TECHNISCHE UNIVERSITÄT MÜNCHEN
LEHRSTUHL FÜR MENSCH-MASCHINE-KOMMUNIKATION
Prof. Dr.-Ing. habil. G. Rigoll

Studienarbeit

Entwicklung einer Applikation zur automatischen Strukturierung und Analyse großer Musikdatenbanken

Verfasser: Peter Grosche
Haiderweg 11
82057 Icking
2555252

Betreuer: Dipl.-Ing. Ronald Müller

Bearbeitungszeit: 1. November 2005 bis 30. April 2006

Ich versichere, die vorliegende Arbeit selbständig angefertigt und nur die angegebenen Hilfsmittel und Quellen verwendet zu haben.

München, den 30. April 2006.

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Organisation und Strukturierung von Musikdatenbanken. Der Schwerpunkt der entwickelten Methoden liegt auf der inhaltsbasierten Analyse von Musikstücken. Hierzu wurden Merkmale untersucht, die anhand der spektralen und periodischen Eigenschaften die Empfindungen Klangfarbe und Rhythmus modellieren. Die inhaltsbasierte Analyse anhand dieser Merkmale beinhaltet einerseits die Klassifikation der Musik nach verschiedenen Gesichtspunkten, wie Genre Erkennung und Erkennen von Live - Aufnahmen, als auch den Vergleich der Ähnlichkeit verschiedener Lieder über eine Abstandsberechnung.

Im Rahmen der Arbeit wurden Anwendungen zur Vereinfachung der Organisation von Musikdatenbanken entwickelt und in den Foobar2000 Audio Player integriert. Unter anderem sind dies eine Genre Erkennung mit Trainingsmöglichkeit durch den Benutzer, eine inhaltsbasierte Suche nach ähnlichen Musikstücken und eine textbasierte Suche mit Fehlertoleranz auf Grundlage der Levenstein Distance.

Inhaltsverzeichnis

Kapitel 1

Einleitung: Motivation und Ziele

Die Organisation von Musikdatenbanken stellt heutzutage aufgrund der stark wachsenden Verbreitung digitaler Musik und den dadurch immer größer werdenden digitalen Musikdatenbanken eine großes Herausforderung dar. Noch vor wenigen Jahren stand die Plattensammlung in Form von CDs oder anderen *greifbaren* Datenträgern im Regal, heute speichern immer mehr Menschen ihre Musik auf *nicht greifbaren* Datenträgern, wie Festplatten.

Online – Musikportale, wie iTunes, mit mehreren Millionen Titeln, bieten Zugriff auf eine unüberschaubare Vielfalt an Musik, die von überall auf der Welt unkompliziert und schnell verfügbar ist. Doch stellt sich die Schwierigkeit in diesen Daten den Überblick zu behalten, im richtigen Moment das zu finden, wonach man sucht und somit einen Nutzen aus dieser schnellen Verfügbarkeit und riesigen Auswahl ziehen zu können.

Hierzu ist es nötig jedes Musikstück der Datenbank sorgfältig mit den dazugehörigen Informationen zu versehen. Ein Minimum dieser Informationen besteht aus Interpret, Titel und Album eines Stückes, zur genaueren Suche, auch nach bestimmten musikalischen Eigenschaften, ist aber eine sehr viel genauere Beschreibung nötig. Besonders hilfreich wären hier Genre und andere inhaltsbasierte Beschreibungen des Musikstils.

Allerdings hat diese textbasierte Beschreibungsform große Nachteile:

- Die Beschreibung muss manuell erfolgen, ist sehr zeitaufwändig und fehleranfällig.
- Es ist schwer einen subjektiven Eindruck von Musik in Worte zu fassen.
- Suchkriterien zu finden ist schwierig.

Besonders der nötige Zeitaufwand und die Fehleranfälligkeit des Verfahrens führen zu unbefriedigenden Suchergebnissen.

Eine Möglichkeit, auch unbekannte Titel in den riesigen Datenbanken zu finden, bieten die Empfehlungsfunktionen einiger Online – Musikportale. Nur beruhen diese Empfehlungen alle auf dem subjektiven Empfinden anderer Nutzer oder einer Redaktion und entsprechen nicht unbedingt den eigenen Präferenzen.

Aus diesen Unzulänglichkeiten der Musikarchivierung heraus entstand einerseits die Idee, die Suche nach bestimmten Musikstücken über die Informationen in Textform weniger fehleranfällig zu gestalten und außerdem einen Weg zu finden – unabhängig von der Meta Information – Datenbanken anhand der musikalischen Eigenschaften der Lieder zu organisieren.

Kapitel 2

Überblick über Music Information Retrieval

Die inhaltsbasierte Analyse von Musikstücken zur Organisation und Indexierung beruht auf der Wahrnehmung der Musik durch den Menschen und lässt sich unter dem Schlagwort *Music Information Retrieval (MIR)* zusammenfassen.

2.1 Wahrnehmung von Musik

Musik ist ein multidimensionales Medium mit sehr vielen Facetten und Ausprägungen. Obwohl der Mensch in der Lage ist verschiedene Arten von Musik wiederzuerkennen, zu unterscheiden und anhand bestimmter Merkmale zu beschreiben, ist es zum gegenwärtigen Stand der Forschung schwierig diesen Prozess, der nicht nur eine Wahrnehmung sondern auch eine kognitive Verarbeitung einschließt, technisch nachzubilden.

Die Wahrnehmung von Musik durch den Menschen wird beeinflusst von unterschiedlichen psychoakustischen Empfindungen: Der Tonhöhe, dem Rhythmus, der Lautheit und der Klangfarbe [Nam01].

Um die Empfindung, die Musik beim Zuhörer auslöst, nachzubilden müssen also Merkmale gefunden werden, die diesen psychoakustischen Empfindungen entsprechen oder sie beschreiben.

2.1.1 Tonhöhe

Die Tonhöhe ist die Beschreibung einer Wahrnehmung, die sich durch das Einstellen eines vergleichenden Sinus-Tones bestimmen lässt [ZF90]. Auch für komplexe

Spektren wird eine Tonhöhe empfunden. Zeitliche Änderungen der Tonhöhe stellen somit eine Melodie dar.

2.1.2 Klangfarbe

Die Klangfarbe eines Musikstückes ist eine Wahrnehmung, die unabhängig von der Lautheit, der Tonhöhe und der Dauer des Signals ist [ZF90] und von der Instrumentalisierung eines Musikstückes beeinflusst wird. Musikinstrumente weisen unterschiedlichste spektrale Verteilungen und zeitliche Hüllkurven auf. Somit ergeben sich für unterschiedliche Instrumentalisierungen unterschiedliche spektrale Verteilungen, die sich nutzen lassen um Musikstücke zu unterscheiden.

2.1.3 Rhythmus

Der Rhythmus eines Liedes wird durch seine Struktur, Komplexität und Ausgeprägtheit, sowie das Tempo beschrieben und stellt eine sehr wichtige Komponente bei der Wahrnehmung von Musik dar. Außerdem bietet er ein gutes Unterscheidungsmerkmal für unterschiedliche Musikstile. Während ein klassisches Musikstück eine schwach ausgeprägte Rhythmusstruktur besitzt, ist ein komplexer und stark ausgeprägter Rhythmus eine wichtige und dominante Eigenschaft der Pop – Musik.

2.2 Stand der Technik

In den letzten Jahren hat die Forschung auf dem Gebiet des MIR immer mehr zugenommen und große Fortschritte gemacht. Hier sollen die neuesten Entwicklungen betrachtet, ihre Gemeinsamkeiten und Unterschiede herausgestellt und die Ergebnisse verglichen werden, um daraus eine mögliche Vorauswahl über die verwendeten Methoden zu treffen.

Bisher entwickelte Verfahren sind die Klassifikation von Musik nach verschiedenen Gesichtspunkten, das Finden zueinander ähnlicher Musikstücke und die automatische Segmentierung von Musikstücken und damit die Möglichkeit eine kurze Zusammenfassung eines Liedes zu erstellen.

2.2.1 Automatische Genre - Erkennung

Die automatische Erkennung von Musikstilen ist der Bereich des MIR, der die größte Aufmerksamkeit erhält. Ein Problem, das bei dieser Aufgabe recht häufig

auftritt ist, dass die Einteilung der Musik in verschiedene Genres oft nicht eindeutig anhand der musikalischen Eigenschaften erfolgen kann, sondern von geographischen oder epochalen Gesichtspunkten und dem persönlichen Geschmack des Künstlers abhängig ist. Außerdem gibt es eine unüberschaubar große Anzahl an Genre und die Einteilung wird von jedem Menschen subjektiv und unterschiedlich vorgenommen.

Aus diesem Grund und dem frühen Stadium des Forschungsgebietes können alle Systeme keine fertigen Genre - Klassifikatoren sein, sondern nur als *Proof of Concept* dienen.

Der grundlegende Aufbau aller entwickelten Systeme orientiert sich an einem dreistufigen Aufbau eines Mustererkennungssystems:

1. Extraktion und Selektion von Merkmalen
2. Training eines Klassifikators
3. Klassifikation

Zuerst werden bestimmte Merkmale berechnet, die die Wahrnehmung der Musik beschreiben, ein Klassifikator mit Beispielen aus jedem Genre trainiert und mit diesem Testdaten klassifiziert.

Die vorhandenen Systeme unterscheiden sich in der Anzahl und der Art der verwendeten Merkmale und Genres, der Größe der verwendeten Datenbank und des verwendeten Klassifikators. Einen sehr umfassenden Überblick über die verschiedenen Verfahren bietet [AP03], neuere Ergebnisse sind in Tabelle 2.1 zusammengefasst.

Bis auf [Pye99] verwenden alle Verfahren unkomprimierte PCM Dateien, auf denen eine DFT zur Transformation in den Frequenzbereich angewendet wird. [Pye99] verwendet nach dem MPEG1-Layer III Verfahren kodierte Dateien, deren Dekodierungsvorgang aufgespaltet und mit der so gewonnenen Frequenzinformation Cepstral - Koeffizienten berechnet werden. Wie der Tabelle 2.1 zu entnehmen sind die erreichten Erkennungsraten im Vergleich zu konventionell berechneten Cepstral - Koeffizienten kaum schlechter, die Rechenzeit des Verfahrens jedoch um den Faktor 5 kürzer.

Obwohl für die verwendeten Merkmale unterschiedlichste Bezeichnungen verwendet werden lassen sich alle in der Tabelle 2.1 angegebenen Merkmale anhand ihrer Wahrnehmungsbeschreibung in die 3 in Kapitel 2.1 beschriebenen Klassen einteilen.

Veröff.	Merkmale	Klass.	n.G.	DB	ER in %
[XMS+03]	Beat Spectrum	SVM	4	60 / 40	93.14
	LPC-Derived-Cepstrum	NN			79.43
	Zero Crossings	GMM			87.69
	MFCC	HMM			88.06
	Spectrum Power				
[Pye99]	MFCC	GMM	6	175 / 175	63 MFCC
	mp3CEP	TreeQ			59 mp3CEP
[TEC01]	Centroid	GMM	6	50 / 750	62
	Rolloff				
	Flux				
	Zero Crossings				
	Low Energy				
	Rhythm				
	MFCC				
[LOL03]	Daubechies Wavelet	GMM	10	1000	64
	Coefficient Histograms	LDA			71
		kNN			62
		SVM			79
[JLZ02]	Spectral Contrast	GMM	5	1500	82

Tabelle 2.1: Übersicht über die verwendeten Merkmale, Klassifikatoren (Klass.), Anzahl der Genre (n.G.), Größe der Datenbank (Trainings- / Testdatensatz) und die angegebenen Erkennungsraten

Klangfarbe Die Merkmale zur Beschreibung der Klangfarbe beschreiben alle die spektrale Verteilung der Energie. Im einfachsten Fall, wie in [DNS01] verwendet, sind dies die rohen Fourier Koeffizienten aus der Transformation des Signals in den Frequenzbereich.

