Masterarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Intelligente Systeme und Robotik

Situationserkennung für autonome Roboter basierend auf Diffusionsprozessen in sensomotorischen Graphen

Marcus Janz
Matrikelnummer: 4298510
marcus.janz@fu-berlin.de

Betreuer: Prof. Dr. Manfred Hild
Eingereicht bei: Prof. Dr. Raúl Rojas

Berlin, 3. Januar 2015

Zusammenfassung

In dieser Arbeit wird ausgehend vom sogenannten *ABC-Learning* eine Situationserkennung für einen Roboter auf Basis sensomotorischer Daten realisiert. Dabei wird der Roboter inklusive seiner Umwelt als ein dynamisches System aufgefasst und die Einflüsse der verschiedenen Situationen auf dieses System untersucht. Anschließend wird das ABC-Learning um ein Verfahren erweitert, das mittels Diffusionsprozessen diese Einflüsse erkennen und den entsprechenden Situationen zuordnen kann. Zusätzlich wird eine Modifikation dieses Verfahrens erläutert und in den Auswertungen der Experimente gezeigt, dass diese zu einer deutlichen Verbesserung führt.

Inhaltsverzeichnis

1 Einleitung

Beobachtet man die Umwelt in der wir Menschen leben, so stellt man fest, dass diese sich ständig ändert und zu zwei verschiedenen Zeitpunkten nie den exakt gleichen Zustand hat. Trotzdem schafft es der Mensch, sich in dieser Welt zurechtfinden und mit ihr zu interagieren. Dies ist ein Hinweis, dass unsere Verhaltensweisen und Aktionen robust gegenüber diesen Änderungen sind. Es ist kaum vorzustellen, wie wir unser Leben bestreiten könnten ohne die Adaptivität unseres Organismus. Ein Beispiel hierfür ist das Laufen: Wir Menschen können uns in verschiedensten Terrains sicher fortbewegen, ohne bewusst darauf Einfluss nehmen zu müssen, wie wir dies tun. Es gibt jedoch auch Fälle, in denen wir uns aktiv anpassen. Beispielsweise verhalten sich die meisten Menschen an ihrem Arbeitsplatz anders, als vielleicht zu Hause bei ihrer Familie. Die mehr oder weniger bewussten Verhaltensänderungen basieren auf den unterschiedlichen Gegebenheiten oder auch Situationen. In den meisten Fällen ist es für uns kein Problem, die grundlegende Situation zu erkennen und unser Verhalten entsprechend anzupassen.

Das in Moore (1965) eingeführte Mooresche Gesetz und im Allgemeinen die Verbesserungen von Technik und Algorithmik in der Robotik, lassen hoffen, dass Roboter irgendwann zunehmend in den Alltag der Menschen integriert werden können. Während die Industrie sie schon seit über 50 Jahren einsetzt, könnte sich ihr Einsatz in Zukunft immer mehr in die öffentlichen Räume, Haushalte und Büros, also in die gleiche dynamische Umwelt verlagern, in der wir uns täglich bewegen. Im Unterschied zu den klar definierten Industrieumgebungen sollten sie hier ähnlich adaptiv wie wir agieren, um erfolgreich zu sein. Zukünftige Roboter sollten entweder derart robuste Verhaltensweisen haben, dass eventuelle Änderungen der Gegebenheiten irrelevant sind oder aber auf Veränderungen reagieren und ihre Verhaltensweisen ändern. Hierfür ist es wichtig, zunächst einmal zu erkennen, wann eine neue Situation eingetreten ist, diese in irgendeiner Form zu kategorisieren und später wiederzuerkennen. Hierfür stehen ihnen verschiedene Sensoren zur Verfügung, die einen Ausschnitt der Umwelt erfassen können. Aufbauend auf diesen Daten lassen sich Rückschlüsse auf die aktuelle Situation ziehen. Für den Bereich der Robotik wäre ein typischer Ansatz, die gewonnenen Daten mit den Methoden des maschinellen Lernens zu analysieren und auszuwerten.

In dieser Arbeit wird ein etwas anderer Ansatz verfolgt. Ein Roboter kann in seiner Gesamtheit bestehend aus Hardware, Software und seiner Umwelt als ein dyna-

misches System aufgefasst werden. Merkmale und Eigenschaften eines bestimmten dynamischen Systems können sich, abhängig von der Situation in der man es betrachtet, ändern. Diesen Zusammenhang am Beispiel des in Abbildung 1 dargestellten Semni zu untersuchen, bildet den Kern dieser Arbeit.

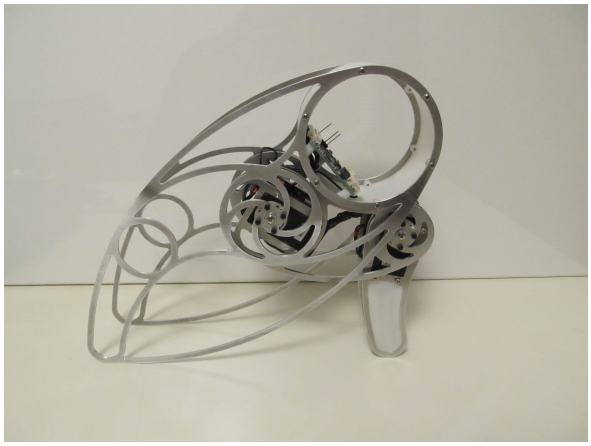


Abbildung 1: Abgebildet ist hier der Roboters Semni des Forschungslabors Neurorobotik der Beuth Hochschule für Technik Berlin. Am Beispiel dieses Roboters wird in dieser Arbeit untersucht, ob es möglich ist, auf Grund der Merkmale und Eigenschaften eines bestimmten dynamischen Systems verschiedene Situationen zu erkennen.

Fragestellung und Zielsetzung

Die Merkmale und Eigenschaften des dynamischen Systems, welches durch den Roboter Semni, seine Regelung und Umwelt gegeben ist, sollen für verschiedene Situationen in einem Graphen gespeichert werden. Aufbauend auf der Annahme, dass die verschiedenen Situationen unterschiedliche Graphen ausbilden, soll in dieser Arbeit überprüft werden, ob es möglich ist, basierend auf sogenannten *Diffusionsprozessen* in diesen Graphen, Rückschlüsse auf die aktuelle Situation zu ziehen.
Zunächst soll ein Verfahren implementiert werden, welches ermöglicht, die Merkmale und Eigenschaften des dynamischen Systems zu explorieren und in Form eines Graphen abzuspeichern. Ein Verfahren, das auf Diffsionsprozessen beruht, soll dann entwickelt und evaluiert werden. Das entwickelte Verfahren sollte für eine spätere Implementierung auf dem Semni möglichst effizient sein, da dessen Ressourcen relativ beschränkt sind. In der vorliegenden Arbeit sollen deswegen verschiedene Heuristiken für die verschiedenen Problemstellungen verwendet werden, da sich in der Forschung gezeigt hat, dass diese oft, trotz geringem Rechenaufwand, schon sehr gute Resultate erzielen können (Gigerenzer u. a. 2011), (Gigerenzer u. Gaissmaier 2011), (Gigerenzer u. Brighton 2009).

Aufbau der Arbeit

Zunächst wird die Arbeit in den Stand der Forschung eingebettet. Die folgenden Kapitel orientieren sich an den Teilaspekten der Problemstellung wie sie in Abbildung 2 dargestellt sind. Als erstes wird auf die Grundlagen eingegangen. Es werden unter anderem zentrale Begriffe eingeführt und erklärt, welche im weiteren Verlauf der Arbeit gebraucht werden oder für das Verständnis notwendig sind. Aufbauend auf den Grundlagen wird danach auf das Verfahren eingegangen, das den Graphen aufbauen soll. Dabei handelt es sich um das ABC-Learning, welches in Abschnitt 4 erläutert wird. Darauf folgend werden der zugehörige Versuchsaufbau und die Hauptpunkte der Implementierung beschrieben. Anschließend wird der Kern der Arbeit vorgestellt, die Situationserkennung, und das entwickelte Verfahren beschrieben und erklärt. Im Anschluss wird der Aufbau und Ablauf der Experimente dargelegt, sowie die Ergebnisse präsentiert und ausgewertet. Es folgt eine Diskussion verschiedener Aspekte des Verfahrens. Zum Abschluss wird ein Fazit gezogen, sowie ein Ausblick über weitere mögliche Forschungen in diesem Bereich gegeben.

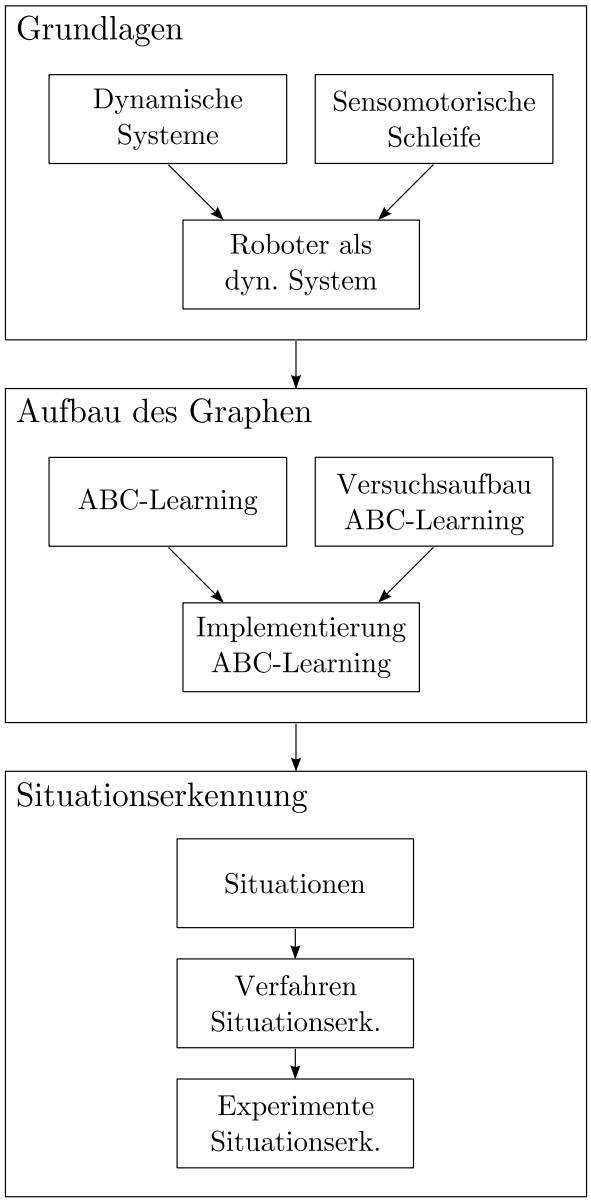


Abbildung 2: Dargestellt ist hier die Zerlegung der Problemstellung in mehrere Teilaspekte, an denen sich der Aufbau dieser Arbeit orientiert. Die Pfeile stellen dabei Abhängigkeiten der einzelnen Aspekte untereinander dar. Beispielsweise baut die Sichtweise, den Roboter als dynamisches System aufzufassen, auf den Begriffen des dynamischen Systems und der sensomotorischen Schleife auf, die im Abschnitt 3 erklärt werden.

2 Stand der Forschung

Betrachtet man die letzten Jahrzehnte der Robotik, so lässt sich nach Murphy (2000) die Forschung auf diesem Gebiet meist in eins von drei Paradigmen einordnen. Der erste und älteste Ansatz funktioniert nach dem Prinzip, dass der Roboter zunächst die Welt wahrnimmt, seine nächste Handlung plant und diese dann ausführt. Der Schwerpunkt liegt hierbei auf dem Planen jeder Handlung in jedem Schritt. Die wahrgenommenen Daten werden in einer internen Repräsentation der Welt zusammengefasst. Weitergehend wird davon ausgegangen, dass die Wahrnehmung dafür sorgen soll, dass die interne Repräsentation immer der externen realen Welt entspricht (Albus u. Meystel 2001). Abgebildet ist dieser Ansatz in Abbildung 3

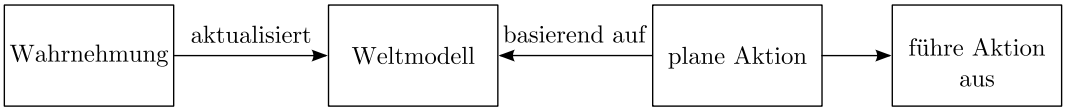


Abbildung 3: Dargestellt ist der erste der drei Ansätze. Die Wahrnehmung aktualisiert das Weltmodell, auf dessen Basis die nächste Akion geplant wird, die anschließend ausgeführt wird.

Dies führt dazu, dass Handlungen nur auf Grund des internen Weltmodells ausgeführt werden und es keine direkte Verbindung von Wahrnehmung und Handlung gibt. Dieser Ansatz eignet sich vor allem dann, wenn das Einsatzgebiet des Roboters leicht zu formalisieren und vorhersehbar ist. Die Welt ist jedoch meistens viel zu komplex, weshalb keine Repräsentation dieser jemals gerecht werden kann. Da die Handlungen nur auf dem internen Weltmodell beruhen, kommt es schnell dazu, dass die Aktionen in der realen Welt nicht zum Erreichen des eigentlichen Ziels führen (Arkin 1998).
Der zweite Ansatz, der als Gegenentwurf zum ersten Ansatz Ende der 1980er Jahre entwickelt wurde, verzichtete komplett auf das Planen von Handlungen. Bei diesem Paradigma führt eine Wahrnehmung direkt zu einer Handlung. Dabei gibt es mehrere Verbindungen von Wahrnehmungen zu Handlungen. Diese laufen alle nebenläufig ab und können so zu einem komplexeren, emergenten Verhalten führen (Murphy 2000), (Arkin 1998), (Brooks 1990). Dargestellt ist dieser Ansatz in Abbildung 4.

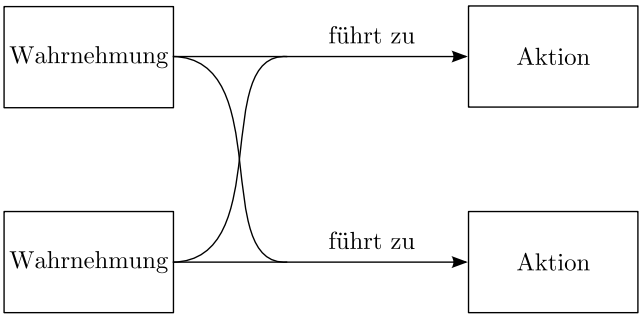


Abbildung 4: Abgebildet ist der zweite Ansatz, bei dem es keine Planung mehr gibt. Die Wahrnehmung führt direkt zu einer Aktion. Es ist auch möglich mehrere Wahrnehmungen mit mehreren Aktionen zu verbinden, die zu einem emergenten Verhalten führen können.

Ein gutes Beispiel hierfür ist das aufstehende Bein in Hild u. Kubisch (2011). So beschreibt Hild in dieser Veröffentlichung, wie ein Bein eines humanoiden Roboters[1] aus liegender Position aufzustehen vermag. Das Bein hat drei aktuierte Gelenke und je einen Winkelsensor. Die Sensoren sind jeweils lokal direkt durch eine Regelschleife mit den Motoren verbunden, wobei die Regelschleife ein Verhalten zeigt, das dazu führt, dass sich die Aktuatoren immer entgegen einer wirkenden Kraft richten. Dieses simple lokale Verhalten an jedem Gelenk führt im Zusammenspiel dazu, dass sich das Bein aufrichtet, wie es in Abbildung 5 zu sehen ist.

[1] http://www.neurorobotik.de/robots/myon_de.php

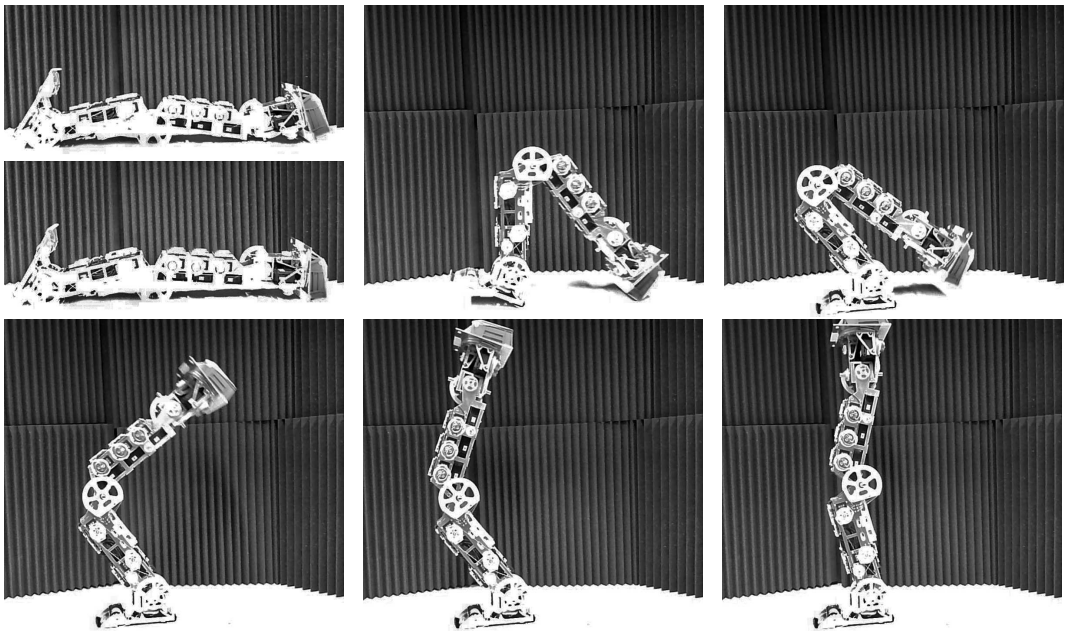


Abbildung 5: Dargestellt ist, wie sich das Bein des humanoiden Roboters „Myon“ aus liegender Postion aufrichtet. Es gibt keinen zentralen Planer. Jedes Gelenk wird lediglich durch eine eigene Regelschleife betrieben. Im Verlauf der einzelnen Bilder ist zu sehen, wie das Bein sich schrittweise aufrichtet. Die Grafik wurde aus Hild u. Kubisch (2011) entnommen.

Auch die in dieser Arbeit verwendete Roboterplattform wird mittels zwei lokaler Regelschleifen gesteuert und kann so komplexe Bewegungen erzeugen.
Der zuletzt genannte Ansatz bietet viele Vorteile bezüglich der Fehlertoleranz und Geschwindigkeit, da eine interne Repräsentation der Welt und die darauf aufbauende Planung entfallen. Nach Murphy (2000) zeigten weitere Forschungsarbeiten jedoch, dass eine seltene Planung, die die einzelnen Verhalten überwacht, viele Probleme des zweiten Ansatzes lösen kann. Diese Erkenntnis führte zum dritten Ansatz, welcher ein hybrider Ansatz der ersten beiden ist und auch in der vorliegenden Arbeit eingesetzt wird. Ein Gesamtziel wird zunächst in mehrere Teilziele zerlegt und jeweils das beste Verhalten ausgewählt, um das Teilziel zu erreichen. Dieses Prinzip ist in Abbildung 6 dargestellt.

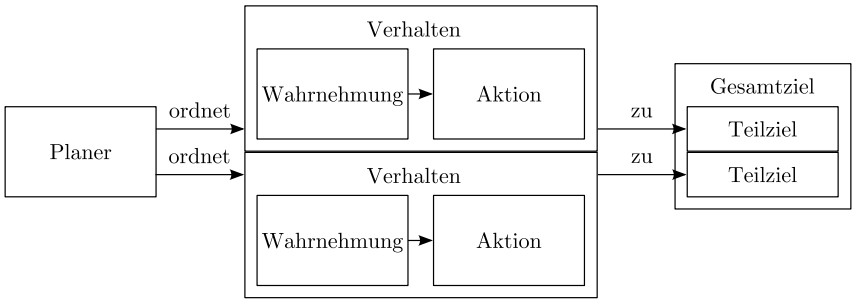


Abbildung 6: Abgebildet ist der hybride dritte Ansatz. Ein Planer ordnet verschiedene Verhalten den Teilzielen zu. Auf diese Weise soll am Ende das Gesamtziel erreicht werden.

Die Sensordaten stehen dabei sowohl den einzelnen Verhalten als auch dem Planer zur Verfügung. Das in dieser Arbeit eingesetzte *Attractor-Based Behavior Control Learning* Hild u. Kubisch (2011) stellt verschiedene Parameter der Regelschleifen durch einen Planer ein, mit dem Ziel den Roboter sich selbst und seine Umwelt erforschen zu lassen. Währenddessen übernehmen die Regelschleifen, dem zweiten Ansatz folgend, die Steuerung des Roboters. Der Planer wird in der vorliegenden Arbeit dahingehend erweitert, dass er die Ergebnisse der Verhalten überwacht, um so Situationen erkennen zu können.

Klassischerweise wird Situationserkennung im Umfeld der Robotik über eine Kamera realisiert. Ein Beispiel hierfür ist Milford (2013). In diesem Paper werden Bildsequenzen in einer Datenbank abgelegt, um diese später mit aktuellen Bildsequenzen abgleichen zu können. Dies kann dafür genutzt werden, um z. B. verschiedene Räumlichkeiten wiederzuerkennen. Abhängig von der Definition des Begriffs *Situation* kann man dies auch als eine Art Situationserkennung interpretieren. Ein anderer Ansatz könnte sein, eine Situation anhand von bekannten Objekten im Umfeld zu erkennen. Arbeiten wie Ortiz (2012) und Bay u. a. (2006) analysieren hierzu ebenfalls Kamerabilder, um markante Punkte von Objekten zu extrahieren und später wieder zuordnen zu können. Der in der vorliegenden Arbeit gewählte Ansatz, Situationen über die Merkmale eines dynamischen Systems wiederzuerkennen, weicht von den anderen genannten Ansätzen deutlich ab.

Diese Arbeit ist weiterhin im Kontext zu anderen Arbeiten zu sehen, die ebenfalls am Forschungslabor Neurorobotik entstanden sind bzw. entstehen werden. Eine thematisch verwandte Arbeit stammt von Stefan Bethge, der seine Diplomarbeit über das Thema „ABC-Learning: Ein Lernverfahren zur modellfreien Selbstexploration autonomer Roboter“ geschrieben hat (Bethge 2014). Dort wurde gezeigt, dass eine modellfreie Selbstexploration mit dem ABC-Learning realisiert werden kann. Weiter-

gehend wurden in diesem Zusammenhang verschiedene Heuristiken zur Exploration untersucht. Eine weitere Arbeit stammt von Benjamin Werner und hat den Titel „Entwicklung eines adaptiven sensomotorischen Algorithmus zur dynamischen Bewegungssteuerung autonomer Roboter“ (Werner 2013). Es wurde untersucht, wie das mit langsamen Bewegungen arbeitende ABC-Learning dynamischer gestaltet werden kann. Dazu wurde das sogenannte *Kick-Fly-Catch-Prinzip* implementiert und später ein *Ramp-Catch-Prinzip* entwickelt. In Abbildung 7 ist eine Übersicht über verwandte Themengebiete des Forschungslabors Neurorobotik gegeben.

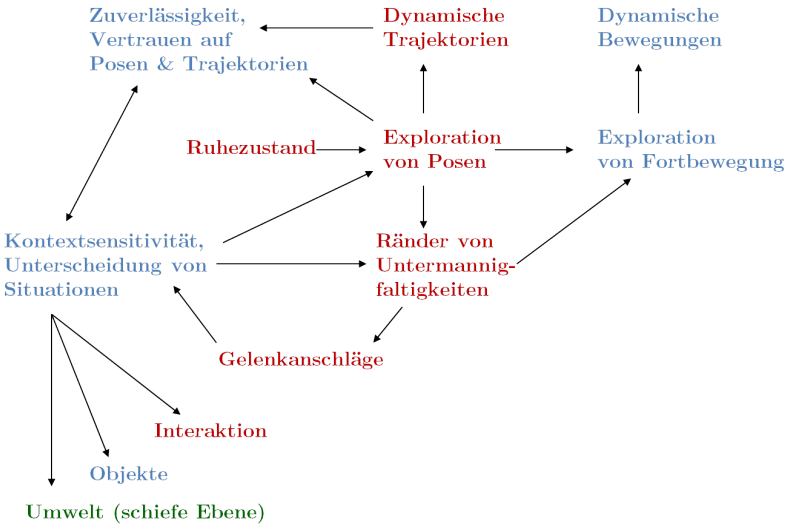


Abbildung 7: Die Abbildung zeigt den Kontext, in dem diese Arbeit am Forschungslabor Neurorobotik entstanden. Diese Grafik ist aus einer internen Präsentation, die die grundlegenden nächsten Schritte in der Arbeitsgruppe zur autonomen Selbstexploration beliebiger Roboter festlegte. In rot sind Themen dargestellt, die bereits bearbeitet wurden. Blau sind Themenfelder die in folgenden Forschungsarbeiten untersucht werden sollen. Das hier untersuchte Thema wurde, im Unterschied zur original Grafik, in grün markiert.

All diese Arbeiten bauen zum Teil auf den gleichen Grundlagen wie die vorliegende Arbeit auf und sollen zusammen mit dieser und anderen kommenden Arbeiten die autonome Selbstexploration beliebiger Roboter ermöglichen.

3 Grundlagen

In diesem Kapitel werden die Grundlagen, die zum Verständnis der Arbeit nötig sind, vermittelt. Dabei wird der erste Abschnitt die Theorie dynamischer Systeme einführen und dabei auf die Begriffe *Attraktor* und *Bifurkation* eingehen. Der nächste Abschnitt erläutert die *sensomotorische Schleife*, sodass darauf aufbauend der Zusammenhang von dynamischen Systemen und Robotern hergestellt werden kann.

3.1 Dynamisches System

Die Idee, Bewegungen physikalischer Systeme mit Gleichungen zu beschreiben, geht auf Isaac Newton zurück. Dieser beschäftigte sich unter anderem mit den Bewegungen der Planeten. Hierzu betrachtete er die Position der Planeten sowie deren Ableitungen: Geschwindigkeit und Beschleunigung. Die mathematischen Mittel, die hierfür verwendet werden, sind Differentialgleichungen (Alligood u. a. 1997).
Ein Beispiel für solch eine Gleichung ist die newtonsche Bewegungsgleichung:

$$m \cdot \ddot{x}(t) = F(x(t), t)$$

Diese beschreibt die Bewegung eines Masseteilchens mit Hilfe der wirkenden Kraft F, die von der Zeit t und der Position eines Masseteilchens $x(t)$ abhängt. Eine Abbildung $x(t)$, die für ein möglicherweise unendliches Intervall die Differentialgleichung erfüllt, wird als dynamisches System bezeichnet (Teschl 2012).

3.1.1 Definition

Nach Alligood u. a. (1997) werden *dynamische Systeme* beschrieben durch einen Zustand $x \in X$ aus einem Zustandsraum $X \subset \mathbb{R}^n$ mit $n \in \mathbb{N}$, welcher eine Mannigfaltigkeit[2] in $\mathbb{R}^n$ bildet, sowie einer Zustandsübergangsfunktion $F : X \to X$ für zeitdiskrete oder einer Differentialgleichung für zeitkontinuierliche Systeme. Somit lässt sich die Entwicklung eines Zustands $x(t)$ über die Zeit $t \in \mathbb{R}$ für zeitkontinuierliche oder $t \in \mathbb{N}$ für zeitdiskrete Systeme beispielsweise als $x'(t) = H(t, x(t))$ mit $H : \mathbb{R} \times X \to X$ oder $x(t) = F(x(t-1))$ formalisieren. Da die Theorie der dynamischen Systeme auf Roboter angewendet werden soll, die ihre Sensorwerte in diskreten Schritten abtasten, wird hier nicht weiter auf zeitkontinuierliche Systeme eingegangen. Weitere Informationen zu diesen Systemen finden sich in Alligood u. a.

[2]Der Begriff der Mannigfaltigkeit wird hier nicht weiter erläutert, da dessen Eigenschaften auch nicht weiter verwendet werden. Totzdem ist es an dieser Stelle der korrekte Begriff.

(1997).

Ein Beispiel für ein zeitdiskretes dynamisches System ist

$$x(t) = a \cdot \sin(x(t-1))$$

mit $x(t) \in X \subset \mathbb{R}$, $t \in \mathbb{N}$ und dem Parameter $a \in \mathbb{R}$.

3.1.2 Attraktor

Betrachtet man dieses System, so kann man feststellen, dass es Stellen gibt, an denen gilt:

$$\forall n \geq 0 : \; x(t) = F^n(x(t)) \; , \; n, t \in \mathbb{N}$$

Diese Punkte werden *Fixpunkte* genannt. Sie unterteilen sich in stabile, instabile Fixpunkte und Sattelpunkte. *Stabile Fixpunkte* x^* sind solche Punkte, für die gilt:

$$\exists \epsilon > 0 \; \forall x \; |x^* - x| < \epsilon : \lim_{k \to \infty} F^k(x) = x^*$$

Im Gegensatz hierzu wird sich der Zustand für *instabile Fixpunkte* bei einer unendlichen kleinen Störung mit der Zeit immer weiter weg bewegen. Sattelpunkte können nur in mehrdimensionalen Systemen auftauchen und verhalten sich jeweils in mindestens einer Richtung wie ein stabiler und instabiler Fixpunkt (Alligood u. a. 1997). Stabile Fixpunkte sind Attraktoren eines dynamischen Systems. *Attraktoren* sind eine Teilmenge des Phasenraums, auf die sich ein dynamisches System über die Zeit hinbewegt und anschließend dort verweilt. Weitere Attraktoren sind sogenannte k-periodische Orbits, quasiperiodische Orbits und chaotische Attraktoren. Ein Zustand x_p wird als periodischer Zustand der Periode k bezeichnet, wenn für ihn

$$x_p = F^k(x_p) \; k \in \mathbb{N}$$

gilt und k die kleinste natürliche Zahl ist, die diese Bedingung erfüllt. Die Menge der Zustände die durch

$$\bigcup_{0<i\leq k} \{F^i(x_p)\}$$

gebildet wird, bezeichnet man als einen k-periodischen Orbit (Alligood u. a. 1997). Die weiteren Attraktoren sind für diese Arbeit nicht relevant und werden daher nicht näher erläutert. Informationen hierzu finden sich in Alligood u. a. (1997), Teschl (2012) oder Argyris u. a. (2010).

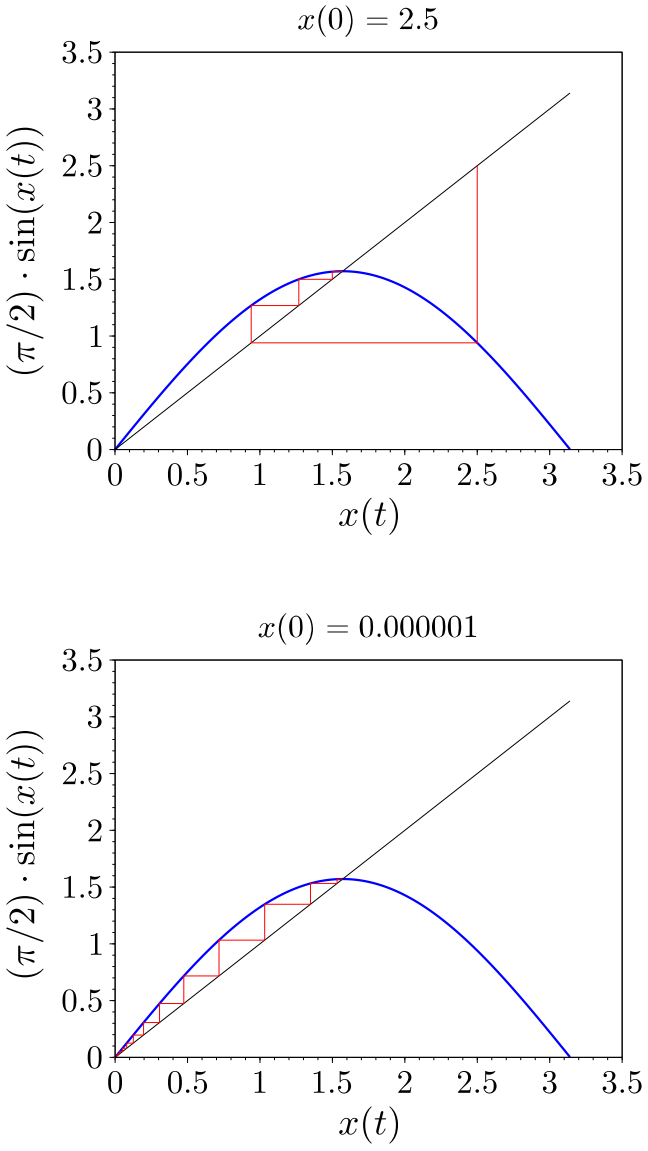


Abbildung 8: Die Abbildung zeigt zwei sogenannte Cobweb-Diagramme des dynamischen Systems für verschiedene Startwerte mit der Übergangsfunktion: $x(t) = (\pi/2) \cdot \sin(x(t-1))$. Cobweb-Diagramme eigenen sich zur Betrachtung des qualitativen Verhaltens einer eindimensionalen Iterierten Abbildung und werden hierzu häufig im Gebiet der dynamischen Systeme eingesetzt. In beiden Diagrammen ist zu sehen, dass sich bei $x(t) = (\pi/2)$ ein stabiler Fixpunkt befindet. Das untere Diagramm verdeutlicht, dass es sich bei $x(t) = 0$ um einen instabilen Fixpunkt handelt.

In Abbildung 8 ist zu sehen, dass das dynamische System mit der Abbildung

$$x(t) = F(x(t-1) = \frac{\pi}{2} sin(x(t-1))$$

für alle Startwerte im Intervall $[0, \pi]$ zwei Fixpunkte hat. Die Fixpunkte befinden sich genau dort, wo die Funktion $f(x) = x$ die Zustandsübergangsfunktion schnei-

det, da hier ein Zustand auf sich selbst abgebildet wird. Der erste Fixpunkt liegt bei $x(t) = 0$. Dieser ist ein instabiler Fixpunkt, wie sich anhand des unteren Diagramms aus Abbildung 8 bereits erahnen lässt. In diesem Diagramm ist zu sehen, dass bei einer kleinen Abweichung von dem Fixpunkt, das System nicht wieder zu diesem zurück läuft. Um sicher zu gehen, bedarf es jedoch einer tiefer gehenden Analyse, da es auch möglich sein könnte, dass die Änderung nur zu groß war. Hierzu betrachtet man die Ableitung am Fixpunkt. Es reicht aus zu überprüfen, ob die Zustandsübergangsfunktion an der zu untersuchenden Stelle (hier für $t = 0$) differenzierbar und der Betrag der Ableitung größer als 1 ist. Da $(\pi/2) \cdot \sin(x(t))$ für $x(t) = 0$ offensichtlich differenzierbar und $|((\pi/2) \cdot \sin(0))'| > 1$ ist, lässt sich auch ein ϵ_0 finden, für das gilt:

$$\exists k \; \forall \epsilon \; 0 < \epsilon < \epsilon_0 : |F^k(0 + \epsilon)| > \epsilon_0 \; .$$

Das bedeutet, dass in dieser Epsilon-Umgebung jede Abweichung vom Fixpunkt verstärkt wird und sich das System somit von diesem wegbewegt (Alligood u. a. 1997).
Auf beiden Diagrammen in Abbildung 8 sieht man, dass sich das System auf den Fixpunkt $x(t) = (\pi/2)$ zu bewegt. Es ist daher zu vermuten, dass es sich hierbei um einen stabilen Fixpunkt handelt. Um dies zu bestätigen muss nun geprüft werden, ob $F(x(t))$ für $x(t) = (\pi/2)$ differenzierbar und der Betrag der Ableitung an dieser Stelle kleiner als 1 ist. Da die Funktion auch hier differenzierbar ist und $|((\pi/2) \cdot \sin((\pi/2)))'| < 1$ gilt, lässt sich nun schließen, dass sich das System in einer Epsilon-Umgebung auf diesen Fixpunkt hinbewegt.
Diese Epsilon-Umgebungen werden nach Alligood u. a. (1997) auch *Basins* genannt und bezeichnen den Einzugsbereich eines Attraktors A, also alle Zustände x aus dem Zustandsraum X für die gilt:

$$\lim_{k \to \infty} f^k(x) \in A \; .$$

Zur Vollständigkeit folgen nun die Stabilitätskriterien für mehrdimensionale Systeme. Nach Alligood u. a. (1997) lassen sich lineare mehrdimensionale Systeme folgendermaßen darstellen:

$$x(t+1) = Mx(t) \; , \; M \in \mathbb{R}^n \times \mathbb{R}^n \; , \; x(t) \in \mathbb{R}^n \; , \; t \in \mathbb{N} \; .$$

Der einzige Fixpunkt solcher Systeme ist im Ursprung. Ohne Beschränkung der Allgemeinheit werden beliebige Verschiebungen des System vernachlässigt, da sich die

Untersuchung des Fixpunkts in diesen Fällen auf die Analyse am Ursprung reduzieren lässt. Sind die Beträge aller Eigenwerte der Matrix M kleiner eins, so bildet der Urspung einen stabilen Fixpunkt. Sind die Beträge größer eins, handelt es sich um einen instabilen Fixpunkt. Gibt es mindestens einen Eigenwert mit Betrag größer eins, einen mit Betrag kleiner eins und sind alle anderen Beträge der Eigenwerte ungleich 1, so befindet sich um Ursprung ein Sattelpunkt.
Diese Kriterien lassen sich auch auf nichtlineare, mehrdimensionale Systeme anwenden. Hierzu wird die Abbildung an der zu untersuchenden Stelle x^* mit Hilfe der Jakobi-Matrix $J(x^*)$ linearisiert. Die Stabilität des Fixpunkts x^* kann nun analog anhand der Jakobi-Matrix $J(x^*)$ bestimmt werden (Alligood u. a. 1997).

