RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN
INSTITUT FÜR INFORMATIK III

Katharina Stollenwerk

Bewegungsklassifikation mittels stochastischer Modelle

2. Mai 2008

Diplomarbeit

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der Mensch wird täglich mit einer Fülle von Bewegungsdaten konfrontiert. Es scheint ihm ein Leichtes zwischen verschiedenen Bewegungen zu unterscheiden. Er nutzt dazu seine Augen, mitunter auch seine Ohren, sowie seine Erfahrung, aufgrund derer er das Gesehene und Gehörte einzuschätzen gelernt hat. Er erkennt, ob eine Person auf einem Bein hüpft oder einen frontalen Tritt ausführt, ob sie geht oder rennt, ja sogar ob sie etwas aus einem Regal nimmt oder etwas hineinlegt.

Überlegt man sich, woran er solche Zuordnungen festmacht, so stellt man problemlos fest, dass etwa auf einem Bein hüpfen und ein frontaler Tritt, vom Bewegungsablauf her allenfalls am Anfang Ähnlichkeiten aufweisen. Beide Bewegungen werden zu Beginn ein Bein aus der Hüfte heraus heben. Der Hüpfende wird sein eines Bein mehr oder weniger in dieser Position halten, um nun mit dem anderen Bein zu hüpfen. Der Tretende jedoch wird sich mit dem bloßen Anheben des Beines nicht begnügen, dafür aber mit dem anderen über die gesamte Zeit der Bewegung fest auf der Erde stehen bleiben.

Das Gehen und das Laufen bzw. Rennen voneinander zu unterscheiden ist für den Menschen ebenfalls sehr einfach. Das liegt weniger daran, dass er die frappierenden Unterschiede zwischen beiden Bewegungsabläufen erkennt, denn die sind gar nicht so auffällig. Vielmehr sieht und erkennt er, dass der Rennende (in seiner Fortbewegung) schneller ist, seine Bewegungen kraftvoller sind.

Das tatsächlich Schwierigste der obigen Beispiele ist die Unterscheidung zwischen dem Hinlegen und dem Aufnehmen eines (kleinen) Gegenstandes. Diese beiden Abläufe sind einander im Groben so ähnlich, dass es genauerer Beobachtung benötigt. Tatsächlich unterscheiden sie sich – abgesehen von dem Gegenstand, der in die Hand genommen oder abgelegt wird – vor allem in der Bewegung der Hand, genauer der Finger. Beim Hinlegen eines Gegenstandes ist die Hand zu Beginn um den Gegenstand geschlossen, und sie öffnet sich, um ihn abzulegen. Beim Aufnehmen ist es genau umgekehrt: Die Hand ist erst geöffnet und schließt sich erst am Ende um den Gegenstand. Je nach Größe des Gegenstandes muss ein Beobachter in unmittelbarer Nähe des Handelnden sein, um diesen Unterschied wahrnehmen zu können. Bei größeren Gegenständen (etwa einer Hantel) ist auch hier ein erfolgreiches Erkennen der Bewegung unproblematisch; entweder der Gegenstand liegt nachher dort oder er befindet sich in der Hand der Person.

Möchte man diese Problemstellung des Zuordnens bzw. des Klassifizieren von Bewegungen auf einen Computer übertragen, so muss man Bewegungen zuerst einmal digitalisieren. Zu diesem Zwecke werden Motion Capture (kurz MoCap) Systeme verwendet. Diese sind in der Lage, Bewegungen eines Darsteller in einem zeitlichen und räumlichen Kontext aufzunehmen, so dass sie von einem Computer weiterverarbeitet werden können, siehe Abschnitt 2.1.

Auf Grundlage dieser Daten kann man nun versuchen, einem Computer beizubringen, Bewegungen zuzuordnen, sie zu klassifizieren. D.h. ähnliche Bewegungen zu erkennen und sie zu gruppieren.

1.1 Aufgabenstellung und Aufbau der Arbeit

Ziel dieser Arbeit ist die Klassifikation von Bewegungsdaten mit Hilfe stochastischer Modelle. Das sind zum einen Gauß'sche Mixturmodelle – kurz GMMs – und zum anderen Hidden Markov Modelle – HMMs. Darüber hinaus analysieren wir die Ergebnisse beider Verfahren und vergleichen beide Vorgehen.

Zu diesem Zwecke sei eine Datenbank $\mathcal{D}$ aus Bewegungen $D \in \mathcal{D}$ gegeben. Verschiedene Realisierungen von Bewegungen desselben Typs können zu einer Bewegungsklasse $\mathcal{C}$ zusammengefasst werden. Auf diese Weise lässt sich jede Bewegung der Datenbank einer Bewegungsklasse zuordnen, und man erhält eine Einteilung von $\mathcal{D}$ in γ disjunkte Bewegungsklassen, $\mathcal{D} = \mathcal{C}_1 \dot{\cup} \ldots \dot{\cup} \mathcal{C}_\gamma$.

Es stellt sich die Frage, wie man eine solche Datenbank nutzen kann, um für eine neue unbekannte Bewegung Q zu erfahren, welche Art von Bewegung sie repräsentiert, d.h. man möchte die Bewegungsklasse finden, zu der Q gehört. Umgekehrt kann man auch überlegen, wie sich aufgrund einer Bewegungsklasse $\mathcal{C}$ Bewegungen in einer Datenbank finden lassen, die der Klasse $\mathcal{C}$ ähneln. In beiden Fällen handelt es sich um eine Klassifizierungsaufgabe. Die in vorliegender Arbeit verwirklichte Lösung dieser Klassifizierungsaufgabe lässt sich in drei Phasen unterteilen: Merkmalsextraktion, Training und Klassifikation.

Kapitel 2 behandelt die Merkmalsextraktion. Sie erstellt eine für das weitere Vorgehen geeignete Datengrundlage. Dazu erläutern wir, was man unter Motion Capture Daten versteht. Anschließend beschreiben wir die eigentliche Merkmalsextraktion und die drei Merkmale Gelenkwinkel, relationale und physikalische Merkmale.

In Kapitel 3 gehen wir auf die Klassifikation von Bewegungsdaten durch GMMs ein. Hier erklären wir zuerst die theoretischen Grundlagen, bevor im Training für jede Bewegungsklasse ein GMM gelernt wird. Dies geschieht auf Grundlage der extrahierten Merkmalsvektoren einer Bewegungsklasse. Im Zuge der Klassifikation soll für eine unbekannte Bewegung entschieden werden, in welche der gelernten Bewegungsklasse sie am besten passt. Es folgen die Präsentation und Auswertung der Klassifikationsergebnisse. Den Abschluss des Kapitels bildet eine Analyse der Ergebnisse. Diese wird anhand einiger exemplarischer Klassen durchgeführt und endet mit einer Bewertung des Klassifikationsverfahrens.

Kapitel 4 erörtert die Klassifikation von Bewegungsdaten mittels HMMs. Es ist analog zum vorangegangenen Kapitel aufgebaut und daher ebenfalls unterteilt in theoretische Grundlagen, Training, Klassifikation, Ergebnisse und Analyse.

Kapitel 2

Merkmalsextraktion

2.1 Motion Capture Daten

Als Motion Capturing (kurz auch Mocap) bezeichnet man die (Video-)Aufnahme von Menschen, Tieren oder Objekten, die sich im realen Raum bewegen sowie die daraus erstellte dreidimensionale Repräsentation dieser Bewegung. Die so entstandenen Daten werden als Motion Capture Daten bezeichnet. Zur Aufnahme von Motion Capture Daten gibt es verschiedene Systeme, z.B. optische, magnetische, mechanische oder Bilderfassungssysteme. Häufig kann man mit den durch ein solches System gewonnenen Rohdaten weiterarbeiten. Mitunter ist es aber auch sinnvoll sie auf eine abstrakte vereinheitlichte Skelettstruktur abzubilden, die unabhängig vom Aufnahmesystem ist.

Hier werden wir Motion Capture Daten durch eine stark vereinfachte Nachbildung des menschlichen Skeletts darstellen, eine sogenannte kinematische Kette. Eine kinematische Kette ist eine hierarchische Struktur aus starren Knochen, die durch Gelenke verschiedener Freiheitsgrade flexibel miteinander verbunden sind (siehe Abb. 2.1). Sei J die Menge der Gelenke. Jedes Gelenk trägt einen seiner Funktion entsprechenden Namen wie etwa 'root', 'lankle' (linkes Fußgelenk, *left ankle*), 'rankle' (rechtes Fußgelenk, *right ankle*), 'lknee' (linkes Knie, *left knee*), usw. Der Einfachheit halber werden auch Endeffektoren wie Finger und Zehen als Gelenke dargestellt. Aus den von einem Motion Capture System aufgenommenen Daten der Bewegungen eines Darstellers lässt sich eine zeitlich gebundene Abfolge von 3D-Gelenkpositionen sowie -winkeln bezogen auf ein festes Skelett ableiten. Wir werden im Folgenden einen Motion Capture Datenstrom als eine Folge von Frames betrachten, wobei jeder Frame die räumliche Lage der Gelenke zu einem bestimmten Zeitpunkt beschreibt. Wenn es um abstraktere Zusammenhänge geht, werden wir auch von einer Pose statt von einem Frame sprechen. Eine Pose ist aus mathematischer Sicht eine Matrix $P \in \mathbb{R}^{3\times|J|}$ wobei $|J|$ die Anzahl der Gelenke bezeichnet. Die j-te Spalte P^j von P entspricht den 3D-Koordinaten des Gelenks $j \in J$.

Ein Motion Capture Datenstrom D (auch Dokument) kann modelliert werden als eine Abbildung

$$D : [1 : T] \longrightarrow \mathcal{P} \subset \mathbb{R}^{3\times|J|}$$

worin $T \in \mathbb{N}$ die Anzahl der Posen, $[1 : T] = \{1, 2, \ldots, T\}$ die äquidistant gesampelte Zeitachse und $\mathcal{P}$ der Posenraum sind. Eine Teilfolge aufeinanderfolgender Frames wird auch Motionclip genannt. Die von einem einzelnen Gelenk über eine Zeitspanne beschriebene Kurve heißt 3D-Trajektorie.

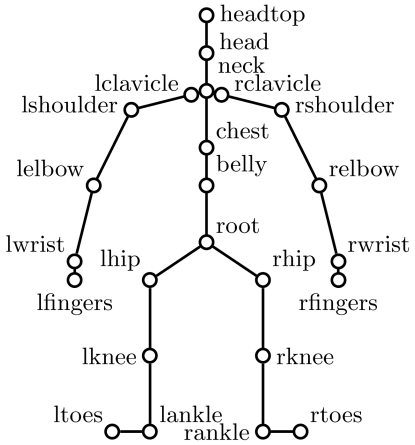


Abbildung 2.1: *Stark vereinfachte Form eines menschlichen Skeletts mit starren Knochen, die über Gelenke miteinander verbunden sind. Gelenke sind in dieser Darstellung mit beschrifteten Kreisen gekennzeichnet.*

Man kann eine Sammlung von Motion Capture Daten strukturieren, indem man sie in Bewegungsklassen einteilt. Eine Bewegungsklasse $\mathcal{C}$ ist eine Menge von Motion Capture Datenströmen, in denen Bewegungen desselben Typs ausgeführt werden. So enthält die in Anhang A vorgestellte Datenbank – das ist gleichzeitig die, auf Grundlage derer wir arbeiten werden – beispielsweise eine Klasse „elbowToKnee1RepsLelbowStart“. Sie enthält 27 Realisierungen einer „elbow-to-knee“ Bewegung, bei der mit dem linken Ellenbogen begonnen wird.

2.2 Merkmalsextraktor

Eine Motion Capture Sequenz besteht also aus einer (zeitlichen) Abfolge von Posen. Diese können ihrerseits charakterisiert werden durch eine Menge von 3D-Koordinaten, die die Positionen von Gelenken in einem Skelett (Abb. 2.1) zu einem festen Zeitpunkt kodieren. Um aus dieser 3D-Punktwolke weitere Informationen gewinnen zu können, wird eine Merkmalsextraktion durchgeführt. Für ein reellwertiges Merkmal kann ein Merkmalsextraktor als eine Funktion $F : \mathcal{P} \longrightarrow \mathbb{R}$ aufgefasst werden. Wurde nicht nur ein reellwertiges Merkmal extrahiert, sondern f viele, also ein Vektor der Länge f, so erhält man die Funktion

$$F : \mathcal{P} \longrightarrow \mathbb{R}^f,$$

die einer Pose $P \in \mathcal{P}$ einen reellwertigen f-dimensionelen Merkmalsvektor $F(P)$ zuweist. Betrachtet man einen Motion Capture Datenstrom $D : [1 : T] \longrightarrow \mathcal{P}$ posenweise, kann ein beliebiger Merkmalsextraktor durch Komposition von F und D, $F \circ D$, darauf angewendet werden. Wir werden in den folgenden Abschnitten drei Merkmalsarten und -mengen definieren, die im weiteren Verlauf Verwendung finden.

2.3 Gelenkwinkel

Zuerst betrachten wir den Gelenkwinkel. Er misst den Winkel zwischen zwei Knochen – definiert durch die 3D-Koordinaten ihrer Gelenke, siehe Abb. 2.2 für ein Beispiel. Des Weiteren zeigt Abb. 2.3 das Ergebnis der Merkmalsextraktion für zwei Gelenkwinkel.

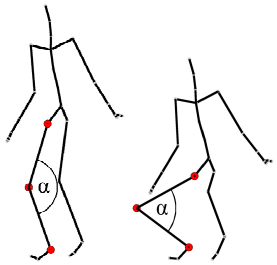


Abbildung 2.2: *Beispiel für den von den zwei Segmenten ('hip','knee') und ('knee','ankle') in der Ebene der drei beteiligten Gelenke eingeschlossenen Winkel α. Er entspricht hier dem Winkel am Knie.*

Ein Vorteil dieses einfachen Merkmals liegt in seiner Unabhängigkeit von Eigenschaften des zugrundeliegenden Skeletts wie etwa der Körpergröße des Darstellers. Jedoch modellieren die Gelenkwinkel eine Bewegung nicht sehr eindeutig, wenn man die Reihenfolge, in der sie auftreten außer Acht lässt. Und genau das wird im Zuge des Trainings und der Klassifikation geschehen, wenn aus den Winkelkonfigurationen ein GMM berechnet wird (siehe Abschnitt 3.2). Ohne Beachtung einer Reihenfolge ist es u.a. schwierig, nur anhand der Winkelkonfigurationen zu unterscheiden, ob jemand einen Arm nach vorne rotiert oder nach hinten, oder ob sich jemand auf einen Stuhl setzt oder von ihm aufsteht.

Ab jetzt definieren, wenn nicht anders ausgezeichnet, $f = 11$ Gelenkkonfigurationen (siehe Tabelle 2.1) einen Merkmalsvektor. Datengrundlage für das Training und die Klassifizierung ist dann die Menge der Posenmerkmalsvektoren über eine gesamte Bewegungssequenz.

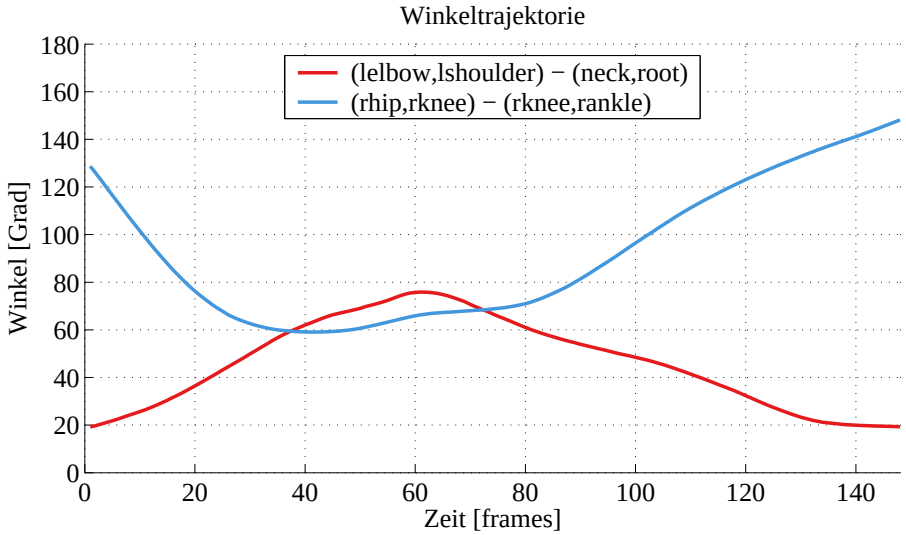


Abbildung 2.3: *Ergebnis der Merkmalsextraktion für den Gelenkwinkel zwischen zwei Knochen einer „elbow-to-knee"-Bewegung. Die abgebildeten Kurven entsprechen dem Winkelverlauf zwischen rechtem Ober- und Unterschenkel (blau) und zwischen linkem Oberarm und Körper (rot).*

Lfnr.	1. Knochen		2. Knochen		Erläuterung
1.	lwrist	lelbow	lelbow	lshoulder	linker Ellenbogen
2.	lhip	lknee	lknee	lankle	linkes Knie
3.	lknee	lhip	root	neck	linke Hüfte
4.	lelbow	lshoulder	neck	root	linke Schulter (bzgl. Körpersenkrechter)
5.	lelbow	lshoulder	lshoulder	neck	linke Schulter (bzgl. Körperhorizontaler)
6.	rwrist	relbow	relbow	rshoulder	rechter Ellenbogen
7.	rhip	rknee	rknee	rankle	rechtes Knie
8.	rknee	rhip	root	neck	rechte Hüfte
9.	relbow	rshoulder	neck	root	rechte Schulter (bzgl. Körpersenkrechter)
10.	relbow	rshoulder	rshoulder	neck	rechte Schulter (bzgl. Körperhorizontaler)
11.	headtop	head	neck	chest	Hals

Tabelle 2.1: *Auswahl der für die Merkmalsvektoren ausgewählten Gelenkkonfigurationen.*

2.4 Relationale Merkmale

Eine in [9] und [7] zur inhaltsbasierten Analyse und Klassifikation [8] von Bewegungsdaten verwendete Merkmalsmenge besteht aus sogenannten *relationalen Features*. Das sind Merkmale, die geometrische Relationen zwischen Gelenken oder Punkten einer Pose beschreiben. Ursprünglich als boolescher Featureextraktor $F : \mathcal{P} \longrightarrow \{0, 1\}$ eingesetzt, wurde er hier für die Berechnung reellwertiger Merkmale $F_r : \mathcal{P} \longrightarrow \mathbb{R}$ genutzt. Während F also beispielsweise eine Antwort auf die Frage „wird der linke Fuß mit einer gewissen Geschwindigkeit bewegt?“ gibt, beschreibt F_r mit welcher Geschwindigkeit der linke Fuß bewegt wird. Dabei geht F_r aus F durch Weglassen des abschließenden Schwellwertvergleichs hervor.

Die Idee dieser einfachen, aber dennoch mächtigen Features lässt sich wohl am besten durch ein Beispiel beschreiben: Nehmen wir also an, wir möchten ein Merkmal berechnen, das – in seiner booleschen Variante – messen kann, ob der linke Fuß vor dem rechten ist. Dazu legt man zuerst eine Ebene durch den Mittelpunkt der beiden Hüftgelenke (‘root’), das (linke) Hüftgelenk (‘lhip’)

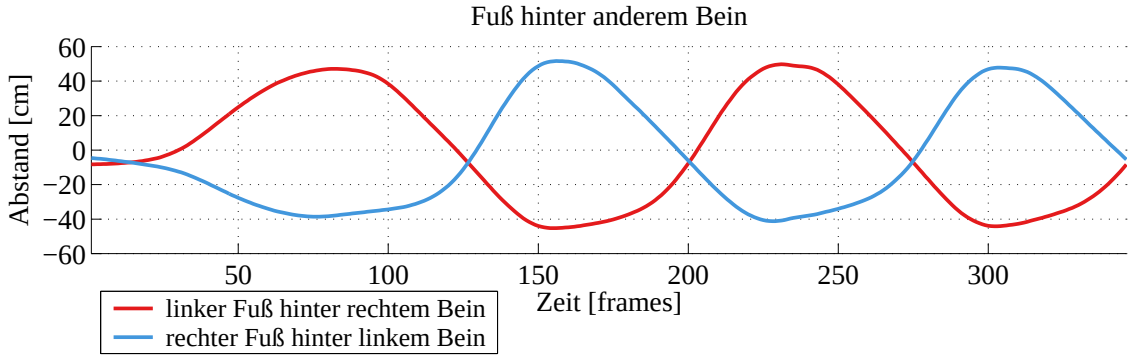
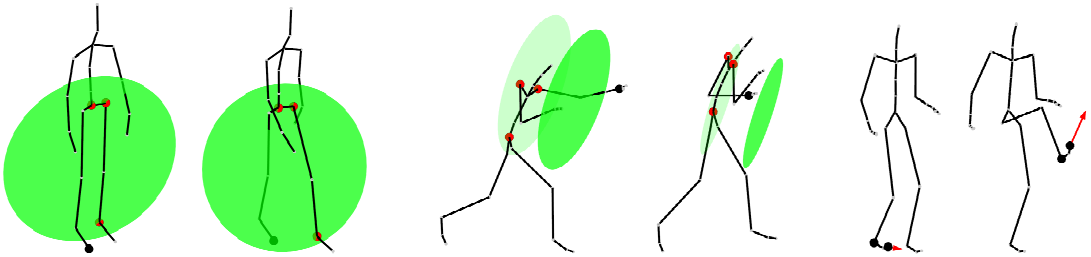


Abbildung 2.4: *Ergebnis der Merkmalsextraktion für die relationalen Features zweier Konfigurationen einer aus vier Schritten bestehenden Gehbewegung. Die abgebildeten Kurven entsprechen dem Abstand (blau) des rechten Fußes zu der durch ‘root’, ‘lhip’ und ‘lfoot’ definierten Ebene und des linken Fußes (rot) zu der durch ‘root’, ‘rhip’ und ‘rfoot’ definierten Ebene.*

und den (linken) Fuß (‘lfoot’). Mit den Koordinaten des anderen (rechten) Fußes lässt sich nun testen, ob er vor oder hinter dieser Ebene ist. Das reellwertige Analogon dieses Feature entspricht dem vorzeichenbehafteten Abstand des Fußgelenks von der definierten Ebene (siehe Abb. 2.4 und Abb. 2.5(a)).



(a) rechter Fuß hinter linkem Bein (b) linke Hand ist vorne (c) Geschwindigkeit des rechten Fußes

Abbildung 2.5: *Beispiele für relationale Merkmale. Betroffene Gelenke sind rot markiert, wenn sie zur Definition einer Ebene (grün) dienen. Gegebenenfalls sind diejenigen Gelenke schwarz eingefärbt, die gegen diese Ebene getestet werden. Sonst markieren schwarze Punkte betroffene Gelenke.*

Das in Abb. 2.5(b) visualisierte Merkmal testet, ob die linke Hand nach vorne ausgestreckt ist. Dazu legt man eine Ebene durch Wurzelknoten (‘root’), linke und rechte Schulter (‘lshoulder’ und ‘rshoulder’). Diese Ebene wird um eine Oberarmlänge nach vorne verschoben. Anschließend kann man mit den Koordinaten des linken Handgelenks (‘lwrist’) testen, ob es vor oder hinter der Ebene liegt. Das letzte abgebildete Merkmal (Abb. 2.5(c)) dient der Beschreibung der Fußgeschwindigkeit. Hier wird die Geschwindigkeit der Zehen des rechten Fußes (‘rtoes’) und des rechten Fußgelenks (‘rankle’) berechnet und sodann getestet, ob beide einen festgelegten Schwellwert überschreiten. Man zieht beide Gelenke heran, da dieses Merkmal den gesamten Fuß berücksichtigen soll und nicht nur Teile. Der Autor von [7] erläutert daraufhin anschaulich, warum es einen Unterschied macht, ob man beide Gelenke nutzt oder nur eines anhand einer Gehbewegung: Setzt der Fuß nach dem Schritt auf den Boden auf, ist die Geschwindigkeit ab diesem Zeitpunkt vernachlässigungswürdig klein. Nicht so die der Zehen, denn sie haben den Weg bis zum Boden noch vor sich bis der gesamte Fuß aufliegt und verzeichnen damit durchaus noch größere Geschwindigkeiten. Der umgekehrte Fall tritt ein, wenn der Gehende einen Fuß zum nächsten Schritt hebt. Hier bleiben die Zehen länger unbewegt auf dem Boden als die Ferse (und damit auch als das Fußgelenk). Aus diesem Grund werden zur Beschreibung der Fußgeschwindigkeit sowohl Zehen als auch Fußgelenk genutzt. Neben geschwindigkeits- und abstandsmessenden Merkmalen gibt es weitere relationale Features, die die relative Nähe etwa zweier Gelenke beurteilen.

Relationale Features sind invariant unter verschiedenen globalen Transformationen, z.B. bleibt das in Abb. 2.4 dargestellte Merkmal unverändert unter globaler Translation und Rotation sowie Skalierung des Skeletts. Eine später neben den Gelenkwinkeln verwendete Merkmalsmenge von relationalen Features besteht aus den $f = 12$ in Tabelle 2.2 aufgelisteten.

Lfnr.	Featurename	ID	Erläuterung
1.	handLeftMoveApartRelRoot	F_{12}	linke Hand entfernt sich vom Wurzelknoten
2.	handRightMoveApartRelRoot	F_{11}	rechte Hand entfernt sich vom Wurzelknoten
3.	handLeftToFrontRelTorso	F_2	linke Hand bewegt sich nach vorne
4.	handRightToFrontRelTorso	F_1	rechte Hand bewegt sich nach vorne
5.	handLeftHighVel	F_{14}	Geschwindigkeit von 'lhand'
6.	handRightHighVel	F_{13}	Geschwindigkeit von 'rhand'
7.	footLeftBackRelLegRight	F_{16}	linker Fuß ist hinter dem rechten Bein
8.	footRightBackRelLegLeft	F_{15}	rechter Fuß ist hinter dem linken Bein
9.	footLeftHighVel	F_{26}	Geschwindigkeit von 'lfoot'
10.	footRightHighVel	F_{25}	Geschwindigkeit von 'rfoot'
11.	footLeftRaisedRelYBodyMin	F_{18}	Abstand von 'lfoot' zum Boden
12.	footRightRaisedRelYBodyMin	F_{17}	Abstand von 'rfoot' zum Boden

Tabelle 2.2: *Verwendete Auswahl der relationalen Merkmale.*

2.5 Physikalische Merkmale

Eine weitere Merkmalsmenge besteht aus physikalisch basierten Merkmalen wie sie in [4, 3] Verwendung fanden, um Bewegungen im Rahmen eines Dynamic-Time-Warping Algorithmus zu vergleichen. Als Erweiterung der dort verwendeten Merkmale wurden hier die kinetischen Energien der einzelnen Körpersegmente (Knochen) berechnet. Die kinetische Energie ist eine (in diesem Fall frameweise) skalare Größe, die mit der Bewegung eines (Starr)Körpers assoziiert wird und hängt ab von Masse und Geschwindigkeit seines Massenpunktes. Sie ergibt sich aus

$$\mathrm{E}_{\mathrm{kin}} = \frac{1}{2} M v^2 + \frac{1}{2} J \omega$$

Hier gehen neben der Translationsgeschwindigkeit v des Massenpunktes und seiner Winkelgeschwindigkeit ω, die Masse M des Starrkörpers sowie das Trägheitsmoment J des Körpers bzgl. seines Massenpunktes ein.

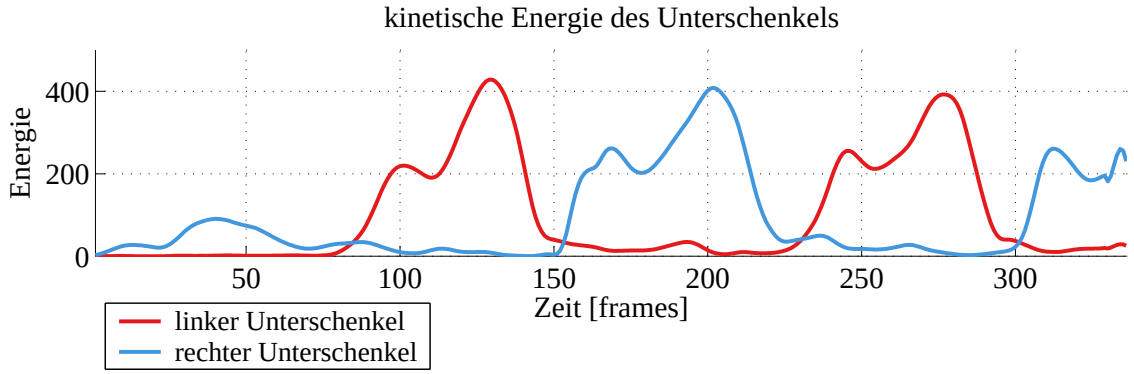


Abbildung 2.6: *Ergebnis der Merkmalsextraktion für die physikalischen Features zweier Merkmale einer aus vier Schritten bestehenden Gehbewegung. Die abgebildeten Kurven entsprechen der kinetischen Energie des rechten (blau) und des linken (rot) Unterschenkels.*

Abb. 2.6 stellt den Verlauf zweier physikalischer Merkmale für eine aus vier Schritten bestehenden Gehbewegung (startend mit dem rechten Fuß) dar. Neben den schon erwähnten Merkmalssätzen von elf Gelenkwinkeln und zwölf relationalen Merkmalen stellten wir noch eine weitere Merkmalsmenge aus $f = 12$ physikalisch basierten Merkmalen zusammen. Tabelle 2.3 enthält die entsprechende Liste.

Lfnr.	Featurename	Erläuterung
1.	lhip_@_lknee	kin. Energie des linken Oberschenkels
2.	rhip_@_rknee	kin. Energie des rechten Oberschenkels
3.	lknee_@_lankle	kin. Energie des linken Unterschenkels
4.	rknee_@_rankle	kin. Energie des rechten Unterschenkels
5.	lfoot_@_ltoes	kin. Energie des linken Fußes
6.	rfoot_@_rtoes	kin. Energie des rechten Fußes
7.	lshoulder_@_lelbow	kin. Energie des linken Oberarms
8.	rshoulder_@_relbow	kin. Energie des linken Oberarms
9.	lelbow_@_lwrist	kin. Energie des linken Unterarms
10.	relbow_@_rwrist	kin. Energie des rechtenen Unterarms
11.	belly_@_chest	kin. Energie des Rumpfes
12.	head_@_headtop	kin. Energie des Kopfes

Tabelle 2.3: *Verwendete Auswahl der physikalischen Merkmale.*

Kapitel 3

Ein GMM-basiertes Klassifikationsverfahren

3.1 GMM - Gauß'sches Mixturmodell

Das Gauß'sche Mixturmodell [13], kurz GMM, ist ein stochastisches Modell, das die Wahrscheinlichkeitsdichtefunktion der zu modellierenden mehrdimensionalen Merkmalsvektoren mittels *einer* gewichteten Überlagerung aus Gauß'schen Normalverteilungen nachbildet. Alle Vektoren werden durch eine einzelne Verteilung wiedergegeben. Die zeitliche Abhängigkeit der Merkmalsvektoren geht dadurch verloren bzw. sie bleibt unberücksichtigt.

Für einen d-dimensionalen Merkmalsvektor x lässt sich die Dichtefunktion eines GMM schreiben als

$$p(x) = \sum_{j=1}^{M} w_j p_j(x). \tag{3.1.1}$$

Sie ist also eine gewichtete Linearkombination von M Basisfunktionen $p_j(x)$, $j = 1, \ldots, M$, die jeweils parametrisiert werden können über einen $d \times 1$ Mittelwertsvektor μ_j sowie eine $d \times d$ Kovarianzmatrix Σ_j:

$$p_j(x) = \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_j)^T(\Sigma_j)^{-1}(x-\mu_j)\right). \tag{3.1.2}$$

Unter Annahme sphärischer Kovarianz, d.h. $\Sigma_j = \sigma_j^2 \mathbb{1}$, vereinfacht sich $p_j(x)$ zu

$$p_j(x) = \frac{1}{(2\pi\sigma_j^2)^{d/2}} \exp\left(-\frac{1}{2}\frac{\|x-\mu_j\|^2}{\sigma_j^2}\right). \tag{3.1.3}$$

3.1.1 Berechnung der Parameter eines GMM

Die Parameter w_j, μ_j und σ_j, die einem GMM zugrunde liegen, werden für eine Sequenz von Merkmalsvektoren mit Hilfe eines k-means Algorithmus berechnet. Es handelt sich dabei um ein Clustering Verfahren [13], das versucht die gesamte Datenmenge in k Bereiche (Cluster) einzuteilen. Jedes Cluster wird durch einen Vektor μ_j, $1 \leq j \leq k$, repräsentiert und jeder Datenpunkt dem ihm

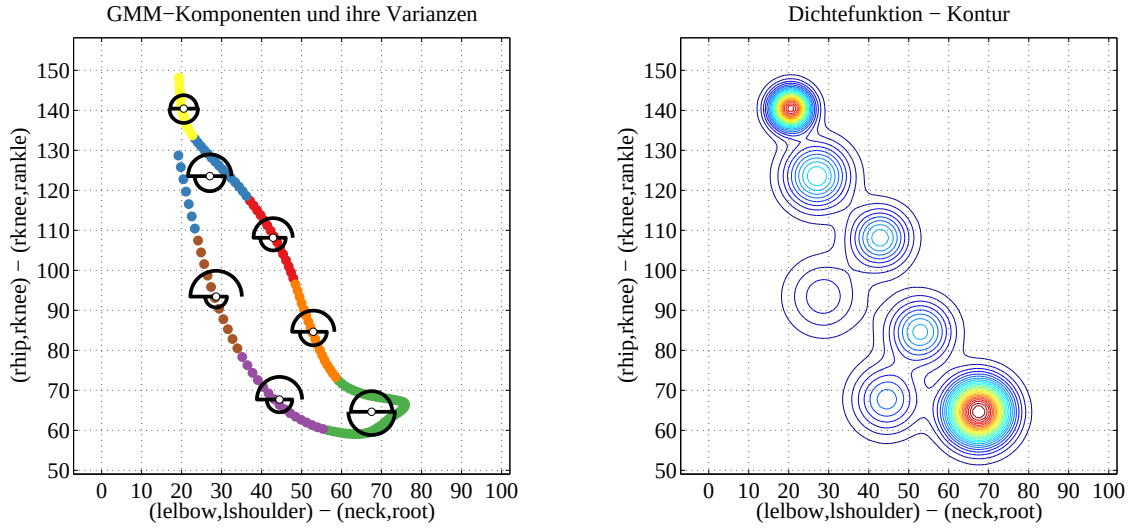
räumlich nächsten Vektor zugeordnet. In unserem Fall liefert der k-means Algorithmus (für $k = M$) die M Komponenten μ_j des GMM in Form von Clustermittelpunkten. Daraus lassen sich Varianz σ_j^2 und Gewicht w_j der einzelnen Komponenten ableiten.

$$\sigma_j^2 = \frac{1}{|X_j|} \sum_{x \in X_j} \|x - \mu_j\|^2 \qquad \text{und} \qquad w_j = \frac{|X_j|}{|X|},$$

wobei X die Menge der Merkmalsvektoren und $X_j \subset X$ diejenigen Merkmalsvektoren, die dem j-ten Cluster zugeordnet sind. Wegen $X_i \cup X_j = \emptyset$ und $\bigcup_{j=1}^{M} X_j = X$ gilt zum einen für die Gewichte $\sum_{j=1}^{M} w_j = 1$, und darüber hinaus ist $0 \leq w_j \leq 1$.

