Weiterentwicklung und Anwendung von Sparse-Coding-Verfahren für die Analyse von Armbewegungstrajektorien

Dissertation

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

vorgelegt der
Fakultät für Informatik und Automatisierung
der Technischen Universität Ilmenau

von

Dipl.-Inf. Christian Vollmer

Tag der Einreichung: 14. Januar 2016
Tag der wissenschaftlichen Aussprache: 28. September 2016

Gutachter:
1. Univ.-Prof. Dr.-Ing. Horst-Michael Groß
   (Technische Universität Ilmenau)
2. Univ.-Prof. Dr. Martin Giese
   (Universität Tübingen)
3. Dr. Julian Eggert
   (Honda Research Institute Europe GmbH)

urn:nbn:de:gbv:ilm1-2016000453

Kurzbeschreibung

Eine von vielen Modalitäten zur Vermittlung von Information in der Kommunikation und Interaktion zwischen Mensch und Maschine ist die Gestik. Mit Hilfe dynamischer Gesten können sowohl Begriffe, als auch Emotionen kommuniziert werden. In dieser Arbeit wird der zeitliche Verlauf der Position einer Gliedmaße bei Ausführung der Geste, die sogenannte Bewegungstrajektorie, betrachtet. Damit eine Maschine Gesten wahrnehmen kann, müssen die Trajektorien mittels Sensoren aufgenommen werden und anschließend durch eine entsprechende Verarbeitung der Daten interpretiert werden. Dabei kommt ein mehrstufiger Verarbeitungsprozess zum Einsatz, der auch als Mustererkennungsprozess bezeichnet wird. Ein Schritt in diesem Prozess ist die Merkmalsextraktion, welche das aufgenommene Signal in einer kompakten Form darstellt und so die Anwendung eines Klassifikators zur Kategorisierung der Geste erlaubt.

Diese Arbeit widmet sich einer Untersuchung zur Anwendung von Sparse Coding, einem biologisch motivierten Verfahren zur Datentransformation, als Merkmalsextraktionsschritt in einem Mustererkennungsprozess für Bewegungstrajektorien. Die Motivation für diese Untersuchung ist die Eigenschaft des Sparse Coding, eine Datenmenge durch eine beschränkte Menge repräsentativer, wiederkehrender Merkmale darstellen zu können. Diese Merkmale werden in einer Lernphase aus Trainingsdaten gelernt und in der Kannphase in einem unbekannten Signal detektiert. Dieses Konzept hat gegenüber konventionellen Methoden zur Merkmalsextraktion in zeitlichen Signalen den Vorteil, dass die Merkmale optimal an die Daten angepasst sind und so die charakteristischen Eigenschaften der Trainingsdaten beschreiben.

In dieser Arbeit wird das Verfahren für die Anwendung auf Bewegungstrajektorien optimiert. Es wird untersucht, unter welchen Rahmenbedingungen Sparse Coding für Bewegungstrajektorien anwendbar ist und wie die aufgenommenen Daten vorverarbeitet werden müssen. Des Weiteren werden die Auswirkungen des Verfahrens auf nachgelagerte Verarbeitungsschritte im Mustererkennungsprozess, wie die Klassifikation und die Generierung von Bewegungstrajektorien, betrachtet. Insbesondere für die Klassifikation ergeben sich dabei interessante Vorteile aus der Art und Weise, wie das Sparse Coding die Daten repräsentiert.

Die Leistungsfähigkeit des Verfahrens beim Einsatz in der Gestenerkennung wird in Experimenten anhand eines, im Rahmen dieser Arbeit selbst erstellten, Datensatzes demonstriert. Um die Generalisierbarkeit des Verfahrens auf andere Anwendungsdomänen zu untersuchen, wird es auf Benchmark-Datensätze aus den Bereichen der Activity Recognition und der Handschrifterkennung angewendet. Des Weiteren wird eine echtzeitfähige Implementierung des Verfahrens in einer Demonstrator-Applikation vorgestellt.

In einer abschließenden Diskussion werden die Vor- und Nachteile des Verfahrens beschrieben. Es werden grundlegende Defizite des Verfahrens und Einschränkungen für den praktischen Einsatz betrachtet. Weiterhin wird ein Ausblick auf mögliche aufbauende Arbeiten und Erweiterungen des Verfahrens gegeben, die es erlauben, die festgestellten Probleme und Defizite zu beheben.

Abstract

One of the modalities for the transmission of information during communication and interaction between a human and a machine is gesticulation. By means of dynamic gestures, concepts and emotions can be communicated. In this work, the temporal evolution of the position of one limb during the performance of the gesture, the so called movement trajectory, is analyzed. For a machine to be able to perceive a gesture, those trajectories must be recorded via sensors, and the data must be interpreted by means of a suitable data processing mechanism. This data processing is usually implemented by a pattern recognition pipeline, consisting of multiple processing steps. One of those steps is the feature extraction, the purpose of which is to represent the incoming data in a compact form and thus enable a classifier to categorize the gesture.

In this work, the applicability of Sparse Coding, a biologically motivated approach for data transformation, as a feature extraction step in the pattern recognition pipeline is investigated. The main motivation for this research is the ability of Sparse Coding to represent a dataset with a minimal set of representative and recurring features. Those features are learnt in a learning phase and are detected in an unknown signal in the application phase. Compared to conventional methods for feature extraction in temporal signals, this approach has the advantage that the features are adapted to the domain specific data and can thus capture optimally the characteristics of the training data.

In this work, the a general Sparse Coding approach is adapted for the application to movement trajectories. It is investigated, under which preconditions Sparse Coding is applicable to movement trajectories, and how the data must be pre-processed. Further, the effects of the application of the approach for down-stream processing steps, like classification and generation of movement trajectories are examined. Particularly for classification, there are interesting advantages arising from the way Sparse Coding is representing the data.

The feasibility of the approach for processing movement trajectories is demonstrated in experiments on a gesture dataset that has been recorded as part of this work. To show the generalizability of the approach to other application domains, it is applied to benchmark dataset from the fields of activity recognition and handwriting recognition. Further a real-time capable implementation of the approach in form of a demonstrator application is described.

In a concluding discussion, the advantages and disadvantages of the approach are detailed. Basic shortcomings of the approach and limitations for the practical applicability are described. Further, an outlook to possible extensions and continuing works is depicted, which might eliminate the problems and reported deficits.

Danksagung

Ich möchte mich an dieser Stelle bei all jenen bedanken, die mir auf dem Weg zu dieser Arbeit geholfen haben.

Ich danke meinem Doktorvater Prof. Dr. Horst-Michael Groß, der mich immer unterstützt und mir die Promotion und die Arbeit am Fachgebiet Neuroinformatik und Kognitive Robotik ermöglicht hat.

Des Weiteren möchte ich dem Honda Research Institute Europe in Offenbach danken, welches das Projekt überhaupt erst ermöglicht hat und bei dem ich mich als gelegentlicher Gast sehr wohl gefühlt habe. Besonderer Dank gilt dabei Dr. Julian Eggert, der mir als Betreuer bei Honda immer mit kreativen Ideen und Rat zur Seite stand.

Meinen Kollegen am Fachgebiet Neuroinformatik und Kognitive Robotik danke ich für eine unvergessliche Zeit und für die fachlichen Diskussionen, die zum Gelingen dieser Arbeit beigetragen haben.

Ich möchte meiner Familie für ihre Unterstützung danken. Insbesondere meiner Frau, die mich stets zum Schreiben motiviert hat.

Zu guter Letzt danke ich meinen Korrekturlesern, die sich unermüdlich durch diese Arbeit gekämpft haben.

Christian Vollmer
Erfurt, 14. Januar 2016

Inhaltsverzeichnis

Mathematische Notation

Matrizen und Vektoren

$(\mathbf{M})_{i,j}$ Alternative Schreibweise für $M_{i,j}$

$\mathbf{1}_{M\times N}$ Matrix $\mathbf{M} \in \mathbb{R}^{M\times N}$ mit $\forall(i,j) : M_{i,j} = 1$

$\mathbf{I}_P$ Einheitsmatrix der Größe $P \times P$

$\mathbf{M}$ Matrix

$\mathbf{0}_P$ Vektor mit Nullen der Länge P

$\mathbf{x}$ Zeilenvektor

$\mathbf{x}_i$ Spalten mit Index i

$H^{(:)}_{i,j}$ Vektor aller Elemente (i,j) über die Scheiben eines Tensors dritter Ordnung $\mathcal{H}$ mit Scheiben $\mathbf{H}^{(m)}$ für $m = 1,\ldots,M$, also $\left(H^{(:)}_{k,n}\right)_m = H^{(m)}_{k,n}$

$M_{i,j}$ Skalares Element der Matrix $\mathbf{M}$ in Zeile i und Spalte j

$s[k]$ Zeitdiskretes Signal s

x_i Skalares Element des Vektors $\mathbf{x}$ in Zeile i

$X_{:,j}$ j-te Spalte einer Matrix $\mathbf{X}$, $(X_{:,j})_i = X_{i,j}$

$X_{i,:}$ i-te Zeile einer Matrix $\mathbf{X}$, $(X_{i,:})_j = X_{i,j}$

Normen

$||\mathbf{M}||_0$ l_0-Pseudonorm: $|\{(i,j) : |M_{i,j}| > 0\}|$

$||\mathbf{x}||_0$ l_0-Pseudonorm: $|\{i : |x_i| > 0\}|$

Wahrscheinlichkeitverteilungen, etc.

$\mathcal{N}(\mu,\Sigma)$ Normalverteilung mit Mittelwert μ und Kovarianzmatrix Σ

$x \sim \mathscr{P}$ Ziehen einer Stichprobe x aus einer Wahrscheinlichkeitsverteilung $\mathscr{P}$

Sonstiges

$\mathbf{v} * \mathbf{w}$ Faltung der Vektoren $\mathbf{v}$ und $\mathbf{w}$ mit $u_i = (\mathbf{v} * \mathbf{w})_i = \sum_j v_{i-j} w_j$

$\mathbf{v} \star \mathbf{w}$ Korrelation der Vektoren $\mathbf{v}$ und $\mathbf{w}$ mit $u_i = (\mathbf{v} \star \mathbf{w})_i = \sum_j v_{i+j} w_j$

$\{x_i\}$ Kurzschreibweise für $\{x_i : i = 1, \ldots, I\}$

$I(x)$ Indikatorfunktion mit $I(x) = 1$, falls x wahr und $I(x) = 0$ sonst

Allgemeine Anmerkungen zur Notation

Es wird versucht, die Notation in dieser Arbeit weitestgehend konsistent zu halten. Dennoch ist es manchmal von Vorteil zur Vereinfachung und Erhöhung der Lesbarkeit eine alternative Notation zu verwenden. Insbesondere bei der Notation und Indizierung von Vektoren und Matrizen ist eine Abweichung von der Standardnotation manchmal sinnvoll, um die konkreten Zusammenhänge zu verdeutlichen. Dabei ergeben sich teilweise alternative Notationen für den gleichen Sachverhalt. Diese sollen im Folgenden vorgestellt werden.

Sei $\mathbf{X} \in \mathbb{R}^{N \times M}$ eine Matrix. Einzelne Elemente der Matrix werden mit $X_{i,j}$ bezeichnet. Es wird folgende Indexnotation eingeführt: Die n-te Zeile der Matrix $\mathbf{X}$ wird mit $X_{n,:}$ bezeichnet. Entsprechend wird eine m-te Spalte mit $X_{:,m}$ bezeichnet.

Abhängig vom Kontext kann eine Matrix durch Aneinanderreihung einer Menge von Vektoren $\mathbf{x}_m$ für $m \in \{1, \ldots, M\}$ bestehen. Dabei muss im Kontext explizit geklärt werden, ob diese Vektoren als Spalten- oder Zeilenvektoren der Matrix verwendet werden. In dieser Arbeit wird eine Matrix üblicherweise aus Spaltenvektoren zusammengesetzt. In diesem Fall gilt $\mathbf{x}_m = X_{:,m}$.

Im vorliegenden Werk wird weiterhin von Tensoren Gebrauch gemacht. Tensoren sind eine Generalisierung der Konzepte von Vektoren und Matrizen auf beliebige Dimensionen. Ein Vektor ist ein Tensor erster Ordnung und eine Matrix ist ein Tensor zweiter Ordnung. Ein Tensor dritter Ordnung kann als *Stapel* von Matrizen verstanden werden. Tensoren höherer Ordnung werden in dieser Arbeit nicht benötigt und daher nicht weiter erläutert.

Eine Menge von Matrizen $\{\mathbf{X}_k \in \mathbb{R}^{M \times N}, k = 1, \ldots, K\}$ kann als Tensor dritter Ordnung $\mathcal{X}$ zusammengefasst werden (siehe Abbildung 1). Die k-te *Scheibe* des Tensors wird dann mit $\mathbf{X}^{(k)}$ bezeichnet. Entsprechend der oben eingeführten Notation ist dann $X^{(k)}_{i,j}$ das Element der k-ten Matrix an Position (i,j), $X^{(k)}_{i,:}$ die i-te Zeile und $X^{(k)}_{:,j}$ die j-te der Spalte.

Des Weiteren bezeichnet $X^{(:)}_{i,j}$ den Vektor der Elemente aller Matrizen an Position (i,j). Sehr selten wird in dieser Arbeit auch die Bezeichnung $X^{(:)}_{:,j}$ für die Matrix bestehend aus den j-ten Spalten aller Matrizen verwendet. Dabei wird im Kontext explizit geklärt, welcher der beiden Indizes über Spalten und welcher über Zeilen der entstehenden Matrix läuft.

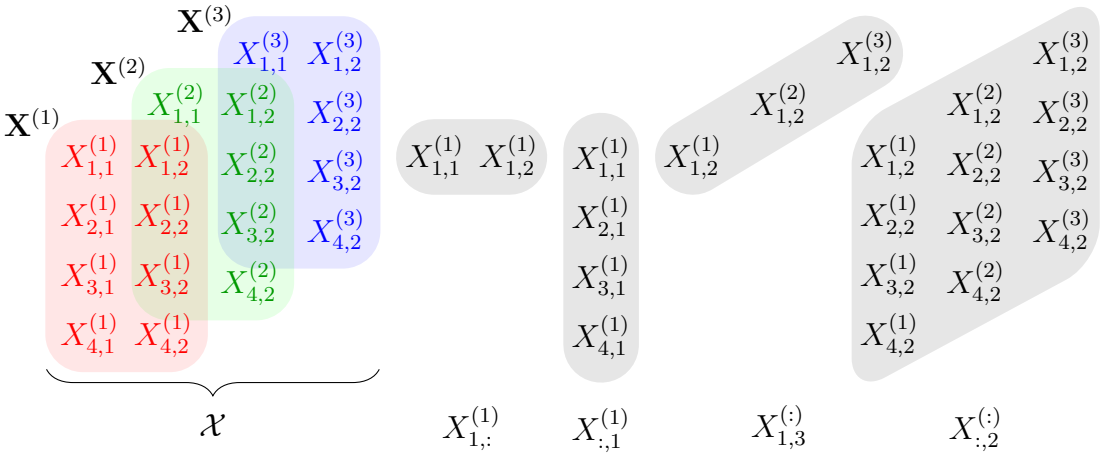


Abb. 1.: *Veranschaulichung eines (4,2,3)-Tensors* $\mathcal{X}$. $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, $\mathbf{X}^{(3)}$ *sind die* Scheiben *des Tensors.* $X^{(1)}_{1,:}$, $X^{(1)}_{:,1}$ *und* $X^{(:)}_{1,3}$ *sind Vektoren, über den Spalten-, Zeilen- bzw. Matrizen-Index.* $X^{(:)}_{:,2}$ *ist die Matrix aller Elemente der zweiten Spalten aller Scheiben.*

Kapitel 1

Einleitung

Um Roboter zu schaffen, die sich ihrer Umwelt anpassen und intelligent mit Menschen interagieren können, muss ihnen die Fähigkeit zur Wahrnehmung ihrer Umgebung verliehen werden. Besonders die Wahrnehmung der dynamischen Aspekte der Umgebung spielt für die Interaktion eine besondere Rolle. Die Umwelt kann dabei durch verschiedene Modalitäten wahrgenommen und beschrieben werden. Mit dem Aufkommen von Tiefenkameras in den letzten Jahren steht mit dem Tiefeneindruck eine sehr mächtige Sensormodalität zur Verfügung, welche die Erfassung von Objekten in der Umgebung des Roboters stark vereinfacht hat. Zusammen mit frei verfügbaren Software-Bibliotheken können so insbesondere die Bewegungen von Menschen, seien es die Bewegungen ihrer Gliedmaßen oder die Spuren ihrer Laufbewegung, erfasst werden und einer tieferen Analyse unterzogen werden. Ein Teilbereich dieser Art von Analysen stellt die Gestenerkennung dar. Dabei beschreibt der Terminus „Geste“ die Teilmenge aller Bewegungen die zur Kommunikation von Begriffen und Konzepten ausgeführt werden. Die in dieser Arbeit entwickelten Verfahren sind aber nicht auf nur auf die Erkennung von Gesten beschränkt, sondern können allgemeiner eingesetzt werden, weshalb die Modellierungs- und Erkennungsaufgabe im Folgenden allgemeiner als Bewegungserkennung bezeichnet wird.

Die Fähigkeit der Erkennung menschlicher Bewegungen kann in einer Vielzahl von Anwendungen zum Einsatz kommen. Mit dem Bereich der Bewegungsrehabilitation ist in den letzten Jahren eine Anwendung entstanden, die zur Zeit vielerorts erforscht und weiterentwickelt wird. Aufgrund der demografischen Entwicklung der Gesellschaft wird diesem Bereich starke Bedeutung beigemessen. Auch die in dieser Arbeit entwickelten Methoden werden zum Teil anhand des Anwendungsszenarios der Bewegungsrehabilitation untersucht. Sie bettet sich damit in die Arbeiten rund um einen robotischen Assistenten für ältere und eingeschränkte Menschen am FG Neuroinformatik und Kognitive Robotik [Gross, Debes et al., 2014; Gross, Mueller et al., 2015; Scheidig et al., 2014; Schröter et al., 2014] ein.

Als Beispielszenario dient im Folgenden ein Ratespiel, bei welchem der zu rehabilitierende Patient eine Reihe von Bewegungsübungen aus einem vordefinierten Repertoire ausführt, die das interagierende System erkennen und benennen muss. In dem Szenario gibt es drei Akteure: den Experten, das System und den Patienten (oder Benutzer). Das Szenario ist schematisch in Abbildung 1.1 dargestellt. Zunächst wird im Vorfeld, in

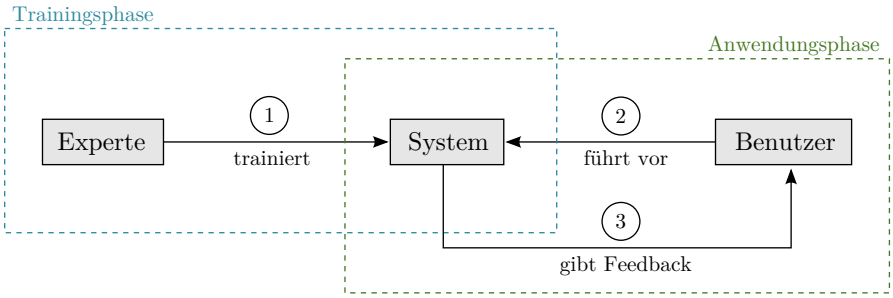


Abb. 1.1.: *Informationsfluss zwischen den Akteuren in der Trainings- und in der Anwendungsphase. In der Trainingsphase trainiert der Experte das System (1). In der Anwendungsphase führt der Patient (oder Benutzer) die Übungen unter Beobachtung des Systems aus (2) und das System gibt dem Anwender Anweisungen oder Feedback über die Qualität der ausgeführten Übungen (3).*

der sogenannten Trainingsphase, das Repertoire an Bewegungsübungen durch einen Experten als Menge von Idealübungen oder *Modellen* definiert. Die Modelle können dabei manuell beschrieben werden oder aus Daten gelernt werden.

Nachdem die Definition der Modelle vorgenommen wurde, kann das System eingesetzt werden. Diese Phase wird als Anwendungsphase bezeichnet. Das System enthält als Wissensbasis die Modelle und der Patient kennt das Repertoire der Übungen. Der Patient führt nun beliebige Übungen aus und das System muss erkennen, welche Übung jeweils ausgeführt wurde. Dazu muss es die Bewegung der Gliedmaßen wahrnehmen (die Datenakquisition), sie in eine Repräsentation überführen die sie mit den gespeicherten Modellen vergleichbar macht (die Vorverarbeitung und die Merkmalsextraktion), den eigentlichen Vergleich mit den gespeicherten Modellen vornehmen und auf Basis der Vergleiche die Bewegung identifizieren (die Klassifikation).

In dieser Arbeit wird das Sparse Coding, ein spezielles Verfahren zur Merkmalsextraktion, als Teil dieser Verarbeitungskette untersucht. Im Folgenden wird das grundlegende Konzept beschrieben.

1.1. Motivation des Sparse Coding für die Merkmalsextraktion in Bewegungstrajektorien

Sparse Coding [Olshausen und D. J. Field, 1996] ist ein biologisch motiviertes Konzept zur Datentransformation, bei dem die Datenmenge durch eine begrenzte Menge an repräsentativen Mustern dargestellt werden soll. Das Hauptthema dieser Arbeit ist die explorative Evaluierung der Eignung von Sparse Coding für die Verwendung in der Merkmalsextraktion eines Mustererkennungsprozesses für Zeitreihen im Allgemeinen und Bewegungstrajektorien im Speziellen.

Der Zweck der Merkmalsextraktion in einem Mustererkennungsprozess ist, den Da-

tenstrom in eine Form zu transformieren, die bedeutungstragende Teile hervorhebt und so die Klassifikation in der folgenden Verarbeitungsstufe erleichtert. So sind z. B. Trajektorien menschlicher Bewegungen dadurch gekennzeichnet, dass ein Mensch aufgrund physischer Beschränkungen nur bestimmte Bewegungen ausführen kann. Des Weiteren ist in einem bestimmten Anwendungsfeld die Menge an bedeutungstragenden Gesten typischerweise stark beschränkt. Das führt dazu, dass in den beobachteten Bewegungen wiederkehrende Muster vorhanden sind. So kann man sich komplexe Bewegungen als aus einzelnen Elementarbewegungen zusammengesetzt vorstellen. Abstrahiert man nun von der konkreten Form der Elementarbewegungen und reduziert die Trajektorie auf eine Form, die angibt, wann welche Elementarbewegung vorgekommen ist, so erhält man eine sehr stark reduzierte Repräsentation der Daten (siehe Abbildung 1.2). Dabei kann man die Elementarbewegungen als Merkmale interpretieren, die aus dem Rohdatenstrom extrahiert werden können. Diese Darstellung hat unter anderem den Vorteil, dass nachgelagerte Klassifikatoren auf dieser reduzierten Form sehr effizient arbeiten können.

Beim Sparse Coding ist das Ziel, die Datenmenge oder den Datenstrom durch eine begrenzte Menge an repräsentativen Mustern darzustellen. In dieser Arbeit werden diese Muster „Primitive“ genannt. Die Primitive werden dabei in einer Trainingsphase aus den Daten gelernt und in der Anwendungsphase zur Transformation oder Codierung des Datenstromes verwendet. Das Lernen erfolgt dabei in einem Optimierungsprozess, der in Kapitel 4 detailliert erläutert wird. Die auf Primitiven basierende Darstellung hat unter anderem die folgenden Vorteile:

Rauschunempfindliche Repräsentation Während der Anwendungsphase erfolgt die Codierung eines Eingangsdatums durch Detektion der Primitive. Diese Detektion ist tolerant gegenüber Rauschen auf dem Eingangsdatum und die codierte Repräsentation des Eingangsdatums ist invariant gegenüber dem Rauschen.

Komprimierte Repräsentation Die Primitive sind an die Eingangsdaten angepasst. Um ein Eingangsdatum darzustellen, müssen nur wenige Primitive aktiviert werden. Dies kann z. B. durch ein Bitmuster repräsentiert sein, wodurch der Speicherbedarf zur Repräsentation vieler Beispiele signifikant reduziert werden kann.

Verteilte Codierung Die Repräsentation durch Aktivierung einer kleinen Teilmenge von Primitiven wird auch als „Verteilte Codierung“ bezeichnet. Im Mustererkennungsprozess nachgelagerte Klassifikatoren können von dieser verteilten Codierung profitieren. Darauf wird in Abschnitt 2.4.2 näher eingegangen.

Daten-Exploration Die Erkenntnis, aus welchen „Bausteinen“ eine Datenmenge besteht kann per se von Interesse sein. So kann z. B. Sparse Coding von Bewegungstrajektorien die Menge der in den Daten „versteckten“ elementaren Bewegungen aufzeigen. Auf diesen Aspekt wird in Abschnitt 2.4.1 eingegangen.

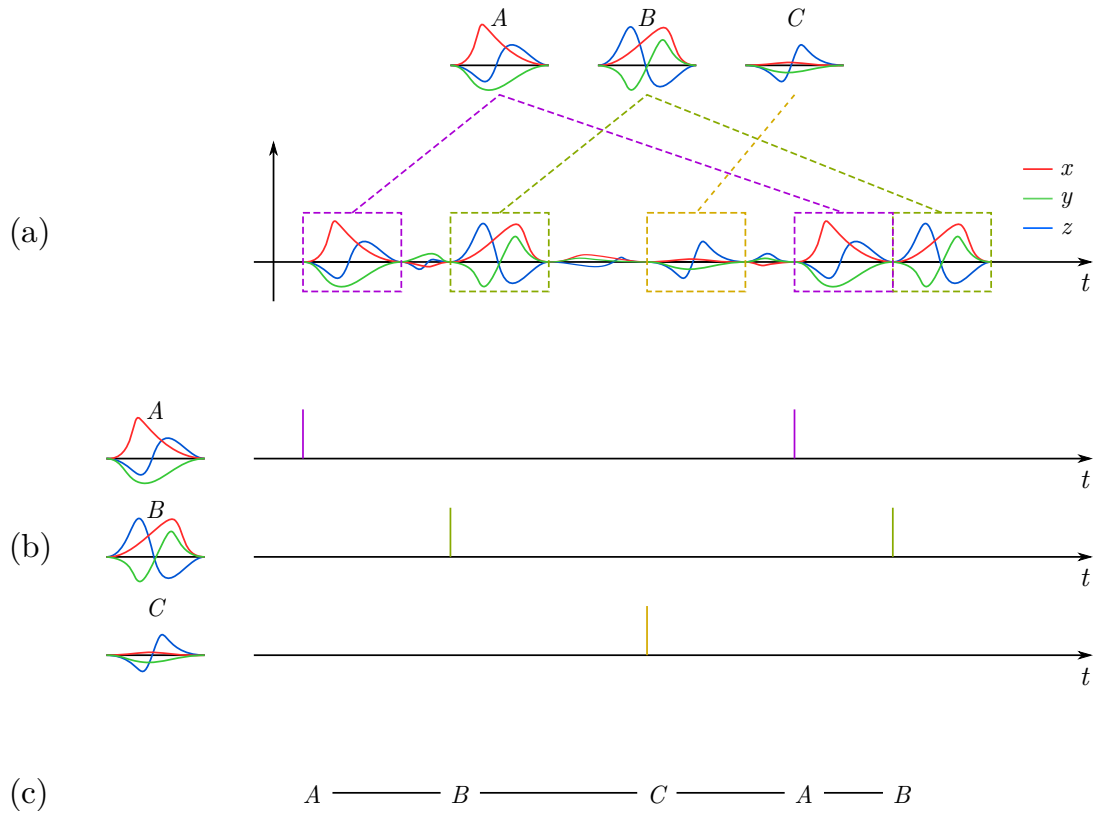


Abb. 1.2.: *Spärliche Repräsentation einer Trajektorie, welche Primitive enthält (a). Abstrahiert man von der konkreten Form der Primitive und speichert nur die Information über das zeitliche Auftreten, so erhält man eine sehr kompakte Repräsentation (b). Man kann noch einen Schritt weiter gehen und auch die konkreten Zeiten ignorieren und nur die Reihenfolge des Auftretens der Primitive speichern (c) und so zu einer Repräsentation in Form von Symbolfolgen gelangen.*

Modellierung nach biologischem Vorbild In Abschnitt 2.4.1 wird eine der grundlegenden Arbeiten im Bereich des Sparse Coding vorgestellt, welche das Konzept auf Pixel-Bildern natürlicher Szenen anwendet. Dabei entstehen Wavelet-artige Primitive, die identisch zu bestimmten Rezeptoren des visuellen Systems von Säugetieren sind. Damit kann das Sparse Coding als ein Modell eines bestimmten Teilprozesses der biologischen Informationsverarbeitung interpretiert werden.

Durch diese aufwändige Codierung wird ein großer Teil der Erkennungsleistung bereits in der Merkmalsextraktion umgesetzt. Dies hat erhebliche Auswirkungen auf die Gestaltung der Abläufe beim Training des Systems. Weiter oben wurde dargestellt, dass das Szenario im Wesentlichen aus drei Akteuren, dem Experten, dem System und dem Patienten besteht, wobei der Experte in einer Lernphase die Parameter des Systems definiert und der Patient in der Anwendungsphase mit dem System interagiert. Bei genauerer Betrachtung ergibt sich aber, dass die Lernphase selbst aus mehreren

Teilschritten besteht. Die beiden in der Lernphase zu trainierenden Teilsysteme sind die Merkmalsextraktion und die Klassifikation. Für beide Teilsysteme müssen Parameter definiert oder gelernt werden. Dabei muss zuerst die Merkmalsextraktion trainiert werden, da die Klassifikation bereits transformierte Trainingsdaten benötigt.

Diese Zweistufigkeit kann man ausnutzen, um den Aufwand beim Anlernen des Systems auf eine spezifische Anwendung zu reduzieren. Im Folgenden wird das Beispiel der Rehabilitationsübungen betrachtet. Im allgemeinen Fall müsste der Therapeut zum Anlernen des Systems eine relativ große Menge von Trainingsbeispielen aller Übungen aus dem Repertoire sammeln, um Merkmalsextraktion und Klassifikation zu trainieren. Die Merkmalsextraktion würde ein Basisalphabet lernen, welches an das ausgewählte Repertoire angepasst ist. Ist die Variation im Repertoire allerdings hinreichend groß, so ist anzunehmen, dass das Basisalphabet auch genereller anwendbar, also auch für Übungen verwendbar ist, die nicht im Repertoire enthalten sind. Ein weiterer, vorgelagerter Experte (z.B. der „Hersteller“ des Systems) könnte also bereits ein Basisalphabet für eine grobe Anwendungsdomäne, wie z. B. „menschliche Bewegungen“, in Trainingsphase A (Training der Merkmalsextraktion) anlernen und der Experte für die spezifische Anwendung - in unserem Fall der Therapeut - könnte dieses vorab gelernte Alphabet verwenden und müsste nur noch mit relativ wenigen Trainingsbeispielen den Klassifikator in Trainingsphase B (Training des Klassifikators) lernen oder könnte die konkreten Übungen im Idealfall sogar in einer symbolischen Form aus Elementen des Basisalphabets manuell zusammenstellen.

Ein verwandtes Beispiel aus der Sprachverarbeitung kann hier als Analogie dienen. Beim Mustererkennungsprozess in der Sprachverarbeitung wird, ähnlich wie hier, eine aufwändige Merkmalsextraktion durchgeführt. Im einfachsten Fall dienen Phoneme als Elemente oder Primitive der Sprache, und der Klassifikator modelliert Sequenzen dieser Phoneme. Die Menge der Phoneme unterscheidet sich zwischen verschiedenen Sprachen. Würde man Phoneme aller Sprachen der Welt in der Merkmalsextraktion modellieren, so wäre die Erkennungsleistung des Gesamtsystems relativ schlecht. Deshalb beschränkt man typischerweise die Menge der Phoneme auf die Teilmenge, welche tatsächlich in der konkreten Sprache vorkommen, für die das Gesamtsystem aktuell eingesetzt wird. So kann ein Hersteller eines Sprachverarbeitungssystems, Phonem-Sätze (Merkmale) für verschiedene Sprachen anbieten, und der Anwender wählt einen dieser fertigen Parametersätze und muss nur noch die Sequenzen im Klassifikator für die in seiner Anwendungsdomäne typischen Wörter oder Sätze trainieren. Die Phoneme entsprechen hier den Bewegungsprimitiven und die verschiedenen Sprachen würden groben Anwendungsklassen, wie „Sport-Bewegungen“, „Manipulation“, etc. entsprechen.

1.2. Ziele und Beiträge dieser Arbeit

In dieser Arbeit soll das Sparse Coding [1] auf seine Eignung als Merkmalsextraktionsschritt für die Generierung neuer Trajektorien und die Klassifikation beobachteter Trajektorien untersucht werden. Dazu soll ein Verfahren zur Anwendung von Sparse Coding auf Bewegungstrajektorien weiterentwickelt werden. Es soll untersucht werden, inwieweit sich in vorangegangenen Arbeiten (siehe Abschnitt 1.3) auf Basis einer kleinen Datenmenge aufgestellte Thesen über die Eignung von Sparse Coding zum Lernen für nachgelagerte Verarbeitungsschritte nützlicher Muster (Primitive) in Bewegungsdaten in realen Daten und Benchmark-Datensätzen stützen lassen. Ziel ist es, ein generisches Verfahren zum Lernen versteckter Primitive in Bewegungsdaten zu entwickeln, welches es erlaubt, komplexe Trajektorien in einer Art symbolischen Beschreibung als Folge der gelernten Primtive zu beschreiben. Damit soll die Komplexität der Daten durch ein lernendes Verfahren in der Merkmalsextraktion reduziert werden, sodass die nachgelagerten Verarbeitungsschritte der Generierung und Klassifikation mit sehr einfachen Modellen realisiert werden können. In Kapitel 7 wird darauf eingegangen, inwiefern die hier aufgelisteten Ziele tatsächlich erreicht werden konnten.

Die Beiträge dieser Arbeit werden im Folgenden in abfallender Reihenfolge nach ihrem Anteil an der Arbeit aufgelistet:

Lernen von Merkmalen mittels Sparse Coding Den Hauptteil dieser Arbeit stellt die Entwicklung eines Verfahrens zum Lernen von Merkmalen in menschlichen Bewegungstrajektorien dar. Wie bereits oben angedeutet, ist die Extraktion von gelernten Merkmalen ein Schritt im Mustererkennungsprozess. Die Merkmale können dabei heuristisch definiert sein oder aus Daten gelernt werden. In dieser Arbeit wird das Sparse Coding als Technik für das Lernen von Merkmalen untersucht und auf den speziellen Fall von Bewegungstrajektorien angepasst. Das entwickelte Verfahren bietet die Möglichkeit, Bewegungstrajektorien sehr kompakt darzustellen und kann zur Merkmalsextraktion eingesetzt werden, wodurch der Aufwand für nachgelagerte Verarbeitungsschritte, wie Klassifikation und Prädiktion erheblich reduziert wird.

Klassifikation auf Basis von Sparse Coding Setzt man das in dieser Arbeit entwickelte Verfahren zur Merkmalsextraktion ein, so erhält man eine sehr kompakte Beschreibung für Bewegungstrajektorien. Diese Beschreibung ermöglicht den Einsatz sehr einfacher Modelle zur Klassifikation. In dieser Arbeit wird der Einfluss der Merkmalsextraktion auf den Klassifikationsschritt untersucht. Dabei werden verschiedene Modelle zur Klassifikation verwendet, die von der spärlichen Codierung profitieren.

Generierung von Trajektorien auf Basis von Sparse Coding Die Merkmalsextraktion lässt sich auch im umgekehrten Sinne für die Erzeugung neuer Trajektorien

[1] Genauer, die Nicht-negative Matrixfaktorisierung (NMF) [Lee et al., 1999]

anwenden. Die Trajektorien einer bestimmten Klasse weisen oft ein typisches Aktivierungsmuster von Merkmalen auf. Lernt man dieses Aktivierungsmuster für alle Klassen von Trajektorien aus einem Trainingsdatensatz, so kann man neue Trajektorien einer bestimmten Klasse erzeugen, indem man die gelernten Merkmale in der klassenspezifischen Weise aktiviert. In dieser Arbeit wurde ein Modell für die Generierung von Trajektorien nach diesem Schema entwickelt.

Demonstrator-Applikation Die im Rahmen dieser Arbeit entwickelten Verfahren wurden als Module einer Gesamtapplikation implementiert. Diese wird im Folgenden „Demonstrator“ genannt. Die Module des Demonstrators implementieren alle Schritte des Mustererkennungsprozesses, von der Datenaufnahme mittels einer Tiefenkamera, über das Skelett-Tracking, die Vorverarbeitung, die Merkmalsextraktion, unter anderem mittels Sparse Coding, bis hin zur Klassifikation. Um diese Applikation echtzeitfähig zu gestalten, mussten vor allem die Implementationen der rechenaufwändigen Algorithmen zum Sparse Coding im Hinblick auf Geschwindigkeit und Speichereffizienz optimiert werden. Des Weiteren wurden alle Module mit Hilfe des Robotik-Frameworks MIRA [Einhorn et al., 2012; Mira 2015] integriert.

Im Vordergrund der Arbeit steht die Evaluierung des Sparse Coding als alternatives Werkzeug zur Merkmalsextraktion im Mustererkennungsprozess. Obwohl ein vollständiges System bis hin zur Klassifikation und Generierung entwickelt wurde, war es nicht das Ziel der Arbeit einen Klassifikator zu entwickeln, der an der Leistungsfähigkeit aktueller High-End-Klassifikatoren gemessen werden kann. Vielmehr dient das System als Rahmen zur Evaluierung und Demonstration der Möglichkeiten des Sparse Coding und soll auch als solches verstanden werden.

1.3. Vorangegangene Arbeiten

Diese Arbeit baut auf ersten Untersuchungen zum Sparse Coding für Bewegungstrajektorien in der Dissertation von Hellbach [Hellbach, 2009] auf. Im Folgenden sollen kurz die Unterschiede und Erweiterungen gegenüber der Arbeit von Hellbach erläutert werden. Hellbach untersucht die Verwendbarkeit der Nicht-negativen Matrix Faktorisierung (NMF) für die Dekomposition von Bewegungstrajektorien. Dafür wird eine Trajektorie durch Quantisierung und Rasterisierung in ein Binärbild überführt, wodurch die NMF exakt wie auf beliebigen anderen Binärbildern eingesetzt werden kann. Damit ist es Hellbach möglich, einige wenige Primitive in dieser Repräsentation zu lernen. Er beschreibt außerdem ein Verfahren zur Prädiktion von Trajektorien auf Basis der gelernten Primitive. Diese Arbeit erweitert die Arbeiten von Hellbach im Wesentlichen um die folgenden Aspekte:

Direkte Repräsentation Während Hellbach, wie oben beschrieben, zuerst eine Transformation in ein Binärbild vornimmt, wird die Trajektorie in dieser Arbeit haupt-

sächlich in ihrer ursprünglichen Form als Vektor interpretiert. Das Sparse Coding operiert dann direkt in dem dadurch beschriebenen Vektorraum. Dies bringt einen enormen Geschwindigkeitszuwachs und macht die Verarbeitung von Datenmengen der Größe der in dieser Arbeit verwendeten Benchmark-Datensätze erst möglich.

Lokale Spärlichkeit Hellbach verwendet in seiner Arbeit die Standardformulierung der NMF. In dieser Arbeit wird diese Formulierung um einen Term in der Energiegleichung des Optimierungsproblems erweitert, die bestimmte negative Eigenschaften der Standardformulierung beseitigt und so die Interpretation einer Trajektorie, als aus einer Aneinanderreihung von Primitiven zusammengesetzt, unterstützt.

Klassifikation und Generierung Während das Hauptaugenmerk von Hellbachs Arbeiten auf der prinzipiellen Verwendbarkeit und einer ersten Evaluation von Sparse Coding für Bewegungstrajektorien liegt, wird in dieser Arbeit neben einer weitergehenden Evaluierung vor allem auch die Verwendung und der Nutzen der gewonnenen spärlichen Repräsentation für nachgelagerte Verarbeitungsschritte, wie die Klassifikation und die Generierung neuer Trajektorien untersucht.

Systemintegration Um den anwendungsspezifischen Aspekt dieser Arbeit zu unterstützen, wurde ein vollständiges System zur Demonstration der Eigenschaften und Leistungsfähigkeit der Verfahren entwickelt. Dieses System ist modular aufgebaut und basiert auf dem Robotik-Framework MIRA. Es implementiert alle Verarbeitungsschritte eines Mustererkennungsprozesses. Um dies zu realisieren, musste vor allem die Implementierung des Sparse Coding durch massive Parallelisierung auf Echtzeitfähigkeit hin optimiert werden.

Systematische Evaluierung auf mehreren Datensätzen Während Hellbach Sparse Coding eher prinzipiell an einem Datensatz, bestehend aus Bewegungstrajektorien von vier Bewegungsarten, untersucht, werden die Verfahren in dieser Arbeit an einem breiteren Spektrum von Bewegungsdaten evaluiert. Das Spektrum reicht dabei von einfachen und komplexen Gesten, die im eigenen Labor aufgenommen wurden, bis hin zu Datensätzen aus dem Bereich der Activity Recognition, die komplexe Bewegungen aus Tätigkeiten im Haushaltsumfeld oder der industriellen Fertigung enthalten.

1.4. Gliederung der Arbeit

Diese Arbeit ist wie folgt gegliedert. In Kapitel 2 wird ein Überblick über den aktuellen Stand der Forschung im Bereich der Modellierung und Erkennung menschlicher Bewegungen und Gesten vorgestellt. Dabei wird zunächst auf klassische Verfahren eingegangen. Danach werden Methoden vorgestellt, die, wie das in dieser Arbeit vorgestellte Verfahren, auf Sparse Coding beruhen.

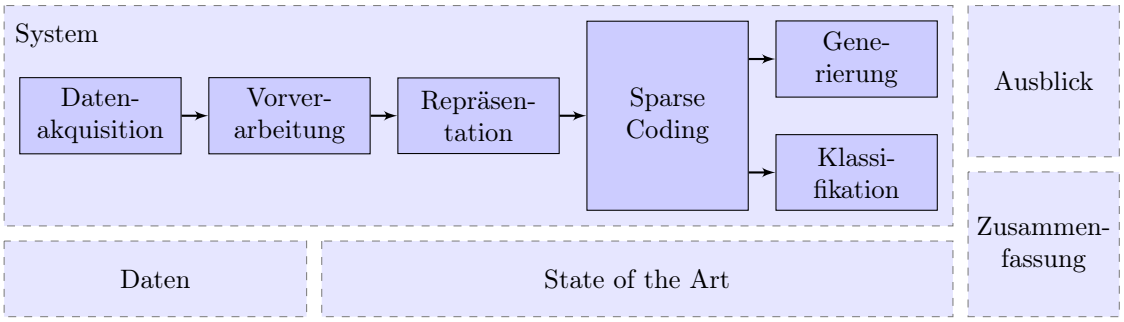


Abb. 1.3.: *Gliederungsgrafik, welche die Arbeit begleitet und als Orientierungshilfe dienen soll.*

In Kapitel 3 wird zunächst das angezielte Rahmenszenario für diese Arbeit vorgestellt. Danach wird ein im Rahmen dieser Arbeit entwickeltes echtzeitfähiges Gesamtsystem vorgestellt, welches den vollständigen Mustererkennungsprozess implementiert. Es wird vor allem auf die Datenaufnahme und die nötigen Vorverarbeitungsschritte für den Einsatz des Sparse Coding eingegangen. Um einen Eindruck der im Rahmen der Arbeit verwendeten Daten zu vermitteln, werden weiterhin mehrere Datensätze vorgestellt, die im weiteren Verlauf der Arbeit für Experimente und Veranschaulichungen der Eigenschaften der entwickelten Verfahren herangezogen werden.

In Kapitel 4 wird das in dieser Arbeit entwickelte Merkmalsextraktionsverfahren vorgestellt, welches den Hauptteil der Arbeit darstellt. Dabei wird das generelle Konzept des Sparse Codings erläutert, das Sparse Coding formal als Optimierungsproblem eingeführt und notwendige Anpassungen für die Verwendbarkeit in der Analyse von Bewegungstrajektorien vorgestellt.

In Kapitel 5 wird auf die Auswirkungen des entwickelten Merkmalsextraktionsverfahrens auf die nachgelagerten Schritte der Generierung und Klassifikation von Trajektorien eingegangen. Dabei werden verschiedene Modelle vorgestellt und ihre Eigenschaften bezüglich der jeweiligen Aufgabe analysiert.

In Kapitel 6 werden Ideen für fortführende Arbeiten vorgestellt. Dabei werden einige Fragestellungen beleuchtet, die aus Zeitgründen in dieser Arbeit offen geblieben sind. Kapitel 7 fasst die Inhalte der Arbeit zusammen.

Die Arbeit wird durch eine Grafik begleitet, die als Leseleitfaden dienen soll (siehe Abbildung 1.3). Die Grafik orientiert sich dabei am Ablauf des Mustererkennungsprozesses. Die einzelnen Blöcke der Grafik werden jeweils als Kapitel oder Teilkapitel in dieser Arbeit behandelt. Die Grafik wird am Anfang jedes Kapitels gezeigt, wobei der zum aktuellen Kapitel gehörende Block jeweils hervorgehoben ist.

1.5. Eigene Publikationen

Im Folgenden wird ein Überblick über die während der Arbeit am FG Neuroinformatik und Kognitive Robotik entstandenen Publikationen gegeben.

1.5.1. Publikationen mit direktem Bezug zum Thema der Arbeit

In den folgenden Publikationen wurden Teile des in dieser Arbeit behandelten Themas veröffentlicht.

Hellbach, Vollmer, Eggert: „Learning Motion Primitives using Spatio-Temporal NMF“, *DAGM Workshop New Challenges in Neural Computation*, 2011 [Hellbach et al., 2011] stellt erst Ergebnisse der Dekomposition einer kleinen Menge einfacher Bewegungstrajektorien auf Grundlage der gitterbasierten Repräsentation vor (siehe Abschnitt 4.4). Es wird gezeigt, dass es grundsätzlich möglich ist, interpretierbare Basisvektoren zu erzeugen.

Vollmer, Eggert, Groß: „Modeling Human Motion Trajectories by Sparse Activation of Motion Primitives Learned from Unpartitioned Data“, *German Conference on Artificial Intelligence (KI)*, 2012 [Vollmer et al., 2012b] zeigt die Generierung von Handschrift-Trajektorien auf Basis gelernter Primitive und gelernter Modelle zur klassentypischen Aktivierung der Primitive (siehe auch Abschnitt 5.1). Statt der gitterbasierten Repräsentation wird hier die direkte Repräsentation einer Trajektorie als Vektor verwendet. Es wird erstmals die Implementierung der lokalen Spärlichkeit vorgestellt (siehe Abschnitt 4.2).

Vollmer, Eggert, Groß: „Generating Motion Trajectories by Sparse Activation of Learned Motion Primitives“, *International Conference on Artificial Neural Networks (ICANN)*, 2012 [Vollmer et al., 2012a] ist ähnlich zur obigen Publikation, hebt aber das unüberwachte Lernen von Primitiven aus einem unpartitionierten (nicht segmentierten) Datenstrom hervor. Des Weiteren werden die Wirkungen bestimmter Parameter, wie der Anzahl der Primitive, anhand von Experimenten demonstriert (siehe Abschnitt 4.5).

Vollmer, Groß, Eggert: „Learning Features for Activity Recognition with Shift-Invariant Sparse Coding“, *International Conference on Artificial Neural Networks (ICANN)*, 2013 [Vollmer, Gross et al., 2013] zeigt die generellere Anwendbarkeit des Verfahrens zur Merkmalsextraktion in der Activity Recognition. Anhand mehrerer Benchmark-Datensätze wird gezeigt, dass sich die gelernten Primitive als Features zur Klassifikation eignen (siehe Abschnitt 5.2).

Vollmer, Hellbach, Eggert, Groß: „Sparse Coding of Human Motion Trajectories with Non-negative Matrix Factorization“, *Neurocomputing*, 2014 [Vollmer, Hellbach et al., 2014] ist eine frühe Journalpublikation und zeigt das Lernen von Primitiven auf Grundlage der gitterbasierten Repräsentation (siehe Abschnitt 4.4). Außerdem wird die Idee der Prädiktion ausführlicher vorgestellt und es wird ein erstes Modell zur Klassifikation, basierend auf den gelernten, lokal spärlichen Aktivierungen (siehe Abschnitt 4.2), vorgestellt.

1.5.2. Weitere Publikationen des Autors ohne direkten Bezug zur Arbeit

Die folgenden weiteren Publikationen des Autors stehen nicht in unmittelbarem Zusammenhang mit dem Thema dieser Arbeit und werden nur der Vollständigkeit halber aufgelistet.

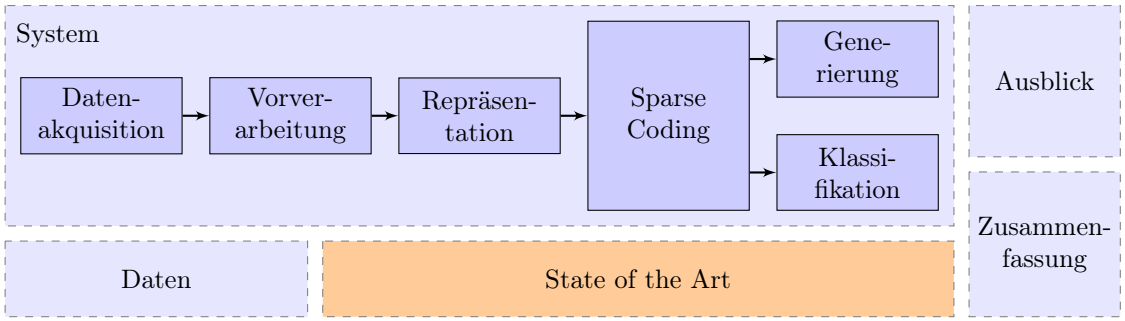
Henry, Vollmer, Ferris, Fox: „Learning to Navigate through Crowded Environments“, *International Conference on Robotics and Automation (ICRA)*, 2010 [Henry et al., 2010] beschreibt das Lernen von Navigationsstrategien für einen mobilen Roboter, der sich in Menschenmengen bewegen muss.

Vollmer, Schaffernicht, Groß: „Exploring Continuous Action Spaces with Diffusion Trees for Reinforcement Learning“, *International Conference on Artificial Neural Networks (ICANN)* [Vollmer, Schaffernicht et al., 2010] zeigt ein Verfahren zur Exploration kontinuierlicher Aktionsräume im Reinforcement Learning.

Weinrich, Vollmer, Groß: „Estimation of human upper body orientation for mobile robotics using an SVM decision tree on monocular images“, *International Conference on Intelligent Robots and Systems (IROS)*, 2012 [Weinrich et al., 2012] stellt ein Verfahren zur Schätzung der Orientierung des Oberkörpers in Kamerabildern vor.

Kapitel 2

Stand der Forschung



Dieses Kapitel gibt einen Überblick über einige Publikationen mit direktem Bezug zur vorliegenden Arbeit. Nach einer Definition der Begriffe „Geste“ und „Gestenerkennung“ wird zunächst auf klassische Verfahren zur Repräsentation und Klassifikation von Gesten eingegangen. Danach werden grundlegende Arbeiten zum Sparse Coding vorgestellt, welche die Anwendung des Verfahrens als Merkmalsextraktionsschritt im Mustererkennungsprozess motivieren. Am Ende wird auf jüngere Publikationen zur Anwendung von Sparse Coding auf Bewegungsdaten eingegangen, welche einen ähnlichen Ansatz wie die vorliegende Arbeit verfolgen.

2.1. Definition von „Geste“ und „Gestenerkennung“

[Pavlovic et al., 1997] beschreibt den Prozess der Gestenerkennung als ein Zusammenspiel aus der ausführenden Person, welche das „mentale Konzept“ der Geste durch eine Bewegung in ein sichtbares Signal umwandelt und dem Erkennungssystem, welches dieses Signal wahrnimmt, erkennt und in eine formale Beschreibung übersetzt.

Der Begriff der Gestenerkennung ist in der Literatur nur sehr unklar definiert und wird für viele unterschiedliche Arten von Erkennungsaufgaben verwendet. So kann als Geste die intrinsische Bewegung der Hand [Suarez et al., 2012] gemeint sein, oder auch die Bewegung eines oder beider Arme [Zhang et al., 2013]. Selbst die Mimik des Gesichtes oder die Bewegung des gesamten Körpers [Mitra et al., 2007] können als Geste bezeichnet werden. Des Weiteren kann eine Geste als einzelne Pose z. B. der Hand oder der Arme

definiert sein [LaViola, 1999], als Folgen von Posen oder als Trajektorie über die Zeit [Bobick et al., 1995].

Den meisten Beschreibungen gemein ist das Konzept, dass Gesten durch einen statischen und einen dynamischen Aspekt definiert sind [Just et al., 2009; Pavlovic et al., 1997]. Der statische Aspekt besteht darin, dass eine Geste zu einem bestimmten Zeitpunkt als ein statisches Gestenmodell durch einen Punkt in einem Konfigurationsraum (z. B. der Raum der Gelenkwinkel eines Skelettmodells, Parameter eines Gesichtsmodells, Position einer Hand im 3D-Raum, der Raum der 200x200 Grauwertbilder etc.) beschrieben werden kann. Der dynamische Aspekt besteht darin, dass eine Geste durch eine Folge oder Trajektorie von Konfigurationen in diesem Raum beschrieben wird.

In dieser Arbeit werden Gesten als gezielte Bewegungen eines Arms betrachtet, mit denen die Person einen bestimmten Begriff oder ein Symbol vermitteln will. Es wird also davon ausgegangen, dass die Person eine Reihe klar definierter Gesten kennt und einer Geste eine klare Bedeutung hat. Die Geste ist dabei im einfachsten Fall durch die Trajektorie der Position der Hand im 3D-Raum definiert.

2.2. Repräsentation von Gesten

Um eine Geste zu repräsentieren, müssen geeignete Mittel sowohl für den statischen, als auch für den dynamischen Aspekt definiert werden [Kaaniche, 2009]. Der statische Aspekt einer Geste wird, wie oben beschrieben, durch einen Punkt im gewählten Konfigurationsraum beschrieben. Konventionelle Verfahren lassen sich bezüglich der Repräsentation des statischen Aspekts grob in zwei Gruppen einteilen: modellbasierte Verfahren und ansichtsbasierte Verfahren.

Bei den modellbasierten Verfahren wird das beobachtete Subjekt (eine Person oder die Hand einer Person) durch ein physisches Modell nachgebildet. Dazu wird meist ein Analyse-durch-Synthese-Ansatz [Edwards et al., 1998] verwendet, der ein Modell der Person oder der Hand durch Variation der beschreibenden Parameter auf die beobachtete Person anpasst. Die Parameter dieses Modells konstituieren den Konfigurationsraum der Geste.

Da die Bestimmung der Modellparameter sehr aufwändig ist, haben sich in den neunziger Jahren ansichtsbasierte Methoden durchgesetzt. Solche Verfahren [T. E. Starner et al., 1995; Yamato et al., 1992] arbeiten rein auf Basis von Kamerabildern. Der Konfigurationsraum der Geste ist durch den Raum der möglichen Kamerabilder definiert. Eine Folge von Bildern in einer Videosequenz definiert eine Trajektorie durch diesen Raum.

Bei Verfahren zur Gestenerkennung ist insbesondere die Forderung bestimmter Invarianzeigenschaften von Bedeutung. Eine Geste soll also unabhängig von der relativen Position zum Aufnahmesystem (Rotations- und Translationsinvarianz) erkannt werden können. Des Weiteren soll sie unabhängig von physiologischen Eigenschaften, wie der Körpergröße der ausführenden Person, sein (Skalierungsinvarianz). Von besonderer Be-

deutung bei der Gestenerkennung ist aber vor allem die Toleranz gegenüber zeitlichen Variationen während der Ausführung der Geste.

Während die räumlichen Invarianzen, also Rotations-, Translations- und Skalierungsinvarianz durch geeignete Repräsentation und Normierung der Daten erreicht werden [1], können die zeitlichen Variationen nur durch Verwendung spezieller Verfahren behandelt werden. Besonders aufgrund dieser Eigenschaft ist eine große Anzahl an Verfahren zur Gestenerkennung entstanden, die das Problem auf teilweise sehr unterschiedlichen Wegen lösen.

