Diplomarbeit:

Anwendungen der Slow Feature Analysis in der humanoiden Robotik

Sebastian Höfer

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

Betreuer: Dr. Manfred Hild
Gutachter: Prof. Dr. Hans-Dieter Burkhard
Prof. Dr. Laurenz Wiskott

Berlin, den 17. November 2010

Für meine Eltern

Erklärungen

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Ich erkläre hiermit mein Einverständnis, dass die vorliegende Arbeit in der Bibliothek des Institutes für Informatik der Humboldt-Universität zu Berlin ausgestellt werden darf.

Ort, Datum Sebastian Höfer

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit der Slow Feature Analysis (SFA), einem unüberwachten Lernverfahren aus der theoretischen Biologie, und ihrer Anwendung in der humanoiden Robotik. Die SFA ist ein Algorithmus zur Extraktion von abstrakten, semantisch gehaltvollen Signalen aus einem vektoriellen Eingabesignal. Es wird untersucht, welche Signale die SFA, angewandt auf propriozeptive, nicht-visuelle Sensordaten eines humanoiden Roboters, findet, und wie diese für die Eigenwahrnehmung und Steuerung des Roboters verwendet werden können.

Die Arbeit gliedert sich in einen theoretischen und einen praktischen Teil. Im theoretischen Teil wird die SFA unter verschiedenen Gesichtspunkten betrachtet, indem der Algorithmus und Methoden zur Analyse seiner Ergebnisse präsentiert werden. Desweiteren werden Erweiterungen der SFA und ihre Verwandtschaft zu anderen Verfahren herausgestellt. Der Ausführungsschritt der SFA wird in einem künstlichen neuronalen Modell formuliert und hinsichtlich dieses Modells optimiert.

Der praktische Teil stellt zwei Anwendungen der SFA auf einer humanoiden Roboterplattform vor. Zum einen wird die SFA zur Erkennung von statischen Posen des Roboters sowie zur Dimensionsreduktion des sensorischen Raumes verwendet. Die Ergebnisse werden mit anderen Verfahren zur Dimensionsreduktion verglichen. Zum anderen wird die SFA auf eine dynamische Bewegung, ein zweibeiniges Laufmuster angewandt. Es wird gezeigt, dass die SFA innerhalb einer sensomotorischen Schleife zur Generierung des Laufmusters verwendet werden kann. Dabei wird die Reaktivität des Laufmusters erhöht, ohne dessen Stabilität wesentlich einzuschränken.

Abstract

This thesis deals with the Slow Feature Analysis (SFA), an unsupervised learning method stemming from the domain of theoretical biology, and its application in humanoid robotics. SFA is an algorithm that extracts abstract semantic features from a vectorial input signal. It is investigated which features SFA learns from proprioceptive, non-visual sensory data from a humanoid robot, and how the extracted features can be used for the robot's self-perception and control.

The thesis is divided into a theoretical and a practical part. The theoretical part describes the SFA algorithm and methods for the analysis of its results. Moreover, extensions and its relation to other methods are presented. The execution step of the SFA algorithm is reformulated in terms of an artificial neural model and optimised with respect to this model.

In the practical part, two applications of SFA for a humanoid robot platform are presented. First, SFA is employed for the detection of static postures performed by the robot, as well as for dimensionality reduction of the sensory state space. The obtained results are compared to other methods for dimensionality reduction. Secondly, SFA is applied to a dynamic motion, more precisely, to a biped gait pattern. It is shown that SFA can be used within the sensorimotor loop that generates the gait pattern, while at the same time increasing the reactivity of the gait pattern without profound loss in stability.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Bei der Untersuchung komplexer dynamischer Systeme ist man daran interessiert, aus verfügbaren Rohdaten abstrakte und semantisch gehaltvolle Informationen über den Zustand eines Systems zu gewinnen. Im theoretischen Sinne ist ein dynamisches System das Modell eines zeitabhängigen Prozesses, beispielsweise mathematischer oder physikalischer Art. In einem weiter gefassten Sinne bilden aber auch reale Systeme wie autonome Roboter im Zusammenspiel mit ihrer Umwelt ein dynamisches System aus, welches jedoch aufgrund seiner Komplexität meistens nicht in allen Belangen simuliert werden kann. Autonome Roboter erfahren und beeinflussen ihre Umwelt mittels diverser Sensoren und Aktuatoren, und eine prinzipielle Frage ist, auf welche Weise Sensorik und Aktuatorik gekoppelt werden müssen, damit Roboter komplexes und sinnvolles Verhalten aufweisen.

In den letzten Jahrzehnten wurden im Bereich der Robotik neue Ideen entwickelt, um sich dieser Fragestellung anzunähern, wobei alte Paradigmen der künstlichen Intelligenz, der so genannten *Good old-fashioned Artificial Intelligence*, in Frage gestellt wurden. Im besonderen werden die Prinzipien des *Embodiments* sowie der *Situatedness* hervorgehoben: Ersteres besagt, dass Kognition und Körper nicht getrennt voneinander betrachtet werden dürfen, letzteres, dass Wissen untrennbar vom Handeln ist [PB06]. Insbesondere wird der Ansatz verworfen, ausschließlich nach spezifischen, nicht verallgemeinerbaren Lösungen für anspruchsvolle kognitive Aufgabenstellungen zu suchen. Stattdessen verlagert sich der Blick auf die Suche nach fundamentalen Prinzipien, welche dem Roboter ermöglichen, den geschickten Umgang mit seiner eigenen Physis, seinen Sensoren und Aktuatoren zu erlernen und Aufgaben unterschiedlicher Art selbständig zu bewältigen.

Die *Neurorobotik* hat zusätzlich das Ziel, Strukturen im Roboter so zu konzipieren, dass sie auf gewissen Abstraktionsebenen die Architektur und Dynamik des Gehirns widerspiegeln. Die Hoffnung ist, auf diese Weise Erkenntnisse aus Experimenten mit den Robotern zu gewinnen, die Rückschlüsse auf biologische Prinzipien erlauben [SSK05].

Bevor man sich jedoch mit der Aktuatorik eines Roboters beschäftigt, sollte man zunächst die Frage beantworten, auf welche Art und Weise ein Roboter sinnvolle Informationen aus den ihm zur Verfügung stehenden sensorischen Daten extrahieren kann, mit anderen Worten: Gibt es allgemeine, weitgehend problemunabhängige Prinzipien, welche biologisch motiviert sind und

einem autonomen Roboter ermöglichen, abstrakte und semantisch gehaltvolle Informationen über sich selbst und seine Umwelt zu erlangen?

Diese Arbeit beschäftigt sich mit der Fragestellung, inwiefern das *Lernprinzip der Langsamkeit* ein solches Prinzip darstellt. Bevor die für diese Arbeit gewählte Vorgehensweise zur Bearbeitung dieser Fragestellung präsentiert wird, soll zunächst ebendieses Lernprinzip näher beleuchtet werden.

1.1 Langsamkeit als Lernprinzip

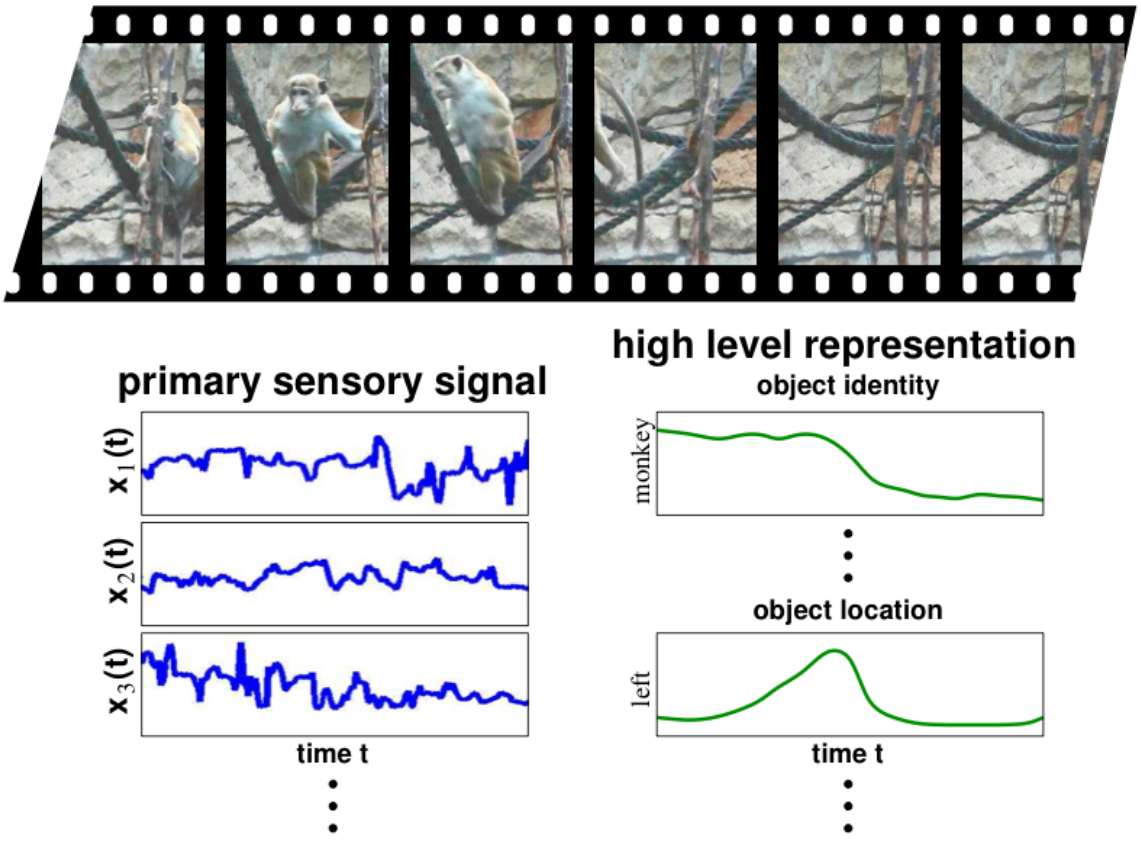


Abbildung 1.1: Langsamkeit als Lernprinzip: Verlässt der Affe den Bildausschnitt, variieren die versteckten Variablen Objektidentität und Objektposition des Affens auf einer langsameren Zeitskala als die Helligkeitswerte einzelner Pixel im Bild, welche sich relativ schnell ändern. (Aus [Ber06])

Das *Lernprinzip der Langsamkeit*, im folgenden auch *Langsamkeitsprinzip* genannt, basiert auf der Annahme, dass semantisch gehaltvolle Signale auf einer langsameren Zeitskala variieren als flüchtige und weniger aussagekräftige. Wie genau diese Annahme zu verstehen ist, zeigt ein einfaches Beispiel, welches [Ber06] entnommen ist. Abbildung 1.1 oben zeigt Ausschnitte aus einer Videosequenz, in welcher ein Affe zu sehen ist. Der menschliche Betrachter erkennt sofort, dass sich der Affe zunächst in der Nähe der Bildmitte befindet, sich dann nach links bewegt und schließlich den Bildausschnitt vom Betrachter aus gesehen zur linken Seite verlässt.

Unten links sind die Helligkeitswerte dreier ausgewählter Pixel der Sequenz über die Zeit aufgetragen. Es wird deutlich, dass die separate Betrachtung einzelner Pixelwerte keine Aussage über die Position des Affen im Bild zulässt, oder darüber, ob sich überhaupt ein Objekt im Bild befindet. Über die Zeit betrachtet variieren die Pixelwerte zudem deutlich stärker als die hypothetischen Signalen der Objektidentität (Befindet sich ein Objekt im Bild?) sowie der Objektposition (Wo im Bild befindet sich das Objekt?), welche unten rechts in Abbildung 1.1 zu sehen sind. Es ist also ein Verfahren gesucht, welches aus den Pixelwerten der Videosequenz – oder allgemeiner aus den sensorischen Daten einer beliebigen Sequenz – Signale extrahiert, welche semantisch höherwertige Informationen einer solchen Sequenz kodieren.

Die soeben angestellte Betrachtung macht deutlich, dass Langsamkeit hier nicht als Tiefpassfilterung zu verstehen ist, welche lediglich lokal operiert und daher keine der gesuchten Merkmale aus dem Bild extrahieren kann. Für eine geeignete Formulierung des Langsamkeitsprinzips muss also ein globaler Ansatz verfolgt werden, welcher alle sensorischen Werte in Anspruch nimmt, und aus diesen in geeigneter Form die gesuchten Signale extrahiert.

Formulierung des Langsamkeitsprinzips

Die ersten mathematischen Formulierungen des Langsamkeitsprinzips gehen auf [Föl91] und [Mit91] zurück. Die von Mitchison formulierte Zielfunktion besagt, dass die zeitliche Variation des Ausgangssignals, ausgedrückt durch das Quadrat der Ableitung des Signals, möglichst gering sein soll, und wird in ähnlicher Form auch in [Bec93] sowie bei der *Slow Feature Analysis (SFA)* [Wis98, WS02] verwendet. Die Herangehensweise von Mitchison sowie Becker besteht darin, diese Zielfunktion durch ein Gradientenverfahren zu minimieren, Mitchison beispielsweise leitet zu diesem Zweck eine anti-Hebbsche Lernregel her. Bei der Slow Feature Analysis wird die Zielfunktion um weitere Nebenbedingungen erweitert und durch ein geschlossenes Verfahren gelöst, welches auf der *Hauptkomponentenanalyse* (engl.: *Principal Component Analysis, PCA*) bzw. dem generalisierten Eigenwertproblem beruht.

Eine weitere Implementation des Langsamkeitsprinzips wird in [KED+01] vorgestellt. Auch hier wird ein Gradientenabstieg verwendet, jedoch angewandt auf eine andere Zielfunktion, deren Gradient analytisch berechnet wird. Es ist zu bemerken, dass das Prinzip der Langsamkeit hier als Prinzip der *zeitlichen Stabilität* (engl.: *temporal stability*) bezeichnet wird [WKV06].

Im Rahmen dieser Arbeit wird die eben erwähnte Slow Feature Analysis verwendet, welche sich aufgrund ihrer robusten und effizienten Implementation gut für die Verwendung im Bereich der Robotik eignet. Ausgehend von der allgemeinen Struktur des SFA-Verfahrens soll im folgenden Abschnitt die genaue Zielsetzung dieser Arbeit vorgestellt werden.

1.2 Zielsetzung

In dieser Arbeit werden zwei Hauptziele verfolgt: Zum einen sollen möglichst viele Aspekte der SFA im Hinblick auf ihre Verwendung in der humanoiden Robotik untersucht werden. Diese Betrachtungen sind vor allem theoretischer Natur und dienen dem Zweck, ein Verständnis für

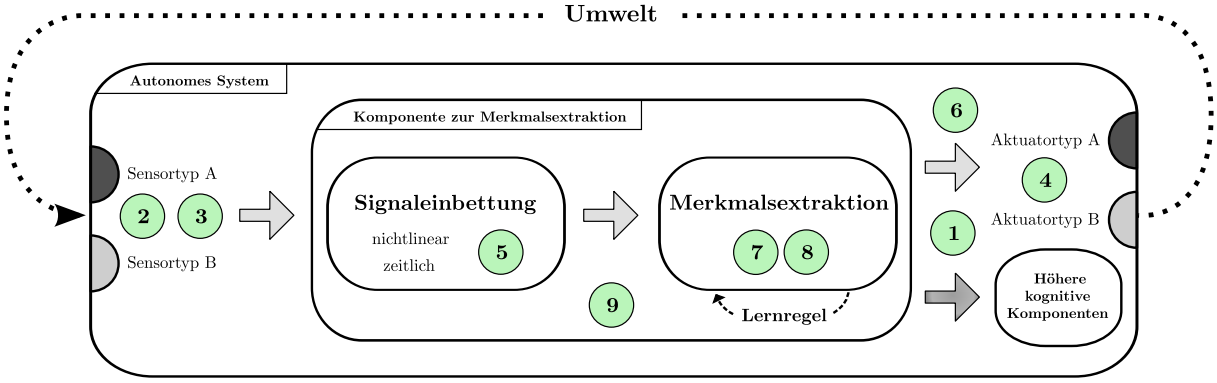


Abbildung 1.2: Schema eines Modells zur Merkmalsextraktion. Die Nummern verorten die in dieser Arbeit bearbeiteten Fragestellungen (siehe Text).

das Verfahren und seine Implementation zu bekommen.

Zum anderen sollen in dieser Arbeit praktische Anwendungen der SFA entwickelt und vorgestellt werden. Diese stellen nicht nur eine Verifikation der theoretischen Erkenntnisse dar, sondern sind – ganz im Sinne des *Embodiment*-Prinzips – essentiell für eine begründete Beurteilung der Eignung dieses Verfahrens für die Robotik.

Um die behandelten theoretischen und praktischen Fragestellungen besser zu verstehen, soll hier der SFA-Algorithmus zunächst in einem verallgemeinerten Kontext dargestellt werden. So unterliegt die SFA strukturell grundlegenden Prinzipien aus den Bereichen der Digitaltechnik, des maschinelles Lernens und auch der Neurobiologie. Alle diese Bereiche beschäftigen sich unter anderem mit der Fragestellung, wie informative Signale aus einem Rohdatenstrom extrahiert werden können. Diese Aufgabenstellung soll im folgenden *Merkmalsextraktion* genannt werden. Zwar ist dieser Begriff bereits in vielen Domänen wie beispielsweise der Signal- und Bildverarbeitung durch konkrete Verfahren belegt, soll hier jedoch in einem allgemeinen Sinne verstanden werden: So soll insbesondere auch eine Vorverarbeitung wie beispielsweise eine einfache Tiefpassfilterung des Signals als eine primitive Form der Merkmalsextraktion verstanden werden.

Abbildung 1.2 stellt ein Schema dar, welches ein in der Umwelt situiertes autonomes System zeigt. Dieses System verfügt ein oder mehrere *Komponenten zur Merkmalsextraktion* wie beispielsweise die SFA. Von diesen Komponenten können prinzipiell beliebig viele hintereinander und parallel geschaltet werden, wobei in der Regel die abstraktesten Merkmale am Ende einer solchen Hierarchie extrahiert werden. In der Abbildung ist exemplarisch eine prototypische Struktur einer Komponente zur Merkmalsextraktion abgebildet. Dabei ist anzumerken, dass das angegebene Schema keinen Anspruch auf Vollständigkeit erhebt, sondern versucht verschiedene Ideen und Eigenschaften von Systemen zur Merkmalsabstraktion auf einen Nenner zu bringen, indem eine Zweiteilung in eine *Signaleinbettung* und die eigentliche *Merkmalsextraktion* vorgenommen wird[1].

[1]Diese Zweiteilung, insbesondere die Hervorhebung der nichtlinearen Einbettung, ist für die Modellierung

Das System, in dieser Arbeit ein humanoider Roboter, kann seine Umwelt über Sensoren verschiedener Modalitäten wahrnehmen, wobei die Sensorwerte vom Zustand des Systems und der Umwelt abhängen. Die Werte der Sensoren werden an die Komponenten zur Merkmalsextraktion weitergeleitet, welche ein oder mehrere Ausgangssignale generieren. Diese Ausgangssignale, auch *Merkmale* oder *Features* genannt, werden an die Aktuatoren oder Module mit höheren kognitiven Funktionen gereicht. In Bezug auf die SFA ergeben sich damit sofort mehrere Fragestellungen:

1. Was für Signale extrahiert die SFA, wenn sie auf sensorische Daten eines humanoiden Roboters angewandt wird?

2. Wie unterscheiden sich die Ergebnisse bei der Anwendung der SFA auf verschiedene sensorische Modalitäten?

3. Welche Rolle spielt die ausgeführte Aktion des Roboters, wie reagiert die SFA auf statische und dynamische Bewegungen?

4. Wie können die extrahierten SFA-Signale verwendet werden? Eignen sie sich zur direkten Steuerung eines Roboters?

Es ist ergänzend zu erwähnen, dass ich mich in dieser Arbeit ausschließlich auf die Auswertung nicht-visueller Sensordaten beschränke, da es zum Zeitpunkt der Arbeit bereits eine große Anzahl an Anwendungen der SFA auf Bild- und Videodaten gibt.

Betrachten wir nun den Schritt der Einbettung des Eingangssignals näher. Dieser Schritt ist optional und kann auf verschiedene Weisen geschehen, und insbesondere die *nichtlineare* und die *zeitliche* Einbettung sind hervorzuheben: Bei der nichtlinearen Einbettung wird das Eingangsignal mittels einer nichtlinearen Funktion transformiert, so dass – einfach ausgedrückt – das Eingangssignal in einen höherdimensionalen Raum abgebildet wird und somit bestimmte Eigenschaften und Regularitäten besser zum Vorschein treten können. Die zeitliche Einbettung dagegen ermöglicht, Informationen aus der unmittelbaren Vergangenheit zu integrieren. Auch die Kombination von nichtlinearer und zeitlicher Einbettung ist möglich. Insgesamt kann man sich den Schritt der Einbettung als Aufblähung des Signalraumes vorstellen, die dem Zweck dient, dem eigentlichen Extraktionsschritt eine größere und qualitativ angereicherte Basis an Eingangssignalen zur Verfügung zu stellen. In Hinblick auf die SFA ergibt sich somit folgende Fragestellung:

---

vieler Verfahren aus den zuvor genannten Domänen üblich. Dazu nur einige Beispiele: Schon im künstlichen Neuronenmodell von McCulloch und Pitts [MP43] wurde die Nichtlinearität als wichtiges Modellelement hervorgehoben, und auch in der Neurowissenschaft gibt es Hinweise darauf, dass im Gehirn komplexe nichtlineare Operationen in den Dendriten durchgeführt werden [Mel94], während den Neuronen die Extraktion von Merkmalen zugeschrieben wird. Ebenso wurde in der Filtertechnik früh die Verwendung von Filtern auf Basis einer Linearkombination nichtlinearer Kernel in Betracht gezogen [Wie58]. Die nichtlineare Einbettung ist vor allem in Form des so genannten *Kernel Tricks* [ABR64] bekannt, und fand spätestens durch die Entwicklung von Support-Vektor-Maschinen [CV95] weite Verbreitung. Ebenso ist zeitliche Einbettung implizit in der Filtertechnik und explizit durch Arbeiten von Takens [Tak81] seit längerem bekannt. Dass die Wahl der Einbettung nicht explizit erfolgen muss, zeigen die so genannten *Echo State Networks* [Jae01] (auch *reservoir computing*), welche eine neuronale Struktur auf zufällige Weise generieren; in dieser Struktur können sich vielfältige Substrukturen ausbilden, die als nichtlineare und zeitliche Einbettungen fungieren.

5. Welche Möglichkeiten zur Signaleinbettung gibt es, und wie wirken sie sich auf die SFA aus?

Im nächsten Schritt erfolgt die eigentliche Detektion und Extraktion der Merkmale. Um sinnvolle Merkmale extrahieren zu können, muss diese Komponente hinreichend auf die eingebetteten Eingangsdaten angepasst sein. Dies kann entweder dadurch erfolgen, dass das Modul manuell auf die Eingangsdaten angepasst wird, oder indem Lernregeln angewandt werden, mit welchen das Modul trainiert wird. Man unterscheidet dabei grundsätzlich zwischen *unüberwachten* und *überwachten* Lernregeln: Unüberwacht bedeutet, dass der Lernalgorithmus nur ein Eingangssignal bzw. Trainingssignal zur Verfügung hat und selbst lernen muss, wie er eine bestimmte Eigenschaft – wie beispielsweise Langsamkeit – aus dem Signal extrahiert. Beim überwachten Lernen hingegen ist zusätzlich zum Trainingssignal ein Lösungssignal vorgegeben, welches das richtige Ergebnis für die gegebenen Trainingsdaten darstellt; der Lernalgorithmus versucht dann, aus den Trainingsdaten ein Signal zu extrahieren, welches dem Lösungssignal entspricht. Beiden Varianten liegt die Hypothese zugrunde, dass die erlernten Parameter zur Extraktion auch auf ungesehene, vom Trainingssignal verschiedene aber qualitativ ähnliche Daten anwendbar sind. Oftmals begegnet man dabei dem Problem der *Überanpassung*, dass also die gelernten Parameter gut auf die Trainingsdaten, aber nicht auf andere Daten anwendbar sind. Ebenso ist bei Lernverfahren nicht immer unmittelbar klar, was tatsächlich gelernt wurde, und wie bzw. unter welchen Voraussetzungen sich die Lösung für eine angedachte Anwendung eignet. Da es sich bei der SFA um ein unüberwachtes Lernverfahren handelt, müssen folgende Fragen untersucht werden:

6. Wie gut sind die von der SFA gelernten Parameter auf ungesehene Signale anwendbar?

7. Welche Methoden gibt es, um Lösungen der SFA zu analysieren und nachzuvollziehen?

Wie zuvor angedeutet, gibt es auch etliche andere Methoden zur Merkmalsextraktion, welche teils ähnliche aber auch andere Herangehensweisen als die SFA aufweisen. Daher soll in dieser Arbeit auch folgende Fragestellung beleuchtet werden:

8. Weist die SFA Verwandtschaft zu anderen Verfahren auf? Welche Erweiterungen der SFA gibt es und sind denkbar?

Eng verbunden damit ist auch die Frage, ob die biologische Motivation der SFA nicht nur von theoretischem Interesse ist, sondern ob das Verfahren auch durch künstliche neuronale Strukturen modelliert werden kann. Als letzte Frage ergibt sich also:

9. Wie kann die SFA in einem künstlichen neuronalen Modell implementiert werden?

1.3 Experimentalplattform

Um die soeben genannten Zielsetzungen verfolgen zu können, ist eine geeignete Roboterplattform zur Durchführung praktischer Untersuchungen notwendig. Zu diesem Zweck wird in dieser Arbeit die so genannte *A-Serie* verwendet, ein am Labor für Neurorobotik der Humboldt-Universität zu Berlin entwickelter humanoider Roboter. Abbildung 1.3 zeigt einen der Roboter

dieser Serie. Die A-Serie wurde im Rahmen des ALEAR-Projektes (*Artificial Language Evolution on Autonomous Robots*, `http://www.alear.eu`) entwickelt und zeichnet sich vor allem durch vielfältige und redundant ausgelegte sowohl propriozeptive als auch exterozeptive Sensorik aus: Zum einen besitzt die A-Serie Winkelsensoren für die Positionen der 21 Motoren sowie 16 über den Körper verteilte Beschleunigungssensoren. Zum anderen befindet sich im Kopf eine Kamera, deren Daten über einen am Rücken befestigten PDA verarbeitet werden. Zum Zeitpunkt dieser Arbeit existieren mehrere Roboter dieser Serie, von denen in dieser Arbeit namentlich die Roboter Aida, Aimee, Anita und April zum Einsatz kommen.

Abbildung 1.4 zeigt den schematischen Aufbau der A-Serie, bei welcher vor allem die acht so genannten *Accelboards*, welche durch grüne Pfeile gekennzeichnet sind, interessant sind. Auf jedem Accelboard befinden sich neben einer Recheneinheit, welche u. a. für die Ansteuerung der Motoren zuständig ist, zwei Beschleunigungssensoren, jeweils einer in x- und in y-Richtung; dabei sind die Sensoren in x-Richtung alle nach vorn in der sagittalen Ebene (von hinten nach vorne) ausgerichtet, während die Sensoren in y-Richtung teils auf senkrecht zur Transversalebene (also nach oben bzw. unten) und teils auf in der frontalen Ebene (links nach rechts, bzw. rechts nach links) auftretende Veränderungen reagieren. Abkürzend werden im folgenden die Sensoren in x-Richtung als *sagittal*, die senkrecht zur Transversalebene zeigenden Sensoren als *senkrecht* sowie die zu den Seiten ausgerichteten Sensoren als *frontal* bezeichnet.

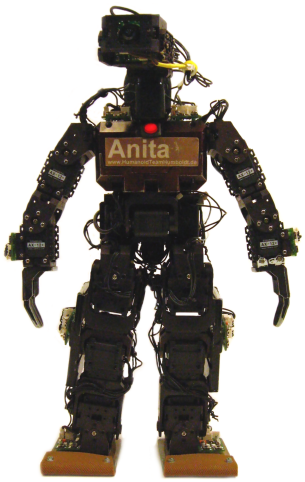


Abbildung 1.3: Anita, Roboter der A-Serie

Die Beschleunigungssensoren haben einen hohen Stellenwert für die Eigenwahrnehmung des Roboters, da sie im Gegensatz zu Gelenkstellungen und Motorposition auch die Lage des Roboters im Raum einfangen. Dies liegt selbstverständlich daran, dass durch die Erdgravitation eine konstante, gerichtete Kraft in Richtung Erdmittelpunkt auf den Roboter wirkt, und die Sensoren somit einen absoluten Bezug zur Umwelt haben.

Neben den Beschleunigungssensoren sind auch die eingangs erwähnten Motorpositionen von Interesse. Sie können als Sensoren mit relativem Bezug, nämlich zur weitgehend umweltunabhängigen Haltung des Roboters betrachtet werden.

Die Sensoren und Aktuatoren operieren in einem Wertebereich von 16 bit, die Werte sind ganzzahlig und vorzeichenbehaftet. Zur einfacheren Handhabung werden in dieser Arbeit alle Werte auf den Bereich $[-1; 1]$ normiert.

Neben den propriozeptiven Sensoren verfügt die verwendete humanoide Roboterplattform über eine Kamera. Im Rahmen dieser Arbeit wurden allerdings keine visuellen Daten verwendet, weswegen hier nicht weiter auf die Eigenschaften der Kamera eingegangen wird. Es wurden bereits ausführliche Untersuchungen zur Auswertung von visuellen Daten unter Verwendung der SFA, auch in der Robotik, vorgenommen, auf welche an dieser Stelle verwiesen werden soll [Nic06].

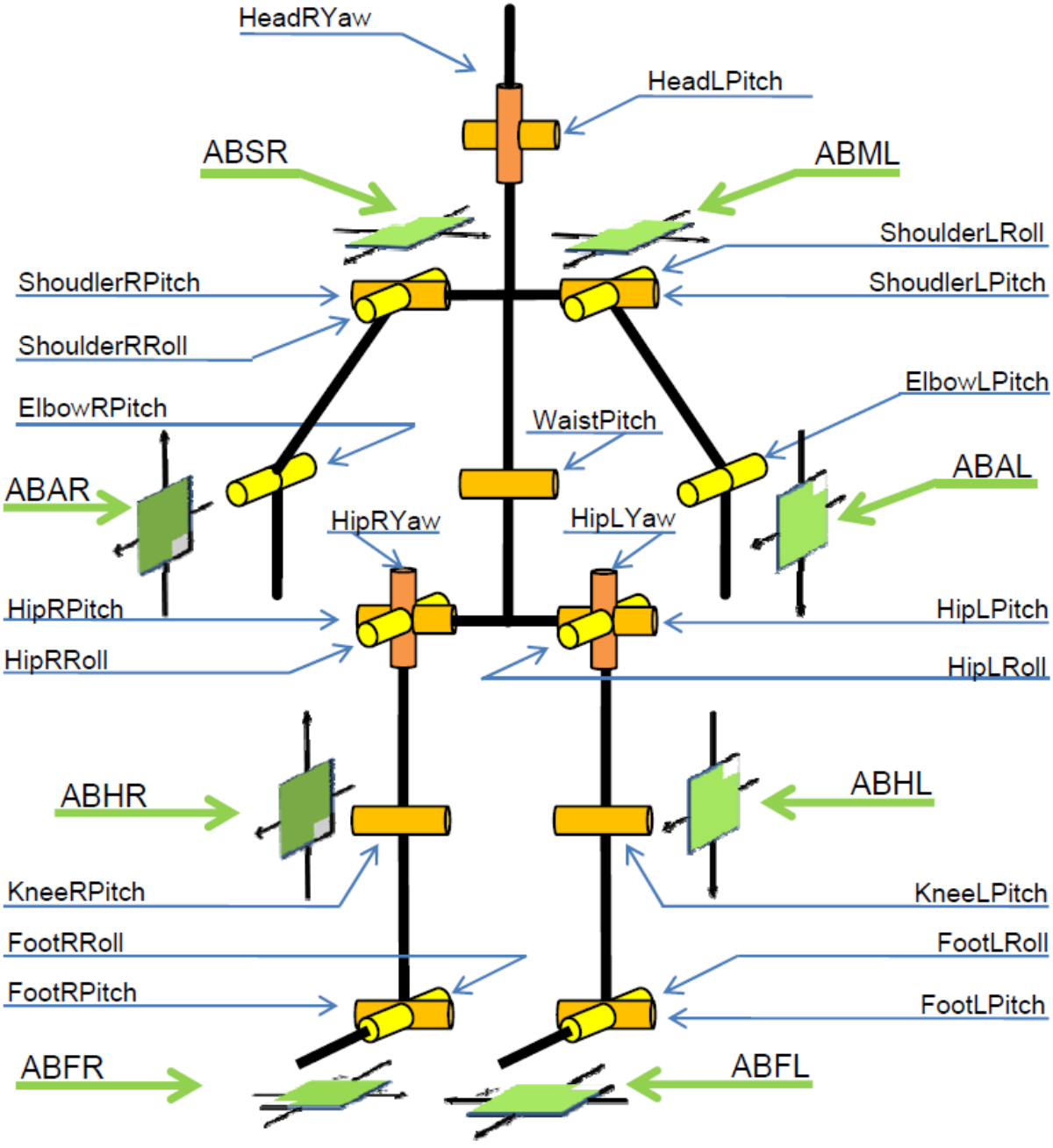


Abbildung 1.4: Schematischer Aufbau der humanoiden A-Serie-Roboter

Implementierte Bewegungsmuster

Zum Zeitpunkt dieser Arbeit sind bereits diverse Bewegungsmuster, wie Zeigebewegungen, Laufen, Aufstehen und Hinlegen etc. implementiert, was eine große Bandbreite an analysierbaren Daten zur Verfügung stellt. Bewegungsmuster wurden auf zwei verschiedene Arten entwickelt und implementiert: Zum einen durch die so genannte Keyframetechnik, zum anderen durch neuronal implementierte sensomotorische Schleifen. Die Keyframetechnik kann mit der Erstellung von Trickfilmen verglichen werden: Man wählt mehrere markante Posen, die Keyframes, der zu implementierenden Bewegung aus und interpoliert zwischen diesen Keyframes. Zusätzlich kann man einstellen, wie lange der Übergang zwischen zwei Keyframes dauern soll. Eine solche Verkettung von Keyframes heißt *Motionnetz*. Die Keyframetechnik hat den Vorteil, dass man in relativ kurzer Zeit komplexe Bewegungsmuster generieren kann. Der offensichtliche Nachteil besteht darin, dass die so generierten Bewegungen kaum adaptiv sind oder auf Umwelteinflüsse reagieren können, da in der Regel keine sensorische Information – außer der aktuellen Motorposition – einfließt.

Bei sensomotorischen Schleifen hingegen ist das Ziel, Verhalten durch reaktive Interaktion mit der Umwelt zu generieren: In jedem Zeitschritt wird über die Sensoren der Zustand der Welt abgefragt und in eine Steuerungsstruktur, in diesem Fall ein neuronales Netz, gegeben. Die Steuerungsstruktur erzeugt eine Ausgabe, welche im nächsten Zeitschritt auf die Aktuatoren gegeben wird. Durch die Aktion wird wieder der Zustand der Umwelt des Roboters verändert, welcher erneut vom Roboter abgefragt wird. Während die Konzeption und Analyse solcher System schwieriger ist als mit der Keyframetechnik, sind sie jedoch in der Regel adaptiver und spiegeln besser biologische Prinzipien von Bewegungsmustern wider.

Für weitere technische Details zur A-Serie sei auf den Anhang verwiesen.

1.4 Aufbau der Arbeit

Diese Arbeit gliedert sich in einen theoretischen und einen praktischen Teil. Der theoretische Teil beginnt mit Kapitel 2, welches sich der Slow Feature Analysis widmet. Das Ziel des Kapitels ist es, den Stand der Forschung zu beleuchten und eine ausführliche Übersicht über verwandte Verfahren und Erweiterungen der SFA zu geben.

In Kapitel 3 wird das künstliche Neuronenmodell beschrieben, welches bei der Ansteuerung des Roboters zum Einsatz kommt. Es werden die Grundlagen digitaler Filter und deren neuronale Implementation vorgestellt. Zudem widme ich mich der Fragestellung, wie der Ausführungsschritt der SFA in diesem neuronalen Modell formuliert werden kann. Zusätzlich wird eine alternative nichtlineare Einbettung für die SFA auf Basis von künstlichen Neuronen entwickelt.

Der zweite Teil wendet sich zwei praktischen Anwendungen der SFA auf der beschriebenen humanoiden Roboterplattform zu. Kapitel 4 beschäftigt sich mit der Frage, welche Ergebnisse die SFA für weitgehend statische Bewegungen liefert, und es wird untersucht, inwiefern die SFA sich zur Posenerkennung und Dimensionsreduktion des sensorischen Raums eignet. Kapitel 5 wendet sich einer dynamischen Bewegung des Roboters zu und stellt vor, wie die SFA zur Analyse und

Verbesserung eines zweibeinigen Laufmusters verwendet werden kann.

Die Arbeit schließt mit einer Zusammenfassung, welche eine Schlussbetrachtung offener Fragen sowie einen Ausblick in mögliche zukünftige Anwendungen der SFA in der Robotik beinhaltet.

1.5 Eigene Beiträge

Neben der Aufbereitung von Grundlagen sowie Arbeiten anderer Autoren leistet diese Arbeit folgende eigene Beiträge zur aktuellen Forschung:

- Entwicklung einer Methode zur Analyse von SFA-Komponenten durch Rückführung auf Quadriken (Abschnitt 2.4.4).
- Formulierung des Ausführungsschrittes der SFA in einem künstlichen Neuronenmodell (Abschnitt 3.2.1).
- Entwicklung einer alternativen nichtlinearen Einbettung für die SFA auf Basis von künstlichen Neuronen (Abschnitt 3.2.2).
- Anwendung der SFA zur Dimensionsreduktion und Erkennung von Posen eines humanoiden Roboters unter Verwendung nicht-visueller Sensordaten (Kapitel 4).
- Einsatz einer SFA-Komponente als Filterstruktur in einer sensomotorischen Schleife zur Generierung eines zweibeinigen Laufmusters (Kapitel 5).

Teil I

Theorie

Kapitel 2

Slow Feature Analysis (SFA)

Im folgenden Kapitel soll die in dieser Arbeit verwendete Implementation des Lernprinzips der Langsamkeit, die *Slow Feature Analysis (SFA)*, im Detail vorgestellt werden.

Die SFA wurde erstmals in [Wis98] präsentiert und zeichnet sich vor allem durch ihre geschlossene mathematische Formulierung und ihre Robustheit aus. Ausschlaggebend dafür ist die Beobachtung, dass sich ein Lernalgorithmus für das Langsamkeitsprinzip mithilfe der *Principal Component Analysis (PCA)* (siehe Abschnitt 2.5.1) realisieren lässt. Dadurch ist die SFA ein sehr stabiles Verfahren, welches garantiert und mit klar abschätzbarem Berechnungsaufwand eine optimale Lösung für das formulierte Lernproblem liefert. Im Gegensatz dazu beruhen, wie in der Einleitung erwähnt, andere Verfahren für das Langsamkeitsprinzip auf einem Gradientenabstieg und nichttrivialen Konvergenzbedingungen.

In [Ber06] konnte durch Umformulierung des mathematischen Lernproblems ein Algorithmus entwickelt werden, welcher qualitativ gleichwertig zur Originalimplementation ist, aber einen ursprünglich notwendigen, auf der PCA basierenden Vorverarbeitungsschritt in den Algorithmus integriert. In dieser Arbeit wird ausschließlich die neuere Variante, welche im folgenden Kapitel vorgestellt wird, verwendet. Für eine ausführliche Betrachtung der ursprünglichen Formulierung des Algorithmus sei auf [WS02] verwiesen.

Es ist zu erwähnen, dass die Berechnung langsamster Komponenten mittels SFA durchaus rechnenintensiv ist und sich daher im Gegensatz zu rein lokal operierenden Lernverfahren nur bedingt für den Einsatz auf ressourcenarmen Systemen eignet. Jedoch ist lediglich der Lernschritt der SFA numerisch aufwendig: Die SFA kann offline trainiert und online verwendet werden, wobei die Online-Anwendung im wesentlichen durch mehrere Skalar- und Matrizenmultiplikationen realisierbar ist. Allerdings ist in diesem Fall das Resultat nur eine Näherung der optimalen Lösung, welche von der Beschaffenheit der Trainings- und Testdaten abhängt.

Dieses Kapitel ist folgendermaßen aufgebaut: Zunächst wird die SFA und der ihr zugrundeliegende Algorithmus ausführlich erläutert, und es werden einige Details zur in dieser Arbeit verwendeten Implementation angeführt. Daraufhin werden Varianten, verwandte Verfahren und Erweiterungen der SFA beleuchtet, um ein besseres Verständnis für die Funktionsweise und Anwendbarkeit dieses Verfahrens zu erlangen. Im letzten Teil schließlich werden Anwendungen der SFA vorgestellt, die ihre Einsatzmöglichkeiten und Leistungsfähigkeit demonstrieren.

2.1 Lernproblem

Bei der SFA handelt es sich um einen unüberwachten Lernalgorithmus, welcher zu einem vektoriellen Eingangssignal $\mathbf{x}(t)$ eine Eingabe-Ausgabe-Funktion berechnet, welches das so langsam wie möglich über die Zeit variierende Ausgangssignal $\mathbf{y}(t)$ berechnet. Das Langsamkeitsprinzip lässt sich also als Optimierungsproblem mit Nebenbedingungen wie folgt formulieren[1]:

Gegeben: Eingangssignal $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_I(t)]^T$ im Zeitintervall $t \in [t_0, t_1]$

Gesucht: Eingabe-Ausgabe-Funktion $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_J(\mathbf{x})]^T$
derart, dass für das Ausgabesignal $\mathbf{y}(t) = [y_1(t), y_2(t), \ldots, y_J(t)]$ mit $y_j(t) := g_j(\mathbf{x}(t))$ für alle $j \in \{1, \ldots, J\}$ gilt:

$$\Delta_j := \Delta(y_j) := \langle \dot{y_j}^2 \rangle \qquad \text{ist minimal,} \tag{2.1}$$

unter folgenden zusätzlichen Nebenbedingungen

$$\langle y_j \rangle = 0 \qquad \text{(Mittelwertzentrierung),} \tag{2.2}$$

$$\langle y_j^2 \rangle = 1 \qquad \text{(Standardabweichung),} \tag{2.3}$$

$$\forall i < j: \quad \langle y_i y_j \rangle = 0 \qquad \text{(Dekorrelation)} \tag{2.4}$$

Die spitzen Klammern stehen dabei für die Mittelung über die Zeit:

$$\langle f(t) \rangle := \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} f(t) dt$$

Ist f mehrdimensional, so erfolgt die Mittelung komponentenweise, d. h.
$\langle [f_1, \ldots, f_n]^T \rangle = [\langle f_1 \rangle, \ldots, \langle f_n \rangle]^T$.

Die Gleichung (2.1) formuliert das Lernproblem, die zeitliche Variation des Eingangssignals zu minimieren: Der Mittelwert des Quadrates der Ableitung, also der Grad der zeitlichen Veränderung der Funktionswerte, soll minimal sein. Der Index j signalisiert, dass man ist in der Regel nicht nur an der langsamsten, sondern an den J langsamsten Komponenten interessiert ist.

Durch die Nebenbedingungen (2.2) sowie (2.3) wird die triviale Lösung $y_j(t) = const$ vermieden, da eine konstante Funktion eine Varianz von 0 hat. Gleichung (2.2) wurde lediglich eingeführt, um eindeutigere und besser vergleichbare Ergebnisse zu erhalten[2]. Ebenso kann Nebenbedingung (2.3) fallengelassen werden, wenn sie in die Zielfunktion (2.1) integriert wird, indem die rechte Seite durch $\langle y_j^2 \rangle$ geteilt wird. Gleichung (2.4) schließlich soll verhindern, dass die einzelnen langsamen Komponenten einander einfach reproduzieren, stattdessen sollen sie tatsächlich verschiedene Informationen enthalten. Zudem kann dadurch eine Ordnung auf den einzelnen Komponenten definiert werden, so dass y_1 das langsamste Signal ist, y_2 das zweitlangsamste etc.

[1] Wie üblich wird der in Klammern angegebene Zeitindex t weggelassen, sofern es die Lesbarkeit verbessert.
[2] Sonst müsste Gleichung (2.3) durch $\langle (y_j - \langle y_j \rangle)^2 \rangle = 1$ ersetzt werden

Im dieser Arbeit wird in der Regel von den langsamen *Komponenten* y_j als Ergebnis der SFA gesprochen. Im biologischen Kontext werden diese Komponenten auch als *Zellen* bezeichnet.

Das gegebene Lernproblem ist ein Optimierungsproblem aus der Variationsrechnung, dessen analytische Lösung aufwendig ist. In dieser Arbeit beschränke ich mich auf die in [Ber06] vorgestellte numerische Lösung mittels Slow Feature Analysis.

Der im folgenden Abschnitt vorgestellte Algorithmus beruht auf der Idee, die Eingabe-Ausgabe-Funktion $\mathbf{g}(\mathbf{x})$ derart einzuschränken, dass sie sich als Linearkombination einer endlichen Menge von Basisfunktionen darstellen lässt. Dadurch wird das Problem erheblich vereinfacht. Werden als Basisfunktionen *nichtlineare* Funktionen gewählt, ist der Algorithmus imstande, auch nichtlineare Merkmale aus den Eingabedaten zu extrahieren. Die Verfahrensweise ähnelt der des so genannten *Kernel-Tricks* [ABR64], bei welchem die Eingangssignale in einen nichtlinearen, in der Regel höherdimensionalen Zustandsraum abgebildet werden[3]. Diese Methode wird bevorzugt, da die Anwendung eines linearen Verfahrens auf den neuen, nichtlinearen Zustandsraum äquivalent zur Anwendung eines nichtlinearen Verfahrens auf den ursprünglichen Zustandsraum ist, lineare Verfahren aber wesentlich besser zu beherrschen sind.

Durch die Einschränkung der Eingabe-Ausgabe-Funktion gelingt es, immer ein globales Optimum im Sinne des vereinfachten Lernproblems zu finden. Der Algorithmus kann als Lernalgorithmus aufgefasst werden, da er zuerst auf einer Sequenz von Trainingsdaten lernt und dann auf ungesehene Daten angewandt werden kann. Für die Trainingsdaten berechnet die gelernte Eingabe-Ausgabe-Funktion die optimale Lösung, angewandt auf ungesehene Daten stellt sie eine Näherung der optimalen langsamsten Komponenten dar – natürlich unter der Voraussetzung, dass die ungesehenen Daten hinreichende Ähnlichkeit zu den Trainingsdaten aufweisen.

2.2 Vereinfachtes Lernproblem

Betrachten wir erneut das Eingangssignal $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_I(t)]^T$ sowie die gesuchte Eingabe-Ausgabe-Funktion $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_J(\mathbf{x})]^T$. Jede Komponente der Eingabe-Ausgabe-Funktion $g_j(\mathbf{x})$ wird nun als gewichtete Summe über einer Menge von K nichtlinearen Funktionen $h_k(\mathbf{x})$ definiert:

$$g_j(\mathbf{x}) := \sum_{k=1}^{K} w_{jk} h_k(\mathbf{x}).$$

Dieser Schritt stellt eine nichtlineare Expansion dar. Zur Vereinfachung wird abkürzend die Schreibweise $\mathbf{z}(t) := \mathbf{h}(\mathbf{x}(t))$ eingeführt. Das Optimierungsproblem kann nun als linear in den Komponenten von $\mathbf{z}$ betrachtet werden, da nur noch die Gewichte $\mathbf{w}_j = [w_{j1}, \ldots, w_{jK}]^T$ errechnet bzw. gelernt werden müssen.

Ersetzt man in Gleichung (2.1) $\dot{y_j}$ durch sein linearisiertes Pendant, so erhält man

$$\Delta(y_j) = \langle {y_j}^2 \rangle = \langle (\mathbf{w}_j^T \dot{\mathbf{z}}(t))^2 \rangle = \mathbf{w}_j^T \langle \dot{\mathbf{z}} \dot{\mathbf{z}}^T \rangle \mathbf{w}_j \tag{2.5}$$

[3] In Abschnitt 2.5.7 wird näher auf eine Erweiterung der SFA, die so genannte Kernel-SFA eingegangen.

Damit die Nebenbedingungen aus den Gleichungen 2.2-2.4 erfüllt sind, müssen folgende Voraussetzungen erfüllt sein:

1. Die Komponenten von $\mathbf{z}(t)$ haben einen Mittelwert von 0 sowie eine Varianz von 1 und sind dekorreliert.
2. Die Gewichtsvektoren sind normiert, also $||\mathbf{w}_j|| = \sqrt{w_{j1}^2 + \ldots + w_{jK}^2} = 1$.
   (Insbesondere gilt damit auch, dass $w_{j1}^2 + \ldots + w_{jK}^2 = \mathbf{w}_j^T \mathbf{w}_j = 1$, also das Skalarprodukt eines Gewichtsvektors mit sich selbst ebenfalls 1 ist.)
3. Die Menge der Gewichtsvektoren ist orthonormal, also orthogonal und normiert.

Der im nächsten Abschnitt vorgestellte Algorithmus erfüllt diese Voraussetzungen. Dann lässt sich nachprüfen, dass auch die Nebenbedingungen erfüllt sind:

$$\langle y_j \rangle = \mathbf{w}_j^T \underbrace{\langle \mathbf{z} \rangle}_{=0} = 0 \tag{2.6}$$

$$\langle y_j^2 \rangle = \mathbf{w}_j^T \underbrace{\langle \mathbf{z}\mathbf{z}^\mathbf{T} \rangle}_{=\mathbf{I}} \mathbf{w}_j = \mathbf{w}_j^T \mathbf{w}_j = 1 \tag{2.7}$$

$$\forall i < j: \quad \langle y_i y_j \rangle = \mathbf{w}_i^T \underbrace{\langle \mathbf{z}\mathbf{z}^\mathbf{T} \rangle}_{=\mathbf{I}} \mathbf{w}_j = \mathbf{w}_i^T \mathbf{w}_j = 0 \tag{2.8}$$

$\langle \mathbf{z}\mathbf{z}^\mathbf{T} \rangle$ ist gerade die Kovarianzmatrix von $\mathbf{z}$ und entspricht wegen Voraussetzung 1 der Identitätsmatrix $\mathbf{I}$.

Unter den gegebenen Nebenbedingungen lässt sich das neue Problem darauf reduzieren, die normierten Eigenvektoren sowie die zugehörigen Eigenwerte von $\langle \dot{\mathbf{z}}\dot{\mathbf{z}}^\mathbf{T} \rangle$ zu finden, die $\Delta(y_j)$ minimieren:

$$\Delta(y_j) = \mathbf{w}_j^T \langle \dot{\mathbf{z}}\dot{\mathbf{z}}^T \rangle \mathbf{w}_j = \lambda_j \mathbf{w}_j^T \mathbf{w}_j = \lambda_j \tag{2.9}$$

Sind $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_K$ die normierten Eigenvektoren, aufsteigend geordnet entsprechend ihrer Eigenwerte $\lambda_1, \lambda_2, \ldots, \lambda_K$, so ist $\mathbf{w}_1$ der Parameter für die langsamste Komponente, $\mathbf{w}_2$ der für die zweitlangsamste etc. Damit ergeben sich die Lösungen für das Optimierungsproblem.

2.3 Algorithmus

Der Algorithmus, wie er in [Ber06] formuliert ist, setzt voraus, dass das Eingangssignal mittelwertzentriert ist. Bezeichnen wir das ursprüngliche Eingangssignal bzw. das bereits nichtlinear expandierte Signal mit $\tilde{\mathbf{x}}$, so berechnet sich das mittelwertzentrierte Signal $\mathbf{x}$ wie folgt:

$$\mathbf{x} = \tilde{\mathbf{x}} - \langle \tilde{\mathbf{x}} \rangle \tag{2.10}$$

Dadurch ist Nebenbedingung (2.2) automatisch erfüllt:

$$\langle y_j \rangle = \langle \mathbf{w}_j^T \mathbf{x} \rangle = \mathbf{w}_j^T \langle \mathbf{x} \rangle = 0 \tag{2.11}$$

Das Zielfunktion 2.1 sowie die Nebenbedingungen (2.3) und (2.4) können nun umgeschrieben werden zu:

$$\begin{aligned} \Delta(y_j) &= \langle \dot{y_j}^2 \rangle && (2.12)\\ &= \mathbf{w}_j^T \langle \dot{\mathbf{x}} \dot{\mathbf{x}}^T \rangle \mathbf{w}_j && (2.13)\\ &=: \mathbf{w}_j^T \mathbf{A} \mathbf{w}_j && (2.14) \end{aligned}$$

und für $i < j$

$$\begin{aligned} \langle y_i y_j \rangle &= \mathbf{w}_i^T \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{w}_j^T && (2.15)\\ &=: \mathbf{w}_i^T \mathbf{B} \mathbf{w}_j. && (2.16) \end{aligned}$$

Nun lässt sich wie bereits in Abschnitt 2.1 erwähnt Nebenbedingung (2.3) in die Zielfunktion (2.1) integrieren:

$$\Delta(y_j) = \frac{\langle \dot{y_j}^2 \rangle}{\langle y_j^2 \rangle} = \frac{\mathbf{w}_j^T \mathbf{A} \mathbf{w}_j}{\mathbf{w}_j^T \mathbf{B} \mathbf{w}_j} \tag{2.17}$$

Aus der linearen Algebra ist bekannt, dass sich dieses Problem als generalisiertes Eigenwertproblem auffassen lässt, da

$$\mathbf{AW} = \mathbf{BW}\mathbf{\Lambda}, \tag{2.18}$$

wobei $\mathbf{W} = [w_1, \ldots, w_n]$ die Matrix der generalisierten Eigenvektoren und $\mathbf{\Lambda}$ die Diagonalmatrix der zugehörigen Eigenwerte $\lambda_1, \ldots, \lambda_n$ ist. $\mathbf{A}$ und $\mathbf{B}$ sind jeweils Kovarianzmatrizen von $\dot{\mathbf{x}}$ bzw. $\mathbf{x}$ und sind somit symmetrisch. Es lässt sich zeigen, dass die Eigenvektoren symmetrischer Matrizen reellwertig und orthogonal sind. Werden die Eigenvektoren zudem normalisiert, lässt sich leicht zeigen, dass somit auch Nebenbedingungen (2.3) und (2.4) erfüllt sind:

$$\begin{aligned} \langle y_j^2 \rangle &= \mathbf{w}_j^T \mathbf{B} \mathbf{w}_j = 1, && (2.19)\\ \langle y_i y_j \rangle &= \mathbf{w}_i^T \mathbf{B} \mathbf{w}_j = 0, \qquad i \neq j && (2.20) \end{aligned}$$

Durch Einsetzen von (2.18) in (2.17) ergibt sich dann:

$$\Delta(y_j) = \lambda_j. \tag{2.21}$$

Werden die Eigenvektoren aufsteigend gemäß ihrer Eigenwerte sortiert, ergeben sich die Koeffizientenvektoren für die langsamsten Komponenten.

Zusammenfassend lässt sich der SFA-Algorithmus auf folgende Weise beschreiben:

1. **Nichtlineare Expansion:**
   Expandiere das Eingangssignal und zentriere das expandierte Signal bzgl. seines Mittelwerts:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) - \langle \mathbf{h}(\mathbf{x}) \rangle \tag{2.22}$$

2. **Extraktion der langsamen Komponenten:**
   Löse das generalisierte Eigenwertproblem

$$\mathbf{AW} = \mathbf{BW\Lambda} \tag{2.23}$$
$$\text{mit} \quad \mathbf{A} := \langle \dot{\mathbf{z}}\dot{\mathbf{z}}^{\mathbf{T}} \rangle \tag{2.24}$$
$$\text{und} \quad \mathbf{B} := \langle \mathbf{z}\mathbf{z}^{\mathbf{T}} \rangle \tag{2.25}$$

   und sortiere die Eigenvektoren aufsteigend entsprechend ihrer generalisierten Eigenwerte $\lambda_1, \ldots, \lambda_n$. Definiere die Eingabe-Ausgabe-Funktion als

$$\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_J(\mathbf{x})]^T \tag{2.26}$$
$$\text{mit} \quad g_j(\mathbf{x}) = \mathbf{w}_j^T(\mathbf{h}(\mathbf{x}) - \langle \mathbf{h}(\mathbf{x}) \rangle) \tag{2.27}$$

   und die Ausgabe als

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t)), \tag{2.28}$$

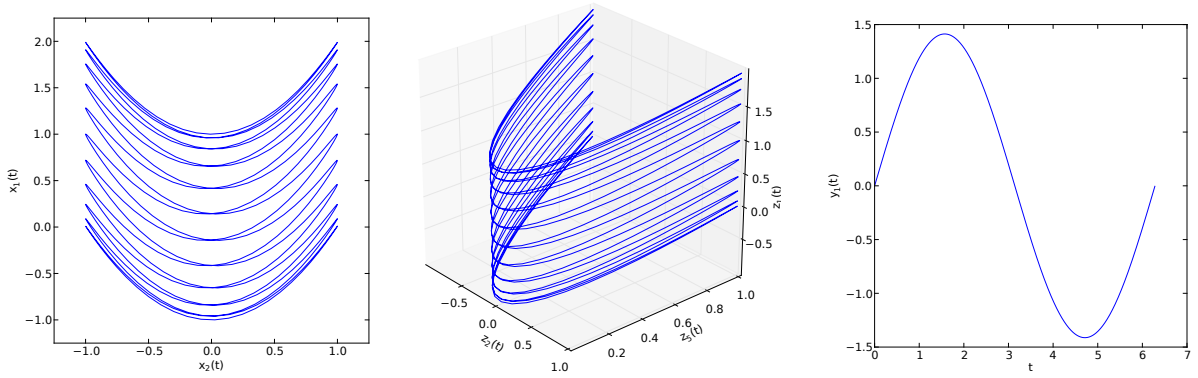
   wobei y_j für alle $j = 1, \ldots, J$ die Nebenbedingungen (2.2)-(2.4) erfüllt.