3.1.3 Bifurkation

Verändert man nun den Parameter a (bisher $a = \pi/2$) des Systems, wird sich der Attraktor zunächst verschieben. Ändert man den Parameter weiter, wird es zu einer sogenannten Bifurkation kommen. Eine *Bifurkation* beschreibt eine Änderung des Systems, die zur Folge hat, dass sich die Attraktoren des Systems qualitativ ändern. Eine qualitative Änderung ist eine Verminderung oder Vermehrung der Attraktoren, eine Umwandlung einer oder mehrerer Attraktoren von einer Art in eine andere (z. B. stabiler Fixpunkt wird zu p-Orbit). Eine Verschiebung eines Attraktors ist jedoch keine qualitative Änderung und somit auch keine Bifurkation. Dies wird in Abbildung 9 an dem bereits verwendeten Beispielsystem verdeutlicht.

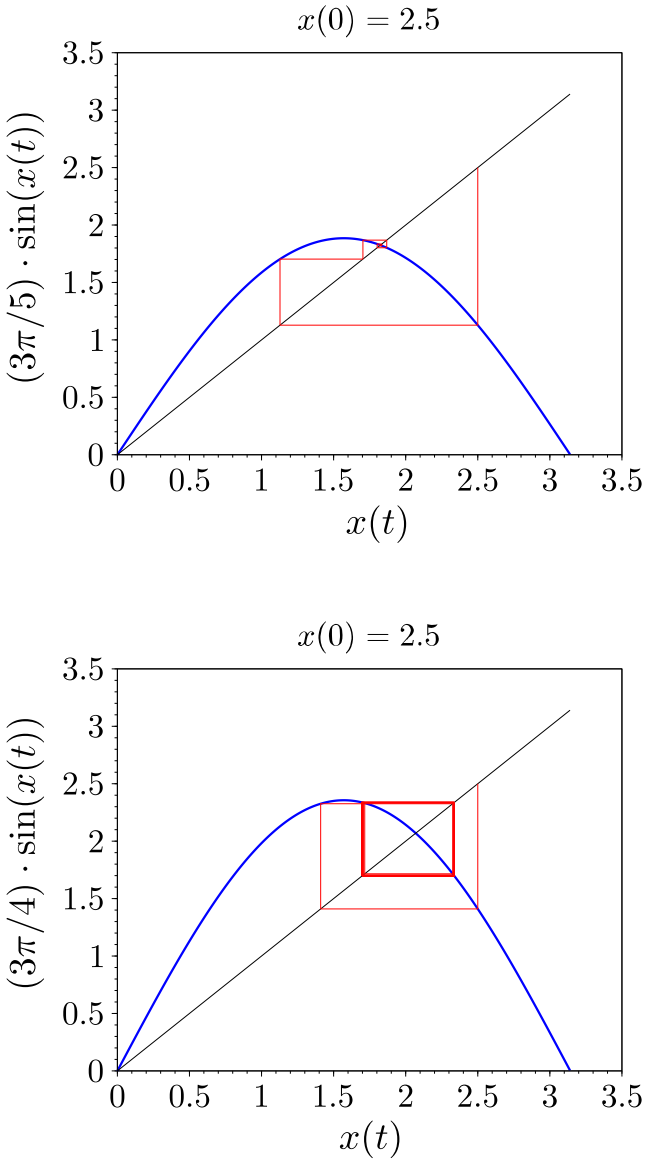


Abbildung 9: Dargestellt sind hier zwei Cobweb-Diagramme des dynamischen Systems mit der Übergangsfunktion $x(t) = (a\pi/2) \cdot \sin(x(t-1))$ mit verschiedenen Werten für a. Oben ist zu sehen, dass es einen stabilen Fixpunkt gibt, dieser jedoch im Vergleich zu Abbildung 8 verschoben ist. Es handelt sich hierbei nicht um eine Bifurkation. Im unteren Diagramm ist zu sehen, dass ein 2-periodischer Orbit entstanden ist. In diesem Fall liegt eine Bifurkation vor.

Mit Hilfe dieses Begriffs wird in Abschnitt 4 beschrieben, wie sich das System beim Wechsel verschiedener Parameter des ABC-Learning verhält.

3.2 Sensomotorische Schleife

Der Begriff *Sensomotorik* bezeichnet das Zusammenwirken von Sensorik und Motorik (Klinke u. Baumann 2010). Eine sensomotorische Schleife verbindet die senso-

rischen Eingänge mit den Motoren. Diese Verbindung kann dabei durch eine Regelung geschehen. Die Regelung ordnet einem sensorischen Input einen Ansteuerungswert der Motoren zu. Dieser Ansteuerungswert führt zu einer Aktion der Motorik. Abhängig von der Morphologie des Roboters, führt dies zu einem Einfluss auf die Umwelt. Der sensorische Eingang und der Roboter, welcher durch seine Sensorik, Regelung, Motorik und Morphologie beschrieben wird, können als Vektoren $x(t) \in \mathbb{R}^n$ und $y(t) \in \mathbb{R}^m$ mit $n, m \in \mathbb{N}$ zum Zeitpunkt $t \in \mathbb{N}$ formalisiert werden. Im einfachsten Fall entspricht die Einwirkung des Roboters einer Funktion $R : \mathbb{R}^n \to \mathbb{R}^m$ mit $R(x(t)) = y(t)$. Sein Einfluss $y(t)$ ist dazu geeignet, eine Manipulation der Umwelt vorzunehmen. Die Umwelt lässt sich dabei als eine Funktion $U : \mathbb{R}^m \to \mathbb{R}^n$ formalisieren. Angewandt auf $y(t)$ entstehen so neue Sensorwerte $x(t+1)$ (Der u. a. 2012). Dies verdeutlicht den Begriff der sensomotorischen Schleife, da diese neuen Sensorwerte $x(t+1)$ zu neuen Einwirkungen $y(t+1)$ führen und so eine unendliche Abfolge sich gegenseitig bedingender Sensorwerte und Einflüsse auf die Umwelt entsteht. Dieser Kreislauf ist in Abbildung 10 dargestellt.

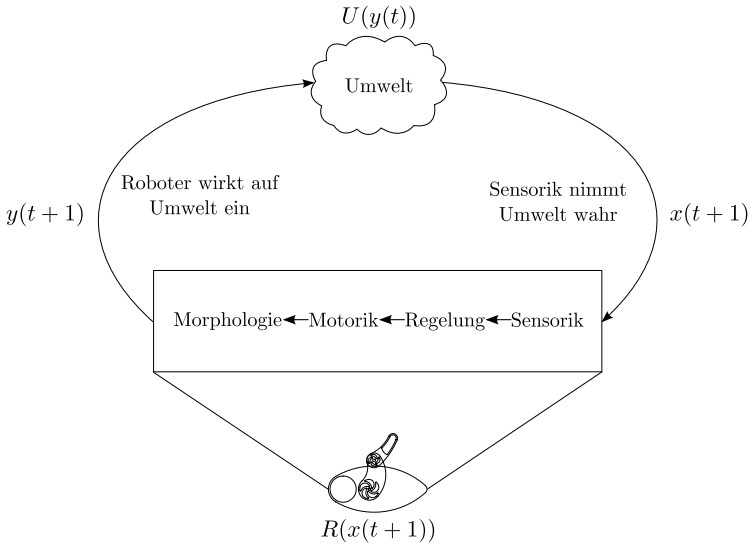


Abbildung 10: Abgebildet ist das Prinzip der sensomotorischen Schleife. Ein Roboter wirkt mittels seiner Motorik auf die Umwelt ein. Diese Manipulation führt dazu, dass die Sensoren wiederum neue Werte messen, auf dessen Basis wieder die Umwelt beeinflusst wird.

Zur Formalisierung des Prinzip bietet es sich an, dieses in ein dazu geeignetes Framework einzubetten. In dieser Arbeit wird hierzu die im vorherigen Abschnitt beschriebene Theorie der dynamischen Systeme genutzt.

3.3 Roboter als dynamisches System

Roboter sind in den meisten Fällen mit Sensorik ausgestattet und regeln auf Grund der gemessenen Werte ihre Motorik. Deshalb kann man alle daran beteiligten Komponenten in ihrer Gesamtheit als sensomotorische Schleife bezeichnen. Zusammen mit seiner Umwelt, die aus Sicht des Roboters als eine Funktion $U : \mathbb{R}^m \to \mathbb{R}^n$ darstellbar ist, lässt sich der Roboter nach obiger Definition als dynamisches System auffassen.
Ein möglicher Zustandsraum dieses Systems kann aus allen möglichen Positionen des Roboters in seiner Umwelt gebildet werden. Dieser Zustand $x(t)$ lässt sich dabei mit Hilfe aller Gelenkwinkel, seiner Lage relativ zur Umwelt und den jeweiligen Winkelgeschwindigkeiten beschreiben. Da die Bewegungen in dieser Arbeit sehr langsam sind, werden jedoch im weiteren Verlauf die Winkelgeschwindigkeiten als konstant null angenommen und der Zustand des Roboters ohne diese beschrieben. Eine weitere Vereinfachung, die getroffen wird, ist, dass sich der in dieser Arbeit betrachtete Roboter nur in einer Ebene innerhalb der Umwelt bewegt, weshalb sich die Lage des Roboters nur über einen sogenannten Körperwinkel beschreiben lässt. Ein Zustand $x(t) \in \mathbb{R}^n$ wird somit folgendermaßen beschrieben:

$$x(t) = (\phi_{\text{Gelenk}_1}, \phi_{\text{Gelenk}_2}, \cdots, \phi_{\text{Gelenk}_{n-1}}, \phi_{\text{Körper}})$$

Am Beispiel des in Abbildung 11 dargestellten Roboter soll dies verdeutlicht werden.

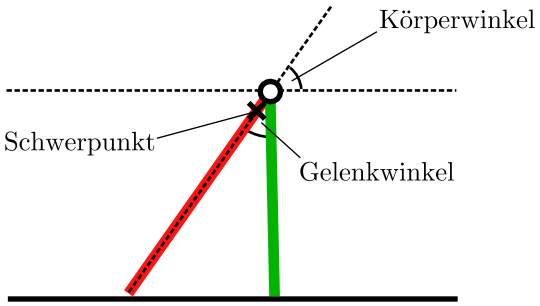


Abbildung 11: Abgebildet ist ein einfacher Roboter. Für diesen Roboter ist der Zustand $x(t)$ zweidimensional. Es gibt einen Gelenkwinkel, der die Stellung der Gliedmaßen zueinander beschreibt, die durch ein Gelenk verbunden sind. Zusätzlich gibt es einen Körperwinkel, der aus der Lage der roten Gliedmaße relativ zum Boden gebildet wird. Für weitergehende Betrachtung ist außerdem der Schwerpunkt eingezeichnet.

Der Zustand $x(t)$ dieses einfachen Roboters wird durch seinen Gelenk- und Körperwinkel bestimmt:

$$x(t) = (\phi_{\text{Gelenk}}, \phi_{\text{Körper}})$$

Der Zustandsraum dieses Roboters bildet somit eine eindimensionale Mannigfaltigkeit in $\mathbb{R}^2$. Diese ist in Abbildung 12 skizziert, wobei gewisse Anschläge angenommen und nur Zustände eingezeichnet wurden, die bei einem gewissen Drehmoment im Gelenk stabil sind. Zustände bei denen der Roboter umfällt wurden durch einen gestrichelten Pfeil dargestellt und bedeuten in diesem Fall immer einen Wechsel der Untermannigfaltigkeit. Wie an diesem Beispiel zu erkennen, sind alle Zustände $x_i, x_j \in U_k$ innerhalb einer Untermannigfaltigkeit *wegzusammenhängend*:

$$\exists h : [a, b] \to U_k \text{ mit } h(a) = x_i \text{ und } h(b) = x_j \text{ und } h \text{ ist stetig}$$

Für Zustände aus verschiedenen Untermannigfaltigkeiten gilt dies nicht immer, wodurch z.B. nach einem Wechsel der Untermannigfaltigkeit der Weg zurück nicht direkt oder gar nicht möglich sein kann.

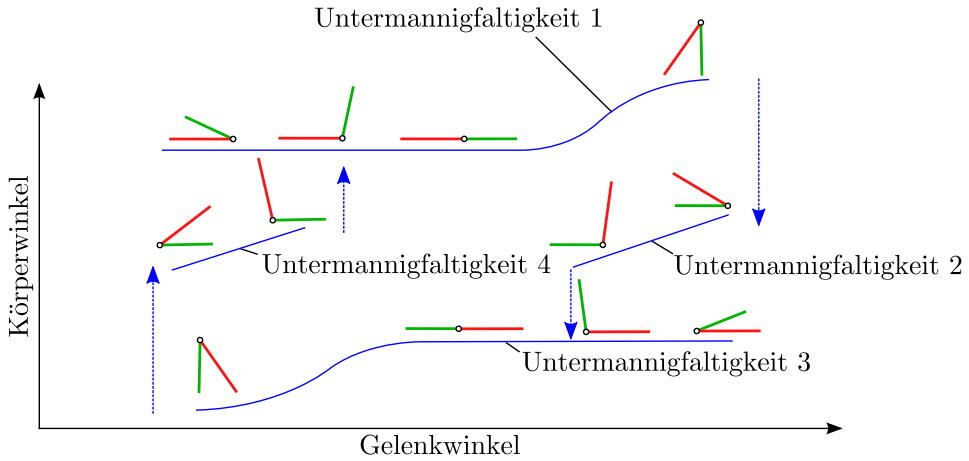


Abbildung 12: Dargestellt ist der Zustandsraum des in Abbildung 11 gezeigten Roboters. Dieser bildet eine Mannigfaltigkeit in $\mathbb{R}^2$ und setzt sich wiederum selbst aus vier Untermannigfaltigkeiten zusammen. Ein Wechsel zwischen diesen erfolgt jeweils durch ein Umfallen, welches durch einen gestrichelten Pfeil markiert ist. Außerdem wurden nur Zustände eingezeichnet die durch den Roboter stabil gehalten werden und Anschläge angenommen, bei denen sich der Zustand in eine Richtung nicht weiter ändert.

Ausgehend von einem Startzustand $x(0)$ können die folgenden Zustände durch eine Übergangsfunktion beschrieben werden, die wie folgt gebildet werden kann: Der Roboter misst mit seinen Sensoren einen Teil seiner Umwelt. Damit erfährt er seinen Zustand $x(t)$ zum Zeitpunkt t. Aufbauend darauf, bestimmt die implementierte Regelung neue Ansteuerungswerte für die Motorik, die abhängig von der Morphologie auf die Umwelt einwirkt $R(x(t)) = y(t)$. Diese Auswirkungen führen zu neuen Messwerten der Sensorik $U(y(t)) = x(t + 1)$. Das dynamische System ist somit beschrieben durch den Zustand $x(t) \in \mathbb{R}^n$ mit $t \in \mathbb{N}$ und der Zustandsübergangsfunktion

$g : \mathbb{R}^n \to \mathbb{R}^n$ mit:

$$x(t+1) = g(x(t)) = U(R(x(t)))$$

Dies ermöglicht es, Roboter und ihr Verhalten mit den mathematischen Mitteln zur Analyse von dynamischen Systemen zu untersuchen (Der u. a. 2012). Darauf aufbauend wird im Abschnitt 4 das ABC-Learning beschrieben, das zur Verhaltenssteuerung eines Roboters dessen Attraktoren bzw. Fixpunkte nutzt.

3.4 Heuristik

In Gigerenzer u. a. (2004) wird eine Heuristik als eine einfache Entscheidungsregel beschrieben, die unter Einbeziehung weniger Informationen schnell zu einem akzeptablen Ergebnis führt. Gerd Gigerenzer beschäftigt sich in seiner Forschung mit solchen Heuristiken und hat in vielen Veröffentlichungen deren Mächtigkeit gezeigt (Gigerenzer u. a. 2011), (Gigerenzer u. Gaissmaier 2011), (Gigerenzer u. Brighton 2009). In diesem Zusammenhang führt Gigerenzer in Gigerenzer u. Selten (2002) den Begriff der *begrenzten Rationalität* ein. Ursprünglich in den 1950er Jahren von Herbert A. Simon eingeführt, wurde er in den Jahren danach mehrfach umgedeutet. Simon beschrieb mit diesem Begriff eine Entscheidungsstrategie von Individuen im Kontext der Wirtschaftspsychologie und grenzte diesen insbesondere von der in vielen Wirtschaftsmodellen getroffenen Annahme ab, dass sich die Teilnehmer *rational* verhalten. Vor diesem Hintergrund bedeutet rationales Verhalten, den Nutzen maximierende Entscheidungen zu treffen. Im Unterschied hierzu bezieht die begrenzte Rationalität die kognitiven Fähigkeiten des Menschen, die Kosten der Entscheidungsfindung und das Konzept eines befriedigenden Nutzenniveaus in die Entscheidungsfindung mit ein. Davon ausgehend führt Gigerenzer diesen Begriff weiter aus und grenzt ihn gegenüber anderen Entscheidungsstrategien wie der Optimierung unter Nebenbedingungen ab. Er nennt dabei drei Eigenschaften, die begrenzte Rationalität typischerweise kennzeichnen. Zunächst gibt es einfache Such-Regeln, mit denen Informationen gewonnen oder Änderungen vorgenommen werden und die solange angewandt werden, bis eine simples Kriterium zum Beenden der Suche erfüllt ist, welches den zweiten Aspekt der begrenzten Rationalität ausmacht. Zuletzt weist diese laut Gigerenzer eine einfache Entscheidungsregel auf, die auf Grund der gesammelten Informationen eine Alternative auswählt, ohne eine aufwändige Optimierung durchzuführen. An dieser Stelle zeigt sich der Zusammenhang von begrenzter Rationalität und Heuristiken. Sie bieten sich immer dann an, wenn zur Lösung eines Problems nicht alle Informationen zur Verfügung stehen und der Aufwand, um mittels komplexer Verfahren die optimale Entscheidung auf Basis aller vorliegenden

Informationen auszuwählen, zu groß ist.

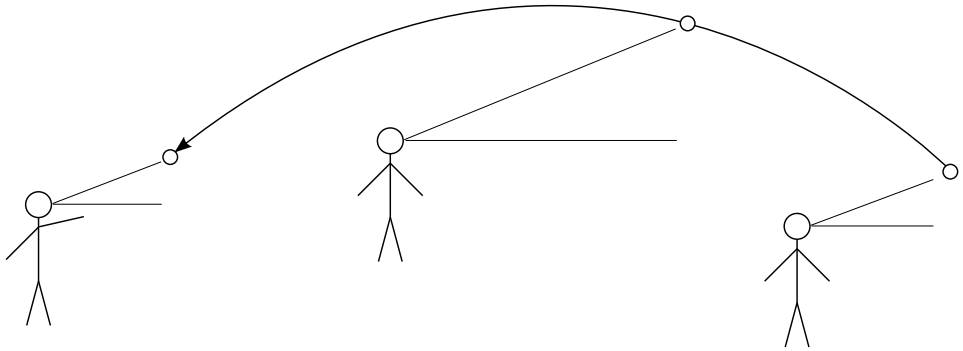


Abbildung 13: Indem der Spieler den Blickwinkel zum Ball konstant hält, schafft er es den Aufschlagorts des Balls zu erreichen und ihn dort fangen zu können. Dies wird als die „Gaze“-Heuristik in McLeod u. Dienes (1996) bezeichnet.

Als Beispiel wird in Gigerenzer u. a. (2004) das Fangen eines Baseballs durch einen Spieler angeführt. Dazu ist es notwendig, dass der Spieler sich zunächst zu der Stelle bewegt, an der der Ball landen wird. Es ist offensichtlich, dass der Spieler weder über alle nötigen Informationen wie Anfangsgeschwindigkeit des Balls, Windgeschwindigkeit usw. verfügt, noch die Kapazität hat die komplexen Berechnungen durchzuführen, die für eine Bestimmung des Aufschlagortes des Balls notwendig sind, bevor der Ball aufschlägt. Als Erklärung dafür, wie der Spieler es trotzdem schaffen kann, den Ball zu fangen, wird die sogenannte „Gaze“-Heuristik aus McLeod u. Dienes (1996) herangezogen. Diese besagt, dass es ausreicht, wenn der Spieler den Ball fixiert und beim Laufen in die ungefähre Richtung des Aufschlagortes seine Laufgeschwindigkeit so anpasst, dass der Blickwinkel zum Ball konstant bleibt, welches in Abbildung 13 verdeutlicht wird.

3.5 Diffusionsprozesse in Graphen

Nach Cormen u. a. (2001) ist ein *Graph* G als ein Tupel zweier Mengen definiert:

$$G = (V, E)$$

In der Menge V sind alle Knoten und in E alle Kanten zwischen diesen enthalten. Eine Kante setzt zwei Knoten über eine Relation R in Verbindung:

$$E = \{(u, v) \in V \times V \ : \ uRv\}$$

Der Graph G ist *ungerichtet*, wenn die Relation R *symmetrisch* ist:

$$\forall u, v \in V : \; uRv \Rightarrow vRu$$

Ist R nicht symmetrisch so ist G gerichtet. Die in dieser Arbeit verwendeten Graphen sind immer gerichtet und wie in Abbildung 14 dargestellt.

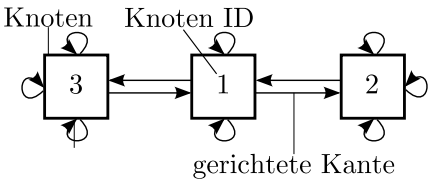


Abbildung 14: Abgebildet ist ein gerichteter Graph. Diese Darstellung wird im Laufe dieser Arbeit immer wieder verwendet und erweitert.

Eine Kante (u, v) geht vom Knoten u aus und zeigt auf v. Diese wird als *inzident* zu v bezeichnet, wobei v *adjazent* zu u ist. Repräsentiert werden kann ein Graph durch eine sogenannte *Adjazenzmatrix* $A : |V| \times |V|$. Ein Eintrag a_{ij} der Matrix A ist beschrieben durch:

$$a_{ij} = \begin{cases} 1 & \text{falls } (v_i, v_j) \in E \\ 0 & \text{sonst} \end{cases}$$

Die Adjazenzmatrix des Graphen aus Abbildung 14 ergibt sich als:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Wird jeder Kante über eine Funktion $w(u, v) : V \times V \to \mathbb{R}$ ein Gewicht zugeordnet, so spricht man von einem *gewichteten Graphen*. Die Einträge der Adjazenzmatrix A können in einem gewichteten Graphen wie folgt gebildet werden:

$$a_{ij} = \begin{cases} w(v_i, v_j) & \text{falls } (v_i, v_j) \in E \\ \infty & \text{sonst} \end{cases}$$

Der Begriff der *Diffusion* bezeichnet in der Chemie die Vermischung verschiedener, sich in Kontakt befindender Stoffe (Falbe u. Regitz 1997). Ein *Prozess* ist ein Vorgang, bei dem sich über eine gewisse Zeit allmählich etwas herausbildet (Scholze-Stubenrecht 2011). Ein *Diffusionsprozess* beschreibt damit einen Vorgang, bei dem sich mittels Vermischung und Verbreitung im Verlaufe der Zeit etwas herausstellt.

Laut Donoser u. Bischof (2013) werden Diffusionsprozesse durch folgende Kriterien beschrieben:

1. Initialisierung
2. Übergangsmatrix
3. Aktualisierungsregel

Zunächst wird das Ergebnis $p \in \mathbb{R}^n$ des Diffusionsprozesses definiert. Dies könnte beispielsweise die Wahrscheinlichkeit $P_t(v_i)$ sein, nach t Zeitschritten in einem Knoten v_i des Graphen zu sein. Möchte man dies für alle Knoten des Graphen als Ergebnis der Diffusion definieren, könnte man dies durch einen Vector $p_t \in \mathbb{R}^{|V|}$ beschreiben:

$$p_t = (P_t(v_1), P_t(v_2), \cdots, P_t(v_{|V|}))$$

Die *Initialisierung* setzt das Ergebnis der Diffusion auf einen Anfangswert. Für das genannte Beispiel wäre dies, falls als Startknoten der Diffusion v_1 gewählt wird, der folgende Vektor:

$$p_0 = (P_0(v_1), P_0(v_2), \cdots, P_0(v_{|V|})) = (1, 0, \cdots, 0)$$

Durch die *Übergangsmatrix* wird jeder Kante des Graphen ein Wert zugeordnet und somit eine gewichtete Adjazenzmatrix bildet. Bezeichne $deg(v_i)$ die Anzahl ausgehender Kanten von v_i, so kann eine Gewichtsfunktion $w(u, v)$ definiert werden als:

$$w(u, v) = \frac{1}{deg(u)}$$

Ein Eintrag a_{ij} der Übergangsmatrix A ist dann gegeben durch:

$$a_{ij} = \begin{cases} w(v_i, v_j) & \text{falls } (v_i, v_j) \in E \\ 0 & \text{sonst} \end{cases} = \begin{cases} \frac{1}{deg(v_i)} & \text{falls } (v_i, v_j) \in E \\ 0 & \text{sonst} \end{cases}$$

Jeder Eintrag a_{ij} beschreibt somit die Wahrscheinlichkeit innerhalb eines Zeitschritts vom Knoten v_i in den Knoten v_j überzugehen, wenn zufällig zwischen allen möglichen, von v_i ausgehenden Übergängen gewählt wird.
Die *Aktualisierungsregel* $F : \mathbb{R}^n \to \mathbb{R}^n$ beschreibt wie das Ergebnis der Diffusion in einem Zeitschritt p_t auf den nächsten p_{t+1} abgebildet wird:

$$p_{t+1} = F(p_t)$$

In dem gewählten Beispiel könnte F folgendermaßen aussehen, wobei p_t als Zeilen-

vektor angenommen wird:

$$p_{t+1} = F(p_t) = p_t A$$

Der hier beispielhaft dargestellte Diffusionsprozess wird als *Random-Walk* bezeichnet.
Die einzelnen Einträge des Ergebnisses des Diffusionsprozesses und der Übergangsmatrix werden in dieser Arbeit als *Aktivierung* bezeichnet. Aktivierung ist ein Begriff aus der Theorie der neuronalen Netze. Wie in Kriesel (2007) beschrieben, werden in einem Neuron verschiedene Eingänge über eine Funktion zu einem Ausgangswert des Neurons zusammengefasst. Dieser kann wieder als Eingang in das Neuron zurückgehen, was als *Rekurrenz* bezeichnet wird. Dieser Ausgang wird als *Aktivierungszustand* oder auch kurz Aktivierung bezeichnet.

4 ABC-Learning

In diesem Kapitel wird das von Hild entwickelte ABC-Learning vorgestellt (Hild u. Kubisch 2011). Dazu wird zunächst auf die sogenannte *Cognitive Sensorimotor Loop* (CSL) eingegangen und anschließend das darauf aufbauende *Attractor-Based Behavior Control* (ABC) erläutert. Abschließend wird die Selbstexploration eines Roboter mittels des ABC-Learning beschrieben.

4.1 Cognitive Sensorimotor Loop

Die Cognitive Sensorimotor Loop ist eine sensomotorische Schleife, die am Forschungslabor Neurorobotik entwickelt wurde. Abbildung 15 stellt diese schematisch dar.

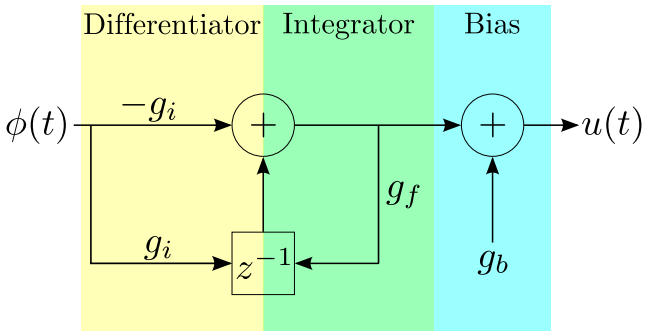


Abbildung 15: Dargestellt ist hier die Cognitive Sensorimotor Loop des Forschungslabors Neurorobotik. Diese eignet sich zur Steuerung eines Motors und kann dabei kognitive Merkmale erkennen lassen. Die Grundelemente, aus dem sie zusammengesetzt ist, sind ein Differentiator, Integrator und Bias

Es erfolgt eine Differentiation des Winkels $\phi(t)$ zu

$$-g_i\dot{\phi}(t) = -g_i\phi(t) + g_i\phi(t-1) = -g_i(\phi(t) - \phi(t-1))$$

mit einer Gewichtung g_i, dem ersten Parameter des CSL. Zur Übersichtlichkeit und zum besseren Verständnis kann dieser Schritt auch vorgelagert sein oder es könnte auch direkt die von einem Sensor gemessene Winkelgeschwindigkeit am Eingang anliegen. Dies führt zu der in Abbildung 16 dargestellten Form des CSL.

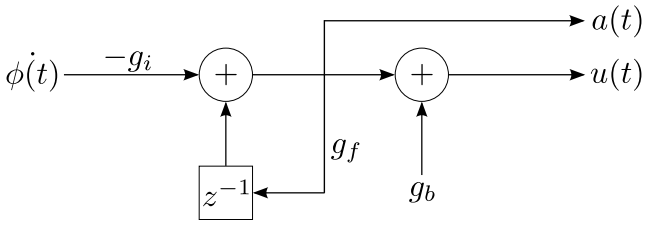


Abbildung 16: Abgebildet ist eine kompaktere Darstellung des CSL. Anstatt eines Winkels bildet die Winkelgeschwindigkeit den sensorischen Eingang. Zusätzlich wurde für die weitere Erklärung eine Zwischenvariable $a(t)$ eingefügt.

Das Ergebnis der Differentiation wird in $a(t)$ integriert und mit g_f gewichtet, dem zweiten Parameter des CSL. Als letzter Schritt wird zu $a(t)$ noch ein Biaswert g_b, der dritte Parameter des CSL, addiert. Daraus folgt, dass der Ansteuerungswert des Motors $u(t)$ definiert ist als:

$$u(t) = -g_i\dot{\phi}(t) + g_f a(t-1) + g_b = -g_i\dot{\phi}(t) + g_f(u(t-1) - g_b) + g_b$$

Damit lässt sich das CSL aus Abbildung 15 unterteilen in einen Differentiator mit dem Parameter g_i, einem Integrator mit dem Parameter g_f und einem Biaswert g_b. Anzumerken ist, dass Winkelgeschwindigkeit und Motorspannung gleichgerichtet sein müssen, sodass bei Vernachlässigung von allen anderen wirkenden Kräften aus einer positiven Motorspannung eine positive Winkelgeschwindigkeit resultiert und vice versa.
Das CSL lässt sich grundsätzlich in vier verschiedenen Modi betreiben, dem Release-, Contraction-, Hold- und Support-Modus. Diese Modi werden mittels der Parameter g_i und g_f eingestellt. Eine Übersicht ist in Tabelle 1 zu finden.

Release	Contraction	Hold	Support
$g_i > 0$	$g_i > 0$	$g_i > 0$	$g_i < 0$
$0 \leq g_f < 1$	$g_f > 1$	$g_f = 1$	$g_f = 0$

Tabelle 1: Übersicht über die verschiedenen CSL-Modi mit den dazugehörigen Parametern

Der erste Modus ist der sogenannte Release-Modus. Hierbei ist $g_i > 0$ und $0 \leq g_f < 1$. Durch den positiven Wert des g_i wird die Winkelgeschwindigkeit negativ auf die Motorspannung gekoppelt. Dies führt dazu, dass der von außen induzierten Bewegung entgegengewirkt wird. Der Integrator wird durch die Wahl des angegebenen Bereichs des g_f zu einem sogenannten *Leaky-Integrator*. Das heißt, dass die Integration den zu integrierenden Wert in jedem Zeitschritt verringert. Ist die Winkelgeschwindigkeit irgendwann Null, so wird die Motorspannung mit der Zeit auch

auf Null geregelt. Das Verhalten des Release-Modus entspricht einer viskosen Reibung. Bewegungen werden, abhängig von der genauen Wahl der Parameter, mehr oder weniger abgebremst. Dieses Verhalten ist Anhand des aus Abschnitt 3.3 bekannten Beispielroboters in Abbildung 17 dargestellt, in der das grüne Segment aus einer instabilen Lage in den stabilen Fixpunkt übergeht.

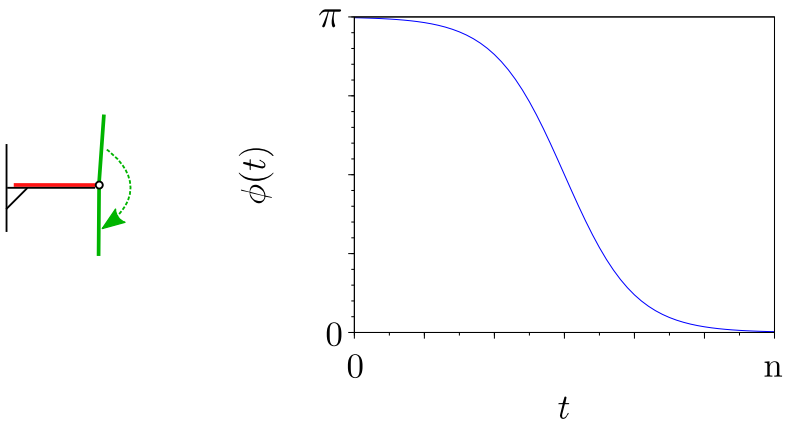


Abbildung 17: Abgebildet ist das Verhalten des Release-Modus am Beispiel des aus Abschnitt 3.3 bekannten Beispielroboters. Das grüne Segment ist nach oben gerichtet und leicht ausgelenkt: $\phi(0) = \pi - \epsilon$. Die Schwerkraft bewegt dieses nach unten, wobei der Release-Modus eine bremsende Wirkung entfaltet. Nach n Zeitschritten kommt das Segment nach unten hängend zum stehen: $\phi(n) = 0$.

Wählt man nun stattdessen den Parameter g_f größer als Eins, erhält man den Contraction-Modus. Dieser funktioniert bis auf den Integrator zunächst ähnlich wie der Release-Modus, d.h. die Winkelgeschwindigkeit wird negativ gewichtet auf die Motorspannung übertragen. Durch ein $g_f > 1$ wird der Integrator zu einem Integrator mit zusätzlichem Feedback. Das bedeutet, dass der zu integrierende Wert immer wieder vergrößert wird. Dies hat zur Folge, dass die Motorspannung irgendwann auf einen Wert anwächst, bei dem das im Motor erzeugte Drehmoment größer als das durch die von außen wirkende Kraft erzeugte Drehmoment wird und sich das Vorzeichen der Winkelgeschwindigkeit somit umkehrt. Daraus folgt wiederum, dass der zu integrierende Wert wieder verringert wird. Bei gut gewählten Parametern führt das zu kontinuierlichen Bewegungen, die sich gegen die von außen wirkende Kraft richten.

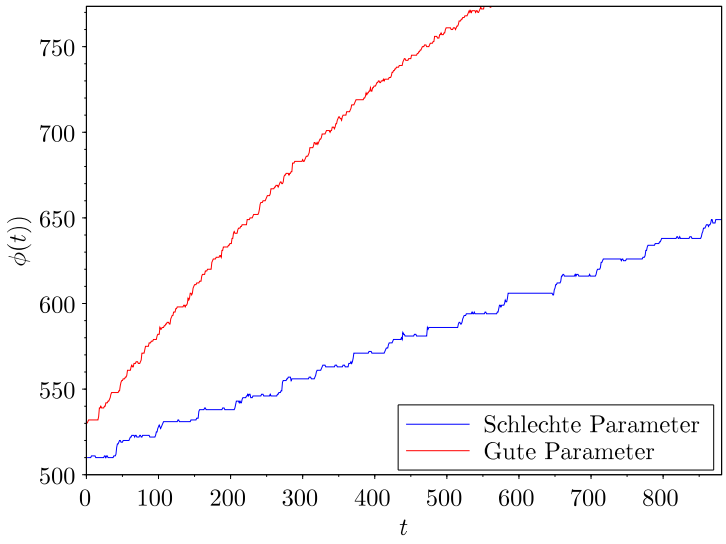


Abbildung 18: Gegenübergestellt sind schlecht gewählte Parameter (blau) und gut gewählten Parameter (rot) für den Contraction-Modus. Beim blauen Graphen ist zu sehen, dass die Bewegung immer wieder zum Stehen kommt, zu erkennen an den waagerechten Abschnitten. Im roten Graphen hingegen ist die Bewegung durchgängig.

Wie in Abbildung 18 zu sehen ist, tritt bei schlecht gewählten Parametern ein Stick-Slip-ähnlicher Effekt auf, bei dem sich folgende Phasen abwechseln. Zunächst kommt die Bewegung zum Stehen und der Integrator lädt sich auf Grund der hohen Haftreibung auf. Dann wird diese überwunden und es kommt zu einer plötzlichen starken Beschleunigung, da die Gleitreibung wesentlich kleiner als die Haftreibung ist. Durch die Erhöhung der Winkelgeschwindigkeit bremst das CSL wieder ab, kommt zum Stehen und baut wieder Kraft auf, um die Haftreibung zu überwinden und so weiter. Auch andere Effekte wie Überschwingen sind oft auf nicht perfekt gewählte Parameter zurückzuführen. In Werner (2013) wurde untersucht wie beim mehrmaligen ausführen einer Bewegung Schritt für Schritt die Parameter angepasst werden können.
Die Funktionsweise des Contraction-Modus soll nun wieder an dem Bekannten Beispielroboter demonstriert werden.