2.3. Klassische Verfahren zur Gestenerkennung

Das eigentliche Problem der Erkennung der Geste als Folge oder Trajektorie im gewählten Konfigurationsraum lässt sich weiter untergliedern in die zwei Teilprobleme Spotting (auch Segmentierung) und Inferenz [Just et al., 2009; Kim, 1999].

Spotting versucht, Beginn und Ende einer Geste in einem kontinuierlichen Datenstrom zu finden. Der Datenstrom enthält im Allgemeinen bekannte und unbekannte Gesten in zufälliger Folge. Die unbekannten Gesten (wirklich unbekannte Gesten oder Transitionen zwischen bekannten Gesten) müssen erkannt und ggf. eliminiert werden. Wenn die Geste isoliert worden ist, kann sie an den Klassifikator weitergegeben werden.

Laut [Pavlovic et al., 1997] ist eine Geste im psychologischen Sinne durch die drei Phasen „Preparation“, „Nucleus“ und „Retraction“ gekennzeichnet. Am Beispiel einer Hand betrachtet, bewegt sich die Hand in der Vorbereitungsphase aus der Ruheposition, um eine Geste zu beginnen. Im Nucleus wird die Geste ausgeführt und in der Retraktionsphase bewegt sich die Hand vom Ende der Geste wieder in eine Ruhelage. Die Autoren führen aus, dass die Phasen Preparation und Retraction häufig durch schnelle Bewegungen gekennzeichnet sind und die Phase Nucleus durch eine relativ langsame Bewegung. Diese Eigenschaft wird häufig genutzt, um Gesten zu isolieren. Dabei werden einfache kinetischen Merkmale der Trajektorie, wie Geschwindigkeit und Beschleunigung, auf das Vorkommen bestimmter Werte oder Wertekombinationen, wie lokale Minima, Nulldurchgänge, Werte unterhalb einer experimentell ermittelten Schwelle etc. untersucht [Glowinski et al., 2009].

Wurde das Gestenintervall durch Spotting gefunden, so kann die isolierte Geste durch einen Klassifikator kategorisiert werden. Das besondere an Klassifikatoren für Gesten ist, dass sie den dynamischen Aspekt modellieren können müssen. Die klassischen Vertreter dieser Art von Verfahren sind zeitliche Neuronale Netze (TDNN), Dynamic Time Warping (DTW) und automatenbasierte Modelle wie diskrete Zustandsautomaten (FSM) und Hidden-Markov-Modelle (HMM). Da sich hauptsächlich auf HMM basierende Verfahren durchgesetzt haben, werden im Folgenden nur Vertreter dieser Klasse vorgestellt.

[1] Die Ermittlung der Transformation zwischen Person und Kamera und der Körpergröße ist im Falle des in dieser Arbeit verwendeten Trackingsystems sehr einfach, da die gesamte Person im 3D-Raum getrackt wird. Details dazu sind in Abschnitt 3.2.1 beschrieben.

Für einen ausführlichen Überblick auch über die anderen Verfahren siehe [Mitra et al., 2007].

Das HMM ist ein probabilistisches Verfahren zur Beschreibung von Zeitreihen [Rabiner, 1989]. Es zeichnet sich besonders durch die Eigenschaft aus, zeitliche und räumliche Variationen durch eine explizite statistische Modellierung behandeln zu können. Ein HMM besteht aus einer diskreten Menge an Zuständen und Transitionen zwischen diesen Zuständen. Jedem Zustand ist eine Wahrscheinlichkeitsverteilung über dem Konfigurationsraum zugeordnet. Eine dynamische Geste ist durch einen Pfad zwischen den Zuständen definiert. Die Parameter des HMM können aus Trainingsdaten gelernt werden. HMMs wurden ursprünglich für die Spracherkennung entwickelt [Rabiner und Levinson, 1981]. Aufgrund der strukturellen Ähnlichkeit zwischen Sprache und Schrift, konnten sie auch erfolgreich für die Handschrifterkennung eingesetzt werden [T. Starner et al., 1994]. [Yamato et al., 1992] beschreibt eines der ersten Systeme für die Gestenerkennung auf Basis von HMM. Das ansichtsbasierte System wird zur Klassifikation von Bildfolgen beim Tennisspielen eingesetzt. Es wird gezeigt, dass es möglich ist, sechs unterschiedliche typische Schlagarten beim Tennisspielen mit einer Erkennungsrate von 90% zu erkennen.

Ein häufiges Zielszenario für den Einsatz auf HMM basierender Systeme ist die Zeichensprache. Wie bei der Spracherkennung und der Handschrifterkennung liegen den Gesten bei der Zeichensprache gewisse grammatikalische Konzepte zugrunde. [T. E. Starner et al., 1995] beschreibt ein System zur Erkennung von amerikanischer Zeichensprache (ASL) in Bildsequenzen. Mit Unterstützung durch einfache Grammatiken konnte eine Erkennungsrate von 98% bei einem Lexikon von 50 Wörtern erreicht werden. Starner beschreibt insbesondere die Eigenschaft der HMMs eine manuelle Segmentierung der Daten unnötig zu machen. Möglich wird das durch das sogenannte „Embedded Training“, bei dem zum Training die HMMs aller Klassen zu einem großen HMM sequentiell konkateniert werden und dieses HMM mit einer entsprechenden Sequenz von Trainingsbeispielen trainiert wird. Bei dieser Art des Trainings findet ein automatisches Alignment der Zustände an die Grenzen der einzelnen Gesten statt, weshalb diese Eigenschaft auch als „Automatic Alignment Property“ bezeichnet wird.

2.4. Grundlagen des Sparse Coding

Bevor im nächsten Abschnitt auf die Anwendung von Sparse Coding auf Bewegungsdaten eingegangen wird, sollen hier zunächst einige grundlegende Arbeiten aus dem Bereich des Sparse Coding vorgestellt werden. Dabei wird zuerst eine aus dem Bereich der Computational Neuroscience stammende Veröffentlichung vorgestellt, die eine biologische Motivation für das Sparse Coding gibt. Danach wird auf informationstheoretische Untersuchungen eingegangen, die das Sparse Coding aus Sicht einer effizienten Codierung von Informationen für technische Anwendungen motivieren.

2.4.1. Hintergrund aus der Computational Neuroscience

Wie codiert das Gehirn Informationen so, dass sie effizient gespeichert werden können und für nachgelagerte Verarbeitungsstufen günstig repräsentiert sind? Olshausen und Field untersuchen in [Olshausen und D. J. Field, 1996] ein Modell zur Beschreibung der Codierung von Bildern in einer frühen Verarbeitungsstufe des Visuellen Cortex V1. Im besonderen Interesse stehen dabei die Simple Cells, die zur Wahrnehmung räumlich lokalisierter, orientierter Kanten dienen. Das Antwortverhalten von Simple Cells lässt sich mit orientierten Wavelets unterschiedlicher Frequenz beschreiben [Hubel et al., 2009]. Ziel der Untersuchungen von Olshausen und Field ist es, eine codierungstheoretische Begründung für dieses Antwortverhalten zu finden, also sowohl für die Form ihrer rezeptiven Felder, als auch für die Form ihrer Spike-Aktivitäten. Anders ausgedrückt, versuchen sie zu ergründen, warum der Visuelle Cortex V1 in der evolutionsgeschichtlichen Entwicklung Rezeptoren mit genau diesem Antwortverhalten ausgeprägt hat.

Um dieses Verhalten zu modellieren, repräsentieren Olshausen und Field das Bild als lineare Überlagerung von Basisfunktionen

$$I(x,y) = \sum_i a_i \phi_i(x,y) \ , \tag{2.1}$$

wobei I das Bild als Funktion der Bildkoordinaten x und y ist, ϕ_i die Basisfunktionen (oder die Basis) und a_i die linearen Koeffizienten sind. Eine Zelle wird dabei durch jeweils eine Basisfunktion repräsentiert und die Aktivierung dieser Zelle für das gegebene Bild als der zugehörige Koeffizient.

Die Gesamtheit der Aktivierungen aller Basisfunktionen für ein gegebenes Bild wird als der „Code“ des Bildes bezeichnet. Die Basis kann prinzipiell beliebig gewählt werden, wobei der resultierende Code wesentlich von der gewählten Basis abhängt. Wählt man als Basis z. B. orientierte Wavelets unterschiedlicher Frequenz, so entspricht der Code in vereinfachter Weise dem Antwortverhalten der Simple Cells.

Die Hypothese von Olshausen und Field ist nun, dass durch das Antwortverhalten der Simple Cells ein spärlicher Code (Sparse Code) realisiert wird. Ein Code ist dann spärlich, wenn für ein gegebenes Eingabedatum nur eine kleine Teilmenge von Basisfunktionen aktiviert ist, sich das Eingabedatum also durch wenige Basisvektoren rekonstruieren lässt.

Um nun Basisfunktionen zu finden, die einen solchen Code erzeugen, setzen Olshausen und Field ein unüberwachtes Lernverfahren ein. Dabei handelt es sich um ein Optimierungsverfahren, welches die Basisfunktionen als Parameter einer Zielfunktion optimiert. Dabei beschreibt die Zielfunktion bestimmte Anforderungen an die Variablen, die sich aus der Hypothese von Olshausen und Field ergeben.

Es wird eine Zielfunktion der Form

$$E = -\text{„Informationserhalt“} - \text{„Spärlichkeit“} \tag{2.2}$$

minimiert. Der Informationserhalt kann dabei durch eine Abweichung der linearen Überlagerung vom Originalbild gemessen werden:

$$\text{„Informationserhalt“} = -\sum_{x,y} \|I(x,y) - \sum_i a_i \phi(x,y)\| \,. \tag{2.3}$$

Die Spärlichkeit wird durch eine zunächst abstrakt formulierte Spärlichkeitsfunktion über den Koeffizienten gemessen:

$$\text{„Spärlichkeit “} = -\sum_i S(a_i). \tag{2.4}$$

Die lineare Überlagerung dieser beiden Teilkriteria ergibt die Zielfunktion:

$$E = \sum_{x,y} \left\| I(x,y) - \sum_i a_i \phi_i(x,y) \right\| + \lambda \sum_i S(a_i). \tag{2.5}$$

Mittels eines Gradientenabstiegs werden nun die Parameter a_i und ϕ_i des Modells optimiert. Dabei kontrolliert der Parameter λ die Stärke der Forderung nach Spärlichkeit.

Olshausen und Field machten die spektakuläre Beobachtung, dass die gelernten Basisfunktionen tatsächlich das Antwortverhalten der Simple Cells beschreiben, was ihre Hypothese stark untermauert. Abbildung 2.1 zeigt die so gelernten Basisfunktionen. Hyvärinen und Hoyer können in [Hyvärinen et al., 2001] sogar zeigen, dass ein zweischichtiges lineares Modell das Verhalten von Simple Cells und Complex Cells [2] simultan lernen kann.

Es ist also anzunehmen, dass das Gehirn das Sparse Coding als Prinzip zur Informationsverarbeitung und -repräsentation verwendet. Die Frage, welche Vorteile diese Codierung bringt, bleibt damit aber noch ungeklärt. Olshausen [Olshausen, 2002] vermutet, der Zweck dieser Codierung sei, ein Vokabular für die Repräsentation bedeutungstragender Strukturen in Bildern zur Verfügung zu stellen, bei dem der Datenstrom durch spärliche, statistisch unabhängige Ereignisse dargestellt wird. Field [David J. Field, 1994] führt aus, dass die Spärlichkeit vonnöten sei, weil sie zu jedem Zeitpunkt eine einfache Beschreibung der Strukturen in Sequenzen natürlicher Bilder darstellt. Foldiak und Baum [Baum et al., 1988; Földiak, 2002] erwähnen weiterhin, dass solch eine Repräsentation nützlich sei, um Assoziationen in späteren Verarbeitungsstufen zu bilden und beziehen sich dabei auf die Informationsspeicherung in assoziativen neuronalen Netzwerken. Diese Argumente führen zu einer pragmatischeren Perspektive, die im Folgenden näher betrachtet werden soll.

[2] Complex Cells sind den Simple Cells im Visuellen Cortex V1 nachgeschaltet und aggregieren ihre Antworten, um komplexere Strukturen zu erkennen.

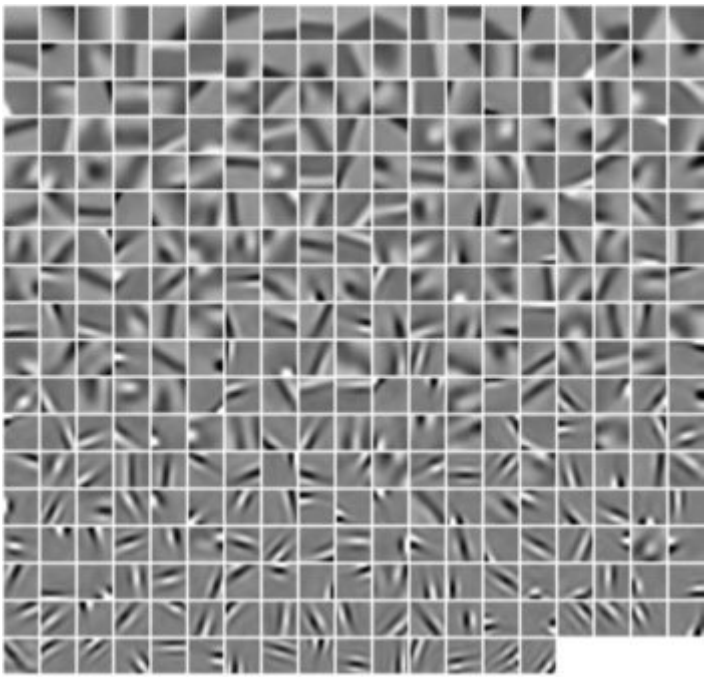


Abb. 2.1.: *Basisvektoren, die aus Bildern natürlicher Szenen gelernt wurden und das Verhalten von Simple Cells erklären. Quelle: [Földiak und Endres, 2008]*

2.4.2. Anwendungsorientierter Hintergrund

Die aktuellen Forschungen zu den Ursachen der Herausbildung bestimmter neuronaler Codierungsparadigmen bewegt sich immer noch im Bereich von Mutmaßungen. Die Grundannahme ist meistens, dass das Gehirn Informationen möglichst effizient codieren muss, vor allem, um Ressourcen, wie Energie, zu sparen und eine schnelle Verarbeitung zu ermöglichen. Auf Basis dieser Annahme wird häufig mit Argumenten aus der Informationstheorie gearbeitet, deren Gegenstand selbst auch die effiziente Codierung von Informationen ist.

Barlow liefert einige erste abstrakte Argumente für den Vorteil einer spärlichen Codierung. So kann laut Barlow [Barlow, 1985, 2009] eine Codierung, welche die Anzahl der aktiven Neuronen minimiert, sogenannte „verdächtige Koinzidenzen“ repräsentieren und detektieren. Barlow interpretiert den Wahrnehmungsprozess als Mittel des Gehirns, ein statistisches Modell der Umwelt zu erstellen und beschreibt mit dem Begriff der „verdächtigen Koinzidenzen“ diejenigen Strukturen oder Ereignisse in einem Signal, welche aus Sicht der Wahrnehmung eine höhere Aufmerksamkeit wert sind. Er beschreibt, dass weder plötzliche Ereignisse, wie das Fallen eines Steins, noch regelmäßige Ereignisse, wie das Ticken einer Uhr einer besonderen Aufmerksamkeit wert sind, wohl aber das gemeinsame Auftreten zweier Ereignisse, insbesondere, wenn dieses gemeinsame Auftreten sehr unwahrscheinlich, also überraschend, ist. Barlow führt weiterhin aus, dass eine Datenrepräsentation oder -codierung genau dann hilfreich für einen Wahrnehmungs-

oder Lernprozess ist, wenn sie wiederkehrende oder koinzidierende Strukturen aufdeckt [Barlow, 1990] und repräsentiert.

Die effiziente Repräsentation und Approximation eines Signals ist eine der fundamentalen Schritte in der Signalverarbeitung und -codierung. Die Effizienz einer Repräsentation ist stark von der Ausnutzung dem Signal inhärenter Merkmale abhängig [Blumensath et al., 2006]. Laut Hoyer [Zhao et al., 2011] ist eine Repräsentation dann von Nutzen, wenn sie die latente Struktur in den Daten explizit macht. Der Autor beschreibt, dass Sparse Coding ein gutes Mittel ist, eine solche Repräsentation als Mittelweg zwischen einer vollständig verteilten Repräsentation (wie sie z. B. mit der PCA gewonnen wird) und einer unären Repräsentation (wie sie mit dem Begriff der „Großmutter-Zelle“ beschrieben werden kann) zu finden. Ein anderes informationstheoretisch motiviertes Beispiel beschreiben Smith und Lewicki [Smith et al., 2005] im Zusammenhang mit der Anwendung von Sparse Coding auf Audiosignale. Sie beschreiben, dass eine nützliche Codierung die Informationsrate (also die Anzahl der übertragenen Symbole pro Codewort) eines Signals reduzieren muss, damit die inhärenten Strukturen sichtbarer werden (denn wird, im Extremfall, nur ein Symbol pro Codewort übertragen, so repräsentiert das Symbol selbst bereits das gesamte Codewort). Aus Sicht von Smith und Lewicki können Codierungsverfahren als Mittel zur Reduktion der Datenrate angesehen werden, die man stufenweise solange anwendet, bis die inhärente Information direkt sichtbar ist. Sie treffen eine Unterscheidung zwischen der sichtbaren Informationsrate, also der Sample-Rate des eigentlichen Signals, und der intrinsischen Informationsrate, also der Rate der „versteckten“ Symbole. In der Sprachverarbeitung ist die sichtbare Informationsrate etwa 50000 $bits/s$, während die Informationsrate der zugrunde liegenden Sprache etwa 200 $bits/s$ beträgt [Rabiner und Levinson, 1981]. Die Autoren führen weiterhin aus, dass die Informationsrate durch Entfernung der Redundanzen, wie die temporale Korrelation zwischen den Samples, reduziert werden kann. Eine Transformation durch Extraktion von Features reduziert die Korrelationen und damit die Redundanzen in den Daten.

Grosse et al. [Grosse et al., 2007] erwähnen, dass die Anwendung des Sparse Coding auf Daten, wie Bilder, Text und Audio, zur Hervorhebung von Mustern mit einem etwas höheren Abstraktionsgrad als dem Rohsignal führt, was zu einen positiven Effekt für die Leistung von Klassifikatoren, wie SVM und Linear Discriminant Analysis (LDA) führt. Auch Field [David J. Field, 1994] beschreibt einen nützlichen Effekt des Sparse Coding [David J. Field, 1994] als direkte Unterstützung der Generalisierung und Erkennung in nachgelagerten Verarbeitungsschritten. Für die folgenden Ausführungen muss an dieser Stelle erklärt werden, dass in der klassischen, neuronalen Sichtweise des Sparse Coding oftmals die Aktivierungen der Basisvektoren quasi als binär interpretiert werden. Ein Basisvektor kann also entweder aktiviert sein oder nicht. Diese Interpretation eignet sich besonders bei der Codierung von Bildern. Bei der Codierung von zeitlichen Signalen, wie den in dieser Arbeit behandelten Trajektorien spielt die tatsächliche Ausprägung der Aktivierung, also ihr eigentlicher Wert eine größere Rolle, da dieser die Skalierung des Basisvektors beschreibt. Dennoch sind die folgenden Betrachtungen auch hier prinzipiell gültig.

Sind die Aktivierungen binär, so kann man das Ereignis, dass ein Basisvektor aktiviert ist, mit einer Bernoulli-Verteilung beschreiben. In einem idealen spärlichen Code haben die Basisvektoren eine kleine Aktivierungswahrscheinlichkeit. Da das Ereignis, dass ein Basisvektor aktiviert ist, sehr selten ist, so führt Field aus, sollten nachgelagerte Klassifikatoren, die auf der Detektion von Features beruhen, unterstützt werden. Angenommen, eine bestimmte Klasse (z. B. eine Geste) zeichnet sich durch ein bestimmtes Feature (die Aktivierung eines bestimmten Basisvektors) aus. Nimmt man an, dass ein Basisvektor eine Aktivierungswahrscheinlichkeit p hat und es n Basisvektoren gibt, so ist die Wahrscheinlichkeit, dass nur genau ein Basisvektor aktiviert ist $P(\text{nur ein Basisvektor ist aktiv}) = (1-p)^n$. Je kleiner p, also je spärlicher der Code, desto größer ist die Wahrscheinlichkeit, dass nur dieser eine Basisvektor in einem gegebenen Eingangssignal vorhanden ist und aktiviert wird. Ein nachgelagertes Verfahren zur Klassifikation des Eingangssignals, welches auf Basis der Detektion von Features beruht, hat es damit einfacher, weil es nur auf das Vorhandensein einzelner Features prüfen muss.

Eines der Ziele einer effizienten Codierung ist die Komprimierung. Field [David J. Field, 1994] argumentiert, dass es unter sorgfältig gewählten Randbedingungen möglich ist, die spärliche Codierung eines Eingabedatums, z. B. eines Bildes, direkt zur Komprimierung zu verwenden. Techniken, wie die Lauflängenkodierung (englisch Run Length Encoding, kurz RLE) verwenden eben jenes Konzept. Field verweist weiterhin auf Arbeiten, welche die Komprimierung von Daten mit Hilfe spärlich besetzter Matrizen beschreiben [Evans, 1985; Jennings, 1991].

Einen konkreteren Nutzen führt Field in [David J. Field, 1987] mit der Erhöhung des Signal-Rausch-Verhältnisses an. Angenommen, ein Eingangssignal muss durch lineare Überlagerung weniger Basisvektoren beschrieben werden. So müssen die linearen Koeffizienten der wenigen beteiligten Basisvektoren einen wesentlich höheren Wert annehmen als die der unbeteiligten Basisvektoren, um die Signalenergie zu erhalten. Je kleiner die Menge der beteiligten Basisvektoren, desto höher ist ihre Aktivierung. Werden die Aktivierungen der Basisvektoren als transformiertes Signal betrachtet, und sind die Aktivierungen aller Basisvektoren grundsätzlich mit Rauschen behaftet, dann führt diese Erhöhung der Aktivierung zu einer Erhöhung des Signal-Rausch-Verhältnisses und damit zu einer Erleichterung für nachgelagerte Verarbeitungsschritte, z. B. die Klassifikation. Im konkreten Fall von Bewegungstrajektorien bedeutet Rauschen nicht nur das durch den Aufnahmeprozess entstehende Sensorrauschen, sondern auch die natürlichen Variationen bei der Erscheinung eigentlich gleicher Bewegungen, wie sie einerseits durch Unterschiede in der relativen Position von aufgenommener Person und Kamera, als auch durch die tatsächlichen Unterschiede in der Ausführung der Bewegungen entstehen. Deshalb ist gerade dieser Punkt von wesentlicher Bedeutung im Kontext dieser Arbeit.

2.5. Sparse Coding von Bewegungstrajektorien

Die oben vorgestellten klassischen Verfahren lösen das Problem der Gestenerkennung durch eine Repräsentation der Geste als Trajektorie in einem Raum statischer Konfigurationen und durch Behandlung des dynamischen Aspekts der Trajektorie durch einen relativ komplexes Verfahren zur Inferenz, welches diese Dynamik explizit modelliert. Dem entgegengesetzt wird beim Einsatz von Sparse Coding bereits ein großer Teil der Dynamik der Geste in der Repräsentation behandelt. Eine Geste wird als Folge von Teiltrajektorien (sogenannte Primitive) im Konfigurationsraum modelliert. Der Klassifikator modelliert nur noch die Folge der Primitive. Die Primitive modellieren also bereits die kurzfristige Dynamik der Geste, während der Klassifikator nur noch abstrakt die grobe Struktur der Geste abbildet.

Es gibt nur sehr wenige Arbeiten, welche Sparse Coding für die Gestenerkennung einsetzen. Erst in den letzten Jahren sind einige Arbeiten veröffentlicht worden, die ähnliche Ansätze verfolgen. Barthelemy et al. [Barthélemy et al., 2011, 2012] setzen Sparse Coding ein, um Bewegungsprimitive aus Handschrift zu lernen. Die Autoren schreiben, dass Sparse Coding im Sinne von Compressed Sensing [Foucart et al., 2013] genutzt werden kann, um für die Klassifikation relevante Features aus Bewegungsdaten zu extrahieren, welche an die Daten angepasst sind. Ähnlich wie in dieser Arbeit verwenden die Autoren ein iteratives Verfahren bestehend aus dem Wechsel zwischen Codierung und Lernen. Im Unterschied zu dieser Arbeit wird aber Matching Pursuit [Mallat et al., 1993] verwendet, um das Signal bei vorhandenem Wörterbuch zu codieren. Weiterhin wird das Optimierungsproblem durch Hinzufügen eines weiteren Freiheitsgrades neben der Verschiebungsinvarianz, eine 2D-Rotationsinvarianz erreicht. Damit sind die Basisvektoren also verschiebungs- und rotationsinvariant. Die Autoren können zeigen, dass es damit möglich ist, einen Datensatz, bestehend aus von einer Person handgeschriebenen Buchstaben mit sehr wenigen Basisvektoren zu codieren. Aus den Experimenten geht hervor, dass die Koeffizienten über verschiedene ähnliche Exemplare sehr ähnlich und stabil sind. Allerdings sind auch die Variationen innerhalb des Datensatzes sehr klein, da alle Buchstaben von der gleichen Person geschrieben wurden.

In [Barthélemy, Larue und Mars, 2012a,b, 2014] erweitern die Autoren den oben beschriebenen Ansatz auf Bewegungstrajektorien im 3D-Raum. Bei der Codierung wird im Matching-Pursuit für jeden Zeitschritt eine Singulärwertzerlegung verwendet, um die optimale Rotation einer Primitive zu ermitteln. Dadurch wird das Verfahren sehr rechenaufwändig. Die Autoren wenden das Verfahren auf Trajektorien einer französischen Zeichensprache an und können zeigen, dass mit sehr wenigen Primitiven sehr stabile Codierungen erreichbar sind (die Koeffizienten sind für gleiche Exemplare sehr ähnlich).

Die Autoren führen aus, dass für die Anwendung auf Bewegungstrajektorien auch die Behandlung unterschiedlicher Ausführungsgeschwindigkeiten wichtig ist und kündigen an, in darauf aufbauenden Arbeiten das Optimierungsproblem um einen Dilatationsparameter zu erweitern, womit die Primitive einen weiteren Freiheitsgrad zur Dehnung und Stauchung besitzen. Es ist allerdings fraglich, ob das Optimierungsproblem mit der

hohen Anzahl an Freiheitsgraden noch effizient lösbar ist.

2.6. Klassifikation auf Basis von Sparse Coding

In [Mayoue et al., 2012] wird das oben beschriebene Verfahren für den 2D-Fall im Rahmen einer Klassifikation angewendet. Die Autoren verwenden zunächst das Verfahren, um eine Handschrift-Datenbank, bestehend aus isolierten Buchstaben, die von einer einzigen Person geschrieben wurden, mit elf Basisvektoren zu codieren. Um zeitliche Variationen auszugleichen, werden zeitlich benachbarte Koeffizienten mittels Hamming-Fenstern zusammengefasst. Die so codierten Exemplare werden dann mittels eines Multi-Layer-Perzeptrons klassifiziert. Mit diesem Verfahren lässt sich eine Klassifikationsrate (Accuracy) von 97% erreichen. Die Autoren behaupten, damit die Leistung anderer State-of-the-Art-Verfahren, unter anderem HMM, auf dem gleichen Datensatz übertreffen zu können.

Klassifikation auf Basis von durch Sparse-Coding gelernten Features wird oft auch bei Audio-Signalen angewendet. [Grosse et al., 2007] beschreiben ein Verfahren zur Klassifikation von Musik-Genres. Sie beschreiben eine Lösung für das häufig bei dieser Art von Aufgaben auftretende Problem, dass nur eine kleine Menge gelabelter Daten für das Training zur Verfügung steht. Die Autoren beschreiben ein Framework namens „Self-taught Learning“ zur Behebung dieses Problems. Zunächst werden Features mittels Sparse Coding auf einer großen Menge ungelabelter Daten gelernt. Mit den gelernten Features werden die gelabelten Trainingsdaten codiert und dann klassifiziert. Laut Grosse et al., werden die Daten durch die Codierung durch bedeutungsvollere Features repräsentiert, welche von den Rohdaten abstrahieren. Die Autoren führen weiter aus, dass durch diesen Effekt Standardklassifikatoren wie SVM und Gaussian Discriminant Analysis (GDA) eine bessere Generalisierungsleistung erzielen, als wenn sie auf den Rohdaten angewendet würden. Die Autoren wenden die Standardklassifikatoren SVM und GDA an und formulieren zusätzlich ein weiteres Naive-Bayes-Modell, welches auf Exponentialverteilungen basiert. Zur Anwendung der Klassifikatoren wird zunächst ein Feature-Vektor berechnet, indem die Features über das codierte Trainingsbeispiel gemittelt werden.

Grosse et al. erzielen die besten Ergebnisse mit der SVM. Sie können zeigen, dass mit Hilfe der durch Sparse Coding gelernten Features eine wesentlich bessere Klassifikationsleistung erreicht werden kann, als bei Verwendung sonst üblicherweise für Audioklassifikation verwendeter Features.

2.7. Fazit

In diesem Kapitel wurden einige Publikationen mit direktem Bezug zu dem in der vorliegenden Arbeit entwickelten Verfahren vorgestellt. Da die Begriffe „Geste“ und „Gestenerkennung“ in der Literatur sehr unklar definiert sind, wurde eine Definition für

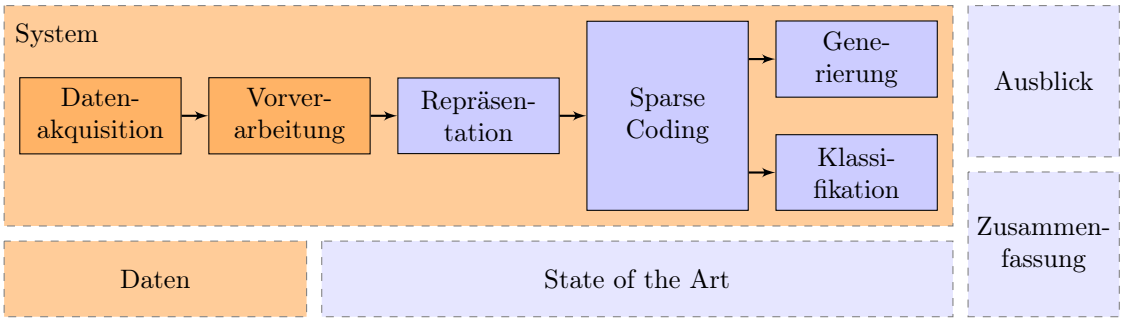
die Verwendung der Begriffe im Rahmen dieser Arbeit gegeben. Danach wurden unterschiedliche Repräsentationsformen von Gesten betrachtet und die in dieser Arbeit verwendete Repräsentationsform dargestellt.

Klassische Verfahren sind meist eine Kombination aus Repräsentation der Geste als Trajektorie in einem Konfigurationsraum und Klassifikation der Trajektorie. Die meisten Verfahren unterscheiden sich sehr in der Wahl des Konfigurationsraumes, verwenden zur Klassifikation aber meist das Hidden-Markov-Modell (HMM). In diesem Kapitel wurden die wichtigsten Vertreter der klassischen Modellierung mittels HMM vorgestellt.

Weiterhin wurden einige grundlegende Arbeiten aus dem Bereich des Sparse Coding vorgestellt. Es wurden theoretische und praktische Aspekte betrachtet, welche die Anwendung des Sparse Coding als Merkmalsextraktionsschritt in einem Mustererkennungsprozess in dieser Arbeit motivieren. Es wurden weiter einige jüngere Arbeiten vorgestellt, welche einen sehr ähnlichen Ansatz wie diese Arbeit verfolgen, deren konkrete Implementierung aber in einigen Teilen sehr von der Zielstellung dieser Arbeit abweicht.

Kapitel 3

Einsatzszenario und verwendete Datensätze



In diesem Kapitel wird der Einsatzrahmen des entwickelten Verfahrens beschrieben. Dabei wird zunächst das Rahmenszenario skizziert, welches als Motivation dieser Arbeit aus applikativer Sicht diente.

Anschließend wird die Einbettung des Verfahrens in den Mustererkunnungsprozess beschrieben. Es wird auf die im Rahmen dieser Arbeit implementierte Demonstratorapplikation eingegangen, welche den gesamten Mustererkennungsprozess realisiert. Dabei werden vor allem die der Merkmalsextraktion mittels Sparse Coding vorgelagerten Verarbeitungsschritte der Datenakquise und Vorverarbeitung erläutert.

Anschließend werden Datensätze vorgestellt, welche zur Bewertung des Verfahrens und in Experimenten genutzt wurden. Dies dient auch der Vermittlung eines Eindrucks der Art von Daten, die mit dem System modelliert werden sollen. Es wird zunächst ein Datensatz vorgestellt, der mittels des Datenakquiseschrittes des oben erwähnten Demonstrators aufgenommen wurde. Daneben werden einige öffentlich verfügbare Benchmark-Datensätze vorgestellt.

3.1. Rahmenszenario

Das ursprüngliche Anwendungsszenario des in dieser Arbeit entwickelten Verfahrens sollte eine Art Spiel zum kognitiven Training eines dementen Patienten sein. Dabei

sollte der Patient mit einem Roboter interagieren, der den Patienten bittet, bestimmte Formen in die Luft zu malen. Der Demonstrator sollte dann die ausgeführte Geste beobachten und ein Feedback über die Korrektheit und Qualität der Ausführung geben.

Aufgrund der zeitlichen Restriktionen des Projektes wurde dann aber eine reduzierte Variante dieses Anwendungsszenarios gewählt, bei dem der beobachtete Benutzer (z. B. ein Patient) einfache Formen aus einem festen Repertoire in die Luft malt, die vom Demonstrator dann erkannt und bezeichnet werden müssen.

Das Repertoire der Gesten wurde als die kleinen Buchstaben des lateinischen Alphabets definiert. Der Benutzer muss diese Buchstaben einzeln, also mit einer Pause zwischen den Buchstaben, bei der die Hand für eine kurze Dauer in Ruhelage verweilen muss, mit der rechten Hand in die Frontalebene vor seinem Körper zeichnen. Der Demonstrator verfolgt die rechte Hand und klassifiziert die ausgeführte Geste, sobald ihr Ende erkannt wurde. Die Bezeichnung der ermittelten Klasse wird grafisch auf dem Display des Roboters dargestellt und ggf. per Sprache ausgegeben.

Der im Rahmen dieser Arbeit entwickelte Demonstrator verwirklicht also die Kernfunktionalität eines Systems, welches erweitert werden kann, um in einer realen Interaktion mit einem Nutzer eingesetzt werden zu können. Dazu muss dann vor allem noch eine Dialogkomponente implementiert werden, welche die Nutzerinteraktion (z. B. Begrüßung, Erklärung, Feedback) steuert und mit der Kernkomponente zur Wahrnehmung und Klassifikation der ausgeführten Gesten in einer Client-Server-Beziehung kommuniziert.

Insgesamt bettet sich der Demonstrator als Teil einer Gesamtapplikation zur Unterstützung älterer und hilfsbedürftiger Menschen durch einen Servicerotober in die praktischen Arbeiten des FG Neuroinformatik und Kognitive Robotik (siehe z. B. Projekt SERROGA [Gross, Debes et al., 2014; Gross, Mueller et al., 2015]) ein. Dabei soll ein Serviceroboter, als ständiger Assistent, allein lebende und/oder an Altersdemenz leidende Personen mit der Durchführung verschiedener Aktivitäten fordern und unterstützen und so das Voranschreiten der geistigen Beeinträchtigung bremsen. Eine dieser Aktivitäten kann dabei die Durchführung des oben beschriebenen Spiels sein.

3.2. Demonstrator: Einbettung in den Mustererkennugsprozess

Im Rahmen dieser Arbeit wurde eine Demonstratorapplikation entwickelt, die den gesamten Mustererkennungsprozess implementiert. Bevor auf die konkrete Implementierung eingegangen wird, soll im Folgenden die typische Vorgehensweise bei der Bewegungserkennung dargestellt werden. Die Verarbeitungsschritte *Datenakquisition*, *Vorverarbeitung*, *Merkmalsextraktion* und *Klassifikation* stellen die Stufen eines typischen Mustererkennungsprozesses dar. Für jeden dieser Verarbeitungsschritte gibt es eine Vielzahl an alternativen Ansätzen. Ein konkretes System setzt sich dann aus einer Kombi-

nation der gewählten Ansätze in den jeweiligen Stufen zusammen.

In der *Datenakquisition* wird die Bewegung der Gliedmaßen mit Hilfe eines Sensors erfasst. Dies kann z. B. durch Anbringen von Beschleunigungssensoren, welche die Bewegungsdaten drahtlos an das Erkennungssystem übertragen, durch eine konventionelle Farbkamera und eine nachgeschaltete Bildanalyse zur Detektion der Gliedmaßen oder durch eine 3D-Wahrnehmung mittels einer Tiefenkamera und einer nachgelagerten Analyse der Tiefendaten erfolgen.

Die mit Hilfe der Datenakquisition gewonnenen Daten weisen typischerweise noch einige ungünstige Eigenschaften auf, die in einem *Vorverarbeitungsschritt* behandelt werden müssen, bevor sie der Merkmalsextraktion und dem Klassifikations- oder Prädiktionsschritt zugeführt werden. So sind die Daten typischerweise verrauscht, was mit klassischen Signalverarbeitungsalgorithmen behandelt werden kann. Des Weiteren weisen die Trajektorien der ausgeführten Bewegungen im Allgemeinen eine beliebige Translation und Rotation im Raum auf, die von der Position des beobachtenden Kamerasystems abhängig ist. Um die Wiedererkennung der ausgeführten Gesten zu ermöglichen, muss diese Translation und Rotatation entfernt werden und die Trajektorie damit in eine *invariante* Form überführt werden. Diese und weitere Vorverarbeitungsschritte werden auch in späteren Kapiteln beschrieben.

Nachdem die Daten vorverarbeitet worden sind werden sie einer *Merkmalsextraktion* unterzogen. Die Rohdaten weisen typischerweise Redundanzen[1] auf und können durch eine geeignete Transformation stark reduziert werden. Dabei werden bestimmte Charakteristika der Daten ausgenutzt und repräsentative Merkmale extrahiert. Die zu extrahierenden Merkmale können dabei von einem Designer definiert oder aus Daten gelernt werden. Das Lernen der Merkmale hat den Vorteil, das kein Designerwissen einfließen muss und das Verfahren auf andere Anwendungsbereiche generalisierbar ist. Der Hauptfokus dieser Arbeit liegt auf der Implementierung eines neuartigen Verfahrens für die Merkmalsextraktion. Es wird eine biologisch inspirierte Form des Lernens von Merkmalen für Bewegungstrajektorien entwickelt und Kapitel 4 gesondert vorgestellt.

Im *Klassifikationsschritt* wird der vorverarbeitete und gegebenenfalls mit Hilfe von Merkmalen reduzierte Datenstrom einem Klassifikator zugeführt, der den Daten die Identität einer Klasse zuordnet. Dabei wird davon ausgegangen, dass die Daten von vornherein bestimmten Klassen entstammen. Im Fall einer Gestenerkennung sind das die Identitäten der vom Benutzer beabsichtigten Gesten. Der Nutzer führt also eine bestimmte Bewegung aus und der Klassifikator soll den beobachteten Datenstrom einem der zuvor manuell definierten oder mit Hilfe von Daten trainierten Modelle zuordnen. Für den konkreten Algorithmus zur Klassifikation gibt es eine Vielzahl von möglichen Alternativen. Die Auswahl wird im Allgemeinen durch die Wahl der konkreten Verfahren in den vorgelagerten Schritten, insbesondere im Merkmalsextraktionsschritt, eingeschränkt. In dieser Arbeit werden einige dieser Alternativen, basierend auf dem bereits

[1] Als Redundanzen werden Abhängigkeiten in den Daten bezeichnet. Ist eine Variable von einer anderen abhängig, so kann sie eliminiert werden, ohne den Informationsgehalt der Daten zu reduzieren.

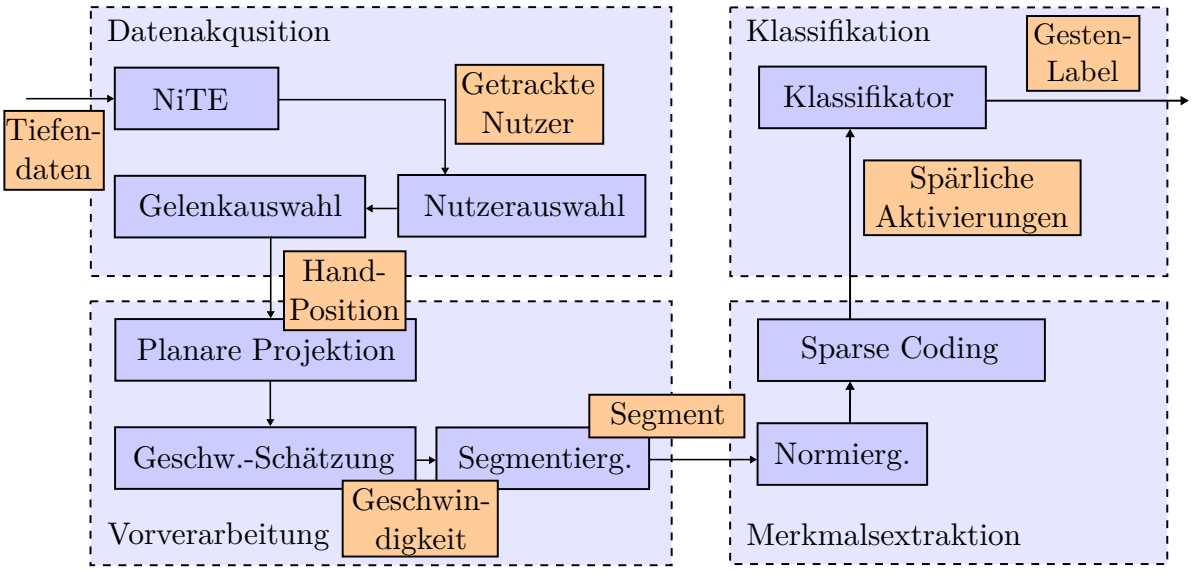


Abb. 3.1.: *Blockschaltbild der Komponenten im Demonstrator. Die blauen Rechtecke stellen die Verarbeitungsblöcke dar. Die gestrichelten, hellblauen Rechtecke stellen die Komponenten dar, die einen Verarbeitungsschritt des Mustererkennungsprozesses implementieren. Die Pfeile zeigen den Datenfluss zwischen den Blöcken. Soweit nicht trivial, sind die Pfeile mit orangen Rechtecken annotiert, welche die weitergegebenen Daten bezeichnen.*

erwähnten Merkmalsextraktionsverfahren, angewendet und ihre Eigenschaften untersucht.

In den folgenden Abschnitten wird die praktische Implementierung der Verarbeitungsstufen im Demonstrator beschrieben. Abbildung 3.1 zeigt ein Blockschaltbild der beteiligten Komponenten und ihre Gruppierung bezüglich der Schritte im Mustererkennungsprozess.

3.2.1. Datenakquise

Dieser Abschnitt beschreibt den Teilprozess der Datenakquise und damit die Transformation der visuellen Szene in ein zeitliches Signal, welches die Bewegung der Gliedmaßen repräsentiert.

Für die Aufnahme von Tiefenbildern und das Tracken der Gliedmaßen von Personen wurde die Tiefenkamera Microsoft Kinect in Verbindung mit Fremdsoftware in Form der freien Software-Bibliothek OpenNI [OpenNI 2015] und der geschlossenen Bibliothek NiTE [NiTE 2015] verwendet. Dies ist im Block „NiTE“ im Blockschaltbild (Abb. 3.1) realisiert. Die Funktionsweise der Personenverfolgung (Tiefenwahrnehmung und Skelett-Tracking) mittels Tiefenkamera ist in Anhang B näher beschrieben.

Die NiTE-Bibliothek kann bis zu 10 Personen im Sichtbereich simultan verfolgen und

liefert deren Skelette (Stick Figure). Da nur die Trajektorie der rechten Hand genau einer Person verfolgt werden soll, muss eine Person aus der Menge ausgewählt werden. Dazu wird die Fähigkeit der NiTE-Bibliothek genutzt, eine bestimmte Menge fest in der Bibliothek definierter Posen detektieren zu können. Es wird die Person ausgewählt, welche als letzte die „Psi"-Pose (die Person winkelt beide Arme nach oben an) ausgeführt hat. Dies ist im Block „Nutzerauswahl" realisiert (siehe Blockschaltbild in Abb. 3.1).

Der Block „Gelenkauswahl" wählt anschließend das Joint der rechten Hand der Person aus und gibt ausschließlich den Positionsvektor weiter.

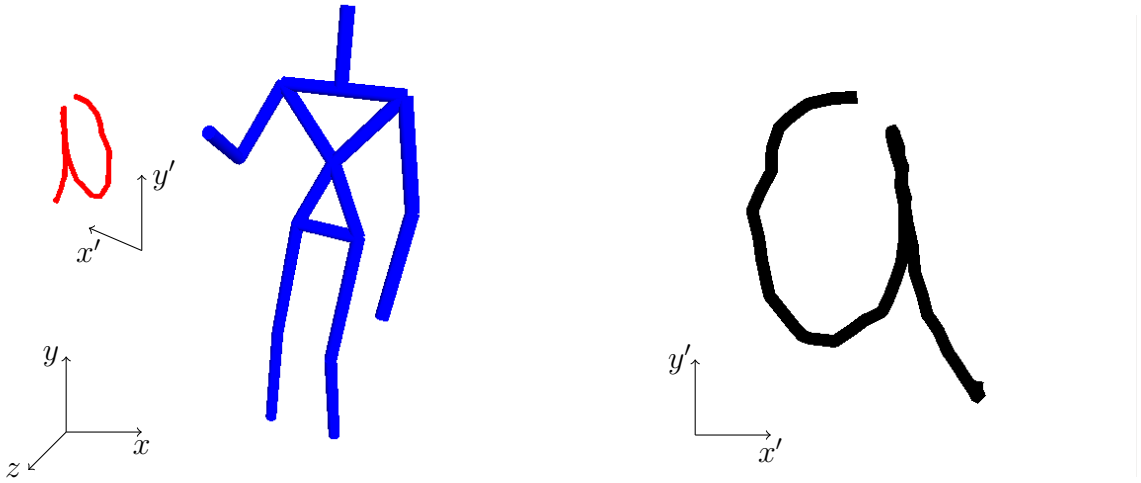
Transformation Die aus dem Skelett-Tracking gewonnene Trajektorie der Position der rechten Hand wird im Folgenden als Rohdatenstrom bezeichnet, da bis hierher noch keine weitere Aufbereitung und Vorverarbeitung vorgenommen wurde. Die Rohdaten weisen noch einige für nachgelagerte Verarbeitungsschritte ungünstige Eigenschaften auf, die durch eine geeignete Vorverarbeitung beseitigt oder zumindest abgemildert werden können. An dieser Stelle sollen vor allem die Rotation und Translation im Raum behandelt werden.

Abhängig von der relativen Position von Benutzer und Kamera weist die Trajektorie eine beliebige Rotation und Translation im Raum auf. Diese kann beseitigt werden, indem die Trajektorie in einem in der Person zentrierten Koordinatensystem betrachtet wird. Da die von der NiTE-Bibliothek generierten Gelenke durch ihre Orientierung bereits Koordinatensysteme definieren, bietet es sich an, eines der zentralen Gelenke als Ego-Koordinatensystem zu wählen. In dieser Arbeit wurde dazu das Torso-Gelenk (siehe Abbildung B.4) verwendet. Da die Personen bei der Ausführung von Gesten im Rahmen dieser Arbeit immer stehen, ist der Torso des getrackten Skeletts relativ steif und bietet somit ein relativ stabiles Koordinatensystem.

Da das Torso-Koordinatensystem bereits in Form einer Quaternion vorliegt, ist es nun relativ einfach, die Koordinaten und die Orientierung eines beliebigen Gelenks in Ego-Koordinaten zu transformieren. Da die Einzelheiten eine Einführung in die Mathematik mit Quaternionen erfordern, sei der Leser für eine detaillierte Beschreibung der Koordinatentransformation mittels Quaternionen auf [Vince, 2011] verwiesen.

Da die im Rahmen dieser Arbeit betrachteten Gesten (handgeschriebene Buchstaben des lateinischen Alphabets) so definiert wurden, dass sie sich im wesentlichen in der Frontalebene der Person abspielen (siehe Abschnitt 3.1), kann eine weitere Vereinfachung der Daten vorgenommen werden, indem die Trajektorie in die Frontalebene projiziert wird. Die Frontalebene wird dabei durch die x- und y-Achse des Torso-Koordinatensystems (siehe Abbildung B.4) definiert.

Damit ist die ursprünglich dreikanalige Trajektorie der rechten Hand im Koordinatensystem der Kamera in eine zweikanalige Trajektorie in der Frontalebene der Person transformiert worden. Abbildung 3.2 zeigt die Trajektorie beim Schreiben des Buchstabens „a" als Projektion auf die Frontalebene im Koordinatensystem der Kamera und im Koordinatensystem der Frontalebene. Im Folgenden werden weitere Vorverarbeitungs-



(a) (b)

Abb. 3.2.: *(a): Auf in die Frontalebene projizierter Verlauf der Position der rechten Hand beim Schreiben des Buchstabens „a“, dargestellt im Koordinatensystem der Kamera x-y-z. Die Frontalebene wurde zur besseren Visualisierung aus dem Torso-Gelenk heraus, 1,5m frontal vor die Person verschoben. (b): Darstellung des geschrieben Buchstabens im Koordinatensystem der Frontalebene x′-y′.*

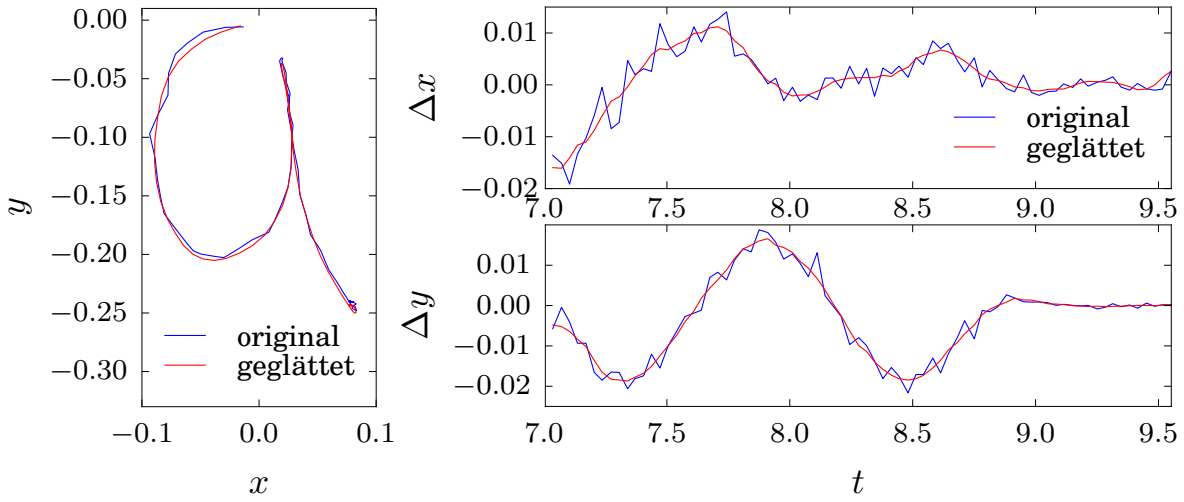
schritte beschrieben, die ausgeführt werden müssen, bevor die Merkmalsextraktion und die Klassifikation durchgeführt werden können.

Die Transformation in Ego-Koordinaten und die anschließende planare Projektion sind im Block „Planare Projektion“ (siehe Blockschaltbild in Abb. 3.1) implementiert.

3.2.2. Vorverarbeitung

In diesem Abschnitt werden die Schritte zur Vorverarbeitung des transformierten Signals beschrieben. Dazu zählen vor allem die Aufbereitung des verrauschten Signals durch Filterung und die Segmentierung der Gesten, um den nachgelagerten Verarbeitungsschritten isolierte Exemplare zur Verfügung stellen zu können.

Signalaufbereitung Durch Ungenauigkeiten im Tracking-System ist die Trajektorie auch nach der Transformation typischerweise etwas verrauscht. Des Weiteren wird für die Verarbeitung im Merkmalsextraktionschritt nicht die Folge der Positionen, sondern die der Geschwindigkeiten benötigt. Motivation dafür ist, dass durch die Verwendung der Geschwindigkeit das Signal invariant gegenüber der absoluten Position im Raum wird. Außerdem ist die Signalenergie bei Ruhelage des Gelenks gering und bei Bewegung hoch, was das Sparse Coding erheblich unterstützt (siehe dazu Abschnitt 4.5.1). Die Filterung und Schätzung der Geschwindigkeiten kann durch Einsatz eines Kalman-



(a) *Position* (b) *Geschwindigkeit*

Abb. 3.3.: *Verlauf von originaler Position und Geschwindigkeit beim Schreiben des Buchstabens „a“. Abbildung a zeigt den Verlauf der originalen Position und der durch Integration der geschätzten Geschwindigkeit berechneten und dadurch geglätteten Position. Abbildung b zeigt den Verlauf der durch Differenzbildung berechneten Geschwindigkeit und die mittels Kalman-Filter geschätzte Geschwindigkeit.*

Filters [Kalman, 1960] erreicht werden.

Der Kalman-Filter ist ein beliebtes Verfahren zur Filterung von Zeitreihen im Bereich der Robotik. Es handelt sich dabei um einen Algorithmus zur Schätzung des Zustands eines linear-gaussschen dynamischen Systems aus verrauschten Messungen. Er arbeitet rekursiv und berechnet die aktuelle Schätzung nur aus der aktuellen Messung und der Schätzung zum letzten Zeitpunkt.

Es wird im Folgenden angenommen, dass die hier behandelte Trajektorie durch einen linear-gaussschen Prozess beschreibbar ist. Die Zustandsvariablen sind dabei die Position und die Geschwindigkeit $\mathbf{x_t} = (x_t, y_t, \Delta x_t, \Delta y_t)^T$. Als Messungen dienen die (verrauschten) Positionen auf der beobachteten Trajektorie. Der Kalman-Filter schätzt daraus nun rekursiv die Zustandsvariablen, berechnet also die geschätzte (nicht verrauschte) Position und die geschätzte Geschwindigkeit zu jedem Zeitpunkt. Er liefert damit eine entrauschte Trajektorie (Verlauf der Variablen x_t und y_t) und den Verlauf der Geschwindigkeit (Verlauf der Variablen Δx_t und Δy_t).