In [TC02] wird aus diesen Koeffizienten eine Beschreibung des Spektrums mittels des Schwerpunktes, dem Abfall am oberen Rand des Spektrums und der zeitlichen Änderung des spektralen Verteilung der Energie abgeleitet. Somit lässt sich die Anzahl der Merkmale deutlich reduzieren, ohne dabei sehr viel relevante Information über das Spektrum zu verlieren.

Eine weitere kompakte Beschreibung der spektralen Eigenschaften sind die *Cepstral - Koeffizienten* (MFCC oder auch mp3CEP). In der Sprachverarbeitung weit verbreitet werden sie auch bei der Verarbeitung von Musik sehr häufig eingesetzt: In [LR04], [XMS+03], oder in oben beschriebener Form auch in [Pye99].

Rhythmus Zu den rhythmusbeschreibenden Merkmalen gehört das in [XMS+03] eingesetzte *Beat Spectrum* und auch das in [TEC01] verwendete *Beat Histogramm*. Trotz der unterschiedlichen Namen handelt es sich um ähnliche bis gleiche Verfahren. Alle verwenden eine Autokorrelationsfunktion (AKF) um die Korrelation des Musiksignals zu unterschiedlichen Zeitpunkten zu berechnen und daraus den Rhythmus ableiten zu können. Einzig in der Vorverarbeitung gibt es Unterschiede.

Tonhöhe Eine einfache und häufig verwendete Methode die Tonhöhe zu repräsentieren ist die *Zero Crossing Rate* zu bestimmen [AML04]. Diese beschreibt die Anzahl der Nulldurchgänge im Zeitbereich und kann somit ein Maß für die Grundfrequenz eines Signals sein, da bei einem Sinuston zweimal pro Periode ein Nulldurchgang auftritt. Bei rauschähnlichen Signalen wie Musik treten aber wesentlich mehr Nulldurchgänge auf, so dass dieses Verfahren weniger die Grundfrequenz als vielmehr die *Rauschigkeit* eines Signals beschreibt. Bisher wurden wirkliche tonhöhen-basierte Merkmale nur in [TC02] verwendet. Das entwickelte Verfahren berechnet aus den Maxima einer AKF, die auf die zeitliche Amplitudenenvelope angewendet wird, ein Tonhöhen-Histogramm, aus dem sich die vorherrschende Tonhöhe, die Struktur der Harmonischen und die Ausgeprägtheit der Tonhöhe berechnen lässt. Die erreichten Erkennungsraten der Genre Klassifikation allein auf Grundlage der Tonhöhe lagen allerdings nur bei 23%.

Die Tatsache, dass alle sehr verschiedene Genres verwenden, lassen die Ergebnisse nur schlecht vergleichen, es lassen sich allerdings einige grundlegende Schlüsse daraus ziehen. Höhere Erkennungsraten ermöglicht die Verwendung verschiedenartiger Merkmale, die verschiedene Wahrnehmungen beschreiben. So werden in

[Pye99] durch die ausschließlichen Verwendung von Cepstral - Koeffizienten eine erheblich niedrigere Erkennungsrate erreicht als in [XMS+03], wo zusätzlich Merkmale zur Beschreibung weiterer Eigenschaften verwendet wurden.

Außerdem lässt sich aus dem direkten Vergleich einiger Erkennungsraten ableiten, dass die Verwendung von *Support Vector Machines (SVM)* unter den verwendeten Klassifikatoren die beste Wahl darstellt.

2.2.2 Inhaltsbasierte Ähnlichkeitssuche

Die inhaltsbasierte Suche nach ähnlichen Musikstücken gleicht der Genre - Klassifikation insoweit, dass auch hier Merkmale berechnet und miteinander verglichen werden. Allerdings handelt es sich hierbei nicht um einen vorherigen Trainingsprozess mit anschliessender Klassifikation sondern um den Vergleich eines Musikstückes mit einer großen Anzahl anderer und das Auffinden des oder der ähnlichsten Musikstücke aus dieser Menge.

Für diese Aufgabe können die selben Merkmale verwendet werden, da auch hier eine Wahrnehmung beschrieben werden muss.

Zur Ähnlichkeitsbestimmung wird oftmals eine *Nearest Neighbor* Suche verwendet [AHH+03]. Auch die *Earth Movers Distance* [LS01] und der *Monte Carlo Ansatz* [AF04] zur Bestimmung der Wahrscheinlichkeit, mit der die für ein Lied berechneten Merkmale in dem *Gaussian Mixture Model* eines anderen Liedes vorhanden sind.

Die Ergebnisse der verschiedenen Verfahren lassen sich noch schlechter vergleichen als die der Genre - Klassifikation, da die *Ähnlichkeit* zweier Musikstücke noch stärker vom subjektiven Empfinden des Einzelnen abhängig ist.

2.2.3 Segmentierung und Zusammenfassung von Musik

Die Zusammenfassung von Musikstücken dient dem *Audio Thumbnailing*, angelehnt an die Miniaturvorschau von Bildern. Es wird versucht ein komplettes Musikstück auf die wesentlichen Bereiche – im Normalfall ist dies der Refrain – zu komprimieren und so dem Hörer einen kurzen Abschnitt des Stückes vorspielen zu können, der das komplette Lied am besten beschreibt. Bisher entwickelte Systeme verwenden ebenfalls die in Kapitel 2.2.1 vorgestellten Merkmale [CF02]. Durch eine Abstandsberechnung jeder Kombination dieser Merkmalsvektoren läßt sich derjenige Abschnitt finden, der das Lied am besten repräsentiert, also zum gesamten Lied die größte Ähnlichkeit aufweist.

Kapitel 3

Merkmalsgewinnung

Die digitale Musik wird heute zum größten Teil komprimiert gespeichert. Der Standard unter den datenreduzierten Audioformaten ist, trotz neuerer Alternativen mit besserer Qualität, immer noch der vom Fraunhofer Institut für Integrierte Schaltungen 1991 entwickelte ISO - MPEG1 Layer III - Audio Standard [tec01]. Besser bekannt unter der Dateiendung mp3. Gegenüber den konkurrierenden Formaten, wie z.B. AAC oder ogg, bietet das mp3 - Format den Vorteil einer großen Bekanntheit und Verbreitung beim Nutzer und wird deshalb auch in Zukunft noch *der* Standard unter den Kompressoren bleiben.

Aufgrund der großen Verbreitung und dem Aufbau dieses Formates wurde es als Grundlage für diese Arbeit gewählt.

3.1 Funktionsweise der mp3 - Kompression

MPEG1 - Layer III ist ein so genannter *perceptual coder*, d.h. er nutzt die psychoakustischen Eigenschaften des Ohres um die Datenrate zu reduzieren, ohne das diese Reduktion vom Menschen wahrgenommen werden kann.

3.1.1 Der Enkoder

Der Enkoder des MPEG1 - Layer III Audioformates enthält die Intelligenz, die zur unhörbaren Datenreduktion nötig ist [Hac00]. Der Enkoder ist nicht von der ISO - Norm betroffen, er muss allerdings einen ISO - konformen Bitstrom erzeugen, sodass er von einem ISO - konformen Dekoder abgespielt werden kann.

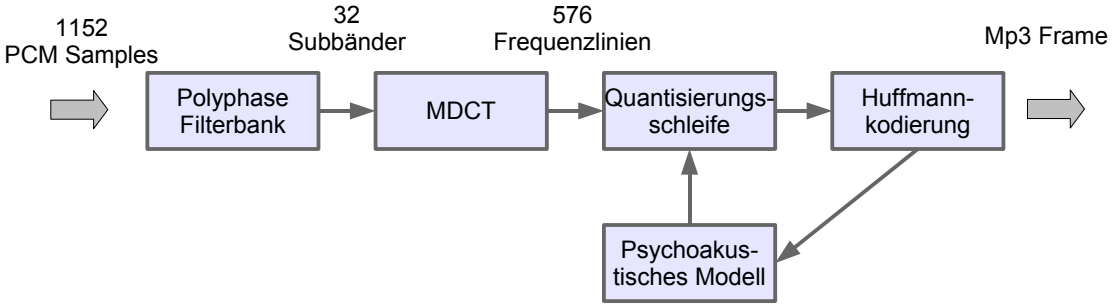


Abbildung 3.1: MPEG1 – Layer III Enkoder

Polyphase Filterbank

Das PCM kodierte und in Frames zu jeweils 1152 Samples eingeteilte Audiosignal [Pan95] wird zunächst mittels einer Polyphase Filterbank vom Zeit- in den Frequenzbereich transformiert. Diese besteht aus 32 parallelen Bandpassfiltern, die das gesamte Spektrum des Audiosignals von $0 - \frac{f_s}{2}$ in 32 gleichbreite Subänder einteilen. Hierzu werden immer 32 Samples im Zeitbereich in 32 Samples im Frequenzbereich umgewandelt [Kap02]. Somit ergeben sich für jeden Frame aus den ursprünglich 1152 Samples im Zeitbereich 32 Frequenzwerte zu 36 verschiedenen Zeitpunkten.

Modifizierte Cosinus Transformation

Zusätzlich zur Filterbank wird eine Modifizierte Cosinus Transformation (MDCT) eingesetzt, bei der es sich um eine eindimensionale Cosinustransformation handelt, deren Transformationsfenster sich jeweils zu 50% überlappen.

Die MDCT wird auf die 32 Frequenzbänder mit jeweils 36 Zeitsamples pro Frame der Polyphase-Filterbank angewendet, um die Frequenzauflösung auf Kosten der Zeitauflösung noch weiter zu erhöhen. Da sich die Transformationsfenster jeweils zur Hälfte überlappen, ergeben sich 18 Frequenzwerte pro Subband. Die 1152 Zeitsamples je Frame werden also in $18 \cdot 32$ Frequenzwerte pro Granulat transformiert, wobei zwei Granulate einen Frame bilden.

