Lehrstuhl für Mensch-Maschine-Kommunikation
Technische Universität München

Graphische Modelle in der Mustererkennung

Marc A. Al-Hames

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Olaf Stursberg
Prüfer der Dissertation:	1. Univ.-Prof. Dr. Gerhard Rigoll
	2. Prof. Dr. Horst Bunke, Universität Bern / Schweiz

Die Dissertation wurde am 23.10.2007 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 26.03.2008 angenommen.

Kurzfassung

Graphische Modelle verbinden die Wahrscheinlichkeits- und die Graphentheorie: Mit Knoten und Kanten werden statistische Abhängigkeiten zwischen Variablen ausgedrückt. Dadurch können Probleme intuitiv erfasst und häufig mit geringer Rechenkomplexität gelöst werden.

In dieser Arbeit wird untersucht, ob und wie Graphische Modelle für verschiedene Mustererkennungsprobleme verwendet werden können: Zur kombinierten Erkennung von Schnittpunkten und Szenenwechseln in Videos wird ein zweistufiges Modell entwickelt. Bimodale Benutzereingaben werden mit einem asynchronen Modell fusioniert. Um Gruppenaktionen in Konferenzen aus fehlerbehafteten Daten robust zu erkennen, wird ein multimodales Hidden Markov Modell mit einem linearen dynamischen System verbunden. Für einen automatischen Zusammenschnitt von Konferenzvideos werden Graphische Modelle, die Daten segmentieren und klassifizieren können, verwendet und Trainingsstrukturen entworfen.

Die entwickelten Graphischen Modelle werden theoretisch analysiert, Berechnungsvorschriften und Lernalgorithmen abgeleitet, die jeweiligen Erkennungsleistungen experimentell evaluiert und mögliche Erweiterungen aufgezeigt.

Inhaltsverzeichnis

1

Einleitung

Graphische Modelle (GM) verbinden die Wahrscheinlichkeits- mit der Graphentheorie [82]. Mit Knoten und Kanten werden statistische Abhängigkeiten zwischen Variablen modelliert. Dadurch können die Zusammenhänge zwischen den Variablen eines Problemes zum einen intuitiv erfasst [23] und zum anderen die Probleme selbst häufig mit geringerer Rechenkomplexität als auf dem direkten Weg gelöst werden [81].

Viele bekannte Verfahren der Signalverarbeitung und der Mustererkennung – wie z. B. das Gauß Mixtur Modell [25], das Hidden Markov Modell [126] oder lineare dynamische Systeme [66] – können auch als GM beschrieben werden. Die effizienten Berechnungsvorschriften für diese Modelle – wie z. B. der Vorwärts-Rückwärts-Algorithmus [14], die Kalman Filter Formeln [85] oder die Rauch Rekursion [128] – sind dann Spezialfälle der allgemeinen Nachrichtenpropagation in GM. Dadurch können die Gemeinsamkeiten und Unterschiede von Verfahren verschiedener Teildisziplinien verglichen werden, Methoden von speziellen Verfahren auch auf andere Verfahren angewendet werden, insbesondere aber verschiedene Verfahren miteinander verbunden werden, um neuartige Methoden zu entwickeln. Dabei werden GM häufig auch zum prototypischen Beschreiben und Entwerfen von Lösungen verwendet; die allgemeine Nachrichtenpropagation erlaubt dann ein standardisiertes Ableiten von effizienten Berechnungsvorschriften für das entworfene Modell.

GM bieten die Möglichkeit beim Entwurf nicht das Problem an das Modell, sondern das Modell an das Problem anzupassen. Durch diese Gestaltungsmöglichkeiten können GM für verschiedene Probleme eingesetzt werden: Anwendung fanden sie bisher vorallem zur Beschreibung von Expertensystemen [79], in der Bildverarbeitung [161] und zum automatischen Auswerten großer Datenbestände [34]. Seit kurzem werden sie auch vermehrt in der Mustererkennung eingesetzt [30, 112]. Insbesondere wurden dabei bisher verschiedene Modelle für die automatische Spracherkennung entwickelt und erfolgreich angewendet [29, 63].

Ziel dieser Arbeit ist es, sowohl theoretisch als auch mit Experimenten zu untersuchen, ob und wie GM für verschiedene andere Mustererkennungsprobleme eingesetzt werden können.

Insbesondere werden Anwendungen mit mehreren Datenströmen – für die eine multimodale Fusion erforderlich ist – untersucht. Anhand von vier verschiedenen Anwendungen werden unterschiedliche Aspekte von GM eingeführt, vorgestellt und untersucht: Unter anderem werden das problemangepasste Modellieren, die asynchrone Behandlung von Daten, das Kombinieren verschiedener Modelle, die approximative Berechnung sowie die kombinierte Segmentierung und Klassifizierung eingesetzt.

Für jedes Problem wird in dieser Arbeit ein neues GM entworfen, die dafür notwendigen Berechnungsvorschriften und Lernalgorithmen abgeleitet und die jeweilige Leistungsfähigkeit experimentell überprüft und mit anderen Modellen verglichen. Die einzelnen Beiträge dieser Arbeit sind daher folgende:

Kapitel 2 bietet eine umfassende Einführung in GM. Dabei wird eine einheitliche, durchgängige Notation für Probleme und Modelle verschiedener Teildisziplinen eingeführt. Die allgemeinen Berechnungsvorschriften in GM werden hergeleitet. Es wird außerdem gezeigt, wie sich eine Vielzahl verschiedener Modelle als GM abbilden und wie sich davon neue problemangepasste Modelle ableiten lassen. Für einige bekannte Modelle der Mustererkennung wird detailliert gezeigt, wie sich effiziente Berechnungsvorschriften aus der allgemeinen Berechnung in GM herleiten lassen und welche Gemeinsamkeiten zwischen verschiedenen Modellen bestehen.

In Kapitel 3 wird ein neues, zweistufiges GM entwickelt, um Schnittpunkte und Szenenwechsel in Videos zu detektieren. Das Modell integriert signalnahe und semantisch höherwertige Merkmale in zwei Ebenen und optimiert damit die Anordnung der Schnitte und Szenen im Verbund.

Kapitel 4 behandelt ein multimodales GM, das Asynchronitäten in der multimodalen Fusion modelliert: Für das asynchrone Hidden Markov Modell [17] wird die Vorwärtsberechnung durch dynamische Bereichsgrenzen im Trellis-Diagramm verbessert. Dadurch kann die Komplexität sowohl für das Training als auch für die Dekodierung signifikant reduziert werden. Durch diese Komplexitätsreduktion kann das asynchrone Hidden Markov Modell erstmalig auch auf die Fusion von bimodalen Benutzereingabedaten angewendet werden, dies war bisher aufgrund der hohen Asynchronitäten bei Benutzereingaben nicht möglich.

In Kapitel 5 wird ein neues multimodales GM zur robusten Erkennung von Gruppenaktionen in Besprechungen entwickelt. Dazu wird ein Hidden Markov Modell mit einem linearen dynamischen System verbunden. Beide Modelle interagieren während der Erkennung miteinander, um Störungen in den Daten auszugleichen. Allerdings ist das GM wegen seiner hohen Komplexität praktisch nicht mehr berechenbar. Daher wird für das Modell ein approximatives Verfahren zum Berechnen der notwendigen Wahrscheinlichkeiten vorgestellt.

In Kapitel 6 werden GM so erweitert, dass Daten gleichzeitig segmentiert und klassifiziert werden können. Ein aus der Spracherkennung bekanntes, lineares GM [29] wird zu einer links-rechts- und einer ergodischen Struktur erweitert. Eine bekannte Struktur zum unüberwachten Training der Segmentgrenzen wird durch eine neue Modellebene so erweitert, dass auch ein überwachtes Training der Segmentgrenzen für alle bekannten GM möglich ist. Die entwickelten GM werden dann angewendet, um aus Konferenzaufnahmen mit mehreren Kameras ein einzelnes Video zusammenzuschneiden.

Die Arbeit behandelt verschiedene Anwendungen der Mensch-Maschine-Kommunikation und der Mustererkennung, um verschiedene Aspekte der GM aufzuzeigen und zu untersuchen. Dabei wird in den ersten drei Kapiteln eine umfassende und genaue Erklärung der Modelle geboten; insbesondere werden die Berechnungsverfahren sowie das Lernen der Modellparameter ausführlich behandelt. In den Kapiteln 5 und 6 werden die Modelle teilweise nur noch auf graphischer Ebene behandelt. Es werden nur Besonderheiten der Berechnungen im Detail hervorgehoben.

Damit folgt der Aufbau der Arbeit auch der prinzipiellen Funktionsweise von GM. Sind das Grundprinzip und die Möglichkeiten und Grenzen der vorhandenen Algorithmen bekannt, ist kein Detailwissen mehr erforderlich, da die Algorithmen für jedes Modell formal gleich abgeleitet werden können. Eine Problemmodellierung findet dann nur noch auf der graphischen Ebene in sehr abstrakter Form statt: Lösungsansätze werden nur noch „als Graph skizziert“, dabei werden statistische Konsequenzen der Modellierung sichtbar. Die zugrundeliegenden Wahrscheinlichkeitsverteilungen für das entworfene Modell und die dafür benötigten Algorithmen ergeben sich dann durch den Formalismus automatisch.

2

Graphische Modelle

GM sind eine Mischung aus Graphen- und Wahrscheinlichkeitstheorie [82]: Bedingte Unabhängigkeitsaussagen von Variablen werden auf Graphen übertragen; dadurch erhält der Graph Markov-Eigenschaften. Ein bestimmtes GM repräsentiert dann alle Wahrscheinlichkeitsverteilungen, die diese Unabhängigkeitsaussagen erfüllen [23]. Verschiedene Arten von GM können verwendet werden, um unterschiedliche Unabhängigkeitsaussagen auszudrücken.

Viele probabilistische Probleme werden dann durch einfache Graphenalgorithmen gelöst. Durch die formalen Markov-Eigenschaften der Graphen ist dabei jedoch stets sichergestellt, dass die zugrundeliegenden Wahrscheinlichkeitsaussagen eingehalten werden. Eine Berechnung in einem GM führt damit zum gleichen Ergebnis wie die direkte Berechnung über die Wahrscheinlichkeitstheorie, ist jedoch häufig signifikant geringer in der benötigten Zeitkomplexität und zusätzlich fast immer wesentlich intuitiver.

Durch die Verbindung von Graphen- und Wahrscheinlichkeitseigenschaften haben GM fünf Haupteigenschaften, die nach den englischen Anfangsbuchstaben auch als die SALAD-Properties beschrieben werden [24]:

Struktur: Eine intuitive, leicht zugängliche Methode zum Untersuchen von Strukturen in natürlichen Zusammenhängen, z. B. von kausalen oder verknüpften Ereignissen [124].

Algorithmen: Standardisierte Verfahren zum Entscheiden unter Unsicherheit, z. B. dem probabilistischen Schließen oder dem Berechnen von Auftrittswahrscheinlichkeiten bei gegebenen Beobachtungen [81, 123].

Sprache: Eine streng formale, abstrakte, mathematische, aber durch die Graphentheorie auch intuitive Sprache zum Beschreiben von kausalen und probabilistischen Problemen. Dadurch können verschiedene Verfahren der Mustererkennung und Wahrscheinlichkeitstheorie durch die einheitliche Sprache der GM zusammengefasst werden [94].

Approximation: Wenn eine vollständige Berechnung zu komplex wird, bieten GM standardisierte Verfahren zum approximativen Schließen. Durch die Struktur, die standardisierten Algorithmen und die Sprache der GM können dabei Aussagen für den durch die Approximation entstehenden Fehler gemacht werden [77, 83, 102].

Datenbank: Zusammenfassend stellen GM eine Art probabilistische Datenbank dar. Das Lernen von Struktur und Parametern entspricht dem Speichern von Information in der Datenbank. Algorithmen zum probabilistischen Schließen entsprechen dann Anfragen an die Datenbank. Dabei können durch die formale Sprache alle Anfragen – problemunabhängig – durch einheitliche, wahlweise auch approximative Algorithmen behandelt werden [23].

GM erlauben daher einen einheitlichen Blickwinkel auf verschiedene Probleme. Durch diese Vereinheitlichung wird auch eine gemeinsame Beschreibung und häufig ein Zusammenfassen der Methoden ermöglicht (wie z. B. im Rahmen dieser Arbeit die Verbindung eines Kalman-Filters mit einem Hidden Markov Modell (HMM) in Kapitel 5). Weiterhin können viele Algorithmen, die speziell für ein bestimmtes Problem ausgelegt sind, als Spezialfälle der allgemeineren GM-Algorithmen beschrieben werden (z. B. der HMM Vorwärts-Rückwärts-Algorithmus [14] oder die Vorwärtsrekursion des Kalman-Filters [85]). Dadurch werden Gemeinsamkeiten sowie Vor- und Nachteile verschiedener Verfahren unterschiedlicher Teildisziplinen vergleichbar.

Weiterhin sind Graphen eine natürliche und intuitive Darstellungsform von Problemen [82]. Dadurch wird sowohl die Beschreibung neuer Algorithmen als auch das Verständnis für verschiedene Probleme erhöht. Vereinfachungen können so leichter abgeleitet werden. Die einheitliche Struktur sowie die intuitive Graphennotation der GM erlauben eine einfache Formulierung von Problemen und Lösungen. Häufig können GM daher zum schnellen Entwickeln und Beschreiben von neuen Ideen verwendet werden. Die standardisierten Algorithmen erlauben dann ein sofortiges Überprüfen der Entwürfe; Konsequenzen bestimmter Annahmen werden sofort ersichtlich. GM können daher auch als eine Art rapid prototyping für Algorithmen angesehen werden. Die so entworfenen Methoden können für viele Probleme sofort eingesetzt werden. Es ist jedoch auch möglich, in späteren Verfeinerungsschritten von den generellen GM-Standardalgorithmen speziellere Methoden abzuleiten, die nicht mehr unbedingt allgemeingültig sind (in dieser Arbeit z. B. die Komplexitätsreduktion des asynchronen HMM in Kapitel 4).

Die Theorie der GM baut fast ausschließlich auf Pearls Forschungsergebnissen zu bedingter Unabhängigkeit und Nachrichtenübertragung in Graphen [122, 123, 125] auf und wurde später formal von Lauritzen zusammengefasst [94]. Grundlegend einführende mathematische GM Werke [46, 57, 143, 159] behandeln vorwiegend verschiedene Nachrichtenübertragungsverfahren in Graphen, deren Effizienz sowie Beweise, unter welchen Umständen die Nachrichtenübertragung exakt ist. Weitere Arbeiten

versuchen auch die Theorie neuronaler Netze [84] oder der unscharfen (*fuzzy*) Logik [34] in den GM Formalismus einzubringen. Angewendet werden GM heute vorwiegend in der Bildverarbeitung [161], in Expertensystemen [79] oder zum Auswerten großer Datenbestände (*data mining*) [34]. Erst seit dem Aufkommen von Software-Werkzeugen, die auch die Behandlung von dynamischen Daten ermöglichen – wie z. B. Murphys Bayes Net Toolbox (BNT) [112] oder Bilmes Graphical Model Toolkit (GMTK) [30] –, finden GM auch vermehrt Anwendung in der Mustererkennung.

In diesem Kapitel soll eine Einführung in GM geboten werden. Dabei soll insbesondere auf die Mustererkennung eingegangen werden. Diese Einführung kann nicht auf alle Details eingehen, Beweise und Spezialfälle werden vernachlässigt. Jedoch soll die Methodik der GM erläutert und wichtige GM-Algorithmen abgedeckt werden. Es soll gezeigt werden, wie Mustererkennungsverfahren durch GM abgebildet werden und wie sich wichtige Algorithmen direkt aus den GM-Algorithmen ergeben.

Dazu werden zuerst die Graphentheorie (Abschn. 2.1) und die bedingte Unabhängigkeit (Abschn. 2.2) zusammengefasst. Dann wird auf die zwei wichtigsten Arten von GM eingegangen: ungerichtete Graphen, die als Markov Random Fields (MRF) bezeichnet werden (Abschn. 2.3) und gerichtete Graphen, als Bayes'sche Netze (BN) bezeichnet (Abschn. 2.4). Weiterhin wird auf die dynamische Erweiterung der BN zum Modellieren von Zeitreihen eingegangen (Abschn. 2.5). Die Nachrichtenübermittlung in GM wird anhand des Verbundbaum-Algorithmus (*junction tree*) hergeleitet (Abschn. 2.8), mit dem alle Berechnungen in GM vorgenommen werden können. Es wird außerdem auf das Lernen von Daten eingegangen (Abschn. 2.9). Anhand von Beispielmodellen wird dann gezeigt, wie GM in der Mustererkennung angewendet werden können. Dabei wird vor allem der Zusammenhang zwischen traditioneller Beschreibung (z. B. dem Vorwärts-Rückwärts-Algorithmus) und der GM Beschreibung hergestellt (Abschn. 2.10).

2.1 Graphentheoretische Grundbegriffe

Im Rahmen dieser Arbeit wird die Graphennotation von Lauritzen [94] verwendet[1], die sich für GM etabliert hat [23]. Ein **Graph** ist das geordnete Paar von der endlichen Menge der **Knoten** (*vertices*) $V = (V_0, \ldots, V_N)$ und der endlichen Menge der **Kanten** (*edges*) E und wird mit

$$\mathcal{G} = (V, E) \tag{2.1}$$

gekennzeichnet. Dabei ist die Menge der Kanten eine Teilmenge der Produktmenge aller geordneten Paare $E \subseteq V \times V$. Weiterhin sei der Graph $\mathcal{G}$ **schlicht** (*simple*): die

[1] Diese Notation weicht in einigen Teilen von der Graphentheorie ab. Soweit vorhanden, werden deutsche Begriffe nach [86] oder [90] verwendet, die wiederum häufig aus dem Französischen [21] stammen. Weiterhin werden englische Abkürzungen nach [23, 94] verwendet, auch wenn deutsche Übersetzungen teilweise vorhanden sind. Dies soll den Vergleich mit der Literatur vereinfachen.

Kante zwischen den Knoten $(V_i, V_j) \in E, \forall V_i, V_j \in V$ kommt in $\mathcal{G}$ maximal einmal vor (keine Parallelkanten) und keine Kante hat Start- und Endpunkt am selben Knoten (schleifenfrei), d. h. $(V_i, V_i) \notin E, \forall V_i \in V$. Die Abbildungen 2.1 auf der nächsten Seite und 2.2 auf Seite 11 zeigen einige solcher Graphen und veranschaulichen die folgenden graphentheoretischen Beziehungen der Knoten; dabei wird jeder Knoten durch einen Kreis und jede Knotenmenge durch eine Wolke dargestellt.

Eine Kante heißt **einfache** oder **gerichtete Kante** (*directed edge*), wenn $(V_i, V_j) \in E$, aber $(V_j, V_i) \notin E$, sie wird mit $V_i \rightarrow V_j$ gekennzeichnet. Eine Kante bei der sowohl $(V_i, V_j) \in E$ und $(V_j, V_i) \in E$ wird als **zweifache** oder **ungerichtete Kante** (*undirected edge*) bezeichnet und mit $V_i \sim V_j$ gekennzeichnet.

Die Knoten V_i und V_j sind **Nachbarn** (*neighbors*), wenn $V_i \sim V_j$. Alle Nachbarn von V_i werden in der Menge der Nachbarn $\mathrm{ne}(V_i) = \{V_j : V_i \sim V_j\}$ zusammengefasst. Die Nachbarn der Knotenmenge $V' \subset V$ sind alle Nachbarn aller Knoten in V', solange diese Nachbarn nicht selbst in V' enthalten sind, d. h.

$$\mathrm{ne}(V') = \bigcup_{V_i \in V'} \mathrm{ne}(V_i) \backslash V' = \bigcup_{V_i \in V'} \{V_j : V_i \sim V_j\} \backslash V'. \tag{2.2}$$

Der Knoten V_i ist ein **Vorgänger** oder **Elternknoten** (*parent*) von V_j, wenn $V_i \rightarrow V_j$. Der Knoten V_j wird dann auch als **Nachfolger** oder **Kindknoten** (*child*) von V_i bezeichnet. Alle Eltern von V_j werden in der Menge der Eltern $\mathrm{pa}(V_j) = \{V_i : V_i \rightarrow V_j\}$ zusammengefasst. Analog werden alle Kinder von V_i in der Menge der Kinder $\mathrm{ch}(V_i) = \{V_j : V_i \rightarrow V_j\}$ zusammengefasst. Die Eltern der Knotenmenge $V' \subset V$ sind alle Eltern aller Knoten in V', solange diese Eltern nicht selbst in V' enthalten sind, d. h.

$$\mathrm{pa}(V') = \bigcup_{V_i \in V'} \mathrm{pa}(V_i) \backslash V' = \bigcup_{V_i \in V'} \{V_j : V_j \rightarrow V_i\} \backslash V'. \tag{2.3}$$

Analog sind die Kinder der Knotenmenge $V' \subset V$ alle Kinder aller Knoten in V', solange diese Kinder nicht selbst in V' enthalten sind, d. h.

$$\mathrm{ch}(V') = \bigcup_{V_i \in V'} \mathrm{ch}(V_i) \backslash V' = \bigcup_{V_i \in V'} \{V_j : V_i \rightarrow V_j\} \backslash V'. \tag{2.4}$$

Der **Rand** (*boundary*) einer Knotenmenge $V' \subset V$ sind alle Nachbarn und Eltern von V'

$$\mathrm{bd}(V') = \mathrm{pa}(V') \cup \mathrm{ne}(V'). \tag{2.5}$$

Die **geschlossene Hülle** oder der **Abschluss** (*closure*) einer Knotenmenge $V' \subset V$ ist der Rand von V' und alle Knoten in V'

$$\mathrm{cl}(V') = \mathrm{bd}(V') \cup V' = \mathrm{pa}(V') \cup \mathrm{ne}(V') \cup V'. \tag{2.6}$$

Die **Markov Decke** (*Markov blanket*) eines Knotens V_i in einem Graphen sind alle Eltern und Kinder von V_i sowie alle Eltern der Kinder:

$$\mathrm{bl}(V_i) = \mathrm{pa}(V_i) \cup \mathrm{ch}(V_i) \cup \{V_j : \mathrm{ch}(V_j) \cap \mathrm{ch}(V_i) \neq \emptyset\}. \tag{2.7}$$

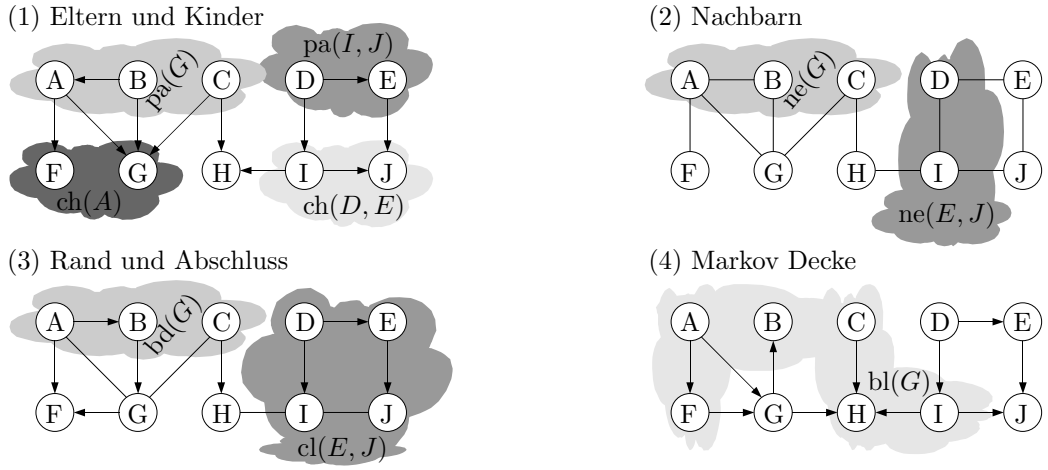

Abbildung 2.1: Verdeutlichung einiger graphentheoretischer Grundbegriffe. (1) zeigt Eltern- und Kindbeziehungen in einem gerichteten Graphen. (2) zeigt Nachbarn in einem ungerichteten Graphen. (3) zeigt den Rand und den Abschluss und (4) eine Markov Decke.

Zwischen den Knoten V_i und V_j gibt es einen **gerichteten Weg** (*path*) der Länge n, wenn es die Kantenfolge $(V_i, V_1), (V_1, V_2), \ldots, (V_{n-1}, V_j)$ mit $(V_{k-1}, V_k) \in E, \forall k$ im Graphen gibt. Dann **führt** (*leads*) V_i zu V_j, gekennzeichnet durch $V_i \mapsto V_j$. Wenn es sowohl einen gerichteten Weg von V_i zu V_j als auch einen gerichteten Weg von V_j zu V_i gibt, sind die beiden Knoten **verbunden** (*connected*), dies wird mit $V_i \rightleftharpoons V_j$ gekennzeichnet; die Verbindung selbst wird dann als **Weg** (*trail*) bezeichnet. Eine **Kette** (*chain*) zwischen den Knoten V_i und V_j ist eine Verbindung zwischen den beiden Knoten, bei dem die Richtung der Kanten keine Rolle spielt; also $(V_i, V_1), (V_1, V_2), \ldots, (V_{n-1}, V_j)$ mit (V_{k-1}, V_k) oder $(V_k, V_{k-1}) \in E, \forall k$. Ein Weg, der im selben Knoten V_i startet und endet, wird als **Kreis** oder **Zyklus** (*cycle*) bezeichnet. Eine **Sehne** (*chord*) ist eine Kante zwischen zwei nicht benachbarten Knoten eines Zyklus, womit die Sehne selbst nicht Bestandteil des Zyklus ist. Ein Zyklus mit n Knoten, der keine Sehne enthält, wird mit C_n bezeichnet.

Ein Graph $\mathcal{G}^{\rightarrow}$, der nur gerichtete Kanten enthält, wird als **antisymmetrischer** oder **gerichteter Graph** (*directed graph*) bezeichnet. Ein gerichteter Graph, der keinen Zyklus enthält, wird als **gerichteter, zyklusfreier Graph** (*directed acyclic graph*) (DAG) bezeichnet und mit $\mathcal{G}^{D}$ gekennzeichnet. Ein Graph $\mathcal{G}^{\sim}$, der nur ungerichtete Kanten enthält, wird als **symmetrischer** oder **ungerichteter Graph** (*undirected graph*) bezeichnet. Aus einem gerichteten Graphen $\mathcal{G}^{\rightarrow}$ wird ein **moralisierter Graph** (*moralised graph*) $\mathcal{G}^{m}$, indem zu jeder gerichteten Kante $(V_i, V_j) \in E$ die gegenläufige Kante (V_j, V_i) zu den Kanten E hinzugefügt wird und weiterhin die Menge der Elternknoten $\mathrm{pa}(V_i)$ für jeden Knoten $V_i \in V$ durch eine ungerichtete Kante untereinander verbunden wird, d. h. (V_i, V_j) und (V_j, V_i) mit

$V_i, V_j \in \mathrm{pa}(V_k), \forall V_k \in V$ wird zu der Menge der Kanten E hinzugefügt. Bildlich gesprochen entsteht $\mathcal{G}^m$ aus $\mathcal{G}^{\rightarrow}$, indem alle gerichteten Kanten durch ungerichtete ersetzt werden und für jeden Knoten des Graphens seine Eltern untereinander durch ungerichtete Kanten miteinander verbunden werden.

Die **Vorfahren** (*ancestors*) eines Knotens V_i sind alle Knoten V_j, von denen es einen gerichteten Weg zu V_i gibt, aber keinen gerichteten Weg von V_i zu V_j:

$$\mathrm{an}(V_i) = \{V_j : V_j \mapsto V_i, V_i \not\mapsto V_j\}. \tag{2.8}$$

Analog sind die **Nachfahren** (*descendants*) eines Knotens V_i alle Knoten V_j, für die es einen gerichteten Weg von V_i zu V_j, aber keinen gerichteten Weg von V_j zu V_i gibt:

$$\mathrm{de}(V_i) = \{V_j : V_j \not\mapsto V_i, V_i \mapsto V_j\}. \tag{2.9}$$

Die **Nicht-Nachfahren** (*non-descendants*) eines Knotens V_i sind alle Knoten des Graphens ohne die Nachfahren und ohne V_i:

$$\mathrm{nd}(V_i) = V \backslash (\mathrm{de}(V_i) \cup V_i). \tag{2.10}$$

Eine Knotenmenge V' ist eine **Vorfahrenmenge** (*ancestral set*), wenn alle Eltern und Nachbarn aller Knoten in V' wiederum in V' enthalten sind, d. h. $V' \subseteq V$ mit $\mathrm{bd}(V_i) \subseteq V', \forall V_i \in V'$. In einem gerichteten Graphen $\mathcal{G}^{\rightarrow}$ ist V' eine Vorfahrenmenge, wenn $\mathrm{an}(V_i) \subseteq V', \forall V_i \in V'$. In einem ungerichteten Graphen $\mathcal{G}^{\sim}$ ist V' eine Vorfahrenmenge, wenn alle Knoten, die mit allen Knoten aus V' verbunden sind, selbst in V' enthalten sind. Also ist V' eine Vorfahrenmenge, wenn $\nexists V_i \rightleftharpoons V_j$ mit $V_i, V_j \notin V'$. Die kleinste mögliche Vorfahrenmenge, die V' enthält, wird als Ancestral Hull bezeichnet und mit $\mathrm{An}(V')$ gekennzeichnet.

Die Knotenmenge $V' \subseteq V$ ist ein **Knoten-Separator** (*separator*) der Knoten V_i und V_j, wenn alle Wege von V_i nach V_j durch V' gehen, d. h. $\mathcal{G}' = (V \backslash V', [(V \backslash V') \times (V \backslash V')] \cap E), \nexists V_i \mapsto V_j$. Die Knotenmenge $V' \subseteq V$ ist ein **Separator** (*separator*) der Knotenmengen V^a und V^b, wenn V' ein Knoten-Separator zwischen allen Knoten aus V^a und allen Knoten aus V^b, d. h. $\mathcal{G}' = (V \backslash V', [(V \backslash V') \times (V \backslash V')] \cap E), \nexists V_i \mapsto V_j, \forall V_i \in V^a, V_j \in V^b$. Ein Separator V' ist **minimal**, wenn das Entfernen eines beliebigen Knotens V_i aus V' dazu führt, dass $V' \backslash V_i$ kein Separator mehr ist.

Der **Teil-** oder **Untergraph** (*vertex induced subgraph*) $\mathcal{G}' = (V', E')$ eines Graphens $\mathcal{G}$ wird durch die Knotenuntermenge $V' \subseteq V$ und allen Kanten aus E, die sowohl den Start- als auch den Endpunkt in V' haben, gebildet: $\mathcal{G}' = (V', E')$ mit $V' \subseteq V$ und $E' = E \cap [V' \times V']$, jeder Graph ist damit auch sein eigener Untergraph. Ein Graph ist **vollständig** (*complete*), wenn alle Knoten V untereinander mit einer gerichteten oder ungerichteten Kante verbunden sind, d. h. $(V_i, V_j) \in E$ oder $(V_j, V_i) \in E, \forall V_i, V_j \in V$ und $i \neq j$. Eine **Clique** bzw. ein **vollständiger Untergraph** (*clique, complete subgraph*) ist ein Untergraph von $\mathcal{G}$, bei dem alle Knoten untereinander verbunden sind. Ein vollständiger Untergraph $\mathcal{G}'$, der auch

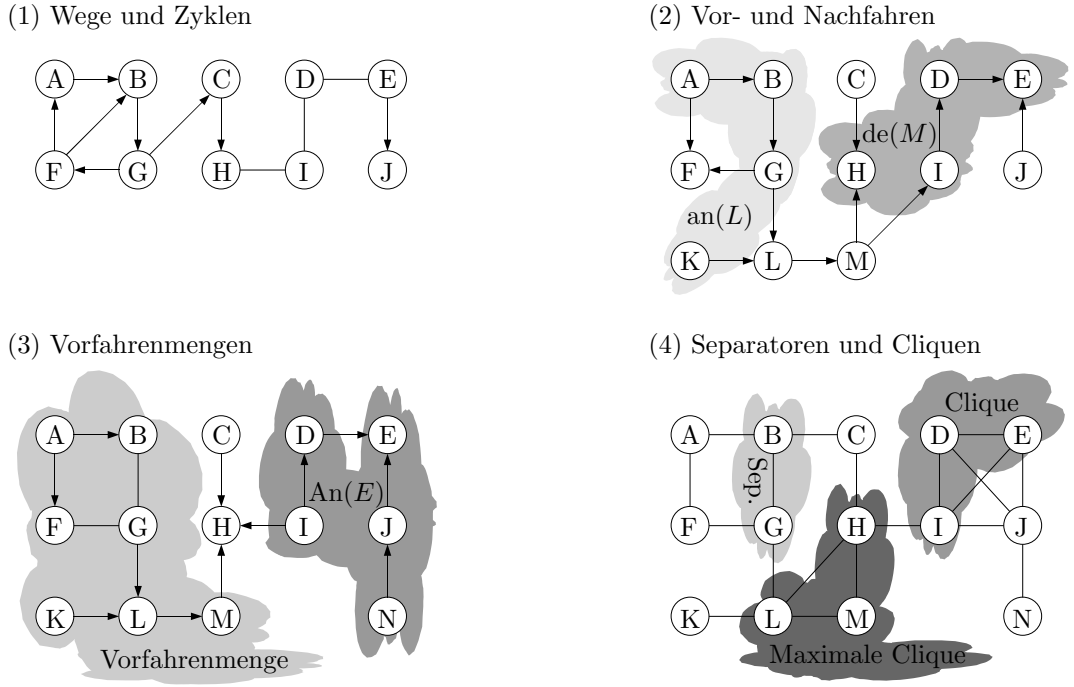

Abbildung 2.2: In (1) führt von A ein gerichteter Weg zu H ($A \mapsto H$). Die Knoten H und E sind verbunden ($H \rightleftharpoons E$). $A \to B \to G \to F \to A$ ist ein Zyklus, unterbrochen durch die Sehne $F \mapsto B$. (2) zeigt Vor- und Nachfahren und (3) die dazugehörigen Mengen. In (4) ist $\{B, G\}$ ein Separator der Knoten A und C. $\{I, D, E\}$ ist eine Clique und $\{H, L, M\}$ eine maximale Clique.

maximal ist, wird als **maximale Clique** bzw. als **maximal vollständiger Untergraph** (*maxclique*)[2] bezeichnet. Dabei bezieht sich die Maximalität darauf, dass das Hinzufügen eines beliebigen weiteren Knotens aus V zu V' dazu führen würde, dass der Untergraph $\mathcal{G}'$ nicht mehr vollständig wäre; d. h. mit $V' \subset V$, $V'' \subset V$ und $V' \cap V'' = \emptyset$ gilt, dass $(V_i, V_j) \in E$ oder $(V_j, V_i) \in E, \forall V_i, V_j \in V'$ und $i \neq j$, aber $\exists V_i, V_j \in V' \cup V''$, so dass sowohl $(V_i, V_j) \notin E$ und $(V_j, V_i) \notin E$.

Ein Graph ohne Zyklus ist ein **Wald** (*forest*). Ein Wald, in dem von jedem Knoten jeder andere Knoten erreicht werden kann, heißt **Baum** (*tree*). Die **Blätter** (*leaf*) eines Baumes haben nur einen Nachbarn. In einem Baum, in dem vom Knoten V_i zu jedem anderen Knoten V_j genau ein gerichteter Weg führt, heißt **Wurzelbaum** (*rooted tree*). Der Knoten V_i ist dann die Wurzel. Aus einem ungerichteten Baum wird ein Wurzelbaum, indem ein beliebiger Knoten V_i als Wurzel ausgewählt wird und dann alle Kanten von dieser Wurzel weggerichtet werden.

[2] In [94] bezeichnet Lauritzen nur einen maximal vollständigen Untergraphen auch als Clique. Die Bezeichnung Maxclique wird nicht verwendet. Sowohl in der Graphentheorie als auch in vielen Veröffentlichungen [23, 26, 113] zu GM wird jedoch zwischen Clique und Maxclique unterschieden.

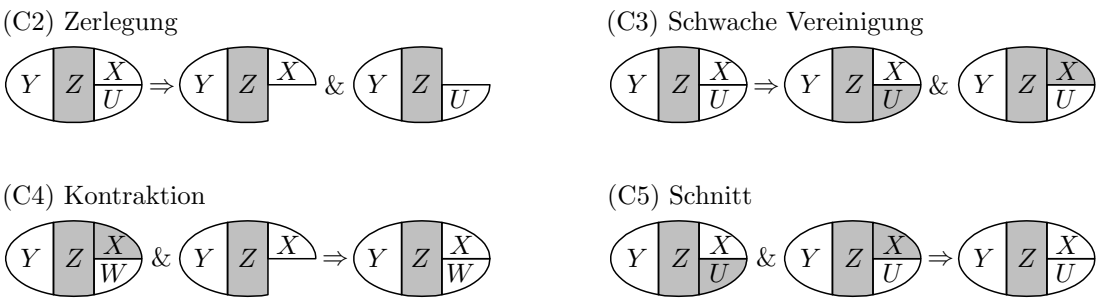

Abbildung 2.3: Unabhängigkeitseier nach Pearl [123]: Eine graphische Interpretation der bedingten Unabhängigkeitsaxiome (C2) - (C5). Die Symmetrie ist trivial und daher nicht gezeigt.

2.2 Bedingte Unabhängigkeit

Sei X eine diskrete Zufallsvariable, die definiert ist für den endlichen Wertebereich $x_i \in \mathcal{X}$. Sei $p(X = x_i) \rightarrow \mathbb{R}$ das Wahrscheinlichkeitsmaß, dass X den Wert x_i annimmt. In dieser Arbeit wird dabei auch die Kurznotation $p(x_i) = p(X = x_i)$ verwendet. Analog sei $p(x^a) = p(x_1, \ldots, x_N)$ das Wahrscheinlichkeitsmaß, dass die Zufallsvariablen $X^A = \{X_1, \ldots, X_N\}$ die Werte $X_1 = x_1, \ldots, X_N = x_N$ annehmen. Weiterhin kann $p(x_i)$ auch für kontinuierliche Zufallsvariablen X definiert werden. Es gelte[3], dass $0 \leq p(x_i) \leq 1, \forall x_i$. Weiterhin gelte, dass $\sum_{x_i \in \mathcal{X}} p(x_i) = 1$ bzw. $\int_{x_i \in \mathcal{X}} p(x_i) \mathrm{d}x = 1$. Damit sei $p(x_i)$ eine Wahrscheinlichkeitsdichtefunktion (WDF).

Seien X, Y und Z drei Zufallsvariablen. X und Y sind **bedingt unabhängig** (*conditional independent*) [49], bei gegebener Variable Z, wenn

$$p(x_i \,|\, y_j, z_k) = p(x_i \,|\, z_k), \tag{2.11}$$

für alle Kombinationen von $x_i \in \mathcal{X}$, $y_j \in \mathcal{Y}$ und $z_k \in \mathcal{Z}$. Das heißt, wenn Z bekannt ist, enthält Y keine weitere Information über X und ist daher irrelevant zur Berechnung der Wahrscheinlichkeit $p(x_i \,|\, y_j, z_k)$, dies wird gekennzeichnet mit

$$X \perp\!\!\!\perp Y \,|\, Z. \tag{2.12}$$

Diese bedingte Unabhängigkeit hat fünf für GM relevante Eigenschaften [124]:

(C1) Symmetrie: $X \perp\!\!\!\perp Y \,|\, Z \Leftrightarrow Y \perp\!\!\!\perp X \,|\, Z$

(C2) Zerlegung: $X \perp\!\!\!\perp Y \,|\, Z$ und $U = f(X) \Rightarrow U \perp\!\!\!\perp Y \,|\, Z$

[3]Für kontinuierliche Zufallsvariablen X ist mathematisch eigentlich $p(x_i) = 0, \forall x_i$, da es sich um einen Punkt einer Wahrscheinlichkeitsdichtefunktion (WDF) handelt. In der Mustererkennung ist es jedoch üblich [25], entweder den Wert der WDF so zu verwenden, als ob es sich um eine Wahrscheinlichkeit handelt oder impliziert über einen Bereich 2ϵ um den gesuchten Punkt x_i herum zu integrieren, d. h. die Wahrscheinlichkeit $p(x_i) := p(x_i - \epsilon \leq X \leq x_i + \epsilon)$ zu verwenden. Beide Ansätze werden in der Praxis erfolgreich verwendet [30, 112, 166].

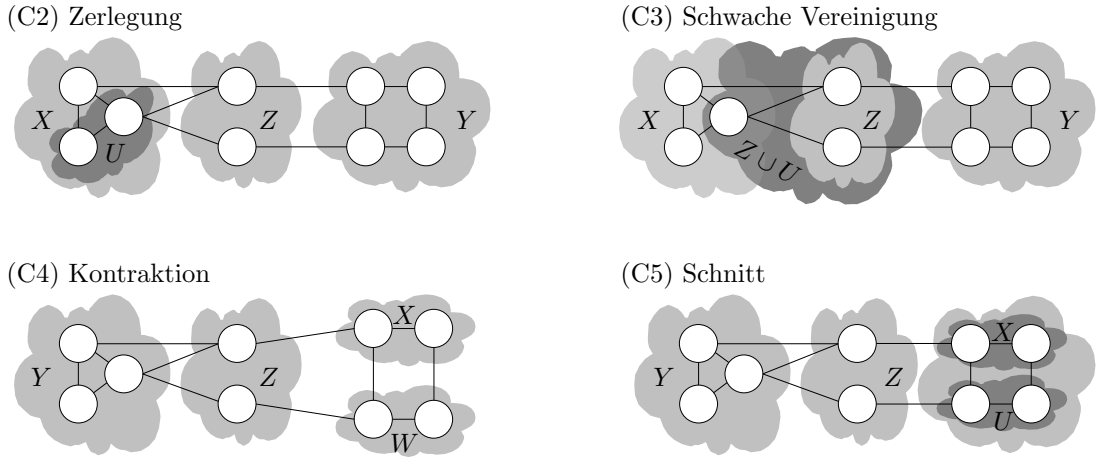

Abbildung 2.4: Beispielhafte Verdeutlichung des graphischen Äquivalents der Axiome (C2) - (C5). Dabei repräsentiert jeder Knoten (dargestellt durch Kreise) eine Zufallsvariable. Knotenmengen sind durch Wolken hinterlegt. Durch die graphische Separation werden (C2) - (C5) implizit erfüllt [23].

(C3) Schwache Vereinigung: $X \perp\!\!\!\perp Y \,|\, Z$ und $U = f(X) \Rightarrow X \perp\!\!\!\perp Y \,|\, (Z, U)$

(C4) Kontraktion: $X \perp\!\!\!\perp Y \,|\, Z$ und $Y \perp\!\!\!\perp W \,|\, (X, Z) \Rightarrow Y \perp\!\!\!\perp (W, X) \,|\, Z$

(C5) Schnitt: $X \perp\!\!\!\perp Y \,|\, (Z, U)$ und $Y \perp\!\!\!\perp U \,|\, (X, Z) \Rightarrow Y \perp\!\!\!\perp (X, U) \,|\, Z$

Abbildung 2.3 zeigt eine Verdeutlichung der Axiome (C2) - (C5) nach Pearl [123]. Die Eigenschaften (C1) - (C4) sind für alle Wahrscheinlichkeitsverteilungen erfüllt [124]. Der Schnitt (C5) ist jedoch nur dann gültig, wenn die zugrunde liegende Wahrscheinlichkeitsverteilung $p(x_i, y_j, z_k) > 0, \forall x_i, y_j, z_k$ strikt positiv ist [23]. Das Axiom (C5) hat daher für Verteilungen mit deterministischen Eigenschaften (bei denen eine Aussage zwangsläufig zu einer bestimmten Aktion führt) keine Gültigkeit.

Die Axiome (C1) - (C5) haben auch ein direktes graphisches Äquivalent: Wenn der Knoten V_x die Zufallsvariable X repräsentiert bzw. die Knotenmenge V^x die Zufallsvariablen $X_1, X_2, \ldots, X_N$, können die Unabhängigkeitsaxiome auch durch graphische Separation dargestellt werden, dies ist in Abbildung 2.4 gezeigt.

Wenn nun bedingte Unabhängigkeitsaussagen derart auf Graphen übertragen werden, erhält man die Verbindung von Wahrscheinlichkeits- und Graphentheorie und damit GM [82]. Dabei erhält der Graph Markov-Eigenschaften, d. h. bedingte Unabhängigkeiten, die der Graph ausdrückt [23]. Abhängig vom Graphen sowie den implizierten Unabhängigkeiten, erhält man verschiedene Arten von GM. Diese werden verwendet, um unterschiedliche bedingte Unabhängigkeitsaussagen auszudrücken. Im Folgenden sollen die Definition, die Markov-Eigenschaften und die sich daraus ergebenden Eigenschaften von ungerichteten GM (sogenannte Markov Random Fields) und gerichteten GM (Bayes'sche Netze) formal hergeleitet werden.

2.3 Markov Random Fields

Markov Random Fields (MRF) [51, 75, 88] sind ungerichtete Graphen, die bedingte Unabhängigkeitsaussagen zwischen Variablen ausdrücken. Jeder Knoten des Graphens repräsentiert eine Zufallsvariable. Eine Kante zwischen zwei Knoten drückt dann aus, dass die den Knoten zugrundeliegenden Variablen miteinander interagieren können (aber nicht müssen). MRF werden hauptsächlich in der Bildverarbeitung eingesetzt [161], jedoch kann z. B. auch ein HMM [126] als ein MRF repräsentiert werden [23]. Um die formale Definition eines MRF herzuleiten, sollen hier zuerst die allgemeinen Markov-Eigenschaften von ungerichteten Graphen erläutert werden.

2.3.1 Markov-Eigenschaften von ungerichteten Graphen

Gegeben seien die Zufallsvariablen $X_{V_1}, \ldots, X_{V_N}$ und ein ungerichteter Graph $\mathcal{G}^{\sim} = (V, E)$. Jedem Knoten V_i des Graphens sei genau eine Zufallsvariable X_{V_i} zugeordnet. Die bedingte Unabhängigkeit der Variablen $X_{V_i} \perp\!\!\!\perp X_{V_j} \,|\, X_{V_k}$ wird im Folgenden durch die Knoten-Kurznotation

$$V_i \perp\!\!\!\perp V_j \,|\, V_k \tag{2.13}$$

ausgedrückt. Die Wahrscheinlichkeitsverteilung $p(x_{V_1}, \ldots, x_{V_N})$ erfüllt [94] dann die

(P) paarweise Markov-Eigenschaft, wenn für ein beliebiges Paar von nicht benachbarten Knoten $V_i \in V$ und $V_j \in V$ mit $V_i \not\sim V_j$ gilt, dass

$$V_i \perp\!\!\!\perp V_j \,|\, V \backslash (V_i, V_j), \tag{2.14}$$

(L) lokale Markov-Eigenschaft, wenn für einen beliebigen Knoten $V_i \in V$ gilt, dass

$$V_i \perp\!\!\!\perp V \backslash \mathrm{cl}(V_i) \,|\, \mathrm{bd}(V_i), \tag{2.15}$$

(G) globale Markov-Eigenschaft, wenn für drei disjunkte Knotenteilmengen $V^a \subset V$, $V^b \subset V$ und $V^s \subset V$, wobei V^s ein Separator von V^a und V^b ist, gilt, dass

$$V^a \perp\!\!\!\perp V^b \,|\, V^s. \tag{2.16}$$

Abbildung 2.5 zeigt diese drei Markov-Eigenschaften anhand von ungerichteten Beispielgraphen. Neben den Markov-Eigenschaften erfüllt die Wahrscheinlichkeitsverteilung $p(x_{V_1}, \ldots, x_{V_N})$ die

(F) Faktorisierung von Graphen, wenn es für jede Clique $C \subseteq V$ des Graphens eine nicht negative Funktion ψ_C gibt, die von $X_{V_1}, \ldots, X_{V_N}$ nur über die in der Clique C enthaltenen Zufallsvariablen $X^C \subseteq \{X_{V_1}, \ldots, X_{V_N}\}$ abhängt und

$$p(x_{V_1}, \ldots, x_{V_N}) = \prod_{c\text{: Cliquen C}} \psi_c(x^c). \tag{2.17}$$

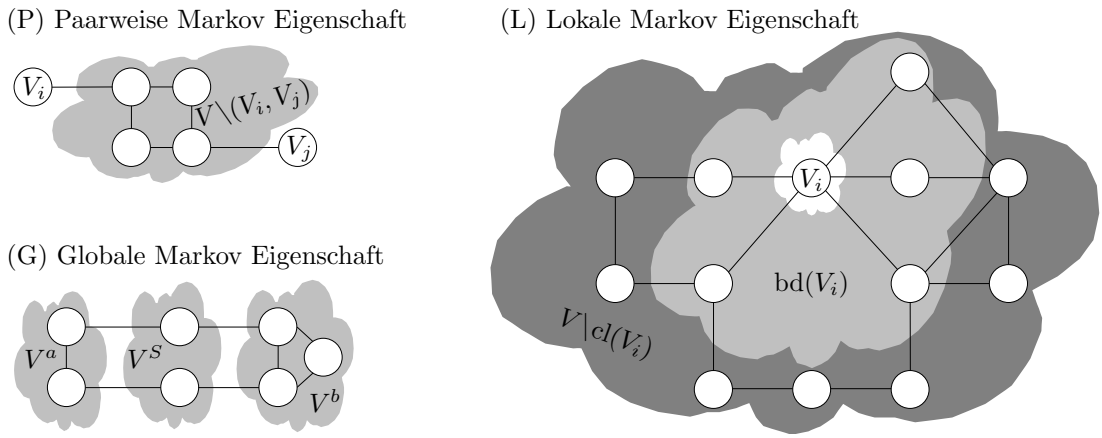

Abbildung 2.5: Paarweise, lokale und globale Markov-Eigenschaften von ungerichteten Graphen.

Dabei sind die Funktionen ψ_c nicht eindeutig bestimmt. In der Regel werden die Cliquen c daher als maximale Cliquen gewählt, wodurch die Beschreibung eindeutig wird. Wenn also $\mathcal{C}$ die eindeutige Menge aller maximalen Cliquen in $\mathcal{G}^{\sim}$ bezeichne, dann gelte für (F), dass

$$p(x_{V_1}, \ldots, x_{V_N}) = \prod_{c \in \mathcal{C}} \psi_c(x^c). \tag{2.18}$$

So gewählte Cliquefunktionen $\psi_c(x^c)$ werden häufig auch als Cliquepotentiale bezeichnet [161]. Für jeden ungerichteten Graphen $\mathcal{G}^{\sim}$, der die Zufallsvariablen X_{V_1}, $\ldots, X_{V_N}$ repräsentiert, gilt das **alphabetische Theorem** [94]:

$$(\mathrm{F}) \Rightarrow (\mathrm{G}) \Rightarrow (\mathrm{L}) \Rightarrow (\mathrm{P}). \tag{2.19}$$

Erfüllt ein Graph die Faktorisierung, so erfüllt er automatisch auch alle Markov-Eigenschaften. Die Umkehrung des Theorems ist nur in bestimmten Fällen, aber nicht im Allgemeinen gültig.

Tatsächlich wird der Graph $\mathcal{G}^{\sim}$ eines MRF daher genau so gewählt, dass die den Zufallsvariablen $X_{V_1}, \ldots, X_{V_N}$ zugrundeliegende Wahrscheinlichkeitsverteilung $p(x_{V_1}, \ldots, x_{V_N})$ durch $\mathcal{G}^{\sim}$ faktorisiert wird. Dann gilt für MRF die Faktorisierung (F). MRF erfüllen daher gemäß dem alphabetischem Theorem automatisch auch die globale (G), die lokale (L) und die paarweise (P) Markov-Eigenschaft. Daraus kann nun die formale Definition eines MRF hergeleitet werden.

2.3.2 Definition von Markov Random Fields

Ein MRF ist ein ungerichteter Graph $\mathcal{G}^{\sim} = (V, E)$, der alle Wahrscheinlichkeitsverteilungen $p(x_{V_1}, \ldots, x_{V_N})$ repräsentiert, die eine Faktorisierung (F) gemäß $\mathcal{G}^{\sim}$ erfüllen [23]. Daher gilt: Wenn $\mathcal{C}$ die Menge aller maximalen Cliquen in $\mathcal{G}^{\sim}$ bezeichnet,

gibt es für jede maximale Clique $c \in \mathcal{C}$ eine positive Funktion, die nur von den in der Clique enthaltenen Variablen X^C abhängt:

$$\psi_c(x^c) \geq 0, \forall c \in \mathcal{C}. \tag{2.20}$$

Weiter ist die Verteilung $p(x_{V_1}, \ldots, x_{V_N})$ faktorisierbar bezüglich $\mathcal{G}^\sim$, d. h.

$$p(x_{V_1}, \ldots, x_{V_n}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(x^c), \tag{2.21}$$

wobei Z eine Normalisierung darstellt:

$$Z = \sum_{x_{V_1}, \ldots, x_{V_n}} \prod_{c \in \mathcal{C}} \psi_c(x^c). \tag{2.22}$$

Ein MRF erfüllt damit gemäß dem alphabetischen Theorem auch die globale (G), die lokale (L) und die paarweise (P) Markov-Eigenschaft.

Durch diese Definition repräsentiert der Graph eine Gruppe von Wahrscheinlichkeitsverteilungen. Für diese können nun viele Operationen graphentheoretisch durchgeführt werden. So sind z. B. zwei Variablen X_1 und X_2, deren Knoten V_1 und V_2 durch den den Knoten V_S im Graphen separiert sind, auch bedingt unabhängig: $X_1 \perp\!\!\!\perp X_2 \,|\, X_S$.

2.4 Bayes'sche Netze

Bayes'sche Netze (BN) [43, 79, 80] sind gerichtete Graphen, die bedingte Unabhängigkeitsaussagen zwischen Variablen ausdrücken. Jeder Knoten des Graphens repräsentiert eine Zufallsvariable. Eine gerichtete Kante zwischen zwei Knoten drückt aus, dass die Variable des Elternknotens Einfluss auf die Variable des Kindknotens hat. BN werden z. B. in Expertensystemen [34, 124] und in der Mustererkennung [30, 112] eingesetzt. Viele bekannte Mustererkennungsverfahren (z. B. wiederum HMM) können als BN dargestellt werden [23]. Um BN zu definieren, sollen zuerst wieder die Markov-Eigenschaften von gerichteten Graphen erläutert werden.

2.4.1 Markov-Eigenschaften von gerichteten Graphen

Gegeben seien die Zufallsvariablen $X_{V_1}, \ldots, X_{V_N}$ und ein gerichteter, zyklusfreier Graph (DAG) $\mathcal{G}^D = (V, E)$. Jedem Knoten V_i des Graphens sei genau eine Zufallsvariable X_{V_i} zugeordnet. Die bedingte Unabhängigkeit der Variablen $X_{V_i} \perp\!\!\!\perp X_{V_j} \,|\, X_{V_k}$ wird im Folgenden wieder durch die Knoten-Kurznotation $V_i \perp\!\!\!\perp V_j \,|\, V_k$ aus Gleichung (2.13) ausgedrückt. Die Wahrscheinlichkeitsverteilung $p(x_{V_1}, \ldots, x_{V_N})$ erfüllt [87, 122, 125] dann die

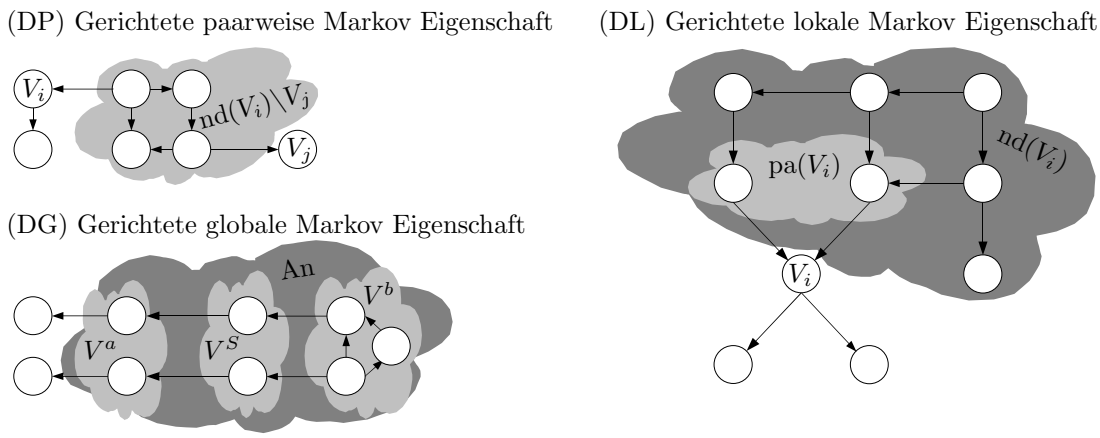

Abbildung 2.6: Paarweise, lokale und globale Markov-Eigenschaften von gerichteten Graphen.

(DP) gerichtete paarweise Markov-Eigenschaft, wenn für ein beliebiges Paar von nicht-benachbarten Knoten $V_i \in V$ und $V_j \in \mathrm{nd}(V_i)$ gilt, dass

$$V_i \perp\!\!\!\perp V_j \,|\, \mathrm{nd}(V_i)\backslash V_j, \tag{2.23}$$

(DL) gerichtete lokale Markov-Eigenschaft, wenn jeder beliebige Knoten V_i bei gegebenen Elternknoten $\mathrm{pa}(V_i)$ bedingt unabhängig von allen Nicht-Nachfahren ist

$$V_i \perp\!\!\!\perp \mathrm{nd}(V_i) \,|\, \mathrm{pa}(V_i), \tag{2.24}$$

(DG) gerichtete globale Markov-Eigenschaft, wenn für die disjunkten Knotenteilmengen $V^a \subset V$, $V^b \subset V$ und $V^s \subset V$ gilt, dass

$$V^a \perp\!\!\!\perp V^b \,|\, V^s, \tag{2.25}$$

immer dann, wenn V^a und V^b im Teilgraphen $\mathcal{G}^m_{\mathrm{An}(V^a \cup V^b \cup V^s)}$ durch V^s separiert sind. Dabei ist $\mathcal{G}^m_{\mathrm{An}(\cdot)}$ der moralisierte Graph der Ancestral Hull von V^a, V^b und V^s.

Abbildung 2.6 zeigt diese drei gerichteten Markov-Eigenschaften anhand von Beispielgraphen. Neben den Markov-Eigenschaften erfüllt die Wahrscheinlichkeitsverteilung $p(x_{V_1}, \ldots, x_{V_N})$ die

(DF) gerichtete, rekursive Faktorisierung, wenn es für jeden Knoten $V_i \in V$ eine nicht-negative Funktion gibt, die nur von den in $V_i \cup \mathrm{pa}(V_i)$ enthaltenen Zufallsvariablen abhängt und

$$p(x_{V_1}, \ldots, x_{V_N}) = \prod_{v \in V} f_v\big(x_v, \mathrm{pa}(x_v)\big). \tag{2.26}$$

Für jeden gerichteten, zyklusfreien Graphen $\mathcal{G}^D$, der die Zufallsvariablen $X_{V_1}, \ldots, X_{V_N}$ repräsentiert, gilt [94]:

$$(\text{DF}) \Leftrightarrow (\text{DG}) \Leftrightarrow (\text{DL}) \Rightarrow (\text{DP}). \tag{2.27}$$

Erfüllt ein Graph die gerichtete, rekursive Faktorisierung, so erfüllt er automatisch auch alle Markov-Eigenschaften. Im Falle von gerichteten Graphen sind die lokale und die globale Markov-Eigenschaft sogar äquivalent zur Faktorisierung. Allerdings kann im Allgemeinen nicht von der paarweisen Markov-Eigenschaft auf die Faktorisierung zurückgeschlossen werden. Weiterhin ist (DF) auch äquivalent zur d-Separierung in Graphen (siehe nächsten Abschnitt). Dadurch können viele Operationen in BN durch einfache Graphenoperationen durchgeführt werden.

BN sind so definiert, dass sie (DF) erfüllen (siehe Abschnitt 2.4.3). Dann erfüllen sie nach Gleichung (2.27) automatisch auch die globale Markov-Eigenschaft. Dadurch sind alle Berechnungen, die im moralisierten, ungerichteten Graphen $\mathcal{G}^m$ des Graphens $\mathcal{G}^D$ durchgeführt werden, auch für die dem BN zugrundeliegenden Zufallsvariablen gültig [23]. Dies wird später ausgenutzt, um alle Berechnungen effektiv in Verbundbäumen, die eine bestimmte Form eines moralisierten Graphens sind, durchzuführen (siehe dazu Abschnitt 2.8).

2.4.2 d-Separierung

Alternativ zu (DF) oder (DG) kann in DAG auch die d-Separierung (*d-separation*) [79, 122, 150] verwendet werden. Diese bietet eine rein graphentheoretische Möglichkeit zur Überprüfung von Unabhängigkeit von Variablen, ist jedoch vollkommen äquivalent zu (DF).

In einem DAG $\mathcal{G}^D$ gibt es genau drei verschiedene Verbindungstypen [122], die in Abbildung 2.7 gezeigt sind. Eine Verbindung zwischen den Knoten V_i und V_j heißt

divergierend, wenn es zwischen V_i und V_j einen Knoten V_s gibt, von dem aus zu beiden Knoten V_i und V_j ein gerichteter Weg führt, d. h. $V_s \mapsto V_i$ und $V_s \mapsto V_j$,

seriell, wenn es zwischen V_i und V_j einen Knoten V_s gibt, über den ein gerichteter Weg $V_i \mapsto V_s \mapsto V_j$ führt,

konvergierend, wenn es zwischen den Knoten V_i und V_j einen Knoten V_s gibt, zu dem von beiden Knoten V_i und V_j ein gerichteter Weg führt, d. h. $V_i \mapsto V_s$ und $V_j \mapsto V_s$.

Zwei Knoten V_i und V_j – und damit auch die ihnen zugrunde liegenden Zufallsvariablen X_{V_i} und X_{V_j} – sind in einem DAG **d-separiert** (*d-separated*) [122, 150], wenn alle Wege zwischen V_i und V_j durch einen Knoten V_s gehen und die Verbindung zwischen V_i und V_j entweder

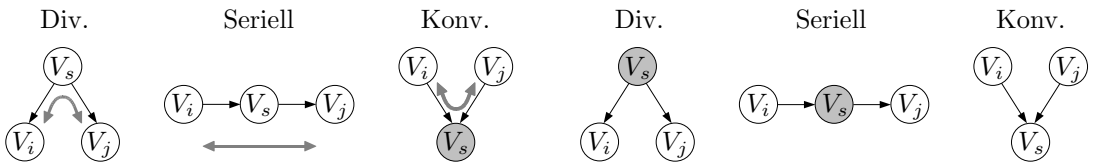

Abbildung 2.7: Verbindungsmöglichkeiten von DAG: divergierend, seriell und konvergierend. Variablen, deren Zustand bekannt ist, sind grau dargestellt. In den linken drei Beispielen ist ein Informationsfluss zwischen den Knoten V_i und V_j möglich, da die Variablen d-verbunden sind. In den rechten drei Beispielen ist kein Informationsfluss möglich, da die Knoten d-separiert sind.

- divergierend ist und der Zustand von V_s bekannt oder
- seriell ist und der Zustand von V_s bekannt,
- konvergierend ist und der Zustand von V_s und allen Nachfahren $\mathrm{de}(V_s)$ nicht bekannt ist.

Alle Knoten, die nicht d-separiert sind, sind **d-verbunden** (*d-connected*). Da die d-Separierung im Graphen äquivalent zu (DG) ist, gilt in einem DAG, in dem der Zustand einiger Knoten durch die Beobachtung $\vec{o}$ bekannt ist [79]: Wenn die Knoten V_i und V_j in $\mathcal{G}^D$ bei gegebener Beobachtung $\vec{o}$ d-separiert sind, so sind auch die den Knoten zugrunde liegenden Zufallsvariablen bei gegebener Beobachtung unabhängig, also $X_i \perp\!\!\!\perp X_j \,|\, \vec{o}$.

d-Separierung kann auch als Informationsblockierung aufgefasst werden: Wenn Variablen d-separiert sind, kann keine Information zwischen ihnen ausgetauscht werden. Sind Variablen d-verbunden, so fließt Information zwischen ihnen. Dies ist in Abbildung 2.7 gezeigt. Dabei sind Variablen, deren Zustand bekannt ist (observiert), grau hinterlegt; Variablen, deren Zustand unbekannt ist (unbeobachtet), sind weiß. Ein möglicher Informationsfluss zwischen den Knoten V_i und V_j ist mit einem Pfeil gekennzeichnet. Die d-Separierung gibt damit auch an, welche Variablen für eine konkrete Berechnung benötigt werden (nur die d-verbundenen). Der Bayes-Ball-Algorithmus [138] bietet eine effiziente Möglichkeit, die d-Separierung graphentheoretisch in linearer Zeitabhängigkeit von der Anzahl der Knoten zu bestimmen.

2.4.3 Definition von Bayes'schen Netzen

Ein BN ist ein DAG $\mathcal{G}^D$, der alle Wahrscheinlichkeitsverteilungen $p(x_{V_1}, \ldots, x_{V_N})$ repräsentiert, die eine gerichtete, rekursive Faktorisierung (DF) gemäß $\mathcal{G}^D$ erfüllen [79]. Daher gilt: Zu jeder Zufallsvariable X_{V_i} gibt es eine positive Funktion, die nur von den Zufallsvariablen des Knotens selbst und seinen Eltern, also $V_i \cup \mathrm{pa}(V_i)$, abhängt:

$$0 \leq f_{V_i}\big((x_{V_i}, \mathrm{pa}(x_{V_i})\big) \leq 1, \tag{2.28}$$

mit

$$\sum_{v \in V} f_v\big(x_v, \mathrm{pa}(x_v)\big) = 1. \tag{2.29}$$

Weiter ist die Verteilung $p(x_{V_1}, \ldots, x_{V_N})$ rekursiv faktorisierbar bezüglich $\mathcal{G}^D$, d. h.

$$p(x_{V_1}, \ldots, x_{V_N}) = \prod_{v \in V} f_v\big(x_v, \mathrm{pa}(x_v)\big). \tag{2.30}$$

Häufig ist die Funktion dabei eine bedingte Wahrscheinlichkeit

$$f_{V_i}\big((x_{V_i}, \mathrm{pa}(x_{V_i})\big) = p(x_{V_i} \,|\, \mathrm{pa}(x_{V_i})), \tag{2.31}$$

jedoch sind auch andere Funktionen möglich, solange Gleichung (2.28) und (2.29) erfüllt werden. Daher kann f_{V_i} z. B. auch deterministisch sein, so dass ein bestimmter Wert eines Elternknotens eine eindeutige Aktion auslöst. Dies wird insbesondere zur Segmentierung [29] verwendet.

Gemäß Gleichung (2.27) erfüllt ein BN damit auch die gerichtete globale (DG), lokale (DL) und paarweise (DP) Markov-Eigenschaft. Durch diese Definition repräsentiert der Graph eine Gruppe von Wahrscheinlichkeitsverteilungen. Für diese können viele Operationen graphentheoretisch durchgeführt werden. So kann z. B. die bedingte Unabhängigkeit von zwei Zufallsvariablen X_1 und X_2 bei einer gegebenen Beobachtung $\vec{o}$ mittels der d-Separierung überprüft werden: Sind die Knoten V_1 und V_2 im Graphen durch den Knoten V_s d-separiert, so sind auch die Variablen bedingt unabhängig $X_1 \perp\!\!\!\perp X_2 \,|\, \vec{o}$. Auf diese Weise kann die Komplexität von Berechnungen reduziert werden, da nur d-verbundene Variablen berücksichtigt werden müssen.

2.5 Dynamische Bayes'sche Netze

Bisher wurde davon ausgegangen, dass die Menge der benötigten Knoten bekannt ist: Der Graph $\mathcal{G}$ modelliert eine fixe Anzahl von Zufallsvariablen. Daher können mit BN nur statische Daten bzw. Daten, bei denen die Anzahl der beobachteten Zeitschritte bekannt sind, modelliert werden. Soll jedoch eine Sequenz von Merkmalen – z. B. der dynamischer Merkmalsvektor $\vec{o}$ – modelliert werden, so ist die Länge T der Merkmalsequenz im Allgemeinen nicht a-priori bekannt und kann zudem für unterschiedliche Ereignisse der selben Klasse variabel sein. Die Anzahl benötigter Variablen im Graphen ist dann abhängig von der jeweiligen Beobachtung $\vec{o}$. Um dies zu modellieren, werden Dynamische Bayes'sche Netze (*dynamic Bayesian network, DBN*) verwendet [26, 29, 113]. In einem DBN werden statische BN über die Zeit wiederholt und miteinander verknüpft. Obwohl das Grundprinzip von DBN prinzipiell immer gleich ist, gibt es verschiedene Beschreibungsmöglichkeiten, hier soll nur auf die zwei wichtigsten eingegangen werden.

Die expressivste Form DBN zu beschreiben, bietet die Notation nach Bilmes [26]. Diese ist auch in Abbildung 2.8 gezeigt. Hierbei werden drei Graphen

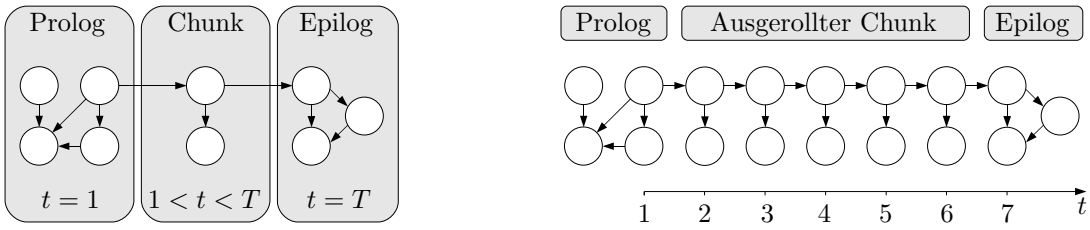

Abbildung 2.8: Beschreibung eines DBN mit Prolog, Zeitschlitz (*chunk*) und Epilog (links) und das – beispielhaft für sieben Beobachtungszeitpunkte – ausgerollte Netzwerk (rechts). Der Prolog modelliert die ersten T_P Zeitpunkte (hier $T_P = 1$) der Beobachtung und kann zusätzlich noch eine beliebige Anzahl von Knoten vor dem ersten Zeitpunkt enthalten, wodurch Vorbedingungen modelliert werden können. Der Epilog modelliert die letzten T_E Zeitpunkte (hier $T_E = 1$) und kann ebenfalls eine beliebige Anzahl von Knoten nach der letzten Beobachtung enthalten; dadurch können Endbedingungen festgelegt werden. Alle Beobachtungen zwischen Prolog und Epilog werden durch das Ausrollen des Chunks modelliert (hier fünfmal).

$$\mathcal{G}^P = (V^P, E^P),\ \mathcal{G}^C = (V^C, E^C) \text{ und } \mathcal{G}^E = (V^E, E^E), \tag{2.32}$$

sowie drei weiteren Verbindungsmengen

$$E^{P,C} \subseteq V^P \times V^C,\ E^{C,C} \subseteq V^C \times V^C \text{ und } E^{C,E} \subseteq V^C \times V^E \tag{2.33}$$

verwendet. $\mathcal{G}^P$ wird auch als Prolog, $\mathcal{G}^C$ als Zeitschlitz (*chunk*) und $\mathcal{G}^E$ als Epilog bezeichnet.

