weg zu einem leistungsfähigen Laufen gezeigt.

1.1 RoboCup

Der RoboCup ist eine internationale wissenschaftliche Initiative mit dem Ziel, die Entwicklung von Robotik und Künstlicher Intelligenz zu fördern (siehe [20, 34, 2]). Seit 1997 findet dazu einmal im Jahr eine RoboCup-Weltmeisterschaft statt, um die aktuellen Forschungsergebnisse in Roboter-Fußballspielen zu evaluieren. Dabei treten Mannschaften aus Robotern oder Softwareagenten¹ an, um autonom gegeneinander zu spielen. Um verschiedene Forschungsschwerpunkte zu ermöglichen und trotzdem Fairness zu garantieren, gibt es dabei unterschiedliche Ligen. Darunter befindet sich, von einigen zweibeinigen Einzeldemonstrationen abgesehen, derzeit nur eine Liga für Roboter mit Beinen. In dieser Liga werden vierbeinige Roboter von Sony verwendet, deren Hardware von den einzelnen Mannschaften nicht modifiziert werden darf.

Die Humboldt-Universität zu Berlin, vertreten durch den Lehrstuhl für Künstliche Intelligenz des Institutes für Informatik, nimmt seit 1999 als Mannschaft (AiboTeamHumboldt, früher Humboldt Heroes, siehe [1]) an der Sony-Liga des RoboCup teil. Seit 2001 tritt das AiboTeamHumboldt dabei international als Teil des von ihm initiierten GermanTeam² [10, 31] auf. Das GermanTeam wurde 2004 RoboCup-Weltmeister in seiner Liga.

1.2 Roboter und Spielfeld

In der Sony-Liga wurden bis zum Ende des Jahres 2000 Roboter des Modells Sony Aibo ERS-110 (Abbildung 1.1 a) verwendet und in den darauf folgenden drei Jahren das Modell Aibo ERS-210(A) (siehe Abbildung 1.1 b). Seit 2004 ist zusätzlich das Modell Aibo ERS-7 (Abbildung 1.1 c) zugelassen. Alle diese Modelle haben vier Beine mit je drei Motoren und einen beweglichen Kopf mit Farbkamera. Zahlreiche

¹Computerprogramme, die autonom eine ihnen zugewiesene Aufgabe erfüllen

²deutsche Nationalmannschaft der Sony-Liga des RoboCup bestehend aus Humboldt-Universität zu Berlin, Universität Bremen, Technische Universität Darmstadt und Universität Dortmund

1 Einleitung



Abbildung 1.1: Die verwendeten Roboter, zum Teil mit Trikot, in jeweils typischer Haltung: **a)** ERS-110, **b)** ERS-210, **c)** ERS-7

weitere Sensoren sowie alle für autonomes Handeln benötigten Komponenten (Prozessor, Haupt- und Wechselspeicher, Akku) sind bereits integriert. Die Hardware-Unterschiede zwischen den einzelnen Modellen führen zwar zu verschiedenen Ergebnissen, aber Modellierung und Herangehensweise können gemeinsam benutzt werden.

In dieser Arbeit wird für die Grundlagen und erste Ergebnisse zunächst hauptsächlich ein ERS-210 verwendet und danach das Erreichte auf einen ERS-7 übertragen und angepasst, um schließlich für beide aktuell verwendeten Modelle gute Ergebnisse zu erhalten.

Das in der Sony-Liga verwendete Spielfeld (siehe Abbildung 1.2) wird von Jahr zu Jahr modifiziert. Dabei sollen zum einen die Spiele interessant bleiben, zum anderen aber realitätsferne Vereinfachungen abgebaut werden, zum Beispiel durch Verdoppelung der Spielfeldgröße oder Weglassen von einzelnen Landmarken (farbkodierten Orientierungspunkten) oder einer äußeren Bande.

Eines der wenigen unverändert gebliebenen und speziell für das Laufen besonders interessanten Merkmale ist das Material des Untergrundes, ein grüner Teppich. Da das Laufen der verwendeten Roboter deutlich vom Untergrund abhängt, wird auch in allen dieser Arbeit zugrunde liegenden Experimenten ein solcher Teppich verwendet. Die Benutzung eines anderen Untergrundes würde bei gleicher Vorgehensweise zu abweichenden Ergebnissen führen und wird in dieser Arbeit nicht weiter untersucht.



Abbildung 1.2: Das noch aktuelle Spielfeld der Liga für vierbeinige Sony-Roboter, hier bei der RoboCup-Weltmeisterschaft 2002 in Fukuoka

1.3 Motivation

Bevor man sich über andere Aspekte (Wahrnehmung, künstliche Intelligenz) Gedanken macht, muss geklärt werden, wie sich ein solcher Roboter fortbewegen kann, wie also mit vier Beinen à drei Motoren eine Laufbewegung zu realisieren ist. Ohne dieses Wissen kann man keine Verhaltenssteuerung implementieren, da dann noch nicht bekannt ist, über welche Aktionsmöglichkeiten der Roboter verfügt. Außerdem ist es schwer, dem Roboter zu einer robusten Wahrnehmung seiner Umgebung zu verhelfen, ohne etwas über mögliche Geschwindigkeiten oder Erschütterungen der Roboterbewegung zu wissen.

Daher ist hier zum Beispiel zu untersuchen, wie der Roboter die Balance halten und Umfallen vermeiden kann, welches das geeignetste Laufmuster ist, wie man möglichst flexibel auf sich ändernde Laufrichtungs-, Laufgeschwindigkeits- und Drehgeschwindigkeitswünsche reagieren kann und welche Optimierungsverfahren erfolgreich einsetzbar sind.

Da neben anderen Forschern auch alle an RoboCup-Ligen mit mehrbeinigen Robotern teilnehmenden Teams vor diesem Problem stehen, gibt es dazu schon einige Arbeiten, etwa zu omnidirektionalem Laufen (beliebige Mischungen aus Vorwärtslaufen, Seitwärtslaufen und Drehen) [14], schnellem Geradeauslaufen [5] oder automatisierter Optimierung [21]. Möglichst viele dieser zum Teil gegenläufigen Konzepte sollen in dieser Arbeit vereint werden.

1.4 Ziel

Ziel ist es hier, unter weitgehend bekannten Rahmenbedingungen wie Roboterabmessungen und Untergrundbeschaffenheit das Laufen gegebener vierbeiniger Roboter zu optimieren. Dabei kommt es weniger darauf an zu verstehen, warum ein bestimmtes Laufen besser als ein anderes ist. Vielmehr geht es darum, den Weg aufzuzeigen, wie man zu einem solchen Laufen kommt, welche Erfahrungen dabei in den letzten Jahren gemacht worden sind und welche Berechnungs- und Optimierungsverfahren zum Einsatz kommen. Im Vordergrund steht also die praktische Umsetzung, ent-

sprechend stammen alle Daten und Messwerte in den folgenden Kapiteln von realen Robotern.

Auf Simulationen der verwendeten Roboter und ihrer Bewegungen wird bewusst komplett verzichtet, weil derzeit keine ausreichend genaue Simulation existiert. Obwohl viel Zeit in die Erstellung geeigneter Software für die Roboter investiert wurde, enthält diese Arbeit keine Quelltextauszüge, da dies nicht dem Verständnis der Problematik dienen würde. Der komplette Quelltext befindet sich unter <http://www.uwe-dueffert.de/code.shtml>.

1.5 Integration

Die im Rahmen dieser Arbeit entstandene Software ist während der gesamten Entwicklungszeit in die Software des GermanTeam [4, 7, 33] eingebettet und verwendet auch deren Architektur. Diese Architektur [31] unterteilt das Fußballspielen (oder jede andere komplexere Aufgabe) in separate Teilaufgaben, genannt Module.

Für die Optimierung des Laufens sind neben dem Modul zur Generierung von Laufmustern (*WalkingEngine*) auch die Module Bildverarbeitung (*ImageProcessor*), Selbstlokalisierung (*SelfLocator*), Verhaltenssteuerung (*BehaviorControl*) und Kopfsteuerung (*HeadControl*) relevant und es wird an entsprechender Stelle auf sie eingegangen werden.

Durch die Integration ist sichergestellt, dass die verwendeten Verfahren praxistauglich und wiederverwendbar sind und mit aktuellen Entwicklungen (zum Beispiel neuen Robotermodellen in der Sony-Liga) Schritt gehalten wird. Der durch die Integration entstehende, zum Teil erhebliche Mehraufwand wurde aus diesem Grund bewusst in Kauf genommen.

1.6 Begriffe

Stabilität

Ziel ist in dieser Arbeit immer die Erstellung eines schnellen, stabilen Laufens. Stabil ist das Laufen, wenn der Roboter dabei weder umkippt noch ins Straucheln gerät, sondern stattdessen reproduzierbar weitgehend das passiert, was durch die Ansteuerung beabsichtigt wurde.

Es werden zwei Arten von Stabilität unterschieden. Statische Stabilität bedeutet, dass man die Bewegung zu jedem Zeitpunkt anhalten kann, ohne dass der Roboter dann kippt, weil das Lot seines Schwerpunktes stets in dem von den Bodenberührungspunkten gebildeten Polygon liegt (nach [27]). Dynamische Stabilität hingegen fordert, dass der Roboter während der ununterbrochenen Bewegung nicht umkippt.

Ansteuerung

Das Modul zur Verhaltenssteuerung generiert eine Folge von Bewegungswünschen, aus der die Ansteuerung der Beinmotoren resultiert. Diese Ansteuerung schwankt in der Praxis, zum Beispiel beim Fußballspielen, relativ stark, um auf die sich rasch ändernde Umwelt reagieren zu können. Es genügt daher nicht, ausschließlich konstante

Spezialfälle wie schnelles Vorwärtslaufen zu optimieren, ohne dabei Rücksicht auf andere durch eine Verhaltenssteuerung angeforderte Bewegungen und die dorthin führenden Übergänge zu nehmen. Es gibt deutliche Unterschiede bei der erreichbaren Geschwindigkeit zwischen konstanter Bewegungsanforderung und ständiger Korrektur von Drehgeschwindigkeit und Laufrichtung, um den Roboter an einem bestimmten Punkt ankommen oder auf einer vorgegebenen Wegstrecke verbleiben zu lassen.

Omnidirektionalität

Omnidirektionalität ist die Fähigkeit, in jede Richtung laufen zu können, also Vorwärts-, Seitwärts- und Drehgeschwindigkeit in beliebigem Verhältnis mischen zu können. Da die verwendeten Roboter dazu prinzipiell in der Lage sind, ist es wünschenswert, diese Fähigkeit in der Ansteuerung durch ein übergeordnetes Verhalten ausnutzen zu können.

Odometrie

Zu einer gegebenen Ansteuerung sollte der Roboter auch blind eine Vorstellung³ davon haben, wie er sich dadurch tatsächlich bewegen wird. Diese Fähigkeit, nur aufgrund der eigenen Gelenk- beziehungsweise Fußbewegungen auf die zurückgelegte Strecke schließen zu können, wird als Odometrie bezeichnet. Idealerweise sind per Ansteuerung gewünschte, tatsächlich erreichte und durch Odometrie berechnete Lageveränderung identisch. Um diesem Ideal nahe zu kommen, sind in der Regel Korrekturparameter nötig, die die auftretenden Unterschiede zwischen berechneten (erwarteten) und tatsächlich gemessenen Geschwindigkeiten minimieren.

Laufparameter und Kinematik

Eine Laufbewegung hat neben solchen Korrekturparametern auch immer zahlreiche weitere Parameter wie Schrittlänge oder Schulterhöhe, die den Verlauf der Fußpositionen bestimmen. Mittels Kinematik, der Lehre von der geometrischen Beschreibung von Bewegungen ohne Berücksichtigung von Kräften, lassen sich diese Fußpositionen und die zu den Fußpositionen führenden Beingelenkwinkel ineinander umrechnen.

Vorwärtskinematik berechnet dabei Fußpositionen aus Gelenkwinkeln, Inverse Kinematik hingegen Gelenkwinkel aus Fußpositionen. Bevor mit Inverser Kinematik Beingelenkwinkel berechnet werden können, müssen daher erst einmal gute Werte für alle verwendeten Laufparameter gefunden werden, sei es durch Berechnung eines Optimums oder durch gezieltes Ausprobieren in Verbindung mit geeigneten Lernverfahren.

³der Roboter benötigt ein Modell seiner Position im Raum, das auch ohne visuelle Informationen aktualisiert werden kann

PID-Regler

Bevor das Ansteuersignal für einen gewünschten Beingelenkwinkel den dafür jeweils zuständigen Motor erreicht, durchläuft es noch einen PID-Regler. Dieser ist dafür zuständig, aus dem aktuellen Gelenkwinkel (Ist) und dem gewünschten Gelenkwinkel (Soll) eine geeignete Motoransteuerung abzuleiten. Dazu hat er drei Parameter: P (proportional) mit starker Ansteuerung bei hoher Abweichung zwischen Soll und Ist, I (integral) mit starker Ansteuerung bei lang anhaltender Abweichung und D (differential) mit starker Ansteuerung bei Erhöhung der Abweichung.

Laufmuster, Duty-Factor und Vollschrift

Je nachdem, zu welchem Zeitpunkt ein mehrbeiniger Roboter seine Beine beim Laufen hebt und senkt, unterscheidet man verschiedene Laufmuster oder Gangarten wie Trab und Galopp. Laufmuster lassen sich unter anderem dadurch charakterisieren, wie viel Prozent der Zeit ein einzelner Fuß durchschnittlich Bodenkontakt hat, oder anders ausgedrückt: wie viel Prozent der Füße durchschnittlich am Boden sind. Dieser Anteil wird auch Duty-Factor genannt. Beim Trab sind beispielsweise abwechselnd zwei Beine am Boden und die anderen beiden in der Luft, der Duty-Factor beträgt daher 0,5.

Ein Vollschrift ist ein Schritt mit jedem Bein. Folglich wiederholt sich eine konstante Laufbewegung nach jedem Vollschrift, ein Laufmuster ist also durch die Spezifikation eines Vollschriftes vollständig charakterisiert.

Kalibrierung

Als Kalibrierung wird in dieser Arbeit der Vorgang bezeichnet, die tatsächlich gemessene Geschwindigkeit durch Modifikation der Ansteuerung der angeforderten Geschwindigkeit anzunähern. Wird beispielsweise diagonales Laufen angefordert, der Roboter läuft dabei aber mehr geradeaus als zur Seite, so wird die angesteuerte Seitwärtsgeschwindigkeit erhöht. Dann stimmen zwar angesteuerte Vorwärts- und Seitwärtsgeschwindigkeit nicht mehr überein, dafür entspricht die real ausgeführte Bewegung der ursprünglich angeforderten.

Einheiten und Koordinaten

In dieser Arbeit werden alle Längen in Millimetern, Geschwindigkeiten in Millimetern pro Sekunde, Winkel im Bogenmaß und Winkelgeschwindigkeiten in Bogenmaß pro Sekunde angegeben, falls nicht explizit eine andere Einheit verwendet wird.

In dreidimensionalen Koordinatensystemen wird, wenn nicht anders angegeben, ein Rechtssystem benutzt: Die x-Achse zeigt aus Sicht des Koordinatensystems (zum Beispiel aus Sicht eines Roboters) nach vorne, die y-Achse nach links und die z-Achse nach oben.

2 Modellierung

2.1 Ziel

Um Robotern, wie denen in Abbildung 2.1, das Laufen mit Beinen zu ermöglichen, bedarf es eines Modells, das aus einer gewünschten Bewegungsanforderung eine Folge von Gelenkwinkeln für alle beteiligten Gelenke generiert. Ein solches Modell wird zusammen mit seiner Entwicklung in diesem Kapitel vorgestellt.

2.1.1 Laufeigenschaften

In der Domäne RoboCup ist Laufgeschwindigkeit ein entscheidendes Kriterium. Teams, deren Roboter deutlich langsamer als die der schnellsten Mannschaften laufen, haben dadurch einen kaum zu kompensierenden Nachteil. Unter den Teams mit ausreichender Laufgeschwindigkeit werden jedoch andere Eigenschaften des Laufens interessanter und auch spielentscheidend.

Wünschenswert ist hier zum Beispiel ein omnidirektionales und ansteuerungstolerantes Laufen, damit eine Verhaltenssteuerung möglichst schnell und unkompliziert auf die sich rasch verändernde Umgebung reagieren kann, ohne die Stabilität des Laufens zu gefährden. Die bisherigen Versuche zur automatisierten Optimierung der Laufparameter betrachteten hingegen nur die maximale Geschwindigkeit [5, 21] oder zusätzlich nur derzeit nicht so sehr RoboCup-relevante Eigenschaften wie Toleranz gegenüber verschiedenen Untergründen [15].

Ziel ist es also, ein konkurrenzfähiges Laufmodell zu schaffen, das den Anforderungen im RoboCup gerecht wird. Dieses Modell soll zudem die Auswahl und Benutzung verschiedener Parameter sowie eine objektive Bewertung des daraus resultierenden Laufens ermöglichen.

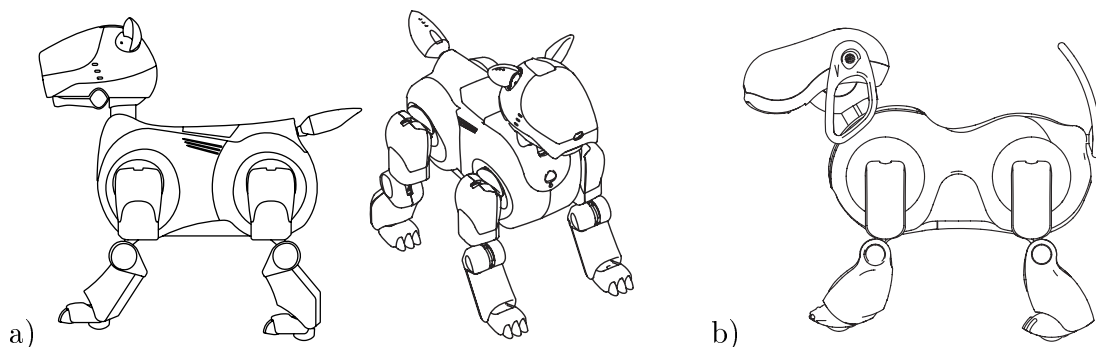


Abbildung 2.1: Schematische Darstellungen eines Sony a) ERS-210 und b) ERS-7

2.1.2 Linearität der Ansteuerung

Es ist wünschenswert, dass sich die real erreichte Laufgeschwindigkeit proportional zur Ansteuerung verhält, und meines Wissens gehen alle bisher in der Sony-Liga verwendeten Laufmodelle von dieser Annahme aus. Leider handelt es sich dabei in der Praxis nur um eine grobe Näherung.

Besonders in den Grenzbereichen gibt es Abweichungen: Statt ganz langsam zu laufen wird sich der Roboter aufgrund von Reibung und Trägheit kaum oder gar nicht von der Stelle bewegen, während er bei voller Ansteuerung wegen der Trägheit seiner Teile nicht ganz die theoretisch erwartete Geschwindigkeit erreicht. Daher führt die Annahme proportional zur Ansteuerung steigender Laufgeschwindigkeit dazu, dass man entweder nur den linear ansteuerbaren Teil benutzen kann oder Abweichungen zwischen erwarteter und realer Geschwindigkeit in Kauf nimmt, die die Odometrie bei fehlenden Sichtinformationen verschlechtern.

Diese Proportionalitätsannahme erleichtert also zwar Modellierung und Kalibrierung, allerdings verbunden mit Performance- und/oder Qualitätseinbußen. Daher wird in Abschnitt 2.4 eine Möglichkeit vorgestellt, diese Annahme zu umgehen und somit lokale Kalibrierung und Optimierungen zu erlauben.

2.2 Kinematik

Alle im RoboCup verwendeten und später in dieser Arbeit vorgestellten Laufmodelle beschreiben die Position der Füße relativ zum Roboterkörper. Nach der Berechnung der gewünschten Fußpositionen müssen die Gelenkwinkel bestimmt werden, die zum Erreichen dieser Fußpositionen führen. Mit dieser Problematik befasst sich die Inverse Kinematik.

Die am beziehungsweise im Bein eines Sony Aibo befindlichen Gelenke ermöglichen es, alle Fußpositionen zu erreichen, die sich in für Laufbewegungen interessanter Lage zur Schulter befinden. Es existiert also mindestens eine Lösung für die zu berechnenden Gelenkwinkel. Zu deren Bestimmung ist im Allgemeinen ein nichtlineares Gleichungssystem zu lösen, was oft nur numerisch möglich ist und mehrere Lösungen liefern kann. Für den mit einem Sony Aibo vorgegebenen Körperbau gibt es hingegen eine analytische Lösung, und unter Verwendung geeigneter und an entsprechender Stelle vorgestellter Annahmen ist diese sogar eindeutig.

Im Abschnitt 2.2 wird stets das linke Vorderbein betrachtet und wenn nicht anders angegeben das Koordinatensystem der zugehörigen Schulter verwendet. Für die anderen Beine sind aufgrund der Symmetrien an geeigneten Stellen andere Vorzeichen und für die Hinterbeine eine andere Oberschenkellänge l_1 notwendig, der Rechenweg bleibt der gleiche. In ihrem Koordinatensystem hat die Schulter die Koordinaten $(0, 0, 0)$, der Fuß die Wunschkoordinaten (x, y, z) , die Unterschenkellänge betrage l_2 und die Gelenkwinkel seien θ_1 , θ_2 und θ_3 (siehe Abbildung 2.2).

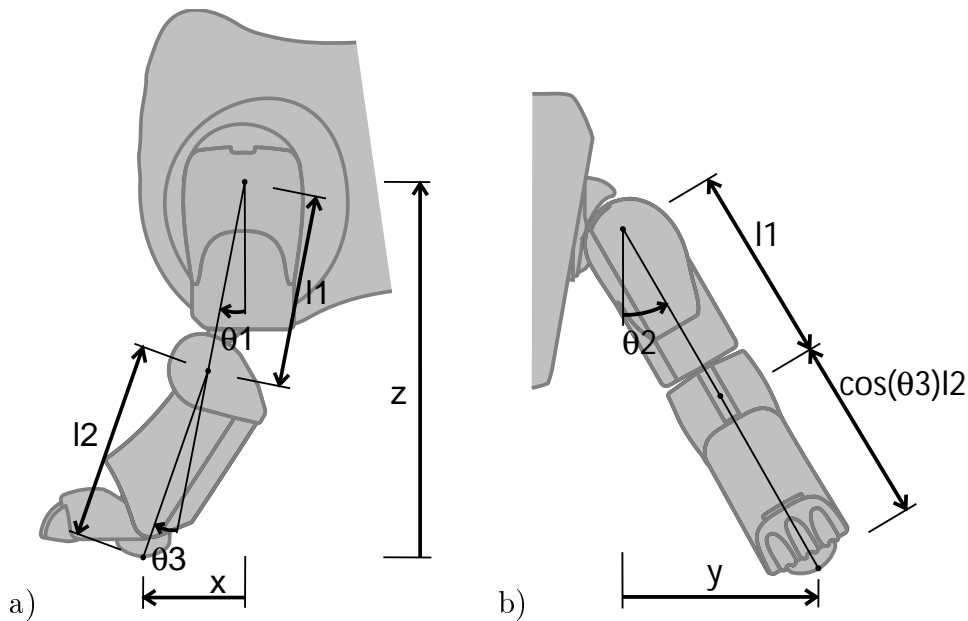


Abbildung 2.2: Linkes Vorderbein mit Längen und Winkeln: **a)** Seitenansicht zur Berechnung von Kniegelenkwinkel θ_3 , **b)** Frontansicht zur Berechnung von Schultergelenkwinkel θ_2 bei gegebenem θ_3

2.2.1 Vorwärtskinematik

Zunächst jedoch betrachten wir die zugrunde liegende Vorwärtskinematik. Dabei wird aus gegebenen Gelenkwinkeln die resultierende Fußposition berechnet. Die Fußposition (x, y, z) relativ zur Schulter ergibt sich durch geeignete Koordinatentransformation des lokalen Koordinatensystems des Fußes in das der Schulter. Aufgrund der gegebenen (vereinfachten) Anatomie des Roboters besteht diese aus fünf Einzeltransformationen:

1. Drehung im Uhrzeigersinn um Schultergelenkwinkel θ_1 um die y -Achse
2. Drehung gegen den Uhrzeigersinn um Schultergelenkwinkel θ_2 um die x -Achse
3. Verschiebung um Oberschenkellänge l_1 in Richtung der negativen z -Achse
4. Drehung im Uhrzeigersinn um Kniegelenkwinkel θ_3 um die y -Achse
5. Verschiebung um Unterschenkellänge l_2 in Richtung der negativen z -Achse

Im Folgenden bezeichne $\text{Rot}_n(\alpha)$ eine Rotation von α entgegen dem Uhrzeigersinn um die n -Achse sowie $\text{Trans}(v)$ eine Translation um den Vektor v . Dann ergibt sich in homogenen Koordinaten, die zur Zusammenfassung dieser fünf Einzeltrans-

formationen hilfreich sind, die Fußposition relativ zur Schulter wie folgt aus den Gelenkwinkeln:

$$\begin{aligned}
 \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} &= \text{Rot}_y(-\theta_1) \cdot \text{Rot}_x(\theta_2) \cdot \text{Trans} \begin{pmatrix} 0 \\ 0 \\ -l_1 \end{pmatrix} \cdot \text{Rot}_y(-\theta_3) \cdot \text{Trans} \begin{pmatrix} 0 \\ 0 \\ -l_2 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \\
 &= \begin{pmatrix} \cos(\theta_1) & 0 & -\sin(\theta_1) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta_1) & 0 & \cos(\theta_1) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_2) & -\sin(\theta_2) & 0 \\ 0 & \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \\
 &\quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta_3) & 0 & -\sin(\theta_3) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta_3) & 0 & \cos(\theta_3) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -l_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \\
 &= \begin{pmatrix} l_2 \cos(\theta_1) \sin(\theta_3) + l_2 \sin(\theta_1) \cos(\theta_2) \cos(\theta_3) + l_1 \sin(\theta_1) \cos(\theta_2) \\ l_1 \sin(\theta_2) + l_2 \sin(\theta_2) \cos(\theta_3) \\ l_2 \sin(\theta_1) \sin(\theta_3) - l_2 \cos(\theta_1) \cos(\theta_2) \cos(\theta_3) - l_1 \cos(\theta_1) \cos(\theta_2) \\ 1 \end{pmatrix} \cdot
 \end{aligned} \tag{2.1}$$

2.2.2 Inverse Kinematik

Die Aufgabe der Inversen Kinematik ist an dieser Stelle die Auflösung der Gleichung 2.1 nach $(\theta_1, \theta_2, \theta_3)$, da man diese Winkel aus gegebenem (x, y, z) berechnen will. Das Wissen um die Anatomie des Roboters führt zu einer geeigneten Reihenfolge bei der Berechnung der gesuchten Winkel.