Psychoakustisches Modell

Die eigentliche Komprimierung der Daten erfolgt bei der Quantisierung des Signals unter der Berücksichtigung eines psychoakustischen Modells, dass die akustisch irrelevanten Teile entfernt. Hierzu wird das Unvermögen des menschlichen Hörsystems ausgenutzt das Quantisierungsrauschen wahrzunehmen, dass unter der

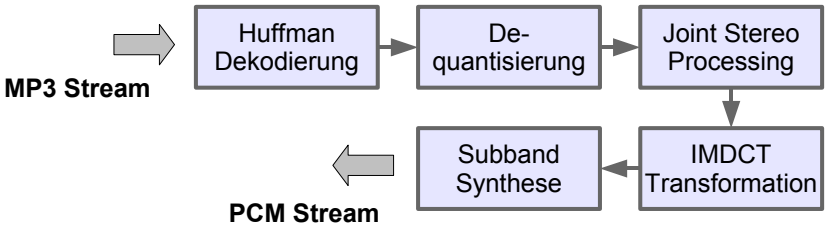


Abbildung 3.2: MPEG1 – Layer III Dekoder

Mithörschwelle liegt. Signale werden von einem lauteren Signal in der spektralen oder zeitlichen Nachbarschaft verdeckt, wenn sie unter seine Vor-/Nachverdeckung oder Mithörschwelle fallen. Da der Abstand des Quantisierungsrauschens vom eigentlichen Ton abhängig ist von der Anzahl der zur Quantisierung verwendeten Bits, können diese maskierenden Eigenschaften ausgenutzt werden um das Signal unhörbar in seiner Datenrate zu reduzieren.

Quantisierung

Auf Grundlage der im Psychoakustischen Modell gespeicherten Informationen werden für jedes Sample nur die Anzahl an Bits zur Quantisierung genutzt, die nötig sind, um das Quantisierungrauschen unter die Mithörschwelle zu drücken und somit unhörbar zu machen.

Huffman Kodierung

Die Huffman Kodierung entfernt redundante Information verlustlos aus dem kodierten Bitstrom. Hierzu werden häufigeren Werten kleine Codewörter und seteneren Werten größere Codewörter zugeordnet.

3.1.2 Der Dekoder

Der Dekoder ist quasi-invers zum Enkoder [tec01]. Alle vom Enkoder vorgenommenen Verarbeitungsschritte muss dieser rückgängig machen, um aus dem Bitstrom wieder normale PCM Samples im Zeitbereich zu erzeugen, die wiedergegeben werden können.

Huffman Dekodierung

In diesem Arbeitsschritt wird die vom Enkoder zur weiteren Datenreduktion vorgenommene Huffman Kodierung rückgängig gemacht.

Dequantisierung

Die vorgenommene Quantisierung wird rückgängig gemacht und jedem Sample wieder die die Bit - Tiefe des Ausgangssignales zugewiesen.

Joint Stereo Processing

Der MPEG1 – Layer III Standard unterstützt drei verschiedene Möglichkeiten der Stereo Kodierung.

Den *normalen* Stereo Modus und zwei weitere so genannte Joint-Stereo-Kodierungen, die die Eigenschaft der Stereo Signale zur Datenreduktion ausnutzen, dass sich die beiden Kanäle nur wenig von einander unterscheiden.

Stereo Der normale Stereo Modus speichert zwei Mono-Spuren unabhängig von einander und kann dadurch keine weitere Datenreduktion vornehmen.

MS-Stereo Bei der *MS-Stereo* Kodierung verwendet werden anstatt der einzelnen Kanäle L und R die Summe $M = R+L$ und die Differenz $S = R-L$ gebildet. Da sich der rechte und linke Kanal nur wenig unterscheiden enthält der Differenzkanal nur wenig Information und kann deshalb stärker komprimiert werden.

Intenstity-Stereo Die *Intensity Stereo* Methode ist im Gegensatz zur MS - Kodierung verlustbehaftet und kann deshalb Signale auf Kosten der Qualität stärker komprimieren. Sie nutzt die Eigenschaft des Gehörs aus, sehr hoch- und sehr tieffrequente Töne nicht orten zu können und fasst deshalb beide Kanäle in diesen Bereichen zusammen.

Inverse Modifizierte Cosinus Transformation

Die IMDCT transformiert die Frequenzwerte zurück in den Zeitbereich, also in 36 Zeitsamples für jedes der 32 Subbänder.

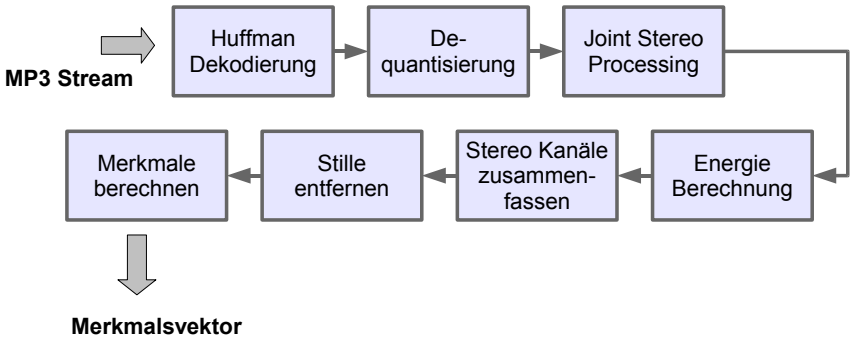


Abbildung 3.3: Gewinnung der Frequenzinformation aus dem mp3 – Bitstrom

Subband-Synthese

Die Subband Sysnthese erzeugt aus den $32 \cdot 36$ Zeitsamples wieder 1152 PCM Samples.

3.1.3 Gewinnung der Frequenzinformation

Der *normale* Weg, die Frequenzinformation einer Audiodatei zu gewinnen ist es auf das Signal eine DFT anzuwenden und so in den Frequenzbereich zu transformieren.

Da bei der Kodierung einer Datei in das mp3 - Format aber schon eine Frequenztransformation vorgenommen wird – siehe Kapitel 3.1.1 – und die Frequenzinformation in Form der Subband Daten gespeichert ist, besteht die Möglichkeit diese zu verwenden. Der Vorteil dieser Methode ist eine sehr viel schnellere Berechnung, da der rechenintensivste Abschnitt der mp3-Dekodierung, die IMDCT und die darauffolgende Filterbank Synthese, weggelassen werden kann. Siehe hierzu Kapitel 3.1.2. Außerdem ist keine erneute Transformation in den Frequenzbereich nötig. Es können somit zwei Transformationen eingespart werden.

Nachteilig an diesem Verfahren ist allerdings, dass keine freie Wahl der Fensterlänge und damit der Frequenz- bzw. Zeitauflösung möglich ist, sondern man an den Aufbau des mp3-Bitstroms gebunden ist, der alle 13 ms ein Granulat mit 576 Frequenzwerten enthält. Außerdem ist dieses Verfahren auf die Verwendung von mp3 - Dateien beschränkt. Alternative Audiokompressionsverfahren lassen sich mit diesem Verfahren somit nicht ohne vorherige Anpassungen verwenden.

Aufspalten des Dekodierungsprozesses

Um die Subband Information aus einer mp3 - Datei zu gewinnen muss der Dekodierungsprozess an der richtigen Stelle unterbrochen werden. Nicht vorgenommen werden darf die Wandlung zurück in den Zeitbereich, die von der IMDCT und der Subband-Synthese vorgenommen wird.

Energieberechnung

Für jedes Granulat werden 576 Frequenzwerte in 32 Subbändern erhalten. Jeweils 2 Granulate entsprechen einem Frame und entstehen aufgrund der 50% Fensterüberlappung der MDCT. Diese werden zusammengefasst indem der quadratische Mittelwert über alle 18 Werte in jedem Subband gebildet wird.

$$RMS_{gr}(sb) = \sqrt{\frac{1}{18}\sum_{ss=1}^{18} S_{ss}^2} \tag{3.1}$$

Somit erscheint alle 13 ms ein Kurzzeitspektrum mit 32 Energiewerten. Beide aufeinanderfolgende Granulate eines Frames werden mit dem arithmetischen Mittel zu einem Kurzzeitspektrum pro Frame zusammengefasst.

$$E_{fr}(sb) = \frac{1}{2}\sum_{gr=1}^{2} RMS_{gr}(sb) \tag{3.2}$$

Es erscheint alle 26 ms ein 32 dimensionaler Subbandvektor der die Frequenzinformation des jeweiligen Frames enthält. Es wird also das Kurzzeitspektrum für jeden Frame berechnet.

Zusammenfassen der Stereokanäle

Da der Dekoder zum Zeitpunkt der Frequenzinformationgewinnung die Stereokanäle schon dekodiert hat, liegen zwei von einander unabhängige Monokanäle vor. Diese beiden Kanäle werden mittels des arithmetischen Mittels zu einem zusammengefasst.

$$E_{fr}(sb) = \frac{E_{fr,L}(sb) + E_{fr,R}(sb)}{2} \tag{3.3}$$

Somit wird ein *pseudo* Mono-Kanal erzeugt, der die Information beider Stereokanäle enthält und für die weitere Berechnung verwendet wird.

Somit steht die Frequenzinformation in Form von 32 - dimensionalen Subbandvektoren, die alle 26 ms berechnet werden, zu Verfügung. Diese Kurzzeitspektren bieten eine Frequenzauflösung von $689Hz$ und eine Zeitauflösung von 26 ms. Im Vergleich ließe sich durch die Berechnung der Kurzzeitspektren mit einer DFT ein besseres Verhältnis von Frequenz- und Zeitauflösung erreichen. Eine Fensterlänge von 26 ms entspricht einer sehr viel höheren Frequenzauflösung von 38 Hz.

Dieser Nachteil des verwendeten Verfahrens gegenüber einer diskreten Frequenztransformation geht einher mit dem Vorteil einer erheblich schnelleren Ausführung, da zwei gegensätzliche Transformationen, eine in den Zeitbereich und eine zurück in den Frequenzbereich, entfallen.

Entfernen von Stille

Aus diesem, sich so für jede Datei ergebendem Vektor mit 32 Energieeinträgen pro Frame werden all diejenigen Frames entfernt, deren Energie gleich oder nahe Null ist, um nur noch relevante Informationen über das Audiosignal in dem Vektor zu erhalten.

3.2 Merkmale

Nach der Gewinnung der Frequenzinformation wird diese zu Berechnung charakteristischer Merkmale verwendet. Da mit dem verwendeten Ansatz das Audiosignal nicht im Zeitbereich zu Verfügung steht, können nur Merkmale aus dem Spektrum des Audiosignals berechnet, also nur eine Analyse im Frequenzbereich vorgenommen werden.