3. **Wiederholung:**
   Bei Bedarf kann das Ausgabesignal $\mathbf{y}(t)$, oder die Kombination verschiedener Komponenten erneut als Eingabesignal $\mathbf{x}(t)$ für den Lernalgorithmus verwendet werden. Dann weiter bei Schritt 1.

   Die iterative Anwendung des Algorithmus auf den zuvor erhaltenen Signalen ermöglicht das Erlernen von Komponenten, welche nicht durch lineare oder quadratische Polynome angenähert werden können. Bei einer zweimaligen Anwendung der quadratischen SFA können Komponenten dritter und vierter Ordnung, bei dreimaliger Anwendung Komponenten bis zu achter Ordnung, etc., gefunden werden. Das Erlernen von Komponenten höherer Ordnung ließe sich auch durch eine nichtlineare Expansion höherer Ordnung durchführen, bei welcher aber die Dimensionalität des Phasenraums exponentiell ansteigen würde. Die iterative Anwendung hingegen ist weitaus weniger aufwendig, allerdings auch nur, wenn bei jeder Wiederholung nur eine beschränkte Auswahl von Ausgabekomponenten in die nächste SFA-Runde gegeben wird.

   Wird die SFA mehrmalig hintereinander angewandt, spreche ich in dieser Arbeit dabei auch von einzelnen SFA-*Einheiten* und SFA-*Iterationen*, die Anwendung wird als (quadratische) SFA *mit Wiederholung* bezeichnet.

Der Schritt der nichtlinearen Expansion ist optional. Wird er nicht durchgeführt, so wird der Algorithmus als *lineare SFA* oder auch *SFA*^{1} bezeichnet. Wird eine quadratische Expansion durchgeführt, wird der resultierende Algorithmus als *quadratische SFA* oder *SFA*^{2} bezeichnet.



(a) Eingabedaten (b) Expansion (c) Langsamste Komponente

Abbildung 2.1: Plots nach verschiedenen Schritten der SFA. **a)** Das Eingangssignal $\mathbf{x}(t)$. **b)** Expandiertes Eingabesignal, in der Grafik sind $z_1(t) := x_1(t)$, $z_2(t) := x_2(t)$ und $z_5(t) := x_2^2(t)$ zu sehen. **c)** Ausgabesignal $\mathbf{y}_1(t)$, welches der normalisierten Version (Varianz von 1) der Funktion $sin(t)$ entspricht.

2.3.1 Implementation und Anwendungsbeispiel

Betrachten wir an einem einfachen Beispiel die Arbeitsweise der SFA. Abbildung 2.1 zeigt dabei das Eingangs-, das Ausgangssignal sowie zwei Zwischenschritte eines Beispiels, welches [WS02] entnommen ist. Als Eingabesignal werden die Funktionen $x_1(t) = \sin(t)+\cos^2(11t)$ sowie $x_2(t) = \cos(11t)$ im Intervall $t \in [0, 2\pi]$ verwendet, welche in a) aufgetragen sind. b) zeigt ausgewählte Komponenten des Signals nach der Expansion. Offensichtlich ist die langsamste Komponente des Eingangsvektors $\mathbf{x}(t) = [x_1, x_2]^T$, wie sich leicht nachrechnen lässt, die Funktion $y(t) = x_1(t) - x_2(t)^2 = \sin(t)$. Tatsächlich wird diese Komponente auch durch die SFA gefunden (siehe c).

Zu beachten ist, dass in diesem Beispiel ebenso wie in allen Anwendungen der SFA in dieser Arbeit die Ableitung numerisch angenähert wird. Es wird dazu die einfache Differenz $\dot{\mathbf{x}}(t) := \mathbf{x}(t) - \mathbf{x}(t-1)$ verwendet.

Für alle Experimente und Simulationen wurde das in Python implementierte *Modular Toolkit for Data Processing (MDP)* [ZWWB09] verwendet, welches eine große Anzahl an implementierten Verfahren zum maschinellen Lernen, zur Dimensionsreduktion etc. zur Verfügung stellt. Wie im Anhang B aufgelistet, sind im Laufe dieser Arbeit verschiedene auf dem MDP-Toolkit basierende Module entstanden.

Es ist zu beachten, dass das Vorzeichen einer langsamen Komponente willkürlich ist. Dies liegt daran, dass die Eigendekomposition, auf welcher der SFA-Algorithmus beruht, keine Präferenz bzgl. des Vorzeichens der Lösungen hat; genauer gesagt ist auch das negierte Pendant eines Eigenvektors einer Matrix $\mathbf{A}$ wieder ein Eigenvektor von $\mathbf{A}$.

Eine problematische Eigenschaft der verfügbaren Implementation ist, dass vor allem bei manchen synthetischen Daten die Kovarianzmatrizen $\mathbf{A}$ oder $\mathbf{B}$ singulär werden können, wodurch ein oder mehrere Eigenwerte Null sind. In diesem Fall können die Gewichtsvektoren $\mathbf{w}$ nicht bestimmt werden. Diese numerische Instabilität kann auftreten, wenn die Daten eine hohe Regularität aufweisen und dimensional überrepräsentiert sind, d. h. mehrere Einzelsignale stark untereinander

korreliert sind und daher die echte Dimensionalität der Daten wesentlich geringer ist. Sie tritt bei allen Implementationen auf, welche auf dem generalisierten Eigenwertproblem beruhen, worauf in [Kon09] hingewiesen wird.

In der Regel hat man zwei Möglichkeiten zur Umgehung dieses Problems: Eine Möglichkeit ist, die Daten mit gaußschem Rauschen von möglichst geringer Amplitude zu versehen, wie beispielsweise in [FSW07] vorgestellt. Eine andere Möglichkeit ist, vor der Expansion eine PCA auf die Daten anzuwenden und nur Dimensionen mit einer hohen Varianz in die SFA zu geben. Die PCA wird im in Kürze folgenden Abschnitt 2.5.1 genauer vorgestellt, wenn die Verwandtschaft der SFA zu anderen Verfahren beleuchtet wird.

2.3.2 Optimale Komponenten

Bevor auf weitere Variationen und Anwendungen der SFA eingegangen wird, soll die Analyse optimaler Komponenten, wie sie in [Wis03a] durchgeführt wurde, erwähnt werden. In der genannten Arbeit wird die Fragestellung behandelt, welche Form optimale langsamste Komponenten annehmen, wenn keinerlei oder nur wenige gezielte Anforderungen an die Eingabedaten gestellt werden. Es wird gezeigt, dass optimale langsamste Komponenten die Form harmonischer Oszillationen annehmen; bei langsameren Komponenten ist die Frequenz dieser Oszillationen entsprechend niedriger als bei weniger langsamen. Insbesondere können diese optimalen Komponenten auch in der Praxis auftreten, wenn eine extreme Überanpassung an die Daten erfolgt. Zudem wird gezeigt, dass die genaue Trainingsprozedur, also wie lange beispielsweise die einzelnen Trainingsdaten präsentiert werden und v. a. ob verschiedene Trainingsbeispiele gleich lang präsentiert werden, ausschlaggebend für ein mögliches Auftreten dieser optimalen Komponenten ist. In dieser Arbeit wird sich zeigen, dass eine Überanpassung wahrscheinlicher ist, je weniger Trainingsdaten benutzt und je mehr SFA-Iterationen ausgeführt werden.

2.4 Methoden zur Analyse von SFA-Komponenten

2.4.1 η-Wert

Neben dem Δ-Wert (2.1) wird in [WS02] ein intuitiveres empirischeres Maß für die Langsamkeit, der η-Wert vorgestellt:

$$\eta(y(t)) = \frac{T}{2\pi}\sqrt{\Delta(y(t))} \tag{2.29}$$

für $t \in [t_0, t_0+T]$. Diese Gleichung gibt für die reine Sinusschwingung $y(t) := \sqrt{2}\sin(n2\pi t/T)$ mit einer natürlichen Zahl n von Oszillationen gerade die Anzahl an Oszillationen zurück, also $\eta(y) = n$. Je langsamer die Komponente ist, desto kleiner ist auch dieser Wert. Testsignale, die nur approximativ normalisiert worden sind, werden vor der Berechnung exakt normalisiert (beispielsweise mittels eines *Whitenings*, siehe 2.5.1), damit der η-Wert nicht durch einen Skalierungsfaktor verfälscht wird.

2.4.2 Korrelationskoeffizient nach Pearson

Als Maß für die Ähnlichkeit von zwei Signalen wird der Korrelationskoeffizient nach Pearson verwendet, welcher den linearen Zusammenhang zweier Signale angibt. Dieses Maß erweist sich als nützlich, wenn man z. B. herausfinden möchte, welche Sensorwerte welchen von der SFA errechneten langsamen Komponenten ähneln, oder um bei einer mehrstufigen SFA gleiche Komponenten, die nach mehreren SFA-Iterationen in anderer Reihenfolge ausgegeben werden, ausfindig zu machen. Der Korrelationskoeffizient liegt immer innerhalb des Intervalls $[-1;1]$ und ergibt sich durch

$$\rho := \frac{\mathrm{Cov}(X_1, X_1)}{\sqrt{\mathrm{Var}(X_1)}\sqrt{\mathrm{Var}(X_2)}} \tag{2.30}$$

wobei X_1 und X_2 zwei stochastische Zufallsvariablen sind, $\mathrm{Var}(X_i)$ die Varianz von X_i, $(i = 1, 2)$ und $\mathrm{Cov}(X_1, X_2)$ die Kovarianz von X_1 und X_2 bezeichnen. In den meisten Fällen ist lediglich der Betrag von Interesse. Ein absoluter Wert nahe 1 bedeutet hohe Korrelation, ein Wert nahe 0 geringe.

Für zwei eindimensionale, zeitdiskrete Signale $x_1 := x_1(t)$ und $x_2 := x_2(t)$ lässt sich ρ folgendermaßen berechnen:

$$\rho = \frac{\langle (x_1 - \langle x_1 \rangle)(x_2 - \langle x_2 \rangle) \rangle}{\sqrt{\langle (x_1 - \langle x_1 \rangle)^2 \rangle}\sqrt{\langle (x_2 - \langle x_2 \rangle)^2 \rangle}}, \tag{2.31}$$

wobei die spitzen Klammern wie zuvor die Mittelung über die Zeit bezeichnen.

Es ist zu bemerken, dass der Korrelationskoeffizient im folgenden nur als Richtwert verwendet wird, um Hinweise auf die Ähnlichkeit zwischen Sensorwerten, SFA-Komponenten sowie zwischen Komponenten verschiedener SFA-Einheiten ausfindig zu machen. Die Einschränkung der Aussagekraft des Korrelationskoeffizienten ist aus zweierlei Gründen notwendig: Zum einen kann zwischen den Sensorwerten und den SFA-Komponenten wegen der quadratischen Expansion der SFA-Komponente ein starker quadratischer Zusammenhang auftreten, welcher sich unter Umständen nicht im Korrelationskoeffizienten niederschlägt. Zum anderen kann ohne eine formale Einführung des Signifikanzbegriffs selbst bei hohem Korrelationskoeffizienten streng genommen nicht automatisch von einem statistischen Zusammenhang geredet werden. Nichtsdestotrotz ist die intuitive Verwendung des Korrelationskoeffizienten ungemein hilfreich, um auf effiziente Weise eine große Menge an verschiedenen Signalen zumindest näherungsweise miteinander zu vergleichen. Um SFA-Komponenten exakt zu begutachten und Aussagen über ihre Ähnlichkeit untereinander bzw. ihre Abhängigkeit von Sensorwerten zu treffen, wird die im nächsten Abschnitt vorgestellte Quadratische-Form-Analyse verwendet.

2.4.3 Quadratische Form

Während die zuvor vorgestellten Analysemethoden eher darauf abzielen, Aussagen über die von der SFA berechneten langsamsten Komponenten zu treffen, bietet es sich an, direkt die Koeffizientenmatrix bzw. die Komponenten der Eingabe-Ausgabe-Funktion $g_j(\mathbf{x})$ zu betrachten. In [BW06] und [BW07] wird dazu eine allgemein anwendbare Methode zur Analyse quadratischer

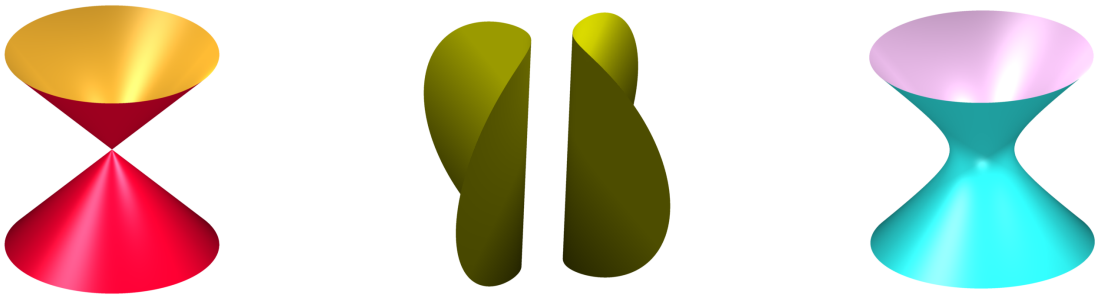


Abbildung 2.2: Verschiedene Quadriken: Ein Doppelkegel, ein hyperbolischer Zylinder und ein einschaliges Hyperboloid.

Formen vorgestellt, d. h. von inhomogenen quadratischen Gleichungen der Form:

$$y_j = g_j(\mathbf{x}) = \underbrace{\frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}}_{\text{quadratischer Teil}} + \underbrace{\mathbf{f}^T\mathbf{x}}_{\text{linearer Teil}} + c. \tag{2.32}$$

Dabei ist $\mathbf{x}$ das n-dimensionale (mittelwertzentrierte) Eingangssignal, $\mathbf{H}$ eine $n \times n$-Matrix, $\mathbf{f}$ ein n-dimensionaler Koeffizientenvektor und c eine Konstante. Wie sich zeigen lässt, ist jede durch die SFA berechnete Eingabe-Ausgabe-Funktion in dieser Form darstellbar.

Ausgehend von dieser Form können nun die optimalen Stimuli für diese Gleichung berechnet werden, d. h. die Vektoren $\mathbf{x}^+$ und $\mathbf{x}^-$, welche g unter der Bedingung $||\mathbf{x}^+|| = ||\mathbf{x}^-|| = r$ für ein festes $r \in \mathbb{R}$ maximieren bzw. minimieren. Die Normierung des Eingangsvektors ist notwendig, da sonst unendlich große bzw. kleine $\mathbf{x}$ die Gleichung optimieren würden. Wird für r ein großer Wert gewählt, dominiert der quadratische Teil, sonst der lineare. In der Regel wählt man r als Mittelwert der Norm der Eingangsvektoren, damit $\mathbf{x}^+$ und $\mathbf{x}^-$ repräsentativ für die Eingabedaten sind.

Die gesamte Betrachtung kann zunächst auch summandenweise erfolgen: Für $r = 1$ können die Eigenvektoren von $\mathbf{H}$ betrachtet werden, wobei der Eigenvektor mit dem größten positiven (negativen) Eigenwert gerade der maximal exzitatorische (inhibitorische) Stimulus für den quadratischen Term ist. Die entsprechenden exzitatorischen und inhibitorischen Einflüsse für den linearen Term lassen sich direkt an den Koeffizienten f ablesen.

In den erwähnten Artikeln wird zudem eine Methode vorgeschlagen, um für beliebige r die optimalen Stimuli der gesamten Gleichung zu berechnen. Zu beachten ist allerdings, dass die die vorgestellte Analyse nur für quadratische SFA ohne Wiederholung sinnvoll ist.

2.4.4 Quadriken

Eine weitere Sichtweise auf die SFA stammt aus dem Bereich der analytischen Geometrie und lässt sich direkt aus der quadratischen Form herleiten. Bereits in der ersten Publikation zur SFA wird darauf hingewiesen, dass die SFA *Invariance Manifolds*, zu deutsch etwa *Invarianzmannigfaltigkeiten*, aus dem Eingangssignal extrahiert [Wis98]. Diese Mannigfaltigkeiten entsprechen bei der quadratischen SFA so genannten *Quadriken*.

Betrachten wir noch einmal Gleichung 2.32 aus dem letzten Abschnitt und setzen sie gleich einem Wert μ_j; bringen wir diesen Wert auf die andere Seite und setzen die Gleichung gleich Null, so ergibt sich folgende homogene quadratische Gleichung:

$$\frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{f}^T\mathbf{x} + (c - \mu_j) = 0. \tag{2.33}$$

Alle Nullstellen dieser Gleichung beschreiben nun eine Hyperfläche, welche aufgrund der quadratischen Form der Gleichung Quadrik genannt wird. Abhängig von den Werten der Koeffizienten sowie der Dimension I von $\mathbf{x}$ ergeben sich als Quadriken verschiedene geometrische Figuren. Ist $I = 2$, so entsprechen die Quadriken gerade den Kegelschnitten, d. h. den Kurven, die entstehen, wenn man die Oberfläche eines unendlichen Kegels oder Doppelkegels mit einer Ebene schneidet. In Abbildung 2.2 sind verschiedene Quadriken für $I = 3$ zu sehen.

Eine SFA-Komponente entspricht genau genommen nicht einer Quadrik, sondern einem Quadrikenbüschel: Wird ein bestimmter Ausgangswert μ_j festgehalten, so bilden alle Eingabewerte $\mathbf{x}$, für welche $y_j(\mathbf{x}) = \mu_j$ bzw. $y_j(\mathbf{x}) - \mu_j = 0$ gilt, eine Quadrik.

Tatsächlich werden quadratische Formen und Quadriken in der Robotik verwendet, um Invarianzen bzgl. des Verhaltens eines Roboter zu modellieren. So wird in [Sel05] gezeigt, dass Quadriken existieren, die invariant für bestimmte Bewegungen eines Roboterarmes mit mehreren Drehgelenken sind. In [HKHM11] wird gezeigt, wie Quadriken als Kernel-Funktionen genutzt werden können, um mit Hilfe von *Quadrik-repräsentierenden Neuronen (QREN)* zu lernen, wenn ein Roboter eine bestimmtes Verhalten ausführt. Diese QRENs können mittels einer einfachen Backpropagation-Lernregel trainiert werden, um die Invarianzen zu lernen. Somit kann beispielsweise ein QREN trainiert werden, welches invariant gegenüber der genauen Motorstellung ist, aber alle Zustände kodiert, in welchen der Roboter aufrecht steht.

Eine Möglichkeit, um eine Quadrik zu charakterisieren und zu visualisieren, ist, sie mittels einer Hauptkomponententransformationen in eine Normalform zu bringen und so die gemischten Monome, also z. B. $x_i x_j, i \neq j$, zu eliminieren. Eine andere Möglichkeit ist einen Gradientenabstieg zu verwenden, um sukzessive die Punkte der Quadrik zu bestimmen. Die Idee ist, mit einer beliebigen Nullstelle zu starten, dann eine Verschiebung des Punktes um einen kleinen zufälligen oder gerichteten Wert auszuführen und schließlich dem Gradienten ∇y_j zu folgen, bis wieder ein Punkt auf der Quadrik erreicht ist. Ein Vorteil dieser Methode ist, dass sie auch zur Online-Exploration einer Quadrik verwendet werden kann.

Es müssen bei der Analyse einer SFA-Komponente drei Dinge bedacht werden: Zunächst können nur statische Kontexte sinnvoll ausgewertet werden, d. h. solche, in denen eine SFA-Komponente einen quasi konstanten Wert für einen bestimmten Stimulus ausgibt; in Kapitel 4 beispielweise kodieren die langsamsten Komponenten verschiedene Posen des Roboters, und ein Wert $y_j \approx \mu_j$ kodiert die Pose *Stehen*. Mit Hilfe der Quadriken kann darauf rückgeschlossen werden, welche sensorischen Konfigurationen die SFA dieser Pose zuordnet.

Als zweites ist zu beachten, dass für $I > 3$ keine Visualisierung der Quadrik mehr möglich ist. Ist das Eingabesignal hochdimensional, kann die Analyse einer SFA-Komponente mittels Quadrik durchgeführt werden, indem drei der Eingänge ausgewählt und variabel gehalten werden,

während die anderen Eingänge auf sinnvolle fixe Werte gesetzt werden. Um bei dem Beispiel der Posenerkennung zu bleiben, können beispielsweise drei Sensoren eines Arms ausgewählt werden; die übrigen Sensoren werden auf den jeweiligen Mittelwert festgesetzt, den der Sensor während der stehenden Pose hat. Auf diese Weise können sukzessive die Auswirkungen sensorischer Veränderungen auf die SFA-Komponenten betrachtet werden.

Zuletzt ist zu bemerken, dass auch die Betrachtung von Quadriken streng genommen nur für die quadratische SFA ohne Wiederholung möglich ist. Allerdings kann auch für eine SFA mit Wiederholung eine ähnliche Analyse durchgeführt werden, da sich auch eine solche Komponente als ein Polynom, wenn auch höheren Grades, darstellen lässt. So lässt sich beispielsweise eine SFA-Komponente aus einer zweiten quadratischen SFA-Wiederholung als ein Polynom vierten Grades darstellen. In diesem Fall handelt es sich aber bei den geometrischen Figuren natürlich nicht mehr um Quadriken, sondern um Hyperflächen höherer Ordnung. Außerdem ist ab der dritten Wiederholung, wenn es sich nicht mehr um Polynome höchstens vom Grad vier handelt, die analytische Nullstellenbestimmung nicht mehr anwendbar; stattdessen kann dann beispielsweise auf die vorgeschlagene Gradientenabstieg-Methode zurückgegriffen werden.

2.5 Erweiterungen und Verwandtschaft zu anderen Verfahren

Nachdem der SFA-Algorithmus auf den letzten Seiten eingehend vorgestellt wurde, soll nun ein kurzer Überblick über die Verwandtschaft der SFA zu anderen Verfahren sowie daraus resultierende Erweiterungen gegeben werden. Einige der vorgestellten Verfahren und Erweiterungsmöglichkeiten werden im weiteren Verlauf der Arbeit verwendet und näher ausgeführt.

2.5.1 SFA und PCA

Die *Principal Component Analysis (PCA)* (deutsch: *Hauptkomponentenanalyse*), auch *Karhunen-Loève-Transformation* genannt, ist ein Verfahren aus der Statistik, welches in der Regel zur Strukturierung und Vereinfachung von umfangreichen Datensätzen verwendet wird. In der Statistik betrachtet man häufig Stichproben, welche durch Zufallsvariablen mit mehreren Realisierungen modelliert werden. Beispielsweise kann eine Zufallsvariable ein bestimmtes Merkmal, wie *Körpergröße* modellieren, wobei die Realisierungen die konkreten Werte verschiedener Versuchspersonen sind; ebenso können die verschiedenen Zufallsvariablen auch Sensordaten und ihre Realisierungen Messwerte über die Zeit darstellen. In jedem Fall lassen sich die Datensätze in einer Matrix $\mathbf{X} \in \mathbb{R}^{p \times n}$ zusammenfassen, wobei n die Anzahl der Zufallsvariablen und p die Anzahl ihrer jeweiligen Realisierungen bezeichnet.

Die grundlegende Annahme der PCA ist, dass die Varianz ein Maß für den Informationsgehalt eines Signals darstellt. Daher ist das Ziel der PCA, die Eingabematrix $\mathbf{X}$ so zu rotieren, dass die Varianz entlang der Hauptachsen der rotierten Matrix maximal ist. Zudem sollen nach der Rotation die Spalten der resultierenden Matrix, welche mit $\mathbf{Y} \in \mathbb{R}^{q \times n}$ bezeichnet werden soll, aufsteigend nach ihrer Varianz sortiert sein. Nach der Rotation weist $\mathbf{Y}$ die gleiche Dimensionalität wie $\mathbf{X}$ auf, jedoch kann die PCA auch zur Dimensionsreduktion verwendet werden, indem

$q < p$ gewählt wird, also die Spalten mit geringerer Varianz weggelassen werden.

Die PCA ist ein gängiges Verfahren zum einen zur Extraktion von versteckten Merkmalen aus einem komplexen Datensatz, zum anderen zur Vorverarbeitung von Daten, vor allem um Redundanzen zu eliminieren. Wie zuvor erwähnt, ist die Eliminierung von redundanten Informationen aus einem Datensatz in bestimmten Fällen auch für die Anwendbarkeit der SFA notwendig.

Auf die genaue Implementation der PCA soll an dieser Stelle nicht eingegangen werden, sie kann in der Fachliteratur, wie beispielsweise in [DHS00] nachgelesen werden. Es ist jedoch wichtig zu wissen, dass die PCA auf einer Eigendekomposition der Kovarianzmatrix der Eingabematrix $\mathbf{X}$ beruht. Dies zeigt auch, wie die PCA zur Berechnung der SFA verwendet werden kann: Ausgehend davon, dass $\mathbf{X} = \mathbf{x}(t)$, die Eingabematrix also ein mehrdimensionales zeitdiskretes Signal ist, kann die PCA auf dessen numerische Ableitung $\dot{\mathbf{x}}(t)$ angewandt werden. Jedoch sind für die SFA nicht die Dimensionen mit der höchsten, sondern der *niedrigsten* Varianz interessant; man spricht in diesem Fall auch von einer *Minor Component Analysis (MCA)*. Wie eingangs erwähnt, wurde die SFA ursprünglich so auf Basis der PCA implementiert. Um die Nebenbedingungen der SFA zu erfüllen, muss jedoch vor Anwendung der MCA ein *Whitening* der Eingabematrix durchgeführt werden, welches im folgenden beschrieben wird.

Whitening

Beim *Whitening* oder *Sphering* handelt es sich um eine affine Transformation, welche eine Menge von Signalen dekorreliert und normalisiert. Das bedeutet, dass die Eingabematrix $\mathbf{X} \in \mathbb{R}^{p \times n}$ so transformiert wird, dass ihre Kovarianzmatrix der Identitätsmatrix entspricht. Betrachten wir wieder den Fall $\mathbf{X} = \mathbf{x}(t)$, so bedeutet dies, dass alle Signale $x_i(t), i = 1, \ldots, n$ einen Mittelwert von Null und eine Varianz von Eins haben sowie dekorreliert voneinander sind. Mittels der PCA lässt sich für jede Eingabematrix eine Whiteningmatrix bestimmen, welche die Eingabe in eine solche bereinigte Form überführt. Ein solches bereinigtes Signal wird im folgenden auch als *white* bezeichnet.

Wie eine konkrete Whiteningmatrix berechnet wird, lässt sich ebenfalls in [DHS00] nachlesen.

Das Whitening ermöglicht, verschiedene Datensätze bzgl. bestimmter Gesichtspunkte besser zu vergleichen, da es die statistischen Momente erster und zweiter Ordnung, also Mittelwert und Varianz bzw. Kovarianz eliminiert. Wie zuvor erwähnt kann der η-Wert, welcher die Langsamkeit eines Signals misst, erst dann sinnvoll angewandt werden, wenn die Daten normalisiert sind.

2.5.2 SFA und ICA

Unter dem Namen *Independent Component Analysis (ICA)* (deutsch: *Unabhängigkeitsanalyse*) gruppiert man eine Reihe statistischer Verfahren, welche zur Extraktion statistisch unabhängiger Komponenten aus einem mehrdimensionalen Datensatz verwendet werden [HKO01]. Die ICA ist nahe verwandt mit der *Blind Source Separation (BSS)* (deutsch: Blinde Quellentrennung) und lässt sich am besten anhand dieser Problemstellung erklären: Angenommen, es gibt m gemessene Signale x_j sowie n unbekannte Quellsignale s_i. Beispielsweise könnten die Quellsignale n miteinander kommunizierende Sprecher darstellen, wobei das Gespräch über m Mikrophone auf-

gezeichnet wird. Wenn die Sprecher allerdings gleichzeitig reden, kommen an den Mikrophonen nicht die reinen Signale s_i, sondern Mischsignale an. Die Aufgabenstellung der BSS besteht darin, die n unbekannten Quellsignale s_i aus den m beobachteten Signalen x_j zu rekonstruieren.

Bei der linearen BSS wird dabei angenommen, dass die beobachteten Signale als Linearkombination der Quellsignale darstellbar sind, also dass $\mathbf{x} = \mathbf{A}\mathbf{s} = \sum_i \mathbf{a}_i s_i$, wobei $\mathbf{A}$ *Mixingmatrix* genannt wird und $\mathbf{a}_i$ die i-te Spalte von $\mathbf{A}$ bezeichnet. Das Ziel ist nun eine *Demixingmatrix* $\mathbf{D} \approx \mathbf{A}^{-1}$ zu finden, so dass $\mathbf{y} = \mathbf{D}\mathbf{x} \approx \mathbf{s}$.

Eine mögliche Annahme ist, dass die n unbekannten Quellsignale s_i stochastisch unabhängig voneinander sind, und in diesem Fall kann die ICA für die BSS verwendet werden.

Interessanterweise lässt sich die lineare SFA als eine spezielle Variante der ICA auffassen, wie in [BBW06] gezeigt wurde und im folgenden beschrieben werden soll.

TDSEP und ICA zweiter Stufe

Die ursprüngliche Idee, zusätzlich die zeitliche Struktur des beobachteten Signals in die BSS zu integrieren, stammt aus [MS94] und wurde in [ZM98] zum TDSEP-Algorithmus (TDSEP = **T**emporal **D**ecorrelation source **SEP**aration) weiterentwickelt. BSS-Algorithmen, welche keine zeitliche Struktur einbeziehen und beispielsweise nur versuchen, die Kovarianzmatrix des beobachteten Signales zu diagonalisieren (die Annahme dabei ist, dass die Dekorrelation der beobachteten Signale die Quellsignale rekonstruiert), schlagen in bestimmten einfach zu konstruierenden Beispielen fehl: Nimmt man beispielsweise als Quellen eine normalverteilte Zufallsvariable sowie eine zweite zeitlich verzögerte Kopie dieser Zufallsvariable und mischt diese auf beliebige nichttriviale Weise, so wird ein BSS-Algorithmus ohne Einbeziehung der zeitlichen Struktur die beiden Quellen nicht rekonstruieren können. Die Idee von TDSEP ist daher, simultan mehrere zeitverzögerte Kovarianzmatrizen zu diagonalisieren.

TDSEP wird in [BBW06] als *ICA zweiter Stufe* bezeichnet, da sie lediglich statistische Methoden zweiter Stufe, nämlich die Dekorrelation benutzt, welches ein notwendiges, aber kein hinreichendes Kriterium für stochastische Unabhängigkeit ist. Die zeitverzögerte Kovarianzmatrix für ein zeitdiskretes mittelwertzentriertes und orthonormalisiertes Signal $\mathbf{x}(t) =: \mathbf{x}_t$ sei folgendermaßen definiert:

$$\tilde{\mathbf{C}}^{(\mathbf{x})}(\Delta t) = \langle \mathbf{x}_t \mathbf{x}_{t-\Delta t}^T \rangle \tag{2.34}$$

In der Regel wird die symmetrische Version der zeitverzögerten Kovarianzmatrix verwendet:

$$\mathbf{C}^{(\mathbf{x})}(\Delta t) = \frac{1}{2} \left(\langle \mathbf{x}_t \mathbf{x}_{t-\Delta t}^T \rangle + \langle \mathbf{x}_{t-\Delta t} \mathbf{x}_t^T \rangle \right) \tag{2.35}$$

Das Ziel von TDSEP ist nun, die zeitverzögerten Kovarianzmatrizen $\mathbf{C}^{(\mathbf{y})}(\Delta t)$ der Zielfunktion $\mathbf{y} = \mathbf{w}^T \mathbf{x}$ für mehrere Δt zu diagonalisieren.

Äquivalenz von TDSEP und linearer SFA

In diesem Abschnitt soll [BBW06] folgend die Äquivalenz von linearer SFA und TDSEP mit einer Stufe Zeitverzögerung, d. h. dass die SFA $\mathbf{C}^{(\mathbf{y})}(\Delta t)$ für $\Delta t = 1$ diagonalisiert, gezeigt

werden. Um dies zu zeigen, nehmen wir an, dass die Ableitung wie in Abschnitt 2.3.1 erwähnt durch endliche Differenzen angenähert wird. Dann lässt sich die Kovarianzmatrix von $\dot{\mathbf{x}}_t$ wie folgt umformulieren:

$$\begin{aligned}
\mathbf{A} &:= \langle \dot{\mathbf{x}}_t \dot{\mathbf{x}}_t^T \rangle = \langle (\mathbf{x}_t - \mathbf{x}_{t-1})(\mathbf{x}_t - \mathbf{x}_{t-1})^T \rangle && (2.36)\\
&= \underbrace{\langle \mathbf{x}_t \mathbf{x}_t^T \rangle}_{=\mathbf{I}} + \underbrace{\langle \mathbf{x}_{t-1} \mathbf{x}_{t-1}^T \rangle}_{=\langle \mathbf{x}_t \mathbf{x}_t^T \rangle = \mathbf{I}} - \langle \mathbf{x}_t \mathbf{x}_{t-1}^T \rangle - \langle \mathbf{x}_{t-1} \mathbf{x}_t^T \rangle && (2.37)\\
&\underset{(2.35)}{=} 2\mathbf{I} - 2\mathbf{C}^{(\mathbf{x})}(1) && (2.38)
\end{aligned}$$

In Schritt (2.37) und im folgenden wird ohne Beschränkung der Allgemeinheit angenommen, dass das Eingabesignal white ist. Für das linearisierte SFA-Optimierungsproblem aus Gleichung 2.5 ergibt sich nun:

$$\begin{aligned}
\Delta(y_j) &= \langle \dot{y_j}^2 \rangle = \mathbf{w}_j^T \mathbf{A} \mathbf{w}_j \underset{(2.38)}{=} 2\underbrace{\mathbf{w}_j^T \mathbf{I} \mathbf{w}_j}_{=1} - 2\mathbf{w}_j^T \mathbf{C}^{(\mathbf{x})}(1)\mathbf{w}_j && (2.39)\\
&= 2 - 2\mathbf{w}_j^T \mathbf{C}^{(\mathbf{x})}(1)\mathbf{w}_j && (2.40)
\end{aligned}$$

In (2.39) wurde die geforderte Orthonormalität der Gewichtsvektoren (siehe Abschnitt 2.2, Bedingung 3) ausgenutzt. Das Optimierungsproblem lässt sich nun so von einem Minimierungs- in ein Maximierungsproblem umformulieren, so dass die konstanten Terme, welche während der Optimierung keine Rolle spielen, wegfallen:

$$\tilde{\Delta}(y_j) := 1 - \frac{1}{2}\Delta(y_j) = \mathbf{w}_j^T \mathbf{C}^{(\mathbf{x})}(1)\mathbf{w}_j \quad (2.41)$$

Somit wird nach den orthonormalen Gewichtsvektoren gesucht, welche die um einen Schritt zeitverzögerte symmetrische Kovarianzmatrix $\mathbf{C}^{(\mathbf{x})}(1)$ maximieren. Dies ist analog zur Herleitung der ursprünglichen SFA-Formulierung (siehe Abschnitt 2.2) offensichtlich der Eigenvektor dieser Kovarianzmatrix mit dem größten zugehörigen Eigenwert; da die Multiplikation der so erhaltenen Eigenvektormatrix mit dem Eingabesignal zudem die Eingabe dekorreliert, was der Diagonalisierung der um einen Zeitschritt verzögerten Kovarianzmatrix der Eingabematrix entspricht, ist die Äquivalenz der linearen SFA mit numerischer Ableitung zu TDSEP mit $\Delta t = 1$ gezeigt.

ISFA

Ausgehend von der im letzten Abschnitt gezeigten Äquivalenz von TDSEP bzw. ICA zweiter Stufe und linearer SFA wurde die *Independent SFA* entwickelt [Bla05]. Das Verfahren ermöglicht es, mithilfe der SFA nichtlineare BSS durchzuführen. Dabei wird die SFA als zusätzliches Kriterium zur Auswahl der Quellen ausgenutzt, da bei nichtlinearer BSS die Quellen nicht mehr eindeutig nur durch das Kriterium der stochastischen Unabhängigkeit gefunden werden können: Angenommen, das Mixing erfolgt durch eine quadratische Funktion, so sind die Quellen s_1 und s_2 ebenso unabhängig wie s_1^2 und s_2^2 oder s_1^2 und s_2.

Die ISFA beruht daher auf einer Kombination einer auf TDSEP basierenden ICA- sowie der SFA-Zielfunktion. Betrachten wir zunächst die ICA-Zielfunktion; der Einfachheit halber sei in

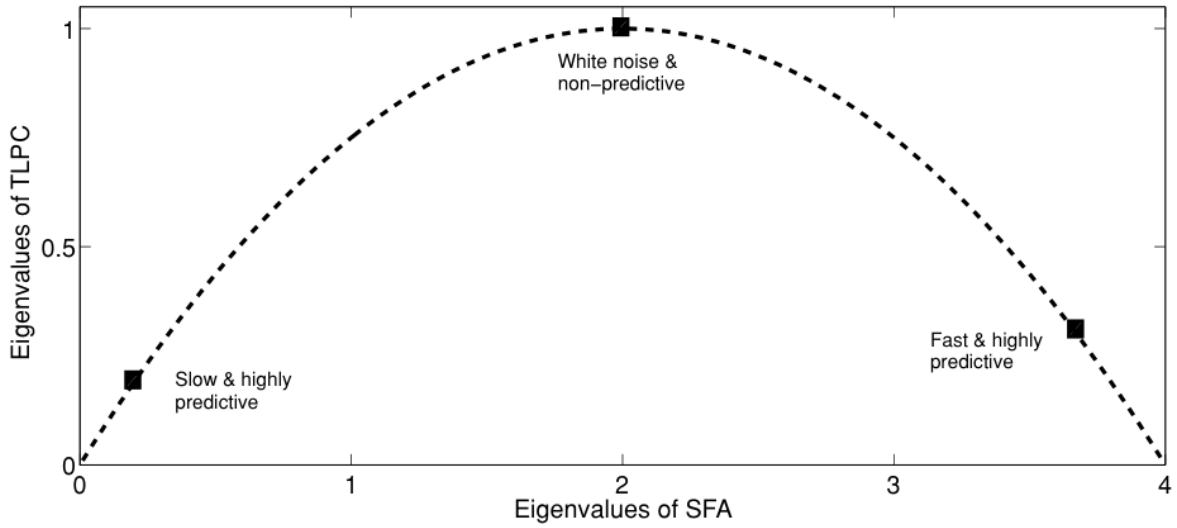


Abbildung 2.3: Zusammenhang zwischen den Eigenwerten der Komponenten der SFA und des TLPC. TLPC präferiert Vorhersagbarkeit gegenüber Langsamkeit, vermeidet aber Komponenten mit hohem Zufallsgrad. (Aus [CS08])

der folgenden Darstellung wie in [MS94] statt mehreren nur ein einziges festes Δt als Verzögerung gewählt, so dass sich das Problem auf die Diagonalisierung der zeitverzögerten Kovarianzmatrix $\mathbf{C}^{(\mathbf{y})}(\Delta t)$ beschränkt[4]. Wegen der Annahme, dass $\mathbf{y}$ white ist, ist die unverzögerte Autokovarianzmatrix $\mathbf{C}^{(\mathbf{y})}(0)$ bereits diagonalisiert und taucht somit nicht in der Zielfunktion auf:

$$\Psi_{\mathbf{ICA}} = \sum_{i,j=1 i\neq j}^{N} (\mathbf{w}_j^T C_{ij}^{(\mathbf{y})}(\Delta t)\mathbf{w}_j)^2. \tag{2.42}$$

Dabei soll $\Psi_{\mathbf{ICA}}$ hinsichtlich der Gewichte $\mathbf{w}_j$ minimiert werden. $C_{ij}^{(\mathbf{y})}$ bezeichnet wie gewöhnlich das Element der i-ten Zeile und der j-Spalte der Matrix $\mathbf{C}^{(\mathbf{y})}$. Für die SFA lässt sich unter Verwendung von 2.41 die zu maximierende Zielfunktion $\Psi_{\mathbf{SFA}} = \sum_{i=1}^{N} (C_{ii}^{(\mathbf{y})}(\Delta t))^2$ formulieren, so dass sich als Zielfunktion für die ISFA ergibt:

$$\Psi_{\mathbf{ISFA}} = b_{\mathbf{ICA}}\Psi_{\mathbf{ICA}} - b_{\mathbf{SFA}}\Psi_{\mathbf{SFA}}. \tag{2.43}$$

$b_{\mathbf{ICA}}$ und $b_{\mathbf{SFA}}$ dienen als Parameter zur Gewichtung der jeweiligen Terme. Der Optimierungsalgorithmus hinsichtlich dieser Zielfunktion ist etwas komplexer als der SFA-Algorithmus und kann in [Bla05] nachgelesen werden.

2.5.3 Temporally Local Predictive Coding

Eine Verwandtschaft der SFA zu einem informationstheoretischen Verfahren konnte in [CS08] gezeigt werden. Sie basiert auf der in [TPB99] entwickelten *Information-Bottleneck-Methode*, welche eine Methode zur Extraktion relevanter Eigenschaften aus Daten ist. Gegeben zwei Zufallsvariablen X und R sind die Eigenschaften von X gesucht, welche die beobachteten Zustände von R

[4]Die Erweiterung der Zielfunktion auf mehrere Zeitschritte Δt kann in [BBW06] nachgelesen werden.

am besten erklären. Dazu wird die Variable X in eine komprimierte Repräsentation Y überführt, während Y so viel Information wie möglich über R behalten soll. Sei der mittlere Informationsgehalt zweier Zufallsvariablen als $I(X;Y) = E_{X,Y}\{\log_2(\frac{p(X,Y)}{p(X)p(Y)})\}$ definiert, welche die Stärke des statistischen Zusammenhangs von X und Y angibt[5]. Das Information-Bottleneck-Problem lässt sich dann formalisieren als:

$$\min \mathcal{L} := I(X;Y) - \beta I(Y;R). \tag{2.44}$$

Der erste Term minimiert dabei die Komplexität der Kompression von X in Y, während der zweite Term die Präzision der Repräsentation von R durch Y erhöht.

Ausgehend vom Information-Bottleneck-Problem lässt sich eine Formulierung des so genannten *Predictive-Coding-Prinzips* herleiten; im Kontext lernender Agenten bedeutet Predictive Coding, dass ein Agent aus seinen sensorischen Daten Informationen extrahiert, welche gut die Zukunft vorhersagen. Aus informationstheoretischer Sicht bedeutet das, dass die Eingabedaten in einer internen Zustandsvariable gespeichert werden, so dass die Vorhersage der Zukunft aus dieser Zustandsvariable bestmöglich ist. Diese Zustandsvariable kann also als komprimierte Darstellung der vergangenen Erfahrung des Agenten verstanden werden. Somit lässt sich Gleichung (2.44) in folgende Gleichung überführen

$$\min \mathcal{L}^{\mathrm{TLPC}} := I(\text{Vergangenheit};\text{Zustand}) - \beta I(\text{Zustand};\text{Zukunft}), \tag{2.45}$$

die als Zielfunktion für das *Temporally Local Predictive Coding (TLPC)* bezeichnet wird. Wieder forciert der erste Term eine sparsame Kodierung des internen Zustands, welcher die Vergangenheit repräsentiert, während der zweite Term die Vorhersagbarkeit der Zukunft aus dem Zustand maximiert. Im folgenden soll lediglich der Fall betrachtet werden, dass der aktuelle Zustand Y nur vom letzten Eingabesignal X abhängt und durch eine lineare Transformation berechnet werden kann. Während in 2.5.2 gezeigt wurde, dass die SFA die Matrix $\Sigma^{\mathrm{SFA}} = 2\mathbf{I} - 2\mathbf{C}^{(\dot{\mathbf{x}})}(1)$ (Gleichung (2.35)) diagonalisiert, lässt sich hier zeigen, dass durch $\mathcal{L}^{\mathrm{TLPC}}$ die Matrix

$$\Sigma^{\mathrm{TLPC}} = \mathbf{I} - (\mathbf{C}^{(\dot{\mathbf{x}})}(1))^2 \tag{2.46}$$

diagonalisiert wird. Durch geeignetes Umstellen von (2.35) und Einsetzen in (2.46) ergibt sich für die jeweils berechneten Eigenwerte folgende Beziehung:

$$\lambda_i^{\mathrm{TLPC}} = \lambda_i^{\mathrm{SFA}} - \frac{1}{4}(\lambda_i^{\mathrm{SFA}})^2. \tag{2.47}$$

Abbildung 2.3 zeigt die möglichen Eigenwerte von SFA und TLPC gegeneinander aufgetragen. Die Grafik macht deutlich, welche Beziehung zwischen langsamen und prädiktiven Komponenten besteht: TLPC versucht möglichst zufällige Fluktuationen zu vermeiden und macht dabei keinen Unterschied zwischen langsamen und schnellen Signalen, solange sie hohe Voraussagbarkeit haben. Allerdings extrahieren TLPC und SFA die gleichen Komponenten in der gleichen Reihenfolge; doch während die SFA allen Komponenten wegen der Einschränkung der Varianz von Eins (Gleichung (2.3)) die gleiche Amplitude zuweist, gewichtet TLPC Komponenten, die eine höherere Vorhersagbarkeit aufweisen, durch eine höhere Amplitude.

[5] $E_X\{X\}$ gibt dabei den Erwartungswert von X und $p(X) := p(X = x)$ die Wahrscheinlichkeit an, dass die Zufallsvariable X den Wert x annimmt.

2.5.4 Probabilistische SFA

In [TS07] wird eine probabilistische Interpretation der SFA vorgestellt. Genauer gesagt wird gezeigt, dass Maximum-Likelihood-Learning in einem linearen Gaußschen Zustandsraum mit einer Markov-Bedingung äquivalent zur SFA ist.

Als Vorteil der probabilistischen Version der SFA wird u. a. herausgestellt, dass die Nebenbedingungen, die in die SFA integriert sind, sich auf eine weniger strikte Weise formulieren lassen. So sollen ja die Varianz von Eins (Gleichung (2.3)) das Verschwinden von Signalen mit geringer Amplitude verringern, während die Dekorrelationsbedingung (Gleichung (2.4)) die Reproduktion der Signale verhindert. Im vorgestellten probabilistischen Framework werden diese Bedingungen implizit dadurch eingehalten, dass die gewählte A-priori-Wahrscheinlichkeit eine normalisierende Wirkung hat, und sich die Dekorrelation durch die Wahl einer faktorisierten Gaußschen Verteilung ergibt. Kurz zusammengefasst wird folgendes Modell vorgeschlagen:

$$p(\mathbf{y}(t)|\mathbf{y}(t-1), \mathbf{\Lambda}, \mathbf{\Sigma}) = \mathcal{N}(\mathbf{\Lambda}\mathbf{y}(t-1), \mathbf{\Sigma}) \tag{2.48}$$

$$p(\mathbf{y}(1)|\mathbf{\Sigma}^*) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}^*). \tag{2.49}$$

Dabei sind $\mathbf{\Sigma}^*$, $\mathbf{\Sigma}$ und $\mathbf{\Lambda}$ Diagonalmatrizen, welche die initiale Varianz, die Prozessvarianz und die Stärke der Korrelation der latenten Variable $\mathbf{y}$ an zwei aufeinanderfolgenden Zeitpunkten bezeichnen. Intuitiv wird der gesamte Prozess langsamer, je stärker $\mathbf{y}(t)$ und $\mathbf{y}(t-1)$ korreliert sind, also für $\lambda_j \to 1$. Die Autoren beschreiben im weiteren Verlauf ihrer Arbeit, wie dieses probabilistische Modell gelernt werden kann, und dass ähnliche Ergebnisse wie mit der SFA erzielt werden.

2.5.5 Contextual SFA

Ein weiterer interessanter Ansatz, um die SFA auf ein qualitativ angereichertes Eingangssignal anzuwenden, ist die *Contextual SFA (cSFA)* [Dei10]. Dieser Methode liegt die Beobachtung zugrunde, dass oft kontextuelle Information zur Verfügung steht, welche dazu genutzt werden könnte, um die grundsätzlich unüberwacht agierende SFA bzgl. ihrer Ausgabe in eine Richtung zu lenken. Damit dies möglichst ohne starke Modifikation des SFA-Algorithmus erfolgt, wird nicht der SFA-Algorithmus selbst verändert, sondern die SFA auf ein kontextuell bewertetes Signal angewandt. Beispielsweise können die von der SFA verarbeiteten Eingabedaten ein Sensorsignal eines Roboters sein, welches durch ein Motorsignal bewertet ist, damit die SFA nur bei motorischen Aktionen bzw. bestimmten Aktionen lernt. In [Dei10] werden als Varianten für eine cSFA einerseits *affirmative* und andererseits *suppressive* Bewertungsfunktionen vorgeschlagen; dabei heißt affirmativ, dass bei hohen Werte im Kontextsignal stärker gelernt wird, während bei der suppressiven cSFA höhere Werte im Kontextsignal das Lernen hemmen. Eine einfache affirmative Variante ist die Subtraktion des Kontextsignals a vom eigentlichen Eingabesignal x:

$$\tilde{\mathbf{x}} = \mathbf{x} - a\mathbf{1} =: \mathbf{x} - \mathbf{a}. \tag{2.50}$$

Sei beispielsweise das Kontextsignal eine eindimensionale zeitabhängige Funktion $a(t)$, welches mit dem Einheitsvektor der Dimension von $\mathbf{x}(t)$ multipliziert wird. Betrachten wir nun, wel-

chen Δ-Wert (2.1) die auf das kontextbewertete Signal $\tilde{\mathbf{x}}$ angewandte SFA für die langsamsten Komponenten berechnet:

$$\begin{aligned}\Delta(y_j) &\underset{(2.9)}{=} \mathbf{w}_j^T \langle \dot{\tilde{\mathbf{x}}} \dot{\tilde{\mathbf{x}}}^T \rangle \mathbf{w}_j^T && (2.51)\\ &\underset{(2.50)}{=} \mathbf{w}_j^T \langle (\dot{\mathbf{x}} - \dot{\mathbf{a}})(\dot{\mathbf{x}} - \dot{\mathbf{a}})^T \rangle \mathbf{w}_j && (2.52)\\ &= \mathbf{w}_j^T (\langle \dot{\mathbf{x}} \dot{\mathbf{x}}^T \rangle - \langle \dot{\mathbf{x}} \dot{\mathbf{a}}^T \rangle - \langle \dot{\mathbf{a}} \dot{\mathbf{x}}^T \rangle + \langle \dot{\mathbf{a}} \dot{\mathbf{a}}^T \rangle) \mathbf{w}_j && (2.53)\end{aligned}$$

Wie zu sehen ist, optimiert die SFA in diesem Fall also implizit hinsichtlich dreier zusätzlicher Terme, wobei der Term $\langle \dot{\mathbf{a}} \dot{\mathbf{a}}^T \rangle = \dot{a}^2 \mathbf{I}$ konstant ist und ignoriert werden kann. Durch die Terme $\langle \dot{\mathbf{x}} \dot{\mathbf{a}}^T \rangle$ und $\langle \dot{\mathbf{a}} \dot{\mathbf{x}}^T \rangle$ wird eine hohe Korrelation von $\dot{\mathbf{x}}$ und $\dot{\mathbf{a}}$ belohnt. Da die Korrelationsterme negativ in die Zielfunktion eingehen und diese minimiert wird, wird das Ausgabesignal y_j daher sowohl hinsichtlich Langsamkeit als auch hoher Korrelation zum Kontextsignal optimiert.

Statt einer Subtraktion des Kontextsignals $a(t)$ vom Eingabesignal $\mathbf{x}(t)$ sind natürlich auch multiplikative Gewichtungen möglich. Setzen wir $\tilde{\mathbf{x}} := f\mathbf{x}$ für eine beliebige Funktion $f(t) := \hat{f}(\dot{a}(t))$, so gilt:

$$\Delta(y_j) = \mathbf{w}_j^T \langle f^2 \dot{\mathbf{x}} \dot{\mathbf{x}}^T \rangle \mathbf{w}_j \quad (2.54)$$

Sei beispielsweise $\hat{f}(x) := \frac{1}{\sqrt{1+(\mu x)^2}}$, so erhalten wir eine einfache suppressive cSFA,

$$\Delta(y_j) = \mathbf{w}_j^T \left\langle \frac{1}{1+(\mu \dot{a})^2} \dot{\mathbf{x}} \dot{\mathbf{x}}^T \right\rangle \mathbf{w}_j, \quad (2.55)$$

da eine hohe Aktivität des Kontextsignals über die Zeit das Eingabesignal unterdrückt. Der Parameter μ kann dabei genutzt werden, um die Wirkung des Kontextsignals gegenüber der Langsamkeitsbedingung zu gewichten.

2.5.6 SFA mit zeitlicher Einbettung

Eine Möglichkeit, den Informationsgehalt des Eingabesignals zu erhöhen, ist der SFA in jedem Zeitschritt verzögerte Kopien des Eingabesignals zur Verfügung zu stellen. Diese Technik wird als *zeitliche Einbettung* (englisch: *time embedding*) oder auch *Takens-Einbettung* bezeichnet. Letztere Bezeichnung wurde zu Ehren F. Takens' eingeführt, der in [Tak81] beweisen konnte, unter welchen Bedingungen bestimmte Attraktoren von dynamischen Systemen durch die zeitliche Einbettung rekonstruiert werden können. Bei der zeitlichen Einbettung gibt es zwei Parameter, zum einen die Anzahl der Zeitschritte m, auch *tap delay* genannt, welche verzögert als zusätzliche Eingabesignale verwendet werden, zum anderen der Abstand zwischen den Zeitpunkten τ, welcher als *gap* bezeichnet wird. Als zeitlich eingebetteter Eingabevektor ergibt sich dann für ein eindimensionales Ursprungssignal $x(t)$:

$$\tilde{\mathbf{x}} := [x(t - m\tau), x(t - (m-1)\tau), \ldots, x(t - 2\tau), x(t - \tau), x(t)]^T \quad (2.56)$$

Die Definition lässt sich einfach auf einen mehrdimensionalen Eingabevektor erweitern.

Betrachtet man die Eingabe $x(t)$ als Ausgabe eines dynamischen Systems, erhält die SFA durch die zeitliche Einbettung mehr Informationen über den Phasenraum dieses Systems. Wir erinnern uns, dass die von der SFA berechneten langsamen Komponenten Linearkombinationen aus den Komponenten des (expandierten) Eingabesignals sind. Somit könnte die SFA eine Komponente der Form $y_j = \ldots + \alpha x_i(t) + \beta x_i(t - \tilde{\tau}) + \ldots$ für $\tilde{\tau} \in \{\tau, 2\tau, \ldots, m\tau\}$ berechnen. Angenommen $\alpha = -\beta$, so erhalten wir $\alpha(x_i(t) - x_i(t - \tilde{\tau}))$, was offensichtlich eine numerische Variante der Ableitung $\dot{x}_i(t)$ (gewichtet durch α) darstellt. Somit wird klar, dass der SFA durch die zeitliche Einbettung die Möglichkeit gegeben wird, die Ableitung des Eingangssignals zur Berechnung der langsamsten Komponente zu benutzen.

Im Abschnitt 2.6.3 wird gezeigt, wie unter Verwendung der zeitlichen Einbettung Eigenschaften pseudochaotischer dynamischer System extrahiert werden können.

Volterra-Expansion

Eine interessante Betrachtungsweise der SFA mit zeitlicher Einbettung ergibt sich aus Sicht der digitalen Filtertechnik. In Abschnitt 2.4.3 wurde die Quadratische-Form-Analyse vorgestellt, welche sich zunutze macht, dass jede SFA-Komponente einer quadratischen SFA in einen konstanten, linearen und quadratischen Teil aufgetrennt werden kann. Verallgemeinert man diese Form zu einem unendlichen Polynom mit unbegrenzter zeitlicher Einbettung, so erhält man für ein eindimensionales Eingangssignal $x(t)$ die so genannte *Volterra-Expansion* [Mat91]:

$$\begin{aligned} y(t) = {} & h_0 + \\ & \sum_{m_1=0}^{\infty} h_1(m_1) x(t - m_1) + \\ & \sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} h_2(m_1, m_2) x(t - m_1) x(t - m_2) + \ldots + \\ & \sum_{m_1=0}^{\infty} \ldots \sum_{m_N=0}^{\infty} h_N(m_1, \ldots, m_N) x(t - m_1) \ldots x(t - m_N) + \ldots, \end{aligned} \tag{2.57}$$

wobei $h_N(m_1, m_2, \ldots, m_N)$ als *Kernel* N-ten Grades des durch diese Reihe gegebenen *Volterra-Filters* bezeichnet wird. Die Kernel entsprechen den Filterkoeffizienten, das heißt Kernel h_0 entspricht dem konstanten Part, h_1 dem linearen, h_2 dem quadratischen etc.