Die Initialisierung des k-means Algorithmus erfolgt zufällig. Daher ist das Ergebnis, wie etwa die Lage der Zentren und damit auch die für sie zu berechnenden Varianzen und Gewichte, des Clustering bei jeder Initialisierung potentiell verschieden.

Ein Beispiel. Nach Ausführung des k-means Algorithmus auf den Merkmalsvektoren $X_D = \{F(D(t)) \mid t \in [1 : T]\}$ eines Motion Capture Datenstroms D – zur Erinnerung: F ist der Merkmalsextraktor – erhält man einerseits Clustermittelpunkte μ_j, andererseits eine Zuordnung der Merkmalsvektoren zu den ihnen nächstgelegenen Mittelpunkten. Anschließend werden für die Clustermittelpunkte μ_j Varianz σ_j^2 und Gewicht w_j bestimmt, und man erhält somit ein GMM. Die Abbildungen 3.1(a) und 3.1(b) veranschaulichen das GMM einer Beispielbewegung



(a) (b)

Abbildung 3.1: *(a) zeigt die Einteilung der Merkmalsvektoren für $f = 2$ Merkmale einer Bewegung („elbow-to-knee“) in Cluster sowie das daraus erstellte GMM. Das GMM wird hier repräsentiert durch Clustermittelpunkte μ_j (weiß), sowie Gewichte w_j (untere schwarze Halbkreise) und Varianzen σ_j^2 (obere schwarze Halbkreise). (b) zeigt dasselbe GMM wie in (a) dargestellt als Höhenmodell. Die blauen (niedrige Werte) bis roten (hohe Werte) Linien entsprechen den Isolinien des Höhenmodells.*

3.2 Training

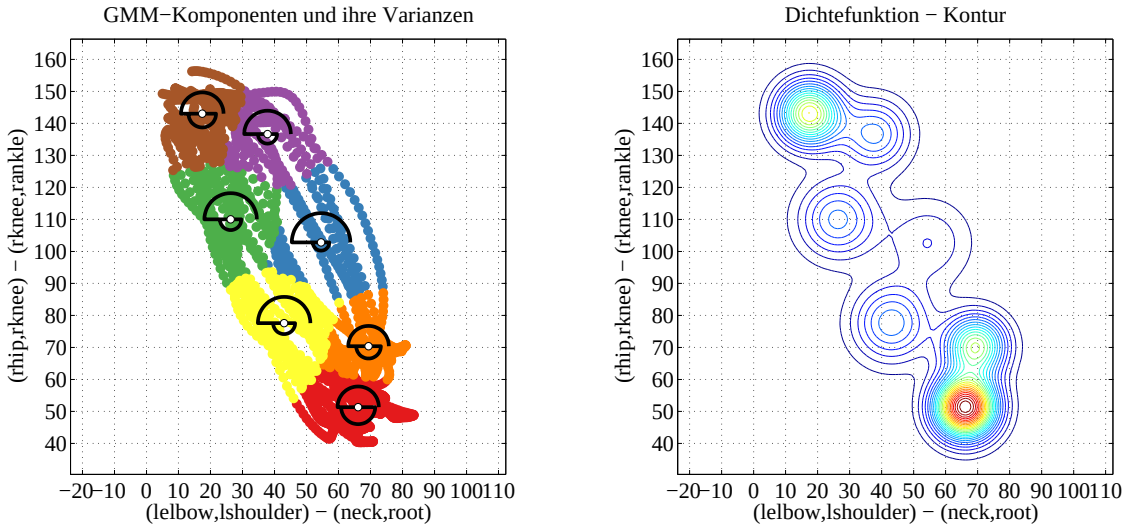
Ziel des Trainings ist eine Repräsentation einer Bewegungsklasse durch ein GMM (Klassenmodell). Ein Klassenmodell einer Bewegungsklasse $\mathcal{C}$ ist eine Überlagerung von M Gaußverteilungen, wobei jede mit einem Mixturkoeffizienten w_j gewichtet wird. Die zugehörige Wahrscheinlichkeitsdichtefunktion für einen Merkmalsvektor $x_t := F(D(t))$, $x_t \in \mathbb{R}^f$ ist dann – geschrieben in Form einer Likelihoodfunktion für die Klasse $\mathcal{C}$, $p(\cdot|\mathcal{C})$ –

$$p(x_t|\mathcal{C}) = \sum_{j=1}^{M} w_j p_j(x_t). \tag{3.2.1}$$

Diese Gleichung beschreibt die Wahrscheinlichkeit (auch Likelihood) eines für die Bewegungsklasse $\mathcal{C}$ gelernten Klassenmodells für *einen* Vektor x_t (zum Zeitpunkt t). Im Zuge des Trainings wird das zu einer Bewegungsklasse gehörige GMM auf Grundlage der Merkmalsvektoren $X_\mathcal{C}$ aller Trainings-Motion-Capture-Daten $D \in \mathcal{C}$ einer Klasse $\mathcal{C}$ der Datenbank $\mathcal{D}$ erstellt.

Das Klassenmodell entspricht dann der Wahrscheinlichkeitsdichtefunktion der gesamten Sequenz von Merkmalsvektoren $X_\mathcal{C}$. Oder, in Form einer Likelihoodfunktion ausgedrückt, entspricht es der Wahrscheinlichkeit, dass die Daten $X_\mathcal{C}$ durch das Klassenmodell repräsentiert werden.

Für eine Liste von Trainingsklassen sowie beanspruchte Zeiten zum Erstellen der Klassen-GMMs (inklusive Merkmalsextraktion) siehe Tabellen A.1 und A.2 im Anhang A. Das Ergebnis des Trainings einer Klasse auf Grundlage zweier Merkmale wird in Abb. 3.2 veranschaulicht.



(a) (b)

Abbildung 3.2: *(a) zeigt die Einteilung der Merkmalsvektoren für $f = 2$ Merkmale einer Klasse („elbow-to-knee“) in Cluster sowie das daraus gelernte GMM. Das GMM wird hier repräsentiert durch Clustermittelpunkte μ_j (weiß), sowie Gewichte w_j (untere schwarze Halbkreise) und Varianzen σ_j^2 (obere schwarze Halbkreise). (b) zeigt dasselbe GMM wie in (a) dargestellt als Höhenmodell. Die blauen (niedrige Werte) bis roten (hohe Werte) Linien entsprechen den Isolinien des Höhenmodells.*

3.3 Klassifikation

Die Klassifikation strebt an, einen unbekannten Motion Capture Datenstrom Q (auch Anfrage) in eine Datenbank $\mathcal{D}$ einzuordnen, d.h. eine Klasse in $\mathcal{D}$ zu finden, der Q am meisten ähnelt. Auch Q ist modellierbar als eine Abbildung $Q : [1 : T] \longrightarrow \mathcal{P}$, mit $T \in \mathbb{N}$ vielen Posen, $[1 : T] = \{1, \ldots, T\}$ die Zeitachse (abgetastet unter fester Samplingrate) und $\mathcal{P}$ der Posenraum. Bleibt zu klären, wie man die Ähnlichkeit zwischen der Anfrage Q und einer Klasse $\mathcal{C}$ misst.

Wir werden dazu im Folgenden drei verschiedene Ansätze betrachten. Erstens werden wir Q nur aufgrund ihrer Merkmalsvektoren mit den gelernten Klassenmodellen aller Klassen vergleichen (Abschnitt 3.3.1). Zweitens kann man neben Merkmalsvektoren auch Clusterzentren für die Anfrage extrahieren und diese als Datengrundlage für den Vergleich verwenden (Abschnitt 3.3.2). Drittens besteht die Möglichkeit auch für Q ein komplettes GMM zu erstellen, um zur Klassifikation zwei GMMs heranzuziehen (das der Klasse und das der Anfrage, Abschnitt 3.3.3).

Ergebnis der Klassifikation einer Anfrage Q ist für jeden der drei Ansätze ein Vektor S der Länge γ aus Bewertungen (oder Scores), der für jede Klasse $\mathcal{C}$ einen Wert $S_{\mathcal{C}}$ beinhaltet. $S_{\mathcal{C}}$ ist eine Beurteilung der Klassifikationsgüte, wenn Q der Klasse $\mathcal{C}$ zugeteilt würde. In den Tabellen 3.1 und 3.2 sind die Berechnungszeiten für die im Folgenden Klassifikationsansätze für eine Folge von Anfragen zusammengestellt.

3.3.1 Klassifikation anhand von Merkmalsvektoren

Dieser Ansatz ist mit den geringsten Vorverarbeitungskosten verbunden; für eine Anfrage Q der Länge T müssen lediglich die Merkmalsvektoren $X_Q = \{F(Q(t)) := x_t \mid t \in [1 : T]\}$ extrahiert werden. Anschließend errechnet sich die Bewertung dieser Merkmalsvektoren bezogen auf eine Bewegungsklasse $\mathcal{C}$ aus der Likelihoodfunktion von X_Q und $\mathcal{C}$, nämlich

$$p(X_Q|\mathcal{C}) = p(x_1, \ldots, x_T|\mathcal{C}). \tag{3.3.1}$$

Damit die Bewertung nicht von der Länge einer Anfrage beinflusst wird, führen wir einen Normierungskoeffizienten $\frac{1}{T}$ ein. Es ist des Weiteren üblich anzunehmen, dass die Merkmalsvektoren X_Q stochastisch unabhängig sind, d.h. für zwei Ereignisse A, B gilt $p(A, B) = p(A)p(B)$. Gleichung (3.3.1) vereinfacht sich damit zu

$$p(x_1, \ldots, x_T|\mathcal{C}) = \left(\prod_{t=1}^{T} p(x_t|\mathcal{C}) \right)^{\frac{1}{T}}.$$

Der tatsächliche Score ergibt sich nach Logarithmierung zu

$$S_{\mathcal{C}}^{\text{feat}}(X_Q) = \log p(x_1, \ldots, x_T|\mathcal{C}) = \frac{1}{T} \sum_{t=1}^{T} \log p(x_t|\mathcal{C}), \tag{3.3.2}$$

wobei sich $p(x_t|C)$ nach Gleichung (3.2.1) berechnet. Die Bewertung einer Folge von Merkmalsvektoren wird hier also über die Summe posenweiser log-Likelihoods definiert. Abb. 3.3 zeigt das Ergebnis der Anwendung von $S_{\mathcal{C}}^{\text{feat}}(X_Q)$ auf 20 Klassen, worin X_Q die Merkmalsvektorfolge einer elbow-to-knee-Bewegung ist. Wir werden diesen Klassifikator später auch als GMM-Data Klassifikator bezeichnen.

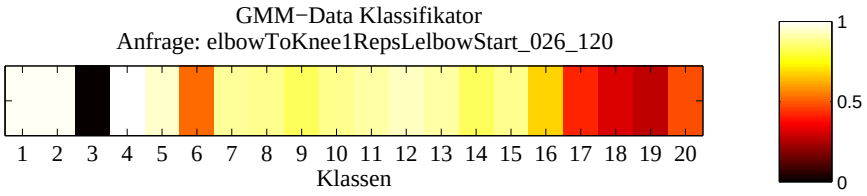


Abbildung 3.3: *Ergebnis der Anwendung von $S_{\mathcal{C}}^{\text{feat}}$ (nach Normierung auf $[0,1]$) auf die Klassen der Datenbank $\mathcal{D}^{20}$ für eine elbow-to-knee-Bewegung. Für Training und Klassifikation wurden $f = 2$ Merkmale verwendet (rechtes Knie und linke Schulter bzgl. Körpersenkrechter). Die höchsten Werte werden (bei einer rein visuellen Beurteilung) für Klasse 1 (cartwheel), 2 (depositFloor) oder 4 (elbowToKnee) angenommen.*

Anmerkung. Wir rechnen durchweg mit logarithmierten Likelihoods, sowohl in diesem Klassifikationsansatz als auch in allen weiteren. Für die Score-Werte hat das folgenden Effekt: Geht die Wahrscheinlichkeit – hier etwa $p(X_Q|\mathcal{C})$ – gegen null, strebt der Score gegen minus Unendlich. Für die Klassifikation stellt das kein Problem dar, für die Visualiserung der Scores hingegen sehr wohl.

Wir verwenden für die Visualisierungen der Bewertungen stets relative auf das Intervall $[0,1]$ normierte Vektoren und Matrizen. Für einen Vektor (oder eine Spalte einer Matrix) Y der Länge $\gamma \in \mathbb{N}$ mit Einträgen $Y = (y_1, y_2, \ldots, y_\gamma)$ berechnen wir den relativen Vektor $\overline{Y} = (\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_\gamma)$ wie folgt:

$$\overline{Y} = \frac{Y - \min\limits_{1 \le i \le \gamma} (y_i)}{\max\limits_{1 \le j \le \gamma} (y_j) - \min\limits_{1 \le i \le \gamma} (y_i)}.$$

Diese Transformation ist ordnungserhaltend, d.h. aus $y_i \le y_j$ folgt $\overline{y}_i \le \overline{y}_j$. Damit werden Ergebnisse verschiedener Klassifikationen – entweder zweier Anfragen oder zweier Klassifikatoren untereinander – vergleichbarer. Für einen einzelnen Vektor (respektive eine einzelne Spalte einer Matrix) kann die Aussagekraft jedoch verlorengehen. Dies tritt ein, wenn ein betragsmäßig sehr großer Eintrag darin vorkommt. Denn in diesem Fall rücken die Werte bei der Visualisierung farblich nah zusammen und machen eine optische Unterscheidung oder Bewertung schwierig bis unmöglich. Bei den in dieser Arbeit abgebildeten Grafiken kam es zu solchen Ausreißern nicht.

3.3.2 Klassifikation anhand von Clusterzentren

Im Großen und Ganzen entspricht diese Klassifizierungsmethode sehr der vorangegangenen. Jedoch werden in der Vorverarbeitungsphase nicht nur Merkmale X_Q der Anfrage Q extrahiert, sondern außerdem über ein k-means Verfahren Clusterzentren μ_j ermittelt. Für $k = M$ bilden diese die Datengrundlage $X_{\tilde{Q}} = \{\mu_j \mid j \in [1 : M]\}$. Wie in Abschnitt 3.3.1 ergibt sich auch hier die Bewertung der Anfrage bzgl. einer Klasse $\mathcal{C}$ aus der Summe der log-Likelihoods, diesmal jedoch nicht posen-, sondern clusterweise,

$$S_{\mathcal{C}}^{\text{cluster}}(X_Q) = \frac{1}{M} \sum_{j=1}^{M} \log p(\mu_j|\mathcal{C}). \tag{3.3.3}$$

Wählt man stets dieselbe Anzahl Cluster M für Anfragen Q, so kann man auf den Normalisierungskoeffizienten $\frac{1}{M}$ auch verzichten, ohne das Gesamtergebnis zu verfälschen. Wir werden ihn

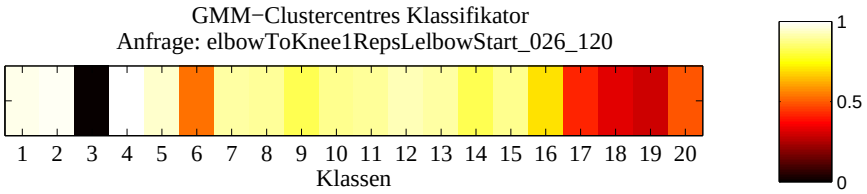


Abbildung 3.4: *Ergebnis der Anwendung von $S_{\mathcal{C}}^{\text{cluster}}$ (nach Normierung auf $[0,1]$) auf die Klassen der Datenbank $\mathcal{D}^{20}$ für eine elbow-to-knee-Bewegung. Für Training und Klassifikation wurden $f = 2$ Merkmale verwendet (rechtes Knie und linke Schulter bzgl. Körpersenkrechter). Hier werden die höchsten Werte bei einer rein visuellen Beurteilung für Klasse 1 (cartwheel), 2 (depositFloor) oder 4 (elbowToKnee) angenommen.*

jedoch aus Konsistenzgründen beibehalten. Für das Ergebnis der Anwendung von $S_{\mathcal{C}}^{\text{cluster}}$ auf eine Bewegung und 20 Klassen siehe Abb. 3.4.

3.3.3 Klassifikation anhand der GMMs

In einem weiteren Ansatz sollen zwei GMMs direkt verglichen werden. Für die Vorverarbeitung einer Anfrage Q bedeutet das erst einmal, dass nicht nur Merkmalsvektoren X_Q extrahiert werden und Clusterzentren μ_j berechnet, sondern auch Kovarianzen Σ_j und Gewichte w_j, also ein komplettes GMM $\theta_Q = (w_{Qj}, \mu_{Qj}, \Sigma_{Qj})_{j=1,\dots,M}$ erstellt wird.

Der Vergleich eines Anfragemodells θ_Q mit einem Klassenmodell $\theta_C = (w_{Ci}, \mu_{Ci}, \Sigma_{Ci})_{i=1,\dots,N}$ wurde hier auf zwei verschiedene Weisen realisiert: Einerseits über eine Kombination aus einer sogenannten Earth Movers Distance, kurz EMD, und symmetrischer Kullback Leibler Distanz, auch KL-Distanz Die Theoretischen Grundlagen hierfür werden in [5] erläutert und in [14, 16] implementiert. Wir verwenden diese Implementation für die in diesem Abschnitt aufgeführten Berechnungen. Andererseits wurde ein Monte Carlo Sampling verwendet, um θ_Q und θ_C basierend auf den von ihnen gesampelten Daten zu vergleichen Dieses Verfahren wird in [1] beschrieben und ebenfalls in [14] implementiert.

EMD und KL. Zur Berechnung des Abstandes zwischen θ_C und θ_Q gehen Logan und Salomon in [5] wie folgt vor: Sie berechnen eine Kostenmatrix $\Delta = (d_{ij})$, für $i = 1, \dots, N$ und $j = 1, \dots, M$. Diese Matrix stellen sie unter Verwendung der symmetrischen KL-Distanz zusammen, die für zwei Clusterzentren μ_{Ci} und μ_{Qj} die Form

$$d_{ij} = \Sigma_i \Sigma_j^{-1} + \Sigma_j \Sigma_i^{-1} + (\mu_i - \mu_j)^2 \left(\Sigma_i^{-1} + \Sigma_j^{-1}\right)$$

annimmt. Des Weiteren suchen sie eine Flussmatrix $F = (f_{ij})$, der für einen Eintrag f_{ij} den Fluss zwischen μ_{Ci} und μ_{Qj} bezeichnet und die Gesamtkosten

$$W(\theta_Q, \theta_C, F) = \sum_{i=1}^{N} \sum_{j=1}^{M} d_{ij} f_{ij}$$

minimiert. Ein Eintrag f_{ij} entspricht somit den Kosten, die aufzuwenden sind, um Wahrscheinlichkeitsmasse von einem Cluster μ_{Ci} (Quelle) auf ein anderes μ_{Qj} (Senke / Ziel) umzuverteilen.

Bildlich kann man sich die Quellcluster als Erdhügel vorstellen und die Zielcluster als Erdlöcher. Der Fluss entspricht dann der Arbeit eines Baggers, der versucht die Erdhügel in die Löcher umzuschichten; daher der Name *Earth Movers Distance*. Dabei unterliegt f_{ij} einigen Bedingungen, die u.a. dafür sorgen, dass Masse nur von θ_C nach θ_Q befördert wird und nicht umgekehrt, dass weder mehr von θ_C geschickt wird als vorhanden ist, noch θ_Q mehr erhält, als es fassen kann und dass insgesamt soviel Masse wie möglich bewegt wird (vergleiche [17]).

Nach der Berechnung eines solchen minimalen Flusses ist die EMD definiert als die errechnete Arbeit normalisiert mit dem Gesamtfluss:

$$\mathrm{EMD}(\theta_C, \theta_Q) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} d_{ij} f_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{M} f_{ij}}$$

Wir werden hier jedoch den negativen Abstand betrachten, damit die Klassifikation einheitlich als Maximierungsaufgabe behandelt werden kann. Darüber hinaus gilt im Zuge dieser Diplomarbeit $M = N$ für die Anzahl der Komponenten eines GMM. So ergibt sich die Bewertung einer Klasse zu

$$S_{\mathcal{C}}^{\mathrm{emd}}(X_Q) = -\mathrm{EMD}(\theta_C, \theta_Q) = -\frac{\sum_{i=1}^{M}\sum_{j=1}^{M} d_{ij} f_{ij}}{\sum_{i=1}^{M}\sum_{j=1}^{M} f_{ij}}. \tag{3.3.4}$$

In Abb. 3.5 wurde $S_{\mathcal{C}}^{\mathrm{emd}}$ für die elbow-to-knee-Bewegung auf 20 Klassen ausgewertet. Wenn wir uns im weiteren Verlauf auf diesen Klassifikationsansatz beziehen, nennen wir ihn KL-EMD Klassifikator.

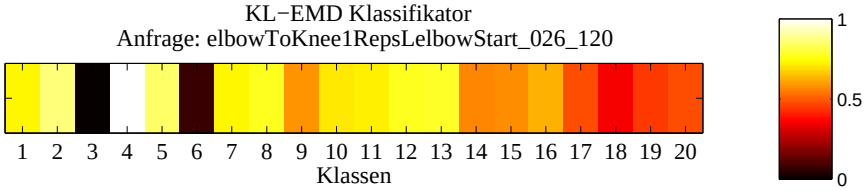


Abbildung 3.5: *Ergebnis der Anwendung von $S_{\mathcal{C}}^{\mathrm{emd}}$ (nach Normierung auf $[0, 1]$) auf die Klassen der Datenbank $\mathcal{D}^{20}$ für eine elbow-to-knee-Bewegung. Für Training und Klassifikation wurden $f = 2$ Merkmale verwendet (rechtes Knie und linke Schulter bzgl. Körpersenkrechter). Die höchsten Werte werden hier (visuelle Beurteilung) für Klasse 4 (elbowToKnee) angenommen.*

Die EMD ist im Allgemeinen keine Metrik. Auch hier nicht, denn sie definiert nur dann eine Metrik, wenn die GMMs mit denselben Gewichten arbeiten (und nicht jedes seine eigenen hätte) und der Abstandsmatrix Δ eine Metrik zugrunde liegt. Sonst sind zwar Nichtnegativität und Symmetrie erfüllt, nicht aber die Dreiecksungleichung (siehe [17]).

Monte Carlo Sampling. Die Autoren von [1] schlagen ein auf Monte Carlo Sampling basierendes Verfahren vor, um zwei GMMs zu vergleichen. Im Wesentlichen entspricht diese Methode der in Abschnitt 3.3.1 vorgestellten, mit dem Unterschied, dass sie symmetrisiert wurde und hier die Datengrundlage nicht benötigt wird. Sie wird ersetzt durch vom Klassenmodell und vom Anfragemodell gesampelte Daten $\tilde{X}_{\mathcal{C}}$ und $\tilde{X}_Q$. Beide Datenvektoren werden unter Verwendung einer festen Anzahl Sampels erstellt. Eine Bewertung ergibt sich aus der Likelihoodfunktion der gesampelten

Daten unter dem jeweils anderen GMM, bzw. aus

$$S_{\mathcal{C}}^{\mathrm{mc}}(X_Q) = -\left(p(\tilde{X}_Q|Q) + p(\tilde{X}_{\mathcal{C}}|\mathcal{C}) - p(\tilde{X}_Q|\mathcal{C}) - p(\tilde{X}_{\mathcal{C}}|Q)\right). \tag{3.3.5}$$

Angesichts eines hohen Rechenaufwands, der mit dem Monte Carlo Sampling verbunden ist, sollte man diese Methode eher nur dann verwenden, wenn die wahre Datengrundlage in der anschließenden Likelihoodberechnung unberücksichtigt bleiben soll, oder aber aufgrund der Größe der Datenbank nicht mitgespeichert wurde.

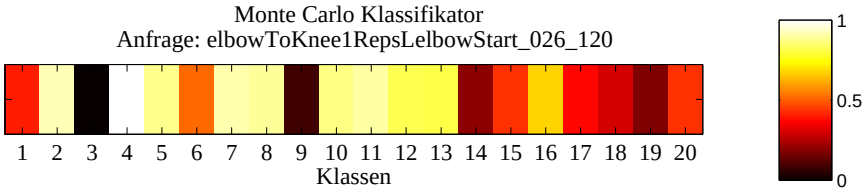


Abbildung 3.6: *Ergebnis der Anwendung von $S_{\mathcal{C}}^{\mathrm{mc}}$ (nach Normierung auf $[0,1]$) auf die Klassen der Datenbank $\mathcal{D}^{20}$ für eine elbow-to-knee-Bewegung. Für Training und Klassifikation wurden $f = 2$ Merkmale verwendet (rechtes Knie und linke Schulter bzgl. Körpersenkrechter). Die 4. Klasse (elbowToKnee) ist augenscheinlich die mit dem höchsten Wert.*

3.3.4 Vorhersage einer Klasse

Die Bewertungsmodelle spiegeln negiert den Abstand einer zu klassifizierenden Anfrage Q zum jeweiligen Klassenmodell wider. Für jede Klasse erhält man auf diese Weise eine Vorhersage, wie gut ihr gelerntes Klassenmodell (GMM) zu der Merkmalsvektorfolge X_Q der Anfrage passt. Eine Entscheidung für eine Klasse $\tilde{\mathcal{C}} \in \mathcal{D} = (\mathcal{C}_1 \dot{\cup} \ldots \dot{\cup} \mathcal{C}_\gamma)$ fällt über eine Maximumsbildung

$$\tilde{\mathcal{C}} = \arg \max_{\mathcal{C} \in \{\mathcal{C}_1,\ldots,\mathcal{C}_\gamma\}} S_{\mathcal{C}}(X_Q)$$

und kann damit aus statistischer Sicht (vor allem für die beiden ersten Methoden) als eine Art Maximum Likelihood Schätzung angesehen werden.

Sieht man sich unter diesem Aspekt Abbildungen 3.3 - 3.6 an, so würde man die Anfrage in 3.5 und 3.6 in Klasse 4 (elbowToKnee) einordnen. Dies wäre auch die richtige Klasse. Bei den beiden anderen Bildern ist eine eindeutige Klassifizierung mit bloßem Auge kaum möglich. Hier würde man zwischen den Klassen 1 (cartwheel), 2 (depositFloor) und 4 (elbowToKnee) schwanken.

3.4 Ergebnisse

Um einen Eindruck von der Güte des vorgestellten Klassifikationssystems zu gewinnen, werden wir nicht nur eine einzelne Anfrage an das System stellen, sondern eine ganze Reihe von Anfragen. Zu Demonstrationszwecken wurde eine kleine Datenbank $\mathcal{D}^{20}$ verwendet, bestehend aus einer Auswahl von zwanzig Klassen der Datenbank $\mathcal{D}^{57}$ in Anhang A Tabelle A.1. Eine Auflistung der zwanzig Klassen ist in Tabelle A.2 zusammengestellt. $\mathcal{D}^{20}$ wird unterteilt in zwei disjunkte Teildatenbanken, von denen eine als Trainingsdatenbank $\mathcal{D}_T^{20}$, die andere als Evaluationsdatenbank $\mathcal{D}_E^{20}$

dient. Wir werden außerdem nun das gesamte Merkmalssortiment aus $f = 11$ Gelenkwinkeln (siehe Tabelle 2.1) für Training und Klassifikation nutzen. Die Anzahl der Komponenten für ein GMM wird auf sieben festgelegt.

In einem ersten Schritt werden die Klassen in $\mathcal{D}_T^{20}$ trainiert, d.h. für jede Klasse ein sie repräsentierendes GMM erstellt (siehe Abb. 3.2 für ein zweidimensionales Beispiel der Klasse elbow-to-knee). Im nächsten Schritt werden alle Motion Capture Daten Q in $\mathcal{D}_E^{20}$ nacheinander als Anfragen an $\mathcal{D}_T^{20}$ gestellt.

3.4.1 Prädiktionsmatrix (Prediction Matrix)

Ordnet man alle $n = |\mathcal{D}_E^{20}|$ vielen Anfragen nebeneinander und klassenweise untereinander an, so erhält man eine $\gamma \times n$ Prädiktionsmatrix, mit $\gamma = 20$ die Anzahl Klassen in $\mathcal{D}_T^{20}$. Die Tabellen 3.1 und 3.2 zeigen eine Gegenüberstellung der Berechnungszeiten der einzelnen Klassifikatoren für die kleine Datenbank $\mathcal{D}^{20}$ und die große Datenbank $\mathcal{D}^{57}$, die in Anhang A aufgelistet sind.

Klassifikator	Zeit [sec]
GMM vs Daten	14.561
GMM vs Clusterzentren	3.194
KL und EMD	25.537
Monte Carlo Sampling	54.118

Tabelle 3.1: *Zeit zum Erstellen einer Prädiktionsmatrix für 236 Evaluationsdaten in 20 Klassen der Datenbank $\mathcal{D}^{20}$.*

Klassifikator	Zeit [sec]
GMM vs Daten	113.974
GMM vs Clusterzentren	25.297
KL und EMD	206.146
Monte Carlo Sampling	461.654

Tabelle 3.2: *Zeit zum Erstellen einer Prädiktionsmatrix für 655 Evaluationsdaten in 57 Klassen der Datenbank $\mathcal{D}^{57}$.*

Abb. 3.7 auf Seite 22 zeigt eine solche Prädiktionsmatrix für jeden der vier Klassifikationsansätze. Abgebildet ist jedoch nicht die absolute Prädiktionsmatrix, sondern eine leicht modifizierte. Da die Klassifikation selbst letztendlich spaltenweise durchgeführt wird, ist es sinnvoll die absolute Prädiktionsmatrix durch eine relative zu ersetzen, die spaltenweise mit dem Maximum der Spalte normiert wurde.

Betrachtet man die Prädiktionsmatrizen, so kann man aus ihnen ersehen, wo es zu Klassenverwechslungen kommen kann. Vom Gesamteindruck erscheinen die Abbildungen 3.7 (a) und (b) in weiten Teilen heller als (c) und (d). Dies deutet darauf hin, dass die Klassen in (c) und (d) besser differenziert werden können als die in (a) und (b). Leider heißt das nicht, dass sie auch richtiger klassifizieren.

Schaut man sich die Matrizen etwas genauer an, so stellt man fest, dass beinahe durchgehend hohe Werte für die Klassen 17, 18, 19 und 20 untereinander angenommen werden. Es ist damit wahrscheinlich, dass sie in einer Klassifikation verwechslungsgefährdet sind. Es handelt sich dabei um Bewegungen des Typs walk, walkBackwards, walkLeftCircle und walkRightCircle – also ‘gehen’, ‘rückwärts gehen’, ‘im Kreis linksherum gehen’ und ‘im Kreis rechtsherum gehen’ – und damit tatsächlich um sehr ähnliche Bewegungen. Ähnlich steht es u.a. um die Klassen 2 und 5 (depositFloor und grabFloor – ‘auf den Boden legen’ und ‘vom Boden aufheben’) 14 und 15 (rotateArmsBackward und rotateArmsForward – ‘Arme rückwärts rotieren’ und ‘Arme vorwärts rotieren’).

Bezogen auf Abb. 3.7 (a) und (b) stellt man fest, dass so gut wie alle Anfragen tendentiell gut auf die erste Klasse (cartwheel – ‘Radschlag’) passen. Die Cartwheel-Bewegung ist relativ komplex. Dies wird durch ein eher allgemein gehaltenes GMM modelliert. Allgemein gehalten bedeutet, dass

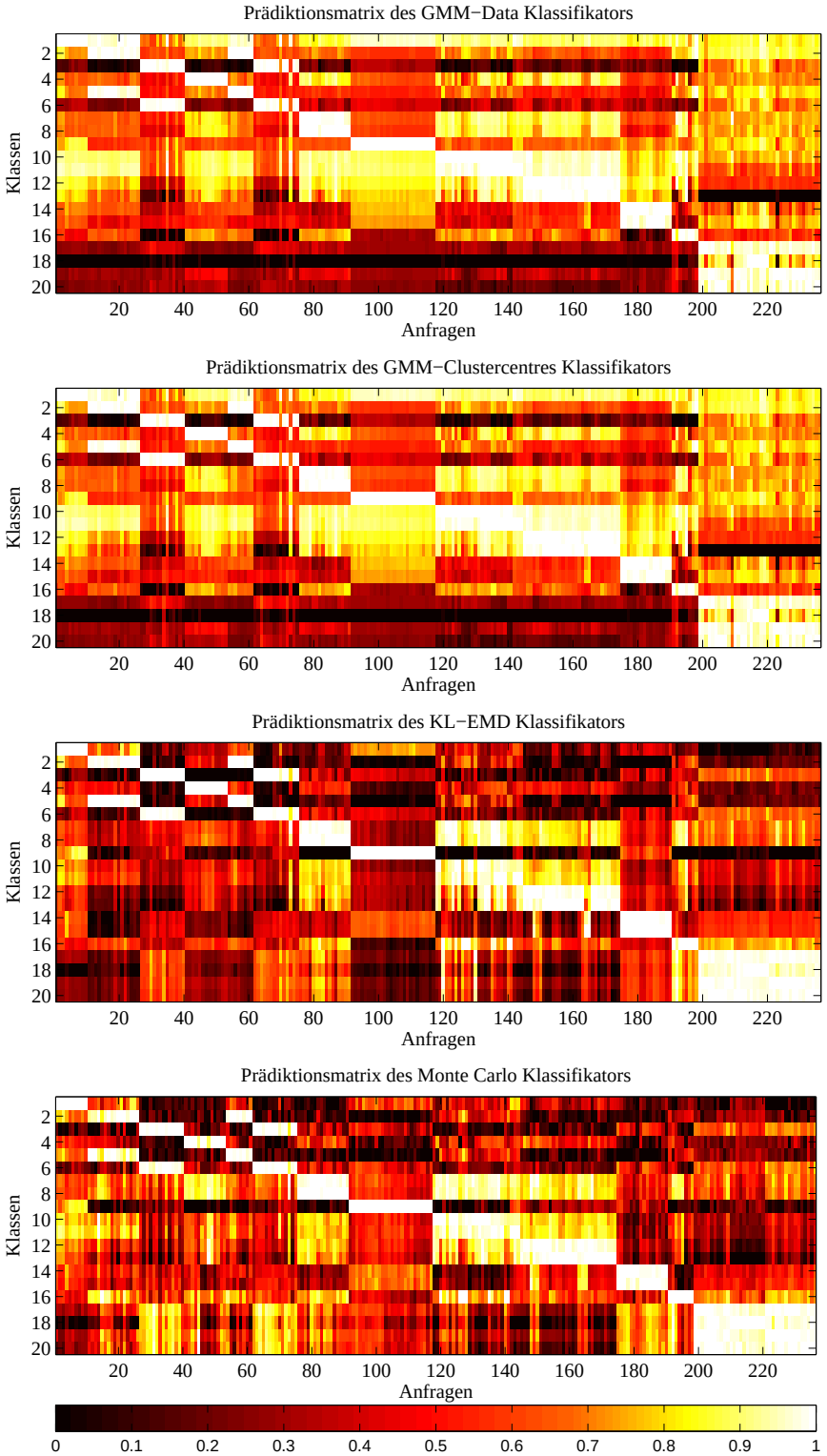


Abbildung 3.7: *Relative Prädiktionsmatrizen für die vier in Abschnitt 3.3 vorgestellten Klassifikationsverfahren auf Grundlage der Datenbank* $\mathcal{D}^{20}$.

