SPRECHER-KLASSIFIKATION IN VIDEOS

Thilo Stadelmann

Diplomarbeit

Sprecherklassifikation in Videos

zur Erlangung des akademischen Grades Diplom-Informatiker (FH)

vorgelegt dem Fachbereich Mathematik, Naturwissenschaften und Informatik der Fachhochschule Gießen-Friedberg

Thilo Stadelmann im August 2004

Referent: Professor Dr. P. Kneisel
Korreferent: Professor Dr. O. Hoffmann

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Pohlheim, 10. August 2004
Thilo Stadelmann

Vorwort

Wenn man wie ich seinen persönlichen Bildungsweg recht geradlinig über die Schule und das Abitur durch das Studium hin zum Diplom gegangen ist, stellt die Diplomarbeit doch die erste wirklich große, zusammenhängende Arbeit dar: Zum ersten Mal habe ich mich so intensiv und beinahe ausschließlich mit einem ganz speziellen Thema befasst. Das war teilweise berauschend, teilweise erdrückend, manchmal deprimierend und meistens spannend. Und ein bisschen Stolz spielt schon mit, wenn ich nun zurückblicke und sehe, wie es sich entwickelt hat.

Ein Wort zur Entstehungsgeschichte: Die vorliegende Arbeit ist nicht in erster Linie ein Resultat des in vier Jahren Studium angesammelten Wissens. Statt dessen war mir der Stoff des behandelten Fachgebietes über weite Strecken der Arbeit hinweg vorher unbekannt. Ich denke da beispielsweise, aber keineswegs ausschließlich, an die Mathematik, welche den statistischen Modellierungsansätzen innewohnt. Und hätte ich zu Anfang geahnt, was hiermit an fachlicher und außerfachlicher Leistung, aber auch an Disziplin und Geduld auf mich zu kommt, mich hätte wahrscheinlich schon vor dem Startschuss der Mut verlassen. Doch Stück für Stück, Herausforderung für Herausforderung war dieser Weg gangbar. Und das nicht zuletzt, da das nun zurückliegende Studium eben nicht nur die Fähigkeit zur Anwendung gesammelten Wissens vermittelte. Vielmehr lernte ich hier, dass schwierige Aufgaben auf der Grundlage des Gelernten auch lösbar sind, wenn weder ich selbst noch ein Tutor von vorneherein den Lösungsweg klar vor sich sehen. Und diese *Ahnung*, dass komplizierte Dinge trotzdem machbar sind, eben keine Hexerei oder nur "den Anderen" vorbehalten, hat sich Stück für Stück mit der Entwicklung dieser Arbeit in *Gewissheit* verwandelt.

Ich möchte an dieser Stelle noch die Möglichkeit nutzen, gerade denjenigen zu danken, deren nicht immer sichtbare Unterstützung maßgeblich dazu beigetragen hat, dass diese Arbeit in ihrer vorliegenden Form entstehen konnte: Ich danke...

Jesus Christus, dass er mich in dieser Zeit ganz besonders gesegnet und beschenkt hat. Professor Peter Kneisel, meinem Referenten, für seine "Bereitschaft zu allen Schandtaten", die mir viel Sicherheit gegeben hat. Professor Oskar Hoffmann, meinem Korreferenten, dass er so unkompliziert eingesprungen ist. Professor Bernd Freisleben von der Marburger Universität für das Vertrauen und die Möglichkeiten, die er mir mit diesem Thema gegeben hat. Diplom-Informatiker Ralph Ewerth dafür, dass er für alle Fragen und Vorschläge immer offen und ansprechbar war und mir als direkter Ansprechpartner in Marburg viel

seiner kostbaren Zeit spendete. Anne Schwalb für die vielen Stunden Diskussion über mathematische Spitzfindigkeiten ;-). Martin Schwalb für die vielen Stunden Diskussion über Informatiker-Spitzfindigkeiten. Manuel T. Schmidt für die Gestaltung der Umschlagseite und das Korrekturlesen der Arbeit. Tobias J. Schurr und Anne Schwalb ebenfalls für das Korrekturlesen der Arbeit. Rebekka Bachmann, meinen Freunden und meiner Familie für ihre Geduld und offene Ohren. Danke Ihnen/Euch allen!

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

h

Kapitel 1

Einleitung

1.1 Motivation

"Was als ein Strom nützlicher Informationen begann, ist zu einer Sturzflut geworden."

Neil Postman[1]

Es gibt sie in Wort und Schrift, in Film und Bild, Audio und Video: Medien. Sie sind ein großes Rad im Getriebe der modernen Gesellschaft. Sie transportieren Tatsachen genauso wie Meinungen, vermitteln Information ebenso durch Erwähnung als auch durch Auslassung von Meldungen. Medien sind wichtig heutzutage. So wichtig, dass sich eine eigene Wissenschaft mit ihnen beschäftigt: Die *Medienwissenschaft*.

Die Medienwissenschaft analysiert alle Arten von Medien anhand folgender Fragestellung: "Wer, sagt Was, zu Wem, in Welchem Kanal, mit Welchem Effekt"[2]. Doch die immer größer werdende Anzahl an gespeichertem und neu produziertem Informationsmaterial wird immer unüberschaubarer.

Hier setzt das Projekt *"Medienumbrüche – Methoden und Werkzeuge zur rechnergestützten medienwissenschaftlichen Analyse"* an: Gefördert von der Deutschen Forschungsgemeinschaft[3] und beheimatet an der Phillips-Universität Marburg[4], will es Medienwissenschaftler mit modernster Informationstechnologie für ihre Arbeit ausrüsten. Die vorliegende Diplomarbeit gliedert sich als einzelner Beitrag in dieses Projekt ein.

Das sichtbare Ergebnis des Gesamtprojekts ist die Software *"Mediana"*. Sie vereint Ansätze zur Untersuchung von Video-Daten unter einer einheitlichen Oberfläche. Dabei

[1] *Neil Postman*, amerikanischer Medienkritiker, *1931 [ZITA 04].

[2] Die sogenannte *Laswell-Formel*, nach Harold D. Laswells Modell der Massenkommunikation von 1948. Sie steckte den Rahmen der Kommunikationswissenschaft, dem heutigen theoretischen Teilgebiet der Medienwissenschaft, ab [WIKI 04a].

[3] Als Teil des DFG-Sonderforschungsbereichs 615: *"Medienumbrüche, Medienkulturen und Medienästhetik zu Beginn des 20. Jahrhunderts und im Übergang zum 21. Jahrhundert"* [DFG 02].

[4] Wegen des interdisziplinären Charakters des Projekts sind viele verstreute Forschergruppen an dessen Bearbeitung beteiligt. Der informatikspezifische Anteil residiert dabei in Marburg.

benutzt sie das *MPEG-7*-Metaformat, um diese Daten direkt mit den gewonnenen Informationen anzureichern. Die Liste der verfügbaren Analysemöglichkeiten reicht dabei von Schnitt- und Szenenwechselerkennung über die Detektion und Trennung von Hintergrund und Objekten bis zur Extraktion eingeblendeter Texte oder der Auffindung und Zuordnung im Bild befindlicher Gesichter.

Nun sind Videos jedoch meistens multimedial ausgelegt: Neben dem obligatorischen Bilderstrom enthalten quasi alle aus dem Alltag bekannten Filme auch Tondaten: Fernsehsendungen, Kinofilme, sogar die Aufzeichnungen kleiner Digitalkameras bieten dem Zuschauer zusätzlich einen akustischen Reiz. Die im Rahmen dieser Diplomarbeit entwickelte Programmbibliothek *"SC_Lib"* wendet sich als erster Teil des Projekts Medienumbrüche diesem Audio-Datenstrom zu. Ziel der Bemühungen soll – dies sei dem nächsten Kapitel schon vorweg genommen – eine Zuordnung von Sprachabschnitten zu Sprechern sein.

Von dieser Aufgabenstellung geht eine ganz besondere Faszination aus: Zum einen ist die Anwendung der Ergebnisse vielseitig einsetzbar. Sei es nun die Indizierung und späteren Abfrage großer Videoarchive nach einzelnen Sprechern, oder die Identifizierung des Menschen anhand des biometrischen Merkmals Stimme – beide Szenarien sind mit dem hier betrachtetetn verwandt, momentan stark im Wachstum begriffen, und werden in naher Zukunft wohl noch an Bedeutung gewinnen. Zum anderen stellt sich so eine fachlich höchst interessante Aufgabe, denn im Schnittgebiet von Informatik, Mathematik, Linguistik und vielen weiteren Wissenschaften ist noch viel Raum für eigene Ideen und Ansätze. Die Disziplin ist relativ jung, und die Herausforderung harrt ihrer Meisterung. Denn die wenigen bisher erschienen Veröffentlichungen mit ähnlicher Zielsetzung wie die vorliegende Arbeit konnten gemeinsam noch keine befriedigenden Ergebnisse vorweisen[5].

1.2 Zielsetzung

Ein wenig muss die letzte Aussage des vorangegangenen Kapitels doch noch relativiert werden: Trotz ihres geringen Alters ist in der Disziplin Sprachverarbeitung schon sehr viel geleistet worden. Vieles, was noch kürzlich als reine Zukunftsmusik galt, ist heute realisierbar. Kapitel 2 wird sich ausführlicher mit dieser Thematik beschäftigen.

Eine Sache aber zumindest blieb bis heute weitgehend unangetastet: Der Umgang mit Freiheit. Unter bekannten und vor allem beschränkten Voraussetzungen funktionieren Sprach- und Sprechererkennung schon sehr gut. Wenn die Stimme bekannt, das Umfeld fix und die Aufnahmebedingungen kontrollierbar sind, führen bewährte Techniken zu verblüffenden Erfolgen. Kommt aber Unsicherheit ins Spiel – Freiheit im Bezug auf die Anzahl und Identität der Sprecher oder die Geräuschkulisse während der Aufzeichnung –

[5] [NISH 99] beispielsweise versuchte sich an der Klassifikation von Sprechern in Filmen, erreichte jedoch nur Recall- und Precision-Raten um die 60 Prozent, woraufhin die beteilgten Forscher sich wieder aussichtsreicheren Gebieten zuwanden [NISH 03b]. [LI 03] ging das Problem des Arbeitens auf Filmen durch sehr viele gemachte Einschränkungen an, warf jedoch mehr Fragen auf, als gelöst werden konnten.

sind neue Techniken von Nöten.

Die eigentliche Aufgabenstellung dieser Arbeit ist nun die Implementierung einer Software, welche Videos anhand der auftretenden Sprecher indizieren kann. Diese Software soll als Baustein in das Programmpaket Mediana integriert werden und somit tatsächlich "in freier Wildbahn" zum Einsatz kommen. Der vorliegende schriftliche Aspekt integriert sich in diesen Kontext, indem er das Umfeld des zu erstellenden Programms, dessen Entwicklung sowie seine Leistungsfähigkeit beschreibt. Aber wie es sich für einen Aspekt gehört, kann er nicht für sich alleine stehen: Diese Zeilen als gebundenes Büchlein und das in ihnen beschriebene Programm bilden gemeinsam diese Diplomarbeit.

Was ist nun genau zu verstehen unter einem Programm, welches Videos anhand der auftretenden Sprecher indiziert? Sprachsegmente, die von ein und demselben Sprecher geäußert wurden (durchaus in unterschiedlichen Szenen, unter verschiedenen Bedingungen), sollen identisch markiert werden. Sprachsegmente unterschiedlicher Sprecher sollen dementsprechend verschiedene Marken bekommen. Der Begriff "Video" soll hierbei so allgemein wie möglich verstanden werden: Wenn mit Ton untermalte, bewegte Bilder Sprache enthalten, soll eine Bearbeitung möglich sein, weitestgehend unabhängig von

- der Anzahl der Sprecher,
- der Identität der Sprecher (sie ist vorher unbekannt),
- der Länge der Sprachabschnitte,
- den Aufnahmebedingungen und
- dem akustischen Hintergrund der Aufnahme.

Wie bereits angedeutet, erfordert dies nicht nur die Anwendung existierender Standardverfahren, sondern stellt einen noch jungen Forschungsbereich dar. "Entwicklung einer Software" heißt in diesem Kontext also auch "Entwicklung neuartiger Verfahren", in einem Wort: Forschung.

1.3 Weiteres Vorgehen

Der Rest dieser Arbeit gliedert sich wie folgt:

Kapitel 2 ordnet das Thema Sprecherklassifikation in den Kontext seines Forschungsbereichs ein und zeigt dessen historische Wurzeln und Entwicklung auf. Auf diese Weise wird die Fragestellung auch gegen benachbarte Gebiete abgegrenzt. Anschließend wird ein Überblick über den Stand der Forschung in den relevanten Bereichen gegeben. Für die einzelnen Ansätze findet dabei jeweils eine kurze Diskussion im Hinblick auf die Relevanz für diese Arbeit statt.

Im darauf folgenden Kapitel 3 geht es um das Design der zu erstellenden Software: Worauf baut sie auf, wie geht sie vor? Das beinhaltet sowohl den softwaretechnischen Entwurf wie auch die Frage nach den verwendeten Verfahren. Abschnitt 3.4 beschreibt dann den Kern der Entwicklungsarbeit: Neue Ideen und Ansätze zur Verbesserung, Erweiterung und Rekombination bewährter Verfahren werden hier im Detail dargestellt und erläutert.

Kapitel 4 knüpft daran an, indem dort Eigenheiten der konkreten Implementierung betrachtet werden: Von der Wahl der Sprache und Mittel bis hin zu programmiertechnischen Besonderheiten der zuvor dargestellten Algorithmen. Des Weiteren gehört zur Realisierung, dass hier die Erfolge der Bemühungen überprüft werden: Ausgehend von den zuvor formulierten Zielen wird eine Testsystematik entwickelt und die daraus resultierenden Ergebnisse werden präsentiert.

Um Zusammenfassung und Ausblick geht es in Kapitel 5: Hier werden die erzielten Ergebnisse diskutiert und interpretiert, es erfolgt ein Ausblick auf mögliche zukünftige Arbeiten und die Einbindung des Erreichten in bestehende Systeme.

Die Anhänge enthalten schließlich all das, was überdies nützlich ist: Anhang A beschreibt einige mathematische Herleitungen und Beweise, die zum besseren Verständnis der verwendeten Verfahren beitragen, aber hier nicht zum ersten Mal veröffentlicht werden. Außerdem enthält er eine Formelsammlung für den im weiteren Verlauf wichtig werdenden Mechanismus des *Gauß'schen Misch-Modells mit Integriertem Hintergrund*[6]. Anhang B gibt einen Überblick über die zu Testzwecken mit-entwickelte graphische Benutzeroberfläche "Gardener". Darauf folgt in Anhang C eine kurze Übersicht über die Organisation des dieser Arbeit beiligenden Datenträgers. Anhang D trägt der Tatsache Rechnung, dass beinahe die gesamte Fachliteratur im bearbeiteten Gebiet ausschließlich in englischer Sprache verfügbar ist. So wird an dieser Stelle den in dieser Arbeit meist verwendeten deutschen Begriffen und Akronymen ihr fremdsprachliches Pendant bzw. ihr voller Wortlaut gegenübergestellt.

Den Abschluss bilden das Verzeichnis verwendeter Quellen sowie ein Index zum schnellen Auffinden wichtiger Begriffe und Passagen.

[6] Abschnitt 2.4.5 bringt hierzu eine Einführung.

Kapitel 2

Grundlagen

2.1 Kontextuelle Einordnung

Sprecherklassifikation ist als Teilgebiet der *Sprachverarbeitung* zuzuordnen. Diese ist keine ureigene Informatik-Domäne, sondern deutlich interdisziplinär ausgelegt: Verschiedene Wissenschaften tragen gemeinsam zum besseren Verständnis der Vorgänge bei der Sprachproduktion und -verarbeitung durch Menschen bei, um das dedizierte Ziel gemeinsam zu erreichen: Robuste maschinelle Analyse gesprochener Sprache [RABI 93].

So liefern Biologie und Medizin die anatomischen Grundlagen, anhand derer statistische Modelle der akustischen Ereignisse durch Mittel der höheren Mathematik erstellt werden können. Die Linguistik trägt zum Verständnis des Aufbaus und der Konzeption einzelner Sprachen, ihrer Unterschiede und Gemeinsamkeiten, bei. Und aus der Psychologie stammen Erklärungsmodelle für die Vorgänge beim menschlichen Sprachverständnis. Die Informatik schließlich stellt die Methoden bereit, um dieses gesammelte Wissen in algorithmischer Form für eine Maschine einsetzbar zu machen. Dazu benötigt sie wiederum die Ergebnisse der Signalverarbeitung, einer Disziplin im Schnittpunkt von Physik und Elektrotechnik, um die akustischen Impulse der Sprache in maschinenlesbare elektrische Signale zu verwandeln.

Sprachverarbeitung gliedert sich nun ihrerseits in mehrere Forschungsbereiche auf, je nach verfolgter Fragestellung. Zuerst muss die Unterscheidung zwischen *Sprachsynthese*, *-anreicherung* sowie *-erkennung* gemacht werden: Soll Sprache künstlich erzeugt oder zum Zweck der Verfremdung umgestaltet werden, handelt es sich um einen Fall für den Bereich Sprachsynthese. Mit Sprachanreicherung hingegen ist die Modifikation der Sprache zum besseren Verständnis durch den Hörer zu gemeint. Dies kann unter anderem die Reduktion von Nebengeräuschen beinhalten. Im Zweig der Erkennung kann nun auf die veschiedenen Informationsebenen der Sprache eingegangen werden: Während die Frage nach dem Inhalt zur wirklichen *Spracherkennung* führt, kann natürlich auch die Herkunft des Signals, die Information über die Identität des Sprechers, ausgewertet werden. Daneben existieren natürlich noch weitere Fragestellungen, die jedoch eine untergeordnete Rolle spielen. Abbildung 2.1 gibt hier einen Überblick.

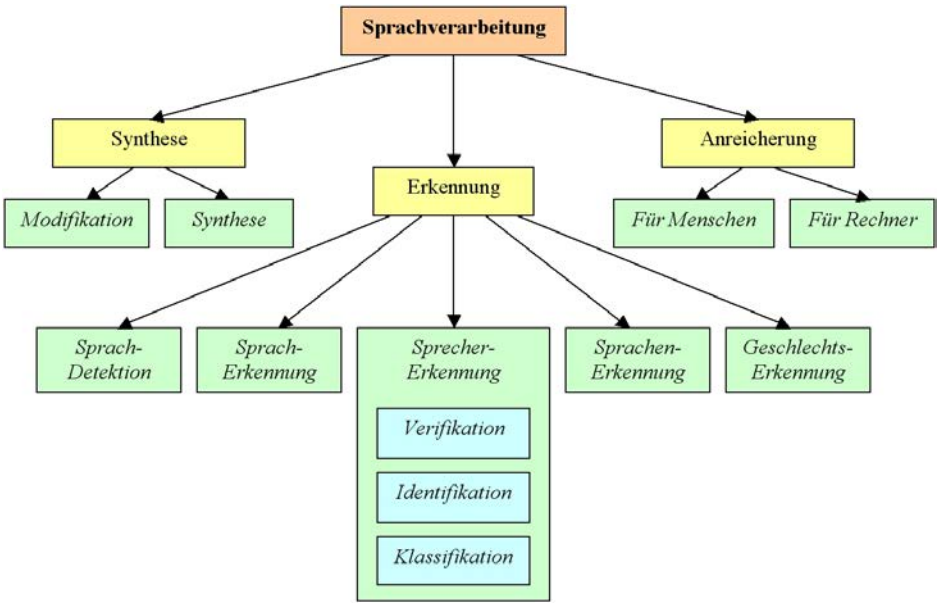

Abbildung 2.1: Ein Blick über das Gebiet der Sprachverarbeitung (nach [DUNN 03]).

Mit der *Sprechererkennung* ist nun schon fast das Thema dieser Arbeit erreicht – die weitere Unterscheidung erfolgt eher aufgrund der Komplexitätsstufe der Aufgabenstellung denn aufgrund ihrer Andersartigkeit: *Sprecher-Verifikation* untersucht, ob die Identität eines erkannten Sprechers mit einer gegebenen Identität übereinstimmt. Diese binäre Entscheidung ist beispielsweise bei biometrischen Zutrittskontrollen sinnvoll, wo sich eine Person mit einem Ausweis (die "gegebene" Identität) und ihrer Stimme authentifizieren muss. *Sprecher-Identifikation* geht darüber hinaus, indem zu einem gegebenen Stimm-Muster keine weitere Angabe gemacht wird. Es ist nun Aufgabe des Systems, die Identität des Sprechers anhand einer Datenbank mit Referenzmustern zu ermitteln oder aber die Entscheidung zu treffen, die Person sei unbekannt. In diesem Fall spricht man von *"open set"* Identifikation.

Der letzte Fall, *Sprecherklassifikation*, erweitert nun die Sprecheridentifikation um eine zeitliche Komponente: Ein beliebiger akustischer Eingabestrom wird auf das Vorhandensein von Sprachabschnitten untersucht. Ein gefundener Abschnitt wird gegebenenfalls in Segmente einzelner Sprecher unterteilt. Nun wird für jeden Abschnitt die Identität des Sprechers festgestellt und so ein Index für den Eingabestrom geschaffen, welcher jedem Zeitpunkt genau einen Sprecher oder aber die Abwesenheit von Sprache zuordnet.

Die Details können hierbei natürlich je nach Material und konkreter Aufgabenstellung variieren, was sich teils deutlich in den zu verwendenden Verfahren niederschlägt: Sind die Anzahl und Identitäten der im Strom enthaltenen Sprecher vorher bekannt? Sind alle zu untersuchenden Materialien von ähnlichem Typ, was Qualität und Nebengeräusche angeht? Sind die zu erwartenden Texte bekannt oder sind die Sprecher kooperativ (sind

sie der Erkennungsbemühungen eines Automaten gewahr und unterstützen dies)?

Auch was die Nomenklatur der Verfahren angeht, herrscht nicht unbedingt Einheitlichkeit: Was in dieser Arbeit durchweg Sprecherklassifikation genannt wird, tritt teilweise auch unter den Bezeichnungen *Sprecher-Verfolgung*, *Sprecher-Indizierung* oder *Sprecher-Clustering* auf. Hier steht jeweils das verfolgte Ziel bzw. ein verwendetes Verfahren als Namensgeber Pate.

2.2 Historische Wurzeln

Die Forschung im Bereich der Sprachverarbeitung begann vor circa 40 Jahren mit dem Teilgebiet der Spracherkennung. Aus dessen Methoden entstand bald darauf die Sprechererkennung [CAMP 97]. Die übergreifende Verwendung identischer Ansätze und Werkzeuge in den beiden Gebieten beruht auf der Tatsache, dass natürlich beide Arten von Information – das Gesprochene wie auch den Sprecher betreffend – in ein- und demselben Audiostrom kodiert sind: Während dessen Kurzzeit-Analyse Informationen über Phoneme[1], Worte, schließlich Sätze enthüllt, zeigt eine Beobachtung derselben Merkmale über längere Zeit stark sprecherspezifische Eigenschaften. In Abschnitt 2.4 werden diese Zusammenhänge noch näher beleuchtet.

Im Verlauf der parallelen Entwicklung der Sprach- und Sprechererkennung konnte nicht zuletzt durch ständige Zunahme der Leistungsfähigkeit von Computern eine stetige Verbesserung der Verfahren erfolgen. Denn viele der eingesetzten Signalverarbeitungs- und Modellierungstechniken verzehren sich nach Rechenleistung, so dass Bearbeitung in Echtzeit beispielsweise erst in jüngster Vergangenheit erreichbar wurde [LU 02c].

Auch die Kerngebiete der Forschung und Anwendung haben sich im Lauf der Zeit gewandelt. Während der Pionierzeit war an Erkennung gesprochener, kontinuierlicher Sprache kaum zu denken, man arbeitete an und mit kurzen, bekannten Phrasen (einzelnen Ziffern zum Beispiel). Mittlerweile sind jedoch kontinuierliche Spracherkenner mit großem Vokabular sogar in Form von Standardsoftware für jedermann käuflich erwerbbar[2]. Dennoch ist die Forschung hier noch nicht abgeschlossen.

In den 1980er Jahren, bis hinein in die spätern 1990er Jahre, war immer mehr das Telefon als Medium in das Zentrum der Aufmerksamkeit gerückt. Aufkommendes Homeshopping, der erste Vorläufer des heutigen internetbasierten e-Commerce, wurde über Telefonverbindungen abgewickelt, und es entstand enormer Bedarf an Automatisierungssystemen [CAMP 97]. So befasste man sich auch zum ersten Mal mit dem Thema Nebengeräusche und Robustheit, da die Qualität telefonisch übertragener Sprache zu sehr zu

[1] Unter einem *Phonem* versteht man die kleinste Einheit einer Sprache, die über Bedeutungsunterschiede entscheidet [WIKI 04a].

[2] Zu erwähnen sind hier hauptsächlich die Systeme *ViaVoice* von IBM sowie *Dragon Naturally Speaking* der Firma ScanSoft.

wünschen übrig ließ. Die Ausläufer dieser Bemühungen sind heute noch im Einsatz und lassen sich in Aktion erleben, wenn man beispielsweise die automatische Fahrplanauskunft der Deutschen Bahn anruft oder sich mit telefonischer Kontoführung bei diversen 24-Stunden-Banken beschäftigt.

Eine weitere Anwendung, ebenfalls aufgekommen in den 1980er Jahren und bis heute aktuell, ist Sprechererkennung zum Einsatz in Sicherheits- und Zutrittskontrollsystemen [FENG 02]. Durch die Sicherheitsoffensive im Zuge der Ereignisse des 11. Septembers 2001 wurde gerade die Entwicklung solcher biometrischer Verfahren stark forciert, doch schon seit 1995 führt das NIST[3] jährliche Wettbewerbsrunden durch, um dieser Technologie aus den Kinderschuhen zu helfen.

In letzter Zeit ins Zentrum gerückt ist jedoch eine Anwendung, bei der es um die Beschaffung, Bereitstellung und den Zugriff auf multimediale Daten geht: "Multimedia Document Retrieval". Durch die enorme Flut an digital verfügbaren Inhalten stieg der Bedarf an Techniken, um die Verwaltung und effiziente Nutzung der Dokumente zu ermöglichen. Hier findet teilweise eine Vereinigung von Sprach- und Sprechererkennung statt [VISW 00], oder aber auch die Verbindung von Audio- und Videoinformationen [TSEK 98], um bessere Leistungen durch die Ausnutzung sämtlicher Informationsquellen zu gewährleisten.

2.3 Typischer Aufbau eines Sprecherklassifikations-Systems

Sprecherklassifikations-Systeme sind typischerweise aus einigen speziellen Modulen aufgebaut, wobei die genaue Zusammensetzung natürlich innerhalb bestimmter Parameter aufgabenspezifisch ist. Dabei unterscheidet sich der Aufbau nicht sehr von demjenigen anderer Systeme aus der Sprachverarbeitungs-Familie. Abbildung 2.2 zeigt die übliche Abfolge von Aktionen und deren Ergebnisse skizzenhaft auf:

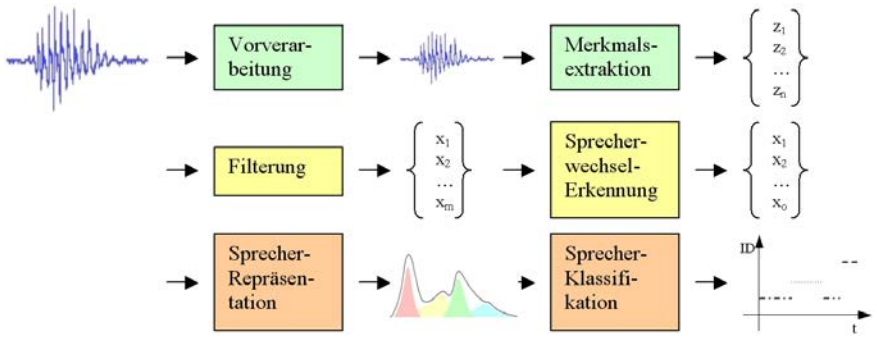

Abbildung 2.2: Typischer Ablauf eines Sprecherklassifikations-Laufs.

[3]Das amerikanische *"National Institute of Standards & Technology"*, [REYN 00].

Nachfolgend soll auf die einzelnen Phasen kurz eingegangen werden:

Mit *Vorverarbeitung* ist die Behandlung und Aufbereitung des Signals innerhalb oder außerhalb des Systems gemeint, die im allgemeinen ein irgendwie geartetes Rohsignal auf ein genau spezifiziertes Level bringt. Beispielsweise könnte hier die Sample-Rate angeglichen werden, eine Vorverstärkung vorgenommen oder das Signal bandbegrenzt werden. Wenn zum Beispiel Sprache in Telefonqualität untersucht werden soll, bewegt sich deren *Spektrum* nur im Bereich zwischen 300 und 3400 Hertz, eine Limitierung auf dieses Frequenzband eliminiert so auf jeden Fall schon irrelevante und störende Signalanteile.

Darauf folgt die Stufe der *Merkmalsexktraktion*: Der hochdimensionale Raum der Rohdaten (des Signals) mit seiner großen Menge an Information wird derart auf einen niederdimensionalen Raum abgebildet, dass die Menge der Daten rapide sinkt, ohne dass der Informationsgehalt in den für die Aufgabenstellung relevanten Bereichen merklich abnimmt. Konkret wird hier also die schiere Menge an auszuwertender Informtionen insgesamt verkleinert, indem beispielsweise nicht mehr 16.000 mal pro Sekunde ein Messwert (Sample) vorliegt, sondern ein *Merkmalsvektor* die Ereignisse einiger Millisekunden zusammenfasst. An dieser Stelle eröffnet sich eine der Hauptmöglichkeiten zur Verbesserung des Gesamtsystems [ELLI 03]: Alle nachfolgenden Stufen arbeiten nur noch auf den Merkmalsvektoren anstatt auf dem Signal. Wenn hier das Wesentliche zwingend gemacht wurde, also eine Transformation in einen geeigneten *Merkmalsraum* stattfand, fällt die anschließende Klassifikation leicht, andernfalls kann sie misslingen. Eine sehr gute Veranschaulichung hierzu findet sich in [CAMP 97], Seite 1450-1451.

Unter *Filterung* sollen im Folgenden all jene Verarbeitungsschritte zusammengefasst sein, welche die Menge der Merkmalsvektoren im Nachhinein beeinflussen: Hier geht es um die im Bezug auf die Merkmalsextraktion nachträgliche Eliminierung irrelevanter Information. Beispielsweise könnten hier Stille oder ganz allgemein nicht-sprachliche Anteile entfernt werden (Segmentierung), oder die für die Weiterverarbeitung störenden *nichtstimmhaften* Anteile der Sprache. Nähere Erläuterungen hierzu finden sich auch in Abschnitt 2.4.2).

Die *Sprecherwechsel-Erkennung* untersucht die übriggebliebenen Sprachabschnitte auf Wechsel innerhalb der Identität der Sprecher. An dieser Stelle geht es noch nicht um eine endgültige Identifikation oder gar Klassifikation der Sprecher. Vielmehr ist es der Zweck dieser Stufe, möglichst lange, zusammenhängende Sprachsegmente zu erzeugen, die mit Sicherheit von einem einzigen Sprecher stammen. Dies arbeitet der anschließenden Repräsentations- und Klassifikationsstufe zu, die jeweils so viele Daten wie möglich über einen Sprecher verarbeiten sollte, um die Grundlage für die statistische Analyse zu schaffen, gleichzeitig aber natürlich auf möglichst "reine" Segmente angewiesen ist [BONA 02]. Allerdings gibt es auch die entgegengesetzte Meinung, dass die Segmentierung als Vorverarbeitungsschritt nur zur Fehlermultiplikation beiträgt und besser mit der Klassifikation in einem Schritt geschehen sollte [AJME 03].

Die *Repräsentationsstufe* zielt wieder auf eine Dimensionalitätsreduktion ab: Hier soll, abstrahierend von den einzelnen Merkmalsvektoren, ein *Modell* eines einzelnen Sprechers erstellt werden, das auf eine bestimmte Art und Weise – Abschnitt 2.4.4 geht darauf genauer ein – diesen Sprecher dem System allgemein bekannt macht und dessen Besonderheiten kompakt darstellt.

Während der *Klassifikation* werden nun die erstellten Modelle miteinander verglichen, um einander ähnliche Modelle anhand eines Kriteriums zu vereinen und somit auszusagen, dass deren Sprecher identisch sind. Als Ergebnis entsteht der schon erwähnte Index des Ausgangsmaterials anhand der Sprecheridentitäten.

Der modulare Aufbau eines Sprecherklassifikators hat seine Vorteile: Einzelne Verarbeitungsschritte lassen sich als Blackboxes realisieren, sind somit in ihrer Entwicklung unabhängig voneinander und als Glied der Verarbeitungskette austausch- und erweiterbar. Ein derart aufgebautes System ist einfach zu warten und zu ergänzen - es entspricht also auch softwaretechnischen Anforderungen, einmal abgesehen von der Tatsache, dass es der Algorithmik sehr entgegenkommt. Diese Ideen werden beispielsweise auch in [BECC 99] vertreten. Der Nachteil einer solchen gliedweisen Verarbeitung ist natürlich, dass sich früh gemachte Fehler in die nachfolgenden Stufen durchschleifen und so einen multiplikativen Einfluss auf das Gesamtergebnis haben: Leisten beispielsweise alle sechs Module in Abbildung 2.2 eine Genauigkeit von 90 Prozent, degradiert dies die Gesamtleistung des Systems zu vielleicht überraschend geringen $0.9^6 \approx 53$ Prozent.

2.4 Stand der Forschung

2.4.1 Merkmalsextraktion

Wenn man fachfremden Personen von dem Gebiet der automatischen Sprachverarbeitung und den Herausforderungen der Sprechererkennung erzählt, ist die erste Reaktion meist wie folgt: *"Wie, Sprecher erkennen? Da muss man doch bloß die Frequenzen in der Stimme beobachten, dann hat man's doch schon. . . "*. Und im Prinzip ist dieser intuitive Ansatz auch gar nicht verkehrt, da er doch direkt aus der Biologie abgeleitet ist: Das menschliche Hörvermögen basiert fast ausschließlich auf den Frequenzen wahrgenommener Geräusche [MÄKI 00].

Etwas Signalverarbeitung

Zur Ermittlung der im Signal enthaltenen Frequenzen ist es nötig, dessen *Spektrum*[4] zu

[4]Das *Spektrum* eines Signals, auch als Amplituden- oder Powerspektrum bezeichnet, lässt sich als eine Art Histogramm vorstellen: Allen möglichen Tonhöhen (Frequenzen) sind deren Ausprägung (Amplituden) im untersuchten Signalabschnitt zugeordnet. Diese Darstellungsform unterschlägt allerdings die Information des Phasengangs [HOFF 03].

ermitteln. Dies kann nicht auf Basis einzelner Messwerte (Samples) geschehen, weshalb ein bestimmter Signalabschnitt zu einem Rahmen (Frame) zusammengefasst und analysiert wird. Die Größe des Rahmens entscheidet dabei sowohl über die Genauigkeit des ermittelten Spektrums als auch über die zeitliche Auflösung des Verfahrens, wobei zwischen beidem ein Zusammenhang besteht: Je größer der Rahmen, desto genauer die Auflösung des Frequenzgangs, aber desto niedriger auch die Genauigkeit der zeitlichen Auflösung. Um dem entgegenzuwirken, werden häufgig einander überlappende Rahmen eingesetzt: Der nächste Rahmen beginnt nicht am Ende seines Vorgängers, sondern wird nur um einen Bruchteil der Rahmenlänge weitergeschoben. Dies erlaubt relativ lange Rahmen bei gleichzeitig hoher Zeitauflösung.

Es existieren verschiedene Verfahren zur Bestimmung des Spektrums. Die offensichtliche Variante ist die Filterung des Signals mittels einer Filterbank: Trianguläre Bandpass-Filter im Abstand der gewünschten Frequenzauflösung lassen jeweils nur den Signalanteil passieren, der ihrem Frequenzband entspricht. Ein Aufsummieren der Ausgaben dieser Filter liefert die Amplitude für die entsprechende Frequenz [RABI 93].

Die andere Variante, welcher im Fall digitaler Analyse meist der Vorzug gegeben wird, beruht auf der *Fourier-Analyse* des Signals: Hierbei wird ein stationäres[5] Signal durch eine Reihenentwicklung mittels bestimmter Funktionen angenähert. Aus den enstehenden Koeffizienten ergeben sich dann Amplitude und Frequenz der im Signal enthaltenen Subbänder. Ein Problem ist, dass Sprache an sich kein stationäres Signal ist. Durch die Bearbeitung kurzer Rahmen wie oben beschrieben lässt sie sich jedoch als quasi-stationär auffassen. Wenn nun für die "bestimmte Funktion" in der Reihenentwicklung eine Sinus- oder Cosinusfunktion eingesetzt wird, lässt sich der Algorithmus der Fast Fourier Transformation (FFT) anwenden, der seit vielen Jahren als das Standardverfahren der Signalverarbeitung gilt. In letzter Zeit tauchten jedoch immer mehr Veröffentlichungen zum Thema *Wavelet-Analyse* auch im Zusammenhang mit Sprachverarbeitung auf [PHAM 02]: Die Verwendung nahezu beliebiger Funktionsabschnitte[6] zur Reihenentwicklung bietet mancherlei Vorzüge und überwindet Beschränkungen der FFT. Der deutlichste ist hier wohl die Verbesserung der Frequenzauflösung im hohen Frequenzbereich, wie die Literaturrecherche in [MULL 02] aufzeigt.

Vom Spektrum zum Merkmal

Um aus dem gewonnenen Spektrum nun aber taugliche Merkmale zu generieren, müssen noch einige weitere Verarbeitungsstufen durchlaufen werden. Diese sollen die Robustheit der Ergebnisse gegenüber Störungen erhöhen. Zuerst wird dazu das Spektrum auf eine

[5] Ein stochastisches, also vom Zufall abhängiges, Signal wird als *stationär* bezeichnet, wenn seine statistischen Kennwerte keine Funktionen der Zeit sind [HOFF 03].

[6] Die Bezeichnung *"Wavelet"* lässt sich im Deutschen wörtlich mit "Wellchen" oder "kleine Welle" wiedergeben, in Anlehnung an die visuelle Darstellung eines Kurvenabschnitts.

Mel-Skala[7] aufgetragen. Hierdurch kommen Frequenzen im Bereich über 1kHz in Relation zu den Tiefen besser zur Geltung. Anschließend wird das resultierende Mel-Spektrum logarithmiert. Dies verwandelt eventuell vorhandene, durch Faltung[8] eingeflochtene Verunreinigungen des Signals in eine rein additive Beziehung zwischen dem Nutz-Signal und der Störung. Zum Abschluss erfolgt noch eine Cosinus-Transformation[9], welche bewirkt, dass die resultierenden Koeffizienten völlig unabhängig voneinander, also unkorreliert, sind. Dieses Ergebnis nennt sich *Cepstrum*[10], die beteiligten Koeffizienten *"Mel-Frequency Cepstral Coefficients"* (MFCC).

Es existiert noch eine weitere, völlig andere Möglichkeit, cepstrale Koeffizienten zu berechnen: Die der *Linearen Vorhersage* (LP), wobei ein Signalmesswert als Linearkombination seiner Vorgänger dargestellt wird. Dieses Verfahren führt neben verschienen anderen möglichen Sprachmerkmalen[11] auch zu den *"Linear Prediction Cepstral Coefficients"* (LPCC). Untersuchungen sehen MFCC's und LPCC's generell als ähnlich performant bezüglich der Sprach- oder Sprechererkennungs-Leistung an. Meistens wird jedoch ersteren aufgrund von höherer Robustheit gegenüber Störeinflüssen der Vorzug gegeben [DAVI 80].

Biologische Rechtfertigung

Die cepstrale Analysemethode birgt nicht nur die genannten technischen Vorteile – sie korrespondiert auch mit dem heutigen Verständnis der Sprachproduktion des Menschen:

Wie Abbildung 2.3 darstellt, sitzen am oberen Ende der Luftröhre die Stimmbänder. Durch sie strömt beim Sprechen Luft in den blau eingefärbten Vokal-Trakt, der durch seine Form die Grundfrequenz des zu artikulierenden Tons, wie er von den Stimmbändern gebildet wurde, verändert. Die Resonanzfrequenzen des Vokal-Trakts sind dabei deutlich als Spitzen im Spektrum der Sprache eines Menschen zu sehen. Sie werden *Formanten* oder *Formant-Frequenzen* genannt und tragen neben der Information über den ausge-

[7] Die *Mel-Skala* ist dem menschlichen Hörgefühl nachempfunden: Ein Ton, der wahrnehmungsgemäß doppelt so hoch klingt wie ein anderer, hat so auch den doppelten Mel-Wert. Das resultiert in einer Skala, die im Bereich unter 1000 Hertz beinahe linear verläuft, darüber aber näherungsweise logarithmisch skaliert ist. Die genaue Umrechnungsformel von [Hz] in [Mel] lautet $m = 1127,01048 * log(1 + f/700)$, woraus folgt, das $1000Hz = 1000Mel$ [WIKI 04b].

[8] Die Operation der *Faltung* erzeugt aus zwei Funktionen (oder, in diesem konkreten Fall, zwei Signalen) eine Dritte, welche den Anteil an Übereinstimmung zwischen der ersten und der umgekehrt verschobenen zweiten Funktion darstellt: $(f * g)(t) = \int f(\tau)g(t-\tau)d\tau$ [WIKI 04b]. Anschaulich kann man sich also vorstellen, dass ein resultierender Messwert aus einer Linearkombination der gewichteten Messwerte aus beiden Ausgangssignalen besteht. In diese Linearkombination können dabei Werte "aus der Umgebung" der Position des Resultats einfließen, also solche, die davor oder danach liegen. Die Größe dieser Umgebung kann dabei beliebig sein, solange sie endlich ist.

[9] Die *Diskrete Cosinus-Transformation* (DCT) ist wiederum eine Reihenentwicklung, ähnlich der FFT, nur dass hier ausschließlich Cosinusterme Verwendung finden.

[10] *"Cepstrum"* ist das Resultat eines Wortspiels mit dem Begriff Spektrum: Es spielt auf dessen Eigenschaft an, Faltung in Addition zu verwandeln, und invertiert deshalb den ersten Teil des Wortes in Anlehnung an die Definition der Faltung [CHUN 92].

[11] Zu erwähnen sind die *Linear Spectral Pairs* (LSP). Weitere Merkmale finden sich in [CAMP 97].

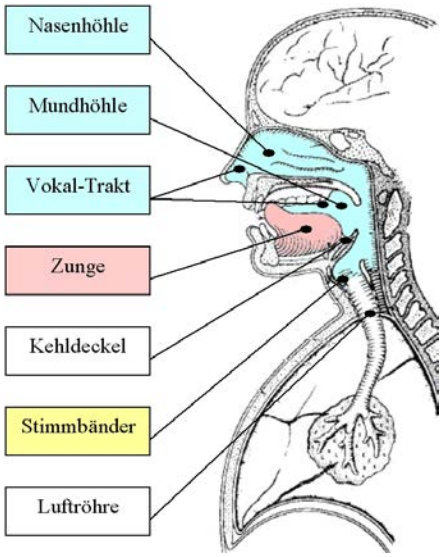

Abbildung 2.3: Die menschlichen Organe der Stimmbildung (nach [CAMP 97]).

sprochenen Laut auch Informationen über die Identität des Sprechers, da sie unmittelbar von der anatomischen, nicht veränderbaren Form und Länge des Vokal-Trakts abhängen. Andersherum heißt das, dass sich aus einer irgendwie gearteten Repräsentation des Sprachsignals durch Merkmale anhand der Formanten die anatomischen Eigenschaften des Vokal-Trakts und damit die Identität[12] der sprechenden Person ableiten lassen. An dieser Stelle folgt die Begründung für die weite Verbreitung der cepstralen Merkmale im Bereich der Sprachverarbeitung: Hier werden die Einflüsse der Stimmbänder[13] von denen des Vokal-Trakts getrennt[14], so dass ein exaktes mathematisches Modell der realen Vorgänge entsteht. In diesem kann nun der Einfluss des Vokal-Trakts individuell und lösgelöst von den Stimmbändern betrachtet werden [AFFE 96].

Das Modell ist allerdings nur solange zutreffend, wie sich die Sprachproduktion auch wirklich innerhalb seiner Parameter (der *Quelle-Filter-Annahme*, wie Abbildung 2.4 sie darstellt) bewegt. Dies trifft genau genommen nur auf die stimmhafte Sprache zu, den Teil, den wir als Vokale oder wohlgeformte Konsonanten wahrnehmen. Sein Spektrum zeigt starke Periodizität, hervorgerufen durch die Vibration der Stimmbänder und Formant-Frequenzen. Im Gegensatz dazu hat nicht-stimmhafte Sprache eine Charakteristik wie breitbandiges Rauschen, da sie durch stark verengte Stimmbänder sozusagen "hervorgedrückt" wird. Dies ist beispielsweise bei dem Laut "s" in "Hass" der Fall. Abschnitt 2.4.2 wird sich unter anderem der Fragestellung widmen, wie sich die hier getroffene Modellannahme durch Ausschluss der unpassenden Sprachanteile aufrechterhalten lässt.

[12] Die Form des Vokal-Trakts ist ähnlich einzigartig und bezeichnend für einen Menschen wie die Beschaffenheit seiner Iris. Daher lässt sich von einer plausiblen Schätzung der Konfiguration dieses biometrischen Merkmals direkt auf die Identität des Sprechers schließen.

[13] In diesem Modell als *Quelle* bezeichnet, im Vergleich zum Vokal-Trakt, der als *Filter* wirkt.

[14] Dieser schon oben beschriebene Vorgang der Dekonvulation einer Faltung ist auch mit dem Begriff *"homomorphe Transformation"* belegt.

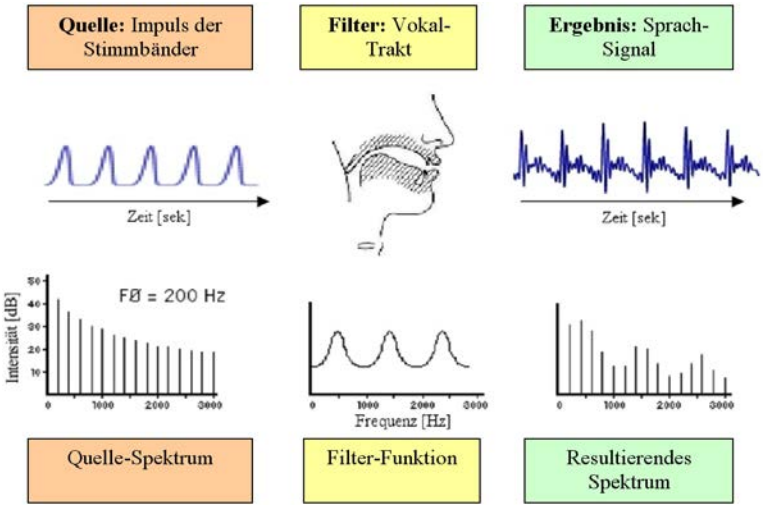

Abbildung 2.4: Das Quelle-Filter Modell (nach [DUNN 03]).

Arbeiten mit cepstralen Merkmalsvektoren

Viele Forschungsvorhaben setzten sich bereits mit den MFC-Koeffizienten auseinander, um mehr über ihre Möglichkeiten und Grenzen zu erfahren [COMB 96]. Ihre Effektivität ist zwar seit Beginn der 1980er Jahre unbestritten, doch schon so einfach klingende Fragen wie diejenige nach der optimalen Anzahl Koeffizienten für eine bestimmte Aufgabe mussten bislang unbeantwortet bleiben. So weist [CAMP 97] nur auf den "Fluch der Dimensionalität"[15] hin, ohne aber konkrete Angaben zu machen. Und [SUN 93] thematisiert diese Frage zwar auch, kommt aber durch die Verwendung zeitlicher Ableitungen[16] und weiterer Verfahren zu Ergebnissen, welche außerhalb dieses Kontextes an Gültigkeit verlieren. Nichtsdestotrotz gibt es Arbeiten über die Auswahl der zu verwendenden Koeffizienten: [PETR 03] spricht sich für die selektive Verwendung nur bestimmter Koeffizienten aus[17], und [LIN 99] sieht gar Vorteile in der Verwendung nur bestimmter Subbänder des Ausgangssignals[18].

Im Rahmen des Projekts Medienumbrüche, welches sich ja stark mit den Methoden der Anreicherung von Medien mit Meta-Informationen mittels MPEG-7 befasst, ist es sinnvoll, noch folgende Neuerung der letzten Zeit kurz zu beleuchten: Die MPEG-7-Audio-

[15] Der *Curse of Dimensionality* besagt, dass die notwendige Anzahl zugrundeliegender Daten exponentiell mit der Dimensionalität der Merkmalsvektoren wachsen muss, um den später anzuwendenden statistischen Verfahren nicht die Grundlage und Rechtfertigung zu rauben.

[16] Eine recht weit verbreitete Methode, um weitere Informationen nutzbar zu machen, ist die Verwendung der Ableitung der Merkmalsvektoren nach der Zeit. Die so erhaltenen Merkmale werden mit einem vorangestellten "delta" bezeichnet, also beispielsweise $\Delta MFCC$ [YOUN 97].

[17] Die Veröffentlichung ermittelt die Koeffizienten c_0, c_6 und c_7 als negativ für die Erkennungsleistung von Sprechern, und c_1, c_2 und c_3 sollen gegenüber dem Rest zumindest eine untergeordnete Rolle spielen.

[18] Hier wird das Band von 3-8kHz als sprecherspezifischer gegenüber dem Bereich von 0-5kHz angesehen.

Merkmale. Sie sind im Großen und Ganzen vergleichbar mit den MFCC's, was ihre Leistung angeht, unterscheiden sich aber in ihrer Gewinnung: Zur Dimensionalitätsreduktion am Schluss wird nicht die DCT, sondern die PCA und ICA[19] angewandt, und die Frequenzachse ist nicht Mel-, sondern logarithmisch skaliert. Die MPGE-7-Audiomerkmale sind deshalb nicht allzu sehr auf Sprache als Ziel der Audio-Analyse festgelegt wie MFCC's und bieten sich dementsprechend auch für andere Fragestellungen wie beispielsweise Musikanalyse an.