2.2.2.1 Berechnung von Kniegelenkwinkel θ_3

Die Festlegung von Kniegelenkwinkel θ_3 beschränkt die möglichen Fußpositionen auf eine Kugeloberfläche rund um die Schulter, θ_3 ist somit ausschließlich vom Abstand zwischen Schultergelenk und gewünschter Fußposition abhängig. Ist dieser Abstand nicht größer als die Beinlänge, dann ergibt sich der Kniegelenkwinkel nach dem Kosinussatz.

Der von Oberschenkel l_1 und Unterschenkel l_2 eingeschlossene Winkel (siehe Abbildung 2.2 a) ist der Ergänzungswinkel von θ_3 zu π , also ist

$$\begin{aligned}
 \cos(\pi - \theta_3) &= \frac{l_1^2 + l_2^2 - (x^2 + y^2 + z^2)}{2 l_1 l_2}, \text{ und damit} \\
 \theta_3 &= \pi \pm \arccos \left(\frac{l_1^2 + l_2^2 - (x^2 + y^2 + z^2)}{2 l_1 l_2} \right) \\
 &= \mp \arccos \left(\frac{(x^2 + y^2 + z^2) - l_1^2 - l_2^2}{2 l_1 l_2} \right).
 \end{aligned}$$

Hier liefert die Inverse Kinematik zwei Lösungen, da theoretisch zwei verschiedene Kniepositionen zum Erreichen der gewünschten Fußposition führen. Die Kniegelenke

der verwendeten Roboter lassen sich auch tatsächlich in beide Richtungen beugen, in eine Richtung jedoch erheblich weiter als in die andere. Unter der Annahme, dass die Kniegelenke nur in ihrer Vorzugsrichtung gebeugt werden, bleibt als Lösung:

$$\theta_3 = \arccos\left(\frac{x^2 + y^2 + z^2 - l_1^2 - l_2^2}{2l_1l_2}\right) . \quad (2.2)$$

2.2.2.2 Berechnung von Schultergelenkwinkel θ_2

Mit bekanntem Kniegelenkwinkel θ_3 lässt sich nun der Schulterwinkel θ_2 berechnen, da dieser dann nur noch vom gegebenen seitlichen Wunschabstand y des Fußes abhängt (siehe Abbildung 2.2b). Die Festlegung von θ_2 beschränkt die noch möglichen Fußpositionen auf einen Kreis um die Schulterachse im Abstand y neben dem Roboter. Es ist

$$y = \sin(\theta_2) \cdot (l_1 + l_2 \cdot \cos(\theta_3)) \quad , \text{ also}$$

$$\theta_2 = \arcsin\left(\frac{y}{l_1 + l_2 \cos(\theta_3)}\right) \quad , \text{ da anatomiebedingt } |\theta_2| \leq \frac{\pi}{2}. \quad (2.3)$$

2.2.2.3 Berechnung von Schultergelenkwinkel θ_1

Sind der Schultergelenkwinkel θ_2 nahe $\pi/2$ und der Kniegelenkwinkel θ_3 nahe 0, das Bein also seitlich ausgestreckt, so ist θ_1 irrelevant und sei 0, da die Drehung eines seitlich ausgestreckten Beines um die eigene Achse die Fußposition nicht verändert.

Andernfalls müssen die Gleichungen aus 2.1 nach θ_1 aufgelöst werden. Da θ_2 und θ_3 schon berechnet sind, lassen sich diese Gleichungen durch Einführung der Hilfsvariablen a , b , d , β (siehe Abbildung 2.3) entscheidend vereinfachen. Seien also:¹

$$a = l_2 \sin(\theta_3) \quad (2.4)$$

$$b = (l_1 + l_2 \cos(\theta_3)) \cos(\theta_2) \quad (2.5)$$

$$d = \sqrt{a^2 + b^2} \quad \text{und}$$

$$\beta = \arctan(b, a) \quad .$$

Mit d und β lassen sich a und b aber auch anders darstellen:

$$a = d \cos(\beta) \quad \text{und} \quad (2.6)$$

$$b = d \sin(\beta) \quad . \quad (2.7)$$

Mit diesen Hilfsvariablen ergibt sich:

$$\begin{aligned} x &= l_2 \cos(\theta_1) \sin(\theta_3) + l_2 \sin(\theta_1) \cos(\theta_2) \cos(\theta_3) + l_1 \sin(\theta_1) \cos(\theta_2) && \text{nach 2.1} \\ &= a \cos(\theta_1) + b \sin(\theta_1) && \text{nach 2.4 und 2.5} \\ &= d \cos(\theta_1) \cos(\beta) + d \sin(\theta_1) \sin(\beta) && \text{nach 2.6 und 2.7} \\ &= d \cos(\theta_1 + \beta) && \text{Additionstheorem} \\ z &= d \sin(\theta_1 + \beta) \quad . && \text{analog} \end{aligned}$$

¹In dieser Arbeit wird für arctan stets die Parameterreihenfolge $\arctan(x, y)$ (Mathematica-Syntax) und nicht die Reihenfolge $\arctan(y, x)$ (C-Syntax) verwendet.

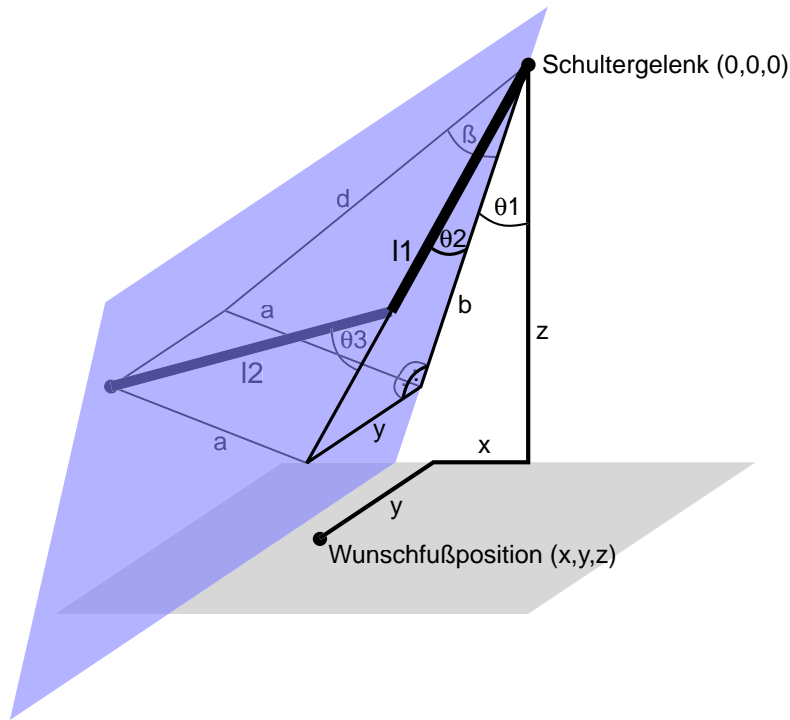


Abbildung 2.3: Modell eines Roboterbeins zur Berechnung von Schultergelenkwinkel θ_1 mit allen verwendeten Hilfsvariablen

Daraus erhält man schließlich:

$$\begin{aligned} \theta_1 + \beta &= \arctan(z, x) \quad , \text{ und somit} \\ \theta_1 &= \arctan(z, x) - \beta \quad . \end{aligned} \quad (2.8)$$

Zusammengefasst ergeben sich also, von den Spezialfällen *Bein seitlich ausgestreckt* und *Fußpunkt nicht erreichbar* abgesehen, folgende Gelenkwinkel:

$$\begin{aligned} \theta_3 &= \arccos\left(\frac{x^2 + y^2 + z^2 - l_1^2 - l_2^2}{2l_1l_2}\right) \\ \theta_2 &= \arcsin\left(\frac{y}{l_1 + l_2 \cos(\theta_3)}\right) \\ \theta_1 &= \arctan(z, x) - \arctan\left((l_1 + l_2 \cos(\theta_3)) \cos(\theta_2), l_2 \sin(\theta_3)\right) \quad . \end{aligned} \quad (2.9)$$

2.2.3 Anatomiebedingte Korrektur

Die bisherigen Berechnungen gehen davon aus, dass bei einem komplett nach unten ausgestreckten Bein alle drei Gelenkwinkel 0 sind. Während das für ein Drahtgittermodell offensichtlich stimmt, weicht die Anatomie eines Sony Aibo etwas von

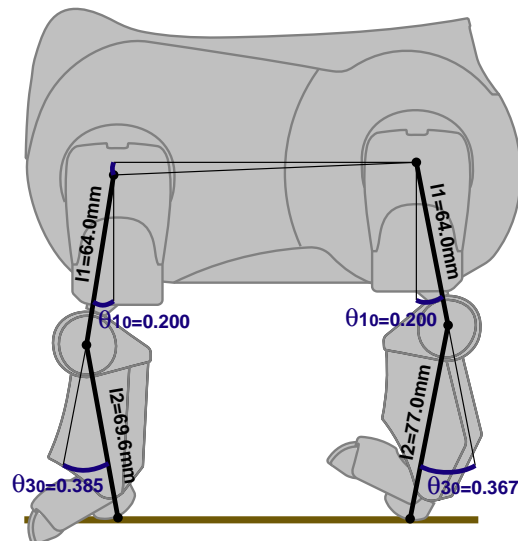


Abbildung 2.4: ERS-210: Das Anfordern eines Winkels von 0 Grad an allen Bein-
gelenkmotoren führt zwar zu senkrecht ausgestreckten Außenverkleidungen der Bei-
ne, aber nicht zu Winkeln von 0 Grad zwischen den Gelenken.

dieser Annahme ab. Hier führt das Anfordern eines Winkels von 0° an allen Gelenken zwar zu einem senkrecht ausgestreckten Bein, die tatsächlichen Winkel zwischen den Gelenken sind aber keineswegs 0 (siehe Abbildung 2.4).

Auch ohne Korrektur dieses Unterschiedes lassen sich relativ gute Laufparameter finden, nur sieht das reale Laufen anders aus, als vom fehlerhaften Modell vorhergesagt. Bis kurz nach der RoboCup-Weltmeisterschaft 2003 in Padua war sich das GermanTeam dieses Fehlers nicht einmal bewusst, dabei ist die Korrektur sehr einfach.

Nach korrekter Berechnung der Winkel θ_i zwischen den Gelenken werden durch Addition von Konstanten (siehe Abbildung 2.4) die Winkel ψ_i zwischen den Beinteilen bestimmt, die der Roboter zur Ansteuerung benötigt. Unter Einbeziehung dieser Korrektur ergeben sich somit folgende an die Beine zu sendende Winkel:

$$\begin{aligned}\psi_3 &= \theta_3 + \theta_{3_0} \\ \psi_2 &= \theta_2 \\ \psi_1 &= \theta_1 - \theta_{1_0} \quad .\end{aligned}$$

2.3 Ein einfaches Laufmodell

Jeder der verwendeten Roboter (siehe Abbildung 2.1) hat vier Beine mit je drei aktiven Gelenken, also 12 Schrittmotoren und damit 12 Freiheitsgrade. Alle 8 Millisekunden (=1 Frame²) muss jedem der Gelenke ein Zielwinkel vorgegeben wer-

²kleinste Zeiteinheit in der Kommunikation mit der Roboterhardware: der Empfang von Sensordaten und das Abschicken von Gelenkwinkeln erfolgt jeweils in Vielfachen eines Frames

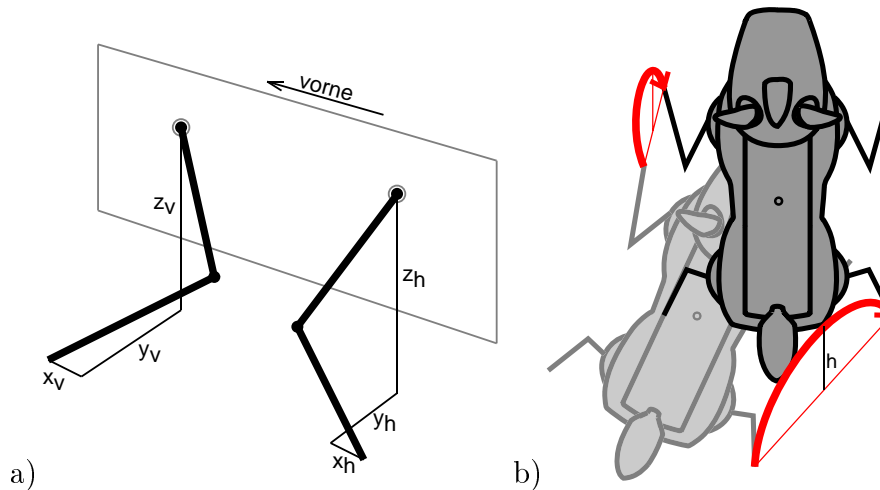


Abbildung 2.5: Das erste Laufmodell und seine Parameter: **a)** Die Seitenansicht enthält alle räumlichen Parameter. **b)** Die Draufsicht zeigt, dass die Schwerpunktsverlagerung des Roboters das Ziel der in Halbellipsen erfolgenden Fußbewegungen bei einem Schritt impliziert.

den. Um unter diesen Voraussetzungen eine brauchbare Laufbewegung erzeugen zu können, muss die Anzahl der Freiheitsgrade reduziert werden. Dazu sind geeignete Annahmen nötig, mögliche derartige Annahmen werden im Folgenden vorgestellt.

2.3.1 Stehen

Zunächst wird der einfachste Fall untersucht: Der Roboter steht, die Füße befinden sich also in konstanter Position relativ zum Körper. Dabei werden folgende Annahmen getroffen: Alle Schulter- (beziehungsweise Hüft-)gelenke sollen sich in gleicher Höhe befinden, die Füße hängen also quasi unter der frei schwebenden Schulterebene. Außerdem sollen die rechten und linken Beine symmetrisch zueinander stehen. Damit genügen zur Beschreibung des Stehens die in Abbildung 2.5 a) gezeigten noch freien Parameter. Die Gelenkwinkel, die nötig sind, um die aus der Wahl dieser Parameter resultierenden Fußpositionen zu erreichen, werden dann mittels Inverser Kinematik aus Abschnitt 2.2 berechnet.

2.3.2 Schwerpunktbewegung

Bevor einzelne Beine angesteuert werden, ist zu überlegen, dass sich der Roboter dadurch letztlich fortbewegen soll. Er soll also pro Zeiteinheit seinen Schwerpunkt um ein bestimmtes Stück verlagern und seine Ausrichtung um einen bestimmten Winkel drehen, und das möglichst gleichmäßig und ohne dabei zu stark zu wackeln. Mit den angeforderten Lauf- und Drehgeschwindigkeiten wird also zunächst nur die Position des Schwerpunktes des Roboters bewegt. Die jeweils am Boden verbleibenden Füße haben damit zu jeder Zeit eine bekannte Position relativ zum sich kontinuierlich bewegenden Körper.

Daraus lassen sich alle Gelenkwinkel der zugehörigen Beine mittels Inverser Kinematik berechnen. Die Position der am Boden verbleibenden Füße relativ zum Roboterkörper hängt somit nicht nur von der aktuell angeforderten Bewegung ab, sondern von der Schwerpunktbewegung seit dem Absetzen dieser Füße und damit auch von früheren Bewegungsanforderungen.

2.3.3 Schrittberechnung

Betrachten wir eine konstante Bewegungsanforderung. Dabei soll der Roboter nach einem Vollschrift (einem Schritt mit jedem Bein) wieder mit der gleichen Beinhaltung an einer neuen Position stehen. Also wird bei einem Schritt der jeweilige Fuß dorthin gesetzt, wo er sich einen Vollschrift später bei gleicher Beinhaltung befinden müsste.

Auf dem Weg dorthin soll sich der Fuß auf einer Halbellipse (siehe Abbildung 2.5 b) über dem Boden bewegen. Nach diesem Verfahren werden Füße bei einem Schritt also stets an die Positionen gesetzt, die für die aktuelle Bewegungsanforderung ideal erscheinen.

Während des Laufens werden Schritte dann begonnen, wenn die Füße in Laufrichtung weit voneinander entfernt sind: Die gerade abgesetzten Füße befinden sich vor ihrer Ruheposition, die anzuhebenden Füße dagegen hinter ihrer. Um aus dem Stand (alle Füße in Ruheposition) in eine Laufbewegung und umgekehrt zu kommen, muss daher einmalig mit zwei Beinen ein Schritt halber Länge (und Dauer) gemacht werden.

Der Einfachheit halber werden stets die beiden diagonal angeordneten Beine gleichzeitig gehoben und gesenkt. Das hat sich auf Antrieb als ausreichend stabil und zudem deutlich schneller herausgestellt, als die Beine einzeln zu setzen. Diese Gangart wird auch als *Trab* bezeichnet. Ein vierbeiniger Roboter, der zwei Füße in der Luft hat, ist nicht mehr statisch stabil, könnte also umkippen. In der Realität geschieht dies beim Laufen jedoch nicht, da die angehobenen Beine rechtzeitig wieder am Boden sind, der Roboter also dynamisch stabil ist.

Damit ergibt sich dann auch der Zeitpunkt des Beinhebens: Es sind stets zwei Beine am Boden (*Stemmphase*) und zwei in der Luft (*Schwingphase*), das eine Beinpaar wird sofort gehoben, das andere einen halben Vollschrift später. Jedes Bein ist somit genau die Hälfte der Zeit am Boden (*Duty-Factor* 0,5) und die andere Hälfte in der Luft.

2.3.4 Statisch vs. dynamisch stabiles Laufen

Als statisch stabil wird eine Bewegung bezeichnet, bei der der Schwerpunkt eines Roboters stets über einem Polygon von mindestens drei Berührungspunkten mit dem Boden ist (siehe Abschnitt 1.6), sodass der Roboter nicht umkippen kann. Es gibt Anwendungsfälle, in denen diese Forderung sinnvoll ist, für einen Roboter zum Transport von Gefahrgut zum Beispiel muss diese Eigenschaft unter Umständen sogar nachweisbar sein.

In den Fußball-Ligen des RoboCup erweist sich diese Forderung als unnötig. Dies deutete sich bereits bei der ersten RoboCup-Weltmeisterschaft mit einer Liga für

vierbeinige Sony-Roboter 1999 in Stockholm an. Das ab Werk integrierte statisch stabile Laufen war langsam und schien insbesondere langsamer als nötig zu sein.

Daraufhin beschlossen alle teilnehmenden Teams, ein eigenes, für den Einsatz im Roboterfußball optimiertes Laufmodell zu entwickeln. Wie die meisten verzichtete dabei auch das AiboTeamHumboldt erfolgreich auf statische Stabilität und ließ die Roboter stattdessen jeweils die diagonal angeordneten Beine gleichzeitig anheben und senken, sodass immer nur zwei Beine zu jeder Zeit Bodenkontakt hatten (Trab, Duty-Factor 0,5).

Nur wenige der teilnehmenden Teams versuchten, den statisch stabilen Ansatz weiter auszureizen, insbesondere das Team LRP³ [16] und anfangs auch das Team CMU⁴ [23]. Beide erzielten zwar deutliche Fortschritte, ihre Laufbewegungen waren aber letztendlich nicht konkurrenzfähig. Durch den hohen Duty-Factor (0,75, da stets drei von vier Beinen am Boden sind) wird die Laufgeschwindigkeit zu stark begrenzt. Das schnellste als statisch stabil modellierte Laufen mit einem Sony Aibo ERS-210 ist etwa halb so schnell wie das schnellste inzwischen bekannte, dynamisch stabile Laufen [21].

Die Forderung nach statischer Stabilität kann auch dazu führen, dass man sich in falscher Sicherheit wiegt. Ein statisch gerade noch stabiles Laufen kann dynamisch durchaus instabil werden, beispielsweise wenn Radialkräfte beim Kurvenlaufen auf den Roboter wirken. Dabei ist es schwierig, bei gegebener fehlerbehafteter Sensorik die für dynamische Stabilität einzuhaltenden Grenzwerte exakt genug zu berechnen. Der von statischer Stabilität erwartete Vorteil kann also bei ausreichender hoher Geschwindigkeit völlig verschwinden, und hohe Geschwindigkeit ist schließlich wesentlich im RoboCup.

2.3.5 Parameter

Die folgenden Parameter (siehe auch Abbildung 2.5) haben sich in ersten Experimenten als ausreichend für die Beschreibung des Laufens erwiesen und wurden manuell so gewählt, dass ein möglichst schnelles, noch stabiles Laufen entsteht:

- $x_v=24$ mm: Abstand der Vorderfüße zur Schulter in Körperrichtung
- $x_h=20$ mm: Abstand der Hinterfüße zur Schulter entgegen der Körperrichtung
- $y_v=17$ mm: seitlicher Abstand der Vorderfüße von der zugehörigen Schulter
- $y_h=17$ mm: seitlicher Abstand der Hinterfüße von der zugehörigen Schulter
- $z_v=103$ mm: Höhe der Schulter über den Vorderfüßen
- $z_h=104$ mm: Höhe der Schulter über den Hinterfüßen
- $h=25$ mm: Höhe des Beinanehens während eines Schrittes
- $T=128 \cdot 8$ ms=1024 ms: Dauer eines Vollschrilles (Schritt mit allen vier Beinen)

³Team des Laboratoire de Robotique de Paris, Frankreich

⁴Team der Carnegie Mellon University, Pittsburgh, USA

Dieses erste Laufmodell wurde von uns, dem AiboTeamHumboldt (damals Humboldt Heroes), beim RoboCup 2000 in Melbourne verwendet [36]. Es verhilft einem ERS-110 zu einer Geschwindigkeit von 140 mm/s im Gegensatz zu den knapp 100 mm/s des ab Werk integrierten Laufens.

2.4 Erweitertes Laufmodell

Das in Abschnitt 2.3 vorgestellte Laufmodell zeigte noch eine ganze Reihe von Schwächen. Einige davon sind auf zu starke Vereinfachung zurückzuführen, die für einen ersten Ansatz durchaus sinnvoll waren. Andere dagegen beruhen auf Fehleinschätzungen, die ligaweit zum Teil bis heute bestehen.

Im Folgenden wird daher ein erweitertes Laufmodell beschrieben, das ein gutes, omnidirektional nutzbares Laufen erlaubt und die inzwischen aufgedeckten Unzulänglichkeiten weitestgehend vermeidet. Außerdem sind die Resultate anderer Teams sowie die eigenen Erfahrungen der letzten Jahre berücksichtigt worden, um zu allen bisherigen Ansätzen mindestens konkurrenzfähig sein zu können.

2.4.1 Verbesserungen

Abhängigkeit idealer Fußpositionen vom Bewegungswunsch

Das Ausrechnen neuer idealer (Ruhe-)Fußpositionen für die beim nächsten Halbschritt⁵ benutzten Beine liefert nicht immer gute Ergebnisse, weil beim vorigen Schritt (mit anderen Beinen) eine ganz andere Bewegungsrichtung gewünscht worden sein kann. Dadurch stehen die auf dem Boden verbleibenden Füße noch günstig für den letzten, aber unter Umständen ungünstig für den neuen Schritt.

Dies ist auch der Hauptkritikpunkt am ersten Modell aus Abschnitt 2.3, bei dem sich verändernde Bewegungswünsche dazu führen konnten, dass der Roboter aufgrund der unvorhersehbar zueinander stehenden diagonalen Beinpaare ins Straucheln kam. Das 2000 von UNSW⁶ eingeführte „Rädermodell“ [13] umgeht diesen Nachteil und hat sich daher inzwischen ligaweit durchgesetzt. Folglich wird das Rädermodell auch in dieser Arbeit verwendet und dabei erklärt, warum dieses vereinfachende Modell dem vermeintlich genaueren so deutlich überlegen ist.

Aufrechter Gang vs. geneigter Roboterkörper

Welche Körperneigung am besten zum Laufen geeignet ist, hängt von der Anatomie des verwendeten Roboters ab. Ein Sony Aibo ERS-210 läuft erheblich schneller (und darüber hinaus stabiler), wenn er mit den Vorderbeinen quasi auf den Ellenbogen läuft. Erstmals unter Beweis gestellt wurde dies 2000 in Melbourne vom Team UNSW [13] (200 mm/s).

Die daraus resultierende vergleichsweise starke Neigung des Roboterkörpers nach vorne wurde im Modell in Abschnitt 2.3 noch nicht berücksichtigt, dort bewegen

⁵halber Vollschritt: vollständiger Schritt mit zwei der vier Beine, nicht Schritt halber Länge

⁶Team der University of New South Wales, Australien

sich die Füße stets relativ zur Ebene durch die Schultern, bei starker Neigung also nicht parallel zum Untergrund. Mit zusätzlichen Parametern kann die Neigung der Schulterebene und die der Fußtrajektorien zum Boden entkoppelt werden.

Inzwischen lassen alle Teams mit konkurrenzfähigem Laufen (stabil, >200 mm/s) ihre Sony Aibos ERS-210 mit abgesenktem Oberkörper laufen, und das mit bis zu 270 mm/s [5], 291 mm/s [21], 295 mm/s [30] beziehungsweise 311 mm/s [32]. Das schnellste bekannte aufrechte Laufen ist nur etwa halb so schnell. Die Anatomie des Nachfolgemodells ERS-7 begünstigt das Laufen auf den vorderen Unterschenkeln zusätzlich durch abgerundete Unterschenkel und führt unter anderem dadurch zu nochmals deutlich höherer Laufgeschwindigkeit.

Lokale Optimier- und Kalibrierbarkeit

Das einfache Laufmodell ging davon aus, dass sich der Roboter mit den gleichen Parametern, etwa Schulterhöhe oder Dauer eines Schrittes, in alle Richtungen bewegen kann. Es hat sich jedoch herausgestellt, dass Spezialaufgaben wie schnelles Drehen oder schnelles Geradeauslaufen mit modifizierten Parametern deutlich besser zu bewältigen sind. Das erweiterte Laufmodell in diesem Kapitel wird das berücksichtigen, um beispielsweise nicht wie bisher eine Verbesserung des Vorwärtslaufens mit einer Verschlechterung des Rückwärtslaufens erkaufen zu müssen.