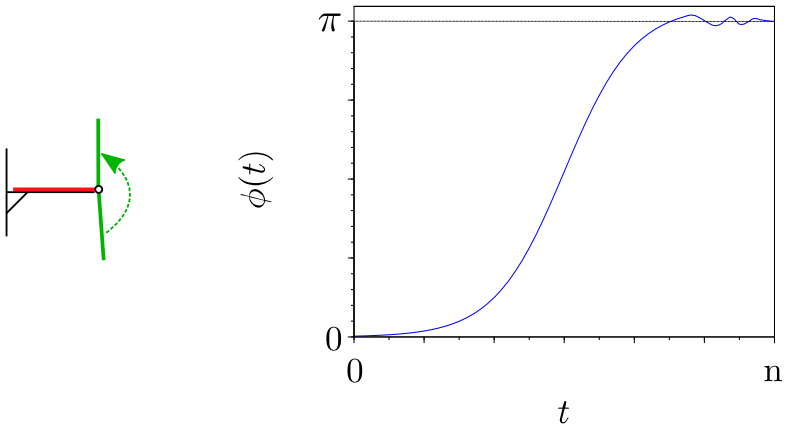


Abbildung 19: Abgebildet ist die Funktionsweise des CSL im Contraction-Modus. Das grüne Segment ist zu Beginn leicht ausgelenkt: $\phi(0) = 0 + \epsilon$. Das CSL fängt an sich gegen die Schwerkraft zu richten. Wenn das Segment den Scheitelpunkt $\phi(t) = \pi$ überschreitet, ändert sich irgendwann die Bewegungsrichtung, da die Schwerkraft das Segment nun in die andere Richtung drückt. Es kann so zu einer dauerhaften Oszillation oder einer abklingenden Schwingung kommen.

In Abbildung 19 ist der Bewegungsablauf des Beispielroboters, der mit einem CSL in Contraction-Modus geregelt wird, dargestellt. Ein Roboter dieser Art und in dieser Lage hat auf Grund der Reibung ohne Regelung einen stabilen Fixpunkt in dem Punkt, wo das grüne Segment nach unten hängt, in Abbildung 19 entspricht dies einer Auslenkung um 0°. Außerdem gibt es einen instabilen Fixpunkt bei 180°. Zunächst wird davon ausgegangen, dass das grüne Segment entweder leicht ausgelenkt ist, wie in Abbildung 19 zu sehen, oder zumindest durch Sensorrauschen eine kleine „Bewegung“ wahrgenommen wird. Wie beschrieben fängt das CSL an, sich gegen die anliegende Kraft, hier die Erdgravitation g, zu richten. Dies führt zu einer Aufwärtsbewegung des Segments. Bei gut gewählten Parametern ist die Winkelgeschwindigkeit v in dieser Phase im Verhältnis zum Integrator zu klein, um das Vorzeichen des Ausgangs des CSL umzukehren. Sobald das Segment sich um 90° gedreht hat und sich dem vormals instabilen Fixpunkt nähert, nimmt die Wirkung F_R von g auf das Drehmoment immer weiter ab, da diese folgendermaßen definiert ist:

$$F_R = mg \sin(\phi(t))$$

Dies führt dazu, dass v im Verhältnis zum Integrator wächst. Im Idealfall sind die Parameter so gut gewählt, dass bei Erreichen des ehemals instabilen Fixpunktes die Winkelgeschwindigkeit und der Integrator sich egalisiert haben und das Segment zum Stehen kommt. Ansonsten kann es wie in Abbildung 19 rechts dargestellt, dadurch zu einem leichten Überschwingen kommen. Dies führt dazu, dass nach dem

Überschreiten des vormals instabilen Fixpunkts, die Erdgravitation zusätzlich beschleunigt und v so groß wird, dass das CSL die Richtung ändert und sich mit weniger Geschwindigkeit wieder in Richtung dieses Fixpunkts bewegt. Entweder entsteht dann eine leichte Oszillation um den ehemals instabilen Fixpunkt oder die Bewegung kommt nach ein paar Durchläufen in diesem zum Stehen.
Anhand des beschriebenen Verhaltens kann man erkennen, dass der Contraction-Modus den ursprünglich stabilen Fixpunkt destabilisiert und den ehemals instabilen Fixpunkt stabilisiert. Somit lassen sich mit Hilfe des Release-Modus stabile Fixpunkte und mit dem Contraction-Modus instabile Fixpunkte des Ausgangssystems ohne Regelung einregeln. Im weiteren Verlauf der Arbeit werden, insofern nicht explizit anders angegeben, ausgehend vom ungeregelten System, Fixpunkte als stabil oder instabil bezeichnet, auch wenn diese in dem neuen System inklusive der Regelung ihre anziehende oder abstoßende Wirkung verlieren. Die Idee, durch das abwechselnde Umschalten von Release- und Contraction-Modus verschiedene stabile und instabile Fixpunkte zu erreichen, ist die Grundlage des in Kapitel 4 beschriebenen ABC-Learning.

Die letzten beiden CSL-Modi spielen eine eher untergeordnete Rolle für die in dieser Arbeit relevanten Verfahren und werden daher der Vollständigkeit halber nur kurz erläutert. Der Hold-Modus, der zumindest in abgewandelter Form auch in dieser Arbeit eingesetzt wird, sorgt für das Halten der aktuellen Position. Dabei wird abhängig von der genauen Wahl der Parameter die Position komplett steif oder aber federnd gehalten.
Der letzte Modus, der Support-Modus, wird in dieser Arbeit überhaupt nicht eingesetzt und wird in anderen Projekten genutzt, um von außen intendierte Bewegungen zu unterstützen.

4.2 Symmetriebrechung

Schaltet man in einem stabilen Fixpunkt in den Contraction-Modus oder in einem instabilen Fixpunkt in den Release-Modus, so ist es nicht eindeutig, in welche Richtung sich das Gelenk bewegt. Verdeutlicht wird dies in Abbildung 20. Hängt das grüne Segment nach unten und wird in den Contraction-Modus geschaltet, so kann es mit oder gegen den Uhrzeigersinn anfangen zu drehen (rechts). Wird das Segment durch den Contraction-Modus im instabilen Fixpunkt stabilisiert und dann in den Release-Modus geschaltet, kann es ebenso mit oder gegen den Uhrzeigersinn nach unten fallen (links).

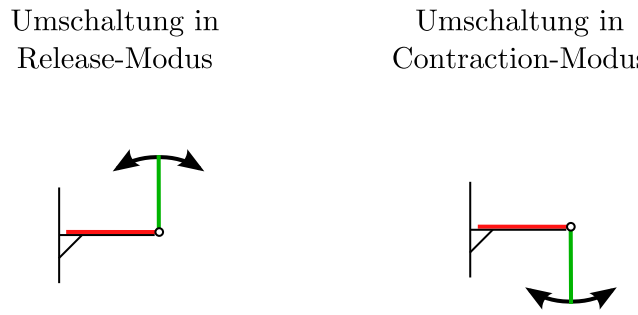


Abbildung 20: Abgebildet ist hier das Problem der Uneindeutigkeit beim Umschalten der CSL-Modi. Beim Umschalten in einen anderen Modus gibt es jeweils zwei Richtungen in die sich das grüne Sement bewegen kann. Für ein gezielte Umschaltung in eine Richtung bedarf es deshalb einer Symmetriebrechung.

Für ein gezieltes Umschalten in eine Richtung ist somit eine sogenannte *Symmetriebrechung* notwendig. Nach von Lüde u. a. (2009) bildet diese einen zentralen Mechanismus zur Ordnungsbildung. Sie beschreibt den Übergang von einem symmetrischen Zustand in einen Zustand, in dem die Symmetrie aufgehoben wurde. Hierzu wird zumeist ein Parameter des zugrundeliegenden nichtlinearen Systems geändert, wobei oft verschiedene Alternativen möglich sind, die Symmetrie zu brechen. Beim Umschlaten des CSL-Modus kann z. B. dessen Bias dazu genutzt werden, die Symmetrie des Systems aufzuheben und somit das Gelenk gezielt in eine Richtung zu bewegen. Für die in dieser Arbeit verwendeten Drehgelenke gibt es jeweils zwei Richtungen, in die die Symmetrie gebrochen werden kann.

4.3 Attractor-Based Behavior Control

In der Natur lässt sich häufig das Muster von Anspannung (Contraction) und Entspannung (Release) finden, bei dem durch gegenseitiges Abwechseln dieser Phasen ein bestimmtes Ziel erreicht werden soll. Beispiele hierfür sind die Atmung des Menschen (Larsen u. Ziegenfuß 2004) und die Fortbewegung der Qualle (Gemmell u. a. 2013), welche zum großen Teil durch das Abwechseln von Kontraktions- und Entspannungsphasen realisiert werden, wie in Abbildung 21 dargestellt.

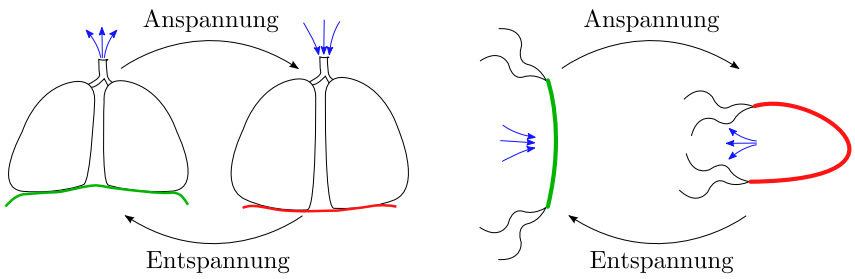


Abbildung 21: Dargestellt sind zwei Beispiele der Natur, in denen das Prinzip von An- und Entspannung genutzt wird. Links die Atmung des Menschen und rechts die Fortbewegung einer Qualle skizziert.

Anhand der Verschiedenartigkeit der zwei Beispiele zeigt sich, wie universell das Prinzip ist. Gemein ist ihnen, dass es jeweils Phasen gibt (Entspannungsphasen), in denen wenig bis keine Energie gebraucht wird. Der Grund hierfür ist, dass sich das System passiv, in diesen Fällen zum Großteil auf Grund von Elastizitäten, wieder in seinen vorherigen Zustand bewegt. Mit der bereits eingeführten Theorie der dynamischen Systeme lässt sich somit sagen, dass die Kontraktion das System aus seinem stabilen Fixpunkt x^* hinaus in einen ehemals instabilen Punkt bewegt, jedoch im Basin von x^* bezüglich des entspannten Systems verharrt, sodass beim Entspannen x^* wieder seine anziehende Wirkung entfaltet und das System sich diesem Fixpunkt wieder nähert.
Das Attractor-Based Behavior Control realisiert diese Phasen mithilfe des CSL und seinen Contraction und Release genannten Modi. In dieser Arbeit wird ein Roboter mit Rotationsgelenken betrachtet auf den als einzige Kraft, die Schwerkraft der Erde einwirkt. Die Kraft die an diesen Fixpunkten auf das Gelenk des Roboters einwirkt lässt sich mit der bereits bekannten Formel für F_R beschreiben:

$$F_R = mg\sin(\phi(t))$$

Das Halten eines instabilen Punktes erfordert keine Energie, da die wirkende Kraft der Gravitation Null ist:

$$F_R = mg\sin(\pi) = 0$$

Muss dieser Punkt auf Grund kleiner Störungen immer wieder stabilisiert werden, wird nur wenig Energie benötigt, da F_R einen Wert nahe der Null hat. Das Halten eines stabilen Fixpunktes im Release-Modus benötigt ebenfalls keine Energie, da auch hier die wirkende Kraft Null ist:

$$F_R = mg\sin(0) = 0$$

Bei kleinen Störungen wird ebenfalls keine Energie benötigt, da die Schwerkraft diese wieder ausgleicht. Es wird hauptsächlich dann Energie verbraucht, wenn von einem stabilen Fixpunkt in einen instabilen gewechselt wird. Unter Umständen wird auch Energie für den Weg vom instabilen in den stabilen Fixpunkt gebraucht, um die Bewegungen zu dämpfen und unerwünschte Effekte wie Beschädigungen des Roboters oder Überschwingen zu vermeiden. Für ein autonomes System kann es daher erstrebenswert sein, diese Fixpunkte anzusteuern, um so Energie zu sparen. Im Release-Modus wird das System durch die Schwerkraft in den stabilen Fixpunkt bewegt. Im Contraction-Modus sind die Stabilitäten der Fixpunkte vertauscht, wie es in Abbildung 22 dargestellt ist.

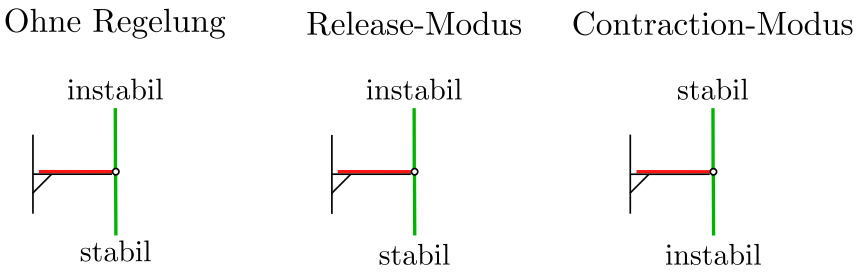


Abbildung 22: Abgebildet sind die Stabilitäten der Fixpunkte ohne Regelung, mit Release-Modus und Contraction-Modus. Während die Stabilitäten der Fixpunkte ohne Regelung und mit Release-Modus gleich sind, sind sie im Contraction-Modus umgekehrt.

Solange sich das System nicht genau im instabilen Fixpunkt befindet, wird es den stabilen Fixpunkt anstreben, da dieser ein Attraktor ist. Das Verhalten des System wird somit über die Attraktoren bestimmt. Deswegen wird dieses Verfahren Attractor-Based Behavior Control genannt. Der Vorteil ist, dass kein Modell des Roboters oder der Welt benötigt wird, um die Attraktoren anzusteuern, da sich das System wegen ihrer anziehenden Wirkung „wie von selbst“ zu diesen hinbewegt. Aufbauend auf dem Attractor-Based Behavior Control wurde das ABC-Learning zur modellfreien Selbstexploration autonomer Roboter entwickelt.

4.4 Selbstexploration mittels ABC-Learning

Das ABC-Learning ist ein Lernverfahren zur Selbstexploration eines autonomen Roboters. Es verwendet das bereits vorgestellte Attractor-Based Behavior Control. Beschrieben wird dieses in Hild u. Kubisch (2011) und in Bethge (2014). Ziel dieses Verfahrens ist es, den Zustandsraum des Roboters zu explorieren. Wie im Abschnitt 3.1 beschrieben, bildet dieser eine Mannigfaltigkeit in $\mathbb{R}^n$. Für den Semni ist diese beispielhaft in Abbildung 23 dargestellt. Die Mannigfaltigkeit bildet den Zustandsraum des Semni, wenn er sich frei und auf ebenem Boden liegend bewegen kann.

Zustände, die viel Halteenergie benötigen, sind rot eingefärbt. Je weniger Energie benötigt wird, desto grüner ist die Einfärbung. Ist ein Zustand weiß gefärbt, benötigt diese fast gar keine Energie.

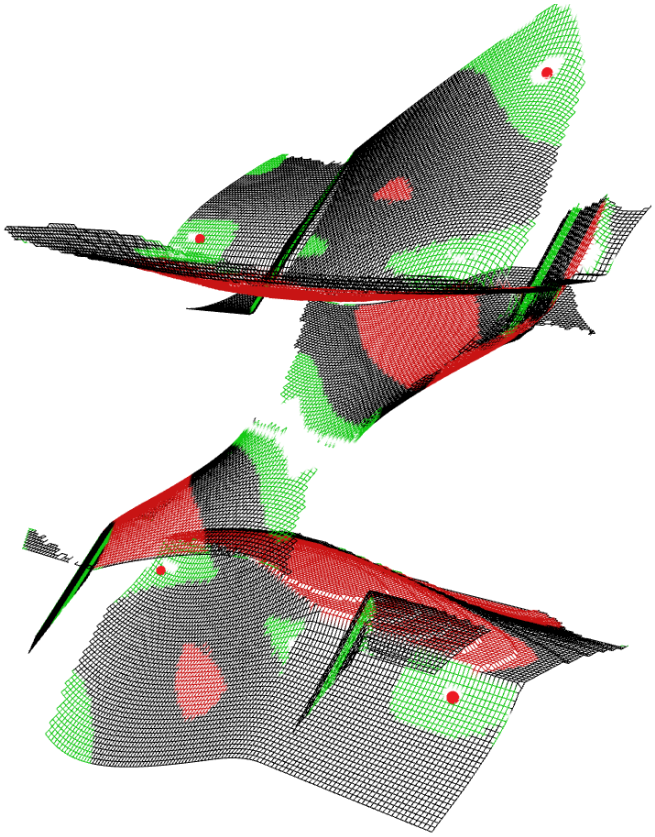


Abbildung 23: Dargestellt ist hier die Mannigfaltigkeit des Semni. Diese entsteht, wenn er sich auf dem Boden liegend frei bewegen kann. Die roten Kugeln markieren beispielhaft Postures, die durch das ABC-Learning bestimmt wurden.

Ein Punkt $x(t)$ im Zustandsraum des Semni wird, wie im Abschnitt 3.3 beschrieben, aus dessen Lage und seinen Gelenkwinkeln gebildet:

$$x(t) = (\phi_{\text{Hüfte}}, \phi_{\text{Knie}}, \phi_{\text{Körper}})$$

Die Bezeichnungen der Gelenke und sonstiger Teile des Semni sind in Abbildung 24 dargestellt.

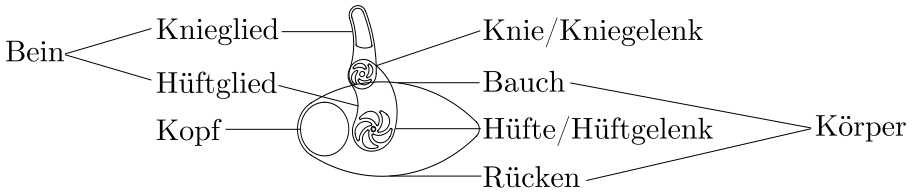


Abbildung 24: Dargestellt sind hier die Bezeichnungen der einzelnen Teile des Semni, wie sie in dieser Arbeit verwendet werden.

Zur Exploration wird der Roboter mittels des Contraction- und Release-Modus des CSL von einem Fixpunkt in den Nächsten gebracht. Dies geschieht dadurch, dass nach Erreichen eines Fixpunktes in einem Gelenk der CSL-Modus von Release zu Contraction bzw. von Contraction zu Release umgeschaltet wird. Dieses eine Gelenk wird dadurch destabilisiert, wobei alle anderen Gelenke aktiv oder passiv stabilisiert werden. Die Destabilisierung des aktuellen Gelenkwinkels führt dazu, dass sich das Gelenk in Richtung seines neuen stabilen Fixpunktes bewegt, während die anderen Gelenke folgen, wie in Abbildung 25 dargestellt.

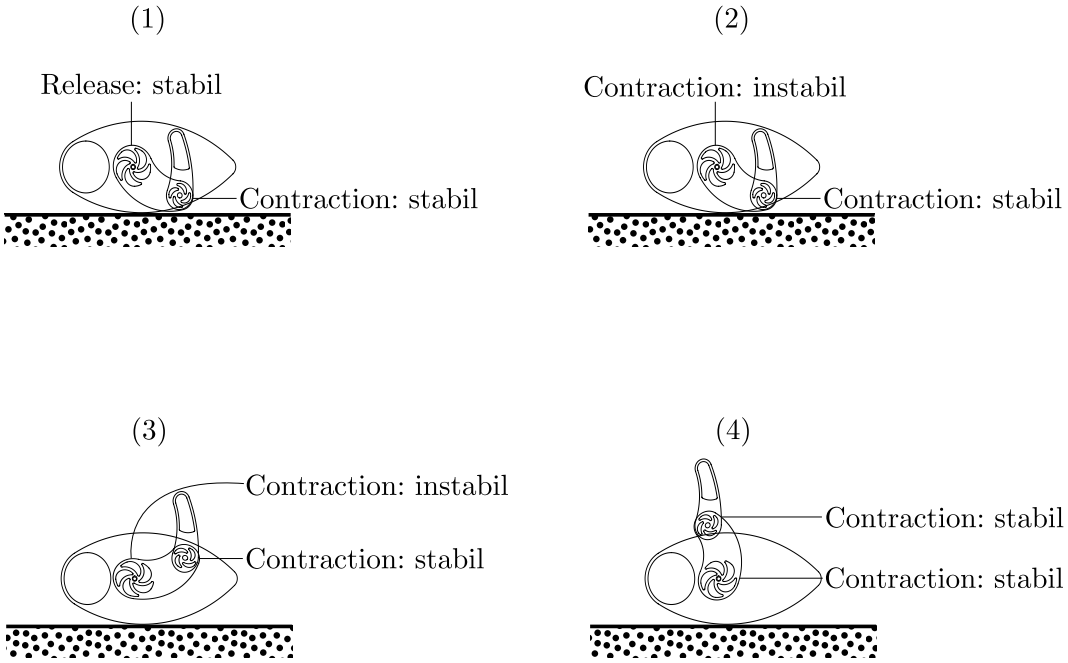


Abbildung 25: Abgebildet ist ein Bewegungsablauf des Semni nach dem Umschalten des CSL-Modus in einem Gelenk. Während ein Gelenk destabilisiert wird, gleicht das andere die Bewegungen dahingehend aus, dass es selbst stabil bleibt.

So hangelt sich das System von Fixpunkt zu Fixpunkt. Gelernt werden hierbei die sogenannten *Postures* in jedem Fixpunkt und ihre Verbindungen untereinan-

der. Eine Posture besteht beim Semni aus den zwei Gelenkwinkeln, der Ausrichtung des Körpers relativ zur Schwerkraft und den in jedem Gelenk aktuell eingeschalteten CSL-Modi. Eine Posture repräsentiert somit einen Punkt auf der Mannigfaltigkeit, wobei im weiteren Verlauf der Arbeit nur solche als Postures bezeichnet werden, die sich in einem stabilen Fixpunkt bezüglich des dynamischen Systems bestehend aus Semni inklusive Regelung und der Umwelt mit der Übergangsfunktion $F(x(t)) : \mathbb{R}^3 \to \mathbb{R}^3$ befinden. Zusätzlich wird eine Posture durch die verwendeten CSL-Modi beschrieben. Somit ist eine Posture p_i definiert als:

$$p_i = (x(t), \mathrm{CSL}_{\mathrm{Hüfte}}, \mathrm{CSL}_{\mathrm{Knie}}),\ x(t) \text{ ist stabiler Fixpunkt bezüglich } F(x(t))$$

Eine Verbindung zwischen zwei Postures bedeutet, dass es möglich ist, mittels Umschaltung des CSL-Modus in einem Gelenk, von der einen Posture in die andere zu gelangen, was als Relation R zwischen zwei Postures formalisiert werden kann. Die Verbindungen sind gerichtet, da es, wie in Abschnitt 3.3 beschrieben, nicht immer möglich ist, wieder auf direktem Weg zurück in die vorherige Posture zu kommen. Es entsteht somit ein gerichteter Graph $G = (P, E)$, der die verschiedenen Fixpunkte über die CSL-Modi verbindet:

$$P = \bigcup \{p_i\}$$

$$E = \{(p_i, p_j) \in P \times P \ : \ p_i R p_j\}$$

Dieser Graph entsteht auf Basis von sensomotorischen Regelschleifen und bildet eine Repräsentation für den Verlauf der sensomotorischen Aktionen. Außerdem wird er mit verschiedenen Sensor- und Motorwerten angereichert und kann deshalb als *sensomotorischer Graph* bezeichnet werden. Im Folgenden werden Postures mit der ID i mit p_i und Verbindungen zwischen zwei Postures mit den IDs i und j als e_{ij} bezeichnet, wobei die Kante e_{ij} von p_i aus zu p_j geht. Ein Ausschnitt eines solchen Graphen ist in Abbildung 26 abgebildet.

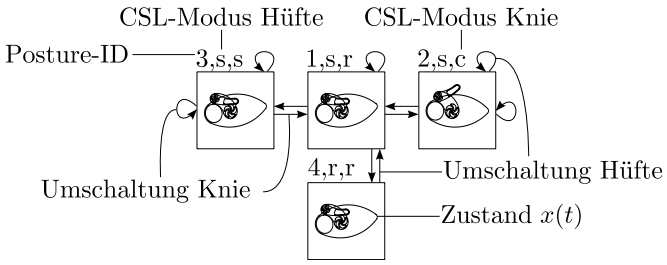


Abbildung 26: Dargestellt ist ein Ausschnitt eines Graphen, wie er vom ABC-Learning gebildet wird. Die Knotenmenge bilden die Postures. Eine Kante zwischen zwei Postures bedeutet, dass von entlang der Pfeilrichtung durch Umschalten des CSL-Modus in einem Gelenk, von der einen in die Andere Posture gewechselt werden kann. Die drei CSL-Modi, die verwendet werden sind: Contraction-Modus (c), Release-Modus (r) und der sogenannte Stall-Modus (s), der im Folgenden erklärt wird.

In Abbildung 23 sind beispielhaft Postures in der Mannigfaltigkeit des Semni eingezeichnet. Die Energieeffizienz des Attractor-Based Behavior Control lässt sich hier bereits erkennen, da alle Postures, markiert durch eine rote Kugel, in einem weißen Bereich liegen und somit wenig Energie benötigen. Bei der Kontraktion eines Gelenks kann es dazu kommen, dass der Roboter entweder mit sich selbst oder einem Hindernis in der Umwelt kollidiert. Beispielsweise kollidiert das Hüftglied in Abbildung 26 in den Postures p_1 bis p_3 mit dem Kopf. Wird durch diese Kollision die Bewegung des Roboters gestoppt, würde sich der Integrator immer weiter aufladen und es könnte zu einer Beschädigung kommen. Deswegen ist es wichtig auch diese Punkte zu erkennen, abzuspeichern und in einen sogenannten *Stall*[3] Modus umzuschalten. In diesem Modus wird der Winkel des Gelenks einfach mit einem konstanten Drehmoment gehalten. Diese Gelenkwinkel sind somit nicht unbedingt energieeffizient, bilden aber trotzdem eine Posture und werden in den Graphen mit aufgenommen.
Für jedes Gelenk kann immer nur in einen anderen Modus umgeschaltet werden. Außerdem gibt es, wie in 4.2 beschrieben, jeweils zwei Richtungen, in die dies geschehen kann. Beim Semni gibt es somit bei Erreichen einer Posture vier Möglichkeiten zu explorieren, da es für zwei Gelenke je zwei Richtungen gibt, in die umgeschaltet werden kann, wie in Abbildung 27 gezeigt.

[3]Englisch für „steckenbleiben“, „blockieren“, „festfahren“

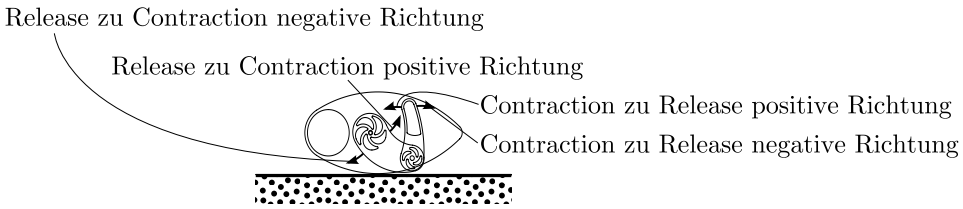


Abbildung 27: Abgebildet sind die möglichen Umschaltungen in einer Posture. Hüfte und Knie können jeweils in zwei Richtungen umgeschaltet werden. Positive Richtung meint dabei gegen den Uhrzeigersinn und negative Richtung entsprechend andersherum. Insgesamt ergeben sich beim Semni in jeder Posture vier Möglichkeiten umzuschalten.

Eine Untersuchung zu verschiedenen Heuristiken beim Umschalten der Modi gibt es in Bethge (2014). So ist es z. B. möglich, zufällig die nächste Aktion zu wählen oder aber auch immer zunächst mit dem gleichen Gelenk in die selbe Richtung so weit wie möglich zu explorieren. Dies hat u.a. Auswirkungen auf die Dauer der vollständigen Exploration oder welche Untermannigfaltigkeiten zuerst exploriert werden.
Aufbauend auf diesem Lernverfahren und dem dadurch gelernten Graphen wird in dieser Arbeit eine Situationserkennung realisiert.

5 Versuchsaufbau für Experimente mit dem ABC-Learning

5.1 Grundlegender Aufbau

Da das Aufspielen eines Programms langwierig und das Aufspüren von Fehlern im Programmcode am Semni selbst kompliziert ist, sowie die Visualisierung technisch bedingt auf einem externen Rechner stattfinden muss, wurde entschieden, auf dem Semni ein einfaches Programm zum Lesen und Schreiben von Sensor- und Motorwerten auszuführen und alles Weitere auf einen externen Rechner auszulagern.

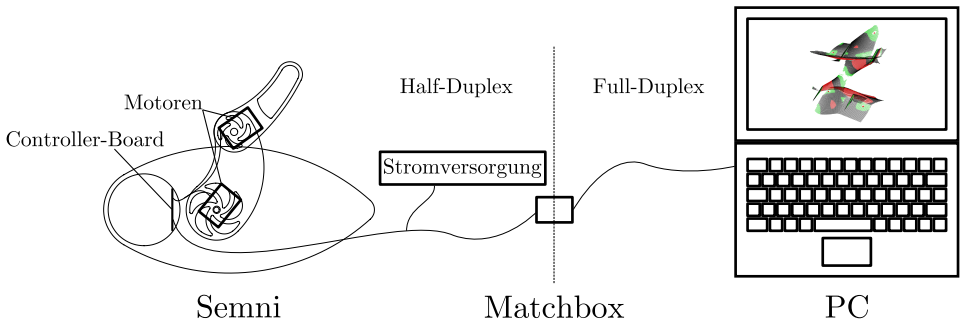


Abbildung 28: Gezeigt ist der schematische Aufbau für das ABC-Learning. Ein PC ist über USB mit einer sogenannten *Matchbox* verbunden, welche den Semni und PC galvanisch entkoppelt und USB zu RS-485 konvertiert. Der Semni ist über RS-485 ebenfalls mit der Matchbox verbunden. Auf diesem Weg können der Semni und PC miteinander kommunizieren.

5.2 Hardware

5.2.1 Semni

Der Roboter Semni wurde am Forschungslabor Neurorobotik entwickelt und gefertigt. Eine schematische Darstellung ist in Abbildung 29 zu finden. Er besteht aus einem elliptischen Körper und einer geschwungenen Gliedmaße mit zwei Gelenken. Die Form beschränkt seine Bewegungen auf zwei Dimensionen im Raum und ermöglicht ihm, weitgehend von jedem Zustand in jeden anderen zu gelangen. Letzteres ist für Lern- und Explorationsverfahren besonders wichtig, da es sonst passieren könnte, dass das Verfahren in eine Position steuert, aus der es selbstständig nicht wieder heraus kommt. Dies hätte zur Folge, dass diese Verfahren unter Umständen immer wieder unterbrochen werden müssten und der ständigen Aufmerksamkeit eines Menschen bedürften.

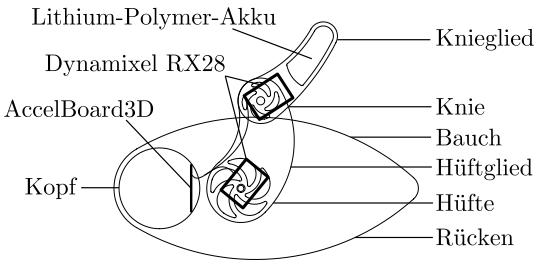


Abbildung 29: Abgebildet ist eine schematische Darstellung des Semni. Wichtige Komponenten wie das Controller-Board, Motoren und Akku sind extra ausgewiesen. Außerdem sind nochmal die Bezeichnungen der Körperteile aufgeführt.

Das im Kopf befindliche AccelBoard3D ist eine im Forschungslabor Neurorobotik entwickelte Platine, die in vielen verschiedenen Projekten des Labors wie z. B. dem humanoiden Roboter Myon zum Einsatz kommt. Diese ist bestückt mit einem ARM Cortex-M3-Prozessor und verschiedener Sensorik wie z. B. Beschleunigungssensoren. Des Weiteren sind verschiedene Schnittstellen verbaut, die die Kommunikation mit einem PC, der Sensorik oder Motorik erlauben. Im Semni wurde je ein Motor im Hüft- und Kniegelenk verbaut. Verwendet wurden hierbei zwei Dynamixel-RX28. Da auch diese in anderen Projekten wie dem Myon verbaut sind, lassen sich sensomotorische Schleifen und Lernverfahren, die sich auf dem Semni bewährt haben, relativ einfach in andere Projekte migrieren. Weitere Details zum AccelBoard3D und Dynamixel-RX28 finden sich in Thiele (2014).

Programme für den Semni lassen sich in C schreiben und dann auf das AccelBoard3D flashen. Es existiert eine Firmware, die alle grundlegenden Funktionen, wie beispielsweise das Lesen von Sensoren oder das Setzen von Motorspannungen übernimmt. Außerdem gibt es einen Bootloader, der über die serielle Schnittstelle kommuniziert. Dieser ermittelt automatisch die Baudrate und bietet die Möglichkeit Sensorwerte zu lesen, Motorwerte zu schreiben und Programme zu starten. Diese Programme werden ebenfalls in C geschrieben und können über den Bootloader in einen Programmspeicher geladen werden. Es kann immer nur ein Programm in den Speicher geladen und anschließend ausgeführt werden, wobei dieses mehrere Unterprogramme enthalten und verwalten kann.

5.2.2 Externer PC

Als externer PC werden verschiedene Laptops mit Windows-Betriebssystem genutzt. Ausschlaggebender Faktor für eine Umsetzung auf Windows ist, dass der Treiber für die Matchbox ohne Anpassung nur unter Windows verfügbar ist. Die einzige weitere Einschränkung für den zu verwendenden PC ist, dass dieser einen USB-Anschluss

besitzt. Es bedarf keiner besonders leistungsfähigen CPU oder ähnlichem. Die im Rahmen dieser Arbeit entwickelte Software lief sowohl auf neuen wie auch auf älteren Rechnern.

5.3 Software

5.3.1 Programm zum Lesen der Sensorwerte und Setzen der Motorwerte auf dem Semni

Der Semni führt im 100-Hertz-Rhythmus folgende zwei Aktionen aus: Zuerst werden die Sensorwerte Strom, Winkel, Temperaturen der beiden Motoren, sowie die Werte des Beschleunigungssensors und die Spannung des AccelBoard3D abgefragt und zusammen mit einer Checksumme über die USART-Schnittstelle versendet. Anschließend wird fünf Millisekunden auf ein Datenpaket an der USART-Schnittstelle gewartet. Trifft ein Paket ein und ist die Checksumme korrekt, werden die in dem Paket enthaltenen Werte für die Motoren gesetzt.

5.3.2 Programm zur Umsetzung des ABC-Learning für den PC

Auf dem Rechner läuft ein Programm, welches in Purebasic[4], einem Basic-Dialekt, geschrieben wurde. Ein bereits bestehendes Programm, das sowohl eine erste Kommunikation zum Lesen der Sensorwerte als auch eine Visualisierung einiger dieser Werte enthielt, diente als Grundlage. Dieses wurde um die Möglichkeit des Schreibens von Motorwerten und weiterer Visualisierungen wie beispielsweise der Motor- und Sensorwerte erweitert. Darauf aufbauend konnten die weiteren Implementierungen des ABC-Learning und der Situationserkennung entstehen, die in den nächsten Kapiteln beschrieben sind.

[4] http://www.purebasic.com (21.11.2014)

6 Implementierung des ABC-Learning

Zu Beginn der Erstellung dieser Arbeit lag keine vollständige Implementierung des ABC-Frameworks vor. Deshalb wurde als erster Schritt das ABC-Learning als Grundlage für die Situationserkennung implementiert. In diesem Abschnitt wird auf die größten Schwierigkeiten, deren Lösungen, verwendete Heuristiken und allgemeine Informationen zu der Implementierung eingegangen. Bei den verschiedenen Entscheidungen für die jeweiligen Verfahren und Wahl der Parameter, wurde bereits versucht, eine spätere Umsetzung der Situationserkennung zu vereinfachen, da es zu diesem Zeitpunkt bereits erste Ideen gab, wie man diese umsetzen könnte.

6.1 Realisierung der Symmetriebrechung

Die in Abschnitt 4.2 beschriebene Symmetriebrechung ließ sich in der praktischen Anwendung nicht mittels eines Biaswertes erreichen. Dies hatte verschiedene Gründe.

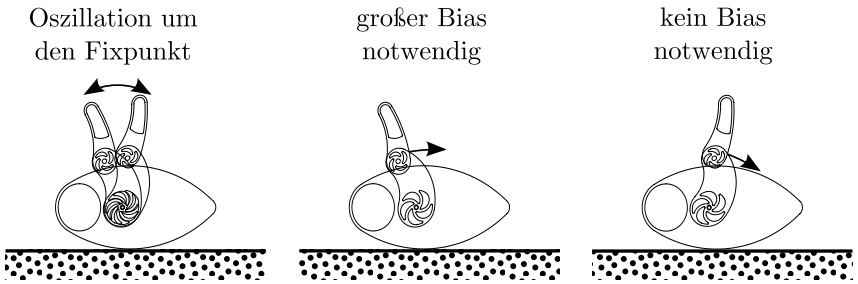


Abbildung 30: Dargestellt ist hier ein Problem der Symmetriebrechung. Bei oszillierenden Bewegungen um den Fixpunkt, kann es passieren, dass in unterschiedlichen Positionen umgeschaltet wird. Muss die Bewegung erst am Fixpunkt vorbei, wäre ein großer Bias nötig. Ist das Bein bereits vom Fixpunkt aus gesehen in die gewünschte Richtung ausgelenkt, ist eventuell gar kein Bias mehr nötig.

Zunächst bedurfte es für verschiedene Postures verschiedene Biaswerte um die gewünschte Symmetriebrechung zu realisieren. Bei einer entsprechend großen Anzahl an Postures, war es einfach unhandlich für jede einzeln die Biaswerte zu bestimmen. Hinzukommend führt das verwenden von Biaswerten zu dem Problem, dass sie eventuell das Ergebnis der folgenden Posture verfälschen, da sich das Gelenk auf Grund des Bias zu weit drehen könnte. Darüber hinaus ist die Startposition in einer Posture beim Umschalten in einen anderen Modus nicht immer gleich. Gerade bei stabilisierten Postures, mit einem Gelenk im Contraction-Modus, kann es, wie beschrieben, zu leichten Oszillationen um den Fixpunkt herum kommen. Dies führt dazu, dass der Bias manchmal so groß gewählt sein muss, um das Gelenk ein kleines Stück gegen die Schwerkraft über den Fixpunkt hinaus zu bewegen und manchmal gar

kein Bias nötig ist, weil die Schwerkraft bereits in die richtige Richtung wirkt. Das Problem wird in Abbildung 30 verdeutlicht. Auf Grund dieser Problematik wurde der Parameter g_b des CSL nicht genutzt, d.h. auf 0 gesetzt.