Intuitive Merkmale

Daneben existieren auch noch einige andere, eher intuitive Merkmale eines Audiosignals. Sie dienen dessen Einordnung in verschiedene Klassen (Sprache, Musik, Knall, ...) oder aber der Unterscheidung von stimmhafter und nicht-stimmhafter Sprache. Hierzu sind die *Nulldurchlaufsrate* sowie die *Energie* eines Signalabschnitts zu zählen. Unter der Energie versteht man dabei die aufsummierten und quadrierten Amplituden, meist wiedergegeben auf einer Dezibel-Skala [SMIT 04]. Die Nulldurchlaufsrate gibt einfach die Anzahl der Vorzeichenwechsel im analysierten Abschnitt auf Messwert-Ebene wieder. Ein weiteres Merkmal gerade zur Unterscheidung männlicher, weiblicher und kindlicher Stimmen bietet die durchschnittliche Grundfrequenz (Tonhöhe) der Stimme [EZAI 01].

2.4.2 Sprach-Filterung

Zu Beginn der nun im System folgenden Analysephase ist es wichtig, eine gute Grundlage für die darauf aufbauenden Algorithmen zu schaffen: Eine ordentliche *Segmentierung* der Rahmen[20] ihren Inhalt betreffend, eine Aufteilung also namentlich in Sprache und Sonstiges.

Audio-Segmentierung

Eine wegweisende Arbeit auf dem Gebiet der Audio-Segmentierung ist [LU 02b]. Hier wird mittels einiger neuer, intuitiver Merkmale ein Audiostrom in Sprache, Musik, Nebengeräusche und Stille zerlegt. Die Ergebnisse der Segmentierung sind mit 96 Prozent Genauigkeit sehr gut, außerdem ist das Schema sehr ressourcenschonend, so dass es Echtzeitkriterien deutlich erfüllt. [EL-M 00] nutzt zur Unterscheidung zwischen Sprache und Musik das Merkmal "Line Spectral Frequencies" (LSF) und erreicht damit eine zeitliche

[19] *Principal* bzw. *Independent Component Analysis* sind mathematische Methoden zur Reduktion der Dimensionalität von Eingabevektoren: Sie werden derart in einen anderen Vektorraum projiziert, dass sich die Merkmale mit der höchsten Diskriminierungsfähigkeit in den unteren Koeffizienten niederschlagen. Mit der Unterschlagung hoher Koeffizienten verringert man nun zwar die Gesamtleistung, behält die diskriminativsten Merkmale jedoch bei.

[20] Zur Erinnerung: Die Merkmalsextraktion ist abgeschlossen, und von nun an arbeiten alle Module nicht mehr auf den Rohdaten des Signals, sondern auf den Merkmalen, die jeweils aus einem kurzen Rahmen des Signals extrahiert wurden.

Auflösung hinab bis zu einem Rahmen von 20ms bei allerdings nur maximal 82,5 Prozent Genauigkeit. Soll allerdings ein ähnlich gutes Ergebnis wie oben erzielt werden, schmilzt dieser Vorteil zusammen, und Rahmenverzögerungen von einer Sekunde werden nötig. Dies ist wiederum vergleichbar mit obigem Ansatz von Lu et al.

Einen etwas anderen Ansatz und Weg verfolgt [CASE 00]: Dort geht es um die nachträgliche Aufteilung eines zusammengesetzten Audiostroms in seine ursprünglichen Quellen. Dies soll mittels Unterraum-Analyse des Spektrogramms[21] geschehen. Dazu wurde eine neue Methode, *Independent Subspace Analysis* genannt, entwickelt, welche die schon bekannte *Independent Component Analysis* um einige hilfreiche Merkmale erweitert.

Wie wichtig dieser Schritt der Audio-Segmentierung für die darauf aufsetzenden Module ist, zeigt auch noch einmal [RAMA 03] auf: Anhand des weltgrößten Archivs digitalisierter Video- und Sprachdokumente[22] wird der Zusammenhang zwischen späterer (Sprach-)Erkennungsleistung und der Performanz der Segmentierung dargestellt.

Verfeinerung durch Endpunkt-Detektion

Ob nun als Verfeinerungsschritt nach erfolgter Segmentierung oder als eigentlicher Segmentierungsschritt auf Daten ohne viele Nebengeräusche: Die Detektion von Endpunkten der Sprache, also der exakten Anfangs- und Endpunkte einer Äußerung hat lange Tradition im Sprachverarbeitungs-Bereich. Das mit Abstand am häufigsten hierzu herangezogene Sprachmerkmal ist die Kurzzeit-Energie: Dies basiert auf der Tatsache, dass stimmhafte Sprache (im Vergleich zu nicht-stimmhafter Sprache oder Rauschen beispielsweise) sehr energiereich ist. Eine Verfolgung der Energiekontur kann deshalb, so denn einige zusätzliche Absicherungen getroffen werden, höchst erfolgversprechend sein. So muss in [LI 02] zwischen drei Zuständen unterschieden werden, in Abhängigkeit von welchen sich die Interpretation der Messwerte ändert. Auch [MARZ 02] kommt mit dem Merkmal Energie sehr gut zurecht, teilt das Spektrum zuvor jedoch in eine hoch- und eine niederfrequentes Band auf. Außerdem kommt ein adaptives Schwellwertverfahren zum Einsatz.

[HAIG 93] wählt einen anderen Ansatz, indem hier cepstrale Merkmale in das Zentrum der Analyse gerückt werden. Begründung hierfür ist die geringe Robustheit des Energie-Kriteriums gegenüber Störeinflüssen im Signal. Diese zeigt sich beispielsweise in den Ergebnissen in [GREE 99], wo durchschnittlich nur circa 60 Prozent Genauigkeit bei der Endpunkt-Detektion erreicht wurden, allerdings einschließlich einer Unterscheidung zwischen stimmhafter und nicht-stimmhafter Sprache. Dass moderne Verfahren hier allerdings Besseres leisten und so eine mit den cepstralen Merkmalen vergleichbare Performanz

[21] Ein *Spektrogramm* ist eine Visualisierungsmethode für den zeitlichen Verlauf eines Spektrums: Auf der x-Achse ist die Zeit aufgetragen. Jedem Zeitpunkt ist nun ein eingefärbter Balken zugeordnet, der für jede mögliche Frequenz einen Bildpunkt enthält. Die Farbgebung jedes Punktes gibt nun Auskunft über den Grad des Vorhandenseins der entsprechenden Frequenz zum entsprechenden Zeitpunkt im Signal.

[22] Zusammengetragen im sogenannten *VHF-Korpus*, einem Projekt der "The Survivors of the Shoah Visual History Foundation", welches Interviews mit Holokaust-Überlebenden archiviert.

erzielen können, zeigt nicht zuletzt [LI 03] in einem Ansatz, der hier kurz in seiner Algorithmik skizziert werden soll:

Adaptive Stille-Erkennung

Der Algorithmus arbeitet Kameraeinstellungs-basiert[23], da sich empirisch bestätigte, dass die Geräuschcharakteristiken, wie auch immer sie beschaffen sein mögen, innerhalb dieser Zeit quasi-konstant bleiben. Nun werden die einzelnen Audio-Rahmen nach ihrer Energie sortiert und in 10 Gruppen quantisiert. Somit stellt die durchschnittliche Energie der ersten Gruppe das untere Ende der Stille/Geräusch-Energie dar, während die letzte Gruppe das obere Ende der Sprachenergie darstellt[24]. Es gilt nun, einen Schwellwert zwischen beiden Extremen zu finden, welcher die Sprache sauber von ihrem Umfeld trennt. Dies geschieht anhand der Summe der ersten drei Gruppen-Mittelwerte und der Summe der letzten drei Gruppen-Mittelwerte. Auf diese Weise kann sich der Schwellwert jeder Geräuschsituation anpassen.

Ausschluss nicht-stimmhafter Sprachanteile

Ein Wort noch zu der Unterdrückung nicht-stimmhafter Sprache: Wie bereits erwähnt, sollte letztere aus Gründen der Korrektheit der Modellierung aus dem zu analysierenden Material entfernt werden. Weiter ist sie durch ihre Energiearmut anfälliger gegen Nebengeräusche. Wie Untersuchungen zeigten, muss das jedoch nicht zwingend zu schlechteren Ergebnissen bei der abschließenden Sprechererkennung führen. Nichtsdestotrotz verringert sich die Komplexität der nachfolgenden Verfahren durch ein Ausschneiden der nicht-stimmhaften Anteile alleine durch die Dezimierung des zu untersuchenden Datenmaterials [AJME 02].

Eine Methode, nicht-stimmhafte Äußerungen zu identifizieren, liegt in der Verfolgung der Autokorrelations-Sequenz des Signals: Falls eine eindeutige Grundfrequenz der Schwingung auffällt, handelt es sich mit großer Sicherheit um stimmhafte Sprache, ansonsten nicht. Dies ist begründet in der Erzeugung stimmhafter Sprache, in welcher die Formant-Frequenzen des Vokal-Trakts starken Einfluss haben. Die Periodizität stimmhafter Sprache lässt sich auch anschaulich in Abbildung 2.5 erfassen.

Die zweite Methode nutzt, neben der Kurzzeit-Energie, die Nulldurchlaufsrate als Merkmal: Nicht-stimmhafte Sprache hat hier einen deutlich höheren Wert vorzuweisen

[23] Eine *Kameraeinstellung* ist die kleinste semantische Einheit innerhalb eines Videos. Sie besteht aus der Zeit zwischen zwei *Schnitten*. Ihr übergeordnet ist die *Szene*, welche semantisch und zeitlich zusammenhängende Kameraeinstellungen zusammenfasst, die überdies einem höheren Konzept oder einer Geschichte folgen [LI 03].

[24] Diese Schlussfolgerung ist aufgrund des oben erwähnten Zusammenhangs zwischen Energiewert und Audiotyp möglich.

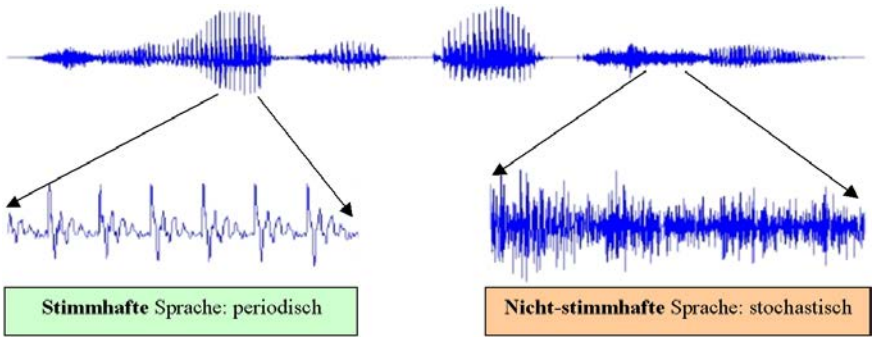

Abbildung 2.5: Struktur der Sprache am Beispiel des englischen Satzes *"she had your dark suit"* (nach [DUNN 03]). Deutlich sind die verschiedenen Worte und, in ihnen, die verschiedenen Energielevel und Periodizitäten der Laute zu erkennen.

als ihr periodisches Pendant, weshalb sich beide Merkmale in Kombination zur Diskrimination eignen [KWON 02].

2.4.3 Sprecherwechsel-Erkennung

Nach der Audio-Segmentierung folgt nun der Schritt der *Sprecher-Segmentierung*, oft auch *Sprecherwechsel-Erkennung* genannt. Ziel ist hier die Zusammenfassung aufeinander folgender Sprachsegmente, die mit Sicherheit von identischen Sprechern geäußert wurden. So sollen möglichst lange, sprecherhomogene Abschnitte als Ausgangslage für die nachfolgenden Schritte geschaffen werden.

In [LIU 99] wird diese Stufe als Vorverarbeitung für einen Spracherkenner eingesetzt, was an den verwendeten Techniken deutlich wird: Ein Phonem-Dekoder unterteilt die ankommende Sprache zuerst einmal in Phoneme und ordnet sie jeweils einer von vier verschiedenen Klassen zu. Anschließend kommt ein dem BIC[25] ähnliches gewichtetes Likelihood-Verfahren[26] zum Einsatz, um die Ähnlichkeit aufeinander folgender Phoneme auszudrücken. Ist diese geringer als ein empirisch gesetzter Schwellwert, wird ein Sprecherwechsel angenommen. Das Verfahren arbeitet dabei immer auf einer Phonemsequenz von zwei Sekunden Länge und bietet eine zeitliche Auflösung in der Region der Länge eines Phonems[27]. Interessant ist hier die Feststellung der Forscher, dass circa 80 Prozent der Sprecherwechsel innerhalb ihrer Nachrichtensendungs-Testdaten außerhalb von Sprachregionen lagen. Die Sprecher fielen sich also nicht gegenseitig ins Wort, so dass Aussagen unterschiedlicher Sprecher auch meistens durch Pausen getrennt waren.

[25] Das *Bayes'sche Informations-Kriterium* wird in Abschnitt 2.4.6 noch im Detail behandelt.

[26] Das englische Wort *likelihood* bezeichnet die Wahrscheinlichkeit, dass ein bereits geschehenes Ereignis mit bekanntem Ergebnis genau zu diesem Ausgang kam. Im Gegensatz dazu meint *probability* die Wahrscheinlichkeit, dass ein noch in der Zukunft liegendes Ereignis ein bestimmtes Ergebnis verursachen wird [WEIS 99]. Da das Deutsche diese Unterscheidung zwischen rückwärts- und vorwärtsgerichteter Wahrscheinlichkeit nicht kennt, werden im Folgenden die beiden englischen Begriffe beibehalten.

[27] Ein Phonem ist im Durchschnitt etwa 100 Millisekunden lang [LIU 99].

[KWON 02] kommt an dieser Stelle zu einem ähnlichen Resultat: Auch hier zeigte sich anhand der Testdaten (Nachrichtensendungen und Spielfilme), dass Sprecherwechsel meistens durch Pausen eingeleitet wurden. Dies wurde hier zum Anlass genommen, innerhalb von Sprachsegmenten gar nicht mehr nach Wechseln Ausschau zu halten. Vielmehr wurde eine neue Metrik entwickelt, um aufeinander folgende Segmente ab einer bestimmten Länge[28] zu untersuchen. Das neue Distanzmaß basiert dabei auf einer gewichteten Euklid'schen Distanz, wobei sich die Gewichte aus der *Linearen Diskriminanz Analyse* (LDA) nach Fischer ableiten: Somit werden Inter- und Intraklassenvarianz zueinander ins Verhältnis gesetzt und die daraus ableitbare Information genutzt. Das Ergebnis ist für Nachrichtensendungen mit 6,8 Prozent Fehlerquote recht anschaulich, nur bei Filmen bricht es ein. Das mag an der Segmentlänge liegen, die im Durchschnitt bei nur 2,5 Sekunden und damit oftmals unter der Mindestlänge liegt.

Dunn et al. testen in [DUNN 00] gleich zwei Systeme. Das erste, *Interner Segmentierer* genannt, braucht dabei jedoch zwingend a-priori-Wissen über Anzahl und Identität der zu erwartenden Sprecher, um mittels explizit vorher bekannter Sprechermodelle und einem UBM[29] eine Ähnlichkeitswertung auf Rahmen-Basis zu ermitteln. Diese zeitliche Auflösung und minimale Segmentlänge ist enorm und zeigt, was bei einer weniger allgemeinen Aufgabenstellung möglich ist. Der zweite Ansatz zur *Externen Segmentierung* teilt den Datenstrom der Merkmalsvektoren zunächst stur in 100 Rahmen lange[30] Segmente ein. Zwischen aufeinander folgenden Segmenten wird nun durch GMM's [31] ein Likelihood-Wert ermittelt, anhand dessen ein ungeleitetes, agglomeratives Clustering-Verfahren[32] zum Einsatz kommt. Die Anzahl der Cluster (Sprecher) ist dabei auch schon vorher bekannt.

Mehrstufige Sprecher-Segmentierung

Der folgende Ansatz soll wieder etwas ausführlicher dargestellt werden. Er hat eine längere Entwicklung hinter sich, wurde er doch in [LU 02a] das erste Mal publiziert und erfuhr kurz darauf Erweiterung in [LU 02c] durch die Verwendung weiterer Merkmale und deren Verschmelzung. Somit ist dieses das einzige der hier vorgestellten Verfahren, welches nicht nur anhand der MFFC's arbeitet. Wenig später wurde der Algorithmus nochmals angepasst und in [WU 03] erneut publiziert. Hier wurde jedoch nur der Schwerpunkt der Schilderung etwas verschoben und eine neue Art der Audio-Segmentierung mittels UBM's dargestellt. Sie dient dem Vorverarbeitungsschritt Sprecherwechsel-Erkennung als Liefe-

[28] Kwon et al. geben zwei Sekunden als die minimal nötige Zeit für die aussagekräftige Erkennung von Sprecherwechseln an.

[29] Ein *Universal Background Model* ist eine Spezialform des im nächsten Abschnitt ausführlich erleuterten *Gauß'schen Misch-Modells*: Es ist sehr groß und repräsentiert nicht einen einzelnen Sprecher oder einen Laut, sondern die gesamte Alternativ-Hypothese für die Situation des statistischen Tests. In diesem Fall ist das UBM also auf alle anderen möglichen Sprecher trainiert.

[30] In diesem Fall entsprechen 100 Rahmen einer Dauer von einer Sekunde.

[31] *GMM* ist die Abkürzung für die schon früher aufgetauchten *Gauß'schen Misch-Modelle*.

[32] Siehe hierzu auch die Erklärung in Abschnitt 2.4.6.

rant von Aussagen über die Sicherheit, mit der ein bestimmtes Segment als Grundlage einer Entscheidung herangezogen werden kann.

Der eigentliche Algorithmus kann deshalb vollständig aus [LU 02c] entnommen werden. Er arbeitet zweistufig, wobei die erste Stufe (zu viele) Anhaltspunkte für mögliche Wechsel in der Sprecheridentität liefert. Dazu durchläuft sie die Merkmalssequenz mit einem 2,5 Sekunden langen, um jeweils 0,5 Sekunden vorrückenden Fenster, berechnet eine Distanz und entscheidet anhand der Erfüllung dreier Kriterien über einen möglichen Wechsel:

- $D(i, i+1) > D(i+1, i+2)$
- $D(i, i+1) > D(i-1, i)$
- $D(i, i+1) > T_i$

$D(i, j)$ ist hierbei die *Divergence Shape*-Distanz zwischen den Segmenten (Fenstern) i und j, wie sie in [CAMP 97] hergeleitet wurde. T_i ist ein Schwellwert, der sich nach folgender Formel adaptiv aus den vergangenen Werten berechnet:

$$T_i = \alpha \cdot \frac{1}{N} \sum_{n=0}^{N} D(i-n-1, i-n) \tag{2.1}$$

Hierbei wiederum ist N die Anzahl der zur Berechnung herangezogenen, zurückliegenden Werte[33] und α ist ein datenspezifischer Gewichtungsfaktor, der von Lu et al. auf den Wert 1,2 gesetzt wurde.

Die darauf aufsetzende zweite Stufe der Verfeinerung baut nun aus den Segmenten, welche die Stufe eins ohne potenziellen Wechsel passieren, schrittweise ein Sprechermodell auf: Für jedes Segment werden jeweils Mittelwert-Vektor $\vec{\mu}$ und Kovarianzmatrix Σ bestimmt.

$$\vec{\mu} = \frac{1}{T} \sum_{t=1}^{T} \vec{x_t} \tag{2.2}$$

$$\sigma_{k,l}^2 = \frac{1}{T} \sum_{t=1}^{T} (x_{t_k} - \mu_k) \cdot (x_{t_l} - \mu_l), \quad \text{[k,l]=1,...,D} \tag{2.3}$$

$$\Sigma = \begin{pmatrix} \sigma_{1,1}^2 & \sigma_{1,2}^2 & \cdots & \sigma_{1,D}^2 \\ \sigma_{2,1}^2 & \sigma_{2,2}^2 & \cdots & \sigma_{2,D}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{D,1}^2 & \sigma_{D,2}^2 & \cdots & \sigma_{D,D}^2 \end{pmatrix} \tag{2.4}$$

x_t ist dabei ein D-dimensionaler Merkmalsvektor aus dem betrachteten Segment der Länge T, und k und l sind spezifische Komponenten eines Vektor.

[33] Ähnlich wie bei der Berechnung eines gleitenden Mittelwerts muss hier eine Sonderbehandlung für weniger zurückliegende Werte eingeführt werden: In diesem Fall wächst N von eins bis zu seinem endgültigen Wert kontinuierlich an.

Ist das vorangehende Segment desselben Sprechers hinreichend genau durch seine beiden statistischen Maßzahlen beschrieben, bekommt das aktuelle Modell eine neue Komponente, ansonsten wird die vorangegangene mit den neuen Werten aufgewertet. Dieses Verfahren ähnelt der Trainingsmethode für GMM's, weshalb es *Quasi-GMM* genannt wird[34]. Wenn nun ein potenzieller Sprecherwechsel von der ersten Stufe gemeldet wird, wird ein Likelihood-Vergleich des aktuellen Segments mit dem Modell des vorangegangenen Sprechers angestrebt.

Mittels des BIC wird so ein schwellwertfreies Verfahren umgesetzt, denn dieses Kriterium liefert eine binäre Entscheidung abhängig vom Vorzeichen seines Ergebnisses: Ein Wert größer null legt nahe, dass die beiden getesteten Modelle (das des letzten Sprechers und das des aktuellen Segments) zu unterschiedlichen Sprechern gehören. Ein negativer Wert sagt dementsprechend aus, dass die Sprecher identisch sind[35].

Die zweistufige Vorgehensweise ist notwendig, da zum Zeitpunkt der potenziellen Wechsel nicht genügend Information für den BIC-Test zur Verfügung steht. Haben sich allerdings einige Segmente ohne Sprecherwechsel angesammelt, liefert dieser aussagekräftigere Ergebnisse als das rein schwellwert- und distanzmaßbasierte Verfahren. Dieser Zusammenhang zeigt auch deutlich, in welchen Situationen der Algorithmus ordentlich arbeiten kann: Eine längerer Folge nicht zu kurzer Segmente mit wenigen Wechseln erkennt er spielend. Wird der Verfeinerungsstufe allerdings im Segmenttakt ein potenzieller Wechsel zur Überprüfung vorgelegt, sinkt die Leistung erheblich, denn das BIC benötigt viele Daten für sichere Entscheidungen. So berichten Lu et al. auch, das die Performanz unterhalb von drei Sekunden Segmentlänge nicht mehr ausreichend ist. Vielleicht noch eine Bemerkung zur minimalen Segmentlänge: In einer Untersuchung über den Einfluss der Länge der Testphrasen auf das Ergebnis einer Sprecherverifikation wird das unterste Ende der notwendigen Phrasenlänge mit 1,5 Sekunden angesetzt [HE 99a].

2.4.4 Sprecher-Repräsentation

Nun sind sämtliche zuverlässige Sprach-Rahmen als solche markiert. Überdies wurden längere Abschnitte in Segmente unterteilt, welche mit Sicherheit von nur einem Sprecher stammen. Dies sind gute Voraussetzungen für die nächste Aufgabe, das Erstellen einer internen Repräsentation der Stimme eines Sprechers im System. Diese Repräsentationen werden im Fachsprachgebrauch allgemein *Modelle* genannt. Nichtsdestotrotz verbergen sich hinter dieser einheitlichen Nomenklatur verschiedene Konzepte:

[34] Tieferes Verständnis hierfür wird sich sicherlich bei der Lektüre des nächsten Abschnitts einstellen.

[35] Im Gegensatz zu der später eingeführten Definition des BIC arbeitet die Implementierung von Lu et al. mit den Determinanten der einzelnen Kovarianzmatrixen anstatt mit Log-Likelihood-Werten als Maß für die Anpassung des Modells an seine Daten. So ist es zu erklären, dass hier ein Wert größer null unterschiedliche Sprecher suggeriert, während sich dies später genau umgekehrt verhält.

Vektor-Quantisierung

Das älteste dieser Konzepte ist sicherlich das der *Vektor-Quantisierung* (VQ). Hierbei werden die vielen Merkmalsvektoren des abzubildenden Sprachabschnitts durch einige wenige synthetische Vektoren beschrieben. Diese können aus einem Clustering-Prozess als die Zentroide der gefundenen Cluster entstehen [KINN 00], oder aber mittels eines Gradientenabstiegsverfahrens gewonnen werden [VOLZ 01]. Die so ermittelten Referenzvektoren werden in einem Feld zusammengefasst, welches den Namen *Codebuch* trägt. Die Ähnlichkeit einer Menge von Merkmalsvektoren zu einem Codebuch wird dabei folgendermaßen bestimmt: Jeder Vektor der Menge wird auf denjenigen Referenzvektor abgebildet, der ihm laut einer Metrik am nächsten ist. Der verbleibende Quantisierungsfehler, welcher zwischen der Ursprungsmenge und der Menge der Abbildungen existiert, ist ein Maß für die Passform des Modells zu den Daten.

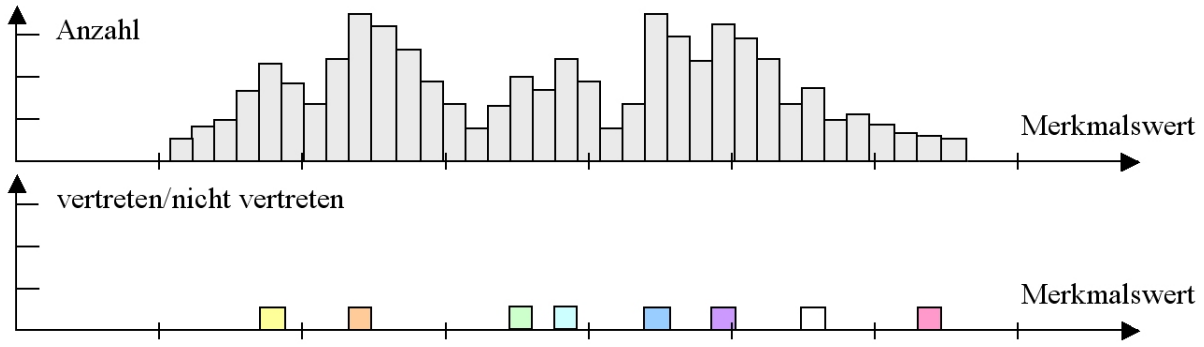

Abbildung 2.6: Hypothetisches Histogramm einer Menge eindimensionaler Merkmalsvektoren (oben) und ihre Repräsentation durch ein Codebuch der Ordnung acht. Nach [REYN 95].

Abbildung 2.6 verdeutlicht den Prozess der Codebuchbildung: Im oberen Bildabschnitt ist das Histogramm einer Menge eindimensionaler Merkmalsvektoren aufgetragen. Der Anschauung halber könnte man beispielsweise annehmen, es handle sich bei den Merkmalsvektoren um einfache Frequenzen. Jeder Histogrammbalken repräsentierte dann die Auftrittshäufigkeit einer bestimmten Frequenz, angefangen ganz links bei 0Hz. Der untere Teil der Grafik zeigt nun die Modellierung dieser Datenmenge durch ein Codebuch mit zehn Einträgen: Die am häufigsten anzutreffenden Frequenzen scheinen repräsentativ für das Signal zu sein und werden deshalb in das Codebuch aufgenommen.

Statistische Modelle

Das VQ-Modell macht sich zu Nutze, dass die über längere Zeit gemittelten Merkmalsvektoren eher zu sprecher- als zu sprachspezifischen Besonderheiten tendieren. Dies ist bei statistischen Ansätzen grundsätzlich auch der Fall. Verschieden ist allerdings die

Repräsentation dieser Erkenntnis: Während im VQ-Ansatz diskrete Zentroide die Repräsentanten darstellen, werden sie im statistischen Umfeld durch Erwartungswert $\vec{\mu}$ und Kovarianzmatrix Σ beschrieben. Dies erlaubt generell eine genauere Modellierung, da der Umgang mit Ungewissheit systemimmanent umgesetzt ist[36].

Die statistische Art der Modellierung kennt für diese Aufgabenstellung die Darstellung als *Wahrscheinlichkeitsdichte*[37], falls die Merkmalsvektoren als untereinander unabhängig angesehen werden können[38]. Ein Blick auf Abbildung 2.6 zeigt, dass die Form des Histogramms im oberen Teil der Grafik jedoch mit keiner bekannten Kurve einer *Wahrscheinlichkeitsdichtefunktion* korrespondiert. Die Lösung ist die Nutzung der *Gauß'schen Mischverteilung*: Mehrere *Glockenkurven* zusammen können eine willkürliche Verteilungsfunktion annähern. Die entstehende Kurve und ihre Zusammensetzung aus mehreren einzelnen Glockenkurven ist in Abbildung 2.7 visualisiert[39].

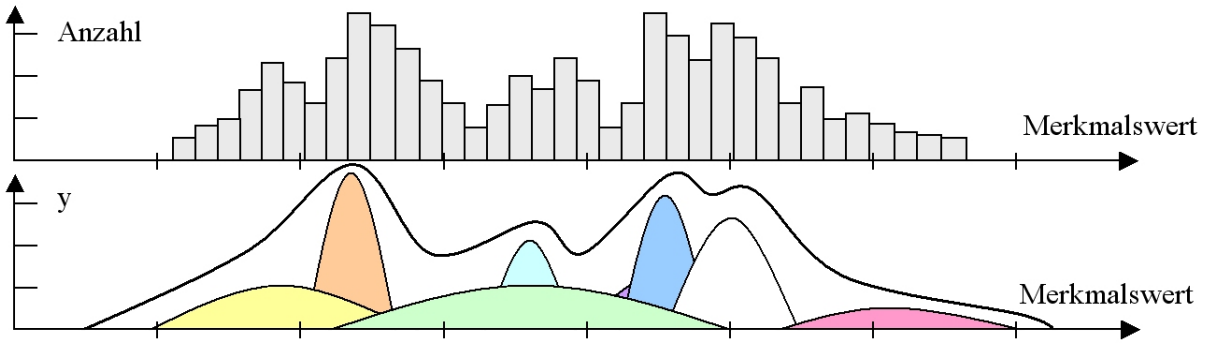

Abbildung 2.7: Hypothetisches Histogramm einer Menge eindimensionaler Merkmalsvektoren (oben) und Repräsentation durch ein Gauß'sches Misch-Modell (unten). Zu sehen ist die unskalierte Kontur des GMM's und die acht Gaußdichtefunktionen, welche diese Kontur aufspannen. Nach [REYN 95].

[36]Dies scheint auf den ersten Blick ambivalent: Ungewissheit führt zu höherer Genauigkeit? Ja, denn durch das Herunterbrechen vieler Merkmalsvektoren auf einen Repräsentanten geht natürlich Information verloren: Nicht alle Ausgangsvektoren fallen mit diesem zusammen. Eine "unscharfe" Modellierung, in der die einzelnen Rohwerte in einem definierten Umfeld mit definierter Wahrscheinlichkeit streuen, wird dem Szenario viel eher gerecht.

[37]Eine *Wahrscheinlichkeitsdichtefunktion* ist eine Funktion, deren Integral von $-\infty$ bis $+\infty$ gleich eins ist. Das Integral über ein beliebiges Intervall a bis b sagt dabei aus, mit welcher Wahrscheinlichkeit eine Zufallsvariable, die einer dieser Dichtefunktion zugrundeliegenden Verteilung folgt, einen Wert aus dem Intervall a bis b annimmt. In Verbindung damit steht die *Wahrscheinlichkeitsverteilungsfunktion*. Sie ist die Stammfunktion der Dichte und sagt aus, mit welcher Wahrscheinlichkeit eine dieser Verteilung folgende Zufallsvariable einen Wert kleiner oder gleich ihrem Funktionswert annimmt.

[38]Diese Annahme ist nicht zu 100 Prozent zutreffend, schließlich folgt die Aneinanderreihung von Lauten in artikulierter Sprache schon einem bestimmten Schema. Statistisch betrachtet kann sie jedoch durchaus gehalten werden, was auch durch die so erzielten Ergebnisse gerechtfertigt wird.

[39]Die visuelle Darstellung schummelt dabei ein wenig: Natürlich muss auch die Dichtefunktion einer Mischverteilung den Flächenwert eins haben, die entstehende Kurve muss also im Nachhinein noch skaliert werden, um dies zu gewährleisten. Diese Skalierung ist in der Abbildung nicht enthalten, um den Zusammenhang zwischen Mischdichte und den Einzeldichten besser herauszustellen.

Wenn nun ein Sprecher auf diese Art - mittels mehrerer Gauß-Funtionen - repräsentiert wird, paart sich der Vorteil des statistischen Umgangs mit Unsicherheit mit dem Vorzug des VQ-Modells, die Merkmalsvektoren durch mehrere Repräsentanten darzustellen. Dies wird erkauft mit dem Nachteil, dass deutlich mehr ursprüngliche Merkmalsvektoren zum Erstellen eines vernünftigen statistischen Modells herangezogen werden müssen als im VQ-Fall [NISH 03a]. Außerdem ist die Trainingsprozedur[40] rechnerisch deutlich komplexer, mittlerweile aber trotzdem in Echtzeit machbar.

Wird nun eine solche Mischverteilung zur Sprechermodellierung herangezogen, spricht man von einem *Gauß'schen Misch-Modell!→ siehe GMM* oder kurz *GMM*. Formal ist es wie folgt definiert [REYN 95]:

$$\lambda = \{p_i, \vec{\mu}_i, \Sigma_i\}, \quad i = 1, ..., M \tag{2.5}$$

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} p_i b_i \vec{x} \tag{2.6}$$

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} e^{(-\frac{1}{2}(\vec{x}-\vec{\mu_i})^t \Sigma_i^{-1}(\vec{x}-\vec{\mu_i}))} \tag{2.7}$$

Ein Modell λ setzt sich zusammen aus den Erwartungswerten und Kovarianzen seiner M Komponenten, jeweils gewichtet mit einem Faktor p[41], welcher der Beschränkung $\sum_{i=1}^{M} p_i = 1$ unterliegt. Die Dichtefunktion $p(\vec{x}|\lambda)$ des Modells ist also die mit diesen Faktoren gewichtete Summe der einzelnen Gaußdichtefunktionen $b_i(\vec{x})$ der Komponenten. D ist dabei die Dimensionalität eines einzelnen Merkmalsvektors $\vec{x}$. Anschaulich kann man sich nun die entstandenen Komponenten des Modells als Repräsentanten einiger breiter, akustischer Klassen (bestimmte Vokale, Konsonanten, ...) vorstellen, welche typisch für den modellierten Sprecher sind und direkt mit der physiologischen Beschaffenheit seines Vokal-Trakts korrespondieren.

Die genaue Erscheinungsform des GMM ist nun noch abhängig von der Beschaffenheit der Kovarianz-Matrix Σ: Sie kann eine vollständig gefüllte Matrix sein, oder aber eine Diagonalmatrix, mit Elementen also nur auf der Hauptdiagonalen. Eine kurze Retrospektive auf Abschnitt 2.4.1 verdeutlicht: Bei Verwendung der MFCC's ist eine volle Kovarianzmatrix nicht sinnvoll, da die einzelnen Koeffizienten durch ihr Entstehungsverfahren untereinander unabhängig sind. Außerdem besagt der *Fluch der Dimensionalität* ja, dass für zusätzliche Parameter auch zusätzliche Trainingsdaten vorhanden sein müssen, die generell immer sehr knapp sind. Deshalb wird in der Praxis häufig der diagonalen Kovarianzmatrix der Vorzug gegeben, wobei die Kovarianz dann anschaulich zum reinen Varianzvektor $\vec{\sigma}^2$ degeneriert. Doch selbst wenn die statistische Unabhängigkeit nicht

[40] Das Generieren eines Modells aus einer Menge an Rohdaten wird generell als *Training* bezeichnet. Im Gegenzug nennt sich die Ermittlung eines Ähnlichkeitswertes zwischen Modell und Rohdaten *Test*. Gleiches gilt für die Bezeichnung der jeweiligen Rohdaten: *Trainings-* und *Testdaten* werden sie genannt.

[41] Erwartungswert, Kovarianz und Gewichtungsfaktor bilden zusammen jeweils eine *Komponente* des GMM. Diese drei Werte aller Komponenten zusammen heißen die *Parameter* des des Modells.

gegeben wäre, zeigt [REYN 95], dass sich der Effekt einer vollen Kovarianzmatrix durch mehrerer diagonale nachahmen lässt, so dass an dieser Stelle mathematisch keine Generalität verloren geht. Dies hat überdies den Vorzug, dass die einzelnen Gaußdichtefunktionen komponentenweise als eindimensionale Funktionen berechnet werden können, was eine deutliche Vereinfachung gegenüber dem mehrdimensionalen Fall mit sich bringt. Anhang A.1 führt hier den entsprechenden Beweis.

Modell-Training mittels Erwartungs-Maximierung

Soll die Ähnlichkeit eines statistischen Modells zu einem Satz Merkmalsvektoren überprüft werden, liefert der Testalgorithmus als Ergebnis einen Likelihood-Wert zurück. Genau umgekehrt ist der Vorgang beim Training: Hier sollen die Modell-Parameter derart ermittelt werden, dass der Likelihood-Wert zwischen dem anzupassenden Modell und seinen Trainingsdaten maximal wird[42]. Die Likelihood der kompletten Daten $X = \{\vec{x_1}, ..., \vec{x_T}\}$ unter der Bedingung, dass sie von dem Modell λ beschrieben werden, is folglich definiert als:

$$p(X|\lambda) = \prod_{t=1}^{T} p(\vec{x_t}|\lambda) \tag{2.8}$$

Leider lässt sich die Gleichung nicht direkt nach λ auflösen, um ein analytisches Maximum zu finden. Doch es existiert ein iteratives Trainingsverfahren, welches garantiert zu einem lokalen Maximum der Likelihoodfunktion findet: Der *Erwartungs-Maximierungs* (*EM*) Algorithmus. Er ermittelt in jedem Schritt, ausgehend von einem initialen Modell λ, ein neues Modell $\overline{\lambda}$, für welches $p(X|\overline{\lambda}) \geq p(X|\lambda)$ gilt. Darauf wird das neue Modell zum Ausgangsmodell, und der Schritt wiederholt sich, bis das Verfahren tatsächlich konvergiert oder durch ein beliebiges Kriterium[43] abgebrochen wird. Die verwendeten Formeln für die Parameterermittlung sind dabei:

$$p(i|\vec{x_t}, \lambda) = \frac{p_i \cdot b_i(\vec{x_t})}{\sum_{m=1}^{M} p_m \cdot b_m(\vec{x_t})} \tag{2.9}$$

$$\overline{p_i} = \frac{1}{T} \sum_{t=1}^{T} p(i|\vec{x_t}, \lambda) \tag{2.10}$$

$$\vec{\overline{\mu_i}} = \frac{\sum_{t=1}^{T} p(i|\vec{x_t}, \lambda) \cdot \vec{x_t}}{\sum_{t=1}^{T} p(i|\vec{x_t}, \lambda)} \tag{2.11}$$

[42] Diese Art der Parameterermittlung nennt sich *Maximum Likelihood* (ML) Methode . Sie basiert auf der Theorie, dass jenes Parameterset eine Menge Daten am besten beschreibt, welches den Wert seiner Likelihood-Funktion maximiert [HENZ 97].

[43] Denkbar wäre hier beisliepsweise, Konvergenz immer nach genau 10 Iterationen anzunehmen.

$$\vec{\sigma_i}^2 = \frac{\sum_{t=1}^{T} p(i|\vec{x_t}, \lambda) \cdot \vec{x_t}^2}{\sum_{t=1}^{T} p(i|\vec{x_t}, \lambda)} - \vec{\mu_i}^2 \tag{2.12}$$

Hierbei ist $p(i|\vec{x_t}, \lambda)$ die a posteriori Wahrscheinlichkeit dafür, dass der aktuelle Merkmalsvektor $\vec{x_t}$ von der i-ten Misch-Komponente beschrieben wird.

Für die Ermittlung des ersten initialen Modells zu Beginn des EM-Algorithmus gibt es keine einheitliche Theorie. Sogar zufälliges Zuordnen der Merkmalsvektoren zu M Klassen, um daraufhin deren Erwartungswerte und Varianzen zu berechnen, hat sich als ausreichend erwiesen. Allerdings kann durch sinnvolleres Aufteilen (beispielsweise mittels einiger Iterationen eines unüberwachten Clustering-Algorithmus wie des k-Means [ESTE 00]) die nicht unerhebliche Rechenzeit des EM-Algorithmus verkürzt werden.

Weiteren Einblick in die Herkunft dieser Formeln und deren Interpretation liefert auch [BILM 98]. Hier ist zudem die Variante für den Aufbau eines *Hidden Markov Models* (HMM) gegeben. Ein HMM ist dabei ein weiteres, vor allem in der Spracherkennung gerne eingesetztes statistisches Modell [RABI 89]. Man kann es sich als eine Art Zustandsautomaten mit $z = 1, ..., Z$ Zuständen vorstellen, wobei jeder Zustand durch eine Wahrscheinlichkeitsdichtefunktion $b_z(\vec{x})$ und eine Zustandsübergangsmatrix A_z definiert ist. Die Wahrscheinlichkeitsdichte $b_z(\vec{x})$ kann dabei eine beliebige Dichtefunktion sein, also beispielsweise auch ein GMM. Ihr Wert gibt an, wie gut der aktuelle Zustand den zu testenden Merkmalsvektor beschreibt. Die Zustandsübergangsmatrix A_z enthält für jeden der Z Zustände Zeile und Spalte, so dass in jeder ihrer Zellen eine Wahrscheinlichkeitsdichtefunktion $a_{kl}(\vec{x})$ die Wahrscheinlichkeit angibt, mit der bei vorliegendem Merkmalsvektor $\vec{x}$ von Zustand k in Zustand l gewechselt wird [RABI 86]. Zur vollständigen Definition des Modells gehört nun noch der Anfangszustand π desselben.

Durch das Mittel der Zustandsübergänge lässt sich so eine Bedingung für die Reihenfolge der zu erwartenden Daten formulieren: In Abhängigkeit nur von dem letzten Datum wird zu einem Zustand mit vielleicht ganz anderen Erkennungseigenschaften gewechselt[44]. Ausgangspunkt für diese Entwicklung war die Erkenntnis, dass zu modellierende Daten oft auf einem gewissen Abschnitt ein recht gleichförmiges Verhalten zeigen, dann aber sprunghaft zu einem anderen Verhalten wechseln. Dort verharren sie wieder eine Weile, bevor der Prozess weitergeht. Ein Beispiel aus der textabhängigen Sprach-Erkennung mag dies verdeutlichen:

Ein HMM zur Erkennung des Wortes "`und`" mag aus drei Zuständen bestehen: Der erste enthält eine Wahrscheinlichkeitsdichte, die trainiert auf die Erkennung des Lautes `\u` ist. Der nächste Zustand ist spezialisiert auf ein `\n`, der letzte auf `\d`. Der Anfangszustand ist derjenige, welcher ein `\u` erkennt, und die Zustandsübergangsmatrix ist dergestalt, dass bei jedem richtig erkannten Laut in den nächsten Zustand gewechselt wird, bei einem Nicht-Erkennen jedoch wieder in den Ersten. Bei erfolgreichem Durchlaufen aller drei

[44] Formal bezeichnet man eine Folge von Ereignissen, die sich solcherart modellieren lassen, als *Markov-Kette* [STUT 02], was zur Namensgebung des Modells beitrug.

Zustände wurde somit das Wort "und" erkannt, obwohl vielleicht die einzelnen Modelle der Phoneme so unterschiedlich zueinander sind, dass diese Aufgabe mit einem einzigen GMM schwierig geworden wäre.

Neuronale Netze

Ein im Ansatz schon anderes Prinzip verfolgt der Zweig der Sprechererkennung mittels *Neuronaler Netze* (NN). Denn hier wird nicht ein Modell eines Sprechers aufgebaut, sondern ein Klassifikator, der zwischen einer von vorne herein bekannten Anzahl Sprecher unterscheiden kann. Man kann sich diese Idee also mit einigem Recht als eine Art großes Modell für eine klar definierte Gruppe von Sprechern vorstellen. Neuronale Netze haben die Eigenschaft, scharf zwischen einer bestimmten Menge an Klassen zu unterscheiden, was bedeutet, dass sie im Gegensatz zu den statistischen Modellen keine Möglichkeit haben, mit einander überlappenden Klassen umzugehen, was aber nicht unbedingt ein Nachteil sein muss[45]. Ein wirklicher Nachteil allerdings ist die Festlegung auf eine bestimmte Gruppe von Sprechern oder Klassen: Kommt eine neue hinzu, muss der langwierige Trainingsprozess erneut für alle Klassen erfolgen. Trotzdem finden Unterarten Neuronaler Netze immer wieder Anwendung in Gebieten, in denen Cluster willkürlicher Form gebildet und klassifiziert werden müssen, so auch in [LAPI 02] zum Zweck der Sprechererkennung.

Mit ihrer Arbeitsweise fallen Neuronale Netze zur Sprechererkennung schon beinahe in die Kategorie der Klassifikatoren, da sie ja nicht direkt einen Sprecher, sondern die Unterschiede zwischen einer Gruppe von Sprechern modellieren. Dies bringt ihnen auch einige Vorteile: Im Gegensatz zu GMM's und HMM's, die jeweils die Information eines Sprechers möglichst gut darstellen wollen, kann hier auf die zusätzliche Information der Klassenunterschiede eingegangen werden. Im Bild gesprochen kann man sich diesen Vorteil folgendermaßen verdeutlichen: Wenn zwei bestimmte Personen, ein Mann und eine Frau, anhand ihres Äußeren unterschieden werden sollen, kann das durch den Vergleich ihrer besonderen, aber eventuell gemeinsamen Merkmale geschehen (die GMM/HMM-Methode): Beide haben zwei Augen, einen Kopf, vier Gliedmaßen,.... Natürlich wird auch dieser Ansatz irgendwann zum Erfolg führen, doch ein Blick auf die Klassenunterschiede (der NN-Ansatz), die geschlechtsspezifischen in diesem Fall, brächte das Ergebnis vielleicht schneller und sicherer.

2.4.5 Sprach-Anreicherung

Erlangung von Unabhängigkeit im Bezug auf Aufnahmebedingungen und akustisches Umfeld sei die größte noch verbliebene Herausforderung im Bereich der Sprachverarbeitung,

[45] Zu einem Nachteil gereichte es nur, wenn die statistischen Modelle jeweils immer sinnvoll mit der Überlappung umgingen und trotz unzulänglicher Daten die von höherer Warte betrachtet "richtigen" Entscheidungen treffen würden; dem ist allerdings nicht immer so.

sagt [REYN 00]. Sie wird oft unter dem Begriff *Sprach-Anreicherung*[46] zusammengefasst [LOGA 98]. Einen guten Überblick über dieses Gebiet gibt [GONG 95], wobei die dortigen Informationen noch auf dem Stand von vor circa zehn Jahren sind. Der folgende Abschnitt möchte auch neue Entwicklungen auf diesem Terrain erfassen. Die zu diesem Zweck entwickelten Verfahren konzentrieren sich dabei auf drei verschiedene Ansatzpunkte oder Typen innerhalb der Verarbeitungskette eines Sprachverarbeitungs-Systems [ROSE 94]:

1. *Die Aufnahme- und Übertragungsbedingungen selbst, bis hin zu dem Weg des Signals durch die digitale Pforte.* Diese Techniken arbeiten mit speziellen Mikrofonen und Übertragungsmedien, aber auch mit *Geräusch-Auslöschungs* Algorithmen während der Aufzeichnung. Hiermit sind sie jedoch meist außerhalb des Eingriffsbereichs des Sprachverarbeitungs-Systems und bleiben deshalb bei den folgenden Betrachtungen außen vor.

2. *Die Signalverarbeitung zu Anfang des Software-kontrollierten Prozesses.* Hierzu zählen Techniken zur nachträglichen Verbesserung des *Signal-Geräusch-Abstands* genauso wie auch die Bildung spezieller Merkmale, welche eine höhere Robustheit gegenüber Geräuscheinflüssen aufweisen.

3. *Die Repräsentation von Sprache oder Sprecher an sich.* Hier wird die Reinigung von Störeinflüssen direkt in den Modellierungsprozess integriert, indem Wissen über die akustische Umgebung während der Aufnahme eingebracht wird.

Zu den Typ-Zwei Techniken zählt beispielsweise die Methode der *Cepstralen Mittelwert-Subtraktion* (CMS) oder -*Normalisierung* (CMN) [YU 02]. Hier wird (bei cepstralen Merkmalsvektoren) der Langzeitmittelwert gebildet und von den einzelnen Merkmalsvektoren subtrahiert. Die Idee dahinter ist, dass die Langzeitstatistiken des Cepstrums zwar schon sprecherspezifisch sind, jedoch immer noch leicht variieren. Bei Annahme konstanter Hintergrundgeräusche sollte eine Entfernung des Mittelwertes also gerade diese hintergrundspezifischen spektralen Veränderungen eliminieren und die sprecherspezifische Information damit besser zur Geltung bringen. Tatsächlich leistet das Verfahren dort, wo diese Annahme gerechtfertigt ist, Erstaunliches[47]. Ist allerdings von variierendem Hintergrund auszugehen, kann sich der Vorteil ins Gegenteil verkehren, wie [DUNN 00] klarstellt: Die Subtraktion bringt hier neue Verzerrungen mit sich, anstatt die alten zu lindern.

GMM mit Integriertem Hintergrund

[46] Dieser Begriff ist im Ursprung etwas weiter gefasst als das hier spezifizierte Teilgebiet. Zur *Sprach-Anreicherung* zählen auch Methoden, um Sprache für Menschen verständlicher zu machen.

[47] Beispielsweise im fahrenden Auto, wo die Motorengeräusche relativ konstant sind [CAMP 97].

Die Methode der *Geräusch-Maskierung* hingegen gehört dem dritten Typ von Verfahren an. Ursprünglich publiziert in [HOLM 86], bietet sie ein Rahmenwerk, um zwischen Training und Test der Sprechermodelle gleiche Bedingungen herzustellen: Wenn einzelne Subbänder[48] durch Geräusche oberhalb eines bestimmten Schwellwertes B überlagert sind, kann der in ihnen vielleicht trotzdem enthaltenen Sprachinformation nicht mehr getraut werden. Zur Bestimmung der Modellparameter eines statistischen Modells kommen nun also nur diejenigen Merkmalsvektor-Komponenten zum Zuge, deren Werte oberhalb des Schwellwertes B liegen. Die übrigen sind für das Trainingsverfahren unsichtbar, eben maskiert.