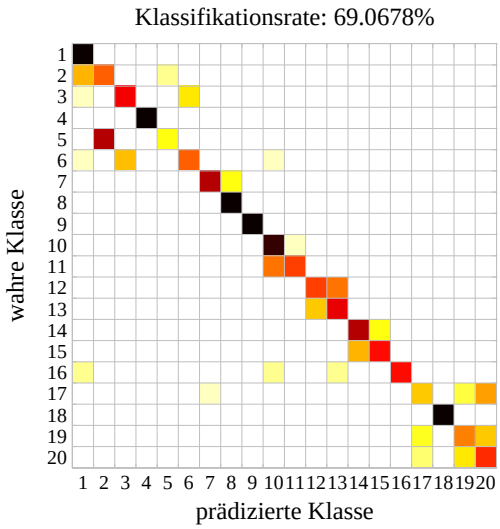
die Clusterzentren weit im Raum verstreut liegen und die Varianzen hoch sind. Daher passen die Merkmalsvektoren vieler Anfragen recht gut zu diesem GMM. Bei den beiden anderen Abb. (c) und (d) ist eine solche „globale“ Ähnlichkeit zu allen Anfragen nicht auffindbar, da die entsprechenden Klassifikatoren symmetrisch sind. Deswegen beurteilen sie nicht nur die Wahrscheinlichkeit des Klassenmodells unter der Anfrage, sondern auch umgekehrt.

3.4.2 Klassifikationsmatrix (Confusion Matrix)

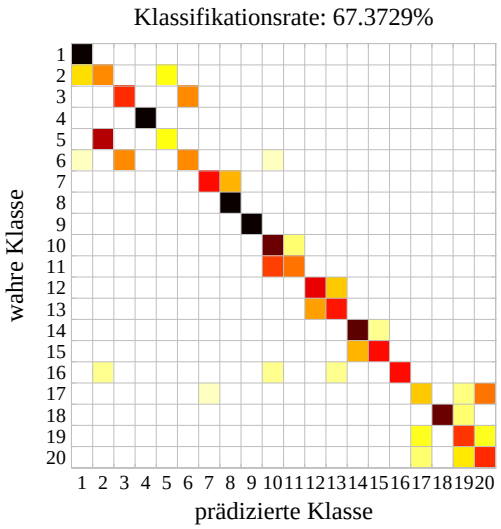
Um einen Eindruck von der (absoluten) Klassifikationsgüte zu gewinnen, kann man eine sog. Klassifikationsmatrix berechnen. In ihr werden richtig und fehlerhaft klassifizierte Anfragen gegenübergestellt. Dabei entsprechen die Zeilen der Matrix der wahren Klasse und die Spalten der jeweils prädizierten Klasse. Ein Eintrag an der Stelle (i, j) enthält die Anzahl der Anfragen aus Klasse i, die Klasse j zugewiesen wurden. Die Idealform einer Klassifikationsmatrix eine Diagonalmatrix, denn dann wären alle Evaluationsdaten richtig klassifiziert worden. Aus der Klassifikationsmatrix lässt sich eine weitere Kenngröße berechnen: die Klassifikationsrate. Sie ist das Verhältnis richtiger Klassifikationen zur Gesamtzahl der Anfragen.

Die Abbildungen 3.8(a) - 3.8(d) zeigen Klassifikationsmatrizen für die vier vorgestellten Klassifikatoren. Was in den Prädiktionsmatrizen schon erkennbar war, sieht man hier bestätigt: es ist überall schwierig die Klassen 17 (walk) und 19 (walkLeftCircle) und 20 (walkRightCircle) zu unterscheiden, dagegen hebt sich Klasse 18 (walkBackwards) im Ergebnis gut von den anderen Gehbewegungen ab. Sie wurde von nur einem Klassifikator (dem GMM-Centres Klassifikator, Abb. 3.8(b)) teilweise fehlerhaft zugeordnet. Schaut man sich die entsprechenden Prädiktionsmatrizen an dieser Stelle (Spalten 214 bis 220 enthalten walkBackwards-Bewegungen) genauer an, so stellt man fest, dass der gesamte Bereich (Klassen 17 bis 20, Spalten 199 bis 236) zwar sehr hell ist, sich aber dennoch ein kleiner zusammenhängender Teil als eine Nuance heller als die direkte Nachbarschaft präsentiert. Damit werden die entsprechenden Anfragen vom jeweiligen Klassifikator der Klasse 18 zugeordnet. Ferner werden die Klassen 2 und 5 (depositFloor und grabFloor), 3 und 6 (depositHigh und grabHigh), 12 und 13 (punchFront und punchSide), sowie 14 und 15 (rotateArmsBackward und rotateArmsForward) verwechselt. Alles in allem wenig überraschend, da diese Bewegungen sehr ähnlich sind. Für eine genauere Analyse sei der Leser auf Abschnitt 3.5 verwiesen.

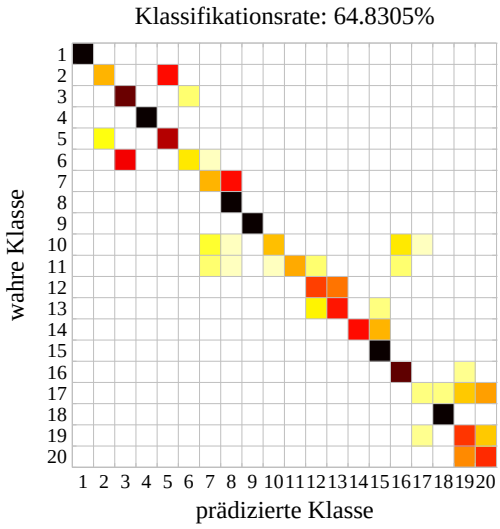
Am Ende dieses Abschnitts sind noch zwei weitere Klassifikationsmatrizen abgebildet (Abb. 3.9 Seite 25), die auf Grundlage der gesamten Datenbank $\mathcal{D}^{57}$ für die beiden Klassifikatoren GMM-Data und KL-EMD erstellt wurden. Beim Betrachten fällt nach wie vor auf, dass die bei $\mathcal{D}^{20}$ verwechselten und falsch klassifizierten Daten weiterhin falsch klassifiziert und verwechselt werden. Des Weiteren kommt es bei Bewegungen die entweder rechts- oder linksseitig bzw. vorwärts oder rückwärts ausgeführt werden zu Vertauschungen. Dies ist der Fall bei Klassen 27 bis 32 (Bewegungen, bei denen entweder beide oder ein Arm vorwärts und rückwärts rotiert wird). Auch verschiedene Gehbewegungen sind nicht gut auseinanderhaltbar (Klassen 50 bis 57). Beim KL-EMD fällt weiter auf, dass zwischen Klassen 46 und 36 (standUpSitFloor und sitDownFloor) und diesen ähnlichen nicht gut unterschieden wird. Auch beim GMM-Data Klassifikator treten hier kleinere Verwechslungen auf.



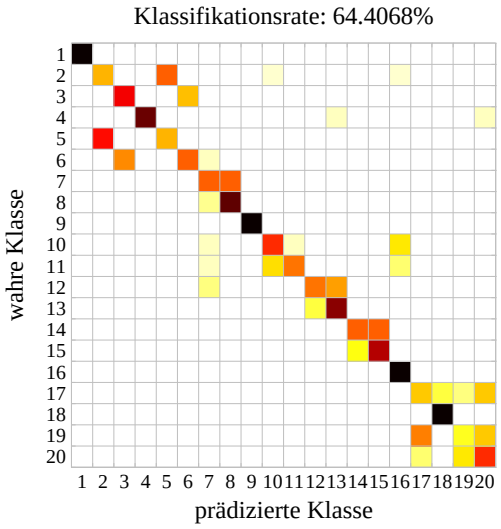
(a) GMM-Data Klassifikator



(b) GMM-Centres Klassifikator

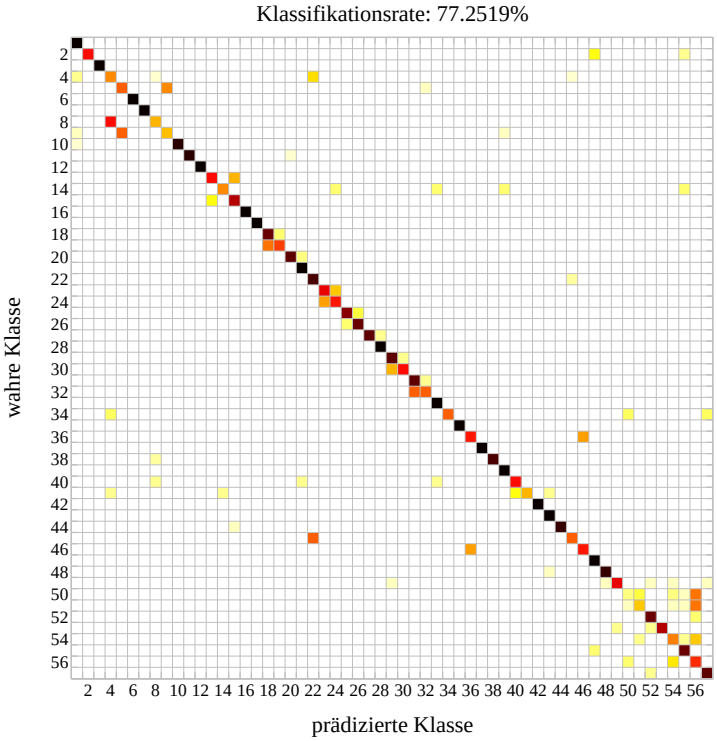


(c) KL-EMD Klassifikator

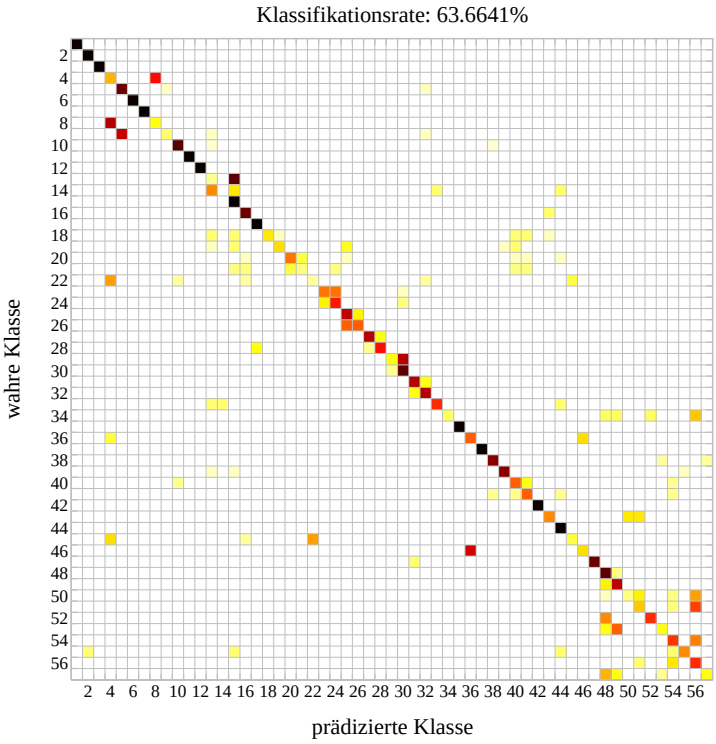


(d) Monte Carlo Klassifikator

Abbildung 3.8: *Klassifikationsmatrix und -rate der GMM-Data, GMM-Centres, KL-EMD und Monte Carlo Klassifikatoren auf Grundlage der Datenbank* $\mathcal{D}^{20}$.



(a)
GMM-Data
Klassifikator



(b)
KL-EMD
Klassifikator

Abbildung 3.9: *Klassifikationsmatrix sowie -rate für den GMM-Data Klassifikator und den KL-EMD Klassifikator auf Grundlage der Datenbank $\mathcal{D}^{57}$ unter Verwendung von Gelenkwinkeln.*

3.4.3 GMM – Variation der Komponentenanzahl

Die Anzahl der Komponenten M für ein GMM wurde zwar fest gewählt, aber es ist dennoch interessant zu sehen, was sich bei einer Erhöhung oder Verminderung dieser Anzahl an den Klassifikationsergebnissen ändert. Dies wurde für $M \in \{3, 5, 9, 11, 15\}$ getestet.

Das Klassenmodell einer Bewegungsklasse bei kleinem M (etwa $M = 3$) verliert an Repräsentativität. Ein GMM mit weniger Mixturkomponenten muss sich räumlich in den Daten zentrierter positionieren und größere Varianzen haben, um sie darzustellen als ein GMMs mit einer größeren Anzahl von Komponenten. Dies schlägt sich vor allem in den Klassifikationsraten nieder (siehe auch Tabelle 3.3). Für größere M ähnelt ein Modell immer mehr den Daten, aus denen es erstellt wurde. Dies wirkt sich positiv auf diejenigen Klassifikatoren aus, die entweder auf den Anfragedaten direkt, oder auf daraus berechneten Clusterzentren agieren. Beim Monte Carlo Sampling ist eine Erhöhung der Komponentenanzahl abträglich. Das liegt darin begründet, dass ein GMM einer Anfrage Q für eine größere Komponentenzahl zu spezifisch wird gegenüber dem GMM einer Klasse $\mathcal{C}$. Die von den beiden GMMs gesampelten Daten $\tilde{X}_Q$ und $\tilde{X}_\mathcal{C}$ unterstützen diese Entwicklung. Zwar wird in Gleichung (3.3.5) – wir wiederholen sie kurz –

$$S^{\mathrm{mc}}_{\mathcal{C}}(X_Q) = -\Big(p(\tilde{X}_Q|Q) + p(\tilde{X}_\mathcal{C}|\mathcal{C}) - p(\tilde{X}_Q|\mathcal{C}) - p(\tilde{X}_\mathcal{C}|Q)\Big)$$

die Wahrscheinlichkeit von $\tilde{X}_Q$ unter $\mathcal{C}$, $p(\tilde{X}_Q|\mathcal{C})$, einigermaßen groß ausfallen, die von $\tilde{X}_\mathcal{C}$ unter Q, $p(\tilde{X}_\mathcal{C}|Q)$, dagegen weniger. Somit bleibt der Gesamtwert $S^{\mathrm{mc}}_{\mathcal{C}}$ eher klein, da der Term $p(X_Q|Q)$ kein Gegengewicht enthält wie $p(X_\mathcal{C}|\mathcal{C})$.

Ein entscheidender Nachteil, eine höhere Anzahl M von Komponenten zu wählen, liegt aus berechnungstechnischer Sicht in den damit entstehenden Problemen. Entartungen, wie etwa leere Clusterzentren nach Ausführung des M-means Algorithmus oder Clusterzentren mit nur einem Punkt (dieses fällt genau in einen Datenpunkt und hat später Varianz null) treten bei größerem M ($M > 9$) erheblich häufiger auf. Dazu kommt die höhere Rechenzeit bei nicht sehr viel genaueren Klassifikationsraten (vergleiche Tabelle 3.4).

Die in Abschnitt 3.4.2 erwähnte Klassifikationsrate bewertet nur die jeweils beste Klassenzuweisung. Neben dieser besten Klassifizierung kann man jedoch auch die zweit-, dritt- und viertbeste Klasse in der Rate berücksichtigen. Das Ergebnis wird in Tabelle 3.5 aufgeführt.

Klassifikator	Klassifikationsrate in % für unterschiedliche Anzahlen M von Komponenten					
	$M = 3$	$M = 5$	$M = 7$	$M = 9$	$M = 11$	$M = 15$
GMM vs Daten	70.382	73.588	77.252	78.168	79.237	77.252
GMM vs Clusterzentren	68.855	71.756	74.809	77.710	78.473	78.931
KL und EMD	65.344	64.885	63.664	64.275	65.802	64.885
Monte Carlo Sampling	62.137	63.206	60.153	59.237	55.420	55.420

Tabelle 3.3: *Klassifikationsraten für verschieden komplexe GMMs (bezogen auf die Anzahl M der Komponenten) für die Datenbank $\mathcal{D}^{57}$ aus 655 Evaluationsdaten und 57 gelernte Klassen.*

Klassifikator	Klassifikationszeiten [sec] für unterschiedliche Anzahlen M von Komponenten					
	$M = 3$	$M = 5$	$M = 7$	$M = 9$	$M = 11$	$M = 15$
GMM vs Daten	67.477	90.571	113.974	137.077	161.943	212.055
GMM vs Clusterzentren	22.433	23.433	25.297	27.129	29.422	35.571
KL und EMD	49.140	113.233	206.146	325.317	479.630	872.164
Monte Carlo Sampling	297.348	385.404	461.654	561.217	658.637	825.117

Tabelle 3.4: *Übersicht der Klassifikationszeiten (Berechnung einer Prädiktionsmatrix) ohne Vorverarbeitungsaufwand für verschieden komplexe GMMs (bezogen auf die Anzahl M der Komponenten) für die Datenbank $\mathcal{D}^{57}$ aus 655 Evaluationsdaten und 57 gelernten Klassen.*

Klassifikator		Top 1	Top 2	Top 3	Top 4
$M = 3$	GMM vs Daten	70.382	86.718	90.229	93.282
	KL und EMD	65.344	83.817	88.397	92.061
$M = 7$	GMM vs Daten	77.252	90.840	93.435	95.420
	KL und EMD	63.664	84.122	89.924	92.824
$M = 9$	GMM vs Daten	78.168	90.382	93.588	96.183
	KL und EMD	64.275	83.969	88.702	92.977

Tabelle 3.5: *Klassifikationsraten für die vier besten Klassen (Top 1 bis Top 4) ausgewertet für GMMs mit 3, 7 und 9 Komponenten und den GMM-Data sowie den KL-EMD Klassifikator. Die Ergebnisse beziehen dich auf die Datenbank $\mathcal{D}^{57}$.*

3.4.4 Variation der Feature I – relationale Merkmale

Bisher haben wir nur die Standardmerkmalsmenge der Gelenkwinkel betrachtet. In diesem Abschnitt werden wir Training und Klassifikation mit den relationalen Merkmalen (Abschnitt 2.4) durchführen.

In Analogie zur vorangegangenen Präsentation der Ergebnisse bei Verwendung von Gelenkwinkeln werden wir uns zuerst die (relativen) Prädiktionsmatrizen ansehen, anschließend die Klassifikationsmatrizen. Allerdings schränken wir uns dabei auf nur zwei Klassifikatoren ein, und zwar den GMM-Data und den KL-EMD Klassifikator.

Prädiktionsmatrizen. Wie schon bei den Gelenkwinkeln, so scheint auch für relationale Merkmale die Prädiktionsmatrix des GMM-Data Klassifikators (Abb. 3.10 (a)) die Differenzierung verschiedener Klassen schlechter als beim KL-EMD Klassifikator (Abb. 3.10 (b)), da letztere in weiten Teilen dunkler eingefärbt ist. Wieder ist dies kein Garant für ein besseres Klassifikationsergebnis. Man stellt in beiden Fällen fest, dass es auch hier zu paarweisen Verwechslungen ähnlicher Klassen kommen wird. So erachten beide Klassifikatoren jogLeftCircle und jogRightCircle (Klassen 7 und 8) sowie diverse Gehbewegungen (sneak, walk, walkLeftCircle, walkRightCircle, bzw. Klassen 16, 17, 19, 20) für gleichartig. Dabei wird das Rückwärtsgehen in beiden Matrizen eher einem Schleichen gleichgesetzt als einer der anderen Gehbewegungen.

Beim GMM-Data Klassifikator schlechter zu unterscheiden als beim KL-EMD sind des Weiteren die Klassen 10, 11 sowie 12, 13 (kickLFront, kickLSide sowie punchLFront, punchLSide). Zwar

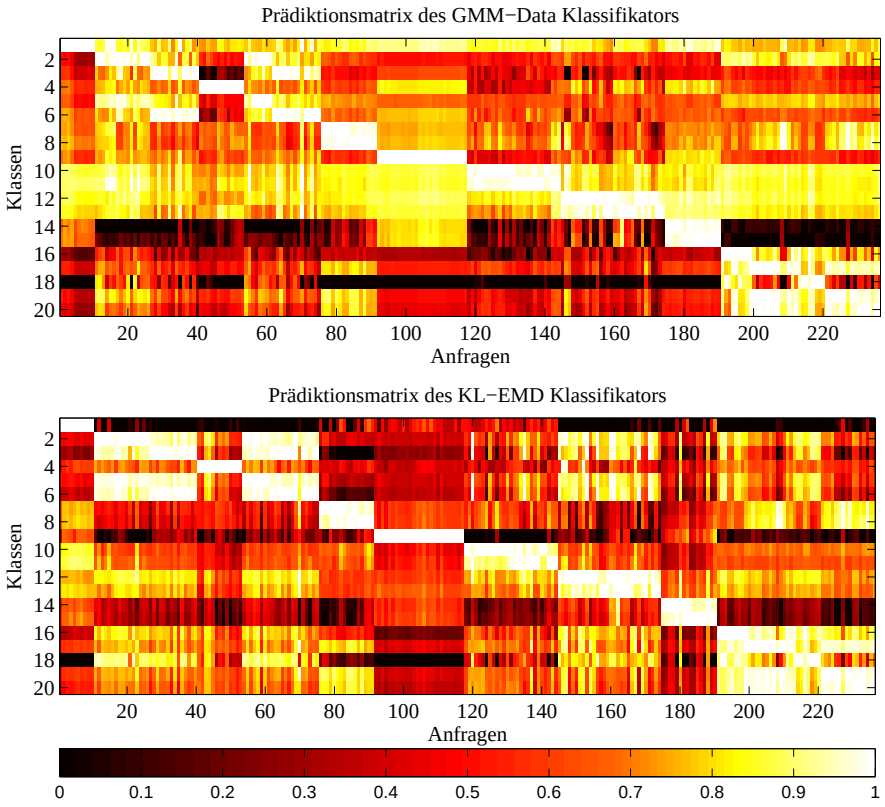


Abbildung 3.10: *Relative Prädiktionsmatrizen für zwei der vier in Abschnitt 3.3 vorgestellten Klassifikationsverfahren unter Verwendung von relationalen Merkmalen.*

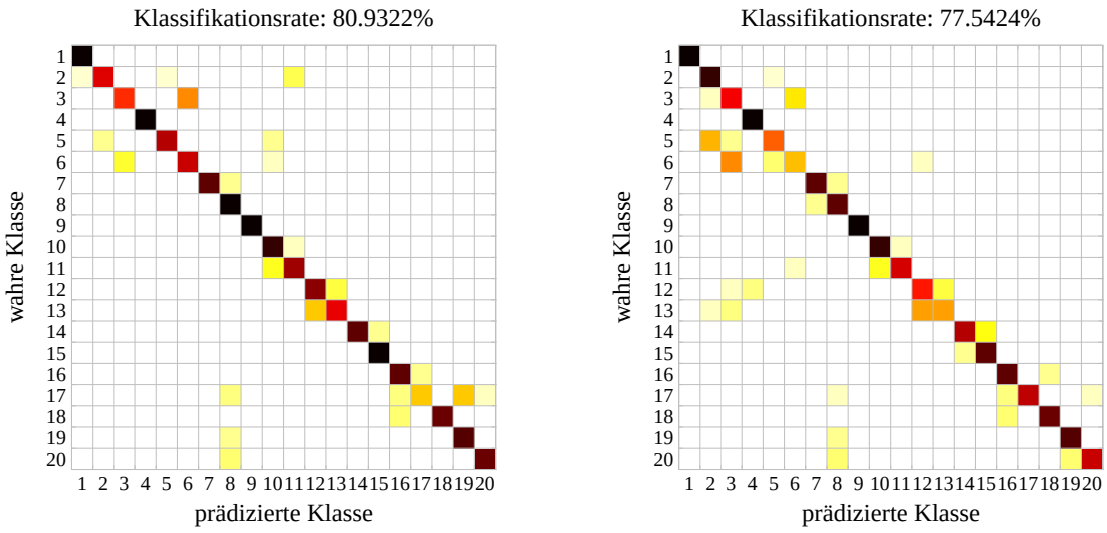
sind diese Klassen auch beim KL-EMD Klassifikator in hohem Maße verwechslungsgefährdet, aber man kann hier in Teilen erkennen, wohin sie (richtiger- oder fälschlicherweise) klassifiziert werden.

Unter dem KL-EMD Klassifikator haben auch die Klassen 2 und 3 (depositFloorR und depositHighR) bzw. 5 und 6 (grabFloorR und grabHighR) in der Prädiktionsmatrix große Ähnlichkeit.

Eine Überlegung, die dazu geführt hatte auch relationale Merkmale für die Klassifikation zu betrachten, war die Möglichkeit, mit ihnen implizit Zeitinformationen zu integrieren. Dies sollte über Merkmalskombinationen wie HANDLEFTMOVEAPARTRELROOT und HANDLEFTTOFRONTRELTORSO sowie die rechtsseitigen Äquivalenta geschehen, die für die entsprechende Bewegung unverwechselbare Merkmalsvektoren liefern. Mit Hilfe solcher Kombinationen wäre es gegebenenfalls möglich die Klassen 14 und 15 (rotateArmsLBackward1Reps und rotateArmsLForward1Reps) voneinander zu trennen. Hierbei muss man allerdings auch bedenken, dass wir für die gewählten zwölf Merkmale „nur“ sieben Mixturkomponenten zur Verfügung haben, die sich ihrerseits beliebig über die Daten verteilen können. Es kann also durchaus geschehen, dass für eine Bewegung wenige charakteristische Merkmale im GMM von vielen nicht repräsentativen Merkmalen überdeckt werden. D.h. die sieben Mixturkomponenten spiegeln vorwiegend die nicht repräsentativen Merkmale wider und blenden die charakteristischen aus. Schaut man sich die entsprechenden Resultate in der Prädiktionsmatrix an, so kann man auch mit bloßem Auge eine kleine Verbesserung beim GMM-Data Klassifikator feststellen, eine etwas größere beim KL-EMD. Bei letzterem kann man

nun Vermutungen darüber anstellen, wohin welche Evaluationsbewegung (Spalten 175 bis 190, Zeilen 14 und 15 in der Prädiktionsmatrix) klassifiziert würde; das war vorher – mit den Gelenkwinkeln – nicht möglich.

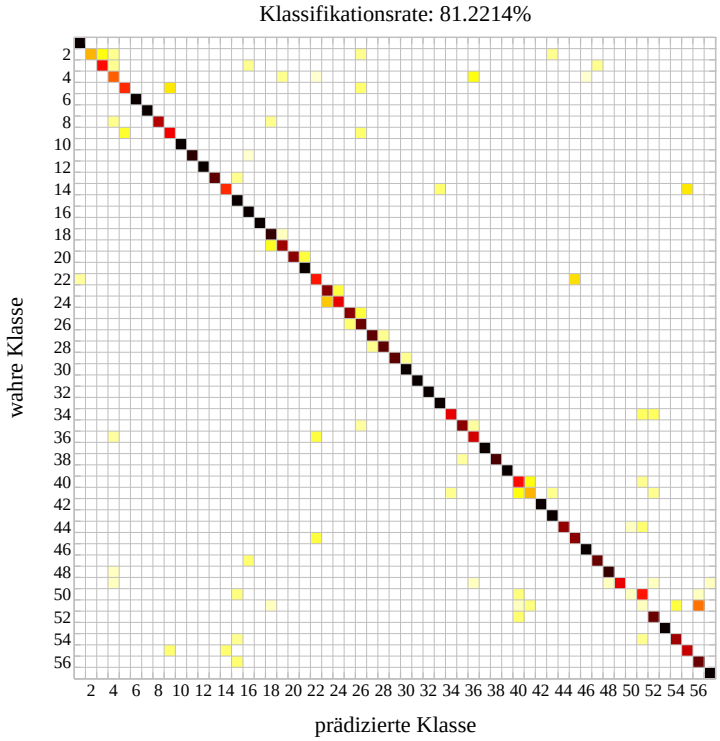
Klassifikationsmatrizen. Wirft man abschließend einen Blick auf die jeweiligen Klassifikationsmatrizen (Abbildungen 3.11(a) und 3.11(b)), bestätigen sie das Bild der Prädiktionsmatrizen. Es bleibt schwierig, Klassen 12 und 13 zu trennen (punchFront und -Side, KL-EMD). Ebenso verhält es sich für die Klasse 17 (walk, GMM-Data), aber auch für die Klassen 3 und 6 untereinander (deposit- und grabFloor, beide Klassifikatoren). Verbessert haben sich hier u.a. 10 und 11 (kick), 12 und 13 (punch) sowie walk-Bewegungen. Und auch bei den rotateArmsL[Backward,Forward]-Bewegungen (Klassen 14 und 15) haben die relationalen Merkmale tatsächlich für beide Klassifikatoren zu einer Steigerung der Klassifizierbarkeit beigetragen. Zu nennenswerten Verschlechterungen kam es nicht.



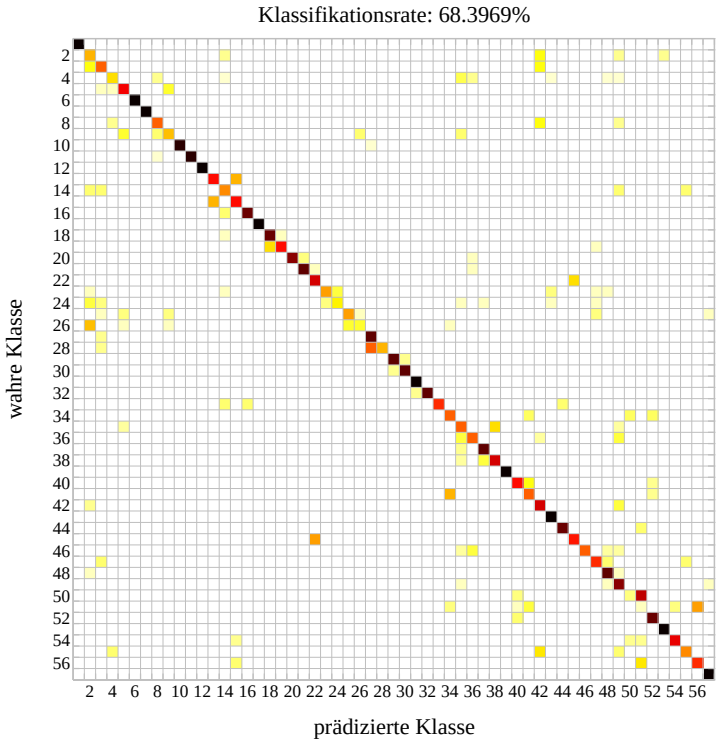
(a) GMM-Data Klassifikator (b) GMM-Centres Klassifikator

Abbildung 3.11: *Klassifikationsmatrix und -rate der GMM-Data und KL-EMD Klassifikatoren auf Grundlage der Datenbank $\mathcal{D}^{20}$ bei Verwendung relationaler Merkmale.*

Bei den großen Klassifikationsmatrizen der Datenbank $\mathcal{D}^{57}$ in Abb. 3.12 kann man im Falle des GMM-Data Klassifikators (die obere Matrix (a)) beinahe von einer ausgeprägten Diagonalen sprechen. Ausnahmen bilden neben den Klassen 2 bis 5 – hierbei handelt es sich um clap- (Klatschen) und deposit-Bewegungen – die Klassen 40 und 41 (sneak) sowie 50 und 51 (walk). Auch beim KL-EMD Klassifikator fällt die Klassifikation besser aus als bei den Gelenkwinkeln. Nach wie vor fehl schlägt sie überwiegend für die punch-Klassen (Klassen 23 bis 26), die jog[Left, Right]Circle-Klassen (13 und 15), aber auch für die walk-Klassen (50 und 51) und ähnliche. All diese werden überwiegend paarweise untereinander vertauscht. Weitere Kandidaten für Falschzuordnungen sind die Klassen 33 bis 36. Darunter sind Klassen sitDownChair (35) und sitDownFloor (36). SitDownChair wird überwiegend mit Klasse 38 vertauscht, die sitDownTable-Bewegungen enthält. Ähnliche Verwechslungen treten für das Paar standUpLieFloor und lieDownFloor (Klassen 45 und 22) auf.