Abbildung 3.3a zeigt den Verlauf der Position der rechten Hand beim Schreiben des Buchstaben „a“ in der Frontalebene. Es wird zum einen der Verlauf der originalen Koordinaten und zum anderen der durch Integration der geschätzten Geschwindigkeiten berechnete (also der geglättete) Verlauf gezeigt. Abbildung 3.3b zeigt den Verlauf der Geschwindigkeit. Es wird zum einen die durch Differenzbildung der Positionen berechnete Geschwindigkeit, als auch die mittels Kalman-Filter geschätzte und damit geglättete

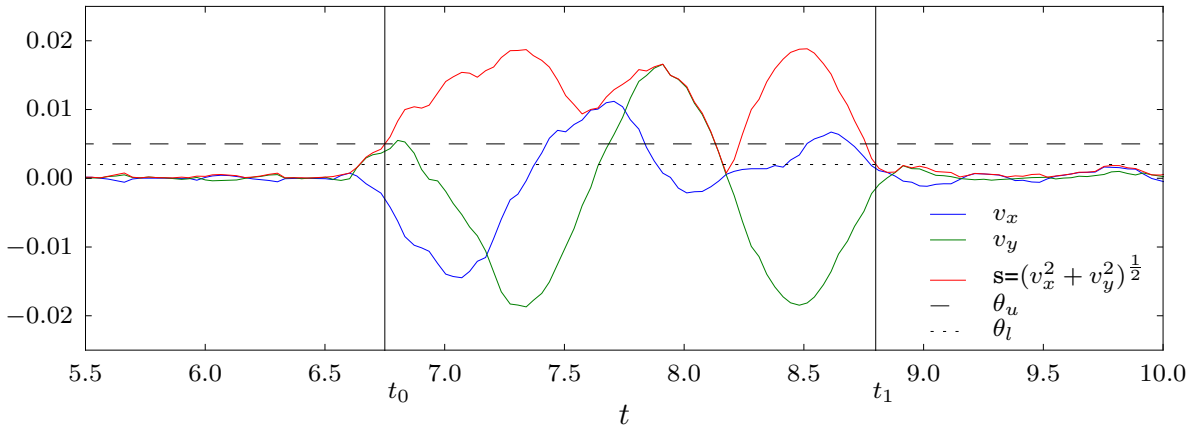


Abb. 3.4.: *Segmentierung einer Geste mittels Schwellen für die Geschwindigkeit und einer Hysterese. θ_u und θ_l definieren die untere und obere Schwelle der Hysterese für den Betrag der Geschwindigkeit s. Der Zeitpunkt des Überschreitens von θ_u definiert den Startpunkt t_0. Das Unterschreiten von θ_l definiert den Endpunkt t_1.*

Geschwindigkeit gezeigt.

Die Schätzung der Geschwindigkeit ist im Block „Geschwindigkeitsschätzung“ (siehe Blockschaltbild in Abb. 3.1) implementiert.

Segmentierung In Abschnitt 2.3 wurde beschrieben, dass bei der Klassifikation zwischen zwei Betriebsarten, der isolierten Klassifikation und der kontinuierlichen Klassifikation, unterschieden werden kann. Im hier beschriebenen Demonstrator wird die isolierte Klassifikation implementiert. Dazu muss der kontinuierliche Datenstrom, also konkret der Verlauf der Geschwindigkeit der rechten Hand, kontinuierlich auf Beginn und Ende einer gerade ausgeführten Geste hin analysiert werden.

Zum Finden des Start- und Endzeitpunktes einer Geste wird eine einfache Heuristik verwendet. Eine Geste beginnt immer mit einer Pause (Ruhelage der Hand) und endet mit einer Pause. Eine Pause ist dadurch gekennzeichnet, dass die Hand für eine definierte Dauer in Ruhelage verweilt. Ein Übergang von der Ruhelage zur Bewegung ist durch das Überschreiten einer definierten Geschwindigkeit der Hand gekennzeichnet. Entsprechend ist der Übergang von Bewegung in die Ruhelage durch ein Unterschreiten einer definierten Schwelle gekennzeichnet. Der Einsatz einer Hysterese verhindert einen ständigen Wechsel in der Nähe der Schwellen. Das Prinzip ist in Abbildung 3.4 dargestellt.

Wenn der Endzeitpunkt eines Segmentes festgestellt wurde, wird das Segment, also das Teilsignal, beginnend bei t_0 und endend bei t_1, der Merkmalsextraktion übergeben.

Die Segmentierung ist im gleichnamigen Block (siehe Blockschaltbild in Abb. 3.1) umgesetzt.

3.2.3. Merkmalsextraktion und Klassifikation

Die Schritte der Merkmalsextraktion und der Klassifikation werden als zentrale Inhalte dieser Arbeit in den Kapiteln 4 und 5 ausführlich beschrieben und werden deshalb hier nur kurz umrissen.

Das durch die Segmentierung erzeugte, isolierte Teilsignal wird nun im Merkmalsextraktionsschritt auf enthaltene charakteristische Merkmale hin untersucht. Dazu wird das Signal zunächst normiert. Dies ist durch den Block „Normierung“ (siehe Blockschaltbild in Abb. 3.1) realisiert. Der Normierungsfaktor wird vorab in der Lernphase ermittelt (siehe Abschnitt 3.2.4). Dann wird das Segment mittels Sparse Coding im Codierungsmodus, also mit fester, in der Lernphase ermittelter, Basis codiert (siehe dazu Abschnitt 4.3.1). Als Ergebnis liegt das Signal in spärlich codierter Form vor. Für die konkreten Details sei der Leser auf Kapitel 4 verwiesen. Im Blockschaltbild (Abb. 3.1) wird das Sparse Coding im gleichnamigen Block umgesetzt.

Nach der Merkmalsextraktion wird die Klassifikation durchgeführt (siehe dazu Abschnitt 5.2.2). Dazu wird zunächst das Segment des spärlich codierten Signals mittels Aggregation über die Zeit in einen Merkmalsvektor überführt, der dann mittels eines Naive-Bayes-Ansatzes klassifiziert wird. Konkret wird zur Aggregierung die Mittelwertbildung und zur Klassifikation der Gaussian-Naive-Bayes-Klassifikator (siehe Abschnitt 5.2.1) verwendet. Im Blockschaltbild (Abb. 3.1) wird dies durch den Block „Klassifikator“ implementiert.

3.2.4. Lern- und Kannphase

Einige der Module des Demonstrators werden durch bestimmte Parameter gesteuert, die vor Einsatz des Systems bestimmt werden müssen. Das Bestimmen der Parameter wird oft als Lernphase bezeichnet, während die Anwendung des Systems mit den bestimmten Parametern als Kannphase bezeichnet wird.

Die Parameter sind von den zu verarbeitenden Daten abhängig, und so muss für die Lernphase eine Menge an Trainingsdaten zur Verfügung stehen, die den in der Kannphase zu erwartenden Daten entsprechen.

Die Parameter der folgenden Module müssen bestimmt werden: Der Block „Geschwindigkeitsschätzung“ (siehe Abbildung 3.1) benötigt die Parameter des Kalman-Filters, welche manuell bestimmt werden können. Im Block „Normierung“ wird für jeden Kanal ein Normierungsfaktor benötigt, welcher aus den Trainingsdaten bestimmt werden kann. Im Block „Sparse Coding“ ist die Bestimmung der Basisvektoren und der Spärlichkeitsparameter notwendig. Die Basisvektoren werden aus den Trainingsdaten gelernt und die Spärlichkeitsparameter werden während des Lernens heuristisch bestimmt (siehe Abschnitt 4.5). Der Block „Geschwindigkeitssegmentierung“ benötigt als Parameter die Schwellen der Geschwindigkeit für Beginn und Ende des Segmentes (siehe Abschnitt 3.2.2). Diese müssen manuell festgelegt werden. Des Weiteren muss der Block „Klassifikator“ mit gelabelten Segmenten trainiert werden.

Die in der Lernphase ermittelten Parameter werden als Parametersatz persistent gespeichert und stehen den Units in der Kannphase zur Verfügung.

3.2.5. Integration mittels MIRA

Um die oben dargestellten Verarbeitungsschritte realwelttauglich auf einem mobilen Robotersystem zu implementieren, wurde die Robotik-Middleware MIRA verwendet [Einhorn et al., 2012].

MIRA ist ein in Kooperation zwischen der TU Ilmenau und der Metralabs GmbH entwickeltes Framework zur Implementierung verteilter Systeme. Es stellt dabei grundlegende Funktionalitäten zur effizienten Kommunikation zwischen verteilten Software-Modulen bereit und bietet Möglichkeiten zur Visualisierung der zwischen den Modulen kommunizierten Daten. MIRA stellt die beiden Konzepte der „Unit“ und des „Channels“ bereit. Eine Unit abstrahiert einen datenverarbeitenden oder -produzierenden Prozess und ein Channel abstrahiert die Datenkommunikation zwischen zwei Units.

Der modulare Aufbau des vorgestellten Mustererkennungsprozesses eignet sich gut für die Implementierung mittels Units und Channels, da jeder Verarbeitungsschritt in einer oder mehreren Units implementiert werden kann und die in einem Schritt produzierten Daten mittels eines Channels zum konsumierenden Schritt transportiert werden können.

Zur Visualisierung der auf den Channels publizierten Daten wird das im MIRA enthaltene Tool *miracenter* verwendet. Im miracenter können sämtliche aktive Units gesteuert werden und die publizierten Daten mittels verschiedener Visualisierungsmethoden dargestellt werden. Abbildung B.5 zeigt einen Screenshot des miracenter während des Einsatzes.

3.3. Verwendete Datensätze

Um dem Leser einen Eindruck der in dieser Arbeit behandelten Daten zu vermitteln, sollen in diesem Abschnitt die verwendeten Datensätze vorgestellt werden. Die Experimente in dieser Arbeit werden mit drei Arten von Datensätzen durchgeführt: ein selbst erstellter Datensatz, welcher aus Trajektorien der rechten Hand bei der Ausführung von Gesten besteht (und mit Hilfe des in Abschnitt 3.2 vorgestellten Demonstrators gewonnen wurde), ein frei verfügbarer Datensatz, bestehend aus Handschrift-Trajektorien und eine Gruppe von Datensätzen aus der Activity Recognition.

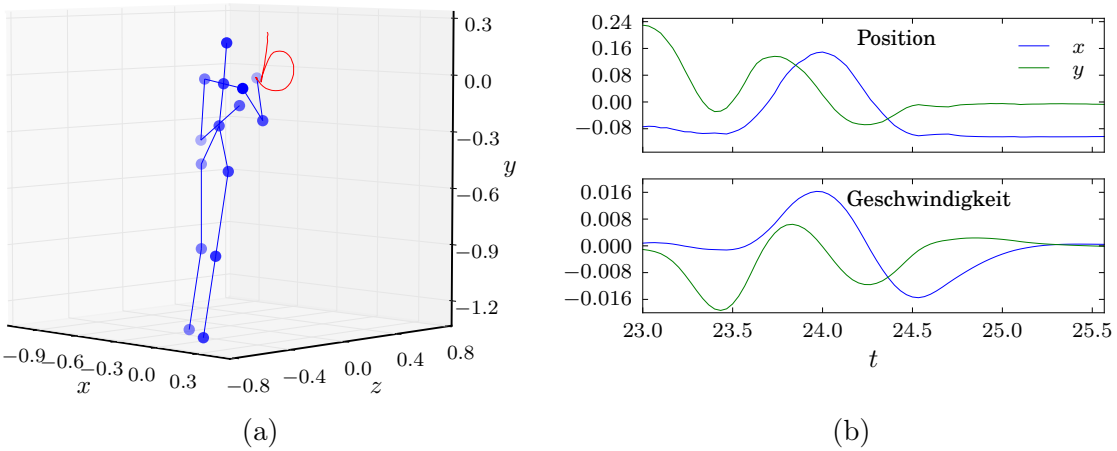
Die drei Arten von Datensätzen weisen dabei einen unterschiedlichen Grad der Komplexität bezüglich der vorkommenden Inter- und Intraklassen-Variation auf und eignen sich daher auch unterschiedlich gut zur Verdeutlichung bestimmter Eigenschaften der in den Kapiteln 4 und 5 vorgestellten Verfahren.

Die Datensätze werden in den folgenden Abschnitten vorgestellt. Es wird dabei jeweils darauf eingegangen, in welchen Kapiteln und warum sie dort verwendet werden.

3.3.1. Gesten-Trajektorien

Um die Eigenschaften der in dieser Arbeit untersuchten Verfahren in einem Szenario zu untersuchen, welches dem in Abschnitt 3.1 vorgestellten Rahmenszenario entspricht, wurde ein eigener Datensatz erstellt. Dieser Datensatz wurde mit der im Rahmen dieser Arbeit entwickelten Demonstrator-Applikation aufgezeichnet, welche in Abschnitt 3.2 vorgestellt wurde. Eine Darstellung der Daten findet sich in Anhang A.2. Im Folgenden wird der Datensatz mit GT (für „Gesture Trajectories“) bezeichnet.

Die Daten bestehen aus den Geschwindigkeitsverläufen der rechten Hand bei Ausführung von Gesten. Die Gesten bestehen dabei aus „in die Luft geschriebenen“ handschriftlichen kleinen Buchstaben. Die Bewegungen wurden von 15 Personen ausgeführt und mittels eines Verfahrens zur Verfolgung der Gliedmaßen mit einer Abtastrate von 30Hz erfasst. Pro Person wurden jeweils zehn Aufnahmen gemacht. Bei jeder Aufnahme wurde das gesamte Alphabet hintereinander weg geschrieben. Die Bewegungen finden dabei wesentlich in der Frontalebene der Person statt, die durch ein zweidimensionales Koordinatensystem mit x- und y-Richtung beschrieben wird. Abbildung 3.5 zeigt beispielhaft einen kurzen Abschnitt des Aufnahmeprozesses.



(a) (b)

Abb. 3.5.: *Aufnahme des Eingangssignals. (a) zeigt eine Person, die den Buchstaben „b“ in die Luft schreibt. (b) zeigt die Verläufe von Position und Geschwindigkeit in der Frontalebene.*

3.3.2. Handschrift-Trajektorien

In [Williams et al., 2006] wird ein Datensatz vorgestellt, welcher mit Hilfe eines Grafiktablets gewonnen und im UCI Machine Learning Repository [Asuncion et al., 2007] als „Character Trajectories Data Set“ (im Folgenden mit CTDS bezeichnet) veröffentlicht wurde. Der Datensatz besteht aus 2858 Aufnahmen einzelner Buchstaben des lateini-

schen Alphabets, die von einer Person auf das Grafiktablet geschrieben wurden. Eine Aufnahme besteht aus einer Trajektorie von Punkten (x,y,p), wobei x und y die Koordinaten auf dem Tablet bezeichnen und p den Druck des Stiftes auf das Tablet. Das Tablet zeichnet die Daten mit einer Frequenz von 200Hz auf. Nach der Aufnahme wurden die Daten numerisch differenziert, geglättet und normiert. Desweiteren wurden die einzelnen Aufnahmen jeder Klasse zeitlich so verschoben, dass die Abweichung vom mittleren Geschwindigkeitsprofil minimiert wird. Eine Veranschaulichung der Daten findet sich in Anhang A.1.

Dieser Datensatz wird vor allem im Kapitel 5.1 für Experimente zur Generierung neuer, klassentypischer Trajektorien auf Basis von Sparse Coding verwendet. Da alle Buchstaben von einer Person geschrieben sind, sind die Variationen innerhalb einer Klasse gering. Der Datensatz eignet sich daher gut für die Extraktion von Primitiven und für grundlegende Untersuchungen und Veranschaulichungen zur Anwendung von Sparse Coding auf Bewegungstrajektorien. Die extrahierten Primitive sind allerdings personenspezifisch, womit keine Aussagen über die Generalisierbarkeit auf mehrere Personen, also Szenarien mit größerer Intraklassenvarianz, möglich sind.

3.3.3. Daten aus der Activity Recognition

In Abschnitt 5.2.3 wird das in dieser Arbeit entwickelte Verfahren im Szenario der klassischen Activity Recognition für den Einsatz in der kontinuierlichen Klassifikation (also Klassifikation eines Sliding-Window in einem kontinuierlichen Datenstrom) evaluiert. Dazu wird das Verfahren auf entsprechende Datensätze angewendet. Als Grundlage wird dazu die Arbeit von Plötz et al. [Plötz et al., 2011] herangezogen, welche Verfahren zur Merkmalsextraktion auf vier frei verfügbare Datensätze aus dem Bereich der Activity Recognition anwendet und evaluiert. Die Datensätze werden im Folgenden kurz vorgestellt. Weitere Details finden sich in [Plötz et al., 2011].

Ambient Kitchen 1.0 Der erste Datensatz „Ambient Kitchen 1.0“ (AK) entstammt einer Arbeit von Pham et. al [Pham et al., 2009]. Er besteht aus Aktivitäten der Nahrungszubereitung, wie z. B. Zubereiten eines Sandwiches, Zubereitung von Salat, Schneiden von Gemüse etc. Insgesamt werden zehn für die Nahrungszubereitung typische Aktivitäten ausgeführt. Die Bewegungen werden mit vier Küchenutensilien (siehe Abbildung A.3), drei Arten von Messern und einem Löffel, ausgeführt, die jeweils mit einem tri-axialen Beschleunigungssensor ausgestattet wurden. Der Datensatz besteht aus insgesamt vier Stunden an Aufnahmen von 20 Personen. Die Daten wurden mit einer Sampling-Frequenz von 40Hz aufgenommen. In Anhang A.3.1 sind die Utensilien und ein Ausschnitt aus den Daten dargestellt.

Darmstadt Daily Routines Der zweite Vergleichsdatensatz wurde von Stikic et al. [Stikic et al., 2008] unter dem Namen „Darmstadt Daily Routines“ (DA) veröffentlicht.

Er besteht aus Aufnahmen von 35 Alltagsaktivitäten, z. B. Zähneputzen, Tischdecken etc. (siehe Abbildung A.4). Die Bewegungen wurden mittels zwei tri-axialen Beschleunigungssensoren aufgezeichnet. Einer der Sensoren wurde am Armgelenk getragen, ein anderer in der Hosentasche. Die Daten wurden bereits vorverarbeitet und mit einer Sampling-Frequenz von 2,5Hz veröffentlicht. In [Plötz et al., 2011] wurde beobachtet, dass die Daten des Sensors am Handgelenk ausreichen, um die Aktivitäten hinreichend gut klassifizieren zu können. In dieser Arbeit werden deshalb ebenfalls nur die Daten dieses einen Sensors verwendet. Ein Ausschnitt der Daten ist in Anhang A.3.2 dargestellt.

Skoda Mini Checkpoint Der dritte Datensatz wurde in [Zappi et al., 2008] unter dem Namen „Skoda Mini Checkpoint“ (SK) vorgestellt. Er besteht aus Aktivitäten eines Arbeiters während der Qualitätskontrolle in der Fertigung von Kraftfahrzeugen (siehe Abbildung A.5). Der Arbeiter trägt eine Reihe von Beschleunigungssensoren an Armen und Beinen. Der Arbeiter führt zehn Aktivitäten aus. Insgesamt wurden Aufnahmen mit einer Länge von drei Stunden erzeugt. Die Sampling-Frequenz beträgt 96Hz. In [Plötz et al., 2011] wurde beobachtet, dass die Daten eines Sensors am rechten Arm bereits ausreichen, um die Daten hinreichend gut klassifizieren zu können. Um die Vergleichbarkeit zu sichern, werden in dieser Arbeit ebenfalls nur die Daten dieses einen Sensors verwendet. Ein Ausschnitt der Daten ist in Anhang A.3.3 dargestellt.

Opportunity Gesture Challenge Roggen et al. stellen in [Roggen et al., 2010] einen Datensatz vor, der im Rahmen der „Opportunity Gesture Challenge“ (OC) aufgenommen wurde. Der Datensatz besteht aus Alltagsaktivitäten in einer künstlichen Küche. Die Personen tragen mehrere tri-axiale Beschleunigungssensoren. Außerdem wurden einige Gegenstände mit Beschleunigungssensoren ausgestattet. Es wurden mehrere Personen an verschiedenen Tagen während der Ausführung von Aktivitäten unterschiedlicher Komplexität aufgenommen. In [Plötz et al., 2011] wurde eine Teilmenge von zehn Aktivitäten geringer Komplexität berücksichtigt. Um die Vergleichbarkeit der Ergebnisse sicher zu stellen wird in dieser Arbeit ebenso vorgegangen. Ein Ausschnitt der Daten ist in Anhang A.3.4 dargestellt.

3.4. Fazit

In diesem Kapitel wurden das Zielszenario, die praktische Implementierung des Mustererkennungsprozesses und die drei Arten der in dieser Arbeit verwendeten Datensätze vorgestellt.

Der Demonstrator stellt eine echtzeitfähige Implementierung des gesamten Mustererkennungsprozesses dar. Der durch die NiTE-Bibliothek erzeugte Datenstrom des getrackten Skeletts wird mittels einfacher Transformationen auf die Trajektorie der rechten Hand reduziert. Weiter wird durch Projektion auf die Frontalebene und Schätzung

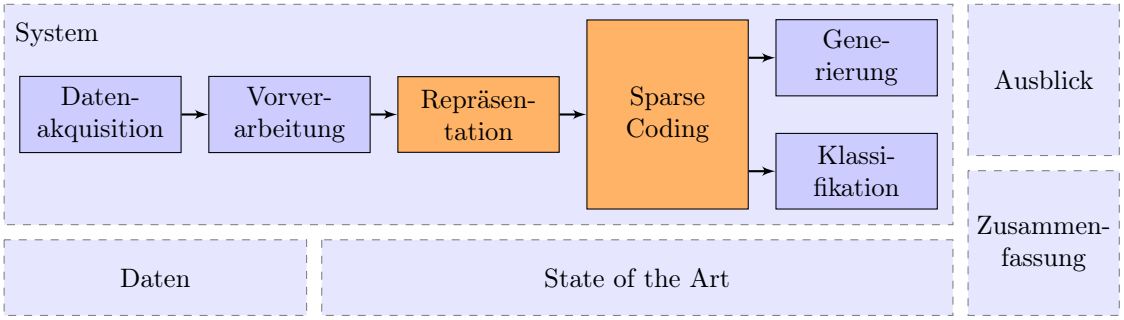
der Geschwindigkeit ein zweidimensionaler Verlauf der Geschwindigkeiten erzeugt, welcher in der Merkmalsextraktion mittels Sparse Coding in eine spärlich codierte Form gebracht und letztendlich im Klassifikationsschritt mittels Gaussian-Naive-Bayes klassifiziert wird.

Durch die Implementierung aller Schritte auf Basis der Robotik-Middleware MIRA konnte ein echtzeitfähiges Gesamtsystem geschaffen werden, welches realwelttauglich ist und einfache Gesten klassifizieren kann. Dennoch stellt der Demonstrator nur eine Kernkomponente dar, die für den praktischen Einsatz noch um eine Komponente zur Realisierung eines Dialogs mit dem Benutzer erweitert werden muss. Dies war nicht mehr Gegenstand der Arbeit.

Der Datensatz CTDS (siehe Abschnitt 3.3.2) besteht aus Handschrift-Trajektorien und wird im weiteren Verlauf der Arbeit für die Untersuchung und Darstellung der grundlegenden Eigenschaften der Anwendung von Sparse Coding auf Bewegungstrajektorien verwendet. Der Datensatz GT (siehe Abschnitt 3.3.1) besteht aus Trajektorien der rechten Hand bei der Ausführung einfacher Gesten. Die Gesten sind dabei ebenfalls Buchstaben, die „in die Luft“ gemalt werden. Der Datensatz ist prinzipiell ähnlich zu dem Datensatz CTDS, allerdings gibt es hier größere Intra-Klassenvarianzen, da Aufnahmen von mehreren Personen gemacht wurden, während beim CTDS nur Aufnahmen einer Person enthalten sind. Die Datensätze AK, DA, SK und OC (siehe Abschnitt 3.3.3) entstammen der Activity Recognition. Während CTDS und GT aus isolierten Trajektorien bestehen, enthalten diese Datensätze kontinuierliche Datenströme. Sie eignen sich daher zur Untersuchung der Eignung des Verfahrens für die kontinuierliche Klassifikation.

Kapitel 4

Sparse Coding für Bewegungstrajektorien



In diesem Kapitel wird das Kernthema der Arbeit, die Merkmalsextraktion mittels Sparse Coding, beschrieben. Das Ziel ist es, zu zeigen, dass Sparse Coding genutzt werden kann, um ein Alphabet an Basisbewegungen aus einer gegebenen Menge an Bewegungsdaten zu lernen. Weiterhin soll gezeigt werden, dass eine einzelne Trajektorie als Folge der Basisbewegungen dargestellt werden kann, was zu einer erheblichen Reduktion der Komplexität der Daten für nachfolgende Verarbeitungsschritte führt.

Zunächst wird ganz abstrakt die grundlegende Idee des Sparse Coding betrachtet. Danach wird ein Überblick über das Verfahren gegeben, sodass weniger an mathematischen Details interessierte Leser die folgenden Abschnitte zur genauen Beschreibung des Verfahrens überspringen können.

Das Verfahren besteht aus einem Optimierungsproblem, welches zunächst beginnend bei einem einfachen Grundproblem bis zum vollständigen, in dieser Arbeit verwendeten Optimierungsproblem schrittweise aufgebaut und erläutert wird. Danach wird der in dieser Arbeit verwendete Algorithmus zur Lösung des Problems vorgestellt. Dabei wird schrittweise vorgegangen, und jeweils die Auswirkungen der Erweiterungen in Experimenten an künstlichen sowie realen Daten gezeigt.

Nach der detaillierten Beschreibung des Verfahrens werden Ergebnisse an realen Daten illustriert und Fragen im praktischen Einsatz, wie z. B. das Vorgehen bei der Parameterwahl, diskutiert.

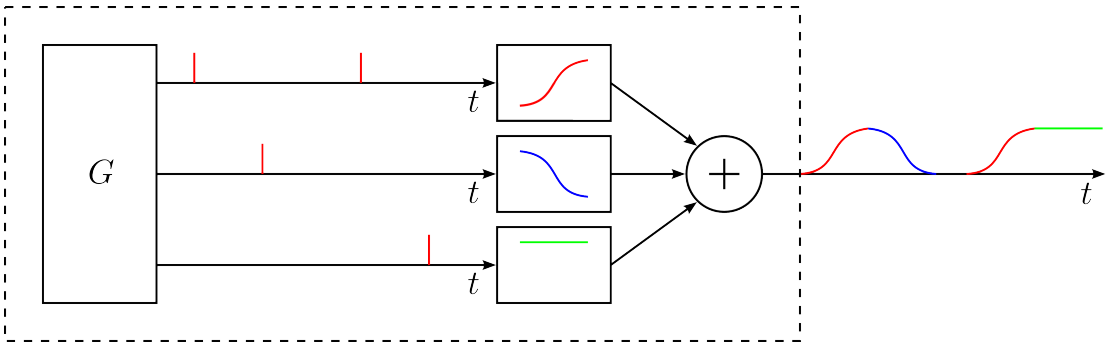


Abb. 4.1.: *Generatives Signalmodell des Sparse Coding. Die Signalquelle G erzeugt ein Signal, bestehend aus diskreten Ereignissen, welche die Primitive aktivieren, die dann linear überlagert werden und das Ausgangssignal ergeben. Dabei sind die Signalquelle, das diskrete Signal und die Primitive als „versteckte" Elemente zu verstehen (gestrichelter Kasten). Von außen sichtbar ist nur das kontinuierliche Ausgangssignal.*

Die Inhalte dieses Kapitels wurden im Wesentlichen in [Vollmer et al., 2012b], [Vollmer, Hellbach et al., 2014] und [Hellbach et al., 2011] veröffentlicht.

4.1. Grundlegende Idee und Zusammenfassung des Verfahrens

Während in Abschnitt 1.1 bereits einige allgemeine Vorteile des Sparse Coding für die Repräsentation von Daten genannt wurden, soll hier die grundlegende Idee des Verfahrens vorgestellt werden.

4.1.1. Generatives Signalmodell

Abbildung 4.1 zeigt ein grafisches Schema des Signalmodells, welches dem Sparse Coding für Zeitreihen zugrunde liegt. Eine Signalquelle G erzeugt eine Sequenz von diskreten Ereignissen auf einer Menge von Kanälen (hier sind drei Kanäle abgebildet). Durch diese Ereignisse werden die Primitive aktiviert. Die lineare Überlagerung der Primitive ergibt das kontinuierliche Ausgangssignal. Dabei werden die Signalquelle, das diskrete Signal und die Primitive als „versteckt", also unbekannt und nicht beobachtbar, betrachtet. Nur das kontinuierliche Ausgangssignal ist von außen sichtbar.

Dieses Signalmodell ist hilfreich, um die dem Signal zugrunde liegenden Strukturen zu bestimmen. Im Zusammenhang mit der Erkennung menschlicher Bewegungen kann man die Bestandteile des Modells auch etwas konkreter interpretieren. Die Signalquelle G kann als die Intention der beobachteten Person, eine bestimmte Teilbewegung auszuführen, aufgefasst werden. Die diskreten Ereignisse sind dann die Ausführungszeitpunkte der Teilbewegungen. Die Primitive sind dann die tatsächlichen Bewegungen,

die durch den Bewegungsapparat des Menschen erzeugt werden, und die kontinuierliche Überlagerung ist das durch das Erkennungssystem beobachtete Signal.

Die Annahme ist nun, dass das diskrete Signal, also die Aktivierungen bestimmter Teilbewegungen, eine explizitere und direktere Repräsentation der Intention der Person darstellt, weil es von den konkreten Details der Ausführung, also der konkreten Form der Primitive, befreit ist und für nachgelagerte Verarbeitungsschritte, wie eine Inferenz der Intention der Person durch einen Klassifikator, von Vorteil ist.

Ziel des in dieser Arbeit vorgestellten Verfahrens ist es nun, mit Hilfe von Sparse Coding das diskrete Signal und die Primitive aus einer Menge von Trainingsdaten mit Hilfe eines unüberwachten Lernverfahrens aufzudecken.

4.1.2. Lernen der Primitive als Merkmalsextraktion

Sparse Coding kann als Verfahren zur Merkmalsextraktion, ähnlich einer FFT oder der PCA, interpretiert werden. Während bei der FFT die Basisvektoren durch harmonische Funktionen vorgegeben sind, werden diese bei der PCA durch Minimierung des quadratischen Fehlers gelernt. Der im Folgenden erläuterte Ansatz des Sparse Coding verfolgt ein ähnliches Prinzip wie die PCA und optimiert ein Zielkriterium durch Adaption eines linearen Systems von Basisvektoren an die Eingangsdaten. Diese Optimierung umfasst neben der Minimierung des quadratischen Fehlers auch die Minimierung der Anzahl der aktiven Basisvektoren. Dabei gilt ein Basisvektor als aktiv, wenn der zugehörige lineare Koeffizient ungleich 0 ist, also der Basisvektor einen Anteil zu einem gegebenen Eingangsdatum beiträgt. Durch das Erzwingen einer kleinen Anzahl aktiver Basisvektoren drängt man den Optimierungsprozess dazu, ein gegebenes Eingangsdatum mit möglichst wenigen Basisvektoren zu beschreiben. Dies führt unter sorgfältig gewählten Randbedingungen dazu, dass sich die Basisvektoren so herausbilden, dass sie als Abschnitte oder Teile des Eingangsdatums interpretiert werden können. Da dem Optimierungsprozess von vornherein nur eine begrenzte Anzahl von Basisvektoren zur Verfügung steht, ist er weiterhin gezwungen, einzelne Basisvektoren für mehrere Eingangsdaten zu verwenden. Anders ausgedrückt, teilen sich mehrere Eingangsdaten einzelne Basisvektoren. Damit repräsentieren also die Basisvektoren nach der Optimierung Teile der jeweiligen Eingangsdaten und werden in vielen verschiedenen Eingangsdaten verwendet. Sie können so als elementare Bausteine betrachtet werden, die dem gesamten Datensatz zugrunde liegen. Jedes Eingangsdatum ist aus einer Teilmenge der Bausteine zusammengesetzt. Dies ist vergleichbar mit einer Menge von Wörtern einer Sprache, wobei die Buchstaben die Bausteine sind, die zu Wörtern zusammengesetzt werden können.

Hat man so die Bausteine eines Datensatzes gelernt, so können einzelne Eingabedaten als Komposition der Bausteine in einer abstrakteren Darstellung repräsentiert werden. Diese abstrahiert von den reinen Sensordaten auf eine Art symbolische Beschreibung, wobei die Bausteine als Symbole interpretiert werden können. Diese Komposition manifestiert sich in den linearen Koeffizienten, welche angeben, welchen Anteil ein Baustein am jeweiligen Eingabedatum hat, oder, anders ausgedrückt, wie stark ein Basisvektor

aktiviert ist. Die Aktivierungen der Basisvektoren werden dann in einem Feature-Vektor zusammengefasst.

Diese Idee des Findens von Basiselementen oder Grundbausteinen eines Datensatzes und die darauf folgende Darstellung der Eingangsdaten durch Komposition oder Aktivierung der Grundbausteine ist die Grundmotivation der Verwendung des Sparse Coding in dieser Arbeit.

4.1.3. Das Verfahren im Überblick

Bevor in den folgenden Abschnitten das Verfahren detailliert beschrieben wird, fasst dieser Abschnitt das Verfahren zusammen. Der weniger an Details interessierte Leser kann dann die Abschnitte 4.2, 4.3 und 4.4 überspringen und bei Abschnitt 4.5 fortfahren, wo Ergebnisse illustriert werden.

Das in dieser Arbeit verwendete Verfahren wird als Optimierungsproblem formuliert. Das Optimierungsproblem besteht in der Minimierung einer Zielfunktion, welche die Differenz zwischen den Trainingsdaten und einer Rekonstruktion der Trainingsdaten durch lineare Überlagerung (Aktivierung) der Basisvektoren misst. Zusätzlich wird ein Spärlichkeitsmaß zur Zielfunktion addiert. Dieses Spärlichkeitsmaß ist abhängig von den Aktivierungen und erhöht den Wert der Zielfunktion, wenn viele Aktivierungen einen Wert größer als null haben. Über einen Spärlichkeitsparameter kann der Grad der geforderten Spärlichkeit kontrolliert werden. Die folgende Gleichung zeigt eine zum Zweck der Übersichtlichkeit vereinfachte Form des Optimierungsproblems (das Optimierungsproblem ist in Abschnitt 4.2 genauer formuliert und beschrieben):

$$\min_{\mathbf{W},\mathbf{H}} \ \|\mathbf{V} - \mathbf{W}\mathbf{H}\|_2^2 + \lambda\|\mathbf{H}\|_1$$

Dabei ist $\mathbf{V}$ die Matrix der Eingangsdaten, $\mathbf{W}$ sind die Basisvektoren, $\mathbf{H}$ sind die Aktivierungen und λ ist der Spärlichkeitsparameter.

Optimiert man diese Zielfunktion nur über die Aktivierungen und betrachtet die Basis als fest, so wird das Problem als Codierungsproblem bezeichnet. Bei der Optimierung werden dann die Aktivierungen so variiert, dass bei einer bestimmten geforderten Spärlichkeit die Trainingsdaten durch Überlagerung der gegebenen Basisvektoren bestmöglich rekonstruiert werden. Die Basis muss dabei so gewählt werden, dass sie die Eigenschaften der Daten gut abbilden kann und ist damit stark vom Problem abhängig. Häufig werden eine Wavelet-Basis oder eine DCT-Basis gewählt.

Da eine gewählte Basis im Allgemeinen die Signaleigenschaften nicht optimal abbilden kann, ist es wünschenswert, eine an die Daten angepasste Basis zu finden. Hält man im oben genannten Optimierungsproblem nun die Aktivierungen fest und betrachtet die Basis als variabel, so kann durch Minimierung eine optimale Basis bei gegebenen Aktivierungen gelernt werden. Das Problem des Lernens der Basis wird auch als Dictionary-Learning bezeichnet.

Um eine Dekomposition der Trainingsdaten vorzunehmen, müssen sowohl Aktivierun-

gen, als auch Basis gelernt werden. Eine gleichzeitige Optimierung über beide Parameter stellt allerdings ein nicht-konvexes Optimierungsproblem dar, welches nicht effizient lösbar ist. Deshalb behilft man sich mit einer iterativen Lösungsstrategie und nähert sich der Lösung mittels Alternierung von Codierung und Dictionary-Learning. Dazu werden die Variablenwerte von Aktivierungen und Basis zunächst geeignet initialisiert. Im Codierungsschritt wird zunächst nur über die Aktivierungen optimiert. Im folgenden Dictionary-Learning-Schritt wird dann nur über die Basis optimiert. Nun folgt wieder ein Codierungsschritt usw. Dies wird so lange wiederholt, bis ein Konvergenzkriterium erreicht ist. Beide Teilschritte werden in dieser Arbeit durch einen Gradientenabstieg implementiert. In einer Iteration wird in jedem Teilschritt ein Schritt des Gradientenabstiegs durchgeführt.

Entscheidend für die Güte des Ergebnisses ist die Wahl des Spärlichkeitsparameters. Wird er zu hoch gewählt, also die Spärlichkeit überbetont, ist die Rekonstruktion schlecht und damit die Abbildung der Eingangsdaten nicht gegeben. Wird er zu niedrig gewählt, also die Rekonstruktion überbetont, sind die Aktivierungen nicht mehr spärlich und die Basisvektoren überlappen sich, wodurch sie nicht mehr als teilebasierte Dekomposition interpretiert werden können.

Durch das Spärlichkeitsmaß können bestimmte Forderungen an die konkrete Form der Spärlichkeit gestellt werden. Üblicherweise wird die L1-Norm über den Aktivierungen verwendet, was nachweisbar zu spärlichen Lösungen führt. In der praktischen Anwendung auf Bewegungsdaten zeigen sich allerdings Probleme. So wird die Idealvorstellung von zu diskreten Zeitpunkten spärlich aktivierten Primitiven nicht realisiert. Stattdessen zeigen Experimente, dass die Aktivierungen über lokale Zeitintervalle „verschmiert“ sind. Dieses Problem wurde in dieser Arbeit durch Erweiterung der Zielfunktion um einen zusätzlichen Spärlichkeitsterm, die sogenannte lokale Spärlichkeit, behandelt. Die lokale Spärlichkeit setzt lokal benachbarte Aktivierungen in Abhängigkeit zueinander und führt einen Wettbewerb benachbarter Aktivierungen ein. Dadurch lassen sich die Verschmierungen reduzieren und es entstehen lokal spärliche, isolierte Aktivierungen.

Im praktischen Einsatz wird in eine Lernphase und eine Kannphase unterschieden. Während der Lernphase werden die Primitive (Basisfunktionen) und Aktivierungen gelernt. In der Kannphase werden die zuvor gelernten Basisfunktionen als fest betrachtet und nur noch das Codierungsproblem optimiert, um die Aktivierungen und somit die Codierung des Signals zu erhalten. Damit ist in der Kannphase nur noch ein relativ einfaches Problem zu lösen und der Rechenaufwand im Echtzeiteinsatz begrenzt.

Abschnitte 4.2, 4.3 und 4.4 beschreiben das Optimierungsproblem und den Lösungsalgorithmus im Detail. Beginnend bei Abschnitt 4.5 werden konkrete Ergebnisse illustriert und der praktische Einsatz diskutiert.

4.2. Optimierungsprobleme

Im Folgenden wird zuerst die Grundformulierung des Optimierungsproblems hergeleitet. Danach wird mit der Nicht-negativen Matrixfaktorisierung (NMF) eine speziellere Formulierung dargestellt, die einige Vorteile hat und in dieser Arbeit als wesentlicher Kern des Verfahrens verwendet wird. Zuletzt wird eine Erweiterung der NMF um Translationsinvarianz beschrieben, die sich besonders in der Anwendung auf Zeitreihen als vorteilhaft erweist.

4.2.1. Das Grundproblem

Es sei eine Menge von N Eingangssignalen, z. B. eine Menge von Zeitreihen, gegeben. Dies können sowohl ganze Zeitreihen oder durch Fensterung erzeugte Teilstücke eines sehr langen Signals sein. Für die folgenden Betrachtungen ist es hilfreich, diese Signale als Vektoren zu betrachten. Hat man z. B. ein Signal $s(t)$ mit einer Anzahl P Samples mit einem Sample-Abstand von Δt abgetastet, so kann man daraus den Vektor $\mathbf{v} \in \mathbb{R}^P$ mit $\mathbf{v} = (v_1, \ldots, v_P)^T = (s(\Delta t), s(2\Delta t), \ldots, s(P\Delta t))^T$ formen. Eine Menge von N Eingangssignalen wird dann mit den Vektoren $\{\mathbf{v}_n \in \mathbb{R}^P,\, n = 1, \ldots, N\}$ beschrieben.

Es wird ein lineares Modell eingeführt, welches die Repräsentation der Daten durch lineare Überlagerung von K Basisvektoren $\{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$ mit $\mathbf{w}_k \in \mathbb{R}^P$ als Spalten einer Matrix $\mathbf{W} \in \mathbb{R}^{P \times K}$ und der zugehörigen Aktivierungen $\mathbf{h}_n \in \mathbb{R}^K$ beschreibt, sodass gilt:

$$\forall n : \mathbf{v}_n \approx \mathbf{W}\mathbf{h}_n \ .$$

Diese Beschreibung kann auch als generatives Modell verstanden werden, wie es schon weiter vorn in Abbildung 4.1 gezeigt wurde. Dabei beschreibt ein Aktivierungsvektor $\mathbf{h}_n$ das versteckte Signal und der Vektor $\mathbf{v}_n$ das entsprechende sichtbare Signal, welches durch lineare Transformation des versteckten Signals mit $\mathbf{W}$ entsteht. Abbildung 4.2 zeigt ein künstliches Beispiel mit drei Basisvektoren, zwei Eingangsvektoren und ihren zwei Aktivierungsvektoren.

Erste Formulierungen des Sparse Coding gingen zunächst von einer festen, vorgegebenen Basis $\mathbf{W}$ aus, die zumeist als Wavelet-Basis, wobei die Basisfunktionen Wavelets mit unterschiedlicher Frequenz und unterschiedlicher Verschiebung sind, oder in Form einer diskreten Cosinustransformation (DCT) gewählt wurde. Diese Basis wird häufig auch als „Dictionary“ (engl. Wörterbuch) bezeichnet. Bei einer festen Basis besteht das Problem des Sparse Coding darin, spärliche Vektoren $\{\mathbf{h}_n : n = 1, \ldots, N\}$ zu finden, sodass gilt

$$\mathbf{v}_n = \mathbf{W}\mathbf{h}_n + \epsilon = \mathbf{r}_n + \epsilon \ .$$

Der Vektor $\mathbf{r}_n = \mathbf{W}\mathbf{h}_\mathbf{n}$ beschreibt die sogenannte Rekonstruktion eines Eingangsdatums durch die entsprechenden Faktoren. In der Regel weicht die Rekonstruktion vom

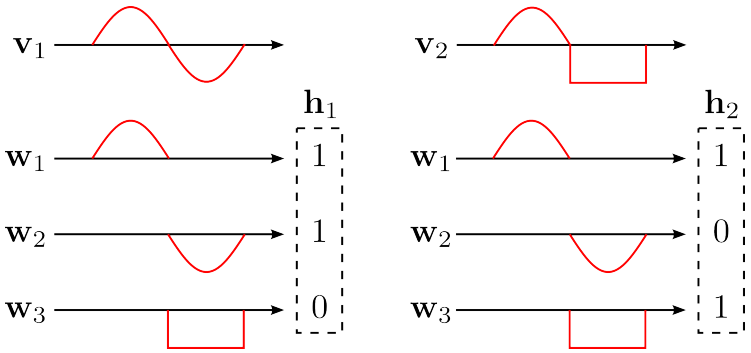


Abb. 4.2.: *Beispiel zweier Eingangssignale* $\mathbf{v}_1$ *und* $\mathbf{v}_2$ *und ihrer Repräsentation durch Aktivierungen* $\mathbf{h}_1$ *bzw.* $\mathbf{h}_2$ *dreier Basisvektoren* $\mathbf{w}_1$, $\mathbf{w}_2$ *und* $\mathbf{w}_3$. *Es wird zunächst angenommen, dass die Basisvektoren fest sind und von einem Designer gewählt wurden.*

Eingabedatum ab. Diese Abweichung wird durch den Rauschterm $\epsilon \sim \mathcal{N}(\mathbf{0}_P,\sigma\mathbf{I}_P)$ beschrieben. Dabei wird die vereinfachende Annahme getroffen, dass der Fehler einer Normalverteilung unterliegt.

Das Problem kann durch die Minimierung der folgenden Zielfunktion gelöst werden:

$$\forall n : \min_{\{\mathbf{h}_n\}} \frac{1}{2} \sum_n \|\mathbf{v}_n - \mathbf{W}\mathbf{h}_n\|_2^2, \text{ sodass } \sum_n \|\mathbf{h}_n\|_0 < c \ .$$

Die Nebenbedingung beschränkt die Anzahl der aktiven Basisvektoren. Dabei ist

$$\|\mathbf{h_n}\|_0 = |\{i : |h_{n,i}| > 0\}|$$

die l_0-Pseudonorm, welche die Einträge in einem Vektor zählt, die ungleich 0 sind. Die Basis $\mathbf{W}$ und Konstante c sind dabei Parameter des Optimierungsproblems. Dieses Problem wird auch als „Codierungsproblem“ bezeichnet.

Fasst man alle Eingabevektoren als Spaltenvektoren in einer Matrix $\mathbf{V} \in \mathbb{R}^{P\times N}$ und alle Vektoren $\mathbf{h}_n$ als Spalten einer Matrix $\mathbf{H} \in \mathbb{R}^{K\times N}$ zusammen, so kann das Optimierungsproblem auch in folgender Form beschrieben werden:

$$\min_{\mathbf{H}} \frac{1}{2}\|\mathbf{V} - \mathbf{W}\mathbf{H}\|_2^2, \text{ sodass } \|\mathbf{H}\|_0 < c \ .$$

Dabei beschreibt $\|\mathbf{H}\|_0 = |\{(i,j) : |H_{i,j}| > 0\}|$ die l_0-Pseudonorm auf $\mathbf{H}$ und zählt die Einträge, deren Wert ungleich 0 ist.

Durch die Verwendung der l_0-Pseudonorm ist das Problem in dieser Form allerdings NP-schwer. Hier behilft man sich einer Approximation der Form

$$\min_{\mathbf{H}} \frac{1}{2}\|\mathbf{V} - \mathbf{W}\mathbf{H}\|_2^2, \text{ sodass } \|\mathbf{H}\|_1 < c \ . \tag{4.1}$$

Dabei wird die Forderung nach Spärlichkeit nun durch die l_1-Norm umgesetzt: $\|\mathbf{H}\|_1 = \sum_{i,j} |H_{i,j}|$. Diese Nebenbedingung kann in Form linearer Beschränkungen dargestellt werden, womit die Lösungsmenge konvex wird und das Problem mit Verfahren aus dem Bereich der konvexen Optimierung behandelt werden kann.

Aus der Optimierungstheorie ist bekannt, dass ein Optimierungsproblem mit linearen Nebenbedingungen durch ein äquivalentes unbeschränktes Problem in folgender Weise dargestellt werden kann:

$$\min_{\mathbf{H}} \frac{1}{2}\|\mathbf{V} - \mathbf{W}\mathbf{H}\|_2^2 + \lambda\|\mathbf{H}\|_1 \ . \tag{4.2}$$

Dabei korrespondiert ein großer Wert von λ mit einem kleinen Wert für c. Dieses Problem ist in der Literatur als „Least Absolute Shrinkage and Selection Operator“ (LASSO) [Kukreja et al., 2005] bekannt. Es kann auch als regularisierte Minimierung eines quadratischen Fehlerterms interpretiert werden. In der klassischen Optimierung wird dabei statt der oben gezeigten Regularisierung mittels l_1-Norm auch häufig die Regularisierung mittels l_2-Norm gewählt, was auch als „Ridge-Regression“ bezeichnet wird. Es existiert eine Vielzahl an Literatur zur vergleichenden Gegenüberstellung der Effekte dieser beiden Regularisierungsterme. Dabei lässt sich zeigen, dass die Regularisierung mittels l_1-Norm spärliche Lösungen bevorzugt [Hastie et al., 2009].

4.2.2. Adaption der Basis

Bisher wurde von von einer festen Basis ausgegangen. Zur Wahl der Basis sind gewisse Annahmen über das zu analysierende Signal nötig. Dabei kann nicht immer garantiert werden, dass eine bestimmte Basis die Signaleigenschaften optimal abbilden kann. Eine Lösung für dieses Problem ist, die Basis an das Signal zu adaptieren. Dazu wird das Optimierungsproblem erweitert, indem auch über die Basis als Parameter der Zielfunktion optimiert wird. Dies führt zur Definition des vollständigen Optimierungsproblems des Sparse Coding:

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{V} - \mathbf{W}\mathbf{H}\|_2^2 + \lambda\|\mathbf{H}\|_1 \\ &\text{sodass } \forall k : \|\mathbf{w}_k\|_2 = 1 \end{aligned} \tag{4.3}$$

Die Nebenbedingung sichert, dass keine degenerierten Lösungen erzeugt werden. Ohne die Nebenbedingung könnte ein lokales Minimum auch durch Minimierung der Aktivierungen und eine Kompensation durch Hochskalieren der Basisvektoren erreicht werden.

Diese Formulierung entspricht der Zielfunktion des biologisch motivierten Sparse Coding für natürliche Bilder, wie sie in Abschnitt 2.5 dargestellt wurde. Dieses Problem kann auch als Zerlegung der Matrix $\mathbf{V}$ in zwei Faktormatrizen $\mathbf{W}$ und $\mathbf{H}$ unter Spärlichkeitsbedingungen interpretiert werden und wird daher auch als spärliche Matrixfaktorisierung bezeichnet.

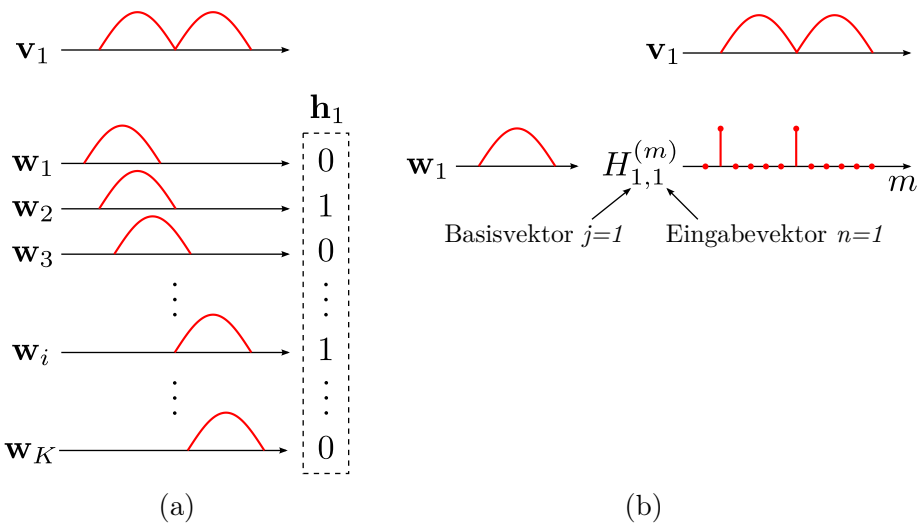


Abb. 4.3.: *Realisierung von Verschiebungsinvarianz (a) durch Erweiterung der Basis um alle möglichen Verschiebungen eines Basisvektors und (b) durch Transformation eines Basisvektors und vorhalten von Aktivierungen für alle möglichen Verschiebungen.*

Für diese Form des Optimierungsproblems existieren einige effiziente Lösungsverfahren. Bevor in Abschnitt 4.3 auf konkrete Algorithmen zur Lösung dieses Problems eingegangen wird, sollen an dieser Stelle zwei Erweiterungen des Grundproblems vorgestellt werden, die im Rahmen dieser Arbeit von wesentlicher Bedeutung sind.

4.2.3. Transformationsinvariantes Sparse Coding

Um zeitlich (oder, im Fall von Bildern, räumlich) verschiebbare Muster abbilden zu können, braucht man für jede Form eines Basisvektors Varianten in allen möglichen Verschiebungen. Im Falle eines reinen Codierungsproblems ist es üblich, die Matrix $\mathbf{W}$ um alle möglichen Verschiebungen zu erweitern. Abbildung 4.3a zeigt dieses Prinzip schematisch.

Zum Lernen der Matrix $\mathbf{W}$ würde dieser Ansatz allerdings möglicherweise zur Herausbildung unterschiedlicher Basisvektoren führen, da keine Notation für die eigentliche Identität der verschobenen Varianten eines Basisvektors existiert. Mit einem Trick kann diese Verbindung allerdings explizit gemacht werden. Dazu führt man eine Menge $\mathcal{T} = \{\mathbf{T}^{(m)} : m \in \mathcal{M}\}$ abstrakter Transformationsoperatoren ein, mit denen die Basisvektoren transformiert werden und so für jeden ursprünglichen Basisvektor implizit eine Familie transformierter Basisvektoren entsteht. Zusätzlich müssen nun Aktivierungen für jede mögliche Transformation der Basisvektoren gehalten werden. Dies wird durch einen Tensor dritter Ordnung $\mathcal{H}$ repräsentiert, dessen Scheiben $\mathbf{H}^{(m)}$ für $m \in \mathcal{M}$ die Aktivierungen für die m-te Transformation aller Basisvektoren halten. Ein Eingangssignal

wird dann als lineare Überlagerung aller transformierten Basisvektoren rekonstruiert:

$$\mathbf{v}_n \approx \sum_m \mathbf{T}^{(m)}\mathbf{W}H^{(m)}_{:,n}m$$

wobei $\mathbf{T}^{(m)}\mathbf{W}$ die m-te Transformation aller Basisvektoren ist. Dieses Konzept ist für die Realisierung der Verschiebungsinvarianz in Abbildung 4.3b dargestellt.

Damit ergibt sich folgende Form des Optimierungsproblems:

$$\min_{\mathbf{W},\mathcal{H}} \frac{1}{2}\|\mathbf{V} - \sum_m \mathbf{T}^{(m)}\mathbf{W}\mathbf{H}^{(m)}\|_2^2 + \lambda \sum_m \|\mathbf{H}^{(m)}\|_1$$
$$\text{sodass } \forall k : \|\mathbf{w}_k\|_2 = 1 \ .$$

Prinzipiell sind alle möglichen Transformationen realisierbar. Allerdings lassen sich nur wenige Transformationen effizient umsetzen. Im Fall von Zeitreihen und Bildern ist insbesondere die Verschiebungsinvarianz von Interesse. Bei Zeitreihen können damit Muster unabhängig von ihrem konkreten zeitlichen Auftreten modelliert werden. Bei Bildern können die Muster unabhängig vom Ort im Bild modelliert werden. Im Folgenden soll daher näher auf die Verschiebungsinvarianz eingegangen werden.

Verschiebungsinvarianz

Für die Repräsentation der Verschiebungsinvarianz wird als Indexmenge $\mathcal{M}$ der Menge an Transformationen $\mathcal{T} = \{T^{(m)} : m \in \mathcal{M}\}$ die Menge der möglichen Verschiebungsvektoren gewählt. Für den Fall von Zeitreihen ist dies die zeitliche Verschiebung, also $\mathcal{M} = \mathbb{N}$. Im Fall von Bildern ist das die Menge der Verschiebungsvektoren in x- und y-Richtung, also $\mathcal{M} = \mathbb{N}^2$. In Anhang C.1 ist die konkrete Form der Operatoren als Verschiebungsmatrizen näher beschrieben.

Im Fall der Verschiebungsinvarianz können die Rekonstruktion und auch einige Terme der Gradienten effizient über Faltungsoperationen berechnet werden. Es wird zunächst die Rekonstruktion betrachtet, wie sie bisher formuliert wurde

$$\mathbf{R} = \sum_m \mathbf{T}^{(m)}\mathbf{W}\mathbf{H}^{(m)}.$$

Mit der konkreten Form der Verschiebungsoperatoren, wie sie in Anhang C.1 dargestellt ist, kann gezeigt werden, dass sich die Elemente der Rekonstruktion wie folgt ergeben

$$R_{i,n} = \sum_k \sum_m W_{i-m,k}H^{(m)}_{k,n}.$$

Dabei ist die Summe über m eine Faltung und kann entsprechend auch effizient implementiert werden. Um diese Darstellung explizit zu machen, wird eine Faltung zweier

Vektoren $\mathbf{v}$ und $\mathbf{w}$ als

$$u_i = (\mathbf{v} * \mathbf{w})_i = \sum_j v_{i-j} w_j$$

definiert.

Sei $W_{k,:}$ mit $(W_{k,:})_i = W_{k,i}$ der k-te Zeilenvektor von $\mathbf{W}$ und damit der k-te Basisvektor. Entsprechend sei $H^{(:)}_{k,n}$ mit $\left(H^{(:)}_{k,n}\right)_m = H^{(m)}_{k,n}$ der Vektor der Aktivierungen aller Verschiebungen des k-ten Basisvektors im n-ten Eingangsdatum. Damit gilt

$$R_{:,n} = \sum_k W_{:,k} * H^{(:)}_{k,n} \ .$$

Mit Hilfe von Fouriermethoden kann dieser Ausdruck effizient berechnet werden. Dazu werden die Faktoren mittels diskreter Fouriertransformation (DFT) in den Fourierraum transformiert, wo die Faltung durch eine einfache Multiplikation berechnet werden kann.

