Analyse des Laufverhaltens von humanoiden Robotern mit der Slow Feature Analysis

Sebastian Höfer

10. August 2009

Studienarbeit

Betreuer: Dr. Manfred Hild

Institut für Informatik

Lehrstuhl für Künstliche Intelligenz

Prof. Dr. Hans-Dieter Burkhard

Zusammenfassung

Die Slow Feature Analysis (SFA) ist ein Lernverfahren, welches sich langsam verändernde Komponenten aus einem mehrdimensionalen Eingabesignal extrahiert. Sie findet eine Eingabe-Ausgabe-Funktion, die es erlaubt, aus dem Eingabesignal mehrere ihrer Langsamkeit nach geordnete, unkorrelierte Komponenten zu berechnen. Diese Eingabe-Ausgabe-Funktion kann offline berechnet werden und liefert die langsamsten Komponenten, welche die optimale Lösung innerhalb einer eingeschränkten Familie von Funktionen darstellen. In der Regel kann bei geeigneter Wahl von Trainingsdaten diese Eingabe-Ausgabe-Funktion auch auf unbekannten Testdaten annähernd optimale Ergebnisse liefern und das Verfahren somit online nutzbar gemacht werden.

Die SFA wird auf Beschleunigungssensordaten einer humanoiden Roboterserie angewandt, um Informationen über den aktuellen Zustand des Roboters zu gewinnen. Dabei geht es vor allem darum, für ein bestimmtes Laufmuster mit der SFA eine Komponente zu extrahieren, welche dem Roboter als Indikator dienen kann, sobald er droht das Gleichgewicht zu verlieren und umzufallen. Zuerst werden verschiedene Anwendungsmöglichkeiten der SFA evaluiert und auf den gleichen Daten gelernt und getestet. Die gefundenen Komponenten werden vorgestellt und interpretiert, darunter einige Komponenten, die sich als Kandidaten für eine Umfalldetektion eignen. Dann werden die von der SFA gelernten Eingabe-Ausgabe-Funktionen auch auf unbekannte Testdaten des gleichen sowie anderer Modelle derselben Roboterfamilie angewandt. Die Analyse zeigt, dass die gelernten Parameter in Abhängigkeit von den Trainingsdaten teilweise robust genug sind, um zu generalisieren und auf anderen Robotern gleicher Bauart als Umfalldetektion zu dienen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Eines der großen Probleme in der Mustererkennung ist es, invariante Elemente in einem gegebenen Eingabesignal zu erkennen. Bei visuellem Input durch eine Videokamera beispielsweise, welcher in mehreren einzelnen Photozellen bzw. Pixeln vorliegt, ist man in der Regel daran interessiert, Objekte über den Verlauf der Zeit zu identifizieren und ihre Lage und Bewegung während eines gewissen Zeitraums zu berechnen. Die von der Gruppe um Laurenz Wiskott entwickelte Slow Feature Analysis (SFA) ist ein unüberwachter Lernalgorithmus, welcher invariante Komponenten aus einem bestehenden Eingangssignal extrahiert. Die Autoren konnten damit sehr gute Ergebnisse im Bereich der Verarbeitung von visuellen Daten erzielen. So konnten sie nachweisen, dass die SFA zuvor unbekannte, sich über ein visuelles Sichtfeld bewegende Objekte als langsamste Komponenten identifizieren und unter anderem deren Lage im Raum bestimmen kann [WS02].

Nun liegt der Gedanke nahe, die SFA auch auf die Robotik und die Analyse anderer Daten zu übertragen. Bei der Entwicklung von humanoiden Robotern ist man sehr daran interessiert, propriozeptive Funktionen zu realisieren, indem der Roboter die ihm zur Verfügung stehenden sensorischen Daten auf sinnvolle Weise auswertet und zur Steuerung seines Verhaltens nutzt. So könnte durch die SFA möglicherweise berechnet werden, in welcher Haltung oder Position sich der Roboter gerade befindet, welche Bewegung er gerade vollführt, o. ä.

Die Erkennung von Haltungen und Gesten von humanoiden Robotern mit der SFA wurde bereits von Michael Spranger untersucht [MSM09]. Dabei wurden sämtliche dem Roboter zur Verfügung stehenden Daten genutzt, dazu gehörten Beschleunigungssensoren, Winkel der Gelenke und Motoren sowie aufbereitete visuelle Daten. Die visuellen Daten allerdings stammten nicht von der Kamera des Roboters, sondern wurden extern aufgenommen und zeigten den Roboter selbst. Die Versuche ergaben, dass verschiedene Haltungen des Roboters durch die SFA klar unterschieden und geclustert werden konnten.

Der Fokus dieser Arbeit liegt auf der Analyse von Beschleunigungssensoren, die an den betrachteten humanoiden Robotern angebracht sind. Die Fragestellung ist, welche Informationen sich nur aus der Analyse der Beschleunigungssensordaten mit der SFA ergeben. Im Detail wird ein konkretes Laufmuster betrachtet, aus welchem mit Hilfe der SFA charakteristische Komponenten des Laufens extrahiert werden sollen. Ein besonderes

Augenmerk liegt dabei darauf, eine Komponente zu finden, die dem Roboter signalisiert, wann er droht das Gleichgewicht zu verlieren und umzufallen.

Im ersten Teil dieser Arbeit werde ich eine genaue mathematische Definition des Lernproblems angeben und dann den der SFA zugrunde liegenden Algorithmus von Wiskott und Sejnowski erläutern. Der zweite Teil erklärt die am Roboter durchgeführten Experimente und stellt vor, welche Ergebnisse die Analyse der Beschleunigungssensordaten mit der SFA hervorgebracht hat. Im letzten Kapitel wird schließlich eine Zusammenfassung der Ergebnisse, offener Fragen sowie ein Ausblick auf die praktische Anwendbarkeit der SFA bei der Analyse von Beschleunigungssensordaten gegeben.

Kapitel 2

Slow Feature Analysis (SFA)

2.1 Motivation

Um zu veranschaulichen, welche Informationen durch die SFA gefunden werden sollen, sei ein Beispiel aus [WS02] angeführt. Abbildung 2.1 zeit drei Objekte, genauer drei konturierte Buchstaben, welche sich nacheinander langsam über ein visuelles Feld bewegen. Dieses visuelle Feld besteht aus vielen einzelnen Photorezeptoren, die auf dunkle und helle Grautöne mit einem positiven bzw. negativen Ausschlag reagieren. Diese Rezeptoren lassen sich als ein mehrdimensionales Eingangssignal $\mathbf{x}(t)$ in Abhängigkeit von der Zeit auffassen. Beispielhaft sind drei einzelne Photorezeptoren $x_1(t), x_2(t)$ und $x_3(t)$ eingezeichnet. Die Konturierung der Buchstaben bewirkt, dass die einzelnen Rezeptoren nicht mit einer rein positiven oder rein negativen Ausgabe reagieren, sobald die Buchstaben sich an ihnen vorbeibewegen, sondern in kurzer Abfolge sowohl negative als auch positive Signale erzeugen. Die Ausgabe der eingezeichneten Rezeptoren ist in Abbildung 2.2 links aufgeführt: Wie man sehen kann, produziert jedes Objekt aufgrund seiner Konturierung mehrere aufeinanderfolgende negative und positive Ausschläge, welche vor allem die Klassifizierung der Objektidentität erschweren.

Abbildung 2.2 rechts stellt dar, welche Daten man gerne durch einen geeigneten Algorithmus errechnen würde: Zum einen die eindeutige Feststellung der Objektidentität, welche als *Was*-Information bezeichnet wird, zum anderen Informationen zur räumlichen Lage der Objekte zu bestimmten Zeitpunkten, welche die so genannte *Wo*-Information ist. Diese erwünschten Informationen sind allerdings nur implizit in den vorhandenen Sensordaten enthalten, und somit ist eine Eingabe-Ausgabe-Funktion gesucht, welches die gesuchten *Was*- und *Wo*-Informationen aus den Sensordaten extrahiert. Offensichtlich sind die *Was*- und *Wo*-Information allerdings eng mit einander verknüpft und nicht unvergleichbar, da diese beide über einen ähnlichen Zeitraum hinweg variieren.

Die Grundannahme der SFA besteht nun darin, dass die *langsamsten* Komponenten des Signals (über die Zeit gesehen) den höchsten Abstraktionsgrad haben. Im vorangegangenen Beispiel erscheint diese Annahme einleuchtend, denn während durch die Konturierung die Ausgabe der einzelnen Photorezeptoren flackert, verändert sich z. B. die Objektidentität nicht. Eine simple Tiefpassfilterung genügt aber offensichtlich nicht, da dadurch nicht die gesuchten implizit in der Summe der sensorischen Signale enthaltenen

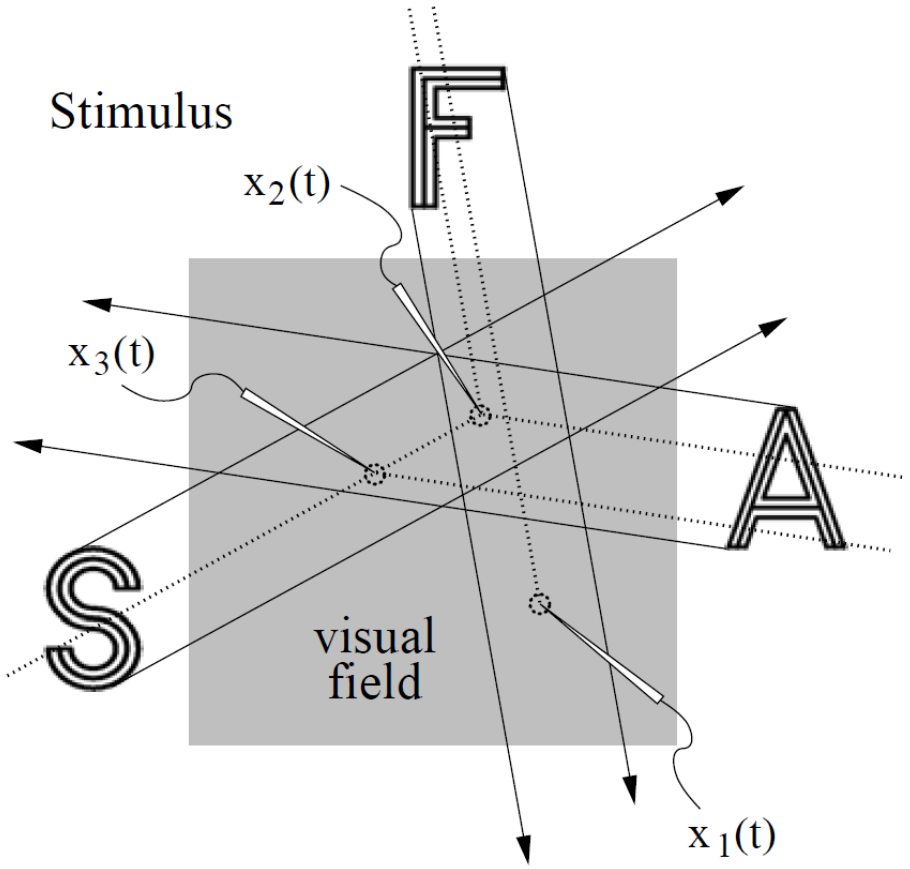

Abbildung 2.1: Drei verschiedene Objekte die sich nacheinander über ein visuelles Feld bewegen.

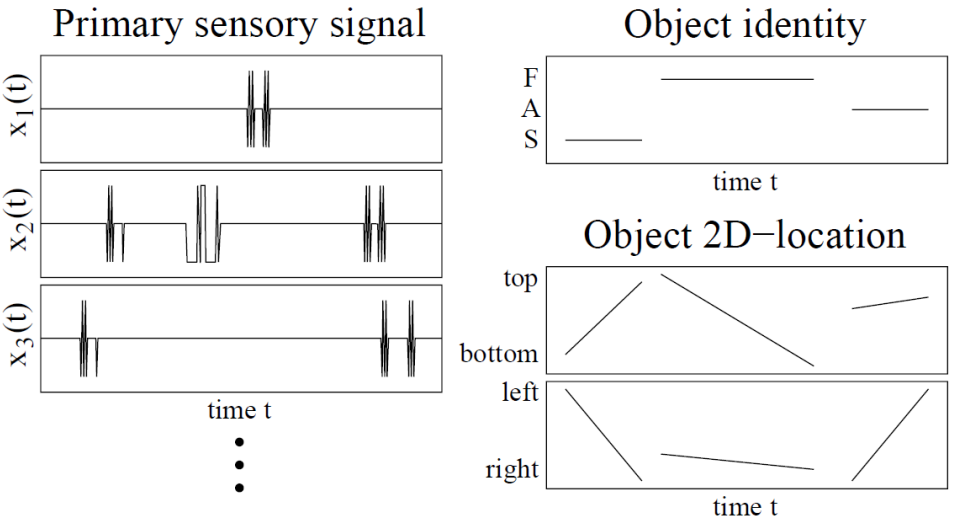

Abbildung 2.2: Sensordaten und gesuchte zu berechnende Informationen.

langsamen Komponenten extrahiert werden. Die gesuchte Funktion darf also nicht nur lokal operieren, sondern muss globale Informationen integrieren.

Der von Wiskott und Sejnowski vorgeschlagene Algorithmus kann prinzipiell als Offline-Verfahren gesehen werden, da die genaue Berechnung der langsamsten Komponenten aufwendig ist. Die Autoren schlagen daher vor, dass die Parameter, die der Algorithmus offline auf den Trainingsdaten lernt, auch für ungesehene reale Instanzen zu benutzen, um das Verfahren auch online nutzbar zu machen. Wie wir sehen werden, beschränkt sich Anwendung der Parameter auf ungesehene Instanzen auf wenige Matrixadditionen und Multiplikationen. Allerdings sind die online berechneten Komponenten in der Regel verrauschter und ungenauer, in der Praxis jedoch meist ausreichend.

2.2 Das Lernproblem

Im folgenden soll eine exakte mathematische Definition des im letzten Kapitel vorgestellten Lernproblems angegeben werden. 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 $\mathbf{y}(t)$ ergibt.

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:

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

unter folgenden zusätzlichen Nebenbedingungen

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

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

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

Die spitzen Klammern stehen dabei für Mittelung über die Zeit:

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

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: Die Ableitung, also der Grad der Veränderung der Funktionswerte, quadratisch über die Zeit gemittelt soll minimal sein[1]. Somit ist klar, dass gerade

[1] Das Quadrat dient dabei wie üblich als Ersatz für den Betrag.

die langsamste Komponente aus dem Eingangssignal extrahierte Komponente Gleichung (2.1) erfüllt. Es ist zu beachten, dass in der Regel nicht nur eine, sondern J langsame Komponenten berechnet werden.

Die Nebenbedingungen (2.2) sowie (2.3) helfen die triviale Lösung $y_j(t) = const$ zu vermeiden, da eine konstante Funktion eine Varianz von 0 hat. Gleichung (2.2) wurde lediglich eingeführt, um eindeutigere und vergleichbarere Ergebnisse zu erhalten[2]. Gleichung (2.4) schließlich soll verhindern, dass die einzelnen langsamen Komponenten einander einfach reproduzieren, stattdessen sollen sie tatsächlich verschiedene Informationen enthalten. Zudem definiert sie eine Ordnung auf den einzelnen Komponenten, so dass y_1 das langsamste Signal ist, y_2 das zweitlangsamste, etc.

Das Lernproblem ist ein Optimierungsproblem aus der Variationsrechnung, dessen analytische Lösung aufwendig ist. In [Wis03] werden verschiedene analytische Lösungmethoden des Problems diskutiert, in dieser Arbeit beschränke ich mich auf die in [WS02] vorgestellte numerische Lösung mittels Slow Feature Analysis. Diese Idee dieses Algorithmus', welcher im nächsten Kapitel vorgestellt wird, ist die Eingabe-Ausgabe-Funktionen g_j derart einzuschränken, dass sie sich als Linearkombination einer endlichen Menge nichtlinearer Funktionen darstellen lassen. Dadurch wird das Problem erheblich vereinfacht. Eine solche Verfahrensweise zur Vereinfachnung ist gängig und findet z. B. auch bei Support-Vektor-Maschinen Anwendung [CV95]. Mit diesen Einschränkungen gelingt es, in jedem Fall ein globales Optimum für das nun vereinfachte Problem zu finden. Die Idee ist, den Algorithmus als Lernalgorithmus aufzufassen, und das globale Optimum für eine Reihe von Trainingsdaten zu finden. Die gewonnenen Parameter, welche die Lösung und somit das globale Optimum für die Trainingsdaten darstellen, werden auf reale Eingabedaten angewendet. Die Lösung für die Trainingsdaten stellt allerdings in der Regel nicht die optimale Lösung für die realen Eingabedaten dar, sondern wird als gute Näherung betrachtet, um die langsamsten Komponenten einer realen Probleminstanz zu finden. Im folgenden Abschnitt wird ein Algorithmus für das oben beschriebene eingeschränkte Optimierungsproblem vorgestellt.

2.3 Das vereinfachte 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 $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_J(\mathbf{x})]^T$. Wir definieren nun jedes Element der Eingabe-Ausgabe-Funktion $g_j(\mathbf{x})$ als gewichtete Summe über einer Menge von K ($K > \max(I, J)$) nichtlinearen Funktionen $h_k(\mathbf{x})$, d. h.

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

Dieser Schritt stellt eine nichtlineare Expansion dar. Sei $\mathbf{z}(t) := \mathbf{h}(\mathbf{x}(t))$. 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.