Nádas et al. entwickelten nun in [NADA 88] diesen Ansatz wie folgt weiter: Bei Verwendung von Merkmalsvektoren, die auf logarithmierten Energiewerten einer Filterbank basieren[49], entwickelten sie eine statistische Umgebung für das oben beschriebene Verfahren. Hier wird das Hintergrundgeräusch nicht als fester Schwellwert, sondern als unabhängiger, statistischer Prozess beschrieben, der im Grunde ebenso modelliert werden kann wie der Sprachprozess: Durch eine Mischung mehrerer Gauß-Verteilungen. Die Vor- und Nachteile für diesen Schritt entsprechen dabei denjenigen bei der Erweiterung des VQ-Modells zum GMM.

In [NADA 89] wurde die Interaktion zwischen dem Sprach- und Hintergrundprozess auf Basis der Log-Energiewerte folgendermaßen beschrieben: Die beobachtbaren Daten z entstehen aus dem eigentlichen Signal x und dem Störeinfluss y durch eine Maximum-Funktion $z = max(x, y)$. Die Wahl dieser Formel findet ihre Rechtfertigung dabei eher in der Beobachtung des Resultats und der Einfachheit der weiterführenden Formeln denn in präzisen mathematischen Überlegungen. Der Ansatz wurde als *MIXMAX-Modell* bekannt.

[ROSE 91] vereinte erstmals dieses Verfahren mit einem Standard-GMM zum Ziel der Sprechererkennung. Außerdem wurde der Formelsatz verallgemeinert, um prinzipiell mit beliebigen Arten der Signal-Hintergrund-Interaktion umgehen zu können. Eine spezielle Lösung wurde für ein rein additives Interaktionsmodell unter Verwendung linearer Merkmalsvektoren gegeben: $z = x + y$. Des Weiteren wurde gezeigt, dass der Fall additiver Überlagerung innerhalb der Log-Domäne durch die Benutzung obiger Maximum-Funktion angenähert werden kann[50]: $(e^z = e^x + e^y) \approx (z = max(x, y))$. In Experimenten zeigte sich, dass das Verfahren zwar für impulsartige Nebengeräusche ungeeignet ist, jedoch unter konstanten Bedingungen den Vorteil der Schwellwertfreie besitzt: Es wird lediglich ein Muster des Störgeräuschs und die Art seiner Interaktion mit dem Signal benötigt, schon funktioniert die Sprachanreicherung ohne individuelle Parameter-Feineinstellung von Hand. Diese Methode wurde anschließend noch in [REYN 92] überarbeitet und schlussendlich in [ROSE 94] als *Gauß'sches Misch-Modell mit Integriertem Hintergrund* (GMM-IB[51])

[48] Die Entsprechung der spektralen *Subbänder* (Frequenzabschnitte) findet sich bei Verwendung der MFCC's in den einzelnen Koeffizienten (Dimensionen) eines Merkmalsvektors wieder.

[49] Dies trifft unter anderem auf MFCC's zu.

[50] Dieser Fall wird im Folgenden mit der Bezeichnung *Max-Modell* belegt.

[51] Aus dem Englischen für *Gaussian Mixture Model with Integrated Background*.

umfassend präsentiert.

Nach diesem Überblick über Ursprung und Entstehungsgeschichte des GMM-IB soll nun näher auf seine formale Definition und anschauliche Funktionsweise eingegangen werden. Zunächst wird die Sprachkomponente λ_s wie beim herkömmlichen GMM auch durch die Formeln 2.5 bis 2.7 beschrieben. Hinzu kommt jetzt noch der Hintergrundprozess λ_b, der entsprechend ähnlich aufgebaut ist[52]:

$$\lambda_b = \{q_j, \vec{\mu_j}, \Sigma_j\}, \text{j=1,...,N} \tag{2.13}$$

Hintergrund $\vec{y_t}$ und Signal $\vec{x_t}$ sind jedoch nicht in dieser Form direkt beobachtbar. Vielmehr sind sie durch den Einfluss einer zunächst einmal abstrakten Funktion $f(.)$ miteinander verflochten. Beobachtbar ist nur das Ergebnis $\vec{z_t}$ dieser Verflechtung. Abbildung 2.8 veranschaulicht diesen Prozess:

Sprache und Hintergrund stellen jeweils eigenständige stochastische Prozesse dar, die in jeder Zeiteinheit $t = 1, ..., T$ ein Datum $\vec{x_t}$ bzw. $\vec{y_t}$ generieren. Wie diese beschaffen sind, hängt mit dem jeweiligen Zustand i oder j des Modells zusammen: Sie sind Verweise auf die i-te (j-te) Komponente der Mischdichtefunktion $p(\vec{x_t}|\lambda_s)$ bzw. $p(\vec{y_t}|\lambda_b)$. Die jeweils aktuellen $\vec{x_t}$- und $\vec{y_t}$-Vektoren interagieren nun miteinander, was durch die Funktion $f(.)$ angedeutet ist. Ergebnis dieser Interaktion ist das am Ende hörbare Geräusch, welches aus Sprache und Nebengeräusch besteht.

Mathematisch ergibt sich so für die Dichte einer einzelnen Beobachtung $\vec{z_t}$ die folgenden Formeln:

$$p(\vec{z_t}|i_t, j_t, \lambda_s, \lambda_b) = \int\int_C b_{i_t}(\vec{x_t}) \cdot a_{j_t}(\vec{y_t}) d\vec{x_t} d\vec{y_t} \tag{2.14}$$

$$p(\vec{z_t}|\lambda_s, \lambda_b) = \sum_{i=1}^{M} \sum_{j=1}^{N} p_i \cdot q_j \cdot p(\vec{z_t}|i, j, \lambda_s, \lambda_b) \tag{2.15}$$

Hierbei beschreibt Formel 2.14 die Dichte für einen speziellen Zustand (i_t, j_t). Das enthaltene Integral ist hierbei ein sogenanntes Linien- oder Kurvenintegral, dessen Integrationskontur von der genauen Beschaffenheit der Interaktionsfunktion $f(.)$ abhängt. Im Falle der MIXMAX-Annahme mit $z = f(x, y) = max(x, y)$ wäre dies also eine Figur in der x-y-Ebene, die für $y \geq x$ eine Gerade auf Höhe des y-Wertes darstellt, bis sie für $y < x$ auf die x-Achse abfällt[53].

[52] Der Übersichtlichkeit halber werden sämtliche wichtigen Formeln das GMM-IB betreffend nochmals in Anhang A.4 wiederholt.

[53] In Abbildung 3.4 ist diese Kontur zusammen mit weiteren Beispielen visualisiert.

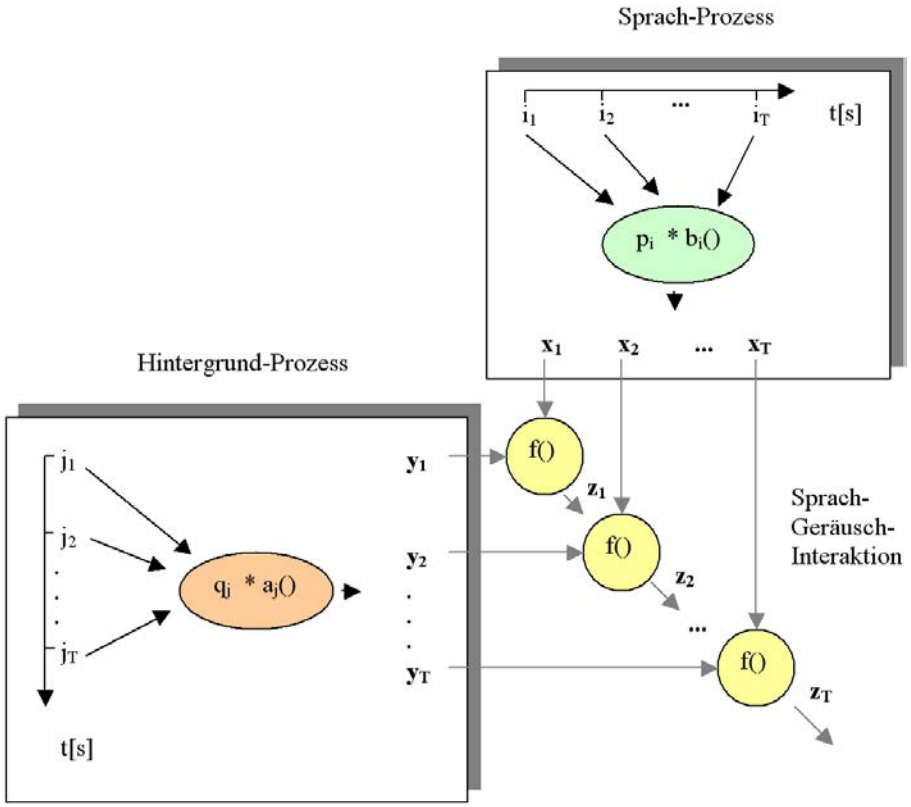

Abbildung 2.8: Signal und Geräusch als eigenständige, einander beeinflussende Prozesse (nach [ROSE 94]).

Vielleicht ist an dieser Stelle nochmals eine Rekapitulation angebracht: Was ist bis jetzt formuliert worden, wozu kann dies führen? Es existiert eine Audioaufzeichnung, deren beobachtbare Werte z_t durch eine zunächst abstrakt eingeführte Funktion $f(.)$ aus dem reinen Signal x_t und einem Störeinfluss y_t hervorgehen. Diese beiden Werte sind verborgen, also nicht beobachtbar! Das Auftreten der einzelnen x_t wird nun als generativer, stochastischer Prozess angesehen, dessen Parameter $\lambda_s = \{p_i, \vec{\mu}_i, \Sigma_i\}$ $(i = 1, ..., M)$ bekannt sind. Gleiches gilt für den Hintergrundprozess, der als λ_b bezeichnet wird. Nun lässt sich das Auftreten der z_t's einzig und allein durch die Kenntnis der beiden Prozesse λ_s und λ_b sowie der konkreten Form von $f(.)$ anhand von Formel 2.15 erklären[54].

Das ist an sich noch kein großer Schritt zur Lösung des hier vorgestellten Problems. Interessant wird es allerdings, wenn dieser Vorgang rückwärts gedacht wird: Angenommen, es existiert schon ein explizites statistisches Modell λ_b, das aus reinen Werten des Nebengeräusches über die normale Trainingsprozedur für GMM's entstanden ist. Weiterhin angenommen, es existiert ebenfalls ein Modell für die gesamte verrauschte Beobachtung, erstellt auf identischem Weg. Dann ließe sich nach Umstellung obiger Formeln das Modell λ_s der reinen Sprache erstellen, befreit von den störenden, aber explizit bekannten Einflüs-

[54] Das heißt, es lässt sich ein statistisches Modell erstellen.

sen des Hintergrunds[55]. Diese Formeln lassen sich natürlich wieder nicht analytisch lösen. Stattdessen muss der iterative EM-Algorithmus erneut eingesetzt werden. Nach einiger Umstellung *sieht man leicht*[56]:

$$p(i_t{=}i, j_t{=}j|\vec{z_t}, \lambda) = \frac{p(\vec{z_t}|i_t{=}i, j_t{=}j, \lambda) \cdot p_i \cdot q_j}{\sum_{i=1}^{M}\sum_{j=1}^{N} p(\vec{z_t}|i_t{=}i, j_t{=}j, \lambda) \cdot p_i \cdot q_j} \tag{2.16}$$

$$\overline{p_i} = \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|\vec{z_t}, \lambda) \tag{2.17}$$

$$\overline{\mu_i} = \frac{\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|z_t, \lambda) \cdot E\{x_t|z_t, i_t{=}i, j_t{=}j, \lambda\}}{\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|z_t, \lambda)} \tag{2.18}$$

$$\overline{\sigma_i}^2 = \frac{\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|z_t, \lambda) \cdot E\{x_t^2|z_t, i_t{=}i, j_t{=}j, \lambda\}}{\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|z_t, \lambda)} - \overline{\mu_i}^2 \tag{2.19}$$

Die Wahrscheinlichkeitsdichte in Formel 2.16 ist dabei genauso wie die beiden bedingten Erwartungswerte $E\{\}$ abhängig von speziellen Formen von $f(.)$, welche an dieser Stelle keine weitere Beachtung finden soll. Es ist jedoch noch auf einige notatorische Besonderheiten obiger Gleichungen hinzuweisen: λ_s und λ_b wurden zugunsten größerer Übersichtlichkeit zu λ zusammengefasst. Des Weiteren bezeichnen x_t, μ_i und σ_i^2 hier und im weiteren Verlauf dieser Arbeit jeweils willkürliche Komponenten der entsprechenden Vektoren $\vec{x_t}$, $\vec{\mu_i}$ und $\vec{\sigma_i}^2$. Diese Annahme ist gerechtfertigt dadurch, dass – man konnte es ebenfalls bereits der Notation σ_i^2 anstatt Σ_i entnehmen – hier wieder mit diagonalen Kovarianzen gearbeitet wird. Dies führt, wie Anhang A.1 zeigt, zu der Möglichkeit komponentenweiser Berechnung. Mit diesen Formeln ist es nun möglich, den Likelihood-Wert der zu testenden Daten Z unter Annahme des Modells λ anzugeben:

$$p(Z|\lambda) = \prod_{t=1}^{T}\sum_{i=1}^{M}\sum_{j=1}^{N} p_i \cdot q_j \cdot p(i_t{=}i, j_t{=}j|\vec{z_t}, \lambda) \tag{2.20}$$

[55] Um verfrühtem Enthusiasmus vorzubeugen: Natürlich ist mit dieser "Befreiung" von Nebengeräuschen nicht gemeint, dass zerstörte oder durch Überlagerung fehlende Signalanteile der Original-Sprache wiederhergestellt werden können. Vielmehr bietet sich so die Möglichkeit, ein statistisches Modell genau auf den Signalanteilen einer verrauschten Aufnahme basieren zu lassen, die nicht zum Hintergrund gehören. Anschaulich heißt das beispielsweise: Ein Sprecher artikuliert seine Sätze vor dem Hintergrund einer dröhnenden Bass-Anlage. Deren Statistik ist bekannt. So wird nun ein Sprechermodell erstellt, in welchem die hohen Frequenzen der Stimme wie üblich modelliert sind. In den tiefen Frequenzen allerdings wird es relativ leer aussehen, da alle Signalanteile, die in den Bereich der Bass-Anlage fallen, außen vor gelassen werden. Allerdings wird nicht einfach jede tiefe Frequenz verworfen, sondern durch das statistische Rahmenwerk wird so viel Information wie möglich auch aus diesem Bereich genutzt.

[56] Ein so oder in ähnlicher Form in der einschlägigen Fachliteratur häufig eingesetzter Euphemismus: Zu übersetzen ist er gewöhnlich mit "nach wochenlangen, komplexen Berechnungen kam glücklicherweise gegen Ende die rettende Idee…" ;-).

Die GMM-IB Technik hat bereits in einigen Bereichen ihre Eignung zeigen können, so beispielsweise auch in [FLEX 01] zur Auswertung von EEG-Daten[57]. Bisher wurde sie allerdings ausschließlich mittels synthetischer Daten, jedoch niemals anhand alltäglicher Situationen, getestet.

Weitere modellbasierte Verfahren

Mark Gales verfolgt in [GALE 95] einen im Grundsatz etwas anderen Ansatz: Zwar ist es auch hier erklärtes Ziel, einmal angefertigte Sprechermodelle unter verschiedenen akustischen Bedingungen nutzbar zu machen. Doch als Grundlage dienen a priori vorhandene Hidden Markov Modelle reiner Sprache und Nebengeräusche. Im Umfeld der Sprecheridentifikation und Sprecher-Adaption[58], wo die Probanden dem Erkennungssystem tatsächlich vorher bekannt sind, bringt diese Methode der *Parallelen Modell-Kombination* (PMC[59]) sehr gute Ergebnisse [ROSE 00].

In [SANK 96] wurde eine Methode publiziert, die ihre Wurzeln in den Arbeiten rund um das GMM-IB hat. Sie nennt sich *Stochastische Zuordnung* und beschreibt ein Verfahren für Maximum-Likelihood Näherungen bestimmter Filter-Funktionen. Mittels dieser können korrumpierte Merkmalsvektoren dergestalt transformiert werden, dass sie besser zu in Stille trainierten Modellen passen. Andersherum ist es auch möglich, "saubere" Modelle an verunreinigte Sprache anzupassen. Dabei beruht die Filterung einzig auf den schon bekannten sauberen Modellen, den zu testenden Daten und dem Wissen um die Klasse[60] des Hintergrundprozesses. [SIOH 97] hingegen greift eine andere Idee auf und vermengt sie mit Gales' PMC-Verfahren, um sowohl additive wie auch faltungsbasierende Interaktionen zwischen Signal und Geräusch gleichzeitig und in einem Schritt auszulöschen.

[LOGA 98] untersucht all diese Verfahren im Rahmen einer Dissertation. Als Ergebnis steht am Schluss eine Erweiterung für herkömmliche HMM's, welche erneut an die Techniken im Zusammenhang mit [ROSE 94] angelehnt sind. Ziel dieses Ansatzes ist wiederum die Verbesserung von Sprach-, nicht von Sprechererkennungs-Systemen.

Ein eher intuitiv geprägter, aber nichtsdestotrotz sehr interessanter Ansatz findet sich noch in [YOSH 01]: Hier werden, nach Subbändern getrennt, GMM's für verschiedene Intensitäten *Weißen Rauschens*[61] erstellt. Eine ankommende, zu klassifizierende Stim-

[57] EEG ist das Akronym für *Elektro-Encephalogramm* und bezeichnet eine Methode zur Messung der Hirnströme.

[58] Die Leistung automatischer Spracherkennung lässt sich deutlich steigern, wenn die Sprachmodelle an den aktuellen Sprecher angepasst werden. Dieser Vorgang heißt *Sprecher-Adaption*.

[59] Aus dem Englischen für *Parallel Model Combination*.

[60] Additiv oder mittels Faltung.

[61] Als *Weißes (Gauß'sches) Rauschen* bezeichnet man ein Signal, dessen einzelne Werte völlig unkorreliert sind und welches sich aus sämtlichen Frequenzen gleicher Intensität zusammensetzt [HOFF 03]. Vom menschlichen Gehör wird es trotzdem als unangenehm hart und hochfrequent, beinahe klirrend, wahrgenommen. Daneben existieren noch weitere spezifizierte Arten von Rauschen, welche der Anschauung

me wird nun auch in Subbänder zerlegt, und für jedes Band das am besten passende Geräuschmodell ausgewählt. Das Verfahren zeigt gute, einem Standard-GMM System überlegene Leistungen. Hervorzuheben bleibt noch, dass die Modelle Weißen Rauschens offenbar auch für die Eliminierung anderer, verwandter Geräuscharten wie beispielsweise Umgebungsgeräusche in Bus, Auto oder Büro geeignet sind.

2.4.6 Klassifikation

Nun, da das System interne Repräsentationen der Sprecher einzelner Segmente hat, und nachdem diese durch die Sprachanreicherung auf ein einheitliches akustisches Niveau gebracht wurden, kann ein Vergleich dieser Repräsentationen untereinander erfolgen: *"Welche der vielen unterschiedlichen Repräsentationen gehören im Grunde zu einem einzigen Sprecher? "* ist die Fragestellung, welcher die Klassifikationsphase nachgeht. Sie ist leicht verschieden von derjenigen in reinen Sprechererkennungs-Systemen, wo es gilt, genau einer Repräsentation eine von mehreren bekannten Identitäten zuzuweisen.

Für den letzteren Fall existieren viele grundverschiedene Ansätze. Einerseits kann einfach anhand der Likelihood-Werte entschieden werden, die ein statistisches Sprechermodell als Ergebnis des Tests einer Menge von Merkmalsvektoren ermittelt: Der wahrscheinlichste Sprecher einer Äußerung ist derjenige, für welchen der Likelihood-Wert dieser Daten maximal ist. Eine weitere Methode ist die der Klassifikation mittels *Polynomieller Klassifikatoren*. Für sie gilt dabei dasselbe, was gegen Ende von Abschnitt 2.4.4 über Neuronale Netze gesagt wurde: Sie haben ihre Vorteile durch die Ausnutzung der Interklasseninformation, sind zudem gut in bestehende statistische Systeme zu integrieren und nicht sehr rechen- und speicherintensiv. In der Leistung sind sie mit herkömmlichen statistischen Systemen vergleichbar [CAMP 02]. Nicht zuletzt soll auch die Methode der *Support Vector Machines* (SVM) Erwähnung finden: Diese Methode des maschinellen Lernens wurde ebenfalls bereits erfolgreich auf dem Gebiet der Sprechererkennung angewandt [FENG 02].

Clustering

Im Fall der Sprecherklassifikation aber muss auf andere Verfahren zurückgegriffen werden: Es gilt nicht nur, ähnliche Repräsentationen zu vereinen. Die Anzahl der am Ende zu erstellenden Sprecherklassen muss zusätzlich automatisch ermittelt werden. Dies ist eine klassische Aufgabe für *agglomerative (hierarchische) Clustering-Verfahren*. Die allgemeine Funktionsweise ist dabei die folgende [KOJR 02]:

- Beginne mit jeder Repräsentation in einem eigenen Cluster.

halber mit Farbwerten beschrieben werden: Während *Pinkes Rauschen* auch noch relativ "hart" klingt, wirkt *Braunes Rauschen* sehr weich, etwa wie ein sanfter Windstoß.

- Berechne die Distanzmatrix anhand einer geeigneten *Metrik*, in welcher die Entfernung zwischen jedem Paar von Clustern festgehalten wird.
- Vereine die beiden Cluster, die sich am ähnlichsten sind. Streiche die entsprechenden beiden Zeilen der Distanzmatrix und füge eine neue hinzu, in welche der Abstand des neu entstandenen Clusters zu allen weiteren eingetragen wird. Vermerke das Niveau (den Abstand), auf welchem die Verschmelzung stattfand.
- Wiederhole die letzten beiden Schritte, bis nur noch ein großer Cluster existiert.

Durch die Verschmelzungen entsteht mit der Zeit ein Baum, in welchem die Blätter die ursprünglichen Repräsentationen sind, und dessen Wurzel der eine große Cluster am Schluss des Verfahrens ist. Visualisiert man diesen Baum und wählt dabei die Länge der Äste äquivalent zu dem Verschmelzungsniveau der Vereinigung auf dieser Stufe, entsteht ein *Dendrogramm*. Um letzten Endes eine aussagekräftige Aufteilung der Daten zu erhalten, muss nun an geeigneter Stelle in diesem Baum ein Schnitt durchgeführt werden. Die resultierenden einzelnen Äste sind dabei die finalen Cluster. Zusammen betrachtet spricht man nun auch von einer bestimmten *Partitionierung* der Daten, wie auch Abbildung 2.9 verdeutlicht.

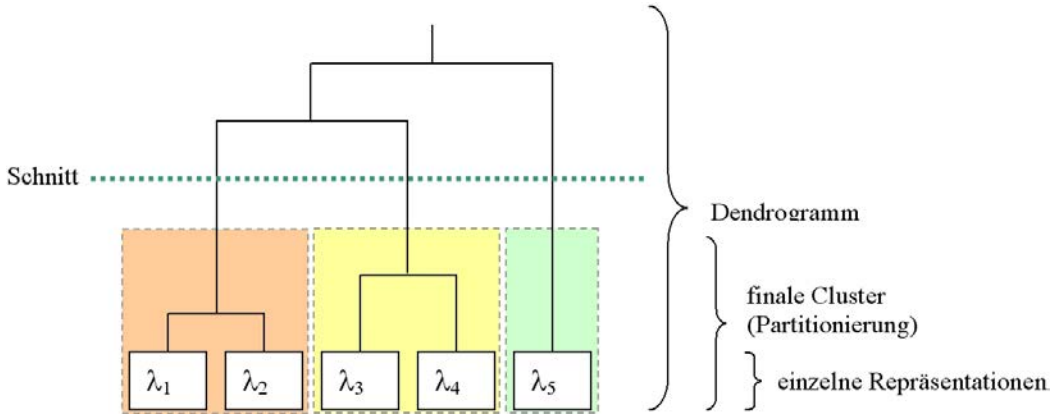

Abbildung 2.9: Dendrogramm und finale Partitionierung einer Menge von Sprechermodellen

Hierarchisches Clustering findet tatsächlich sehr häufig Anwendung in der Sprecherklassifikation. So beispielsweise in [MEIG 02], wo es für die Klassifikation von Sprechern über Dokumentgrenzen[62] hinweg eingesetzt wird. Ursprünglich wurde es jedoch in der Sprecheradaption zur Verbesserung der Wortfehlerrate als unüberwachte Vorstufe zur Spracherkennung verwendet [JIN 97]. Aber eine Übertragung auf andere Bereiche ließ nicht lange auf sich warten [SOLO 98].

[62] Unter *Dokument* ist in diesem Kontext eine abgeschlossene Audioaufzeichnung zu verstehen. Dokumente könnten also beispielsweise die Austrahlungen der Spätnachrichten mehrerer Tage sein, in denen jeweils identische Sprecher auftreten.

Der große Vorteil der agglomerativen Verfahren besteht darin, dass sie die Gesamtheit aller Daten (Sprechermodelle im speziellen Fall) betrachten können, um global zu entscheiden, welche sich am meisten ähneln, und daher verschmolzen werden sollen. Dies geht allerdings einher mit einem Laufzeitproblem: Durch die Berechnung der Distanzmatrix steigt der Rechenaufwand exponentiell mit der ursprünglichen Anzahl zu betrachtender Klassen [LIU 03]. Deshalb werden in letzter Zeit häufig neue Methoden zum *Online-Clustering* präsentiert: Verfahren, welche einen Datenstrom in Echtzeit clustern können, also immer nur auf den bereits vorhandenen Daten arbeiten. Die Vorteile liegen in einem nur linearen Aufwand und der Chance, den Datenstrom nicht zwischenspeichern zu müssen. Durch die zum Entscheidungszeitpunkt nur unvollständig vorliegenden Daten fehlen diesem Ansatz jedoch wichtige Informationen, die ihn im Vergleich zu herkömmlichen Verfahren in der Leistung schlechter aussehen lassen[63]. Ein weiterer Nachteil hierarchischer Verfahren ist der der Fehlerfortpflanzung: Wird zu Beginn eine falsche Verschmelzung getätigt, beeinträchtigt dies alle darauf aufbauenden Schritte. Dieses und andere Probleme behandelt unter anderem [WALL 04].

Daneben existieren noch einige Spezialfälle als Lösungen für sehr spezifische Probleme, wie etwa der Ansatz in [MCLA 99] zum Clustern von Sprachäußerungen, die genau zwei Sprecher enthalten. McLaughlin et al. schlagen hierin ein Verfahren vor, welches die Segmente nicht zuerst in sprecherhomogene Abschnitte unterteilt. Statt dessen erweitern sie das Clustering-Verfahren dergestalt, dass die inhomogenen Segmente genau zwei statt einem Cluster zugewiesen werden können.

Distanzmaße

Um während des Aufbaus der Distanzmatrix die Ähnlichkeit zwischen zwei Datensätzen zu bestimmen, ist ganz allgemein ein *Distanz- oder Abstandsmaß* nötig[64]. Die Art dieses Distanzmaßes ist in der Regel sehr aufgabenspezifisch, obwohl es auch allgemein verwendbare wie die *L_p-Metrik* gibt: $d(x,y) = \sqrt[L]{\sum_{d=1}^{D}(x_d - y_d)^p}$. Durch Wählen von $L = p = 2$ ergibt sich daraus die *Euklid'sche Distanz*. Prinzipiell kann jede Funktion $d(x,y)$ als Abstandsmaß dienen, welche folgende Bedingungen erfüllt [ESTE 00]:

$$d(x,y) = d \in \mathbb{R}^{\geq 0} \tag{2.21}$$

$$d(x,y) = 0 \Leftrightarrow x = y \tag{2.22}$$

$$d(x,y) = d(y,x) \tag{2.23}$$

[63] Jedoch nicht so viel schlechter, wie man anhand der Informationslage vielleicht mutmaßen würde. Liu et al. wagen deshalb in [LIU 03] die Vermutung, dass in agglomerativen Verfahren noch einiges an brach liegendem Potenzial zu entdecken sein müsste.

[64] Natürlich ist auch die positive Definition beider Begriffe möglich: Nähe- oder Ähnlichkeitsmaß. Ist 0 die minimale Distanz und 1 die maximale, so gilt der Zusammanhang: *Ähnlichkeitsmaß = 1 - Distanzmaß*. Dies läßt sich durch geeignete Normierung immer erreichen.

$$d(x,z) \leq d(x,y) + d(y,z) \tag{2.24}$$

Die letzte Gleichung[65] muss nicht erfüllt sein. Ist sie es trotzdem, ist das Distanzmaß auch eine *Metrik*.

In der Literatur sind mehrere spezialisierte Abstandsmaße zur Sprecherklassifikation gebräuchlich. Sie basieren beinahe ausschließlich auf dem Vergleich von Likelihood-Werten, was natürlich wieder mit der statistischen Modellierung in der Vorstufe einhergeht. Das bedeutet auch, dass zwei Sprechermodelle nicht direkt miteinander verglichen werden können. Vielmehr findet der Vergleich immer zwischen einem Modell und einer Menge von Merkmalsvektoren statt[66]. Üblich ist beispielsweise das *Generalisierte Likelihood-Verhältnis* (GLR[67]), welches nach seinem Erfinder auch *Gish-Metrik* genannt wird:

$$d_{GLR}(\lambda_x, \lambda_y) = -\frac{L(X \cup Y|\lambda_{x\cup y})}{L(X|\lambda_x) \cdot L(Y|\lambda_y)} \tag{2.25}$$

$$= -\log L(X \cup Y|\lambda_{x\cup y}) + \log L(X|\lambda_x) + \log L(Y|\lambda_y) \tag{2.26}$$

Hierbei sind λ_x und λ_y Sprechermodelle, X und Y die Trainingsmengen der Merkmalsvektoren dieser Modelle und $L(.)$ ist die Likelihood-Funktion der Daten unter der Bedingung des Modells. $X \cup Y$ stellt die Vereinigung der beiden Ursprungsmengen dar, $\lambda_{x\cup y}$ ist entsprechend das Modell trainiert auf dieser vereinigten Merkmalsmenge.

Der große Vorteil der GLR ist ihre Genauigkeit: Hier wird tatsächlich gemessen, inwiefern sich die Verschmelzung zweier Modell auf die Güte auswirkt, mit der dieses neue, große Modell auf die ursprünglichen Segmente passt. Ein großer Nachteil ist dafür die Komplexität: Für jeden Vergleich muss ein neues statistisches Modell geprägt werden, mit allem, was dies an EM-Iterationen und aufwendigen Rechenoperationen mit sich bringt [MEIG 02]. Trotzdem wird die GLR gerne und oft eingesetzt, zum Beispiel in [SOLO 98], [BONA 02] oder [LIU 03].

Ein anderes Maß überwindet diese Laufzeitprobleme, jedoch zu Lasten der Wirksamkeit: Die CLR[68] setzt jeweils die Ähnlichkeit einer Datenmenge zu ihrem eigenen Modell ins Verhältnis zu der Ähnlichkeit derselben Datenmenge zu einem zweiten Modell:

[65] Sie wird *Dreiecksungleichung* genannt.

[66] Es ließen sich zwar auch die Modellparameter, also Erwartungswerte und (Ko-)Varianzen, direkt miteinander vergleichen. Die so entstehenden Resultate würden jedoch keiner Anschauung mehr genügen und abstrahierten sehr stark von der eigentlichen Problemstellung. Im Gegensatz dazu stellt der Likelihood-Wert als Ergebnis des Tests von Daten und Modell anschaulich die Wahrscheinlichkeit dar, mit der diese Daten von jenem Modell erzeugt wurden. In diesem Kontext ist "Ähnlichkeit" viel eher fassbar.

[67] Aus dem Englischen für *Generalized Likelihood Ratio*.

[68] Aus dem Englischen für *Cross Likelihood Ratio*. Der Name *Cross Entropy* ist ebenfalls gebräuchlich.

$$d_{CLR}(\lambda_x, \lambda_y) = \frac{L(X|\lambda_x)}{L(X|\lambda_y)} + \frac{L(Y|\lambda_y)}{L(Y|\lambda_x)} \tag{2.27}$$

Hier wird also ein aufwendiges Training zugunsten mehrerer, günstigerer Tests vermieden, da gar keine neuen Modelle erstellt werden müssen. Anwendung findet dieses Distanzmaß beispielsweise in [MEIG 02] oder [NISH 03a].

Abbruch-Bedingung

Oben wurde vereinfachend gesagt, dass *"... an geeigneter Stelle in diesem Baum ein Schnitt durchgeführt werden ..."* muss. Die geeignete Stelle ist dabei jedoch nur schwer automatisch zu bestimmen. Solomonoff et al. führten in [SOLO 98] deshalb einen Schätzer für die Größe der *Cluster-Reinheit* ein. Anhand dessen sollte am Ende die Partition mit dem höchsten Reinheitswert gewählt werden. Dieser Ansatz wurde in [AJME 02] nochmals erweitert.

Aufgrund der hohen Komplexität hierarchischer Clusteringverfahren wäre jedoch ein Kriterium wünschenswert, welches schon während des Verschmelzungsprozesses angibt, dass an der aktuellen Stelle terminiert werden darf. So ließe sich überflüssige Arbeit vermeiden. [JIN 97] und [LIU 03] verwenden *Cluster-Interne Streuung* als globales Kriterium, um diese Funktionalität umzusetzen. Da dieses Maß aber zu vielen kleinen Clustern, im Extremfall also zu einem eigenen Cluster für jedes Segment, tendiert, wird es jeweils noch gewichtet mit einem Straffaktor für die Anzahl an Clustern:

$$W = \sum_{c=1}^{C} N_c \cdot \Sigma_c \cdot \sqrt{C} \tag{2.28}$$

C ist hierbei die Anzahl Cluster, N_c ist die Anzahl an Merkmalsvektoren, die in Cluster c zusammengefasst sind. Σ_c ist die Kovarianzmatrix dieses Clusters. Das eigentliche Kriterium ist dann am Ende $|W|$, also die Determinante von W.

Zuletzt sei noch auf das *Bayes'sche Informations-Kriterium* (BIC[69]) als Abbruchbedingung hingewiesen. Ursprünglich stellt es ein Modell-Auswahlverfahren dar, anhand dessen dasjenige statistische Modell gefunden werden kann, welches im *Maximum Likelihood*-Sinn am besten zu einer Menge Merkmalsvektoren passt [CHEN 98]. Es ist wie folgt definiert:

$$BIC(\lambda) = \log L(X|\lambda) - \alpha \cdot \frac{1}{2} \cdot len(\lambda) \cdot \log N_X \tag{2.29}$$

[69] Aus dem Englischen für *Bayesian Information Criterion*. In den Ingenieurwissenschaften ist auch die Bezeichnung *Minimum Description Length* (MDL) gebräuchlich.

λ ist hierbei das zu testende Modell mit $len(\lambda)$[70] Parametern[71], X die beschreibende Menge Merkmalsvektoren und N_X die Anzahl dieser Merkmalsvektoren. α ist ein ein Straf-Faktor, welcher frei und aufgabenabhängig wählbar ist und bestimmt, zu welchem Grad große Modelle bevorzugt oder benachteiligt werden sollen. Im Einklang mit der Theorie ist dabei jedoch nur der Wert $\alpha = 1$.

Um diese Formel jetzt auf die Anwendung als Abbruchbedingung eines Clustering-Prozesses umzumünzen, sind einige Umstellungen nötig, die in Anhang A.2 genauer ausgeführt sind. Unter den dort gemachten Annahmen ergibt sich folgende Formel, um den *Hypothesentest*[72] durchzuführen:

$$\Delta BIC(\lambda_x, \lambda_y) = N_{X\cup Y} \log L(X \cup Y|\lambda_{X\cup Y}) - N_X \log L(X|\lambda_X) - N_Y \log L(Y|\lambda_Y) - \alpha \cdot P \tag{2.30}$$

Dabei ist P der Ausdruck für die Komplexität der Modelle. Das Interessante ist, dass der letzte (Straf-)Term komplett wegfällt, wenn gilt, dass $len(\lambda_{X\cup Y}) = len(\lambda_X) + len(\lambda_Y)$. In diesem Fall ist das Verfahren komplett schwellwertfrei, bedarf also keinerlei Anpassung mehr an die genaue Aufgabenstellung oder die vorliegenden Daten. In der Form ohne diesen letzten Term fällt dann auch auf, dass die Formel des ΔBIC bis auf den nicht-negativen Vorfaktor identisch ist mit derjenigen des *Generalisierten Likelihood-Verhältnisses*: $\Delta BIC = -d_{GLR}$. Ersteres lässt sich so also nicht nur als Verfahren zur Modellselektion und als Abbruchbedingung, sondern auch direkt als Distanzmaß einsetzen.

Chen et al. benutzten das BIC in [CHEN 98] zur Sprechersegmentierung. [TRIT 99] entwickelte das dortige Verfahren zur Tauglichkeit für kurze Sprachsegmente weiter und gab noch vertiefende Erklärungen zur grundlegenden BIC-Theorie. Als schwellwertfreies Stopp-Kriterium wird es schließlich in [AJME 03] eingesetzt, während es in [NISH 03a] seinen ursprüngliche Zweck erfüllt, indem es zur Auswahl eines geeigneten Modells die Entscheidung zwischen komplexem GMM und einfachem VQ-Modell trifft.

[70] *len()* als Funktionsname ist hierbei dem englischen Wort "length" entlehnt und soll somit an die Größe des Modells erinnern.

[71] Die Anzahl der Parameter sagt etwas aus über die Größe und Komplexität des Modells.

[72] Im Grunde handelt es sich bei der Entscheidung, ob der nächste Verschmelzungsschritt vollzogen werden soll oder nicht, um einen Problemfall, der mittels statistischer Tests zur Ablehnung einer Hypothese $H_0 : X \sim \lambda_x; Y \sim \lambda_y$ angegangen werden kann. Die eigentlich zu bestätigende Aussage, dass beide Datensätze einer gemeinsamen Quelle entstammen, steckt dabei in der Gegenhypothese $H_1 : X \cup Y \sim \lambda_{x\cup y}$. Kann H_0 nicht abgelehnt werden, muss weiterhin von unterschiedlichen Sprechern ausgegangen werden.

Kapitel 3

Konzept

3.1 Das Grundgerüst

Sprecherklassifikation ist eine komplexe Aufgabe, soviel ist bislang sicherlich deutlich geworden. Und wie vielleicht der Forschungsüberblick im Kapitel zuvor gezeigt hat, beinhaltet sie viele notwendige Zwischenschritte, die an sich schon eine längere Untersuchung rechtfertigen würden. Zu Beginn des Programmentwurfs stand deshalb die Suche nach einem geeigneten Rahmenwerk, möglichst im Quelltext verfügbar, welches die grundlegenden Techniken bereits implementiert. Es sollte die Sicherheit geben, sich nicht zwingend mit jedem einzelnen Vorverarbeitungsschritt auseinandersetzen zu müssen, jedoch auch flexibel sein, um bei Bedarf angepasst werden zu können.

Nach einigem Suchen fiel die Wahl auf die C++ Klassenbibliothek *SV_Lib*[1] von Dr. Jialong He [HE 99b]. Sie bietet eine klassenorientierte, sehr offene Struktur an, in welcher diverse Routinen von der Merkmalsextraktion bis zur Sprecherextraktion bereits implementiert sind. Diese lassen sich recht einfach in eigene Anwendungen einbinden. Allerdings war das Paket anfangs nicht quelloffen verfügbar, was sich jedoch sehr bald direkt mit dem Autor klären ließ [HE 03].

Dass die hier entwickelte Software stark auf dieses Grundgerüst aufbaut, zeigt sich schon an der Namensgebung: *SC_Lib*[2] nennt sich die neu erschaffene Bibliothek. Sie erweitert He's Code um die im vorigen Kapitel als erforderlich dargestellten Verfahren zur Sprecherklassifikation. Zu Test- und Demonstrationszwecken wurde des weiteren ein Rahmenprogramm namens *SCiVo*[3] erstellt. Es nimmt die Dienste der SC_Lib in Anspruch und leistet mit ihrer Hilfe die in der Aufgabenstellung geforderte Funktionalität. Wenn im Verlauf dieser Arbeit also von der "entwickelten Software" oder Vergleichbarem gesprochen wird, ist meist die Funktionalität der Klassenbibliothek SC_Lib gemeint. Erzielte Ergebnisse dagegen basieren immer auf Tests der Bibliothek durch SCiVo.

[1] Aus dem Englischen für *Speaker Verification Library*, zu deutsch "Sprecher-Verifikations-Bibliothek". Weitere Informationen zu dieser Bibliothek finden sich im nächsten Kapitel.

[2] *SC* als Akronym für englisch *Speaker Classification*, zu deutsch "Sprecher-Klassifikation".

[3] *SCiVo* steht für *Speaker Classification in Videos*, *Sprecherklassifikation in Videos*.

Zur Eingrenzung des zu bewältigenden Stoffumfangs trägt auch die folgende Entwurfsentscheidung bei: Die als Vorverarbeitungsschritt notwendige Segmentierung des Audiostroms in markierte Sprach- und Nichtsprachbereiche wird als Eingabe für das Programm vorausgesetzt. Zwar existieren mit [LU 02b] hervorragende Ansätze zur Bewältigung dieses Problems, doch gehören diese nicht mehr zum Kern der gestellten Aufgabe. Wichtig ist allein die Feststellung, dass es entsprechende Verfahren gäbe, die entwickelte Software also mit einigem Recht als völlig automatisches System bezeichnet werden darf[4]. Gleiches gilt für die Aufteilung des Audiostroms anhand der Szenen des Videos, dem er entliehen ist. Auch diese Information wird im Programm benötigt und muss zu Anfang per Hand mitgegeben werden, doch existieren sogar innerhalb von Mediana bereits Verfahren, welche dieses Problem in Software lösen können.

Darüber hinaus existieren noch einige weitere Vorgaben, die den Aufbau der Software entscheidend prägen: Entwickelt in Zusammenarbeit mit der *Arbeitsgruppe Verteilte Systeme* der Phillips-Universität Marburg, muss sie in den dort vorgegebenen Rahmen passen. Dies erfordert zum einen obigen Bibliothekscharakter, um in das Gesamtprojekt integrierbar zu sein. Zum anderen ist auf Portabilität auf unterschiedliche Betriebssysteme und gute Skalierbarkeit auch bei der Ausführung in einem Rechner-*Grid*[5] zu achten.

3.2 Klassen-Einteilung

Die Klassenhierarchie der SC_Lib ist teilweise schon determiniert durch die Struktur von Jialong He's SV_Lib, da zum Teil deren Basisklassen weiter genutzt werden. Im Folgenden Abschnitt soll der Aufbau und das Zusammenwirken der einzelnen Komponenten etwas näher beleuchtet werden. Dabei geht es nicht in erster Linie um eine vollständige Darstellung sämtlicher Eigenschaften und Methoden der einzelnen Klassen. Vielmehr soll die zu Grunde liegende Idee sichtbar werden, um eine Hilfestellung zum Verständnis der später erläuterten Algorithmik zu bieten[6].

Abbildung 3.1 stellt deshalb und aus Gründen der besseren Übersichtlichkeit nur die reinen Klassen und ihre Beziehungen untereinander dar, nicht jedoch ihren internen Aufbau mit seinen Methoden und Attributen. Des Weiteren glänzt die Klasse `SV\_Data` hier durch Abwesenheit: Sie ist *der* Container für Daten wie z.B. Merkmalsvektoren und an

[4]Tatsächlich beschäftigt sich bereits ein weiterer Mitarbeiter des Projekts Medienumbrüche mit dieser Aufgabe, so dass entsprechende Programmroutinen in Kürze im System verfügbar sein werden. Für die Dauer dieser Diplomarbeit war jedoch noch Handarbeit zur Erstellung der Segmentierungslisten notwendig.

[5]Unter einem *Grid* versteht man in diesem Zusammenhang die Kopplung der Rechenleistung heterogener Rechnerstrukturen über ein Netzwerk.

[6]Eine reine API-Beschreibung der SC_Lib findet sich in elektronischer Form auf dem beiliegenden Datenträger wie in Anhang C beschrieben. Sie schließt die hier bestehende Lücke, indem alle öffentlichen Attribute und Methoden der Klassen aufgeführt und kurz erläutert werden.

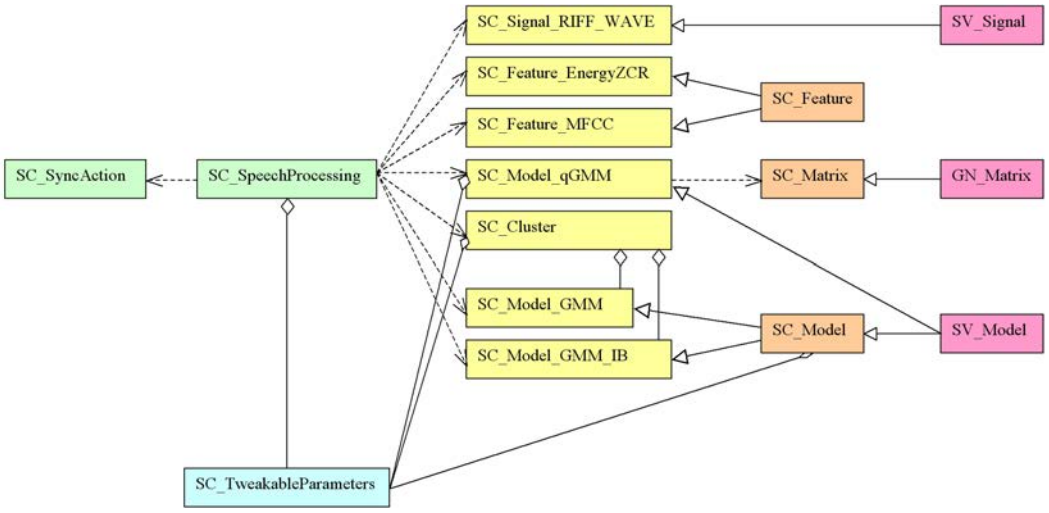

Abbildung 3.1: Klassendiagramm der SC_Lib. Die rötlich eingefärbten Kästchen rechts außen stellen Basisklassen der SV_Lib dar.

praktisch jedem Datenaustausch beteiligt, wird also auch in so gut wie jeder Klasse benötigt.

SC_SyncAction

Die Klasse `SC_SyncAction` steht in der Verantwortung, alle für einen Sprecherklassifikations-Lauf notwendigen globalen Daten bereitzuhalten und Operationen auf ihnen anzubieten. Sie synchronisiert also alle auszuführenden Aktionen. Im einzelnen bedeutet dies, dass sie beispielsweise zwischen den verschiedenen möglichen Zeiteinheiten im System umrechnet: Denn einerseits wäre da das den Ton bereitstellende Video, welches die SC_Lib zwar nie zu Gesicht bekommt, dessen Framerate[7] jedoch die Einheit vorgibt, in welcher Eingaben erfolgen und Ergebnisse präsentiert werden[8]. Intern wird jedoch andererseits in Audiorahmen gezählt[9], welche der Benutzer durch Angabe von Rahmenlänge und Überlappung spezifiziert hat. Und schließlich gibt es noch die Ebene der einzelnen Signalabtastwerte in der Audiodatei, auf welche beim Schreiben und Lesen derselben zurückgegriffen werden muss.

Darüber hinaus hält die Klasse `SC_SyncAction` die Parameter des zu verarbeitenden Audiosignals (nicht das Signal selber) und führt eine Liste, `frameList` genannt, in der für jeden einzelnen Audiorahmen des Signals die während der Verarbeitung ermittelten

[7] Mit *Framerate* wird auf neudeutsch die Anzahl Rahmen pro Zeiteinheit bezeichnet.

[8] Als Eingabe dienen dem System wie gesagt Audiosegmentierungs- und Szenenlisten. Zumindest die Zeitpunkte der Szenenwechsel sind dabei maximal videorahmengenau angegeben, da sie optisch analysiert werden. Deshalb macht es Sinn, auch alle weiteren Ein- und Ausgaben mit genau dieser Genauigkeit zu verlangen bzw. auszugeben.

[9] Intern spielt das Originalvideo keine Rolle, hier geht es nur um Audiodaten. Da die Analysemethoden des Signalverarbeitungstraktes auf Rahmenbasis arbeiten, ist es sinnvoll, diese Zählweise beizubehalten.

Eigenschaften festgehalten werden. Das sind beispielsweise die schon durch die Eingabe bestimmten Zeitpunkte der Szenenwechsel oder die Information, ob der Rahmen Sprache oder Sonstiges enthält. Im Verlauf des Klassifikationsprozesses stoßen immer mehr Informationen hinzu, bis am Ende für jeden Rahmen eine eindeutige Sprecher-ID feststeht. Auf diese Liste kann nun von außerhalb über spezialisierte Methoden zugegriffen werden, die beispielsweise Start- und Endpunkt des nächsten Sprachsegments zurückliefern oder aber einem bestimmten Segment eine neue Eigenschaft (z.B. "nicht-stimmhafte Sprache") zuweisen.

SC_SpeechProcessing

`SC_SpeechProcessing` kapselt alle Algorithmen, die zur Verarbeitung des Audiosignals notwendig sind. Dies fängt an mit dem Laden des Signals und der Extraktion von Merkmalen, geht über die Vorverarbeitungsschritte zur Ausfilterung von Stille und nichtstimmhafter Sprache sowie der Sprecherwechsel-Erkennung, und endet schließlich bei Methoden zur Sprechermodellierung. Alle diese Verfahren arbeiten dabei szenenorientiert, wobei eine Szene hier im Prinzip jeder durch Anfangs- und Endpunkt gekennzeichnete Abschnitt des Audiosignals sein kann. Wichtig ist allein, dass einzelne Abschnitte *unabhängig voneinander* bearbeitet werden können, was zu Parallelisierung und damit zu der gewünschten Skalierbarkeit für Grid-Computing führt. Doch natürlich macht die szenenweise Analyse auch darüber hinaus Sinn: Innerhalb einer Szene kann mit einiger Gewissheit ein relativ gleichförmiger akustischer Hintergrund angenommen werden. So können für alle Verfahren, die der Sprachanreicherung vorangehen, akzeptable Arbeitsbedingungen erreicht werden. Des Weiteren verändert sich natürlich auch das Laufzeitverhalten und die Ressourcenausnutzung zum Besseren, wenn immer nur einige Sekunden anstatt Stunden an Audiodaten im Speicher verarbeitet werden müssen.