In der Praxis kann die Volterra-Reihe aufgrund ihrer Infinitheit nur angenähert werden. Dazu entwickelt man diese nur bis zu einem Polynom bestimmter Ordnung sowie mit begrenzter zeitlicher Einbettung. Häufige Anwendung findet daher vor allem das Volterra-Filter zweiter Ordnung mit finitem Kernel [LLC92], welches gerade der quadratischen SFA mit zeitlicher Einbettung entspricht. $x(t)$ kann dabei auch ein beliebig dimensionales Eingabesignal sein, so dass sich beim quadratischen Teil der Volterra-Expansion nicht nur Mischterme der gleichen Eingabekomponente $x_i(t - m_1)x_i(t - m_2)$, sondern auch alle Mischterme paarweise verschiedener Eingabekomponenten $x_i(t - m_1)x_j(t - m_2), i \neq j$ ergeben.

In Abschnitt 3.3 folgt eine Einführung in die Grundlagen von digitalen Filtern sowie Volterra-Filtern zweiter Ordnung. Diese bilden die Grundlage, um in Kapitel 5 ein Steuersignal für ein

zweibeiniges Laufmuster unter Verwendung der SFA zu generieren.

2.5.7 Alternative Expansionen und Kernel-SFA

Werden bei der cSFA die Eingangssignale variiert, so dass die von der SFA extrahierten langsamen Komponenten bestimmte gewünschte Eigenschaften aufweisen, lässt sich alternativ auch die Art der nichtlinearen Expansion beeinflussen. Der Ansatz ist ebenso direkt und einfach wie die Veränderung des Eingangssignales, da nur der Expansionsschritt ausgetauscht werden muss; interessant ist zu untersuchen, welche Expansionen in welchen Fällen die geeigneteren sind. Der einfachste Weg ist Polynome höherer Ordnung zu verwenden. Allerdings ist nicht klar, ob diese einen Vorteil gegenüber der Hintereinanderschaltung mehrerer SFA-Einheiten bringen; zudem sind sie noch stärker vom Fluch der Dimensionalität betroffen.

Eines der Probleme mit der polynomiellen Expansion ist, dass Polynome prinzipiell unbeschränkt in ihrem Wertebereich sind. Besonders bei sehr langsamen Komponenten mit einzelnen sehr starken Ausschlägen werden diese Ausschläge durch die Einheits-Varianz-Beschränkung u. U. unverhältnismäßig stark skaliert. Daher wird in praktischen Anwendungen bei der wiederholten SFA oft ein Clipping durchgeführt, um die Signale vor der Verarbeitung durch die nächste SFA-Einheit auf einen dem Eingabebereich entsprechenden Wertebereich zu beschränken (z. B. in [FSW07]). Eine andere Möglichkeit die Unbeschränktheit zu umgehen ist beschränkte nichtlineare Funktionen wie z. B. den tangens hyperbolicus als Expansionsfunktion zu benutzen. Eine nichtlineare Expansion unter Verwendung des tangens hyperbolicus wird in Abschnitt 3.1 vorgestellt.

Kernel-SFA

Eine alternative Sichtweise zur SFA auf expandierten Daten ist die in [Nic06] vorgestellte *Kernel-SFA*, welche sich aus der Formulierung der *Kernel-PCA* [SSM98] herleiten lässt. Sie beruht auf der Anwendung des im folgenden erläuterten *Kernel-Tricks*, welcher erstmals in [ABR64] vorgestellt wurde.

Eine Abbildung $k : X \times X \to \mathbb{R}$ heißt Kernel, wenn sie das Skalarprodukt in einem *Merkmalsraum* (engl.: *feature space*) F definiert, so dass $k(x, y) = \phi(x)^T \phi(y)$, wobei $\phi : X \to F$ die so genannte *Merkmalsabbildung* (engl.: *feature mapping*) ist. In der Regel ist der Merkmalsraum F nicht explizit bekannt und muss nicht berechnet werden, sondern ist nur implizit durch die Kernelfunktion k gegeben. Durch den Kernel-Trick kann somit jedes lineare Lernverfahren, welches ausschließlich auf dem Skalarprodukt basiert, in ein nichtlineares Verfahren transformiert werden, ohne den u. U. sehr großen Merkmalsraum explizit berechnen zu müssen.

Da bei der SFA das Skalarprodukt zur Multiplikation der Gewichtsvektoren $\mathbf{w}_j$ mit den Eingabedaten zum Einsatz kommt, müssen für die Kernel-SFA auch die $\mathbf{w}_j$ im Merkmalsraum F liegen. Dazu werden die Gewichtsvektoren als eine Linearkombination von *Stützvektoren* (engl.: *support vectors*) $\mathbf{z}_i$, welche in den Merkmalsraum abgebildet werden, repräsentiert:

$$\mathbf{w}_j = \sum_{i=1}^{M} u_{ij} \phi(\mathbf{z}_i). \tag{2.58}$$

M entspricht dabei der Anzahl der verwendeten Stützvektoren und u_{ij} sind die gesuchten Koeffizienten. Es lässt sich zeigen, dass, gegeben eine Menge geeigneter Stützvektoren, die Kernel-SFA analog zur SFA mit Expansion gelöst werden kann. Das Problem verschiebt sich dahin, die richtigen Stützvektoren für die geeignet gewählten Kernelfunktionen zu finden. In [Nic06] werden dabei verschiedene populäre Kernelfunktionen bzgl. ihrer Lernfähigkeit auf visuellen Eingabedaten verglichen. Es wird gezeigt, dass die durch Kernel-SFA gelernten Komponenten dabei in der Regel langsamer sind als durch die SFA mit polynomieller Expansion gefundene; auf der anderen Seite sind sie schwieriger zu interpretieren und analytisch vorherzusagen.

2.6 Anwendungen der SFA

Um einen Überblick über die Anwendbarkeit der SFA zu bekommen, sollen neben den zuvor vorgestellten Verwandtschaften und Erweiterungsmöglichkeiten im nächsten Abschnitt einige bisher realisierte Anwendungen mit der SFA vorgestellt werden. Da die SFA biologisch motiviert ist, werden zunächst Anwendungsmöglichkeiten in der Modellierung neuronaler Strukturen des visuellen Cortex vorgestellt, welche auf visuellen Daten operieren. Da die SFA jedoch sehr flexibel und überhaupt nicht domäneneingeschränkt ist, konnten aber auch schon verschiedene andere Anwendungen der SFA realisiert werden.

2.6.1 Complex cells

Die ersten Arbeiten ([WS02], [BW02]) zur SFA beschäftigen sich v. a. mit der Modellierung von Eigenschaften bestimmter Neuronen im visuellen Cortex, den so genannten *complex cells*. Dieser Zelltyp kommt im primären visuellen Cortex (V1), sekundären visuellen Cortex (V2) sowie im Brodmann-Areal (V3) vor und zeichnet sich dadurch aus, dass er ebenso wie die so genannten *simple cells* auf orientierte Kanten und Raster reagiert, allerdings invariant bzgl. der räumlichen Lage der Orientierungen in seinem rezeptiven Feld[6] ist.

Um complex cells als Ausgabekomponenten der SFA zu erhalten, wurden als Eingangsdaten Naturbilder (engl.: *natural images*) bzw. Ausschnitte aus diesen verwendet. In [BW02] beispielsweise wurden die Eingabesequenzen für die SFA generiert, indem aus 36 Naturbildern eines zufällig ausgewählt wurde, und ein 16×16 Pixelfenster durch Translation, Rotation und Zoom über das Bild bewegt wurde. Um die Dimensionalität zu verringern, wurde auf die Eingabedaten zuvor eine PCA angewandt, welche die 50 Komponenten mit der höchsten Varianz berechnete.

Um die erhaltenen SFA-Komponenten mit natürlichen Neuronen zu vergleichen, wurden die optimalen Stimuli für natürliche und mit SFA simulierten Zellen verglichen. Der optimale exzitatorische (inhibitorische) Stimulus für eine Zelle ist dabei definiert als das Eingabesignal, für welches die Zelle die höchste (niedrigste) Ausgabe generiert. Abbildung 2.4 zeigt die optimal exzitatorischen (S+) bzw. inhibitorischen (S-) Stimuli für die langsamsten 48 SFA-Komponenten

[6] Das rezeptive Feld einer Zelle ist der Ausschnitt des retinalen Bildes, welcher Einfluss auf die Ausgabe der Zelle haben kann. Wenn also eine Zelle beispielsweise nur auf Änderungen in einem 5×5 Pixel großen Ausschnitt des Gesamtbildes reagiert, so bildet dieser Ausschnitt das rezeptive Feld der Zelle.

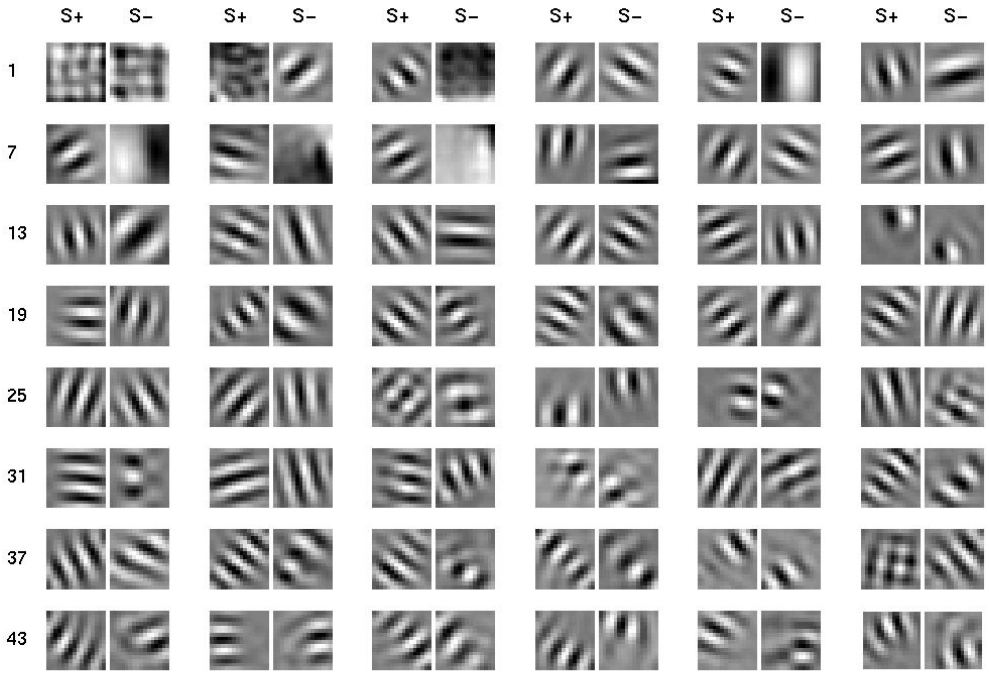


Abbildung 2.4: Optimale exzitatorische (S+) und inhibitorische (S-) Stimuli für die langsamsten 48 SFA-Komponenten. Außer den ersten beiden Komponenten weisen alle Komponenten die gleichen Charakteristiken wie *complex cells* auf. (Aus [BW02])

bzw. -Zellen, die auf den wie zuvor beschrieben generierten Daten berechnet wurden. Mit Ausnahme der ersten beiden wiesen die optimalen Stimuli der durch die SFA generierten Zellen hohe Ähnlichkeit mit experimentell ermittelten optimalen Stimuli echter Neuronen auf, d. h. fast alle Zellen und waren invariant gegenüber Phasenverschiebung, und viele Zellen wiesen eine Orientierungs- auf Frequenzabstimmung auf.

2.6.2 Grid cells und Place cells

Abgesehen von *complex cells* können mit der SFA (in Kombination mit ICA bzw. Wettbewerbslernen) auch *grid cells*, *head-orientation cells* und *place cells* gelernt werden [FSW07]. Die *place cells* wurden dabei aus den *grid cells* durch einen zusätzlichen ICA-Schritt gewonnen. Diese Zelltypen kodieren abstraktere Eigenschaften wie die räumliche Lage bzw. die Orientierung eines Lebewesens im Raum und konnten bei verschiedenen Nagern experimentell nachgewiesen werden ([OD71], [TMR90]). Sie treten dann auf, wenn beispielsweise eine Ratte ihren Weg durch ein Labyrinth suchen muss. Dabei lernt die Ratte nach einiger Zeit unabhängig von ihrer eigenen Orientierung, wo im Raum sie sich befindet, d. h. je nach ihrer Position im Raum feuern bestimmte spezialisierte Zellen.

Um diese Zelltypen zu erhalten, wurde eine 3D-Umgebung simuliert, durch welche sich eine simulierte Ratte bewegen und visuelle Eingabedaten erhalten konnte. Die Ratte verfolgte

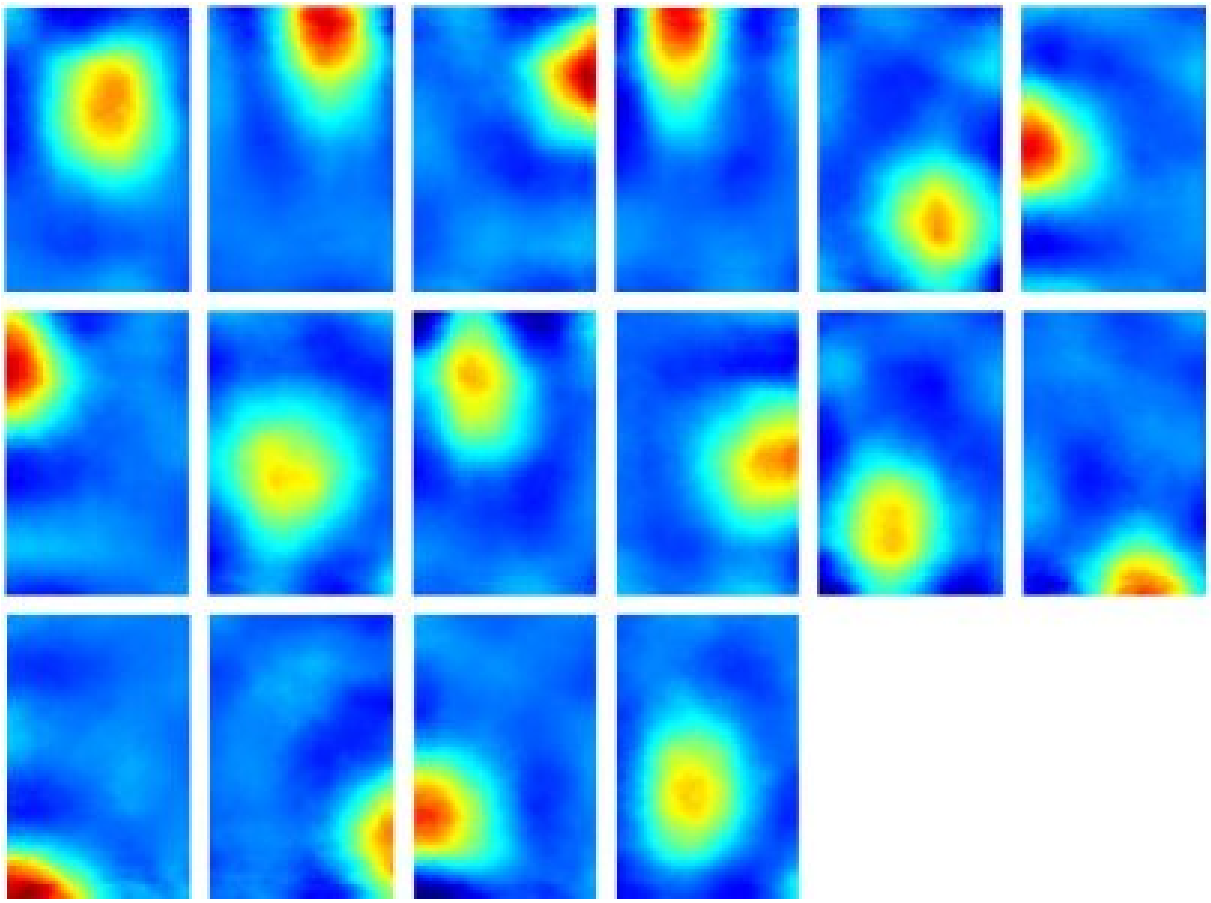


Abbildung 2.5: *Place cells*, welche von der SFA und einem ICA-Schritt nur aus egozentrischen visuellen Daten einer simulierten Ratte gewonnen wurden. (Aus [FSW07])

verschiedene Bewegungsmuster durch den Raum, in denen sie über längere Zeiträume hinweg Eingabedaten sammelte. Die so generierten Daten wurden dann in eine mehrstufige, zu immer größeren rezeptiven Feldern konvergierenden Kaskaden von SFA-Einheiten gegeben. Einige der letzten SFA-Einheiten schließlich wiesen starke Ähnlichkeiten mit *grid cells* auf. Diese sind im Gegensatz zu *place cells* nicht räumlich festgelegt, sondern feuern an mehreren Orten im Raum, jedoch in einem bestimmten Muster: So diskretisieren *grid cells* den Raum, indem sie nur an bestimmten Kästchen eines fiktiven Gitters feuern, welches man sich über den Ursprungsraum gelegt vorstellen muss. Verschiedene *grid cells* feuern nun an verschiedenen Kästchen innerhalb des Gitters. Dies lässt sich dadurch ausdrücken, dass ihr Reaktionsverhalten nicht stochastisch unabhängig ist. Daher lassen sich durch einen ICA-Schritt auf der Menge der *grid cells* Zellen ähnlich der *place cells* generieren, deren Verhalten stochastisch unabhängig und daher räumlich lokalisiert ist.

2.6.3 Extraktion versteckter Variablen aus dynamischen Systemen

In [Wis03b] wird eine interessante vom biologischen Kontext losgelöste Applikation der SFA vorgestellt, die Analyse nichtstationärer Zeitreihen. Stationarität ist zwar bei der Untersuchung von

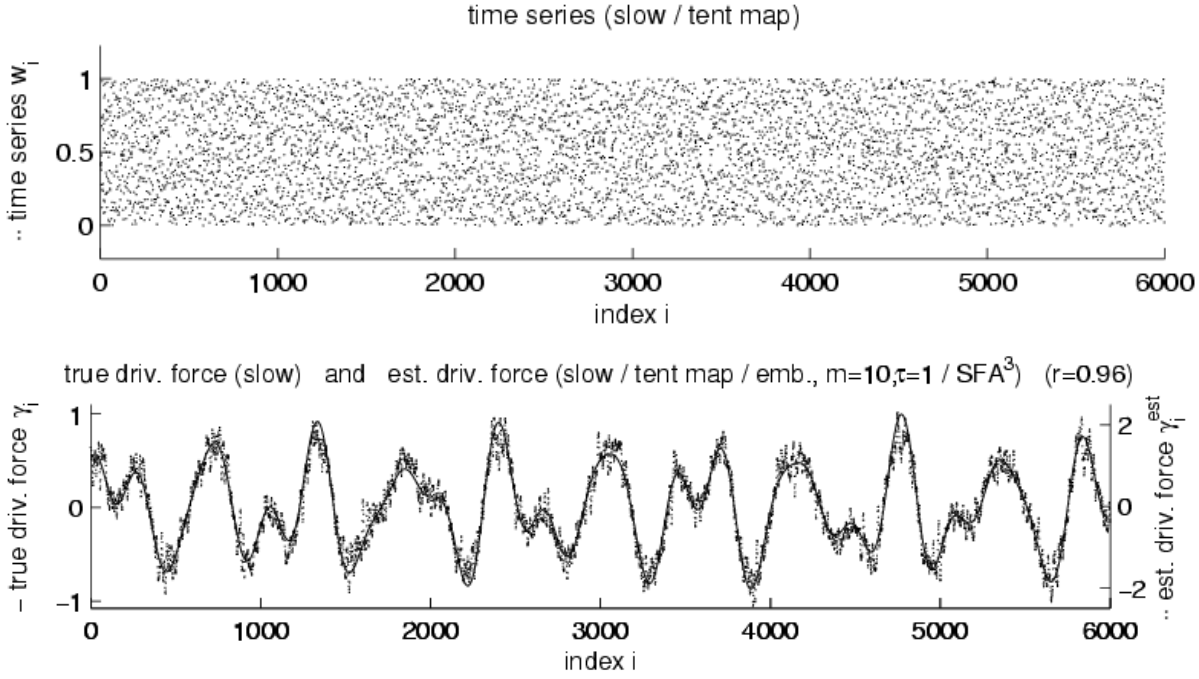


Abbildung 2.6: Oben: Die Ausgabe der Tent-Map-Funktion bei Variation des Parameters μ. Das zugrundeliegende System erscheint chaotisch und keinerlei Struktur ist zu erkennen. **Unten:** Verlauf des echten Parameters (durchgezogene Linie) und des von der SFA extrahierten (gepunktete Linie). Die Korrelation von Original- und rekonstruiertem Signal liegt bei $\rho = 0.96$. (Aus [Wis03b])

Zeitreihen eine gängige Annahme, jedoch entspricht sie in der Regel nicht den realen Gegebenheiten. Gerade im Bereich Sensorik sind langsame Drifts oder seltene, aber plötzlich vorkommende Sprünge im Wertebereich von Sensoren nicht ungewöhnlich. In der Analyse von EEG-Daten beispielsweise schränkt das fast unvermeidliche langsame Verrutschen der am Kopf der Versuchsperson befestigen Sensoren sowie deren Austrocknung und der dadurch verminderten elektrischen Leitfähigkeit die generalisierte Interpretation der Daten ein. Die explizite Modellierung solcher Nichtstationaritäten ist jedoch im allgemeinen kompliziert oder nicht möglich.

In dem genannten Artikel wird eine theoretische Möglichkeit zur Erkennung von Nichtstationaritäten mit der SFA in einer Simulation aufgezeigt. Als Grundlage dient eine so genannte *Tent Map* [Cas97], welche beispielsweise durch $f_\gamma(t) = \gamma \min(x, 1-x)$ definiert werden kann. Der Gegenstand der Untersuchung ist dann das dynamische zeitdiskrete System $x(t+1) = f_\gamma(x(t))$. Je nach Wahl des Parameters γ weist dieses System vorhersagbares oder chaotisches Verhalten auf. Der Name *Tent Map* rührt dabei daher, dass ihr Bildbereich im Intervall $[0, 1]$ bei $\gamma = 1$ die Form "$\bigwedge$" annimmt.

In der Simulation wurde der Parameter γ langsam variiert und dient somit als versteckte Variable (in dem Artikel wird sie als *Driving force*, zu deutsch etwa *treibende Kraft*, bezeichnet) für das auf der Tent Map basierende dynamische System. Dabei wurde γ nur in dem Wertebereich variiert, in welchem das System chaotisches Verhalten aufweist. Das Ausgabesignal der Tent Map wurde zeitlich eingebettet (siehe 2.5.6) und der entstandene Vektor in die SFA gegeben. Für die

Einbettung wurden jeweils die letzten zehn zeitlich aufeinanderfolgenden Datenpunkte verwendet. Abbildung 2.6 zeigt die Ausgabe der Tent Map, den veränderten Parameter γ sowie die von der SFA ausgegebene langsamste Komponente. Es ist sehr gut zu erkennen, dass die SFA die Veränderung des Parameters γ ohne explizite Kenntnis der Struktur des dynamischen Systems rekonstruieren kann. Weitere erfolgreiche Versuche wurden mit anderen chaotischen Systemen und langsamer bzw. abrupt variierenden Parameterverläufen unternommen.

Während die Problematik besteht, die Parameter für Zeiteinbettung in Hinblick auf die effiziente und machbare Berechnung zu wählen, konnte in [KK09] auch gezeigt werden, dass durch ungeeignete Parameterwahl die versteckten Variablen nicht oder falsch durch die SFA extrahiert werden. Sobald dem Verlauf der versteckten Variable nämlich selbst eine langsame Komponente zugrunde gelegt werden kann, z. B. weil sie einer frequenzmodulierten Sinusfunktion entspricht (siehe z. B. $\tilde{x}_1(t)$ in 2.3.1), so findet die SFA möglicherweise ein Signal, welches langsamer als die Variable ist. Die Autoren stellen die Hypothese auf, dass dies einerseits mit der Wahl der Parameter für die zeitliche Einbettung, andererseits mit der Vorhersagbarkeit des dynamischen Systems (Wertebereich von γ bei der Tent Map) zusammenhängt.

2.6.4 Dimensionsreduktion für bestärkendes Lernen

Ein wichtiges biologisch motiviertes Lernverfahren ist das so genannte *bestärkende Lernen* oder *Reinforcement Learning (RL)* [SB98]. Die Idee des Verfahrens ist, einen Agenten selbständig eine Strategie für eine Aufgabenstellung erlernen zu lassen, indem man ihn je nachdem, wie gut seine Strategie ist, belohnt oder bestraft. Genauer gesagt werden Zustände als erstrebenswert oder nicht markiert, und der Agent soll versuchen möglichst nur erstrebenswerte Zustände zu erreichen. Die Belohnung bzw. die Bestrafung ist ein positiver oder negativer numerischer Wert, welcher als *Reward* bezeichnet wird. Eine gute Strategie im Sinne des RL ist nun, dass der Agent sich merkt, welche Zustände hohen Reward haben und welche Aktionen er zum Erreichen dieser Zustände ausführen muss. Das heißt, der Reward eines Zustands wird auf die Zwischenzustände, die durchlaufen werden müssen, um ihn zu erreichen, zurückpropagiert.

Das Verfahren ist in den letzten Jahren auf vielfältige Art und Weise kritisiert und verfeinert worden. Eines der schwerwiegendsten Probleme des RL ist seine Skalierbarkeit. Im einfachsten Fall weist der Agent jedem Zustand oder jedem Zustands-Aktions-Paar einen Wert zu, der anzeigt, ob es sich um einen Zustand handelt, der den Reward des Agenten maximiert. Wird nun der Zustandsraum bzw. dessen Dimensionalität sehr groß, muss eine große Anzahl an Zuständen repräsentiert werden. Eine Möglichkeit dessen Herr zu werden, ist ein Dimensionsreduktionsverfahren zu benutzen, um den Zustandsraum kompakter darzustellen. In [LWW10] wird vorgestellt, wie SFA zur Dimensionsreduktion für das RL verwendet werden kann. Der Zustandsraum besteht aus einem 155×155 Pixel großen Bild, in welchem ein Agent in Form eines Fisches dargestellt ist, welcher ein bestimmtes Ziel erreichen müssen. Es wird gezeigt, dass die durch die SFA extrahierten Komponenten ermöglichen, diese komplexe Lernaufgabe mithilfe des RL zu lösen.

2.7 Zusammenfassung

In diesem Kapitel wurde das Verfahren der Slow Feature Analysis ausführlich beschrieben. Es wurden das Lernverfahren sowie die Implementation erläutert und an einem einfachen Beispiel die Funktionsweise der SFA aufgezeigt. Ebenso wurde gezeigt, dass die Verwendung der SFA in der Robotik durchaus motiviert ist, wie die aufgezeigte Verwandtschaft von quadratischer SFA und Quadriken zeigt. Quadriken stellen auch eine von mehreren Methoden dar, welche der Analyse von SFA-Lösungen dienen und in den späteren Kapiteln dazu genutzt werden, um ein tieferes Verständnis sowohl für die Vorgehensweise des Algorithmus als auch die Problemstellung zu bekommen.

Desweiteren wurden Untersuchungen bzgl. der Verwandtschaft der SFA zu anderen Verfahren ebenso wie entwickelte Erweiterungen vorgestellt. Die Anzahl der vorgestellten Arbeiten macht deutlich, dass die SFA eingehend und aus verschiedenen Blickwinkeln beleuchtet worden ist und immer noch Forschungsgegenstand ist. Das vielseitige Interesse lässt zudem darauf schließen, dass diesem Verfahren eine hohe Bedeutung und ein großes Potential zugerechnet wird. Dies wird auch anhand der bisher realisierten Anwendungen, welche zum großen Teil in den Bereich der theoretischen Biologie und der Neurowissenschaften fallen, deutlich.

Im nächsten Kapitel wird die Sichtweise auf eine andere Ebene verlegt, indem künstliche neuronale Netze eingeführt werden, und der Nutzen dieses Modells für die Betrachtung der SFA untersucht wird.

Kapitel 3

Neuronale Implementation der SFA

Wie eingangs erwähnt, distanzieren sich derzeitige Strömungen in der kognitiven Robotik von der *Good Old-Fashioned Artificial Intelligence*, welche Verhalten komplexer Organismen und Systeme für durch rein symbolische Ansätze erfassbar hielt. Neben den zuvor erwähnten Konzepten des *Embodiments* und der *Situatedness* werden auch neue Prinzipien postuliert, wie der Agent Aktionen zur Interaktion mit seiner Umwelt generieren soll: Die Hypothese ist, dass intelligentes Verhalten aus parallel ablaufenden und untereinander schwach gekoppelten Prozesse emergiert, die mit dem sensomotorischen Apparat verbunden sind [PB06].

Die in dieser Arbeit verwendete Roboterplattform ist gemäß dieser Anforderungen konzipiert, um Erkenntnisse über komplexes Verhaltens in lebenden Organismen zu gewinnen. Die Roboter sind mit redundanter Sensorik ausgestattet und darauf optimiert, dass Steuerungen mittels neuronaler sensomotorischer Schleifen implementiert werden können. Dabei kommt ein einfaches Neuronenmodell zum Einsatz, welches im folgenden Abschnitt erläutert wird. Dieses Neuronenmodell fängt zwar in keiner Weise die Komplexität natürlicher Neuronen ein; die Überlegung, sensomotorische Prozesse auf dem Roboter in diesem Modell zu implementieren, hat jedoch im Hinblick auf die Aussagekraft der gewonnen Implementation einen klaren Vorteil: Die Neuronen operieren wie ihre natürlichen Pendants lokal und erlauben nicht ohne weiteres die Anwendung aufwendiger mathematischer Verfahren, die nicht in einer konnektionistischen Struktur vorliegen. Die Hypothese ist *nicht*, dass im menschlichen Gehirn keine mathematisch komplexen Verfahren realisiert sind – eher ist damit gemeint, dass ihre Realisation im Gehirn eine andere ist. In der Tat ist es an vielen Beispielen ersichtlich, dass durch lokale Prozesse komplexe Verfahren implementiert werden können. Als Beispiel sei die Principal Component Analysis genannt: Die mathematisch exakte Lösung beruht auf einer Eigendekomposition der Kovarianzmatrix des Eingabesignals, jedoch kann sie ebenso durch eine lokale, die *Hebbsche Lernregel*, in einem neuronalen Netz implementiert werden [Oja82]. Somit hilft die Einschränkung, dass einfache – seien es neuronale oder andere konnektionistische – Modelle benutzt werden müssen, nicht den Blick dafür zu verlieren, wie komplexes Verhalten aus einem Wechselspiel von lokalen Modulen emergiert und durch gezielte Modifikation der Kopplungen dieser Module erlernt werden kann.

Neuronale Implementation der SFA-Lernregel

Vorab zu erwähnen ist, dass sich die in diesem Kapitel vorgestellte neuronale Implementation auf den Ausführungsschritt der SFA beschränkt. Es erfolgt also keine Implementation einer Online-Lernregel für die SFA. Dies geschieht aus folgenden Gründen nicht im Rahmen dieser Arbeit: Zunächst existieren wie in der Einleitung erwähnt für das Langsamkeitsprinzip bereits verschiedene Gradienten-basierte Verfahren [Mit91, Bec93, KED+01], welche teils auf neuronalen Strukturen beruhen, teils einfach auf solche übertragen werden könnten. Diese Ansätze sind allerdings nicht ohne weiteres kompatibel mit der SFA, d. h. die Lernregeln können nicht so umformuliert werden, dass sie auf die von der SFA gelernten Gewichtsvektoren angewandt werden können. Man müsste daher andere neuronale Verfahren in Betracht ziehen, wie solche für die Principal Component Analysis, welche neuronal beispielsweise durch *generalisiertes Hebbsches Lernen (GHL)* implementiert werden kann [San89]. Jedoch bestehen die langsamsten Komponenten der SFA nicht aus den Signalen mit der höchsten, sondern der geringsten Varianz; während man in der Regel aber nur an ein paar der langsamsten Komponenten interessiert ist, müssten mit GHL alle Komponenten extrahiert werden. Dies ist insbesondere für hochdimensionale Eingabesignale oder Expansionen nicht praktikabel. Daher müsste man Algorithmen zur *Minor Component Analysis (MCA)* verwenden, welche entgegengesetzt zur PCA die Signale mit der geringsten Varianz extrahieren. So wie für die PCA Algorithmen auf Basis der hebbschen Lernregel existieren, existieren für die MCA inzwischen verschiedene neuronale Algorithmen auf Basis der *anti-Hebbschen Lernregel*[1]. Jedoch weisen gerade viele der aktuellen Verfahren, die nicht nur die *Minor Component*, sondern *Minor Subspaces* extrahieren sollen, noch Instabilitäten bzgl. ihrer Konvergenz auf[2]. Obwohl also eine prinzipielle Idee für die Implementation einer Online-Lernregel für die SFA besteht, würde eine genaue Evaluation der verfügbaren Algorithmen im Hinblick auf die Verwendung als Online-Lernregel für die SFA den Rahmen dieser Arbeit sprengen.

Der Aufbau dieses Kapitels ist folgender: Zunächst wird das verwendete Neuronenmodell vorgestellt. Daraufhin werden Komponenten zur effizienten neuronalen Implementation des Ausführungsschrittes der SFA entwickelt. Zuletzt werden in sehr knapper Form die Grundlagen digitaler Filterung präsentiert und neuronale Komponenten zur Filterung vorgestellt, welche in Kapitel 5 benötigt werden.

3.1 Neuronenmodell

Das verwendete Neuronenmodell ist ein Derivat des Modells, welches ursprünglich von den Neurobiologen McCulloch und Pitts [MP43] vorgeschlagen wurde und in Abbildung 3.1 links abgebildet ist. Ein Neuron besitzt n Eingänge $x_1, \ldots, x_n$ sowie einen Ausgang y. Die Eingänge werden zunächst zu einer internen *Aktivierung* a durch Faktoren w_i gewichtet aufsummiert. Das Ausgangssignal y wird berechnet, indem eine *Transferfunktion* $f(a)$ auf die Aktivierung angewandt

[1] Bereits in [WS02] wurde bemerkt, dass die SFA durch eine *anti-Hebbsche Lernregel* implementierbar ist.
[2] Für eine Übersicht verschiedener solcher Algorithmen siehe beispielsweise [Fio02].

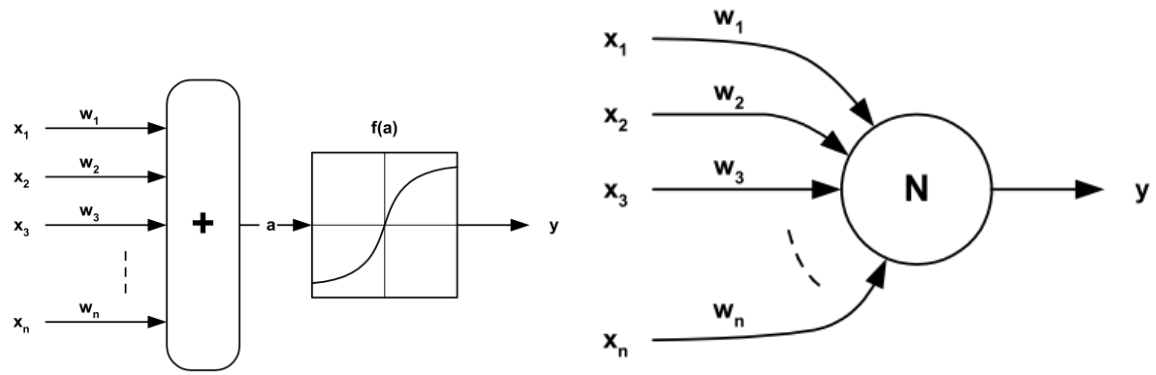


Abbildung 3.1: Links: Neuronenmodell nach McCulloch und Pitts. **Rechts:** Vereinfachte graphische Darstellung eines Neurons, bei welcher der Biasterm in das Neuron geschrieben wird. (Aus [Hil07])

wird. Insgesamt ergibt sich also für die Berechnung der Aktivität eines Neurons:

$$y := f(a), \qquad a := \sum_{i=1}^{n} w_i x_i, \qquad w_i, x_i \in \mathbb{R} \tag{3.1}$$

Das Modell ist zeitdiskret, d. h. jeweils in einem Zeitschritt wird die aktuelle Ausgabe jedes Neurons eines Netzwerkes berechnet. Zwar ist bekannt, dass echte Neuronen eher so genannten *spiking neurons* ähneln, welche zeitkontinuierlich modelliert werden und Aktivität nicht durch die Stärke des Ausgangssignals (amplitudenkodiert), sondern durch die Feuerungsrate (frequenzkodiert) kodieren[3], jedoch sind diese Modelle aufwendiger in der Berechnung und Handhabung. Gerade aus dem Grund, dass die neuronalen Netze effizient auf der Roboterhardware laufen sollen, wird daher das soeben eingeführte zeitdiskrete Neuronenmodell verwendet.

Der Einfachheit halber werden zudem die Eingangsgewichte w_i zum Gewichtsvektor $\mathbf{w} \in \mathbb{R}^N$ und die Eingänge zu $\mathbf{x} \in \mathbb{R}^N$ zusammengefasst, wodurch sich für die Berechnung des Ausgangssignals die Skalarproduktschreibweise anbietet. Wird die Tatsache integriert, dass sich ein Neuron als zeitdiskretes dynamisches System betrachten lässt, ergibt sich als Berechnungsvorschrift für die Ausgabe eines einzelnen Neurons:

$$y(t+1) = f(\mathbf{w}^T(t)\mathbf{x}(t)) \tag{3.2}$$

Neuronales Netz

Werden mehrere Neuronen miteinander verbunden, so entsteht ein *neuronales Netz*. Bei der Verbindung ist lediglich zu beachten, dass immer nur Ausgänge von Neuronen an die Eingänge anderer Neuronen gekoppelt werden, nie umgekehrt. Im Kontext eines Netzes unterscheidet man verschiedene Neuronentypen:

Eingabeneuronen. Das Eingabeneuron besitzt keinerlei Eingänge, sondern nur einen Ausgang und dient als Platzhalter, um dem Netz ein Eingabesignal zur Verfügung zu stellen. Im Fall des Roboters sind dies Sensorwerte oder Ausgaben anderer neuronaler Module.

[3] Für eine ausführliche Einführung zu spiking neurons siehe z. B. [GK02].

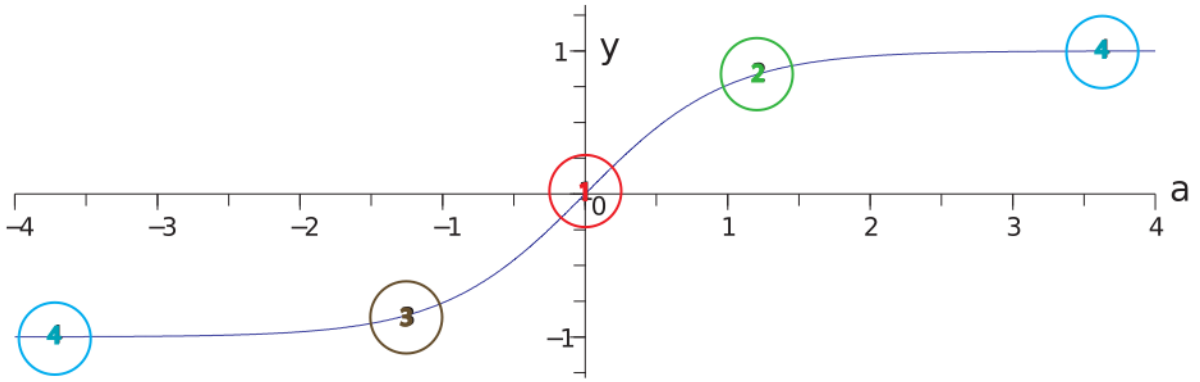


Abbildung 3.2: Der tangens hyperbolicus und seine verschiedenen Arbeitsbereiche. (Aus [Wer08])

Ausgabeneuronen. Diese Neuronen besitzen $1, \ldots, n$ Eingänge, welche aufsummiert und ohne Anwendung der Transferfunktion (d. h. $y = \mathbf{w}^T\mathbf{x}$) als Ausgabe des Netzes dienen. Die Ausgabe kann entweder an andere neuronale Module oder Aktuatoren des Roboters geleitet werden.

Versteckte Neuronen. Versteckte Neuronen entsprechen dem oben beschriebenen Neuronentyp mit Aktivierung und Transferfunktion, welches Eingabewerte zu einer Ausgabe y verarbeitet.

Biasneuron. Das Biasneuron ist ein Spezialfall des Eingabeneurons, jedoch dass es immer den gleichen konstanten Wert zurückgibt. Wie im folgenden Abschnitt zur Transferfunktion erläutert wird, kann es dazu genutzt werden, den Arbeitsbereich eines Neurons zu verschieben.

Zudem kann zwischen *Feed-Forward-Netzen*, bei dem keine Schleifen bzgl. der Neuronenverbindungen erlaubt sind, und *rekurrenten Netzen*, welche die Rückführung des Ausgabesignals in dasselbe Neuron oder ein in dem Zeitschritt bereits berechnetes Neuron gestattet, unterschieden werden.

Transferfunktion

Um die Definition des Neurons abzuschließen, wird als letztes die Transferfunktion eingeführt. Die Transferfunktion sollte gewisse Eigenschaften haben, welche sich als nützlich für die Verwendung und Analyse der neuronalen Netze erweisen:

Nichtlinearität. Mit Hilfe der Nichtlinearität der Transferfunktion kann ein neuronales Netz nichtlineare Eigenschaften aus den Eingangssignalen extrahieren. Zudem machen mehrschichtige Feed-Forward-Netze erst durch die Verwendung von nichtlinearen Transferfunktionen Sinn.[4]

[4] Betrachten wir zwei in Reihe geschaltete Neuronen mit linearer Transferfunktion $f(x) = bx + c$ und jeweils einem Eingang, so sieht man, dass die Ausgabe auch durch ein Neuron berechnet werden kann: $y_1 = f(w_1 x) = bw_1 x + c$ und $y_2 = w_2 y_1 = bw_2 y_1 + c = \underbrace{b^2 w_1 w_2}_{\tilde{w}} x + \underbrace{bcw_2 + c}_{\tilde{c}} =: \tilde{w}x + \tilde{c}$.

Beschränktheit. Damit die Ausgabe eines Neurons nicht ins Unendliche steigt, sollte die Transferfunktion einen beschränkten Bildbereich haben. Dies ist besonders für Lernalgorithmen wichtig, welche die Gewichte $\mathbf{w}$ minimieren oder maximieren.

Differenzierbarkeit und Umkehrbarkeit. Beispielsweise für die Formulierung von Lernregeln ist es von Nutzen, wenn die Transferfunktion stetig differenzierbar ist und eine Umkehrfunktion besitzt.

All diese Eigenschaften werden durch den *tangens hyperbolicus* (tanh) erfüllt, welcher in Abbildung 3.2 gezeigt ist. Zudem sind die verschiedenen *Arbeitsbereiche* des tanh mit Zahlen markiert; je nachdem in welchem Bereich sich die Eingangssignale befinden, weist der tangens hyperbolicus nämlich verschiedene Transfercharakteristiken auf.

Sind die Eingangssignale beispielsweise sehr klein und bewegen sich um die Null, so befindet sich das Neuron im linearen Arbeitsbereich (1), in welchem das Eingangssignal nahezu unverändert auf das Ausgangssignal übertragen wird. Ist das Eingangssignal stattdessen sehr groß ($\approx |x| > 2$), so befindet es sich im Sättigungsbereich (4) des tangens hyperbolicus, womit sich dieser wie die *Sprungfunktion* (d. h. es werden nur Werte ≈ -1 bzw. $\approx +1$ ausgegeben) verhält. Desweiteren besitzt der tangens hyperbolicus jeweils einen quasi-logarithmischen (2) sowie quasi-exponentiellen (3) Arbeitsbereich.

Durch Skalierung der Eingangsgewichte mit einem konstanten Faktor und Verschiebung durch ein Biasneuron lassen sich die verschiedenen Arbeitsbereiche auch gezielt ausnutzen.

Weitere Argumente für die Verwendung des tangens hyperbolicus als Transferfunktion sowie seine effiziente Berechnung durch Lookup-Tables sind in [Hil07] ausgeführt.

3.2 Komponenten der neuronalen SFA

Nach der Erläuterung des Neuronenmodells folgt nun die Beschreibung neuronaler Strukturen zur Ausführung der SFA. Dazu werden Komponenten für die eigentliche Merkmalsextraktion, die nichtlineare Expansion sowie die zeitliche Einbettung vorgestellt.

3.2.1 SFA-Einheit

Während die Lösung des Eigenwertproblems, welches für Durchführung des in Abschnitt 2.3 vorgestellten SFA-Algorithmus notwendig ist, einen hohen rechnerischen Aufwand hat, ist die Anwendung einer offline bereits gelernten SFA sehr einfach zu realisieren. Zur Anwendung der gelernten SFA-Komponenten sind lediglich folgende Schritte notwendig:

1. **Nichtlineare Expansion** des Eingabesignals
2. **Mittelwertzentrierung** des expandierten Signals
3. **Anwendung** der SFA-Gewichtsmatrix

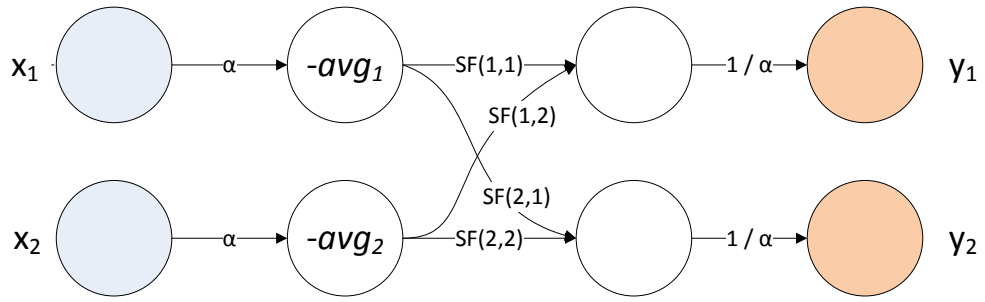


Abbildung 3.3: Anwendung einer offline gelernten SFA auf ein zweidimensionales Eingangsignal. Im ersten Schritt wird das Signal skaliert, um in den linearen Arbeitsbereich des tanh zu gelangen, und dann mittelwertzentriert. Als nächstes wird die Multiplikation mit der SFA-Matrix durchgeführt und im letzten Schritt das Signal schließlich wieder hochskaliert.

Abbildung 3.3 zeigt am Beispiel eines zweidimensionalen Eingangssignals, wie ein neuronales Netz aussieht, welches eine offline gelernte SFA ausführt. Während der Lernphase der SFA werden der Mittelwertvektor $\mathbf{avg} \in \mathbb{R}^n$ des Eingangssignals $\mathbf{x}(t) \in \mathbb{R}^n$ sowie die Gewichtsmatrix $\mathbf{SF} := [\mathbf{w}_1, \ldots, \mathbf{w}_n]$ gelernt. Damit hohe Eingangswerte durch den tanh nicht unnötig verzerrt werden, wird das Eingangssignal vor der Mittelwertzentrierung zudem durch einen Faktor α skaliert, welcher vom Wertebereich des Eingangssignales abhängt. Da auf den im Roboter verbauten Prozessoren mit Festkommazahlen gearbeitet wird, darf α allerdings aus Gründen der Genauigkeit nicht unendlich klein sein. Die Sensordaten der verwendeten Roboter befinden sich in einem Wertebereich von $[-1; 1]$, daher liefert die Wahl von $\alpha = 0.125$ bereits gute Ergebnisse. Damit das Ausgabesignal der SFA-Einheit sich wieder im ursprünglichen Wertebereich bewegt, werden die Signale schließlich wieder durch den Faktor $\frac{1}{\alpha}$ hochskaliert.

3.2.2 Expansion

Als nächstes wird eine Einheit benötigt, welche eine nichtlineare Expansion durchführt. Ein generelles Problem bei der Verwendung vorgestellten Neuronen ist, dass Multiplikation nicht per se möglich ist, sondern wie alle Rechenoperationen außer Addition und Subtraktion durch ein neuronales Netz angenähert werden muss. Jedoch ist stark zu vermuten, dass im Fall der SFA keine exakte Multiplikation notwendig ist, sondern die Tatsache ausreicht, dass die Expansion eine deutliche Nichtlinearität aufweist; die Hypothese, dass die SFA auch für andere Expansionen geeignet ist und in diesem Fall ihre Gewichte auf die gegebene Expansion anpasst, wurde bereits in der Arbeit [Nic06] bestätigt. Selbstverständlich muss dann aber sowohl im Lern- als auch im Ausführungsschritt mit der gleichen Expansion gearbeitet werden.

Um eine multiplikationsähnliche Funktion durch ein Netz aus tanh-Neuronen zu implementieren, müssen die nichtlinearen Arbeitsbereiche des tangens hyperbolicus geschickt ausgenutzt werden. Wie im vorigen Abschnitt erwähnt besitzt der tangens hyperbolicus jeweils einen annähernd exponentiellen sowie logarithmischen Arbeitsbereich. Da offensichtlich $xy = \exp(\ln x + \ln y)$ gilt, kann die Multiplikation auf eine Addition zurückgeführt werden, in dem in zwei Schritten

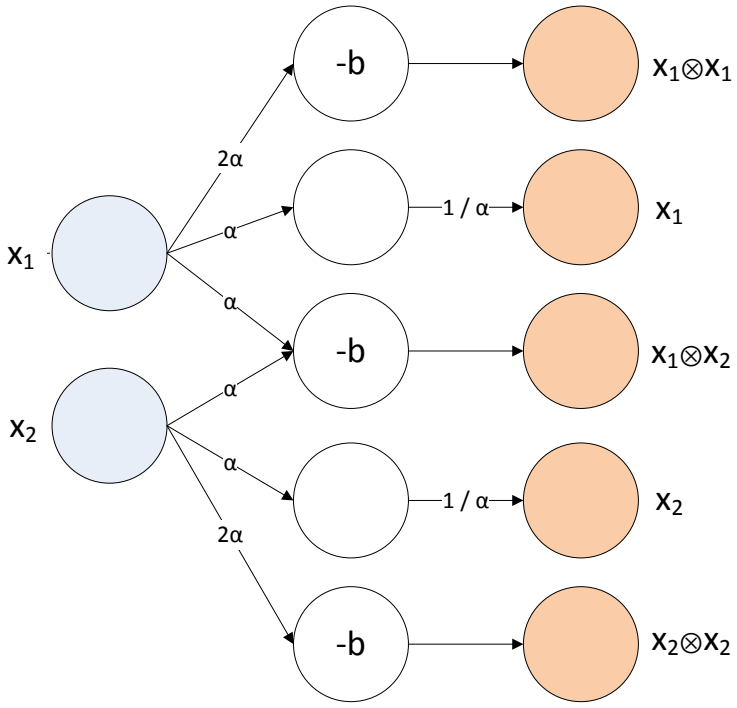


Abbildung 3.4: Expansionseinheit unter Ausnutzung der Nichtlinearität des tanh.

die Eingaben zuerst jeweils einzeln in den logarithmischen Bereich verschoben werden, und als zweites deren Summe in den exponentiellen Bereich verschoben wird. Aus solchen Multiplikationsmodulen, angewandt auf alle Tupel (x_i, x_j) des Eingangssignals $\mathbf{x}$, ließe sich dann ein Netz zur quadratischen Expansion bauen.

Neben der Annäherung einer Multiplikation kann allerdings auch ein anderer Weg gegangen werden, um eine nichtlineare Expansion vorzunehmen. Insbesondere kann die Nichtlinearität des tangens hyperbolicus direkt ausgenutzt werden und als Ersatz für die Multiplikation die Funktion

$$x \otimes y := \frac{1}{\alpha} \tanh(\alpha(x+y) - b) \approx \beta \exp(x+y), \qquad b > 0, \qquad 0 < \alpha, \beta < 1 \tag{3.3}$$

verwendet werden, welche offensichtlich die Exponentialfunktion angewandt auf eine Summe approximiert. Der Faktor α dient wie zuvor der Skalierung der Eingangssignale x und y und sollte so gewählt werden, dass $x, y \in [-0.1; 0.1]$. Der Summand b ist ein Biasterm, welcher die Verschiebung in den exponentiellen Arbeitsbereich des tangens hyperbolicus leistet. β hingegen ist kein Parameter, sondern verdeutlicht, dass die Steigung des tangens hyperbolicus in der Regel um einiges schwächer als die der Exponentialfunktion ist. Vorteile dieser Expansion sind, dass pro expandiertem Term nur genau ein Neuron benötigt wird, sowie dass die Sättigung des tangens hyperbolicus verhindert, dass die Werte ins Unendliche steigen[5]. Da sich der tangens hyperbolicus immer im Bereich $[-1; 1]$ bewegt, lässt sich der Wertebereich der SFA-Komponente sogar exakt über die Norm des zugehörigen Gewichtsvektors bestimmen.

[5]Dies gilt natürlich für alle auf dem tangens hyperbolicus basierenden Expansionen, so auch für die kurz zuvor vorgestellte approximierte Multiplikation.

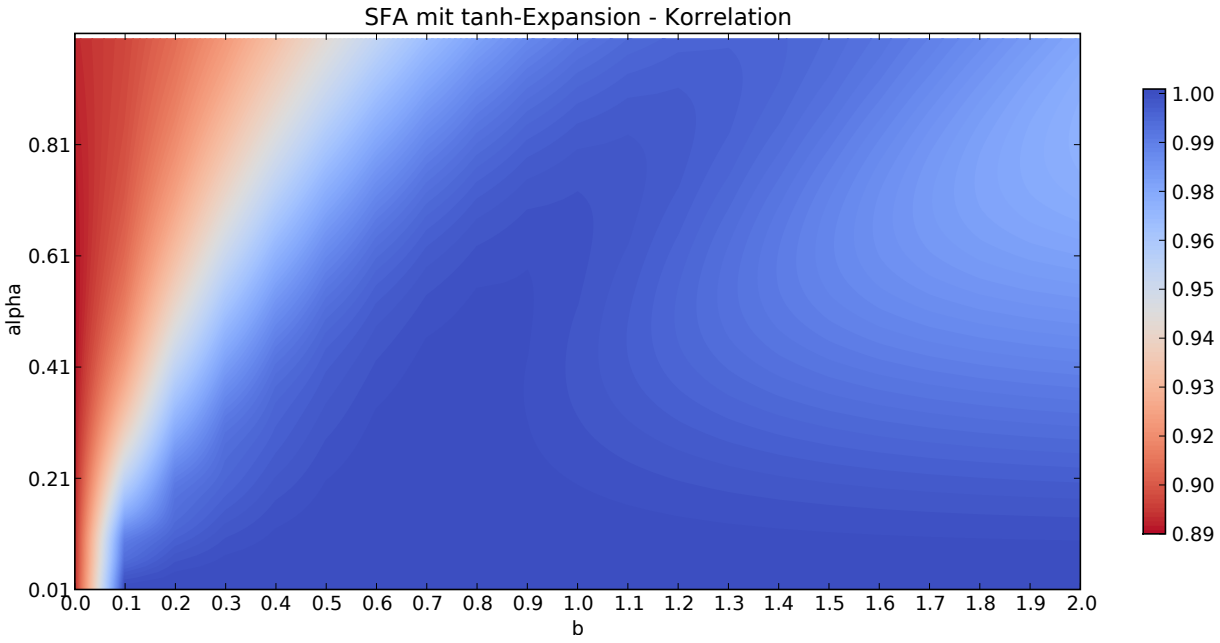


Abbildung 3.5: Korrelationskoeffizient der langsamsten Komponenten der quadratischen und der tanh-expandierten SFA in Abhängigkeit von den Parametern α und b.

Im folgenden wird die eben vorgestellte Expansion kurz tanh-Expansion genannt, außerdem wird von einer tanh-SFA gesprochen werden.

Parameterevaluation

Um herauszufinden, welche expandierten Basisfunktionen sich durch den Einfluss der Parameter α und b ergeben und welche Auswirkungen sie auf die extrahierten SFA-Komponenten haben, werden die Parameter variiert und für jede Konfiguration eine SFA mit tanh-Expansion auf mehrere Datensätze durchgeführt. Die erhaltenen Komponenten werden dann mit denen der quadratischen SFA verglichen, indem deren Korrelationskoeffizienten sowie deren η-Werte betrachtet werden. Die tanh-Expansion wird mit den Parametern $\alpha = 0.01, 0.02, \ldots, 0.99, 1.0$ und $b = 0, 0.1, \ldots, 1.9, 2.0$ evaluiert. Da der tangens hyperbolicus eine punktsymmetrische Funktion ist ($\tanh(-x) = -\tanh(x)$), genügt es Werte $b > 0$ zu betrachten.

