ABC-Learning: Ein Lernverfahren zur modellfreien Selbstexploration autonomer Roboter.

Diplomarbeit

zur Erlangung des akademischen Grades
Diplominformatiker

HUMBOLDT-UNIVERSITÄT ZU BERLIN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
INSTITUT FÜR INFORMATIK

eingereicht von: Stefan Bethge
geboren am: 11. 12. 1984
in: Berlin

Gutachter/-innen: Prof. Dr. Hans-Dieter Burkhard
Prof. Dr.-Ing. Beate Meffert
Betreuer: Dr. Manfred Hild

eingereicht am: 16. Juni 2014 verteidigt am: 24. November 2014

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit dem Attractor-Based Behavior Control Learning oder kurz ABC-Learning, einem unüberwachten Lernverfahren für autonome mobile Roboter. Das ABC-Learning versetzt Roboter mit beliebiger Morphologie in die Lage, den eigenen Bewegungsspielraum ohne jegliches Vorwissen selbstständig zu erkunden und dabei strukturiertes Wissen zu speichern. Die Erkundung basiert auf den Eigenschaften des physikalischen Systems und findet tendenziell energetisch günstige Bewegungen und Positionen. Die Arbeit führt die notwendigen theoretischen Grundlagen für das Verfahren auf und beschreibt jeweils Besonderheiten der Implementation in einer für diese Arbeit geschriebenen Simulationsumgebung und auf einem mobilen Roboter, der der Simulation zugrunde liegt. Es wird untersucht, welche Auswirkungen verschiedene vorgeschlagene Entscheidungsheuristiken für bestimmte Situationen während der Exploration auf die Qualität des Lernfortschritts haben. Zudem werden Erweiterungen beschrieben, die das gespeicherte Wissen kompakt halten.

Abstract

The present thesis deals with the Attractor-Based Behavior Control Learning or short ABC-Learning, an unsupervised learning method for autonomous mobile robots. The method enables robots with arbitrary morphology to independently explore their own range of motion without any prior knowledge and thereby generating structured data about it. The exploration is based on properties of the physical system and finds movements and positions that tend to be energetically favorable. The work provides the necessary basis for the method and describes the specifics for implementations in a simulation environment written for this thesis and on a mobile robot that the simulation is based on, respectively. For various proposed decision heuristics for certain situations during the exploration, it henceforth examines the impact on the quality of the learning progress. In addition, extensions are described that keep the stored knowledge compact.

Inhaltsverzeichnis

1. Einleitung

In den letzten Jahrzehnten haben sich viele technische Entwicklungen, insbesondere Feinmechanik und Antriebstechnik, Batterietechnik, leistungsfähige Elektronik und Rechentechnik als Grundlage für die mobile autonome Robotik soweit entwickelt, dass robotische Systeme nicht mehr nur in Laborsituationen denkbar sind, sondern diese auch für tatsächliche Einsätze, z. B. zur Erkundung anderer Planeten oder zur Katastrophenhilfe in Fukushima bereit sein könnten. Gleichzeitig ist jedoch der Fortschritt in der Entwicklung von Software, die eine Eigenständigkeit bei der Bewegung und ein einfaches mehr oder weniger intelligentes Verhalten ermöglicht, ungleich langsamer vorangeschritten. Meist wird die Umgebung vom Menschen verstanden und interpretiert während Roboter daraufhin sehr eingeschränkte Ziele für den Menschen erfüllen sollen, ohne selbst die Fähigkeit zu besitzen, ihr Handeln in der Umwelt zu verstehen.

Die natürliche Umwelt, besonders in Situationen, die selbst für Menschen nicht zugänglich sind, bietet beliebig komplexe Umgebungen und Hindernisse. Zudem verändert sie sich möglicherweise kurzfristig, und auch der Roboter selbst kann sich durch sie verändern. Erst wenn Roboter in der Lage sind, die Umwelt selbst zu erfahren und mit den so gewonnen Erkenntnissen Entscheidungen zu treffen, sind sie flexibel und anpassungsfähig genug, um nicht in einfach erscheinenden Situationen auf menschliche Hilfe und Intelligenz angewiesen zu sein. Die Konzepte Erfahrung und Erkenntnis sind bisher durch Berechnungen nicht adäquat fassbar geworden. Die moderne kognitive Robotik hat daher von der Herangehensweise der traditionellen *Artificial Intelligence* (AI) Abstand genommen, die die Intelligenz natürlicher Organismen als eine Vielzahl von Algorithmen sieht, die gesammelte Informationen über die Welt zuerst innerhalb von abstrakten Modellen verrechnen und aus welchen dann Schlussfolgerungen abgeleitet werden und mit welchen Planung für späteres Verhalten erfolgt.

Spätestens seit Rodney Brooks' einflussreichem Aufsatz "Intelligence without representation" (Brooks, 1991) gilt es, Modelle möglichst zu vermeiden und die Welt selbst als ein Modell zu verwenden. Nicht zuletzt, weil wir nicht wissen, welche internen Modelle von unserem Handeln und Denken wir Menschen durch unsere Selbstwahrnehmung sehen und erschaffen und welche Modelle unser Gehirn tatsächlich benutzt. Aufgrund der langen Zeit, die die Evolution benötigt hat, einfache Mobilität, Reaktivität und andere überlebenswichtige Fähigkeiten zu entwickeln und der damit verglichen kurzen Zeit, viele weitere komplexe

Fähigkeiten von Lebewesen und dem Menschen herauszubilden, sieht Brooks als Basis für Intelligenz vor allem die Fähigkeit, mit der Welt zu interagieren und sich in ihr zu bewegen. Bevor Roboter in der Lage sind, die Auswirkungen ihrer motorischen Aktionen auf den Zustand des eigenen Körpers und in einem späteren Schritt auch auf die Umwelt zu beobachten, zu erwarten und zu eruieren, ist es für die Entwicklung von intelligenteren Robotern nicht zielführend, komplexe modellbasierte Fähigkeiten zu entwickeln, die meist nur in isolierten Situationen funktionieren und mangels dieser Universalität auch nicht zusammen mit anderen Fähigkeiten zu neuem Verhalten führen können. Es ist also notwendig, sich zuerst auf die grundlegendsten motorischen und sensorischen Fähigkeiten zu konzentrieren.

Aus der Kognitionswissenschaft kommend, hat das Konzept des *Embodiment* zur *embodied AI* geführt, wonach Bewusstsein, wie wir es kennen, nur innerhalb eines Körpers entstehen und existieren kann. Das für physische Interaktion mit der Welt notwendige Wissen über die Umwelt ist nur durch das Erleben mittels eines realen Körpers möglich, denn jeder Körper erlangt sein eigenes Weltwissen und benötigt seine eigenen, speziell angepassten Aktionen. Die bereits in der Antike gedachte Trennung in Körper und Seele, in Hardware und Software, gerät dadurch in Zweifel. Das Gehirn ist weniger ein zentrales Steuersystem, dem die Körperfunktionen gehorchen als vielmehr ein Netzwerk, das die sensorischen Signale mit den motorischen Ausgängen verbindet und in dem die Prozesse innerhalb des Körpers über hochkomplexe Verschaltung mit den Prozessen außerhalb des Körpers in Verbindung stehen (Pfeifer und Bongard, 2006).

Gleichzeitig kann man sich der Frage nach den Grundlagen für intelligentes Verhalten und Bewusstsein nähern, wie es etwa Dietrich Dörner tut. Er konstruiert auf dem Papier (Dörner, 2001) und in einer Simulation (Dörner, 2002) ein technisches Lebewesen, das die Umwelt wahrnimmt, Bedürfnisse hat und diese befriedigen möchte. Später bilden sich daraus einfache Formen von Sozialverhalten und Moral auf dieser Grundlage heraus. Durch die gezielte parallele Beschreibung einiger der Fähigkeiten natürlicher Lebewesen macht er eine Vorstellung davon möglich, wie unser Bewusstsein auf der Basis einfacher perzeptiver Fähigkeiten entstanden sein könnte.

Betrachtet man die Entwicklung von Primaten, die erst in ihren ersten Lebensjahren die Fähigkeit entwickeln, zu tasten, zu interagieren und sich fortzubewegen, liegt die Annahme nahe, dass – anders als beispielsweise bei Insekten, die mit all ihren Fähigkeiten aus dem Ei schlüpfen – komplexe Bewegungsmuster interaktiv erlernt werden und damit nicht nur feste und unveränderliche Bewegungen sind, sondern vielmehr in jeder Situation neu ausprobiert und angepasst werden. Darüber hinaus gibt es nicht nur Entwicklungsschritte in den kognitiven und sensorischen Fähigkeiten der einzelnen Individuen sondern auch in der morphologischen Beschaffenheit. Der Mensch ist in seinen ersten Wochen und Monaten nicht in der Lage, die Kraft aufzubringen, die ihm später das Krabbeln und Laufen ermöglicht. Er ist zudem klein und gut gepolstert, um während des Ausprobierens verschiedenster Aktionen möglichst nicht Gefahr zu laufen, sich zu verletzen. Nicht zuletzt steht einem Kleinkind im

Idealfall immer ein beschützendes Elternteil zur Seite, was ebenfalls zu den Voraussetzungen der Entwicklung unserer Fähigkeiten gezählt werden muss. Auch nach der Kindheit sind viele Tiere noch in der Lage, ihre Fähigkeiten immer wieder an die Umwelt und an neue Herausforderungen anzupassen. Zudem können Verletzungen und Altersbeeinträchtigungen die Form und den Aktionsspielraum des Körpers auch nach dem Auswachsen noch beträchtlich ändern. Es muss also ein andauernder Lern- und Adaptionsprozess auf verschiedenen Zeitskalen stattfinden, der bei hoher Flexibilität auf nur wenige Konstanten setzt.

1.1. Motivation und Zielstellung

Für die Entwicklung der Fähigkeiten autonomer Roboter bedeutet das, dass es Erfolg versprechender sein kann, zuerst mit einer einfachen und robusten Morphologie selbstständig Erfahrungen zu sammeln, anstatt isolierte und vergleichsweise komplexe Fähigkeiten als menschlicher Architekt der “Roboterseele” einzupflanzen. (Im Abschnitt 4 wird die in dieser Arbeit verwendete reduzierte Robotermorphologie *Semni* vorgestellt.) Die Verfahren sollten dabei möglichst von der Form und der Dimensionalität der Freiheitsgrade unabhängig sein. Menschen wie Tiere können sich zum Beispiel auch nach dem vollständigen Verlust eines Beines weiterhin fortbewegen, jedoch mit dann deutlich anderen Bewegungen. Das Entscheidende zum Erreichen einer Vorwärtsbewegung ist dabei aber weniger, wie genau mit welchen Gliedmaßen welche Muskelkontraktionen ausgeführt werden, sondern wie der Prozess des Laufens durch Gewichtsverlagerung und alternierende Belastung von verschiedenen Stützstellen, beispielsweise den Füßen und Händen, eine möglichst energieeffiziente und stabile Bewegung erzeugt. Abhängig von der eigenen Konstitution aber auch der Situation und Aufgabe kann es so zu den verschiedensten Fortbewegungsarten wie Kriechen, Robben, vierbeinigem Klettern an Berghängen oder zweibeinigem Laufen mit Unterstützung durch einen Stock kommen. Eine solche Flexibilität ist bisher künstlich nicht erreicht.

Künstliche informationsverarbeitende Systeme verwenden häufig Algorithmen, mit denen Reaktion und Aktionen berechnet und erzeugt werden. Dabei ist es fast immer notwendig, das gewünschte Verhalten dieser Algorithmen durch eine mehr oder weniger große Anzahl von Parametern festzulegen. Ein Anpassen der Parameter von Hand durch den Entwickler nach Erfahrungswerten kann in einfachen Fällen ausreichen, ist aber ab einem bestimmten Komplexitätsgrad nicht mehr praktikabel. Alternativ können diese Parameter berechnet werden, allerdings kann das nicht immer analytisch geschehen – insbesondere wenn die Systeme, in denen die Algorithmen arbeiten, nicht vollständig bekannt oder zu komplex sind. Außerdem bedeutet die Berechnung, dass sich der Konstrukteur erneut Modelle für bestimmte Teile der Welt überlegen muss, aus denen sich ideale Parameter ergeben. Eine weitere Möglichkeit ist es, die Parameter über die Zeit mit dafür geeigneten Lernverfahren im System anzupassen und dabei zu verbessern. Verbessern heißt in der Praxis meist, dass die Resultate

nach einer Parameteränderung mehr einem bestimmten gewünschten Ziel oder Verhalten entsprechen sollen. Es müssen also die beobachteten Änderungen der Ausgabegrößen auf eine sinnvolle Veränderung der Parameter zurückgeführt werden. Beispielhafte Verfahren dafür sind *Backpropagation*, *Reinforcement Learning* oder *Simulated Annealing*.

Ein Problem dieser Verfahren ist es häufig, dass am Anfang noch kein Vorwissen vorhanden ist und damit Beschädigung von Gelenken und anderen Teilen des eigenen Körpers möglich ist, die durch zu schnelle und zu kraftvolle Bewegungen erfolgen. Lässt man alle Möglichkeiten für die entstehende Struktur offen, ist der Suchraum schnell so groß, dass in absehbarer Zeit nur ein kleiner Teil der Möglichkeiten ausprobiert und gelernt werden kann (vgl. Bernstein, 1967). Dies ist nach Richard E. Bellman auch als *Curse of Dimensionality* bekannt. Zwar können z. B. beim Reinforcement Learning oder der künstlichen Evolution mit geeigneten Fitnessfunktionen und Beschränkungen des Suchraumes sehr gute Ergebnisse erreicht werden, diese müssen aber für verschiedene Zielstellungen geschickt formuliert werden und die resultierenden Strukturen sind danach schwer zu analysieren und auf geänderte Gegebenheiten anzupassen.

Desweiteren ist es eine häufige Vorgehensweise, die ersten Schritte ohne die Gefahr der Beschädigung zu gehen, indem die Verfahren mit simulierten Modellen der Zielplattform durchgeführt werden, um so die Regler erst auf den echten Roboter zu übertragen, wenn sie in der Simulation ihre Aufgabe erfüllen. Nahezu unabhängig von der Komplexität der Hardware zeigt sich aber, dass Simulationen die echte Welt und ihre komplexe Physik immer nur angenähert abbilden können.[1] Daraus ergibt sich, dass Simulationen zwar durchaus erfolgreich genutzt werden können, um mit ihnen neue Prinzipien auszuprobieren, Morphologien zu verändern ohne sie wirklich bauen zu müssen und Strukturen zu generieren, bei denen exakte Übereinstimmung des Verhaltens nicht unbedingt entscheidend ist. Für die Feinabstimmung von vielen Parametern mittels Lern- und Optimierungsverfahren für komplexe dynamische Systeme (vgl. Abschnitt 3) sind die Unterschiede aber in vielen Fällen zu groß, so dass Regler für dynamische Bewegungen im Allgemeinen aus Simulationen nicht ohne weitere Anpassung auf echte Systeme übertragen werden können und ein “lebenslanges” Lernen ohnehin auf der Hardwareplattform notwendig ist.

In dieser Arbeit soll nun das Lernverfahren *Attractor-Based Behavior Control* oder ABC-Learning (Hild, 2011) beschrieben und erweitert werden, mittels dessen ein Roboter mit beliebiger Morphologie in die Lage versetzt wird, den eigenen Bewegungsspielraum ohne jegliches Vorwissen selbstständig zu erkunden und dabei strukturiertes Wissen zu speichern. Die Erkundung basiert auf den Eigenschaften des physikalischen Systems und findet ten-

[1] Bereits ein einfaches schwingendes Doppelpendel zeigt chaotisches Verhalten, bei dem die Schwingungstrajektorie stark von den Anfangsparametern abhängt. Ein echtes Doppelpendel und das Schwingungsverhalten in einer Simulation exakt nachzubilden und die gleiche Trajektorie zu berechnen ist, wenn überhaupt, nur für eine kurze Zeitspanne möglich.

denziell energetisch günstige Bewegungen und Positionen, die damit die “echte Welt” als Modell verwenden. Die gelernten Bewegungen können während der Laufzeit des Roboters fortlaufend mit neuen Erfahrungswerten verbessert und angepasst werden, so dass die Struktur auch individuellen Veränderungen z. B. durch Alterung und Abnutzung des Roboters jederzeit angepasst ist. In späteren Entwicklungsschritten können Bewegungen mit dem bereits gelernten Vorwissen schneller und gezielter ablaufen und dabei wiederum neue Informationen liefern. Das Verfahren erlaubt unter Verwendung von propriozeptiven Sensoren die Erkundung der physikalischen Eigenschaften des eigenen Körpers in der Interaktion mit der Umwelt und versucht so, die Fähigkeit zur sensomotorischen Selbstorganisation als ersten Schritt zu eigenständigem Verhalten aufzubauen.

1.2. Aufbau der Arbeit

Nach einem Überblick über angrenzende vorangegangene Arbeiten im Bereich der autonomen explorativen Lernverfahren für Roboter, werden im Folgenden physikalische und theoretische Grundlagen dargelegt (Kapitel 3), ein im Rahmen dieser Arbeit entwickelter Simulator (Kapitel 4) und die dafür nötigen physikalischen Grundlagen beschrieben (Abschnitt 4.1). Das Lernverfahren und verwendete Motorreglerstrukturen werden in Kapitel 5 vorgestellt, die Implementationen des Lernverfahrens innerhalb der Simulationsumgebung und separat für den Embedded-Prozessor der realen Roboterplattform *Semni* darauf in Kapitel 6 beschrieben. Es folgt ein Vergleich der Umsetzung des Lernverfahrens im Simulator und auf der Hardware (Abschnitt 6.3). Nach der Beschreibung von vergleichenden Experimenten mit dem Lernverfahren, werden die Resultate dieser ausgewertet und gegenübergestellt (Kapitel 7). Die Zusammenfassung in Kapitel 8 gibt einen abschließenden Überblick und beschreibt innerhalb der Architektur mögliche weiterführende Arbeiten und Ansätze, die nicht mehr im Rahmen dieser Arbeit Platz gefunden haben.

2. Stand der Forschung

In den Arbeiten von Christensen et al. (2009) werden modulare Roboter verwendet, welche aus mehreren identischen Modulen bestehen, die manuell zusammengesetzt werden oder welche während der Laufzeit auch selbstständig vom Roboter neu konfiguriert werden können. Dafür wurde ein lebenslanges Lernverfahren angestrebt, dass z. B. zu Laufbewegungen führen kann, ohne die Regler von Hand gestalten zu müssen. Das Verfahren benutzt kein Modell der Morphologie und benutzt ähnlich wie das ABC-Learning ausschließlich lokale voneinander unabhängige Regler, welche während der Laufzeit auf dem Roboter angepasst werden. Dafür wird für jeden Regler einzeln ein einfaches Lernen fester Aktionen durch bestärkendes Lernen (Q-Learning) von Winkeltabellen vollzogen, welches durch beschleunigende Heuristiken erweitert wird. Als gemeinsamer Belohnungswert wird die Geschwindigkeit der Laufbewegung verwendet. Anders als bei vergleichbaren Arbeiten ist hier die Morphologie aber unbekannt, so dass auch die Regler ohne Vorannahmen gelernt werden.

Bei Maes und Brooks (1990) werden diskrete Aktionen, z. B. die Beine des betrachteten 6-beinigen Roboters vorwärts oder rückwärts zu setzen, zusätzlich zu beobachteten Bedingungen in jedem Zeitschritt binär mit positivem und negativem Feedback versehen. Die Korrelation des Feedbacks mit dem Zustand der einzelnen Aktionen, also ob diese aktiv waren oder nicht, wird als Maß dafür ermittelt, wie sehr die Aktion an positivem oder negativem Feedback beteiligt ist. Weiterhin wird als Relevanz einer Aktion die Differenz zwischen der Korrelation für positives Feedback und der für negatives Feedback ermittelt. Damit wird für jedes Verhalten in einer Situation bestimmt, wie wahrscheinlich es ist, ob ein Verhalten aktiv wird. Als weiteres Maß werden die Zuverlässigkeit der Aktion bestimmt und wie sehr eine Aktion Feedback generiert, wenn gleichzeitig eine der vorhandenen Bedingungen zutrifft. Es werden mit diesen Maßen in verschiedener Gewichtung Aktionen ausgewählt, die aktiviert werden, der Rest wird nicht aktiviert. Damit können voneinander unabhängige Aktionen parallel exploriert werden und es kann ähnlich wie beim ABC-Learning kooperatives Verhalten entstehen, ohne dass dieses durch interne Kommunikation koordiniert wird. Durch die Wahl z. B. bestimmter Sensoren als Feedback-Werte wird durch den Menschen indirekt ein Ziel vorgegeben, auf das hin optimiert wird. Im Beispiel wird ein 6-beiniges Laufen erzielt, das von allein symmetrische Aktionsfolgen findet, bei denen immer drei Beine auf dem Boden verbleiben.

Demiris und Dearden (2005) bestimmen ein kinematisches Vorwärtsmodell durch Selbstexploration mit “Motor Babbling”, also zufälliger Motoransteuerung, das durch visuelle Selbstwahrnehmung ein Bayesian Belief Network trainiert. Gleichzeitig können aber andere Individuen imitiert werden indem deren Aktionen beobachtet werden. Durch Invertierung dieses Modells kann dieses im Anschluss genutzt werden um Aktionen auszuführen oder wahrzunehmen und zu erkennen.

Mittels intrinsisch motiviertem Lernen wird in Baranès und Oudeyer (2010) für redundante Systeme eine inverse Kinematik explorativ gelernt indem Motor Babbling im Arbeitsraum statt im Konfigurationsraum angestrebt und damit der Suchraum erheblich verkleinert wird. Es werden vom Roboter selbstständig Ziele heuristikbasiert generiert, deren Erreichen das Lernen möglichst noch unbekannter Informationen ermöglicht. Dabei werden kleine Winkeländerungen für jedes Gelenk vorgenommen und anhand der Änderung die Jacobimatrix lokal bestimmt, mit welcher sich die inverse Kinematik approximativ berechnen lässt. Die Exploration muss bei n Gelenken jedoch in jedem besuchten Punkt des Zustandsraumes aus $2n$ verschiedenen möglichen Aktionen lernen. Im Vergleich zum ABC-Learning, das bereits auf realer Hardware getestet wurde, sind hier nur Simulationsergebnisse vorhanden und der Fokus liegt vorerst nicht auf der Exploration bestimmter Aktionen sondern auf statischen Bewegungen zu bestimmten Punkten im Arbeitsraum.

Ralf Der und Georg Martius entwickelten eine Reihe von Prinzipien zur autonomen Exploration ohne Zielstellung, vor allem die Homöokinese (Der et al., 1999; Der und Martius, 2006; Der et al., 2012). Diese hat im Gegensatz zur Homöostase das Ziel, das System in einer ständigen jedoch nicht zufälligen Bewegung zu halten. Es wird die Perspektive eingenommen, dass Verhalten nicht durch externe Vorgaben motiviert werden soll sondern immer von einem internen Drang erzeugt werden muss. Diese interne Perspektive benötigt daher ein Weltmodell, das sich ausschließlich innerhalb des Sensorraumes befindet. Dabei wird der Fehler zwischen Vorhersage durch das Weltmodell und dem wirklichen Verhalten durch Anpassung des Reglers über einen Gradientenabstieg minimiert. Es wird vom Zustand des tabula rasa ausgegangen, das heißt es gibt kein Vorwissen über den Körper oder die Welt und keine künstlichen Beschränkungen. Das Verfahren baut selbstständig ein Weltmodell auf, um damit Vorhersagen aus den vorherigen Sensordaten treffen zu können und ist in diesen Eigenschaften dem ABC-Learning ähnlich. Im Unterschied zum ABC-Learning fehlt jedoch ein zielgerichtetes Ansteuern von zuvor besuchten Konfigurationen.

Das Weltmodell wird als kognitive Fähigkeit interpretiert, also als Wahrnehmung der Welt und des Roboters selbst. Motorische Aktivität wird zu Anfang noch durch zufällige Bewegungen erzeugt, die dann durch zielgerichtete Aktionen weitergeführt werden. Diese werden so gestaltet, dass sie einerseits hohe Sensitivität gegenüber den Sensorwerten besitzen, gleichzeitig jedoch noch eine gewisse Vorhersagbarkeit behalten. Die Parameter für die Gelenkaktoren und das Weltmodell werden im Gegensatz zum ABC-Learning gleichzeitig gelernt, wodurch von Anfang an hochdynamische Bewegungen entstehen können. Diese

Prinzipien haben wie das in dieser Arbeit beschriebene Verfahren ebenfalls zum Ziel, die eigenständige Entwicklung von morphologieunabhängigen sensomotorischen Verhaltensweisen zu ermöglichen. Das ABC-Verfahren ist jedoch besonders für modulare Roboter mit einer verteilten und voneinander unabhängigen Rechenleistung geeignet, was aufgrund der verwendeten Matrixinversion hier nicht möglich ist. Problematisch ist zudem, dass konkretes Verhalten kaum als Struktur abgespeichert und später stabil reproduziert werden kann. In der ebenfalls im Labor für Neurorobotik entstandenen Diplomarbeit von Matthias Markl wird ein Explorationsverfahren auf Basis der Homöokinese auf dem humanoiden Roboter Myon experimentell untersucht und ausgewertet (Markl, 2010).

3. Theoretische Grundlagen

Im Folgenden wird ein kurzer Überblick über die Theorie dynamischer Systeme gegeben, die als geeignetes Werkzeug zur Analyse von Bewegungen physikalischer Systeme als Grundlage für das ABC-Learning herangezogen wird und es werden die hier wichtigen Begriffe definiert. Dabei wird das Gesamtsystem aus Roboter, Reglerstrukturen und seiner Umwelt als dynamisches System verstanden. Darauf folgend werden für das Lernverfahren eingesetzte Regelmechanismen zur Aktuierung der Gelenke von Robotern illustriert und deren Bezug zu dynamischen Systemen hergestellt und abschließend mit Beispielen verdeutlicht.

3.1. Dynamische Systeme

Ein *dynamisches System* ist nach Meiss (2007) und Broer et al. (2010) ein Prozess, dessen Zustand sich über die Zeit verändert und ist definiert als Tupel (T, X, f) mit dem Zeitraum T, dem Zustandsraum $X \subset \mathbb{R}^n$ und einer Funktion f. Ein Zustandsvektor hat eine solche Dimensionalität, dass er den Zustand des Systems vollständig und eindeutig zu einem Zeitpunkt beschreibt. Es ist im Kontext von mechanischen Systemen sinnvoll, den Zustandsraum als Mannigfaltigkeit $M \subset \mathbb{R}^n$ zu verstehen.

Sei $x : T \to X$ und $x(t)$ der Zustand des Systems zum Zeitpunkt t. Ist $T = \mathbb{N}_0$ oder $T = \mathbb{Z}$, die Zeit also diskret, dann wird durch

$$f : X \to X,\ x(t+1) := f(x(t)), t \in T \tag{3.1}$$

ein Zustandsübergang von Zeitschritt t zum Zeitschritt $t+1$ definiert. Ist andernfalls $T = \mathbb{R}$ und das System damit zeitkontinuierlich, dann gibt die Funktion f einen Fluss $f : X \to X$ an, der die Dynamik mit der Differentialgleichung $\dot{x}(t) = f(x(t))$ als Vektorfeld beschreibt. Der Zustandsraum wird auch Phasenraum genannt (Arrowsmith und Place, 1994). Da es im Weiteren um Roboter mit diskret abgetasteten Sensorwerten handelt, werden sich die Betrachtungen aber auf diskrete dynamische Systeme beschränken.

Mit der *Konfiguration* sei ein Tupel bezeichnet, das die Gelenkwinkel φ_i aller Gelenke eines Roboters enthält. Die Menge aller möglichen Konfigurationen wird als *Konfigurationsraum* bezeichnet, welcher ein Unterraum des Zustandsraumes ist.

Eine *Trajektorie O* ist eine Teilmenge des Zustandsraumes und ist die Menge aller Zustände, die das System von einem Startzeitpunkt t_0 bis zu einem Endzeitpunkt t_n durchfährt,

$$O := \{x(t_0), x(t_1), \ldots, x(t_n)\}, x(t_n) = f^n(x(t_0)),\ n \in \mathbb{Z}, t_n \in T\,. \tag{3.2}$$

Dabei sei $f^m(x) := \underbrace{f \circ f \circ \cdots \circ f(x)}_{\text{m-Mal}}$ die m-te Iterierte von f.

Eine Trajektorie, dessen Startpunkt nach p Zeitschritten erneut besucht wird, nennt man *periodischer Orbit* oder *Periode-p-Orbit*:

$$O_p := \{x(t_0), x(t_1), \ldots, x(t_p)\}, \text{mit } x(t_0) = x(t_p) \tag{3.3}$$

Für das in dieser Arbeit beschriebene Lernverfahren sind insbesondere *Fixpunkte* von Interesse. Ein Fixpunkt ist ein Punkt $p \in X$, für den gilt $p = f^n(p),$ für alle $n \geq 0$. Diese können weiter in stabile und instabile Fixpunkte unterschieden werden. Ein stabiler Fixpunkt ist ein *Attraktor* und besitzt ein *Basin* $B \subseteq X$, so dass für jeden Startpunkt $b \in B$ für $t \to \infty$ das System in den Fixpunkt gelangt. Existiert dieses Basin nicht, so ist der Fixpunkt instabil.

Allgemeiner ist ein Attraktor A eine Untermenge des Zustandsraumes, für die gilt

- A ist f-invariant, also

$$x \in A \Rightarrow f(x) \in A, \tag{3.4}$$

- A besitzt eine offene Umgebung W_A^+ für die ein $n_k \in \mathbb{N}^+$ existiert, so dass

$$W_A^+ = \{x \in X | \forall n > n_k : f^n(x) \in A\}, \tag{3.5}$$

  welche Basin genannt wird und eine stabile Menge ist,

- A ist minimal, es existiert also keine echte Teilmenge von A, die die ersten beiden Bedingungen erfüllt. (Milnor, 2006)

Analog zu Attraktoren kann ein dynamisches System einen Repellor besitzen (oder auch mehrere Repelloren), innerhalb dessen das System verbleibt. Anders als bei einem Attraktor wird das System jedoch bei der kleinsten Störung aus dem Repellor und von ihm weg getragen. Ein Repellor besitzt im Gegensatz zum Attraktor eine Umgebung, $W_R^- = \{x \in X | \forall n > n_k : f^- n(x) \in R\}$, das heißt x ist rückwärts asymptotisch zum Repellor R.