[2]Sonst müsste Gleichung (2.3) durch $\langle (y_j - \langle y_j \rangle)^2 \rangle = 1$ ersetzt werden

Setzt man dies in die Gleichung (2.1) ein, so erhält man

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

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.

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

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

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

$$\forall j' < j: \quad \langle y_{j'} y_j \rangle = {\mathbf{w_{j'}}}^T \underbrace{\langle \mathbf{z}\mathbf{z}^\mathbf{T} \rangle}_{=\mathbf{I}} \mathbf{w_j} = {\mathbf{w_{j'}}}^T \mathbf{w_j} = 0 \tag{2.8}$$

$\langle \mathbf{z}\mathbf{z}^\mathbf{T} \rangle$ 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 von $\langle \mathbf{z}\mathbf{z}^\mathbf{T} \rangle$ zu finden, die Δ_j minimieren. Dies sind gerade die normierten Eigenvektoren mit dem niedrigsten Eigenwert. Es gilt dann:

$$\Delta_j = \mathbf{w_j^T} \langle \dot{\mathbf{z}} \dot{\mathbf{z}}^T \rangle \mathbf{w_j} = \lambda_j \mathbf{w_j^T} \mathbf{w_j} = \lambda_j \tag{2.9}$$

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. und somit die Lösungen für das Optimierungsproblem.

Für eine tiefergehende mathematische Analyse des Problems siehe [Wis03].

Im folgenden werden verschiedene Schreibweisen für das unbearbeitete, das exakt anhand der Trainingsdaten normalisierte sowie ein ungefähr normalisiertes Signal verwendet. Das unbearbeitete Signal, also der direkte Input mit beliebigem Mittelwert und Varianz, wird mit $\tilde{\mathbf{x}}$ bezeichnet, das exakt normalisierte Signal mit einem Mittelwert von 0 und einer Varianz von 1 mit $\mathbf{x}$. Ein Test- oder reales Signal, welches mit den aus dem Lernverfahren gewonnenen Werten normalisiert wird, und dessen Mittelwert voraussichtlich nur ungefähr 0 und deren Varianz nur ungefähr 1 beträgt, wird mit $\mathbf{x}'$ bezeichnet (insbesondere bezeichnet $\mathbf{x}'$ nicht die Ableitung von $\mathbf{x}$).

2.4 Grundlagen

Bevor der SFA-Algorithmus im nächsten Abschnitt detailliert erläutert werden, sollen kurz zwei für diesen Algorithmus grundlegende Verfahren vorgestellt werden.

2.4.1 Principal Component Analysis

Die *Principal Component Analysis (PCA)* (zu deutsch: *Hauptkomponentenanalyse*) ist ein Verfahren aus der Statistik, welches in der Regel dazu verwendet wird, in Stichprobendatensätzen die Zufallsvariablen mit dem höchten Informationsgehalt zu finden. Dabei wird die Varianz als ein Maß für den Informationsgehalt angesehen. Häufig sind die Zufallsvariablen untereinander korreliert, so dass zudem eine Reduktion der untereinander korrelierten Zufallsvariablen auf unkorrelierte Hauptkomponenten gesucht ist.

Betrachten wir beispielsweise die Datensätze einer Umfrage, in der n Personen m Fragen beantwortet haben. Die Daten lassen sich in einer $n \times m$-Matrix auftragen und somit als mehrdimensionaler Vektorraum auffassen. Mathematisch gesehen wird nun bei der PCA eine Hauptachsentransformation durchgeführt, bei der der ursprüngliche Vektorraum in einen Vektorraum mit neuer Basis transformiert wird. Genauer gesagt findet eine Rotation statt, derart, dass die neuen Basisvektoren entlang der Richtung der Achsen mit der jeweils höchsten Varianz der Ursprungsdaten verlaufen; die erste Hauptkomponente zeigt also in die Richtung der höchsten Varianz, die zweite in die Richtung der zweithöchsten, etc. Meistens werden je nach Bedarf einige oder mehrere der letzten Komponenten weggelassen, um dadurch eine übersichtlichere Darstellung der betrachteten Daten zu erhalten.

Im einzelnen werden bei der PCA folgende Schritte durchgeführt:

1. **Eingangssignal:** Es ist ein n-dimensionaler Zufallsvektor von Stichproben $\mathbf{x}$ gegeben. (Dieser Zufallsvektor entspricht einer Matrix mit n Zeilen – dem Umfang der Stichprobe, z. B. Anzahl der Versuchspersonen – und m Spalten – den Merkmalen, z. B. der Anzahl der Fragen.)

2. **Mittelwertzentrierung:** Von jeder Zufallsvariable im Eingangssignal wird ihr Mittelwert abgezogen, so dass wir den mittelwertzentrierten Zufallsvektor $\tilde{\mathbf{x}}$ erhalten.

3. **Kovarianzmatrix:** Es wird die Kovarianzmatrix Σ von $\tilde{\mathbf{x}}$ berechnet.

4. **Eigenwerte und Eigenvektoren der Kovarianzmatrix:** Da die Kovarianzmatrix quadratisch ist, lassen sich von dieser die zugehörigen Eigenwerte λ_i und Eigenvektoren v_i, $i \in \{1, \ldots, n\}$, berechnen, mit der Eigenschaft:

$$\Sigma v_i = \lambda_i v_i$$

5. **Komponenten wählen:** Die Eigenwerte werden absteigend ihrer Größe nach sortiert und die ersten $k \leq n$ in der Diagonalmatrix Λ aufgeführt. Gemäß ihrer Eigenwerte werden auch die Eigenvektoren auf dieselbe Art geordnet, und ebenso die ersten k Eigenvektoren in der *Ladungsmatrix* Γ aufgeführt:

$$\Lambda = \operatorname{diag}(\lambda_{i_1}, \lambda_{i_2}, \ldots, \lambda_{i_k})$$
$$\Gamma = (v_{i_1}, v_{i_2}, \ldots, v_{i_k})$$

Mögliche Kriterien zur Bestimmung von k sind in der angegebenen Literatur zu finden.

6. **Hauptachsentransformation:** Die um die Hauptachsen rotierte und unter Umständen dimensionsreduzierte Ergebnismatrix $\mathbf{y}$ berechnet sich nun durch Multiplikation der transponierten Ladungsmatrix Γ^T (n Spalten) mit dem transponierten mittelwertzentrierten Zufallsvektor $\tilde{\mathbf{x}}^T$ (n Zeilen)

$$\mathbf{y} = \Gamma^T \tilde{\mathbf{x}}^T$$

Für eine detaillierte mathematische Betrachtung der PCA siehe z. B. [LF96].

Während die PCA wie oben beschrieben normalerweise benutzt wird, um die Komponenten mit dem höchsten Informationsgehalt zu finden, wird sie in der SFA dazu verwendet, die Komponenten mit der geringsten Veränderung zu finden. Wie im Abschnitt 2.5 vorgestellt, werden dann beim Schritt 5 der PCA nicht die k Komponenten mit den größten, sondern die Eigenvektoren mit dem *kleinsten* zugehörigen Eigenwerten ausgewählt.

2.4.2 Sphering

Beim *Sphering* oder *Whitening* handelt es sich um eine affine Transformation, um eine Menge von Zufallsvariablen zu dekorrelieren und zu normalisieren. Das bedeutet, dass ein n-dimensionaler Zufallsvektor von Stichproben $\mathbf{x}$ so transformiert wird, dass die Kovarianzmatrix des resultierenden Zufallsvektors $\mathbf{x_W}$ der Identitätsmatrix entspricht. Im einzelnen kann die so genannte Whiteningmatrix $\mathbf{S}$ berechnet werden durch

$$\mathbf{S} = \Gamma \Lambda^{-\frac{1}{2}},$$

wobei Γ der Matrix der Eigenvektoren und Λ der Diagonalmatrix der Eigenwerte der Kovarianzmatrix Σ entspricht; diese lassen sich, wie im letzten Abschnitt dargelegt, mit der PCA berechnen. Bei Λ handelt es sich um eine Diagonalmatrix, die Potenz berechnet sich durch $\mathrm{diag}(\lambda_{i_1}, \lambda_{i_2}, \ldots, \lambda_{i_k})^{-\frac{1}{2}} = \mathrm{diag}(\lambda_{i_1}^{-\frac{1}{2}}, \lambda_{i_2}^{-\frac{1}{2}}, \ldots, \lambda_{i_k}^{-\frac{1}{2}})$.

$\mathbf{x_W}$ berechnet sich nun durch

$$\mathbf{x_W} = \mathbf{S}^t \tilde{\mathbf{x}},$$

wobei $\tilde{\mathbf{x}}$ der mittelwertzentrierte Zufallsvektor ist. Somit ist $\mathbf{x_W}$ nicht nur unkorreliert, sondern zudem haben alle resultierenden Zufallsvariablen den Mittelwert 0 und eine Varianz von 1.

Für eine genauere Betrachtung siehe [DHS00].

2.5 Algorithmus

Im folgenden werden nun die Schritte des SFA-Algorithmus im einzelnen vorgestellt und erläutert:

1. **Eingangssignal:**

   Als Trainingsdaten ist ein I-dimensionales Eingangssignal $\tilde{\mathbf{x}}(t)$ gegeben.

2. **Normalisierung des Eingangssignals:**

   Zunächst wird das Eingabesignal normalisiert, somit erhält man

$$\begin{aligned} \mathbf{x}(t) &:= [x_1(t), \ldots, x_I(t)]^T && (2.10)\\ \text{mit} \quad x_i &:= \frac{\tilde{x_i}(t) - \langle \tilde{x_i} \rangle}{\sqrt{\langle (\tilde{x_i} - \langle \tilde{x_i} \rangle)^2 \rangle}} && (2.11)\\ \text{und somit} \quad \langle x_i \rangle &:= 0 && (2.12)\\ \text{sowie} \quad \langle x_i^2 \rangle &:= 1 && (2.13) \end{aligned}$$

   Dieser Normalisierungsschritt lässt sich wahlweise auch durch ein Sphering wie in Schritt 4 realisieren.

3. **Nichtlineare Expansion:**

   Um das expandierte Signal $\tilde{\mathbf{z}}$ zu erhalten, wird die nichtlineare Funktion $\tilde{\mathbf{h}}(\mathbf{x})$ gebildet. Verwendet man dafür lediglich die Monome vom Grad 1, so spricht man von der *linearen SFA* oder *SFA*^{1}, werden zusätzlich die Monome vom Grad 2 sowie die gemischten Terme verwendet, spricht man von der *quadratischen SFA* oder *SFA*^{2}.

   Im Falle der *SFA*^{2} gilt

$$\tilde{\mathbf{z}}(t) := \tilde{\mathbf{h}}(\mathbf{x}(t)) := [x_1(t), x_2(t), \ldots, x_I(t), x_1(t)x_1(t), x_1(t)x_2(t), \ldots, x_I(t)x_I(t)]^T \tag{2.14}$$

   und $\tilde{\mathbf{h}}$ bzw. $\tilde{\mathbf{z}}$ haben damit die Dimension $K = I + I(I+1)/2$.

4. **Sphering:**

   Nun muss das Signal $\tilde{\mathbf{z}}$ normalisiert werden, um den Algorithmus darauf anwenden zu können und die Bedingungen 2.6-2.8 zu erfüllen. Dazu wird ein so genanntes *Sphering* (siehe 2.4.2) durchgeführt:

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &:= \mathbf{S}(\tilde{\mathbf{h}} - \langle \tilde{\mathbf{z}} \rangle) && (2.15)\\ \text{bzw.} \quad \mathbf{z}(t) &:= \mathbf{S}(\tilde{\mathbf{z}} - \langle \tilde{\mathbf{z}} \rangle) && (2.16)\\ \text{mit} \quad \langle \mathbf{z} \rangle &= \mathbf{0} && (2.17)\\ \text{und} \quad \langle \mathbf{z}\mathbf{z}^\mathbf{T} \rangle &= \mathbf{I} && (2.18) \end{aligned}$$

   Dabei ist $\mathbf{S}$ die Spheringmatrix. Es wichtig anzumerken, dass diese Spheringmatrix zwar von den spezifischen Trainingsdaten abhängt, aber für die Anwendung auf Testdaten oder reale Probleminstanzen *nicht* neu berechnet wird.

   Statt einer PCA kann in diesem Schritt auch eine *Singular Value Decomposition (SVD)* angewendet werden. Wiskott und Sejnowski schreiben in [WS02], dass diese vor allem dann vorzuziehen ist, wenn einige Eigenwerte sehr nah an 0 sind.

5. **Hauptkomponentenanalyse (PCA):**

   Auf der Kovarianzmatrix des abgeleiteten Signals $\langle \dot{\mathbf{z}}\dot{\mathbf{z}}^T \rangle$ wird eine Hauptkomponentenanalyse durchgeführt, um die J Eigenvektoren mit den kleinsten Eigenwerten λ_j zu finden:

   $$\mathbf{w_j}: \qquad \langle \dot{\mathbf{z}}\dot{\mathbf{z}}^T \rangle \mathbf{w_j} = \lambda_j \mathbf{w_j} \tag{2.19}$$

   $$\text{mit} \qquad \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_J \tag{2.20}$$

   Damit erhalten wir die Eingabe-Ausgabe-Funktion

   $$\mathbf{g}(\mathbf{x}) := [g_1(\mathbf{x}), \ldots, g_J(\mathbf{x})]^T \tag{2.21}$$

   $$\text{mit} \qquad g_j(\mathbf{x}) := \mathbf{w_j}^T \mathbf{h}(\mathbf{x}) \tag{2.22}$$

   und das Ausgabesignal

   $$\mathbf{y}(t) := \mathbf{g}(\mathbf{x}(t)) \tag{2.23}$$

   welches die Nebenbedingungen aus den Gleichungen (2.2) (Mittelwert von 0), (2.3) (Varianz von 1) sowie (2.4) (Unkorreliertheit) erfüllt.

   Wie in Schritt 4 ist hier in der Regel die SVD der PCA vorzuziehen.

   Der folgende Schritt ist optional:

6. **Wiederholung:**

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

   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.

7. **Test / Anwendung:**

   In der Regel werden die vorhandenen Daten in eine Trainings- und eine Testdatenmenge aufgeteilt. Die Testdaten (bzw. die realen Daten) $\tilde{\mathbf{x}}'(t)$ werden zunächst approximativ normalisiert, allerdings mit den in Schritt 2 errechneten Parametern, d. h. unter Verwendung des Mittelwerts und der Varianz von $\tilde{\mathbf{x}}$, bzw. der für $\tilde{\mathbf{x}}$ errechneten Spheringmatrix. Als nächstes wird $\tilde{\mathbf{x}}'(t)$ nichtlinear expandiert (Schritt 3), um dann das expandierte Signal mit der beim Lernen erhaltenen Spheringmatrix aus Schritt 4 approximativ zu normalisieren. Zuletzt wird mit den in Schritt

5 errechneten Gewichtsvektoren $\mathbf{w_j}$ die Ausgabefunktion $\mathbf{y}'(t)$ berechnet mit den Eigenschaften:

$$\mathbf{y}'(t) := \mathbf{g}(\mathbf{x}'(t)) \tag{2.24}$$

$$\text{mit} \quad \langle \mathbf{y}' \rangle \approx \mathbf{0} \tag{2.25}$$

$$\text{und} \quad \langle \mathbf{y}' \mathbf{y}'^{\mathbf{T}} \rangle \approx \mathbf{I} \tag{2.26}$$

2.6 Demonstration des Algorithmus

Zur Veranschaulichung wurde das Beispiel aus Abbildung 2, [WS02] implementiert. Abbildung 2.3 zeigt dabei das Eingangs-, das Ausgangssignal sowie zwei Zwischenschritte. Als Eingabesignal wurden die Funktionen $\tilde{x_1}(t) = \sin(t) + \cos^2(11t)$ sowie $\tilde{x_2}(t) = \cos(11t)$ im Intervall $t \in [0, 2\pi]$ verwendet, welche in a) aufgezeichnet sind. Offensichtlich ist die langsamste Komponente des Eingangsvektors $\mathbf{x}(t) = [x_1, x_2]^T$, wie sich leicht nachrechnen lässt, die Funktion $y(t) = \tilde{x}_1(t) - \tilde{x}_2(t)^2 = \sin(t)$. Tatsächlich wird diese Komponenten auch durch die SFA gefunden (siehe d). (b) und (c) zeigen Teile des expandierten Signals vor und nach dem Spheringschritt.

Das Eingangssignal wurde für die Bearbeitung durch den Algorithmus diskretisiert. Dabei lieferte in diesem Beispiel bereits eine Abtastrate von 50 Schritten gute Lösungen zur Herstellung der langsamsten Komponente $\mathbf{y}_1$, für eine gute Darstellung schnellerer Komponenten müssen höhere Abtastraten verwendet werden.

Für Details zur Implementation siehe Anhang.