Die verschiebungsinvariante NMF wird auch kurz als Shift-NMF bezeichnet. Im Folgenden wird implizit immer die Shift-NMF verwendet. Aufgrund der einfacheren Darstellbarkeit wird aber die abstrakte Notation der Transformationsoperatoren soweit wie möglich beibehalten. Die Faltungsnotation wird nur explizit angegeben, wenn dies im Kontext und für weitere Detailerläuterungen notwendig ist.

4.2.4. Spärliche NMF

Eine weitere Modifikation des Problems erweist sich als nützlich. Dabei setzt man voraus, dass die Matrix $\mathbf{V}$ nicht-negativ ist und verlangt, dass die Faktoren $\mathbf{W}$ und $\mathbf{H}$ ebenfalls nicht-negativ sind. Dieses Problem wird „Spärliche nicht-negative Matrixfaktorisierung“ (NMF) genannt. Lee und Seung [Lee et al., 1999] zeigen, dass durch diese Beschränkung die Herausbildung interpretierbarer Basisvektoren und die Spärlichkeit der Aktivierungen unterstützt werden, da durch die additive Überlagerung positiver Basisvektoren eine teilebasierte Beschreibung der Daten entsteht. Es wird damit ausgeschlossen, dass sich Paare von Basisvektoren ausbilden, deren Anteile sich gegenseitig auslöschen können. Weiterhin kann dadurch in der Nebenbedingung der Betragsterm vereinfacht werden, womit sich das modifizierte Problem wie folgt darstellt:

$$\begin{aligned} &\min_{\mathbf{W},\mathcal{H}} \frac{1}{2}\|\mathbf{V} - \sum_m \mathbf{T}^{(m)}\mathbf{W}\mathbf{H}^{(m)}\|_2^2 + \lambda \sum_{m,i,j} H^{(m)}_{i,j} \\ &\text{sodass } \forall k : \|W_{:,k}\|_2 = 1 \\ &\qquad \mathbf{W} \geq 0 \\ &\qquad \mathcal{H} \geq 0, \end{aligned} \tag{4.4}$$

wobei $\mathcal{H} \geq 0$ bedeutet, dass alle Einträge im Tensor $\mathcal{H}$ nicht-negativ sind. Die Formulierung als NMF hat neben den oben genannten Eigenschaften auch noch einige algorithmische Vorteile, die in Abschnitt 4.3 beschrieben werden.

Nun sind allerdings im Allgemeinen die Eingangsdaten und damit die Matrix $\mathbf{V}$ nicht nicht-negativ. So können z. B. Bewegungstrajektorien je nach Interpretation der Sensordaten negative und positive Werte annehmen. Es gibt mehrere Möglichkeiten diesen Fall zu behandeln. Im Rahmen dieser Arbeit wurden zwei dieser Möglichkeiten untersucht.

Die erste Möglichkeit ist, die Forderung nach Nicht-Negativität der Matrix $\mathbf{W}$ aufzuheben. Damit kann auch die Matrix $\mathbf{V}$ sowohl positive als auch negative Einträge annehmen. Diese Formulierung ähnelt wieder sehr stark dem ursprünglichen Sparse Coding. Mit dieser Behandlung hebt man daher einige der positiven Eigenschaften der NMF auf. So können sich nun z. B. Paare von Basisvektoren herausbilden, deren Anteile sich gegenseitig auslöschen. Damit können diese Basisvektoren nicht mehr als Teile einer teilebasierten Beschreibung interpretiert werden. Weiterhin werden einige der algorithmischen Vorteile der NMF aufgehoben, wie im Abschnitt 4.3 gezeigt wird.

Die zweite Möglichkeit wird im Rahmen dieser Arbeit als „Split-NMF“ bezeichnet. Dabei werden die Eingangsdaten in negative und nicht-negative Teile getrennt und in eine nicht-negative Matrix projiziert. Sei $\mathbf{V} \in \mathbb{R}^{P \times N}$, so kann daraus eine nicht-negative Matrix $\mathbf{V}' \in {\mathbb{R}_+}^{2P \times N}$ in folgender Form gebildet werden, die dann anstatt der originalen Matrix $\mathbf{V}$ verwendet wird.

$$V' = \begin{pmatrix} \mathbf{V}^+ \\ \mathbf{V}^- \end{pmatrix}, \text{ mit}$$

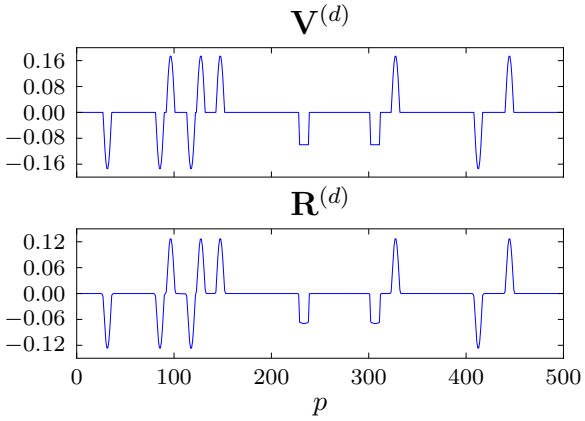
$$(\mathbf{V}^+)_{i,j} = \begin{cases} V_{i,j} & \text{,falls } V_{i,j} \geq 0 \\ 0 & \text{,sonst} \end{cases}, \quad (\mathbf{V}^-)_{i,j} = \begin{cases} -V_{i,j} & \text{,falls } V_{i,j} < 0 \\ 0 & \text{,sonst} \end{cases} -$$

Für die Experimente im Rahmen dieser Arbeit wurde die Split-NMF angewendet. Sie hat bei allen Experimenten gute Ergebnisse gezeigt.

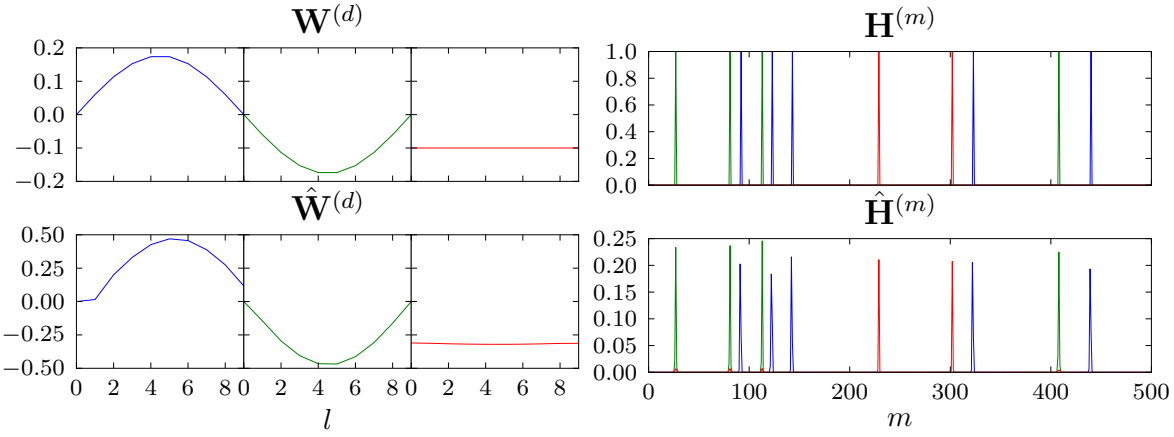
Abbildung 4.4 zeigt die Anwendung des bis hierher beschriebenen Verfahrens auf ein künstlich erzeugtes Signal. Das Signal wurde erzeugt, indem drei Basisvektoren manuell definiert und Aktivierungen zufällig mittels eines Poisson-Prozesses erzeugt wurden. Durch Faltung der Aktivierungen mit den manuell definierten Basisvektoren entsteht das künstliche sichtbare Signal. Wie gezeigt wird, erhält man nach Anwendung der spärlichen Shift-NMF eine gute Approximation sowohl der Basisvektoren als auch der Aktivierungen.

4.2.5. Lokal-spärliche NMF

Mit dem Gewicht λ ist es möglich, den Einfluss des Spärlichkeitsterms und damit die Spärlichkeit der Aktivierungen zu kontrollieren. Allerdings wirkt dieser Einfluss nur global. In bestimmten Situationen ist es hilfreich, eine noch feinere Kontrolle über die



(a) *Eingang und Rekonstruktion*



(b) *Basisvektoren* (c) *Aktivierungen*

Abb. 4.4.: *Anwendung der Spärlichen Shift-NMF auf ein künstlich erzeugtes Signal* $\mathbf{V}^{(d)}$ *(siehe (a) oben). Das Signal wurde durch Faltung händisch definierter Basisvektoren* $\mathbf{W}^{(d)}$ *(siehe (b) oben) und Aktivierungen* $\mathbf{H}^{(m)}$ *(siehe (c) oben) erzeugt. Unterschiedliche Basisvektoren k und die entsprechenden Aktivierungen sind farblich codiert. Daraus wurden die Basisvektoren* $\hat{\mathbf{W}}^{(d)}$ *(siehe (b) unten) und Aktivierungen* $\hat{\mathbf{H}}^{(m)}$ *(siehe (c) unten) gelernt und Rekonstruktion* $\mathbf{R}^{(d)}$ *(siehe (a) unten) erzeugt. Es ist zu erkennen, dass die gelernten Basisvektoren und Aktivierungen die künstlich erzeugten approximieren. Durch die Normierung der Basisvektoren, variiert allerdings die Skalierung der Aktivierungen.*

Spärlichkeit der Aktivierungen zu haben. Abbildung 4.5 (oben) zeigt die Aktivierungen nach Anwendung der Shift-NMF auf Bewegungstrajektorien. Die Aktivierungen sind über einen lokalen Bereich „verschmiert“. Die Modellvorstellung ist jedoch, dass die Aktivitäten ein diskretes Auftreten einer Bewegungsprimitive markieren. Um dieses Modell stärker zu forcieren, wird eine weitere Nebenbedingung eingeführt und das Optimierungsproblem wie folgt erweitert

$$\begin{aligned}
&\min_{\mathbf{W},\mathcal{H}} \frac{1}{2}\|\mathbf{V} - \sum_m \mathbf{T}^{(m)}\mathbf{W}\mathbf{H}^{(m)}\|_2^2 + \lambda_1 S_1(\mathcal{H}) + \lambda_2 S_2(\mathcal{H}) \\
&\text{sodass } \forall k : \|W_{:,k}\|_2 = 1 \\
&\qquad \mathbf{W} \geq 0 \\
&\qquad \mathcal{H} \geq 0 \\
&\text{mit } S_1(\mathcal{H}) = \sum_{m,i,j} H_{i,j}^{(m)} \\
&\qquad S_2(\mathcal{H}) = \sum_{n,k}\sum_m H_{k,n}^{(m)} \sum_{k'}\sum_{m'} \kappa_{k,k'}(m-m') H_{k',n}^{(m')}.
\end{aligned}$$

Die Funktion S_1 beschreibt dabei den bisher verwendeten Spärlichkeitsterm. Die Funktion S_2 führt einen Wettbewerb zwischen Aktivierungen unterschiedlicher Transformationen ein. Dabei beschreibt die Funktion κ das Gewicht oder die Stärke des Wettbewerbs zweier Aktivierungen. Beschränkt man sich bei den möglichen Transformationen auf Verschiebungen, wie in Abschnitt 4.2.3 beschrieben, so kann Sie z. B. wie folgt definiert werden

$$\kappa_{k,k'}(x) = \begin{cases} 0 & \text{, falls } k = k' \text{ und } x = 0 \\ (1 - |x|) \cdot I(|x| < 1) & \text{, sonst .} \end{cases}$$

Dabei ist $I(x)$ die Indikatorfunktion, deren Wert 1 ist, wenn das Argument wahr ist und sonst 0. In dieser Form realisiert κ eine mit der Entfernung x linear abnehmende Gewichtsfunktion. Für den Fall, dass $k = k'$, $x = 0$ ($m = m'$), also eine Aktivierung mit sich selbst im Wettbewerb stehen würde, wird das Gewicht auf 0 gesetzt.

Bei näherer Betrachtung der Funktion S_2 fällt auf, dass die Summe über m' eine Faltung beschreibt. So kann sie auch in folgender Weise definiert werden

$$S_2(\mathcal{H}) = \sum_{k,n}\sum_m H_{k,n}^{(m)} \left(\sum_{k'} \kappa_{k,k'} * H_{k',n}^{(:)} \right)_m$$

und bei entsprechender Repräsentation von $\mathcal{H}$ effizient mittels Fouriermethoden implementiert werden. Das Ergebnis der Faltung $\widetilde{H}_{k',n}^{(m)} = \left(\sum_{k'} \kappa_{k,k'} * H_{k',n}^{(:)} \right)_m$ wirkt als inhibitorischer Einfluss aller Aktivierungen (Gewichtet mit der Gewichtsfunktion, in

Abhängigkeit von der „Entfernung“ $m - m'$) auf die Aktivierung $H_{k,n}^{(m)}$.

Mit Hilfe des durch den Spärlichkeitsterm eingeführten Wettbewerbs, wird erzwungen, dass sich nur wenige Aktivierungen in einem lokalen Bereich durchsetzen. Somit ist es möglich, die „Verschmierungen“ der Aktivierungen über einen lokalen Bereich zu reduzieren. Abbildung 4.5 (unten) zeigt diesen Effekt. Damit kann die Spärlichkeit der Repräsentation insgesamt erhöht werden.

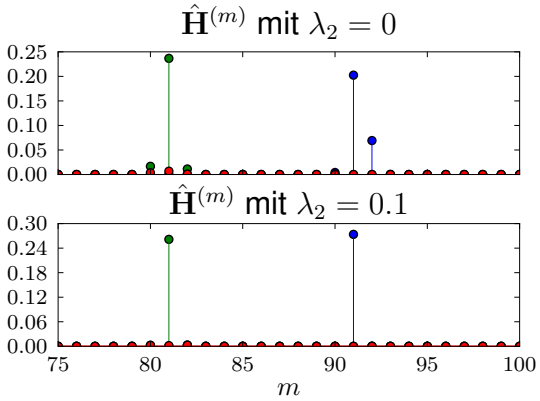


Abb. 4.5.: *Ausschnitt der Aktivierungen ohne (oben) und mit (unten) Einsatz der lokalen Spärlichkeit. Die Aktivierungen wurden aus den gleichen Daten wie in Abbildung 4.4 gelernt.*

4.2.6. Erweiterung auf multivariate Signale

In der bisherigen Formulierung wurden Signale als univariat betrachtet. Bewegungstrajektorien sind allerdings typischerweise multivariat. So ist die Trajektorie eines Gelenkes eines getrackten Skeletts trivariat oder ein gesamtes Skelett mit 15 Gelenken mit je drei Raumdimensionen 45-multivariat. Die Formulierung wird deshalb in folgender Weise erweitert. Pro Kanal werden nun je eine Eingangsmatrix $\mathbf{V}^{(d)}$, sowie Basismatrizen $\mathbf{W}^{(d)}$ mit $d \in \{1, \ldots, D\}$ für D Kanäle vorgehalten. Die Eingangsmatrizen $\mathbf{V}^{(d)}$ für $d = 1, \ldots, D$ werden als Scheiben im Tensor $\mathcal{V} \in \mathbb{R}^{P,N,D}$ zusammengefasst und die Basismatrizen $\mathbf{W}^{(d)}$ für $d = 1, \ldots, D$ als Scheiben im Tensor $\mathcal{W} \in \mathbb{R}^{L,K,D}$.

Damit wird das Optimierungsproblem wie folgt erweitert

$$\min_{\mathcal{W},\mathcal{H}} \frac{1}{2} \sum_d \|\mathbf{V}^{(d)} - \sum_m \mathbf{T}^{(m)} \mathbf{W}^{(d)} \mathbf{H}^{(m)}\|_2^2 + \lambda_1 S_1(\mathcal{H}) + \lambda_2 S_2(\mathcal{H}) \tag{4.5}$$

$$\begin{aligned} \text{sodass } & \forall k : \|W_{:,k}^{(:)}\|_2 = 1 \\ & \forall d : \mathbf{W}^{(d)} \geq 0 \\ & \forall m : \mathbf{H}^{(m)} \geq 0 \ . \end{aligned}$$

Entsprechend wird die Rekonstruktion erweitert

$$\mathbf{R}^{(d)} = \sum_m \mathbf{T}^{(m)} \mathbf{W}^{(d)} \mathbf{H}^{(m)} \ . \tag{4.6}$$

Eine Aktivierung $H_{k,n}^{(m)}$ aktiviert nun alle Kanäle eines Basisvektors gleichzeitig und ist deshalb unabhängig von d. Die Spärlichkeitsterme bleiben daher unberührt. Des Weiteren erfolgt die Normierung der Basisvektoren nun über alle Kanäle.

4.3. Algorithmen

Im Folgenden sollen die grundlegenden Lösungsalgorithmen für die oben vorgestellten Probleme dargestellt werden. In den letzten Jahren ist eine große Menge an Literatur zu effizienten Lösungsverfahren für das Sparse Coding entstanden. Die folgende Darstellung beschränkt sich daher auf die im Rahmen dieser Arbeit verwendeten Algorithmen zur Lösung der verschiebungsinvarianten NMF für multivariate Signale mit lokaler Spärlichkeit, wie sie in Abschnitt 4.2.6 vorgestellt wurde.

Für das Folgende ist es hilfreich, Teile der Zielfunktion explizit zu benennen. Diese werden wie folgt benannt:

$$F(\mathcal{W},\mathcal{H}) = \underbrace{\frac{1}{2} \sum_d \| \mathbf{V}^{(d)} - \sum_m \mathbf{T}^{(m)} \mathbf{W}^{(d)} \mathbf{H}^{(m)} \|_2^2}_{F_r(\mathcal{W},\mathcal{H})} + \underbrace{\lambda_1 S_1(\mathcal{H})}_{F_1(\mathcal{H})} + \underbrace{\lambda_2 S_2(\mathcal{H})}_{F_2(\mathcal{H})} \tag{4.7}$$

Das Problem, wie es in Gleichung 4.5 dargestellt ist, ist durch die simultane Optimierung über beide Tensoren $\mathcal{W}$ und $\mathcal{H}$ nicht-konvex und kann daher auch nicht direkt mit Mitteln der konvexen Optimierung behandelt werden. Der typische Ansatz zur Lösung des Problems ist daher, es in zwei Teilprobleme aufzuteilen, wobei jeweils einer der beiden Parameter als fest betrachtet wird und nur über den anderen Parameter optimiert wird. Beide Teilprobleme sind dann jeweils konvex und werden iterativ, alternierend optimiert, womit sich die beiden Parameter der Lösung des Originalproblems annähern. Im Folgenden sollen die beiden Teilprobleme näher betrachtet werden.

4.3.1. Codierungsproblem

Beim ersten Teilproblem bleibt der Parameter $\mathcal{W}$ fest. Dabei wird nur über die Aktivierungsmatrizen $\mathcal{H}$ optimiert, womit es sich um ein reines Codierungsproblem handelt. Dieses Problem ist konvex und kann durch einen modifizierten Gradientenabstieg gelöst werden. Dabei wird in einer Iteration zunächst ein Gradientenabstieg ausgeführt. Da dadurch die Lösung aus der durch die Nebenbedingung beschränkten Lösungsmenge

(der nicht-negative Orthant [1]) heraus laufen kann, muss sie nach jedem Schritt in den nicht-negativen Orthanten zurück projiziert werden [2].

Obwohl die Unstetigkeit des Spärlichkeitsterms durch die Verwendung der Betragsnorm im Allgemeinen einen Gradientenabstieg unmöglich macht, ist dies im speziellen Fall der NMF dennoch möglich, da die Lösungsmenge auf den nicht-negativen Orthanten beschränkt ist und sich die Lösung damit nur im stetigen Teil der Funktion bewegt.

Der Gradient der Zielfunktion bezüglich $\mathcal{H}$ ergibt sich dann wie folgt

$$\begin{aligned}
&\nabla_{\mathbf{H}^{(m)}} F = \nabla_{\mathbf{H}^{(m)}} F_r + \nabla_{\mathbf{H}^{(m)}} F_1 + \nabla_{\mathbf{H}^{(m)}} F_2 \\
\text{mit } &\nabla_{\mathbf{H}^{(m)}} F_r = \sum_d (-\mathbf{T}^{(m)}\mathbf{W}^{(d)})^T (\mathbf{V}^{(d)} - \mathbf{R}^{(d)}) \ , \\
&\nabla_{\mathbf{H}^{(m)}} F_1 = \lambda_1 \mathbf{1}_{K \times N} \ , \\
&\nabla_{\mathbf{H}^{(m)}} F_2 = \lambda_2 \mathbf{G}^{(m)} \text{ mit } G^m_{k,n} = \left(\sum_{k'} \kappa_{k,k'} * H^{(:)}_{k',n} \right)_m \ .
\end{aligned}$$

Dabei ist $\mathbf{R}^{(d)}$ die Rekonstruktion, wie in Gleichung 4.6 definiert. Nachdem ein Schritt des Gradientenabstiegs durchgeführt wurde, muss die aktuelle Zwischenlösung in den zulässigen Lösungsbereich zurück projiziert werden. Dies kann durch Setzen der negativen Einträge von $\mathbf{W}^{(d)}$ auf Null erreicht werden. Algorithmus 1 zeigt einen Schritt des Gradientenabstiegs des Codierungsproblems der NMF. Dabei ist η_1 die Schrittweite des

Algorithmus 1 Schritt des Gradientenabstiegs für das Codierungsproblem der NMF.

Funktion $\mathcal{H} \leftarrow$ NMFCodingStep$(\mathcal{V}, \mathcal{W}, \mathcal{H}, \mathcal{T}, \lambda_1, \lambda_2, \eta_1)$
  $\mathcal{R} \leftarrow$ Reconstruct$(\mathcal{V}, \mathcal{W}, \mathcal{H}, \mathcal{T})$
  $\forall m : \mathbf{H}^{(m)} \leftarrow \mathbf{H}^{(m)} - \eta_1 \nabla_{\mathbf{H}^{(m)}} F$ ▷ Schritt des Gradientenabstiegs
  $\forall m : \mathbf{H}^{(m)} \leftarrow$ ProjectNonNeg$(\mathbf{H}^{(m)})$
Ende Funktion

Funktion $\mathcal{R} \leftarrow$ Reconstruct$(\mathcal{V}, \mathcal{W}, \mathcal{H}, \mathcal{T})$
  $\forall d : \mathbf{R}^{(d)} \leftarrow \sum_m \mathbf{T}^{(m)}\mathbf{W}^{(d)}\mathbf{H}^{(m)}$
Ende Funktion

Funktion $\mathbf{X} \leftarrow$ ProjectNonNeg$(\mathbf{X})$
  $\mathbf{X} \leftarrow \mathbf{X}^+$, mit $(\mathbf{X}^+)_{i,j} = \begin{cases} X_{i,j} & \text{,falls } X_{i,j} \geq 0 \\ 0 & \text{,sonst} \end{cases}$
Ende Funktion

[1] Ein Orthant ist die Generalisierung eines Quadranten eines kartesischen Koordinatensystems auf einen mehrdimensionalen euklidischen Raum.

[2] Negative Anteile werden einfach auf Null gesetzt.

Gradientenabstiegs. In Zeile 4 wird das potentiell negative Ergebnis durch Setzen der negativen Werte auf Null in den nicht-negativen Lösungsbereich zurück projiziert.

4.3.2. Dictionary-Learning-Problem

Das zweite Teilproblem der NMF ergibt sich, wenn $\mathcal{H}$ fest bleibt und nur über $\mathcal{W}$ optimiert wird. Dieses Problem wird als Dictionary-Learning-Problem bezeichnet, weil nur die Basis, also das Wörterbuch, optimiert oder gelernt wird. Da die Spärlichkeitsterme nicht von $\mathcal{W}$ abhängig sind, können sie für die Optimierung ignoriert werden. Damit ergibt sich als Gradient für das Dictionary-Learning-Problem

$$\begin{aligned}\nabla_{\mathbf{W}^{(d)}} F &= \nabla_{\mathbf{W}^{(d)}} F_r \\ &= \sum_m (\mathbf{T}^{(m)})^T (-\mathbf{V}^{(d)} + \mathbf{R}^{(d)})(\mathbf{H}^{(m)})^T \ .\end{aligned}$$

Auch hier muss die Lösung durch setzen der negativen Einträge auf Null nach jedem Schritt in den zulässigen Lösungsbereich projiziert werden. Des Weiteren muss die Nebenbedingung der Forderung nach Einheitsnorm umgesetzt werden. Durch Normierung der Basisvektoren nach jedem Iterationsschritt kann diese Forderung erfüllt werden. Dies kann jedoch zur Erhöhung des Wertes der Zielfunktion führen und beeinträchtigt die Konvergenz. Da dieses Problem eher von theoretischer Relevanz ist, soll an dieser Stelle nicht darauf eingegangen werden. Algorithmus 2 zeigt einen Schritt des Gradientenabstiegs des Dictionary-Learning-Problems der NMF. Dabei ist η_2 die Schrittweite des

Algorithmus 2 Schritt des Gradientenabstiegs für das Dictionary-Learning-Problem der NMF.

1: **Funktion** $\mathcal{W} \leftarrow$ NMFDictLearnStep$(\mathcal{V}, \mathcal{W}, \mathcal{H}, \mathcal{T}, \eta_2)$
2: $\quad \mathcal{R} \leftarrow \text{Reconstruct}(\mathcal{V}, \mathcal{W}, \mathcal{H}, \mathcal{T})$
3: $\quad \forall d : \mathbf{W}^{(d)} \leftarrow \mathbf{W}^{(d)} - \eta_2 \nabla_{\mathbf{W}^{(d)}} F$ ▷ Schritt des Gradientenabstiegs
4: $\quad \forall d : \mathbf{W}^{(d)} \leftarrow \text{ProjectNonNeg}(\mathbf{W}^{(d)})$
5: $\quad \forall d : \mathbf{W}^{(d)} \leftarrow \text{NormBasis}(\mathcal{W})$
6: **Ende Funktion**
7:
8: **Funktion** $\mathcal{W} \leftarrow$ NormBasis$(\mathcal{W})$
9: $\quad \forall d : \mathbf{W}^{(d)} \leftarrow \overline{\mathbf{W}}^{(d)}, \text{mit } \overline{W}^{(d)}_{i,k} = \frac{W^{(d)}_{i,k}}{\left\| W^{(:)}_{:,k} \right\|_2}$ ▷ Spaltenweise Normierung
10: **Ende Funktion**

Gradientenabstiegs. Die Funktionen Reconstruct und ProjectNonNeg wurden bereits in Algorithmus 1 definiert.

4.3.3. Alternierendes Update

Zur Lösung des Gesamtproblems werden die Schritte des Codierungsproblems und des Dictionary-Learning-Problems in alternierender Weise angewendet. Dazu werden beide Tensoren zuerst zufällig mit einem Gaussschen Rauschen initialisiert. Dann werden abwechselnd die Basisvektoren und die Aktivierungen aktualisiert. Dies wird solange wiederholt, bis die beiden Faktoren oder der Wert der Zielfunktion konvergieren. Auf konkrete Konvergenzkriterien wird in Abschnitt 4.5 eingegangen. Algorithmus 3 zeigt den alternierenden Gradientenabstieg für die spärliche NMF.

Algorithmus 3 Gradientenabstieg für die NMF.

1: **Funktion** $\mathcal{W},\mathcal{H} \leftarrow \mathrm{NMF}(\mathcal{V}, \mathcal{W}, \mathcal{H}, \mathcal{T}, \lambda_1, \lambda_2, \eta_1, \eta_2)$
2: initialisiere $\mathcal{W}$ und $\mathcal{H}$ zufällig
3: **wiederhole**
4: $\mathcal{W} \leftarrow \mathrm{NMFDictLearnStep}(\mathcal{V}, \mathcal{W}, \mathcal{H}, \mathcal{T}, \eta_1)$
5: $\mathcal{H} \leftarrow \mathrm{NMFCodingStep}(\mathcal{V}, \mathcal{W}, \mathcal{H}, \mathcal{T}, \lambda_1, \lambda_2, \eta_2)$
6: **bis** Konvergenz
7: **Ende Funktion**

4.3.4. Multiplikative Update-Regeln

Ein praktisches Problem des oben vorgestellten Verfahrens ist das Einstellen der Schrittweiten η_1 und η_2. Das Konvergenzverhalten hängt sehr sensibel von den gewählten Schrittweiten ab. Wählt man die Schrittweiten zu groß, neigt der Gradientenabstieg zu Schwingungen und Divergenz. Wählt man die Schrittweiten zu klein, konvergiert das Verfahren nur sehr langsam. Lee und Seung [Lee et al., 1999] stellen eine modifizierte Variante des Gradientenabstiegs vor, bei dem die beiden Faktoren durch Multiplikation aktualisiert werden. Dabei lassen sich die Schrittweiten heraus kürzen und fallen somit weg.

Wendet man den Trick auf die hier vorgestellten Gleichungen an, ergibt sich ein Schritt im Codierungsproblem damit zu

$$\mathbf{H}^{(m)}_{k,n} \leftarrow \mathbf{H}^{(m)}_{k,n} \odot \frac{\sum_d (\mathbf{T}^{(m)} W^{(d)}_{:,k})^T V^{(d)}_{:,n}}{\sum_d (\mathbf{T}^{(m)} W^{(d)}_{:,k})^T R^{(d)}_{:,n} + \lambda_1 + \lambda_2 \left(\sum_{k'} \kappa_{k,k'} * H^{(:)}_{k',n} \right)_m} \,.$$

Und ein Schritt im Dictionary-Learning-Problem ergibt sich zu

$$\mathbf{W}^{(d)} \leftarrow \mathbf{W}^{(d)} \odot \frac{\sum_m (\mathbf{T}^{(m)})^T \mathbf{V}^{(d)} (\mathbf{H}^{(m)})^T}{\sum_m (\mathbf{T}^{(m)})^T \mathbf{R}^{(d)} (\mathbf{H}^{(m)})^T} \,.$$

Der Operator $\odot$ beschreibt die elementweise Multiplikation, während die Division eben-

falls elementweise erfolgt. Eine wichtige Voraussetzung für die Anwendung des Verfahrens ist, dass die Eingangsdaten und die Faktoren nicht-negativ sind. Durch das multiplikative Update ist weiterhin sichergestellt, dass die Faktoren nicht-negativ bleiben. Damit entfällt auch die Projektion der Lösung nach einem Schritt in den Nicht-negativen Lösungsbereich und der Algorithmus kann weiter vereinfacht werden. Algorithmus 4 zeigt die NMF mit multiplikativem Update.

Algorithmus 4 Multiplikative NMF.

Funktion $\mathcal{W}$,$\mathcal{H} \leftarrow$ NMFMul$(\mathcal{V}, \mathcal{W}, \mathcal{H}, \mathcal{T}, \lambda_1, \lambda_2)$
  initialisiere $\mathcal{W}$ und $\mathcal{H}$ zufällig
  wiederhole
    $\mathcal{R} \leftarrow \text{Reconstruct}(\mathcal{W}, \mathcal{H}, \mathcal{T})$
    $\mathbf{H}^{(m)}_{k,n} \leftarrow \mathbf{H}^{(m)}_{k,n} \cdot \sum_d \frac{(\mathbf{T}^{(m)} W^{(d)}_{:,k})^T V^{(d)}_{:,n}}{(\mathbf{T}^{(m)} W^{(d)}_{:,k})^T R^{(d)}_{:,n} + \lambda_1 + \lambda_2 \left(\sum_{k'} \kappa_{k,k'} * H^{(:)}_{k',n}\right)_m}$
    $\mathcal{R} \leftarrow \text{Reconstruct}(\mathcal{W}, \mathcal{H}, \mathcal{T})$
    $\mathbf{W}^{(d)} \leftarrow \mathbf{W}^{(d)} \odot \frac{\sum_m (\mathbf{T}^{(m)})^T \mathbf{V}^{(d)} . (\mathbf{H}^{(m)})^T}{\sum_m (\mathbf{T}^{(m)})^T \mathbf{R}^{(d)} (\mathbf{H}^{(m)})^T}$
    $\mathcal{W} \leftarrow \text{NormBasis}(\mathcal{W})$
  bis Konvergenz
Ende Funktion

4.3.5. Trainings- und Anwendungsphase

Der bisher vorgestellte Algorithmus beschreibt das Lernen von Basisvektoren und Aktivierungen aus einer Menge von Eingangssignalen. Wenn das Verfahren als Merkmalsextraktionsschritt in einem Mustererkennungsprozess angewendet wird, so wird es in der Trainings- und in der Anwendungsphase in zwei unterschiedlichen Modi betrieben.

In der Trainingsphase steht dem Mustererkennungsprozess eine Menge von Trainingssignalen, gegebenenfalls mit Klasseninformation, zur Verfügung. Diese kann z. B. durch Fensterung eines einzigen, langen Signals entstanden sein. Das Verfahren wird nun in seiner vollständigen Form, wie oben vorgestellt, als unüberwachtes Lernverfahren angewendet, um die Basisvektoren zu lernen, welche dann als Wörterbuch für die später beschriebene Anwendungsphase gespeichert werden. Die simultan gelernten Aktivierungen sind für den Merkmalsextraktionsschritt nicht mehr von Bedeutung, können aber als transformierte Eingangssignale an nachgelagerte Verarbeitungsschritte für das Training weiterer Modelle, wie Klassifikatoren, etc. weitergegeben werden.

In der Anwendungsphase wird das Sparse Coding als reines Codierungsverfahren mit einem festen Wörterbuch (also fester Basis $\mathcal{W}$) ausgeführt. Dabei entfällt der Schritt des Dictionary-Learnings und die Iteration wird nur mit dem Codierungsschritt (siehe Algorithmus 1) ausgeführt. Die codierten Signale, also die Aktivierungen, werden dann an den folgenden Verarbeitungsschritt weitergegeben.

Gegenüber konventionellen lernenden Merkmalsextraktionsverfahren, wie der PCA, hat der hier vorgestellte Algorithmus den Nachteil, dass die Codierung durch einen aufwändigen iterativen Gradientenabstieg erfolgt, während die Codierung mit der PCA in geschlossener Form durch eine Matrixmultiplikation erfolgen kann. Sie ist also grundsätzlich komplexer.

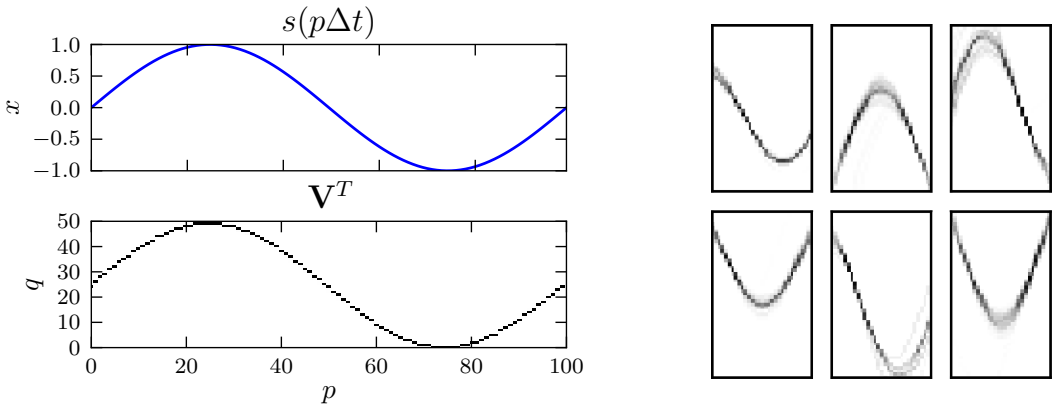
Um diesen Nachteil auszugleichen, kann in der Anwendungsphase ein alternativer Sparse-Coding-Algorithmus namens „Matching Pursuit“ (MP) [Mallat et al., 1993] eingesetzt werden. Im folgenden wird die Idee des MP kurz umrissen. Der Algorithmus wird in Anhang C.2 detaillierter beschrieben. MP ist ein heuristisches Verfahren für das Sparse Coding. Es geht iterativ vor. In der ersten Iteration wird das Tripel aus Index n des Eingangsdatums, Index k der Primitive und Verschiebung m der Primitive im Eingangsdatum gesucht, welches die Korrelation zwischen der k-ten Primitive mit dem n-ten Eingangsdatum bei Verschiebung m maximiert. Dieses Tripel, zusammen mit dem Wert der Korrelation, beschreibt die erste gefundene Aktivierung. Für die nächste Iteration wird die entsprechend aktivierte Primitive vom Eingangsdatum subtrahiert. Das Ergebnis ist das sogenannte Residuum und dient als Eingangsdatum für die nächste Iteration. So wird in jeder Iteration eine weitere Aktivierung gefunden, solange bis ein Konvergenzkriterium erfüllt ist. Das kann z. B. ein Unterschreiten einer bestimmten Signalenergie des Residuums sein oder das Erreichen einer maximalen Anzahl an Iterationen und damit einer maximalen Anzahl an Aktivierungen.

Im Vergleich zur NMF hat das Verfahren eine geringere Komplexität (pro Iteration eine Kreuzkorrelation jeder Primitive mit jedem Eingangsdatum, wohingegen bei der NMF pro Iteration eine Kreuzkorrelation und eine Faltung jeder Primitive mit jedem Eingangsdatum notwendig ist). Außerdem lässt sich die Spärlichkeit durch Begrenzung der Anzahl der Iterationen und damit der Aktivierungen direkt steuern. Des Weiteren lässt sich auch eine lokale Spärlichkeit viel leichter erreichen, indem heuristisch in Nachbarregionen zu bereits gefundenen Aktivierungen keine weiteren Aktivierungen gesucht werden.

Im Rahmen dieser Arbeit konnte dieses Verfahren allerdings aus Zeitgründen nicht implementiert und näher betrachtet werden. Aufgrund der oben beschriebenen Eigenschaften, scheint sich das Verfahren aber sehr gut für die praktische Anwendung zu eignen und sollte daher Gegenstand aufbauender Arbeiten sein (siehe dazu Abschnitt 6.3.2).

4.4. Repräsentation der Eingabedaten

Bis jetzt wurde davon ausgegangen, dass ein Eingangssignal $s(t)$ direkt als Eingabevektor $\mathbf{v} \in \mathbb{R}^P$ mit $\mathbf{v} = (v_1, \ldots, v_P)^T = (s(\Delta t), s(2\Delta t), \ldots, s(P\Delta t))^T$ repräsentiert wird. In dieser Arbeit wird diese Form der Eingabecodierung die „direkte“ Codierung genannt. Hellbach [Hellbach, 2009] stellt in seiner Arbeit eine alternative Repräsentation der Eingangsdaten in Form eines Binärbildes vor und wendet darauf die NMF an. In die-



(a) *gridbasierte Repräsentation* (b) *Basisvektoren*

Abb. 4.6.: *(a) Transformation eines Signals $s(t)$ in eine Matrix $\mathbf{V}$ als gridbasierte Eingabecodierung. $\mathbf{V}$ kann als Binärbild interpretiert werden. (b) Basisvektoren, die aus einer Menge von gridbasierten Sinus-Kurven gelernt wurden. In den Basisvektoren ist die Varianz des Datensatzes durch Teilbelegungen der Gitterzellen abgebildet.*

ser Arbeit wird diese Art der Codierung „gridbasiert“ genannt. Im Folgenden sollen die beiden Ansätze und ihre Eigenschaften diskutiert werden. Aus Gründen der einfacheren Notation wird die Darstellung auf univariate Signale beschränkt. Beide Verfahren sind aber auch auf multivariate Signale anwendbar und die Aussagen gelten auch dafür ohne Einschränkungen.

Bei der gridbasierten Codierung wird jede Stützstelle des Eingangssignals in einen Binärvektor transformiert. Dazu wird der Wertebereich des Signals in Q Intervalle $\{[(q-1)\Delta x, q\Delta x), q = 1, \ldots, Q\}$ aufgeteilt. Der Binärvektor $\mathbf{v}_p \in \mathbb{R}^Q$ wird dann als eine 1-aus-Q-Codierung des quantisierten Signals definiert, also

$$v_{p,q} = \begin{cases} 1 & , \text{ falls } s(p\Delta t) \in [(q-1)\Delta x, q\Delta x) \\ 0 & , \text{ sonst } . \end{cases}$$

Ein einzelnes Eingangssignal kann dann als Matrix $\mathbf{V} \in \mathbb{R}^{P \times Q}$ dargestellt werden, wobei in jeder Zeile nur jeweils ein Element 1 ist und alle anderen 0. $\mathbf{V}$ kann somit auch als Binär-Plot des Signals interpretiert werden. Abbildung 4.6a zeigt die Transformation an einem einfachen Beispiel.

Diese Darstellung hat zwei prinzipielle Vorteile. Zum einen sind die Eingangsdaten von vornherein nicht-negativ. Damit kann die NMF direkt darauf angewendet werden, ohne das noch weitere Transformationen der Repräsentation wie bei der direkten Codierung (siehe Abschnitt 4.2.4) notwendig sind. Zum anderen kann, wie oben schon erwähnt, jedes Eingangsdatum als Binärbild behandelt werden. Damit kann eine große Auswahl speziell für Bilddaten optimierter Sparse-Coding-Algorithmen angewendet werden.

Ein weiterer Vorteil ist, dass sich die Varianz der Daten, also die Unsicherheit, in

den gelernten Basisvektoren abbilden lässt. Abbildung 4.6b zeigt sechs Basisvektoren, die aus einem künstlichen Datensatz gewonnen wurden. Der Datensatz bestand aus einer Menge von 500 Eingangssignalen in Form von Sinus-Funktionen, deren Frequenz und Amplitude leicht variiert wurde und die mit $P = 100$ Stützstellen abgetastet und mit $Q = 50$ Intervallen quantisiert wurden. Durch die gridbasierte Repräsentation hat das Verfahren die Möglichkeit, Variationen in den Daten durch Teilbelegungen der Gitterzellen der Basisvektoren zu erfassen, wie in Abbildung 4.6b an den Grauwerten der Gitterzellen zu erkennen ist. Die direkte Eingaberepräsentation bietet keine Möglichkeit, die Unsicherheit abzubilden.

Es ist allerdings bisher unklar, wie sich diese Eigenschaft praktisch auswirkt und unter welchen Rahmenbedingungen sie tatsächlich nützlich ist. Sie bleibt daher von eher theoretischer Bedeutung. Zusätzlich bringt die gridbasierte Eingaberepräsentation einige Nachteile gegenüber der direkten Repräsentation mit sich. So ist zur adäquaten Erfassung der Signaleigenschaften häufig eine sehr feine Quantisierung notwendig. Dadurch erhöht sich der Speicherbedarf zur Repräsentation aller Eingabedaten und Basisvektoren enorm. Somit können übliche Datenmengen mit handelsüblichen Rechnern nicht mehr behandelt werden. Für den Einsatz in einem realwelttauglichen Mustererkennungssystem ist also die direkte Eingabecodierung vorzuziehen.

4.5. Praktischer Einsatz

In diesem Abschnitt sollen einige praktische Details beim Einsatz des Verfahrens auf realen Daten näher betrachtet werden. Dazu zählen Dinge, wie eine geeignete Vorverarbeitung der Daten, sinnvolle Konvergenzkritierien für die iterative Optimierung und das Vorgehen bei der Wahl der Parameter.

Für die folgenden Erläuterungen werden Daten verwendet, die im Rahmen dieser Arbeit selbst aufgenommen wurden (für eine detaillierte Darstellung siehe Kapitel 3.3.1).

4.5.1. Vorverarbeitung

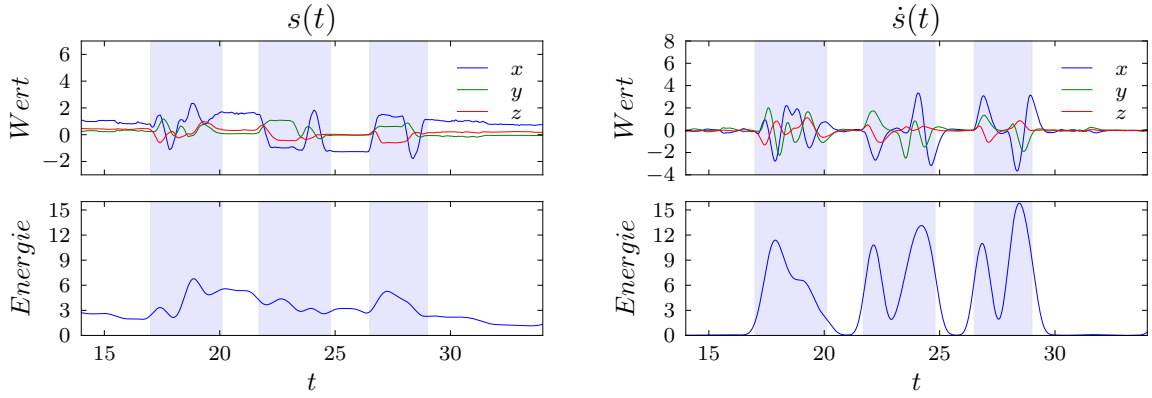
Mit einer günstigen Vorverarbeitung kann man die Herausbildung sinnvoller und interpretierbarer Basisvektoren fördern. Dazu müssen einige Eigenschaften des Optimierungsprozesses näher betrachtet werden. Aufgrund der Formulierung der Energiefunktion (vgl. Gleichung 4.7) ist der Optimierungsprozess bestrebt, den Rekonstruktionsfehler zu minimieren und dabei möglichst wenige Aktivierungen zu verwenden. Die Optimierung wird daher besonders dort, wo die lokale Signalenergie [3] relativ hoch ist, zuerst Aktivierungen ausbilden und die Basisvektoren entsprechend zuerst an diese Signalanteile

[3] Die lokale Signalenergie eines Signals s zum Zeitpunkt t_0 sei das Integral des Signalbetrages über Fenster $[t_0, t_0 + \Delta t]$, formal $\int_{t=t_0}^{t_0+\Delta t} |(s(t)|_2^2 w(t - t_0)$ mit einer geeigneten Fensterfunktion w (z. B. Rechteck-Funktion).

anpassen. Es ist also sinnvoll, das Signal in einer Vorverarbeitung so zu transformieren, dass die interessanten Signalteile eine große Signalenergie aufweisen.

Signalinterpretation

Im Folgenden wird eine typische Bewegungstrajektorie der rechten Hand einer Person betrachtet. Abbildung 4.7 zeigt zwei unterschiedliche Interpretationen der Sensorsignale.



(a) *Position* (b) *Geschwindigkeit*

Abb. 4.7.: *Zwei Interpretationen des Signalverlaufs der rechten Hand bei Ausführung dreier Gesten. Abbildung (a) zeigt den Verlauf der Postion (oben) und darunter die Signalenergie. Abbildung (b) zeigt den Verlauf der Geschwindigkeit (oben) und darunter wiederum die Signalenergie. Die Signalabschnitte, in denen eine Geste ausgeführt wurde, sind blau hinterlegt.*

In Abbildung 4.7a besteht das Signal aus dem Verlauf der Position der Hand über die Zeit. Das Signal wurde mittelwertfrei gemacht und normiert. Die Signalabschnitte, in denen tatsächliche eine Geste vollführt wurde, sind blau hinterlegt. Zwischen diesen Abschnitten wurde die Hand nicht bewegt. Obwohl diese Ruhephasen keine Information enthalten, tragen sie dennoch zur Varianz der Daten bei. Das Sparse Coding wird versuchen, für diese Zwischenräume Basisvektoren auszubilden und entsprechend auch Aktivitäten für diese Abschnitte ausbilden.

In Abbildung 4.7b besteht das Signal aus dem Verlauf der Geschwindigkeit der Hand über die Zeit [4]. Das Signal wurde auch hier mittelwertfrei gemacht und normiert. Der Verlauf der Signalenergie zeigt, dass in den Zwischenräumen die Signalenergie klein bleibt und damit wenig zur Varianz der Daten beiträgt. Somit wird sich das Sparse Coding zuerst auf die blau hinterlegten Bereiche konzentrieren und die Ruhephasen vernachlässigen. Das Verfahren kann also dahingehend unterstützt werden, dass es nur

[4]Da die Geschwindigkeit nicht direkt ableitbar ist, muss sie geschätzt werden. Die Schätzung wurde hier mittels eines Kalman-Filters [Kalman, 1960] durchgeführt (siehe auch Abschnitt 3.2.2).

für Signalabschnitte, die auch Bewegungen und Gesten enthalten, Basisvektoren ausbildet.

Die Wahl der Geschwindigkeit als Merkmalsraum unterstützt also das Verfahren bei der Herausbildung interpretierbarer Basisvektoren. Im Folgenden wird deshalb der Verlauf der Geschwindigkeit in x- und y-Richtung als Eingangssignal für das Sparse Coding verwendet. Dabei wird die Geschwindigkeit der n-ten Aufnahme in x-Richtung durch $V^{(0)}_{:,n}$ und in y-Richtung durch $V^{(1)}_{:,n}$ repräsentiert.

Normierung

Normierung wird generell als ein sinnvoller Vorverarbeitungsschritt angesehen. Allerdings kann die Normierung auch negative Effekte haben. Wenn die Muster, nach denen gesucht wird, bereits durch eine hohe Varianz der Daten in den entsprechenden Signalabschnitten gekennzeichnet sind, so ist eine Normierung unter Umständen nicht nötig. Sie kann sogar kontra-produktiv sein. Haben z. B. einzelne Kanäle des Signals nur sehr geringe Amplituden und insgesamt ein ein geringes Signal-Rausch-Verhältnis, so würde eine Normierung dieses Rauschen hoch skalieren. Dadurch würde die Varianz in diesen Kanälen unter Umständen die Varianz der Daten dominieren, und das Sparse Coding würde Basisvektoren bevorzugt für unbedeutende Signalabschnitte ausbilden. Die Sinnfälligkeit der Normierung hängt also sehr stark von den Signaleigenschaften und der Charakteristik der gesuchten Muster ab. Es kann deshalb an dieser Stelle keine generelle Empfehlung für oder gegen die Anwendung der Normierung gegeben werden.

4.5.2. Konvergenzkriterien

Als Konvergenzkriterien für die iterative Minimierung der Energiegleichung 4.7 durch Algorithmus 3 bzw. 4 haben sich einige Heuristiken etabliert. So kann z. B. eine obere Grenze für den Rekonstruktionsterm $F_r(\mathcal{H},\mathcal{W}) < B$ (vgl. Gleichung 4.7) gewählt werden. Diese wird jedoch durch die Wahl der Spärlichkeitsparameter λ_1 und λ_2 beeinflusst, da sie sich auf den Rekonstruktionsfehler indirekt auswirken. Ein Kriterium, welches von den beiden Parametern unabhängig ist, ist eine obere Grenze für die Änderung des Rekonstruktionsfehlers $|F_r(\mathcal{H}_{(i)},\mathcal{W}_{(i)}) - F_r(\mathcal{H}_{(i-1)},\mathcal{W}_{(i-1)})| < B$, wobei die geklammerten Indizes hier die Nummer des Iterationsschrittes angeben. Diese Formulierung hat immer noch den Nachteil, dass die Grenze abhängig von der Skala des Fehlers ist. Deshalb hat sich die folgende Formulierung als praktisch erwiesen

$$\frac{|F_r(\mathcal{H}_{(i)},\mathcal{W}_{(i)}) - F_r(\mathcal{H}_{(i-1)},\mathcal{W}_{(i-1)})|}{|F_r(\mathcal{H}_{(i)},\mathcal{W}_{(i)})|} < B \ ,$$

welche die relative Fehlerdifferenz misst. Wählt man z. B. $B = 0{,}001$ (die Änderung des Rekonstruktionsfehlers entspricht 0,1% der Energie des Residuums), so kann davon ausgegangen werden, dass die Minimierung hinreichend abgeklungen ist und einen sta-

tionären Punkt erreicht hat. Im Fall des addidiven Gradientenabstiegs in Algorithmus 3, ist es weiterhin sinnvoll, eine obere Schranke für die Anzahl der Iterationsschritte anzugeben. Denn im Fall zu hoch gewählter Schrittweiten η_1 und η_2 kann es hier zu starken Oszillationen oder sogar zur Divergenz des Rekonstruktionsfehlers kommen.

Aufgrund vieler lokaler Minima ist die Güte der gefundenen Lösung sehr stark von der Initialisierung abhängig. Um dieses Problem abzumildern, wird die Minimierung mehrmals mit zufälliger Initialisierung der Parameter ausgeführt. Diese Methode wird häufig auch als Multistart-Ansatz bezeichnet.

4.5.3. Bestimmung der Parameter

Die wesentlichen Parameter des Verfahrens sind die Anzahl und Größe der Basisvektoren, sowie die Spärlichkeitsparameter λ_1 und λ_2 (vgl. Gleichung 4.5). Im Folgenden soll das Vorgehen zur Wahl der Parameter näher betrachtet werden.

Anzahl der Basisvektoren

Der Wahl der Anzahl K der Basisvektoren können mehrere Erwägungen zugrunde gelegt werden. So kann z. B. die Dimension des Ausgangsraumes bestimmend sein, wenn diese durch nachgelagerte Verarbeitungsschritte vorgegeben ist. Bei K Basisvektoren mit je M Verschiebungen wird ein Eingangsdatum nach der Codierung mit $K \cdot M$ Aktivierungen repräsentiert. Häufig werden mehrere Verschiebungen eines Basisvektors zu einer Aktivierung zusammengefasst (siehe Aggregierung der Aktivierungen für die Klassifikation in Abschnitt 5.2.1). Sei die Anzahl der zusammengefassten Aktivierungen je Basisvektor M', dann ist die Dimension des Ausgaberaumes $K \cdot M'$.

Wenn die Anzahl der Basisvektoren nicht derart beschränkt ist, so kann sie anhand der Eigenschaften der Daten heuristisch bestimmt werden. In Abschnitt 4.5.1 wurde bereits ausgeführt, dass durch Adaption und Aktivierung der Basisvektoren die Energie des Residuums verringert wird. Dabei werden die Basisvektoren zuerst an die Signalanteile mit der größten Energie angepasst. Ein bestimmtes Muster in den Daten hat dabei eine höhere Energie, wenn es öfter in den Daten auftritt. Ein Basisvektor, der an dieses Muster adaptiert wird und somit Aktivierungen für alle Vorkommen dieses Musters in den Daten ausbildet, verringert entsprechend stark die Energie des Residuums. Sobald durch die fortschreitende iterative Minimierung die dominanten Muster hinreichend gut durch Basisvektoren beschrieben werden, werden Signalanteile mit weniger Energie durch verbleibende Basisvektoren erfasst. Mit Erhöhung der Anzahl der Basisvektoren spezialisieren sich immer mehr Basisvektoren auf immer seltenere Muster und verringern damit die Energie des Residuums nur noch wenig.

Die weiter vorn beschriebene Modellvorstellung war aber, dass die Basisvektoren wiederkehrende, charakteristische Muster in den Daten abbilden. Dies ist nur dann gegeben, wenn sich die Basisvektoren nur auf die dominanten Muster in den Daten konzentrieren, was wiederum nach den obigen Ausführungen nur dann der Fall ist, wenn die Anzahl der

Basisvektoren hinreichend beschränkt ist. Ziel ist es also, die Anzahl der Basisvektoren gering zu halten, aber dennoch die Energie des Residuums hinreichend zu minimieren.