Außerdem gab es bisher kein gutes Konzept, um die Abweichungen zwischen Theorie und Praxis in das Laufmodell einfließen zu lassen. Nachträglich eingeführte, globale lineare Korrekturparameter sorgten wenigstens dafür, dass die real erreichten Maximalgeschwindigkeiten für (jeweils einzelnes) Vorwärts-, Rückwärts-, Seitwärtslaufen und Drehen mit den Vorhersagen des Modells übereinstimmten.

Diagonales Laufen, Laufen mit halber Geschwindigkeit und auch die Mischung von Laufen und Drehen wichen dagegen mitunter deutlich von der Theorie ab. Das Laufmodell in diesem Kapitel wird, um dies zu vermeiden, erstmals lokal kalibrierbar sein.

2.4.2 Rädermodell

Bei der Verwendung vierbeiniger Roboter läge es eigentlich nahe, das Vorhandensein von Beinen auszunutzen, die Füße also zum Beispiel unabhängig voneinander auf nach bestimmten Kriterien berechnete Optimalpositionen zu setzen. Die separate Ansteuerung einzelner Beine kann für Spezialfälle wie das Ertasten unbekanntes Gelände durchaus sinnvoll sein.

Für normales Laufen ist sie eher hinderlich, weil es viel schwieriger ist, dabei eine geeignete Stellung der Beine zueinander in allen Fällen sicherzustellen und damit die Stabilität des Laufens zu gewährleisten, was an dem in Abschnitt 2.3 benutzten Laufmodell zu sehen war. Gutes Laufen erfolgt stattdessen meist nach einem vorgegebenen Schema, in der Natur sind das Central-Pattern-Generatoren⁷ [8] und bei Sony Aibos das im Jahr 2000 von UNSW eingeführte [13] und seitdem verfeinerte „Rädermodell“.

⁷aus der Natur bekannter Mechanismus, die Bewegung von Extremitäten durch ein zentral generiertes Muster anzuregen

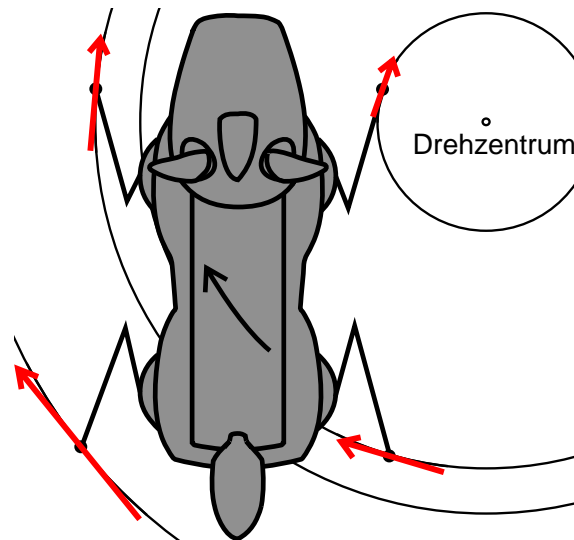


Abbildung 2.6: Gleichzeitiges Laufen und Drehen mit Rädermodell: Die Ruhepositionen der Füße bewegen sich dabei auf Kreisbahnen um ein gemeinsames Drehzentrum. Realisiert wird dies durch die aus den gedachten Radbewegungen abgeleiteten Schritte (rot) tangential zu den optimalen Kreisbahnen.

Aus jeder konstanten Bewegungsanforderung bestehend aus Vorwärts-, Seitwärts- und Drehgeschwindigkeit resultiert die Bewegung des Roboters auf einer Kreisbahn um ein fixes Drehzentrum (*Instantaneous Center of Rotation*, ICR). Folglich bewegen sich auch die Ruhepositionen der Füße des Roboters auf Kreisbahnen verschiedenen Durchmessers um dieses Drehzentrum. Ist die Drehgeschwindigkeit 0, so befindet sich das Drehzentrum im Unendlichen beziehungsweise die Ruhepositionen aller Füße bewegen sich mit gleicher Geschwindigkeit in die gleiche Richtung.

Stellt man sich nun die Füße als an ihren Ruhepositionen montierte winzige Räder vor, so folgt für jedes Rad, mit welcher Geschwindigkeit es sich in welche Richtung zu drehen hat, nämlich jeweils entlang der fußeigenen Kreisbahn um das gemeinsame Drehzentrum (siehe Abbildung 2.6). Statt Drehung winziger Räder macht nun jeder Fuß einen Schritt entsprechender Größe in die entsprechende Richtung, also tangential zu seinem Kreis um das Drehzentrum.

Die Fußpositionen relativ zum Roboterkörper sind beim Rädermodell ausschließlich von der aktuellen Bewegungsanforderung und der zeitlichen Position innerhalb eines Vollschrittes abhängig, nicht jedoch von alten Bewegungsanforderungen oder Fußpositionen. Dadurch ist zu jeder Zeit eine konsistente Stellung aller Beine zueinander sichergestellt.

Diese Eigenschaft stellt einen wesentlichen Unterschied zum ersten Laufmodell aus Abschnitt 2.3 dar. Dort verbleiben die jeweils nicht am Schritt beteiligten Füße an konstanter Position am Boden. Die Stellung dieser Füße relativ zu ihrer jeweiligen Schulter hängt damit von der Änderung der Lage des Schwerpunktes des Roboters ab, also keineswegs nur von der aktuellen Bewegungsanforderung, sondern von der

Entwicklung der Bewegungsanforderung seit dem Absetzen dieser Füße. Bei Änderung der angeforderten Roboterbewegung kann es also passieren, dass die am Boden verbliebenen Füße ungünstig für die aktuell angeforderte Bewegung stehen. Das Laufen wirkt bei starker Änderung der Bewegungsrichtung daher unsicher und der Roboter strauchelt gelegentlich, obwohl die gewünschte Fußposition zum Zeitpunkt des Absetzens genau bestimmt worden war.

Das Rädermodell toleriert dagegen zugunsten einer konsistenten Stellung der Beine zueinander bewusst die fehlerhafte Berechnung einzelner Fußpositionen. Bei von Null verschiedener Schrittlänge werden die Füße nicht exakt auf ihren Ideal-Kreisbahnen gesetzt, sondern auf eine Strecke tangential dazu (siehe Abbildung 2.6). Die Bewegung eines am Boden verbleibenden Fußes relativ zur Schulter wird stets als Gerade modelliert, obwohl es sich bei Drehung eigentlich um einen Kreisbogen handeln müsste. Ändert sich die Bewegungsanforderung, werden auch die am Boden befindlichen Füße in die neue korrekte Stellung zu ihrer Schulter gebracht.

Durch all diese Effekte können einzelne am Boden befindliche Füße gezwungen werden, ihre alte Position zu verlassen, also zu rutschen, wenn das erforderlich ist, um die Beine in eine konsistente Stellung zueinander entsprechend der aktuellen Bewegungsanforderung zu bringen.

Was beim Rädermodell wie ein Rechenfehler aussieht, dient also der Stabilität des Laufens und so letztlich auch der Geschwindigkeit, weil Straucheln weder auftritt noch durch künstliche Beschränkung von Geschwindigkeit oder Beschleunigung unterbunden werden muss. Daher ist das Rädermodell trotz dieser Ungenauigkeiten der vermeintlich genaueren Berechnung aus Abschnitt 2.3 deutlich überlegen.

2.4.3 Parameter

Hier werden bewusst nur diejenigen Parameter verwendet, die nach den Erfahrungen der letzten Jahre einen entscheidenden Einfluss auf Geschwindigkeit und Qualität des Laufens haben. Diese lassen sich in vier Gruppen einteilen:

1. Parameter zur Bestimmung der Ruheposition der Füße relativ zum Körper, wie sie auch schon im ersten Modell aus dem vorigen Kapitel benutzt wurden:
 - x_v, y_v, z_v : Position eines Vorderfußes relativ zum Mittelpunkt zwischen den beiden vorderen Schultern
 - x_h, y_h, z_h : Position eines Hinterfußes relativ zum Mittelpunkt zwischen den beiden hinteren Schultern
2. Parameter zur Beschreibung der Trajektorie der Füße (siehe Abbildung 2.7):
 - h_v, h_h : maximale Höhe der Füße über dem Boden während eines Schrittes
 - $tilt_v, tilt_h$: Tangens des Winkels der theoretischen Fußtrajektorie zum Boden; durch die damit beschriebene Neigung der Fußtrajektorie kann man bestimmen, wie energisch der Roboter auf den Boden auftreten oder die Füße am Boden belassen soll, um damit die Bodenhaftung zu erhöhen und Wegrutschen zu vermeiden

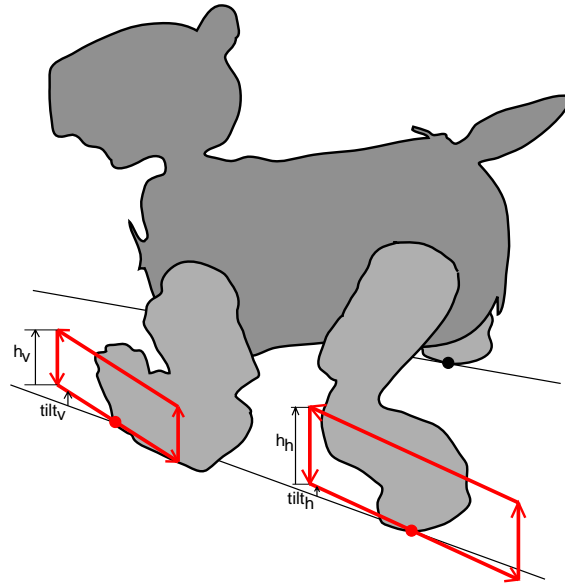


Abbildung 2.7: Die Parameter der Fußbewegung im verwendeten Laufmodell: Fußbewegungen erfolgen in Parallelogrammen, deren Richtung und Länge sich aus dem Rädermodell ergibt.

3. Parameter zur Bestimmung der Gangart:

- gp_v, gp_h : Zeitanteil eines Vollschrilles, in dem ein Vorder- beziehungsweise Hinterfuß theoretisch Bodenkontakt hat
- $l = (l_{vl}, l_{vr}, l_{hl}, l_{hr})$: relative Zeit des Anhebens für jedes der vier Beine, $(0, 0,5, 0,5, 0)$ beschreibt beispielsweise den hauptsächlich verwendeten Trab und bedeutet, dass der linke Vorder- und der rechte Hinterfuß jeweils zu Beginn eines Vollschrilles (0) gehoben werden, die anderen beiden dagegen einen halben Vollschrift später (0,5)

4. T : die Dauer eines Vollschrilles in Frames à 8 Millisekunden

In der Vergangenheit wurde mehrfach gezeigt, dass durch die Einführung zusätzlicher Parameter Verbesserungen möglich sind, zum Beispiel durch die bewusste Variation der Körperhöhe während des Laufens (*CanterAction*, siehe [14]) oder das Zulassen komplexerer Fußtrajektorien als Ellipsen oder Parallelogramme (*FreeFormQuad*, siehe [5]). Zum einen hat sich aber herausgestellt, dass hohe Geschwindigkeiten nahe der maximal bekannten Geschwindigkeit auch ohne derartige Parameter möglich sind [21], und zum anderen ist bei diesen Parametern meist völlig unklar, ob sie in die Omnidirektionalität übertragen werden können und wie sie dabei angepasst werden müssen.

Während zum Beispiel eine bewusste Gewichtsverlagerung quer zur Laufrichtung beim Vorwärtslaufen (Verlagerung von rechts nach links) sinnvoll sein kann, ist sie beim Seitwärtslaufen (Verlagerung von vorne nach hinten) hingegen aufgrund anderer Gewichts- und Symmetrieverhältnisse vielleicht eher hinderlich.

Wegen ihrer unbekanntem Einflüsse auf omnidirektionales Laufen wird hier eine Reihe weiterer Parameter bewusst weggelassen, obwohl bekannt oder anzunehmen ist, dass sie das Laufen zumindest in Einzelfällen verbessern können. Dadurch erspart man sich eine erhebliche Erhöhung der Komplexität der Optimierung sowie die Untersuchung fehlerhafter Annahmen zu den Auswirkungen dieser zusätzlichen Parameter auf omnidirektionales Laufen.

2.4.4 Parametersatzauswahl und -interpolation

Bisher war nur von einem Parametersatz und dessen Bestandteilen die Rede. Es hat sich jedoch herausgestellt, dass für verschiedene Bewegungsanforderungen auch verschiedene Parametersätze optimal sind. Das GermanTeam hat 2003 zum Beispiel einen Parametersatz fürs Vorwärts- und einen fürs Rückwärtslaufen verwendet und beim Wechsel zwischen beiden interpoliert.

Also wurde ein Modell gesucht, das diesen Sachverhalt möglichst umfassend berücksichtigt. Die Verwendung mehrerer Parametersätze bietet außerdem den Vorteil, auf die in der Vergangenheit benutzten globalen Korrekturparameter, etwa zur Kompensation unerwünschter Drehbewegung beim Seitwärtslaufen, verzichten zu können, weil man einfach für bisher schlecht korrigierte Fälle einen weiteren Parametersatz verwenden kann.

Aus diesen Überlegungen heraus entstand ein Modell, das 127 verschiedene Parametersätze benutzt. Ein solcher Parametersatz enthält zusätzlich zu den bisherigen Parametern noch Informationen darüber, für welche Bewegungsanforderung er optimiert ist und welche Bewegung man ansteuern muss, um genau diese Anforderung umzusetzen. Da omnidirektionales Laufen bereits mit einem Parametersatz möglich ist, scheinen 127 Parametersätze auf den ersten Blick relativ viele zu sein. Knapp die Hälfte davon geht durch Rechts-Links-Symmetrie aus den anderen hervor. Trotzdem spricht aus Gründen der Übersichtlichkeit vorerst nichts dagegen, 127 Parametersätze zu verwenden, die sich initial ausschließlich dadurch unterscheiden, für welche Bewegungsanforderung sie gedacht sind.

Einzelne Parametersätze können anschließend gezielt optimiert und kalibriert werden, wobei zur Vermeidung unnötiger Arbeit auch Symmetrien berücksichtigt werden. Wenn an dieser Stelle bereits bekannt wäre, wodurch sich die optimierten Parametersätze voneinander unterscheiden, könnte man statt derart vieler Parametersätze einfach wenige Korrekturparameter verwenden.

Alle bisherigen Annahmen in diese Richtung haben sich jedoch als ungenau oder unnötig beschränkend herausgestellt. Durch die offenbar auftretenden Nichtlinearitäten hat man bei Verwendung eines einzigen Parametersatzes nur die Wahl, einen Kompromiss zwischen erreichbarer durchschnittlicher Genauigkeit und erreichbarer Maximalgeschwindigkeit zu finden. Mehrere Parametersätze hingegen lassen sich unabhängig voneinander kalibrieren und optimieren.

Bewegungsanforderungen haben üblicherweise drei Komponenten, nämlich Vorwärts-, Seitwärts- und Drehgeschwindigkeit $(\dot{x}, \dot{y}, \dot{\varphi})$, aus denen sich bei Bedarf die Laufgeschwindigkeit $v = |(\dot{x}, \dot{y})|$ berechnen lässt. Für die Auswahl eines geeigneten Parametersatzes sind diese drei Komponenten ungünstig, weil ein gleich-

mäßiges Raster über sie die interessanten Bereiche zu ungenau, die uninteressanten oder unerreichbaren (wie maximal schnelles Laufen und maximal schnelles Drehen gleichzeitig) hingegen zu genau unterteilt.

Statt $(\dot{x}, \dot{y}, \dot{\varphi})$ werden daher drei andere Komponenten verwendet, nämlich Laufrichtung, Dreh-Lauf-Verhältnis und Gesamtgeschwindigkeit (α, δ, r) , das entspricht also einer Transformation in Polarkoordinaten. Die Verwendung von acht Laufrichtungen, drei Geschwindigkeitsstufen und sieben Dreh-Lauf-Verhältnissen führt zu 127 sinnvollen Kombinationen, da Drehungen ohne Laufanteil keine Laufrichtung benötigen (siehe Abbildung 2.8).

Die Wahl der zur Normierung verwendeten Höchstgeschwindigkeiten $v_{max} = 300$ und $\dot{\varphi}_{max} = 2,7$ orientiert sich an den bisher mit einem ERS-210 erreichten Geschwindigkeiten, stellt aber keine Beschränkung dar, durch künftige Optimierung sind also auch normierte Geschwindigkeiten größer 1 zulässig. Die verwendeten Komponenten einer Bewegungsanforderung sind somit:

$$\begin{aligned}
 \text{Laufrichtung } \alpha &= \arctan(\dot{x}, \dot{y}) \\
 &\rightarrow \left[-\pi, -\frac{3\pi}{4}, -\frac{\pi}{2}, -\frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4} \right] \\
 \text{Dreh-Lauf-Verhältnis } \delta &= \frac{2}{\pi} \arctan\left(\frac{v}{v_{max}}, \frac{\dot{\varphi}}{\dot{\varphi}_{max}}\right) \\
 &\rightarrow \left[\underbrace{-1}_{\text{Rechtsdrehung}}, -\frac{3}{10}, -\frac{1}{10}, 0, \frac{1}{10}, \frac{3}{10}, \underbrace{1}_{\text{Links-drehung}} \right] \\
 \text{Gesamtgeschwindigkeit } r &= \sqrt{\left(\frac{v}{v_{max}}\right)^2 + \left(\frac{\dot{\varphi}}{\dot{\varphi}_{max}}\right)^2} \\
 &\rightarrow [\text{langsam, mittelschnell, schnell}] \quad .
 \end{aligned}$$

Während es sich bei den Laufrichtungen und den Dreh-Lauf-Verhältnissen um feste Vorgaben handelt, sind die drei Geschwindigkeitsstufen relativ zu interpretieren, also als *langsam*, *mittelschnell* und *maximal schnell*. Das hat die Vorteile, zum einen für jede Lauf- und Drehrichtung eine separate Maximalgeschwindigkeit mit eigens optimiertem Parametersatz angeben zu können, und zum anderen für einen möglichst großen Bereich, nämlich für alle nicht maximal schnellen Bewegungsanforderungen, die gleichen Parameter verwenden zu können und jeweils nur die Ansteuerung kalibrieren zu müssen. Zudem können die drei Geschwindigkeitsstufen jeweils so gewählt werden, dass die durchschnittliche Abweichung zwischen Bewegungsanforderung und realer Bewegung über den gesamten Geschwindigkeitsbereich minimal ist.

Für eine konkrete Bewegungsanforderung wird ein passender Parametersatz durch lineare Interpolation aus den acht benachbarten (jeweils rechter und linker Nachbar pro Raumdimension α , δ und r , Nachbarn sind nicht notwendigerweise alle verschieden) der 127 Parametersätze berechnet. Dazu werden zuerst die vier benachbarten Interpolationen für exakte Laufrichtung berechnet, daraus die zwei benachbarten

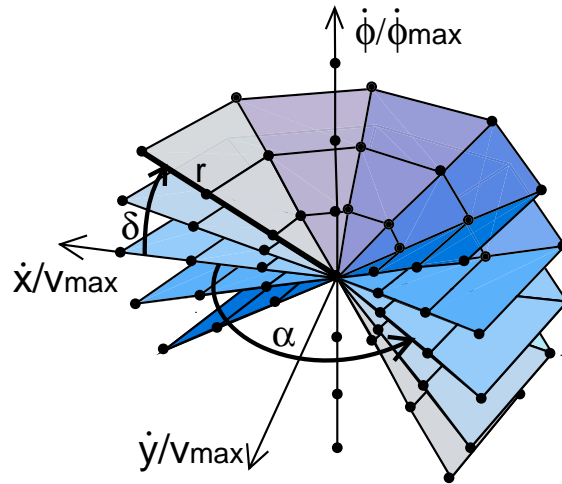


Abbildung 2.8: Die Lage der verwendeten 127 Parametersätze (schwarze Punkte: Stand, 6 Drehungen, $8 \times 5 \times 3$ mit Laufen): Der Azimut α beschreibt die Laufrichtung, die Deklination δ das normierte Dreh-Lauf-Verhältnis und der Radius r die normierte Gesamtgeschwindigkeit.

Interpolationen für exaktes Dreh-Lauf-Verhältnis, und aus diesen beiden schließlich die lineare Interpolation für exakte normierte Gesamtgeschwindigkeit.

Damit ist ein kontinuierlicher Übergang zwischen allen verwendeten Parametersätzen sichergestellt. Eine kontinuierliche Änderung der angeforderten Bewegung führt also auch zu einer kontinuierlichen Änderung der letztendlich verwendeten Parameter, unabhängig davon, aus welchen der 127 Parametersätze sie eigentlich zusammengesetzt sind.

Außerdem ist auf diese Weise für jede Kombination aus Laufrichtung und Dreh-Lauf-Verhältnis eine Maximalgeschwindigkeit bekannt. Lineare Interpolation wird dabei verwendet, weil sie als Einzige (zum Beispiel im Gegensatz zu einer auf Polarkoordinaten beruhenden Interpolation) keine Annahmen über die Abhängigkeit einzelner Parameter von der Laufrichtung macht, einfach zu berechnen ist und so eine realistische Abschätzung für die Mischung zweier Parametersätze liefert.

2.4.5 Parameterbestimmung

Das Modell beschränkt sich bewusst auf diejenigen Parameter, die nach der Erfahrung der letzten Jahre einen spürbaren Einfluss auf Güte und Geschwindigkeit des Laufens haben. Dennoch sind an dieser Stelle noch zwei entscheidende Fragen offen, nämlich welche Werte für diese Parameter in Kombination zu einem guten Laufen führen und wie schnell das aus diesen Werten resultierende Laufen dann ist beziehungsweise welche Ansteuerung zur gewünschten Geschwindigkeit führt. Mit diesen Fragen wird sich das nächste Kapitel befassen.

3 Optimierung

Oft unterscheiden sich gute Laufbewegungen von schlechten gar nicht durch die Wahl des Laufmodells, sondern allein durch die Wahl geeigneter Parameter für das gleiche Modell. Das im vorigen Kapitel vorgestellte Laufmodell hat sehr viele Parameter, obwohl bereits bewusst auf einige von anderen Teams benutzte Parameter verzichtet wurde. Um die Fähigkeiten eines solchen Laufmodells ausnutzen zu können, müssen gute Werte für eine Vielzahl von Parametern gefunden werden. In diesem Kapitel wird daher untersucht, auf welche Weise sich bessere Werte für die Parameter eines gegebenen Laufmodells bestimmen lassen.

3.1 Optimierungsmethoden

Neben dem aufwändigen und ineffizienten manuellen Ausprobieren verschiedener Parameter existieren auch mehrere Ansätze, mit denen man schneller zu guten Parametersätzen gelangen kann. Unter diesen Ansätzen werden hier diejenigen Verfahren ausgewählt, die für den konkreten Fall der Optimierung der Laufbewegungen eines Sony Aibo geeignet erscheinen.

3.1.1 Berechenbarkeit eines optimalen Laufmusters

Es gibt mehrere Bestrebungen, ein optimales Laufen für Aibos oder vergleichbare Roboter zu berechnen, zum Beispiel [12, 11], die kontinuierlich bessere Ergebnisse liefern. Allerdings sind diese bisher nicht ohne weiteres auf reale Roboter zu übertragen. Alle bisher auf theoretischem Wege erzielten Laufbewegungen waren in der Praxis deutlich von ihrer berechneten Maximalgeschwindigkeit entfernt oder zumindest den manuell oder automatisiert auf realen Robotern optimierten Laufbewegungen spürbar unterlegen, erreichen also nicht deren Geschwindigkeit oder Stabilität.

Offenbar existieren also für die Zielplattform Aibo noch zu viele nicht oder schlecht modellierte Parameter. Während qualitative Aussagen, etwa dass eine bestimmte Gangart wie Trab viel versprechend aussieht, durchaus aus der Berechnung eines Laufmusters zu erhalten sind, können quantitative Aussagen wie ein guter Wert für einen bestimmten Parameter derzeit in Experimenten auf realer Hardware wesentlich besser gefunden werden. Aus diesem Grund wird die Optimierung in dieser Arbeit ausschließlich auf realen Robotern in ihrer Zielumgebung durchgeführt.

3.1.2 Von der Natur inspirierte Laufmuster

Die Übernahme von Laufmustern und Laufparametern aus der Natur führt nur dann zu guten Ergebnissen, wenn auch die zugehörige Anatomie stark an die Natur angelehnt ist, zum Beispiel bei [29, 17, 25, 28]. Das optimale Laufen für einen

bestehenden Roboter wie Sony Aibo kann aber ganz anders aussehen als das für ein Lebewesen mit vergleichbaren Proportionen, weil zahlreiche anatomische Details sehr unterschiedlich sind.

Die offensichtlichste Folge aus diesen Unterschieden dürfte das Laufen auf den vorderen Ellenbogen sein, das dem uneingeweihten Betrachter aufgrund der Proportionen des Roboters sehr merkwürdig erscheint, aber für einen Aibo weitaus stabiler und effizienter als das Laufen mit ausgestreckteren Beinen ist. Dafür gibt es sogar Vorbilder in der Natur, nämlich in Australien beheimatete rattenartige Tiere, die zwar kleiner als ein Aibo sind, sich aber bei vergleichbarer Anatomie schnell mit vorne nach unten geneigtem Körper fortbewegen.

Es gibt zahlreiche Forschungsarbeiten mit dem Ziel, das Laufen der biologischen Vorbilder zu verstehen, indem Reflexe und Central-Pattern-Generatoren nachgebaut werden [3, 24, 18, 19]. Solche allgemeinen Funktionsprinzipien können im Gegensatz zu konkreten Körperneigungswinkeln oder Schrittlängen sehr wohl auf bestehende Roboter übertragen werden. Für die meisten Reflexe (Beugereflexe bei Kollisionen, vestibuläre Reflexe bei Neigungsveränderungen) scheint die Mechanik eines Sony Aibo allerdings zu träge zu sein. Da darüber hinaus Reflexe vor allem in unbekanntem Terrain benötigt werden, wird auf sie in dieser Arbeit nicht weiter eingegangen.

3.1.3 Vorgehensweise

Da sich gute Werte für die Vielzahl an Parametern des Laufmodells weder praxistauglich berechnen noch aus der Natur übernehmen noch auf einfache Weise manuell bestimmen lassen, bietet sich eine so weit wie möglich automatische Optimierung dieser Parameter mittels genetischer Algorithmen¹ oder Reinforcement-Learning² an. Dazu sind aber einige Vorbereitungen nötig.

Wie bereits von den ersten Implementatoren eines leistungsfähigen Laufens mit Sony Aibo ERS-210 [14] richtig festgestellt wurde, ist mit der Vorbereitung einer geeigneten Testumgebung einiger Aufwand verbunden, was in Abschnitt 3.2 beleuchtet wird.