Jeder dieser drei Graphen beschreibt dabei ein statisches BN. Eine Abfolge von Beobachtungen $\vec{o}$ der Länge T wird nun modelliert, indem die statischen BN wiederholt und miteinander verbunden werden. Bezeichne T_P die Menge an beobachteten Knoten (bzw. Zeitpunkten) im Prolog und T_C und T_E analog die Zeitpunkte im Chunk und Epilog. Dann modelliert der Prolog $\mathcal{G}^P$ die ersten T_P Beobachtungen von $\vec{o}$. Analog modelliert der Epilog $\mathcal{G}^E$ die letzten T_E Beobachtungen von $\vec{o}$. Alle verbleibenden $T - T_P - T_E$ Zeitpunkte werden durch den Chunk modelliert. Dazu wird $\mathcal{G}^C$ solange wiederholt, bis alle verbleibenden Observierungen modelliert sind. Dieser Prozess wird auch als Ausrollen des DBN bezeichnet. Die Verbindung zwischen den Knoten des Prologs und des Chunks werden dabei durch $E^{P,C}$ modelliert, die Verbindung zwischen zwei aufeinanderfolgenden Zeitschlitzen des Chunks durch $E^{C,C}$ und die Verbindung zwischen dem letzten Chunk und dem Epilog mittels $E^{C,E}$. Der Prozess des Ausrollens eines DBN ist auch in Abbildung 2.8 verdeutlicht. Prinzipiell handelt es sich bei dem ausgerollten DBN wiederum um ein größeres statisches BN. Daher gelten alle Eigenschaften von BN auch für das ausgerollte Netzwerk eines DBN. So können mit DBN Observierungen mit verschiedenen Längen modelliert werden und dann mit den selben Methoden eines BN berechnet werden.

Häufig werden alle Zeitpunkte gleich modelliert, so dass keine besondere Behandlung des Prologs und Epilogs erforderlich ist. In einer vereinfachten Form wird für den Prolog und Epilog dann der gleiche Graph wie für den Chunk verwendet. In der Notation nach Murphy [113] besteht ein DBN dann aus nur einem Graphen $\mathcal{G} = (V, E)$, wobei E als die Intra-Verbindungsmenge (oder Matrix) bezeichnet wird, da sie die Verbindung der Knoten innerhalb eines Zeitschlitzes beschreibt und einer weiteren Verbindungsmatrix $E^{t,t+1} \subseteq V \times V$, die die Verbindung zweier zeitlich aufeinander folgender Zeitschlitze beschreibt (auch als Inter-Verbindung bezeichnet). Der Graph $\mathcal{G}$ wird dann T-fach ausgerollt, um alle Zeitpunkte der Observierung $\vec{o}$ zu modellieren. Wiederum ist das ausgerollte Netzwerk dann ein statisches BN.

Im Rahmen dieser Arbeit wird je nach Anwendung eine der beiden Beschreibungsformen verwendet. Sofern eindeutig, wird auch nur ein Ausschnitt aus dem ausgerollten Netzwerk gezeigt.

2.6 Weitere Graphische Modelle

Bisher wurden im Rahmen dieser Arbeit MRF und BN vorgestellt sowie DBN als dynamische Erweiterung. Dabei beschreibt jedes GM einen Satz von Bedingungen und repräsentiert damit die Wahrscheinlichkeitsverteilungen, die mindestens diese Bedingungen erfüllen [23]. Abbildung 2.9 zeigt, welche Verteilungen mit den hier vorgestellten GM dargestellt werden können.

Einige Verteilungen können nur durch MRF oder nur durch BN dargestellt werden: So können MRF keine kausalen Beziehungen abbilden, da zwei verbundene Variablen grundsätzlich miteinander interagieren. BN dagegen können keine gleichwertige Interaktion abbilden. Verteilungen, die sowohl durch MRF als auch durch BN darstellbar sind, werden als zerlegbare Modelle bezeichnet. Die Graphen dieser Verteilungen sind trianguliert, d. h. um den moralisierten Graphen des BN zu erhalten, müssen keine Kanten eingefügt werden. Diese Modelle werden daher auch als perfekte oder triangulierte (*chordal*) Graphen bezeichnet. Viele praktisch relevante Modelle, wie z. B. das HMM oder der Kalman Filter, haben einen perfekten Graphen. Sie können daher sowohl durch MRF als auch durch BN repräsentiert werden. Dies hat auch praktische Konsequenzen, da für perfekte Graphen besonders effiziente Berechnungsverfahren existieren.

BN und MRF sind die verbreitetste Form von GM, jedoch gibt es Modelle, die noch weitere Wahrscheinlichkeitsverteilungen repräsentieren können. Kettengraphen (*chain graphs*) [37] enthalten gerichtete und ungerichtete Kanten. Sie werden insbesondere eingesetzt, um Neuronale Netze [131] in GM einzubinden. Kettengraphen sind sehr mächtig und ihre Berechnungen subsumieren sowohl BN als auch MRF. Allerdings ist die Berechnung komplex und wird daher praktisch kaum eingesetzt. Tanner Graphen [147] werden zur Fehlerkorrektur in der Nachrichtentechnik verwendet. Factor Graphen [91] sind wiederum eine Oberklasse von MRF, BN und auch

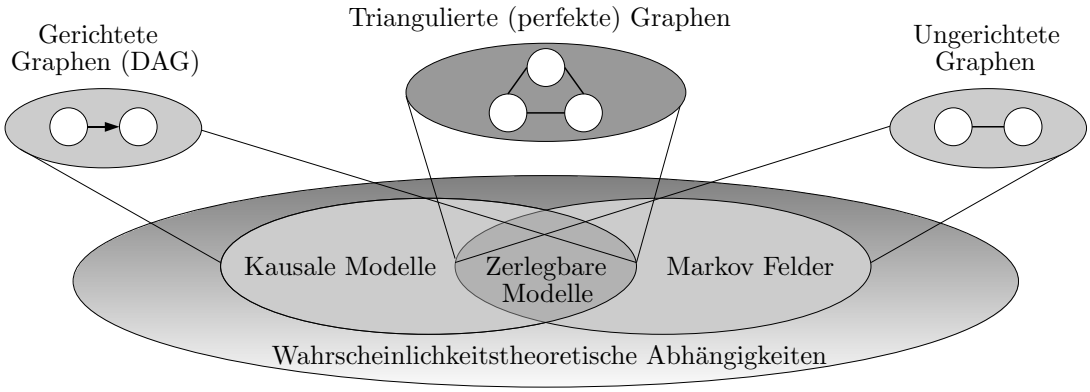

Abbildung 2.9: Verschiedene Typen von GM und die Wahrscheinlichkeitsverteilungen, die sie repräsentieren können. Nach einer Vorlage von Pearl [123].

Tanner Graphen. Sie müssen nicht zyklusfrei sein und sind daher mächtiger als BN; allerdings nicht mehr zwangsweise kausal. Sie werden hauptsächlich zur Beschreibung iterativer Turbo-Codes eingesetzt. Berechnungen innerhalb der Factor Graphen können z. B. durch den Sum-Product-Algorithmus [91] durchgeführt werden. Mit Nebenbedingungen umfasst dieser Algorithmus auch die Viterbi-Dekodierung [151], den Vorwärts-Rückwärts-Algorithmus für Markov-Modelle [126], Pearls Nachrichtenübertragungsverfahren für BN [123] sowie Fast-Fourier-Transformationen. Factor Graphen bieten daher eine sehr universelle Sichtweise auf verschiedene Probleme, finden jedoch in der Mustererkennung bisher kaum Anwendung.

2.7 Nomenklatur

Im Rahmen dieser Arbeit wird eine Nomenklatur aufbauend auf [113] verwendet: Dabei gibt die Form des Knotens die verwendete Wahrscheinlichkeitsdichtefunktion an. Diese Notation wird um die Kantennotation von [29] erweitert, um deterministische Beziehungen auszudrücken. Die verwendeten Knoten und Kanten sind in Abbildung 2.10 gezeigt.

Ein Knoten ist entweder diskret, kontinuierlich oder deterministisch. Weiterhin kann ein Knoten entweder beobachtet oder verborgen sein. Ein **diskreter Knoten** wird durch ein Quadrat dargestellt. Die zugrundeliegende Zufallsvariable nimmt genau einen von N möglichen Zuständen an. Ein **kontinuierlicher Knoten** wird durch einen Kreis dargestellt und die Wahrscheinlichkeitsdichtefunktionen durch eine Normalverteilung repräsentiert. **Deterministische Knoten** werden durch Achtecke dargestellt. Ein deterministischer Knoten führt abhängig vom Zustand der Elternknoten eine wohldefinierte Aktion aus; er kann daher auch als ein Sonderfall einer diskreten Wahrscheinlichkeitsverteilung angesehen werden, bei der ein bestimmter Zustand mit der Wahrscheinlichkeit eins auftritt. Üblicherweise werden determinis-

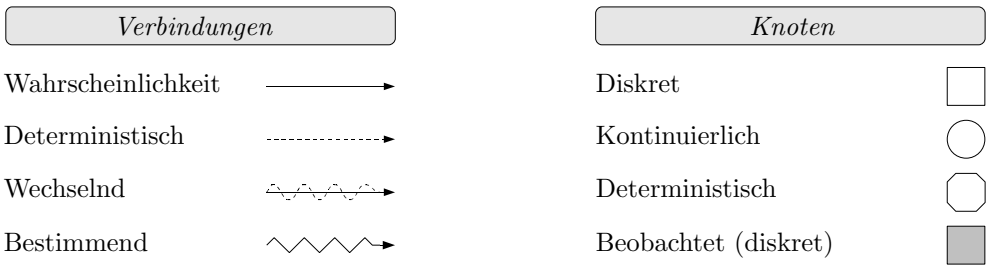

Abbildung 2.10: In dieser Arbeit verwendete Knoten- und Kantentypen. Die Notation ist angelehnt an [113] und erweitert um die deterministischen Komponenten von [29].

tische Knoten jedoch durch Entscheidungsbäume realisiert [30].

Ein **beobachteter** Knoten wird grau, ein **verborgener** weiß dargestellt. Bei einem beobachteten Knoten ist der Zustand der Zufallsvariable bekannt. Sie werden z. B. verwendet, um Observierungen in das GM zu integrieren. Bei einem verborgenen Knoten ist der Zustand unbekannt, zur Berechnung muss daher über alle möglichen Zustände summiert bzw. zum Viterbi-Dekodieren maximiert werden. Verborgene Knoten werden z. B. verwendet, um Markov-Ketten abzubilden.

Die Beziehung zwischen zwei Knoten wird durch gerichtete Kanten ausgedrückt. Ein durchgezogener Pfeil stellt dabei eine **Wahrscheinlichkeitsbeziehung** zwischen den Knoten dar. Der Kindknoten hat dann eine bedingte Wahrscheinlichkeit, die abhängig vom Zustand des Elternknotens ist. Ein gestrichelter Pfeil repräsentiert eine **deterministische** Beziehung der Knoten. Der Kindknoten nimmt dann abhängig vom Zustand des Elternknotens genau einen definierten Zustand ein. Ein Pfeil, der sowohl eine durchgezogene als auch eine gestrichelte Wellenlinie hat, stellt eine **wechselnde** Beziehung zwischen den Knoten dar. Abhängig vom Zustand eines **bestimmenden** Knotens (verbunden mittels einer gezackten Linie) kann die Beziehung wahrscheinlichkeitsbezogen oder deterministisch sein. Dies wird z. B. in Segmentierungsmodellen verwendet: Der bestimmende Knoten gibt dabei an, ob eine Klassengrenze vorliegt. Liegt keine Klassengrenze vor, wird deterministisch die Klasse aus dem vorherigen Klassenzustand übernommen. Liegt eine Klassengrenze vor, wird mittels einer Übergangswahrscheinlichkeit (z. B. einer Klassenbigrammwahrscheinlichkeit) die Folgeklasse ausgewählt.

2.8 Effiziente Berechnung in Bayes'schen Netzen

Berechnungen können in einem BN prinzipiell direkt durch Auflösen der Faktorisierung in Gleichung (2.30) nach den gewünschten Variablen und explizites Ausrechnen aller Variablenkonfigurationen durchgeführt werden. Daraus kann jedoch keine allgemeingültige Berechnung abgeleitet werden: Für jeden Graphen und für jede Anfrage

(d. h. für jede gesuchte Variable) müsste Gleichung (2.30) explizit neu aufgelöst und berechnet werden. Es lassen sich mit diesem Verfahren daher keinerlei generelle Berechnungsvorschriften ableiten.

Genereller und wesentlich effizienter können Berechnungen in BN durch Pearls Nachrichtenpropagierung (*message passing*) [122, 123] und darauf aufbauenden Verfahren [81, 107, 139] durchgeführt werden. Diese sind für jedes BN gültig und können daher bei gegebenen Graphen und Parametern die Konfiguration jeder Variable im Graphen berechnen. Diese Berechnung findet jedoch nicht direkt im BN statt, sondern in einem ungerichten Graphen des BN – dem Verbundbaum (*junction tree*) (JT). Die Nachrichtenpropagierungsverfahren für BN werden daher häufig verallgemeinernd als JT-Algorithmus[4] bezeichnet.

Zuerst sollen benötigte Berechnungen in einem GM erläutert werden. Dann wird erklärt, wie aus einem BN der Verbundbaum erstellt wird. Zuletzt wird dann HUGIN – eine auf Pearl aufbauende Methode – gezeigt, um innerhalb des Verbundbaumes Nachrichten zu propagieren.

2.8.1 Benötigte Berechnungen in Graphischen Modellen

Für HMM können drei Probleme formuliert werden [126], die auch bei GM Anwendung finden:

Berechnung der Auftrittswahrscheinlichkeit: Für eine gegebene Beobachtung $\vec{o}$, d. h. bei bekanntem Wert einiger Knoten des Graphens und bei gegebenen BN-Parametern λ soll die Auftrittswahrscheinlichkeit der Beobachtung

$$p(\vec{o}\,|\,\lambda) \tag{2.34}$$

berechnet werden. Dabei enthält λ sowohl die Struktur $\mathcal{G}$ als auch die Parameter der Wahrscheinlichkeitsverteilungen. Üblicherweise wird dies zur Klassifikation eingesetzt, indem für jede Klasse k ein Modell λ_k ausgewertet wird und die Beobachtung $\vec{o}$ dann der Klasse mit der höchsten Auftrittswahrscheinlichkeit $\widehat{k} = \operatorname{argmax}_k p(\vec{o}\,|\,\lambda_k)$ zugeordnet wird. Die Berechnung von $p(\vec{o}\,|\,\lambda)$ erfolgt durch direktes Anwenden des JT-Algorithmus.

Viterbi-Dekodierung: Bei einer gegebenen Beobachtung $\vec{o}$, d. h. bekanntem Wert einiger Knoten des Graphens und bei gegebenen BN-Parametern λ, soll die

[4] Die verallgemeinernde Bezeichnung aller Verfahren als JT-Algorithmus ist irreführend: Alle Verfahren führen die Berechnungen zwar innerhalb eines Verbundbaumes durch; zum einen unterscheiden sich die tatsächlichen Nachrichtenpropagierungen jedoch teilweise deutlich, zum anderen könnte die Propagierung einiger Verfahren prinzipiell auch auf anderen Graphen als dem Verbundbaum durchgeführt werden. Im Anhang von [113] findet sich dazu ein Vergleich der Verfahren und eine ausführliche Diskussion.

wahrscheinlichste Konfiguration aller (oder einiger) nicht beobachteter, verborgener (*hidden*) Knoten

$$\vec{h}^* = \underset{\vec{h}}{\operatorname{argmax}}\, p(\vec{h}, \vec{o} \,|\, \lambda) \tag{2.35}$$

bestimmt werden. Gesucht wird also der wahrscheinlichste Zustand aller verborgenen Knoten. Diese können durch die Anwenden des JT-Algorithmus bestimmt werden, indem alle Summationen durch Maximierungen ersetzt werden. Dieses Vorgehen entspricht dann dem Viterbi-Algorithmus [151].

Lernen der Modellparameter: Bei I gegebenen Beispielobservierungen $\vec{o}_i, i \in \{1, \ldots, I\}$ der Klasse k und gegebener Netzstruktur $\mathcal{G}$ sollen die Parameter der Wahrscheinlichkeitsverteilungen λ_k gelernt werden. Üblicherweise wird hier bei GM ein Maximum Likelihood Ansatz gewählt bei dem

$$\widehat{\lambda}_k = \underset{\lambda}{\operatorname{argmax}} \sum_{i=1}^{I} \log p(\vec{o}_i \,|\, \lambda) \tag{2.36}$$

die (logarithmierte) Erzeugungswahrscheinlichkeit der Observierungen maximiert wird. Die Parameter λ können dabei z. B. über das Gradientenabstiegsverfahren [113] oder den Expectation-Maximisation-Algorithmus (siehe Abschnitt 2.9) [50] bestimmt werden. Alle Lernverfahren setzen jedoch zuerst voraus, die vollständige Statistik des Modells zu bestimmen, also die Konfiguration aller Knoten bei gegebenen Modellparametern und Observierung zu errechnen. Der JT-Algorithmus ist deswegen eine benötigte Routine der Lernverfahren.

Die drei Probleme lassen sich daher zurückführen auf das effiziente Berechnen der Zustände aller (oder der gesuchten) Knoten im Graphen. Um sie zu lösen, wird jeweils der JT-Algorithmus verwendet, der im folgenden Abschnitt erläutert wird. Die Berechnung der Auftrittswahrscheinlichkeit ist direkt durch den Algorithmus gelöst. Die Viterbi-Dekodierung entspricht dem Algorithmus, allerdings werden alle (im folgenden erläuterten) Summationen durch Maximierungen ersetzt. Zum Lernen der Modellparameter ist der JT-Algorithmus eine notwendige Unterroutine, auf das tatsächliche Lernen aus der daraus gewonnen Statistik wird dann in Abschnitt 2.9 eingegangen. Der JT-Algorithmus hat für GM daher die gleiche Bedeutung wie der Vorwärts-Rückwärts-Algorithmus [126] für HMM[5].

[5] Tatsächlich ist der HMM Vorwärts-Rückwärts-Algorithmus nur ein Spezialfall des allgemeinen JT-Algorithmus. Dies wird in Abschnitt 2.10.2 gezeigt.

2.8.2 Der Verbundbaum

Im vorherigen Abschnitt wurde gezeigt, dass das effiziente Berechnen aller Zustände im Graphen sowohl für das Berechnen von Auftrittswahrscheinlichkeiten, das Dekodieren von Sequenzen als auch das Lernen von Modellparametern notwendig ist. Es ist daher wichtig, dass diese Berechnung so effizient wie möglich durchgeführt wird. Prinzipiell ist es möglich, alle Berechnungen direkt mittels der Faktorisierung aus Gleichung (2.30) durchzuführen, praktisch ist es jedoch effizienter, die Berechnungen auf der moralisierten, ungerichteten Version $\mathcal{G}^m$ des BN durchzuführen, dem JT [95]. Aufgrund der gerichteten globalen Markov-Eigenschaft (DG) sind alle Berechnungen, die im JT durchgeführt werden, auch für das BN gültig, da der JT mehr Kanten als das ursprüngliche BN hat: Der JT drückt immer gleich viele oder weniger, jedoch nie mehr bedingte Unabhängigkeiten aus. JT finden in verschiedenen Problemen Anwendung. Aus den unterschiedlichen Problemstellungen haben sich auch unterschiedliche Namen entwickelt, daher wird der JT auch als join tree, tree of belief universes, cluster tree oder clique tree bezeichnet [73].

Hier soll zuerst gezeigt werden, wie der JT aus dem BN erzeugt wird und welche Eigenschaften sich daraus ergeben. Im nächsten Abschnitt wird dann gezeigt, wie – mittels HUGIN Nachrichtenpropagierung – auf dem JT Wahrscheinlichkeiten aktualisiert werden. Die fünf folgenden Schritte zur Erzeugung des JT aus dem BN folgen im Wesentlichen [44, 45]. Sie sind rein graphentheoretisch, die tatsächlich zugrundeliegenden Wahrscheinlichkeitsverteilungen werden erst für die tatsächliche Nachrichtenpropagierung benötigt. Die Schritte sind auch in Abbildung 2.11 anhand eines beispielhaften BN visualisiert.

Eltern verbinden: Alle Co-Eltern werden mit einer ungerichteten Kante verbunden (*marrying parents*). Sind die Eltern $\mathrm{pa}(V_i)$ eines beliebigen Knotens V_i in $\mathcal{G}^D$ nicht miteinander verbunden, so wird paarweise eine ungerichtete Kante zwischen allen Eltern eingefügt. Das Verbinden der Co-Eltern ist notwendig, damit es später mindestens eine Clique gibt, die $p(V_i \,|\, \mathrm{pa}(V_i))$ vollständig aufnehmen kann.

Moralisieren: Alle gerichteten Kanten des Graphens werden in ungerichtete gewandelt. Zusammen ergeben das Verbinden der Eltern und das Verwerfen der Kantenrichtungen den moralisierten Graphen $\mathcal{G}^m$.

Triangulieren: In den moralisierten Graphen $\mathcal{G}^m$ werden solange Kanten eingefügt, bis es keinen Zyklus ohne Sehne mit mehr als vier Knoten gibt. Das heißt, nach dem Triangulieren gibt es im Graphen kein C_n, mit $n > 3$ mehr (kann z. B. mittels maximum cardinality search [148] überprüft werden). Nur durch das Triangulieren erfüllt der JT die running-intersection-Eigenschaft: Ein Knoten, der in zwei Cliquen C^1 und C^2 enthalten ist, ist dann auch in jeder Clique auf dem Weg zwischen C^1 und C^2 enthalten, d. h. der Knoten ist ein Separator

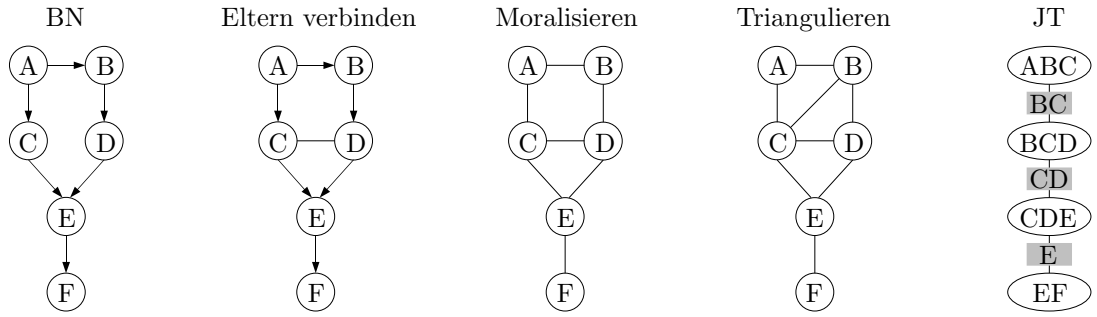

Abbildung 2.11: Erzeugen des JT aus dem BN: Zwischen allen nicht verbunden Co-Eltern wird eine ungerichtete Kante eingefügt (hier zwischen den Knoten C und D) – so gibt es später die Clique CDE, die die Verteilung $p(E \mid C, D)$ absorbieren kann. Ohne das Verbinden der Co-Eltern würden CE und DE im JT getrennte Cliquen bilden und $p(E \mid C, D)$ könnte nicht repräsentiert werden. Aus dem gerichteten Graphen wird dann durch das Verwerfen aller Kantenrichtungen der moralisierte Graph $\mathcal{G}^m$. Das anschließende Triangulieren stellt sicher, dass es keinen Zyklus ohne Sehne mit mehr als vier Knoten gibt. Nur so ist ein Knoten, der in zwei Cliquen enthalten ist, automatisch auch ein Separator. Die Triangulation ist nicht eindeutig: Hier hätten statt der Knoten B und C alternativ auch A und D miteinander verbunden werden können. Zuletzt formen alle maximalen Cliquen den JT. Dabei sind Knoten, die in mehreren Cliquen enthalten sind, auch ein Separator; hier explizit in den grauen Feldern angegeben.

zwischen den beiden Cliquen. Das Triangulieren des Graphens ist dabei nicht eindeutig. Im einfachsten Fall wird die Triangulation erfüllt, wenn alle Knoten des Graphens miteinander verbunden werden. Jedoch steigt die Rechenzeit der Nachrichtenpropagierung mit der Größe der maximalen Clique im Graphen exponentiell an [45]. Es sollte daher eine Triangulation gefunden werden, die zu möglichst kleinen maximalen Cliquen führt. Dieses Problem ist NP-schwer, jedoch sind effiziente heuristische Verfahren bekannt, z. B. für statische [16, 89] und dynamische BN [28].

Maximale Cliquen bestimmen: Die eindeutige Menge aller maximalen Cliquen im triangulierten Graphen wird, z. B. mit [163], bestimmt.

Verbundbaum aufstellen: Alle maximalen Cliquen formen zusammen den JT. Knoten, die in mehreren Cliquen enthalten sind, sind dann auch ein Separator zwischen diesen Cliquen.

Die bisherigen Schritte zur Erzeugung des JT aus dem BN waren rein graphentheoretisch. Der so erzeugte Graph kann jedoch alle Wahrscheinlichkeitsverteilungen des BN ausdrücken. Dazu muss der Baum mit den Wahrscheinlichkeiten des BN

initialisiert und die Information propagiert werden. Diese Initialisierung und Propagierung ist abhängig von der gewählten Nachrichtenpropagierung. Das HUGIN Verfahren dazu wird im nächsten Abschnitt erläutert. Unabhängig vom gewählten Verfahren erfüllt der JT nach der Initialisierung Eigenschaften, die ihn für Berechnungen von BN effizient machen. Bezeichne

$$\psi_c(x^c) \tag{2.37}$$

das Cliquepotential der Clique C, das nur von den in dieser Clique enthaltenen Zufallsvariablen X^C abhängt. Weiterhin bezeichne

$$\phi_s(x^s) \tag{2.38}$$

das Potential des Separators S, das nur von den in diesem Separator enthaltenen Zufallsvariablen X^S abhängt. Für jedes Paar von benachbarten Clustern und Separatoren gilt dann, dass

$$\sum_{x_i \in X^C \setminus X^S} \psi_c(x^c) = \phi_s(x^s) \tag{2.39}$$

das Aufsummieren des Cliquepotentials über alle nicht im Separator enthaltenen Variablen das Separator-Potential ergibt. Diese Eigenschaft wird als **lokale Konsistenz** (*locally consistent*) [73] des Baumes bezeichnet. Weiterhin enthält der JT die durch das BN repräsentierte Verbundwahrscheinlichkeit. Diese kann anhand der Cliquen- und Separator-Potentiale des JT berechnet werden als

$$p(x_{V_1}, \ldots, x_{V_N}) = \frac{\prod_{c \in \mathcal{C}} \psi_c(x^c)}{\prod_{s \in \mathcal{S}} \phi_s(x^s)}, \tag{2.40}$$

wobei $\mathcal{C}$ wieder die Menge aller Cliquen und $\mathcal{S}$ die Menge aller Separatoren im JT bezeichnet. Außerdem kann jede beliebige Variable X_{V_i} oder jeder Satz von Variablen X^A aus dem Baum marginalisiert werden, indem alle nicht benötigten Variablen entweder in einer beliebigen Clique $c \in \mathcal{C}$

$$p(x^a) = \sum_{x_i \in X^C \setminus X^A} \psi_c(x^c) \tag{2.41}$$

oder in einem beliebigem Separator $s \in \mathcal{S}$

$$p(x^a) = \sum_{x_i \in X^S \setminus X^A} \phi_s(x^s), \tag{2.42}$$

der die benötigten Variablen X^A enthält, aufsummiert werden. Sollte keine Clique c und kein Separator s alle benötigten Variablen X^A enthalten, kann Gleichung (2.40) verwendet werden, um dann die Marginalisierung auf der Verbundwahrscheinlichkeit durchzuführen. Dabei müssen in Gleichung (2.40) nicht alle Cliquen $c \in \mathcal{C}$ und

alle Separatoren $s \in \mathcal{S}$ des JT berechnet werden, sondern nur die Cliquen und Separatoren, die tatsächlich benötigt werden, um alle in X^A enthaltenen Variablen abzudecken.

Das Marginalisieren, d. h. Berechnen von bestimmten Variablen des BN, ist ein häufig benötigter Vorgang, z. B. während der Berechnung von Auftrittswahrscheinlichkeiten, dem Lernen von Daten und dem Viterbi-Dekodieren. Das direkte Marginalisieren aus der vollständigen Verbundwahrscheinlichkeit des BN – nach Gleichung (2.30) – ist dabei ineffizient, da alle Variablen, die das BN repräsentiert, mit in die Berechnung einbezogen werden. Durch die Berechnung über Gleichung (2.41) oder (2.42) ist es dagegen nur notwendig, über die in der Clique enthaltenen Variablen zu summieren. Da eine beliebige Clique, die die benötigte Variable enthält, gewählt werden kann, wird in der Praxis die kleinste (d. h. mit den wenigsten Variablen) gewählt. Dadurch können sich signifikant weniger Summationen ergeben. Das Marginalisieren über den JT ist daher in der Regel deutlich effizienter als die direkte Berechnung über das BN.

2.8.3 Nachrichtenpropagierung im Verbundbaum

In diesem Abschnitt soll in Grundzügen gezeigt werden, wie die Wahrscheinlichkeiten des BN im JT repräsentiert werden und wie neue Informationen im JT verbreitet werden. Diese Schritte werden als Nachrichtenpropagierung (*message passing, message propagation*) bezeichnet. Prinzipiell sind verschiedene Verfahren der Propagierung möglich, die meisten bauen jedoch auf Pearls Methoden [122, 123] auf. In dieser Arbeit soll nur auf das HUGIN Verfahren [81] eingegangen werden, das heute häufig eingesetzt wird.

Das HUGIN Verfahren besteht aus zwei Schritten: der Initialisierung des JT und der anschließenden globalen Nachrichtenpropagierung über alle Cliquen. Die Initialisierung muss dabei nur einmal nach der Erstellung des JT ausgeführt werden. Die Propagierung ist jedesmal erforderlich, wenn eine neue Information vorliegt, d. h. immer wenn eine neue Observierung für einen Knoten des Netzes vorliegt. Zur **Initialisierung** des JT wird zuerst jede Clique

$$\psi_c(x^c) = 1, \ \forall c \in \mathcal{C} \tag{2.43}$$

und jeder Separator

$$\phi_s(x^s) = 1, \ \forall s \in \mathcal{S} \tag{2.44}$$

des JT mit dem Wert eins belegt. Für jede Variable X_{V_i} des BN wird dann genau eine Clique $c \in \mathcal{C}$ in der X_{V_i} und alle Eltern $\mathrm{pa}(X_{V_i})$ enthalten sind gewählt. In das Cliquepotential $\psi_c(x^c)$ der ausgewählten Clique c wird nun die X_{V_i} zugrundeliegende Funktion hineinmultipliziert:

$$\psi_c^*(x^c) = \psi_c(x^c) \ f_v(x_{V_i}, \mathrm{pa}(x_{V_i})), \tag{2.45}$$

wobei $\psi_c^*(x^c)$ das neue durch X_{V_i} aktualisierte Potential bezeichnet. Durch das Verbinden der Eltern (erster Schritt beim Aufstellen des JT) ist sichergestellt, dass es mindestens eine Clique gibt, die die Funktion $f_v(x_{V_i}, \mathrm{pa}(x_{V_i}))$ aufnehmen kann. Nach dieser Initialisierung wird Gleichung (2.40) erfüllt, da

$$p(x_{V_1}, \ldots, x_{V_N}) = \frac{\prod_{c \in \mathcal{C}} \psi_c(x^c)}{\prod_{s \in \mathcal{S}} \phi_s(x^s)} = \frac{\prod_{v \in V} f_v(x_v, \mathrm{pa}(x_v))}{1} = \prod_{v \in V} f_v(x_v, \mathrm{pa}(x_v)) \quad (2.46)$$

die Verbundwahrscheinlichkeit, die das BN repräsentiert, auch durch den JT ausgedrückt wird. Allerdings ist der Baum noch nicht lokal konsistent, da Gleichung (2.39)

$$\sum_{x_i \in X^C \setminus X^S} \psi_c(x^c) \neq \phi_s(x^s) = 1 \quad (2.47)$$

nicht erfüllt ist. Daher können Wahrscheinlichkeiten noch nicht aus dem JT herausmaginalisiert werden. Zuerst müssen alle benachbarten Cliquen ihre Potentiale angleichen, um eine lokale Konsistenz herzustellen.

Dies geschieht mittels **lokaler Ausbreitung** von Nachrichten. Die Clique c_1 mit dem Potential $\psi_{c_1}(x^{c_1})$ gleicht ihr Potential an die benachbarte Clique c_2 mit dem Potential $\psi_{c_2}(x^{c_2})$ an, indem von c_2 über den Separator s mit dem Potential $\phi_s(x^s)$ eine Nachricht an c_1 geschickt wird: Das Separatorpotential $\phi_s(x^s)$ wird gespeichert und im Anschluss durch die Clique c_2 aktualisiert

$$\phi_s^*(x^s) = \sum_{x_i \in X^{C_2} \setminus X^S} \psi_{c_2}(x^{c_2}), \quad (2.48)$$

wobei $\phi_s^*(x^s)$ das aktualisierte Separatorpotential bezeichnet. Durch diesen Schritt enthält der Separator nun die Information, die c_1 und c_2 gemeinsam haben. Im zweiten Schritt der lokalen Ausbreitung wird dieser aktualisierte Separator durch die Clique c_1 absorbiert:

$$\psi_{c_1}^*(x^{c_1}) = \frac{\phi_s^*(x^s)}{\phi_s(x^s)} \, \psi_{c_1}(x^{c_1}). \quad (2.49)$$

Durch diese zwei Schritte haben c_1 und c_2 ihren gemeinsamen Teil – den Separator – aneinander angeglichen. Die beiden Cliquen sind dadurch lokal konsistent, da jetzt Gleichung (2.39) erfüllt ist. Gleichung (2.48) und (2.49) zusammen formen die lokale Nachrichtenübertragung zwischen zwei benachbarten Cliquen.

Damit die lokale Konsistenz für alle Paare von benachbarten Cliquen erfüllt ist, müssen alle Cliquen ihre Potentiale angleichen. Durch eine **globale Ausbreitung** von Nachrichten kann dies erreicht werden. Zuerst wird eine beliebige Clique des JT als Wurzel ausgewählt und der JT in einen Wurzelbaum gewandelt (alle Kanten zwischen den Cliquen werden von der Wurzel aus weggerichtet). Dann werden in

zwei Schritten, ausgehend von der Wurzelclique, rekursiv Nachrichten eingesammelt (*collect evidence*) und dann wiederum verteilt (*distribute evidence*).

Jede Clique versendet erst dann eine Nachricht an ihre Eltern, wenn sie vorher von allen Kindern eine Nachricht erhalten hat. Dies kann, mittels lokaler Nachrichtenübertragung, ausgehend von der Wurzel, mit dem folgenden rekursiven Algorithmus erreicht werden:

Algorithmus 1 Rekursives, globales Einsammeln von Nachrichten

```
procedure SAMMELN(Clique c_1, Clique c_2)
    for c_i ∈ ch(c_2) do          ▷ Zuerst Nachrichten von allen Kindern einsammeln
        Sammeln(c_2, c_i)
    end for                       ▷ Wenn alle Kinder eine Nachricht an c_2 gesendet haben,
    Sende Nachricht von c_2 zu c_1    ▷ schickt c_2 eine Nachricht an die Clique c_1
end procedure
```

Haben alle Kinder eine Nachricht an die Wurzel geschickt, werden Nachrichten wiederum an alle Kinder propagiert. Auch dieses Verteilen der Nachrichten kann durch einen rekursiven Algorithmus, ausgehend von der Wurzel, erreicht werden:

Algorithmus 2 Rekursives, globales Verteilen von Nachrichten

```
procedure VERTEILEN(Clique c)
    for c_i ∈ ch(c) do                          ▷ Alle Kinder von c
        Sende Nachricht von c zu c_i    ▷ werden durch eine Nachricht aktualisiert
        Verteilen(c_i)                  ▷ und aktualisieren dann wiederum ihre Kinder
    end for
end procedure
```

Sobald alle Blätter (Cliquen ohne Kinder) des JT eine Nachricht erhalten haben, ist die globale Nachrichtenpropagierung beendet. Dieses Sammeln und Verteilen von Nachrichten in einem JT ist in Abbildung 2.12 beispielhaft veranschaulicht. Somit ist der JT konsistent und beliebige Wahrscheinlichkeiten können herausmarginalisiert werden.

Jedesmal wenn über einen Knoten eine neue Information vorliegt (in der Regel eine neue Beobachtung), muss der Prozess der globalen Ausbreitung neu durchgeführt werden, damit der JT wieder konsistent wird. Zum Vergleich: In einem HMM bleiben die Struktur und die gelernten Parameter gleich, jedoch muss mit jeder neuen Beobachtung ein neuer Durchlauf der Vorwärtsberechnung durchgeführt werden. Im Falle des JT bleibt die Struktur und die Initialisierung erhalten, jedoch muss jede neue Beobachtung durch den Baum propagiert werden. Tatsächlich wird in Abschnitt 2.10.2 gezeigt, dass die Vorwärtsberechnung des HMM genau einer Form von Nachrichtenpropagierung durch einen bestimmten JT (dem des HMM) entspricht.

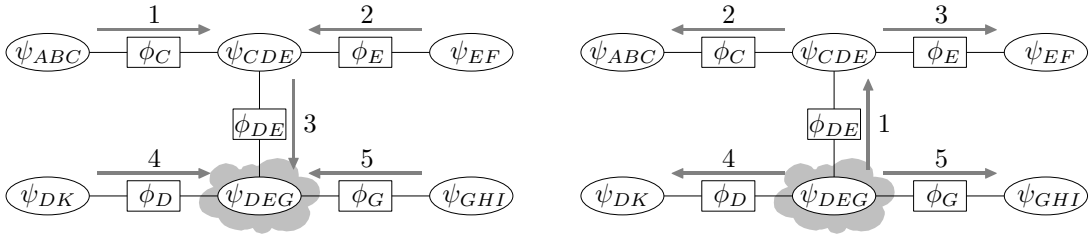

Abbildung 2.12: Globales Sammeln (links) und Verteilen (rechts) von Nachrichten. Ausgehend von der Wurzel DEG werden zuerst rekursiv Nachrichten eingesammelt. Eine Clique schickt ihre Nachricht erst dann an die aufrufende Clique, wenn sie selbst von allen Kindern eine Nachricht erhalten hat. CDE erhält daher erst Nachrichten von ABC und EF, bevor dann eine Nachricht an die Wurzel DEG gesendet wird. Sobald die Wurzel von allen Kindern eine Nachricht erhalten hat, propagiert sie die aktualisierte Information durch den JT. Dazu schickt sie Nachrichten an alle Kinder. Jede Clique die eine Nachricht erhält, schickt diese weiter an alle eigenen Kinder. Der Prozess ist dann beendet, wenn alle kinderlosen Cliquen eine Nachricht erhalten haben.

Für die globale Nachrichtenpropagierung müssen in einem Baum mit r Cliquen $r-1$ Nachrichten eingesammelt und weitere $r-1$ Nachrichten verteilt werden. Neben der Anzahl der Cliquen im JT hängt die Komplexität aber auch von der Größe der Cliquen ab, da entsprechend viele Wahrscheinlichkeiten aktualisiert werden müssen. In einfachen, unverzweigten JT (z. B. dem JT eines HMM) beträgt die Zeitkomplexität der gesamten Nachrichtenpropagierung dann $\mathcal{O}(N^{J+1})$, wobei N die Kardinalität der Knoten und J die maximale Anzahl der Eltern eines Knotens ist. In generellen Bäumen, die beliebig verzweigt sein können, erhöht sich die Zeitkomplexität der Propagierung auf

$$\mathcal{O}(N^w), \tag{2.50}$$

wobei N wiederum die Kardinalität der Knoten und w die Kardinalität der größten Clique im JT bezeichnet. Um eine effiziente Nachrichtenübertragung zu erhalten, ist es daher wichtig bei der Erzeugung des JT eine Triangulation zu finden, die möglichst kleine Cliquen erzeugt.

Die hier beschriebene HUGIN Propagierung ist nur eine Möglichkeit der Nachrichtenübertragung in JT. Eine detailliertere, prozedurale Anleitung zu dieser Propagierung – mit vielen Implementierungsdetails – findet sich in [73]. Andere verwendete Verfahren sind z. B. die Shenoy-Shafer Propagierung [139] oder die praktisch sehr relevanten array Techniken [73]. Die meisten Propagierungsverfahren sind wiederum Unterklassen des generalized distributive law (GDL) [107].

2.9 Lernen von Bayes'schen Netzen

Beim Lernen von GM gibt es zwei Ebenen und darin jeweils zwei prinzipielle Ansätze: Die erste Ebene ist die Festlegung der Anzahl der Knoten, das Aufstellen der Netzstruktur sowie das Festlegen der Art der WDF (z. B. diskret oder kontinuierlich) für jeden Knoten. In der zweiten Ebene werden dann die Parameter der WDF (z. B. der Mittelwert) für die Knoten bestimmt.

Die Netzstruktur kann entweder manuell durch einen Experten aufgestellt werden oder automatisch aus einem Datensatz gelernt werden. Beim automatischen Strukturlernen wird die Anzahl der Knoten, die Art der WDF in den Knoten sowie die Verbindungen der Knoten untereinander automatisch aus einem Datensatz gelernt [69]. Hierzu sind bisher nur wenige Verfahren bekannt. Häufig wird die Netzstruktur durch eine Exhaustionsmethode, also eine vollständige Suche (*brute force*) durch verschiedene Konfigurationen ermittelt und dann die Netzstruktur mit der größten Datenwahrscheinlichkeit ausgewählt. Diese Verfahren sind jedoch für viele Probleme der Mustererkennung mit großen Datensätzen und vielen Variablen praktisch nicht anwendbar. Dies gilt insbesondere für dynamische Daten: Hier steigt zum einen die Anzahl der benötigten Knoten mit der Länge der Trainingsdaten, wodurch eine Suche schnell unmöglich wird. Zudem ist die Netzstruktur durch die unterschiedlichen Längen der Daten variabel, was die Gestaltung der Suche deutlich erschwert. Strukturlernen wird daher für praktische Probleme derzeit kaum angewendet. Das Aufstellen des Graphens erfolgt daher häufig manuell.

In der zweiten Lernebene werden die Parameter der WDF für alle Knoten des GM bestimmt. Auch hier ist prinzipiell die Festlegung durch einen Experten oder das automatische Lernen aus Daten möglich. In Expertensystemen – z. B. für die medizinische Diagnostik – werden die Wahrscheinlichkeitsverteilungen häufig manuell festgelegt [79]. In der Mustererkennung ist es jedoch sinnvoller, die Parameter automatisch aus Trainingsdaten zu ermitteln. Hierzu sind verschiedene überwachte und unüberwachte Methoden für GM bekannt [65, 69, 82, 113, 115].

2.9.1 Maximum Likelihood Lernen

Hier wird auf das praktisch relevante Lernverfahren, das die Datenwahrscheinlichkeit maximiert (*maximum likelihood*) [65], eingegangen. Dazu seien I Trainingsbeispiele $\mathcal{O} = \{\vec{o}_1, \ldots, \vec{o}_I\}$ gegeben. Dann berechnet sich die Gesamtdatenwahrscheinlichkeit (*data-likelihood*) als

$$\mathcal{L}(\lambda \,|\, \mathcal{O}) = \log \prod_{i=1}^{I} p(\vec{o}_i \,|\, \lambda) = \sum_{i=1}^{I} \log p(\vec{o}_i \,|\, \lambda). \tag{2.51}$$

Dabei bezeichnet λ den gesuchten Parametersatz, also alle Parameter aller WDF des GM; nicht jedoch die Netzstruktur $\mathcal{G}^D$. Durch Ausnutzen von (F) wird diese

Funktion für ein GM mit N_V Knoten zu

$$\mathcal{L}(\lambda \,|\, \mathcal{O}) = \sum_{i=1}^{I} \log \prod_{v=1}^{N_V} f_v \left(x_v, \mathrm{pa}(x_v) \,|\, \vec{o}_i\right) = \sum_{i=1}^{I} \sum_{v=1}^{N_V} \log f_v \left(x_v, \mathrm{pa}(x_v) \,|\, \vec{o}_i\right). \quad (2.52)$$

Beim Maximum Likelihood Lernen werden die Parameter $\widehat{\lambda}$ so bestimmt, dass diese Gesamtdatenwahrscheinlichkeit maximiert wird:

$$\widehat{\lambda} = \underset{\lambda}{\operatorname{argmax}}\, \mathcal{L}(\lambda \,|\, \mathcal{O}). \quad (2.53)$$

Enthält der Trainingsdatensatz $\mathcal{O}$ Daten für alle N_V Knoten des GM, kann die Datenwahrscheinlichkeit direkt maximiert werden. Dazu muss jedoch jedes Trainingsbeispiel $\vec{o}_i$ für alle Knoten einen Eintrag vorweisen, es dürfen keine fehlenden Werte auftreten. Die einfachste Form des Lernens in diesem vollständig beobachteten Fall ist dann ein frequenzbasiertes Lernen.

2.9.2 Beobachtete Knoten: Frequenzbasiertes Lernen

Wenn alle Knoten des Graphens beobachtet sind und für alle Knoten auch Trainingsdaten vorliegen, kann die Gesamtdatenwahrscheinlichkeit aus Gleichung (2.52) durch verschiedene überwachte Lerntechniken (*supervised learning*) maximiert werden [113]. Enthält das GM keine deterministischen, sondern nur Knoten, die eine Wahrscheinlichkeit

$$f_v \left(x_v, \mathrm{pa}(x_v)\right) = p(x_v \,|\, \mathrm{pa}(x_v)), \forall v \in V \quad (2.54)$$

repräsentieren, ist ein frequenzbasiertes Lernen möglich. Dazu werden die konkreten Wahrscheinlichkeitsverteilungen der Knoten in Gleichung (2.52) eingesetzt und die Parameter durch direktes Maximieren bestimmt.

Dies soll hier beispielhaft für ein GM mit diskreten Knoten gezeigt werden. Die Gesamtdatenwahrscheinlichkeit ergibt sich dann zu:

$$\begin{aligned}\mathcal{L}(\lambda \,|\, \mathcal{O}) &= \sum_{i=1}^{I} \sum_{v=1}^{N_V} \log \prod_{j=1}^{J} \prod_{k=1}^{K} p(x_v = k \,|\, \mathrm{pa}(x_v) = j)\ \delta[x_v = k, \mathrm{pa}(x_v) = j \,|\, \vec{o}_i] \\ &= \sum_{i=1}^{I} \sum_{v=1}^{N_V} \sum_{j=1}^{J} \sum_{k=1}^{K} \log p(x_v = k \,|\, \mathrm{pa}(x_v) = j)\ \delta[x_v = k, \mathrm{pa}(x_v) = j \,|\, \vec{o}_i],\end{aligned} \quad (2.55)$$

wobei $k \in \{1, \ldots, K\}$ alle möglichen diskreten Zustände des Knotens v und $j \in \{1, \ldots, J\}$ die Produktmenge aller diskreten Zustände von allen Eltern von v umfasst[6]. Die Doppelsumme über k und j umfasst damit alle Einträge der WDF des

[6]Prinzipiell sind k und j abhängig von v, da jeder Knoten eine unterschiedliche Anzahl von Zuständen enthalten kann. Zur Vereinfachung der Schreibweise wird hierauf verzichtet, die Ableitungen gelten analog.

Knotens v. Die diskrete Diracfunktion $\delta[x_v = k, \mathrm{pa}(x_v) = j \,|\, \vec{o}_i]$ wird genau dann eins, wenn im Trainingsbeispiel $\vec{o}_i$ für den Knoten $x_v = k$ und gleichzeitig für die Eltern des Knotens $\mathrm{pa}(x_v) = j$ eintritt. Für alle anderen Beispiele wird die Funktion zu null. Die Gesamtdatenwahrscheinlichkeit hat sich daher auf ein Abzählen von Auftrittskombinationen der Trainingsbeispiele vereinfacht.

Mit einem Lagrangen Multiplikator und Ableiten nach den gesuchten Parametern kann Gleichung (2.55) direkt maximiert werden. Für die Wahrscheinlichkeit, dass der Knoten v den Zustand k einnimmt, wenn die Eltern im Zustand j sind, ergibt sich dann die Lernregel:

$$\widehat{p}\,(x_v = k \,|\, \mathrm{pa}(x_v) = j) = \frac{\sum_{i=1}^{I} \delta[x_v = k, \mathrm{pa}(x_v) = j \,|\, \vec{o}_i]}{\sum_{k'=1}^{K} \sum_{i=1}^{I} \delta[x_v = k', \mathrm{pa}(x_v) = j \,|\, \vec{o}_i]}, \tag{2.56}$$

die lediglich die Auftrittsfrequenz dieser Zustandskombination in den Trainingsdaten auswertet. Analog kann für den frequenzbasierten Ansatz eine Lernregel für Gaußknoten und andere WDF hergeleitet werden. Eine ausführliche Herleitung dazu findet sich z. B. im Anhang von [113].

Sind nur sehr wenige Trainingsbeispiele pro Knoten vorhanden oder ist die Anzahl der möglichen Zustände sehr groß, ist ein frequenzbasiertes Maximum Likelihood Lernen problematisch, da viele Kombinationen von k und j in den Trainingsdaten dann nicht auftreten und die entsprechenden Wahrscheinlichkeiten zu Null werden. Hier sollte für die Wahrscheinlichkeitseinträge ein Prior verwendet werden. Zum Lernen der Parameter kann dann z. B. eine maximum-a-posteriori Technik verwendet werden [69].

Maximum Likelihood Lernen kann ebenfalls nicht direkt angewendet werden, wenn einige Knoten des Graphens während der Trainingsphase nicht beobachtet sind oder nicht für alle Knoten des Graphens Trainingsmaterial vorliegt. Dann muss ein Näherungsverfahren wie der nachfolgend erklärte Expectation-Maximisation-Algorithmus eingesetzt werden.

2.9.3 Teilweise unbeobachtete Knoten: EM-Lernen

Bei vielen praktischen Problemen enthält der Datensatz $\mathcal{O}$ unvollständige Einträge: Es sind nicht für alle Knoten des Graphens Trainingsdaten vorhanden oder einige Knoten sind während des Trainings nicht beobachtet (*hidden*). Dies tritt insbesondere häufig bei dynamischen Problemen, z. B. der Verwendung von Markov-Ketten, auf. Für ein BN mit verborgenen Knoten wird die Gesamtdatenwahrscheinlichkeit aus Gleichung (2.52) zu

$$\mathcal{L}(\lambda \,|\, \mathcal{O}) = \sum_{i=1}^{I} \log p(\vec{o}_i \,|\, \lambda) = \sum_{i=1}^{I} \log \sum_{h'=1}^{H} p(x^h = h', x^o = \vec{o}_i), \tag{2.57}$$

wobei x^h die Menge der verborgenen Variablen und $h' \in \{1, \ldots, H\}$ ihre möglichen Konfigurationen bezeichnet. Die observierten Variablen x^o nehmen den Wert der aktuellen Beobachtung $\vec{o}_i$ an. Diese Funktion kann aufgrund der Summe innerhalb des Logarithmus nicht mehr geschlossen optimiert werden. Zum Optimieren der Datenwahrscheinlichkeit müssen daher Näherungsverfahren wie das Gradientenverfahren [31] oder der Expectation-Maximisation-Algorithmus (EM) [10, 25, 50, 108] verwendet werden. In der Praxis am verbreitetsten ist die Optimierung mit dem EM-Algorithmus, dessen Anwendung auf BN hier kurz beschrieben wird.

Bezeichne λ wiederum den gesuchten Satz der Parameter aller WDF aller Knoten des GM und λ' einen beliebigen vorhandenen Satz von Parametern. Dann kann eine Hilfsfunktion über den Erwartungswert der vollständigen Datenwahrscheinlichkeit

$$\begin{aligned} Q(\lambda, \lambda') &= \Big\langle \log p(\mathcal{O}, h' \,|\, \lambda) \Big\rangle_{\mathcal{O},\lambda'} \\ &= \sum_{i=1}^{I} \sum_{h'=1}^{H} \log p(x^h = h', x^o = \vec{o}_i \,|\, \lambda)\; p(x^h = h' \,|\, x^o = \vec{o}_i, \lambda') \end{aligned} \tag{2.58}$$

aufgestellt werden. Dabei umfasst h' wiederum alle möglichen Konfigurationen der verborgenen Variablen. Dempster [50] hat gezeigt, dass das Maximieren dieser Hilfsfunktion

$$\widehat{\lambda} = \operatorname*{argmax}_{\lambda} Q(\lambda, \lambda') \tag{2.59}$$

auch die Datenwahrscheinlichkeit $\mathcal{L}(\lambda \,|\, \mathcal{O})$ maximiert. Anstelle von Gleichung (2.57), die nicht geschlossen maximiert werden kann, wird daher Gleichung (2.58) maximiert. Durch Ausnutzen der Faktorisierung von BN wird diese Hilfsfunktion zu

$$\begin{aligned} Q(\lambda, \lambda') &= \sum_{i=1}^{I} \sum_{h'=1}^{H} \log \prod_{v=1}^{N_V} p(x_v \,|\, \mathrm{pa}(x_v), x^o = \vec{o}_i, x^h = h')\; p(x^h = h' \,|\, x^o = \vec{o}_i, \lambda') \\ &= \sum_{i=1}^{I} \sum_{h'=1}^{H} \sum_{v=1}^{N_V} \log p(x_v \,|\, \mathrm{pa}(x_v), x^o = \vec{o}_i, x^h = h')\; p(x^h = h' \,|\, x^o = \vec{o}_i, \lambda'). \end{aligned} \tag{2.60}$$

Dabei ist $p(x^h = h' \,|\, x^o = \vec{o}_i, \lambda')$ der Schätzschritt (*expectation*) des Algorithmus, der durch den JT-Algorithmus für ein beliebiges BN berechnet werden kann [93]. Der Maximierungsschritt (*maximisation*) hängt von den konkreten Wahrscheinlichkeitsverteilungen der Knoten ab, kann jedoch z. B. für diskrete oder Gaußknoten geschlossen aus der Hilfsfunktion bestimmt werden. Eine Ableitung für diskrete Knoten findet sich z. B. in [113] und für Gaußknoten z. B. in [10].

Der EM-Algorithmus wird dann iterativ durchgeführt: Zuerst wird ein beliebiger Parametersatz λ' gewählt. Für diesen Satz wird die Auftrittswahrscheinlichkeit der verborgenen Knoten mittels des JT-Algorithmus berechnet. Durch den Maximierungsschritt werden die initialen Parameter dann verbessert. Die so gewonnen

Parameter λ werden in einem weiteren Schätzschritt verwendet und wiederum maximiert. Dieser Wechsel von Schätzen und Maximieren der Parameter wird iterativ wiederholt, bis der Algorithmus konvergiert.

2.10 Anwendungen von Bayes'schen Netzen

In diesem Abschnitt sollen Beispiele von BN gezeigt werden. Dabei soll insbesondere auf Anwendungen der Mustererkennung eingegangen werden. Es werden noch keine neuen Modellstrukturen eingeführt, sondern bekannte Modelle als BN dargestellt. Dies soll zum einen die Vielseitigkeit der BN zeigen, zum anderen aber auch wie bekannte Verfahren der Mustererkennung durch die intuitive graphische Struktur modelliert werden können und wie sich die speziellen Algorithmen aus dem allgemeinen JT-Algorithmus ableiten lassen. In den Anwendungskapiteln bauen viele im Rahmen dieser Arbeit entwickelte Modelle auf den folgenden vorgestellten Verfahren auf oder werden aus ihnen zusammengesetzt.

2.10.1 Statische Modelle

Generatives Modell

Abbildung 2.13 zeigt, wie verschiedene statische Mustererkennungsverfahren als BN repräsentiert werden können. Beim generativen Modell [76, 101] erzeugt jede Klasse eine WDF. Die Verbundwahrscheinlichkeit von beobachtetem Merkmal x und der Klasse c_i ergibt sich aus der Faktorisierung des Graphens zu

$$p(c_i, x) = p(x \mid c_i)\, p(c_i). \tag{2.61}$$

Daraus ergibt sich die Klassenauftrittswahrscheinlichkeit bei insgesamt K möglichen Klassen zu

$$p(c_i \mid x) = \frac{p(c_i, x)}{p(x)} = \frac{p(x \mid c_i)\, p(c_i)}{\sum_{c_j=1}^{K} p(x \mid c_j)\, p(c_j)}, \tag{2.62}$$

die direkt zur Klassifikation angewendet werden kann. Dabei kann $p(x \mid c_i)$ durch jede beliebige WDF repräsentiert werden; häufig verwendet werden Gaußkurven. Die aus der Faktorisierung abgeleitete Gleichung (2.62) zeigt auch, dass es sich um einen Bayes Ansatz [132] handelt.

Diskriminatives Modell

Durch das Vertauschen der Kind- und Elternbeziehung im generativen Ansatz – also Drehen der Pfeilrichtung –, erhält man ein diskrimatives Modell [76]. Hier erzeugt jedes Merkmal x eine Auftrittswahrscheinlichkeit für die Klasse c_i, so dass das Modell zu

$$p(c_i, x) = p(c_i \mid x)\, p(x) \tag{2.63}$$

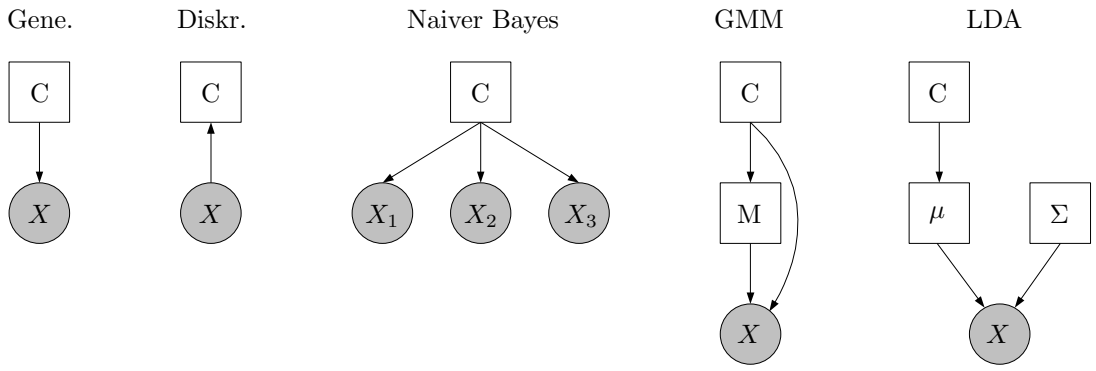

Abbildung 2.13: Beispiele von Bayes'schen Netzen bekannter statischer Modelle. Beim generativen Ansatz erzeugt die Klasse ein Merkmal. Beim diskriminativen Ansatz erzeugt jedes Merkmal eine Klasse. Beim naiven Bayes Klassifikator erzeugt jede Klasse mehrere unabhängige Merkmale. Beim Gauß Mixtur Modell (GMM) erzeugt eine Klasse mehrere Mixturen, die dann überlagert werden, um ein Merkmal zu erzeugen. Bei der linearen Diskriminanz Analyse (LDA) wird ein Merkmal durch eine Kovarianz und einen klassenabhängigen Mittelwert beschrieben.

faktorisiert wird. Die Auftrittswahrscheinlichkeit der Klasse bei einer gegebenen Observierung $p(c_i \mid x)$ kann dann direkt und ohne Marginalisierung aus dem Klassenknoten C abgelesen werden.

Naiver Bayes Klassifikator

Erzeugt eine Klasse beim generativen Ansatz nicht nur ein, sondern gleichzeitig N Merkmale, ergibt sich ein naiver Bayes Klassifikator [54, 55, 104]. Die divergierende Verbindung im GM drückt die Unabhängigkeit der verschiedenen Merkmale bei gegebener Klasse aus. Durch die Faktorisierung ergibt sich dann für diesen Graphen die Verbundwahrscheinlichkeit der N Merkmale $x_1, \ldots, x_N$ und der Klasse c_i zu

$$p(c_i, x_1, \ldots, x_N) = p(x_1 \mid c_i) \ \ldots \ p(x_N \mid c_i)\, p(c_i) = p(c_i) \prod_{j=1}^{N} p(x_j \mid c_i) \tag{2.64}$$

und daraus durch Umformung die Klassenauftrittswahrscheinlichkeit bei K möglichen Klassen:

$$p(c_i \mid x_1, \ldots, x_N) = \frac{p(c_i, x_1, \ldots, x_N)}{p(x_1, \ldots, x_N)} = \frac{p(c_i) \ \prod_{j=1}^{N} p(x_j \mid c_i)}{\sum_{c_k=1}^{K} p(c_k) \ \prod_{j=1}^{N} p(x_j \mid c_k)}. \tag{2.65}$$

Wiederum kann dabei die reale Merkmalauftrittswahrscheinlichkeit $p(x_i \mid c_i)$ durch beliebige WDF approximiert werden. Es ist ebenfalls möglich, dass unterschiedliche Merkmale durch unterschiedliche Typen von WDF modelliert werden, z. B. teilweise kontinuierlich und teilweise diskret.

Gauß Mixtur Modell

Wird das generative Modell so erweitert, dass eine Klasse mehrere Mixturen erzeugt, die dann zur Erzeugung des Merkmals überlagert werden, ergibt sich ein Gauß Mixtur Modell (GMM) [10, 25, 101]. Mit der Klasse c_i, den Mixturen m und dem Merkmal x faktorisiert der Graph die Verbundwahrscheinlichkeit zu

$$p(c_i, m, x) = p(x \mid m, c_i)\, p(m \mid c_i)\, p(c_i). \tag{2.66}$$

Daraus ergibt sich durch Umformung und Marginalisieren über alle M Mixturen bzw. alle K Klassen die Klassenauftrittswahrscheinlichkeit der Klasse c_i bei gegebenem Merkmal x zu

$$\begin{aligned} p(c_i \mid x) &= \frac{p(c_i, x)}{p(x)} = \frac{\sum_{m=1}^{M} p(c_i, m, x)}{\sum_{c_k=1}^{K} \sum_{m=1}^{M} p(c_k, m, x)} \\ &= \frac{\sum_{m=1}^{M} p(x \mid m, c_i)\, p(m \mid c_i)\, p(c_i)}{\sum_{c_k=1}^{K} \sum_{m=1}^{M} p(x \mid m, c_k)\, p(m \mid c_k)\, p(c_k)} \\ &= \frac{1}{Z}\, p(c_i) \sum_{m=1}^{M} p(x \mid m, c_i)\, p(m \mid c_i), \end{aligned} \tag{2.67}$$

wobei Z eine für die Klassifikation nicht benötigte Normalisierung darstellt und daher im Folgenden vernachlässigt wird. Bezeichnet $\alpha_m^i = p(m \mid c_i)$ das Mixturgewicht für die Klasse c_i, die Auftrittswahrscheinlichkeit der Mixtur m, ergibt sich für die Klassenauftrittswahrscheinlichkeit

$$p(c_i \mid x) = p(c_i) \sum_{m=1}^{M} \alpha_m^i\, p(x \mid m, c_i), \tag{2.68}$$

die allgemeine Mixtur Modell Klassifikationsformel. Werden alle Mixturen aller Klassen durch Gaußkurven $p(x \mid m, c_i) = \mathcal{N}(x, \mu_i, \Sigma_i)$ repräsentiert, ergibt sich ein GMM:

$$p(c_i \mid x) = p(c_i) \sum_{m=1}^{M} \alpha_m^i\, \mathcal{N}(x, \mu_i, \Sigma_i). \tag{2.69}$$

Eine Erweiterung auf mehrdimensionale Merkmale $\vec{x}$ erfolgt analog durch die Verwendung mehrdimensionaler Gaußkurven $\mathcal{N}(x, \vec{\mu}_i, \underline{\Sigma}_i)$. Prinzipiell sind auch bei dem Mixtur Modell andere WDF als Gaußkurven und auch verschiedene WDF für verschiedene Klassen möglich. Weiterhin ist es möglich, das GMM um eine naive Bayes Komponente zu erweitern, so dass Merkmale durch verschiedene Mixturen, aber voneinander unabhängig, modelliert werden können.

Lineare Diskriminanzanalyse

Werden die Kovarianzmatrix $\underset{\sim}{\Sigma}$ und der Mittelwertsvektor $\vec{\mu}$ der Gaußkurven nicht implizit mit in den kontinuierlichen Gaußknoten modelliert, sondern explizit in eigenen Knoten angegeben, kann auch die lineare Diskriminanzanalyse (LDA) [55, 62, 105] als BN ausgedrückt werden. Dabei wird der Merkmalvektor $\vec{x}$ unterschieden durch eine klassenunabhängige Kovarianzmatrix $\underset{\sim}{\Sigma}$ und einem von der Klasse c_i abhängigen Mittelwertvektor $\vec{\mu}_i$. Der in Abbildung 2.13 gezeigte Graph faktorisiert die Verbundwahrscheinlichkeit zu

$$p(c_i, \vec{x}, \vec{\mu}_i, \underset{\sim}{\Sigma}) = p(\vec{x} \,|\, \vec{\mu}_i, \underset{\sim}{\Sigma})\; p(\underset{\sim}{\Sigma})\; p(\vec{\mu}_i \,|\, c_i)\; p(c_i). \tag{2.70}$$

Daraus ergibt sich wiederum durch Umformung und Marginalisieren die Klassenauftrittswahrscheinlichkeit der Klasse c_i bei gegebenem Merkmalvektor $\vec{x}$

$$p(c_i \,|\, \vec{x}) = \frac{p(c_i, \vec{x})}{p(\vec{x})} = \frac{\sum_{k=1}^{K} \sum_{\underset{\sim}{\Sigma}} p(c_i, \vec{x}, \vec{\mu}_k, \underset{\sim}{\Sigma})}{\sum_{c_k=1}^{K} \sum_{k=1}^{K} \sum_{\underset{\sim}{\Sigma}} p(c_k, \vec{x}, \vec{\mu}_k, \underset{\sim}{\Sigma})}. \tag{2.71}$$

Die Kovarianzmatrix $\underset{\sim}{\Sigma}$ ist für alle Klassen identisch, ihre Auftrittswahrscheinlichkeit ist daher $p(\underset{\sim}{\Sigma}) = 1$. Dann folgt aus Gleichung (2.70) und (2.71) für die Auftrittswahrscheinlichkeit

$$p(c_i \,|\, \vec{x}) = \frac{p(c_i, \vec{x})}{p(\vec{x})} = \frac{\sum_{k=1}^{K} p(\vec{x} \,|\, \vec{\mu}_k, \underset{\sim}{\Sigma})\; p(\vec{\mu}_k \,|\, c_i)\; p(c_i)}{\sum_{c_k=1}^{K} \sum_{k=1}^{K} p(\vec{x} \,|\, \vec{\mu}_k, \underset{\sim}{\Sigma})\; p(\vec{\mu}_k \,|\, c_k)\; p(c_k)}. \tag{2.72}$$

Sei $p(\vec{\mu}_k \,|\, c_i)$ dann, und nur dann, eins, wenn $\vec{\mu}_k$ tatsächlich der Mittelwert der Klasse c_i ist, d. h. $p(\vec{\mu}_i \,|\, c_i) = 1, \forall i$ und $p(\vec{\mu}_k \,|\, c_i) = 0, \forall k \neq i$. Weiterhin sei $p(\vec{x} \,|\, \vec{\mu}_i, \underset{\sim}{\Sigma}) = \mathcal{N}(\vec{x}, \vec{\mu}_i, \underset{\sim}{\Sigma})$ die normalverteilte WDF der Klasse c_i. Damit ergibt sich für die Klassenauftrittswahrscheinlichkeit

$$p(c_i \,|\, \vec{x}) = \frac{\mathcal{N}(\vec{x}, \vec{\mu}_i, \underset{\sim}{\Sigma})\; p(c_i)}{\sum_{c_k=1}^{K} \mathcal{N}(\vec{x}, \vec{\mu}_k, \underset{\sim}{\Sigma})\; p(c_k)}. \tag{2.73}$$

Wird diese Wahrscheinlichkeit logarithmiert und dann als eine Entscheidungsfunktion [132]

$$f_i(\vec{x}) = (\vec{x} - \vec{\mu}_i)^T \underset{\sim}{\Sigma}^{-1} (\vec{x} - \vec{\mu}_i) + \log p(c_i) \tag{2.74}$$

ausgedrückt, ergibt sich die lineare Diskriminanzfunktion. Dabei wurden hier bereits alle Faktoren, die für alle Klassen gleich sind (und daher beim Maximieren wegfallen), vernachlässigt.

Quadratische, Mixtur- und Quadratische-Mixtur-Diskriminanzanalyse

Durch Einführen einer weiteren Verbindung vom Klassenknoten C zum Kovarianzknoten Σ in Abbildung 2.13 wird aus der LDA die quadratische Diskriminanzanalyse

(QDA) [48]

$$p(c_i \mid \vec{x}) = \frac{\mathcal{N}(\vec{x}, \vec{\mu}_i, \underset{\sim}{\Sigma}_i)\ p(c_i)}{\sum_{c_k=1}^{K} \mathcal{N}(\vec{x}, \vec{\mu}_k, \underset{\sim}{\Sigma}_k)\ p(c_k)}, \tag{2.75}$$

bei der neben dem Mittelwert auch die Kovarianzmatrix von der Klasse abhängt. Modelle wie die Mixtur-Diskriminanzanalyse (MDA) oder die quadratische Mixtur-Diskriminanzanalyse (QMDA) verwenden einen zusätzlichen Knoten, um Merkmale durch mehrere Mixturen zu modellieren. Entsprechende BN dazu werden in [23] gezeigt.

2.10.2 Das Hidden Markov Modell

Dynamische Merkmalssequenzen werden häufig mit dem Hidden Markov Modell (HMM) [14, 15] verarbeitet. In dieser Arbeit sollen sie nur aus Sicht der GM detailliert erläutert werden, um die Prinzipien der DBN und des JT-Algorithmus zu zeigen. Die Grundlagen der HMM sollen jedoch nur umrissen werden: Ein HMM modelliert den zeitlichen Verlauf mit einer Markov-Kette, wobei jeder Zustand in einem zweiten stochastischem Prozess ein Symbol der Merkmalsequenz emittiert. Durch diesen doppelten Prozess können zeitliche Verzerrungen modelliert werden. HMM wurden daher erfolgreich in vielen Anwendungen, insbesondere der Spracherkennung, eingesetzt [166]. Eine detailliertere Beschreibung findet sich in [126, 127], Möglichkeiten und Grenzen werden in [27] abgewogen, Implementierungsdetails finden sich in [166].