Fixpunkte können auch als p-Orbit der Periodenlänge eins betrachtet werden. Desweiteren existieren quasiperiodische Orbits, die eine periodische Struktur besitzen. Deren Punkte liegen immer sehr dicht beieinander, stimmen aber nie exakt mit einem der vorherigen überein.

Die Trajektorie, die sich zwischen zwei Fixpunkten ergibt, wenn der Zustand sich von einem Fixpunkt entfernt und gleichzeitig auf einen stabilen Fixpunkt zubewegt, wird als *homokliner* bzw. *heterokliner* Orbit bezeichnet, je nachdem ob der Start- und Endzustand gleich oder verschieden sind (Siehe dazu auch Abbildung 5.1).

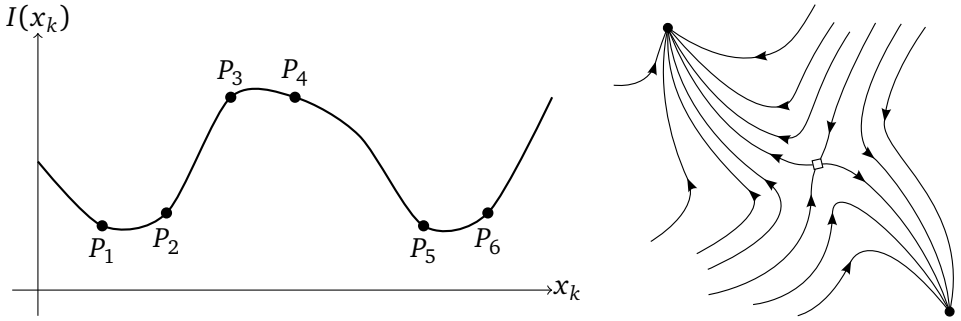

Abbildung 3.1.: Exemplarische dynamische Systeme. Links: stabile und instabile Fixpunkte entlang der beliebigen Komponente x_k des Zustandsraumes und dem Energieniveau $I(x_k)$. Rechts: zwei stabile Fixpunkte und zugehörige Trajektorien innerhalb ihrer Basins. Der mittlere Punkt ist ein *Sattelpunkt*, an dem der Phasenraum in eine stabile und eine instabile Untermannigfaltigkeit zerlegt werden kann. Die Trajektorien die zum Sattelpunkt führen, gehören zu dessen stabiler Untermannigfaltigkeit und bilden eine *Separatrix*, die die Basins der beiden stabilen Fixpunkte voneinander trennt. Über die heteroklinen Orbits kann der Zustand des Systems von instabilen zu stabilen Fixpunkten wandern.

Wird einer der Parameter des Systems verändert, kann sich die Attraktorlandschaft grundlegend ändern und beispielsweise aus einem stabilen Fixpunkt ein instabiler Fixpunkt werden. Gleichermaßen können Attraktoren verschwinden oder neue entstehen. Solche Veränderungen des Systems werden *Bifurkation* genannt.

3.2. Einfache Regelschleifen

Eine zentrale Rolle in der Robotik spielt die Ansteuerung von Aktoren, mit denen erst Bewegungen und Verhalten möglich werden. Die Regelungstechnik, deren früheste Erfindung mit dem Fliehkraftregler von James Watt entstand, wird historisch der Kybernetik zugeordnet und beschäftigt sich inzwischen als Teilgebiet der Automatisierungstechnik mit den der Regelung beliebiger Größen zugrundeliegenden Prinzipien. Im Folgenden werden nur die grundlegendsten Begriffe genannt.

Eine *Regelungskette* besteht aus einer oder mehreren Eingangsgrößen von Sensoren, deren Verarbeitung und der Generierung von Ausgangsgrößen, die an die Aktoren weitergegeben

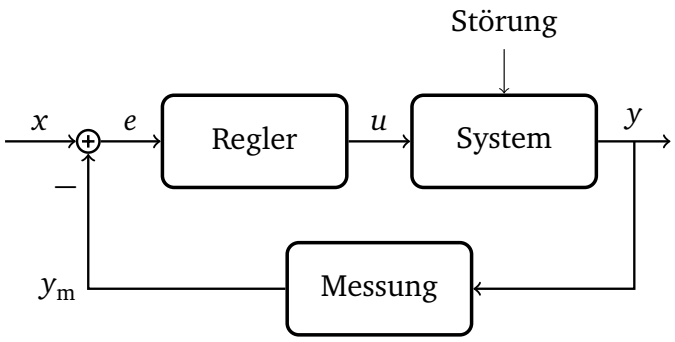

Abbildung 3.2.: *Einfacher Regelkreis*. Der Regelkreislauf besteht aus den Eingangsgrößen x (Sollgröße) und y_m (Istgröße), aus denen die Regelabweichung e und durch den Regler die Stellgröße u berechnet werden. Die Rückführung der Regelgröße y geschieht, nachdem diese eine Auswirkung im System erzielt hat und durch eine Messung als Messgröße y_m wieder am Eingang des Reglers anliegt.

werden. Die Ansteuerung oder Steuerung von veränderlichen Größen innerhalb eines Systems geschieht der Art des Systems entsprechend durch verschiedene Medien, beispielsweise durch Dampfdruck, elektrische Spannung oder durch die Ausschüttung von Neurotransmittern. Der Datenfluss ist vorwärtsgerichtet, die Sensoren können als Datenquelle und die Aktoren als Datensenke betrachtet werden.

In einem *Regelkreis* werden ein oder mehrere Sollwerte zusammen mit dem gemessenen Istwert verrechnet, im Normalfall wird die Differenz gebildet, die auch als Regelabweichung oder Fehler bezeichnet wird. Die Veränderung einer Stellgröße wird aus dieser Differenz mittels des *Reglers* bestimmt und an einen Aktor weitergeleitet, so dass der Istwert direkt oder indirekt beeinflusst wird. Der Sollwert wird damit sofort eingestellt oder graduell über die Zeit angenähert. Entscheidend für eine Regelung ist im Vergleich zur *Steuerung*, dass die tatsächliche Auswirkung der Ausgangsgrößen auf das System durch Sensoren gemessen und als neuer Istwert in den Regelkreis zurückgeführt wird. Es wird damit eine Rückkopplungsschleife erzeugt, die auch mit Störungen umgehen kann, deren Stabilität aber durch Wahl der geeigneten internen Strukturen und Parameter gewährleistet werden muss.

Eine häufige Reglerstruktur ist ein *proportionaler Regler* oder kurz p-Regler, der die Differenz zwischen Ist- und Sollwert mit einem Proportionalitätsfaktor multipliziert und zum Ausgang leitet. Dadurch arbeitet der Aktor so, dass der Fehler proportional zu seiner Größe verringert wird. Da sich so mit abnehmendem Fehler auch die Stärke der Aktivität des Aktors verringert, kann es zu einer bleibenden Regelabweichung kommen, bei der die Aktivität des Aktors nicht mehr ausreicht, den verbleibenden kleinen Fehler zu verringern.

Daraufhin ist es oft sinnvoll, den multiplizierten Fehler nicht nur direkt zum Ausgang zu geben sondern diesen zusätzlich über die Zeit zu integrieren bzw. im diskreten Fall gewichtet

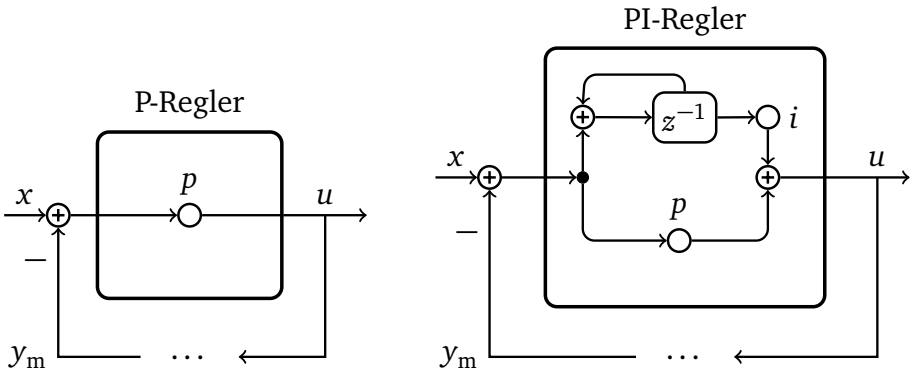

Abbildung 3.3.: Einfache diskrete Regelschleifen. Leere Kreise symbolisieren eine Multiplikation mit dem darüber angegeben Faktor, Kreise mit Plus eine Addition der eingehenden Werte. Ein Minus am Pfeil gibt einen Vorzeichenwechsel an.

aufzusummieren. Der resultierende *Proportional-Integral-Regler* oder kurz PI-Regler hat damit die Eigenschaft, die Ausgangsgröße solange zu vergrößern, wie ein Fehler besteht. Damit wird mit längerem Bestehen der Abweichung immer stärker gegen den Fehler gearbeitet. Die Zeitverzögerung durch die Integration kann abhängig von der Integrationskonstante und vor allem der Abtastfrequenz Selbstoszillationen und Überschwingen erzeugen und so das Gesamtsystem instabil werden lassen. Es ist daher notwendig, geeignete Parameter zu ermitteln, die das gewünschte Verhalten hervorrufen. Weitere Reglertypen und Verfahren zum Reglerentwurf finden sich z. B. bei Nise (2004).

3.3. Cognitive Sensorimotor Loops

Cognitive Sensorimotor Loops (Hild, 2011), im Folgenden mit CSL abgekürzt, sind Regelschleifen, welche zur dynamischen Ansteuerung eines Motors verwendet werden können und dabei kognitive Eigenschaften aufweisen (siehe Abbildung 3.4).

Es wird als Eingangsgröße lediglich ein relativer Sensorwert des Gelenkes benötigt. Für viele Anwendungen ist das die Winkelgeschwindigkeit. Die Vermeidung von absoluten Winkeln bei der Ansteuerung und Sensorik hat den Vorteil, nicht von einer oft aufwändigen genauen Kalibrierung abzuhängen. Ist nur ein absoluter Winkelsensor vorhanden, kann der linksseitige Differenzenquotient

$$\dot{\varphi}(t) \approx v(t) := \frac{\varphi(t) - \varphi(t - \Delta t)}{\Delta t} \tag{3.6}$$

berechnet werden, der die Ableitung des Weges nach der Zeit und damit die Geschwindigkeit $\dot{\varphi}(t)$ annähert. Dadurch werden jedoch die hochfrequenten Anteile des Signales verstärkt,

also genau das bei Sensoren unvermeidbare Rauschen (genaueres dazu in Kubisch, 2010, S. 50 ff.). Da die Reglerstruktur einen Integrator enthält, ist ein sonst vorgelagertes Tiefpassfilter jedoch nicht mehr nötig. Am Ausgang wird zur Ansteuerung die Motorspannung ausgegeben, welche einem Zieldrehmoment entspricht.

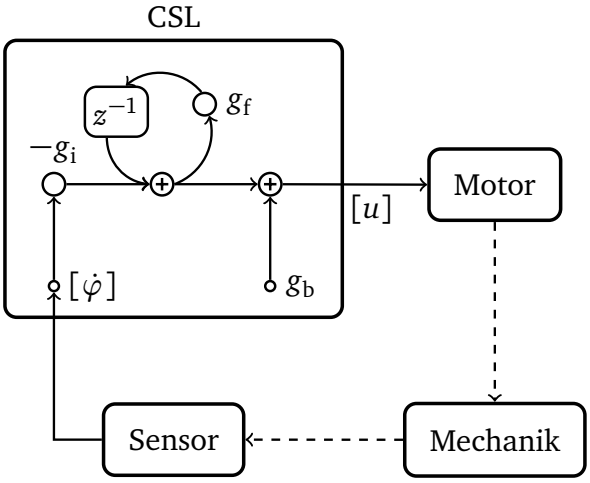

Abbildung 3.4.: *Cognitive Sensorimotor Loop*. Der Regelkreis besteht aus dem Regler, der einen Eingang (Sensor) und einen Ausgang (Motor) besitzt und den weiteren Bestandteilen Motor, Mechanik und Sensorik. Durch die Zeitverzögerung und den Faktor g_f wird eine Integration diskret angenähert, die ohne Veränderung aufsummiert ($g_f = 1$), leaky integriert bzw. ein Tiefpassfilter darstellt ($g_f < 1$) oder verstärkt ($g_f = 1+\epsilon$). Mittels des Biaswertes g_b kann eine konstanter Offset am Ausgang erzeugt werden. Insbesondere der Parameter g_f bestimmt das Verhalten der Regelschleife.

Die Berechnung des Ausgangs aus der Eingangsgröße geschieht wie folgt:

$$u(t) = -g_i\dot{\varphi}(t) + z(t-1) + g_b, \quad u(0) := 0 \tag{3.7}$$

$$z(t) = g_f[-g_i\dot{\varphi}(t) + z(t-1)]. \tag{3.8}$$

Der Drehsinn des Winkels $\varphi(t)$ und das Motorsignal $u(t)$ müssen das gleiche Vorzeichen aufweisen.

Im Vergleich zu herkömmlichen Reglern fehlt der CSL-Struktur streng genommen ein Sollwert, auf den eine rückgekoppelte Eingangsgröße eingeregelt wird. Es findet kein Vergleich einer Sollgröße mit dem Istwert der Geschwindigkeit statt. Man kann jedoch den fehlenden Sollwert als einen Vergleich mit 0 interpretieren und erhält so – je nach Wahl des Parameters g_i – einen I-Regler mit Vorverstärkung, dessen Ziel es ist, die Geschwindigkeit auf 0 zu bringen. Ein herkömmlicher Regler hat jedoch immer das Ziel einen Sollwert einzuregeln während sich die CSL-Struktur auf einer Trajektorie im Zustandsraum bewegt.

Die Schleifenverstärkung g_i wird einmalig für ein bestimmtes System eingestellt und kompensiert die Verluste, die z. B. durch Reibung während der Rückführung vom Ausgang zum Eingang auftreten. Sie enthält implizit den Proportionalitätsfaktor k_N des Motors (Drehzahlkonstante), der das Verhältnis von Spannung und Drehzahl angibt, $g_i = 1/k_\mathrm{N}$, so dass die Schleifenverstärkung 1 ist.

Der Rückkopplungs-Parameter g_f wird dagegen abhängig vom gewünschten Verhalten gewählt, welche exemplarisch in Abbildung 3.5 und als Übersicht in Tabelle 3.1 dargestellt sind.

Die Parameter können anhand der folgenden Vorgehensweise iterativ ermittelt werden: Man setze $g_\mathrm{f} := 1$, $g_\mathrm{i} := 0$. Es wird also der Hold-Modus eingestellt, der jedoch noch keine Eingangssignale erhält. Dann wird g_i in kleinen Schritten von z. B. 0,5 erhöht. Der Motorregler wird vor jedem nächsten Schritt deaktiviert, das Gelenk wieder in die Ausgangsposition gebracht und der Regler dann erneut aktiviert. Dies wird solange wiederholt, bis das betrachtete Gelenk an dieser Position ohne Versatz gehalten wird, der Hold-Modus also das erwartete Verhalten zeigt. Wird der Hebel am Gelenk von außen bewegt, sollte auch dann die vorherige Position wieder eingenommen werden und andernfalls g_i weiter erhöht werden. Dabei kann es je nach Anwendung auch gewünscht sein, dass die Regelung ein oder zwei mal überschwingt. Danach kann der Parameter g_f unabhängig von g_i für die anderen Modi eingestellt werden. In Abbildung 3.5 sind vier verschiedene Verhaltensmodi der CSL-Struktur veranschaulicht.

Tabelle 3.1.: Übersicht über die verschiedenen CSL-Modi und ihre Parameter.

Release	Hold	Contraction	Support
$g_\mathrm{i} > 0$	$g_\mathrm{i} > 0$	$g_\mathrm{i} > 0$	$g_\mathrm{i} < 0$
$0 \leq g_\mathrm{f} < 1$	$g_\mathrm{f} = 1$	$g_\mathrm{f} > 1$	$g_\mathrm{f} = 0$

Der **Release**-Modus in Bild 1 koppelt die eingehende Winkelgeschwindigkeit auf den Motorausgang, wechselt jedoch das Vorzeichen. Die Integrationskonstante liegt zwischen 0 und 1, wobei 0 einer proportionalen Regelung mit dem Proportionalitätsfaktor g_i entspricht. Je größer g_f gewählt wird, desto größer wird gegen die Bewegung gesteuert, was einer zusätzlichen viskosen Reibung entspricht (siehe Kapitel 4.1.3).

Bei $g_\mathrm{f} = 1$ wird die Bewegung kompensiert und die aktuelle Position beim Einschalten des CSL-Reglers gehalten. Anschaulich kann dies auch als virtuelle Feder beschrieben werden, deren Steifigkeit durch den Parameter g_i eingestellt werden kann. Dieser Modus ist als

Hold-Modus in Bild 2 von Abbildung 3.5 dargestellt.

Direkt danach schließt sich der **Contraction**-Modus an (Bild 3), der für das ABC-Learning von besonderem Interesse ist. Er zeigt bei behutsamer Wahl des Parameters g_f die Eigenschaft, der anfänglichen Geschwindigkeit immer stärker entgegen zu arbeiten. Solange die resultierende Bewegung – die nun in entgegengesetzter Richtung geschieht – langsamer ist und damit am Eingang der Schleife ein Wert anliegt, durch den der Integrator nicht sein Vorzeichen wechselt, entsteht bei dem im Bild 3 gezeigten frei hängenden Pendel eine Bewegung gegen die Schwerkraft. Da die Integrationskonstante etwas größer als 1 ist, wächst das Ausgangssignal des Integrators solange, bis die Bewegung weniger Kraft als zuvor erfordert (vgl. Abb. 6.1). Wenn ein instabiler Fixpunkt überschritten ist (das Pendel ist aufgerichtet) wirkt die Schwerkraft nun andersherum. Der interne Wert des Integrators und damit der Ausgang zum Motor nimmt nun ab und dreht ebenfalls sein Vorzeichen, so dass der instabile Fixpunkt nun von der anderen Seite angefahren wird. Idealerweise kommt das System im Fixpunkt zum Stillstand, der Eingang und der Integrator haben den Wert 0. Normalerweise kommt die Bewegung aber besonders durch die Haftreibung nie zum Stillstand und befindet sich fortwährend in leichter Bewegung um den instabilen Fixpunkt. Das CSL im Contraction-Modus hat die Eigenschaft, die Fixpunkte des dynamischen Systems insofern zu invertieren, dass die vormals anziehenden Basins der stabilen Fixpunkte abstoßend werden und die der instabilen Fixpunkte anziehend.

Der **Support**-Modus (Bild 4) steuert nicht gegen die Bewegung sondern gibt zusätzlich Drehmoment in der Richtung der Bewegung, er verringert also je nach Wahl der Parameters g_i die Reibung verschieden stark oder hebt sie bei idealer Einstellung vollständig auf.

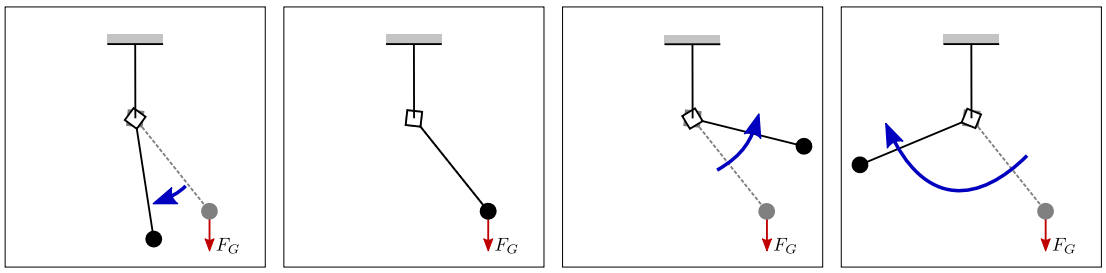

Abbildung 3.5.: Verschiedene Verhaltensmodi des Cognitive Sensorimotor Loops am Beispiel eines frei hängenden Pendels. Von links nach rechts: Bild 1. *Release*, Bild 2. *Hold*, Bild 3. *Contraction*, Bild 4. *Support*

Eine empirische Untersuchung der Auswirkung der Parameter g_i und g_f auf das Verhalten des Contraction-Modus wurde in (Werner, 2013, S. 33 ff.) unternommen.

Ist ein CSL-Regler für ein Gelenk aktiv, wird das System in einen stabilen oder instabilen

Fixpunkt gebracht, je nach Wahl des Parameters g_f. Der in der Einleitung beschrieben Forderung Rodney Brooks' folgend, die Welt als ihr eigenes Modell zu benutzen, besitzt das CSL kein internes Modell. Die CSL-Struktur ist im Hinblick auf traditionelle winkelbasierte Ansteuerungen damit kognitiv, da die Fixpunkte nicht aufgrund eines internen Modells durch einen herkömmlichen Positionsregler angefahren werden, sondern da sie als physikalisch exakte Eigenschaften jedes dynamischen Systems durch Reaktion auf das System selbst gefunden werden. Im Folgenden werden einige Anwendungsbeispiele zur Verdeutlichung der Funktionsweise des CSL-Paradigmas gegeben.

3.3.1. Aufstehendes Bein

Dieses Beispiel ist ein Experiment, das von Matthias Kubisch als Demonstrator entwickelt worden ist (siehe Kubisch et al. 2011).

Das verwendete robotische Bein ist Teil des humanoiden Roboters *Myon*, der im Labor für Neurorobotik an der Humboldt-Universität zu Berlin im Rahmen des ALEAR-Projektes entwickelt und gebaut worden ist. Dieser ist ein leichtgewichtiger modularer Roboter, der als Gesamtsystem und auch einzelne Gliedmaßen zerlegt unabhängig von externer Energieversorgung und Rechenleistung betrieben werden kann. Die Gelenke werden von Servomotoren aktuiert, welche größtenteils – inspiriert von biologischen Muskeln und Sehnen – durch Drahtseile in einer agonistisch-antagonistischen Bauweise mit den Gliedmaßen verbunden sind.

Neben Experimenten mit dem gesamten Roboter lassen sich aufgrund der Autonomie der einzelnen Module auch isolierte Experimente durchführen. Bild 3.6 illustriert mit einer Folge von Standbildern eine durch das zuvor beschriebene CSL erzeugte Aufstehbewegung, die mit nur einem Bein des Myons vollführt wird. Dabei ist in dem Bein in den drei sagittalen Gelenken, also am Fußgelenk, dem Knie und an der Hüfte jeweils ein CSL-Regler im Contraction-Modus aktiv. Diese sind von den anderen jeweils unabhängig, das heißt jedes Gelenk wird ohne Daten der anderen Gelenke angesteuert. Da hier aber neben der Gravitation, die auf einen starren Körper wirkt, auch weitere Kräfte über die kinematische Kette auftreten, sind diese CSLs nur in der Berechnung voneinander unabhängig. Über die Physik wirken sie mit- und gegeneinander und das Gesamtverhalten ergibt sich aus der Morphologie und den physikalischen Eigenschaften des Beines.

Bei richtig gewählten Parametern ist es so möglich, dass sich ein Bein aus einer liegenden Position selbstständig erhebt und sich im aufgerichteten Stand selbst balanciert, ohne dabei bei einem der Regler die Parameter oder deren Verhalten zu ändern. Der Phasenraum dieses Systems enthält zwei Basins um die vormals instabilen Fixpunkte, von denen das eine zum gezeigten Attraktor und damit einer Aufstehbewegung führt, das andere zu einer

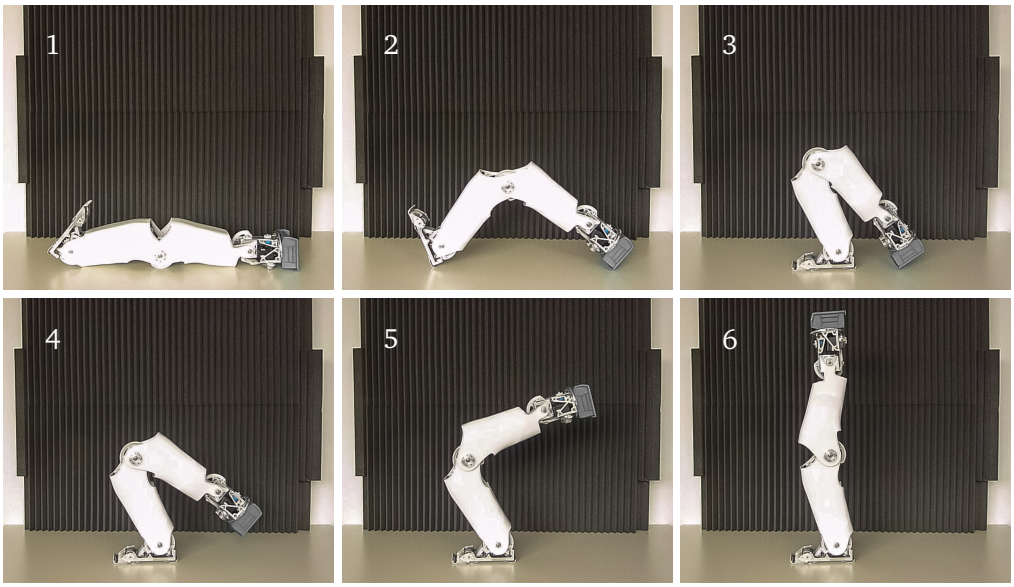

Abbildung 3.6.: Eine Bewegungsfolge, die ein Bein aufrichtet. Bild 1 zeigt eine Startkonfiguration, in der das System sich aufrichten kann. In Bild 2 hat sich das Knie gegen die Gravitation erhoben und bewegt sich weiter. Der Fuß bewegt sich gegen den nun Gegenkraft aufbringenden Boden, auf dem die Ferse inzwischen steht. Bild 3 zeigt den Moment der Gewichtsverlagerung ausschließlich auf den Fuß, die geschieht nachdem durch die Aufrichtung der Schwerpunkt immer mehr zum Fuß verlagert worden ist. Das graue obere Ende hebt vom Boden ab und das Knie dreht kurz danach seine Richtung, da die Gravitation nun nur noch auf den Oberschenkel wirkt. Die fortlaufende Aufrichtung und Stabilisation zeigen Bild 5 und 6.

Brückenkonfiguration, bei denen zwei Stützstellen auf dem Boden verbleiben. Die Startkonfiguration bestimmt, in welchen Attraktor das System gerät. Es sei betont, dass für die gesamte Bewegung kein Umschalten oder Verändern der Parameter geschieht.

3.3.2. CSL-basiertes aufrechtes Stehen

Anstatt nur ein einzelnes Bein im Attraktor zu halten, war es auch wünschenswert, einen vollständigen Myon-Roboter aufrecht balancieren zu lassen und gegen leichte Störungen oder einen unebenen Boden robust zu werden. Dafür wurde auf den Hüft- und Halsgelenken, die Bewegungen innerhalb der Frontalebene erzeugen und auf den Fuß-, Hüft- und Halsgelenken in der Saggitalebene jeweils ein CSL im Contraction-Modus aktiviert. Die Knie und die Fußgelenke in der Frontalebene wurden auf eine feste Winkelposition eingestellt. Der Kopf wird damit aufrecht gehalten, und die Arme hängen passiv herab.

Die Hüftgelenke und die Füße halten den Körper so bereits innerhalb des Basins des stabilen Fixpunktes, solange die Störungen nicht ausreichen, dieses zu verlassen. Jedoch ist es zusätz-

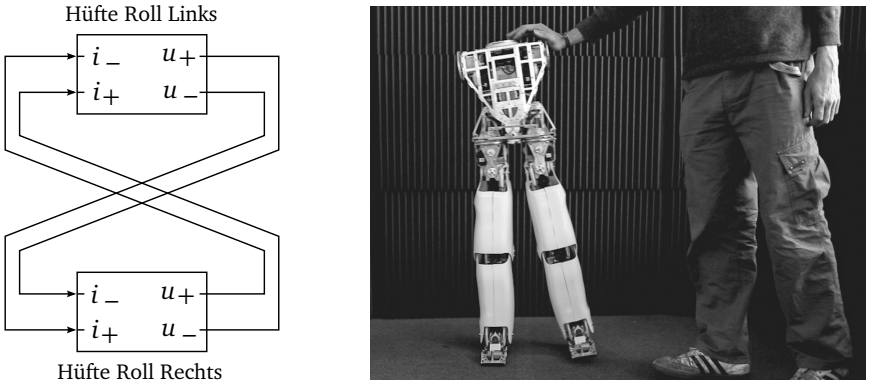

Abbildung 3.7.: Links: Kopplung zweier CSL-Strukturen um Bewegungen gegeneinander zu vermeiden. Rechts: Balancierender Myon-Körper ohne Arme und Kopf.

lich nötig, zu verhindern, dass die Hüftgelenke in der Frontalebene gegeneinander arbeiten und die Motoren sich dadurch aufheizen oder durch gegenseitige Beeinflussung der Regler die Beine immer weiter nach innen driften und den stabilen Stand des Körpers verhindern. Dafür werden diese beiden Instanzen des CSL miteinander inhibitorisch verkoppelt. Das Ausgangssignal beider CSL-Strukturen wird am Eingang der jeweils anderen verfügbar gemacht, siehe Abbildung 3.7. Das jeweilige Ausgangssignal $u(t)$ wird dabei in einen Positiven Anteil $u_+(t) = max(0, u(t))$ und einen negativen Anteil $u\_(t) = min(0, u(t))$ geteilt. Mittels dieser neuen Information wird bei einer Bewegung der beiden CSL-Regler in verschiedene Richtungen der interne Parameter g_f moduliert, so dass abhängig vom anderen Hüftgelenk das Verhalten graduell vom Contraction-Modus zu einem Release-CSL verändert wird und so letztendlich ein Gleichgewichtszustand erreicht wird (Kubisch et al., 2011).