(a)
GMM-Data
Klassifikator



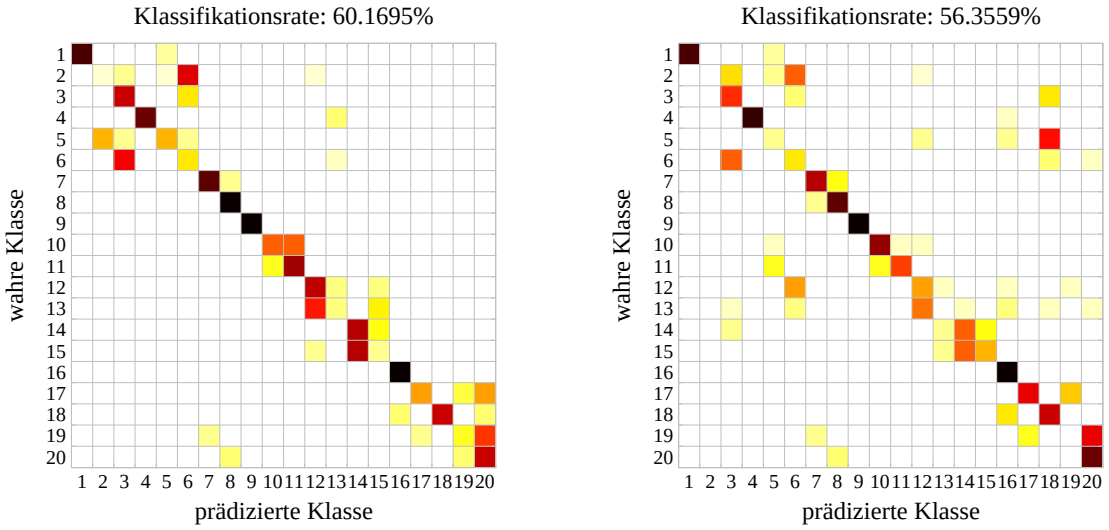
(b)
KL-EMD
Klassifikator

Abbildung 3.12: *Klassifikationsmatrix sowie -rate für den GMM-Data Klassifikator und den KL-EMD Klassifikator auf Grundlage der Datenbank $\mathcal{D}^{57}$ unter Verwendung relationaler Merkmale.*

3.4.5 Variation der Feature II – physikalische Merkmale

Wenden wir uns nun der dritten Merkmalsmenge aus physikalisch basierten Merkmalen zu (siehe Abschnitt 2.5). In diesem Abschnitt werden wir die Prädiktionsmatrizen nicht untersuchen, sondern unsere Aufmerksamkeit ausschließlich auf die Klassifikationsmatrizen lenken.

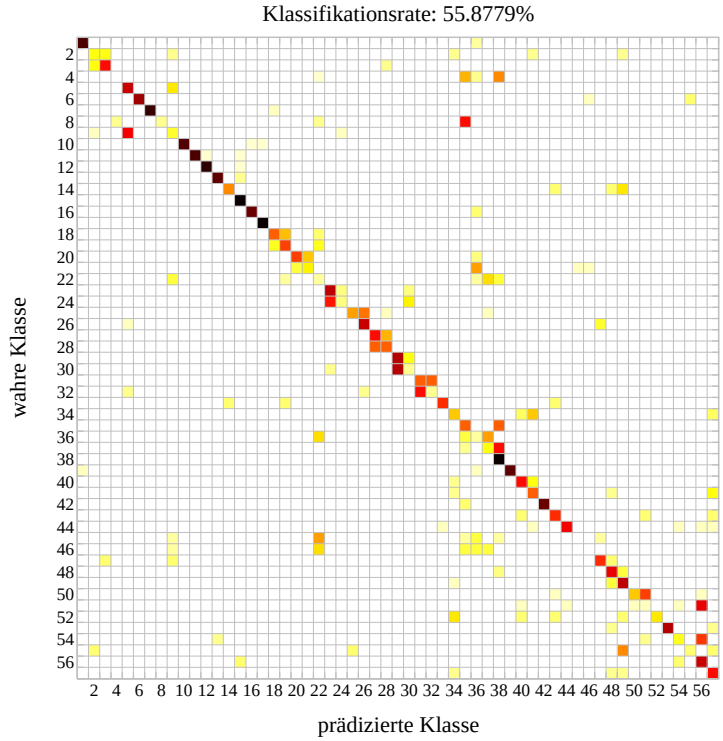
Klassifikationsmatrizen. Die Abbildungen 3.13(a) und 3.13(b) zeigen die Klassifikationsmatrizen des GMM-Data und des KL-EMD Klassifikators. Sieht man sich die Klassifikationsraten an, so scheinen diese Merkmale weniger gut zur Beschreibung der Daten geeignet, wenn man sie anschließend mit GMMs klassifizieren will. Keiner der beiden Klassifikatoren kann anhand physikalischer Merkmale Klassen wie punch (Klassen 12 und 13), kick (Klassen 10 und 11), rotateArms (Klassen 14 und 15) oder die Gehbewegungen (Klassen 17 bis 20) auseinanderhalten. Ebenso verhält es sich für deposit- oder grab-Bewegungen (Klassen 2 und 5 sowie 3 und 6). Insbesondere für die Klassen der Armrotationen liefern diese Merkmale eine zur Unterscheidung ungeeignete Datengrundlage. Nicht dass alle rotateArms-Bewegungen falsch klassifiziert würden, sie werden alle gleich klassifiziert. Hier liegt es tatsächlich vorwiegend an den Merkmalen: Die für diese Bewegungen interessanten Segmente sind der linke Unter- und Oberarm. In allen anderen Segmenten geschieht eher nichts. Was die Bewegungsenergie angeht – und nichts anderes ist kinetische Energie – so hat der Merkmalsverlauf hier keine bewegungsspezifischen Höhen und Tiefen. Von daher gestaltet sich eine korrekte Klassifikation hier schwierig.



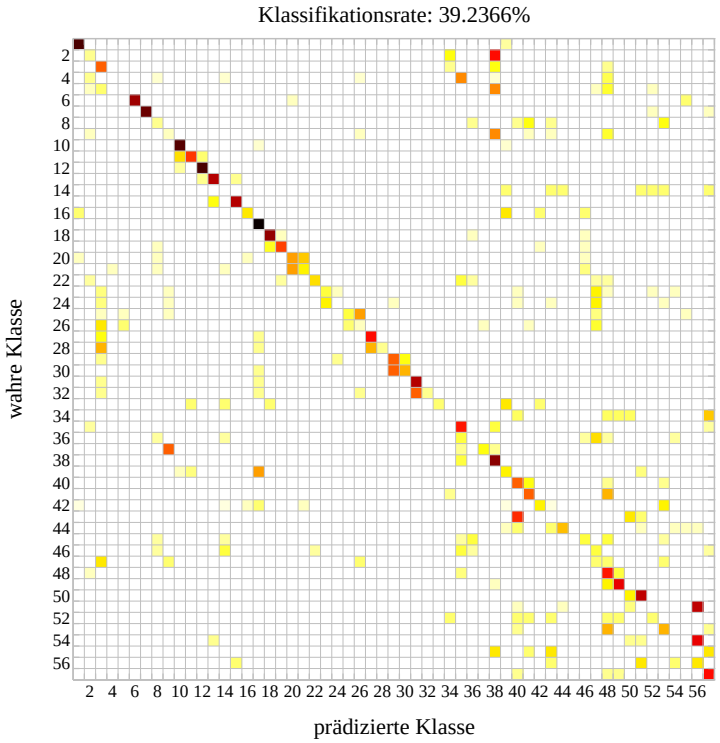
(a) GMM-Data Klassifikator (b) GMM-Centres Klassifikator

Abbildung 3.13: *Klassifikationsmatrix und -rate der GMM-Data und KL-EMD Klassifikatoren auf Grundlage der Datenbank $\mathcal{D}^{20}$ bei Verwendung physikalischer Merkmale.*

Schaut man sich zum Abschluss noch die Klassifikationsmatrizen für die Gesamtdatenbank $\mathcal{D}^{57}$ an – diese sieht man in Abb. 3.14 – so stellt man fest, dass beide vorangegangenen Merkmalssätze für eine GMM-Klassifikation besser geeignet sind als die physikalischen Merkmale. Zwar werden vom GMM-Data Klassifikator immerhin noch ca. dreizehn Klassen relativ gut zugeordnet (relativ gut bedeutet hier, dass ungefähr 75% der Anfragen korrekt klassifiziert wurden), es handelt sich



(a)
GMM-Data
Klassifikator



(b)
KL-EMD
Klassifikator

Abbildung 3.14: *Klassifikationsmatrix sowie -rate für den GMM-Data Klassifikator und den KL-EMD Klassifikator auf Grundlage der Datenbank $\mathcal{D}^{57}$ für physikalische Merkmale.*

dabei aber meist um Klassen, die sehr gut von den anderen unterschieden werden können. Dazu zählen cartwheel-Bewegungen (Radschlag, Klasse 1) und hop-Bewegungen (Hüpfen, Klassen 10, 11, 12), darüber hinaus auch elbowToKnee-Bewegungen (Klassen 6 und 7) und jog[Left,Right]Circle-Bewegungen (Klassen 13 und 14). Die beiden letztgenannten Bewegungspaare werden bei den Gelenkwinkeln weniger gut zugeordnet. Für den KL-EMD Klassifikator könnte man fast von einer chaotischen Klassifikationsmatrix sprechen, und eine Klassifikationsrate von unter 40% bestätigt dieses Bild. Zählt man hier die Klassen die zu 75% und mehr richtig zugeordnet wurden, kommt man auf magere neun. Dazu zählen aber bis auf Klasse 11 alle für den GMM-Data Klassifikator genannten. Vertauschungen treten bei verschiedenen Klassenpaaren auf (rotateArms mit o.g. Problemen), ansonsten scheint die Einordnung eher zufällig.

3.5 Analyse

Im Folgenden werden wir uns einige Aspekte, die zu Fehlklassifikationen führen können, etwas genauer ansehen. Dabei beschränkt sich diese Analyse auf die zwei Merkmalssätze aus Gelenkwinkeln (siehe Tabelle 2.1) und relationalen Merkmalen (siehe Tabelle 2.2) unter Verwendung des GMM-Data und des KL-EMD Klassifikators. Wir nutzen ferner die Klassifikationsergebnisse, die aufgrund der Datenbank $\mathcal{D}^{20}$ ermittelt wurden. Zur Illustration greifen wir uns einige Klassen repräsentativ heraus.

3.5.1 Die Klassen depositFloorR und grabFloorR

Prinzipiell ist bei diesen beiden Klassen einzusehen, dass eine klare Unterscheidung schwierig wird. Erstens sind diese Bewegungen an sich sehr ähnlich (vergleiche Abb. 3.15). Der Unterschied besteht im relevanten Körpersegment (hier wäre das der rechte Arm mitsamt Hand) letztendlich nur in der Reihenfolge der Handlungen. Beim Ablegen (deposit) wird die Hand des Akteurs zuerst um den Gegenstand geschlossen sein und sich öffnen, um ihn abzulegen. Beim Aufheben (grab) ist es aber genau umgekehrt.

(a) eine depositFloor-Bewegung

(b) eine grabFloor-Bewegung

Abbildung 3.15: *Bewegungsabläufe einer (a) depositFloorR- und einer (b) grabFloorR-Bewegung – im Grunde nicht unterscheidbar.*

Selbst für einen (menschlichen) Beobachter ist dieser Unterschied nicht unbedingt erkennbar, je nachdem wie weit er vom Handelnden entfernt ist. Damit sind wir auch schon beim zweiten Problem: Die Auflösung der Daten. Das ist einerseits das Standardskelett, mit dem in dieser Arbeit durchweg gearbeitet wird, andererseits sorgen die gewählten Merkmale (elf Gelenkwinkel) für eine weitere Vergröberung. Das Skelett besteht aus 24 Gelenken (siehe Abb. 2.1 sowie [10]). Die Handpartie wird dabei von zwei Gelenken modelliert, nämlich durch 'wrist' und 'fingers'. Insgesamt sind das unzureichende Voraussetzungen für eine Unterscheidung.

GMM-Data. Der Blick auf die Klassifikationsmatrix bei den *Gelenkwinkeln* ist hier zuerst überraschend. Zwar wird die Klasse grabFloorR erwartungsgemäß häufig mit depositFloorR verwechselt (zu 75%). Umgekehrt aber landet depositFloorR meistens nicht in der grabFloorR-Klasse, sondern wird häufiger mit einem Radschlag (cartwheel) identifiziert.

Die Klassifikationsmatrix spiegelt allerdings nur ein absolutes Bild der Klassifikation wider. Sie unterscheidet nicht, ob die Klassifikation in gewissem Sinne nur zufällig richtig war. Daher ist es sinnvoll, sich auch die entsprechende Prädiktionsmatrix anzusehen. Hier zeigt sich, dass die Entscheidung über die Klassenzuordnung nicht eindeutig ist. Für beide Klassen ist es in sieben Fällen mit bloßem Auge nicht erkennbar, in welche Klasse die jeweilige Bewegung gehört. Allerdings heben sich beide Klassen relativ gut von fast allen anderen ab. Einzige Ausnahme bildet der Radschlag.

Bleibt die Frage, wieso so viele der vom-Boden-Aufheben-Bewegungen als ein Radschlag erkannt werden. Dazu ist anzumerken, dass der Radschlag eine sehr komplexe Bewegung ist und in vielen Gelenkwinkeln ein breit gefächertes Wertespektrum hat. Folglich erhält man als Ergebnis des Trainings ein sehr allgemeines Modell. D.h. die Zentren μ_j liegen weit verstreut über den Datenpunkten, Varianzen σ_j sind tendentiell hoch und die a priori Wahrscheinlichkeiten bzw. Gewichte w_j einander ähnlich. Dies erklärt gleichzeitig, dass das GMM des Radschlages bei beinahe allen Evaluationsdaten hohe Werte in der Klassifikation erzielt.

Schaut man sich zusätzlich Trainings- und Evaluationsdaten an, stellt man fest, dass diejenigen Bewegungen, die zu Trainingszwecken herangezogen wurden, allesamt aus genau einer Aktion mit den Händen nahe dem Boden bestehen, bei denen der Darsteller in die Hocke geht. Bei den Evaluationsdaten besteht der Großteil der Bewegungen nicht aus einem einfachen Ablegen, sie sind etwas komplexer: Der Darsteller bleibt länger mit dem Oberkörper gebeugt, oder er legt mehrere Dinge ab. Noch wichtiger jedoch, er geht selten in die Hocke. Solche Bewegungen unterscheiden sich in ihren Gelenkwinkeln teilweise erheblich von den dem Training zugrundeliegenden Daten und können von dem Klassen-GMM daher nicht gut genug repräsentiert werden. Sie identifizieren sich somit am ehesten mit dem Allgemeinen GMM des Radschlags.

Betrachtet man das Ganze für die *relationalen Merkmale*, so stellt man fest, dass es auch hier vereinzelt zu Verwechslungen mit der Radschlagbewegung kommt, jedoch nicht in derselben Häufigkeit. Zwar ist für diese Klassen das globale Klassifikationsergebnis besser, aber verglichen mit den Gelenkwinkeln ist die Aussagekraft der relationalen Merkmale hier geringer. Das lässt zumindest die Färbung der entsprechenden Bereiche der Prädiktionsmatrix schließen. Helle Bereiche, die auf Verwechslungen hindeuten, sind über die gesamten ersten dreizehn Klassen ausgeprägter als bei den Gelenkwinkeln.

KL-EMD. Zu derartigen Querschlägern wie bei dem GMM-Data Klassifikator, dass eine deposit-Bewegung häufig mit einem Radschlag verwechselt wird, kommt es hier auch bei den *Gelenkwinkeln*

nicht. Das liegt in diesem Falle an der eben so fatalen „Allgemeinheit“ des Radschlagklassenmodells. Sieht man sich Definitionen und Erläuterungen der KL-EMD in Abschnitt 3.3.3 an, so wird klar, dass das auf eine deposit-Bewegung passende Modell mit relativ kleinen Varianzen, dem eines Radschlages nicht ähnelt. Es bleibt daher bei den erwarteten Vertauschungen zwischen deposit und grab, und das aus oben genannten Gründen.

Bei den *relationalen Merkmalen* treten Verwechslungen vorwiegend zwischen den deposit- und grab-Bewegungen untereinander auf. Auffällig an der Prädiktionsmatrix hier ist die zusätzliche blockweise Ähnlichkeit zwischen den [deposit,grab]FloorR- und [deposit,grab]HighR-Bewegungen. Der Grund für diese Ähnlichkeit liegt in der verwendeten Merkmalsmenge. Keines der Merkmale misst eine Bewegung nach oben oder nach unten. Sie messen den Abstand der (linken) Hand zum Wurzelknoten, also dem ‘root’-Gelenk, (HANDLEFTMOVEAPARTRELROOT), den Abstand der (linken) Hand zum Rumpf (HANDLEFTMOVETOFRONTRELTORSO) oder die Geschwindigkeit (HANDLEFTHIGHVEL) der Hand.

3.5.2 Die Klassen punchLFront1Reps und punchLSide1Reps

Auch hier ist eine Verwechslung vorprogrammiert. Man ist mit *Gelenkwinkeln* nicht in der Lage zwischen Bewegungsrichtungen zu unterscheiden. Ob eine Boxbewegung nach vorne geht oder zur Seite, aus den maßgeblich betroffenen Gelenkwinkeln (dem linken Ellenbogen und der linken Schulter) kann man das nicht ablesen. Sie zeigen nur an, dass der Arm ausgestreckt und wieder angewinkelt wird.

Die *relationalen Merkmale* könnten hier eine bessere Chance auf Erfolg haben, denn die Merkmalsmenge enthält zwei Merkmale von denen eines geeignet wäre, ein Seitwärtsboxen von einem Nach-vorne-Boxen zu unterscheiden. Es geht um die Merkmale HANDLEFTMOVEAPARTRELROOT und HANDLEFTTOFRONTRELTORSO. Während für den frontalen Boxhieb beide Merkmale ausschlagen würden, sollte idealerweise für einen Seitwärtsboxer das MOVEAPART-Merkmal zwar ausschlagen, das TOFRONT-Merkmal hingegen nur kleine Werte annehmen. Wird diese Bewegung jedoch in der Realität ausgeführt, ist das nicht immer so. Dies wir in Abb. 3.16 veranschaulicht.

GMM-Data. Verwendet man *Gelenkwinkel*, zeigen die Boxbewegungen auch hohe Ähnlichkeit mit den Trittbewegungen. Dies lässt sich einerseits damit erklären, dass große Teile der Bewegungsabläufe tatsächlich ähnlich sind. Um sie andererseits anhand der Gelenkwinkel gut unterscheiden zu können, sollten bei den Boxbewegungn nur die Merkmale des Arm- und Schulterbereichs ausschlagen. Bei den Trittbewegungen sollten es nur Merkmale den Bein- und Hüftbereich betreffend sein. Da unsere Daten jedoch nicht ideal sind, werden beim Boxen häufig ein oder zwei Schritte gemacht, und beim Treten hängen die Arme nicht teilnahmslos am Körper. Diese Kombination sorgt beim Gelenkwinkel für eine hohe Ähnlichkeit beider Klassenpaare.

Wie oben erwähnt hätten die *relationalen Merkmale* hier besser abschneiden können. Ein wenig besser abgeschnitten haben sie auch, das liest man zum einen aus der Klassifikationsmatrix ab, zum anderen ist es auch in der Prädiktionsmatrix erkennbar. Auch wichtig ist, dass sich im Verhältnis zu den Gelenkwinkeln eine erhebliche Verbesserung der Trennung von Box- und Trittbewegungen eingestellt hat.

KL-EMD. Hier werden die Boxbewegungen – wir betrachten *Gelenkwinkel* – nicht mit den Trittbewegungen verwechselt, dafür aber die Trittbewegungen mit den Laufbewegungen (jog). Obwohl

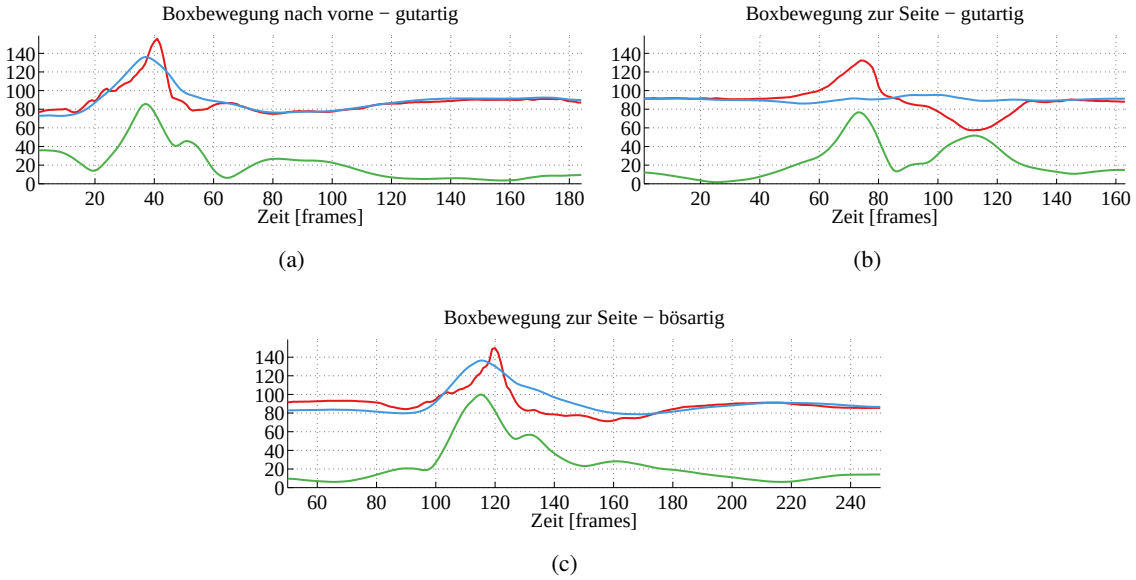


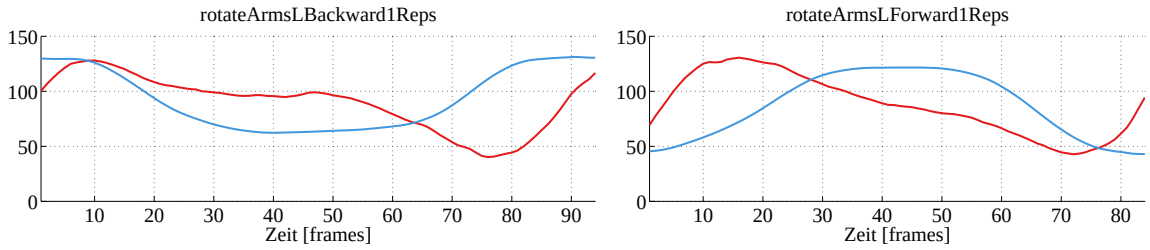
Abbildung 3.16: *Unterschied zwischen einer punchFront-Bewegung (a), einer wie erwartet ausgeführten punchSide-Bewegung (b) sowie einer vom Verständnis her untypisch ausgeführten punchSide-Bewegung (c); dargestellt anhand von drei relationalen Merkmalen, nämlich* HANDLEFTMOVEAPARTRELROOT *'die linke Hand entfernt sich vom Wurzelknoten' (rot),* HANDLEFTTOFRONTRELTORSO *'die linke Hand bewegt sich nach vorne bzgl. Rumpf' (blau) und* HANDLEFTHIGHVEL *'Geschwindigkeit der linken Hand' (grün).*

dieser Abschnitt die Boxbewegungen behandelt, werden wir das kurz erläutern. Der Grund für die Verwechslung liegt in den Merkmalen. Betrachtet man darüber hinaus noch nicht einmal mehr die Daten, die der Bewegung zugrunde liegen, sondern nur noch ein sie repräsentierendes Modell, so ist die Vergröberung der Daten noch höher als mit nur elf Merkmalsvektoren. Beim Laufen werden – wie beim Treten auch – hauptsächlich die Kniegelenke betroffen sein, aber auch Hüfte, Schultern und ggf. die Ellenbogen. Beim Treten sind ebenfalls Knie und Hüfte betroffen, die Arme werden zwecks Balancehalten i.a. auch nicht schlaff herabhängen. Das Bewegungsmodell eines solchen Tritts kann daher Ähnlichkeiten zu den Laufbewegungen aufweisen.

Die *relationalen Merkmale* werfen hier vorwiegend die beiden Boxbewegungen durcheinander. Es kommt außerdem zu einzelnen Vertauschungen mit den deposit- und grab-Bewegungen. Die Bewegungsabläufe – abgesehen von der Geschwindigkeit – können auch hier als ähnlich empfunden werden. Eine bessere Unterscheidung von Box- und Trittbewegungen gemessen an den Gelenkwinkeln setzt sich weiterhin fort.

3.5.3 Die Klassen rotateArmsLForward1Reps und rotateArmsLBackward1Reps

Für die *Gelenkwinkel* kann man das Scheitern einer korrekten Klassifikation hier auf das gewählte Modell schieben und ein wenig auf die Merkmale. Die beiden Bewegungen unterscheiden sich grundlegend nur in der Richtung, in der der Arm rotiert wird. Für den Gelenkwinkel der Schulter heißt das, er nimmt erst zu und dann ab, oder aber andersherum, er nimmt erst ab und dann wieder



(a) (b)

Abbildung 3.17: *Eine rotateArmsLBackward1Reps- (a) und eine rotateArmsLForward1Reps-Bewegung (b), dargestellt mit den Vektoren, die den Daten Zeitinformation geben sollen. Das sind die Merkmale* HANDLEFTMOVEAPARTRELROOT *also 'die linke Hand entfernt sich vom Wurzelknoten' (rot) und* HANDLEFTTOFRONTRELTORSO *d.h. 'die linke Hand bewegt sich bzgl. Rumpf nach vorne' (blau). In beiden Fällen wird der Abstand (zum Wurzelknoten bzw. zum Rumpf) dargestellt.*

zu. Da bei einem GMM alle Daten zeitlich gleichberechtigt sind, bzw. die Zeitinformation komplett verloren geht, hat die Klassifikation wenig Chancen auf Erfolg. Dies gilt für beide Klassifikatoren, und wir werden sie – für den Gelenkwinkel – nicht getrennt betrachten.

Wie schon in Abschnitt 3.4.4 erwähnt, sollten die *relationalen Merkmale* implizit Zeitinformationen in die Daten integrieren. Grundsätzlich sind die beiden Merkmale HANDLEFTMOVEAPARTRELROOT und HANDLEFTTOFRONTRELTORSO zum einen geeignet, zum anderen auch in der Lage dies zu realisieren (siehe Abb. 3.17).

GMM-Data. Vergleicht man das Ergebnis der *relationalen Merkmale* mit dem der Gelenkwinkel, stellt man eine Verbesserung des Klassifikationsergebnis' fest. Zwar ist das in der Prädiktionsmatrix nicht direkt ersichtlich, schaut man jedoch genauer, kann man an den überwiegend richtigen Stellen die Klassenzugehörigkeit erkennen. Von daher scheinen die relationalen Merkmale beim Vergleich der Daten mit dem Klassenmodell erfolgreicher zu sein, als es die Gelenkwinkel waren.

KL-EMD. Hier kann man auch für die *relationalen Merkmale* keine Verbesserung feststellen. Sie schneiden im Allgemeinen ähnlich zu den Gelenkwinkeln ab. Da die für berechneten GMMs ein einzelnes Merkmalspaar nicht unbedingt ins Gewicht fällt, wundert dieses Ergebnis wenig.

3.5.4 Die Klassen der Gehbewegungen

Gemeint sind in diesem Abschnitt die Klassen walk2Steps, walkBackwards sowie walk[Left,Right]-Circle. Bis auf die Rückwärtsgehbewegungen werden diese Klassen untereinander recht oft vertauscht. Grund ist auch hier eine sehr hohe Ähnlichkeit der Daten. Das gilt für *beide Merkmalssätze*.

Für die *relationalen Merkmale* verbessert sich die Klassifikationssituation zum einen, zum anderen verschlechtert sie sich aber auch. Sie verbessert sich dahingehend, dass die (absolute) Unterscheidung zwischen den Klassen 19 und 20 (walkLeftCircle und walkRightCircle) besser gelingt und sich darüber hinaus die Rückwärtsgehbewegung mehr von den übrigen Gehbewegungen abgrenzt. Sie verschlechtert sich, denn die Rückwärtsgehbewegung weist nun höhere Übereinstim-

mung mit den Schleichbewegungen auf (sneak). Darüber hinaus sind sowohl die sneak- als auch die beiden jog-Bewegungen (jogLeftCircle und jogRightCircle) in den Kreis der verwechslungsgefährdeten Klassen eingestiegen. Ein hier ebenfalls interessanter Aspekt sind Klassen, die mit den o.g. Gehbewegungen verwechselt werden könnten, und es aus verschiedenen Gründen nicht dazu kam.

GMM-Data. Dazu zählen – im Falle von *Gelenkwinkeln* – u.a. die jog[Left,Right]Circle Bewegungen. Diese sind im Grunde genommen schnellere Ausführungen der entsprechenden Gehbewegungen. Aber sie werden nicht mit den Gehbewegungen verwechselt, und auch die Gehbewegungen nicht mit den Jogbewegungen. Das liegt vor allem daran, dass verschiedene Gelenkwinkel (etwa Knie, Schulter und Ellenbogen) beim Joggen weit stärker ausschlagen als beim Gehen.

Des Weiteren wäre das Rückwärtsgehen (walkBackwards3StepsRstart) ein Verwechslungskandidat für das Vorwärtsgehen, als seine zeitliche Umkehrung. Jedoch wird auch diese Klasse mit keiner der anderen Gehbewegungen vertauscht. Das liegt vor allem daran, dass die entsprechenden Bewegungen in der genutzten Datenbank keine verdrehte Version des Vorwärtsgehen ist. Dreht man den zeitlichen Verlauf einer Rückwärtsgehbewegung um, so denkt man beim Betrachten eher an ein Schlurfen.

Bei den *relationalen Merkmalen* kann man an dieser Stelle keine Aussagen über Nichtverwechslungen machen. Dafür sind neben den oben genannten Klassen auch noch die Tritt- und Boxbewegungen in der Prädiktionsmatrix im Bereich der Gehbewegungen auffällig hell gefärbt. Der Grund hierfür liegt u.a. in der Merkmalsmenge. Zwei der Merkmale sind stark darauf ausgerichtet, Gehbewegungen zu erkennen. Das sind FOOTLEFTBACKRELLEGRIGHT und FOOTRIGHTBACKRELLEGLEFT. Sowohl bei den Tritt- als auch bei den Boxbewegungen machen die Darsteller häufig ein oder zwei Schritte nach vorne, während und/oder nach der Bewegung, auf der das Hauptaugenmerk liegt. So kommt es zu den Ähnlichkeiten mit reinen Gehbewegungen.

KL-EMD. In der Prädiktionsmatrix sind die Klassen der Gehbewegungen sehr homogen eingefärbt (*Gelenkwinkel*). Eine Unterscheidung gestaltet sich hier auch für das Rückwärtsgehen schwieriger als beim GMM-Data Klassifikator, da die Daten durch eine bloße Modellbetrachtung weiter vergröbert wurden.

Der homogen gefärbte Bereich ist bei den *relationalen Merkmalen* um die Schleichbewegungen erweitert, wird aber unterbrochen von den Rückwärtsgehbewegungen.

3.5.5 Fazit

Gelenkwinkel. Wir fassen noch einmal zusammen, dass primär *drei Punkte* zum Scheitern der Bewegungsklassifikation führen können. Das sind *erstens* die Merkmale. Gelenkwinkel sind beispielsweise ungeeignet, um Bewegungsrichtungen zu beschreiben. Folglich können Bewegungen einzelner Extremitäten wie Arme oder Beine nach vorne oder zur Seite nicht unterschieden werden. *Zum Zweiten* ist das die Auflösung der Daten. Diese beschränkt sich beim Skelett auf 24 Gelenke, bei den Merkmalen sogar auf nur elf. Bewegungen, bei denen die Merkmale charakteristische Anteile nicht treffen (etwa bei den Aufheben- oder Ablegen-Bewegungen), werden in diesem Fall repräsentiert durch weniger aussagekräftige Anteile und aufgrund dieser mit anderen Bewegungen verglichen. Ein *dritter Punkt* ist das Modell selbst. Bei der Reduktion der Daten auf GMMs geht die Reihenfolge der Daten und damit die zugrundeliegende Zeitinformation verloren. Dies führt dazu,

dass verschiedene Bewegungen, die man in zwei verschiedene und gegensätzliche Richtungen (z.B. vorwärts und rückwärts) ausführen kann, für gleich gehalten werden.

Relationale Merkmale. Allgemein gelten hier dieselben Einschränkungen wie für die Gelenkwinkel. Dass die GMMs die zeitliche Abfolge unberücksichtigt lassen ist ein Problem, das auch durch Hinzufügen von Zeitinformation – z.B. mit Geschwindigkeitsmerkmalen – über die Merkmalsvektoren nicht vollständig behoben werden kann. Die Auflösung der Daten ist ein weiteres Erschwernis, das u.a. dafür sorgt, dass nicht alle wichtigen Eigenschaften (Bewegungsrichtung, Geschwindigkeit, etc.) einer Bewegung erfasst werden können. Hinzu kommt, dass einzelne Merkmale letztendlich für die Klassifikation keine ausreichend große Rolle spielen.

3.5.6 Bewertung

Viele der Fehlklassifikationen lassen sich auf Bewegungen zurückführen, die einander im Groben ähneln. Bei Bewegungen wie depositFloorR und grabFloorR ist es im Grunde genommen unmöglich, sie anhand der Bewegungsabläufe des zugrundeliegenden Skeletts unterscheiden zu wollen. Also ist es das Zusammenspiel von zu geringer Auflösung der Daten und sehr ähnlichen Bewegungsabläufen, das zum Scheitern führt.

Ein erheblicher Nachteil der Gelenkwinkel ist, dass sie die Bewegungen zu ungenau beschreiben. Eine Unterscheidung von Bewegungsrichtungen kann man hier von vorneherein ausschließen. Dafür sind sie invariant unter winkelerhaltenden Transformationen wie Skalierung, Rotation und Translation.

Das Vorhaben Zeitinformation anhand von Merkmalskombinationen in die Daten zu kodieren, geht durch Verwenden von relationalen Merkmalen zum Teil auf. Die rotateArm-Bewegungn werden besser klassifiziert, wenngleich eine ausgeprägte Unterscheidung zwischen vorwärts- und rückwärtsrotierendem Arm ausbleibt. Bei den Boxbewegungen wird das angestrebte Ziel nicht erreicht.

Ein weiterer Grund für Fehlklassifikationen liegt im gewählten Modell. Ein GMM modelliert die Daten, die es als Eingabe bekommt, als zeitlich simultan; es lässt jegliche zeitliche Abfolge außer Acht. Vor allem für die rotateArm-Bewegungen wirkt sich das negativ auf die Klassifikation aus.

Abschließend zu diesem Abschnitt listen die nachstehenden Tabellen 3.6 und 3.7 zusammenfassend die Klassifikationsraten für Gelenkwinkel, relationale und physikalische Merkmale der Datenbanken $\mathcal{D}^{20}$ und $\mathcal{D}^{57}$ auf, wenn man neben der favorisierten Klasse auch noch die zweit-, dritt- und viertplazierten Klassen berücksichtigt.