Abbildung 2.14 (links) zeigt wie ein diskretes HMM als DBN dargestellt werden kann. Für eine Merkmalsequenz mit T Beobachtungen $\vec{o}_t$ besteht das Modell aus $2T$ Variablen. T Variablen modellieren eine Markov-Kette, jede Variable hat dabei N mögliche Zustände. Der Zustand q_t zum Zeitpunkt t hängt nur vom vorherigen Zustand q_{t-1} ab. Dies zeigt sich im Modell durch die serielle Verbindung der Zustandsvariablen. In den ersten Zustand wird mit der Wahrscheinlichkeit $p(q_1)$ eingesprungen. Alle weiteren Zustände ergeben sich durch die serielle Verbindung über die Zustandsübergangswahrscheinlichkeit $p(q_t \mid q_{t-1})$. Der zweite stochastische Prozess ist die Erzeugung von Beobachtungen: In jedem Zeitschritt erzeugt der aktuelle Zustand q_t generativ (vgl. dazu die statischen Modelle in Abschnitt 2.10.1) das aktuelle Symbol der Beobachtung $\vec{o}_t$ mit der Wahrscheinlichkeit $p(\vec{o}_t \mid q_t)$. Zu den T Zustandsvariablen kommen daher noch einmal T Variablen für die Beobachtungen. Üblicherweise werden in einem HMM nur die Merkmale beobachtet, während die Zustände verborgen (*hidden*) bleiben. Das DBN eines HMM faktorisiert die Verbundwahrscheinlichkeit der Beobachtung $\vec{o}$ und der Zustandssequenz $\vec{q}$ dann zu:

$$p(\vec{o}, \vec{q}) = p(q_1)\ p(\vec{o}_1 \mid q_1) \prod_{t=2}^{T} p(q_t \mid q_{t-1})\ p(\vec{o}_t \mid q_t). \tag{2.76}$$

Wird die Zustandsübergangswahrscheinlichkeit als $a_{ij} = p(q_t = j \mid q_{t-1} = i)$, die Zustandseinsprungswahrscheinlichkeit als $\pi_i = p(q_1 = i)$ und die Symbolemissions-

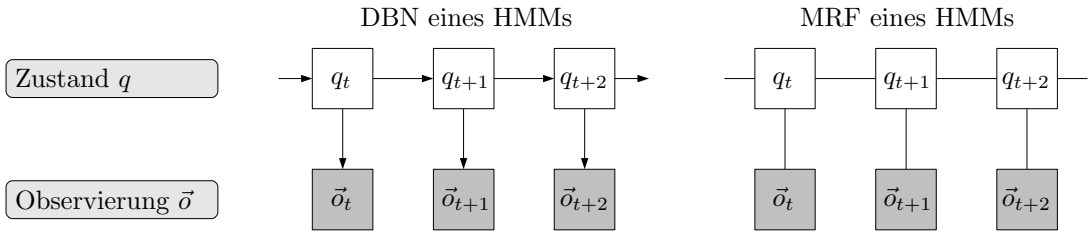

Abbildung 2.14: Das Hidden Markov Modell als dynamisches Bayes'sches Netz und als Markov Random Field. Der Zustand modelliert den zeitlichen Ablauf durch eine Markov-Kette. In einem zweiten stochastischen Prozess emittiert der Zustand zu jedem Zeitpunkt ein Symbol.

wahrscheinlichkeit als $\underset{\sim}{B}_i(\vec{o}) = p(\vec{o}_t = \vec{o} \,|\, q_t = i)$ definiert, ergibt sich für die Faktorisierung

$$p(\vec{o}, \vec{q}) = \pi_{q_1} \; \underset{\sim}{B}_{q_1}(\vec{o}_1) \; \prod_{t=2}^{T} a_{q_{t-1},q_t} \; \underset{\sim}{B}_{q_t}(\vec{o}_t) \tag{2.77}$$

die bekannte HMM Gleichung [126].

Abbildung 2.14 (rechts) zeigt, dass das HMM auch als MRF beschrieben werden kann. Dieses faktorisiert die Verbundwahrscheinlichkeit der Beobachtung $\vec{o}$ und der Zustandssequenz $\vec{q}$ zu:

$$p(\vec{o}, \vec{q}) = \frac{1}{Z}\psi(q_1, \vec{o}_1) \; \prod_{t=2}^{T} \psi(q_{t-1}, q_t) \; \psi(q_t, \vec{o}_t), \tag{2.78}$$

wobei die Potentialfunktionen ψ mit den maximalen Cliquen des DBN übereinstimmen. Das MRF eines HMM entspricht dem moralisierten DBN. Das HMM ist daher ein Beispiel eines triangulierten, perfekten Graphens: Es ist sowohl kausal als auch verknüpft beschreibbar.

Nachrichtenpropagierung im Hidden Markov Modell

Abbildung 2.15 (links) zeigt eine mögliche Triangulation des HMM-Graphens und den sich daraus ergebenden Verbundbaum (rechts). An diesem Verbundbaum soll gezeigt werden, dass der HMM Vorwärts-Rückwärts-Algorithmus [14] ein Spezialfall der allgemeinen Nachrichtenpropagierung in BN ist. In dem gezeigten triangulierten Graphen sind die eingefügten Kanten zwischen q_{t-1} und $\vec{o}_t$ im Triangulationsprozess prinzipiell nicht notwendig, da ein HMM bereits trianguliert ist. Das zusätzliche Einfügen von Kanten in den moralisierten Graphen ist jedoch immer möglich ohne die Korrektheit der Berechnung zu beeinflussen (der Graph repräsentiert weniger Unabhängigkeiten, vgl. dazu Abschnitt 2.8.3). Mit dieser gewählten Triangulation kann jedoch eine Analogie zu dem bekannten HMM-Algorithmus hergestellt werden.

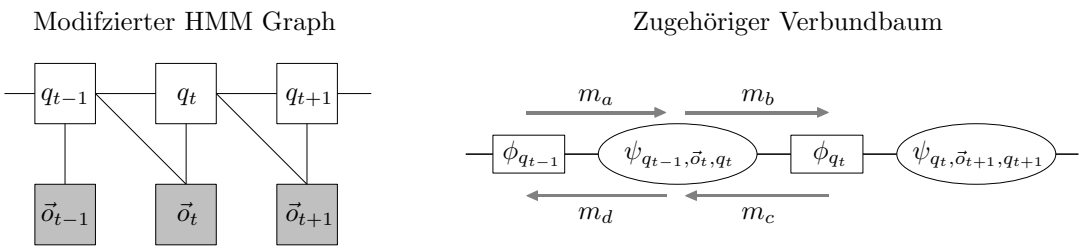

Abbildung 2.15: Triangulierter Graph des HMM DBN (links). Dabei werden während der Triangulation jeweils zwischen q_{t-1} und $\vec{o}_t$ zusätzliche, nicht notwendige Kanten eingefügt, um eine Analogie zum bekannten Vorwärts-Rückwärts-Algorithmus herzustellen. Die Korrektheit der Berechnung wird nicht beeinflusst, da der gezeigte Graph weniger Unabhängigkeitsaussagen repräsentiert. Mit der gewählten Triangulation ergibt sich der rechts gezeigte Verbundbaum, bei dem die Beobachtung $\vec{o}_t$, der Zustand q_t und der Vorgängerzustand q_{t-1} eine maximale Clique bilden. Die Cliquen sind dann jeweils durch den aktuellen Zustand q_t separiert und es ergeben sich nur vier verschiedene zu übertragende Nachrichten m_a, m_b, m_c und m_d.

Mit der gewählten Triangulation ergibt sich der in Abbildung 2.15 (rechts) gezeigte Verbundbaum. Dabei bilden im ersten Zeitschritt der Zustand q_1 und die Beobachtung $\vec{o}_1$ eine maximale Clique. Alle weiteren maximalen Cliquen bestehen aus der aktuellen Beobachtung $\vec{o}_t$, dem aktuellen Zustand q_t und dem Vorgängerzustand q_{t-1}. Benachbarte Cliquen werden jeweils durch den aktuellen Zustand q_t separiert. Da es sich bei dem Verbundbaum um einen einfachen, unverzweigten Baum handelt, gibt es bei der globalen Nachrichtenausbreitung nur vier verschiedene Nachrichten m_a, m_b, m_c und m_d zwischen den Cliquen und Separatoren. Die folgende Analyse der Nachrichten ist in ihren Grundzügen [23] entnommen, hier aber detaillierter erläutert.

Die erste maximale Clique enthält den ersten Zustand und die erste Beobachtung und hat das Potential

$$\psi_{q_1,\vec{o}_1}(q_1, \vec{o}_1). \tag{2.79}$$

Alle weiteren maximalen Cliquen enthalten den aktuellen sowie den vorherigen Zustand und die aktuelle Beobachtung. Sie haben daher die Potentialfunktion

$$\psi_{q_{t-1},\vec{o}_t,q_t}(q_{t-1}, \vec{o}_t, q_t). \tag{2.80}$$

Zur Initialisierung werden diese Cliquepotentiale zuerst mit dem Wert eins belegt. Danach wird jede WDF – wie in Abschnitt 2.8.3 beschrieben – in das ihr zugehörige Cliquepotential multipliziert. Damit ergibt sich für die Cliquepotentiale nach der

Initialisierung:

$$\psi_{q_1,\vec{o}_1}(q_1,\vec{o}_1) = p(q_1)\; p(\vec{o}_1 \mid q_1), \tag{2.81}$$

$$\psi_{q_{t-1},\vec{o}_t,q_t}(q_{t-1},\vec{o}_t,q_t) = p(q_t \mid q_{t-1})\; p(\vec{o}_t \mid q_t). \tag{2.82}$$

Alle Separatoren enthalten nur den aktuellen Zustand. Sie haben daher das Separatorpotential

$$\phi_{q_t}(q_t). \tag{2.83}$$

Diese Separatoren werden zur Initialisierung nur mit dem Wert eins belegt. Ein weiteres Absorbieren von WDF in die Separatoren ist nicht notwendig, da alle Verteilungen bereits von den Cliquen aufgenommen wurden. Dann ergibt sich nach der Initialisierung für alle Separatorpotentiale:

$$\phi_{q_t}(q_t) = 1. \tag{2.84}$$

Vorwärtspfad: Nach dieser Initialisierung wird der Baum durch die globale Ausbreitung aktualisiert. Dazu werden vier verschiedene Nachrichten in zwei Richtungen zwischen den Separatoren und Cliquen ausgetauscht. Ein Separator aktualisiert die nachfolgende Clique mit der Nachricht m_a, die sich aus der Absorptionsgleichung (2.49) ergibt:

$$\psi^*_{q_{t-1},\vec{o}_t,q_t}(q_{t-1},\vec{o}_t,q_t) = \frac{\phi^*_{q_{t-1}}(q_{t-1})}{\phi_{q_{t-1}}(q_{t-1})}\; \psi_{q_{t-1},\vec{o}_t,q_t}(q_{t-1},\vec{o}_t,q_t). \tag{2.85}$$

Einsetzen der initialisierten Separator- und Cliquepotentiale aus Gleichung (2.82) und (2.84) ergibt für die zu aktualisierende Clique:

$$\begin{aligned} \psi^*_{q_{t-1},\vec{o}_t,q_t}(q_{t-1},\vec{o}_t,q_t) &= \frac{\phi^*_{q_{t-1}}(q_{t-1})}{1}\; \psi_{q_{t-1},\vec{o}_t,q_t}(q_{t-1},\vec{o}_t,q_t) \\ &= \phi^*_{q_{t-1}}(q_{t-1})\; \psi_{q_{t-1},\vec{o}_t,q_t}(q_{t-1},\vec{o}_t,q_t) \\ &= \phi^*_{q_{t-1}}(q_{t-1})\; p(q_t \mid q_{t-1})\; p(\vec{o}_t \mid q_t). \end{aligned} \tag{2.86}$$

Nach der Aktualisierung der Clique wird die neue Information mit der Nachricht m_b an den nächsten Separator weitergereicht. Aus Gleichung (2.48) ergibt sich für diese Nachricht m_b von der Clique zum Separator die Doppelsumme über alle N möglichen vorherigen Zustände und alle theoretisch möglichen Beobachtungen

$$\phi^*_{q_t}(q_t) = \sum_{q_{t-1}} \sum_{\vec{o}_t} \psi^*_{q_{t-1},\vec{o}_t,q_t}(q_{t-1},\vec{o}_t,q_t). \tag{2.87}$$

Einsetzen des aktualisierten Cliquepotentials aus Gleichung (2.86) ergibt für den Separator:

$$\phi^*_{q_t}(q_t) = \sum_{q_{t-1}} \sum_{\vec{o}_t} \phi^*_{q_{t-1}}(q_{t-1})\; p(q_t \mid q_{t-1})\; p(\vec{o}_t \mid q_t). \tag{2.88}$$

Die aktuelle Beobachtung $\vec{o}_t$ ist aus der Merkmalsequenz bekannt. Diese Beobachtungsevidenz kann mittels einer Diracfunktion $p(\vec{o}_t \,|\, q_t)\ \delta[\vec{o}_t]$ eingebunden werden und es ergibt sich

$$\phi^*_{q_t}(q_t) = \sum_{q_{t-1}} \sum_{\vec{o}_t} \phi^*_{q_{t-1}}(q_{t-1})\ p(q_t \,|\, q_{t-1})\ p(\vec{o}_t \,|\, q_t)\ \delta[\vec{o}_t]. \tag{2.89}$$

Durch die Einbindung der Diracfunktion für die bekannte Beobachtung entfällt die Marginalisierungskomponente über $\vec{o}_t$ und es ergibt sich als Aktualisierungsregel für den Separator

$$\phi^*_{q_t}(q_t) = \sum_{q_{t-1}} \phi^*_{q_{t-1}}(q_{t-1})\ p(q_t \,|\, q_{t-1})\ p(\vec{o}_t \,|\, q_t). \tag{2.90}$$

Mit der üblichen HMM Notation für die Übergangswahrscheinlichkeit $a_{ij} = p(q_t = j \,|\, q_{t-1} = i)$ und der diskreten Emissionmatrix $\underset{\sim}{B}_i(\vec{o}) = p(\vec{o}_t = \vec{o} \,|\, q_t = i)$ wird die Aktualisierungsregel zu:

$$\phi^*_{q_t}(q_t) = \sum_{i=1}^{N} \phi^*_{q_{t-1}}(q_{t-1} = i)\ a_{ij}\ \underset{\sim}{B}_j(\vec{o}). \tag{2.91}$$

Nach Umbenennung des Separatorpotentials in $\alpha_t(j) = \phi^*_{q_t}(q_t)$ ergibt sich

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i)\ a_{ij}\ \underset{\sim}{B}_j(\vec{o}) \tag{2.92}$$

aus der Nachrichtenpropagierung im Verbundbaum des HMM, der Vorwärts-Algorithmus [126].

Rückwärtspfad: Dies kann analog für die rückwärtigen Nachrichten m_c und m_d im Verbundbaum durchgeführt werden. Der Separator aktualisiert eine vorherige Clique mit einer rückwärtsgerichteten Nachricht m_c, die sich aus Gleichung (2.49) ergibt:

$$\psi^*_{q_{t-1},\vec{o}_t,q_t}(q_{t-1}, \vec{o}_t, q_t) = \frac{\phi^*_{q_t}(q_t)}{\phi_{q_t}(q_t)}\ \psi_{q_{t-1},\vec{o}_t,q_t}(q_{t-1}, \vec{o}_t, q_t). \tag{2.93}$$

Einsetzen der initialisierten Separator- und Cliquepotentiale aus Gleichung (2.82) und (2.84) ergibt für die zu aktualisierende Clique:

$$\psi^*_{q_{t-1},\vec{o}_t,q_t}(q_{t-1}, \vec{o}_t, q_t) = \phi^*_{q_t}(q_t)\ p(q_t \,|\, q_{t-1})\ p(\vec{o}_t \,|\, q_t). \tag{2.94}$$

Nach der Aktualisierung der Clique wird die neue Information mit der Nachricht m_d an den vorausgehenden Separator weitergereicht. Die rückwärtsgerichtete Nachricht m_d von der Clique zum Separator ergibt sich dabei wieder aus Gleichung (2.48)

als Doppelsumme über alle N möglichen Zustände und alle theoretisch möglichen Beobachtungen

$$\phi^*_{q_{t-1}}(q_{t-1}) = \sum_{q_t} \sum_{\vec{o}_t} \psi^*_{q_{t-1},\vec{o}_t,q_t}(q_{t-1}, \vec{o}_t, q_t). \tag{2.95}$$

Einsetzen des aktualisierten Cliquepotentials aus Gleichung (2.94) ergibt für den Separator:

$$\phi^*_{q_{t-1}}(q_{t-1}) = \sum_{q_t} \phi^*_{q_t}(q_t)\ p(q_t \,|\, q_{t-1})\ p(\vec{o}_t \,|\, q_t), \tag{2.96}$$

wobei die Marginalisierung über $\vec{o}_t$ wiederum entfällt, weil die Beobachtung $\vec{o}_t$ aus der Merkmalsequenz als Evidenz vorliegt und daher über eine Diracfunktion eingebunden werden kann. Mit den üblichen HMM Bezeichnungen ergibt sich aus dieser Aktualisierungsregel:

$$\phi^*_{q_{t-1}}(q_{t-1}) = \sum_{j=1}^{N} \phi^*_{q_t}(q_t = j)\ a_{ij}\ \underset{\sim}{B}_j(\vec{o}_t). \tag{2.97}$$

Nach Umbenennung des Separatorpotentials in $\beta_t(i) = \phi^*_{q_t}(q_t)$ und Umformung ergibt sich

$$\beta_{t-1}(i) = \sum_{j=1}^{N} a_{ij}\ \underset{\sim}{B}_j(\vec{o}_t)\ \beta_t(j) \tag{2.98}$$

aus der Nachrichtenpropagierung im Verbundbaum des HMM, die Rückwärtsrekursion [126].

Der HMM Vorwärts-Rückwärts-Algorithmus [14] ist daher ein Spezialfall der allgemeinen Nachrichtenpropagierung in BN, wenn als Verbundbaum die in Abbildung 2.15 gezeigte Triangulation des HMM gewählt wird. Für diese Triangulation ergibt sich aus der Nachrichtenpropagierung automatisch die effektive Vorwärts- und Rückwärtsberechnung. Andere Triangulationen des HMM führen zu anderen Propagierungen innerhalb des HMM, z. B. der γ-Rekursion [23].

2.10.3 HMM mit erweiterten Emissionsarten

Im vorherigen Abschnitt wurde gezeigt, wie mit Hilfe von DBN ein HMM umgesetzt werden kann und dass der darauf ablaufende JT-Algorithmus dem bekannten Vorwärts-Rückwärts-Algorithmus entspricht. Dabei wurden bisher nur diskrete HMM behandelt, bei denen die Emissionswahrscheinlichkeit durch eine Matrix $\underset{\sim}{B}_i(\vec{o}_t) = p(\vec{o}_t \,|\, q_t)$, also für einen Satz von diskreten Symbolen, beschrieben wird. Hierzu sind verschiedene Erweiterungen bekannt, um z. B. auch kontinuierliche Observierungen modellieren zu können. Diese Erweiterungen können ebenfalls durch DBN beschrieben werden. Außerdem ist es mit DBN möglich, verschiedene statische Klassifikatoren mit der dynamischen Markov-Kette zu verbinden und so neue HMM-Arten mit problemangepasster Merkmalmodellierung zu entwickeln. Dabei dienen

die statischen Modelle als Baustein zur Modellierung der Merkmale und die Markov-Kette als Baustein zur Modellierung der zeitlichen Abhängigkeit. Drei so gebildete Modelle sind in Abbildung 2.16 gezeigt.

Kontinuierliches Hidden Markov Modell

Werden die diskreten Emissionsknoten durch Gaußknoten ersetzt, ergibt sich ein kontinuierliches HMM (Abbildung 2.16, links). Dieses hat die gleiche Struktur wie ein diskretes HMM und faktorisiert deswegen identisch. Die Emissionswahrscheinlichkeit ist nun jedoch nicht mehr mit einer Emissionsmatrix $\underset{\sim}{B}_i$, sondern für jeden Zustand mit einer Gaußkurve $p(\vec{o}_t \,|\, q_t) = \mathcal{N}(\vec{o}_t, \vec{\mu}_{q_t}, \underset{\sim}{\Sigma}_{q_t})$ beschrieben, so dass sich für die Verbundwahrscheinlichkeit der Zustände und der Beobachtung

$$p(\vec{o}, \vec{q}) = p(q_1)\, \mathcal{N}(\vec{o}_1, \vec{\mu}_{q_1}, \underset{\sim}{\Sigma}_{q_1}) \prod_{t=2}^{T} p(q_t \,|\, q_{t-1})\, \mathcal{N}(\vec{o}_t, \vec{\mu}_{q_t}, \underset{\sim}{\Sigma}_{q_t}) \tag{2.99}$$

ergibt.

Hidden Markov Modell mit Gauß Mixtur Modellierung

Durch Einführen eines zusätzlichen Mixtur-Knotens wird jede Observierung durch mehrere Gaußkurven erklärt und es ergibt sich ein HMM mit Gauß Mixtur Modellierung (Abbildung 2.16, rechts). Bei einem Modell mit M Mixturen faktorisiert das HMM dann zu

$$p(\vec{o}, \vec{q}) = p(q_1) \sum_{m=1}^{M} p(\vec{o}_1 \,|\, q_1, m)\, p(m \,|\, q_1) \prod_{t=2}^{T} p(q_t \,|\, q_{t-1}) \sum_{m=1}^{M} p(\vec{o}_t \,|\, q_t, m)\, p(m \,|\, q_t), \tag{2.100}$$

wobei die Mixturen hier bereits aus der Verbundwahrscheinlichkeit herausmaginalisiert wurden, da es überlicherweise nicht von Interesse ist, welche Gaußkurve die Observierung tatsächlich erzeugt hat. Werden wieder die üblichen HMM-Bezeichnungen [126] verwendet, die Mixturgewichte – wie aus dem GMM bekannt – mit $\alpha_m^{q_t} = p(m \,|\, q_t)$ bezeichnet und alle Mixturen als Gaußkurven $p(\vec{o}_t \,|\, q_t, m) = \mathcal{N}(\vec{o}_t, \vec{\mu}_m^{q_t}, \underset{\sim}{\Sigma}_m^{q_t})$ modelliert, ergibt sich als Faktorisierung

$$p(\vec{o}, \vec{q}) = \pi_{q_1} \left(\sum_{m=1}^{M} \alpha_m^{q_1}\, \mathcal{N}(\vec{o}_t, \vec{\mu}_m^{q_1}, \underset{\sim}{\Sigma}_m^{q_1}) \right) \prod_{t=2}^{T} a_{q_{t-1}, q_t} \left(\sum_{m=1}^{M} \alpha_m^{q_t}\, \mathcal{N}(\vec{o}_t, \vec{\mu}_m^{q_t}, \underset{\sim}{\Sigma}_m^{q_t}) \right), \tag{2.101}$$

die bekannte Gleichung für ein HMM mit GMM Emissionen. Häufig erhält dabei nicht jeder Zustand q_t eigene Mixturen $\mathcal{N}(\vec{o}_t, \vec{\mu}_m^{q_t}, \underset{\sim}{\Sigma}_m^{q_t})$, sondern alle Zustände teilen sich insgesamt M Mixturen $\mathcal{N}(\vec{o}_t, \vec{\mu}_m, \underset{\sim}{\Sigma}_m)$. Dann sind zwar die Mixtur-Gewichte $\alpha_m^{q_t} = p(m \,|\, q_t)$ abhängig vom aktuellen Zustand q_t und legen die Gewichtung der

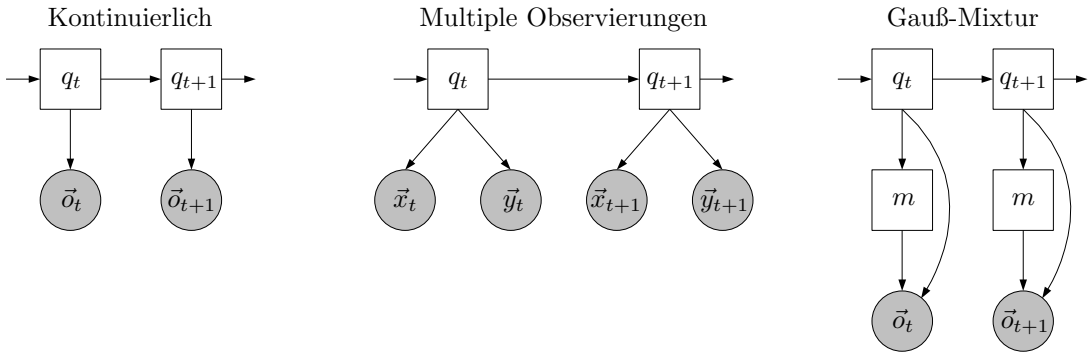

Abbildung 2.16: Erweiterungen des HMM um verschiedene Emissionsformen. Kontinuierliche HMM (links) verwenden Gaußkurven, um die Emission zu modellieren. Die Verbindung des naiven Bayes mit dem HMM (Mitte) führt zu einem Modell, bei dem mehrere Observierungen von einem Zustand erzeugt werden. Die einzelnen Observierungen sind dabei voneinander unabhängig. Prinzipiell können die Merkmale dabei auch unterschiedlich modelliert werden, z. B. gemischt diskret und kontinuierlich. Kontinuierliche HMM können mittels eines Mixturknotens so erweitert werden, dass jede Emission mit mehreren Gaußkurven modelliert wird (rechts).

vorhandenen Mixturen fest – wobei einige durchaus mit der Wahrscheinlichkeit null gewichtet sein können –, die Mixturen selbst werden aber über die Zeit und über alle Zustände geteilt. Dann ergibt sich die vereinfachte Faktorisierung

$$p(\vec{o}, \vec{q}) = \pi_{q_1} \left(\sum_{m=1}^{M} \alpha_m^{q_1} \, \mathcal{N}(\vec{o}_t, \vec{\mu}_m, \Sigma_m) \right) \prod_{t=2}^{T} a_{q_{t-1}, q_t} \left(\sum_{m=1}^{M} \alpha_m^{q_t} \, \mathcal{N}(\vec{o}_t, \vec{\mu}_m, \Sigma_m) \right). \tag{2.102}$$

Hidden Markov Modell mit multiplen Observierungen

Durch die Verbindung einer Markov-Kette mit dem naiven Bayes Klassifikator ergibt sich ein HMM mit multiplen Observierungen (Abbildung 2.16, Mitte). Dabei erzeugt jeder Zustand mehrere Observierungen, in dem gezeigten Modell die Merkmale $\vec{x}$ und $\vec{y}$; die Merkmale sind dabei voneinander unabhängig. Das Modell faktorisiert die Verbundwahrscheinlichkeit der beiden Merkmalsequenzen und der Zustände zu

$$p(\vec{x}, \vec{y}, \vec{q}) = p(q_1) \; p(\vec{x}_1 \,|\, q_1) \; p(\vec{y}_1 \,|\, q_1) \prod_{t=2}^{T} p(q_t \,|\, q_{t-1}) \; p(\vec{x}_t \,|\, q_t) \; p(\vec{y}_t \,|\, q_t). \tag{2.103}$$

In dem hier gezeigten Beispiel sind beide Merkmalsequenzen kontinuierlich modelliert, es ist jedoch auch möglich, verschiedene Merkmale unterschiedlich zu modellieren, z. B. gemischt kontinuierlich und diskret. Dabei kann z. B. auch wieder ein

GMM eingebunden werden, so dass ein Merkmal z. B. mit nur einer Gaußkurve, das zweite Merkmal diskret und das dritte Merkmal mit einem GMM modelliert wird. HMM mit multiplen, problemangepassten Observierungen können so den vorhanden Daten angepasst werden. Eine Vektorquantisierung oder ein Neusamplen von Merkmalen für eine frühe Fusion kann entfallen, da nicht die Daten dem Modell, sondern das Modell den Daten angepasst wird.

2.10.4 HMM mit erweiterten Markov-Ketten

Bisher wurde gezeigt, wie HMM als DBN dargestellt werden können und wie mittels Modellbildung aus Bausteinen die Emissionswahrscheinlichkeit durch verschiedene statische Modelle verändert werden kann. Durch das Austauschen der Observierungsknoten konnten so verschiedene, an die Merkmale angepaßte, HMM erzeugt werden. Es ist jedoch nicht nur möglich, die Observierung eines HMM zu verändern: DBN bieten die Flexibilität, auch den dynamischen Prozess durch verschiedene „zeitliche Bausteine“ zu verändern, um an das Problem angepasst zu werden. Drei solche zeitlich veränderte Modelle sind in Abbildung 2.17 gezeigt.

Faktorielles Hidden Markov Modell

Das faktorielle HMM [68] (Abbildung 2.17, links) verwendet mehrere parallel ablaufende Markov-Ketten, um die Observierung zu erzeugen. Die Ketten interagieren dabei nicht, erzeugen aber alle zusammen in jedem Zeitschritt ein Merkmal. Die Merkmalsequenz wird daher durch mehrere parallel laufende dynamische Prozesse erklärt. Bei M Prozessen faktorisiert das Modell die Verbundwahrscheinlichkeit der Observierung und der Markov-Ketten zu:

$$p(\vec{o}, \vec{q}^1, \ldots, \vec{q}^M) = \prod_{m=1}^{M} p(q_1^m) \prod_{t=2}^{T} p(q_t^m \mid q_{t-1}^m) \prod_{t=1}^{T} p(\vec{o}_t \mid q_t^1, \ldots, q_t^M). \tag{2.104}$$

Durch die Verwendung mehrerer paralleler Prozesse sind faktorielle HMM sehr flexibel in der Anwendung und können genutzt werden, um reale Prozesse der Merkmalerzeugung abzubilden. Durch die Faktorisierung verkleinert sich außerdem der benötigte Zustandsraum. Um den gleichen Zustandsraum aufzuspannen, müsste ein normales HMM mit einer deutlich höheren Anzahl an Zuständen ausgestattet werden [113]. Auf der anderen Seite steigt die benötigte Rechenzeit exponentiell mit der Anzahl der verwendeten Markov-Ketten, wodurch faktorielle HMM in der Praxis schnell unberechenbar werden.

Input-Output Hidden Markov Modell

Das Input-Output HMM [20] (Abbildung 2.17, Mitte) wird verwendet, um die Verbundwahrscheinlichkeit von zwei Merkmalsequenzen $\vec{x}$ und $\vec{y}$ zu modellieren, die

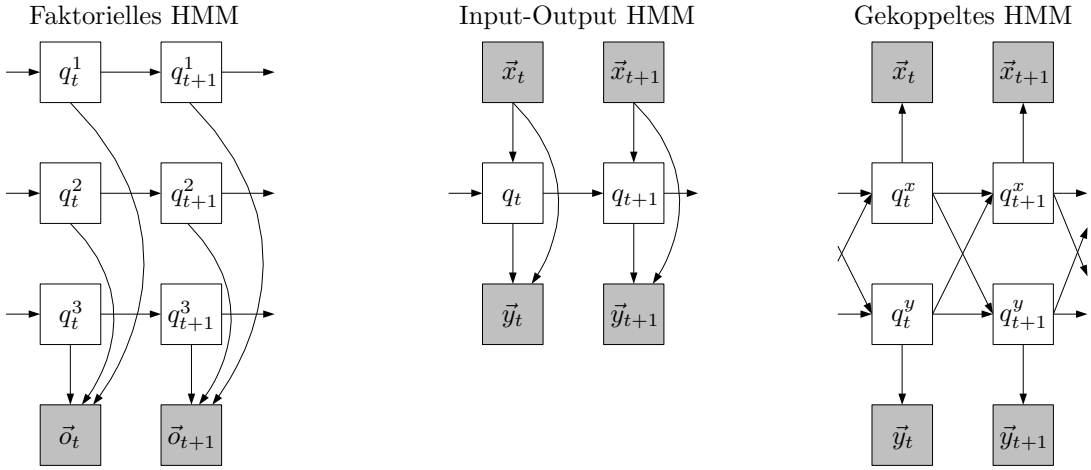

Abbildung 2.17: Erweiterungen des HMM mit verschiedenen Markov-Ketten. Das faktorielle HMM (links) verwendet parallel ablaufende Markov-Ketten, um eine Beobachtung zu erzeugen, ohne dass die Markov-Ketten aufeinander Einluss haben. Jedes Merkmal wird daher durch mehrere zeitlich verknüpfte, aber unabhängige Ursachen erklärt. Das Input-Output HMM (Mitte) verwendet zwei Merkmalsequenzen, wobei die erste Sequenz als Eingang des Systems betrachtet wird und den aktuellen Zustand mitbestimmt. Der Zustand erzeugt dann wiederum – wie ein normales HMM – die zweite Merkmalsequenz. Beim gekoppelten HMM (rechts) laufen zwei HMM parallel ab. Jedes HMM erzeugt dabei ein eigene Merkmalsequenz. Die Zustände beider HMM sind jedoch miteinander gekoppelt und beeinflussen sich daher wechselseitig.

beide die gleiche Länge T aufweisen müssen. Die Merkmalsequenz $\vec{x}$ wird als Eingang des Systems betrachtet und bestimmt den Zustand q_t des Modells mit. Die Markov-Kette hängt daher nicht nur vom vorherigen Zustand q_{t-1}, sondern auch von der augenblicklichen Observierung $\vec{x}_t$ des Eingangs ab. Die Markov-Kette selbst erzeugt dann aber wieder – wie in einem gewöhnlichen HMM – eine Merkmalsequenz $\vec{y}$, die als Ausgang des Systems betrachtet wird. Das Modell faktorisiert die beiden Merkmals- und die Zustandsequenz daher zu

$$p(\vec{x}, \vec{y}, \vec{q}) = p(q_1 \mid \vec{x}_1)\; p(\vec{y}_1 \mid q_1, \vec{x}_1) \prod_{t=2}^{T} p(q_t \mid q_{t-1}, \vec{x}_t)\; p(\vec{y}_t \mid q_t, \vec{x}_t). \tag{2.105}$$

Vom Input-Output HMM sind zwei verschiedene Ausführungen bekannt. Bei der hier gezeigten hängt der Systemausgang $\vec{y}$ nicht nur vom aktuellen Zustand q_t, sondern auch vom Systemeingang $\vec{x}_t$ ab. Die Sequenz $\vec{y}$ ist daher sehr stark durch den Eingang $\vec{x}$ bestimmt. In einer zweiten Ausführung gibt es keine Verbindung zwischen den Knoten x und y. Die Merkmalsequenz $\vec{y}$ wird nur durch die Markov-Kette bestimmt und ist daher etwas weniger stark mit dem Systemeingang $\vec{x}$ verbunden.

Input-Output HMM werden vor allem zur Fusion von stark korrelierten Daten, z. B. der audio-visuellen Spracherkennung [17, 20], eingesetzt. Sie setzen jedoch voraus, dass beide Merkmalsequenzen korreliert und synchron zueinander sind.

Gekoppeltes Hidden Markov Modell

Das gekoppelte (*coupled*) HMM [36] (Abbildung 2.17, rechts) verwendet zwei parallele HMM, um zwei Merkmalsequenzen $\vec{x}$ und $\vec{y}$ zu modellieren. Im Gegensatz zum faktoriellen HMM interagieren die beiden parallelen Markov-Ketten beim gekoppelten HMM jedoch stark. Der Zustand q_t^x des ersten HMM hängt nicht nur vom vorherigen Zustand q_{t-1}^x ab, sondern auch vom vorherigen Zustand q_{t-1}^y des zweiten HMM. Umgekehrt hängt der Zustand q_t^y des zweiten HMM ebenfalls nicht nur vom vorherigen Zustand q_{t-1}^y ab, sondern auch vom vorherigen Zustand q_{t-1}^x des ersten HMM. Das Modell faktorisiert die Verbundwahrscheinlichkeit der beiden Merkmal- $\vec{x}$ und $\vec{y}$ und der beiden Zustandsequenzen $\vec{q}^x$ und $\vec{q}^y$ deshalb zu

$$\begin{aligned} p(\vec{x}, \vec{y}, \vec{q}^x, \vec{q}^y) = {} & p(q_1^x)\; p(q_1^y)\; p(\vec{x}_1 \mid q_1^x)\; p(\vec{y}_1 \mid q_1^y) \\ & \prod_{t=2}^{T} p(q_t^x \mid q_{t-1}^x, q_{t-1}^y)\; p(q_t^y \mid q_{t-1}^x, q_{t-1}^y)\; p(\vec{x}_t \mid q_t^x)\; p(\vec{y}_t \mid q_t^y). \end{aligned} \tag{2.106}$$

Das gekoppelte HMM wird hauptsächlich zur Datenfusion eingesetzt. Die Daten müssen dabei nicht so stark wie beim Input-Output HMM korreliert sein. Jedoch setzt auch das gekoppelte HMM voraus, dass beide Merkmalsequenzen die gleiche Länge aufweisen und zudem synchron zueinander sind. Theoretisch ist es möglich, das Modell um weitere Markov-Ketten und Merkmalsequenzen zu erweitern. Das Modell ist dann jedoch praktisch nicht mehr vollständig berechenbar und es müssen Näherungslösungen verwendet werden [169]. Gekoppelte HMM werden daher fast ausschließlich mit nur zwei Merkmalsequenzen verwendet.

Andere Erweiterungen für Hidden Markov Modelle

Weitere Erweiterungen des HMM sind z. B. das hierarchische, das auto-regressive, das asynchrone oder das multi-stream HMM.

Die Zustände des hierarchischen HMM (HHMM) [61] emittieren keine Symbole, sondern bis zu einer vorgegebenen Tiefe wiederum hierachische HMM. Erst in der letzten Stufe emittieren die Zustände dann Symbole. Auf diese Weise können mehrschichtige Prozesse oder ganze Grammatiken vereinfacht abgebildet werden. Ein GM des HHMM ist in [114] gezeigt. In praktischen Anwendungen enthalten HHMM so viele Freiheitsgrade, dass für das Training sehr viel Daten vorliegen müssen.

Beim auto-regressiven HMM [113] ist nicht nur die Markov-Kette, sondern auch die Beobachtung über die Zeit verbunden, d. h. zwischen den Knoten $\vec{y}_t$ und $\vec{y}_{t+1}$ wird

ein zusätzlicher Pfeil eingefügt. Dadurch sind zwei aufeinanderfolgende Beobachtungen bei gegebenem Zustand nicht mehr unabhängig voneinander. In praktischen Anwendungen wird das auto-regressive HMM jedoch kaum eingesetzt.

Das asynchrone HMM (AHMM) [17] wird verwendet, um Merkmalströme, die unterschiedlich lang und asynchron sein können, zu modellieren. Dabei findet das Modell die optimale zeitliche Anordnung der beiden Ströme zueinander. Das AHMM wird in dieser Arbeit in Kapitel 4 ausführlich erläutert.

Multi-Stream HMM [166] werden ebenfalls verwendet, um mehrere Merkmalsequenzen miteinander zu fusionieren. Sie sind nicht so flexibel wie AHMM, dafür jedoch leichter zu berechnen. Sie werden in dieser Arbeit in Kapitel 6 ausführlich erläutert.

Für verschiedene Anwendungen und Spezialfälle sind weitere graphische Modellerweiterungen zu HMM bekannt, auf sie soll hier nicht eingegangen werden. Beispiele und weitere Quellen dazu finden sich z. B. in [113, 169].

2.10.5 Lineare dynamische Systeme

Lineare dynamische Systeme (*linear dynamical systems, Gaussian state-space models, LDS*) [66] haben in ihrer Grundform die gleiche Struktur wie ein HMM, dabei jedoch nur kontinuierliche Knoten [134]. Abbildung 2.18 (links) zeigt ein solches LDS: Die beobachtete Systemausgabe $\vec{y}_t$ hängt dabei nur vom augenblicklichen, verborgenen Systemzustand $\vec{x}_t$ ab. Im Gegensatz zum HMM ist dieser Zustand nicht diskret, sondern ein reeller Vektor $\vec{x}_t \in \mathbb{R}^N$ der Dimension N. Wiederum analog zum HMM ist der Systemzustand $\vec{x}_t$ jedoch nur abhängig vom vorherigen Zustand $\vec{x}_{t-1}$, wobei die Systemübergänge mittels Gauß'schen Wahrscheinlichkeiten beschrieben werden können.

Durch die identische Struktur faktorisiert das gezeigte LDS die Verbundwahrscheinlichkeit von Zustand $\vec{x}$ und Systemausgabe $\vec{y}$ identisch wie ein HMM den diskreten Zustand und die Beobachtung faktorisiert:

$$p(\vec{x}, \vec{y}) = p(\vec{x}_1)\; p(\vec{y}_1 \mid \vec{x}_1)\; \prod_{t=2}^{T} p(\vec{x}_t \mid \vec{x}_{t-1})\; p(\vec{y}_t \mid \vec{x}_t). \tag{2.107}$$

Jedoch sind die Übergangs- und Ausgabewahrscheinlichkeiten durch Gaußprozesse beschrieben:

$$p(\vec{x}_t \mid \vec{x}_{t-1}) = \mathcal{N}(\vec{x}_t, \underset{\sim}{A}\vec{x}_{t-1}, \underset{\sim}{Q}), \tag{2.108}$$

$$p(\vec{y}_t \mid \vec{x}_t) = \mathcal{N}(\vec{y}_t, \underset{\sim}{C}\vec{x}_t, \underset{\sim}{R}), \tag{2.109}$$

wobei $\underset{\sim}{A}$ die Systemzustandsübergangs- und $\underset{\sim}{C}$ die Systemausgabematrix ist und $\underset{\sim}{Q}$ bzw. $\underset{\sim}{R}$ das System- und Beobachtungsrauschen beschreibt. Das LDS kann mit den

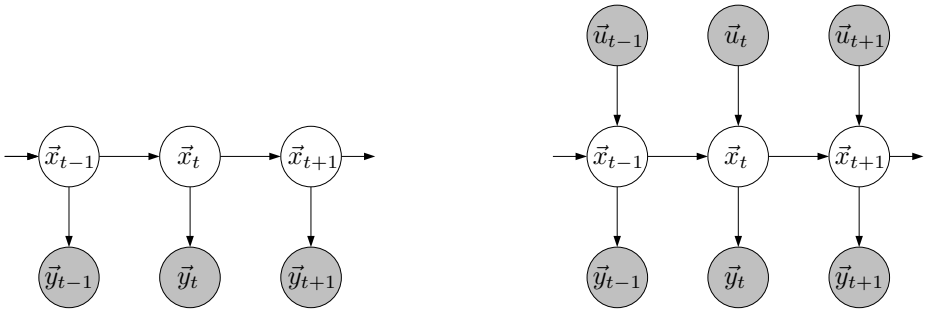

Abbildung 2.18: Lineare dynamische Systeme: Bei beiden Modellen wird die Systemausgabe durch einen verborgenen, kontinuierlichen Systemzustandsvektor erzeugt. Links hängt der Systemzustand nur vom vorherigen Zustand ab. Rechts hängt der Systemzustand sowohl vom vorherigen Zustand als auch von einer Systemeingabe ab. Beide Strukturen werden auch als Kalman Modelle bezeichnet, da der Kalman Filter den Vorwärtspfad in diesen Modellen beschreibt.

gleichen Parametern auch über die Zustandsgleichungen

$$\vec{x}_t = \underset{\sim}{A}\vec{x}_{t-1} + \vec{v}_t, \tag{2.110}$$

$$\vec{y}_t = \underset{\sim}{C}\vec{x}_t + \vec{w}_t \tag{2.111}$$

beschrieben werden. Dabei sind $\vec{v}_t = \mathcal{N}(\vec{v}_t, \vec{0}, \underset{\sim}{Q})$ und $\vec{w}_t = \mathcal{N}(\vec{w}_t, \vec{0}, \underset{\sim}{R})$ das System- und Beobachtungsrauschen. Normalerweise werden dabei – wiederum analog zum HMM – alle Parameter des LDS $\underset{\sim}{A}$, $\underset{\sim}{C}$, $\underset{\sim}{Q}$ und $\underset{\sim}{R}$ zeitinvariant modelliert.

Durch Einführen eines weiteren Knotens kann das LDS so erweitert werden, dass der Systemzustand $\vec{x}_t$ nicht nur vom vorherigen Zustand $\vec{x}_{t-1}$, sondern auch von einer Systemeingabe $\vec{u}_t$ abhängt. Diese Modellstruktur ist in Abbildung 2.18 (rechts) gezeigt. Dieses Modell faktorisiert die Verbundwahrscheinlichkeit von Systemeingabe $\vec{u}$, Zustand $\vec{x}$ und Systemausgabe $\vec{y}$ zu:

$$p(\vec{u}, \vec{x}, \vec{y}) = p(\vec{x}_1 \mid \vec{u}_1)\; p(\vec{y}_1 \mid \vec{x}_1) \prod_{t=2}^{T} p(\vec{x}_t \mid \vec{x}_{t-1}, \vec{u}_t)\; p(\vec{y}_t \mid \vec{x}_t), \tag{2.112}$$

wobei die Systemeingabe, die Systemausgabe und die Systemübergangswahrscheinlichkeiten wiederum durch Gaußprozesse beschrieben werden:

$$p(\vec{x}_t \mid \vec{x}_{t-1}, \vec{u}_t) = \mathcal{N}(\vec{x}_t, \underset{\sim}{A}\vec{x}_{t-1} + \underset{\sim}{B}\vec{u}_t, \underset{\sim}{Q}) \tag{2.113}$$

$$p(\vec{y}_t \mid \vec{x}_t) = \mathcal{N}(\vec{y}_t, \underset{\sim}{C}\vec{x}_t, \underset{\sim}{R}), \tag{2.114}$$

mit der Systemzustandsübergangs- $\underset{\sim}{A}$, der Systemeingangs- $\underset{\sim}{B}$ und der Systemausgabematrix $\underset{\sim}{C}$ und dem System- $\underset{\sim}{Q}$ bzw. Beobachtungsrauschen $\underset{\sim}{R}$. Dieses System

kann durch die Zustandsgleichungen

$$\vec{x}_t = \underset{\sim}{A}\vec{x}_{t-1} + \underset{\sim}{B}\vec{u}_t + \vec{v}_t \tag{2.115}$$

$$\vec{y}_t = \underset{\sim}{C}\vec{x}_t + \vec{w}_t \tag{2.116}$$

beschrieben werden, wobei $\vec{v}_t = \mathcal{N}(\vec{v}_t, \vec{0}, \underset{\sim}{Q})$ und $\vec{w}_t = \mathcal{N}(\vec{w}_t, \vec{0}, \underset{\sim}{R})$ wieder das System- und Beobachtungsrauschen enthalten.

Nachrichtenpropagierung: der Kalman Filter und die Rauch Rekursion

Die Nachrichtenpropagierung in den beiden gezeigten Modellen ist das kontinuierliche Pendant zur HMM Vorwärts-Rückwärts-Berechnung, die in Abschnitt 2.10.2 gezeigt wurde. Werden die Vorwärts- und Rückwärtsformeln für das LDS explizit aus der allgemeinen Nachrichtenpropagierung in BN hergeleitet, ergeben sich für die Vorwärtsberechnung die Kalman Filter Formeln [85, 156] und für die Rückwärtsberechnung die Rauch Rekursion [128]. Beide Modelle in Abbildung 2.18 werden daher häufig auch als Kalman Modelle bezeichnet [23, 113].

Erweiterungen

Analog zum Input-Output HMM ist es bei der in Abbildung 2.18 rechts gezeigten Struktur auch möglich, die Systemausgabe $\vec{y}_t$ direkt abhängig von der Systemeingabe $\vec{u}_t$ zu modellieren. Dazu muss lediglich eine Verbindung zwischen den Knoten $\vec{u}$ und $\vec{y}$ eingefügt werden (die entsprechende Struktur ist für das Input-Output HMM in Abbildung 2.17 gezeigt). Das System erhält dann noch eine Matrix $\underset{\sim}{D}$, die den Eingang direkt auf den Ausgang abbildet. Dazugehörige Systemzustandsgleichungen sind z. B. in [113] hergeleitet.

Neuere Erweiterungen der LDS sind z. B. das wechselnde Kalman Filter Modell [111], das mit einem weiteren diskreten Zustand zwischen mehreren dynamischen Prozessen auswählt, oder das faktorielle wechselnde Kalman Filter Modell [160], das mehrere parallele Prozesse verwendet, um die Ausgabe zu beschreiben.

3

Schnitt- und Szenenerkennung

Moderne Computer – auch im Preissegment für Privatanwender – verfügen heute über mehrere hundert Gigabytes an Festplattenspeicherplatz. DVD-Brenner mit denen bis zu 9.4 Gigabytes an Daten auf DVD-Rohlinge gebrannt werden können, gehören zur Grundausstattung von heutigen Computern. Seit einigen Jahren sind Breitbandinternetverbindungen in Deutschland weit verbreitet. Immer mehr Anbieter stellen inzwischen auch Videos zum digitalen Abruf (*video-on-demand*) zur Verfügung. Daher stellt heutzutage das Abrufen, Speichern und Tauschen auch von sehr großen digitalen Videoarchiven kein Problem mehr da.

Problematischer ist jedoch das Auffinden des richtigen Videos im Datenbestand [42]. Bis heute findet diese Suche fast ausschließlich über den Namen des Filmes oder den Titel des Programms statt, z. B. „Episode 1734 einer Serie“. Diese Anfragen könnten deutlich vereinfacht werden, wenn die Videodatenbank auch intuitive Abfragen unterstützte. So wäre es wünschenswert, nach Personen, Handlungen oder bestimmten Ereignissen suchen zu können. Ein solches System könnte dann natürlichere Abfragen, wie „die Folge von Friends, in der Julia Roberts eine Gastrolle spielt“ oder „die Spielfilme in denen Tom Cruise und Matt Damon mitspielen“ verarbeiten.

Ein System, das solche intuitiven Anfragen unterstützt, benötigt jedoch vielfältige Informationen über das Programm, die Personen, die Orte und die Handlung. Diese können entweder über Metadaten in den Archiven zur Verfügung gestellt werden, wie es z. B. über den MPEG-7 Standard [103] versucht wird; oder sie werden automatisch aus dem Videoarchiv extrahiert. Wünschenswert ist eine automatische Extraktion, da sie eine manuelle Annotation der Videos überflüssig macht.

Das automatische Bereitstellen von Information aus den Videodaten erfordert jedoch Algorithmen aus vielen anspruchsvollen, bisher nicht gelösten Forschungsgebieten wie dem automatischen Indexieren [33, 71, 141, 142], der Personenidentifikation [2, 171], Sprach- [59] und Sprechererkennung [38], Sprachverstehen [154] und dem automatischen Zusammenfassen [137].

In diesem Kapitel wird einer der ersten Schritte für eine Analyse von Videodaten behandelt: das automatische Auffinden von Schnitten, Szenen und Kapiteln. Bishe-

rige Arbeiten behandeln die Erkennung von Schnitten und Szenen fast immer separat (ein Überblick und Vergleich aktueller Systeme wird in Abschnitt 3.2 gegeben). Schnitte und Szenenwechsel interagieren jedoch, da es z. B. fast keinen Szenenwechsel ohne einen Schnitt gibt.

Im Rahmen dieser Arbeit [12] wird deshalb ein zweistufiges GM entwickelt, das mit signalnahen und semantisch höherwertigen Merkmalen die Anordnung von Schnitten und Szenenwechseln im Verbund optimiert. Dadurch hat das Modell in der Dekodierungsphase mehr Informationen zur Erkennung der Szenenwechsel und Schnitte zur Verfügung.

3.1 Videoschichten

Ein Video oder eine TV-Übertragung kann hierarchisch in verschiedene Schichten aufgeteilt werden. Jede Schicht beschreibt eine Einheit des Videos. Abbildung 3.1 zeigt eine solche Aufteilung in Anlehnung an [135]. Diese einzelnen Schichten sollen hier kurz beschrieben werden.

Einzelbilder: Die unterste Schicht eines Videos sind die Einzelbilder (*frames*), dabei handelt es sich um Vollbilder im Progressive-Format. Eine Aufzeichnung im europäischen PAL-TV-Format mit 50 Halbbildern pro Sekunde hat daher 25 Einzelbilder pro Sekunde. Ein Video des amerikanischen NTSC-TV-Formats hat 30 Vollbilder und ein Kinofilm üblicherweise 24 Vollbilder pro Sekunde [117].

Kameraeinstellungen: Eine ununterbrochene, von einer Kamera aus einer Perspektive aufgenommene Sequenz von Bildern formt eine Kameraeinstellung (*shot*) [142]. Je nach Genre kann ein Video mehrere hundert Kameraeinstellungen pro Stunde beinhalten. Der Schnittpunkt zwischen zwei Kameraeinstellungen (*shot boundary*) kann dabei unterschiedliche Formen haben. Bei einem harten Schnitt (*cut*) wird eine Kameraeinstellung innerhalb von einem Bild durch eine zweite, andere Einstellung ersetzt (in Abbildung 3.2 oben gezeigt). Graduelle Schnittpunkte wechseln die Kameraeinstellung über mehrere Bilder. Dabei sind z. B. Überblendungen (*dissolving*), Einblendungen (*fade-in*) oder Ausblendungen (*fade-out*) möglich. In den Zwischenschritten können Mischformen aus beiden Einstellungen auftreten. Abbildung 3.2 (Mitte) zeigt eine Ausblendung bei der die Kameraeinstellung langsam in einen schwarzen Bildschirm gewandelt wird. Kamerazooms und Kameraschwenks (*pan*), wie in Abbildung 3.2 unten gezeigt, stellen hingegen keinen Schnittpunkt dar, da die Kamera ununterbrochen aufnimmt.

Szenen: Mehrere Kameraeinstellungen, die vom Kontext zusammengehören, werden in einer Szene zusammengefaßt [1]. Ursprünglich stammt die Szene aus

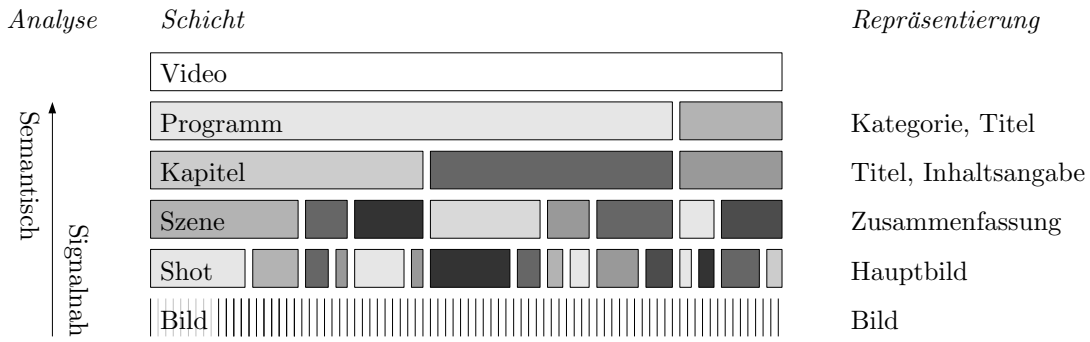

Abbildung 3.1: Die Unterteilung eines Videos in verschiedene Videoschichten sowie beispielhafte Repräsentationsformen für jede Schicht. Ein Kapitel einer DVD kann z. B. durch einen Titel beschrieben werden. Jede Schicht erfordert für die Analyse andere Merkmale. Im Allgemeinen benötigen höhere Schichten dabei mehr Kontextinformation als niedrige Schichten.

dem Theater und umfasst den Handlungsstrang bis zum Umbau der Bühne, also einem Ortswechsel. Heute ist der Begriff der Szene weitergefasst und kann z. B. mehrere Kameraeinstellungen am gleichen Ort, mit den gleichen Personen oder dem gleichen Gesprächsinhalt umfassen. Ein Video kann bis zu hundert verschiedene Szenen pro Stunde enhalten. Szenenwechsel sind weniger streng definiert als Kameraeinstellungswechsel; das Ende einer Szene wird jedoch üblicherweise erreicht, wenn ein vollständiger Wechsel der Handlung erfolgt.

Kapitel: Mehrere Szenen, die einen größeren Handlungsstrang formen, werden zu Kapiteln (*chapter, story unit*) zusammengefasst. Diese werden insbesondere bei der Strukturierung von Filmen auf DVDs verwendet. Üblicherweise hat ein Video nur einige wenige Kapitel pro Stunde.

Programm: Alternativ oder zusätzlich zu den Kapiteln kann das Video auch noch in verschiedene Programme unterteilt werden. Ein Programm ist eine Unterteilung des Videos in verschiedene Kategorien wie Nachrichten, Spielfilm oder Sportübertragung. Dies ist insbesondere für Fernsehübertragungen sinnvoll. DVD Spielfilme hingegen haben üblicherweise nur ein einziges Programm.

Jede Videoschicht kann auf verschiedene Arten repräsentiert werden. In Abbildung 3.1 sind einige Möglichkeiten gezeigt. So kann eine Kameraeinstellung z. B. durch ein Hauptbild (*keyframe*) dargestellt werden, eine Szene z. B. durch eine Zusammenfassung, den Ort der Szene oder die in der Szene enthaltenen Personen. Ein Kapitel wird häufig nur mit einem kurzen Titel beschrieben, möglich wäre jedoch auch eine Zusammenfassung.

Abbildung 3.2: Schnittpunkte zwischen Szenen: Ein harter Schnitt (oben) erfolgt innerhalb von einem Bild. Ein gradueller Wechsel (Ausblenden, Mitte) streckt sich über mehrere Bilder. Kameraschwenks (ohne Abbildung) und Zoomen (unten) sind keine Schnitte, da die Kamera ununterbrochen aufnimmt. Aufgrund ihrer ähnlichen Bild- und Bewegungseigenschaften werden sie bei der automatischen Analyse jedoch häufig mit graduellen Schnitten verwechselt.

Um die verschiedenen Ebenen des Videos zu analysieren, also z. B. Wechsel zu erkennen oder eine Zusammenfassung zu erstellen, werden für jede Schicht andere Merkmale benötigt. Schnitte können aus signalnahen Merkmalen erkannt werden, während für die Szenenwechsel bereits semantische Informationen benötigt werden. Im Allgemeinen benötigen die höheren Schichten für eine automatische Analyse immer mehr semantische und Kontextinformation.

3.2 Automatische Schnitt- und Szenenerkennung

Zahlreiche wissenschaftliche Arbeiten beschäftigen sich mit den verschiedenen Schichten in Videos und deren automatischer Analyse. Insbesondere die Schnitterkennung wurde mit verschiedenen Methoden untersucht. Dabei basieren die meisten Arbeiten auf komplexen Merkmalen, die mittels relativ einfacher – zum Teil adaptiver – schwellwertbasierter Verfahren ausgewertet werden. Eine umfassende Einführung in diese Verfahren mit einem Vergleich der Erkennungsleistungen gibt [33]. Darauf aufbauende Methoden werden in [116] vorgestellt und evaluiert. Neuere Verfahren verwenden auch Methoden der Mustererkennung zur automatischen Schnitterkennung: so werden z. B. in [71] Support Vector Machines verwendet. Eine standartisierte Evaluierung der verschiedenen Verfahren zur automatischen Erkennung von Schnitten bietet z. B. TRECVID [141, 142].

Ein Grund für die zahlreichen Arbeiten auf dem Gebiet der automatischen Schnitterkennung ist, dass sie verhältnismäßig gut maschinell verarbeitet werden können: Schnitte sind klar definiert und können theoretisch aus dem reinen Bildstrom – ohne Kontext- und Audioinformation – des Videos erkannt werden. Allerdings sind sie nicht optimal für Datenbankabfragen geeignet: Während der Sprecher einer Nachrichtensendung das Thema ohne eine Schnitt wechseln kann, besteht ein Spielfilm aus mehreren hundert verschiedenen Kameraeinstellungen. Es ist daher nur sehr selten zu erwarten, dass ein Nutzer direkt nach einer bestimmten Kameraeinstellung in einem Film sucht. Auf der anderen Seite können Schnitte jedoch sehr gut als zusätzliches Merkmal zur Analyse von höheren Ebenen verwendet werden.

Dagegen sind Szenen und Kapitel sehr gut für intuitive Abfragen geeignet, da sie größere und thematisch zusammenhängende Einheiten des Videos gruppieren. Daher werden z. B. Spielfilme auf DVD häufig anhand von Szenen oder Kapiteln indexiert. Das automatische Auffinden von Szenen- und Kapitelwechseln aus dem Video ist deshalb ein zunehmendes Forschungsgebiet: So zeigt z. B. [41] die automatische Segmentierung von Nachrichtensendungen in einzelne Szenen. Ereignisse in Sportsendungen werden in [22] untersucht und [137] stellt eine Methode vor, mit der der Inhalt von Spielfilmen automatisch strukturiert wird.

Wie in Abbildung 3.1 und im vorherigen Abschnitt gezeigt, benötigen die unterschiedlichen Schichten eines Videos unterschiedliche Merkmale zur automatischen Analyse. Schnitt- und Szenenwechsel wurden daher bisher fast ausschließlich separat untersucht. Vorhergehende Arbeiten verarbeiten die beiden Ebenen sequentiell und modellieren üblicherweise die Interaktion zwischen den Ebenen nicht. Häufig verfügt die Ebene zur Analyse der Szenen nicht über die Information für mögliche Schnittpositionen. Auch die Schnitterkennung wird fast ausschließlich ohne Szeneninformation durchgeführt. Dabei hat eine hohe Konfidenz für einen nicht vorhandenen Schnitt eine sehr hohe Auswirkung auf die Wahrscheinlichkeit eines Szenenwechsels. Auch umgekehrt induziert in einem Spielfilm eine hohe Konfidenz für einen Szenenwechsel mit fast sicherer Wahrscheinlichkeit auch einen Schnitt. Es wäre daher sinnvoll, beide Schichten gemeinsam zu analysieren. Dazu wurde in [1] ein mehrstufiges Modell vorgestellt, das vielversprechende Ergebnisse geliefert hat.

Im Rahmen dieser Arbeit wird gezeigt, wie signalnahe und semantische Merkmale in einem zweistufigen GM miteinander verbunden werden können. In einer ersten Stufe werden die Schnitte analysiert. In der zweiten Stufe wird nach Szenenwechseln gesucht. Das Modell kann daher Schnitte und Szenen kombiniert analysieren. Durch die Verbindung beider Schichten lernt das Modell automatisch den Einfluss der beiden Ebenen aufeinander und optimiert ihre Anordnung in der Dekodierungsphase. In einem weiteren Experiment wurde die Szenenschicht durch eine Kapitelschicht ersetzt, ohne die prinzipielle Funktionsweise des GM zu ändern. So kann das Modell auch auf die kombinierte Schnitt- und Kapitelerkennung angewendet werden. Prinzipiell kann das GM auch auf weitere Schichten des Videos erweitert werden.

3.3 Merkmale

3.3.1 Signalnahe Merkmale

Zur automatischen Erkennung von Schnitten werden signalnahe Merkmale extrahiert. Die im Folgenden vorgestellten Merkmale sind ähnlich zu den in [33, 116] vorgeschlagenen und werden bereits in verschiedenen Systemen zur automatischen Erkennung von Schnitten eingesetzt.

Zuerst werden die RGB-Bilder des Videos in Graustufenbilder gewandelt. Dies ermöglicht eine einfachere und schnellere Verarbeitung. Dazu wird die Intensität $I_t(x, y)$ für alle Pixel (x, y) aus den RGB-Bildern berechnet:

$$I_t(x, y) = 0.3\ R_t(x, y) + 0.59\ G_t(x, y) + 0.11\ B_t(x, y), \tag{3.1}$$

wobei die drei Grundfarben Rot, Grün und Blau unterschiedlich stark gewichtet werden, so dass natürlich wirkende Graustufenbilder entstehen [155]. Weiterhin gelte die Annahme, dass für den Zeitpunkt $t = 0$ für alle Pixel gilt, dass die Intensitäten $I_0(x, y) = 0, \forall x, y$ sind. Somit kann auch für das erste Bild $t = 1$ des Videos ein Differenzbild berechnet werden. Zuerst wird für alle Zeitschritte t die **durchschnittliche Intensitätsdifferenz** zwischen zwei aufeinanderfolgenden Bildern bestimmt

$$x_t^{\text{I-diff}} = \frac{\sum_{x=1}^{X} \sum_{y=1}^{Y} |I_t(x, y) - I_{t-1}(x, y)|}{XY}. \tag{3.2}$$

Zusätzlich wird aus jedem Intensitätsbild $I_t(x, y)$ ein Grauwerthistogramm $H_t(i)$ mit 256 Stufen erstellt. Dazu werden die Intensitätswerte jedes Bildes nach aufsteigender Helligkeit sortiert und dann die Anzahl der Pixel, die in die jeweilige Helligkeitsstufe fallen, bestimmt. Aus diesen Histogrammen wird die **durchschnittliche Grauwerthistogrammdifferenz** zwischen zwei aufeinanderfolgenden Bildern bestimmt:

$$x_t^{\text{H-diff}} = \frac{\sum_{i=0}^{256} |H_t(i) - H_{t-1}(i)|}{256} \tag{3.3}$$

Weiterhin werden die ersten 15 Koeffizienten $0 \leq u, v \leq 4$, mit $u + v \leq 4$ der diskreten Kosinus Transformation (DCT) für jedes Bild berechnet:

$$C_t(u, v) = \sum_{x=1}^{X} \sum_{y=1}^{Y} I_t(x, y) \cos\left(\frac{(2x + 1)\ u\pi}{2X}\right) \cos\left(\frac{(2y + 1)\ v\pi}{2Y}\right) \tag{3.4}$$

und daraus die **durchschnittliche Frequenzdifferenz** zwischen zwei aufeinanderfolgenden Bildern bestimmt:

$$x_t^{\text{C-diff}} = \frac{\sum_{u=0}^{4} \sum_{v=0}^{4-u} |C_t(u, v) - C_{t-1}(u, v)|}{15}. \tag{3.5}$$

Diese drei Merkmale formen zusammen den zeitabhängigen **signalnahen Merkmalvektor**

$$\vec{x}_t = \left[x_t^{\text{I-diff}}, x_t^{\text{H-diff}}, x_t^{\text{C-diff}}\right], \tag{3.6}$$

der zu jedem Bild t des Videos berechnet wird.

Als weitere signalnahe Merkmale wurden auch Farbdifferenzbilder, blockweise DCT Koeffizienten, die globale Bewegung in Bildern, der optischer Fluss und auch akustische Merkmale wie eine gefensterte Signalenergie oder Mel-Frequenz-Cepstrum Koeffizienten (*Mel frequency cepstral coefficients, MFCC*) [60] implementiert und evaluiert. Details zu diesen Merkmalen sowie ein umfangreicher Vergleich der Erkennungsleistung mit einzelnen Merkmalen – sowohl mit schwellwertbasierten Methoden als auch mit BN – finden sich in [167]. Da diese erweiterten Merkmale jedoch auf die Erkennungsleistung des hier vorgestellten GM keinen signifikanten Einfluss hatten, werden sie in dieser Arbeit nicht weiter verwendet.

3.3.2 Semantisch höherwertige Merkmale

Für die Ebene der Szenenerkennung werden semantisch höherwertige Merkmale semiautomatisch erzeugt. Zuerst werden dazu die gesprochenen Texte aus den DVD Untertiteln extrahiert und gespeichert. Auf diese Weise kann eine fehleranfällige automatische Spracherkennung vermieden werden, die das spätere Erkennungsergebnis unter Umständen verfälscht. Prinzipiell wäre es jedoch möglich, statt der Untertitel eine automatische Spracherkennung einzusetzen, wodurch das System auch für im Fernsehen ausgestrahlte Sendungen eingesetzt werden könnte – ein Rückgang der Erkennungsrate wäre dann jedoch zu erwarten.

Die gesprochenen Worte werden dann der Transkription frei erhältlicher Drehbücher aus dem Internet zugeordnet, wobei zu jedem gesprochenen Satz im Video die korrespondierende Niederschrift im Drehbuch gesucht wird. Dabei wird eine lineare Suche durchgeführt (von der letzten bekannten Zeitinstanz im Drehbuch wird nicht mehr rückwärts gesucht) und weiche Entscheidungen sind zugelassen (Sätze müssen nicht genau wörtlich übereinstimmen). Details zu dem implementierten Suchalgorithmus finden sich in [168]. Auf diese Weise kann zu jedem Frame des Videos aus den Drehbüchern automatisch

- der augenblickliche Sprecher,
- der gegenwärtige Ort,
- sowie alle in der Szene gezeigten Personen

extrahiert werden. Diese Information wird dann für jedes Frame des Videos in einen digital kodierten semantischen Merkmalvektor geschrieben. Die extrahierte Sprache hingegen wird lediglich für das Mapping verwendet, wegen der hohen Komplexität

gesprochener Sprache jedoch nicht mehr als Merkmal für die Szenenerkennung genutzt. Konfidenzen zu den jeweilig erkannten Personen und Orten werden ebenfalls nicht verwendet.

Da die gesprochene Sprache in den Videos nicht immer perfekt mit dem Drehbuch übereinstimmt (es kommt häufig vor, dass Sätze während des Drehs hinzugefügt, weggelassen oder verändert werden), ist die tatsächliche Zuordnung fehlerbehaftet. Im Durchschnitt sind bei dem hier verwendeten Verfahren 75% der gewonnenen semantischen Merkmale korrekt im Vergleich zu einer händischen Annotation der Daten. Der vorgestellte Prozess ist von der Erkennungsleistung daher vergleichbar zu vollautomatischen Methoden wie der automatischen Sprach- [59] oder Personenerkennung [2, 38, 171]. Der Fokus dieser Arbeit liegt auf der Erkennung der Szenen und Schnitte, nicht aber auf der Merkmalgewinnung. Daher wird nur das beschriebene semiautomatische Verfahren angewendet, jedoch keine vollautomatischen Methoden zur Merkmalgewinnung evaluiert. Das hier verwendete Verfahren zur Gewinnung von semantischen Merkmalen ist deutlich einfacher anzuwenden als die entsprechenden vollautomatischen Methoden und kann daher auf einen großen Datensatz mit relativ hoher Konfidenz angewendet werden. Der Hauptnachteil des Verfahres ist die Abhängigkeit von verfügbaren Drehbüchern zu dem Videomaterial. Das Verfahren kann daher z. B. nicht auf Live-Sendungen angewendet werden.

Allerdings kann das Modul zur Gewinnung der semantischen Merkmale auch problemlos durch automatische Methoden ausgewechselt werden, ohne die prinzipielle Funktionsweise des GM zur Schnitt- und Szenenerkennung zu beeinflussen.

Eine detaillierte Beschreibung der Untertitelextraktion, des Algorithmus zur Zuordnung von gesprochenem Text zu den Drehbüchern, der Gewinnung und digitalen Codierung der semantischen Merkmale sowie einer Evaluierung der Robustheit der einzelnen Merkmale für verschiedene Videodaten findet sich in [168].

3.4 Zweistufiges Graphisches Modell

Im Rahmen dieser Arbeit wird ein GM entwickelt, das mit den im vorherigen Abschnitt beschriebenen signalnahen und semantisch höherwertigen Merkmalen gleichzeitig für die Schnitt- und Szenenwechselerkennung in Videos verwendet werden kann. Abbildung 3.3 zeigt das dazu entwickelte zweistufige Modell. Ein Zeitschlitz besteht aus vier Variablen: den diskret modellierten, beobachteten semantisch höherwertigen Merkmalen $\vec{s}_t$, den kontinuierlich modellierten, beobachteten signalnahen Merkmalen $\vec{x}_t$ und den beiden verborgenen, diskreten Zuständen für die Schnitterkennung h_t und Szenenerkennung z_t. Dabei hat die Schnitterkennung h_t einen unmittelbaren Einfluss auf die Szenenerkennung z_t im selben Zeitschlitz. Dies wird modelliert durch die direkte Eltern-Kind-Beziehung von Schnitt- und Szenenzustand. Der eigentliche zeitliche Verlauf zwischen den Einzelbildern wird über die beiden parallelen Markov-Ketten unabhängig für die Schnitt- und Szenenschicht modelliert.