3.2.1 Frame - Energie

Die Energie pro Frame beschreibt die Amplitudenenvelope des Musiksignals, also den zeitlichen Verlauf der Energie über das gesamte Spektrum und ist somit ein Maß für die Lautheit [TC00] und die Schwankung des Signals. Zur Berechnung wird die Summe über alle Energiewerte des Subandvektors des jeweiligen Frames gebildet.

$$\overline{E}_{fr} = \frac{1}{32} \sum_{sb=1}^{32} E_{fr}(sb) \tag{3.4}$$

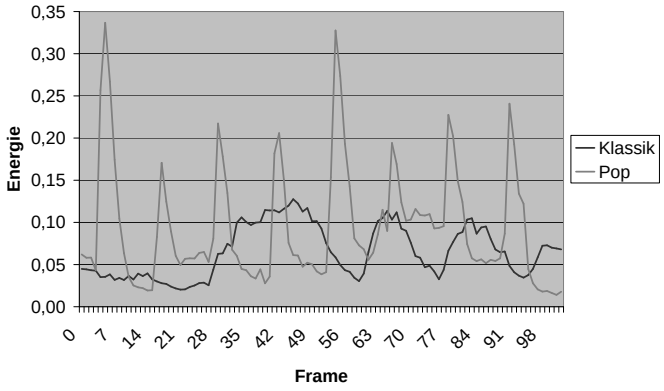


Abbildung 3.4: Vergleich des Verlaufes der Frame - Energie für Pop- und klassische Musik

3.2.2 Kurzzeitspektrum

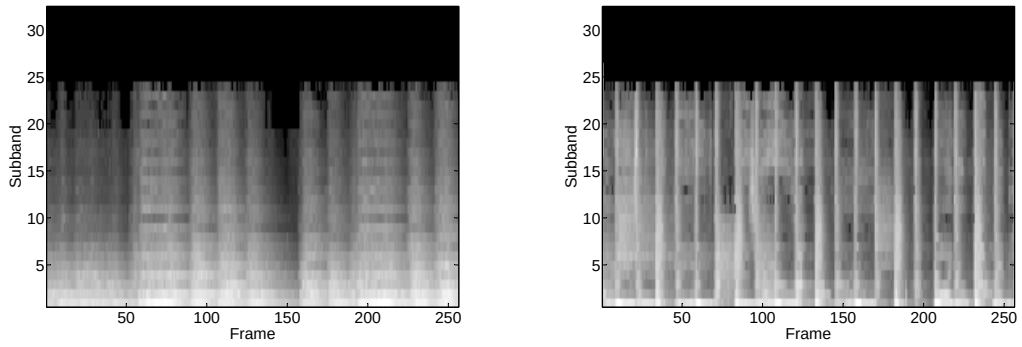
Die spektrale Verteilung der Energie im Frame wird beschrieben anhand der Verteilung der Energie auf die Subbänder. Hierzu wird die Energie der einzelnen Subbänder auf die im Frame vorhandene Gesamtenergie normiert, um eine spektrale Beschreibung zu erhalten, die unabhängig ist vom Pegel.

$$E_{fr}(sb)_n = \frac{E_{fr}(sb)}{\overline{E}_{fr}} \tag{3.5}$$

3.2.3 Spectral Centroid

Der *Spectral Centroid* [LR04] beschreibt den Schwerpunkt der Energieverteilung im Subbandvektor für jeden Frame und ist somit ein Maß für die *Helligkeit* des Signals. Je mehr hochfrequente Anteile das Spektrum enthält, umso höher liegt der Schwerpunkt. Der Centroid ist also eine sehr kompakte Beschreibungsmöglichkeit der spektralen Energieverteilung.

$$Spectral\,Centroid = \frac{\sum_{sb=1}^{32} sb \cdot E_{fr}(sb)}{\sum_{sb=1}^{32} E_{fr}(sb)} \tag{3.6}$$



(a) Klassik (b) Pop

Abbildung 3.5: Kurzzeitspektren eines klassischen und eines Pop - Musikstücks

3.2.4 Spectral Flux

Der *Spectral Flux* [AP03] ist die Differenz der spektralen Energieverteilung der Frame - Energie zweier nacheinanderfolgender Frames und beschreibt somit die zeitliche Änderung der Energieverteilung.

$$Spectral\ Flux = \frac{\sum_{sb=1}^{32} \left(E_{fr}(sb) - E_{fr-1}(sb)\right)^2}{\max\ E_{fr}(sb)^2 \cdot 32} \tag{3.7}$$

Von der Differenz wird der Betrag gebildet und auf die maximale in einem Subband vorhandene Energie normalisiert.

3.2.5 Spectral Rolloff

Der *Spectral Rolloff Faktor* [TEC01] ist ein Percentil der spektralen Energieverteilung. Dieser Faktor beschreibt also welcher Anteil der Gesamtenergie pro Frame unterhalb von welcher Frequenz liegt. Hier wurde ein Rolloff Faktor von $0,95$ gewählt, sodass der Faktor beschreibt, in welchen Subbändern – beginnend beim Tiefsten – sich 95% der Energie befindet. Der Faktor ist somit ein Maß für die spektrale Verteilung der Energie und dem Abfall des Spektrums zu hohen Frequenzen hin.

$$\sum_{sb=1}^{rolloff} E_{fr}(sb) \leq 0.95 \cdot \sum_{sb=1}^{32} \overline{E}_{fr} \tag{3.8}$$

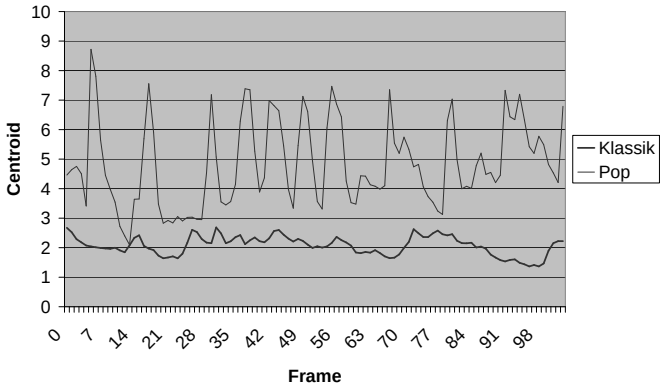


Abbildung 3.6: Vergleich des Spectral Centroid eines klassischen und eines Pop – Musikstücks

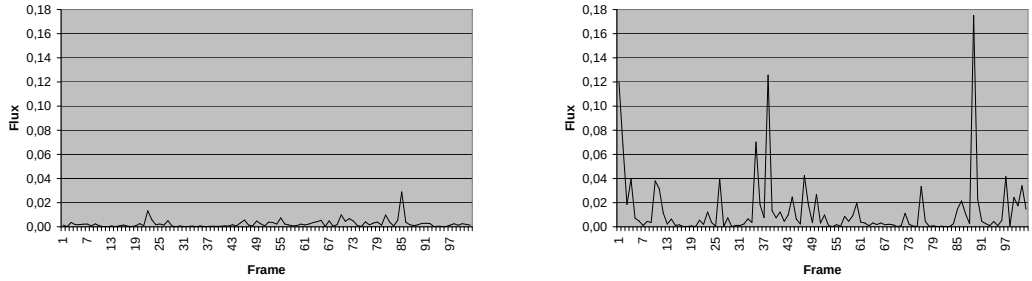
3.2.6 Low Energy

Low Energy [TC00] beschreibt welcher Anteil an Frames in einem Fenster von $N_w = 40$ Frames geringere Energie $\overline{E}_{fr}$ enthalten, als der Mittelwert der Energie $\overline{E}_w$ in diesem Fenster. Das Merkmal ist somit ein Maß für die zeitlichen Amplitudenschwankung eines Signales und besonders geeignet um Musik- und Sprachsignale voneinander zu unterscheiden, da Sprache großen zeitlichen Schwankungen unterliegt.

$$Low\ Energy = \frac{N(\overline{E}_{fr} \leq \overline{E}_w)}{N_w} \tag{3.9}$$

3.2.7 Rhythmus

Periodizität in einem Musiksignal ist charakteristisch für Rhythmus. Schlaginstrumente bilden ein Muster das sich periodisch über die Zeit hinweg wiederholt. Somit zeigt die spektrale Energieverteilung Wiederholungen mit einer zeitlichen Verzögerung, die mit dem Rhythmus des Liedes korreliert. Durch die Berechnung der AKF [TEC01] mehrerer nacheinanderfolgender Frames können die rhythmischen Eigenschaften des jeweiligen Musikstückes beschrieben und die Präsenz von rhythmusgebenden Instrumenten erkannt werden.



(a) Klassik (b) Pop

Abbildung 3.7: Vergleich des Spectral Flux für Pop und klassische Musik

Die AKF beschreibt die Ähnlichkeit zweier Signale zueinander [Rig02b]. Mit dieser Eigenschaft lassen sich periodisch wiederkehrende Frames innerhalb eines Stückes erkennen und auf ihre Periode und Ausgeprägtheit hin untersuchen.

Für jeden Subbandvektor eines 40 ms langen Analysefensters wird die AKF mit jedem der darauffolgenden 80 Subbandvektoren gebildet. Dieser Wert entspricht einer Verzögerung von 2 Sekunden und soll sicherstellen, dass auch bei langsamen Rhythmen noch eine volle Periode in dieses Fenster fällt.

$$AKF(\tau, sb) = \frac{1}{40} \sum_{fr=1}^{40} E_{fr}(sb) \cdot E_{fr+\tau}(sb) \tag{3.10}$$

$$1 \leq \tau \leq 80$$

Somit ergibt sich für jedes Fenster ein Wert der AKF für jedes τ. Wobei τ die zeitliche Verschiebung ist zu der die AKF berechnet wird und $E_{fr}(sb)$ der Subbandvektor an der Stelle 0 und $E_{fr+\tau}(sb)$ der Vektor an der Stelle τ.

Zur reinen Rhythmus und nicht Melodieerkennung wird ein Gewichtungsfaktor $G(sb)$ eingeführt, der hohe und tiefe Frequenzen stärker gewichtet, als das mittlere Spektrum. Rhythmusinstrumente sind sehr ausgeprägt in den hohen (z.B. Hihat, Becken) und niedrigen (z.B. Basedrum) Bereichen des Spektrums. Wohingegen melodietragende Instrumente ihren Schwerpunkt in den mittleren Lagen des Spektrums haben.

Nach der Gewichtung wird die Summe über alle Werte der AKF gebildet.

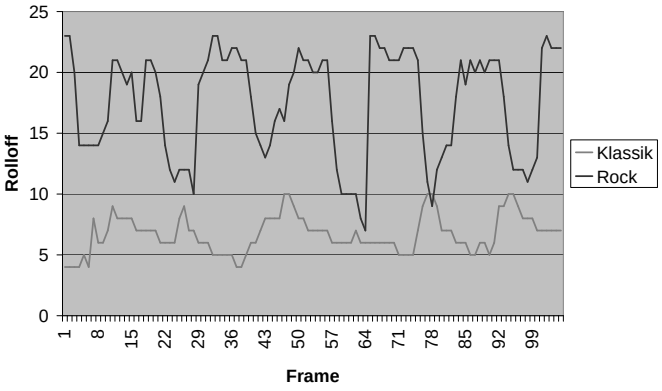


Abbildung 3.8: Vergleich des Rolloff Faktors für Rock und klassische Musik

$$SUM(\tau) = \sum_{sb=1}^{32} AKF(\tau, sb) \cdot G(sb) \tag{3.11}$$

Somit ergibt sich für jeden Werte der Verzögerung τ ein Wert für die Korrelation. Aus diesen 80 Werten wird das Maximum gesucht um die größte Korrelation zu erhalten.

$$R_w = \max\ SUM(\tau) \tag{3.12}$$

Wobei der Wert R_w der maximalen Korrelation und die dazugehörige zeitliche Verzögerung τ als Merkmal gespeichert werden.

3.2.8 Cepstral - Koeffizienten

Cepstral - Koeffizienten werden seit längerem erfolgreich in der Spracherkennung eingesetzt um die Anregungs- von der Vokaltraktübertragungsfunktion zu trennen [Rig02a] und können in diesem Bereich gute Erfolge erzielen. Deshalb fanden sie auch Anwendung in der Verarbeitung von musikalischen Signalen und stellten sich auch auf diesem Gebiet als sehr geeignet heraus.