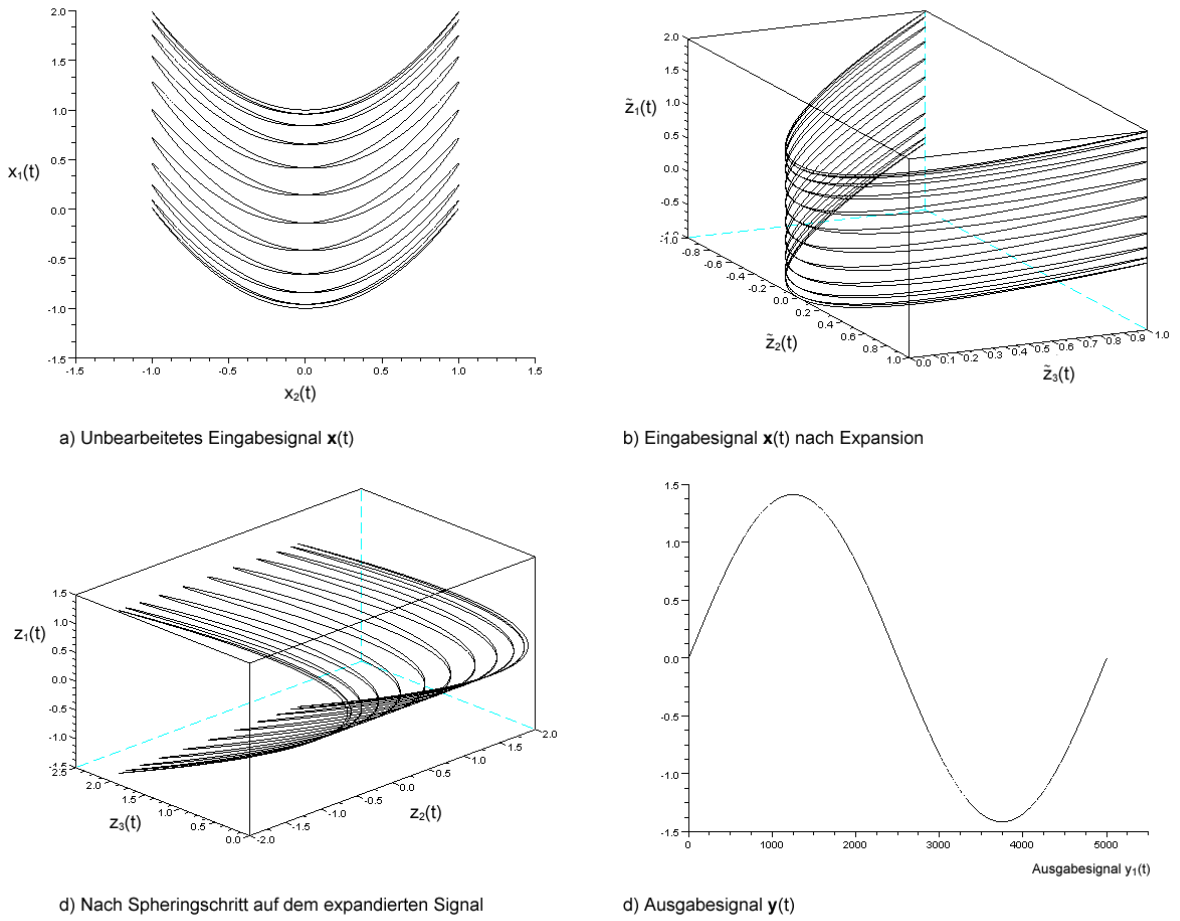

Abbildung 2.3: Plots nach verschiedenen Schritten der SFA. **a)** Das Eingangssignal $\mathbf{x}(t)$. **b)** Expandiertes Eingabesignal, in der Grafik sind $\tilde{z_1}(t) := x_1(t)$, $\tilde{z_2}(t) := x_2(t)$ und $\tilde{z_3}(t) := x_2^2(t)$ zu sehen. **c)** Expandiertes Signal nach dem Sphering. Man sieht deutlich die Drehung der Funktion im Raum durch die Hauptachsentransformation. **d)** Ausgabesignal $\mathbf{y}(t)$, welches der normalisierten Version der Funktion (d. h. Mittelwert von 0, Varianz von 1) $sin(t)$ entspricht.

Kapitel 3

Anwendung der SFA auf Robotersensordaten

Da sich die SFA als eine mächtige Methode zur Erkennung von Invarianten und langsamen Prozessen zu eignen scheint, liegt der Gedanke nah, das Verfahren auf die Robotik anzuwenden. In der Robotik ist es bei der Entwicklung von Steuerelementen sehr wichtig, langsame Veränderungen des Zustands des Roboters zu erkennen. Beispielsweise ist es beim Laufen notwendig aus dem sensorischen Eingaben zu extrahieren, wann der Roboter in eine gefährliche Lage gerät und umzufallen droht, damit entsprechende gegensteuernde Bewegungen durchgeführt werden können.

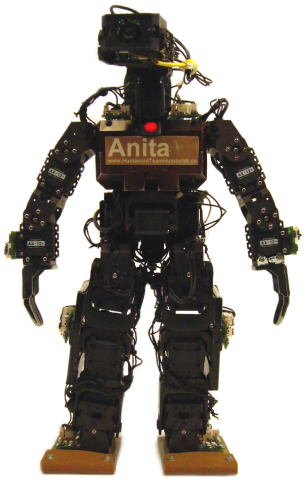

Abbildung 3.1: Anita, Roboter der A-Serie

Als humanoide Roboterplattform wird in dieser Arbeit die an der Humboldt-Universität Berlin entwickelte A-Serie verwendet, die im Rahmen des ALEAR-Projektes (*Artificial Language Evolution on Autonomous Robots*, `http://www.alear.eu`) enstanden sind.

In den folgenden Abschnitten wird zunächst kurz näher auf die verwendete Roboterplattform eingegangen. Daraufhin werden Roboterlaufsequenzen mit Hilfe der SFA analysiert und die daraus abgeleiteten Ergebnisse vorgestellt.

3.1 Plattform

Die A-Serie ist eine robuste humanoide Roboterplattform, welche mit einem visuellen System und propriozeptorischen Sensoren ausgestattet ist. Abbildung 3.2 zeigt den schematischen Aufbau der A-Serie, bei welcher für die Analyse mittels SFA vor allem die 8 so genannten *Accelboards*, welche mit grünen Pfeilen gekennzeichnet sind, interessant sind. Auf jedem Accelboard befinden sich neben einer Recheneinheit, welche für die Ansteuerung der Motoren zuständig ist, zwei Beschleunigungssensoren, jeweils einer in x- und in y-Richtung; dabei sind die Sensoren der x-Richtung alle nach vorn in der sagittalen

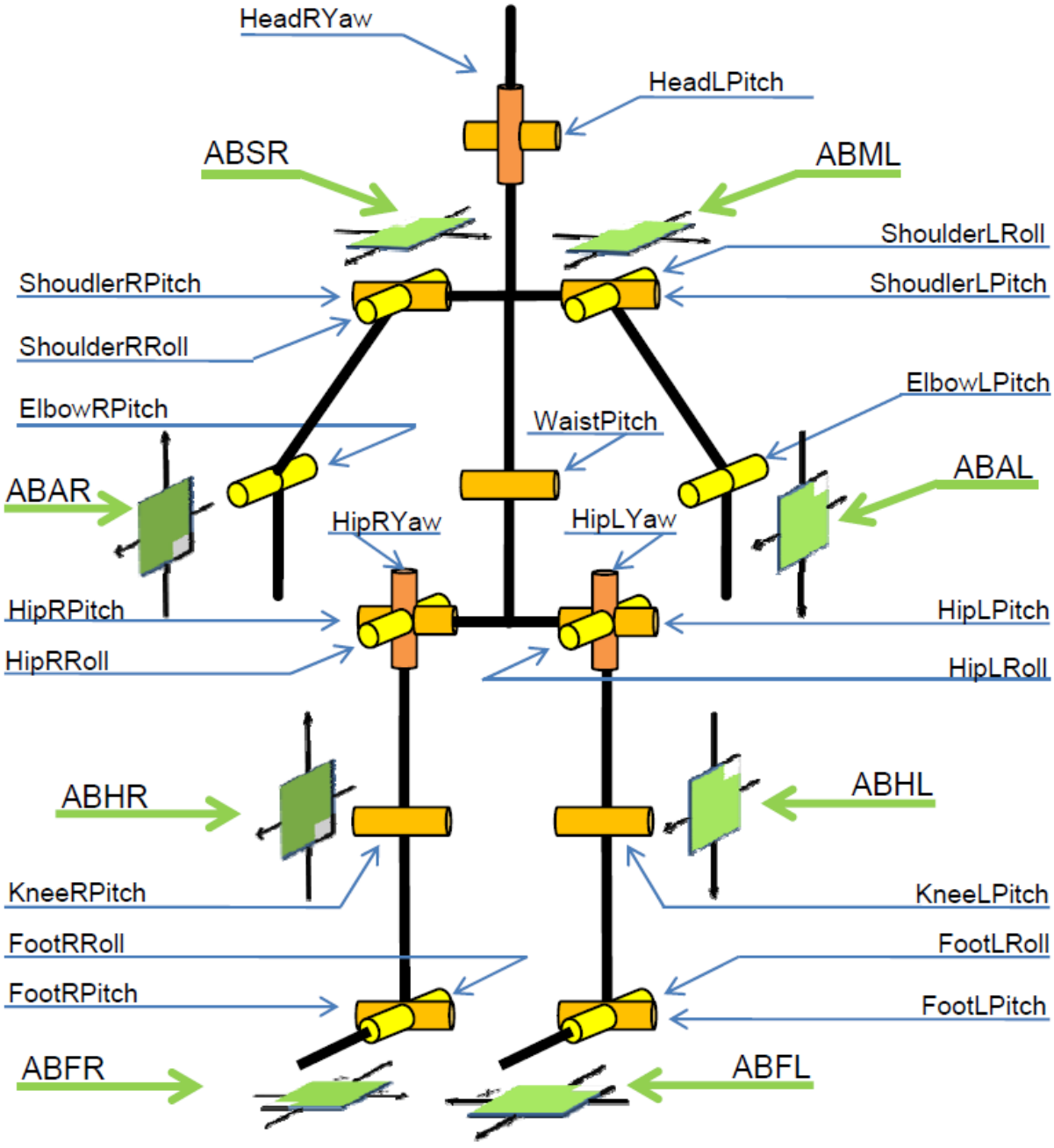

Abbildung 3.2: Schematischer Aufbau der Humanoiden-A-Serie

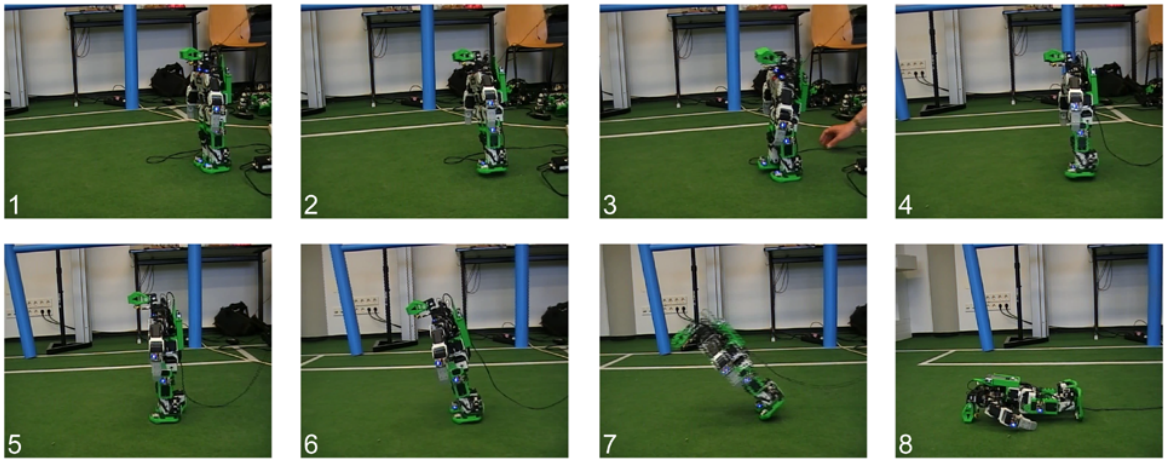

Abbildung 3.3: Roboter vollführt eine Laufbewegung und fällt zu Boden.

Ebene (von hinten nach vorne) ausgerichtet, während die Sensoren der y-Richtung teils auf senkrecht zur Transversalebene (also nach oben bzw. unten) und teils auf parallel zur frontalen Ebene (links nach rechts, bzw. rechts nach links) auftretende Veränderungen reagieren. Die Sensoren geben in einem Wertebereich von 16 bit Beschleunigungswerte zurück, aus welchen der Roboter auf seine Lage im Raum, Geschwindigkeit, etc. schließen kann.

Zum Zeitpunkt dieser Arbeit sind bereits diverse Verhaltensmuster, wie Zeigebewegungen, Laufen, Aufstehen und Hinlegen etc. implementiert, welche eine große Bandbreite an analysierbaren Daten zur Verfügung stellt. In dieser Arbeit werde ich mich auf die Betrachtung eines konkreten Laufmusters konzentrieren, um zu erörtern, inwiefern mit Hilfe der SFA eine Umfalldetektion zu realisieren möglich ist. Dabei werden Daten von verschiedenen Robotern der Serie verwendet, namentlich Aida, Aimee, Anita und April.

Für weitere technische Daten zur A-Serie siehe Anhang.

3.2 Einführende Betrachtung

Da a priori nicht klar ist, welche Ergebnisse durch Anwendung der SFA auf die Beschleunigungssensordaten man erhält, ist es hilfreich, sich zunächst zu überlegen, welche Art von Informationen man in den langsamsten Komponenten eines Vektors von Beschleunigungsdaten erwarten kann. Dies ist für die Auswertung der Ergebnisse unabdingbar, sobald man erkennen will, welche Informationen tatsächlich in den Daten kodiert sind.

Abbildung 3.3 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. Dabei wird zunächst eine Schwingung in der frontalen Ebene erzeugt, die den Roboter

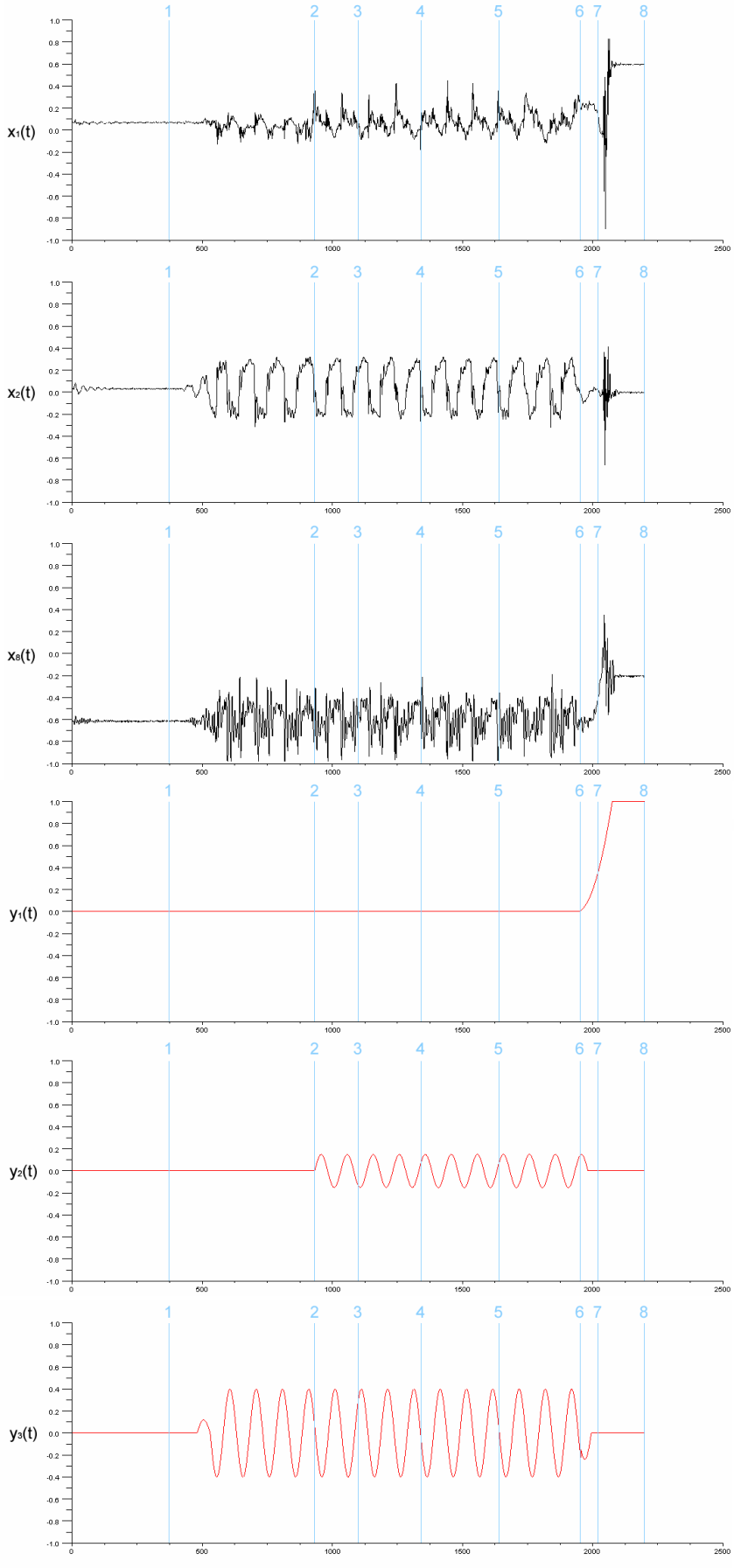

Abbildung 3.4: x_1, x_2, x_3: Daten von drei Beschleunigungssensoren. y_1, y_2, y_3: Mögliche gesuchte langsamste Komponenten. Siehe Text für Details.

das Gewicht von einem Bein auf das andere verlagern lässt. Dies dient als Grundlage, um die Vorwärtsbewegung der Beine des Roboters in der Sagittalebene zu starten, welches dann das Laufverhalten in Gang setzt.

Leider kann es bei diesem Laufmuster vorkommen, dass der Roboter instabil wird und umfällt. Dabei stolpert der Roboter quasi über seine eigenen Füße, da zum jetzigen Zeitpunkt keine zusätzliche Reflexschleife implementiert ist, die die Stabilität der Bewegung überwacht. Da der Roboter häufiger auf rauhen Untergründen fällt, besteht die Vermutung, dass der Roboter mit einem seiner Füße am Boden hängenbleibt.