4. Dynamische Simulation von Robotern

Bevor ich das beschriebene Lernverfahren auf einem realen Roboter umgesetzt habe, wollte ich es zuerst in einer Simulationsumgebung testen und entwickeln. Um verschiedenen Beschränkungen verfügbarer Umgebungen zu entgehen, habe ich für diesen Teil der Arbeit eine eigene Simulationsumgebung erstellt. Mögliche dreidimensionale Simulationsumgebungen wie *V-REP* oder *Gazebo* erlauben lediglich eine Ansteuerung von Gelenken über Positions- oder Geschwindigkeitsregler, jedoch nicht die direkte Ansteuerung über Drehmomente, wie es für ein eigenes Motormodell erforderlich ist. Zudem sind nur manche Projekte quelloffen, so dass sich keine eigenen Änderungen vornehmen lassen und im Zweifelsfall nicht überprüft werden kann, ob sich darin Eigenheiten oder Fehler befinden, die zu unerwünschtem Verhalten führen. Zudem ist eine Berechnung der vollständigen dreidimensionalen Physik hier unnötig. Auch verfügbare zweidimensionale Umgebungen wie z. B. *Physion* treffen mit ihrer Bedienoberfläche und den Datenformaten eine Vorauswahl, die später hinderlich sein kann und bieten nur noch geringe Vorteile gegenüber einer eigenen Implementation.

Der für diese Arbeit gewählte Roboter *Semni* (siehe Abbildung 4.1) ist aus Gründen der besseren Anschauung der Messdaten morphologisch auf Bewegungen in einer Ebene beschränkt und besitzt nur zwei Freiheitsgrade um so die spätere Darstellung und Analyse der Systemzustände und des gesamten Zustandsraumes zu vereinfachen. Die Simulationsumgebung ist daher ebenfalls auf zwei Raumdimensionen begrenzt, kann aber auch weitere Freiheitsgrade simulieren.

Anders als eine bereits existierende analytische berechnende Simulation, die nur für den Semni ausgelegt ist und damit einige Annahmen machen kann, sollte dieser Simulator interaktiv benutzt werden können und für beliebige zweidimensionale Morphologien und Situationen verwendet werden können und neben statischen Berechnungen auch dynamische Abläufe darstellen. Außer der Berechnung der Lageveränderung durch Abrollen und Gravitation ist es also auch nötig, alle weiteren Bewegungsgleichungen zu berechnen, die für Beschleunigung, Bewegung und Kontakt des Roboters mit sich selbst und der Umwelt entstehen. Die Berechnungen sind im Abschnitt 4.1 beschrieben.

Die beiden verwendeten Servoeinheiten werden durch ein einfaches Motormodell simuliert, das neben der Getriebeübersetzung die Wechselwirkung zwischen elektrischer Klemmenspannung, die zur Bewegung des Rotors führt und gleichzeitig der Induktion einer rückwirkenden

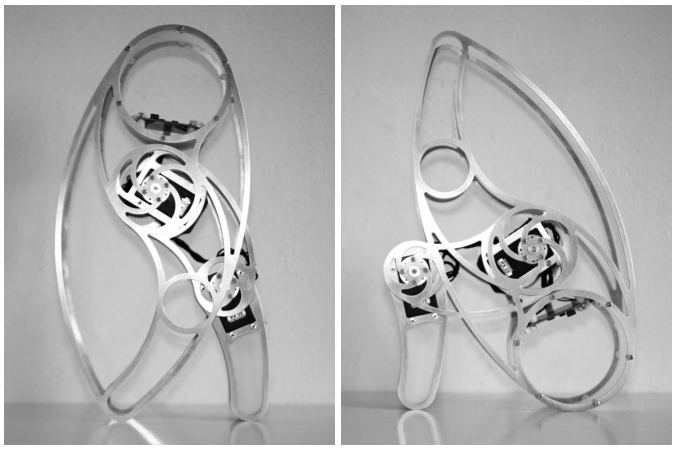

Abbildung 4.1.: Der Roboter *Semni* des Labors für Neurorobotik. Er besitzt zwei rotatorische Freiheitsgrade – ein “Hüftgelenk” und ein “Kniegelenk”. Die Körperpose kann also zusammen mit einem Körperwinkel durch drei Größen angegeben werden. Der Bewegungsspielraum ist quasi-zweidimensional innerhalb der Sagittalebene, da die Drehachsen beide parallel zueinander stehen und die Form der Außenkanten entsprechend gestaltet ist. Ein Umfallen auf eine der Außenseiten bewirkt, dass sich der Roboter nicht mehr selbstständig bewegen kann und wird damit aus dem Bewegungsspielraum ausgeschlossen. Die Gelenke werden mit je einem Dynamixel RX-28 aktuiert, welche über einen zentralen STM32-Prozessor angesteuert werden. Strom-, Temperatur- und Beschleunigungssensoren befinden sich in den RX-28 bzw. auf der Platine im “Kopf”.

elektrischen Spannung durch Kräfte auf den Rotor von außen abbildet. Die Parameter dafür, die im Abschnitt 4.1.4 genauer beschrieben werden, wurden teilweise experimentell ermittelt und teilweise aus Herstellerdaten abgeleitet.

Einen Simulator für die Arbeit an einem Roboter, insbesondere in der Experimentierphase, zu verwenden, hat verschiedene Vorteile. Zum einen kann der Entwicklungszyklus beschleunigt werden, da die zumindest beim Semni notwendigen Wartezeiten, Programme auf den Roboter zu übertragen, entfallen. Der Simulator ist anders als sonst üblich nicht als kompiliertes Programm entwickelt worden sondern benutzt den Webbrowser als Plattform, so dass auf die dort inzwischen recht ausgereiften Entwicklungswerkzeuge zurückgegriffen werden kann. Je nach Browser gehört es neben dem Standardumfang wie einem Debugger, Analysewerkzeugen für Speicherbedarf und Performance auch dazu, Programmcode zur Laufzeit des Simulators abzuändern und sich Ergebnisse direkt ansehen zu können. Zudem kann der Simulator damit auf jeder Plattform schnell verfügbar gemacht werden, ohne eine aufwändige Installation oder Konfiguration vorzunehmen. Auch die simulierte Geschwindigkeit bei den im Rahmen für diese Arbeit vollzogenen Experimenten kann mit mindestens dreifacher Echtzeitgeschwindigkeit als ausreichend bezeichnet werden. An dieser Stelle seien Leser darauf hingewiesen, dass der Simulator in seiner aktuellen Form einfach aufgerufen und ausprobiert werden kann und auf Bildschirmfotos daher hier verzichtet wird. Der URL des Simulators ist `http://neurorobotik.de/simni/simulator.html`. Eine

kurze Bedienungsanleitung und der gesamte Quellcode sind im Elternverzeichnis ebenfalls verfügbar.

Weiterhin ist es mit einem Simulator einfacher, Motorströme und -spannungen, interne Speicherzustände, wie die bereits gelernten Daten und einzelne Schritte und Zustände des Programms zu beobachten, um während der Entwicklung der Verfahren leichter Aufschluss über Fehler und unvorhergesehene Abläufe zu erlangen. Der Ablauf eines Programms auf dem Mikrocontroller des Semni wäre im Vergleich dazu nur schwer direkt einsehbar. Ein simulierter Roboter hat zudem den Vorteil, sich nicht abzunutzen oder sich selbst zu beschädigen. Man kann ein Experiment daher auch ohne weiteres über Nacht sich selbst überlassen und wenn nötig auch mit verschiedenen Startparametern viele Läufe parallel starten. Der wirkliche Roboter muss in der Regel dauerhaft überwacht werden, da z. B. ein schädigendes Aufheizen der Motoren, Tiefentladen des Akkus oder andere mechanische Schäden nicht immer ausgeschlossen werden können. Obwohl ein Ziel des Verfahrens ist, einige dieser Probleme möglichst irrelevant zu machen, ist während der Entwicklung natürlich davon noch nicht auszugehen. Letztendlich kostet jedes Exemplar des Roboters Semni einen bestimmten Geldbetrag, eine weitere Instanz eines Simulators ist dagegen kostenfrei.

Es gilt jedoch bei jeder Simulation, insbesondere von Experimenten mit physikalisch-mechanischen Vorgängen, dass unabhängig vom Aufwand, mit welchem die Parameter mit Messungen und Beobachtungen abgeglichen werden, immer nur eine mehr oder weniger grobe Annäherung an die Realität erreicht werden kann und damit die Ergebnisse auch nur auf höherer Ebene Aussagekraft haben können. Exakte Werte und exakte Verhaltensweisen lassen sich im Allgemeinen nicht ohne Abweichungen auf einer echten Hardware reproduzieren. Gleichwohl ist es möglich, qualitative Aussagen zu komplexen Zusammenhängen zu treffen und zu erkennen, welche Probleme im Weiteren besonderer Aufmerksamkeit bedürfen.

4.1. Physikalische Grundlagen

Die Simulation von Robotern unterliegt bei vielen Anwendungen einem Kompromiss aus Realitätstreue, Rechenzeitbedarf und auch dem Entwicklungsaufwand. Oft werden die physikalischen Berechnungen auf eine Starrkörpersimulation beschränkt und alle Körper werden in ihrer Ausdehnung als konstant angenommen. Eine elastische oder bleibende Verformung wird nicht berechnet. Die Bewegungen können damit durch einfache mechanische Berechnungen und durch gewöhnliche Differentialgleichungen beschrieben werden.

4.1.1. Bewegungen

Das Modell der Newton'schen Mechanik erlaubt die Berechnungen unter Verwendung des Impulses und des Drehimpulses. Der Impuls ist eine Größe für die Bewegungsmenge eines Masseteilchens. Es gilt, dass der Gesamtimpuls $\vec{p}$ gleich dem Produkt von Masse und Geschwindigkeit eines Massepunktes ist:

$$\vec{p} = m\vec{v}\,. \tag{4.1}$$

Impuls und Geschwindigkeit sind damit Vektoren mit gleicher Richtung. Nach Newton's zweitem Gesetz überträgt sich der Impuls $\vec{p}$ auf einen Massepunkt, wenn eine Kraft $\vec{F}$ über eine Zeit t auf ihn wirkt; er wird beschleunigt,

$$\frac{\mathrm{d}\vec{p}}{\mathrm{d}t} = \vec{F}\,. \tag{4.2}$$

Impulse lassen sich in der Simulation also aus den auftretenden Kräften bestimmen, die sich neben der konstanten Gravitation vor allem aus Aktoren, Verbindungen mit anderen Körpern, Reibung und Kollision ergeben. Sind für einen Massepunkt Masse und Impuls bekannt, lässt sich aus Gleichung 4.1 die aktuelle Geschwindigkeit berechnen. Da die Geschwindigkeit die Ableitung des zurückgelegten Weges nach der Zeit ist, kann durch Integration der Weg bestimmt und für jeden Zeitschritt z. B. angezeigt werden. Die Integration muss in Simulationen mit diskreter Zeit auch mit diskreten Integrationsverfahren näherungsweise berechnet werden. Ein einfaches aber nicht stabiles Verfahren dafür ist das explizite Newton-Euler-Verfahren. Weitere Verfahren mit guter Stabilität sind verschiedene Runge-Kutta-Verfahren und strukturerhaltende symplektische Verfahren wie das Störmer-Verlet-Verfahren (Leimkuhler und Reich, 2004).

Um auch die Rotationen zu bestimmen, ist die Betrachtung des Drehimpulses nötig. Der Drehimpuls $\vec{L}$ wird unter Verwendung eines Ortsvektors $\vec{r}$ und des Impulses $\vec{p}$ berechnet,

$$\vec{L} = \vec{r} \times \vec{p}\,. \tag{4.3}$$

Analog zur Impulsänderung ist der Euler'sche Drehimpulssatz definiert, nachdem die Änderung des Drehimpulses durch Einwirken eines Drehmomentes geschieht,

$$\frac{\mathrm{d}\vec{L}}{\mathrm{d}t} = \vec{M}\,. \tag{4.4}$$

Die Beschränkung auf starre Körper erlaubt es, Körper nicht mehr als Ausdehnung einzelner Massepunkte aufzufassen, sondern den von der Orientierung abhängigen Trägheitstensor

für den gesamten Körper zu verwenden. Der Drehimpuls eines Körpers lässt sich so mit der Winkelgeschwindigkeit $\vec{\omega}$ und dem Trägheitstensor Θ mittels

$$\vec{L} = \Theta\,\vec{\omega} \tag{4.5}$$

berechnen (Goldstein et al., 2012, S. 197 ff.).

Für Bewegungen unter Nebenbedingungen, wie z. B. beschränkte Bewegungsbahnen bei Gelenken ist es mitunter schwierig, diese allein mit den Newton'schen Gesetzen zu berechnen. Unter Verwendung von verallgemeinerten Koordinaten und verallgemeinerten Impulsen lassen sich die Bewegungen mit der Lagrange-Formulierung der Mechanik oder der Hamilton'schen Mechanik bestimmen. Eine Nebenbedingung müsste man innerhalb der Newton'schen Mechanik mit weiteren Kräften auf die bewegten Teilchen darstellen, die man als Funktionen des Ortes und der Geschwindigkeit kennen müsste. Die Lagrange-Mechanik führt verallgemeinerte Koordinaten ein, die solche Koordinaten erlauben, dass eine nun geradlinige Bewegungsbahn die Nebenbedingungen erfüllt, ohne die Wechselwirkungskräfte überhaupt darstellen zu müssen (Arnold (1989); Leimkuhler und Reich (2004); Schönhammer (2010)).

4.1.2. Kollisionen

Kollidieren zwei Körper miteinander, überträgt sich abhängig von Masse und Geschwindigkeit m_1 und v_1 des ersten Körpers und m_2 und v_2 des zweiten Körpers ein Teil des Impulses von einem auf den anderen Körper. Nach dem Impulserhaltungssatz und dem Energieerhaltungssatz ist die Summe der Impulse und der Energien stets gleich groß. Es gilt

$$m_1\vec{u}_1 + m_2\vec{u}_2 = m_1\vec{v}_1 + m_2\vec{v}_2, \tag{4.6}$$

$$\tfrac{1}{2}m_1u_1^2 + \tfrac{1}{2}m_2u_2^2 = \tfrac{1}{2}m_1v_1^2 + \tfrac{1}{2}m_2v_2^2\,, \tag{4.7}$$

wobei u_1 und u_2 die Geschwindigkeiten vor der Kollision und v_1 und v_2 die Geschwindigkeiten nach der Kollision angeben. Sind die Größen vor der Kollision bekannt, lassen sich somit die neuen Geschwindigkeiten berechnen:

$$v_1 = \left(\frac{m_1 - m_2}{m_1 + m_2}\right)u_1 + \left(\frac{2m_2}{m_1 + m_2}\right)u_2 \tag{4.8}$$

$$v_2 = \left(\frac{m_2 - m_1}{m_1 + m_2}\right)u_2 + \left(\frac{2m_1}{m_1 + m_2}\right)u_1\,. \tag{4.9}$$

Die neuen Richtungsvektoren ergeben sich aus dem Reflexionsgesetz, der Einfallswinkel ist gleich dem Ausfallswinkel.

4.1.3. Reibung

Neben den Bewegungen von Objekten selbst, wirken bei Berührung von zwei Körpern entgegen den Kräften, die die Bewegung hervorbringen, Reibkräfte, die sich nach ihrer Charakteristik dreigeteilt betrachten lassen. Stehen zwei Körper zueinander vollständig still, wirkt die von den Materialien und dem Anpressdruck abhängige *Haftreibung*, die bis zu einer maximalen Kraft $F_{\mathrm{H}}^{\mathrm{Max}}$ die Körper an der Bewegung hindert, also der Bewegungskraft F eine entgegengesetzte betragsgleiche Kraft F_{H} aufbringt. Sei μ_{H} der Reibungskoeffizient für die Haftreibung und F_{N} die Kraft senkrecht zur Fläche. Dann gilt für $v = 0$

$$F_{\mathrm{H}} = -F \tag{4.10}$$

$$F_{\mathrm{H}} \leq \vec{F}_{\mathrm{H}}^{\mathrm{Max}} = \mu_{\mathrm{H}} F_{\mathrm{N}} \,. \tag{4.11}$$

Ist die auftretende Kraft größer, beginnt eine Bewegung entlang der Oberflächen, die der *Gleitreibung* und nicht mehr der Haftreibung unterliegt. Diese unterteilt sich wiederum in die *trockene Reibung*, eine konstante Reibkraft, und die *viskose Reibung*, die von der Geschwindigkeit abhängig ist und mit ihr ansteigt. Die trockene Reibkraft wird auch als *Coulomb'sche Reibung* bezeichnet und ist nur von der Normalkraft F_{N} (Kraft senkrecht zur Fläche) und einem für die beteiligte Materialkombination spezifischen Koeffizienten μ_{G} abhängig,

$$F_{\mathrm{G_T}} = \mu_{\mathrm{G}} F_{\mathrm{N}} \,. \tag{4.12}$$

Die viskose Reibkraft ist dagegen zusätzlich von der Geschwindigkeit v bzw. der Winkelgeschwindigkeit ω und einem eigenen Koeffizienten μ_{V} abhängig,

$$F_{\mathrm{G_V}} = \mu_{\mathrm{V}} F_{\mathrm{N}} v \,. \tag{4.13}$$

Bewegt sich ein Körper aus dem Stillstand heraus, wirkt für den Moment des Loslösens nur die trockene Reibung, da die Geschwindigkeit gleich null ist. Diese ist meist kleiner als die Haftreibung zuvor, so dass ein ruckartiges Losreißen geschieht. Dieses Verhalten wurde erstmals von Stribeck (1902) beschrieben, nach dem dieses Verhalten als Stribeck-Effekt benannt ist. Ist die Haftreibung deutlich größer als die nachfolgende Gleitreibung, kommt es zum sogenannten Stick-slip-Effekt, bei dem die Trägheit des Körpers diesen noch weiter gleiten lässt, bevor der antreibende Körper hinterher kommt, die Antriebskraft lässt nach. Es kommt wieder zum Festhaften, bis kurze Zeit später wieder eine Antriebskraft wirkt. So kommt es zu einem periodischen Losreißen und Stehenbleiben, das sich oft als hörbare Oszillation mit hoher Frequenz äußert.

Die genannten grundlegenden Zusammenhänge sind einfache Modelle und damit nur näherungsweise korrekt und es existieren viele weitere Modelle, die komplexere Berechnungen

erfordern, um größere Deckung mit der Realität zu bieten. Einen Überblick über einige Modelle findet man z. B. in Olsson (1998) oder Kubisch (2008).

4.1.4. Motorverhalten

Gleichspannungsmotoren können in verschiedenen Bauarten ausgeführt sein, von denen hier stellvertretend nur eine betrachtet werden soll, die permanenterregte Gleichstrommaschine. Diese besteht aus einem unbeweglichen Teil, dem Stator, der als Hohlzylinder aufgebaut ist. Im Stator ist durch Permanentmagnete ein Magnetfeld aufgebaut, das in der rotatorischen Ebene des Zylinders jeweils halbseitig die magnetischen Pole ausbildet. In diesem Hohlraum befindet sich der drehbar gelagerte Anker oder Rotor, der von einer oder mehreren Spulen umwickelte Metallkerne besitzt. Die Kontakte der Spulen werden über einen Kommutator nach außen geführt, über den während der Drehung zwei Bürsten schleifen und damit Kontakt zur Stromversorgung herstellen. Der Aufbau ist so gestaltet, dass gerade jeweils die Kontakte geschlossen werden, dessen Spule ein Magnetfeld erzeugen, das dem Permanentmagnetfeld des Stators abstoßend gegenüber steht. Der Kommutator dient als Polwender und erzeugt einen drehzahlabhängigen Wechselstrom. Die Achse des Rotors verbindet diesen mit dem Antrieb bzw. Abtrieb nach außen. Weitere Details finden sich z. B. in (Fischer, 2009, S. 31 ff.).

Die Anzahl der Motorwicklungen hat Einfluss auf die Stärke der Magnetfelder und auf den elektrischen Widerstand der Spulen. Sobald der Motor sich bewegt, werden die Spulen von Strom durchflossen, es kommt zu Stromwärmeverlust und der Leiter erwärmt sich. Der elektrische Widerstand von Leitern wie Kupfer ist temperaturabhängig und steigt mit dieser an. Je größer der Widerstand ist, desto geringer ist das Drehmoment, das der Motor bei einer gegebenen Klemmenspannung erzeugt. Die Feldstärke der Permanentmagnete verringert sich bei Erwärmung ebenfalls. Misst man den resultierenden Strom, erhält man also eine Aussage über das vom Motor aufgebrachte Drehmoment.

Wird der Rotor bewegt, von außen oder durch eine anliegende Spannung, induziert das wechselnde Magnetfeld nach dem Induktionsgesetz in den Spulen wiederum eine Spannung, die der antreibenden Spannung entgegengerichtet ist. Die resultierende *gegenelektromotorische Kraft* (im Englischen mit *Back-EMF* bezeichnet) dämpft die Bewegung und verringert die Motorleistung.

4.2. Implementation des Simulators

Die Implementation einer vollständigen physikalisch möglichst exakten Simulation mit den im Vorhergehenden beschriebenen Grundlagen ist ein aufwändiger Prozess, der es nötig macht, die simulierten Situationen mit realen Experimenten abzugleichen. Anstelle einer eigenen Umsetzung des physikalischen Verhaltens habe ich daher nach einer bestehenden Lösung gesucht, die die Anforderungen erfüllt. Neben einigen dreidimensionalen Physikbibliotheken wie der *Open Dynamics Engine* (ODE) oder der *Bullet Physics Library* existieren auch Bibliotheken, die sich auf eine zweidimensionale Simulation beschränken und damit Vorteile bei Rechenzeit und Komplexität in der Verwendung vorweisen können.

Der entwickelte Simulator *Simni* verwendet daher die quelloffene Physikbibliothek Box2D[2], die eine solide und performante impulsbasierte physikalische Berechnung der Drehmomente, Beschleunigungen und Geschwindigkeiten unter Beachtung von Reibung für starre Körper liefert. Zudem werden Kollisionen erkannt und entsprechende Kollisionsantworten berechnet. Weitere physikalische Effekte wie elastische Verformung, komplexere Reibung, thermische Effekte und damit veränderte Eigenschaften sowie jegliche Motoreigenschaften werden jedoch nicht umgesetzt. Die zeitliche Auflösung ist diskret und die Schrittweite ist konstant aber beliebig wählbar. Als Integrationsverfahren für die numerische Berechnung der Bewegungsgleichungen benutzt Box2D ein symplektisches Eulerverfahren[3], welches vergleichsweise stabile Berechnungen sicherstellt. Um Oszillationen und Instabilitäten gering zu halten, werden die Physik und die Regler in der hier vorgestellten Simulation mit einem konstanten Zeitschritt von 1,04 ms berechnet, was einer Frequenz von 960 Hz entspricht. Die Berechnung der Reibung zwischen Körpern in Box2D beschränkt sich auf die trockene Reibung und lässt sich bis auf einen entsprechenden Koeffizienten nicht weiter beeinflussen. Die Reibung innerhalb der Drehgelenke wird im Folgenden beschrieben.

Darüber hinaus ist es notwendig, ein Motormodell für die im Semni verwendeten Dynamixel RX-28 umzusetzen. Dieses Motormodell wird aus Performanzgründen auf die Berechnung der resultierenden entgegengerichteten Spannung aus der Geschwindigkeit des Gelenks und auf die Berechnung des Drehmoments aus der Klemmenspannung begrenzt. Damit wird keine thermische, und damit elektrische, Veränderung der Motorwicklungen und Motorlager berechnet. Anstatt den zeitlichen Verlauf der Temperatur und die Widerstandsänderung nachzubilden, wird eine konstante Arbeitstemperatur von 65 °C angenommen, die für die realen RX-28 üblich ist und ein dem entsprechender konstanter Widerstandswert berechnet und verwendet, so dass die Motoren keine zu großen Drehmomente aufbringen können und die Bewegungen damit in den ersten Minuten nach dem virtuellen Anschalten etwas

[2] Box2D findet sich inklusive einiger Dokumentation und Beispiele unter `http://www.box2d.org`.
[3] Eine entsprechende Aussage des Autors dazu findet sich unter `http://www.box2d.org/forum/viewtopic.php?f=8&t=1609`.

langsamer als in der Realität sind. Da es hier aber um quasistatische Bewegungen gehen soll, ist dieser Unterschied nicht relevant.

Die simulierte Servoeinheit Dynamixel RX-28 verwendet als Motor den RE-max 17 der Firma Maxon, der sich durch eine geringe Rotorträgheit auszeichnet, da der Rotor eisenlos ist. Die Rotorträgheit beeinflusst das Beschleunigungsverhalten, wird aber wegen quasistatischer Bewegungen ebenfalls vernachlässigt. Die Bewegung von Motoren gerade im niedrigen Drehzahlbereich unterliegt desweiteren stark den Effekten der Haftreibung und trockener Reibung, welche gesondert implementiert werden.

Mittels der im vorigen Abschnitt beschriebenen Zusammenhänge lässt sich ein sehr einfaches Modell der Gleichstrommaschine aufstellen, das das Verhalten als Aktor sowohl als auch als Generator berechnet (Nise, 2004). Die notwendigen Parameter sind die Drehmomentkonstante k_m, die Drehzahlkonstante k_n und der Widerstand der Wicklungen R. Die Motorspannung U_Motor berechnet sich dann aus der aktuellen Winkelgeschwindigkeit $\omega = \dot{\phi}$ und der anliegenden Klemmenspannung U_in mittels

$$U_\mathrm{Motor} = U_\mathrm{in} - k_\mathrm{n}\omega\,. \tag{4.14}$$

Das am Abtrieb zur Verfügung stehende Drehmoment berechnet sich über

$$M = U_\mathrm{Motor}\frac{k_\mathrm{m}}{R}\,. \tag{4.15}$$

Anhand des Datenblattes[4] des RE-max 17 ist der Innenwiderstand bei 25 °C mit 8,3 Ω gegeben. Aus Erfahrungswerten der Temperaturmessung des Motorgehäuses über den verfügbaren Temperatursensor des RX-28 habe ich eine Schätzung der durchschnittlichen Arbeitstemperatur des Rotors und damit der Kupferwicklungen auf 65°C vorgenommen. Mit $T_0 = 25\,°\mathrm{C}$ und $T = 65\,°\mathrm{C}$ und der Beziehung

$$\rho(T) = \rho(T_0)\cdot(1 + \alpha\cdot(T - T_0)) \tag{4.16}$$

und dem spezifischen Widerstands-Temperaturkoeffizient für Kupfer $\alpha = 3{,}9\cdot 10^{-3}$ lässt sich der spezifische Widerstand für 65 °C entsprechend errechnen (vgl. Hering, 2005, S. 5), man erhält $R = \rho(T) = 9{,}59\,\Omega$.

Im RE-max 17-Datenblatt finden sich ebenfalls die Drehzahlkonstante k_n, die den Proportionalitätsfaktor zwischen Drehzahl und Spannung angibt und die Drehmomentkonstante k_m, die den Proportionalitätsfaktor zwischen Strom und Drehmoment angibt. Diese sind dort jedoch nur für den Motor selbst angegeben und müssen für den RX-28 für die Übersetzung

[4] Das Datenblatt zum RE-max 17 findet sich unter `http://www.maxonmotor.com/medias/sys_master/8807237582878/13_145_DE.pdf`

des Getriebes und die damit verbundene erhöhte Reibung umgerechnet werden. Das verwendete Getriebe im RX-28 hat nach Messungen im Team des Labors für Neurorobotik eine Übersetzung von 1 : 195 und einen Wirkungsgrad von etwa 62,5%. Bei zusätzlicher Umrechnung von Umdrehungen pro Minute in Bogensekunden ergibt sich für die Drehzahlkonstante $k_{\mathrm{n}}^{\text{RE-max}} = 889\,\mathrm{min/V}$

$$k_{\mathrm{n}} = \frac{k_{\mathrm{n}}^{\text{RE-max}}}{60} \cdot 2\pi \cdot \frac{1}{195} = 0{,}4774\,\frac{\mathrm{rad}}{\mathrm{s\,V}}\,. \tag{4.17}$$

Die Drehmomentkonstante $k_{\mathrm{m}}^{\text{RE-max}} = 10{,}7\,\mathrm{mNm/A}$ wird unter Berücksichtigung des Wirkungsgrades des Getriebes und der gewünschten Größenordnung mittels

$$k_{\mathrm{m}} = (\frac{k_{\mathrm{m}}^{\text{RE-max}}}{1000}) \cdot 195 \cdot 0{,}625 = 1{,}3041\,\frac{\mathrm{N\,m}}{\mathrm{A}} \tag{4.18}$$

angepasst.

Das Motormodell ist als Blockschaltbild Teil der Abbildung 4.2.

Das Modell simuliert eine analoge Ansteuerung des Motors für Klemmspannungen im Bereich von $-12\,\mathrm{V}$ bis $12\,\mathrm{V}$, die sich direkt auf die Feldstärke auswirken. Der Dynamixel RX-28 ist dagegen nur mit einer Pulsweitenmodulation (PWM) ansteuerbar, die mit einer Abtastrate von $16\,\mathrm{kHz}$ zwischen zwei Plateaus mit $0\,\mathrm{V}$ bzw. $12\,\mathrm{V}$ umschaltet. Je nach angesteuertem Wert wird der Anteil des Zeitfensters für die beiden Bereiche angepasst.