6.1.1 Verfahren zur Symmetriebrechung beim Umschalten von Contraction nach Release

Gelöst wurde das Problem für das Umschalten vom Contraction-Modus in den Release-Modus, indem zunächst der sogenannte Constant-Velocity-Modus eingeschaltet wurde. Dieser sorgt dafür, dass das Gelenk mit einer bestimmten Geschwindigkeit in eine Richtung gedreht wird. Realisiert wird dieser Modus mit einem etwas erweiterten CSL, das sich folgendermaßen beschreiben lässt:

$$u(t) = v_{target} - g_i\dot{\phi}(t) + u(t-1)$$

Der Parameter g_i muss dabei größer als Null sein, während g_f auf Eins gesetzt wird. Als neuer Parameter wird für diesen Modus v_{target} eingeführt, die Zielgeschwindigkeit. Dieser wird auf das Ergebnis aus $-g_i\dot{\phi}(t)$ addiert, wodurch der Soll-Ist-Wert-Vergleich geschieht. Durch die Wahl von $g_f = 1$ wird die Differenz aus Zielgeschwindigkeit und gewichteter Winkelgeschwindigkeit über die Zeit aufsummiert. Es handelt sich somit um einen PI-Regler, zum Regeln einer Zielgeschwindigkeit. Solange die Winkelgeschwindigkeit kleiner als die Zielgeschwindigkeit ist, wächst der Ansteuerungswert des Motors an und die Winkelgeschwindigkeit nimmt zu. Ist die Winkelgeschwindigkeit größer als die Zielgeschwindigkeit wird der Ansteuerungswert verringert und die Winkelgeschwindigkeit nimmt ab. Dieser Modus ist geeignet, aus beliebigen Positionen das Gelenk in beide Richtungen zu bewegen. Des Weiteren sind die einzelnen Übergänge bei jeder Wiederholung immer sehr ähnlich und reproduzierbar. Diese Eigenschaft ist auch für die Situationserkennung im folgenden Kapitel von Bedeutung.

Nachdem ein Modus zum gezielten Bewegen des Gelenks in eine Richtung eingeschaltet wurde, muss detektiert werden, wann dieser Modus wieder beendet werden soll. Ziel ist es, vom instabilen Fixpunkt in den stabilen überzugehen. Idealerweise soll mit Hilfe des Constant-Velocity-Modus das Gelenk bis zum stabilen Fixpunkt bewegt und dann dort verharrt werden. Um zu erkennen, wann dieser Punkt erreicht ist, wird der Motorausgang im Zeitverlauf betrachtet. In Abbildung 31 sind die durch einen Tiefpass gefilterten Ansteuerungswerte der Motoren für beide Gelenke zu sehen. Bei dieser Aufnahme wurden die Gelenke jeweils im Constant-Velocity-Modus

betrieben und von einem instabilen Fixpunkt aus über den stabilen Fixpunkt hinaus verfahren.

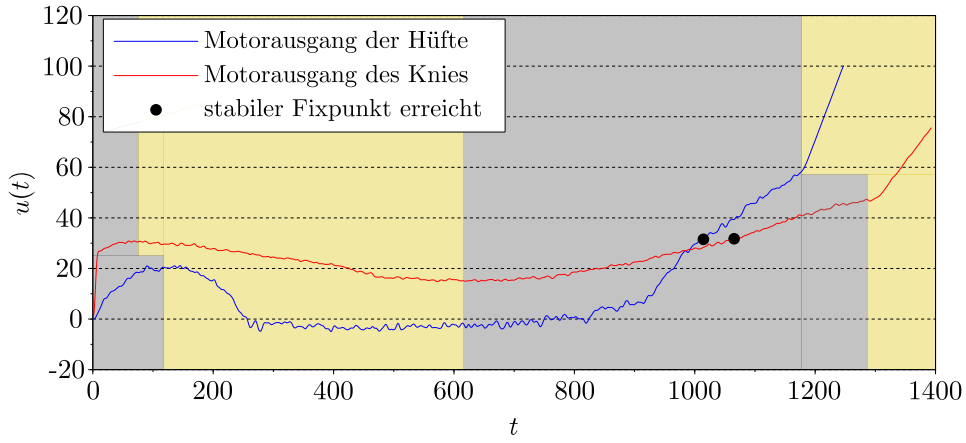


Abbildung 31: Dargestellt sind hier die Motorausgänge von Knie und Hüfte des Semni. In beiden Fällen war das jeweils andere Gelenk fixiert und die Bewegung wurde in der Luft, d.h. ohne externe Anschläge an Wänden, Böden oder Ähnlichem, ausgeführt. Knie und Hüfte haben ähnliche Verläufe, die sich grundsätzlich in vier Bereiche aufteilen lassen, die alternierend in gelb und grau hinterlegt sind.

Wie in Abbildung 31 zu sehen ist, haben Knie und Hüfte ähnliche Verläufe, die sich grundsätzlich in vier Bereiche aufteilen lassen, die alternierend in gelb und grau hinterlegt sind. Zu Beginn wachsen die Motorausgänge an. Dies liegt daran, dass zunächst die Haftreibung überwunden und anschließend die träge Masse auf den Zielwert beschleunigt wird. Im Anschluss nehmen die Motorausgänge zunehmend ab. Zu dieser Phase wirkt die Schwerkraft unterstützend auf die Bewegung ein. Im Verlauf des Hüftgelenks ist sogar zu beobachten, wie der Motorausgang das Vorzeichen wechselt, da das Gelenk sich allein durch die Schwerkraft schneller als die Zielgeschwindigkeit drehen würde, weshalb die Regelung in der Hüfte die Bewegung abbremst. Die Minima der Motorausgänge befinden sich ungefähr in der Mitte zwischen dem Beginn der Bewegung, also dem instabilen Fixpunkt, und dem stabilen Fixpunkt. Die Bewegung beginnt in der Nähe des ersten lokalen Maximums. Der stabile Fixpunkt ist durch einen schwarzen Punkt gekennzeichnet. Da die Bewegung bei 0° beginnt, der stabile Fixpunkt sich ungefähr bei 180° befindet und die Bewegung mit einer konstanten Geschwindigkeit ausgeführt wird, befinden sich die Minima in etwa bei 90°. Hier wirkt sich die Schwerkraft am meisten auf das Drehmoment der Gelenke aus:

$$F_{90^\circ} = m * g * \sin(\frac{\pi}{2}) \geq m * g * \sin(\phi) = F_\phi \; \forall \phi \in \mathbb{R}$$

Deswegen muss dort am wenigsten Drehmoment in bzw. am meisten Drehmoment entgegen der Bewegungsrichtung vom Motor erzeugt werden. In der Phase danach nimmt die Auswirkung der Schwerkraft auf das Drehmoment wieder ab, wodurch die Motorausgänge wieder anwachsen. An den markierten Stellen, bei denen der stabile Fixpunkt erreicht wurde, lässt sich ablesen, wie groß der Motorausgang hier sein muss, um die Bewegung an dieser Stelle mit der Zielgeschwindigkeit fortzuführen. Der Motorausgang würde anschließend immer weiter anwachsen und sein Maximum bei 270° erreichen, da hier die Schwerkraft am meisten der Bewegung entgegenwirkt. Bei den dargestellten Verläufen zeigt sich jedoch ein anderes Verhalten, da die Gelenke einen blockierenden Anschlag erreichen. Diese letzte Phase ist daran zu erkennen, dass es zu einem deutlich erkennbaren Knick in der Kurve kommt, welcher im Prinzip eine Nichtlinearität darstellt, die hier jedoch durch den Filter geglättet wird. Der Motorausgang steigt plötzlich viel stärker an, da die Bewegungsgeschwindigkeit auf einmal Null wird und sich unabhängig vom immer weiter ansteigenden Motorausgang nicht mehr ändert.

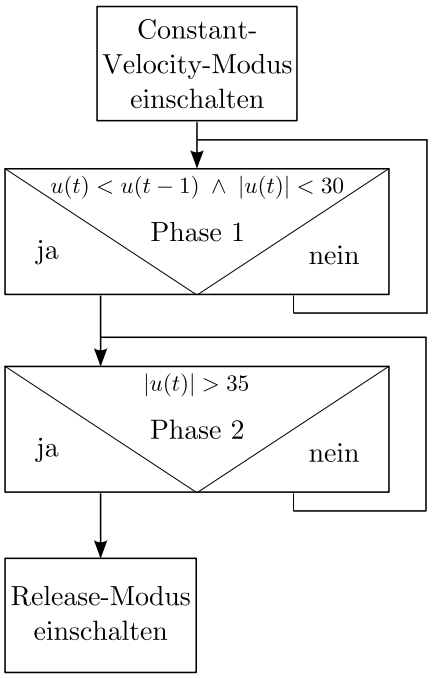


Abbildung 32: Abgebildet ist das Verfahren zur Symmetriebrechung beim Umschalten von Contraction nach Release. Zunächst wird darauf gewartet, dass der geglättete Ansteuerungswert $u(t)$ fällt und betragsmäßig kleiner 30 ist. Anschließend wird sobald dieser betragsmäßig wieder über 35 liegt der Release-Modus eingeschaltet.

Diese Beobachtungen wurden für die Erkennung des stabilen Fixpunkts folgender-

maßen genutzt: Nach Umschalten in den Constant-Velocity-Modus befindet sich das Verfahren in Phase 1. Es muss zunächst die Zielgeschwindigkeit erreicht werden. Dabei muss die Reibung und eventuell auch die wirkende Schwerkraft überwunden werden. Es wird gewartet, dass der Motorausgang in die Phase übergeht, in der er konstant fällt und betragsmäßig unterhalb des Wertes liegt, der im stabilen Fixpunkt benötigt wird, um die Zielgeschwindigkeit zu halten. Sobald dies geschieht wird in Phase 2 gewechselt. Die Schwerkraft wirkt nun in Bewegungsrichtung. In diesem Zustand wird darauf gewartet, dass der Motorausgang den Grenzwert aus Phase 1 wieder überschreitet. Dies ist der Punkt an dem in den Release-Modus umgeschaltet wird. Das Verfahren ist in Abbildung 32 dargestellt
Dass dieses Verfahren in dem Fall funktioniert, wenn das Gelenk sich frei dreht, lässt sich in Abbildung 31 erkennen, auf deren Grundlage es auch entwickelt wurde. Für den Fall, dass bereits ein Anschlag zwischen 0° und 180° vorhanden ist, gilt es, einige theoretische Überlegungen anzustellen. Zu unterscheiden sind hierbei zwei verschiedene Anschläge:

1. Blockierenden Anschlag
2. Nicht blockierenden Anschlag

Als erstes gibt es den bereits erwähnten blockierenden Anschlag, bei dem sich das Gelenk festfährt. Außerdem kann es zu einem nicht blockierenden Anschlag kommen, wobei sich das Gelenk weiter bewegen kann. In den Abbildungen 33 und 34 sind diese zwei Arten dargestellt. Es werden jeweils der Ausgangszustand $x(0)$, d.h. der instabile Fixpunkt (linkes Bild), der Moment des Anschlagens $x(t)$ (mittleres Bild) und ein Folgezustand $x(t + k)$ nach dem Anschlagen (rechtes Bild) gezeigt. Der Übergang geschieht, indem mittels Constant-Velocity-Modus vom instabilen Fixpunkt das Kniegelenk in die eine oder andere Richtung gedreht wird. Wenn man davon ausgeht, dass beim Stabilisieren des Fixpunkts eine leichte Oszillation stattfindet, herrscht beim Wechsel aus dieser Posture zumindest eine kleine Bewegungsfreiheit um den Fixpunkt herum vor. Daraus folgt, dass der Wechsel von Phase 1 in Phase 2 immer vor dem Anschlag liegt. Es reicht daher aus, sich im Folgenden auf die Bedingung für den Wechsel in den Release-Modus aus Phase 2 zu konzentrieren.

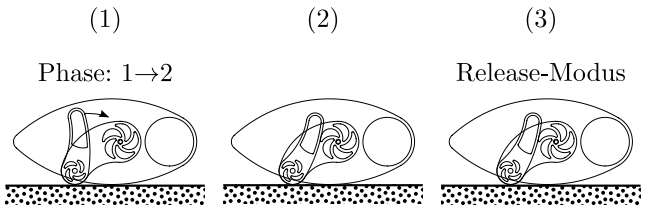


Abbildung 33: Gezeigt wird hier ein blockierender Anschlag. Das Knieglied stößt beim Verfahren gegen ein Bauteil des Semni und bleibt stehen. Trotz steigendem Ansteuerungswert des Motors bleibt die Winkelgeschwindigkeit konstant bei 0.

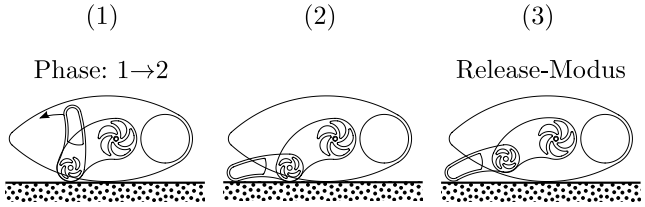


Abbildung 34: Abgebildet ist ein nicht blockierender Anschlag. Das Knieglied schlägt bei der Bewegung auf den Boden auf. Da das Kniegelenk im Release-Modus ist, hebt das Kniegelenk bei steigendem Motorwert das Hüftglied an, während das Knieglied am Boden weiter gleitet.

Im Falle des in Abbildung 33 dargestellten blockierenden Anschlages ist das einfach zu erkennen. Wie die Bilderserie verdeutlicht, schlägt das Gelenk an und bewegt sich danach nicht mehr weiter. Da bei diesem Anschlag die Geschwindigkeit augenblicklich Null wird und bleibt, wächst der Motorausgang immer weiter an und wird irgendwann den in Phase 2 gesetzten Schwellwert überschreiten. Folglich wird in diesen Fällen der Release-Modus am Anschlag eingeschaltet. Beim nicht blockierenden Anschlag aus Abbildung 34 ist eine etwas weitergehende Betrachtung notwendig. Nachdem das Gelenk vom instabilen Fixpunkt (linkes Bild) startet und am Untergrund anschlägt (mittleres Bild), richtet sich das Knieglied auf (rechtes Bild). Nun kommt es darauf an, wie sich in solchen Fällen der Motorausgang verhält. Beim Aufrichten muss das Gelenk mindestens das gleiche Gewicht wie beim aus Abbildung 31 beschriebenen Szenario bewegen. In diesem Szenario richtet sich die Schwerkraft ab dem stabilen Fixpunkt gegen die Bewegung. Jedoch ist die Auswirkung der Schwerkraft auf das Drehmoment in diesem Punkt minimal bzw. hat exakt am Fixpunkt gar keine Wirkung auf das Drehmoment. Auch beim Aufrichten, wie es im rechten Bild zu sehen ist, richtet sich die Schwerkraft gegen die Bewegung. Die Auswirkung der Schwerkraft kann am Anschlag nicht kleiner sein, da im stabilen Fixpunkt aus Abbildung 31 der kleinste Einfluss der Schwerkraft vorliegt. Der Motorausgang muss den gewählten Schwellwert überschreiten, weil weder das Gewicht noch die Wirkung der Schwerkraft am Anschlag kleiner werden als in dem Szenario aus Abbildung 31. Auch hier wird am Anschlag der Release-Modus aktiviert.

In beiden Fällen wird das Gelenk nach dem Einschalten des Release-Modus anschließend durch die Schwerkraft am Anschlag gehalten. In der Theorie sollte das Gelenk gebremst in die eine oder andere Richtung fallen. Dort würde das Gelenk ebenfalls zum Anschlag gelangen und locker liegend verharren. Somit hat sich das Verfahren als eine valide Implementierung für diesen Teil des Attractor-Based Behavior Control bewiesen.

6.1.2 Verfahren zur Symmetriebrechung beim Umschalten von Release zu Contraction

Beim Umschalten vom Release- in den Contraction-Modus wird eine gezielte Symmetriebrechung benötigt, um die Bewegung in eine bestimmte Richtung zu lenken. Im Gegensatz zu dem im vorangegangen Abschnitt beschriebenen Verfahren wird hier weder der Constant-Velocity-Modus genutzt und noch der in Abschnitt 4 beschriebene Bias verwendet. In diesem Fall wird der Integrator beim Umschalten mit einem Bias vorinitialisiert.

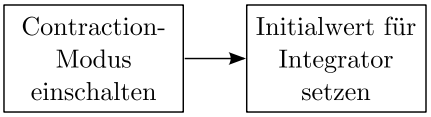


Abbildung 35: Dargestellt ist das Verfahren zu Symmetriebrechung beim Umschalten von Release nach Contraction. Zunächst wird der Contraction-Modus eingeschaltet und anschließend der Integrator mit einem Initialwert gefüllt.

Dieser ist so gewählt, dass er in die gewünschte Bewegungsrichtung führt und betragsmäßig groß genug ist, die Haftreibung zu überwinden. Es ist ausreichend, einen Wert für alle Postures zu benutzen, da diese Umschaltung nicht so kritisch ist. Er muss nur groß genug sein, um in jeder Posture eine Bewegung in die gewünschte Richtung zu induzieren. Im schlimmsten Fall führt ein zu großer Wert in der einen oder anderen Posture dazu, dass ein instabiler Fixpunkt übersprungen werden kann. Typischerweise sind dies Fixpunkte, die auch bei perfekter Wahl des initialen Motorwerts ab und zu vom Contraction-Modus übersprungen werden, da deren Basin, welches sie in diesem Modus ausbilden, sehr klein ist und deswegen kleinste Abweichungen in der Ausgangslage bereits darüber entscheiden, ob dieser Fixpunkt erreicht wird oder nicht. Insofern ist das zuverlässige Überspringen solcher Fixpunkte auch als Vorteil anzusehen, da es die Übergänge zuverlässiger macht. Des Weiteren sind von Anfang an die entsprechenden CSL-Parameter für den Contraction-Modus gesetzt.

Durch die ruckartige Bewegung, die durch das Setzen des initialen Ansteuerungswerts des Motors geschieht, fängt das CSL an, diesen Wert wieder zu verringern. Die äußeren Kräfte fangen an, auf das Gelenk einzuwirken und es in Richtung des stabilen Fixpunkts zu drängen, da es durch die initiale Bewegung aus diesem heraus bewegt wurde. Das führt dazu, dass das CSL sich gegen die Kräfte richtet und dem in entgegengesetzte Richtung liegenden instabilen Fixpunkt zustrebt. Dieser Ablauf ist in Abbildung 36 dargestellt. Mittels des soeben beschriebenen und in Abbildung 35 schematisch dargestellten Verfahrens kann gezielt in eine Richtung der jeweils nächste instabile Fixpunkt von einem beliebigen stabilen Fixpunkt aus angesteuert werden.

Abbildung 36: Abgebildet ist ein Beispiel für einen Bewegungsablauf, der durch das Verfahren zu Symmetriebrechung beim Umschalten von Release nach Contraction erfolgt. Links wird im stabilen Fixpunkt in den Contraction Modus umgeschaltet und der Integrator mit einem Initialwert gefüllt. Dadurch wird eine Bewegung induziert und die Schwerkraft fängt an dieser entgegenzuwirken. Dies führt dazu, dass sich der Contraction-Modus des CSL gegen die Schwerkraft richtet und den ehemals instabilen Fixpunkt stabilisiert.

6.2 Aufbau des sensomotorischen Graphen

Nachdem im Abschnitt 6.1 beschrieben wurde, wie der Übergang von einem Fixpunkt in den nächsten umgesetzt wurde, wird in diesem Abschnitt der Aufbau des sensomotorischen Graphen erläutert. Hierzu müssen zunächst die Postures erkannt und anschließend inklusive der zugehörigen Übergänge abgespeichert werden. Des Weiteren bedarf es einer Strategie für das Auswählen der jeweils nächsten Aktion. Die Lösungen für diese Probleme, die dazu nötig sind, einen sensomotorischen Graphen wie in Abbildung 37 aufzubauen, werden im Folgenden dargelegt.

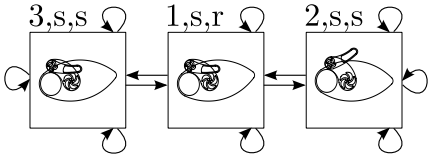


Abbildung 37: Abgebildet ist ein einfaches Beispiel für einen sensomotorischen Graphen, wie er durch das hier beschriebene ABC-Learning aufgebaut wird.

6.2.1 Erkennung des Erreichens einer Posture

Ist der Roboter ausgeschaltet, sind alle Gelenke in einer Art Release-Modus. Deswegen wird beim Einschalten auch zunächst in jedem Gelenk dieser Modus aktiviert, damit der Roboter in dem Startzustand $x(0)$ verharrt. Es wird weiterhin angenommen, dass $x(0)$ stabil ist und somit eine Posture bildet. Dies wäre beispielsweise nicht der Fall, wenn er zu Beginn vom Menschen in irgendeinem instabilen Zustand $x_i(0)$ gehalten wird und nach dem Einschalten losgelassen wird oder sich während des Anschaltens mitten in einer von außen induzierten Bewegung befindet. Sind diese Bedingungen erfüllt, so kann dieser initiale Zustand als erste Posture erkannt werden, da der Semni sich in jedem Gelenk in einem stabilen Fixpunkt und im Release-Modus befindet.
Ausgehend von dieser Posture wird in einem Gelenk das Verfahren zur Symmetriebrechung aus Abschnitt 6.1.2 angewandt. Bei diesem Verfahren ist das Gelenk die ganze Zeit im Contraction-Modus und kommt unter Umständen nie vollständig zum Stehen. Zur Erkennung der Posture reicht es deswegen nicht aus, nur zu detektieren, dass die Bewegung gestoppt hat. Um den Aufwand zu minimieren, wurde an dieser Stelle ein langsamer, jedoch einfacher und zuverlässiger Detektor entwickelt: Wie bereits beschrieben, kommt es bei Erreichen des instabilen Fixpunkts oft zu mehr oder weniger kleinen Oszillationen. Die Bewegungen auf dem Weg hin zu diesem Fixpunkt haben meistens keine bzw., bei einem Sprung auf eine andere Untermannigfaltigkeit, maximal eine Richtungsumkehr. Als erstes Kriterium zur Detektion des instabilen Fixpunkts dient daher die Anzahl der Richtungsumkehrungen. Um möglichst nahe am Fixpunkt zu sein, wird die Grenze zum Erkennen des Fixpunktes nicht bei 1 sondern bei 10 gesetzt, da z. B. bei etwas stärkerem Überschwingen das Gelenk nach ein paar Schwingungen immer näher um den Fixpunkt oszilliert. Wurde die gewählte Grenze an Richtungsumkehrungen überschritten, wird die aktuelle Konfiguration als Posture erkannt. In Abbildung 38 sind verschiedene Verläufe der Ansteuerungswerte des Motors beim Übergang von einem stabilen in einen instabilen Fixpunkt dargestellt.

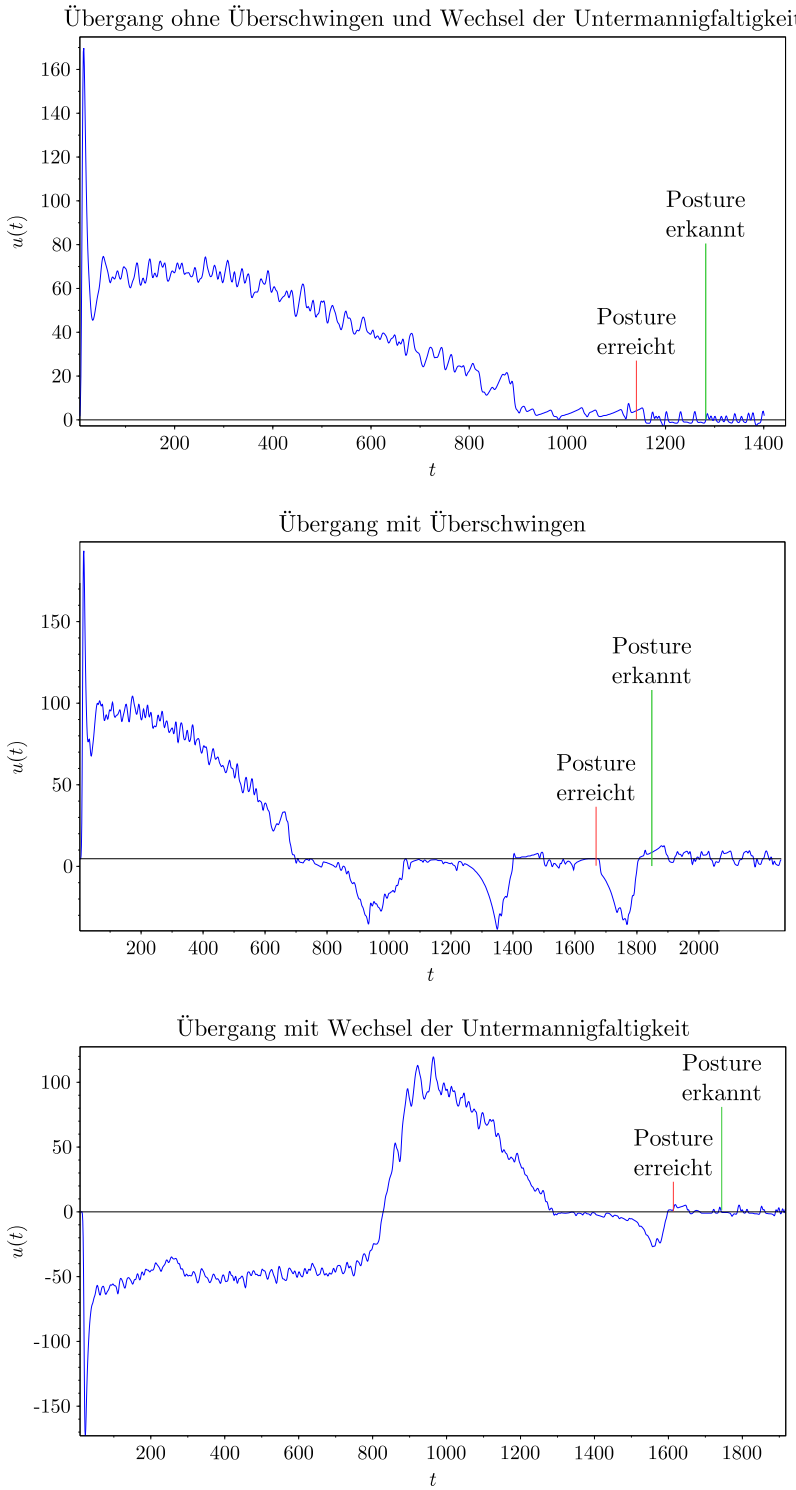


Abbildung 38: Dargestellt sind die Motorwerte bei verschiedenen Übergängen vom stabilen in den instabilen Fixpunkt. Zu sehen ist außerdem, wann die Posture erreicht wurde und wann das Verfahren dies detektiert.

Kommt die Bewegung im Fixpunkt an und verharrt dort ohne Oszillation, würde dieses Verfahren fehlschlagen. Ein Timeout fängt diese Fälle ab. Es ist lang genug gewählt, sodass jeder Übergang vom instabilen in den stabilen Fixpunkt innerhalb dieser Zeit abgeschlossen ist. Empirisch wurde hierfür eine Dauer von 60 Sekunden bestimmt. Ist der Timeout abgelaufen, so wird der Zustand $x(t)$ zu diesem Zeitpunkt als Posture erkannt. Spätestens nach dem Timeout wird dadurch immer eine gültige Posture erkannt. Zusätzlich wurde die Möglichkeit geschaffen, per Druck einer Taste auf der Tastatur des PCs dem Programm das Erreichen der Posture mitzuteilen, um so Zeit zu sparen, wenn man z. B. Experimente durchführen will.

Für das Erreichen des stabilen Fixpunktes ist keine komplizierte Auswertung notwendig. Hier wird die Erkennung des stabilen Fixpunkts bereits durch das in Abschnitt 6.1.1 beschriebene Verfahren übernommen. Im Prinzip kann man davon ausgehen, dass der Fixpunkt erreicht ist, sobald vom Constant-Velocity- in den Release-Modus umgeschaltet wird. Da es in einigen Fällen vorkommen kann, dass nicht genau am Fixpunkt sondern kurz davor oder dahinter in den Release-Modus geschaltet wird, gibt es zunächst eine kleine Wartezeit bis die Posture als erkannt gilt, um so den von außen wirkenden Kräften Zeit zu geben, das Gelenk näher an den Fixpunkt zu bewegen. Hierzu wurden die Beispiele für das Verfahren in Abbildung 39 erweitert.

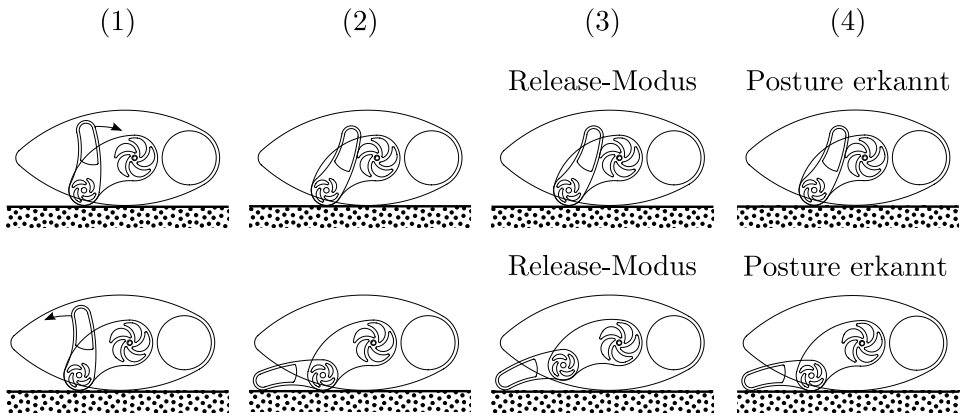


Abbildung 39: Abgebildet sind hier die erweiterten Beispiele aus Abschnitt 6.1.1. Die ersten drei Schritte sind wie bekannt. Zwischen Schritt 3 und 4 wird eine weile gewartet, so das der stabile Fixpunkt seine anziehende Wirkung entfalten kann. Danach wird die Posture erkannt und gegebenenfalls abgespeichert.

Als letzten Sonderfall für das Erreichen einer Posture sind die Anschläge zu betrachten, die im Contraction-Modus angefahren werden. Hier wird eine Posture dann als erreicht angesehen, sobald in den bereits erwähnten Stall-Modus umgeschaltet wird.

Dabei spielt es keine Rolle von welchem Fixpunkt man kommt, da zu diesem Zeitpunkt das Gelenk sich aufgrund des Anschlags nicht weiter bewegen kann und so die Posture am Anschlag erreicht hat.

6.2.2 Repräsentation des sensomotorischen Graphen

Wie in Abschnitt 4 definiert, besteht eine Posture aus den Gelenkwinkeln, dem Körperwinkel und den CSL-Modi jedes Gelenks. Zusätzlich zu diesen Informationen werden eine ID, die aktuellen Motorströme, die Spannung am AccelBoard3D und eine Liste mit allen Nachbarn in einer Structure gespeichert. Außerdem gibt es eine Reihe an frei wählbaren Feldern, in denen beliebige Informationen zum Debuggen oder für andere Verfahren gespeichert werden können. Die Liste der Nachbarn hat eine eigene Structure, die zunächst einen Pointer zu der benachbarten Posture enthält. Außerdem wird in ihr gespeichert, welches Gelenk gerade in welche Richtung umgeschaltet wurde. Die resultierenden Belegungen für das Hüftgelenk (h) und Kniegelenk (k) können der Tabelle 2 entnommen werden.

Umschaltung	h	k
Hüfte von Release nach Contraction in positive Richtung	1	0
Hüfte von Release nach Contraction in negative Richtung	-1	0
Knie von Release nach Contraction in positive Richtung	0	1
Knie von Release nach Contraction in negative Richtung	0	-1
Hüfte von Contraction nach Release in positive Richtung	1	0
Hüfte von Contraction nach Release in negative Richtung	-1	0
Knie von Contraction nach Release in positive Richtung	0	1
Knie von Contraction nach Release in negative Richtung	0	-1

Tabelle 2: Übersicht über die Belegung der Variablen zum Merken der letzten Umschaltung aus der Nachbars-Structure

Auch hier gibt es zusätzliche freie Felder. Es bietet sich an, in diesen zusätzliche Informationen zu speichern, die den Übergang von der einen in die andere Posture charakterisieren, um diese für spätere Auswertungen nutzen zu können. In dieser Arbeit wird zunächst die Zeit in Millisekunden und die Anzahl an Regelungsschritte gespeichert, die für den Übergang gebraucht wurde. Außerdem wird für jedes Gelenk jeweils der maximale und minimale Ansteuerungswert des Motors und fließende Strom erfasst. Weiterhin werden die jeweiligen Summen der Motorausgänge und Ströme über die Zeit gebildet und in der Structure abgelegt. Solche Informationen können für Verfahren wie dem von Benjamin Werner Werner (2013) genutzt

werden. Abschließend wird abgespeichert wie oft diese Verbindung durchfahren wurde. Mit Hilfe des zuletzt genannten Wertes werden in jedem Durchlauf alle anderen Werte immer wieder mit den Werten des aktuellen Durchlaufs dieser Verbindung gemittelt. Somit ist der Durchschnittswert $D(n)$ der jeweiligen Felder durch folgende Definition gegeben, wobei die aktuelle Anzahl an Durchläufen durch n gegeben ist und w den neusten Wert kennzeichnet:

$$D(0) = 0$$
$$D(n) = \frac{D(n-1) \cdot (n-1) + w}{n}$$

Alle Postures werden in einer Liste gespeichert, wobei der Listenindex der ID der Posture entspricht. Die entstehende, gesamte Datenstruktur zum Speichern des Graphen wird in Abbildung 40 beispielhaft dargestellt.

<table>
<tr><td rowspan="2">ID</td><td rowspan="2">CSL Hip</td><td rowspan="2">CSL Knee</td><td rowspan="2">...</td><td colspan="4">Neighbors</td></tr>
<tr><td>*</td><td>h</td><td>k</td><td>...</td></tr>
<tr><td>1</td><td>r</td><td>r</td><td>...</td><td>*2</td><td>0</td><td>1</td><td>...</td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="4">...</td></tr>
<tr><td>2</td><td colspan="7">...</td></tr>
<tr><td>...</td><td colspan="7">...</td></tr>
</table>

Abbildung 40: Abgebildet ist eine schematische Darstellung der Datenstruktur zur Speicherung des sensomotorischen Graphen. In vertikaler Richtung sind die einzelnen Postures aufgelistet. Horizontal werden exemplarisch einige Felder einer Posture dargestellt. Das Feld „Neighbors“ ist für jede Posture eine eigene Liste. In ihr ist ein eigener Datentyp gespeichert, dessen Felder auszugsweise gezeigt werden. Mit „*2“ ist ein Zeiger auf p_2 gekennzeichnet.

Die Datenstruktur wird beim Beenden des Programms in einer Textdatei auf der Festplatte gesichert und beim nächsten Start wieder ausgelesen. Erfahrungen gehen nicht verloren, wodurch Experimente unterbrochen und später fortgeführt werden können.

Nach Einschalten des Semni wird, wie im letzten Abschnitt beschrieben, die erste Posture erkannt. Dies führt dazu, dass versucht wird, diese in der Datenstruktur abzulegen. Für jede erkannte Posture wird geprüft, ob es bereits eine ähnliche Posture im Graphen gibt. Zu jeder bestehenden Posture, die in beiden Gelenken die gleichen CSL-Modi hat, wird der euklidische Abstand über die Gelenkwinkel und Körperwinkel bestimmt. Gibt es keine Posture mit den gleichen CSL-Modi, die nah genug an der aktuellen Posture ist, so wird eine neuer Eintrag für diese Posture angelegt. Nach erfolgreichem Übergang einer Posture in die nächste wird dieser in

der vorherigen Posture abgelegt bzw. aktualisiert, wenn er schon vorhanden ist.

6.2.3 Heuristik zur Wahl der nächsten Aktion

Wie in Abschnitt 4 beschrieben, gibt es beim Semni nach Erreichen einer Posture vier Möglichkeiten zu explorieren. Bei der Auswahl der nächsten Aktion werden zwei Entscheidungen getroffen. Einerseits, welches Gelenk als nächstes umgeschaltet wird und andererseits in welche Richtung man dies tut. Der einfachste Ansatz ist, zufällig zu entscheiden, welches Gelenk in welche Richtung umgeschaltet wird. Dieser ist völlig ungerichtet und dient keinem besonderen Ziel. Ein Ziel könnte beispielsweise sein, mit möglichst wenigen Übergängen alle Potsures und Verbindungen zu finden. Ohne den gesamten Graphen bereits zu kennen, ist es nicht möglich, immer die richtige Aktion für dieses Ziel auszuwählen. Um trotzdem eine möglichst gute Entscheidung treffen zu können, bietet es sich an, auf Heuristiken zurückzugreifen.
Für die Situationserkennung spielt nur die Dauer der Exploration eine Rolle. Diese soll möglichst kurz sein, um die Experimente zu beschleunigen. Es wurde eine Heuristik gewählt, die dazu geeignet ist, die Exploration nicht unnötig zu verlängern: Als nächste Aktion wird diejenige ausgewählt, die auf dem kürzesten Weg zu einer neuen Verbindung liegt. Es gibt dabei verschiedene Fälle zu betrachten. Ist man in einer Posture angekommen, in der noch nicht alle Möglichkeiten ausprobiert wurden, so wird aus den noch übrigen Möglichkeiten gleichverteilt zufällig die nächste Aktion gewählt. Sollte man eine Posture erreichen, in der bereits alle Möglichkeiten ausprobiert wurden, so wird nach der nächsten Posture gesucht, in der dies noch nicht der Fall ist. Dort angekommen, wird wie bereits beschrieben eine der noch offenen Möglichkeiten zufällig gewählt. Die Abstände zu den jeweiligen Knoten werden dabei mittels Dijkstras Algorithmus berechnet, wie er in Cormen u. a. (2001) beschrieben ist. Jedem Übergang wird dabei eine Weglänge von Eins zugemessen. Ein Beispiel hierfür findet sich in den Abbildung 41 bis 43.