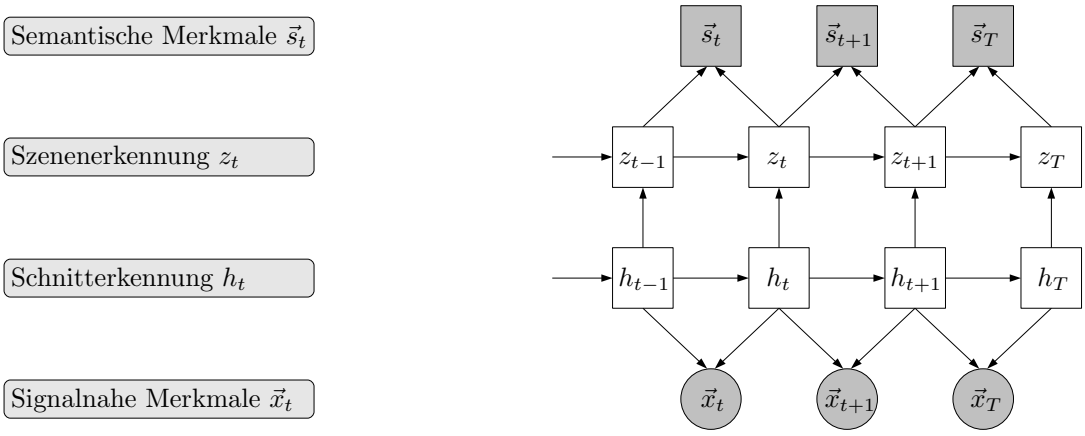

Abbildung 3.3: Zweistufiges GM zur kombinierten Erkennung von Schnitten und Szenenwechseln in Videos. Für die Schnitte werden die signalnahen und für die Szenen die semantisch höherwertigen Merkmale verwendet. Beide Ebenen werden durch zwei parallele Markov-Ketten modelliert. Zusätzlich hat die Erkennung von Schnitten einen direkten Einfluss auf die Erkennung von Szenenwechseln – modelliert durch die Eltern-Kind-Beziehung der beiden Schichten.

Eine weitere Besonderheit des Modells ist, dass die verborgenen Zustände des Modells nicht nur mit den Merkmalen des gegenwärtigen, sondern auch mit denen des nachfolgenden Bildes verbunden sind. Dabei ist nur der Epilog eine Ausnahme, der nur mit den Merkmalen des letzten Bildes verbunden ist. Bei der Wandlung vom interlaced- zum progressive-Format kann es zu Artefaktbildungen kommen: Dabei besteht ein Frame aus den Halbbildern zweier verschiedener Kameraeinstellungen, d. h. das Frame setzt sich aus einem Halbbild vor und nach dem Schnittpunkt zusammen. Durch die Verwendung differenzieller Merkmale wird insbesondere die Veränderung zweier aufeinanderfolgender Bilder detektiert. Durch die Möglichkeit der Artefakte kann es dann aber zu einer unerwünschten Doppeldetektion eines Schnitts kommen, da das Artefaktframe einen großen Unterschied sowohl zu dem Frame vor als auch nach dem Schnitt hat. Durch die hier verwendete Merkmalmodellierung kann diese Doppeldetektion bereits auf Modellebene ausgeschlossen werden. Weiterhin kann das Modell durch diese doppelte Merkmalverbindung – je nach gewünschtem Verhalten – so trainiert werden, dass ein Schnitt wahlweise vor, nach oder sowohl vor und nach dem Wechsel der Kameraeinstellung angezeigt wird.

3.4.1 Schnitterkennungsschicht

Zuerst soll die Schicht zur Erkennung der Schnitte beschrieben werden. Sie ist modelliert mit einer Markov-Kette erster Ordnung. Jeder Knoten h_t repräsentiert einen diskreten, verborgenen Zustand und ist verbunden mit den signalnahen Merkmalen $\vec{x}_t$ und $\vec{x}_{t+1}$ des augenblicklichen und des nächsten Bildes. Lediglich im Prolog ist das

Merkmal $\vec{x}_1$ nur durch den ersten Zustand h_1 erzeugt und im Epilog ist der letzte Zustand h_T nur mit den signalnahen Merkmalen des letzten Vollbildes $\vec{x}_T$ verbunden. Durch die Markov-Kette erster Ordnung und den damit verbundenen Merkmalen ist die Struktur der Schnitterkennung ähnlich zu einem HMM. Der Hauptunterschied besteht in der doppelten Merkmalverbindung in jedem Zustand. Bis zum Zeitpunkt τ ergibt sich dann für die Verbundwahrscheinlichkeit der Schnitterkennungsebene aus (DF):

$$p(h_1, \ldots, h_\tau, \vec{x}_1, \ldots, \vec{x}_\tau, \vec{x}_{\tau+1}) = p(h_1)\; p(\vec{x}_1 \,|\, h_1) \prod_{t=2}^{\tau} p(h_t \,|\, h_{t-1})\; p(\vec{x}_t \,|\, h_t, h_{t-1}), \tag{3.7}$$

wobei $p(h_1)$ die Wahrscheinlichkeit eines Schnittes im ersten Frame beschreibt. Solange das GM nicht in ein weiteres System eingebettet ist, startet eine Sequenz immer mit einem Schnitt. Der Zustandsübergang – also implizit die Wahrscheinlichkeit von aufeinanderfolgenden Schnitten – wird mit der Zustandsübergangswahrscheinlichkeit $p(h_t \,|\, h_{t-1})$ modelliert. Die signalnahen Merkmale werden abhängig von den Zuständen h_t und h_{t-1} mit der Wahrscheinlichkeit $p(\vec{x}_t \,|\, h_t, h_{t-1})$ modelliert. Wie in Abbildung 3.3 gezeigt, werden diese Auftrittswahrscheinlichkeiten für die signalnahen Merkmale dabei durch Gaußkurven modelliert.

Die Parameter der Knoten können dabei – wie in Abschnitt 2.9 erläutert – mit dem EM-Algorithmus trainiert werden. Prinzipiell könnte diese Schnitterkennungsschicht dann bereits unabhängig vom übrigen Modell genutzt werden; im Rahmen dieser Arbeit wird sie jedoch nur in Kombination mit der Szenenerkennung verwendet.

3.4.2 Szenenerkennungsschicht

Die Struktur der Szenenerkennungsschicht ist ähnlich zur Schnitterkennung: Der diskrete Zustand z_t ist verbunden mit den semantisch höherwertigen Merkmalen des Bildes t und des nächsten Bildes $t+1$. Jedoch ist der Zustand z_t der Markov-Kette nicht nur abhängig vom vorherigen Zustand z_{t-1}, sondern auch vom aktuellen Zustand der Schnitterkennung h_t. Dadurch haben die Resultate der Schnitterkennung einen Einfluss auf die Erkennung von Szenenwechseln. Bis zum Zeitpunkt τ ergibt sich dann für die bedingte Verbundwahrscheinlichkeit aus (DF):

$$\begin{aligned} &p(z_1, \ldots, z_\tau, \vec{s}_1, \ldots, \vec{s}_\tau, \vec{s}_{\tau+1} \,|\, h_1, \ldots, h_\tau) \\ &\qquad = p(z_1 \,|\, h_1)\; p(\vec{s}_1 \,|\, z_1) \prod_{t=2}^{\tau} p(z_t \,|\, z_{t-1}, h_t)\; p(\vec{s}_t \,|\, z_t, z_{t-1}), \end{aligned} \tag{3.8}$$

wobei $p(z_1 \,|\, h_1)$ die Wahrscheinlichkeit einer neuen Szene im ersten Frame beschreibt. Dabei startet eine Sequenz wiederum immer mit einer neuen Szene, solange das GM

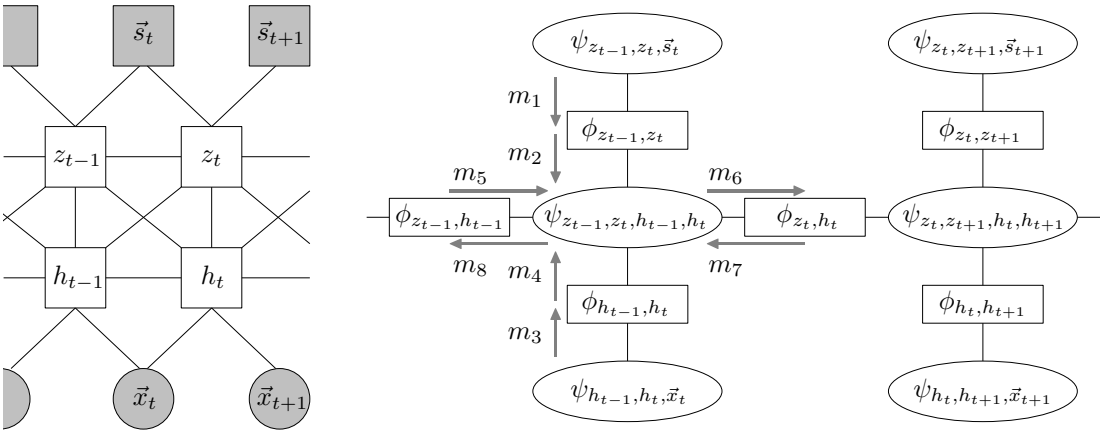

Abbildung 3.4: Mögliche Triangulierung des Modells (links) und sich ergebender Verbundbaum (rechts) mit den notwendigen Nachrichten für eine Vorwärts- und Rückwärtsrekursion.

nicht in ein weiteres Modell eingebunden wird. Die Zustandsübergangswahrscheinlichkeit $p(z_t \mid z_{t-1}, h_t)$ ist abhängig vom aktuellen Zustand der Schnitterkennung und vom vorherigen Szenenzustand. Die semantisch höherwertigen Merkmale werden abhängig von den Zuständen z_t und z_{t-1} mit der Wahrscheinlichkeit $p(\vec{s}_t \mid z_t, z_{t-1})$ modelliert. Aufgrund der diskreten Kodierung von $\vec{s}_t$ werden sie im Gegensatz zu den signalnahen Merkmalen jedoch durch eine diskrete WDF repräsentiert.

Im Rahmen der Arbeit wurden auch zu Abbildung 3.3 modifizierte Strukturen evaluiert, so z. B. das Verwenden von signalnahen Merkmalen für die Szenenerkennung (graphisch entspricht das einer zusätzlichen gerichteten Kante von z_t zu $\vec{x}_t$). Dies führt jedoch zu keiner verbesserten Erkennung, da die Information bereits implizit durch die Schnittschicht modelliert ist.

3.4.3 Nachrichtenpropagierung im Modell

Im Folgenden soll die Nachrichtenpropagierung, die sich aus dem vorgestellten GM ergibt, hergeleitet werden. Abbildung 3.4 zeigt dazu eine mögliche (intuitive, aber nicht unbedingt optimale) Triangulierung des Modells und den sich daraus ergebenden Verbundbaum. Aus dem Verbundbaum ergibt sich für die signalnahen Merkmale die Cliquepotentiale

$$\psi_{h_{t-1},h_t,\vec{x}_t}(h_{t-1}, h_t, \vec{x}_t), \tag{3.9}$$

für die semantisch höherwertigen Merkmale die Cliquenpotentiale

$$\psi_{z_{t-1},z_t,\vec{s}_t}(z_{t-1}, z_t, \vec{s}_t) \tag{3.10}$$

und für die Zustände die Cliquenpotentiale

$$\psi_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1}, h_{t-1}, z_t, h_t). \tag{3.11}$$

Zeitlich aufeinanderfolgende Zustände werden getrennt durch die Separatoren mit den Potentialen $\phi_{z_t,h_t}(z_t, h_t)$. Die signalnahen Merkmale werden von den Zuständen durch die Separatoren mit den Potentialen $\phi_{h_{t-1},h_t}(h_{t-1}, h_t)$ und die semantisch höherwertigen Merkmale durch die Separatorpotentiale $\phi_{z_{t-1},z_t}(z_{t-1}, z_t)$ getrennt.

Initialisierung

Zur Initialisierung der Zustandscliquen werden die Übergangswahrscheinlichkeiten sowohl für die Szenen als auch für die Schnitte in das Cliquenpotential multipliziert. Es ergibt sich dann

$$\psi_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1}, h_{t-1}, z_t, h_t) = p(z_t \mid z_{t-1}, h_t)\; p(h_t \mid h_{t-1}), \tag{3.12}$$

für alle Cliquen mit $t > 1$ bzw. für den Prolog die vereinfachte Einsprungswahrscheinlichkeit

$$\psi_{z_1,h_1}(z_1, h_1) = p(z_1 \mid h_1)\; p(h_1). \tag{3.13}$$

Die Cliquen mit den signalnahen Merkmalen werden mit der Emissionswahrscheinlichkeit für die Merkmale bei gegebenen Zuständen initialisiert:

$$\psi_{h_{t-1},h_t,\vec{x}_t}(h_{t-1}, h_t, \vec{x}_t) = p(\vec{x}_t \mid h_t, h_{t-1}). \tag{3.14}$$

Analog ergibt sich für die Cliquen mit den höherwertigen Merkmalen nach der Initialisierung:

$$\psi_{z_{t-1},z_t,\vec{s}_t}(z_{t-1}, z_t, \vec{s}_t) = p(\vec{s}_t \mid z_t, z_{t-1}). \tag{3.15}$$

Alle Separatorpotentiale werden mit dem Wert eins belegt, so dass

$$\begin{aligned} &\phi_{z_t,h_t}(z_t, h_t) = 1, \\ &\phi_{h_{t-1},h_t}(h_{t-1}, h_t) = 1, \\ &\phi_{z_{t-1},z_t}(z_{t-1}, z_t) = 1. \end{aligned} \tag{3.16}$$

Nachrichten von den Merkmalscliquen zu den Zustandscliquen

Die Nachrichten m_1 bis m_4 von den Merkmals- zu den Zustandscliquen sind sowohl für den Vorwärts- als auch für den Rückwärtspfad identisch. Zuerst wird mit der Nachricht m_1 der Separator nach Gleichung (2.48)

$$\begin{aligned} \phi^*_{z_{t-1},z_t}(z_{t-1}, z_t) &= \sum_{\vec{s}_t} \psi_{z_{t-1},z_t,\vec{s}_t}(z_{t-1}, z_t, \vec{s}_t) \\ &= \sum_{\vec{s}_t} p(\vec{s}_t \mid z_t, z_{t-1}) \end{aligned} \tag{3.17}$$

aktualisiert. Die Beobachtung $\vec{s}_t$ ist aus der Merkmalsequenz bekannt und kann daher – wie in Abschnitt 2.10.2 gezeigt – als Dirac eingebunden werden. Die Summation über alle möglichen Beobachtungen entfällt dadurch und es ergibt sich für den aktualisierten Separator:

$$\begin{aligned}\phi^*_{z_{t-1},z_t}(z_{t-1},z_t) &= \sum_{\vec{s}_t} p(\vec{s}_t \mid z_t, z_{t-1})\ \delta[\vec{s}_t] \\ &= p(\vec{s}_t \mid z_t, z_{t-1}).\end{aligned} \tag{3.18}$$

Gemäß Gleichung (2.49) wird daraus die Zustandsclique mit der Nachricht m_2 aktualisiert:

$$\begin{aligned}\psi^*_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t) &= \frac{\phi^*_{z_{t-1},z_t}(z_{t-1},z_t)}{\phi_{z_{t-1},z_t}(z_{t-1},z_t)}\ \psi_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t) \\ &= \phi^*_{z_{t-1},z_t}(z_{t-1},z_t)\ \psi_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t) \\ &= p(\vec{s}_t \mid z_t, z_{t-1})\ p(z_t \mid z_{t-1}, h_t)\ p(h_t \mid h_{t-1}).\end{aligned} \tag{3.19}$$

Die Nachrichten m_3 und m_4 von den signalnahen Merkmalen zu der Zustandsclique sind analog zu m_1 und m_2. Es ergibt sich daher nach der Absorption der Nachricht m_4

$$\begin{aligned}&\psi^{**}_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t) \\ &\qquad = p(\vec{x}_t \mid h_t, h_{t-1})\ p(\vec{s}_t \mid z_t, z_{t-1})\ p(z_t \mid z_{t-1}, h_t)\ p(h_t \mid h_{t-1})\end{aligned} \tag{3.20}$$

für die aktualisierte Zustandsclique. Das Potential ψ^{**} enthält nun die Information über beide Merkmale und kann daher sowohl im Vorwärts- als auch im Rückwärtspfad verwendet werden.

Vorwärtspfad

Für den Vorwärtspfad wird zuerst mit der Nachricht m_5 die nachfolgende Clique durch den Separator aktualisiert. Es ergibt sich aus der Absorptionsgleichung (2.49):

$$\begin{aligned}&{\psi^{***}}_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t) \\ &\quad = \frac{\phi^*_{z_{t-1},h_{t-1}}(z_{t-1},h_{t-1})}{\phi_{z_{t-1},h_{t-1}}(z_{t-1},h_{t-1})}\ \psi^{**}_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t) \\ &\quad = \phi^*_{z_{t-1},h_{t-1}}(z_{t-1},h_{t-1})\ \psi^{**}_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t) \\ &\quad = \phi^*_{z_{t-1},h_{t-1}}(z_{t-1},h_{t-1})\ p(\vec{x}_t \mid h_t, h_{t-1})\ p(\vec{s}_t \mid z_t, z_{t-1})\ p(z_t \mid z_{t-1}, h_t)\ p(h_t \mid h_{t-1}).\end{aligned} \tag{3.21}$$

Die so aktualisierte Clique ψ^{***} enthält nun alle benötigten Informationen von den vorherigen Zustandscliquen und über die Merkmale zum augenblicklichen Zeitpunkt.

Mit der Nachricht m_6 zum Separator ergibt sich dann aus Gleichung (2.48)

$$\begin{aligned}
&\phi^*_{z_t,h_t}(z_t,h_t)\\
&= \sum_{h_{t-1}}\sum_{z_{t-1}} \psi^{***}_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t)\\
&= \sum_{h_{t-1}}\sum_{z_{t-1}} \phi^*_{z_{t-1},h_{t-1}}(z_{t-1},h_{t-1})\; p(\vec{x}_t \mid h_t,h_{t-1})\; p(\vec{s}_t \mid z_t,z_{t-1})\; p(z_t \mid z_{t-1},h_t)\; p(h_t \mid h_{t-1})
\end{aligned} \tag{3.22}$$

die Vorwärtsrekursion des Modells.

Rückwärtspfad

Für den Rückwärtspfad bleiben die Nachrichten m_1 bis m_4 von den Merkmalen zu den Zustandsclique identisch. Dann wird mit der Nachricht m_7 die Clique durch den zeitlich nächsten Separator aktualisiert. Dafür ergibt sich aus der Absorptionsgleichung (2.49):

$$\begin{aligned}
&\psi^{***}_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t)\\
&\qquad= \frac{\phi^*_{z_t,h_t}(z_t,h_t)}{\phi_{z_t,h_t}(z_t,h_t)}\; \psi^{**}_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t)\\
&\qquad= \phi^*_{z_t,h_t}(z_t,h_t)\; \psi^{**}_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t)\\
&\qquad= \phi^*_{z_t,h_t}(z_t,h_t)\; p(\vec{x}_t \mid h_t,h_{t-1})\; p(\vec{s}_t \mid z_t,z_{t-1})\; p(z_t \mid z_{t-1},h_t)\; p(h_t \mid h_{t-1}).
\end{aligned} \tag{3.23}$$

Die so aktualisierte Clique ψ^{***} enthält wiederum alle benötigten Informationen. Damit kann mit der Nachricht m_8 der zeitlich vorherige Separator aktualisiert werden. Es ergibt sich dann aus Gleichung (2.48):

$$\begin{aligned}
&\phi^*{}_{z_{t-1},h_{t-1}}(z_{t-1},h_{t-1})\\
&\quad= \sum_{h_t}\sum_{z_t} \psi^{***}_{z_{t-1},h_{t-1},z_t,h_t}(z_{t-1},h_{t-1},z_t,h_t)\\
&\quad= \sum_{h_t}\sum_{z_t} \phi^*_{z_t,h_t}(z_t,h_t)\; p(\vec{x}_t \mid h_t,h_{t-1})\; p(\vec{s}_t \mid z_t,z_{t-1})\; p(z_t \mid z_{t-1},h_t)\; p(h_t \mid h_{t-1})
\end{aligned} \tag{3.24}$$

die Rückwärtsrekursion des Modells. Diese wird jedoch nur für das Training des Modells, nicht aber in der Dekodierungsphase benötigt.

3.4.4 Dekodierung

Mit der Szenen- und der Schnittschicht wird die Gesamtwahrscheinlichkeit für das Modell zum Zeitpunkt τ zu

$$\begin{aligned}
&p(h_1,\ldots,h_\tau,z_1,\ldots,z_\tau,\vec{x}_1,\ldots,\vec{x}_\tau,\vec{x}_{\tau+1},\vec{s}_1,\ldots,\vec{s}_\tau,\vec{s}_{\tau+1})\\
&\qquad= p(h_1,\ldots,h_\tau,\vec{x}_1,\ldots,\vec{x}_\tau,\vec{x}_{\tau+1})\; p(z_1,\ldots,z_\tau,\vec{s}_1,\ldots,\vec{s}_\tau,\vec{s}_{\tau+1} \mid h_1,\ldots,h_\tau),
\end{aligned} \tag{3.25}$$

die sich aus dem Produkt der Wahrscheinlichkeiten aus Gleichung (3.7) und (3.8) ergibt. Diese Wahrscheinlichkeit kann mit dem JT-Algorithmus und der im vorherigen Abschnitt hergeleiteten Vorwärtsrekursion effizient für jeden Zeitpunkt τ berechnet werden.

Die Dekodierung selbst wird dann bildweise durchgeführt, indem über die signalnahen und semantisch höherwertigen Merkmale marginalisiert wird und dann in den beiden verborgenen diskreten Zuständen h_τ und z_τ maximiert wird. Das Resultat für jedes Bild kann dann einen von vier verschiedenen Werten annehmen:

- ein Schnitt und ein Szenenwechsel,
- ein Schnitt ohne einen Szenenwechsel,
- ein Szenenwechsel ohne einen Schnitt oder
- weder Schnitt noch Szenenwechsel.

Der Hauptvorteil dieser Dekodierungsstragie ist, dass sie online, mit nur einem Bild Verzögerung, durchgeführt werden kann. Trotzdem stellt die Modellstruktur sicher, dass Schnitte und Szenenwechsel im Verbund optimiert werden.

3.4.5 Training

Prinzipiell sind für das Modell ein unüberwachtes und ein überwachtes Training möglich. Beim überwachten Training wird zuerst die Schnitterkennungsschicht mit den signalnahen Merkmalen trainiert. Dabei sind die wahren Schnittpunkte des Datenmaterials in der Trainingsphase bekannt. Dann wird darauf aufbauend die Szenenschicht mit den semantisch höherwertigen Merkmalen trainiert. Dabei sind wiederum die wahren Szenenwechsel des Datenmaterials bekannt. Außerdem wird während des Trainings der Szenenschicht auch die bereits gelernte Schnittschicht verwendet. Die Szenenschicht verfügt also während des Trainings bereits über die Schnittinformation und kann daher die gemeinsame Auftrittswahrscheinlichkeit lernen. Das Training selbst wird in beiden Schichten mit dem EM-Algorithmus (wie in Abschnitt 2.9 beschrieben) durchgeführt. Die dafür benötigte vollständige Statistik wird mit der in Abschnitt 3.4.3 hergeleiteten Vorwärts- und Rückwärtsrekursion errechnet.

Bei der zweiten Trainingsstrategie werden beide Schichten gleichzeitig, aber ohne Kenntniss der wahren Schnittpunkte in den Daten, trainiert. Dies führt zu einer unüberwachten Merkmaldekomposition in der Schnittschicht. Diese hat dann nur noch einen unterstützenden Charakter für die Szenenschicht und wird so optimiert, dass die Datenwahrscheinlichkeit für die Szenen optimal wird. Damit hat die Schnittschicht dann aber keinen tatsächlichen Bezug zu den Schnitten im Datenmaterial und kann daher in der Dekodierungsphase auch nicht zur Schnitterkennung eingesetzt werden. Ein solches Training ist also nur sinnvoll, wenn keine Schnitterkennung

benötigt wird und die signalnahen Merkmale lediglich als Unterstützung zur Szenenerkennung genutzt werden sollen.

Um das Modell tatsächlich sowohl für die Schnitt- als auch für die Szenenerkennung zu nutzen, wird im Rahmen dieser Arbeit daher nur das überwachte Training angewendet.

3.5 Experimente

3.5.1 Daten

Um das neue GM zu evaluieren, wird es für beide Schichten mit einer zur Detektion von Schnitten und Szenen häufig verwendeten Schwellwertmethode wie z. B. in [33] beschrieben, verglichen. Als Daten werden sechs Episoden von unterschiedlichen auf DVD verfügbaren Serien aus verschiedenen Genres verwendet. Das Videomaterial liegt im europäischen PAL TV-Format interlaced mit 50 Halbbildern pro Sekunde vor und wird in das progressive-Format mit dann 25 Vollbildern pro Sekunde umgewandelt. Anschließend werden die in Abschnitt 3.3 beschriebenen signalnahen und semantisch höherwertigen Merkmale extrahiert, die sowohl für die Schwellwertmethode als auch für das GM verwendet werden. Zusammen haben die vorliegenden Daten eine Länge von ca. vier Stunden (ca. 360 000 Vollbilder). Darin wurden insgesamt knapp 2 000 harte Schnitte und 90 Szenenwechsel manuell annotiert. Graduelle Schnitte werden im Rahmen dieser Arbeit nicht verwendet, da sie andere, komplexere Merkmale benötigen.

Auf diesen Daten werden vier verschiedene Experimente durchgeführt: Die Erkennung von Schnitten mit zwei verschiedenen Trainingsdatensätzen, die Erkennung von Szenenwechseln und außerdem die Erkennung von Kapitelgrenzen für die im GM die Szenenschicht durch eine Kapitelschicht ausgewechselt wird.

3.5.2 Bewertungsmaße

Zum Vergleich der Methoden werden drei im Bereich des information retrieval verbreitete Gütemaße verwendet [133]: Recall ist der Anteil der richtig aufgefundenen Ereignisse zu allen in den Daten vorhandenen Ereignissen:

$$r = \text{recall} = \frac{\text{richtig erkannte Ereignisse}}{\text{richtig erkannte Ereignisse} + \text{nicht erkannte Ereignisse}}. \tag{3.26}$$

Precision ist der Anteil an erkannten Ereignissen, die tatsächlich in den Daten vorhanden waren, zu allen erkannten Ereignissen:

$$p = \text{precision} = \frac{\text{richtig erkannte Ereignisse}}{\text{richtig erkannte Ereignisse} + \text{falsch erkannte Ereignisse}}. \tag{3.27}$$

Sowohl recall als auch precision haben für sich genommen nur eine begrenzte Aussagekraft. Erst die gemeinsame Betrachtung beider Maße ist ein Indikator für die Güte eines Systems. Dazu werden recall und precision im F_1-Maß zusammengefasst:

$$F_1(r, p) = \frac{2rp}{r + p}, \tag{3.28}$$

das zwischen 0% und 100% liegt, wobei ein F_1-Wert von 100% bedeutet, dass das System alle Ereignisse in den Daten detektiert hat und dabei kein Ereignis zuviel erkannt hat.

Ein Schnitt kann bildgenau festgelegt und erkannt werden. Dies ist jedoch für Szenen- und Kapitelwechsel nicht immer sinnvoll: Zum einen können Szenenwechsel häufig nicht bildgenau festgelegt werden, zum anderen ist eine Erkennung auf bildgenauer Basis häufig weder möglich noch notwendig. Für die Szenen- und Kapitelerkennung wird daher ein zeitlicher Offset eingeführt. Ein Ereignis, das innerhalb dieses Offsets detektiert wird, wird als richtig erkannt gewertet. Erkennungsergebnisse für die Szenen- und Kapitelerkennung werden daher im Folgenden immer in Abhängigkeit von verschiedenen Offsets angegeben.

3.5.3 Schnitterkennung

Für die Evaluation der automatischen Schnitterkennung werden zwei Experimente mit verschiedenen Trainingskonfigurationen durchgeführt. In der ersten Konfiguration sind die Trainings- und Testdaten ähnlich: Die erste Hälfte jeder Serie aus den Daten wird für das Training der Modelle verwendet. In der Trainingsphase stehen daher alle Genres zur Verfügung. Die zweite, unbekannte Hälfte jeder Serie wird dann für den Test verwendet. Für die zweite Konfiguration wird ein realistisches Trainingsszenario gewählt: Es wird lediglich eine vollständige Serie zum Training verwendet. Es steht also während der Trainingsphase nur ein Genre zur Verfügung. Alle anderen unbekannten Serien aus anderen Genres werden dann zum Test verwendet. Im realistischen Szenario können die Testdaten daher vollkommen andere Eigenschaften als die Trainingsdaten aufweisen.

Die Erkennungsergebnisse für beide Szenarien sind in Tabelle 3.1 sowohl für das hier vorgestellte GM als auch für das Schwellwertvergleichsverfahren gezeigt. Dabei sind die Ergebnisse nach recall, precision und F_1-Maß aufgeschlüsselt.

Wie zu erwarten, sind für beide Systeme die Erkennungsraten für die ähnliche Konfiguration besser als für die realistische. Im ähnlichen Szenario erreicht das Vergleichsverfahren einen F_1-Wert von 92%, das GM erreicht hier 95%. Dies entspricht einer relativen Fehlerreduktion von 37,5% bei Verwendung identischer Merkmale für beide Verfahren.

Bei der realistischen Konfiguration wird diese Tendenz deutlich verstärkt: Das Schwellwertverfahren generalisiert aus der limitierten Menge an Trainingsdaten, die nur aus einem Genre stammen, nur unzureichend auf die anderen Genres. Es erzielt

Tabelle 3.1: Recall-, Precision- und F_1-Ergebnisse für die automatische Schnitterkennung mit dem vorgestellten GM und dem Schwellwertverfahren. Sowohl für das ähnliche als auch das realistische Trainingsszenario erkennt das GM mehr Schnitte richtig als das Schwellwertverfahren.

	GM (in %)			Schwellwert (in %)		
Training	Recall	Precision	F_1	Recall	Precision	F_1
Konf. 1: ähnlich	98,5	91,8	95,0	96,7	87,6	92,0
Konf. 2: realistisch	94,8	88,0	91,3	84,8	73,5	78,7

daher nur einen F_1-Wert von 78,7% und verliert mehr als 13% auf die ähnliche Konfiguration. Das GM hingegen generalisiert auch aus den limitierten Trainingsdaten und dem einen Genre noch verhältnismäßig gut auf die anderen Genres. Für die realistische Konfiguration fällt der F_1-Wert daher nur um weniger als 4% im Vergleich zur ähnlichen Konfiguration und erreicht einen absoluten Wert von 91,3%. Dies entspricht einer relativen Fehlerreduktion von mehr als 59% im Vergleich zum schwellwertbasierten Verfahren bei gleichen Daten und gleichen Merkmalen.

Bei der Interpretation der Ergebnisse ist zu beachten, dass Schwellwertmethoden bekannt sind, die bessere Ergebnisse erzielen. Diese verwenden häufig adaptive Erweiterungen und deutlich komplexere signalnahe Merkmale. Im Rahmen dieser Arbeit werden nur relativ einfache Merkmale verwendet. Die Experimente zeigen jedoch deutlich, dass das GM die Information aus diesen Merkmalen deutlich besser generalisiert und nutzt als die Schwellwertmethode. Die Schnitterkennung des GM profitiert daher deutlich von der gemeinsamen Optimierung der Schnitte und Szenen. Auch im Vergleich zu den Ergebnissen der Schnitterkennung in der TRECVID Evaluation 2004 ist der F_1-Wert von 91,3% sehr gut. Die besten Systeme in TRECVID erreichten einen F_1-Wert von knapp über 90% [140]; wobei zu beachten ist, dass in dieser Evaluation die deutlich komplexere Aufgabe des Erkennens von graduellen Schnitten enthalten war, die im Rahmen dieser Arbeit nicht behandelt wird.

3.5.4 Szenenwechselerkennung

Die Szenenerkennungsschicht des GM wird in Abhängigkeit von verschiedenen Offsets evaluiert. Wird ein in den Daten vorhandener Szenenwechsel nicht auf das Bild genau erkannt, aber innerhalb eines durch den Offset festgelegten Zeitfensters um den tatsächlichen Zeitpunkt, so wird die Erkennung trotzdem als richtig gewertet.

Die so erstellten Erkennungsergebnisse sind für vier verschiedene erlaubte Zeitfenster sowohl für das GM als auch für die Vergleichsmethode in Tabelle 3.2 gezeigt. Wie zu erwarten, steigen mit zunehmendem Offset – also zunehmender Toleranz der Erkennungsergebnisse – auch die F_1-Werte für beide Verfahren. Für alle Offsets erreicht das GM deutlich höhere Raten als die schwellwertbasierte Methode.

Tabelle 3.2: Recall-, Precision und F_1-Ergebnisse für die automatische Szenenerkennung mit dem vorgestelltem GM und dem Schwellwertverfahren. Dabei sind die Ergebnisse abhängig vom erlaubten Offset: Die Erkennung wird als richtig gewertet, wenn das System einen Wechsel innerhalb des gegebenen Zeitintervalls detektiert.

	GM (in %)			Schwellwert (in %)		
Offset	Recall	Precision	F_1	Recall	Precision	F_1
+/- 2s	44	63	52	8	11	9
+/- 5s	44	63	52	11	14	12
+/- 10s	49	70	58	23	29	26
+/- 20s	56	78	65	48	62	54

Für ein Zeitfenster von +/- 20 Sekunden erreicht das Schwellwertverfahren einen F_1-Wert von 54% und das GM 65%. Allerdings erscheint ein Toleranzbereich von mehr als zehn Sekunden für die Szenenerkennung nicht sinnvoll, da teilweise sogar die Szenendauer kürzer ist. Für kleinere Offsets sinken die F_1-Werte der Schwellwertmethode deutlich, so wird bei zehn Sekunden Offset nur noch ein F_1-Wert von 26% erreicht. Bei einem sehr engen Zeitfenster von zwei Sekunden sinkt der F_1-Wert auf unter 10%, das Verfahren wird damit praktisch unbrauchbar.

Im Vergleich dazu sinkt auch bei dem hier vorgestellten GM der F_1-Wert mit kleineren Offsets ab. Jedoch ist der Rückgang deutlich geringer als bei der Vergleichsmethode. So erreicht das GM bei einem Zeitfenster von zehn Sekunden immer noch einen F_1-Wert von 58% und bei dem sehr engen Zeitfenster von zwei Sekunden 52%. Dies zeigt, dass auch die Szenenschicht des GM die Merkmale zur Detektion von Szenenwechseln deutlich besser verwerten kann als das Vergleichsverfahren. Das GM profitiert auch in dieser Schicht von der gemeinsamen Optimierung von Schnitten und Szenen.

Trotzdem ist das erreichte F_1-Ergebnis von 58% für ein relativ tolerantes Zeitfenster von zehn Sekunden für eine praktische Anwendung noch nicht ausreichend. Für bessere Ergebnisse würden jedoch mehr semantisch höherwertige Merkmale benötigt. Insbesondere die Auswertung und Interpretation der gesprochenen Sprache könnte voraussichtlich deutlich zur Erkennung von Szenenwechseln beitragen.

3.5.5 Kapitelwechselerkennung

Im letzten Experiment wird das gleiche GM mit denselben Merkmalen verwendet, allerdings wird die Szenenschicht des Modells durch eine Kapitelschicht ersetzt. Dazu wird die Modellstruktur beibehalten, allerdings in der zweiten Schicht keine Szenen-, sondern Kapitelwechsel trainiert. Kapitel formen wesentlich größere Einheiten eines Videos. In einer Episode des vorhandenen Datenmaterials sind deutlich weniger

Tabelle 3.3: Recall-, Precision und F_1-Ergebnisse für die automatische Kapitelerkennung mit dem vorgestelltem GM und dem Schwellwertverfahren. Wieder in Abhängigkeit eines Offsets: Die Erkennung wird als richtig gewertet, wenn das System einen Wechsel innerhalb des gegebenen Zeitintervalls detektiert.

	GM (in %)			Schwellwert (in %)		
Offset	Recall	Precision	F_1	Recall	Precision	F_1
+/- 2s	56	27	36	13	5	7
+/- 5s	63	30	41	13	5	7
+/- 10s	63	30	41	29	11	16
+/- 20s	75	37	50	54	21	30

Kapitel als Szenen. Kapitel sind noch weniger strikt definiert und daher deutlich schwerer zu erkennnen, normalerweise wird zumindest ein rudimentäres Verständnis des Inhalts benötigt.

Tabelle 3.3 zeigt die Ergebnisse für die Kapitelerkennung. Die F_1-Werte sind für alle erlaubten Offsets niedriger als die vergleichbaren Szenenerkennungswerte. Trotzdem erzielt das GM wiederum deutlich bessere Werte als das Schwellwertverfahren. Wiederum zeigt sich, dass die kombinierte Optimierung in beiden Ebenen zu einer besseren Informationsverwertung führt und daher bessere Ergebnisse als beim Schwellwertverfahren erreicht werden können. Berücksichtigt man weiterhin die relativ limitierten semantisch höherwertigen Merkmale, ist das erreichte F_1-Ergebnis von 50% mit dem GM für ein erlaubtes Zeitfenster von 20 Sekunden vielversprechend. Sobald das automatische Interpretieren der gesprochenen Sprache möglich ist erscheint es daher sinnvoll, das vorgestellte Modell auf eine dritte Ebene zu erweitern, so dass Schnitte, Szenen und Kapitel gleichzeitig ausgewertet werden.

3.6 Kapitelzusammenfassung und Ausblick

In diesem Kapitel wurde ein im Rahmen dieser Arbeit entworfenes GM zur kombinierten Schnitt- und Szenenwechselerkennung in Videos vorgestellt. Das Modell integriert signalnahe und semantisch höherwertige Merkmale in zwei Modellebenen und optimiert damit die Anordnung von Schnitten und Szenenwechseln im Verbund.

In Experimenten wurde das hier entworfene GM mit einem Standard-Schwellwertverfahren verglichen. Für die Schnitterkennung erreicht das GM einen F_1-Wert von 91,3%, eine Verbesserung von 12,6% gegenüber dem Schwellwertverfahren, das entspricht einer relativen Fehlerreduktion von über 59%. Für die Szenenerkennung mit einer Toleranz von 20 Sekunden konnte die Erkennungsrate um 11% gegenüber dem Vergleichsverfahren gesteigert werden. Bei einer Toleranz von zehn Sekunden für die Szenenerkennung konnte mit dem GM ein F_1-Wert von 58% erreicht werden. In

Abbildung 3.5: Im Rahmen dieser Arbeit prototypisch entworfene Anwendung des GM am Beispiel von zwei Videos. Anhand der semantisch höherwertigen Merkmale wird in den Videos automatisch der Ort und die in der Szene anwesenden Personen angezeigt sowie der augenblickliche Sprecher hervorgehoben. Im unteren Teil des Bildes werden neben Zeit- und Frameinformation auch die automatisch erkannte Szene und Kameraeinstellung (*shot*) angezeigt.

einem prototypischen Experiment wurde die Szenenebene durch eine Kapitelebene ausgetauscht. Auch hier erreichte das GM gegenüber dem Vergleichsverfahren deutliche Verbesserungen. Allerdings konnte bei einem Toleranzbereich von 20 Sekunden nur ein F_1-Wert von 50% erreicht werden.

Die Experimente bestätigen die theoretische Annahme, dass in dem GM beide Ebenen von der gemeinsamen Optimierung profitieren. In allen Experimenten konnten die Erkennungsraten gegenüber dem Vergleichsverfahren gesteigert werden. Bei gleichem Trainingsmaterial und gleichen Merkmalen konnte also durch die Verbundoptimierung mehr Information genutzt werden.

Die Ergebnisse der Schnitterkennung sind bereits sehr gut und mit F_1-Werten über 90% auch praktisch nutzbar. Hingegen sind die Szenen- und Kapitelwechselerkennung noch zu fehleranfällig. Dies liegt zum Teil an den relativ einfachen verwendeten Merkmalen. Bessere semantisch höherwertige Merkmale – z. B. die Interpretation der gesprochenen Sprache – führen hier voraussichtlich zu besseren Ergebnissen; allerdings werden die Verfahren zur automatischen Analyse der Sprache zur Zeit selbst noch erforscht und sind daher gegenwärtig noch nicht einsetzbar.

Die – trotz der einfachen semantischen Merkmale – vielversprechenden Ergebnisse des Modells lassen jedoch eine praktische Anwendung erwarten, sobald bessere Merkmale automatisch eingesetzt werden können. Das Kapitelerkennungsexperiment zeigt auch, dass eine Erweiterung auf eine dritte Kapitelebene sinnvoll sein könnte. Das vorgestellte GM ist hier leicht erweiterbar.

Abbildung 3.5 veranschaulicht prototypisch eine in dieser Arbeit entstandene Anwendung des hier vorgestellten GM anhand von zwei Videos: In jedes Bild der Videos wird automatisch die erkannte Hintergrundinformation eingeblendet. Im obe-

ren Teil werden die automatisch extrahierten, semantisch höherwertigen Merkmale angezeigt: der Ort, die in der Szene anwesenden Personen, und hervorgehoben, der aktuelle Sprecher. Unten im Bild sind, neben der Zeit und dem augenblicklichen Frame, die mit dem GM automatisch erkannte Szene und Kameraeinstellung eingeblendet.

4

Bimodale Fusion mit dem AHMM

In einem multimodalem Bediensystem [118] können die Nutzer verschiedene Modalitäten – wie z. B. Sprache und Gesten – gleichzeitig oder abwechselnd nutzen, um eine Systemeingabe zu erzielen. Dadurch wird die Mensch-Maschine-Interaktion natürlicher und intuitiver [118]. Aufgrund unterschiedlicher Benutzermuster [119] kann es bei solchen Systemen jedoch zu unterschiedlichen Anordnungen zwischen den Merkmalsequenzen der Modalitäten kommen (siehe Abschnitt 4.1). Für die Fusion der einzelnen Modalitäten zu einem gültigen Systemkommando müssen diese Asynchronitäten zwischen den Merkmalsequenzen ausgegelichen werden. In dieser Arbeit wird dazu das asynchrone Hidden Markov Modell (AHMM) [17] verwendet.

Das AHMM ist eine Erweiterung des HMM, das die Verbundwahrscheinlichkeit von zwei dynamischen Merkmalsequenzen modellieren kann, auch wenn diese asynchron zueinander sind. Das AHMM ähnelt dem IOHMM [20] (Abschnitt 2.10.4) und dem Paar-HMM [56], die ebenfalls beide für die multimodale Fusion eingesetzt werden. Das hier vorgestellte AHMM hat jedoch kaum Gemeinsamkeiten mit dem gleichnamigen asynchronen HMM, das in [63] vorgestellt wurde. Dieses wird verwendet, um in einem Datenstrom Datenverluste auszugleichen. Es wird – entgegen dem hier vorgestellten Konzept – nicht für die multimodale Fusion verwendet.

Das hier verwendete AHMM wird genutzt, um die gemeinsame Auftrittswahrscheinlichkeit von zwei Datenströmen, die die gleiche Klasse beschreiben, zu modellieren. Das im Rahmen dieser Arbeit behandelte AHMM wurde in verschiedenen Arbeiten bereits erfolgreich für Fusionsprobleme angewendet: In [17, 19] wird das Modell zur audio-visuellen Spracherkennung eingesetzt, in [18] zur multimodalen Personenidentifikation und in [170] wird das AHMM als erste Stufe eines zweistufigen HMM zur Erkennung von Ereignissen in Konferenzen verwendet.

Das AHMM generalisiert die Verbindung zwischen den Datenströmen, sogar wenn diese nicht synchron sind. Dies ist der Hauptvorteil des Modells gegenüber anderen Markov-Modellen, wie dem gekoppelten HMM (siehe Abschnitt 2.10.4). Der Hauptnachteil des AHMM ist die sehr hohe Rechenkomplexität, sowohl für das Lernen als auch für die Vorwärtsberechnung beim Dekodieren. Das Modell wird für hohe

Asynchronitäten und lange Merkmalsequenzen praktisch nicht mehr berechenbar.

Im Rahmen dieser Arbeit [11] wird deshalb die Vorwärtsberechnung des Modells vereinfacht: Im Trellis-Diagramm werden unmögliche Pfade von vornherein ausgeschlossen. Dadurch kann die Rechenkomplexität des AHMM erheblich reduziert werden. Trotzdem ist die entwickelte Methode mathematisch exakt, da nur tatsächlich nicht benötigte Punkte auch nicht berechnet werden. Durch dieses verbesserte Verfahren kann das AHMM auf eine größere Anzahl von Problemen angewendet werden, insbesondere Probleme, bei denen die Merkmalströme sehr lang sind oder bei denen die Asynchronität zwischen den Merkmalströmen sehr hoch ist. Dadurch kann das AHMM in dieser Arbeit auch auf das Problem der Fusion bimodaler Benutzereingabedaten angewendet werden (Abschnitt 4.4). Außerdem erlaubt der entwickelte Algorithmus das Anwenden einer Skalierungsprozedur, um die Berechnungen auf einen numerisch handhabbaren Wertebereich zu begrenzen. Eine solche exakte Skalierungsprozedur ist mit der ursprünglichen Vorwärtsberechnung nicht möglich.

4.1 Asynchronitäten in der multimodalen Fusion

Beim Zusammenfügen von mehreren Datenströmen kann es auf zwei Arten zu Asynchronitäten kommen. Zum einen kann es sich bei den Datenströmen eigentlich um synchrone Prozesse handeln, die jedoch durch einen technischen Fehler asynchron zueinander werden. Zum anderen kann die Asynchronität aber auch schon in den Datenströmen vorliegen und durchaus gewollt sein. Beide Arten sollen im Folgenden kurz erläutert werden.

4.1.1 Asynchronitäten durch technische Fehler

Technische Fehler können zu Asynchronitäten in einem zusammengefügten Datenstrom führen, wenn die beiden Ausgangsdatenströme das selbe Ereignis beschreiben und prinzipiell synchron zueinander sind. Dies ist in Abbildung 4.1 beispielhaft für zwei Datenströme $\vec{x}$ und $\vec{y}$ gezeigt. Bei voller Synchronität der beiden Ausgangsdatenströme ergibt sich der fusionierte Merkmalstrom $\vec{z}$, wie in (a) gezeigt. Startet die Aufnahme der beiden Ströme jedoch um nur einen einzigen Frame versetzt (b), ergibt sich ein stark verfälschter fusionierter Ausgabestrom. Auch für den Fall eines Frameverlusts (c) oder einer Frameverdopplung (d) – was beides z. B. bei Web-Kameras auftreten kann – wird der Ausgabestrom ebenfalls stark verfälscht.

Dabei ist insbesondere zu beachten, dass ein Fehler in einem der beiden Ausgangsströme erhebliche Folgefehler im fusionierten Strom nach sich zieht. Üblicherweise werden erst wieder korrekte Symbole erzeugt, wenn ein neuer Synchronisationspunkt erreicht wird. Je nach Häufigkeit dieser Synchronisationspunkte im System können daher durch einen einzigen falschen Frame längere Sequenzen des fusionierten Merkmalstroms unbrauchbar werden.

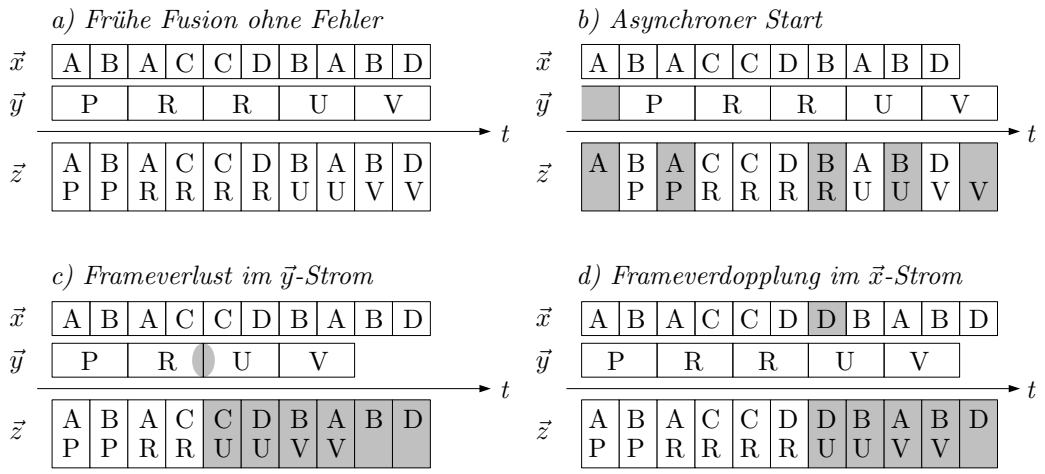

Abbildung 4.1: Asynchronitäten in der multimodalen Fusion, bedingt durch verschiedene technische Fehler. Fehlerhafte Merkmale sind grau hinterlegt. Ein einzelner Fehler in einem der beiden Ausgangsströme $\vec{x}$ oder $\vec{y}$ kann zu erheblichen Folgefehlern im fusionierten Strom $\vec{z}$ führen. Diese Fehler enden in der Regel erst wieder, wenn der nächste Synchronisationspunkt erreicht wird.

Das AHMM kann solche Fehler auch ohne Synchronisation abfangen: Das Konzept lässt zu, dass die beiden Ströme in der Dekodierungsphase zueinander verschoben werden. Ein einzelner Fehler in einem der beiden Ströme führt daher beim AHMM auch nur zu einem einzelnen Fehler im fusionierten Strom, Folgefehler wie in Abbildung 4.1 treten nicht mehr auf.

4.1.2 Asynchronitäten in multimodalen Dialogsystemen

Im Gegensatz zu Asynchronitäten, die durch technische Fehler beim Zusammenfügen von Daten entstehen können, sind die Asynchronitäten in multimodalen Dialogsystemen gewollt. Multimodale Bediensysteme [118] sind so gestaltet, dass die Mensch-Maschine-Interaktion natürlicher und intuitiver für den Benutzer ist. Dazu können mehrere Eingabemedien – z. B. Tastatur, Maus, Sprach- und Stifteingabe – gleichzeitig oder nacheinander genutzt werden. Außerdem können die Nutzer die Eingabemedien wechseln, um auf verschiedene Arten die gleiche Systemeingabe zu erreichen.

Die Herausforderung solcher Systeme ist es, die verschiedenen Eingaben der verschiedenen Medien zu gültigen Systemkommandos zusammenzufügen. Dabei muss die Fusion sowohl über die Zeit als auch über die verschiedenen Medien erfolgen. Insbesondere die bimodale Kombination von Sprache und Geste wurde dabei in den letzten 20 Jahren erforscht. Das bekannteste Beispiel dafür ist Bolts „Put-that-there“ Anwendung [32], bei der Benutzer mit einer beliebigen Kombination aus Sprache und Geste verschiedene Objekte selektieren, verschieben oder manipulieren können. Da-

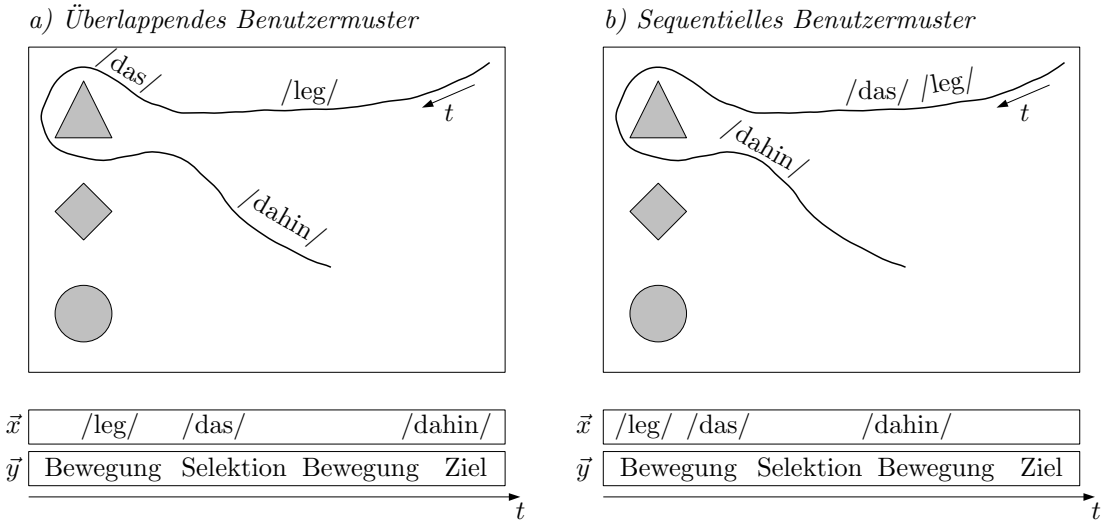

Abbildung 4.2: Asynchronität in einem Sprach-Gesten-Eingabesystem durch unterschiedliche Nutzermuster. Links und rechts wird das gleiche Objekt (Dreieck) mit der gleichen Gestentrajektorie (schwarzer Linie) ausgewählt. Es werden auch die gleichen Worte gesprochen, nur die zeitliche Anordnung im Verhältnis zur Geste ist unterschiedlich. Dadurch ergeben sich die unten gezeigten Gesten- und Sprachdatenströme. Bis auf die asynchrone Anordnung sind die Ströme für das überlappende und das sequentielle Benutzermuster identisch. Ein bimodales System muss beide Sequenzen trotz der asynchronen Verzerrung als das selbe Systemkommando erkennen.

bei kann es im kombinierten Sprach-Gesten-Strom zu erheblichen Asynchronitäten kommen, die von der Art der Nutzung abhängen.

Dies ist in Abbildung 4.2 gezeigt: Zwei verschiedene Nutzer selektieren ein Objekt und weisen das System mit einer Spracheingabe an, es zu verschieben. Obwohl in beiden Fällen die gleiche Gestentrajektorie und die gleichen Worte benutzt werden, erhält man unterschiedliche fusionierte Datenströme, da die Worte einmal überlappend mit der Selektion und einmal bereits vor der Selektion gesprochen werden. Das gleiche Systemkommando kann also aufgrund unterschiedlicher Verhaltensmuster unterschiedlicher Nutzer [119] stark unterschiedlich ausfallen. Letztendlich liegt aber nur eine asynchrone Verschiebung der Daten zueinander vor. Verschiedene Ansätze zum Lösen dieses Fusionsproblems wurden bereits vorgeschlagen [13, 32, 118, 119], fast alle Methoden beruhen jedoch auf einem hohen Wissen über die Domäne, um das Fusionsproblem zu lösen.

Da bei diesem Fusionsproblem die Asynchronität sehr groß werden kann, konnte das AHMM bisher aufgrund der hohen Komplexität nicht angewendet werden. Durch den in dieser Arbeit verbesserten Vorwärts-Algorithmus und die dadurch erzielte Komplexitätsreduktion kann das AHMM jedoch auch mit sehr großen Asyn-

chronitäten arbeiten. In Abschnitt 4.4 wird daher anhand von Experimenten gezeigt, wie das AHMM verwendet werden kann, um die Asynchronität in einer bimodalen Sprach- und Gestenanwendung auszugleichen. Dazu wird keinerlei Domänenkenntnis benötigt, wodurch das System sehr flexibel ist und leicht erweitert werden kann.

4.2 Das asynchrone Hidden Markov Modell

Das AHMM modelliert die Verbundwahrscheinlichkeit $p(\vec{x}, \vec{y})$ zweier Beobachtungssequenzen $\vec{x}$ mit der Länge T und $\vec{y}$ mit der Länge S. Dabei werden $\vec{x}$ und $\vec{y}$ auch als gemeinsame Observierung $\vec{o} = \{\vec{x}, \vec{y}\}$ bezeichnet. Beide Beobachtungssequenzen beschreiben dasselbe Ereignis (d. h. Musterklasse), können aber sowohl unterschiedliche Abtastraten aufweisen, innerhalb der Vektoren asynchrone Verzerrungen haben, als auch zeitlich vollständig asynchron zueinander sein. Ohne Beschränkung der Allgemeinheit gelte, dass $S \leq T$ ist (sollte $\vec{y}$ länger als $\vec{x}$ sein, ist eine triviale Erweiterung des AHMM Konzeptes notwendig). Zugunsten einer durchgängigeren Schreibweise und ohne Beschränkung der Allgemeinheit gelte außerdem, dass die Beobachtungssequenzen $\vec{x}$ und $\vec{y}$ nur diskrete Symbole enthalten, d. h. $\vec{x}_t \in x^1 \dots x^X, \forall t$ und $\vec{y}_s \in y^1 \dots y^Y, \forall s$, wobei X die Kardinalität des $\vec{x}$-Symbolschatzes und Y die Kardinalität des $\vec{y}$-Symbolschatzes bezeichne. Die Erweiterung auf kontinuierliche Beobachtungssequenzen erfolgt dabei analog zum Übergang von diskreten auf kontinuierliche HMM [126], ohne die hier beschriebene grundsätzliche Funktionsweise des AHMM zu beeinflussen.

Die vom AHMM modellierte Verbundwahrscheinlichkeit $p(\vec{x}, \vec{y})$ kann nicht direkt bestimmt werden, da die Anzahl möglicher Ausrichtungen von $\vec{x}$ und $\vec{y}$ abhängig von dem Verhältnis der Längen T und S exponentiell ansteigend ist und die Berechnung daher schnell unausführbar wird. Deshalb werden zwei verborgene Variablen (*hidden variables*) eingeführt. Die erste Variable $q_t = 1 \dots N$ ist synchronisiert mit dem Merkmalstrom $\vec{x}$ und identisch zur Zustandsvariablen (*state*) in normalen HMM [14, 15, 126, 127, 166]. Dabei wird die vollständige Anzahl an möglichen Zuständen des Modells mit N bezeichnet. Analog zu normalen HMM emittiert jeder Zustand zu jedem Zeitpunkt t immer genau ein Symbol vom Strom $\vec{x}$. Weiterhin kann jeder Zustand $q_t = i$ zu jedem Zeitpunkt t gleichzeitig auch ein Symbol vom Strom $\vec{y}$ emittieren. Die Wahrscheinlichkeit, dass der Zustand $q_t = i$ sowohl ein Symbol von $\vec{x}$ als auch von $\vec{y}$ emittiert, wird mit ϵ_i[1] bezeichnet. Die Gegenwahrscheinlichkeit $\bar{\epsilon}_i = 1 - \epsilon_i$ gibt damit an, wie wahrscheinlich $q_t = i$ lediglich ein Symbol

[1] Die erste Veröffentlichung über das AHMM [17] geht von einer Wahrscheinlichkeit $\epsilon(i, t)$ aus, die nicht nur vom Zustand $q_t = i$, sondern auch vom Zeitpunkt t abhängt. Eine solche Wahrscheinlichkeit lässt sich aber weder automatisch aus Daten lernen noch implementieren, da sie von der jeweiligen Beobachtungslänge T abhängig ist. In späteren Veröffentlichungen und in allen verfügbaren Implementierungen wird ϵ_i daher tatsächlich nur noch in Abhängigkeit vom Zustand $q_t = i$ verwendet. In dieser Arbeit wird die Emissionswahrscheinlichkeit für ein Symbolpaar daher ausschließlich in Abhängigkeit vom Zustand betrachtet.

vom $\vec{x}$-Strom emittiert. Die zweite verborgene Variable $\tau_t = 0 \dots S$ modelliert die zeitliche Anordnung (*alignment*) zwischen dem $\vec{x}$- und dem $\vec{y}$-Strom. Immer wenn ein Zustand $q_t = i$ ein Symbol von $\vec{y}$ emittiert, wird die Anordnungsvariable τ_t um eins inkrementiert, bis alle Symbole von $\vec{y}$ emittiert wurden und damit $\tau_t = S$ ist.

4.2.1 Das AHMM als Graphisches Modell

Das AHMM kann als ein GM mit vier Variablen in jedem Zeitschlitz dargestellt werden. Abbildung 4.3 zeigt dieses Modell. Die zeitliche Modellierung erfolgt analog zum HMM mit einer Markov-Kette erster Ordnung: dem Zustand $q_t = i$, mit $i \in 1 \dots N, \forall t$. Analog zum HMM enthält der Zustandsknoten q_t damit im Prolog q_0 die Einsprungswahrscheinlichkeit $\vec{\pi}$, im Zeitschlitz q_t die Übergangsmatrix $\underset{\sim}{A}$ und im Epilog q_T einen optionalen Aussprungsvektor. Die Emissionswahrscheinlichkeit für Symbolpaare wird durch den binären Knoten ϵ modelliert. Dieser ist nur abhängig vom gegenwärtigen Zustand q_t. Die Emissionswahrscheinlichkeit ϵ wird damit durch eine $2 \times N$ Wahrscheinlichkeitstafel realisiert. Deterministisch abhängig vom Emissionsknoten ist die zeitliche Anordnung $\tau_t = s$, mit $s \in \{0, \dots, S\}, \forall t$ und $\tau_t \geq \tau_{t-1}, \forall t$. Jedesmal wenn der Knoten ϵ auf die Emission eines Symbolpaares entscheidet, wird der Zähler τ inkrementiert. Dies kann sowohl im Prolog τ_0 als auch im Zeitschlitzen τ_t z. B. durch einen dynamischen Entscheidungsbaumzähler implementiert werden. Damit gibt τ die zeitliche Anordnung zwischen $\vec{x}$ und $\vec{y}$ an. Um sicherzustellen, dass zum Zeitpunkt T alle Symbole vom $\vec{y}$-Strom emittiert wurden, wird im Epilog $\tau_T = S$ festgelegt. Die eigentliche Beobachtung wird durch den Observierungsknoten $\vec{o}$ modelliert. Ähnlich dem Konzept der wechselnden Eltern (Abschnitt 2.7) [26] wird hierzu ein wechselnder Kindknoten verwendet. Abhängig vom Elternknoten ϵ wird in $\vec{o}_t$ entweder nur ein Symbol von $\vec{x}$ emittiert (d. h. die Emissionswahrscheinlichkeit für das Symbol von $\vec{y}$ wird deterministisch auf eins gesetzt und hat damit keinen Einfluss auf die Gesamtwahrscheinlichkeiten in dem Zeitschlitz) oder ein Symbolpaar von $\vec{x}$ und $\vec{y}$. Falls ein Symbolpaar emittiert wird, gibt die zeitliche Anordnung τ_t deterministisch vor, welches Symbol $\vec{y}_{\tau_t}$ verwendet wird (τ dient somit als Zeiger auf das nächste Symbol von $\vec{y}$). Wie beim HMM ist die Beobachtung $\vec{o}$ auch abhängig vom gegenwärtigen Zustand $q_t = i$. Die tatsächliche Modellierung von $\vec{o}$ kann dabei je nach gewünschtem Verhalten verschieden ausgelegt sein, z. B. diskret oder kontinuierlich, als gemeinsame Verbundwahrscheinlichkeit von $\vec{x}$ und $\vec{y}$, bei gegebenem Zustand abhängig oder unabhängig voneinander.

Während sowohl die Struktur als auch die Parameter des AHMM einfach zu beschreiben sind, gibt es zur Zeit kein standartisiertes GM-Werkzeug in dem ein AHMM implementiert werden kann. Das Konzept wechselnder Kindknoten bzw. das Konzept von Null-emittierenden Zuständen ist vergleichbar mit dem Konzept wechselnder Eltern und daher sowohl in der Behandlung innerhalb des Verbundbaumes als auch in der Implementierung trivial. Jedoch werden wechselnde Kindknoten zur Zeit in keinem der verfügbaren GM-Werkzeuge zur Verfügung gestellt. Eine Imple-

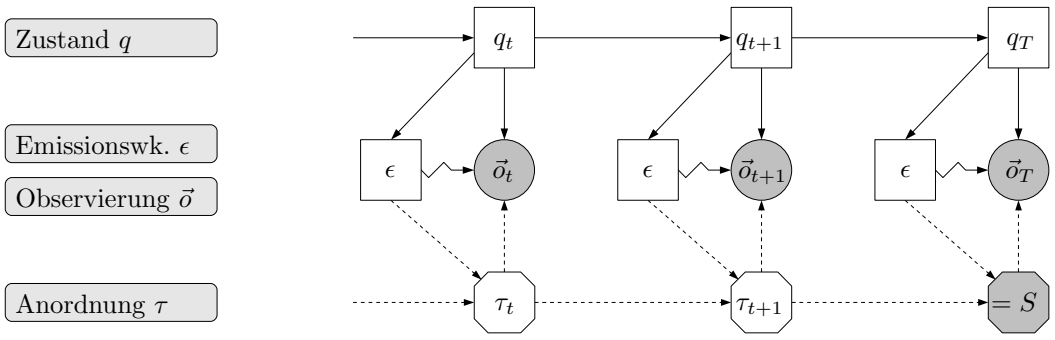

Abbildung 4.3: Graphisches Modell des AHMM. Die zeitliche Modellierung erfolgt durch eine Markov-Kette erster Ordnung (Zustand q). Weiterhin hat das Modell die binäre Emissionswahrscheinlichkeit eines Symbolpaares ϵ. Diese hängt vom gegenwärtigen Zustand q_t ab. Bei jeder Emission eines Symbolpaares wird der deterministische Zähler τ erhöht, bis $\tau_t = S$. Die Emission eines einzelnen Symboles oder eines Symbolpaares wird durch den wechselnden Kindknoten $\vec{o}$ realisiert. Dieser ist abhängig vom gegenwärtigen Zustand q_t und vom Emissionsknoten ϵ. Dabei wird abhängig vom Zustand von ϵ entweder nur das Symbol $\vec{x}_t$ emittiert oder zeitgleich auch das Symbol $\vec{y}_{\tau_t}$, wofür deterministisch auf den Zähler τ zurückgegriffen wird.

mentierung des AHMM als GM ohne wechselnde Kindknoten ist aber nicht möglich. Dadurch wird die Behandlung von Asynchronitäten in GM erheblich eingeschränkt.

Da das AHMM jedoch aus einer einfachen Markov-Kette erster Ordnung mit davon abhängigen Kindknoten besteht, handelt es sich bei dem Verbundbaum des AHMM um einen einfachen, unverzweigten Baum. Der darauf ausgeführte JT-Algorithmus degradiert damit, wie bereits in Abschnitt 2.10.2 gezeigt, zu einem Vorwärts-Rückwärts-Algorithmus. Dieser kann direkt analysiert und implementiert werden. Weiterhin konnte im Rahmen dieser Arbeit durch den direkten Ansatz eine deutliche Komplexitätsreduktion des AHMM entwickelt werden.

Im Folgenden soll daher gezeigt werden, wie das AHMM mit einem Vorwärts-Rückwärts-Algorithmus umgesetzt werden kann und welche Vorteile daraus entwickelt werden können.

4.2.2 Drei-dimensionales Trellis-Diagramm

Die Funktionsweise des AHMM kann auch in einem drei-dimensionalen Trellis-Diagramm wie in Abbildung 4.4 (links) dargestellt werden. Die Zeitachse t und die Zustandsachse q_t sind identisch zu denen in einem HMM Trellis-Diagramm. Der Wert q_t zeigt also an, in welchem verborgenen Zustand das Modell zum Zeitpunkt t ist. In dem hier gezeigtem Beispiel handelt es sich um ein links-rechts Modell (also $q_t \geq q_{t-1}, \forall t$), das außerdem im ersten Zustand startet ($q_0 = 1$) und im letzten

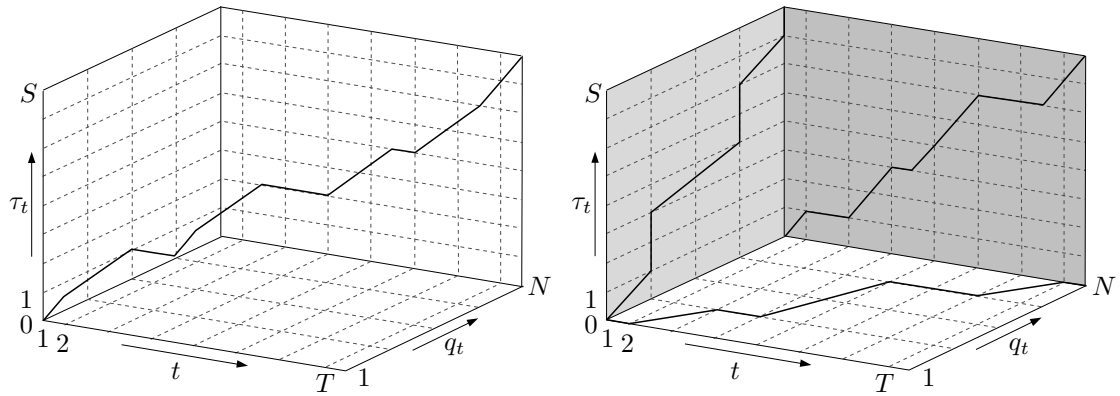

Abbildung 4.4: Drei-dimensionales Trellis-Diagramm des AHMM (links), mit den Achsen Zeit t, Zustand q_t und der Verzerrung τ_t zwischen den Merkmalströmen $\vec{x}$ und $\vec{y}$. Hier ein links-rechts-Modell, das im letzten Zustand N endet. Der beispielhafte Verlauf startet im ersten Zustand ohne ein Symbol von $\vec{y}$ zu emittieren, d. h. $\tau_0 = 0$. Mit jedem Zeitschritt t wird ein Symbol von $\vec{x}$ emittiert. Jedes Mal wenn mit der Wahrscheinlichkeit ϵ_i auch ein Symbol von $\vec{y}$ emittiert wird, erhöht sich der Zähler τ_t. Zum Endzeitpunkt T müssen alle Symbole von $\vec{y}$ emittiert sein, also $\tau_t = S$. Rechts ist die Projektionen des Pfades durch das Trellis-Diagramm auf die Ebenen gezeigt. Die untere Ebene (weiß) zeigt den Verlauf der Zustände q_t über die Zeit t, die linke Ebene (hellgrau) den Verlauf der Zustände über die zeitliche Verzerrung τ_t. Die Ebene zeigt also, welches $\vec{y}$-Symbol von welchem Zustand emittiert wurde. Die hintere Ebene (dunkelgrau) zeigt die zeitliche Verzerrung τ_t und der Pfad damit die optimale Anordnung vom $\vec{x}$ zum $\vec{y}$-Strom.

Zustand endet ($q_T = N$). Die dritte Achse des Trellis-Diagramms τ_t repräsentiert die zeitliche Verzerrung zwischen den beiden Merkmalströmen $\vec{x}$ und $\vec{y}$. Zu jedem Zeitpunkt t wird ein Symbol von $\vec{x}$ emittiert. Dies entspricht einem Schritt nach rechts in dem Trellis-Diagramm. Der hier gezeigte beispielhafte Verlauf durch das Trellis-Diagramm startet im ersten Zustand ohne ein Symbol von $\vec{y}$ zu emittieren – damit ist $\tau_0 = 0$. Wenn ein Zustand auch ein Symbol von $\vec{y}$ emittiert, entspricht das einer diagonalen Bewegung nach rechts-oben in dem Trellis-Diagramm – so zu sehen beim Übergang von $t = 1 \rightarrow 2$, q_2 emittiert also sowohl ein $\vec{x}$ als auch ein $\vec{y}$-Symbol. Als Endbedingung gilt, dass zum Zeitpunkt $t = T$ auch alle Symbole von $\vec{y}$ emittiert worden sind. Das heißt zum Zeitpunkt $t = T$ muss $\tau_T = S$ sein. Im AHMM muss ein Zustand jedoch in jedem Zeitschritt ein Symbol von $\vec{x}$ emittieren, daher ist ein Schritt nach oben ohne einen gleichzeitigen Schritt nach rechts nie möglich. Diese Tatsache wird in Abschnitt 4.2.4 dieser Arbeit ausgenutzt, um die Wahrscheinlichkeitsberechnung zu vereinfachen und die Komplexität des Algorithmus deutlich zu verringern [11]. Das Modell kann im ergodischen Fall natürlich zu jedem Zeitpunkt t

zu jedem der Zustände auf der Zustandsachse springen. In dem hier gezeigten links-rechts Modell kann der Zustand zu jedem Zeitpunkt t erhöht werden (z. B. zu sehen beim Übergang von $t = 2 \rightarrow 3$), bis der letzte Zustand N erreicht ist.