Eine weiteres Detail der Implementierung ist, dass mit Box2D für die einzelnen Gelenke zwar Zielgeschwindigkeiten festgelegt werden können, für welche dann intern Kräfte eingeregelt werden, die die Bewegungen erzeugen. Es ist aber ursprünglich nicht vorgesehen, diese Kräfte direkt zu setzen, wie es für einen eigenen Motorregler, insbesondere das zuvor beschriebene CSL, und das Motormodell sinnvoll ist. Daher habe ich die Box2D-interne Funktion “SolveVelocityConstraints” für Rotationsgelenke überschrieben, so dass wahlweise auch ein vorgegebenes Drehmoment in Impulse umgerechnet werden kann, die dann zu den bereits von Box2D berechneten Impulsen addiert werden. Der eigentliche Box2D-Programmtext musste dabei nicht geändert werden. Der in Gleichung 4.4 definierte Euler’sche Drehimpulssatz kann dafür verwendet werden. Die Änderung des Drehimpulses für ein gefordertes Drehmoment ist

$$d\vec{L} = \vec{M}\,dt\,. \tag{4.19}$$

Die Reibung der Gelenke ist für ein möglichst realitätsnahes Verhalten besonders wichtig, da wie zuvor beschrieben nicht nur gedämpfte Bewegungen simuliert werden müssen, sondern einige Effekte erst mit den verschiedenen Reibungstypen auftreten. Daher ist es nötig, zumindest qualitativ die drei Reibungsarten umzusetzen. Eine sehr hohe Genauigkeit

bedeutet immer auch sehr viel komplexere Modelle. Der in Box2D enthaltene Geschwindigkeitsregler für Gelenke erlaubt es, eine Zielgeschwindigkeit und ein maximales Drehmoment vorzugeben, mit dem diese Geschwindigkeit angesteuert wird. Da für die Bewegung der Gelenke ohnehin die direkte Drehmomentsteuerung verwendet wird, lässt sich mit diesem Geschwindigkeitsregler eine Winkelgeschwindigkeit von 0 rad/s einregeln während man ein sehr geringes maximales Drehmoment festlegt, das proportional zur Haftreibungskonstante μ_H ist. Bleibt eine einwirkende Kraft oder die resultierende Kraft des Motormodells unter diesem Wert, kommt keine Bewegung zu Stande. Ist das einwirkende Drehmoment größer, kann eine Bewegung stattfinden, diese wird jedoch weiterhin mit dem vorgegebenen maximalen Drehmoment gebremst, was der trockenen Reibung entspricht. Der Übergang von Haftreibung zu trockener Gleitreibung ist damit aber ohne den Stribeck-Effekt oder eine Hysterese umgesetzt[5]. Eine eigene Umsetzung dieser Reibungsart hätte es notwendig gemacht, auf die Summe der aktuell wirkenden Drehmomente eines Gelenkes zuzugreifen (die Haftreibung ist von der Geschwindigkeit und dem Drehmoment abhängig), was aber mit der vorhandenen Box2D-API und damit ohne weiteres Studium des Quellcodes nicht möglich ist. Aus Zeitgründen und da ich bereits mit den genannten Mitteln recht gute Ergebnisse erreichen konnte, habe ich diese Möglichkeit nicht weiter verfolgt. Die viskose Gleitreibung, die nicht durch diesen Trick abgedeckt wird, ist ausschließlich geschwindigkeitsabhängig. Sie kann daher leicht mittels der unter 4.1.3 angegebenen Gleichung berechnet werden und als zusätzlich wirkendes Drehmoment an die Physikengine übergeben werden. Zusätzlich ist zu bemerken, dass für den betrachteten Anwendungsfall von kugelgelagerten Gelenken für alle drei Reibungsarten die Abhängigkeit von der Normalkraft F_N so gering ist, dass diese vernachlässigt werden kann.

Das verwendete Getriebe im RX-28 hat wie bereits beschrieben eine Übersetzung von 1 : 195 und einen Wirkungsgrad von etwa 62,5%. Der Wirkungsgrad kann auch als Getriebereibung interpretiert werden, der die Bewegung des Motors hemmt. Dabei ist es aber von Bedeutung, ob der Motor eine Last hält oder absenkt, wobei eine Reibung unterstützend wirkt, oder ob der Motor antreibt oder eine Last anhebt, wobei die Reibung hinderlich wirkt. Der Wirkungsgrad ist damit richtungs- und belastungsabhängig bzw. es wirkt eine belastungsabhängige Reibung (siehe Le-Tien und Albu-Schäffer, 2012, S. 2). Diese Reibung wird nicht im Modell umgesetzt, da dafür ebenfalls das auf ein Gelenk wirkende Drehmoment notwendig wäre. Da aber für den Semni die Belastungsgrenzen der Motoren im Normalfall nicht erreicht werden, ändert sich das Verhalten dadurch unwesentlich.

Die weiteren Daten, mit denen der Körper des Semni in der Simulation nachgebildet wird, stammen aus den ursprünglichen Konstruktionsdaten von Torsten Siedel, die im Programm SolidWorks vorliegen. Daraus lassen sich Polygondaten, Masseschwerpunkte der einzelnen Körperteile und Gewichte entnehmen. Die Gewichte wurden aber zusätzlich mit einer Waage

[5] Im Nachhinein hat sich herausgestellt, dass das Verhalten der Motorregler dadurch im Vergleich mit der echten Hardware gewisse Unterschiede zeigt. Weiteres dazu in Kapitel 6.

bestimmt um Schrauben und Akkugewichte mit einzubeziehen. Die Werte sind im Anhang in Tabelle A.2 zu finden.

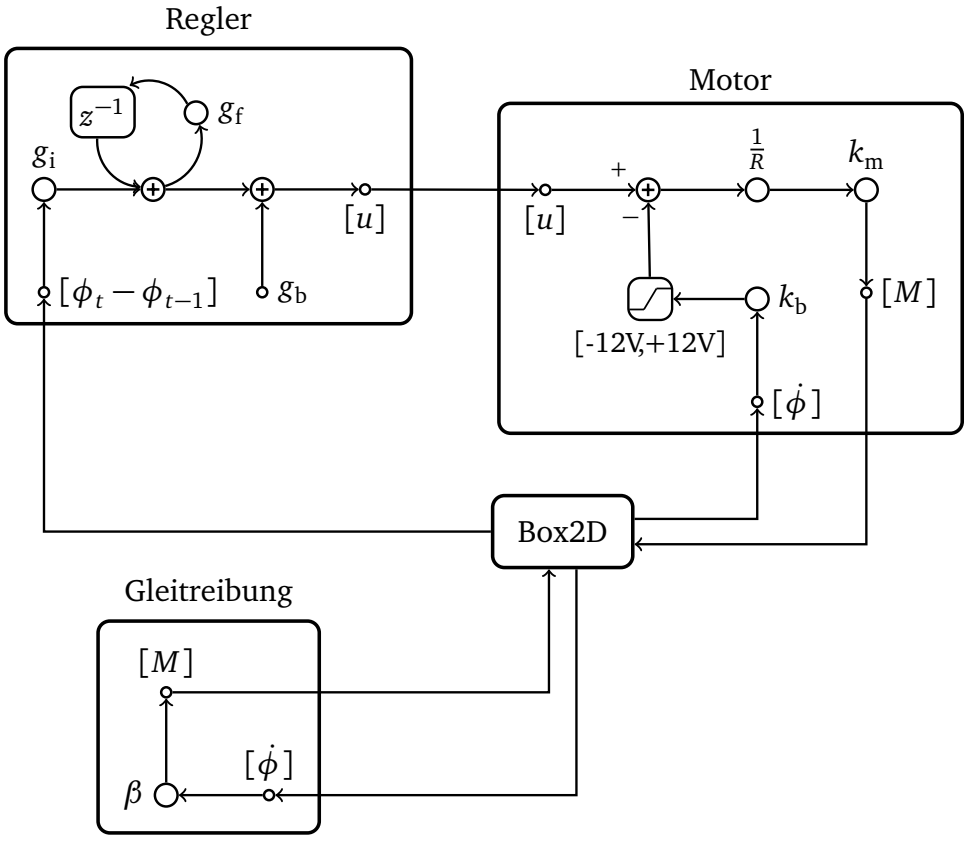

Abbildung 4.2.: Die einzelnen Komponenten der Simulation: CSL-Regler, Motormodell, Reibungsmodell und Box2D. Plus oder Minus an einem Pfeil geben das Vorzeichen bzw. seine Änderung an. Eckige Klammern geben das Formelzeichen für eine Größe an. Das Motorsignal u entspricht einer Spannung, M einem Drehmoment und $\dot{\phi}$ einer Winkelgeschwindigkeit. Innerhalb der Physiksimulation werden Motorträgheit, resultierende Kräfte am Hebelarm und trockene Reibung sowie Haftreibung im Gelenk berechnet.

5. Attractor-Based Behavior Control

Das ABC-Learning ist eine eigenständige Lernarchitektur auf Grundlage der Cognitive Sensorimotor Loops zur Selbsterkundung und Selbst-Erkundung des sensomotorischen Zustandsraumes eines Roboters.

Im Unterschied zu anderen Lernverfahren ist das ABC-Lernverfahren auf die zugrundeliegenden dynamischen Systeme echter physikalischer Systeme ausgerichtet. Der Roboter wird als dynamisches System betrachtet und für Bewegungen die CSL-Struktur verwendet. Dadurch ist es möglich, Bewegungen direkt im Einklang mit allen auftretenden mechanischen Eigenschaften und Effekten auszuführen ohne konkrete Modelle für diese zu kennen oder zu implementieren. Durch systematische Exploration des Zustandsraumes werden Fixpunkte gefunden und dabei als strukturiertes Wissen über das zuvor unbekannte dynamische System gespeichert. Dabei kommt unverzüglich die Frage auf, was mit dem gelernten Wissen geschieht und wie daraus Aktionen und Fähigkeiten entstehen können, die weiterhin die Eigenschaften der dynamischen Systeme vorteilhaft ausnutzen. Vorteilhaft heißt in diesem Zusammenhang z. B. statt dauerhaft angesteuerter starrer Bewegungen – welche die Motoren unnötig erhitzen und Probleme mit der Energieversorgung erzeugen – in den Fixpunkten energiesparend anzusteuern, indem Reglerstrukturen zum Erreichen bestimmter Systemkonfigurationen die Motoren in geeigneten Positionen deaktivieren oder mit geringerer Steifigkeit verwenden. Das ABC-Learning stellt eine Grundlage dar, auf die weitere Verfahren zur Erkundung und Entdeckung von Verhalten aufsetzen sollen.

Die Selbstexploration erlaubt es, eng an die Gegebenheiten gekoppelte Informationen zu erlangen, ohne vorher festgelegte Modelle zu verwenden. Sie erstellt jedoch ihrerseits ein Modell der Realität, und kann damit Vorhersagen über die unmittelbare Zukunft leisten und mit dem Wissen über den zeitlichen Verlauf einer bereits zuvor vollführten Bewegung potentiell komplexeres Verhalten generieren. Das bereits in Abschnitt 2 genannte Verfahren von Der und Martius verwendet ebenfalls ein Weltmodell zur Vorhersage, das vollständig von Null an gelernt wird. Als Weltmodell wird eine Funktion F mit $x_t = F(x_{t-1}, y_{t-1}) + \xi_t$ der vorherigen Sensorwerte x_{t-1} und Motorwerte y_{t-1} und dem Modellfehler ξ_t gebildet. Die Aktionen mit der höchsten Instabilität werden zuerst exploriert. Im Vergleich dazu werden beim ABC-Learning nicht alle Zustandsübergänge in jedem Zeitschritt erfasst. Es wird stattdessen mittels der CSL-Modi zwischen wenigen Verhaltensweisen gewechselt, so dass nur anhand der Fixpunkte Wissen repräsentiert und gespeichert wird, die das dynamische System

bereits sehr gut beschreiben und nutzt damit auch die implizite Semantik des dynamischen Systems um sinnvolle Aktionen zu vollziehen. Zur Beschleunigung der Exploration werden für das ABC-Lernverfahren in diesem Kapitel mögliche Entscheidungsheuristiken beschrieben und verglichen. Das ABC-Learning verwendet am Anfang nur einen immer gleichen CSL-Regler, der erst geändert oder vollständig ausgetauscht wird, wenn das dynamische System für den quasistatischen Fall vollständig erkundet ist, anstatt diesen gleichzeitig zu verändern und zu lernen. Aufgrund der offenen Struktur der Wissenrepräsentation, der nicht unbedingt festgelegten Algorithmen zur Ansteuerung und Erkennung sind die Ergebnisse des Lernprozesses für weitere Entwicklungen sehr viel besser intuitiv und analytisch zugänglich als bei Der und Martius und erlauben leicht Erweiterungen mit anderen Algorithmen, die einzelne Probleme auch isoliert lösen können.

Durch die Konzentration auf dynamische Systeme bestehen Parallelen zu tatsächlichem biologischen Verhalten. Betrachtet man die beiden Modi Contraction und Release des CSLs, lässt sich eine Analogie zu natürlichen Mechanismen herstellen, bei denen eine sukzessive Muskelkontraktion und Entspannung wichtig ist. Beispiele sind die Atmung, Schwimmbewegungen von Quallen, Flügelschläge oder der Herzschlag. Innerhalb des ABC-Learnings sind solche Bewegungen einfache Wechsel zwischen zwei benachbarten Zuständen, die eine Anspannung und eine Entspannung kodieren.

Eine Teilmenge der entdeckten Fixpunkte kann direkt mit sehr menschlichen Begriffen für Körperhaltungen wie Sitzen, Stehen oder Liegen benannt werden. Es kann gemutmaßt werden, dass auch wir diese Positionen deshalb für jeweils längere Zeitspannen einnehmen und daher auch einzeln benennen, da diese für uns leicht stabil zu halten und damit energieeffizient sind (Hild, 2011). Diese Begriffe werden jedoch nur in einigen natürlichen Sprachen für Lebewesen außer dem Menschen und für nicht belebte Dinge verwendet (vgl. Spranger und Loetzsch, 2009).

5.1. Verfahren

Die Ansteuerung jedes Gelenkes geschieht durch jeweils eine CSL-Struktur, welche durch ihre Eigenschaften das dynamische System direkt für das Lernverfahren zugänglich macht. Es ist nicht nötig, Modelle für das lernende System zu erstellen oder zu optimieren. Die CSLs arbeiten ausschließlich mit lokalen Informationen der jeweiligen Gelenke und aktuieren nur diese. Eine Kopplung von zwei Strukturen ist aber für spätere Zielstellungen ebenfalls denkbar, wie es z. B. in 3.3.2 beschrieben ist. Die Bewegungen werden bei der Erkundung auf langsame Abfolgen zwischen Fixpunkten beschränkt, so dass die Dynamik durch Trägheit und Schwingungsverhalten vorerst außer Acht gelassen werden kann.

Durch das Verändern der Parameter, auch sprunghaft, kann das Verhalten einer CSL-Struktur

instantan verändert werden. Als Beispiel sei ein einfaches frei schwingendes Pendel betrachtet, also eine Masse, die mit dem Pendelarm an einem aktuierten Gelenk befestigt ist und sich frei drehen kann. Dabei handelt es sich um ein dynamisches System mit einem Freiheitsgrad, welches einen stabilen Fixpunkt (Hängen) und einen instabilen Fixpunkt (Stehen) besitzt. Befindet sich das System gerade in seinem instabilen Fixpunkt und die CSL-Regelschleife im Contraction-Modus, wird das Pendel aufrecht stabilisiert. Aus jeder anderen Position heraus wird das System ebenfalls in die aufrechte Position gebracht und dort stabilisiert. Werden zu einem beliebigen Zeitpunkt die Parameter so verändert, dass in den Release-Modus umgeschaltet wird, wird das System in den stabilen Fixpunkt gebracht, das Pendel fällt zum hängenden Zustand, dem stabilen Fixpunkt zurück. Es ist damit also prinzipiell möglich, den Systemzustand nach Belieben von einem Fixpunkt zu einem anderen zu ändern.

Befindet sich ein Roboter mit mehreren Freiheitsgraden in einem instabilen Fixpunkt, während er durch CSLs in mehreren Gelenken, jedoch nicht unbedingt in allen, in diesem Fixpunkt stabilisiert wird, kann durch das Umschalten einer der CSL-Strukturen vom Contraction-Modus zum Release-Modus das System in dieser Dimension in einen stabilen Fixpunkt gelangen, also passiv stabil werden, während die anderen Dimensionen weiterhin durch den Contraction-Modus stabilisiert sind. Das Gesamtsystem ist damit in einen anderen instabilen Fixpunkt gelangt und der neue Fixpunkt ist mit dem vorherigen über einen heteroklinen Orbit verbunden. Anschaulich balanciert das System auf dem Rücken einer “Bergkette”, während es auf dieser entlangwandert (siehe Abbildung 5.1). Nach dem Center Manifold-Theorem ist es nur nötig, das System in einigen wenigen wichtigen Dimensionen zu stabilisieren während sich die restlichen Dimensionen passiv verhalten. So wird der Energiebedarf minimiert, da das System, wenn möglich, seine physikalischen Eigenschaften ausnutzen kann. Diese Fixpunkte sind genau die Punkte im Zustandsraum, die in ihrer lokalen Umgebung mit dem geringsten Energieaufwand zu halten sind (instabile Fixpunkte) oder solche die gar keine Energie brauchen (stabile Fixpunkte). Gleichzeitig ist es daher möglich, immer nur in jeweils einer Dimension den CSL-Modus zu ändern ohne dabei die Erreichbarkeit aller Fixpunkte einzuschränken (Carr, 1981).

Als *Posture* sei im Folgenden ein Tupel $p = (\varphi_1, \ldots, \varphi_n, m_1, \ldots, m_n, a_\mathrm{x}, a_\mathrm{y}, a_\mathrm{z}) \in P$ bezeichnet, das die Attribute eines Roboters enthält, die seine Haltung, Lage und internen Zustände beschreiben. Dazu zählen alle Gelenkwinkel φ_i, die Ausrichtung des Körpers z. B. durch Werte eines Beschleunigungssensors mit drei Achsen $a_\mathrm{x}, a_\mathrm{y}, a_\mathrm{z}$ und die CSL-Modi m_i aller n Gelenke, die zu dieser Konfiguration und Position geführt haben.

Durch das systematische Anfahren von Fixpunkten innerhalb des Zustandsraumes wird ein gerichteter Graph $\mathcal{G} := (V, E)$ mit der Knotenmenge $V := \{v_1, v_2, \ldots, v_m\}$, $v_i \in P$ und der Kantenmenge $E \subset \{(u, v) |\, u \neq v\}$, $u, v \in V$ als Erinnerungsstruktur aufgebaut, dessen Knoten jeweils eine Posture speichern. Die ausgehenden Kanten verbinden jeden Knoten jeweils mit genau den Knoten, die durch Umschalten des CSL-Modus eines der Gelenke von

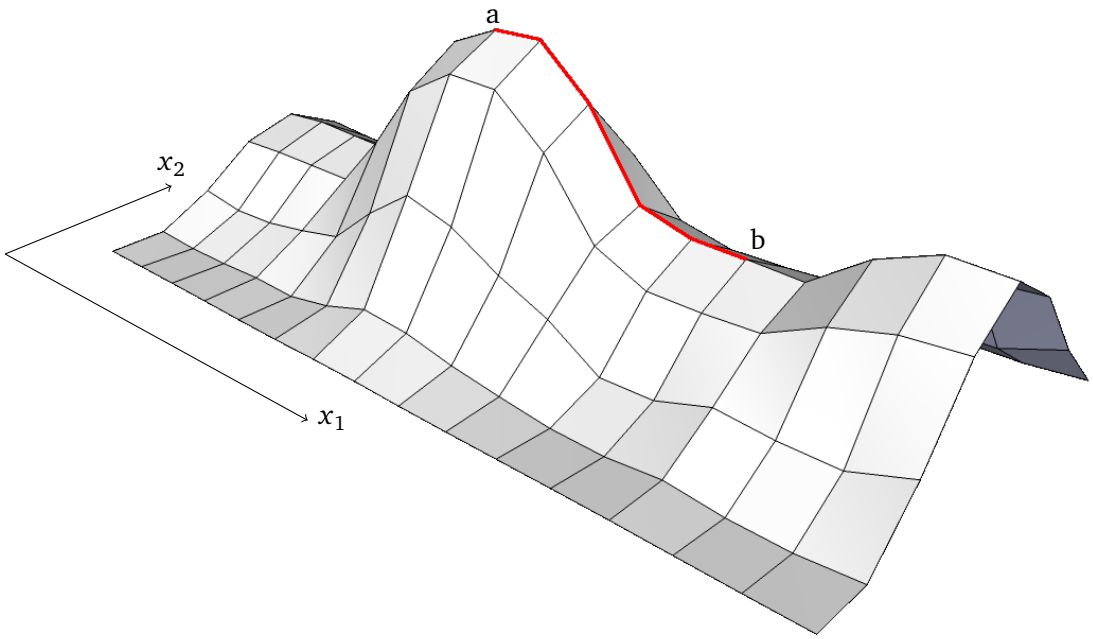

Abbildung 5.1.: Veranschaulichung eines dynamisches Systems mit heteroklinem Orbit. Die beiden Achsen x_1 und x_2 können aktiv geregelt werden, die z-Achse stellt eine dritte passive Größe dar. Es sind zwei instabile Fixpunkte a und b eingezeichnet, zwischen denen ein heterokliner Orbit existiert. Im Punkt a muss in beiden Dimensionen stabilisiert werden, um im instabilen Fixpunkt zu bleiben. Der Fixpunkt b ist in der Dimension x_1 passiv stabil und muss daher nur in der Dimension x_2 aktiv stabilisiert werden. Der Punkt b ist ein Sattelpunkt.

diesem Knoten aus erreicht werden. Die Größe der resultierenden Struktur ist abhängig von der Morphologie und der Anzahl der Gelenke, und kann daher nicht allgemein angegeben werden. Jeder Knoten stellt einen der im System vorhandenen Fixpunkte dar, jedoch können auch verschiedene CSL-Modi den selben Fixpunkt erreichen und stabilisieren und so mehrere verschiedene Knoten den selben Fixpunkt darstellen. Eine bisher nur angedachte Erweiterung wird in Abschnitt 5.3 beschrieben, mit der jeder Fixpunkt nur einmal gespeichert wird. Die Kanten repräsentieren die heteroklinen Orbits, für die es später sinnvoll sein kann, die abgefahrene Trajektorie ebenfalls zu speichern, d. h. $E := (u, v, O)$ mit $O \subset X$. Da für die Bewegung CSLs verwendet werden, werden ausschließlich – u. A. im Hinblick auf die Stabilität – semantische Punkte im Zustandsraum gefunden. Im Vergleich zu anderen Lernverfahren muss nicht der gesamte Zustandsraum durchsucht werden, da durch das Entlangwandern an den heteroklinen Orbits alle Fixpunkte gefunden werden können. Viele spätere Prozesse, wie z. B. eine Dimensionsreduktion, werden unnötig, da die entstehende Struktur bereits dünnbesetzt (engl. *sparse*) ist. Darüber hinaus sind diese Punkte robust gegen Drifteffekte und z. B. auch bei Lageveränderungen adaptiv, so lange CSLs für die Stabilisierung benutzt werden. Diese sind aber prinzipiell auch im Verlauf des Lernprozesses gegen andere adaptive Regelstrukturen austauschbar, wie das in Werner (2013) untersuchte

Kick-and-Fly-Verfahren.

Der Zustandsraum bzw. die Zustandsmannigfaltigkeit kann anhand der Unstetigkeiten des Körperwinkels in Untermannigfaltigkeiten unterteilt werden, siehe Abbildung 5.2. Diese treten auf, wenn statt der langsamen Bewegungen des Motorreglers plötzlich kurzzeitig eine hohe Geschwindigkeit auftritt, die durch eine freie Beschleunigung durch die Gravitation entsteht – ein Umfallen auf eine andere Seite des Körpers, das über hohe Werte der Beschleunigungssensoren einfach detektiert werden kann. Dieses Umfallen ist insofern interessant, dass es einzelne Bereiche von wegzusammenhängenden Punkten unterteilt, innerhalb derer jeder Knoten von jedem anderen über mindestens einen Weg erreichbar ist. Bewegt man sich über eine Kante, bei der der Körper umfällt, ist der Rückweg von der erreichten Untermannigfaltigkeit in die Vorherige nicht immer möglich, insbesondere nicht über den gleichen Weg.

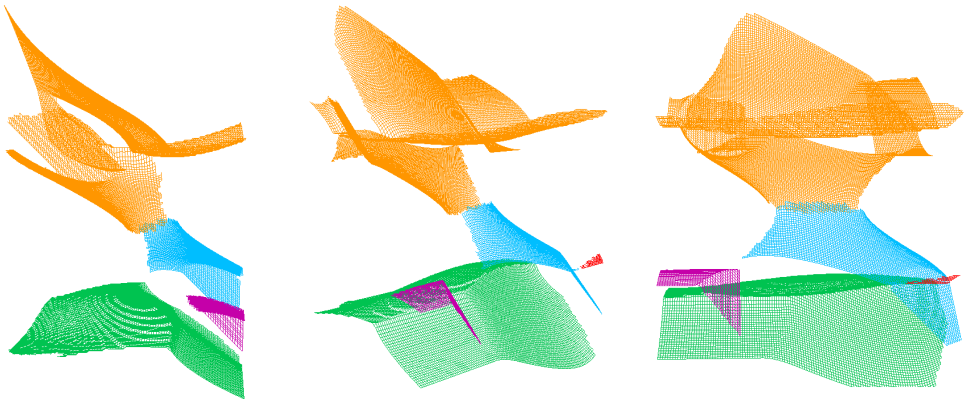

Abbildung 5.2.: Berechnete Mannigfaltigkeit mit Schnitten und Einfärbungen. Links: Frontalansicht mit Hüftwinkel auf der horizontalen Achse und Körperwinkel auf der vertikalen. Mitte: Perspektivische Ansicht mit allen drei Dimensionen – nach links Hüftwinkel, nach rechts Kniewinkel, nach oben Körperwinkel. Rechts: Seitenansicht auf Kniewinkel und Körperwinkel. (Erstellung des Datensatzes Manfred Hild)

Die Anzahl der möglichen Konfigurationen wächst bei Lernverfahren wie z. B. dem Reinforcement Learning exponentiell mit der Anzahl der Gelenke. Da für alle Freiheitsgrade n eine vorher festgelegte Anzahl möglicher Aktionen a versucht werden kann und diese in m verschiedenen Zuständen resultieren können, ist die Anzahl der möglichen Bewegungen $b = (am)^n$. Durch die Betrachtung als dynamisches System und die Eigenschaft der CSLs, nur auf heteroklinen Orbits zwischen zwei Fixpunkten entlangzulaufen, sind die möglichen Aktionen beim ABC-Learning auf vorwärts und rückwärts in jeder Dimension begrenzt. Die Komplexität wächst damit hier nur linear mit der Anzahl der Gelenke, da die einzelnen

CSLs voneinander unabhängig sind. Die Anzahl aller verschiedenen Bewegungen ist damit $b = 2nm$. Für den hier betrachteten Semni bedeutet das, dass es für jeden Knoten vier ausgehende Kanten gibt. Die im nächsten Abschnitt vorgestellten Modi der Gelenke können dies weiter einschränken: Wird eine Posture durch den Modus $s+$ oder $s-$ gehalten, sind nach dem Diagramm in Abbildung 5.3 nur noch drei ausgehende Kanten möglich. Befinden sich beide Gelenke der Posture in einem der beiden Haltemodi, sind zwei ausgehende Kanten möglich. Das Verfahren konvergiert, wenn alle Knoten keine weiteren unbesuchten Nachfolgeknoten mehr besitzen.

5.2. Heuristiken

Nachdem es ermöglicht ist, zwischen Fixpunkten zu wechseln, ist eine Systematik wünschenswert, die möglichst effektiv die gesamte Mannigfaltigkeit erkundet. Anstatt dafür direkt Parametervektoren der verschiedenen CSL-Modi zu betrachten, werden im Folgenden Bewegungen auf den Untermannigfaltigkeiten in positiver und negativer Richtung entlang der heteroklinen Orbits in jeder Dimension angestrebt und die dafür nötigen CSL-Modi in einer darunter liegenden Ebene bestimmt. Mittels der verschiedenen CSL-Modi ist es möglich, von instabilen zu stabilen Fixpunkten zu wechseln (Release) und von stabilen zu instabilen (Contraction). Dabei kommt es jedoch zu zwei Besonderheiten. Betrachtet man die rechte Hälfte von Abbildung 3.1, ist zu erkennen, dass von einem stabilen Fixpunkt mehrere über heterokline Orbits erreichbare instabile Fixpunkte existieren können. Für Rotationsgelenke in der Robotik bedeutet das anschaulich, dass ein Gelenk in positiver wie in negativer Richtung in einen instabilen Zustand gelangen kann – und nicht selten ist das der Fall. Aus einem stabilen Fixpunkt heraus verhält sich der Contraction-Modus dabei von außen betrachtet zufällig, z. B. in Abhängigkeit von Rauschen oder thermischen Effekten. Zudem kann ein Fixpunkt weniger punktförmig sein und stattdessen als zusammenhängende Menge von Fixpunkten, einer sogenannten Ruhemenge vorliegen, (siehe Markl, 2010, S. 7 ff.). Die linke Hälfte von Abbildung 3.1 zeigt daher auch Fixpunktpaare, die die Grenzen einer solchen Menge darstellen. Für ein deterministisches Verhalten ist es daher nötig, eine Symmetriebrechung im instabilen Fixpunkt durch den Release-Modus vorzunehmen. Dafür wird für eine gewünschte positive oder negative Bewegungsrichtung des Contraction-Modus jeweils eine leichte Vorspannung beim vorherigen Release-Modus über den Parameter g_{b} erzeugt, die so gut wie keine Positionsveränderung bewirkt. Beim Umschalten zum Contraction-Modus entfällt diese Spannung wieder und es entsteht eine leichte Bewegung in die andere Richtung, die am Eingang des CSLs als kleine Geschwindigkeit anliegt und gegen die der Contraction-Modus daraufhin arbeitet. Die gewünschte Richtung kann so also im Vorhinein festgelegt werden. Anstatt des bisherigen Modus mit der Release-Eigenschaft erhalten wir also zwei Modi, die mit $r+$ und $r-$ abgekürzt werden sollen.