Zu acht Zeitpunkten der Sequenz wurden Momentaufnahmen gemacht, vor dem Start der Laufbewegung, während des Laufens sowie kurz vor, während und nach dem Umfallen. Abbildung 3.4 zeigt oben die Daten von drei Beschleunigungssensoren während der vollführten Bewegung, die Stellen, an denen die Momentaufnahmen gemacht wurden, sind in den Graphen markiert. $x_1(t)$ und $x_2(t)$ sind die in sagittale bzw. in frontale Richtung zeigenden an der linken Schulter angebrachten Sensoren `ABML x` und `ABML y`; $x_8(t)$ ist der Sensor `ABAR y`, der sich am rechten Arm befindet und senkrecht nach oben weist. In Eingabesignal $x_2(t)$ ist deutlich die frontale Schwingung zu erkennen, Signal $x_1(t)$ lässt eine sagittale Pendelbewegung erahnen. $x_8(t)$ lässt ebenfalls die Zyklizität des Gangmusters erkennen, variiert aber nicht so stark. Insgesamt sind die Daten stark verrauscht, was neben der Ungenauigkeit der Sensoren auch auf die Eigenresonanz des Gesamtkörpers zurückzuführen ist. Deutlich zu erkennen ist außerdem, wie an Zeitpunkt sechs die Schwingung in allen Signalen abbricht, da hier der Roboter das Gleichgewicht verliert und daraufhin zu Boden fällt. Der folgende Sturz löst eine starke Erschütterung aus, welche sich durch einen starken Ausschlag in allen Sensoren widerspiegelt.

Welche langsamen Komponenten, die sich aus den Beschleunigungssensoren extrahieren lassen könnten, sind nun vorstellbar bzw. gesucht? Bis zum Fall des Roboters ist die aufrechte Position des Roboters sicherlich eine Invariante. So ist vorstellbar, wie in Abbildung 3.4 in y_1 angedeutet, dass die SFA eine Komponente findet, welche der Position (d. h. aufrecht, liegend oder in Schieflage) des Roboters entspricht. Diese ist bereits teilweise in x_8 repräsentiert, jedoch ist dieses Eingangssignal stark durch die Schwingungen in sagittaler und frontaler Ebene verrauscht.

Eine kurze theoretische Vorüberlegung zeigt, dass es durchaus plausibel ist anzunehmen, dass die quadratische SFA eine solche Komponente extrahieren kann: Betrachten wir zwei Sensoren, welche auf dem gleichen Accelboard liegen, und bezeichnen deren Werte mit v_1 und v_2. Diese Sensoren stehen orthogonal zueinander und spannen einen zweidimensionalen Vektorraum auf. Nun lässt sich ein durch die beiden Sensoren im zweidimensionalen Raum bezeichneter Punkt auch in Polarkoordinaten angeben, womit sich für einen Punkt P die Darstellung $P = r\cos(\phi)\binom{1}{0} + r\sin(\phi)\binom{0}{1}$ ergibt. Betrachtet man nun die Sensoren wieder getrennt, so ergeben sich $v_1 = r\cos(\phi)$ und $v_2 = r\sin(\phi)$. Nehmen wir an, die quadratische SFA verwendet für eine langsamste Komponente $\tilde{y}$ nur die Sensoren v_1 und v_2, bzw. jeweils deren Quadrate, und gewichtet diese zudem jeweils mit dem gleichen Faktor k, so würde sich $\tilde{y}$ durch folgende Gleichung ergeben:

$$\tilde{y} := v_1^2 + v_2^2 = kr^2\cos(\phi)^2 + kr^2\sin(\phi)^2 = kr^2\underbrace{(\cos(\phi)^2 + \sin(\phi)^2)}_{=1} = kr^2$$

Damit würde durch diese Summe die Richtungsinformation der Sensoren vollständig herausfallen, und sich eine Komponente ergeben, welche nur die Amplitude widerspiegelt. Bis auf den Jitter, welcher durch die Erschütterung beim Umfallen entsteht, wäre diese Komponente also schon recht nah an der von uns gesuchten Komponente y_1.

Weitere denkbare Komponenten wären die bereinigten Schwingungen in sagittaler (in Abbildung 3.4 y_2) sowie frontaler Ebene (in 3.4 y_3). Voraussichtlich wird die Schwingung in sagittaler Ebene allerdings nicht so sinusförmig, sondern etwas zackiger als in der Grafik angedeutet sein.

Besonders interessant wäre vor allen Dingen, wenn eine dieser Komponenten tatsächlich schon dann auffälliges Verhalten zeigt, sobald der Roboter *droht* umzufallen. Diese Komponente könnte in die Reflexschleife eingebunden werden, um damit den Roboter rechtzeitig zu stabilisieren. Daher wird besonders darauf zu achten sein, eine solche Warnkomponente zu finden.

3.3 Methodik und betrachtete Daten

Die folgenden Abschnitte befassen sich tiefergehend mit der Analyse mehrerer aufgenommener Laufsequenzen unterschiedlicher Roboter der A-Serie. Alle Sequenzen wurden mit dem Verfahren aus [Wer08] generiert. Es werden zuerst verschiedene Kombinationen und Zusammenstellungen von SFA-Modulen probiert, um für die Beschleunigungsdaten eine möglichst optimale Lernkonfiguration zu finden. Dazu wird zunächst auf jeweils identischen Daten gelernt und getestet, womit Verzerrungen aufgrund von mangelnder Generalisierung der gelernten Parameter ausgeschlossen sind, so dass die Semantik der gefundenen Komponenten möglicherweise leichter herauszufinden ist.

Nachdem mögliche Komponenten auf den gleichen Lern- und Testdatensätzen gefunden wurden, wird einerseits herauszufinden sein, inwiefern die gelernten Parameter für einen einzelnen Roboter zu verschiedenen Zeitpunkten stabile Ergebnisse liefern, anderseits, inwieweit die auf einem Roboter gelernten Parameter auf andere Roboter gleicher Bauart übertragbar sind. Die Generalisierung zwischen den Roboterplattformen ist von großer Wichtigkeit, nicht nur weil die Roboter den gleichen Aufbau haben und somit viel Lernaufwand gespart werden könnte, wenn die gelernten Parameter nicht nur auf einem Roboter funktionieren würden, sondern auch weil sich der Zustand des Roboters im Laufe der Zeit durch Verschleiß verändert, was die Anwendbarkeit der gelernten Parameter gefährdet.

Als ein Gütemaß für die durch die SFA berechneten Komponenten wird der in [WS02] vorgestellte η-Wert verwendet:

$$\eta(y) := \frac{T}{2\pi}\sqrt{\Delta(y)} \tag{3.1}$$

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, damit der η-Wert nicht durch einen Skalierungsfaktor verfälscht wird.

Desweiteren wird als Maß für die Ähnlichkeit von zwei Signalen der Korrelationskoeffizient verwendet, z. B. um herauszufinden, welche Eingabewerte mit welchen ausgegebenen Komponenten korrelieren, 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

$$\rho := \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}} \tag{3.2}$$

wobei $\mathrm{Var}(X)$ die Varianz von X und $\mathrm{Cov}(X, Y)$ die Kovarianz von X und Y bezeichnen. In der Regel interessiert uns allerdings lediglich der Betrag. Ein absoluter Wert nahe 1 bedeutet hohe Korrelation, ein Wert nahe 0 geringe.

3.4 Komponentenanalyse in Abhängigkeit von der Lernstruktur

In diesem Abschnitt sollen zwei eng mit einander verbundene Fragen beantwortet werden: Zum einen welche Informationen die SFA überhaupt aus Beschleunigungssensordaten eines Roboters finden kann, zum anderen mit welcher Lernstruktur diese Informationen am besten und effizientesten gefunden werden. Zu diesem Zweck wurden verschiedene Lernstrukturen getestet, und es wurde versucht die gefundenen Komponenten zu interpretieren.

Die naheliegendste Idee für eine Lernstruktur ist, die SFA einmal oder wiederholt auf der Gesamtheit der Beschleunigungssensordaten anzuwenden. Wie zuvor erwähnt liegt eine besondere Stärke der SFA darin, dass durch Wiederholung höherdimensionale Signale gefunden werden können, ohne eine exponentielle Berechnungszeit in Kauf nehmen zu müssen. Zusätzlich kann die wiederholte SFA die gefundenden Komponenten glätten und entrauschen. Die Ergebnisse werden im Abschnitt 3.4.2, *Simples wiederholtes Lernen* vorgestellt.

Im nächsten Schritt wird betrachtet, welche Ergebnisse man erhält, wenn man nur auf einer Teilmenge der Beschleunigungssensordaten lernt. Dazu wurden die Sensoren nach ihrer jeweiligen Ausrichtung getrennt und Sensoren gleicher Ausrichtung jeweils einzeln wiederholt durch die SFA geschickt. Der Abschnitt 3.4.3, *Simples wiederholtes Lernen auf beschränkten Daten* fasst die daraus gewonnenen Erkenntnisse zusammen.

Als letztes wird eine hierarchische SFA-Struktur evaluiert, ähnlich wie sie von Sejnowski und Wiskott zum Finden von Invarianten in visuellem Input verwendet wurde. Die Idee ist, zunächst wie in der vorigen Struktur gesondert auf andersgerichteten Sensoren zu lernen, dann aber die einzelnen SFA-Ausgaben wieder zusammenzuführen und nochmals zusammengefasst durch eine oder mehrere SFA-Schichten zu schicken. Die Hoffnung ist, dass sich durch die getrennte Vorverarbeitung eine bessere Auftrennung in die verschiedenen anatomischen Ebenen des Roboters ergibt, und die Zusammenfassung schließlich

noch abstraktere Komponenten findet. Der Abschnitt beschäftigt sich genauer mit dieser Lernstruktur.

3.4.1 Trainings- und Testdaten

Als Lerndaten wurden zwei Sequenzen analysiert: Eine ca. 17-sekündige Laufsequenz des Roboters April mit anschließendem Umfallen nach vorne sowie eine ähnliche ca. 35-sekündige Sequenz des Roboters Aida. Es wurden zum Lernen und Testen jeweils die gleichen Daten verwendet, d. h. die aus der April-Laufsequenz gelernten Parameter wurden nur auf die April-Laufsequenz angewendet, das gleiche bei der Aida-Sequenz. In folgenden werden vor allem Ergebnisse aus der April-Laufsequenz vorgestellt, Ergebnisse der Aida-Sequenz werden zum Vergleich herangezogen und erwähnt, falls sie Abweichungen aufweisen.

Die Sequenz von April weißt folgende Charakteristika auf, welche ganz oder teilweise von der SFA in den Beschleunigungssensordaten gefunden werden sollten[1]: Unmittelbar mit dem Start der Aufzeichnung beginnt das Pendeln in frontaler Ebene, bei $t = 300$ wankt der Roboter deutlich stärker rechts, bei $t = 400$ beginnt das Vorwärtslaufen. Daraufhin läuft der Roboter 10 Sekunden, bis er bei $t = 1400$ nach vorne umfällt. Die jeweils gleich gerichteten Schultersensoren `ABSR` und `ABML` weisen eine sehr starke Korrelation von $0,99$ in frontaler (`ABML y` mit `ABSR y`) und $0,96$ in sagittaler Richtung (`ABML x` mit `ABSR x`) auf. Die sagittalen Schultersensoren korrelieren außerdem sehr stark mit den sagittalen Hüftsensoren ($\approx 0,88$), weswegen bei Betrachtung von Zusammenhängen zwischen Beschleunigungssensoren und SFA-Ergebnissen sich lediglich auf diese Schultersensoren bezogen wird.

Die frontalen und sagittalen Schultersensoren sind untereinander sehr schwach korreliert, der Korrelationskoeffizient von `ABML x` und `ABML y` beträgt lediglich $-0,06$.

In der Sequenz von Aida steht der Roboter einige Sekunden, bevor bei $t = 300$ das Pendeln in frontaler Ebene beginnt. Bei $t = 900$ startet der Roboter das Laufen. bei $t = 1750$ sieht man ein starkes Wanken nach rechts, bis der Roboter dann bei $t = 2000$ nach vorne umfällt. Nachdem der Roboter einige Sekunden liegt, wird er bei $t = 2350$ wieder aufgerichtet, wenig später endet die Aufzeichnung. Wie bei der April-Sequenz sind die Schultersensoren mit gleicher Ausrichtung stark korreliert, zwischen den verschiedenen Richtungen besteht eine stärkere Abhängigkeit, der Korrelationskoeffizient von `ABML x` und `ABML y` beträgt hier $-0,21$.

3.4.2 Simples wiederholtes Lernen

Bei der ersten Lernmethode werden ein bis mehrere SFA^2-Module hintereinandergeschaltet und auf der Gesamtheit der verfügbaren Beschleunigungssensoren angewandt. Dabei kann sowohl die Anzahl der SFA-Runden, als auch die Anzahl der von einer Runde zur

[1] Im folgenden sind immer die Zeitschritte gemäß der Sensordaten angegeben. Da die Sensoren eine Abtastrate von 100 Hz haben, entsprechen 100 Zeitschritte 1 Sekunde.

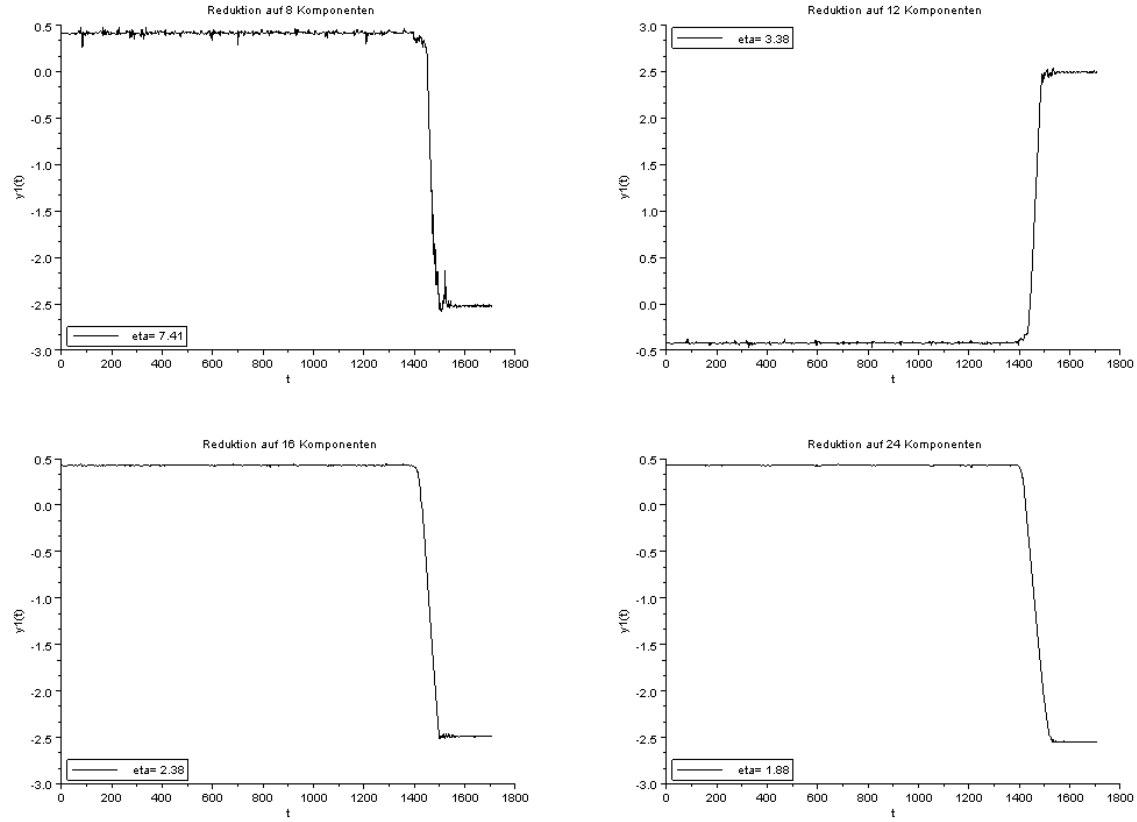

Abbildung 3.5: Langsamste durch die sechs mal wiederholte SFA entdeckte Komponente der Anita-Lauf-Fall-Sequenz mit Reduktion auf 8, 12, 16 und 24 Komponenten pro Runde.

nächsten weitergereichten Komponenten variiert werden. Zunächst halten wir eine Anzahl von sechs Iterationen fest und betrachten, wie sich die langsamste Komponente bei Weiterreichen der 8, 12, 16 und 24 langsamsten Komponenten pro Runde verhält. Abbildung 3.5 zeigt die langsamste Komponente y_1 jeweils nach einer variierten Anzahl von weitergereichten Komponenten[2]. Sie erinnert sehr stark an die gesuchte langsamste Komponente in Abbildung 3.4 und korrespondiert offensichtlich mit der Position des Roboters im Raum. Die stärkste Verbesserung brachte der Sprung von 8 auf 12 Komponenten. Mit mehr als 24 weitergereichten Komponenten verbesserten sich die Ergebnisse nicht mehr maßgeblich.

In den folgenden Anwendungen wurden zunächst nach jeder Runde 24 Komponenten weitergereicht, da die Charakteristik der Kurven so gut sichtbar war. Bei der praktischen Anwendung sollte allerdings die Anzahl der pro Schritt beibehaltenen Komponenten reduziert werden, um die Berechnung schneller zu machen.