Melfrequenz Kepstral Koeffizienten (MFCC) berechnen sich in 4 Schritten [Log00]:

- Fensterung des Signals in quasistationäre Abschnitte

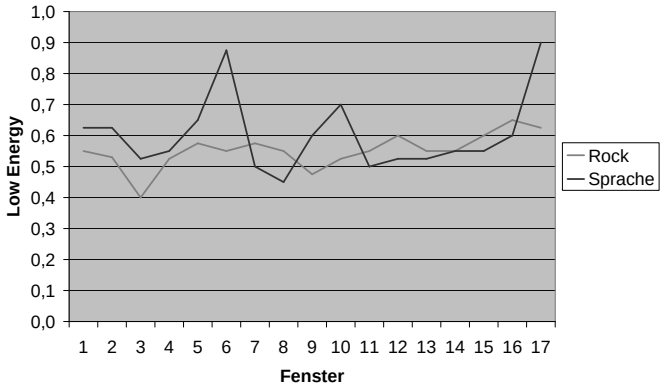


Abbildung 3.9: Vergleich des Low Energy Anteils für Rock – Musik und Sprache

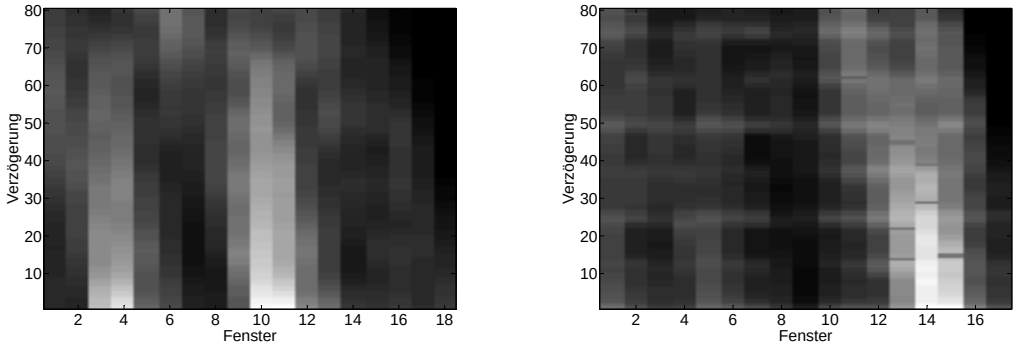
- Transformation in den Frequenzbereich
- Logarithmierung
- Transformation auf die Melskala
- Anwendung einer diskreten Kosinus Transformation

Nachdem in diesem Fall die Frequenzinformation schon in Kurzzeitspektren vorliegt, kann auf die Fensterung und Wandlung in den Frequenzbereich verzichtet werden. Auf die Transformation auf die Melskala wurde in diesem Fall auch verzichtet, da in Versuchen bisher nur gezeigt werden konnte, dass die Transformation auf die Melskala *keine Nachteile* [Log00] auf die Verarbeitung von Musiksignalen hat.

Somit ergibt sich die Berechnung der Cepstral - Koeffizienten zu

$$c_n = DCT\left(\log \overline{E}_w(sb)\right) \tag{3.13}$$

Wobei $\overline{E}_w(sb)$ der Mittelwert der Subbandenergie über ein Analysefenster von 40 Frames ist.



(a) Klassik (b) Pop

Abbildung 3.10: Vergleich der AKF für Klassik und Pop - Musik. Bei der Pop - Musik lassen sich bei einer Verzögerung von jeweils 12 Frames Maxima der Korrelation erkennen.

3.2.9 Merkmalsvektor

Die frameweise berechneten Merkmale werden zu einem größeren Analysefenster zusammengefasst. Hierzu wird über Fenster von 40 Frames – dies entspeicht einer Dauer von $1,04$ Sekunden – der Mittelwert und die Standardabweichung gebildet. Die Analysefenster überlappen sich jeweils zu 50%, sodass jede halbe Sekunde ein Merkmalsvektor berechnet wird, der das Musiksignal über die Dauer von einer Sekunde beschreibt. Zusammen mit den fensterweise berechneten Merkmalen, Low Energy, AKF und den ersten 10 Cepstral - Koeffizienten ergibt sich der Merkmalsvektor mit 85 Merkmalen.

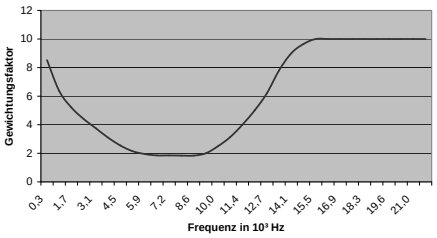


Abbildung 3.11: Gewichtung der einzelnen Subbänder zur Rhythmuserkennung

Es entstehen für ein n Sekunden langes Musikstück $n \cdot 2 - 1$ Merkmalsvektoren, die dieses Lied beschreiben. Diese Merkmalsvektoren bilden die Grundlage für die weitere Verarbeitung des Musiksignals.

Kapitel 4

Inhaltsanalyse von Musikstücken

Die inhaltsbasierte Analyse von Musikstücken stellt eine Klassifikation anhand der musikalischen Eigenschaften der Musik dar. Hier ist die Einteilung der Musik anhand der Musikstile und eine Unterscheidung nach Live- und Studioaufnahmen untersucht worden.

Bei dieser Aufgabe handelt es sich um einen klassischen dreistufigen Aufbau eines Mustererkennungssystems: Nach der Berechnung der Merkmale in Kapitel 3.2.9 muss mit diesen ein Klassifikator trainiert werden, sodass dieser in der Lage ist die Klassentrennung vorzunehmen.

4.1 Theorie der Support Vector Machines

Zur Klassifikation der Merkmale wurden *Support Vector Machines (SVM)* verwendet, da sich diese zur Klassifikation von Musik, wie auch in vielen anderen Bereichen der Mustererkennung, als sehr geeignet erwiesen haben.

SVMs transformieren ein linear nicht trennbares Problem in einen höherdimensionalen Merkmalsraum, in dem die Datensätze linear trennbar sind.

Ein Trainingsdatensatz

$$(x_1, x_2, ..., x_n), \quad x_i \in R^n \tag{4.1}$$

mit den dazugehörigen Klassen

$$(y_1, y_2, ..., y_n), \quad y_i \in \{-1, +1\} \tag{4.2}$$

soll in zwei Klassen eingeteilt werden. Eine Ebene

$$\omega^T \Phi(x) + b = 0 \tag{4.3}$$

im hochdimensionalem Merkmalsraum soll diese trennen, wobei Φ die Transformation in den höherdimensionalen Merkmalsraum ist. Diese Ebene wird von den Support Vektoren aufgespannt.

Daraus ergibt sich die Bedingung für den Klassifikator:

$$y(x) = sgn[\omega^T \Phi(x) + b] \tag{4.4}$$

Hieraus ergibt sich zur Bestimmung der trennenden Hyperebene für nicht-linear trennbare Datensätze das Optimierungsproblem [HCL01]:

$$\min_{\omega,b,\xi} \quad \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n}\xi_i \tag{4.5}$$

mit der Lösung:

$$y_i\left(\omega^T\Phi(x_i) + b\right) \geq 1 - \xi_i, \quad \xi_i > 0 \tag{4.6}$$

Wenn der Datensatz nicht linear trennbar ist, wird der nicht-lineare SVM Klassifikator verwendet um die Merkmalsvektoren x_i mittels einer nichtlinearen Transformation Φ in einen höherdimensionalen Merkmalsraum zu transformieren um in diesem hochdimensionalem Raum den Datensatz linear trennen zu können. $C \geq 1$ ist die Gewichtung des Fehlers und $K(x, y) \equiv \Phi(x_i)^T\Phi(y_i)$ die Kernel Funktion.

Vier Arten von Kernelfunktionen finden hauptsächlich Verwendung:

- Linearer Kernel
  $K(x_i, y_i) = x_i^T y_i$
- Polinominal Kernel vom Grad d
  $K(x_i, y_i) = (\gamma x_i^T y_i)^d, \quad \gamma > 0$
- Radialer Kernel (RBF)
  $K(x_i, y_i) = \exp(-\gamma\|x_i - y_i\|^2), \quad \gamma > 0$
- Sigmoider Kernel
  $K(x_i, y_i) = \tanh(\gamma x_i^T y_i)$

Die Parameter, anhand derer sich die SVM Klassifikation beeinflussen lässt sind somit die Fehlerbestrafung C, auch Komplexität genannt, die verwendete Kernelfunktion und deren Parameter γ und der Exponent d.

4.2 Genre - Erkennung

Die Genre - Erkennung ist der Versuch die Musik anhand der Musikstile in bestimmte, vordefinierte Kategorien einzuteilen.

4.2.1 Musikalische Genres

Bei dieser Aufgabe tritt das Problem auf, dass die Genre - Einteilung sehr subjektiv und abhängig vom jeweiligen Zuhörer und dem kulturellen Umfeld ist. Vor diesem Hintergrund können Genres als Klassen von Musik betrachtet werden, die gemeinsame Eigenschaften haben. Diese Eigenschaften beinhalten z.B. die Instrumentalisierung, Rhythmus und Tempo. Bei der in dieser Arbeit verwendeten Einteilung wurde keine Rücksicht auf andere Eigenschaften gelegt, die der Genre - Einteilung dienen, wie kulturelle, zeitliche oder geographische Unterschiede.

4.2.2 Datenbank zur Genre - Erkennung

Zur Evaluierung der Ergebnisse der Genre - Klassifikation wurde die Musik in 4 Klassen eingeteilt. Diese 4 Klasse entsprechen den Musikstilen *Rock*, *Pop*, *Klassik* und *Jazz/Blues/Folk*. Aufgrund der Schwierigkeiten, die bei der manuellen Genre - Einteilung auftreten handelt es sich hierbei nicht um eine musikwissenschaftlich korrekte, sondern eine subjektive Einteilung anhand der Stilrichtungen.

Zusätzlich wurde eine Klasse *Sprache* verwendet, damit die Eignung des System zur Unterscheidung von Musik und Sprache getestet werden kann. Diese Klasse enthielt verschiedenste Hörbücher.

Der zum Training des Klassifikators verwendete Datensatz enthielt je Klasse ca. $2,5$ Stunden an Daten. Diese bestanden sowohl aus ganzen, diesen Klassen zugeordneten Liedern und zusätzlich besonders aussagekräftigen Samples. In den Trainingsdatensatz wurden nur Daten aufgenommen, die eindeutig einer der Klassen zugeordnet werden konnten, grenzwertige oder nicht eindeutige zuordbare Musikstücke wurden nicht verwendet.

Die Klasse *Sprache* enthielt aufgrund der geringeren Varianz weniger Trainingsdaten.

4.2.3 Klassifikation

Die Genre - Klassifikation der Musikstücke erfolgte auf Basis der in Kapitel 3.2.9 fensterweise berechneten Merkmalsvektoren. Da jeder dieser Vektoren für sich klas-

Klasse	Samples	Dauer in Minuten
Rock	14894	124,1
Pop	14497	120,8
Klassik	15630	130,2
Sprache	12749	106,2
Jazz/Blues/Folk	14594	121,6

Tabelle 4.1: Struktur der Trainingsdaten zur Genre - Klassifikation

sifiziert wird ergibt sich für jede Sekunde des Musikstückes ein Klassifikationsergebnis.

Anhand aller Ergebnisse eines Liedes wird ein Mehrheitsentscheid durchgeführt und das Lied der Klasse zugeordnet, die am häufigsten erkannt worden ist. Dieses Verfahren gewichtet somit die Klassifikationsergebnisse häufig wiederkehrender und verhältnismäßig lange andauernder Abschnitte eines Musikstückes höher, während kurze Abschnitte das Ergebnis nur wenig beeinflussen.