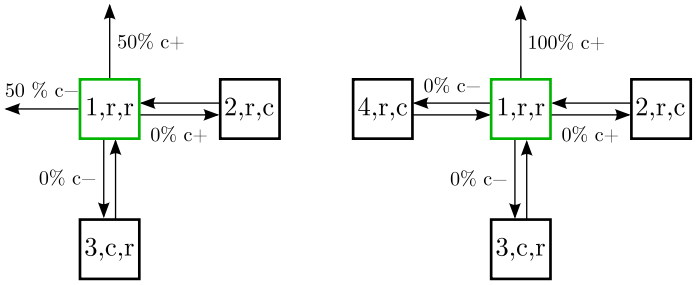


Abbildung 41: Abgebildet sind zwei Fälle, in denen in der aktuellen Posture jeweils noch mindestens eine Möglichkeit zur Exploration offen ist. Jedes Quadrat repräsentiert eine Posture. Links sind noch zwei Möglichkeiten offen, weshalb zwischen diesen gleichverteilt zufällig gewählt wird. Rechts ist nur noch eine Aktion offen, die deswegen als nächstes gewählt wird.

Die Beschriftung innerhalb der Postures, die durch Quadrate dargestellt werden, ist gegliedert in eine ID und den Betriebsmodi der Hüfte und des Knies, in genannter Reihenfolge, separiert durch Kommata. Dabei steht „c“ für den Contraction- und „r“ für den Release-Modus. Zeigt ein Pfeil ins Leere, so wurde diese Möglichkeit noch nicht ausprobiert. Vertikale Pfeile sind Umschaltungen in der Hüfte und horizontale Pfeile im Knie. Ein Rechtspfeil bzw. ein Pfeil nach oben bedeutet eine positive Drehrichtung, wohingegen die anderen Pfeile eine negative Drehrichtung kennzeichnen. Die aktuelle Posture ist durch einen grünes Quadrat gekennzeichnet. An den von dort ausgehenden Pfeilen sind jeweils Prozente notiert, die die Wahrscheinlichkeit angeben, dass diese Aktion als nächstes gewählt wird. In Abbildung 41 sind zwei Graphen zu sehen, bei denen die jeweils aktuelle Posture Aktionen hat, die noch nicht ausprobiert wurden. Im linken Fall sind Aktionen bereits exploriert. Für die nächste Aktion kommen diese nicht mehr in Frage, was durch die „0%“ gekennzeichnet ist. Zwischen den anderen beiden Aktionen wird zufällig mit einer Wahrscheinlichkeit von jeweils 50 Prozent gewählt. Im rechten Fall ist nur noch eine Möglichkeit noch nicht exploriert, weshalb diese zu 100 Prozent als nächstes ausprobiert wird.

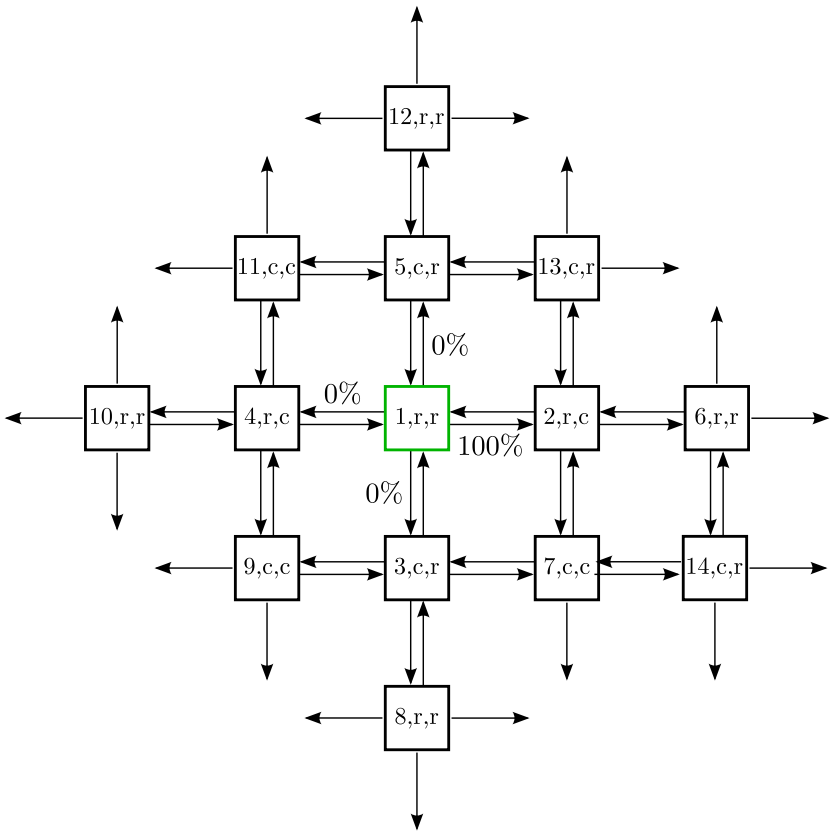


Abbildung 42: In diesem Fall ist in der aktuellen Posture bereits alles exploriert. Es wird nach dem nächsten Knoten gesucht, der noch nicht voll exploriert ist. Da ausgehend von diesem Knoten mehrere nicht voll explorierte Postures existieren, die den gleichen minimalen Abstand haben, wird p_6 ausgewählt, da diese die kleinste ID hat. Es wird als nächste Aktion die Hüfte in positive Richtung umgeschaltet, um dorthin zu gelangen.

In Abbildung 42 ist ein Fall zu sehen, bei dem in der aktuellen Posture alle Aktionen bereits mindestens einmal ausprobiert wurden. Dies führt dazu, dass nach der nächsten Posture gesucht wird, bei der mindestens eine Aktion noch frei ist. Infrage hierfür kommen die Postures p_6 bis p_{14}. Die Postures p_2 bis p_5 sind bereits voll exploriert. Von den genannten Postures sind p_6 bis p_{13} über zwei Kanten zu erreichen, während p_{14} über drei Kanten zu erreichen ist und damit als nächstes rausfällt. Zwischen den übrigen Postures entscheidet die ID. Die niedrigste ID (Nummer 6) wird als nächstes angesteuert. Als nächste Aktionen wird zweimal „c+“ im Knie vorgemerkt, um die Berechnung nicht nach jedem Schritt wieder neu ausführen zu müssen. Nach der ersten Ausführung von „c+“ ist p_2 die neue aktuelle Posture und die Anzahl der vorgemerkten Aktionen verringert sich um Eins. Es entsteht der in Abbildung 43 dargestellte Zustand.

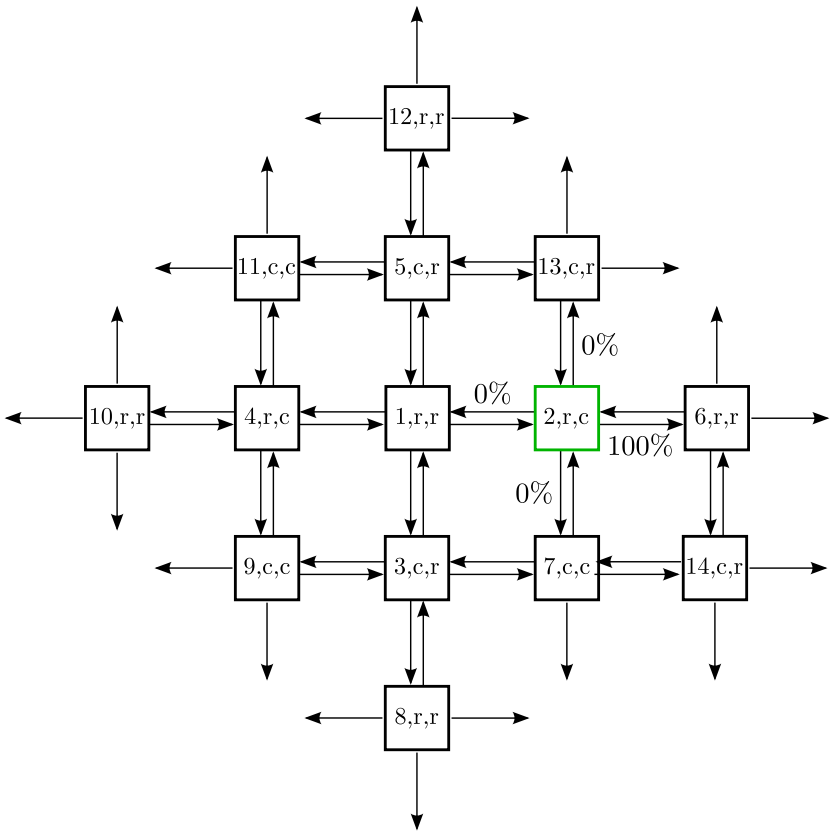


Abbildung 43: Auf dem Weg zu p_6 wird zunächst p_2 erreicht. Hier wird nicht neu nach der nächsten Aktion gesucht, sondern die nächste vorgemerkte Aktion ausgeführt.

In p_2 wird die nächste vorgemerkte Aktion ausgeführt und die Anzahl der vorgemerkten Aktionen verringert. Die neue aktuelle Posture ist p_6. Es gibt keine vorgemerkten Aktionen mehr, deshalb wird zwischen den unexplorierten Kanten nach bekanntem Schema die nächste Aktion ausgewählt.
Nach diesem Prinzip wird immer weiter verfahren, bis der gesamte Graph exploriert ist. Wenn es keine offenen Möglichkeiten mehr in allen erreichbaren Postures gibt, wird komplett zufällig die nächste Aktion gewählt.

Es ist klar, dass diese Heuristik schneller exploriert, als eine zufällige Wahl der nächsten Aktion, da diese zielgerichtet nach offenen Möglichkeiten sucht. Des Weiteren ist ersichtlich, dass auf diese Art und Weise der gesamte Graph irgendwann exploriert ist und man nicht etwa in einem lokalen Minimum verharrt. Außerdem bietet dieses Verfahren die Möglichkeit, es leicht für andere Ziele zu erweitern, wie später in Abschnitt 7.6 beschrieben ist.

7 Situationserkennung

Dieses Kapitel behandelt den Kern der Arbeit, die Situationserkennung. An dieser Stelle soll zuerst der Bezug zum ABC-Learning hergestellt werden. Als Nächstes wird der Begriff einer Situation näher betrachtet und für diese Arbeit festgelegt. Anschließend wird das Verfahren beschrieben, welches eine Situationserkennung realisieren soll. Abschließend wird auf die Implementierung dieses Verfahrens eingegangen.

7.1 Situationserkennung und ABC-Learning

Das ABC-Learning exploriert den Zustandsraum des Roboters und speichert die Ergebnisse in einem Graphen ab. Wie in Abschnitt 4.4 erwähnt, lässt sich dieser Graph auf Grund seiner Entstehungsweise und den in ihm enthaltenen Daten auch als sensomotorischer Graph bezeichnen. Bei der Entstehung des Graphen laufen Diffusionsprozesse ab, die eine gezielte Exploration von unbesuchten Kanten erlauben. Die Idee ist, den aufgebauten, sensomotorischen Graphen zu erweitern und auf dessen Basis ein Verfahren zur Situationserkennung zu realisieren. Die Diffusionsprozesse und der sensomotorische Graph des ABC-Learning bilden eine erste Basis für die Situationserkennung, wie sie in dieser Arbeit beschrieben wird.

7.2 Situationen

Seit Entstehung hat sich der Begriff der Situation weiter entwickelt und wurde in verschiedenen Bereichen unterschiedlich benutzt. In der Philosophie kann nach Großheim (2005) der Begriff der Situation, als das augenblicklich im Gehirn eines Sprechenden oder Hörenden vorhandene „Weltbild“ verstanden werden. Dies ist für diese Arbeit zu spezifisch, da es ausschließlich auf den Menschen bezogen wird, kann jedoch als eine erste Grundlage dienen, da die Welt aus Sicht des Semni große Bedeutung für den verwendeten Situationsbegriff hat. Nach Scholze-Stubenrecht (2011) bezeichnet eine Situation Verhältnisse oder Umstände, in denen sich jemand befindet, bzw. die einen allgemeinen Zustand kennzeichnen. Auch diese Bedeutung enthält Elemente, die für eine Begriffsbildung der Situation in diesem Zusammenhang nützlich sind. In dieser Arbeit geht es darum, in welchen groben Gegebenheiten, oder auch allgemeinen Umständen, sich der Roboter Semni befindet, die ihm erlauben sich in einem bestimmtem Rahmen zu bewegen. Eine Situation S_i ist somit definiert als die Menge aller Zustände $x(t)$, die unter den gegebenen Umständen mit einem gewissen Drehmoment in den Gelenken stabil gehalten werden kann:

$$S_i = \bigcup \{x(t)\} \text{ , } x(t) \text{ kann vom Semni stabil gehalten werden}$$

Die Situation S_i beschreibt somit den Zustandsraum unter den vorherrschenden Bedingungen und bildet damit eine Mannigfaltigkeit in $\mathbb{R}^3$. Kleinste Änderungen, wie das leichte Ankippen der Unterlage, würden nach dieser Definition verschiedene Situationen hervorrufen, da sich die Menge der Zustände ebenfalls leicht verschieben würde. Dies wird in Abbildung 44 verdeutlicht.

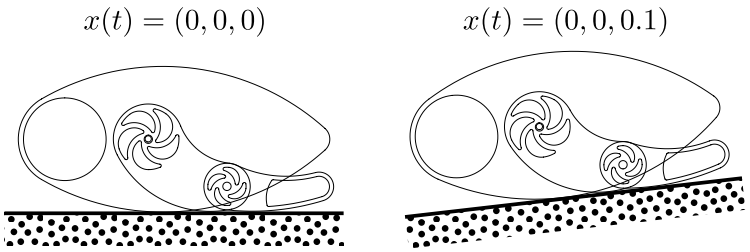


Abbildung 44: Abgebildet sind zwei Gegebenheiten, die sich nur durch eine kleine Änderung des Bodens unterscheiden. Die Zustand des Semni wird links als $x(t) = (0, 0, 0)$ angenommen. Dementsprechend führt die kleine Änderung des Untergrunds dazu, dass der Zustand rechts sich ändert. Allein dies würde nach bisheriger Definition genügen, Links und Rechts als verschiedene Situationen anzusehen.

Diese Definition würde unnötig viele Situationen unterscheiden. Deswegen wird diese im Folgenden noch weiter eingeschränkt:

$$S_i = \bigcup \{x(t)\} \text{ , } x(t) \text{ ist stabiler Fixpunkt im Release- oder Contraction-Modus}$$

Nach dieser Definition besteht eine Situation S_i aus allen Fixpunkten, die das dynamische System, bestehend aus Semni und Umwelt, hat, wobei alle Fixpunkte gemeint sind, die auf Grund des Release- oder Contraction-Modus entstehen. Situationen sind dann verschieden, wenn es zu einer Bifurkation kommt. In Abbildung 45 sind einige solcher Situationen dargestellt.

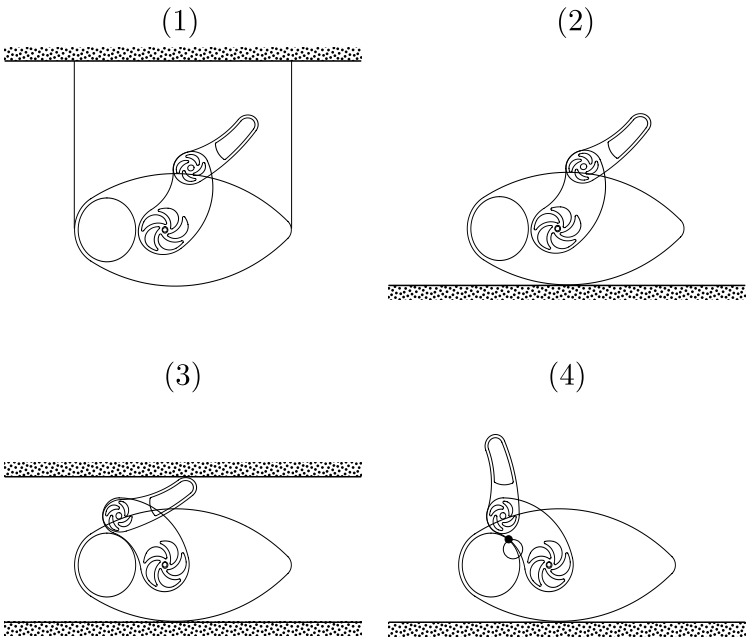


Abbildung 45: Abgebildet sind vier verschiedene Situationen, die sich nach dem in dieser Arbeit verwendeten Situationsbegriff unterscheiden. Oben links ist eine Situation, in der sich der Semni frei bewegen kann und in der Luft hängt. Rechts davon ist eine Situation in der er sich ebenfalls frei bewegen kann, jedoch auf einem Untergrund liegt. Unten links ist eine Situation zu sehen, in der er in einer Art Tunnel befindet und seine Bewegungsmöglichkeiten stark eingeschränkt sind. In der letzten Situation ist die Hüfte des Semni am Kopf fixiert, während er sein Knie weitgehend ungehindert bewegen kann.

Zusätzlich werden Situationen danach unterschieden in welcher Lage sich der Semni befindet, wenn ihm die äußeren Gegebenheiten nicht erlauben alle Lagen zu erreichen. Beispielhaft wurde dies in den Abbildungen 46 und 47 dargestellt.

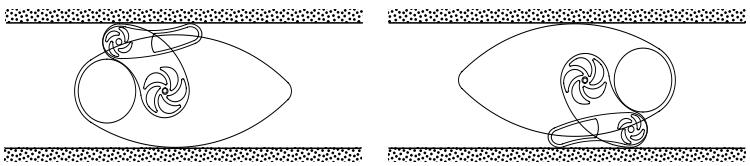


Abbildung 46: Dargestellt sind zwei verschiedene Situationen, die sich nur durch die Lage des Semni unterscheiden. Des Weiteren ist seine Bewegung von oben und unten beschränkt. Es ist für ihn nicht möglich, von der einen Lage in die andere zu wechseln so lange er auf diese Weise beschränkt wird. Deswegen sind dies zwei verschiedene Situationen.

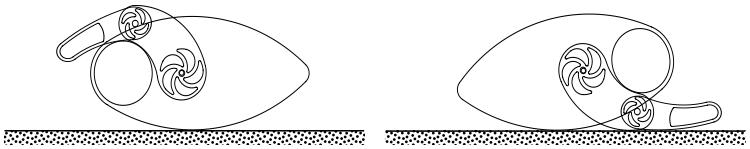


Abbildung 47: Abgebildet ist hier der Semni in einer Situation in verschiedenen Lagen. Bei diesem Beispiel befindet sich der Semni jeweils auf einem Tisch und kann sich frei bewegen. Folgend wird die Lage der linken Abbildung als Rückenlage und die der rechten Abbildung als Bauchlage bezeichnet. In diesem Fall wird nicht von zwei verschiedenen Situationen gesprochen, da es für den Semni aus eigener Kraft möglich ist, von der einen in die andere Lage zu wechseln.

Der hier eingeführte Begriff einer Situation wird nun dahingehend untersucht, welchen Einfluss die verschiedenen Situation auf das ABC-Learning haben.

7.3 Einfluss verschiedener Situationen auf das ABC-Learning

7.3.1 Einfluss verschiedener Gegebenheiten auf die Fixpunkte

Das ABC-Learning exploriert die Fixpunkte des Semni. Die vorherrschenden Gegebenheiten haben verschiedene Einflüsse auf diese. Es kann dazu kommen, dass Fixpunkte verschwinden bzw. nicht mehr erreichbar sind. Ein Beispiel hierfür ist in Abbildung 48 gegeben.

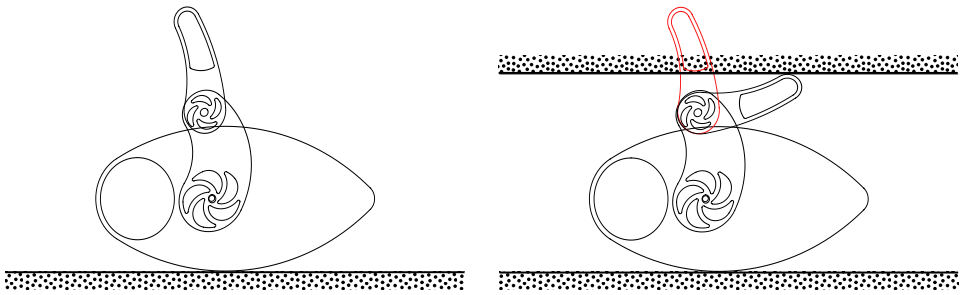


Abbildung 48: In diesen zwei Gegebenheiten ist zu sehen, wie ein Fixpunkt nicht mehr erreichbar ist. Deswegen sind dies nach Definition auch zwei verschiedene Situationen

In der linken Situation ist der instabile Fixpunkt, in dem sich das Knie gegen die Schwerkraft richtet, zu sehen. Bei der anderen Situation ist dieser nicht mehr erreichbar, da das Knie vorher gegen die Decke stößt und in den Stall-Modus wechselt.
Ein weiterer Einfluss kann sein, dass Fixpunkte verschwinden und an anderer Stelle neue entstehen. Außerdem ist es möglich, dass sich die Art des Fixpunkts ändert oder neue Fixpunkte hinzukommen. Abbildung 49 zeigt einen weiteren möglichen Einfluss der Gegebenheiten auf die Fixpunkte.

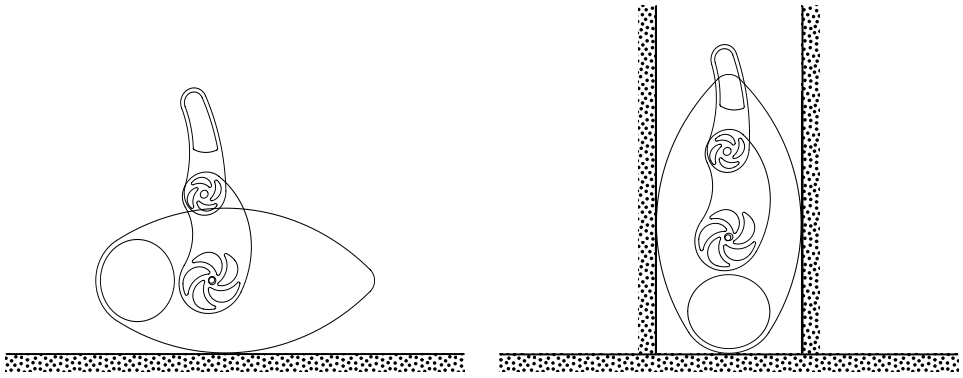


Abbildung 49: In diesen zwei Gegebenheiten ist zu sehen, wie ein Fixpunkt an einer Stelle verschwindet und an einer anderen Stelle neu entsteht. Der Fixpunkt „beide Gelenke richten sich gegen die Schwerkraft“ ist im rechten Bild an einer anderen Stelle entstanden, nämlich im Körper- und Hüftwinkel um 90° gedreht.

Links ist wieder der instabile Fixpunkt zu sehen, bei dem sich beide Gelenke gegen die Schwerkraft richten. Rechts davon ist zu sehen, dass der instabile Fixpunkt, bei dem sich beide Gelenke gegen die Schwerkraft richten und das Bein in die Luft zeigt, an einer anderen Stelle entstanden ist. Der Körper- und Hüftwinkel sind nun um 90° gedreht, während der Kniewinkel gleich bleibt. Der alte Fixpunkt ist verschwunden und während an einer anderen Stelle ein neuer entstanden ist.

Es wurde gezeigt, dass verschiedene Gegebenheiten, so wie sie hier definiert wurden, verschiedenste Einflüsse auf die Fixpunkte des Semni haben und nach Definition so verschiedene Situationen bedingen. Dies hat unmittelbar Folgen für das ABC-Learning bzw. den gebildeten Graphen, auf die im folgenden Abschnitt eingegangen wird.

7.3.2 Einfluss verschiedener Situationen auf den sensomotorischen Graphen: Äußere und innere Situation

Der sensomotorische Graph verändert sich abhängig von der Situation, da die verschiedenen Situationen verschiedene Fixpunkte haben und das ABC-Learning anhand dieser den Graphen aufbaut. Ist ein Fixpunkt nicht mehr erreichbar, so verschwindet auch die Posture in dem zugehörigen sensomotorischen Graph. Verschwindet eine Posture und taucht an anderer Stelle wieder auf, so führt dies im Graph dazu, dass eine neue Posture entsteht, während die alte Posture nicht mehr existiert. Ändert sich die Art eines Fixpunkts, ändern sich in der Posture die verwendeten Betriebsmodi des CSL. Dies sind zunächst die direkten Auswirkungen auf die Postures an sich, welche die Knoten des Graphen bilden. Jedoch führt die Änderung

einer Situation auch zu Veränderungen der Verbindungen zwischen den Postures. Ist ein Fixpunkt nicht mehr erreichbar, verschwindet auch die zugehörige Posture. Alle vormals benachbarten Postures, die erhalten geblieben sind, haben nun keine Verbindung mehr dorthin, sondern zu einer anderen oder neuen Posture.

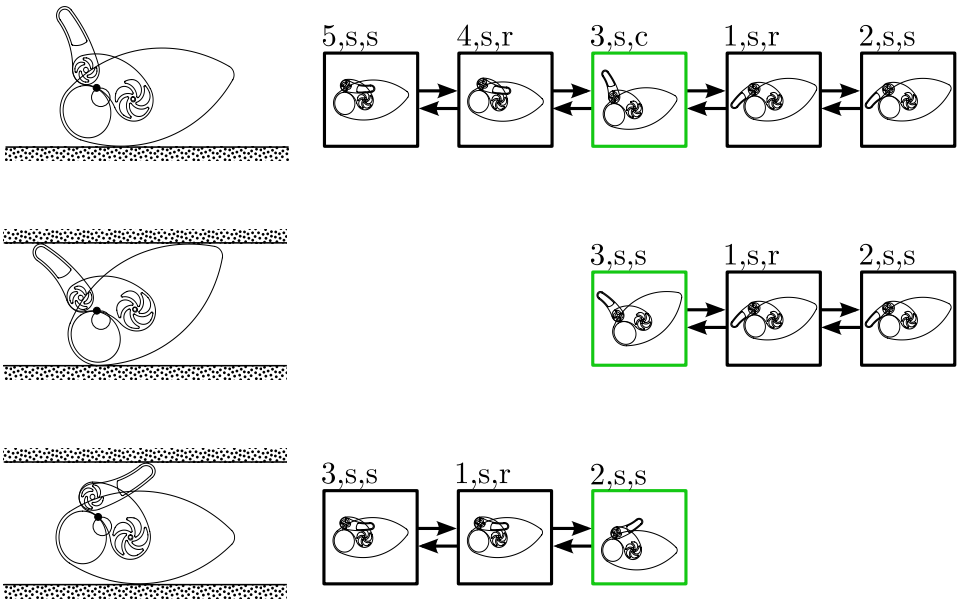


Abbildung 50: Abgebildet ist eine Gegenüberstellung von äußerer und innerer Situation für drei verschiedene Situationen. Links ist jeweils die äußere und rechts die zugehörige innere Situation zu sehen. Bei der inneren Situation stellt jeder kleine Semni eine Posture dar, wobei ein Pfeil anzeigt, dass es einen Übergang zwischen den Postures in Richtung des Pfeils gibt. Links über jeder Posture sind eine ID, sowie die Betriebsmodi von Hüfte und Knie angegeben. Die Bedeutung der Richtungen der einzelnen Pfeile entspricht der aus den Abbildungen 41-43. Zur Übersichtlichkeit wurden bei allen Umschaltungen die Pfeile weggelassen, bei denen kein Wechsel der Posture erfolgt. Beispielsweise sind alle Pfeile in vertikaler Richtung weggelassen, da nach Annahme in diesem Beispiel eine Aktion in der Hüfte keine Änderung bewirkt.

Im Abschnitt 7.2 wurde eine Bedeutung des Begriffs der Situation in der Psychologie vorgestellt. Dieser hatte im Kern das aktuelle Weltbild eines Menschen als Situation definiert. Stellt man den Semni ins Zentrum dieser Definition, kann eine Situation auch nach seinem *Weltbild* definiert werden. Beim Semni bildet dieses der sensomotorische Graph, der durch das ABC-Learning gelernt wird. Ausgehend von diesem wurde der zuvor beschriebene Situationsbegriff deklariert. Zur Unterscheidung wird der aktuelle komplett explorierte sensomotorische Graph einer Situation als *innere Situation* und der allgemeine von außen beobachtbare Zustand als *äußere Situation* definiert. Zur Verdeutlichung sind in Abbildung 50 für verschiedene Situationen die jeweils äußere und innere Situation gegenüber gestellt. Die dargestellten äußeren Situationen lassen sich zunächst alle dadurch beschreiben, dass die Hüfte fest an

den Kopf des Semni gebunden ist. Zur Vereinfachung wird angenommen, dass jede Umschaltung in der Hüfte dazu führt, dass in der Posture verharrt wird. Außerdem wird hier davon ausgegangen, dass die Hüfte direkt mit dem Stall-Modus gestartet wurde. Die obere äußere Situation unterscheidet sich von den anderen zwei äußeren Situationen dadurch, dass es hier keine Begrenzung von oben gibt. Die zugehörige innere Situation ist dadurch gekennzeichnet, dass für das Knie je zwei Postures im Stall- und Release-Modus, sowie eine Posture im Contraction-Modus existieren. Des Weiteren führt eine Umschaltung des Knies in den Anschlags-Postures jeweils zur Release-Posture, wobei einmal in positive und einmal in negative Richtung umgeschaltet werden muss. Eine weitere Umschaltung in die jeweils gleiche Richtung führt zur Contraction-Posture. Alle Umschaltungen sind symmetrisch. Zu jeder Aktion von einer Posture in eine andere gibt es auch eine Inverse. Die mittlere äußere Situation aus Abbildung 50 unterscheidet sich indessen dadurch, dass die Kniebewegung von oben durch ein Hindernis beschränkt wird. Dabei befindet sich das Knie auf der linken Seite des Kopfes. Die passende innere Situation hat mit der vorherigen inneren Situation eine Anschlags- und eine Release-Posture, sowie die Übergänge zwischen diesen gemein. Weiterhin ist das Verhalten der Aktionen in der Hüfte und die Symmetrie aller Übergänge gleich. Es fehlen jedoch die andere Anschlags- und Release-Posture, sowie die Contraction-Posture inklusiver dazugehörigen Verbindungen. Hinzugekommen ist eine neue Anschlagsposture. Diese ist über die verbliebene Release-Posture in positive Richtung erreichbar. Die letzte äußere Situation ist ähnlich zu der vorherigen, nur ist hier das Knie rechts vom Kopf. Die innere Situation sieht der vorherigen zwar ähnlich, hat jedoch letztlich nichts mit dieser gemeinsam. Keine Posture und somit keine Verbindung lässt sich in der jeweils Anderen wiederfinden. Zur ersten inneren Situation verhält sich diese genauso, wie die zweite innere Situation sich zur ersten verhalten hat. In diesem Fall sind die jeweils andere Anschlags- und Release-Posture gleich. Die neue Stall-Posture ist über die Release-Posture in negativer Richtung zu erreichen.

Wird der allgemeine Zustand der äußeren Situation in einer Art und Weise verändert, dass sich die Fixpunkte des Semni nicht ändern, ist auch kein Unterschied in der inneren Situation zu beobachten. Zu sehen ist dies in Abbildung 51. Die grundlegende äußere Situation wurde aus dem vorherigen Beispiel übernommen. Statt den Semni durch eine Decke zu begrenzen, wurde er in eine Box gesteckt. Diese ist jedoch so groß, dass sich das Knie frei bewegen kann. Da die Hüfte weiterhin fixiert ist, lässt sich für den Semni diese Änderung nicht wahrnehmen. Die innere Situation bleibt die gleiche. Eine äußere Situation wird nur dann als eine andere Situation

bezeichnet, wenn durch die Änderung ein Einfluss auf die innere Situation entsteht, da diese ebenfalls auf den Fixpunkten beruht. Wäre dem nicht so, könnte man nicht mehr von der inneren auf die äußere Situation schließen, da diese sich nicht mehr unterscheiden würden.

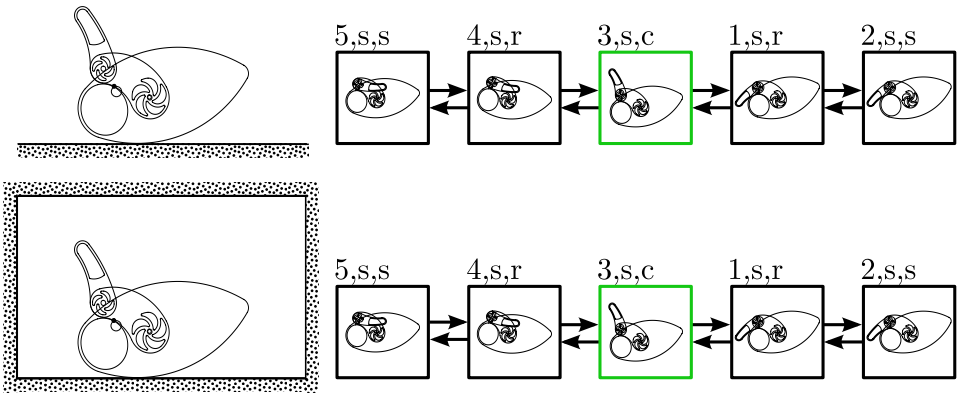


Abbildung 51: Dargestellt ist eine Änderung der äußeren Gegebenheiten, die keinen Einfluss auf die Bewegungsmöglichkeiten des Semni hat. Die innere Situation bleibt unverändert.

Anhand der vorangegangenen Beispiele wurde verdeutlicht, welchen Einfluss eine Änderung der äußeren Situation auf die innere Situation hat. Während grobe Änderungen in der äußeren Situation, etwa das Einziehen einer beschränkenden Decke, einen relativ kleinen Effekt auf die innere Situation haben können, bewirkt die bloße Änderung der Lage des Knies eine komplette Änderung der inneren Situation.

7.4 Verfahren zum Schließen von der inneren auf die äußere Situation

Die Grundidee der auf dem sensomotorischen Graphen basierenden Situationserkennung ist es, von der inneren auf die äußere Situation zu schließen. Nachdem die verschiedenen Einflüsse der äußeren Situationen auf die innere Situation dargelegt wurden, wird nun das Verfahren beschrieben, dass aus diesen Einflüssen die äußere Situation schlussfolgert. Dazu gehören zwei Phasen: Die erste Phase besteht darin, die verschiedenen Situationen zu erlernen. Als erstes soll detektiert werden, dass sich die äußere Situation verändert hat. Anschließend wird die neue Situation erlernt. Die zweite Phase besteht in dem Wiedererkennen verschiedener Situationen. Auch hier soll zuerst erkannt werden, dass sich die äußere Situation geändert hat. Abschließend wird der neuen inneren Situation die entsprechende äußere Situation zugeordnet.

7.4.1 Erlernen verschiedener Situationen

Bevor eine Situation erkannt wird, muss diese zunächst erlernt werden. Beim Lernen bieten sich grundsätzlich zwei verschiedene Ansätze: Der erste Ansatz ist das überwachte Lernen. Nach Mohri u. a. (2012) bezeichnet dieses alle Lernverfahren, bei denen die später zu klassifizierenden Daten bereits kategorisiert und entsprechend markiert sind. Das bedeutet, dass von den bereits vorsortierten Daten abstrahiert wird, um später andere Daten in die gleichen Kategorien einordnen zu können. Ein Vertreter dieser Lernverfahren sind Entscheidungsbäume (Larose 2014). In diesem Fall bedeutet dies, dem Verfahren werden die verschiedenen Situationen vorgegeben. In einfachster Form gibt man die vorkategorisierte Datenstruktur in Form einer Datei in das Lernverfahren ein. Für etwas mehr Autonomie könnte man den Semni während der Exploration die ganze Zeit darüber informieren, in welcher Situation er sich gerade befindet. So kann er die Datenstruktur selber mit der Kategorisierung anreichern, die ihm von außen gegeben wird. Lernen würde in beiden Fällen bedeuten, den sensomotorischen Graphen aufzubauen, um die Informationen zu den Situationen zu erweitern und dabei von den verschiedenen Situationen zu abstrahieren, indem beispielsweise gleiche Postures und Verbindungen nur einmal gelernt werden.

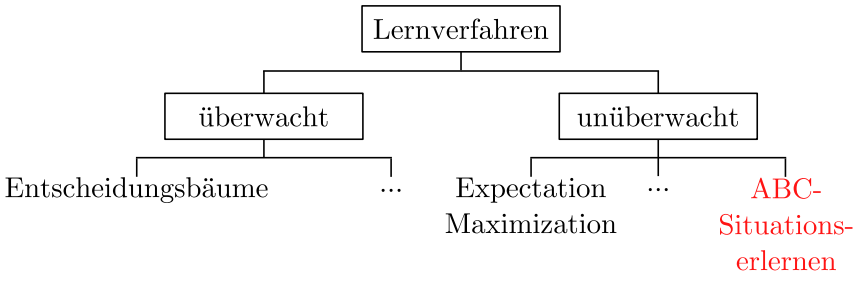


Abbildung 52: Abgebildet ist eine Ontologie von Lernverfahren. Diese untergliedern sich in überwachte und unüberwachte Verfahren. Ein Vertreter von überwachten Lernverfahren sind Entscheidungsbäume. Expectation Maximization ist wiederum ein unüberwachtes Verfahren. Das Lernverfahren zum Erlernen der Situationen, hier ABC-Situationserlernen genannt, ist ebenfalls ein unüberwachtes Lernverfahren.