Zum Vergleich wird zusätzlich eine lineare Pseudoexpansion herangezogen, die ebenso wie die tanh-Expansion die expandierten Terme als Summen bildet, jedoch im Gegensatz zu jener keinerlei Nichtlinearität beinhaltet. Diese stellt aber keine sinnvolle Expansion dar, sondern soll die Bedeutung der Nichtlinearität des tangens hyperbolicus illustrieren.

Betrachten wir zunächst das synthetische Beispielsignal aus Abschnitt 2.3.1, welches durch $\tilde{x_1}(t) = \sin(t) + \cos^2(11t)$ und $\tilde{x_2}(t) = \cos(11t)$ im Intervall $t \in [0, \ldots, 2\pi]$ gegeben ist. Abbildung 3.5 zeigt, bei welchen Parameterkonfigurationen die Korrelation der tanh- sowie der quadratischen SFA-Komponenten am höchsten ist. Es ist deutlich zu erkennen, dass sich durch die Erhöhung von α die Korrelation verschlechtert, aber auch dass mit kleinerem α die Wahl von b immer weniger ins Gewicht fällt. Zum einen lässt sich daraus schließen, dass sich das Ergebnis

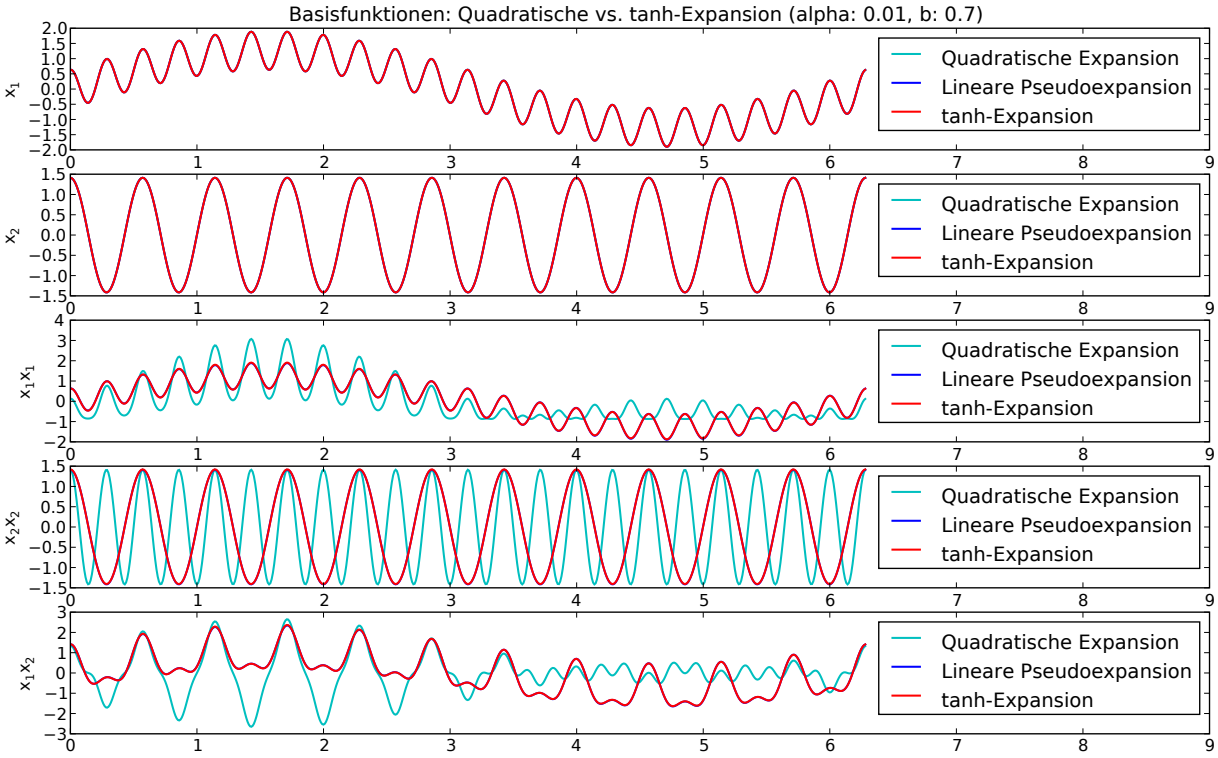


Abbildung 3.6: Basisfunktionen bei quadratischer Expansion, linearer Pseudoexpansion und tanh-Expansion ($\alpha = 0.01, b = 0.7$). Die Summen $x_i + x_j$ sind fast identisch mit den tanh-expandierten Termen $x_i \otimes x_j$, weswegen sich die Kurven überlagern.

verschlechtert, wenn das Signal vor dem tanh nicht genug gestaucht wird, zum anderen, dass im Bereich $b = 0.6, \ldots, 0.9$ die Nichtlinearität des tanh gut ausgenutzt werden kann. Die höchste Korrelation nahe Eins zwischen y_1 der quadratischen SFA sowie der SFA mit tanh-Expansion ergibt sich für $\alpha = 0.01, b = 0.7$.

Abbildung 3.6 zeigt die Basisfunktionen der quadratischen, der tanh-Expansion sowie der linearen Pseudoexpansion. Alle expandierten Signale wurden zur besseren Vergleichbarkeit jeweils mittelwertzentriert und zu einer Varianz von Eins normiert (es wurde allerdings keine Dekorrelation der Signale durchgeführt). Tatsächlich erscheinen die gemischten tanh-Terme fast identisch zu denen der linearen Pseudoexpansion, also $x_1 \otimes x_2$ ähnelt stark $x_1 + x_2$ etc.; auch die Korrelationskoeffizienten der tanh-Basisfunktionen mit den linearen Basisfunktionen liegen sehr nahe 1. Dass die Terme jedoch nicht vollkommen identisch sind, zeigt sich dadurch, dass die lineare Pseudoexpansion nicht genügt, um das erwartete langsame Sinussignal zu finden. Abbildung 3.7 zeigt, dass die – wenn auch nur schwach sichtbare – Nichtlinearität des tangens hyperbolicus benötigt wird, damit die Komponente y_1 durch die SFA richtig zu extrahiert werden kann. Bei der linearen SFA kann der Term $\cos^2(11t)$ nicht aus y_1 elimiert werden, da keine Möglichkeit besteht, das Quadrat des Kosinus anzunähern. Dies gelingt allerdings mit der tanh-Expansion: Die Korrelation zwischen y_1 der quadratischen SFA und der tanh-SFA liegt bei $0.999999914703 \approx 1$.

Die Parameterverteilung bestätigt sich auch für reale Daten. Zur Evaluation wurden Beschleunigungsdaten des A-Serie-Roboters verwendet: statische Sequenzen, in welcher der Robo-

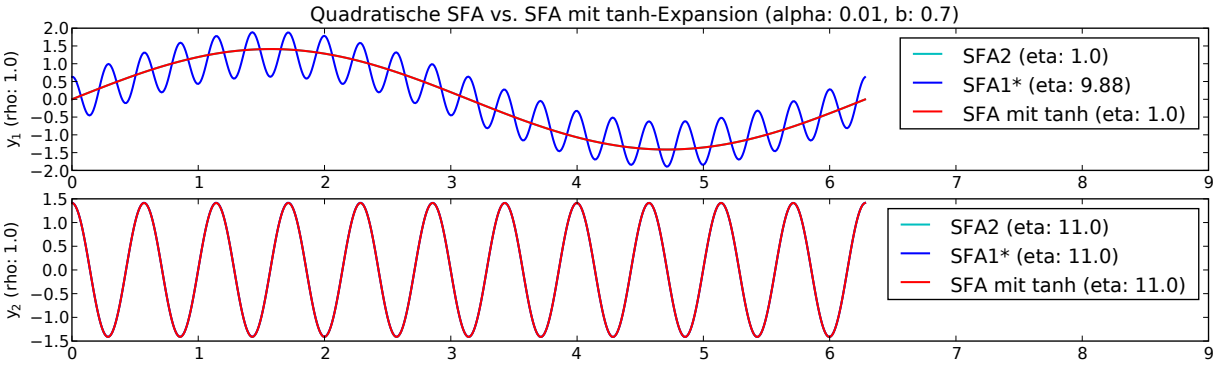
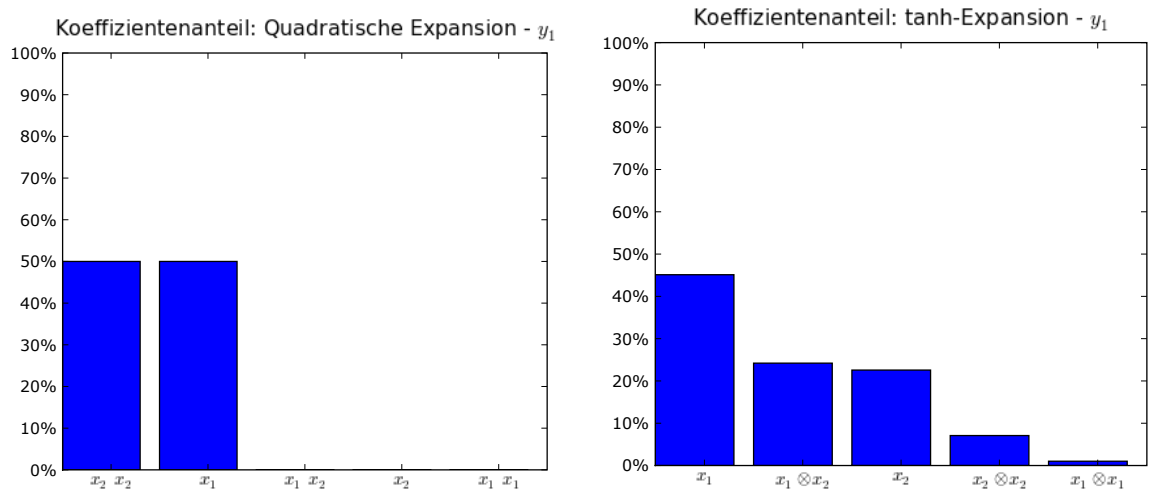


Abbildung 3.7: Gegenüberstellung von quadratischer SFA, SFA mit linearer Pseudoexpansion (SFA1*) und einer tanh-SFA ($\alpha = 0.01, b = 0.7$).

ter verschiedene Posen ausführt (siehe Abschnitt 4.1.2), sowie dynamische Sequenzen, in welcher der Roboter läuft und schließlich umfällt (siehe Abschnitt 5.2.1). Es ergibt sich jeweils wieder $\alpha = 0.01, b = 0.7$ als Parameterkonfiguration mit der höchsten Korrelation zur quadratischen SFA, ebenso ist die Verteilung der Korrelationen in Abhängigkeit von den Parametern ähnlich. Der Übersicht halber sind an dieser Stelle aber nur Ergebnisse für die synthetischen Daten abgebildet.

Von Interesse ist nun, wie trotz offensichtlich sehr unterschiedlicher Basisfunktionen die gleichen langsamsten Komponenten gefunden werden. Dies soll am Beispiel der synthetischen Daten betrachtet werden. Zunächst wird ein Nachteil der hinsichtlich der Korrelation zu den quadratischen SFA-Komponenten optimalen Parameter deutlich: Die hohe Stauchung der Eingangswerte durch das niedrige α hat zur Folge, dass die SFA-Gewichte w_{ij} unter Umständen sehr hohe Werte annehmen. Durch die hohe Stauchung verringert sich nämlich der nichtlineare Einfluss des tangens hyperbolicus, wodurch die Basisfunktionen, welche nützliche Nichtlinearitäten beinhalten, deutlich stärker gewichtet werden. Ansonsten kommen die benötigten Nichtlinearitäten nicht klar genug zum Vorschein, um in einer Linearkombination die hochfrequenten Anteile des langsamsten Signals zu entfernen. Diese Problematik erschwert zum einen eine direkte Evaluation der langsamsten Komponenten und ihrer Koeffizienten. Zum anderen ergibt sich dadurch aber vor allem ein gravierender Nachteil für die reale Hardware, welche mit eingeschränkten Wertebereichen operiert. Daher empfiehlt es sich den Parameter α solange zu erhöhen, bis eine tatsächliche Verschlechterung des Ergebnisses, sei es bzgl. der Korrelation zur quadratischen SFA-Komponente oder aus einem anderen Zweck, zu beobachten ist.

Abbildung 3.8 stellt die Gewichtsvektoren für y_1 bzgl. der verschiedenen Expansionen dar. Allerdings wird an dieser Stelle aus eben genannten Gründen der Parameter $\alpha = 0.1$ verwendet, b bleibt unverändert. Während die quadratische Expansion für y_1 das erwartete Ergebnis liefert, findet die tanh-Expansion die Komponenten durch eine andere Linearkombination, welche die Eigenschaften der zur Verfügung stehenden Basisfunktionen verwendet. Tatsächlich kann bei der tanh-SFA keine der Basisfunktionen weggelassen werden, ohne dass die langsamste Komponente deutlich von ihrem richtigen Kurvenverlauf abweicht.



(a) Quadratische SFA (b) tanh-SFA

Abbildung 3.8: SFA-Gewichtsvektoren für y_1 bei quadratischer sowie tanh-Expansion.

Abschließend lässt sich sagen, dass die tanh-Expansion eine sinnvolle und effizient berechenbare Alternative zur quadratischen SFA darstellt. Insbesondere eignet sie sich sehr gut für die Verwendung auf den verfügbaren Roboterplattformen. Zwar erlauben die wenigen betrachteten Testdaten weder eine Garantie dafür, dass die gefundenen Parameter die optimale Wahl darstellen, noch dass jede mit der quadratischen SFA berechenbare Komponente auch durch eine tanh-SFA gefunden werden kann. Letztere Gefahr der Nichtdarstellbarkeit verschwindet jedoch zunehmend mit einer größeren Anzahl an Eingangsdimensionen; intuitiv liegt das daran, dass sich Funktionen beliebig genau durch endliche Reihen mit geeigneten Basisfunktionen approximieren lassen, beispielsweise Polynome bei der Taylorreihe oder Sinusoidale bei der Fourierreihe (Approximationssatz von Stone-Weierstraß). In diesem Sinne kann auch eine Menge von geeignet gewählten, verschieden parametrisierten tangens-hyperbolicus-Funktionen verwendet werden. Daher ist davon auszugehen, dass bei geeigneter Wahl der Parameter α und b die tanh-SFA eine ausreichende Approximation darstellt. In Kapitel 5 zur Anwendung der SFA in einer sensomotorischen Schleife wird die tanh-SFA noch einmal aufgegriffen und gezeigt, dass diese sich tatsächlich für den praktischen Einsatz eignet.

3.2.3 Zeitliche Einbettung

Eine weitere Komponente, welche für die vorgestellten Anwendungen der SFA benötigt wird, ist die zeitliche Einbettung von Signalen. Glücklicherweise lässt sich dies mit dem verwendeten zeitdiskreten Neuronenmodell sehr einfach bewerkstelligen. Wir erinnern uns, dass in Abschnitt 2.5.6 die für die zeitliche Einbettung verwendeten Parameter m, die Gesamtanzahl der Zeitschritte, sowie τ, der Abstand zwischen den Zeitschritten, genannt wurden. Das Netz zur Einbettung eines eindimensionalen Signals ist in Abbildung 3.9 abgebildet: Es besteht aus m Neuronen, einem pro Verzögerungsschritt, und von jedem τ-ten Neuron führt eine Synapse zu einem Ausgabeneuron. Die Abbildung zeigt dabei den Fall für $\tau = 1$. Wie zuvor wird das Eingangssignal vor und

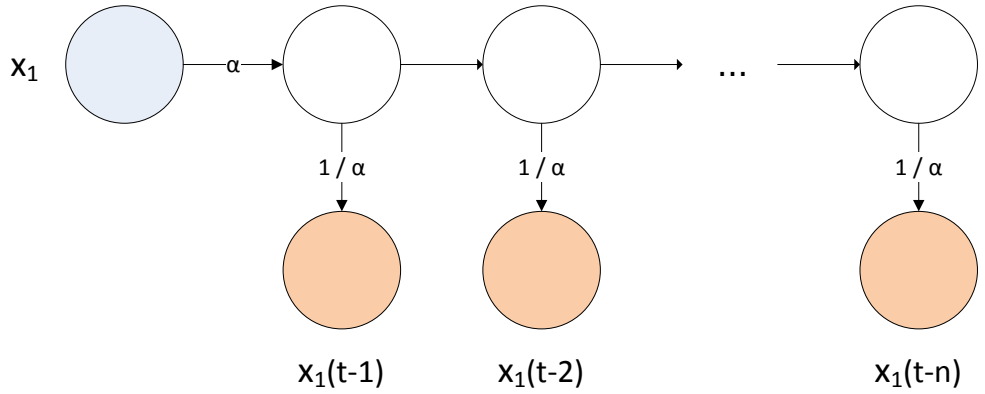


Abbildung 3.9: Zeitliche Einbettung eines Eingangssignals durch sukzessive zeitliche Verzögerung für $\tau = 1$.

nach dem Eintreten ins neuronale Netz so skaliert, dass es durch die Nichtlinearität des tangens hyperbolicus nicht verzerrt wird.

3.3 Neuronale Filterung

In diesem Abschnitt sollen einige Grundlagen der digitalen Filtertechnik ebenso wie die neuronale Realisierung solcher Filter erklärt werden. Diese Kenntnis dieser Strukturen wird in Kapitel 5 benötigt, wenn ein humanoiden Laufmusters mit der SFA analysiert und modifiziert wird. Dabei ist im Rahmen dieser Arbeit nur eine sehr knappe Abhandlung der Grundlagen möglich, welche sich vor allem an [Zak05] orientiert.

3.3.1 Digitale Filter

Digitale Filter werden zur Manipulation von Signalen verwendet, meist zum Sperren oder Durchlassen bestimmter Frequenzbereiche. Häufige Verwendung aufgrund ihrer einfachen Analyse und Synthese finden die so genannten *linearen zeitinvarianten Systeme* (LZI). Lineare Filter sind solche Systeme, und man unterscheidet zwischen *FIR*- und *IIR*-Filtern, welche sich bzgl. ihres Verhaltens auf einen Einheitsimpuls[6], in ihrer *Impulsantwort* unterscheiden: Erstere haben eine endliche Impulsantwort (engl.: *finite impulse response*), letztere eine unendliche (engl.: *infinite impulse response*). Insbesondere sind FIR-Filter nichtrekursiv implementierbar, d. h. sie weisen in der Regel[7] keine Rückkopplungen auf, während IIR-Filter immer Rückkopplungszweige in ihrer Struktur beinhalten. Die allgemeine Filtergleichung für ein Filter mit einem jeweils eindimensionalen, zeitdiskreten Eingangssignal $x(t)$ und Ausgangssignal $y(t)$ ist gegeben durch:

$$d_0 y(t) = \sum_{m=0}^{p} c_m x(t-m) - \sum_{m=1}^{q} d_m y(t-m). \tag{3.4}$$

[6] Im diskreten Fall entspricht der Einheitsimpuls einem potentiell unendlichen Signal, welches aus Nullen besteht und an einer Stelle einen Wert von Eins annimmt, also $x = [\ldots, 0, 0, 0, 1, 0, 0, 0, \ldots]$.

[7] Es gibt spezielle FIR-Filterstrukturen, die Rückkoppelungen aufweisen, welche im folgenden aber nicht betrachtet werden.

Die Konstanten p und q bestimmen die Ordnung des Systems, d. h. $p = 0, 1, 2, \ldots$, $q = 0, 1, 2, \ldots$ und $d_0 = 1$ (per Konvention). Wenn $q = 0$ und $p > 0$, so ist das System nichtrekursiv und somit ein FIR-Filter, für $q > 0$ und $p > 0$ handelt es sich um ein IIR-Filter.

Wenn der genaue Frequenzgang des gewünschten Filters bekannt ist, kann man sich Verfahren wie der *Z-Transformation* bedienen, um ein Filter zu synthetisieren. Dieses Verfahren wird in dieser Arbeit allerdings nicht verwendet und daher wird nicht weiter darauf eingegangen. Es ist allerdings wichtig anzumerken, dass die genannten Arten von Analyse und Synthese nur für lineare Filter möglich sind; ansonsten kann kein Frequenz- und Phasengang bestimmt werden. Anschaulich ist das so zu erklären, dass durch die Linearitätsbedingung eine eindeutige Trennung der Frequenzanteile des Ausgangssignals bei Kenntnis der linearen Filterstruktur und -koeffizienten überhaupt erst möglich ist. Betrachten wir das Eingangssignal als eine Summe von Sinus- und Kosinusschwingungen, so können durch lineare Filter nur die Amplitude sowie die Phase der Signale (durch die Filterkoeffizienten und die zeitliche Verzögerung) beeinflusst werden, nicht aber die Frequenzen. Bei einer nichtlinearen Transformation hingegen, beispielsweise durch Einführung der Multiplikation der Eingangssignale, können neue Schwingungen im Ausgangssignal auftreten, welche Vielfache der Frequenzen von Schwingungen im Eingangssignal aufweisen. Die Frequenzanteile dieser Schwingungen lassen sich daher nicht von der Impulsantwort des Filter ablesen.

Von IIR zu FIR

Es gibt eine einfache Möglichkeit, aus einem gegebenen IIR-Filter ein FIR-Filter mit ähnlichen Eigenschaften zu generieren. Alle linearen Filter sind vollständig durch ihre – im Falle von IIR-Filtern unendliche – Impulsantwort bestimmt. Um aus einem IIR-Filter ein FIR-Filter zu erhalten, muss die Rekursion aufgerollt werden: Dazu wird zunächst die Impulsantwort $h(t)$ des IIR-Filters über m Zeitschritte gemessen. Um ein FIR-Filter mit der gleichen Impulsantwort während dieser m Zeitschritte zu erhalten, können die Werte der Impulsantwort des IIR-Filters direkt als Koeffizienten des FIR-Filters übernommen werden, also $c_i = h(i)$ für $i = 1, \ldots, m$. Alternativ kann zusätzlich eine *Fensterfunktion* $w(n)$ verwendet werden, womit sich die Filtergewichte durch $c_i = h(i)w(i)$ berechnen. Fensterfunktionen werden zur weiteren Anpassung der Filtereigenschaften wie Übergang zwischen Durchlass- und Sperrbereich und Dämpfung im Sperrbereich verwendet. Beispiele für Fensterfunktionen sind das Hamming-Fenster $w(n) = 0.54 + 0.46\cos(\frac{2\pi n}{M})$ oder das von-Hann-Fenster $w(n) = \frac{1}{2}\left[1 + \cos(\frac{2\pi n}{M})\right]$ jeweils mit Fensterbreite M und $n = -\frac{M}{2}, \ldots, \frac{M}{2} - 1$.

3.3.2 Neuronale Darstellung

Die Darstellung 3.4 lässt sich unter der Verwendung rekursiver Synapsen sowie der zeitlichen Einbettung aus dem letzten Abschnitt direkt in ein neuronales Modell übersetzen:

$$\tilde{y}(t) = \sum_{m=0}^{q} \tanh(c_m x(t-m)) - \sum_{m=1}^{p} \tanh(d_m \tilde{y}(t-m)). \tag{3.5}$$

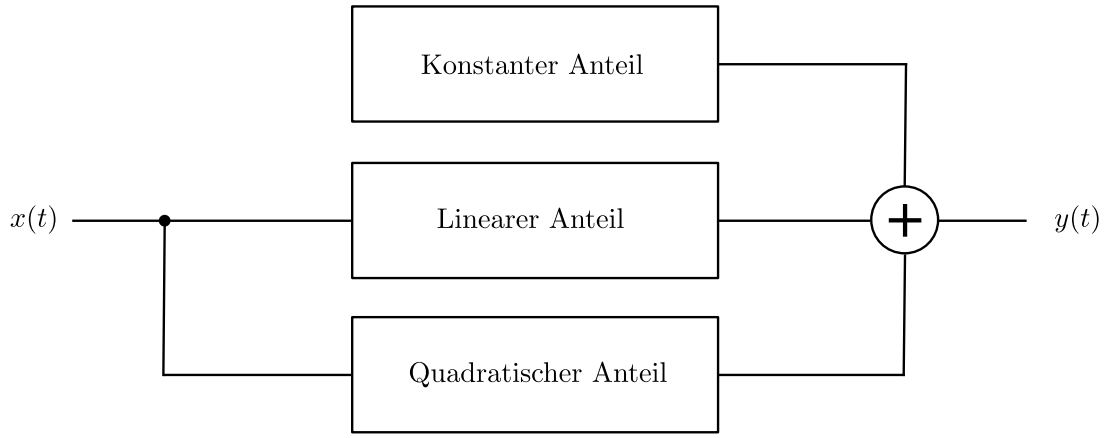


Abbildung 3.10: Schematische Ansicht eines Volterra-Filters. Die Struktur des Filters entspricht gerade der einer quadratischen SFA-Komponente.

Dabei muss darauf geachtet werden, dass das Eingangssignal eine kleine Amplitude aufweist, damit die Verzerrung des tangens hyperbolicus nicht ins Gewicht fällt und somit die Linearität erhalten bleibt. Es gilt dann $y(t) \approx \tilde{y}(t)$.

Es ist zu bemerken, dass die Übertragung eines IIR-Filters mittels der obigen Gleichung in der Regel eine Struktur liefert, welche keine unendliche Impulsantwort mehr hat. Dies liegt daran, dass das Signal durch die Sättigung des tangens hyperbolicus sukzessive in die Null gezogen wird. Ein solches Tiefpassfilter wird auch *leaky integrator* genannt. Auch bei FIR-Filtern kann es durch den tangens hyperbolicus durch eine Abschwächung des Signals bei mehreren in Reihe geschalteten Neuronen kommen. Dieser Effekt kann mit Verbindungsgewichten größer Eins kompensiert werden.

3.3.3 Nichtlineare Filter: Volterra-Filter

Im Abschnitt 2.5.6, in welchem die SFA mit zeitlicher Einbettung eingeführt wurde, wurde bereits die Verwandtschaft der SFA mit der Volterra-Reihe bzw. den Volterra-Filtern aufgezeigt. Im folgenden wird das Volterra-Filter zweiter Ordnung vorgestellt, welches schematisch in Abbildung 3.10 dargestellt ist und durch folgende Gleichung ausgedrückt werden kann:

$$\begin{aligned} y(t) = {} & h_0 + \\ & \sum_{m_1=0}^{m} h_1(m_1) x(t-m_1) + \\ & \sum_{m_1=0}^{m} \sum_{m_2=0}^{m} h_2(m_1, m_2) x(t-m_1) x(t-m_2), \end{aligned} \tag{3.6}$$

wobei m die zeitliche Einbettung darstellt und $h_N(m_1, m_2, \ldots, m_N)$ dem Kernel N-ten Grades des *Volterra-Filters* bezeichnet. Die Gleichung liefert somit einen Spezialfall der allgemeinen Volterra-Expansion aus Gleichung 2.57. Die Kernel werden im folgenden auch als Koeffizienten bezeichnet.

Um die Schreibweise zu vereinfachen, soll im folgenden die Filtergleichung in Vektor- und Matrixschreibweise verwendet werden,

$$y(t) = h_0 + \mathbf{h_1}(t)^T \tilde{\mathbf{x}}(t) + \tilde{\mathbf{x}}(t)^T \mathbf{H_2}(t) \tilde{\mathbf{x}}(t), \tag{3.7}$$

wobei $\tilde{\mathbf{x}} := [x(t-m\tau), x(t-(m-1)\tau), \ldots, x(t-2\tau), x(t-\tau), x(t)]$ das zeitliche eingebettete Eingangssignal bezeichnet. Auch dieses Filter kann analog zur Vorgehensweise im vorigen Abschnitt auf einfache Weise neuronal realisiert werden.

Adaption von Volterra-Filtern

Wie zuvor erwähnt, ist das Volterra-Filter zweiter Ordnung äquivalent zur quadratischen SFA mit zeitlicher Einbettung, was bedeutet, dass die SFA eine unüberwachte Lernmethode zur Bestimmung der Kernel darstellt. In Abschnitt 5 wird diese Relation aufgegriffen, um die SFA als Filter in einer sensomotorischen Schleife zu verwenden. Dabei werden die Ergebnisse mit einfachen überwachten Lernverfahren zur Adaption der Filterkoeffizienten verglichen.

Klassischerweise findet man die Filterkoeffizienten bzw. Kernel für ein Volterra-Filter durch überwachte Lernregeln wie *least mean squares (LMS)* oder *recursive least squares (RLS)*. Bei solchen überwachten Lernverfahren werden die Kernel schrittweise so optimiert, dass ein vorgegebenes Trainingssignal durch Anwendung des Filters auf ein Eingabesignal möglichst gut approximiert wird. Beim LMS-Verfahren wird dazu bzgl. des mittleren quadratischen Fehlers minimiert. In dieser Arbeit wird nur der Einfachheit halber das langsamer konvergierende, aber stabile LMS-Verfahren aus [Zak05] angewandt.

Bezeichne $y(t)$ das vom Filter berechnete Ausgangssignal und $d(t)$ das erwünschte Ausgangssignal. Der Abschätzungsfehler $e(t)$ ergibt sich durch:

$$e(t) := d(t) - y(t). \tag{3.8}$$

Die Updateregel für die Filtergewichte in jedem Zeitschritt lässt sich wie folgt formulieren:

$$h_0(t+1) := h_0(t) + \mu_0 e(t) \tag{3.9}$$

$$\mathbf{h_1}(t+1) := \mathbf{h_1}(t) + \mu_1 e(t) \tilde{\mathbf{x}}(t) \tag{3.10}$$

$$\mathbf{H_2}(t+1) := \mathbf{H_2}(t) + \mu_2 e(t) \tilde{\mathbf{x}}(t) \tilde{\mathbf{x}}(t)^T, \tag{3.11}$$

wobei die Lernrate für den konstanten Teil mit μ_0, für den linearen Teil mit μ_1 und für den quadratischen Teil mit μ_2 bezeichnet wird. $\tilde{\mathbf{x}}(t)$ bezeichnet die zeitliche Einbettung des Eingangssignals $\mathbf{x}(t)$ Die angegebenen Gleichungen lassen sich einfach aus dem LMS-Ansatz herleiten, d. h. mittels der Update-Regeln wird der quadratische Fehler $e^2(t)$ minimiert.

3.4 Zusammenfassung

In diesem Kapitel wurde ein zeitdiskretes Neuronenmodell vorgestellt, welches auf der in dieser Arbeit verwendeten Roboterplattform zum Einsatz kommt. Es wurde gezeigt, wie der Ausführungsschritt der SFA auf dieses Neuronenmodell übertragen werden kann. Insbesondere wurde

gezeigt, wie die nichtlineare Transferfunktion der Neuronen, der tangens hyperbolicus, ausgenutzt werden kann, um eine effiziente nichtlineare Expansion zu implementieren. Im letzten Teil wurde vorgestellt, dass digitale Filter auf sehr einfache Weise neuronal implementiert werden können, und dass eine strukturelle Äquivalenz zwischen SFA und Volterra-Filtern besteht. Die vorgestellten Filter werden in Abschnitt 5 aufgegriffen, wenn die SFA als Filter in einer sensomotorischen Schleife zum Einsatz kommt.

Teil II

Praxis

Kapitel 4

Posenerkennung und Dimensionsreduktion

Eine wichtige Aufgabenstellung in der humanoiden Robotik ist das Erlernen einer geeigneten Repräsentation körperlicher Zustände. Ein autonomer Agent soll dabei auf Basis seiner eigenen Handlungen Modelle bilden, die ihm ermöglichen, eigene statische und dynamische Zustände zu klassifizieren und wiederzuerkennen. Im ersten Schritt ist man an der Repräsentation statischer Zustände interessiert, d. h. der Repräsentation von Haltung und Lage des Roboters im Raum. So ist es für einen autonomen humanoiden Roboter beispielsweise für die Planung weiterer Aktionen ungemein wichtig zu wissen, ob er gerade aufrecht steht, sitzt, liegt oder eine bestimmte Bewegung ausführt. In der Robotik und der künstlichen Intelligenz ist die Frage, auf welche Weise Agenten selbständig eine Repräsentation verschiedener Haltungen und Bewegungen lernen können, nach wie vor Gegenstand aktueller Forschung.

Zur Findung einer geeigneten Repräsentation von Roboterzuständen ist die Verwendung von Sensoren unumgänglich; im Falle der hier betrachteten A-Serie (siehe Abschnitt 1.3 und Anhang A) verfügt der Roboter über Motorpositions- sowie Beschleunigungssensoren. Erstere sind zur Bestimmung der Motor- und Gelenkpositionen und der lageunabhängigen Morphologie des Roboters dienlich, während sich in letzteren vor allen Dingen die Lage des Roboters und seiner einzelnen Körperteile in Bezug zu seiner Umwelt widerspiegelt. Es darf jedoch nicht außer acht gelassen werden, dass die Beschleunigungssensoren auch dafür verwendet werden, um Bewegungsabläufe des Roboters zu kontrollieren. Dies erfordert neben einer hohen Genauigkeit auch eine hohe Reaktivität der Sensoren, da unvorhergesehene Störungen möglichst frühzeitig durch gegensteuernde Bewegungen ausgeglichen werden müssen. In Kapitel 5 werde ich mich eingehend mit einem auf dem Roboter implementierten Bewegungsmuster beschäftigen und zeigen, wie die SFA zur Verbesserung bestimmter Aspekte dieses Bewegungsmusters eingesetzt werden kann. Da an dieser Stelle statische Posen Gegenstand der Untersuchung sind, schlägt sich die Sensibilität dieser Sensoren jedoch als ungewolltes Rauschen im Signal nieder, da im Gegensatz zu den Motorwinkelsensoren selbst in der Ruheposition immer leichte Aktivität im Beschleunigungssensorsignal zu erkennen ist.

Eine weitere wichtige Anforderung an die Posenerkennung ist die effiziente Berechenbarkeit.

Vor allem muss der Berechnungsaufwand in einem adäquaten Verhältnis zur Aufgabenstellung stehen. So erscheint es beispielsweise unverhältnismäßig, für die Unterscheidung von nur zwei Aktivitätsklassen *Stehen* und *Liegen* einen aufwendigen Klassifikationsalgorithmus auf ein hochdimensionales Signal anzuwenden, welches durch die vielen Sensoren eines Roboters gegeben ist. Stattdessen sucht man eine kompaktere Darstellung der Zustände des Roboters, welche die gleichen oder zumindest hinreichend ähnliche Eigenschaften wie die durch die Sensoren gegebene Darstellung hat.

Im folgenden Kapitel soll gezeigt werden, dass die SFA eine Alternative zu gängigen Dimensionsreduktionsverfahren sowie Methoden zur Posenerkennung darstellt. Insbesondere für die Auswertung von Beschleunigungssensoren besitzt die SFA wichtige Eigenschaften, um die Unzulänglichkeiten der Rohdaten zu beseitigen. Die SFA projiziert einen hochdimensionalen Sensordatenstrom auf ein niedrigdimensionalen Raum, welcher den Roboterzustand hinreichend charakterisiert und kompakt darstellt.

Es wird jedoch nicht auf darauf eingegangen, wie gut die von der SFA generierten Komponenten auch generalisieren. Um qualitative Aussagen über die Generalisierbarkeit machen zu können, wären eine wesentlich größere Menge an Trainingsdaten sowie ausführliche Untersuchungen notwendig, welche Posen in Abhängigkeit welcher Trainingsdaten erkannt werden, welche Trainingsdaten wie gut generalisieren etc. Stattdessen liegt der Schwerpunkt dieses Kapitels darauf, eine repräsentative Menge von Trainingsdaten mittlerer Länge sehr detailliert zu analysieren und zu evaluieren, welche Signale die SFA aus diesen Daten extrahiert, ob sich diese Signale für eine Reduktion des sensorischen Zustandsraumes eignen, und wie die SFA im Vergleich zu anderen Verfahren einzuschätzen ist.

Das Kapitel ist folgendermaßen gegliedert: Zu Beginn wird analysiert, welche Ergebnisse die SFA auf Sequenzen erzielt, in welchen der Roboter verschiedene Posen ausführt. Dazu werden zunächst bereits erzielte Ergebnisse zur Erkennung einfacher Posen mittels der SFA vorgestellt. Es folgt eine theoretische sowie praktische Betrachtung der Eigenschaften der SFA, welche für die genannten Ziele hilfreich sind. Daraufhin widme ich mich dem Thema der Dimensionsreduktion. Es werden grundlegende Konzepte erklärt, um schließlich die SFA als Verfahren zur Dimensionsreduktion für Sensordaten von einem humanoiden Roboter zu testen und zu bewerten. Zum besseren Vergleich werden die gleichen Daten wie in der Arbeit [Ste10] verwendet, in welcher bereits verschiedene Dimensionsreduktionsverfahren hinsichtlich ihrer Anwendbarkeit für die humanoide Robotik evaluiert wurden. Es wird gezeigt, dass das Lernprinzip der Langsamkeit eine sinnvolle Zielfunktion darstellt, um den Zustandsraum kompakt zu repräsentieren und eine robuste Klassifikation von Posen zu ermöglichen, wenn es auf Beschleunigungssensordaten angewandt wird.

4.1 Langsamste Komponenten

Die eigene oder fremde Haltung ist nicht nur für die Beurteilung des eigenen Zustandes und Planung weiterer Aktionen notwendig, sondern kann auch Gegenstand menschlicher Kommunikation sein. Menschliche Sprachen haben in der Regel ein ausgefeiltes Vokabular, um die eigene statische

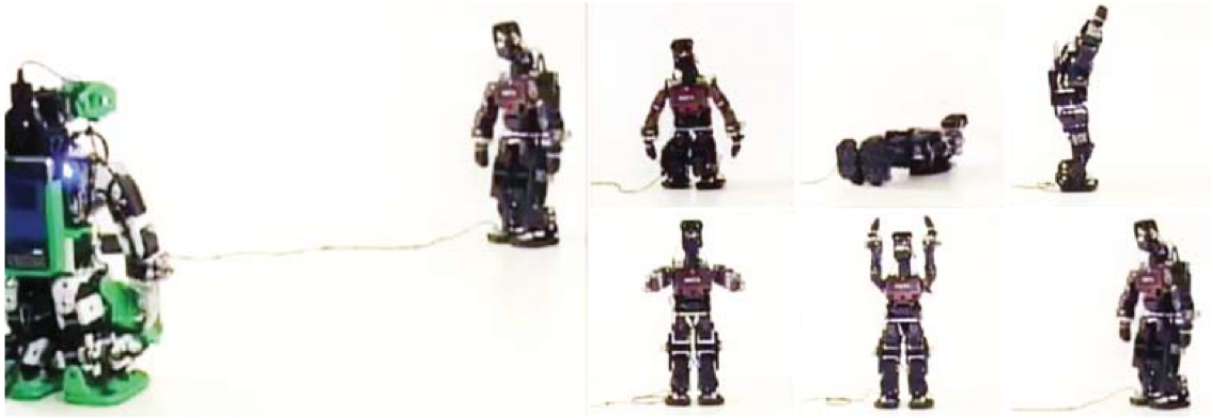


Abbildung 4.1: Ein Roboter beobachtetet einen anderen Roboter dabei, wie er verschiedene Bewegungen durchführt. (Aus [SL09])

wie auch dynamische Aktivität zu verbalisieren und somit anderen mitzuteilen. Die Fragestellung, wie Konzepte gelernt und verbalisiert werden können, ist ein aktuelles Forschungsgebiet. Ein Ansatz zur Annäherung an diese Frage ist, Kommunikation und Sprache in situierten Agenten zu simulieren und somit Sprache auf vollständig analysierbaren künstlichen Systemen zu betrachten, um dadurch Rückschlüsse auf die Entwicklung von Sprache beim Menschen zu ziehen. Hierbei spielen autonome Roboter so genannte *Sprachspiele* (engl: *Language games*), um eine gemeinsame Sprache über ihre Perzeption zu entwickeln [Ste01]. Den Robotern wird dazu ein fundamentales Framework zur Verfügung gestellt, welches ihnen erlaubt, neue Bedeutungen und Wörter zu erfinden, wahrzunehmen und zu kategorisieren. In [SL09] wurde untersucht, welche sprachlichen Phänomene zu beobachten sind, wenn die Pose eines Roboters als Thema eines Sprachspiels gewählt wird. Dabei wurde der k-Means-Algorithmus auf ein sensorisches Eingabesignal angewandt, um u. a. die Aktivitätsklassen *Aufrecht* und *Liegend* zu extrahieren.

In einer ersten Studie zur Erkennung von Roboterposen mittels SFA konnte in [SHH09] gezeigt werden, dass die Klassifikation von Posen erleichtert wird, wenn sie nicht auf den Originaldaten, sondern auf per SFA vorverarbeiteten Daten erfolgt. Die Trainingsdaten bestanden aus einer Sequenz, in welcher ein Roboter steht, läuft, mit den Armen gestikuliert, sich hinlegt und wieder aufsteht (siehe Abbildung 4.1). Als sensorische Eingabedaten wurden dabei sowohl exterozeptive visuelle als auch propriozeptive Motorpositions- und Beschleunigungssensordaten verwendet. Während die propriozeptiven Daten von dem Roboter, der verschiedene Posen ausführt, aufgenommen wurden, stammen die visuellen Daten von der Kamera eines zweiten beobachtenden Roboters. Dies stellt zwar eine unnatürlich anmutende Vereinfachung dar, da einem externen Beobachter im Realfall nicht die propriozeptiven Daten des Beobachteten zur Verfügung stehen; jedoch dient dies nur als Ersatz dafür, dass dem Roboter kein physikalisches Körpermodell zur Verfügung steht, was ihm erlauben würde, ähnliche Daten aus reiner Beobachtung selbst zu generieren. Als visuelle Daten wurden keine Rohdaten verwendet, sondern aus dem Bild wurden sieben translations- und skalierungsinvariante Features berechnet, welche Informationen über Form und Ausrichtung des Objektes gaben. Es konnte gezeigt werden, dass die quadratische SFA, ange-

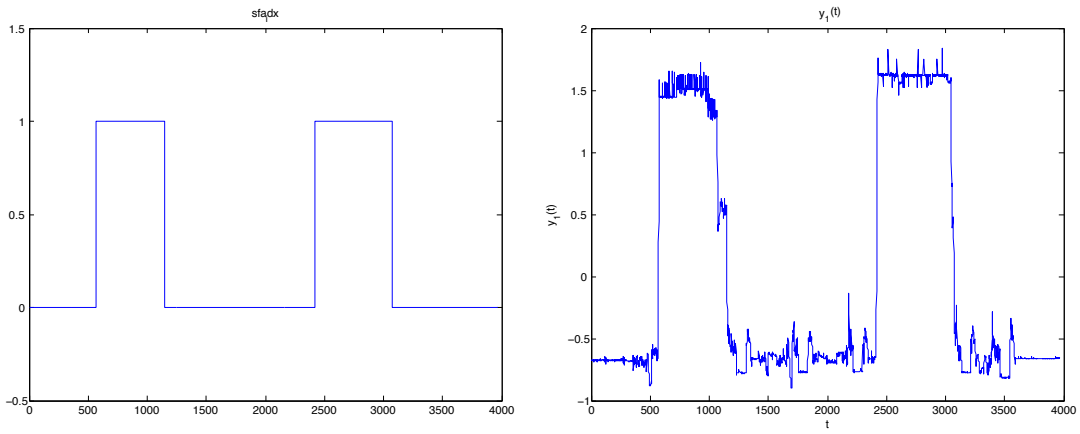


Abbildung 4.2: Links: Ground-truth-Signal. Der höhere Wert besagt, dass der Roboter zu diesem Zeitpunkt aufrecht ist, der niedrigere, dass der Roboter liegt. Während der aufrechten Position ist aber erlaubt, dass der Roboter läuft oder mit den Armen gestikuliert. **Rechts**: Die quadratische SFA extrahiert die beiden Aktivitätsklassen *Aufrecht* und *Liegend* aus einem multimodalen 86-dimensionalen Sensordatenstrom. (Aus [SHH09])

wendet auf die Gesamtzahl der Daten, eine sehr gute Klassifikation der zwei Aktivitätsklassen *Aufrecht* und *Liegend* ermöglicht (siehe Abbildung 4.2).

4.1.1 Theoretische Vorüberlegungen

Als nächstes soll erörtert werden, warum das Langsamkeitsprinzip eine gute Zielfunktion für die Trennung von verschiedenen Posen darstellt. Ähnliche Überlegungen wurden bereits in meiner Studienarbeit [Höf09] angestellt und werden an dieser Stelle aufgegriffen und weitergeführt.

Abbildung 4.3 zeigt einige Momentaufnahmen einer Roboterbewegung, in welcher der Roboter zunächst aufrecht steht und sich dann auf den Bauch legt; Abbildung 4.4 zeigt die Daten jeweils zweier ausgewählter Motorpositions- sowie Beschleunigungssensoren während dieser Sequenz. Wie eingangs erwähnt charakterisieren die Motorpositionssensoren die Morphologie des Roboters, während die Beschleunigungssensoren die Lage des Roboters zur Umwelt repräsentieren. Dies lässt sich an der vorgestellten Sequenz deutlich erkennen: Da der Körper des Roboters vor und nach der ausgeführten Bewegung ausgestreckt ist, reagieren die Motorsensoren nur während der Bewegung, die ausgewählten Beschleunigungssensoren hingegen spiegeln – da sie in der betroffenen Körperachse liegen – am Ende der Sequenz eine Lageveränderung gegenüber der Ausgangssituation wider. Jedoch sind die Beschleunigungssensoren sehr empfindlich, wodurch während des Übergangs ein sehr unruhiger Kurvenverlauf zu beobachten ist.

Die SFA kann nun aufgrund der Zielfunktion, die Langsamkeit des Signals zu optimieren, zwei Dinge leisten: Zum einen kann sie die Haltung (d. h. aufrecht, liegend oder in Schieflage) aus den Beschleunigungsdaten extrahieren, da sie in den meisten Anwendungsfällen um einiges weniger variiert als andere Aktivitäten des Roboters. Zum anderen sind die extrahierten Komponente wegen des Langsamkeitsprinzips glatter und entrauscht. In Abbildung 4.5 links ist eine hypothetische langsamste Komponente y_1 abgebildet, die genau diese beiden Punkte veranschaulicht:

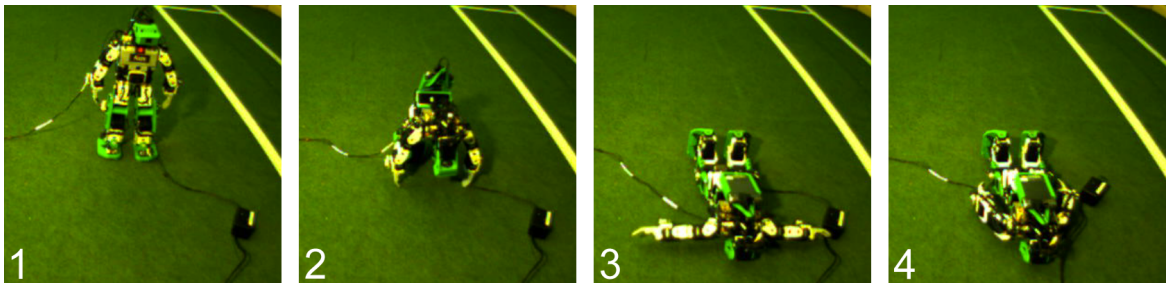


Abbildung 4.3: Momentaufnahmen einer kurzen Bewegungssequenz des Roboters.

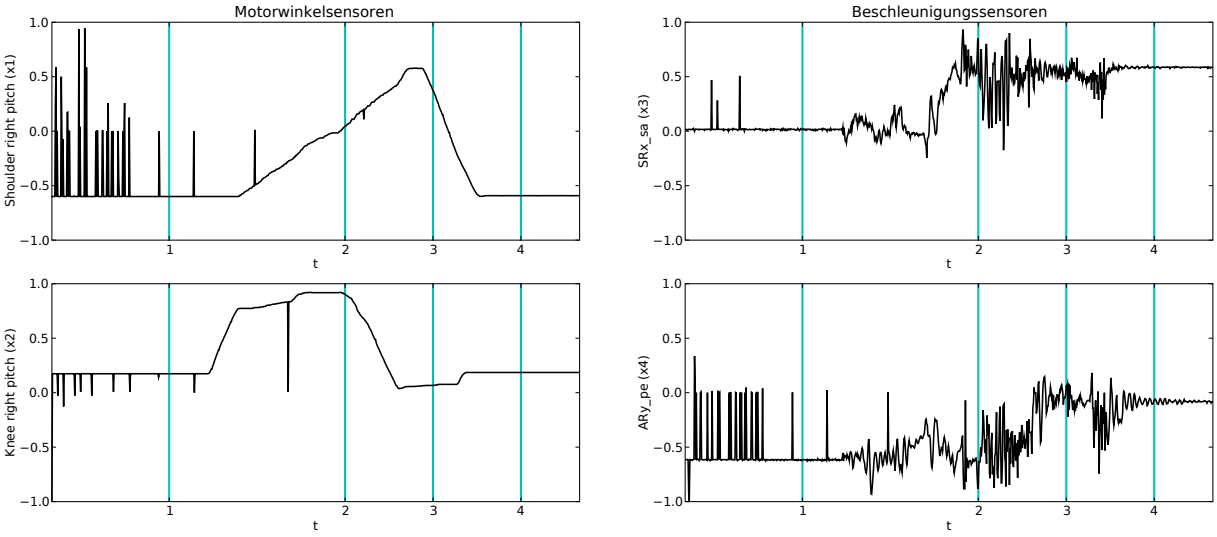


Abbildung 4.4: Ausgewählte Sensorwerte der Bewegungssequenz.

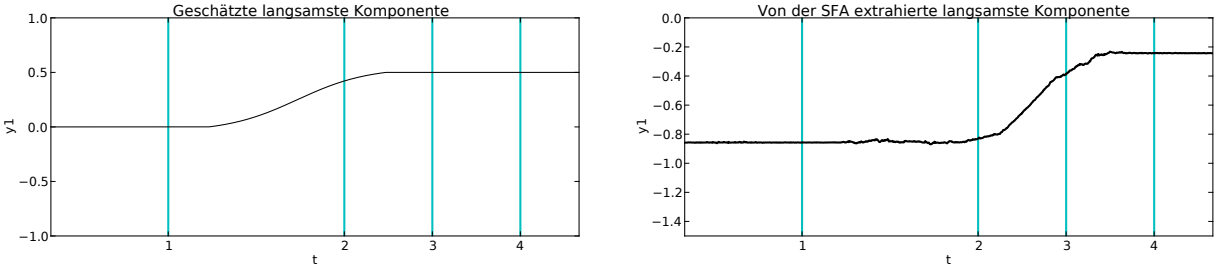


Abbildung 4.5: Geschätzte und von der SFA extrahierte langsamste Komponente.

Vergleicht man y_1 mit den Beschleunigungssensoren, so wird deutlich, dass sie die langsame Veränderung der Haltung registriert, aber keinerlei Störungen mehr aufweist. Dass dies tatsächlich erreicht werden kann, zeigt die rechte Grafik, welche die durch Wiederholung mehrerer SFA-Einheiten, auf den zuvor gezeigten Trainingsdaten und den Beschleunigungssensoren trainierte langsamste Komponente abbildet[1].

Ein paar Überlegungen beleuchten, warum gerade eine Linearkombination quadratischer Terme, wie sie die SFA findet, in Bezug auf die Beschleunigungssensoren eine gute Basis darstellt, insbesondere um die Ausrichtung des Roboters zum Boden zu extrahieren. Es ist augenscheinlich, dass die zuvor genannte Komponente y_1 starke Ähnlichkeit zum in Abbildung 4.4 rechts unten abgebildeten senkrechten Beschleunigungssensor `ARy_pe` hat. Das heißt, dass die Information der Ausrichtung bereits in den Sensordaten sichtbar ist. Die langsamste Komponente unterscheidet sich nun vor allem dadurch, dass sie keinerlei Stör- bzw. dynamische Anteile mehr aufweist, welche sich durch Erschütterungen und schnelle Bewegungen in den Beschleunigungssensoren niederschlagen. Das bedeutet, dass diese dynamischen Anteile extrahiert und vom Sensorsignal abgezogen werden müssen, um eine glatte Komponente wie y_1 zu erhalten.

Es lässt sich auf einfache Weise erklären, nach welchem Prinzip diese Anteile berechnet werden können: Betrachten wir dazu zwei Sensoren, welche auf dem Accelboard `ABAR` am Arm (siehe Abbildung 1.4 auf Seite 8) liegen, und bezeichnen deren Werte in Abhängigkeit von der Zeit mit $x_1(t)$ und $x_2(t)$. Sei $x_1(t)$ der sagittale `ARx_sa` und $x_2(t)$ der senkrechte Sensor `ARy_pe`. Die beiden Sensoren stehen orthogonal zueinander und spannen einen zweidimensionalen Vektorraum auf. Sie bilden dabei die kartesischen Koordinaten des zweidimensionalen Vektors $\mathbf{x}$, dessen Richtung und Länge von der Lage des Accelboards im Raum abhängt. Wegen der Erdgravitation ist der Sensorvektor $\mathbf{x}(t) \approx \binom{0}{0}$, wenn die Sensoren bzw. das Accelboard parallel zur Erdoberfläche ausgerichtet ist – im Fall von `ABAR`, wenn der Roboter auf der Seite liegt. Nun lässt sich der Vektor $\mathbf{x} = \mathbf{x}(t)$ statt in kartesischen auch in Polarkoordinaten angeben, womit sich die Darstellung

$$\mathbf{x} = r\cos(\phi)\begin{pmatrix}1\\0\end{pmatrix} + r\sin(\phi)\begin{pmatrix}0\\1\end{pmatrix} \tag{4.1}$$

ergibt. Die Richtung des Vektors wird also zu jedem Zeitpunkt durch einen Winkel ϕ und die Länge durch einen Koeffizienten r bestimmt. Betrachten wir die Sensoren nun wieder getrennt, so ergeben sich offensichtlich $x_1 = r\cos(\phi)$ und $x_2 = r\sin(\phi)$. Sei $\tilde{y}(\mathbf{x})$ nun eine hypothetische Funktion, welche die Quadrate der Sensorwerte x_1 und x_2 addiert:

$$\tilde{y} := x_1^2 + x_2^2 = r^2\cos(\phi)^2 + r^2\sin(\phi)^2 = r^2\underbrace{(\cos(\phi)^2 + \sin(\phi)^2)}_{=1} = r^2.$$

$\tilde{y}$ entspricht also dem Quadrat der Vektorlänge. Wie in Abbildung 4.6 deutlich zu sehen ist, enthält $\tilde{y}$ keinerlei Richtungsinformation mehr. Das bedeutet, dass zwar schnelle Wechsel während der Bewegungen oder durch Erschütterungen bei Bodenkontakt registriert werden, $\tilde{y}$ aber unabhängig von der Lage einen Wert nahe Null annimmt, wenn sich der Roboter in der Ruhestellung

[1] Die abgebildete langsamste Komponente ergibt sich nach fünf SFA-Iterationen mit 32 weitergereichten langsamsten Komponenten je Iteration, angewandt auf die Beschleunigungssensoren des Roboter in der beschriebenen Bewegungssequenz.

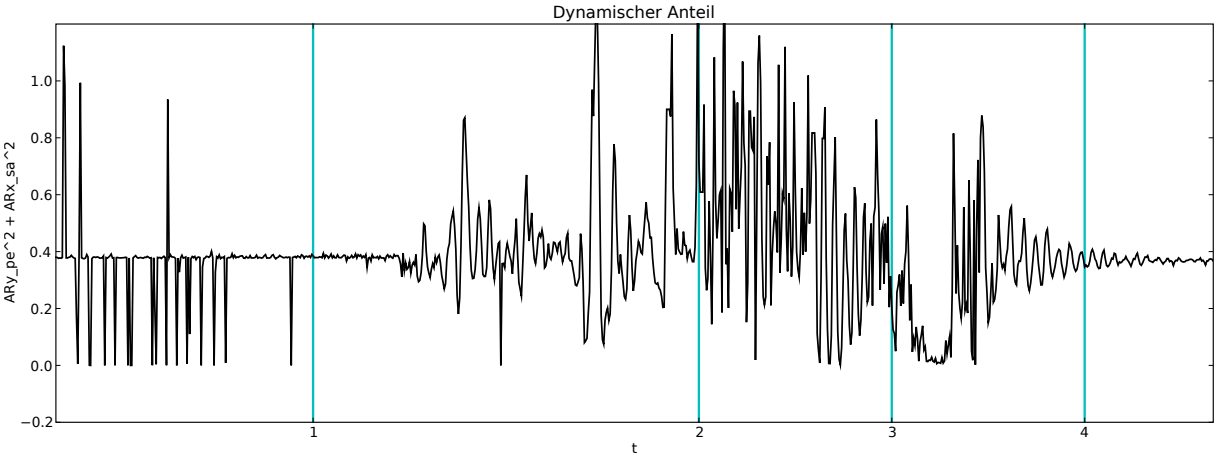


Abbildung 4.6: Dynamische Anteile der Bewegung, welche durch die Quadratesumme zweier orthogonaler Sensoren entsteht. Vergleicht man das Signal mit den Sensorwerten aus Abbildung 4.4, sieht man, dass die Information zur Ausrichtung aus dem Sensorsignal verschwunden ist.

befindet. Die quadratische SFA könnte zur Berechnung einer Ausrichtungskomponente nun $\tilde{y}$ in geeigneter Skalierung als Korrekturterm verwenden: Die Monome x_1^2 und x_2^2 stehen der SFA nach der quadratischen Expansion zur Verfügung, und sie werden mit Koeffizienten w_i und w_j gewichtet. Nehmen wir an, dass diese die Gewichte negativ und identisch sind, also $\tilde{w} := -w_i = -w_j$, so könnte eine Detektion der senkrechten Ausrichtung des Roboterkörpers durch

$$y_1(t) = g_1(\mathbf{x}) \approx w_1 x_1 + w_2 x_2 + \ldots + w_i x_1^2 + w_j x_2^2 + \ldots \underset{(w_i = w_j = -\tilde{w})}{=} w_1 x_1 + w_2 x_2 + \ldots - \tilde{w} r^2 + \ldots$$

berechnet werden. In der Praxis entstehen viele weitere solcher Korrekturterme, welche die SFA zur Glättung des verwenden kann. Das erklärt auch, warum die SFA bessere Ergebnisse liefert, wenn mehr Sensoren als Eingabe zur Verfügung stehen. Ebenso ergeben sich durch die Iteration mehrerer SFA-Einheiten immer mehr Monome höherer Potenzen, aus welchen sich weitere Korrekturterme errechnen lassen und mit deren Hilfe der Kurvenverlauf immer weiter geglättet werden kann.