Die zweite Besonderheit zeigt sich beim Contraction-Modus. Dieser hat die Eigenschaft, gegen alle von Außen wirkenden Kräfte zu arbeiten. Dies geschieht, wie in Abschnitt 3.3 erläutert, als interessanter Verhaltensmodus gegen die Schwerkraft beim Heben einer Gliedmaße, beim Aufrichten eines mobilen Roboters indirekt durch Abdrücken am Boden und auch beispielsweise bei Interaktion mit dem Menschen. Gelangt innerhalb einer Bewegung ein Gelenk in den Zustand, in dem der Integrator nicht entladen ist, eine weitere Bewegung aber Aufgrund eines Winkelanschlages nicht möglich ist, ist das Eingangssignal bei 0 und der Integrator verstärkt sich selbst. Der Wert am Ausgang wächst damit immer weiter und man läuft Gefahr, die Motoren oder die Mechanik zu beschädigen. Anstatt aber diesen Zustand nur zu verhindern, kann auch diese Position später semantisch verwendet werden. Daher werden zwei Haltemodi eingeführt, die eine solche Blockierung kodieren und die Position energiearm halten. Diese werden mit $s+$ und $s-$ (für engl. *stall*) bezeichnet. Diese Wahrnehmung kann mit dem natürlichen Schmerz verglichen werden, der zuvor erlebten Gefahrensituationen vorbeugt.

Um nun ausgehend von einer angestrebten Richtung und dem aktuellen Modus eines Gelenkes einen neuen Parametervektor zu erhalten, ist der in Abbildung 5.3 gegebene Zustandsautomat hilfreich. Um Beispielsweise einen instabilen Fixpunkt, der durch den Contraction-Modus gehalten wird, mit der angestrebten Richtung + zu verlassen, ergibt es sich, die Parameter für den Release-Modus $r+$ zu setzen.

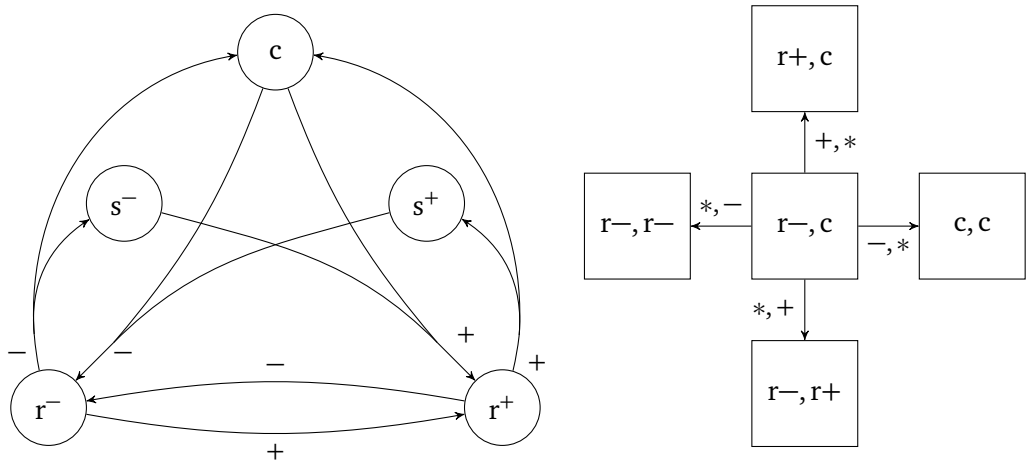

Abbildung 5.3.: Links: Zustandsübergangsdiagramm zur Ermittlung des nächsten CSL-Modus aus dem aktuellen für eine bestimmte Richtung bei einem Gelenk. Zwischen den verschiedenen Zuständen von c und s wird abhängig von einer Blockierungserkennung während der Bewegung entschieden. Es ist die Annahme enthalten, dass es an einem Winkelanschlag nicht sinnvoll ist, erneut in die Richtung des Anschlages zu explorieren, und es wird stattdessen der passive Release-Modus gewählt. Rechts: Beispielhafte Modusübergänge mit angestrebter Richtung für zwei Gelenke.

Wenn jedes Gelenk als eindimensional angenommen wird, ist es nach dem Speichern einer neuen Posture nur noch nötig zu entscheiden, welche Richtung bei welchem Gelenk als nächstes eingeschlagen werden soll. Jeder Knoten hat also bei n Gelenken maximal $2n$ ausgehende Kanten. Da im Vorhinein nicht bekannt ist, wie die Struktur der möglichen Bewegungen gestaltet ist, kann eine Heuristik aufgebaut werden, die Anhand des Vorwissens durch bereits erfolgte Erkundung weitere Entscheidungen trifft. Einige Möglichkeiten sind in Tabelle 5.1 gegenübergestellt, deren einzelne Mechanismen im Folgenden beschrieben werden.

Tabelle 5.1.: Übersicht über einige mögliche Heuristiken zur Wahl des nächsten Parametersatzes und der jeweiligen Eigenschaften.

Heuristik	Eigenschaften
Wähle ein Gelenk und einen neuen Modus jeweils zufällig	Findet schwer erreichbare Postures bzw. schwach vernetzte Knoten erst nach unter Umständen sehr langer Zeit. Es werden Postures wiederholt, die bereits exploriert sind. Kann in lokalen Minima verharren.
Wähle für ein zufällig gewähltes Gelenk zufällig eine Richtung	
Wähle möglichst noch unbesuchte Kanten des aktuellen Knotens	Wiederholt für den aktuellen Knoten keine Kanten, so lange noch nicht alle besucht sind. Die Auswahl der unbesuchten Kanten kann nach verschiedenen Regeln erfolgen. Kann in lokalen Minima verharren.
Wähle möglichst unbesuchte Kanten und wähle Kanten andernfalls anhand von zusätzlichem Diffusionsprozess, der den Weg zum am wenigsten explorierten Teilgraphen bevorzugt	Wiederholt Postures nur, wenn bereits alle Kanten im aktuellen Knoten besucht sind und diese auf dem Weg zu weiteren noch unerkundeten Kanten liegen. Findet auch schlecht vernetzte Postures auf dem kürzesten Weg.

Die einfachsten Heuristiken sind solche, die keinerlei bereits vorhandene Informationen nutzen, zum Beispiel indem zufällig eine der vorhandenen Möglichkeiten auswählt wird. Die Exploration des gesamten Suchraumes ist damit bereits möglich, jedoch ist der Zeitbedarf je nach dessen Beschaffenheit unnötig hoch. Kanten werden auch gewählt, wenn sie bereits exploriert sind und es kann zu lokalen Minima kommen, während wenig verbundene Gebiete nur schwer erreicht werden. Um zuerst möglichst viele noch unbekannte Möglichkeiten

zu versuchen, und damit mittels der Heuristik implizit eine Lernmotivation vorzugeben, kann auf das erzeugte Wissen zurückgegriffen werden, das bei der Wahl einer Richtung entsteht: Bereits besuchte Richtungen werden erst dann erneut gewählt, wenn ausgehend vom aktuellen Knoten alle möglichen Kanten existieren.

Sind bereits alle Kanten besucht kann eine der Kanten zufällig gewählt werden. Zählt man mit, wie oft Kanten gewählt worden sind, kann auch dann, wenn alle Kanten existieren, jene Richtung gewählt werden, die bisher am wenigsten exploriert wurde. Ein besseres Vorgehen ist es jedoch, eine Kante mittels eines Diffusionsmechanismus zu wählen.

Das Bestreben, in möglichst noch unbekannte Bereiche des Graphen zu gelangen, wird bisher nur in den Knoten lokal durch die Wahl der nächsten unbesuchten Kante erreicht. Zusätzlich ist es aber auch möglich, das bereits gelernte Wissen aus dem gesamten Graphen zu verwenden. Wenn im Verlauf der Exploration für eine aktuelle Posture lokal keine unbesuchten ausgehenden Kanten mehr existieren, gelangt man über eine der bereits besuchten Kanten zu anderen nachfolgenden Knoten. Es ist wünschenswert, genau die Kante zu wählen, welche zu noch möglichst wenig bekannten Bereichen der Mannigfaltigkeit führt, das heißt Knoten mit noch möglichst vielen nicht besuchten Kanten findet. Um eine geeignete Wahl unter den besuchten Kanten zu treffen, wird dafür in jedem Zeitschritt der Simulation ein Diffusionsprozess berechnet, durch welchen jedem Knoten lokal eine Eigenaktivität a_k zugeordnet wird, die den Wert 1 besitzt, solange noch keine der von diesem Knoten aus möglichen Kanten ausprobiert worden ist. Der Wert wird auf 0 gesetzt, wenn bereits alle Kanten exploriert sind. Abstufungen ergeben sich aus der Anzahl der Gelenke und der Anzahl der bereits besuchten Kanten. Ist bereits die Hälfte der Möglichkeiten exploriert, ergibt sich beispielsweise eine Eigenaktivität von 0,5. Die Gesamtaktivität des Knotens ergibt sich aus der mit c_i gewichteten Summe dieser Eigenaktivität und dem mit c_e gewichteten Mittelwert der Aktivitäten aller Nachfolgerknoten, d. h. es existiert eine Kante zu diesen Knoten. Diese Dabei sollte die Summe der beiden Faktoren c_i und c_e stets 1 sein, damit die Aktivität nicht verstärkt wird oder verloren geht, die Werte für jeden Knoten also konvergieren. Die Aktivität der erreichbaren Knoten fließt also rückwärts durch den Graphen.

Die ausgehenden Kanten sind für jeden Knoten die möglichen Wege, neues Wissen zu erlangen. Je größer der Aktivitätswert des über eine Kante erreichbaren Knotens ist, desto wahrscheinlicher ist es, in dieser Richtung noch unbekannte Postures zu entdecken. Der Parameter c_e sollte sehr viel kleiner als c_i gewählt werden, so dass die Aktivität, die so von einem unterexplorierten Knoten in besser erkundete Bereiche des Graphen fließt, schnell genug abnimmt und sich verschiedene Aktivitätsniveaus bilden können. Damit wird auf dem Weg zu weiter entfernten unerkundeten Knoten zudem ein Weg gewählt, der Knoten besucht, die ebenfalls noch gering erkundet sind. Auch wenn es sich nicht vermeiden lässt, irgendwann Knoten erneut zu besuchen, die bereits vollständig erkundet sind, können bei wiederholten Transitionen z. B. die gespeicherten Sensorwerte verbessert werden, die mit Rauschen oder anderen stochastischen Effekten behaftet sind.

Die Aktivität a_k für den Knoten k im Posturegraphen $\mathcal{G} = \{V, E\}$ bei n Freiheitsgraden berechnet sich durch

$$a_k := c_\mathrm{i} M_k \underbrace{+ \frac{c_\mathrm{e} \sum_{\{k,j\} \in O_k} a_j}{|O_k|}}_{\text{wenn } O_k \neq 0} \tag{5.1}$$

wobei S_k die Anzahl der Gelenke von Posture k mit CSL-Modus $m \in \{s+, s-\}$,

$$D_k := 2n - S_k$$

die maximale Anzahl ausgehender Kanten des Knotens k,

$$M_k := \begin{cases} 1, & \text{wenn Grad des Knotens } g(k) = 0 \\ 1 - \frac{g(k)}{D_k}, & \text{sonst} \end{cases}$$

die Eigenaktivität des Knotens k und

$$O_k := \{\{u, v\} \in E \mid u = k\}$$

die Menge der von Knoten k ausgehenden Kanten ist.

Die Folge ist eine sehr viel kürzere Zeit für die Exploration des Graphen, besonders eine Reduktion der Gesamtzeit, die für das Besuchen aller möglichen Postures benötigt wird. Auch nur über einen Knoten erreichbare Bereiche werden so zielgerichtet gefunden.

Während die günstigste Kante eindeutig bestimmt werden kann, wenn bereits jede Richtung einmal gewählt wurde, muss im anderen Fall – es sind noch Richtungen unbesucht – eine Auswahl getroffen werden. Da über die Auswirkung der unbesuchten ausgehenden Kanten noch nichts bekannt ist, kann beispielsweise in einer festen Reihenfolge gewählt werden oder es kann eine Entscheidung anhand des zuvor verwendeten Gelenkes oder der zuvor gegangenen Richtung getroffen werden. Bei der Wahl, welche der noch nicht explorierten Kanten als nächste gewählt wird, wurden die in Tabelle 5.2 aufgeführten Varianten untersucht.

Es sind neben diesen Varianten aber auch weitere Heuristiken denkbar, die beispielsweise auf die Morphologie abgestimmte Kriterien einbeziehen. Beim Semni haben die beiden Gelenke verschiedene Auswirkungen auf Wechsel zwischen den Untermannigfaltigkeiten. Diese werden hauptsächlich durch Bewegungen des Schultergelenks erreicht, während das Kniegelenk seinen Modus beibehält. Es wäre hier also eine weitere Möglichkeit, abwechselnd die Richtung und das Gelenk zu verändern. Dies wurde aber in dieser Arbeit aus Zeitgründen nicht untersucht.

Wenn eine der Aktionen der jeweiligen Varianten nicht möglich ist, weil die resultierende Kante bereits exploriert ist, kann entweder zufällig eine noch unbesuchte Kante gewählt

Tabelle 5.2.: Betrachtete Varianten der Heuristik.

Variante	Strategie
1	Behalte die angestrebte Richtung des CSL-Reglers bei.
2	Behalte das Gelenk bei, für welches der CSL-Modus geändert wird.
3	Wechsle die Richtung nach Besuchen eines Fixpunktes.
4	Wechsle das Gelenk nach Besuchen eines Fixpunktes.
1 + 2	Wenn möglich, wechsle weder das Gelenk noch die Richtung.
2 + 3	Wenn möglich, wechsle die Richtung und behalte das zuvor bewegte Gelenk bei.
1 + 4	Wenn möglich, wechsle das Gelenk und behalte die letzte Richtung bei.
3 + 4	Wenn möglich, wechsle das Gelenk und die Richtung.

werden oder es wird eine alternative Aktion gewählt, die eine andere Kante bestimmt. Die Reihenfolge dieser Alternativen kann wiederum für jede Variante separat festgelegt werden.

Je nachdem wie viele Gelenke das System besitzt, sind für Varianten mit Gelenkwechsel mehrere Wechsel möglich, solange bis alle Gelenke verwendet worden sind.

Ein exemplarischer Posturegraph nach einem kurzen Explorationslauf mit der Heuristik $2+3$ mit den jeweilig bestimmten Aktivitätswerten ist in Abbildung 5.4 dargestellt. Die beschriebenen Heuristiken wurden auf beiden Plattformen implementiert und werden Qualitativ für Ergebnisse vom Simulator Simni in Kapitel 7 verglichen und diskutiert.

5.3. Erweiterungen

Betrachtet man die resultierenden Graphen, z. B. die linke Darstellung von Abbildung 5.5 sieht man schnell, dass viele Postures den gleichen Fixpunkt speichern und lediglich der dorthin führende Modus ein anderer ist, im Beispiel Knoten 21 bis 27. Das passiert besonders bei stabilen Fixpunkten, bei denen eine Änderung von einem Release-Modus zu einem anderen keine Bewegung hervorruft, aber auch wie hier an einer Grenze der Untermannigfaltigkeit, die durch den unüberwindbaren Kontakt mit dem eigenen Körper entsteht. Eine Möglichkeit, jeden Fixpunkt nur einmal abzuspeichern ist es, Knoten mit verschiedenen Modi und dem selben Fixpunkt als einen Knoten zusammenzufassen. Wenn es zwei Knoten gibt, deren Konfiguration nahezu identisch ist – eine leichte Bewegung durch Vorspannung kann toleriert

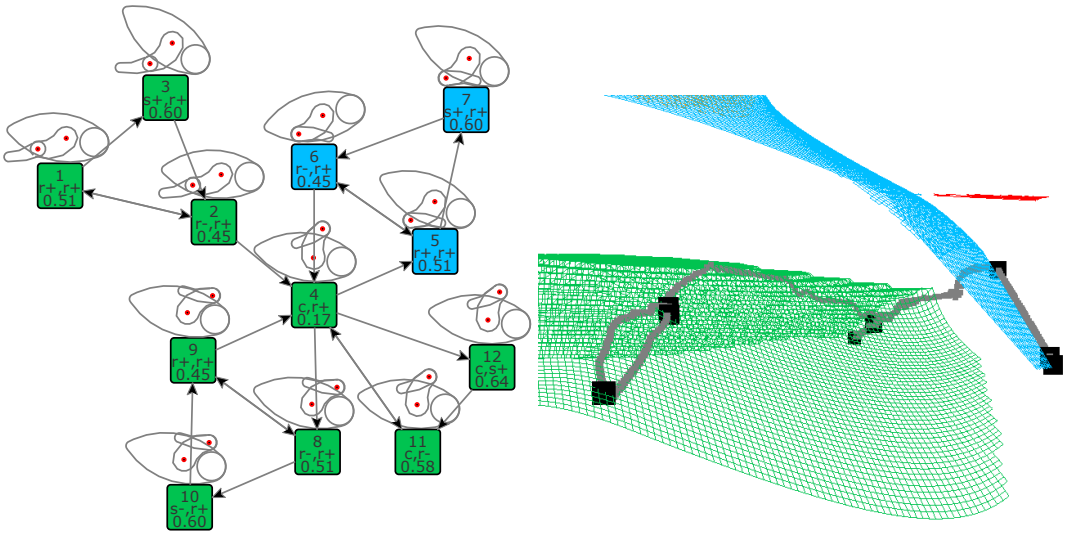

Abbildung 5.4.: Links: Beispielhafter Explorationsgraph einer kurzen Exploration mit Aktivitätswerten (der letzte Wert in jedem Knoten). Die erste Zeile gibt den Namen der Knoten als eine laufende Nummer an. Die Semni-Zeichnungen auf den Knoten geben die resultierende Konfiguration grafisch an. In der zweiten Zeile ist der Modus aufgeführt, der die Posture hält. Es ist zu erkennen, dass Knoten mit höherer Vernetzung geringere Aktivitätswerte besitzen (z. B. Knoten 2 und 4). Zu diesem Zeitpunkt haben bis auf Knoten 4 noch alle Knoten offene Kanten. Knoten 4 besitzt eine Eigenaktivität von 0, erhält aber noch Aktivität der nachfolgenden Knoten. Die Farben der Knoten geben verschiedene Untermannigfaltigkeiten an. Rechts: Im Bild sind die Fixpunkte jedes Knotens innerhalb der Mannigfaltigkeit als schwarze Quadrate und die dazwischen liegenden Trajektorien in grau gekennzeichnet. Die Trajektorieren liegen nicht immer exakt auf der Mannigfaltigkeit, da diese nur für den statischen Fall berechnet ist und leichte unterschiede zwischen Rechnung und tatsächlichen Messwerten bestehen.

werden – sollte es zwischen ihnen in beide Richtungen eine verbindende Kante geben, die es erlaubt zwischen diesen Modi beliebig zu wechseln. Jeder weitere Knoten, der mit einem dieser Knoten bidirektional verbunden ist und die gleiche Konfiguration enthält, ist folglich auch direkt oder indirekt erreichbar und kann ebenfalls hinzugenommen werden. Weitere Knoten, die nur über eine zu ihnen hinführenden Kante verbunden sind (im Beispiel Knoten 25, 26 und 27), sollten nicht mit einbezogen werden, da noch nicht gesichert ist, ob diese während der weiteren Exploration eine rückführende Kante erhalten. Ist dies nicht der Fall, wäre es nicht möglich über einen solchen Knoten die von den anderen gruppierten Knoten zuvor erreichbaren Kanten zu besuchen.

Ersetzt man all diese Knoten nun durch einen einzigen mit der gleichen Konfiguration, können jedoch nicht einfach die jeweils zuvor ausgehenden Kanten als neue ausgehende Kanten des neuen Gruppenknotens verwendet werden. Bei einem erneuten Besuch dieses Knotens wäre es nun nicht mehr genauso möglich, alle nachfolgenden Knoten zu erreichen, da die resultierende Trajektorie nicht nur von der alten und der neuen Konfiguration sondern auch

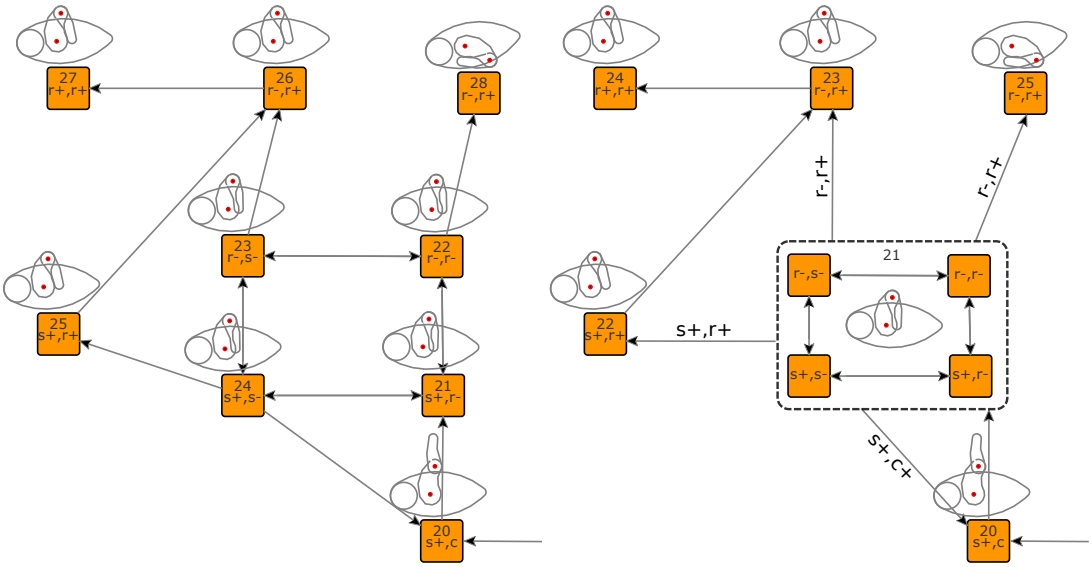

Abbildung 5.5.: Links: Ungruppierter Ausschnitt aus einem Explorationsraphen. Rechts: Gruppieren von zusammenhängenden Knoten, die den selben Fixpunkt speichern. Die Knoten 25, 26 und 27 in der linken Abbildung können je nach Verbindungen bei späterer Exploration auch in die Gruppe aufgenommen werden.

dem vorherigen CSL-Modus abhängt, welche bisher in jedem Knoten separat gespeichert war. Daher ist es eine Möglichkeit, zusätzlich zum zweigeteilten Release-Modus auch den Contraction-Modus in einen $c+$ und einen $c-$ zu spalten, welche so implementiert werden können, dass vor dem Setzen der Parameter für das Contraction-Verhalten für eine kurze Zeit eine Vorspannung in die jeweilige Richtung ermöglicht wird. Weiterhin wird der Modus für eine Bewegung in jeder Kante gespeichert und beim erneuten Entlangwandern auch wieder von dort gelesen. Damit ist es möglich, dass ein solcher gruppierter Knoten alle aus- und eingehenden Kanten erhält, die seinen enthaltenen Knoten zuvor einzeln angehörten. Die Suche nach Knoten, die gruppiert werden können, und das modifizieren des Graphen sollte in einer “Schlafphase” nach einer Exploration geschehen, in der keine Bewegungen stattfinden. Damit kann verwendeter Speicherplatz wieder reduziert werden und die gespeicherten Informationen enthalten nur die wirklich notwendigen Erfahrungen. Innerhalb dieser Ruhephase können ebenfalls weitere Aufräumarbeiten geschehen, die abgespeicherte Daten kombinieren und reduzieren. Speichert man während der Exploration an jeder Kante die Winkelwerte aller Gelenke, also die jeweiligen Trajektorien und die dazugehörige Stromaufnahme der Motoren, können bei erneutem Besuchen der Kante Schätzungen über das Erreichen des Fixpunktes berechnet werden oder eine Optimierung der Reglerparameter daraus abgeleitet werden. Die Winkelwerte können später wieder gelöscht werden um Speicherplatz zu sparen. Bisher wurden gruppierte Knoten noch nicht in einer Implementation getestet.

5.4. Exploration mit dem Roboter

Im Folgenden werden die Resultate eines vollständigen Explorationslaufes[6] im Simulator Simni gezeigt, mit dem die gesamte Mannigfaltigkeit erkundet ist. Die Anzahl der Knoten ist hier mit 152 vergleichsweise hoch. Die Anzahl der eigentlichen Fixpunkte entspricht der in (Hild, 2011) genannten Zahl von 33 Postures[7]. Der Graph speichert jedoch, wie bereits erläutert, Fixpunkte auch mehrfach ab. Die benötigte Zeit zur Exploration ist abhängig von den Detektoren für Fixpunkte und den Parametern der verwendeten Regler, bewegt sich aber immer unterhalb weniger Stunden. Die besuchten Untermannigfaltigkeiten lassen sich in die drei Körperlagen “Bauchlage” (Grün), “Rückenlage” (Orange) und “Stütz” auf dem Unterarm (hellblau) unterteilen. Prinzipiell sind auch die anderen beiden Untermannigfaltigkeiten, die in Abbildung 5.2 dargestellt sind, erreichbar. Diese werden hier jedoch aufgrund von Unterschieden zwischen analytischer Berechnung und dynamischer Simulation nicht erreicht und sind ebenfalls von der konkreten Einstellung der CSL-Parameter abhängig.

Zwischen der orangen und hellblauen Untermannigfaltigkeit existieren nur zwei Kanten, die zur orangen Untermannigfaltigkeit führen, der Rückweg ist nur über eine Kante zur grünen Untermannigfaltigkeit möglich, von der einige Kanten zur hellblauen existieren. Es wird also sichtbar, dass die Trajektorien, die den Roboter zum Umfallen und Verlassen der Untermannigfaltigkeit bringen nur selten angefahren werden. Es ist also zu erwarten, dass die Exploration für diese Morphologie in dem Wechsel der Untermannigfaltigkeiten von der Entscheidungsheuristik weniger stark beeinflusst wird, da aus dieser gewissen Isolation erst das Finden der richtigen Kante befreit. Die starke Trennung der Untermannigfaltigkeiten ist durch die ellipsoide Form des Semni zu erklären, so dass für den Wechsel ein großer Teil des Körpers angehoben und gedreht werden muss und die relativ kurzen Arme dazu nur in wenigen Situationen geeignet sind.

[6] Vollständig soll insofern verstanden werden, dass es keine weiteren Knoten gibt, die durch das Explorationsverfahren über einen der vorhanden Knoten erreicht werden können. Jeder Knoten hat damit seine maximale Anzahl von ausgehenden Kanten und der Aktivitätswert jedes Knotens ist 0.

[7] Das Verhalten an einigen kritischen Punkten hängt von der konkreten Implementation ab und es können so auch weniger Fixpunkte gefunden werden.

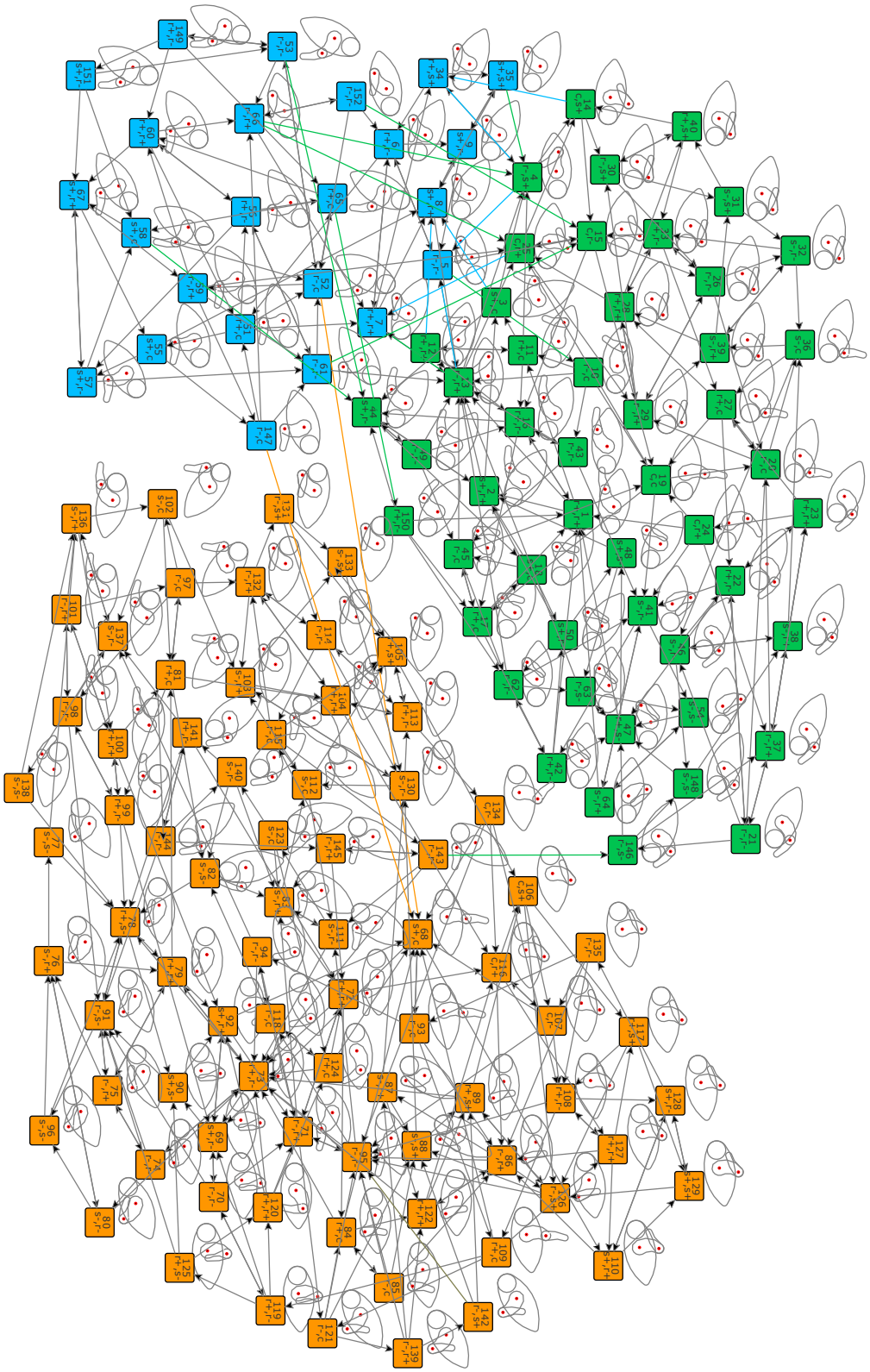

Abbildung 5.6.: Beispielhafter Explorationsgraph einer vollständigen Exploration. Zur Verdeutlichung der Grenzen der Untermannigfaltigkeiten sind Kanten, die einen Wechsel der Untermannigfaltigkeit erwirken, entsprechend eingefärbt.