Dies ist nicht der erste Versuch, einem Sony Aibo mittels Verfahren wie genetischen Algorithmen zu besserem Laufen zu verhelfen. Gregory S. Hornby [15] fehlte vor mehreren Jahren die Erfahrung von fünf Jahren RoboCup, weshalb seine richtungsweisenden Ergebnisse heute im RoboCup-Umfeld nicht mehr konkurrenzfähig sind. Anderen Ansätzen mangelte es an geeigneter Automatisierbarkeit oder einem praxistauglichen Gütemaß, zum Beispiel wurde nur maximale Geradeausgeschwindigkeit verwendet [30, 21], während hier der Schwerpunkt auf der Optimierung omnidirektionalen Laufens liegt.

Die Anforderung, omnidirektional laufen zu können, ist nicht selbstverständlich. Die Sony Aibos haben aber bereits gezeigt, dass sie dazu in der Lage sind, wes-

¹heuristische Optimierungsverfahren, die meist nur die Parameter eines bereits gegebenen Lösungsansatzes optimieren, wobei die biologische Evolution als Vorbild dient

²maschinelles Lernverfahren, das die Suche nach besseren Werten dort fortsetzt, wo es am erfolversprechendsten erscheint, also beispielsweise in Richtung der größten durchschnittlichen Verbesserung im vorigen Schritt

halb man das in einem übergeordneten Verhalten auch ausnutzen möchte. Daher wird zunächst (in Abschnitt 3.3) ein Laufen gesucht, das mit konstanten Parametern omnidirektional verwendbar und dabei schnell und stabil ist. Ein brauchbarer Startpunkt dafür ist der bisher vom GermanTeam verwendete Laufparametersatz. Am Ende dieses Schrittes hat man einen Parametersatz für das Laufmodell, mit dem sich der Roboter zwar in alle Richtungen akzeptabel bewegen kann, aber nichts davon besonders gut beherrscht.

Es hat sich jedoch gezeigt, dass Einzelaufgaben wie Geradeauslaufen deutlich schneller zu bewältigen sind [21], ein darauf optimiertes Laufen aber nur noch sehr schlecht diagonal oder rückwärts funktioniert. Als Kompromiss verwendeten wir im GermanTeam zur Weltmeisterschaft 2003 in Padua [33] die dynamische Überblendung zwischen zwei verschiedenen Parametersätzen für Vorwärts- und Rückwärtslaufen.

Das in Abschnitt 2.4 vorgestellte Laufmodell kann je nach Bewegungsanforderung weitaus mehr spezialisierte Parametersätze benutzen. Ziel der Untersuchungen in Abschnitt 3.4 ist es daher, für jede Bewegungsanforderung zu entscheiden, welcher Parametersatz dafür der geeignetste ist und deshalb dafür verwendet werden soll. Dazu müssen alle in Frage kommenden Parametersätze vermessen, verglichen und wenn notwendig optimiert werden.

In Abschnitt 3.5 schließlich werden die ausgewählten Parametersätze noch kalibriert, also nur durch Modifikation der angesteuerten Geschwindigkeiten dafür gesorgt, dass die tatsächlich ausgeführte Bewegung möglichst exakt der angeforderten entspricht.

3.2 Lokalisierung

In der Sony-Liga des RoboCup verwenden die Roboter zur Lokalisierung normalerweise einzeln stehende, farbkodierte Landmarken am Rande des Spielfeldes. Die darauf aufbauende, vom GermanTeam und zahlreichen weiteren Teams verwendete Lokalisierungsmethode ist für die automatische Suche nach besseren Laufparametern aus mehreren Gründen ungeeignet: Die Lokalisierungsqualität ist positionsabhängig, sie ist beispielsweise in der Feldmitte schlechter als dicht an einer Landmarke. Des Weiteren ist die Lokalisierung lichtabhängig, da eine Reihe von farbkodierten Landmarken möglichst exakt erkannt und von anderen farbigen Objekten in der Umwelt unterschieden werden muss.

Aufgrund der daraus resultierenden Unsicherheiten hat sich die Verwendung eines probabilistischen Ansatzes (Monte-Carlo-Lokalisierung, siehe [9]) als sinnvoll erwiesen. Eine solche Methode vermeidet zwar robust Sprünge in der Positionsbestimmung, ist aber andererseits auch verhältnismäßig träge, die modellierte Position folgt der tatsächlichen also nur verzögert. Weiterhin geht ein probabilistisches Verfahren davon aus, dass man mit einer wahrscheinlichen Position mehr anfangen kann als mit gar keiner oder einer aus zwei Möglichkeiten gemittelten. Während diese Annahme im Fußballspiel sicher zutrifft, stimmt sie für die Bewertung eines Laufparametersatzes nicht.

Außerdem wird eine als bekannt vorausgesetzte Odometrie zur Weiterführung der Position bei nicht vorhandener oder inkonsistenter Sichtinformation verwendet. Bei der Untersuchung eines noch unvermessenen Laufparametersatzes soll hingegen die Geschwindigkeit aus der Position des Roboters bestimmt werden und nicht andersherum.

Für die Optimierung des Geradeauslaufens benutzen mehrere Teams (UNSW³, Austin Villa⁴) einfach zwei Landmarken, die auf einem Spielfeld der Sony-Liga ohnehin vorhanden sind. Zwischen diesen wird hin- und hergelaufen. Dadurch muss nur die Position der dichten Marke genauer bestimmt werden.

Dieser Ansatz, zuerst beschrieben von Hornby [15], ist bestechend einfach und erlaubt die für maximal schnelles Geradeauslaufen ausreichende Geschwindigkeitsmessung. Leider ist er aber für das Optimieren omnidirektionalen Laufens nicht so gut geeignet, weil man damit zum Beispiel nicht den Abstand zur Optimallinie bestimmen kann und auch nicht zwischen Abweichung durch Drehung und Abweichung durch Seitwärtslaufen unterscheiden kann. Die Hinzunahme weiterer Markierungen wie Feldlinien löst dieses Problem nicht zufrieden stellend, weil die Linien ungleichmäßig über das Feld verteilt und untereinander leicht verwechselbar sind. Die gesuchte Position des Roboters lässt sich damit nicht immer beziehungsweise nicht genau bestimmen.

Häufiger sieht man auch den Einsatz von Deckenkameras zur Bestimmung der genauen Position von Robotern. Damit sind aber mehrere Nachteile wie perspektivische Verzerrungen, externe Bildverarbeitung, notwendige kontinuierliche Funkverbindung⁵ und zusätzliche Anschaffungskosten verbunden. Für genaue Messungen sind eine hochauflösende Kamera, je nach verfügbarer Räumlichkeit ein Weitwinkelobjektiv und in der Regel eine zusätzliche Markierung der Roboter nötig.

Außerdem spielt es bei der Verwendung einer Deckenkamera keine Rolle mehr, ob die zu untersuchende Bewegung die Wahrnehmung des Roboters zum Beispiel durch starkes Wackeln beeinträchtigt. Andere externe Positionsbestimmungsmöglichkeiten wie Laserscanner wurden aus vergleichbaren Gründen nicht in die engere Wahl gezogen.

3.2.1 Orientierungshilfe

Die bisherigen Lokalisierungsmethoden sind also für die Bestimmung der noch unbekanntes Laufgeschwindigkeit und Qualität eines neuen, auf Omnidirektionalität ausgelegten Laufparametersatzes schlecht geeignet. Deshalb wurde die in Abbildung 3.1 gezeigte Lokalisierungshilfe entwickelt, die die genannten Probleme weitgehend vermeidet. Sie kommt im Gegensatz zu Hornby [15] ohne das Laufen behindernde Kabel aus und setzt weder externe Sensoren wie Laserscanner oder Deckenkameras noch ei-

³University of New South Wales, Australien

⁴University of Texas at Austin, USA

⁵Netzwerkverkehr erhöht bei den verwendeten Robotern die Prozessorlast und führt infolgedessen zu stockenden Bewegungen

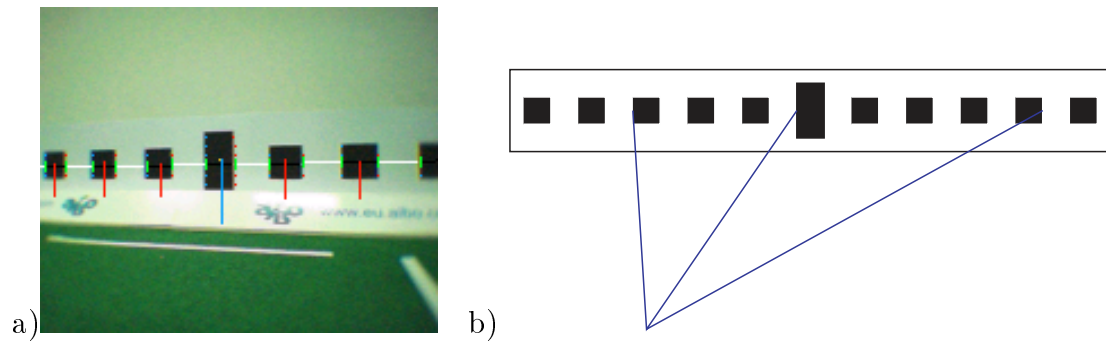


Abbildung 3.1: Die zur Lokalisierung verwendete Orientierungshilfe: **a)** Ein Bild aus der Kamera eines ERS-210 mit als erkannt markierten Teilen der Lokalisierungshilfe, **b)** Schematische Darstellung von Lokalisierungshilfe und möglichen Winkeln für die Positionsberechnung

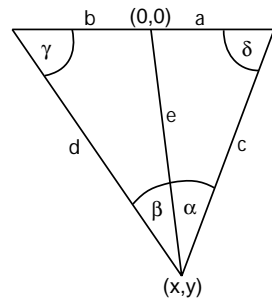
ne ebenfalls das Laufen beeinträchtigende kontinuierliche Funkverbindung zu diesen voraus.

Die entwickelte Lokalisierungshilfe besteht aus elf weißen A4-Blättern, auf die jeweils ein schwarzes Rechteck geklebt ist. Damit ist sie transportabel und bietet sehr hohe Kontraste, kann also ohne weitere Kalibrierung der Bildverarbeitung in verschiedenen Umgebungen eingesetzt werden, solange es ausreichend hell ist und keine anderen ähnlich kontrastreichen Gegenstände in der Nähe zu sehen sind. Das mittlere der schwarzen Rechtecke ist 100 mm breit und 200 mm hoch, alle anderen haben eine Größe von 100 mm × 100 mm.

Ziel dieser Konstruktion ist es, bei verschiedenen Abständen zur Lokalisierungshilfe von einem maximal großen Bereich im Kamerabild die reale Größe des Abgebildeten zu kennen und eine eindeutige Zuordnung treffen zu können. Daraus lassen sich Richtung und Abstand der Kamera zum Mittelpunkt des mittleren, großen schwarzen Rechtecks möglichst genau bestimmen. Beim Kameraöffnungswinkel der verwendeten Roboter ist dies etwa im Abstand von 60 cm bis 260 cm möglich, da hier jeweils mindestens drei Viertel der Bildbreite zur Größenbestimmung herangezogen werden können. Ab einem Abstand von 60 cm passen drei der schwarzen Blöcke ins Bild und bis zu einem Abstand von 260 cm nimmt die gesamte Lokalisierungshilfe mehr als 3/4 der Bildbreite ein.

Die Kopfsteuerung des Roboters ist stets bemüht, das mittlere, große schwarze Rechteck in der Bildmitte zu halten. Zunächst wird im Bild auf ausreichend dichten horizontalen Scan-Linien nach Schwarz-Weiß-Übergängen gesucht (kleine rote und blaue Punkte am Rande der schwarzen Blöcke in Abbildung 3.1 a).

Zwischen den Schwarz-Weiß-Übergängen werden die schwarzen Blöcke der Lokalisierungshilfe vermutet. Daher wird jeweils in der Mitte zwischen einem Weiß-Schwarz- und einem Schwarz-Weiß-Übergang auf einer senkrechten Linie nach dem oberen und unteren Ende des Schwarzblockes gesucht. Existieren diese Enden in einem zur Breite des Schwarzblockes passenden Abstand, so ist die Mitte zwischen oberem und unterem Ende des Schwarzblockes ein Kandidat für die Blockmitte.



$$\gamma = \pi - \alpha - \beta - \delta$$

Nach Sinussatz gilt:

$$\sin(\alpha)/a = \sin(\delta)/e$$

$$\sin(\beta)/b = \sin(\gamma)/e$$

Abbildung 3.2: Zusammenhänge zwischen den Winkeln bei der Positionsberechnung mit der entwickelten Lokalisierungshilfe

Dann wird mit linearer Regression eine Linie durch all diese Blockmittelpunkt-kandidaten gelegt (weiß in Abbildung 3.1 a). Auf dieser Linie lassen sich die subpixelgenauen Positionen der Übergänge zwischen Schwarz und Weiß (die Stellen mit dem Mittelwert der Helligkeiten rechts und links des Überganges) bestimmen und auf Konsistenz (gleichmäßige Abstände) prüfen.

Die Positionen eines in der Mitte und der beiden am weitesten außen befindlichen Übergänge werden in Winkel zur Kamera umgerechnet und den exakten Positionen auf der realen Lokalisierungshilfe zugeordnet.

3.2.2 Positionsberechnung

In Abbildung 3.2 sind die Längen a und b der gesehenen Teile der Lokalisierungshilfe zwischen den Schwarz-Weiß-Übergängen bekannt, ebenso wie die zugehörigen beobachteten Winkel α und β . Durch Anwendung des Sinussatzes in beiden Teildreiecken mit Gleichsetzung der gemeinsamen Strecke e erhält man zwei mögliche Lösungen für den Winkel δ :

$$\delta_1 = \arccos \left(\frac{|(2a + b) \sin(\beta) - b \sin(2\alpha + \beta)|}{\sqrt{2((a + b)(a + b - b \cos(2\alpha) - a \cos(2\beta)) + ab(\cos(2\alpha + 2\beta) - 1))}} \right)$$

$$\delta_2 = \arccos \left(\frac{-|(2a + b) \sin(\beta) - b \sin(2\alpha + \beta)|}{\sqrt{2((a + b)(a + b - b \cos(2\alpha) - a \cos(2\beta)) + ab(\cos(2\alpha + 2\beta) - 1))}} \right)$$

Nur eine der beiden potentiellen Lösungen führt zur korrekten Innenwinkelsumme in den betrachteten Dreiecken in Abbildung 3.2 und ist damit der gesuchte Winkel δ . Die Position (x, y) der Kamera relativ zum mittleren der benutzten drei Schwarz-Weiß-Übergänge ergibt sich dann unter erneuter Zuhilfenahme des Sinussatzes als:

$$\begin{aligned}
c &= \frac{a \sin(\pi - \alpha - \delta)}{\sin(\alpha)} \\
x &= -c \sin(\delta) \\
y &= -a + c \cos(\delta) \quad .
\end{aligned}$$

Da mit der Kameraposition (x, y) auch gleich die Blickrichtung zum mittleren der drei Schwarz-Weiß-Übergänge bekannt ist, und der Roboter bestimmen kann, mit welchem Bildpixel und welchen Kopfgelenkwinkeln er eben diesen Schwarz-Weiß-Übergang gesehen hat, erhält man daraus die gesuchte Position und Ausrichtung des Roboters relativ zur Lokalisierungshilfe.

3.2.3 Glättung

Die mit der entwickelten Lokalisierungshilfe aus den Einzelbildern gewonnenen Rohpositionen sind im Vergleich zu anderen Lokalisierungsansätzen recht genau (siehe Abschnitt 3.2.4). Es gibt keine nennenswerten Ausreißer, da es bei solchen aufgrund der Redundanz zu viele Unstimmigkeiten im aufgenommenen Bild gegeben hätte, die durch die Konsistenzprüfung (gleichmäßige Blockgrößen und -abstände) verworfen worden wären. Daher verbleiben zwei betrachtenswerte Fehlerarten.

Eine Fehlerart sind Messfehler der Kopfgelenkwinkel, die mit mechanischer Beanspruchung zunehmen und eine ungenaue Positions- und Richtungsbestimmung nach sich ziehen. Ein Laufen, das den Kopf stark erschüttert, erschwert dadurch die Positionierung, verschlechtert die Positionstreuung des Laufens und wird daraufhin schlechter bewertet. Da somit Parametersätze, die wegen schlechter Bewertung ohnehin nicht in die engere Wahl kommen, die Ursache dieser Fehler sind, wird gegen diese Fehlerart nichts weiter unternommen.

Die zweite Fehlerart ist ein leichtes Rauschen, das zum Beispiel durch Unschärfe (Fixfokuskamera), Bildrauschen und Erschütterungen hervorgerufen wird. Um diese Fehlerart zu eliminieren, wurden verschiedene Ansätze ausprobiert.

Das Legen einer quadratischen Kurve durch die letzten Messwerte mit Minimierung der Fehlerquadratsumme etwa liefert auf den ersten Blick geeignete Abschätzungen der aktuellen Position. Betrachtet man jedoch auch die darauf folgenden Messungen, so sieht man, dass die Glättungswirkung dieser Methode vernachlässigbar ist.

Auch ein Kalman-Filter⁶ (siehe [35]), der für den Umgang mit verrauschten Messwerten gedacht ist, liefert nur mäßige Ergebnisse. Während er mit gleichartigen Situationen (konstant gute Messwerte oder konstant hohes Rauschen) gut zurechtkommt, bereitet der Übergang dazwischen wie auch Richtungswechsel des Roboters Schwierigkeiten, da das selbst antrainierte Vertrauen in die Messwerte plötzlich nicht mehr stimmt.

⁶ein Filter, der die Varianzen und Kovarianzen der bisherigen Messwerte und seiner bisherigen Vorhersagen benutzt, um aus dem aktuellen Messwert eine rauschbefreite Vorhersage abzuleiten

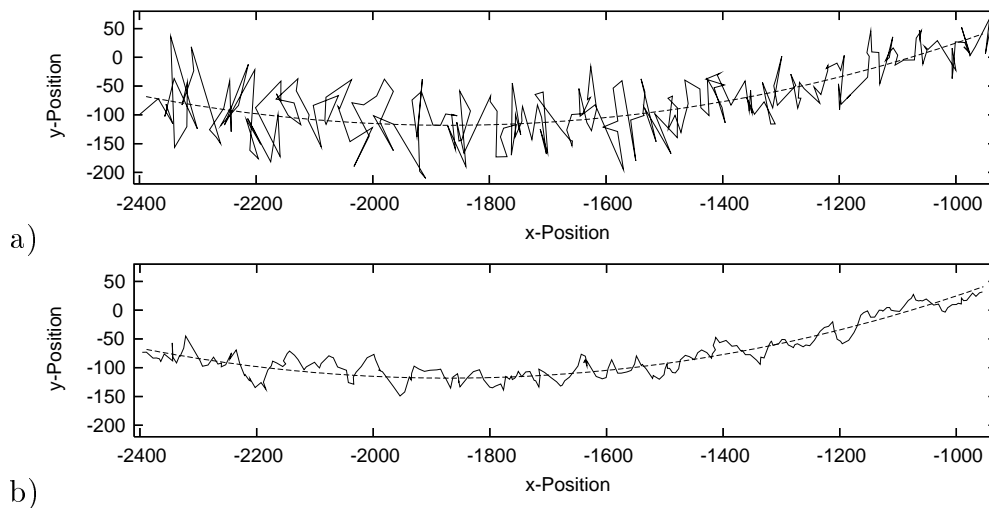


Abbildung 3.3: Bestimmung der Roboterposition mit der Lokalisierungshilfe während zehn Sekunden Vorwärtslaufens mit leichter Drehung im Vergleich zur Optimallinie: **a)** Rohdaten, **b)** das Gleiche mit PID-Glättung

Letztendlich liefert die Verwendung eines einfachen PID-Reglers die besten Resultate. Sei $m_i = (x_{i_m}, y_{i_m}, \varphi_{i_m})$ die zum Zeitpunkt i aus einem Bild gemessene Position und $p_i = (x_{i_p}, y_{i_p}, \varphi_{i_p})$ die für diesen Zeitpunkt modellierte Position, dann wird folgende Regelungsvorschrift zur Glättung verwendet:

$$p_i = p_{i-1} + \underbrace{0,15}_{P} \cdot (m_i - p_{i-1}) + \underbrace{0,25}_{I} \cdot \sum_{j=1}^i (m_j - p_{j-1}) \quad .$$

Die Wirkung dieses PID-Reglers ohne Differentialanteil ist in Abbildung 3.3 b zu sehen.

3.2.4 Positionsgenauigkeit

Die Kenntnis der genauen Roboterparameter wie Kameraöffnungswinkel und Gelenkwinkel erlaubt die Bestimmung der Roboterposition sehr genau, der systematische Fehler liegt in der Größenordnung der Messgenauigkeit bei manueller Messung, also unter einem Zentimeter. Interessanter sind daher die durch Fehlmessungen auftretenden Abweichungen.

Das Kamerabild eines Sony ERS-210 hat eine horizontale Auflösung von 176 Pixeln. Ein ERS-7 kann auch Bilder mit etwas mehr als doppelter Auflösung liefern, allerdings haben solche Bilder keine homogene Helligkeitsverteilung mehr, weil die Verdopplung auf Interpolation der Messwerte der drei Farbkanäle beruht. Deshalb werden in den Experimenten zu dieser Arbeit generell nur die niedriger aufgelösten Bilder verwendet.

Im benutzten Abstand von 60 cm bis 260 cm zur Lokalisierungshilfe ist bei geeigneter Blickrichtung von mindestens $3/4$ der Bildfläche (132 Pixel) die Größe des Gesehenen bekannt. Zur Bestimmung der Position wie in Abschnitt 3.2.2 sind zwei

Winkel nötig, wobei jeder davon etwa $3/8$ der Bildbreite (64 Pixel, Winkel etwa $0,38$) entspricht. Dank subpixelgenauer Positionsbestimmung (jeweils mittlerer Grauwert zwischen weißer und schwarzer Fläche) kann mit einem Messfehler zwischen zwei Übergängen von $1/2$ Pixel (Winkel etwa $0,003$) gerechnet werden. Damit ergeben sich im Abstand von 2600 mm folgende erwartete maximale Abweichungen von der korrekten Position:

$$\begin{aligned} \alpha = 0,38 + 0,003, \quad \beta = 0,38 - 0,003 &\quad \rightarrow \Delta y = 56,6 \text{ mm} \\ \alpha = 0,38 + 0,0015, \quad \beta = 0,38 + 0,0015 &\quad \rightarrow \Delta x = 11,3 \text{ mm} \end{aligned} .$$

Um diese theoretischen Überlegungen zu bestätigen, wurde die Position eines ERS-210 (wie in Abbildung 3.3 gezeigt) zehn Sekunden lang während des Vorwärtslaufens mit leichter Drehung vermessen. Die Wurzel der mittleren quadratischen Abweichung von der Idealposition betrug dabei ohne Glättung 15,9 mm in x-Richtung (Abstand zur Lokalisierungshilfe) und 38,3 mm in y-Richtung (parallel zur Lokalisierungshilfe).

Bei Verwendung der vorgestellten PID-Regelung steigt die durchschnittliche Abweichung in x-Richtung geringfügig auf 18,3 mm, weil der PID-geregelte Wert nur vergangene Messungen berücksichtigen kann und daher der realen Position des kontinuierlich vorwärts strebenden Roboters etwas verzögert folgt. Die Abweichung der y-Position verringert sich dagegen erheblich auf 14,6 mm, da dort vorrangig Rauschen eliminiert werden kann. Beim verwendeten PID-Regler wird die deutliche Glättung also damit erkauft, dass die modellierte Position der gemessenen bei höheren Geschwindigkeiten nur verzögert folgt, jedoch durch den Integralparameter längst nicht so stark wie etwa bei einfacher Mittelbildung.

3.3 Omnidirektionale Optimierung

Nachdem ein Verfahren zur genauen Positionsbestimmung eines Roboters zur Verfügung steht, ist es möglich, gute Parameter für das in Abschnitt 2.4 vorgestellte Laufmodell objektiv zu ermitteln.

In diesem Abschnitt wird zunächst nach einem einzigen Parametersatz gesucht, der ein gutes omnidirektionales Laufen erlaubt, um möglichst wenig zwischen verschiedenen, für bestimmte Bewegungsanforderungen optimierten Parametersätzen umschalten zu müssen. Ein solcher omnidirektionaler Parametersatz ist dann eine brauchbare Standardlösung, die zudem den spezialisierten Parametersätzen erwartungsgemäß ähnlicher ist als ein auf etwas anderes spezialisierter Parametersatz. Daher sollte das vom Laufmodell aus Abschnitt 2.4 verwendete Interpolieren zwischen einem omnidirektionalen Parametersatz und beispielsweise einem Spezialisten für schnelles Drehen besser aussehen als die Interpolation zwischen zwei Spezialisten.

3.3.1 Der Parcours

Der Roboter muss möglichst schnell und präzise einen fest vorgegebenen Parcours ablaufen, der verschiedene Lauf- und Drehrichtungen beinhaltet, da die spätere Ansteuerung durch ein Fußballspielverhalten vergleichbar aussehen wird. Durch die

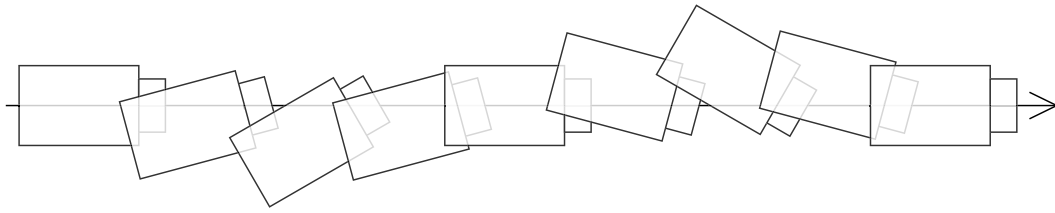


Abbildung 3.4: Der Parcours zur Untersuchung omnidirektionalen Laufens gibt Sollposition und -ausrichtung für einen Roboter vor.

Benutzung eines solchen Parcours unterscheidet sich dieser Ansatz von anderen Methoden, bei denen in der Regel nur das Laufen in jeweils eine konstante Bewegungsrichtung optimiert wird. Durch die Wahl eines geeigneten Parcours kann ein Parametersatz darauf getestet werden, wie gut omnidirektionales Laufen mit diesem Parametersatz möglich ist.