Abbildung 4.8 zeigt den Verlauf der relativen Energie des Residuums in Abhängigkeit von der Anzahl der Basisvektoren nach Optimierung für eine kleine Teilmenge der Gestendaten (hier soll nur das Prinzip gezeigt werden). Die relative Energie des Residuums ist ein Maß für die Rekonstruktionsgüte und ist die Energie des Residuums normiert auf die Energie der Eingangsdaten:

$$\overline{F_r} = \frac{\sum_d \|\mathbf{V}^{(d)} - \mathbf{R}^{(d)}\|_2^2}{\sum_d \|\mathbf{V}^{(d)}\|_2^2} \ . \tag{4.8}$$

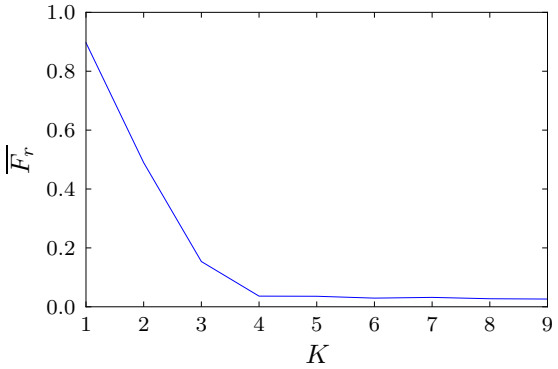


Abb. 4.8.: *Verlauf der relativen Energie des Residuums $\overline{F_r}$ in Abhängigkeit von der Anzahl K der Basisvektoren. Da hier nur das Prinzip gezeigt werden soll, wurde zur Erhöhung der Deutlichkeit der Darstellung während der Berechnung die Datenmenge auf 10 Exemplare einer Gestenklasse (in die Luft gemalter Buchstabe „a“) beschränkt (der gesamte Datensatz ist in Abschnitt 3.3.1 beschrieben). Diese Datenmenge lässt sich bereits mit vier Basisvektoren hinreichend gut beschreiben.*

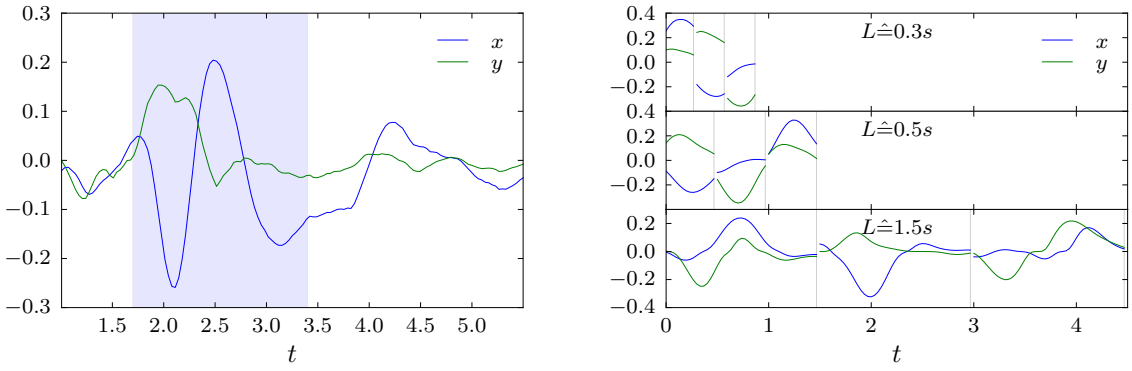
Der Verlauf zeigt, dass die Energie ab $K = 5$ nur noch unwesentlich abnimmt und somit bei $K = 4$ alle wesentlichen Muster in den Daten durch Basisvektoren erfasst sind und eine Erhöhung der Basisvektoren dazu führen würde, dass sich einige Basisvektoren überspezialisieren und keine generellen Muster abbilden. In der praktischen Anwendung hat sich gezeigt, dass eine Verringerung der relativen Energie um weniger als 10% die Hinzunahme eines weiteren Basisvektors nicht mehr rechtfertigt.

Länge der Basisvektoren

Auch die Länge der Basisvektoren muss nach heuristischen Kriterien bestimmt werden. Hier spielt vor allem der Verwendungszweck der Basisvektoren und Aktivierungen eine große Rolle. Geht es darum, größere zusammenhängende Muster zu finden, so müssen die Basisvektoren entsprechend groß gewählt werden. Sollen hingegen Teile von Mustern identifiziert werden, die eventuell sogar als Alphabet zur Komposition größerer Muster

interpretiert werden sollen, wie es in dieser Arbeit angestrebt wird, so müssen die Basisvektoren entsprechend kleiner gewählt werden. Insgesamt hängt also die Länge der Basisvektoren von der Länge der in den Daten zu erwartenden Muster ab.

Abbildung 4.9 zeigt Basisvektoren für drei unterschiedliche Werte der Länge L, die aus dem Geschwindigkeitsverlauf der rechten Hand bei Ausführung einer Geste extrahiert wurden. Für $L = 0{,}3s$ sind die Basisvektoren relativ kurz und es werden nur sehr kleine Merkmale der Daten abgebildet. Ist das Ziel interpretierbare Teile von Gesten in den Daten zu finden, so sind diese Merkmale sicher zu kurz. Für $L = 1{,}5s$ hingegen kann durch einen Basisvektor bereits eine ganze Geste (vgl. Abbildung 4.9a) abgedeckt werden. Die Strukturen, die durch die Basisvektoren abgebildet werden, sind bereits zu sehr auf einzelne Muster spezialisiert. Ein Wert von $L = 0{,}5s$ ist hier ein guter Kompromiss, denn es werden Teile von Mustern mit mittlerer Komplexität abgebildet.



(a) *Ausschnitt des Eingangssignals* (b) *Basisvektoren*

Abb. 4.9.: *Abbildung (b) zeigt Basisvektoren unterschiedlicher Länge L, die aus drei Trainingsläufen aus dem Eingangssignal (a) gelernt wurden. Die Basisvektoren sind aneinander gereiht und durch eine vertikale Linie getrennt. Ein kurzer Ausschnitt des Eingangssignals ist in Abbildung a dargestellt. Der Signalabschnitt, in dem eine Geste ausgeführt wurde, ist blau hinterlegt. Wie zu sehen ist, bilden die Basisvektoren mit zunehmender Länge komplexere Strukturen ab.*

Zuletzt soll noch erwähnt werden, dass es auch Verfahren gibt, welche die Größe der Basisvektoren während der Optimierung anpassen. Obwohl diese Ansätze vielversprechend erscheinen, wurden sie im Rahmen dieser Arbeit nicht weiter untersucht und könnten Gegenstand aufbauender Arbeiten sein.

Spärlichkeitsparameter

Das Ergebnis der Optimierung hängt sehr sensibel von den Spärlichkeitsparametern λ_1 und λ_2 (vgl. Gleichung 4.5) ab. Zudem sind diese beiden Parameter nicht unabhängig voneinander. Neben einer automatischen Parametersuche mittels Grid Search hat sich daher folgende Methode als hilfreich erwiesen. Zuerst wird $\lambda_2 = 0$ und λ_1 auf einen

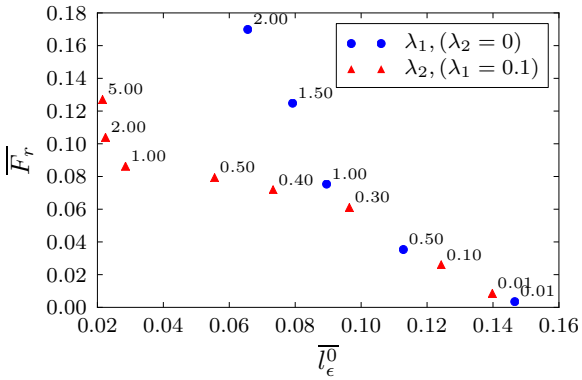


Abb. 4.10.: *Paretofront ohne und mit Verwendung der lokalen Spärlichkeit. Die Hochachse misst die relative Energie des Residuums bei Konvergenz. Die Rechtsachse misst die relative $\overline{l_0^\epsilon}$-Pseudonorm für $\epsilon = 0{,}1$ (siehe Gleichung 4.9). Die blauen Punkte markieren den Verlauf der Paretofront mit reiner globaler Spärlichkeit durch Variation von λ_1 und konstantem $\lambda_2 = 0$. Die roten Dreiecke markieren den Verlauf bei Verwendung lokaler Spärlichkeit durch Variation von λ_2 und konstantem $\lambda_1 = 0{,}1$. Bei reiner globaler Spärlichkeit (blau) lässt sich mit $\lambda_1 = 1{,}0$ eine akzeptable Spärlichkeit von 9% und ein akzeptables Residuum von 8% erreichen. Bei Verwendung der lokalen Spärlichkeit (rot) lässt sich mit $\lambda_2 = 1{,}0$ bei nur geringer Erhöhung des Residuums auf etwa 9% eine wesentlich stärkere Spärlichkeit von 2,5% erreichen.*

minimalen Wert, z. B. $\lambda_1 = 10^{-3}$, gesetzt. Mit diesen beiden Parametern wird die Optimierung bis zur Konvergenz ausgeführt. Nun werden die Aktivierungen visuell inspiziert. Sind die Aktivierungen noch nicht hinreichend spärlich, so wird der Parameter λ_1 sukzessive erhöht und die Optimierung mit jedem Parameter erneut ausgeführt. Sinnvolle Werte sind z. B. $\{1 \cdot 10^{-3}, 3 \cdot 10^{-3}, 7 \cdot 10^{-3}, 1 \cdot 10^{-2}, 3 \cdot 10^{-3}, \ldots, 1, 1{,}5\}$. Wenn ein guter Wert für λ_1 bestimmt wurde, sind in der Regel noch lokale „Verschmierungen“ der Aktivitäten (siehe Abschnitt 4.2.5) vorhanden. Deshalb wird nun er Wert von λ_2 sukzessive erhöht, solange bis die lokalen Verschmierungen hinreichend gut unterdrückt sind. Die oben für λ_1 angegebenen Werte sind auch hier sinnvoll.

Ein weiteres Hilfsmittel zur Wahl der Spärlichkeitsparameter bildet das Verhältnis zwischen Rekonstruktionsgüte und Spärlichkeit der Repräsentation. Die Maximierung der Rekonstruktionsgüte und die Maximierung der Spärlichkeit bilden zwei entgegengesetzte Ziele. Dabei führt eine Verbesserung der einen Zielgröße zu einer Verschlechterung der anderen. Mit Variation der Spärlichkeitsparameter kann man das Verhältnis der beiden Größen beeinflussen. Abbildung 4.10 zeigt die Paretofronten zwischen der Rekonstruktionsgüte und der Spärlichkeit, zum einen bei Variation nur der globalen Spärlichkeit und ohne Einsatz der lokalen Spärlichkeit und zum anderen bei Variation der lokalen Spärlichkeit. Die Rekonstruktionsgüte wird dabei durch die relative Energie des Residuums (siehe Gleichung 4.8) gemessen.

Die Spärlichkeit wird durch die relative Anzahl der Aktivitäten, deren Wert über einer

gewissen Schwelle liegt, gemessen und ist wie folgt definiert

$$\overline{l_0^\epsilon} = \frac{|\{(m,n,k) : H_{k,n}^{(m)} > \epsilon\}|}{MNK} \,. \tag{4.9}$$

Bei Einsatz nur der globalen Spärlichkeit ergibt sich z. B. bei $\lambda_1 = 1{,}0$ ein guter Kompromiss. Dabei beträgt die relative Energie des Residuums noch 8% und es sind etwa 9% aller Aktivierungen aktiv. Setzt man allerdings die lokale Spärlichkeit mit $\lambda_2 = 1{,}0$ ein, so kann unter unwesentlicher Erhöhung der relativen Energie des Residuums eine weitere Reduktion des Anteils der aktiven Aktivierungen auf etwa 2,5% erreicht werden.

4.6. Eigenschaften der spärlichen Repräsentation

Im Folgenden wird beispielhaft die Anwendung des Sparse Coding auf reale Daten betrachtet. Dabei sollen einige Eigenschaften der entstehenden Repräsentation näher betrachtet werden, die vor allem auch für nachgelagerte Verarbeitungsschritte, wie sie in späteren Kapiteln beschrieben sind, von Bedeutung sind.

Entstehung interpretierbarer Basisvektoren

Abbildung 4.11 (oben) zeigt einen Ausschnitt des Eingangssignals. Aus diesem Signal wurden mittels der Shift-NMF Basisvektoren und Aktivitäten gelernt. Dabei wurden die Spärlichkeitsparameter zu $\lambda_1 = 1{,}0$, $\lambda_2 = 10{,}0$ gewählt. Die Anzahl der Basisvektoren wurde zu $K = 12$ gewählt. Ihre Länge zu $L = 15$ Samples, was bei der Abtastrate von 30Hz 0,5s entspricht. Abbildung 4.11 zeigt die Rekonstruktion des Signals (Mitte) und die Aktivierungen (unten). Abbildung 4.12 zeigt die gelernten Basisvektoren.

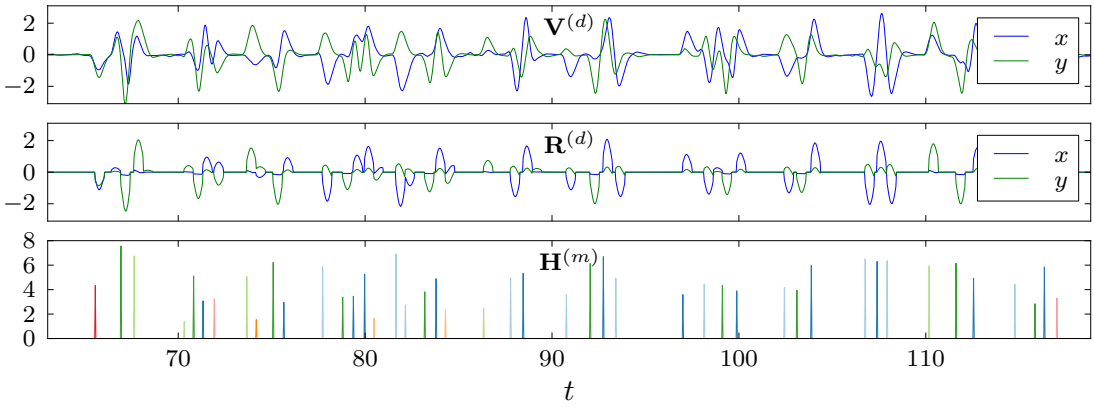
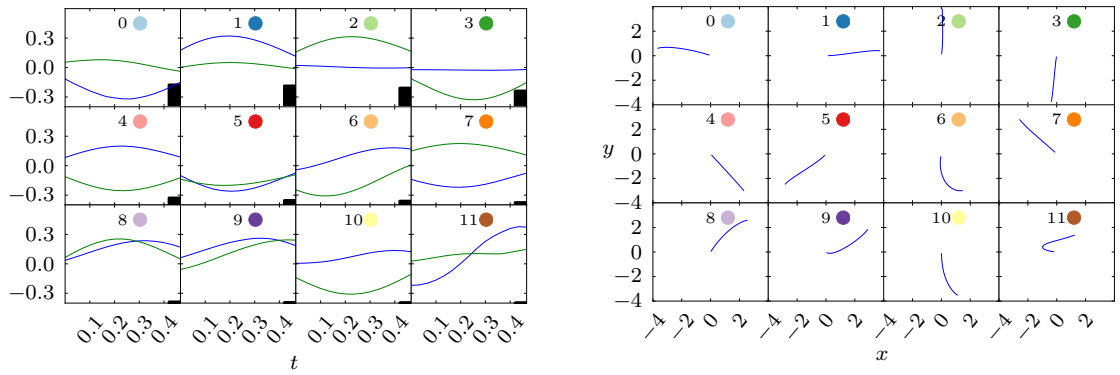


Abb. 4.11.: *Eingangssignal (oben), Rekonstruktion (Mitte) und Aktivierungen (unten). Die Aktivierungen sind farblich codiert. Abbildung (4.12) zeigt die zugehörigen Basisvektoren. Die Farbe des korrespondierenden Basisvektors ist in Abbildung (4.12) durch einen entsprechend gefärbten Punkt markiert.*



(a) *Geschwindigkeiten* (b) *Position*

Abb. 4.12.: *Basisvektoren (a) im Geschwindigkeitsraum und im Positionsraum (b). Die Darstellung im Positionsraum wurde durch Integration der Basisvektoren entlang der Zeitachse erzeugt und dient nur der Veranschaulichung. Die Farbe des Punktes entspricht der Farbe der Aktivierungen in Abbildung 4.11. Der relative Beitrag zur Rekonstruktion eines Basisvektors ist durch den schwarzen Balken in der jeweiligen Gitterzelle markiert (normiert auf die Höhe der Gitterzelle). Die Basisvektoren wurden nach ihrem relativen Beitrag zur Rekonstruktion geordnet.*

Es werden nun die Eigenschaften der gelernten Basisvektoren in Abbildung 4.12 betrachtet. Mit der gewählten Anzahl der Basisvektoren von $K = 12$ sind 73% der Varianz der Daten abgedeckt (die relative Energie des Residuums beträgt 17%). Die Länge der Basisvektoren wurde mit $0{,}5s$ so gewählt, dass sie charakteristische Merkmale abdecken. Wie in Abbildung 4.12b gezeigt ist, bilden die Basisvektoren im Wesentlichen Strichzüge in die vier Hauptrichtungen, sowie in die diagonalen Richtungen ab. Wie an den Balken in Abbildung 4.12a zu sehen ist, kann durch die ersten vier Basisvektoren, welche die Züge in die vier Hauptrichtungen beschreiben, schon der größte Teil der Daten rekonstruiert werden.

Wie an den Aktivierungen in Abbildung 4.11 (unten) zu sehen ist, werden die Basisvektoren zu diskreten Zeitpunkten aktiviert. Durch den Einsatz der lokalen Spärlichkeit treten keine Verschmierungen der Aktivierungen auf. An der Rekonstruktion in Abbildung 4.11 (Mitte) ist erkennbar, dass durch die diskrete Aktivierung der Basisvektoren das Eingangssignal zwar nicht perfekt rekonstruiert wird, aber wesentliche Merkmale des Signals erhalten bleiben.

Klassenübergreifende Verwendung von Basisvektoren

In diesem Abschnitt soll gezeigt werden, dass Basisvektoren als Alphabet betrachtet werden können, welches den Eingangsdaten zugrunde liegt. Dazu soll die Verwendung der Basisvektoren in unterschiedlichen Gesten untersucht werden. In den oben vorgestellten Eingangsdaten werden 26 unterschiedliche Klassen von Gesten, entsprechend

den 26 Buchstaben des Buchstabenalphabetes, ausgeführt. Abbildung 4.13 illustriert die Verwendung der Basisvektoren in den verschiedenen Klassen.

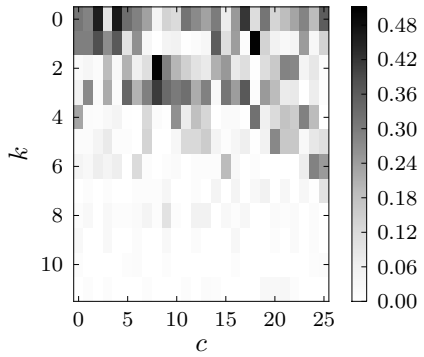


Abb. 4.13.: *Verwendung der Basisvektoren in den unterschiedlichen Klassen von Gesten. Als Datensatz wurde der im Rahmen dieser Arbeit erstellte Gestendatensatz, bestehend aus 26 Klassen in die Luft gemalter Buchstaben, gewählt (siehe Abschnitt 3.3.1). Der Index k zählt die Basisvektoren, während c die Klassen zählt. Ein Wert von* 0,5 *in der k-ten Zeile und c-ten Spalte bedeutet, dass in* 50% *der Exemplare der k-ten Klasse der c-te Basisvektor enthalten ist.*

Die Werte in dem Gitter geben die relative Häufigkeit an, mit der ein Basisvektor k in einer gegebenen Klasse c aktiv war. Ein Basisvektor gilt dann als aktiv, wenn seine Aktivierung einen Wert von 0,1 überschreitet. An dieser Grafik ist zu erkennen, dass die Basisvektoren 0 bis 3 über viele Klassen hinweg relativ häufig verwendet werden. Die Basisvektoren 4 bis 6 werden schon in etwas weniger Klassen intensiv verwendet, während die restlichen Basisvektoren zwar immer noch über viele Klassen hinweg verwendet werden, aber nur noch von wenigen Exemplaren pro Klasse. Basisvektoren werden also klassenübergreifend verwendet, was ihre Interpretation als Basisalphabet der Gesten unterstützt.

4.7. Fazit

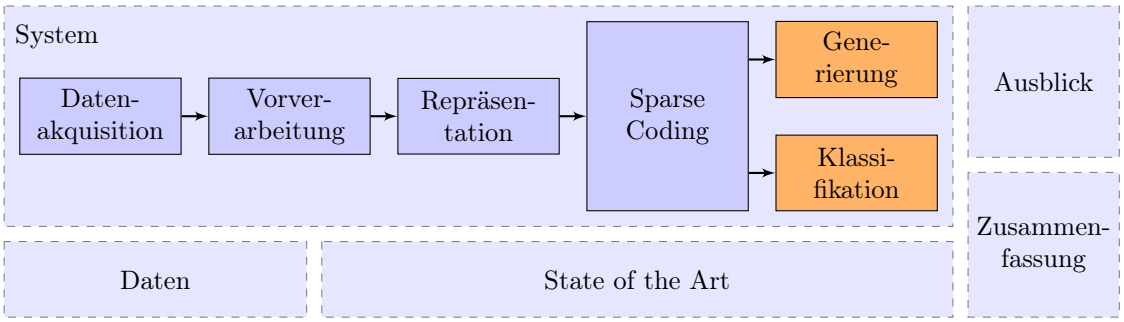
In diesem Kapitel wurde das Sparse Coding als Kernthema der Arbeit vorgestellt. Nach einer Diskussion der Motivation des Kapitels und Betrachtung des Ansatzes unter verschiedenen Blickwinkeln, wurde das Optimierungsproblem schrittweise hergeleitet. Danach wurden Algorithmen zur Lösung des Problems vorgestellt und ihre Anwendung an künstlichen sowie realen Daten gezeigt. Zuletzt wurde auf die Wahl der Parameter in der praktischen Anwendung des Verfahrens eingegangen.

Es wurde gezeigt, dass mit Hilfe des hier entwickelten Verfahrens Primitive aus einer Menge an Bewegungsdaten gelernt werden können, die als Alphabet einfacher Bewegungen interpretiert werden können, aus denen eine komplexe Bewegung zusammengesetzt ist. Diese Dekomposition einer Bewegungstrajektorie und die Repräsentation als

einfache Folge von Aktivierungen der Primitive macht eine Anwendung einfacher Verfahren in nachgelagerten Verarbeitungsschritten möglich. Im nächsten Kapitel wird die Nutzung dieser Eigenschaften zur Generierung und Klassifikation von Bewegungsdaten vorgestellt.

Kapitel 5

Generierung und Klassifikation von Bewegungstrajektorien mittels Sparse Coding



In diesem Kapitel sollen die Vorteile des Sparse Coding als Merkmalsextraktionsschritt in einem Mustererkennungsprozess für nachgelagerte Verarbeitungsschritte betrachtet werden. Im Rahmen dieser Arbeit wurden dazu zwei typische Aufgaben im Bereich der Verarbeitung von Bewegungstrajektorien betrachtet, die Generierung von Trajektorien und die Klassifikation beobachteter Trajektorien.

Die Generierung neuer Trajektorien hat ihren Anwendungsbereich vor allem in der mobilen Robotik, in der automatischen Manipulation und in der Prädiktion. Die Aufgabe besteht hier darin, nach Vorgabe einer bestimmten Klasse von Trajektorien, ein neues Exemplar dieser Klasse zu erzeugen. Das kann z. B. in der mobilen Robotik eine bestimmte Form von Ausweichbewegung während der Fahrt sein oder in der automatischen Manipulation eine bestimmte charakteristische Teilbewegung zur Manipulation eines Objektes.

Die Aufgabe bei der Klassifikation ist, wie schon im Abschnitt 3.2 beschrieben, einer beobachteten, also unbekannten, Trajektorie eine Klassenzugehörigkeit zuzuordnen. Dies ist der finale Schritt z. B. bei der Gestenerkennung.

Obwohl die Definition des Mustererkennungsprozesses im engeren Sinne in dieser Arbeit bisher nur die Klassifikation umfasste, kann die Generierung durch Erweiterungen

des letzten Verarbeitungsschrittes auch mit diesem Prozess beschrieben werden. Dabei spielt die Merkmalsextraktion für beide Anwendungsbereiche eine wesentliche Rolle.

Im Folgenden wird zuerst die Generierung auf Basis des Sparse Coding vorgestellt. Dabei werden komplexe Trajektorien durch zeitgesteuerte Aktivierung von Primitiven erzeugt. Zur Aktivierung der Primitiven wird ein neuronaler Ansatz vorgestellt.

Danach wird auf die Klassifikation eingegangen. Dort wird gezeigt, dass das Sparse Coding auch als ein Ansatz zum Lernen von Merkmalen betrachtet werden kann, mit deren Hilfe eine Trajektorie durch einen kompakten Merkmalsvektor beschrieben werden kann. Diese Darstellung macht die Verwendung sehr einfacher Standardmodelle für die Klassifikation möglich. Es werden zwei Experimente vorgestellt. Das erste Experiment betrachtet das Verfahren im Kontext der isolierten Klassifikation mit dem in dieser Arbeit selbst erstellten Datensatz (siehe Abschnitt 3.3.1). Das zweite Experiment betrachtet das Verfahren im Kontext der kontinuierlichen Klassifikation anhand einer Reihe von Datensätzen aus dem Bereich der Activity-Recognition und zeigt, dass es auch im Vergleich mit anderen Merkmalsextraktionsverfahren sehr gut abschneidet.

Im Rahmen der Klassifikation wird weiterhin auf ein Konzept namens Self-taught-Learning eingegangen, bei dem die Leistung des Klassifikators durch Training auf zusätzlichen ungelabelten Daten verbessert werden kann.

Die Inhalte dieses Kapitels wurden im Wesentlichen in [Vollmer et al., 2012b], [Vollmer et al., 2012a] und [Vollmer, Gross et al., 2013] veröffentlicht.

5.1. Generierung klassentypischer Trajektorien durch Aktivierung von Bewegungsprimitiven

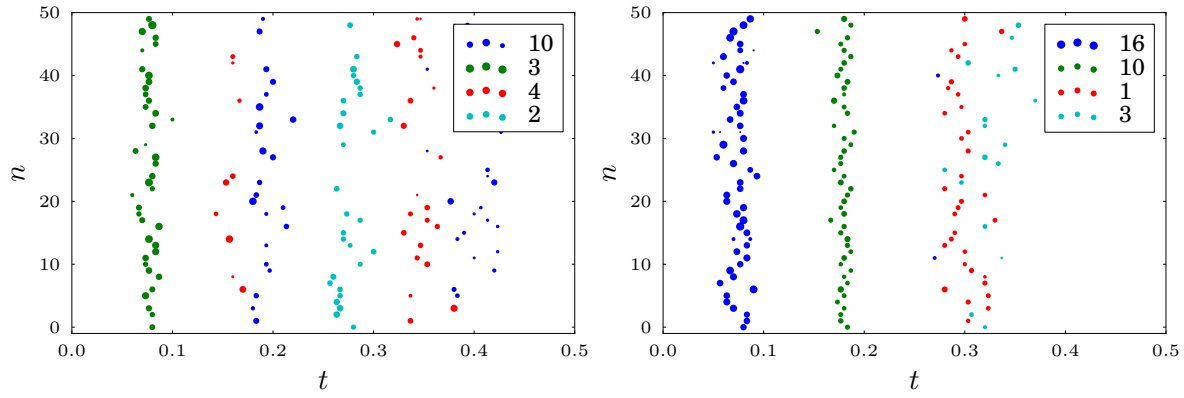
Die Idee, Bewegungstrajektorien mittels Aktivierung von Primitiven zu erzeugen, ist durch Beobachtungen der Analyse der Bewegungserzeugung bei Lebewesen inspiriert. Entsprechende Studien [E Bizzi et al., 1995; D'Avella, Saltiel et al., 2003] konnten zeigen, dass die motorische Steuerung aus einer Hierarchie mit zwei Ebenen besteht, wobei in der unteren Ebene Motorprimitive aktiviert werden, die zu einer bestimmten Sequenz an Muskelkontraktionen führen, um einfache Bewegungen zu formen. Die obere Ebene steuert die sequentielle Aktivierung der Motorprimitive, um komplexe Bewegungen auszuführen. Sie realisiert damit einen Plan zur Bewegungsausführung. Weiterhin konnte gezeigt werden, dass einzelne Motorprimitive für eine Vielzahl komplexer Bewegungen verwendet werden und somit ein Basisalphabet der komplexen Bewegungen bilden [D'Avella und Emilio Bizzi, 2005].

Da die mittels Sparse Coding aus den Daten gelernten Basisvektoren ebenfalls als Primitive betrachtet werden können (siehe Kapitel 4), liegt es nahe, diese für die Generierung von Trajektorien zu verwenden. Die untere Ebene der oben beschriebenen Hierarchie wird dann bereits durch die gelernten Basisvektoren implementiert. Dabei kann ein Basisvektor als eine Primitive interpretiert werden, die eine Sequenz von Punkten

im Raum (z. B. der Raum der Positionen der rechten Hand oder der Raum der Geschwindigkeiten) beschreibt. Aktiviert wird diese Primitive durch die Aktivierung des Basisvektors.

Lernt man aus einer Menge an Trajektorien mittels Sparse Coding Basisvektoren und deren Aktivierungen, so lässt sich beobachten, dass für ähnliche Trajektorien die Aktivierungen wiederum ähnlich sind. Im Folgenden werden dazu Ergebnisse der Anwendung von Sparse Coding auf einen Datensatz bestehend aus Handschrift-Trajektorien, den sogenannten „Character Trajectories Dataset“, gezeigt. Eine Beschreibung des Datensatzes befindet sich in Abschnitt 3.3.2.

Abbildung 5.1 zeigt die Aktivierungen der vier Basisvektoren mit der höchsten durchschnittlichen Aktivierung über alle Exemplare der Klasse für die Buchstabenklassen „a“ und „b“ [1]. Die Aktivierungen der anderen Basisvektoren wurden aus Gründen der Übersichtlichkeit nicht mit abgebildet. Anhand der Grafik ist erkennbar, dass die Aktivierungsmuster über alle Exemplare einer Klasse größtenteils konsistent sind.



(a) *Klasse „a“* (b) *Klasse „b“*

Abb. 5.1.: *Aktivierungen von je 50 Exemplaren der Buchstabenklassen „a“ und „b“ für die vier Basisvektoren mit der höchsten durchschnittlichen Aktivierung pro Klasse. Die Hochachse zählt die präsentierten Beispiele und die Längsachse die Zeit. Eine Aktivierung ist mit einem Punkt gekennzeichnet. Die Größe der Punkte skaliert mit der Stärke der Aktivierungen. Die Stärken der Aktivierungen wurden zuvor auf das Einheitsintervall normiert.*

Wenn es gelingt, dieses klassentypische Aktivierungsmuster mittels eines generativen Modells zu beschreiben, so kann man bei gegebener Klasse ein neues Exemplar der Klasse durch Erzeugung eines Aktivierungsmusters mittels dieses Modells und darauffolgender Aktivierung der Basisvektoren entsprechend des Musters generieren. Im Folgenden soll ein Modell vorgestellt werden, welches die klassentypischen Aktivierungsmuster

[1] Die Parameter beim Lernen wurden wie folgt gewählt: Anzahl der Basisvektoren $K = 20$, Länge der Basisvektoren $L = 20$ (entspricht bei einer Abtastrate von $200Hz$ $0{,}1s$), Spärlichkeitsparameter $\lambda_1 = 0{,}5, \lambda_2 = 20$.

modellieren kann und damit die obere Schicht der Hierarchie zur Bewegungssteuerung realisiert.

Für die Behandlung der dargestellten Problemstellung sind zwei Klassen von Verfahren denkbar. Das sind zum einen probabilistische Verfahren, welche die temporale Komponente explizit modellieren, wie z. B. das State-Duration-HMM [Rabiner, 1989], die um eine temporale Komponente erweiterte Variante des Hidden-Markov-Modells. Das Modell und die Implementierung des Lernalgorithmus sind allerdings recht komplex und es steht keine frei verwendbare Implementierung zur Verfügung. Die zweite Klasse sind die neuronalen Verfahren, welche einen zeitlichen Generierungsprozess modellieren können, z.B. das Leaky-Integrate-and-Fire-Neuronenmodell (LIF). Das LIF wird im Wesentlichen durch eine Intensitätsmatrix (eine einfache Statistik der Aktivierungsmuster einer Klasse) parametrisiert und ist damit wesentlich einfacher zu lernen als das State-Duration-HMM. Aufgrund der einfachen Verwendbarkeit wird dieses Modell im Folgenden zur Generierung von Aktivierungen verwendet.

Die Statistik der Aktivierungsmuster einer Klasse wird dabei in einer klassenspezifischen Intensitätsmatrix festgehalten. Dazu werden die Klassen mit $c \in \{1, \ldots, C\}$ nummeriert und für jede Klasse c eine Intensitätsmatrix $\mathbf{I}^{(c)} \in [0,1]^{P \times K}$ mit

$$\mathbf{I}^{(c)}_{p,k} = \frac{1}{N_c} \sum_{n=1}^{N_c} \Theta_\epsilon \left(H^{(p)}_{k,n} \right)$$

definiert, wobei N_c die Anzahl der Exemplare in Klasse c zählt und Θ_ϵ die binäre Schwellwertfunktion (Heavyside-Funktion) mit einer Schwelle ϵ ist. Ein Element $\mathbf{I}^{(c)}_{p,k}$ zählt die relative Häufigkeit der Aktivierung des Basisvektors k zum diskreten Zeitpunkt p. Diese Statistik ignoriert zunächst die Stärke der Aktivierung, doch darauf wird später eingegangen. In Abbildung 5.2a (blaue Kurve) sind die Werte der Intensitätsmatrix für die Klasse „a“ visualisiert.

Die zeitlichen Variationen innerhalb einer Klasse (siehe Abbildung 5.1) erschweren die Berechnung einer Intensitätsmatrix, weil sie dazu führen, dass die Intensitäten über die Zeit verteilt sind. Diese Variationen lassen sich mit Hilfe eines Optimierungsverfahrens aus den Aktivierungsmustern heraus rechnen. Dazu wird mit jedem Exemplar einer Klasse ein Paar von Parametern (a_n, b_n) assoziiert, wobei mit $a_n \in \mathbb{R}$ die Verschiebung des Exemplars und mit $b_n \in \mathbb{R}$ eine Streckung des Exemplars notiert wird. Diese beiden Parameter werden über allen Exemplaren der Klasse simultan optimiert. Dazu wird eine Zielfunktion iterativ maximiert, welche die Korrelation einer linearen Interpolation der Aktivierungen aller Exemplare unter Berücksichtigung ihrer aktuellen (bzgl. des Iterationsschrittes) Verschiebung und Streckung misst. Der Algorithmus ist detailliert in Anhang D.1 beschrieben. Die Intensitätsmatrix, die sich nach Optimierung der Verschiebungs- und Streckungsparameter ergibt, wird $\widehat{\mathbf{I}}^{(c)}$ genannt. Abbildung 5.2a (links, rote Kurve) visualisiert die optimierte Intensitätsmatrix. Im Vergleich zur nicht optimierten Intensitätsmatrix (blaue Kurve) sind die Intensitäten stärker an be-

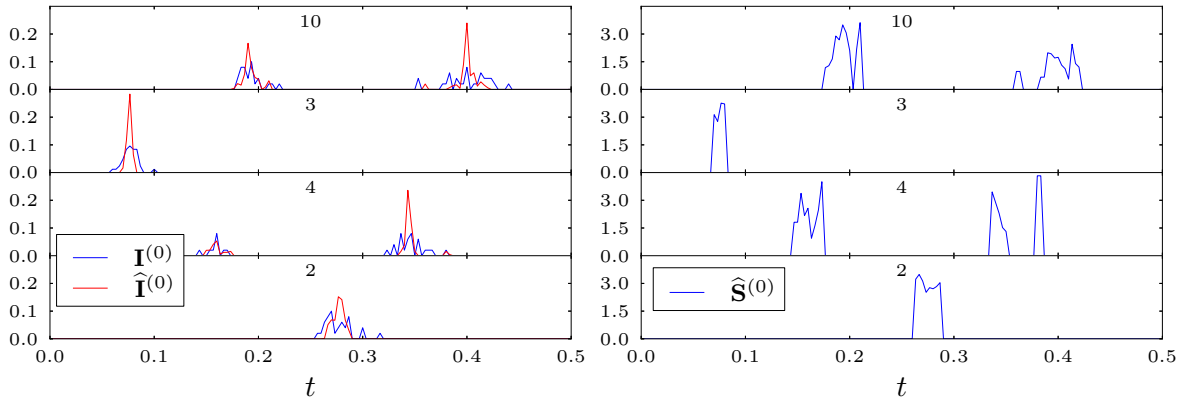


Abb. 5.2.: *Darstellung der zur Generierung mit dem Leaky-Integrate-And-Fire-Neuronenmodell (LIF) verwendeten Intensitäts- und Skalierungsmatrizen. Abbildung (a) zeigt die originale Intensität* $\mathbf{I}^{(0)}$ *(blau) und die durch Optimierung der Verschiebungs- und Dehnungsparameter berechnete Intensität* $\widehat{\mathbf{I}}^{(0)}$ *(rot) im Vergleich beispielhaft für die Basisvektoren 10, 3, 4 und 2. Abbildung (b) zeigt die ausgerichtete Skalierungsmatrix* $\widehat{\mathbf{S}}^{(c)}$ *zur Skalierung der aktivierten Basisvektoren.*

stimmten Zeitpunkten konzentriert und charakterisieren das mittlere klassenspezifische Aktivierungsmuster besser.

Bisher wurde die Skalierung der Aktivierungen außen vor gelassen. Die Information über die Skalierungen wird nun in einer weiteren Matrix $\mathbf{S}^{(c)}$ mit

$$\mathbf{S}_{p,k}^{(c)} = \frac{1}{N_c I_{p,k}^{(c)}} \sum_{n=1}^{N_c} H_{k,n}^{(p)}$$

erfasst. Entsprechend wird auch hier, mit Hilfe der durch Optimierung ermittelten Verschiebungs- und Streckungsparameter, eine optimierte Skalierungsmatrix $\widehat{\mathbf{S}}^{(c)}$ bestimmt. Abbildung 5.2b zeigt die Skalierungsmatrix für die Klasse „a“.

Das Paar $\left(\widehat{\mathbf{I}}^{(c)}, \widehat{\mathbf{S}}^{(c)}\right)$ beschreibt nun eine Statistik der Aktivierungsmuster und die Skalierung der aktivierten Basisvektoren in Klasse c.

Zur eigentlichen Generierung eines neuen Exemplars einer Klasse wird nun der Verlauf der Intensität eines Basisvektors als Eingabe für ein LIF-Neuron verwendet. Das LIF-Neuron akkumuliert das eingehende Signal solange, bis eine Schwelle erreicht wird. Dann löst es als Ausgabe einen Spike aus und entlädt die akkumulierte Energie. Das Akkumulieren des Eingangssignals kann durch einen verlustbehafteten Integrator beschrieben werden. Dazu wird nun der Zustand eines Neurons, welches die Aktivierungen

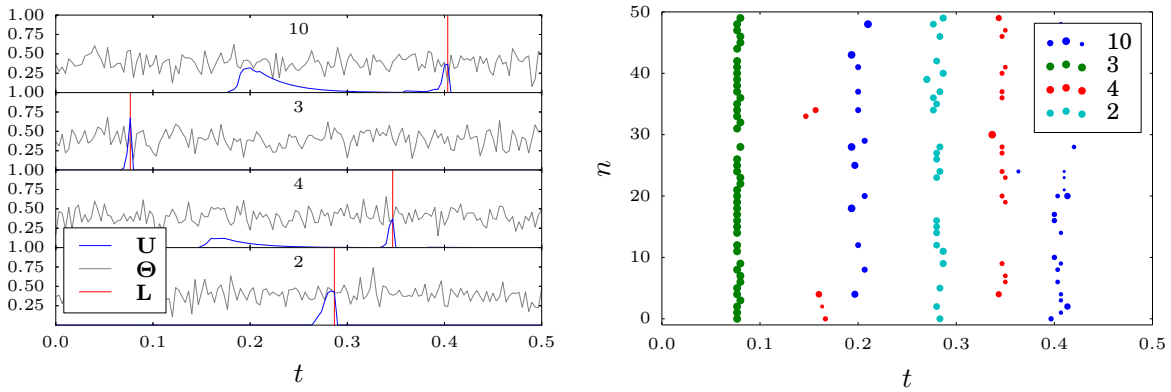
für Basisvektor k modelliert, zum Zeitpunkt p als $U_{p,k}$ eingeführt:

$$U_{p,k} = \begin{cases} (1-\nu)U_{p-1,k} + \widehat{I}_{p,k} & \text{, falls } p - p' \geq \delta t_{ref} \\ 0 & \text{, falls } p - p' < \delta t_{ref} \end{cases},$$

Der Zustand zum Zeitpunkt $p = 0$ wird mit $U_{0,k} = 0$ definiert. p' bezeichnet den Zeitpunkt des letzten Spikes. Wurde innerhalb einer Refraktärzeit δt_{ref} kein Spike erzeugt (erster Fall), so wird die Intensität des Basisvektors k abzüglich eines Verlustterms, welcher durch den Parameter $\nu \in (0,1)$ gesteuert wird, akkumuliert. Liegt der Zeitpunkt des letzten Spikes noch innerhalb der Refraktärzeit δt_{ref} (zweiter Fall), so verbleibt die Energie auf 0. Die Refraktärzeit sichert die Spärlichkeit der Spikes.

Diesem Akkumulator nachgeschaltet ist eine Schwellwertoperation, die bei Erreichen einer zeitabhängigen, verrauschten Schwelle einen Spike erzeugt. Um die bisherige Notation konsistent zu halten, werden die Zustände aller Neuronen zu allen Zeitpunkten als Matrix $\mathbf{U} \in \mathbb{R}^{P \times K}$ zusammengefasst. Die verrauschten Schwellen für alle Neuronen werden mit einer Matrix $\mathbf{\Theta} \in \mathbb{R}^{P \times K}$ mit $\Theta_{p,k} \sim \mathcal{N}(\mu_\theta, \sigma_\theta)$ notiert. Die erzeugten Spikes werden als Matrix $\mathbf{L} \in \{0,1\}^{P \times K}$ notiert.

Abbildung 5.3a zeigt die Werte der beteiligten Variablen während der Generierung von Spikes. Wie zu erkennen ist, wird ein Spike erzeugt, wenn die Ladung die zufällige Schwelle erreicht. Zum gleichen Zeitpunkt sinkt die Ladung auf 0. Abbildung 5.3b zeigt die Erzeugung von Spikes für 50 Exemplare. [2]



(a) (b)

Abb. 5.3.: *Generierung von Spikes mit einem Integrate-and-Fire-Modell. Erreicht die Ladung* $\mathbf{U}$ *eines Neurons die Schwelle* $\mathbf{\Theta}$*, so wird ein Spike* $\mathbf{L}$ *erzeugt und die Ladung entlädt sich auf 0. Abbildung a zeigt die Generierung für ein Exemplar der Klasse „a“. Abbildung b zeigt die generierten Spikes für 50 Exemplare (vgl. Abbildung 5.1a). Die Spikes wurden zusätzlich noch mit der Skalierungsmatrix* $\widehat{\mathbf{S}}^{(0)}$ *skaliert.*

Die Aktivierungsmuster sind sehr ähnlich zu denen der originalen Daten (vgl. Abbil-

[2] Die Parameter zur Erzeugung dieser Muster wurden wie folgt gewählt: $\nu = 0{,}1$, $\delta t_{ref} = 50ms$

dung 5.1a). Diese Ergebnisse zeigen, dass das Modell in der Lage ist, aus einer gegebenen Statistik eines klassentypischen Aktivierungsmusters neue Aktivierungsmuster zu erzeugen.

Um letztendlich eine vollständige Trajektorie zu erzeugen, müssen die generierten Spikes entsprechend der klassenspezifischen Skalierungsmatrix $\widehat{\mathbf{S}}^{(c)}$ skaliert und die Basisvektoren entsprechend aktiviert werden. Da die generierten und skalierten Spikes als Aktivierungen interpretiert werden, kann die Aktivierung entsprechend der Rekonstruktionsformel durch eine Faltung mit den Basisvektoren realisiert werden

$$\mathbf{r}^{(d)} = \sum_k W^{(d)}_{:,k} * \left(\widehat{S}^{(c)}_{:,k} \odot L^{(c)}_{:,k} \right) \ ,$$

wobei $\mathbf{r}^{(d)}$ Kanal d der erzeugten Trajektorie ist und $\odot$ die elementweise Multiplikation notiert.

Mit Hilfe des oben vorgestellten Verfahrens werden nun im Folgenden die Aktivierungsmuster für eine Menge von Buchstabenklassen modelliert. Weiterhin werden für jede Klasse eine Menge neuer Buchstaben generiert. Dies wird auf Buchstaben beschränkt, die nur aus einem einzigen Strichzug bestehen. Dazu gehören z.B. die Buchstaben „a“ und „b“, nicht aber z. B. „x“. Abbildung 5.4 zeigt je 5 Exemplare der generierten Buchstaben pro Klasse zusammen mit je einem Trainingsbeispiel zum Vergleich. Die Variationen zwischen den erzeugten Exemplaren einer Klasse sind durch die Stochastizität beim Sampling der Aktivierungen mit Hilfe des I&F-Neurons für die jeweiligen Basisvektoren begründet. Dadurch variieren die Aktivierungszeitpunkte und die Aktivierungsstärke zum jeweiligen Zeitpunkt.

Die Ergebnisse zeigen, dass es möglich ist, mit dieser Methode typische Aktivierungsmuster für Klassen ähnlicher Trajektorien zu modellieren. Mit Hilfe des I&F-Modells ist es weiterhin möglich, klassentypische Aktivierungsmuster zu generieren und durch entsprechende Aktivierung der Basisvektoren kontinuierliche Trajektorien zu erzeugen.

Die erzeugten Trajektorien weisen allerdings nur geringe zeitliche Variationen auf. Die Ursache dafür ist, dass, wie oben beschrieben, zur Berechnung der Intensitätsmatrix genau diese Variationen aus den Trainingsdaten heraus gerechnet wurden und somit nicht im Modell abgebildet werden. Man könnte das Modell erweitern, indem man hier eine Verteilung der die Variationen beschreibenden Parameter a_n und b_n mitführt und diese während der Generierung verwendet, um Verschiebungen und Streckungen zu erzeugen.

5.2. Klassifikation spärlich codierter Trajektorien

In Abschnitt 5.1 wurde gezeigt, dass ähnliche Trajektorien ähnliche Aktivierungsmuster aufweisen. Die Ähnlichkeit wird also von der Repräsentation im Eingaberaum auf die Repräsentation im Raum der Aktivierungen übertragen. Aufgrund dieser Erhaltung der Ähnlichkeit kann eine Klassifikation von Trajektorien auch auf Basis der Aktivierungen

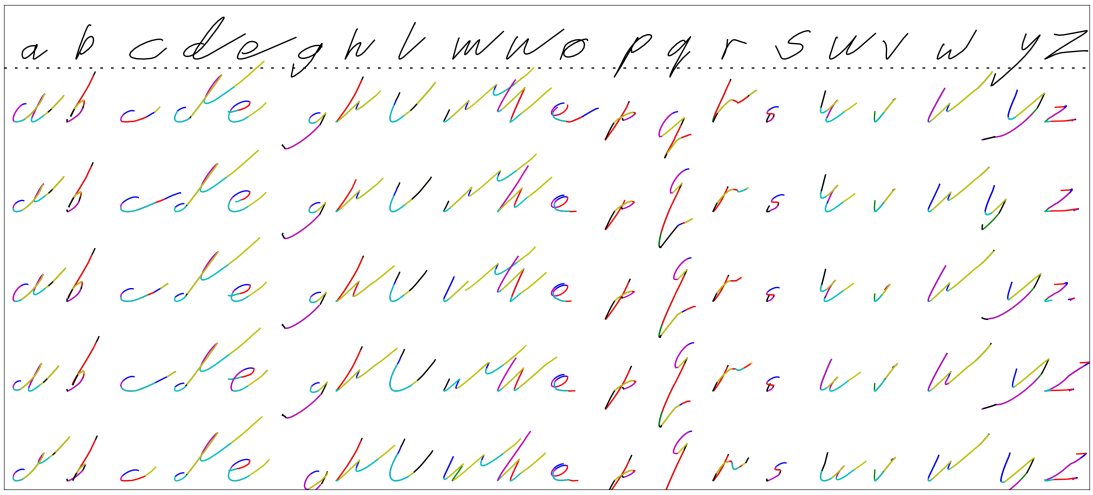
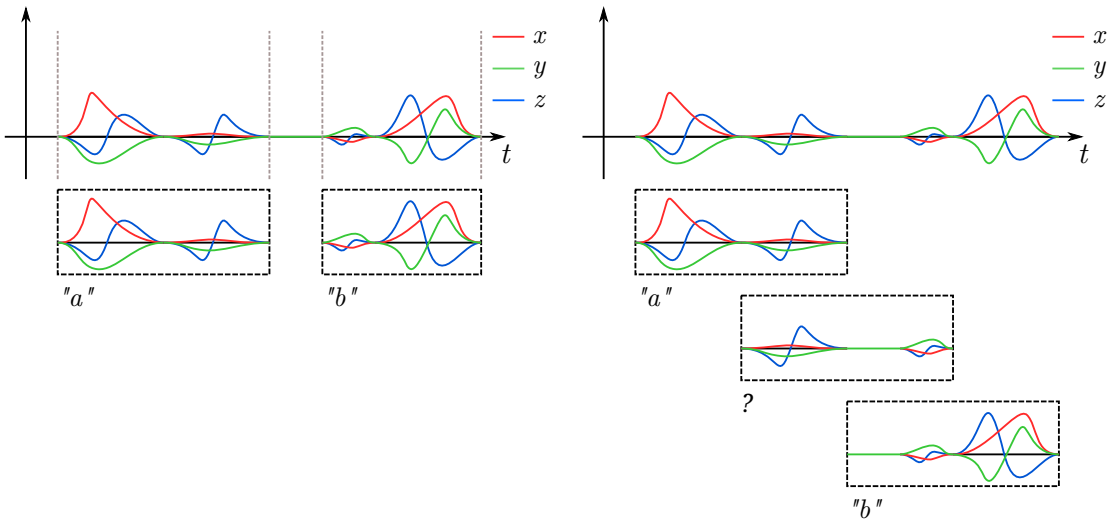


Abb. 5.4.: *Die erste Zeile zeigt pro Klasse je ein Exemplar aus der Menge der Trainingsbeispiele. Darunter sind je fünf generierte Buchstaben pro Klasse angeordnet. Die unterschiedlichen Basisvektoren sind farblich markiert.*

erfolgen. Die Vorverarbeitung mittels Sparse Coding wird in diesem Kontext auch als eine Merkmalsextraktion interpretiert, welche die Daten mittels einer Detektion spärlicher Merkmale, die gelernt und damit an die Daten angepasst sind, reduziert und die wesentlichen Charakteristika in einer sehr kompakten Form repräsentiert. Durch diese reduzierte Darstellung können die Modelle zur Klassifikation sehr einfach gehalten werden. Im Folgenden wird ein solches Modell zur Klassifikation von Trajektorien auf Basis ihrer Aktivierungen vorgestellt.

Bei der Klassifikation von Zeitreihen in einem Echtzeit-Szenario, wie der anfangs beschriebenem Gestenerkennung, gibt es grundsätzlich zwei Ansätze. Der erste Ansatz (siehe Abbildung 5.5a) besteht darin, das kontinuierlich fortlaufende Signal durch einen Vorverarbeitungsprozess zu segmentieren, also bei einer vorgeführten Geste durch eine Heuristik Beginn und Ende der Vorführung zu bestimmen und dem Klassifikator zur genauen Bestimmung der Geste genau den Teil der Zeitreihe vom Beginn bis zum Ende der Geste zu übergeben. Im Folgenden wird dieses Szenario als „isolierte Erkennung“ bezeichnet, da dem Klassifikator immer die gesamte Geste, isoliert vom Kontext, präsentiert wird (siehe dazu auch Abschnitt 2.3).

Das zweite und für den Klassifikator anspruchsvollere Szenario ist die „kontinuierliche Erkennung“ (siehe Abbildung 5.5b). Dabei wird in einem Sliding-Window-Ansatz ein „Fenster“ fester Länge über das kontinuierliche Signal geschoben. In jedem Zeitschritt wird das Fenster um eine feste Distanz, z.B. genau ein Sample, verschoben. Zu jedem Zeitpunkt wird dem Klassifikator das letzte Fenster präsentiert. Das Szenario ist deshalb anspruchsvoller, weil dem Klassifikator, z. B. im Fall der Gestenerkennung, auch Fenster mit unvollständigen Gesten präsentiert werden.



(a) *Isolierte Erkennung* (b) *Kontinuierliche Erkennung*

Abb. 5.5.: *Schema der isolierten Erkennung (a) und der kontinuierlichen Erkennung (b) je eines Exemplars der Buchstabenklassen „a“ und „b“. Bei der isolierten Erkennung (a) werden dem Klassifikator segmentierte Gesten übergeben. Der Klassifikator ordnet dem Segment dann eine Klasse zu. Bei der kontinuierlichen Erkennung (b) werden dem Klassifikator überlappende Fenster übergeben, die auch unvollständige oder gar keine Gesten enthalten können.*

Für alle folgenden Betrachtungen wird angenommen, dass, wie im Kapitel 4 vorgestellt, $\mathbf{V}^{(d)} \in \mathbb{R}^{P \times N}$ mit $d \in 1, \ldots, D$ einen Datensatz von N D-kanaligen Zeitreihen darstellt, die entweder, im Fall der isolierten Erkennung, jeweils ganze Gesten enthalten, oder, im Fall der kontinuierlichen Erkennung, Fenster aus dem Sliding-Window-Ansatz darstellen. Im Fall der isolierten Erkennung ist P die Länge des längsten Exemplars und alle kürzeren Exemplare werden bis zur Länge P mit Nullen aufgefüllt. Im Fall der kontinuierlichen Erkennung entspricht P der Fensterlänge. Segmente und Fenster werden im Folgenden allgemein als Exemplare bezeichnet.

Auf die Segmentierung bzw. die Fensterung folgt nun die Merkmalsextraktion. Es sei bemerkt, dass im hier vorgestellten Fall der Merkmalsextraktion mittels Codierung durch Shift-invariantes Sparse Coding (also Sparse Coding im Codierungsmodus mit fester Basis) auch zuerst das kontinuierliche Signal codiert werden und danach die Segmentierung bzw. Fensterung des bereits codierten Signals erfolgen kann. Das kann sinnvoll sein, wenn im Merkmalsextraktionsschritt Information über weite Teile des Signals miteinander verrechnet werden soll und eine Fensterung davor Zusammenhänge zerstören würde.

Jedes Exemplar wird nun mittels Sparse Coding im Codierungsmodus, also mit fester Basis $\mathbf{W}^{(d)}$, welche zuvor beim Training des Merkmalsextraktionsschrittes gelernt wurde, in seine spärliche Repräsentation als Matrix der Aktivierungen $H^{(:)}_{:,n}$ transformiert.

Da in dieser Form der Notation der Aktivierungen nicht eindeutig ist, welcher Index über die Zeilen und welcher über die Spalten der Matrix zählt, wird für alle weiteren Betrachtungen die Matrix der Aktivierungen für das n-te Exemplar $\mathbf{A}^{(n)} \in \mathbb{R}^{M \times K}$ mit $A^{(n)}_{m,k} = H^{(m)}_{k,n}$ eingeführt, welche in der k-ten Spalte die Aktivierungen aller Verschiebungen m des k-ten Basisvektors enthält.

Zum Training des Klassifikators muss zu jedem Exemplar aus dem Trainingsdatensatz die Information über die Klassenzugehörigkeit vorliegen. Diese Klasseninformation wird im Folgenden mit $\mathbf{y} \in \mathbb{N}^N$ bezeichnet. Dabei ist y_n eine natürliche Zahl, welche die Klasse des n-ten Fensters notiert.

Für die folgenden Untersuchungen wird zunächst das einfachere Szenario der isolierten Erkennung gewählt. Später folgen dann auch Untersuchungen mit der kontinuierlichen Erkennung.

5.2.1. Klassifikation aggregierter Aktivierungen

In diesem Abschnitt wird zunächst das Konzept der Transformation eines Exemplars oder Fensters in einen Merkmalsvektor und der anschließenden Klassifikation theoretisch beschrieben. Im folgenden Abschnitt 5.2.2 wird die Anwendung des Konzeptes an einem konkreten Beispiel gezeigt.