Zur Erfüllung ihrer Aufgaben verwenden die `SC_SpeechProcessing`-Methoden die Dienste anderer Klassen, erstellen also beispielsweise Objekte, die Merkmalsvektoren bereithalten oder Sprecherrepräsentationen bereitstellen. Diese Daten werden jedoch nicht klassenintern gespeichert. Vielmehr obliegt die Verwaltung der berechneten Daten dem aufrufenden Programm, im Falle dieser Arbeit also dem Testprogramm SCiVo. Dieses nimmt die Ergebnisse eines Algorithmus entgegen und übergibt sie entsprechend bei Bedarf der nächsten Stufe zur Weiterverarbeitung.

Der letzte in `SC_SpeechProcessing` enthaltene Algorithmus aus dem Portfolio der Sprecherklassifikation bewirkt das Clustering. Er wird hier gesondert von seinen Zuarbeitern genannt, da er nicht szenenbasiert arbeitet, sondern erst nach Vorhandensein aller Analyseergebnisse global tätig wird[10]. Eine Parallelisierung der anfallenden Arbeit lässt sich jedoch auch hier erzielen und wird in Abschnitt 3.4.4 genauer erläutert.

[10] Eine Quizfrage zwischendurch: Auf welches Clusteringverfahren lässt das wohl schließen?

Daneben existieren natürlich noch eine Reihe weiterer Methoden, welche hauptsächlich dem Test (Generierung synthetischer Testdaten), der Fehlersuche und Überwachung (Ausgabe von Zwischenergebnissen auch in akustischer Form) oder der Zuarbeit bereits genannter Verfahren dienen. Detailliertere Angaben hierzu finden sich – das gilt übrigens auch für alle anderen den Programmcode betreffenden Fragen – in der schon erwähnten API-Beschreibung oder im Quelltext selber: Er ist recht ausführlich, allerdings der Internationalität halber in englischer Sprache, kommentiert.

SC_TweakableParameters

Bisher wurden als notwendige Benutzer-Eingaben für das Sprecherklassifikations-System nur die obligatorische Audiodatei sowie die Angaben zu Szeneneinteilung, Audiosegmentierung und Rahmengröße genannt. Tatsächlich gibt es jedoch noch eine Reihe von *Hyperparametern*, die das Gesamtverhalten beeinflussen. Diese sind einerseits so trivial wie beispielsweise der Pfad für die Ausgabe von Debug-Informationen, andererseits aber auch wichtige Steuerparameter für das Verhalten einiger Subsysteme. Gemein ist ihnen, dass es für sie alle sinnvoll ist, vom Benutzer des Programms den momentanen Bedürfnissen angepasst werden zu können. Durch das Algorithmendesign wurde zwar wo es ging versucht, aufgabenspezifische Schwellwerte zu vermeiden – teilweise gelang dies auch –, doch blieben immer noch genügend solcher Werte bestehen, um einige Gedanken in deren adäquate Verwaltung zu investieren.

`SC_TweakableParameters` kapselt nun *alle* veränderbaren Parameter der gesamten SC_Lib, um sie zentral verwalten zu können. Dies bringt eine enorme Vereinfachung bei der Arbeit am Code mit sich, denn so kann es nicht passieren, dass eine Veränderliche übersehen oder vergessen wird. Es fällt nun auch enorm leicht, einen genauen Überblick über die momentane Parametrisierung zu erhalten, ohne erst sämtliche Codefragmente nach literalen Werten absuchen zu müssen. Dabei hat das Verfahren natürlich den Nachteil, softwaretechnisch unschön zu sein: Eine Klasse mit unglaublich vielen Eigenschaften, die in der Praxis als Singleton[11] existiert und in den verschiedensten Klassen referenziert[12] ist, weckt Erinnerungen an globale Variablen. Doch der Zweck heiligt hier ausnahmsweise die Mittel.

SC_Model*

Die Klassen, denen `SC_Model` vorangestellt ist, enthalten konkrete Implementierungen

[11] Unter einem *Singleton* versteht die Softwaretechnik ein Objekt einer Klasse, welches tatsächlich nur ein einziges mal instanziiert sein darf. Der Begriff ist in diesem Zusammenhang leicht irreführend, da `SC_TweakableParameters` diesen Zwang nicht ausübt. Faktisch existiert jedoch nur eine Instanz.

[12] Sämtliche Klassen, die auf verschiedene Parameter angewiesen sind, halten einen Verweis auf eine Instanz der `SC_TweakableParameters`, der sie ihre Einstellungen entnehmen können. Daneben erwarten noch einige Algorithmen diese Instanz als Übergabewert, da sie temporär auf ihr Wissen angewiesen sind.

verschiedener Sprecherrepräsentations-Verfahren. Sie basieren dabei alle auf der SV_Lib-Basisklasse `SV_Model`. `SC_Model_GMM` und `SC_Model_GMM_IB` haben dabei so viel gemein, dass sie den gemeinsamen Vorfahren `SC_Model` bekamen, während `SC_Model_qGMM` vom Aufbau und der verfolgten Zielsetzung her anders angelegt ist und deshalb direkt von `SV_Model` abgeleitet ist.

Wie den anderen Modellierungsansätzen und beispielsweise auch den `SV_Data`-Objekten der SV_Lib ist es auch diesen Repräsentationen möglich, eine verkettete Liste zu bilden, was sich in der Implementierung als sehr sinnvoll für das Formen unzusammenhängender, sprecherhomogener Segmente herausstellen wird.

Ein Wort noch zu der Klasse `SC_Model_GMM`: Sie scheint auf den ersten Blick überflüssig zu sein, da ein GMM schon in `SV_Model_GMM` implementiert ist. Noch mehr zur Verwunderung mag beitragen, dass beide auf derselben Quelle [REYN 95] basieren, also im Wesentlichen gleiche Ziele und Ansätze haben. Tatsächlich war die erneute Implementierung zunächst nur ein Test: Wurden die Grundlagen richtig verstanden, funktioniert die eigene Implementierung, stimmen die Ergebnisse mit denen des SV-Modells überein? Erst im Nachhinein zeigte sich, dass diese Mühe keineswegs umsonst war. Das `SC_Model_GMM` ist in seiner Verwendung etwas flexibler und kann daher besser in das entwickelte Rahmenwerk zur Sprecherklassifikation integriert werden.

SC_Cluster

Die Objekte der Klasse `SC_Cluster` haben die Aufgabe, alle Abschnitte des Audiosignals zu repräsentieren, die von dem Klassifikationsverfahren einem einzigen bestimmten Sprecher zugewiesen wurden. Zu diesem Zweck halten sie eine Liste, genannt `segmentList`, vor, in welcher die einzelnen Sprachabschnitte mit Anfangs- und Endpunkt festgehalten sind. Außerdem besitzen sie Verweise auf die Merkmalsvektoren und Sprecher-/Hintergrundmodelle dieser Segmente. Diese sind jeweils als verkettete Liste organisiert, so dass ein Zugriff auf das i-te Element dieser Listen die Merkmalsvektoren oder Repräsentationen hervorbringt, welche mit dem Segment an i-ter Stelle der `segmentList` korrespondieren.

Neben einer Sprecheridentifikations-Nummer stellt die Klasse `SC_Cluster` noch verschiedene Methoden im Zusammenhang mit dem Clustering-Prozess zur Verfügung: So lassen sich zwei Cluster zu einem großen verschmelzen, oder die Ähnlichkeit zwischen einem Clusterpaar kann mittels wählbarer Distanzmaße ermittelt werden. Auch ein Maß zur Bewertung eines Verschmelzungsvorgangs[13] ist enthalten. Schlussendlich existieren noch Methoden, um alle Daten, auf welche ein Cluster-Objekt mittels verketteter Listen verweist, zu zerstören und die in Anspruch genommenen Ressourcen explizit freizugeben.

[13] Also ein Abbruch-Kriterium für den Clustering-Vorgang.

SC_Feature*

Die Klassen mit dem Präfix `SC_Feature` dienen der Implementierung von Verfahren zur Merkmalsextraktion, nicht zur Speicherung der extrahierten Merkmale. Dies übernimmt traditionellerweise[14] die Klasse `SV_Data`.

Die Basisklasse `SC_Feature` ist beinahe eine exakte Kopie ihres Vorbildes, der Klasse `SV_Feature`. Diese bestand auf dem Konzept, dass jeder Merkmalsextraktions-Algorithmus eine eigene Kopie des Signals vorhalten musste, um auf jeden Fall ein durch Vorverarbeitung unangetastetes Signal zur Bearbeitung zu erhalten. Aus Gründen der Ressourcenschonung[15] wurde dieses Konzept geändert, so dass es nun möglich ist, auf ein zentral vorliegendes Signal zuzugreifen. Dies machte eine Umstellung sämtlicher Methoden nötig, so dass eine einfache Ableitung der SC-Klassen von ihren SV-Pendants nicht sehr sinnvoll war.

Für die Klasse `SC_Feature_MFCC` gilt Ähnliches wie für ihre Basisklasse: Sie ist im Grunde eine Kopie der `SV_Feature_MFCC`, hat jedoch `SC_Feature` als Elternteil, um von den Vorteilen der zentralen Signalvorhaltung zu profitieren. `SC_Feature_EnergyZCR` ist hingegen eigenständig entwickelt und stellt Methoden bereit, um die Merkmale Energie und Nulldurchlaufsrate zu ermitteln.

SC_Matrix

`SC_Matrix` erweitert die SV_Lib-Basisklasse `GN_Matrix` um einige weitere, weniger komplizierte, jedoch nicht weniger benötigte Funktionen. Dazu gehören die Multiplikation, Addition und Subtraktion von Matrizen mit Vektoren oder anderen Matrizen, die Implementierung des in [CAMP 97] entwickelten Distanzmaßes zur Ermittlung der Ähnlichkeit zweier Matrizen "Divergence Shape", oder die Bestimmung der Spur[16] einer Matrix.

Daneben existieren hier noch Kopien der Funktionen zur Bestimmung vom Erwartungswert und Kovarianz einer Menge von Merkmalsvektoren, wie sie schon in der SV_Lib implementiert sind. Einziger Unterschied ist hier, dass beliebige Merkmalstypen[17] als Eingabe dienen können.

SC_Signal_RIFF_WAVE

[14] Es handelt sich hierbei um eine in der SV_Lib eingeführte Tradition, die so tief in ihrem Kontrollfluss verankert ist, dass es kaum Sinn macht, sie zu ändern. Darüber hinaus spricht eigentlich auch nichts dagegen, sie beizubehalten.

[15] Wer hat schon gerne zwei mal 180 Minuten Audiomaterial im Hauptspeicher?

[16] Die *Spur* einer quadratischen Matrix ist die Summe der Elemente auf der Hauptdiagonalen.

[17] Die Beliebigkeit gilt hier im Hinblick auf ihren Datentyp.

`SC_Signal_RIFF_WAVE` entstammt der SV_Lib-Basisklasse `SV_Signal`, um einen Spezialfall zu implementieren: Das Laden von Audiosignalen im Microsoft-Windows-spezifischen RIFF-WAV-Format[18]. Neu ist hier die Funktion, ein bereits extrahiertes Signal nach der Verarbeitung[19] wieder in eine Datei zu schreiben. Dies dient vor allem der akustischen Überprüfung der im System erzielten Ergebnisse. Außerdem besteht nun die Möglichkeit, das Signal zentral in dieser Klasse vorzuhalten, anstatt es zu kopieren und separat in jedem Objekt der Merkmalsextraktion zu speichern.

SC_Aux

`SC_Aux` schließlich ist gar keine Klasse. Vielmehr ist es ein Modul, eine getrennte Quelltexteinheit, welche häufig und überall benötigte Funktionen bibliotheksglobal zur Verfügung stellt. Hierzu gehören solche zur Sortierung von Feldern, zum Umgang mit verketteten Listen oder zur Debug-Ausgabe von Objekten in Textdateien.

3.3 Verwendete Verfahren

Der Forschungsüberblick in Abschnitt 2.4 hat für jeden Teilschritt der zu bewältigenden Aufgabe mehrere alternative Vorgehensweisen vorgeschlagen, die allesamt ihre spezifischen Vor- und Nachteile haben. Teilweise wurde auch schon auf den Grad ihrer Eignung für das hier im Speziellen behandelte Problem hingewiesen. Der folgende Abschnitt möchte nun die endgültige "Wahl der Waffen" darstellen und motivieren. Eine detaillierte Darstellung der einzelnen Algorithmen folgt dann für die Fälle, wo dies notwendig erscheint, in Abschnitt 3.4.

Merkmale

Anfangen sollte man sicherlich mit den verwendeten Merkmalen. Sie sind die Grundlage des Systems und sollten als solche dessen Zielsetzung schon nach Möglichkeit unterstützen. Daher sind die an sie gestellten Anforderungen folgende: Hohe Spezifität für sprecherdiskriminierende Eigenschaften, geringe Anfälligkeit gegen Nebengeräusche und sichere Ermittlung über kurze Zeiträume. Aufgrund der ersten beiden Punkte ist es quasi unmöglich, eine andere Wahl als *MFC-Koeffizienten* zu treffen. Sie sind bis heute das Maß, welches von noch keinem anderen Merkmal übertroffen werden konnte. Nun stellt sich noch die Frage nach der Methode, anhand derer die spektrale Darstellung des Signals ermittlet wird. Wavelet-basierte Verfahren sind stark im Fortschritt begriffen und ermöglichen robustere Annäherungen des tatsächlichen Spektrums über kürzere Zeiträume

[18] Das *WAVE-Dateiformat* ist Teil des von Microsoft spezifizierten "Resource Interchange File Formats" (RIFF) und erlaubt die Speicherung von Audiosignalen [MICR 91]

[19] Beispielsweise nach der Entfernung aller Messwerte, die keine stimmhafte Sprache repräsentieren.

hinweg. FFT-basierte Methoden sind hingegen weit verbreitet, ihre Stärken und Schwächen sind bekannt. Schlussendlich fiel die Wahl auf die *Fouriertransformation*, da so die Vergleichbarkeit gegenüber anderen Arbeiten auf diesem Gebiet besser gegeben ist und so die Neuprogrammierung der Merkmalsextraktions-Algorithmen der SV_Lib vermieden werden konnte.

Als unterstützende Merkmale werden zusätzlich noch *Rahmen-Energie* und *Nulldurchlaufsrate* gemessen. Sie werden einzig von den Filterungsschritten zur Trennung stimmhafter Sprache von Anderem benötigt.

Filterung

Eigentlich müsste das zu entwickelnde System auf den Vorverarbeitungsschritt der Filterung verzichten können. Denn schließlich verlangt es als Eingabe ja schon eine Vorsegmentierung des Audiomaterials in Sprache und Sonstiges. Die Rechtfertigung der Notwendigkeit dieser Stufe ergibt sich jedoch aus der gewünschten Genauigkeit des Systems. Kurze Pausen oder Atemholen innerhalb eines als Sprache gekennzeichneten Segments kann die Sprechermodelle ziemlich durcheinander bringen. Bei der Anlegung der für die Experimente benötigten Segmentierungslisten *per Hand* konnte jedoch nicht audiorahmengenau geschnitten werden[20], so dass ein maschineller Durchlauf diesen groben Resultaten noch einmal den nötigen Feinschliff verpasst.

Es existieren nicht sehr viele Alternativen, diese Aufgabe zu bewältigen. Meistens wird sie durch Anwendung eines Schwellwerts auf die Energiekontur gelöst. Allerdings sollte das System nicht auf die Feinabstimmung eines Schwellwerts für jeden neuen akustischen Hintergrund vertrauen müssen, sondern sich vielmehr selber anpassen können. Die Wahl fiel deshalb auf das in [LI 03] vorgestellte Verfahren der *adaptiven Stille-Erkennung*. Es erfüllt die Anforderungen und wird überdies in der entsprechenden Veröffentlichung so genau beschrieben, dass eine Implementierung "vom Blatt weg" möglich erscheint. Die späteren (hörbaren) Ergebnisse geben der Einschätzung recht, dass für diese Aufgabe ein Rückgriff auf höherentwickelte Methoden und Merkmale wie MFCC's nicht nötig ist.

Sprecherwechsel-Erkennung

Für den Bereich der Sprecherwechsel-Erkennung ist es vor allem wichtig, dass das Verwendung findende Verfahren auch mit äußerst kurzen Sprachäußerungen umzugehen weiß. In der Literatur ist jedoch für die wenigsten Ansätze eine robuste Funktionsweise für Segmente unter drei Sekunden Länge angegeben[21]. Nirgends ist jedoch ein System zu finden,

[20] Die Präzision der so erstellten Grundwahrheit liegt ungefähr im Bereich eines Videorahmens, beträgt also normalerweise 25 Millisekunden.

[21] Die in solch kurzen Äußerungen enthaltenen Informationen sind sehr sprach- und kaum sprecherspezifisch. Um sich dies zu verdeutlichen, sollte man einmal versuchen, eine Stimme anhand der Begrüßung

welches Segmente unter zwei Sekunden Länge verarbeitet.

Als zweites Kriterium gilt für die Sprecherwechsel-Erkennung wie auch für alle weiteren Teilbereiche des Systems die möglichst weitgehende Unabhängigkeit von Schwellwerten. Dies gab den Ausschlag für die Verwendung des in [LU 02c] dargestellten Verfahrens, das allerdings noch weiter angepasst werden musste. Die Inanspruchnahme statistischer Methoden gab den Anlass zu vermuten, dass es bei den erwarteten stark variablen Daten bessere Ergebnisse zeigen könnte als ein rein metrik-basiertes System.

Sprecherrepräsentation und Sprachanreicherung

Die Methoden zur Sprecherrepräsentation und der Verringerung des Einflusses von Nebengeräuschen sind eng miteinander verzahnt: Soll ein modellgestütztes Schema zur Sprachanreicherung benutzt werden, ist es nicht sinnvoll, andersartige Sprechermodelle zu verwenden. Daher ergibt sich aus der Wahl des Ersteren implizit die Verwendung des Letzteren.

An modellbasierten Verfahren existiert nun hauptsächlich die Methode der Parallelen Modell-Kombination von Gales sowie der Ansatz von Rose et al.: GMM's mit Integriertem Hintergrund. PMC muss für den hier betrachteten Spezialfall leider ausscheiden, da die Voraussetzungen nicht gegeben sind: Die Methode benötigt sowohl reine Modelle des Hintergrund- als auch des Sprachprozesses. Dies kann hier leider nicht vorausgesetzt werden. Das GMM-IB-Verfahren hingegen passt zu den Anforderungen und Voraussetzungen und wurde in der Literatur mehrfach als Meilenstein der Entwicklung gelobt[22]. Darüber hinaus wurde es schon erfolgreich unterschiedlichen Einsatzgebieten angepasst, wobei jedoch niemals Tests an realen Daten veröffentlicht wurden. Die Wahl fiel deshalb auf das *GMM-IB* [ROSE 94] zur Sprachanreicherung, denn es passt zum Problem, lässt gute Leistungen erhoffen und bietet Raum zu einem eigenständigen Beitrag zur Forschung.

Somit steht auch die Wahl der Sprechermodellierung fest: Das GMM-IB ist ja direkt diese Repräsentation. Und für die Modellierung des integrierten Hintergrundmodells kommt ein Standard-*GMM* zum Einsatz [REYN 95]. Diese Wahl hat ihre Vor- und leider auch Nachteile: Zum einen ist es so möglich, die nach gründlicher Abwägung der Alternativen beste Methode zur Hintergrundkompensation einzusetzen. Andererseits ist von GMM's bekannt, dass sie eine recht große Menge an Trainingsdaten benötigen, um ihre exzellente Leistung zu erbringen. Reynolds et al. sprechen in ihrer oben genannten Publikation von 15 Sekunden, andere Forscher gehen auf bis zu fünf Sekunden herab. Unumstritten bleibt jedoch, dass für sehr kurze Segmente unter drei Sekunden Länge ein hybrider VQ-GMM-Ansatz wie in [NISH 03a] bessere Leistungen zeigen könnte.

"Hallo" am Telefon zu identifizieren. Dies wird den meisten Menschen selbst bei bekannten Stimmen oft schwer fallen.

[22] Vergleiche hierzu beispielsweise [LOGA 98].

Das Dilemma ist also, eine Wahl treffen zu müssen zwischen einerseits dem Umgang mit kurzen Segmenten unter Vernachlässigung des Einflusses von Nebengeräuschen, andererseits aber dem genau entgegengesetzten Fall: Hintergrundkompensation auf Kosten der Segmentkürze. Die Wahl fiel hier auf letzteres, denn Nebengeräusche werden in so gut wie jedem Fall auftreten. Die Segmentlänge hingegen ist zwar ein gewichtiges, doch nicht omnipräsentes Problem in Videos – es gibt auch lange Segmente, jedoch kaum solche ohne akustischen Hintergrund.

Ein kurzer Blick soll hier noch auf die nicht-modellbasierten Verfahren zur Sprachanreicherung geworfen werden: Viele davon werden in [REYN 95] untersucht, wobei die Autoren zu dem Schluss kommen, dass hier einzig die CMN-Methode sinnvoll ist. In Szenarien mit wechselndem akustischen Hintergrund wird allerdings von ihrer Verwendung abgeraten, da sie mehr Probleme schaffen als kompensieren würde[23].

Sprecherklassifikation

Zum Zweck der Klassifikation kommt ein *hierarchisches Clusteringsverfahren* zum Einsatz. Es zieht sich wie ein roter Faden durch die Literatur, dass solche das Mittel der Wahl zur unüberwachten Einteilung von Daten in Klassen sind. Alternativ hätte einer der neuen Online-Algorithmen verwendet werden können, doch es ist wichtig für die Erfüllung der gestellten Aufgabe, keine Erkennungsleistung zu verschenken, und Echtzeitverarbeitung ist nicht gefordert. Doch natürlich bedeutet diese Wahl so auch einen deutlich höheren Ressourcenverbrauch, was Lauf- und Rechenzeit des Algorithmus betrifft.

Die Wahl des Stopp-Kriteriums fiel auf das BIC, da ihm von vielen Seiten gute Leistungen zugesprochen werden und es überdies einen natürlichen, nicht aufgaben- und datenabhängigen Schwellwert zur Entscheidung bietet. Für die Distanzberechnung kann zwischen GLR und CLR gewählt werden. Beide werden in verwandten Publikationen häufig eingesetzt, und da eine Implementierung nicht sehr aufwendig und Leistungsunterschiede zwischen beiden Methoden im Voraus schlecht ausmachbar sind, sollen die nachfolgenden Experimente die endgültige Wahl treffen.

3.4 Ausgewählte Kapitel der Algorithmik

Die Auswahl der im Folgenden dargestellten Algorithmen wurde anhand ihrer Bedeutung für das Gesamtsystem und dem Anteil an Eigenleistung getroffen, welcher in ihre Entwicklung einfloss. Der SV_Lib entliehene Verfahren[24] fallen also genauso durch dieses Raster wie unbedeutendere Hilfsfunktionen[25]. Das beschränkt die Gruppe der Kandidaten im Wesentlichen auf jene Schlüsselverfahren, über die weiter oben schon auf anderer

[23] Vergleiche hierzu auch Abschnitt 2.4.5.

[24] Wie beispielsweise die Merkmalsextraktion.

[25] Was allerdings nicht heißt, dass ihre Funktionsweise durch weniger kreative Energie beflügelt wurde.

Ebene Manches gesagt wurde. Hier soll nun ihre Funktionsweise detailliert erläutert und der Einfluss eigener Ideen deutlich gemacht werden.

3.4.1 Reinigung der Sprachsegmente

An dieser Stelle muss also eine Feinabstimmung der als Sprache markierten Audioabschnitte erfolgen. Wie bereits gesagt, basiert dies auf der Beobachtung der Energiekontur, da Sprache für ihre im Vergleich zu Anderem hohe Energie bekannt ist. Natürlich trifft dies grundsätzlich auch auf Musik oder Toneffekte in Filmen zu, doch sollten für sich alleine stehende solche durch die Segmentierungslisten ja bereits grob unterschieden und als "Sonstiges" markiert worden sein. Es geht also nur noch um Feinabstimmung innerhalb der als Sprache markierten Audioanteile, weshalb die Annahme gerechtfertigt ist. Denn im direkten Vergleich ist Sprache am energiereichsten.

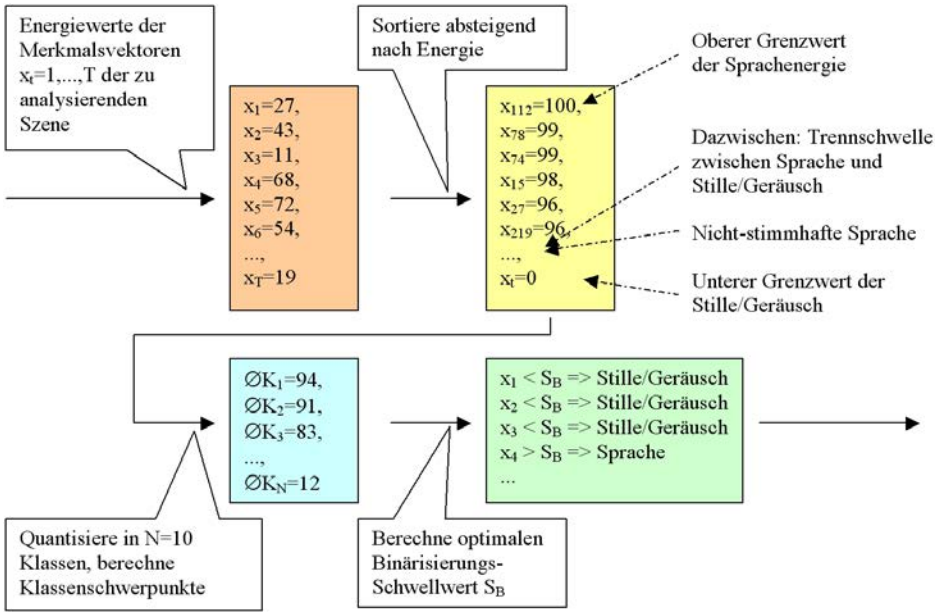

Abbildung 3.2: Der Ablauf des Algorithmus zur Trennung von Sprache und Hintergrund. Grundannahme ist, dass Sprachrahmen normalerweise höhere Energie besitzen als Nebengeräusche oder Stille.

Die Grundstruktur des Algorithmus [LI 03] wurde auch bereits in Abschnitt 2.4.2 besprochen und ist noch einmal bildlich in Abbildung 3.2 verdeutlicht. Neu ist hier allerdings die Methode der Schwellwertfindung: Das ursprünglich von Li et al. vorgeschlagene Verfahren hat sich in diesem Kontext als wirkungslos herausgestellt, die Ergebnisse schienen sehr willkürlich. Doch glücklicherweise fand sich Ersatz:

Die Schwellwertmethode von Otsu [OTSU 79] wurde ursprünglich entwickelt, um anhand von Histogrammen von Grauwertbildern eine optimale Schwelle zur Binärisierung

des Bildes zu ermitteln. Sie verwendet Verfahren der *Diskriminations-Analyse*[26], um dieses Ziel zu erreichen. Auf Basis des in [KILT 03] enthaltenen Beispielprogramms wurde dieses Verfahren auf das hiesige Problem angepasst, denn die Fragestellung ist - auf einem abstrakteren Niveau - quasi analog: Zur Quantisierung wird der Bereich zwischen der minimalen und maximalen Energie in N Bereiche unterteilt, und ein Histogramm für diese Bereiche wird erstellt. Statt einem Grauwert repräsentiert nun jeder Histogramm-Balken jedoch die durchschnittliche Energie der Rahmen, die ihm zugeordnet sind. Die N Klassen sind also äquidistant zwischen Minimum und Maximum verteilt, und der Schwellwert wird auf Basis der Anzahl der ihnen zugewiesenen Rahmen ermittelt. Das Ergebnis ist der Index n der Klasse, an welcher optimalerweise getrennt werden sollte, und durch deren Energiemittelwert ist der endgültige Schwellwert gefunden: Sprache oder nicht Sprache statt Schwarz oder Weiß.

Somit wurden alle Signalanteile, welche keine Sprache enthalten, entsprechend markiert. Doch wie soll weiter mit ihnen verfahren werden? Im System wird eine Unterscheidung zwischen Sprachpausen und tatsächlicher Stille/Geräuschen getroffen: Tritt innerhalb eines Sprachabschnittes plötzlich mindestens ein Rahmen auf, der nicht als Sprache gekennzeichnet ist, unterteilt dies das zuvor zusammenhängende Sprachsegment in zwei einzelne. Dies hat Folgen – beispielsweise für die spätere Sprecherwechsel-Erkennung. Alternativ kann ein von obigem Algorithmus aus einem Sprachsegment aussortierter Abschnitt bis zur Länge von $pauseSilenceThreshold$ Millisekunden auch als "Pause" markiert werden: Er behält so das Attribut "Sprache", wird jedoch nicht mehr zu den folgenden Sprachverarbeitungs-Schritten herangezogen. Beim Programmstart kann der Wert des Parameters $pauseSilenceThreshold$ vom Benutzer frei spezifiziert werden.

Experimente haben gezeigt, dass auf diese Weise ein sehr reines, zur Modellierung geeignetes Sprachsignal entsteht. Sogar nicht-stimmhafte Sprache wird aussortiert und als Stille/Geräusch markiert. Soll allerdings nachher ein Modell des reinen Hintergrundgeräuschs auch auf Basis dieser Rahmen erstellt werden, stören diese Stimmanteile dort sehr stark. Deshalb wird obiger Algorithmus zur Stilleerkennung nochmals in leichter Modifikation auf das Ergebnis seines ersten Durchlaufs angewandt:

Nicht-stimmhafte Sprache besitzt ein Energieniveau am untersten Rand der Sprache, weshalb sie wie oben beschrieben aussortiert wurde. Innerhalb des Ausschusses allerdings ist sie am oberen Rand, die echten Nebengeräusche/Stille wieder am unteren Rand zu finden. Ebenso sollte es sich der Theorie nach mit der Nulldurchlaufsrate verhalten. Das oben dargestellte Verfahren wird also nun zweifach angewandt, jeweils auf das Merkmal Energie und Nulldurchlaufsrate. Und unabhängig voneinander wird jeweils anhand des entstandenen Schwellwertes eine Trennung durchgeführt. Wahlweise kann dieses Verfahren iteriert werden, um mehr potenziell sprachgeprägte Signalanteile auszufiltern.

[26] *Diskriminations-Analyse* ist die Suche nach der Menge von Parametern, welche zwischen zwei oder mehr natürlich auftretenden Gruppen unterscheidet. Nach [KILT 03].

3.4.2 Erkennung von Sprecherwechseln

Der von Lu et al. entwickelte Algorithmus wird hier in modifizierter Form dargestellt: Die erste, unwesentliche Änderung ist, dass er hier nur auf den Daten einer Szene anstatt auf einem ganzen Video arbeitet. In [LU 02c] war dies der Fall, da das Verfahren nicht nur zur Wechselerkennung, sondern auch zur anschließenden Sprecherklassifikation verwendet wurde. Dies ist hier jedoch nicht nötig, so dass an der zuvor begonnenen szenenweisen Verarbeitung festgehalten werden kann.

Die zweite Anpassung ist weitreichender: Während im Original ein Drei-Sekunden-Fenster jeweils um eine halbe Sekunde versetzt über die Merkmalsvektoren geschoben wird, um jederzeit Wechsel detektieren zu können, soll hier jeweils ein ganzes zusammenhängendes Sprachsegment wie dort ein solches Fenster behandelt werden. Das hat den Nachteil, dass Sprecherwechsel nur zwischen getrennten Sprachsegmenten erkannt werden können[27], jedoch folgenden Vorteil: In Filmen kommt es häufiger vor, dass innerhalb von drei Sekunden mehr als ein Sprecherwechsel vorkommt, jedoch meist getrennt durch nichtsprachliche Signalanteile. Diese würden mit der Fenstertechnik übersehen werden, was später unreine Sprachsegmente zur Folge hätte, die unbedingt vermieden werden sollen. Experimente haben gezeigt, dass die Grundannahme von Wechseln nur innerhalb kurzer Pausen in circa 80 Prozent der Fälle haltbar ist [LIU 99], so dass der Nachteil des Verpassens der anderen tatsächlich nicht so schwer wiegt.

Schlimmer ist schon, dass so natürlich auch zu kurze Segmente in die Verarbeitung gelangen und eventuell stören würden. Sie schließen sich jedoch glücklicherweise selber durch folgendes Phänomen aus: Zur Errechnung des Abstands der Kovarianzmatrizen zweier benachbarter Segmente nach folgender Formel der Divergence-Shape-Distanz [CAMP 97] ist die Invertierung der Matrizen notwendig:

$$d(i,j) = Spur\left((C_i - C_j) \cdot ({C_i}^{-1} - {C_j}^{-1})\right) \tag{3.1}$$

War das Segment zu kurz, passiert es häufig, dass die korrespondierende Kovarianzmatrix singulär wird[28] und folglich nicht invertierbar ist. In diesem Fall wird das Segment als zu kurz markiert und im weiteren Verlauf der Analyse übergangen.

Der eigentliche Algorithmus funktioniert nun so wie schon in Abschnitt 2.4.3 beschrieben. Abbildung 3.3 veranschaulicht das zweistufige Prinzip nochmals grafisch: Zuerst werden die Kovarianzmatrizen der Segmente erstellt und anhand ihrer Ähnlichkeit und eines Schwellwertes[29] über potenzielle Wechsel entschieden. Tritt kein solcher auf, kommt es

[27] Hier wird nun auch der Sinn der oben gemachten Unterscheidung zwischen Sprachpause und wirklicher Segmentsunterbrechung sichtbar: Stellte jeder Nicht-Sprachrahmen innerhalb eines Abschnitts eine Segmentgrenze dar (an der ein Wechsel der Sprecheridentität vermutet werden muss), so sinkt die Segmentlänge rasch unter das erforderliche minimale Niveau, und keines der nachfolgenden Subsysteme kann mehr mit den Daten arbeiten.

[28] Eine Matrix heißt *singulär*, wenn ihre Determinante gleich null ist. In diesem Fall existiert auch keine Inverse der entsprechenden Matrix.

[29] Dessen Bestimmung kann analog zur Erstellung eines gleitenden Mittelwertes gesehen werden, nur

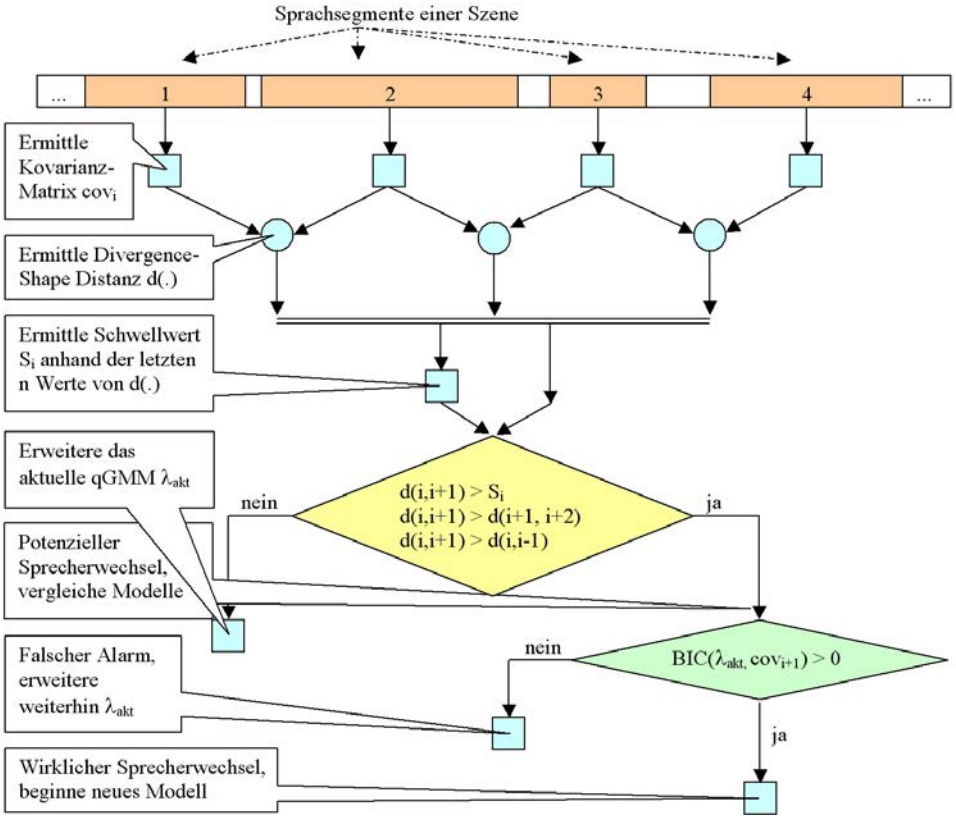

Abbildung 3.3: Ablaufdiagramm des Algorithmus zur Sprecherwechsel-Erkennung. Dargestellt ist nicht ein beispielhafter Fall, sondern der generelle Kontrollfluss.

zum Modellbau. Dieser Schritt soll nun etwas näher beschrieben werden:

Zuerst soll das erste Sprachsegment einer Szene betrachtet werden. Es stellt mit Sicherheit einen Sprecherwechsel dar[30], weshalb ein neues qGMM-Modell erstellt wird: Dieses bekommt eine Misch-Komponente, deren Mittelwert und Kovarianz aus dem ersten Segment ermittelt werden. Nun soll dieses Modell mit den Daten des zweiten Segments verfeinert werden[31]. Hierzu wird überprüft, wie "gefestigt" die Annäherung der Kovarianz der ersten Modellkomponente schon ist: Sie wird mit den Daten des zweiten Sprachsegments angereichert und die Veränderung zu ihrer Ursprungsform betrachtet: Ist die Divergence-Shape-Distanz zwischen beiden Matrizen kleiner als ein Schwellwert, scheint die aktuelle Modellkomponente ordentlich ausgebildet zu sein. Mit den Daten des zweiten Segments wird deshalb eine neue aufgemacht. Aus dem qGMM mit einer Komponente ist nun eines mit zweien geworden.

dass das Ergebnis im Nachhinein hochskaliert wird.

[30] Zu Beginn einer neuen Szene wird immer ein neuer Sprecher angenommen. Zum einen, da die Verarbeitung szenenweise stattfindet und die letzte Szene vielleicht gar nicht (von dieser Maschine) bearbeitet wurde. Zum anderen, da durch sicherlich verschiedene akustische Bedingungen an dieser Stelle sowieso kein aussagekräftiges Ergebnis erzielt werden könnte.

[31] Unter der Annahme natürlich, dass kein potenzieller Wechsel stattfand.

Der Schwellwert zur Bestimmung, ob die Komponente "fertig" ist, errechnet sich dabei folgendermaßen: Die zu testende Matrix wird mit einem Faktor zwischen eins und zwei multipliziert und zwischen dem Original und dem Ergebnis die Divergence-Shape-Distanz berechnet. Deren Wert gilt nun als Schwellwert für obiges Verfahren. Anschaulich bedeutet dies, dass die alte Matrix sich durch die Ergänzung mit zusätzlichen Daten um nicht mehr als $(1 - Faktor) \cdot 100$ Prozent verändern darf. Der $Faktor$ kann dabei vom Benutzer angegeben werden.

Das obige Verfahren zur Modellverfeinerung läuft nun, bis die (parametrisierte, vom Benutzer veränderbare) maximale Anzahl Komponenten erreicht ist oder ein potenzieller Wechsel stattfindet. In diesem wird das so erstellte Modell benutzt, um eine etwas verlässlichere Grundlage für die notwendige Entscheidung zu erlangen. Das ΔBIC zwischen dem letzten Modell und der aktuellen Kovarianzmatrix wird dabei als gewichtete Summe des ΔBIC zwischen jeder einzelnen Komponente des Modells und der neuen Kovarianzmatrix berechnet.

3.4.3 Eliminierung von Hintergrundgeräuschen

Um die Freiheit von Hintergrundgeräuschen mittels GMM-IB's zu bewerkstelligen, sind auch an dieser Stelle noch einige Vorbereitungen zu treffen: Das Hintergrundmodell ist zu erstellen, wozu alle dazu zählenden Audiorahmen zusammengestellt werden müssen. Ist es konstruiert, findet es gemeinsam mit der Zusammenstellung der Sprachrahmen Eingang in den Trainingsalgorithmus des Sprachmodells. Zuvor ist allerdings noch zu ermitteln, ob die zu betrachtende Äußerung lang genug ist und welche Ordnung[32] die Modelle für Sprache und Geräusch haben sollen. Im Einzelnen läuft dies folgendermaßen ab:

Die Methode `buildSpeakerModels()` der Klasse `SC_SpeechProcessing` durchläuft schrittweise alle Sprachäußerungen einer Szene. Eine Äußerung ist dabei ein zusammenhängendes Sprachsegment oder eine Folge zusammenhängender Segmente, die von der Sprecherwechsel-Erkennung als mit Sicherheit sprecherhomogen markiert wurden. Somit ist gewährleistet, das die maximale zu diesem Zeitpunkt verfügbare Information über die Stimme eines Sprechers genutzt wird. Im Umfeld einer solchen Äußerung wird nun also nach reinen Nebengeräuschrahmen gesucht, die den akustischen Hintergrund während der Stimmaktivität mutmaßlich in seiner Reinform repräsentieren. Diese Eigenschaft der Rahmen kann dabei jedoch nur angenommen werden, in dem vorgeschlagenen System ist sie nicht überprüfbar. Um trotzdem die Wahrscheinlichkeit gering zu halten, dass Rahmen zur Erstellung des Hintergrundmodells herangezogen werden, die mit dem eigentlichen Nebengeräusch während der Sprachperiode nichts zu tun haben, werden nur diejenigen

[32] Unter der *Ordnung* eines statistischen Modells versteht man die Anzahl seiner Komponenten. Beim GMM oder GMM-IB sind dies also die einzelnen Mischdichten, bei einem HMM kann auch die Anzahl der Zustände gemeint sein.

genutzt, welche[33]

- als Geräusch oder Stille, jedoch nicht als nicht-stimmhafte Sprache oder Sprach-Pause deklariert wurden,
- in Geräuschpassagen zwischen Beginn und Ende eines Sprachsegments auftreten,
- zwischen dem Beginn des aktuellen Sprachsegments und dem Ende des vorangegangenen (oder dem Szenenbeginn) liegen oder
- zwischen dem Ende des aktuellen Sprachsegments und dem Beginn des nächsten (oder dem Szenenende) liegen.

Wenn die einzelnen Sprachsegmente längere Zeit auseinander liegen, kann es sein, dass auf diese Weise sehr viele Audiorahmen zur Modellgenerierung herangezogen werden, für welche die Wahrscheinlichkeit gering ist, dass sie auch dem Hintergrund der Sprache entsprechen. Deshalb wird den letzten beiden Bedingungen noch die Beschränkung auferlegt, dass die zu Anfang und zu Ende des aktuellen Segments liegenden Rahmen nur einen spezifizierbaren Zeitabschnitt weit von diesem entfernt sein dürfen. Enthält das aktuelle Segment beispielsweise die Worte "hallo Welt", so könnten für die Hintergrundmodellierung die Rahmen in der Pause zwischen den Worten sowie die $vectorsPerGaussian$ Sekunden davor und danach benutzt werden.

Der Parameter $vectorsPerGaussian$ ist dabei vom Benutzer spezifizierbar und wird auch in der Heuristik zur Ermittlung der Modellordnungen verwendet: Dieses Verfahren basiert auf der Umkehrung der Idee des "Fluchs der Dimensionalität": Je mehr Trainingsdaten vorhanden sind, desto mehr Parameter können verlässlich ermittelt werden, desto besser passt das Modell auf die Daten und damit die Stimme. Faktisch kann hier also ein Wert spezifiziert werden, der die Menge an Merkmalsvektoren angibt, welche das Vorhandensein einer neuen Modellkomponente rechtfertigen: Im Beispiel könnten 473 Merkmalsvektoren verfügbar sein, während der Wert von $vectorsPerGaussian$ bei 200 läge. Dies äußert sich in einer Ordnung von $473/200 = 2$ für das zu erstellende Modell. Auf diese Art ergibt sich auch die Zurückweisung zu kurzer Segmente: Ist die Bedingung für mindestens eine Modellkomponente nicht erfüllt, ermittelt die Heuristik also null Komponenten als angemessen, so wird das entsprechende Segment (sei es nun Sprache oder Hintergrund) abgewiesen, und es wird kein Modell erstellt.

Auf diese Weise werden also die Segmente untersucht und alle zusammengehörigen Geräuschrahmen in einem eigenen Container gespeichert. Anhand dessen wird das Geräuschmodell mittels der Standard-GMM-Trainingsprozedur wie in Abschnitt 2.4.4 beschrieben trainiert, falls genügend Rahmen vorhanden waren. Nun kann die Erstellung des Sprechermodells angegangen werden[34]. Sollte dabei kein Hintergrundmodell erstellt

[33] Die Idee, Hintergrundmodelle aus den Audiorahmen zwischen Sprachpausen abzuleiten, wurde beispielsweise schon in [LOGA 98] angeregt, ihre Tauglichkeit wurde jedoch noch nirgends näher untersucht.

[34] Es basiert nur auf den Audiorahmen, die als reine Sprache ohne die Zusätze Stille, Geräusch, Sprachpause oder nicht-stimmhafte Sprache markiert wurden.

worden sein, geschieht dies ebenfalls mittels Standard-GMM-Training, und das GMM-IB degeneriert zum GMM. Dies ist jedoch nicht der Regelfall, sondern lediglich ein Versuch, trotz unvollständiger Daten zu Ergebnissen zu kommen.

Formalismen im GMM-IB

Die grundlegenden Prinzipien des GMM-IB wurden bereits in Abschnitt 2.4.5 auf abstraktem Niveau erläutert und sollen hier nicht noch einmal präsentiert werden. Vielmehr soll der dort dargestellte allgemeine Formelapparat auf das spezielle Problem dieser Arbeit angepasst werden. Zur Erinnerung: Rose et al. entwickelten in [ROSE 94] allgemeine Formeln für die Interaktion zwischen Signal und Störeinfluss in beliebiger Form, präzisierten jedoch drei Spezialfälle, wie Abbildung 3.4 zeigt.

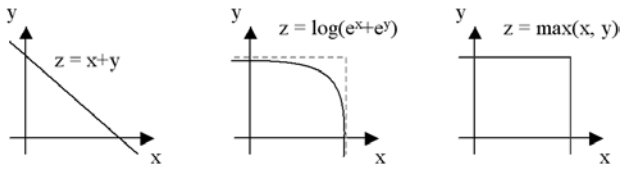

Abbildung 3.4: Drei Arten der Signal-Geräusch-Interaktion und die zugehörigen Kurven für die Integration in Formel 2.14: Additiv (links), Additiv bei logarithmierten Merkmalsvektoren (mittig) und Max-Modell (rechts). Nach [ROSE 94].

Für diese Arbeit ist dabei die Form rechts außen von Belang: Das *Max-Modell*. Dies mag zuerst verwundern, hat aber folgende Bewandtnis: Die zu eliminierenden Geräusche haben ursprünglich einen additiven Einfluss auf das Sprachsignal[35]. Durch die Verwendung von MFCC-Merkmalsvektoren[36] muss jedoch der mittlere Fall aus Abbildung 3.4 berücksichtigt werden. Dieser hat den Nachteil, dass für die folgenden Formeln keine geschlossene Lösung mehr gefunden werden kann. Deshalb kommt als Annäherung das Max-Modell zum Einsatz, wie es auch Nádas et al. in [NADA 88] vorschlagen.

Mit dieser Festlegung auf eine spezielle *Signalbeeinträchtigungs-Funktion* können die Formeln in Abschnitt 2.4.5 konkretisiert werden: Die Auftrittswahrscheinlichkeit eines bestimmten Merkmalsvektors unter der Annahme bestimmter Vorder- und Hintergrundzustände in Gleichung 2.14, wird zu[37]:

[35] Veranschaulichen lässt sich dies recht einfach am Beispiel eines Hollywood-Spielfilms: Dieser wird viele Nebengeräusche in Form von Musik und Effekten aufweisen, auch während Sprache. Wird dieser Film in Deutschland gezeigt, ist dies meist in einer lokalisierten Fassung der Fall; der Film wurde also nachsynchronisiert. Mischtechnisch wurden dabei die deutsche Tonspur und der originale Klang aufaddiert, um die komplette Audiospur zu formen. Voilá – spätestens jetzt ist der Effekt additiv! Im Gegensatz dazu existieren noch Faltungseffekte, die während der Übertragung zum Beispiel im Telefonkanal auftreten könnten. Im Bereich von Filmen sind diese jedoch vernachlässigbar.

[36] Diese entstehen ursprünglich aus einem logarithmierten Spektrum.

[37] Es sei nochmals darauf hingewiesen, dass aus Gründen der Übersichtlichkeit Anhang A.4 eine Zusammenstellung sämtlicher relevanter Formeln des GMM-IB wiederholt.

$$p(\vec{z_t}|i_t = i, j_t = j, \lambda) = a_j(\vec{z_t}) \cdot B_i(\vec{z_t}) + b_i(\vec{z_t}) \cdot A_j(\vec{z_t}) \tag{3.2}$$

Hierbei sind $B_i()$ und $A_j()$ Gauß'sche Verteilungsfunktionen[38] bestimmter Vorder- und Hintergrundzustände, $b_i()$ und $a_j()$ die entspfrechenden Dichtefunktionen[39].

Weiter kann nun eine geschlossene Form für die bedingten Erwartungswerte in den Gleichungen 2.18 und 2.19 angegeben werden[40]:

$$p(x_t = z_t|i_t = i, j_t = j, \lambda) = \frac{b_i(z_t) \cdot A_j(z_t)}{a_j(z_t) \cdot B_i(z_t) + b_i(z_t) \cdot A_j(z_t)} \tag{3.3}$$

$$\begin{aligned} E\{x_t|z_t, i_t = i, j_t = j, \lambda\} &= p(x_t = z_t|i_t = i, j_t = j, \lambda) \cdot z_t \\ &\quad + (1 - p(x_t = z_t|i_t = i, j_t = j, \lambda)) \\ &\quad \cdot E\{x_t|x_t < z_t, i_t = i, j_t = j, \lambda\} \end{aligned} \tag{3.4}$$

$$\begin{aligned} E\{x_t^2|z_t, i_t = i, j_t = j, \lambda\} &= p(x_t = z_t|i_t = i, j_t = j, \lambda) \cdot z_t^2 \\ &\quad + (1 - p(x_t = z_t|i_t = i, j_t = j, \lambda)) \\ &\quad \cdot E\{x_t^2|x_t < z_t, i_t = i, j_t = j, \lambda\} \end{aligned} \tag{3.5}$$

Gleichung 3.3 ist dabei die Wahrscheinlichkeit, dass im aktuellen Modellzustand (i, j) die Beobachtung z gleich der reinen Sprache x ist, also nicht verunreinigt wurde. Die so entwickelte Formulierung zeigt etwas anschaulicher den eigentlichen Sinn der Formeln 3.4 und 3.5, als dies unter Verwendung von Gleichung 3.2 möglich wäre. Für eine ausführlichere Erklärung zu diesem Sachverhalt sei auf [ROSE 94] verwiesen.