Bzgl. der Anzahl der Iterationen wurden gute Ergebnisse mit sechs SFA^2-Runden erzielt. Mehr als sechs Iterationen verbesserten die Komponenten, gemessen am optischen

[2]Zwischen den verschiedenen Iterationsstufen ist zudem eine Vorzeichenumkehr zu erkennen, welche mit dem Algorithmus zur Berechnung der Eigenvektoren zusammenhängt.

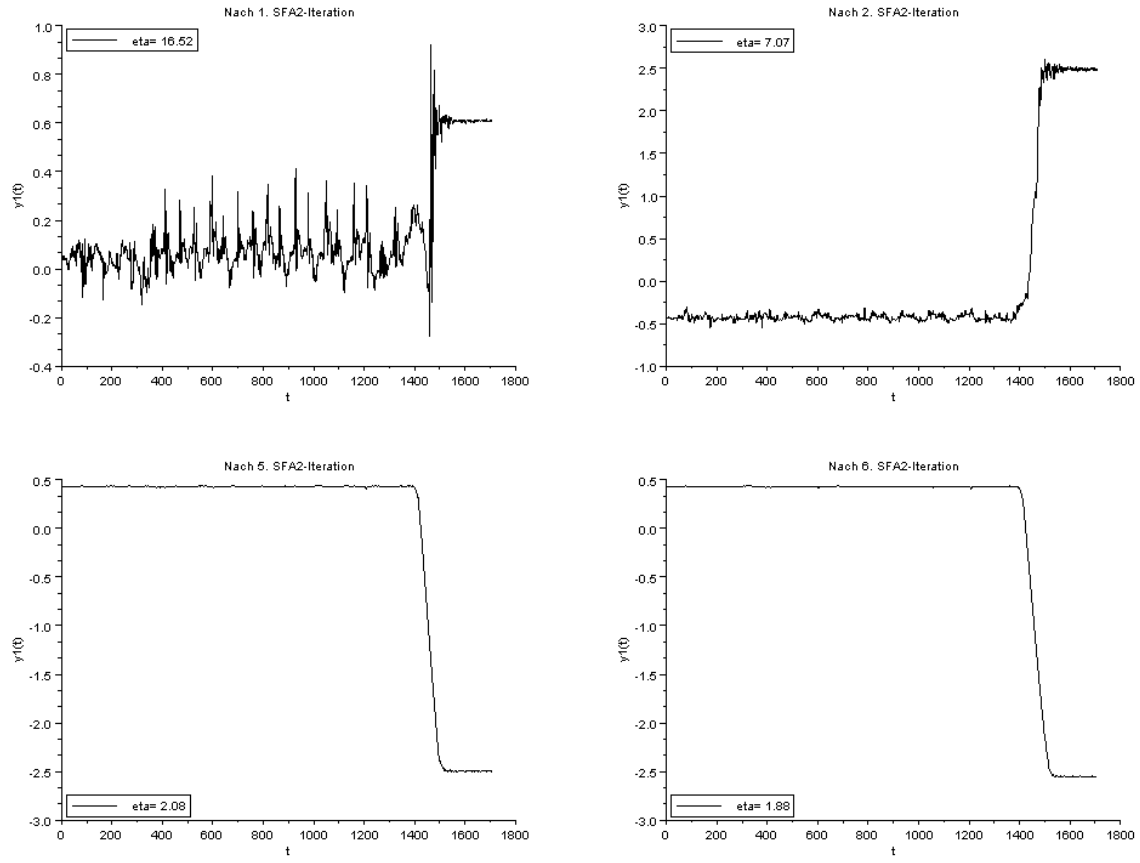

Abbildung 3.6: Langsamste durch die hierarchische SFA entdeckte Komponente der Anita-Lauf-Fall-Sequenz nach 1, 2, 5 und 6 SFA^2-Iterationen.

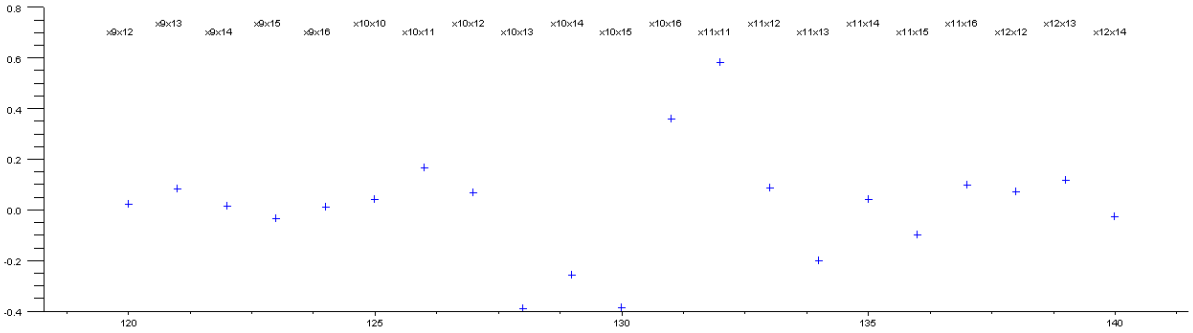

Abbildung 3.7: Ausschnitt aus dem Koeffizientenvektor $\mathbf{w_1}$; es sind die am stärksten gewichteten Eingangssensoren nach der ersten SFA^2-Runde für y_1 aufgetragen.

Eindruck und den η-Werten, nicht maßgeblich. Abbildung 3.6 zeigt die jeweils langsamste Komponente nach verschiedenen Iterationen. Es lässt sich deutlich mit dem Auge wie auch am η-Wert erkennen, wie die Kurve durch die weiteren Iterationen immer weiter geglättet wird, maßgeblich in der zweiten Iteration.

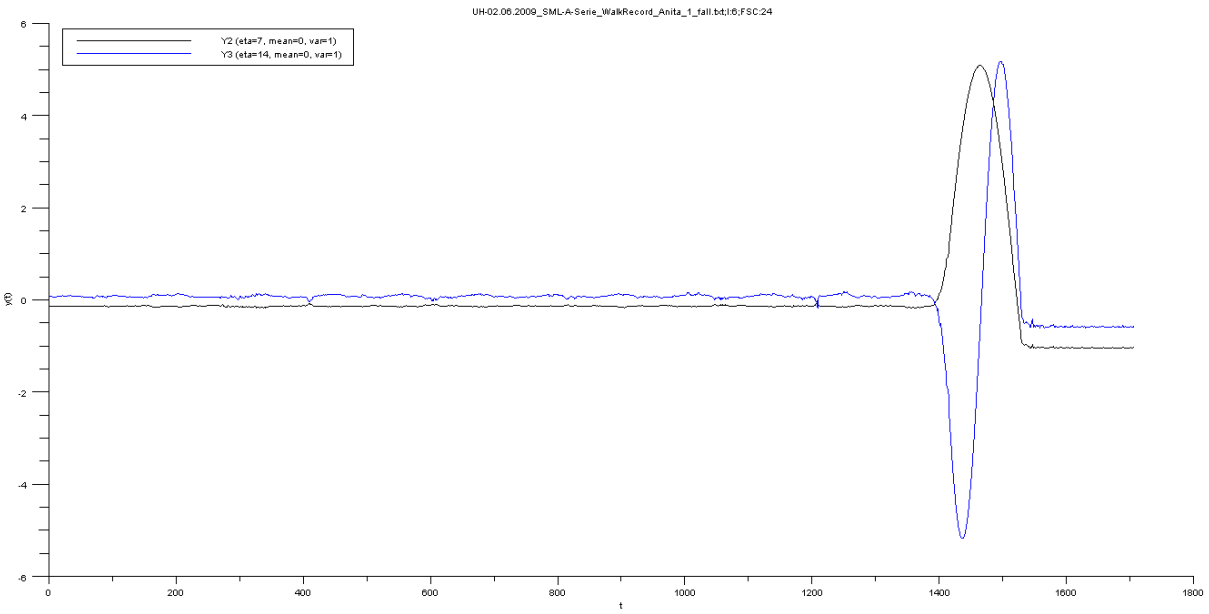

Abbildung 3.8: Zweit- und drittlangsamste durch die hierarchische SFA entdeckte Komponente der Anita-Lauf-Fall-Sequenz nach 6 SFA^2-Iterationen.

Um herauszufinden, welche Sensoren zumindest nach der ersten Runde am stärksten für die langsamste Komponente benutzt wurden, kann man sich die Koeffizientenvektoren $\mathbf{w_j}$ für y_j ansehen. Eine Betrachtung dieser ergab, dass bei der Anita-Laufsequenz für die langsamste Komponente zwar andere Komponenten verwendet wurden als bei der Aida-Laufsequenz, allerdings bei beiden vorrangig Sensoren mit transversaler und sagittaler, weniger frontaler Ausrichtung. Abbildung 3.7 zeigt die am stärksten gewichteten Eingabesensoren, welche auf der Anita-Sequenz in der ersten SFA-Runde gelernt wurden. Während hier das Monom x_{11}^2 dominierte, war es bei der Aida-Laufsequenz $x_{10}x_{12}$ (beide transversal).

Vergleicht man die Koeffizienten über die verschiedenen Iterationen und in Abhängigkeit von der Anzahl der weitergereichten Komponenten hinweg, so fällt auf, dass eher jeweils die späteren SFA-Komponenten, also y_{20}-y_{24} hohe Gewichte erhalten.

Die zweit- und drittlangsamsten Komponenten sind in Abbildung 3.8 zu sehen. Die Komponenten unterscheiden sich von der langsamsten lediglich dadurch, dass sie nach dem Umfallen stärker oszillieren. Zunächst sind diese Komponenten nicht redundant, insofern als dass sie die geforderte Dekorrelationsbedingung erfüllen. Dass diese stärker oszillierenden Komponenten gefunden werden können, lässt sich folgendermaßen erklären: Die langsamste Komponente springt, sobald der Roboter umfällt, von einem Ausgangswert auf einen anderen. Zwar geschieht dies hier in einigen Zeitschritten und kann durch die Zwischenzustände des Roboters beim Fallen interpoliert werden, doch nehmen wir an, es würde sich um eine sofort umschaltende Sprungfunktion handeln. Zerlegt man die Sprungfunktion mit der Fourieranalyse in ihre Basisfrequenzen, so erhält man eine unendliche Anzahl von Basisfunktionen, nämlich das gesamte Frequenzspektrum, als Ergebnis; dabei sind natürlich die meisten Basisfunktionen stark gedämpft. Die Idee ist also, dass die

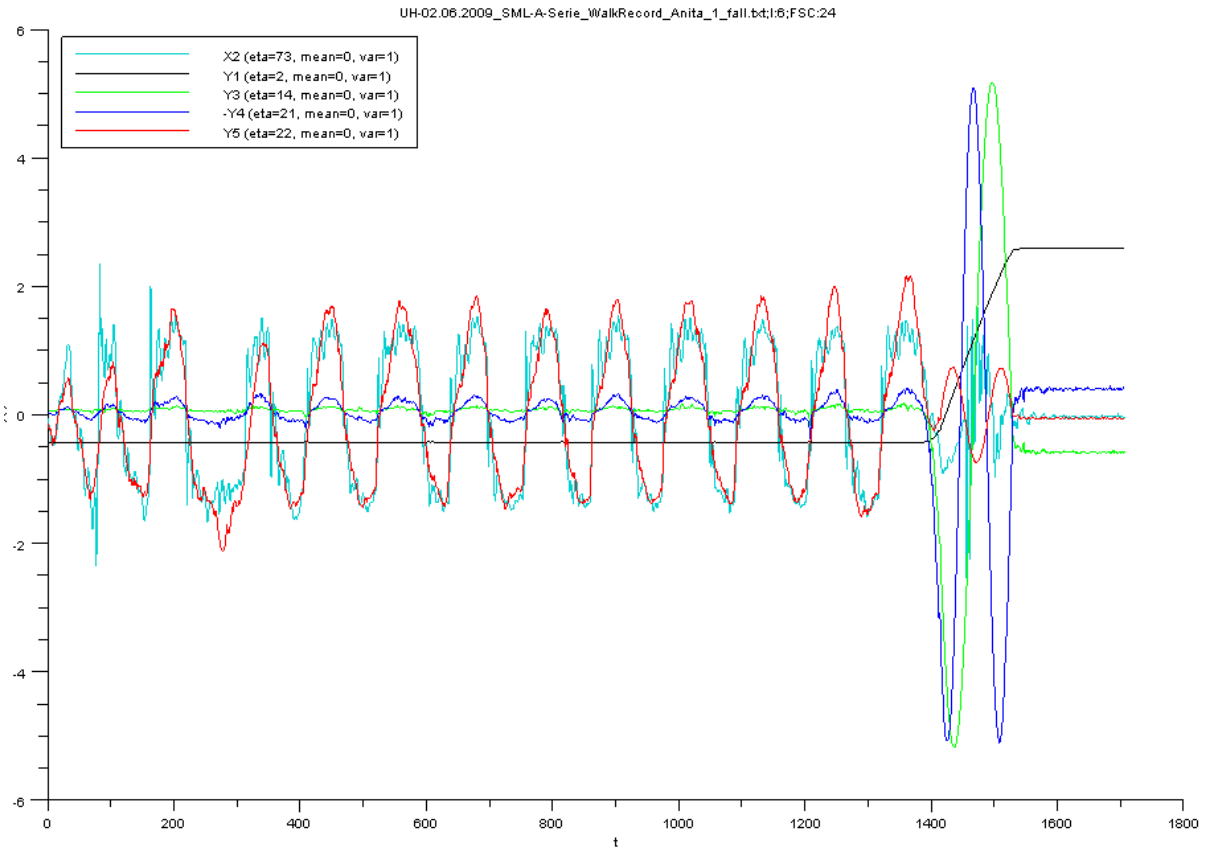

Abbildung 3.9: Frontale Komponente, durch die simple wiederholte SFA in der Anita-Lauf-Fall-Sequenz nach 6 SFA^2-Iterationen gefunden, sowie der frontale an der Schulter befestigte `ABML y`-Beschleunigungssensor.

SFA bei den betrachteten Komponenten die Parameter findet, die andere Basisfunktionen des Frequenzspektrums der Sprungfunktion stärker gewichten. Betrachtung weiterer Komponenten zeigt auch, dass sich dies beliebig weit fortzusetzen scheint, da an der Stelle, an welcher der Roboter umfällt, in anderen Komponenten noch stärkere Oszillationen auftreten können.

Sagittale und frontale Komponenten

Als nächstes sollten Komponenten für die frontale und sagittale Pendelbewegung des Roboters beim Laufen ausfindig gemacht werden. Dazu wurde der Korrelationskoeffizient zwischen `ABML y` und den SFA-Ausgabekomponenten nicht auf den gesamten Daten, sondern lediglich einem Ausschnitt zwischen $t = 1$ bis $t = 1200$ betrachtet. Dieser Abschnitt beinhaltet nicht das Umfallen, sondern lediglich die Stehen, laterales Einschwingen und Laufen. Damit können auch Komponenten ausfindig gemacht werden, die zwar die frontale Pendelbewegung gut extrahieren, aber beim Umfallen ähnlich wie die zweit- und drittlangsamste Komponente im Vergleich zu den Ursprungsdaten völlig anderes Verhalten aufweisen. Auf dem genannten Abschnitt zeigen y_4 und y_5 eine hohe Korrelation von $0,93$ bzw. $0,90$ zu `ABML y`; die Ähnlichkeit ist in Abbildung 3.9 deutlich zu erkennen, auch

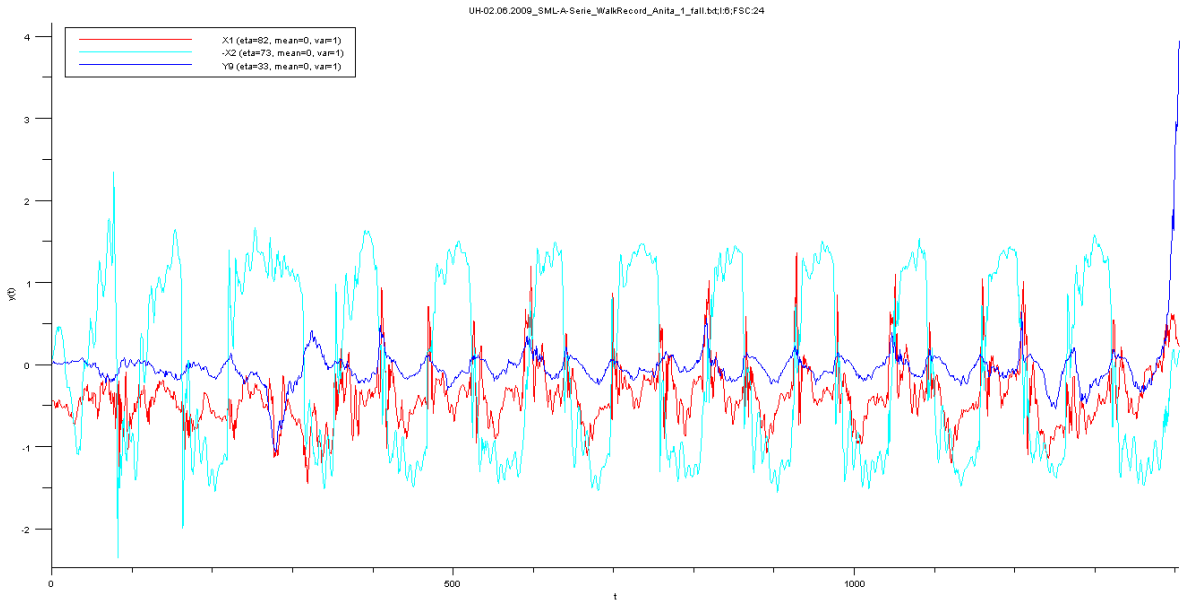

Abbildung 3.10: Sagittale Komponente, durch die simple wiederholte SFA in der Anita-Lauf-Fall-Sequenz nach 6 SFA^2-Iterationen gefunden.

wenn die Signale stark geglättet sind.