Eine einfache Methode zur Klassifikation besteht darin, ein gegebenes Fenster in einen kompakten Merkmalsvektor zu überführen und diesen dann mit einem Standardklassifikator für Vektorräume zu klassifizieren (Referenzen dazu in Abschnitt 2.6). Dazu werden die Features über das gesamte Fenster aggregiert (siehe Abbildung 5.6). Wird z. B. die Stärke der Aktivierung jeder Primitive im gesamten Fenster gemittelt, so erhält man die durchschnittliche Aktivierung einer Primitive für ein Exemplar. Bei K Primitiven ergibt sich dadurch für jedes Fenster ein Merkmalsvektor $\mathbf{x} \in \mathbb{R}^K$ mit $x_k = \frac{1}{M} \sum_m A_{m,k}$. Neben der Aggregation durch Mittlung über den Zeitverlauf sind auch andere Methoden, wie die Bildung des Maximums über den Zeitverlauf oder eine binäre Schwellwertoperation denkbar. Darauf wird später noch eingegangen. Die Merkmalsvektoren aller Fenster können in einer Datenmatrix $X \in \mathbb{R}^{N \times K}$ zusammengefasst werden.

Wie weiter vorn schon beschrieben wurde, stehen zusätzlich die Label-Informationen zu allen Merkmalsvektoren als $\mathbf{y} \in \mathbb{N}^N$ zur Verfügung. Mit Hilfe dieser Information kann nun ein beliebiger Klassifikator für Vektorräume trainiert werden.

Aggregation der Aktivierungen Die Wahl der Aggregierungsfunktion beeinflusst die Auswahl der anwendbaren Klassifikatoren maßgeblich. Im Falle der Mittlung der Aktivierungen pro Primitive über die Zeit, wie sie bereits oben vorgestellt wurde, beschreibt x_k die durchschnittliche Aktivierung der k-ten Primitive im Exemplar. Da es sich um einen reellwertigen Merkmalsvektor handelt, ist die Anwendung beliebter Klassifikatoren, wie der Support-Vektor-Maschine [Burges, 1998], des Decision-Trees [Quinlan, 1986] oder des K-Nearest-Neighbor-Klassifikators [Cover et al., 1967] denkbar.

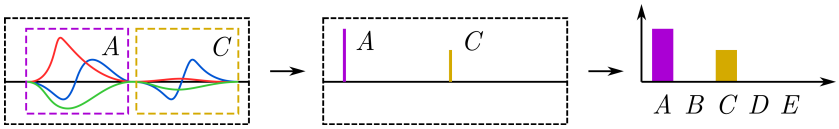


Abb. 5.6.: *Schema der Überführung eines Exemplars (links) in seine Aktivierungsfolge (Mitte) und Aggregation zu einem Featurevektor (rechts). Die Buchstaben A, B, C, etc. bezeichnen die Basisvektoren. Das Exemplar (links) besteht aus einer Folge der beiden Basisvektoren A und C. Die Aktivierungsfolge (Mitte) zeigt die durch Sparse Coding ermittelten Aktivierungen. Durch Aggregation über die Zeit ergibt sich ein Featurevektor, der nur noch die Stärke der Aktivierung aller möglichen Basisvektoren im Fenster enthält, hier durch ein Balkendiagramm dargestellt.*

Weiterhin ist eine Maximumsbildung der Aktivierungen pro Primitive über die Zeit denkbar, womit x_k die maximale Aktivierung einer Primitive im Exemplar beschreibt. Dies ist insbesondere dann nützlich, wenn das Sparse Coding so parametrisiert wird, dass Primitive in den meisten Fällen höchstens einmal in einem Exemplar vorkommen und fast alle Aktivierungen null sind. Die Maximumsbildung beschreibt dann, ob und wie stark die jeweilige Primitive im Exemplar vorkommt. Da es sich auch hier um einen reellwertigen Merkmalsvektor handelt, eignen sich hier die gleichen Klassifikatoren wie bei der Mittlung.

Auf Basis der Maximumsbildung kann man noch eine weitere Reduktion vornehmen, indem man eine binäre Schwellwertoperation der Art

$$x_k = \Theta_\epsilon \left(\max_m A_{m,k} \right) = \begin{cases} 1 & , \max_m A_{m,k} \geq \epsilon \\ 0 & , sonst \end{cases}$$

anwendet, womit $x_k \in \{0, 1\}$ beschreibt, ob die k-te Primitive im Fenster vorkommt oder nicht. Das Verfahren kann damit als eine Art Merkmals-Detektor interpretiert werden und der Feature-Vektor im Sinne einer Bag-of-Words-Beschreibung (siehe dazu [Csurka et al., 2004]).

Naive-Bayes-Ansätze Im Rahmen der in dieser Arbeit durchgeführten Experimente haben sich insbesondere die Modelle nach dem Naive-Bayes-Ansatz als erfolgreich erwiesen. Dabei handelt es sich um statistische Modelle, die Aufgrund der Annahme, dass die beteiligten Variablen unabhängig sind, sehr einfach formuliert werden können und dadurch wenige Parameter haben. Bei diesen Modellen wird die Klassenzugehörigkeit c^* eines Merkmalsvektors $\mathbf{x}$ mittels Maximierung der Likelihood über alle Klassen c bestimmt:

$$c^* = \max_c P(\mathbf{x}|c) \ .$$

Dabei beschreibt $P(\mathbf{x}|c)$ die klassenspezifische Verteilung.

Im Fall der Aggregation durch Mittlung der Aktivierungen und des dadurch entstehenden reellen Merkmalsvektors, eignet sich das Gaussian-Naive-Bayes-Modell, bei dem die klassenspezifische Verteilung mittels einer multivariaten Gaussverteilung modelliert wird:

$$P(\mathbf{x}|c) = \mathcal{N}(\mathbf{x}; \mu_c, \mathbf{\Sigma}_c) \ ,$$

wobei der klassenspezifische Mittelwert $\mu_c \in \mathbb{R}^K$ und die klassenspezifische Kovarianzmatrix $\mathbf{\Sigma}_c \in \mathbb{R}^{K \times K}$ sehr einfach aus den Trainingsdaten bestimmt werden können.

Im Fall der Aggregation durch eine binäre Schwellwertoperation und dem dadurch entstehenden binären Merkmalsvektor $\mathbf{x} \in \{0,1\}^K$, eignet sich das Bernoulli-Naive-Bayes-Modell, welches die Klassen mittels multivariaten Bernoulli-Verteilungen modelliert:

$$P(\mathbf{x}|c) = \prod_k p_{c,k}^{x_k}(1 - p_{c,k}^{1-x_k}) \ ,$$

wobei $p_{c,k} \in [0,1]$ die Wahrscheinlichkeit für das Vorkommen von Primitive k in Klasse c ist. Dabei sei bemerkt, dass das vollständige multivariate Bernoulli-Modell auch noch Faktoren für die Abhängigkeiten zwischen den Komponenten hat, die aber aufgrund der Unabhängigkeitsannahme hier weg fallen. Die Parameter $p_{c,k}$ können einfach durch Bestimmen der relativen Häufigkeit der Primitive pro Klasse ermittelt werden.

5.2.2. Isolierte Gesten-Klassifikation

Im Folgenden soll gezeigt werden, dass eine Klassifikation auf Basis der sehr stark reduzierten Repräsentation durch die Merkmalsvektoren mit den oben vorgestellten Modellen möglich ist. Als Datensatz wird dafür der im Rahmen dieser Arbeit selbst erstellte Datensatz verwendet (siehe Abschnitt 3.3.1). Das Rahmenszenario bleibt, wie oben beschrieben, die isolierte Klassifikation. Die Daten bestehen damit aus isolierten Exemplaren und der zugehörigen Klasseninformation.

Um eine Referenz für die Güte der Klassifikation zu erhalten, wird der Datensatz zuerst mit einem Standardverfahren, dem Hidden-Markov-Modell (HMM) [Rabiner, 1989; Rabiner und Levinson, 1981], klassifiziert. Danach wird der gleiche Datensatz unter Verwendung des Sparse Coding und Berechnung eines Merkmalsvektors mittels Aggregation klassifiziert und dem Referenzergebnis mit dem Standardverfahren gegenübergestellt.

Das Problem wird zunächst auf die Klassifikation der Buchstaben „a“, „b“ und „c“ beschränkt. Zur Ermittlung des Referenzergebnisses wurde der Datensatz bestehend aus diesen drei Buchstaben mittels HMM klassifiziert. Dafür wurden HMMs mit Gaussschen Emissionsverteilungen (siehe [Rabiner, 1989]) gewählt. Die Anzahl der Zustände der HMM wurde mittels einer systematischen Suche optimiert. Die Experimente wurden mit zehnfacher Kreuzvalidierung durchgeführt. Mit diesem Ansatz wird eine durch-

Aggregation	Klassifikator	# Primitive	Accuracy
-	Gaussian HMM	-	98,0%
Mittelwert	Gaussian-Naive-Bayes	3	92,2%
Mittelwert	Gaussian-Naive-Bayes	9	97,6%
Mittelwert	Gaussian-Naive-Bayes	21	**98**,**0**%
Mittelwert	Gaussian-Naive-Bayes	30	98,0%
Binärer Schwellwert	Bernoulli-Naive-Bayes	3	63,3%
Binärer Schwellwert	Bernoulli-Naive-Bayes	9	96,8%
Binärer Schwellwert	Bernoulli-Naive-Bayes	21	97,6%
Binärer Schwellwert	Bernoulli-Naive-Bayes	30	**98**,**0**%

Tab. 5.1.: *Ergebnisse der Klassifikation mittels Sparse Coding und Aggregation im Vergleich.*

schnittliche Accuracy [3] von 98,0% erreicht.

Klassifikation mittels Naive-Bayes Diesem Ergebnis soll nun eine Klassifikation nach Transformation mittels Sparse Coding, anschließender Aggregierung und Klassifikation mittels eines merkmalsvektorbasierten Klassifikators gegenübergestellt werden. Es werden im Folgenden zwei Kombinationen aus Aggregierungen und Klassfikatoren verglichen. Dies sind zum einen die Aggregierung mittels Mittelwertbildung und die Klassifikation mittels Gaussian-Naive-Bayes (Mean-GNB) und zum anderen die Aggregierung mittels binärer Schwellwertoperation und die Klassifikation mittels Bernoulli-Naive-Bayes (Binary-BNB). In beiden Fällen wird zunächst ein Sparse Coding mit globaler und lokaler Spärlichkeit (siehe Abschnitt 4.2.5) durchgeführt.

Tabelle 5.1 zeigt die Ergebnisse in Abhängigkeit von der Anzahl der verwendeten Basisvektoren nach Optimierung der restlichen Parameter mittels systematischer Suche. Mit beiden Modellen konnte eine Accuracy von ebenfalls 98,0% erreicht werden. Das Mean-GNB-Modell erreicht dieses Ergebnis mit einer optimalen Anzahl von 21 Primitiven, während beim Binary-BNB-Modell 30 Primtive notwendig sind. Dies liegt unter anderem darin begründet, dass beim Mean-GNB-Modell auch die Stärke der Aktivierungen Informationen für die Diskriminierung der Klassen enthält, was beim Binary-BNB-Modell, bei dem die Aktivierungen binarisiert werden, durch eine größere Anzahl an Basisvektoren kompensiert wird.

Tabellen 5.2 und 5.3 zeigen die Konfusionsmatrizen für die beiden Modelle mit ihren jeweiligen als optimal ermittelten Parametern (21 Primitive bei Mean-GNB und 30 Primitive beim Binary-BNB). Die Klassifikatoren verwechseln einzelne Exemplare der Buchstaben „a“ und „c“, da diese Klassen sehr ähnliche Features haben.

Die Ergebnisse zeigen, dass es möglich ist, mit dem hier beschriebenen Ansatz eine

[3] In einem Mehrklassenproblem, wie dem hier gezeigten, ist die Accuracy (häufig auch Accuracy-Score) der Anteil der korrekt klassifizierten Beispiele.

		Prädiziert		
		a	b	c
Real	a	0,981	0	0,019
	b	0	1	0
	c	0,037	0	0,963

Tab. 5.2.: *Konfusionsmatrix für Mean-GNB mit 21 Primitiven.*

		Prädiziert		
		a	b	c
Real	a	0,975	0	0,025
	b	0	1	0
	c	0,029	0	0,971

Tab. 5.3.: *Konfusionsmatrix für Binary-BNB mit 30 Primitiven.*

vergleichbar gute Klassifikationsrate (Accuracy) wie bei der Klassifikation mittels HMM zu erreichen. Sowohl die Klassifikation der gemittelten Aktivierungen mittels Gaussian-Naive-Bayes, als auch die Klassifikation der binären Features mittels Bernoulli-Naive-Bayes liefern gute Ergebnisse. Es ist zu beobachten, dass bei Mean-GNB bereits relativ wenige (z. B. 3) Primitive ausreichen, um eine akzeptable Klassifikationsrate zu erzielen, wogegen bei Binary-BNB erst ab 9 Primitive akzeptable Klassifikationsraten erreicht werden. Dies liegt daran, dass bei Mean-GNB auch im konkreten Wert der durchschnittlichen Aktivierung für die Klassifikation nützliche Information codiert ist und somit auch mit weniger Features (Primitiven) eine Unterscheidung möglich ist.

Insgesamt interessant an diesen Ergebnissen ist, dass bereits die sehr reduzierte Form der binären Darstellung (also, ob ein bestimmtes gelerntes Feature vorhanden ist oder nicht) ausreicht, um die Klassen erfolgreich trennen zu können.

Klassifikation mittels K-Nearest-Neighbor Sowohl das HMM, als auch Mean-GNB und Binary-BNB sind generative Ansätze in parametrischer Form, bei dem eine Klasse durch ein generatives Modell (eine parametrische Wahrscheinlichkeitsverteilung) modelliert wird. Die einfache Form der Repräsentation durch Merkmalsvektoren macht allerdings auch die Verwendung einiger weiterer Modelle möglich. Sehr beliebt ist das K-Nearest-Neighbor-Modell (KNN) [Cover et al., 1967], weil dort keine Annahmen über die den Klassen zugrundeliegenden Verteilungen notwendig sind. Tabelle 5.4 zeigt Ergebnisse der Anwendung eines K-Nearest-Neighbor-Klassifikators auf die gemittelten Aktivierungen (Tabelle 5.5 zeigt die Konfusionsmatrix für 21 Primitive).

Mit diesem Modell kann eine Accuracy von 99,6% erreicht werden, was eine signifikante Verbesserung zu den bisherigen Ergebnissen darstellt. Allerdings ist das Modell auch sehr stark von der Nachbarschaftsstruktur der Daten abhängig, und so muss im konkreten Einzelfall untersucht werden, ob dieses Modell einsetzbar ist.

Die bisherigen Experimente wurden nur mit Daten aus drei Klassen ausgeführt. Abbildung 5.7 zeigt einen Vergleich der Anwendung von HMM und K-Nearest-Neighbors mit Sparse-Coding-Features mit steigender Anzahl von Klassen, bis hin zum vollständigen Datensatz mit 26 Klassen. Abbildung 5.8 zeigt die Konfusionsmatrizen für 10 bzw. 26 Klassen.

Aggregation	Klassifikator	# Primitive	Accuracy
Mittelwert	KNN	3	95,6%
Mittelwert	KNN	9	98,8%
Mittelwert	KNN	21	**99,6**%
Mittelwert	KNN	30	99,2%

Tab. 5.4.: *Ergebnisse der Klassifikation mittels Sparse Coding, Aggregation und Klassifikation mittels K-Nearest-Neighbor.*

		Prädiziert		
		a	b	c
Real	a	1	0	0
	b	0	1	0
	c	0,012	0	0,988

Tab. 5.5.: *Konfusionsmatrix für K-Nearest-Neighbors mit 21 Primitiven.*

Das KNN mit SC-Features zeigt in allen Fällen eine signifikant höhere Accuracy als das HMM. Die durch Sparse Coding gelernten Features sind also allgemein für eine Klassifikation geeignet. Insgesamt ist aber bei beiden Verfahren bei hohen Anzahlen von Klassen die Klassifikationsrate eher mäßig, was auf einen eher schwer zu klassifizierenden Datensatz hindeutet, für den komplexere Verfahren notwendig sind. Es sei nochmals erwähnt, dass das Ziel dieser Arbeit nicht vorrangig die Klassifikation dieses Datensatzes ist, sondern die Untersuchung der Eignung von Sparse Coding als Merkmalsextraktionsverfahren.

Verbesserungen durch lokale Spärlichkeit Anhand des Mean-GNB-Modells soll nun noch gezeigt werden, welchen Einfluss die lokale Spärlichkeit auf die Leistung des Klassifikators hat. Abbildung 5.9 zeigt die Accuracy des Modells Mean-GNB in Abhängigkeit vom Parameter λ_2, der die lokale Spärlichkeit bestimmt.

Die Ergebnisse zeigen, dass die lokale Spärlichkeit einen Einfluss auf die Leistung des Klassifikators hat. Die gelernten Features sind bei erhöhter lokaler Spärlichkeit nützlicher für die Klassifikation. Wird λ_2 zu hoch gewählt, nimmt die Accuracy allerdings wieder ab, da hohe Spärlichkeitsparameter generell die Rekonstruktionsgüte und damit den Informationsgehalt der spärlichen Codierung beeinträchtigen. Hier ist also in der konkreten Anwendung immer eine Parametersuche nach einem guten Wert notwendig.

Self-taught Learning

In diesem Abschnitt sollen einige Betrachtungen zum Thema Self-taught Learning (siehe z. B. [Grosse et al., 2007]) durchgeführt werden. Das Konzept behandelt ein prakti-

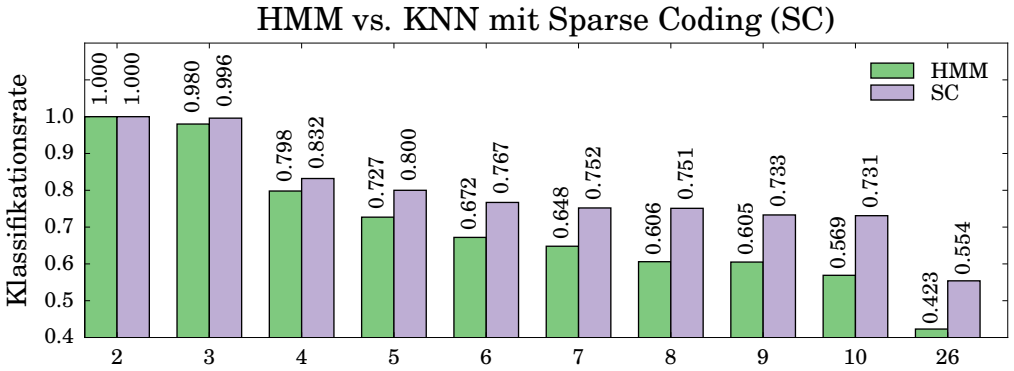


Abb. 5.7.: *Vergleich der Klassifikationsrate (Accuracy) von HMM und K-Nearest-Neighbors auf durch Sparse Coding gelernten Features für eine unterschiedliche Anzahl von Klassen. Ganz links wurden die Klassen „a“ und „b“ verwendet, rechts daneben „a“, „b“ und „c“, usw. Die Ergebnisse für elf bis 25 Klassen wurden aus Platzgründen ausgelassen.*

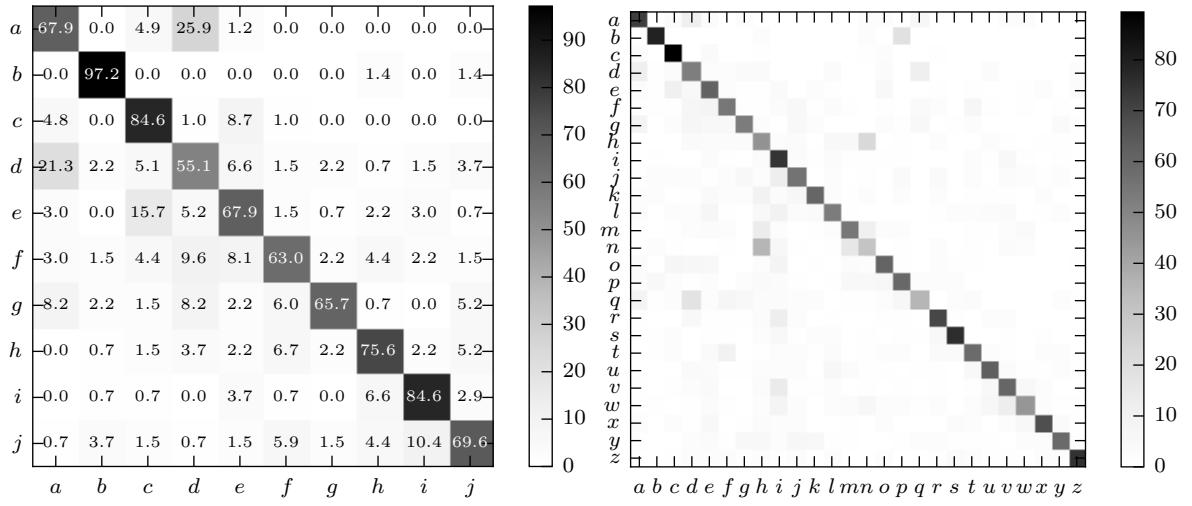
sches Problem bei Machine-Learning-Verfahren, welches darin besteht, dass häufig nur eine sehr begrenzte Menge von Trainingsdaten zur Verfügung steht. In [Grosse et al., 2007] wird z.B. der Fall beschrieben, dass eine audiobasierte Identifikation zweier Sprecher durchgeführt werden soll, aber nur wenige gelabelte Samples dieser Sprecher zum Training des Klassifikators zur Verfügung stehen. Die Autoren führen aus, dass es bei Verwendung von Feature-Learning-Verfahren, wie dem Sparse Coding möglich ist, die Leistung der Klassifikatoren zu erhöhen, wenn man weitere ungelabelte Samples zur Verfügung hat, die aus einer ähnlichen Verteilung wie die der gelabelten Samples entstammen, also z.B. beliebige Sprach-Samples von anderen Sprechern. Diese zusätzlichen Samples können beim Training der Merkmalsextraktion mit verwendet werden, da diese ja unüberwacht erfolgt. Die größere Datenmenge kann dazu führen, dass sich bessere Merkmale herausbilden und somit auch die Codierung besser wird.

Auf den Kontext dieser Arbeit angewendet, bedeutet das, dass es möglich wäre, die Leistung des Klassifikators zu verbessern, wenn zum Training weitere Exemplare ohne oder mit anderen Labels zur Verfügung stünden. Dieser Fall wird im Folgenden simuliert, indem, wie oben die Klassen „a“, „b“ und „c“ bestimmt werden sollen, aber für das Lernen der Primitive auch die Exemplare anderer Klassen zur Verfügung stehen.

Tabelle 5.6 zeigt die Ergebnisse unter Hinzunahme von Exemplaren anderer Klassen zum Training der Merkmalsextraktion. Es ist erkennbar, dass die Hinzunahme der Klasse „d“ und „e“ eine signifikante Reduktion der Fehlerrate [4] mit sich bringt.

Das Konzept des Self-taught-learning ist also auch auf das in dieser Arbeit vorgestellte Verfahren anwendbar. Die Wirksamkeit dieses Konzeptes hängt natürlich stark von der Struktur der Daten ab. Da hier nur wenige Klassen verwendet worden sind und damit das

[4]Die Fehlerrate (engl. Error Rate) ist der Anteil der falsch klassifizierten Beispiele (Fehlerrate = 1-Accuracy).



(a) (b)

Abb. 5.8.: *Konfusionsmatrizen für die Klassifikation der ersten 10 Klassen (a) bzw. aller 26 Klassen (b).*

Problem vereinfacht wurde, bedarf es weiterer Untersuchungen, inwiefern das Konzept auf andere Datensätze anwendbar ist.

5.2.3. Kontinuierliche Klassifikation für die Activity-Recognition

Im Folgenden soll die Anwendung des Verfahrens auf den Fall der kontinuierlichen Erkennung angewendet werden. Das in Abschnitt weiter vorn vorgestellte Szenario der kontinuierlichen Erkennung mit der Fensterung eines kontinuierlichen Signals, der Überführung jedes Fensters in einen Merkmalsvektor und der anschließenden auf Merkmalsvektoren basierenden Klassifikation ist ein beliebtes Verfahren, insbesondere im Bereich

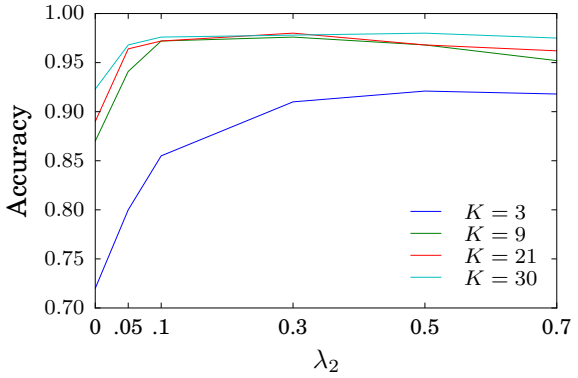


Abb. 5.9.: *Accuracy des Mean-GNB-Modells für verschiedene Anzahlen von Primitiven K, abhängig vom Parameter für die lokale Spärlichkeit λ_2.*

# Primitive	Klassen zum Lernen	Accuracy
30	a,b,c	98,0%
30	a,b,c,d	98,5%
30	a,b,c,d,e	98,8%
30	a,b,c,d,e,f	98,8%

Tab. 5.6.: *Accuracy beim Lernen der Merkmalsextraktion mit zusätzlichen, ungelabelten Daten.*

der Activity-Recognition. Um die generelle Verwendbarkeit des vorgestellten Verfahrens zu belegen, soll es nun auch auf Datensätze aus diesem Bereich angewendet werden. Die Untersuchungen beziehen sich dabei auf die Arbeit von Plötz et al. [Plötz et al., 2011], in der vier konventionelle Verfahren zur Merkmalsextraktion auf vier frei verfügbare Datensätze aus dem Bereich der Aktivitätserkennung angewendet und evaluiert wurden. Die Datensätze wurden in Abschnitt 3.3.3 beschrieben. Diese Ergebnisse werden um die Anwendung des in dieser Arbeit entwickelten Verfahrens auf diese vier Datensätze erweitert.

Weiterhin werden die auf den in dieser Arbeit erstellten Datensatz (siehe Abschnitt 3.3.1) angewendet, welcher im Folgenden mit GT (für „Gesture Trajectories“) bezeichnet wird. Im Unterschied zu den anderen Datensätzen wurden die Daten hier allerdings nicht mit Hilfe von Beschleunigungssensoren gewonnen, sondern durch Aufnahme der Position von Gelenken mit Hilfe von Tiefenkamera und Tracking (siehe Abschnitt 3.2.1). Um eine Vergleichbarkeit herzustellen, wird hier zunächst aus dem Verlauf der Position über die Zeit der Verlauf der Beschleunigung berechnet [5]. Da Plötz et al. in jedem Datensatz nur jeweils einen Sensor betrachten, wird auch hier die Trajektorie nur eines Gelenks ausgewählt. Es wird die rechte Hand gewählt, da die Bewegungen in dem Datensatz hauptsächlich (siehe Abschnitt 3.3.1) mit der rechten Hand ausgeführt wurden. Die Daten werden mit einer Sampling-Frequenz von 30Hz abgetastet.

Referenzverfahren Plötz et al. [Plötz et al., 2011] verwenden die Datensätze zur Evaluierung einer Reihe an Merkmalsextraktionsverfahren. Sie wenden dazu das jeweilige Merkmalsextraktionsverfahren auf den jeweiligen Datensatz an und klassifizieren die erzeugten Merkmale mittels eines Nearest-Neighbor-Klassifikators. Bei den verglichenen Verfahren handelt es sich zum Teil um klassische Verfahren zur Merkmalsextraktion als auch um neuere Verfahren welche Feature-Learning implementieren. Die Verfahren sollen im Folgenden kurz vorgestellt werden. Für weitere Details sei auf die Publikation [Plötz et al., 2011] verwiesen.

Eines der einfachsten Verfahren zur Extraktion von Merkmalen ist die Berechnung statistischer Metriken über einem Fenster $\mathbf{A}^{(n)} \in \mathbb{R}^{M \times K}$ der Länge M mit K Kanälen.

[5] Zuerst wird die Geschwindigkeit mittels Kalman-Filter geschätzt (siehe dazu auch Abschnitt 3.2.2) und dann durch Differenzbildung die Beschleunigung berechnet.

Dazu wird über jedem der Kanäle eine statistische Maßzahl bestimmt. Die Maße aller Kanäle werden dann zum Merkmalsvektor konkateniert. Bei Plötz et al. sind die Kanäle eines Frames die x-, y- und z-Koordinaten eines ausgewählten Beschleunigungssensors. Zur Berechnung der statistischen Metriken berechnen sie zusätzlich noch den Nick- und Gier-Winkel des Sensors. Über diesen fünf Kanälen berechnen sie nun jeweils den Mittelwert, die Standardabweichung, die Energie, die Entropie. Zusätzlich berechnen sie die paarweise Korrelation zwischen den Kanälen x, y, und z. Daraus resultiert ein 23-dimensionaler Merkmalsvektor für jedes Fenster. Das Verfahren wird im Folgenden mit STAT bezeichnet.

Als weiteres Verfahren verwenden Plötz et al. die diskrete Fouriertransformation (DFT) [Cooley et al., 1969]. Dabei werden über allen Kanälen separat die Beträge einer festen Anzahl komplexer Fourierkoeffizienten ermittelt. Diese werden dann zum Merkmalsvektor konkateniert. Plötz et al. verwenden die ersten zehn Fourierkoeffizienten. Das Verfahren wird im Folgenden mit FFT (für Fast Fourier Transform, ein Algorithmus, welcher die DFT implementiert) bezeichnet.

Die beiden bisher angesprochenen Verfahren sind konventionelle Merkmalsextraktionsverfahren insofern, dass die Gestalt der Merkmale von Designer des Systems fest gewählt ist. Im Unterschied dazu sind die folgenden Verfahren „echte“ Feature-Learning-Verfahren (wie auch das in dieser Arbeit entwickelte Sparse-Coding-Verfahren), welche die Gestalt der Merkmale aus den Daten lernen und damit die Abhängigkeit vom Designerwissen stark reduzieren.

Als erstes Feature-Learning-Verfahren verwenden Plötz et al. die Principal Component Analysis (PCA) [Jolliffe, 2002]. Da die PCA nur in einem Vektorraum angewendet werden kann, muss zuerst das Fenster in einen solchen überführt werden. Dazu werden die einzelnen Kanäle des Fensters zu einem Vektor konkateniert. Bei K Kanälen und einer Fenster-Länge von M, ergibt sich so ein Vektorraum der Dimension $K \cdot M$ auf dem die PCA angewendet wird. Die berechneten Eigenvektoren können als an die Daten adaptierte Merkmale interpretiert werden. Die Koeffizienten zur Repräsentation der Eingangsdaten im PCA-Raum stellen die Merkmalsausprägungen dar und bilden den Merkmalsvektor. Da der PCA-Raum sehr hochdimensional ist und somit auch die Anzahl der Eigenvektoren sehr hoch ist und zugleich der Relevanz eines Eigenvektors zur Repräsentation der Daten proportional zu seinem Eigenwert ist, ist es sinnvoll, nur die Teilmenge der Eigenvektoren mit relativ hohem Eigenwert zu verwenden. Damit sind die wesentlichen Merkmale gut repräsentiert und die Dimension des Merkmalsraumes ist nicht zu hoch, um für nachgelagerte Verarbeitungsschritte problematisch zu werden. Plötz et al. verwenden die 30 relevantesten Eigenvektoren.

Als zweites Feature-Learning-Verfahren verwenden Plötz et al. ein Deep Belief Network (DBN) [Bengio, 2009]. Dabei handelt es sich um ein neuronales Verfahren, welches ein mehrschichtiges neuronales Netz schichtweise lernt, indem es zwei verbundene Schichten jeweils als Restricted Boltzmann Machine (RBM) [Hinton et al., 2006] behandelt. Eine RBM weist im Ansatz sehr starke Ähnlichkeiten zum in dieser Arbeit vorgestellten Verfahren der NMF auf. Es wird ebenfalls eine Energiefunktion minimiert, welche die

Rekonstruktionsgüte der Dekomposition der Daten in Merkmale und ihre Ausprägungen misst. Es bestehen leichte Unterschiede in der Formulierung der Energiefunktion und in den Nebenbedingungen. Jede Schicht des DBN führt eine Merkmalstransformation durch. Somit kann das DBN als konzeptionell mächtiger im Vergleich zur NMF angesehen werden. Es hängt allerdings stark von den Daten ab, ob sich diese Mächtigkeit auch in einer besseren Leistung manifestiert. In der Literatur wurde mehrfach gezeigt, dass mit Hilfe von DBN nützliche Merkmale aus Daten gelernt und für die Klassifikation verwendet werden können. Plötz et al. verwenden ein Netz mit vier Schichten. In jeder Hidden-Schicht befinden sich 1024 Neuronen, in der Ausgabeschicht befinden sich 30 Neuronen. Damit ist der resultierende Merkmalsvektor 30-dimensional.

Die Untersuchungen von Plötz et al. werden um das in dieser Arbeit vorgestellte Verfahren erweitert, welches im Folgenden kurz mit SISC (für Shift-invariant Sparse Coding) bezeichnet wird. Die Parameter wurden mittels systematischer Suche (Grid-Search) optimiert [6].

Evaluierung Um die Vergleichbarkeit mit den in [Plötz et al., 2011] vorgestellten Ergebnissen zu sichern, wird die dort vorgestellte Evaluierungsmethode verwendet. Diese soll im Folgenden kurz beschrieben werden.

Die Daten liegen in Form eines diskreten Signals mit drei Kanälen vor (Plötz et al. betrachten nur einen Sensor). Dieses Signal wird in überlappende Fenster gleicher Größe zerteilt. Die Fenster haben dabei eine Länge von 64 Samples. Durch die unterschiedlichen Sampling-Raten der Datensätze ergeben sich somit Fenster unterschiedlicher zeitlicher Länge. Obwohl fragwürdig ist, ob die Wahl einer festen Anzahl von Samples pro Fenster anstatt einer festen zeitlichen Länge des Fensters sinnvoll ist, wird diese Vorgabe im Folgenden eingehalten, um die Vergleichbarkeit der Auswertungen sicherzustellen. Die Fenster haben einen Überlapp von 50%. Die Evaluierung wird mit 10-facher Kreuzvalidierung ausgeführt.

Abbildung 5.10 zeigt einen Vergleich der Accuracy für die verschiedenen Datensätze und unter Verwendung der verschiedenen Merkmalsextraktionsverfahren. Insgesamt ist erkennbar, dass die in dieser Arbeit vorgestellte Methode die Datensätze sehr erfolgreich klassifizieren kann. Bei den Datensätzen AK, SK und OC ist sie den anderen Methoden signifikant überlegen. Beim Datensatz DA liegt sie knapp hinter der FFT und nimmt damit den zweiten Platz ein. Beim Datensatz SK liegt sie nur insignifikant vor der FFT. Interessant ist, dass die FFT für drei der fünf Datensätze den zweiten oder ersten Platz einnimmt. Der Grund dafür liegt wahrscheinlich in der Tatsache, dass der FFT, vergleichbar zum SISC, bereits eine Verschiebungsinvarianz inhärent ist, denn es werden lediglich die Beträge der komplexen Fourier-Koeffizienten verwendet. Die Phasenverschiebungen, welche die Lokalisierung eines Merkmals innerhalb eines Fensters codieren, werden ignoriert. Dies ist ähnlich zum Effekt der Aggregation der Merkmals-

[6] Die optimierten Parameter lauten wie folgt: Länge der Basisvektoren $L = 0{,}5s$, Anzahl der Basisvektoren $K = 20$, globaler Spärlichkeitsparameter $\lambda_1 = 0{,}1$, lokale Spärlichkeit $\lambda_2 = 0{,}1$.

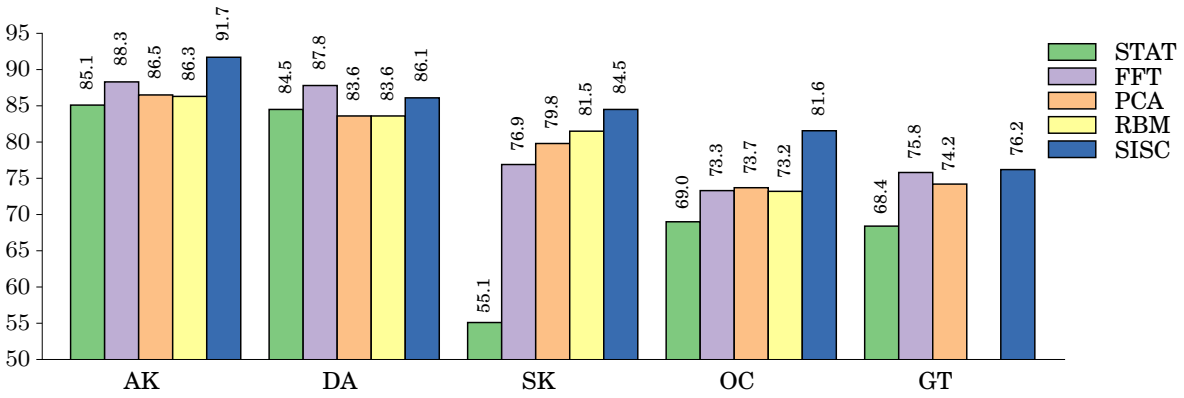


Abb. 5.10.: *Accuracy des KNN-Klassifikators unter Einsatz verschiedener Verfahren zur Merkmalsextraktion. Die Verfahren STAT (Extraktion einfacher statistischer Kenngrößen), FFT (Diskrete Fourieranalyse), PCA (Principal Component Analysis), RBM (Restricted Boltzman Machines oder Deep Belief Networks) und SISC (das in dieser Arbeit entwickelte Verfahren Shift-invariant Sparse Coding) wurden auf die Datensätze AK (Ambient Kitchen 1.0, Zubereitung von Mahlzeiten), DA (Darmstadt Daily Routines, Haushaltstätigkeiten), SK (Skoda Mini Checkpoint, Wartungstätigkeiten an Fahrzeugen), GT (der in dieser Arbeit erstellte Datensatz „Gesture Trajectories“, bestehend aus einfachen Gesten) angewendet (Für Details zu den Datensätzen, siehe Kapitel 3.3) . Für den Datensatz GT wurde keine Evaluierung mittels der Methode RBM durchgeführt.*

ausprägungen über ein Fenster beim SISC. Beide Verfahren haben also den Vorteil der Verschiebungsinvarianz. Auch das Verfahren RBM könnte theoretisch eine Verschiebungsinvarianz implementieren. Im Gegensatz zu SISC und FFT muss diese allerdings aus den Daten gelernt werden und wird damit wahrscheinlich weniger gut umgesetzt als die explizite Formulierung in der FFT und im SISC.

Ein wesentlicher Nachteil des SISC gegenüber der FFT ist allerdings der immens höhere Rechenaufwand. Während für die FFT sehr effiziente, geschlossene Lösungsverfahren existieren, ist das SISC ein inkrementelles Verfahren. Es hängt also von den technischen Rahmenbedingungen und den Daten ab, ob sich der Aufwand für die Verwendung von SISC lohnt.

5.2.4. Untersuchungen zu dynamischen Modellen

Die bisher entwickelten Modelle transformieren ein gegebenes Exemplar in einen Merkmalsvektor und ignorieren damit die Tatsache, dass die Daten eigentlich eine zeitliche Sequenz darstellen und damit auch Beziehungen wie Reihenfolge und zeitliche Abfolge der Aktivierungen innerhalb eines Exemplars.

Im Rahmen dieser Arbeit wurden auch Modelle untersucht, welche die Charakteristik der Daten als Zeitreihe explizit behandeln und die zeitlichen Abhängigkeiten in den

Daten modellieren. Zur Unterscheidung dieser Modellklasse, werden diese Modelle als sequenzbasierte Modelle bezeichnet, da sie Sequenzen von Merkmalen modellieren. Die Motivation zur Untersuchung dieser Verfahren war, dass auch in der zeitlichen Abfolge oder auch nur in der Reihenfolge nützliche Information für die Klassifikation enthalten sein kann.

Es wurde ein dynamisches Modell, das Activity String Matching (siehe Anhang D.2), entwickelt, welches auf der Eigenschaft des in dieser Arbeit entwickelten Codierungsverfahrens basiert, eine Zeitreihe durch diskrete Primitive darstellen zu können. Mit diesem Modell konnte allerdings keine signifikante Verbesserung der Klassifikation nachgewiesen werden. Die Ursache dafür liegt wahrscheinlich darin begründet, dass die „zeitlose“ Repräsentation bereits ausreicht, um die Daten bestmöglich klassifizieren zu können. Die zeitlichen Beziehungen zwischen den Aktivierungen enthalten also für die untersuchten Datensätze keine weitere nützliche Information. Wohlgemerkt, bezieht sich das allerdings nur auf die untersuchten Datensätze. Würde es sich z. B. um einen Datensatz handeln, bei dem sich die Exemplare unterschiedlicher Klassen aus denselben wenigen Primitiven zusammensetzen würden und eine Unterscheidung nur über die Reihenfolge der Primitive möglich wäre, so würden komplexere Modelle eventuell eine bessere Leistung zeigen.

5.3. Fazit

In diesem Kapitel wurde die Generierung und Klassifikation von Bewegungstrajektorien auf Basis einer Vorverarbeitung mittels Sparse Coding vorgestellt. Es wurde gezeigt, dass aufbauend auf dem Konzept der Zerlegung und Repräsentation eines Datensatzes aus Bewegungstrajektorien mittels Bewegungsprimitiven und Modellierung des zeitlichen Regimes der typischen Abfolge von Primitiven für Klassen von Trajektorien neue Trajektorien dieser Klassen erzeugt werden können.

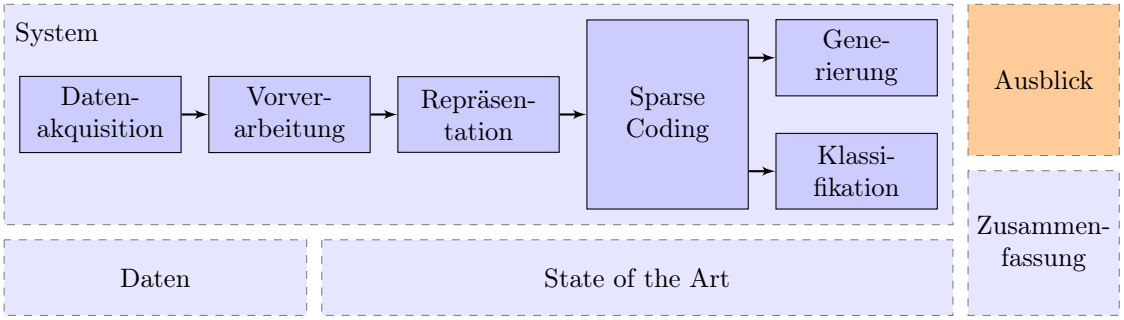
Weiter konnte gezeigt werden, dass es mit der durch das Sparse Coding stark reduzierten Repräsentation der Daten dennoch möglich ist, die Daten zu klassifizieren. Die Repräsentation macht die Verwendung sehr einfacher Standardansätze für die Klassifikation möglich, da durch das Sparse Coding bereits ein Großteil der Komplexität des Problems in der Merkmalsextraktion abgefangen wird. Es wurde an einem einfachen Datensatz gezeigt, dass im Szenario der isolierten Klassifikation Ergebnisse erzielt werden können, die mit denen von etablierten Standardverfahren, wie dem HMM, vergleichbar sind. Weiterhin wurde gezeigt, dass in einem realistischen Szenario der kontinuierlichen Klassifikation für die Activity-Recognition teilweise bessere Ergebnisse erzielt werden konnten, als das mit Standardverfahren für die Merkmalsextraktion der Fall ist.

Insbesondere interessant ist, dass das Konzept des Self-taught-Learning auf das vorgestellte Verfahren anwendbar ist, womit es prinzipiell möglich ist, die Klassifikationsleistung durch Hinzunahme ungelabelter Daten aus der gleichen Anwendungsdomäne zu erhöhen.

Ein Problem des Ansatzes ist dennoch die starke Abhängigkeit von einer Reihe an Parametern, wie der Anzahl der Primitive und der Spärlichkeitsparameter. Außerdem ist das Verfahren vor allem in der Lernphase sehr rechenaufwändig. Hier muss im Einzelfall abgewogen werden, ob der Aufwand durch die Tatsache, dass einfache Standardklassifikatoren verwendet werden können, aufgewogen werden kann.

Kapitel 6

Diskussion des Verfahrens und mögliche Erweiterungen



In diesem Kapitel folgt eine kritische Betrachtung der in der vorliegenden Arbeit dargestellten Ergebnisse. Zunächst sollen dabei die Probleme der Anwendung des Verfahrens für die Modellierung von Gesten, welche sich aus den praktischen Erfahrungen ergeben haben, betrachtet werden. Danach wird darauf eingegangen, inwiefern die am Anfang dieser Arbeit definierten Zielstellungen erreicht werden konnten.

In den darauf folgenden Abschnitten werden mögliche Erweiterungen des Verfahrens und Themenstellungen für aufbauende Arbeiten diskutiert, welche die zuvor beschriebenen Probleme teilweise beheben können. Dabei wird sowohl auf Erweiterungen des Sparse Coding in der Merkmalsextraktion, als auch auf Erweiterungen der Klassifikation eingegangen.

6.1. Kritische Betrachtung des Verfahrens

Ein schon in der Grundkonzeption des Verfahrens begründetes Problem ist die statische Natur der durch das in dieser Arbeit implementierte Sparse Coding gelernten Bewegungsprimitive. Das Sparse Coding wurde ursprünglich zur Codierung von Bildern und zum Lernen von Bild-Features verwendet [Olshausen und D. J. Field, 1996]. Merkmale in Bildern weisen typischerweise nur Skalierungs- und Verschiebungsvarianzen auf und keine intrinsischen Deformationen. Verschiebungsinvariantes Sparse Coding in

Verbindung mit Techniken wie Auflösungspyramiden kann diese Invarianzen abbilden. Demgegenüber sind menschliche Bewegungen durch lokale zeitliche Variationen charakterisiert. Während bei routinierten Bewegungen einer Einzelperson noch in gewissem Maße angenommen werden kann, dass die Bewegungen auch zeitlich sehr ähnlich sind, so ist spätestens bei personenunabhängiger Modellierung von Gesten anzunehmen, dass die zeitlichen Variationen relativ stark sind. Eine durch Sparse Coding gelernte Bewegungsprimitive kann nur eine diskrete und statische Folge räumlicher Koordinaten abbilden. Zeitliche Variationen können nicht innerhalb der Primitive modelliert werden (andere Methoden, wie Dynamic Movement Primitives (DMP) [Schaal, 2003] oder Hidden-Markov-Modelle (HMM) [Rabiner, 1989] modellieren diese zeitlichen Variationen explizit. Das Lernen vieler solcher Primitive in Form vom DMP oder HMM ist allerdings theoretisch im Vergleich zum Lernen statischer Primitive sehr komplex und zum aktuellen Zeitpunkt noch nicht hinreichend praktikabel gelöst).

Dieses Problem kann teilweise durch die folgenden Maßnahmen gemindert werden. Zum einen können die Bewegungsprimitive durch Definition des entsprechenden Parameters zeitlich sehr kurz gemacht werden. Je kürzer die Primitive sind und damit die Abdeckung eines Teils der Geste durch eine Primitive, desto weniger zeitliche Variation kann innerhalb des durch die Primitive abgedeckten Teils der Trajektorie vorkommen. Wenn man allerdings die Primitive kürzer macht, geht auch der Effekt der Codierung verloren, da die Primitive dann keine charakteristischen und bedeutungstragenden (also „seltenen“) Teile von Gesten modellieren. Hier muss also im konkreten Anwendungsfall abgewogen werden, wie kurz die Primitive sein dürfen, um zwar zeitliche Variationen zuzulassen, aber dennoch nützlich im Sinne der Codierung zu sein.

Eine andere Maßnahme wäre, die Primitive dennoch relativ lang zu lassen, dafür aber die Anzahl der Primitive zu erhöhen und zu hoffen, dass sich für verschiedene zeitliche Varianten einer Bewegung unterschiedliche Primitive ausbilden. Damit würden allerdings diese zeitlichen Varianten auf unterschiedliche „Codes“ abgebildet werden und die nachgelagerte Interpretation müsste dafür sorgen, den Zusammenhang dieser Codes wiederherzustellen. Das Problem wäre also nur verlagert. Des Weiteren bedeutet eine höhere Anzahl an Primitiven auch eine Erhöhung der Dimension des Lösungsraumes, womit zur Ausbildung hinreichend guter Primitive auch wesentlich mehr Trainingsdaten notwendig sind.

Eine weitere Möglichkeit zur Lösung des Problems ist, die zeitliche Skalierung explizit zu behandeln und die Primitive neben der Verschiebungsinvarianz mit einem weiteren Parameter für die zeitliche Skalierung zu versehen. Hierbei handelt es sich um eine mögliche Erweiterung des Verfahrens, welche detaillierte in Abschnitt 6.3 betrachtet wird.

Neben diesem grundsätzlichen Problem haben sich im praktischen Einsatz einige weitere Probleme gezeigt, deren Klärung weitere Untersuchungen erfordert, die aber grundsätzlich lösbar erscheinen. Eine dieser Beobachtungen betrifft die Anzahl der zu lernenden Primitive. Mit allen in dieser Arbeit für Experimente verwendeten Datensätze hat sich gezeigt, dass ab ca. 30 Primitive sowohl Duplikate auftreten, als auch teilweise

Primitive im Wesentlichen ihre initialen Werte beibehalten und somit nicht adaptiert worden sind. Dies kann darin begründet sein, dass die Menge der Trainingsdaten für eine, durch Erhöhung der Anzahl der Primitive wachsende, Anzahl zu lernender Parameter nicht ausreicht. Dies konnte jedoch nicht belegt werden und bedarf weiterer Untersuchungen.

Ein weiterer Nachteil des Verfahrens ist der relativ hohe Rechenaufwand und der iterative Charakter des Lösungsalgorithmus. In der Anwendungsphase muss zur Ermittlung der spärlichen Aktivierungen ein Gradientenabstieg durchgeführt werden. Unter Umständen muss dieser zur Ermittlung eines guten lokalen Minimums sogar mehrfach mit unterschiedlichen Startwerten durchgeführt werden. Im Vergleich zu geschlossenen Lösungsverfahren, welche bei üblichen Merkmalsextraktionsverfahren für zeitliche Signale (z. B. DFT oder PCA) eingesetzt werden, ist das Sparse Coding damit relativ langsam. Dieses Problem kann vor allem durch Untersuchung von Alternativen zum Gradientenabstieg behoben werden. In Abschnitt 6.3.2 werden zwei Alternativen (Matching Pursuit und geschlossene Lösungsverfahren) betrachtet.

6.2. Erreichung der angestrebten Ziele der Arbeit

Im Folgenden werden die zu Beginn dieser Arbeit definierten Ziele (siehe Abschnitt 1.2) aufgegriffen und eine Bewertung der Erfüllung dieser Ziele durchgeführt.

Entwicklung eines generischen Verfahrens zum Lernen von Primitiven in Bewegungsdaten und Untersuchung der Anwendbarkeit Es wurde ein Verfahren für die Anwendung von Sparse Coding auf Bewegungstrajektorien auf Grundlage vorangegangener Arbeiten weiterentwickelt. Weiter wurde eine Untersuchung der Eignung von Sparse Coding für die Merkmalsextraktion auf Bewegungstrajektorien durchgeführt. Es konnte belegt werden, dass mit Hilfe des entwickelten Verfahrens Bewegungsprimitive bestimmt werden können, die nützliche Merkmale für die Klassifikation und Generierung von Trajektorien sind. Das Verfahren ist aber nur unter bestimmten Randbedingungen einsetzbar. So müssen in den Daten klare, wiederkehrende Muster vorhanden sein, welche die Daten eindeutig charakterisieren. Insbesondere darf die zeitliche Variation innerhalb der Muster nicht zu groß sein. Des Weiteren eignet sich das Verfahren nur für Probleme mit einer beschränkten Anzahl an Klassen. So zeigt das Verfahren auf Datensätzen wie dem Buchstaben-Datensatz (siehe Abschnitt 3.3.2) mit Beschränkung auf wenige Klassen gute Ergebnisse, da die Varianz innerhalb der Klassen relativ gering ist. Auch auf Datensätzen aus der Activity Recognition (siehe Abschnitt 3.3.3) mit wiederkehrenden, routinierten Bewegungen, welche ebenfalls eine geringe Intra-Klassenvarianz aufweisen, arbeitet das Verfahren gut. Bei erhöhter Intra-Klassenvarianz zeigt das Verfahren allerdings keine guten Ergebnisse, da starke Varianzen, insbesondere lokale zeitliche Variationen, durch die statischen Features nicht gut abgebildet werden können.

Die Güte der Klassifikation ist zudem sehr stark von den Parametern der Merkmalsex-

traktion durch Sparse Coding abhängig. Diese müssen aufwändig an die Charakteristika der Daten angepasst werden (z. B. ist die zeitliche Länge der Primitive abhängig von der durchschnittlichen Länge der in den Daten vorkommenden Muster). Dieses Problem wird etwas abgeschwächt, wenn man eine systematische Suche nach den optimalen Parametern durchführen kann, was aber die Verfügbarkeit von entsprechend hohe Rechenkapazität voraussetzt.

Finden einer symbolischen Repräsentation für Bewegungstrajektorien Eines der Ziele dieser Arbeit war das Finden einer Transformation hin zu einer symbolischen Beschreibung von Trajektorien in Form von Symbolfolgen. Das Mittel dazu sollte das in dieser Arbeit entwickelte Verfahren sein, indem es zunächst durch Lernen der Primitive ein Symbolalphabet aus den Daten lernt und dann durch Codierung die Trajektorie als Folge von Symbolen darstellt. Die Symbolkette muss dabei aus den Aktivierungen nach Durchführung der Codierung abgeleitet werden. Dies ist allerdings nur dann möglich, wenn die Aktivierungen entsprechend spärlich, diskret und klar definiert sind. Bei Verwendung des auf Gradientenabstieg basierenden Lösungsverfahrens ist diese Voraussetzung allerdings nicht hinreichend gut erfüllt. Die Aktivierungen sind bei realen Daten, selbst bei Einsatz der Erweiterung um lokale Spärlichkeit (siehe Abschnitt 4.2.5), noch zu sehr gestreut und nicht spärlich genug, um eine direkte Ableitung einer Symbolkette zu erlauben. Dies hat sich auch bei dem Versuch eines auf Symbolketten basierenden Klassifikators, des Activity-String-Matching (ASM) (siehe Anhang D.2), gezeigt. Um solche Symbolketten dennoch ableiten zu können, ist eine weitere Bearbeitung der Aktivierungen durch Schwellwertoperationen notwendig. Um klar definierte Symbolketten zu erhalten, müssen allerdings die Schwellen so hoch definiert werden, dass wesentliche Teile der Trajektorien aufgrund geringer, verteilter Aktivierungen ausgelöscht werden und die Trajektorien für die Klassifikation nicht mehr hinreichend gut repräsentiert sind.

Abhilfe kann auch hier die Verwendung von Matching Pursuit schaffen (siehe Abschnitt 6.3), denn hier wird die Spärlichkeit direkt durch sukzessive Auswahl von Kandidatenaktivierungen erreicht, was zwangsläufig zu diskreten und klar isolierten Aktivierungen führt, die eine Ableitung von Symbolkette viel eher zulassen würden.

Aufgrund der mäßigen Resultate der Symbolkettentransformation wurde auch einer der zu Beginn der Arbeit angestrebten Pfade hin zu einer Klassifikation auf Basis solcher Symbolketten aufgegeben. Stattdessen wurden alternative Methoden im Sinne einer Bag-of-Words-Transformation (siehe Abschnitt 5.2.2) untersucht. Diese haben im Rahmen der oben erläuterten Einschränkungen sehr gute Ergebnisse gezeigt.