Klassifikator		Top 1	Top 2	Top 3	Top 4
Gelenkwinkel	GMM vs Daten	69.068	93.644	97.034	97.458
	KL und EMD	64.831	88.983	94.492	97.458
relationale	GMM vs Daten	80.932	93.644	95.763	97.458
	KL und EMD	77.542	93.644	96.186	96.610
physikalische	GMM vs Daten	60.169	83.051	88.136	90.678
	KL und EMD	56.356	72.458	78.390	83.051

Tabelle 3.6: *Klassifikationsraten der vier besten Klassen (Top 1 bis Top 4) ausgewertet für GMMs mit Gelenkwinkeln, relationalen und physikalischen Merkmalen für den GMM-Data sowie den KL-EMD Klassifikator. Die hier verwendete Datenbank ist $\mathcal{D}^{20}$.*

Klassifikator		Top 1	Top 2	Top 3	Top 4
Gelenkwinkel	GMM vs Daten	77.252	90.840	93.435	95.420
	KL und EMD	63.664	84.122	89.924	92.824
relationale	GMM vs Daten	81.221	91.450	94.046	95.573
	KL und EMD	68.397	84.580	88.550	91.298
physikalische	GMM vs Daten	55.878	75.115	82.595	85.344
	KL und EMD	39.237	54.809	62.443	66.565

Tabelle 3.7: *Klassifikationsraten der vier besten Klassen (Top 1 bis Top 4) ausgewertet für GMMs mit Gelenkwinkeln, relationalen und physikalischen Merkmalen für den GMM-Data sowie den KL-EMD Klassifikator. Die hier verwendete Datenbank ist* $\mathcal{D}^{57}$.

Dabei fällt ein erheblicher Anstieg der Klassifikationsrate bei Hinzunahme der zweitbesten Bewertung auf. Der Grund hierfür liegt in der Tatsache, dass viele der Klassen genau eine direkt konkurrierende haben, mit der sie vorzugsweise verwechselt werden. In diesen Fällen ist – war es nicht schon die erste – die Zweitwahl beinahe immer die korrekte. Dies gilt beispielsweise für die Klassenpaare depositFloor und grabFloor, jogLeftCircle und jogRightCircle, kickFront und kickSide, rotateArmsForward und rotateArmsBackward sowie walk2StepsRstart und walkRightCircle4StepsRstart und äquivalente.

Kapitel 4

Ein HMM-basiertes Klassifikationsverfahren

4.1 HMM - Hidden Markov Modell

Das Hidden Markov Modell (HMM) kann als ein zweistufiger stochastischer Prozess aufgefasst werden [15]. Es besteht aus einer Markovkette mit einer festzulegenden Anzahl von Zuständen, denen Wahrscheinlichkeiten bzw. Wahrscheinlichkeitsdichten zugeordnet sind. Diese Zustandsketten ermöglichen die Modellierung des zeitlichen Verlaufs der (Verteilung der) Merkmalsvektoren. Geschwindigkeitsunterschiede und -variationen lassen sich durch Wiederholen und Überspringen von Zuständen beschreiben. Das Modell trägt das zusätzliche Namensattribut verborgen („hidden“), da die Zustände selbst nicht direkt aus den Daten hervorgehen. Um auf die Zustände schließen zu können, müssen die beobachtbaren Daten (Observationsdaten) herangezogen werden. Ein einzelner Zustand entspricht einer Datenwolke im Merkmalsraum.

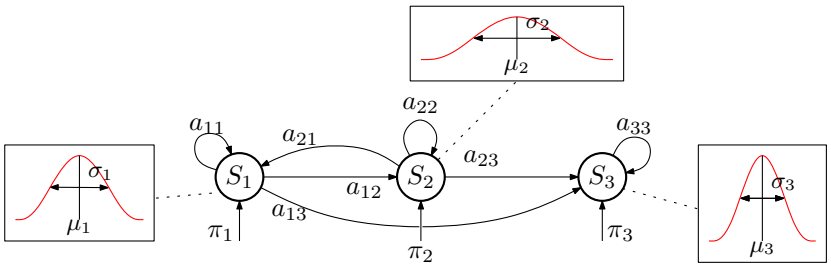


Abbildung 4.1: *Die einem Hidden Markov Modell unterliegende Markovkette mit drei Zuständen. Hier wird jeder Zustand eindimensional durch eine Gaußglocke repräsentiert, parametrisiert über einen Mittelwert μ_i und eine Varianz σ_i, $i = 1, 2, 3$.*

Etwas ausführlicher lässt sich ein HMM auch wie folgt charakterisieren: Es ist ein endlicher Zustandsautomat mit einer gewissen Anzahl N von Zuständen, $\mathcal{S} = \{S_1, \ldots, S_N\}$. Tatsächlich handelt es sich nicht um einen einfachen endlichen Zustandsautomaten, sondern – wie schon zu Beginn erwähnt – um eine Markovkette, genauer gesagt um eine Markovkette erster Ordnung. Diese verkörpert eine wichtige Annahme über die Wahrscheinlichkeiten. In einer Markovkette erster Ordnung hängt die Wahrscheinlichkeit des nächsten Zustandes nur vom aktuellen ab. Man nennt ein

solches System auch gedächtnislos, da Ereignisse aus der Vergangenheit nicht eingehen:

$$\begin{aligned} &P(s_t = S_j \mid s_{t-1} = S_i, s_{t-2} = S_k, \ldots, s_1 = S_l) \\ &= P(s_t = S_j \mid s_{t-1} = S_i). \end{aligned}$$

Hierbei bezeichnet s_t denjenigen Zustand, der zum Zeitpunkt t angenommen wird. Man könnte also etwa für eine Beobachtungssequenz $x = (x_1, x_2, \ldots, x_T)$ von einer dazugehörigen Zustandssequenz $s = (s_1, s_2, \ldots, s_T)$ sprechen, wobei jedes s_t, $1 \leq t \leq T$ einem S_i aus $\mathcal{S}$, $1 \leq i \leq N$ entspricht. Somit sind die Übergangswahrscheinlichkeiten von einem Zustand S_i in einen Zustand S_j gegeben durch

$$a_{ij}(t) = P(s_t = S_j \mid s_{t-1} = S_i)$$

und können in einer Zustandstandsübergangsmatrix zusammengefasst werden

$$(a_{ij}(t))_{\substack{1 \leq i \leq N \\ 1 \leq j \leq N}} = \begin{pmatrix} a_{11}(t) & \ldots & a_{1N}(t) \\ \vdots & \ddots & \ldots \\ a_{N1}(t) & \ldots & a_{NN}(t) \end{pmatrix}.$$

Die Markovkette ist zudem auch homogen, d.h. sie ist unabhängig vom Zeitpunkt t. Es gilt also $a_{ij} = a_{ij}(t)$ für alle t und die Zustandsübergangsmatrix hat die Form

$$A = (a_{ij})_{\substack{1 \leq i \leq N \\ 1 \leq j \leq N}}, \tag{4.1.1}$$

wobei jeder Eintrag von A größer oder gleich null ist, $a_{ij} \geq 0$, und sie sich zeilenweise zu eins aufsummieren, $\sum_{j=1}^{N} a_{ij} = 1$.

Neben der Zustandsübergangsmatrix wird für jeden Zustand eine Anfangswahrscheinlichkeit definiert und in einem Vektor $\pi = \{\pi_i\}_{1 \leq i \leq N}$ zusammengefasst. Die Einträge sind gegeben durch

$$\pi_i = P(s_1 = S_i), \qquad 1 \leq i \leq N,$$

und entsprechen der Wahrscheinlichkeit, dass der i-te Zustand Einstiegspunkt in das Modell ist. Es gilt auch hier, dass jeder Eintrag des Vektors nicht negativ ist und die Summe der Einträge eins ergeben. Ein mit einem nullwertigen Eintrag versehener Zustand kann daher niemals Startzustand sein.

Bei den hier verwendeten *kontinuierlichen* HMMs wird jeder Zustand $S_j \in \mathcal{S}$ durch eine Wahrscheinlichkeitsdichtefunktion b_j aus M Komponenten beschrieben. Häufig – und auch bei uns – kommen hierbei multivariate Gaußverteilungen zum Einsatz. Die Beobachtungswahrscheinlichkeit eines d-dimensionalen Vektors x im j-ten Zustand ergibt sich zu

$$b_j(x) = \sum_{m=1}^{M} w_{jm} p_{jm}(x) \qquad 1 \leq j \leq N$$

und entspricht so einem GMM für jeden Zustand (siehe Abschnitt 3.1). Dabei ist w_{jm} Gewicht des Zustands S_j und Verteilung m und $p_{jm}(x)$ Basisfunktion. Diese ist für die Gaußverteilung parametrisierbar über einen $d \times 1$ Mittelwertsvektor μ_{jm} sowie eine $d \times d$ Kovarianzmatrix Σ_{jm}, ganz in Analogie zu Gleichung 3.1.2:

$$p_{jm}(x) = \frac{1}{(2\pi)^{d/2} |\Sigma_{jm}|^{1/2}} \exp\left(-\frac{1}{2} (x - \mu_{jm})^T (\Sigma_{jm})^{-1} (x - \mu_{jm}) \right).$$

Wir werden uns hier – wie auch zuvor bei den GMMs – auf sphärische Kovarianz beschränken, d.h. $\Sigma_{jm} = \sigma_{jm}^2 \mathbb{1}$, so dass die Verteilung $p_{jm}(x)$ für jeden Zustand $1 \leq j \leq N$ gegeben ist durch

$$p_{jm}(x) = \frac{1}{(2\pi\sigma_{jm}^2)^{d/2}} \exp\left(-\frac{1}{2}\frac{\|x-\mu_{jm}\|^2}{\sigma_{jm}^2}\right).$$

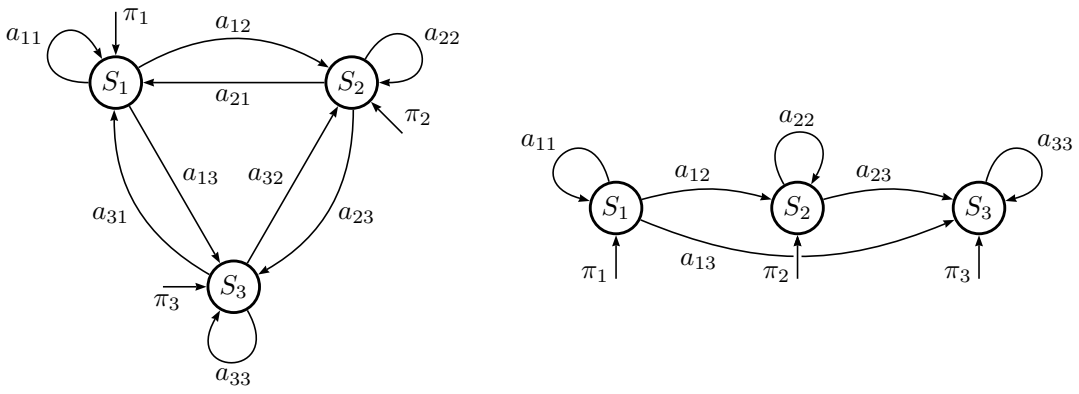
Außerdem machen wir eine weitere Einschränkung. Um die Ergebnisse von HMM- und GMM-Klassifikation später direkt vergleichen zu können, vor allem auf die Fragestellung hin, was das Hinzufügen einer gewissen Zeitinformation bewirkt, werden wir jeden Zustand mit nur einem einzelnen Gaußkern ausstatten (d.h. $M = 1$) und in den Formeln – der Übersicht halber – den zweiten Laufindex weglassen:

$$b_j(x) = w_j p_j(x) = p_j(x) \qquad 1 \leq j \leq N.$$

Mit diesen Definitionen lässt sich ein HMM auch kompakt schreiben als $\lambda = (A, B, \pi)$. Darin ist A die Matrix der Zustandsübergangswahrscheinlichkeiten, B enthält die Parameter der Wahrscheinlichkeitsverteilung der Zustände, $B = (b_1, \ldots, b_N)$, (in unserem Fall also Mittelwerte und Varianzen) und π ist der Vektor der Anfangswahrscheinlichkeiten.

4.1.1 Typen von HMMs

Die Übergangswahrscheinlichkeiten des in Abb. 4.1 dargestellten HMM sind nicht für alle Zustandspaare positiv. Ein Übergang von S_3 zurück in S_2 kann nicht stattfinden. Seine Wahrscheinlichkeit ist null ($a_{32} = 0$). Es gibt jedoch auch Modelle, in denen die Übergangswahrscheinlichkeiten für ein beliebiges Zustandspaar positiv (und nicht null) sind. Ein solches HMM nennt man auch *vollständig verbunden* oder *ergodisch*. In diesem Fall ist die Zustandsübergangsmatrix voll besetzt, siehe Abb. 4.2(a).



(a) ergodisches HMM (b) Bakis-HMM

Abbildung 4.2: *Zwei HMMs mit drei Zuständen, dargestellt als Zustandsübergangsgraph. Informationen über die Ausgabeverteilungen sind nicht enthalten, sondern nur die erlaubten Übergänge der jeweils unterliegenden Markovkette. Nicht angezeigte Übergangswahrscheinlichkeiten haben den Wert null.*

In einem weiteren erwähnenswerten Spezialfall, dem sogenannten *links-rechts-HMM* (alternativ auch als *Bakis*-HMM bezeichnet) sind nur Selbstübergänge – also Verweilen in einem Zustand – und Übergänge in einen noch nicht besuchten Zustand erlaubt; Rücksprünge sind nicht gestattet, siehe Abb. 4.2(b). Die Bezeichnung links-rechts-Modell rührt daher, dass man die Zustände so anordnen kann, dass Übergänge in einen anderen Zustand stets von links nach rechts stattfinden. Die dazugehörige Matrix der Übergangswahrscheinlichkeiten entspricht dann einer oberen Dreiecksmatrix (mit Diagonaleinträgen für die Selbstübergänge).

4.1.2 Problemstellungen

Nach Rabiner [15] ergeben sich für Hidden Markov Modelle drei wesentliche Problemstellungen:

(1) (Evaluation- / Klassifikation) Hat man ein festes HMM $\lambda = (A, B, \pi)$ sowie eine bestimmte Sequenz von Beobachtungen $x = (x_1, x_2, \ldots, x_T)$ gegeben, so möchte man die Wahrscheinlichkeit bestimmen, dass x von λ produziert wurde, also die Likelihood $\mathrm{P}(x)$ berechnen.

(2) (Dekodierung) In diesem Szenario sucht man die in Bezug auf die Beobachtungssequenz x der Länge T, $x = (x_1, x_2, \ldots, x_T)$, wahrscheinlichste (optimalste) Folge der (versteckten) Zustände $(s_1, s_2, \ldots, s_T)$, bei vorgegebenem Modell λ, d.h. maximiere $\mathrm{P}(x|s_1, \ldots, s_T)$.

(3) (Lernen) Hier geht es um die Anpassung der Modellparameter A, B und π von λ, um für eine Beobachtungsfolge $x = (x_1, x_2, \ldots, x_T)$ die Wahrscheinlichkeit $\mathrm{P}(x|\lambda)$ zu maximieren. Dazu sind i.a. auch die Anzahl N der Zustände und M der Mixturkomponenten pro Zustand gegeben. Tatsächlich sollte man von Beobachtungsfolg*en* sprechen, denn zum Lernen eines HMM wird nicht nur eine einzelne Sequenz herangezogen, sondern mehrere.

Für die Klassifikation von Signalen sind vor allem Problemstellungen 1 und 3 relevant. Wir werden daher auf die Lösung dieser beiden später etwas genauer eingehen (in den Abschnitten 4.3.1 und 4.1.3). Zur Dekodierung einer Beobachtungsfolge verwendet man den Viterbi-Algorithmus. Da wir ihn hier nicht weiter erläutern werden, verweisen wir auf die Literatur [15].

4.1.3 Berechnung der Parameter eines HMM

Für die Berechnung eines die Wahrscheinlichkeit der Beobachtungen maximierenden HMM ist keine analytische Lösung bekannt. Stattdessen beschränkt man sich auf die Suche eines lokalen Maximums und verwendet dazu iterative Methoden wie etwa Gradiententechniken oder EM-Algorithmen (Expectation Maximation). Bei uns kommt ein Spezialfall eines EM-Algorithmus zur Anwendung, der gleichzeitig auch ein Standardverfahren zur Berechnung lokal optimaler Parameter eines HMM ist, der Baum-Welch-Algorithmus.

Unerheblich welchen iterativen Weg man wählt, es ist stets ein Anfangsmodell – etwa $\lambda^{(0)}$ – zur Initialisierung notwendig. Bevor wir uns also in medias res begeben, erläutern wir kurz die Berechnung eines Anfangsmodells $\lambda^{(0)} = (A^{(0)}, B^{(0)}, \pi^{(0)})$.

Initialisierung. Zur Berechnung der räumlichen Lage der Zustände erstellen wir ein GMM für alle Trainingsdaten der Klasse und erhalten dadurch Clustermittelpunkte ${\mu_i}^{(0)}$ sowie Varianzen ${\sigma_i}^{(0)}$, $1 \leq i \leq N$, für die Zustände. Auf dieser Grundlage kann man die Trainingsdaten den Zuständen

zuordnen und Anfangswerte für Zustandsübergangsmatrix $A^{(0)}$ und Anfangswahrscheinlichkeiten $\pi^{(0)}$ berechnen. Dies geschieht durch Abzählen:

$$a_{ij}{}^{(0)} = \frac{\text{Anzahl Übergänge von } S_i \text{ nach } S_j}{\text{Anzahl Übergänge}}$$

$$\pi_i{}^{(0)} = \frac{\text{Anzahl Trainingssequenzen, die zum Zeitpunkt } t = 1 \text{ im Zustand } S_i \text{ sind}}{\text{Anzahl Trainingssequenzen}}$$

Dabei verwenden wir zur Berechnung von $\pi_i{}^{(0)}$ jedoch nicht nur den ersten Frame jeder Trainingssequenz, sondern die ersten sechs.

Baum-Welch Algorithmus. Mit dem so erlangten HMM wird der Baum-Welch-Algorithmus initialisiert. Er liefert dann ein an die Beobachtungssequenzen angepasstes und im statistischen Sinne (lokal) optimales Modell. Zu diesem Zwecke definieren wir eine Variable

$$\xi_t(i,j) = \mathrm{P}\left(q_t = S_i, q_{t+1} = S_j | X\right),$$

die die Wahrscheinlichkeit angibt, dass man beim Übergang von Zeitpunkt t auf $t+1$ vom Zustand S_i in S_j wechselt. Dann ist die erwartete Anzahl von Übergängen von S_i nach S_j die Summe über alle Zeitpunkte t von ξ. Um eine Schätzung $\overline{a_{ij}}$ von a_{ij} zu erhalten, muss diese Summe noch durch die erwartete Anzahl von Übergängen, die von S_i ausgehen, dividiert werden:

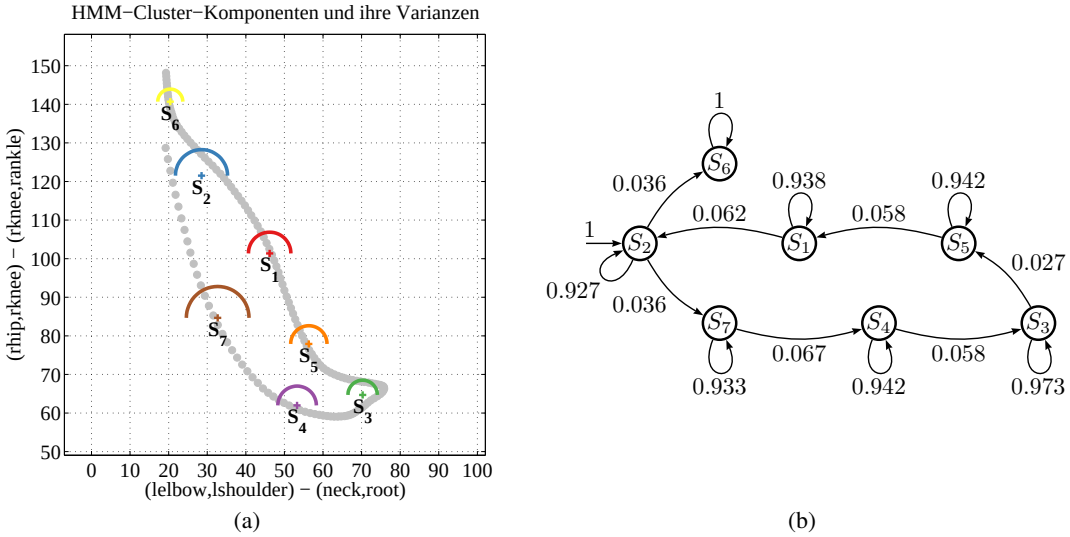
$$\begin{aligned}\overline{a_{ij}} &= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)} \\ &= \frac{\text{erwartete Anzahl von Übergängen von } S_i \text{ nach } S_j}{\text{erwartete Anzahl von Übergängen aus } S_i \text{ heraus}}\end{aligned}$$

Für $\overline{\pi_i}$, sowie $\overline{\sigma_j}$ und $\overline{\mu_j}$ gibt es ähnliche Formeln. Dazu sei aber auf die Literatur verwiesen [15].

Der Algorithmus passt die Parameter schrittweise an: Zur Ermittlung eines neuen Modells werden die Parameter des aktuellen Modells herangezogen, um die rechte Seite der Formeln zu berechnen. Anschließend stellt die linke Seite die Parameter des neuen Modells dar. Dieses Vorgehen wird iteriert bis sich Konvergenz einstellt. Man kann zeigen, dass diese Parameterschätzungen tatsächlich eine Verbesserung darstellen, wenn nicht schon ein lokales Maximum erreicht ist.

Zur effektiven Berechnung der neuen Modellparameter können sogenannte Vorwärts- und Rückwärts-Variablen verwendet werden. Aus diesem Grund nennt man obiges Vorgehen zur Parameterbestimmung auch Vorwärts-Rückwärts-Algorithmus. Da wir sie zu jetzigen Zeitpunkt noch nicht explizit brauchen, führen wir diese beiden Variablen etwas später in Abschnitt 4.3.1 ein.

Ein Beispiel. Nach Extraktion der Merkmalsvektoren X_D,eines Motion Capture Datenstroms D, werden die so gewonnenen Beobachtungsdaten als Grundlage zur Berechnung der initialen und endgültigen Modellparameter verwendet. Der Baum-Welch-Algorithmus liefert zum einen für jeden Zustand S_i ein GMM mit Clustermittelpunkt μ_j sowie Varianz σ_j, siehe Abb. 4.3(a). Zum anderen erhält man ebenfalls die Zustandsübergangsmatrix A, deren Struktur in Abb. 4.3(b) visualisiert ist.



(a) (b)

Abbildung 4.3: *(a) zeigt den Verlauf der Merkmalsvektoren für $f = 2$ Merkmale einer Bewegung („elbow-to-knee“) sowie die räumliche Lage der Zustände des HMM, repräsentiert durch den Mittelwert μ_j (Kreuze) und die Varianz σ_j (Halbkreise) ihrer GMM-Komponente. (b) zeigt die unterliegende Struktur des in (a) dargestellten Klassen-HMM.*

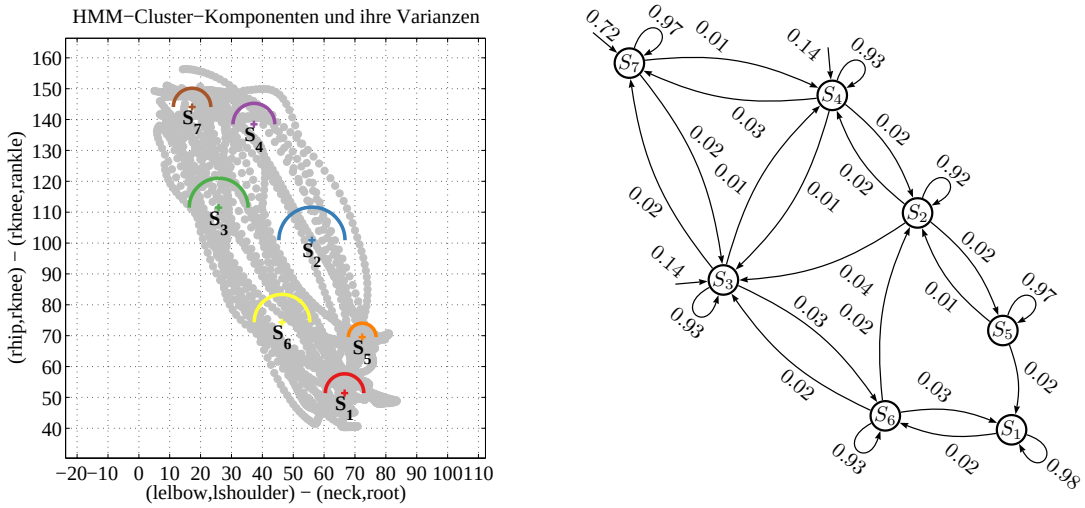
4.2 Training

In Analogie zur Repräsentation einer Klasse als GMM in Abschnitt 3.2, soll auf der Grundlage von Trainingsdaten einer Bewegungsklasse $\mathcal{C}$ ein HMM als Klassenmodell geschätzt werden. In diesem Fall ist das Klassenmodell ein Tupel (A, B, π), wobei A die Struktur der zugrundeliegenden Markovkette enthält, π die Verteilung der Startzustände und B die Parameter der zustandsweisen Gaußverteilungen, über die die Ausgabewahrscheinlichkeiten berechnet werden. Das tatsächliche Training der Parameter erfolgt mittels des in Abschnitt 4.1.3 erläuterten Baum-Welch Algorithmus. Das Ergebnis des Trainings einer Klasse auf Grundlage zweier Merkmale wird in Abb. 4.4 veranschaulicht.

4.3 Klassifikation

Nun geht es wieder darum, eine unbekannte Motion Capture Anfrage Q in eine Datenbank $\mathcal{D}$ einzuordnen, indem man eine Klasse in der Datenbank findet, die zu der Anfrage ähnliche Bewegungen enthält, d.h. deren Klassenmodell die Anfrage repräsentieren könnte.

Zu diesem Zwecke werden wir die Anfrage Q aufgrund ihrer Merkmalsvektoren mit den gelernten Klassenmodellen aller Klassen vergleichen. Ergebnis der Klassifikation ist ein Vektor S der Länge γ, der für jede Klasse $\mathcal{C}$ eine Bewertung $S_\mathcal{C}$ enthält. $S_\mathcal{C}$ beurteilt Klassifikationsgüte, falls Q der Klasse $\mathcal{C}$ zugeordnet würde.



(a) (b)

Abbildung 4.4: *(a) zeigt den Verlauf der Merkmalsvektoren für $f = 2$ Merkmale einer Klasse („elbow-to-knee“) sowie die räumliche Lage der Zustände des HMM, repräsentiert durch den Mittelwert μ_j (Kreuze) und die Varianz σ_j (Halbkreise) ihrer GMM-Komponente. (b) zeigt die unterliegende Struktur des Klassen-HMM. Der Übersicht halber wurden Übergänge, deren Wahrscheinlichkeit kleiner als $0,01$ sind weggelassen. Wahrscheinlichkeiten größer null haben noch die Übergänge von S_1 nach S_5, von S_5 nach S_6 sowie von S_6 nach S_5.*

4.3.1 Klassifikation anhand von Merkmalsvektoren

Die Klassifikation einer Anfrage führt uns gleichzeitig zur Lösung des in Abschnitt 4.1.2 vorgestellten Evaluations- und Klassifikationsproblems (vergleiche Problemstellung 1, Seite 44). Es ging dabei um die Berechnung der Wahrscheinlichkeit $\mathrm{P}(x)$, dass eine Beobachtungssequenz $x = (x_1, x_2, \ldots, x_T)$ von einem vorgegebenen Modell λ generiert wurde. Wir werden daher zuerst das allgemeine Vorgehen zur Lösung erläutern, bevor wir es auf unsere Daten anwenden.

Naive Herangehensweise. Ein möglicher Weg zur Bestimmung der genauen Wahrscheinlichkeit $\mathrm{P}(x)$ für eine Beobachtung x geht über die Aufzählung aller denkbaren Zustandssequenzen der Länge T über N Zuständen. Für eine feste Zustandsfolge $S = (s_1, s_2, \ldots, s_T)$ ist die gemeinsame Wahrscheinlichkeit von x und S

$$\mathrm{P}(x, S) = \mathrm{P}(x|S)\,\mathrm{P}(S)\,.$$

Da man annimmt, dass die Beobachtungen statistisch unabhängig sind und aufgrund der Eigenschaften eines HMM (Homogenität, Markovkette erster Ordnung) gilt

$$\mathrm{P}(x|S) = \prod_{t=1}^{T} \mathrm{P}(x_t|s_t) = \prod_{t=1}^{T} b_{s_t}(x_t) \qquad \text{sowie}$$

$$\mathrm{P}(S) = \prod_{t=1}^{T} \mathrm{P}(s_t|s_{t-1}) = \pi_{s_1} \prod_{t=2}^{T} a_{s_{t-1}s_t},$$

zusammen also

$$\mathrm{P}(x, S) = \pi_{s_1} b_{s_1}(x_1) \prod_{t=2}^{T} a_{s_{t-1}s_t} b_{s_t}(x_t).$$

Allerdings haben wir uns in diesem Szenario auf eine spezielle Folge von Zuständen beschränkt. Uns aber interessiert die Likelihood von x unter Berücksichtigung aller möglichen Zustandsfolgen. Daher muss obiges Produkt für alle möglichen Zustandsfolgen S aufsummiert werden:

$$\mathrm{P}(x) = \sum_{\text{alle } S} \mathrm{P}(x, S)$$

Aus mathematischer Sicht interessant – liefert diese Gleichung doch eine implizite Darstellung für $\mathrm{P}(x)$ – kommt sie aus programmiertechnischer Sicht aufgrund zu hoher Komplexität nicht in Frage. Denn die Formel setzt sich zusammen aus N^T vielen Summanden mit je $2T - 1$ vielen Multiplikationen.

Der Vorwärts-Rückwärts-Algorithmus. Anstatt einen Algorithmus mit exponentieller Laufzeit zu verwenden, greift man auf einen auf dynamische Programmierung basierenden Ansatz zurück. Es handelt sich um den *Vorwärts-Rückwärts-Algorithmus*. Tatsächlich benötigt man zur Berechnung von $\mathrm{P}(x)$ entweder die Vorwärts- oder die Rückwärts-Komponente; hier werden wir die Vorwärts-Komponente benutzen. Zu diesem Zweck definiert man eine Vorwärtsvariable $\alpha_t(i)$, die die Wahrscheinlichkeit angibt, im t-ten Zeitschritt in den Zustand S_i überzugehen, gleichgültig wie der vorangegangene Zustandspfad ausgesehen haben mag,

$$\alpha_t(i) = \mathrm{P}(x_1, x_2, \ldots, x_t, q_t = S_i).$$

Man kann nachrechnen (siehe etwa [2]), dass $\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(x_{t+1})$, für $1 \leq t \leq T-1$ und $1 \leq j \leq N$. Initialisiert man $\alpha_1(i) = \pi_i b_i(x_1)$ für $1 \leq i \leq N$, so lassen sich die $\alpha_T(i)$, $1 \leq i \leq N$ zu $\mathrm{P}(x)$ aufsummieren:

$$\mathrm{P}(x) = \sum_{i=1}^{N} \alpha_T(i).$$

Der Vollständigkeit halber erwähnen wir, dass sich in analoger Weise auch die sogenannte Rückwärts-Variable $\beta_t(i)$ definieren lässt. Es ist

$$\beta_t(i) = \mathrm{P}(x_{t+1}, x_{t+2}, \ldots, x_T | s_t = S_i).$$

Anwendung. Auf Grundlage der aus einer Anfrage Q der Länge T extrahierten Merkmalsvektoren $X_Q = (x_1, x_2, \ldots, x_T)$ ist die Bewertung für HMMs durch $\mathrm{P}(X_Q|\mathcal{C})$ gegeben. Da die zur Realisierung eingesetzte Implementation [11] Log-Likelihoods verwendet, ergibt sich die Scorefunktion für HMMs nach Gewichtung mit der Länge der der Anfrage zu

$$S_{\mathcal{C}}^{\mathrm{hmm}} = \frac{1}{T} \log \mathrm{P}(X_Q|\mathcal{C})$$

Sie wird mittels Vorwärtsalgorithmus effizient berechnet. Dabei stellt $\mathrm{P}(X_Q|\mathcal{C})$ die Abhängigkeit von der Bewegungsklasse $\mathcal{C}$ und dem damit verbundenen Klassenmodell dar. Das Ergebnis der Auswertung von $S_{\mathcal{C}}^{\mathrm{hmm}}$ auf 20 Klassen bezogen auf eine elbow-to-knee-Bewegung ist in Abb. 4.5 dargestellt. Wir werden diesen Klassifikator im weiteren Verlauf der Arbeit auch als HMM-Data Klassifikator bezeichnen.

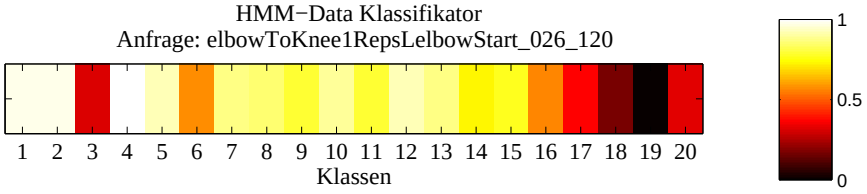


Abbildung 4.5: *Ergebnis der Anwendung von $S_{\mathcal{C}}^{\text{hmm}}$ (nach Normierung auf [0,1]) auf die Klassen der Datenbank $\mathcal{D}^{20}$ für eine elbow-to-knee-Bewegung. Für Training und Klassifikation wurden $f = 2$ Merkmale verwendet (rechtes Knie und linke Schulter bzgl. Körpersenkrechter). Die höchsten Werte werden (bei einer rein visuellen Beurteilung) für Klasse 1 (cartwheel), 2 (depositFloor) oder 4 (elbowToKnee) angenommen.*

4.3.2 Vorhersage einer Klasse

Wie auch schon bei den GMMs (Abschnitt 3.3.4) wird die Zugehörigkeit einer Anfrage Q zu einer Klasse $\tilde{\mathcal{C}} \in \mathcal{D} = (\mathcal{C}_1 \dot{\cup} \ldots \dot{\cup} \mathcal{C}_\gamma)$ über eine Art Maximum Likelihood Schätzung getroffen:

$$\tilde{\mathcal{C}} = \arg \max_{\mathcal{C} \in \{\mathcal{C}_1, \ldots, \mathcal{C}_\gamma\}} S_{\mathcal{C}}(X_Q).$$

Unter diesem Aspekt würde man die in Abbildung 4.5 gezeigte Anfrage richtigerweise der Klasse 4 (elbowToKnee) zuweisen.