Abbildung 4.4 (rechts) zeigt die Projektionen des Pfades durch das drei-dimensionale Trellis-Diagramm auf die Ebenen. Die untere Ebene (weiß) zeigt dabei den Verlauf der Zustände q_t über die Zeit t, entspricht also einem gewöhnlichen HMM Trellis-Diagramm, bei dem zu jedem Zeitpunkt der Zustand abgelesen werden kann. Die linke Ebene (hellgrau) zeigt den Verlauf der Zustände q_t über die zeitliche Verzerrung τ_t, sie zeigt also, welches $\vec{y}$-Symbol von welchem Zustand emittiert wurde. Die wichtigste Ebene in der Projektion ist die hintere (dunkelgrau), sie zeigt die zeitliche Verzerrung τ_t über die Zeit t. Der Pfad zeigt also die optimale zeitliche Anordnung vom $\vec{x}$ zum $\vec{y}$-Merkmalstrom. Zum Vergleich: Im Falle eines einfachen HMM mit früher Fusion würde der Pfad in dieser Ebene einer Gerade vom Ursprung in die rechte obere Ecke entsprechen, da in diesem Fall stets gleichzeitig ein $\vec{x}$- und ein $\vec{y}$-Symbol emittiert wird. In Abschnitt 4.2.7 wird gezeigt, wie mittels des Viterbi-Algorithmus ohne zusätzlichen Rechenaufwand alle in Abbildung 4.4 (rechts) gezeigten Pfade berechnet werden können. Das AHMM kann dann nicht nur dazu verwendet werden, unbekannte Merkmalfolgen zu klassifizieren, sondern auch um zwei Merkmalströme optimal zeitlich aneinander auszurichten.

4.2.3 Modellparameter

Ein AHMM wird mit den Verteilungen $\lambda = \left\{\underset{\sim}{A},\, \vec{\pi},\, \vec{\epsilon},\, p(\vec{x}_t \,|\, q_t = i),\, p(\vec{x}_t,\, \vec{y}_{\tau_t} \,|\, q_t = i)\right\}$ parametrisiert:

Die Zustandseinsprungsverteilung: $\pi_i = p(q_1 = i)$ gibt an, wie wahrscheinlich das Modell zum Zeitpunkt $t = 1$ im Zustand $q_1 = i$ startet. Für alle möglichen N Zustände werden diese im Zustandseinsprungsvektor (*initial state distribution*) $\vec{\pi} = \{\pi_i\}_{i=1\ldots N}$, mit $0 \leq \pi_i \leq 1, \forall i$ und $\sum_{i=1}^{N} \pi_i = 1$ zusammengefasst, der der Einsprungswahrscheinlichkeit in normalen HMM entspricht.

Die Zustandsübergangsverteilung: $a_{ij} = p(q_{t+1} = j \,|\, q_t = i)$ gibt an, wie wahrscheinlich das Modell aus dem Zustand $q_t = i$ in den Zustand $q_{t+1} = j$ wechselt. Für alle möglichen $N \times N$ Übergänge werden diese in der Zustandsübergangsmatrix (*state transition distribution*) $\underset{\sim}{A} = \{a_{ij}\}_{i,j=1\ldots N}$, mit $0 \leq a_{ij} \leq 1, \forall i, j$ und $\sum_{j=1}^{N} a_{ij} = 1$ zusammengefasst, die wiederum der Zustandsübergangsmatrix in normalen HMM entspricht.

Die Symbolpaaremissionswahrscheinlichkeit: $\epsilon_i = p(\tau_t = s \,|\, \tau_{t-1} = s-1,\, q_t = i)$ gibt an, wie wahrscheinlich das Modell im Zustand $q_t = i$ nicht nur ein Symbol von $\vec{x}$, sondern gleichzeitig auch ein Symbol von $\vec{y}$ emittiert. Für alle N Zustände werden diese im Emissionsvektor (*pair emission distribution*) $\vec{\epsilon} = \{\epsilon_i\}_{i=1\ldots N}$, mit $0 \leq \epsilon_i \leq 1, \forall i$ zusammengefasst.

Die Einzelsymbolemissionswahrscheinlichkeit: $p(\vec{x}_t \,|\, q_t = i)$, gibt an, mit welcher Wahrscheinlichkeit das Modell im Zustand $q_t = i$ das Symbol $\vec{x}_t$ emittiert. Diese Emissionsverteilung (*single symbol emission distribution*) ist dabei wiederum identisch zu der normaler HMM. Sie kann als diskrete multinormale Verteilung oder auch kontinuierlich (z.B. mit Gauß Mixturen) modelliert werden.

Die Paarsymbolemissionswahrscheinlichkeit: $p(\vec{x}_t,\, \vec{y}_{\tau_t} \,|\, q_t = i)$ gibt an, wie wahrscheinlich das Modell im Zustand q_t das Symbolpaar $\vec{x}_t$ und $\vec{y}_{\tau_t}$ emittiert. Diese Emissionsverteilung (*symbol pair emission distribution*) kann wiederum diskret multinormal oder auch kontinuierlich modelliert werden. Weiterhin kann die Auftrittswahrscheinlichkeit für das Symbolpaar $\vec{x}_t$ und $\vec{y}_{\tau_t}$ verschieden formuliert werden: direkt als Verbundauftrittswahrscheinlichkeit $p(\vec{x}_t, \vec{y}_{\tau_t} \,|\, q_t = i)$ oder bei gegebenem Zustand $q_t = i$ voneinander unabhängig $p(\vec{x}_t, \vec{y}_{\tau_t} \,|\, q_t = i) = p(\vec{x}_t \,|\, q_t = i)\; p(\vec{y}_{\tau_t} \,|\, q_t = i)$ oder bedingt voneinander abhängig $p(\vec{x}_t,\, \vec{y}_{\tau_t} \,|\, q_t = i) = p(\vec{y}_{\tau_t} \,|\, \vec{x}_t,\, q_t = i)\; p(\vec{x}_t \,|\, q_t = i)$. Dies erlaubt eine problemabhängige Flexibilität, die in anderen multimodalen Markov-Modellen häufig nicht möglich ist. So zum Beispiel in einem frühen Fusions HMM, in dem die Symbolpaaremission grundsätzlich als Verbundwahrscheinlichkeit formuliert sein muss.

4.2.4 Vereinfachte Vorwärtsberechnung

Um das AHMM zur Klassifizierung von unbekannten Mustern $\vec{o} = (\vec{x},\, \vec{y})$ zu verwenden, wird – wie auch bei anderen HMM Typen – die Klasse k^* mit der höchsten Musterproduktionswahrscheinlichkeit

$$k^* = \underset{k \in K}{\operatorname{argmax}}\; p(\vec{x},\, \vec{y} \,|\, \lambda_k) \tag{4.1}$$

ausgewählt. Dazu ist es erforderlich, die Verbundwahrscheinlichkeit

$$p(\vec{x},\, \vec{y} \,|\, \lambda_k) \tag{4.2}$$

zu berechnen. Dies ist nicht direkt möglich. Mit den eingeführten Hilfsvariablen q_t und τ_t wurde jedoch ein Vorwärts-Algorithmus entwickelt [17], der eine relativ effiziente Berechnung erlaubt. Im Rahmen dieser Arbeit wird dieser Algorithmus noch einmal deutlich vereinfacht und dadurch die Komplexität der Verbundwahrscheinlichkeitsberechnung signifikant reduziert. Zuerst soll der in [17] entwickelte Vorwärts-Algorithmus hier detailliert erläutert werden, ohne jedoch auf Bereichsgrenzen für die Berechnungen einzugehen. Die für den Vorwärts-Algorithmus benötigte Vorwärtspfadvariable α ist **definiert** als:

$$\alpha(i, s, t) = p(q_t = i, \tau_t = s, \vec{x}_t, \vec{y}_s). \tag{4.3}$$

Das Modell kann im ersten Zeitschritt entweder nur ein Symbol von $\vec{x}$ oder ein Symbolpaar von $\vec{x}$ und $\vec{y}$ emittieren. Der **Initialisierungsschritt** ist daher:

$$\alpha(i,0,1) = \pi_i \, (1-\epsilon_i) \, p(\vec{x}_1 \,|\, q_1 = i), \tag{4.4}$$

$$\alpha(i,1,1) = \pi_i \, \epsilon_i \, p(\vec{x}_1, \, \vec{y}_1 \,|\, q_1 = i). \tag{4.5}$$

Solange noch kein Symbol von $\vec{y}$ emittiert wurde (d. h. $s = 0$), ist der **Induktionsschritt**:

$$\alpha(i,s,t) = (1-\epsilon_i) \, p(\vec{x}_t \,|\, q_t = i) \, \sum_{j=1}^{N} a_{ji} \, \alpha(j,0,t-1). \tag{4.6}$$

Sobald einmal ein Symbol von $\vec{y}$ emittiert wurde (d. h. $s > 0$), verändert sich der **Induktionsschritt** zu:

$$\begin{aligned} \alpha(i,s,t) = {} & \epsilon_i \, p(\vec{x}_t, \vec{y}_s \,|\, q_t = i) \, \sum_{j=1}^{N} a_{ji} \, \alpha(j,s-1,t-1) \\ & + (1-\epsilon_i) \, p(\vec{x}_t \,|\, q_t = i) \, \sum_{j=1}^{N} a_{ji} \, \alpha(j,s,t-1). \end{aligned} \tag{4.7}$$

Und zum endgültigen Berechnen der Verbundwahrscheinlichkeit gilt die **Terminierung**:

$$p(\vec{x}, \, \vec{y} \,|\, \lambda_k) = \sum_{j=1}^{N} \alpha(j,S,T). \tag{4.8}$$

Der in [17] vorgeschlagene AHMM Vorwärts-Algorithmus hat keine Bereichsgrenzen für die drei Parameter i, s und t in der Vorwärtsberechnung angegeben. Jedoch ist die Komplexität des Algorithmus mit $\mathcal{O}(N^2ST)$ angegeben. Jeder Induktionsschritt benötigt ungefähr N Additionen. Daher wird diese Komplexität erreicht, wenn die Vorwärtsvariable α in Gleichung (4.6) und (4.7) für alle Kombinationen von $1 \leq i \leq N$, $1 \leq t \leq T$ und $0 \leq s \leq S$ berechnet wird.

Abbildung 4.5 zeigt das AHMM Trellis-Diagramm, wenn die Zustandsachse vernachlässigt wird – d. h. eine Projektion auf die (t,τ_t)-Ebene. In dieser Abbildung entspricht der ursprüngliche AHMM Vorwärts-Algorithmus dem Berechnen von allen Punkten innerhalb der (t,τ_t)-Ebene (und natürlich dem Berechnen von jeweils einer Ebene für jeden Zustand i). Dies ist jedoch nicht notwendig.

Zur vereinfachten Erläuterung sollen im Weiteren die Zustände q_t des AHMM ohne Beschränkung der Allgemeinheit vernachlässigt werden. In Abbildung 4.5 sind die Initialisierungspunkte durch schwarze Quadrate gekennzeichnet. Das Modell kann nur auf zwei Arten initialisiert werden: Entweder es emittiert gemäß Gleichung (4.4) nur ein Symbol von $\vec{x}$ (also $\tau_0 = 0$) oder es emittiert gemäß Gleichung (4.5) bereits im ersten Schritt sowohl ein Symbol von $\vec{x}$, als auch von $\vec{y}$ (d. h. $\tau_0 = 1$). Der Terminierungsschritt aus Gleichung (4.8) erzwingt, dass alle Symbole sowohl von $\vec{x}$, als auch

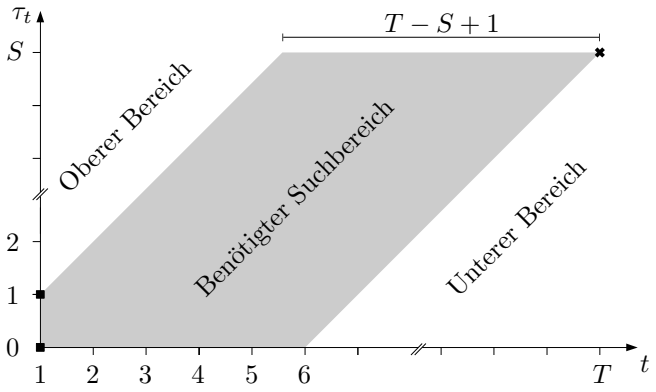

Abbildung 4.5: Zwei-dimensionale Projektion des AHMM Trellis-Diagramms auf die (t,τ_t)-Ebene. Die schwarzen Quadrate markieren die möglichen Initialisierungspunkte, das schwarze Kreuz den Terminierungspunkt. Der ursprüngliche Vorwärts-Algorithmus berechnet alle Punkte innerhalb des Diagramms. Tatsächlich berechnet werden müssen jedoch nur die Punkte in dem grau gekennzeichneten Suchbereich, von allen anderen Punkten aus gibt es keinen gültigen Pfad zum Endpunkt.

von $\vec{y}$ emittiert sein müssen, es muss also $t = T$ und $\tau_T = S$ gelten. In Abbildung 4.5 ist dieser Punkt durch ein schwarzes Kreuz gekennzeichnet. Sowohl die Initialisierungspunkte als auch der Terminierungspunkt sind durch die Vorwärtsberechnung also genau festgelegt.

Zuerst soll der obere Bereich des Trellis-Diagramms betrachtet werden. Zu jedem Zeitpunkt t wird ein Symbol von $\vec{x}$ emittiert. Zusätzlich kann mit der Wahrscheinlichkeit ϵ_i auch ein Symbol von $\vec{y}$ emittiert werden. Daher kann das Modell zu keinem Zeitpunkt mehr Symbole von $\vec{y}$ als von $\vec{x}$ emittiert haben, d. h. $\tau_t \leq t, \forall\, t$. Jedoch gilt in dem oberen Bereich $\tau_t > t$. Mit dem ursprünglichen Vorwärts-Algorithmus werden die $\alpha(i,s,t)$-Werte innerhalb dieses Bereichs berechnet und es gilt im Allgemeinen auch $\alpha(i,s,t) > 0$, aber die Werte werden für den Terminierungsschritt nicht wiederverwendet. Die Berechnung der Punkte innerhalb des oberen Bereichs ist daher nicht erforderlich.

Nun soll der untere Bereich des Trellis-Diagramms betrachtet werden. Es gilt wiederum, dass zu jedem Zeitpunkt t ein Symbol von $\vec{x}$ emittiert wird und zusätzlich mit der Wahrscheinlichkeit ϵ_i auch ein Symbol von $\vec{y}$. Das AHMM kann aber zu keinem Zeitpunkt nur ein einzelnes Symbol von $\vec{y}$ emittieren. Es ist innerhalb des Trellis-Diagramms daher nicht möglich, nur einen Schritt nach oben ohne gleichzeitig auch einen Schritt nach rechts zu machen. Die einzigen gültigen Schritte innerhalb des Diagramms sind nach rechts oder nach rechts-oben. Von allen Punkten innerhalb des unteren Bereichs kann der Terminierungspunkt daher nicht mehr mit gültigen Schritten erreicht werden. Der ursprüngliche Vorwärts-Algorithmus berechnet auch die Punkte innerhalb des unteren Bereichs. Im Allgemeinen gilt auch für diese Werte

$\alpha(i,s,t) \geq 0$; sie werden jedoch nicht im Terminierungsschritt verwendet. Daher ist eine Berechnung der Punkte im unteren Bereich nicht erforderlich.

Zusammenfassend können nur Punkte innerhalb des in Abbildung 4.5 grau gekennzeichneten benötigten Suchbereichs tatsächlich auf einem gültigen Pfad im Terminierungspunkt enden. Daher haben auch nur diese Punkte einen Einfluss auf die gesuchte Wahrscheinlichkeit $p(\vec{x},\, \vec{y}\,|\,\lambda_k)$. Deshalb müssen auch nur die $\alpha(i,s,t)$-Werte innerhalb dieses Bereichs berechnet werden. Dies kann genutzt werden, um die Komplexität des Algorithmus zu verringern, indem die Berechnungen des Vorwärts-Algorithmus nur an den tatsächlich benötigten Punkten im Trellis-Diagramm ausgeführt werden. Dazu müssen die $\alpha(i,s,t)$-Werte in Gleichung (4.6) nur für

$$1 \leq i \leq N,\ 2 \leq t \leq T - S \tag{4.9}$$

berechnet werden. Entsprechend müssen in Gleichung (4.7) die Werte nur für

$$1 \leq i \leq N,\ 2 \leq t \leq T \text{ und } \max\left\{1, t-(T-S)\right\} \leq s \leq \min\left\{S, t\right\} \tag{4.10}$$

berechnet werden. Die in dieser Arbeit entwickelte Vorwärtsberechnung ist noch einmal in Algorithmus 3 auf der nächsten Seite zusammengefasst. Das Verfahren reduziert die Komplexität der Berechnung deutlich, wie im nächsten Abschnitt gezeigt wird. Dabei ist die Vorgehensweise mathematisch exakt und führt zu der gleichen Wahrscheinlichkeit $p(\vec{x},\, \vec{y}\,|\,\lambda_k)$, wie die ursprüngliche Vorwärtsberechnung.

Die selbe Technik kann mit der gleichen Komplexitätsreduktion analog auch für die Viterbi-Dekodierung (Abschnitt 4.2.7) und die Rückwärtsberechnung $\beta(i,s,t)$ (Abschnitt 4.2.8) angewendet werden. Damit wird durch das hier entwickelte Verfahren die Komplexität sowohl im Training als auch in der Erkennungsphase verringert. Weiter ist es mit der optimierten Berechnung möglich, eine Skalierung anzuwenden (Abschnitt 4.2.6), die mit der ursprünglichen Berechnung nicht möglich ist.

4.2.5 Komplexitätsbetrachtung

Die Zeitkomplexität[2] des ursprünglichen AHMM Vorwärts-Algorithmus [17] ist

$$\mathcal{O}(N^2 S T), \tag{4.11}$$

wobei N wiederum die Anzahl der Zustände in einem ergodischen Modell ist und T und S die Längen der Merkmalströme $\vec{x}$ und $\vec{y}$. Dies kann bei zu langen Datenströmen schnell unberechenbar werden. In [17] wird daher vorgeschlagen, die zeitliche Verzerrung zwischen $\vec{x}$ und $\vec{y}$ so einzugrenzen, dass $|t - \frac{T}{S}| < k$. Dabei gibt die

[2]Prinzipiell ist die Platzkomplexität des AHMM identisch zur Zeitkomplexität. Wird jedoch keine Segmentierung benötigt, muss nicht das gesamte Trellis-Diagramm, sondern nur zwei aufeinanderfolgende (τ_t, q_t)-Ebenen gespeichert werden, so dass sich die Platzkomplexität – bei gleichbleibender Zeitkomplexität – auf $\mathcal{O}(2NS)$ verringert.

Algorithmus 3 AHMM Vereinfachte Vorwärtsberechnung

Benötigt: Modellparameter λ und unbekannte Merkmalsequenz $\vec{o} = \{\vec{x}, \vec{y}\}$
Stellt sicher: Berechnung von $p(\vec{x}, \vec{y} \,|\, \lambda)$ in $\mathcal{O}\big(N^2[TS - S^2 + T]\big)$

function VORWÄRTSBERECHNUNG($\vec{x}, \vec{y}, \lambda$)
  for $1 \leq i \leq N$ **do** ▷ Initialisierung
    $\alpha(i, 0, 1) = \pi_i \, (1 - \epsilon_i) \, p(\vec{x}_1 \,|\, q_1 = i)$
    $\alpha(i, 1, 1) = \pi_i \, \epsilon_i \, p(\vec{x}_1, \vec{y}_1 \,|\, q_1 = i)$
  end for
  for $2 \leq t \leq T - S$ **do** ▷ Induktion, solange $s = 0$
    for $1 \leq i \leq N$ **do**
      $\alpha(i, 0, t) = (1 - \epsilon_i) \, p(\vec{x}_t \,|\, q_t = i) \; \sum_{j=1}^{N} a_{ji} \, \alpha(j, 0, t - 1)$
    end for
  end for
  for $2 \leq t \leq T$ **do** ▷ Induktion, für $s > 0$
    for $\max\big\{1, t - (T - S)\big\} \leq s \leq \min\big\{S, t\big\}$ **do**
      for $1 \leq i \leq N$ **do**
        $\alpha(i, s, t) = \; \ldots$
          $\epsilon_i \, p(\vec{x}_t, \vec{y}_s \,|\, q_t = i) \; \sum_{j=1}^{N} a_{ji} \, \alpha(j, s - 1, t - 1) + \; \ldots$
          $(1 - \epsilon_i) \, p(\vec{x}_t \,|\, q_t = i) \sum_{j=1}^{N} a_{ji} \, \alpha(j, s, t - 1)$
      end for
    end for
  end for
  return $p(\vec{x}, \vec{y} \,|\, \lambda) = \; \sum_{i=1}^{N} \alpha(i, S, T)$ ▷ Terminierung
end function

Konstante k an, wie weit die beiden Ströme maximal zeitlich zueinander verzerrt werden dürfen. Durch diese Methode wird die Zeitkomplexität auf $\mathcal{O}(kN^2T)$ und damit auf die k-fache HMM-Komplexität verringert. Auf der anderen Seite bedeutet dieses Vorgehen eine starke Eingrenzung des Suchbereichs im (t, τ_t)-Raum. Unter Umständen werden so gute oder optimale zeitliche Verzerrungen von vornherein ausgeschlossen. Für audio-visuelle Fusionsprobleme ist dieses Vorgehen in einigen Fällen noch anwendbar, da die Asynchronität zwischen dem audio- und dem visuellen Kanal häufig nur wenige Zeitschlitze beträgt. Daher schadet in diesem Fall die Eingrenzung der erlaubten Verzerrung nicht. Wie in Abschnitt 4.1 gezeigt wurde, kann für andere Probleme, wie z. B. die multimodale Fusion von Sprache und Gesten, die Asynchronität zwischen den Merkmalströmen deutlich höher liegen und von Beispiel zu Beispiel signifikant variieren. Eine vorher festgelegte maximale Verzerrung ist daher für solche Probleme nicht bestimmbar. Damit limitiert das Einschränken des Verzerrungsbereichs die Vorteile des AHMM gegenüber anderen multimodalen Konzepten deutlich.

Mit der verbesserten Berechnung, die in dieser Arbeit vorgeschlagen wird, werden in der Vorwärts- und Rückwärtsberechnung in dem Trellis-Diagramm nur Punkte berechnet, die tatsächlich wiederverwendet werden. Dadurch werden unnötige Berechnungen vermieden. Trotzdem ist das Ergebnis der Berechnung mathematisch exakt, weil die vernachlässigten Punkte niemals zu einem gültigen Pfad durch das Trellis-Diagramm führen können. Durch Ausnutzen dieser Eigenschaft kann die Zeitkomplexität der Vorwärts- und Rückwärtsberechnung (und analog der Viterbi-Dekodierung) auf

$$\mathcal{O}(N^2[TS - S^2 + T]) \tag{4.12}$$

reduziert werden. Abhängig von der Länge der Merkmalsströme $\vec{x}$ und $\vec{y}$ ist die Reduktion der benötigten Operationen im Vergleich zum ursprünglichen Algorithmus zwischen 0 (für den Fall, dass $S = 1$) und N^2T^2 (für den Fall, dass $T = S$, d. h. gleich lange Merkmalsströme). Im schlechtesten Fall ist der Strom $\vec{x}$ doppelt so lang wie der Strom $\vec{y}$ (d. h. $T = 2S$). Für die optimierte Berechnung werden dann $N^2(\frac{T^2}{4} + T)$ Rechenoperationen benötigt. Auch in diesem Fall benötigt das optimierte Verfahren jedoch $N^2\frac{T^2}{4}$ weniger Berechnungen als der ursprüngliche Algorithmus und ist damit signifikant schneller.

Trotz dieser deutlichen Komplexitätsreduktion wird das exakte Ergebnis erzielt und keine Limitierung an die mögliche Asynchronität zwischen den Merkmalsströmen gesetzt. Alle Vorteile des AHMM werden daher ausgenutzt und es kann auf eine größere Anzahl von Problemen angewendet werden.

4.2.6 Skalierung

Abgesehen von der im vorherigen Abschnitt gezeigten deutlichen Reduktion der Komplexität, erlauben die neuen Bereichsgrenzen für die Berechnung der Gleichungen (4.6) - (4.7) auch eine einfache und exakte Skalierungsprozedur. Da es sich in der Vorwärtsberechnung um Wahrscheinlichkeiten handelt, liegen die Werte zwischen $0 \leq \alpha(i, s, t) \leq 1, \forall i, s, t$. Durch den Induktionsschritt konvergieren diese Werte für längere Merkmalsequenzen gegen 0. Für gewöhnliche HMM wird daher nach jedem Induktionsschritt eine Skalierung durchgeführt [126]. Dadurch können HMM für beliebig lange Merkmalsequenzen ohne numerische Probleme berechnet werden. Die hier eingeführten Bereichsgrenzen erlauben eine analoge Skalierung für das AHMM, die mit dem ursprünglichen Algorithmus nicht möglich ist.

In Analogie zur Skalierungsprozedur für HMM [126], müssen, um im AHMM skalierte Werte zu erhalten, nach jedem Zeitschritt t alle $\alpha(i, s, t)$-Werte innerhalb der (q_t, τ_t)-Ebene aufaddiert werden. Im ursprünglichen Algorithmus sind einige dieser $\alpha(i, s, t) > 0$, obwohl es von oder zu ihnen keinen gültigen Pfad durch das Trellis-Diagramm gibt. Werden diese Werte innerhalb der Ebene trotzdem mit in den Skalierungsfaktor aufsummiert, wird die durch die Vorwärtsprozedur erhaltene Wahrscheinlichkeit effektiv mit Wahrscheinlichkeiten multipliziert, die eigentlich kei-

nen Einfluss auf die tatsächliche Observierungswahrscheinlichkeit $p(\vec{x}, \vec{y} \,|\, \lambda)$ haben. Deshalb wird die nach Skalierung gewonnene Wahrscheinlichkeit

$$p^0(\vec{x}, \vec{y} \,|\, \lambda) < p(\vec{x}, \vec{y} \,|\, \lambda) \tag{4.13}$$

immer zu klein im Vergleich zur tatsächlichen Wahrscheinlichkeit sein. Dies kann nur verhindert werden, wenn ausschließlich gültige Pfadpunkte innerhalb der Ebene aufsummiert werden. Allerdings entspräche dies genau den Bereichsgrenzen der in dieser Arbeit vorgestellten Vorwärtsprozedur. Es ist anzumerken, dass der Fehler durch das Berechnen von $p^0(\vec{x}, \vec{y} \,|\, \lambda)$ statt von $p(\vec{x}, \vec{y} \,|\, \lambda)$ keinen praktischen Einfluss auf die Erkennungsleistung des AHMM hat, da der Fehler in allen Modellen gemacht wird. Die Berechnung ist jedoch nicht mathematisch korrekt, und es gibt keine Möglichkeit, die tatsächliche Datenwahrscheinlichkeit zu berechnen.

Dies ist mit dem hier vorgestellten Vorwärts-Algorithmus trotz Skalierung möglich. Hier werden nur $\alpha(i, s, t)$-Werte berechnet, die auf einem gültigen Pfad durch das Trellis-Diagramm zur Terminierung beitragen. Die Skalierung erfolgt dann analog zur HMM-Skalierung: Für jeden Zeitschritt t werden die $\alpha(i, s, t)$-Werte für die in Gleichung (4.9) und (4.10) gegebenen Bereichsgrenzen berechnet. Wird die höchste berechnete zeitliche Verzerrung im Zeitschritt t dabei mit s_t bezeichnet, kann der Skalierungskoeffizient gemäß

$$c(t) = \frac{1}{\sum_{s=0}^{s_t} \sum_{i=1}^{N} \alpha(i, s, t)} \tag{4.14}$$

berechnet werden. Dann wird eine Skalierung analog zu HMM [126] angewendet: Nach jedem Zeitschritt t werden alle $\alpha(i, s, t)$ mittels

$$\widetilde{\alpha}(i, s, t) = c(t)\ \alpha(i, s, t) \tag{4.15}$$

skaliert. Im nächsten Zeitschritt werden dann für die Induktion in Algorithmus 3 nicht mehr die $\alpha(i, s, t-1)$, sondern die skalierten $\widetilde{\alpha}(i, s, t-1)$ verwendet. Durch Aufsummieren aller logarithmierten Skalierungskoeffizienten

$$\log p(\vec{x}, \vec{y} \,|\, \lambda) = \sum_{t=1}^{T} \log c(t) \tag{4.16}$$

erhält man schließlich die exakte logarithmierte Datenwahrscheinlichkeit[3]. Mit dem hier vorgeschlagenen Vorwärts-Algorithmus ist es daher möglich, eine exakte Skalierung für das AHMM anzuwenden, und dadurch alle Werte in einem numerisch handhabbaren Bereich zu begrenzen. Trotzdem kann auf diese Weise die tatsächlich Datenwahrscheinlichkeit bestimmt werden. Analog wird diese Skalierung auch für den Rückwärtspfad $\beta(i, s, t)$ angewendet.

[3]Im Allgemeinen wird diese nicht mehr in die unlogarithmierte Wahrscheinlichkeit zurückgerechnet, da es sich um sehr kleine Werte handelt.

4.2.7 Viterbi-Dekodierung

Von dem Vorwärts-Algorithmus kann ein Viterbi-Algorithmus [151] abgeleitet werden [17]. Je nach Ausführung erhält man dadurch entweder die beste Zustandssequenz, die beste zeitliche Anordnung von $\vec{x}$ und $\vec{y}$ oder beides. Für die Viterbi-Dekodierung werden alle oder Teile der Additionen in Gleichung (4.6) und (4.7), je nach gewünschtem Verhalten, durch Maximierungen ersetzt. Wird sowohl die beste zeitliche Anordnung als auch die beste Zustandssequenz gesucht, wird die Viterbi-Vorwärtsvariable definiert als

$$\delta(i,s,t) = \max_{\substack{q_1,\ldots,q_{t-1}\\ \tau_1,\ldots,\tau_{t-1}}} p(q_1,q_2,\ldots,q_t=i,\,\tau_1,\tau_2,\ldots,\tau_t=s,\,\vec{x}_1,\ldots,\vec{x}_t,\,\vec{y}_1,\ldots,\vec{y}_s). \tag{4.17}$$

Damit ergibt sich für den Induktionsschritt aus Gleichung (4.7):

$$\delta(i,s,t) = \max \left\{ \begin{array}{l} \epsilon_i\, p(\vec{x}_t,\vec{y}_s \,|\, q_t=i) \max\limits_{1\le j\le N} \left\{ a_{ji}\, \delta(j,s-1,t-1) \right\} \\ (1-\epsilon_i)\, p(\vec{x}_t \,|\, q_t=i) \max\limits_{1\le j\le N} \left\{ a_{ji}\, \delta(j,s,t-1) \right\} \end{array} \right\}. \tag{4.18}$$

Alle anderen Schritte aus dem Vorwärts-Algorithmus werden analog durch Maximierungen anstelle der Additionen angepasst. Durch Einführen von zwei Rückverfolgungsvektoren ψ_t für die Zustände und Σ_t für die zeitliche Anordnung kann ausgehend von $\delta(q_T,S,T)$ rückwärts die beste Zustandssequenz (wie in Abbildung 4.4, rechts in der unteren, weißen Ebene gezeigt) und die beste zeitliche Verzerrung von $\vec{x}$ und $\vec{y}$ (wie in Abbildung 4.4, rechts in der hinteren, dunkelgrauen Eben gezeigt) erhalten werden. Wird nur die beste Zustandssequenz bei Beachtung aller möglichen zeitlichen Verzerrungen gesucht, wird die Viterbi-Vorwärtsvariable als

$$\delta(i,s,t) = \max_{q_1,\ldots,q_{t-1}} p(q_1,q_2,\ldots,q_t=i,\,\tau_1,\tau_2,\ldots,\tau_t=s,\,\vec{x}_1,\ldots,\vec{x}_t,\,\vec{y}_1,\ldots,\vec{y}_s) \tag{4.19}$$

definiert und in Gleichung (4.7) nur die inneren Additionen durch Maximierungen ersetzt, aber die äußere Summe beibehalten. Wird dagegen nur die beste zeitliche Anordnung, nicht aber die beste Zustandssequenz gesucht, wird die Viterbi-Vorwärtsvariable analog als

$$\delta(i,s,t) = \max_{\tau_1,\ldots,\tau_{t-1}} p(q_1,q_2,\ldots,q_t=i,\,\tau_1,\tau_2,\ldots,\tau_t=s,\,\vec{x}_1,\ldots,\vec{x}_t,\,\vec{y}_1,\ldots,\vec{y}_s) \tag{4.20}$$

definiert. Dann wird in Gleichung (4.7) nur die erste Addition durch eine Maximierung ersetzt.

Durch diese Art der Viterbi-Dekodierung kann das AHMM also nicht nur zur Klassifikation von Merkmalsequenzen genutzt werden, sondern auch um die beste zeitliche Anordnung zweier Merkmalströme zu erhalten. Durch die Anwendung der optimierten Berechnung gemäß Gleichung (4.9) und (4.10) wird auch für die Viterbi-Dekodierung die Zeitkomplexitätsreduktion wie in Abschnitt 4.2.5 erreicht.

4.2.8 EM-Training

Hilfsvariablen

Um die Parameter λ eines AHMM automatisch aus Daten zu lernen, kann ein Expectation-Maximisation (EM) Trainingsalgorithmus [10, 25, 50, 108] entworfen werden [17]. Dazu werden neben der Vorwärtsvariablen $\alpha(i, s, t)$ noch drei weitere Hilfsvariablen benötigt. Vergleichbar zu normalen HMM wird eine Rückwärtsvariable

$$\beta(i, s, t) = p(\vec{x}_{t+1}, \ldots, \vec{x}_T, \vec{y}_{s+1}, \ldots, \vec{y}_S \,|\, q_t = i, \tau_t = s) \tag{4.21}$$

definiert, die analog zu Algorithmus 3 effizient durch dynamische Programmierung berechnet werden kann. Auch hier können wieder die verbesserten Bereichsgrenzen aus Gleichung (4.9) und (4.10) angewendet werden, um die Komplexität zu verringern.

Zur Berechnung der Emissionswahrscheinlichkeiten im Schätzschritt des EM-Algorithmus werden die beiden Teile der Vorwärtsberechnung $\alpha(i, s, t)$ benötigt, bei denen ein Zustand jeweils nur ein bzw. zwei Symbole emittiert. Es wird daher eine Hilfsvariable

$$\alpha^0(i, s, t) = p(q_t = i, \tau_t = s, \vec{x}_t, \vec{y}_s \,|\, \tau_{t-1} = s) \tag{4.22}$$

als der Teil von $\alpha(i, s, t)$ definiert, bei dem der Zustand $q_t = i$ zum Zeitpunkt t nur ein Symbol von $\vec{x}$ emittiert. Dabei handelt es sich also um den Teil von $\alpha(i, s, t)$, der durch Gleichung (4.4), (4.6), sowie dem zweiten Summanden von (4.7) berechnet wird. Analog wird

$$\alpha^1(i, s, t) = p(q_t = i, \tau_t = s, \vec{x}_t, \vec{y}_s \,|\, \tau_{t-1} = s - 1) \tag{4.23}$$

als der Teil von $\alpha(i, s, t)$ definiert, bei dem der Zustand $q_t = i$ zum Zeitpunkt t ein Symbolpaar von $\vec{x}$ und $\vec{y}$ emittiert. Es handelt sich also um den Teil von $\alpha(i, s, t)$, der durch Gleichung (4.5) sowie dem ersten Summanden von (4.7) berechnet wird. Tatsächlich werden in Implementierungen von AHMM zuerst nur $\alpha^0(i, s, t)$ und $\alpha^1(i, s, t)$ berechnet und dann aufsummiert, um auch $\alpha(i, s, t)$ zu erhalten. Dadurch erhöht sich der Rechenaufwand für diese beiden Hilfsvariablen in der Vorwärtsberechnung der Trainings Phase nicht.

Q-Funktion

Im Folgenden bezeichne $\vec{q} = [q_1, \ldots, q_T]$ eine bestimmte (mit $\vec{x}$ synchrone) Zustandssequenz der Länge T – also genau einen Pfad durch die in Abbildung 4.4 gezeigte weiße Ebene. Weiterhin sei $\vec{\tau} = [\tau_0, \ldots, \tau_T]$, mit $\tau_0 = 0$, eine bestimmte Synchronisation zwischen $\vec{x}$ und $\vec{y}$ – also genau ein Pfad durch die in Abbildung 4.4 gezeigte dunkelgraue Ebene. Um die Q-Hilfsfunktion für den EM-Algorithmus aufzustellen, muss zuerst die Auftrittswahrscheinlichkeit $p(\vec{x}, \vec{y}, \vec{q}, \vec{\tau} \,|\, \lambda)$ einer bestimmten Zustandssequenz und einer bestimmten zeitlichen Anordnung für die Merkmalströme $\vec{x}$ und $\vec{y}$,

bei gegebenen AHMM Parametern λ bestimmt werden. Mit den in Abschnitt 4.2.3 eingeführten Modellparametern berechnet sich diese Wahrscheinlichkeit als

$$p(\vec{x},\vec{y},\vec{q},\vec{\tau}\,|\,\lambda) = \\ \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1},q_t} \prod_{t=1}^{T} \Big(\delta[\tau_t-\tau_{t-1}]\;(1-\epsilon_{q_t})\;p(\vec{x}_t\,|\,q_t) \;+\; \delta[1-\tau_t+\tau_{t-1}]\;\epsilon_{q_t}\;p(\vec{x}_t,\vec{y}_{\tau_t}\,|\,q_t)\Big), \tag{4.24}$$

wobei $\delta[\cdot]$ die diskrete Dirac-Funktion bezeichne. Die beiden Dirac-Funktionen legen damit anhand der Synchronisation $\vec{\tau}$ fest, ob im Zustand q_t ein oder zwei Symbole emittiert werden. Wobei immer nur genau eine der beiden Dirac-Funktionen den Wert eins annimmt. Wird diese Wahrscheinlichkeit logarithmiert, stellt ihr Erwartungswert bezüglich gegebener Beobachtung $\vec{x},\vec{y}$ und Modellparametern λ' die Q-Hilfsfunktion für den EM-Algorithmus dar:

$$\begin{aligned} Q(\lambda,\lambda') &= \Big\langle \log p(\vec{x},\vec{y},\vec{q},\vec{\tau}\,|\,\lambda)\Big\rangle_{\vec{x},\vec{y},\lambda'} \\ &= \sum_{\vec{q}\in\mathcal{Q}}\sum_{\vec{\tau}\in\Delta} \log p(\vec{x},\vec{y},\vec{q},\vec{\tau}\,|\,\lambda)\;p(\vec{q},\vec{\tau}\,|\,\vec{x},\vec{y},\lambda') \\ &= \sum_{\vec{q}\in\mathcal{Q}}\sum_{\vec{\tau}\in\Delta} \log p(\vec{x},\vec{y},\vec{q},\vec{\tau}\,|\,\lambda)\;\frac{p(\vec{x},\vec{y},\vec{q},\vec{\tau}\,|\,\lambda')}{p(\vec{x},\vec{y}\,|\,\lambda')}, \end{aligned} \tag{4.25}$$

wobei $\mathcal{Q}$ die Menge aller möglichen Zustandssequenzen des Modells (also die gesamte weiße Ebene in Abbildung 4.4) und Δ die Menge aller möglichen zeitlichen Anordnungen von $\vec{x}$ und $\vec{y}$ (also die gesamte dunkelgraue Ebene in Abbildung 4.4) bezeichne. Einsetzen von Gleichung (4.24) in (4.25) mit Umformen führt zu

$$Q(\lambda,\lambda') = \\ \Bigg(\sum_{\vec{q}\in Q}\sum_{\vec{\tau}\in\Delta}\sum_{t=1}^{T} \log\Big(\delta[\tau_t-\tau_{t-1}]\;(1-\epsilon_{q_t})\;p(\vec{x}_t\,|\,q_t) + \delta[1-\tau_t+\tau_{t-1}]\;\epsilon_{q_t}\;p(\vec{x}_t,\vec{y}_{\tau_t}\,|\,q_t)\Big) \\ + \sum_{\vec{q}\in Q}\sum_{\vec{\tau}\in\Delta}\log(\pi_{q_1}) + \sum_{\vec{q}\in Q}\sum_{\vec{\tau}\in\Delta}\sum_{t=2}^{T}\log(a_{q_{t-1},q_t})\Bigg)\;\frac{p(\vec{x},\vec{y},\vec{q},\vec{\tau}\,|\,\lambda')}{p(\vec{x},\vec{y}\,|\,\lambda')}, \tag{4.26}$$

in der die Modellparameter a_{ij}, π_i, ϵ_i sowie die Emissionswahrscheinlichkeiten voneinander separiert sind und daher getrennt voneinander optimiert werden können. Durch die beiden Dirac-Funktionen, von der immer nur genau eine den Wert eins annehmen kann, können so auch die Emissionswahrscheinlichkeiten voneinander getrennt optimiert werden.

Schätzformeln

Durch Einführen von Lagrange-Multiplikatoren und getrenntem Ableiten der Q-Funktion (4.26) nach a_{ij}, π_i und ϵ_i erhält man

$$\widehat{\pi}_i = \frac{\sum_{s=0}^{1} p(\vec{x}, \vec{y}, q_1 = i, \tau_1 = s \,|\, \lambda')}{\sum_{s=0}^{1} p(\vec{x}, \vec{y}, \tau_1 = s \,|\, \lambda')}, \tag{4.27}$$

$$\widehat{a}_{ij} = \frac{\sum_{t=2}^{T} \sum_{s=1}^{S} p(\vec{x}, \vec{y}, q_t = j \,|\, q_{t-1} = i, \tau_t = s, \lambda')}{\sum_{t=2}^{T} \sum_{s=1}^{S} p(\vec{x}, \vec{y}, q_{t-1} = i \,|\, \tau_t = s, \lambda')}, \tag{4.28}$$

$$\widehat{\epsilon}_i = \frac{\sum_{t=1}^{T} \sum_{s=1}^{S} p(\vec{x}, \vec{y}, \tau_t = s, \tau_{t-1} = s-1 \,|\, q_t = i, \lambda')}{\sum_{t=1}^{T} \sum_{s=0}^{S} p(\vec{x}, \vec{y} \,|\, q_t = i, \lambda')}. \tag{4.29}$$

Bei diskreter Modellierung der Ausgabewahrscheinlichkeiten $b_i(v) = p(\vec{x}_t = v \,|\, q_t = i)$ und $b_i(v, w) = p(\vec{x}_t = v, \vec{y}_{\tau_t} = w \,|\, q_t = i)$ ergibt sich – wiederum nach Einführung von Lagrange-Multiplikatoren und Ableiten von Gleichung (4.26):

$$\widehat{b}_i(v) = \frac{\sum_{t=1}^{T} \sum_{s=0}^{S} \delta[x_t = v] \; p(\vec{x}, \vec{y}, \tau_t = s, \tau_{t-1} = s \,|\, q_t = i, \lambda')}{\sum_{t=1}^{T} \sum_{s=0}^{S} p(\vec{x}, \vec{y}, \tau_t = s, \tau_{t-1} = s \,|\, q_t = i, \lambda')}, \tag{4.30}$$

$$\widehat{b}_i(v, w) = \frac{\sum_{t=1}^{T} \sum_{s=0}^{S} \delta[x_t = v] \; \delta[y_s = w] \; p(\vec{x}, \vec{y}, \tau_t = s, \tau_{t-1} = s \,|\, q_t = i, \lambda')}{\sum_{t=1}^{T} \sum_{s=0}^{S} p(\vec{x}, \vec{y}, \tau_t = s, \tau_{t-1} = s \,|\, q_t = i, \lambda')}. \tag{4.31}$$

Diese Gleichungen (4.27) - (4.31) können aus den Hilfsvariablen $\alpha(i, s, t)$, $\alpha^0(i, s, t)$, $\alpha^1(i, s, t)$ und $\beta(i, s, t)$ als iterative Lernvorschriften (*maximisation*) für die AHMM Parameter, basierend auf der gegenwärtigen Schätzung (*estimation*), ausgedrückt werden. Dann ergibt sich als Lernvorschrift für die **Einsprungswahrscheinlichkeiten**:

$$\widehat{\pi}_i = \frac{\sum_{s=0}^{1} \alpha(i, s, 1) \; \beta(i, s, 1)}{\sum_{i=1}^{N} \sum_{s=0}^{1} \alpha(i, s, 1) \; \beta(i, s, 1)}. \tag{4.32}$$

Mit der Kurznotation

$$f^0(i, s, t) = \alpha(i, s, t-1) \; a_{ij} \; (1 - \epsilon_j) \; p(\vec{x}_t \,|\, q_t = j) \; \beta(j, s, t) \tag{4.33}$$

$$\text{und } f^1(i, s, t) = \alpha(i, s-1, t-1) \; a_{ij} \; \epsilon_j \; p(\vec{x}_t, \vec{y}_s \,|\, q_t = j) \; \beta(j, s, t) \tag{4.34}$$

ergibt sich als Lernvorschrift für die **Übergangswahrscheinlichkeiten**:

$$\widehat{a}_{ij} = \frac{\sum_{t=2}^{T} \big(\sum_{s=1}^{S} f^0(i, s, t) + \sum_{s=0}^{S} f^1(i, s, t) \big)}{\sum_{n=1}^{N} \sum_{t=2}^{T} \big(\sum_{s=0}^{S} f^0(n, s, t) + \sum_{s=1}^{S} f^1(n, s, t) \big)}. \tag{4.35}$$

Für die **Emissionswahrscheinlichkeit eines Symbolpaares** ergibt sich:

$$\widehat{\epsilon}_i = \frac{\sum_{s=0}^{S} \sum_{t=1}^{T} \alpha^1(i, s, t) \; \beta(i, s, t)}{\sum_{s=0}^{S} \sum_{t=1}^{T} \alpha(i, s, t) \; \beta(i, s, t)}. \tag{4.36}$$

Bei diskreter Modellierung der **Ausgabewahrscheinlichkeiten** $b_i(v) = p(\vec{x}_t = v \,|\, q_t = i)$ und $b_i(v,w) = p(\vec{x}_t = v, \vec{y}_{\tau_t} = w \,|\, q_t = i)$ ergeben sich die Lernvorschriften:

$$\widehat{b}_i(v) = \frac{\sum_{t=1}^{T}\sum_{s=0}^{S} \alpha^0(i,s,t)\ \delta[\vec{x}_t - v]\ \beta(i,s,t)}{\sum_{t=1}^{T}\sum_{s=0}^{S} \alpha^0(i,s,t)\ \beta(i,s,t)}, \tag{4.37}$$

$$\widehat{b}_i(v,w) = \frac{\sum_{t=1}^{T}\sum_{s=0}^{S} \alpha^1(i,s,t)\ \delta[\vec{x}_t - v]\ \delta[\vec{y}_s - w]\ \beta(i,s,t)}{\sum_{t=1}^{T}\sum_{s=0}^{S} \alpha^1(i,s,t)\ \beta(i,s,t)}. \tag{4.38}$$

Die Lernvorschriften für kontinuierliche Ausgabewahrscheinlichkeiten bei AHMM sind hier nicht hergeleitet. Sie hängen von der Modellierung (z. B. Mixturen von Gaußkurven), sowie der Art der Modellierung von Symbolpaaren (z. B. als Verbund- oder bedingter Wahrscheinlichkeit) ab. Ausgehend von der Q-Funktion (4.26) ist die Herleitung für diese Lernformeln im kontinuierlichen Fall analog zu der bei normalen HMM. Eine ausführliche EM-Herleitung für Mixturen von Gaußkurven findet sich z. B. in [25]; eine frequenzbasierte Herleitung z. B. in [126].

4.3 Experimente zum Zeitverhalten des AHMM

Im Rahmen dieser Arbeit wird die theoretisch festgestellte Reduktion an benötigten Rechenoperationen auch experimentell überprüft. Sowohl der ursprüngliche als auch der verbesserte Vorwärts-Algorithmus wurden dazu implementiert[4]. Anhand von realen Merkmalsequenzen verschiedener Längen wird dann die Anzahl benötigter Rechenoperationen zur Dekodierung für beide Verfahren gemessen. Abbildung 4.6 zeigt einen beispielhaften Messverlauf der benötigten Rechenoperationen. In diesem Beispiel hat das AHMM fünf ergodische Zustände und diskrete Observierungswahrscheinlichkeiten. Die Länge des $\vec{x}$-Stromes ist auf $T = 250$ festgelegt. Die Länge des $\vec{y}$-Stromes wird zwischen $1 \leq S \leq 250$ variiert. Der Messverlauf zeigt, dass die benötigten Rechenoperationen für den ursprünglichen Algorithmus linear mit dem Verhältnis der Längen $\frac{S}{T}$ ansteigen und für $S = T$ das Maximum erreichen. Die Kurve der benötigten Rechenoperationen des verbesserten Algorithmus verläuft dagegen parabelförmig und hat – wie theoretisch zu erwarten – bei $S = \frac{T}{2}$ ihr Maximum.

Dieser Messverlauf ist exemplarisch für alle vorgenommenen Messungen bei verschiedenen Sequenzlängen, Zuständen und lineare, links-rechts- oder ergodischen Modellen. In allen Experimenten kann die theoretische Zeitkomplexität im Wesentlichen verifiziert werden. Allerdings gibt es bei den Messungen eine minimale Abweichung vom zu erwartenden Ergebnis: Theoretisch sollte die Anzahl der benötigten Rechenoperationen für den hier vorgestellten Algorithmus immer kleiner oder höchstens gleich der Anzahl der Operationen des ursprünglichen Algorithmus sein. Für sehr

[4] S. Bengio stellte später seine auf dem ursprünglichen Algorithmus [17] beruhende AHMM Implementierung zur Verfügung. Die hier durchgeführten Experimente konnten damit noch einmal bestätigt werden.

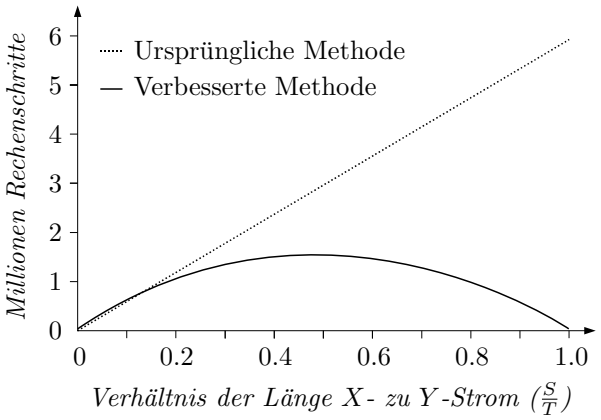

Abbildung 4.6: Vergleich der benötigten Rechenoperationen für die Dekodierung einer Beobachtung mit einem AHMM mit fünf ergodischen Zuständen. Die Länge des $\vec{x}$-Stromes ist konstant $T = 250$, die Länge des $\vec{y}$-Stromes variiert zwischen $1 \leq S \leq 250$.

kurze $\vec{y}$-Ströme, bei denen $\frac{S}{T} < 0{,}1$ ist, führt der erweiterte Vorwärts-Algorithmus aber tatsächlich zu einer minimal höheren Anzahl an Rechenschritten als der ursprüngliche Algorithmus. In dem hier vorgestellten Algorithmus müssen für jeden Zeitschritt t die Bereichsgrenzen gemäß (4.10) berechnet werden (Algorithmus 3, Zeile 12). Dies verursacht eine fixe, kleine Anzahl von zusätzlichen Berechnungen, die nicht in die Zeitkomplexität miteingerechnet wurden. Diese zusätzlichen Berechnungen haben jedoch nur einen Einfluss, wenn die Dekodierungszeit durch den kurzen $\vec{y}$-Strom sowieso sehr kurz ist. Die Extraberechungen haben daher keinen praktischen Einfluss und können vernachlässigt werden.

4.4 Experimente zur bimodalen Fusion

In einem zweiten Experiment wird das optimierte AHMM verwendet, um bimodale Sprach- und Gestenkommandos zu fusionieren. Hier kann die Asynchronität zwischen den Datenströmen sehr groß werden, da die Nutzer die Freiheit haben die Modalitäten sowohl überlappend als auch sequentiell zu nutzen (siehe auch Abschnitt 4.1). Durch die erreichte Komplexitätsreduktion kann das AHMM auch auf solche Probleme mit sehr großen Asynchronitäten angewendet werden.

4.4.1 Daten und Merkmale

Die Daten zur bimodalen Fusion von Sprach- und Gestenkommandos enthalten elf mögliche Kommandowörter und zehn mögliche Mausgesten. Diese können in beliebiger zeitlicher Kombination verwendet werden, um eines von 25 möglichen System-

kommandos zu erzeugen. Jedes Kommando kann dabei mit verschiedenen Kombinationen von Sprache und Gestik erzielt werden. Jedoch ist für ein gültiges Kommando stets die Information von beiden Modalitäten erforderlich.

Die Sprach-Gesten-Datenbank enthält insgesamt 1872 bimodale Kommandos. Die Beispiele in der Datenbank decken alle möglichen Benutzerinteraktionsmuster ab [119], da die Kommandos entweder

synchron – d. h. Sprache und Geste überlappen sich vollständig, müssen jedoch nicht zur gleichen Zeit synchron starten bzw. enden –,

teilweise überlappend – d. h. Sprache und Geste überlappen sich zum Teil, aber nicht vollständig, wobei das Kommando sowohl mit der Sprache als auch mit der Geste beginnen kann – oder

sequentiell – d. h. Sprache und Geste überlappen sich gar nicht, sondern erfolgen nacheinander, wobei das Kommando sowohl mit der Sprache als auch mit der Geste beginnen kann –

sind. Die Datenbank ist aufgeteilt in 1240 Beispiele für das Trainings- und 632 Beispiele für das Test-Set. Dabei enthält das Test-Set sowohl Beispiele von aus dem Test-Set bekannten als auch von unbekannten Personen.

Aus der Sprachmodalität werden alle 10ms innerhalb eines Hamming-Fensters mit 20ms Breite zwölf Mel-Frequenz-Cepstrum Koeffizienten (*Mel frequency cepstral coefficients, MFCC*) [60, 132, 166], die Energie sowie die erste und zweite Ableitung der Koeffizienten und der Energie berechnet. Diese 39-dimensionalen MFCC-Merkmalsvektoren werden dann mit einem k-Means-Vektorquantisierer auf 100 Symbole diskretisiert. Die Erkennungsrate für die elf isolierten Wörter aus der Sprachmodalität mit einem diskreten, ergodischen HMM mit 15 Zuständen liegt bei ca. 92,5%.

Für die Gestenmodalität wird bei jeder Mausbewegung bzw. bei jeder Richtungsänderung eine von acht möglichen Richtungen (oben, unten, links, rechts und die vier Diagonalrichtungen) gespeichert, woraus sich aus den Gestenmerkmalen ein Wegediagramm ohne Dynamikinformation ergibt. Die Erkennungsrate für die zehn isolierten Gesten mit einem diskreten, ergodischen HMM mit 20 Zuständen liegt bei ca. 90%.

4.4.2 Fusion mit dem AHMM und mit Vergleichsmodellen

Das AHMM wird auf dieses Sprach-Gesten-Fusionsproblem angewendet. Dazu wird jeweils ein AHMM pro Systemkommando mit den bimodalen Daten trainiert. In der Erkennungsphase wird die in Abschnitt 4.2.4 beschriebene, optimierte Vorwärtsberechnung angewendet, um jedes Test-Beispiel einer der 25 Systemkommandoklassen zuzuordnen.

Zum Vergleich werden auch HMM mit früher und später Fusion evaluiert. Für die frühe Merkmalfusion werden die Sampleraten der beiden Modalitäten durch Hochsamplen des Gestenstromes aneinander angepasst. Dann werden beide Modalitäten zu einem größeren Merkmalvektor zusammengefügt. Mit diesen kombinierten Merkmalen wird dann ein HMM pro Systemkommando trainiert.

Für das HMM mit später Fusion werden individuelle Sprach- und Gesten-HMM trainiert. Die beiden Ströme werden in der Erkennungsphase unabhängig voneinander dekodiert. Das endgültige Systemkommando wird dann durch eine Ergebnisfusion des Sprach- und des Gesten-HMM erzeugt: Dazu wird die Wahrscheinlichkeiten der drei wahrscheinlichsten Erkennungsergebnisse sowohl aus dem Sprach- als auch aus dem Wortstrom miteinander multipliziert, unmögliche Kombinationen gelöscht und dann das Systemkommando mit der höchsten Gesamtwahrscheinlichkeit ausgewählt.

In [162] wurde ein Dynamic Time Warping Algorithmus (DTW) auf die gleichen bimodalen Daten angewendet. Dabei wurde eine individuelle Erkennung der beiden Modalitäten durchgeführt und dann eine späte Fusion durch Kombination der individuellen Ergebnisse angewendet. Außerdem wurde der DTW Algorithmus so erweitert, dass eine gleichzeitige asynchrone Dekodierung der beiden Merkmale durchgeführt werden konnte; Details dazu finden sich in [162]. Im Ergebnisteil dieser Arbeit werden zum Vergleich auch diese DTW Ergebnisse angegeben.

4.4.3 Ergebnisse

Abbildung 4.7 zeigt die Anordnung von Geste und Sprache für ein Beispiel aus der Datenbank sowie die gefundene Anordnung durch das HMM mit früher Fusion und das AHMM. Es handelt sich hierbei um ein Beispiel mit teilweiser Überlappung der beiden Modalitäten: Die Geste und das Sprachkommando starten synchron zum Zeitpunkt $t = 0$. Während die Geste bei $t = 32$ endet, wird das Sprachkommando noch bis $t = 100$ fortgesetzt.

Durch die Anwendung des in (4.20) beschriebenen Viterbiverfahrens kann mit dem AHMM auch die beste zeitliche Anordnung (im Sinne der maximalen Gesamtproduktionswahrscheinlichkeit) zwischen den zwei Modalitäten gefunden werden. Für das Beispiel ist die auf diese Weise mit dem AHMM gefundene Anordnung als durchgezogener Verlauf in Abbildung 4.7 gezeigt. Das AHMM modelliert die wahre Anordnung nicht vollständig korrekt, in diesem Beispiel werden die Symbole des Gestenstromes im Verhältnis zur Sprache zu langsam emittiert: Der Gestenstrom beginnt erst bei $t = 3$ und endet für die optimale Anordnung erst bei $t = 60$. Trotzdem wird durch das AHMM die ungefähre Charakteristik des Sprach-Gestenkommandos erhalten.

Mit einem HMM mit früher Fusion kann diese Anordnung nicht nachgebildet werden. Das HMM emittiert zu jedem Zeitpunkt sowohl ein Merkmal von der Sprache als auch von der Geste. Die frühe Fusion erlaubt daher als zeitliche Anordnung

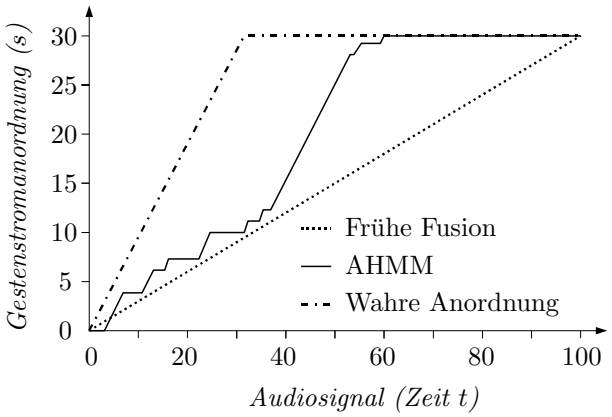

Abbildung 4.7: Anordnung vom Gesten- zum Sprachstrom für ein Beispiel aus der Datenbank. In der wahren Anordnung des Beispiels starten sowohl die Geste als auch das Sprachkommando zum Zeitpunkt $t = 0$. Die Geste endet bei $t = 32$, während das Sprachkommando noch bis $t = 100$ andauert. Mit einem HMM mit früher Fusion kann diese Anordnung nicht nachgebildet werden. Das AHMM kann die wahre Anordnung auch nicht vollständig rekonstruieren, jedoch stimmt die generelle Charakteristik mit dem wirklichen Verlauf überein.

immer nur eine gerade Linie zwischen Ursprung und dem Ende des Beispiels.

Dies hat große Auswirkung auf die Erkennungsleistung des HMM mit früher Fusion. Es wurden in dieser Arbeit viele verschiedene Konfigurationen getestet, aber kein Modell konnte die hohe Asynchronität der Daten modellieren. Trotz des relativ großen Trainingsdatensatzes konnte das HMM mit früher Fusion die teilweise erheblichen zeitlichen Verschiebungen nicht generalisieren. Das Modell kann deswegen ähnliche Datensätze nicht erkennen, wenn Sprache und Geste unterschiedlich zeitlich angeordnet sind. Dadurch sind die Erkennungsraten des HMM auch nur unwesentlich besser als die Ratewahrscheinlichkeit. Die frühe Fusion kann daher für dieses Problem nicht erfolgreich angewendet werden.

Mit einem HMM-Ansatz mit später Fusion kann das Problem wesentlich besser modelliert werden. Es wurden verschiedene Kombinationen von Sprach- und Gesten-HMM mit unterschiedlicher Anzahl von Zuständen evaluiert. Die Erkennungsraten für drei verschiedene Konfigurationen sind in Tabelle 4.1 angegeben. Mit einem HMM mit 15 Zuständen für den Sprachstrom und einem HMM mit fünf Zuständen für den Gestenstrom können nach der späten Fusion 67,19% der Systemkommandos richtig erkannt werden.

Bei Verwendung des DTW Ansatzes mit später Fusion konnten immer noch über 62% der Kommandos richtig erkannt werden. Durch die asynchrone Erweiterung des DTW kann dies auf 72,75% gesteigert werden. Damit kann der prinzipiell relativ einfache DTW Algorithmus durch das Zulassen asynchroner Verschiebungen bereits über 5,5% mehr Kommandos richtig erkennen als das HMM mit später Fusion.

Tabelle 4.1: Erkennungsraten für die Fusion von bimodalen Sprach- und Gestenkommandos zu Systemkommandos. Für das AHMM sind die Ergebnisse in Abhängigkeit der verwendeten Zustände und für das HMM in Abhängigkeit der verwendeten Wort- (W) und Gesten-Zustände (G) angegeben. Zum Vergleich sind auch die Ergebnisse bei Verwendung eines DTW Algorithmus mit später und asynchroner Fusion angeben; diese sind [162] entnommen.

Modell	Zustände	Rate (%)
Asynchrones HMM	5	73,66
	10	77,08
	25	77,62
HMM mit später Fusion	10W/5G	64,20
	10W/10G	60,86
	15W/5G	67,19
DTW mit später Fusion		62,08
Asynchroner DTW		72,75

Diese Erkennungsraten können durch Anwendung des AHMM noch einmal deutlich gesteigert werden. Tabelle 4.1 zeigt die Ergebnisse für unterschiedliche AHMM-Zustände. Schon mit fünf Zuständen werden 73,66% der Kommandos richtig zugeordnet und damit alle anderen Modelle überboten. Bei Verwendung von 25 Zuständen können 77,62% der Systemkommandos richtig erkannt werden. Gegenüber dem asynchronen DTW entspricht das einer Verbesserung von 4,87%. Gegenüber dem besten HMM mit später Fusion erzielt das AHMM eine Verbesserung von 10,43%, eine relative Fehlerreduktion von fast 31,8%. Dies zeigt, dass das Ausnutzen von Asynchronitäten in der Dekodierung zu deutlich besseren Fusionsergebnissen führt. Durch die eingebrachte Verbesserung des Vorwärts-Algorithmus kann die Dekodierung der Daten in Echtzeit erfolgen, das AHMM ist daher auch praktisch für die bimodale Fusion anwendbar.

4.5 Kapitelzusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde die Vorwärtsberechnung des asynchronen Hidden Markov Modells durch Einführung von dynamischen Bereichsgrenzen im Trellis-Diagramm verbessert. Dadurch werden nur noch Punkte im Trellis-Diagramm berechnet, die tatsächlich auf einem möglichen Pfad liegen. Auf diese Weise konnte die Komplexität sowohl für das Training als auch für die Dekodierung erheblich reduziert werden. Trotzdem ist die mit dem verbesserten Vorwärts-Algorithmus berechnete Wahrscheinlichkeit mathematisch exakt. Mit dem verbesserten Vorwärts-

Algorithmus konnte dann auch eine Skalierungsprozedur entwickelt werden, die den Wertebereich der Berechnungen auf numerisch sinnvolle Bereiche begrenzt.

Durch die Reduktion der Komplexität in der Dekodierungsphase konnte das AHMM im Rahmen dieser Arbeit auch erstmalig auf die Fusion von bimodalen Benutzereingabedaten angewendet werden. Da die auftretenden Asynchronitäten hier besonders hoch sein können, war es mit der bisherigen Vorwärtsberechnung nicht möglich, dieses Problem zu bearbeiten.

Die theoretisch hergeleitete Komplexitätsreduktion des entwickelten Algorithmus wurde auch durch Experimente belegt. In einem weiteren Experiment wurde das AHMM auf die Fusion von Sprach- und Gestenkommandos angewendet. Hier konnte durch das Nutzen der asynchronen Information die Erkennungsrate um mehr als 10% gegenüber einem HMM mit später Fusion gesteigert werden. Dies zeigt das große Potential asynchroner Modelle für die multimodale Fusion.

In begleitenden Arbeiten wurde das AHMM auch eingesetzt, um Konferenzen zu analysieren [7] oder die beste Kameradarstellung einer Konferenz zu bestimmen (Details dazu in Kapitel 6) [5]. Diese beiden Aspekte wurden in diesem Kapitel nicht behandelt, zeigen aber weitere Möglichkeiten, das AHMM anzuwenden.

Das AHMM bietet noch einige Erweiterungsmöglichkeiten, die bisher nicht behandelt wurden. Insbesondere die Erweiterung auf mehr als zwei Datenströme scheint für die multimodale Fusion vielversprechend. Allerdings steigt die Zeitkomplexität mit jedem weiteren Datenstrom stark an, da eine Vielzahl neuer Freiheitsgrade für die möglichen zeitlichen Anordnungen aller Ströme untereinander hinzukommen. Ein solches Modell würde praktisch sehr schnell unberechenbar werden. Es sollte daher erforscht werden, wie die zeitliche Anordnung sinnvoll eingegrenzt werden kann, ohne die prinzipielle Funktionsweise des AHMM zu beeinträchtigen.

Aus Anwendungssicht wäre es interessant, asynchrone Knoten in aktuelle GM-Softwarewerkzeuge einzubinden. Dazu muss das Konzept der wechselnden Eltern lediglich auf wechselnde Kinder erweitert werden. Dies führt nur zu kleineren Softwareänderungen, die prinzipielle Funktionsweise des JT-Algorithmus wäre nicht betroffen. Eine Änderung wird aber erst möglich sein, wenn für die wichtigen Toolboxen – wie GMTK – auch der Source-Code veröffentlicht wird. Eine Erweiterung würde dann ein einfaches, aber sehr vielversprechendes Verbinden asynchroner und anderer GM erlauben.

5

Robuste Konferenzanalyse mit einem HMM-LDS

In einer kürzlich veröffentlichten Studie [70] wurden die Teilnehmer gebeten, Emotionen und Stimmungen zu nennen, die ihrer Meinung nach häufig in Konferenzen oder Besprechungen von den Konferenzteilnehmern empfunden werden. Die häufigste Antwort – von zwei Drittel der Teilnehmer genannt – war „langweilig“. Die zweithäufigste Stimmung – immer noch von einem Drittel aller Teilnehmer genannt – war „verärgert“. Dies impliziert, dass viele Personen Besprechungen und Konferenzen als nicht besonders effektiv und zum Teil sogar als Zeitverschwendung empfinden.

Auf der anderen Seite können Konferenzen, Besprechungen und auch Vorträge bei vielen Personen einen beträchtlichen Teil des Arbeitsalltags ausmachen. Für den Informationsfluss in Unternehmen oder anderen gößeren Organisationen sind sie zum Teil unerlässlich: So werden z. B. bei der Intel Corporation mehr als drei Millionen Stunden für benötigte Konferenzräume und fast sechs Millionen Stunden für Telephonkonferenzen pro Jahr veranschlagt. Besprechungen und Konferenzen sind in großen Firmen so wichtig, dass Intel jedes Jahr 56 Tausend Schulungsstunden nur darüber abhält, wie eine Besprechung möglichst effektiv wird [72].

Dies zeigt, dass es zwischen der Wichtigkeit von Besprechungen in großen Organisationen auf der einen Seite und der Wahrnehmung der Teilnehmer über diese Besprechungen auf der anderen Seite eine sehr große Diskrepanz gibt. In Projekten wie dem ICSI Meeting Project [78, 110], M4 – The Multimodal Meeting Manager [99, 100], Computers in the Human Interaction Loop (CHIL) [152] oder Augmented Multi-party Interaction (AMI) [4, 40] wird daher erforscht, wie Computer genutzt werden können, um die Effektivität von Konferenzen, Besprechungen und Vorträgen zu steigern, und wie sie automatisch analysiert werden können.

Ein erster Ansatz zur automatischen Auswertung von Besprechungen ist die Segmentierung in Gruppendiskussionsphasen (*meeting group actions*), z. B. Diskussion, Präsentation oder Monolog. Die Abfolge solcher Phasen kann dann als eine erste, sehr grobe Strukturierung der Besprechung genutzt werden, um darauf aufbauend

wichtige Punkte zu finden, eine Agenda zu erstellen oder die Besprechung zusammenzufassen [106]. Weil diese erste Unterteilung elementar für die weiterführende Analyse von Besprechungen ist, wurden bereits verschiedene automatische Methoden, basierend auf HMM [106], DBN [53] oder auch semantischen Merkmalen [130] vorgestellt und auf aufgezeichnete Besprechungen erfolgreich angewendet. Einen umfassenden Vergleich verschiedener Methoden hierzu gibt [3].

In echten Besprechungen können die Daten jedoch auf veschiedene Weise gestört sein: Ereignisse wie das versehentliche Zuschlagen einer Tür oder Hintergrundgespräche können die Audioaufzeichnung überdecken oder stören. Die Videoaufzeichnungen können ganz oder teilweise durch Personen maskiert werden, die die Konferenz verlassen oder betreten; ein tragbarer Computer kann vor einem Teilnehmer stehen und ihn damit für die Kamera verdecken. Diese realistischen Bedingungen beeinflussen das Verhalten der automatischen Methoden und führen im Allgemeinen zu einer Verschlechterung der Erkennungsleistung.

Im Rahmen dieser Arbeit [8, 9] wird deshalb ein neuer multimodaler Ansatz für die automatische Klassifikation von Gruppenaktionen in Besprechungen entworfen: Ein LDS und ein gekoppeltes HMM werden zu einem GM verbunden. Dabei wird die akustische Information über ein HMM als treibender Eingang für das LDS verwendet, das wiederum die visuelle Information filtert. Auf diese Weise kann das GM mehr Information aus dem visuellen Kanal nutzen.

5.1 Konferenzdaten

Die Daten für diese Arbeit wurden im Rahmen des M4-Projektes [99, 100] im IDIAP smart meeting room [109] aufgezeichnet. Eine Skizze dieses Besprechungsraumes ist in Abbildung 5.1 (links) gezeigt. Der Raum ist mit einem Tisch für vier Personen, einem Whiteboard und einem Projektor mit dazugehöriger Leinwand ausgestattet.