Generierung und Klassifikation mit sehr einfachen Modellen auf Basis spärlich codierter Bewegungen Das Ziel der Entwicklung eines Klassifikators und eines Generators für Trajektorien auf Basis der in dieser Arbeit entwickelten Merkmalsextraktion konnte generell erreicht werden. Die Rahmenbedingungen für die Erzielung guter Ergebnisse sind allerdings sehr strikt. Im realen Einsatz sind die Ergebnisse im Vergleich

mit Verfahren aus dem aktuellen State-of-the-Art teilweise wesentlich schlechter, die prinzipielle Machbarkeit konnte allerdings belegt werden. Für den praktischen Einsatz ist das Verfahren im aktuellen Entwicklungsstand damit nicht geeignet. Die möglichen Erweiterungen des Verfahrens, wie sie in Abschnitt 6.3 erläutert werden, könnten hier allerdings substanzielle Verbesserungen bringen. Dies wird teilweise auch durch ähnliche Arbeiten anderer Forschungsgruppen angedeutet (mehr dazu in Abschnitt 6.3.1).

Demonstrator-Applikation mit echtzeitfähiger Implementierung Im Rahmen dieser Arbeit wurde ein Gesamtsystem entwickelt, welche alle Verarbeitungsschritte eines Mustererkennungsprozesses, von der Datenaufnahme mittels Tiefenkameras, über die Vorverarbeitung, die Merkmalsextraktion mittels Sparse Coding, bis hin zur Klassifikation bzw. Generierung implementiert. Das System wurde mit den Robotik-Framework MIRA modular als verteiltes System umgesetzt. Insbesondere wurde das rechenaufwändige Sparse Coding mit Hilfe des Frameworks OpenMP durch Parallelverarbeitung echtzeitfähig gemacht. Das angestrebte Ziel der Implementierung eines echtzeitfähigen Demonstrators konnte also erreicht werden.

Assistent zur Unterstützung der Bewegungsrehabilitation Eines der ursprünglichen Nebenziele dieser Arbeit war die Entwicklung eines Assistenten zur Überwachung und Unterstützung der Bewegungsrehabilitation auf Basis des entwickelten Verfahrens. Der Assistent sollte dabei im Dialog mit dem Benutzer die Durchführung einer Reihe von Übungen überwachen und während oder nach der Durchführung Feedback über die Qualität der Ausführung geben. Wenn z. B. der Benutzer bei der Durchführung einer Übung mit dem Arm örtlich oder zeitlich signifikant von der vorgegebenen Idealtrajektorie der Übung abgewichen ist, so sollte der Assistent darauf akustisch oder visuell hinweisen und den Benutzer so bei der Korrektur bei erneuter Ausführung der Übung unterstützen.

Um diese Funktionalität umzusetzen, ist im Kern ein Maß zur Bewertung der Ausführungsqualität notwendig. Da sich die Arbeit vorrangig mit der Merkmalsextraktion durch Sparse Coding befasst, sollte dieses Maß auch aus diesem Rahmen heraus definiert werden. Die Idee war, Abweichungen aus der optimalen Rekonstruktion der Trajektorie und der beobachteten Trajektorie zu messen und so ein Differenzmaß zwischen Modell und Beobachtung zu definieren. Hier konnte allerdings kein Maß gefunden werden, welches sich für diese Zwecke eignet. Die Definition eines Maßes aus dem Sparse Coding heraus scheint für diese Anwendung ungeeignet.

6.3. Erweiterungen und aufbauende Arbeiten

Dieser Abschnitt beschreibt eine Reihe möglicher Erweiterungen des Verfahrens, welche in aufbauenden Arbeiten untersucht werden könnten. Diese Erweiterungen ergeben

sich hauptsächlich aus Problemen des Verfahrens, die während dieser Arbeit beobachtet wurden.

6.3.1. Modellierung weiterer Invarianzeigenschaften

Eines der Probleme des Verfahrens ist der Mangel an für die Modellierung menschlicher Bewegungen notwendigen Invarianzeigenschaften. Im aktuellen Entwicklungsstand ist im Verfahren die zeitliche Verschiebungsinvarianz formuliert. Damit ist es möglich, wiederkehrende temporale Muster, unabhängig von ihrem zeitlichen Auftreten zu modellieren und zu lernen. Dies ist die Grundvoraussetzung für die Anwendung auf Bewegungstrajektorien. Wie in Abschnitt 6.1 beschrieben, sind Bewegungstrajektorien aber vor allem auch durch lokale zeitliche Variationen charakterisiert. Diese können mit dem Verfahren bisher nicht explizit behandelt werden. Eine weitere wünschenswerte Invarianzeigenschaft ist die Rotationsinvarianz, bei der die Bewegungsprimitive beliebig rotiert werden können. Diese beiden Invarianzeigenschaften werden im Folgenden näher betrachtet.

Skalierungsinvarianz

Eine Möglichkeit, die zeitliche Skalierung zu behandeln ist, die Primitive neben der Verschiebungsinvarianz mit einem weiteren Parameter für die zeitliche Skalierung zu versehen. Das Optimierungsverfahren wird also um einen zusätzlichen Freiheitsgrad erweitert und die Primitive können sowohl zeitlich verschoben als auch in gewissen Grenzen skaliert werden. Damit ist zwar weiterhin nicht die intrinsische zeitliche Deformation einer Primitive abgebildet, aber zumindest eine homogene zeitliche Skalierung.

Der Nachteil dieser Erweiterung ist der immens erhöhte Rechenaufwand durch die Erhöhung der Dimension des Lösungsraumes. Im aktuellen Verfahren bilden die Aktivierungen einen 3-Tensor mit den drei Indizes für Eingangsdatum, Basisvektor und zeitliche Verschiebung. Bei einer Erweiterung um zeitliche Skalierung wäre ein 4-Tensor notwendig. Entsprechend erhöhte sich die Anzahl der möglichen Lösungen exponentiell. Durch die Forderung der Spärlichkeit ist es dennoch durchaus denkbar, dass sich gute Lösungen finden lassen. Ein größeres Problem dieser Erweiterung ist jedoch die effiziente Implementierung. Während sich die Verschiebungsinvarianz effizient mit Hilfe der DFT implementieren lässt (die notwendigen Faltungen werden im Fourierraum als Multiplikationen implementiert), gibt es für die Skalierungsinvarianz kein solches Hilfsmittel. Die Implementierung müsste also so erfolgen, dass man eine Menge diskreter Skalierungsstufen definiert (die Aktivierungen für die Skalierungen eines Basisvektors werden in der vierten Dimension des Aktivierungstensors gehalten) und in einem Iterationsschritt des aktuellen Verfahrens zunächst in einer weitere Schleife die Korrelationen für jede Skalierungstufe der Basisvektoren ermittelt und dann über einen neu zu entwickelnden Spärlichkeitsterm die Spärlichkeiten zwischen den skalierten Varianten eines Basisvektors so koppelt, dass nur jeweils eine Skalierungstufe aktiviert wird. Neben dem

immens erhöhten Rechenaufwand durch die zusätzliche innere Schleife, käme also auch ein weiterer Spärlichkeitsparameter hinzu, um den neuen Term zu kontrollieren, was zusammen mit den anderen Parametern einen immensen Aufwand für das Tuning nach sich ziehen würde. Eine Alternative könnte auch hier wieder der Einsatz von Matching Pursuit sein, welches Kandidaten für Aktivierungen nach bestimmten Kriterien iterativ explizit auswählt und so die Spärlichkeitsterme unnötig macht. Darauf wird in Abschnitt 6.3.2 detaillierter eingegangen.

Rotationsinvarianz

Eine weitere wünschenswerte Eigenschaft ist die Rotationsinvarianz. Eine Bewegung soll im Idealfall auch unter beliebiger Rotation wiedererkannt werden können. Dabei können Rotationen prinzipiell durch zwei Ursachen bedingt sein. Zum einen kann das Aufnahmesystem relativ zur Person rotiert sein, zum anderen kann die Person die Geste tatsächlich rotiert ausführen (z. B. eine Handgeste in verschiedenen Stellungen des Armes). Die erste Ursache wurde in dieser Arbeit durch einen Schritt in der Vorverarbeitung behandelt, indem Bewegungstrajektorien in das egozentrische Koordinatensystem der Person transformiert werden (siehe Abschnitt 3.2.1). Damit können relative Rotationen und Translationen zwischen Kamera und Person eliminiert werden. Die zweite Ursache kann so nicht ausgeschlossen werden. Hier könnte eine rotationsinvariante Codierung helfen. Dabei würden die Bewegungsprimitive um einen Rotationsparameter erweitert (eigentlich mehrere Parameter, da die Rotation im dreidimensionalen Raum durch drei Winkel beschrieben werden kann). Wie schon oben bei der zeitlichen Skalierungsinvarianz beschrieben worden ist, erhöht das aber die Komplexität des Problems enorm und ist mit dem in dieser Arbeit entwickelten und auf Gradientenabstieg basierenden Verfahrens nicht mehr effizient lösbar.

In Abschnitt 2.5 werden die Arbeiten einer anderen Forschungsgruppe beschrieben, die ein ähnliches Ziel verfolgen. Dort wird 3D-rotationsinvariantes Sparse Coding durch eine Kombination aus Matching Pursuit und einer Singulärwertzerlegung (engl. Singular Value Decomposition, kurz SVD) implementiert. Bei der Auswahl von Kandidaten-Aktivierungen im Codierungsschritt wird für jede Primitive und jede Aktivierung dieser Primitive (also für jeden Zeitschritt des Signals) eine Optimierung des Rotationsparameters per SVD durchgeführt. Das ist sehr rechenaufwändig, führt aber zum gewünschten Ergebnis. In den Arbeiten wird das Verfahren auf Gesten in der französischen Zeichensprache angewendet. Hier wäre eine Untersuchung der Anwendung dieses Verfahrens auf die in dieser Arbeit verwendeten Datensätze interessant.

Variable Größe der Basisvektoren

Ein weiterer Mangel an der aktuellen Umsetzung des Verfahrens ist die feste zeitliche Länge der Basisvektoren. Die in den Daten vorkommenden Muster sind im Allgemeinen nicht gleich lang. Denn ist ein Basisvektor kürzer als das Muster, so kann nicht das

ganze Muster abgebildet werden, was die Güte der Rekonstruktion beeinträchtigt. Ist der Basisvektor länger als das Muster, so bildet er Teile ab, die eigentlich nicht zum Muster gehören, was ebenfalls die Rekonstruktion beeinträchtigt.

Es wäre wünschenswert, wenn die Länge der Basisvektoren an die Länge des entsprechenden Musters adaptiert werden könnte. Eine Idee für eine Lösung wäre, die Varianz der durch die betrachtete Primitive abgedeckten Teile der Eingangstrajektorien zu untersuchen. Ist die Varianz am Beginn und Ende dieser Teile hoch, so ist davon auszugehen, dass diese Abschnitte nicht zum Muster gehören und der Basisvektor kann verkürzt werden. Ist die Varianz in den benachbarten Abschnitten des abgedeckten Teils der Trajektorien gering, so kann man davon ausgehen, dass diese Abschnitte zum Muster gehören und die Primitive kann verlängert werden.

6.3.2. Verwendung anderer Algorithmen

Eine vielversprechende Erweiterung des Verfahrens ist die Verwendung von Matching Pursuit [Mallat et al., 1993] anstelle des Gradientenabstiegs im Codierungsschritt. Matching Pursuit ist ein heuristisches Verfahren für die Codierung und funktioniert im Prinzip als ein auf Korrelation basierender Feature-Detektor. Es geht iterativ vor und sucht bei einem gegebenen Eingangsdatum und einer gegebenen Basis in einer Iteration die maximale Korrelation über alle Basisvektoren und alle Verschiebungen des jeweiligen Basisvektors. Das die Korrelation maximierende Paar aus Index des Basisvektors und Index der Verschiebung beschreibt die gefundene Aktivierung. Die Rekonstruktion mit nur dieser Aktivierung wird vom Eingangsdatum subtrahiert und so der durch die gefundene Aktivierung abgedeckte Teil eliminiert. Das Ergebnis bildet das Residuum und stellt die Eingabe für den nächsten Iterationsschritt dar. So wird in jedem Iterationsschritt eine weitere Aktivierung gefunden, solange bis ein Konvergenzkriterium erfüllt ist oder die Spärlichkeitsbedingung verletzt wird. Das Verfahren wird detailliert in Anhang C.2 beschrieben. Matching Pursuit wird konventionell in der Audiosignalverarbeitung verwendet und ist aufgrund der einfachen Operationen sehr effizient. Zusätzlich hat es die Eigenschaft, die Spärlichkeit explizit umzusetzen. Ebenso könnte man die lokale Spärlichkeit (siehe Abschnitt 4.2.5) explizit in der Auswahl der Kandidatenaktivierung umsetzen. Damit könnte man die Nachteile des auf Gradientenabstieg basierenden Verfahrens eliminieren und würde per Definition klar abgegrenzte, diskrete Aktivierungen erhalten, ohne aufwändig Parameter tunen zu müssen, um einen Optimierungsprozess zur Ausbildung der gewünschten Spärlichkeit zu zwingen.

Auch für das Lernen der Primitive, den Dictionary-Learning-Schritt, gibt es einige alternative Algorithmen, deren Untersuchung sich lohnen könnte. So beschreibt [Le Roux et al., 2009] ein Verfahren, welches durch geschickte Formulierung eine Lösung mittels geschlossener Lösungsverfahren möglich macht. Damit könnten die Geschwindigkeitsnachteile des iterativen Gradientenabstiegs eliminiert werden.

Die bisher genannten Ansätze formulieren die Verschiebungsinvarianz explizit. In der Bildverarbeitung hat sich in den letzten Jahren der Ansatz durchgesetzt, die Verschie-

bungsinvarianz nicht explizit durch verschiebbare Features zu formulieren, sondern Feature-Detektoren auf Bild-Patches zu lernen. Dazu werden die Trainingsbilder in Patches zerschnitten, auf denen die Features gelernt werden. Wenn wiederkehrende Features in den Bildern existieren, dann gibt es auch gewisse statistische Häufungen von ähnlichen Patches, auf die die Feature-Detektoren automatisch trainiert werden. Es wäre interessant, diesen Ansatz auch auf Bewegungstrajektorien zu testen. Damit wären viele für die Bildverarbeitung entwickelte Algorithmen auch auf Bewegungstrajektorien anwendbar.

6.3.3. Verbesserungen des Klassifikators

Die bisherigen Vorschläge für Erweiterungen beziehen sich nur auf die Merkmalsextraktion durch Sparse Coding. Im Klassifikationsschritt sind auch einige offensichtliche Erweiterungen möglich. So könnte zum Beispiel vor der eigentlichen Klassifikation eine Feature-Selektion auf den gelernten Features durchgeführt werden. Während beim Lernen noch nicht ihre Relevanz für die spätere Klassifikation mit einbezogen wird, könnten hier im Nachhinein für die Klassifikation nicht relevante Merkmale eliminiert werden.

Des Weiteren könnte die Nützlichkeit der Beziehungen zwischen Primitiven innerhalb einer Trajektorie, wie Timing und Ordnung, für die Klassifikation weiter untersucht werden. In dieser Arbeit wurde dazu mit dem Activity-String-Matching (ASM) (siehe Anhang D.2) bereits der Versuch unternommen. Allerdings waren damit die Klassifikationsergebnisse nicht besser als mit den Bag-of-Words-Modellen (welche die Beziehungen zwischen den Primitiven ignorieren). Es wurden auch Versuche mit dem Einsatz eines Hidden Markov Model (HMM) zur Modellierung von Sequenzen von Aktivierungen unternommen, die ebenfalls keine besseren Ergebnisse zeigten. Die Ursache dafür könnte zum Teil in der geringen Komplexität der Daten liegen, die eine solche Modellierung einfach nicht erfordert. Weiterhin ist zum Einsatz solcher Modelle eine weitere Bearbeitung der Aktivierungen nach der Codierung, z. B. durch Schwellwertoperationen, nötig, die teilweise den Informationsgehalt stark verringert und so das Ergebnis negativ beeinflusst.

6.4. Fazit

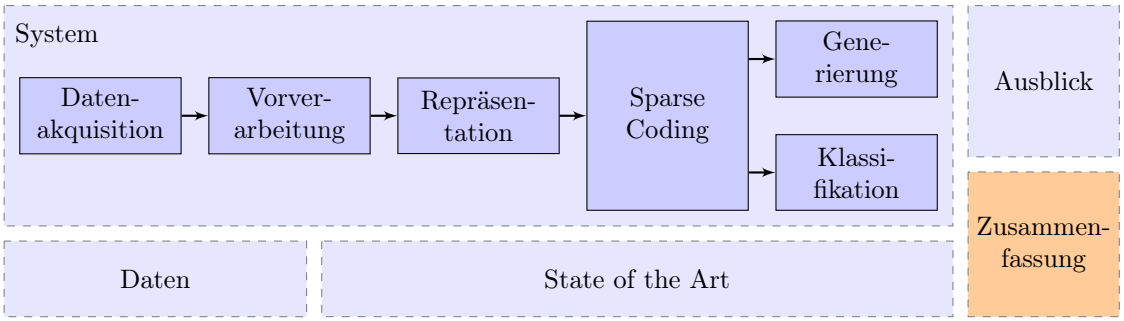
In diesem Kapitel wurden einige kritische Punkte und Mängel am aktuellen Entwicklungsstand des Verfahrens betrachtet. Des Weiteren wurden mögliche Lösungen für die Probleme und Ideen für aufbauende Arbeiten vorgestellt.

Das Fazit der kritischen Betrachtungen ist, dass in dieser Arbeit in Form einer Machbarkeitsstudie belegt werden konnte, dass man das Sparse Coding in der implementierten Form auf Bewegungstrajektorien anwenden kann, dass sich damit Primitive als nützliche Features für die Klassifikation und zur abstrakten Beschreibung von Trajektorien lernen lassen.

Für den realen Einsatz ist das Verfahren in der aktuellen Form jedoch noch nicht hinreichend weit entwickelt. Dazu muss es vor allem um die Abbildung weiterer wichtiger Invarianzeigenschaften erweitert werden. Weiterhin muss das Verfahren durch Einsatz anderer Algorithmen für die Codierung beschleunigt werden.

Kapitel 7

Zusammenfassung



Nachdem in Kapitel 6 eine kritische Betrachtung der Ergebnisse und ein Ausblick auf mögliche Themenstellungen für aufbauende Arbeiten dargestellt wurde, soll in diesem Kapitel das Anliegen und die grundlegenden Erkenntnisse dieser Arbeit noch einmal zusammengefasst werden.

Die ursprüngliche Motivation für die tiefer gehende Untersuchung der Eignung von Sparse Coding für Bewegungstrajektorien waren vielversprechende Ergebnisse aus einer vorangegangenen Arbeit von Sven Hellbach [Hellbach, 2009]. Nach anfänglichen Recherchen zeigte sich, dass Sparse Coding in der Vergangenheit hauptsächlich für Bilddaten und Audiosignale, jedoch noch nicht für Signale aus der Gestenerkennung eingesetzt worden ist. Die Ergebnisse auf Audiosignalen und Arbeiten zur Modellierung neuronaler Mechanismen zur Steuerung von Beinmuskeln von Fröschen durch Sparse Coding, ließen aber den Schluss zu, dass sich eine weitere Exploration der Eignung für solche Signale lohnt.

Die grundlegende Idee des Verfahrens war, Bewegungen als Aneinanderreihung primitiver Teilbewegungen (die sogenannten Basisprimitive) zu modellieren, die ihrerseits eine Art domänenspezifisches Alphabet konstituieren. Alle Bewegungen der betrachteten Anwendungsdomäne sollten sich aus den Primitiven des Alphabetes zusammensetzen lassen. Zum Finden solcher Primitive in einem Datensatz aus einer beliebigen Domäne wurde das Sparse Coding eingesetzt. Das Sparse Coding schien für diesen Zweck geeignet, weil durch Kontrolle der Spärlichkeitsbedingungen die Herausbildung solcher Komponenten erzwungen werden kann.

Um die tatsächliche Eignung des Verfahrens für die Dekomposition von Bewegungstrajektorien zu belegen, wurde es für die Verwendung auf Bewegungstrajektorien angepasst. Dazu musste insbesondere eine weitere Spärlichkeitsbedingung formuliert werden, welche die Herausbildung spärlicher Codes auf Bewegungstrajektorien unterstützt. Mit dieser Anpassung wurde das Verfahren auf frei verfügbaren Benchmark-Datensätzen in Form einer Merkmalsextraktion mit anschließender Klassifikation getestet. Für die Klassifikation konnten dafür sehr einfache Naive-Bayes-Modelle verwendet werden. Im einfachsten Fall wurde dazu aus den im Eingangsdatum gefundenen Primitiven ein binärer Feature-Vektor gebildet, der mit einem Naive-Bayes-Modell mit Bernoulli-Verteilung klassifiziert wurde. Diese Feature-Repräsentation wird häufig auch als Bag-of-Words-Modell bezeichnet. Hier konnte gezeigt werden, dass mit dem Ansatz ein Lernen von nützlichen Merkmalen für die Klassifikation möglich ist. Dabei hat das Verfahren insbesondere bei Datensätzen aus der Activity Recognition im Vergleich mit konventionellen Merkmalsextraktionsverfahren für zeitliche Signale gut abgeschnitten.

Des Weiteren wurde das Verfahren im Sinne eines Proof-of-Concept für die Generierung neuer Trajektorien auf Basis der gelernten Primitive getestet. Dazu wurden Primitive aus Handschrift-Trajektorien gelernt. Die Statistik der typischen Aktivierungsmuster innerhalb einzelner Buchstaben-Klassen wurde ebenfalls gelernt und anschließend verwendet, um neue Buchstaben bestimmter Klassen zu erzeugen. Die Resultate haben gezeigt, dass auch die Generierung prinzipiell möglich ist.

Um die Eignung des Verfahrens für Echtzeit-Anwendungen zu untersuchen, wurde eine Demonstrator-Applikation entwickelt, welche alle Stufen des Mustererkennungsprozesses, von der Aufnahme von Trajektorien von Körperposen mittels Tiefenkameras, über die Vorverarbeitung, die Merkmalsextraktion mit dem entwickelten Verfahren, bis zur Klassifikation implementiert. Da die Anwendung des Sparse Coding sehr rechenaufwändig ist, mussten die Algorithmen so formuliert werden, dass sie parallelisierbar sind. Mittels Parallelprogrammierung konnte so eine hinreichend schnelle Implementierung des Verfahrens geschaffen werden. Mit der Demonstrator-Applikation konnte gezeigt werden, dass eine Verwendung des Verfahrens unter Echtzeit-Bedingungen möglich ist.

Der anwendungsorientierte Rahmen der Arbeit sollte die Entwicklung eines Assistenten zur Unterstützung in der Bewegungsrehabilitation sein. Dabei sollte das Verfahren eingesetzt werden, um die Ausführung bestimmter Bewegungsübungen zu überwachen und zu bewerten. Wenngleich dieses Konzept in dieser Arbeit nicht umgesetzt werden konnte, so kann die Demonstrator-Applikation als ein Schritt auf dem Weg zu einem vollständigen System gesehen werden.

Da die vorliegende Arbeit explorativer Natur ist, muss an ihrem Ende auch eine abschließende Bewertung der Untersuchungen stehen. Wie schon oben beschrieben, konnte belegt werden, dass mit dem entwickelten Verfahren erfolgreich Primitive aus Bewegungsdaten gelernt werden können. Die gelernten Komponenten entsprechen der Modellvorstellung, ein Alphabet zu konstituieren, welches die Bestandteile der Trajektorien im Datensatz enthält. Natürlich kann nicht davon ausgegangen werden, dass damit ein tatsächlich den Bewegungen zugrunde liegendes Alphabet gefunden wird. Es wird le-

diglich eine Dekomposition gefunden, welche den Nebenbedingungen des Optimierungsproblems entspricht. Man kann also nicht behaupten, die „wahren“ Grundbestandteile der Bewegungen gefunden zu haben.

Es konnte aber gezeigt werden, dass sich die gefundenen Primitive in der Interpretation als Features von Bewegungstrajektorien für die Klassifikation eignen und dass mit ihnen eine Generierung neuer Trajektorien realisiert werden kann. Unter engen Rahmenbedingungen ist das Sparse Coding als Merkmalsextraktion konventionellen Verfahren hinsichtlich der damit erreichten Klassifikationsleistung überlegen.

Problematisch ist jedoch der hohe Aufwand für die Parametersuche. Die Parameter beeinflussen die Güte des Ergebnisses sehr empfindlich, und es gibt Abhängigkeiten zwischen den Parametern. Ein weiteres Problem des Verfahrens ist der Mangel an für die Verarbeitung von Bewegungstrajektorien wichtigen Invarianzeigenschaften. Hier hat das Konzept klare Nachteile gegenüber anderen Verfahren aus dem State-of-the-Art. Im Ausblick wurden allerdings eine Reihe möglicher Veränderungen und Erweiterungen aufgezeigt, die diese Probleme beheben können.

Insgesamt ist das Sparse Coding eine interessante Alternative zu konventionellen Merkmalsextraktionsverfahren, auch deshalb, weil es sich in den aktuellen Trend der Feature-Learning-Verfahren, wie den Deep Belief Networks, einreiht. Eine weitergehende Untersuchung gerade im Hinblick auf die vorgeschlagenen Verbesserungen könnte sich also lohnen.

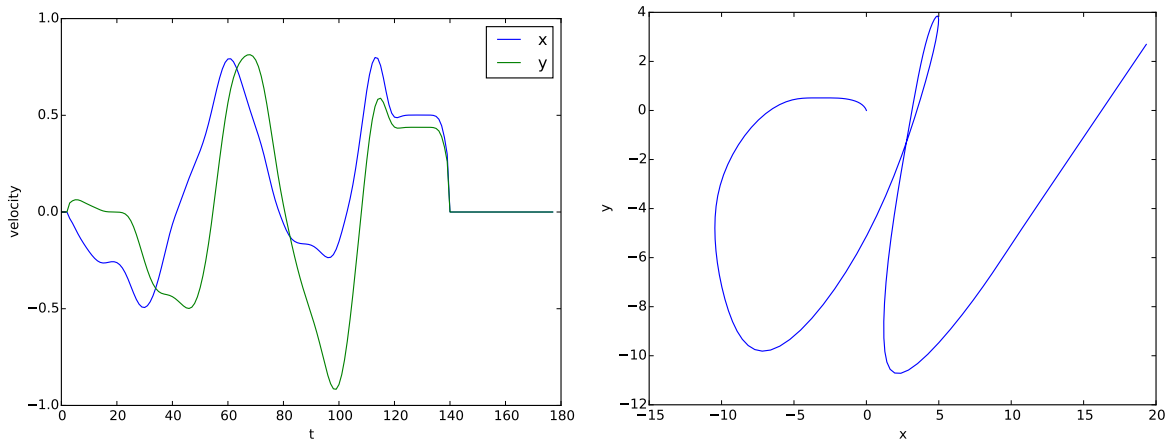
Anhang A

Visualisierung der Test-Datensätze

Um einen visuellen Eindruck der für die Experimente in dieser Arbeit verwendeten Daten zu vermitteln, werden hier Exemplare bzw. Ausschnitte der Daten gezeigt.

A.1. Datensatz „Character Trajectories Data Set“

Abbildung A.1 zeigt den Geschwindigkeitsverlauf und den Positionsverlauf eines Exemplars des Buchstabens „a“ aus dem Datensatz „Character Trajectories Data Set“ (siehe Abschnitt 3.3.2 für weitere Details).

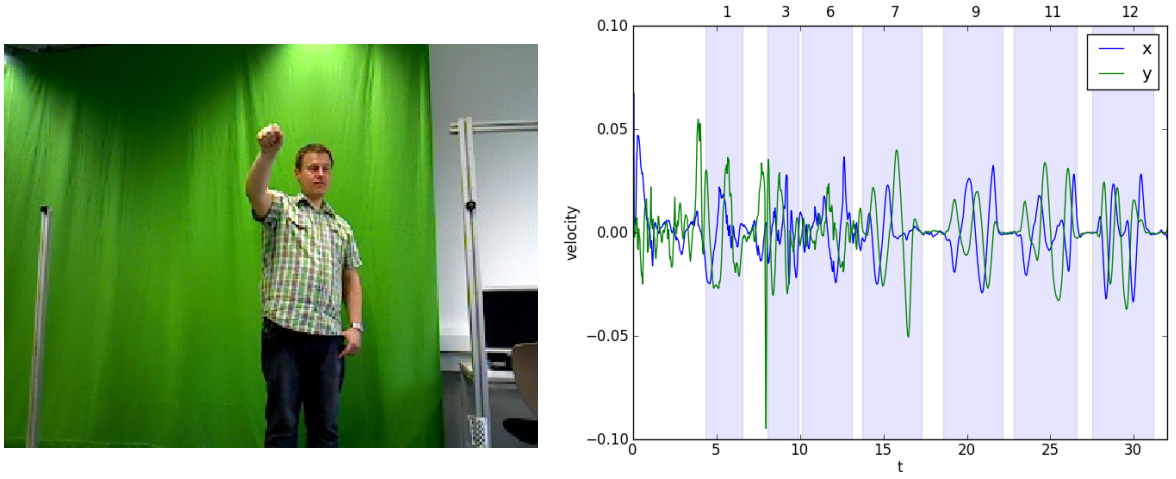


(a) (b)

Abb. A.1.: *(a) Verlauf der Geschwindigkeiten in x- und y-Richtung beim, Schreiben des Buchstabens „a“, bezogen auf das Koordinatensystem des Grafiktabletts. (b) Aus dem Geschwindigkeitsverlauf rekonstruierter Verlauf der Position in x-y-Koordinaten (nur zur Veranschaulichung der Gestalt der Buchstaben).*

A.2. Demonstrator-Datensatz

Abbildung A.2 zeigt eine Person während der Aufnahme des im Rahmen dieser Arbeit erstellten Datensatzes (siehe Abschnitt 3.3.1 für weitere Details). Daneben ist ein Ausschnitt aus den Daten dargestellt.



(a) (b)

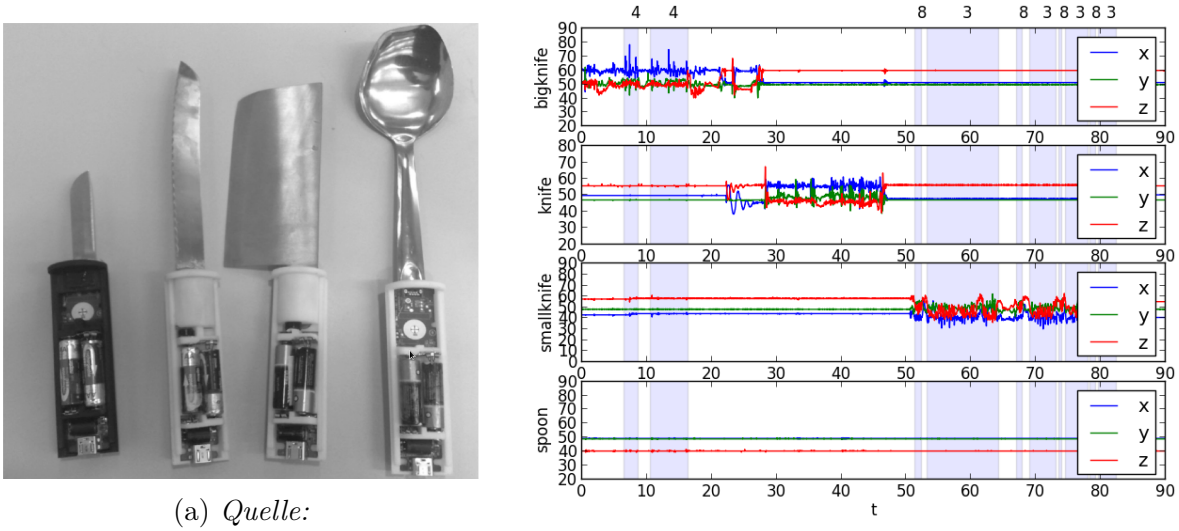
Abb. A.2.: *(a) Person während der Aufnahme bei Durchführung einer Geste (malen eines Buchstabens „in die Luft“). (b) Ausschnitt aus den Daten. Die Ordinate zeigt die Geschwindigkeit der in die Frontalebene projizierten Trajektorie. Die Intervalle, in denen eine Buchstabe gemalt worden ist, sind blau hinterlegt. Das Label des jeweiligen Buchstabens (die Buchstaben sind nummeriert) ist über dem blauen Bereich annotiert.*

A.3. Benchmark-Datensätze aus dem Bereich Activity-Recognition

Im Folgenden sind Ausschnitte von Datensätzen aus dem Bereich der Activity Recognition dargestellt. Diese Datensätze wurden verwendet, um die Eignung des in dieser Arbeit entwickelten Verfahrens für die Activity Recognition zu prüfen und das Verfahren mit anderen Publikationen vergleichen zu können, welche mit diesen Daten arbeiten (siehe Abschnitt 5.2.3 für weitere Details).

A.3.1. Datensatz „Ambient Kitchen 1.0“

Abbildung A.3 zeigt die zur Aufnahme der Daten für den Datensatz „Ambient Kitchen 1.0“ (siehe Abschnitt 3.3.3 für weitere Details) verwendeten Küchenutensilien und einen Ausschnitt der Daten.



(a) *Quelle: [Pham et al., 2009]*

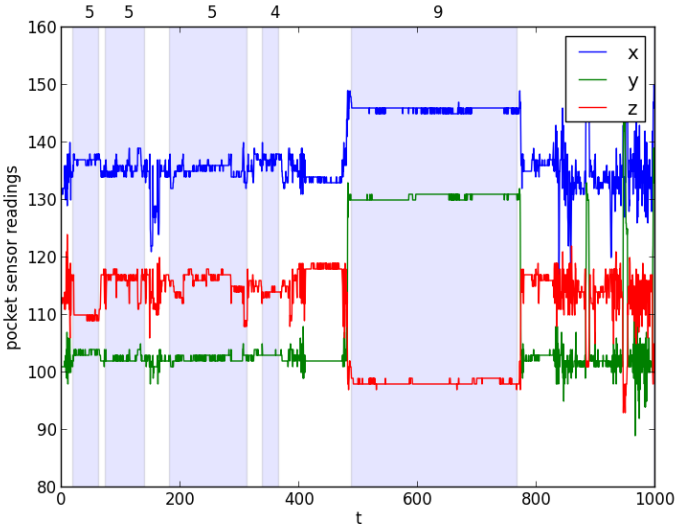
(b)

Abb. A.3.: *(a) Küchenutensilien, welche mit triaxialen Beschleunigungssensoren ausgestattet wurden und der Aufnahme von Aktivitäten bei der Zubereitung von Mahlzeiten dienen. (b) Ausschnitt der Rohdaten. Die Ordinate misst die Beschleunigungswerte im Rohmaß der Sensoren. Die Intervalle, in denen eine Aktivität stattgefunden hat, sind blau hinterlegt. Das Label der jeweiligen Aktivität ist über dem blauen Bereich annotiert.*

A.3.2. Datensatz „Darmstadt Daily Routines“

Abbildung A.4 zeigt Personen, welche mit Beschleunigungssensoren am Oberkörper und am Arm ausgestattet sind, bei der Ausführung von Haushaltsaktivitäten. Daneben wird ein Ausschnitt der so für den Datensatz „Darmstadt Daily Routines“ (siehe Abschnitt 3.3.3 für weitere Details) gewonnenen Daten gezeigt.

(a) *Quelle: [Stikic et al., 2008]*

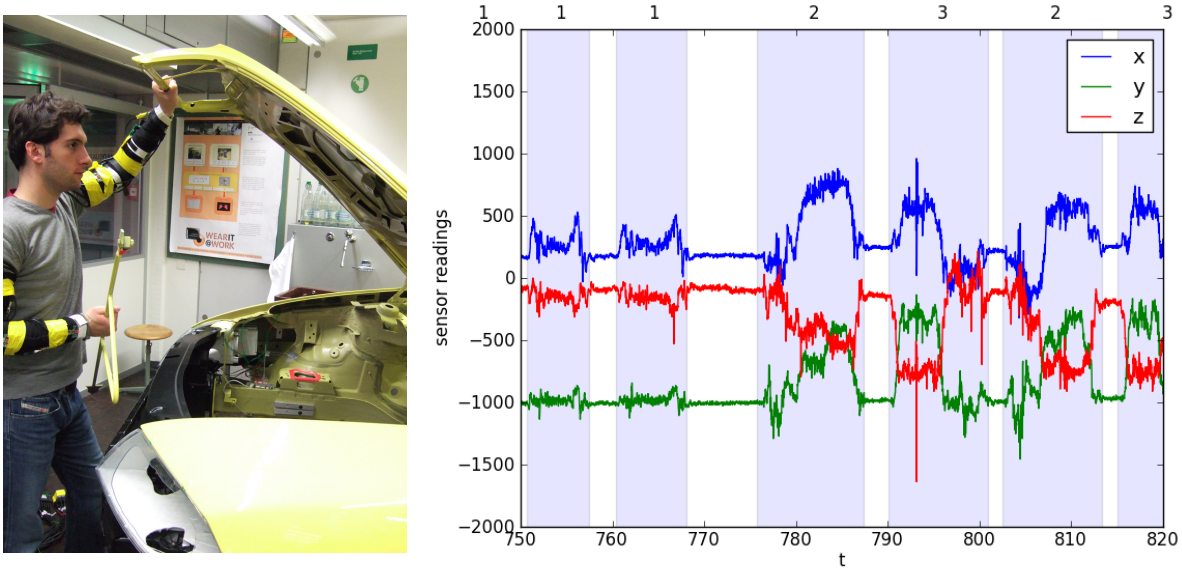


(b)

Abb. A.4.: *(a) Personen bei der Durchführung von Haushaltsaktivitäten. (b) Auszug aus den Beschleunigungswerten des Sensors am Körper. Die Ordinate misst die Beschleunigung im Rohmaß der Sensoren. Die Intervalle, in denen eine Aktivität stattgefunden hat, sind blau hinterlegt. Das Label der jeweiligen Aktivität ist über dem blauen Bereich annotiert.*

A.3.3. Datensatz „Skoda Mini Checkpoint“

Abbildung A.5 zeigt eine mit Beschleunigungssensoren an den Armen ausgestattete Person bei der Durchführung von Wartungstätigkeiten an einem Fahrzeug. Daneben ist ein Ausschnitt des dadurch erzeugten Datensatzes „Skoda Mini Checkpoint“ (siehe Abschnitt 3.3.3 für weitere Details) dargestellt.

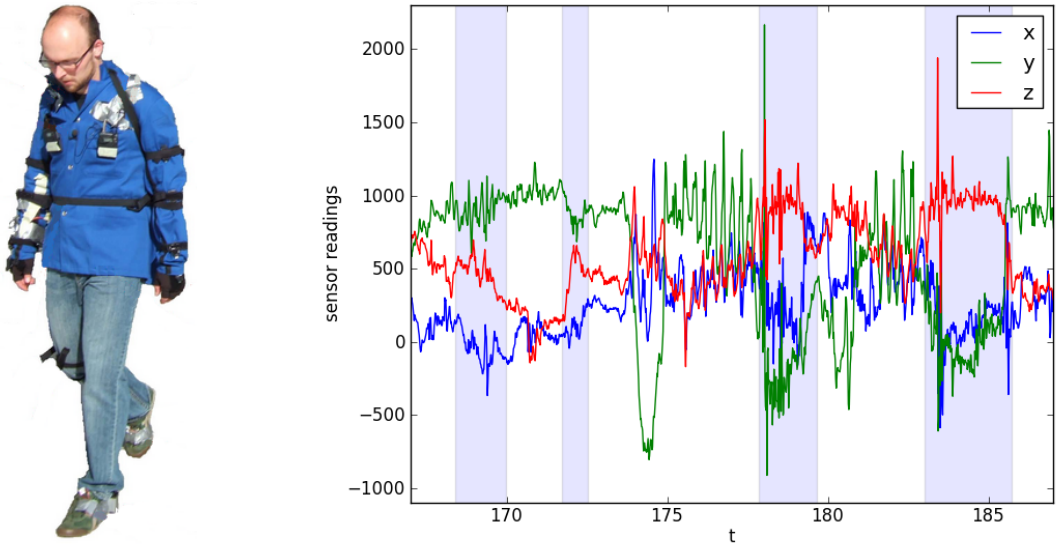


(a) *Quelle: [SkodaMC 2015]* (b)

Abb. A.5.: *(a) Person bei der Durchführung von Qualitätskontrollen an einem Fahrzeug. (b) Auszug aus den Beschleunigungswerten eines Sensors an der rechten Hand. Die Ordinate misst die Beschleunigung im Rohmaß der Sensoren. Die Intervalle, in denen eine Aktivität stattgefunden hat, sind blau hinterlegt. Das Label der jeweiligen Aktivität ist über dem blauen Bereich annotiert.*

A.3.4. Datensatz „Opportunity Challenge“

Abbildung A.6 zeigt eine im Rahmen der Aufnahme des Datensatzes „Opportunity Challenge“ (siehe Abschnitt 3.3.3 für weitere Details) mit Beschleunigungssensoren ausgestattete Person. Daneben ist ein Ausschnitt aus den Daten dargestellt.



(a) *Quelle: [Roggen et al., 2010]* (b)

Abb. A.6.: *(a) Person, welche mit Beschleunigungssensoren ausgestattet wurde. (b) Auszug aus den Beschleunigungswerten eines Sensors an der rechten Hand. Die Ordinate misst die Beschleunigung im Rohmaß der Sensoren. Die Intervalle, in denen eine Aktivität stattgefunden hat, sind blau hinterlegt. Das Label der jeweiligen Aktivität ist über dem blauen Bereich annotiert.*

Anhang B

Personenverfolgung mit Tiefenkameras

Die folgenden Abschnitte beschreiben die Tiefenwahrnehmung und das Tracking von Personen mit Hilfe einer Tiefenkamera und zweier Softwarebibliotheken und dienen als weitergehende Erläuterungen zu Abschnitt 3.2.1.

B.1. Tiefenwahrnehmung mittels OpenNI und NiTE

Für die Aufnahme von Tiefenbildern und das Tracken der Gliedmaßen von Personen wurde die Tiefenkamera Microsoft Kinect in Verbindung mit Fremdsoftware in Form der freien Software-Bibliothek OpenNI [OpenNI 2015] und der geschlossenen Bibliothek NiTE [NiTE 2015] verwendet.

Tiefenkameras haben bezüglich der Wahrnehmung von Bewegungen einige Vorteile gegenüber Farbkameras. Sie funktionieren unabhängig von der Helligkeit, Farbe und Textur, geben durch die Entfernungswerte Aufschluss über die Größe von Objekten und Personen und vereinfachen damit z. B. fensterbasierte Suchverfahren und die Vorder-/Hintergrundtrennung signifikant. Im Folgenden soll das Prinzip der Tiefenwahrnehmung kurz umrissen werden.

Derzeit haben sich drei Verfahren für die visuelle Tiefenwahrnehmung etabliert: Die Tiefenwahrnehmung nach dem Time-of-flight-Verfahren, Stereokameras und die Tiefenwahrnehmung mittels strukturiertem Licht. Während aufgrund geringerer Kosten Stereokameras lange das Feld anführten, ist mit der Einführung der Microsoft XBox 360 Kinect ein sehr preisgünstiger Sensor, der mittels strukturiertem Licht arbeitet, eingeführt worden und hat die Stereokamera in Szenarien wie dem hier Angestrebten quasi abgelöst. Die zweite Generation der Kinect, die Microsoft mit der XBox One vertreibt, arbeitet sogar nach dem Time-of-Flight-Prinzip und ermöglicht damit eine wesentlich höhere Datenqualität. Allerdings war diese erst nach Abschluss der im Rahmen dieser Arbeit durchgeführten Untersuchungen verfügbar und so wurde hier die erste Generation der Kinect verwendet.

Eine Tiefenkamera, die mittels strukturiertem Licht arbeitet, besteht aus einem Projektor, einem Detektor und einem Microcontroller. Die Anordnung ist in Abbildung B.1 dargestellt. Der Projektor projiziert ein bestimmtes Lichtmuster in den Raum (siehe

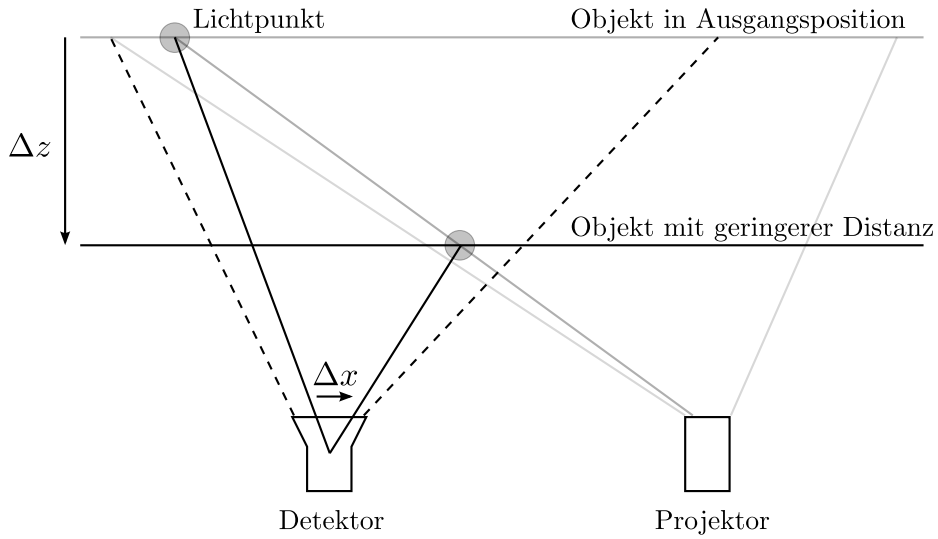
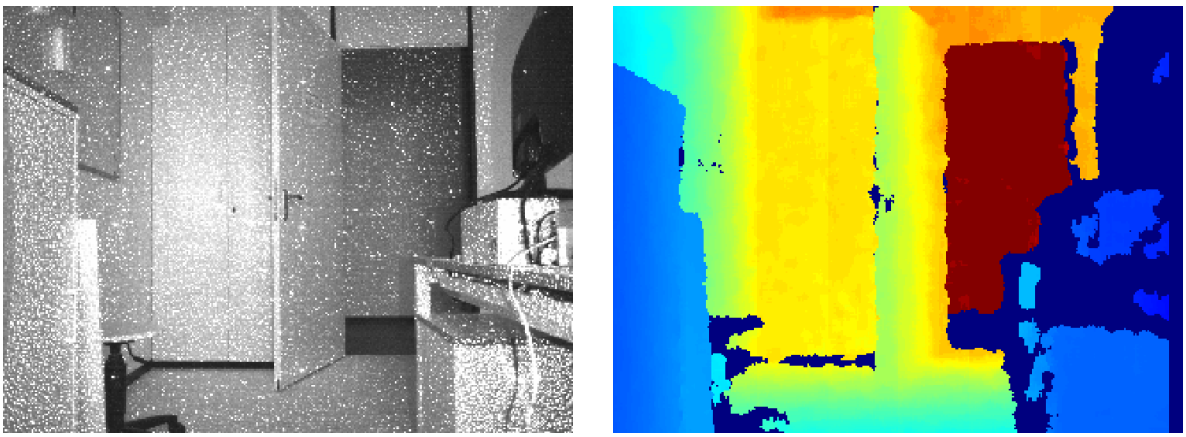


Abb. B.1.: *Prinzip der Tiefenermittlung mittels Triangulation. Die Anordnung besteht aus einem Projektor und einem Detektor mit einem festen Basisabstand. Ein Lichtpunkt wird auf ein Objekt projiziert. Bei Verringerung der Distanz zum Objekt um Δz ist die Projektion des Lichtpunktes im Detektor um Δx verschoben, wodurch sich auf die Entfernung zum Objekt schließen lässt.*

Abbildung B.2a). Der Detektor besteht aus einem CMOS-Sensor, der parallel zum Projektor ausgerichtet ist und einen Videostrom des durch die Szene reflektierten Lichtmusters liefert, den wiederum der Microcontroller verrechnet. Das projizierte Lichtmuster besteht aus einer zufälligen, aber kohärenten und dichten Anordnung von Lichtpunkten. Betrachtet man einen beliebig großen Ausschnitt der Projektion des Lichtmusters, so ist dieser im ganzen Lichtmuster eindeutig. In Abbildung B.1 ist die Projektion eines Lichtpunktes stellvertretend für das ganze Lichtmuster dargestellt. Dieser Lichtpunkt wird vom Detektor an einer bestimmten Position auf dem CMOS-Sensor wahrgenommen. Verschiebt sich nun die Projektionsfläche (ein Objekt oder eine Person) um eine bestimmte Strecke Δz entlang der Sichtachse der Anordnung, so verschiebt sich die Projektion des Punktes auf dem CMOS-Sensor um Δx. Die Verschiebung Δx steht dabei in linearem Zusammenhang zu Δz und kann mit Hilfe des Strahlensatzes bestimmt werden. Dazu werden weitere Größen, wie der Abstand zwischen Projektor und Detektor, der sogenannte Basisabstand, benötigt, die durch die Geometrie der Anordnung vorgegeben und fest sind. Eine genaue Beschreibung der geometrischen Zusammenhänge findet sich in [Khoshelham et al., 2012].

Die Verschiebung Δx wird für jeden Bildpunkt durch eine Suche der Umgebung des Bildpunktes in einem Referenzmuster ermittelt. Das Referenzmuster ist das projizierte Lichtmuster, welches zuvor bei Projektion auf eine Fläche mit bekannter Entfernung aufgezeichnet und gespeichert wurde. Diese Suche wird mittels Korrelation der Umgebung des Bildpunktes entlang der Basislinie, also entlang der x-Achse in Richtung des



(a) *Projiziertes Lichtmuster* (b) *Resultierendes Tiefenbild*

Abb. B.2.: *Das durch den Projektor der Tiefenkamera projizierte IR-Lichtmuster als Grauwertbild (a) und das durch die Disparitätsberechnung ermittelte Tiefenbild (b). Nahe Regionen sind blau codiert und ferne Regionen rot.*

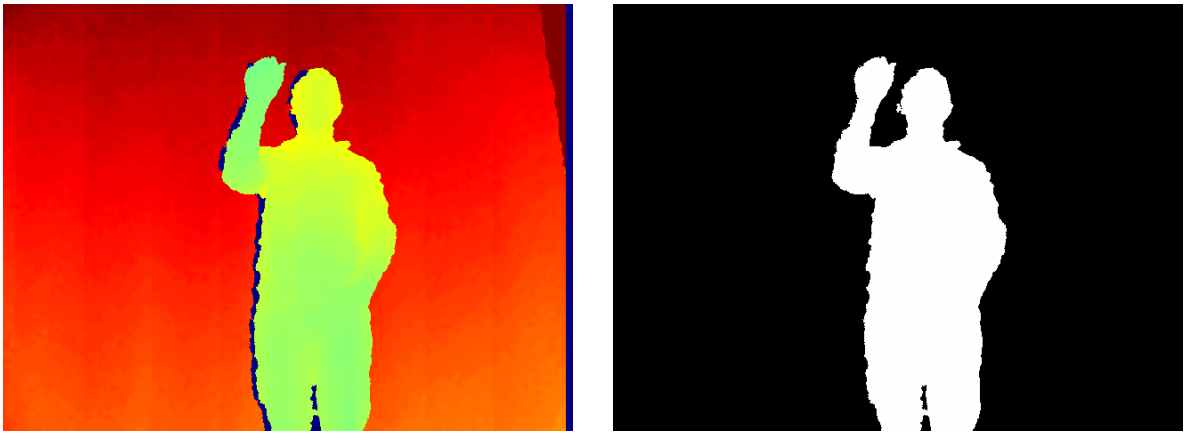
Projektors, durchgeführt. So können die lokalen Verschiebungen (Disparitäten) über das gesamte Bild bestimmt werden und die Entfernungen zur Kamera (Tiefenwerte) berechnet werden.

Abbildung B.2b zeigt ein exemplarisches Tiefenbild in einem Büroraum mit mehreren Objekten (Tisch, Schrank) in unterschiedlicher Tiefe.

B.2. Skelett-Tracking

Seit dem Aufkommen preisgünstiger Tiefenkameras wurde eine Vielzahl von Algorithmen zur Verfolgung der Gliedmaßen und Bewegungen eines Menschen in einem Tiefenbild veröffentlicht. In dieser Arbeit wurde für das Skelett-Tracking die Software-Bibliothek NiTE verwendet. Leider ist die Software geschlossen, und es wurden keine Angaben über die verwendeten Algorithmen veröffentlicht. Es soll daher nur der prinzipielle Ansatz zum Tracking vorgestellt werden, wie er in aktuellen Systemen wie der Microsoft XBox 360 implementiert ist.

Konventionelle Ansätze für das Skelett-Tracking in Tiefenbildern segmentieren zuerst das Tiefenbild in Teile mit Hintergrund und Personen. Dazu wird jedem Pixel eine Person oder der Hintergrund zugeordnet. Abbildung B.3b zeigt eine so erzeugte Segmentierung, die auf Basis des Tiefenbildes in Abb. B.3a entstanden ist. Dabei sind die weißen Pixel der Person zugeordnet, während die schwarzen Pixel zum Hintergrund gehören. In Tiefenbildern kann die Segmentierung mit sehr einfachen Verfahren erfolgen, die meist mit der Annahme arbeiten, dass Objekte im Bild einen stetigen Tiefenverlauf haben. Tritt zwischen zwei Pixeln eine große Tiefendifferenz auf, so wird dies als Objektgrenze interpretiert.



(a) *Tiefenbild* (b) *Segmentierung*

Abb. B.3.: *Die aus dem Tiefenbild (a) erzeugte Segmentierung (b) beschreibt, welche Pixel zur Person (weiß) und welche zum Hintergrund (schwarz) gehören.*

Für das Skelett-Tracking werden im weiteren Verlauf nur die Bereiche des Tiefenbildes betrachtet, die in der Segmentierung der Person zugeordnet wurden. Konventionelle Ansätze versuchen nun meist, ein Modell des menschlichen Körpers in das rückprojizierte, d.h. das aus dem Tiefenbild in Weltkoordinaten zurück berechnete, Personensegment einzupassen. Das verwendete Personenmodell kann im einfachsten Fall aus einer Stick-Figure bestehen. Probleme bereiten dabei die oft hohe Anzahl an Freiheitsgraden der Modelle, die aufwändige Optimierungsverfahren notwendig machen, und einen Echtzeiteinsatz erschweren.

In den aktuellen Systemen von Microsoft ist ein Ansatz implementiert, der in [Shotton et al., 2013] vorgestellt wurde. Das Verfahren geht pixelweise vor und ordnet jedem Pixel im Tiefenbild direkt ein Körperteil zu. Der Klassifikator für die Körperteile wurde mit künstlich erzeugten Tiefendaten trainiert. Dazu wurden digitale Personenmodelle mit echten Motion-Capture-Daten animiert. Zusätzlich wurden bestimmte Parameter der Personenmodelle, wie Größe, Umfang, etc. variiert. So konnte eine umfangreiche Datenbasis geschaffen werden, die eine große Anzahl an real vorkommenden Variationen abdeckt. Durch diesen Ansatz der Datengenerierung war es zudem möglich, die Körperteile automatisch zu labeln. Aus den gerenderten virtuellen Modellen wurden so gelabelte Tiefendaten generiert. Der Klassifikator wird nun auf Basis von Merkmalen trainiert, welche die Umgebung eines Pixels beschreiben. Die Autoren zeigen, dass diese Information ausreicht, um die Zugehörigkeit eines Pixels zu einem Körperteil robust zu klassifizieren. Nachdem jedes Pixel im Tiefenbild auf diese Weise klassifiziert ist, wird nun nach Clustern vom Pixeln mit gleicher Klasse gesucht. Die Cluster-Zentren des rückprojizierten Tiefenbildes stellen die Schwerpunkte der Gliedmaßen dar. Für eine genaue Beschreibung des Verfahrens sei der Leser an [Shotton et al., 2013] verwiesen.