6. Implementation des Lernverfahrens

Die Implementation des ABC-Lernverfahrens erfordert neben einer Heuristik auf der abstrakteren Ebene zuerst verschiedene Detektoren zur Erkennung der unterschiedlichen Situationen in denen Wissen gespeichert wird und Methoden, die Sensordaten aufbereiten, Mehrdeutigkeiten disambiguieren und mit Limitationen der Plattform umgehen. Für den Simulator Simni und den Roboter Semni traten dabei verschiedene Spezialfälle auf.

6.1. Simulation

In diesem Abschnitt werden einige Details zur Implementation des CSLs und des ABC-Lernverfahrens im bereits beschriebenen Simulator Simni erläutert.

Der Simulator berechnet die physikalische Simulation mit einer Frequenz von ∼1000 Hz. Die Frequenz der Ansteuerung der Motoren sollte um einiges geringer gewählt werden, damit die Motorausgangswerte zuerst einen Effekt erzielen können, bevor dieser als Rückkopplung in den Regelkreis zurückfließt und es so nicht zu Selbstoszillationen kommt. Das Lernverfahren benötigt eine Implementation der CSL-Regelschleife. Diese lässt sich rechnerisch unaufwändig mittels Gleichung 3.8 berechnen und wird hier mit einer Frequenz von 100 Hz aktualisiert, welche ebenfalls die Frequenz der Hardwareimplementation für den Semni ist.

Die Exploration besteht im Wesentlichen aus zwei Schritten: im ersten Schritt werden die CSL-Modi gesetzt und danach das Gesamtsystem aus Regler und Roboter sich selbst überlassen. Im zweiten Schritt ist es nötig, zu erkennen dass dadurch ein Fixpunkt erreicht ist und diesen zu speichern. Zur Erkennung wurde im Simulator ein naiver Ansatz versucht, der direkt mit der Definition für Fixpunkte bzw. Attraktoren ableitbar ist. Es wird der Zustand des Systems über eine gewisse Zeitspanne beobachtet und ein Attraktor daran erkannt, dass sich ein p-Orbit abzeichnet, das heißt Teile der Trajektorie mehrmals in der gesamten Spanne zu finden sind. Der Aufwand zur Parameteranpassung ist dabei gering. Es werden die Winkelwerte der Gelenke und des Körpers für die letzten 10 Sekunden gespeichert und ein Ausschnitt der letzten 50 Werte innerhalb der restlichen Historie gesucht. Der Wertevergleich beinhaltet eine kleine Toleranzschwelle. Wird dieser Ausschnitt gefunden, handelt es sich um

einen stabilen oder instabilen Fixpunkt. Andere Kriterien sind möglich, die nach weiteren Wiederholungen suchen oder spezifisch auf den Verlauf der CSL-Ausgangswerte schauen, siehe Abschnitt 6.2.

Es kann die Situation entstehen, dass die beim Umschalten der CSLs resultierenden Trajektorien bei konsekutiven Wanderungen entlang der heteroklinen Orbits nicht immer identisch sind, so dass das System abhängig von der Startkonfiguration innerhalb des gleichen Attraktors in einem weiteren Fixpunkt endet, statt in einem eigentlich bereits bekannten Fixpunkt. Dies ist einerseits möglich, wenn durch die Haftreibung der instabile Fixpunkt zu einer Ruhemenge verwischt und der Detektor dadurch an verschiedenen Stellen einen Fixpunkt erkennt – abhängig davon, in welchem Zustand sich das System vor dem Wechsel zu diesem Fixpunkt befunden hat. Dies lässt sich mit geschickteren Detektoren vermeiden, die einen Mittelpunkt der Ruhemenge bestimmen. Andererseits kommt es aber auch dazu, dass das System in einem Knoten auf Trajektorien mit kleinen Unterschieden ankommt, weil sich z. B. die Akkuspannung verändert hat und damit die Startkonfiguration zur nächsten Posture unterschiedlich ist. Durch diese Unterschiede kommt es z. B. zu einer Berührung des Bodens oder des eigenen Körpers mit den Armen, die bei einem anderen Mal nicht stattgefunden hat. Da das ABC-Verfahren aber darauf setzt, dass sich die Bewegung durch die Mannigfaltigkeit deterministisch steuern lässt, ist es nötig dieses Verhalten möglichst zu vermeiden. Sofern es möglich ist, sollte zuerst versucht werden, die Ansteuerung zu optimieren. Es sollten die CSL-Parameter so angepasst werden, dass es in instabilen Fixpunkten zu möglichst wenig Eigenbewegung kommt und der Detektor einen Fixpunkt mit nur geringer Streuung erkennen kann. Bei größerer Haftreibung ist das aber nur begrenzt möglich. Im Simulator Simni wurde daher eine Strategie verwendet, die in einer unsicheren Situation ein Ausschlussverfahren verwendet.

Gibt es einen Knoten im Graph, der dem aktuellen Knoten folgt und der durch den aktuellen CSL-Modus erwartet werden kann, so sollte mit dem nächsten erkannten Fixpunkt dieser Knoten gefunden werden. Schlägt diese Überprüfung fehl, so ist die zuvor beschriebene Situation eingetreten und der Fixpunkt wird nicht als neuer Knoten gespeichert. Stattdessen wird überprüft, ob der erwartete Knoten auch durch einen Positionsregler[8] erreicht werden kann. Dadurch wird sichergestellt, dass der gespeicherte Knoten tatsächlich hätte erreicht werden sollen und dass sich der Roboter in der gleichen Untermannigfaltigkeit befindet. Bei Erreichen des gespeicherten Fixpunktes durch den Positionsregler wird wieder auf den CSL-Regler umgestellt, so dass dessen Fixpunkt erkannt und gespeichert werden kann. Damit sind Mehrdeutigkeiten in der Struktur umgangen, jedoch ist nicht unbedingt gesagt, welcher der verschiedenen Fixpunkte der “bessere” ist. Es wäre alternativ möglich, beide Fixpunkte zu behalten und sich anhand der Häufigkeit der wiederholten Besuche später für einen zu entscheiden. Zudem kann vor dem Wechsel des CSL-Modus für den nächsten

[8] Der Positionsregler ist im Simni durch einen PI-Regler implementiert, der den aktuellen Winkel und einen Zielwinkel als Eingangsgrößen erhält. Die Parameter P und I wurden von Hand eingestellt.

Knoten darauf gewartet werden, dass das in leichter Bewegung befindliche CSL, abhängig davon ob sich dieses im Contraction-Modus oder im Release-Modus befindet, die im Knoten gespeicherte zuvor verwendete Konfiguration möglichst annimmt und so von der gleichen Startkonfiguration zum erwarteten Knoten traversiert. Wird der erwartete Knoten durch den Positionsregler nicht erreicht, wird die Exploration an einem anderen Ort fortgeführt, ohne dabei Verbindungen zum vorherigen Knoten herzustellen.

Während die Parameter für den Contraction-Modus als konstante Werte zu bestimmen sind[9] ist der Release-Modus mit einem festen positiven oder negativen Biaswert zwar vorgespannt und leistet so die Bestimmung der resultierenden Richtung des nachfolgenden Contraction-Modus. Gleichzeitig muss dadurch aber auch eine Symmetriebrechung in einem instabilen Fixpunkt erfolgen, der Biaswert also hoch genug gewählt werden, so dass in allen möglichen Situationen die Haftreibung überwunden wird. Da die Haftreibung in der Praxis meist sehr viel höher als die erst mit der Bewegung einsetzende trockene Reibung ist (vgl. Abschnitt 4.1.3), und hier speziell die Haftreibung der Motoren auch im gleichen Bereich ist, wie die Kräfte, die für manche der Bewegungen notwendig sind, erfolgt mit diesem Biaswert in ungünstigen Situationen auch eine Bewegung des ganzen Roboters, also weit mehr als eine leichte Vorspannung der Gelenke oder ein einfaches “Fallenlassen” des Armes aus einem instabilen Fixpunkt heraus. Dafür kann, wie im nächsten Abschnitt beschrieben, ein Regler für eine konstante Geschwindigkeit verwendet werden. Eine weitere, innerhalb des Simulators ausreichende, Möglichkeit ist es, zwei verschiedene Biaswerte festzulegen, von denen einer groß genug ist, die Haftreibung zu überwinden. Sobald eine Bewegung begonnen hat, wird der zweite Biaswert gesetzt, der nur die Bewegung aufrecht erhält und die Vorspannung leistet.

Die für diese Arbeit verwendeten Grafiken wurden zum großen Teil mit Simni erzeugt, insbesondere die Graphen in Abschnitt 7 und die Ansichten der analytischen Mannigfaltigkeit.

6.2. Hardware

Die Implementation des ABC-Verfahrens auf der Hardware ist mit den Erfahrungen der Simulation entstanden. So wurden andere Detektoren verwendet und beschriebene Spezialfälle vermieden. Gleichzeitig bedeutet die Arbeit mit den nun echten Sensoren auch, dass der Körperwinkel nur über den dreiachsigen Beschleuningungssensor bestimmt werden kann, dessen Daten immer mit Störungen einhergehen. Dieser befindet sich zudem innerhalb der runden Aussparung, die einem Kopf ähnelt. Die Drehung des Körpers geschieht jedoch immer

[9] Die verwendeten Parameter für den Contraction-Modus sind für beide Gelenke $g_\mathrm{i} = 3$ und $g_\mathrm{f} = 1.01$. Der Biaswert für die Release-Modi ist $g_\mathrm{b} = 0.02$.

um einen Auflagepunkt der elliptischen Außenkontur, so dass der Winkel nur annähernd genau bestimmt werden. Der Winkel kann berechnet werden, indem man von den vorhandenen drei Achsen nur die yz-Ebene betrachtet, die genau in der Bewegungsebene des Roboters liegt. Der Körperwinkel α lässt sich mittels des Arkustangens und den Skalierungswerten n_y und n_z berechnen:

$$\alpha := \arctan(\frac{a_y/n_y}{a_z/n_z}). \tag{6.1}$$

Statt des Arkustangens, der in dieser Form keinen kontinuierlichen Winkel berechnet, kann in einer Implementation auch die in Programmiersprachen übliche Bibliotheksfunktion atan2(y,x) verwendet werden, welche mittels einer Fallunterscheidung den Winkel im richtigen Quadranten berechnet.

Die Erkennung der Fixpunkte betrachtet statt der Trajektorien in der Implementation des Simni hier die Winkelgeschwindigkeiten der Gelenke und ordnet bei den Release- und Contraction-Modi und resultierenden Geschwindigkeit nahe Null der aktuellen Konfiguration einen Fixpunkt zu. Für den Contraction-Modus werden zusätzlich die gemessenen Motorströme betrachtet, die aus der Ansteuerung resultieren. Diese geben Auskunft über das tatsächlich aufgebrachte Drehmoment und erlauben damit zu erkennen, ob ein CSL im Contraction-Modus noch Kraft aufbringt, die zwar keine Bewegung mehr erzeugt, aber noch gegen die Haftreibung arbeitet (vgl. 4.1.4). Ist der gemessene Strom unter einem vorher bestimmten Strom, der für das Gelenk im ungünstigsten Fall nötig ist, damit die Haftreibung überwunden wird, und ist die Geschwindigkeit gleichzeitig nahe Null, wird ein Fixpunkt zugeordnet. Die Erkennung von periodischen Attraktoren ist auf diese Weise nicht möglich, hier aber auch nicht nötig, da das CSL zwar die Eigenschaft besitzt um den Fixpunkt zu pendeln, jedoch die obige Bedingung an den Umkehrpunkten erfüllt ist.

Da zumindest im Semni keine dedizierten Geschwindigkeitssensoren vorhanden sind, ist es nötig, diese durch Differenziation des Winkelwertes zu bestimmen. Ein guter numerischer Wert ließ sich erreichen, indem ein einfacher linksseitiger Differenzenquotient über den letzten und aktuellen Zeitschritt berechnet wird und dieser anschließend tiefpassgefiltert wird. Das dafür entworfene IIR-Filter zweiten Grades ist im Abschnitt B im Anhang dokumentiert.

Die Release-Modi in positiver und negativer Richtung wurden anders als im Simulator nicht mit statischen Biaswerten implementiert. Stattdessen wird ein PI-Controller verwendet, der eine konstante Geschwindigkeit in der gewünschten Richtung regelt. Das ist für den Semni sinnvoll, da die bereits im Simulator beobachteten Unterschiede zwischen Haftreibung und trockener Reibung hier noch deutlicher zum tragen kommen: Ein konstanter Biaswert muss so gewählt werden, dass die Haftreibung in jeder Situation überwunden wird um eine Symmetriebrechung zu liefern, gleichzeitig darf er aber nicht zu groß sein damit der Roboter dadurch nicht aus Fixpunkten herausbewegt wird. Da sich die Reibungsverhältnisse

mit zunehmendem Alter der Hardware auch ändern können, ist es kaum möglich, mit einem Biaswert allein zurecht zu kommen. Ein Geschwindigkeitsregler hat gleichzeitig den Vorteil, Körperteile nicht unkontrolliert fallen zu lassen und vermeidet damit ungewollt hohe Geschwindigkeiten.

Es ist damit aber nötig, zu erkennen, wann ein stabiler Fixpunkt erreicht ist um daraufhin den Geschwindigkeitsregler anzuhalten. Dafür ist ein Detektor nötig, der unterscheidet, ob sich ein Gelenk noch frei bewegt oder ob die Bewegung mit mehr Kraft als zuvor aufrecht erhalten wird, also der Boden, ein Hindernis oder der eigene Körper erreicht ist. Die Ableitung des Stromes erscheint dafür geeignet. Da der Strom ebenfalls von einem mit Rauschen überlagerten Sensor stammt, ist auch hier eine gleichzeitige Tiefpassfilterung sinnvoll. Dafür wurde statt einer nachgeschalteten Filterung ein Filter verwendet, das gleichzeitig tiefpassfiltert und ableitet (Selesnick, 2002) und dadurch eine geringere Latenz besitzt. Die genaue Berechnung der Filterkoeffizienten dieses FIR-Filters ist im Anhang beschrieben. Das erhaltene Signal wird nun abhängig von der zuvor gewählten Release-Richtung betrachtet und anhand eines Schwellwertes entschieden, ob der Strom schnell genug ansteigt und damit über einen Kontakt informiert. Prinzipiell könnte auch für die Berechnung der gefilterten Geschwindigkeit das Filter nach Selesnick benutzt werden, jedoch wären dafür hier andere Koeffizienten notwendig gewesen. Die Implementation eines IIR-Filters benötigt meist weniger Programmspeicher für die Koeffizienten, so dass dieser statt eines weiteren FIR-Filters mit einer eigenen Liste von Parametern gewählt worden ist.

Der im Semni verwendete STM32F103-Prozessor besitzt beschränkte Arbeitsspeicherkapazität, welche mit 20 kB nicht für Programmvariablen und den gesamten Explorationsgraph ausreichend sind. Daher wird der Graph direkt in den Flash-Speicher geschrieben, welcher mit 128 kB mehr Reserven bietet. Der Graph wird damit außerdem auch nach dem Ausschalten des Roboters gespeichert und wird beim nächsten Einschalten weiter verwendet und erweitert und dient als persistentes Gedächtnis des Roboters.

6.3. Vergleich von Simulation und Hardware

Wie bereits in Kapitel 4 beschrieben, ist bei der Erstellung einer Simulationsumgebung immer ein guter Kompromiss zwischen Genauigkeit, Rechenkomplexität und Entwicklungsaufwand notwendig. Nach der Wahl geeigneter Rechenmodelle, deren Qualität durch höhere Komplexität oft nur noch geringfügig steigerbar ist, müssen für diese möglichst gute Parameter gefunden werden, damit ein bestimmtes Systemverhalten gut abgebildet wird. Für die Umsetzung des Lernverfahrens ist das Verhalten der CSL-Struktur von besonderer Relevanz.

Abbildung 6.1 zeigt ein Aufrichten des Hüftgelenkes durch einen CSL-Regler im Contraction-

Modus, einmal mit einem physischen Semni und einmal mit dem simulierten Semni aufgenommen. Die Länge des Zeitfensters ist bei beiden Graphen 1500 Zeitschritte lang, also 15 Sekunden. Im Vergleich ist nach dem Erreichen des instabilen Fixpunktes (bei Zeitschritt 2100 oben und 1200 unten) zu sehen, dass auf dem simulierten Semni eine Oszillation um den Fixpunkt verbleibt. Auf der Hardware ließ sich dieses Verhalten durch entsprechende Parameterwahl vollständig vermeiden. Der simulierte Semni schaukelt auf seiner elliptischen Außenkante bei Erreichen des Fixpunktes stärker als Semni in der Realität[10]. Das ist damit zu erklären, dass in Box2D keine Rollreibung implementiert ist, welche das Abrollen in der Realität dämpft. Das Verhalten des Contraction-Modus in der Simulation ist anschaulich dem einer realen Implementation insbesondere wegen der Umsetzung der Haftreibung und des Motormodells und der Wahl geeigneter Parameter sehr ähnlich und die Unterschiede liegen hauptsächlich im zeitlichen Verlauf und der etwas geringeren Stabilität.

Beide Implementationen der CSL-Struktur besitzen kognitive Eigenschaften, jedoch werden nicht immer die gleichen Fixpunkte erreicht. Abbildung 6.2 und Abbildung 6.3 zeigen Fixpunkte innerhalb der Mannigfaltigkeit für eine kurze Exploration mittels der Heuristik-Variante $2 + 3$ mit 22 Knoten, die einmal im Simulator und einmal auf der Hardware durchgeführt wurde. Es ist hier gut zu sehen, dass durch die verschiedenen Umsetzungen des Release-Modus manche Bewegungen anders ausgeführt werden. Beispielsweise werden in den Knoten 5 bis 7 im Simulator die hellblaue Untermannigfaltigkeit erreicht, während der Zustand auf der Hardware in der grünen Untermannigfaltigkeit verbleibt. Auf der Hardware erkennt der Detektor nach dem Verlassen von Knoten 4 und dem nachfolgenden Erreichen von Knoten 5 bereits das Berühren des Bodens und damit, dass der stabile Fixpunkt erreicht ist. In der Simulation, die nur einen konstanten Bias verwendet, bewegt die Hüfte den Körper noch über den etwas herausstehenden Fuß des unteren Armsegmentes und damit in die hellblaue Untermannigfaltigkeit. Nachfolgende Knoten werden damit in andere Bereiche und Fixpunkte gelangen.

Obwohl letztendlich nahezu der gleiche Bereich erkundet und in beiden Fällen nach kurzer Zeit auf die hellblaue Untermannigfaltigkeit gewechselt wurde, sind neben den Fixpunkten selbst auch die genauen Positionen einzelner Fixpunkte mitunter etwas verschieden und der entstandene Graph zeigt dadurch andere Verbindungen. Ein einzelner Knoten liegt immer innerhalb der zum Fixpunkt gehörigen Ruhemenge, jedoch hängt es wie bereits beschrieben von den Detektoren ab, wo genau ein Knoten gespeichert wird. Es kann daher für die Zukunft lohnenswert sein, statt nur einem Fixpunkt die gesamte zur Posture gehörige Ruhemenge zu speichern. Für den Contraction-Modus lässt sich die Ruhemenge grob bestimmen, indem anhand der Oszillation um einen bereits bekannten Knoten die Grenzen gespeichert werden. Ein stabiler Fixpunkt des Release-Modus kann links- oder rechtsseitig erreicht werden (siehe Abbildung 3.1), so dass bei erneutem Erreichen eines Knotens der Bereich zwischen dem

[10] Idealerweise wäre hier auch jeweils der Körperwinkel mit eingezeichnet, um dieses Schaukeln zu zeigen. Es ist aber auch anhand des Verlaufes des CSL-Ausganges zu erkennen.

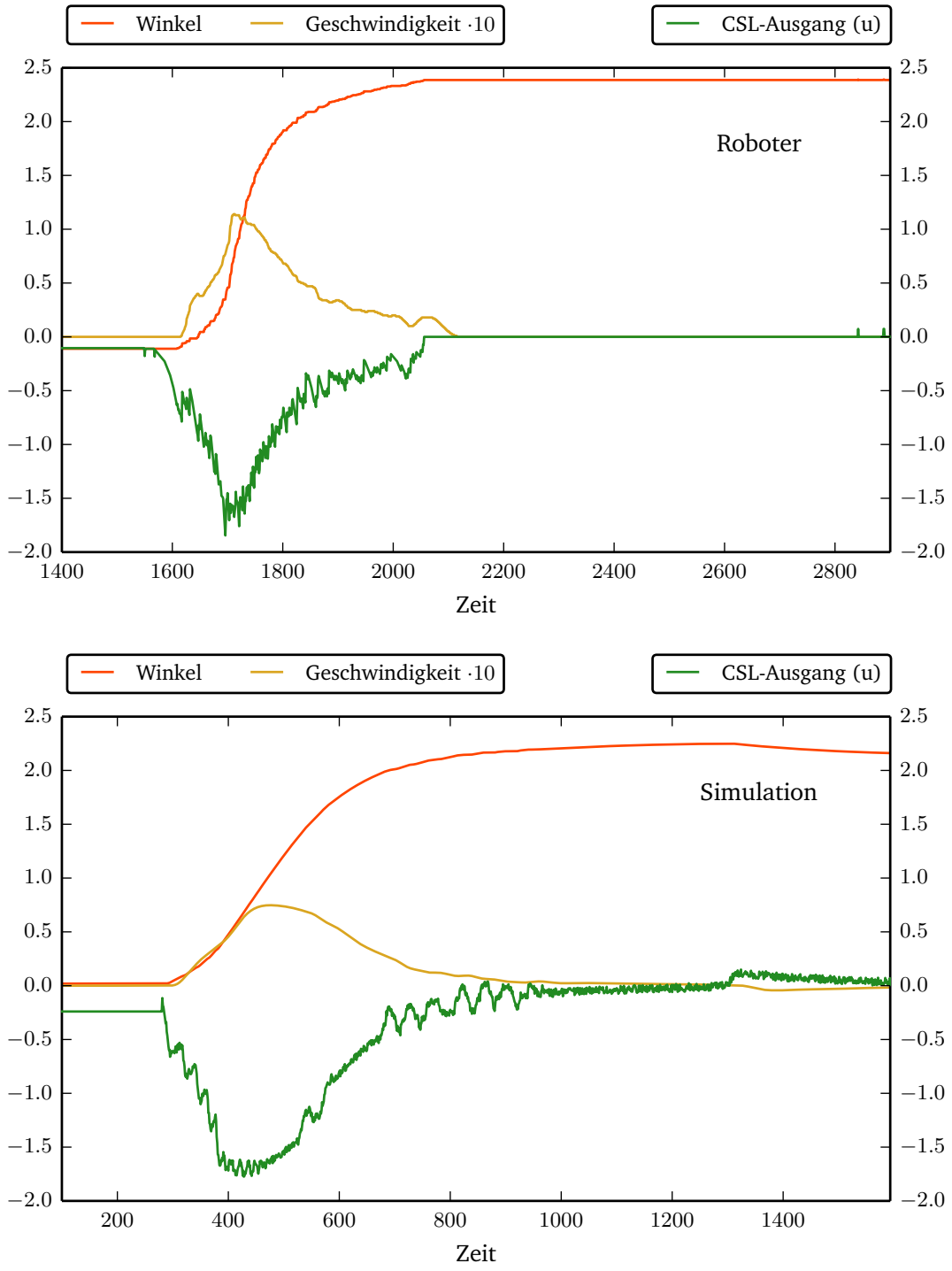

Abbildung 6.1.: Verhalten der CSL-Implementation auf dem Roboter Semni (oben) und im Simulator Simni (unten) mit den von Hand bestimmten Parametern. Die Einheiten sind im Bogenmaß für Winkel, $10 \cdot \mathrm{rad/t}$ für Geschwindigkeiten und Volt für den CSL-Ausgang. Das Verhalten unterscheidet sich aufgrund unterschiedlicher CSL-Parameter und begrenzter Simulationsgenauigkeit geringfügig.

bereits gespeicherten und dem neu erkannten Fixpunkt als Ruhemenge betrachtet werden kann.

Für den Simulator ist damit gezeigt, dass dieser die CSL-Regelschleife und das physikalische Verhalten des Roboters nicht exakt aber brauchbar genau abbilden kann. Das ABC-Verfahren ist auf beiden Plattformen erprobt worden und es sind Graphstrukturen entstanden, die jeweils das zugrunde liegende dynamische System strukturieren und beschreiben. Leichte Unterschiede dabei sind den Eigenheiten der verschiedenen Umgebungen geschuldet. Die Topologie der Strukturen ist aber nahezu identisch und zeigt, dass das Verfahren abstrahierende Informationen speichert, die robust gegenüber leichten Veränderungen des Systems oder der Umwelt sind.

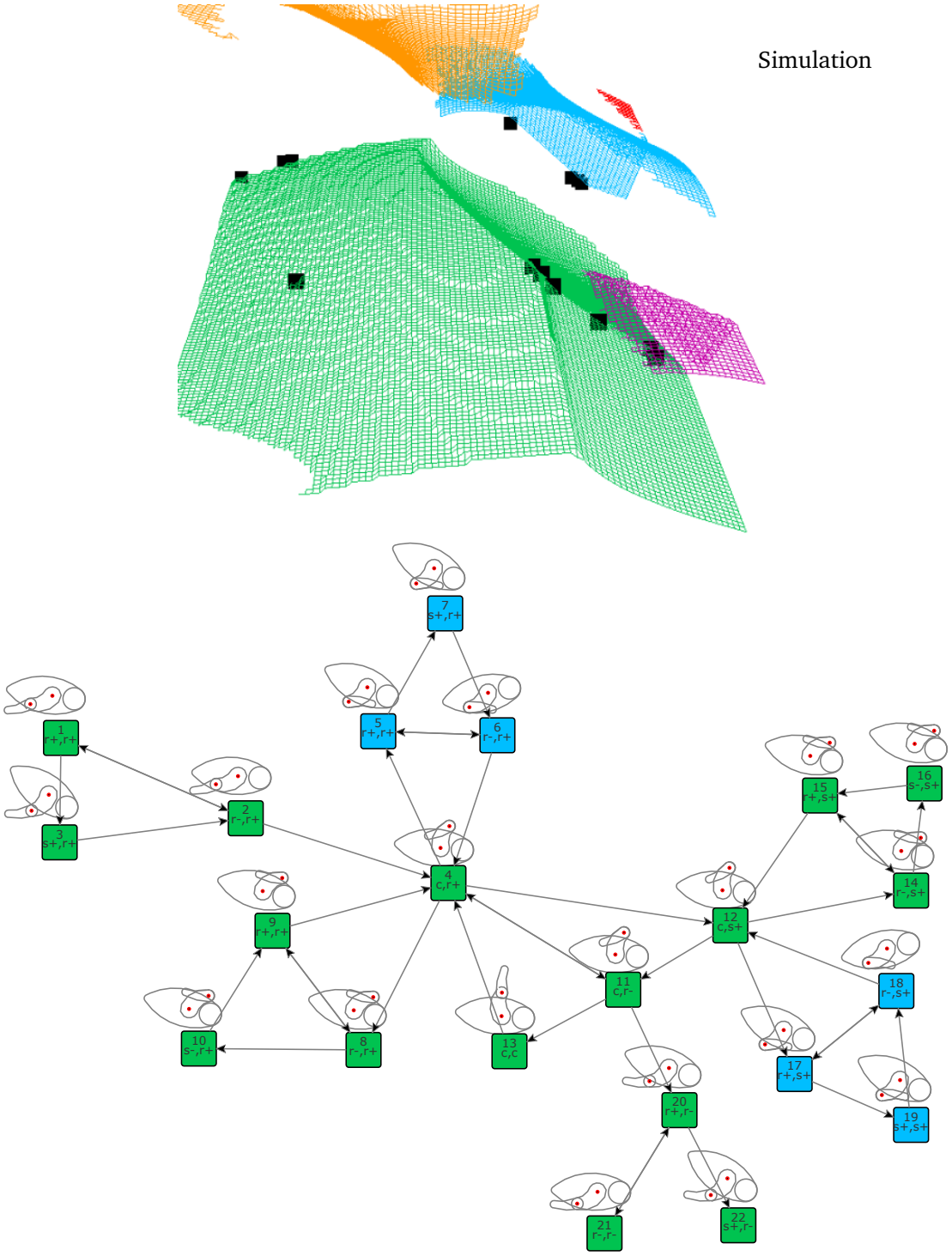

Abbildung 6.2.: Explorationsgraph mit Variante $2+3$ in der Simulation und Fixpunkte innerhalb der analytisch bestimmten Mannigfaltigkeit.