Eine sagittale Komponente zu finden war etwas schwieriger, da diese offensichtlich nicht so sauber von der SFA extrahiert worden ist. Die beste Korrelation auf dem Laufabschnitt von `ABML x` mit den SFA-Ergebnissen ergab sich mit $\rho = 0,37$ in Komponente y_9; ein kurzer Ausschnitt ist in Abbildung 3.10 zu sehen. Zum besseren Vergleich sind zusätzlich `ABML x` (x_1) und `ABML y` (x_2) aufgetragen. Allerdings weist auch `ABML y` mit $\rho = 0.30$ eine starke Korrelation zu dieser Komponente auf.

Tabelle 3.1 zeigt, wie sich die η-Werte der langsamsten sowie einer repräsentativen frontalen und sagittalen Komponente jeweils mit der Anzahl der Iterationen verändern. Es ist deutlich zu sehen, dass der η-Wert bei allen Komponenten mit erhöhter Anzahl der Iterationen sinkt, aber auch dass bei den frontalen und sagittalen Komponenten die Korrelationen zu den korrespondierenden Beschleunigungssensoren immer weiter sinken. Letzteres lässt sich durch die zunehmende Glättung der Komponenten erklären. Zudem finden sich die frontalen und sagittalen Komponenten mit jeder Iteration in einer späteren Komponente, da die vorderen Komponenten allesamt zu Variationen der ersten Komponenten mit stärkerer Oszillation beim Umfallen werden.

Komponente für Umfalldetektion

Leider scheint sich allerdings keine der bisher genannten Komponenten dazu zu eignen, eine tatsächliche Umfallprävention zu unterstützen. Die langsamen Komponenten verändern sich nämlich erst, sobald der Roboter schon dabei ist umzufallen, die vorgestellten Pendelkomponenten extrahieren zu regelmäßig die oszillierenden Komponenten. Jedoch wurde spätestens ab der zweiten Iteration eine Komponente sichtbar, welche einen deutlichen Peak hat, kurz bevor der Roboter fällt und bevor die langsamste Komponente ansteigt.

#SFA-Runden	1	2	3	4	5	6
$\eta(y_1)$	7,2	3,08	2,39	2,08	1,88	1,75
η(frontal)	30,88 ($y_2, \rho = 0,97$)	25,59 ($y_3, \rho = 0,96$)	23,94 ($y_4, \rho = 0,95$)	23,06 ($y_4, \rho = 0,94$)	22,85 ($y_5, \rho = 0,94$)	22,33 ($y_5, \rho = 0,94$)
η(sagittal)	71,57 ($y_5, \rho = 0,53$)	51,06 ($y_7, \rho = 0,48$)	41,0 ($y_6, \rho = 0,51$)	36,43 ($y_7, \rho = 0,46$)	29,54 ($y_6, \rho = 0,43$)	33,36 ($y_9, \rho = 0,37$)

Tabelle 3.1: η-Werte bei Variation der Anzahl der SFA^2-Iterationen. Es ist jeweils die langsamste Komponente, eine frontale und eine sagittale Komponente angegeben. Bei den frontalen und sagittalen Komponenten ist in Klammern angeben, in welcher Ausgabekomponente der Runde sie auftrat und welche Korrelation sie auf dem Ausschnitt $t = 1 \ldots 1200$ zu `ABML x` bzw. `ABML y` aufweist. Als repräsentierte frontale und sagittale Komponenten wurden jeweils diejenigen gewählt, welche am stärksten zum `ABML`-Sensor der gleichen Richtung korreliert sind, aber gleichzeitig nicht zu sehr zum `ABML`-Sensor der anderen Richtung korreliert sind; sonst wäre nämlich nach der sechsten Iteration y_2 mit $\rho(y_2, \texttt{ABML x}) = 0,54$ die repräsentierte sagittale Komponente, während sie aber tatsächlich fast identisch zur langsamsten Komponente (Roboterposition) ist und mit $\rho(y_2, \texttt{ABML y}) = 0,47$ auch sehr stark mit dem sagittalen Sensor korreliert ist. Zum Vergleich noch die η-Werte der normalisierten Sensorendaten: $\eta(\texttt{ABML x}) = \eta(x_1) = 83,38$, $\eta(\texttt{ABML y}) = \eta(x_2) = 73,02$.

Abbildung 3.11 zeigt diese Komponente, welche rein optisch vom Kurvenverlauf her am meisten der sagittalen Komponente ähnelt, aber tatsächlich stärker frontal korreliert ist ($\rho(y_8, \texttt{ABML y}) = -0,44$, $\rho(y_8, \texttt{ABML x}) = 0,22$). Außerdem erkennt diese Komponente offensichtlich den Start der Laufbewegung bei $t = 300$. Die Komponente trat in dieser klaren Form erst ab der zweiten Iteration auf, deutete sich aber schon nach der ersten Iteration in der siebtlangsamsten Komponente an.

Um herauszufinden, welche Sensoren am stärksten in diese vermeintliche Warnkomponente eingehen, wurden die gelernten Koeffizienten der entsprechenden Komponente aus der zweiten Iteration $y_{8(2)}$[3] betrachtet. Wie im vorigen Absatz erwähnt, weist diese Komponente bereits nach der zweiten Iterationen den Warnpeak auf. Exemplarisch wurden die zehn SFA-Komponenten aus der ersten Iteration $y_{l(1)}$, die mit den höchsten Gewichten in $y_{8(2)}$ eingehen, dahingehend analysiert, welche Monome aus dem expandierten normalisierten Signal wiederum jeweils in sie eingehen. Darunter wurden exemplarisch auch nur jeweils die zehn am stärksten in die Komponenten der ersten Iterationen eingehenden Monome betrachtet, welche vom Gesamtgewicht her im Schnitt 20% an der jeweiligen $y_{l(1)}$-Komponente ausmachten. Zunächst war zu erkennen, dass bis auf eine Ausnahme unter den jeweils zehn am stärksten betrachteten Komponenten nur quadratische Terme in $y_{l(1)}$ eingingen. Viel interessanter war aber, dass in neun von zehn analysierte Komponenten aus $y_{l(1)}$ am stärksten solche Monome eingingen, die Sensorwerte vom Accelboard `ABFR` vom rechten Fuß verwendeten. Dies spricht wie davor vermutet dafür, dass ein Hängenbleiben des rechten Fußes am Boden für das Umfallen verantwortlich ist.

Auch bei der Aida-Sequenz konnten wie in Abbildung 3.12 zu sehen eine bzw. zwei

[3] $y_{8(2)}$ = achtlangsamste Komponente nach zweiter Iteration.

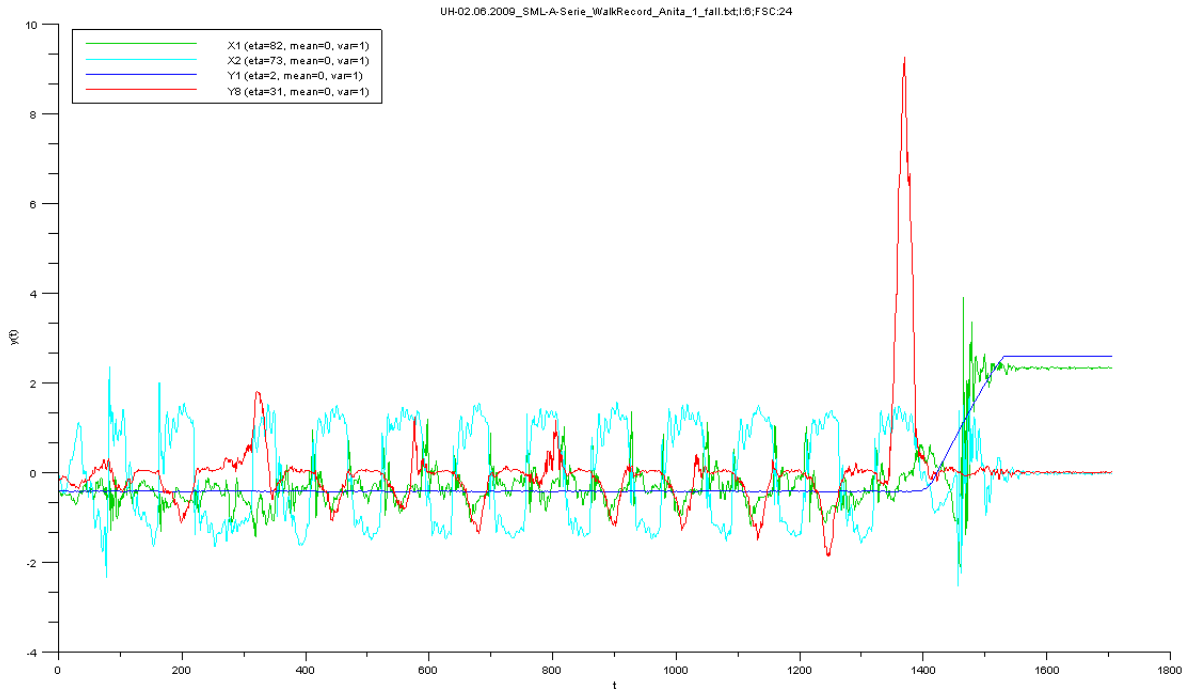

Abbildung 3.11: Mögliche Warnkomponente in der Anita-Lauf-Fall-Sequenz.

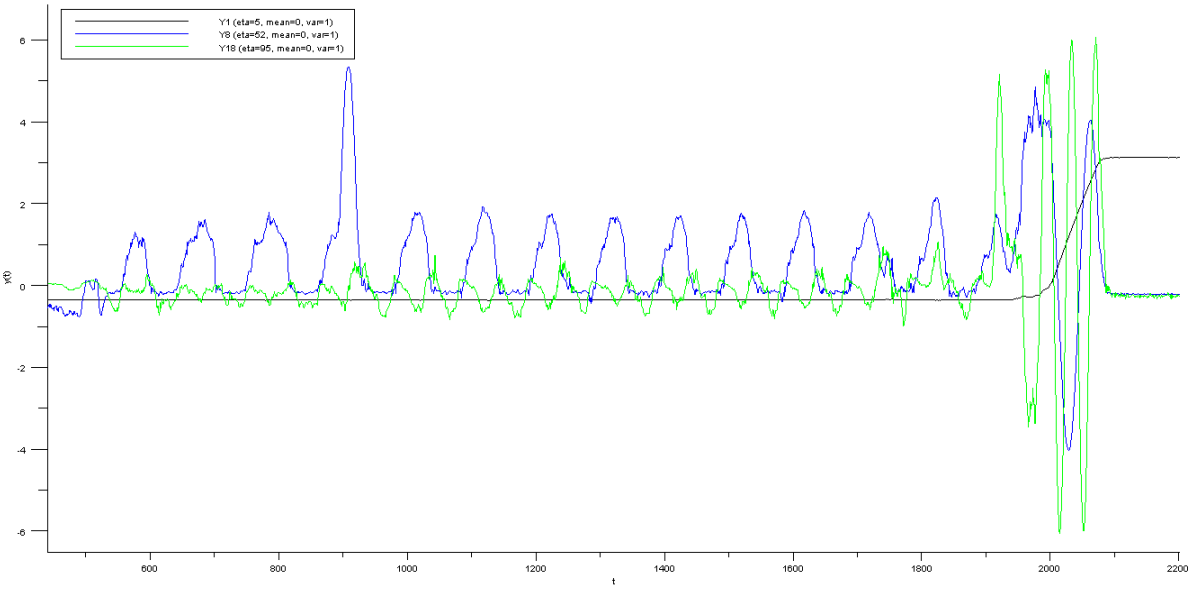

Abbildung 3.12: Mögliche Warnkomponente(n) in der Aida-Lauf-Fall-Sequenz.

mögliche Warnkomponenten gefunden werden. y_8 erkennt den Start der Laufbewegung bei $t = 900$, weist allerdings keinen Warnpeak auf, während erst bei y_{18} ein deutlicher Peak an $t = 1920$ zu sehen ist. Das erste mal ist ein auffälliger Peak an $t = 1920$ nach drei Iterationen in Komponente $y_{13(3)}$ aufgetreten.

Es muss sich bei der später folgenden generalisierten Betrachtung zeigen, ob diese Komponenten zuverlässig extrahiert werden können und auch auf noch nicht gesehenen Eingabedaten als Indikator dienen. Ebensowenig dürfen sie einen zu hohen Fehler zweiter Art aufweisen.

3.4.3 Simples wiederholtes Lernen auf beschränkten Daten

Wie im letzten Abschnitt wird eine einfache wiederholte SFA^2 verwendet, diesmal jedoch immer nur auf einem Teil der Beschleunigungsdaten. Zur Sensorgruppe **X** werden die acht sagittalen Sensoren `ABML x`, `ABSR x` (Schulter), `ABAL x`, `ABAR x` (Arme), `ABHL x`, `ABHR x` (Hüfte), `ABFL x` und `ABFR x` (Füße) zusammengefasst. Die frontale Gruppe **Y** besteht aus `ABML y`, `ABSR y` (Schulter), `ABFL y` und `ABFR y` (Füße); die Gruppe **Z** besteht aus den transversalen, d. h. senkrecht nach oben bzw. unten zeigenden Sensoren `ABAL y`, `ABAR y` (Arme), `ABHL y`, `ABHR y` (Hüfte). Jede Gruppe wird gesondert als Eingabe für die SFA verwendet und analysiert, zudem werden in einem letzten Versuch **Y** und **Z** zur Gruppe **YZ** zusammengefasst.

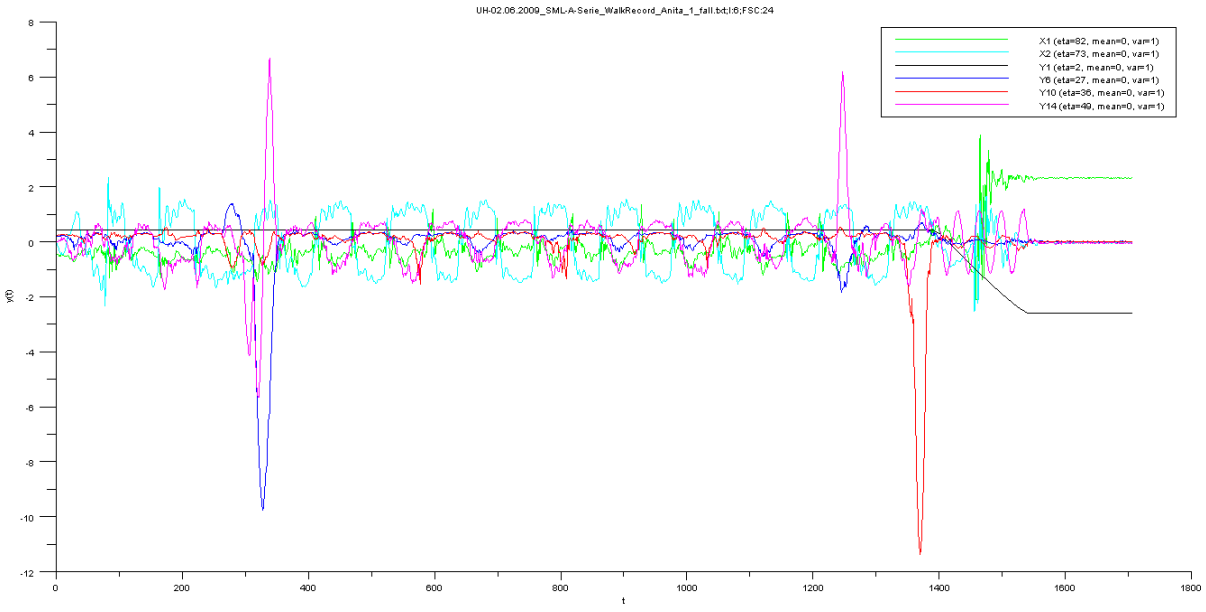

Abbildung 3.13: Einige interessante durch die SFA auf Sensorgruppe **X** gefundene Komponenten in der April-Lauf-Fall-Sequenz.

Von besonderem Interesse waren natürlich die Ergebnisse der SFA auf der Sensorgruppe **X** mit der Frage, ob sich ohne den direkten Einfluss der frontalen Sensoren die sagittale Pendelbewegung besser extrahieren ließ. Abbildung 3.13 zeigt einige auf dieser Sensorgruppe gefundene Komponenten nach sechs SFA^2-Iterationen.