4.3 Erkennen von Live - Aufnahmen

Die Klassifikation eines Musikstückes in Studio- oder Live-Aufnahme basiert auf der Suche nach typischen Geräuschen einer Zuhörerschaft, wie z.B. Klatschen, Pfeifen und Jubeln.

Die Klassifikation eines Liedes erfolgt in zwei Schritten:

- Klassifikation des Anfangs und Ende eines Liedes auf Klatschen oder Musik
- Mehrheitsentscheid zur Einordnung des Liedes

4.3.1 Datenbank zur Erkennung von Live - Aufnahmen

Das Traningsmaterial für die Live - Klassifikation bestand aus insgesamt 65 Minuten Audiodaten. Als Positivbeispiele wurden aus Live - Aufnahmen manuell Abschnitte ausgeschnitten, in denen nur Publikum zu hören ist. Somit besteht das Positivmaterial aus Publikumsgeräuschen, die von unterschiedlichsten Liedern, Alben und damit Konzerten stammen und ein breites Spektrum an Klatschgeräuschen abdecken.

Die Negativbeispiele bestehen aus zufällig ausgewählten, kurzen Musik - Abschnitten verschiedener Musikrichtungen. Zusätzlich sind in den Daten Samples typischer

Liedanfänge und -enden enthalten, wie z.B. lang ausklingende Gitarrenklänge um das Material zu verbessern.

Das Trainingsmaterial der Garbage - Klasse enthält verschiedenste Formen von Rauschen, Knacksen und ähnlichen Störgeräuschen sowie Ausschnitte von Sprachaufnahmen.

Klasse	Anzahl der Sample	Dauer in Minuten
Live	2673	22,3
nicht Live	3469	28,6
Garbage	1785	14,9

Tabelle 4.2: Struktur der Trainingsdaten zur Live - Klassifiaktion

Die höhere Varianz der Musik erfordert eine größere Anzahl an Trainingsdaten in der Negativ - Klasse.

4.3.2 Klassifikation

Der Anfang und das Ende eines Liedes wird fensterweise nach Klatschen oder Musik klassifiziert. Sodass eine Aussage getroffen werden kann, ob in der Aufnahme, bevor die Musik beginnt, bzw. nachdem sie endet, Publikumsgeräusche vorhanden sind. Hierzu werden die in Kapitel 3.2.9 berechneten Merkmalsvektoren verwendet.

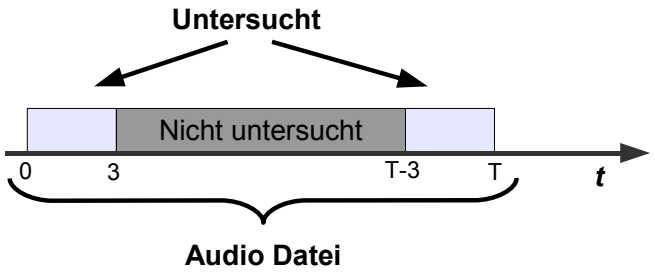


Abbildung 4.1: Klassifikation am Anfang und Ende eines Liedes

Es erfolgt eine Einteilung 3 Klassen:

- Musik
- Publikumsgeräusche
- Garbage

Die *Garbage* Klasse beinhaltet alle Geräusche, die weder Musik noch Klatschen sind, also z.B. Stille, Rauschen und Knacksen das am Ende einer Aufnahme recht häufig vorkommt, als auch Sprache.

Bei der Klassifikation werden alle als *Garbage* klassifizierten Fenster übersprungen und somit ein Verfahren ermöglicht, das Analysefenster dynamisch über Lied - Anfang und -Ende gleiten zu lassen um nach aussagekräftigen Fenstern – Musik oder Klatschen – zu suchen.

Anhand einer festzulegenden Schwelle kann dann eine Entscheidung über die Aufnahmesituation des Liedes getroffen werden.

Kapitel 5

Inhaltsbasierte Ähnlichkeitssuche

Die *inhaltsbasierte Ähnlichkeitssuche* beschreibt den Versuch anhand spezifischer Merkmale eines Musikstückes *ähnliche* oder *das ähnlichste* Musikstück aus einer beliebig großen Datenbank zu finden. Somit ist es möglich homogene Abspiellisten zu erstellen, indem ein Titel vorgegeben wird und das System ähnliche Musikstücke findet.

In Verbindung mit der externen Datenbank eines Online – Musikladens ergäbe sich die Möglichkeit die unüberschaubare Menge an Künstlern und Titeln schnell nach den eigenen Präferenzen zu durchsuchen.

Der Vorteil dieses Verfahrens ist die Reduktion der Suche auf rein inhaltsbasierte Informationen.

5.1 Ähnlichkeit von Musik

Das Problem der Suche nach ähnlichen Musikstücken liegt in der Definition des Begriffs, da diese sehr subjektiv ist und sich kaum reproduzieren lässt.

Die Klangfarbe spielt beim Ähnlichkeitsempfinden eine große Rolle, da diese das Maß für die Instrumentalisierung der Musik ist. Musik mit dominanten, verzerrten elektrischen Gitarren ist subjektiv einem klassischen Musikstück mit überwiegend Streichinstrumenten nicht ähnlich.

Auch der Rhythmus ist eine wichtige Eigenschaft für die Ähnlichkeit zweier Lieder. Ein schnelles Lied mit ausgeprägtem Rhythmus hat nur geringe Ähnlichkeit zu einem langsamen Lied mit kaum rhythmischer Struktur.

5.2 Verwendete Merkmale

Das entwickelte System zur Bestimmung des *Abstandes* zwischen beliebigen Liedern nutzt ebenfalls die in Kapitel 3.2.9 beschriebenen Merkmale.

Für diese Aufgabe ist es notwendig eine Beschreibung zu finden, die unabhängig von der Länge des Liedes ist. Somit sind die fensterweise berechneten Merkmalsvektoren für die Bestimmung eines Abstandes zwischen den Liedern nicht verwendbar.

Hier wird die einfache Methode gewählt, den Mittelwert und die Standardabweichung der fensterweise berechneten Merkmale über die gesamte Länge des Liedes zu berechnen. Es ergibt sich somit für jedes Lied ein Vektor mit 170 Merkmalen, der für jedes Lied die Position in einem 170 - dimensionalen Raum beschreibt. Über diese Position im Raum lassen sich zu jedem Lied die oder der nächste Nachbar und damit die ähnlichsten Lieder aus der Datenbank finden.

5.3 Abstandsberechnung

Die Bestimmung der Ähnlichkeit zwischen den Merkmalsvektoren erfolgt mittels einer *Nearest Neighbor* Suche [MA97].

Jeder Vektor p, der ein Lied repräsentiert hat 170 Einträge

$$p = (p_0, p_1, ..., p_{169}) \tag{5.1}$$

Der Abstand eines Anfragevektors p und eines beliebigen Vektors aus der Datenbank q, also zweier Punkte im Raum berechnet sich dann über den Euklidischen Abstand

$$dist(p,q) = \sqrt{\left(\sum_{0 \leq i \leq 169} (p_i - q_i)^2 \right)} \tag{5.2}$$

Der nächste Nachbar zu einem bestimmten Lied aus der Datenbank mit einer bestimmten Anzahl N an Liedern lässt sich somit durch den Vergleich der berechneten Abstände und dem Auffinden des Vektors mit dem geringsten Abstand erhalten.

$$\min\ dist(p,q)\,, \quad (0 \leq q \leq N) \tag{5.3}$$

Mit dieser Methode ist es möglich die Datenbank nach dem Merkmalsvektor mit dem geringsten Abstand zum Anfragevektor zu durchsuchen.

Kapitel 6

Textbasierte Suche mit Fehlertoleranz

6.1 Einführung

In den vorhergehenden Kapitel wurden inhaltsbasierte Möglichkeiten vorgestellt Musikstücke anhand ihrer musikalischen Eigenschaften und der Wahrnehmung des Menschen zu analysieren. Dies ermöglicht es dem Benutzer gänzlich unbekannte, aber bestimmten Eigenschaften entsprechende Musikstücke zu finden.

Oftmals ist aber der Nutzer auf der Suche nach einem *bestimmten* Musikstück von dem er allerdings nur Fragmente des Titels oder des Interprets kennt oder sich dessen Schreibweise nicht sicher ist. Herkömmliche Suchfunktionen stoßen hier sehr schnell an ihre Grenzen, da sie nicht dynamisch auf Schreibfehler, unterschiedliche Schreibweisen und andere Abweichungen reagieren können. Ein großes Problem stellen Fehler dar, die bei der Erstellung der Datenbank, also dem Versehen eines jeden Musikstückes mit Titel, Interpret, Album und eventuell einer Vielzahl von weiteren Tags, aufgetreten sind. Wird so ein Musikstück unter falschem Namen indexiert lässt es sich in einer großen Datenbank mit einer normalen, bool'schen Suche kaum oder nur schwer wiederfinden.

6.2 Distance - Metrics

Anders als die klassische Suche nach dem bool'schen Kriterium – wahr oder nicht-wahr – bieten approximative String Matching Verfahren den Vorteil ein Maß für die Ähnlichkeit zweier Strings zu berechnen. Distance Metrics [Nav01] berechnen

		B	i	l	l	y		J	o	e	l
	0	1	2	3	4	5	6	7	8	9	10
B	1	**0**	1	2	3	4	5	6	7	8	9
i	2	1	**0**	1	2	3	4	5	6	7	8
l	3	2	1	**0**	1	2	3	4	5	6	7
l	4	3	2	1	**0**	1	2	3	4	5	6
y	5	4	3	2	1	**0**	1	2	3	4	5
	6	5	4	3	2	1	**0**	1	2	3	4
I	7	6	5	4	3	2	1	**1**	2	3	4
d	8	7	6	5	4	3	2	2	**2**	3	4
o	9	8	7	6	5	4	3	3	2	**3**	4
l	10	9	8	7	6	5	4	4	3	3	**3**

Tabelle 6.1: Levenstein Distance zwischen Billy Joel und Billy Idol

diesen Abstand indem sie versuchen den einen String mit möglichst geringem Aufwand in den anderen zu überführen und aus den dafür benötigten Operationen den Abstand der beiden Strings zu bestimmen.

Hierzu werden verschiedenen Operationen verwendet:

- Insertion: $\delta(\epsilon, a)$, Einsetzen des Buchstaben a.
- Deletion: $\delta(a, \epsilon)$, Entfernen des Buchstaben a.
- Substitution: $\delta(a, b)$ für $a \neq b$, Ersetzen von a durch b.
- Transposition: $\delta(ab, ba)$ für $a \neq b$, Vertauschen von a und b.

Diese Operationen eines einzelnen Buchstaben zur Zeit können mit unterschiedlichen Gewichtungen versehen werden und somit der Abstand zweier Strings von einander anhand der nötigen Operationen angegeben werden.

6.3 Levenstein Distance

Die Levenstein Distance erlaubt *Insertions*, *Deletions* und *Substitutions*. Jede dieser Operationen wird mit der Gewichtung 1 versehen und die Anzahl der nötigen Operationen bestimmt.