4.1.2 Trainingsdaten

Als Trainingsdaten werden mehrere Bewegungssequenzen eines A-Serie-Roboters verwendet. Die aufgenommen Sequenzen haben eine Gesamtlänge von 120 Sekunden und wurden mit einer Taktrate von 100 Hz aufgenommen, d. h. pro Sekunde stehen 100 Datenpunkte zur Verfügung. Der Roboter vollführt dabei verschiedene Bewegungen und verharrt jeweils für einige Sekunden in bestimmten Posen. Insgesamt kommen in den Trainingsdaten die fünf Posen *Stehen*, *Hocken*, *Liegen (Bauch)*, *Liegen (Rücken)* und *Spagat* vor, wobei es allerdings nicht zwischen allen Posen Übergänge gibt; so kann der Roboter beispielsweise nicht aufstehen, wenn er auf dem Rücken liegt, sondern muss sich zuvor erst auf den Bauch drehen. Alle Zustände, die keiner der Posen zugeordnet werden können, werden im folgenden mit einem einzigen Zwischenzustand identifiziert. Als sensorische Qualitäten stehen 21 Winkelsensoren der Motoren sowie 16 Beschleunigungssen-

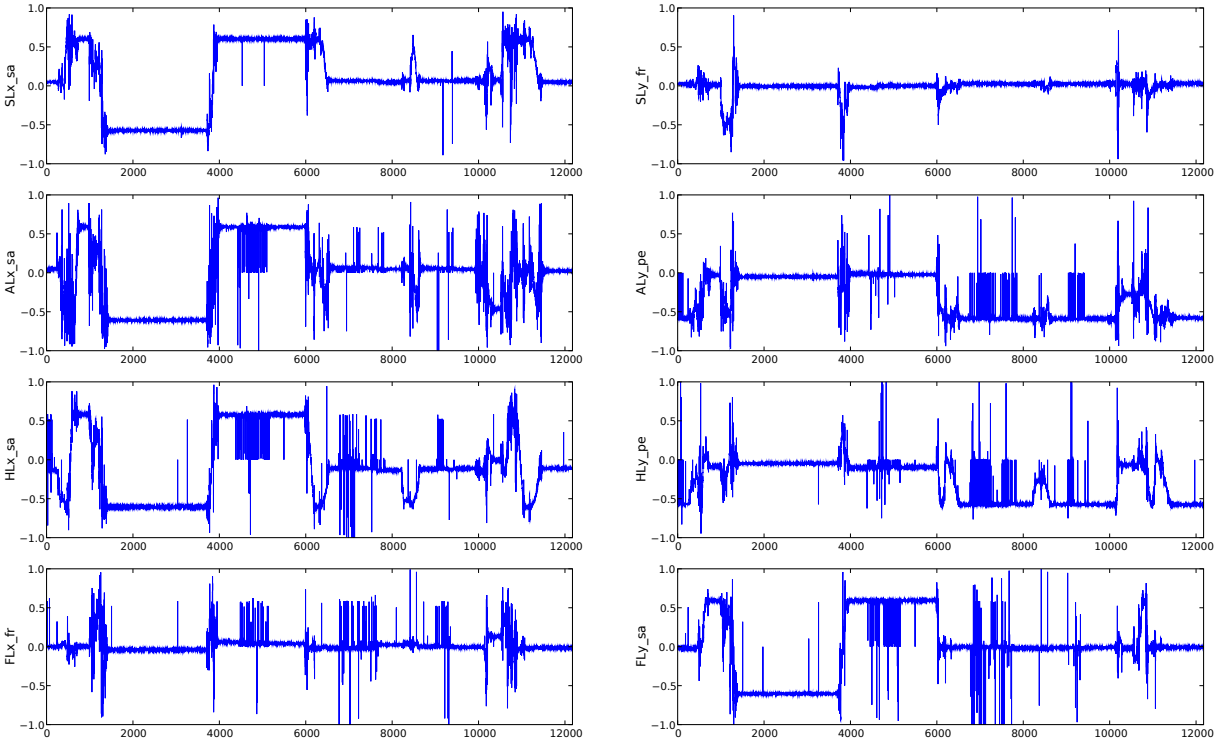


Abbildung 4.7: Beschleunigungssensorwerte der linken Körperhälfte des Roboters, entnommen der Trainingsdatensequenz.

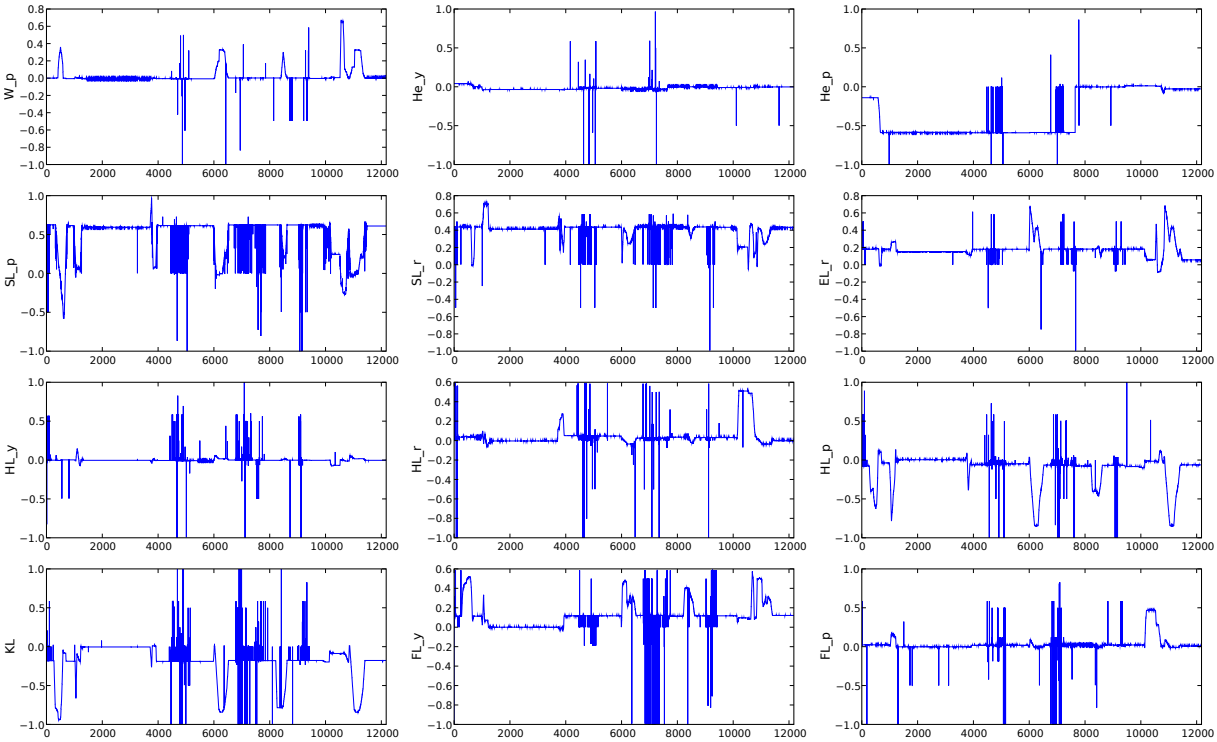


Abbildung 4.8: Motorpositionswerte der linken Körperhälfte sowie der zentralen Körperteile des Roboters, entnommen der Trainingsdatensequenz.

t	*Pose*
0-260	Stehen
750-1000	Liegen (Bauch)
1400-3700	Liegen (Rücken)
4000-6000	Liegen (Bauch)
6500-8100	Stehen
8180-8360	Hocken
8600-10000	Stehen
10150-10380	Spagat
10620-10720	Liegen (Bauch)
11300-12000	Stehen

Tabelle 4.1: Auflistung der Posen in den Trainingsdaten.

sorwerte zur Verfügung.

Abbildung 4.7 zeigt die Beschleunigungssensoren der linken Körperhälfte für die gesamte Trainingssequenz. Die beiden Körperhälften sind symmetrisch und die rechte Körperhälfte ist mit den gleichen Sensoren bestückt. Bei den Kürzeln steht der erste Buchstabe für die Position am Roboter (`S`: Schulter, `A`: Arm, `H`: Hüfte, `F`: Fuß), der zweite Buchstabe für die Körperhälfte (`L`: links, `R`: rechts). Es folgt eine Einteilung in x- oder y-Richtung, welche weiter durch das darauf folgende Buchstabenpaar aufgeschlüsselt wird: `fr` bedeutet Ausrichtung parallel zur frontalen Ebene, `pe` senkrecht zur Transversalebene und `sa` in sagittaler Richtung des Sensors. In Abbildung 1.4 auf Seite 8 sowie in Abbildung 4.9 können die genauen Ausrichtungen nachvollzogen werden.

In Abbildung 4.8 sind Motorpositionswerte aufgetragen. Dabei werden wieder symmetrisch auf beide Körperhälften verteilte Sensoren durch die der linken Körperhälfte repräsentiert. Bei den Kürzeln stehen die ersten Buchstaben für die Position am Roboter (`He`: Kopf (engl.: head), `W`: Bauch (engl.: waist), `S`: Schulter, `E`: Ellbogen, `H`: Hüfte, `K`: Knie, `F`: Fuß), danach gegebenfalls ein weiterer Buchstabe für die Körperhälfte sowie zuletzt ein Buchstabe für die Wirkrichtung: `y(aw)`, `p(itch)` und `r(oll)`.

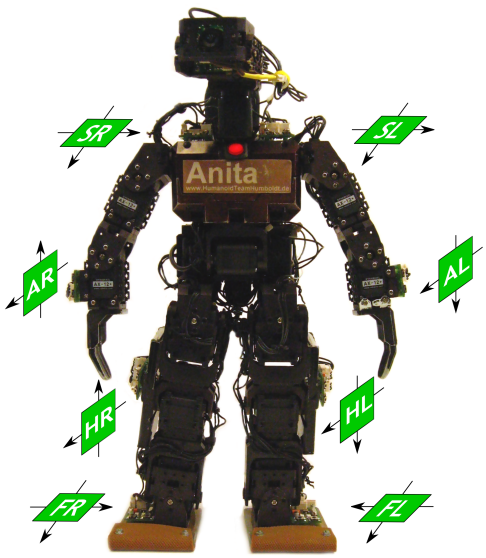


Abbildung 4.9: Schematische Darstellung der Beschleunigungssensorachsen an der A-Serie.

Tabelle 4.1 listet auf, zu welchem Zeitpunkt in der gesamten Trainingssequenz welche Pose ausgeführt wurde. Es ist deutlich zu erkennen, dass die Zeitdauer, wie lange die Posen angefahren werden, stark variiert. So dominiert die Pose *Liegen (Bauch)*, während *Liegen (Rücken)* und die sensorisch weniger stark ausschlaggebende Pose *Hocken* nur sehr kurz angefahren werden. Es wird daher eine wichtige Fragestellung sein, in-

wiefern sich diese ungleiche Verteilung auf die gelernten SFA-Komponenten auswirkt.

Es ist wichtig zu erwähnen, dass die Sensordaten leicht verrauscht sind und ebenso Übertragungsfehler nicht auszuschließen sind. So sind in der Abbildung 4.7 beispielsweise bei `ALx_sa` zu den Zeitpunkten $t = 4400 \ldots 5100$ oder bei `ALy_pe` an $t = 6700 \ldots 7900$ solche Übertragungsfehler deutlich zu erkennen.

4.1.3 Quadratische-Form-Analyse

Bevor die SFA als Verfahren zur Dimensionsreduktion betrachtet wird, soll in diesem Abschnitt eine Auswertung der SFA-Komponenten mit der Quadratische-Form-Analyse erfolgen, welche in Abschnitt 2.4.3 vorgestellt wurde. Insbesondere wird ein Vergleich zwischen der Eignung der Motorpositions- sowie der Beschleunigungssensordaten gezogen. Dazu werden eine quadratische SFA-Einheit auf den jeweiligen Sensordaten trainiert und ausgeführt sowie die optimalen Stimuli für die langsamsten Komponenten ermittelt. Die Energie für die Berechnung der optimalen Stimuli wird auf den Mittelwert der Norm der Eingangsvektoren gesetzt und beträgt bei den Beschleunigungsdaten $r = 1$ und bei den Motordaten $r = 0.95$.

Beschleunigungssensoren

Betrachten wir zunächst die langsamsten SFA-Komponenten, welche auf den Beschleunigungsdaten trainiert wurden. Dabei wird der Einfachheit halber nur eine Körperhälfte betrachtet, da sich die Sensorwerte beider Körperhälften in der betrachteten Trainingssequenz aufgrund der symmetrischen Anordnung der Sensoren stark ähneln.

In Abbildung 4.10 sind die vier langsamsten Komponenten mit den jeweiligen optimalen Stimuli zu sehen. Weniger langsame Komponenten zeigen keine hervorstechende Struktur oder Korrelation zu Sensoren und erweisen sich auch für die Dimensionsreduktion nicht als nützlich.

Zunächst fällt auf, dass die SFA-Komponenten teils hohe Ähnlichkeit zu den Sensordaten aufweisen, was auch durch die Korrelationskoeffizienten bestätigt wird: So ist y_1 stark mit den sagittalen Sensoren korreliert, am stärksten mit dem Schultersensor: $\rho(\texttt{SLx\_sa}, y_1) = 0.92$. Insbesondere unterscheidet diese Komponente deutlich, ob der Roboter auf dem Bauch oder auf dem Rücken liegt und entspricht bei Werten knapp über der Nulllinie aufrechten Posen. y_2 weist starke Ähnlichkeit zu den senkrechten Sensoren auf, beispielsweise beträgt die Korrelation zum Armsensor $\rho(\texttt{ALy\_pe}, y_2) = -0.83$. Dadurch unterscheidet y_2 deutlich zwischen Stehen und Liegen; allerdings ist auch ein deutlicher Unterschied zu erkennen, je nachdem ob der Roboter auf dem Bauch oder dem Rücken liegt, was durch die lediglich senkrechten Ebene ausgerichteten Sensoren nicht zu leisten ist. y_3 hingegen ist zu keinem der Sensorwerte stark korreliert, am stärksten ist die Korrelation zum senkrechten Hüftsensor mit $\rho(\texttt{HLy\_pe}, y_3) = -0.42$. y_4 wiederum weist mit $\rho(\texttt{SLy\_fr}, y_4) = 0.85$ eine hohe Korrelation zum frontalen Schultersensor auf.

Die optimalen Stimuli bestätigen die Korrelationen zwischen Sensordaten und langsamsten Komponenten: Am optimal exzitatorischen Stimulus für y_1 haben alle sagittalen Sensoren einen positiven Anteil, der sagittale Schultersensor ist der dominierende Wert. Die Komponente wird somit stark positiv, wenn der Roboter auf dem Bauch liegt. Es bleibt zu klären, woher die hohen

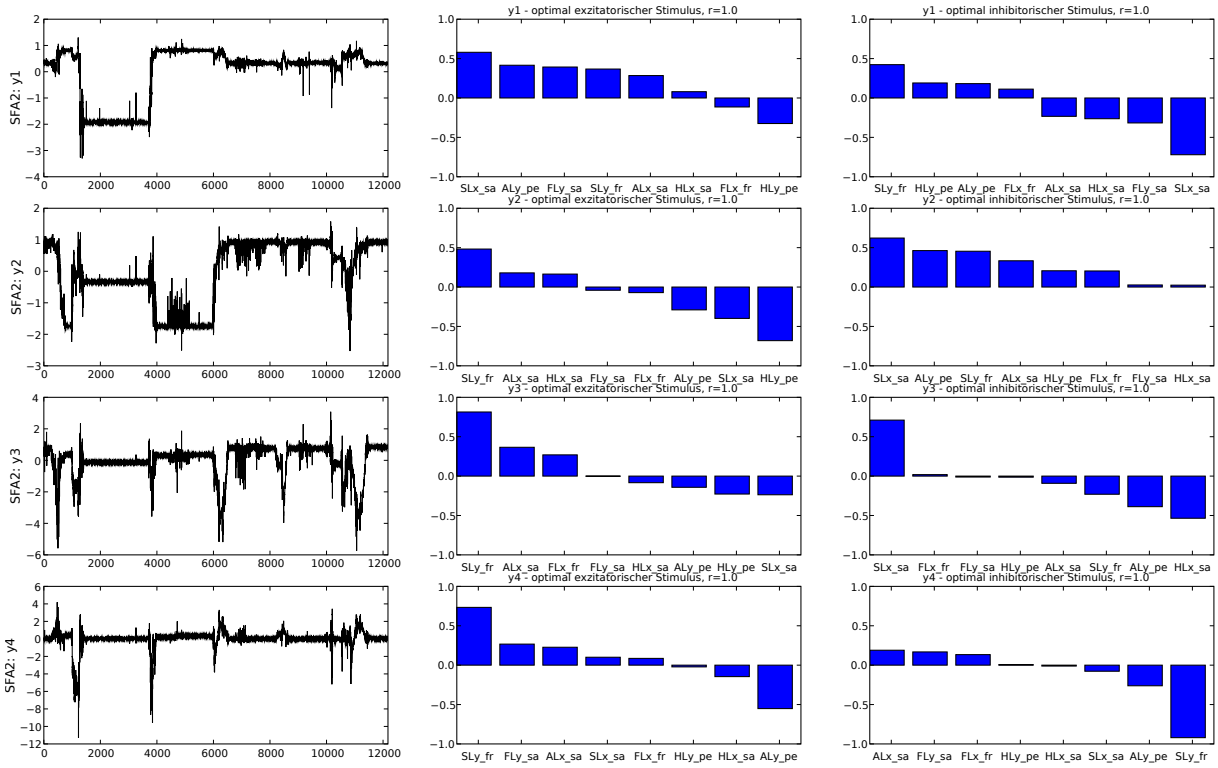


Abbildung 4.10: Die vier langsamsten Komponenten nach einer quadratischen SFA auf den Beschleunigungssensordaten. Rechts von den Komponenten sind jeweils die optimalen Stimuli zu sehen.

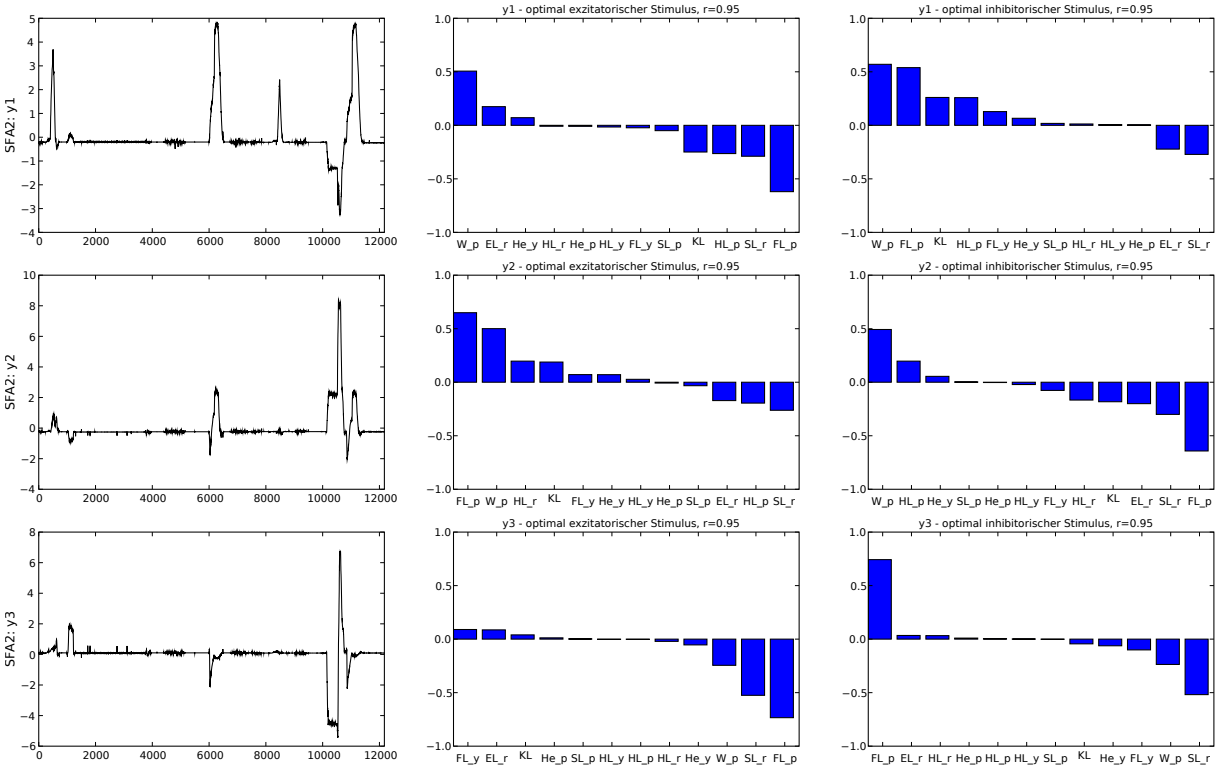


Abbildung 4.11: Die drei langsamsten Komponenten nach einer quadratischen SFA auf den Motorpositionsdaten. Rechts von den Komponenten sind jeweils die optimalen Stimuli zu sehen.

positiven bzw. negativen Anteile von `ALy_pe`, `HLy_pe` sowie `SLy_fr` rühren. Da `ALy_pe` einen hohen positiven und `HLy_pe` einen hohen negativen Anteil hat, und die Korrelation der beiden Sensoren mit $\rho(\texttt{ALy\_pe}, \texttt{HLy\_pe}) = 0.747$ relativ hoch ist, kann man davon ausgehen, dass die beiden Sensorwerte einander nahezu aufheben. Somit ist der Gesamtanteil dieser Sensorachse in der Tat relativ gering. `SLy_fr` hingegen hat einen deutlich höheren Anteil als `HLy_pe`, ebenso unterscheidet sich der Kurvenverlauf der beiden Sensoren stärker; die Korrelation der Sensoren zueinander beträgt $\rho(\texttt{SLy\_fr}, \texttt{HLx\_fr}) = -0.45$. Beide Sensoren zeigen aber über die gesamte Sequenz hinweg nur während der Bewegungen, d. h. Übergängen von einer Pose zur anderen, Aktivität. Es lässt sich sagen, dass diese Sensoren nur die dynamischen Teile der Sequenz erfassen und wahrscheinlich für die Glättung des Signals während der Übergänge genutzt werden. Der inhibitorische Stimulus für y_1 ist eindeutiger, da die sagittalen Sensoren die stärksten negativen Anteile haben. y_1 nimmt also einen hohen Wert an, wenn der Roboter auf dem Rücken liegt. Die hohe positive Anteil von `SLy_fr` dient vermutlich wieder der Glättung, während die anderen Sensoren, vor allem `ALy_pe` und `HLy_pe`, ohnehin eher niedrige Werte annehmen, wenn der Roboter auf dem Rücken liegt.

Die Tatsache, dass y_2 stark zu `ALy_pe` und `HLy_pe` negativ korreliert ist, deckt sich mit den stark negativen Anteilen dieser Sensoren im optimal exzitatorischen Stimulus von dieser Komponente. Zudem sind auch starke Anteile sagittaler Sensoren zu erkennen, welche eine Unterscheidung von Rücken- und Bauchlage ermöglichen. Wie bei y_1 erscheint die Glättung der Komponente eine plausible Erklärung für den hohen Anteil von `SLy_fr` zu sein.

y_3 weist keine hohe Korrelation zu einem der Sensorwerte auf, und auch die optimalen Stimuli lassen nicht auf eine konkrete Charakteristik dieser Komponente schließen. Zwar lassen sich die Grundposen prinzipiell unterscheiden, jedoch ist die Komponente insgesamt stark verrauscht. Es ist jedoch erkennbar, dass diese Komponente zudem auf die Posen *Hocken* und *Spagat* reagiert. Zudem lassen die hohen Anteile der frontalen Sensoren am optimal exzitatorischen Stimulus darauf schließen, dass in dieser Komponente Dynamik erfasst wird, sie also hohe Werte bei Posenwechseln annimmt. Negativ wird diese Komponente durch sagittale Werte dominiert, wobei davon auszugehen ist, dass sich `SLx_sa` und `HLx_sa` gerade aufheben.

Die Komponente y_4 wird bzgl. ihrer optimalen Stimuli deutlich von `SLy_fr` dominiert, was auch die hohe Korrelation zu diesem Sensor erklärt. Zusätzlich dazu haben die sagittalen Sensoren einen signifikanten Anteil, der sich erstaunlicherweise im Kurvenverlauf der Komponente optisch jedoch nicht widerspiegelt. Vielmehr scheint y_4 noch stärker als y_3 die Dynamik der Bewegung anstatt der statischen Posen zu erfassen. Eine mögliche Erklärung dafür ist, dass sich die sagittalen Eingangssignale mit positivem Anteil sowie die sagittalen und senkrechten Eingangssignale mit negativem Anteil gerade aufheben.

Motorpositionssensoren

Analog zur Analyse der Beschleunigungssensoren wurde eine Analyse der Motorpositionssensoren durchgeführt. Es werden wieder nur bzgl. der Körpersymmetrie nicht redundante Sensoren verwendet, jedoch ergeben sich fast identische Ergebnisse unter Verwendung aller Motorposi-

tionssensoren. In Abbildung 4.10 sind die drei langsamsten Komponenten mit den jeweiligen optimalen Stimuli zu sehen. Es wird sofort deutlich, dass die SFA-Komponenten keinerlei Rückschlüsse auf die Posen zulassen, sondern nur die dynamischen Bewegungen registrieren. Zudem sind die drei Komponenten einander allesamt sehr ähnlich. Die späteren Komponenten $y_i, i > 3$ sehen fast identisch aus und spiegeln ebensowenig die Roboterposen wider.

Die Korrelationen zu den Sensordaten sind für die SFA auf Motorpositionsdaten insgesamt niedriger als für die SFA auf Beschleunigungssensordaten. Für die SFA auf Motorpositionsdaten weist y_1 mit $\rho(\texttt{HL\_p}, y_1) = -0.81$ eine hohe Korrelation zum Pitchgelenk der Hüfte auf, y_2 ist stark mit dem Bauchgelenk korreliert $\rho(\texttt{W\_p}, y_2) = -0.73$, während y_3 mit $\rho(\texttt{FL\_p}, y_3) = -0.49$ die höchste Korrelation mit dem Pitchgelenk am Fuß aufweist. Offensichtlich handelt es sich bei allen drei korrelierten Sensoren um Pitch-Sensoren, was auch durch visuellen Vergleich der Kurven deutlich wird und die Ähnlichkeit der langsamsten Komponenten zueinander erklärt. Auch bei den optimalen Stimuli dominieren die Pitch-Winkel, die anderen Sensoren werden höchstwahrscheinlich zur Glättung des Signals herangezogen.

Positiv zu erwähnen ist die Tatsache, dass die langsamsten Komponenten kein Rauschen mehr aufweisen, was die Anwendung der SFA auf den Motordaten für andere Anwendungsfälle interessant machen könnte. Jedoch scheiden die Motorsensoren für die weitere Betrachtung zur Dimensionsreduktion und Unterscheidung von Posen aus.

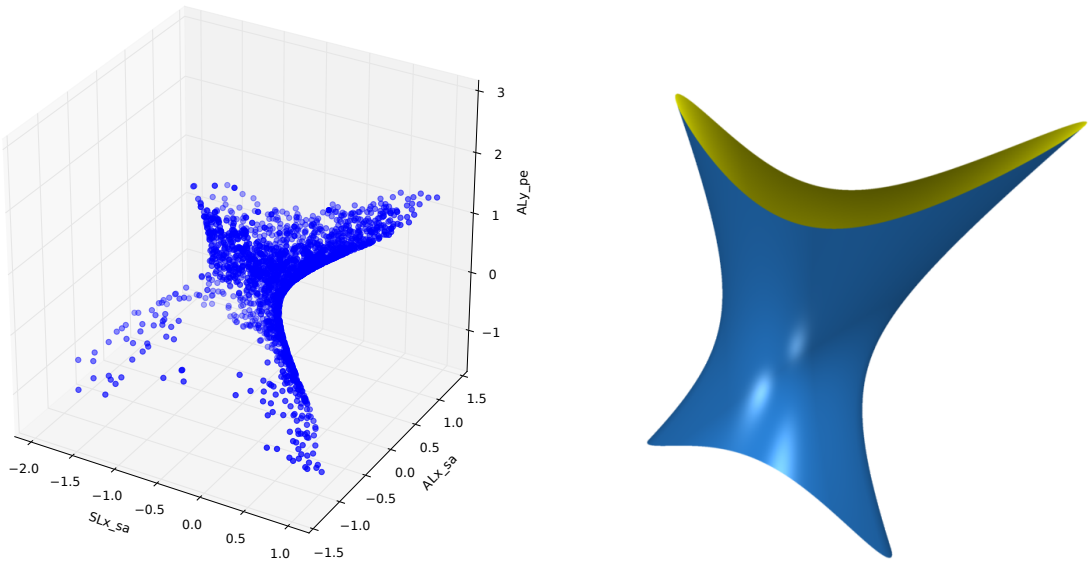
Es ist noch zu erwähnen, dass im Rahmen der Arbeit darauf verzichtet wurde, die Kombination der beiden Sensormodalitäten in der SFA zu analysieren. In einem einfachen Versuch wurde die SFA auf die kombinierten Eingabedaten angewandt, jedoch dominierten in den Ausgabekomponenten die Motorkomponenten, wodurch keine besseren Ergebnisse erzielt wurden, als wenn auf die Motorsensoren komplett verzichtet wurde. Im Ausblick werden weitere Vorschläge zur möglichen Kombination von Sensormodalitäten gemacht.

4.1.4 Analyse mittels Quadriken

Wie in Abschnitt 2.4.4 vorgestellt, können für stationäre Szenarien die Quadriken einer SFA-Komponente analysiert werden. Diese Analyse soll an dieser Stelle zusätzlich zur Quadratischen-Form-Analyse für eine der betrachteten Komponente unter Verwendung der Beschleunigungssensoren durchgeführt werden.

Da y_2 eine hohe Ähnlichkeit zu `ALy_pe` aufweist, wird betrachtet, welche Bewegungen der Roboter mit dem linken Arm ausführen kann, so dass y_2 den sensorischen Zustand weiter als *Stehen* klassifiziert. Dazu werden drei Sensoren des linken Arms als variabel gewählt (`ALx_sa`, `ALy_pe` und `SLx_sa`) und $\mu_2 = \langle y_2 \rangle_{t=9500...9900} = 0.915$ gesetzt, was der Durchschnittswert ist, während der Roboter absolut ruhig steht. Die restlichen Sensoren werden auf ihren jeweiligen Mittelwert während des gleichen Zeitraums gesetzt.

Abbildung 4.12 zeigt die für diese Sensoren resultierende Quadrik, ein einschaliges Hyperboloid. Die Quadrik wurde durch die in Abschnitt 2.4.4 vorgestellte Gradientenabstiegmethode berechnet (Abbildung 4.12a), wobei die Eingabewerte für die Sensoren $\mathbf{x}_i$ auf $[-1.5; 1.5]$ be-



(a) Gradientenabstieg (b) Analytische Berechnung

Abbildung 4.12: Quadrik der Komponente y_2 für $\mu_2 = 0.915$. Links die durch eine Gradientenabstiegsmethode berechnete Quadrik, rechts die analytisch erhaltene. Die Quadrik repräsentiert die möglichen Bewegungen des linken Arms, während der Roboter steht, und entspricht einem einschaligen Hyperboloid.

schränkt wurden. In der Tat würden für die Analyse auch Werte im Bereich ± 0.6 genügen, da dies aufgrund der Kalibrierung der Beschleunigungssensoren den maximalen bzw. minimalen Wert für statische Haltungen des Roboters darstellt. Die Erweiterung des Wertebereichs lässt jedoch deutlicher erkennen, dass der Gradientenabstieg das richtige Ergebnis liefert, welches in Abbildung 4.12b zu sehen ist.

Betrachten wir zuerst den Zusammenhang zwischen den beiden sagittalen Sensoren. Die Sensoren sind stark gekoppelt, da sie ungefähr die gleichen Werte annehmen, wenn die Morphologie des Oberkörpers nicht verändert wird, d. h. wenn der Arm angelegt am Körper bleibt. Wird `ALy_pe` fixiert, so äußert sich der Zusammenhang der beiden sagittalen Sensoren in Form einer Ellipse. Interessanterweise hat die Position des Armes in der Tat eine Auswirkung auf y_2: Wird der Arm durch Drehung des Pitchgelenks nach vorne bewegt, so muss der Roboter seinen Oberkörper nach hinten lehnen, um auf der Quadrik zu bleiben. Dagegen hat eine Bewegung des Armes nach hinten keine Auswirkung.

Es ist zu vermuten, dass die Abhängigkeit vom sagittalen Armsensoren eliminiert werden kann, indem die Trainingsdaten entsprechende sensorische Situationen aufweisen.

Um zu sehen, inwiefern die Bewegung des Armes parallel zum Körper nach oben Einfluss auf die Quadrik hat, werden Paare aus je einem sagittalen und dem senkrechten Sensor betrachtet. Die Relation der beiden Sensoren entspricht geometrisch jeweils einer Hyperbel. Jedoch sind die Sensorpaare nicht so stark gekoppelt wie die sagittalen Sensoren, sondern erlauben eine freie Bewegung des Arms ohne starke Veränderung von y_2. Insbesondere sind die möglichen Konfigurationen der beiden Armsensoren `ALx_sa` und `ALy_pe` größtenteils nicht relevant, da sie aufgrund der Morphologie des Roboters nicht auftreten können: So sind für `ALx_sa` beliebige Werte er-

laubt, wenn der Arm vollständig ausgestreckt, also waagerecht zum Erdboden ist (`ALy_pe` ≈ 0.6). Jedoch kann der Arm in dieser Haltung nicht mehr so bewegt werden, so dass der sagittale Sensorwert einen Wert verschieden von Null einnimmt.

4.2 Dimensionsreduktion zur sensorischen Kartographierung

Nachdem in den letzten Abschnitten der Schwerpunkt darauf lag, zu verstehen und zu erklären, welche Ergebnisse die SFA angewandt auf verschiedene sensorische Qualitäten liefert, soll nun der Bezug zu einer konkreten Anwendung hergestellt werden.

Betrachten wir jeden der verfügbaren Sensorwerte als Komponente eines Vektors, so lässt sich der durch den Roboter wahrnehmbare *Zustandsraum* als n-dimensionaler Vektorraum beschreiben, wobei n der Anzahl der Sensorwerte entspricht. Eine kompaktere Darstellung des Zustandsraum kann man nun durch eine so genannte *Dimensionsreduktion* des Zustandsraumes erreichen. Die Idee besteht darin, den n-dimensionalen sensorischen Zustandsraum in einen m-dimensionalen Zustandsraum mit $m \ll n$ zu überführen, wobei im neuen Zustandsraum bestimmte Kriterien des Ursprungsraums erhalten bleiben. Da eine kompakte Darstellung des vom Roboter wahrnehmbaren Zustandsraumes gesucht ist, bestehen diese Kriterien in der weitgehenden Erhaltung sowohl lokaler als auch globaler Nachbarschaftsverhältnisse sowie der Unterscheidbarkeit relevanter Zustände.

Man unterscheidet grundsätzlich zwei Verfahren zur Dimensionsreduktion: Die *Merkmalsselektion* und die *Merkmalsextraktion*. Bei der Merkmalsselektion werden bestimmte Dimensionen der Eingabedaten ausgewählt, während bei der Merkmalsextraktion eine lineare oder nichtlineare Transformation der Eingabedaten erfolgt. Die Struktur dieser Verfahren kann, muss aber nicht zwingend dem in der Einleitung vorgestellten allgemeinen Schema für Verfahren zur Merkmalsextraktion folgen.

Es sind zum Zeitpunkt dieser Arbeit bereits diverse Dimensionsreduktionstechniken bekannt, welche jeweils verschiedene Vor- und Nachteile haben[2]. In der Diplomarbeit von André Stephan wurden mehrere merkmalsextrahierende Verfahren betrachtet und auf Daten der A-Serie-Roboter angewandt und einander gegenübergestellt [Ste10]. Im folgenden wird die SFA als eine weitere Möglichkeit zur Merkmalsextraktion herangezogen und mit den Ergebnissen aus der Arbeit von André Stephan verglichen. Zudem liegt der Fokus auf der einfachen visuellen Auswertung der gewonnenen Dimensionsreduktion, weswegen nur zweidimensionale Reduktionen in Betracht gezogen werden[3]. Die entstehenden zweidimensionalen Reduktionen werden auch als *sensorische Karten* bezeichnet.

Wie bereits erwähnt, ist eine Anforderung an ein gutes Dimensionsreduktionsverfahren, dass sowohl lokale als auch globale Nachbarschaften nach der Reduktion erhalten bleiben. Identifizieren wir die Zustände mit Haltungen und Posen des Roboters, so ist es beispielsweise wünschenswert, dass ähnlichere Posen wie *Hocken* und *Stehen*, welche sich beide dadurch auszeichnen, dass

[2] Für einen detaillierten Übersichtsartikel über gängige Methoden siehe beispielsweise [Fod02].

[3] Zwar hätten auch dreidimensionale Dimensionsreduktionen ausgewertet werden können, jedoch wurde darauf wegen besserer Vergleichbarkeit zu den Ergebnissen aus [Ste10] verzichtet.

der Roboter aufgerichtet ist, näher beieinander sind als *Stehen* und *Liegen*. Selbstverständlich muss diese Ähnlichkeit aber auch schon im ursprünglichen Zustandsraum gegeben sein, was im konkreten Beispiel der Beschleunigungssensordaten der Fall ist. Ebenso sollten ähnliche Posen, die in den Ursprungsdaten unterschieden werden können, auch nach der Dimensionsreduktion hinreichend gut unterscheidbar bleiben.

4.2.1 Methodik und Gütekriterien

Für die Anwendung der SFA werden verschiedene Parameter und Konfigurationen evaluiert. Zum einen wird die Anzahl der nacheinander ausgeführten SFA-Einheiten von eins bis fünf variiert; bei mehr Iterationen findet eine zu starke Überanpassung statt. Zum anderen werden immer zwei verschiedene langsame Komponenten ausgewählt, um den dimensionsreduzierten Raum aufzuspannen. Dabei werden jeweils alle Paare aus den zehn langsamsten Komponenten betrachtet. Im Gegensatz zur Vorgehensweise in [Ste10] wird keinerlei zusätzliche Filterung der Eingangsdaten vorgenommen, da die SFA dies bereits von selbst leistet. Im Gegensatz zur bisherigen Analyse werden im folgenden die Sensorwerte beider Körperhälften verwendet. Die Anzahl der langsamsten Komponenten pro SFA-Einheit wird auf 48 fixiert, zudem wird für alle Komponenten jeder SFA-Einheit ein Clipping auf ± 2 vorgenommen. Bei späteren SFA-Iterationen führen sonst sehr starke Überschwinger dazu, dass die Komponenten inadäquat große Wertebereiche aufweisen. Aufgrund der Ergebnisse der Quadratische-Form-Analyse wird die SFA nur auf Beschleunigungssensoren angewandt.

Zur Vereinfachung werden für den Ergebnisteil folgende Schreibweisen verwendet: Die SFA-Komponente y_n, die nach der k-ten SFA-Einheit gewonnen wird, bezeichne ich als $y_n[k]$. Um mehrere Komponenten ab einem bestimmten k zusammenzufassen, schreibe ich auch $y_n[k \geq l]$ für eine Startiteration l. Eine zweidimensionale Reduktion, bestehend aus den SFA-Komponenten $y_n[k]$ und $y_m[k]$ der k-ten SFA-Einheit, wird als $Y_{n,m}[k]$ bezeichnet. Zum Beispiel wird also die Reduktion, die man durch die langsamsten beiden Komponenten nach einer einzigen SFA-Iteration erhält, mit $Y_{1,2}[1]$ bezeichnet.

Silhouette

Da a priori nicht klar ist, nach welcher SFA-Einheit die besten Ergebnisse erzielt werden und welche zwei Komponenten der betrachteten SFA-Einheit bei der zweidimensionalen Dimensionsreduktion die verschiedenen Posen am besten trennen, wird zur Evaluation das *Silhouettemaß* aus [Rou87] verwendet. Die Silhouette wird normalerweise als Maß für die Qualität eines Clusterings verwendet, erweist sich jedoch als hilfreich, um zu ermitteln, wie gut unterscheidbar verschiedene Posen nach einer Dimensionsreduktion bleiben. Nehmen wir an, jeder Wert oder, geometrisch betrachtet, Punkt $\mathbf{x}_t := \mathbf{x}(t)$ für ein $t \in \{1, \ldots, T\}$ der Eingabedaten ist genau einer von k Klassen bzw. Clustern $C_i, i = 1, \ldots, k$ zugeordnet. Bezeichne f_C die Funktion, die jedem Punkt $\mathbf{x}_t$ seinen Cluster C zuordnet. Seien nun $a(\mathbf{x}_t)$ die durchschnittliche Distanz von $\mathbf{x}_t$ zu allen anderen Punkten in dem Cluster $f_C(\mathbf{x}_t)$ sowie $b(\mathbf{x}_t)$ die Distanz des zu $\mathbf{x}_t$ nächsten Clusters, dem $\mathbf{x}_t$ nicht

zugehört. Mathematisch formuliert heißt das:

$$a(\mathbf{x}_t) = \text{dist}(\mathbf{x}_t, f_C(\mathbf{x}_t)) \tag{4.2}$$

$$b(\mathbf{x}_t) = \min_{1 \leq i \leq k} \{\text{dist}(\mathbf{x}_t, C_i)\}, \qquad C_i \neq f_C(\mathbf{x}_t). \tag{4.3}$$

$\text{dist}(\mathbf{x}_t, C_i)$ bezeichnet dabei die euklidische Distanz zwischen einem Punkt $\mathbf{x}_t$ und dem Mittelpunkt des Clusters C_i; in diesem Fall wird die euklidische Distanz verwendet. Intuitiv nimmt $a(\mathbf{x}_t)$ einen hohen Wert an, wenn der Punkt $\mathbf{x}_t$ weit weg vom Clustermittelpunkt liegt, was an einer deformierten Clusterform oder einer schlechten Zuordnung von $\mathbf{x}_t$ liegen kann. Nimmt $b(\mathbf{x}_t)$ hingegen einen hohen Wert an, ist dies ein Zeichen dafür, dass $f_C(\mathbf{x}_t)$ eine gute Zuordnung für $\mathbf{x}_t$ ist, bzw. dass $\mathbf{x}_t$ sich nicht in der unmittelbaren Nähe eines anderen Clusters befindet. Die Silhouette eines Wertes bzw. eines Clusterings sind schließlich definiert[4] als:

$$s(\mathbf{x}_t) = \frac{b(\mathbf{x}_t) - a(\mathbf{x}_t)}{\max\{a(\mathbf{x}_t), b(\mathbf{x}_t)\}} \tag{4.4}$$

$$S(\mathbf{X}) = \frac{1}{T} \sum_{t=1}^{T} s(\mathbf{x}_t), \tag{4.5}$$

mit $\mathbf{X} := (\mathbf{x}_1, \ldots, \mathbf{x}_T)^T$, d. h. die Matrix, welche zeilenweise die Werte $\mathbf{x}_t$ enthält. Aus Definition 4.4 ist unmittelbar ersichtlich, dass für alle $\mathbf{x}_t$ gilt $-1 \leq s(\mathbf{x}_t) \leq 1$. Damit $s(\mathbf{x}_t)$ nahe 1 liegt, muss $a(\mathbf{x}_t)$ sehr viel kleiner als $b(\mathbf{x}_t)$ sein. Da a ein Maß für die Distanz eines Punktes von seinem eigenen Cluster ist, bedeutet also $s(\mathbf{x}_t) \approx 1$, dass die Zuordnung von $\mathbf{x}_t$ zu seinem Cluster gut ist. Mit einer ähnlichen Argumentation erkennt man, dass ein Wert von $s(\mathbf{x}_t)$ nahe -1 bedeutet, dass der Punkt $\mathbf{x}_t$ besser seinem Nachbarcluster zugeordnet werden sollte. $s(\mathbf{x}_t) \approx 0$ schließlich bedeutet, dass der Punkt sich genau zwischen zwei Clustern befindet. Um eine Aussage über die Silhouette einer ganze Punktmenge zu treffen, wird Formel 4.5 verwendet, welche die Silhouetten der einzelnen Punkte aufsummiert und wieder auf den Wertebereich $[-1; 1]$ normiert.

Um mit der Silhouette eine qualitative Aussage über die Wahl der Komponenten und der SFA-Einheit zu treffen, wurde zunächst in den Originaldaten jeder Datenpunkt manuell genau einer Pose zugeordnet oder als Zwischenzustand markiert. Formal bedeutet das, dass jede Pose jeweils einem Cluster entspricht; als Zwischenzustand markierte Punkte werden ignoriert, da für diese das Silhouettemaß aufgrund der Tatsache ungeeignet ist, dass für den Zwischenzustand in der Regel kein repräsentativer Mittelwert gefunden werden kann. Dann wird auf dem dimensionsreduzierten Raum die Silhouette gemäß Formel 4.5 berechnet. Die Annahme ist, dass ein Silhouettewert nahe 1 darauf hindeutet, dass die verschiedenen Posen nach der Dimensionsreduktion gut voneinander unterscheidbar sind.

Bewertung von Formerhaltung und Nachbarschaftstreue

Um auch eine Aussage über zuvor nicht manuell zugeordnete Zustände treffen zu können, werden zum einen die Trajektorien betrachtet, die beim Übergang von einer Pose in eine andere entstehen. Zum anderen wird die *Procrustes*-Analyse [LdVC95] angewandt, welche auch in [Ste10]

[4] Da im ursprünglichen Anwendungsfall die Güte eines Clustering bestimmt werden soll, wird bei Gleichung 4.5 normalerweise die Clusterzuordnung als Argument verwendet, d. h. $S(f_C)$.

zum Einsatz kam, um die Erhaltung der Distanzinformationen nach der Transformation durch ein Dimensionsreduktionsverfahren zu bewerten.

Bei der Procrustes-Analyse werden zwei ein- oder mehrdimensionale Punktmengen $\mathbf{X}$ und $\mathbf{Y}$ bestmöglich deckungsgleich übereinander gelegt, um die Ähnlichkeit dieser Punktmengen besser zu vergleichen. Die Idee ist dabei, Translations-, Rotations- und Skalierungseffekte zu entfernen, welche zwar die lokale und globale Form der Punktmengen nicht beeinflussen, aber ihre Vergleichbarkeit mittels einer einfachen euklidischen Norm beeinträchtigen. Um die Procrustes-Analyse durchzuführen, werden dazu $\mathbf{X}$ und $\mathbf{Y}$ zunächst auf die gleiche Dimension gebracht, indem die Punktmenge mit niedrigerer Dimension in den zusätzlichen Dimensionen mit Nullen aufgefüllt wird. Als nächstes werden beide Punktmengen mittelwertzentriert und auf eine Varianz von 1 gebracht, womit Translations- und Skalierungseffekte beseitigt sind. Bezeichnen also von nun an $\tilde{\mathbf{X}}$ und $\tilde{\mathbf{Y}}_R$ die ursprünglichen Punktmengen, allerdings auf die gleichen Dimension gebracht sowie normalisiert, also beide mit Mittelwert Null sowie Varianz Eins. Als letztes ist nun eine geeignete Rotation von $\tilde{\mathbf{Y}}_R$ bzgl. $\tilde{\mathbf{X}}$ gesucht. Dazu kann beispielsweise die Singulärwertzerlegung verwendet werden:

$$\tilde{\mathbf{X}}^T\tilde{\mathbf{Y}}_R = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T. \tag{4.6}$$

So ergeben sich durch folgende Gleichungen die orthogonale Rotationsmatrix, welche $\tilde{\mathbf{Y}}_R$ relativ zu $\tilde{\mathbf{X}}$ rotiert, sowie nach der Rotation angepassten Daten $\tilde{\mathbf{Y}}$:

$$\mathbf{Q} := \mathbf{V}\mathbf{U}^T, \tag{4.7}$$

$$\tilde{\mathbf{Y}} := \tilde{\mathbf{Y}}_R\mathbf{Q}. \tag{4.8}$$

Die Procrustes-Analyse liefert dann durch Berechnung der euklidischen Norm auf den bereinigten Daten ein Dissimilaritätsmaß, d. h. ein niedrigerer Wert deutet auf eine höhere Ähnlichkeit von $\mathbf{X}$ und $\mathbf{Y}$ hin:

$$PROC(\mathbf{X}, \mathbf{Y}) := \tilde{PROC}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}}) = \sum_{i=1}^{n} ||\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i||^2 \tag{4.9}$$

Im folgenden werden die ursprünglichen Beschleunigungssensordaten mit einem SFA-Komponentenpaar verglichen, so dass folgende abkürzende Schreibweise verwendet wird:

$$PROC(Y_{n,m}[k]) := PROC(\mathbf{X}, Y_{n,m}[k]), \tag{4.10}$$

wobei wie erwähnt $\mathbf{X} := (\mathbf{x}_1, \ldots, \mathbf{x}_T)^T$ und $Y_{n,m}[k]$ normalisiert sind und $Y_{n,m}[k]$ bzgl. $\mathbf{X}$ mit Nullen aufgefüllt sowie rotiert wurde.

4.2.2 Ergebnisse

Betrachten wir zunächst die Kurvenverläufe der langsamsten Komponenten nach verschiedenen SFA-Iterationen. Die Komponenten nach einer SFA-Iteration wurden bereits in Abschnitt 4.1.3 untersucht. Abbildungen 4.13, 4.14 und 4.15 zeigen die Kurvenverläufe nach zwei, drei bzw. fünf Iterationen mit zusätzlichem Clipping bei $[-2.0; 2.0]$. Zwischen der dritten und fünften Iteration

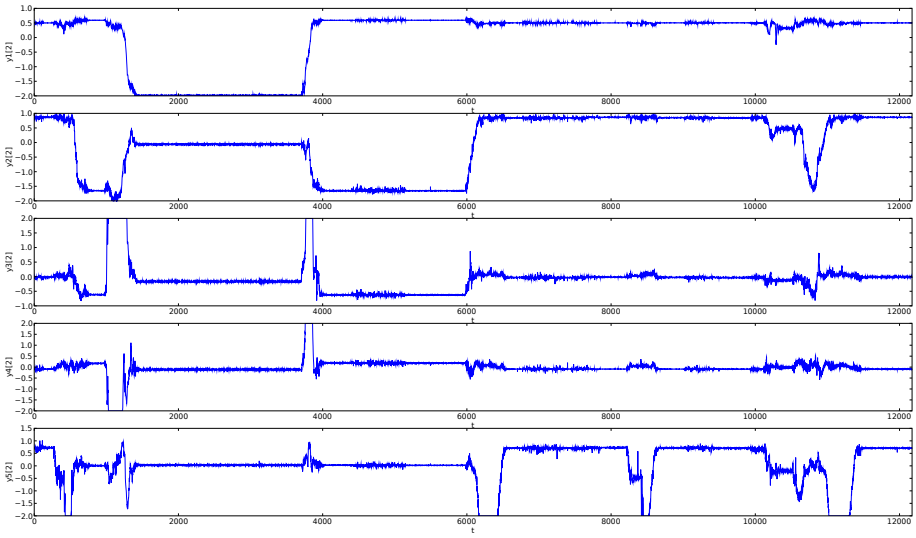


Abbildung 4.13: Die fünf langsamsten Komponenten nach zwei Iterationen.

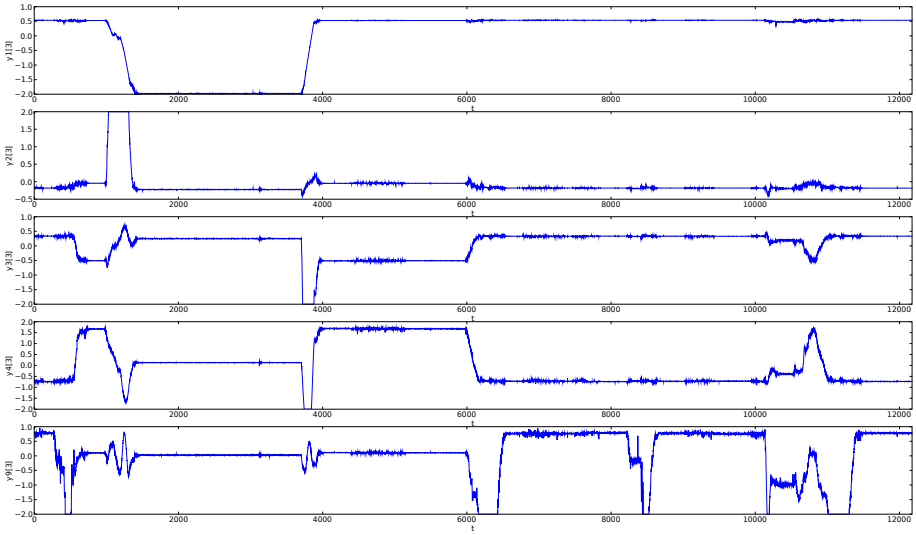


Abbildung 4.14: Ausgewählte langsamste Komponenten nach drei Iterationen.

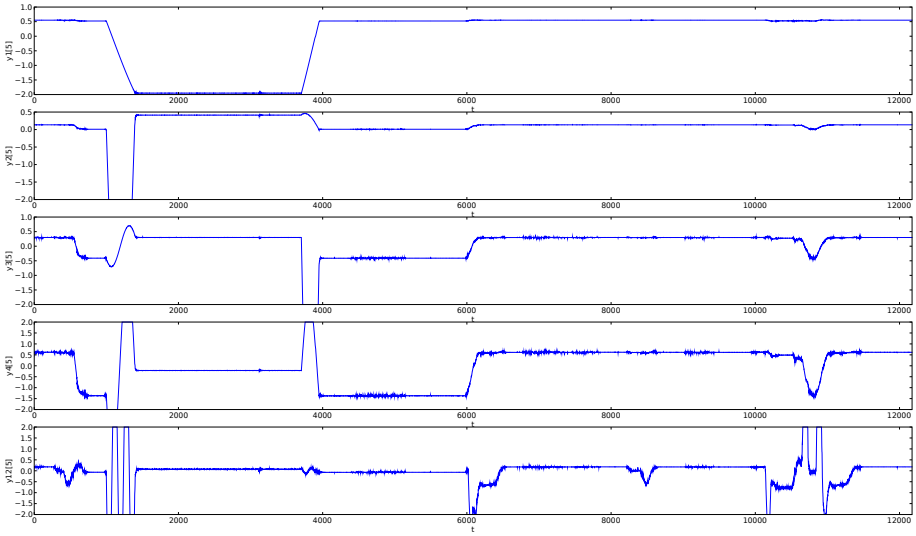


Abbildung 4.15: Ausgewählte langsamste Komponenten nach fünf Iterationen.

verändert sich von der Charakteristik her kaum mehr etwas an den Komponenten. Ungünstig ist, dass ab der dritten Iteration in fast allen Komponenten außer y_1 starke Überschwinger bei einigen Posenwechseln entstehen, welche einen unnatürlichen Kurvenverlauf zur Folge haben. Solche Überschwinger lassen sich mit der Analyse optimaler SFA-Komponenten erklären und deuten wie in Abschnitt 2.3.2 erwähnt auf eine Überanpassung an die Trainingsdaten hin. Daher wird auch auf eine Betrachtung von mehr als fünf SFA-Iterationen verzichtet.

y_1 ist eine rein binäre Komponente, die bei der Pose *Liegen (Rücken)* einen Wert von ungefähr -2.0 annimmt und sonst nahe 0.5 liegt. Sie ist durchweg am stärksten zum linken sagittalen Schultersensor `SLx_sa` korreliert. Dieser Wert sinkt mit mehr Iterationen nur leicht, für die fünfte Iteration beträgt er immer noch $\rho(\texttt{SLx\_sa}, y_1[5]) = 0.81$. Die grundlegende Charakteristik der Komponente verändert sich über alle SFA-Iterationen nicht, was durch die Korrelation der langsamsten Komponenten zueinander bestätigt wird: $\rho(y_1[1], y_1[5])) = 0.96$.