Der Datensatz besteht aus 57 Besprechungen[1] mit vier Teilnehmern (P1 - P4) und einer Länge von jeweils ungefähr fünf Minuten. Alle Besprechungen werden audio-visuell aufgezeichnet: Jeder Teilnehmer trägt ein omni-direktionales Ansteckmikrophon. Auf der Mitte des Tisches ist ein Mikrophonarray (A1) mit acht omnidirektionalen Mikrophonen platziert. Am Kopfende des Tisches steht eine binaurale Puppe (B) mit zwei weiteren Mikrophonen. Videos werden mit insgesamt drei Kameras im Raum aufgenommen. Zwei Kameras zeichnen jeweils zwei Teilnehmer von der linken (L) bzw. rechten (R) Seite des Raumes auf. Eine weitere Kamera (Z) steht am Kopfende des Tisches und zeichnet eine Totale des Raumes mit allen vier Teilnehmern, dem Whiteboard und der Leinwand auf. Bei einer Änderung wird jedes auf die Leinwand projezierte Bild, mit einem Zeitstempel versehen, als Bild gespeichert. Auf das Whiteboard geschriebene Daten werden als x-y-Koordinaten und die

[1] Ursprünglich wurden im M4-Projekt 60 Besprechungen aufgezeichnet, drei wurden jedoch aus Datenschutzgründen vor der Veröffentlichung wieder aus dem Datensatz entfernt.

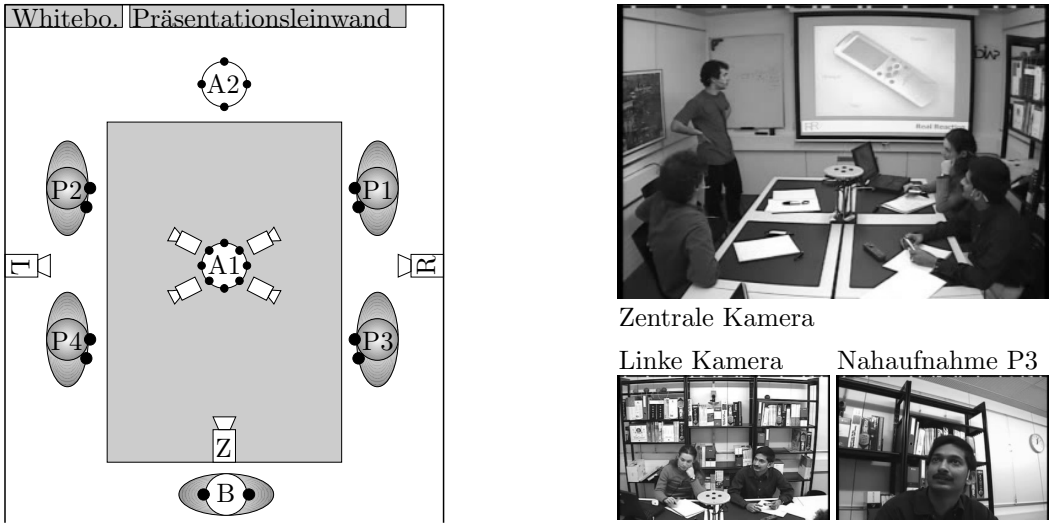

Abbildung 5.1: Skizze des IDIAP Besprechungsraumes [109] (links, nicht maßstabsgerecht). Der Raum ist mit einem Tisch für vier Personen (P1-P4), einem Projektor mit Leinwand und einem Whiteboard ausgestattet. Die Daten werden zeitsynchronisiert mit mehreren Mikrophonen (schwarze Kreise) und Kameras aufgezeichnet. Rechts sind Beispielbilder der zentralen (Z) und linken (L) Kamera sowie eine Nahaufnahme der Person P3 gezeigt.

individuellen Notizen aller vier Teilnehmer mit digitalen Logitech I/O Stiften gespeichert. Alle Aufnahmen wurden mit einem gemeinsamen, zentralen Zeitstempel versehen, wodurch ein hohes Mass an Synchronität der Daten gewährleistet wird.

Störungen in den Daten

Um den Einfluss von verschiedenen Störungen auf die Erkennungsleistung zu evaluieren, werden die Daten künstlich verfälscht. Die akustischen Aufnahmen sowohl von den Nah- als auch den Fernfeldmikrophonen werden mit einem Hintergrundgespräch mit 10dB SNR überlagert. Um Verdeckungen (z. B. durch einen Computer oder Personen) zu simulieren, werden die visuellen Aufnahmen mit Störungen an verschiedenen Positionen überlagert. Dazu werden die Aufnahmen der rechten und linken Kamera jeweils in zwei Regionen aufgeteilt, um für jede Person ein individuelles Video zu erhalten. Dies ist beispielhaft für die Person P4 in Abbildung 5.2 (a) gezeigt. Die so erhaltenen Videos der vier Personen werden dann für drei verschiedene Datensätze an verschiedenen Positionen je zu einem Drittel mit einem grauen Balken überlagert (linkes, mittleres und rechtes Drittel der Videos; Abbildung 5.2, b-d). Für einen weiteren Datensatz werden die Videos auch mit einem grauen Kreuz, das 5/9 des Bildes bedeckt, überlagert (Abbildung 5.2, e). In einem letzten Evaluierungsdatensatz werden die Videos mit 10dB Gauß'schem Rauschen überlagert (ohne

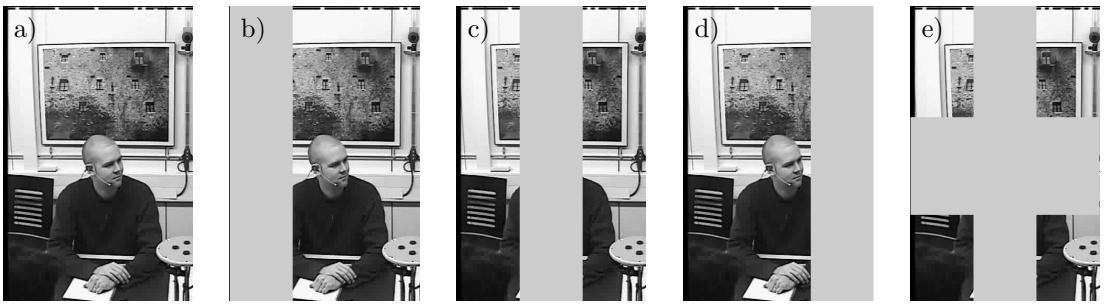

Abbildung 5.2: Aus der rechten Kamera ausgeschnittene Region mit der Person P4 (a) und dasselbe Bild mit den überlagerten Störungen (b-e).

Abbildung). Gerade weil es sich bei den so erzeugten Daten um künstliche und nicht natürliche Störungen handelt, hat der Datensatz durch die definierten Störungen gute Eigenschaften, um die Robustheit von Erkennern zu testen. Die Daten wurden deshalb unter anderem schon in anderen Arbeiten zur robusten Erkennung von Personenaktionen in Besprechungen verwendet [172].

Die ersten 30 aufgezeichneten Besprechungen des Datensatzes werden zum Trainieren und die verbleibenden 27, unbekannten Besprechungen zum Testen der Modelle verwendet. Dabei werden für das Training nur ungestörte Daten (also ohne akustische oder visuelle Verdeckungen) und zum Testen Datensätze mit verschiedenen Kombinationen der akustischen und der visuellen Störungen verwendet.

Erweiterter AMI Datensatz

Für den Datensatz des AMI Projektes – der in Kapitel 6 verwendet wird – wird der Raum noch einmal erweitert: Jeder Teilnehmer trägt dann zusätzlich ein Kondensator-Headset-Mikrophon. Die in Abbildungs 5.1 gezeigten vier Kameras für die Nahaufnahmen der Teilnehmer sowie das zweite Mikrophonarray (A2) werden ebenfalls erst für die AMI Daten zum Raum hinzugefügt und sind für den in diesem Kapitel benutzten M4-Datensatz noch nicht vorhanden.

5.2 Besprechungsanalyse und Gruppenaktionen

Ähnlich wie in der Videoanalyse in Abschnitt 3.1 kann auch eine Besprechung in verschiedenen vertikalen Ebenen analysiert werden. Dies ist beispielhaft in Abbildung 5.3 verdeutlicht. Eine Besprechung wird als eine Sequenz von Ereignissen modelliert. Je nach Anwendung wird dabei eine unterschiedliche Ebene und damit auch andere Ereignisse verwendet, um die Besprechung zu analysieren. So kann z. B. eine Besprechung als eine Abfolge von sehr elementaren Gruppentätigkeiten wie „Informationsaustausch“ oder „Entscheidung“ modelliert werden. Die gleiche Besprechung kann aber auch in einer anderen Ebene als eine Sequenz von Gruppeninteresse (z. B.

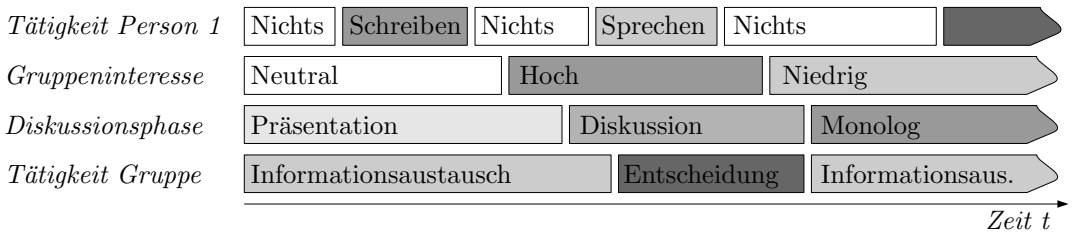

Abbildung 5.3: Modellierung einer Besprechung als Sequenz von Ereignissen in verschiedenen vertikalen Ebenen. Innerhalb jeder Ebene kann die Besprechung als eine Abfolge von definierten Ereignissen analysiert werden. So hat z. B. jeder Teilnehmer eine eigene Abfolge von individuellen Aktionen, alle Teilnehmer zusammen befinden sich aber auch in einer der Gruppendiskussionsphasen. Je nach Anwendung sind weitere, nicht gezeigte Ebenen denkbar. Im Rahmen dieser Arbeit wird die Abfolge der Gruppendiskussionsphasen untersucht.

zum Finden von besonders wichtigen Punkten in der Besprechung) oder auch als eine Sequenz individueller Aktionen der Teilnehmer [153, 172] modelliert werden.

Im Rahmen dieser Arbeit werden die häufig verwendeten [3, 53, 106, 130] acht Gruppendiskussionsphasen (*meeting group actions*)

$$G = \{G_{\mathrm{D}}, G_{\mathrm{M,P1}}, G_{\mathrm{M,P2}}, G_{\mathrm{M,P3}}, G_{\mathrm{M,P4}}, G_{\mathrm{N}}, G_{\mathrm{P}}, G_{\mathrm{W}}\},$$

verwendet. Diese werden auch als Gruppenaktionen bezeichnet. Die möglichen Aktionen dabei sind:

Diskussion, $G_{\mathbf{D}}$: zwei oder mehr Personen in der Besprechung reden miteinander,

Monolog, $G_{\mathbf{D},P_i}$: die Person P_i redet ohne unterbrochen zu werden,

Notiz, $G_{\mathbf{N}}$: die Personen notieren etwas,

Präsentation, $G_{\mathbf{P}}$: eine Person steht vor oder neben der Leinwand und gibt eine Präsentation,

Whiteboard, $G_{\mathbf{W}}$: eine Person schreibt etwas an das Whiteboard.

In dem hier verwendeten M4-Datensatz hat eine fünfminütige Besprechung durchschnittlich fünf Gruppendiskussionsphasen. Jede Besprechung wird dann als eine Sequenz dieser Phasen modelliert. Diese Sequenz kann entweder direkt als sehr grobe Strukturierung verwendet werden oder alternativ kann aufbauend auf die Gruppendiskussionsphasen eine Agenda oder eine Zusammenfassung erstellt werden [106] – dabei dienen die Wechsel der Phasen als erste Anhaltspunkte für Themenänderungen oder wichtige Diskussionspunkte.

5.3 Merkmale

Für die automatische Auswertung der Besprechungen werden aus den aufgezeichneten Daten audio-visuelle Merkmale extrahiert. Durch die Geometrie des Besprechungsraumes befindet sich jede Person an einer von sechs verschiedenen vordefinierten Positionen (*locations*) L:

$$L = \{C_1, C_2, C_3, C_4, W, P\},$$

dabei ist $C_1 - C_4$ sitzend auf oder stehend nahe bei einem der vier Stühle, W vor oder neben dem Whiteboard und P vor oder neben der Präsentationsleinwand. Aus den Kameras und dem Mikrophonarray können mit dieser Information für jede Position im Raum audio-visuelle Merkmale extrahiert werden. Diese sind nicht personen-, sondern ausschließlich positionsabhängig.

Zusätzlich können mit den Daten der Ansteckmikrophone sprecherabhängige Merkmale extrahiert werden. Diese sind unabhängig von der Position der Person im Raum. Insgesamt werden so 68 verschiedene personen- und positionsabhängige audio-visuelle Merkmale aus den Daten extrahiert. Dabei werden sowohl das Mikrophonarray, die Ansteckmikrophone als auch die Kameras verwendet.

5.3.1 Visuelle Merkmale

Aus den Kameraaufzeichnungen der Besprechung werden globale Bewegungsmerkmale (*global motions*) [153] extrahiert. Diese repräsentieren eine holistische, globale Bewegung in dem Video. Obwohl es sich bei diesen Bewegungsmerkmalen um relativ einfache visuelle Merkmale handelt, haben sie zwei wesentliche Vorteile: Im Vergleich zu anderen visuellen Merkmalen (z. B. dem optischen Fluss) können sie sehr schnell berechnet werden. Zum anderen reduzieren sie den Videostrom auf wenige Merkmale. Diese können dann leicht weiterverarbeitet werden, erhalten aber gleizeitig die Charakteristik der Bewegung im Video. Aufgrund dieser Eigenschaften wurden globale Bewegungsmerkmale z. B. bereits erfolgreich für die Erkennung von Aktionen in Besprechungen [153, 173] und für die automatische Gestenerkennung [58] eingesetzt.

Zur Gewinnung der Merkmale aus den Kameraaufzeichnungen wird für jede der sechs Positionen L im Besprechungsraum und für jeden Zeitpunkt t ein Differenzbild $I^L_{d'}(x, y, t)$ durch pixelweise Subtraktion zweier aufeinanderfolgender Bilder $I^L(x, y, t)$ aus dem Video erzeugt:

$$I^L_{d'}(x, y, t) = \left| I^L(x, y, t) - I^L(x, y, t-1) \right|. \tag{5.1}$$

Ein so aus dem Video erzeugtes Differenzbild zeigt Abbildung 5.4. Durch den statischen Hintergrund und die fixe Kameraposition kann dieses Bild als Bewegung der Personen interpretiert werden. Die Differenzbilder werden dann mit einem Schwell-

Abbildung 5.4: Zwei Bilder der rechten Kamera und das daraus erzeugte (ungefilterte) Differenzbild. Dieses kann als Bewegung der Personen im Besprechungsraum interpretiert werden. Zur besseren Visualisierung wurden hier zwei Bildern verwendet, die vier Sekunden auseinander liegen. Für die Erzeugung der globalen Bewegungsmerkmale wird das Differenzbild aus unmittelbar aufeinanderfolgenden Bildern berechnet.

wert S gefiltert, so dass

$$I_d^L(x,y,t) = \begin{cases} 0 & , \text{ wenn } I_{d'}^L(x,y,t) \leq S \\ I_{d'}^L(x,y,t) & , \text{ sonst} \end{cases} \tag{5.2}$$

eventuelles Rauschen unterdrückt wird. Aus der Sequenz der Differenzbilder werden dann für jede Position L sieben globale Bewegungsmerkmale extrahiert.

Das Bewegungszentrum in x- und y-Richtung ergibt sich aus dem Masseschwerpunkt des Differenzbildes:

$$m_x^L(t) = \frac{\sum_{(x,y)} x \; I_d^L(x,y,t)}{\sum_{(x,y)} I_d^L(x,y,t)} \tag{5.3}$$

und

$$m_y^L(t) = \frac{\sum_{(x,y)} y \; I_d^L(x,y,t)}{\sum_{(x,y)} I_d^L(x,y,t)}. \tag{5.4}$$

Durch die Änderung der Bewegung in x- und y-Richtung wird die Dynamik beschrieben:

$$\Delta m_x^L(t) = m_x^L(t) - m_x^L(t-1) \tag{5.5}$$

und

$$\Delta m_y^L(t) = m_y^L(t) - m_y^L(t-1). \tag{5.6}$$

Weiterhin wird die Standardabweichung in der Bewegung in x- und y-Richtung berechnet:

$$\sigma_x^L(t) = \frac{\sum_{(x,y)} I_d^L(x,y,t) \; \left(x - m_x^L(t)\right)}{\sum_{(x,y)} I_d^L(x,y,t)} \tag{5.7}$$

und

$$\sigma_y^L(t) = \frac{\sum_{(x,y)} I_d^L(x,y,t)\ \left(y - m_y^L(t)\right)}{\sum_{(x,y)} I_d^L(x,y,t)}. \tag{5.8}$$

Als letztes Merkmal wird die Intensität der Bewegung berechnet:

$$i^L(t) = \frac{\sum_{(x,y)} I_d^L(x,y,t)}{\sum_{(x,y)} 1}, \tag{5.9}$$

dabei beschreiben kleine Intensitäten eine kleine Bewegung oder einen stationären Zustand an der Position, während große Intensitäten auf eine größere Bewegung schließen lassen.

Diese sieben Merkmale werden dann für jeden Zeitpunkt t zu dem von der Position L abhängigen Merkmalvektor

$$\vec{o}_t^L = [m_x^L(t), m_y^L(t), \Delta m_x^L(t), \Delta m_y^L(t), \sigma_x^L(t), \sigma_y^L(t), i^L(t)]^T \tag{5.10}$$

zusammengefasst. Dadurch werden die sehr hochdimensionalen Videodaten auf lediglich sieben Merkmale pro Zeitschritt und Position reduziert. Trotzdem bleiben die wesentlichen Bewegungseigenschaften der Personen im Besprechungsraum erhalten. Durch das Zusammenfügen der Merkmale für alle sechs Positionen im Raum ergibt sich der endgültige globale Bewegungsmerkmalvektor

$$\vec{o}_t^{\,\mathrm{V}} = [\vec{o}_t^{C_1}, \vec{o}_t^{C_2}, \vec{o}_t^{C_3}, \vec{o}_t^{C_4}, \vec{o}_t^{W}, \vec{o}_t^{P}]^T, \tag{5.11}$$

der die gesamte Bewegung im Besprechungsraum zum Zeitpunkt t mit insgesamt nur 42 visuellen Merkmalen beschreibt. Die Merkmalrate korreliert dabei mit der Framerate des Videos, d. h. pro Sekunde werden 25 Merkmalvektoren erzeugt. Insgesamt werden dann für eine Besprechung mit einer Länge von L Sekunden deshalb $T_V = L \cdot 25\frac{1}{s}$ Merkmalvektoren erzeugt. Alle Merkmalvektoren $\vec{o}_t^{\,\mathrm{V}}$ einer Besprechung bilden zusammen den gesamten visuellen Merkmalverlauf $\vec{o}^{\,\mathrm{V}}$, der dann ingesamt T_V 42-dimensionale Beobachtungen enthält.

5.3.2 Akustische Merkmale

Aus den Ansteckmikrophonen werden der Personen werden alle 10ms jeweils vier Mel-Frequenz-Cepstrum Koeffizienten (*Mel frequency cepstral coefficients, MFCC*) [60, 132] und zusätzlich die Energie extrahiert. Dazu wird ein Hamming-Fenster mit 25ms Breite verwendet. Aus den Merkmalen aller vier Personen wird dann der 20-dimensionale sprecherabhängige akustische Merkmalvektor $\vec{o}_t^{\,\mathrm{MFCC}}$ geformt.

Im Rahmen der Arbeit wurde auch mit höherdimensionalen und anderen MFCC Konfigurationen experimentiert, so z. B. mit dem in der Spracherkennung verbereiteten 39-dimensionalen MFCC-Vektor [145], der sich aus den ersten zwölf MFCC

Koeffizienten, der Signalenergie und der ersten und zweiten Ableitung ergibt. Für alle vier Personen führt dies jedoch zu einem 156-dimensionalen akustischen Merkmalvektor, wodurch sich Probleme ergeben: Durch den relativ kleinen Trainingsdatensatz stehen nicht genügend Beispiele zur Verfügung, um alle Parameter zu bestimmen. Es konnte daher keine Verbesserung der Erkennungsleistung festgestellt werden. Weiterhin enthalten hochdimensionale MFCC-Merkmalvektoren im wesentlichen Informationen über die gesprochene Sprache, diese ist für die Erkennung der Gruppenaktionen jedoch nicht erforderlich. Die Verwendung des 20-dimensionalen Merkmalvektors $\vec{o}_t^{\,\mathrm{MFCC}}$ hat sich daher für die automatische Besprechungsanalyse im Rahmen dieser Arbeit als ausreichend gezeigt.

Positionsabhängige akustische Merkmale

Mit der Steered Response Power-Phase Transformation (SRP-PHAT) [52] wird aus den Mikrophonarraydaten für jede der sechs Positionen L im Raum die Sprachaktivität bestimmt. Dabei beschreibt das SRP-PHAT-Maß mit einem kontinuierlichen Wert die Sprachaktivität an einer der definierten Positionen. Die Daten für alle sechs Positionen werden dann mit einem Schwellwert gefiltert. Liegt die Sprachaktivität zum Zeitpunkt t an der Position L unter einem vordefinierten Schwellwert, dann wird für diese Position und diesen Zeitpunkt eine Null vergeben; liegt die Sprachaktivität über dem Schwellwert wird eine Eins vergeben. So erhält man aus den Mikrophonarraydaten eine sechs-dimensionale, binäre Sprach-Pause-Segmentierung (*binary speech and silence segmentation, BSP*) $\vec{o}_t^{\,\mathrm{BSP}}$ [106], die angibt, an welchen der sechs Positionen des Raumes gerade gesprochen wird[2].

Zusammen ergeben die sprecherabhängigen Merkmale $\vec{o}_t^{\,\mathrm{MFCC}}$ und die positionsabhängige Sprach-Pause-Segmentierung $\vec{o}_t^{\,\mathrm{BSP}}$ dann

$$\vec{o}_t^{\,\mathrm{A}} = [\vec{o}_t^{\,\mathrm{MFCC}}, \vec{o}_t^{\,\mathrm{BSP}}]^T, \tag{5.12}$$

den 26-dimensionalen akustischen Merkmalvektor.

Alle T_A Merkmalsvektoren $\vec{o}_t^{\,\mathrm{A}}$ einer Besprechung bilden zusammen den gesamten akustischen Merkmalverlauf $\vec{o}^{\,\mathrm{A}}$, der dann ingesamt T_A 26-dimensionale Beobachtungen enthält.

Für eine Besprechung werden weniger visuelle als akustische Merkmale extrahiert, d. h. $T_A > T_V$. Soweit es für die Modelle erforderlich ist – z. B. für eine frühe Fusion – wird eine Anpassung vorgenommen. Dazu wird der visuelle Merkmalsstrom so interpoliert, dass er die gleiche Länge wie der akustische Merkmalstrom hat. Dann enthalten sowohl der akustische als auch der visuelle Merkmalverlauf jeweils $T = T_A = T_V$ Beobachtungen.

[2] Die SRP-PHAT Daten wurden im Rahmen des M4-Projektes vom IDIAP (`http://www.idiap.ch`) zur Verfügung gestellt und in dieser Arbeit nicht selbst erzeugt.

5.4 Kombiniertes HMM-LDS

Zur Erkennung der Gruppenaktionen aus gestörten Daten wird in dieser Arbeit ein GM verwendet, das ein HMM und ein LDS kombiniert. Das LDS filtert dabei die gestörten visuellen Daten und vereinfacht damit die Erkennung für das HMM. Durch die direkte Kombination der beiden Modelle in einem GM handelt es sich jedoch nicht um eine sequentielle Abarbeitung, sondern das LDS nutzt gleichzeitig die Dekodierungsinformation aus dem HMM, um damit die visuellen Daten zu filtern. Beide Modelle beeinflussen sich also gegenseitig.

Die Verbindung eines HMM mit einem LDS in einem GM wird als gemischtes HMM-LDS-System (*mixed-state GM*) bezeichnet. Es wurde ursprünglich entwickelt und erfolgreich eingesetzt um menschliche Gesten zu erkennen [121]. Dabei wurde jedoch stets nur ein HMM mit einem LDS verbunden. Im Rahmen dieser Arbeit wird dieser Ansatz so erweitert, dass ein gekoppeltes HMM mit parallelen Markov-Ketten mit einem LDS verbunden wird. Dadurch kann das Modell gleichzeitig eine multimodale Fusion durchführen. Der treibende Eingang des LDS hängt dann nicht nur von den visuellen Daten ab, sondern nutzt auch die akustische Information zum Filtern des visuellen Kanals.

5.4.1 LDS zur Filterung der visuellen Daten

Wie in Abschnitt 2.10.5 beschrieben, kann ein LDS mit der Systemzustandsübergangsmatrix $\underset{\sim}{A}$, der Systemeingangsmatrix $\underset{\sim}{B}$ und der Systemausgabematrix $\underset{\sim}{C}$ mit den drei Zustandsgleichungen

$$\vec{x}_1 = \underset{\sim}{B}\vec{u}_1 + \vec{v}_1, \tag{5.13}$$

$$\vec{x}_t = \underset{\sim}{A}\vec{x}_{t-1} + \underset{\sim}{B}\vec{u}_t + \vec{v}_t, \tag{5.14}$$

$$\vec{o}_t^{\,\mathrm{V}} = \underset{\sim}{C}\vec{x}_t + \vec{w}_t \tag{5.15}$$

beschrieben werden. Dabei ist $\vec{v}_t = \mathcal{N}(\vec{v}_t, \vec{0}, \underset{\sim}{Q})$ und $\vec{w}_t = \mathcal{N}(\vec{w}_t, \vec{0}, \underset{\sim}{R})$ das System- bzw. das Beobachtungsrauschen, $\vec{x}_t$ ist der verborgene, kontinuierliche Systemzustand, $\vec{u}_t$ die das System treibende Eingangsgröße und $\vec{o}_t^{\,\mathrm{V}}$ die Observierung. Wie ebenfalls in Abschnitt 2.10.5 gezeigt, kann dieses physikalische System auch durch Wahrscheinlichkeitsverteilungen dargestellt werden:

$$p(\vec{x}_1 \,|\, \vec{u}_1) = \mathcal{N}(\vec{x}_1, \underset{\sim}{B}\vec{u}_1, \underset{\sim}{Q}), \tag{5.16}$$

$$p(\vec{x}_t \,|\, \vec{x}_{t-1}, \vec{u}_t) = \mathcal{N}(\vec{x}_t, \underset{\sim}{A}\vec{x}_{t-1} + \underset{\sim}{B}\vec{u}_t, \underset{\sim}{Q}), \tag{5.17}$$

$$p(\vec{o}_t^{\,\mathrm{V}} \,|\, \vec{x}_t) = \mathcal{N}(\vec{o}_t^{\,\mathrm{V}}, \underset{\sim}{C}\vec{x}_t, \underset{\sim}{R}). \tag{5.18}$$

Mit diesem System kann der visuelle Kanal der Beobachtung gefiltert werden. Als Systemausgabe des LDS werden dafür die visuellen Merkmale $\vec{o}_t^{\,\mathrm{V}}$ als Beobachtung verwendet.

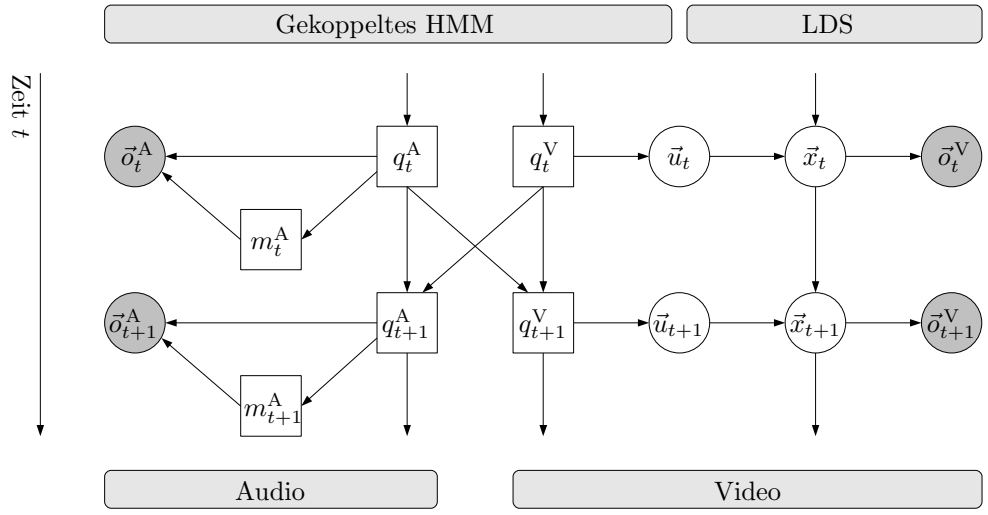

Abbildung 5.5: GM des gemischen HMM-LDS Systems. Das gekoppelte HMM modelliert mit zwei parallelen HMM den akustischen und den visuellen Kanal. Es führt dadurch die multimodale Fusion durch. Dabei wird der akustische Kanal durch ein HMM mit GMM Ausgang modelliert. Das visuelle HMM hat keine echte Beobachtung, sondern erzeugt den treibenden Eingang für das LDS. Dieses physikalische System filtert dann den visuellen Kanal. Durch das gekoppelte HMM hat dann auch der akustische Kanal einen Einfluss auf die Filterung des visuellen Kanals.

Das zugehörige GM ist in Abbildung 5.5 gezeigt. Dabei verläuft die Zeit in dieser Darstellung von oben nach unten; ein Zeitschlitz umfasst also alle Knoten innerhalb einer Zeile (zuzüglich des leicht nach unten versetzen Mixturknotens m). Das beschriebene LDS besteht in jedem Zeitschlitz aus den drei rechten, kontinuierlichen Knoten $\vec{u}_t$, $\vec{x}_t$ und $\vec{o}_t^{\,\mathrm{V}}$. Der zeitliche Verlauf wird dabei über die vertikale Verbindung zwischen $\vec{x}_t$ und $\vec{x}_{t+1}$ modelliert. Dieses LDS ist in dem GM direkt mit einem gekoppeltem HMM verbunden. Das HMM dient dabei als treibende Eingangsgröße für das physikalische System, ist also mit den Knoten $\vec{u}_t$ verbunden.

5.4.2 Gekoppeltes HMM für die Erkennung

In [121] wurde das kombinierte System zur Erkennung von Gesten genutzt. Daher gab es nur eine Modalität und ein einfaches HMM als treibender Eingang war ausreichend. Hier soll das System genutzt werden, um die akustischen und visuellen Merkmale zu fusionieren, um damit Gruppenaktionen zu erkennen. Dabei soll vorallem auch die akustische Information genutzt werden, um den fehlerhaften visuellen Kanal zu unterstützen. Deswegen wird das in [121] vorgeschlagene Modell im Rahmen dieser Arbeit auf ein multimodales, gekoppeltes HMM als treibender Eingang erweitert.

Wie in Abschnitt 2.10.4 beschrieben, besteht ein gekoppeltes HMM aus zwei parallelen Markov-Ketten, die in der Erkennungsphase miteinander interagieren. Dabei hängt der Zustand eines HMM nicht nur vom vorherigen Zustand, sondern auch vom vorherigen Zustand des zweiten HMM ab. Dadurch ist das Modell sehr gut zur Datenfusion mehrer paralleler Ströme geeignet.

Das verwendete HMM ist ebenfalls in Abbildung 5.5 gezeigt. Dabei wird der akustische Strom mit einem HMM mit GMM Observierung modelliert und besteht deshalb in jedem Zeitschlitz aus dem Zustand q_t^{A}, dem Mixturknoten m_t^A und den akustischen Merkmalen $\vec{o}_t^{\,\mathrm{A}}$ als Beobachtung. Dieses HMM ist gekoppelt mit dem visuellen HMM, das aus dem Zustand q_t^{V} und der Ausgabe $\vec{u}_t$ besteht. Das visuelle HMM hat jedoch keine echte Beobachtung, sondern erzeugt in jedem Zeitschritt statt einer Observierung den treibenden Eingang $\vec{u}_t$ des physikalischen Systems.

5.4.3 Kombination der Systeme

In dem GM, das in Abbildung 5.5 gezeigt ist, formen das gekoppelte HMM und das LDS zusammen das gemischte HMM-LDS System. Der Knoten $\vec{u}_t$ ist sowohl Bestandteil des HMM als auch des LDS; er stellt also die Verbindung zwischen beiden Systemen her. Dadurch ist die Ausgabe des visuellen HMM gleichzeitig die Systemeingabe des LDS. Durch die Verwendung eines gekoppelten HMM beeinflusst der Zustand des akustischen HMM auch den Zustand des visuellen HMM. Damit beeinflussen die akustischen Merkmale automatisch auch das LDS und damit die Filterung des visuellen Kanals. Die Filterung selbst prognostiziert die augenblickliche Bewegung auf Basis der vorherigen visuellen Merkmale und des augenblicklichen Zustands des gekoppelten HMM. Dadurch können Störungen in den Bewegungen (wie z. B. die hier auftretenden Verdeckungen) teilweise ausgeglichen werden. Dem HMM selbst steht dann mehr Information für die Dekodierung zur Verfügung.

Mit dem kombinierten GM in Abbildung 5.5 faktorisiert die Verbundwahrscheinlichkeit der audio-visuellen Observierungen und der Zustände dann zu

$$
\begin{aligned}
&p(\vec{o}^{\,\mathrm{A}}, \vec{o}^{\,\mathrm{V}}, \vec{q}^{\,\mathrm{A}}, \vec{q}^{\,\mathrm{V}}, \vec{m}^{\,\mathrm{A}}, \vec{u}, \vec{x}) = \\
&\overbrace{p(q_1^{\mathrm{A}}) \prod_{t=2}^{T} p(q_t^{\mathrm{A}} \mid q_{t-1}^{\mathrm{A}}, q_{t-1}^{\mathrm{V}}) \prod_{t=1}^{T} p(\vec{o}_t^{\,\mathrm{A}} \mid m_t^{\mathrm{A}}, q_t^{\mathrm{A}})\, p(m_t^{\mathrm{A}} \mid q_t^{\mathrm{A}})}^{\text{Akustischer Teil des gekoppelten HMM}} \\
&\underbrace{p(q_1^{\mathrm{V}}) \prod_{t=2}^{T} p(q_t^{\mathrm{V}} \mid q_{t-1}^{\mathrm{A}}, q_{t-1}^{\mathrm{V}}) \prod_{t=1}^{T} p(\vec{u}_t \mid q_t^{\mathrm{V}})}_{\text{Visueller Teil des gekoppelten HMM}} \underbrace{p(\vec{x}_1 \mid \vec{u}_1) \prod_{t=2}^{T} p(x_t \mid x_{t-1}, \vec{u}_t) \prod_{t=1}^{T} p(\vec{o}_t^{\,\mathrm{V}} \mid \vec{x}_t)}_{\text{LDS}},
\end{aligned}
\tag{5.19}
$$

wobei hier davon ausgegangen wird, dass der akustische und der visuelle Merkmalvektor jeweils die gleiche Länge $T = T_S = T_A$ haben. Eine Anpassung für unter-

schiedliche Merkmalraten ist, wie bei einem normalen gekoppelten HMM, jedoch trivial.

Der Mixturknoten ist nur eine Hilfsvariable und wird in der Verbundwahrscheinlichkeit nicht explizit benötigt. Die Variable m_t^{A} kann daher herausmarginalisiert werden. Es ergibt sich dann bei Verwendung von M Gaußmixturen die Verbundwahrscheinlichkeit:

$$
\begin{aligned}
&p(\vec{o}^{\,\mathrm{A}}, \vec{o}^{\,\mathrm{V}}, \vec{q}^{\,\mathrm{A}}, \vec{q}^{\,\mathrm{V}}, \vec{u}, \vec{x}) = \\
&\quad p(q_1^{\mathrm{A}}) \prod_{t=2}^{T} p(q_t^{\mathrm{A}} \mid q_{t-1}^{\mathrm{A}}, q_{t-1}^{\mathrm{V}}) \prod_{t=1}^{T} \sum_{m=1}^{M} p(\vec{o}_t^{\,\mathrm{A}} \mid m^{\mathrm{A}}, q_t^{\mathrm{A}})\, p(m^{\mathrm{A}} \mid q_t^{\mathrm{A}}) \\
&\quad p(q_1^{\mathrm{V}}) \prod_{t=2}^{T} p(q_t^{\mathrm{V}} \mid q_{t-1}^{\mathrm{A}}, q_{t-1}^{\mathrm{V}}) \prod_{t=1}^{T} p(\vec{u}_t \mid q_t^{\mathrm{V}})\;\; p(\vec{x}_1 \mid \vec{u}_1) \prod_{t=2}^{T} p(x_t \mid x_{t-1}, \vec{u}_t) \prod_{t=1}^{T} p(\vec{o}_t^{\,\mathrm{V}} \mid \vec{x}_t).
\end{aligned}
\tag{5.20}
$$

5.4.4 Approximative Berechnung im Modell

Ohne das HMM könnte das LDS mit Nachrichtenpropagation im Verbundbaum (d. h. hier die Rauch Rekursion [128] bzw. der Kalman Filter [85]) berechnet werden. Hier aber ist der treibende Eingang selbst wieder ein dynamisches System. Dadurch werden die Berechnungen selbst für kürzeste Beobachtungen, also sehr kleine T, unberechenbar [121]: Zum ersten Zeitpunkt besteht die WDF von $\vec{x}_1$ aus nur einer Gaußverteilung. Unter der Annahme, dass der Ausgang des HMM U mögliche Werte annehmen kann, besteht die WDF $\vec{x}_2$ zum nächsten Zeitpunkt allerdings schon aus einer Mixtur von U Gaußkurven. Das heißt zum Zeitpunkt t besteht die WDF aus U^t Gaußkurven. Dies ist selbst für kurze Merkmalsequenzen praktisch nicht berechenbar.

In [121] wurde daher für das gemischte HMM-LDS System ein approximatives Verfahren für die Nachrichtenpropagation entwickelt. Dieses basiert auf strukturiert abweichendem Schließen (*structured variational inference*) [83]. Diese Methode kann mit kleinen Änderungen auch auf das hier vorgestellt gekoppelte HMM-LDS angewendet werden.

Zuerst sollen die Grundzüge des Verfahrens erläutert werden, ohne in alle Details zu gehen. Beweise zum strukturiert abweichendem Schließen, eine ausführlichere Herleitung des angewendeten Verfahrens und Fehlerabschätzungen finden sich z. B. in [64, 83, 120, 121].

Kurzeinführung in das strukturiert abweichende Schließen

Für das approximative Schließen mit strukturierter Abweichung wird ein Graph $\mathcal{G}^{\approx}$ gesucht, der eine ähnliche Verbundwahrscheinlichkeit wie das ursprüngliche Modell $\mathcal{G}$ repräsentiert. Dabei soll das neue GM $\mathcal{G}^{\approx}$ jedoch vollständig berechenbar

sein. Häufig – aber nicht zwangsweise – besteht es aus mehreren Untergraphen des ursprünglichen, unberechenbaren GM. $\mathcal{G}^{\approx}$ faktorisiert dann die Verbundwahrscheinlichkeit gemäß einer Hilfsverteilung $Q(\cdot \mid \lambda)$, die von den gleichen Variablen wie die ursprüngliche Verbundwahrscheinlichkeit $p(\cdot)$ abhängt, jedoch zusätzlich abhängig von einem neuen Variationsparameter λ ist.

Bei einem Modell $\mathcal{G}^{\approx}$ mit der Menge der beobachteten Knoten o und den verborgenen Knoten h ergibt sich dann für die Auftrittswahrscheinlichkeit einer Konfiguration der verborgenen Variablen die Hilfsfunktion $Q(h \mid o, \lambda)$. Zum approximativen Schließen wird diese Hilfsverteilung mit dem Variationsparameter so angepasst, dass sie möglichst ähnlich zu der ursprünglichen, aber unberechenbaren Auftrittswahrscheinlichkeit $p(h \mid o)$ ist. Dazu wird die Kullback-Leibler-Divergenz [92]

$$D(Q \parallel P) = \int_h Q(h \mid o, \lambda) \, \log \frac{Q(h \mid o, \lambda)}{p(h \mid o)} \, \mathrm{d}h, \tag{5.21}$$

die ein Maß für die Unterschiedlichkeit der beiden Verteilungen darstellt, minimiert:

$$\lambda^* = \operatorname*{argmin}_{\lambda} D(Q \parallel P). \tag{5.22}$$

Die sich daraus ergebende Verteilung

$$Q(h \mid o, \lambda^*) \tag{5.23}$$

ist dann – im Sinne der strukturierten Abweichung mit dem Graphen $\mathcal{G}^{\approx}$ – die beste berechenbare Approximation der unberechenbaren Verteilung $p(h \mid o)$. Dabei ist jedoch zu beachten, dass die sich ergebende Verteilung in Gleichung (5.23) nur optimal für den vorgegebenen Graphen $\mathcal{G}^{\approx}$ ist. Die Güte der Approximation hängt also stark von der Wahl der Struktur $\mathcal{G}^{\approx}$ ab; diese sollte daher möglichst ähnlich zum ursprünglichen unberechenbaren GM $\mathcal{G}$ sein.

Ziel der strukturierten Abweichung ist es also, zuerst einen neuen Graphen zu finden, der die Variablen der ursprünglichen Wahrscheinlichkeitsverteilung $p(\cdot)$ mit der Hilfsverteilung $Q(\cdot)$ möglichst ähnlich faktorisiert, dabei aber gleichzeitig vollständig berechenbar ist. Außerdem muss das neue GM noch von einem Variationsparameter λ abhängen. Dieser neue Graph wird nur einmal aufgestellt und dann nicht mehr verändert. Zum approximativen Schließen wird die Hilfsverteilung $Q(\cdot)$ dann mit dem Variationsparameter für jede Beobachtung jeweils neu über eine Minimierung der Kullback-Leibler-Divergenz (5.21) möglichst gut an die ursprüngliche, nicht berechenbare Verteilung angepasst.

Strukturiert abweichendes Schließen für das HMM-LDS-System

Für das gemischte HMM-LDS System aus Abbildung 5.5 ergibt sich für die bekannte Observierung $\left\{\vec{o}^{\,\mathrm{A}}, \vec{o}^{\,\mathrm{V}}\right\}$, nach Marginalisierung von $\vec{u}$ aus Gleichung (5.20), für die

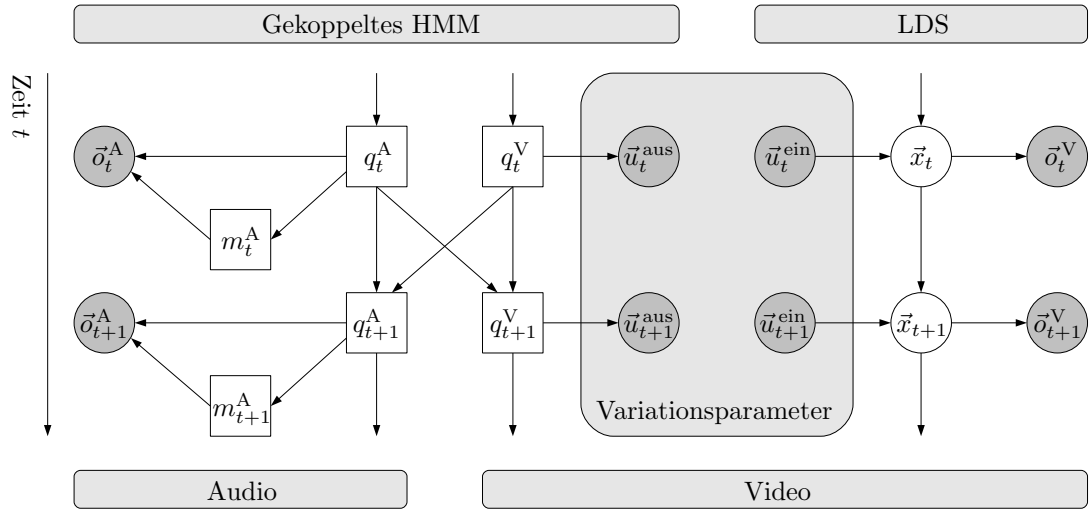

Abbildung 5.6: Zerlegung des gemischten HMM-LDS in ein gekoppeltes HMM und ein LDS. Diese beiden hier voneinander getrennten Modelle können mit der Nachrichtenpropagation für GM vollständig berechnet werden. Durch die Variationsparameter $\vec{u}_t^{\,\mathrm{aus}}$ und $\vec{u}_t^{\,\mathrm{ein}}$ sind sie jedoch trotzdem lose miteinander verbunden.

verborgenen Variablen $\vec{q}^{\,\mathrm{A}}, \vec{q}^{\,\mathrm{V}}$ und $\vec{x}$ die Auftrittswahrscheinlichkeit

$$p(\vec{q}^{\,\mathrm{A}}, \vec{q}^{\,\mathrm{V}}, \vec{x} \,|\, \vec{o}^{\,\mathrm{A}}, \vec{o}^{\,\mathrm{V}}), \tag{5.24}$$

die selbst für kleine T praktisch nicht berechenbar ist. Die Auftrittswahrscheinlichkeit ist aber sowohl zur Dekodierung unbekannter Beobachtungen als auch als Prozedur während des Lernens der Modellparameter erforderlich.

Mit dem Graphen in Abbildung 5.6 ist eine Approximation von Gleichung (5.24) mittels strukturiert abweichenden Schließens möglich. Den Vorschlägen in [121] folgend, wird in dieser Arbeit das LDS und das gekoppelte HMM in zwei parallele Modelle aufgeteilt. Die Hilfsverteilung $Q(\cdot)$ faktorisiert daher in zwei Komponenten $Q_{\mathrm{HMM}}(\cdot)$ für das gekoppelte HMM und $Q_{\mathrm{LDS}}(\cdot)$ für das physikalische System:

$$Q\left(\vec{q}^{\,\mathrm{A}}, \vec{q}^{\,\mathrm{V}}, \vec{x} \,|\, \vec{o}^{\,\mathrm{A}}, \vec{o}^{\,\mathrm{V}}, \lambda\right) = Q_{\mathrm{HMM}}\left(\vec{q}^{\,\mathrm{A}}, \vec{q}^{\,\mathrm{V}} \,|\, \vec{o}^{\,\mathrm{A}}, \lambda\right)\; Q_{\mathrm{LDS}}\left(\vec{x} \,|\, \vec{o}^{\,\mathrm{V}}, \lambda\right). \tag{5.25}$$

Durch das aufgespaltene Modell wird die Variable $\vec{u}$ in beiden parallelen Modellen zu einem beobachteten Knoten. Dann ist $\vec{u}_t^{\,\mathrm{aus}}$ die Observierung des HMM und $\vec{u}_t^{\,\mathrm{ein}}$ der Eingang des LDS. Zusammen formen

$$\lambda = \left\{\vec{u}_1^{\,\mathrm{aus}}, \ldots, \vec{u}_T^{\,\mathrm{aus}}, \vec{u}_1^{\,\mathrm{ein}}, \ldots, \vec{u}_T^{\,\mathrm{ein}}\right\} \tag{5.26}$$

die Variationsparameter, die die beiden Teilmodelle miteinander verbinden. Dafür werden sie für jede unbekannte Beobachtung $\left\{\vec{o}^{\,\mathrm{A}}, \vec{o}^{\,\mathrm{V}}\right\}$ so gewählt, dass $Q(\cdot)$ die ur-

sprüngliche Verteilung (5.24) optimal approximiert. Dazu wird die Kullback-Leibler-Divergenz

$$\lambda^* = \underset{\lambda}{\operatorname{argmin}} \sum_{\vec{q}^{\mathrm{A}}} \sum_{\vec{q}^{\mathrm{V}}} \int_{\vec{x}} Q\left(\vec{q}^{\mathrm{A}}, \vec{q}^{\mathrm{V}}, \vec{x} \,|\, \vec{o}^{\mathrm{A}}, \vec{o}^{\mathrm{V}}, \lambda\right) \log \frac{Q\left(\vec{q}^{\mathrm{A}}, \vec{q}^{\mathrm{V}}, \vec{x} \,|\, \vec{o}^{\mathrm{A}}, \vec{o}^{\mathrm{V}}, \lambda\right)}{p(\vec{q}^{\mathrm{A}}, \vec{q}^{\mathrm{V}}, \vec{x} \,|\, \vec{o}^{\mathrm{A}}, \vec{o}^{\mathrm{V}})} \,\mathrm{d}\vec{x} \tag{5.27}$$

zwischen der usprünglichen Verteilung $p(\cdot)$ und der Approximation $Q(\cdot)$ minimiert. Das erste Theorem in [64] gibt dafür eine allgemeine, iterative Lösung. Da die in dieser Arbeit neu eingeführte Information über den Audio-Strom in dem gekoppelten HMM implizit mit in der visuellen Markov-Kette modelliert ist, ergibt sich eine ähnliche Lösung wie für ein normales gemischtes HMM-LDS. Für die Minimierung von Gleichung (5.27) können daher die in [121] gefundenen Optimierungsvorschriften so angepasst werden, dass auch die akustischen Merkmale Einfluss auf die Variationsparameter nehmen. Dann ergibt sich für ein LDS mit U-dimensionalem Eingang

$$\vec{u}_t^{*,\mathrm{ein}} = \left[u_t^{*,\mathrm{ein}}(1), \ldots, u_t^{*,\mathrm{ein}}(i), \ldots, u_t^{*,\mathrm{ein}}(U)\right]^T \in \mathfrak{R}^U, \tag{5.28}$$

für die i-te Komponenten von $\vec{u}_t^{*,\mathrm{ein}}$ die Optimierungsvorschrift

$$u_t^{*,\mathrm{ein}}(i) = \sum_{q_t^{\mathrm{V}}} p(\vec{u}^{\,\mathrm{aus}} = i \,|\, q_t^{\mathrm{V}})\; p(q_t^V \,|\, \vec{o}^{\mathrm{A}}, \vec{u}_1^{\mathrm{aus}}, \ldots, \vec{u}_T^{\mathrm{aus}}). \tag{5.29}$$

Mit dem Erwartungswert

$$\langle \vec{x}_t \rangle = \int_{\vec{x}_t} \vec{x}_t \; p(\vec{x}_t \,|\, \vec{o}^{\mathrm{V}}, \vec{u}_1^{\mathrm{ein}}, \ldots, \vec{u}_T^{\mathrm{ein}}) \mathrm{d}\vec{x}_t \tag{5.30}$$

ergibt sich für den i-ten Eintrag von $\vec{u}_t^{*,\mathrm{aus}}$ des HMM-Ausgangs die Lernvorschrift

$$u_t^{*,\mathrm{aus}}(i) = \exp\left\{ \vec{d}_i^{\,T} \; \underset{\sim}{Q}^{-1} \left(\langle \vec{x}_t \rangle - \underset{\sim}{A} \langle \vec{x}_{t-1} \rangle - \frac{1}{2} \vec{d}_i \right) \right\}, \tag{5.31}$$

wobei $\underset{\sim}{Q}$ wiederum die Kovarianzmatrix des Systemrauschens und $\vec{d}_i$ die zum Zustand i gehörigen Symbolemissionswahrscheinlichkeiten, d. h. die i-te Spalte der Symbolemissionsmatrix bezeichnen. Um die Berechnung auch für den ersten Zeitschritt zu ermöglichen, wird $\langle \vec{x}_0 \rangle = \vec{0}$ definiert.

Mit diesen optimierten Variationsparametern λ^* kann $\vec{u}^{\,\mathrm{aus}} = \{\vec{u}_1^{\mathrm{aus}}, \ldots, \vec{u}_T^{\mathrm{aus}}\}$ als eine virtuelle Beobachtung für das gekoppelte HMM angesehen werden. Dabei stellen die einzelnen $\vec{u}_t^{\mathrm{aus}}$ keine Beobachtungssymbole dar, sondern enthalten bereits die Emissionswahrscheinlichkeiten für alle Zustände i. So kann $Q_{\mathrm{HMM}}\left(\vec{q}^{\mathrm{A}}, \vec{q}^{\mathrm{V}} \,|\, \vec{o}^{\mathrm{A}}, \lambda^*\right)$ mit der normalen Nachrichtenpropagation in Verbundbäumen (in diesem Fall ergibt sich ein einfacher Vorwärts-Rückwärts-Algorithmus) vollständig berechnet werden. Ebenso kann $\vec{u}^{\,\mathrm{ein}} = \left\{\vec{u}_1^{\mathrm{ein}}, \ldots, \vec{u}_T^{\mathrm{ein}}\right\}$ als virtueller treibender Eingang für das LDS

betrachtet werden und damit kann $Q_{\text{LDS}}\left(\vec{x} \mid \vec{o}^{\,\text{V}}, \lambda^*\right)$ ebenfalls durch die normale Nachrichtenpropagation (in diesem Fall ergibt sich der Kalman Filter bzw. die Rauch Rekursion) vollständig berechnet werden.

Die Hilfsverteilung (5.25) kann dann als das Produkt der beiden Teilhilfsfunktionen Q_{HMM} und Q_{LDS} auch für sehr große T berechnet und als eine Approximation für die nicht berechenbare Verteilung (5.24) genutzt werden.

Es ist jedoch zu beachten, dass die Gleichungen (5.29) und (5.31) voneinander abhängen: Zum Optimieren der LDS-Parameter $\vec{u}^{\,\text{ein}}$ mit Gleichung (5.29) muss zuerst das gekoppelte HMM berechnet werden. Für diese Berechnung sind jedoch die fiktiven Beobachtungen $\vec{u}^{\,\text{aus}}$ erforderlich, die sich wiederum aus Gleichung (5.31) ergeben. Dafür ist jedoch die Berechnung des LDS erforderlich. Die Optimierung muss daher iterativ durchgeführt werden und es ergibt sich der in [121] vorgestellte Algorithmus zum approximativen Schließen:

Algorithmus 4 Iteratives, approximatives Schließen für das gemischte HMM-LDS

Benötigt: Unbekannte Beobachtung $\{\vec{o}^{\,\text{A}}, \vec{o}^{\,\text{V}}\}$ und Modellparameter
  function AUFTRITTSWAHRSCHEINLICHKEIT($\vec{o}^{\,\text{A}}, \vec{o}^{\,\text{V}}$)
    Initialisiere $\langle \vec{x}_t \rangle$
    repeat
      Bestimme $\vec{u}^{\,\text{aus}}$ aus $\langle \vec{x}_t \rangle$ mit Gleichung (5.31)
      Schätze $p(q_t^V \mid \vec{o}^{\,\text{A}}, \vec{u}_1^{\,\text{aus}}, \ldots, \vec{u}_T^{\,\text{aus}})$ aus $\vec{o}^{\,\text{A}}$ und $\vec{u}^{\,\text{aus}}$ mit JT im HMM
      Bestimme $\vec{u}^{\,\text{ein}}$ aus $p(q_t^V \mid \vec{o}^{\,\text{A}}, \vec{u}_1^{\,\text{aus}}, \ldots, \vec{u}_T^{\,\text{aus}})$ mit Gleichung (5.29)
      Schätze $\langle \vec{x}_t \rangle$ aus $\vec{o}^{\,\text{V}}$ und $\vec{u}^{\,\text{ein}}$ mit JT im LDS
    until Lösung konvergiert
  end function

Dabei gilt die Lösung als konvergiert, wenn sich die Variationsparameter nicht mehr deutlich verändern, oder ein festgelegter Fehlerschwellwert unterschritten wird. Dafür muss dann in jedem Schätzdurchlauf der durch die Approximation entstehende Fehler neu abgeschätzt werden, Grenzwerte und Methoden dafür sind z. B. in [120] angegeben.

5.4.5 Lernen der Modellparameter

Das Lernen der Modellparameter des vorgestellten gemischten HMM-LDS kann – wie für GM allgemein in Abschnitt 2.9 gezeigt – mit dem EM-Algorithmus erfolgen. Dabei wird im Schätzschritt zum Berechnen der Erwartungswerte jedoch keine vollständige Berechnung durchgeführt (die selbst für kleine Merkmalsequenzen nicht berechenbar ist), sondern es wird das im vorherigen Abschnitt gezeigte approximative Verfahren verwendet.

Dadurch entsteht beim Lernen eine doppelte Approximation: zum einen verursacht durch die Schätzung des EM-Algorithmus und zum anderen durch die vom

EM-Algorithmus als Prozedur verwendete Approximation für den Erwartungswert. Es kann jedoch gezeigt werden [83], dass das approximative Berechnen des Erwartungswertes als Teil eines generalisierten EM-Verfahrens angesehen werden kann, für das wiederum Konvergenzbeweise existieren. Die doppelte Approximation hat daher keinen praktisch relevanten Einfluss auf das Lernen.

Auf die explizite Ableitung der Lernformeln für die Parameter wird hier verzichtet. Eine vollständige Auflistung aller benötigten Formeln für das gemischte HMM-LDS System, allerdings ohne Herleitung, gibt [120]. Detaillierte Herleitungen zu den HMM Lernformeln (die nur minimal für das gekoppelte System modifziert werden müssen) finden sich z. B. in [10, 126], detaillierte Herleitungen zu den LDS Lernformeln (die wiederum nur minimal für das gekoppelte System modifziert werden müssen) zeigt [66].

5.5 Experimente

Das hier vorgestellte gemischte HMM-LDS System mit parallelen Markov-Ketten wird mit den in Abschnitt 5.1 vorgestellten Konferenzdaten evaluiert. Dabei soll jeder Besprechungsphase eine der acht in Abschnitt 5.2 vorgestellten Gruppendiskussionsaktionen zugeordnet werden. Die Phasen sind dabei bereits anhand der Transkription segmentiert; es handelt sich also um eine reine Klassifikationsaufgabe ohne automatische Segmentierung.

Als Merkmale werden die in Abschnitt 5.3 vorgestellten globalen Bewegungsmerkmale für den visuellen Kanal und die MFCC- und Sprach-Pause-Merkmale (BSP) für den akustischen Kanal verwendet.

5.5.1 Vergleichsmodelle

Das gemischte HMM-LDS System wird mit drei unimodalen und einem multimodalem HMM verglichen. Dabei werden Modelle mit unterschiedlicher Anzahl verborgener Zustände evaluiert. Jedes unimodale HMM wird jeweils nur mit einer der drei Modalitäten – visuell, MFCC oder BSP – trainiert und evaluiert. Diesen HMM steht also sowohl im Training als auch im Test nur die Information von einem Kanal zur Verfügung. Dazu müssen die unterschiedlichen Merkmalraten der Kanäle nicht untereinander angepasst werden.

Für das multimodale HMM wird eine frühe Merkmalfusion durchgeführt. Dazu werden die Merkmalraten der Kanäle aneinander angeglichen, indem der visuelle Kanal auf die Länge des akustischen Kanals interpoliert wird. Dann werden die visuellen und die akustischen Merkmale in einem 68-dimensionalen Merkmalstrom zusammengefügt, der dann sowohl für das Training als auch den Test verwendet wird.

5.5.2 Testdatensätze

Alle Modelle werden mit den ungestörten Daten aus insgesamt 30 Besprechungen trainiert. Für die Testdatensätze werden die Daten der verbleibenden 27 unbekannten Besprechungen mit den in Abschnitt 5.1 beschriebenen Störungen behaftet. Jedoch werden für keines der Modelle mit Störungen behaftete Daten auch im Training verwendet; deren Eigenschaften können also nicht mitgelernt werden.

Für den ersten Testdatensatz (a) wird in keinem der Kanäle eine Störungen verwendet. In drei Testsätzen wird ein vertikaler Balken links (b), in der Mitte (c) bzw. rechts (d) über die Videos gelegt, der jeweils ein Drittel des Bildes verdeckt. In einem weiteren Testdatensatz (e) wird ein graues Kreuz über die Videos gelegt. Diese Störungen der Datensätze (b-e) sind auch in Abbildung 5.2 gezeigt. In einem letzten visuell gestörten Testdatensatz (f) werden die Videos mit einem Gauß'schen Rauschen mit 10dB SNR überlagert. In den visuell gestörten Datensätzen (b-f) wird der akustische Kanal ohne Störungen verwendet.

Es werden außerdem noch zwei akustisch gestörte Datensätze evaluiert: Für den ersten akustisch gestörten Datensatz (I) werden die Aufnahmen der Ansteckmikrophone (MFCC) mit einem Hintergrundgespräch mit 10dB SNR überlagert. Für den zweiten akustisch gestörten Datensatz (II) werden die Aufnahmen des Mikrophonarrays (BSP) mit 10dB SNR Störgeräusch überlagert.

In einem letzen Datensatz (AV) werden alle drei Kanäle mit Störungen behaftet: Sowohl die Ansteckmikrophone als auch das Mikrophonarray werden wiederum mit einem 10dB SNR Hintergrundgespräch überlagert und der visuelle Kanal durch einen grauen Balken in der Mitte der Videos verdeckt.

5.5.3 Ergebnisse

Tabelle 5.1 zeigt die Erkennungsergebnisse der Gruppenaktionen für die verschiedenen Testdatensätze und für alle angewendeten Modelle. Für die unimodalen HMM können die besten Ergebnisse mit fünf Zuständen erreicht werden. Die beiden akustischen Merkmale MFCC und BSP haben dann mit den unimodalen HMM schon gute Erkennungsraten von 83,1% bzw. 83,6%. Die globalen Bewegungsmerkmale aus dem visuellen Kanal enthalten weniger Information über die Gruppenaktionen, so dass nur eine Erkennungsrate von 67,2% erreicht wird. Nach der Merkmalfusion trägt der visuelle Kanal dann aber zu einer höheren Erkennungsleistung bei: Mit einem multimodalen HMM mit früher Merkmalfusion und zehn verborgenen Zuständen können dann bereits 85,2% und mit dem gemischten HMM-LDS, mit zehn Zuständen, 88,7% der Gruppenaktionen richtig erkannt werden.

Dieser Beitrag des visuellen Kanals zur Erkennung wird noch einmal deutlich größer, wenn die akustische Information gestört ist. Für den Datensatz mit MFCC Rauschen (I) fällt die unimodale Erkennungsrate um 22% absolut auf nur noch 61,1%. Das multimodale HMM-LDS hingegen ist kaum von der Störung im MFCC Kanal

Tabelle 5.1: Erkennungsergebnisse der Gruppenaktionen in Besprechungen für die Testdatensätze mit den verschiedenen Störungen. Getestet werden drei unimodale HMM, die jeweils nur auf die korrespondierenden Merkmale angewendet werden, ein multimodales HMM mit früher Fusion und das gemischte HMM-LDS System.

Testdatensatz		Unimodal (in %) MFCC	BSP	Visuell	Multimodal (in %) HMM	HMM-LDS
a)	Ungestört	83,1	83,6	67,2	85,2	88,7
b)	Links verdeckt			40,9	82,6	87,8
c)	Mitte verdeckt			44,3	83,5	76,5
d)	Rechts verdeckt			52,2	85,2	86,1
e)	Kreuz verdeckt			33,0	79,1	81,7
f)	Visuelles Rauschen			42,6	84,4	87,8
I)	MFCC Rauschen	61,1			80,9	87,0
II)	BSP Rauschen		75,4		83,5	87,0
AV)	A-V Störung				80,0	81,7

beeinflusst und fällt nur um weniger als 2% auf immer noch 87% Erkennungsrate. Für die Störung des BSP Kanals (II) ergeben sich ähnliche Ergebnisse: Das unimodale Modell fällt deutlich in der Erkennungsleistung ab und erreicht nur noch 75,4%, während das HMM-LDS wiederum nur relativ schwach von der Störung beeinflusst wird und nur auf 87% abfällt. Diese Ergebnisse zeigen, dass der visuelle Kanal in einem multimodalen System deutlich zur Erkennung beitragen kann.

Dieser Effekt kann auch umgekehrt beobachtet werden. Wenn der visuelle Kanal gestört wird (b-f), fällt die Erkennungsrate des visuellen HMM auf durchschnittlich 42,6% und damit um fast 25% absolut im Vergleich zu den ungestörten Daten. Je nach Art der Störung ist die Verschlechterung dabei unterschiedlich stark. Im Vergleich dazu erreicht das multimodale HMM-LDS auch bei einem gestörtem visuellen Kanal immer noch eine Erkennungsrate von durchschnittlich 84% und fällt durch die Störung daher um weniger als 5% ab.

Die Verwendung eines multimodalen Systems hat für dieses Problem also deutliche Vorteile gegenüber allen drei unimodalen Erkennern. Bei dem Vergleich des multimodalen HMM mit früher Fusion mit dem hier vorgestellten multimodalen gemischten HMM-LDS System zeigt sich, dass das HMM-LDS bis auf einen Datensatz (c) für alle Evaluierungen bessere Erkennungsraten als das multimodale HMM erzielt. Für den ungestörten Datensatz ergibt sich eine Verbesserung von 85,2% auf 88,7%, dies entspricht einer relativen Fehlerreduktion von über 23%. Für das MFCC Rauschen (I) erzielt das HMM-LDS eine relative Fehlereduktion von fast 32% (6,1% absolute Fehlerreduktion). Für die Daten mit gestörtem visuellen Kanal kann die Verbesserung des HMM-LDS gegenüber dem multimodalen HMM bis zu 5,2% (30%

relative Fehlerreduktion) betragen (b).

Die Ergebnisse zeigen, dass das HMM-LDS wesentlich mehr Information aus dem visuellen Kanal nutzt als ein multimodales HMM. Insbesondere haben Störungen in den Kanälen weniger Einfluss auf die Erkennungsleistung. Das HMM-LDS ist daher robuster als das multimodale HMM und alle unimodalen HMM.

5.6 Kapitelzusammenfassung und Ausblick

In diesem Kapitel wurde ein neues multimodales, gemischtes HMM-LDS GM zur robusten Erkennung von Gruppenaktionen in Besprechungen vorgestellt. Das GM kann Störungen in dem akustischen oder visuellen Kanal kompensieren. Dazu werden die Merkmalströme in einem gekoppelten HMM fusioniert. Das HMM selbst ist wiederum mit einem LDS verbunden. Dabei dient das HMM als treibender Eingang des physikalischen Systems. Mit dem LDS werden Störungen ausgeglichen, wovon das HMM in der Erkennung profitiert. Die so in einem GM zusammgefassten Modelle interagieren daher während der Erkennung miteinander und das Gesamtsystem wird robuster gegenüber fehlerbehafteten Merkmalen.

Allerdings ist das GM mit normaler Nachrichtenpropagation im Verbundbaum praktisch nicht mehr berechenbar. Es wurde daher ein approximatives Verfahren – basierend auf strukturiert abweichendem Schließen – für das GM vorgestellt. Dazu muss das HMM wieder vom LDS entkoppelt werden, wobei die beiden Teilmodelle über einen Variationsparameter lose miteinander verbunden bleiben. Durch die Entkopplung wird das GM wieder vollständig berechenbar. Während der Dekodierung wird der Variationsparameter dann so angepasst, dass das berechenbare Modell möglichst gut das ursprüngliche Modell repräsentiert.

Das vorgestellte HMM-LDS wurde in einem experimentellen Teil auf die Erkennung von Gruppenaktionen in Besprechungen angewendet und mit drei unimodalen und einem multimodalen HMM verglichen. Dabei erzielte das vorgestellte GM für alle bis auf eine Konfiguration das beste Erkennungsergebnis. So konnte das GM z. B. für ungestörte Daten eine Erkennungsrate von 88,7% erreichen, dies enspricht einer relativen Fehlerreduktion von 23,6% gegenüber einem multimodalen HMM mit früher Fusion. Bei visuellen Störungen kann das GM, je nach Art der Verdeckung, den relativen Fehler um fast 30% gegenüber dem HMM senken. Dies zeigt die hohe Robustheit, die sich aus der direkten Verbindung des gekoppelten HMM mit dem LDS ergibt. Störungen werden besser kompensiert und die Erkennungsrate für die Gruppenaktionen daher weniger stark von fehlerhaften Daten beeinflusst.

Das vorgestellte multimodale HMM-LDS ist nicht nur auf die Anwendung von Gruppenaktionen in Besprechungen limitiert, sondern kann auch für andere Fusionsanwendungen, bei denen verschiedene Kanäle gestört sein könnten, eingesetzt werden. Außerdem kann das Modell auch auf weitere parallele Markov-Ketten erweitert werden, so wurden z. B. in einer begleitenden Arbeit [9] bereits drei parallele

Ströme verwendet. Außerdem kann das gekoppelte HMM als treibendes System je nach Anwendung auch durch andere Modelle – z. B. mit asynchroner Modellierung – ersetzt werden, dann allerdings muss die Approximation neu für das Modell hergeleitet werden.

6

Automatischer Konferenzvideoschnitt

In Kapitel 5 wurde gezeigt, wie eine Besprechung in eine Abfolge von Gruppenaktionen unterteilt werden kann. Diese Abfolge dient als eine erste grobe Gliederung der Besprechung und kann z. B. zum Erstellen einer Agenda verwendet werden. In diesem Kapitel wird gezeigt wie Besprechungen in intelligenten Konferenzräumen [109] und Videokonferenzen [136] in eine Abfolge von Kameraeinstellungen unterteilt werden können. Diese Abfolge kann dann direkt genutzt werden um nur den aktiven Kanal an andere Videokonferenzteilnehmer zu übertragen oder Besprechungen effektiv zu archivieren. Beide Anwendungen sollen im Folgenden erläutert werden.

In einer Videokonferenz befinden sich die Teilnehmer an verschiedenen Orten. Üblicherweise wird jeder Teilnehmer mit einer Kamera und einem Mikrophon aufgenommen [136]. Diese audio-visuellen Daten werden dann zu allen anderen Teilnehmern übertragen. Die Übertragung selbst erfolgt entweder über einen zentralen Computer oder direkt von Teilnehmer zu Teilnehmer. Die Audiodaten werden dabei häufig so vorverarbeitet, dass nur aktive Sprecher tatsächlich wiedergegeben und alle anderen Kanäle stummgeschaltet werden. Dieser Vorgang ist vergleichbar mit der Sprecherauswahl in Telefonkonferenzen[1]. Für den visuellen Kanal einer Videokonferenz gibt es hingegen derzeit drei gängige Verfahren:

- Viele Anbieter übertragen und zeigen lediglich das Video des aktuellen Sprechers, alle anderen Videos werden unterdrückt. Für die Auswahl des zu zeigenden Videos wird die Information des akustischen Kanals wiederverwendet[2].

- Andere Anbieter zeigen alle oder eine Auswahl mehrerer Konferenzteilnehmer parallel an. Dazu werden die einzelnen Teilnehmervideos auf eine kleinere Grö-

[1] z. B. Skype (`www.skype.com`), Spiderphone (`www.spiderphone.com`), Vapps (`www.vapps.com`)
[2] z. B. Visual Nexus (`www.visualnexus.com`)

ße skaliert und so in einem Videodatenstrom zusammengefasst, dass alle Teilnehmer gleichzeitig auf den Bildschirm passen[3].

- Einige kommerzielle Videokonferenzbetreiber bieten Systeme und Videokonferenzraumausstattungen, die auf spezieller Hardware basieren. Diese verwenden sehr große Bildschirme oder mehrere Bildschirme, um alle Teilnehmer parallel in hoher Auflösung zu zeigen[4].