Interessant ist nun jedoch, dass die weitere Lösung dieser Gleichungen[41] in der Literatur nirgends zu Ende gebracht wurde: [ROSE 94] veröffentlicht zwar eine Formel für den Erwartungswert von x (Gleichung 3.4), im Verlauf der Nachforschung zu einer Entsprechung für den Erwartungswert von x^2 zeigte sich jedoch, dass den Autoren dort ein Fehler unterlaufen ist[42]. Das Max-Modell des GMM-IB ist bislang also immer nur fehlerhaft bzw. unvollständig publiziert worden.

[38] Die *Gauß'sche Verteilungsfunktion* ist das Integral über die Gauß-Dichte von $-\infty$ bis zu ihrem Argument: $B(z) = \int_{-\infty}^{z} b(x)dx = \frac{1}{2} \cdot \left(erf(\frac{z-\mu}{\sigma} \cdot \frac{\sqrt{(2)}}{2}) + 1 \right)$. Anschaulich gibt sie die Wahrscheinlichkeit wieder, dass ein Wert kleiner oder gleich dem Argument von einem Prozess erzeugt wurde, welcher dieser Verteilung genügt. Das *Gauß'sche Fehlerintegral* $erf()$ berechnet sich dabei als $erf(x) = \int_0^x e^{-\frac{t^2}{2}}$ [ABRA 64].

[39] Hier und im Folgenden gilt: Mit b oder B bezeichnete Funktionen gehören zu dem Vordergrund-(Sprach-)Prozess, a und A zum Hintergrund-(Geräusch-)Prozess.

[40] Man beachte die Vernachlässigung der Vektorschreibweise mit der selben Begründung wie schon in Abschnitt 2.4.5.

[41] Einschließlich Gleichung 3.5, welche bisher auch unveröffentlicht ist. Sie ergibt sich jedoch leicht aus $E\{x_t^2|z_t, i_t = i, j_t = j, \lambda\} = \int \int_C x_t^2 \cdot p(x_t|z_t, i_t = i, j_t = j, \lambda)dx_t dy_t = a_j(z_t) \cdot \int_{-\infty}^{z_t} x_t^2 \cdot b_i(x_t)dx_t + z_t^2 \cdot b_i(z_t) \cdot \int_{-\infty}^{z_t} a_j(z_t)dy_t = \frac{a_j(z_t)B_i(z_t)E\{x_t^2|x_t < z_t, i_t = i, j_t = j, \lambda\} + z_t^2 b_i(z_t)A_j(z_t)}{p(z_t|i_t = i, j_t = j, \lambda)}$ anhand des Rechenweges in [ROSE 94].

[42] Eine Nachfrage bei den Autoren ergab, dass es sich tatsächlich um einen Fehler handelt [REYN 04].

Im Folgenden soll deshalb die Herleitung des restlichen Formelapparats für das Max-Modell beschrieben werden. Als Ausgangspunkt der Überlegungen dient dabei [NADA 89], die ursprüngliche Quelle auch der Formeln von Rose et al.: Die dort präsentierten Herleitungen sind allerdings für standardisierte Normalverteilungen gemacht, Gauß-Dichte bzw. -Verteilungsfunktionen also, bei denen für den Erwartungswert $\mu = 0$ und die Varianz $\sigma^2 = 1$ gilt. Der Zusammenhang zu der in [ROSE 94] dargestellten Form mit parametrisierten Normalverteilungen ist dabei in Anhang A.3 gegeben. Mit den Formeln A.38 und A.39 folgt damit aus den Angaben in [NADA 89]:

$$E\{x_t | x_t < z_t, i_t = i, j_t = j, \lambda\} = \mu_i - \sigma_i^2 \cdot \frac{b_i(z_t)}{B_i(z_t)} \tag{3.6}$$

$$E\{x_t^2 | x_t < z_t, i_t = i, j_t = j, \lambda\} = \left(\mu_i^2 + \sigma_i^2\right) - \sigma_i^2 \cdot \frac{b_i(z_t) \cdot (z_t + \mu_i)}{B_i(z_t)} \tag{3.7}$$

Der Vergleich zeigt: Gleichung 3.6 enthält korrekterweise den Faktor σ_i^2, welcher in den Ausführungen von Rose et al. versehentlich zu einem reinen σ verkam. Damit ist der Formelapparat nun vollständig, und der EM-Algorithmus kann angewandt werden, um die Parameter des Sprachmodells auch in Anwesenheit des Nebengeräuschs durch die Gleichungen 2.17 - 2.19 zu ermitteln. Die vor dessen Einsatz notwendige Initialisierung der Modellkomponenten wird durch zehn Iterationen des k-means-Algorithmus bewältigt, was zu von vorneherein recht aussagekräftigen, wenn auch noch nicht hintergrundbefreiten Clustern[43] führt.

Etwas mehr Anschauung

In Abschnitt 2.4.5 wurde ja schon auf die Verbindung zwischen den statistischen Formeln des GMM-IB und der etwas intuitiveren Technik der Geräusch-Maskierung hingewiesen: Fällt der Wert eines Merkmalsvektors in den Bereich, der durch das Geräuschmodell abgedeckt ist, oder liegt er gar darunter, so wird die entsprechende Komponente des Sprechermodells in Richtung $-\infty$ verschoben. Ein Blick in Abbildung 3.5 verdeutlicht dies:

Die Grafik zeigt die beiden extremen Fälle, dass entweder das Signal oder der Hintergrund in der Beobachtung z völlig die Oberhand gewonnen haben. Im ersten Fall rechts im Bild führt dies dazu, dass die Beobachtung selbst dem Signal gleichgesetzt wird[44] und somit komplett in die Neubestimmung der Modellparameter mit einfließt. Im zweiten Fall, in der Abbildung links zu sehen, ist die Beobachtung komplett maskiert und geht gar nicht in die Parameterverfeinerung ein[45]. Am deutlichsten wird diese Maskierung, wenn

[43] Auch der EM-Algorithmus und damit das Ermitteln von Modellkomponenten kann als Clusteringprozess aufgefasst werden: Gesucht werden die M Klassenzentren (Komponenten), welche die gegebenen Daten optimal beschreiben.

[44] Da $a(z) \approx 0, A(z) \approx 1$ und $0 < b(z) \leq \frac{1}{\sqrt{2\pi} \cdot \sigma}$ sowie $0 < B(z) \leq 1$ folgt $p(x = z | \ldots) = 1$.

[45] Da in diesem Fall wegen $A(z) \approx 0$ $p(z | \ldots) = 0$ gilt.

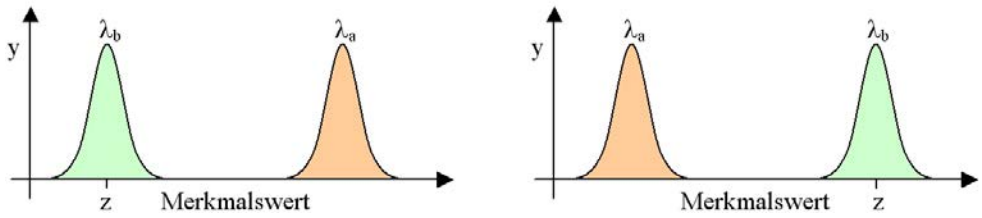

Abbildung 3.5: Beispiel eines eindimensionalen Merkmalsvektors z und der Dichtefunktions-Grafen des Vorder- und Hintergrundmodells. Links: Geräusch maskiert Signal vollkommen. Rechts: Signal und Beobachtung stimmen komplett überein.

der dargestellte Merkmalsvektor beispielsweise als Amplitude einer bestimmten Frequenz f angesehen wird. Während die y-Achse in Abbildung 3.5 nur die Höhe der Glockenkurve angibt[46], stellt die x-Achse in diesem Fall den Merkmalswert "Amplitude" dar. Ein größerer x-Wert verheißt also eine höhere Amplitude auf der Frequenz f, was intuitiv gleichbedeutend ist damit, dass die höhere Amplitude die niedrigere "übertönt" oder eben maskiert. Dem statistischen Rahmenwerk ist es nun jedoch zu verdanken, dass diese Maskierung nicht vollständig sein muss. Ist der Abstand zwischen den Amplituden gering, überlappen sich die Dichtefunktionen also, kann auch aus einem vom Nebengeräusch überlagerten Sprachsignal Information gewonnen werden.

Dies ist in den Formeln 3.6 und 3.7 zur Bestimmung der Erwartungswerte von x und x^2 verborgen. Abbildung 3.6 verdeutlicht deren Wirkungsweise.

Es ist deutlich zu sehen, dass der zur Neubestimmung der Modellparameter heranzuziehende Wert von $E\{x|z,\ldots\}$ unter dem Einfluss des Hintergrundmodells in die negative Richtung[47] beeinflusst wird. Dieser Einfluss lässt sofort nach, wenn die Beobachtung dem Dunstkreis des Hintergrundmodells entronnen ist. Wenn das auf diese Weise ermittelte Vordergrundmodell nun in einer Umgebung mit gleichem oder geringerem Signal-Geräusch-Abstand zu Testzwecken eingesetzt wird, wird das dortige Hintergrundmodell es erneut maskieren [REYN 92]. Zu dem Testergebnis tragen auf diese Weise alle Signalkomponenten entsprechend ihrer Wahrscheinlichkeit, frei von Geräuschverunreinigung zu sein, bei. Ein gutes Hintergrundmodell[48] vorausgesetzt, fallen hier also nur die nicht-verunreinigten, da nicht vom Geräuschmodell maskierten Signalkomponenten ins Gewicht, während die unzuverlässigen Signalanteile entsprechend wenig bis gar nichts zur Modellerstellung bzw. Testwertermittlung beitragen.

[46] Welche sich direkt aus deren Parameter σ ergibt und deshalb keine direkt anschauliche Relevanz hat.

[47] Um im Beispiel von eben zu bleiben: Hin zu geringerer Amplitude bei natürlich gleichbleibender Frequenz.

[48] Ein Hintergrundmodell kann dann als *gut* oder *passend* bezeichnet werden, wenn es möglichst exakt alle verunreinigenden Signalanteile modelliert, ohne jedoch durch zu hohe Toleranzen auch weite Teile des sauberen Nutzsignals zu verdecken.

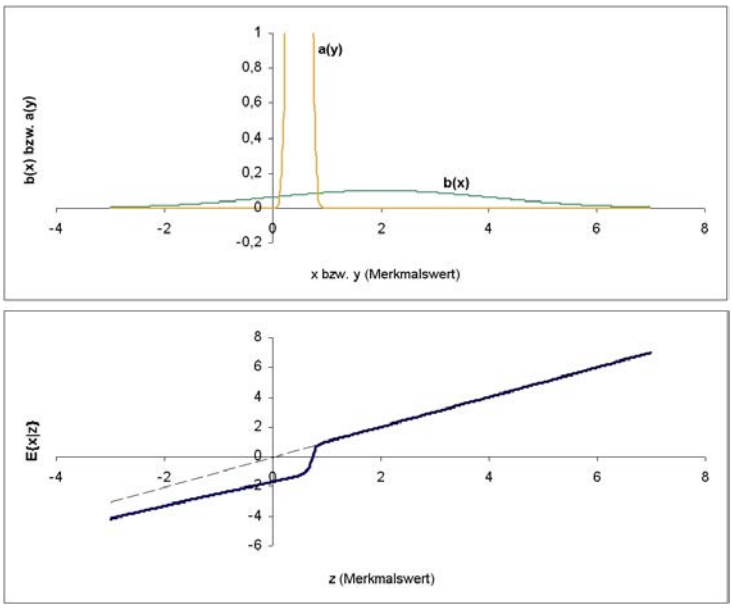

Abbildung 3.6: Entwicklung von $E\{x|z,\ldots\}$ in Abhängigkeit des Wertes der Beobachtung z bei gegebener Signal- (grün) und Hintergrunddichte (rötlich, der Skalierung halber abgeschnitten). Die Werte wurden nach den Angaben in [ROSE 94] durch das erstellte System gemessen. Die gestrichelte Hilfslinie in der unteren Grafik verdeutlicht den Verlauf, den $E\{x|z,\ldots\}$ in Abwesenheit von Nebengeräuschen nehmen würde: $x = z$.

Das Endprodukt der Sprecherrepräsentation

An dieser Stelle soll nun wieder zu der algorithmischen Ebene der Sprechermodell-Erstellung zurückgekehrt werden: Die Methode `buildSpeakerModels` hat nun also bestenfalls für jeden sprecherhomogenen Bereich einer Szene eine GMM für die dort herrschenden Hintergrundbedingungen sowie ein GMM-IB erstellt, welches die Stimme befreit von dem Einfluss ihres akustischen Hintergrunds repräsentiert. Schlechtestenfalls war das Sprachsegment zu kurz, um modelliert zu werden, und wurde verworfen. Es wird so in der Endauswertung[49] keine Rolle mehr spielen, da es nicht aussagekräftig erarbeitet werden kann. Der zweite "schlechte" Fall ist, dass das Sprachsegment zwar lang genug für die Modellerstellung war, jedoch nicht genug Daten für ein Hintergrundmodell gesammelt werden konnten. Dann existiert ein reines GMM für die Sprache ohne Anreicherung, also inklusive Nebengeräuschen.

Wenn das Sprachsegment nicht zu kurz zur Verarbeitung war, erfolgt nun die Erstellung eines `SC_Cluster`-Objekts: Start- und Endpunkt des Segments, das Sprach- und Nebengeräuschmodell[50] sowie die mit Sprachabschnitten korrespondierenden Merkmals-

[49] Dem *Sprecherindex*.

[50] Sollten keine Daten für ein Geräuschmodell vorgelegen haben, werden an dieser Stelle Platzhalter eingeschoben: Ein Modell der Ordnung null bzw. ein leerer Datencontainer, so dass in der Clusteringstufe generell immer von einer einheitlichen Struktur der zu verarbeitenden Objekte ausgegangen werden kann: Jeder Cluster enthält für jedes von ihm repräsentierte Segment ein Hintergrundmodell einer Ordnung ≥ 0.

vektoren werden hier zusammengefasst. Anschaulich präsentiert so jedes Cluster-Objekt erst einmal einen eigenen Sprecher.

3.4.4 Klassifikation

Die Klassifikation setzt nun ein, wenn alle einzelnen Clusterobjekte der modellierbaren Sprachsegmente verfügbar sind. Das Vorgehen ist dabei typisch für ein agglomeratives Clusteringverfahren: Anhand einer auf einem Distanzmaß beruhenden Distanzmatrix werden die beiden einander ähnlichsten Cluster herausgegriffen und zu einem verschmolzen. Anschließend überprüft ein globales Kriterium, ob diese Verschmelzung von Vorteil war, was eine Fortführung des Prozesses bedeutet, oder ob sie schlecht war, was zu einem Abbruch des Prozesses und Ausgabe der Ergebnisse führt.

Über die Wahl des BIC als Abbruchbedingung sowie der Distanzmaße GLR und CLR als Metriken wurde weiter oben schon berichtet, und auch die Funktionsweisen dieser Verfahren standen schon zur Debatte. Im Folgenden soll es daher um einen weniger hervortretenden Aspekt des Klassifikationsprozesses gehen: Die Verschmelzung einzelner Cluster und das Weiterarbeiten mit solchen zusammengesetzten Clustern mittels der bestehenden Methoden des Modell-Testens.

Cluster-Verschmelzung

Wenn zwei Cluster zu einem einzigen verschmolzen werden, bedeutet dies anschaulich, dass die einzelnen Sprachsegmente, die von den ursprünglichen Clustern repräsentiert wurden, nun einem einzigen Sprecher zugeordnet werden. Es ist nun also möglich, das Sprechermodell für den resultierenden Cluster anhand der Daten beider Ausgangspunkte zu verfeinern, um ein umfassenderes Abbild seiner Stimme zu erstellen.

Die naheliegende Methode, das Gesamtsprechermodell zu generieren, wäre nun sicherlich, sämtliche Sprachrahmen aus den Ursprungsclustern zusammenzukopieren und erneut ein GMM-IB darauf zu trainieren. Doch die einzelnen Sprachsegmente sind höchstwahrscheinlich mit unterschiedlichen Hintergrundgeräuschen verunreinigt. Und für diese existieren jeweils eigenständige GMM's. Wenn nun auch die Hintergrund-Merkmalsvektoren zusammenkopiert würden[51], um so ein Gesamthintergrundmodell zu erstellen, welches wiederum zur Sprachanreicherung während des Trainingsvorgangs für das Gesamtsprechermodell beiträgt, geht sehr wahrscheinlich Information verloren:

Angenommen, die beiden Ursprungscluster enthalten jeweils ein Sprachsegment, einmal verunreinigt von Nebengeräuschen im unteren bis mittleren Frequenzbereich, einmal kompromittiert von Störungen im mittleren bis hohen Frequenzbereich. Wird nun ein gemeinsames Nebengeräuschmodell aus diesen Daten gewonnen, wird dieses den gesamten

[51] Die an dieser Stelle im Programm letztendlich auch gar nicht mehr zur Verfügung stehen.

Frequenzbereich von niedrig bis hoch maskieren, und aus den beiden Sprachäußerungen ließe sich kaum Information gewinnen, anhand derer ein verlässliches Sprechermodell konstruierbar wäre.

An dieser Stelle kommt der Vorteil der vorgeschlagenen Datenhaltung zum Tragen: Die Merkmalsvektoren der einzelnen Sprachsegmente, die zu einem Cluster gehören, werden nicht einfach zusammenkopiert, sondern sind in einem eigenen Container pro Segment untergebracht. Diese Container sind jedoch untereinander als Liste verknüpft und lassen sich rasch durch Methoden der Containerklasse `SV_Data` verarbeiten. Ähnlich ist es mit den einzelnen Hintergrundmodellen. Auf Basis dieser Infrastruktur wurde eine Erweiterung des GMM-IB eingeführt:

Erweiterung des GMM-IB

Zum Trainieren eines Gesamtsprechermodells werden die Sprachmerkmalsvektoren zwar alle zusammenkopiert, bei der Iteration über die Merkmalsvektoren wird jedoch für jeden neuen Vektor $\vec{z_t}$ anhand seines Index' t und dem Wissen über die Länge der Listenelemente überprüft, ob er noch aus dem selben Segment stammt wie sein Vorgänger. Fand an der aktuellen Position ein Segmentwechsel statt, wird auch das Hintergrundmodell umgeschaltet, also das nächste aus der verketteten Liste der Hintergrundmodelle gewählt. Nun wird auch klar, wozu Platzhaltermodelle in diese Liste eingefügt werden mussten, falls kein wirkliches Hintergrundmodell ermittelt werden konnte: Die Liste darf nicht unterbrochen werden, damit der Umschaltvorgang reibungslos ablaufen kann. Natürlich muss die Trainingsprozedur im Falle eines reinen Platzhaltermodells Vorkehrungen treffen, um in diesem Fall auf die Standard-Trainingsmethode eines GMM umzusteigen:

Wie den Neubestimmungsformeln 2.17 bis 2.19 anzusehen ist, erfordern diese jeweils eine Summenbildung über die Anzahl N der Hintergrundkomponenten. Gilt im Falle eines Platzhalters $N = 0$, so würden die Formeln scheitern und keine neuen Werte berechnet werden. Deshalb wurde in diesem speziellen Fall eine *virtuelle* Hintergrundkomponente eingeführt, deren Dichtefunktion $a(\vec{z_t})$ immer den Wert null und deren Verteilungsfunktion $A(\vec{z_t})$ immer den Wert eins annimmt. Daraus folgt $p(\vec{z_t}|\ldots) \approx b(\vec{z_t})$[52] und damit $E\{\vec{x_t}|\vec{z_t},\ldots\} \approx \vec{z_t}$[53]. Dies lässt sich so interpretieren, dass für den Fall der virtuellen Hintergrundkomponente der Standard-GMM-Trainingsansatz verfolgt wird: Die Beobachtung $\vec{z_t}$ geht direkt ohne die Beachtung eines akustischen Hintergrunds in die Berechnung der Sprachmodell-Parameter ein.

Auf ähnliche Art wird natürlich auch beim *Testen* eines GMM-IB gegen die Daten verschiedener Segmente vorgegangen: Auch hier kann pro Segment ein eigenes Hintergrundmodell verwendet werden, und in Abhängigkeit von dessen Ordnung wird zwischen dem GMM-IB- und dem GMM-Algorithmus gewechselt.

[52] $p(\vec{z_t}|\ldots)$ ist dabei die Gleichung 2.14.
[53] $E\{\vec{x_t}|\vec{z_t},\ldots\}$ steht hier für Gleichung 3.4.

Wenn man dieses Prinzip auf die Spitze treibt, erhält man interessanterweise das, was in [ROSE 94] als der noch zu leistende Forschungsbeitrag zum GMM-IB bezeichnet wird: Eine Methode, um mit impulsartigen Nebengeräuschen umzugehen. Denn nichts spräche dagegen, ein Sprechersegment in viele kleine Unterabschnitte aufzuteilen, für welche jeweils ganz explizite Hintergrundmodelle existierten. So ließe sich beispielsweise eine Dialogszene vor dem akustischen Hintergrund einer Schiesserei deutlich besser modellieren, wenn jeder abgegebene Schuss durch ein spezielles Modell mit extremen Spitzen im Frequenzbereich des Knalls repräsentiert würde und ansonsten ein gemäßigtes Modell zum Einsatz käme, als wenn die ganze Szene lang ein einzelnes Modell den Hintergrund darstellte, welches jedoch weder die Schüsse noch die sonstigen Störeinflüsse explizit beschriebe.

Dieses Verfahren wurde im Zuge dieser Arbeit zwar nicht explizit so getestet, doch die nötigen Grundlagen wurden entwickelt und umgesetzt, so dass nichts gegen einen Einsatz wie eben beschrieben spräche. Dies würde die größte Einschränkung des GMM-IB, nur für gleichförmigen Hintergrund tauglich zu sein, erheblich abschwächen.

Auf eine andere Interpretationsweise dieses Aspektes sei hierbei noch aufmerksam gemacht: Die Wechsel der Hintergrundmodelle bei Verlassen des Merkmalsraums eines Segments ließen sich als Zustandswechsel auffassen, die Verharrung bei einem Hintergrundmodell wäre analog dazu das Verbleiben in einem Zustand. Abbildung 3.7 visualisiert dies. Mit dieser Anschauungshilfe fällt es nicht schwer, den Bogen zu der Analogie mit einem *Hidden Markov Modell* zu spannen, welches die Daten in ein durch einen Zustandsautomaten modelliertes zeitliches Korsett zwingt.

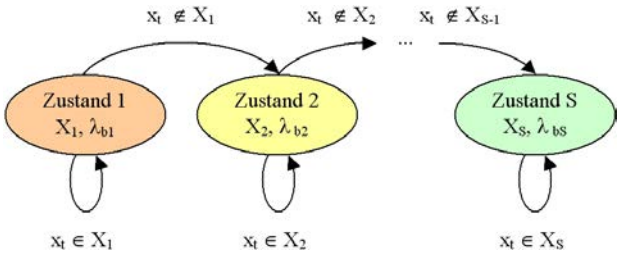

Abbildung 3.7: Die Trainings- und Testalgorithmen des GMM-IB aufgefasst als endlicher Zustandsautomat, wobei der Wechsel des Hintergrundmodells als Zustandswechsel aufgefasst wird. Die Zustandsübertritts-Wahrscheinlichkeit ist dabei solange eins, wie die Merkmalsvektoren $\vec{x_t} \in X_s$ aus dem aktuellen Segment $s \in \{1, ..., S\}$ stammen. Ist dies nicht mehr gegeben, wird ebenfalls mit Wahrscheinlichkeit eins in den Folgezustand gewechselt.

Modellkombination

Die obigen Überlegungen wurden ursprünglich angestoßen durch die Idee, das aus zwei einzelnen Clustern resultierende Sprechermodell durch Neuberechnung anhand der Ausgangsmerkmalsvektoren zu erstellen. Und so interessant die daraus entstandenen Schlussfolgerungen auch sind[54], diese Grundannahme stellt an sich noch nicht die optimale Lösung für das Problem der Modellkombination dar: Wie bereits angesprochen, ist der EM-Algorithmus ob seiner iterativen Rechenvorschrift kein sehr ressourcenschonendes Verfahren, und eine komplette Modellerstellung bei jeder Clusterverschmelzung sprengt den Rahmen des Erträglichen[55].

Auf der Suche nach performanteren Methoden, welche überdies die erbrachten Vorleistungen[56] nicht völlig ignorieren, fiel die Wahl auf ein Verfahren, welches auch in [DUNN 00] erwähnt wird: Ein statistisches Modell einer Stimme lässt sich intuitiv aus zwei einzelnen, unabhängigen Modellen derselben Stimme dadurch erstellen, dass die ursprünglichen Misch-Komponenten zusammenkopiert und die einzelnen Gewichte renormalisiert[57] werden.

Dies resultiert zwar mit der Zeit in einer Explosion der Komponentenanzahl, und sicherlich werden auf diese Weise auch Komponenten in einem Modell auftauchen, die beinahe identische Parameter haben und deshalb zusammengefasst werden könnten. Um jedoch durch falsche Zusammenfassungen keine Genauigkeit zu verlieren, und da ansonsten die hergeleitete Form des ΔBIC nicht angewandt werden könnte[58], wurde in diesem Kontext darauf verzichtet, zumal sich die Literatur für diesen Ansatz ausspricht [AJME 02].

[54] Und so wichtig und plausibel sie in *anderen* Bereichen auch sein mögen, zum Beispiel beim Testen des Modells.

[55] Abschnitt 4.2.3 wird hier Laufzeitbeispiele präsentieren.

[56] Die beiden erstellten Sprechermodelle, die nun verschmolzen werden sollen.

[57] Ihre Summe muss eins ergeben.

[58] Wie Anhang A.2 darlegt, lässt sich in der Formel des *Bayes'schen Informations-Kriteriums* der aufgaben- und datenabhängige Strafterm eliminieren, wenn das resultierende Modell als Ordnung die Summe der Ordnungen seiner Elternmodelle erhält.

Kapitel 4

Realisierung

4.1 Aspekte der Implementierung

4.1.1 Wahl der Werkzeuge

Im Verlauf dieser Arbeit wurde es schon mehrmals notwendig, kurz auf implementierungsspezifische Details die entwickelten Algorithmen betreffend einzugehen, um im gegebenen Zusammenhang bestimmte Entscheidungen zu begründen. Im Folgenden soll das Thema Implementierung etwas systematischer angegangen werden. Dazu gehört zu Anfang auch die grundsätzliche Wahl der Programmiersprache und der verfügbaren Bibliotheken.

Erstere Wahl viel auf die Sprache *C++*. Die Gründe hierfür sind schnell zusammengestellt: C++ bietet die Möglichkeit, schnelle und effiziente Algorithmen ohne einen großen Korb nützlichen, doch nicht zwingend notwendigen Ballasts zu erstellen. Aufgrund dieser Eigenschaft wurde es auch schon für alle weiteren zeitkritischen Verfahren innerhalb von Mediana verwendet, so dass sich hier auch eine projektinterne Tradition fortführt, die etwas zur Kompatibilität der einzelnen Subsysteme untereinander beiträgt.

Ein weiterer Grund für C++ war natürlich auch die Verwendung der Klassenbibliothek SC_Lib als Fundament. Diese ist komplett in dieser Sprache entwickelt worden, und so macht es auch hier aus Gründen der Kompatibilität auf Quelltextebene und der damit verbundenen besseren Überführbarkeit mancher Lösungsansätze Sinn, C++ zu wählen.

Ansonsten wird in den Entwicklungen SC_Lib und SCiVo auf keine externen Bibliotheken außer den zum ANSI-Standard gehörenden verwiesen. Dies hat natürlich eine leichtere Portabilität zur Folge, welche im Entwurf auch gefordert wurde. Es schließt jedoch auch den Verzicht auf grafische, plattformspezifische Oberflächen ein[1], so dass SCiVo und die

[1]Bei grafischen Benutzerschnittstellen (GUI aus dem englischen "Graphical User Interface") kann, von einigen portablen Bibliotheken wie `QT` oder `Swing` einmal abgesehen, eigentlich davon ausgegangen werden, dass sie plattformgebunden sind. Und selbst der Einsatz einer Mutliplattform-GUI verlangt durch die Bereitstellung eines Grafik-Subsystems mehr von seinem Wirtssystem, als es die reine Aufgabe der Sprecherklassifikation erfordern würde.

SC_Lib rein kommandozeilenorientiert arbeiten[2]. Leider wird dieses Ansinnen der völligen Plattformunabhängigkeit durchkreuzt durch die Implementierung der SV_Lib, welche teilweise auf Microsoft-Windows-spezifische Methoden zurückgreift. Doch da sie im Quelltext verfügbar ist, ließe sich dieses Problem im Nachhinein beheben.

Die letzte Bemerkung kollidiert jedoch mit einem hehren Prinzip: Dass der Quelltext der Klassenbibliothek verfügbar war, ist ein besonders zu schätzender Glücksfall, jedoch keinesfalls die Norm. Deshalb sollte diese Tatsache nicht dadurch überstrapaziert werden, dass bei jeder Gelegenheit die schon bestehende Klassenstruktur umgebogen und deren Quelltext angepasst wird. Vielmehr sollte auf das Bestehende aufgebaut werden. Deshalb galt für die Programmerstellung die Regel, die Originalquellen nach Möglichkeit nicht zu verändern[3].

Eine weitere Implementierungsrichtlinie galt für die Entwicklung identischer Methoden für unterschiedliche Argumenttypen[4]. Hier wurde jeweils auf die Implementierung mittels Templates[5] zurückgegriffen, um unnötige Mehrarbeit zu vermeiden. Die so entstandenen Funktionen befinden sich in dem Paket `SC_Aux`. Dies sind jedoch alles eher Kleinigkeiten im Hinblick auf die Entscheidung, die Bibliothek SC_Lib objektorientiert zu entwickeln. Dieses Konzept ist jedoch so weit verbreitet, dass die spezifischen Merkmale der Verfolgung dieses Paradigmas hier keiner weiteren Erläuterung bedürfen.

Bleibt noch zu erwähnen, dass streng genommen auch die Organisation des Quelltextes zu den Implementierungsregeln zählt. Und in diesem Bereich wurde im Laufe der Entwicklung auf verschiedene Kleinigkeiten geachtet, die hier genannt werden sollen: Variablen betreffend kann gesagt werden, dass ihnen durchweg sprechende, auf ihre Bedeutung aufmerksam machende Namen gegeben wurden. Ebenfalls zur besseren Verständlichkeit sollte beitragen, dass algorithmisch interessante Teile des Quelltextes[6] ausführlich kommentiert wurden. Dabei wurde jedoch sowohl im Fließtext als auch bei der Wahl von Namen für Bezeichner ausschließlich aus dem englischen Sprachumfang geschöpft, um den Programmcode international nicht zu disqualifizieren.

[2] Zur Entwarnung sei hier schon auf den Anhang B hingewiesen, welcher die Entwicklung einer prototypischen Oberfläche beschreibt.

[3] Dies gelang bis auf drei Sonderfälle: Die Klasse `SV_Data` enthielt einen Programmierfehler in ihren Konstruktoren, der behoben werden musste. Die Klasse `GN_Matrix` warf teilweise Ausnahmen, anstatt Unstimmigkeiten durch entsprechende (und mathematisch korrekte) Rückgabewerte mitzuteilen. Und alle auf der Klasse `SV_Model` aufbauenden Klassen mussten bezüglich des nicht vorhandenen Rückgabewertes ihrer Methode `TrainModel()` angepasst werden.

[4] Ein gutes Beispiel ist hier eine Funktion zur Ermittlung des Maximums zweier Werte. Normalerweise müsste hier jeweils eine Version für `int`-Argumente, eine für `double`-Argumente etc. existieren.

[5] Aus dem Englischen für *Vorlage*.

[6] Diese Beschreibung trifft auf einen sehr großen Teil des Codes zu.

4.1.2 Die Bibliothek SV_Lib

Als Fundament dieser Arbeit wurde die Klassenbibliothek SV_Lib schon des öfteren erwähnt. Hier soll nun kurz auf deren Aufbau und Funktionalität eingegangen werden. Die folgende Liste enthaltener Klassen und deren Funktionalitäten ist keineswegs vollständig, sondern beschränkt sich auf die für diese Arbeit wesentlichen Teile. Für Vollständigkeit sei auf [HE 99b] verwiesen.

- Klassen zum Einlesen eines digitalisierten Audiosignals
  - `SV_Signal` Einlesen von Audiodaten im Rohformat
- Klassen zum Aufnehmen von und Arbeiten mit Merkmalsvektoren
  - `SV_Data` Container für Merkmalsvektoren
  - `SV_DataIO` Laden und Speichern von `SV_Data`-Objekten auf einem Datenträger
- Klassen zur Extraktion und Vorverarbeitung von Merkmalen
  - `SV_Feature` Basisklasse der Merkmalsextraktion
  - `SV_Feature_MFCC` Extraktion von MFCC's
  - `SV_Feature_Pitch` Extraktion der durchschnittlichen, wahrgenommene Tonhöhe
- Klassen zur Sprecherrepräsentation
  - `SV_Model` Basisklasse der Sprecherrepräsentation
  - `SV_Model_GMM` Gauß'sches Misch-Modell
  - `SV_Model_CHMM` Hidden Markov Modell mit kontinuierlicher Dichtefunktion
  - `SV_Model_DHMM` HMM mit diskreter Dichtefunktion
  - `SV_Model_VQ` VQ-Modell
- Zusatzfunktionen
  - `GN_FFT` Implementierung von FFT und DCT
  - `GN_Filter` Methoden zum Entwurf und der Anwendung digitaler Filter auf ein Signal
  - `GN_Matrix` Methoden zum Umgang mit Matrizen, unter anderem Invertierung und Bildung der Determinanten
  - `GN_Rand` Erzeugung gleich- oder normalverteilter Zufallszahlen
  - `GN_SigWin` Methoden zur Fensterung eines Signals. Implementiert sind unter anderem Hamming-, Hanning-, Dreiecks- und Blackman-Fenster

– `SV_Error` Globale Methode zur Fehlerbehandlung, Makros zum einfachen Anfordern von Speicher für mehrdimensionale Felder

Die SV_Lib verfügt als reine Klassenbibliothek über eine beträchtliche Anzahl unterschiedlicher Verfahren im Bereich der automatischen Sprachverarbeitung, jedoch nicht über ein Programm, welches diese Funktionen kapselt und dem Endanwender über eine Schnittstelle verfügbar macht. Eine sehr gute Dokumentation allerdings[7] und sinnvoll gewählte Codebeispiele machen den Programmierer schnell mit ihrer Einsatzweise vertraut.

4.1.3 Zur numerischen Stabilität iterativer Verfahren

Die meisten Aspekte der Implementierung sind von nun an recht geradlinig und bedürfen kaum weiterer Erläuterung. Die wenigen Konzepte, die eine Erwähnung wert wären, wurden zum großen Teil schon im vorangegangenen Kapitel besprochen, da sie eng mit der algorithmischen Struktur des Systems verbunden sind: Da wäre beispielsweise die Vorhaltung der Hintergrundmodelle und Merkmalsvektormengen in verketteten Listen, oder die globale Speicherung aller Informationen über jeden einzelnen Audiorahmen in einem Konstrukt namens `frameList`. Der Vollständigkeit halber sei hier noch erwähnt, dass es sich bei letzterem um eine zweispaltige Matrix des Datentyps `unsigned long int`[8] handelt, welche pro Zeile in der ersten Spalte die Informationen über den jeweiligen Rahmen in Form von Bitflags und in der zweiten Spalte die ihm zugeordnete eindeutige Sprecher-ID[9] verwaltet.

Ein Fall jedoch spielt noch eine besondere Rolle und findet deswegen hier gebührende Beachtung: Die Problematik der numerischen Stabilität iterativer Verfahren. Speziell im Fall der Anwendung des EM-Algorithmus auf die Parameter des GMM-IB traten hier gehäuft Schwierigkeiten auf, deren Lösung im Folgenden beschrieben werden soll.

Als iteratives Verfahren verbessert der Erwartungs-Maximierungs-Algorithmus in jeder Runde eine Menge von Parametern, bis für die als Abbruchbedingung verwendete Likelihood-Funktion Konvergenz angenommen werden kann. Nun setzt sich der Wert dieser Likelihood-Funktion nach Formel 2.20 aus dem Produkt mehrerer Summen zusammen. Das kann zu Problemen führen, da beim Rechnen mit Wahrscheinlichkeiten sehr kleine Werte[10] auftreten können, vor allem zu Beginn des Algorithmus, wo das Modell noch nicht gut an die Daten angepasst ist. Es passiert also häufig, dass das zu berechnende Produkt

[7] Die Dokumentation der SV_Lib verfügt über die beiden hervorstechenden Eigenschaften, kurz und trotzdem komplett zu sein. Sie diente daher als Vorlage zur Erstellung der API-Beschreibung der SC_Lib.

[8] Noch eine Implementierungsregel: Variablen, die mit Sicherheit nur positive Werte annehmen können, wurden bewusst als `unsigned` deklariert.

[9] Weitere Erläuterungen über die Art dieser Informationen und die Form der Sprecher-ID können der Abbildung B.2 und ihrem Umfeld entnommen werden: Dort enthalten sind eine Erklärung über das Zustandekommen der ID's, und das Beispiel eines Klassifikationsprotokolls enthält einen Auszug aus der `frameList`, welchem die enthaltenen Daten entnommen werden können.

[10] Faktoren mit zweistellig-negativen Exponenten sind keine Seltenheit.

mit Wert null angegeben wird, nicht aufgrund des wahren Wertes, sondern wegen Rechenungenauigkeiten und Limitationen der Gleitpunktarithmetik moderner Rechner.

Das Mittel der Wahl, um dieser Widrigkeiten Herr zu werden, heißt *logarithmische Verarbeitung*: Durch Umwandlung der Likelihood- in eine Log-Likelihood-Funktion[11] verwandeln sich Multiplikationen in Additionen und Divisionen in einfache Subtraktionen. Jetzt können zwar auch noch numerische Schwierigkeiten auftreten[12], diese beeinträchtigen die Genauigkeit jdeoch nur noch in einem vernachlässigbar kleinen Rahmen beeinträchtigen, anstatt den gesamten Wert unbrauchbar zu machen: Sind zwei aufeinander folgende Ergebnisse der EM-Iterationen gleich null, bricht das Verfahren ab, da es Konvergenz annehmen muss. Unterscheiden sich beide Ergebnisse jedoch um einen noch so kleinen Wert, beispielweise den Faktor 10^{-100}, kann der EM-Algorithmus fortgesetzt werden und noch viele Parameteränderungen vollbringen.

Mit diesem einfachen Konzept lässt sich zum Beispiel der EM-Algorithmus des Standard-GMM stabilisieren. Für das GMM-IB hingegen sind noch ein paar weitere Eingriffe nötig. Zum einen kann hier die Berechnung des Gauß'schen Fehlerintegrals fehlschlagen: He's SV_Lib bietet zwar eine Methode an, die sehr effizient eine Berechnung vornehmen kann, doch versagt diese leider bei sehr kleinen Werten des Integrals, indem sie anstelle des wirklichen Wertes eine Null zurück liefert. Dass es sich tatsächlich um einen falschen Wert handelt, zeigt sich daran, dass in diesen Fällen der Wert der Gauß'schen Dichtefunktion für identische Parameter ungleich null ist. Da bekannt ist, dass die Gauß-Dichte asymptotisch gegen die x-Achse verläuft, muss es also eine Fläche unter der Kurve geben, wenn der aktuelle Funktionswert über null liegt. Motiviert von dieser einfachen Feststellung wurde eine Approximationsfunktion für das Gauß'sche Fehlerintegral und den Fall des Versagens der althergebrachten Variante geschaffen:

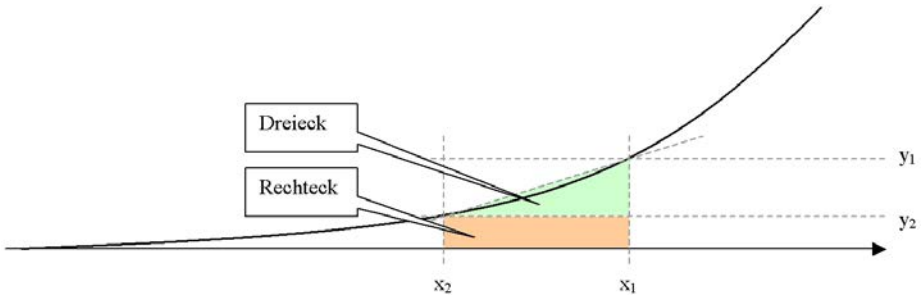

Abbildung 4.1: Approximation des Gauß'schen Fehlerintegrals durch Trapeze.

Sie geht davon aus, dass sich die Fläche unter der Glockenkurve wie in Abbildung 4.1 gezeigt näherungsweise durch die Summe der Flächeninhalte kleiner Trapeze bestimmen

[11] Wie es auch allgemein in der Literatur üblich ist: Anstatt $p(Z|\lambda)$ wird $\log p(Z|\lambda)$ berechnet und ausgegeben.

[12] Beispielsweise ist auf einem Rechner mit x86-Architektur und Variablen vom Typ `double` $1,0 + 10^{-17} = 1,0$, da ein Unterschied in der 17. Stelle hinter dem Komma in dessen Fließkomma-Arithmetik nicht abbildbar ist.

lässt:

$$\int_{-\infty}^{x} b(x)dx \approx \sum \left(\underbrace{((x_1 - x_2) \cdot y_2)}_{\text{Rechteck}} + \underbrace{\left(\frac{(x_1 - x_2) \cdot (y_1 - y_2)}{2} \right)}_{Dreieck} \right) \tag{4.1}$$

Die Summation nimmt dabei so lange neue Trapeze in die Berechnung auf, wie sich das Ergebnis noch verändert. Bleibt es (aufgrund der Rechenungenauigkeit) gleich, ist auch mit diesem Verfahren das Ende des Möglichen erreicht, und es liefert sein Ergebnis zurück. Diese Methode zeigt in der Praxis gute Leistungen als Ersatz für He's Implementierung für die Fälle, wo letztere aufgeben musste.

Das zweite Problem während des Trainings[13] eines GMM-IB liegt in dem Auftreten von Divisionen durch null verborgen: Auch ohne das Auftreten numerischer Schwierigkeiten nehmen die Dichte- und Verteilungsfunktionen manchmal völlig zu Recht den Wert null an. Dies führt dann in Gleichungen, wo sie im Nenner stehen[14], zu einer Division durch null, und mit dem resultierenden Ergebnis[15] kann nicht weiter gerechnet werden. Nach den Angaben in [TAND 03] wurde deshalb eine Funktion entwickelt[16], diese Werte zu entdecken.

Sind die Fehlerwerte aufgespürt, muss ihnen ein sinnvoller Wert für die Weiterverarbeitung gegeben werden. Welcher das genau ist, ist später eigentlich nicht mehr relevant, solange er endlich ist: Gilt $A_j(z_t) = 0$ oder $B_i(z_t) = 0$, wird auch die Wahrscheinlichkeit für das Auftreten der Beobachtung unter der Annahme dieser Komponente $p(z|i_t = i, j_t = j, \lambda) = 0$ (Gleichung 3.2). So werden die bedingten Erwartungswerte unabhängig von ihrem tatsächlichen Wert (solange er endlich ist) bei ihrem Eingang in die Parameterneuermittlung mit dem Faktor Null skaliert, da auch Formel 2.16 den Wert null annimmt.

Abgesehen davon ist es trotzdem interessant[17] zu beobachten, wie siech eine solche Null-durch-Null-Situation auch mathematisch lösen ließe: Angenommen, es gelte $\frac{b(z)}{B(z)} = \frac{0}{0}$. Nun kann nach der Regel von de l'Hospital folgendes aufgestellt werden:

[13] Hier gilt wie übrigens auch für die obigen Bemerkungen auch, dass sie sich ebenfalls auf den Test-Algorithmus beziehen.

[14] Es gibt drei solcher Fälle: $p(x = z|\ldots)$ (Gleichung 3.3), $E\{x|x < z, \ldots\}$ (Gleichung 3.6) und $E\{x^2|x < z, \ldots\}$ (Gleichung 3.7).

[15] C++ hat die Eigenart, in diesem Fall keine Ausnahme zu werfen, sondern dem Ergebnis einen bestimmten Fehlerwert zu geben: `1.INF` deutet an, dass der Wert nicht endlich ist. Wird trotzdem weiter mit ihm gerechnet, muss zwangsläufig jedes Ergebnis, in welches er eingeht, auch unendlich werden.

[16] Es existiert unter dem Microsoft Visual C Compiler zwar die Funktion `finite()`, doch ist diese nicht ANSI-C-konform.

[17] Jedoch wegen obiger Betrachtung nicht notwendig.

$$\frac{b(z)}{B(z)} = \frac{b(z)'}{B(z)'} = \frac{\frac{\sqrt{2}\cdot e^{-\frac{(z-\mu)^2}{2\cdot\sigma^2}}\cdot(\mu-z)}{2\cdot\sqrt{\pi}\cdot\sigma^2}}{\frac{\sqrt{2}\cdot e^{-\frac{(z-\mu)^2}{2\cdot\sigma^3}}}{2\cdot\sqrt{\pi}\cdot\sigma}} = \frac{\mu - z}{\sigma^2} \tag{4.2}$$

Die bedingten Erwartungswerte ließen sich so auch für diesen Fall analytisch exakt bestimmen: Unter obiger Annahme würde $E\{x|x<z,\ldots\} = z$ und $E\{x^2|x<z,\ldots\} = \sigma^2+z^2$. Ist die Modellkomponente also vollständig unpassend[18], tendiert der Erwartungswert der reinen Beobachtung unter der Bedingung, dass sie durch Nebengeräusche überlagert ist, gegen die Beobachtung selbst. Das ist anschaulich sofort klar, da keine vernünftige andere Annahme getroffen werden kann: Der Wert muss irgendwo anders liegen, doch das Modell erlaubt keine Rückschlüsse auf den tatsächlichen Ort. Deshalb wird die einzige Konstante, die Beobachtung an sich, als Wert gewählt, doch da sie mit Sicherheit unzuverlässig ist, wird sie durch den im vorigen Absatz beschriebenen Mechanismus keinen Eingang in die neuen Parameter finden.

4.1.4 Zur Bedienung

Bevor es nun an die Präsentation der Ergebnisse geht, sei hier noch eine kurze Einführung in die Benutzung von SCiVo sowie eine Beschreibung der vom Benutzer spezifizierbaren Parameter der SC_Lib gegeben. Sie ist notwendig, um die anschließend erläuterten Experimente nachvollziehen zu können.

SCiVo benötigt zum Starten mindestens vier Übergabeparameter: Die Pfade zu der zu analysierenden Audiodatei, zu deren Segmentierungsliste[19] und deren Szenenliste[20]. Der vierte unbedingt erforderliche Parameter ist ein `double`-Wert, welcher die Rahmenrate des zugrundeliegenden Videos beschreibt. Diese ist nötig, um die Umrechnungen zwischen Video- und Audiorahmen korrekt durchführen zu können.

Nun folgen 35 optionale Parameter, die das Verhalten des Programms teils stark beeinflussen. Sie können jedoch auch von hinten beginnend zusammenhängend weggelassen werden, um in diesem Fall durch Standardwerte ersetzt zu werden. Im Einzelnen sind dies:

1. `firstScene`: Die erste zu bearbeitende Szene in der angegebenen Audiodatei.

[18] D.h. $b(z) = B(z) = 0$.

[19] Die *Segmentierungsliste* ist eine Textdatei, welche die beschriebene Audiodatei in Segmente von Sprache und Sonstigem aufteilt. Sie muss dabei folgendem Format genügen: Ein `'#'` leitet eine Kommentarzeile ein, diese wird nicht beachtet. Jede andere Zeile enthält zu Anfang eine Zahl, welche die Videorahmennummer darstellt, bei der das folgende Segment beginnt (es endet automatisch einen Rahmen vor Beginn des nächsten). Durch ein Tabulator-Zeichen getrennt von dieser Nummer steht eine Null oder Eins, wobei die Null das beginnende Segment als Sprache enthaltend auszeichnet, eine Eins hingegen als Sonstiges.

[20] Die *Szenenliste* ist prinzipiell ähnlich aufgebaut wie die Segmentierungsliste, jedoch enthalten die einzelnen Zeilen nur eine Videorahmennummer. Diese gibt jeweils den ersten Rahmen einer neuen Szene an.

2. `lastScene`: Die letzte zu bearbeitende Szene. Auf diese Weise kann eine Datei auch nur teilweise analysiert werden.

3. `sceneSelection`: Hier kann mittels Bitflags angegeben werden, welche der Szenen zwischen `first-` und `lastScene` tatsächlich analysiert werden sollen, falls Auslassungen gewünscht sind. Die Nummer des gesetzten Bits korrespondiert dabei, startend bei eins, mit der auszuwählenden Szenennummer[21]. Ist der Parameter gleich null, wird er nicht beachtet.

4. `phaseSelect`: Hier können (ebenso mittels Bitflags) verschiedene Vorverarbeitungs- und Filterungsschritte an- und abgewählt werden: Bit wins korrespondiert mit der Trennung von Sprache und Nebengeräuschen, Bit zwei mit der Markierung nichtstimmhafter Rahmen. Bit drei schließlich steht für die Sprecherwechsel-Erkennung.

5. `audioFrameSize`: Wie groß sollen die zu erstellenden Audiorahmen sein (in Millisekunden)?

6. `audioFrameStep`: Angabe über den Abstand in Millisekunden, den zwei aufeinander folgende Rahmen haben sollen. Hiermit kann eine Überlappung der Rahmen spezifiziert werden.

7. `energyQuantizationLevel`: Parameter während der Sprache-Geräusch-Trennung und der Markierung nicht-stimmhafter Sprache: Anzahl der Quantisierungsstufen.