Ein Parcours gibt zu jeder aktuellen Position und zukünftigen Zeit eine Sollposition und -richtung vor. Da man die Laufgeschwindigkeit auf dem Parcours nicht vorgeben, sondern gerade herausfinden möchte, hängen die zukünftigen Sollpositionen von der jeweils letzten bekannten Istposition ab. Die Sollposition zum aktuellen Zeitpunkt hat die gleiche x-Koordinate wie die Istposition und dient somit als Fortschrittsanzeiger auf dem Parcours in Hauptlaufrichtung. Soll-x-Koordinaten in der Zukunft ergeben sich durch Annahme einer Geschwindigkeit von 300 mm/s (ERS-210) beziehungsweise 400 mm/s (ERS-7), die geringfügig über der zu erwartenden Maximalgeschwindigkeit der Roboter auf dem Parcours liegt. Zu gegebener x-Koordinate liefert ein Parcours zugehörige Soll-y-Koordinate und Sollandrehwinkel.

Der hier verwendete Parcours ist zweigeteilt. Während der Vorwärtsbewegung des Roboters wird sein Drehwinkel variiert (siehe Abbildung 3.4), und zwar sinusförmig mit einer Amplitude von $\pi/4$ und einer Periode von 900 Millimetern in Laufrichtung. Die Rückwärtsbewegung hingegen erfolgt gerade. Der vorwärts gerichtete Teil des Parcours wird dabei doppelt so stark gewichtet, weil das der fürs Fußballspielen wichtigere Teil ist. Rückwärtslaufen muss zwar auch funktionieren, aber nicht unbedingt mit ständig wechselnder Richtung.

Ein Roboter kann einem solchen Parcours nur dann gut folgen, wenn das reale Laufen einigermaßen der gewünschten Ansteuerung entspricht. Ist die Abweichung zu stark, so führt der Versuch, dem Parcours zu folgen oder zu ihm zurückzukehren, zum Teil sogar zu noch stärkerer Abweichung. Dies ist zum Beispiel der Fall, wenn das Laufmodell ohne die anatomiebedingte Korrektur aus Abschnitt 2.2.3 verwendet wird. Mit der Korrektur kann der Roboter dem vorgegebenen Parcours erheblich besser folgen.

Um einem Parcours zu folgen, bedarf es auf der anderen Seite einer Ansteuerung, die die Abweichung zwischen Soll- und Istposition minimiert und gleichzeitig dafür sorgt, dass der Parcours möglichst schnell durchschritten wird. Eine solche Ansteuerung ist der von einem praktischen Anwendungsfall wie Fußballspielen pro-

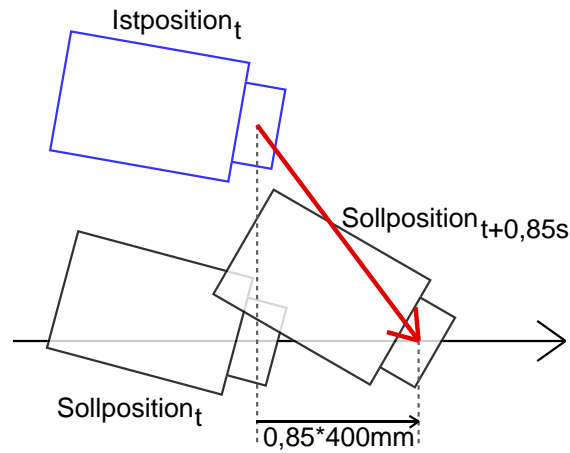


Abbildung 3.5: Regelung der Ansteuerung des Laufens, um den Roboter dem Parcours folgen zu lassen: Die angesteuerte Bewegung (rot) resultiert aus dem Abstand zur Sollposition 0,85 Sekunden später.

duzierten Ansteuerung wesentlich ähnlicher als die sonst üblicherweise vermessenen konstanten Bewegungsanforderungen.

Das Laufen muss in der Lage sein, mit unterschiedlicher, aber jeweils für sinnvoll erachteter Ansteuerung zurechtzukommen. Daher kann man selbst von einem noch nicht genau bekannten Laufen verlangen, einem Parcours inklusive zugehöriger Ansteuerung folgen zu können, wenn dieser Parcours wie der verwendete keine unmöglichen Bewegungen (etwa ständige Richtungssprünge) fordert.

Daher wird zur Ansteuerung folgende Regelung verwendet: Es wird stets die Zielposition 0,85 Sekunden in der Zukunft angesteuert, also angestrebt, den Unterschied zwischen Ist und Soll innerhalb von 0,85 Sekunden zu korrigieren (siehe Abbildung 3.5). Während der Drehwinkel sich auch etwas schneller korrigieren ließe, führt der Versuch schnellerer Positionskorrektur dazu, dass der Roboter nicht mehr vorwärts kommt, weil er nur noch mit Positionskorrekturen beschäftigt ist. Die verwendete Regelung ist somit ein guter Kompromiss zwischen den gegenläufigen Zielen schnell vorwärts zu kommen und dicht an der Optimallinie zu bleiben.

Diese Regelung hat sich als geeignet erwiesen, einen Roboter auch mit bisher nicht vermessenem Laufen zügig dem vorgegebenen Parcours folgen zu lassen. Dabei hat die Regelung selbst einen deutlichen Einfluss auf die Bewertung eines konkreten Laufens. Das ist akzeptabel, weil sie als integraler Bestandteil der Aufgabe, dem Parcours zu folgen, betrachtet werden kann. Die vorgestellte Regelung begünstigt schnelles und exaktes Laufen, kommt mit den bisher untersuchten Laufparametern gut zurecht, stellt aber nicht sicher, dass Laufparameter, die von den getesteten erheblich abweichen, auch fair bewertet werden.

Der hier vorgestellte Parcours inklusive seiner Ansteuerung wird nicht nur in diesem Abschnitt verwendet, sondern zusätzlich im Abschnitt 3.6 noch einmal. Er dient

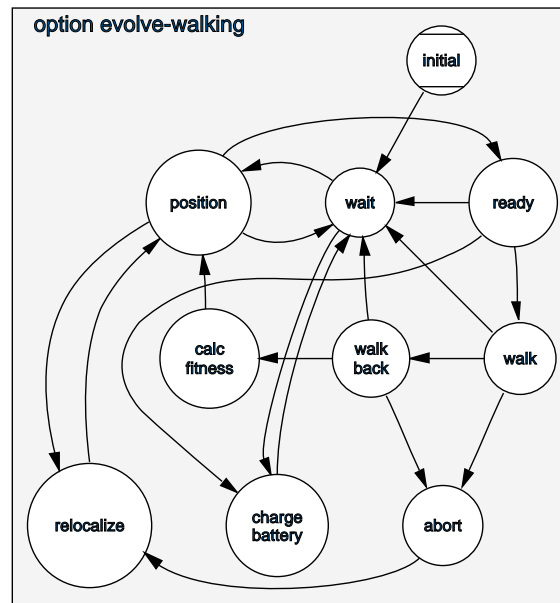


Abbildung 3.6: Xabsl-Zustandsautomat des Verhaltens zur Optimierung omnidirektionalen Laufens: Die eigentliche Arbeit wie die Evolution der Parameter oder die Bestimmung der Fitness erfolgt innerhalb einzelner Zustände (Kreise).

somit als Benchmark, mit dem gezeigt wird, dass die Optimierungsschritte in Abschnitt 3.3 bis 3.5 das Laufen verbessern.

Damit die Optimierung automatisch ablaufen kann, muss der Roboter selbstständig mit einer Reihe von Spezialfällen zurechtkommen, die sonst von einem menschlichen Trainer erkannt und behandelt würden. Dazu zählen:

- Ladestand des Akkus unzureichend: Der Roboter muss das Training einstellen, den letzten Stand abspeichern und auf sich aufmerksam machen.
- Timeout: Der aktuelle Parametersatz ist so schlecht, dass der Roboter es innerhalb einer festgesetzten Zeit nicht schafft, zum Ausgangspunkt zurückzukehren. Also muss er diesen Versuch abbrechen und mit bekannten Standardparametern zum Ausgangspunkt zurückkehren.
- Unerwartete Situationen: Der Roboter muss aufstehen, wenn er umgestürzt ist und er muss mit einem Suchverhalten beginnen, wenn er sich verirrt hat beziehungsweise die Lokalisierungshilfe nicht mehr sieht.

Diese Verhaltensweisen werden auf die gleiche Weise spezifiziert wie das vom GermanTeam entwickelte Fußballspielverhalten, nämlich mit einem hierarchischen Zustandsautomaten, der im XML-Dialekt Xabsl⁷ [26] formalisiert ist. Der Zustandsautomat des verwendeten Verhaltens ist in Abbildung 3.6 zu sehen.

⁷eXtensible Agent Behavior Specification Language

3.3.2 Fitnessfunktion und Gütemaß

Ziel dieser Arbeit ist nach wie vor ein Verfahren, dessen Verwendung zur Bestimmung im RoboCup-Umfeld tauglicher Laufparameter für einen Sony Aibo führt. In das von einer Fitnessfunktion bestimmte Gütemaß für einen Laufparametersatz gehen daher neben der maximalen Laufgeschwindigkeit auch die Stärke der Erschütterung des Roboters (und somit die Beeinträchtigung von Laufstabilität und Bildverarbeitung) sowie explizit und implizit die Richtungstreue des Laufens ein. Letzteres wird dadurch erreicht, dass stets die Sollposition auf dem vorgegebenen Parcours angesteuert wird (siehe Abschnitt 3.3.1), was entsprechend länger dauert, wenn das reale Laufen von dieser Ansteuerung abweicht.

Die resultierende Güte oder Fitness kann als bewertete Geschwindigkeit interpretiert werden. Sie gibt die auf dem Parcours erreichte Durchschnittsgeschwindigkeit korrigiert um unerwünschte Positionsabweichungen und Erschütterungen an. Dazu werden folgende Messgrößen verwendet:

- \dot{x} : durchschnittliche Laufgeschwindigkeit in x-Richtung (entlang des Parcours auf die Lokalisierungshilfe zu beziehungsweise von ihr weg) in mm/s
- Δy : gemittelter Betrag der Abweichung der y-Position, also durchschnittlicher Abstand zur Optimallinie
- $\Delta\varphi$: gemittelter Betrag der Abweichung der Roboterrichtung, also durchschnittlicher Abstand zur Optimalrichtung
- \ddot{z} : gemittelter Betrag der Beschleunigung in z-Richtung, diese ist unbeabsichtigt und als unangenehm hartes Stampfen zu erkennen
- p_{blind} : Anteil der Bilder an der Gesamtbildzahl, in denen nicht eindeutig die Lokalisierungshilfe erkannt werden konnte, die verwendeten Laufparameter führten bei hohem p_{blind} also offenbar zu stärkerer Abweichung der Realität von der Ansteuerung oder zu stärkerer Erschütterung als akzeptabel

Weitere mögliche Größen wurden bewusst weggelassen, weil sie keinen relevanten Einfluss auf die Güte des Laufens haben, der nicht auch schon in den benutzten Größen enthalten ist. Dazu zählen Beschleunigungen in andere Richtungen, etwa Gewichtsverlagerung von rechts nach links sowie die Differenz aus angesteuerten und real gemessenen Gelenkwinkeln, weil für höchste Leistung etwas mehr angesteuert werden muss, als tatsächlich erreicht werden kann.

Die fünf verwendeten Messgrößen müssen so gewichtet werden, dass sie in einem geeigneten Verhältnis zueinander stehen. Mit einem zum Drehen ungeeigneten Parametersatz könnte ein Roboter zum Beispiel die Drehrichtungsvorgabe weitgehend ignorierend den Parcours sehr schnell absolvieren. Dennoch darf er dafür nicht mit hoher Fitness belohnt werden, weil es hier ja gerade nicht das Ziel ist, schnellstmögliches Geradeauslaufen zu fördern. Also muss eine Abweichung vom vorgegebenen Drehwinkel so stark bestraft werden, dass weniger starkes Drehen als gefordert zugunsten höherer Geschwindigkeit nicht zu besserer Fitness führt.

| Fitnessberechnung | \dot{x} | Δy | $\Delta\varphi$ | \ddot{z} | p_{blind} | Fitness |
|-------------------|-----------|------------|-----------------|-------------------|--------------------|---------|
| guter Lauf | 186,6 | 26,5 | 0,131 | $1,22 \cdot 10^6$ | 0,049 | |
| Bewertung | 186,6 | -4,4 | -4,3 | -7,2 | -2,0 | = 168,7 |
| schlechterer Lauf | 159,0 | 37,9 | 0,154 | $1,33 \cdot 10^6$ | 0,128 | |
| Bewertung | 159,0 | -6,3 | -5,1 | -8,3 | -5,1 | = 134,2 |

Tabelle 3.1: Die geeignete Wichtung der Komponenten der Fitnessfunktion erlaubt die Unterscheidung zwischen guten und schlechten Parametersätzen.

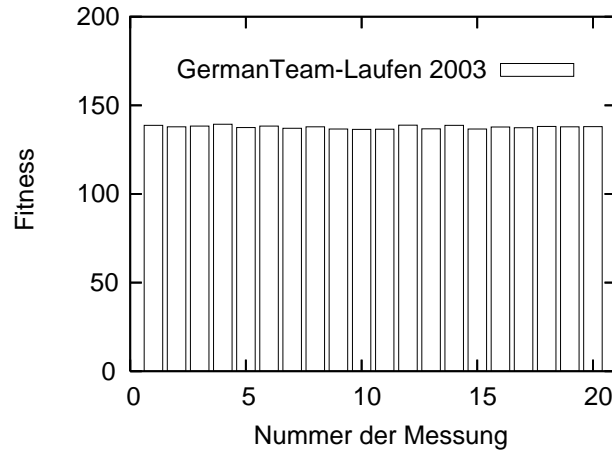


Abbildung 3.7: ERS-210: Die Ergebnisse von 20 Messungen der Fitness des 2003 vom GermanTeam verwendeten Laufens liegen dicht beieinander. Daher wird künftig jeder Parametersatz nur einmal bewertet.

Aus diesen Überlegungen heraus wurde die Fitness F eines Parametersatzes P wie folgt definiert, die zugehörige Bewertung zweier Beispielläufe ist Tabelle 3.1 zu entnehmen:

$$F(P) = \dot{x} - \Delta y/6 - 33\Delta\varphi - (10^{-5}\ddot{z} - 5) - 40p_{\text{blind}}$$

Mit dieser Fitnessfunktion wurde zunächst mehrfach das vom GermanTeam im RoboCup 2003 verwendete Laufen bewertet. Die ermittelte Fitness lag dabei zwischen 136,4 und 139,3, im Mittel bei 137,7 mit einer Standardabweichung von $\sigma = 0,84$ (siehe Abbildung 3.7). Aufgrund der geringen Schwankungen wird im Folgenden jeder Parametersatz nur einmal vermessen.

3.3.3 Optimierungsvoraussetzungen

Durch die vorgestellten Methoden steht erstmals ein Verfahren zur Verfügung, mit dem sich ein für Omnidirektionalität konzipiertes Laufen objektiv bewerten lässt. Bevor man mit der eigentlichen Optimierung beginnt, lohnt es sich zu untersuchen (und jetzt auch überprüfen zu können), durch welche Änderungen Leistungsstei-

gerungen zu erwarten sind. Anschließend ist dafür zu sorgen, dass das verwendete Optimierungsverfahren diese Änderungen auch durchführen oder ausnutzen kann.

Welche Schrittlänge zu maximaler Laufgeschwindigkeit führt, hängt sowohl vom verwendeten Parametersatz als auch von der Laufrichtung ab. Deshalb erlaubt es das Laufmodell aus Abschnitt 2.4, für verschiedene Richtungen auch verschiedene Maximalgeschwindigkeiten und damit -schrittlängen anzugeben. Für einen noch unvermessenen Parametersatz sind diese aber noch nicht bekannt.

Die Beschränkung der Schrittlänge führt gegenüber maximaler Ausreizung der Beinlänge zu schnellerem und stabilerem Laufen. Deshalb wird hier zunächst wie vor der Einführung des Laufmodells aus Abschnitt 2.4 eine laufrichtungsunabhängige Schrittlängenbeschränkung mit zwei Parametern verwendet: maximale Schrittlänge in Vorwärts- (x-) und Seitwärts- (y-) Richtung für alle Beine. Bisher wurde die Schrittlänge dazwischen durch die Ellipse

$$\sqrt{(x/x_{max})^2 + (y/y_{max})^2} \leq 1$$

beschränkt. Das Zulassen größerer diagonalen Schritte durch das weniger restriktive

$$\sqrt[4]{(x/x_{max})^4 + (y/y_{max})^4} \leq 1$$

führt (im Gegensatz zur Erhöhung von x- und y-Schrittlänge) zu drei Prozent höherer Fitness (siehe Abbildung 3.8) und wird daher hier verwendet, solange der untersuchte Parametersatz noch nicht vermessen ist. Eine weitere Lockerung der Schrittlängenbeschränkung hat sich als nicht geschwindigkeitssteigernd erwiesen.

Die maximale Schrittlänge ist durch die Anatomie der Roboter beschränkt und weitgehend ausgereizt, was zum Beispiel an sich berührenden Knien der Vorder- und Hinterbeine zu sehen ist.

Ein aussichtsreicher Kandidat für Optimierungen ist dagegen die Schrittfrequenz. So hat sich etwa die vom GermanTeam verwendete Zeit für einen Vollschrift von 1,02 Sekunden im Jahr 2000 auf 0,64 Sekunden zur Weltmeisterschaft 2003 verkürzt. Eine weitere Erhöhung der Schrittfrequenz ohne begleitende Maßnahmen führt bei einem ERS-210 aber nicht zu weiterer Erhöhung der Laufgeschwindigkeit.

Das liegt augenscheinlich daran, dass der Roboter die Füße nicht mehr rechtzeitig vom Boden anzuheben in der Lage ist und anfängt, wie auf Eis zu rutschen anstatt von der Stelle zu kommen. Dafür ist neben der Anatomie des Roboters und der Trägheit seiner Teile auch die verwendete Fußtrajektorie verantwortlich. Die unter anderem im ersten Laufmodell dafür benutzten Halbellipsen (siehe Abbildung 2.5 b) führen zwar zu einem geschmeidig aussehenden Laufen, aber auch dazu, dass der Roboter die Füße, insbesondere bei Schritten, die deutlich länger als hoch sind, langsamer hebt und senkt, als es günstig erscheint.

Daher ist es zumindest einfacher, gute Parametersätze mit Bewegung der Füße in Rechtecken beziehungsweise Parallelogrammen zu finden, weshalb das aktuelle Laufmodell (siehe Abschnitt 2.4) ebenfalls Parallelogramme benutzt. Nichtsdestotrotz ist die Frage nach der optimalen Fußtrajektorie noch nicht gelöst.

| (P, I, D)-Werte | Schultergelenk θ_1 | Schultergelenk θ_2 | Kniegelenk θ_3 |
|------------------|---------------------------|---------------------------|-----------------------|
| ERS-210-Standard | (22, 4, 8) | (20, 4, 6) | (35, 4, 5) |
| ERS-7-Standard | (28, 8, 1) | (20, 4, 1) | (28, 8, 1) |
| P erhöht | (38, 8, 1) | (30, 4, 1) | (40, 8, 1) |

Tabelle 3.2: Verwendete Werte für die PID-Regler der Gelenkmotoren

Ein weiterer Grund für die Trägheit der Beine bei höheren Geschwindigkeiten liegt in der bisher ausschließlichen Verwendung der Standardwerte für die motoreigenen PID-Regler. Diese Standardwerte stellen an sich einen guten Kompromiss zwischen Genauigkeit und Geschwindigkeit dar und vermeiden ein Überschwingen über den Sollwert.

Eine Erhöhung dieser Werte, insbesondere des interessanten Proportionalanteils, führt in Spezialfällen (Bein unbelastet, große Sprünge in der Ansteuerung) leicht zum Überschwingen oder Zittern, im Falle schnellen Laufens (schnelle Änderung des Sollwertes ohne große Sprünge, Beine belastet) aber zu spürbarer Verbesserung des Laufens. Diese ist sogar unerwartet deutlich: Durch Erhöhung des P-Wertes ohne Änderung anderer Parameter steigt die Fitness um etwa zehn Prozent (siehe Abbildung 3.8).

Dennoch werden die PID-Werte nicht zur Evolution freigegeben, sondern nur manuell so weit erhöht, wie es Stabilität und Zittern zulassen und Verbesserungen messbar sind. Das schützt zum einen die Motoren vor Überlastung und vermeidet zum anderen die Notwendigkeit, ein geeignetes und vom Roboter bestimmbares Maß für Stabilität und Zittrigkeit zu finden. Tabelle 3.2 enthält die verschiedenen verwendeten PID-Werte.

Vergleicht man das Laufen eines ERS-210 und eines ERS-7 mit den gleichen ursprünglich für einen ERS-210 entwickelten Laufparametern und den jeweiligen Standard-PID-Werten, dann fällt auf, dass ein ERS-7 bei einzelnen extremen Laufbewegungen, die auf einem ERS-210 noch funktionieren, die Füße nicht mehr schnell genug vom Boden hebt. Die Standard-PID-Werte eines ERS-7 führen aus dem gleichen Grund zu Bewegungen, die vom Beobachter als weicher als beim ERS-210 wahrgenommen werden.

Ein PID-Regler soll die Abweichung zwischen Soll- und Ist-Gelenkwinkel korrigieren. Der standardmäßig niedrige Proportionalanteil P führt dazu, dass gegen geringe Abweichungen nur wenig unternommen wird. Der Versuch, sehr langsam zu laufen, führt also eher zum Treten auf der Stelle. Große Abweichungen werden zwar stärker korrigiert, aber aufgrund der zunehmenden Gegenwehr der Trägheit nicht schnell genug.

Die Erhöhung des Proportionalanteils der PID-Werte verbessert daher hauptsächlich besonders langsames und besonders schnelles Laufen. Negative Auswirkungen sind nur bei ungeeigneter Parameterwahl, etwa bei sehr schnellen Bewegungen bei sehr geringer Geschwindigkeit (Tippeln), in Form von unruhigerem Lauf (erhöhtem Zittern) zu bemerken.

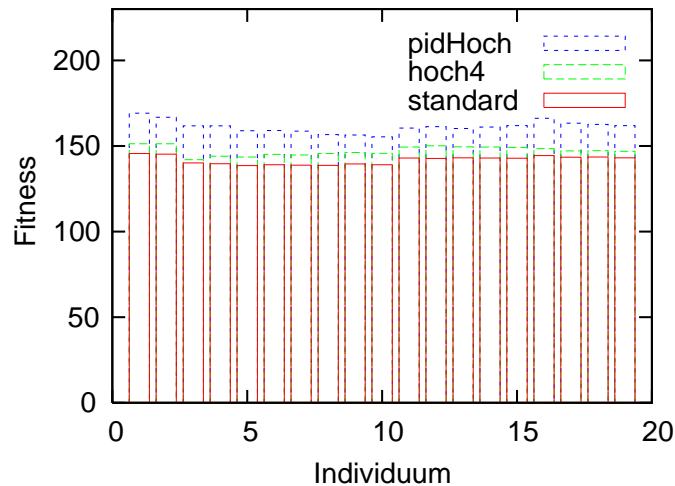


Abbildung 3.8: ERS-210: Die gemessene Fitness des Laufens steigt durch weniger restriktive Schrittlängenbeschränkung („hoch4“) um drei Prozent und durch Erhöhung des Proportionalparameters der motoreigenen PID-Regler um etwa zehn Prozent. Zu sehen sind hier die Fitnesswerte verschiedener Laufparametersätze mit Standard-PID-Werten und mit erhöhtem P-Wert.

3.3.4 Lernmethode

Aufgrund der Erfahrungen bei der manuellen Feineinstellung von Laufparametern ist davon auszugehen, dass die alleinige Verwendung von Verfahren wie Gradientenabstieg wegen der oftmals unerwarteten Zusammenhänge zwischen Parameteränderungen und Ergebnisänderungen ungeeignet ist. Eine Möglichkeit, zu besseren Parameterkombinationen zu kommen, ist daher die weitgehend blinde Mutation und Kreuzung geeigneter Parametersätze. Auch andere Autoren sind der Meinung, dass der Einsatz von Evolutionsverfahren ein geeigneter Weg ist, derartige Probleme zu lösen [37].

In [5] und besonders in [21] wurde gezeigt, dass man mit Reinforcement-Learning die Untersuchung des großen Suchraumes trotz zufälliger Mutation in eine geeignete Richtung lenken kann, zumindest wenn man ein einfaches Optimierungskriterium wie maximale Geschwindigkeit einer gleich bleibenden Bewegung verwendet.

Die auch hier zu erwartende Beschleunigung des Optimierungsprozesses durch Reinforcement-Learning erkauft man sich mit zusätzlichen Annahmen zum Einfluss einzelner Parameter auf die Gesamtfitness. Wie beim Gradientenabstieg führt die Annahme, dass sich ein Optimum in Richtung der lokal günstigsten Änderung der Fitness befindet, dazu, dass man lokale Optima schlechter verlassen kann und neben dem Reinforcement-Weg liegende Parameterkombinationen unbetrachtet bleiben. Deshalb wird in dieser Arbeit auf den mit der Verwendung von Reinforcement-Learning verbundenen zusätzlichen Aufwand verzichtet.

Eine weitere Möglichkeit zur Beschleunigung der Suche liefert [6]. Dort wird ein neuronales Netz verwendet, das nur noch solche Parametersätze zum Test auf realen Robotern vorschlägt, deren Fitness bisher nur schlecht abgeschätzt werden kann. Während dieses Verfahren zur groben Untersuchung des gesamten Suchraumes besser geeignet ist, erschwert es die Suche nach Optima in der Nähe bekannter Parametersätze.

Da seit geraumer Zeit weitgehend alle als gut bekannten Laufbewegungen relativ ähnlich aussehen (bezüglich solcher Parameter wie Schulterhöhe, Körperneigung und Schrittfrequenz) und trotzdem kontinuierlich schnellere gefunden werden, sind Verbesserungen eher in deren Nähe als an anderen Enden des Suchraumes zu erwarten.

Deshalb wird in den in dieser Arbeit beschriebenen Experimenten ein einfaches, weitgehend klassisches Evolutionsverfahren verwendet (siehe dazu auch [22]), um bessere Parameter für omnidirektionales Laufen zu erhalten.

3.3.5 Kreuzung, Mutation und Population

Für die Evolution wird eine Population von Parametersätzen benutzt. Jeder Parametersatz entspricht einem Individuum, jeder einzelne Parameter einem Gen. Da keine hier nutzbaren Zusammenhänge zwischen den Parametern eines Parametersatzes bekannt sind, werden alle Parameter als gleichwertige Gene auf einem einzigen Chromosom angesehen.