Alle drei Möglichkeiten den visuellen Kanal zu zeigen haben jedoch Nachteile: Systeme, die auf mehreren Bildschirmen oder anderer spezieller Hardware basieren, sind üblicherweise sehr teuer, relativ unflexibel und können nicht auf eine beliebige Anzahl von Teilnehmern erweitert werden. Die parallele Anzeige aller Konferenzteilnehmer in einem großen Video ist auf wenige Konferenzteilnehmer beschränkt; bei zu vielen Teilnehmern werden die individuellen Videos zu klein. Lediglich das Video des aktiven Sprechers anzuzeigen, ist die einfachste Variante, bei der auch am wenigsten Daten zu allen Teilnehmern übermittelt werden müssen. Allerdings enthält das Video dann nur relativ wenig Zusatzinformation im Vergleich zu einem reinen Telefonkonferenzsystem: die Information über die Zuhörer – z. B. ein zustimmendes Nicken oder ein ablehnendes Kopfschütteln des Projektleiters – geht auf diese Weise verloren.

Konferenzen und Besprechungen haben hohe multimodale Eigenschaften [3], es kann daher sehr wichtig sein auch Personen zu zeigen, die gerade nicht sprechen. Professionelle Regisseure von Diskussionssendungen blenden z. B. von Zeit zu Zeit vom Redner auf andere Diskussionsteilnehmer oder das Publikum, um visuelle Reaktionen oder Gesten zu zeigen. Ein Videokonferenzsystem sollte daher weder nur den aktuellen Sprecher noch alle Teilnehmer gleichzeitig zeigen. Vielmehr sollte es zu jedem Zeitpunkt einen Teilnehmer auf der Basis von sowohl akustischer als auch visueller Information auswählen und zu den anderen Teilnehmern übertragen.

Auch bei Konferenzen, die in einem intelligenten Konferenzraum stattfinden, tritt ein vergleichbares Problem auf. Hier befinden sich alle Teilnehmer in einem Raum und werden mit Kameras und Mikrophonen aufgenommen (ein solcher Raum wurde bereits in Abschnitt 5.1 erläutert). Die Aufzeichnungen können dann nach (und teilweise auch während) der Besprechung in einem speziellen Meeting Browser [157, 158] zusammen mit automatisch extrahierter Information betrachtet werden. So können Personen, die während der Besprechung nicht anwesend waren, Entscheidungen nachvollziehen. Für Personen, die in der Besprechung waren, kann die Aufzeichnung als Protokoll dienen. Allerdings ist es auch bei diesen Aufzeichnungen nicht sinnvoll, alle aufgezeichneten Videodaten parallel darzustellen – so müssten z. B. für den in Abschnitt 5.1 vorgestellten Konferenzraum sieben Videos gleichzeitig angezeigt werden. Es ist daher auch hier erforderlich für jeden Zeitpunkt der Besprechung aus allen

[3] z. B. Intercall (`www.intercall.com`) oder Visual Nexus (`www.visualnexus.com`)
[4] z. B. Radvision (`www.radvision.com`) oder Polycom (`www.polycom.com`)

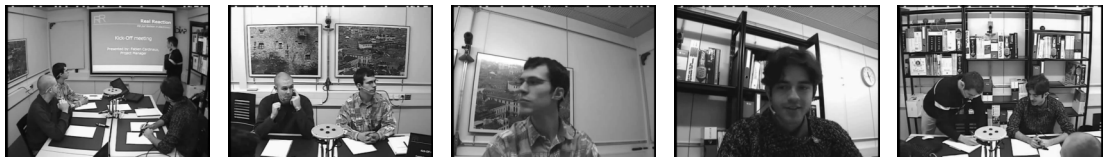

Abbildung 6.1: Beispielhafter Zusammenschnitt einer Konferenz. Über die Zeit wechselt die Kameraperspektive je nach Handlung und Ereignissen, dadurch ergibt sich aus allen verfügbaren Kameras ein einzelnes, geschnittenes Video. In dieser symbolischen Darstellung wechselt die Perspektive nach jedem Frame; in einem wirklichen Zusammenschnitt wären die einzelnen Einstellungen deutlich länger und würden auch in der Länge variieren.

verfügbaren Kameras eine auszuwählen, die die Besprechung zu diesem Zeitpunkt am besten repräsentiert. Die Besprechung wird dann als eine Abfolge von verschiedenen Kameraeinstellungen dargestellt und kann so als ein einzelner Videostrom gespeichert und wiedergegeben werden.

Es ist also sowohl für Videokonferenzen als auch für Besprechungen in einem intelligenten Konferenzraum erforderlich, aus allen verfügbaren Kameras für jeden Zeitpunkt eine einzelne Kamera oder Kameraeinstellung auszuwählen. Diese Einstellung sollte den Inhalt und die Ereignisse der Besprechung am besten repräsentieren. Diese Kameraeinstellung wird dann – im Falle von Videokonferenzen – an alle Teilnehmer übertragen oder – im Falle von aufgezeichneten Besprechungen – archiviert. Das Problem kann daher auch als automatischer, virtueller Konferenzregisseur [6] bezeichnet werden. Während in Kapitel 3 nach vorhandenen Schnittpunkten in Videos gesucht wurde, sollen hier die Schnittpunkte erst vom System festgelegt werden. Abbildung 6.1 zeigt beispielhaft einen solchen Zusammenschnnitt einer Konferenz. Über die Zeit wechselt dabei die Kameraperspektive. In diesem einfachen Beispiel wechselt die Kamera aus Darstellungsgründen nach jedem Frame. In einem wirklichem Zusammenschnitt sind die einzelnen Persepektiven deutlich länger und können in ihrer Länge je nach Handlung der Konferenz und den stattfindenden Ereignissen stark variieren.

Obwohl das Problem des automatischen Videoschnitts kommerziell interessant ist, wurde es bisher noch nicht gründlich erforscht. Vorherige Arbeiten beschäftigen sich vor allem mit festgelegten Regeln zur Kameraauswahl [6, 146] oder zeigen, wie menschliche Kamerabewegungen automatisch gelernt werden können [97]. In [98] wird eine einzelne Kamera mit sehr hoher Auflösung verwendet, um eine Konferenz aufzuzeichnen, das entwickelte System wählt dann automatisch relevante Bereiche der Aufnahme aus. Eine Benutzerstudie mit professionellen Kameraleuten untersucht grundlegende Richtlinien für Aufnahmen [149]. Für die automatische Überwachung bietet [144] ein Verfahren, dass aus mehreren Kameras jeweils eine auswählt; die Entscheidung basiert dabei auf Qualitätsaspekten. Die Resultate dieser Arbeiten

können daher nicht direkt angewendet werden, um automatisch ein Konferenzvideo zusammenzuschneiden.

In dieser Arbeit wird daher vorgeschlagen, die Kameraauswahl als ein Mustererkennungsproblem zu beschreiben. Jede mögliche Kamera oder Kameraeinstellung wird dann als eine Musterklasse beschrieben. Aus vorhandenen Beispieldaten können so GM trainiert werden. Das Problem der Kameraauswahl wird dann auf das automatische Erkennen der Klassen aus den Konferenzdaten reduziert. Auf diese Weise wird das System sehr flexibel und kann leicht auf neue Szenarien angepasst werden: Für einen Konferenzraum mit neuen Kameras oder neuen Einstellungen muss lediglich jeweils eine neue Klasse je neuer Kameraeinstellung trainiert werden, ohne dass die anderen Klassen davon beeinflusst werden. Insbesondere erlaubt dieser Ansatz auch eine multimodale Auswahl der Kamera: Dazu wird das System sowohl mit akustischen als auch mit visuellen Merkmalen trainiert. Im Gegensatz zu regelbasierten Verfahren kann der Videoschnitt damit sehr einfach an neue Modalitäten oder Ereignisse angepasst werden, da diese implizit mittrainiert werden.

Bisher war der Schwerpunkt dieser Arbeit, wie mit GM klassifiziert werden kann. Für die automatische Kameraauswahl ist es jedoch erforderlich, die Daten gleichzeitig zu segmentieren und zu klassifizieren, d. h. die Konferenz automatisch in eine Abfolge von Kameraeinstellungen zu zerlegen. Obwohl die bisher vorgestellten Modelle prinzipiell alle auch zur Segmentierung verwendet werden können – indem im JT-Algorithmus die Summen durch Maximierungsoperationen ersetzt werden –, ist dazu ein relativ starker Eingriff in die Algorithmen erforderlich. Dazu muss die Viterbi-Dekodierung für jedes Modell neu hergeleitet werden (wie z. B. in Abschnitt 4.2.7 für das AHMM). Dadurch geht ein Teil der Flexibilität von GM verloren. In diesem Kapitel wird daher gezeigt, wie GM so erweitert werden können, dass eine automatische Segmentierung mit verschiedenen Modellen vorgenommen werden kann. Dabei ist die in diesem Kapitel gezeigte Segmentierungsstruktur sowie das dafür benötigte Training prinzipiell modellunabhängig. Die in dieser Arbeit gezeigten Dekodierungs- und Trainingsstrukturen können daher auch auf andere GM angewendet werden.

6.1 Konferenzdaten und Merkmale

Die Daten für die automatische Kamerauswahl wurden im AMI Projektes [4] im IDIAP smart meeting room [109] aufgezeichnet und sind öffentlich verfügbar [40]. Der Besprechungsraum wurde bereits in Abbildung 5.1 auf Seite 109 gezeigt und in Abschnitt 5.1 detailliert beschrieben. Der Raum ist mit einem Tisch, einem Whiteboard und einem Projektor mit Leinwand ausgestattet. Wie im M4 Projekt hat auch im AMI Projekt jede Besprechung vier Teilnehmer. Jeder Teilnehmer wird mit einem Kondensator-Headset- und einem omni-direktionalen Ansteckmikrophon aufgenommen. Fernfeldaufnahmen werden mit zwei Mikrophonarrays durchgeführt. Jede Besprechung wird mit sieben Kameras aufgezeichnet: Vier Kameras zeichnen

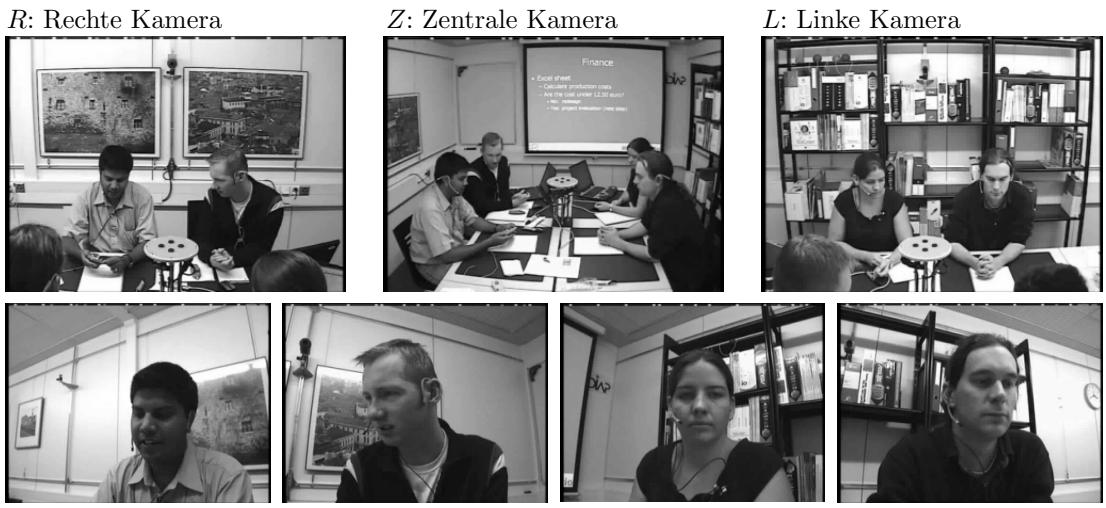

Abbildung 6.2: Beispielbilder aller sieben Kameras im IDIAP Besprechungsraum. Die Kameras sind im AMI Projekt nach ihrer Position im Raum und nicht nach der gezeigten Perspektive bezeichnet. Abbildung 5.1 auf Seite 109 zeigt eine Skizze des Raumes mit den Kamerapositionen und der Platzierung der Teilnehmer um den Tisch.

Nahaufnahmen der Teilnehmer auf $(C_1 - C_4)$, zwei Kameras nehmen den Raum von links (L) und rechts (R) auf und eine zentrale Kamera (Z) zeigt alle vier Teilnehmern zusammen mit dem Whiteboard und der Projektorleinwand. Abbildung 6.2 zeigt Beispielbilder aller sieben im Raum vorhandenen Kameras.

Die AMI Daten sind von den technischen Eigenschaften vergleichbar mit den M4 Daten. Die Besprechungen sind jedoch im AMI Projekt natürlicher, da die Teilnehmer in insgesamt vier aufeinanderfolgenden Besprechungen gemeinsam eine Aufgabe lösen mussten, dafür aber teilweise widersprüchliche Anweisungen hatten. Im Gegensatz zu den gespielten Besprechungen des M4 Projektes kommt es daher in den AMI Daten zu vielen natürlichen Interaktionen der Teilnehmer untereinander.

Für den Videoschnitt werden in dieser Arbeit aus dem Datensatz 24 Videos mit jeweils fünf Minuten Länge und wechselnden Teilnehmern verwendet; davon werden 18 Videos für das Training der Modelle und die verbleibenden sechs unbekannten Videos zum Testen verwendet. Für den automatischen Videoschnitt werden die in Abschnitt 5.3 beschriebenen 42 globalen Bewegungsmerkmale $\vec{o}^{\mathrm{V}}$ extrahiert. Aus den Audioaufnahmen der Headsetmikrophone werden – wie ebenfalls in Abschnitt 5.3 beschrieben – 12 MFCC, die Energie sowie die erste und zweite Ableitung pro Teilnehmer als akustische Merkmale $\vec{o}^{\mathrm{A}}$ extrahiert. Soweit erforderlich werden die beiden Merkmalvektoren aneinander angepasst, indem der visuelle Merkmalvektor auf die Länge des akustischen Vektors interpoliert wird. Eine ausführliche Beschreibung aller Merkmale und einen Vergleich in einem regelbasierten System gibt [6].

6.2 Verfügbare Videomodi

Für den automatischen Videoschnitt muss für jeden Zeitpunkt der Besprechungen (d. h. für jedes Frame) eine Kamera, ein Kameraauschnitt oder eine Perspektive ausgewählt werden. Im Rahmen dieser Arbeit werden die möglichen Kameraeinstellungen als Videomodus V_m bezeichnet, wobei jeder Modus durch eine Klasse repräsentiert wird. Die Menge aller möglichen Videomodi hängt dabei vom Szenario, der vorhandenen Hardware und der benötigten Systemeigenschaft ab. Eine Konferenz kann dann unabhängig vom Szenario – vergleichbar zu den Ereignissen in Abschnitt 5.2 – als eine Sequenz von verschiedenen Videomodi modelliert werden.

Im Falle einer Videokonferenz würden die Kameras der Teilnehmer jeweils einen Modus repräsentieren, diese könnten z. B. mit einem weiteren Modus für Präsentationsfolien ergänzt werden. Für eine Videokonferenz mit vier Teilnehmern ergäbe sich dann im einfachsten Fall ein Szenario mit fünf verschiedenen Modi:

Videomodi 1 - 4: zeigt die Nahaufnahmen $C_1 - C_4$ der Teilnehmer und

Videomodus 5: zeigt die Präsentationsfolie.

Eine solche Videokonferenz stellt daher für das hier vorgestellte System ein fünf Klassenproblem dar. Für das Archivieren von Besprechungen könnte im einfachsten Fall jede Kamera im Raum einen eigenen Modus repräsentieren. Für den IDIAP Besprechungsraum würden sich so sieben verschiedene Modi ergeben:

Videomodi 1 - 4: zeigt die Nahaufnahmen $C_1 - C_4$ der Teilnehmer,

Videomodus 5: zeigt die zentrale Kamera Z mit der Leinwand,

Videomodus 6: zeigt die linke Kamera L mit den Personen P1 und P3 und

Videomodus 7: zeigt die rechte Kamera R mit den Personen P2 und P4.

Eine archivierte Besprechung im IDIAP Raum stellt für das hier vorgestellte Konferenzvideoschnittsystem daher ein Problem mit sieben Klassen dar.

Durch diese Definition kann die physische Erzeugung des Videos von der Generierung der Modussequenz getrennt werden. Sowohl das tatsächliche Zusammenschneiden der Videos, als auch die Generierung der Modussequenz können je nach Anwendung individuell angepasst werden; als Schnittstelle dient immer nur die Abfolge der Videomodi. Während das in dieser Arbeit vorgestellte System die Sequenz der verschiedenen Einstellungen erzeugt, generiert ein weiteres System anhand der Abfolge der Modi dann das zusammengeschnittene Video. Dies kann abhängig vom Szenario zentral auf einem Server oder z. B. individuell bei den Teilnehmern erfolgen. Im Falle von archivierten Besprechungen ist es auch möglich, alle aufgzeichneten Videos zu speichern und erst beim Abrufen durch den Browser den Zusammenschnitt

anhand der Modussequenz durchzuführen, dann reduziert sich der physische Zusammenschnitt auf ein Umschalten zwischen verschiedenen Videos[5].

Das vorgeschlagene System ist nicht auf die beiden vorgestellten Szenarien bzw. den IDIAP Besprechungsraum limitiert. Neue Modi oder vollständig neue Szenarien können ergänzt werden: Benötigt man z. B. für Diskussionen eine Einstellung, bei der eine Person in die Ecke des Bildes einer anderen Person eingeblendet wird (die Korrespondenteneinstellung in vielen Nachrichtensendungen), würde dies als ein neuer Modus definiert werden. Nicht-Diskriminatives-Training vorausgesetzt, müsste dafür lediglich eine neue Klasse trainiert werden, ohne dass die anderen bereits vorhanden Klassen beeinflusst werden. Auf diese Weise kann das System leicht auf verschiedene Szenarien und Anwendungen adaptiert werden, ohne dass der zugrundeliegende Erkennungsprozess verändert werden muss.

6.3 Graphische Modelle zur Segmentierung

Im Rahmen dieser Arbeit wurden mit GM bisher Merkmalsequenzen klassifziert. Es wurde also die Klasse k mit den Parametern λ_k gesucht, die die Observierung $\vec{o}$ am wahrscheinlichsten erzeugt hat:

$$k^* = \underset{k \in K}{\operatorname{argmax}}\, p(\vec{o}_1, \ldots, \vec{o}_T \mid \lambda_k). \tag{6.1}$$

Für den automatischen Videoschnitt werden jedoch Modelle benötigt, die die Merkmalsequenz automatisch segmentieren und klassifizieren. Dabei soll einer Beobachtung $\vec{o}$ eine Sequenz von Klassen k mit zeitlichen Klassengrenzen zugeordnet werden. In Anlehnung an Gleichung (2.35) wird also mit

$$\{k_1^*, \ldots, k_T^*\} = \underset{k_1,\ldots,k_T}{\operatorname{argmax}}\, p(\vec{o}_1, \ldots, \vec{o}_T \mid \lambda_{k_1}, \ldots, \lambda_{k_T}) \tag{6.2}$$

die Abfolge von Klassen gesucht, die die Observierung am wahrscheinlichsten erzeugt hat.

Die Grundlagen für eine solche Segmentierung mit GM sowie einige wichtige Modelle werden in [29] für die Spracherkennung vorgestellt. Dabei werden insbesondere die Eigenschaften von [30] genutzt. Für das Segmentieren wird die Nachrichtenpropagation in Verbundbäumen (Abschnitt 2.8.3) modifiziert: Alle Summen werden durch Maximierungen ersetzt. Für jeden Knoten im GM wird damit ausschließlich die wahrscheinlichste Konfiguration bestimmt und weiterpropagiert[6]. Dies wird auch

[5] In dieser Form ist das in dieser Arbeit entwickelte System für den JFerret Browser [157] implementiert.

[6] Es ist theoretisch auch möglich, nur für den Klassenknoten – also die Schicht in der segmentiert wird – die Summen durch Maximierungen zu ersetzen, aber für alle anderen Schichten die Summe über alle Wahrscheinlichkeiten beizubehalten. Dies wird jedoch praktisch nicht angewendet.

als Viterbi-Dekodierung bezeichnet. Prinzipiell kann eine so modifzierte Nachrichtenpropagation auf jedes GM angewendet werden, allerdings ist der direkte Bezug von Knoten zu Klassen dann noch nicht gegeben. Jedes Modell müßte daher individuell sowohl an das Problem als auch die Klassen angepasst werden; das widerspricht jedoch dem Grundprinzip der GM.

Im Rahmen dieser Arbeit sollen daher aufbauend auf [29] generelle Strukturen für die Segmentierung vorgestellt werden. Diese stellen selbst wiederum GM da und können daher in Kombination mit anderen GM – ohne Modifikation der Algorithmen – sowohl für das Training als auch die Dekodierung verwendet werden.

6.3.1 Lineare Grundstruktur

Das einfachste Modell zur integrierten Segmentierung und Klassifizierung ist [29] entnommen und in Abbildung 6.3 gezeigt. Ein Zeitschlitz dieses Modells besteht aus der aktuellen Beobachtung $\vec{o}_t$, einem gemeinsamen Zustandspool q_t, der Übergangswahrscheinlichkeit a_t, dem aktuellen Zustandszeiger q_t^k, der Klassenwechselwahrscheinlichkeit w_t und der aktuellen Klasse k_t. Der Graph faktorisiert die Verbundwahrscheinlichkeit dann zu

$$
\begin{aligned}
&p(\vec{o}_1, \ldots, \vec{o}_T, q_1, \ldots, q_T, a_1, \ldots a_T, q_t^k, \ldots, q_T^k, w_1, \ldots w_T, k_1, \ldots, k_T) = \\
&p(\vec{o}_1 \,|\, q_1)\; f(w_1 \,|\, q_1^k, a_1)\; p(a_1 \,|\, q_1)\; f(q_1 \,|\, q_1^k, k_1)\; f(q_1^k)\; p(k_1) \qquad (6.3)\\
&\prod_{t=2}^{T-1} p(\vec{o}_t \,|\, q_t)\; f(w_t \,|\, q_t^k, a_t)\; p(a_t \,|\, q_t)\; f(q_t \,|\, q_t^k, k_t)\; f(q_t^k \,|\, a_{t-1}, q_{t-1}^k, w_{t-1})\; p(k_t \,|\, k_{t-1}, w_{t-1}) \\
&p(\vec{o}_T \,|\, q_T)\, f(w_T = 1 \,|\, q_T^k, a_T)\, p(a_T \,|\, q_T)\, f(q_T \,|\, q_T^k, k_T)\, f(q_T^k \,|\, a_{T-1}, q_{T-1}^k, w_{T-1}) \\
&p(k_T \,|\, k_{T-1}, w_{T-1}),
\end{aligned}
$$

wobei die zweite Zeile den Prolog, die dritte Zeile die Zeitschlitze und die letzten beiden Zeilen den Epilog beschreiben. Diese Wahrscheinlichkeit kann durch Umformung auch vereinfacht als

$$
\begin{aligned}
p(\vec{o}_1, \ldots, &\vec{o}_T, q_1, \ldots, q_T, a_1, \ldots a_T, q_1^k, \ldots, q_T^k, w_1, \ldots w_T, k_1, \ldots, k_T) = \\
&\prod_{t=1}^{T} p(\vec{o}_t \,|\, q_t)\; f(q_t \,|\, q_t^k, k_t)\; p(a_t \,|\, q_t) \\
&f(q_1^k)\; p(k_1) \prod_{t=2}^{T} f(q_t^k \,|\, a_{t-1}, q_{t-1}^k, w_{t-1})\; p(k_t \,|\, k_{t-1}, w_{t-1}) \qquad (6.4)\\
&f(w_T = 1 \,|\, q_T^k, a_T) \prod_{t=1}^{T-1} f(w_t \,|\, q_t^k, a_t)
\end{aligned}
$$

ausgedrückt werden – dabei ist jedoch der direkte Bezug zwischen dem Prolog, den Zeitschlitzen und dem Epilog nicht mehr sichtbar.

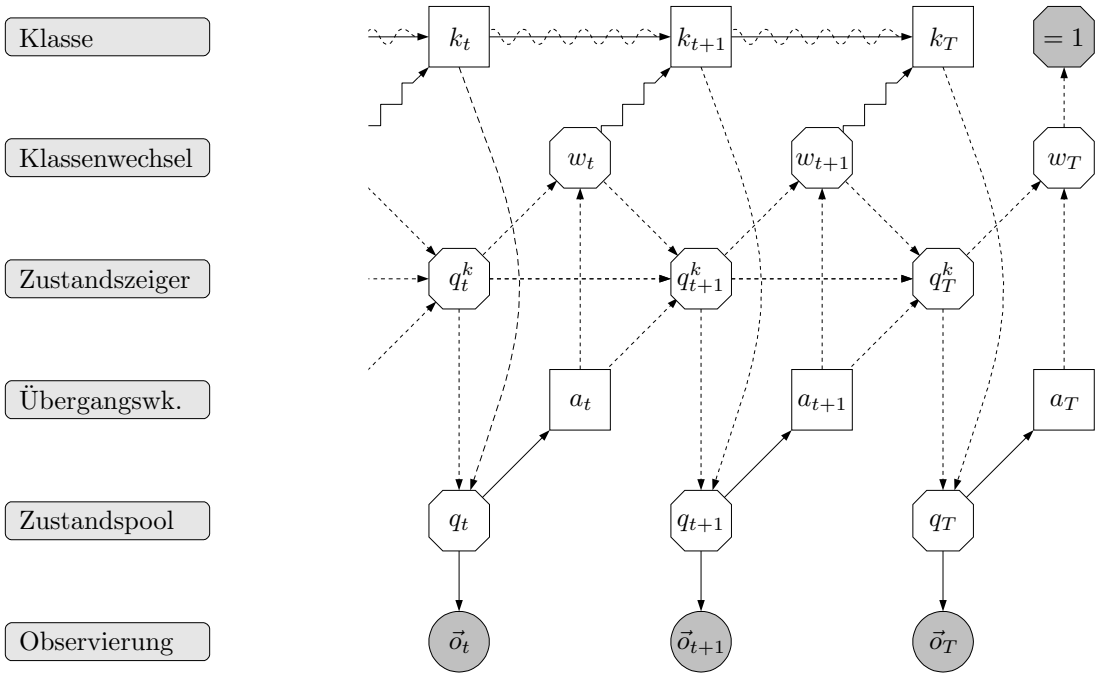

Abbildung 6.3: Lineare Dekodierungsstruktur für integriertes Segmentieren und Klassifizieren. Das Modell trennt die Zustände und damit die aktuelle Beobachtung von der Klasse. Ein gemeinsamer Pool von Zuständen wird genutzt, um die Beobachtung zu modellieren. Ein Wechsel des Zustands erfolgt über die Übergangswahrscheinlichkeit; der aktuelle Zustand ist im Zustandszeiger abgelegt. Darauf aufbauend wird die Klasse modelliert. Nur in bestimmten Zuständen kann diese wechseln und damit eine Segmentierung stattfinden. Die Verbindung zwischen dem aktuellem Zustand und der Klasse wird ausschließlich über den Klassenwechselknoten hergestellt.

6.3.2 Modellparameter

In Gleichung (6.3) und (6.4) stellen $p(\cdot)$ Wahrscheinlichkeits- und $f(\cdot)$ deterministische Beziehungen zwischen den Knoten da. Diese Beziehungen der Knoten zueinander sollen im Folgenden erläutert werden. Das lineare Grundmodell zur Segmentierung wird mit ingesamt sechs Verteilungen und deterministischen Funktionsknoten beschrieben:

Observierung: modelliert die Wahrscheinlichkeit der Observierung $p(\vec{o}_t \,|\, q_t)$ in Abhängigkeit des Zustands. In dem linearen Grundmodel in Abbildung 6.3 ist diese Wahrscheinlichkeit mit einem Gaußknoten modelliert, d. h. $p(\vec{o}_t \,|\, q_t) = \mathcal{N}(\vec{o}_t, \vec{u}_{q_t}, \underset{\sim}{\Sigma}_{q_t})$. Je nach Anwedung kann die Beobachtung aber auch mit einem diskreten Knoten, einem GMM oder einer der anderen in Abbildung 2.16 auf Seite 49 gezeigten Emissionsformen modelliert werden.

Zustandspool: enthält alle möglichen Zustände des Modells. Werden zwischen den Klassen keine Zustände geteilt, enthält der Pool bei K möglichen Klassen mit jeweils N Zuständen insgesamt $N \times K$ Zustände. Es ist jedoch auch möglich, dass sich mehrere Klassen einen Zustand teilen, dann enthält der Pool entsprechend weniger Zustände. Dies kann z. B. genutzt werden, wenn in mehreren Videomodi „Stille“ auftritt, diese wird dann mit nur einem Zustand modelliert werden, der von allen Klassen genutzt wird. Der Zustandspool selbst ist eine deterministische Funktion $f(q_t \,|\, q_t^k, k_t)$, die abhängig vom Zustandszeiger q_t^k und der augenblicklichen Klasse k_t einen der Zustände aus dem Pool auswählt. Die Zuordnung vom Zustandszeiger und der Klasse auf den Zustand q_t des Pools wird nicht gelernt, sondern bei der Modellbildung festgelegt. Im einfachsten Fall, d. h. wenn keine Zustände geteilt werden, wird jeder Zustand des Pools von genau einer Kombination von Klasse k und Zustandszeiger q_t^k ausgewählt. Teilen sich mehrere Klassen einen Zustand, dann verweisen mehrere Kombinationen von k und q_t^k auf den gleichen Zustand im Pool.

Übergangswahrscheinlichkeit: modelliert für jeden Zustand aus dem Pool die Übergangswahrscheinlichkeit $p(a_t \,|\, q_t)$. Da es sich um ein lineares Modell handelt, gibt es hier keine Übergangswahrscheinlichkeiten zu verschiedenen Zuständen, d. h. das Modell kann in einem Zeitschritt nur in dem vorherigen Zustand bleiben, d. h. $p(a_t = 0 \,|\, q_t)$, oder zum nächsten Zustand springen, d. h. $p(a_t = 1 \,|\, q_t)$. Zustandsrücksprünge oder das Überspringen eines Zustandes sind mit diesem Modell nicht möglich. Die Wahrscheinlichkeit $p(a_t = 0 \,|\, q_t)$ kann daher als eine Aussprungswahrscheinlichkeit bzw. als eine Verweildauer im Zustand q_t angesehen werden. Eine weitere Verarbeitung der Aussprungswahrscheinlichkeit, also der tatsächliche Zustands- oder in bestimmten Fällen Klassenwechsel, findet erst im Zustandszeiger und Klassenwechselknoten statt.

Zustandszeiger: modelliert den aktuellen Zustand des Modells. Im Prolog wird zum ersten Zeitpunkt $t = 1$ unabhängig von der Klasse immer in den ersten Zustand eingesprungen, d. h. $f(q_1^k = 1), \forall k$. In allen folgenden Zeitschritten wird abhängig vom vorherigen Zustand q_{t-1}^k, vom Klassenwechselknoten w_{t-1} und der Übergangswahrscheinlichkeit a_{t-1} deterministisch der Zustandszeiger q_t^k neu gesetzt, d. h. $f(q_t^k \,|\, a_{t-1}, q_{t-1}^k, w_{t-1})$. Findet ein Wortwechsel statt, d. h. $w_{t-1} = 1$ wird unabhängig von der Klasse immer in den ersten Zustand gesprungen, d. h. $f(q_t^k = 1 \,|\, a_{t-1}, q_{t-1}^k, w_{t-1} = 1)$. Findet kein Wortwechsel statt, d. h. $w_{t-1} = 0$ wird geprüft ob ein Zustandswechsel stattfindet. Findet ein Zustandswechsel statt, d. h. $a_{t-1} = 1$, dann wird der Zustandszeiger inkrementiert, also $f(q_t^k = i \,|\, a_{t-1} = 0, q_{t-1}^k = i - 1, w_{t-1} = 0)$. Findet kein Zustandswechsel statt, d. h. $a_{t-1} = 0$, wird der vorherige Zustand beibehalten, also $f(q_t^k = i \,|\, a_{t-1} = 0, q_{t-1}^k = i, w_{t-1} = 0)$.

Klassenwechsel: modelliert in Abhängigkeit vom Zustandszeiger und der Über-

gangswahrscheinlichkeit deterministisch, ob ein Klassenwechsel stattfindet, also $f(w_t \mid q_t^k, a_t)$. Ein Klassenwechsel kann nur auftreten, wenn der Zustandszeiger q_t^k auf den letzten Zustand N der Klasse k zeigt. Für alle anderen Zustandspositionen kann kein Klassenwechsel stattfinden, d. h. $f(w_t = 0 \mid q_t^k = i, a_t), \forall i \neq N$. Zeigt der Zustandszeiger auf den letzten Zustand, also $q_t^k = N$, kann ein Klassenwechsel stattfinden. Die Klassenwechselwahrscheinlichkeit entspricht dann der Aussprungswahrscheinlichkeit aus dem letzten Zustand und wird daher direkt aus a_t übernommen, d. h. $f(w_t = 1 \mid q_t^k = N, a_t = 1) = p(a_t = 1 \mid q_t)$. Im Epilog wird mit dem deterministisch angehängten Kindknoten sichergestellt, dass $f(w_T = 1 \mid q_T^k, a_T)$. Dadurch kann eine Merkmalsequenz nur mit einer abgeschlossenen Klasse enden. Für Probleme, bei denen eine Sequenz auch mitten in einer Klassen enden kann (wenn z. B. die Merkmalsequenz nur einen Ausschnitt aus einer längeren Sequenz darstellt), würde dieser zusätzliche Knoten und damit die Festlegung auf eine abgeschlossene Klasse entfallen.

Klassenwahrscheinlichkeit: modelliert mit einem wechselnden Knoten die aktuelle Klasse. Der Klassenwechselknoten w_{t-1} ist dabei das bestimmende Elternteil. Liegt kein Klassenwechsel vor, d. h. $w_{t-1} = 0$, wird deterministisch die Klasse aus dem vorherigen Zeitschritt beibehalten, also $f(k_t = i \mid k_{t-1} = i, w_{t-1} = 0)$. Liegt jedoch ein Klassenwechsel vor, d. h. $w_{t-1} = 1$ wechselt der Klassenknoten sein Verhalten: Die nächste Klasse wird dann anhand einer Klassenbigrammverteilung $p(k_t = i \mid k_{t-1} = j, w_{t-1} = 1)$ ausgewählt. Die erste Klasse der Sequenz wird im Prolog mit einer Unigrammklassenverteilung $p(k_1 = i)$ ausgewählt.

Obwohl das Modell in Abbildung 6.3 mehr Faktorisierungen enthält, kann es – solange keine Zustände geteilt werden und die Emission mit einer Gaußkurve oder einem GMM modelliert wird – in ausgerollter Form als ein lineares HMM mit $N \times K$ Zuständen dargestellt werden. Durch das Einfügen einer zusätzlichen Kante zwischen der Klasse k_t und dem Klassenwechsel w_t kann das Modell für jede Klasse k eine unterschiedliche Anzahl von Zuständen N_k verwenden. Der Zustandspool enthält dann – ohne Zustandsteilung – insgesamt $N = \sum_{k=1}^{K} N_k$ Zustände. Dadurch können bestimmte Klassen mit weniger oder mehr Zuständen modelliert werden. Dies wird z. B. in [29] genutzt, um die in der automatischen Spracherkennung genutzten Klassen „Kurze Pause“ (*short pause, sp*) und „Stille“ (*silence, sil*) getrennt zu modellieren.

Der Hauptnachteil des in Abbildung 6.3 gezeigten Modells ist die lineare Informationsverarbeitung, die die Modellierung von Merkmalsequenzen weniger flexibel als ergodische oder links-rechts-Modelle macht. Da sich der Klassenwechsel w_t und der Zustandszeiger q_t^k den Übergangsknoten a_t teilen, muss dieser gleichzeitig die Zustandsübergangs- und die Klassenaussprungswahrscheinlichkeit repräsentieren. Es ist daher in diesem GM ausschließlich möglich, die Klasse zu wechseln, wenn das Modell im letzten Zustand der Klasse ist.

6.3.3 Links-rechts- und ergodische Modellerweiterung

Für einige Anwedungen sind links-rechts-Modelle, bei denen von einem Zustand in jeden anderen höheren Zustand gewechselt werden kann, oder ergodische Modelle, bei denen von jedem Zustand in jeden anderen Zustand gewechselt werden kann, besser geeignet. Für einige Anwendungen kann es auch sinnvoll sein, dass ein Klassenwechsel aus jedem – und nicht nur dem letzen – Zustand der Klasse möglich ist. Beides ist mit dem linearen Modell nicht möglich.

Im Rahmen dieser Arbeit wird das in [29] vorgestellte Modell daher so erweitert, dass auch links-rechts- und ergodische Zustandsübergänge möglich sind. Das erweiterte Modell ist in Abbildung 6.4 gezeigt. Der Knoten mit der Übergangswahrscheinlichkeit a_t wird um einen Knoten für die Klassenaussprungswahrscheinlichkeit e_t erweitert. Der Knoten a_t repräsentiert dann nur noch die Zustandsübergangswahrscheinlichkeiten und nicht mehr die Klassenaussprungswahrscheinlichkeit. Für eine ergodisches Modell ist die Matrix in a_t vollbesetzt und im Falle eines links-rechts-Modells hat a_t eine obere Dreiecksform. Für den Epilog wird die Zustandsübergangswahrscheinlichkeit a_t nicht mehr benötigt, da das Modell den Zustand nach dem letzten Zeitschritt nicht mehr wechselt.

Getrennt von den Zuständen wird die Klassenaussprungswahrscheinlichkeit e_t mit einem eigenen Knoten modelliert. Dieser gibt an, mit welcher Wahrscheinlichkeit aus einem Zustand in eine neue Klasse gesprungen wird. Nicht mehr benötigt wird für das erweiterte Modell die deterministische Verbindung zwischen den Zustandszeigern zweier aufeinanderfolgender Zeitschlitze, also q^k_{t-1} und q^k_t. Der Zustand zum Zeitpunkt t wird über die Zustandsübergangswahrscheinlichkeit a_t bestimmt. Bei dem linearen Modell wird im Falle eines Zustandswechsels der vorherige Zustand inkrementiert; dafür ist die Verbindung zwischen den Zustandszeigern erforderlich. Für das erweiterte Modell enthält a_t bereits die Information, in welchen Zustand gesprungen wird, d. h. q^k_t benötigt keine Information vom vorherigen Zustandszeiger. Das Modell faktorisiert die Verbundwahrscheinlichkeit zu

$$
\begin{aligned}
p(\vec{o}_1,\ldots,\vec{o}_T, q_1,\ldots,q_T, a_1,\ldots a_T, e_1,\ldots,e_T, q^k_1,\ldots,q^k_T, w_1,\ldots w_T, k_1,\ldots,k_T) = \\
&\left(f(w_T = 1 \mid q^k_T, e_T) \prod_{t=1}^{T-1} f(w_t \mid q^k_t, e_t) \right) \\
&\left(\prod_{t=1}^{T} p(\vec{o}_t \mid q_t)\ f(q_t \mid q^k_t, k_t)\ p(a_t \mid q_t)\ p(e_t \mid q^k_t) \right) \\
&f(q^k_1)\ p(k_1) \prod_{t=2}^{T} f(q^k_t \mid a_{t-1}, w_{t-1})\ p(k_t \mid k_{t-1}, w_{t-1}),
\end{aligned}
\tag{6.5}
$$

mit den gleichen Modellparametern wie in Abschnitt 6.3.2. Lediglich die Übergangswahrscheinlichkeit a_t und der Zustandszeiger q^k_t müssen modifiziert und der Klassenaussprungsknoten e_t hinzugefügt werden:

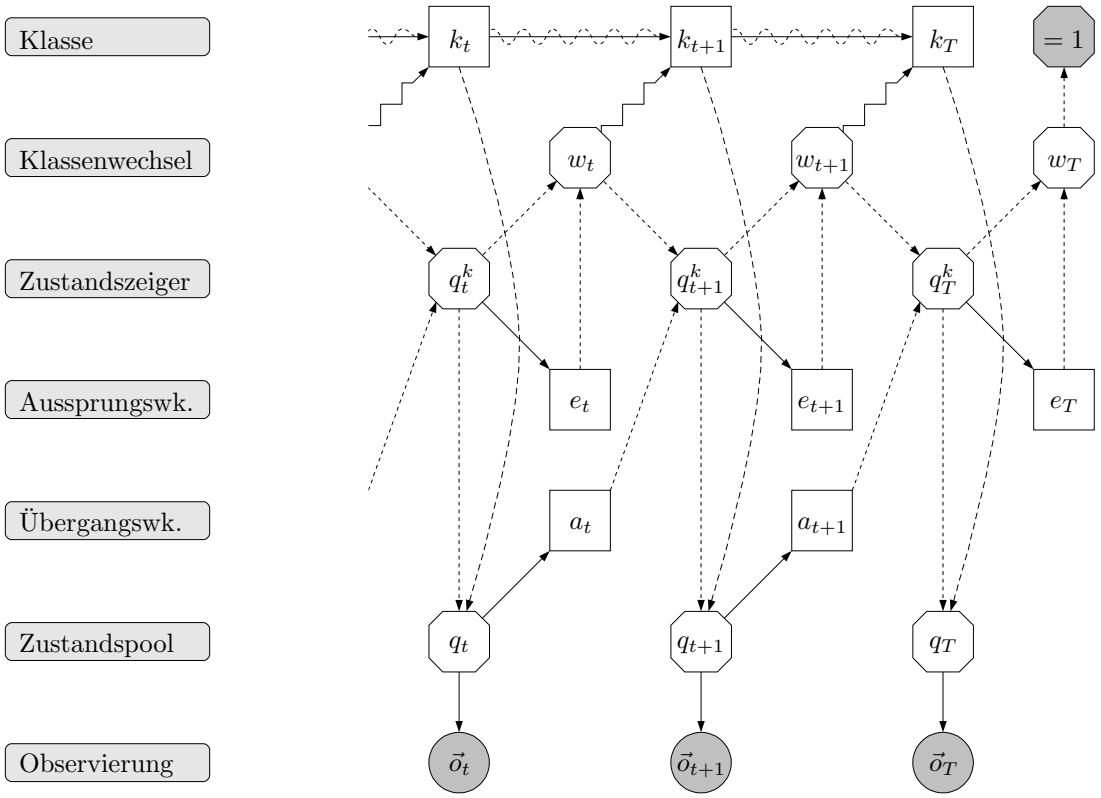

Abbildung 6.4: links-rechts- bzw. ergodische Dekodierungsstruktur für integriertes Segmentieren und Klassifizieren. Im Vergleich zur linearen Dekodierungsstruktur in Abbildung 6.3 wird die Zustands- und die Klassenaussprungswahrscheinlichkeit unabhängig voneinander in eigenen Knoten a_t und e_t modelliert. Dadurch ist es möglich, den Zustandszeiger nicht nur zu inkrementieren, sondern in jeden Zustand der selben Klasse zu springen. Die Verbindung zwischen q_{t-1}^k und q_t^k wird nicht mehr benötigt, da sich der neue Zustand q_t^k aus der Übergangswahrscheinlichkeit a_t ergibt. Außerdem ist es möglich nicht nur aus dem letzen Zustand heraus die Klasse zu wechseln, sondern für jeden Zustand getrennt eine Aussprungswahrscheinlichkeit zu modellieren.

Übergangswahrscheinlichkeit: modelliert für jeden Zustand aus dem Zustandspool die Wahrscheinlichkeit $p(a_t \mid q_t)$, dass im gleichen Zustand geblieben wird bzw. mit welcher Wahrscheinlichkeit in einen anderen Zustand der selben Klasse gesprungen wird. Hat $p(a_t \mid q_t)$ eine obere Dreiecksform, handelt es sich um ein links-rechts-Modell, d. h. von einem Zustand einer Klasse kann nur in einen höheren Zustand gesprungen werden. Ist $p(a_t \mid q_t)$ voll besetzt, handelt es sich um ein ergodisches Modell, bei dem von jedem Zustand jeder andere Zustand erreicht werden kann. Im Epilog wird der Knoten a_t nicht mehr benötigt, da das Modell nach dem letzten Zeitpunkt den Zustand nicht mehr wechselt.

Aussprungswahrscheinlichkeit: modelliert für jeden Zustand die binäre Wahrscheinlichkeit $p(e_t \mid q_t^k)$, dass aus der aktuellen Klasse ausgesprungen wird. Werden alle $p(e_t = 1 \mid q_t^k = i) = 0, \forall i \neq N$ und $p(e_t = 1 \mid q_t^k = N) > 0$ gesetzt, kann eine Klasse wieder nur im letzten Zustand verlassen werden. Hier ist die Aussprungswahrscheinlichkeit nur abhängig vom Zustand innerhalb der Klasse, nicht aber von der Klasse. Durch eine Verbindung von k_t zu e_t kann die Wahrscheinlichkeit zusätzlich abhängig von der Klasse modelliert werden.

Zustandszeiger: modelliert den aktuellen Zustand. Der Zustandszeiger wählt deterministisch einen Zustand aus dem Pool aus, d. h. $f(q_t^k \mid a_{t-1}, w_{t-1})$. Im Prolog wird immer in den ersten Zustand $f(q_1^k = 1), \forall k$ eingesprungen. Durch einen zusätzlichen Wahrscheinlichkeitsknoten könnte dies noch um eine Einsprungwahrscheinlichkeit erweitert werden. In allen weiteren Zeitschritten ist die Auswahl – im Vergleich zum linearen Modell – nicht mehr abhängig vom vorherigen Zustand, sondern nur noch vom Klassenwechselknoten w_{t-1} und vom Zustandsübergangsknoten a_{t-1}. Findet ein Klassenwechsel statt, d. h. $w_{t-1} = 1$ wird unabhängig von der neuen Klasse immer der erste Zustand ausgewählt, d. h. $f(q_t^k = 1 \mid a_{t-1}, w_{t-1} = 1)$, auch dies könnte jedoch noch um einem zusätzlichen Einsprungswahrscheinlichkeitsknoten erweitert werden. Findet kein Klassenwechsel statt, d. h. $w_{t-1} = 0$ wird der neue Zustand anhand der Wahrscheinlichkeit in a_{t-1} bestimmt, d. h. $f(q_t^k = i \mid a_{t-1}, w_{t-1} = 0) = p(a_{t-1} = i \mid q_{t-1})$.

Durch eine zusätzliche Verbindung zwischen k_t und w_t kann das Modell auch eine unterschiedliche Anzahl von Zuständen N_k für jede Klasse modellieren. Durch eine zusätzliche Verbindung zwischen k_t und e_t können für unterschiedliche Klassen auch unterschiedliche Klassenaussprungswahrscheinlichkeiten modelliert werden. Durch das Einfügen eines zusätzlichen Wahrscheinlichkeitsknotens mit einer Verbindung zum Zustandszeiger kann das Modell um eine Zustandseinsprungswahrscheinlichkeit erweitert werden; dann ist es im Prolog und bei einem Klassenwechsel nicht mehr zwingend erforderlich in den ersten Zustand einzuspringen. Wird noch eine zusätzliche Verbindung zwischen der Klasse und dem Einsprungsknoten hinzugefügt, wird die Einsprungswahrscheinlichkeit auch in Abhängigkeit von der Klasse modelliert.

6.3.4 Multi-Stream und gekoppelte Modellerweiterung

Für die multimodale Fusion mehrerer Merkmalströme kann das Modell auch auf ein Multi-Stream bzw. ein gekoppeltes Modell erweitert werden. Das Modell mit zwei parallelen Strömen ist in Abbildung 6.5 gezeigt. Das Modell ist aus einem Audio- (A) und einem Video-Untermodell (V) zusammengesetzt, die jeweils identisch zu dem Modell in Abbildung 6.3 sind. Beide Merkmalströme werden separat verarbeitet, die Vereinigung findet erst auf Klassenebene statt. Dazu gibt es einen gemeinsamen Klassenknoten k_t und einen gemeinsamen Klassenübergangsknoten w_t. Der Klassenknoten k_t ist wiederum identisch zu dem Modell in Abbildung 6.3.

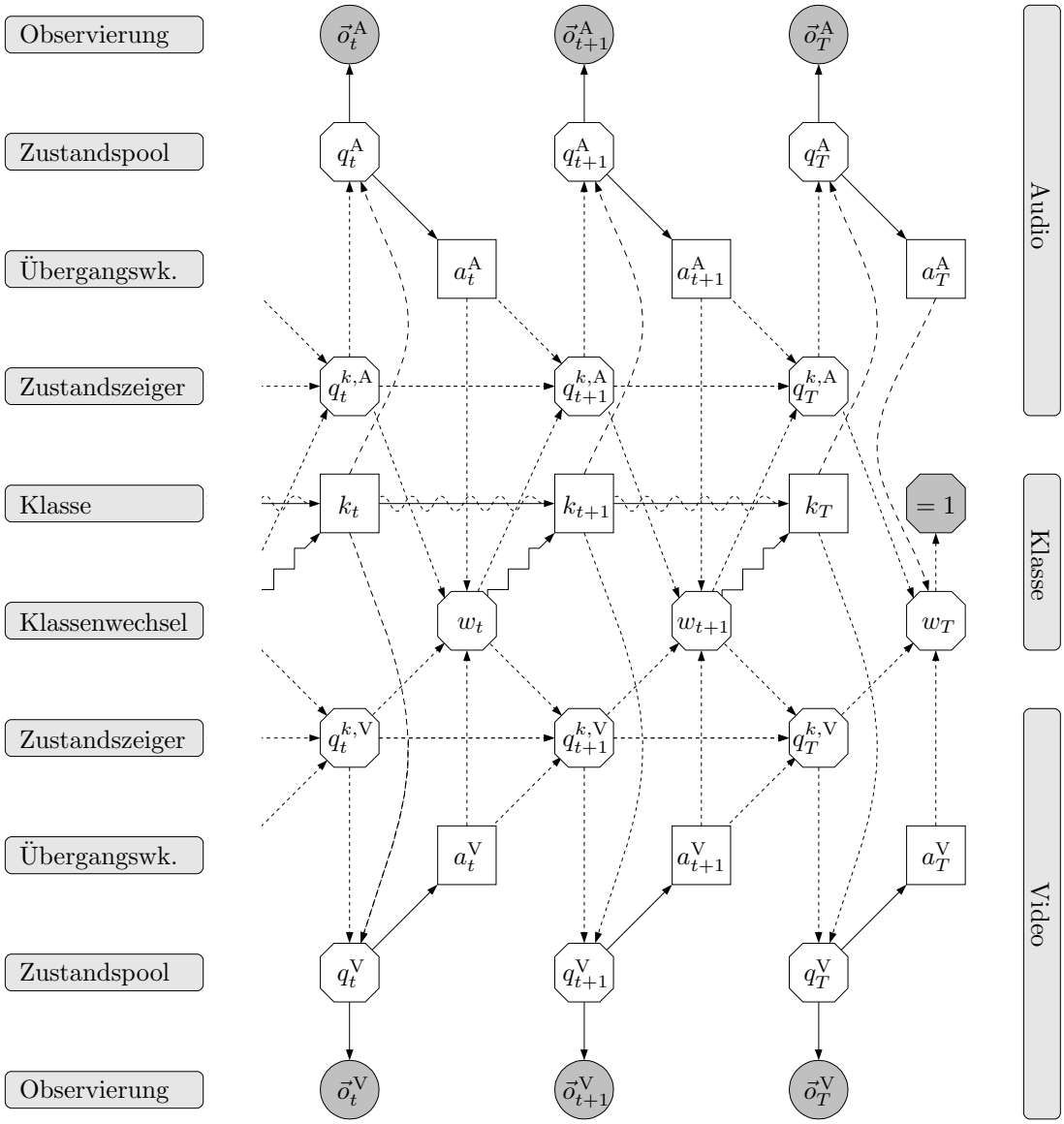

Abbildung 6.5: Dekodierungsstruktur für integriertes Segmentieren und Klassifizieren mit einem Audio- und einem Video-Merkmalstrom. Die Struktur setzt sich aus zwei linearen Untermodellen zusammen, die unabhängig voneinander jeweils eine Beobachtung modellieren. Das Modell ist semi-asynchron, da sich die Zustände der beiden Untermodelle unterschiedlich voneinander entwickeln können. Synchronisationspunkt ist die Klassenebene. Ein Klassenwechsel kann nur dann stattfinden, wenn beide Untermodelle ihn zulassen. Durch Einfügen einer Überkreuzverbindung $q_t^{k,\mathrm{A}} \rightarrow a_t^{\mathrm{V}}$ und $q_t^{k,\mathrm{V}} \rightarrow a_t^{\mathrm{A}}$ werden die beiden Ströme gekoppelt. Sie können sich dann immer noch mit unterschiedlichen Geschwindigkeiten entwickeln, sind dann jedoch stärker aneinander gebunden.

Lediglich der Klassenwechselknoten w_t hat – im Vergleich zu dem Modell mit einem Strom – ein verändertes Verhalten: Dieser hängt sowohl vom Audio- als auch vom Videostrom ab: $f(w_t \,|\, q_t^{k,\mathrm{V}}, q_t^{k,\mathrm{A}}, a_t^{\mathrm{V}}, a_t^{\mathrm{A}})$. Ein Klassenwechsel findet nur statt, wenn beide Untermodelle im letzten Zustand sind, d. h. $q_t^{k,\mathrm{A}} = N^{\mathrm{A}}$ und $q_t^{k,\mathrm{V}} = N^{\mathrm{V}}$. Dann ergibt sich die Klassenaussprungswahrscheinlichkeit aus dem Produkt der beiden einzelnen Aussprungswahrscheinlichkeiten a_t^{V} und a_t^{A}, d. h. $f(w_t = 1 \,|\, q_t^{k,\mathrm{V}} = N^{\mathrm{V}}, q_t^{k,\mathrm{A}} = N^{\mathrm{A}}, a_t^{\mathrm{V}}, a_t^{\mathrm{A}}) = p(a_t^{\mathrm{V}} = 1 \,|\, q_t^{\mathrm{V}} = N^{\mathrm{V}})\; p(a_t^{\mathrm{A}} = 1 \,|\, q_t^{\mathrm{A}} = N^{\mathrm{A}})$.

Alle anderen Parameter des Modells sind analog zu dem linearen Modell in Abschnitt 6.3.2, aber doppelt ausgelegt (jeweils einmal für das Audio- und das Video-Untermodell). Es ergibt sich dann für die Gesamtverbundwahrscheinlichkeit die Faktorisierung

$$
\begin{aligned}
& p(\vec{o}_1^{\,\mathrm{A}}, \ldots, \vec{o}_T^{\,\mathrm{A}}, q_1^{\mathrm{A}}, \ldots, q_T^{\mathrm{A}}, a_1^{\mathrm{A}}, \ldots a_T^{\mathrm{A}}, q_1^{k,\mathrm{A}}, \ldots, q_T^{k,\mathrm{A}}, \\
& \quad \vec{o}_1^{\,\mathrm{V}}, \ldots, \vec{o}_T^{\,\mathrm{V}}, q_1^{\mathrm{V}}, \ldots, q_T^{\mathrm{V}}, a_1^{\mathrm{V}}, \ldots a_T^{\mathrm{V}}, q_1^{k,\mathrm{V}}, \ldots, q_T^{k,\mathrm{V}}, w_1, \ldots w_T, k_1, \ldots, k_T) = \\
& \quad \prod_{t=1}^{T} p(\vec{o}_t^{\,\mathrm{V}} \,|\, q_t^{\mathrm{V}})\; f(q_t^{\mathrm{V}} \,|\, q_t^{k,\mathrm{V}}, k_t)\; p(a_t^{\mathrm{V}} \,|\, q_t^{\mathrm{V}})\; p(\vec{o}_t^{\,\mathrm{A}} \,|\, q_t^{\mathrm{A}})\; f(q_t^{\mathrm{A}} \,|\, q_t^{k,\mathrm{A}}, k_t)\; p(a_t^{\mathrm{A}} \,|\, q_t^{\mathrm{A}}) \\
& \quad f(q_1^{k,\mathrm{V}})\; f(q_1^{k,\mathrm{A}})\; p(k_1) \qquad (6.6) \\
& \quad \prod_{t=2}^{T} f(q_t^{k,\mathrm{V}} \,|\, a_{t-1}^{\mathrm{V}}, q_{t-1}^{k,\mathrm{V}}, w_{t-1})\; f(q_t^{k,\mathrm{A}} \,|\, a_{t-1}^{\mathrm{A}}, q_{t-1}^{k,\mathrm{A}}, w_{t-1})\; p(k_t \,|\, k_{t-1}, w_{t-1}) \\
& \quad f(w_T = 1 \,|\, q_T^{k,\mathrm{V}}, q_T^{k,\mathrm{A}}, a_T^{\mathrm{V}}, a_T^{\mathrm{A}}) \prod_{t=1}^{T-1} f(w_t \,|\, q_t^{k,\mathrm{V}}, q_t^{k,\mathrm{A}}, a_t^{\mathrm{V}}, a_t^{\mathrm{A}}).
\end{aligned}
$$

Das Modell wird auch als semi-asynchron bezeichnet [29], da sich die beiden Merkmalströme über die unabhängigen Zustände mit unterschiedlichen Geschwindigkeiten entwickeln können. Es ist nicht erforderlich, dass beide Untermodelle gleichzeitig den Zustand wechseln. Die beiden Ströme werden erst auf Klassenebene synchronisiert, da ein Klassenwechsel nur dann möglich ist, wenn beide Untermodelle ihn zulassen. Es handelt sich jedoch nicht um ein voll asynchrones Modell, wie in Kapitel 4, da die beiden Merkmalströme für das Multi-Stream-Modell gleich lang sein müssen und keine Verschiebung der Merkmalströme zueinander stattfindet.

Durch eine zusätzliche Überkreuzverbindung zwischen dem Zustandszeiger des einen Untermodells und der Übergangswahrscheinlichkeit des anderen Untermodells, also $q_t^{k,\mathrm{A}} \rightarrow a_t^{\mathrm{V}}$ und $q_t^{k,\mathrm{V}} \rightarrow a_t^{\mathrm{A}}$, wird aus dem Multi-Stream ein gekoppeltes Modell. Dann können die Zustandswechsel der beiden Merkmalströme immer noch individuell erfolgen, sind aber stärker an das Verhalten des jeweils anderen Stroms gebunden. Das Modell erhält dadurch jedoch deutlich mehr Parameter, da die beiden Übergangswahrscheinlichkeiten a_t^{A} und a_t^{V} dann jeweils von zwei Zuständen abhängen. Durch zusätzliche Knoten e_t^{A} und e_t^{V}, wie in Abbildung 6.4, kann das Modell auch auf ein links-rechts- bzw. ergodisches Modell erweitert werden. Dies kann entweder nur

in einem oder in beiden Untermodellen erfolgen. Mit einer Verbindung zwischen der Klasse k_t und dem Klassenwechsel w_t können unterschiedliche Zustände für verschiedene Klassen modelliert werden. Mit einem zusätzlichen Wahrscheinlichkeitsknoten, der auf den Zustandszeiger zeigt, kann eine Einsprungswahrscheinlichkeit modelliert werden.

6.3.5 Training

Prinzipiell wäre das Lernen der Parameter für die Modelle in Abbildung 6.3, 6.4 und 6.5 direkt mit dem in Abschnitt 2.9 beschriebenem EM-Algorithmus möglich. Allerdings wäre dann nur die Beobachtung $\vec{o}$ observiert, alle anderen Knoten würden unüberwacht gelernt werden. Obwohl der EM-Algorithmus dies zulässt und speziell für das Lernen von nicht beobachteten Knoten geeignet ist, käme es damit zu einer ungewollten unüberwachten Dekomposition des Klassenknotens. Der Zusammenhang zwischen Klasse und Beobachtung bei einer unüberwachten optimalen Zustandsbelegung würde so nicht gelernt werden.

Überwachtes Training

Für das Training ist es daher notwendig, den Klassenknoten k_t zu beobachten und damit überwacht zu lernen. Dies hat jedoch auch Konsequenzen auf den Klassenwechselknoten w_t: Die bestimmende Verbindung von w_t zu k_t entfällt, da der Klassenknoten im Training rein deterministisch ist. Dann muss jedoch sowohl von k_t als auch von k_{t-1} eine deterministische Verbindung auf w_t zeigen, damit der Knoten w_t – und damit alle im Modell darunter liegenden Knoten – ableiten können, wann im Training ein Klassenwechsel stattfindet. Durch das einfache Beobachten des Klassenknotens k_t sind daher wesentliche Änderungen am Modell notwendig.

Diese Modelländerungen können durch eine allgemeinere Trainingsstruktur vermieden werden. Diese Struktur kann dann für viele Modelle mit integrierter Segmentierung und Klassifizierung genutzt werden, ohne dass die Dekodierungsstruktur wesentlich modifizert werden muss. In den Beispielen zu [30] wird eine solche allgemeine Trainingsstruktur für klassenweises Training vorgestellt. Diese Struktur ordnet einer Merkmalsequenz $\vec{o}$ eine Abfolge von Klassen zu, dabei wird jedoch nicht die Position der Klassengrenzen angegeben. Es findet also ein teilweise unüberwachtes Lernen der Klassen statt, bei der nur die richtige Abfolge, nicht aber die richtigen Segmentgrenzen gelernt werden[7]. Für das Training der Videomodi und viele andere Anwendungen ist es jedoch erforderlich, auch die richtigen Klassengrenzen zu lernen, da nicht nur die Abfolge der Klassen, sondern auch die Position der Segmentgrenzen wichtig ist.

[7] Dieses Vorgehen ist in der Spracherkennung verbreitet: Die Trainingsdaten werden meistens auf Wortebene annotiert. Die Modelle werden dann auf Phonemebene trainiert, wobei nur die Abfolge der Phoneme im Wort, nicht aber die Phonemgrenzen bekannt sind.

Trainingsstruktur

Die Trainingsstruktur wird daher im Rahmen dieser Arbeit so erweitert, dass sowohl ein unüberwachtes Lernen der Klassenabfolge ohne bekannte Segmentgrenzen als auch ein überwachtes Lernen mit bekannten Klassengrenzen durchgeführt werden kann. Dieses erweiterte Modell ist in Abbildung 6.6 gezeigt. Die obersten drei Schichten stellen dabei die Trainingsstruktur dar, die unteren Schichten das zu trainierende Modell – hier das lineare Modell aus Abbildung 6.3. Die Trainingsstruktur besteht in jedem Zeitschritt aus den drei Knoten Klasse k_t, Klassenzähler κ_t und dem Merkmalzähler t, dabei sind alle Knoten beobachtet.

Der Merkmalzähler t wird für ein überwachtes Lernen der Segmentgrenzen benötigt. Er wird im Prolog auf $t = 1$ gesetzt und in jedem weiteren Zeitschritt inkrementiert. Über den Merkmalzähler hat das Trainingsmodell daher für jeden Zeitschlitz im Modell integriert die Information in welchem Zeitschritt es sich befindet.

Vom Merkmalzähler hängt direkt der Klassenzähler κ_t ab. Auch dieser wird im ersten Zeitschritt auf eins gesetzt, d. h. $f(\kappa_1 = 1 \,|\, t = 1)$. Für alle weiteren Zeitpunkte hängt die Implementierung $f(\kappa_t \,|\, \kappa_{t-1}, t, w_t)$ des Klassenzählers von den Trainingsdaten ab und muss für jedes Trainingsbeispiel neu erstellt werden. Dabei bleibt $f(\kappa_t = i \,|\, \kappa_{t-1} = i, t, w_t)$, wenn in dem Trainingsbeispiel zum Zeitpunkt t kein Klassenwechsel stattfindet. Findet jedoch ein Klassenwechsel statt, wird der Klassenzähler inkrementiert, so dass $f(\kappa_t = i + 1 \,|\, \kappa_{t-1} = i, t, w_t)$. Damit zählt der Klassenzähler lediglich die Anzahl der Klassenwechsel in den Trainingsbeispielen und stellt die Wechsel an der richtigen Position sicher. Er enthält jedoch keine Information in welche Klasse gewechselt wird. Gleichzeitig stellt die deterministisch abhängige Kindbeziehung zum Klassenwechselknoten w_t sicher, dass bei einem vorhandenen Klassenwechsel in den Trainingsdaten auch der Klassenwechselknoten auf $w_t = 1$ gesetzt ist und damit die Übergänge in allen darunter liegenden Knoten mitgelernt werden. Durch diese Implementierung als Kindknoten muss der Klassenwechselknoten w_t und auch sonst kein anderer Modellknoten verändert werden. Im Epilog gibt es noch einen Ende-Knoten, der sicherstellt, dass alle Klassenwechsel durchgeführt wurden, und gleichzeitig für den Klassenwechselknoten sicherstellt, dass die Trainingssequenz mit einer abgeschlossenen Klasse beendet wird.

Der Klassenknoten k_t hängt wiederum nur vom Klassenzähler κ_t ab. Auch das Verhalten des Klassenknotens wird abhängig von den Trainingsdaten erstellt. Im Klassenknoten wird deterministisch die Abfolge der Klassen im Trainingsbeispiel abgelegt, jedoch keine Information über die Position der Segmentgrenzen. Der Klassenzähler κ_t dient als Zeiger, welche Klasse gerade aktuell ist. Da bei jedem Klassenwechsel der Zähler κ_t inkrementiert wird, wird damit automatisch auch auf die nächste Klasse im Klassenknoten k_t gezeigt. Dadurch verhalten sich ausgehende Verbindungen vom Klassenknoten identisch zu dem Klassenknoten in den Dekodierungsmodellen und es ist nicht erforderlich, das zu trainierende Modell zu modifizieren.

Die Trainingsstruktur ist daher selbst wiederum ein GM, das mit jedem anderen

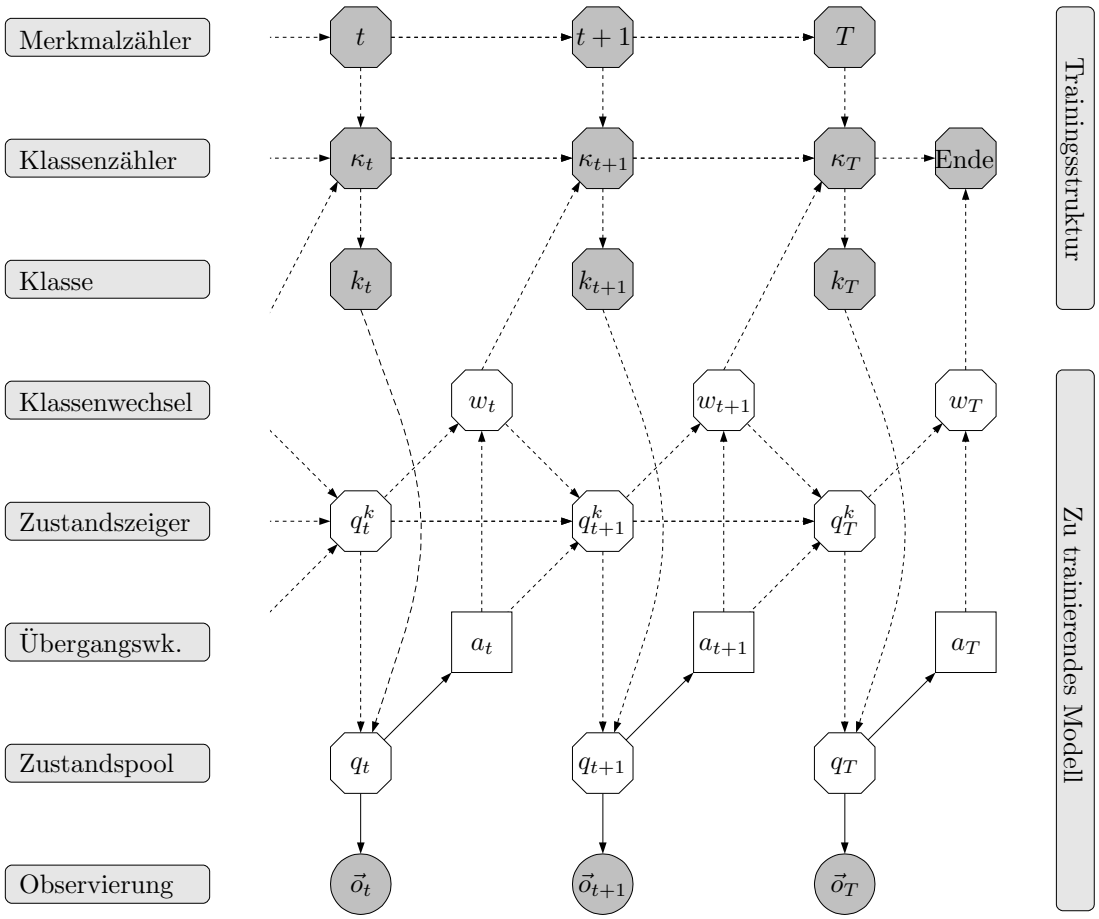

Abbildung 6.6: Trainingsstruktur zum Lernen von Segmentgrenzen. Die drei Schichten der Trainingsstruktur können mit jedem GM verbunden werden. Das zu trainierende Modell (hier das Modell aus Abbildung 6.3) muss für das Training nicht modifiziert werden, da die Trainingsstruktur die gleichen Schnittstellen wie die Dekodierungsstruktur zur Verfügung stellt.

GM verknüpft werden kann. Dabei stellt die Trainingsstruktur solche Schnittstellen zur Verfügung, dass das zu trainierende Modell nur minimal verändert werden muss. Nur die bestimmende Verbindung vom Klassenwechselknoten w_t auf den Klassenknoten k_t entfällt. Dafür wird eine Verbindung von w_t auf den Klassenzähler κ_t eingeführt. Durch diese abgehende Verbindung muss kein Knoten des zu trainierenden Modells für das Training modifiziert werden. Auf diese Weise kann prinzipiell jedes beliebige GM mit der Trainingsstruktur trainiert werden. Dabei werden im Training die Parameter aller Wahrscheinlichkeitsknoten, nicht aber die deterministischen Knoten gelernt.

Wird in der Trainingsstruktur der Merkmalzähler entfernt, enthält der Klassenzähler nur noch die Information, wieviel Klassen in der Sequenz enhalten sind, die

Klassengrenzen werden dann jedoch unüberwacht festgelegt. Durch das Entfernen der obersten Knotenschicht ändert sich das Modell daher vom überwachten Lernen der Segmentgrenzen zu einem unüberwachten Lernen der Klassenabfolge. Wird der Klassenknoten während des Trainings um eine weiteren Wahrscheinlichkeitsknoten erweitert und gleichzeitig k_t und k_{t+1} verbunden, kann mit dem Modell auch die Klassenbigrammwahrscheinlichkeit mitgelernt werden.

6.4 Experimente

Die hier vorgestellten GM zur integrierten Segmentierung und Klassifizierung werden angewendet, um aus den in Abschnitt 6.1 vorgestellten Konferenzdaten automatisch ein Video zu schneiden. Dafür muss jedem Zeitpunkt der Konferenzen einer der in Abschnitt 6.2 vorgestellten sieben Videomodi zugeordnet werden. Segmentgrenzen – d. h. die Zeitpunkte der Schnitte – sind dabei nicht bekannt. Die Systeme mussten die Konferenz daher automatisch segmentieren und jedem Segment dann einen Videomodus zuordnen. Als Merkmale werden die in Abschnitt 6.1 erläuterten akustischen MFCC-Merkmale $\vec{o}^{\mathrm{A}}$ und die globalen Bewegungen in den einzelnen Kameras als visuelle Merkmale $\vec{o}^{\mathrm{V}}$ verwendet.