Zunächst lässt sich feststellen, dass die langsamste positionsbeschreibende Komponente sehr unproblematisch und ebenso schnell wie auf den gesamten Daten gefunden wurde, die nächstlangsameren Komponenten y_2 bis y_5 zeigten alle starke Ähnlichkeit zur langsamsten Komponente. Betrachtet man die Korrelationskoeffizienten der weiteren Kompo-

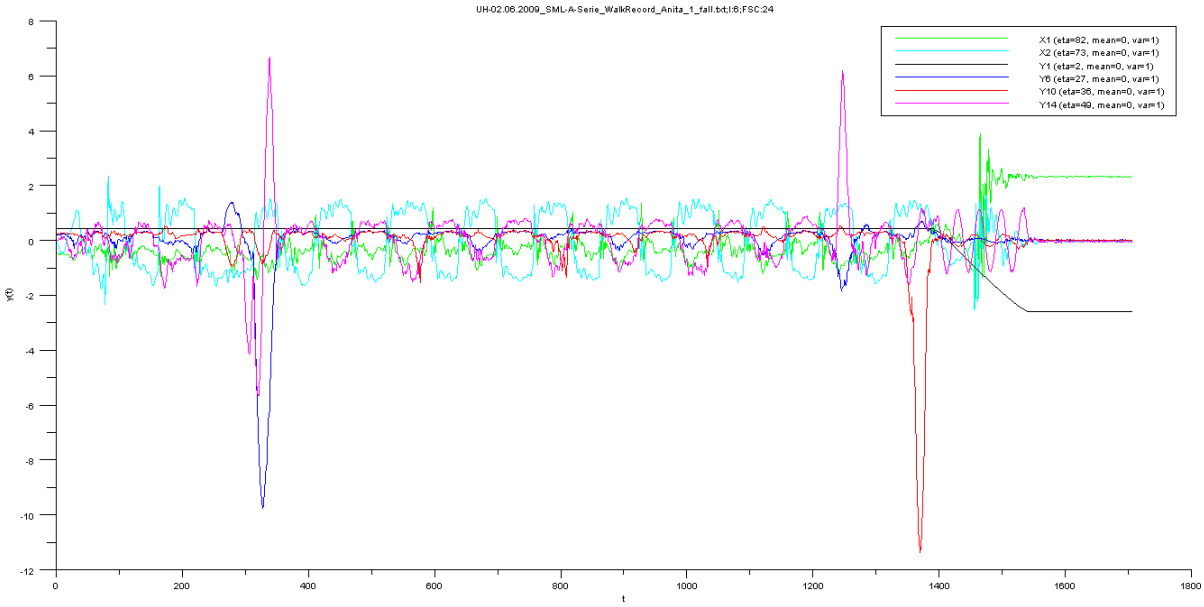

Abbildung 3.14: Vergrößerter Ausschnitt einiger interessanter durch die SFA auf Sensorgruppe **X** gefundener Komponenten in der April-Lauf-Fall-Sequenz.

nenten mit den sagittalen und frontalen Schultersensoren, so ist sichtbar, dass diese im Vergleich zum Lernen auf allen Daten insgesamt niedriger sind. Ein noch gravierenderer Unterschied zum Lernen auf allen Daten besteht darin, dass zudem fast alle Komponenten in etwa gleich starke Korrelation zu den frontalen und sagittalen Sensordaten aufweisen. Trotzdem konnten durch die Betrachtung des Kurvenverlaufs bestimmte Kurven eher der sagittalen oder frontalen Ebene zugeordnet werden. Abbildung 3.14 zeigt einen vergrößerten Ausschnitt einiger Komponenten. y_6 scheint stärker die sagittale Komponente ausdrücken, ist aber auf dem Abschnitt $t = 1 \ldots 1200$ fast gleich stark mit ca. $0,3$ mit `ABML x` (x_1) und `ABML y` (x_2) korreliert.

Es ist interessant, dass y_{14} eine derart starke Ähnlichkeit zum frontalen Schultersensor zeigt, wobei die Korrelation nur bei 0,39 liegt (zum Vergleich Korrelation zu `ABML y`: 0,30). Jedoch ist dies wahrscheinlich vor allem der Tatsache geschuldet, dass bei der Verbauung der Sensoren im Roboter die Sensoren nicht hundertprozentig nur in eine Richtung weisen, wodurch es zu Überschneidungen in den verschiedenen Ebenen kommen kann.

Sehr gut erkennen lässt sich in y_{10} an $t = 1350$ ein deutlicher Peak vor dem Fallen, ebenso wie zwei Peaks um das Starten der Laufbewegung herum bei $t = 280$ und $t = 320$. Es erscheint plausibel, dass diese Komponenten den sagittalen Fußsensoren hohe Gewichte zuordnen und möglicherweise das Hängenbleiben des Fußes darstellen.

Auch in der Aida-Sequenz konnten ähnliche Komponenten und Korrelationen beobachtet werden.

Bei den **Y**- und **Z**-Gruppen wurden erwartungsgemäß schlechtere Ergebnisse gefunden, da hierbei jeweils nur auf vier Sensoren gelernt wurde. Auch hier wurde die langsamste Komponente gefunden, die Korrelationskoeffizienten waren insgesamt niedriger. Allerdings

wurde weder bei der einen noch bei der anderen eine sauber extrahierte Komponente mit sagittalem Pendeln gefunden, frontale Komponenten sind zumindest bei **Y** gut sichtbar. Bei der **Y**-Gruppe gab es dennoch Komponenten, die den Peak beim Start des Laufens aufwiesen, jedoch keinen Warnpeak vor dem eigentlichen Fallen. Noch schlechter Ergebnisse wurden bei der **Z**-Gruppe erzielt, diese hatte erwartungsgemäß auch keine gute frontale Komponente vorzuweisen. Die Ergebnisse auf der zusammengefassten **YZ**-Gruppe ergaben auch keine neuen Ergebnisse, außer dass die sagittale Ebene – bis auf die langsamste Komponente, welche das Umfallen repräsentiert – kaum aus diesen Sensoren extrahiert werden konnte.

Als Fazit lässt sich sagen, dass sich lediglich die Gruppe der rein sagittalen Komponenten als Option für eine Umfalldetektion mit der SFA eignet. Mit dieser können gut ungewöhnliche Bewegungen in sagittaler Ebene erkannt werden, allerdings eignen sie sich nicht so gut dazu, sauber die eigentlichen Pendelbewegungen in frontaler und sagittaler Ebene zu extrahieren.

3.4.4 Hierarchisches Lernen mit Wiederholung

Da die normale wiederholte SFA keine sehr gute Extraktion der Pendelbewegung gestattete, diese aber in der im letzten Abschnitt vorgestellten getrennten SFA-Durchführung teilweise besser zu sehen waren, liegt die Idee nahe, zunächst getrennt auf den verschiedenen Ebenen zu lernen und die Komponenten dann in einer oder mehreren SFA-Runden zusammenzufassen.

Abbildung 3.15 zeigt die verwendete SFA-Hierarchie. Die acht Sensoren in sagittaler Richtung (**X**-Gruppe) werden von den in frontaler und transversaler Richtung (**YZ**-Gruppe) getrennt in jeweils eine SFA^1 geleitet. Die lineare SFA dient der Vorfilterung und Zusammenfassung, während die folgende SFA^2-Schicht der Extraktion nicht-linearer Merkmale dient. In der dritten Schicht werden die Ausgangssignale in einer linearen SFA^1 zusammengefasst und schließlich von einer oder mehreren SFA^2-Schichten weiterverarbeitet.

Bei der hierarchischen SFA-Lernstruktur können verschiedene Parameter angepasst werden. Auch hier wurde die Anzahl der finalen SFA^2-Iterationen sowie die Anzahl der weitergereichten Komponenten variiert. Dabei zeigten sich ähnliche Ergebnisse wie bei der simplen SFA, nur dass bei bei mehr als 28 Komponenten sowie bei der Erhöhung der Iterationen auf 8 und mehr eine Degeneration stattfand, d. h. dass mehrere wenig aussagekräftige Signale mit fast konstanter Ausgabe statt der bisher bekannten Positionsinvariante unter den langsamsten Komponenten befanden. Somit machte es bei dieser Lernstruktur keinen Sinn, mit mehr als 8 Iterationen und mehr als 24 weitergereichten Komponenten zu lernen.

Insgesamt ergaben sich leider weder neue noch bessere Komponenten als mit der einfachen SFA. Abbildung 3.16 zeigt die am stärksten frontal korrelierten Komponenten y_5 sowie y_8 (0,89 und -0,7 auf $t = 1 \ldots 1200$), wobei bei y_8 zudem gut der Peak vor dem Umfallen zu sehen ist. Die stärkste Ähnlichkeit zur sagittalen Ebene wies mit einer Kor-

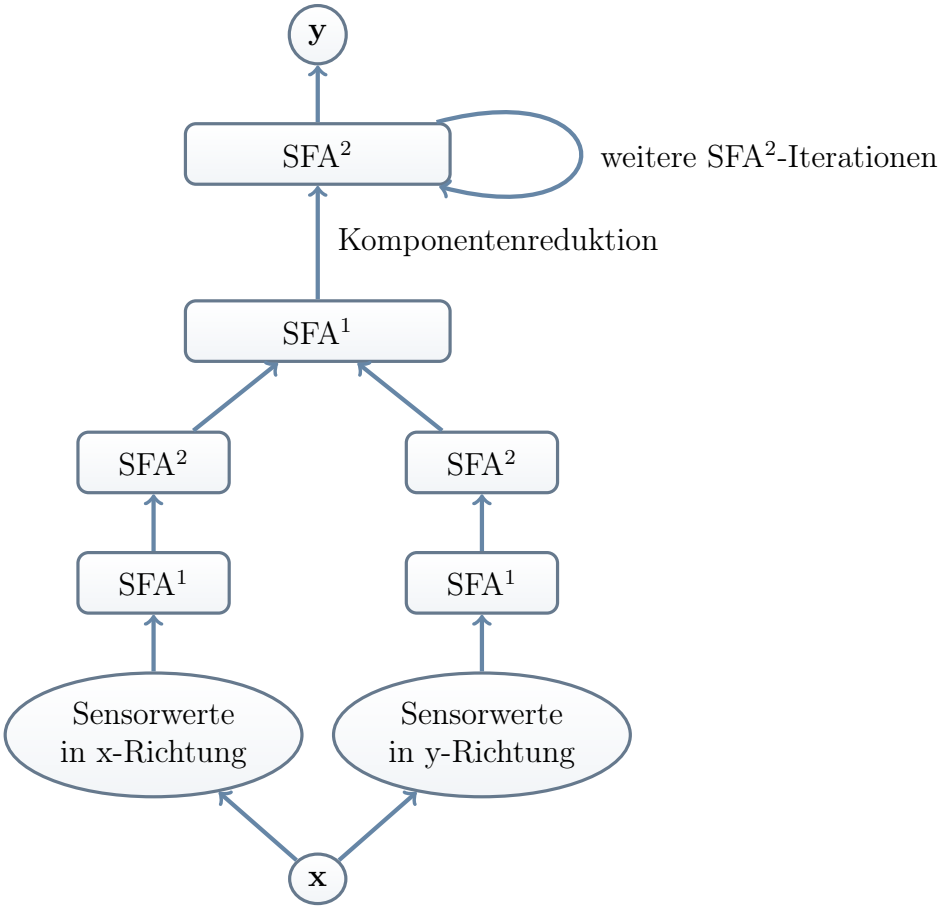

Abbildung 3.15: Hierarchische SFA

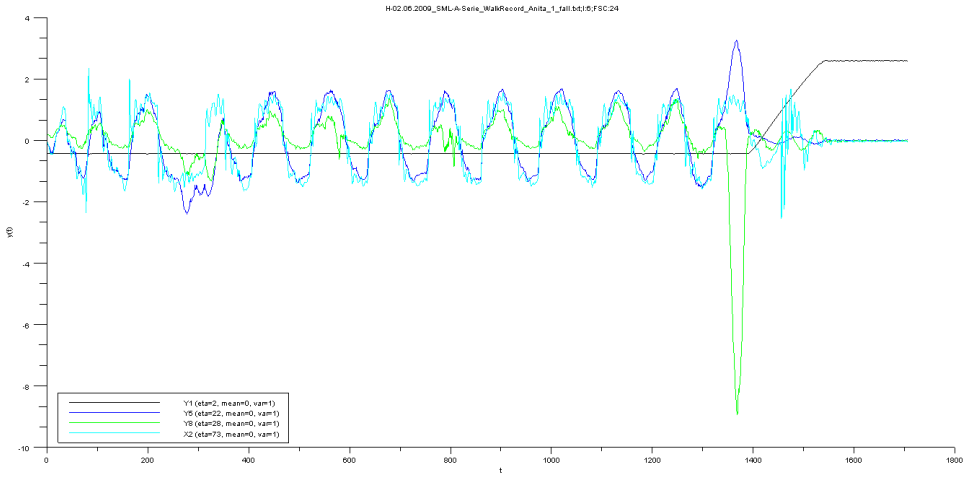

Abbildung 3.16: Hierarchisches Lernen: SFA-Komponenten, welche mit der frontalen Schwingung der Laufbewegung korrelieren sowie der entsprechende `ABML y`-Beschleunigungssensor (x_2). y_8 zeigt einen deutlichen Peak, bevor sich y_1 verändert.

relation von 0,56 y_9 auf, hatte aber optisch keinen höheren Abstraktionsgrad als die bei der einfachen SFA gefundenen Komponenten. Allerdings wies auch sie vor dem Umfallen einen stärkeren Ausschlag an $t = 1380$ auf.

3.4.5 Fazit

Zusammenfassend lässt sich sagen, dass mit der einfachen wiederholten SFA^2 auf der Gesamtheit der Sensoren sehr gute Ergebnisse erzielt wurden. Im Prinzip wurden alle Komponenten, die in Abschnitt 3.2 für möglich gehalten wurden, gefunden. Sogar eine wahrscheinlich mit dem Hängenbleiben des Fußes zusammenhängende Komponente, die kurz vor dem Umfallen einen deutlichen Peak zeigt, wurde gefunden. Mit genauerer Auswahl der Komponenten und der Anzahl der Iterationen ist eventuell noch eine deutlichere Extraktion der sagittalen Ebene und eine zuverlässigere Extraktion des Warnsignals möglich.

Noch ist allerdings nicht klar, ob die gefundenen Komponenten tatsächlich dafür genutzt werden können, um dem drohenden Fallen eines Roboters rechtzeitig entgegenzusteuern. Bei den folgenden Betrachtungen muss evaluiert werden, ob die Komponenten, welche vermeintliche Warnpeaks kurz vor dem Sturz aufwiesen, auch auf unbekannte Daten und andere Roboterplattformen generalisierbar sind.

Auch auf der **X**-Gruppe konnten gute Komponenten gefunden werden. Dies lässt hoffen, dass der Rechenaufwand zusätzlich minimiert werden kann, indem nur ausgewählte Sensoren für die SFA-Berechnungen benutzt werden müssen, um gute Ergebnisse zu erhalten. Allerdings ist die Optimierung der Berechnung hinsichtlich Effizienz nicht Gegenstand dieser Arbeit.

Mit der vorgestellten hierarchischen Struktur konnten auf den Beschleunigungssensordaten keine besseren Ergebnisse als mit der einfachen wiederholten SFA erzielt werden.

3.5 Generalisierung

Wurden mit der SFA bisher gute Ergebnisse erzielt, wenn auf dem gleichen Datensatz gelernt und getestet wird, stellt sich die Frage, wie und ob die gelernten Parameter auch robust auf unbekannten Testdaten sind. In diesem kurzen Abschnitt werden dazu ein paar Versuche und Ergebnisse vorgestellt. Alle Versuche wurden mit der simplen wiederholten SFA durchgeführt.

Neben den bisherigen Parametern, nämlich der Anzahl der SFA-Runden und der Anzahl der pro Stufe beibehaltenen Komponenten, lassen sich jetzt zudem die Trainingsdaten variieren. Das Hauptaugenmerk bei der Auswahl der Trainingsdaten liegt auf der Umfalldetektion, die Zusammenstellung der Trainingsdaten erfolgt daher nach folgenden Kriterien: Zum einen, in welcher Kombination Abschnitte, in denen der Roboter steht, läuft und fällt, ausgewählt werden; zum anderen, ob durch die Länge der einzelnen Abschnitte eine implizite Gewichtung dieser stattfindet. Zuletzt kann variiert werden, ob nur Daten eines einzelnen oder mehrerer Roboter verwendet werden.

Der erste Abschnitt wird sich mit der Zusammenstellung von Trainingsdaten sowie der Wahl der Anzahl der SFA-Runden und der Anzahl der pro Stufe beibehaltenen Komponenten befassen. Danach werden diese Parameter angewendet, um die bisher bekannten Komponenten auf unbekannten Testdaten zu finden.

Trainingsdaten

In den ersten Versuchen wurden zunächst Testdaten aus Sequenzen mit der Aida-Plattform gearbeitet. An Daten standen insgesamt sechs Abschnitte zur Verfügung: Jeweils zwei, in denen der Roboter nur läuft, in welchen der Roboter nach hinten umfällt sowie in welchen er nach vorne umfällt. Als letzte Zusammenstellung wurden Fallabschnitte dreier verschiedener Roboter kombiniert. Im Detail wurden folgende Trainingsdaten verwendet:

D_1 Gleichgewichtung von Lauf-, Liege- und Fallelementen, kombiniert aus zwei Sequenzen vom gleichen Roboter (Aida). Von jeder Sequenz wurden 40 Sekunden Laufen und 20 Sekunden Liegen verwendet, außerdem aus der ersten Sequenz das Umfallen nach vorne, aus der zweiten das Umfallen nach hinten.

D_2 Fokus auf Umfallausschnitten des gleichen Roboters (Aida). Aus drei Sequenzen wurden die Umfallausschnitte extrahiert und kombiniert, erstens ein normaler Rückwärtsfall, zweitens ein Ausschnitt, in welchem der Roboter stark nach hinten wankt und dann nach vorne fällt, drittens ein langsamerer Rückwärtsfall.

D_3 Kombination von verschiedenen Robotern: Jeweils eine Vorwärts- und Rückwärtsfall-Sequenz von Aida, Vorwärtsfall von April, Rückwärtsfall von Aimee. Alle Sequenzen beinhalten wenige Sekunden Laufen und Liegen.