4.4 Ergebnisse

Zur Beurteilung der Klassifikation von Bewegungsdaten mit Hilfe von HMMs greifen wir wieder zurück auf die in den Abschnitten 3.4.1 und 3.4.2 Prädiktions- und Klassifikationsmatrizen. Auch hier verwenden wir zur Demonstration die kleinere Datenbank $\mathcal{D}^{20}$, eingeteilt in disjunkte Evaluations- und Trainingsdatenbank ($\mathcal{D}_E^{20}$ und $\mathcal{D}_T^{20}$). Als Merkmal verwenden wir die in Tabelle 2.1 aufgelisteten $f = 11$ Gelenkwinkel.

Anmerkung. Aus technischen Gründen haben wir uns für ein ergodisches HMM als Klassenmodell entschieden und nicht für ein links-rechts HMM. Zum einen kam es zu Programmabbrüchen beim Training der Klassenmodelle, zum anderen vor, dass keine Anfrage aus einer Klasse einen gültigen Pfad durch ihr Klassenmodell und/oder das einer anderen Klasse fand. Denn die Merkmalsvektoren der jeweilige Anfrage konnte ab einem bestimmten Zeitpunkt keinem der Zustände mehr zugeordnet werden. In einer zeitweisen Behelfslösung wurde dem Modell im Falle einer solchen Anfrage ein festgelegter, minimaler Score zugewiesen – schließlich war es für die Anfrage nicht repräsentativ. Dieses Vorgehen führte jedoch häufig zu Prädiktionsmatrizen, die von dem festgelegten Score dominiert wurden und faktisch keine Aussagekraft mehr hatten.

Durch die Verwendung ergodischer HMMs nutzen wir einen entscheidenden Vorteil des links-rechts Modells nicht: die Zeitinformation, die es aufgrund der möglichen Zustandsübergänge induziert. In einem ergodischen Modell sind Zustandsübergänge je nach Konnektivität willkürlich. Im Klassenmodell in Abb. 4.4 (b) Seite 47 wäre es beispielsweise möglich, in Zustand S_7 zu starten,

von dort aus in S_3 überzugehen und wieder zurück nach S_7. Alle anderen Zustände blieben unberücksichtigt, und der Score eines solchen Weges ist eher hoch. In einem links-rechts Modell hätte sich die Anfrage durch das Modell „quälen“ müssen. Der Score fiele entsprechend gering aus.

4.4.1 Prädiktionsmatrix (Prediction Matrix)

Die folgenden Tabellen 4.1 und 4.2 vergleichen die Berechnungszeiten des HMM-Data Klassifikators mit zwei der GMM-Klassifikatoren jeweils für die kleine Datenbank $\mathcal{D}^{20}$ und die große $\mathcal{D}^{57}$.

Klassifikator	Zeit [sec]
HMM vs Daten	219.766
GMM vs Daten	14.561
GMM - KL und EMD	25.537

Tabelle 4.1: *Zeit zum Erstellen einer Prädiktionsmatrix für 236 Evaluationsdaten in 20 Klassen der Datenbank $\mathcal{D}^{20}$.*

Klassifikator	Zeit [sec]
HMM vs Daten	1721.054
GMM vs Daten	113.974
GMM - KL und EMD	206.146

Tabelle 4.2: *Zeit zum Erstellen einer Prädiktionsmatrix für 655 Evaluationsdaten in 57 Klassen der Datenbank $\mathcal{D}^{57}$.*

Die aus dieser Klassifikation für den HMM-Data Klassifikator hervorgegangene relative Prädiktionsmatrix ist in Abb. 4.6 oben dargestellt. Um einen direkten Vergleich zu der GMM-Klassifikation zu haben, ist darunter die relative Prädiktionsmatrix des GMM-Data Klassifikators abgebildet. Zur Erinnerung: Es ist sinnvoll relative Prädiktionsmatrizen zu verwenden, da die Wahl einer geeigneten Klasse spaltenweise erfolgt, und die relative Prädiktionsmatrix aus der absoluten durch spaltenweise Normierung mit dem jeweiligen Spaltenmaximum hervorgeht.

Schaut man sich die Prädiktionsmatrix des HMM-Data Klassifikators genau an, so sieht man eine hohe Werteähnlichkeit z.B. für die Klassen 3 und 6. Die betroffenen Klassen enthalten Bewegungen des Typs depositHigh und grabHigh (‘oben hinlegen’, ‘von oben nehmen’, wobei ‘oben’ sich auf die Höhe des Zielobjekts bezieht). Diese Bewegungen sind im Ablauf sehr ähnlich und können aufgrund der Skelettauflösung und der Merkmalswahl nur schlecht unterschieden werden. Vergleichbar verhält es sich mit Klassen wie kickLFront (Klasse 10) und kickLSide (Klasse 11) (‘frontaler Tritt mit dem linken Fuß’ und ‘Tritt zur Seite mit dem linken Fuß’). Zwar schlägt in diesem Fall nicht die Feinheit des Skeletts zu, wohl aber die Merkmale, die Richtungen nicht berücksichtigen. Des Weiteren entsprechen sich die Klassen 17, 19 und 20 (Gehbewegungen) zu sehr, um ordentlich voneinander getrennt werden zu können.

Vergleicht man die Prädiktionsmatrix des HMM-Data Klassifikators darüber hinaus mit der des GMM-Data Klassifikators sind Unterschiede erst bei sorgfältiger Betrachtung auszumachen. Die Klassen 10 und 11 (kickLFront und kickLSide) sind im HMM-Modell etwas besser voneinander zu unterscheiden, ebenso die Klassen 14 und 15 (rotateArmsBackward und rotateArmsForward – ‘Arme rückwärts rotieren’ und ‘Arme vorwärts rotieren’). Insgesamt jedoch gewinnt man den Eindruck, dass der vermehrte Berechnungsaufwand der HMMs für diese Merkmalsmenge ungerechtfertigt ist. Bevor man diese Urteil jedoch endgültig fällt, ist zu berücksichtigen, dass die vorliegende Klassenauswahl vorwiegend Klassen enthält, die aufgrund der Gelenkwinkel tatsächlich schwierig zu unterscheiden sind. Einzige Ausnahme bildet das Klassenpaar der Armrotationen (14 und 15). Möglicherweise fallen die Unterschiede im Merkmalsverlauf hier jedoch nicht ausreichend ins Gewicht, um eine Differenzierung zu ermöglichen.

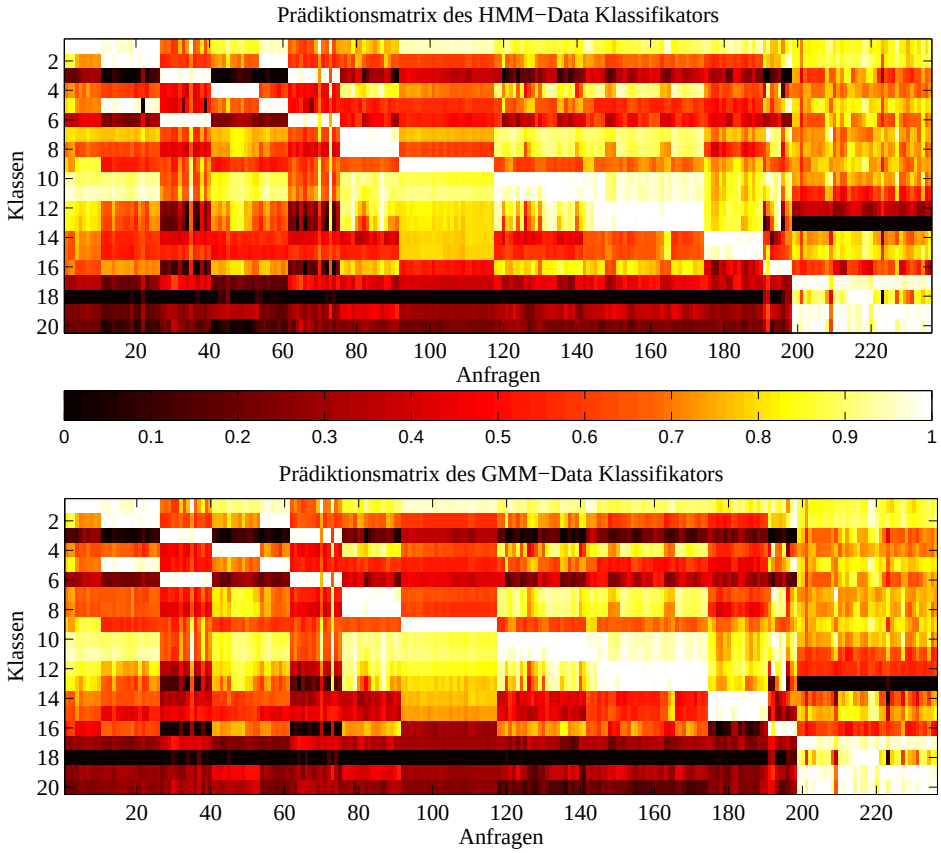


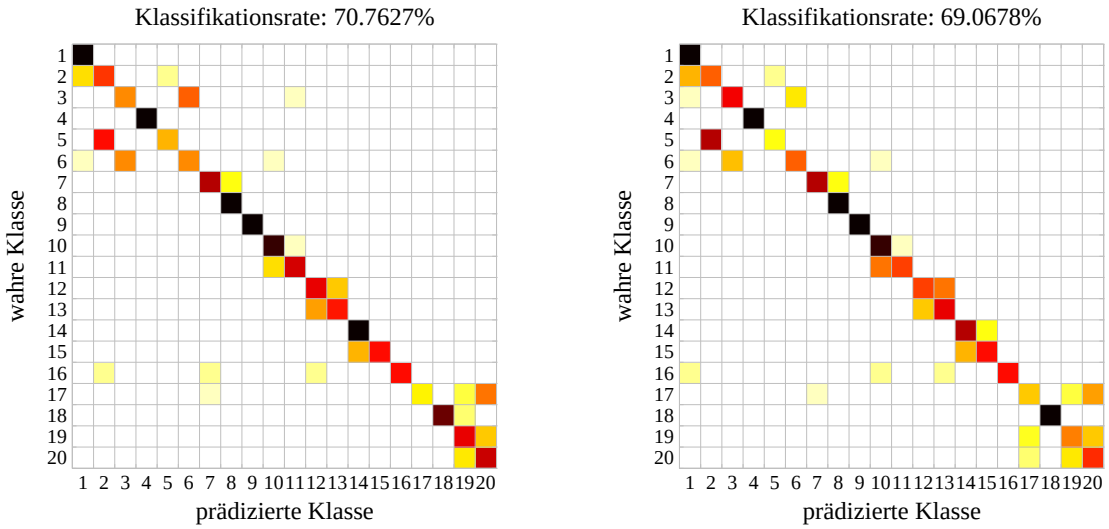
Abbildung 4.6: *Relative Prädiktionsmatrix des HMM-Data Klassifikators (oben) und des GMM-Data Klassifikators (unten) auf Grundlage der Datenbank $\mathcal{D}^{20}$, unter Verwendung von Gelenkwinkeln.*

4.4.2 Klassifikationsmatrix (Confusion Matrix)

Die absolute Klassifikationsgüte lässt sich anhand einer Klassifikationsmatrix beurteilen. Sie enthält zum einen korrekt und fehlerhaft klassifizierte Anfragen, zum anderen auch das Verhältnis korrekter Klassifikationen zur Gesamtzahl der Anfragen, die Klassifikationsrate.

Die Abb. 4.7(a) zeigt die Klassifikationsmatrix des HMM-Data Klassifikators. Wie schon zuvor bei den Prädiktionsmatrizen ist direkt daneben (Abb. 4.7(b)) die Klassifikationsmatrix des GMM-Data Klassifikators zum Vergleich abgebildet. Auch hier bestätigen sich die aus der Prädiktionsmatrix hervorgegangenen Vermutungen. Die Klassen 2 und 5 (depositFloor und grabFloor) sowie 3 und 6 (depositHigh und grabHigh) werden nach wie vor erheblich verwechselt. Ebenso verhält es sich für die Klassen 12 und 13 (punchFront und punchSide) sowie 19 und 20 (walkLeftCircle und walkRightCircle). Das ist aber aufgrund der Wahl der Merkmale nicht verwunderlich. Bei den rotateArms-Bewegungen (14 rotateArmsBackward und 15 rotateArmsForward) stellt man fest, dass die Unterscheidung relativ gut ausgefallen ist. So wurden alle Bewegungen, in denen der Darsteller den (linken) Arm vorwärts rotiert richtig zugeordnet; rotiert er ihn rückwärts kommt es zu drei Fehlklassifikationen.

Stellt man die Ergebnisse des HMM- und GMM-Data Klassifikators gegenüber, so erkennt man in der absoluten Klassifikationsrate nur eine kleine Verbesserung für den HMM-Data Klassifikator. Schaut man sich jedoch die Klassen einzeln an und vernachlässigt dabei die Klassen 2, 3, 5 und 6, bei denen sich die Zuordnung teilweise sogar verschlechtert hat, erkennt man weitere kleine Verbesserungen. So werden die kickLSide-Bewegungen (Klasse 11) nun häufiger richtig zugeordnet; die kickLFront-Bewegungen (Klasse 10) sind gleich geblieben. Gleiches gilt für die Klassen 14 (rotateArmsBackward) und 15 (rotateArmsForward). Das Ergebnis für die einfache, aus zwei Schritten bestehende Gehbewegung (walk, Klasse 17) fällt leicht schlechter aus als beim GMM-Data Klassifikator. Obgleich sich keine deutliche Verbesserung in der Klassifikation der Klassen 19 und 20 (Gehbewegungen im Kreis links- oder rechtsherum) einstellt, so werden sie nun nicht mehr mit 17 verwechselt; darüber hinaus auch etwas besser klassifiziert.



(a) HMM-Data Klassifikator

(b) GMM-Data Klassifikator

Abbildung 4.7: *Klassifikationsmatrix und -rate der HMM-Data und GMM-Data Klassifikatoren auf Grundlage der Datenbank* $\mathcal{D}^{20}$.

Erweitert man die Klassifikation auf die 57 Klassen der Datenbank $\mathcal{D}^{57}$, erhält man für die Klassifikationsmatrix die in Abb. 4.8 dargestellte Grafik. Fehlklassifikationen der kleineren Auswahl $\mathcal{D}^{20}$ werden übernommen und vermehren sich teilweise. Letzteres trifft vor allem auf die Klassen depositFloor (Klasse 4) und walkLeftCircle4StepsRstart (Klasse 54) zu. Ansonsten kommt es zu den üblichen Verwechslungen zwischen rechts- und linksseitigen Bewegungen (wie etwa jogLeftCircle, Klasse 13 und jogRightCircle, Klasse 15) und gleichartigen Bewegungen verschiedener Geschwindigkeiten wie sneak (‘Schleichen’ Klassen 40 und 41), walk (‘Gehen’, Klassen 50 bis 57), jog (‘Joggen’, Klassen 13, 14, 15), run (‘Laufen’, Klasse 33). Bei den Gehbewegungen sind vor allem die Klassen walk2StepsLstart und walk2StepsRstart (50 und 51) schlecht von anderen Gehbewegungen zu unterscheiden. Sehr kleine bis keine Vertauschungen treten bei den Klassen 27 bis 37 der Armrotationen auf.

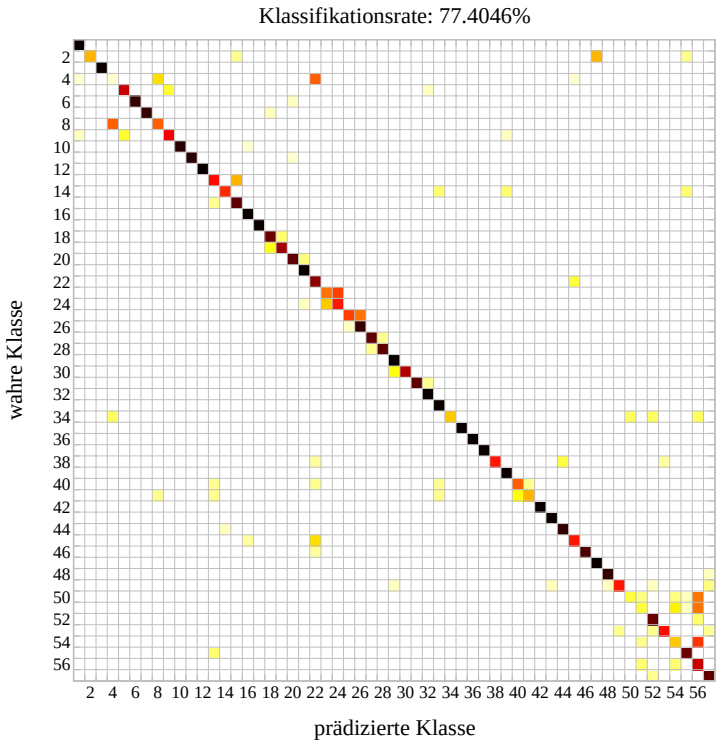


Abbildung 4.8: *Klassifikationsmatrix sowie -rate für den HMM-Data Klassifikator auf Grundlage der Datenbank $\mathcal{D}^{57}$ unter Verwendung von Gelenkwinkeln.*

4.4.3 Variation der Feature I - relationale Merkmale

In diesem Abschnitt werden nun die Klassifikationsergebnisse unter Verwendung relationaler Merkmale (vergleiche Abschnitt 2.4) erörtert. Dazu schauen wir uns zuerst die relativen Prädiktionsmatrizen an, bevor wir uns den Klassifikationsmatrizen zuwenden.

Prädiktionsmatrizen. Eine Auffälligkeit, die bei der Prädiktionsmatrix des HMM-Data Klassifikators in Abb. 4.4.3 sofort hervorsticht ist ein dunkler Streifen, der sich durch fast alle Anfragen der Klassen 14 und 15 (rotateArms) sowie 16 bis 20 (sneak und walk) zieht. Ausnahmen und damit in diesem Bereich heller eingefärbt sind bei den letzten fünf Klassen nur diejenigen Anfragen, die später den Klassen 7 oder 8 zugeordnet werden. Ein solches Ergebnis ist passend, denn es handelt sich bei den Klassen 7 und 8 um jogLeftCircle- und jogRightCircle-Bewegungen. Damit fallen sie in die Kategorie der Gehbewegungen. Bei den Gelenkwinkeln wird es nicht zu Verwechslungen der jog-Bewegungen mit einer der walk-Bewegungen kommen. Dort kommt es nur zu Vertauschungen untereinander. Umgekehrt können sich einige der walk-Bewegungen von den Bewertungen her jedoch in den Klassen der jog-Bewegungen wiederfinden.

In puncto anstehende Fehl- und nicht eindeutige Klassifikationen bleibt es bei den verwechslungsgefährdeten Paaren ähnlicher Bewegungen. Das sind Bewegungen, die rechts- oder linksseitig ausgeführt werden sowie die deposit- und grab-Bewegungen. Nichtsdestotrotz zeichnen sich im Vergleich zu den Gelenkwinkeln Verbesserungen für die Klassen 10 und 11 (kickFront und kickSide)

ab, vor allem für die Zuordnung zur 11. Klasse. Ähnliches gilt für die Klassen 12 und 13 (punchFront und punchSide) und auch für die Klassen 19 und 20 (walkLeftCircle und walkRightCircle). In allen Fällen kann man die noch ausstehenden Zuordnungen in der Prädiktionsmatrix ablesen.

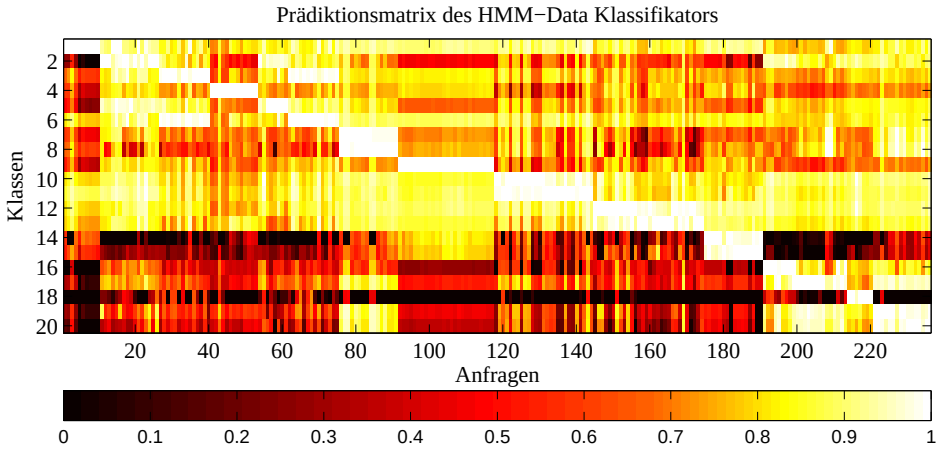
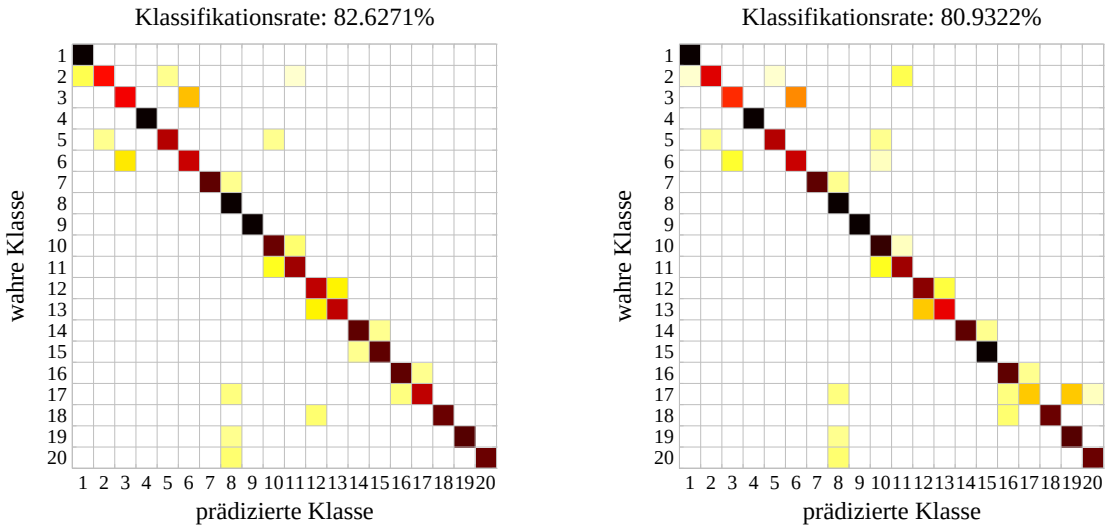


Abbildung 4.9: *Relative Prädiktionsmatrix des HMM-Data Klassifikators auf Grundlage der Datenbank $\mathcal{D}^{20}$ unter Verwendung von relationalen Merkmalen.*



(a) HMM-Data Klassifikator (b) GMM-Data Klassifikator

Abbildung 4.10: *Klassifikationsmatrix und -rate der HMM-Data und GMM-Data Klassifikatoren auf Grundlage der Datenbank $\mathcal{D}^{20}$ bei Verwendung relationaler Merkmale.*

Klassifikationsmatrizen. Schaut man sich die Klassifikationsmatrix in Abb. 4.10(a) an, erhärten sich die für die Prädiktionsmatrix ausgesprochenen Vermutungen: Wirklich Probleme bei der Klassifikation machen nur die deposit- und grab-Bewegungen (Klassen 2 und 5 sowie 3 und 6), insbesondere die Bewegungen, die mit einem Gegenstand auf dem Boden interagieren ([deposit,grab]Floor). Bei den anderen Klassen kommt es zu wenigen Fehlzuordnungen, die darüber hinaus meist in eine

sehr ähnliche Klasse fallen (kickFront und kickSide – 11 und 12 –, sneak und walk – 16 und 17). Verglichen mit der Klassifikationsmatrix des GMM-Data Klassifikators (Abb. 4.10(b)) schneiden nur die Klassen 13 (punchSide) und 17 (walk) in der Zuordnung besser ab.

Werfen wir ganz zum Schluss noch einen Blick auf die Klassifikationsmatrix der großen Datenbank $\mathcal{D}^{57}$ in Abb. 4.11. Dabei stellen wir fest, dass sich die Klassifikationsrate im Verhältnis zu den 20 Klassen nicht wesentlich verändert hat. Gleichzeitig sieht man, dass hier neben den deposit- und grab-Klassen weitere Bewegungsklassen vorkommen, die sehr schlecht getrennt werden. Dazu zählen walk2Steps[L,R]start (Klassen 50 und 51) und sneak2Steps[L,R]start ('Schlurfen', Klassen 40 und 41). Während die sneak-Bewegungen vorwiegend untereinander falsch zugeordnet werden, verteilen sich die walk-Bewegungen über alle anderen ähnlichen Bewegungen, z.B. 'Gehen', 'Schleichen' und 'Joggen'.

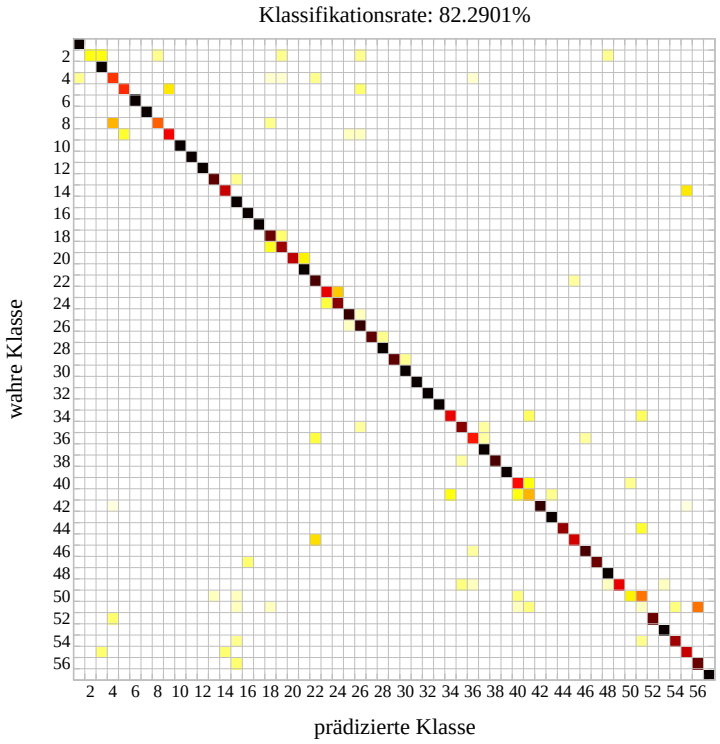
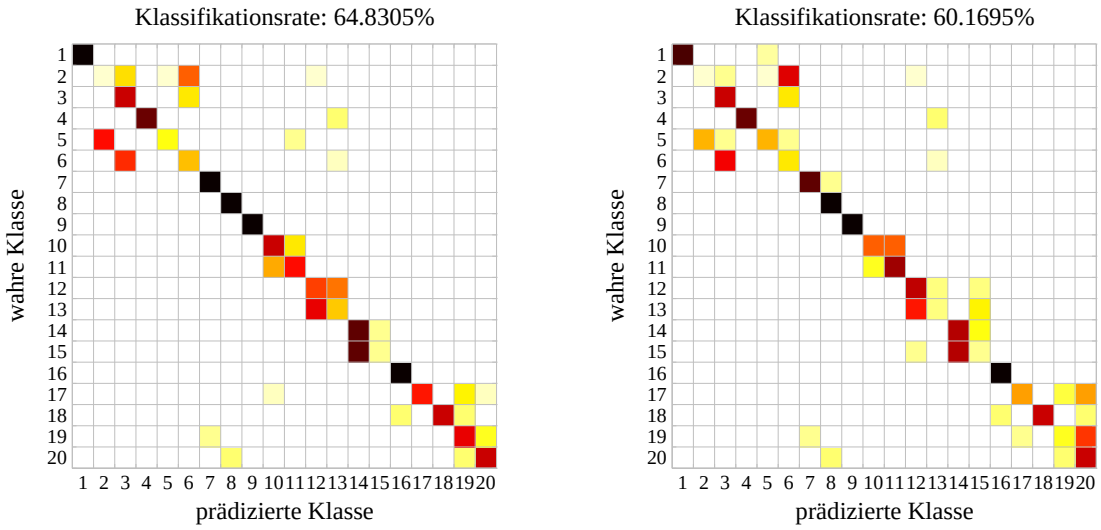


Abbildung 4.11: *Klassifikationsmatrix sowie -rate für den HMM-Data Klassifikator auf Grundlage Datenbank $\mathcal{D}^{57}$ für relationale Merkmale.*

4.4.4 Variation der Feature II - physikalische Merkmale

Nachdem wir uns gerade die Klassifikationsergebnisse der relationalen Merkmale angesehen haben, wenden wir uns nun der dritten Merkmalsmenge zu, den physikalischen Merkmalen, also den kinetischen Energien einer Auswahl von Körpersegmenten (siehe Abschnitt 2.5). Da wir uns hier primär für das Abschneiden dieser Merkmalsmenge gegenüber den beiden anderen interessieren, werden wir die Prädiktionsmatrix unerwähnt lassen und uns ausschließlich auf die Klassifikationsmatrizen konzentrieren.



(a) HMM-Data Klassifikator

(b) GMM-Data Klassifikator

Abbildung 4.12: *Klassifikationsmatrix und -rate der HMM-Data und GMM-Data Klassifikatoren auf Grundlage der Datenbank $\mathcal{D}^{20}$ bei Verwendung physikalischer Merkmale.*

Klassifikationsmatrizen. Insgesamt betrachtet schneidet diese Merkmalsmenge in der Klassifikation merklich schlechter ab als die beiden anderen Merkmalssätze. Für die kleine Datenbank $\mathcal{D}^{20}$ werden vor allem Klassenpaare wie kickL[Front,Side], punch[Front,Side] und rotateArmsL[Backward,Forward] durcheinandergebracht. Abb. 4.12(a) zeigt dies deutlich. Um eine Gegenüberstellung der Ergebnisse von HMM und GMM zu erleichtern, wurde die Klassifikationsmatrix des GMM-Data Klassifikators erneut abgebildet (Grafik 4.12(b)). Eine bemerkenswerte Verbesserung im Klassifikationsergebnis hat sich jedoch höchstens für die Klassen 17 und 19 eingestellt (walk2Steps und walkLeftCircle). Nicht besser, aber ebenso gut wie für andere Klassifikatoren oder Merkmalssätze, schneiden die Klassen jog[Left,Right]Circle ab (7 und 8) ab.

Im Gegensatz zu den beiden vorangegangenen Merkmalssätzen ist es mit den physikalischen Merkmalen nach wie vor nicht möglich, die beiden rotateArmsL-Bewegungen zu unterscheiden. Zwar erkennen sie, dass es sich um tendentiell gleiche Klassen handelt, eine Trennung bringen sie jedoch nicht zustande. Dieser Misserfolg ist tatsächlich auf die Merkmale zurückzuführen: Die interessanten Segmente sind für die Rotationsbewegungen der linke Unter- und Oberarm. Da der linke Arm einmal aus der Schulter vorwärts bzw. rückwärts gedreht wird, ist ein charakteristischer Energieanstieg oder -abfall in den betroffenen Segmenten nicht zu verzeichnen. Der Versuch einer Unterscheidung erübrigt sich damit auch bei Verwendung von HMMs. Eine Verbesserung der Zuordnung ergab sich nur bezogen auf die Klasse der Schleichbewegungen (sneak, Klasse 16), hier einwandfrei erkannt.

Erweitern wir unsere Betrachtung auf die Datenbank $\mathcal{D}^{57}$, Abb. 4.13 zeigt die dazugehörig Klassifikationsmatrix, setzen sich auch hier Fehler fort. Eine korrekte Klassifikation der rotateArms-Bewegungen ist ebenso aussichtslos wie die der punch- oder kick-Bewegungen. Bewegungen wie standUpLieFloor (Klasse 45) oder standUpSitFloor (Klasse 46) werden sicher, jedoch fälschlicherweise, der Klasse lieDownFloor (22) zugewiesen.

Verglichen mit den Ergebnissen des GMM-Klassifikationsverfahrens schnitten ein paar Klassen besser ab. Darunter sind jogOnPlace (Klasse 14) und staircase[Down, Up] (Klassen 43 und 44). Bezogen auf die beiden anderen Merkmalssätze sind keine nennenswerten Verbesserungen zu verzeichnen.

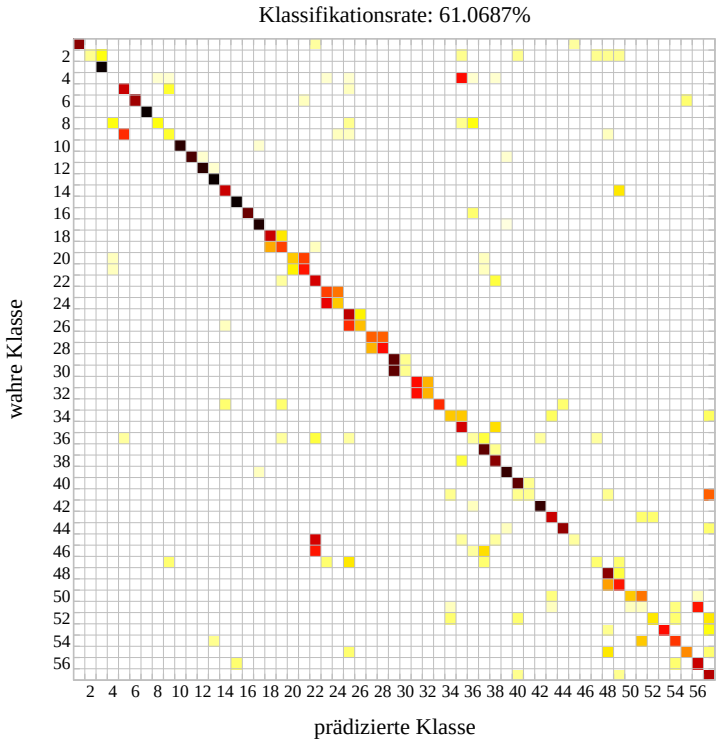


Abbildung 4.13: *Klassifikationsmatrix sowie -rate für den HMM-Data Klassifikator auf Grundlage Datenbank $\mathcal{D}^{57}$ für physikalische Merkmale.*

4.5 Analyse

Wir grenzen die genauere Analyse einiger typischen Klassen auf die Merkmalsmenge der Gelenkwinkel (siehe Tabelle 2.1) und die der relationalen Merkmale (siehe Tabelle 2.2) ein. Außerdem nutzen wir die Klassifikationsergebnisse der Datenbank $\mathcal{D}^{20}$.