8. `pauseSilenceThreshold`: Wie viele Millisekunden Länge muss ein zusammenhängendes nichtsprachliches Teilstück innerhalb eines Sprachsegments vorweisen, um letzteres in zwei einzelne Segmente aufzuteilen?

9. `filterBankSize`: Die Anzahl der Filter in der Filterbank zur Bestimmung der MFCC's.

10. `FFTsize`: Die Anzahl der Signal-Messwerte, die zur Durchführung der FFT herangezogen werden. Der Wert muss einer Zweierpotenz entsprechen, wobei die SC_Lib automatisch Zeropadding[22] anwendet, falls die Anzahl zur Verfügung stehender Rahmen zu gering sein sollte.

11. `dEnergy`: Soll anstatt des ersten MFC-Koeffizienten dessen erste zeitliche Ableitung gespeichert werden? Ein Bool'scher Wert.

12. `windowed`: Soll eine Fensterfunktion zur Signalverarbeitung benutzt werden? Ebenfalls Bool'sch.

[21] Konkret wählt also beispielsweise die Binärzahl `11010` die Szenen eins, drei und vier zur Bearbeitung aus, da die Bits eins, drei und vier den Wert eins haben. Die einzelnen Bits sind also von null beginnend durchnummeriert, und Bit null kann, da es nicht ausgewertet wird, einen beliebigen Wert zugewiesen bekommen. Dieses Prinzip gilt auch für alle weiteren Verwendungen von Bitflags in den folgenden Parametern.

[22] Mit *Zeropadding* wird das Auffüllen fehlender Messwerte durch Nullen bezeichnet.

13. `preEpmhasizeFactor`: Ein `double`-Wert zur Bestimmung des Vorverstärkungsfaktors vor der Signalverarbeitung.

14. `MFCCorder`: Die Anzahl der MFC-Koeffizienten.

15. `MFCCcoeffSelection`: Bitflags zur Selektion nur bestimmter MFCC's. Startend bei eins stehen die Bits für die jeweilige Koeffizientennummer.

16. `CMN`: Ein Bool'scher Wert, welcher die Anwendung der Cepstralen Mittelwert-Normalisierung steuert.

17. `lowCut`: Definiert einen Hochpass-Filter auf das Signal vor Beginn der Merkmalsextraktion: Tiefere Frequenzen werden ausgefiltert.

18. `highCut`: Definiert entsprechend einen Tiefpass-Filter. Die Hoch- und Tiefpass-Filterung wird als Ganzes nur durchgeführt, wenn sowohl $lowCut > 0$ als auch $highCut < f_{max}$ gilt. Die Maximalfrequenz f_{max} ist dabei bestimmt durch die halbe Abtastfrequenz.

19. `segmentLengthThreshold`: Sprachsegmente von geringerer Länge (in Millisekunden)werden von vorne herein vom Analysevorgang ausgeschlossen.

20. `lastNdistances`: Anzahl zurückliegender Werte, aus denen der adaptive Schwellwert während der Sprecherwechsel-Erkennung wird.

21. `adaptiveThresholdAlpha`: Der Verstärkungsfaktor für den Mittelwert der N zurückliegenden Distanzwerte.

22. `percentDifference`: Wie unterschiedlich (in Prozent) müssen zwei Kovarianzmatrizen während der Sprecherwechsel-Erkennung mindestens sein, damit die aktuelle Modellkomponente des qGMM weiter verfeinert wird?

23. `deltaBIClambda`: Straffaktor für die Komplexität der Modelle in der ΔBIC-Implementierung des qGMM.

24. `maxMixtures`: Die maximale Anzahl Modellkomponenten im qGMM.

25. `varianceLimit`: Ein minimaler Wert für die Varianzen im GMM und GMM-IB, wie er in [REYN 95] vorgeschlagen wird: Niedrigere Werte werden künstlich auf diesen Wert heraufgesetzt, um Singularitäten zu vermeiden.

26. `EMthreshold`: Differenz der Log-Likelihood-Werte von zwei aufeinander folgenden Durchläufen des EM-Algorithmus(während des GMM- oder GMM-IB-Trainings), die unterschritten werden muss, um das Verfahren Konvergenz annehemend zu beenden.

27. `maxNoiseModelOrder`: Maximale Anzahl an Komponenten für ein Hintergrundmodell während des normalen Trainings.

28. `maxSpeakerModelOrder`: Maximale Anzahl an Komponenten für ein Sprechermodell während des normalen Trainings (während der Clusterverschmelzung kann dieser Wert überschritten werden).

29. `vectorsPerGaussian`: Anzahl Merkmalsvektoren, die mindestens erreicht sein muss, damit ein GMM oder GMM-IB eine weitere Komponente erhält. Sind weniger Merkmalsvektoren verfügbar, hat das Modell null Komponenten, das Segment ist also zu kurz zur Analyse.

30. `distanceMeasure`: Hier kann das Distanzmaß für den Klassifikationsprozess gewählt werden: Null für die CLR, Eins für die GLR.

31. `terminationCriterion`: Hier kann die Abbruchbedingung für den Klassifikationsprozess gewählt werden: Das BIC wird mit der Null gewählt, eine Eins steht für "immer verschmelzen". Am Ende wird es also nur einen großen Cluster geben, aber es wird sichtbar sein, welche Cluster wann mit welcher Sicherheit verschmolzen wurden. Eine Zwei steht für "nie verschmelzen", der Clusteringprozess wird damit also ausgeschaltet.

32. `debugMode`: Hier können wiederum mittels Bitflags verschiedene Stufen der Informationsvermittlung zugeschaltet werden:

    - Bit Eins führt zur Erstellung der Datei `speakerModels.txt`: Plots sämtlicher Parameter der erstellten Sprechermodelle.
    - Bit zwei leistet das Obige für alle Hintergrundmodelle, die Ausgabedatei heißt `noiseModels.txt`.
    - Bit drei steuert die Ausgabe zusätzlicher Informationen über den Ablauf des Klassifikationsprozesses: `dist.txt` und `BIC.txt` enthalten die Werte der Distanzmatrix und der BIC-Entscheidungen.
    - Bit vier erstellt eine Datei, die Einblick in den gesamten Trainingsablauf aller GMM-IB bietet: `gmm_ib.txt` enthält sämtliche Modellkomponenten in jeder Runde des EM-Algorithmus, `gmm_ib_likelihood.txt` die dazugehörigen Likelihood-Werte der Daten.
    - Bit fünf leistet identisches für GMM's: `gmm.txt` und `gmm_likelihood.txt`.
    - Bit sechs führt dazu, dass zusätzlich zu den Endergebnissen die Informationen über alle Audiorahmen in der Datei `frameList.txt` ausgegeben werden.
    - Das siebte Bit steuert die Ausgabe hörbarere Ergebnisse: Ist es gesetzt, werden die zum Sprachmodell-Training benutzten Rahmen als `speech_XXXX.wav`, die zum Training des Geräuschmodells als `noise_XXXX.wav` und alle Rahmen eines Segments gemeinsam als `complete_XXXX.wav` ausgegeben. `XXXX` steht hierbei für die aktuelle Szenen- und Segmentnummer, wie sie auch in die Sprecher-ID's Eingang findet.
    - Das letzte Bit schließlich schaltet die Ausgabe der Datei `segStat.txt`: Statistiken über die Brutto- und Nettolänge von Sprachsegmenten vor und nach der Reinigung von "Unrat" wie beispielsweise nicht-stimmhafter Sprache.

33. `debugDir`: Spezifiziert den Pfad, in welchem obige Informationen abgelegt werden sollen.

34. `speechSegLength`: Zu Testzwecken kann hier die genaue Länge (in Audiorahmen) für die zu bearbeitenden Sprachsegmente festgelegt werden: Längere Segmente werden zerschnitten, kürzere von der weiteren Bearbeitung ausgeschlossen. Eine Null führt zur Verwendung der Originallängen.

35. `explicitModelFile`: Hier kann der Pfad zu einer Datei angegeben werden, welche eine Liste von Dateinamen explizit extern erstellter Hintergrundmodelle für bestimmte Szenen und Segmente enthält. Jede Zeile dieser Datei muss zuerst die Szenennummer, darauf folgend die Segmentnummer und schließlich den Pfad der Datei enthalten. Eine Null anstelle der wirklichen Szenen- oder Segmentnummer wirkt dabei als Joker: Das Modell wird dann für alle Segmente bzw. Szenen benutzt.

Eine ähnliche Liste wird bei Fehlbedienung des Programms automatisch sichtbar, so dass der Anwender jederzeit über die Bedienung informiert werden kann. Wurde SCiVo nun entsprechend mit Daten gefüttert, sieht ein typischer Programmablauf für den Anwender etwa folgendermaßen aus:

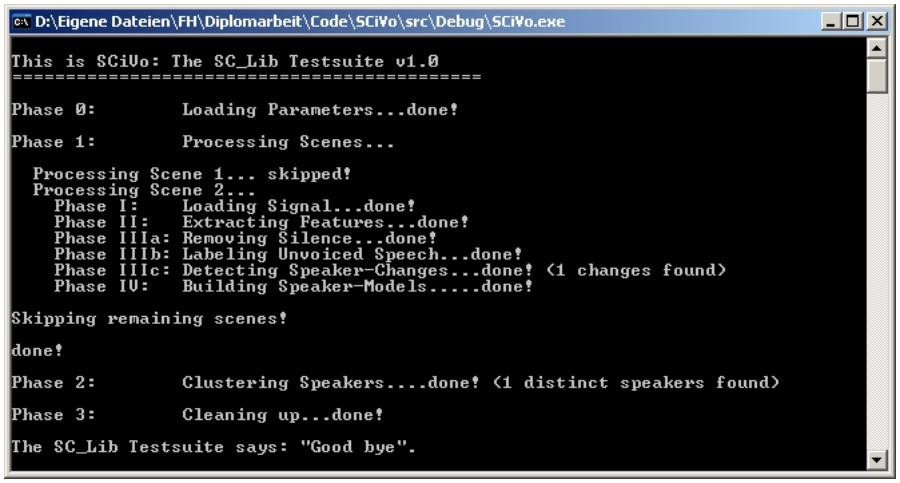

Abbildung 4.2: Bildschirmausgabe von SCiVo während eines Klassifikationsdurchlaufs.

Am Ende steht[23] ein Ablaufprotokoll, welches exemplarisch in Abbildung B.2 dargestellt ist. In Anhang B findet sich auch die Lösung für das Problem, diese doch recht kryptische Programmschnittstelle bedienbar zu machen: Zum Zwecke nachfolgender Experimente wurde der Prototyp einer grafischen Benutzeroberfläche geschaffen, welcher – im Vergleich zur nackten Kommandozeile – einige Komfortfunktionen[24] bietet.

[23] Abgesehen von den zusätzlich schaltbaren Debug-Ausgaben.
[24] Beispielsweise die Erläuterung obiger Parameter inklusive sinnvoller Standardwerte.

4.2 Experimente

4.2.1 Testphilosophie

In den vorangegangenen Kapiteln und Abschnitten dieser Arbeit wurde der Entwurf und die anschließende Entwicklung der zu erstellenden Software eingehend beschrieben: Zum Zwecke der Sprecherklassifikation, der Erstellung eines Index für das Auftreten von Sprechern in Videos, wurde eine Klassenbibliothek zur Bereitstellung der entsprechenden Funktionalität, SV_Lib genannt, erstellt. Zu Demonstrations- und Testzwecken wurde weiterhin das Rahmenprogramm SCiVo geschrieben, welches die Funktionen der SV_Lib kapselt und einem Endanwender zur Verfügung stellt. Um die nachfolgenden Experimente bedienungsseitig komfortabler zu gestalten, wurde weiterhin der Prototyp einer grafischen Oberfläche für SCiVo entworfen, welche den Namen "Gardener" trägt.

Ziel all dieser Bemühungen war es, für die Klassifikationsaufgabe möglichst Unabhängigkeit gegenüber

- eventuell vorhandenen Nebengeräuschen und variablen Aufnahmebedingungen,
- der Länge der Testsegmente und
- der Anzahl und Identität der Sprecher

zu erlangen[25]. Der Grad des Erreichens dieser Ziele soll nun anhand folgender mehrstufiger Vorgehensweise überprüft werden:

Erste Testphase: Zu Beginn sollen die in der Klasse `SC_TweakableParameters` zusammengefassten Parameter, welche nicht direkt mit den oben genannten Zielen korrespondieren, auf ihren Einfluss auf die Gesamtleistung des Systems hin untersucht werden. Dies sind vor allem die Schrauben, welche zum Getriebe der Signalverarbeitung und Merkmalsextraktion gehören. Ziel dieser Phase ist dabei, ein möglichst optimales Basissystem zu finden, aufgrund dessen mit den eigentlichen Experimenten für die Parameter der neu entwickelten Verfahren begonnen werden kann.

In dieser Stufe geht es also speziell um neun Parameter, welche in verschiedenen Kombinationen zueinander getestet werden müssten. Um hier den Aufwand gering zu halten, wird jedoch folgende Verfahrensweise eingeführt: Die Werte von `audioFrameSize`, `AudioFrameStep`, `filterBankSize` und `FFTsize` stehen in enger Wechselwirkung zueinander und müssen sicherlich als Ganzes betrachtet werden. Ähnlich verhält es sich mit den Parametergruppen `MFCCorder` und `dEnergy` sowie mit `MFCCcoeffSelection`, `lowCut` und `highCut`. Hier kann jeweils innerhalb der Gruppe der beste Parametersatz ermittelt werden, welcher dann wiederum als Basissystem für den nächsten Gruppentest dienen mag[26].

[25] Vergleiche hierzu auch Abschnitt 1.2.

[26] Natürlich sind die Parameter der einzelnen Gruppen nicht völlig unabhängig voneinander. Verdeutlicht man sich jedoch, dass diejenigen der ersten Gruppe ganz stark mit der reinen Signalverarbeitung

Keine Erwähnung finden hier `windowed` und `preEmphasizeFactor`, denn Anwendung einer Fenstertechnik und Vorverstärkung des Signals sind in der einschlägigen Fachliteratur so weit verbreitet, dass eine erneute Untersuchung hier nicht notwendig erscheint. Aufgrund der Diskussion in Abschnitt 2.4.5 wird auch der Einfluss von `CMN` nicht weiter untersucht.

Zweite Testphase: Ausgehend von dem eben bestimmten Basissystem sollen die optimalen Parameter eines schon etwas mehr zum eigentlichen Kern dieser Arbeit gehörenden Vorverarbeitungsschrittes ermittelt werden: Diejenigen der Sprecherwechsel-Erkennung. Die Verfahren zur Sprachfilterung mit ihren gewählten Parametern hingegen funktionieren im Betrieb auch anschaulich[27] so gut, dass sie wohl keiner weiteren Tests mehr bedürfen. Im Detail sind also die Werte von `lastNdistances` und `adaptiveThresholdAlpha` sowie `percentDifference` und `deltaBIClambda` zu überprüfen, wiederum jeweils getrennt in zwei Gruppen. Denn während die ersten beiden Werte der Vorentscheidung über einen potenziellen Sprecherwechsel zuarbeiten, gehören letztere in den Bereich der Überprüfung der dort aufgestellten Hypothesen, in den Bereich des qGMM's. Eine Trennung macht also auch hier Sinn.

An dieser Stelle existieren nun ebenfalls Parameter, die nicht notwendigerweise getestet werden müssen: `pauseSilenceThreshold` wird so festgesetzt, dass es zu keiner weiteren Aufteilung der Ursprungssegmente kommt, da bei der Erstellung der Segmentierungslisten per Hand darauf geachtet wurde, jeweils sprecherhomogene Segmente zu markieren. Und `segmentLengthThreshold` wird auf den Wert null eingestellt, damit jedes noch so kurze Segment die Chance erhält, von der Sprecherwechsel-Erkennung mit einem Nachbarn vereint zu werden, um so noch Eingang in den Klassifikationsprozess zu finden. Die Parameter `lastNdistances` und `maxMixtures` schließlich können als relativ unabhängig vom Gesamtergebnis angesehen werden und bekommen daher auch feste Werte zugeteilt.

Dritte Testphase: Hier geht es um die Wirksamkeit des entwickelten Prozesses zur Sprachanreicherung und Klassifikation. Anhand des zuvor aufgebauten Basissystems soll dessen Leistungsfähigkeit im Vergleich zu einem System gezeigt werden, welches ohne Geräuschmodellierung rein auf Basis von GMM's arbeitet. Dies geschieht mittels der Parameter `EMthreshold`, `maxNoiseModelOrder`, `maxSpeakerModelOrder`, `vectorsPerGaussian` und `distanceMeasure` zuerst anhand synthetisch angereicherter Daten, bevor die Systeme ihr Können unter realen Bedingungen anhand der im nächsten Abschnitt beschriebenen Videos darstellen dürfen. Daneben sollen die Tests noch Einsicht in das Laufzeitverhalten der Algorithmen bieten.

zusammenhängen, während die folgenden zur Merkmalsextraktion gehören sowie die letzten schließlich zur Merkmalsselektion, kann die vorgeschlagene paketweise Verarbeitung anschaulich nachvollzogen werden.

[27] Durch Anhören der bei Wahl des enstrechenden Debug-Modus erstellten Audiodateien kann auf eine sehr gute Eliminierung störender Einflüsse geschlossen werden.

Bewertungsmaße

Wichtig ist nun noch die Einführung der zur Bewertung herangezogenen Maße: Je nach Testphase können hier unterschiedliche Kombinationen auftreten, doch insgesamt kann aus folgendem Fundus geschöpft werden:

Recall und *Precision*: In der Literatur wird zur Beschreibung der Leistungsfähigkeit eines binären Klassifikationsprozesses beinahe ausschließlich das Tupel Recall und Precision eingesetzt: Ersterer Wert beschreibt anschaulich die Fähigkeit des Verfahrens, *alle relevanten* Ergebnisse zu präsentieren, während letzterer ein Maß ist für die Eigenschaft des Algorithmus, *ausschließlich relevante* Resultate anzugeben [VOOR 02]. Diese Methodik wird im weiteren Verlauf zur Beschreibung der Leistungsfähigkeit des Sprecherwechsel-Erkenners eingesetzt.

Erkennung und *Genauigkeit*: Eine ähnliche Aussage wie diejenige der Werte Recall und Precision über die Güte des gesamten Klassifikationsprozesses wäre natürlich auch wünschenswert, jedoch passen deren Definitionen ob der Beschränkung ihres Einsatzbereichs auf rein binären Problemstellungen nicht direkt zu der hier behandelten Problemklasse. Deshalb wurde im Folgenden eine leichte Umdefinition und, um Verwechslungen vorzubeugen, eine nur sinngemäße Übertragung der Namen beider Maße ins Deutsche vorgenommen, welche die eigentliche Aussage unter Berücksichtigung der veränderten Rahmenbedingungen auf die neue Domäne portiert. Zunächst seien jedoch noch einige wichtige Festlegungen genannt, um die nachfolgende Definition auf einen festen Grund zu stellen:

Als Atom der Messung dient ein einzelnes Sprachsegment, so wie es ursprünglich in den Segmentierungslisten angegeben ist. Ein Cluster ist im Folgenden definiert als die Menge von einem oder mehreren Segmenten, welche von der Klassifikation als von demselben Sprecher geäußert markiert wurden. Die Identität eines Clusters ist bestimmt durch die größte Anzahl sprecherhomogener Segmente in ihm: Enthält ein Cluster beispielsweise drei Segmente von Sprecher A und nur zwei Segmente von Sprecher B, so gilt der Cluster als den Sprecher A modellierend, die zwei Segmente von B wurden also falsch zugeordnet. Weiterhin gilt in dem Fall, dass mehrere Cluster für einen einzigen Sprecher existieren, dass nur die Segmente desjenigen Clusters, welcher die größte Anzahl Segmente repräsentiert, als *richtig* klassifiziert angesehen werden. Alle Segmente desselben Sprechers in anderen Clustern gelten als falsch zugeordnet, da sie dem größten hätten zugeteilt werden müssen.

Somit folgt für die Erkennung: Sie ist definiert als das Verhältnis der Anzahl dem *richtigen* Cluster zugeordneter Segmente zu der Anzahl insgesamt verfügbarer Sprachsegmente. Die Genauigkeit hingegen ergibt sich aus dem Verhältnis der Anzahl zur Identität ihrer Cluster passender Segmente zu der Gesamtanzahl in Clustern befindlicher Segmente[28].

[28] Hier wird das fälschliche Aufteilen der Segmente eines Sprechers auf mehrere Cluster also nicht

$$\text{Erkennung} = \frac{\text{Anz. } \textit{richtig} \text{ zugeordneter Segmente}}{\text{Anz. insgesamt verfügbarer Sprachsegmente}} \tag{4.3}$$

$$\text{Genauigkeit} = \frac{\text{Anz. zur Identität ihres Clusters passender Segmente}}{\text{Anz. insgesamt geclusterter Segmente}} \tag{4.4}$$

Anzahl Cluster: Die richtige Anzahl Cluster fließt zwar auch in die Berechnung der Erkennung mit ein, gibt jedoch einzeln aufgeführt noch einmal einen Eindruck der Leistungsfähigkeit eines bestimmten Systems.

Sicherheit: Für den Fall, dass zwei konkurrierende Parametersätze über identisch gute Clusteranzahl-, Erkennungs- und Genauigkeits-Werte verfügen, muss noch auf ein weiteres Maß zurückgegriffen werden können, um die feinen Unterschiede besser herauszuarbeiten. Hierzu bietet sich der erste negative Wert[29] des ΔBIC während des Clusteringprozesses an: Je weiter er unter null liegt, desto schlechter würden nach der Verschmelzung die Daten durch die angestrebte Partitionierung beschrieben werden. Andersherum kann man auch sagen, dass, je negativer dieser Wert ist, sich das System um so sicherer ist, an genau dieser Stelle des Prozesses einen Schnitt zu machen. Liegt der Wert nur ganz knapp unter null, würde eine Vereinigung der Cluster also nur geringen Schaden anrichten, und ein kleines "Zünglein an der Waage"[30] hätte diese Entscheidung vielleicht noch kippen können. Der letzte Wert des ΔBIC wird also im Folgenden mit dem Begriff Sicherheit belegt und als Detailmaß verwendet:

$$\text{Sicherheit} = \underset{\Delta BIC<0}{\arg\max}(\Delta BIC) \tag{4.5}$$

Sprecherwechsel: Ähnlich wie die Anzahl Cluster zur Bewertung der Gesamtleistung sinnvoll ist, wirkt die Anzahl detektierter Sprecherwechsel als Indikator für die Leistungsfähigkeit der Vorverarbeitungs-Stufen.

Verschmelzungsweg: Da aufgrund der starken Nebengeräusche in den späteren Experimenten damit zu rechnen ist, dass die Klassifikation zu früh mit der Clusterverschmelzung aufhört, wird dort die Abbruchbedingung des Clusterings teilweise abgeschaltet, so dass am Ende immer ein einziger großer Cluster für sämtliche Sprachsegmente entsteht. Natürlich kann anhand der ΔBIC-Werte trotzdem abgelesen werden, wo das Verfahren normalerweise einen Schnitt angesetzt hätte, doch nun kann zusätzlich folgender interessanten

nochmals bestraft, da sich dies ja schon in einer niedrigeren Erkennungs-Rate niederschlägt. Vielmehr werden inhomogene Cluster schlecht bewertet.

[29] Zur Erinnerung: Wenn das ΔBIC einen Wert < 0 zurückliefert, würde eine Clusterverschmelzung die Gesamtwahrscheinlichkeit, dass die gegebenen Daten von den erstellten Modellen beschrieben werden, verringern. Der Klassifikationsprozess endet deshalb an dieser Stelle und liefert die aktuelle Anzahl Cluster ohne diese letzte Verschmelzung zurück.

[30] Eventuell ein anderes gesprochenes Wort im untersuchten Audiosegment, welches die Äußerungen vom Laut her ähnlicher hätte erscheinen lassen, oder etwas weniger Nebengeräusch.

Fragestellung nachgegangen werden: Was wäre, wenn das Terminations-Kriterium aus irgendeinem Grund die korrekte AnzahlSprecher wüsste und trotz negativen ΔBIC-Werten fortfahren würde? Würden im Folgenden zuerst alle Segmente identischer Sprecher in gemeinsamen Clustern vereint, bevor der erste nicht-sprecherhomogene Cluster entstünde? Der Verschmelzungsweg ist nun das Verhältnis der Anzahl anhand obiger Fragestellung korrekt ausgeführter Verschmelzungen zu der Gesamtanzahl der Verschmelzungen, die notwendig ist, um einen einzigen großen Cluster zu bilden. Ein Wert von eins bedeutet hier also, dass zuerst alle wirklich zusammengehörigen Segmente in ihre jeweiligen Cluster unterteilt wurden, bevor unterschiedliche Sprecher sich einen Cluster teilten. Umgekehrt weist ein niedriger Wert darauf hin, dass Segmente unterschiedlicher Sprecher in einem gemeinsamen Cluster landeten, noch bevor alle Segmente eines Sprechers beisammen waren.

4.2.2 Datenmaterial

Zu Testzwecken wurden verschiedene Arten von Videos ausgewählt, anhand derer die spezifischen Eigenschaften des Systems aufgezeigt werden sollen. Dabei wurden jeweils nur bestimmte Szenen der einzelnen Videos anhand der Aussage, die durch sie getroffen werden könnte, ausgewählt, um die mühsame Erstellung der Segmentierungslisten per Hand nicht über die Maßen zu beanspruchen: Enthält eine Szene beispielsweise einen längeren Dialog ohne Nebengeräusche, kann genau dieser Fall mit ihrer Hilfe untersucht werden. Er muss dann nicht zwingend noch einmal anhand eines anderen Videos aufgegriffen werden.

Ein großer Teil der Videos entstammt dabei dem MPEG-7-Testset, um Vergleichbarkeit zu verwandten Arbeiten herzustellen: Dadurch, dass es weltweit verfügbar und standardisiert ist, können die mit Hilfe der MPEG-7-Daten erlangten Ergebnisse dieser Arbeit zu anderen Publikationen in Relation gesetzt werden. Sie spielen also vor allem in den letzten Tests der dritten Phase eine Rolle, wo es um die endgültige Leistungsfähigkeit des Systems geht. Bis dahin wird hauptsächlich die Datei `news2.wav` als Testmaterial herangezogen, da sie als Nachrichtensendung ein typisches und wohlwollendes Umfeld für die Sprecherklassifikation bildet und darüber hinaus leicht mit synthetischen Daten angereichert werden kann, um andere, determinierte akustische Bedingungen herzustellen.

Bevor nun auf die einzelnen Videos und ihre Szenen eingegangen wird, erfolgt noch eine kurze Beschreibung der notwendigen Vorverarbeitungsschritte, denen sich die einzelnen Kandidaten unterziehen mussten: Zuerst stand die Extraktion des Audiostroms aus dem Video auf dem Programm: Da die SV_Lib nicht direkt mit Videodaten umgehen kann (und dies auch gar nicht zu können braucht), wurde die jeweilige Tonspur separat und unkomprimiert in einer Microsoft-RIFF-WAVE-Datei abgelegt. Diese wurde vor der Verwendung noch folgendermaßen bearbeitet, um gleiche Voraussetzungen unter allen Probanden zu schaffen und einen gewissen Standard einzuführen, von dem die SV_Lib ausgehen kann:

- Reduktion auf einen einzigen Kanal (Mono) durch Zusammenmischen eventuell vor-

handener Stereokanäle zu je 50 Prozent.

- Bandbegrenzung durch eine Tiefpassfilterung auf den Bereich 0-8000Hz. Dies ist der Frequenzbereich der Sprache, darüber sind ausschließlich Nebengeräusche zu erwarten, so dass diese Filterung (einmal abgesehen von der Verhinderung des Aliasing-Effekts[31] bei der nachfolgenden Konvertierung der Abtastrate) zur Verminderung der Umgebungsgeräusche beiträgt.
- Konvertierung der Abtastfrequenz auf 16.000Hz.

Als Endergebnis liegt schließlich eine WAV-Datei im Format 16kHz/Mono vor. Tabelle 4.1 gibt Aufschluss über die in dieser Form verwendeten Daten, unterstützt durch die folgenden kurzen Beschreibungen ihrer spezifischen Eigenschaften.

Name	Typ	Szene(n)
N-TV Nachrichten (news2.wav)	Nachrichtensendung (Deutsch)	1-7 04:03 min
23 - Nichts ist wie es scheint (23.wav)	Kinofilm (Deutsch)	1-12 10:07 min
MPEG-7: Hallo (hallo.wav)	Schauspiel (Deutsch)	2 01:01 min
MPEG-7: Docon (docon.wav)	Zeichentrick (Spanisch)	5 01:46 min
MPEG-7: Lanc (lanc.wav)	Reportage (Englisch)	1-5 01:15 min
MPEG-7: NHK Video (nhkvideo.wav)	Reportage (Asiatisch)	4 01:16 min
MPEG-7: Camilo e Filho (camiloefilho.wav)	Comedy-Serie (Portugiesisch)	2 05:38 min

Tabelle 4.1: Zusammenstellung der verwendeten Testvideos und ihrer Eigenschaften. Die Nummerierung der Szenen bezieht sich dabei auf die Einteilung durch die per Hand erstellten Szenensegmentierungs-Listen.

`news2.wav` stellt wie gesagt den Standardfall dar: Lange Sprachsegmente, wenige Sprecher, klar strukturierter Aufbau und wenige Nebengeräusche wurden schon in vielen anderen Arbeiten auf dem Gebiet der Sprecherklassifikation anhand von Nachrichtensendungen getestet und zeichnen auch diese Produktion aus. Dieser Fall soll also zum Ermitteln des Basissystems dienen. Die schon angesprochenen synthetischen Anreicherungen beschreiben dabei diverse Mischungen des nebengeräuschfreien Ausgangssignals des Videos mit generiertem Weißen, Braunen oder Pinken Rauschen. So werden Untersuchungen über die Tauglichkeit des GMM-IB in unterschiedlichen, aber bestimmten[32] SNR-Situationen möglich. Tabelle 4.2 gibt einen Überblick über die erstellten Varianten. Weitere Änderungen

[31] Die Abtastfrequenz muss mindestens doppelt so hoch sein wie die höchste im Signal vorhandene Frequenz, sonst werden höhere Frequenzen nicht korrekt abgebildet und erscheinen im Spektrum fälschlicherweise im Bereich niederer Frequenzen. Dieser Effekt nennt sich *Aliasing*.

[32] Die *Bestimmtheit* des Nebengeräusches ist in diesem Fall der eigentliche Vorteil: Verschiedenste akustische Hintergründe finden sich auch direkt in den anderen Testdaten. Nun besteht jedoch Kontrolle

am Ausgangsmaterial stellen Modifikationen der Ursprungssegmentlänge dar: Originär lange Sprachabschnitte einer Person werden künstlich in Bereiche weniger Sekunden unterteilt, um daran die Tauglichkeit der Verfahren für kurze Segmente zu testen.

Name	Szene 3	Szene 5	Szene 6	Szene 7
news2 brown4	geringes Braunes Rauschen	-	-	-
news2 brown4_3rdBrown4	geringes Braunes Rauschen	-	geringes Braunes Rauschen	-
news2 brown8	mittleres Braunes Rauschen	-	-	-
news2 brown8_3rdBrown8	mittleres Braunes Rauschen	-	mittleres Braunes Rauschen	-
news2 brown16	starkes Braunes Rauschen	-	-	-
news2 brown16_3rdBrown16	starkes Braunes Rauschen	-	starkes Braunes Rauschen	-
news2 pink1_10percent	mittleres Pinkes Rauschen	-	-	-
news2 white1	starkes Weißes Rauschen	-	-	-
news2 white1_50percent_brown4	starkes Weißes Rauschen	-	-	schwaches Braunes Rauschen

Tabelle 4.2: Varianten des news2-Videos. Angegeben ist für die untersuchten Szenen, mit welcher Art künstlichem Nebengeräusch sie angereichert wurden. Von oben nach unten ergibt sich wahrnehmbar eine immer stärkere Verunreinigung des Sprachsignals.

Das Video enthält insgesamt drei unterschiedliche Sprecher, wobei die Szenen für die anfänglichen Testphasen so gewählt wurden, dass hier jeweils auch schon zwei unterschiedliche Sprecher auftreten.

`23.wav` ist ein typischer Kino-Spielfilm: Viele akustische Effekte und Filmmusik sowie sehr kurze Sprachsegmente prägen das Bild. Deshalb ist dieses Video bestens geeignet zu zeigen, inwieweit die vorliegende Arbeit ihr Ziel erreicht hat. Geprägt ist das (akustische) Bild durch abgehackte klingende Dialoge mit insgesamt zwölf teilweise einmalig autretenden Sprechern unter ständig wechselnden Bedingungen, was das erstellte System anhand der Szenen eins bis zwölf erfahren darf. Da vergleichbares nicht in den MPEG-7-Testdaten zu finden war, wurde auf diesen propritären Fall zurückgegriffen. Die folgenden Daten sind jedoch ihre Herkunft betreffend eher ”standardisiert“, da sie dem MPEG-7-Testatz entstammen.

über den Hintergrund sowie die Chance, durch explizite Modellierung des in seiner Reinform vorliegenden Rauschens exakte Hintergrundmodelle zu trainieren, was eine Untersuchung des GMM-IB unter Laborbedingungen erlaubt: Wie gut könnte die Methode abschneiden, wenn die Hintergrundmodelle nur gut genug die realen Bedingungen widerspiegeln würden?

`hallo.wav` stellt einen Spezialfall dar: Ein Ein-Mann-Theaterstück unter freiem Himmel (starkes, jedoch stationäres Rauschen durch Wind), in welchem der Schauspieler nur das Wort "Hallo" in verschiedenen Tonlagen immer wieder ausspricht. Wegen des kurzen Wortes sind auch die Segmente sehr kurz, und interessant ist, inwieweit das System mit der verstellten Stimme umgehen wird. Die Grundannahme ist, dass die drei als unterschiedlich identifizierbaren Stimmen auch als drei Personen gewertet werden sollen. Zu Testzwecken wurde lediglich Szene zwei mit Segmentierungslisten ausgestattet, da sie bereits alle relevanten Besonderheiten beinhaltet.

`docon.wav` ist ein spanischer Zeichentrickfilm. Überzeichnete Figuren lassen auf gute Sprecherdiskriminierung hoffen, jedoch sind die Sprachsegmente filmtypisch wieder recht kurz. Als Referenzszene dient hier die fünfte, in welcher sich drei unterschiedliche Sprecher Wortgefechte liefern. In diesem und den folgenden Testfällen soll auch ermittelt werden, inwieweit das System mit unterschiedlichen Sprachen und der damit andersartigen Stimmcharakteristik umzugehen weiß.

`lanc.wav` zeigt einige kurze Szenen eines Touristenführer-Programms in englischer Sprache. Die mittellangen Sprachsegmente werden durchweg von Musik begleitet, und für jeden der vier Sprecher existieren nur sehr wenige Sprachsegmente. Dafür tritt in jeder der untersuchten Szenen zwei bis fünf nur jeweils ein einziger, ansonsten nicht wiederkehrender Sprecher auf.

`nhkvideo.wav` ist eine Reportage in einer asiatischen Sprache: Eine einzige Stimme ist während der Szenen eins bis vier unter verschiedenen Aufnahmebedingungen zu vernehmen: Teils als Moderator unter Studio-Bedingungen mit und ohne Musikbegleitung, teils vor Ort als Interviewer mit entsprechend verrauschtem Hintergrund.

`camiloefilho.wav` schließlich ist ein typischer Vertreter einer Comedy-Serie: Die ausgewählte Szene zwei ist ein langer Dialog kurzer Segmente zwischen einem Mann und einer Frau vor dem Hintergrund eines sich amüsierenden Publikums. Die Schauspieler stellen teils extreme emotionale Zustände dar, welche sich in stark verfremdeten Stimmen niederschlagen.

Mittels diesen Ensembles scheint es nun möglich, das oben formulierte Ziel systematisch zu untersuchen.

4.2.3 Ergebnisse

Phase eins – Parameterfindung

Wie schon gesagt, soll diese Phase der Findung möglichst optimaler Signalverarbeitungs- und Merkmalsextraktions-Parameter im Bezug auf oben genannte Ziele dienen. Da es an dieser Stelle im System jedoch noch nicht wirklich möglich ist, sinnvoll mit Nebengeräuschen umzugehen, und die Verarbeitung unterschiedlicher Sprecheridentitäten erst in

den Bereich der Klassifikationsstufe fällt, liegt das Hauptaugenmerk dieses Tests klar auf dem Umgang mit kurzen Segmenten.

Deshalb zeigt Tabelle 4.3 einige Zahlen zu den im Folgenden zum Testen verwendeten Segmentlängen. Interessant ist dabei das Verhältnis von Brutto-[33] zu Nettosegmentlänge[34]: Von einer Sprachäußerung bleibt nach Entfernung aller störender Information kaum mehr als die Hälfte übrig.

Brutto-Länge	Netto min. [s]	Netto max. [s]	Netto ϕ [s]	Netto/Brutto [%]
Original	2,260	21,375	9,278	53,98
3 Sekunden	0,700	2,335	1,645	54,74
1,5 Sekunden	0,215	1,225	0,811	54,79
1 Sekunde	0,070	0,840	0,543	54,30

Tabelle 4.3: Statistiken zur verwendeten Segmentlänge während der ersten Testphase. Die Daten entstanden bei einer Rahmenlänge von 10ms bei 5ms Überlappung.

Die Daten beziehen sich dabei auf die Szenen drei, fünf, sechs und sieben des `news2`-Videos, um einen Fall zu konstruieren, bei welchem das System zwar möglichst einfache Daten vorgelegt bekommt, jedoch immerhin zwei unterschiedliche Sprecher auseinanderhalten muss: Szene sechs enthält nämlich die Stimme eines männlichen Interviewpartners vor dem Hintergrund mäßigen Rauschens, während die übrigen Szenen Beispiele der Stimme der Moderatorin unter Studiobedingungen, also maximal nebengeräuschfrei, beinhalten. Mit Hilfe dieser Szenen kann also überprüft werden, ob der gewählte Parametersatz bei gegebener Segmentlänge in der Lage ist, identische Sprecher in unterschiedlichen Szenen auch bei Vorhandensein eines weiteren Sprechers als gleich zu erkennen.

Zuerst soll nun der Einfluss der vier Parameter `audioFrameSize`, `audioFrameStep`, `filterBankSize` und `FFTsize` untersucht werden. Für alle weiteren freien Parameter, so sie denn nicht nur der Ablaufsteuerung dienen[35] und damit keinen direkten Einfluss auf die Parametrisierung der Algorithmen haben, gelten deshalb feste Werte, welche in Tabelle 4.4 dargestellt sind. Informell gesprochen ist das so entstandene System also ein Standard-GMM-Klassifikator mit möglichst konservativer Parametrisierung im Einklang mit der gelesenen Literatur. Als Distanzmaß arbeitet die GLR auf MFC-Koeffizienten zehnter Ordnung.

Nun, da die Vorbedingungen geklärt sind, kann mit dem eigentlichen Test obiger Parameter begonnen werden. Dessen Ergebnis kann direkt der Abbildung 4.3 entnommen

[33] Die *Bruttosegmentlänge* beschreibt die Länge eines Sprachsegments vor Abzug nicht-sprachlicher Anteile.

[34] Die *Nettosegmentlänge* bezieht sich auf die Länge eines Sprachsegments nach Abzug aller irrelevanten Signalanteile und bezeichnet letztendlich die Segmentlänge, welche die nachfolgenden Klassifikationsstufen zu Gesicht bekommen.

[35] Wie beispielsweise die Wahl der Szenen, Vorverarbeitungsschritte oder des Ausgabeverzeichnisses.

Parameter	Wert
dEnergy	0
windowed	1
preEpmhasizeFactor	0,97
MFCCorder	10
MFCCcoeffSelection	2046
CMN	0
lowCut	0
highCut	8000
lastNdistances	4
adaptiveThresholdAlpha	1,2
percentDifference	10
deltaBIClambda	1
maxMixtures	32
varianceLimit	0,01
EMthreshold	100
maxNoiseModelOrder	0
maxSpeakerModelOrder	8
vectorsPerGaussian	200
distanceMeasure	1
terminationCriterion	0

Tabelle 4.4: Basissystem für die erste Testphase. Der Wert der `vectorsPerGaussian` wird dabei jeweils so an die Rahmengröße und Überlappung angepasst, dass eine Gesamtmenge von einer Sekunde Sprachdaten zur Modellerstellung nötig wird.

werden. Bei der Wahl der zu überprüfenden Einstellungen wurde darauf geachtet, möglichst den gesamten Bereich der Möglichkeiten abzudecken, welche in der verwendeten Literatur beschrieben werden. Erkennbare Sackgassen wurden dabei schon frühzeitig von der weiteren Verfolgung ausgeschlossen.

Wie zu erwarten war, bewähren sich die kurzen Rahmen besser als längere, da somit ja mehr Merkmalsvektoren zur Erstellung einer soliden Modellgrundlage erzeugt werden. Gerade bei kurzen Segmenten kann dies das sprichwörtliche Zünglein an der Waage sein, welches bei 10ms Rahmen noch zu perfekter Ermittlung der Anzahl an Clustern/Sprechern führt, während hier bei 32ms schon vermehrt Fehler auftreten. Auch ein Blick auf den Faktor Sicherheit zeigt, dass kurze Rahmenlängen mit viel Überlappung zu bevorzugen sind.

Zusätzlich scheint die Tendenz hin in Richtung größerer Filterbänke und eines vermehrten Einzugbereichs der FFT zu gehen. Die Erfahrung lehrt hier jedoch Vorsicht: In der Literatur werden typische Filterbänke mit maximal 50 Filtern überliefert, so dass die hier erzielten Ergebnisse mit Bänke von bis zu 120 Filtern kritisch betrachtet werden müssen. Eventuell handelt es sich nur um zufällige Ausreißer, und da die Güte der Ergebnisse für mehr als 60 Filter hier auch nicht zwingend zu deren Verwendung mahnt, sollen diese hohen Werte im Folgenden ignoriert werden. Ähnlich verhält es sich mit der FFT-Länge: Zwar erhöht sich bei längerer Beobachtungsdauer die Frequenzauflösung, falls jedoch die Rahmenlänge so gering ist, dass am Ende mit Nullen aufgefüllt werden muss, ergibt sich

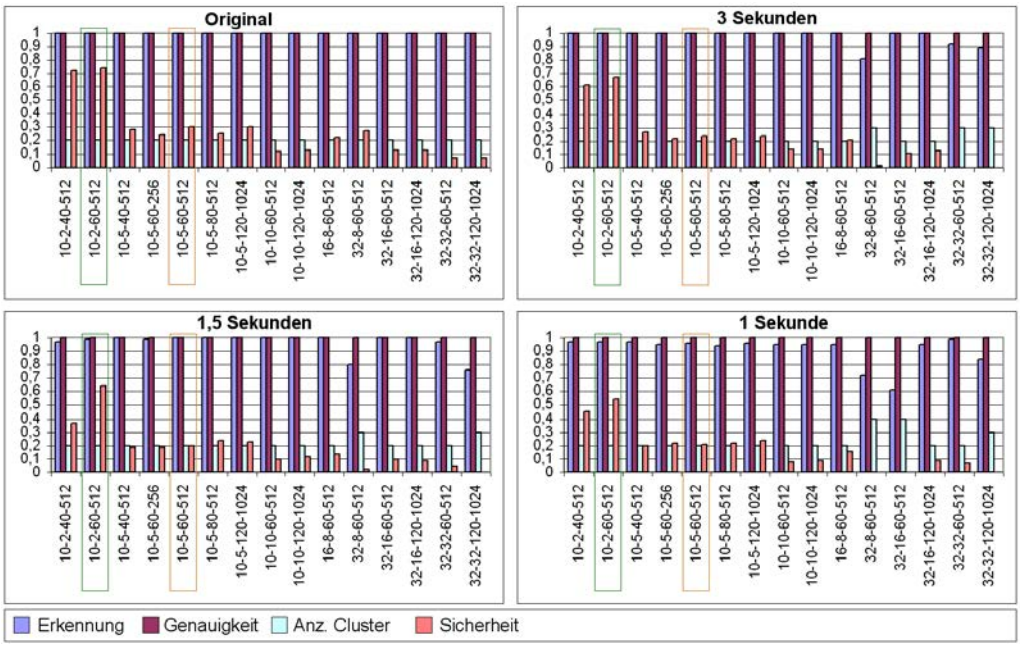

Abbildung 4.3: Ergebnis des Tests der ersten vier Parameter. Zur besseren Visualisierung wurde die Anzahl Cluster mit dem Faktor $\frac{1}{10}$, die Sicherheit mit $\frac{1}{10000}$ skaliert. Die vier Zahlen in der Beschriftung der x-Achse stellen die Werte der zu testenden Parameter in obiger Reihenfolge dar.

ein Spektrum, welches mit wachsendem Anteil besagter Nullen immer weiter von der Realität entfernt ist. Bei 10ms Rahmenlänge und 16.000Hz Abtastfrequenz stehen der FFT nur 160 reale Messwerte zur Verfügung – dies sollte bei der Wahl dieses Parameters bedacht werden.

Aus obigen Überlegungen heraus muss schließlich der in der Grafik grün umrandete Parametersatz als der beste angesehen werden: 10ms Rahmenlänge bei 8ms Überlappung[36], 60 Filtern und 512 Messwerten FFT-Länge. Dieses Wertequadrupel leistet deutlich mehr als das in der einschlägigen Literatur und den Standardwerten der SV_Lib vorgeschlagene System, welches in der Grafik rötlich umfasst ist. Mit ersteren Werten also eingeflochten in das Basissystem kann nun der nächste Parametersatz untersucht werden: Die die Merkmalsextraktion betreffenden Parameter `MFCCorder` und `dEnergy`.

Abbildung 4.4 kann entnommen werden, dass der Sicherheits-Wert mit zunehmender Anzahl an Koeffizienten ansteigt, während gleichzeitig jedoch der Erkennungs-Wert bei kurzer Segmenten stetig sinkt. Ein guter Kompromiss scheinen hier MFCC's 12. Ordnung zu sein: Sie bieten hohe Sicherheit bei mit dem Basissystem (in der Grafik rötlich umrandet) vergleichbarer Erkennung. Die Umwandlung des ersten MFC-Koeffizienten in seine Ableitung scheint nicht einheitlich positiv oder negativ bewertbar zu sein. Es ist allerdings zu beachten, dass in dem gewählten Testvideo sehr kontrollierte Bedingungen

[36] Aus 2ms Verschiebung des Rahmenfensters folgen 8ms Überlappung.

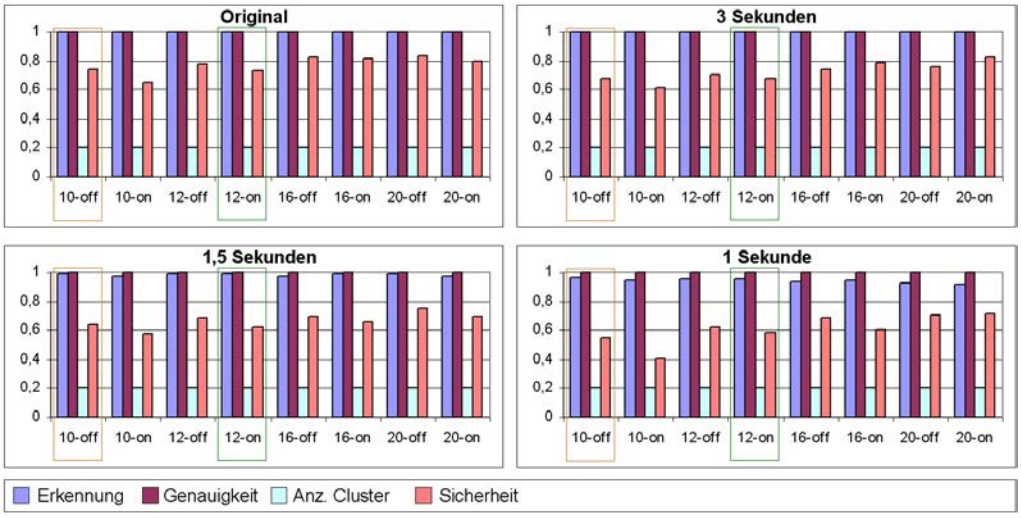

Abbildung 4.4: Ergebnis des Tests der Parameter `MFCCorder` und `dEnergy`. Die Anzahl Cluster sowie die Sicherheit wurden wiederum wie oben angegeben skaliert. Die Beschriftung an der x-Achse enthält die entsprechend gewälten Parameterwerte.

vorherrschen, die auftretenden Sprecher also immer vergleichbar laut reden. Da der erste Koeffizient nun die durchschnittliche Signalenergie darstellt, also einen mit Lautstärke eng verknüpften Wert, mag er in diesem speziellen Fall eines homogenen Umfelds tatsächlich auch in seiner Urform sprecherdiskriminative Informationen enthalten. In Übereinstimmung mit den Aussagen von Abschnitt 2.4.1 wird für das Basissystem jedoch die in der Grafik grün eingefasste Variante der Umwandlung in den Delta-Koeffizienten gewählt, da die vorliegenden Testergebnisse dies nicht explizit verbieten und das Verfahren Vorteile in geräuschreicheren Umgebungen verspricht.

An dieser Stelle sei noch einmal kurz auf die Interpretation der Werte von Erkennung und Genauigkeit eingegangen: Letzterer war in den vergangenen Experimenten durchweg auf 100 Prozent, was bedeutet, dass das Verfahren in einem Cluster niemals Segmente unterschiedlicher Sprecher zusammenfasste. Der Erkennungs-Wert hingegen schwankte durchaus etwas, was natürlich teilweise mit einer falschen Anzahl an Clustern begründbar ist, jedoch auch noch eine andere Bewandtnis hat: Es ist ein Zeichen dafür, dass manche Segmente zu kurz für die Verarbeitung waren und deshalb als fehlend diesen Wert drückten. Da die Ausgangssegmente jedoch für jeden Parametersatz identische Größe hatten, konnte eine Verschlechterung der Erkennung bei wechselnden Parametern nur geschehen, wenn die vorgelagerte Sprecherwechsel-Erkennung es nicht schaffte, kurze Segmente identischer Sprecher zusammenzufassen. Oder wenn die ihr vorgelagerten Sprachfilter-Stufen zu viel eigentlich brauchbare Sprachinformation als nicht-stimmhaft oder gar nicht-sprachlich markierten. In diesen Fällen bedeutet ein niedriger Erkennungs-Wert also, dass die Vorverarbeitungsstufen mit den gewählten Parametern nicht optimal eingestellt sind und ineffizient unterhalb ihrer Möglichkeiten betrieben werden.