$y_2[1]$ ist deutlich zum senkrechten Sensor `ALy_pe` korreliert, und auch $y_2[2]$ weist eine hohe Korrelation zu den senkrechten Armsensoren auf ($\rho(\texttt{ARy\_pe}, y_2[2]) = -0.76$). Sie unterscheidet deutlich die Posen *Stehen*, *Liegen (Rücken)* und *Liegen (Bauch)*. Ab der dritten Iteration ähnelt die Komponente eher den frontalen Schultersensoren, allerdings mit recht niedrigen Korrelationen von $\rho(\texttt{SRy\_fr}, y_2[3]) = -0.62$, $\rho(\texttt{SRy\_fr}, y_2[4]) = 0.53$ und $\rho(\texttt{SRy\_fr}, y_2[5]) = 0.51$.

$y_3[2]$ weist Ähnlichkeit zu $y_2[k \geq 3]$ auf und ist ebenfalls eine interessante binäre Komponente, die aufrechten Stand (*Stehen* und *Hocken*) sowie *Liegen (Bauch)* unterscheidet. In der zweiten Iteration ähnelt diese Komponente mit $\rho(\texttt{SLy\_fr}, y_3[2]) = -0.73$ dem frontalen Schultersensor. Die Komponenten $y_2[k \geq 3]$ weisen ebenso eine – wenn auch nicht so starke – Korrelation zu `SLy_fr` auf. Ab der dritten Iteration verändert sich die drittlangsamste Komponente, die Komponenten $y_3[k \geq 3]$ sind ternäre Signale, welche aufrechten Stand (*Stehen* und *Hocken*), *Liegen (Rücken)* und *Liegen (Bauch)* unterscheiden.

Die viertlangsamste Komponente y_4 weist nur nach der ersten Iteration eine sehr hohe Korrelation zu den frontalen Sensoren auf, so ist $\rho(\texttt{SLy\_fr}, y_4[1]) = -0.88$. Sie ähnelt in der zweiten Iteration $y_3[2]$. Ab der dritten Iteration übernimmt y_4 die Funktion von $y_2[2]$ ($\rho(y_2[1], y_4[3]) = 0.91$, $\rho(y_2[1], y_4[5]) = -0.76$), d. h. sie detektiert die Posen *Stehen*, *Liegen (Rücken)* und *Liegen (Bauch)*. Die Komponenten $y_5[k \geq 3]$ (nicht abgebildet) sind fast identische Kopien von $y_4[k \geq 3]$, lediglich mit umgedrehtem Vorzeichen.

Ähneln die Komponenten $y_5[k \geq 3]$ (nicht abgebildet) ab der dritten SFA-Iteration deutlich ihren Vorgängerkomponenten $y_4[k \geq 3]$, weist $y_5[2]$ wegen starker Überschwinger keine signifikante Korrelation zu Sensoren oder späteren Komponenten auf. Allerdings ist sie die erste Komponente dieser Iteration, in der wie bei $y_2[1]$ eine Detektion der Posen *Hocke* und *Spagat* zu erkennen ist.

Die späteren Komponenten $y_6[2]$ bis $y_{10}[2]$ (nicht abgebildet) weisen zu keinem Sensor eine höhere Korrelation als 0.3 auf, ähneln aber alle der frontalen Schulterkomponente. Dass die Korrelation nicht hoch ausfällt, liegt daran, dass bei den Posenübergängen keine einzelnen Peaks wie im Schultersensor, sondern Überschwinger mit verschiedener Frequenz auftreten.

Bzgl. weniger langsamer Komponenten ist ab der dritten SFA-Iteration $y_9[3]$ die einzig inter-

essante Komponente. Sie weist das Verhalten von $y_5[2]$ auf und erkennt dynamische Übergänge sowie Hocke und Spagat. Sie taucht für $k \geq 4$ erst wieder in höheren Komponenten, nämlich in $y_{13}[4]$ und $y_{12}[5]$ auf.

Nach dieser ausführlichen Betrachtung der langsamsten Komponenten lässt sich bereits voraussagen, welche Komponentenpaare eine gute Dimensionsreduktion ergeben. Die eine Komponente sollte $y_2[1]$, $y_3[2]$ oder $y_4[k \geq 3]$ sein, um *Stehen*, *Liegen (Rücken)* und *Liegen (Bauch)* zu unterscheiden. Die andere Komponente muss zusätzlich die Posen *Hocken* und *Spagat* erkennen; dies wird unter den zehn langsamsten jeweils nur durch die Komponenten $y_3[1]$, $y_5[2]$ und $y_9[3]$ erfüllt.

Silhouette und Procrustes

Werfen wir nun einen Blick auf die Silhouette- und Procrusteswerte der SFA-Komponentenpaare. Konturplots, welche den Einfluss der Parameter auf die Gütemaße verdeutlichen, sind in den Abbildungen 4.16 und 4.17 zu sehen. Sie verdeutlichen die Werte der Gütemaße für alle möglichen Paare der zehn langsamsten Komponenten für eine bis fünf SFA-Iterationen. Betrachten wir zunächst die Silhouette, so ergeben sich für die verschiedenen Kombinationen von SFA-Komponenten und -Einheiten ein Höchstwert von $S(Y_{4,9}[2]) = 0.725$ sowie ein Tiefstwert von $S(Y_{5,9}[1]) = 0.001$; dieser Tiefstwert stellt mit ein paar anderen Werten aus der ersten SFA-Iteration allerdings Ausreißer dar, da die Silhouettewerte im Schnitt bei 0.62 (mit Varianz 0.01) liegen. Abbildung 4.16 zeigt die Silhouettewerte verschiedener SFA-Komponentenpaare und SFA-Iterationen. Die weißen Stellen rechts unten in der Grafik bedeuten, dass der Silhouettewert unter 0.42 liegt und daher nicht in die Farbskala mitaufgenommen wurden. Dass bei mehr SFA-Iterationen generell bessere Silhouettewerte auftreten, liegt wie zuvor erwähnt vor allem daran, dass sich bei mehr Iterationen die weniger langsamen Komponenten mehr ähneln. Die besten Silhouettewerte je Iteration sind:

$$\begin{aligned} S(Y_{2,3}[1]) &= 0.67 \\ S(Y_{2,5}[2]) &= 0.714 \\ S(Y_{4,9}[3]) &= 0.725 \\ S(Y_{5,9}[4]) &= 0.722 \\ S(Y_{5,7}[5]) &= 0.718 \end{aligned}$$

Wie erwartet befinden sich $S(Y_{2,5}[2])$ und $S(Y_{4,9}[3])$ unter den besten Werten, da sie sowohl die Lage (*Stehen*, *Liegen*) als auch die Posen *Hocken* und *Spagat* eindeutig voneinander trennen.

Betrachten die Verteilung Procrusteswerte der in Abbildung 4.17, so ist klar erkennbar, dass weniger langsame Komponenten eine immer schlechtere Repräsentation der Originaldaten darstellen. Der niedrigste und somit beste Wert ergibt sich mit $PROC(Y_{1,2}[1]) = 0.31$, während der schlechteste Wert bei $PROC(Y_{6,8}[3]) = 0.99$ liegt. $Y_{1,2}[1]$ hat allerdings nur einen Silhouettewert von $S(Y_{1,2}[1]) = 0.65$, da die Posen *Hocken* und *Stehen* zusammenfallen. Wie man in 4.17 sieht,

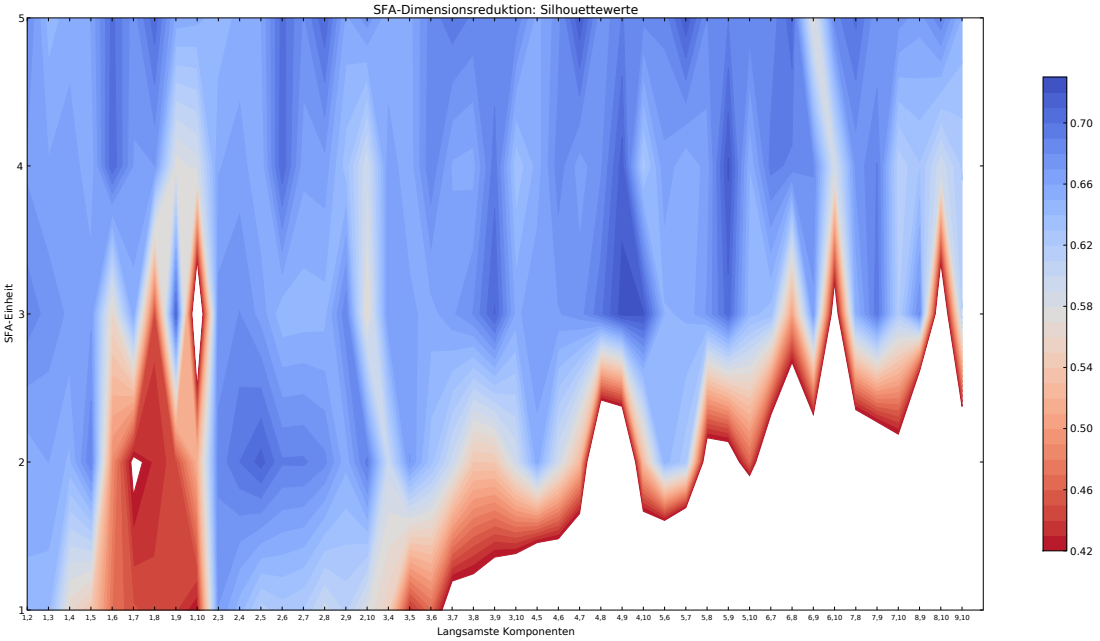


Abbildung 4.16: Silhouettewerte in Abhängigkeit von ausgewählten SFA-Komponenten und -Iterationen. Je *größer* der Wert, desto besser sind die Posen unterscheidbar.

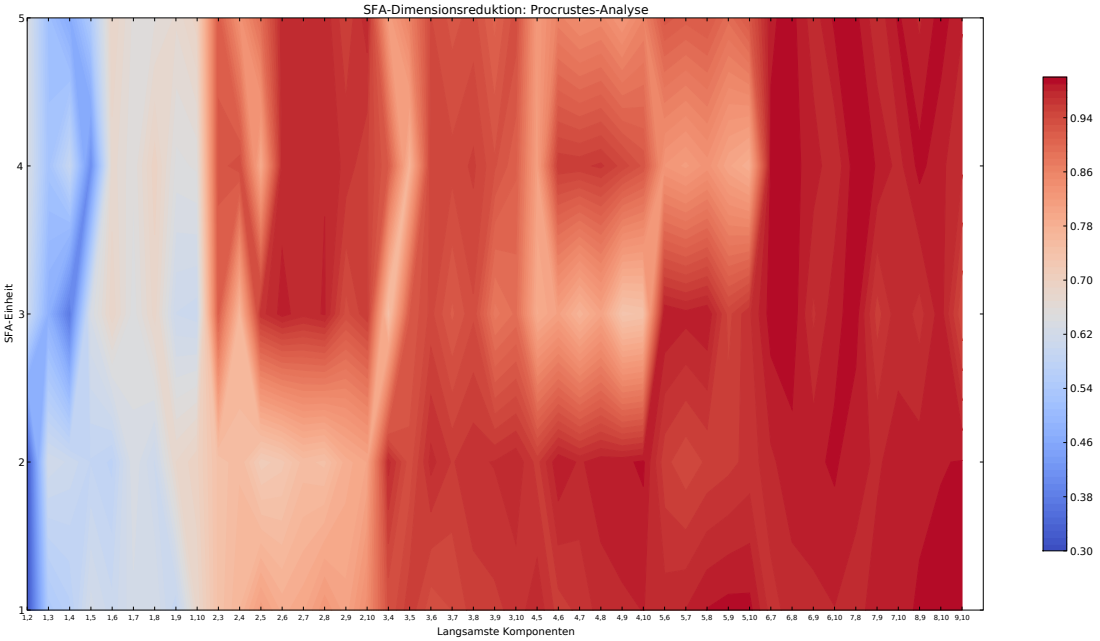


Abbildung 4.17: Procrusteswerte in Abhängigkeit von ausgewählten SFA-Komponenten und SFA-Iterationen. Hier deutet ein *kleinerer* Wert auf eine größere Ähnlichkeit mit den Originaldaten hin.

erreicht tatsächlich die Mehrheit der Reduktionen einen äußerst schlechten Wert. Tatsächlich sind bzgl. des Procrusteswerts alle Paare, die die jeweils langsamste Komponente $y_1[k]$ enthalten, besser als solche Paare, die die langsamste Komponente nicht enthalten. Es ist allerdings nicht überraschend, dass weniger langsame Komponenten schlechte Ergebnisse aufweisen, da sie immer weniger zu den Sensoren korreliert sind. Ebenso haben höhere Komponenten stärker oszillierende Überschwinger und bilden daher zu den Eingabedaten unähnliche Kurvenverläufe aus. Die Procrusteswerte für die fünf Dimensionsreduktionen mit den besten Silhouettewerten betragen:

$$\begin{aligned} PROC(Y_{2,3}[1]) &= 0.74 \\ PROC(Y_{2,5}[2]) &= 0.71 \\ PROC(Y_{4,9}[3]) &= 0.74 \\ PROC(Y_{5,9}[4]) &= 0.80 \\ PROC(Y_{5,7}[5]) &= 0.91 \end{aligned}$$

Insgesamt stellt die Dimensionsreduktion $Y_{2,5}[2]$ also das beste Ergebnis dar, obwohl sie einen allenfalls mittelmäßigen Procrusteswert aufweist. Sie stellt dennoch einen guten Kompromiss zwischen Glättung durch die SFA und Ähnlichkeit zu den Sensoren dar, wobei sie vor allem keinerlei Überschwinger bei den Posenübergängen hat. Abbildung 4.18 zeigt den durch $Y_{2,5}[2]$ reduzierten sensorischen Raum unter Hervorhebung von Posen und Übergangstrajektorien. Die Übergangstrajektorien stellen dar, welche sensorischen Zustände im reduzierten Raum erreicht werden, wenn der Übergang von einer Pose zur nächsten erfolgt. Um die Darstellung übersichtlicher zu halten, wurde nur jeder fünfte Punkt einer Trajektorie verbunden. Es ist gut zu erkennen, dass alle Posen klar voneinander getrennt sind, dass alle Punkte einer Trajektorie relativ dicht beieinander liegen und vor allem, dass die meisten Übergangstrajektorien in vertretbarem Maße oszillieren. Bei allen Übergangstrajektorien, die im reduzierten Zustandsraum nicht auf dem direkten Weg von einer Pose zu anderen führen, lässt sich das damit erklären, dass auch die Bewegungen über Zwischenzustände führen. Betrachten wir z. B. die rote Trajektorie in der unteren rechten Grafik von Abbildung 4.18, welche das Aufstehen aus der Bauchlage repräsentiert: Der Roboter muss sich schrittweise aufrichten, was dazu führt, dass er zwischenzeitlich mit gebeugtem Rücken steht, den er dann im letzten Schritt der Aufstehbewegung begradigt. Dieser Zwischenzustand ist in der oberen Grafik der Abbildung durch Bild 8 dargestellt. Eine ähnliche Bewegung vollführt der Roboter, wenn er aus der Hocke aufsteht (linkes Bild, lila Trajektorie). In Abbildung 4.19 sind die beiden Komponenten $y_2[2]$ und $y_5[2]$ noch einmal gegen die Zeit aufgetragen. Wie zuvor erwähnt, detektiert $y_2[2]$ sehr gut Lageänderungen in der senkrechten aber auch sagittalen Achse des Roboters, allerdings keinerlei morphologische Veränderungen, z. B. wenn der Roboter sich hinhockt. Dies wird allerdings durch die Komponente $y_5[2]$ registriert, welche verhältnismäßig kleine Reaktionen auf Lageveränderungen zeigt, aber v. a. Posenwechsel und die Beugung im Hüftgelenk des Roboter registriert. Zwar ist die Korrelation von `HLy_pe` und $y_5[2]$ gering, doch durch manuellen Vergleich der beiden Signale entsteht der Eindruck, dass $y_5[2]$ eine fehlerbereinigte Version des Hüftsensors ist. Insbesondere ist bei beiden Komponenten

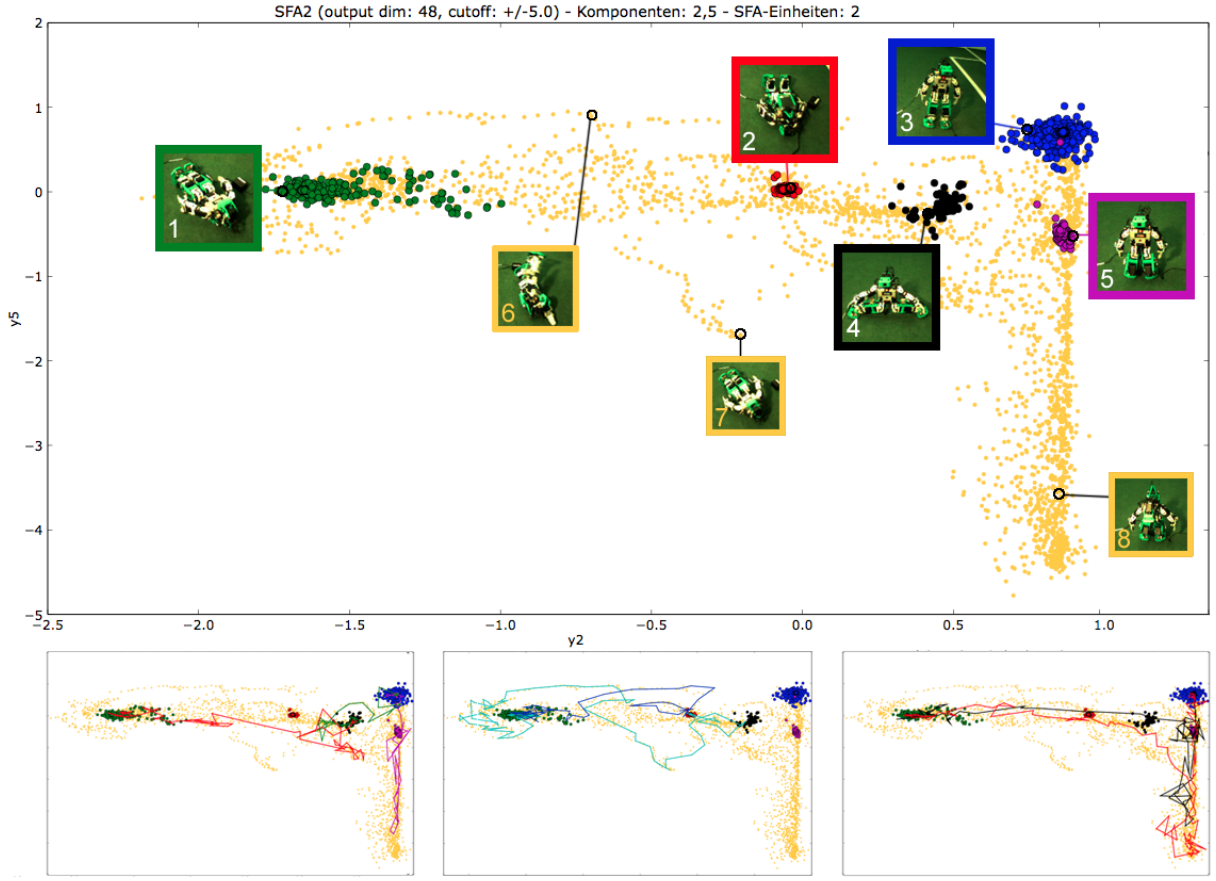


Abbildung 4.18: $Y_{2,5}[2]$ **Oben:** Datenpunkte und farblich hervorgehobene Posen. *Liegen (Rücken)* grün, *Liegen (Bauch)* rot, *Stehen* blau, *Spagat* schwarz, *Hocken* lila. Die Zwischenzustände sind gelb, wobei Bild 6 zeigt, wie der Roboter sich vom Rücken auf den Bauch dreht, Bild 7, wie er sich vom Bauch auf den Rücken dreht und Bild 8, wie er aus der Hocke aufsteht. **Unten links:** Drei Übergangstrajektorien *Spagat* → *Liegen (Bauch)* (rot), *Stehen* → *Spagat* (grün) und *Hocken* → *Stehen* (lila). **Unten Mitte:** Zwei Übergangstrajektorien *Liegen (Bauch)* → *Liegen (Rücken)* (hellblau) und *Liegen (Rücken)* → *Liegen (Bauch)* (dunkelblau). **Unten rechts:** Zwei Übergangstrajektorien *Liegen (Bauch)* → *Stehen* (rot) und *Stehen* → *Liegen (Bauch)* (schwarz).

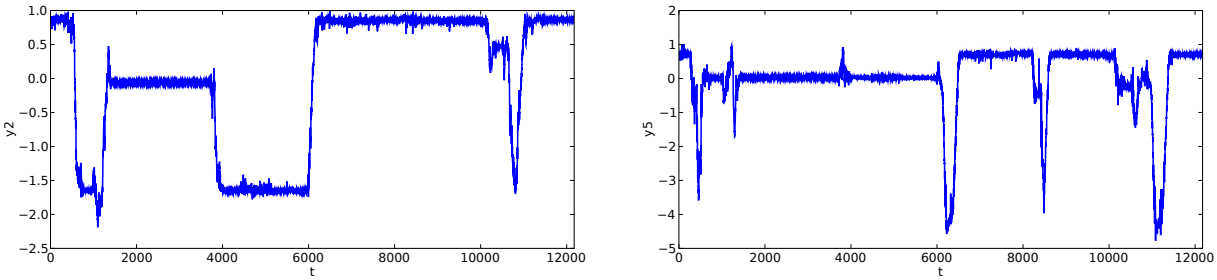


Abbildung 4.19: $y_2(2)$ und $y_5(2)$ gegen die Zeit aufgetragen. Höchste Korrelationskoeffizienten mit Sensoren: $\rho(\texttt{ARy\_pe}, y_2(2)) = -0.76$, $\rho(\texttt{HLy\_pe}, y_5(2)) = 0.38$.

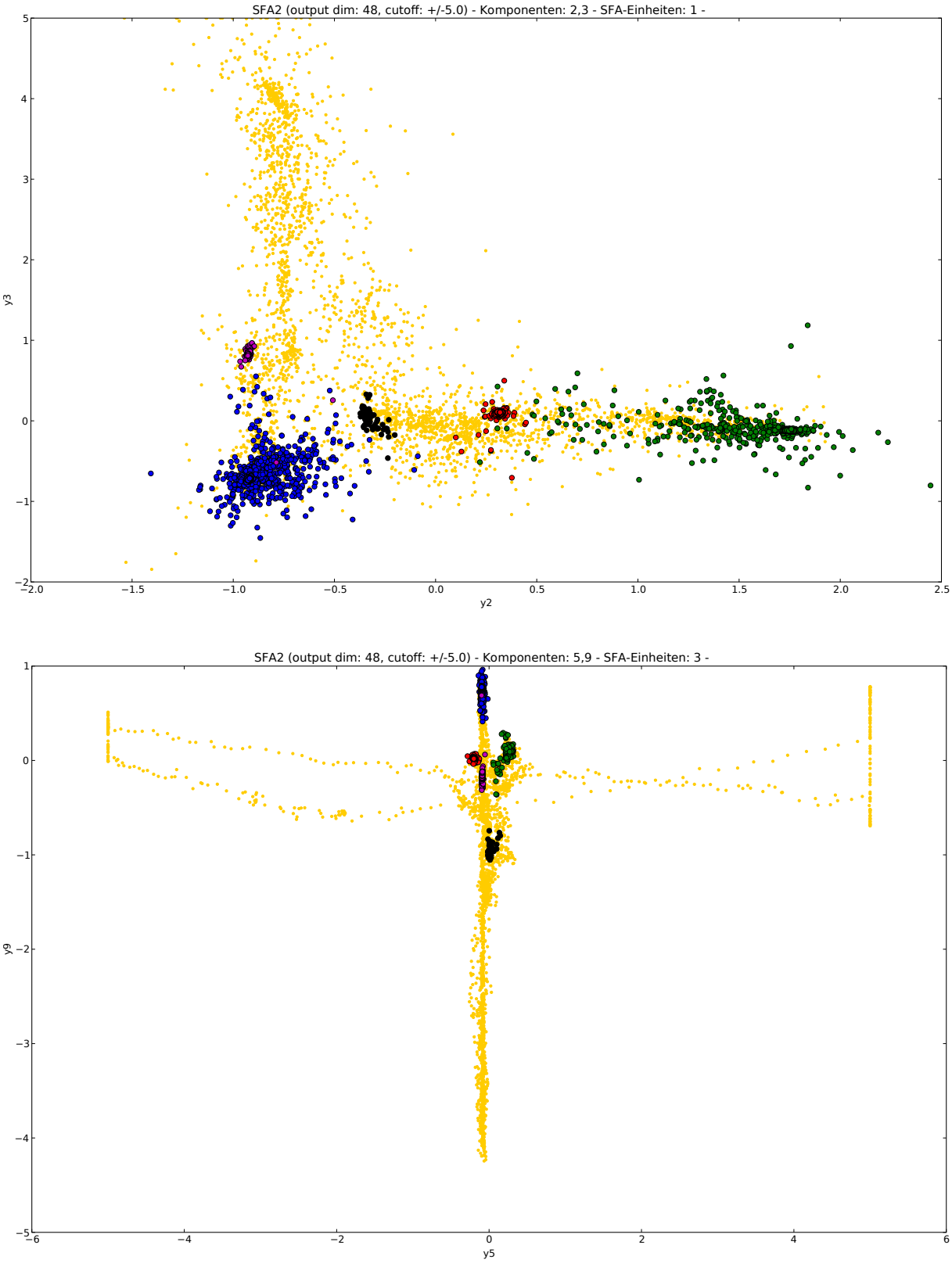


Abbildung 4.20: Oben: $Y_{2,3}[1]$. **Unten:** $Y_{5,9}[3]$ Für die Farblegende siehe Abbildung 4.18.

keine Überanpassung in Form von Überschwingern zu erkennen.

In Abbildung 4.20 sind zum Vergleich zusätzlichen $Y_{2,3}[1]$ (oben) und $Y_{5,9}[3]$ (unten) zu sehen. Während die $Y_{2,3}[1]$ eigentlich eine gute Dimensionsreduktion darstellt, sind bei $Y_{5,9}[3]$ deutlich die Überschwinger erkennbar. Somit scheiden Komponentenpaare höherer Iteration generell aus, da sie einen unbrauchbaren Kurvenverlauf aufweisen. Ebenso erschwert die Überanpassung eine Generalisierung der Komponenten auf ungesehene Daten.

4.2.3 Vergleich zu LLE und PCA

Um die Ergebnisse der SFA in den Kontext anderer Dimensionsreduktionsverfahren einordnen zu können, sollen sie mit den beiden in [Ste10] am besten bewerteten Verfahren verglichen werden[5], der bereits erwähnten *Principal Component Analysis (PCA)* sowie dem *Locally Linear Embedding (LLE)*. Die Idee der jeweiligen Verfahren soll im folgenden kurz erläutert werden, für eine Abhandlung sei beispielsweise auf die eben genannte Arbeit verwiesen. Es werden die Silhouette- und Procrusteswerte[6] für den gleichen Datensatz berechnet sowie die resultierenden zweidimensionalen Dimensionsreduktionen bewertet. Für beide Algorithmen wurden die Implementationen aus dem MDP-Toolkit verwendet [ZWWB09].

PCA

Wie in Abschnitt 2.5.1 erläutert, ist das Ziel der PCA, eine Rotationsmatrix zu finden, welche die Eingabematrix so dreht, dass die Varianz entlang der Hauptachsen nach der Rotation maximal ist. Zudem sortiert die PCA berechneten Signale aufsteigend nach ihrer Varianz. Unter der Annahme, dass die Komponenten mit der höchsten Varianz auch die verschiedenen Posen unterscheiden und enthalten, kann die PCA für eine Dimensionsreduktion sinnvolle Ergebnisse liefern.

Wie in [Ste10] wird vor der PCA eine Medianfilterung durchgeführt, da sich die sensorischen Daten wie bereits erwähnt viele Messfehler aufweisen.

Abbildung 4.21 zeigt die entstehende zweidimensionale sensorische Karte. Wie auch der Silhouette-Wert zeigt, kann die PCA die verschiedenen Posen nicht ganz so gut wie die SFA voneinander trennen. So ist die Pose *Liegen (Bauch)* (grün) nicht an einem Ort konzentriert, sondern über den linken Teil der Grafik verstreut. Der Procrusteswert hingegen ist sehr viel besser; dies ist allerdings nicht verwunderlich, da es sich bei der PCA um ein lineares Verfahren handelt. Die genauen Werte betragen:

$$S_{\mathrm{PCA}}(Y_{1,2}) \quad = \quad 0.63 \qquad\qquad PROC_{\mathrm{PCA}}(Y_{1,2}) = 0.06.$$

[5] Da in der genannten Arbeit sowohl Motor- als auch Beschleunigungssensoren verwendet wurden, sich dies für die SFA nicht als vorteilhaft erwiesen hat, werden die genannten Verfahren der Fairness halber auch nur auf Beschleunigungsdaten angewandt.

[6] Es wird eine normierte Version des Procrusteswertes verwendet, weswegen die numerischen Werte nicht mit denen aus [Ste10] übereinstimmen.

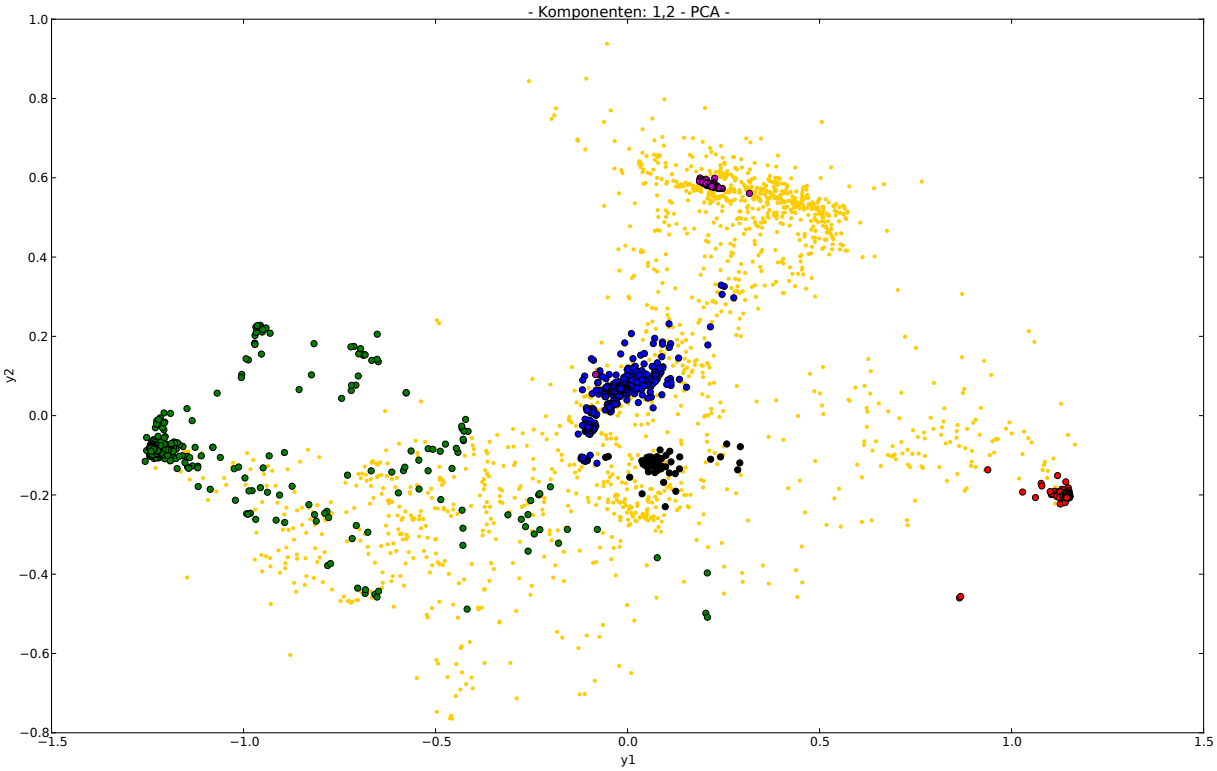


Abbildung 4.21: Dimensionsreduktion mittels PCA. Die Eingabedaten bestehen nur aus Beschleunigungssensoren, die Rohdaten werden durch ein Medianfilter mit Fenstergröße fünf vorverarbeitet. Für die Farblegende siehe Abbildung 4.18.

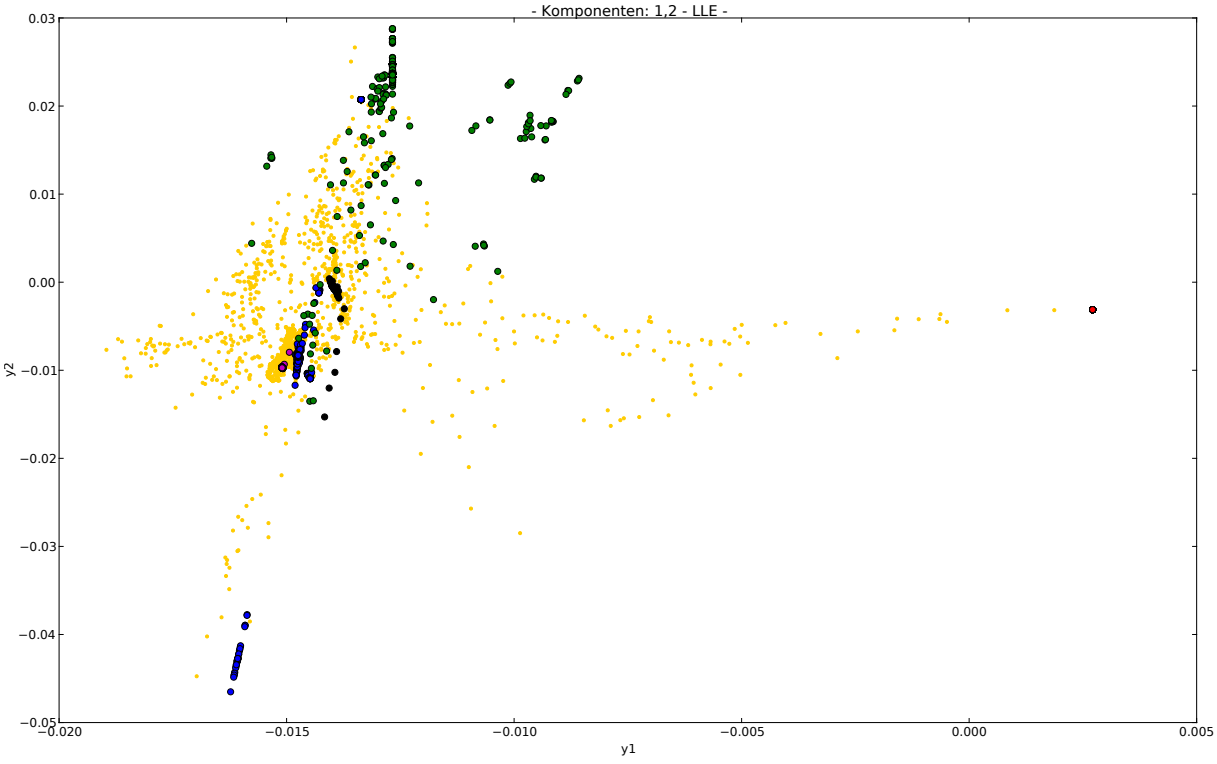


Abbildung 4.22: Dimensionsreduktion mittels LLE. Die Eingabedaten bestehen nur aus Beschleunigungssensoren, die Rohdaten werden durch ein Medianfilter mit Fenstergröße fünf vorverarbeitet. Für das LLE wird die Nachbarschaftsgröße auf $k = 91$ gesetzt. Für die Farblegende siehe Abbildung 4.18.

LLE

Die Kernidee beim LLE ist, vor allem lokale Beziehungen in den Daten nach der Dimensionsreduktion beizubehalten. Dazu wird zunächst für jeden Punkt aus dem Datensatz die k-Nachbarschaft bestimmt, d. h. seine k nächsten Nachbarn. Dabei wird der Parameter k zuvor festgesetzt. Als nächstes wird für jeden Punkt eine Linearkombination aus seinen k Nachbarn berechnet, d. h. jeder Punkt wird aus seinen Nachbarn *rekonstruiert*. Dadurch werden für jeden Punkt seine linearen Abhängigkeiten und die geometrische Struktur des Zustandsraumes repräsentiert. Im letzten Schritt wird mittels eines Eigenvektor-basierten Verfahrens eine Dimensionsreduktion berechnet, welche den Rekonstruktionsfehler im niedrigdimensionalen Zielraum minimiert.

Wie in [Ste10] wird eine Nachbarschaft von $k = 91$ gewählt. Es ergeben sich folgende Silhouette- und Procrusteswerte:

$$S_{\mathrm{LLE}}(Y_{1,2}) \quad = \quad 0.56 \qquad\qquad PROC_{\mathrm{LLE}}(Y_{1,2}) = 0.36.$$

Für das LLE wird ebenfalls eine Medianfilterung vorgenommen. Auch hier ist der Procrusteswert deutlich besser als bei der SFA, jedoch ist auch hier die Unterscheidbarkeit der verschiedenen Posen, wie durch die Silhouette angezeigt wird, schlechter als bei SFA und PCA. Wie in Abbildung 4.22 zu sehen ist, sind vor allem die Posen *Liegen (Bauch)* sowie *Liegen (Rücken)* nicht an einem Punkt konzentriert, sondern sich erstrecken über weite Areale der Grafik. Die den restlichen Posen zugeordneten Datenpunkte sind lokal und zusammenhängend, können durch ihre Verschränkung mit den Liegeposen aber dennoch nicht gut unterschieden werden. Ein Grund kann sein, dass die Nachbarschaftsgröße k nicht optimal gewählt ist. Jedoch erfolgt an dieser Stelle keine Evaluation der optimalen Parameterwahl.

Zusammenfassend lässt sich sagen, dass die PCA und die LLE wie in [Ste10] gute Procrusteswerte aufweisen, auch wenn hier nur Beschleunigungssensordaten und keine Motorpositionsdaten verwendet werden. Die im Vergleich zur SFA schlechten Silhouettewerte erklären sich dadurch, dass die Algorithmen darauf abzielen, im Ergebnis nahe an den Originaldaten zu bleiben, während die SFA vornehmlich hinsichtlich Langsamkeit optimiert. Dadurch kommt es bei der SFA zu starken Verzerrungen des dimensionsreduzierten Raums gegenüber den Originaldaten, jedoch mit dem positiven Nebeneffekt, dass eine klarere Trennung der verschiedenen Posen zustande kommt.

Zusätzlich ist zu bemerken, dass der LLE-Algorithmus hohe Komplexität hat und dementsprechend auf dem verfügbaren Datensatz deutlich mehr Zeit für die Berechnung benötigt als PCA und SFA.

4.3 Zusammenfassung

Dieses Kapitel hat sich mit der Frage beschäftigt, welche Ergebnisse die SFA für die Erkennung von Posen unter Verwendung von propriozeptiven Sensoren liefert, und ob sie sich für eine Dimensionsreduktion eines sensorischen Raumes eignet. Die Untersuchung ergab, dass die SFA

für beide Anwendungen vielversprechende Ergebnisse liefert. In den Sensordaten erkennbare Zustände, welche hinreichend lange gehalten werden, tauchen in den langsamsten Komponenten auf und sind gut unterscheidbar. Die Rolle der sensorisch weniger ausschlaggebenden Posen nehmen in den betrachteten Daten die Hock- und Spagatpose ein, die in der Tat schwieriger zu entdecken sind, aber dennoch in den SFA-Komponenten auftauchen. Es wird zudem deutlich, dass bei den verwendeten Trainingsdaten mehr als zwei SFA-Iterationen eine Überanpassung an die Eingabedaten zur Folge haben.

Ein Problem mit der Anwendung der SFA liegt in der Natur unüberwachter Lernverfahren: Es ist nicht offensichtlich, welche Komponenten in welcher Iteration die beste Lösung darstellen. Dies ist insofern problematisch, als dass die SFA die Forderung der Nachbarschaftstreue schlecht erfüllt. Somit muss je nach Anwendungsfall abgewogen werden, ob diese Anforderung fallengelassen werden kann. In manchen Fällen ist die Umordnung des neuen Repräsentationsraums in vielen Fällen nicht unbedingt schlecht, solange die Abstände zwischen den sensorischen Zuständen zumindest sinnvoll sind. Im Falle der Beschleunigungsdaten bietet die Verschlechterung der exakten Nachbarschaftstreue sogar einen echten Mehrwert, da die Trajektorien bei Posenübergängen bei den SFA-Komponenten deutlich glatter und lokal beschränkter als bei den Originaldaten sind. Das wirkt sich auch äußerst positiv auf die Zwischenzustände aus. Durch die Unähnlichkeit der Reduktion zu den Originaldaten wird allerdings auch ihre automatisierte qualitative Bewertung erschwert.

Um die SFA für den produktiven Einsatz auf den Robotern nutzbar zu machen, sind weitere Untersuchungen hinsichtlich optimaler Trainingsdaten und der Generalisierung der Komponenten notwendig. Insbesondere ist wichtig zu evaluieren, wie sich sehr lange Trainingsdatensätze auswirken. Dabei ist die Frage von großer Bedeutung, ob Posen, die nur selten angefahren werden, nach einiger Zeit aus den SFA-Komponenten verschwinden und sensorisch stark hervorstechende Merkmale wie *Stehen* und *Liegen* dominieren, oder ob sich einige Komponenten auch auf seltenere Posen spezialisieren können. Ebenso wird mit zunehmender Länge der Eingabedaten ebenso wie mit höherer Dimensionalität des Eingaberaums voraussichtlich die Überanpassung zurückgehen, weswegen gesonderte Betrachtungen über die optimale Anzahl an SFA-Iterationen notwendig sein werden. Zusammen mit der Verhinderung einer Überanpassung geht als weiterer wichtiger Aspekt die Untersuchung die Generalisierbarkeit der Komponenten auf ungesehene Daten einher.

Das nächste Kapitel wendet sich von der Verwendung der SFA zur sensorischen Klassifikation ab und beschäftigt sich mit dem Einsatz der SFA in dynamischen Bewegungsabläufen. Insbesondere geht es darum, die glättende Eigenschaft der SFA auszunutzen, und eine SFA-Komponente als hochreaktives Filter in einer sensomotorischen Schleifen zu nutzen.

Kapitel 5

Analyse und Verbesserung eines zweibeinigen Laufmusters

Nachdem im letzten Abschnitt der Fokus auf der Analyse von statischen Roboterposen lag, wende ich mich in diesem Kapitel der Frage zu, wie die SFA zur Analyse und Verbesserung von dynamischen Bewegungsabläufen verwendet werden kann. Dabei geht es konkret um ein neuronal implementiertes zweibeiniges Laufmuster, welches in [Wer08] entwickelt wurde. Bereits in meiner Studienarbeit [Höf09] habe ich analysiert, welche Komponenten die SFA aus den Beschleunigungsdaten des Roboters extrahieren kann, wenn dieser läuft bzw. durch unvorhergesehenen Gleichgewichtsverlust umfällt. Dabei konnte gezeigt werden, dass bei Lauf-Fall-Sequenzen die Veränderung der Lage des Roboters als langsamste Komponente extrahiert wird. Weitere langsamste Komponenten zeigen für das Laufen charakteristische Signale wie die Pendelbewegungen in der frontalen und sagittalen Richtung. Es wurden weiterhin Betrachtungen bzgl. der Qualität der Signale in Abhängigkeit von der Anzahl von SFA-Iterationen und weitergereichten Komponenten je Iteration sowie bzgl. der Generalisierbarkeit der SFA-Komponenten angestellt. Es wurde gezeigt, dass mit der SFA eine Umfalldetektion realisierbar ist, da die SFA zuverlässig die Lage des Roboters extrahiert.

Bei näherer Betrachtung wird allerdings klar, dass diese Umfall*detektion* nicht unbedingt eine Umfall*prävention* darstellt. Dazu ist es nicht nur notwendig, dass die SFA den Zustand der Roboters richtig extrahiert, sondern sie muss es auch hochreaktiv und ohne Zeitverzögerung tun. Es besteht zwar die Hoffnung, dass die SFA im Prinzip diese Reaktivität aufweist, da sie eine punktförmige Transformation darstellt. Allerdings lassen bereits die Ergebnisse aus [Höf09] darauf schließen, dass eine robuste Umfallprävention unter ausschließlicher Verwendung einer iterierten quadratischen SFA schwer realisierbar ist: Eine erhöhte Anzahl an Iterationen und weitergereichten Komponenten ist einerseits notwendig für eine ausreichende Glättung und somit für die Stabilität des Signals, andererseits hat sie auch eine Überanpassung an die Trainingsdaten zur Folge. Letztere führt dazu, dass die Komponenten bei ungesehenen Daten stark oszillieren, wodurch trotz hoher Reaktivität keine ausreichende Stabilität mehr gegeben ist.

Aufbauend auf den Ergebnissen von [Höf09] sollen in diesem Kapitel folgende Hauptziele verfolgt werden: Zum einen soll gezeigt werden, dass sich das Problem der Vereinbarkeit von Stör-

unanfälligkeit und Reaktivität sowie der Generalisierbarkeit der SFA-Komponenten durch die Verwendung einer zeitlich eingebetteten SFA besser beherrschen lässt. Zum anderen wird unter Rückgriff auf die Tatsache, dass zeitlich eingebettete SFA-Komponenten formal Volterra-Filtern entsprechen (siehe Abschnitt 2.5.6), die SFA-Komponente als Filter in der Steuerungsstruktur des Laufverfahrens verwendet. Es wird gezeigt, dass sich dadurch die Reaktivität der sensomotorischen Schleife ohne gravierende Einbußen bzgl. der Stabilität erhöht. Zusätzlich stellt diese Anwendung der SFA einen Machbarkeitsnachweis für die Verwendung der SFA in sensomotorischen Schleifen dar.

Das Kapitel ist folgendermaßen gegliedert: Zunächst wird das betrachtete Laufmuster sowie das neuronale Netz, durch welches es gesteuert wird, erläutert. Daraufhin wird untersucht, welche Ergebnisse die SFA mit verschiedenen Konfigurationen angewandt auf die Trainingsdatensätze liefert. Aufbauend auf den Ergebnissen aus [Höf09] wird zunächst eine quadratische SFA betrachtet, dann wird die SFA mit zeitlicher Einbettung evaluiert. Aufbauend auf den gewonnenen Erkenntnissen wird untersucht, unter welchen Voraussetzungen eine gelernte SFA-Komponente in der sensomotorischen Schleife, welche das Laufen generiert, eingesetzt werden kann. Die SFA-Lösung wird mit anderen Filtern verglichen und es wird diskutiert, welche Vorteile und Verbesserungen für das Laufverhalten durch die vorgestellten Lösungen gewonnen werden können.

5.1 Laufmuster

Abbildung 5.1 zeigt ein paar Momentaufnahmen einer kurzen Sequenz einer Laufbewegung eines humanoiden Roboters der A-Serie. Das Gangmuster wurde von Benjamin Werner in [Wer08] entwickelt und wird mit Hilfe einer sensomotorischen Schleife erzeugt. Im Ausgangszustand steht der Roboter, was einen stabilen Fixpunkt der sensomotorischen Schleife darstellt. Wird der Roboter manuell oder durch Aktivierung eines neuronalen Oszillators in der frontalen Ebene angestoßen, wird zunächst eine Schwingung in dieser Ebene erzeugt, die den Roboter das Gewicht von einem Bein auf das andere verlagern lässt. Dies dient als Grundlage, um die Vorwärtsbewegung der Beine des Roboters in sagittaler Richtung zu starten, welches dann das Laufverhalten in Gang setzt.

Leider kann es bei dem Laufmuster bereits bei der Phase der frontalen Schwingung vorkommen, dass die Laufbewegung instabil wird und der Roboter umfällt. Dies kann verschiedene Ursachen haben: Zum einen läuft der Roboter Gefahr auf Untergründen mit hoher Reibung mit seinen Füßen am Boden hängenzubleiben. Zum anderen kann eine leichte Neigung des Untergrunds dazu führen, dass der Roboter entweder seitlich oder vorwärts fällt, da er seine Bewegung zu stark in die abgeneigte Seite auslenkt. Die Sequenz in Abbildung 5.1 zeigt, wie der Roboter zunächst frontal zu pendeln anfängt, dann beginnt vorwärts zu laufen und schließlich umfällt, höchstwahrscheinlich da eher mit seinem Fuß am Teppichboden hängenbleibt.

Das gesamte neuronale Netz ist relativ komplex, weswegen an dieser Stelle nicht im Detail die gesamte Struktur des Netzes vorgestellt werden kann. Jedoch wird im nächsten Abschnitt das Modul erläutert, welches für die Generierung der frontalen Pendelbewegung zuständig ist.

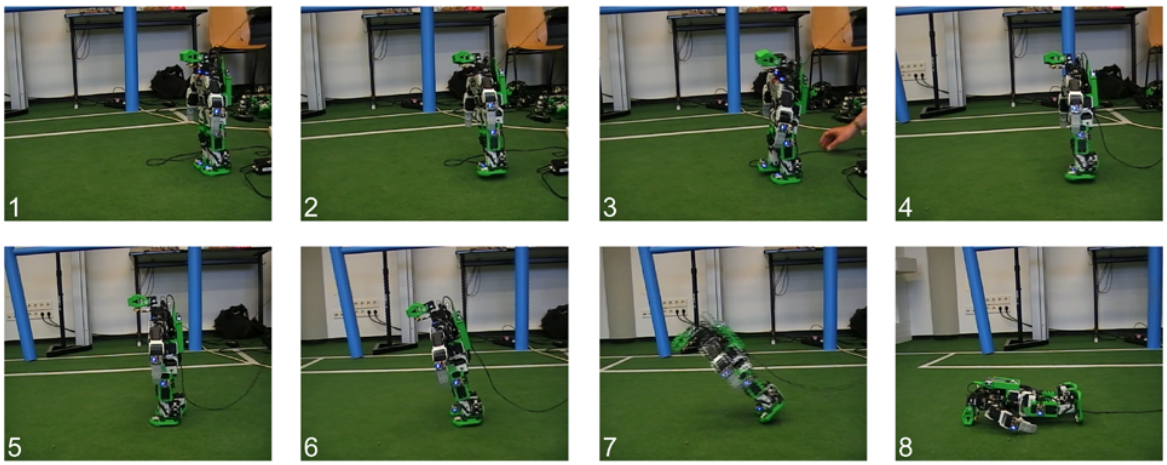
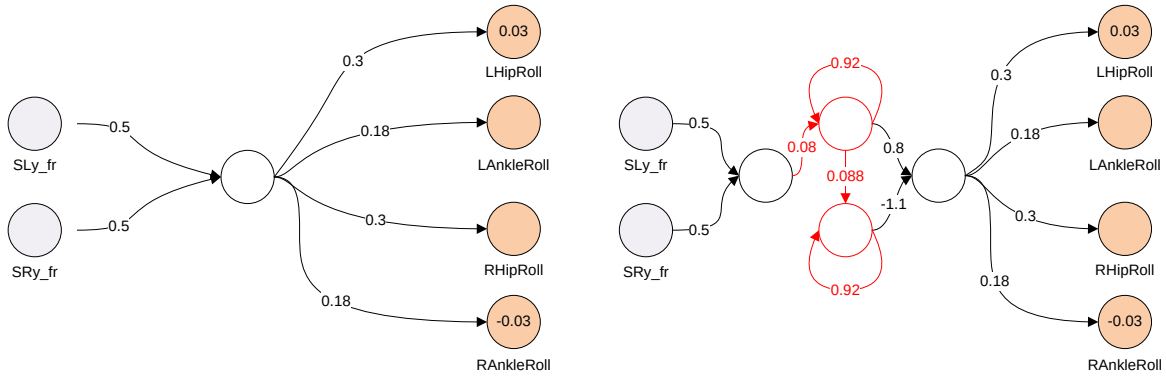


Abbildung 5.1: Roboter vollführt eine Laufbewegung und fällt zu Boden.



(a) Ohne Filter (b) Mit IIR-Filter

Abbildung 5.2: Komponente des neuronalen Netzes, welche die zum Laufen notwendige stabile Schwingung in der frontalen Ebene generiert. **Links:** Netz ohne Filterung. **Rechts:** Netz mit IIR-Filter bestehend aus zwei Leaky-Integrator-Neuronen.

Dieses Modul wird in Abschnitt 5.3 erneut Gegenstand der Betrachtung sein, wenn ein Teil dieser Struktur durch ein SFA-Modul ersetzt wird, um die Reaktivität des Laufmusters zu erhöhen.

5.1.1 Generierung des Laufmusters

In Abbildung 5.2 sind zwei Netze abgebildet, welche die Schwingung in der frontalen Richtung generieren. Als Eingänge der Netze dienen zwei Beschleunigungssensorwerte, welche jeweils in frontale Richtung zeigen und an den Schultern des Roboters angebracht sind. In Abbildung 5.2a ist die ursprüngliche Form des Netzes zu sehen, in welchem die gemittelten Sensorwerte direkt an die Motoren gesendet werden. Das Signal, was verschieden gewichtet an die Motoren gesendet wird, wird im folgenden als *Steuersignal* bezeichnet. Die direkte Verwendung der Sensorwerte als Steuersignal hat jedoch zur Folge, dass auch hochfrequente Signalanteile an die Motoren geleitet werden. Das führt zum einen dazu, dass durch häufiges Umspulen der Motor stärker belastet

wird, wodurch der Stromverbrauch steigt und höherer Verschleiß auftritt. Zum anderen sinkt die Stabilität des Laufmusters, da die meisten hochfrequenten Anteile eher durch Störsignale und sensorisches Rauschen hervorgerufen werden, welche eine ruhige und gleichmäßige Bewegung verhindern.

Die rechte Abbildung zeigte eine verbesserte Version der soeben vorgestellten Struktur. Das gemittelte Sensorsignal wird hierbei in die rotgefärbte neuronale Struktur geführt, welche aus zwei *Leaky-Integrator*-Neuronen besteht. Dabei generiert das zweite, untere Neuron ein Signal, welches zu dem Signal des ersten, oberen Neurons um 90° phasenverschoben ist. Die Ausgangswerte der beiden Filterneuronen werden wieder gewichtet zusammengeführt und schließlich das entstehende Signal mit verschiedenen Gewichtungen an die Motoren an den Rollgelenken der Hüfte sowie der Knöchel geleitet. Die verwendeten Leaky-Integrator-Neuronen übernehmen die Funktion eines Tiefpassfilters und generieren ein glatteres Steuersignal. In Abschnitt 3.3.1 wurden verschiedene Filtertypen beschrieben, dabei entspricht das hier zur Anwendung kommende neuronale Filter aufgrund der Rekursivität formal einem IIR-Filter. Jedoch hat auch diese zweite Variante einen Nachteil: Die Funktionsweise des verwendeten Tiefpassfilters beruht darin, die eingehenden Sensorwerte über mehrere Zeitschritte hinweg zu mitteln, was einerseits den positiven Effekt der Stabilisierung hat, andererseits die Reaktivität der sensomotorischen Schleife verringert. Insbesondere bei unvorhergesehenen Störungen, z. B. durch Hindernisse, verringert sich die Reaktionszeit des Systems, um die Störung auszugleichen.

In Abschnitt 5.3 wird daher der Versuch unternommen, das IIR-Filter durch eine Filterstruktur auf Basis der SFA zu ersetzen, um die Reaktivität der sensomotorischen Schleife wieder zu erhöhen, ohne die Stabilität zu verschlechtern. Ein wichtiges Ziel des folgenden Abschnittes ist daher herauszufinden, wie durch die SFA eine gut generalisierbare Komponente extrahiert werden kann, welche als Steuersignal sowohl stabil als auch hinreichend reaktiv ist.

5.2 Analyse der Laufsequenzen

Das Ziel der folgenden Abschnitte ist, mittels der SFA eine langsame, gut generalisierende Komponente zu finden, welche die frontale Oszillation der Laufbewegung extrahiert und als Steuersignal für das Laufen dienen kann. Meine Studienarbeit [Höf09] konzentrierte sich auf die Analyse von solchen Sequenzen, während derer der Roboter beim Laufen das Gleichgewicht verliert und umfällt. Die auf diesen Sequenzen gelernte langsamste SFA-Komponente repräsentierte immer, ob der Roboter stand oder lag. Weniger langsame Komponenten repräsentierten zwar die oszillierende Laufbewegung, jedoch wird im folgenden gezeigt, dass diese Pendelbewegungen auch aus reinen Laufsequenzen extrahiert werden können; daher beschränke ich mich in diesem Kapitel auch auf die Analyse solcher Sequenzen, in welchen der Roboter weitgehend stabil läuft und nicht umfällt.

Zur Extraktion solcher Sequenzen wird die quadratische SFA sowie die quadratische SFA mit zeitlicher Einbettung betrachtet. Die in Abschnitt 3.2.2 vorgestellte Expansion unter Verwendung des tangens hyperbolicus wird an dieser Stelle nicht gesondert behandelt, da die Analyse der SFA mit quadratischer Expansion einfacher ist; so wird eine Untersuchung ausgewählter Komponen-

ten unter Verwendung der quadratischen Form aus Abschnitt 2.4.3 durchgeführt. Um aber die praktische Anwendbarkeit der tanh-SFA zu zeigen, wird diese zu einem späteren Zeitpunkt zur Generierung des Laufmusters verwendet. Dabei ergeben sich zur quadratischen SFA qualitativ gleichwertige Ergebnisse.