Die für die Evolution notwendige Nutzung realer Roboter ist recht zeitaufwändig. Darum wurde die vorherige Wahl von Evolutionsparametern wie Populationsgröße und Mutationsrate nach Augenmaß getroffen, statt diese Evolutionsparameter selbst schon zu evolvieren. Für zügigen Fortschritt bei blinder Mutation und nur relativ ungenau messbaren Fitnessunterschieden genügt eine kleine Population von zum Beispiel zehn Individuen. Die Ausgangspopulation besteht aus verschiedenen Mutationen eines vorgegebenen Parametersatzes.

Pro Generation werden von den zehn Parametersätzen einer Population die fünf mit der schlechtesten Fitness selektiert und durch Mutationen und Kreuzungen der besseren Hälfte ersetzt, quasi als Kompromiss zwischen schnellem Fortschritt und Vermeidung von Ausreißern und Sackgassen. Die Nachkommen entstehen dabei in 40 Prozent der Fälle durch Mutation und in 60 Prozent durch Kreuzung.

Bei Mutation werden einzelne Gene mit einer Wahrscheinlichkeit von 30 Prozent mutiert, und zwar gleichverteilt um bis zu plus/minus sechs Prozent des zulässigen Wertebereichs um den ehemaligen Wert. Bei Kreuzung wird in Anlehnung an [32] für jedes Gen zufällig zwischen den Werten der Elternteile interpoliert beziehungsweise in Richtung des Besseren auch extrapoliert.

Die Verwendung dieser Werte für die Evolutionsparameter führt dazu, dass sich einzelne Individuen mess- und sichtbar voneinander unterscheiden, aber in der Regel keine Sprünge in einem Schritt von guten zu völlig unbrauchbaren Individuen auftreten. Außerdem lässt sich so ohne viel Vorwissen über den Zusammenhang der

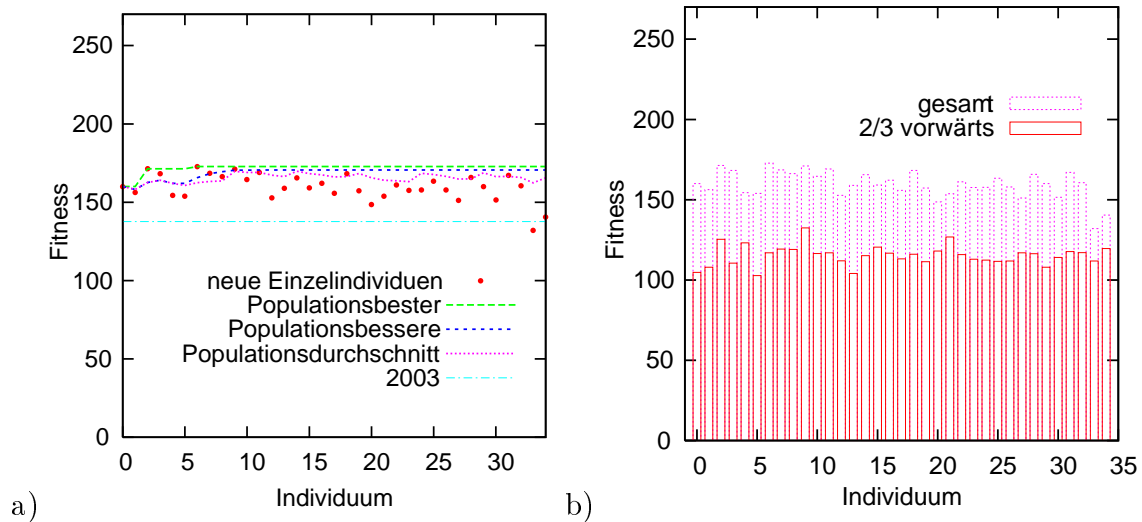


Abbildung 3.9: ERS-210: **a)** Die Fitness bei der Evolution omnidirektionaler Parametersätze steigt zwar anfangs, dann aber lange Zeit nicht weiter. **b)** Die Aufteilung der Fitness der Parametersätze in ihre Vorwärtskomponente (Wichtung 2/3) und Rückwärtskomponente offenbart, dass besonders gute Parametersätze fürs Vorwärtslaufen meist besonders schlecht fürs Rückwärtslaufen geeignet sind.

einzelnen Parameter in relativ kurzer Zeit und ohne viele Fehlversuche ein sinnvoller Teil des Suchraumes abdecken.

3.3.6 Ergebnisse der Evolution

Die ersten Testläufe erfolgten mit einem ERS-210 und auf Grundlage des im Jahr 2003 vom GermanTeam zur RoboCup-Weltmeisterschaft benutzten Parametersatzes. Bereits nach wenigen Mutationsschritten, also innerhalb der ersten Generationen, entstand dabei ein deutlich besserer Parametersatz mit einer Fitness von 145,7 statt der bisherigen 137,7 (siehe Abbildung 3.7).

Aufgrund seiner höheren Fitness wird dieser erste mit der vorgestellten Lernmethode gewonnene Parametersatz für alle weiteren Evolutionsversuche benutzt, sowohl für die Roboter vom Typ ERS-210 als auch später für die ERS-7. Das Aibo-TeamHumboldt benutzte diesen Parametersatz sogar unverändert auf einem ERS-7 bei der German Open⁸ 2004 und gewann.

Abbildung 3.9 a zeigt die Ergebnisse einer Evolution omnidirektionaler Laufparameter. Dabei werden schnell Parametersätze mit höherer Fitness gefunden, allerdings folgt dann lange Zeit keine weitere Verbesserung. Eine nähere Untersuchung ergab, wie in Abbildung 3.9 b gezeigt, dass besonders gute Parametersätze für den Vorwärtsteil des Parcours besonders schlechte Werte für den Rückwärtsteil liefern.

Da das benutzte Laufmodell jedoch ohnehin verschiedene Parametersätze verwenden kann, wird im Folgenden darauf verzichtet, einen einzigen Parametersatz für

⁸offene deutsche Meisterschaft des RoboCup, <http://www.robocup-german-open.de>

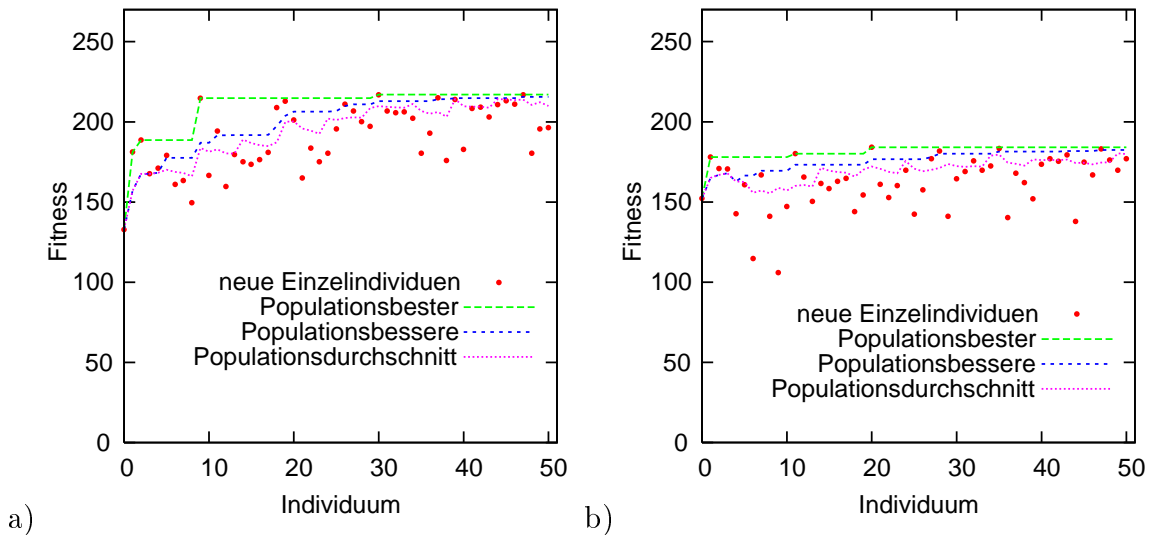


Abbildung 3.10: ERS-210: Evolution von Laufparametersätzen mit zwei getrennten Populationen für Vorwärts- und Rückwärtsteil des Parcours, **a)** Entwicklung der Fitness in der Population mit Vorwärtslaufparametern, **b)** Entwicklung der Fitness in der Population mit Rückwärtslaufparametern

omnidirektionales Laufen optimieren zu wollen. Stattdessen wird ein Parametersatz fürs Vorwärtslaufen und ein anderer fürs Rückwärtslaufen optimiert.

Das erhoffte Ziel, einen Parametersatz zu erhalten, der die meisten Bewegungswünsche abdeckt, ist damit nicht aufgegeben. Ein Parametersatz, der auf dem Vorwärtsteil des Parcours zu hoher Fitness führt, taugt in der Regel sehr wohl auch zum Rückwärtslaufen, nur eben nicht mit maximaler Schrittlänge beziehungsweise maximaler Geschwindigkeit. Statt einer Population von Parametersätzen werden daher künftig zwei Populationen verwendet, eine mit Parametersätzen fürs Ablaufen des vorwärts gerichteten Teils des Parcours und eine für den rückwärts gerichteten Teil.

Abbildung 3.10 zeigt die Entwicklung der Fitness der Laufparametersätze bei Verwendung zweier getrennter Populationen. Auf diese Weise erhält man Parametersätze fürs Vorwärts- und Parametersätze fürs Rückwärtslaufen, die jeweils eine deutlich höhere Fitness als die Parametersätze in Abbildung 3.9 haben, die beides können müssen.

Die bisherigen Versuche erfolgten auf einem Roboter des Typs ERS-210. Da inzwischen ein Nachfolgemodell (ERS-7) zur Verfügung steht, werden die weiteren Versuche mit einem ERS-7 durchgeführt. Erfreulicherweise ist dabei zu beobachten, dass ein guter Parametersatz für omnidirektionales Laufen eines ERS-210 auch für einen ERS-7 zu guten Resultaten führt.

Darüber hinaus hat ein ERS-7 ein höheres Potential: Durch modifizierte Anatomie wie geringfügig erhöhte Beinlänge und verstärkte Motoren sind deutlich höhere Maximalgeschwindigkeiten möglich, und zwar rund 400 mm/s [32] im Vergleich zu 300 mm/s beim ERS-210.

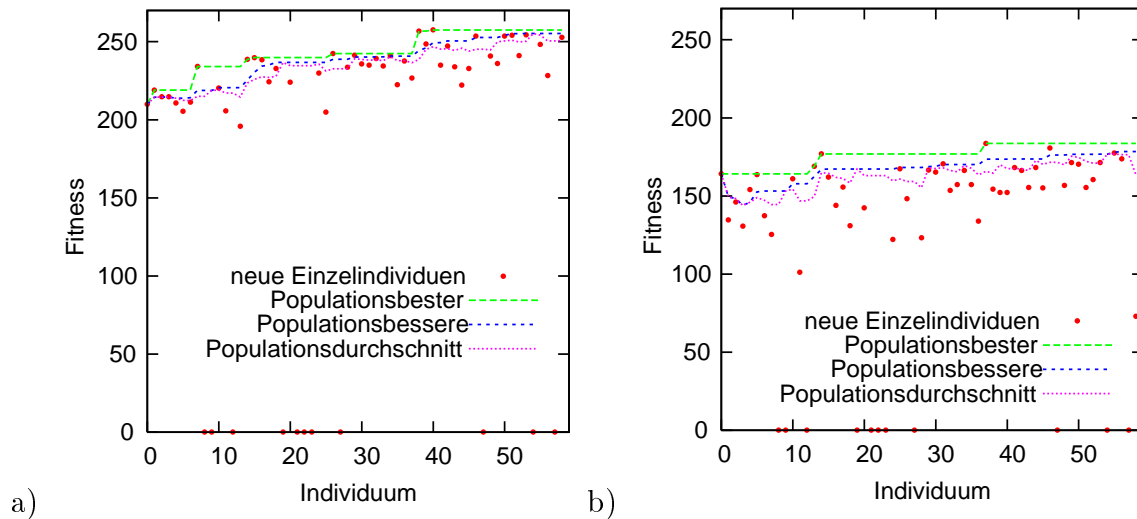


Abbildung 3.11: ERS-7: Evolution von Laufparametersätzen mit zwei getrennten Populationen für Vorwärts- und Rückwärtsteil des Parcours, **a)** Entwicklung der Fitness in der Population mit Vorwärtslaufparametern, **b)** Entwicklung der Fitness in der Population mit Rückwärtslaufparametern

In Abbildung 3.11 ist die Entwicklung der Fitness von Laufparametersätzen für einen ERS-7 zu sehen. Als Ausgangspunkt dient wieder der gleiche, ursprünglich für einen ERS-210 entwickelte Parametersatz wie in den beiden vorigen Evolutionsversuchen. Auch für den ERS-7 werden getrennte Populationen mit Parametern fürs Vorwärts- und Rückwärtslaufen verwendet.

Hier sind die Unterschiede zwischen Vorwärts- und Rückwärtslaufen noch stärker ausgeprägt als beim ERS-210, da veränderte anatomische Details (beispielsweise abgerundete vordere Unterschenkel) vorrangig das Vorwärtslaufen verbessern. Dadurch haben Parametersätze für den ERS-7 beim Vorwärtslaufen eine deutliche höhere Fitness als beim Rückwärtslaufen oder auf einem ERS-210.

3.4 Vermessung verschiedener Parametersätze

Das im Abschnitt 2.4 vorgestellte Laufmodell verwendet 127 Parametersätze, die man einzeln optimieren kann. Allerdings ist das nicht in allen Fällen nötig und oft nicht einmal sinnvoll. Einen brauchbaren Anfang für diese 127 Parametersätze liefert das Duplizieren der beiden in Abschnitt 3.3 gefundenen Parametersätze für Vorwärtslaufen inklusive Drehung und Korrektur sowie für Rückwärtslaufen inklusive Korrektur.

Aus der Vergangenheit und der Arbeit anderer ([32]) sowie durch manuelle Modifikationen sind jedoch kompatible Parametersätze bekannt, die zumindest in Spezialfällen wie Geradeauslaufen oder Drehen bessere Ergebnisse liefern.

Um dabei objektiv entscheiden zu können, für welche der 127 Bewegungsanforderungen welcher Parametersatz benutzt werden soll, müssen die in Frage kommenden Parametersätze vermessen werden. Stellt sich dabei heraus, dass eine bestimmte

Bewegungsanforderung mit keinem der getesteten Parametersätze gut umzusetzen ist, kann man gezielt einen Parametersatz für diese Bewegungsanforderung optimieren. In den Fällen, in denen offensichtlich ist, durch welche Parameteränderung das Problem zu lösen ist, kann diese Optimierung sehr schnell durch manuelle Parameterkorrektur erfolgen. Andernfalls sind Verfahren aus solchen Arbeiten nutzbar, in denen Laufparameter für eine konstante Bewegungsanforderung optimiert werden [5, 21, 32].

3.4.1 Geschwindigkeitsmessung

Zunächst soll die Geschwindigkeit eines konkreten konstanten Laufens, also eines Parametersatzes bei einer Bewegungsanforderung, möglichst exakt vermessen werden. Zu bestimmen sind dabei die Rotationsgeschwindigkeit $\dot{\varphi} = d\varphi/dt$ sowie die Translationsgeschwindigkeiten $\dot{x} = dx/dt$ nach vorne und $\dot{y} = dy/dt$ nach links. Dazu wird die mit fliegendem Start in einer bestimmten Zeit t insgesamt zurückgelegte Strecke gemessen. Damit dies möglich ist, wird zunächst je nach zu vermessender Bewegung eine geeignete Startposition berechnet. Sei dabei (x_0, y_0, φ_0) die Startposition und -ausrichtung und (x_t, y_t, φ_t) die Position des Roboters nach Zeit t , dann ist:

$$\begin{aligned} \varphi_t &= \varphi_0 + t \cdot \dot{\varphi} \\ x_t &= x_0 + \int_0^t (\dot{x} \cdot \cos(\varphi_t) - \dot{y} \cdot \sin(\varphi_t)) dt \end{aligned} \quad (3.1)$$

$$y_t = y_0 + \int_0^t (\dot{x} \cdot \sin(\varphi_t) + \dot{y} \cdot \cos(\varphi_t)) dt \quad . \quad (3.2)$$

Die Drehgeschwindigkeit ist dabei:

$$\dot{\varphi} = \frac{\varphi_t - \varphi_0}{t} \quad . \quad (3.3)$$

Falls der Roboter sich bei der Bewegung nicht dreht ($\dot{\varphi} = 0$) lassen sich die Gleichungen 3.1 und 3.2 wie folgt nach \dot{x} und \dot{y} umstellen:

$$\begin{aligned} \dot{x} &= \frac{(x_t - x_0) \cos(\varphi_0) + (y_t - y_0) \sin(\varphi_0)}{t} \\ \dot{y} &= \frac{(y_t - y_0) \cos(\varphi_0) - (x_t - x_0) \sin(\varphi_0)}{t} \quad . \end{aligned}$$

Im Falle $\dot{\varphi} \neq 0$ dagegen erhält man stattdessen:

$$\dot{x} = \frac{\dot{\varphi}}{2 \sin\left(\frac{\dot{\varphi} \cdot t}{2}\right)} \left((x_t - x_0) \cos\left(r_0 + \frac{\dot{\varphi} \cdot t}{2}\right) + (y_t - y_0) \sin\left(r_0 + \frac{\dot{\varphi} \cdot t}{2}\right) \right) \quad (3.4)$$

$$\dot{y} = \frac{\dot{\varphi} \cdot \sin\left(\frac{\dot{\varphi} \cdot t}{2}\right) \left((y_0 - y_t) \cos\left(\varphi_0 + \frac{\dot{\varphi} \cdot t}{2}\right) - (x_0 - x_t) \sin\left(\varphi_0 + \frac{\dot{\varphi} \cdot t}{2}\right) \right)}{\cos(\dot{\varphi} \cdot t) - 1} \quad . \quad (3.5)$$

Auf diese Weise lassen sich die gesuchten Geschwindigkeiten \dot{x} , \dot{y} und $\dot{\varphi}$ bei geeigneter Wahl von Startpunkt (x_0, y_0, φ_0) und geeigneter Messzeit t aus der bis zur Endposition (x_t, y_t, φ_t) zurückgelegten Strecke nach Gleichung 3.3, 3.4 und 3.5 berechnen.

3.4.2 Drehvermessung

Die eben vorgestellte Rechnung funktioniert nur dann genau genug, wenn während der zu vermessenden Bewegung ausreichend lange und zusammenhängend Position und Ausrichtung des Roboters bestimmt werden können. Bei der Vermessung schneller Drehungen ist dies leider nicht der Fall.

Bei schneller Drehung lässt sich die Drehgeschwindigkeit am genauesten dadurch bestimmen, dass man die Zeit zwischen mehreren Nulldurchgängen des Drehwinkels misst, also die Zeit für mehrere komplette Umdrehungen. Es ist bei schneller Drehung allerdings schwierig, den Kopf so zu bewegen, dass die Lokalisierungshilfe so lange im Bild gehalten werden kann, wie sie sich in sichtbarer Position (vor dem Roboter) befindet. Dies gelingt den verwendeten Robotern bis zu einer Drehgeschwindigkeit von etwa einer Umdrehung in 5 Sekunden ($\dot{\varphi}=1,2\text{rad/s}$), wobei die Laufgeschwindigkeiten \dot{x} und \dot{y} noch sicher bestimmt werden können.

Bei höheren Drehgeschwindigkeiten wird der Kopf starr geradeaus gehalten, damit die Lokalisierungshilfe durch Kopfbewegungen nicht versehentlich aus dem Bild gerät oder wegen Bewegungsunschärfe unerkennbar wird. Da die Lokalisierungshilfe durch die starre Kopfhaltung jeweils nur kurz (wenn auch sicherer) zusammenhängend erkannt wird, kann die Laufgeschwindigkeit nicht mehr sicher genug bestimmt werden, was bei sehr schneller Drehung aber toleriert werden kann.

3.4.3 Parametersatzvermessung

Die Geschwindigkeitsmessung mit fliegendem Start erfordert je nach gewünschter Bewegung die Berechnung einer günstigen Startposition. Diese ist jeweils so gewählt, dass der Roboter bei Ausführung der geforderten Bewegung die Lokalisierungshilfe mehrere Sekunden im Blick haben kann, ohne dabei mit der Bande um den Testteppich oder der Lokalisierungshilfe zu kollidieren. Die Messung beginnt erst zwei Sekunden nach Start der Bewegung, weil der Roboter seine endgültige Geschwindigkeit nicht viel früher erreicht hat.

Um einen Parametersatz komplett zu vermessen, ist die Entwicklung der Laufgeschwindigkeit mit steigender Ansteuerung in alle Richtungen zu untersuchen. Dazu genügt die Betrachtung der im Laufmodell aus Abschnitt 2.4 verwendeten acht Laufrichtungen und sieben Dreh-Lauf-Verhältnisse.

Unter Beachtung von Symmetrien ist es dabei ausreichend, nur Drehen in eine Richtung sowie Laufen in acht Richtungen mit drei Dreh-Lauf-Verhältnissen, also insgesamt $1 + 3 \cdot 8 = 25$ Laufrichtungen, zu vermessen. Das ist trotzdem noch relativ aufwändig, wenn auch gut automatisiert. Dafür ist dieser Schritt pro neuen Parametersatz nur einmal nötig, bei spezialisierten Parametersätzen in der Regel sogar nur ein Teil davon, zum Beispiel die Vermessung des Drehens.

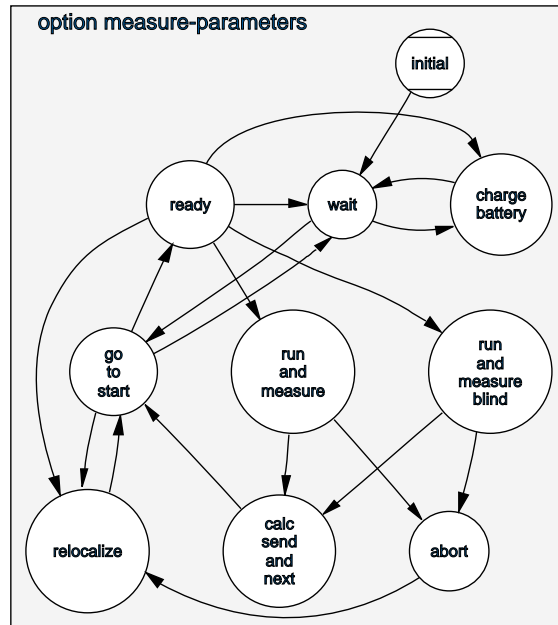


Abbildung 3.12: Xabsl-Zustandsautomat des Verhaltens zur Vermessung eines Parametersatzes: Die eigentliche Arbeit wie die Auswahl der nächsten zu vermessenden Bewegungsanforderung oder die Bestimmung der realen Geschwindigkeit erfolgt innerhalb der Zustände (Kreise).

Von der entstandenen Verhaltenssteuerung (Abbildung 3.12) werden nun einzeln die 25 Lauf- und Drehrichtungen mit jeweils steigender Geschwindigkeit vermessen. Dazu wird die normierte Gesamtgeschwindigkeit (einheitenlos, siehe Abschnitt 2.4.4) in die aktuelle Richtung schrittweise um 0,05 erhöht.

Die nächste Laufrichtung wird erst dann zu vermessen begonnen, wenn trotz erhöhter angesteuerter Geschwindigkeit die gemessene Geschwindigkeit oder deren Verhältnis zur angesteuerten signifikant gesunken ist. Auf diese Weise erhält man pro Richtung etwa 15 bis 35 Messwerte (entspricht einer angesteuerten normierten Geschwindigkeit von 0,75 bis 1,75), je nachdem, wie schnell der untersuchte Parametersatz den Roboter in diese Richtung fortbewegen kann.

Exemplarisch wird hier das Ergebnis der Vermessung eines Parametersatzes, der für die German Open 2004 vom GermanTeam-Mitglied Microsoft Hellhounds entwickelt wurde, für zwei Laufrichtungen dargestellt und erläutert.

Sehen wir uns zunächst das Geradeauslaufen in Abbildung 3.13 an. Bei der Ansteuerung geringer Geschwindigkeiten liegt die gemessene Geschwindigkeit niedriger als gewünscht, bei etwa 80 mm/s stimmen angesteuerte und gemessene Geschwindigkeit überein. Ab 80 mm/s ist die gemessene Geschwindigkeit höher als erwartet und erreicht schließlich bei etwa 370 mm/s ihr Maximum. Eine weitere Erhöhung der Ansteuerung führt dann nicht mehr zu höherer Laufgeschwindigkeit.

Das Laufmodell aus Abschnitt 2.4 verwendet pro Laufrichtung drei Parametersätze, einen für niedrige, einen für mittlere und einen für maximale Geschwindigkeit.

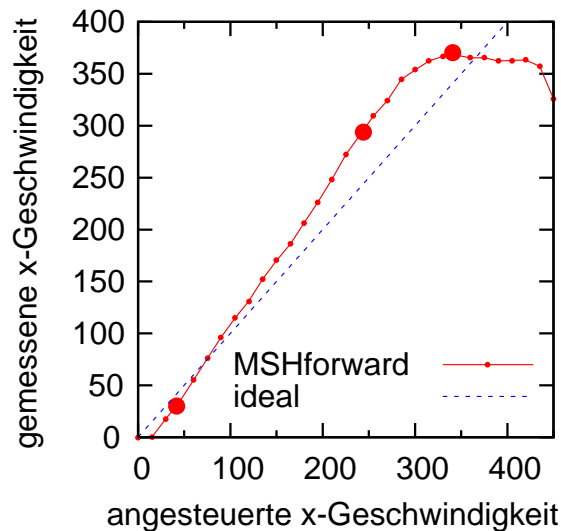


Abbildung 3.13: Ergebnis der Vermessung der Vorwärtslaufgeschwindigkeit eines ERS-7 mit einem einzelnen fixen Parametersatz (MSHforward): Markiert sind die aufgrund der Vermessung gewählten Werte für niedrige, mittlere und hohe Geschwindigkeit.

Eines der Ziele dieser Konstruktion ist es, mit möglichst wenigen zu kalibrierenden Parametersätzen die Abweichung zwischen angeforderter und gemessener Geschwindigkeit über den gesamten Geschwindigkeitsbereich zu minimieren. Um dies zu erreichen, wurden manuell drei geeignete Geschwindigkeiten ausgewählt (und in Abbildung 3.13 markiert), zwischen denen lineare Interpolation die Messkurve gut annähert.