Die letzte Experimentierstufe in Phase eins soll nun noch Auskunft darüber geben, ob

die in [LIN 99] und [PETR 03] beschriebenen Resultate auch zu besseren Ergebnissen im Kontext dieser Arbeit führen können: Erstere Quelle beschreibt die gezielte Verwendung nur spezieller Subbänder, während letztere für einen selektiven Einsatz nur bestimmter MFC-Koeffizienten eintritt.

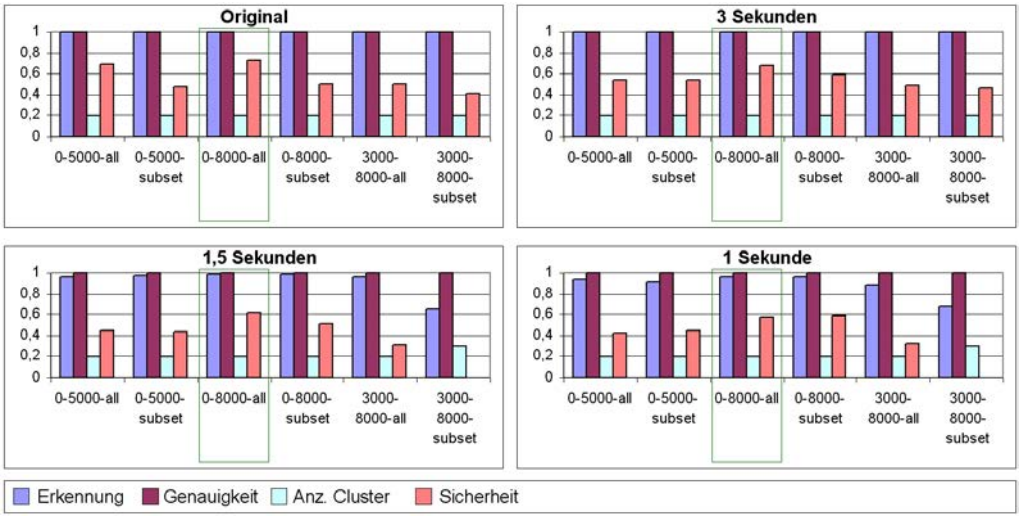

Abbildung 4.5: Ergebnis der Merkmalsselektions-Tests. Die Skalierung der Werte ist identisch mit derjenigen in Abbildung 4.4. Die x-Achsen-Beschriftung enthält die Werte der Parameter `lowCut`, `highCut` sowie die Aussage, ob mittels `MFCCcoeffSelection` eine Auswahl an Koeffizienten in Einklang mit Abschnitt 2.4.1 getroffen wurde.

Abbildung 4.5 zeigt jedoch, dass beide Änderungen die Ergebnisse nur verschlechtern, was im Falle der Selektion des oberen Sprachbandes, des entschlackten Merkmalsvektors und kurzer Segmente sogar zu einer Falscheinschätzung der Anzahl Sprecher seitens des Systems führt. Folglich bleibt hier das Basissystem bei den zuvor ermittelten Werten, wie auch der grün eingefärbte Parametersatz im Diagramm zeigt.

Phase zwei – Vorverarbeitung

Zunächst sollen die Parameter `lastNdistances` und `adaptiveThresholdAlpha` untersucht werden: Sie korrespondieren mit dem Verfahren zur adaptiven Schwellwertbestimmung, welches bei der Detektion potenzieller Sprecherwechsel zum Einsatz kommt. Das Ergebnis in Abbildung 4.6 zeigt dabei, dass alterierende Einstellungen auf die Ergebnisse langer Segmente kaum einen Einfluss haben. Für kürzere Sprachperioden hingegen bringen eine größere Menge zu Rate gezogener Werte bei der Schwellwertbestimmung und ein höherer Verstärkungsfaktor besserer Endergebnisse, weshalb mit diesen Werten, wie in der Abbildung grün umrandet, weiter experimentiert wird.

Interessant ist es noch zu verfolgen, wie mit kürzer werdenden Sprachsegmenten die Anzahl gefundener Sprecherwechsel steigt: Eigentlich enthält jede Szene nur genau einen

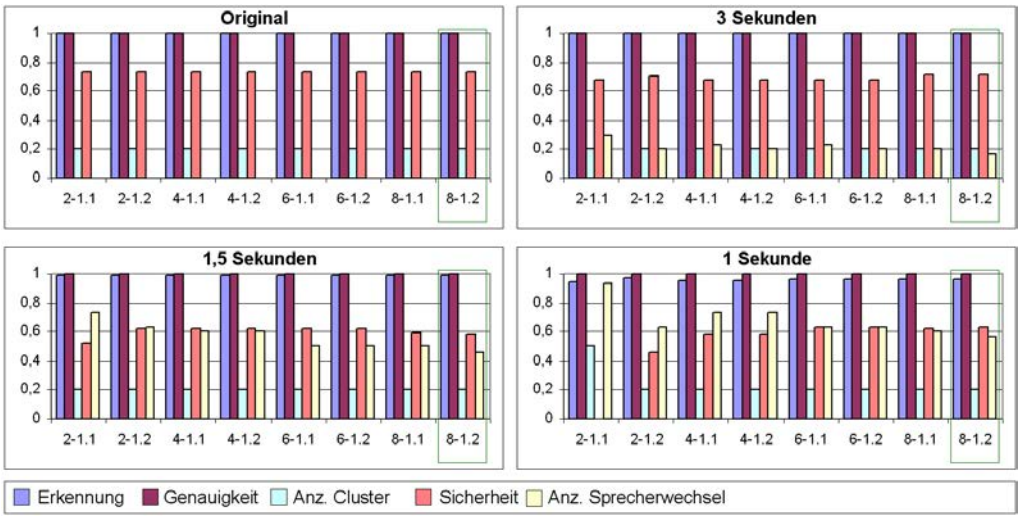

Abbildung 4.6: Experimente zur Sprecherwechsel-Erkennung. Sicherheit und Anzahl Cluster sind wieder entsprechend den obigen Diagrammen skaliert, neu hinzu gekommen ist der Wert der Anzahl erkannter Sprecherwechsel, skaliert um den Faktor $\frac{1}{30}$. Die Einstellungen der Parameter `lastNdistances` und `adaptiveTresholdAlpha` finden sich in der Beschriftung der x-Achse.

Sprecher, also keine Wechsel, was bei original langen Segmenten auch richtig erkannt wird. Mit zunehmender Kürze fehlen dem Algorithmus aber scheinbar genügend Daten zur sicheren Entscheidungsfindung, weshalb hier deutlich mehr Wechsel als real vorhanden proklamiert werden.

Der nächste Test der Parameter `percentDifference` und `deltaBIClambda` sollte also gezielt auch Parameter enthalten, welche zu großen zusammenhängenden Sprachsegmenten, also weniger Wechseln tendieren: Ein kleiner Wert letzterer Veränderlichen sollte dabei große Modelle nicht über die Maßen bestrafen, also dem Algorithmus einen Anreiz geben, weniger Wechsel zu detektieren.

Parameter	Wert 1	Wert 2	Wert 3
percentDifference	5	10	20
deltaBIClambda	1,0	1,1	1,2

Tabelle 4.5: Werte für den Test der qGMM-Parameter während der Sprecherwechsel-Erkennung: Jeder Wert von `percentDifference` wurde in Kombination mit allen drei `deltaBIClambda`-Einstellungen überprüft.

Interessanterweise zeigt sich, dass für sämtliche in Tabelle 4.5 angegebenen Wertekombinationen völlig identische Ergebnisse sowohl für Erkennung und Genauigkeit als auch für die Anzahl der Sprecherwechsel und Cluster sowie für die Sicherheit ermittelt werden. Der Algorithmus scheint also recht robust gegen Änderungen an dieser Stelle zu sein. Für das Basissystem heißt dies, dass weiterhin mit den Werten 10,0 als gutem Mittelwert für

`percentDifference` sowie 1,0 als einzigem auch mit der Theorie in Einklang stehendem Wert für das `deltaBIClambda` gearbeitet wird.

Phase drei – Sprecherklassifikation in Videos

Zu Beginn des wirklichen Tests steht noch einmal eine kurze Phase der Parameterfindung: Einige Werte für den Modellierungs- und Klassifikationsprozess bedürfen noch der Untersuchung. Im einzelnen sind dies die Parameter `EMthreshold`, `maxNoiseModelOrder`, `maxSpeakerModelOrder`, `varianceLimit` und `distanceMeasure`. Die Ergebnisse stimmen hier jedoch gut mit der eigenen Erwartung sowie der einschlägigen Fachliteratur überein, weshalb auf eine ausführliche Besprechung verzichtet werden kann. Somit steht das Ausgangssystem zum Zwecke der Sprecherklassifikation in seiner Gesamtheit wie in Tabelle 4.6 dargestellt fest.

Parameter	Wert	Parameter	Wert
phaseSelect	7	audioFrameSize	10
audioFrameStep	2	energyQuantizationLevel	10
pauseSilenceThreshold	999999	filterBankSize	60
FFTsize	512	dEnergy	1
windowed	1	preEpmhasizeFactor	0,97
MFCCorder	12	MFCCcoeffSelection	8190
CMN	0	lowCut	0
highCut	8000	segmentLengthThreshold	0
lastNdistances	8	adaptiveThresholdAlpha	1,2
percentDifference	10	deltaBIClambda	1
maxMixtures	32	varianceLimit	0,01
EMthreshold	10	maxNoiseModelOrder	0
maxSpeakerModelOrder	16	vectorsPerGaussian	200
distanceMeasure	1		

Tabelle 4.6: Die Parameter des endgültigen Basissystems für die letzte Testphase.

Der nächste Test soll zeigen, inwiefern die im Verlauf dieser Arbeit entwickelten Verfahren zur Sprachanreicherung einen wirklichen Vorteil gegenüber diesem Basissystem bringen, sobald Nebengeräusche ins Spiel kommen. Zu diesem Zweck wird es im Folgenden mit einem System ähnlicher Parametrisierung verglichen, welches jedoch zur Sprecherrepräsentation ein GMM-IB 16. Ordnung mit integriertem Nebengeräusch-GMM achter Ordnung verwendet. Testgrundlage sind die bereits bekannten Szenen des `news2`-Videos, jedoch teilweise angereichert mit synthetischem Rauschen sämtlicher coleur wie in Tabelle 4.2 beschrieben.

Durch die Zusammenstellung der Testfälle lassen sich, neben dem allgemeinen Umgang des jeweiligen Verfahrens mit Nebengeräuschen, vor allem folgende zwei Fälle überprüfen: Zum einen, inwieweit sich das System von identischem Hintergrundgeräusch dazu verleiten lässt, eigentlich grundverschiedene Sprecher als identisch anzusehen. Zum anderen,

in wie weit das System identische Sprecher auch vor wechselndem Hintergrund als solche erkennt. Für das GMM-IB-System gilt dabei, dass sämtliche Hintergrundmodelle zuvor extern anhand des reinen Nebengeräuschs trainiert wurden, um wirklich optimale, mit der Literatur vergleichbare Testbedingungen zu schaffen. Die Ergebnisse präsentiert Abbildung 4.7.

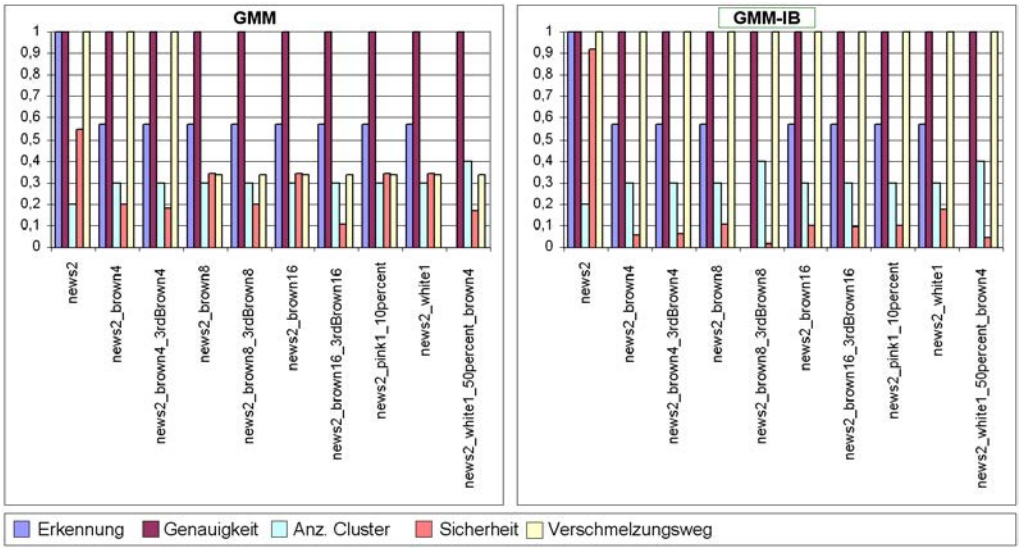

Abbildung 4.7: Vergleich des Basissystems mit dem neu erstellten GMM-IB-Verfahren. Die x-Achse enthält die Dateinamen der jeweiligen Testvideos.

Zuerst fällt auf, dass beide Verfahren die Anzahl Sprecher bei zugemischtem Störeinfluss durchweg falsch einschätzen, dies jedoch im Falle des GMM-IB mit deutlich geringerer Sicherheit. Das ist ein positives Ergebnis, wenn man bedenkt, dass es sich bei der Anzahl Cluster jeweils um Falschprognosen handelt. Anschaulich ließe sich sagen, der Kandidat läge zwar daneben, behaupte aber wenigstens nicht mit an Arroganz grenzender Sturheit, er habe die Wahrheit für sich gepachtet. Doch einmal abgesehen von der Anschauung liefert auch die Mathematik den entsprechenden Interpretationsansatz: Ein negativer ΔBIC-Wert bedeutet ja, dass eine Verschmelzung die Gesamtwahrscheinlichkeit der Partitionierung herabsetzt, oder anders ausgedrückt, dass durch sie bereits vorhandene Information abnimmt[37]. Ein von unten näher an null liegender Wert geht also einher mit ähnlicheren Clustern, was auf die Reduktion der Störeinflüsse zurückgeführt werden kann, so dass die Sprachinformation besser zum Tragen kommt.

[37]Diese Denkweise wird klarer am Beispiel der positiven ΔBIC-Werte: Je größer sie ausfallen, desto mehr Gewinn lässt sich aus der Verschmelzung zweier Cluster ziehen. Ein Wert knapp über null heißt nicht etwa, dass sich die beiden Kandidaten gerade noch genug ähneln, um noch als von dem selben Sprecher geäußert worden zu sein. Vielmehr sind sie beinahe gleich, weshalb die Wahrscheinlichkeit der Daten unter dem Gesamtmodell nicht stark wächst. Die komplette Information zur Modellierung des Gesamtsprechers ist jeweils schon in den Teilmodellen enthalten.

Des Weiteren ist zu beobachten, dass der Verschmelzungsweg im Falle des GMM-IB immer Korrekt verläuft. Ungeachtet von Nebengeräuschen wurden hier also gleiche Sprecher entsprechend erkannt, während das GMM jeweils dazu tendierte, gleiche Hintergründe zu verschmelzen. Das GMM-IB-System muss also, was den Umgang mit variablen akustischen Hintergründen angeht, dem Basissystem vorgezogen werden. Dieses Bild wird lediglich getrübt durch die Tatsache, dass selbst unter den optimalen Bedingungen expliziter Hintergrundmodelle eine richtige Ermittlung der Anzahl Sprecher unmöglich war. Der folgende Abschlusstest soll nun zeigen, wie sich das entwickelte System in der Praxis – in wirklichen Filmen unterschiedlicher Sprache und Länge ohne a-priori-Hintergrundmodelle – im Vergleich zu seinem Konkurrenten, dem Basissystem, verhält.

Um dabei den Grad der Abhängigkeit der Repräsentations-, Anreicherungs- und Klassifikationsverfahren von den Vorverarbeitungsschritten zu ermitteln, werden die Experimente in zwei Varianten, jeweils einmal mit und ohne Sprecherwechsel-Erkennung, durchgeführt. Denn die Sprecherwechsel-Erkennung ist der mit Abstand fehlerträchtigste Teil der Filterungs-Stufe: Da zu dieser Zeit die kürzesten Segmente und unkompensierte Hintergrundgeräusche auftreten, hat er mit den schlechtesten Voraussetzungen zu kämpfen, wobei gleichzeitig eine an dieser Stelle falsch getroffene Entscheidung[38] zu schwerwiegenden Fehlklassifikationen in den darauf aufbauenden Stufen führen wird. Ein Test ohne diesen Fehlerherd könnte deshalb interessante Resultate liefern, hat jedoch natürlich mit dem Nachteil zu kämpfen, sehr kurze Segmente verarbeiten zu müssen.

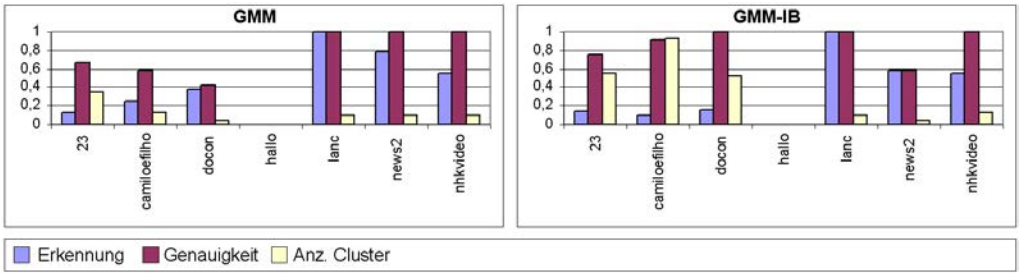

Abbildung 4.8: GMM und GMM-IB im Vergleich unter realen Bedingungen (ohne Sprecherwechsel-Erkennung). Die Anzahl Cluster wurde, um den Wertebereich nicht zu verlassen, um den Faktor $\frac{1}{50}$ skaliert.

Abbildung 4.8 visualisiert die so erlangten Ergebnisse: Während diese für die gemäßigteren Aufgaben `lanc`, `nhkvideo` und `news2` noch recht gut ausfallen und aufgrund zu kurzer Segmente bei `hallo` gar keine Aussage getroffen werden kann, sieht das Bild bei den drei wirklichen Filmen rechts im Bild nicht sehr positiv aus: Die höheren Genauigkeiten bei Verwendung des GMM-IB gehen sicher mit der enormen Anzahl an Clustern einher[39], so dass sie nicht zu sehr zu loben ist. Und die geringere Clusteranzahl im Fal-

[38] Ein übersehener Wechsel.

[39] Dies zeigt wieder den typisch antiproportionalen Zusammenhang zwischen den beiden Maßen: Wenn jedes Segment seinen eigenen Cluster erhält, führt dies zwar zu miserabler Erkennung, doch perfekter Genauigkeit.

le des GMM rührt eher von der Zusammenfassung ähnlicher Hintergrundgeräusche denn dem wirklichen Erkennen von Sprechern, auch wenn so natürlich mehrere Zufallstreffer die Erkennungs-Rate nach oben treiben. Wie wird sich dieses nicht sehr beeindruckende Bild wohl bei Verwendung der Sprecherwechsel-Erkennung ändern?

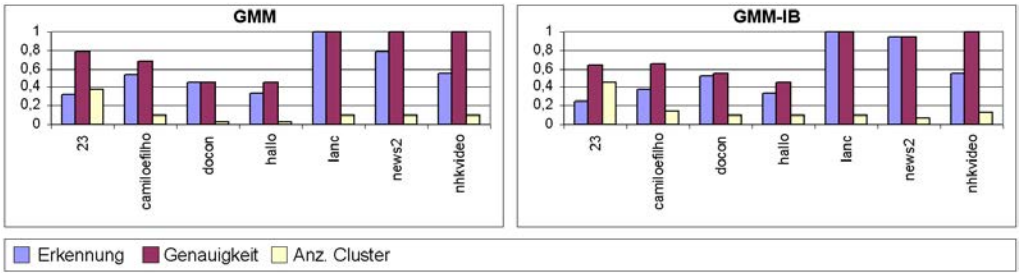

Abbildung 4.9: GMM und GMM-IB im Vergleich unter realen Bedingungen (mit Sprecherwechsel-Erkennung). Die Werteskalierung efolgte analog zu obiger Abbildung.

Wie Abbildung 4.9 zu entnehmen ist, hatte dies tatsächlich einen positiven Einfluss auf das Gesamtergebnis, auch wenn die Sprecherwechsel-Erkennung wie befürchtet oft verschiedene Sprecher als identisch zusammenfasste und so die Hauptverantwortung für die schlechten Genauigkeitsraten trägt. Wie Tabelle 4.10 hier deutlich zeigt, liegt das Problem nicht in zu vielen falsch positiv erkannten Wechseln (die Precision-Rate ist doch vergleichsweise hoch), sondern an der Masse "vergessener" Wechsel, welche sich in dem schlechten Recall-Wert niederschlagen.

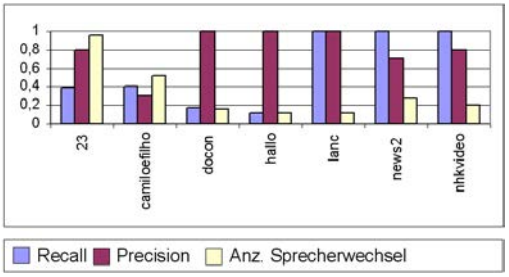

Abbildung 4.10: Die Leistung der Sprecherwechsel-Erkennung im obigen Klassifikationslauf. Die Anzahl Wechsel wurde mit dem Wert $\frac{1}{25}$ skaliert.

Zurück zu den Endergebnissen der Klassifikation in Abbildung 4.9 lässt sich nun sagen, dass für die gemäßigteren Fälle nun tatsächlich gute Ergebnisse erzielt wurden. Die Erkennungsleistung für Filme liegt aber weiterhin so deutlich unter 50 Prozent, dass es sogar kaum Sinn macht, hier einen Sieger zu küren. Weder das GMM-basierte Basissystem noch die Neuentwicklung können hier überzeugen, auch wenn das Basissystem rein zahlenmäßig besser darzustellen versteht. Dies relativiert sich jedoch bei der anschaulichen Überlegung, dass beispielsweise in der Zwei-Sprecher-Situation des `camiloefilho`-Videos

eine rein zufällige Zuordnung der Segmente auf Sprecher kaum geringere Erkennungs-Leistung gezeigt hätte. So ist es auch nur ein schwacher Trost festzustellen, dass beide Systeme erwartungsgemäß unabhängig von der Sprache eines Audiodokumentes arbeiten können.

So bleibt nun noch zu untersuchen, wie sich das Laufzeitverhalten der Systeme unter den verschiedenen Testbedingungen entwickelt hat.

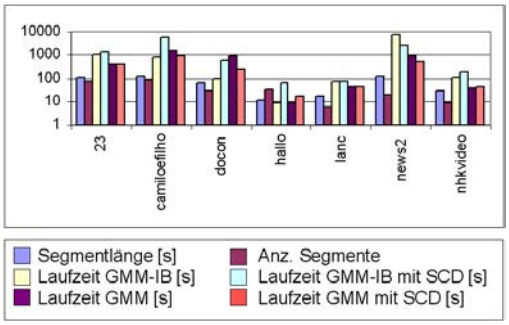

Abbildung 4.11: Statistiken zum Laufzeitverhalten. Aufgrund der großen Diskrepanz der Laufzeiten untereinander musste auf eine logarithmische Skalierung der y-Achse ausgewichen werden.

Der Theorie halber müsste der Aufwand, gemessen in Sekunden Laufzeit, für die Benutzung des GMM-IB deutlich höher als bei Verwendung des reinen GMM's liegen, da iterationen zusätzlich über die Anzahl der Hintergrundmodell-Komponenten berechnet werden müssen. Weiter sollte er eigentlich exponentiell mit der Anzahl zu klassifizierender Segmente steigen[40]. Beides ist der Grafik in Abbildung 4.11 nicht direkt zu entnehmen. Dies hängt einerseits damit zusammen, dass das Clustering kaum bis zum Ende durchgeführt wird, sondern im Falle des GMM-IB schon häufig früh abbricht, während das GMM-System hier mehr Iterationen durchführt. Außerdem wird bei Nichtzustandekommen eines Hintergrundmodells von algorithmischer Seite ja auf ein reines GMM zur Klassifikation ausgewichen, so dass auch hier noch einmal Zeit eingespart werden kann.

Festzuhalten bleibt also, was sich am Beispiel des `news2`-Videos am deutlichsten zeigen lässt: Für ein knapp über vier Minuten langes Audiodokument kann die Bearbeitungszeit leicht in die Stunden steigen, wenn viele Segmente weniger Sprecher tatsächlich korrekt zusammengefasst werden müssen. Echtzeit ist mit den präsentierten Verfahren also nur bis zum Erreichen der Klassifikationsstufe machbar, obwohl auch hier schon das EM-basierte Modelltraining je nach gewählter Einstelllung und Genauigkeit an dieser Grenze nagt. Das hierarchische Clusteringverfahren hingegen ist für zeitkritische Prozesse wenig geeignet.

[40]Nicht exponetiell mit der *Länge* der Segmente: Diese erhöht zwar auch den Aufwand beim Testen der Modelle, doch nur für zusätzliche Segmente muss die Distanzmatrix während des Clusteringprozesses erweitert werden, was die Anzahl zu berechnender Distanzen, also auszuführender Modelltests, vervielfacht.

Kapitel 5

Zusammenfassung und Ausblick

5.1 Zusammenfassung

Nachdem der Weg zur Erstellung einer Sprecherklassifikations-Software nun detailliert beschrieben wurde und sämtliche Experimente mit der konkreten Implementierung zu ihrem Abschluss kamen, ist es nun an der Zeit, Rückschau zu halten: Was war, in komprimierter Form, Gegenstand der Aufgabenstellung und Inhalt dieser Arbeit, welchen Anteil haben eigene Überlegungen an der zustande gekommenen Lösung? Im Folgenden wird sich dieser Frage gewidmet, während die weiteren Abschnitte dieses Kapitels untersuchen, inwieweit die erzielten Ergebnisse eine Erreichung der gesteckten Ziele darstellen und wie anhand der gewonnenen Erkenntnisse weiter verfahren werden kann.

Die vorliegende Arbeit dokumentiert in erster Linie den Entstehungsprozess einer Software zur Indizierung von Videos anhand auftretender Sprecher. Zu diesem Zweck wurde zunächst ein umfassender Überblick über den aktuellen Stand der Forschung im Bereich der Sprachtechnologie und speziell auf dem Feld der Sprecherklassifikation gegeben. Anhand dessen wurde die Klassenbibliothek SV_Lib erstellt, welche in den relevanten Teilbereichen jeweils die aktuellsten Verfahren implementiert und die in der Literatur überlieferten Algorithmen teilweise an die hier im Besonderen vorherrschenden Bedürfnisse anpasst. Zu Testzwecken wurden weiterhin das Rahmenprogramm SCiVo sowie die grafische Oberfläche Gardener implementiert.

Als Teilprojekt gliedert sich dieses Softwaregeflecht in das System Mediana der Arbeitsgruppe Verteilte Systeme der Phillips-Universität Marburg ein. Da es in diesem Kontext jedoch der erste sich mit Audiodaten im Allgemeinen und Sprachtechnologie im Besonderen auseinandersetzende Flügel des Projekts ist, konnte auf keine bestehende Infrastruktur oder bereits gesammelte Erfahrung zurückgegriffen werden: Als Softwareprojekt quasi "auf der grünen Wiese" konnten vielmehr Grundlagen geschaffen werden.

Den Kern der Arbeit stellt die Anpassung des GMM-IB-Verfahrens von Rose et al. [ROSE 94] auf den hier beschriebenen Anwendungsfall dar: Nicht nur wurde der diesem statistischen Modell innewohnende Formelapparat von Fehlern befreit und die von den Au-

toren beschriebene Lücke zum Umgang mit impulsartigen Nebengeräuschen algorithmisch geschlossen; durch die erstmalige Anwendung auf realen Testdaten wurde das Verfahren auch aus seinem bisherigen Laborstatus herausgehoben.

5.2 Diskussion

Wie die Experimente mit synthetischen Testdaten im vorangegangenen Kapitel belegen, bietet die vorgeschlagenen Sprachanreicherungs-Technik tatsächlich einige Vorteile gegenüber dem GMM-Basissystem. Unter realen Bedingungen jedoch bricht dieser Vorteil völlig ein, und es scheint beinahe, als würden die komplexen Verfahren in seinem Inneren das GMM-IB sogar daran hindern, wenigstens auf das Leistungsniveau seines Vorfahren zu gelangen.

Aus dieser Tatsache lässt sich auf Zweierlei schließen: Zum einen auf das große Potential des erstellten Systems und, indirekt, auf die Tatsache, dass die zu seiner Implementierung führenden Gedanken in der Anwendung tatsächlich wie geplant funktionieren, die Implementierung also gleichermaßen wie die eigentliche Idee korrekt ist. Zum anderen zeigen sich jedoch auch deutlich die Grenzen des Ansatzes: Sind perfekt passende Hintergrundmodelle nicht verfügbar, lässt sich mit Annäherungen daran kaum vernünftig arbeiten.

Somit kann die Erreichung der eingans beschriebenen Ziele folgendermaßen beurteilt werden: Der Umgang mit nicht a priori explizit bekannten Sprechern kann als gewährleistet angesehen werden. Dies wird eindeutig durch die Experimente mit dem Nachrichten- und Reportagematerial in Testphase drei belegt. Hintergrundgeräusche hingegen sind nur akademisch bewältigt, in der Praxis mangelt es noch an ausreichend genauen Hintergrundmodellen. Das Problem (zu) kurzer Sprachsegmente schließlich muss weiterhin als ungelöst angesehen werden: Weder die Sprecherwechsel-Erkennung noch die Klassifikation konnten mit den gewählten Verfahren[1] in diesem Bereich punkten.

Ansonsten darf die Relation der erzielten Ergebnisse zu anderen Arbeiten nicht vernachlässigt werden: Bisher wurde das Gebiet der Sprecherklassifikation in *Videos*[2] von Forschern stark gemieden, und die wenigen publizierten Resultate machen entweder doch von vorne herein Annahmen über die zu analysierenden Daten, um aus diesem Informationsvorsprung Profit zu schlagen [LI 03] oder bewegen sich mit ihren Ergebnissen auf weniger anspruchsvollem Material im Bereich um 60 Prozent Recall und Precision [NISH 99]. In so fern liegen die hier präsentierten Ergebnisse nur knapp unterhalb dessen, was in den Forschungslaboratorien dieser Welt momentan produziert wird.

[1] Diese Verfahren wurden immerhin aus einer großen Menge an Bewerbern auch unter dem Gesichtspunkt ausgewählt, gerade im Bereich kurzer Segmente noch vernünftig arbeiten zu können.

[2] *Video* ist in diesem Kontext ein Synonym für Unbeschränktheit und damit Unmachbarkeit.

5.3 Taktischer Ausblick

Auch wenn die Arbeit an der SC_Lib und ihrem Supplement mit dieser Arbeit zuerst einmal ihren Abschluss gefunden hat, gibt es doch noch einige Ansätze, mit denen an der aktuellen Stelle fortgefahren werden könnte:

Da wäre zunächst der Status der Software: Die Entwicklung ist abgeschlossen, die Dokumentation geschrieben, und Version 1.0 ist einsatzbereit. Doch wie für diese Versionsnummer üblich gibt es noch diverse Punkte, an denen Verbesserungen möglich währen. Zu nennen ist hier sicherlich die Behandlung von Fehlern: Sämtliche in der Entwicklung auftretenden Ausnahmesituationen wurden zwar abgefangen, doch für den Einsatz in beliebigen Produktivumgebungen müsste hier deutlich mehr gemacht werden. Auch könnte an dem Zusammenspiel zwischen SC- und SV_Lib weiter gearbeitet werden, um einige momentan stark spezialisierte Algorithmen auch für beispielsweise andere Modelle zugänglich zu machen.

Diese Liste ließe sich sicherlich noch etwas weiter fortführen, doch soll es an dieser Stelle damit genügen. Denn der wichtigere Aspekt in diesem Zusammenhang ist sicherlich der Fachliche: Dort ist sicherlich die Frage nach einer besseren Methode zur Erstellung der Hintergrundmodelle die wichtigste. Zwar wurden auch hier einige Überlegungen in diesen Algorithmus investiert, doch sprechen die Ergebnisse leider gegen deren Erfolg. Eine interessante Möglichkeit mag hier folgendes sein, was auch in [YOSH 01] schon anklang: Vorgefertigte Hintergrundmodelle für die gängigsten Geräuschsituationen in verschiedenen Intensitätsstufen, aufgeteilt auf ihre Subbänder und immer nur einen kurzen zeitlichen Rahmen abdeckend, könnten aufgrund der Erweiterung des GMM-IB zum Umgang mit Impulsen folgendermaßen genutzt werden:

In kurzer zeitlicher Abfolge wird ein Test sämtlicher so a priori verfügbarer Hintergrundmodelle gegen die aktuellen Merkmalsvektoren durchgeführt. Das Modell mit dem höchsten Log-Likelihood-Wert wird zum Hintergrundmodell für diesen Abschnitt erklärt und im Rahmen des GMM-IB verwendet. Eine genügend große Auswahl an fertigen Hintergrundmodellen sowie viel Rechenzeit vorausgesetzt, könnte so ein leistungsfähigeres System entstehen.

Zwei weitere Ideen zielen auf die Verbesserung der vorgelagerten Verarbeitungsstufen ab: Mittels Wavelet-Analyse anstatt eines FFT-basierten Verfahrens könnten die zeitliche Auflösung der Merkmalsvektoren verbessern. Die Forschung in diesem Bereich ist zwar noch nicht abgeschlossen, doch zumindest hoffnungsvoll. Und eine weniger intuitiv geprägte Detektion nicht-stimmhafter Sprache könnte weiter zur Verbesserung der Modellreinheit beitragen. Zu denken ist hier beispielsweise an ein Verfahren, wie [AJME 02] es vorschlägt.

5.4 Strategischer Ausblick

Wie steht es um die Anknüpfungspunkte der erstellten Software an angrenzende Systeme, um ihr Verhältnis zu verwandten Technologien? Zunächst muss hier natürlich der Kontext des Projekts Medienumbrüche betrachtet werden: Die Integration in diesen Rahmen hat zwar noch nicht stattgefunden, jedoch ist alles dazu vorbereitet: Lediglich die Erstellung einer passenden grafischen Benutzerschnittstelle auf Java-Seite[3] steht noch aus. In so fern bietet sich hier also eine Kopplung an dort bereits erstellte oder demnächst verfügbare Teilsysteme an, beispielsweise die Schnitt- und Szenendetektion oder die Audiosegmentierung, so dass die momentan noch zwingend handerstellten Segmentierungslisten wegfallen könnten und das System tatsächlich vollautomatisch arbeitet.

Der nächste Anknüpfungspunkt ergibt sich direkt aus den verwendeten Daten: Um Personen innerhalb von Videos zu identifizieren, ist die Stimme nur ein Charakteristikum unter vielen, das Gesicht ist ein Weiteres. Arbeiten wie [TSEK 98] oder [CHOU 99] zeigen deutlich, welcher Gewinn sich aus der Verknüpfung von Sprecher- und Gesichtserkennung ziehen lässt. Und mit Methoden wie in [GEOR 00] publiziert ließe sich so manches bewerkstelligen: Die Autoren entwickelten dort ein System, welches anhand dreier Aufnahmen eines Gesichts unter unterschiedlichen Bedingungen in der Lage ist, ein 3D-Modell des Kopfes unter beliebigen Beleuchtungsbedingungen mit verblüffender Genauigkeit zu *generieren*. Ein ähnlich generativer Ansatz für den Bereich der Sprachtechnologie wäre ein wahrer Segen.

Zum Schluss bleibt natürlich noch der erneute Hinweis auf die Verwandtschaft der hier entwickelten Verfahren zur Spracherkennung: Mit vergleichsweise wenigen Änderungen wäre es so möglich, nicht nur den Sprecher, sondern auch das Gesagte zu erkennen. Die SC_Lib könnte so nicht nur zur Indizierung von Videos eingesetzt werden, sondern auch deren Inhalt transkribieren und in Annäherung an [VISW 00] einen Volltextzugriff ermöglichen. Doch diese Ideen bleiben einer anderen Arbeit vorbehalten...

[3] Die Programmiersprache *Java* wird innerhalb von Mediana für die Oberflächengestaltung genutzt, während Algorithmen eher in C++ gehalten sind.

Anhang A

Mathematische Herleitungen

A.1 Mehrdimensionale Gauß-Dichte-Funktion mit diagonaler Kovarianz-Matrix

Im Folgenden wird gezeigt, wie sich die mehrdimensional Gauß-Dichtefunktion $b(\vec{x})$ im Fall einer diagonalen Kovarianzmatrix komponentenweise eindimensional berechnen lässt. Hierzu wird zuerst eine Hilfsfunktion $f(\vec{y})$ benötigt:

$$b(\vec{x}) = \left(\frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}}\right) e^{\underbrace{\left(-\frac{1}{2}\underbrace{(\vec{x}-\vec{\mu})}_{=\vec{y}}{}'\Sigma^{-1}\underbrace{(\vec{x}-\vec{\mu})}_{=\vec{y}}\right)}_{=f(\vec{y})}} \tag{A.1}$$

$f(\vec{y})$ lässt sich nun wie folgt umformen:

$$f(\vec{y}) = -\frac{1}{2}\cdot(y_1, y_2, \ldots, y_D)\cdot\begin{pmatrix}\sigma_1^{-2} & 0 & \cdots & 0\\ 0 & \sigma_2^{-2} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \sigma_D^{-2}\end{pmatrix}\cdot\begin{pmatrix}y_1\\ y_2\\ \vdots\\ y_D\end{pmatrix} \tag{A.2}$$

$$= -\frac{1}{2}\cdot(y_1, y_2, \ldots, y_D)\cdot\begin{pmatrix}\sigma_1^{-2}\cdot y_1\\ \sigma_2^{-2}\cdot y_2\\ \vdots\\ \sigma_D^{-2}\cdot y_D\end{pmatrix} \tag{A.3}$$

$$= -\frac{1}{2}\sum_{d=1}^{D}\sigma_d^{-2}\cdot y_d^2 \tag{A.4}$$

$$= \sum_{d=1}^{D} -\frac{1}{2} \cdot \frac{(x_d - \mu_d)^2}{\sigma_d^2} \tag{A.5}$$

D ist hierbei die Dimension der Merkmalsvektoren $\vec{x}$. Eingesetzt in die Ursprungsformel ergibt sich nun:

$$b(\vec{x}) = \left(\frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \right) \cdot e^{\sum_{d=1}^{D} \frac{-(x_d-\mu_d)^2}{2 \cdot \sigma_d^2}} \tag{A.6}$$

$$= \left(\frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \right) \cdot e^{\frac{-(x_1-\mu_1)^2}{2 \cdot \sigma_1^2} \cdot \frac{-(x_2-\mu_2)^2}{2 \cdot \sigma_2^2} \cdot \ldots \cdot \frac{-(x_D-\mu_D)^2}{2 \cdot \sigma_D^2}} \tag{A.7}$$

$$= \left(\frac{1}{(2\pi)^{D/2}} \cdot \frac{1}{\sqrt{\Sigma}} \right) \cdot e^{\frac{-(x_1-\mu_1)^2}{2 \cdot \sigma_1^2} \cdot \frac{-(x_2-\mu_2)^2}{2 \cdot \sigma_2^2} \cdot \ldots \cdot \frac{-(x_D-\mu_D)^2}{2 \cdot \sigma_D^2}} \tag{A.8}$$

$$= \left(\frac{1}{\prod_{d=1}^{D} \sqrt{2\pi}} \cdot \frac{1}{\sqrt{\sigma_1^2 \cdot \sigma_2^2 \cdot \ldots \cdot \sigma_D^2}} \right) \cdot e^{\frac{-(x_1-\mu_1)^2}{2 \cdot \sigma_1^2} \cdot \frac{-(x_2-\mu_2)^2}{2 \cdot \sigma_2^2} \cdot \ldots \cdot \frac{-(x_D-\mu_D)^2}{2 \cdot \sigma_D^2}} \tag{A.9}$$

$$= \prod_{d=1}^{D} \left(\frac{1}{\sqrt{2\pi} \cdot \sqrt{\sigma_d^2}} \right) \cdot e^{\left(\frac{-(x_d-\mu_d)^2}{2 \cdot \sigma_d^2} \right)} \tag{A.10}$$

$$= \prod_{d=1}^{D} b(x) \tag{A.11}$$

Wobei $b(x)$ die eindimensionale Gauß-Dichtefunktion ist.

A.2 Bayes'sches Informations-Kriterium

Das in [CHEN 98] und [TRIT 99] entwickelte ΔBIC[1] zur Verwendung als Abbruchkriterium in agglomerariven Clusteringprozessen wird von Ajmera et al. in [AJME 03] so verändert, dass der aufgabenspezifisch anzupassende Strafterm $\alpha \cdot P$ entfällt. Hierdurch wird das Verfahren schwellwertfrei, was es um so interessanter macht für die Verwendung in Systemen, welche auf Robustheit und Generalität aus sind. Leider ist es durch einige Schreibfehler in [AJME 03] etwas schwierig, die Entwicklung der Formel auf einen Blick nachzuvollziehen. Deshalb soll die Entwicklung hier nochmals vollständig und korrekt dargestellt werden:

Zu Anfang sollen nochmals die Formeln für den Startpunkt und das zu erreichende Ziel dargestellt werden: Ausgegangen wird vom BIC, das einen Wert für die Tauglichkeit eines Modells λ für die Beschreibung eines Datensatzes X in Abhängigkeit von seiner

[1] Wie das vorangestellte Δ bereits andeutete, handelt es sich beim ΔBIC direkt um die Differenz zweier BIC-Werte.

Komplexität liefert. Ziel soll ein Ausdruck sein, der angibt, ob zwei verschiedene Datensätze besser durch ein komplettes Modell beschrieben werden, oder ob getrennte Modelle die Wirklichkeit besser abbilden:

$$BIC(\lambda) = \log L(X|\lambda) - \frac{\alpha}{2} \cdot len(\lambda) \cdot \log N_X \tag{A.12}$$

$$\Delta BIC(\lambda_1, \lambda_2) = BIC(\lambda_1) - BIC(\lambda_2) \tag{A.13}$$

$$\Delta BIC(\lambda_1, \lambda_2) \geq 0 \Rightarrow \lambda_1 \text{ präferrieren, sonst } \lambda_2 \tag{A.14}$$

λ_1 steht dabei für die Modellierung der zwei Datensätze X und Y mit einem Gesamtmodell ($\lambda_1 = \lambda_{X \cup Y}$), während λ_2 für die Modellierung mittels zweier Modelle steht ($\lambda_2 = \{\lambda_x, \lambda_y\}$). Zur Erinnerung sei nochmals gesagt, dass $len(\lambda)$ die Komplexität des Modells λ und N_Z die Anzahl Merkmalsvektoren im Datensatz Z benennt. Daher folgt:

$$BIC(\lambda_1) = \log L(X \cup Y|\lambda_1) - \frac{\alpha}{2} \cdot len(\lambda_1) \cdot \log N_{X \cup Y} \tag{A.15}$$

$$BIC(\lambda_2) = \log L(X|\lambda_x) + \log L(Y|\lambda_y) - \frac{\alpha}{2} \cdot \left[\underbrace{len(\lambda_x) + len(\lambda_y)}_{=len(\lambda_2)} \right] \cdot \log \underbrace{(N_X + N_Y)}_{=N_{X \cup Y}} \tag{A.16}$$

Damit lässt sich Gleichung A.13 folgendermaßen umstellen:

$$\begin{aligned} \Delta BIC(\lambda_1, \lambda_2) &= \left[\log L(X \cup Y|\lambda_1) - \frac{\alpha}{2} \cdot len(\lambda_1) \cdot \log N_{X \cup Y} \right] \\ &\quad - \left[\log L(X|\lambda_x) + \log L(Y|\lambda_y) - \frac{\alpha}{2} \cdot len(\lambda_2) \cdot \log N_{X \cup Y} \right] \qquad \text{(A.17)} \\ &= \log L(X \cup Y|\lambda_1) - [\log L(X|\lambda_x) + \log L(Y|\lambda_y)] \\ &\quad - \frac{\alpha}{2} \cdot len(\lambda_1) \cdot \log N_{X \cup Y} \\ &\quad + \frac{\alpha}{2} \cdot [len(\lambda_x) + len(\lambda_y)] \cdot \log N_{X \cup Y} \qquad \text{(A.18)} \\ &= \log L(X \cup Y|\lambda_1) - [\log L(X|\lambda_x) + \log L(Y|\lambda_y)] \\ &\quad - \frac{\alpha}{2} \cdot [len(\lambda_1) - len(\lambda_2)] \cdot \log N_{X \cup Y} \qquad \text{(A.19)} \end{aligned}$$

Da ein Kriterium gesucht wird, anhand dessen λ_1 auszuwählen oder zu verwerfen ist, findet die folgende Umformung der Gleichung A.19 in eine Ungleichung in Einklang mit Formel A.14 statt:

$$\log L(X \cup Y|\lambda_1) - [\log L(X|\lambda_x) + \log L(Y|\lambda_y)] - \frac{\alpha}{2} \cdot [len(\lambda_1) - len(\lambda_2)] \cdot \log N_{X \cup Y} \geq 0 \tag{A.20}$$

$$\log L(X \cup Y|\lambda_1) - [len(\lambda_1) - len(\lambda_2)] \geq \log L(X|\lambda_x) + \log L(Y|\lambda_y) \tag{A.21}$$

Angenommen, λ_1 enstand nun durch einfache Konkatenation der Komponenten von λ_x und λ_y, oder es gilt aus sonst einem Grund $len(\lambda_1) = len(\lambda_2)$ [2], dann reduziert sich Formel A.21 zu:

$$\log L(X \cup Y|\lambda_1) \geq \log L(X|\lambda_x) + \log L(Y|\lambda_y) \tag{A.22}$$

Dieses Resultat stimmt nun wieder mit [AJME 03] überein.

A.3 Zusammenhang zwischen Standard- und parametrisierter Normalverteilung

Im Folgenden soll gezeigt werden, dass eine standardisierte Gauß-Dichte- und Verteilungsfunktion, welche ein mittels μ und σ z-transformiertes Argument übergeben bekommt, identisch ist zu ihrem mit μ und σ parametrisierten Pendant, welches das untransformierte Argument erhält. Zuerst soll die Verteilungsfunktion $\Phi()$ der Standard-Normalverteilung $N(0,1)$ betrachtet werden. Dieser Nomenklatur folgend bezeichnet $\phi()$ die zugehörige Dichtefunktion, während $B()$ und $b()$ für die parametriserten Varianten $N(\mu, \sigma)$ stehen sollen:

$$\Phi\left(\frac{z-\mu}{\sigma}\right) = \int_{-\infty}^{\frac{z-\mu}{\sigma}} \phi(x)dx \tag{A.23}$$

$$= \int_{-\infty}^{\frac{z-\mu}{\sigma}} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{x^2}{2}} dx \tag{A.24}$$

$$= \frac{1}{2} \cdot \left(erf(\frac{z-\mu}{\sigma} \cdot \frac{\sqrt{2}}{2}) + 1 \right) \tag{A.25}$$

Der letzte Schritt folgt dabei nicht direkt, sondern wurde mittels des Computer-Algebra-Programms "Derive 6" [TEXA 03] nachfollzogen. Die Funktion $erf()$[3] repräsentiert hier und im weiteren Verlauf das Gauß'sche Fehlerintegral. Unter der Annahme, dass $\sigma > 0$ gilt, folgt weiter:

$$\frac{1}{2} \cdot \left(erf(\frac{z-\mu}{\sigma} \cdot \frac{\sqrt{2}}{2}) + 1 \right) = \frac{1}{\sigma} \cdot \frac{1}{2} \cdot \left(erf(\frac{z-\mu}{\sigma} \cdot \frac{\sqrt{2}}{2}) \cdot \sigma + \sigma \right) \tag{A.26}$$

[2] Anschaulich heißt das am Beispiel eines spezifischen GMM, dass das resultierende Modell zehn Mischkomponenten haben muss, wenn seine Vorgänger jeweils sechs und vier Komponenten hatten.

[3] Aus dem englischen für *Gaussian Error Function*.

$$= \frac{1}{\sigma} \cdot \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}} dx \quad (A.27)$$

$$= \frac{1}{\sigma} \cdot \int_{-\infty}^{z} \phi(\frac{x-\mu}{\sigma}) dx \quad (A.28)$$

$$= \int_{-\infty}^{z} \frac{1}{\sigma} \cdot \phi(\frac{x-\mu}{\sigma}) dx \quad (A.29)$$

$$= \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{\frac{-(x-\mu)^2}{2\sigma^2}} dx \quad (A.30)$$

$$= \int_{-\infty}^{z} b(x) dx \quad (A.31)$$

$$= B(z). \quad (A.32)$$

Auf ähnliche Weise lässt sich für die Dichtefunktion vorgehen:

$$\phi\left(\frac{z-\mu}{\sigma}\right) = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{\left(\frac{z-\mu}{\sigma}\right)^2}{2}} \quad (A.33)$$

$$= \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{\left(\frac{z^2}{\sigma^2} - \frac{2z\mu}{\sigma^2} + \frac{\mu^2}{\sigma^2}\right)}{2}} \quad (A.34)$$

$$= \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(z-\mu)^2}{2\sigma^2}} \quad (A.35)$$

$$= \sigma \cdot \frac{1}{\sqrt{2\pi} \cdot \sigma} \cdot e^{-\frac{(z-\mu)^2}{2\sigma^2}} \quad (A.36)$$

$$= \sigma \cdot b(z). \quad (A.37)$$

Es bleibt also als Ergebnis festzuhalten, dass gilt:

$$\Phi\left(\frac{z-\mu}{\sigma}\right) = B(z) \quad (A.38)$$

$$\phi\left(\frac{z-\mu}{\sigma}\right) = \sigma \cdot b(z) \quad (A.39)$$

A.4 Der Formelapparat des GMM-IB im Überblick

An dieser Stelle sollen die über weite Bereiche dieser Arbeit verstreuten Formeln des GMM-IB nochmals übersichtlich und vor allem zusammenhängend präsentiert werden. Dies mag dem schnelleren Auffinden und dem raschen Nachschlagen dienlich sein, wenn weiter oben auf bestimmte Gleichungen verwiesen wird.