Nach Tabelle 6.1 ergibt sich für den Vergleich der beiden Strings „Billi Joel“ und „Billy Idol“ eine Levenstein Distance LD von 3. Auf die Länge des Anfragestrings

		A	l	a	n	i	s		M	o	r	i	s	s	e	t	t	e
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A	1	**0**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
l	2	1	**0**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
a	3	2	1	**0**	1	2	3	4	5	6	7	8	9	10	11	12	13	14
n	4	3	2	1	**0**	1	2	3	4	5	6	7	8	9	10	11	12	13
i	5	4	3	2	1	**0**	1	2	3	4	5	6	7	8	9	10	11	12
s	6	5	4	3	2	1	**0**	1	2	3	4	5	6	7	8	9	10	11
	7	6	5	4	3	2	1	**0**	1	2	3	4	5	6	7	8	9	10
M	8	7	6	5	4	3	2	1	**0**	1	2	3	4	5	6	7	8	9
o	9	8	7	6	5	4	3	2	1	**0**	1	2	3	4	5	6	7	8
r	10	9	8	7	6	5	4	3	2	1	**0**	1	2	3	4	5	6	7
i	11	10	9	8	7	6	5	4	3	2	1	**0**	**1**	2	3	4	5	6
s	12	11	10	9	8	7	6	5	4	3	2	1	0	**1**	2	3	4	5
e	13	12	11	10	9	8	7	6	5	4	3	2	1	1	**1**	**2**	3	4
t	14	13	12	11	10	9	8	7	6	5	4	3	2	2	2	1	**2**	**3**

Tabelle 6.2: Levenstein Distance Alanis Morissette

normiert ergibt sich eine *relative Levenstein Distance* von

$$rLD = \frac{LD}{N_{Anfrage}} = \frac{3}{10} = 0,3 \tag{6.1}$$

Für das Beispiel in Tabelle 6.2, den Vergleich des falsch geschriebenen „Alanis Moriset“ mit dem richtigen „Alanis Morissette“ ergibt sich eine Levenstein Distance $LD = 3$. Normiert auf die Länge des Anfrage Strings ergibt sich eine *relative Levenstein Distance* von

$$rLD = \frac{LD}{N_{Anfrage}} = \frac{3}{14} = 0,21 \tag{6.2}$$

Um dieses Verfahren in einer Suchfunktion zu nutzen wird eine Schwelle der relativen Distanz festgelegt, unterhalb derer der Vergleichsstring als Ergebnis der Suche ausgegeben wird. Somit führen unterschiedliche Schreibweisen, Rechtschreibfehler und andere ungewollte Unterschiede der Strings trotzdem zu Ergebnisse der Suche und sie stellt somit eine Möglichkeit dar die stringbasierte Suche weniger fehleranfällig zu machen.

Kapitel 7

Implementierung

Die vorgestellten Funktionen zur Vereinfachung der Organisation von Musikdatenbanken wurden zur unkomplizierten Anwendung in den mp3 - Player *Foobar2000* integriert.

7.1 Foobar2000

Foobar2000 ist ein frei verfügbarer [Pla] Audio Player für Windows, der sich dadurch auszeichnet, dass ein sehr umfangreiches und mächtiges *Software Development Kit (SDK)* frei verfügbar ist. Dieses ermöglicht es externen Entwicklern eigene Erweiterungen zu erstellen, die die Funktionalität des Players erweitern. Aus diesem Grund gibt es für Foobar2000 eine große Anzahl an Plugins.

Die Wahl viel auf Foobar2000, da dieser eine Datenbank zur Verwaltung vieler Musikdateien unterstützt und dank der *Open Component Architecture* leicht um neue Funktionen erweitert werden kann.

Zur Entwicklung der Plugins wurde Foobar und das SDK in der Version 0.8.3 verwendet.

7.1.1 Foobar2000 SDK

Das *Software Development Kit* [Pla] des Foobar Audio Players bietet umfassende Funktionen, die zur Entwicklung eigener Plugins nötig sind. Die wichtigsten, hier verwendeten Funktionen sind:

- Datenbank Verwaltung mit Zugriff auf Dateiinformationen und Tags

- Playlist Verwaltung
- Integration in das User Interface mit Menüeinträgen und Fenstern

Außerdem bietet das SDK eigene Klassen zur String-Manipulation, zum Fenster-Management und viele weitere Funktionen.

Zum Einstieg in die Plugin - Entwicklung empfiehlt sich die Readme Datei, die im SDK [Pla] enthalten ist sowie die verfügbaren *Developer Tutorials* [Tuta]. Diese erklären Schrittweise die grundlegende Struktur des SDK und der Plugins. Außerdem gibt es im Entwicklerforum [For] eine Vielzahl von im Sourcecode frei verfügbaren Plugins, die sich als Anregung zum Verstehen des Aufbaus verwenden lassen. Dieses Forum ist die erste Anlaufstation bei Fragen und Problemen mit dem SDK, mit vielen hilfsbereiten Mitgliedern.

7.2 Implementierte Funktionen

7.2.1 Klassifikation

Die Funktion der entwickelten Genre - und Live - Klassifikation wird durch das Plugin *Foo_Classification* bereitgestellt. Dieses beinhaltet die Merkmalsberechnung und anschliessende Klassifikation der Musikdaten.

Dieses Plugin stellt zwei Kontextmenüeinträge zu Verfügung, einen im Hauptmenü *Components*, der jede Datei der Datenbank im Hintergrund klassifiziert sowie einen im Kontextmenü der Playlist, der nur die markierten Songs verarbeitet. Die Datenbank wird nach dem Dateinamen der zu verarbeitenden Datei abgefragt, mit diesem wird die Merkmalsgewinnung und Klassifikation aufgerufen und als Rückgabewerte werden die Ergebnisse der Klassifikation und der Vektor zur Ähnlichkeitssuche erhalten.

Diese Ergebnisse werden als Meta Information zu der verarbeiteten Datei in die Datenbank geschrieben. Diese Tags sind:

- AUTOLIVE: Ergebnis der Live-Klassifikation, *Live* oder *Studio*
- AUTOGENRE: Ergebnis der Genre-Klassifikation, *Rock*, *Pop*, *Classic*, *Speech*, *Jazz/Blues/Folk*
- SIMILARITY: 170 - dimensionaler Vektor zur Ähnlichkeitssuche

Die Speicherung als Meta Information in der Datenbank ermöglicht die weitere Verwendung dieser Information an beliebiger Stelle in Foobar2000 [Tag] und damit eine inhaltsbasierte Organisation der Datenbank mit den von Foobar2000 bereitgestellten Funktionen der Meta - Informations - Verarbeitung.

7.2.2 Inhaltsbasierte Ähnlichkeitssuche

Die Suche nach ähnlichen Musikstücken in der Datenbank ist in dem Plugin *Foo_Similarity* implementiert. Für diese Suche wird der von der Klassifiktaion berechnete und als Tag gespeicherte Similarity Vektor verwendet. Da Foo_Similarity im Hinblick auf eine schnelle Suche keine Merkmale berechnet können nur Titel verarbeitet und gefunden werden, für die vorher mit Foo_Classification die notwendigen Merkmale berechnet wurden.

Die Suche nach ähnlichen Titeln läßt sich im Kontextmenü der Playlist starten. Für diese werden die *Similarity* Vektoren der Anfragedatei mit jedem in der Datenbak vorhandenen Vektor verglichen und die eingestellte Anzahl nächster Nachbarn im Ergebnisfenster ausgegeben. Diese Ergebnisliste lässt sich als Playlist exportieren.

7.2.3 Textbasierte Suche mit Fehlertoleranz

Die Suchfunktion nach textbasierter Meta - Information auf Basis der Levenstein Distance *Foo_DBsearch_Levenstein* wurde als Erweiterung für die Foobar2000 Standard Suchfunktion *Foobar Database Search* [Sea] implementiert. Für diese Suchfunktion gibt es ein eigenens Mini-SDK [Sea], das es ermöglicht eigene Suchfilter zu integrieren. Die Levenstein Suche ist so neben den Standard - Filtern, (match exact, match all, boolean...) auswählbar und kann auch auf alle Suchkriterien und -orte zurückgreifen. So ist es möglich die zu durchsuchenden Tags zu wählen und ob die gesamte Datenbank oder nur die momentane Playlist durchsucht werden soll. Der hier verwendete Levenstein Algorithmus wurde von Daniel Modrow im Rahmen einer Diplomarbeit [Mod04] entwickelt.

7.2.4 Textbasierte Suche nach Duplikaten

Die Suche nach mehrfach in der Datenbank vorhandenen Titeln wurde in das Plugin *Foo_Find_Duplicates* integriert. Dieses Plugin lässt sich über das Hauptmenü *Components* aufrufen und vergleicht jeden in der Datenbank gespeicherten Titel im Format *ARTIST - TITEL* mit jedem anderen. Wenn eine bestimmte Levenstein Distance unterschritten wird das momentane Paar als gleich erachtet und der Ergebnisliste hinzugefügt.

7.2.5 Training des Genre - Klassifikators

Um das selbstständige Trainieren des Genre - Klassifikators zu ermöglichen, damit der Nutzer die Einteilung der Musik nach eigenen Kriterien vornehmen kann, wurde das Plugin *Foo_User_Training* entwickelt. Es lassen sich sechs beliebige, frei benennbare Genres definieren und jedem der Genres aussagekräftige Lieder zuordnen. Nachdem eine aussreichenden Anzahl an Trainingssamples hinzugefügt wurde, kann das Training gestartet werden. Hierbei werden erst die Merkmale berechnet und mit diesen der Klassifikator trainiert. Dieser so erstellte Klassifikator wird bei der Klassifikation durch Foo_Classification bevorzugt behandelt.

7.2.6 Vergabe von Bewertungen

Um dem Nutzer eine eigene Wertung der Musikstücke vornehmen zu lassen wurde *Foo_Rating* entwickelt. Diese Plugin implementiert die Möglichkeit ein *Rating* vorzunehmen. Jedem Lied kann ein Wert von 0 bis 5 zugewiesen werden, der dem persönlichen Interesse an diesem Stück entspricht. Diese Zahl wird unter dem Tag *RATING* in der Datenbank gespeichert.

7.2.7 Evaluierungsfunktion

Das Plugin *Foo_Evaluation* diente hauptsächlich der Evaluierung der Genre - und Live - Erkennung. Der Ordnername jeder Datei in der Playlist wird mit dem AUTOLIVE bzw. AUTOGENRE Tag vergleichen. Die Live Evaluierung gibt die Anzahl der richtig und falsch erkannten sowie der fälschlicherweise akzeptierten und zurückgewiesenen aus. Die Genre - Evaluierung gibt die Konfusionsmatrix aller Ergebnisse aus. Siehe dazu Kapitel 8.1.

7.2.8 Playlist - Kopier - Funktion

Foo_Copy bietet die Möglichkeit eine beliebige Abspielliste, erstellt mit den entwickelten Organisations- und Strukturierungsfunktionen, direkt aus dem Programm heraus an ein beliebiges externes Abspielgerät wie mp3-Player oder Ipod zu kopieren. Hierbei lässt sich die Größe des Zieldatenträgers angeben, auf die die Dateigröße der Abspielliste angepasst wird und nur die Dateien kopiert werden, für die physikalischer Speicherplatz zu Verfügung steht.