Im Hinblick auf die Tatsache, dass die SFA in einer sensomotorischen Schleife für die Generierung des Laufens eingesetzt werden soll, werden ausschließlich Beschleunigungssensoren betrachtet. Die Auswertung von Motorwinkelsensoren bietet sich nicht an, da diese lediglich die Konfiguration des Roboters ohne Einbeziehung der Umwelt darstellen und beispielsweise kaum Auskunft über die Neigung des Roboters geben. In dem Fall, dass die neuronale Struktur übersteuert oder unvorhergesehene äußere Einflüsse eintreten, so dass der Roboter droht in eine Richtung umzufallen, äußert sich dies in der Regel nicht in den Motorwinkelsensoren, dagegen umso deutlicher in den Beschleunigungssensoren, beispielsweise an den Schultern.

5.2.1 Trainings- und Testdaten

Zur Analyse werden verschiedene Roboter und aufgenommene Bewegungssequenzen herangezogen. Die Trainingssequenzen haben eine Länge von ca. 60 Sekunden und beinhalten lediglich das Laufen des Roboters. Die Trainingssequenzen werden danach unterschieden, mit welchem neuronalen Netz sie generiert wurden: Entweder mit der Struktur aus Abbildung 5.2a, ohne Filterung der Beschleunigungssensorwerte (als *Netz ohne Filter* bezeichnet), oder aus 5.2b, unter Verwendung eines tiefpassgefilterten Steuersignals (als *IIR-Netz* bezeichnet).

Die Sequenzen beinhalten lediglich Laufdaten und es treten während des Laufens keine zufälligen oder gezielt herbeigeführten externen Störungen auf. Durch den griffigen Untergrund, welcher in einem Teppich besteht, können allerdings leichte Unregelmäßigkeiten auftreten, welche aber durch die Länge der Sequenzen und die Betrachtung verschiedener Sequenzen im Mittel keinen gravierenden Einfluss haben.

Da die Ergebnisse auf den verschiedenen Robotern keine qualitativen Unterschiede aufweisen, wird im folgenden nicht zwischen den einzelnen Robotern unterschieden. Zudem weisen sowohl die betrachteten Trainingsdaten als auch die resultierenden SFA-Komponenten eine hohe Gleichförmigkeit und keine qualitativen Unterschiede auf. Daher stammen die angegebenen Grafiken immer von einer exemplarischen konkreten Trainingssequenz, die repräsentativ für beliebige andere Sequenzen des gleichen Typs ist.

Um auf die Ähnlichkeit der Sensoren und der SFA-Komponenten schließen zu können, wird wie im letzten Kapitel der Korrelationskoeffizient verwendet. Eine Betrachtung der Korrelationskoeffizienten der Sensoren untereinander ergibt, dass die Ähnlichkeit der Signale zwischen den Körperhälften bei den meisten Sensoren relativ niedrig ist. Es weisen unter den gleich gerichteten Sensoren nur die frontalen Schultersensoren ($|\rho(\texttt{SLy\_fr}, \texttt{SRy\_fr})| \geq 0.98$) eine hohe Korrelation zueinander auf. Sieht man sich die Korrelationskoeffizienten sowie die Kurvenverläufe der Sensoren in Abbildung 5.3 an, so erkennt man, dass sich die gleichgerichteten Sensoren immer stärker unterscheiden, je näher sie sich zu den Füßen befinden. Dies lässt darauf schließen, dass beim Aufprall des Fußes auf den Boden eine Kraft auf den Roboterkörper wirkt, welche sich über

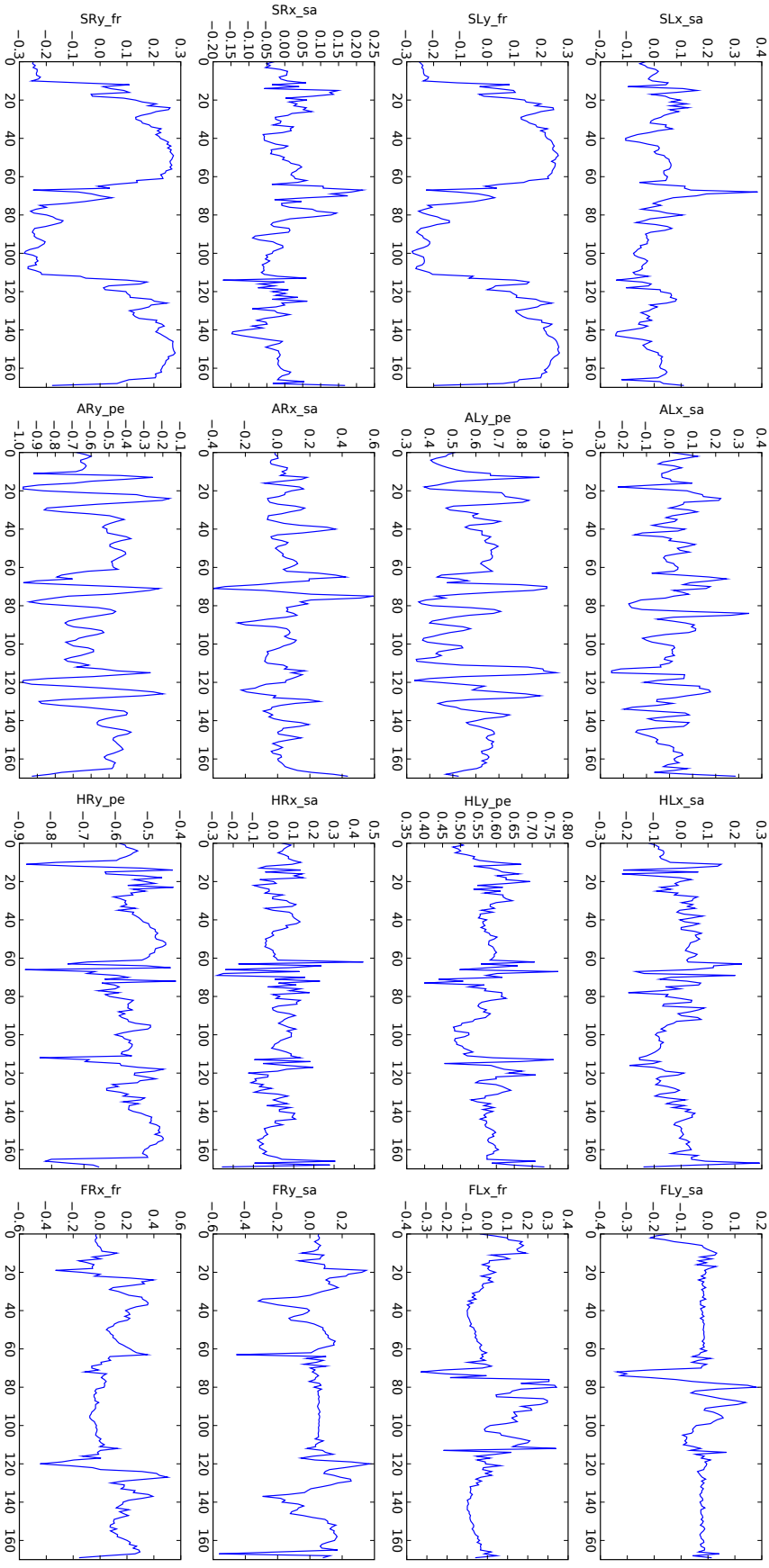


Abbildung 5.3: Beschleunigungssensordaten eines kurzen Ausschnittes aus einer Laufsequenz

die Gliedmaßen hinweg entlädt und zum Kopf hin immer schwächer wird. Es ist zu vermuten, dass die Sensoren, welche näher an den Füßen sind, stärker korreliert sind, wenn eine zeitlich verschobene Variante des Korrelationskoeffizienten betrachtet wird.

Bei der Suche nach langsamsten Komponenten, welche die frontale Pendelbewegung extrahieren, werden im folgenden vor allen Dingen die Schultersensoren zum Vergleich herangezogen. Da die beiden frontalen Schultersensoren der beiden Körperhälften hohe Ähnlichkeit zueinander aufweisen, genügt der Vergleich mit einem der beiden Sensoren.

5.2.2 Quadratische SFA

In [Höf09] wurden die extrahierten langsamsten Komponenten verschiedener SFA-Konfigurationen qualitativ untersucht und miteinander verglichen. Dabei wurden die als Eingabedaten verwendeten Sensoren, die Anzahl der hintereinander ausgeführten SFA-Einheiten sowie die Anzahl der pro Einheit verwendeten langsamsten Komponenten aus der davorliegenden Einheit variiert. Desweiteren wurde eine SFA-Hierarchie mit getrennter Vorverarbeitung der Eingabedaten evaluiert, die jedoch keine signifikante Verbesserung ergab und daher im folgenden nicht betrachtet wird.

Bei der ersten Versuchsreihe werden ein bis mehrere quadratische SFA-Einheiten hintereinander geschaltet und auf die Gesamtheit der verfügbaren Beschleunigungssensoren angewandt. Dabei kann sowohl die Anzahl der SFA-Iterationen, als auch die Anzahl der von einer Iteration zur nächsten weitergereichten Komponenten variiert werden. Auf eine vorangestellte lineare SFA wird wie in [Höf09] verzichtet, da diese eine vernachlässigbare Auswirkung auf die Güte der extrahierten Komponenten hat.

Zunächst wird eine Anzahl von 24 weitergereichten langsamsten Komponenten pro Stufe festgehalten und betrachtet, wie sich die langsamste Komponente je Iteration verhält. Abbildung 5.4 zeigt die langsamste Komponente y_1 jeweils nach einer, zwei, fünf und sechs Iterationen. Die Länge der Trainingsdaten beträgt 60 Sekunden. Wie deutlich zu sehen ist, wird die frontale Pendelbewegung als langsamste Komponente extrahiert. Werden kürzere Ausschnitte als Trainingssequenzen verwendet, so kann es passieren, dass in höheren SFA-Iterationen die Kovarianzmatrizen singulär werden, oder zumindest dass in den langsamsten Komponenten eine langsamere Schwingung gefunden wird, als eigentlich in den Daten vorhanden ist. Letzterer Fall kann schon nach drei Iterationen auf einer 5-sekündigen Sequenz auftreten. Es ist davon auszugehen, dass durch längere Trainingsdaten die Überanpassung geringer ist und somit die Komponenten besser generalisieren.

Zudem wird das Ergebnis der SFA auf der Variante vom Laufmuster betrachtet, welches die frontalen Beschleunigungssensoren der Schulter als Steuersignal verwendet. In Abbildung 5.5 sind die entsprechenden Kurvenverläufe zu sehen. Die frontale Komponente ist aber ab der fünften Iteration in y_2 zu finden, die langsamste Komponente in diesen Iterationen kann nicht näher gedeutet werden. Anstatt dass der η-Wert der frontalen Komponente sinkt, steigt er daher in späteren Iterationen. Daher eignet sich aufgrund der schwachen Glättung keine dieser Komponenten als Steuersignal.

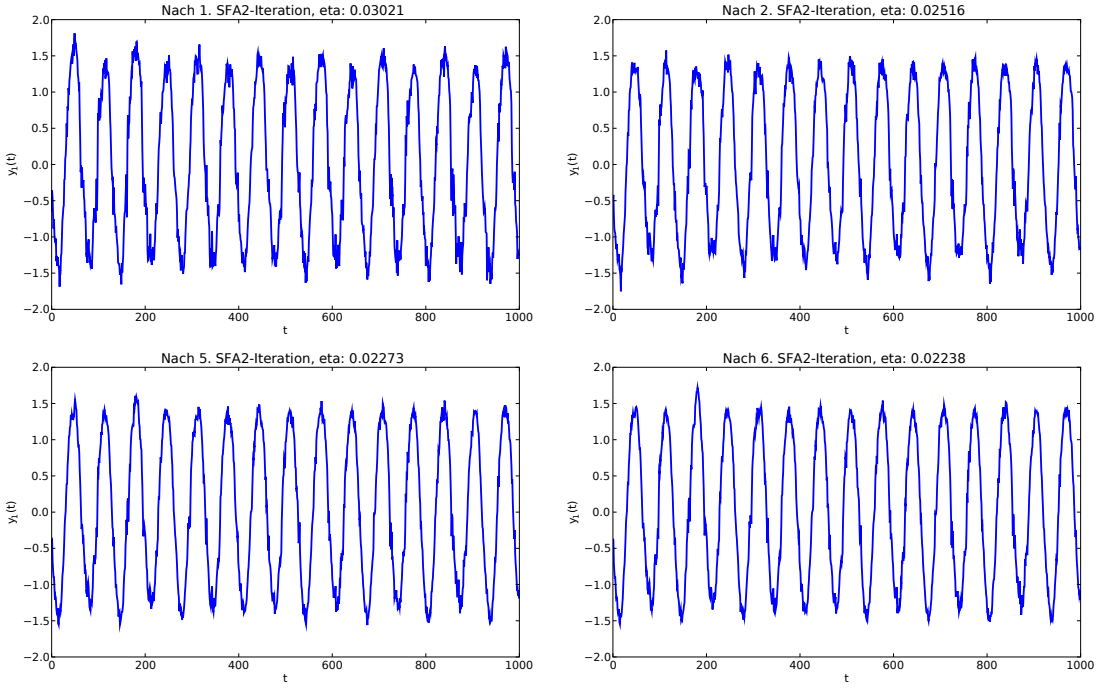


Abbildung 5.4: Frontal korrelierte durch die quadratische SFA entdeckte Komponente nach 1, 2, 5 und 6 SFA^2-Iterationen. Die Trainingsdaten bestehen aus einer 60-sekündigen Laufsequenz, es ist ein zehnsekündiger Ausschnitt aus dieser dargestellt.

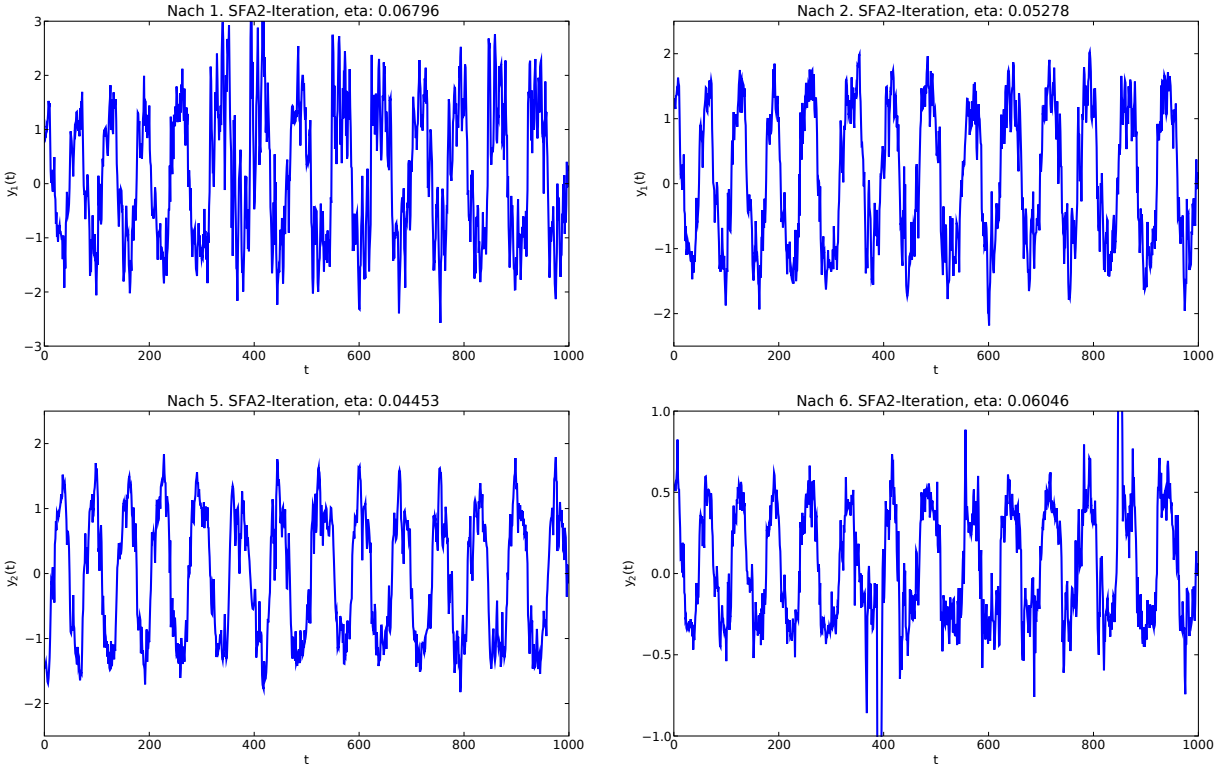


Abbildung 5.5: SFA angewandt auf ein Signal, welches durch Laufen ohne Filterung generiert wurde: Frontal korrelierte durch die quadratische SFA entdeckte Komponente nach 1, 2, 5 und 6 SFA^2-Iterationen. Die Trainingsdaten bestehen aus einer 60-sekündigen Laufsequenz, es ist ein zehnsekündiger Ausschnitt aus dieser dargestellt.

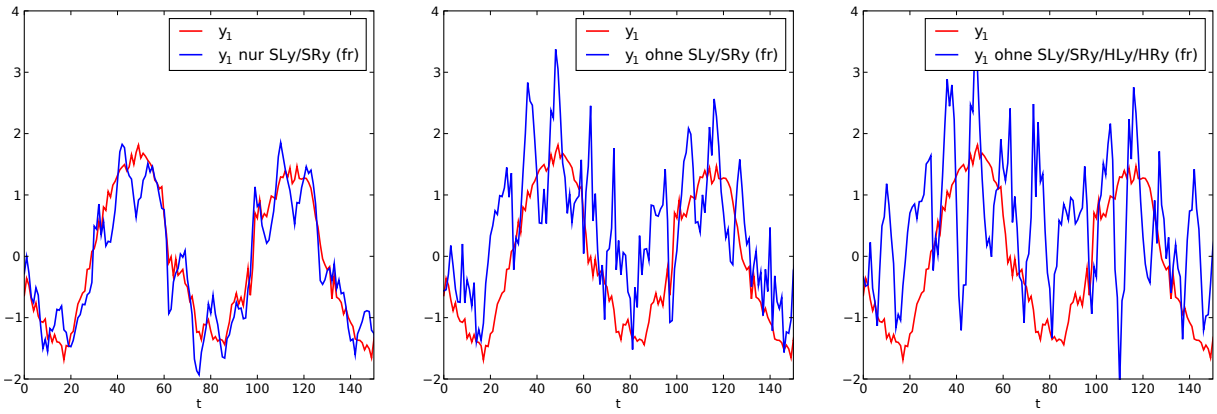


Abbildung 5.6: SFA-Signal der langsamsten (frontalen) Komponente, das entsteht, wenn gezielt Sensoren eliminiert werden. Zum Vergleich stehen unter jeder Kurve die η-Werte der manipulierten langsamsten Komponenten angegeben. Der η-Wert der ursprünglichen langsamsten Komponente (rote Kurve) beträgt 0.02948.

Analyse der frontalen Komponente

Bei der Analyse der dynamischen Komponente gestaltet es sich im Gegensatz zu den Daten aus Kapitel 4 um einiges schwieriger, eine sinnvolle Aussage über die Funktionsweise der frontalen Komponente mit Hilfe der Quadratischen-Form-Analyse zu erhalten. Man kann nicht wie zuvor auf die Berechnung der maximal exzitatorischen und inhibitorischen Stimuli zurückgreifen, da es sich bei der frontalen Komponente um ein nicht-statisches Signal handelt, welches oszilliert. Stattdessen ist man eher daran interessiert, wie die Glättung des dynamischen Eingabestroms vonstatten geht.

Daher müssen Analysen direkt auf die Betrachtung der linearen und quadratischen Terme hinauslaufen. Eine einfache Analyse ergibt sich aus der separaten Betrachtung von linearem und quadratischem Teil einer SFA-Komponente. Allerdings lässt sich für die frontale Komponente hier keine interessante Aussage ableiten: Sowohl der lineare als auch quadratische Anteil fangen die dynamische Bewegung ein, d. h. oszillieren, und weisen starke hochfrequente Anteile auf, welche allerdings in der Summe weggefiltert werden.

Eine weitere Möglichkeit, um die jeweiligen Anteile der Sensoren am Signal zu betrachten, liegt darin einen realen Datenstrom zu verwenden und zu modifizieren, indem bestimmte Sensorwerte auf Null gesetzt werden. Diese Herangehensweise macht allerdings in der Praxis nur bei den ersten quadratischen SFA-Iterationen Sinn: Zum einen degenerieren die SFA-Komponenten aufgrund der Überanpassung an die Trainingsdaten mit jeder weiteren Iterationen immer stärker. Zum anderen fallen nicht nur die Null gesetzten Sensorwerte heraus, sondern auch alle Mischterme, deren Bestandteil sie sind, was sich inbesondere in späteren SFA-Iterationen auswirkt. Für die erste SFA-Iteration lässt sich so wie in Abbildung 5.6 zu sehen auf einfache Weise zeigen, dass die beiden frontalen Schultersensoren `SLy_fr` sowie `SRy_fr` auch in der SFA-Komponente die Hauptträger der Oszillation sind, auch wenn die frontalen Fußsensoren offensichtlich ebenso dafür benutzt werden. Werden nur die frontalen Schultersensoren weggelassen, bleibt ein Signal mit vielen hochfrequenten Anteilen und einer nur schwach erkennbaren frontalen Schwingung;

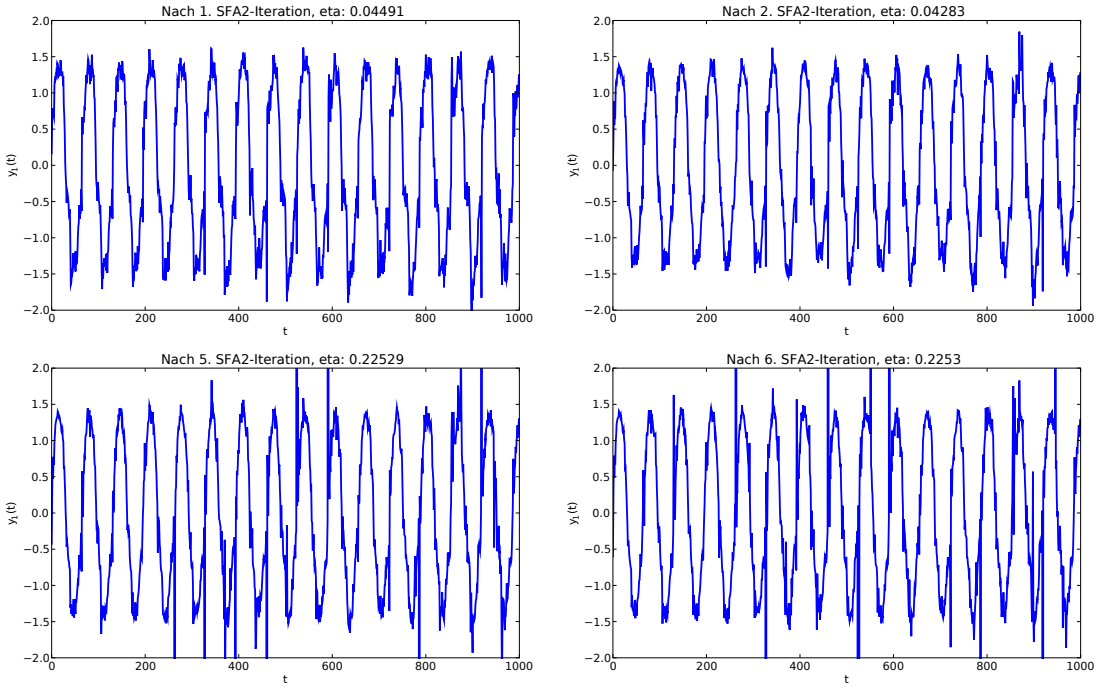


Abbildung 5.7: Generalisierung der SFA: Frontal korrelierte durch die quadratische SFA entdeckte Komponente nach 1, 2, 5 und 6 SFA^2-Iterationen. Die Trainingsdaten bestehen aus einer 60-sekündigen Laufsequenz, die Testdaten stammen bestehen aus einem zehnsekündiger Ausschnitt einer anderen Laufsequenz des gleichen Roboters.

erst bei zusätzlicher Eliminierung der frontalen Fußsensoren ist keinerlei frontale Schwingung mehr zu erkennen.

Generalisierung

In meiner Studienarbeit [Höf09] wurde bereits darauf hingewiesen, dass vor allem die Komponenten zur Posenerkennung stark zur Überanpassung tendieren. Dies liegt vor allen Dingen daran, dass das Umfallen ein unvorhergesehenes Ereignis darstellt, welches nicht gleichförmig wie beispielsweise kontrollierte Aufsteh- aber auch Laufbewegungen ist. Das führt dazu, dass sich jedes Umfallen auch verschieden in den sensorischen Daten niederschlägt.

Allerdings ist tritt auch bei der frontal korrelierten Komponente nach einer höheren Anzahl von Iterationen eine Verschlechterung auf ungesehenen Testdaten auf. Abbildung 5.7 zeigt, dass sich schon bei der zweiten, vor allem aber ab der fünften Iterationen einzelne starke Peaks zeigen. Dieser Effekt konnte auch bei verschiedenen Kombinationen aus Trainings- und Testdaten beobachtet werden und ist wie kurz zuvor erwähnt auf Überanpassung und hohe Potenzen in den höheren Iterationen zurückzuführen. Ebenso sinkt der η-Wert für die frontale Komponente ab der fünften Iteration nicht mehr. Dies spricht jedoch stark gegen die Verwendung dieser Komponente als Steuersignal für das Laufmuster, da starke Einbußen in der Stabilität zu erwarten sind.

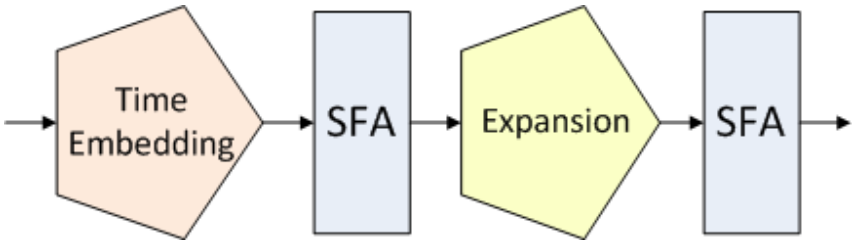


Abbildung 5.8: Verwendetes Modul zur SFA mit zeitlicher Einbettung. Um die Dimensionalität des Signals nach der Expansion gering zu halten, wird nach der zeitlichen Einbettung eine lineare SFA zur Dimensionsreduktion verwendet.

5.2.3 Quadratische SFA mit zeitlicher Einbettung

Wie in Abschnitt 2.5.6 vorgestellt, kann es von Vorteil sein der SFA zusätzlich zeitlich verzögerte Kopien des Eingabesignals zur Verfügung zu stellen. Daher soll nun evaluiert werden, inwiefern sich die Gestalt der erhaltenen Komponenten durch verschiedene SFA-Strukturen mit zeitlicher Einbettung, im folgenden abkürzend als *TE-SFA* (für *time embedded SFA*) bezeichnet, verändert.

Bei der Anwendung der quadratischen SFA auf ein hochdimensionales Eingabesignal, welches zusätzlich zeitlich eingebettet wurde, stellt sich unmittelbar das Problem, dass die kombinatorische Explosion beim Expansionsschritt noch schwerwiegender zum Tragen kommt. Werden beispielsweise Daten von 16 Beschleunigungssensoren um nur zwei zusätzliche Schritte verzögert, also der Tap-Delay $m = 3$ gesetzt, ist das neue Eingabesignal 48-dimensional und wird durch die quadratische Expansion auf bereits 1224 Dimension aufgebläht. Um dies zu vermeiden, sind verschiedene Möglichkeiten denkbar:

Eingeschränkte Anzahl von Sensoren Anstatt alle 16 Beschleunigungssensoren zu verwenden, wird beispielsweise nur eine Auswahl von Sensoren verwendet.

Geringer Tap-Delay Zusätzlich zur Einschränkung der Sensoren kann der Tap-Delay m verringert werden.

Dimensionsreduktion vor der Expansion Bevor die quadratische SFA durchgeführt wird, wird das zeitlich eingebettete Signal zuvor durch eine lineare SFA auf 24 Dimensionen reduziert.

Eine Problematik dieser Variante ist allerdings, dass die gelernten Koeffizienten der quadratischen SFA nicht mehr direkt analysiert werden können.

Um die Problematik des Fluchs der Dimensionalität zu vermeiden, werden im folgenden lediglich die Schulter- und Fußsensoren (acht Sensorwerte insgesamt) sowie ein Tap-Delay von $m = 8$ verwendet. Zudem wird vor der Expansion eine Dimensionsreduktion mittels SFA durchgeführt. Das gesamte Schema ist in Abbildung 5.8 zu sehen. Die η-Werte sowie visuelle Bewertung der so erhaltenen langsamen Komponenten zeigen, dass die Ergebnisse sich gegenüber der nicht einfachen, aber hochdimensionalen SFA-Anwendung kaum verschlechtern. Jedoch wird dadurch der Berechnungsaufwand deutlich verringert. Wie weiter unten beschrieben, kommt diese Struktur

auch in der sensomotorischen Schleife zum Einsatz. Nur zur genaueren Betrachtung der Koeffizienten der quadratischen SFA wird die Dimensionsreduktion weggelassen. Wie zuvor werden Trainingssequenzen analysiert, welche durch das Netz ohne Filter und das IIR-Netz generiert wurden.

Frontale Komponente

Bei der SFA mit zeitlicher Einbettung zeigen sich ähnliche Ergebnisse wie zuvor, nämlich dass aus reinen Laufsequenzen die frontale Pendelbewegung als langsamste Komponente extrahiert wird. Jedoch zeigt Abbildung 5.9, dass bei Erhöhung von m andere Komponenten gefunden werden, genauer gesagt eine langsamere Sinusschwingung. Die Vermutung liegt nahe, dass sich diese langsamere Sinusschwingung bei $m = 16$ durch Addition mehrerer zeitlich verschobener Schwingungen ergibt. Da diese Komponente allerdings nicht hilfreich im Hinblick auf die Verwendung der SFA in der Generierung des Laufmusters ist, wird keine weitere Untersuchung dieser unternommen. Der der an dieser Stelle auftretende Effekt wurde bereits in der Arbeit [KK09], welche bereits in Abschnitt 2.6.3 erwähnt wurde, beobachtet.

Es lässt sich sagen, dass die Parameter τ und m einen ähnlichen Einfluss auf die Ergebnisse haben wie bei der quadratischen SFA die Anzahl der Iterationen sowie die Anzahl der weitergereichten Komponenten je Iteration: Die Erhöhung von m verbessert den η-Wert des Signals deutlich, während die Vergrößerung des Abstandes τ diesen eher leicht verschlechtert. Die jeweilige zum frontalen Sensor ähnlichste Komponente nimmt bzgl. ihrer Korrelation vor allem mit Vergrößerung von τ ab, was an der erhöhten Glättung des Signals liegt.

Eine interessante Beobachtung ist, dass auch für die Variante des Laufmusters, welche ohne Filterung auskommt, von der SFA mit zeitlicher Einbettung eine glatte frontale Komponente extrahiert werden kann. Abbildung 5.10 zeigt, dass die optimale Komponente bei $m = 8$ (links unten) gefunden wird. Der η-Wert dieser Komponente liegt mit 0.0154 auch nah an dem Wert der optimalen Komponente des IIR-Netzes, welche bei $m = 4$ gefunden wird und bei 0.0158 liegt (Abbildung 5.9, rechts oben). Bei höherem m taucht die frontale Komponente nicht mehr klar erkennbar unter den langsamsten Komponenten auf, auch hier wird eine vermeintliche Regularität in dem Laufmuster gefunden. Es ist nicht klar, um was für eine Regularität es sich an dieser Stelle handelt, auch die Korrelationskoeffizienten leisten keine Hilfestellung, um die gefundenen Komponenten zu deuten. Eine mögliche Vermutung ist, dass leichte Schwankungen bzgl. der Intensität während des Laufens, also die Amplitudenveränderung in der Pendelbewegung, extrahiert werden, oder sich durch die Kombination der zeitlich verzögerten Signale zufällige Regularitäten ergeben.

Analyse der frontalen Komponente

Wenden wir uns nun der Frage zu, warum die quadratische SFA mit zeitlicher Einbettung eine gute Glättung der frontalen Pendelbewegung liefert.

Auch hier gestaltet sich die Analyse schwierig, da das zeitlich eingebettete Signal eine weitaus höhere Dimensionalität als das Originalsignal aufweist. Zu diesem Zweck wurde in den bisheri-

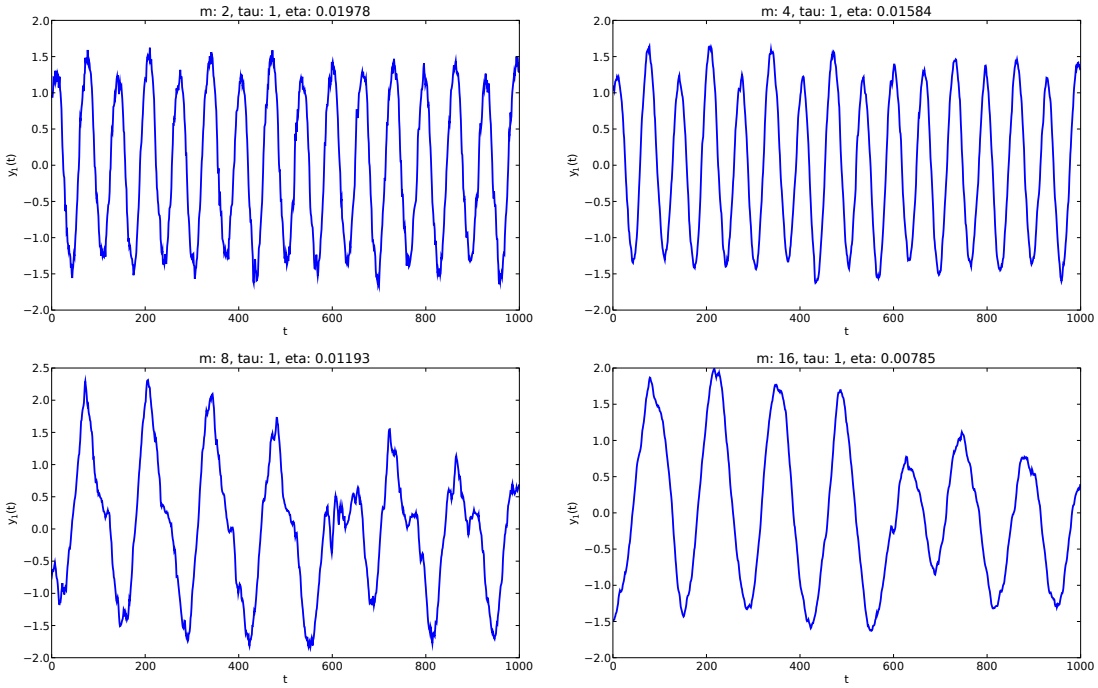


Abbildung 5.9: Frontal korrelierte durch die quadratische TE-SFA entdeckte Komponente. Die Trainingsdaten bestehen aus einer 60-sekündigen Laufsequenz des IIR-Netzes, es ist ein zehnsekündiger Ausschnitt aus dieser dargestellt.

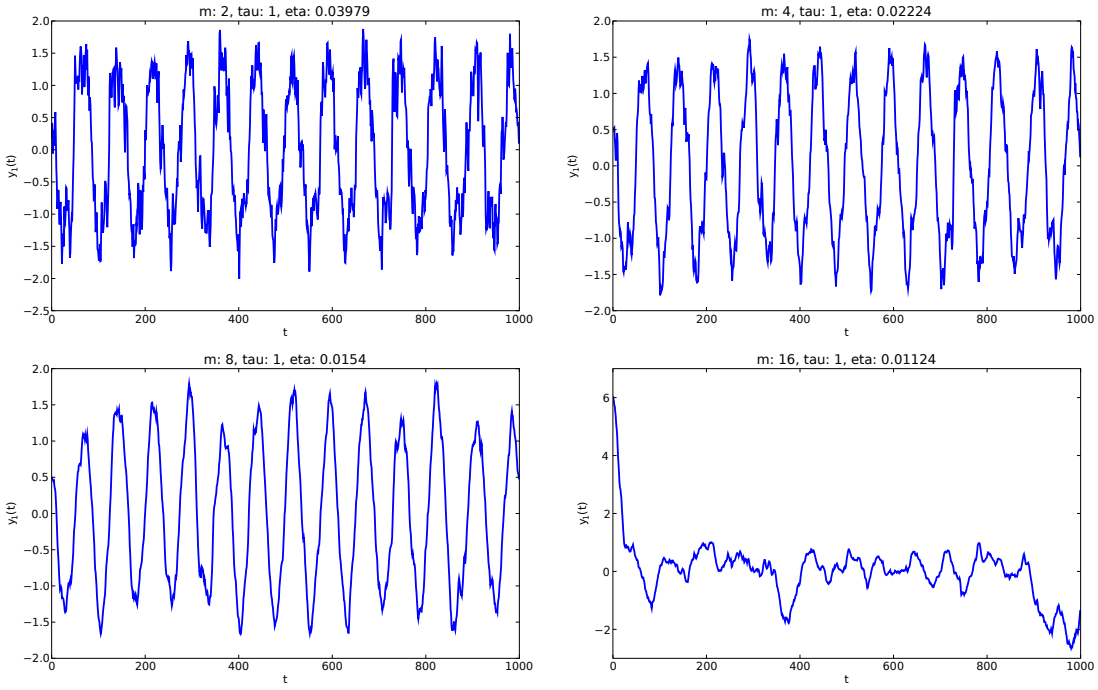


Abbildung 5.10: Quadratische TE-SFA angewandt auf ein Trainingssignal, welches durch das Netz ohne Filter generiert wurde: Frontal korrelierte durch die quadratische SFA entdeckte Komponente bei $m = 2, 4, 8, 16$. Die Trainingsdaten bestehen aus einer 60-sekündigen Laufsequenz, es ist ein zehnsekündiger Ausschnitt aus dieser dargestellt.

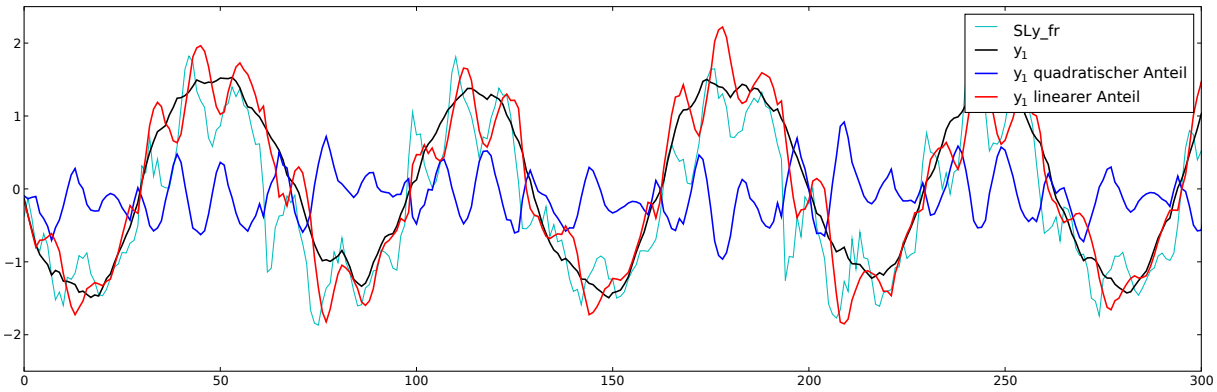


Abbildung 5.11: Quadratischer und linearer Anteil der langsamsten Komponente der SFA mit zeitlicher Einbettung. Zum Vergleich die langsamste Komponente selbst sowie ein frontaler Schultersensor.

gen Auswertungen der quadratischen SFA aus rechentechnischen Gründen eine lineare SFA vorangestellt; dies verhindert allerdings die Untersuchung des direkten Zusammenhangs zwischen Sensoren und der quadratischen SFA-Komponente.

Um die kombinatorische Explosion bei der Expansion gering zu halten, wird daher nur eine quadratische SFA betrachtet, welche auf die acht Beschleunigungswerte von Schultern und Füßen mit einer zeitlichen Einbettung von $m = 4$ (τ=1) angewandt wird. Somit liegt die Dimension des eingebetteten Eingabesignals bei nur 32 und kann direkt analysiert werden. Dem liegt die Hypothese zugrunde, dass auch bei einer TE-SFA mit Eingabedaten geringerer Dimension und kleinerem tap delay die gleichen Regularitäten zu beobachten sind wie bei einer TE-SFA, welche auf einer längeren zeitlichen Einbettung und mehr Sensoren operiert.

Abbildung 5.11 zeigt, welche Rolle der lineare und der quadratische Anteil bei der frontalen Komponente mit zeitlicher Einbettung spielen. Zu sehen ist die langsamste Komponente, welche durch ein IIR-Netz generiert wurde, sowie der lineare und quadratische Anteil der Komponente. Bei der Berechnung der quadratischen Form wird wie zuvor der Mittelwert der Eingabedaten für den Parameter r verwendet. Im Gegensatz zur reinen quadratischen SFA ist hier tatsächlich eine Trennung in Statik und Dynamik zu sehen: Während die lineare Komponente die Oszillation repräsentiert, reagiert die quadratische Komponente auf Störungen; in Summe ergeben die beiden Parts ein weniger störbehaftetes, langsam oszillierendes Signal. Dies ist auch zu beobachten, wenn als Eingabedaten eine Sequenz verwendet wird, welches durch das Netz ohne Filter generiert wurde.

In Abbildung 5.12 ist zu sehen, welchen Einfluss die verschiedenen Sensoren tatsächlich haben. Dazu werden im Ausführungsschritt gezielt nur Untermengen der für das Training verwendeten Sensoren gebildet und der TE-SFA übergeben. Alle anderen Sensorwerte werden auf Null gesetzt. In der Abbildung steht die blaue Kurve jeweils für die auf der verminderten Sensordatenmenge berechnete langsamste Komponente, die rote Kurve stellt die eigentliche langsamste Komponente dar. In der oberen Reihe sind dabei zunächst die langsamsten Komponenten aufgetragen, welche durch Elimination von jeweils vier Sensoren entstehen: Nur frontale Sensoren, nur sagittale Sensoren, nur Schultersensoren sowie nur Fußsensoren (von links nach rechts). In der unteren Reihe

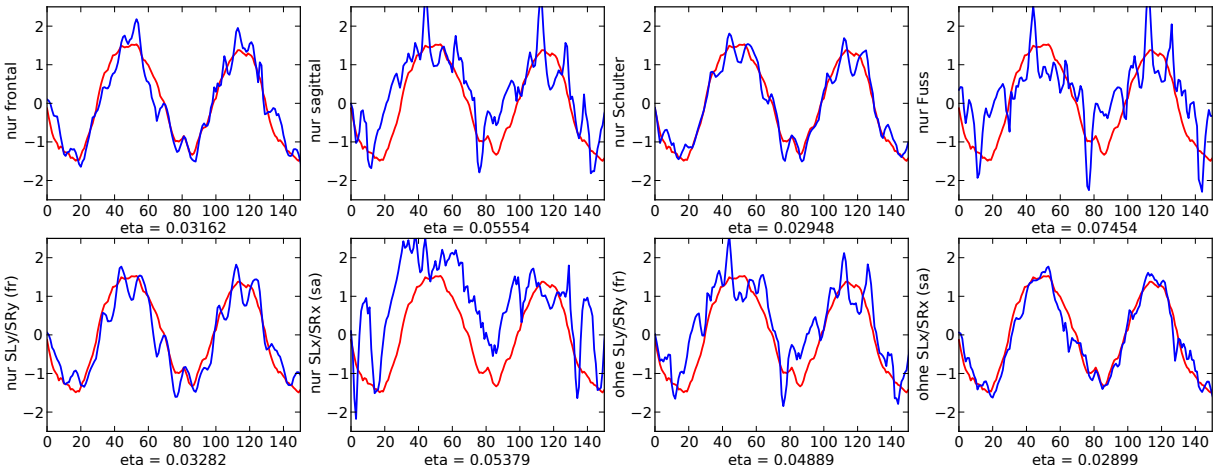


Abbildung 5.12: TE-SFA-Signal der langsamsten Komponente, das entsteht, wenn gezielt Sensoren eliminiert werden. Zum Vergleich stehen unter jeder Kurve die η-Werte der manipulierten langsamsten Komponenten angegeben. Der η-Wert der ursprünglichen langsamsten Komponente (rote Kurve) beträgt 0.0178.

sind zunächst die Komponenten mit Beibehaltung von nur zwei Sensoren, dann die, bei denen jeweils genau diese zwei Sensoren eliminiert werden, zu sehen. Dabei wird die Abhängigkeit der Komponente von den frontalen bzw. sagittalen Schultersensorpaaren betrachtet.

Wie gut zu erkennen ist, weisen die Komponenten, die vornehmlich frontale oder Schultersensoren verwenden, die größte Ähnlichkeit mit dem Originalsignal auf. Daraus lässt sich schließen, dass die Schultersensoren insgesamt einen größeren Einfluss auf die Generierung eines glatten Signals haben, ebenso wie die frontalen Sensoren stärker an der Formung des Signals als die sagittalen beteiligt sind.

Die gleichen Tendenzen lassen sich auch bei Verwendung einer Laufsequenz, welche durch das Netz ohne Filter generiert wurde, erkennen.

Generalisierung

Damit die TE-SFA-Komponente in einer sensomotorischen Schleife verwendet werden kann, muss sie sehr gut auf unvorhergesehene Laufbewegungen generalisieren, aber vor allem robust gegenüber Störungen sein. Abbildung 5.13 zeigt, dass die TE-SFA ein fast genauso gutes Steuersignal aus einer Laufsequenz des Netzes ohne Filter wie aus einer Laufsequenz des IIR-Netzes extrahieren kann. Zudem ist die Generalisierungsfähigkeit der Komponenten stark von der Länge der Trainingsdaten abhängig. Bis zu einer Länge von 45 Sekunden verbessert sich die Langsamkeit der Komponente bzgl. des η-Wertes stetig, danach ist keine weitere Verbesserung zu erkennen. Dass die η-Werte der Netzes ohne Filter generell etwas höher sind, liegt daran, dass die Trainingsdaten auch von der Laufbewegung her insgesamt deutlich unruhiger sind. Beim IIR-Netz zeigt sich desweiteren, dass das optimale frontale Steuersignal, d. h. das Signal mit dem höchsten Korrelationskoeffizienten zu den frontalen Schultersensoren, eher in der zweitlangsamsten Komponente zu finden ist, während es bei dem Netz ohne Filter immer in y_1 vorliegt.

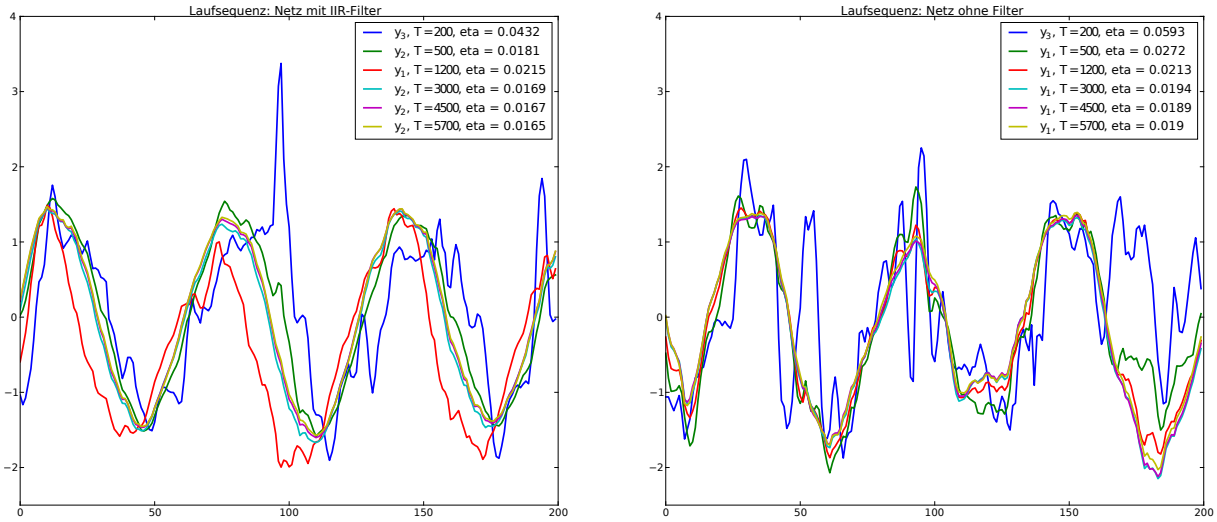


Abbildung 5.13: Generalisierung der TE-SFA-Komponente abhängig von der Länge der Trainingsdaten. Links: IIR-Netz. Rechts: Netz ohne Filter.

5.3 SFA als Filter in einer sensomotorischen Schleife

Die letzten Abschnitte haben gezeigt, dass nur die SFA mit zeitlicher Einbettung in Frage kommt, um ein Steuersignal für eine sensomotorische Schleife zu generieren. Sie bietet zudem den Vorteil, dass sie weniger rechenintensiv ist, da zwei SFA-Einheiten genügen, um bessere Ergebnisse als mit iterierten quadratischen SFA ohne zeitliche Einbettung zu erzielen. Allerdings ist eine Zeitverzögerung nicht auszuschließen, welche die Reaktivität der Komponente beeinträchtigen könnte.

Als nächstes möchte ich mich der Frage zuwenden, wie die im letzten Abschnitt beschriebene SFA-Komponente dazu genutzt werden kann, in einer sensomotorischen Schleife zum Einsatz zu kommen. Die neuronalen Strukturen, welche das Laufen generieren, wurden bereits am Anfang des Kapitels vorgestellt und werden nun unter Anwendung der SFA modifiziert. Daraufhin wird betrachtet, ob die Stabilität der Komponente erhalten bleibt und inwiefern sich die Reaktivität der gesamten Systems erhöht.

5.3.1 SFA-Filtermodul

Zur Filterung des Beschleunigungsdatensignals wird das neuronale Netz, welches in Abschnitt 5.1 beschrieben wurde, wie in Abbildung 5.14 zu sehen modifiziert: Zunächst werden nicht mehr nur die beiden frontalen Schultersensoren, sondern alle Schulter- sowie alle Fußsensoren (`SLx_fr`, `SLy_sa`, `SRx_fr`, `SRy_sa`, `FLx_sa`, `FLy_fr`, `FRx_sa`, `FRy_fr`) verwendet. Diese werden in das zuvor vorgestellte TE-SFA-Modul geleitet. Die langsamste Komponente wird durch ein Gewicht von 0.1 angepasst und wie zuvor auf die Motoren gegeben. In einigen Fällen muss zusätzlich das Vorzeichen der verwendeten langsamsten TE-SFA umgedreht werden, da, wie bereits erwähnt, die SFA keine Präferenz bzgl. des Vorzeichens der Lösung hat. In diesem Fall wird die SFA-Komponente durch ein Gewicht von -0.1 angepasst.

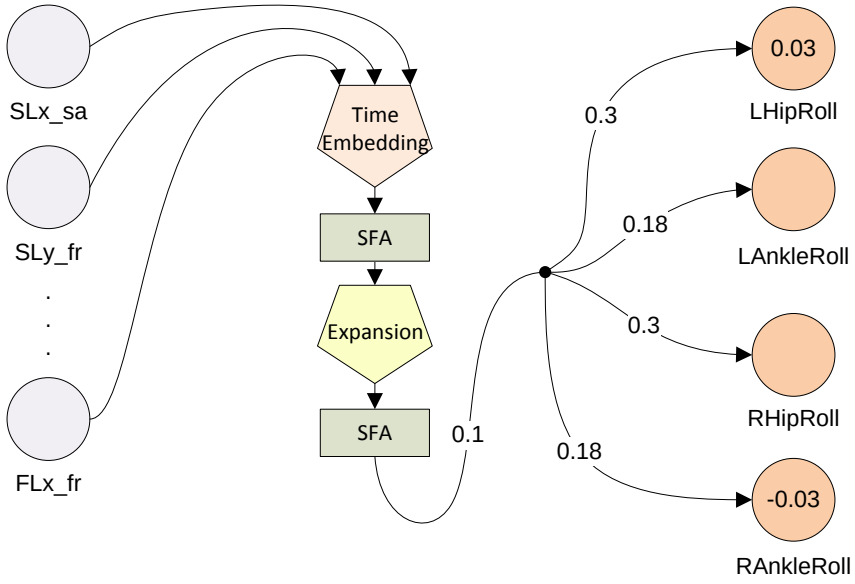


Abbildung 5.14: Komponente des neuronalen Netzes, welche die zum Laufen notwendige stabile Schwingung in der frontalen Ebene generiert.

Bevor das SFA-Modul allerdings verwendet werden kann, muss es offline trainiert werden. Wie im letzten Abschnitt beschrieben, sind längere Trainingssequenzen sinnvoll. Daher wird eine 60-sekündige Laufsequenz verwendet, welche mit dem Netz ohne Filter generiert wurde, d. h. keinerlei Filterung der Eingabedaten verwendet. Werden Trainingsdaten von einer Laufsequenz des IIR-Netzes verwendet, so gelingt es nicht auf Anhieb, eine stabile Laufsequenz zu generieren, weswegen diese Variante hier nicht berücksichtigt wird.

Desweiteren ist zu erwähnen, dass als nichtlineare Expansionen sowohl eine quadratische als auch die tanh-Expansion aus Abschnitt zum Einsatz kommen können, jedoch bzgl. der SFA-Komponenten und der Laufbewegung kein qualitativer Unterschied zu beobachten ist. Wird die tanh-Expansion verwendet, so bedeutet dies, dass die Steuerung des Roboters vollständig durch eine neuronale Struktur erfolgen kann. Daher wird im folgenden ausschließlich die vorgestellte tanh-Expansion verwendet.

5.3.2 Alternative Filtermodule

Um die Qualität des durch die SFA generierten Laufens zu evaluieren, werden weitere Laufsequenzen generiert, welche mittels modifizierter Netze realisiert werden. Folgende Typen sind für einen Vergleich interessant:

Kein Filter Als Grundversion der sensomotorischen Schleife wird das Netz aus Abbildung 5.2a verwendet.

IIR-Filter Als stabile Grundversion wird das Netz aus Abbildung 5.2b verwendet.

FIR-Filter Aus dem IIR-Filter wird mit der Methode aus Abschnitt 3.3.1 ein FIR-Filter generiert. Dabei muss das Tap-Delay auf $m = 50$ gesetzt werden, da sonst kein stabiles Laufen

zustande kommt.

Volterra-Filter Um das SFA-Modul mit einem weiteren quadratischen Filter zu vergleichen, wird ein Volterra-Filter zweiter Ordnung verwendet, welches in Abschnitt 3.3.3 vorgestellt wurde. Um ein stabiles Laufmuster zu generieren, muss der Tap-Delay auf $m = 22$ erhöht werden, und die Eingabesensoren müssen auf `SLx_fr` und `SRx_fr` beschränkt werden.

Zur Adaption des Filters wird der in dem gleichen Abschnitt beschriebene LMS-Algorithmus verwendet. Als Trainingsdaten wird wie für die SFA eine Laufsequenz vom Netz ohne Filter verwendet. Als Zielsignal wird das auf der Laufsequenz berechnete IIR-Filtersignal vorgegeben[1].

5.3.3 Stabilität

Als erstes soll evaluiert werden, ob die verwendeten Filter eine hinreichende gute Stabilität der Laufbewegung gewährleisten. Da alle Filter so optimiert wurden, dass der Roboter 60 Sekunden ohne umzufallen laufen kann, müssen weitere Charakteristiken der generierten Laufbewegungen untersucht werden.

Abbildung 5.15 zeigt 30-sekündige Ausschnitte aus den Laufsequenzen, die mit dem jeweiligen neuronalen Netz generiert wurden. Auf den ersten Blick ist zu erkennen, dass vor allem die FIR- sowie die Volterra-Lösung größere Störungen aufweisen als die anderen Laufmuster. Auch bei weiteren Versuchen wiesen vor allem die IIR- sowie die TE-SFA-Lösung die geringste Störungsanfälligkeit auf.