Im Gegensatz dazu kommt das unüberwachte Lernen laut Mohri u. a. (2012) ohne vormarkierte Daten aus. Als Folge davon baut das Lernverfahren selber Kategorien auf, denen er dann die Daten zuordnet. Klassischerweise handelt es sich dabei um Clustering-Verfahren wie dem in Dempster u. a. (1977) beschriebenen Expectation Maximization. Für das Erlernen der Situationen würde das bedeuten, dass man den Semni frei explorieren lässt und dabei immer mal wieder die Situation ändert, ohne

ihm weitere Informationen zu geben. Er müsste selbständig erkennen, dass sich die Situation verändert hat. Für diese neue Situation legt er dann eine eigene Kategorie an und grenzt sie von den anderen ab.

Der Vorteil des überwachten Lernens besteht darin, dass es das Erlernen wesentlich vereinfacht, da man nur von den vorgegebenen Kategorien abstrahieren muss, um später unbekannte Daten gut zuordnen zu können. Dazu wurden viele verschiedene Methoden veröffentlicht, die sehr gute Resultate erzielen können. Der Nachteil liegt darin, dass kategorisierte Daten gebraucht werden. Dieser Schritt kann unter Umständen sehr aufwendig sein, da er oft nur durch Menschen zu erledigen ist. Beim unüberwachten Lernen verkehren sich die Nachteile zu Vorteilen und vice versa. Die Verfahren werden komplizierter und deswegen eventuell weniger effizient. Dafür benötigt man keine Vorsortierung der Daten mehr. Für die Robotik bringt das unüberwachte Lernen mehr Autonomie der Roboter, da sie nicht mehr abhängig von den vom Menschen bereitgestellten Daten sind. Für die Zwecke einer Exploration sollte man auch ein unüberwachtes Lernen anstreben. Würde man dem Roboter Daten vorgeben, könnte man nicht mehr von einer „Erkundung“, einem in „Erfahrung bringen“ oder „Ausprobieren“ sprechen. Daher versucht das hier gewählte Verfahren ohne vorsortierte Daten auszukommen.

Um ein unüberwachtes Lernen zu realisieren, müssen einige Voraussetzungen angenommen werden. Zunächst wird davon ausgegangen, dass in einer äußeren Situation der sensomotorische Graph immer gleich ist. Außerdem wird vorausgesetzt, dass jede Aktion in einer Posture innerhalb einer äußeren Situation immer den gleichen Übergang produziert. Ferner bedeutet dies, dass für diese eine äußere Situation von jeder Posture genau vier Kanten ausgehen. Damit gilt ein Graph einer Situation als komplett exploriert, sobald die Anzahl der Kanten viermal so groß wie die Anzahl der Postures ist, bzw. jede der vier Möglichkeiten in jeder Posture ausprobiert wurde. Zuletzt wird verlangt, dass in jeder neuen äußeren Situation der Semni den kompletten sensomotorischen Graphen explorieren kann, bevor wieder die Situation gewechselt wird.
Am Anfang hat der Semni überhaupt kein Weltbild. Die Situation nach dem Einschalten ist somit eine neue. Nach Vorraussetzung exploriert er diese mittels ABC-Learning einmal komplett und wählt dann zufällig die nächsten Aktionen aus. Dieser sensomotorische Graph wird als die erste innere Situation gelernt. Danach wird die äußere Situation geändert. Abhängig von der Art der Änderung und der aktuellen Posture, wird es dazu kommen, dass eine der zufällig gewählten Aktionen einen an-

deren Übergang produziert, als er in der ersten inneren Situation erwartet wird. Basierend auf der Annahme, dass innerhalb einer Situation jede Aktion immer wieder zu dem gleichen Übergang führt, kann man an dieser Stelle davon ausgehen, dass sich die äußere Situation geändert haben muss. Daraufhin wird wieder der komplette sensomotorische Graph exploriert und als eine neue innere Situation abgelegt. Dieser grobe Ablauf wird in Abbilung 53 dargestellt.

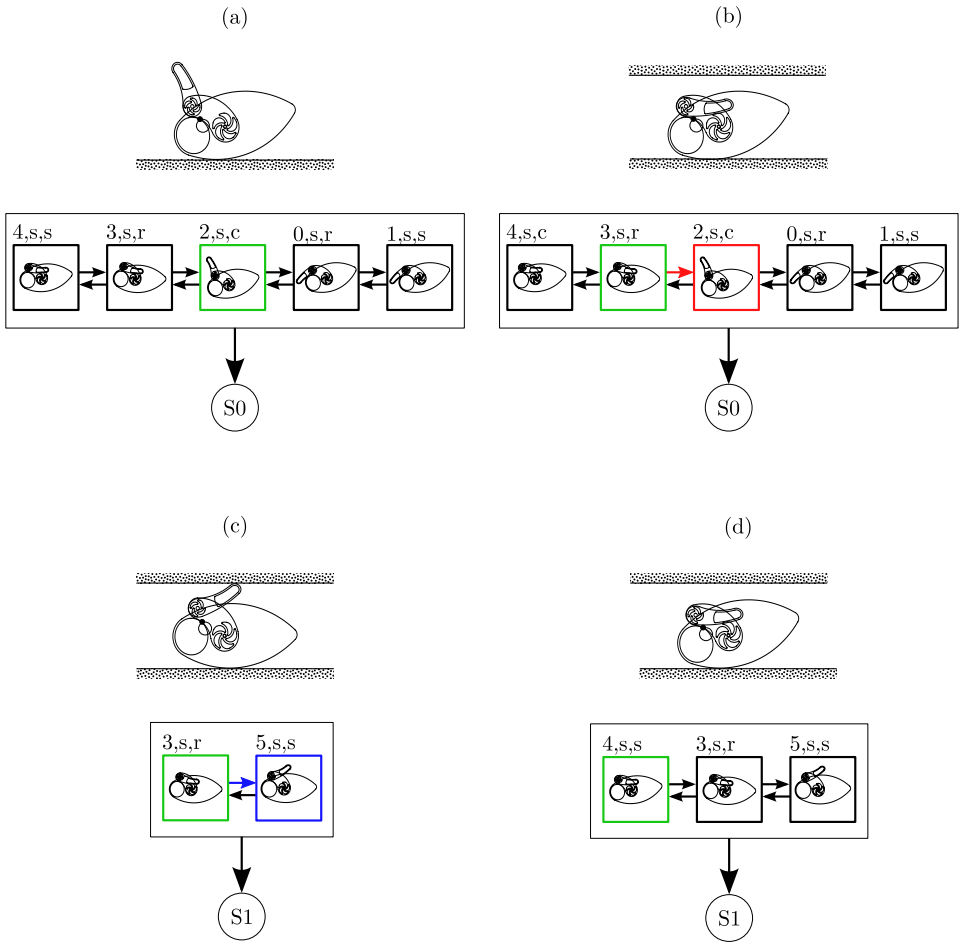


Abbildung 53: Dargestellt ist ein Beispiel, wie eine Situtation erlernt wird. Die erste Situation ist komplett exploriert (a). Danach wird die Situation geändert (b). Nachdem ein unerwarteter Übergang erfolgt, wird eine neue innere Situation gelernt (c) und komplett exploriert (d).

Wieder werden die Situationen betrachtet, in denen die Hüfte fixiert und im Stall-Modus ist, während das Knie sich frei bewegen kann, bzw. durch eine Decke beschränkt wird. Ausgehend von der ersten äußeren Situation links oben, wird der sensomotorische Graph als erste Situation S_1 gelernt. In der Darstellung rechts davon ist eine Decke eingezogen und somit die äußere Situation geändert. In grün ist die

aktuelle Posture markiert und in rot der erwartete Übergang inklusive zugehöriger Posture gekennzeichnet, die von der Aktion „Knie in positive Richtung umschalten“ erwartet wird. Wie anhand der äußeren Situation zu erwarten war, ist der Abbildung unten links zu entnehmen, dass etwas anderes geschehen ist. Es gab, ausgehend von der vorherigen Posture, einen neuen Übergang in eine neue Posture, die blau gekennzeichnet ist. Daraufhin werden diese und alle weiteren Postures und Verbindungen einer neuen Situation S_2 zugeordnet, bis wieder ein vollständiger Graph entstanden ist, wie es unten rechts zu sehen ist.

In dieser Art und Weise können nun beliebig viele Situationen erlernt werden, indem immer dann der komplette sensomotorische Graph exploriert und als neue innere Situation gelernt wird, wenn ein Übergang passiert, der in keiner der bereits bekannten inneren Situationen vorhanden ist. Die Menge der gelernten Situationen wird im Folgenden mit S bezeichnet:

$$S = \{S_1, S_2, \cdots, S_k\} \text{ , mit k gelernten Situationen}$$

Der Vorteil dieses Verfahrens ist es, dass der Semni komplett autonom agiert und keine zusätzlichen Informationen von außen benötigt. Ein Nachteile sind die mehr oder weniger starken Annahmen die hierfür gemacht wurden. Außerdem kann es sein, dass eine Situation sich genau als ein Schnitt verschiedener anderer Situationen darstellt und somit nicht als neue Situation erkannt wird, da sie keine unbekannten Übergänge enthält. Solch eine Situation würde, abhängig von den letzten Übergängen, von dem später beschriebenen Kategorisierungsverfahren abwechselnd immer als einer der zum Schnitt gehörenden Situationen deklariert werden. Die meisten Annahmen und auch das Problem solcher „Schnitt-Situationen“ lässt sich dadurch lösen, dass man zu einem überwachten Lernverfahren wechselt. Wenn man dem Semni vorgibt, wann eine Situation beginnt und wann sie endet, kann eine Aktion auch mehrere Übergänge haben, da klar ist, dass alle zu dieser einen vorgegebenen Situation gehören. Es muss nicht komplett exploriert werden, da auch Teile eines Graphen von außen als eine Situation kategorisiert werden können. Abschließend würden Schnitt-Situationen beim Erlernen nicht durch andere Situationen verdeckt werden, da sie von außen als eigene Situation deklariert werden.

7.4.2 Diffusionsprozess zur Aktivierung der gelernten inneren Situationen

Ziel ist es, von der inneren Situation auf die äußere zu schließen, um so die aktuelle Situation zu erkennen. Dazu soll bei mehreren gelernten Situationen eine als die aktuelle bestimmt werden. Es wird ein Diffusionsprozess definiert, der durch das Ausführen von Aktionen den verschiedenen inneren Situationen sogenannte Aktivierungen zuordnet. Mit jeder Aktion diffundieren Informationen und bilden die Aktivierungen der Situationen. In diesem Fall sendet eine Verbindung des sensomotorischen Graphen eine Aktivierung an die jeweilige Situation. Die Situationen wiederum verarbeiten diese Werte und bilden ihrerseits eine Aktivierung, die rekurrent Einfluss auf die folgenden Aktivierungen hat. Anhand der Aktivierungen wird später bestimmt, welche Situation zur Zeit vorliegt.

An dieser Stelle wird näher auf die Art der Zuordnung der sensomotorischen Graphen zu den Situationen eingegangen. Es werden nicht mehrere sensomotorische Graphen sondern nur ein großer Graph gebildet. In diesem wird von jedem Übergang eine Verbindung zu den Situationen gezogen, in denen dieser Übergang vorkommt. Verdeutlicht wird dies an dem Situationsbeispiel aus Abbildung 53. Dort wurde der komplette sensomotorische Graph als Ganzes der Situation zugeordnet. In Abbildung 54 ist diese Zuordnung für einige Verbindungen detaillierter dargestellt.

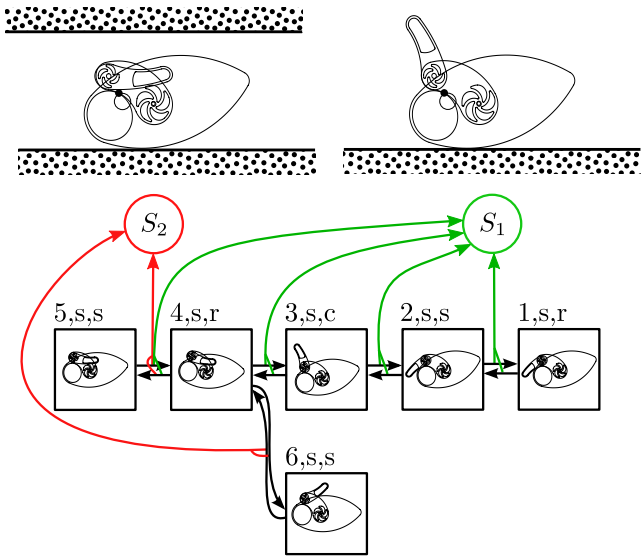


Abbildung 54: Dargestellt ist die Zuordnung der Teilgraphen zu den Situationen. Beispielhaft sind von einigen Übergängen Pfeile zu den mit diesen assoziierten Situation gezogen. Manche Übergänge sind nur einer Situation zugeordnet. Es gibt auch Übergänge, die beiden Situationen zugeordnet sind, wovon hier nur eine als Beispiel gezeigt wird.

Links und rechts sind nochmal die zwei verschiedenen äußeren Situationen abgebildet. Darunter ist der komplette sensomotorische Graph zu sehen, wobei an jedem Übergang, der die Posture ändert, eine Verbindung zur jeweiligen Situation angefügt ist. Der Teil des Graphen der durch die Postures p_1 bis p_4 und den Übergängen zwischen diesen gebildet wird, hat nur Verbindungen zur ersten Situation S_1. Die Übergänge zwischen p_4 und p_5 sind beiden Situationen zugeordnet. Für p_4 gibt es zwei verschiedene Übergänge für die Aktion, in der das Knie in positive Richtung umgeschaltet wird. Die beiden Übergänge, und von dort aus auch alle weiteren, gehören entweder zu der einen oder der anderen Situation. Blendet man alle Übergänge, die nur eine grüne Verbindung haben aus, und entfernt anschließend alle Postures, die nur noch Übergänge zu sich selbst oder keine Übergänge mehr haben, so erhält man wieder den Graphen von der rechten Situation. Vice versa gilt dies für die roten Verbindungen und den Graphen der linken Situation. Für diese Teilgraphen sind alle Übergänge dem jeweiligen Situationsknoten zugeordnet, weshalb in Abbildung 53 der gesamte Graph der Situation zugeordnet wurde. Die Menge aller Situationen, die einem Übergang e_{ij} zugeordnet ist wird im Folgenden mit Ψ_{ij} bezeichnet.

Ausgehend von diesem Gerüst wird erklärt, wie die einzelnen Aktivierungen der Situationen entstehen. Angenommen, der Graph der komplett explorierten Situationen liegt vor, so führt der Semni zu diesem Zeitpunkt zufällige Aktionen aus.

Dadurch werden Übergänge passiert, die zur aktuellen Situation gehören. Mit jedem Mal senden diese eine Aktivierung, im Folgenden mit $a_{e_{ij}}$ für einen Übergang e_{ij} bezeichnet, an alle zugeordneten Situationen. Diese berechnet sich folgendermaßen:

$$a_{e_{ij}} = \frac{\beta}{|\Psi_{ij}|}$$

Die *Aktivierungsrate* β kann einen beliebigen Wert größer als 0 annehmen. Mit diesem Parameter kann gesteuert werden, wie schnell die Aktivierung der zugeordneten Situationen wächst. Je größer dieser Parameter ist, desto schneller wachsen die Aktivierungen der entsprechenden Situationen. In dieser Arbeit wurde der Parameter auf Eins gesetzt. Situationen, die in einem Schritt keine Aktivierung erhalten, setzen ihre eigene Aktivierung auf Null, da diese Situationen nicht vorliegen können, weil der aktuelle Übergang nicht in ihnen vorkommt. Die anderen Situationen addieren die Aktivierung des aktuellen Übergangs auf ihre eigene Aktivierung. Anschließend werden alle Situationsaktivierungen, mit der Summe all dieser normalisiert, sodass sich diese am Ende wieder zu Eins aufaddieren. Für eine Situation S_k, die mit dem aktuellen Übergang e_{ij} assoziiert ist, lässt sich die Aktivierung a_{S_k} folgendermaßen berechnen:

$$a_{S_k}(t) = \begin{cases} \frac{a_{S_k}(t-1)+a_{e_{ij}}}{\beta+\sum\limits_{S_l \in \Psi_{ij}} a_{S_l}(t-1)} & \text{falls } S_k \in \Psi_{ij} \\ 0 & \text{sonst} \end{cases}$$

Gesucht ist ein Vektor $A(t)$, der als Einträge alle aktuellen Aktivierungen der gelernten Situationen enthält:

$$A(t) = (a_{S_1}(t), a_{S_2}(t), \cdots, a_{S_{|S|}}(t))$$

Nach dem Einschalten wird dieser Vektor so initialisiert, dass alle Situationen $S_k \in I$, die einem Übergang von oder zu der vorliegenden Posture p_i zugeordnet sind, eine gleich große Aktivierung $a_I(S_k) : S \to \mathbb{R}$ haben, die sich zu Eins aufsummieren lässt:

$$I = (\bigcup_{p_j \in P} \Psi_{ij}) \bigcup (\bigcup_{p_j \in P} \Psi_{ji})$$

$$a_I(S_k) = \begin{cases} \frac{1}{|I|} & \text{falls } S_k \in I \\ 0 & \text{sonst} \end{cases}$$

$$A(0) = (a_I(S_1), a_I(S_2), \cdots, a_I(S_{|S|}))$$

Die Übergangsmatrix U hat die Aktivierung $a_{e_{ij}}$ am zuletzt getätigten Übergang e_{ij} und ist an allen anderen Stellen Null:

$$U = \begin{pmatrix} u_{11} & \cdots & u_{1|P|} \\ \vdots & \ddots & \vdots \\ u_{|P|1} & \cdots & u_{|P||P|} \end{pmatrix}$$

$$u_{lk} = \begin{cases} a_{e_{ij}} & \text{falls } l = i \ \wedge \ k = j \\ 0 & \text{sonst} \end{cases}$$

Für die Aktualisierungsregel werden außerdem folgende Definitionen gebraucht:

$$s = (1, \cdots, 1)^T \text{ mit } s \in \mathbb{R}^{|P|}$$

$$f_{\Psi_{ij}}(S_k) = \begin{cases} 1 & \text{falls } S_k \in \Psi_{ij} \\ 0 & \text{sonst} \end{cases} \quad \text{mit } f_{\Psi_{ij}} : S \to \{0, 1\}$$

$$\vec{\Psi}_{ij} = (f_{\Psi_{ij}}(S_1), f_{\Psi_{ij}}(S_2), \cdots, f_{\Psi_{ij}}(S_{|S|}))$$

$$\text{diag}(v) = \begin{pmatrix} v_1 & 0 & \cdots & 0 \\ 0 & v_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & v_o \end{pmatrix} \text{ mit diag} : \mathbb{R}^o \to \mathbb{R}^o \times \mathbb{R}^o$$

$$\text{sum}(v) = \sum_{q=1}^{o} v_q \text{ mit sum} : \mathbb{R}^o \to \mathbb{R}$$

$$\text{norm}(v) = (\frac{v_1}{\text{sum}(v)}, \cdots, \frac{v_o}{\text{sum}(v)}) \text{ mit norm} : \mathbb{R}^o \to \mathbb{R}^o$$

Mit Hilfe dieser Definitionen kann die Aktualisierungsregel nach jedem Übergang e_{ij} definiert werden als:

$$A(t+1) = \text{norm}((Us)^T s \vec{\Psi}_{ij} + A(t)\text{diag}(\vec{\Psi}_{ij}))$$

Dieser Diffusionsprozess führt dazu, dass Situationen immer mal wieder in ihrer

Aktivierung wachsen können, aber niemals größer als die Aktivierung der Situation werden können, die gerade vorliegt. Diese ist immer aktiviert, wohingegen alle anderen Situationen immer dann deaktiviert werden, sobald zumindest der eine jeweilige Übergang passiert wird, der die augenblickliche Situation von den anderen unterscheidet. Nach solch einem Übergang fallen alle Situationen auf 0 zurück, die diesem nicht zugeordnet sind, und die zugehörigen Situationen bilden ihre Aktivierungen wie beschrieben. Bei einer Folge von mehrdeutigen Übergängen für die immer gleichen Situationen wächst die Unsicherheit an, da der letzte zwischen diesen eindeutige Übergang immer weiter zurückliegt und an Bedeutung verliert. Dies drückt sich darin aus, dass diese Situationen sich in ihrer Aktivierung langsam annähern. Dies soll abschließend an einem Beispiel vorgeführt werden. Dazu betrachte man den sensomotorischen Graph aus Abbildung 55.

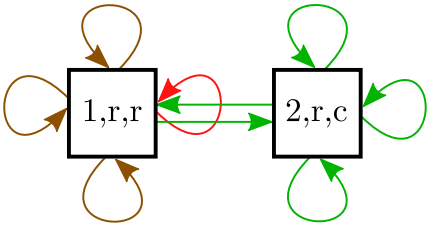


Abbildung 55: Abgebildet ist ein einfaches Beispiel eines sensomotorischen Graphen mit verschiedenen Situationen. Zu sehen sind zwei Postures und verschiedene Verbindungen. Rote und grüne Übergänge gehören jeweils zu verschiedenen Situationen, während braune Übergänge zu beiden Situationen gehören.

Gestartet wird in p_1 und die Aktivierung der roten und grünen Situation sei jeweils 0,5. Ausgehend davon zeigt die Tabelle 3 die verschiedenen Verläufe der Aktivierungen. In der ersten Spalte steht die jeweils aktuelle Posture. Die nächste Spalte zeigt die Aktion an, die von der aktuellen Posture aus ausgeführt wird. Dabei folgen die Pfeile der Semantik der Übergänge aus der Abbildung. Danach folgt die Spalte mit der aktuellen Situation. Diese ist entweder rot oder grün entsprechend den Farben aus dem zugehörigen sensomotorischen Graphen. In der dritten und vierten Spalte findet sich die Aktivierung des durch die Aktion resultierenden Übergangs für die jeweilige Situation. Die letzten beiden Spalten zeigen für jede Situation die Aktivierung nach der Aktion und dem beschriebenen Diffusionsprozess.

Posture	Aktion	Situation	Akt. durch Übergang		Akt. Situation	
			rot	grün	rot	grün
1	→	rot	1,00	0,00	1,00	0,00
1	←	rot	0,50	0,50	0,75	0,25
1	↑	rot	0,50	0,50	0,63	0,38
1	↓	grün	0,50	0,50	0,56	0,44
1	→	grün	0,00	1,00	0,00	1,00
2	←	grün	0,00	1,00	0,00	1,00
1	↑	grün	0,50	0,50	0,25	0,75
1	→	grün	0,00	1,00	0,00	1,00
2	←	grün	0,00	1,00	0,00	1,00
1	→	rot	1,00	0,00	1,00	0,00

Tabelle 3: Dargestellt ist eine Übersicht über den Verlauf der Aktivierungen bei einem beispielhaften Durchlauf durch die Graphen der verschiedenen Situationen.

7.4.3 Auswahlverfahren der aktuellen Situation

Nachdem beschrieben wurde, wie verschiedene Situationen gelernt werden und diese durch das Ausführen von Aktionen Aktivierungen bilden, soll der Schritt zur Schlussfolgerung von der inneren auf die äußere Situation erläutert werden. Nach einer Aktion berechnet jede der gelernten inneren Situationen eine Aktivierung. Je größer diese Aktivierung ist, desto besser passt die innere Situation zur äußeren. Nach jedem Schritt wird die aktuelle Situation danach bestimmt, welche innere Situation die größte Aktivierung hat. Das bedeutet, dass davon ausgegangen wird, dass die aktuelle äußere Situation der äußeren Situation entspricht, welche vorlag, als die innere Situation mit der derzeit größten Aktivierung gelernt wurde. Die Situation $S_y(t)$, die als aktuell vorliegend angenommen wird, bestimmt sich somit folgendermaßen:

$$\begin{aligned} \text{maxId}(v) =& i \text{ mit } \text{maxId}(v) : \mathbb{R}^o \to \mathbb{N} \\ & \text{wobei } \forall\, 1 \leq i, j \leq o \;:\; v_i > v_j \;\vee\; v_i = v_j \wedge\; i \leq j \;,\; i, j \in \mathbb{N} \end{aligned}$$

$$S_y(t) = S_{\text{maxId}(A(t))}$$

Wenn ausreichend Übergänge folgen, die zumindest immer zu zwei inneren Situationen gehören können, kann es auf Grund der Genauigkeit der Rechner dazu kommen, dass zwei oder mehr innere Situationen exakt die gleiche Aktivierung haben. In dem Fall wird unter diesen Situationen weiterhin diejenige ausgewählt, die zuletzt als einzige die höchste Aktivierung hatte. Es wird angenommen, dass Situationswech-

sel selten geschehen und deswegen bei der letzten Entscheidung verharrt, bis diese eindeutig falsifiziert wird.

7.5 Implementierung eines Sicherungsverhaltens

Der zuletzt beschriebene Diffusionsprozess führt dazu, dass mit der Zeit Situationen, die durchgängig mit den letzten Übergängen assoziiert sind, eine wachsende Aktivierung erhalten. Es bildet sich langsam eine Gleichverteilung der Aktivierungen auf alle diese Situationen. Dies drückt eine Unsicherheit über die aktuelle Situation aus. Führt man nun lediglich zufällig die nächsten Aktionen aus, kann es sein, dass es sehr lange dauert, bis sich die Aktivierungen eindeutiger verteilen und mit Sicherheit eine Aussage über die aktuelle Situation getroffen werden kann. Deshalb wurde ein Verfahren implementiert, das ein Sicherungsverhalten gewährleisten soll. Bei ausreichend großer Unsicherheit wird gezielt versucht, Aktionen auszuwählen, die diese Unsicherheit beseitigen. Es wird bestimmt, wann dieses Verfahren beginnen soll, und eine Strategie gewählt, die die Unsicherheit wieder verringert.

7.5.1 Kriterium zum Starten des Sicherungsverhaltens

Die einfachste Methode zur Bestimmung des Starts des Sicherungsverhaltens ist es, die zwei Situationen mit der höchsten Aktivierung zu betrachten. Sobald beim zufälligen Wählen von Aktionen der Betrag der Differenz der beiden Situationen mit der höchsten Aktivierung unter einen bestimmten Grenzwert γ fällt, wird das im Folgenden beschriebene Verfahren gestartet. Formal ausgedrückt wird dieses aufgerufen, wenn gilt:

$$|\max(A(t)) - \max2(A(t))| < \gamma$$

Dabei gibt $\max : \mathbb{R}^o \to \mathbb{R}$ den größten und $\max2 : \mathbb{R}^o \to \mathbb{R}$ den zweitgrößten Wert eines Vektors $v \in \mathbb{R}^o$ zurück. Dieses Kriterium passt sehr gut zu dem Verfahren des Sicherungsverhaltens, da dieses ebenfalls auf Basis der zwei Situationen mit der höchsten Aktivierung arbeitet. Es wäre auch möglich, andere beliebig komplexe Kriterien bzw. Unsicherheitsmaße einzuführen.

7.5.2 Verfahren des Sicherungsverhaltens

Das daraufhin gestartete Verfahren hat die Aufgabe, die Unsicherheit zu mindern. Es wird dafür eine sogenannte *Greedy-Strategie* gewählt. Greedy kommt aus dem Englischen und bedeutet „gierig“. Nach Cormen u. a. (2001) zeichnen sich solche Verfahren dadurch aus, dass sie als nächste Option immer die lokal Beste auswählen, um so ein globales Optimum zu erreichen. Es handelt sich dabei um eine Heuristik,

die nicht immer ein globales Optimum erreicht, sondern auch in lokalen Optima verharren kann. Trotzdem sind sie in einigen Anwendungsszenarien dazu geeignet, eine gute Lösung zu finden. Ein Beispiel für ein Greedy-Verfahren sind Gradientenverfahren. Das Sicherungsverhaltens ist „greedy“, weil versucht wird, die aktuell größte Unsicherheit durch die lokal am nächsten gelegene Aktion auszuräumen, die sich zur Beseitigung dieser Unsicherheit eignet. Ein Ansatz der nicht zu den Greedy-Verfahren gehören würde, könnte beispielsweise die Unsicherheiten zwischen allen Situationen berücksichtigen und in Hinblick darauf versuchen einen Pfad im Graphen zu finden, der nicht nur die aktuell größte Unsicherheit beseitigt, sondern im Ergebnis eine maximale Aktivierung einer Situation produziert, deren Differenz mit allen anderen Aktivierungen oberhalb von γ liegt. Falls möglich wäre dies jedoch wesentlich rechenintensiver als das im Folgenden beschriebene Verfahren.
Gibt es neben der maximalen Aktivierung eine Situation mit einer ähnlich hohen Aktivierung, führt dies zu der größten Unsicherheit über die aktuelle Situation, da diese nicht eindeutig hervorsticht. Um die Unsicherheit zwischen diesen beiden inneren Situationen zu beseitigen, muss eine Aktion gefunden werden, die für beide Situationen unterschiedliche Übergänge zur Folge hat. Damit diese nicht unnötig weiter anwächst und möglichst schnell beseitigt werden kann, wird die nächstgelegene Aktion gewählt, die diese Bedingung erfüllt. Das bedeutet, dass sie, von der aktuellen Posture aus, mit einer minimalen Anzahl von Aktionen zu erreichen ist. Welche der in Frage kommenden Aktionen am nächsten ist, wird mittels des Algorithmus von Dijkstra bestimmt. In Abbildung 56 ist das Verfahren grafisch dargestellt, wobei $\text{max2Id}(v) : \mathbb{R}^o \rightarrow \mathbb{N}$ den Index des zweitgrößten Werts eines Vektors $v \in \mathbb{R}^o$ zurückgibt.

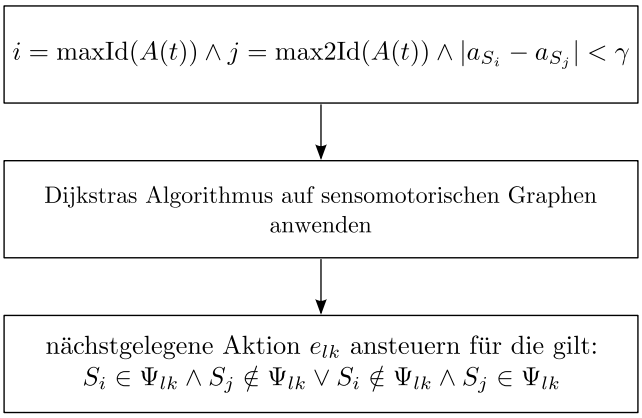


Abbildung 56: Dargestellt ist das Verfahren des Sicherungsverhaltens. Nachdem das Startkriterium für S_i und S_j erfüllt ist, wird zunächst Dijkstras Algorithmus zur Berechnung der kürzesten Pfade ausgehend von der aktuellen Posture zu allen anderen berechnet. Anschließend wird der Übergang e_{lk} angesteuert, dem nur eine der beiden Situationen zugeordnet ist.

Dieses Verfahren wird anhand des sensomotorischen Graphen aus Abbildung 55 und der zugehörigen Tabelle 3 über den Verlauf der Aktivierungen beispielhaft vorgeführt. Am Anfang werden zufällige Aktionen ausgewählt, der Grenzwert zur Aktivierung des Sicherungsverhaltens liegt bei 0,2. Die ersten zufällig gewählten Aktionen führen jeweils zu Übergängen, die mit beiden Situationen assoziiert sind. Nach der vierten Aktion ist der Betrag der Differenz der beiden Situationen unter den Grenzwert gefallen:

$$0{,}56 - 0{,}43 = 0{,}13 < 0{,}2$$

Dies führt dazu, dass nach der nächsten Aktion gesucht wird, die beiden Situationen verschiedene Übergänge zuordnet. Von p_1 aus ist dies die Aktion „→“, da alle anderen Aktionen entweder weiter weg sind oder Übergänge hervorrufen, die beiden Situationen zugeordnet sind. Es wird als nächstes diese Aktion ausgeführt. Die Situation hatte sich bereits vorher geändert. Nach der Aktion ist die Aktivierung der zugehörigen inneren Situation maximal. Die Aktivierungen sind wieder klar verteilt, sodass anschließend zufällige Aktionen gewählt werden. Anschließend unterschreitet der Betrag der Differenz der beiden Situationen den Grenzwert nicht mehr. Es wird im weiteren Verlauf ausschließlich zufällig die nächste Aktion gewählt und das Sicherungsverhalten nicht nochmal angestoßen.

7.6 Erweiterung des ABC-Learning um eine Situationserkennung

In diesem Kapitel soll darauf eingegangen werden wie die aus Abschnitt 6 beschriebene Implementierung des ABC-Learning mit der Situationserkennung aus den vorherigen Abschnitten erweitert wird. Dabei soll zuerst beschrieben werden, wie die

verschiedenen Situationen mit den Übergängen assoziiert wurden. Dann wird ein Kriterium eingeführt, mit dem das ABC-Learning ermittelt, ob alle Situationen voll exploriert sind. Zum Abschluss wird darauf eingegangen, wie das Sicherungsverhalten mit dem ABC-Learning kombiniert wird.

7.6.1 Assoziierung der Übergänge mit den Aktionen

Wie in Abschnitt 6 beschrieben, setzt sich der sensomotorische Graph aus einer Liste von Postures zusammen, die ihrerseits jeweils eine Liste aller ausgehenden Übergänge enthalten. Die Übergänge wurden um eine Liste von assoziierten Situationen erweitert. Sie beinhaltet alle IDs der Situationen, die diesen Übergang enthalten. Die Situationen selber werden in einer eigenen Liste gespeichert. Über die Liste der assoziierten Situationen der Übergänge kann mittels der ID auf diese Situationen zugegriffen werden, um die entsprechenden Aktivierungen der Übergänge auf die Situationen zu übertragen.

7.6.2 Kriterium zur Bestimmung der vollen Exploration aller Situationen

Für das Situationserlernen ist es wichtig zu bestimmen, wann alle Situationen komplett exploriert sind. In Abschnitt 6.2.3 wurde beschrieben, dass die Heuristik zur Wahl der nächsten Aktion solange eine unexplorierte Möglichkeit sucht und ausführt, bis in jeder von der aktuellem Posture p_i erreichbaren Posture, gekennzeichnet durch die Menge P_{p_i} alle Möglichkeiten $|E_{P_{p_i}}$ ausprobiert wurden, also bis gilt:

$$|E_{P_{p_i}}| = 4|P_{p_i}|$$

Für die erste Situation S_1, die gelernt wird, ist dies unproblematisch. Sobald eine zweite Situation S_2 hinzukommt, die mindestens eine Posture p_j gemein hat, wird dies nicht mehr funktionieren. Es besteht die Gefahr, dass man zu früh davon ausgeht, dass alles exploriert wurde. In p_j würde davon ausgegangen werden, dass bereits alle Möglichkeiten exploriert wurden, da dies in S_1 bereits geschehen ist. Dies gilt jedoch nicht für S_2. Das würde dazu führen, dass S_2 eventuell bereits als voll exploriert angenommen wird, obwohl in dieser Posture noch Möglichkeiten offen sind. Das Kriterium wird dahingehend erweitert, dass nur Kanten, die mit der aktuell zu lernenden Situation assoziiert sind, betrachtet werden. Sei P_{S_k} die Menge aller Postures die in Situation S_k vorkommen und von der aktuellen Posture p_i aus erreichbar sind und bezeichne $E_{P_{S_k}}$ alle bisher gelernten Verbindungen dieser

Postures, dann ändert sich das Kriterium zum bestimmen der vollen Exploration zu:

$$\sum_{e_{lm} \in E_{P_{S_k}}} |\Psi_{lm} \bigcap S_k| = 4|P_{S_k}|$$

Des Weiteren muss die Implementierung von Dijkstras Algorithmus, so erweitert werden, dass er nur Kanten benutzt die der zu lernenden Situation zugeordnet sind. Als letztes wird das Problem von Postures betrachtet, die in der ersten aber nicht in der zweiten Situation vorhanden sind. Auf den ersten Blick könnte geschlussfolgert werden, dass es von dort keine Übergänge gibt, die der zu erlernenden Situation S_k zugeordnet sind und somit die Exploration nicht abgeschlossen ist. Da diese Postures in S_k jedoch nicht erreichbar sind, kommen sie nicht in P_{S_k} vor und das Kriterium berücksichtigt diese korrekterweise nicht. In der Implementierung ist dies der Fall, sobald der Dijkstra-Algorithmus keinen Weg mehr zu einer nicht voll explorierten Posture liefert. So lange es noch Postures aus der zu erlernenden Situation gibt, die nicht voll exploriert sind, gibt der Algorithmus einen Weg zu diesen zurück. Sind diese alle exploriert, bleiben nur noch die Postures übrig, die nicht zu der Situation gehören. Da diese aber nicht erreichbar sind, liefert der Algorithmus auch keinen Weg dorthin zurück. Damit eignet sich dieses Ereignis dazu, die Exploration der Situation ab diesem Zeitpunkt als abgeschlossen zu betrachten. Da die Annahme getroffen wurde, dass eine Situation voll exploriert wird, bevor sie sich ändert, sind alle vorherigen Situationen und somit alle Situationen vollständig exploriert.

7.6.3 Integration des Sicherungsverhalten in die Auswahl der nächsten Aktionen

Bis hierhin wurden verschiedene Heuristiken zur Wahl der nächsten Aktion beschrieben und verwendet. Einerseits die in Abschnitt 6.2.3 beschriebene Heuristik zur möglichst schnellen Exploration, andererseits das in Abschnitt 7.5.2 beschriebene Verfahren, um zu differenzieren welche Situation gerade vorliegt. Außerdem wurde an einigen Stellen zufällig die nächste Aktion ausgewählt. In Abbildung 57 wird dargestellt, wie diese Auswahlverfahren miteinander kombiniert werden können. Dazu wird eine Hierarchie der Heuristiken gebildet.