Das Modell selber ist definiert als

$$\lambda_s = \{p_i, \vec{\mu_i}, \Sigma_i\}, \text{i=1..M} \tag{A.40}$$

Des Weiteren existiert ein explizites GMM für den akustischen Hintergrund, das ähnlich definiert ist:

$$\lambda_b = \{q_j, \vec{\mu_j}, \Sigma_j\}, \text{j=1..N} \tag{A.41}$$

In der Praxis (und im weiteren Verlauf dieser Formeln) degenerieren die Kovarianzmatrizen Σ jedoch zu reinen Varianzvektoren $\vec{\sigma}^2$. Die Wahrscheinlichkeit, mit der eine Menge Z an Merkmalsvektoren von diesem Modell erzeugt wurde[4], ist gegeben durch:

$$p(Z|\lambda_s, \lambda_b) = \prod_{t=1}^{T} p(\vec{z_t}|\lambda_s, \lambda_b) \tag{A.42}$$

$$p(\vec{z_t}|\lambda_s, \lambda_b) = \sum_{i=1}^{M}\sum_{j=1}^{N} p_i \cdot q_j \cdot p(\vec{z_t}|i, j, \lambda_s, \lambda_b) \tag{A.43}$$

$$p(\vec{z_t}|i_t, j_t, \lambda_s, \lambda_b) = \int\int_C b_{i_t}(\vec{x_t}) \cdot a_{j_t}(\vec{y_t}) d\vec{x_t} d\vec{y_t} \tag{A.44}$$

$\vec{x_t}$ und $\vec{y_t}$ sind hierbei die nicht beobachtbaren reinen Formen des Signals und des Geräuschs zum Zeitpunkt t, $\vec{z_t}$ ist die daraus resultierende Beobachtung. Im Folgenden wird aus Gründen der Vereinfachung zusammenfassend von $\lambda = \{\lambda_s, \lambda_b\}$ gesprochen. Unter Verwendung der Wahrscheinlichkeit dafür, dass die Modellkomponenten k und l des Vorder- und Hintergrundprozesses das Auftreten des Merkmalsvektors $\vec{z_t}$ beschreiben

$$p(i_t = k, j_t = l|\vec{z_t}, \lambda) = \frac{p(\vec{z_t}|i_t = k, j_t = l, \lambda) \cdot p_i \cdot q_j}{\sum_{i=1}^{M}\sum_{j=1}^{N} p(\vec{z_t}|i_t = i, j_t = j, \lambda) \cdot p_i \cdot q_j} \tag{A.45}$$

ergeben sich folgende Gleichungen für den iterativen Annäherungsprozess der Modellkomponenten während des EM-Algorithmus [5]:

$$\overline{p_i} = \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|\vec{z_t}, \lambda) \tag{A.46}$$

$$\overline{\mu_i} = \frac{\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|z_t, \lambda) \cdot E\{x_t|z_t, i_t{=}i, j_t{=}j, \lambda\}}{\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|z_t, \lambda)} \tag{A.47}$$

$$\overline{\sigma_i}^2 = \frac{\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|z_t, \lambda) \cdot E\{x_t^2|z_t, i_t{=}i, j_t{=}j, \lambda\}}{\sum_{t=1}^{T}\sum_{j=1}^{N} p(i_t{=}i, j_t{=}j|z_t, \lambda)} - \overline{\mu_i}^2 \tag{A.48}$$

[4] Oder im Umkehrschluss: Die Wahrscheinlichkeit, mit der das Modell die wirkliche Quelle von Z beschreibt.

[5] Unter der Prämisse, dass jeweils eine diagonale Kovarianzmatrix $\vec{\sigma}^2$ anstatt der vollen Σ Verwendung findet. Dadurch kann jede Vektorkomponente einzeln errechnet werden, so dass im weiteren Verlauf die Vektorpfeile vernachlässigt werden und die Formeln jeweils für einzelne Vektorkomponente gelten.

Bis jetzt waren diese Formeln allgemein gehalten, es wurde keine spezielle Art der Signal-Geräusch-Interaktion vorausgesetzt. Die konkreten Lösungen beziehen sich nun jedoch explizit auf das für diese Arbeit relevante *Max-Modell*. Unter dieser Annahme wird Gleichung A.44 zu:

$$a(z_t) = b(z_t) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot e^{-\frac{(z_t-\mu)^2}{2\cdot\sigma^2}} \tag{A.49}$$

$$A(z_t) = B(z_z) = \int_{-\infty}^{z_t} c(x_t)dx_t \quad \text{wobei} \quad c(x_t)\in\{a(x_t), b(x_t)\} \tag{A.50}$$

$$p(z_t|i_t{=}i, j_t{=}j, \lambda) = a_j(z_t)\cdot B_i(z_t) + b_i(z_t)\cdot A_j(z_t) \tag{A.51}$$

Somit folgt zur Vorbereitung der bedingten Erwartungswerte in den Formeln A.47 und A.48:

$$p(x_t{=}z_t|i_t{=}i, j_t{=}j, \lambda) = \frac{b_i(z_t)\cdot A_j(z_t)}{a_j(z_t)\cdot B_i(z_t) + b_i(z_t)\cdot A_j(z_t)} \tag{A.52}$$

$$E\{x_t|x_t{<}z_t, i_t{=}i, j_t{=}j, \lambda\} = \mu - \sigma^2\cdot\frac{b_i(z_t)}{B_i(z_t)} \tag{A.53}$$

$$E\{x_t^2|x_t{<}z_t, i_t{=}i, j_t{=}j, \lambda\} = \left(\mu^2+\sigma^2\right) - \sigma^2\cdot\frac{b_i(z_t)\cdot(z_t+\mu_i)}{B_i(z_t)} \tag{A.54}$$

und schließlich:

$$\begin{aligned} E\{x_t|z_t, i_t{=}i, j_t{=}j, \lambda\} &= p(x_t{=}z_t|i_t{=}i, j_t{=}j, \lambda)\cdot z_t \\ &\quad + ((1-p(x_t{=}z_t|i_t{=}i, j_t{=}j, \lambda)) \\ &\quad \cdot E\{x_t|x_t{<}z_t, i_t{=}i, j_t{=}j, \lambda\}\ ) \end{aligned} \tag{A.55}$$

$$\begin{aligned} E\{x_t^2|z_t, i_t{=}i, j_t{=}j, \lambda\} &= p(x_t{=}z_t|i_t{=}i, j_t{=}j, \lambda)\cdot z_t^2 \\ &\quad + ((1-p(x_t{=}z_t|i_t{=}i, j_t{=}j, \lambda)) \\ &\quad \cdot E\{x_t^2|x_t{<}z_t, i_t{=}i, j_t{=}j, \lambda\}\ ) \end{aligned} \tag{A.56}$$

Anhang B

Graphische Benutzerschnittstelle

B.1 Zur Entstehung

Während des Beginns der Experimentierphase stellte sich rasch heraus, dass eine Analyse der von SCiVo als Endergebnis erzeugten Ablaufprotokolle per Hand beschwehrlich und vor allem zeitaufwändig werden würde. Denn um auf einen Blick zu erfassen, ob die Klassifikation korrekt vollzogen wurde, wäre es jeweils nötigt, den als vollständig geklammerten Ausdruck kodierten Clusterverschmelzungsbaum vor dem geistigen Auge wieder zusammenzusetzen:

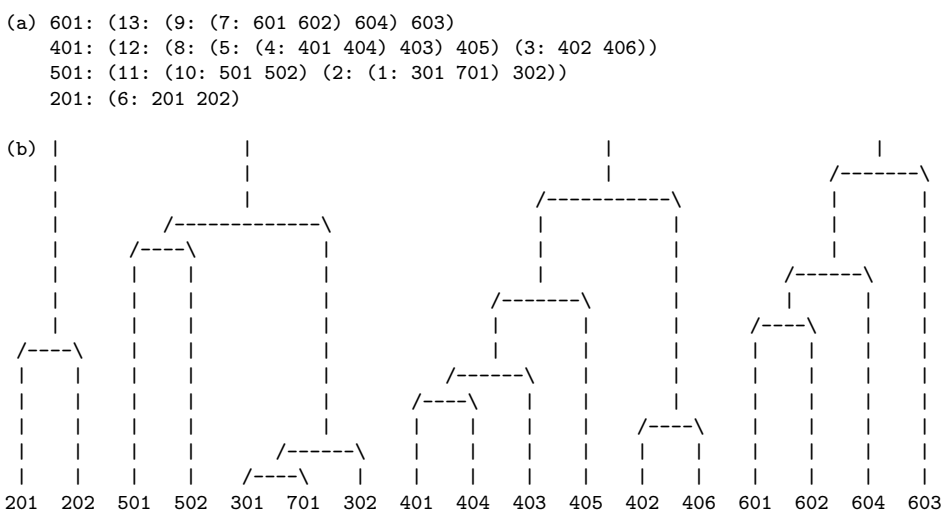

Abbildung B.1: Verlauf eines Clusteringprozesses als geklammerter Ausdruck kodiert (a) und visualisiert in Bäumen (b).

Um an dieser Stelle Zeit zu sparen, wurde die Idee in die Tat umgesetzt, ein kleines Programm mit grafischer Benutzeroberfläche zu gestalten, welches die Aufgabe der Rekonstruktion des Clusterverschmelzungsbaums wahrnehmen sollte. Und da es ursprünglich

tatsächlich nur zur Visualisierung der Baumstruktur, wie Abbildung B.1 sie exemplarisch darstellt, gedacht war, stand auch der Name dieses Hilfsprogramms bald fest: "*Gardener*" - der Gärtner[1].

Als Entwicklungswerkzeug wurde Visual Basic gewählt, da es starke Qualitäten in der schnellen Anwendungsentwicklung [2] vor allem in Verbindung mit grafischen Benutzeroberflächen hat. Und so entstand der erste Prototyp ganz ohne Ambitionen, ein vollständiges Frontend zu sein. Schnell wurden im täglichen Testbetrieb jedoch noch weitere Funktionen notwendig, so beispielsweise die Anzeige eines Zeitstrahl, auf welchem die gefundenen Sprecher farblich[3] markiert abgelesen werden konnten. Und so war es schließlich nur konsequent, die Benutzerführung mittels Tastaturkommandos und Robustheit gegenüber Eingabefehlern zu verfeinern und eine Methode zur Fernsteuerung von SCiVo hinzuzufügen, um den evolutionären Prototypen in eine runde Sache zu verwandeln.

Am Ende dieser Entwicklung steht nun ein Programm, welches zwar keinesfalls den Anspruch erhebt, eine vollständige und ausgereift Benutzeroberfläche für die in dieser Arbeit erstellten Sprecherklassifikations-Verfahren zu sein. Es bietet jedoch genug Funktionalität, um dem Benutzer des Programms die Testfallgenerierung und Ergebnisinterpretation erheblich zu vereinfachen und etwas mehr Anschauung und Bedienbarkeit in die Welt spröden Kommandozeilencharmes zu bringen. Im Folgenden soll eine kurze Einführung in diese Möglichkeiten gegeben werden:

B.2 Funktionalität

Als Kernfunkionalität bietet Gardener wie gesagt die Möglichkeit, die als Endergebnis von SCiVo generierten Ablaufprotokolle[4] der Klassifikation zu analysieren. Ein beispielhaftes Ablaufprotokoll könnte dabei wie in Abbildung B.2 dargestellt aussehen.

Zu Beginn wird der Name der analysierten Audiodatei genannt, gefolgt von einer Liste sämtlicher veränderlicher Parameter im Programm und ihrer Werte zum Zeitpunkt des Durchlaufs[5]. In der Titelzeile erscheint noch die für den Durchlauf benötigte Zeit in Minuten. Unterhalb der Parameter befindet sich die schon erläuterte kodierte Form des Clusterverschmelzungsbaums, und im Anschluss folgt eine Liste, welche jeweils die Beschreibungen zu wichtigen Merkmalsrahmen dieser Audiodatei liefert: Für alle Rahmen,

[1] Denn anschaulich lässt es Bäume wachsen.

[2] Im Englischen *RAD*, *Rapid Application Development*.

[3] An dieser Stelle sei Herrn Martin Schwalb ganz herzlich für die Beisteuerung des Algorithmus zur Farbgebung gedankt.

[4] SCiVo kann Ablaufprotokolle fortlaufend in eine einzige Datei schreiben. Gardener bietet deshalb die Möglichkeit, einzelne Durchläufe aus einem Gesamtprotokoll anhand ihres Erstellungsindex direkt auszuwählen.

[5] Die Abbildung zeigt ein mit einer älteren Programmversion generiertes Protokoll, weswegen noch nicht sämtliche vorher besprochenen Parameter aufgeführt sind. Viele fanden erst später Einzug in das System.

```
-----------------------------------------------=<{     4:25 }>=-----------------------------------------------
../data/news2.wav

audioFrameSize:             10
audioFrameStep:             5
energyQuantizationLevel:    10
pauseSilenceThreshold:      500
filterBankSize:             60
FFTsize:                    512
dEnergy:                    0
windowed:                   1
preEpmhasizeFactor:         0.97
MFCCorder:                  10
MFCCcoeffSelection:         2046
CMN:                        0
lowCut:                     0
highCut:                    8000
segmentLengthThreshold:     1500
lastNdistances:             10
adaptiveThresholdAlpha:     1.1
percentDifference:          5
deltaBIClambda:             1
maxMixtures:                32
varianceLimit:              0.01
EMthreshold:                1
maxNoiseModelOrder:         0
maxSpeakerModelOrder:       8
vectorsPerGaussian:         200
distanceMeasure:            1
terminationCriterion:       0

601: (13: (9: (7: 601 602) 604) 603)
401: (12: (8: (5: (4: 401 404) 403) 405) (3: 402 406))
501: (11: (10: 501 502) (2: (1: 301 701) 302))
201: (6: 201 202)
-------------------------------------------------------------------------------------------------------------
FRAME#  VIDEO#  NOISE   SPEECH  PAUSE   UNVOIC  SILENCE SCENE_B SPK_B   SHORT   START   END     INSG.   SPK_ID
0           0    1       0       0       0       1       1       0       0       0       0       49      0
696        87    1       0       0       0       1       1       0       0       0       0       49      0
1040      130    0       1       0       0       0       0       1       0       1       0       322     201
4671      583    1       0       0       0       1       0       0       0       0       1       529     201
4672      584    0       1       1       0       0       0       0       0       1       0       262     201
7127      890    1       0       0       0       0       0       0       0       0       1       513     201
7128      891    1       0       0       0       1       1       0       0       0       0       49      0
7280      910    0       1       1       0       0       0       1       0       1       0       326     501
16563    2070    0       0       0       0       1       0       0       0       0       1       528     501
16564    2070    0       1       0       0       0       0       0       0       1       0       258     501
18271    2283    1       0       0       0       0       0       0       0       0       1       513     501
18272    2284    1       0       0       0       1       1       0       0       0       0       49      0
18424    2303    0       1       1       0       0       0       1       0       1       0       326     401
21704    2713    0       0       0       1       1       0       0       0       0       1       536     401
21705    2713    0       1       0       0       0       0       0       0       1       0       258     401
22847    2855    1       0       0       0       1       0       0       0       0       1       529     401
22848    2856    0       1       1       0       0       0       0       0       1       0       262     401
23559    2944    1       0       0       0       1       0       0       0       0       1       529     401
23560    2945    0       1       1       0       0       0       0       0       1       0       262     401
27121    3390    0       0       0       1       1       0       0       0       0       1       536     401
27122    3390    0       1       0       0       0       0       0       0       1       0       258     401
29395    3674    0       0       0       1       1       0       0       0       0       1       536     401
29396    3674    0       1       0       0       0       0       0       0       1       0       258     401
33967    4245    1       0       0       0       0       0       0       0       0       1       513     401
33968    4246    1       0       0       0       1       1       0       0       0       0       49      0
34208    4276    0       1       1       0       0       0       1       0       1       0       326     501
36103    4512    0       0       0       1       1       0       0       0       0       1       536     501
36104    4513    0       1       0       0       0       0       0       0       1       0       258     501
39047    4880    1       0       0       0       0       0       0       0       0       1       513     501
39048    4881    1       0       0       0       1       1       0       0       0       0       49      0
39064    4883    0       1       1       0       0       0       1       0       1       0       326     601
41743    5217    0       0       0       1       1       0       0       0       0       1       536     601
41744    5218    0       1       0       0       0       0       0       0       1       0       258     601
42392    5299    0       0       0       1       1       0       0       0       0       1       536     601
42393    5299    0       1       0       0       0       0       0       0       1       0       258     601
45468    5683    0       0       0       1       1       0       0       0       0       1       536     601
45469    5683    0       1       0       0       0       0       0       0       1       0       258     601
46015    5751    1       0       0       0       0       0       0       0       0       1       513     601
46016    5752    1       0       0       0       1       1       0       0       0       0       49      0
46168    5771    0       1       1       0       0       0       1       0       1       0       326     501
48990    6123    1       0       0       0       0       0       0       0       0       1       513     501
```

Abbildung B.2: Ablaufprotokoll eines Klassifikationslaufs mit SCiVo.

welche Szenen- oder Sprachsegment-Grenzen darstellen, sind deren Audio- und Videorahmennummer und binär kodiert alle über sie im Programmverlauf gewonnen Informationen

dargestellt.

Am Ende der Zeile wird die eindeutige Nummer des Sprechers des Audiosegments genannt. Hierbei bedeutet eine Null, dass das entsprechende Segment zu kurz zum Verarbeiten war, während eine dreistellige Zahl Hinweis auf das Auftreten eines der Urprungssegmente eines Clusters gibt: Jedem Segment wird anfangs eine Sprecher-ID zugewiesen, deren erste Stelle die Szene und deren letzten beiden Stellen die Segmentnummer des betrachteten Segments innerhalb dieser Szene angeben. Doch natürlich verliert diese Darstellung durch die Verschmelzung mehrerer Segmente ihren Informationsgehalt, denn bei der Verheiratung zweier Cluster muss einer die ID des anderen übernehmen, ohne natürlich in der durch sie kodierten Position jemals seinen Ursprung gefunden zu haben.

Gardener bereitet diese zwar vollständige, doch etwas unübersichtliche Informationsflut nun so auf, dass sie beinahe mit einem Blick erfassbar wird. Abbildung B.3 stellt dies dar:

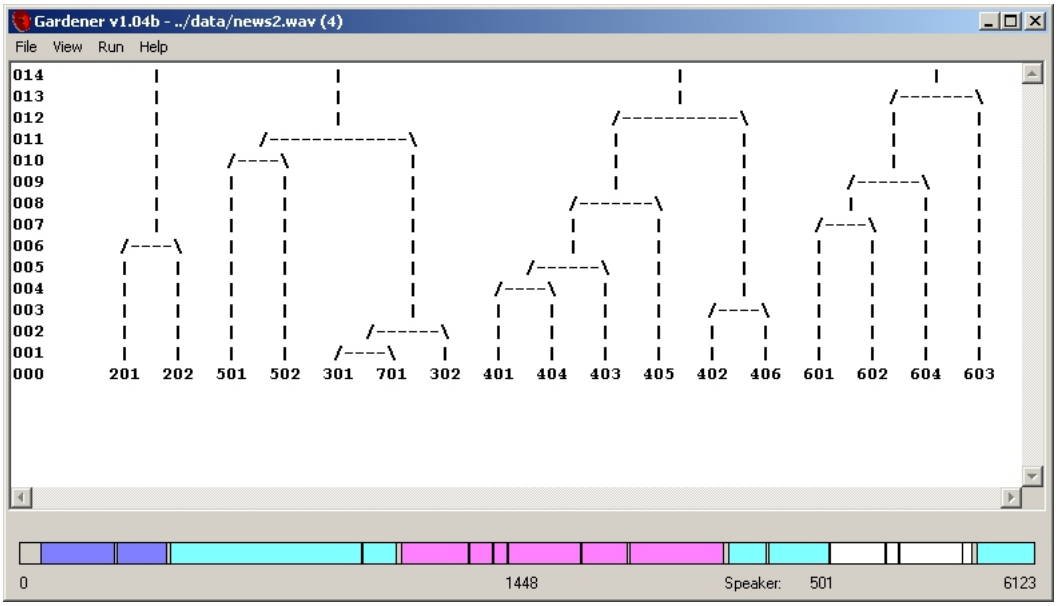

Abbildung B.3: Obiges Ablaufprotokoll in aufbereiteter Form visualisiert in Gardener.

Den Hauptbereich des Programmfensters nimmt die Fläche zur Darstellung des Verschmelzungsbaums[6] ein. Darunter ist der Zeitstrahl zu sehen: In Videorahmeneinheiten skaliert, markiert er alle sprecherhomogenen Segmente in der gleichen Farbe. Beim Überfahren mit der Maus wird der aktuelle Rahmen und dessen Sprecher-ID unterhalb des Zeitstrahls angezeigt. Durch markieren eines Abschnitts mit der Maus läßt sich in die Dartsellung hineinzoomen, um bei einer langen Audiodatei auch die Details beobachten zu können.

[6]Wurden mehrere Sprecher in dem Audiostrom gefunden, existiert ein Baum pro Sprecher.

Daneben existieren noch die Möglichkeiten, sich die Parameter des visualiserten Durchlaufs anzeigen zu lassen oder einen neuen Durchlauf zu starten, wobei in letzterem Fall zuvor sämtliche Parameter der SC_Lib komfortabel verändert oder auf Standardwerten belassen werden können. Die entsprechenden Kommandos finden sich im Menü oder lassen sich per Tastatur direkt aufrufen. Eine Liste aller verfügbarer Tastaturkürzel enthält dabei die integrierte Hilfe, welche über die Tastenkombination `F1` zu erreichen ist.

Damit das Programm Gardener läuffähig ist, benötigt es noch die Datei `gardener.ini`. In ihr sind zum einen zwei Optionen für die Programmbenutzung hinterlegt: Der Pfad zu SCiVo und der Pfad zu den Analyseergebnissen, die auf Tastendruck direkt geöffnet werden sollen. Zum anderen enthält die Datei jedoch auch eine vollständige Liste aller von SCiVo entgegennehmbaren Kommandozeilenparameter inklusive einer jeweiligen Wirkungsbeschreibung. Diese kann bei Bedarf um beliebige weitere Parameter oder andere Beschreibungen ergänzt werden und wird beim Start von SCiVo aus der Oberfläche heraus verwendet: Alle in der Ini-Datei unter dem Abschnitt `[PARAM]` aufgeführten Parameter erscheinen in dem Startdialog als veränderbare Werte und werden beim Start in der Reihenfolge ihres Auftretens an SCiVo übergeben. Dabei ist die Verwendung des Eintrags für `TYPE_*` momentan nur reserviert[7], wird jedoch noch nicht verwendet.

[7]Hiermit könnte in Zukunft eine Typprüfung der Parameter stattfinden, denn die angegebenen Zahlen korrespondiern mit den Visual Basic-Konstanten für Datentypen wie `vbInteger` oder `vbBool` entsprechend den erwarteten Parametern im aufgerufenen Programm SCiVo.

Anhang C

Organisation des beiliegenden Datenträgers

Die vorliegende Arbeit wird begleitet von einem Datenträger, welcher das erstellte Softwaresystem und - zum Zwecke der Veranschaulichung - die verwendeten Testdaten enthält. Bevor sich die folgende Liste detaillierter mit dem Inhalt der CD beschäftigt, mag hier noch ein wichtiger Hinweis zur Verwendung der darauf befindlichen Daten vermerkt werden:

Die untersuchten Videos unterliegen dem Urherberrecht und sind Eigentum ihrer jeweiligen Rechteinhaber. Dasselbe gilt für die hier beigefügten Tonspuren dieser Videos: Sie sind ausschließlich für Forschungs- und Lehrzwecke bestimmt und dürfen weder weitergegeben noch zweckentfremdet eingesetzt werden!

Die Rechte an der Bibliothek SV_Lib *liegen bei Dr. Jialong He. Sie darf frei verwendet werden, zur Benutzung des (hier in veränderter Form vorliegenden) Quelltextes ist jedoch das Einverständnis des Autors unter* `jhe@asu.edu` *einzuholen.*

Die Programme Gardener, SCiVo *und* SC_Lib *sowie ihr Quelltext und dessen Dokumentation unterliegen dem Copyright des Autors. Sie dürfen beliebig verwendet, erweitert, verändert und verbreitet werden. Der Autor bittet lediglich um einen kleinen Hinweis unter* `thilo.stadelmann@gmx.de`*, falls die Software irgendwo von Nutzen sein kann.*

Nach diesen leider notwendigen rechtlichen Aspekten kann sich nun dem Inhaltsverzeichnis des Datenträgers zugewandt werden: Angegeben ist die Verzeichnisstruktur mit einer kurzen Beschreibung des Inhalts:

- `readme.txt`: Beinhaltet eine kurze Einführung zum Umgang mit der CD, welche im Wesentlichen dem Inhalt dieses Kapitels folgt.
- `\data`: Enthält die Tonspuren aller verwendeter Videos sowie die zu deren Untersuchung notwendigen Segmentierungslisten.

  - `\test`: Enthält in diversen Unterordnern die Rohergebnisse der in Abschnitt 4.2 dargelegten Experimente sowie die zu deren Erstellung angefertigten Test-Skripte.

- `\diplomarbeit`: Enthält dieses Dokument im Portable Document Format (PDF).
- `\gardener`: Enthält alle Daten das Programm Gardener betreffend.
  - `\bin`: Die fertig übersetzte, lauffähige Version.
  - `\src`: Der Quelltext und alle weiteren zur Entwicklung benötigten Daten als Projekt für Microsoft Visual Basic 6.
- `\scivo`: Beinhaltet alle Daten das Programm SCiVo betreffend.
  - `\bin`: Die fertig übersetzte, lauffähige Version.
  - `\src`: Der Quelltext Projekt für Microsoft Visual Studio 6.
- `\sclib`: Beinhaltet alle Daten die Bibliothek SV_Lib betreffend.
  - `\doc`: Die API-Dokumentation im HTML-Format.
  - `\include`: Die endgültigen Header-Dateien, welche für das Einbinden und Benutzen der Bibliothek in eigenen Programmen von Nöten sind.
  - `\lib`: Die fertig übersetzte, einbindbare Bibliothek.
  - `\src`: Der Quelltext als Projekt für Microsoft Visual Studio 6.
- `\svlib`: Enthält alle Daten die Bibliothek SV_Lib betreffend.
  - `\cmd`: Einige Beispielanwendungen inklusive Quelltext.
  - `\data`: Diverse Testdaten.
  - `\doc`: Die API-Dokumentation im HTML-Format.
  - `\include`: Die endgültigen Header-Dateien, welche für das Einbinden und Benutzen der Bibliothek in eigenen Programmen von Nöten sind.
  - `\lib`: Die fertig übersetzte, einbindbare Bibliothek.
  - `\src`: Der Quelltext als Projekt für Microsoft Visual Studio 6.

Die Software ist so konfiguriert, dass sie sich direkt von CD aus dem Verzeichnis `gardener\bin\gardener.exe` starten lässt. Ergebnisse und Debug-Ausgaben werden dabei in dem Verzeichnis abgelegt, welches durch die Umgebungs-Variable[1] `%TEMP%` spezifiziert ist. Sollte diese Variable nicht gesetzt oder auf den angegebenen Pfad kein schreibender Zugriff möglich sein, kann dies zu Fehlfunktionen führen. In diesem Fall sollte vor dem Start eines Klassifikationslaufs aus Gardener heraus vor Bestätigung der Parameter ein gültiges Verzeichnis angegeben werden.

[1] Unter der Kommandozeile microsoft'scher Betriebssysteme lassen sich beliebige Variablen deklarieren, deren Werte vom Kommandointerpreter (und von Gardener) expandiert werden, wenn der Variablenname in %-Zeichen eingeschlossen ist. Die genaue Vorgehensweise zum Umgang mit solchen Variablen kann der Dokumentation des jeweiligen Betriebssystems entnommen werden.

Anhang D

Fachwörterbuch

Deutsch	*Englisch*
B	
Bayes'sches Informations Kriterium	Bayesian Information Criterion (BIC)
C	
Cepstralen Mittelwert-Normalisierung	Cepstral Mean Normalization (CMN)
Cepstralen Mittelwert-Subtraktion	Cepstral Mean Subtraction (CMS)
Cluster-Interne Streuung	Within Cluster Disperion
Cluster-Reinheit	Cluster Purity
D	
Diskriminations-Analyse	Discriminant Analysis
E	
Energie	Energy Shorttime Energy (STE)
G	
Gauß'sches Misch-Modell	Gaussian Mixture Model (GMM)
Gauß'sches Misch-Modell mit Integriertem Hintergrund	Gaussian Mixture Model with Integrated Background (GMM-IB)
Generalisiertes Likelihood-Verhältnis	Generalized Likelihood Ratio (GLR)
Geräusch-Auslöschung	Noise Cancellation
Geräusch-Maskierung	Noise Masking
Grundfrequenz	Pitch
K	
Kameraeinstellung	Shot
Kurven-Integral	Contour Integral
L	
Lineare Vorhersage	Linear Prediction (LP)
Linien-Integral	Contour Integral
M	

Deutsch	*Englisch*
Maximum-Likelihood Näherung	Maximum Likelihood (ML) Estimate
Messwert	Sample
N	
Nicht-stimmhafte Sprache	Unvoiced Speech
Nulldurchlaufsrate	Zero Crossing Rate (ZCR)
P	
Parallele Modell-Kombination	Parallel Model Combination (PMC)
R	
Rahmen	frame
S	
Schnitt	Cut
Signal-Geräusch-Abstand	Signal-to-Noise Ratio (SNR)
Signalbeeinträchtigungs-Funktion	Noise Corruption Function
Sprachanreicherung	Speech Enhancement
Sprachverarbeitung	Speech Processing
Sprecherklassifikation	Speaker Classification Speaker Indexing Speaker Tracking
Stimmhafte Sprache	Voiced Speech
Stochastische Zuordnung	Stochastic Matching (SM)
Sprecherwechsel-Erkennung	Speaker Change Detection (SCD) Speaker Segmentation
T	
Tonhöhe	Pitch
W	
Wahrscheinlichkeit	Probability Likelihood
Wahrscheinlichkeitsdichte	Probability Density Density
Wahrscheinlichkeitsdichtfuntion	Probability Density Fuction (PDF)
Wortfehlerrate	Word Error Rate (WER)

Literaturverzeichnis

[ABRA 64] M. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions*, Dover Publications Incorporated, New York, 1964

[AJME 02] J. Ajmera, H. Bourlard, I. Lapidot, I. McCowan, "Unknown-Multiple Speaker Clustering Using HMM", *International Conference on Spoken Language Processing*, Pp. 573-576, 2002

[AJME 03] J. Ajmera, C. Wooters, "A Robust Speaker Clustering Algorithm", *IEEE Automatic Speech Recognition and Understanding Workshop*, 2003

[AFFE 96] V. Affeld, *Spracherkennungsmodul für Windows-basierte Anwendungsprogramme unter spezieller Berücksichtigung des Einsatzes in einem dopplersonographischen Analyselabor*, Diplomarbeit an der FH Giessen-Friedberg, 1996

[BECC 99] C. Becchetti, L.P. Ricotti, *Speech Recognition - Theory and C++ Implementation*, John Wiley & Sons Limited, 1999, ISBN 0471977306

[BILM 98] J.A. Bilmes, *A Gentle Tutorial of the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*, International Computer Science Institute Berkeley CA, April 1998, 02.07.2004, `http://www.icsi.berkeley.edu/ftp/ pub/techreports/1997/tr-97-021.pdf`

[BONA 02] J.F. Bonastre, P. Delacourt, C. Fredouille, T. Merlin, C. Wellekens, "A Speaker Tracking System Based on Speaker Turn Detection for NIST Evaluation", *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2002

[CAMP 97] J.P. Campbell Jr., "Speaker Recognition - A Tutorial", *Proceedings of the IEEE*, Vol. 85, No. 9, Pp. 1437-1462, September 1997

[CAMP 02] W.M. Campbell, K.T. Assaleh, C.C. Broun, "Speaker Recognition With Polynomial Classifiers", *IEEE Transactions on Speech and Audio Processing*, Vol. 10, No. 4, Pp. 205-212, Mai 2002

[CASE 00] M.A. Casey, "Separation of Mixed Audio Sources by Independent Subspace Analysis", *Proceedings of the International Computer Music Conference*, August 2000

[CHEN 98] S.S. Chen, P.S. Gopalakrishnan, "Speaker, Environment and Channel Change Detection and Clustering Via the Bayesian Information Criterion", *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998

[COMB 96] H.P. Combrinck, E.C. Botha, *On the Mel-scaled Cepstrum*, 29.06.2004, `http://citeseer.ist.psu.edu/524151.html`

[CHOU 99] T. Choudhury, B. Clarkson, T. Jebara, A. Pentland, "Multimodal Person Recognition Using Unconstrained Audio and Video", *Proceedings of the Second Conference on Audio- and Video-Based Biometric Person Authentication*, 1999

[CHUN 92] J. Chun, *Re: Cepstrum?*, Beitrag in der Newsgroup `comp.dsp` vom 27 März 1992

[DAVI 80] S. Davis, P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Pp.357-366, 1980

[DAVI 98] N. Davidson *Wave RIFF Format Header and Data Description by Pawn*, 1998, 01.03.2004, `http://www.aros.net/~npawn`

[DFG 02] Deutsche Forschungsgemeinschaft, *Jahresbericht / Annual Report 2002*, 25.06.2003, `http://www.dfg.de/jahresbericht/Wc44629a283c7d.htm`

[DUNN 00] R.B. Dunn, D.A. Reynolds, T.F. Quatieri, "Approaches to Speaker Detection and Tracking in Conversational Speech", *Digital Signal Processing*, Academic Press, No. 10, Pp. 93-112, 2000

[DUNN 03] R.B. Dunn, *Speech Signal Processing and Speech Recognition*, IEEE Signal Processing Society, 29. April 2003

[EL-M 00] K. El-Maleh, M. Klein, G. Petrucci, P. Kabal "Speech/Music Discrimination for Multimedia Applications", *IEEE International Conference on Acoustics, Speech and Signal Processing*, Pp. 2445-2448, 2000

[ELLI 03] D. Ellis, *Audio Signal Recognition for Speech, Music, and Environmental Sounds*, Laboratory for Recognition and Organization of Speech and Audio (LabROSA), Columbia University, New York, 13. November 2003, 30.06.2004, `http://www.ee.columbia.edu/~dpwe/talks/ASA-austin-2003-11.pdf`

[ESTE 00] M. Ester, J. Sander, *Knowledge Discovery in Databases - Techniken und Anwendungen*, Springer-Verlag, Berlin, Heidelberg 2000, ISBN 3540673288

[EZAI 01] H. Ezaidi, J. Rouat, D. O'Shaughnessy, "Combining Pitch and MFCC for Speaker Recognition Systems", *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, Pp. 2825-2828, September 2001

[FENG 02] H. Fenglei, W. Bingxi, "Text-Independent Speaker Verification Using Speaker Clustering and Support Vector Machines", *6th International Conference on Signal Processing*, Vol. 1 , 26-30 August 2002, Pp. 456-459

[FLEX 01] A. Flexer, H. Bauer, C. Lamm, G. Dorffner, "Single Trial Estimation of Evoked Potentials Using Gaussian Mixture Models With Integrated Noise Component", *Lecture Notes in Computer Science*, Vol. 2130, Pp. 609ff, 2001

[GALE 95] M.J.F. Gales, *Model-Based Techniques for Noise Robust Speech Recognition*, Dissertation an der Universität Cambridge (UK), September 1995

[GEOR 00] A. Georghiades, P.N. Belhumeur, D.J. Kriegman, "From Few to Many: Generative Models for Recognition Under Variable Pose and Illumination", *IEEE International Conference on Automatic Face and Gesture Recognition*, Pp. 277-284, 2000

[GONG 95] Y. Gong, "Speech Recognition in Noisy Environments - A Survey" *Speech Communication 16 (1995)*, Pp. 261-291, Elsevier Science, 1995

[GREE 99] M. Greenwood, A. Kinghorn, *SUVing: Automatic Silence/Unvoiced/Voiced Classification of Speech*, 1999, 30.06.2004,
`http://www.dcs.shef.ac.uk/~mark/uni/speech1.pdf`

[HAIG 93] J.A. Haigh, J.S. Mason, "Robust Voice Activity Detection Using Cepstral Features", *IEEE Ten-Con*, Pp. 321-324, 1993

[HE 99a] J. He, L Liu, "Speaker Verification Performance and the Length of Test Sentence", *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, Pp. 305-308, März 1999

[HE 99b] J. He, *C++ KLassenbibliothek "SV_Lib"*, 07.07.2004,
`http://tiger.la.asu.edu/C_lib.htm`

[HE 03] J. He, R. Ewerth, *Persönliche Korrespondenz*, Dezember 2003

[HENZ 97] N. Henze, *Stochastik für Einsteiger*, Vieweg Verlag, 1997, ISBN 3528068949

[HOFF 03] O. Hoffmann, *Biosignal-Verarbeitung - Skript zur Vorlesung*, FH Giessen-Friedberg, 2003

[HOLM 86] J. Holmes, N. Sedgwick, "Noise Compnensation for Speech Recognition Using Probabilistic Models", *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 1986

[JIN 97] H. Jin, F. Kubala, R. Schwartz, "Automatic Speaker Clustering", *Proceedings of the DARPA Speech Recognition Workshop*, Pp. 108-111, Februar 1997

[KILT 03] J. Kilts, *A Threshold Selection Method from Gray-Level Histograms*, Präsentation mit Beispielen, 2003, 09.07.2004, `http://www-users.itlabs.umn.edu/classes/Spring-2003/csci8980/presentations/AThresholdSelectionMethodfromGray-Level.ppt`

[KINN 00] T. Kinnunen, T. Kilpeläinen, P. Fränti, "Comparison of Clustering Algorithms in Speaker Identification"', *Proceedings of the International Conference on Signal Processing and Communications*, Pp. 222-227, September 2000

[KOJR 02] M. Kojro, *Interaktive Software zur Clusteranalyse mit grafischer Ergebnispräsentation*, Diplomarbeit an der FH Giessen-Friedberg, 2002

[KWON 02] S. Kwon, S. Narayanan, "Speaker Change Detection Using A New Weighted Distance Measure", *International Conference on Spoken Language Processing*, 2002

[LAPI 02] I. Lapidot (Voitovetsky), H. Guterman, A. Cohen, "Unsupervised Speaker Recognition Based on Competition Between Self-Organizing Maps" *IEEE Transactions on Neural Networks*, Vol. 13, No. 4, Pp. 877-887, Juli 2002

[LAVA 93] S.M. LaValle, K.J. Moroney, S.A. Hutchinson, "Agglomerative Clustering on Range Data With A Unified Probabilistic Merging Function and Termination Criterion", *IEEE Proceedings of the International Conference on Computer Vision and Pattern Recognition*, Pp. 798-799, Juni 1993

[LI 02] Q. Li, J. Zheng, A. Tsai, Q. Zhou "Robust Endpoint Detection and Energy Normalization for Real-Time Speech and Speaker Recognition", *IEEE Transactions on Speech and Audio Processing*, Vol. 10, No. 3, Pp. 146-157, März 2002

[LI 03] Y. Li, S. Narayanan, C.C.J.Kuo "Content-Based Movie Analysis and Indexing Based on AudioVisual Cues", *IEEE Transactions on Circuits and Systems for Video Technology*, 2003

[LIN 99] Q. Lin, E.E. Jan, C.W. Che, D.S. Yuk, J. Flanagan "Selective Use of the Speech Spectrum and A VQGMM Method for Speaker Identification" *International Conference on Spoken Language Processing*, 1999

[LIU 99] D. Liu, F. Kubala, "Fast Speaker Change Detection for Broadcasting News Transcription and Indexing", *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, Vol. 3, Pp. 1031-1034, 1999

[LIU 03] D. Liu, F. Kubala, "Online Speaker Clustering", *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Pp. 572-575, 2003/2004/2005

[LOGA 98] B.T. Logan, *Adaptive Model-Based Speech Enhancement*, Dissertation an der Universität Cambridge (UK), 1998

[LU 02a] L.Lu, H.J. Zhang, "Real-Time Unsupervised Speaker Change Detection", *Proceedings of the 16th International Conference on Pattern Recognition*, Vol. 2, Pp. 358-361, August 2002

[LU 02b] L. Lu, H.J. Zhang, H. Jiang "Content Analysis for Audio Classification and Segmentation" *IEEE Transactions on Speech and Audio Processing*, Vol. 10, No. 7, Pp. 504-516, Oktober 2002

[LU 02c] L. Lu, H.J. Zhang, "Speaker Change Detection and Tracking in Real-Time News Broadcasting Analysis", *ACM International Conference on Multimedia*, Pp. 602-610, Dezember 2002

[MÄKI 00] V. Makinen, *Front-End Feature Extraction with Mel-Scaled Cepstral Coefficients*, Laboratory of Computational Engineering, Helsinki University of Technology, 14. September 2000

[MARZ 02] M. Marzinzik, B. Kollmeier, "Speech Pause Detection for Noise Spectrum Estimation by Tracking Power Envelope Dynamics", *IEEE Transactions on Speech and Audio Processing*, Vol. 10, No. 2, Pp. 109-118, Februar 2002

[MCLA 99] J.J., McLaughlin, D.A. Reynolds, W. Singer, G.C. O'Leary, "Automatic Speaker Clustering From Multi-Speaker Utterances", *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, May 1999

[MEIG 02] S. Meignier, J.F. Bonastre, I. Magrin-Chagnolleau, "Speaker Utterances Tying Among Speaker Segmented Audio Documents Using Hierarchical Classification - Towards Speaker Indexing of Audio Documents", *Proceedings of the International Conference on Speech and Language Processing*, September 2002

[MICR 91] Microsoft Corporaten, IBM, *Waveform Audio File Format, Multimedia Programming Interface and Data Specification v1.0*, 1991, 08.07.2004, `ftp://ftp.cwi.nl/pub/audio/RIFF-format`

[MULL 02] R. Mullins, *Literature Review*, 03. Juni 2002, 05.05.2004, `http://www.shlrc.mq.edu.au/masters/students/rmullins/SLP804A5.html`

[NADA 88] A. Nádas, D. Nahamoo, M.A. Picheny, "Speech Recognition Using Noise-Adaptive Prototypes", *IEEE Proceedings of the International Conference on Scoustics, Speech and Signal Processing*, Pp. 517-520, 1988

[NADA 89] A. Nádas, D. Nahamoo, M.A. Picheny, "Speech Recognition Using Noise-Adaptive Prototypes", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 37, No. 10, Pp. 1495-1503, Oktober 1989

[NISH 99] M. Nishida, Y. Ariki, "Speaker Indexing for News Articles, Debates and Drama in Broadcast TV Programs", *IEEE*, Pp. 466-471, 1999

[NISH 03a] M. Nishida, T. Kawahara, "Unsupervised Speaker Indexing Using Speaker Model Selection Based on Bayesian Information Criterion", *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, Pp. 172-175, 2003

[NISH 03b] M. Nishida, T. Stadelmann, *Persönliche Korrespondenz*, 05. Dezember 2003

[OTSU 79] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", *IEEE Transaction on Systems, Man and Cybernetics*, Vol. SMC-9, No. 1, Pp. 62-66, 1979

[PETR 03] P.G. Petre, L. Eugen, "On the Influence of Individual Feature Coefficients Over Speaker Recognition", *International Conference on Electrical and Electronical Engineering*, 3.-7. Dezember 2003

[PHAM 02] D.S. Pham, M. Orr, B. Lithgow, R. Mahony, "A Practical Approach to Real-Time Application of Speaker Recognition Using Wavelets and Linear Algebra", *Proceedings of the 6th International Symposium on Digital Signal Processing for Communication Systems*, 2002

[RABI 86] L.R. Rabiner, B.H. Juang, "In Introduction to Hidden Markov Models", *IEEE Acoustics, Speech and Signal Processing Magazine*, Pp. 4-16, Januar 1986

[RABI 89] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedinings of the IEEE*, Vol. 77, No. 2, Pp. 257-286, Februar 1989

[RABI 93] L.R. Rabiner, B.J. Juang, *Fundamentals of Speech Recognition*, Prentice Hall Englewood Cliffs, 1993, ISBN 0130151572

[RAMA 03] B. Ramabhadran, J. Huang, U. Chaudhari, G. Iyengar, H.J. Nock, "Impact of Audio Segmentation and Segment Clustering on Automated Transcription Accuracy of Large Spoken Archives", *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, Pp. 2589-2592, 2003

[REYN 92] D.A. Reynolds, R.C. Rose, "An Integrated Speech-Background Model for Robust Speaker Identification", *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 2, Pp. 185-188, März 1992

[REYN 95] D.A. Reynolds, R.C. Rose, "Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models", *IEEE Transactions on Speech and Audio Processing*, Vol. 3, No. 1, Pp. 72-83, Januar 1995

[REYN 00] D.A. Reynolds, L.P. Heck, *Automatic Speaker Recognition - Recent Progress, Current Applications, and Future Trends* Vortrag im Zuge des AAAS 2000 Meeting on Humans, Computers and Speech Symposium, 19. Februar 2000

[REYN 04] D.A. Reynolds, T. Stadelmann, *Persönliche Korrespondenz*, 15. Mai 2004

[ROSE 91] R.C. Rose, J. Fitzmaurice, E.M. Hofstetter, D.A. Reynolds, "Robust Speaker Identification in Noisy Environments Using Noise Adaptive Speaker Models", *IEEE*, Pp. 401-404, 1991

[ROSE 94] R.C. Rose, E.M. Hofstetter, D.A. Reynolds, "Integrated Models of Signal and Background With Application to Speaker Identification in Noise", *IEEE Transactions on Speech and Audio Processing*, Vol. 2, No. 2, Pp. 245-257, April 1994

[ROSE 00] R.C. Rose, B. Gajic, S. Narayanan, S. Parthasarathy, A. Rosenberg, "Automatic Speech Recognition for Mobile Communication Devices", *IEEE Proceedings of the NORSIG-2000*, 2000

[SANK 96] A. Sankar, C.H. Lee, "A Maximum-Likelihood Approach to Stochastic Matching for Robust Speech Recognition", *IEEE Transanction on Speech and Audio Processing*, Vol. 4, Pp. 190-202, Mai 1996

[SIOH 97] O. Siohan, C.H. Lee, "Iterative Noise and Channel Estimation Under the Stochastic Matching Algorithm Framework", *IEEE Signal Pricessing Letters*, Vol 4, No. 11, Pp. 304-306, November 1997

[SMIT 04] J.O. Smith III, *Decibels*, 01. Januar 2004, 29.06.2004, `http://ccrma-www.stanford.edu/\~jos/mdft/Decibels.html`

[STUT 02] W. Stute, *Stochastik I*, Vorlesungsskript im Wintersemester 2002/2003 an der Justus Liebig Universität Giessen

[SOLO 98] A. Solomonoff, A. Mielke, M. Schmidt, H. Gish, "Clustering Speakers by Their Voices", *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Vol. 2, Pp. 757-760, Mai 1998

[SUN 93] Z.P. Sun, J.S. Mason, "Order Analysis of Combined Features in Speaker Recognition.pdf", *Proceedings of the International Conference on Signal Processing*, Vol. 1, Pp. 704-707, 1993

[TAND 03] I. Tandetnik, *Re: Detecting NaN and infinity*, Beitrag in der Newsgroup `microsoft.public.vc.language` vom 29 April 2003

[TEXA 03] Texas Instruments Incorporated, *Derive 6 Trial Edition*, 2003

[TRIT 99] A. Tritschler, R.A. Gopinath, "Improved Speaker Segmentation and Segments Clustering Using the Bayesian Information Criterion", *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, Pp. 679-682, 1999

[TSEK 98] S. Tsekeridou, I. Pitas, "Speaker Dependent Video Indexing Based on Audio-Visual Interaction", *IEEE*, 1998, Pp. 358-362

[VISW 00] M. Viswanathan, H.S.M. Beigi, S. Dharanipragada, F. Maali, A. Tritschler, "Multimedia Document Retrieval Using Speech and Speaker Recognition", *International Journal on Document Analysis and Recognition*, Springer, Pp. 147-162, 2000

[VOLZ 01] D. Volz, *Vektorquantisierung*, 08. Oktober 2001, 10.07.2004, `http://www.dkfz-heidelberg.de/ibios_old/people/volz/darb/node11.html`

[VOOR 02] E.M. Voorhees, L.P. Buckland, *NIST Special Publication 500-251: The Eleventh Text Retrieval Conference (TREC 2002)*, November 2002

[WALL 04] M. Wallace, S. Kollias, "Robust, Generalized, Quick and Efficient Agglomerative Clustering", *Proceedings of the International Conference on Enterprise Information Systems*, April 2004

[WEIS 99] E.W. Weisstein, "Likelihood", *MathWorld – A Wolfram Web Resource*, 08.08.2004, `http://mathworld.wolfram.com/Likelihood.html`

[WIKI 04a] *Wikipedia - Die freie Enzyklopädie*, 25.06.2004, `http://de.wikipedia.org`

[WIKI 04b] *Wikipedia - The free Encyclopedia*, 30.06.2004, `http://en.wikipedia.org`

[WU 03] T.Y. Wu, L. Lu, K. Chen, H.J. Zhang, "Universal BAckground Models for Real-Time Speaker Change Detection", *Proceedings of the 9th International Conference on Multi-Media Modeling*, Pp. 135-149, Januar 2003

[YOSH 01] K. Yoshida, K. Takagi, K. Ozeki, "A Multi-SNR Subband Model for Speaker Identification Under Noisy Environments" *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, 2001

[YOUN 97] S. Young, J. Odell, D. Ollason, V. Valtchev, P. Woodland, *The HTK Book - Version 2.1*, Cambridge University, revidierte Fassung vom 1. März 1997, 01.07.2004, `http://labrose.ee.columbia.edu/doc/HTKBook21/HTKBook.html`

[YU 02] P. Yu, Z. Cao, "Text-Independent Speaker Identification Under Noisy Conditions Using Speech Enhancement", *Proceedings of the International Symposium on Communication Systems, Networks and Digital Signal Processing*, 2002

[ZITA 04] *www.zitate.de*, 25.06.2004, `http://www.zitate.de/detail-kategorie-6588.htm`

Index