Zunächst fällt auf, dass die Laufbewegungen sich in der Frequenz der frontalen Oszillation unterscheiden. Damit hängt auch die Amplitude zusammen, d. h. wie weit der Roboter sich zur Seite neigt. Um die durchschnittliche Frequenz und Amplitude der Laufbewegungen zu ermitteln, wird wie in [Wer08] der gewichtete Mittelwert der beiden frontalen Schultersensoren betrachtet. Dazu wird mittels *Welch's Methode* [Wel67] die *spektrale Leistungsdichte* (auch *Power-Spektrum-Dichte*) der verschiedenen Laufbewegungen ermittelt. Welch's Methode beruht darauf, eine stückweise Fourieranalyse der Autokorrelation eines Signals durchzuführen, womit sich eine erwartungstreue Schätzung der spektralen Leistungsdichte ergibt. Die Autokorrelation gibt die Selbstähnlichkeit des Signals über die Zeit gesehen an und ist für ein eindimensionales, zeitdiskretes Signal $x(t)$ gegeben durch den Korrelationskoeffizienten des Signals mit einer zeitverzögerten Version des gleichen Signals. Der Korrelationskoeffizient für zeitdiskrete Signale wurde in Abschnitt 2.4.2 wie folgt definiert:

$$\rho = \frac{\langle (x_1 - \langle x_1 \rangle)(x_2 - \langle x_2 \rangle) \rangle}{\sqrt{\langle (x_1 - \langle x_1 \rangle)^2 \rangle} \sqrt{\langle (x_2 - \langle x_2 \rangle)^2 \rangle}} \tag{2.31}$$

Es werden im folgenden nur noch eindimensionale Signale $x(t)$ betrachtet, so dass die Zeitvariable im Index vermerkt werden kann, also $x_t := x(t)$. Zudem soll ohne Beschränkung der Allgemeinheit

[1] Eine Untersuchung könnte sein, ob der LMS-Algorithmus imstande ist, die gleichen Gewichte wie die SFA zu lernen, wenn die langsamste SFA-Komponente als Zielsignal vorgegeben wird. Allerdings lieferte der LMS-Algorithmus keine guten Ergebnisse, wenn so viele Sensoren wie bei der SFA integriert wurden, weswegen dieser Vergleich nicht weiter in Erwägung gezogen wurde.

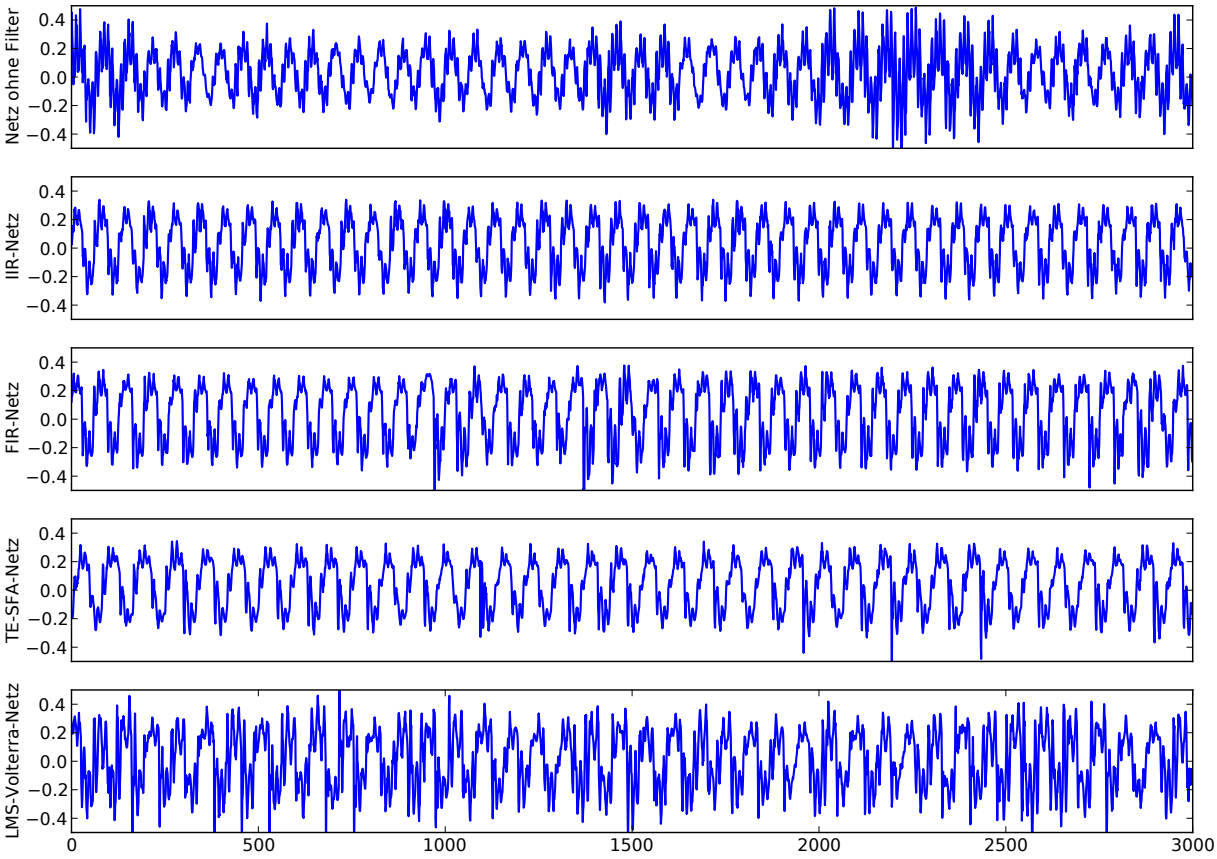


Abbildung 5.15: Laufsequenzen, die durch die vier verschiedenen Netze generiert wurden. Es sind jeweils 30-sekündige Ausschnitte aus einer insgesamt 60-sekündigen Sequenz zu sehen.

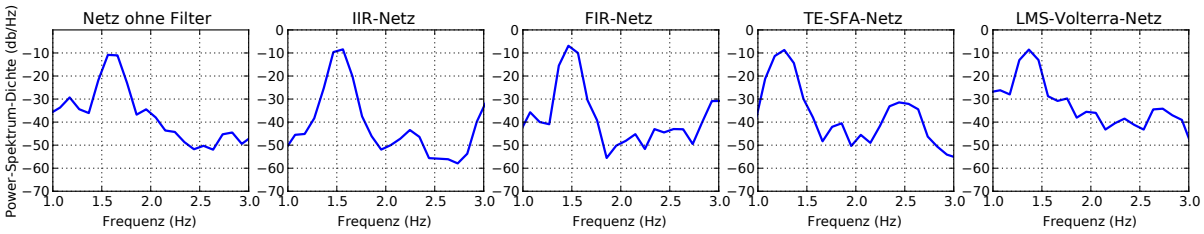


Abbildung 5.16: Power-Dichte-Spektrum der Summe der lateralen Schultersensoren für die betrachteten Laufmuster.

davon ausgegangen werden, dass das Signal mittelwertzentriert ist. Die Autokorrelationsfunktion eines Signals ergibt sich dann durch

$$R_{xx}(\tau) = \frac{\langle x_t x_{t+\tau} \rangle}{\sqrt{\langle x_t^2 \rangle \langle x_{t+\tau}^2 \rangle}}. \tag{5.1}$$

Für ein festes τ gibt $R_{xx}(\tau)$ an, wie sehr sich x_t nach τ Zeitschritten wiederholt. Durch eine Fourieranalyse von $R_{xx}(\tau)$ lässt sich dominierende Frequenzkomponente und somit die Periodizität von x_t ermitteln.

Abbildung 5.16 zeigt das Ergebnis nach Anwendung von Welch's Methode, die Frequenzen mit den jeweils höchsten Amplituden sind in Tabelle 5.1 zu sehen. Die Samplingfrequenz der Sensordaten beträgt 100 Hz, es wurde eine Fenstergröße von 1024 gewählt.

Es ist zunächst deutlich zu erkennen, dass die linearen Methoden allesamt eine höhere Grundfrequenz aufweisen als die TE-SFA sowie das Volterra-Filter. Während die durchschnittliche Frequenz des TE-SFA-Laufmusters bei 1.17 Hz bis 1.36 Hz liegt, was einer Periodenlänge von 0.73 s bis 0.78 s entspricht, liegt das durch das Volterra-Filter generierte Laufmuster bei durchschnittlich 1.36 Hz (Periodenlänge 0.73 s). Die linearen Verfahren hingegen erreichen Frequenzen von 1.46 Hz bis 1.56 Hz. Alle Verfahren weisen eine ähnlich hohe Amplitude auf, die vergleichsweise hohe Amplitude von -6.88 db/Hz für das Laufen mittels IIR-Filter lässt sich dadurch erklären, dass, wie in Abbildung 5.16 zu sehen, die Spitze bei ca. 1.5 Hz deutlich schmaler als bei den anderen Verfahren ist, was zugleich bedeutet, dass die Frequenz beim IIR-Filter weniger stark variiert. Am stärksten scheint diese Variation der Grundschwingung bei der TE-SFA zu sein, wie sich an dem Plateau um 1.27 Hz erkennen lässt.

	Frequenz (Hz)	Periodenlänge (s)	Amplitude (db/Hz)	$\bar{\eta}$
Kein Filter	1.56	0.64	-10.83	3.81
IIR	1.56	0.64	-6.88	3.32
FIR	1.46	0.68	-8.44	3.51
TE-SFA	1.27	0.79	-8.71	3.61
LMS-Volterra	1.36	0.73	-8.54	4.89

Tabelle 5.1: Frequenzen sowie Amplituden der jeweiligen frontalen Grundschwingungen der verschiedenen Laufmuster sowie deren $\bar{\eta}$-Werte (siehe Text).

Ein weiteres mögliches Kriterium, um die Stabilität der Laufmuster zu vergleichen, wäre der η-Wert. Allerdings kann dieser nicht ohne weiteres auf die aufgenommenen Sequenzen angewandt werden, da die jeweilige Frequenz eines Laufmusters einen großen Einfluss auf den resultierenden η-Wert hat. Daher hat tatsächlich das auf der TE-SFA beruhende Laufmuster den geringsten η-Wert, wenn die verschiedenen Sequenzen direkt miteinander verglichen werden. Um diese Bevorteilung der TE-SFA zu beseitigen, wird der η-Wert mit der Periodenlänge gewichtet bzw. durch die Grundfrequenz des Laufmusters geteilt. Dieser Wert wird als $\bar{\eta}$ bezeichnet:

$$\bar{\eta}(x) := \frac{1}{f}\eta(x), \tag{5.2}$$

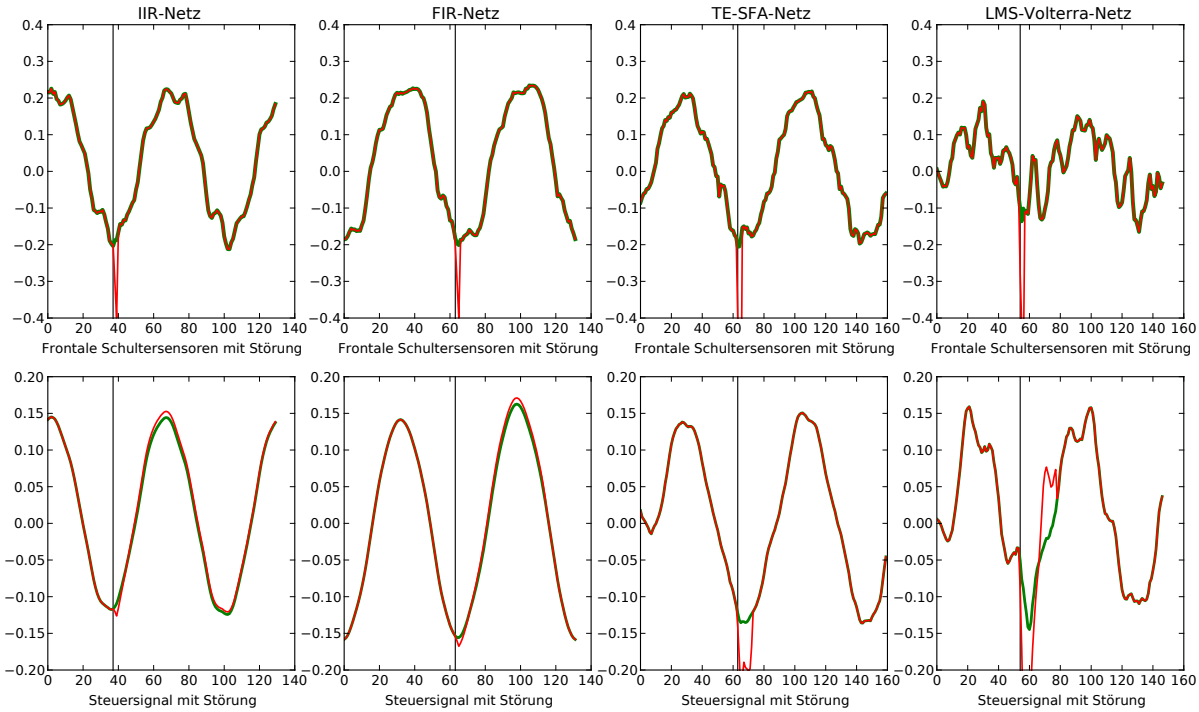


Abbildung 5.17: Simulierte kurze Störung der sensorischen Werte der Laufsequenzen. **Oben**: Die jeweiligen Sensorwerte der frontalen Schultersensoren (grün), sowie der Störwert (rot). **Unten**: Das aus den oberen Sensorwerten generierte Steuersignal ohne (grün) und mit (rot) Störung.

wobei f für die Grundfrequenz des (periodischen) Signals x steht.

Wie Tabelle 5.1 zeigt, generiert das IIR-Netz nach diesem Kriterium wie erwartet das stabilste Laufmuster. Mit einigem Abstand folgen das FIR-Netz sowie das TE-SFA-Netz, mit etwas Abstand vor dem Netz ohne Filter. Deutlich abgeschlagen ist die LMS-Volterra-Filter-Lösung. Möglicherweise könnte die Verwendung einer rekursiven Variante des LMS-Algorithmus (*Recursive Least Squares (RLS)*, siehe [Zak05], Abschnitte 8.3 und 10.5) das Volterra-Filter verbessern, jedoch liegt der Fokus dieser Arbeit auf die Optimierung des TE-SFA-Netzes.

Zu beachten ist zusätzlich, dass die exakten Amplituden der Signale nicht ins Gewicht fallen, da alle Signale vor der Berechnung des $\bar{\eta}$-Wertes auf einen Mittelwert von Null sowie eine Varianz von Eins skaliert werden. Somit lässt sich konstatieren, dass das TE-SFA-Netz zwar nicht die Stabilität der linearen Filterlösungen aufweist, jedoch keine zu starken Einbußen hinnehmen muss.

5.3.4 Reaktivität

Nachdem die Stabilität des TE-SFA-Laufmusters nachgewiesen wurde, muss gezeigt werden, dass das TE-SFA-Netz tatsächlich auch reaktiver ist. Dabei ist reaktiv so zu verstehen, dass eine deutliche Reaktion im Steuersignal sichtbar sein muss, sobald die Laufbewegung essentiell gestört wird.

Es gestaltet sich jedoch als schwierig, eine solche Störung praktisch durchzuführen, da ein manuelles Anstoßen des Roboters sich nicht für jede Laufsequenz auf die exakt gleiche Weise reproduzieren lässt, was einen fairen Vergleich der Reaktivität der Netze unmöglich macht. Stattdessen soll daher eine Störung simuliert werden, indem die Sensordaten an einer Stelle der Laufsequenz gezielt manipuliert werden.

Als Störung wird ein kurzer, dreischrittiger Impuls gewählt, der einen Schlag in die Richtung, in welche der Roboter gerade geneigt ist, repräsentiert. Mit anderen Worten, wenn der Roboter sich zur rechten Seite neigt, wird der Impuls so gewählt, dass er einen starken Stoß des Roboter nach rechts simuliert. Der Impuls wird mit drei Zeitschritten relativ kurz gewählt und setzt die beiden Schultersensoren auf die Werte $[0.2, 0.3, 0.4]$ für einen Stoß nach links bzw. $[-0.2, -0.3, -0.4]$ für einen Stoß nach rechts. Um zudem unabhängig von einer konkreten Periode des Signals zu sein, werden die Sensorsignale aller Laufmuster zunächst gemittelt, um so eine prototypische Periode zu erhalten. Auf dieses prototypische Signal wird dann die simulierte Störung angewandt. Die Simulation kann mit dem gleichen Ergebnis jedoch auch auf ein nicht gemitteltes Signal angewandt werden.

Abbildung 5.17 zeigt die Störung sowie ihren Niederschlag im jeweiligen Steuersignal. Die oberen Graphen zeigen für jedes Laufmuster den Mittelwert der beiden Schultersensoren ohne (grün) sowie mit Störimpuls (rot). Die unteren Graphen zeigen das Steuersignal, das von den jeweiligen Netzen generiert wird, ebenfalls ohne (grün) und mit (rot) Störung. Es ist deutlich zu sehen, dass das IIR- sowie das FIR-Filter eine sehr schwache Reaktion zeigen. Beim IIR-Filter macht sich zudem die Rückkoppelung erkennbar, da das gestörte Steuersignal sogar noch nach über 60 Zeitschritten leicht von dem ungestörten Steuersignal abweicht, während dies bei den restlichen Filtern aufgrund der beschränkten Zeiteinbettung nicht der Fall ist. Die beiden nichtlinearen Filter hingegen weisen trotz der kurzen und nicht besonders erheblichen Störung eine deutliche Reaktion auf. Interessanterweise genügt es für die TE-SFA und das Volterra-Filter, nur die Schultersensoren zu stören, um eine derart starke Reaktion hervorzurufen, jedoch ist eine ähnliche Reaktion zu sehen, wenn der Störimpuls auf alle Sensoren gegeben wird. Ebenso ist die deutliche Reaktion des TE-SFA-Netzes keiner quadratischen Expansion zuzuschreiben, da bei der TE-SFA wie zuvor erwähnt eine tanh-Expansion verwendet wird.

5.4 Zusammenfassung

In diesem Kapitel konnte gezeigt werden, dass sich die SFA dafür eignet, ein geeignetes Steuersignal für ein zweibeiniges Laufmuster zu generieren. Im ersten Teil des Kapitels wurde zunächst das neuronale Netz beschrieben, welches bisher in der sensomotorischen Schleife zur Generierung des Laufmusters verwendet wurde. Daraufhin wurde analysiert, welche Komponenten die SFA aus einem Sensordatenstrom, welcher dieses Laufmuster repräsentiert, extrahieren kann. Es konnte demonstriert werden, dass die SFA ein wichtiges Charakteristikum, nämlich die Schwingung in der frontalen Ebene extrahiert. Als positiven Nebeneffekt glättet die SFA diese Schwingung zudem deutlich. Allerdings zeigte sich, dass eine SFA mit zeitlicher Einbettung verwendet werden muss, damit die extrahierten Komponenten auch hinreichend gut generalisieren. Zudem müssen

bei der Verwendung der SFA mit zeitlicher Einbettung nicht mehrere SFA-Iterationen verwendet werden, um eine glatte Komponente zu extrahieren.

Im zweiten Teil des Kapitels wurden verschiedene alternative neuronale Netze zur Generierung des Steuersignals für das Laufmuster vorgeschlagen. Unter anderem wurde auch ein Netz auf Basis der SFA mit zeitlicher Einbettung betrachtet. Es konnte gezeigt werden, dass das SFA-Netz hinsichtlich Stabilität des Laufmusters mit den IIR- sowie FIR-Filter-Netzen mithalten kann, auch wenn das Laufen eine geringere Dynamik und niedrigere Frequenz hat. Jedoch übertrifft das SFA-Netz die anderen Filter-Ansätze hinsichtlich Reaktivität, wie durch Analyse einer simulierten Störung des Laufmusters gezeigt werden konnte. Zudem ist von Vorteil, dass die SFA-Lösung viele Sensoren integrieren kann, während die anderen Filtermethoden nur auf ein- bzw. niedrigdimensionalen Eingangsdaten angewandt werden konnten. Es kann zum einen davon ausgegangen werden, dass dies die Robustheit des Steuersignals erhöht, zum anderen aber auch, dass eine Abhängigkeit des Steuersignals vom Gesamtzustand des Roboters herstellt wird. Somit kann die sensorische Situation für die Generierung des Steuersignals besser einbezogen und erfasst werden.

Kapitel 6

Zusammenfassung und Ausblick

Diese Arbeit hat gezeigt, auf welche Weise und für welche Ziele die Slow Feature Analysis (SFA) in der Robotik für die Verarbeitung sensorischer Daten und für die motorische Steuerung zur Anwendung kommen kann. Die SFA wurde von verschiedenen theoretischen wie praktischen Gesichtspunkten beleuchtet, und es wurden konkrete Anwendungen auf einem humanoiden Roboter des Labors für Neurorobotik vorgestellt. Dies ist die erste Arbeit, welche in diesem Umfang die Einsatzmöglichkeiten der SFA unter Verwendung von nicht-visuellen Sensordaten aufzeigt, und insbesondere auch ihre Anwendbarkeit in einer sensomotorischen Schleife demonstriert.

Im ersten Teil der Arbeit wurden grundlegende Ideen des Lernprinzips der Langsamkeit, welches durch die SFA implementiert wird, sowie der Algorithmus der SFA selbst behandelt. Es wurden sowohl Varianten und Erweiterungen der SFA, als auch zur SFA ähnliche und verwandte Verfahren aufgezeigt. Es wurden Standardmethoden zur Analyse der SFA-Lösungen vorgestellt, aber auch eine einfache Form der Analyse statischer SFA-Komponenten mittels Quadriken entwickelt und präsentiert.

Um die SFA direkt auf der verfügbaren Roboterplattform einsetzen zu können und die Eignung der SFA als Verfahren der kognitiven Robotik zu unterstreichen, wurde sie in einem Modell für künstliche neuronale Netze formuliert. Um die Ausführung der SFA zusätzlich zu beschleunigen, wurde eine alternative nichtlineare Expansion entwickelt, welche die Nichtlinearität der Transferfunktion des verwendeten Neuronenmodells, im konkreten Fall des tangens hyperbolicus, ausnutzt. Die Expansion kann auf der Roboterplattform effizienter als die quadratische Expansion berechnet werden, und es wurde an verschiedenen Anwendungsbeispielen gezeigt, dass sie sich als Alternative zur quadratischen Expansion eignet.

Der zweite Teil der Arbeit hat sich mit zwei konkreten Anwendungen der SFA für die humanoide Robotik befasst. Zunächst wurde ihre Anwendbarkeit auf statische Daten betrachtet, welche eine Bewegungssequenz eines Roboters kodieren, in welcher er verschiedene Posen ausführt. Es wurde zum einen gezeigt, dass mittels der SFA die Posen gut unterscheidbar extrahiert werden können. Zum anderen ließ sich beobachten, dass die SFA sich auch als Verfahren zur Dimensionsreduktion eignet, da auch Zwischenzustände, bezogen auf die Daten Übergänge zwischen zwei Posen, sinnvoll auf einen zweidimensionalen Raum abgebildet werden konnten.

Als letztes wurde die SFA auf eine dynamische Bewegungssequenz angewandt und als Filter in

einer sensomotorischen Schleife eingesetzt. Es wurde beschrieben, dass und wie die quadratische SFA bei einer Laufbewegung des Roboters die Pendelbewegung in frontaler Richtung extrahiert. Es stellte sich heraus, dass die SFA mit zeitlicher Einbettung diese Bewegung mit geringerem Berechnungsaufwand extrahieren kann, und dass sie insbesondere besser auf ungesehene Daten generalisiert. Dies ermöglichte, die gewonnene Komponente selbst zur Filterung der Sensorwerte und zur Generierung des Steuersignals in der sensomotorischen Schleife, welche die Laufbewegung realisiert, zu verwenden. Das so entstandene Laufmuster wurde hinsichtlich seiner Stabilität und Reaktivität mit anderen Laufmustern verglichen, welche auf Basis des gleichen Netzes, aber unter Verwendung anderer Filter entstanden sind. Es wurde gezeigt, dass das SFA-Laufmuster eine mit dem ursprünglichen Netz vergleichbare Stabilität aufwies, dieses hinsichtlich der Reaktivität aber übertraf. Dieser Umstand kann genutzt werden, um schnelle Reaktionen wie solche zur Umfallprävention zu implementieren.

6.1 Offene Fragen

Aus den präsentierten Analysen und Anwendungen ergeben sich zwangsläufig unbeantwortete Fragen. Es bleibt bzgl. der neuronalen Implementation die zentrale Frage, ob ein mit der SFA kompatibles Online-Lernverfahren entwickelt werden kann. Ein solches Lernverfahren sollte die SFA nicht ersetzen, könnte aber dazu dienen, eine offline gelernte SFA sukzessive zu verbessern und auf nichtstationäre Sensordaten anzupassen. Vor allem bei realen Systemen wie Robotern tritt zwangsläufig Verschleiß auf, welcher sich auf die Eigenschaften der Hardware und der Sensoren auswirkt. Ein Online-Verfahren wäre eine Methode, um im Roboter verwendete SFA-Komponenten, z. B. zur Posenerkennung oder Filterung, inkrementell an die sich verändernden Eigenschaften der Hardware anzupassen.

Damit die SFA praktisch für die propriozeptive Posenerkennung und Dimensionsreduktion zur Anwendung kommen kann, müssen Betrachtungen über die Generalisierbarkeit der SFA-Komponenten angestellt werden. Dazu gehören auch Untersuchungen über die optimale Art und Länge der Trainingsdaten: Einerseits besteht die Gefahr, dass kurze Trainingsdaten zu schlecht generalisierenden Komponenten führen, andererseits können lange Trainingsdatensätze dazu führen, dass sensorisch weniger stark erfassbare Posen sowie selten ausgeführtere Posen nicht mehr erkannt werden können.

Auch die Verwendung der SFA in sensomotorischen Schleifen wirft weitere Fragen auf. Für das konkrete Laufmuster wäre es wünschenswert, dass die SFA-Komponente eine geringere Verzögerung aufweist, um das Laufen insgesamt dynamischer zu machen. Eine Idee wäre, die in Abschnitt 2.5.5 vorgestellte *Contextual SFA (cSFA)* verwenden, um das Zielsignal der SFA zu beeinflussen. Ebenso bedarf es weiterer eingehender Untersuchungen, wie die erhöhte Reaktivität des Laufens praktisch für eine Umfallprävention angewandt werden kann.

6.2 Ausblick

Neben der Klärung der offenen Fragen sind viele weitere Untersuchungen mit der SFA denk-

bar. So sollte für zukünftige Untersuchungen die Integration weiterer sensorischer Modalitäten im Vordergrund stehen. Dies ist insbesondere im Hinblick auf die kürzlich am Labor für Neurorobotik fertiggestellte humanoide Roboterplattform *Myon* (Abbildung 6.1) interessant. Diese übertrifft die A-Serie nicht nur an Größe und Gewicht, sondern verfügt über eine stärker redundant und multimodal ausgelegte Sensorik. Außerdem bringt sie neuartige Konzepte in der humanoiden Robotik zur Anwendung: Die verteilte Architektur gestattet es, einzelne Körperteile vollständig autonom zu betreiben und sogar während des Betriebs abzutrennen. Da Myon technisch auf seinem Vorgänger aufbaut und auch über ähnliche Sensoren und Schnittstellen verfügt, können die in dieser Arbeit vorgestellten neuronalen Implementationen leicht auf die neue Plattform übertragen werden.

Um die verschiedenen Sensoren geeignet zu kombinieren, könnte man die verschiedenen Sensormodalitäten zunächst jeweils einzeln in eine oder mehrere SFA-Einheiten geben und die verschiedenen Zwischensignale schließlich in einer finalen SFA-Einheit kombinieren. Ein anderer Ansatz wäre verschiedene Sensorqualitäten in die gleiche Modalität zu überführen und dann erst die SFA auf sie anzuwenden – so ergibt beispielsweise die Ableitung der Motorwinkelsensoren ebenso wie das Integral der Beschleunigungssensoren ein Signal, welches die Geschwindigkeit am jeweiligen Motor bzw. Körperteil repräsentiert.

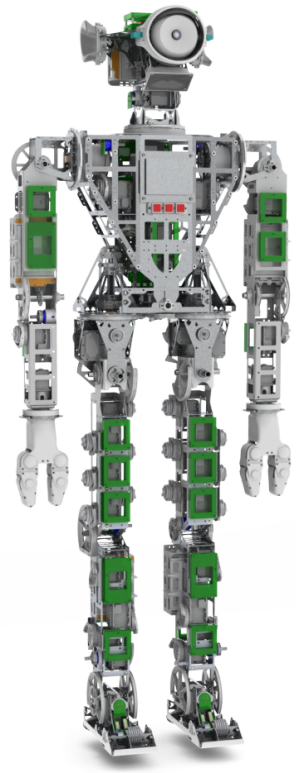


Abbildung 6.1: Myon, Nachfolger der A-Serie und neue humanoide Roboterplattform des Labors für Neurobotik.

Auf den Resultaten dieser Arbeit aufbauend würde die Übertragung der Ergebnisse aus [FSW07] auf die Beschleunigungssensoren eine spannende Untersuchung ergeben. Insbesondere wäre die Integration visueller und motorischer Sensoren denkbar. Genauer gesagt wäre die Idee, analog zu den *grid cells* und *place cells* für den Roboter relevante *posture cells* zu finden. Diese Zellen könnten als Posendetektoren agieren, in dem Sinne, dass jede Zelle sich auf eine bestimmte Pose spezialisiert.

In der Tat gibt es Evidenz für biologische Entsprechungen zu den hypothetisch angenommenen *posture cells*. So konnte beim Rhesusaffen im parietalen Cortex die Existenz so genannter *reach cells* nachgewiesen werden, welche bei der Erwägung und Durchführung von Greifbewegungen aktiv sind [Hyv81]. Insbesondere konnte gezeigt werden, dass sowohl rein visuelle, rein motorische, aber auch bimodale *reach cells* existieren [Blu85]. Auch in anderen Hirnarealen konnten *reach cells* nachgewiesen werden. Eine Hypothese ist, dass das Gehirn mit Hilfe dieser Zellen und durch die Integration visueller und motorischer Sensorinformation die nichttriviale Aufgabenstellung der Hand-Augen-Koordination löst [Wer93]. Die

Ergebnisse zur Emergenz von *place cells* sowie zur Posenerkennung legen die Vermutung nahe, dass die SFA auf einer humanoiden Roboterplattform wie der hier verwendeten solche *reach cells* erlernen kann. Mein Vorschlag, um diese Zellen zu erlernen, wäre den Roboter verschiedene Armbewegungen durchführen zu lassen und die Beschleunigungssensordaten der Bewegungen in eine SFA-Einheit zu geben. Desweiteren beobachtet der Roboter seine Bewegungen durch die im Kopf eingebaute Kamera, dessen Bild in eine zweite SFA-Einheit geleitet wird. Zunächst müssen dann die Ergebnisse der unimodalen SFA-Einheiten ausgewertet werden. Im nächsten, weitaus interessanteren Schritt werden die Komponenten der beiden unimodalen Einheiten in einer bimodalen SFA-Einheit vereinigt. Die Hoffnung ist, dass diese Einheit den Zusammenhang zwischen den Repräsentationen der Handbewegungen in den beiden sensorischen Modalitäten erlernt. Desweiteren sind dann, aufbauend auf den Ergebnissen zur Verwendung von SFA-Komponenten in sensomotorischen Schleifen, weitere Überlegungen zur direkten Verwendung der gelernten Komponenten zur Steuerung oder zum Tracking von Objekten denkbar.

Anhang A: Technische Spezifikation der A-Serie

A-Serie

Die als A-Serie bezeichnete humanoide Roboterplattform des Labors für Neurorobotik wurde auf der Grundlage des Roboterbaukastens Bioloid von der Firma Robotis gebaut. Ein Roboter der Serie ist in Abbildung 1.3 dargestellt, für eine schematische grafische Darstellung siehe Abbildung 1.4.

Höhe: 47 cm

Gewicht: 2,2 kg

AccelBoards: Über den Roboter verteilt befinden sich 8 Platinen mit jeweils einem RISC-Microcontroller und einem 2-achsigen Beschleunigungssensor. Jedes AccelBoard steuert drei Dynamixel-Motoren an und übernimmt einen Teil der Körpersteuerung. Untereinander kommunizieren die AccelBoards über den so genannten *SpinalCord*, dessen Taktrate 100 Hz beträgt.
Mehr Informationen zu der Firmware der AccelBoards finden sich in der Studienarbeit [Thi07].

Aktuatoren: Jedes Gelenk des Roboters wird von einem Dynamixel AX-12 bzw. dessen Nachfolger AX-12+ angetrieben. Diese Motoren enthalten einen Winkelencoder und werden über ein Hochgeschwindigkeits-Bussystem angesteuert.

- Masse: 55 g
- Übersetzungsverhältnis des Getriebes 1/254
- Maximales Haltedrehmoment 165 Ncm
- Maximale Geschwindigkeit 0.196s/60
- Auflösung des Winkelencoders: 0,35°

Sensoren: Die A-Serie besitzt eine Kamera und acht 2-achsige Beschleunigungssensoren vom Typ ADXL213, die über den Körper verteilt sind.

PDA: In den Roboter ist ein PDA integriert, welcher einen Prozessor des Typs PXA272 von Intel enthält. Der PDA ist für die Verarbeitung der Kamerabilder und höhere kognitive Fähigkeiten zuständig.

Anhang B: Implementation

Im Rahmen dieser Arbeit sind verschiedene Bibliotheken zur Analyse der SFA sowie zu ihrer Anwendung auf dem A-Serie-Roboter implementiert worden. Alle Module wurden in Python programmiert und basieren zum Teil auf dem *Modular Toolkit for Data Processing (MDP)* [ZWWB09].

Im folgenden werden die wichtigsten Module kurz vorgestellt. Der kommentierte Quellcode kann unter `http://informatik.hu-berlin.de/~hoefer/sfa/` heruntergeladen werden.

Spezielle MDP-Nodes: `/mdp_custom_nodes`

Für die Arbeit wurden teils weitere Lernverfahren, teils andere SFA-Expansionen evaluiert.

Expansion

`/mdp_custom_nodes/custom_expansion_nodes.py`:

`CustomPolynomialExpansionNode(mdp.Node)`

– Expansionsmodul ähnlich `mdp.nodes.PolynomialExpansionNode`, allerdings mit weiteren Einstellungmöglichkeiten. Außerdem generiert die Expansion eine `labels`-Liste, welche die nachträgliche Analyse und Benennung der eingehenden Sensorwerte erlaubt. (Allerdings ist diese Node daher von der Berechnung her um einige Faktoren langsamer als die Original-Node.)

`VariableExpansionNode(mdp.Node)`

– Die `VariableExpansionNode` erlaubt beliebige Expansion durchzuführen, in dem eine Gewichtsfunktion g, eine Kombinationsfunktion p und eine Transferfunktion a angegeben werden. Es muss zusätzlich der Grad der Expansion angegeben werden, d. h. wie viele Eingänge jeweils miteinander kombiniert werden. (Im polynomiellen Kontext bedeutet das, welchen Grad die Polynome haben.)

Für eine polynomielle Expansion wären die Funktionen folgendermaßen zu definieren: $g(x) := x$, $p(x, y) := xy$, $a(x) := x$.

Es lassen sich aber beispielsweise auch Energie-Detektor-Neuronen simulieren: $g(x) := x^2$, $p(x, y) := x + y$, $a(x) := \sqrt{x}$.

`TanhExpansionNode(VariableExpansionNode)`

– Implementiert die `VariableExpansionNode` mit $g(x) := \alpha x$, $p(x, y) := x + y - b$, $a(x) := \tanh x/\alpha$, für einen Skalierungsfaktor α und einen Bias b (siehe Abschnitt 3.2.2).

`FastTanhExpansionNode(mdp.nodes.QuadraticExpansionNode)`
– Effizientere Implementation der tanh-Expansion (notwendig für Verwendung in sensomotorischer Schleife).

Filter

Folgende Filterstrukturen finden in Kapitel 5 Verwendung.
`/mdp_custom_nodes/filter_nodes.py`:

`LMSAdaptiveVolterraFilter(mdp.Node)`
– Modul zum Trainieren eines adaptiven quadratischen Volterra-Filters mit einer Least-Mean-Squares-Lernregel.

`SimpleIIR(mdp.Node)`
– Einfaches IIR-Filter aus zwei phasenverschobenen Leaky-Integrator-Neuronen.

`SimpleFIR(mdp.Node)`
– Einfaches eindimensionales FIR-Filter.

Zeitliche Einbettung

`/mdp_custom_nodes/time_nodes.py`:

`TimeDelayNode(mdp.Node)`
– Funktioniert fast identisch wie `mdp.nodes.TimeFramesNode`, nur dass nicht in die Zukunft, sondern in die Vergangenheit eingebettet wird. Dies ist bei der Online-Verwendung in einer sensomotorischen Schleife notwendig.

Generierung von SFA-Komponenten für den *BrainDesigner*

Der *BrainDesigner* ist ein Werkzeug, welches am Labor für Neurorobotik entwickelt wurde, um auf einfache Weise neuronale Netze für sensomotorische Schleifen zu erstellen und diese direkt auf dem Roboter auszuführen.

Die eigentlichen Netze werden in einem einfachen XML-Format gespeichert, was gestattet, diese auch auf andere Weise als mit dem BrainDesigner zu generieren. Daher wurde im Rahmen dieser Arbeit ein Konverter programmiert, um SFA-Module (quadratische, mit tanh-Expansion, mit zeitlicher Einbettung) in ein neuronales Netz zu konvertieren, um daraus eine BrainDesigner-Datei (*BDN*) zu generieren.

Neuronale Netze: `/neuralnet`

Das `neuralnet`-Paket basiert auf dem `mdp.Graph`-Bibliothek und gestattet, einfache Feed-Forward-Netze zu erstellen.

`/neuralnet/neuralnet.py`:

`NeuralNet(graph.Graph)`
– Hauptklasse für neuronale Netze.

`NeuralNetNode(graph.GraphNode)`
– Superklasse für Knoten neuronaler Netze.
`Input(NeuralNetNode)`
`Output(NeuralNetNode)`
`TanhNeuron(NeuralNetNode)`
`Component(NeuralNetNode)`
– Wrapperklasse für ein `NeuralNet`-Objekt, um verschachtelte Netze zu ermöglichen.
`ComponentInputNode(NeuralNetNode)`
`StandardSynapse(graph.GraphEdge)`
`WrongInputDimensionException(Exception)`

`/neuralnet/neuralnet_from_sfa.py`:
Funktionen zur Erstellung von SFA-Netz-Instanzen.
`generate_tanh_expansion_net(tanh_expnode, bdn_guid=‘‘SFATanhExpansion‘‘)`
– Generiere ein tanh-Expansionsnetz
`generate_time_delay_net(input_dim, time_frames=1, gap=1, scaling_factor=0.125, bdn_guid=‘‘TimeFrames‘‘)`
– Generiere ein Netz zur zeitlichen Einbettung. Basiert auf `TimeDelayNode`.
`generate_simple_sfa_net(sfanode, bdn_guid=‘‘SFAExecute‘‘)`
– Generiere ein lineares SFA-Netz.
`generate_te_sfa_tanh_net(nodes, input_dim, bdn_guid=‘‘TE_SFA_Tanh‘‘)`
– Generiere ein SFA-Netz mit zeitlicher Einbettung.
`connect_sfa2_components(sfanode, expnode, sfanet, expnet, net_meta, layer_input, input_type)`
– Verbinde eine `CustomExpansionNode` und eine `SFANode` in einem neuronalen Netz.

BDN-Konvertierung: `/bdn`

Der BDN-Konverter basiert auf der in MDP verwendeten Template-Engine *Templet*.

`/bdn/bdn_converter.py`:
`BDNLibraryConverter(object)`
– Erstellt eine BDN-Library aus mehreren neuronalen Netzen.
`BDNModuleConverter(object)`
– Konvertiert ein `NeuralNet`-Objekt in die BDN-Darstellung.
`BDNMainTemplate(templet.Template)`
`BDNModuleTemplate(templet.Template)`
`BDNNodeTemplate(templet.Template)`
`BDNEdgeTemplate(templet.Template)`
`generate_guid(seed)`

Beispiele

`/sfa_bdn_te_sfa.py`:
Beispiel, wie eine SFA mit tanh-Expansion und zeitlicher Einbettung generiert und konvertiert wird.
`/sfa_bdn_acc.py`:
Beispiel, wie eine quadratische SFA generiert und konvertiert wird.

Ansteuerung des A-Serie-Roboters aus Python

Alternativ zur Verwendung des BrainDesigners kann die A-Serie über den so genannten *TransparentMode* auch direkt über eigene Implementationen angesteuert werden. Insbesondere bei großen SFA-Netzen bietet sich diese Variante an, da zum Zeitpunkt der Arbeit nur auf diese Weise die Implementation der SFA in sensomotorischen Schleifen in Echtzeit möglich war.

Schnittstelle zum Roboter: `/robot_connect`

Für die Ansteuerung des Roboters ist ein rudimentäres Framework entstanden, welches die Ausführung beliebiger Motoraktionen sowie das Logging der Sensorwerte gestattet. Im Prinzip ist dieses Framework auch mit anderen Roboterplattformen nutzbar, insofern die entsprechenden Schnittstellen implementiert werden.

Anmerkung: Zur Ansteuerung eines Roboters ist zusätzlich die jeweilige `TransparentMode.dll`-Bibliothek notwendig.

`/robot_connect/applications.py`:
Basisklassen für die Ansteuerung zum Roboter.

`Application(object)`
– Schnittstelle für alle Applikationen.
`RobotApplication(Application)`
– Basisklasse für Roboterapplikationen.
`ASeriesApplication(RobotApplication)`
– Basisklasse für Applikation der A-Serie. Alle A-Serie-Applikation erben von dieser Klasse.
`ASeriesDummy(ASeriesApplication)`
– Dummyklasse für Applikation der A-Serie, welche das Offline-Testen von Applikationen erlaubt.

`/robot_connect/structures.py`:

`TimeDelayTracer(object)`
– Online-Pendant zur `TimeDelayNode`, welche benötigt wird, um eine zeitliche Einbettung während der Ausführung in der sensomotorischen Schleife zu realisieren.

`/robot_connect/indices.py`:
Enthält Klassen zur Auswahl der Motor- und Sensorwerte (nicht angegeben).

Beispiel

`/aseries_neural_walk.py`:
A-Serie-Applikation, welche das neuronale Laufen mit den verschiedenen Filtern aus Abschnitt 5.3 implementiert.

Werkzeuge: `/tools`

SFA-Analyse

`/tools/analyse_sfa.py`:

`generate_coefficient_map(expnode, sfanode, component=0, labels=None)`
– Analyse der quadratischen SFA. Generiert eine Tabelle, welche jedem Tupel von Eingabewerten das zugehörige SFA-Gewicht für die Komponente `component` zuordnet. Ebenso wird eine Tabelle mit den zugehörigen Mittelwerten zurückgegeben.

`generate_equations(expnode, sfanode, labels=None, precision=8, average=True)`
– Gibt die Gleichungen für die SFA-Komponenten in lesbarer Form zurück.

Quadriken

`/tools/sfa_quadrics.py`:

`generate_quadric_gradient(expnode, sfanode, free_inputs, component=0, labels=None)`
– Gibt eine Lambda-Funktion zurück, welche dem Gradienten für eine quadratische SFA-Komponente mit festgesetztem Ausgabewert μ (siehe 2.4.4) entspricht.

`random_gradient_descent(expnode, sfanode, const, free_inputs, data, start=None, iterations=100, random_scale = None, alpha = 0.01, eps = 10e-5, stop_limit=1000, component = 0, domain_limits=None, verbose = False)`
– Führe einen Gradientenabstieg zur Exploration einer SFA-Quadrik wie in Abschnitt 2.4.4 beschrieben durch.

Beispiel

`/dimred_quadrics.py`:
Skript, welches eine Quadrik der quadratischen SFA, angewandt auf die Sensordatensequenz aus Kapitel 4, mittels Gradientenabstieg (offline) exploriert.

Literaturverzeichnis

[ABR64] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964. 5, 15, 33

[BBW06] T. Blaschke, P. Berkes, and L. Wiskott. What is the relationship between Slow Feature Analysis and Independent Component Analysis? *Neural Computation*, 18 (10):2495–2508, 2006. 26, 28

[Bec93] S. Becker. Learning to Categorize Objects Using Temporal Coherence. In *Advances in Neural Information Processing Systems 5*, pages 361–368. Morgan Kaufmann, 1993. 3, 42

[Ber06] P. Berkes. *Temporal slowness as an unsupervised learning principle.* Doktorarbeit, Humboldt-Universität zu Berlin, 2006. 2, 13, 15, 16

[Bla05] T. Blaschke. *Independent Component Analysis and Slow Feature Analysis: Relations and Combination.* Doktorarbeit, Institut für Physik, Humboldt Universität zu Berlin, D-10099 Berlin, 2005. `http://edoc.hu-berlin.de/docviews/abstract.php?lang=ger&id=25458`. 27, 28

[Blu85] Baruch Blum. Manipulation reach and visual reach neurons in the inferior parietal lobule of the rhesus monkey. *Behavioural Brain Research*, 18(2):167 – 173, 1985. ISSN 0166-4328. URL `http://www.sciencedirect.com/science/article/B6SYP-484NDB7-BM/2/f906d79661c7e877dee46a144570d88b`. 115

[BW02] P. Berkes and L. Wiskott. Applying Slow Feature Analysis to Image Sequences Yields a Rich Repertoire of Complex Cell Properties. In José R. Dorronsoro, editor, *Proc. Intl. Conf. on Artificial Neural Networks - ICANN'02*, Lecture Notes in Computer Science, pages 81–86. Springer, 2002. 34, 35

[BW06] P. Berkes and L. Wiskott. On the analysis and interpretation of inhomogeneous quadratic forms as receptive fields. *Neural Computation*, 18(8):1868–1895, 2006. 21

[BW07] P. Berkes and L. Wiskott. Analysis and interpretation of quadratic models of receptive fields. *Nature Protocols*, 2(2):400–407, 2007. 21

[Cas97] M. C. Casdagli. Recurrence plots revisited. *Physica D: Nonlinear Phenomena*, 108 (1–2):12–44, 1997. 37

[CS08] F. Creutzig and H. Sprekeler. Predictive coding and the slowness principle: An information-theoretic approach. *Neural Comput.*, 20(4):1026–1041, 2008. ISSN 0899-7667. 28

[CV95] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 30(3): 273–297, 1995. 5

[Dei10] R. Deimel. Contextual Slow Feature Extraction Framework (in Vorbereitung). 2010. 30

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience Publication, 2000. 25

[Fio02] S. Fiori. A Minor Subspace Algorithm Based on Neural Stiefel Dynamics. 2002. 42

[Fod02] I. Fodor. A Survey of Dimension Reduction Techniques. Technical report, 2002. 73

[Föl91] P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3:194–200, 1991. 3

[FSW07] M. Franzius, H. Sprekeler, and L. Wiskott. Slowness and Sparseness Lead to Place, Head-Direction, and Spatial-View Cells. *PLoS Computational Biology*, 3(8):e166, August 2007. (doi:10.1371/journal.pcbi.0030166). 20, 33, 35, 36, 115

[GK02] W. Gerstner and W. M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002. 43

[Hil07] M. Hild. *Neurodynamische Module zur Bewegungssteuerung autonomer mobiler Roboter*. Doktorarbeit, Humboldt-Universität zu Berlin, 2007. 43, 45

[HKHM11] M. Hild, M. Kubisch, S. Höfer, and H. Mellmann. Using Quadric-Representing Neurons (QRENs) for Real-Time Learning of an Implicit Body Model During Autonomous Self-Exploration (submitted). In *2011 IEEE International Conference on Robotics and Automation (ICRA 2011)*. Shanghai, China, May 9-13 2011. 23

[HKO01] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, 2001. 25

[Hyv81] J. Hyvärinen. Functional mechanisms of the parietal cortex. In E. Grastyn and P. Molnarg, editors, *Sensory Functions*, volume 16, pages 35–49. Pergamon Press and Akademian, Kiado, 1981. 115

[Höf09] S. Höfer. Analyse des Laufverhaltens von humanoiden Robotern mit der Slow Feature Analysis. Studienarbeit, Humboldt-Universität zu Berlin, 2009. 62, 89, 90, 92, 95, 98

[Jae01] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. 2001. URL `http://publica.fraunhofer.de/documents/B-73135.html`. 5

[KED+01] C. Kayser, W. Einhäuser, O. Dümmer, P. König, and K. Körding. Extracting Slow Subspaces from Natural Videos Leads to Complex Cells. In *Artificial Neural Networks – ICANN 2001, Volume 2130 of Lecture Notes in Computer Sciece*, pages 1075–1080. Springer, 2001. 3, 42

[KK09] W. Konen and P. Koch. How slow is slow? SFA detects signals that are slower than the driving force. arXiv.org e-Print archive, `http://arxiv.org/abs/0911.4397v1`, November 2009. 38, 100

[Kon09] W. Konen. On the numeric stability of the SFA implementation sfa-tk. arXiv.org e-Print archive, `http://arxiv.org/abs/0912.1064`, December 2009. 20

[LdVC95] S. Li, O. de Vel, and D. Coomans. Comparative performance analysis of non-linear dimensionality reduction methods. Technical report, James Cook University, North, 1995. 75

[LLC92] S. M. Lau, S. H. Leung, and B. L. Chan. A reduced rank second-order adaptive Volterra filter. In *ISSPA 92, Signal Processing and its Applications*, pages 561–563. Gold Coast, Australia, August, 16-21t 1992. 32

[LWW10] Robert Legenstein, Niko Wilbert, and Laurenz Wiskott. Reinforcement learning on slow features of High-Dimensional input streams. *PLoS Comput Biol*, 6(8):e1000894, 2010. URL `http://dx.doi.org/10.1371/journal.pcbi.1000894`. 38

[Mat91] J. J. Mathews. Adaptive polynomial filters. *IEEE Signal Processing Magazine*, 8 (3):10–26, 1991. 32

[Mel94] B. W. Mel. Information processing in dendritic trees. *Neural Computation*, 6(6): 1031–1085, 1994. 5

[Mit91] G. Mitchison. Removing time variation with the anti-Hebbian differential synapse. *Neural Computation*, 3(3):312–320, 1991. 3, 42

[MP43] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, December 1943. URL `http://dx.doi.org/10.1007/BF02478259`. 5, 42

[MS94] L. Molgedey and H. G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Phys. Rev. Lett.*, 72(23):3634–3637, June 1994. 26, 28

[Nic06] H. Nickisch. Extraction of visual features from natural video data using Slow Feature Analysis. Diplomarbeit, Technische Universität Berlin, 2006. 7, 33, 34, 46

[OD71] J. O´Keefe and J. Dostrovsky. The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely moving rat. *Brain Research*, 34:171–175, 1971. 35

[Oja82] E. Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, November 1982. ISSN 0303-6812. URL `http://dx.doi.org/10.1007/BF00275687`. 41

[PB06] R. Pfeifer and J. C. Bongard. *How the Body Shapes the Way We Think: A New View of Intelligence (Bradford Books)*. The MIT Press, November 01, 2006. ISBN 0262162393. 1, 41

[Rou87] P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987. ISSN 0377-0427. URL `http://portal.acm.org/citation.cfm?id=38772`. 74

[San89] T. D. Sanger. Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network. *Neural Networks*, 2:459–473, 1989. 42

[SB98] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning I: Introduction, 1998. 38

[Sel05] J. M. Selig. *Geometric Fundamentals of Robotics*. Springer, New York, 2005. 23

[SHH09] M. Spranger, S. Höfer, and M. Hild. Biologically Inspired Posture Recognition and Posture Change Detection for Humanoid Robots. In *Proc. IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 562–567. Guilin, China, December 18-22 2009. 61, 62

[SL09] M. Spranger and M. Loetzsch. The semantics of SIT, STAND, and LIE embodied in robots. In Niels A. Taatgen and Hedderik van Rijn, editors, *Proceedings of the 31th Annual Conference of the Cognitive Science Society (Cogsci09)*, pages 2546–2552. Cognitive Science Society, Austin, TX, 2009. 61

[SSK05] A. K. Seth, O. Sporns, and J. L. Krichmar. Neurorobotic models in neuroscience and neuroinformatics. *Neuroinformatics*, 3(3):167–170, 2005. ISSN 1539-2791. 1

[SSM98] B. Schölkopf, A. Smola, and K.-M. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput.*, 10(5):1299–1319, 1998. ISSN 0899-7667. 33

[Ste01] L. Steels. Language Games for Autonomous Robots. *IEEE Intelligent Systems*, 16 (5):16–22, 2001. ISSN 1541-1672. 61

[Ste10] A. Stephan. Sensorische und sensomotorische Karten – ein Vergleich unüberwachter Algorithmen zur Dimensionsreduktion und Vektorquantisierung. Diplomarbeit, Humboldt-Universität zu Berlin, 2010. 60, 73, 74, 75, 84, 86

[Tak81] F. Takens. Detecting strange attractors in turbulence. *D. A. Rand and L.-S. Young. Dynamical Systems and Turbulence, Lecture Notes in Mathematics*, 898:366–381, 1981. 5, 31

[Thi07] C. Thiele. Integrierte Entwicklungsumgebung zur Bewegungssteuerung humanoider Roboter. Studienarbeit, Humboldt-Universität zu Berlin, 2007. 117

[TMR90] J. S. Taube, R. U. Muller, and J. B. Ranck. Head direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis. *The Journal of Neuroscience*, 2:420–435, 1990. 35

[TPB99] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. In *Proc. of 37th Allerton Conference on Communication and Computation. Monticello, IL*, 1999. 28

[TS07] R. Turner and M. Sahani. A Maximum-Likelihood Interpretation for Slow Feature Analysis. *Neural Computation*, 19(4):1022–1038, 2007. ISSN 0899-7667. 30

[Wel67] P. D. Welch. The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms. *IEEE Transactions on Audio Electroacoustics*, AU(15):70–73, 6 1967. 106

[Wer93] W. Werner. Neurons in the Primate Superior Colliculus are Active Before and During Arm Movements to Visual Targets. *European Journal of Neuroscience*, 5(4): 335–340, 4 1993. 115

[Wer08] B. Werner. Sensomotorische Erzeugung eines Gangmusters für humanoide Roboter. Studienarbeit, Humboldt-Universität zu Berlin, 2008. 44, 89, 90, 106

[Wie58] N. Wiener. Nonlinear Problems in Random Theory. 1958. 5

[Wis98] L. Wiskott. Learning Invariance Manifolds. In *Proc. of the 5th Joint Symp. on Neural Computation, May 16, San Diego, CA*, volume 8, pages 196–203. Univ. of California, San Diego, CA, 1998. 3, 13, 22

[Wis03a] L. Wiskott. Slow Feature Analysis: A Theoretical Analysis of Optimal Free Responses. *Neural Computation*, 15(9):2147–2177, September 2003. 20

[Wis03b] L. Wiskott. Estimating Driving Forces of Nonstationary Time Series with Slow Feature Analysis. arXiv.org e-Print archive, `http://arxiv.org/abs/cond-mat/0312317/`, December 2003. 36, 37

[WKV06] R. Wyss, P. König, and P. F. M. J. Verschure. A Model of the Ventral Visual System Based on Temporal Stability and Local Memory. *PLoS Biol*, 4(5), April 18, 2006. URL `http://dx.doi.org/10.1371/journal.pbio.0040120`. 3

[WS02] L. Wiskott and T. Sejnowski. Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, 14(4):715–770, 2002. 3, 13, 19, 20, 34, 42

[Zak05] A. Zaknich. *Principles of adaptive filters and self-learning systems*. Springer London, 2005. 52, 55, 109

[ZM98] A. Ziehe and K.-R. Müller. TDSEP - an efficient algorithm for blind separation using time structure, 1998. 26

[ZWWB09] T. Zito, N. Wilbert, L. Wiskott, and P. Berkes. Modular toolkit for Data Processing (MDP): a Python data processing frame work, 2009. 19, 84, 119

Danksagung

Zum Schluss möchte ich mich dem schwierigsten Teil dieser Arbeit widmen, nämlich die Personen aufführen, ohne die diese Arbeit nicht möglich gewesen wäre und denen ich zu tiefstem Dank verpflichtet bin. Schwierig deswegen, weil es mir unmöglich erscheint, in ein paar kurzen Absätzen niemanden auszulassen und jeden gebührend zu würdigen.

Zunächst geht mein Dank an meine Gutachter Herrn Prof. Burkhard und Herrn Prof. Wiskott sowie ganz besonders an meinen Betreuer Herrn Dr. Hild: Dafür, dass man zu jeder Zeit an seine Tür klopfen kann, um schlaue und auch mal nicht so schlaue Fragen loszuwerden, und dass er auf unnachahmliche Weise um die wissenschaftliche Fortbildung seiner Schützlinge bemüht ist. Ebenso möchte ich dem gesamten NRL-Team für eine wundervolle Arbeitsatmosphäre, viele anregende Gespräche und Nachsichtigkeiten bezüglich meiner manchmal bescheidenen handwerklichen Fähigkeiten danken. André Stephan möchte ich für die Bereitstellung der von ihm aufgenommenen A-Serie-Daten sowie erste Tipps für den Umgang mit diesen danken. Großen Dank verpflichtet bin ich auch Michael Spranger für wertvolle Tipps und sein großes Engagement bezüglich meines wissenschaftlichen Weiterkommens.

Nicht weniger Dank gilt Timur Doumler und Sven Dähne für anregenden interdisziplinären Austausch und sehr hilfreiches und penibles Korrekturlesen; Nicole Wendt, dass sie mich unterstützt und motiviert; und allen meinen wertvollen Freunden, die mich seit vielen Jahre begleiten.

Mein allergrößter Dank schließlich gilt meinen Eltern, die immer an mich geglaubt und alles menschenmögliche getan haben, um mich zu unterstützen. Danke!