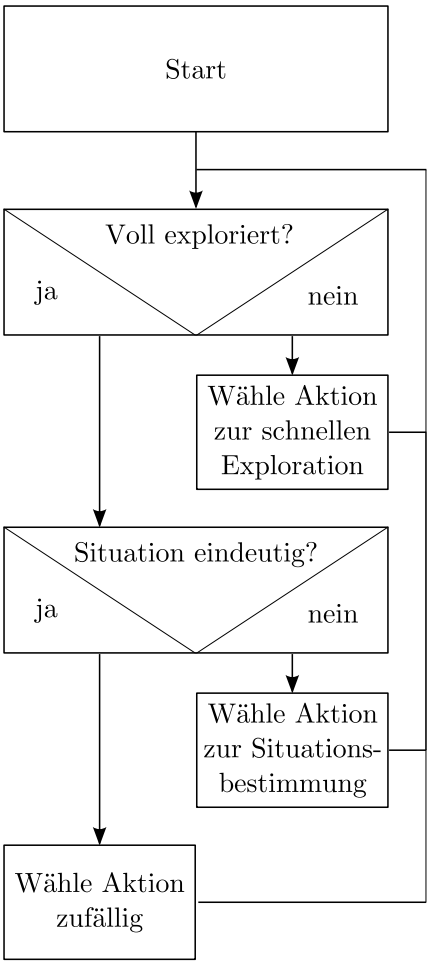


Abbildung 57: Abgebildet ist der Zustandsautomat zur Bestimmung des Verfahrens zur Wahl der nächsten Aktion. Anstelle zufällig zu wählen könnte man den Automaten an dieser Stelle beliebig mit anderen Auswahlverfahren erweitern, die andere Ziele verfolgen.

Muss eine neue Situation gelernt werden, so hat dies Vorrang vor allem Anderen. Solange nicht das in Abschnitt 7.6.2 beschriebene Kriterium zur Bestimmung der vollen Exploration aller Situationen eintritt, wird die nächste Aktion nach der Heuristik zur Wahl der nächsten Aktion zur schnellen Exploration aus Abschnitt 6.2.3 ausgewählt. Danach wird entschieden, ob die nächste Aktion mittels des aus Abschnitt 7.5.2 bekannten Sicherungsverhaltens oder zufällig gewählt wird. Dazu wird das Kriterium zum Starten des Sicherungsverhalten aus Abschnitt 7.5.1 genutzt. Die Evaluation der Situation hat nach der Exploration der Situation Vorrang. Es wird solange die nächste Aktion nach dem in Abschnitt 7.5.2 beschriebenen Sicherungsverhalten ausgewählt, bis das letztgenannte Kriterium nicht mehr zutrifft. Wenn keins der beiden Kriterien einschlägig ist, wird zufällig die nächste Aktion gewählt.

7.7 Komplexität des Verfahrens

In diesem Kapitel soll kurz darauf eingegangen werden, wie rechen- und speicherintensiv das beschriebene Verfahren ist.
Zunächst wird der Speicher mit dem sensomotorischen Graphen gefüllt. Der Speicherbedarf hängt in diesem Fall von der konkreten Implementierung des ABC-Learning ab. Die minimale Anforderung ist, nur den sensomotorischen Graphen abzuspeichern. Dieser setzt sich aus einer ID, den Gelenk- und Körperwinkeln, den CSL-Modi und den Verbindungen zusammen. Mit einer 12-Bit-ID lassen sich 4096 verschiedene Postures identifizieren. Wie sich später zeigt, reicht dies vollkommen aus, da so viele Postures gar nicht erst in den Speicher passen. Die Winkelsensoren des Semni liefern jeweils 10-Bit-Werte. Jede Posture kann somit durch eine 12-Bit-ID, drei 10-Bit-Werte für die Winkel und zwei 2-Bit-Werte[5] für die CSL-Modi beschrieben werden. Damit ergibt sich für jede Posture ein Speicherbedarf von:

$$12\,\text{Bit} + 3 \cdot 10\,\text{Bit} + 2 \cdot 2\,\text{Bit} = 46\,\text{Bit}$$

Zu jeder Posture gehören außerdem vier Verbindungen. Eine Verbindung setzt sich zusammen aus der 12-Bit-Ziel-ID und zwei 2-Bit-Werten[6] zur Beschreibung der Aktion, die zu dieser Verbindung gehört. Damit ergibt sich für jede Posture inklusive der Verbindungen ein Speicherbedarf von:

$$46\,\text{Bit} + 4(12\,\text{Bit} + 2 \cdot 2\,\text{Bit}) = 110\,\text{Bit}$$

Für n Postures werden somit inklusive der Verbindungen $n \cdot 110\,\text{Bit}$ an Speicher benötigt. Legt man die 110 Bit für jede Posture und Verbindungen hintereinander nach ID sortiert in einem Speicherbereich ab, braucht man keinen zusätzlich von der Anzahl der Postures abhängigen Speicher für Zeigerstrukturen, da anhand der ID die Speicheradresse berechnet werden kann. Die Situationserkennung fügt im Worst Case für jede Situation vier neue Verbindungen zu jeder Posture hinzu. Außerdem wird jede Verbindung um die ID einer Situation erweitert. Beschränkt man die Anzahl der Situationen auf 64, so reicht als ID für eine Situation ein 6 Bit Wert aus. Das bedeutet, dass sich der Speicherbedarf für eine Posture inklusive Verbindungen und zugeordneten Situationen folgendermaßen vergrößert:

$$46\,\text{Bit} + 64(64\,\text{Bit} + 4 \cdot 6\,\text{Bit}) = 5678\,\text{Bit}$$

[5] 0 - Release; 1 - Contraction; 2 - Stall+; 3 - Stall-
[6] 0 - Hüfte negativ; 1 - Hüfte positiv; 2 - Knie negativ; 3 - Knie positiv

Als realistischer sollte sich ein Maximum von durchschnittlich acht Verbindungen mit je 8 zugewiesenen Situationen erweisen, wodurch der Speicherbedarf je Posture sich ergibt aus:

$$46\,\text{Bit} + 2(64\,\text{Bit} + 64 \cdot 6\,\text{Bit}) = 942\,\text{Bit}$$

Jede Situation benötigt zusätzlich eine Struktur bestehend aus einer 6-Bit-ID, einer 8-Bit-Aktivierung und einem Bit zur Speicherung, ob diese Situation bei Gleichheit der Aktivierung die Situation ist, die zuletzt die höchste Aktivierung hatte. Bei 64 Situationen ergibt sich zusätzlich ein Speicherbedarf von:

$$64 \cdot (6\,\text{Bit} + 8\,\text{Bit} + 1\,\text{Bit}) = 960\,\text{Bit}$$

Der Semni verfügt über 20 kB Arbeitsspeicher und 128 kB Flashspeicher. Geht man davon aus, dass sonstige Programmvariablen in den Arbeitsspeicher passen, können im Flashspeicher

$$\frac{128\,\text{kB} \cdot 8 - 960\,\text{Bit}}{942\,\text{Bit}} \approx 1112$$

Postures inklusive Verbindungen und Situationszuordnungen gespeichert werden.
Im Unterschied zum Speicherbedarf wirkt sich die Rechenkomplexität des ABC-Learning nicht auf die Rechenintensivität der Situationserkennung aus, weshalb die Komplexität für die Berechnung des ABC-Learning an dieser Stelle nicht betrachtet wird. Bei jedem Übergang wird zur Berechnung der Aktivierung des Übergangs eine Division durchgeführt. Zur Bestimmung des Divisors muss für den Übergang die Anzahl an zugeordneten Situationen bestimmt werden. Dies kann entweder als zusätzlicher Wert in jedem Übergang abgespeichert oder mittels Durchzählen bestimmt werden. Da die Anzahl der zugeordneten Situationen beschränkt ist, kann dies in $\mathcal{O}(1)$, ansonsten in $\mathcal{O}(n)$ bei n Situationen bestimmt werden. Zur Bestimmung der Aktivierung einer Situation wird für jede zugeordnete Situation eine Addition und eine Division ausgeführt. Nicht zugeordnete Situationen werden auf 0 gesetzt. Zur Bestimmung des Divisors werden alle Aktivierungen der Situationen aufsummiert. Ebenfalls gilt hier, dass bei beschränkter Anzahl der Situationen die Aktivierungen in $\mathcal{O}(1)$ und bei unbeschränkt vielen Situationen in $\mathcal{O}(n)$ berechnet werden kann. In der Realität ist allein durch den begrenzten Speicher eine Grenze für die Anzahl an Situationen gegeben, wodurch sich alles in $\mathcal{O}(1)$ berechnen lässt. Da dies für jeden Übergang nur einmal berechnet werden muss und die Übergänge selbst eine gewisse Zeit benötigen, sollte während des nächsten Übergangs genug Zeit sein, die Aktivierungen, die aus dem letzten Übergang resultieren, zu berechnen. Das Sicherungsverhalten, könnte man vermuten, ist so wie bisher beschrieben

etwas rechenintensiver, da es den kompletten Dijkstra-Algorithmus verwendet, welcher in der einfachen Implementierung eine Worst-Case-Laufzeit von $\mathcal{O}(n^2)$ hat, wobei n die Anzahl der Postures darstellt. Dies gilt jedoch nur für Graphen in denen jeder Knoten mit jedem anderen verbunden ist. Da in diesem Fall die Anzahl der Verbindungen auf vier beschränkt ist, reduziert sich die Laufzeit auf $\mathcal{O}(n)$.

8 Experimente zur Evaluation der Situationserkennung

In diesem Kapitel werden die Experimente erläutert, die zur Evaluation der beschriebenen Situationserkennung dienen. Dazu wird der Aufbau und Ablauf der Experimente beschrieben und anschließend auf die Bewertungsmaße eingegangen, nach denen das Verfahren bewertet wird. Zuletzt werden die Versuchsergebnisse vorgestellt und ausgewertet.

8.1 Aufbau und Ablauf der Experimente

Für die Experimente wird der Aufbau des ABC-Learning aus Abschnitt 5 genutzt und erweitert. Ziel ist es einen Aufbau zu schaffen, mit dem ein automatisierter Ablauf realisiert werden kann, um den Zeitaufwand möglichst zu reduzieren und einen ausreichend großen Datensatz zu generieren. Aus diesem Grund wurde der Aufbau als eine Simulation mit echten Daten gestaltet. Zunächst werden echte Daten aufgenommen und später im eigentlichen Versuch wieder abgespielt. Hierzu wurde ein eigenständiges Programm entwickelt, das die echten Daten einliest und als eine Simulation wieder abspielt. Dieses Programm wurde ebenfalls in PureBasic entwickelt, um so möglichst viel Code der Implementierung des ABC-Learning wiederverwenden zu können. Konkret wird der Aufbau und Ablauf in Abbildung 58 illustriert.

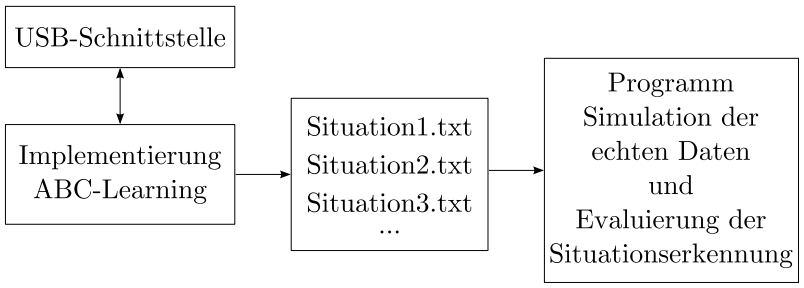


Abbildung 58: Dargestellt ist der Aufbau und Ablauf der Experimente zur Evaluierung der Situationserkennung. Über die USB-Schnittstelle ist die Implementierung des ABC-Learning mit dem Semni verbunden, wie in Abschnitt 5 beschrieben. Der hier gezeigte Aufbau und Ablauf ist aus Sicht der beteiligten Programme und Komponenten.

An der USB-Schnittstelle findet die Kommunikation mit Semni statt. Dies entspricht dem Aufbau aus Abschnitt 5. In dieser Grafik ist Aufbau und Ablauf in Bezug auf die Programme, die auf dem PC ausgeführt werden, abgebildet. Mithilfe des ABC-Learning werden die sensomotorischen Graphen für verschiedene Situationen erstellt und als Textdateien gespeichert. Dazu wird der Semni in eine äußere Situation versetzt und dann das Programm mit einem leeren Graphen gestartet. Danach wird das Programm beendet, der sensomotorische Graph exportiert und wiederum das

Programm mit leerem Graphen gestartet, während der Semni in eine neue äußere Situation gebracht wurde. Auf diese Art und Weise werden die verschiedenen sensomotorischen Graphen für die einzelnen Situationen erstellt. Wurden die Graphen für die zu testenden Situationen generiert, wird das Programm zur Simulation der echten Daten und Evaluation der Situationserkennung gestartet. Dieses liest die sensomotorischen Graphen ein und initialisiert die Visualisierungen sowie das ABC-Learning, welches um die Situationserkennung erweitert wurde. Anschließend wird zufällig gewählt, in welcher Situation das Experiment gestartet werden soll. Danach startet das Experiment. Das Programm wählt als Startzustand $x(0)$ des simulierten Semni die erste Posture des gewählten Graphen. Das ABC-Learning erhält diesen und speichert ihn als erste Posture ab. Von diesem Punkt aus gibt es verschiedene Möglichkeiten wie weiter verfahren werden kann. Es ist möglich alles per Hand zu steuern. Aktionen werden durch Tastaturkommandos ausgewählt, ebenso wird bestimmt, wann ein Situationswechsel geschieht. Dies eignet sich, um für jeden Übergang genau die Entwicklung der Aktivierungen zu beobachten. Die Auswahl der nächsten Aktion kann auch nach dem beschriebenen Verfahren aus Abschnitt 6.2.3 automatisiert ablaufen, um so schnell den Graphen zu explorieren. Außerdem kann entschieden werden, ob zusätzlich nach der vollen Exploration automatisiert zu einem zufälligen Zeitpunkt in eine andere ebenfalls zufällig gewählte Situation gesprungen werden soll. Dieses Verfahren bietet sich vor allem dann an, wenn nur das Ergebnis des Experiments benötigt wird. Zur genaueren Beobachtung der Situationswechsel können diese auch manuell hervorgerufen werden. Unabhängig welche der genannten Methoden gewählt wird, funktioniert das Prinzip der Simulation der echten Daten wie folgt: Ausgehend vom aktuellen Zustand $x(t)$ des Semni, wird bei einer Wahl der nächsten Aktion, für die aktuelle Situation in dem eingelesenen sensomotorischen Graphen überprüft, zu welcher Posture p_i die Aktion bei der Aufnahme geführt hat. Die Zustand x_{p_i} von p_i wird an das ABC-Learning als neuer Semni-Zustand $x(t+1)$ geliefert, welches diesen als neue Posture lernt und inklusive dem Übergang in den zu lernenden sensomotorischen Graphen einfügt, wenn diese nicht schon vorhanden sind. Außerdem wird die beschriebene Situationserkennung auf Basis des gelernten Graphen ausgeführt. Die aufgenommenen Daten bilden die echte Welt für das ABC-Learning und die Situationserkennung. Ein Beispiel hierfür findet sich in Abbildung 59.

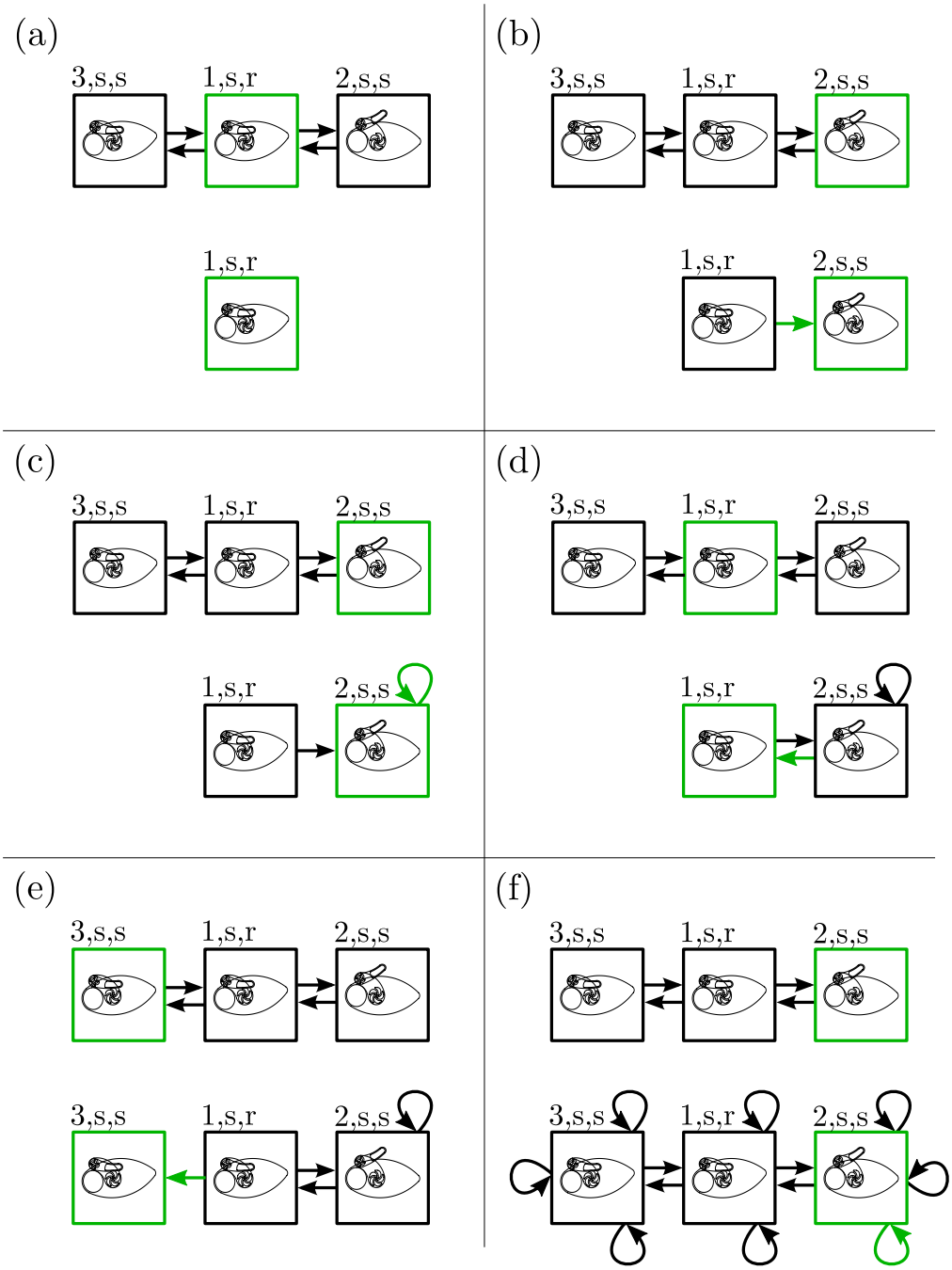


Abbildung 59: Dargestellt wird das Prinzip der Simulation von echten Daten. In jedem Abschnitt sind jeweils oben der sensomotorische Graph, wie er bei der Aufnahme entstanden ist, und unten der anhand der aufgenommenen Daten gelernte Graph zu sehen. Von oben links nach unten rechts ist zu sehen, wie der Graph dieser Situation gelernt wird. Die Graphen sind nach der gleichen Semantik gestaltet, wie sie in den bisherigen Beispielen dieser Art war. In dem unteren Graphen sind auch für Übergänge, die zur Posture zurückführen, Pfeile eingezeichnet, um so unterscheiden zu können, ob die Möglichkeit schon exploriert wurde oder nicht. Mit grün ist im unteren Graphen zusätzlich der letzte Übergang markiert.

Zu Beginn ist zu sehen, dass aus der „Welt“ nur die Startposture p_1 bekannt ist. Anschließend wird das Knie in positive Richtung umgeschaltet. Bei der Aufnahme der Daten hat dies zur Posture p_2 des oberen Graphen geführt. Deswegen wird diese Posture nun als nächstes mit der zugehörigen Verbindung e_{12} gelernt. In den nächsten Schritten werden weitere Verbindungen und die letzte Posture gelernt. Ganz unten rechts ist der komplett gelernte Graph zu sehen. Währenddessen werden die Verbindungen der ersten Situation zugeordnet. Beim Wechsel der Situation wird der obere Graph durch den Graph der neuen Situation ersetzt. Enthält die Situation auch die aktuelle Posture, so wird in dieser Posture geblieben und davon ausgegangen der Situationswechsel während des Betriebs erfolgte. Ist die aktuelle Posture nicht in dieser vorhanden, wird in die erste Posture dieser Situation gewechselt und der Situationswechsel so behandelt, als ob der Semni zwischendurch abgeschaltet und dann in der neuen Situation wieder gestartet wurde.

Durch diesen Aufbau lassen sich in kurzer Zeit viele Experimente durchführen. Wären die Experimente direkt auf dem Semni ausgeführt worden, hätte es auf Grund der Dauer jedes Übergangs viel Zeit benötigt, um einen großen Datensatz zu generieren. Konkret wurden für zehn verschiedene Wahrscheinlichkeiten eines Situationswechsels zehn mal 5000 Übergänge vollzogen. Die Wahrscheinlichkeit gibt dabei an, wie wahrscheinlich es in jedem Übergang ist, dass ein Situationswechsel erfolgt. In Abbildung 60 sind die für die Experimente verwendeten Situationen dargestellt. Es wurden viele ähnliche Situationen ausgewählt. Dadurch lassen sich Vermutungen über die Entwicklung des Graphen bei beliebig vielen Situationen anstellen, da zu erwarten ist, dass es bei beliebig vielen Situationen Gruppen von sehr ähnlichen Situationen gibt. Des Weiteren kann hierdurch die Situationserkennung in schwierigsten Bedingungen getestet werden, da ähnliche Situationen schwerer zu unterscheiden sein sollten als unähnliche. Die Nummerierung der Situationen wird in Abschnitt 8.4 wiederverwendet und entspricht der Reihenfolge, in der sie in einem beispielhaften Durchlauf gelernt wurden.

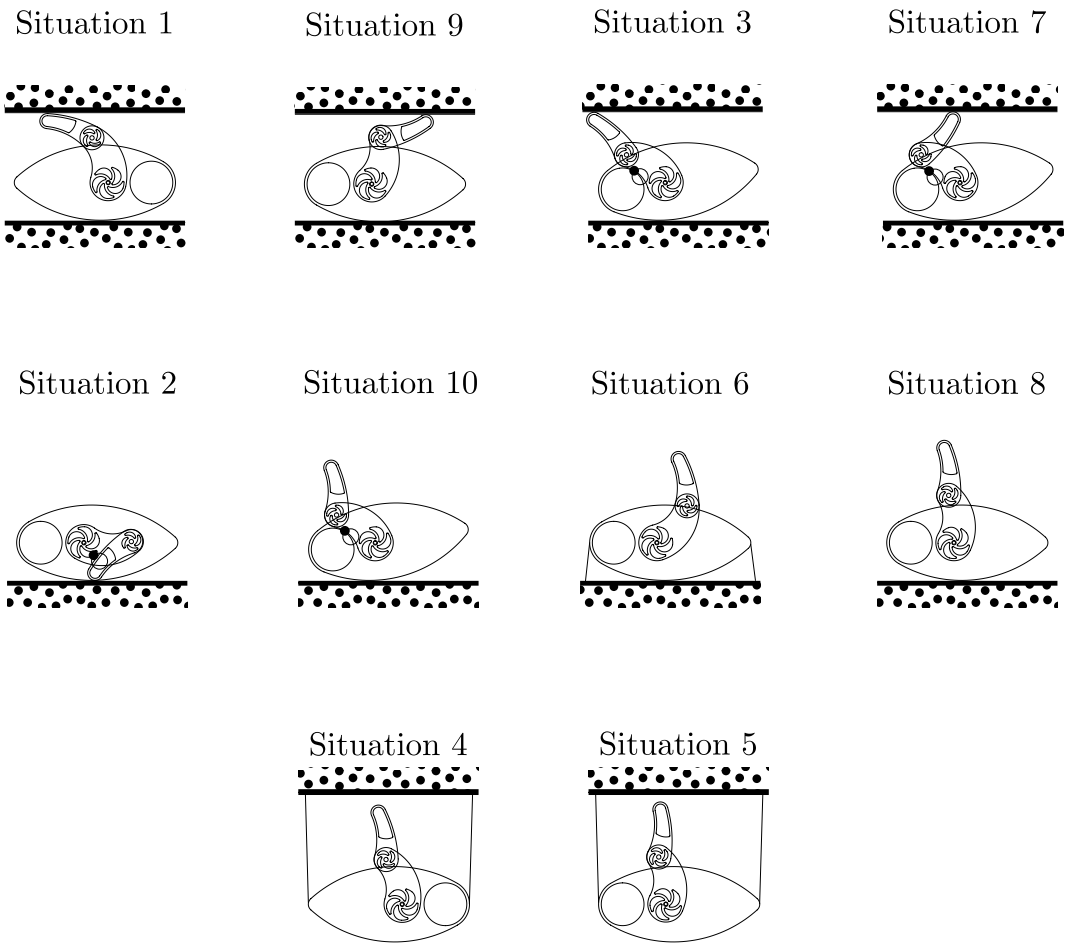


Abbildung 60: Dargestellt sind die für die Experimente verwendeten Situationen, wobei die Nummerierung der Reihenfolge, in der sie in einem Durchlauf gelernt wurden, entspricht. In den S_1 und S_9 kann sich der Semni auf dem Boden liegend frei bewegen, während er von oben durch eine Decke begrenzt wird und in Bauchlage (S_1) oder Rückenlage (S_9) ist. S_2 ist dadurch gekennzeichnet, dass das Knie des Semni festgebunden ist und er in Rückenlage auf dem Boden liegt. Die Situationen S_3, S_7, S_10 haben als Gemeinsamkeit, dass die Hüfte des Semni am Kopf festgemacht macht. In S_3 und S_7 wird er zusätzlich von oben durch eine Decke begrenzt. Sie unterscheiden sich in der Lage des Knieglied. In S_3 ist es mehr auf der Seite des Kopfes und in S_7 auf der anderen Seite. Situation S_4 und S_5 haben gemein, dass der Semni frei in der Luft aufgehangen ist, jeweils einmal in Rückenlage (S_5) und Bauchlage (S_4). In S_6 ist der Körper des Semni am Boden festgebunden. Knie und Hüfte können ansonsten frei bewegt werden. Situation S_8 ist die freieste Situation. In dieser kann sich der Semni frei auf dem Boden liegend bewegen. Rücken- oder Bauchlage spielt dabei keine Rolle, da er in dieser Situation von der einen in die andere Lage wechseln kann.

8.2 Bewertungsmaße

Um einschätzen zu können, wie gut die Situationserkennung funktioniert, werden verschiedene Maße gewählt, mit denen Aussagen hierzu möglich sind. Es soll zunächst gemessen werden, wie oft die Situationserkennung richtig bzw. falsch liegt. Da die Situationserkennung nach jeder Aktion entscheidet, welche Situation gerade vorliegt,

ist es sinnvoll, bei jeder Aktion zu überprüfen, ob sie richtig liegt oder nicht. Als Maß bietet sich an, zu zählen, bei wie vielen Aktionen die Situationserkennung richtig liegt. Die Summe wird anschließend ins Verhältnis zu der Anzahl der insgesamt getätigten Aktionen δ gesetzt. Wenn $S(k)$ die wahre aktuelle Situation und $S_y(k)$ die durch das Verfahren angenommene Situation nach dem k-ten Übergang kennzeichen, ergibt sich die Erkennungsrate ϵ der Situationserkennung wie folgt:

$$\mathrm{BOOL}(L) = \begin{cases} 1 & \text{falls } L \text{ wahre Aussage} \\ 0 & \text{sonst} \end{cases}$$

$$\epsilon = \frac{\sum_{k=1}^{\delta} \mathrm{BOOL}(S(k) = S_y(k))}{\delta}$$

Bei einem Wert von 1,0 liegt die Situationserkennung immer richtig und entsprechend bei einem Wert von 0,0 immer falsch.

Als Nächstes soll gemessen werden, wie lange es nach einem Situationswechsel dauert, dass die richtige Situation detektiert wird. Mithilfe dieses Maßes wird differenziert, wie viele der Aktionen mit einer falschen Situationserkennung durch den Situationswechsel entstanden sind und wie oft ohne einen Situationswechsel die falsche Situation angenommen wird. Dazu werden die Aktionen τ_k aufsummiert, die die Situationserkennung nach jedem Situationswechsel braucht, um die richtige Situation zu erkennen. Normalisiert wird mit der Gesamtanzahl an Situationswechseln λ und ergibt so das soeben beschriebene Maß σ:

$$\sigma = \frac{\sum_{k=1}^{\lambda} \tau_k}{\lambda}$$

Zur Bewertung des in Abschnitt 7.5 beschriebenen Sicherungsverhaltens wurde noch ein weiteres Maß definiert. Das Sicherungsverhalten soll dazu dienen, die Unsicherheit über die aktuelle Situation zu minimieren. Je geringer der Unterschied der Aktivierungen der beiden Situationen mit den höchsten Aktivierungen ist, desto größer ist die Unsicherheit. Deswegen wird als Maß ξ der Unsicherheit in jeder Aktion der Betrag der Differenz zwischen den Aktivierungen dieser beiden Situationen aufsummiert, mit der Anzahl δ der Aktionen normalisiert und anschließend von Eins

abgezogen. Somit lässt sich ξ formalisieren zu:

$$\xi = 1 - \frac{\sum_{k=1}^{\delta} |\max(A(k)) - \max2(A(k))|}{\delta}$$

8.3 Ergebnisse des Verfahrens mit und ohne Sicherungsverhalten

Wie in 8.1 beschrieben, wurden für zehn Wahrscheinlichkeiten zehn Durchläufe mit je 5000 Übergänge durchgeführt. Für jede Wahrscheinlichkeit wurden anschließend die gewählten Bewertungsmaße jeweils über die zehn Durchläufe gemittelt. Im Folgenden sind diese Mittelwerte als Graph über die Wahrscheinlichkeiten dargestellt. Für jedes Bewertungsmaß wurden dabei jeweils die Ergebnisse ohne Sicherungsverhalten blau und mit diesem rot dargestellt.

8.3.1 Erkennungsrate

In Abbildung 61 ist die Erkennungsrate dargestellt. Diese bewegt sich für die gewählten Wahrscheinlichkeiten in einem Bereich von 0,78 bis 0,98. Des Weiteren ist bei beiden Kurven zu beobachten, dass mit zunehmender Wahrscheinlichkeit für einen Situationswechsel die Erkennungrate abfällt, wobei das Verfahren ohne Sicherungsverhalten schneller abfällt.

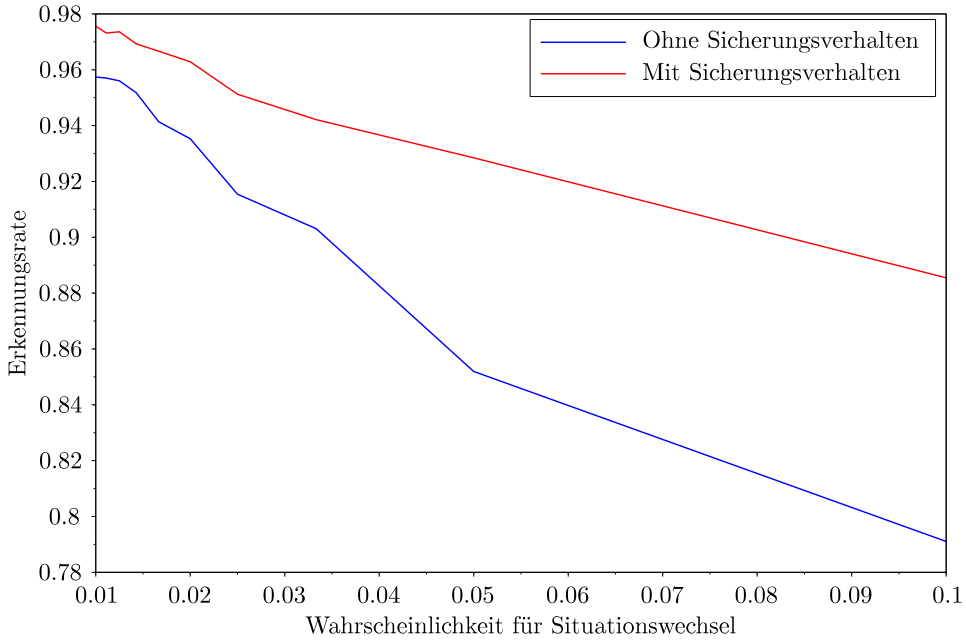


Abbildung 61: Abgebildet ist die Gegenüberstellung der Erkennungsrate des Verfahren ohne und mit Sicherungsverhalten. Das Verfahren liefert mit Sicherungsverhalten bei allen Wechselwahrscheinlichkeiten eine höhere Erkennungsrate.

Allgemein ist dieser Verlauf damit zu erklären, dass nach einem Situationswechsel zunächst fast immer eine Fehleinschätzung vorliegt. Diese wird so lange beibehalten, bis sie sich eindeutig als falsch erwiesen hat. Hierfür sind unter Umständen mehrere Übergänge nötig, in denen jeweils die falsche Situation angenommen wird und die Erkennungsrate verringert. Solche Übergänge werden im weiteren Verlauf *Fehlerkennungen* genannt. Bei vielen Situationswechseln geschieht dies entsprechend öfter, wodurch die Erkennungsrate in diesen Fällen niedriger ist, als bei wenigen Situationswechseln. Das Verfahren mit Sicherungsverhalten zeigt im gesamten Verlauf eine höhere Erkennungsrate. Eine Erklärung hierfür ist, dass das Sicherungsverhalten nach einem Situationswechsel schneller einen eindeutigen Übergang ansteuert, als es zufällig geschieht. Dadurch summieren sich bei jedem Situationswechsel weniger Fehlerkennungen auf, die die Erkennungsrate verringern würden. Die Vermutung bestätigt sich bei den folgenden Ergebnissen zu den Fehlerkennungen nach einem Situationswechsel.

8.3.2 Fehlerkennungen nach einem Situationswechsel

Abbildung 62 zeigt für die verschiedenen Wechselwahrscheinlichkeiten die Anzahl der Fehlerkennungen nach einem Situationswechsel. Wie beschrieben wird die Summe dieser für einen Lauf mit der Anzahl der Situationswechsel normiert. Für beide

Verfahren liegt dieser Wert zwischen 1,5 und 6 Fehlerkennungen je Situationswechsel. Mit zunehmender Wahrscheinlichkeit fällt jeweils die Anzahl der Fehlerkennungen.

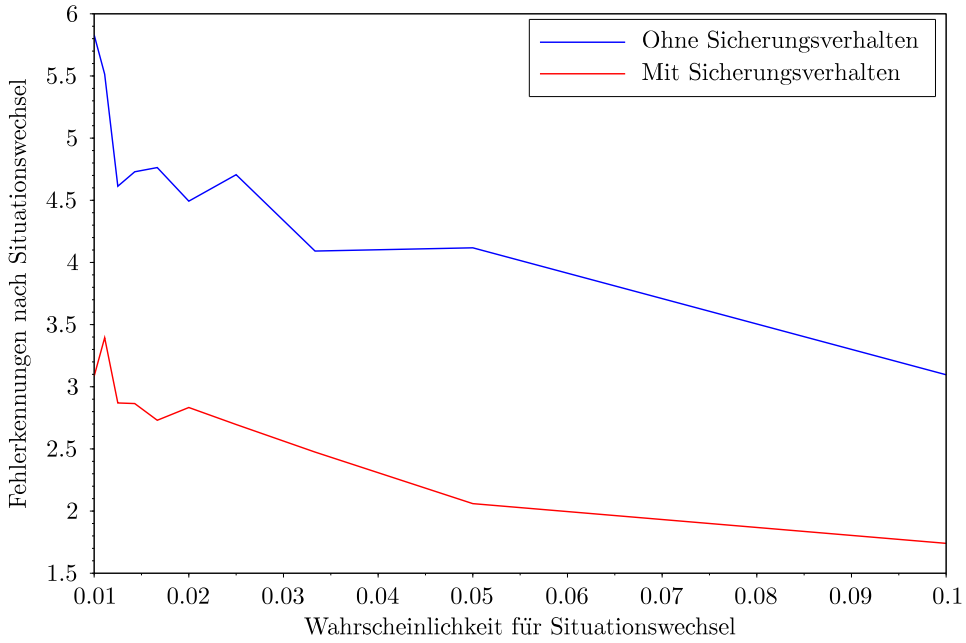


Abbildung 62: Dargestellt ist die Anzahl der Fehlerkennungen nach einem Situationswechsel. Es zeigt sich, dass das Verfahren mit dem Sicherungsverhalten weniger Fehlerkennungen hat als ohne.