Auf Grundlage der bisher betrachteten Messwerte hätte die Entscheidung für die drei Geschwindigkeiten auch nach einfachen Kriterien automatisch erfolgen können. Die Messungen haben jedoch zusätzlich Abweichungen bei Seitwärts- und Drehgeschwindigkeit aufgedeckt, die noch nicht berücksichtigt worden sind.

Jetzt betrachten wir die Vermessung des gleichen Parametersatzes für die Laufrichtung $\pi/4$ (diagonal nach vorne links) und das Dreh-Lauf-Verhältnis 0,1 (Abbildung 3.14). Aus Laufrichtung und Dreh-Lauf-Verhältnis resultiert die Roboterbewegung auf einem Kreis fixen Durchmessers mit variabler Geschwindigkeit.

Anfangs folgen dort die gemessenen Dreh- und Laufgeschwindigkeiten recht gut den angesteuerten. Etwa ab einer angeforderten Seitwärtsgeschwindigkeit von $\dot{y} = 180 \text{ mm/s}$ und zugehöriger Drehgeschwindigkeit von $\dot{\varphi} = 0,36 \text{ rad/s}$ steigt die gemessene Seitwärtsgeschwindigkeit nur noch deutlich langsamer als die angesteuerte an, die gemessene Drehgeschwindigkeit nimmt sogar wieder ab.

Auch hier sind wieder geeignete Stellen für niedrige, mittlere und maximale Geschwindigkeit zu wählen. Damit soll erreicht werden, dass die auftretenden Abweichungen nicht nur erkannt, sondern auch korrigiert werden können. Speziell für maximale Geschwindigkeit ist die Entscheidung dabei nicht trivial und wird daher keiner

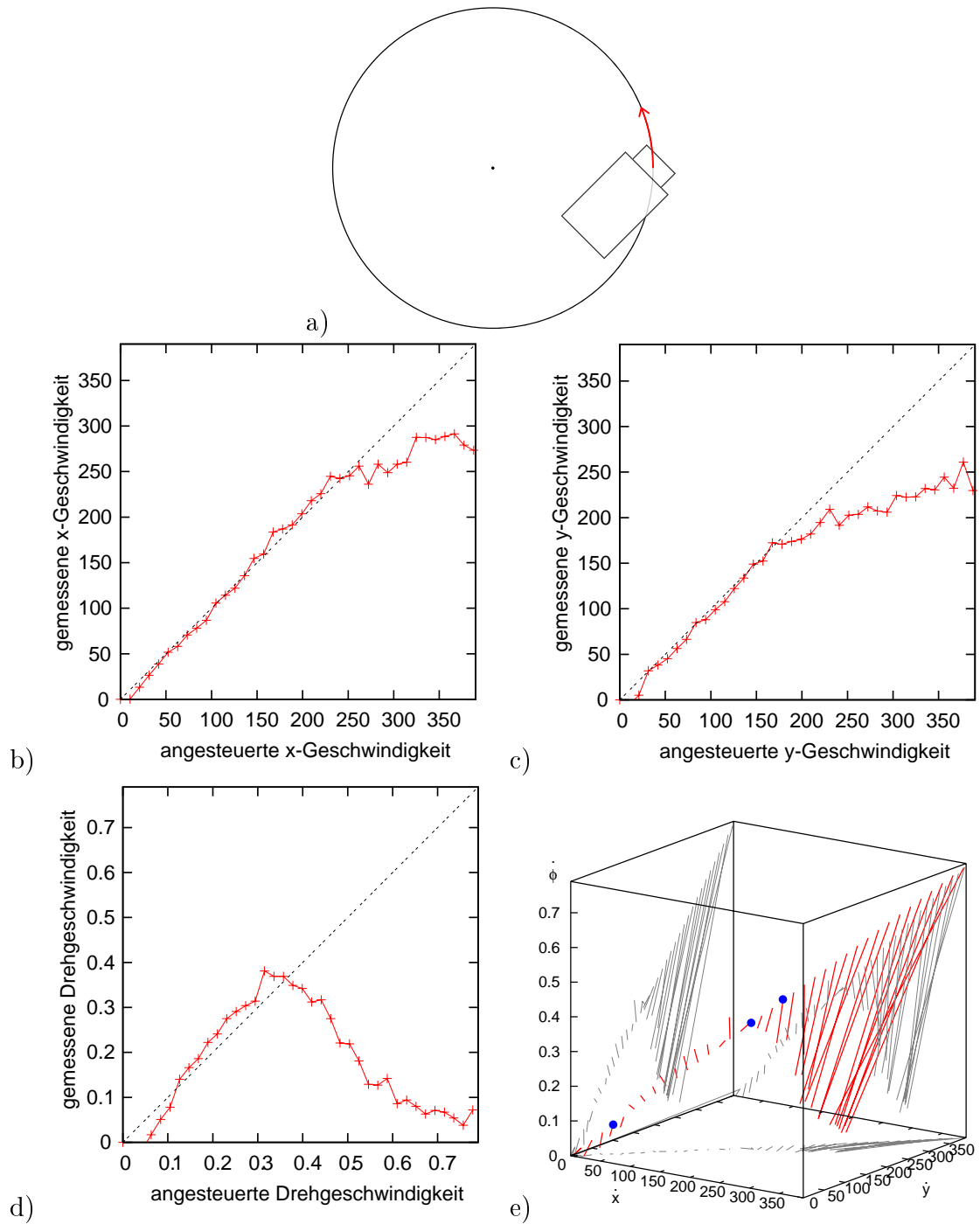


Abbildung 3.14: Vermessung eines ERS-7-Parametersatzes für Laufrichtung $\pi/4$ (diagonal) und Dreh-Lauf-Verhältnis 0,1: **a)** Aus Laufrichtung und Dreh-Lauf-Verhältnis resultiert die Roboterbewegung auf einem Kreis fixen Durchmessers mit variabler Geschwindigkeit. Zu sehen sind angesteuerte und gemessene Werte für **b)** \dot{x} , **c)** \dot{y} und **d)** $\dot{\varphi}$ sowie **e)** der Raumabstand zwischen angesteuerten und gemessenen Geschwindigkeiten (rot) mit Projektionen dieses Abstandes auf drei Ebenen (grau).

Automatik überlassen, da eine Erhöhung der Ansteuerung zwar zu höherer Gesamtgeschwindigkeit führt, aber ab der Stelle, ab der die Drehgeschwindigkeit wieder abnimmt, nicht mehr das gewünschte Dreh-Lauf-Verhältnis erreicht wird. Die Maximalgeschwindigkeit für das gewünschte Dreh-Lauf-Verhältnis liegt daher geringfügig über dieser Stelle, da die gemessene Drehgeschwindigkeit sich durch stärker angesteuerte Drehung noch korrigieren lässt. Die Maximalgeschwindigkeit wurde deshalb auf etwa 210 mm/s Seitwärts- und Vorwärtsgeschwindigkeit und zugehörige 0,42 rad/s Drehgeschwindigkeit festgelegt, siehe blaue Markierungen in Abbildung 3.14 e.

Die genauen Werte für die Maximalgeschwindigkeit in die gewünschte Richtung und die dazu nötige Ansteuerung lassen sich dann mittels Kalibrierung in Abschnitt 3.5 bestimmen, aber durch die gerade erfolgte Vermessung ist bereits eine bessere Abschätzung als vorher möglich. Die mittlere Geschwindigkeit wird hier kurz vor der Stelle gewählt, ab der die gemessene Drehgeschwindigkeit wieder abnimmt, damit lineare Interpolation zwischen den einzelnen Geschwindigkeitsstufen zu einem geringen Durchschnittsfehler führt.

Nun existiert aber nicht nur ein in Frage kommender Parametersatz, sondern mehrere. Diese müssen nicht alle komplett vermessen werden, oft genügt die Vermessung der Spezialfälle, für die ein bestimmter Parametersatz besser geeignet zu sein scheint als die Standardlösung.

Insbesondere für Rückwärtslaufen und Drehen sind spezialisierte Parametersätze gefunden worden. Ein für schnelles Drehen optimierter Parametersatz hat zum Beispiel die Ruhepositionen der Vorderfüße viel dichter an denen der Hinterfüße, um seinen „Wendekreis“ zu verkleinern. Da damit erwartungsgemäß keine ganz großen Vorwärtsschritte mehr möglich sind, braucht Vorwärtslaufen mit diesem Parametersatz nicht vermessen zu werden.

Der Vergleich der gemessenen Drehgeschwindigkeiten in Abbildung 3.15 dagegen belegt, dass der aufs Drehen spezialisierte Parametersatz beim schnellen Drehen dem Standard-Parametersatz deutlich überlegen ist. Daher möchte man für Drehen mit maximaler Geschwindigkeit genau diesen spezialisierten Parametersatz benutzen. Langsames und mittelschnelles Drehen sollen hingegen mit dem omnidirektionalen Standard-Parametersatz erfolgen, damit flüssigere Richtungsänderungen möglich sind.

Da zwischen den Parametersätzen für mittlere und maximale Drehgeschwindigkeit interpoliert werden wird, muss die mittlere Geschwindigkeit so gewählt werden, dass sie noch deutlich vom Maximum des Standard-Parametersatzes entfernt ist. Nur so ist zu erwarten, dass bereits die Interpolation mit einem geringen Anteil des drehspezialisierten Parametersatzes zu weiterer Erhöhung der resultierenden realen Geschwindigkeit führt. Deshalb wurde in diesem Fall eine mittlere Drehgeschwindigkeit von 1,44 rad/s gewählt.

Das Laufmodell gestattet es, auch für dicht nebeneinander liegende Bewegungsanforderungen sehr verschiedene Parametersätze zu verwenden, was allerdings kontraproduktiv ist, da zwischen diesen Parametersätzen interpoliert wird. Daher sollte man darauf achten, möglichst keine Parametersätze zu verwenden, die sich erheblich von ihren Nachbarn unterscheiden, damit der Vorteil einer geringfügig höheren

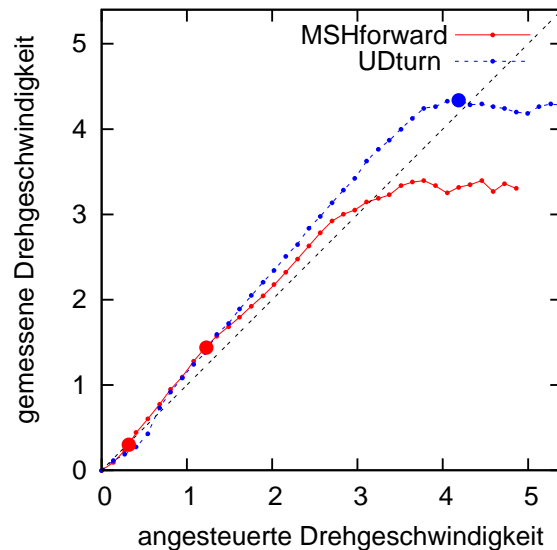


Abbildung 3.15: Vergleich der Vermessung der Drehgeschwindigkeit eines ERS-7 mit einem omnidirektionalen Standard-Parametersatz (MSHforward) und einem aufs Drehen spezialisierten (UDturn): Aufgrund der deutlich höheren Drehgeschwindigkeit will man für schnelles Drehen den spezialisierten Parametersatz benutzen. Markiert sind die gewählten Geschwindigkeiten für langsames, mittelschnelles und maximal schnelles Drehen.

Maximalgeschwindigkeit nicht durch schlechtere Ergebnisse in Bereichen mit Interpolation zwischen den Parametersätzen ausgeglichen wird.

Durch die Vermessung aller in Frage kommenden Parametersätze ist nun bekannt, für welche Bewegungsanforderung sich welcher Parametersatz am besten eignet (siehe auch Tabelle 3.3). Ebenfalls bekannt ist, wie die normierte reale Gesamtgeschwindigkeit aus Laufen und Drehen (siehe Abschnitt 2.4.4) eines konkreten Parametersatzes von der Ansteuerung abhängt. Mit diesem Wissen konnten für jede Bewegungsrichtung geeignete Werte für minimale, mittlere und maximale Geschwindigkeit gewählt werden, die Wahl der 127 Parametersätze ist somit abgeschlossen.

3.5 Kalibrierung

Die einzelnen Parametersätze müssen nun aber noch kalibriert werden, damit die angeforderte Bewegung möglichst exakt mit der tatsächlich ausgeführten übereinstimmt. Solche Abweichungen entstehen zum Beispiel durch Trägheit, Reibung, Ungenauigkeiten im Modell oder mechanische Beschränkungen und resultieren in Geschwindigkeits- und auch Richtungsabweichungen, welche beispielsweise in Abbildung 3.14 zu sehen sind.

Diese Abweichungen wurden durch Parametersatzvermessung bereits bemerkt, aber noch nicht korrigiert. Die Kalibrierung einzelner Parametersätze bietet nun im Gegensatz zur bisher verwendeten, vereinfachenden Annahme weniger linearer

| Parameter | Vorwärtslaufen | Rückwärtslaufen | Drehen |
|------------------------------------|------------------|------------------|------------------|
| x_v | 50,4 mm | 43,2 mm | 33,6 mm |
| y_v | 80,3 mm | 79,1 mm | 78,1 mm |
| z_v | 77,6 mm | 88,5 mm | 86,5 mm |
| x_h | -52,1 mm | -49,7 mm | -29,6 mm |
| y_h | 81,5 mm | 77,7 mm | 76,9 mm |
| z_h | 105,1 mm | 101,9 mm | 113,7 mm |
| h_v | 5 mm | 9,8 mm | 5 mm |
| h_h | 24 mm | 28,6 mm | 24 mm |
| $tilt_v$ | -0,25 | 0,072 | -0,25 |
| $tilt_h$ | 0,016 | -0,092 | 0,045 |
| gp_v | 0,49 | 0,46 | 0,5 |
| gp_h | 0,49 | 0,51 | 0,5 |
| $(l_{vl}, l_{vr}, l_{hl}, l_{hr})$ | (0, 0,5, 0,5, 0) | (0, 0,5, 0,5, 0) | (0, 0,5, 0,5, 0) |
| T | 512 ms | 488 ms | 352 ms |

Tabelle 3.3: Die letztlich gewählten Werte für die wichtigsten der verwendeten Parametersätze: Die Bedeutung der Parameter ist Abschnitt 2.4 zu entnehmen.

Korrekturparameter erstmals die Möglichkeit, nur bei bestimmten Bewegungsanforderungen wie diagonalem Laufen auftretende Abweichungen gezielt zu minimieren.

Das Verhalten zur Kalibrierung ist wieder mit Hilfe von Xabsl spezifiziert (Abbildung 3.16). Die nicht durch Rechts-Links-Symmetrie auseinander hervorgehenden 66 der insgesamt 127 Parametersätze des Laufmodells aus Abschnitt 2.4 werden dabei jeweils für ihre Sollgeschwindigkeit vermessen. Dieser Vorgang dauert etwa 20 Minuten und liefert eine Liste der angeforderten, der korrigiert angesteuerten und der real gemessenen Geschwindigkeiten für alle Parametersätze.

Die Ansteuerung derjenigen Parametersätze, die nicht für Maximalgeschwindigkeit ausgelegt sind, kann dabei vollautomatisch verbessert werden. Dazu wird die verwendete Ansteuerung um die Hälfte der Differenz aus angeforderter und gemessener Geschwindigkeit korrigiert. Auf diese Weise sollte sich die Abweichung zwischen Soll und Ist in jedem Kalibrierdurchgang verringern (im Idealfall halbieren), ohne dabei über das Ziel hinauszuschießen.

Für die Parametersätze, die für Maximalgeschwindigkeit gedacht sind, lässt sich die Korrektur nicht auf gleiche Weise automatisieren. Zum einen möchte man bei Messung einer höheren als der angeforderten Geschwindigkeit nicht die Ansteuerung, sondern die Maximalgeschwindigkeit in diese Laufrichtung korrigieren. Zum anderen ist nicht sichergestellt, dass die Änderung der Ansteuerung um die Hälfte der Differenz zwischen Anforderung und Messung tatsächlich zu einer Verringerung dieser Differenz führt. Es ist stattdessen gut möglich, dass die untersuchte Bewegung bereits ausgereizt. Dann hat zum Beispiel die weitere Erhöhung der angesteuerten

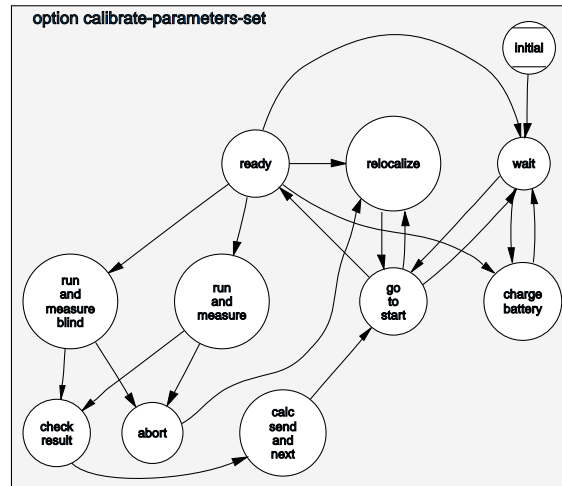


Abbildung 3.16: Xabl-Zustandsautomat des Verhaltens zur Kalibrierung aller Parametersätze für ihre jeweils vorgesehene Bewegungsanforderung: Die eigentliche Arbeit wie die Wahl des nächsten Parametersatzes oder die Vermessung eines solchen erfolgt innerhalb einzelner Zustände (Kreise).

Drehgeschwindigkeit nicht mehr die Erhöhung der gemessenen Drehgeschwindigkeit zur Folge. Daher erfolgt die Korrektur der Ansteuerung der Parametersätze für Maximalgeschwindigkeit manuell und nur auf Grundlage der angeforderten, der korrigiert angesteuerten und der tatsächlich gemessenen Geschwindigkeiten.

Für einen kompletten Kalibrierdurchgang wird auf diese Weise rund eine knappe Stunde benötigt, davon die Hälfte für automatische Vermessung und Korrektur und die andere für manuelle Überprüfung und Kalibrierung der Parametersätze für Maximalgeschwindigkeit.

3.5.1 Kalibrierungsergebnis

Die Kalibrierung hatte das Ziel, den Unterschied zwischen angeforderten und tatsächlich gemessenen Geschwindigkeiten zu minimieren. Um den Effekt der Kalibrierung zu sehen, wurde das Vorwärtslaufen aus Abbildung 3.13 erneut vermessen, diesmal aber mit (in mehreren Schritten) kalibrierter Ansteuerung. Abbildung 3.17 zeigt die dadurch erzielte, wesentlich bessere Übereinstimmung zwischen angeforderter und gemessener Geschwindigkeit.

Nicht aus der Abbildung ersichtlich ist dabei, welche Geschwindigkeit an den drei markierten Parametersätzen jeweils angesteuert wurde, um diese Übereinstimmung zu erreichen. Entscheidend ist nur, wie gut dies durch geeignete Korrektur der Ansteuerung möglich ist, und das nicht nur für die drei kalibrierten (markierten) Geschwindigkeitsstufen, sondern durch Interpolation auch für die Bereiche dazwischen. Die verbleibende Abweichung zwischen mittlerer und maximaler Geschwindigkeit ist der zu zahlende Preis, um eine höhere Maximalgeschwindigkeit zu erzielen.

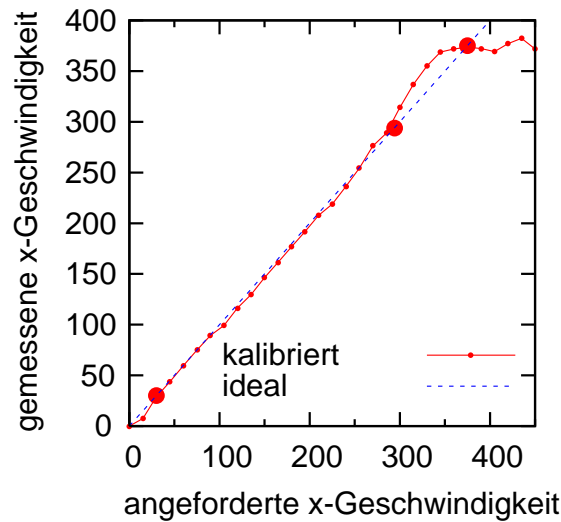


Abbildung 3.17: Ergebnis der Vermessung der Vorwärtslaufgeschwindigkeit eines ERS-7, die sich aus Interpolation zwischen drei kalibrierten Parametersätzen für niedrige, mittlere und hohe Geschwindigkeit (markiert) ergibt: Die Abweichungen zwischen angeforderter und gemessener Geschwindigkeit sind erheblich geringer als ohne Kalibrierung (in Abbildung 3.13).

Neben diesem Beispiel fürs Vorwärtslaufen wurde auch die Durchschnittsabweichung zwischen Soll- und Istgeschwindigkeiten über alle zu kalibrierenden Parametersätze bestimmt. Gemittelt wurden dabei die Beträge der Abweichungen und nicht deren Quadrate, damit einzelne Messfehler und systematisch höhere Abweichungen etwa bei höherer Sollgeschwindigkeit kein zu großes Gewicht bekommen.

Tabelle 3.4 zeigt die ermittelten durchschnittlichen Abweichungen über alle zu kalibrierenden Parametersätze in verschiedenen Stadien des Optimierungsprozesses. Die erste Zeile enthält die bei der vollständigen Vermessung in Abschnitt 3.4.1 verwendeten Parametersätze ohne jegliche Kalibrierung der Ansteuerung, entsprechend sind hier die Abweichungen am höchsten. Dann folgt die erste, direkt aus dieser Messung abgeleitete, manuelle Anpassung der Ansteuerung.

| | $ \Delta\dot{x} $ in mm/s | $ \Delta\dot{y} $ in mm/s | $ \Delta\dot{\varphi} $ in rad/s |
|-----------------|---------------------------|---------------------------|----------------------------------|
| unkalibriert | 12,9 | 12,1 | 0,086 |
| aus Messung | 7,7 | 11,4 | 0,070 |
| Kalibrierung 1 | 6,4 | 7,6 | 0,038 |
| Kalibrierung 2 | 5,8 | 8,0 | 0,030 |
| Kalibrierung 3 | 4,4 | 8,7 | 0,021 |
| 2 Wochen später | 7,9 | 7,5 | 0,046 |
| anderer Roboter | 6,9 | 7,8 | 0,039 |

Tabelle 3.4: Die durchschnittliche Abweichung zwischen Sollgeschwindigkeit und Istgeschwindigkeit eines ERS-7 lässt sich durch Kalibrierung deutlich reduzieren.

Die darauf folgenden drei Kalibrierungsdurchgänge führen, wie in Tabelle 3.4 zu sehen ist, zu weiterer deutlicher Verringerung der Abweichung zwischen Soll und Ist. Die Abweichung der Drehgeschwindigkeit, die besonders bei der Kombination von Drehen mit schnellem Laufen auftrat, wurde erheblich reduziert. Diese Verringerung der Drehabweichung geht allerdings auch mit einer leichten Erhöhung der Abweichung der Seitwärtsgeschwindigkeit einher. Mit weiteren Kalibrierungsschritten, bei denen nur die Seitwärtsgeschwindigkeit nennenswert modifiziert wird, ließe sich dieser Effekt ausgleichen.

Da die ermittelten durchschnittlichen Abweichungen auch Messfehler enthalten, lassen sie sich nicht beliebig verringern. Um zu ermitteln, welche Genauigkeit in der Praxis sinnvoll ist, wurde die Messung mit den Ergebnissen des dritten Kalibrierungsschrittes nach einigen Wochen mit verschiedenen Robotern wiederholt. Dabei fiel sofort auf, dass der ursprünglich zum Kalibrieren verwendete Roboter nach zwei Wochen intensiver Benutzung (Vorbereitung zur Weltmeisterschaft) deutlich schlechtere Ergebnisse lieferte. Auch ein anderer, baugleicher, aber bis dahin nicht zum Kalibrieren verwendeter Roboter lieferte stärkere Abweichungen als der erste Roboter direkt nach dem dritten Kalibrierungsschritt.

Mehr als drei Kalibrierungsschritte wären daher nur sinnvoll, wenn die Kalibrierung abhängig von Faktoren wie dem Verschleißzustand der Roboter und damit regelmäßig und für einzelne Roboter erfolgt, was jedoch mit einem nicht zu rechtfertigenden Aufwand an Zeit und Verschleiß verbunden ist. Da die Abweichungen nach drei Kalibrierungsschritten in Folge aber auch auf anderen Robotern und nach längerer Benutzung noch erheblich niedriger als ohne Kalibrierung sind, ist eine Kalibrierung in drei Schritten ein sinnvoller Kompromiss.

Am Ende der Kalibrierung stehen nun Maximalgeschwindigkeiten und kalibrierte Ansteuerungen für alle Laufrichtungen und Dreh-Lauf-Verhältnisse zur Verfügung, für die das verwendete Laufmodell eigene Parametersätze besitzt. In Abbildung 3.18 sind die Geschwindigkeiten all derjenigen verwendeten Parametersätze zu sehen, die kein Drehen beinhalten.

Dabei ist gut zu erkennen, dass ein deutlich größerer Geschwindigkeitsbereich abgedeckt werden kann als bei der Verwendung nur eines Parametersatzes und der dabei nötigen Beschränkung der Laufgeschwindigkeit etwa durch eine Ellipse. Zusätzlich sind diese höheren Geschwindigkeiten jetzt auch noch kalibriert, die gemessene reale Geschwindigkeit entspricht also besser der durch die Ansteuerung beabsichtigten.

3.6 Optimierungsergebnis

Nach erfolgter Optimierung wird wie angekündigt nochmals der Parcours aus Abschnitt 3.3.1 verwendet, um zu überprüfen, inwieweit sich das Laufen eines Roboters verbessert hat.