Das Ergebnis der Tracking-Verfahren ist eine Zeitreihe von Posen, bestehend aus

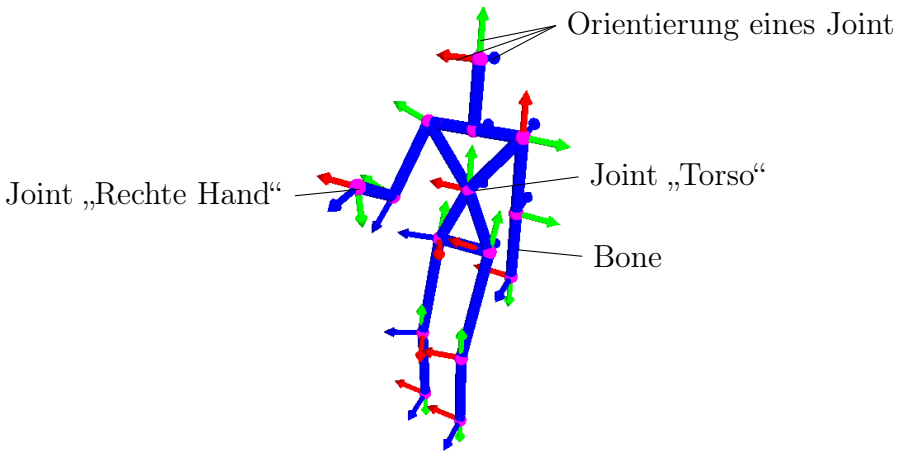


Abb. B.4.: *Durch das Skelett-Tracking verfolgte Person, als Stick-Figure dargestellt. Die Gelenkpositionen sind als Punkte markiert (magenta). Die Verbindungen zwischen zwei Gelenken werden als Bones (blau) bezeichnet. Die Orientierung eines Gelenkes wird durch drei lokale Koordinatenachsen visualisiert. Dabei ist die x-Achse rot, die y-Achse grün und die z-Achse blau markiert.*

Positionen und ggf. Orientierungen von Gelenken. Eine Pose kann mittels einer Stick-Figure veranschaulicht werden. Abbildung B.4 zeigt eine solche Pose als Stick-Figure. Bei Verwendung der Bibliothek NiTE werden 15 Gelenke verfolgt: Der Kopf, der Hals, der linke und rechte Arm mit jeweils drei Gelenken an Schulter, Ellenbogen und Hand, der Torso und das linke und rechte Bein mit jeweils drei Gelenken an Hüfte, Knie und Fuß. Für jedes Gelenk liefert die Bibliothek einen dreidimensionalen Positionsvektor $\mathbf{p}_i \in \mathbb{R}^3$ für $i \in \{1,\ldots,15\}$ und eine Beschreibung der Rotation des Gelenkes in Form einer Quaternion $\mathbf{q}_i \in \mathbb{R}^4$. Eine Pose des Skelettes ist also durch einen Vektor mit $15 \cdot (3+4) = 105$ Elementen $\mathbf{p} \in \mathbb{R}^{105}$ beschrieben. Die Bewegung der Person über die Zeit wird durch eine Trajektorie $\mathcal{T} = ((\mathbf{p}^{(i)},t^{(i)}))_1^T = ((\mathbf{p}^{(1)},t^{(1)}),\ldots,(\mathbf{p}^{(T)},t^{(T)}))$ aus Paaren von Pose und Zeitpunkt beschrieben.

B.3. Visualisierung im Miracenter

Abbildung B.5 zeigt einen Screenshot des Miracenter bei der Ausführung des Demonstrators.

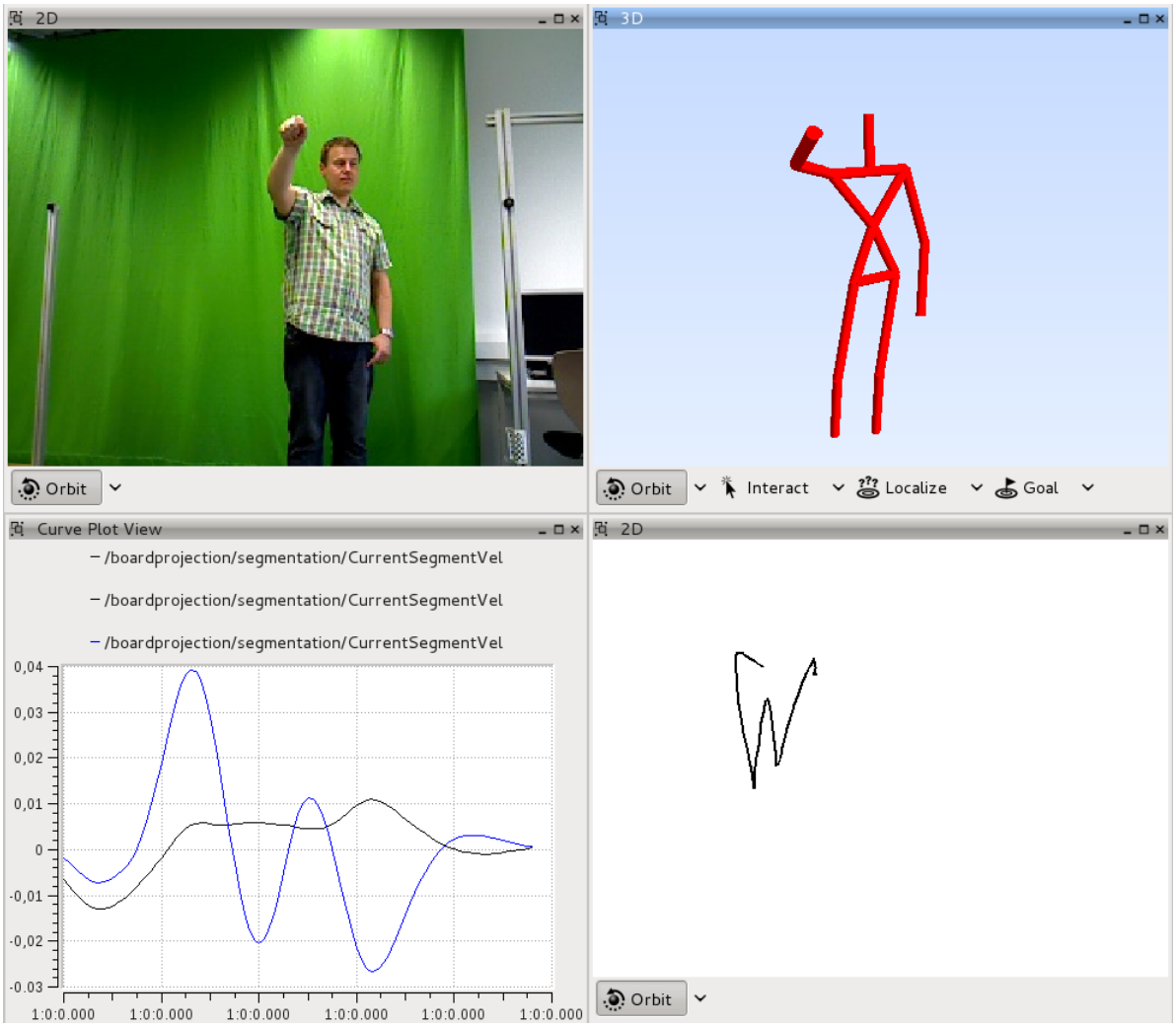


Abb. B.5.: *Screenshot der Visualisierung von Daten im Demonstrator mittels miracenter. Oben links: das von der Kamera produzierte Farbbild. Oben rechts: Die getrackte Person als Stick-Figure. Unten links: Das zuletzt erkannte Segment als Verlauf der Geschwindigkeiten. Unten rechts: Das Segment als Verlauf der Position in der Frontalebene.*

Anhang C

Ergänzungen zum Sparse Coding

Im Folgenden werden einige Ergänzungen zu Kapitel 4 beschrieben.

C.1. Transformationsoperatoren

In Abschnitt 4.2.3 wird die Verschiebungsinvarianz der NMF mit Hilfe abstrakter Transformationsoperatoren definiert. Im Folgenden werden diese Operatoren zur Veranschaulichung konkret beschrieben.

Für den Fall einer einfachen örtlichen bzw. zeitlichen Verschiebung werden die Transformationsoperatoren als Menge von Matrizen definiert, indiziert durch ihre hervorgerufene Verschiebung: $\mathcal{T} = \{\mathbf{T}^{(l)} \in \mathbb{R}^{P' \times P} : l = 1, \ldots, L\}$. Dabei sind z. B. die Transformationsoperatoren für die Verschiebungen $l \in \{0,1,2,\ldots\}$ für Basisvektoren der Länge $P = 3$ und eine Ausgangsdimension $P' = 4$ wie folgt definiert:

$$\mathbf{T}^{(0)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{T}^{(1)} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{T}^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \ldots$$

Transformiert man nun beispielhaft den Vektor $\mathbf{v} = (1,2,3)^T$ und bezeichnet das Ergebnis mit $\mathbf{v}^{(l)}$, dann ergibt sich der verschobene Vektor z. B. zu

$$\mathbf{v}^{(1)} = \mathbf{T}^{(1)}\mathbf{v} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix}$$

Entsprechend sind dann

$$\mathbf{v}^{(0)} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 0 \end{pmatrix}, \mathbf{v}^{(1)} = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 3 \end{pmatrix}, \mathbf{v}^{(2)} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 2 \end{pmatrix}, \ldots$$

C.2. Matching Pursuit

In Abschnitt 4.3.5 wird auf einen alternativen Algorithmus namens Matching Pursuit zur Lösung des Codierungsproblems verwiesen. Dieser Algorithmus wird im Folgenden kurz beschrieben.

Mallat et al [Mallat et al., 1993] beschreiben einen Algorithmus für das Sparse Coding der das Codierungsproblem heuristisch löst. Dabei wird schrittweise jeweils der am besten mit dem Eingangssignal korrelierende Basisvektor gesucht. Sei

$$c_{\mathcal{V},\mathcal{W},\mathcal{T}}(n,m,k) = \sum_d \sum_t V^{(d)}_{p+t,n} \left(\mathbf{T}^{(m)} W^{(d)}_{:,k} \right)_t$$

die Korrelation der m-ten Transformation des k-ten Basisvektors mit dem n-ten Eingangsvektor, dann beschreibt das Tripel

$$(n^*,m^*,k^*) = \arg\max_{n,m,k} c_{\mathcal{V},\mathcal{W},\mathcal{T}}(n,m,k)$$

den Index k^* des am meisten korrelierenden Basisvektors im Eingangsdatum n^* mit der Verschiebung m^*. Weiter beschreibt

$$h^* = c_{\mathcal{V},\mathcal{W},\mathcal{T}}(n^*,m^*,k^*)$$

den Wert dieser Korrelation, der gleich der Aktivierung des Basisvektors ist, wenn man annimmt, dass die Aktivierungen spärlich sind. Das Tupel (n^*,m^*,k^*,h^*) wird in eine Liste $\mathcal{L}$ eingetragen und es wird das Residuum

$$\widetilde{\mathbf{V}}^{(d)} \text{ mit } \widetilde{V}^{(d)}_{:,n^*} = V^{(d)}_{:,n^*} - h^* \cdot \left(\mathbf{T}^{(m^*)} W^{(d)}_{:,k^*} \right)$$

durch Subtraktion des aktivierten Basisvektors gebildet. Danach wird die Prozedur iterativ wiederholt, wobei das Residuum $\widetilde{\mathbf{V}}^{(d)}$ als Eingabevektor für die nächste Iteration dient. Ist die im Residuum enthaltene Signalenergie vernachlässigbar klein, so wird die Iteration abgebrochen. $\mathcal{L}$ repräsentiert dann die spärliche Codierung des Signals. Der Algorithmus des Matching Pursuit in seiner einfachsten Variante ist in Listing 5 dargestellt.

Matching Pursuit ist zwar, wie das in dieser Arbeit entwickelte Verfahren, auch iterativ, dennoch sind alle beteiligten Operationen von geringer Komplexität. Deshalb wird wesentlich weniger Rechenaufwand benötigt. Dies spielt insbesondere beim Einsatz unter Echtzeit-Bedingungen eine übergeordnete Rolle.

Algorithmus 5 Matching Pursuit.

Funktion $\mathcal{L} \leftarrow \mathrm{MP}(\mathcal{V}, \mathcal{W}, \mathcal{T}, B)$
  $\mathcal{L} \leftarrow \emptyset$
  wiederhole
    $(n^*,m^*,k^*) \leftarrow \underset{n,m,k}{\arg\max}\ c_{\mathcal{V},\mathcal{W},\mathcal{T}}(n,m,k)$
    $h^* \leftarrow c_{\mathcal{V},\mathcal{W},\mathcal{T}}(n^*,m^*,k^*)$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(n^*,m^*,k^*,h^*)\}$
    $\mathbf{V}^{(d)} \leftarrow \widetilde{\mathbf{V}}^{(d)}$ mit $\widetilde{V}^{(d)}_{:,n^*} = V^{(d)}_{:,n^*} - h^* \cdot \left(\mathbf{T}^{(m^*)} W^{(d)}_{:,k^*}\right)$
  bis $\|\mathbf{V}^{(d)}\|_2^2 < B$
Ende Funktion

Anhang D

Ergänzungen zu Generierung und Klassifikation

Im Folgenden werden Ergänzungen zur Klassifikation in Kapitel 5 beschrieben.

D.1. Alignment der Aktivierungen ähnlicher Exemplare

In Abschnitt 5.1 wird die Generierung von Trajektorien mit Hilfe durch Sparse Coding gelernter Bewegungsprimitive und deren Aktivierungen beschrieben. Im Lernverfahren ist ein Verfahren zur Angleichung der zeitlichen Verschiebung und Dehnung der klassentypischen Aktivierungsmuster notwendig, welches im folgenden beschrieben wird. Die Ergebnisse der Anwendung des Verfahrens sind in Abschnitt 5.1 dargestellt.

Seien eine Anzahl N von Aktivierungsmustern $x^{(n)} \in \{0,1\}^T$ und ein mittleres Aktivierungsmuster mit

$$y_t = \frac{1}{N} \sum_n x_t^{(n)}$$

gegeben.

Ziel ist es, die Korrelation aller Aktivierungsmuster mit dem mittleren Aktivierungsmuster durch Finden der optimalen Translation a_n und Optimalen Streckung b_n für jedes Aktivierungsmuster x_n zu maximieren.

Ein Gradientenabstieg kann nicht direkt durchgeführt werden, da ein Gradient aus der spärlichen Repräsentation der Daten nicht direkt ermittelt werden kann. Es wird daher eine Kernelfunktion, der Epanechnikov-Kernel, eingeführt, welche differenzierbar sein muss:

$$k(v) = \frac{1}{h} \left(1 - \left(\frac{v}{h}\right)^2\right) \cdot I\left(\left|\frac{v}{h}\right| < 1\right).$$

Dieser Kernel wird später an einem Aufpunkt τ verankert, welcher durch einen Parameter a verschoben und durch b skaliert werden kann, indem eine Transformation g auf

das Argument angewendet wird, welche selbst differenzierbar bzgl. a und b ist.

$$v = g(z;\tau,a,b) = z - (\tau - a)/b,$$

Dadurch ergibt sich

$$k(g(x)) = \frac{1}{h}\left(1 - \left(\frac{x-(\tau-a)/b}{h}\right)^2\right) \cdot I\left(\left|\frac{x-(\tau-a)/b}{h}\right| < 1\right).$$

Der Gradient bzgl. a und b lautet

$$\begin{aligned}
\nabla_a k(g(x;\tau,a,b)) &= \nabla_v k(g(x)) \cdot \nabla_a g(x;\tau,a,b) \\
&\quad -\frac{2}{h^3}\left(x-(\tau-a)/b\right) \cdot \frac{1}{b} \cdot I\left(\left|\frac{x-(\tau-a)/b}{h}\right| < 1\right) \\
\nabla_b k(g(x;\tau,a,b)) &= \nabla_v k(g(x)) \cdot \nabla_b g(x;\tau,a,b) \cdot I\left(\left|\frac{x-(\tau-a)/b}{h}\right| < 1\right) \\
&\quad -\frac{2}{h^3}\left(x-(\tau-a)/b\right) \cdot \frac{\tau-a}{b^2} \cdot I\left(\left|\frac{x-(\tau-a)/b}{h}\right| < 1\right) .
\end{aligned}$$

Wird dieser Kernel nun mit dem Aktivierungsmuster gefaltet und werden die Parameter a und b des Kernels variiert, so wirkt das, als würde man das Aktivierungsmuster selbst verschieben und strecken und das Ergebnis (das interpolierte Aktivierungsmuster) wäre stetig differenzierbar. Im Folgenden wird $\hat{x}^{(n)}(t;a,b)$ die Kernel-Transformation von $x^{(n)}$ genannt:

$$\hat{x}^{(n)}(t;a,b) = \sum_{\tau} x_{\tau}^{(n)} \cdot k(g(x;\tau,a,b)) \ .$$

Die Korrelation zwischen den einzelnen Aktivierungsmustern und dem mittleren Aktivierungsmuster kann nun wie folgt geschrieben werden

$$F(a,b) = \sum_{n,t} \hat{x}^{(n)}(t;a_n,b_n) \cdot y_t \ .$$

Äquivalent dazu kann die Kernel-Transformation auch auf das mittlere Aktivierungsmuster angewendet werden:

$$F(a,b) = \sum_{n,t} x_t^{(n)} \hat{y}(t;a_n,b_n) \ .$$

Nun kann F bzgl. a_n und b_n differenziert werden

$$\begin{aligned}\nabla_{a_n} F(a,b) &= \sum_t x_t^{(n)} \cdot \nabla_{a_n} \hat{y}(t; a_n, b_n) \\ &= \sum_t x_t^{(n)} \sum_\tau y_\tau \cdot \nabla_{a_n} k(g(x; \tau, a_n, b_n)),\end{aligned}$$

(ähnlich für b_n) wodurch ein Gradientenabstieg möglich ist. Der Term

$$\sum_t y_\tau \cdot \nabla_{a_n} k(g(x; \tau, a, b))$$

ist der Gradient des mittleren Aktivierungsmusters an der Stelle t, welcher an der Stelle $x_t^{(n)} \sum_\tau y_\tau \cdot k(g(x; \tau, a, b))$ ausgewertet wird.

In der praktischen Anwendung wird für den Gradientenaufstieg der Algorithmus RProp [Riedmiller et al., 1993] verwendet. RProp wird eigentlich für den Gradientenabstieg beim Lernen von Neuronalen Netzen verwendet. RProp adaptiert die Parameter durch einen Gradientenabstieg mit einer für jeden Parameter individuellen Schrittweite. Des Weiteren ist der Abstieg unabhängig vom Betrag des Gradienten, nur die Richtung wird durch den Gradienten beeinflusst. Dadurch ist der Algorithmus für das hier vorgestellte Problem besonders geeignet, da der Betrag des Gradienten durch die Spärlichkeitsparameter beeinflusst wird und dadurch die Wahl der Schrittweite beim normalen Gradientenabstieg erschwert.

D.2. Activity-String-Matching

Im Folgenden wird mit dem Activity String Matching (ASM) ein alternatives Verfahren zur Klassifikation spärlich codierter Trajektorien beschrieben, welches auf der Abstraktion durch Symbolfolgen beruht. Das Verfahren konnte im Rahmen dieser Arbeit keine signifikant besseren Ergebnisse als einfachere Modelle, welche auf Featurebeschreibungen nach dem Bag-of-Words-Modell beruhen (siehe Abschnitt 5.2.2), liefern, kann aber als Ausgangspunkt für Weiterentwicklungen dienen.

Das Activity String Matching nutzt die Eigenschaft der Codierung durch Sparse Coding, die eigentlich kontinuierliche Zeitreihe durch diskrete Ereignisse darstellen zu können (siehe Abschnitt 4.6). Die Darstellung durch diskrete Ereignisse kann als Sequenz von Symbolen aufgefasst werden, bei der die Symbole die durch die Ereignisse aktivierten Primitive sind. Das Konzept der Transformation ist in Abb. D.1 noch einmal dargestellt. Dabei wird eine Trajektorie über das Sparse Coding und die letztendliche Abstraktion zu Symbolen als Symbolkette der Form (A,B,C,A,B) dargestellt. Durch diese Transformation der Daten sind Algorithmen aus dem Bereich des Sequence Alignment auf das Problem des Matchings zweier Aktivierungsmuster anwendbar. Sequence Alignment dient dem Vergleich von Symbolketten und stammt aus dem Bereich der Textverarbei-

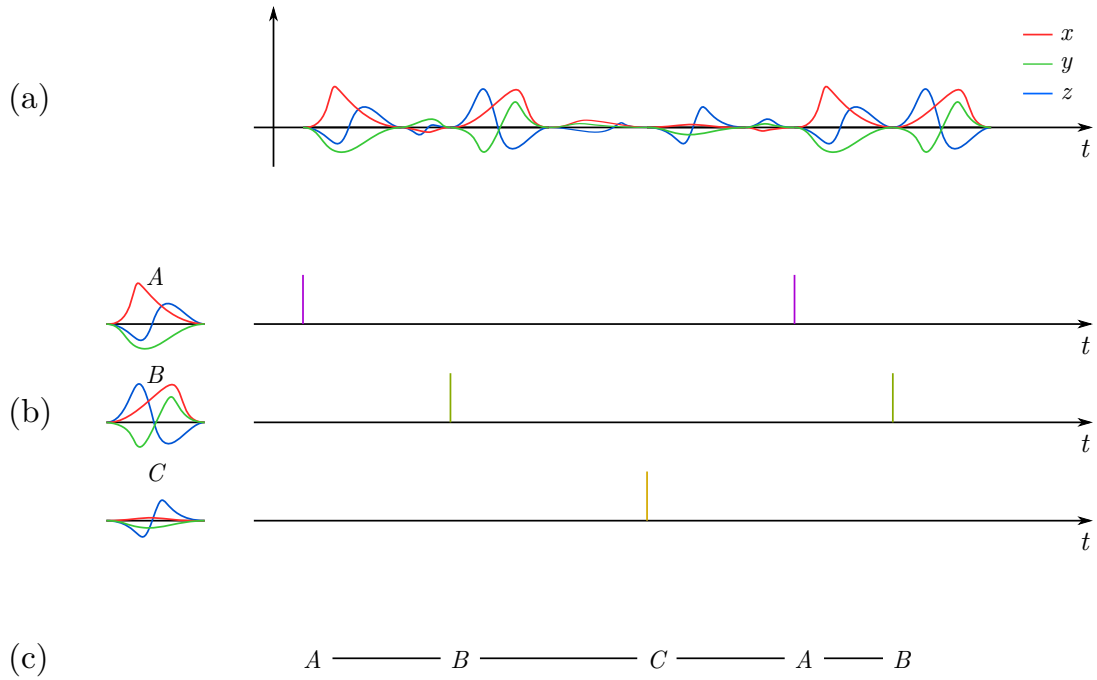


Abb. D.1.: *Spärliche Repräsentation einer Trajektorie (a). Abstrahiert man von der konkreten Form der Primitive und speichert nur die Information über das zeitliche Auftreten, so erhält man eine sehr kompakte Repräsentation (b). Man kann noch einen Schritt weiter gehen und auch die konkreten Zeiten ignorieren und nur die Reihenfolge des Auftretens der Primitive speichern (c) und so zu einer symbolischen Repräsentation gelangen.*

tung. Mittels Sequence Alignment kann die Ähnlichkeit zwischen zwei Symbolketten ermittelt und so ein Ähnlichkeitsmaß zwischen Symbolketten definiert werden.

Mit Hilfe dieses Ähnlichkeitsmaßes ist es möglich, z. B. eine prototypenbasierte Klassifikation durchzuführen. Bei dieser Art der Klassifikation wird eine Datenbank an prototypischen Symbolketten mit bekannter Klasseninformation vorgehalten. Zur Klassifizierung einer neuen, unbekannten Symbolkette, wird diese mit allen Prototypen unter Verwendung des Ähnlichkeitsmaßes verglichen und so der ähnlichste Prototyp ermittelt und seine Klasseninformation als Klassifikationsergebnis verwendet.

Sei also ein Ähnlichkeitsmaß $D : (S) \times (S) \to \mathbb{R}$ zwischen zwei Symbolfolgen gegeben, sodass $D(S_1, S_2) = d$ ein Maß für die Ähnlichkeit zwischen S_1 und S_2 ausdrückt. Sei weiterhin eine Menge von Prototypen $\mathcal{P} = \{P_1, \ldots, P_C\}$ für die Klassen $\{1, \ldots, C\}$ gegeben. Dann wird die Klasse einer unbekannten Symbolfolge S als

$$c^* = K(S) = \underset{c}{\operatorname{argmax}}\{D(S, P_c)\}$$

bestimmt.

Wie bereits erwähnt, lässt sich mit Methoden des Sequence Alignment ein Ähnlichkeitsmaß für Symbolfolgen definieren. Aus der Vielzahl an String-Matching-Algorithmen wird in dieser Arbeit der Needleman-Wunsch-Algorithmus [Needleman et al., 1970] verwendet. Dabei handelt es sich um einen Optimierungsalgorithmus aus der Klasse der Dynamic-Programming-Algorithmen [Bellman, 1954]. Der Algorithmus wurde ursprünglich mit dem Ziel des Vergleichs der Aminosäure-Sequenzen zweier Proteine entwickelt, lässt sich aber auf den Vergleich beliebiger Symbolketten generalisieren.

Die prinzipielle Funktionsweise soll kurz an einem einfachen Beispiel beschrieben werden. Gegeben seien zwei Symbolketten $S_1 = (ABCA)$ und $S_2 = (ABAB)$. Zwischen diesen beiden Symbolketten sind eine Vielzahl von Alignments möglich. z. B.:

$$\begin{array}{cccccc} A & B & C & A & B \\ A & B & - & A & A\,. \end{array}$$

Dabei wird jeder Buchstabe der einen Sequenz mit einem Buchstaben der anderen Sequenz oder einem „Leerzeichen“ (-) aligniert. Die Alignierung gleicher Buchstaben wird als Match, ungleicher als Mismatch bezeichnet. Die Alignierung mit einem Leerzeichen wird als Einfügung bzw. Löschung (sog. Indel-Operation oder kurz Indel) bezeichnet.

Diesem speziellen Alignment werden Kosten zugewiesen. Diese Kosten setzen sich aus den paarweisen Kosten zweier miteinander alignierter Symbole zusammen. Diese paarweisen Kosten werden mittels einer sogenannten Scoring-Funktion Sc definiert. Im einfachsten Fall kann man sie so definieren, dass die Kosten für zwei gleiche miteinander alignierte Symbole 1 sind und für unterschiedliche alignierte Symbole -1. Die Gesamtkosten des Alignments ergeben sich aus der Summe der Einzelkosten, also

$$\begin{aligned} D(S_1, S_2) &= Sc(A,A) + Sc(B,B) + Sc(C,-) + Sc(A,A) + Sc(B,A) \\ &= 1 + 1 + (-1) + 1 + (-1) \\ &= 1\,. \end{aligned}$$

Auf diese Weise lassen sich jedem beliebigen Alignment bestimmte Kosten zuordnen. Der Needleman-Wunsch-Algorithmus findet nun mittels Dynamic-Programming unter allen möglichen Alignments eines mit minimalen Kosten. Diese minimalen Kosten sind das Maß für die Ähnlichkeit zwischen S_1 und S_2. Im Folgenden wird die durch den Needleman-Wunsch-Algorithmus definierte Distanz als $N_{Sc}(S_1, S_2)$ bezeichnet, wobei die Abhängigkeit von der Scoring-Funktion Sc explizit als Parameter angeben ist. Dann lässt sich der Klassifikator als

$$c^* = K_{Sc}(S) = \underset{c}{\operatorname{argmax}}\{N_{Sc}(S, P_c)\}$$

ausdrücken.

Das optimale Alignment wird wesentlich von der Scoring-Funktion beeinflusst. Im einfachsten Fall kann man, wie oben beschrieben, pauschal bei Matches eine 1, bei

Mismatches und Indels eine -1 vergeben. Da die hier verwendeten Symbole allerdings eigentlich die gelernten Bewegungsprimitive beschreiben, welche untereinander mehr oder weniger ähnlich sind, kann man die Scoring-Funktion auch so gestalten, dass für Matches zwischen ähnlichen Primitiven ein höherer Score und für weniger ähnliche Primitive ein geringerer Score vergeben wird. Dies lässt sich in Form einer Scoring-Tabelle beschreiben:

	A	B	C	$-$
A	1	0,7	-1	0
B	0,7	2	$-0{,}5$	0
C	-1	$-0{,}5$	1	0
$-$	0	0	0	0

Die Diagonalelemente beschreiben die Scores der Matches. Diese müssen nicht alle gleich sein. In diesem Beispiel ist der Score für einen Match von B höher als der für einen Match von A oder C. Damit wird ausgedrückt, dass Matches von B wichtiger für das Alignment sind als Matches von A oder C. Die anderen Elemente beschreiben den Score der Mismatches und Indels und drücken aus, wie gut die jeweiligen Symbole „zueinander passen“ oder, anders ausgedrückt, wie sehr Match oder Mismatch „belohnt“ oder „bestraft“ werden. Mit diesen konkreten Werten wird ausgedrückt, dass A und B zueinander ähnlicher sind als z. B. B und C. Die absoluten Werte sind weniger entscheidend, wohl aber die Werte in Relation zueinander, da es am Ende nur auf die Summe der Werte über ein bestimmtes Alignment ankommt.

Im Rahmen dieser Arbeit wurden zwei Ansätze zur Bestimmung der Scoring-Werte untersucht: Zum einen ein Suchverfahren, welches die Werte für Matches, Mismatches und Indels automatisch, mittels Optimierung sucht. Zum zweiten ein Verfahren, welches die Scoring-Werte aus der Ähnlichkeit der den Symbolen entsprechenden Primitive ableitet. Die Verfahren werden im Folgenden näher erläutert.

D.2.1. Suche der Scoring-Werte mittels Evolutionärer Programmierung

Die einfachste Möglichkeit eine Scoring-Tabelle zu definieren, ist, wie oben schon erwähnt, die manuelle Festlegung der Einträge. Dabei ist es sinnvoll, einen relativ hohen Wert für Matches, einen niedrigen oder negativen für Mismatches und einen niedrigen oder neutralen (0) Wert für Indels zu vergeben. Da die Wahl der Werte völlig willkürlich ist, müssen hier viele Werte-Kombinationen ausprobiert werden, um eine gute Score-Funktion zu finden. Die Güte der Score-Funktion kann mittels der durch sie erreichten Klassifikationsgüte definiert werden.

Wenn man die Einträge der Scoring-Tabelle als Vektor in einem Parameterraum interpretiert und die Gütefunktion als Funktion über diesem Raum, so kann die Suche auch als Optimierung aufgefasst werden. Es bietet sich also an, die Suche mittels eines

geeigneten Optimierungsalgorithmus zu automatisieren. Da die Abbildung des Parametervektors auf einen Güte-Wert nicht-linear ist, eignen sich nur nicht-lineare Optimierungsverfahren. Im Rahmen dieser Arbeit wurde dazu ein Optimierungsalgorithmus verwendet, dem das Evolutionary Programming [Fogel et al., 1997] zugrunde liegt. Dabei handelt es sich um ein Optimierungsverfahren, welches durch biologische Evolution inspiriert ist und bei dem neue Lösungen durch „Kreuzung“und „Mutation“ bestehender Lösungen erzeugt werden. Die Fitness von Lösungen wird mit der zu optimierenden Zielfunktion bewertet.

Zur Anwendung des Verfahrens auf das Problem, werden die zu optimierenden Werte als Vektor $\mathbf{r} \in \mathbb{R}^3$ (Match, Mismatch und Indel) interpretiert. Die Menge aller möglichen Scoring-Tabellen ist hier also repräsentiert durch einen Vektorraum $\mathbb{R}^3$.

Zur Bewertung einer Lösung wird direkt die Klassifikationsgüte eines Klassifikators gemessen, der die Scoring-Tabelle als Parameter benötigt. Sei $G_{\mathcal{S}}(K)$ eine Funktion, welche die Klassifikationsgüte eines Klassifikators K auf der Menge von Symbolketten $\mathcal{S}$ misst. Sei weiter $K_{\mathbf{r}} = K_{Sc_{\mathbf{r}}}$ der oben beschriebene prototypbasierte Klassifikator, welcher als Distanzmaß den Needleman-Wunsch-Algorithmus verwendet dessen Parameter die Score-Funktion ist, die durch den betrachteten Lösungsvektor definiert ist. Dann kann die Zielfunktion als

$$F(\mathbf{r}) = G_{\mathcal{S}}(K_{\mathbf{r}})$$

definiert werden.

Zur Optimierung kommt nun, wie schon erwähnt, ein genetischer Algorithmus zum Einsatz, welcher den Parametervektor $\mathbf{r}$ variiert. Der Algorithmus wird dazu mit einer Menge an zufällig gewählten Lösungen initialisiert. Diese Lösungen werden als Individuen bezeichnet. Nun wird die Fitness aller Individuen mit Hilfe der Zielfunktion bestimmt. Aus den besten Individuen wird eine Menge von Paaren ausgewählt, die Mittels eines One-Point-Crossovers gekreuzt werden (es wird genau eine Trennstelle zufällig gewählt, an der die beiden Vektoren getrennt und neu kombiniert werden). Die entstehenden Individuen werden zusätzlich mutiert, indem einzelne Elemente des Vektors mit einem Gaussschen Rauschen beaufschlagt werden.

Die entstehenden Individuen werden bewertet und stellen die Eltern der nächsten Generation dar. Das Verfahren wird für mehrere Generationen wiederholt. Auf diese Weise werden iterativ immer bessere Lösungen erzeugt. Sobald die gemessene Verbesserung der Güte nur noch sehr klein ist, wird die Iteration beendet. Das beste Individuum der letzten Generation stellt dann eine Approximation an die optimale Lösung dar.

D.2.2. Bestimmung der Scoring-Werte über die Ähnlichkeit der Primitive

Da in dieser Arbeit die Symbole eigentlich stellvertretend für Bewegungsprimitive stehen, bietet es sich an, die Ähnlichkeit der Symbole aus der Ähnlichkeit der Bewegungs-

primitive abzuleiten. Damit wird das Problem des Ähnlichkeitsmaßes zwischen Symbolen zurück in den ursprünglichen Raum der kontinuierlichen Trajektorien verlagert. Ein übliches Maß zur Bestimmung der Ähnlichkeit kontinuierlicher Zeitreihen ist die Korrelation. Sei $\mathbf{a} \in \mathbb{R}^L$ die dem Symbol A entsprechende Bewegungsprimitive und $\mathbf{b} \in \mathbb{R}^L$ die dem Symbol B entsprechende Bewegungsprimitive. Die Korrelation zwischen $\mathbf{a}$ und $\mathbf{b}$ ist gegeben durch

$$\kappa = \sum_{1}^{M} a_m b_m \ .$$

Es kann vorkommen, dass die Bewegungsprimitive bis auf eine leichte Verschiebung eigentlich sehr ähnlich sind. Um das Ähnlichkeitsmaß invariant gegenüber dieser Verschiebung zu machen, wird statt der einfachen Korrelation die Kreuzkorrelation

$$(\mathbf{a} \star \mathbf{b})_k = \sum_{m=-\infty}^{\infty} a_m b_{k+m}$$

verwendet, welche die Korrelation bei einer relativen Verschiebung von k beschreibt, und über den Verschiebungsindex k maximiert:

$$Sc(A,B) = \max_k \ (\mathbf{a} \star \mathbf{b})_k \ .$$

Die Primitive wurden während des Lernens normiert (siehe Kap. 4.2.2 und 4.3.2), wodurch ausgeschlossen ist, dass die Korrelation durch unterschiedliche Skalierungen negativ beeinflusst wird. Der vorgestellte Ansatz wird auf mehrkanalige Zeitreihen erweitert, indem die Korrelation über alle Kanäle summiert wird.

Literaturverzeichnis

Asuncion, A. und D. J. Newman (2007). *UCI Machine Learning Repository* (siehe S. 35).

Barlow, H. B. (1985). „The twelfth Bartlett memorial lecture: the role of single neurons in the psychology of perception." In: *The Quarterly journal of experimental psychology. A, Human experimental psychology* 37.2, S. 121–145 (siehe S. 19).

Barlow, H. B. (1990). „Conditions for versatile learning, Helmholtz's unconscious inference, and the task of perception". In: *Vision Research* 30.11, S. 1561–1571 (siehe S. 20).

Barlow, H. B. (2009). „Single units and sensation: A neuron doctrine for perceptual psychology?" In: *Perception* 38.6, S. 371–394 (siehe S. 19).

Barthélemy, Quentin, Anthony Larue und Jérôme I. Mars (2012a). „3D Rotation Invariant Decomposition of Motion Signals". In: *Computer Vision - ECCV 2012. Workshops and Demonstrations*. Hrsg. von Andrea Fusiello, Vittorio Murino und Rita Cucchiara. Bd. 7585. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, S. 172–182 (siehe S. 22).

Barthélemy, Quentin, Anthony Larue und Jérôme I. Mars (2012b). „Quaternionic Sparse Approximation". In: *5th conference on Applied Geometric Algebras in Computer Science and Engineering, La Rochelle : France (2012)*, S. 1–11 (siehe S. 22).

Barthélemy, Quentin, Anthony Larue und Jérôme I. Mars (2014). „Decomposition and dictionary learning for 3D trajectories". In: *Signal Processing* 98, S. 423–437 (siehe S. 22).

Barthélemy, Quentin, Anthony Larue, Aurélien Mayoue, David Mercier und Jérôme I. Mars (2011). „Multivariate Dictionary Learning and Shift & 2D Rotation Invariant Sparse Coding". In: *Statistical Signal Processing Workshop (SSP), 2011 IEEE*, S. 645–648 (siehe S. 22).

Barthélemy, Quentin, Anthony Larue, Aurélien Mayoue, David Mercier und Jérôme I. Mars (2012). „Shift & 2D Rotation Invariant Sparse Coding for Multivariate Signals". In: *IEEE Transactions on Signal Processing* 60.4, S. 1597–1611 (siehe S. 22).

Baum, E. B., J. Moody und F. Wilczek (1988). „Internal representations for associative memory". In: *Biological Cybernetics* 59.4-5, S. 217–228 (siehe S. 18).

Bellman, Richard (1954). „The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60.6, S. 503–516 (siehe S. 131).

Bengio, Yoshua (2009). „Learning Deep Architectures for AI". In: *Foundations and Trends in Machine Learning* 2.1, S. 1–127 (siehe S. 91).

Bizzi, E, S F Giszter, E Loeb, F a Mussa-Ivaldi und P Saltiel (1995). „Modular organization of motor behavior in the frog's spinal cord." In: *Trends in neurosciences* 18.10, S. 442–446 (siehe S. 74).

Blumensath, Thomas und Mike Davies (2006). „Sparse and shift-Invariant representations of music". In: *IEEE Transactions on Audio, Speech and Language Processing* 14.1, S. 50–57 (siehe S. 20).

Bobick, A. F. und A. D. Wilson (1995). „A state-based technique for the summarization and recognition of gesture". In: *Proceedings of IEEE International Conference on Computer Vision*, S. 382–388 (siehe S. 14).

Burges, Christopher J. C. (1998). „A Tutorial on Support Vector Machines for Pattern Recognition". In: *Data Mining and Knowledge Discovery* 2, S. 121–167 (siehe S. 82).

Cooley, James W., Peter A. W. Lewis und Peter D. Welch (1969). „The Fast Fourier Transform and Its Applications". In: *IEEE Transactions on Education* 12.1. Hrsg. von N.J Englewood Cliffs (siehe S. 91).

Cover, T. und P. Hart (1967). „Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13.1 (siehe S. 82, 86).

Csurka, Gabriella, Christopher R. Dance, Lixin Fan, Jutta Willamowski und Cédric Bray (2004). „Visual categorization with bags of keypoints". In: *Proceedings of the ECCV International Workshop on Statistical Learning in Computer Vision*, S. 59–74 (siehe S. 83).

D'Avella, Andrea und Emilio Bizzi (2005). „Shared and specific muscle synergies in natural motor behaviors." In: *Proceedings of the National Academy of Sciences of the United States of America* 102.8, S. 3076–3081 (siehe S. 74).

D'Avella, Andrea, Philippe Saltiel und Emilio Bizzi (2003). „Combinations of muscle synergies in the construction of a natural motor behavior". In: *Nature neuroscience* 6.3, S. 300–308 (siehe S. 74).

Edwards, G. J., C. J. Taylor und T. F. Cootes (1998). „Interpreting face images using Active Appearance Models". In: *Second International Conference on Automatic Face and Gesture Recognition (1998, Nara, Japan)*, S. 300–305 (siehe S. 14).

Einhorn, Erik, Tim Langner, Ronny Stricker, Christian Martin und Horst Michael Gross (2012). „MIRA - Middleware for robotic applications". In: *IEEE International Conference on Intelligent Robots and Systems*, S. 2591–2598 (siehe S. 7, 34).

Evans, David J. (1985). *Sparsity and its applications*. Cambridge University Press (siehe S. 21).

Field, David J. (1987). „Relations between the statistics of natural images and the response properties of cortical cells." In: *Journal of the Optical Society of America A, Optics and image science* 4.12, S. 2379–2394 (siehe S. 21).

Field, David J. (1994). „What Is the Goal of Sensory Coding?" In: *Neural Computation* 6.4, S. 559–601 (siehe S. 18, 20, 21).

Fogel, Lawrence Jerome, Alvin J. Owens und Michael John Walsh (1997). *Artificial Intelligence through Simulated Evolution*. 1965. John Wiley & Sons, Ltd, S. 27–38 (siehe S. 133).

Földiak, Peter (2002). „Sparse coding in the primate cortex“. In: *The Handbook of Brain Theory and Neural Networks*, S. 7 (siehe S. 18).

Földiak, Peter und Dominik Endres (2008). „Sparse coding“. In: *Scholarpedia* 3.1, S. 2984 (siehe S. 19).

Foucart, Simon und Holger Rauhut (2013). *A Mathematical Introduction to Compressive Sensing*. 1. Aufl. Birkhäuser Basel, S. 1–39 (siehe S. 22).

Glowinski, Donald, Antonio Camurri, Carlo Chiorri, Barbara Mazzarino und Gualtiero Volpe (2009). „Validation of an Algorithm for Segmentation of Full-Body Movement Sequences by Perception: A Pilot Experiment“. In: *Gesture-Based Human-Computer Interaction and Simulation*. Springer, S. 239–244 (siehe S. 15).

Gross, Horst-Michael, K. Debes, E. Einhorn, S. Mueller, A. Scheidig, Ch. Weinrich, A. Bley und Ch. Martin (2014). „Mobile Robotic Rehabilitation Assistant for walking and orientation training of Stroke Patients: A report on work in progress“. In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, S. 1880–1887 (siehe S. 1, 26).

Gross, Horst-Michael, Steffen Mueller, Christof Schroeter, Michael Volkhardt, Andrea Scheidig, Klaus Debes, Katja Richter und Nicola Doering (2015). „Robot Companion for Domestic Health Assistance: Implementation, Test and Case Study under Everyday Conditions in Private Apartments*“. In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. Hamburg, S. 5992–5999 (siehe S. 1, 26).

Grosse, R., R. Raina, H. Kwong und A. Y. Ng (2007). „Shift-invariant sparse coding for audio classification“. In: *Cortex* 9.3, S. 8 (siehe S. 20, 23, 87, 88).

Hastie, Trevor, Robert Tibshirani und Jerome Friedman (2009). „Linear Methods for Regression“. In: *The Elements of Statistical Learning*. Springer New York. Kap. 3, S. 43–99 (siehe S. 46).

Hellbach, Sven (2009). „Entwicklung von Methoden zur Unterscheidung und Interpretation von Bewegungsmustern in dynamischen Szenen“. Diss. Technische Universität Ilmenau (siehe S. 7, 59, 107).

Hellbach, Sven, Christian Vollmer und Julian P. Eggert (2011). „Learning Motion Primitives using Spatio-Temporal NMF“. In: *Proc. of the DAGM Workshop New Challenges in Neural Computation 2011, Machine Learning Reports 05/11*, S. 5–8 (siehe S. 10, 40).

Henry, Peter, Christian Vollmer, Brian Ferris und Dieter Fox (2010). „Learning to navigate through crowded environments“. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE, S. 981–986 (siehe S. 11).

Hinton, G. E. und R. R. Salakhutdinov (2006). „Reducing the dimensionality of data with neural networks“. In: *Science (New York, N.Y.)* 313.5786, S. 504–507 (siehe S. 91).

Hubel, D. H. und T. N. Wiesel (2009). „Receptive fields of single neurones in the cat's striate cortex“. In: *The Journal of physiology* 587.12, S. 2721–2732 (siehe S. 17).

Hyvärinen, Aapo und Patrik O. Hoyer (2001). „A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images“. In: *Vision Research* 41.18, S. 2413–2423 (siehe S. 18).

Jennings, A. (1991). *Sparse matrices: Numerical aspects with applications for scientists and engineers*. Bd. 31. 1. Chichester: John Wiley & Sons, Ltd, S. 193–193 (siehe S. 21).

Jolliffe, I .T. (2002). *Principal Component Analysis*. Bd. 98. Springer Series in Statistics. New York: Springer-Verlag, S. 487 (siehe S. 91).

Just, Agnès und Sébastien Marcel (2009). „A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition“. In: *Computer Vision and Image Understanding* 113.4, S. 532–543 (siehe S. 14, 15).

Kaaniche, Mohamed Becha (2009). „Human Gesture Recognition“. Diss. Université de Nice - Sophia Antipolis, S. 129 (siehe S. 14).

Kalman, R. E. (1960). „A New Approach to Linear Filtering and Prediction Problems“. In: *Journal of Basic Engineering* 82.1, S. 35 (siehe S. 31, 62).

Khoshelham, Kourosh und Sander Oude Elberink (2012). „Accuracy and resolution of kinect depth data for indoor mapping applications“. In: *Sensors* 12.2, S. 1437–1454 (siehe S. 118).

Kim, J. H. (1999). „An HMM-based threshold model approach for gesture recognition“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.10, S. 961–973 (siehe S. 15).

Kukreja, Sunil L, Johan Loefberg und Martin J Brenner (2005). „A Least Absolute Shrinkage and Selection Operator (Lasso) for Nonlinear System Identification“. In: *14th IFAC Symposium on System Identification*, S. 6 (siehe S. 46).

LaViola, J. (1999). *A survey of hand posture and gesture recognition techniques and technology*. Techn. Ber. Brown University (siehe S. 14).

Le Roux, Jonathan, Alain de Cheveign und Lucas C Parra (2009). „Adaptive Template Matching with Shift-Invariant Semi-NMF“. In: *Advances in Neural Information Processing Systems* 21 (siehe S. 104).

Lee, D. D. und H. S. Seung (1999). „Learning the parts of objects by non-negative matrix factorization“. In: *Nature* 401.6755, S. 788–791 (siehe S. 6, 49, 57).

Mallat, Stephane G. und Zhifeng Zhang (1993). „Matching pursuits with time-frequency dictionaries“. In: *IEEE Transactions on Signal Processing* 41.12, S. 3397–3415 (siehe S. 22, 59, 104, 124).

Mayoue, Aurelien, Q Barthelemy, S. Onis und A. Larue (2012). „Preprocessing for classification of sparse data: Application to trajectory recognition“. In: *2012 IEEE Statistical Signal Processing Workshop (SSP)*. 2. IEEE, S. 37–40 (siehe S. 23).

Mira (2015). *MIRA*. URL: `http://www.mira-project.org` (besucht am 29. 03. 2015) (siehe S. 7).

Mitra, Sushmita und Tinku Acharya (2007). „Gesture recognition: A survey“. In: *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 37.3, S. 311–324 (siehe S. 13, 16).

Needleman, S. B. und C. D. Wunsch (1970). „A general method applicable to the search for similarities in the amino acid sequence of two proteins“. In: *Journal of molecular biology* 48.3, S. 443–453 (siehe S. 131).

NiTE (2015). *NiTE*. URL: `http://www.openni.ru/files/nite/index.html` (besucht am 03. 04. 2015) (siehe S. 28, 117).

Olshausen, B. A. (2002). *Probabilistic Models of the Brain: Perception and Neural Function*. Hrsg. von M.S. Lewicki R.P.N. Rao, B.A. Olshausen. Neural Information Processing series 13. MIT Press. Kap. 13, S. 257–272 (siehe S. 18).

Olshausen, B. A. und D. J. Field (1996). „Emergence of simple-cell receptive field properties by learning a sparse code for natural images“. In: *Nature* 381.6583, S. 607–609 (siehe S. 2, 17, 97).

OpenNI (2015). *OpenNI*. URL: `http://http://structure.io/openni` (besucht am 03. 04. 2015) (siehe S. 28, 117).

Pavlovic, V. I., R. Sharma und T. S Huang (1997). „Visual interpretation of hand gestures for human-computer-interaction: a review“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.7, S. 677–695 (siehe S. 13–15).

Pham, Cuong und Patrick Olivier (2009). „Slice&Dice: Recognizing food preparation activities using embedded accelerometers“. In: *Lecture Notes in Computer Science (LNCS)* 5859, S. 34–43 (siehe S. 36, 113).

Plötz, Thomas, Nils Y. Hammerla und Patrick Olivier (2011). „Feature learning for activity recognition in ubiquitous computing“. In: *Proceeding IJCAI'11: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*. Bd. 2, S. 1729–1734 (siehe S. 36, 37, 90, 92).

Quinlan, J. R. (1986). „Induction of decision trees“. In: *Machine Learning* 1.1, S. 81–106 (siehe S. 82).

Rabiner, Lawrence R. (1989). „A tutorial on hidden Markov models and selected applications in speech recognition“. In: *Proceedings of the IEEE*. Bd. 77. IEEE, S. 257–286 (siehe S. 16, 76, 84, 98).

Rabiner, Lawrence R. und S. Levinson (1981). „Isolated and Connected Word Recognition - Theory and Selected Applications“. In: *IEEE Transactions on Communications* 29.5, S. 621–659 (siehe S. 16, 20, 84).

Riedmiller, M. und H. Braun (1993). „A direct adaptive method for faster backpropagation learning: the RPROP algorithm“. In: *IEEE International Conference on Neural Networks* (siehe S. 129).

Roggen, Daniel, Alberto Calatroni, Mirco Rossi, Thomas Holleczek, Kilian Förster, Gerhard Tröster, Paul Lukowicz, David Bannach, Gerald Pirkl, Alois Ferscha, Jakob Doppler, Clemens Holzmann, Marc Kurz, Gerald Holl, Ricardo Chavarriaga, Hesam Sagha, Hamidreza Bayati, Marco Creatura und Jose Del R. Millàn (2010). „Collecting complex activity datasets in highly rich networked sensor environments“. In: *INSS 2010 - 7th International Conference on Networked Sensing Systems*, S. 233–240 (siehe S. 37, 116).

Schaal, Stefan (2003). „Dynamic Movement Primitives - A Framework for Motor Control in Humans and Humanoid Robotics“. In: *2nd International Symposium on Adaptive Motion of Animals and Machines (AMAM)* (siehe S. 98).

Scheidig, Andrea, Christof Schröter, Michael Volkhardt, Steffen Müller, Klaus Debes, Horst-Michael Gross, Nicola Döring und Katja Richter (2014). „SERROGA: Servicerobotik für die Gesundheitsassistenz im nutzerzentrierten Entwurf“. In: *Proc. of 7th German AAL Conference (AAL 2014)*. Berlin: VDE Verlag (siehe S. 1).

Schröter, Christof, Steffen Müller, Michael Volkhardt, Erik Einhorn, Horst-Michael Gross und Andreas Bley (2014). „CompanionAble - ein robotischer Assistent und Begleiter für Menschen mit leichter kognitiver Beeinträchtigung“. In: *Proc. of 7th German AAL Conference (AAL 2014)*. VDE Verlag, S. 8 (siehe S. 1).

Shotton, Jamie, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman und Andrew Blake (2013). „Real-time human pose recognition in parts from single depth images“. In: *Studies in Computational Intelligence*. Bd. 411. CVPR '11. Washington, DC, USA: IEEE Computer Society, S. 119–135 (siehe S. 120).

SkodaMC (2015). *Skoda Mini Checkpoint Dataset*. URL: `http://www.ife.ee.ethz.ch/research/groups/Dataset` (besucht am 08. 10. 2015) (siehe S. 115).

Smith, Evan und Michael S Lewicki (2005). „Efficient coding of time-relative structure using spikes“. In: *Neural computation* 17.1, S. 19–45 (siehe S. 20).

Starner, T., J. Makhoul, R. Schwartz und G. Chou (1994). „On-line cursive handwriting recognition using speech recognition methods“. In: *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*. Bd. V. IEEE, S. 125–128 (siehe S. 16).

Starner, Thad Eugene und Alex Pentland (1995). „Visual Recognition of American Sign Language Using Hidden Markov Models“. In: *Media*, S. 189–194 (siehe S. 14, 16).

Stikic, Maja, Tam Huynh, Kristof Van Laerhoven und Bernt Schiele (2008). „ADL recognition based on the combination of RFID and accelerometer sensing“. In: *2008 Second International Conference on Pervasive Computing Technologies for Healthcare*. IEEE, S. 258–263 (siehe S. 36, 114).

Suarez, Jesus und Robin R. Murphy (2012). „Hand gesture recognition with depth images: A review“. In: *Proceedings IEEE International Workshop on Robot and Human Interactive Communication*, S. 411–417 (siehe S. 13).

Vince, John (2011). *Quaternions for Computer Graphics*. 1. Aufl. London: Springer (siehe S. 29).

Vollmer, Christian, Julian P. Eggert und Horst-Michael Gross (2012a). „Generating Motion Trajectories by Sparse Activation of Learned Motion Primitives“. In: *Artificial Neural Networks and Machine Learning - ICANN 2012*. Bd. 7552. Springer, S. 637–644 (siehe S. 10, 74).

Vollmer, Christian, Julian P. Eggert und Horst-Michael Gross (2012b). „Modeling Human Motion Trajectories by Sparse Activation of Motion Primitives Learned from Unpartitioned Data“. In: *Lecture Notes in Computer Science (LNCS)*. Hrsg. von Birte

Glimm und Antonio Krüger. Bd. 7526. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, S. 168–179 (siehe S. 10, 40, 74).

Vollmer, Christian, Horst-Michael Gross und Julian P. Eggert (2013). „Learning Features for Activity Recognition with Shift-Invariant Sparse Coding“. In: *Lecture Notes in Computer Science (LNCS)*. Bd. 8131. Springer, S. 367–374 (siehe S. 10, 74).

Vollmer, Christian, Sven Hellbach, Julian Eggert und Horst-Michael Gross (2014). „Sparse coding of human motion trajectories with non-negative matrix factorization“. In: *Neurocomputing* 124, S. 22–32 (siehe S. 11, 40).

Vollmer, Christian, Erik Schaffernicht und Horst-Michael Gross (2010). „Exploring Continuous Action Spaces with Diffusion Trees for Reinforcement Learning“. In: *Artificial Neural Networks - ICANN 2010*. Bd. 6353. 2. Springer, S. 190–199 (siehe S. 11).

Weinrich, Christoph, Christian Vollmer und Horst-Michael Gross (2012). „Estimation of human upper body orientation for mobile robotics using an SVM decision tree on monocular images“. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, S. 2147–2152 (siehe S. 11).

Williams, Benjamin, Marc Toussaint und Amos Storkey (2006). „Extracting motion primitives from natural handwriting data“. In: *Artificial Neural Networks- ICANN 2006*, S. 634–643 (siehe S. 35).

Yamato, Junji, Jun Ohya und Kenichiro Ishii (1992). „Recognizing human action in time-sequential images using hidden Markov model“. In: *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Comput. Soc. Press, S. 379–385 (siehe S. 14, 16).

Zappi, Piero, Clemens Lombriser, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini und Gerhard Tröster (2008). „Activity Recognition from On-Body Sensors: Accuracy-Power Trade-Off by Dynamic Sensor Selection“. In: *Wireless Sensor Networks*. Bd. 4913. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 17–33 (siehe S. 37).

Zhang, Yinlong, Wei Liang, Jindong Tan, Yang Li und Ziming Zeng (2013). „PCA & HMM Based Arm Gesture Recognition Using Inertial Measurement Unit“. In: *Proceedings of the 8th International Conference on Body Area Networks* 1, S. 7–10 (siehe S. 13).

Zhao, Weizhong, Huifang Ma und Ning Li (2011). „A new non-negative matrix factorization algorithm with sparseness constraints“. In: *Proceedings - International Conference on Machine Learning and Cybernetics* 4, S. 1449–1452 (siehe S. 20).

Abbildungsverzeichnis

Tabellenverzeichnis