Um für dieses Problem Mustererkennungsverfahren anwenden zu können, wird Trainingsmaterial für alle Videomodus-Klassen benötigt. In [5, 74] wurde für die Videomodi ein Satz von sehr elementaren Annotierungsregeln für die IDIAP Konferenzdaten aufgestellt: Unter anderem wurden die Annotatoren aufgefordert, soweit möglich für mindestens zehn Sekunden bei einer Einstellung zu bleiben, um sehr schnelle Wechsel zu verhindern. Ansonsten konnten die Annotatoren frei zwischen den sieben Videomodi wählen. Insbesondere musste nicht immer der Sprecher gezeigt werden. Damit konnten die Annotatoren den Videomodus auswählen, von dem sie meinten, dass er die Konferenz zu diesem Zeitpunkt am besten repräsentiert. Dadurch war der Freiheitsgrad beim Annotieren sehr hoch und die ersten Annotierungen hatten eine sehr niedrige Inter-Annotator-Übereinstimmung ($\kappa < 0.5$). Für das Training von automatischen Methoden waren die Annotierungen daher nicht konsistent genug. Weitere Annotierungsversuche zeigten jedoch, dass einzelne Annotierer sehr konsistent in ihrer Annotierung waren, wenn sie die selbe Konferenz mehrmals annotierten. Dies zeigt dass die Annotierung – und damit die gewünschte Kameraeinstellung – stark von den Annotatoren abhängt. Es wurden daher für den Datensatz nur zwei Annotatoren genutzt, um konsistente Trainings- und Testdaten zu erhalten.

Von den 24 annotierten Videos werden 18 Konferenzen für das Training der Modelle und die verbleibenden sechs unbekannten Konferenzen für den Test-Satz verwendet. Die in dieser Arbeit vorgestellen GM werden mit einem HMM [5, 74] und einem asynchronem HMM [96] verglichen, die beide auf den gleichen Datensatz angewendet werden.

6.4.1 Bewertungsmaße

Zum Vergleich der Methoden werden zwei Gütemaße verwendet: Die Action Error Rate (AER) [3] ist von der Wortfehlerrate in der Spracherkennung abgeleitet. Für die AER wird bestimmt, wieviele Einfügungen (*insertion, Ins*), Löschungen (*deletion, Del*) und Ersetzungen (*substitution, Sub*) notwendig sind, um von der erkannten Sequenz von Videomodi auf die annotierte Sequenz von Videomodi zu kommen. Die AER gibt in Prozent das Verhältnis der Summe der drei Änderungen zur Gesamtanzahl aller Videomodi in der Annotierung an:

$$\mathrm{AER} = \frac{\mathrm{Ins} + \mathrm{Del} + \mathrm{Sub}}{\text{Annotierte Videomodi}} \times 100\%. \tag{6.7}$$

Die AER gibt somit nur an, wie ähnlich die erkannte Abfolge von Videomodi zur Abfolge annotierter Videomodi ist. Die AER gibt jedoch nicht an, wie gut Schnittpunkte zeitlich getroffen werden bzw. ob die gefundene Sequenz auch zeitlich – also die Länge der einzelnen Segmente – mit der annotierten Sequenz übereinstimmt.

Die Frame Error Rate (FER) hingegen vergleicht die zeitliche Relation der gefundenen Sequenz von Videomodi mit der annotierten Sequenz. Dazu wird für jedes Frame der Konferenz der gefundene Videomodus mit dem annotierten Videomodus verglichen. Die Anzahl an Frames, für die nicht der richtige Videomodus erkannt wurde (N_{falsch}), im Verhältnis zu allen Frames der Konferenz (N_{alle}) in Prozent ist dann die FER:

$$\mathrm{FER} = \frac{N_{\mathrm{falsch}}}{N_{\mathrm{alle}}} \times 100\%. \tag{6.8}$$

Die FER gewichtet somit stark die richtige zeitliche Anordnung der gefundenen Videomodi, also sowohl die Schnittpunkte als auch die Klasse.

6.4.2 Ergebnisse

Tabelle 6.1 zeigt die Erkennungsergebnisse als FER und AER für die sieben Videomodi für alle angewendeten Modelle. Das lineare, das links-rechts- und das ergodische Modell haben für jede Klasse jeweils fünf verborgene Zustände q_t^k. Zwischen den Klassen werden keine Zustände geteilt. Die Observierung wird jeweils mit zwei Gauß-Mixturen modelliert.

Alle Multi-Stream-Modelle haben für beide Ströme $q_t^{k,\mathrm{A}}$ und $q_t^{k,\mathrm{V}}$ jeweils drei verborgene Zustände, zwischen den Klassen gibt es wiederum keine gemeinsamen Zustände. Auch für alle Multi-Stream-Modelle werden jeweils zwei Gauß-Mixturen für die Modellierung der Observierung verwendet.

Die erzielten FER der Modelle liegen zwischen knapp 48% und knapp 80%. Die erzielten AER der Modelle liegen zwischen 10% und 25%. Während die niedrigen AER zeigen, dass die Modelle die notwendige Abfolge von Videomodi relativ gut generalisieren können, zeigen die hohen FER auf der anderen Seite, dass die Segmentgrenzen dabei nicht genau genug getroffen werden.

Tabelle 6.1: Frame Error (FER) und Action Error Rate (AER) für den automatischen Videoschnitt mit sieben verschiedenen Videomodi und die verschiedenen GM.

Modell	FER (in %)	AER (in %)
Linear	53,5	15,5
Links-rechts	50,7	10,1
Ergodisch	47,7	12,7
Lineares Multi-Stream	53,1	9,6
Links-rechts Multi-Stream	65,5	18,6
Ergodisches Multi-Stream	79,5	25,5
Lineares, gekoppeltes Multi-Stream	68,9	11,8

Unter den Modellen mit nur einem Merkmalstrom erzielt das lineare Modell sowohl in der FER mit 53,5% falsch erkannten Frames als auch in der AER mit 15,5% das schlechteste Ergebnis. Durch den höheren Freiheitsgrad kann das links-rechts-Modelle die Videomodi besser lernen und erzielt eine AER von nur noch 10,1% (5,4% absolute bzw. 34,8% relative Fehlerreduktion) bzw. eine FER von 50,7% (2,8% absolute bzw. 5,2% relative Fehlerreduktion). Mit noch mehr Freiheitsgraden beim ergodischen Modell sinkt zwar die FER noch weiter auf 47,7% ab (5,8% absolute bzw. 10,8% relative Fehlerreduktion), die AER verbessert sich jedoch nur auf 12,7% (2,8% absolute bzw. 18,1% relative Fehlerreduktion).

Die Multi-Stream-Modelle können die Erkennung nicht verbessern: Zwar erreicht das lineare Multi-Stream-Modell mit 53,1% eine leicht bessere FER als das normale lineare Modell und eine deutlich besser AER von 9,6% (5,9% absolute bzw. 38,1% relative Fehlerreduktion), alle anderen Multi-Stream-Modelle führen jedoch zu einer deutlichen Verschlechterung sowohl der AER als auch der FER. Lediglich das gekoppelte Multi-Stream-Modell kann die Sequenz der Videomodi sehr gut generalisieren und erzielt daher eine AER von 11,8%; dies geschieht jedoch unter starker Vernachlässigung der exakten Schnittpunkte, wodurch das Modell eine FER von nur 68,9% erreicht.

Eine leichte Steigerung der Freiheitsgerade – wie beim Übergang vom linearen zum ergodischen GM – bietet mehr Möglichkeiten zur Anpassung an die Daten und führt damit zu besseren Erkennungsraten. Auf der anderen Seite führen – entgegen der theoretischen Erwartung – die Multi-Stream Modelle mit noch mehr Freiheitsgerade wieder zu einer Verschlechterung der Erkennungsergebnisse. Dies deutet darauf hin, dass für diese Modelle zu wenig Trainingsdaten zum Lernen aller Parameter zur Verfügung stehen bzw. durch den relativ kleinen Trainingsdatensatz eine Überanpassung stattfindet. Dieser Effekt wird noch dadurch verstärkt, dass alle zur Verfügung stehenden GM-Software-Werkzeuge noch nicht für das Lernen von hochdimensiona-

len Merkmalen optimiert sind.

Dies zeigt sich auch daran, dass ein in HTK [166] implementieres HMM auf dem gleichen Datensatz eine FER von 47,9% [5, 74] erzielt. Es ist damit nur unwesentlich schlechter als das hier vorgestellte ergodische GM. Das in Kapitel 4 eingeführte asynchrone HMM erzielt auf dem gleichen Datensatz eine FER von 32,3% [96] und ist damit wesentlich besser als alle anderen hier getesteten Modelle. Dies bestätigt zum einen die Ergebnisse andere Arbeiten (z. B. [7, 170]), wonach die Aufzeichnungen von Konferenzen und Besprechungen hohe asynchrone Eigenschaften zwischen dem visuellen und dem akustischen Kanal haben. Zum anderen zeigt es aber auch, dass die Erkennungsergebnisse stark von der Implementierung des Algorithmus bzw. dem verwendeten Werkzeug abhängen. Während sowohl die HMM- als auch die AHMM-Implementierung stark optimiert ist, ist die verwendete GM-Implementierung [30] sehr allgemein gehalten und außerdem noch in einem prototypischen Stadium und in ständiger Weiterentwicklung. Hierbei handelt es sich nicht um theoretische Weiterverbesserungen, sondern um softwaretechnische, vor allem numerische, Details. Insbesondere die Implementierung der Trainingsverfahren wird ständig verbessert, so dass in Zukunft voraussichtlich mit der gleichen Menge an Trainingsdaten mehr Parameter im GM gelernt werden können. In diesem Kontext sind die mit den GM erzielten Erkennungraten – die teilweise besser als die des HMM sind – sehr vielversprechend. In Zukunft können voraussichtlich alleine durch softwaretechnische Änderungen an den verwendeten Werkzeugen bessere Erkennungsraten mit den selben GM und den selben Daten erzielt werden.

Die Ergebnisse zeigen aber auch, dass selbst das asynchrone HMM – als bestes Modell für den automatischen Videoschnitt – nur ungefähr zwei Drittel aller Frames dem richtigen Videomodus zuordnet. Die vorgestellten GM ordnen sogar nur die Hälfte aller Frames richtig zu. Das Problem stellt eine sehr anspruchsvolle Mustererkennungsaufgabe dar. Dieser Effekt wird durch den relativ kleinen Trainingsdatensatz noch verstärkt. Auf der anderen Seite repräsentieren die Fehlerraten lediglich den Vergleich zur subjektiven Annotation. Eine hohe FER bedeutet daher nicht zwangsläufig, dass das durch die hier vorgestellten Methoden automatisch erstellte Video auch eine schlechte Repräsentierung der Besprechung darstellt. Subjektiv bewertet zeigen die automatisch erstellten Videos durchaus gute Zusammenschnitte. Für eine weitere Analyse könnte es daher sinnvoll sein die Ergebnisse durch Benutzerstudien auch subjektiv zu beurteilen.

6.5 Kapitelzusammenfassung und Ausblick

In diesem Kapitel wurde gezeigt, wie GM so erweitert werden können, dass sie Daten integriert segmentieren und klassifizieren können. Ausgehend von einer grundlegenden linearen Struktur wurden in dieser Arbeit Modelle entwickelt, die eine links-rechts- und ergodische Verarbeitung ermöglichen. Ein bekanntes Multi-Stream-GM

wurde so adaptiert, dass es für die multimodale Fusion von Audio- und Video-Daten verwendet werden kann. Die Methoden zur Segmentierung sind jedoch nicht auf die gezeigten Modelle begrenzt, die gezeigte Segmentierungsstruktur kann prinzipiell auf fast alle GM angewendet werden. Dadurch können bekannte Strukturen für die integrierte Segmentierung und Klassifikation von Daten erweitert werden.

Um die Modelle mit dem EM-Algorithmus trainieren zu können, werden spezielle Trainingsstrukturen benötigt. Diese stammen bisher vor allem aus der Sprachverarbeitung und bieten daher lediglich das Trainieren einer Abfolge von Klassen, jedoch ohne bekannte Klassengrenzen an. Im Rahmen dieser Arbeit wurde daher eine bekannte Trainingsstruktur durch eine zusätzliche Ebene so erweitert, dass die Modelle entweder mit einer Abfolge von Klassen bei unbekannten oder bei bekannten Klassengrenzen – also auf Framebasis – trainiert werden können. Die so entwickelte Trainingsstruktur ist selbst wieder ein GM und kann daher mit vielen anderen Modellen verbunden werden. Das Training ist daher nicht auf die in diesem Kapitel behandelten GM beschränkt, sondern kann prinzipiell auf fast alle GM angewendet werden.

In einem experimentellen Teil, wurden die entwickelten Modelle auf den automatischen Videoschnitt von Konferenz- und Besprechungsaufnahmen angewendet. Dabei wird aus einer Konferenzaufnahme mit mehreren Kameras ein einziges Video zusammengeschnitten. Das System muss daher jedem Zeitpunkt der Konferenz eine der verfügbaren Kameras zuordnen und somit automatisch die Schnittpunkte und die Kameraeinstellung innerhalb der Schnitte festlegen. Auf einem Datensatz, der im IDIAP Besprechungsraum mit sieben Kameras aufgezeichnet wurde, erreichte das entwickelte ergodische Modell eine Action Error Rate (AER) von 12,7% und eine Frame Error Rate (FER) von 47,7% im Vergleich zu den annotieren Kameraeinstellungen. Das ergodische Modell erzielt damit eine relative Fehlerreduktion von 10,8% AER bzw. 18,1% FER gegenüber dem bekannten linearen Modell. Ein HMM erzielt auf dem gleichen Datensatz eine FER von 47,9%, ein AHMM eine FER von 32,3%.

Die erzielten – absolut gesehen relativ schlechten – Ergebnisse zeigen daher zum einen, dass der automatische Videoschnitt ein anspruchsvolles Problem ist. Insbesondere der Vergleich zu dem hochoptimierten AHMM zeigt jedoch auch, dass die zur Zeit verfügbaren Software-Werkzeuge für GM noch sehr allgemein gehalten sind und daher bezüglich des Trainings und der Erkennung noch nicht optimiert sind.

Die kontinuierliche Weiterentwicklung der verfügbaren Implementierungen für GM lassen mit den gleichen Modellen und Daten bessere Ergebnisse erwarten. Insbesondere ein softwaretechnisch verbessertes Training sowie diverse mögliche numerische Optimierungen führen voraussichtlich zu besseren praktischen Umsetzungen der Modelle: So sind z. B. für das EM-Training von GMM viele kleinere Implementierungsverbesserungen möglich, die zu einer deutlich besseren Modellierung der Daten führen können. Viele dieser bekannten praktischen Verfahren sind jedoch noch nicht in die allgemeinen Werkzeuge integriert. Um sehr gute Erkennungsraten zu erzielen, ist es daher zur Zeit häufig notwendig, aus der allgemeinen Nachrichtenpropagierung

Formeln abzuleiten und direkt in optimierter Form zu implementieren.

Die durch die hier vorgestellten Methoden zusammengeschnittenen Videos geben – trotz der relativ hohen Fehlerraten – subjektiv betrachtet die Konferenzen relativ gut wieder. Es scheint daher sinnvoll, ein besseres Fehlermaß für den Videoschnitt zu definieren. Insbesondere die subjektive Bewertung durch Benutzerstudien scheint dabei gut geeignet.

Die vorgestellten Methoden zur integrierten Segmentierung und Klassifizierung sowie deren Trainingsstrukturen sind jedoch nicht auf den vorgestellten automatischen Videoschnitt begrenzt. Prinzipiell können die Modelle auf alle Probleme, bei denen sowohl segmentiert als auch klassifiziert werden muss, angewendet werden. Dabei können durch die allgemein gehaltenen Strukturen für die Modellierung der Klasse bzw. der Segmentierung auch problemangepasste Änderungen vorgenommen werden.

7

Zusammenfassung

In dieser Arbeit wurde untersucht, ob und wie Graphische Modelle für verschiedene Mustererkennungsprobleme verwendet werden können. Zuerst wurde eine umfassende Einführung in die Theorie gegeben. Insbesondere wurde gezeigt, wie bekannte Verfahren der Mustererkennung als Graphisches Modell dargestellt werden können und wie sich daraus effiziente Berechnungsvorschriften ableiten lassen. Es konnte gezeigt werden, wie unterschiedliche Algorithmen und Verfahren verschiedener Teildisziplinen durch Graphische Modelle einheitlich analysiert und damit auch miteinander verglichen und kombiniert werden können.

7.1 Entwickelte Modelle und erzielte Ergebnisse

Für vier verschiedene Anwendungen der Mustererkennung wurden in dieser Arbeit neue Graphische Modelle entwickelt und unterschiedliche Aspekte theoretisch und experimentell untersucht.

Für die automatische Schnitt- und Szenenerkennung in Videos wurde ein neues, zweistufiges Modell vorgestellt. Dieses integriert signalnahe und semantisch höherwertige Merkmale in zwei Modellebenen und optimiert so die Anordnung von Schnitten und Szenen im Verbund. Benötigte Berechnungsvorschriften für das Training und die Dekodierung wurden hergeleitet und analysiert. In Experimenten zur Schnitt- und Szenenerkennung konnte das hier entworfene Modell die Erkennungsraten – im Vergleich zu einer Schwellwertmethode – deutlich steigern. In dem in dieser Arbeit vorgestellten Graphischen Modell profitieren daher sowohl die Schnitt- als auch die Szenenerkennung von der problemangepassten Modellierung und der sich daraus ergebenden gemeinsamen Optimierung.

Zur Fusion von bimodalen Sprach- und Gestenkommandos wurde in dieser Arbeit das asynchrone Hidden Markov Modell weiterentwickelt. Dieses Graphische Modell kann die Verbundwahrscheinlichkeit von zwei Datenströmen modellieren, auch wenn sie asynchron zueinander sind. Allerdings führt eine steigende Asynchronität auch

zu einer höheren Rechenkomplexität. In dieser Arbeit wurde deswegen die Berechnung durch eine Einführung von dynamischen Bereichsgrenzen im Trellis-Diagramm verbessert. So konnte die Komplexität sowohl für das Training als auch für die Klassifikation erheblich reduziert werden. Die entwickelte Berechnungsvorschrift erlaubt außerdem eine einfache Skalierungsprozedur. Durch die in dieser Arbeit erzielte Komplexitätsreduktion konnte das Modell erstmals auch auf die Fusion von Benutzereingaben angewendet werden. Dabei ist die Komplexität der Berechnungen besonders kritisch, da die Asynchronität zwischen den Datenströmen sehr hoch werden kann. In Experimenten zur Fusion von Sprache und Gesten konnte mit dem asynchronen Hidden Markov Modell die Erkennungsrate sowohl gegenüber einem Hidden Markov Modell mit früher als auch mit später Fusion deutlich gesteigert werden.

Um Gruppenaktionen in Besprechungen auch aus fehlerhaften Daten robust erkennen zu können, wurde ein neues multimodales Graphisches Modell vorgestellt: Es kombiniert auf Modellebene ein gekoppeltes Hidden Markov Modell mit einem linearen dynamischen System. Das gekoppelte Hidden Markov Modell fusioniert akustische und visuelle Merkmale und dient dann als treibender Eingang des linearen dynamischen Systems. Damit können eventuelle Störungen in den Daten ausgeglichen werden, wovon das gekoppelte Hidden Markov Modell wiederum in der Erkennung profitiert. Da das kombinierte Modell eine zu hohe Rechenkomplexität hat, wurde ein approximatives Verfahren mittels strukturiert abweichendem Schließen vorgestellt. Die approximativen Berechnungsvorschriften für das Modell wurden hergeleitet und die sich daraus ergebenden Trainings- und Dekodierungsverfahren erläutert. In Experimenten mit ungestörten und fehlerbehafteten Daten erzielte das Modell für alle Konfigurationen wesentlich höhere Erkennungsergebnisse als unimodale Hidden Markov Modelle und für fast alle Konfigurationen bessere Ergebnisse als ein multimodales Hidden Markov Modell mit früher Fusion. Störungen können daher mit dem hier vorgestellten Graphischen Modell besser kompensiert werden.

Für das automatische Zusammenschneiden von Konferenzvideoaufzeichnungen wurden Graphische Modelle zum integrierten Segmentieren und Klassifizieren der Daten vorgestellt. Ausgehend von linearen Modellen wurden links-rechts- und ergodische Erweiterungen entwickelt. Ein Multi-Stream-Modell wurde adaptiert, so dass es auch für die Fusion von Audio- und Video-Daten verwendet werden kann. Für das Training wurde eine Struktur zum unüberwachten Lernen von Klassengrenzen so erweitert, dass eine Abfolge von Klassen sowohl bei bekannten als auch bei unbekannten Segmentgrenzen trainiert werden kann. Die entwickelte Trainingsstruktur ist nicht auf die in dieser Arbeit vorgestellten Modelle beschränkt. In Experimenten wurden die entwickelten Modelle für den Videoschnitt von Konferenzaufnahmen angewendet. Hierbei konnte die Erkennungsrate sowohl durch die links-rechts- als auch durch die ergodische Modellerweiterung im Vergleich zur linearen Grundstruktur verbessert werden. Das Multi-Stream-Modell führte zu keiner Verbesserung. Im Vergleich mit einem Hidden Markov Modell konnten die vorgestellten Graphischen Modelle bei dieser Anwendung keinen signifikanten Vorteil erzielen.

7.2 Diskussion

Die Anwendung auf vier teilweise sehr unterschiedliche Probleme der Mustererkennung zeigt die große Flexibilität von Graphischen Modellen. Im Vergleich zu vielen anderen Methoden erlauben sie, dass das Modell an das Problem angepasst wird. Prinzipiell kann für jede Anwendung ein neues Modell entwickelt werden. Die notwendigen Berechnungsvorschriften können dann formal, teilweise sogar automatisiert, abgeleitet werden. Graphische Modelle erlauben daher, Probleme auf abstrakter Ebene zu modellieren und zu analysieren ohne sich mit algorithmischen Details beschäftigen zu müssen. Statistische Konsequenzen von Annahmen werden dabei im Graphen tatsächlich „sichtbar“.

Durch ihre hohe Flexibilität, ihre vielfältigen Anwendungsmöglichkeiten, aber vor allem den Formalismus bieten Graphische Modelle einen einheitlichen Blickwinkel auf verschiedene Probleme und Verfahren. Unterschiedliche Methoden können so miteinander verglichen, effiziente Berechnungsverfahren von einer Methode auf eine andere übertragen und verschiedene Modelle auch miteinander kombiniert werden. Vor- und Nachteile verschiedener Modellierungen werden dabei direkt sichtbar.

Mit der natürlichen und intuitiven Darstellungsform erhöhen Graphische Modelle das Verständnis für viele Verfahren. Lösungen können graphisch skizziert und formuliert werden. Graphische Modelle bieten daher die Möglichkeit, verschiedene Methoden prototypisch zu entwerfen. Die standardisierten Algorithmen erlauben dann ein sofortiges Überprüfen der entworfenen Lösungen. In späteren Verfeinerungsschritten können dann mit standardisierten Verfahren aus dem allgemeinen Modell verbesserte Berechnungsvorschriften abgeleitet werden.

Die Ergebnisse dieser Arbeit zeigen jedoch auch, dass zur Zeit vor allem die softwaretechnische Umsetzung der Modelle problematisch ist. Obwohl Software für Graphische Modelle verfügbar ist, müssen spezielle Aspekte häufig noch direkt implementiert werden. Dies widerspricht dem Grundgedanken, dass eine Lösung nur skizziert wird, während die notwendigen Berechnungen automatisch durchgeführt werden. Die Software für Graphische Modelle befindet sich noch im Entwicklungsstadium, die Berechnungen sind daher noch nicht optimiert. Insbesondere das Lernen der Modellparameter ist häufig noch nicht optimiert. Durch die Notwendigkeit, spezielle Modelle direkt zu implementieren, ist auch der prototypische Entwurf eingeschränkt, da jedes neue Modell dann zu hohen Softwareentwicklungszeiten führt.

Durch den problemangepassten Entwurf können die zu trainierenden Parameter in Graphischen Modellen stark ansteigen. Es sind daher – z. B. im Vergleich zu einem Hidden Markov Modell – häufig deutlich mehr Trainingsdaten erforderlich. Wird die Flexibilität der Graphischen Modelle voll ausgenutzt und ein Problem z. B. in mehreren Ebenen modelliert, sind außerdem häufig auch Annotierungen für jede Ebene im Modell notwendig. In Kombination mit der höheren Anzahl an freien Parametern kann dies zu einem erheblichen Mehraufwand bei der Erstellung von Trainingsdaten führen.

7.3 Ausblick

Die Ergebnisse dieser Arbeit zeigen, dass Graphische Modelle erfolgreich auf verschiedene Probleme der Mustererkennung angewendet werden können. Jedoch sind sowohl in der Anwendung als auch bei den theoretischen Grundlagen noch verschiedene Weiterentwicklungen möglich.

Approximative Verfahren bieten die Möglichkeit, auch komplexere Zusammenhänge zu modellieren, sie sind jedoch bisher kaum untersucht. Bekannte approximative Verfahren müssen bisher für jedes Modell neu angepasst werden. Die Entwicklung von allgemeingültigen Methoden für verschiedene Graphische Modelle könnte neue, komplexere Modelle ermöglichen.

Kettengraphen sind mit ihren Modellierungseigenschaften sehr mächtig. Sie können z. B. für die Kombination von Bildverarbeitung und Mustererkennung eingesetzt werden. Aufgrund ihrer hohen Komplexität wurden sie bisher jedoch kaum angewendet. Insbesondere in Kombination mit approximativen Verfahren könnte mit ihnen jedoch eine noch größere Anzahl von Problemen modelliert werden.

Das automatische Lernen der besten graphischen Struktur für ein Problem ist bisher kaum erforscht. Für statische Probleme wird bisher fast ausschließlich eine vollständige Suche über alle möglichen Strukturen durchgeführt. Dies ist jedoch praktisch kaum möglich. Für dynamische Probleme sind keine Verfahren zum Strukturlernen bekannt. Solche Verfahren würden jedoch eine optimale statistische Modellierung eines Problems ermöglichen, da der beste statistische Zusammenhang zwischen allen Variablen gefunden würde.

Aus Anwendungssicht sind insbesondere softwaretechnisch viele Verbesserungen für Graphische Modelle möglich. Das Einbinden asynchroner Knoten in Softwarewerkzeuge würde z. B. die Bildung neuer Modelle für die multimodale Fusion erlauben. Durch theoretisch bekannte, numerische Optimierungen für die Lernverfahren der verfügbaren Software sollten auch die Erkennungsraten bei den hier präsentierten Modellen noch weiter steigen.

In der Anwendung bieten Graphische Modelle vor allem die Möglichkeit, problembezogene Klassifikatoren zu entwickeln: Dabei sind zum einen problemabhängige Neuentwicklungen von Modellen, zum anderen aber auch die Kombinationen unterschiedlicher, bekannter Verfahren zu neuen Methoden möglich. Insbesondere könnte die gemeinsame Betrachtung von Merkmalen und Klassifikatoren zu besseren Problemlösungen führen. Durch die Modellierung mit mehreren Modellebenen können Graphische Modelle nicht nur zur Klassifikation, sondern auch zur gleichzeitigen „Interpretation“ der Erkennung genutzt werden. Graphische Modelle könnten daher in Zukunft genutzt werden, um Muster in Daten nicht nur zu erkennen, sondern auch zu verstehen.

Zusammenfassend lassen die Ergebnisse dieser Arbeit und die gezeigte Mächtigkeit bei der Problemmodellierung erwarten, dass Graphische Modelle in Zukunft für viele Mustererkennungsaufgaben erfolgreich eingesetzt werden können.

Abkürzungsverzeichnis

AER ..... Action Error Rate
AHMM ..... Asynchrones Hidden Markov Modell
AMI ..... Augmented Multi-party Interaction
BN ..... Bayes'sches Netz
BSP ..... Binäre Sprach-Pause-Segmentierung
CHIL ..... Computers in the Human Interaction Loop
CHMM ..... Gekoppeltes (*coupled*) Hidden Markov Modell
DAG ..... Gerichteter, zyklusfreier Graph (*direct acyclic graph*)
DBN ..... Dynamisches Bayes'sches Netz
DCT ..... Diskrete Kosinus Transformation
DF ..... Gerichtete rekursive Faktorisierung
DG ..... Gerichtete globale Markov-Eigenschaft
DL ..... Gerichtete lokale Markov-Eigenschaft
DP ..... Gerichtete paarweise Markov-Eigenschaft
EM ..... Expectation-Maximisation
F ..... Faktorisierung von Graphen
FER ..... Frame Error Rate
G ..... Globale Markov-Eigenschaft
GDL ..... Generalized distributive law
GM ..... Graphisches Modell
GMM ..... Gauß Mixtur Modell
HHMM ..... Hierarchisches Hidden Markov Modell
HMM ..... Hidden Markov Modell
IOHMM ..... Input-Output Hidden Markov Modell
JT ..... Verbundbaum (*junction tree*)
L ..... Lokale Markov-Eigenschaft
LDA ..... Lineare Diskriminanzanalyse
LDS ..... Lineares dynamisches System
M4 ..... The Multimodal Meeting Manager

MDA . . . . . Mixtur-Diskriminanzanalyse
MFCC . . . . . Mel-Frequenz-Cepstrum Koeffizienten
MRF . . . . . Markov Random Field
p . . . . . Precision
P . . . . . Paarweise Markov-Eigenschaft
QDA . . . . . Quadratische Diskriminanzanalyse
QMDA . . . . . Quadratische Mixtur-Diskriminanzanalyse
r . . . . . Recall
SRP-PHAT . . . . Steered Response Power-Phase Transformation
WDF . . . . . Wahrscheinlichkeitsdichtefunktion

Symbolverzeichnis

$\perp\!\!\!\perp$ ..... Bedingte Unabhängigkeit
$\mapsto$ ..... Führt zu
$\rightarrow$ ..... Gerichtete Kante
$\sim$ ..... Ungerichtete Kante
$\rightleftharpoons$ ..... Verbunden
$\underset{\sim}{A}$ ..... Systemzustandsübergangsmatrix
a_{ij} ..... Zustandsübergangswahrscheinlichkeit
a_t ..... Übergangswahrscheinlichkeit
α_t ..... Vorwärtsvariable
α^0 ..... Teil von α, bei dem der Zustand i nur ein Symbol emittiert
α^1 ..... Teil von α, bei dem der Zustand i zwei Symbole emittiert
$\widetilde{\alpha}$ ..... Skalierte Vorwärtsvariable
$\alpha_m^{q_t}$ ..... Mixturgewicht der Mixtur m im Zustand q_t
$\mathrm{An}(V^a)$ ..... Ancestral Hull der Knoten V^a
$\mathrm{an}(V_i)$ ..... Vorfahren des Knotens V_i
$\underset{\sim}{B}$ ..... Systemeingangsmatrix
$\underset{\sim}{B}_i(\vec{o})$ ..... Symbolemissionswahrscheinlichkeit
$\mathrm{bd}(V_i)$ ..... Rand des Knotens V_i
β_t ..... Rückwärtsvariable
$\mathrm{bl}(V_i)$ ..... Markov Decke des Knotens V_i
$\underset{\sim}{C}$ ..... Systemausgangsmatrix
C_n ..... Zyklus mit n Knoten, der keine Sehne enthält
$C_t(u, v)$ ..... DCT-Koeffizient zum Zeitpunkt t
$c(t)$ ..... Skalierungskoeffizient für das AHMM zum Zeitpunkt t
$\mathrm{ch}(V_i)$ ..... Kinder des Knotens V_i
$\mathrm{cl}(V_i)$ ..... Geschlossene Hülle des Knotens V_i
δ ..... Viterbi-Vorwärtsvariable
Δ ..... Menge aller möglichen zeitlichen Anordnungen von $\vec{x}$ und $\vec{y}$
$\mathrm{de}(V_i)$ ..... Nachfahren des Knotens V_i

ϵ_i ..... Emissionswahrscheinlichkeit für ein Symbolpaar im Zustand i
$\mathcal{G}^{\rightarrow}$ ..... Gerichteter Graph
$\mathcal{G}^{\sim}$ ..... Ungerichteter Graph
$\mathcal{G}^{\approx}$ ..... Ähnlicher Graph
$\mathcal{G}^{C}$ ..... Chunk
$\mathcal{G}^{D}$ ..... Gerichteter, zyklusfreier Graph
$\mathcal{G}^{E}$ ..... Epilog
$\mathcal{G}^{m}$ ..... Moralisierter Graph
$\mathcal{G}^{m}_{\mathrm{An}(\cdot)}$ ..... Moralisierter Graph der Ancestral Hull
$\mathcal{G}^{P}$ ..... Prolog
$\vec{h}$ ..... Verborgene Knoten eines Modells
$\vec{h}^{*}$ ..... Wahrscheinlichste Konfiguration der verborgenen Knoten
I ..... Anzahl Trainingsbeispiele
$I_t(x, y)$ ..... Intensitätsbild
K ..... Anzahl Klassen
$\widehat{k}$ ..... Erkannte Klasse
L ..... Position im Raum
$\mathcal{L}$ ..... Gesamtdatenwahrscheinlichkeit (Likelihoodfunktion)
λ ..... Satz von Modellparametern bzw. die Variationsparameter
λ_k ..... Satz von Modellparametern für die Klasse k
$\widehat{\lambda}_k$ ..... Gelernte Modellparametern für die Klasse k
M ..... Anzahl der Mixturen in einem Mixtur Modell
$\vec{\mu}$ ..... Mittelwert
N ..... Anzahl der Zustände eines diskreten Knotens
N_V ..... Anzahl der Knoten in einem GM
$\mathcal{N}(\vec{x}, \vec{\mu}, \underset{\sim}{\Sigma})$ ..... Normalverteilung
$\mathrm{nd}(V_i)$ ..... Nicht-Nachfahren des Knotens V_i
$\mathrm{ne}(V_i)$ ..... Nachbarn des Knotens V_i
$\mathcal{O}$ ..... Satz von I Trainingsbeispielen
$\vec{o}$ ..... Beobachtung der Länge T
$\vec{o}^{\,A}_t$ ..... Akustischer Merkmalvektor zum Zeitpunkt t
$\vec{o}^{\,A}$ ..... Akustischer Merkmalvektor der Länge T_A
$\vec{o}^{\,\mathrm{BSP}}_t$ ..... Positionsabhängiger akustischer Merkmalvektor
$\vec{o}^{\,L}_t$ ..... Von der Position L abhängiger Merkmalvektor zum Zeitpunkt t
$\vec{o}^{\,\mathrm{MFCC}}_t$ ..... Sprecherabhängiger, akustischer Merkmalvektor
$\vec{o}^{\,V}_t$ ..... Globaler Bewegungsmerkmalvektor zum Zeitpunkt t
$\vec{o}^{\,V}$ ..... Globaler Bewegungsmerkmalvektor der Länge T_V
$p(X = x_i)$ ..... Wahrscheinlichkeit, dass X den Wert x_i annimmt
$p(x_i)$ ..... Wahrscheinlichkeit, dass X den Wert x_i annimmt
$\mathrm{pa}(V_i)$ ..... Eltern des Knotens V_i
π_i ..... Zustandseinsprungwahrscheinlichkeit

$\phi_s(x^s)$ ..... Potential des Separators S
$\phi_s^*(x^s)$ ..... Aktualisiertes Potential des Separators S
$\psi_c(x^c)$ ..... Cliquepotential der Clique C
$\psi_c^*(x^c)$ ..... Aktualisiertes Cliquepotential der Clique C
$\underset{\sim}{Q}$ ..... Matrix des Systemrauschens
$\widetilde{Q}(\lambda, \lambda')$ ..... Hilfsfunktion zum EM-Lernen
$\mathcal{Q}$ ..... Menge aller mögliche Zustandssequenzen durch ein Modell
q_t ..... Zustand zum Zeitpunkt t
q_t^{A} ..... Zustand des akustischen Teilmodells zum Zeitpunkt t
q_t^k ..... Zustandszeiger
q_t^{V} ..... Zustand des visuellen Teilmodells zum Zeitpunkt t
$\underset{\sim}{R}$ ..... Matrix des Beobachtungsrauschens
$\widetilde{S}$ ..... Länge einer Merkmalsequenz
$\underset{\sim}{\Sigma}$ ..... Kovarianzmatrix
T ..... Länge einer Merkmalsequenz
t ..... Zeitpunkt oder Framenummer
τ_t ..... Zeitliche Anordnung zum Zeitpunkt t
$\vec{u}_t$ ..... Treibende Eingangsgröße eines LDS
$\vec{u}_t^{\,\mathrm{aus}}$ ..... Virtuelle Observierung des HMM beim approximativen Schließen
$\vec{u}_t^{\,\mathrm{ein}}$ ..... Virtueller Eingang des LDS beim approximativen Schließen
w_t ..... Klassenwechselwahrscheinlichkeit
$\vec{x}$ ..... Merkmalvektor der Länge T
$\vec{x}_t$ ..... Systemzustand eines LDS zum Zeitpunkt t
$\vec{x}_t^{\,\text{C-diff}}$ ..... Frequenzdifferenz zwischen den Bildern t und $t+1$
$\vec{x}_t^{\,\text{H-diff}}$ ..... Grauwerthistogrammdifferenz zwischen t und $t+1$
$\vec{x}_t^{\,\text{I-diff}}$ ..... Intensitätsdifferenz zwischen den Bildern t und $t+1$
$\vec{y}$ ..... Merkmalvektor der Länge S
$\vec{y}_t$ ..... Systemausgabe eines LDS zum Zeitpunkt t
Z ..... Normalisierung
$\vec{z}$ ..... Aus $\vec{x}$ und $\vec{y}$ fusionierter Merkmalvektor

Index

Literaturverzeichnis

[1] Adcock, J.; Cooper, M.; Girgensohn, A.; Wilcox, L.: Interactive Video Search Using Multilevel Indexing. In: *Proceedings International Conference on Image and Video Retrieval (CIVR)*, 2005

[2] Aggarwal, G.; Roy-Chowdhury, A.; Chellappa, R.: A System Identification Approach for Video-Based Face Recognition. In: *Proceedings International Conference on Pattern Recognition (ICPR)*, 2004

[3] Al-Hames, M.; Dielmann, A.; Gatica-Perez, D.; Reiter, S.; Renals, S.; Rigoll, G.; Zhang, D.: Multimodal Integration for Meeting Group Action Segmentation and Recognition. In: Renals, S. (Hrsg.); Bengio, S. (Hrsg.): *Proceedings of the 2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, Springer Verlag, 2006

[4] Al-Hames, M.; Hain, T.; Cernocky, J.; Schreiber, S.; Poel, M.; Müller, R.; Marcel, S.; Leeuwen, D. van; Odobez, J.; Ba, S.; Bourlard, H.; Cardinaux, F.; Gatica-Perez, D.; Janin, A.; Motlicek, P.; Reiter, S.; Renals, S.; Rest, J. van; Rienks, R.; Rigoll, G.; Smith, K.; Thean, A.; Zemcik, P.: Audio-Visual Processing in Meetings: Seven Questions and Current AMI Answers. In: Renals, S. (Hrsg.); Bengio, S. (Hrsg.): *Proceedings of the 3rd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, Springer Verlag, 2006

[5] Al-Hames, M.; Hörnler, B.; Müller, R.; Schenk, J.; Rigoll, G.: Automatic Multi-Modal Meeting Camera Selection for Video-Conferences and Meeting Browsing. In: *Proceedings of the 8th International Conference on Multimedia and Expo (ICME)*, 2007

[6] Al-Hames, M.; Hörnler, B.; Scheuermann, C.; Rigoll, G.: Using Audio, Visual, and Lexical Features in a Multi-Modal Virtual Meeting Director. In:

Renals, S. (Hrsg.); Bengio, S. (Hrsg.): *Proceedings of the 3rd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, Springer Verlag, 2006

[7] Al-Hames, M.; Lenz, C.; Reiter, S.; Schenk, J.; Wallhoff, F.; Rigoll, G.: Robust Multi-Modal Group Action Recognition in Meetigs from Disturbed Videos with the Asynchronous Hidden Markov Model. In: *Proceedings of the International Conference on Image Processing (ICIP)*, 2007

[8] Al-Hames, M.; Rigoll, G.: A Multi-modal Graphical Model for Robust Recognition of Group Actions in Meetings from Disturbed Videos. In: *Proceedings of the International Conference on Image Processing (ICIP)*, 2005

[9] Al-Hames, M.; Rigoll, G.: A Multi-Modal Mixed-State Dynamic Bayesian Network for Robust Meeting Event Recognition from Disturbed Data. In: *Proceedings of the 6th International Conference on Multimedia and Expo (ICME)*, 2005

[10] Al-Hames, M.; Rigoll, G.: *Der EM-Algorithmus und Gauß-Mixtur-Modelle.* Skript zum Praktikum Praxis der Mensch-Maschine-Kommunikation, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation. `http://www.mmk.ei.tum.de/~alh/em_skript.pdf`. Version: 2006

[11] Al-Hames, M.; Rigoll, G.: Reduced Complexity and Scaling for Asynchronous HMMs in a Bimodal Input Fusion Application. In: *Proceedings of the 31st International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2006

[12] Al-Hames, M.; Zettl, S.; Wallhoff, F.; Reiter, S.; Schuller, B.; Rigoll, G.: A Two-Layer Graphical Model for Combined Video Shot and Scene Boundary Detection. In: *Proceedings of the 7th International Conference on Multimedia and Expo (ICME)*, 2006

[13] Althoff, F.; Al-Hames, M.; McGlaun, G.; Lang, M.: Towards a New Approach for Integrating Multimodal User Input Based on Evolutionary Computation. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002

[14] Baum, L.E.; Petrie, T.: Statistical Inference for Probabilistic Functions of Finite State Markov Chains. In: *The Annals of Mathematical Statistics* 37 (1966), S. 1554–1563

[15] Baum, L.E.; Petrie, T.; Soules, G.; Weiss, N.: A Maximization Technique Occuring in The Statistical Analysis of Probabilistic Functions of Markov Chains. In: *The Annals of Mathematical Statistics* 41 (1970), Nr. 1, S. 164–171

[16] Becker, A.; Geiger, D.: A Sufficiently Fast Algorithm for Finding Close to Optimal Clique Trees. In: *Artifical Intelligence* 125 (2001), Nr. 1-2, S. 3–17

[17] Bengio, S.: An Asynchronous Hidden Markov Model for Audio-Visual Speech Recognition. In: *Advances in Neural Information Processing Systems (NIPS) 15*, 2003

[18] Bengio, S.: Multimodal Authentication using Asynchronous HMMs. In: *Proceedings of the 4th International Conference on Audio-and Video-Based Biometrie Person Authentication (AVBPA)*, 2003

[19] Bengio, S.: Multimodal Speech Processing Using Asynchronous Hidden Markov Models. In: *Information Fusion* 5 (2004), Nr. 2, S. 81–89

[20] Bengio, Y.; Frasconi, P.: An Input-Output HMM architecture. In: *Advances in Neural Information Processing Systems (NIPS) 7*, 1995

[21] Berge, C.: *La Theorie des graphes et ses applications.* Dunod, Paris, 1966

[22] Bernard, M.; Odobez, J.-M.: Sports event Recognition using Layered HMMs. In: *Proceedings of the International Conference on Multimedia & Expo (ICME)*, 2005

[23] Bilmes, J.: *Graphical Models.* `http://ssli.ee.washington.edu/courses/ee512`. – Umdruck zu Vorlesung EE512 Sommer 2006, University of Washington, Department of Electrical Engineering

[24] Bilmes, J.: *Graphical Models in Speech and Language, and the Graphical Models Toolkit.* `http://www.clsp.jhu.edu/ws03/preworkshop/lecture_bilmes.pdf`. – Umdruck: The Center for Language and Speech Processing Workshop 2003 at Johns Hopkins University

[25] Bilmes, J.: A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models / University of Berkeley. 1997 (ICSI-TR-97-021). – Forschungsbericht

[26] Bilmes, J.: Graphical Models and Automatic Speech Recognition. In: Johnson, M. (Hrsg.); Khudanpur, S.P. (Hrsg.); Ostendorf, M. (Hrsg.); Rosenfeld, R. (Hrsg.): *Mathematical Foundations of Speech and Language Processing.* Springer-Verlag, 2004, S. 191–246

[27] Bilmes, J.: What HMMs Can Do. In: *IEICE - Transactions on Information and Systems* E89-D (2006), Nr. 3, S. 869–891

[28] Bilmes, J.; Bartels, C.: On Triangulating Dynamic Graphical Models. In: *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003

[29] Bilmes, J.; Bartels, C.: Graphical Model Architectures for Speech Recognition. In: *IEEE Signal Processing Magazine* 22 (2005), Nr. 5, S. 89 – 100

[30] Bilmes, J.; Zweig, G.: The Graphical Model Toolkit: An Open Source Software System for Speech and Time-Series Processing. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2002

[31] Binder, J.; Koller, D.; Russell, S.; Kanazawa, K.: Adaptive Probabilistic Networks with Hidden Variables. In: *Machine Learning* 29 (1997), Nr. 2-3, S. 213–244

[32] Bolt, R.A.: „Put-that-there“: Voice and Gesture at the Graphics Interface. In: *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, ACM Press, 1980

[33] Boreczky, J.S.; Rowe, L.A.: Comparison of Video Shot Boundary Detection Techniques. In: *Proceedings Storage and Retrieval for Image and Video Databases*, 1996

[34] Borgelt, C.; Kruse, R.: *Graphical Models: Methods for Data Analysis and Mining*. Jon Wiley & Sons, 2002

[35] Borman, S.: *The Expectation Maximization Algorithm – A short tutorial*. `http://www.seanborman.com/publications/EM_algorithm.pdf`. Version: 2004. – Arbeitspapier

[36] Brand, M.: Coupled Hidden Markov Models for Modeling Interacting Processes / MIT Media Lab Perceptual Computing. 1996 (405). – Forschungsbericht

[37] Buntine, Wray L.: Chain Graphs for Learning. In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 1995

[38] Campbell, J.P.: Speaker Recognition: a Tutorial. In: *Proceedings of the IEEE* 85 (1997), Nr. 9, S. 1437–1462

[39] Campbell, N.: *How to Follow a Conversation Without Listening to the Words*. 2005. – *Invited Talk at the 4th Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, Aufzeichnung verfügbar unter `http://www.idiap.ch/mmm/talk-webcast/mlmi-07`

[40] Carletta, J.; Ashby, S.; Bourban, S.; Flynn, M.; Guillemot, M.; Hain, T.; Kadlec, J.; Karaiskos, V.; Kraaij, W.; Kronenthal, M.; Lathoud, G.; Lincoln, M.; Lisowska, A.; McCowan, I.; Post, W.; Reidsma, D.; Wellner, P.: The AMI Meeting Corpus: A Pre-Announcement. In: *Proceedings of the 2nd Workshop on Machine Learning for Multimodal Interaction*, Springer-Verlag, 2006

[41] Chaisorn, L.; Chua, T.-S.; Lee, C.-H.: The Segmentation of News Video into Story Units. In: *Proceedings of the International Conference on Multimedia & Expo (ICME)*, 2002

[42] Chang, S.-F.; Ma, W.-Y.; Smeulders, A.: Recent Advances and Challenges of Semantic Image/Video Search. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2007

[43] Charniak, E.: Bayesian Networks without Tears: Making Bayesian Networks more Accessible to the Probabilistically Unsophisticated. In: *Artificial Intelligence Magazine* 12 (1991), Nr. 4, S. 50–63

[44] Cowell, R.: Advanced Inference in Bayesian Networks. In: Jordan, M.I. (Hrsg.): *Learning in Graphical Models*. MIT Press, 2001, S. 27 – 49

[45] Cowell, R.: Introduction to Inference for Bayesian Networks. In: Jordan, M.I. (Hrsg.): *Learning in Graphical Models*. MIT Press, 2001, S. 9 – 26

[46] Cowell, R.; Dawid, A.P.; Lauritzen, S.L.; Spiegelhalter, D.: *Probabilistic Networks and Expert Systems*. Corrected second edition. Springer-Verlag, 1999 (Statistics for Engineering and Information Science)

[47] Cox, R.T.: *The Algebra of Probable Inference*. Johns Hopkins University Press, 2002

[48] Croux, C.; Joossens, K.: Influence of Observations on the Misclassification Probability in Quadratic Discriminant Analysis. In: *Journal of Multivariate Analysis* 96 (2005), Nr. 2, S. 384–403

[49] Dawid, A.P.: Conditional Independence in Statistical Theory. In: *Journal of the Royal Statistical Society, Series B (Methodological)* 41 (1979), Nr. 1, S. 1–31

[50] Dempster, A.P.; Laird, N.M.; Rubin, D.B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. In: *Journal of the Royal Statistical Society B* 39 (1977), Nr. 1, S. 1–38

[51] Derin, H.; Kelly, P.A.: Discrete-Index Markov-Type Random Processes. In: *Proceedings of the IEEE* 77 (1989), Nr. 10, S. 1485 – 1510

[52] DiBiase, J.; Silverman, H.; Brandstein, M.: Robust Localization in Reverberant Rooms. In: Brandstein, M. (Hrsg.); Ward, D. (Hrsg.): *Microphone Arrays*. Springer Verlag, 2001, Kapitel 8, S. 157–180

[53] Dielmann, A.; Renals, S.: Dynamic Bayesian Networks for Meeting Structuring. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004

[54] Domingos, P.; Pazzani, M.J.: On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. In: *Machine Learning* 29 (1997), Nr. 2-3, S. 103–130

[55] Duda, R.O.; Hart, P.E.; Stork, D.G.: *Pattern Classification*. Second edition. Wiley & Sons, 2000

[56] Durbin, R.; Eddy, S.; Krogh, A.; Mitchison, G.: *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998. – Kapitel Pairwise alignment using HMMs, S. 80–99

[57] Edwards, D.: *Introduction to Graphical Modelling*. Corrected second edition. Springer-Verlag, 1995 (Springer Texts in Statistics)

[58] Eickeler, S.; Müller, S.; Rigoll, G.: Person-Independent Continuous Online Recognition of Gestures. In: *IEEE Conference on Computer Vision and Computer Vision*, 1998

[59] Evermann, G.; Chan, H.Y.; Gales, M.J.F.; Jia, B.; Mrva, D.; Woodland, P.C.; Yu, K.: Training LVCSR Systems on Thousands of Hours of Data. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005

[60] Fang, Z.; Guoliang, Z.; Zhanjiang, S.: Comparison of Different Implementations of MFCC. In: *Journal of Computer Science and Technology* 16 (2001), Nr. 6, S. 582–589

[61] Fine, S.; Singer, Y.; Tishby, N.: The Hierarchical Hidden Markov Model: Analysis and Applications. In: *Machine Learning* 32 (1998), Nr. 1, S. 41–62

[62] Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. In: *Annals of Eugenics* 7 (1936), S. 179–188

[63] Garg, A.; Balakrishnan, S.; Vaithyanathan, S.: Asynchronous HMM with Applications to Speech Recognition. In: *Proceedings of the 29th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004

[64] Ghahramani, Z.: On Structured Variational Approximations / University of Toronto. 1997, revised version 2002 (CRG-TR-97-1). – Forschungsbericht

[65] Ghahramani, Z.: Learning Dynamic Bayesian Networks. In: Giles, C.L. (Hrsg.); Gori, M. (Hrsg.): *Adaptive Processing of Sequences and Data Structures*. Springer-Verlag, 1998 (Lecture Notes in Artificial Intelligence), S. 168–197

[66] Ghahramani, Z.; Hinton, G.E.: Parameter Estimation for Linear Dynamical Systems / Department of Computer Science, University of Toronto. 1996 (CRG-TR-96-2). – Forschungsbericht

[67] Ghahramani, Z.; Hinton, G.E.: Variational Learning for Switching State-Space Models. In: Jordan, M.I. (Hrsg.); Sejnowski, T.J. (Hrsg.): *Graphical Models: Foundations of Neural Computation*. MIT Press, 2001, S. 315 – 348

[68] Ghahramani, Z.; Jordan, M.I.: Factorial Hidden Markov Models. In: *Machine Learning, Special issue on learning with probabilistic representations* 29 (1997), Nr. 2-3, S. 245–273

[69] Heckerman, D.: A Tutorial on Learning with Bayesian Networks. In: Jordan, M.I. (Hrsg.): *Learning in Graphical Models*. MIT Press, 2001, S. 301 – 354

[70] Heylen, D.; Nijholt, A.; Reidsma, D.: Determining what People Feel and Think when Interacting With Humans and Machines: Notes on Corpus Collection and Annotation. In: Kreiner, J. (Hrsg.); Putcha, C. (Hrsg.): *Proceedings 1st California Conference on Recent Advances in Engineering Mechanics*, 2006

[71] Hoashi, K.; Sugano, M.; Naito, M.; Matsumoto, K.; Sugaya, F.: Video Story Segmentation and its Application to Personal Video Recorders. In: *Proceedings 4th International Conference on Image and Video Retrieval (CIVR)*, 2005

[72] House, C.: *Visual Lexicons: The Quest for Data-Driven Decision Making*. 2005. – *Invited Talk at the 2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, Aufzeichnung verfügbar unter `http://groups.inf.ed.ac.uk/mlmi05/`

[73] Huang, C.; Darwiche, A.: Inference in Belief Networks: A Procedural Guide. In: *International Journal of Approximate Reasoning* 15 (1996), Nr. 3, S. 225–263

[74] Hörnler, B.: *Ein multi-modaler virtueller Konferenz-Regisseur*. 2006. – Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Diplomarbeit

[75] Isham, V.: An Introduction to Spatial Point Processes and Markov Random Fields. In: *International Statistical Review* 49 (1981), Nr. 1, S. 21–43

[76] Jaakkola, T.; Haussler, D.: Exploiting Generative Models in Discriminative Classifiers / Department of Computer Science, University of California. 1998 (1). – Forschungsbericht

[77] Jaakkola, T.; Jordan, M.I.: Improving the Mean Field Approximation Via the Use of Mixture Distributions. In: Jordan, M.I. (Hrsg.): *Learning in Graphical Models*. MIT Press, 2001, S. 163 – 174

[78] Janin, A.; Baron, D.; Edwards, J.; Ellis, D.; Gelbart, D.; Morgan, N.; Peskin, B.; Pfau, T.; Shriberg, E.; Stolcke, A.; Wooters, C.: The ICSI Meeting Corpus. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003

[79] Jensen, F.: *An Introduction to Bayesian Networks*. UCL Press, 1996

[80] Jensen, F.: *Bayesian Networks and Decision Graphs*. Second corrected edition. Springer Verlag, 2002 (Statistics for Engineering and Information Science)

[81] Jensen, F.; Lauritzen, S.; Olesen, K.: Bayesian Updating in Causal Probabilistic Networks by Local Computations. In: *Computational Statistics Quaterly* 4 (1990), S. 269–282

[82] Jordan, M.I. (Hrsg.): *Learning in Graphical Models*. Second Printing. MIT Press, 2001

[83] Jordan, M.I.; Ghahramani, Z.; Jaakkola, T.S.; Saul, L.K.: An Introduction to Variational Methods for Graphical Models. In: Jordan, M.I. (Hrsg.): *Learning in Graphical Models*. MIT Press, 2001, S. 105 – 161

[84] Jordan, M.I. (Hrsg.); Sejnowski, T.J. (Hrsg.): *Graphical Models: Foundations of Neural Computation*. MIT Press, 2001

[85] Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. In: *Transactions of the ASME – Journal of Basic Engineering, Series D* 82 (1960)

[86] Kaufmann, A.: *Einführung in die Graphentheorie*. Verlag R. Oldenbourg München und Wien, 1971 (Orientierung und Entscheidung)

[87] Kiiveri, H.; Speed, T.; Carlin, J. B.: Recursive Causal Models. In: *The Journal of the Australian Mathematical Society, Series A* 36 (1984), S. 30–52

[88] Kindermann, R.; Snell, J.: *Markov Random Fields and Their Applications*. American Mathematical Society, 1980

[89] Kjærulff, U.: Triangulation of Graphs - Algorithms Giving Small Total State Space / Department of Computer Science, Aalborg University, Denmark. 1990 (R90-09). – Forschungsbericht

[90] Knödel, W.: *Graphentheoretische Methoden und ihre Anwendungen*. Springer-Verlag, 1969 (Ökonometrie und Unternehmensforschung)

[91] Kschischang, F.R.; Frey, B.J.; Loeliger, H.-A.: Factor Graphs and the Sum-Product Algorithm. In: *IEEE Transactions on Information Theory* 47 (2001), Nr. 2

[92] Kullback, S.; Leibler, R.A.: On Information and Sufficiency. In: *Annals of Mathematical Statistics* 22 (1951), Nr. 1

[93] Lauritzen, S. L.: The EM Algorithm for Graphical Association Models with Missing Data. In: *Computational Statistics and Data Analysis* 19 (1995), Nr. 2, S. 191–201

[94] Lauritzen, S.L.: *Graphical Models*. Oxford Science Publications, 1996 (Oxford Statistical Science Series)

[95] Lauritzen, S.L.; Spiegelhalter, D.J.: Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. In: *Journal of the Royal Statistical Society, Series B* 50 (1988), S. 157 – 224

[96] Lenz, C.: *Das Asynchrone Hidden Markov Modell und seine Anwendung in der multimodalen Meetinganalyse*. 2007. – Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Diplomarbeit

[97] Liu, Q.; Kimber, D.: Learning Automatic Video Capture from Human's Camera Operations. In: *Proceedings of the International Conference on Image Processing (ICIP)*, 2003

[98] Liu, Q.; Shi, X.; Kimber, D.; Zhao, F.; Raab, F.: An Online Video Composition System. In: *Proceedings of the International Conference on Multimedia and Expo (ICME)*, 2005

[99] *The m4 project – Multimodal Meeting Manager, Project Leaflet*. `http://www.dcs.shef.ac.uk/spandh/projects/m4/pdf/leaflet_lowres.pdf`

[100] *The m4 project – Multimodal Meeting Manager*. `http://www.dcs.shef.ac.uk/spandh/projects/m4/index.html`

[101] MacKay, D.J.C.: Introduction to Gaussian Processes. In: Bishop, C. M. (Hrsg.): *Neural Networks and Machine Learning*. Kluwer, 1998 (NATO ASI Series), S. 133–166

[102] MacKay, D.J.C.: Introduction to Monte Carlo Methods. In: Jordan, M.I. (Hrsg.): *Learning in Graphical Models*. MIT Press, 2001, S. 175 – 204

[103] Manjunath, B.S.; Salembier, P.; Sikora, T.: *Introduction to MPEG-7. Multimedia Content Description Interface.: Multimedia Content Description Interface*. Wiley & Sons, 2002

[104] Maron, M.E.: Automatic Indexing: An Experimental Inquiry. In: *Journal of the ACM* 8 (1961), Nr. 3, S. 404–417

[105] Martinez, A.M.; Kak, A.C.: PCA versus LDA. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23 (2001), Nr. 2, S. 228 – 233

[106] McCowan, I.; Bengio, S.; Gatica-Perez, D.; Lathoud, G.; Monay, F.; Moore, D.; Wellner, P.; Bourlard, H.: Modeling Human Interaction in Meetings. In: *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2003

[107] McEliece, R.; Aji, S. M.: The Generalized Distributive Law. In: *IEEE Transactions on Information Theory* 46 (2000), Nr. 2, S. 325–343

[108] Moon, T.K.: The Expectation-Maximization Algorithm. In: *IEEE Signal Processing Magazine* 13 (1996), Nr. 6, S. 47 – 60

[109] Moore, D.: The IDIAP Smart Meeting Room / IDIAP. Version: 2002. `ftp://ftp.idiap.ch/pub/reports/2002/com02-07.pdf`. 2002 (07). – Forschungsbericht

[110] Morgan, N.; Baron, D.; Bhagat, S.; Carvey, H.; Dhillon, R.; Edwards, J.; Gelbart, D.; Janin, A.; Krupski, A.; Peskin, B.; Pfau, T.; Shriberg, E.; Stolcke, A.; Wooters, C.: Meetings About Meetings: Research at ICSI on Speech in Multiparty Conversations. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2003

[111] Murphy, K.: Switching Kalman Filters / Compaq Cambridge Research Lab. 1998 (98-10). – Forschungsbericht

[112] Murphy, K.: The Bayes Net Toolbox for Matlab. In: *Computing Science and Statistics* 33 (2001)

[113] Murphy, K.: *Dynamic Bayesian Networks: Representation, Inference and Learning*, University of California, Berkeley, Diss., 2002

[114] Murphy, K.; Paskin, M.: Linear Time Inference in Hierarchical HMMs. In: *Advances in Neural Information Processing Systems (NIPS) 13*, 2001

[115] Neapolitan, R.E.: *Learning Bayesian Networks*. Prentice Hall, 2003 (Artificial Intelligence Series)

[116] Nesvadba, J.; Ernst, F.; Perhavc, J.; Benois-Pineau, J.; Primaux, L.: Comparison of Shot Boundary Detectors. In: *Proceedings of the International Conference on Multimedia & Expo (ICME)*, 2005

[117] Nickles, M.: *PC, TV und Video total*. Franzis Verlag, 2002

[118] Oviatt, S.: Multimodal Interfaces. In: *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. Lawrence Erlbaum Associates, Inc., 2003, S. 286–304

[119] Oviatt, S.; Coulston, R.; Tomko, S.; Xiao, B.; Lunsford, R.; Wesson, M.; Carmichael, L.: Toward a Theory of Organized Multimodal Integration Patterns During Human-Computer Interaction. In: *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, ACM Press, 2003

[120] Pavlovic, V.: *Dynamic Bayesian Networks for Information Fusion with Applications to Human-Computer Interfaces*, University of Illinois at Urbana-Champaign, Diss., 1999

[121] Pavlovic, V.; Frey, B.; Huang, T.S.: Time Series Classification Using Mixed-State Dynamic Bayesian Networks. In: *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 1999

[122] Pearl, J.: Fusion, Propagation, and Structuring in Belief Networks. In: *Artificial Intelligence* 29 (1986), S. 241 – 288

[123] Pearl, J.: *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Revised second printing. Morgan Kaufmann Publishers, San Fransisco, California, 1988

[124] Pearl, J.: *Causality: Models, Reasoning, and Inference*. Second Reprint with corrections. Cambridge University Press, 2001

[125] Pearl, J.; Verma, T.: The Logic of Representing Dependencies by Directed Graphs. In: *Sixth National Conference on Artificial Intelligence* Bd. 1 American Association of Artificial Intelligence (AAAI), 1987

[126] Rabiner, L.R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: *Proceedings of the IEEE* 77 (1989), Nr. 2, S. 257–285

[127] Rabiner, L.R.; Juang, B.-H.: Theory and Implementation of Hidden Markov Models. In: *Fundamentals of Speech Recognition*. Prentice Hall PTR, 1993, S. 321 – 389

[128] Rauch, H.E.: Solutions to the Linear Smoothing Problem. In: *IEEE Transactions on Automatic Control* 8 (1963), S. 371–372

[129] Redner, R.A.; Walker, H.F.: Mixture Densities, Maximum Likelihood and the EM Algorithm. In: *SIAM Review* 26 (1984), Nr. 2, S. 195–239

[130] Reiter, S.; Schreiber, S.; Rigoll, G.: Multimodal Meeting Analysis by Segmentation and Classification of Meeting Events based on a Higher Level Semantic Approach. In: *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005

[131] Rigoll, G.: *Neuronale Netze. Eine Einführung für Ingenieure, Informatiker und Naturwissenschaftler*. Expert-Verlag, 1994

[132] Rigoll, G.: *Lecture Script Pattern Recognition*. 2006. – Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München

[133] Rijsbergen, C.J. van: *Information Retrieval*. London, UK : Butterworths, 1979

[134] Roweis, S.; Ghahramani, Z.: A Unifying Review of Linear Gaussian Models. In: *Neural Computation* 11 (1999), Nr. 2, S. 305–345

[135] Rui, Y.; Huang, T.: A Unified Framework for Video Browsing and Retrieval. In: Bovik, A. (Hrsg.): *Handbook of Image and Video Processing*. 2000, S. 705–715

[136] Sabri, S.; Prasada, B.: Video Conferencing Systems. In: *Proceedings of the IEEE* 73 (1985), Nr. 4, S. 671 – 688

[137] Salway, A.; Vassiliou, A.; Ahmad, K.: What Happens in Films. In: *Proceedings of the International Conference on Multimedia & Expo (ICME)*, 2005

[138] Shachter, R.D.: Bayes-Ball: The Rational Pastime (for Determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams). In: Cooper, G. (Hrsg.); Moral, S. (Hrsg.): *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference*, Morgan Kaufmann, San Francisco, 1998

[139] Shafer, G.; Shenoy, P.: Probability Propagation. In: *Annals of Mathematics and Artificial Intelligence* 2 (1990), S. 327–351

[140] Smeaton, A.; Over, P.: *TRECVID-2004: Shot Boundary Detection Task Overview*. 2004. – Result Slides of TRECVID

[141] Smeaton, A.F.: TRECVid - Video Evaluation. In: *American Society for Information Science and Technology (ASIST) Bulletin* (2007)

[142] Smeaton, A.F.; Over, P.; Kraaij, W.: TRECVID: Evaluating the Effectiveness of Information Retrieval Tasks on Digital Video. In: *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, ACM Press, 2004

[143] Smyth, P.: Belief Networks, Hidden Markov Models, and Markov Random Fields: a Unifying View. In: *Pattern Recognition Letters* 18 (1997), Nr. 11-13, S. 1261–1268

[144] Snidaro, L.; Niu, R.; Varshney, P. K.; Foresti, G. L.: Automatic Camera Selection and Fusion for Outdoor Surveillance under Changing Weather Conditions. In: *Proceedings of the Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2003

[145] Stadermann, J.: *Automatische Spracherkennung mit hybriden akustischen Modellen*, Technischen Universität München, Diss., 2005

[146] Sumec, S.: Multi Camera Automatic Video Editing. In: *Proceedings of the International Conference on Computer Vision and Graphics (ICCVG)*, Kluwer Verlag, 2004

[147] Tanner, R.: A Recursive Approach to Low Complexity Codes. In: *IEEE Transactions on Information Theory* 27 (1981), Nr. 5, S. 533 – 547

[148] Tarjan, R.E.; Yannakakis, M.: Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. In: *SIAM Journal on Computing* 13 (1984), Nr. 3, S. 566–579

[149] Uchihashi, S.: Direct Camera Control for Capturing Meetings into Multimedia Documents. In: *Proceedings of the International Conference on Multimedia and Expo (ICME)*, 2001

[150] Verma, T.; Pearl, J.: Causal Networks: Semantics and Expressiveness. In: *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, North-Holland Publishing Co., 1990

[151] Viterbi, A.: Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. In: *IEEE Transactions on Information Theory* 13 (1977), Nr. 2, S. 260 – 269

[152] Waibel, A.; Steusloff, H.; Stiefelhagen, R.: CHIL - Computers in the Human Interaction Loop. In: *Proceedings of the the NIST ICASSP Meeting Recognition Workshop*, 2004

[153] Wallhoff, F.; Zobl, M.; Rigoll, G.: Action Segmentation And Recognition in Meeting Room Scenarios. In: *Proceedings of the International Conference on Image Processing (ICIP)*, 2004

[154] Wang, Y.-Y.; Deng, L.; Acero, A.: Spoken Language Understanding. In: *IEEE Signal Processing Magazine* 22 (2005), Nr. 5

[155] Webers, J.: *Handbuch der Film- und Videotechnik*. 7. Auflage. Franzis Verlag, 2002

[156] Welch, G.; Bishop, G.: An Introduction to the Kalman Filter / Department of Computer Science, University of North Carolina at Chapel Hill. 2001 (TR 95-041). – Forschungsbericht

[157] Wellner, P.; Flynn, M.; Guillemot, M.: Browsing Recorded Meetings with Ferret. In: Renals, S. (Hrsg.); Bengio, S. (Hrsg.): *Proceedings of the first Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI)*, Springer Verlag, 2004

[158] Wellner, P.; Flynn, M.; Tucker, S.; Whittaker, S.: A Meeting Browser Evaluation Test. In: *CHI '05 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM Press, 2005

[159] Whittaker, J.: *Graphical Models in Applied Multivariate Statistics*. Jon Wiley & Sons, 1991 (Wiley Series in Probability & Mathematical Statistics)

[160] Williams, C.K.I.; Quinn, J.; McIntosh, N.: Factorial Switching Kalman Filters for Condition Monitoring in Neonatal Intensive Care. In: Weiss, Y. (Hrsg.); Schölkopf, B. (Hrsg.); Platt, J. (Hrsg.): *Advances in Neural Information Processing Systems (NIPS) 18*. MIT Press, 2006, S. 1513–1520

[161] Winkler, G.: *Image Analysis, Random Field and Markov Chain Monte Carlo Methods*. Second Edition. Springer, 2003 (Applications of Mathematics, Stochastic Modelling and Applied Probability)

[162] Woellmer, M.: *Development, Implementation and Evaluation of a Multidimensional DTW Algorithm for Multimodal Fusion of Potentially Asynchronous Data Streams*. 2007. – Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Bachelorarbeit

[163] Wood, D.R.: An Algorithm for Finding a Maximum Clique in a Graph. In: *Operations Research Letters* 21 (1997), Nr. 5, S. 211–217

[164] Wu, C.F.J.: On the Convergence Properties of the EM Algorithm. In: *The Annals of Statistics* 11 (1983), Nr. 1, S. 95–103

[165] Xu, L.; Jordan, M.I.: On Convergence Properties of the EM Algorithm for Gaussian Mixtures. In: *Neural Computation* 8 (1996), Nr. 1, S. 129–151

[166] Young, S.; Evermann, G.; Hain, T.; Kershaw, D.; Moore, G.; Odell, J.; Ollason, D.; Povey, D.; Valtchev, V.; Woodland, P.: *The HTK Book (for HTK Version 3.2.1)*. Cambridge University Engineering Department, 2002

[167] Zettl, S.: *Merkmalsextraktion und ein Vergleich regelbasierter Verfahren mit statischen Bayes'schen Netzen für die Ereignisdetektion in Videodaten*. 2005. – Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Bachelorarbeit

[168] Zettl, S.: *Schnitt- und Szenenerkennung in Videomaterial anhand von Low- und High-Level-Features mit einem hybriden Bayes'schen Netz*. 2005. – Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Diplomarbeit

[169] Zhang, D.: *Probabilistic Graphical Models for Human Interaction Analysis*, École Polytechnique Fédérale de Lausanne, Switzerland, Diss., 2006

[170] Zhang, D.; Gatica-Perez, D.; Bengio, S.; McCowan, I.; Lathoud, G.: Modeling Individual and Group Actions in Meetings: a Two-Layer HMM Framework. In: *Proceedings of the Second IEEE Workshop on Event Mining: Detection and Recognition of Events in Video, in Association with CVPR*, 2004

[171] Zhao, W.; Chellappa, R.; Phillips, P.J.; Rosenfeld, A.: Face Recognition: A Literature Survey. In: *ACM Computing Surveys* 35 (2003), Nr. 4, S. 399–458

[172] Zobl, M.; Laika, A.; Wallhoff, F.; Rigoll, G.: Recognition of Partly Occluded Person Actions in Meeting Scenarios. In: *Proceedings of the International Conference on Image Processing (ICIP)*, 2004

[173] Zobl, M.; Wallhoff, F.; Rigoll, G.: Action Recognition in Meeting Scenarios using Global Motion Features. In: Ferryman, J. (Hrsg.): *Proceedings Fourth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-ICVS)*, 2003