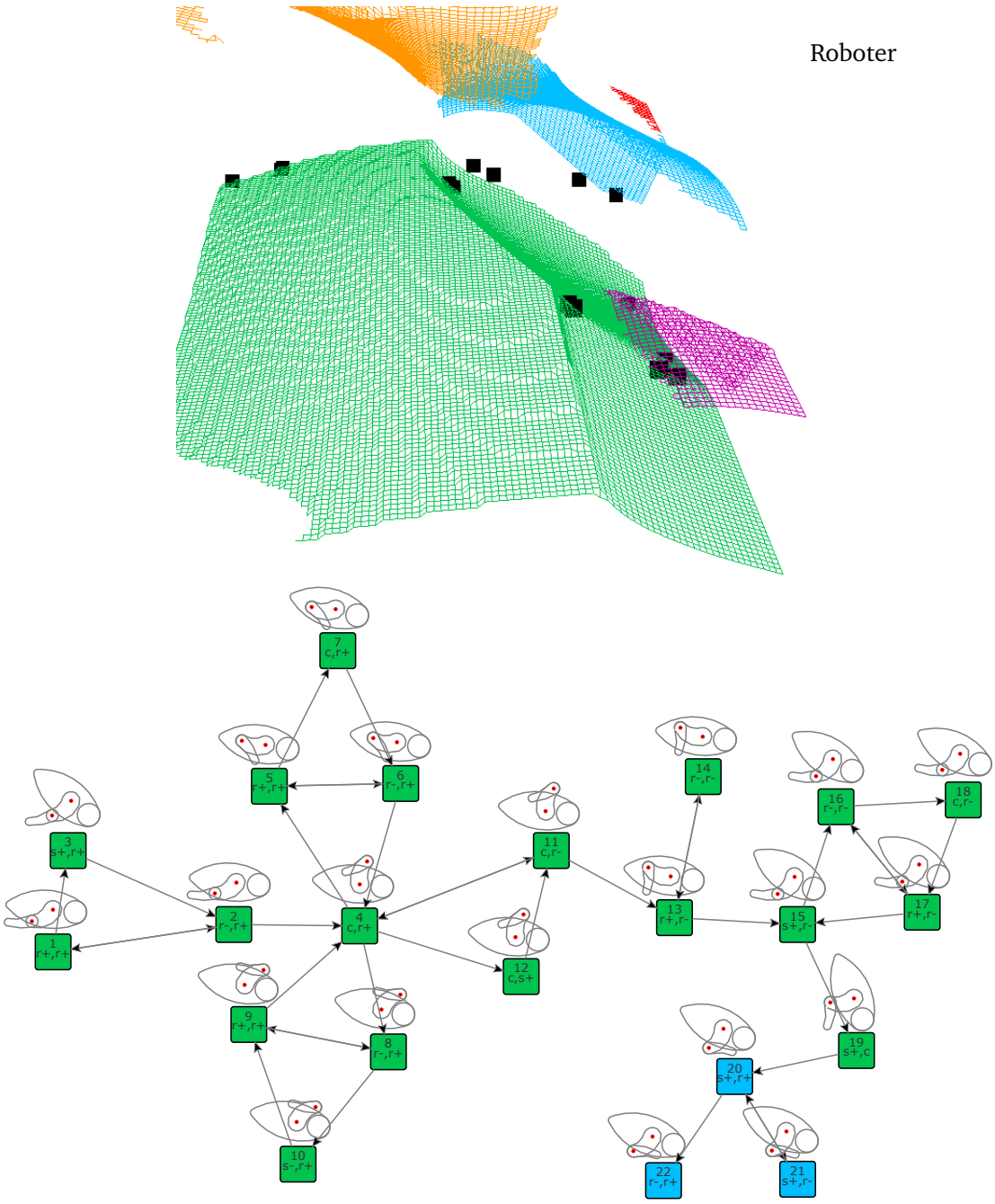

Abbildung 6.3.: Explorationsgraph mit Variante 2 + 3 auf der Hardware und Fixpunkte innerhalb der analytisch bestimmten Mannigfaltigkeit.

7. Experimente

Das folgende Kapitel stellt die Resultate verschiedener Explorationsläufe vor und gegenüber. Dafür wurden die verschiedenen in Tabelle 5.2 bereits beschriebenen Varianten der Heuristik jeweils für 30 Minuten in Echtzeit simuliert ohne dabei Einfluss zu nehmen. Auf wiederholte Experimente wurde verzichtet, da diese in der Simulation nahezu identische Ergebnisse geliefert hätten.

Tabelle 7.1 zeigt für die Varianten den zeitlichen Verlauf der besuchten Untermannigfaltigkeiten. Früher oder später erreicht jede Variante alle Fixpunkte, jedoch sind nach 30 Minuten noch deutliche Unterschiede in Hinblick auf die Häufigkeit der Wechsel zu erkennen.

Variante	Verlauf
1 + 2	
3 + 4	
1 + 4	
2 + 3	

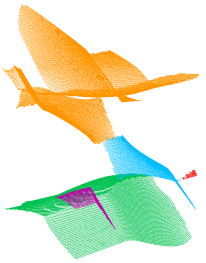

Tabelle 7.1.: Vergleich der besuchten Untermannifaltigkeiten über der Zeit während einer 30-minütigen Exploration. Nachdem ein neuer Fixpunkt erkannt wurde, wird der aktuelle Balken an der rechten Seite um einen weiteren Streifen erweitert, der die Untermannigfaltigkeit farblich kodiert. Die Farben entsprechen den Untermannigfaltigkeiten in der nebenstehenden Abbildung. Bereits existierende, erneut besuchte Knoten werden ebenfalls mit einem Streifen markiert.

Für die einzelnen Varianten lassen sich Analogien zur Suche in Graphen aufstellen, welche in Abbildung 7.1 abstrakt dargestellt sind. Variante 1+2 ändert die Richtung und das Gelenk nicht, bis dies nicht mehr möglich ist. Das entspricht einer kontinuierlichen Bewegung über die Untermannigfaltigkeit, so lange nicht bereits eine Kante in dieser Richtung zuvor exploriert wurde oder eine unüberwindbare Grenze der Untermannigfaltigkeit erreicht ist, also ein Winkelanschlag oder eine Begrenzung durch den eigenen Körper. Da beide Haltemodi $s+$ und $s-$ die Eigenschaft besitzen, die Bewegung in der gleichen Richtung nicht weiter zu erlauben, liegt die nächste mögliche Kante in einer anderen Richtung. Andererseits kann in diesem Fall auch das Gelenk gewechselt werden, für welches in der Folge die Richtung so lange wie möglich beibehalten wird. Welches Verhalten bevorzugt wird, ist ebenfalls Teil der jeweiligen Heuristik. In den hier gezeigten Experimenten wurde auf dieser

zweiten Entscheidungsebene versucht, das Gelenk zu wechseln bzw. wenn das bereits Teil der Heuristik selbst ist, dieses stattdessen beizubehalten. Eine qualitative Untersuchung, welche genauen Auswirkungen diese Alternativentscheidungen haben, steht noch aus.

Die durch die Trajektorien zwischen den angefahrenen Fixpunkten entstehenden Wege über die Mannigfaltigkeit sind damit bei der Variante $1 + 2$ linienartig und ähneln einer Tiefensuche, bei der ebenfalls zuerst immer ein bestimmter Zweig in einem Graphen gewählt wird. Die Erwartung für die Exploration ist daher, dass damit zuerst viele weiter voneinander entfernte Fixpunkte gefunden und die Lücken später gefüllt werden. Im resultierenden Explorationsgraphen in Abbildung 7.2 ist daher auch zu sehen, dass besonders bei den ersten Knoten fast keine Kanten existieren, die eine Verbindung in beide Richtungen herstellen[11]. Die Knoten haben oft eine eingehende Kante von einem ersten Knoten und eine ausgehende Kante zu einem dritten Knoten, das heißt sie werden im Vorübergehen besucht. Erstaunlicherweise ist aber für die Mannigfaltigkeit des Semni anhand der Darstellung in Tabelle 7.1 zu sehen, dass innerhalb der ersten 30 Minuten nur zwei Untermannigfaltigkeiten besucht werden. Die wenigen Knoten, an denen ein Wechsel in eine andere Mannigfaltigkeit möglich ist, benachteiligen diese Strategie insofern, dass es eher zufällig dazu kommt, genau solch einen Knoten zu erreichen. Das Hin und -herlaufen zwischen den Grenzen einer Untermannigfaltigkeit führt dazu, dass der Zustand wie ein Tischtennisball in einem gläsernen Weinballon so lange umherfliegt, bis er zufällig auf den Ausgang durch den Flaschenhals trifft. Gleichzeitig wird eine Untermannigfaltigkeit auch schnell wieder verlassen wenn sich mehr Möglichkeiten dazu bieten (hellblau).

Variante $3 + 4$ ergibt eine Zick-zack-Bewegung über die Mannigfaltigkeit, da nach jedem neuen Fixpunkt die Richtung und das Gelenk geändert wird und hier zwei Gelenke und zwei Richtungen vorliegen. Bei einer anderen Gelenkzahl ist von anderen Mustern auszugehen. Abbildung 7.3 zeigt, dass die lokale Vernetzung stärker als bei Variante $1 + 2$ ist. Durch die abwechselnde Bewegung der Gelenke werden die Untermannigfaltigkeiten weniger schnell wieder verlassen, jedoch ist auch hier ein schnelles Erreichen aller Untermannigfaltigkeiten nicht gegeben, da die Knoten zum Übergang zu einer anderen Untermannigfaltigkeit nur erreicht werden, wenn die Bewegung die Knoten zum Übergang eher zufällig findet. Würde statt der gleichzeitigen Änderung bei einem Fixpunkt das Gelenk geändert und beim nächsten wieder die Richtung, sollte eine Art Breitensuche entstehen, bei der zuerst Schleifen um die Fixpunkte entstehen, so lange nicht alle Kanten besucht sind. Damit könnte sich der Zustand mäanderförmig über die Mannigfaltigkeit bewegen.

Variante $1 + 4$ und $2 + 3$ stellen Mischformen dar, die sich entweder konservativ in der Richtungswahl verhalten und alternierend in der Wahl des Gelenkes (Variante $1 + 4$) oder konservativ in der Wahl des Gelenkes und alternierend in der Bewegungsrichtung (Variante

[11] Die Darstellung der Graphen ist so gewählt, dass statt zwei Kanten in entgegengesetzter Richtung zu zeigen, nur eine mit zwei Pfeilen dargestellt wird.

$2+3$). Die dadurch entstehenden Graphen in den Abbildungen 7.4 und 7.5 zeigen, dass es für das schnelle Erreichen weiter entfernter Fixpunkte vorteilhaft ist, eine dieser beiden Heuristiken zu wählen. Variante 1+4 ist der Variante 3+4 insofern ähnlich, dass durch das ständige Wechseln der Richtung eine Zick-zack-Bewegung entsteht, die sich in gleicher Richtung fortbewegt und dabei beständig das Gelenk wechelt. Dabei werden vergleichsweise häufig Übergangsknoten erreicht, an denen die Untermannigfaltigkeit gewechselt wird.

Variante $2+3$ erreicht am schnellsten auch die weiter entfernten Knoten in der orangen Untermannigfaltigkeit, da nach jeder Bewegung eine Rückkehr zum vorherigen Knoten geschieht. Allerdings ist nicht für jede heterokline Bahn der Rückweg direkt möglich, so dass dabei auch gänzlich andere Fixpunkte erreicht werden. Innerhalb der orangen Mannigfaltigkeit wird hier am längsten exploriert. Sind in einem Knoten für ein Gelenk beide Richtungen exploriert, wird das Gelenk gewechselt. Diese Art der "Breitensuche" bewegt sich eng in der Umgebung der vorherigen Knoten und verlässt diese erst, wenn alle Kanten erkundet sind. Die besuchten Untermannigfaltigkeiten in Tabelle 7.1 zeigen, dass dieses konservative Verhalten, wie es auch manche Nagetiere bei der Futtersuche aus einer Erdhöhle heraus praktizieren (z. B. Degus), erfolgreich sein kann und so Übergangsknoten schnell gefunden und neue Untermannigfaltigkeiten seltener verlassen werden.

Während es ein mögliches Ziel einer Exploration ist, in möglichst kurzer Zeit bereits einen großen Teil der Mannigfaltigkeit stichprobenartig zu erkunden, kann es in späteren Entwicklungen des ABC-Verfahrens auch besser sein, z. B. für die Fähigkeit, den gesamten Körper zu balancieren, möglichst häufig die Untermannigfaltigkeit zu wechseln, also ein Umfallen zu erreichen, wie es hier am ehesten Variante $1+4$ aufweist. Für weniger robuste Roboter kann es sinnvoll sein, das Umfallen möglichst gering zu halten, was durch die Varianten $1+2$ und $3+4$ wahrscheinlich wird. Letztendlich finden aber alle Varianten alle Postures nach einer gewissen Zeit.

Variante	Anzahl der Knoten in 30 Minuten	Mittlere Besuchshäufigkeit
1+2	55 Knoten	2,43 ($\sigma = 1{,}19$)
3+4	67 Knoten	2,44 ($\sigma = 1{,}32$)
1+4	68 Knoten	2,19 ($\sigma = 1{,}36$)
2+3	82 Knoten	2,37 ($\sigma = 1{,}13$)
Zufällig	43,7 Knoten ($\sigma = 6{,}18$)	3,06 ($\sigma = 0{,}17$)

Tabelle 7.2.: Statistische Daten der Beispielgraphen. Zum Vergleich sind die Daten für die zufällige Exploration ebenfalls gegeben, welche aus drei Läufen ermittelt wurden. Die Standardabweichung σ ist daher ebenfalls angegeben.

Bis zu einer vollständigen Exploration muss bei jeder Heuristik jeder Knoten mindestens einmal erkannt werden. Die Wege zwischen diesen können jedoch während der Exploration verschieden häufig traversiert werden und Knoten können verschieden häufig erneut besucht werden. Innerhalb der ersten 30 Minuten könnten beispielsweise vorwiegend Knoten mit dem Release-Modus gefunden werden, welche im Simni schneller erkannt werden als ein Contraction-Modus, da es zu keiner Pendelbewegung um den Fixpunkt kommt. Die Anzahl der innerhalb einer festen Zeitspanne gefundenen Knoten wäre damit höher. Insofern sind die Zahlen in Tabelle 7.2 nur ungenau. Es zeigt sich aber, dass es im Mittel vorerst nur kleine Unterschiede in der Häufigkeit gibt, mit denen Knoten erneut besucht werden. Im Vergleich zur zufälligen Exploration wird deutlich weniger wiederholt besucht und es werden insgesamt deutlich mehr Postures gespeichert. In den Abbildungen der Graphen ist für alle Varianten sichtbar, dass es einen oder einige wenige Knoten gibt, die sehr viel häufiger besucht und über die viele andere Knoten erreicht werden.

Obwohl sich hier klar Unterschiede zwischen den Varianten zeigen, ist nicht abschließend geklärt, wie sich die gleichen Varianten mit anderen CSL-Parametern, Detektoren und Morphologien verhalten. Insbesondere für die Semni-Mannigfaltigkeit zeigt sich, dass es für das Besuchen vieler Untermannigfaltigkeiten nötig ist, einige wenige Knoten zu erreichen, die dann wiederum neue bisher nicht erreichte Untermannigfaltigkeiten preisgeben. Das schnelle Erreichen dieser Knoten kann hier nur exemplarisch auf die Varianten zurückgeführt werden.

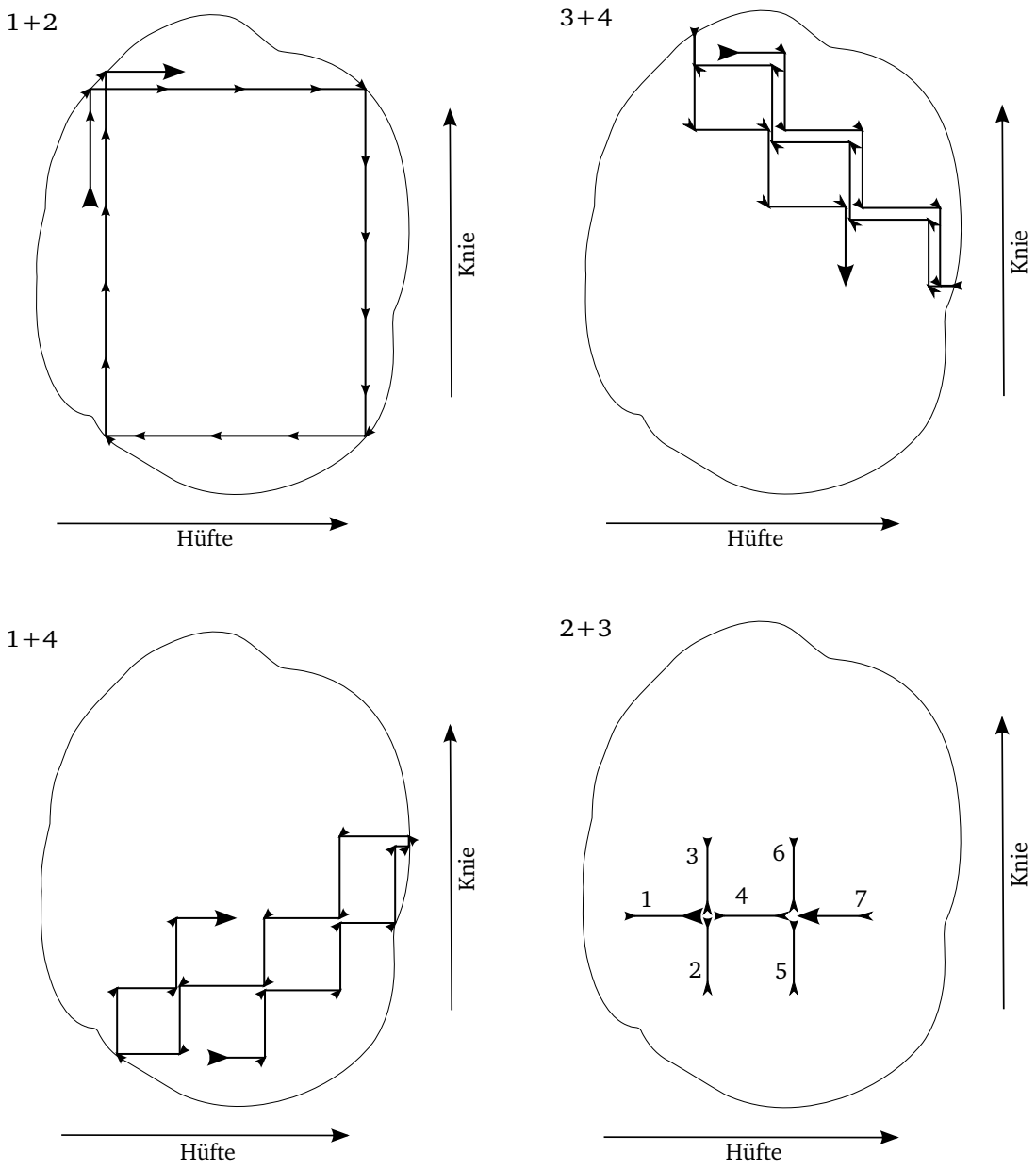

Abbildung 7.1.: Skizzenhafte Bewegungen innerhalb der Mannigfaltigkeiten für die verschiedenen Heuristiken innerhalb einer fiktiven Gitterwelt, bei der die Fixpunkte in regelmäßigen Abständen verteilt und jeweils direkt von ihren Nachbarn erreichbar sind. Größere Pfeile geben Anfangs und Endpunkte des Ausschnittes der Trajektorien an, kleine Pfeile Fixpunkte an denen eine Entscheidung getroffen wird. Zahlen bei Bild 2+3 geben die Reihenfolge der Trajektorien an. In einigen Fällen sind überlagernde Trajektorien dicht nebeneinander Dargestellt, um die Übersichtlichkeit zu verbessern.

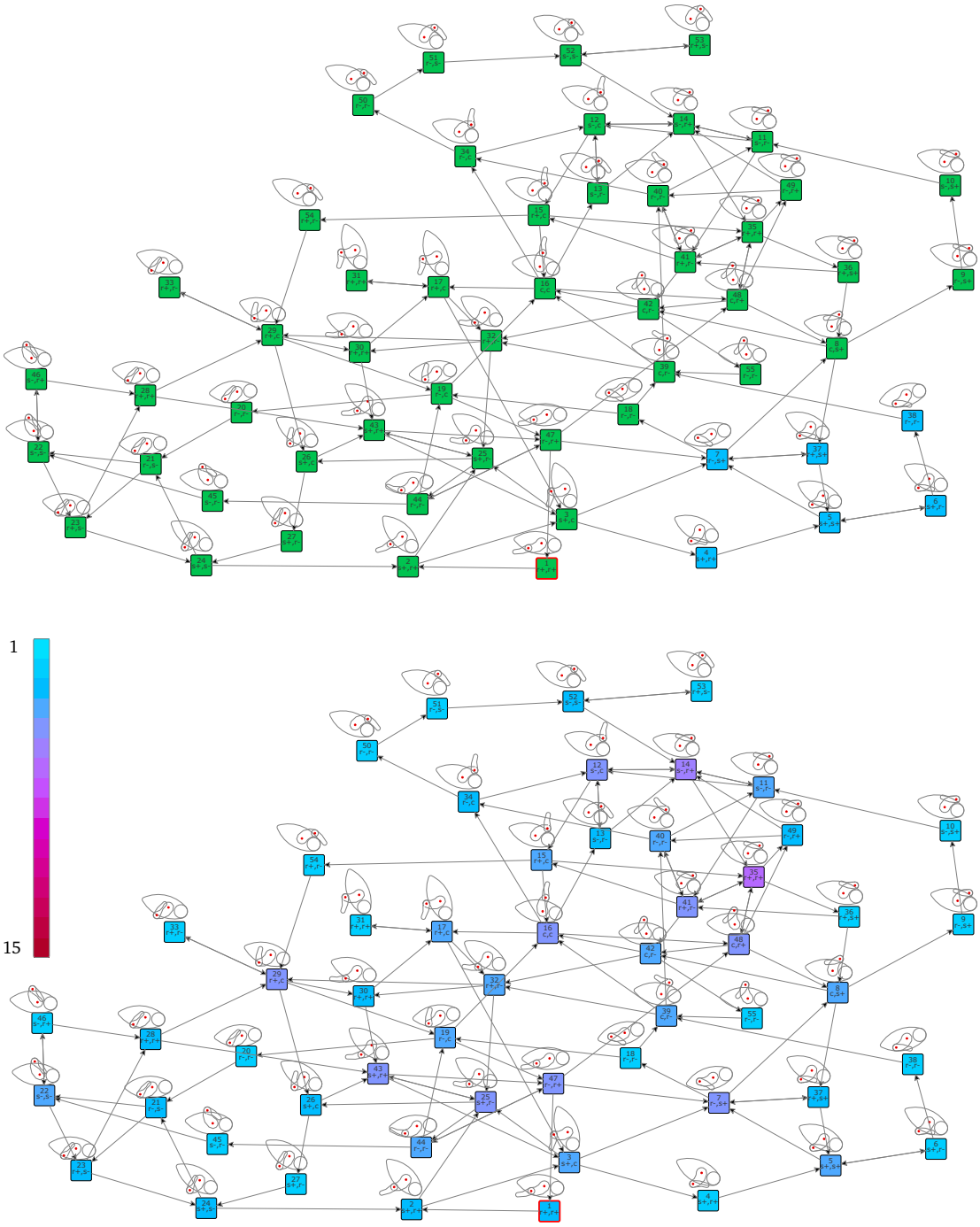

Abbildung 7.2.: 30-Minütige Exploration der Heuristik mit unbesuchten Kanten, Diffusionsprozess und Variante 1+2. Oben: Einfärbung nach besuchter Untermannigfaltigkeit. Unten: Einfärbung nach Anzahl wiederholter Besuche pro Knoten.

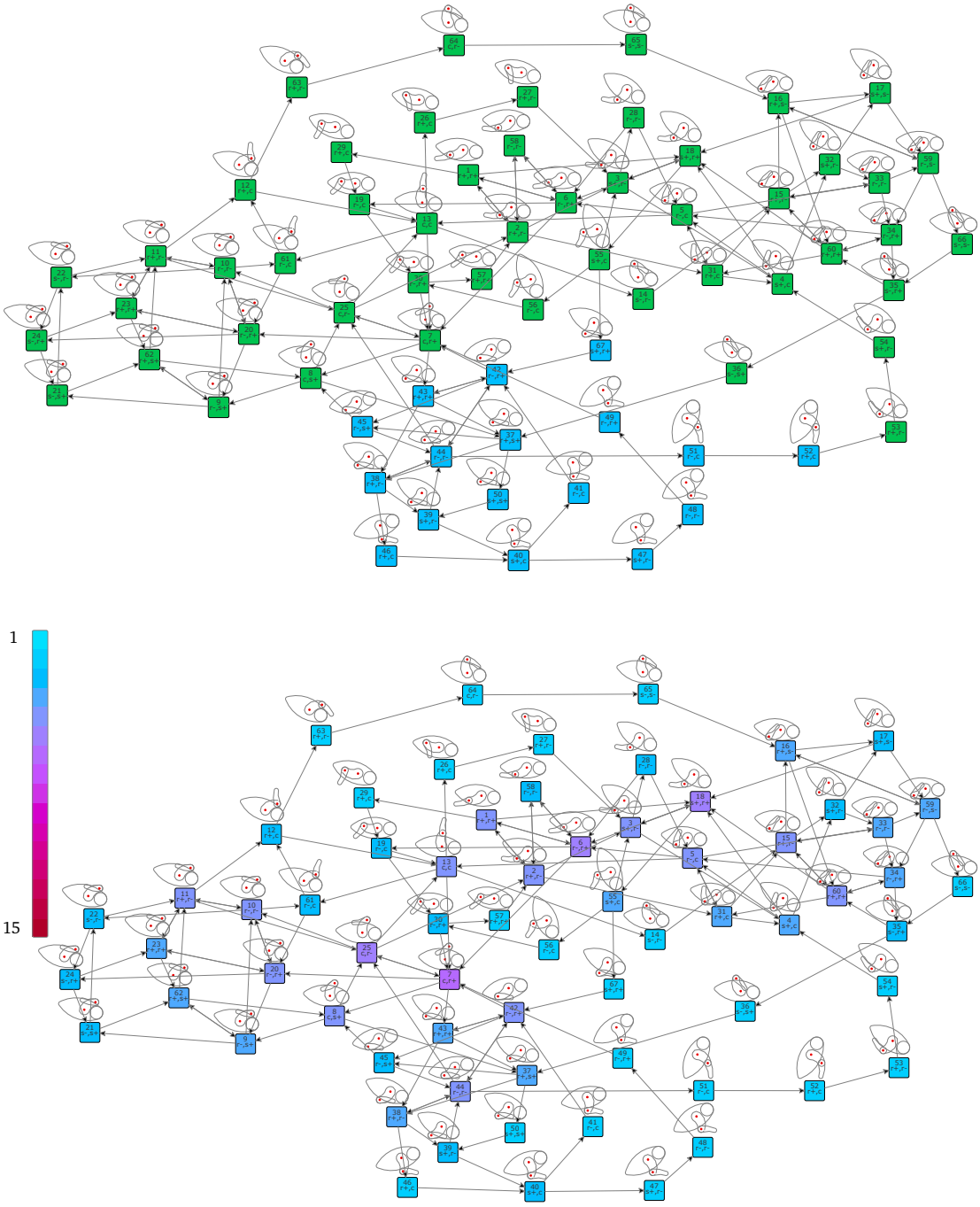

Abbildung 7.3.: 30-Minütige Exploration der Heuristik mit unbesuchten Kanten, Diffusionsprozess und Variante 3 + 4. Oben: Einfärbung nach besuchter Untermannigfaltigkeit. Unten: Einfärbung nach Anzahl wiederholter Besuche pro Knoten.

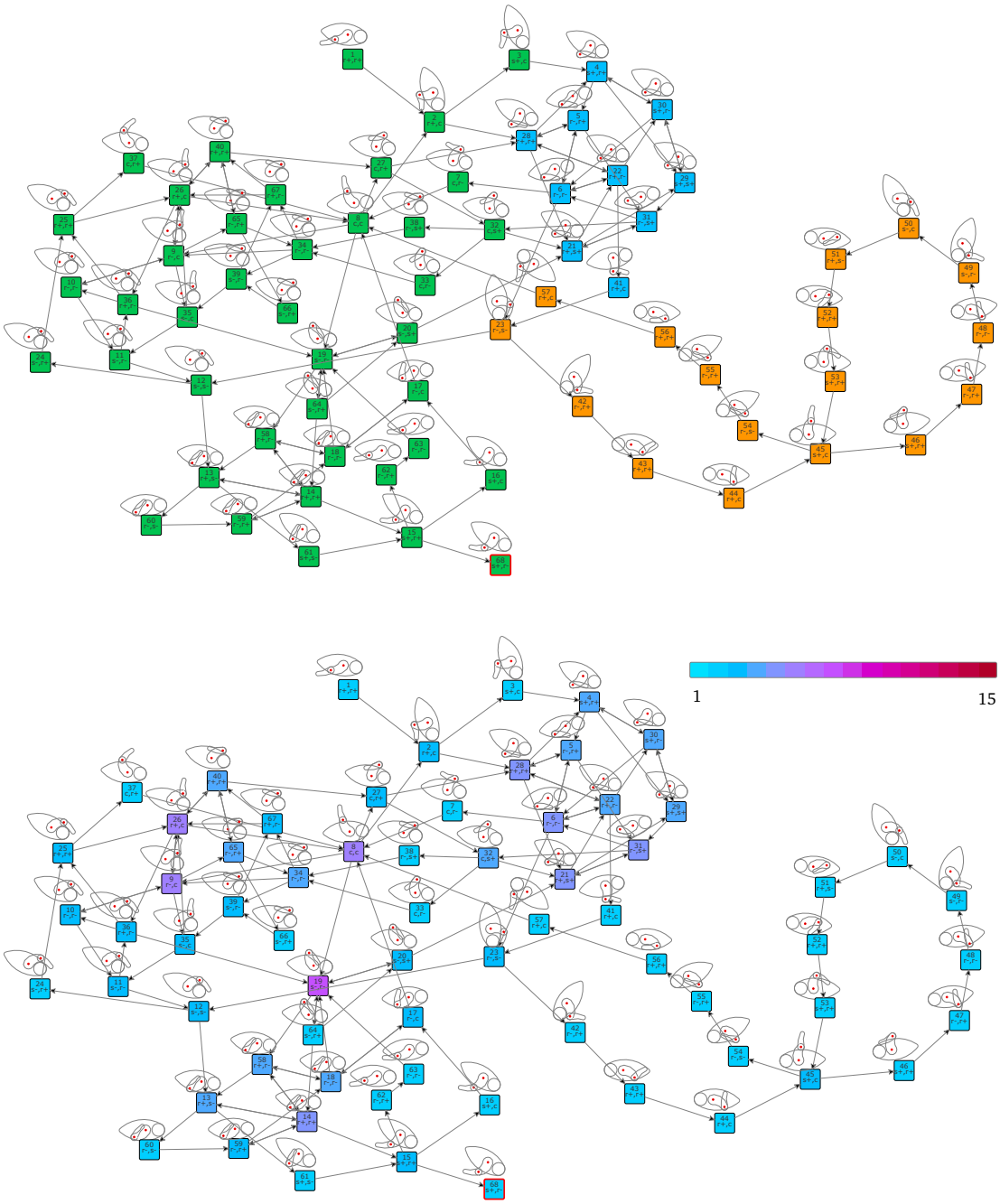

Abbildung 7.4.: 30-Minütige Exploration der Heuristik mit unbesuchten Kanten, Diffusionsprozess und Variante 1+4. Oben: Einfärbung nach besuchter Untermannigfaltigkeit. Unten: Einfärbung nach Anzahl wiederholter Besuche pro Knoten.