Getestet wurde neben den bereits aus dem letzten Abschnitt bekannten Abschnitten von Anita und Aida noch auf folgenden Sequenzen: Zum einen eine Sequenz von Roboter Aimee, in welcher der Roboter kurz steht, dann bei $t = 500$ frontal zu schwingen und $t = 900$ zu laufen beginnt, bis der Roboter bei $t = 3000$ langsam nach hinten umkippt. Zum anderen noch eine Sequenz von Roboter Aida, in welcher der Roboter beim Laufen an $t = 1900$ einmal fast nach vorne kippt, das Gleichgewicht aber wieder erlangt und schließlich das Laufen neu initiiert, bis er dann an $t = 4000$ nach hinten umfällt.

Es wurden als Testdaten ausschließlich Sequenzen verwendet, welche weder ganz noch ausschnittsweise in den Trainingsdaten enthalten sind.

Overfitting

Die ersten Versuche mit den Standardeinstellungen aus dem letzten Abschnitt, d. h. 5 Iterationen und 24 Komponenten pro Stufe, ergaben auf nicht erlernten Testdaten eher schlechte Ergebnisse: Nach der fünften Iterationen wiesen alle der ersten 24 Komponenten starke Jitter auf, ebenso wurden die sagittalen und frontalen Komponenten sehr undeutlich. Die Verringerung der Iterationen brachte etwas bessere Ergebnisse, allerdings wurden die Komponenten umso gleichmäßiger, je weniger Komponenten von einer SFA-Stufe zur nächsten weitergegeben wurden. Wie im Abschnitt 3.4 bemerkt, haben in höheren Stufen vor allem die schnelleren Komponenten hohe Gewichte. Offensichtlich hat die Verwendung weniger langsamer Komponenten ein Overfitting auf den Trainingsdaten zur Folge, in den späteren Komponenten sind also sehr roboter- und sequenzspezifische Informationen zu finden.

Diesem Problem lässt sich begegnen, indem weniger Komponenten pro Iteration weitergereicht werden. Dies führt zwar insgesamt zu einer relativen Verschlechterung, da die Daten insgesamt verrauschter sind, doch kommen zumindest die meisten bekannten Komponenten zum Vorschein. Als Standardeinstellung wurden daher 5 Iterationen und 12 weitergereichte Komponenten verwendet. Zwar sind beim Umfallen hohe Jitter zu sehen, aber während der Lauf-, Steh- und Liegephasen sind die Komponenten auch auf ungelernten Daten stabil. Abbildung 3.17 zeigt die langsamste Komponente in Abhängigkeit von den Trainingsdaten.

Ergebnisse

Zunächst wurden einige Sequenzen auf D_1 getestet. Zuverlässig ließ sich auf allen getesteten Sequenzen y_1 als Anzeige für das Vorwärtsfallen identifizieren, allerdings zeigte diese Komponente keine Reaktion bei Sequenzen, in der der Roboter nach hinten fällt. Das Rückwärtsfallen wurde allerdings immer durch y_2 angezeigt, auch wenn diese noch stärkere frontale Schwingungsanteile beim Laufen aufzeigte. Desweiteren zeigte y_5 eine hohe Korrelation zum sagittalen Schultersensor, die stärkste frontale Korrelation wies y_7 auf. Allerdings wurden auf diesen Trainingsdaten keinerlei Komponenten mit Warnpeaks gefunden werden, die auf ein Hängenbleiben eines Fußes auf dem Boden hindeuten könnten.

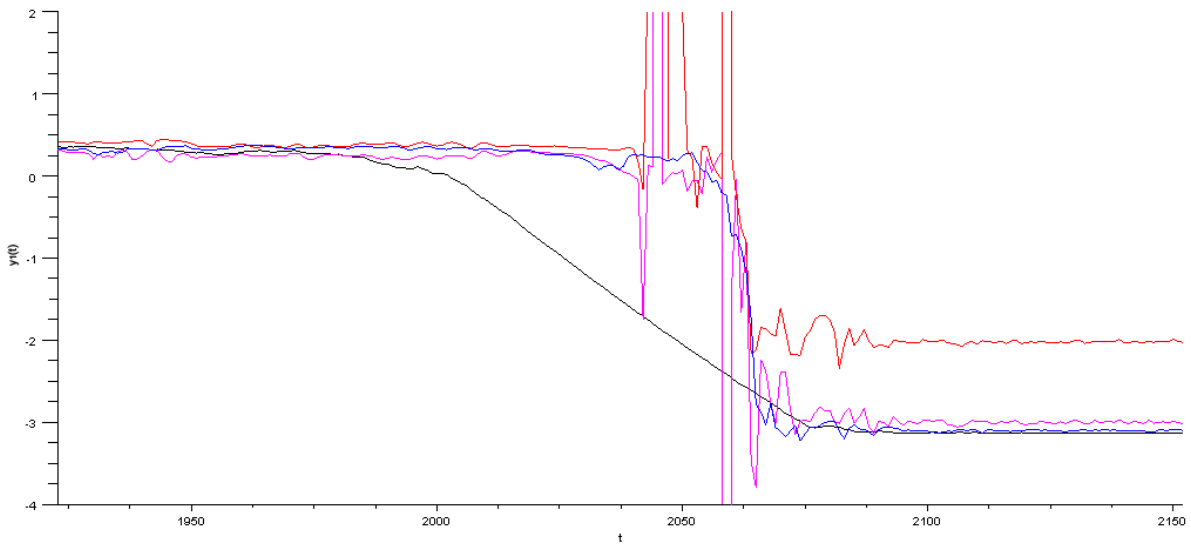

Abbildung 3.17: Langsamste Komponente der Aida-Vorwärtsfall-Sequenz aus verschiedenen Parameterkonfigurationen. Schwarz: Gelernt auf Testdaten, 5 Iterationen, 24 Komponenten pro Stufe. Blau: Gelernt auf Testdaten, 5 Iterationen, 12 Komponenten pro Stufe. Lila: Gelernt auf D_1, 5 Iterationen, 12 Komponenten pro Stufe. Rot: Gelernt auf D_2, 5 Iterationen, 12 Komponenten pro Stufe.

Auf D_2 konnten alle Komponenten wie bei D_1 gefunden werden, jedoch konnten zusätzlich auch zwei Komponenten gefunden werden, welche Warnpeaks aufwiesen. Abbildung 3.18 zeigt, dass in der Anita-Sequenz Warnkomponenten gefunden wurden, durch einen Parametersatz, der auf Aida-Sequenzen gelernt wurde. Auch auf verschiedenen anderen Komponenten konnten ähnliche Warnpeaks gefunden werden, auch wenn sie nicht immer so deutlich sichtbar waren. Dies könnte auch damit zusammenhängen, dass nicht jedes Umfallen dem Hängenbleiben mit einem Fuß geschuldet ist.

Die Tests auf den gemischten Trainingsdaten D_3 brachten keine weiteren Erkenntnisse, sondern waren eher etwas schlechter als zuvor. Zwar wurden die gleichen Komponenten für Vorwärtsfall y_1 und Rückwärtsfall y_2 gefunden, allerdings waren beide Komponenten unruhiger und wiesen stärkere frontale Schwingungsanteile auf. Desweiteren wurden viele frontale und einige sagittale Komponenten gefunden, allerdings keine Warnkomponente, die mit einem Hängenbleiben des Fußes korrelieren würde.

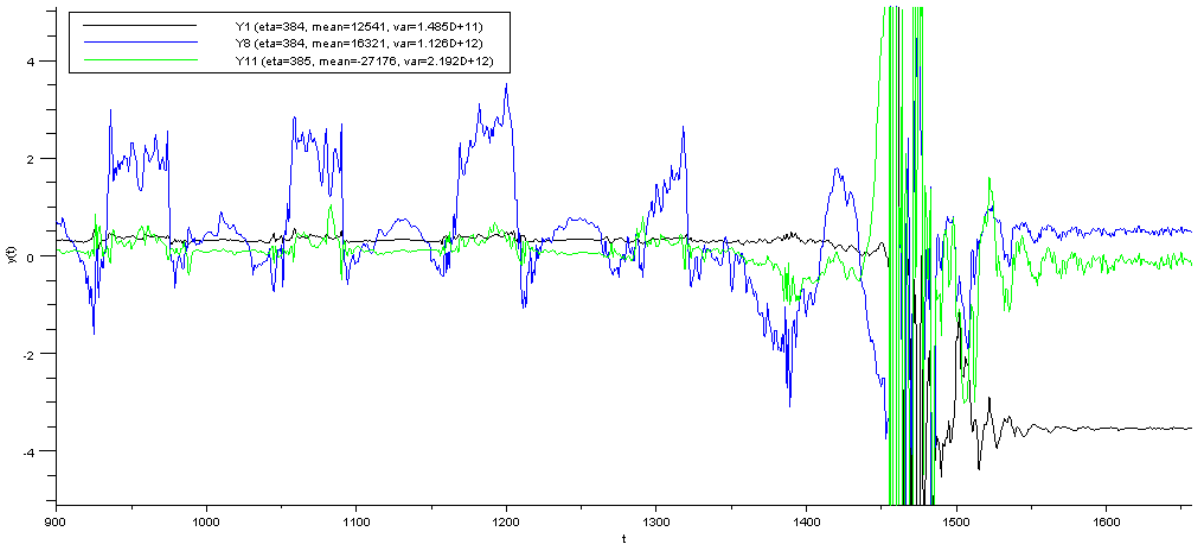

Abbildung 3.18: Potentielle Warnkomponenten bei der Anita-Sequenz, gelernt auf Aida-Sequenzen (D_2). Vor allem y_8 zeigt einen deutlichen negativen Ausschlag an $t = 1390$, bevor der Roboter tatsächlich umfällt.

Kapitel 4

Zusammenfassung und Ausblick

In dieser Arbeit wurde die Slow Feature Analysis vorgestellt und verwendet, um das Laufverhalten humanoider Roboter zu untersuchen.

Die Analyse hat gezeigt, dass die SFA ein robustes Verfahren ist, welches als Offline-Verfahren auf aufgenommenen Roboterbewegungssequenzen langsamste Komponenten extrahieren kann, welche die Position des Roboters, frontale und laterale Schwingungen sowie ungewöhnliche Zustandsveränderungen während des Laufens abbilden. Vor allem letzteres könnte benutzt werden, um dem Roboter als Umfalldetektion dienen, damit er rechtzeitig gegensteuernde motorische Bewegungen vollführen kann. Es wurde gezeigt, dass bereits die auf wenigen Trainingsdaten gelernten Parameter auf unbekannten Testdaten akzeptable Resultate liefern, sowohl auf der identischen Roboterplattform, wie auch Robotern gleicher Bauart. Diese Tatsache hat den positiven Effekt, dass nach einiger Zeit an den Roboterteilen auftretende der Verschleißerscheinungen sich nicht negativ auf die Robustheit der gelernten Parameter auswirken.

In dieser Arbeit wurde allerdings nur die theoretische Möglichkeit aufgezeigt, eine Umfalldetektion für humanoide Roboter mit Hilfe der SFA zu implementieren. Zunächst muss eine möglichst optimale Zusammenstellung der Trainingsdaten sowie Anzahl der Iterationen und benötigten Komponenten gefunden werden. Es ist dabei wie bei jedem Lernverfahren wichtig, eine möglichst gute Abdeckung der Anwendungsfälle zu erzielen, ohne die Generalisierung zu gefährden. Weitere Versuche mit einer breiteren Masse oder einer gezielteren anwendungsspezifischen Auswahl an Trainingsdaten wären denkbar.

Desweiteren muss der Ausführungsschritt der SFA effizient für die jeweilige Zielplattform implementiert werden. Während der Lernschritt, der jedoch auf einer geeigneten Trainingsdatenmenge offline durchgeführt werden kann, relativ kompliziert ist, handelt es sich bei der SFA-Ausführung lediglich um eine bis mehrere Matrixadditionen und Matrixmultiplikationen. Für diese Operationen müsste auf der A-Serie eine geeignete neuronale Implementation gefunden werden, damit die SFA-Komponenten in Echtzeit berechnet werden können.

Zur effizienten Berechnung sind weitere Optimierungen denkbar. Zum einen sollte weiter erforscht werden, inwiefern redundante Sensoren weggelassen werden können, um die Dimension des Eingabesignals zu verkleinern. Im Abschnitt 3.4.3 wurden dahingehend bereits einige Versuche unternommen.

Weiterhin kann dadurch optimiert werden, dass die gefundenen Koeffizienten für die gewünschten Komponenten genauer analysiert werden, so dass Eingabekomponenten mit sehr geringen Koeffizienten sukzessive weggelassen werden, solange sich das Ergebnis nicht verschlechtert.

Zuletzt muss die Möglichkeit in Betracht gezogen werden, dass die SFA unter Umständen nicht auf allen humanoiden Roboterplattformen (z. B. solchen mit einer nicht ausreichenden Anzahl von Sensoren) und allen Laufmustern gewünschte Komponenten extrahiert. Jedoch sehen die in dieser Arbeit vorgestellten Ergebnisse relativ vielversprechend aus, dass bei einem ausreichend mit Sensoren bestückten Roboter mit der SFA gute Ergebnisse erzielt werden können.

Anhang A: Implementation der SFA in Scilab

Die hier beschriebene Implementation der SFA orientiert sich stark an der Matlab-Implementation von Pietro Berkes, beschrieben und herunterladbar unter [Ber03]. Neben der Implementation der eigentlichen SFA-Algorithmen gibt es noch einige Skripte zur vereinfachten Auswertung von Roboter-Sensordaten.

Die Komponenten der Scilab-Implementation sind:

1. `/sfa`: SFA-Algorithmus
2. `/sfa/cov`: Module zur Berechnung von Kovarianzmatrizen, Algorithmen für die Hauptkomponentenanalyse (PCA) und Whitening
3. `/sfa/helpers`: Skripte zur Normalisierung, Berechnung von Korrelationen u. .ä.
4. `/sfa/test_demo`: Demo- und Testskripte, `demo_sfa_figure2.sce` wird in Abschnitt2.6 vorgestellt

Um die SFA auf Eingabedaten zu verwenden, müssen folgende Schritte ausgeführt werden:

1. Als Arbeitsverzeichnis in Scilab muss das Wurzelverzeichnis des Pakets ausgewählt sein.
2. SFA-Bibliothek wird importiert durch
   `chdir(‘‘sfa‘‘), exec(‘‘sfa_include.sce‘‘, -1), chdir(‘‘..‘‘)`
3. SFA^2 lernen und ausführen
   `y = sfa2(x)`
   Wenn das SFA-Objekt später weiter nutzbar sein soll, kann außer der Ergebnismatrix zusätzlich das Handle zurückgegeben werden:
   `[y, hdl] = sfa2(x)`
   wobei `x` die Eingabematrix ist, wobei jede Spalte einem Signal und jede Zeile einem Zeitschritt entspricht.

   Die gelernten Parameter können von der globalen Struktur `SFA_STRUCTS` erhalten werden, beispielsweise mittels

```
global SFA_STRUCTS
learned_weights = SFA_STRUCTS(hdl).SF
```

Die Anwendung von `sfa1(x)` erfolgt analog.

Im Wurzelverzeichnis befinden sich einige Skripte zur Berechnung und Ausführung der Roboterbeschleunigungsdaten, besonders interessant dabei ist `main.sce`, welche die Lern- und Ausführungsschemata aus Abschnitt 3.4 bereitstellt. Im oberen Teil des Skriptes befinden sich Konfigurationsmöglichkeiten zur Ausführung und Ausgabe.

Unter `http://informatik.hu-berlin.de/~hoefer/sfa/` kann der kommentierte Quellcode heruntergeladen werden.

Anhang B: Technische Daten der A-Serie

Die als A-Serie bezeichnete humanoide Roboterplattform des NRL wurde auf der Grundlage des Roboterbaukastens Bioloid von der Firma Robotis gebaut. Ein Roboter der Serie ist in Abbildung 3.1 dargestellt, für eine schematische grafische Darstellung siehe Abbildung 3.2.

- Höhe: 47 cm
- Gewicht: 2,2 kg
- AccelBoards: Über den Roboter verteilt befinden sich 8 Platinen mit jeweils einem RISC-Microcontroller und einem 2-achsigem Beschleunigungssensor. Jedes AccelBoard steuer 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.
- In den Roboter ist eine PDA integriert, welche einen Prozessor des Typs PXA272 von Intel enthält. Der PDA ist für die Verarbeitung der Kamerabilder und höhere kognitive Fähigkeiten zuständig.

Abbildungsverzeichnis

Literaturverzeichnis

[Ber03] Pietro Berkes. sfa-tk: Slow Feature Analysis Toolkit for Matlab (v.1.0.1)., 2003. 40

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 30(3):273–297, 1995. 6

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience Publication, 2000. 9

[LF96] G. Tutz (Hrsg.) L. Fahrmeir, A. Hamerle. *Multivariate statistische Verfahren.* de Gruyter, 1996. 9

[MSM09] 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) (submitted)*. Guilin, China, December 18-22 2009. 1

[Thi07] Christian Thiele. Integrierte Entwicklungsumgebung zur Bewegungssteuerung humanoider Roboter. Studienarbeit, Humboldt-Universität zu Berlin, 2007. 42

[Wer08] Benjamin Werner. Sensomotorische Erzeugung eines Gangmusters für humanoide Roboter. Studienarbeit, Humboldt-Universität zu Berlin, 2008. 16, 19

[Wis03] Laurenz Wiskott. Slow Feature Analysis: A Theoretical Analysis of Optimal Free Responses. *Neural Computation*, 15(9):2147–2177, September 2003. 6, 7

[WS02] Laurenz Wiskott and Terrence Sejnowski. Slow Feature Analysis: Unsupervised Learning of Invariances. *Neural Computation*, 14(4):715–770, 2002. 1, 3, 6, 10, 12, 19