4.5.1 Die Klassen depositFloorR und grabFloorR

Die Verwechslung dieser beiden Klassen untereinander ist auch bei Verwendung von HMMs nicht verwunderlich. Die Gründe hierfür liegen wie schon in Abschnitt 3.5.1 erwähnt in der Ähnlichkeit der beiden Bewegungen an sich, der Skelettauflösung und nicht zuletzt auch in der Wahl der Merkmale (Gelenkwinkel).

HMM-Data. Schaut man sich die Klassifikationsmatrix der *Gelenkwinkel* an, so stellt man fest, dass die Klasse grabFloor nicht selten mit depositFloor verwechselt wird (zu 62,5%). Die deposit-

Floor Bewegungen werden hier – wie schon beim GMM-Data Klassifikator – überwiegend (und fälschlicherweise) als Radschlag (cartwheel) erkannt. Die beiden Bewegungen sind sich sowohl am Anfang als auch am Ende tatsächlich ähnlich. Beugt sich der Darsteller zuerst nach unten und richtet sich dann wieder auf. Nur der Mittelteil des Radschlags fehlt der depositFloor-Bewegung; der Mittelteil in dem die Beine sich nacheinander vom Boden ab- und über den Kopf hinwegheben, bevor sie wieder am Boden angelangen. Warum also wird das 'etwas-auf-dem-Boden-ablegen' als ein Radschlag erkannt?

Das für die jeweilige Klasse gelernte Modell berücksichtigt verschiedenartige Realisierungen der betroffenen Bewegung. Das Klassenmodell repräsentiert sie am Ende alle gemeinsam. Das heißt gleichzeitig auch, dass eine einzelne Anfrage nicht die Möglichkeiten des Modells ausschöpft und einige Zustände mitunter gar nicht besucht werden. Für unsere deposit-Bewegung bedeutet dieser Umstand, dass sie oben erwähnten Mittelteil einfach überspringen kann. Es ist zulässig, nach dem Hinunterbeugen und einer kurzen Verweilen, in ein Aufrichten überzugehen. Um dies zu verdeutlichen wurde in Abb. 4.14 für eine falsch klassifizierte depositFloor-Bewegung sowohl der Pfad durch das favorisierte (cartwheel) als auch der durch das korrekte Klassenmodell (depositFloor) dargestellt. In beiden Fällen wurden nicht alle Zustände besucht, und es gibt dennoch einen gültigen Pfad durch die jeweiligen Modelle. Dass etwas derartiges möglich ist, liegt daran, dass wir uns für die Modellierung einer Bewegungsklasse durch ergodische HMMs entschieden haben (siehe dazu auch Anmerkung in Abschnitt 4.4 auf Seite 49).

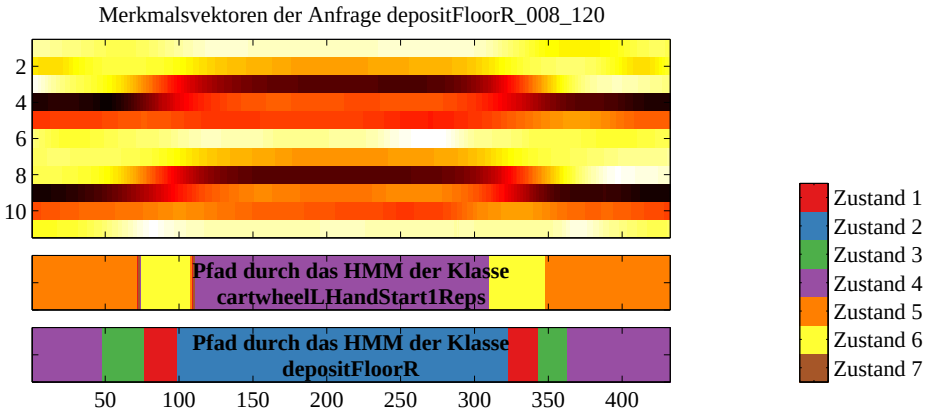


Abbildung 4.14: *Merkmalsdaten (Gelenkwinkel, oben) und Viterbi-Pfad durch die HMMs der Klasse cartwheel (oberer Pfad) und depositFloor (unten) für eine depositFloor-Bewegung. Die Zustände sind farblich kodiert. Bei den Merkmalsdaten entspricht weiß dem Wert 180 und schwarz 0. Der Verlauf ist von weiß über gelb und rot nach schwarz.*

Von allen anderen Klassen (bis auf die cartwheel-Klasse und die jeweilig andere) heben sich depositFloor und grabFloor schon in der Prädiktionsmatrix signifikant ab. Letztendlich übertragen sich hier die Ergebnisse und Erläuterungen der im vorangegangenen Kapitel in Abschnitt 3.5.1 behandelten GMM-Klassifikation mittels GMM-Data Klassifikator.

Bei den *relationalen Merkmalen* heben sich die beiden Klassen nicht mehr so von den anderen ab, wie es bei den Gelenkwinkeln der Fall war. Dies deutet darauf hin, dass die gewählten relationalen Merkmale diese Bewegungen schlechter charakterisieren als die Gelenkwinkel. Dennoch verbesserte sich die Klassifikationsrate beider Klassen. Es kommt noch immer zu Verwechslungen

der deposit-Bewegung mit dem Radschlag in drei Fällen, bei den Gelenkwinkeln waren es fünf. Erheblich besser als depositFloor schneidet grabFloor ab. Eine Ursache hierfür ist, dass in fast allen grabFloor-Bewegungen der jeweilige Akteur beim Aufheben die Knie beugt oder ganz in die Hocke geht. Bei den depositFloor-Bewegungen dagegen beugt sich der Darsteller meist nur im Becken nach vorne und geht nicht in die Hocke.

4.5.2 Die Klassen punchLFront1Reps und punchLSide1Reps

Einen zeitabhängigen Unterschied zwischen den beiden Boxbewegungen kann man anhand keiner der beiden Merkmalssätze ausmachen. Es ist daher von vorneherein unwahrscheinlich, dass die Klassifikation bei der HMM-Klassifikation erfolgreicher ausfällt als im GMM-Klassifikationsverfahren. Auswirkungen verschiedener Ausführungen der Bewegungen und Unzulänglichkeiten der Merkmalssätze wurden in Abschnitt 3.5.1 angesprochen und illustriert. Sie gelten auch hier.

HMM-Data. Für die *Gelenkwinkel* ist die Ähnlichkeit der beiden Bewegungen untereinander beträchtlich. Einen ausschließlich auf die Klassen punchLFront und punchLSide beschränkten Ausschnitt aus der Prädiktionsmatrix ist in Abb. 4.15 links dargestellt. Die angesprochene Ähnlichkeit äußert sich dort in der homogenen Weißfärbung des Ausschnitts. Wirft man einen Blick zurück auf die gesamte Prädiktionsmatrix (Abb. 4.6 auf Seite 51), stellt man nach wie vor – bei den GMMs war es auch so – eine Artverwandtschaft der Boxbewegungen mit den Trittbewegungen (kickL[Front,Side]) fest. Erneut lässt sich dies damit erklären, dass es sich nicht um idealisierte Boxbewegungen handelt; und auch nicht um idealisierte Tritttbewegungen.

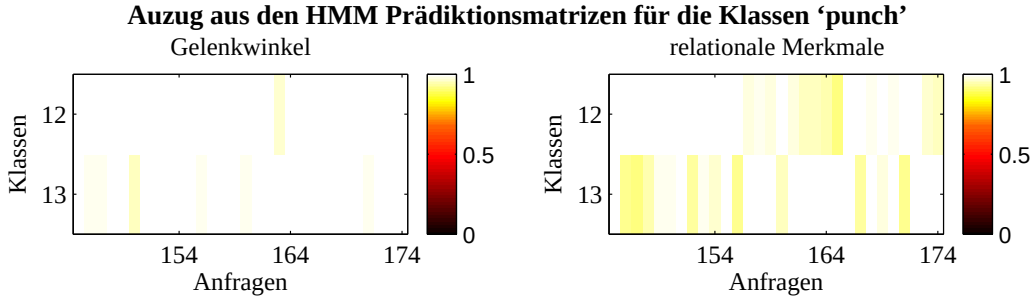


Abbildung 4.15: *Ausschnitt aus den Prädiktionsmatrix der Datenbank $\mathcal{D}^{20}$ auf Grundlage der Gelenkwinkel (links) und relationaler Merkmale (rechts) für die beiden Klassen punchLFront und punchLSide (12 und 13).*

Nutzt man *relative Merkmale* zur Klassifikation zeigt sich in der Klassifikationsmatrix eine leichte Verbesserung der Zuordnung verglichen mit den Gelenkwinkeln. Der abgebildete Ausschnitt der Prädiktionsmatrix (Grafik 4.15 rechts) bestätigt eine bessere Trennbarkeit. Leider sind dabei auch die Fehlklassifikationen überzeugter: Anfrage 160 ist eigentlich eine punchLSide-Bewegung, während es sich bei Anfragen 157 bis 159 um punchLFront-Bewegungen handelt. Die Zeitinformation der Daten kann aufgrund der Verwendung ergodischer HMMs nicht ausgenutzt werden.

4.5.3 Die Klassen rotateArmsLForward1Reps und rotateArmsLBackward1Reps

Gerade für diese beiden Klassen waren die Erwartungen an das HMM-Klassifikationsverfahren hoch. War es doch sowohl für die Gelenkwinkel als auch für die relationalen Merkmale die Reihenfolge der Daten, die einen Unterschied zwischen den beiden Klassen erkennbar machte. Und es war die Reihenfolge, die bei den GMMs nicht berücksichtigt werden konnte. Vertauscht werden beide Klassen höchstens untereinander.

HMM-Data. Schaut man sich die Ergebnisse für die *Gelenkwinkel* an, so sieht man diese Erwartung teilweise bestätigt. Die Klassifikation erkennt alle rotateBackward-Bewegungen als solche. Bei den rotateForward-Bewegungen werden allerdings noch immer nicht alle korrekt zugeordnet. Der Blick auf einen Ausschnitt der Prädiktionsmatrix in Abb. 4.16 links zeigt auch, dass sich vor allem rotateArmsBackward nun von rotateArmsForward abhebt. Während es bei den 'Arm vorwärts rotieren'-Bewegungen einige Fehlzuordnungen gibt, fällt die Trennung von den 'Arm rückwärts rotieren' ordentlich aus. Im Vergleich mit den Ergebnissen des GMM-Data Klassifikators stellte sich also eine bessere Trennung der beiden Klassen ein. Damit verbunden auch eine ist die Klassifikationsrate.

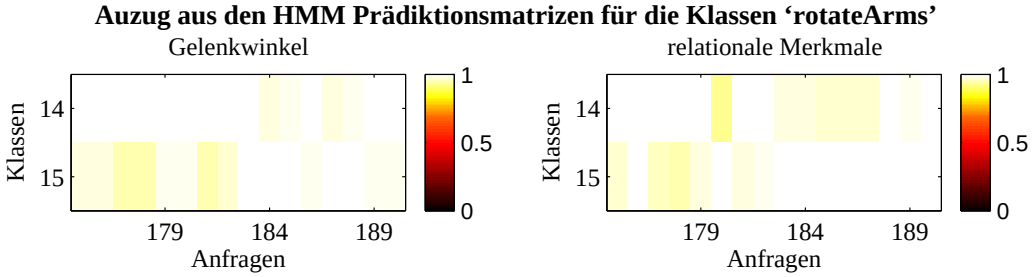


Abbildung 4.16: *Ausschnitt aus den Prädiktionsmatrix der Datenbank $\mathcal{D}^{20}$ auf Grundlage der Gelenkwinkel (links) und relationaler Merkmale (rechts) für die beiden Klassen rotateArmsLBackward und rotateArmsLForward (14 und 15).*

Zusammen mit den Zeitinformationen enthaltenden *relationalen Merkmalen* (vergleiche Abschnitt 3.5.3 und die dort abgebildete Grafik) gelang schon bei den GMMs eine Verbesserung der Klassifikationsrate. Diese setzt sich hier fort, wenngleich beide Klassen eine Fehlzuordnung aufweisen. An der Trennbarkeit der Klassen voneinander hat sich nichts nennenswert geändert, weder verglichen mit der GMM-Klassifikation noch mit den Gelenkwinkel.

4.5.4 Die Klassen der Gehbewegungen

Wir beziehen uns in diesem Abschnitt auf die Klassen walk2Steps, walkBackwards und das Klassenpaar walk[Left,Right]Circle. Es handelt sich um die Klassen 17 bis 20. Unter den genannten wurde für beide Merkmalssätze die Klasse der Rückwärtsgehbewegungen am besten zugeordnet. Grundsätzlich hätte man auch hier eine Erhöhung der Klassifikationsraten erwartet, denn auch hier kann die Zustandsfolge des HMM zeitliche Zusammenhänge kodieren.

HMM-Data. Für die *Gelenkwinkel* war der Erfolg diesbezüglich nicht groß. Zwar heben sich die Gehbewegungen von den anderen Klassen in der Prädiktionsmatrix ab (siehe Gesamtmatrix in Abb. 4.6), untereinander (abgesehen von walkBackwards) aber ist nicht klar ersichtlich, welche Anfrage welcher Klasse zugeordnet wird (vergleiche Ausschnitt in Abb. 4.17 links). Die Klassifikation der Gehbewegungen aus zwei Schritten (walk2Steps) scheitert hier nicht unbedingt am Modell oder den Merkmalen. Betrachtet man das Ergebnis genau, so stellt man Folgendes fest: Verwechslungen von walk2Steps treten vorwiegend mit einer der walk[Left,Right]Circle-Bewegungen auf. Dazu sollten wir anmerken, dass es sich bei den Klassen walk[Left,Right]Circle um vier Schritt lange Gehbewegungen handelt. Walk2Steps besteht aus nur zwei Schritten. Unerheblich mit welchem Bein der erste Schritt gemacht wird, zwei Schritte sind in vieren stets enthalten. Nun hängt es schlussendlich an den Merkmalen, die nicht diskriminativ genug sind, um Bewegungsrichtungen zu repräsentieren. Deshalb können sie ein Geradeausgehen, ein linksherum-im-Kreis-Gehen und ein rechtsherum-im-Kreis-Gehen nicht auseinanderhalten. Aus denselben Gründen kommt es zu vereinzelten Verwechslungen mit jog[Left,Right]Circle, die ebenfalls mit einer Länge von vier Schritten in der Datenbank enthalten sind. Bei den walk[Left,Right]Circle-Bewegungen untereinander sind es von Anfang an die Merkmale, die eine Unterscheidung nicht ermöglichen.

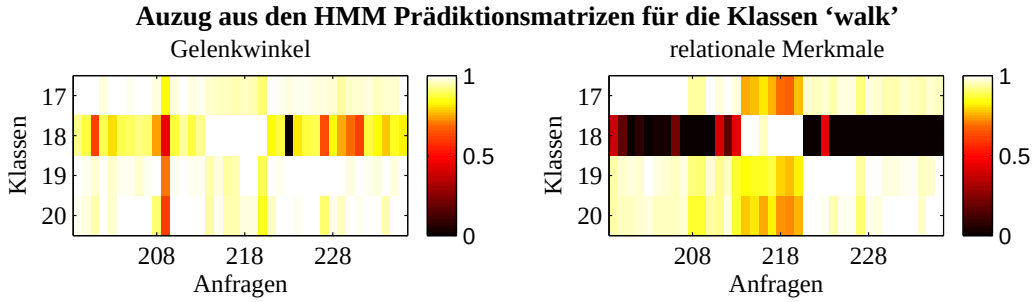


Abbildung 4.17: *Ausschnitt aus den Prädiktionsmatrix der Datenbank $\mathcal{D}^{20}$ auf Grundlage der Gelenkwinkel (links) und relationaler Merkmale (rechts) für die vier Klassen walk, walkBackwards, walkLeftCircle und walkRightCircle (17, 18, 19 und 20).*

Verwendet man *relationale Merkmale*, grenzt sich das Rückwärtsgehen noch stärker von den anderen Gehbewegungen ab, siehe Ausschnitt der Prädiktionsmatrix in Abb. 4.17 rechts. Das reine Gehen (walk2Steps) wird häufiger erkannt als bei den Gelenkwinkeln und dennoch vereinzelt einer der im-Kreis-Joggen- oder der Schleich-Bewegung zugeordnet. Das erstaunt kaum, denn das Schleichen ist eine verlangsamte und vorsichtige Form des Gehens. Darüber hinaus besteht Schleichen seinerseits – wie das Gehen – aus zwei Schritten, und die Darsteller beginnen für beide Bewegungstypen mit dem gleichen Fuß (es ist der linke). Die ebenfalls in einem HMM enthaltene Information über Bewegungsgeschwindigkeit scheint für die Klassen jog[Left, Right]Circle und walk[Left, Right]Circle nicht auszureichen, sie endgültig voneinander trennen zu können. Daher verirren sich die einige Anfragen der walk[Left,Right]Circle-Kategorie in die jogRightCircle-Klasse. Genau genommen wird jeweils eine falsch zugeordnet.

4.5.5 Fazit

Die Mängel in Auflösung der Daten, Aussagekraft der Gelenkwinkel und beschränkten Anzahl Merkmale von Gelenkwinkeln und relationalen Merkmalen wurden schon in Abschnitt 3.5.5 erläutert. Sie bestehen weiterhin und sind hiermit erwähnt. Erhebliche Verbesserungen in der Klassifikation durch Verwendung relationaler Merkmale gab es auch beim HMM-basierten Klassifikationsverfahren. Statt also erneut auf die Merkmale einzugehen, wollen wir uns hier dafür interessieren, was sich mit Nutzung der HMMs geändert hat.

Durch den Einsatz von HMMs kam es bei einigen Klassen zu Verbesserungen der Klassifikationsrate. Dazu zählen hauptsächlich die rotateArms-Klassen. In vielen Fällen führen die HMMs allerdings nicht zu einer höheren Erfolgsquote. Das liegt teilweise daran, dass die enthaltenen Zeitinformationen zur Unterscheidung nicht ausreichen oder gar nicht erst herangezogen werden können. Bei denjenigen Klassen, die eine Unterscheidung durch ihren zeitlichen Verlauf zulassen, war der HMM-Data Klassifikator wie erwartet erfolgreicher als die GMM-basierten Klassifikationsverfahren, obwohl die HMMs vollständig verbunden sein konnten.

4.5.6 Bewertung

Prinzipiell erhofften wir uns durch die Verwendung von Hidden Markov Modellen einen (teilweisen) Erhalt der in den Daten enthaltenen Zeitinformationen in den Modellen. Damit verbunden ist eine Verbesserung der Klassifikation oder zumindest eine bessere Trennung der Klassen. Schaut man sich die Ergebnisse in Prädiktions- und Klassifikationsmatrizen an, erfüllte sich diese Hoffnung im ersten Moment nur teilweise bis gar nicht. Es gibt eine große Anzahl von Klassen, bei denen die Verwendung von HMMs keine Vorteile bringt.

In der Datenbank sind sowohl vor als auch nach Merkmalsextraktion zwei Arten von Bewegungsklassen enthalten. Bei der einen besteht die Unterscheidungsmöglichkeit nur in der zeitlichen Abfolge der Merkmalsvektoren. Beispiele hierfür sind für Gelenkwinkel und relationale Merkmale die Klassen rotateArmsL[Backward,Forward], walk2Steps[L,R]Start und analoge. Bei diesen Klassen gibt es einen klaren zeitlichen Verlauf, durch den sie jeweils voneinander zu unterscheiden sind. Die zweite Art kann sich von den anderen Klassen primär durch die Einmaligkeit ihrer einzelnen Merkmalsvektoren unterscheiden, da der zeitliche Aspekt in diesem Sinne unerheblich ist. Dazu zählen die deposit- und grab-Bewegungen, aber auch – je nach Merkmal – Bewegungen wie walk[Left,Right]Circle4StepsRstart.

Für Klassen der ersten Art kann das HMM vom typischen zeitlichen Merkmalsverlauf profitieren. Es beschreibt die Reihenfolge der Daten mit Hilfe der Zustandsübergangsmatrix A (siehe Gleichung 4.1.1) und liefert erwartungsgemäß mehr Informationen über die Bewegungsklasse als ein entsprechendes GMM. Das führt zu einer besseren Beschreibung der Klasse und einer höheren Klassifikationrate gegenüber einem GMM gleicher Modellordnung. Für Bewegungsklassen des zweiten Typs bringt ein HMM keinen größeren Nutzen als ein GMM, denn die Zustandsübergangswahrscheinlichkeiten a_{ij} enthalten keine zusätzliche Information über die Klasse.

Betrachtet man die Einträge einer Zustandsübergangsmatrix als Gewichtungen der Zustände, z.B. durch Normierung der Spaltensummen über

$$w_i = \frac{1}{N} \sum_{j=1}^{N} a_{ij}, \qquad 1 \leq i \leq N,$$

können die w_i als Gewichte eines GMM aufgefasst werden. Das ist sinnvoll, wenn die Beobachtungsdaten über keine (interessante) zeitliche Strukturierung verfügen. Für solche Daten sind Zustandsübergänge zufällig und nur durch die Auftrittshäufigkeit eines Zustands gewichtet. Der Sinn und Zweck des HMM wurde in diesem Fall zwar verfehlt, aber es kann dennoch die Informationen eines GMM liefern, dessen Komponentenverteilung den Zustandsverteilungen des HMM entspricht.

Dies verdeutlicht, warum das hier verwendete HMM-Klassifikationsverfahren für die beschriebenen Datenbanken keine nennenswert besseren Ergebnisse liefert als die Verwendung des in Kaptel 3 beschriebenen GMM-Klassifikationsverfahren. Außerdem erklärt es die Ähnlichkeit der Prädiktionsmatrizen von HMM- und GMM-Data Klassifikator.

Der höhere rechnerische Aufwand des Baum-Welch-Algorithmus zum Training der Klassenmodelle und des Vorwärts-Algorithmus zur Klassifikation zahlt sich hier für viele Klassen nicht aus und ist infolgedessen auch nicht gerechtfertigt. Zudem erkennt man, dass diejenigen Klassen, bei denen die Zeitinformation zur Unterscheidung sehr wichtig ist, nicht zufällig besser klassifiziert wurden als bei den GMMs. Hierbei scheitern noch immer die Klassen walk2Steps[L,R]Start, meistens jedoch nicht aneinander. Denn häufig werden sie längeren Bewegungsklassen zugeordnet, d.h. solchen, die ebenfalls walk-Bewegungen enthalten und aus vier statt aus nur zwei Schritten bestehen. Von daher wird die enthaltene Zeitinformation genutzt, aber leider falsch.

Zur Übersicht über die erzielten Klassifikationsergebnisse schließen sich noch zwei Tabellen an, die diese übersichtlich untereinander auflisten. Tabelle 4.3 bezieht sich auf die Datenbank aus 20 Bewegungsklassen, $\mathcal{D}^{20}$, Tabelle 4.4 auf die mit 57 Klassen, $\mathcal{D}^{57}$. Bei beiden fällt ein erheblicher Anstieg der Klassifikationsrate auf, sobald die zweitbeste Klasse hinzukommt. Auch hier kommt die Verwechslung sehr ähnlicher Klassen untereinander sehr häufig vor und wird durch Berücksichtigung der jeweiligen Zweitwahlen aufgehoben.

Klassifikator		Top 1	Top 2	Top 3	Top 4
Gelenkwinkel	HMM vs Daten	70.763	94.915	97.458	98.305
	GMM vs Daten	69.068	93.644	97.034	97.458
	GMM – KL und EMD	64.831	88.983	94.492	97.458
relationale	HMM vs Daten	82.627	95.339	96.610	97.458
	GMM vs Daten	80.932	93.644	95.763	97.458
	GMM – KL und EMD	77.542	93.644	96.186	96.610
physikalische	HMM vs Daten	64.831	88.136	92.797	95.763
	GMM vs Daten	60.169	83.051	88.136	90.678
	GMM – KL und EMD	56.356	72.458	78.390	83.051

Tabelle 4.3: *Klassifikationsraten der vier besten Klassen (Top 1 bis Top 4) ausgewertet für HMMs und GMMs mit Gelenkwinkeln, relationalen und physikalischen Merkmalen für den HMM-, den GMM-Data sowie den KL-EMD Klassifikator. Als Datengrundlage diente die Datenbank $\mathcal{D}^{20}$.*

4.6 Ein kleines Experiment zum Schluss

In der Datenbank sind Bewegungen enthalten, bei denen fast ausschließlich ihr zeitlicher Verlauf relevant für eine Klassifikation ist, je nachdem welche Merkmalsmenge man heranzieht. Für die Gelenkwinkel ist ein solches Klassenpaar rotateArms[Backward, Forward]. Initialisiert man beide

Klassifikator		Top 1	Top 2	Top 3	Top 4
Gelenkwinkel	HMM vs Daten	77.405	91.298	94.504	95.420
	GMM vs Daten	77.252	90.840	93.435	95.420
	GMM – KL und EMD	63.664	84.122	89.924	92.824
relationale	HMM vs Daten	82.290	92.672	94.656	95.573
	GMM vs Daten	81.221	91.450	94.046	95.573
	GMM – KL und EMD	68.397	84.580	88.550	91.298
physikalische	HMM vs Daten	61.069	80.153	86.565	89.160
	GMM vs Daten	55.878	75.115	82.595	85.344
	GMM – KL und EMD	39.237	54.809	62.443	66.565

Tabelle 4.4: *Klassifikationsraten der vier besten Klassen (Top 1 bis Top 4) ausgewertet für HMMs und GMMs mit Gelenkwinkeln, relationalen und physikalischen Merkmalen für den HMM-, den GMM-Data sowie den KL-EMD Klassifikator. Als Datengrundlage diente die Datenbank $\mathcal{D}^{57}$.*

Klassen zusammen und berechnet erst in der folgenden Parameteroptimierung getrennte Klassenmodelle, könnte sich auch bei den ergodischen HMMs eine bessere Detektionsrate ergeben.

Für diesen Ansatz kommen nicht alle Klassen in Frage, und die Klassenauswahl ist eine delikate Angelegenheit. Denn so vielversprechend diese Idee im ersten Moment sein mag, kann sie sich leicht zu einer Katastrophe entwickeln. Ein Beispiel: Die Klassen der rotateArmsL-Bewegungen sind zweifelsohne ein sinnvoller Kandidat. Hingegen sind die punchL-Bewegungen nicht geeignet, denn an ihrem zeitlichen Verlauf kann man die punchLFront- und -Side-Bewegungen nicht unterscheiden (vergleiche Abb. 3.16 auf Seite 36). Bewegungen wie walk2Steps[L,R]start sind grundsätzlich geeignete Anwärter für eine gemeinsame Initialisierung, eine anschließende Verwechslung mit [jog,walk]4Steps[Left,Right]Circle kann aber nicht ausgeschlossen werden. Ferner ist zu beachten, dass durch die gemeinsame Modellinitialisierung eine hohe Ähnlichkeit der endgültigen Modelle untereinander bleibt. Somit ist es denkbar, dass sich parallel initialisierte Klassen zwar vom Rest besser unterscheiden, eineinander jedoch gleichzeitig ähnlicher werden.

In einem kleinen Experiment haben wir dieses Vorgehen für die beiden o.g. Klassen getestet, d.h. wir verwendeten die in der Datenbank $\mathcal{D}^{20}$ enthaltenen Klassen rotateArmsL[Backward,Forward] mit Gelenkwinkeln. Die Initialisierung erfolgte nach dem in Abschnitt 4.1.3 beschriebenen Verfahren mit den Trainingsdaten beider Klassen. Im Anschluss daran wurden die Trainingsdaten wieder getrennt und die einzelnen Modellparameter klassenweise optimiert. Überlegt man sich, wie die endgültigen Zustandsübergangsmatrizen aussehen könnten, kommt man zu dem Schluss, dass sie durch Transponieren auseinander hervorgehen sollten. Die Reihenfolge der zu besuchenden Zustände müsste sich für die beiden Bewegungen nämlich genau umkehren.

Die Ergebnisse sind in Abbildungen 4.18 und 4.19 festgehalten. Dort erkennt man, dass die Zustandsübergangsmatrizen beinahe zueinander transponiert sind und die Trefferquote sich für die Zuordnung der beiden Klassen erhöht hat. Die betroffenen Bereiche sind in der Abbildung 4.19 blau umrahmt.

Ein Vorteil der gemeinsamen Initialisierung von Klassen ist, dass ihr zur Berechnung eines Anfangsmodells mehr Daten zur Verfügung stehen. Ein gutes Anfangsmodell ist für die weitere Optimierung wichtig, denn letztere findet nur lokale Maxima. Nachteilig ist, dass das Klassifikationsverfahren nicht mehr vollautomatisch ist, da im Vorfeld über gemeinsam zu initialisierende

Klassen nachgedacht werden muss. Insgesamt ist jedoch das sinnvollste Resultat aus diesem Experiment, dass die Zustandsübergangsmatrix verschiedener Bewegungen diese repräsentieren kann. So entsprechen zueinander transponierte Zustandsübergangsmatrizen beispielsweise Bewegungen, die eine zeitliche Umkehrung voneinander sind.

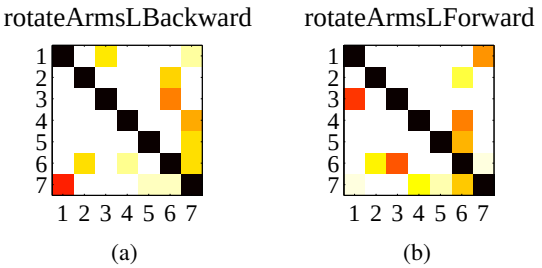


Abbildung 4.18: *Zustandsübergangsmatrizen für die Klassen rotateArmsLBackward (a) und rotateArmsLForward (b). Ein Eintrag an Position (i,j) bedeutet, dass ein Übergang von Zustand i in Zustand j möglich ist. Die Farbgebung zeigt die Wahrscheinlichkeit eines Übergangs an. Weiß zeigt dabei null an, schwarz 0.1 und höher.*

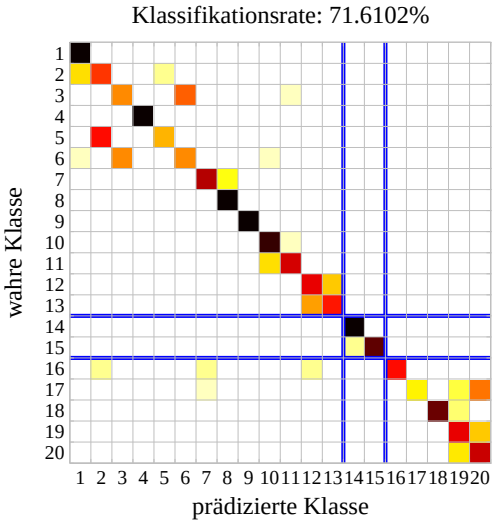


Abbildung 4.19: *Klassifikationsmatrix sowie -rate für den HMM-Data Klassifikator auf Grundlage der Datenbank $\mathcal{D}^{20}$ unter Verwendung von Gelenkwinkeln. Die markierten Klassen rotateArmsBackward und -Forward wurden gemeinsam initialisiert, bevor die getrennte Parameteroptimierung jeweils ein Klassenmodell erstellte.*

Anhang A

Datenbank der Bewegungsklassen

Training und Klassifikation basierten auf einem Auszug aus einer Datenbank systematisch aufgenommener Motion Capture Daten, die festgelegte Bewegungssequenzen beinhaltet. Die Bewegungen wurden von fünf Darstellern mehrere Male ausgeführt und später manuell in einzelne „Bewegungsschnipsel“ getrennt. Die endgültige Auswahl besteht aus 57 geeigneten Klassen mit je 10 bis 50 verschiedenen Realisierungen desselben Bewegungstyps. Die Tabelle A.1 enthält eine Liste der beteiligten Klassen. Zu Trainigs- und Evaluationszwecken wurde die Datenbank in zwei disjunkte etwa gleich große Datenbanken unterteilt. Ein Großteil der Experimente wurde jedoch auf einer kleineren Datenbank ausgeführt. Diese enthält nur noch 20 von den 57 aufgeführten und ist in Tabelle A.2 zusammengestellt.

In beiden Tabellen sind Trainingszeiten für die Berechnung eines Klassen-GMMs aufgeführt. Die Zeiten wurden auf einem 2.00GHz-Intel Pentium 4-M mit 512 MB RAM erzielt. Dazu merken wir an, dass die Zeiten nicht alleine abhängen von der Komplexität der jeweiligen Bewegungsklasse, d.h. der Anzahl Frames pro Motion Capture Datenstrom und der Anzahl Daten in der Klasse, sondern auch davon, wie viele Iterationsschritte der k-means Algorithmus benötigt, um sich auf den k Clusterzentren einzupendeln.

Bei der Berechnung von GMMs ist die maximale Anzahl Iterationen des k-means Algorithmus auf 100 begrenzt. Darüber hinaus wurde die Anzahl der Komponenten eines GMM und damit auch die Anzahl der Clusterzentren auf $k = 7$ fixiert. Auf eine weitere Spalte mit den Trainingszeiten der HMMs wurde verzichtet. Wir merken daher nur an, dass der Zeitaufwand im Vergleich zum Klassifikationsergebnis unverhältnismäßig ist.