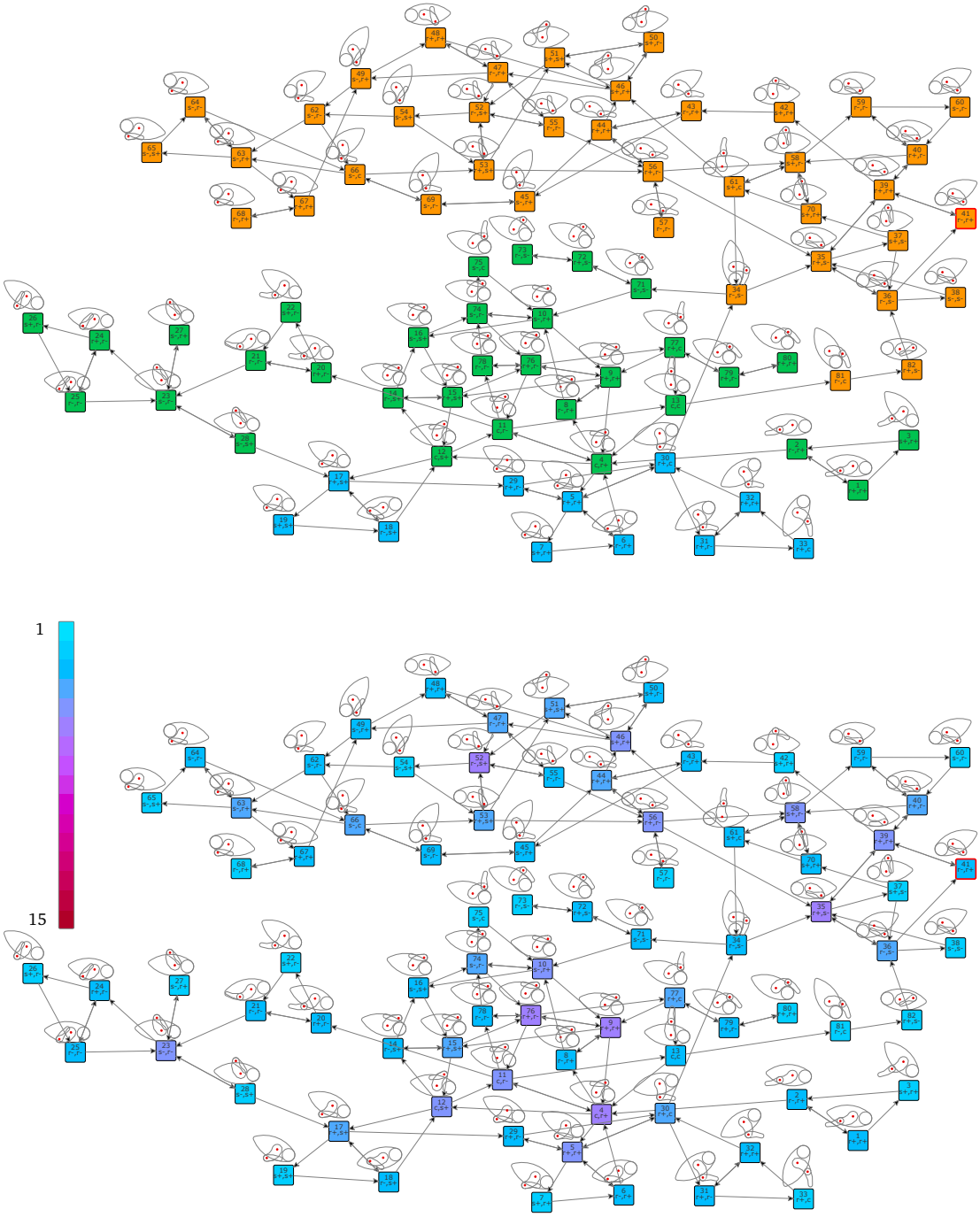

Abbildung 7.5.: 30-Minütige Exploration der Heuristik mit unbesuchten Kanten, Diffusionsprozess und Variante 2+3. Oben: Einfärbung nach besuchter Untermannigfaltigkeit. Unten: Einfärbung nach Anzahl wiederholter Besuche pro Knoten.

8. Zusammenfassung und Ausblick

Die Ziele dieser Arbeit waren es, das ABC-Learning um verschiedene Heuristiken zu erweitern, diese auf verschiedenen Plattformen zu implementieren und ihre Auswirkungen auf das Verhalten des Lernverfahrens zu vergleichen. Mit einem physikalischen Simulator und einer Implementation für ein Embedded-System sind Referenzimplementationen entstanden, die Zusammen mit dieser Arbeit als Basis für zukünftige weitere Entwicklungen des Verfahrens dienen können.

Das ABC-Lernverfahren bildet eine einfache, von der Umwelt und der Situation abhängige Repräsentation der eigenen Bewegungsfähigkeit, das als eine erste Form von sensomotorischer Selbstwahrnehmung betrachtet werden kann. Die entstehende kompakte Repräsentation des zugrundeliegenden dynamischen Systems kann in der Folge für weitere Exploration verwendet werden, so dass die bisher konstanten Reglerparameter variiert, neue dynamische Bewegungen versucht und Bewegungsaktionen mit gänzlich anderen Ansteuerungsmethoden verwendet werden können. Im Vergleich zu anderen Lernprinzipien, z. B. die anfangs genannte Homöokinese, die das Wissen und die Regler in einer Struktur vorhalten, die zentral prozessiert wird, ist es durch lokale Regelschleifen möglich, auch für modulare Roboter Wissen zu erlangen, das nach Änderungen am System noch gültig ist. Iterative Morphologieveränderungen an einem Roboter bedeuten damit nicht, dass zuvor durchgeführte Lernprozesse verworfen werden müssen. Das System kann wie ein natürliches Lebewesen wachsen und auch seine vorherigen Erfahrungen mitwachsen lassen. Damit ist es auch möglich, selbstzerstörerisches Verhalten anfänglich unmöglich zu machen, bis stabile Fähigkeiten erlangt sind. Es ist vergleichsweise einfach, das Verfahren durch andere Regelstrukturen, Heuristiken und Prinzipien zu erweitern, welche auf das zuvor gewonnen Wissen zurückgreifen, ohne die parallel weiter bestehenden bisherigen Strukturen einzuschränken.

In der Diplomarbeit von Benjamin Werner wurde in isolierten Experimenten das Kick-and-Fly-Paradigma untersucht, das zum Ziel hat, die mit dem CSL im Contraction-Modus zuvor ermittelten Trajektorien schneller abzufahren, als es bisher geschieht (Werner, 2013). Wird dieses in das ABC-Learning integriert, ist es möglich, nach dem Bestimmen des Fixpunktes beim Anlegen eines Knotens bei dem nächsten Besuch des Knotens nicht mehr den CSL-Regler zu verwenden, um in den Fixpunkt zu gelangen, sondern eine einfache Motorsteuerung, die die maximale Motorkraft für einen kurzen Zeitabschnitt ansteuert, so dass der Motor und

das verbundene Körperteil in Bewegung gerät. Dann wird wie zuvor auf den CSL-Regler gewechselt, welcher den kleinen Restweg zum Fixpunkt regelt. Die Länge der Kick-Phase wird für eine spezifische Hardware jeweils gelernt, indem zuerst die gesamte Aufrichtbewegung durch das CSL geregelt wird und die erzeugte Spannung gemessen bzw. der Ausgangswert des CSL selbst verwendet wird. Die Summe dieser Werte über die Zeit ist die Energie, die durch die "blinde" Steuerung ebenfalls aufgebracht werden muss, um eine aufgerichtete Position zu erreichen. Ist der größte Teil der zuvor ermittelten Summe durch die Steuerung mit maximalem Drehmoment aufgebracht, wird wieder auf das CSL geschaltet, welches weiterhin Adaptivität und energiesparende Stabilisierung bietet.

Mit den Strukturen aus dem ABC-Learning bietet sich die Möglichkeit, dynamische Controller zu verwenden, die an Kanten der Untermannigfaltigkeiten die Dynamik des Systems ausnutzen. Diese sollen ähnlich wie das CSL, das instabile Fixpunkte für ein Gelenk findet und stabilisiert, den gesamten Körper in seiner Rollbewegung in einem instabilen Fixpunkt halten. Da die Drehung des Körpers nicht direkt beeinflusst werden kann, muss diese über schnelle Bewegungen des Armes geschehen, die verschiedene Massenträgheiten ausnutzen (dieses isolierte Problem ist als "Akrobot" bekannt). Nachdem der Explorationsgraph für langsame Bewegungen erzeugt worden ist, kann die Exploration von diesen quasistatischen Bewegungen zu verschiedenen solcher dynamischer Bewegungen übergehen, und den Graph mit neuem Wissen erweitern.

Während des wiederholten Besuchens von Knoten kann mit beliebigen Verfahren der Parameterraum der CSL-Regler untersucht werden, um je nach Lage des Roboters ideales Verhalten zu erreichen. Das Verhalten des CSLs bei bestimmten Parametern ist unter Anderem abhängig von den Massen und der Länge des Hebelarms und der Reibung im Gelenk. Es ist außerdem möglich, erste Verhaltensregeln ebenfalls in diesem Graph zu speichern, so dass z. B. der aktuelle Akkuladestand verwendet wird, um zu entscheiden, welche Knoten als nächstes anvisiert werden. Es kann sinnvoll sein, bei niedrigem Akkustand möglichst noch letzte selbstschützende Maßnahmen auszuführen, wie z. B. die Bewegung aus einem Knoten, der sich in einem instabilen Fixpunkt befindet, in einen stabilen Fixpunkt, für den der Roboter auch ohne Strom in seiner Position verbleibt.

Eine laufende Masterarbeit im Labor für Neurorobotik von Markus Janz beschäftigt sich mit der Erkennung von verschiedenen Umweltsituationen allein aus den propriozeptiven Explorationsdaten. Für den Fall, dass zuvor mehrmals vollzogene Bewegungen plötzlich nicht mehr ausführbar sind (das heißt es wird statt einem bereits gespeicherten Knoten ein anderer, bisher nicht bekannter Knoten erreicht), ist davon auszugehen, dass sich die Situation in einer vorerst unbekannten Form verändert hat. Situationen reichen von Veränderungen am eigenen Körper, wie Beschädigungen, heißgelaufenen Motoren oder temporären internen Störungen zu Einschränkungen der Bewegungsfreiheit, z. B. wenn ein Hindernis in der Umwelt Grenzen bietet (Semni in einer Schachtel mit Deckel oder ohne). Dafür wird der Graph um eine parallele Gedächtnisebene von Kontextneuronen erweitert, siehe Abbildung 8.1, welche

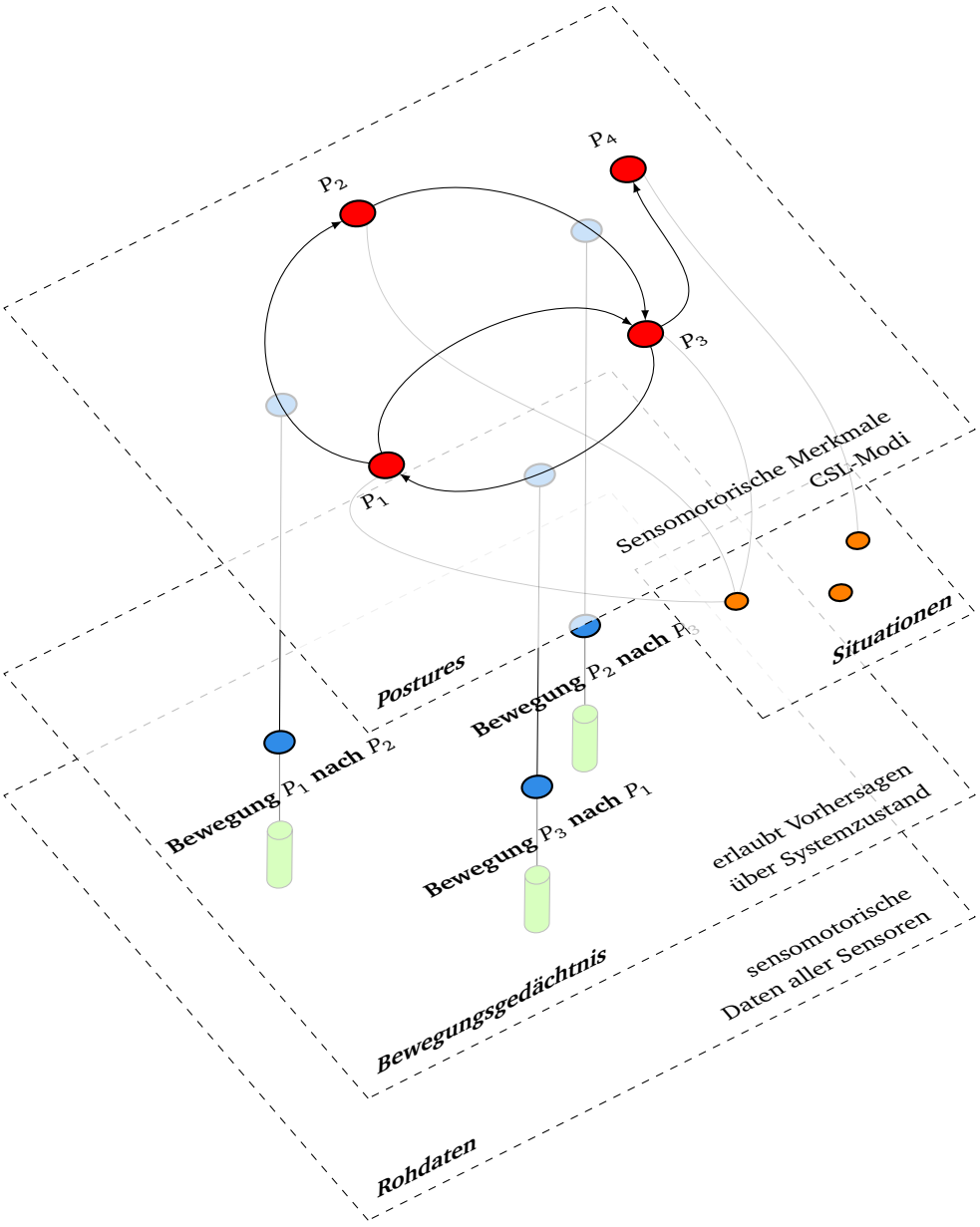

Abbildung 8.1.: Erweiterte Ebenen des ABC-Lernverfahrens. Die oberste Ebene enthält gefundene **Postures** als Knoten und mögliche Übergänge als Kanten. Die beim Abfahren der heteroklinen Orbits mitgeschriebenen **Sensordaten** sind in der untersten Ebene dargestellt. In der Ebene darüber befinden sich abstraktere Informationen, die mit verschiedenen Verfahren aus den rohen Sensordaten gewonnen werden und damit semantisch verdichtet sind (**Bewegungsgedächtnis**). Die vorherigen Rohdaten können entfernt werden, sobald ein zugehöriger Eintrag im Bewegungsgedächtnis erstellt worden ist. Die vertikalen Linien in die oberste Ebenen geben die Zugehörigkeit der Bewegungsdaten zu den entsprechenden heteroklinen Orbits zwischen den Postures an. Die eingeschobene Ebene **Situationen** enthält Marker, denen bidirektional eine oder mehrere Postures zugeordnet werden können.

jeweils eine der inferierten Situationen markieren. Zwischen diesen und einzelnen besuchten Knoten des Explorationsgraphen werden bidirektionale Verbindungen erzeugt, wenn auf eine Situationsänderung geschlossen werden kann. Damit wird der Explorationsgraph in Partitionen unterteilt, deren Knoten mit verschieden hoher Wahrscheinlichkeit in einer bestimmten Situation erreichbar sind. Die damit mögliche Wahrnehmung der Umwelt kann später auch durch zusätzliche Sensorik erweitert und verbessert werden.

Die in dieser Arbeit bearbeiteten Heuristiken sollen in nachfolgenden Arbeiten für eine höhere Anzahl Gelenke untersucht und gleichzeitig um andere Entscheidungsheuristiken erweitert werden. Bereits beschriebene mögliche Erweiterungen wie Knotengruppierung oder die Optimierung des Gelernten in einer “Schlafphase” müssen noch in einer Implementation untersucht werden. Diese und weitere Heuristiken können auch nach einer vollständigen Exploration noch für die Bewegung durch den Graphen genutzt werden. Eine flexible Auswahl verschiedener Heuristiken je nach aktuellem Ziel des Roboters ist ebenfalls denkbar.

Das ABC-Learning ist hiermit ein kleines Stück weiter auf dem Weg, einem beliebigen Roboter zu erlauben, seine grundlegende Selbstwahrnehmung und Bewegungsfähigkeit in der Welt selbst zu erlernen.

A. Tabellen

Tabelle A.1.: Festgestellte Gewichte der Semni-Körperteile.

Körperteil	Gewicht
Körper	120g
Oberarm	135g (63g ohne Servo)
Unterarm (mit LiPo-Akku)	177g (105g ohne Servo)
Dynamixel RX-28	72g
Gesamtgewicht	432g

Tabelle A.2.: Winkelgrenzen der Gelenke.

Gelenk	obere Grenze	untere Grenze	Spanne
Unterarmgelenk	$-3,24$ rad	$1,91$ rad	$5,15$ rad
Oberarmgelenk	$0,94$ rad	$2,72$ rad	$-3,66$ rad

B. Filterkoeffizienten

Der Vollständigkeit halber werden hier die Filterkoeffizienten für das Tiefpassfilter und den differenzierenden Tiefpassfilter der Hardwareimplementation angegeben.

Tabelle B.1.: Koeffizienten s_n des Tiefpass-Differenziators.

Index	Koeffizient	Index	Koeffizient
a_0	0.000355954970989	a_{24}	-0.000355954970989
a_1	0.000705819453735	a_{25}	-0.000705819453735
a_2	0.00104360716994	a_{26}	-0.00104360716994
a_3	0.00136353847812	a_{27}	-0.00136353847812
a_4	0.0016601392649	a_{28}	-0.0016601392649
a_5	0.00192833460858	a_{29}	-0.00192833460858
a_6	0.00216353561247	a_{30}	-0.00216353561247
a_7	0.00236171792218	a_{31}	-0.00236171792218
a_8	0.00251949058345	a_{32}	-0.00251949058345
a_9	0.00263415406231	a_{33}	-0.00263415406231
a_{10}	0.00270374643484	a_{34}	-0.00270374643484
a_{11}	0.00272707695624	a_{35}	-0.00272707695624
a_{12}	0.00270374643484	a_{36}	-0.00270374643484
a_{13}	0.00263415406231	a_{37}	-0.00263415406231
a_{14}	0.00251949058345	a_{38}	-0.00251949058345
a_{15}	0.00236171792218	a_{39}	-0.00236171792218
a_{16}	0.00216353561247	a_{40}	-0.00216353561247
a_{17}	0.00192833460858	a_{41}	-0.00192833460858
a_{18}	0.0016601392649	a_{42}	-0.0016601392649
a_{19}	0.00136353847812	a_{43}	-0.00136353847812
a_{20}	0.00104360716994	a_{44}	-0.00104360716994
a_{21}	0.000705819453735	a_{45}	-0.000705819453735
a_{22}	0.000355954970989	a_{46}	-0.000355954970989
a_{23}	0.0	a_{47}	0.0

Die Koeffizienten des FIR-Filters nach Selesnick lassen sich durch

$$s_n = -\frac{2\pi}{N^2}\sin(\frac{2\pi}{N}(n-\frac{N-1}{2})) \tag{B.1}$$

berechnen, N ist dabei die Anzahl der Koeffizienten, die die Grenzfrequenz ω_c des Tiefpassfilters bestimmen. Für die Anwendung im Semni wurde $N = 48$ gewählt und damit eine Grenzfrequenz von $w_c \approx 2$ Hz angestrebt, siehe Selesnick (2002). Die Berechnung der gefilterten Werte erfolgt wie bei einem herkömmlichen FIR-Filter, welcher allgemein in Abbildung B.1 dargestellt ist.

Für die Tiefpassfilterung der Geschwindigkeit wird ein kaskadiertes IIR-Filter verwendet, dessen Koeffizienten mit einem Filterdesignwerkzeug ermittelt worden sind[12]. Die gewünschte Grenzfrequenz wurde hier mit $\omega_c = 1.5$ Hz festgelegt. Die Berechnung erfolgt entsprechend Abbildung B.2.

Tabelle B.2.: Koeffizienten des IIR-Tiefpassfilters für das erste und zweite IIR-Filter in der Kaskade.

Koeffizient	Wert
α_0	0.001809107551903538
α_1	0.003618215103807076
α_2	0.001809107551903538
β_1	−1.836432253077509460
β_2	0.843668683285123677
α'_0	0.002322370651874689
α'_1	0.004644741303749378
α'_2	0.002322370651874689
β'_1	−1.874725794893530080
β'_2	0.884015277501028929

[12] Iowa Hills IIR Filter Designer, `http://iowahills.com`.

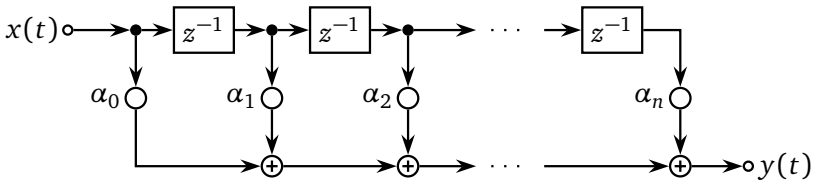

Abbildung B.1.: FIR-Filter mit $n+1$ Koeffizienten.

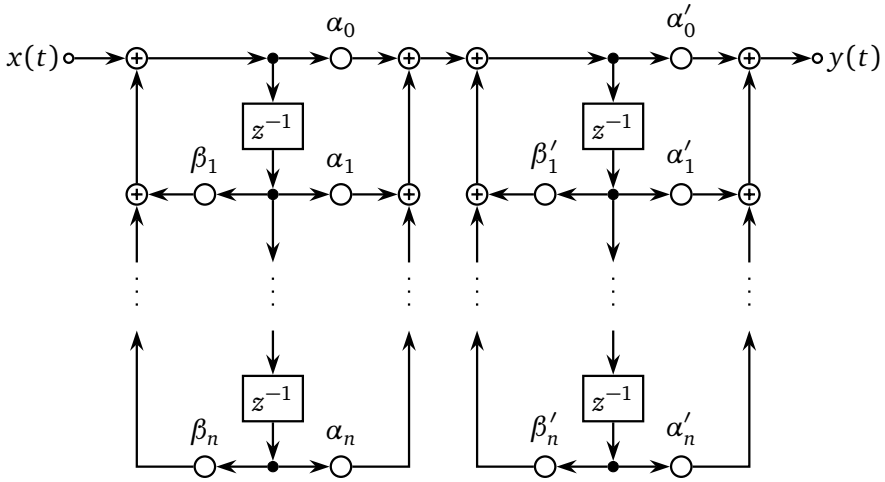

Abbildung B.2.: IIR-Filter mit $4n+2$ Koeffizienten.

Literaturverzeichnis

Arnold, V. I. (1989). *Mathematical Methods of Classical Mechanics*. Springer-Verlag.

Arrowsmith, D. K. und Place, C. M. (1994). *An introduction to dynamical systems*. Cambridge University Press.

Baranès, A. und Oudeyer, P.-Y. (2010). *Intrinsically motivated goal exploration for active motor learning in robots: A case study*. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010).

Bernstein, N. A. (1967). *The co-ordination and regulation of movements*. Pergamon Press, Oxford.

Broer, H., Takens, F., und Hasselblatt, B. (2010). *Handbook of Dynamical Systems*, Band 3. Elsevier Science.

Brooks, R. (1991). *Intelligence Without Representation*. Artificial Intelligence, 47:139–159.

Carr, J. (1981). *Applications of Centre Manifold Theory*. Applied Mathematical Sciences Series, Band 35. Springer-Verlag.

Christensen, D., Bordignon, M., Schultz, U. P., Shaikh, D., und Stoy, K. (2009). *Morphology Independent Learning in Modular Robots*. In: Distributed Autonomous Robotic Systems 8., Seiten 379–391. Springer.

Demiris, Y. und Dearden, A. (2005). *From motor babbling to hierarchical learning by imitation: A robot developmental pathway*. In: EpiRob 2005.

Der, R. und Martius, G. (2006). *From Motor Babbling to Purposive Actions: Emerging Self-exploration in a Dynamical Systems Approach to Early Robot Development.* In: SAB, Band 4095 von Lecture Notes in Computer Science, Seiten 406–421. Springer.

Der, R., Martius, G., und Pfeifer, R. (2012). *The Playful Machine: Theoretical Foundation and Practical Realization of Self-Organizing Robots*. Cognitive Systems Monographs. Springer.

Der, R., Steinmetz, U., und Pasemann, F. (1999). *Homeokinesis: A New Principle to Back Up Evolution with Learning*. Max-Planck-Institut für Mathematik in den Naturwissenschaften.

Dörner, D. (2001). *Bauplan für eine Seele*. Rowohlt Taschenbuch Verlag GmbH, 2. Ausgabe.

Dörner, D. (2002). *Die Mechanik des Seelenwagens*. Verlag Hans Huber.

Fischer, R. (2009). *Elektrische Maschinen*. Carl Hanser Verlag, 14. Ausgabe.

Goldstein, H., Poole, C., und Safko, J. (2012). *Klassische Mechanik*. Lehrbuch Physik. Wiley.

Hering, Ekbert und Bressler, K. (2005). *Elektronik für Ingenieure und Naturwissenschaftler*. Springer Science+Business Media, 5. Ausgabe.

Hild, Manfred und Kubisch, M. (2011). *Self-Exploration of Autonomous Robots Using Attractor-Based Behavior Control and ABC-Learning*. In: 11th Scandinavian Conference on Artificial Intelligence (SCAI 2011).

Kubisch, M. (2008). *Modellierung und Simulation nichtlinearer Motoreigenschaften*. Studienarbeit, Humboldt-Universität zu Berlin.

Kubisch, M. (2010). *Intrinsisch motivierte Exploration sensomotorischer Zustandsräume*. Diplomarbeit, Humboldt-Universität zu Berlin.

Kubisch, M., Benckendorff, C., und Hild, M. (2011). *Balance Recovery of a Humanoid Robot Using Cognitive Sensorimotor Loops (CSLs)*. In: Proceedings of the 14th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines.

Le-Tien, L. und Albu-Schäffer, A. (2012). *Adaptive friction compensation in trajectory tracking control of DLR medical robots with elastic joints*. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012).

Leimkuhler, B. und Reich, S. (2004). *Simulating Hamiltonian Dynamics*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press.

Maes, P. und Brooks, R. A. (1990). *Learning to Coordinate Behaviors*. Seiten 796–802.

Markl, M. (2010). *Sensomotorische Kartografierung und Planung unter Verwendung neuronaler Netze und homöokinetischer Exploration*. Diplomarbeit, Humboldt-Universität zu Berlin, Institut für Informatik.

Meiss, J. (2007). *Dynamical systems*. Scholarpedia, 2(2):1629. revision 121407.

Milnor, J. W. (2006). *Attractor*. Scholarpedia, 1(11):1815. revision 91013.

Nise, N. (2004). *Control Systems Engineering*, Band 1. Wiley.

Olsson, H und Aström, K. (1998). *Friction Models and Friction Compensation*. European Journal of Control, 4:176–195.

Pfeifer, R. und Bongard, J. C. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. The MIT Press.

Schönhammer, K. (2010). *Skript zur Vorlesung Mechanik*. Institut für theoretische Physik, Universität Göttingen.

Selesnick, I. (2002). *Narrowband Lowpass Digital Differentiator Design*. Asilomar Conference on Signals, Systems and Computers, 1:360–364.

Spranger, M. und Loetzsch, M. (2009). *The semantics of SIT, STAND, and LIE embodied in robots*. Proceedings of the 31th Annual Conference of the Cognitive Science Society .

Stribeck, R. (1902). *Die wesentlichen Eigenschaften der Gleit- und Rollenlager*. Zeitschrift des Vereines Deutscher Ingenieure, 46:1342–1348,1432–1437.

Werner, B. (2013). *Entwicklung eines adaptiven sensomotorischen Algorithmus zur dynamischen Bewegungssteuerung autonomer Roboter*. Diplomarbeit, Humboldt-Universität zu Berlin, Institut für Informatik.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Weiterhin erkläre ich, eine Diplomarbeit in diesem Studiengebiet erstmalig einzureichen.

Berlin, den 16. Juni 2014 Stefan Bethge