Die Optimierung des Laufens erfolgte in drei Schritten. Den Anfang bildete die evolutionäre Verbesserung eines omnidirektionalen Parametersatzes. Dann folgte die Vermessung dieses sowie weiterer zur Verfügung stehender Parametersätze und da-

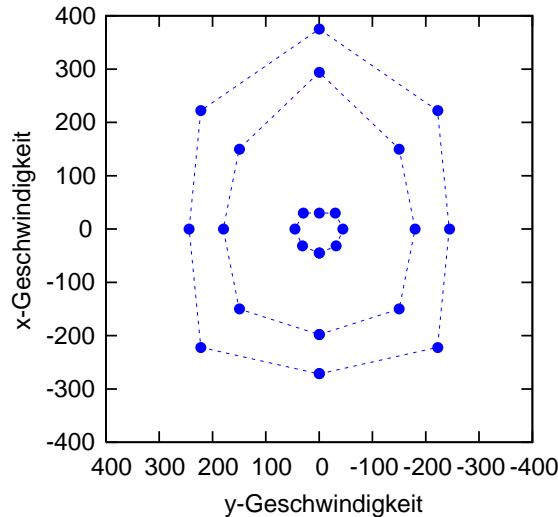


Abbildung 3.18: Die gewählte minimale, mittlere und maximale Geschwindigkeit eines ERS-7 für alle acht Laufrichtungen ohne Drehung: Mit der Wahl jeweils dafür kalibrierter Parametersätze lässt sich ein deutlich größerer Geschwindigkeitsbereich abdecken als durch einen einzigen Parametersatz und Geschwindigkeitsbeschränkung auf eine Ellipse.

raufhin die Auswahl bestimmter Parametersätze für bestimmte Bewegungsanforderungen. Schließlich wurden alle verwendeten Parametersätze für ihren jeweiligen Einsatzzweck kalibriert. Erwartungsgemäß sollte sich durch optimierte, spezialisierte und kalibrierte Parametersätze und die daraus resultierenden richtungsabhängigen, höheren Maximalgeschwindigkeiten die Fitness des Laufens auf dem Parcours deutlich erhöhen.

Abbildung 3.19 zeigt, wie die Fitness der Parametersätze für einen ERS-7 im Laufe des Optimierungsprozesses gestiegen ist. Der vor dieser Arbeit für einen ERS-210 handoptimierte und zur RoboCup-Weltmeisterschaft 2003 eingesetzte Parametersatz („GT2003“) ist relativ schlecht für einen ERS-7 geeignet. Bereits der erste durch Evolution, aber ebenfalls noch für einen ERS-210 entstandene Parametersatz („GO2004“) führt zu deutlich höherer Fitness auf einem ERS-7 und wurde vom AiboTeamHumboldt erfolgreich bei der German Open 2004 eingesetzt.

Die weitere, jetzt ERS-7-spezifische und für Vorwärts- und Rückwärtslaufen getrennte Evolution von Parametersätzen erhöht die Fitness nochmals deutlich („evolviert“). Nur noch geringfügig verbesserte sich die Fitness dagegen durch die Wahl der 127 Parametersätze in Abschnitt 3.4 („unkalibriert“) sowie die in Abschnitt 3.5 erfolgte Kalibrierung („kalibriert“). Die Wahl vieler einzelner Parametersätze verbesserte das Laufen besonders in von der Fitnessfunktion auf dem Parcours nicht erfassten Spezialfällen wie schnellem Drehen. Die Kalibrierung machte das Laufen vor allem genauer, insbesondere bei langsamerer als der maximalen Geschwindigkeit, was ebenfalls nur wenig Einfluss auf die Fitnessfunktion hatte.

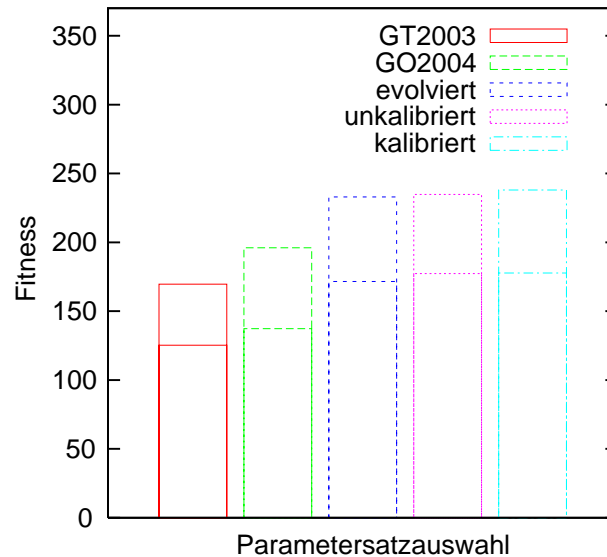


Abbildung 3.19: ERS-7: Das verwendete Evolutionsverfahren erzeugt Parametersätze mit deutlich höherer Fitness, während die Kalibrierung keinen wesentlichen Einfluss auf die Fitness hat. Zu sehen sind die Fitnesswerte ausgewählter Parametersätze beziehungsweise Parametersatzkombinationen, jeweils mit ihrer Unterteilung in Vorwärts- (Wichtung 2/3) und Rückwärtskomponente.

Alle drei Optimierungsschritte (Evolution, Auswahl spezialisierter Parametersätze und Kalibrierung) haben somit einen relevanten Beitrag zur Verbesserung des Laufens geleistet. Dies wurde durch den Gewinn der RoboCup-Weltmeisterschaft 2004 durch das GermanTeam, das die Ergebnisse der in dieser Arbeit vorgestellten Laufoptimierungen nutzte, bestätigt.

4 Zusammenfassung und Ausblick

In dieser Arbeit wurden die Laufbewegungen vierbeiniger Roboter vom Typ Sony ERS-210 und ERS-7 modelliert, optimiert und kalibriert. Das entstandene Laufmodell erlaubt omnidirektionale Fortbewegung und bietet erstmals die Möglichkeit, für verschiedene Laufrichtungen und -geschwindigkeiten auch verschieden optimierte Parametersätze zu verwenden, ohne explizit zwischen diesen umschalten zu müssen. Verbesserungen an einer Laufrichtung müssen so nicht mehr mit Verschlechterungen an anderer Stelle erkauft werden.

Darüber hinaus können bei der Geschwindigkeit der Roboter zwischen Theorie und Realität auftretende Abweichungen erstmals gezielt lokal korrigiert werden. Die Korrektur nur bei Maximalgeschwindigkeit auftretender Effekte hat zum Beispiel keinen Einfluss mehr auf die Genauigkeit des Laufens mit mittlerer Geschwindigkeit. Somit sind erreichbare Maximalgeschwindigkeiten und Abweichung zwischen gewünschter und tatsächlicher Bewegung bis zum Erreichen dieser Geschwindigkeiten voneinander entkoppelt und die Maximalgeschwindigkeiten dürfen richtungsabhängig verschieden sein.

Mit dem verwendeten Optimierungsverfahren wurde versucht, gezielt omnidirektionales Laufen mit sich ändernder Laufrichtung zu verbessern, statt wie sonst üblich nur die Ausführung konstanter Bewegungsanforderungen zu optimieren. Auf diesem Weg konnten dann auch tatsächlich Laufparameter gefunden werden, die zu einem deutlich leistungsfähigeren Laufen führen.

Es hat sich allerdings auch herausgestellt, dass es sich kaum lohnt, omnidirektionales Laufen (mit nur einem konstanten Parametersatz) auf Geschwindigkeit zu optimieren, da für maximal schnelles Vorwärts- und Rückwärtslaufen verschiedene Parameter benötigt werden. Stattdessen ist es wichtiger, ein Modell wie das in Abschnitt 2.4 zu benutzen, dass die Verwendung unterschiedlicher Parametersätze für unterschiedliche Spezialaufgaben unterstützt, den Übergang dazwischen aber so einfach wie möglich gestaltet.

Dafür sind Parametersätze nötig, die nicht nur ausschließlich ihre Spezialaufgabe beherrschen, sondern auch für ähnliche Bewegungsanforderungen geeignet sind. Die Einbeziehung von Laufrichtungskorrekturen in den Optimierungsprozess hat sich als geeignet erwiesen, zu solchen Parametersätzen zu gelangen. Das verwendete Evolutionsverfahren war in der Lage, in kurzer Zeit deutlich bessere als den vorher benutzten Parametersatz zu erzeugen.

Künftige Optimierungsverfahren sollten bestrebt sein, mehrere Optimierungsschritte zu integrieren. Es erscheint insbesondere sinnvoll, Parametersätze für verschiedene Bewegungsanforderungen unter Einbeziehung der Übergänge dazwischen gleichzeitig zu evolvieren, da die Verwendung verschiedener aufeinander abgestimmter Parametersätze der derzeit erfolgreichste Ansatz ist.

Nach der Erzeugung guter Parametersätze wurde die Geschwindigkeit des Laufens mit allen in Frage kommenden Parametersätzen vermessen. Dabei war festzustellen, wie hilfreich ein exaktes Verfahren zur Geschwindigkeitsermittlung ist. Die Bestimmung richtungsabhängiger Maximalgeschwindigkeiten, der objektive Vergleich zweier Parametersätze oder das Bemerkens des nichtlinearen Zusammenhangs zwischen angeforderter und real gemessener Geschwindigkeit wären allein mit manuellen Messmethoden kaum möglich gewesen.

Auch die Kalibrierung des Laufens konnte erst auf Grundlage dieser exakten Messungen erfolgen. Die Kalibrierung wurde erstmals für verschiedene Lauf- und Drehrichtungen und -geschwindigkeiten einzeln durchgeführt, was die Übereinstimmung zwischen angeforderter und tatsächlich erreichter Geschwindigkeit deutlich verbessert. Dies ist besonders in Situationen mit ungenauer Lokalisierung hilfreich, zum Beispiel beim Fußballspielen im RoboCup.

Am Ende des Optimierungsprozesses stand ein Laufmodell mit dafür geeigneten Parametern zur Verfügung, das omnidirektionales Laufen und Drehen mit 30 Prozent höherer Geschwindigkeit als vorher erlaubt. Durch Kalibrierung konnten die auftretenden Abweichungen zwischen Bewegungsanforderung und real gemessener Geschwindigkeit auf ein Drittel reduziert werden, ohne dafür die maximale Laufgeschwindigkeit künstlich beschränken zu müssen. Die Verwendung der vorgestellten Laufbewegungen leistete einen Beitrag zum Gewinn der German Open 2004 durch das AiboTeamHumboldt sowie zum Sieg des GermanTeam bei der RoboCup-Weltmeisterschaft 2004 in Lissabon.

Es wurde Wert darauf gelegt, dass ein möglichst großer Teil der Optimierung und Kalibrierung der Laufparametersätze automatisch erfolgt. Sowohl die eingeführte Kalibrierung von Parametersätzen als auch die Verwendung mehrerer Parametersätze erhöhen den Aufwand deutlich, wobei sich aber der Mehraufwand für einen menschlichen Trainer durch die benutzten Automatisierungen in Grenzen hält.

Darüber hinaus verbessert ein automatisches Verfahren auch die Vergleichbarkeit und Reproduzierbarkeit der Ergebnisse, weil einzelne Rahmenbedingungen wie die Länge der zur Geschwindigkeitsmessung herangezogenen Laufstrecke nicht mehr vom Bediener abhängen. Unumgänglich war an dieser Stelle auch die Verwendung verschiedener Benchmarks, beispielsweise der eingeführten Fitnessfunktion oder genau vermessener Geschwindigkeiten, da ein automatisches Verfahren nur anhand solcher objektiver Kriterien über die Qualität eines Laufparametersatzes entscheiden kann.

Für die genaue Vermessung der Geschwindigkeit des Laufens war eine exakte Lokalisierung nötig, für die ein nicht unerheblicher Aufwand getrieben wurde. Die Verfügbarkeit einer anderweitigen ausreichend genauen Orts- und Richtungsbestimmung der Aibos hätte also viel Arbeit sparen können.

Dennoch ist die gefundene Lösung, eine leicht selbst herstellbare, passive, portable, vom Roboter selbst auswertbare Lokalisierungshilfe, ein geeigneter Weg. Statt teurer externer Hardware (Deckenkamera, Laserscanner) genügten hier einfache Hilfestellungen.

Hat allgemein ein Roboter für eine bestimmte Aufgabe keinen passenden Sensor und steht die nötige Information nicht oder nicht sicher aus externer Quelle zur Verfügung, so ist es oft angebracht, dem Roboter zum Beispiel mit einfachen Markierungen wichtiger Objekte oder Vergleichsgrößen auf die Sprünge zu helfen. Menschen finden geeignete Orientierungspunkte selbstständig in ihrer Umgebung, einem Roboter stellt man diese derzeit besser noch zur Verfügung.

Die am Ende dieser Arbeit benutzten 127 Parametersätze sind nicht optimal. Zahlreiche Einzelfälle hätten sich mit geeigneten Optimierungsverfahren weiter verbessern lassen, zum Beispiel blieb die maximale Seitwärtsgeschwindigkeit eines ERS-7 hinter den Erwartungen zurück. Die Optimierung der Parameter für einzelne konstante Bewegungsanforderungen war jedoch nicht das Ziel dieser Arbeit, da es in dieser Richtung schon einige Veröffentlichungen gab [5, 32].

Stattdessen wurde mehr Wert auf omnidirektionale Verwendbarkeit gelegt, unter anderem durch kontinuierliche Übergänge zwischen allen Parametersätzen. Das Ergebnis ist ein gut funktionierendes Zusammenspiel mehrerer relativ ähnlicher und einzeln kalibrierter Parametersätze, was zu geringen Soll-Ist-Abweichungen bei hoher Geschwindigkeit führt.

Das aus meiner Sicht größte Potential liegt derzeit in der Optimierung der Fußtrajektorie. Es gab durchaus schon erfolgreiche Versuche in dieser Richtung, beispielsweise die Verwendung beliebiger vier Punkte im Raum statt eines Parallelogramms in einer senkrechten Ebene (*FreeFormQuad*, siehe [5], damals nur mit ERS-210). Dennoch sind viele der auftretenden Effekte noch nicht ausreichend verstanden und modelliert, zum Beispiel das intentionale Ausnutzen des Abrollens auf den Unterschenkeln.

Ein weiteres Beispiel ist das Ergebnis von [21], wo sich sehr hohe Laufgeschwindigkeiten mit einem modellierten Duty-Factor von 0,43 erreichen lassen. Dabei wären theoretisch im Durchschnitt weniger als zwei der vier Beine am Boden, was in der Praxis so nicht zu beobachten ist. Stattdessen befinden sich die Füße einen Teil der Zeit, in der sie als angehoben modelliert werden, noch am Boden, was wie die *CanterAction* aus [14] zu einer Auf- und Abbewegung des Roboters führt.

Die Analyse der Ergebnisse von [32] zeigt darüber hinaus, dass die bisher höchste von einem ERS-7 erreichte Laufgeschwindigkeit dem „Missbrauch“ der Parameter der Fußtrajektorie durch einen evolutionären Algorithmus zu verdanken ist. Es wurde beabsichtigt, die Füße wie bisher üblich ein Viereck oder eine Halbellipse mit jeweils konstanten Geschwindigkeiten auf den wenigen Teilstrecken ohne Pause dazwischen abfahren zu lassen. Stattdessen führte evolutionäre Parameterwahl dazu, dass einzelne Beine an den Eckpunkten ihrer Fußtrajektorie gezielt angehalten wurden, was offenbar die Bodenhaftung erhöht, während andere Beine Schritte ausführen.

Auf die Beherrschbarkeit unebenen oder variablen Untergrundes wurde in dieser Arbeit ebenso wenig eingegangen wie auf die Verwendung andersartiger, insbesondere zweibeiniger Roboter. Beides sind jedoch Voraussetzungen für das eingangs erwähnte Ziel des Einsatzes von Robotern in menschlicher Umgebung. Mit beidem sind unzweifelhaft auch eine Menge neuer Probleme verbunden, doch zahlreiche Grund-

gedanken dieser Arbeit sind nach wie vor anwendbar. Es wird ein parametrisierbares Modell für omnidirektionales Laufen benötigt und einzelne Bewegungen müssen sich gezielt optimieren und kalibrieren lassen. Dies sollte auf Grundlage objektiver Daten erfolgen und weitgehend automatisierbar sein. Somit steht eine erfolgreich erprobte Herangehensweise zur Verfügung, die sich auch auf künftige Modellierungen und Optimierungen von Laufbewegungen übertragen lässt.

Es bleibt zu hoffen, dass die Entwicklung von mehrbeinigen Robotern und deren Laufbewegungen weiterhin spannend bleibt. In diesem Sinne ist den Forschern weltweit zu wünschen, dass die Vision des RoboCup, mit autonomen Robotern den dann amtierenden „echten“ Fußballweltmeister in einem fairen Spiel zu besiegen, eines Tages in Erfüllung geht.

A Literaturverzeichnis

- [1] AiboTeamHumboldt. Webseite. <http://www.aiboteamhumboldt.com>, 2004.
- [2] Minoru Asada, Hiroaki Kitano, Itsuki Noda, Manuela Veloso. RoboCup: Today and tomorrow - what we have learned. *Artificial Intelligence*, 110(2):193–214, 1999.
- [3] Aude Billard, Auke Jan Ijspeert. Biologically inspired neural controllers for motor control in a quadruped robot. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN 2000*, Band 6. IEEE, 2000.
- [4] Ronnie Brunn, Uwe Düffert, Matthias Jüngel, Tim Laue, Martin Löttsch, Sebastian Petters, Max Risler, Thomas Röfer, Andreas Sztybryc. GermanTeam 2001. In Andreas Birk, Silvia Coradeschi, Satoshi Tadokoro (Herausgeber), *RoboCup 2001 Robot Soccer World Cup V*, Nummer 2377 in Lecture Notes in Computer Science, Seiten 705–708, Heidelberg, 2002. Springer. Ausführlicher in <http://www.tzi.de/kogrob/papers/GermanTeam2001report.pdf>.
- [5] Jin Chen, Eric Chung, Ross Edwards, Nathan Wong, Eileen Mak, Raymond Sheh, Min Sub Kim, Ales Tang, Nicodemus Sutanto, Bernhard Hengst, Claude Sammut, Will Uther. rUNSWift 2003. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence, Heidelberg, 2004. Springer. Im Erscheinen.
- [6] Ingo Dahm, Jens Ziegler. Using artificial neural networks to construct a meta-model for the evolution of gait patterns. In P. Bidaud, F. Ben Amar (Herausgeber), *Proceedings of the 5th International Conference on Climbing and Walking Robots (CLAWAR 2002)*, Seiten 825–832, Bury St. Edmunds, GB, 2002. Professional Engineering Publishing.
- [7] Uwe Düffert, Matthias Jüngel, Tim Laue, Martin Löttsch, Max Risler, Thomas Röfer. GermanTeam 2002. In Gal A. Kaminka, Pedro U. Lima, Raúl Rojas (Herausgeber), *RoboCup 2002 Robot Soccer World Cup VI*, Nummer 2752 in Lecture Notes in Computer Science, Heidelberg, 2003. Springer. Ausführlicher in <http://www.tzi.de/kogrob/papers/GermanTeam2002.pdf>.
- [8] J.E.J. Duysens, Henry W.A.A. van de Crommert, Bouwien C.M. Smits-Engelsman, Frans C.T. van der Helm. A walking robot called human: lessons to be learned from neural control of locomotion. *Journal of Biomechanics*, 35(4):447–454, 2000.

- [9] Dieter Fox, Wolfram Burgard, Frank Dellaert, Sebastian Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99)*, Seiten 343–349, Orlando, FL, 1999.
- [10] GermanTeam. Webseite. <http://www.robocup.de/germanteam>, 2004.
- [11] Michael Hardt, Maximilian Stelzer, Oskar von Stryk. Efficient dynamic modeling, numerical optimal control and experimental results for various gaits of a quadruped robot. In *CLAWAR 2003 - 6th International Conference on Climbing and Walking Robots*, Seiten 601–608, Catania, Italien, 2003.
- [12] Michael Hardt, Oskar von Stryk. The role of motion dynamics in the design, control and stability of bipedal and quadrupedal robots. In Gal A. Kaminka, Pedro U. Lima, Raúl Rojas (Herausgeber), *RoboCup 2002: Robot Soccer World Cup VI*, Band 2752 of *Lecture Notes in Computer Science*, Seiten 206–223, Heidelberg, 2003. Springer.
- [13] Bernhard Hengst, Darren Ibbotson, Son Bao Pham, John Dalglish, Mike Lawther, Phil Preston, Claude Sammut. The UNSW RoboCup 2000 Sony Legged League team. In Tucker Balch, Gerhard Kraetzschmar (Herausgeber), *RoboCup 2000: Robot Soccer World Cup IV*, Nummer 2019 in *Lecture Notes in Artificial Intelligence*, Seiten 70–72 (64–75), Heidelberg, 2001. Springer.
- [14] Bernhard Hengst, Darren Ibbotson, Son Bao Pham, Claude Sammut. Omnidirectional locomotion for quadruped robots. In Andreas Birk, Silvia Coradeschi, Satoshi Tadokoro (Herausgeber), *RoboCup 2001 Robot Soccer World Cup V*, Nummer 2377 in *Lecture Notes in Computer Science*, Seiten 368–373, Heidelberg, 2002. Springer.
- [15] Gregory S. Hornby, Seiichi Takamura, Jun Yokono, Osamu Hanagata, Takashi Yamamoto, Masahiro Fujita. Evolving robust gaits with Aibo. *IEEE International Conference on Robotics and Automation*, Seiten 3040–3045, 2000.
- [16] Vincent Hugel, Pierre Blazevic. Towards efficient implementation of quadruped gaits with duty factor of 0.75. In *Proceedings of the IEEE International Conference On Robotics and Automation*, Seiten 2360–2365, 1999.
- [17] Nick Jakobi. The minimal simulation approach to evolutionary robotics. In Takashi Gomi (Herausgeber), *Evolutionary Robotics - From Intelligent Robots to Artificial Life (ER'98)*, Seiten 133–190. AAAI Books, 1998.
- [18] Hiroshi Kimura, Yasuhiro Fukuoka, Yoshiro Hada, Kunikatsu Takase. Adaptive dynamic walking of a quadruped robot on irregular terrain by using a neural system model. *Advanced Robotics*, 15(8):859–876, 2001.
- [19] Hiroshi Kimura, Yasuhiro Fukuoka, Yoshiro Hada, Kunikatsu Takase. Three-dimensional adaptive dynamic walking of a quadruped - rolling motion feedback to CPGs controlling pitching motion. In *Proceedings of IEEE Robotics and Automation (ICRA2002)*, Seiten 2228–2233, 2002.

- [20] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa. RoboCup: The robot world cup initiative. In *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, Seiten 340–347. ACM Press, 1997.
- [21] Nate Kohl, Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004. Im Erscheinen.
- [22] John R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.
- [23] Scott Lenser, James Bruce, Manuela Veloso. CMPack: A complete software system for autonomous legged soccer robots. In M. Brian Blake (Herausgeber), *Proceedings of the 5th International Conference on Autonomous Agents*, Montreal, Kanada, 2001.
- [24] M. Anthony Lewis. Gait adaptation in a quadruped robot. *Autonomous Robots*, 12(3):301–312, 2002.
- [25] Hod Lipson, Jordan B. Pollack. GOLEM@home-Webseite. Design and Manufacture of Robotic Lifeforms, <http://demo.cs.brandeis.edu/golem/>, 2000.
- [26] Martin Löttsch, Joscha Bach, Hans-Dieter Burkhard, Matthias Jünger. Designing agent behavior with the extensible agent behavior specification language XABSL. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence, Heidelberg, 2004. Springer. Im Erscheinen.
- [27] Robert B. McGhee, Andrew A. Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3(3/4):331–351, 1968.
- [28] Stefano Nolfi, Dario Floreano. *Evolutionary Robotics*. MIT Press, 2000.
- [29] Stefano Nolfi, Dario Floreano, Orazio Miglino, Francesco Mondada. How to evolve autonomous robots: Different approaches in evolutionary robotics. In Rodney A. Brooks, Patti Maes (Herausgeber), *Artificial Life IV*, Seiten 190–197, 1994.
- [30] Michael J. Quinlan, Stephan K. Chalup, Richard H. Middleton. Techniques for improving vision and locomotion on the sony aibo robot. In Jonathan Roberts, Gordon Wyeth (Herausgeber), *Proceedings of the 2003 Australasian Conference on Robotics and Automation*, Brisbane, Australien, 2003.
- [31] Thomas Röfer. An architecture for a national robocup team. In Gal A. Kaminka, Pedro U. Lima, Raúl Rojas (Herausgeber), *RoboCup 2002 Robot Soccer World Cup VI*, Nummer 2752 in Lecture Notes in Artificial Intelligence, Seiten 417–425, Heidelberg, 2003. Springer.

- [32] Thomas Röfer. Evolutionary gait-optimization using a fitness function based on proprioception. In *8th International Workshop on RoboCup 2004 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence, Heidelberg, 2005. Springer. Im Erscheinen.
- [33] Thomas Röfer, Ingo Dahm, Uwe Düffert, Jan Hoffmann, Matthias Jüngel, Martin Kallnik, Martin Löttsch, Max Risler, Maximilian Stelzer, Jens Ziegler. GermanTeam 2003. In *7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences)*, Lecture Notes in Artificial Intelligence, Heidelberg, 2004. Springer. Ausführlicher in <http://www.tzi.de/kogrob/papers/GermanTeam2003.pdf>. Im Erscheinen.
- [34] The RoboCup Federation. RoboCup-Webseite. <http://www.robocup.org>, 2004.
- [35] Greg Welch, Gary Bishop. An introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, 1995. 2003 aktualisiert.
- [36] Matthias Werner, Helmut Myritz, Uwe Düffert, Martin Löttsch, Hans-Dieter Burkhard. HumboldtHeroes. In Tucker Balch, Gerhard Kraetzschmar (Herausgeber), *RoboCup 2000: Robot Soccer World Cup IV*, Nummer 2019 in Lecture Notes in Artificial Intelligence, Seiten 651–654, Heidelberg, 2001. Springer.
- [37] Krister Wolff, Peter Nordin. Evolution of efficient gait with humanoids using visual feedback. In *Proceedings of the 2nd IEEE-RAS International Conference on Humanoid Robots, Humanoids 2001*, Seiten 99–106. IEEE, 2001.

B Danksagung

Hiermit möchte ich mich bei allen bedanken, die mir direkt oder indirekt bei der Erstellung dieser Diplomarbeit geholfen haben. Ich danke besonders:

- Prof. Hans-Dieter Burkhard dafür, den RoboCup an die Humboldt-Universität gebracht und diese Diplomarbeit betreut zu haben
- dem RoboCup für jahrelange Inspiration und Motivation
- dem GermanTeam, insbesondere Matthias Jünger, Martin Löttsch, Max Risler und Thomas Röfer, für fleißige Mitarbeit an der gemeinsamen Software
- Jan Hoffmann für die Betreuung dieser Arbeit
- Andreas Kunert, Mechthild Düffert, Elisabeth Ludewig und Janine Kuhn fürs Korrekturlesen

Erklärung

Ich, Uwe Düffert, erkläre hiermit, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ich bin damit einverstanden, dass die Humboldt-Universität zu Berlin diese Arbeit öffentlich zugänglich macht, zum Beispiel in ihren Bibliotheken.

Berlin, den 22. Juli 2004