A.1 HDM05_cut_57 oder auch $\mathcal{D}^{57}$

Tabelle A.1: *N - die Gesamtzahl der Bewegungen, av #f - Durchschnittliche Anzahl Frames pro Bewegung, N_E - Anzahl der Bewegungen in der Evaluationsdatenbank, av $\#f_E$ - Durchschnittliche Anzahl Frames pro Bewegung in der Evaluationsdatenbank, N_T - die Anzahl der Bewegungen in der Trainingsdatenbank, av $\#f_T$ - Durchschnittliche Anzahl Frames pro Bewegung in der Trainingsdatenbank, Zeit - Klassentrainingszeit in Sekunden*

Nr.	Bewegungsklasse $\mathcal{C}$	N	av $\#f$	N_E	av $\#f_E$	N_T	av $\#f_T$	Zeit [s]
1	cartwheelLHandStart1Reps	21	411	10	398	11	422	2.534
2	clap1Reps	17	43	8	42	9	44	0.701
3	clapAboveHead1Reps	17	94	8	93	9	95	0.661
4	depositFloorR	32	363	16	434	16	293	2.204
5	depositHighR	28	245	14	223	14	268	2.293
6	elbowToKnee1RepsLelbowStart	27	147	13	139	14	154	1.732
7	elbowToKnee1RepsRelbowStart	27	149	13	150	14	148	1.762
8	grabFloorR	16	269	8	291	8	247	1.132
9	grabHighR	29	259	14	242	15	274	2.524
10	hopBothLegs1hops	36	96	18	97	18	95	1.863
11	hopLLeg1hops	41	75	20	75	21	75	1.893
12	hopRLeg1hops	42	74	21	75	21	73	1.573
13	jogLeftCircle4StepsRstart	17	244	8	245	9	243	1.692
14	jogOnPlaceStartFloor2StepsRStart	14	115	7	114	7	116	0.761
15	jogRightCircle4StepsRstart	17	237	8	236	9	239	1.182
16	jumpDown	14	282	7	298	7	267	1.272
17	jumpingJack1Reps	52	144	26	145	26	144	3.134
18	kickLFront1Reps	29	215	14	210	15	219	1.913
19	kickLSide1Reps	26	233	13	243	13	223	1.682
20	kickRFront1Reps	30	221	15	227	15	215	1.693
21	kickRSide1Reps	30	223	15	248	15	197	1.753
22	lieDownFloor	20	655	10	649	10	661	2.473
23	punchLFront1Reps	30	198	15	199	15	197	2.003
24	punchLSide1Reps	30	175	15	185	15	165	1.812
25	punchRFront1Reps	30	215	15	215	15	215	2.022
26	punchRSide1Reps	28	180	14	195	14	166	1.642
27	rotateArmsBothBackward1Reps	16	109	8	106	8	112	0.841
28	rotateArmsBothForward1Reps	16	116	8	113	8	120	0.721
29	rotateArmsLBackward1Reps	16	108	8	104	8	112	0.751
30	rotateArmsLForward1Reps	16	108	8	102	8	113	0.761
31	rotateArmsRBackward1Reps	16	105	8	102	8	109	0.741
32	rotateArmsRForward1Reps	16	106	8	101	8	112	0.611
33	runOnPlaceStartFloor2StepsRStart	15	85	7	85	8	85	0.611

Tabelle A.1: *Fortsetzung nächste Seite*

Tabelle A.1: *Fortsetzung*

Nr.	Bewegungsklasse $\mathcal{C}$	N	av #f	N_E	av #f_E	N_T	av #f_T	Zeit [s]
34	shuffle2StepsRStart	13	258	6	279	7	240	0.872
35	sitDownChair	20	319	10	310	10	328	1.472
36	sitDownFloor	20	408	10	394	10	422	1.713
37	sitDownKneelTieShoes	17	646	8	638	9	653	2.164
38	sitDownTable	20	271	10	268	10	273	1.592
39	skier1RepsLstart	30	141	15	140	15	143	1.702
40	sneak2StepsLStart	16	236	8	241	8	230	1.162
41	sneak2StepsRStart	16	279	8	298	8	259	1.182
42	squat1Reps	52	193	26	195	26	191	4.126
43	staircaseDown3Rstart	15	223	7	228	8	218	1.051
44	staircaseUp3Rstart	28	296	14	282	14	310	2.393
45	standUpLieFloor	20	525	10	530	10	520	2.193
46	standUpSitFloor	20	403	10	437	10	370	1.712
47	throwBasketball	14	408	7	435	7	380	1.222
48	turnLeft	30	196	15	198	15	194	1.633
49	turnRight	30	197	15	195	15	199	1.953
50	walk2StepsLstart	31	155	15	149	16	162	2.173
51	walk2StepsRstart	31	166	15	178	16	154	1.802
52	walkBackwards2StepsRstart	15	217	7	207	8	226	1.192
53	walkLeft2Steps	16	323	8	328	8	317	1.202
54	walkLeftCircle4StepsRstart	18	328	9	324	9	333	1.322
55	walkOnPlace2StepsLStart	15	153	7	151	8	156	0.832
56	walkRightCircle4StepsRstart	15	329	7	318	8	339	1.222
57	walkRightCrossFront2Steps	16	336	8	338	8	333	1.251
	Gesamtzahl	1329	294613	655	146952	674	147661	90.081

Tabelle A.1: *Ende*

A.2 HDM05_cut_20 oder auch $\mathcal{D}^{20}$

Tabelle A.2: *N - die Gesamtzahl der Bewegungen, av #f - Durchschnittliche Anzahl Frames pro Bewegung, N_E - Anzahl der Bewegungen in der Evaluationsdatenbank, av $\#f_E$ - Durchschnittliche Anzahl Frames pro Bewegung in der Evaluationsdatenbank, N_T - die Anzahl der Bewegungen in der Trainingsdatenbank, av $\#f_T$ - Durchschnittliche Anzahl Frames pro Bewegung in der Trainingsdatenbank, Zeit - Klassentrainingszeit in Sekunden*

Nr.	Bewegungsklasse $\mathcal{C}$	N	av $\#f$	N_E	av $\#f_E$	N_T	av $\#f_T$	Zeit [s]
1	cartwheelLHandStart1Reps	21	411	10	398	11	422	1.913
2	depositFloorR	32	363	16	434	16	293	2.184
3	depositHighR	28	245	14	223	14	268	2.334
4	elbowToKnee1RepsLelbowStart	27	147	13	139	14	154	1.612
5	grabFloorR	16	269	8	291	8	247	1.081
6	grabHighR	29	259	14	242	15	274	2.604
7	jogLeftCircle4StepsRstart	17	244	8	245	9	243	1.712
8	jogRightCircle4StepsRstart	17	237	8	236	9	239	1.222
9	jumpingJack1Reps	52	144	26	145	26	144	3.045
10	kickLFront1Reps	29	215	14	210	15	219	1.773
11	kickLSide1Reps	26	233	13	243	13	223	2.413
12	punchLFront1Reps	30	198	15	199	15	197	1.973
13	punchLSide1Reps	30	175	15	185	15	165	1.773
14	rotateArmsLBackward1Reps	16	108	8	104	8	112	1.051
15	rotateArmsLForward1Reps	16	108	8	102	8	113	0.751
16	sneak2StepsLStart	16	236	8	241	8	230	1.092
17	walk2StepsLstart	31	155	15	149	16	162	1.902
18	walkBackwards2StepsRstart	15	217	7	207	8	226	1.132
19	walkLeftCircle4StepsRstart	18	328	9	324	9	333	1.282
20	walkRightCircle4StepsRstart	15	329	7	318	8	339	1.181
	Gesamtzahl	481	108212	236	53436	245	54776	34.030

Tabelle A.2: *Ende*

Anhang B

Matlab Implementation

Zur Berechnung der Ergebnisse und aller Zwischenschritte wurde MatLab Version 6.5 [6] verwendet. Für die Berechnung von GMMs kam die Netlab Neural Network Toolbox für Matlab von Ian Nabney und Christopher Bishop [12] zum Einsatz. Zur Bewegungsklassifikation mittels KL-EMD Monte Carlo Sampling diente Elias Pampalks MA Toolbox für Matlab [14] erweitert um eine Implementation der Earth Movers Distance von Yossi Ruber [16] (Version heruntergeladen im Januar 2007). Die Berechnung der HMMs erfolgte über die Hidden Markov Model (HMM) Toolbox für Matlab von Kevin Murphy [11].

B.1 GMM Komponenten der Implementation

```
function Y_out = trainDatabase_gmm(features, nclusters, sDB, bVerb, bFixState)
% Y = trainDatabase_gmm(featureSet, nclusters, db, verbose, fixState)
%
% Learns a GMM for every class in a motion capture database.
%
% INPUT
%   featureSet ... (struct) feature set and feature type
%   nclusters .... number of components of the GMM
%   db ........... optional, database to use
%                  '20' -> D20 HDM05_cut_57 (default)
%                  '57' -> D57 HDM05_cut_57
%   verbose ...... (boolean) optional,
%                  true  -> with text output (default)
%                  false -> without text output
%   fixState ..... (boolean) optional,
%                  true -> uses a fixed state for random initializations so that
%                          results are comparable
%                  false -> random initializations
%
% OUTPUT
%   Y ...... (struct, [nclasses x 1]) with fields
%     .class ......... (string) name of the motion class
%     .fileIndizes ... (vector) indices of used files in the class
%     .features ...... (struct) the feature set and feature type
%     .data .......... (matrix, [dim x nframes]) data used for GMM computation
%     .model ......... (struct) GMM representing the motion class
%
```

```
% EXAMPLE
%   Y = trainDatabase_gmm(selectFeatures('jointAngle'), 7, '20', false, false)
%

function GMM = learnGMM_kmeans(theData, nclusters, bVerbose, bFixState, iMaxIt)
% GMM = learnGMM_kmeans(theData, nclusters, verbose, fixState, maxIter)
%
% Learns a GMM of the input data, using a kmeans algorithm to calculate the GMM
% components (component centres, variances and weights).
%
% INPUT
%   theData ...... (matrix) containing the data used to compute the GMM
%                  calculated
%   nclusters .... number of components of the GMM
%   verbose ...... (boolean) optional,
%                  true .... text output (default)
%                  false ... no text output
%   fixedState ... (boolean) optional,
%                  true .... uses a fixed state for random initializations
%                            so that results are comparable
%                  false ... random initializations (default)
%   maxIter ...... (integer) optional, number of iterations for the kmeans
%                  clustering. Default value is 100
%
% OUTPUT
%   GMM .......... (struct) GMM derived from the input data
%

function [score, probs] = scoreGMM_gmmData(classGMM, motionDataMatrix)
% [score, probs] = scoreGMM_gmmData(classGMM, motionDataMatrix)
%
% Computes a real valued rating ("score") for a classification of the motion re-
% presented by "motionDataMatrix" into the motion class provided by "classGMM"
% by means of the GMM-Data resp. GMM-Clustercentres classifier.
% It is also possible to return a vector ("probabilities") holding the single
% (log-) probabilities for "motionDataMatrix" the arithmetic mean of which re-
% sults in "score".
%
% INPUT
%   classGMM ........... (struct) learned GMM representing a motion class
%   motionDataMatrix ... (matrix) representation of a query motion
%
% OUTPUT
%   score ... score/real valued rating for classifying the motion into the class
%             provided
%   probs ... (vector) holds single (log-) probabilities for motionDataMatrix,
%             the arithmetic mean of which results in the score.
%
% EXAMPLE
%   [score, probs] = scoreGMM_gmmData(classGMM, motionDataMatrix)
%

function score = scoreGMM_klEmd(classGMM, motionGMM)
% score = scoreGMM_klEmd(classGMM, motionGMM)
%
% Computes a real valued rating for a classification of the motion represented
```

```
% by motionGMM into the motion class provided by classGMM by means of the KL-EMD
% classifier.
%
% INPUT
%   classGMM .... (struct) learned GMM representing a motion class
%   motionGMM ... (struct) representation of a query motion
%
% OUTPUT
%   score ... score/real valued rating for a classification of the motion into
%             the class provided
%
% EXAMPLE
%   score = scoreGMM_klEmd(classGMM, motionGMM)
%

function score = scoreGMM_mcSamp(classGMM, motionGMM)
% score = scoreGMM_mcSamp(classGMM, motionGMM)
%
% Computes a real valued rating for a classification of the motion represented
% by motionGMM into the motion class provided by classGMM by means of the Monte
% Carlo sampling classifier.
%
% INPUT
%   classGMM .... (struct) learned GMM representing a motion class
%   motionGMM ... (struct) representation of a query motion
%
% OUTPUT
%   score ... score/real valued rating for a classification of the motion into
%             the class provided
%
% EXAMPLE
%   score = scoreGMM_mcSamp(classGMM, motionGMM)
%

function visualizeGMM(data, model, plotOpt, figure_handle)
% visualizeGMM(data, model, plotOpt, figure_handle)
%
% Only for 2d data and model! Plots the model as heightfield in contour
% plot or 3d.
%
% INPUT
%   data ............ (matrix) containing the data used to compute a model
%   model ........... (struct) the data's GMM (or simply a GMM)
%   plotOpt ......... (string) plot options, possible choices:
%                     '3df' -> 3d plot
%                     '3dc' -> contour plot
%   fig_hand ... (optional) handle to a figure into which the result is plotted
%
% OUTPUT
%   n/a
%

function visualizeGMM_clusters(data, model, features)
% visualizeGMM_clusters(data, model, features)
%
% Only for 2d and 3d data and model! Plots the "data" from which a model was
```

```
% computed clustered by the model's centers. Also plots the model's centers, co-
% variances and weights (priors).
%
% INPUT
%   data ....... (matrix) containing the data from which a model was computed
%   model ...... (struct) the data's GMM
%   features ... (struct) the data's feature set and feature type
%
% OUTPUT
%   n/a
%
```

B.2 HMM Komponenten der Implementation

```
function Y_out = trainDatabase_hmm(features, nclusters, nstates, sCovType, ...
    sDB, sInitType, bVerbose, bFixedState)
% Y = trainDatabase_hmm(featureSet, nclusters, nstates, cov_type, db, ...
%   init_type, vb, fixedState)
%
% Learns a HMM for every class in a motion capture database.
%
% INPUT
%   featureSet ... (struct) feature set and feature type
%   nclusters .... number of components of each state GMM
%   nstates ...... number of states of the HMM
%   cov_type ..... optional default 'spherical'
%   db ........... optional, database to use
%                  '20' -> D20 HDM05_cut_57 (default)
%                  '57' -> D57 HDM05_cut_57
%   init_type .... (string) optional, method of initialization. Options are
%                  'kmeans-count' estimates mu and sigma by kmeans and counts
%                     state transitions and init states in order to predict
%                     transmat and pi.
%                  'kmeans' calculates mu and sigma via kmeans and initializes
%                     pi, transmat and mixmat equally distributed (default).
%                  'random' initializes mu at random, sigma at some fixed value,
%                     pi, transmat and mixmat equally distributed
%   vb ........... (boolean) optional,
%                  true  => with text output (default)
%                  false => without text output
%   fixedState ... (boolean) optional,
%                  true  => uses a fixed state for random initializations
%                           so that results are comparable
%                  false => no fixed state is used (default)
%
% OUTPUT
%   Y ...... (struct, [nclasses x 1]) with fields
%     .class ...... (string) name of the motion class
%     .features ... (struct) the feature set and feature type
%     .data ....... (matrix, [dim x nframes]) data used for HMM computation
%     .model ...... (struct) HMM representing the motion class
%
% EXAMPLE
%   Y = trainDatabase_hmm(selectFeatures('jointAngle'), 1, 7, 'spherical', '20')
```

```
%

function model = hmm(dim, ncentres, nstates, sCovType, varargin)
% model = hmm(dim, ncentres, nstates, cov_type, init_type)
%
% Creates a hidden Markov model with specified architecture. Parameters are
% neither initialized nor optimized! It creates an _empty_ model.
%
% INPUT
%   dim ......... (integer) dimension of the data to be represented by the hmm
%   ncentres .... (integer) number of centres per state
%   nstates ..... (integer) number of states
%   cov_type .... (string) covariance structure of each state. Options are
%                 'spherical' ... single variance parameter for each state
%                 'diagonal' .... diagonal matrix for each state
%                 'full' ........ full matrix for each state
%   init_type ... (string) type of initialization. Possible options are
%                 'kmeans-count' estimates mu and sigma by kmeans and counts
%                    state transitions and init states in order to predict
%                    transmat and pi (default).
%                 'kmeans' calculates mu and sigma via kmeans and initializes
%                    pi, transmat and mixmat equally distributed.
%                 'random' initializes mu at random, sigma at some fixed value,
%                    pi, transmat and mixmat equally distributed
%
% OUTPUT
%   model ... (struct) representing the HMM with fields
%               type: 'hmm'
%                nin: the dimension of the data
%            nstates: the number of states
%           ncentres: the number of centres per state
%         covar_type: string for type of variance model
%                 pi: [Qx1 double] initial state probability
%           transmat: [QxQ double] transition matrix
%                 mu: [dxQxM double] means for each state and each centre
%              sigma: [dxdxQxM double] covariances of states
%             mixmat: [QxM double] mixing coefficients per state
%
% where d = dim
%       Q = nstates
%       M = ncentres
%
% Example:
%   model = hmm(11, 1, 7, 'spherical')
%   model = hmm(11, 1, 7, 'spherical', 'kmeans-count')
%

function [mu, sigma, pi, transmat, mixmat] = initializeHMM(theData, nstates, ...
    nclusters, sMethod, bVerbose, bFixedState, iMaxIter)
% [mu, sigma, pi, transmat, mixmat] = initializeHMM(theData, nstates, nclusters,
%     method, verbose, fixedState, maxIter)
%
% Computes an initial HMM for the Baum-Welch algorithm (HMM-training).
%
% INPUT
%   theData ...... (cell or double array) the data from which the model parame-
```

```
%                  ters should be initialized
%   nstates ...... (integer) number of states of the HMM
%   nclusters .... (integer) number of clusters per state
%   method ....... (string) method of initialization. Options are
%                  'kmeans-count' estimates mu and sigma by kmeans and counts
%                     state transitions and init states in order to predict
%                     transmat and pi.
%                  'kmeans' calculates mu and sigma via kmeans and initializes
%                     pi, transmat and mixmat equally distributed.
%                  'random' initializes mu at random, sigma at some fixed value,
%                     pi, transmat and mixmat equally distributed
%   verbose ...... (boolean) if true: print out kmeans error (default: false)
%   fixedState ... (boolean) if true: use fixed state for calculations
%                  (default: false)
%   maxIter ...... (integer) maximum itereations of kmeans (default: 100)
%
% OUTPUT (model parameters)
%   mu ......... (double array) [dxQxM] cluster centres (means)
%   sigma ...... (double array) [dxdxQxM] covariance matrix
%   pi ......... (double array) [Qx1] initial state distribution
%   transmat ... (double array) [QxQ] transition matrix
%   mixmat ..... (double array) [QxM] mixture matrix
% where d is the dimension of the data
%       Q is the number of states (nstates)
%       M is the number of clusters per state (nclusters)
%

function HMM = learnHMM_gmmKmeans(theData, nclusters, nstates, sCovType, ...
    bVerbose, bFixedState, sMethod, iMaxIter)
% HMM = learnHMM_gmmKmeans(theData, nclusters, nstates, cov_type, vb, ...
%   fixedState, init_type, maxIter)
%
% Learns a HMM of the input data, using an EM-Algorithm to calculate the
% final HMM components (component wise GMMs, transition matrix, init states).
%
% INPUT
%   theData ...... (1 x nTrainingFiles cell array) containing the data from
%                  which the HMM will be calculated
%   nclusters .... number of components of each state GMM
%   nstates ...... number of states of the HMM
%   cov_type ..... (string) covariance structure of each state. Choices are
%                  'spherical' ... single variance parameter for each state
%                  'diag' ........ diagonal matrix for each state
%                  'full' ........ full matrix for each state
%   vb ........... (boolean) verbose, optional,
%                  true => text output (default)
%   fixedState ... (boolean) optional,
%                  true .... uses a fixed state for random initializations so
%                            that results are comparable
%                  false ... random initializations (default)
%   init_type .... (string) type of initialization. Possible options are
%                  'kmeans-count' estimates mu and sigma by kmeans and counts
%                     state transitions and init states in order to predict
%                     transmat and pi (default).
%                  'kmeans' calculates mu and sigma via kmeans and initializes
%                     pi, transmat and mixmat equally distributed.
```

```
%                  'random' initializes mu at random, sigma at some fixed value,
%                     pi, transmat and mixmat equally distributed
%   maxIter ...... (integer) optional, number of iterations for the kmeans
%                  clustering. Default value is 20
%
% OUTPUT
%   HMM .......... (struct) HMM derived from the input data
%
% EXAMPLE
%   HMM = learnHMM_gmmKmeans(someData, 1, 7, 'spherical', 0, 1, 'kmeans', 10)
%

function varargout = scoreHMM_fwd(classHMM, motionInp)
% [score, probs] = scoreHMM_fwd(classHMM, motionData)
% score = scoreHMM_fwd(classHMM, motionData)
%
% Computes a real valued rating ("score") for a classification of the query mo-
% tion represented by "motionData" into the motion class provided by "classHMM"
% using the forward algorithm.
% It is also possible to return a vector ("probabilities") holding the single
% probabilities for "motionData".
%
% INPUT
%   classHMM ........ (struct) learned HMM representing a motion class
%   motionData ...... (matrix or cell) representation of a query motion
%
% OUTPUT
%   score ... score/real valued rating for a classification of the motion into
%             the class provided
%   probs ... (vector) holds single probabilities for motionData.
%
% EXAMPLE
%   [score, probs] = scoreHMM_fwd(classHMM, motionData)
%

function varargout = scoreHMM_vit(classHMM, motionInp)
% [score, path] = scoreHMM_vit(classHMM, motionData)
% score = scoreHMM_vit(classHMM, motionData)
%
% Computes a real valued rating ("score") for a classification of the query mo-
% tion represented by "motionData" into the motion class provided by "classHMM"
% using viterbi decoding.
% It is also possible to return a vector ("path") holding the viterbi path the
% "motionData" takes through the model ("classHMM").
%
% INPUT
%   classHMM ..... (struct) learned HMM representing a motion class
%   motionData ... (matrix or cell) representation of a query motion
%
% OUTPUT
%   score ... score/real valued rating for a classification of the motion into
%             the class provided
%   path .... (vector) holds the motionData's viterbi path.
%
% EXAMPLE
%   [score, path] = scoreHMM_vit(classHMM, motionData)
```

```
%

function varargout = experiments_multipleClassInit(scClassNames, sDB, sFeatures)
% newDB = experiments_multipleClassInit(classNames, db, features)
%
% Initializes classes as specified in classNames together before splitting them
% up again in training.
%
% INPUT
%   classNames ... (c x 1 cell) where each row contains a class tuple to be
%                  initialized together and the number of rows specifies the to-
%                  tal of joined initializations. Class names may be wildcarded.
%   db ........... (string, optional) specifies the database to be used. Choices
%                  are ... '20' to use HDM_05_cut_20 [default] and
%                      ... '57' for HDM_05_cut_57
%   features ..... (string, optional) specifies the feature set. Choices are
%                  'jointangle' [default]
%                  'relational' for relational features and
%                  'bk' for physically based features by Björn Krüger
%
% OUTPUT
%   newDB ...... (struct) containing the database with changes in the desired
%                classes. Usually has fields
%         .Y_trained ...... (struct) contains trained databas
%         .Y_evaluation ... (struct) contains evaluation database
%         .groundTruth .... (matrix) holding the true classifications
%         .pred_hmmData ... (matrix) predictin matrix holding the actual
%                           classifications
%
% EXAMPLE
% experiments_multipleClassInit({{'rotateArms*'}}, '20', 'jointAngle')
%


function visualizeHMM_clusters(data, model, features, figure_handle)
% visualizeHMM_clusters(data, model, features, fig_handle)
%
% Only for 2d and 3d data and model! Plots the "data" from which a model was
% computed as well as the model's centers and covariances.
%
% INPUT
%   data ......... (matrix) data from which a model was computed
%   model ........ (struct) the data's HMM
%   features ..... (struct) the data's feature set and feature type
%   fig_handle ... handle to existing figure (optional)
%
%  OUTPUT
%   n/a
%

function visualizeHMM_obsProb(data, model)
% visualizeHMM_obsProb(data, model)
%
% Visualizes state and observation probabilities for the data given the model.
% State probabilities are shown on the left-hand side, observation probabilities
% on the right-hand side.
```

```
%
% INPUT
%   data .... (matrix, nfeatures x nframes) observation data for which the state
%             probability shall be shown
%   model ... (struct) a Hidden Markov Model
%
% OUTPUT
%   n/a
%

function visualizeHMM_viterbipath(data, model)
% visualizeHMM_viterbipath(data, model)
%
% Visualizes the viterbi path (bottom) which the data would take through the mo-
% del as well as the data itself (top).
%
% INPUT
%   data .... (matrix, nfeatures x nframes) observation data for which the state
%             sequence shall be shown
%   model ... (struct) a Hidden Markov Model
%
% OUTPUT
%   n/a
%

function visualizeHMM_viterbipath_twoModels(data, model1, model2)
% visualizeHMM_viterbipath_twoModels(data, model01, model02)
%
% Visualizes the viterbi paths (bottom) which the data would take through each
% of the models as well as the data itself (top).
%
% INPUT
%   data ...... (matrix, nfeatures x nframes) observation data for which the
%               state sequences shall be shown
%   model01 ... (struct) first Hidden Markov Model
%   model02 ... (struct) second Hidden Markov Model
%
% OUTPUT
%   n/a
%
```

B.3 Weitere Komponenten der Implementation

```
function [skel, mot] = readMocapAMC(s_fullFilePath)
% [skel, mot] = readMocapAMC(fullFilePath)
%
% reads out resp. parses an AMC-Mocap file.
%
% INPUT
%   fullFilePath ... (string) full file path of the file to be parsed
%
% OUTPUT
%   skel ... (struct) skeleton file
%   mot .... (struct) motion data
%
```

```
function [sFeatures, aDescription] = selectFeatures(strFeature, iWhich)
% features = selectFeatures(feature, which)
%
% Returns a struct containing the feature set for the selected feature and the
% feature type.
%
% INPUT
%   feature ... (string) feature type, possible choices:
%               'jointAngle' for joint angles,
%               'relational' for relational features,
%               'bk' for physical features (E_kin)
%   which ..... (vector) optional. If specified it restricts the feature set to
%               these entries of the complete set.
%
% OUTPUT
%   features ....... (struct) feature type and feature set
%           .featureType ... (string) the feature type
%           .featureSet .... (matrix or cell array) the feature set
%
% EXAMPLE
%   features = selectFeatures('bk')
%   features = selectFeatures('jointAngle', [1,3,5])
%

function theData = dataAcquisition(skel, mot, features, bVerbose)
% theData = dataAcquisition(skel, mot, featureSet, verbose)
%
% Feature extraction resp. data acquisition, based on feature set and features
% type with optional text output.
%
% INPUT
%   skel ......... (struct) containing the skeleton data (from parser)
%   mot .......... (struct) containing the motion data (from parser)
%   featureSet ... (struct) feature set and feature type
%   verbose ...... (boolean) optional,
%                  true .... with text output (default)
%                  false ... without text output
%
% OUTPUT
%   theData ...... (matrix, [dim x ndata]) containing the data
%
% EXAMPLE
%    theData = dataAcquisition(skel, mot, selectFeatures('jointAngle'))
%

function Y_out = compileEvaluationDatabase(feats, nclusters, db, bVerbose, ...
    bFixedState, bDataOnly)
% Y = compileEvaluationDatabase(featSet, nclusters, db, vb, fixState, dataOnly)
%
% Extracts data of every motion in the (evaluation) motion capture database. If
% "dataOnly" is set "false" this function additionally computes a GMM for each
% motion.
%
% INPUT
%   featSet ..... (struct) feature set and feature type
%   nclusters ... number of components of the GMM
```

```
%   db .......... optional, database to use
%                 '20' ... D20, HDM05_cut_20 (default)
%                 '57' ... D57, HDM05_cut_57
%   vb .......... (boolean) optional,
%                 true .... with text output (default)
%                 false ... without text output
%   fixState .... (boolean) optional,
%                true .... uses a fixed state for random initializations so
%                          that results are comparable
%                false ... random initialization (default)
%   dataOnly .... (boolean) optional,
%                true .... calculation of GMM will be omitted
%                false ... a GMM will be computed for each motion (default)
%
% OUTPUT
%   Y ...... (struct, [nclasses x 1]) with fields
%     .class ...... (string) name of the motion class
%     .features ... (struct) the feature set and feature type
%     .data ....... (matrix, [dim x nframes]) data used to compute the GMM
%     .model ...... (struct) GMM representing the motion
%
% EXAMPLE
%   Y = compileEvaluationDatabase(selectFeatures('jointAngle'), 7, '20', 0, 0)
%

function predMat = computePredictionMatrix(DB_train, DB_eval, sClassifier)
% predMat = computePredictionMatrix(DB_trained, DB_evaluation, classifier)
%
% Computes a prediction matrix for a certain "classifier".
%
% INPUT
%   DB_trained ... (struct, [nclasses x 1]) holding class information for a
%                  training database as output by "trainDatabase_[g,h]mm"
%   DB_eval ...... (struct, [nqueries x 1]) holding motion information for an
%                  evaluation database as output by "compileEvaluationDatabase"
%   classifier ... (string) type of classification to be used for a
%                  prediction matrix. Possible choices are
%                  'gmm-data' ...... GMM vs data classifier
%                  'gmm-centres' ... GMM vs cluster centres classifier
%                  'kl-emd' ........ KL-EMD classifier
%                  'mc-samp' ....... Monte Carlo sampling classifier
%                  'hmm' ........... HMM classifier
%
% OUTPUT
%   predMat ... (matrix) a prediction matrix holding predictions into which
%               class (colums) a certain query (rows) would be classified into
%

function groundTruth = computeGroundTruthMatrix(Y_trained, Y_evaluation)
% trueClassMat = computePredictionMatrix(DB_training, DB_evaluation)
%
% Computes the true classification for a trained database ("DB_training") and an
% evaluation database ("DB_evaluation").
%
% INPUT
%   DB_training ..... (struct, [nclasses x 1]) holding class information for a
```

```
%                     training database as output by "trainDatabase_[g,h]mm"
%   DB_evaluation ... (struct, [nqueries x 1]) holding motion information for an
%                     evaluation database as output by compileEvaluationDatabase
%
% OUTPUT
%   trueClassMat ... (matrix) a classification matrix where the correct class is
%                    indicated by 1 wrong classes by 0
%

function [C, rate] = computeConfusionMatrix(predMat, trueMat)
% [confMat, rate] = computeConfusionMatrix(predMat, trueMat)
%
% Computation of the confusion matrix of a prediction matrix and its true
% classification ("confMat") and of the overall classification rate
% ("rate").
%
% INPUT
%   predMat ... (matrix) a prediction matrix holding predictions into which
%                class (colums) a certain query (rows) would be classified into.
%   trueMat ... (matrix) the true classification for a query into a class
%
% OUTPUT
%   confMat ... (matrix) confusion matrix
%   rate ...... classification rate
%

function visualizePredictionMatrix(predMat, sOption)
% visualizePredictionMatrix(predMat, plotopt)
%
% Plots a prediction matrix or a relative prediction matrix.
%
% INPUT
%   predMat ... (matrix) a prediction matrix holding predictions into which
%                class (colums) a certain query (rows) would be classified into
%   plotopt.... (string) optional
%                'relative' a relative prediction matrix will be plotted, each
%                    column being mapped to [0, 1]
%                '' image-plot of the prediction matrix
%
% OUTPUT
%   n/a
%

function visualizeConfusionMatrix(probMat, trueMat)
% visualizeConfusionMatrix(probMat, trueMat)
%
% Computes and plots the confusion matrix and overall classification rate of a
% prediction matrix and its true classification.
%
% INPUT
%   probMat ... (matrix) prediction matrix holding predictions into which class
%                (colums) a certain query (rows) would be classified into.
%   trueMat ... (matrix) the true classification for a query into a class
%
% OUTPUT
%   n/a
%
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] J.-J. AUCOUTURIER AND F. PACHET, *Improving timbre similarity: How high's the sky?*, Journal of Negative Research Results in Speech and Audio Sciences, vol. 1 (2004). 18, 19

[2] J. A. BILMES, *What hmms can do*, IEICE Transactions, 89-D (2006), pp. 869–891. 48

[3] B. KRÜGER, J. TAUTGES, M. MÜLLER, AND A. WEBER, *Multi-mode tensor representation of motion data*, Journal of Virtual Reality and Broadcasting, (2008). To appear. 10

[4] B. KRÜGER, J. TAUTGES, AND A. WEBER, *Multi-mode representation of motion data*, in The 2nd International Conference on Computer Graphics Theory and Applications (GRAPP 2007), Volume AS/IE, INSTICC Press, March 2007, pp. 21–29. 10

[5] B. LOGAN AND A. SALOMON, *A music similarity function based on signal analysis*, in Proceedings of the 2001 IEEE International Conference on Multimedia and Expo, ICME 2001, Tokyo, August 22-25, 2001. 18

[6] MATLAB, *The MathWorks Deutschland*. http://www.mathworks.de, 2006. 71

[7] M. MÜLLER, *Information Retrieval for Music and Motion*, Springer, 2007. 8, 9

[8] M. MÜLLER AND T. RÖDER, *Motion templates for automatic classification and retrieval of motion capture data*, in Proceedings of the 2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation, 2006, pp. 137–146. 8

[9] M. MÜLLER, T. RÖDER, AND M. CLAUSEN, *Efficient content-based retrieval of motion capture data*, ACM Trans. Graph., 24 (2005), pp. 677–685. 8

[10] M. MÜLLER, T. RÖDER, M. CLAUSEN, B. EBERHARDT, B. KRÜGER, AND A. WEBER, *Documentation mocap database HDM05*, Technical report, No. CG-2007-2, (2007). 34

[11] K. MURPHY, *Hidden markov model (HMM) toolbox for matlab*. Matlab Toolbox. 48, 71

[12] I. NABNEY AND C. BISHOP, *Netlab neural network software*. Matlab Toolbox. 71

[13] I. T. NABNEY, *Netlab: Algorithms for Pattern Recognition*, Springer, 2004. 13

[14] E. PAMPALK, *MA toolbox for matlab - implementing similarity measures for audio*. Matlab Toolbox. 18, 71

[15] L. R. RABINER, *A tutorial on hidden markov models and selected applications in speech recognition*, Proceedings of the IEEE, vol. 77 (1989), pp. 257–286. 41, 44, 45

[16] Y. RUBNER, *An implementation of the earth movers distance*. http://vision.stanford.edu/ rubner/emd/default.htm. 18, 71

[17] Y. RUBNER, C. TOMASI, AND L. J. GUIBAS, *The earth mover's distance as a metric for image retrieval*, International Journal of Computer Vision, vol. 40 (2000), pp. 99–121. 19

Danksagung

Ich möchte den folgenden Personen für ihre große Hilfe und Unterstützung bei der Anfertigung dieser Arbeit danken:

Der gesamten Arbeitsgruppe von *Prof. Dr. Michael Clausen* für die wunderbare Betreuung. Insbesondere bedanke ich mich bei Dr. Meinard Müller für seinen großen Einsatz, die hilfreiche Kritik und konstruktiven Anregungen.

Björn Krüger danke ich für Gedankenaustausch und Korrekturvorschläge,

meinen Eltern für ihre Unterstützung meiner Vorhaben in jeglicher Hinsicht und ein erstes Korrekturlesen,

und meiner kleinen Schwester *Daniela* dafür, dass sie mich in letzter Zeit ertragen hat und Unnötiges von mir fernhielt.

Thomas Haas danke ich für kleinere themenbezogene Unterhaltungen und Diskussionen,

Sonja Kretschmer für gewissenhaftes Korrekturlesen.

Schließlich bedanke ich mich bei all meinen Freunden dafür, dass sie meine Abwesenheit insbesondere in den letzten drei Wochen ertragen haben.

Zuletzt spreche ich einen Dank an meinen Rechner aus, der die ganze Zeit durchgehalten hat.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe.

Bonn, den 2. Mai 2008

Katharina Stollenwerk