Dies wirkt zunächst kontraintuitiv, da die erste Vermutung wäre, dass diese gleich bleiben sollte. Bei näherer Betrachtung stellt sich heraus, dass bei hohen Wahrscheinlichkeiten der Wechsel eher dann passiert, wenn man sich gerade noch in einem eindeutigeren Teil des Graphen befindet. Bei längeren Abständen zwischen den Situationswechseln hat das Verfahren mehr Zeit in einen uneindeutigen Abschnitt des Graphen zu gelangen, wodurch es nach einem Wechsel mehr Übergänge benötigt, wieder in einen eindeutigeren Teil überzugehen. Das Verfahren hat mit Sicherungsverhalten für alle getesteten Wahrscheinlichkeiten weniger Fehlerkennungen nach einem Situationswechsel als ohne diesen. Wie im vorherigen Abschnitt beschrieben, liegt dies an der schnelleren Evaluierung der Situation des Sicherungsverhaltens. Damit bestätigt sich die Vermutung, dass dies ein Grund für die höhere Erkennungsrate ist. Das dies der einzige Grund ist, lässt sich daran zeigen, dass für jeden Durchlauf die Summe aus Fehlerkennungen nach einem Situationswechsel σ und Übergängen mit richtig erkannter Situation gleich der der Gesamtanzahl der vollzogenen Übergänge δ ist. Dies bedeutet, dass alle Fehlerkennungen durch den Situationswechsel hervorgerufen wurden und das Verfahren, nachdem es einmal die richtige Situation erkannt hat, bei dieser Entscheidung bleibt, bis erneut

ein Situationswechsel auftritt. Widersprüchlich scheint, dass sich die durchschnittlichen Fehlerkennungen nach einem Situationswechsel annähern, während sich die Erkennungsraten voneinander entfernen, obwohl nur diese Fehlerkennungen die Erkennungsrate verringern. Dies liegt daran, dass die Erkennungsrate pro Übergang und die Fehlerkennungen pro Situationswechsel abgebildet sind. Um vergleichbare Graphen zu haben, wird zunächst der Erwartungswert μ einer Fehlerkennung durch einen Situationswechsel je Übergang berechnet. Die Wahrscheinlichkeit P_b für einen Situationswechsel vor einem Übergang wird dazu mit den zugehörigen Fehlerkennungen nach einem Situationswechsel σ multipliziert:

$$\mu = \sigma P_b$$

Dargestellt sind diese Graphen in Abbildung 63. Hier ist zu sehen, dass sich der Erwartungswert mit zunehmender Wahrscheinlichkeit erhöht. Außerdem steigt dieser ohne Sicherungsverhalten schneller als mit

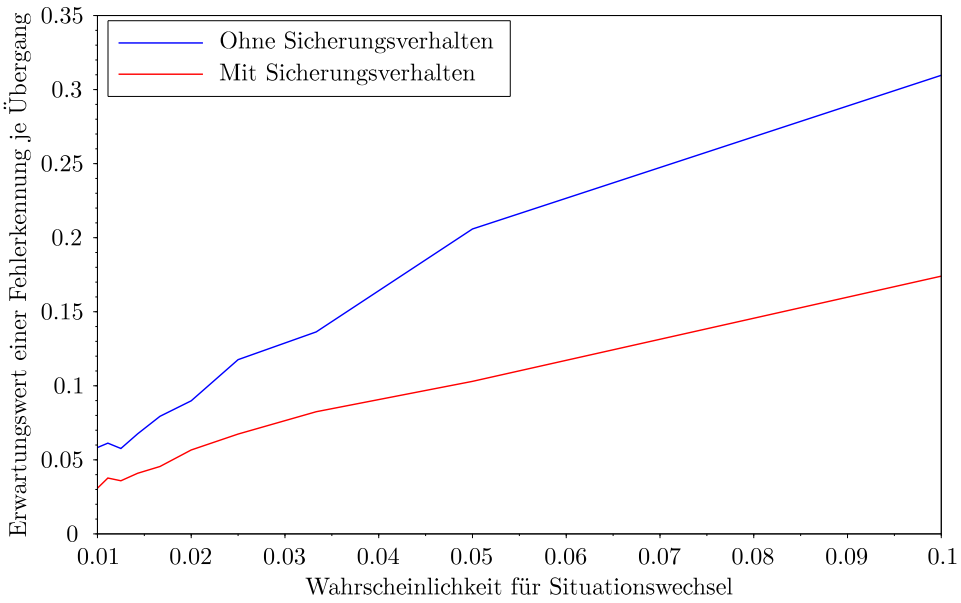


Abbildung 63: Abgebildet ist der Erwartungswert einer Fehlerkennung nach einem Situationswechsel je Übergang. Dieser ist mit Sicherungsverhalten geringer als ohne. Werden zur Berechnung anstatt den theoretischen die empirisch ermittelten Wechselwahrscheinlichkeiten gewählt, addiert sich der Erwartungswert und die Erkennungsrate zu Eins. Daraus lässt sich schließen, dass alle Fehlerkennungen auf einem Situationswechsel beruhen.

Diese Beobachtungen decken sich mit denen der Erkennungsrate und belegen zusätzlich die Erläuterungen zu diesen. Theoretisch sollten sich der Erwartungswert und die Erkennungsrate für jede Wahrscheinlichkeit zu Eins addieren lassen. In diesem Fall

kommen jeweils Werte über Eins heraus. Dies liegt daran, dass der Erwartungswert mit der theoretischen Wahrscheinlichkeit, wie sie der horizontalen Achse zu entnehmen ist, berechnet wurde. Praktisch hat sich jedoch gezeigt, dass die Wahrscheinlichkeiten jeweils geringer waren, wodurch der Erwartungswert kleiner wäre. Dies liegt daran, dass die verwendete Zufallszahlenfunktion von PureBasic keine guten Zufallszahlen liefert. Wird der Erwartungswert mit den empirisch bestimmten Wahrscheinlichkeiten berechnet, addieren sich, bei Vernachlässigung kleiner Rundungsfehler, die Erkennungsrate und der Erwartungswert zu Eins.

8.3.3 Unsicherheit

Das Sicherungsverhalten hat bereits in den vorherigen Bereichen wesentliche Verbesserungen gezeigt. In Abbildung 64 wird die Unsicherheit mit und ohne diesem Verhalten gegenübergestellt. Das eigentliche Ziel des Sicherungsverhaltens ist es, diese zu verringern. Allgemein liegt die Unsicherheit je Übergang für die getesteten Wahrscheinlichkeiten in einem Bereich von 0,2 bis 0,38. Tendenziell steigt diese mit zunehmender Wechselwahrscheinlichkeit an.

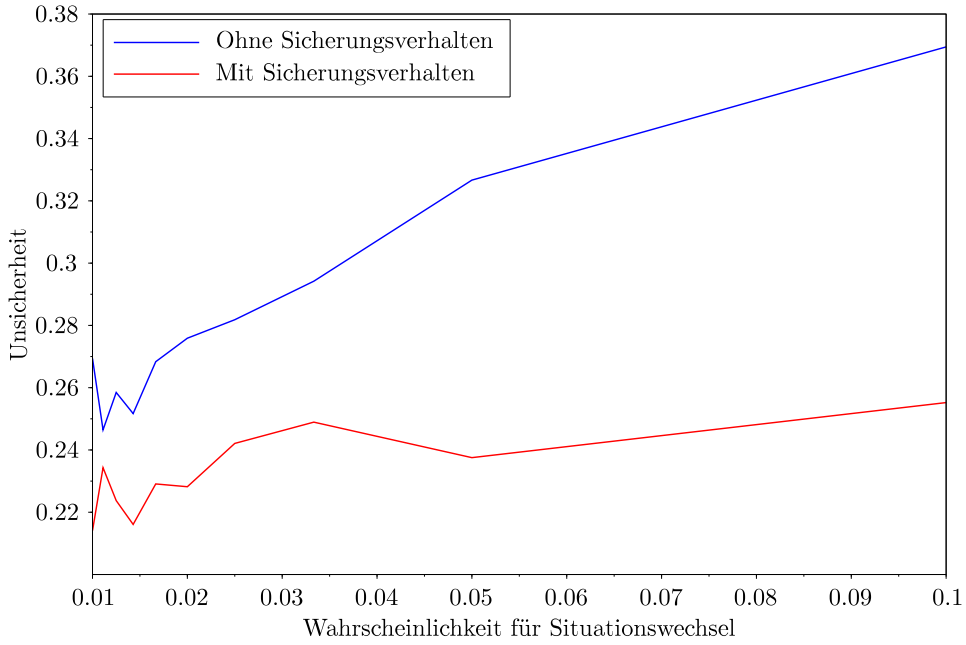


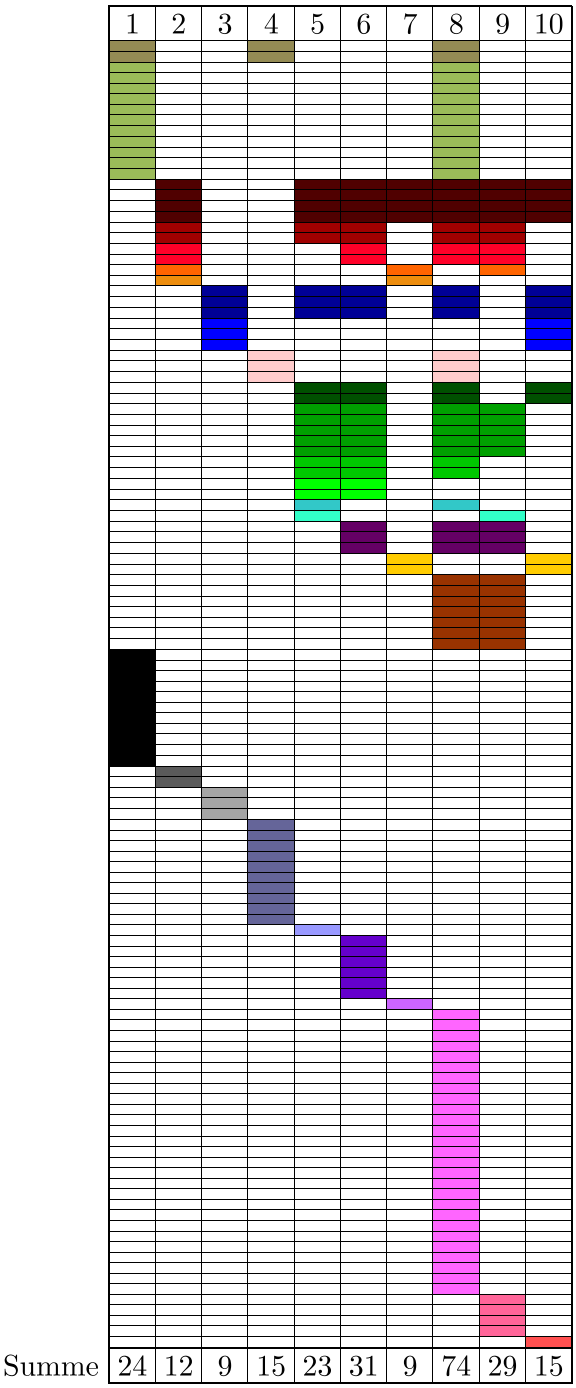
Abbildung 64: Dargestellt ist die Unsicherheit je Übergang für die verschiedenen Wechselwahrscheinlichkeiten. Das Sicherungsverhalten sorgt für alle Wahrscheinlichkeiten dafür, dass die Unsicherheit verringert wird.

Ein Erklärung hierfür ist, dass auch Situationswechsel simuliert werden, bei denen der Semni ausgeschaltet und in einer neuen Situation wieder eingeschaltet wird.

Nach dem Einschalten kann die aktuelle Situation nur auf Grund der Posture bestimmt werden, da noch kein Übergang vorliegt. Dies ist, wie im folgenden Abschnitt beschrieben wird, nicht so eindeutig, weshalb in diesen Fällen die Unsicherheit groß ist. Bei höheren Wahrscheinlichkeiten treten mehr dieser Wechsel auf, wodurch die Unsicherheit insgesamt zunimmt. Dies kann auch als Erklärung dafür dienen, dass die Unsicherheit ohne Sicherungsverhalten schneller anwächst als mit. Umso mehr Wechsel dieser Art geschehen, desto öfter kann das Sicherungsverhalten seinen Vorteil ausspielen, dass es schneller die richtige Situation evaluiert. Im Allgemeinen hat das Sicherungsverhalten eine geringere Unsicherheit und erfüllt somit auch seinen eigentlichen Zweck.

8.4 Analyse des entstandenen sensomotorischen Graphen

In diesem Abschnitt wird der gelernte Graph analysiert, in dem alle Postures aus allen Situationen enthalten sind. In jeder Situation läuft das beschriebene ABC-Learning ab. Dieses exploriert dabei die Fixpunkte und baut den sensomotorischen Graphen auf. Die Situationserkennung ordnet den Verbindungen des Graphen bestimmte Situationen zu. Eine Posture gehört somit zu allen Situationen, die den Übergängen der Posture zugeordnet sind. In Abbildung 65 sind die Postures den Situationen zugeordnet. Jede Zeile stellt eine Posture dar. Entlang der Spalten sind die Situationen angeordnet. Ist eine Posture in einer Situation vorhanden, so ist die jeweilige Zelle, die sich aus der Zeile der Posture und der Spalte der Situation ergibt, eingefärbt. Ist die Zelle leer, kommt die Posture nicht in der Situation vor. Die Farben kodieren bestimmte Klassen von Postures, die sich durch die Kombination an Situationen zusammensetzen, in denen diese vorkommen.



Anzahl	
2	1, 4, 8
11	1, 8
4	2, 5, 6, 7, 8, 9, 10
2	2, 5, 6, 8, 9
2	2, 6, 8, 9
1	2, 7, 9
1	2, 7
3	3, 5, 6, 8, 10
3	3, 10
3	4, 8
2	5, 6, 8, 10
5	5, 6, 8, 9
2	5, 6
1	5, 8
1	5, 9
3	6, 8, 9
2	7, 10
7	8, 9
11	1
2	2
3	3
10	4
1	5
6	6
1	7
27	8
4	9
1	10

Abbildung 65: Abgebildet ist die Zuordnung der Postures zu den einzelnen Situationen. Die Farben gruppieren Postures nach der Situationskombination in der sie vorkommen. Beispielsweise sind Postures, die ausschließlich in den Situation S_1 und S_8 vorkommen, mit einem helleren Olivgrün markiert, während Postures, die nur in S_1 enthalten sind mit Schwarz gekennzeichnet sind. Die Zuordnung der Farben ist rechts neben der Tabelle dargestellt. Die Zahl innerhalb des farbigen Kästchens gibt die Anzahl der Postures an, die in der jeweiligen Kombination an Situationen vorkommen.

Es lässt sich erkennen, dass in etwa die Hälfte der Postures mehreren Situationen und die andere Hälfte jeweils einer einzelnen Situation zugeordnet sind. Außerdem gibt es Postures, die in sehr vielen Situationen vorkommen. Beispielsweise sind die Postures, die mit dem dunkelsten Rot markiert und den Situationen S_2, S_5 bis S_{10} zugeordnet sind, in sieben von zehn Situationen enthalten. In Abbildung 66 ist der Zustand $x(t)$ des Semni dargestellt, der durch Mittelwertbildung aus den zugehörigen Postures dieser Klasse entsteht.

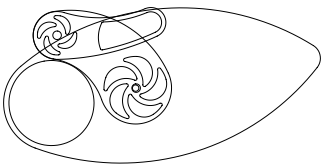


Abbildung 66: Dargestellt ist der Mittelwert aller Postures, die zur Kategorie mit dem dunkelsten Rot gehören. Alle Winkel sind repräsentativ, da die Varianz der Winkel zwischen den zugehörigen Postures sehr klein ist.

Die Varianz der einzelnen Winkel liegt bei 10^{-5} und kleiner. Diese Postures unterscheiden sich hauptsächlich durch die verwendeten CSL-Modi. Alle zugehörigen Situationen sind entweder komplett in Rückenlage oder ermöglichen zumindest eine Rückenlage. Situation S_1 und S_4 sind Situationen in denen der Semni sich nur in Bauchlage bewegen kann. Deswegen kommen diese Postures dort nicht vor. In Situation S_3, die ebenfalls nicht in dieser Klasse ist, ist er zwar in Rückenlage, kann jedoch sein Knie nicht in diese Richtung so weit einknicken, da sein Hüftglied am Kopf festgebunden ist, er von oben durch eine Decke begrenzt wird und das Knie zu Beginn der Situation ausgestreckt war. Allgemein sind die Situationen S_2 und S_5 bis S_{10} öfter miteinander kombiniert. Dies sind alles Situationen mit Rückenlage, in denen sich der Semni mehr oder weniger frei bewegen kann. Anhand der Überschneidungen der Postures zwischen den Situationen, lassen sich so Überkategorien für die Situationen formulieren. Des Weiteren ist, auf Grund der zum Teil bereits in Abbildung 65 zu sehenden großen Überdeckungen von Situationen, abzusehen, dass weitere Situationen immer weniger neue Postures hinzufügen werden. Es ist zu erwarten, dass es ein Satz an Postures gibt, mit dem sich alle Situationen beschreiben lassen. Zu sehen ist außerdem, dass die Situationen S_1, S_4 und S_8 relativ einfach über die Postures allein zu identifizieren, da diese jeweils Bauchlage haben und wesentlich weniger Bauchlagen- als Rückenlagen-Situationen aufgenommen wurden. Wiederum sind drei Situationen nur über jeweils eine Posture eindeutig zu identifizieren. Es ist zu vermuten, dass sich dies bei immer mehr Situationen fortsetzt, desto mehr gelernt werden, da bei mehr Situationen sich die Überschneidungen von Postures zwischen

diesen erhöht. Das bedeutet, dass es schwierig wäre, Situationen nur über die Postures zu erkennen. Deshalb wurde das hier beschriebenen Verfahren zur Situationserkennung auf Basis der Übergänge realisiert, da die Kombinationsmöglichkeiten von Übergängen zwischen den Postures offensichtlich größer ist, als die Anzahl der Postures. Dadurch können mehr Situationen unterschieden werden, als nur mit Hilfe der Postures möglich wäre. Einen Überblick darüber wieviele Postures zwei Situationen miteinander teilen, ist in Abbildung 67 gegeben.

	1	2	3	4	5	6	7	8	9	10
1		0	0	2	0	0	0	13	0	0
2			0	0	6	8	6	8	9	4
3				0	3	3	0	3	0	6
4					0	0	0	5	0	0
5						18	4	17	15	9
6							4	21	16	9
7								4	5	6
8									27	9
9										4
10										

Abbildung 67: Abgebildet ist eine Übersicht wieviele Postures zwei Situationen miteinander gemeinsam haben. Die Zahlen geben die absolute Anzahl der gemeinsamen Postures zwischen den Situationen der zugehörigen Zeile und Spalte an. Zusätzlich wird dies durch farbliche Markierungen unterstützt. Grün sind Situationen mit wenigen und rot mit vielen gemeinsamen Postures markiert.

Auch in dieser Abbildung zeigt sich, dass die Situationen S_5, S_6, S_8, S_9 jeweils viele Postures gemeinsam haben. Besonders S_8 und S_9 haben viele Postures gemeinsam. In S_8 liegt der Semni auf dem Tisch und kann sich frei bewegen. Die einzige Änderung in S_9 ist, dass er zusätzlich von oben durch eine Decke begrenzt ist, sodass er Hüfte und Knie nicht nach oben richten kann. Da ansonsten die Grundbedingungen sehr ähnlich sind, haben diese zwei Situationen viele Postures gemeinsam. Situation S_1 ist ähnlich der Situation S_8. Hier liegt der Semni jedoch auf dem Bauch anstatt auf dem Rücken. Trotzdem gibt es relativ viele gemeinsame Postures mit S_8, da in dieser Situation der Semni auch in Bauchlage geraten kann.
Als Fazit lässt sich die Analyse damit abschließen, dass sich vermutlich alle erkennbaren Situationen durch einen begrenzten Satz an Postures bilden lassen und eine Erkennung deswegen auf den Übergängen beruhen sollte, die mehr Möglichkeiten zulassen. Diese Begrenztheit der Posturevielfalt kann dafür genutzt werden, anhand der gemeinsamen Postures Überkategorien für die Situationen zu definieren, wie beispielsweise Rücken- oder Bauchlage.

9 Diskussion

In diesem Abschnitt wird das beschriebene Verfahren kritisch begutachtet und Problemstellen näher erläutert. Dazu wird der Situationsbegriff noch einmal hinterfragt und anschließend auf die Voraussetzungen bzw. Annahmen für dieses Verfahren eingegangen.

9.1 Diskussion des Situationsbegriffs

Wie im Abschnitt 7.2 beschrieben, kann der Begriff der Situation verschiedene Bedeutungen haben. Ebenso steht und fällt der Erfolg der Situationserkennung mit der genauen Definition der Situationen, die erkannt werden sollen. Beispielsweise kann für bestimmte Einsatzzwecke die Farbe der Umgebung ausschlaggebend für die aktuelle Situation sein. Da der Semni jedoch nicht über die nötigen Sensorqualitäten zur Wahrnehmung von Farbe verfügt, ergibt es keinen Sinn, diese in den hier verwendeten Situationsbegriff zu integrieren. Ein anderes Beispiel ist die Unterscheidung ob eine Decke vorhanden ist oder nicht. Nach Definition der in dieser Arbeit verwendeten Situationen, gibt es keinen Unterschied zwischen freiem Himmel oder dem Vorhandensein einer Zimmerdecke. Eine Decke wird erst dann relevant für die Situation, wenn diese nah genug am Semni ist, um seine Fixpunkte zu beeinflussen. Es handelt sich, bei der hier beschriebenen Situationserkennung nicht um eine allgemeingültige Situationserkennung, sondern nur um eine Erkennung bestimmter Arten von Situationen.

9.2 Diskussion der Annahmen und Voraussetzungen der Situationserkennung

In diesem Abschnitt wird auf die drei Annahmen eingegangen, die zur Realisierung der Situationserkennung getroffen wurden.
Die erste Annahme war, dass eine Situation immer den gleichen sensomotorischen Graphen produziert. Diese Annahme bildet den Kern dieser Arbeit und muss zwingend erfüllt sein. Angenommen der Graph wäre in jedem Durchlauf der Situation anders, so könnte die Situationserkennung, die auf Basis dieses Graphen realisiert wurde, nur so lange diese Situation wiedererkennen, wie dieser Graph sich nicht verändert. Verändert sich der Graph, würde eine neue innere Situation gelernt werden, die der gleichen äußeren Situation zugeordnet wird. In Frage zu stellen ist, ob bei ständig wechselnder Bezeichnung der Situation seitens des Semnis, immernoch von einer zuverlässigen Situationserkennung gesprochen werden kann. Bei der Aufnahme der verschiedenen Situationen hat sich jedoch gezeigt, dass für die gewählten

Situationen der entstehende Graph immer gleich ist. Der einzige ersichtlicher Grund, dass sich der Graph in einer Situation ändert, ist, wenn sich mit der Zeit beispielsweise die Winkelsensoren dekalibriren oder andere Verschleißerscheinungen auftreten. Dies wäre ein schleichender Prozess, der durch das ABC-Learning aufgefangen werden könnte, indem langsam die angesteuerten Postures mitdriften.
Als nächstes wurde die Annahme getroffen, dass innerhalb einer Situation jeder Aktion genau einem Übergang zugeordnet ist. Lässt man diese Annahme fallen, würde das bei dem beschriebenen Verfahren dazu führen, dass einer äußeren Situation mehrere innere Situationen zugeordnet werden. Als Erweiterung könnte implementiert werden, dass, so lange nicht komplett exploriert ist, in der aktuellen Situation immer der zuletzt vollführte Übergang einer Aktion gespeichert und der Situation zugeordnet wird. Sobald diese Situation voll exploriert ist und bei dieser Aktion ein anderer als der zuletzt gelernte Übergang vollführt wird, würde eine neue Situation gelernt werden, die zunächst diesen neuen Übergang enthält. Dies hätte jedoch die Schwachstelle, dass diese Situation eventuell wieder den gleichen Übergang enthält, wenn dieser beim Lernen als letztes auf diese Aktion erfolgt ist. Das Verfahren wäre somit vom „Zufall“ abhängig, ob der noch unbekannte Übergang während des Lernens als letztes auf diese Aktion folgte. Es wäre jedoch möglich, im Falle von zwei komplett gleichen erlernten Situationen, eine wieder zu löschen. Streng genommen kann man einen anderen Übergang für die gleiche Aktion auch als eine andere Situation definieren, da sich der Zustand der Welt und somit die Situation so geändert hat, dass eben dieser andere Übergang folgte. Deshalb kann dies als eine gültige Situationserkennung angesehen werden. Eine andere Lösung für das Problem wäre, alle Übergänge zu einer Aktion mit zu lernen und der Situation zuzuordnen. Das Problem ist, festzustellen wann die Situation voll exploriert ist, da im Voraus nicht klar ist, wieviele Übergänge bei einer Aktion möglich sind. Man könnte als Kriterium einen Grenzwert einführen, wie oft eine Aktion ausgeführt werden muss, um als exploriert zu gelten. Ein weiteres Problem ist die Berechnung der kürzesten Wege. Wenn für eine Aktion mehrere Übergänge möglich sind, ist der Weg, der durch die Aktion beschritten wird, uneindeutig und somit die Bestimmung des kürzesten Weges durch Dijkstras Algorithmus ohne weitere Anpassung nicht immer möglich. Bei der Implementierung des ABC-Learning hat sich gezeigt, dass es bei einer schlechten Implementierung dazu kommen kann, dass einer Aktion innerhalb einer Situation mehrere Übergänge zugeordnet werden. Ob diese Annahme zutrifft oder nicht hängt somit von der Robustheit der Implementierung des ABC-Learning ab.
Die letzte Voraussetzung ist, dass nach einem Wechsel in eine noch nicht erlernte Situation erst dann die Situation wieder gewechselt wird, wenn diese voll exploriert

wurde. Der Grund hierfür ist, dass bei der Exploration einer Aktion mit unbekanntem Ausgang durch das beschriebene Verfahren nicht entschieden werden könnte, ob der Übergang zu der gerade zu lernenden Situation oder einer anderen gehört, wenn diese Voraussetzung nicht gegeben ist. Als eine erste Lösung für dieses Problem könnte man zulassen, dass die Situation dann geändert werden darf, wenn als nächstes eine Aktion folgt, die bereits exploriert wurde und in der neuen Situation einen neuen Übergang produziert. In diesem Fall könnte geschlussfolgert werden, dass sich die Situation geändert hat, da zuvor die Annahme getroffen wurde, dass Aktionen eindeutig sind. Da der bei Erstellung der neuen Situation vollführte Übergang eindeutig ist (sonst wäre keine neue Situation erstellt worden), kann, sobald dieser oder ein anderer bereits gelernter eindeutiger Übergang passiert wird, die Situation wieder erkannt und weiter gelernt werden. Um die Möglichkeit einzuräumen, an beliebigen Stellen während des Lernens wieder die Situation zu ändern, könnte nach einem bereits fest gelerntem eindeutigen Übergang alle folgenden Übergänge zunächst auf Probe dieser Situation zugerechnet werden. Wird dann wieder ein fest gelernter eindeutiger Übergang passiert, können alle Übergänge die zwischen den beiden festen eindeutigen Übergängen vollführt wurden, ebenfalls fest der Situation zugeordnet werden. Auch hiermit könnten nicht alle Fälle abgefangen werden, in denen Situationen beliebig geändert werden, jedoch könnte dies einen ersten Ansatz darstellen.

10 Fazit und Ausblick

In dieser Arbeit wurde das ABC-Learning implementiert, um in verschiedenen Situationen die Fixpunkte zu explorieren. Es wurde untersucht, wie sich die verschiedenen Situationen auf die Fixpunkte auswirken und darauf aufbauend die Folgen für den sensomotorischen Graphen dargelegt, der durch das ABC-Learning aufgebaut wird. Anschließend wurde ein Verfahren beschrieben, dass basierend auf Diffusionsprozessen in dem sensomotorischen Graphen eine Situationserkennung realisiert. Dabei wurde der Begriff der Situation definiert und auf die Einbindung des Verfahrens in das bestehende ABC-Learning eingegangen. Die Experimente haben gezeigt, dass das Verfahren sehr gut funktioniert und mittels des Sicherungsverhaltens noch weiter verbessert werden konnte. Es wurden Problemstellen benannt und mögliche Lösungen aufgezeigt. Außerdem hat eine Analyse der Komplexität des Verfahrens ergeben, dass sowohl Speicherplatz, als auch Rechenkomplexität relativ gering sind und eine Umsetzung direkt auf dem Semni möglich sein sollte. Die Situationserkennung ergänzt das ABC-Learning um eine weitere Komponente zur Selbst- und Umgebungsexploration. Einerseits erfährt der Semni in den verschiedenen Situationen mehr über sich selbst, andererseits ist er nun in der Lage auch Schlussfolgerungen über seine Umwelt zu ziehen. Dies ist eine sinnvolle Erweiterung, da, wie in der Einleitung beschrieben, eine Situationserkennung wichtig für die Adaptivität des Roboters ist und somit ein autonomes Agieren in der dynamische Umwelt erlaubt.

Es gibt verschiedene Ansatzpunkte, um an diese Arbeit anzuschließen. Wie in Abschnitt 7.6 erläutert, ist es möglich, ein weiteres Ziel zur Exploration und Situationserkennung hinzuzufügen. Denkbar wäre, nach der Exploration und der Erkennung der aktuellen Situation, dynamischere Bewegungen auszuprobieren und mit den bisherigen Erfahrungen und Erwartungen abzugleichen und zu konsolidieren. In Werner (2013) wurde ein Algorithmus vorgestellt, der es ermöglicht, die Übergänge zwischen den Postures mit verschiedenen Geschwindigkeiten auszuführen. Dieser ließe sich situationabhängig gestalten und könnte in das beschriebene Framework aus ABC-Learning und Situationserkennung integriert werden.

Ein anderer Ansatzpunkt wäre, die verschiedenen Voraussetzungen für das beschriebene Verfahren der Situationserkennung aufzuweichen und das Verfahren in der beschriebenen oder einer ganz anderen Weise so zu erweitern, dass diese Voraussetzungen eventuell nicht mehr nötig sind. Hierzu müssten zunächst weitere Untersuchungen angestellt werden, welche Auswirkungen das Weglassen der Voraussetzungen

haben können. Im besten Fall wäre es möglich, ganz auf diese verzichten zu können.

Des Weiteren ist es möglich, Algorithmen zu entwickeln, die aus dem gesamten sensomotorischen Graph inklusive der Situationen weitere Informationen extrahieren. Beispielsweise könnten, wie beschrieben, Überkategorien für die Situationen gebildet werden. Außerdem wäre es möglich zu untersuchen, ob die Strukturen weiter komprimiert werden können, indem Situationen an einigen Stellen zusammengefasst werden, in denen sie sich komplett gleichen.

Im Abschnitt 6 wurde beschrieben, dass die Postures und Verbindungen um weitere Informationen wie durchschnittliche Motorspannung usw. erweitert werden können. Anhand dieser zusätzlichen Informationen könnte man weitere Schlussfolgerungen über den Zustand des Roboters ziehen. Beispielsweise könnte für einen Übergang eine erhöhte durchschnittliche Motorspannung dafür sprechen, dass ein zusätzliches Gewicht bewegt wird. Mit diesen Daten könnten auch ganz neue Verhalten implementiert werden. Wenn der Akku des Semni einen niedrigen Ladezustand aufweist, könnte er versuchen Übergänge zu vermeiden, die in der Vergangenheit besonders viel Energie benötigt haben.

Außerdem könnte man die Situationserkennung dahingehend erweitern, dass es kleine kontinuierliche Änderungen, die keine Bifurkation zur Folge haben ebenfalls erkennt. Wenn beispielsweise in allen Postures die gleiche Abweichung auftritt, beispielsweise eine leichte Rotation des Körperwinkels, könnte versucht werden, dies zu detektieren.

Bisher gibt es trotz der stetigen Verbesserungen der Hardware kein allgemeingültiges Verfahren, dass beliebigen Robotern erlaubt, sich robust und adaptiv in ihrer Umwelt zu bewegen. Das ABC-Learning, welches nun um eine Situationserkennung erweitert wurde, kann in Zukunft in vielfältiger Weise erweitert werden, um diesem Ziel Schritt für Schritt näher zu kommen. So soll es beliebigen Robotern irgendwann möglich sein, sich selbst und ihre Umwelt sicher zu explorieren und aufbauend auf den gewonnen Informationen, die gewünschte Adaptivität und Robustheit zu zeigen.

Literatur

Albus u. Meystel 2001

Albus, J. S. ; Meystel, A. M.: *Engineering of Mind: An Introduction to the Science of Intelligent Systems*. Wiley, 2001

Alligood u. a. 1997

Alligood, K. T. ; Sauer, T. D. ; Yorke, J. A.: *Chaos: An Introduction to Dynamical Systems*. Springer, 1997

Argyris u. a. 2010

Argyris, J. H. ; Faust, G. ; Haase, M. ; Friedrich, R.: *Die Erforschung des Chaos*. 2. Auflage. Springer, 2010

Arkin 1998

Arkin, R. C.: *Behavior-based Robotics*. MIT Press, 1998

Bay u. a. 2006

Bay, H. ; Tuytelaars, T. ; Van Gool, H.: *Surf: Speeded up robust features*. In: ECCV, 2006, S. 404–417

Bethge 2014

Bethge, S.: *ABC-Learning: Ein Lernverfahren zur modellfreien Selbstexploration autonomer Roboter*. Humboldt-Universität zu Berlin, Institut für Informatik, Diplomarbeit, 2014

Brooks 1990

Brooks, Rodney A.: *Elephants Don'T Play Chess*. In: Robot. Auton. Syst. 6 (1990), Nr. 1-2, S. 3–15

Cormen u. a. 2001

Cormen, T. H. ; Stein, C. ; Rivest, R. L. ; Leiserson, C. E.: *Introduction to Algorithms*. 2. Auflage. McGraw-Hill Higher Education, 2001

Dempster u. a. 1977

Dempster, A. P. ; Laird, N. M. ; Rubin, D. B.: *Maximum likelihood from incomplete data via the EM algorithm*. In: Journal of the Royal Statistical Society: Series B 39 (1977), S. 1–38

Der u. a. 2012

Der, R. ; Martius, G. ; Pfeifer, R.: *The Playful Machine: Theoretical Foundation and Practical Realization of Self-Organizing Robots*. Springer, 2012

Donoser u. Bischof 2013

Donoser, Michael ; Bischof, Horst: *Diffusion Processes for Retrieval Revisited.* In: CVPR, IEEE, 2013, S. 1320–1327

Falbe u. Regitz 1997

Falbe, J. ; Regitz, M.: *RÖMPP Lexikon Chemie, Band 2: Cm - G.* 10. Auflage. Thieme, 1997

Gemmell u. a. 2013

Gemmell, B. J. ; Costello, J. H. ; Colin, S. P. ; Stewart, C. J. ; Dabiri, J. O. ; Tafti, D. ; Priya, S.: *Passive energy recapture in jellyfish contributes to propulsive advantage over other metazoans.* In: Proceedings of the National Academy of Sciences 110 (2013), Nr. 44, S. 17904–17909

Gigerenzer u. Brighton 2009

Gigerenzer, G. ; Brighton, H.: *Homo Heuristicus: Why Biased Minds Make Better Inferences.* In: Topics in Cognitive Science 1 (2009), Nr. 1, S. 107–143

Gigerenzer u. Gaissmaier 2011

Gigerenzer, G. ; Gaissmaier, W.: *Heuristic Decision Making.* In: Annual Review of Psychology 62 (2011), Nr. 1, S. 451–482

Gigerenzer u. a. 2011

Gigerenzer, G. ; Hertwig, R. ; Pachur, T.: *Heuristics: The Foundations of Adaptive Behavior.* OUP USA, 2011

Gigerenzer u. Selten 2002

Gigerenzer, G. ; Selten, R.: *Bounded Rationality: The Adaptive Toolbox.* MIT Press, 2002

Gigerenzer u. a. 2004

Gigerenzer, G. ; Todd, P. M. ; Marsh, B.: *Cognitive heuristics: Reasoning the fast and frugal way.* In: J. P. Leighton & R. J. Sternberg (Eds.), The nature of reasoning (2004), S. 273–287

Großheim 2005

Grossheim, M.: *Der Situationsbegriff in der Philosophie. Mit einem Ausblick auf seine Anwendung in der Psychiatrie.* In: Symptom und Phänomen. Phänomenologische Zugänge zum kranken Menschen (2005), S. 114–149

Hild u. Kubisch 2011

Hild, M. ; Kubisch, M.: *Self-Exploration of Autonomous Robots Using*

Attractor-Based Behavior Control and ABC-Learning. In: 11th Scandinavian Conference on Artificial Intelligence (SCAI 2011), 2011

Klinke u. Baumann 2010

Klinke, R. ; Baumann, R.: *Physiologie.* Thieme, 2010

Kriesel 2007

Kriesel, D.: *Ein kleiner Überblick über Neuronale Netze.* 2007

Larose 2014

Larose, D. T.: *Discovering Knowledge in Data: An Introduction to Data Mining.* Wiley, 2014

Larsen u. Ziegenfuß 2004

Larsen, R. ; Ziegenfuss, T.: *Beatmung:.* Springer, 2004

von Lüde u. a. 2009

Lüde, R. von ; Moldt, D. ; Valk, R.: *Selbstorganisation und Governance in künstlichen und sozialen Systemen.* Lit, 2009

McLeod u. Dienes 1996

McLeod, P. ; Dienes, Z.: *Do fielders know where to go to catch the ball or only how to get there?* In: Journal of Experimental Psychology: Human Perception and Performance 22 (1996), Nr. 3, S. 531–543

Milford 2013

Milford, M.: *Vision-Based Place Recognition: How Low Can You Go?* In: The International Journal of Robotics Research 32 (2013), Nr. 7, S. 766–789

Mohri u. a. 2012

Mohri, M. ; Rostamizadeh, A. ; Talwalkar, A.: *Foundations of Machine Learning.* The MIT Press, 2012

Moore 1965

Moore, G. E.: *Cramming More Components onto Integrated Circuits.* In: Electronics 38 (1965), Nr. 8, S. 114–117

Murphy 2000

Murphy, R. R.: *Introduction to AI Robotics.* Cambridge, MA, USA : MIT Press, 2000

Ortiz 2012

Ortiz, R.: *FREAK: Fast Retina Keypoint.* In: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, 2012, S. 510–517

Scholze-Stubenrecht 2011

Scholze-Stubenrecht, W.: *Duden - Deutsches Universalwörterbuch: Das umfassende Bedeutungswörterbuch der deutschen Gegenwartssprache.* Bibliographisches Institut, 2011

Teschl 2012

Teschl, G.: *Ordinary Differential Equations and Dynamical Systems.* American Mathematical Society, 2012

Thiele 2014

Thiele, C.: *Design der verteilten Echtzeit-Systemarchitektur DISTAL und Implementierung am Beispiel des humanoiden Roboters Myon.* Humboldt-Universität zu Berlin, Institut für Informatik, Diplomarbeit, 2014

Werner 2013

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

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

3. Januar 2015