7.3 User Interface

Bei der Gestaltung des User Interfaces wurde darauf geachtet einen möglichst großen Nutzen aus den entwickelten Erweiterungen zu ziehen. Natürlich handelt es sich hier nur um einen Vorschlag, der große Vorteil an Foobar2000 ist es, dass jeder Nutzer große Freiheiten hat die Oberfläche seinen eigenen Vorstellungen anzupassen.

Das *Default User Interface* von Foobar2000 wurde durch das *Columns UI* [UI] ersetzt, da dieses mehr Möglichkeiten der Organisation und Sortierung bietet. *Columns UI* teilt das Programmfenster in einzelne Bereiche auf, die frei mit Ansichten belegt werden können. Hauptbestandteil ist die Playlistansicht, die sich im Gegensatz zum Standardinterface durch eine Spaltenansicht auszeichnet. Es können Spalten mit beliebigen Tags hinzugefügt werden, in diesem Fall also die Tags AUTOGENRE, AUTOLIVE und RATING, sodass diese bei jedem Titel angezeigt und die Playlist oder Datenbank nach diesen sortiert werden kann.

Außerdem bietet es die Möglichkeit mehrere so genannte Panels anzuzeigen. Dies sind Erweiterung mit unterschiedlichen Anzeigen. Ein Standardpanel ist der *Playlist Switcher*, der ein Auflistung der verfügbaren Abspiellisten enthält. Andere können das Albumcover anzeigen oder bestimmte Such- und Sortierfunktionen zu Verfügung stellen.

Ein sehr empfehlenswertes Panel ist der *Playlist Tree* [Tutb], ein Plugin das die gesamte Datenbank sortiert nach beliebigen Kriterien anzeigt und daraus Abspiellisten erstellen kann. Hierzu lassen sich Lieder mit beliebigen Tags nach beliebigen Tags sortiert in Ordnern darstellen, diese in ihrer maximalen Dauer, Länge oder Größe beschränken und daraus Playlisten erstellen. Mit diesem Plugin ist es also möglich die gesamte Datenbank sortiert nach Genres anzuzeigen und somit schnellen Zugriff zu erhalten.

Kapitel 8

Ergebnisse

In diesem Kapitel werden die Ergebnisse der einzelnen entwickelten Verfahren vorgestellt und erläutert.

8.1 Evaluierungsmaße

Erkennungsrate Der Vergleich der Erkennungsraten unterschiedlicher Verfahren und unterschiedlicher Parameter erfolgte anhand der Erkennungsrate, also dem Anteil der günstigen Ereignisse zur Anzahl der möglichen Ereignisse

$$ER = \frac{\text{Anzahl der günstigen Ereignisse}}{\text{Anzahl der möglichen Ereignisse}} \tag{8.1}$$

Zusätzlich wurde die *False Acceptance Rate* und die *False Rejection Rate* berechnet. Diese geben eine genauere Beschreibung des Klassifikationsergebnis, als die Erkennungsrate allein.

False Acceptance Rate Die *False Acceptance Rate* ist die Anzahl der fälschlicherweise der positiven Klasse zugeordneten Samples zur Anzahl der gesamten Negativ-Muster.

$$FAR = \frac{\text{Anzahl positiv erkannter Negativ-Muster}}{\text{Anzahl Negativ-Muster}} \tag{8.2}$$

False Rejection Rate Die *False Rejection Rate* ist die Anzahl der fälschlicherweise der negativen Klasse zugewiesenen Positiv-Beispiele zur gesamten Anzahl an positiven Samples.

$$FRR = \frac{\text{Anzahl negativ erkannter Positiv-Muster}}{\text{Anzahl Positiv-Muster}} \tag{8.3}$$

Konfusionsmatrix Zur genaueren Darstellung der Ergebnisse der Klassifikation wird eine *Konfusionsmatrix* verwendet. Diese stellt zu jeder Klasse die Gesamtzahl der Elemente dieser Klasse, die Anzahl der falsch klassifizierten Elemente und zusätzlich welcher Klasse wieviele Elemente zugeordnet wurden gegenüber.

8.2 Ergebnisse der Live - Klassifikation

Das Klassifikationsergebnis ist abhängig von mehreren Parametern:

- Anzahl und Art der verwendeten Merkmale
- Parameter des Klassifikators
- Anzahl der zu analysierenden Fenster
- Höhe der Entscheidungsschwelle

Die Parameterkombination, die die höchste Erkennungsrate ermöglicht muss in einer Evaluierung festgestellt werden.

8.2.1 Testdatensatz

Das Testmaterial auf dem die Evaluierung des Verfahrens erfolgte bestand aus 500 Musikstücken, 250 Live- und 250 Studio - Aufnahmen unterschiedlichster Genres, Interpreten und Aufnahmequalität.

Zwischen Trainings- und Testdatensätzen gibt es keinerlei Überschneidung um aussagekräftige Ergebnisse zu erhalten.

Fenster	Schwelle	Erkennungsrate in %	FAR in %	FRR %
3	2	91,0	8,8	9,2
4	2	91,0	9,6	8,4
4	3	90,8	4,8	13,6
5	3	91,6	5,6	11,2
3	1	89,9	14,0	8,4
7	4	88,4	3,6	19,6
Anfang: 3 Ende: 5	2 3	90,8	5,6	12,8

Tabelle 8.1: Vergleich verschiedener Fenstergrößen und Entscheidungsschwellen

8.2.2 Länge des Analysefensters

Da die Klassifikation eines Musikstückes in Studio- und Liveaufnahme darauf basiert, an den zeitlichen Rändern der Musik, also Beginn und Ende des Musikstückes, das Signal nach den typischen Geräuschen einer Zuhörerschaft zu suchen, ist diese abhängig von dem untersuchten zeitlichen Abschnitt und der Entscheidungsschwelle. In der Tabelle 8.1 sind die Erkennungsraten unterschiedlicher Kombinationen von Analysefensterlängen und Enscheidungsschwellen dargestellt.

Fenster bezeichnet die Anzahl der Fenster, die am Anfang und Ende untersucht werden und *Schwelle* die Anzahl an positiv, also als *Applaus* klassifizierten Fenster die nötig sind, damit das Lied als Liveaufnahme angenommen wird. Anfang und Ende eines Liedes werden unabhängig untersucht und bei einem positiven Ergebnisse am Anfang oder Ende positiv entschieden.

Die höchsten Erkennungsraten lieferte eine Analyse von jeweils 5 Fenstern und das Treffen einer Mehrheitsentscheidung. Durch Fensterüberlappung entspricht dies einer Analysefensterlänge von 3 Sekunden. Durch den Mehrheitsentscheid können somit Liveaufnahmen erkannt werden, die entweder am Anfang oder am Ende 2 Sekunden Publikumsgeräusche enthalten.

8.2.3 Merkmale

In der Tabelle 8.2 sind die Erkennungsraten unter Verwendung verschiedener Merkmalskombinationen dargestellt.

Cepstral - Koeffizienten bezeichnet die Verwendung aller 10 berechneten Cepstral - Koeffizienten, Kurzzeitspektren sind die Subbandvektoren sowie die Gesamtenergie, mit *Spektrumsbeschreibung* werden folgende Merkmale zusammengefasst:

Merkmale	Erkennungsrate in %	FAR in %	FRR in %
Cepstral - Koeffizienten	91,0	5,6	12,4
Spektrumsbeschreibung	85,8	15,6	12,8
Kurzzeitspektren	81,6	22,8	14,0
alle	91,4	5,6	11,6

Tabelle 8.2: Vergleich der verschiedenen Merkmale

- Spectral Centroid
- Spectral Rolloff
- Spectral Flux
- Low Energy
- AKF-Merkmale zur Rhythmusbeschreibung

Besonders die Cepstral - Koeffizienten können sehr gute Ergebnisse erzielen. Vor allem, da es sich hierbei nur um eine Beschreibung mit 10 Werten handelt. Hingegen erreicht man nur unter Verwendung der Kurzzeispektren deutlich schlechtere Erkennungsraten, obwohl diese 66 Werte umfasst. Die Spektrumsbeschreibung in Kombination mit den Rhythmusmerkmalen liegt zwischen diesen beiden Ergebnissen, verwendet aber auch nur 9 Merkmale. Die Verwendung aller Merkmale kann die Erkennungsrate noch einmal erhöhen, allerdings steht die Verbesserung des Klassifikationsergebnisses in keinem Verhältnis zu der Steigerung der Anzahl der Merkmale.

Daher wurde eine Merkmalsselektion vorgenommen, um aus diesen 85 Merkmalen diejenigen mit dem größten Informationsgehalt auszuwählen.

8.2.4 Merkmalsselektion

Die Merkmalsselektion wurde sowohl mit der *SVM Attribute Selection* der *Waikato Environment for Knowledge Analysis – WEKA* [WF05] als auch einem im Rahmen der Diplomarbeit von Ludwig Wüstner entwickelten Verfahren auf Basis des *AdaBoost Algorithmus* [Wüs06] durchgeführt. Die Ergebnisse in Abhängigkeit der Merkmalsanzahl sind in Tabelle 8.3 aufgeführt.

Die höchste Erkennungsrate konnte bei der Verwendung von 40 Merkmalen erreicht werden. Die Halbierung der Merkmalsanzahl führt somit zu besseren Ergebnissen als die Verwendung aller Merkmale.

Anzahl der Merkmale	Erkennungsrate in %	FAR in %	FRR in %
19	90,2	14,8	4,8
37	91,6	6,4	10,4
40	92,0	6,0	10,0
60	89,8	8,4	12,0
85	91,4	5,6	11,6

Tabelle 8.3: Erkennungsraten in Abhängigkeit der Anzahl der selektierten Merkmale

Bei diesen 40 selektierten Merkmalen handelt es sich um:

- Mittelwert der Gesamtenergie
- Mittelwert der Energie der Subbänder 0, 1, 3, 6, 8, 10, 11, 14, 15, 18, 19, 20, 21, 23 und 24
- Standardabweichung der Energie der Subbänder 0, 1, 3, 5, 9, 16, 17, 18, 19, 20, 21 und 25
- Mittelwert und Standardabweichung des Spectral Centroids und Spectral Rolloff Faktors
- Cepstral - Koeffizienten mit Index 0, 1, 2, 3, 4, 6, 7 und 8

8.2.5 Parameter des Klassifikators

In der Tabelle sind die unterschiedlichen Kombinationen der Einstellungen des verwendeten SVM - Klassifikators und die sich dabei ergebenden Erkennungsraten gegenübergestellt. Die Beschreibung der Parameter erfolgte in Kapitel 4.1.

Die Grundlage bilden die aus der Merkmalsselektion hervorgegangenen 40 selektierten Merkmale.

8.3 Ergebnisse der Genre - Klassifikation

Die Parameter, die das Ergebnis der Genre - Klassifikation beeinflussen sind:

- Anzahl und Art der verwendeten Merkmale
- Parameter des Klassifikators

Die optimale Prameterkombination wurde in einer Evaluierung ermittelt.

Kernel	d	C	Erkennungsrate in %	FAR in %	FRR in %
Radial	–	1	92,0	6,0	10,0
	–	3	90,4	8,8	10,4
Polynominal	1	1	89,4	8,0	13,2
	1	3	90,2	9,2	10,4
	2	1	91,2	7,6	10,0
	2	3	91,6	7,2	9,6
	3	1	92,0	6,0	10,0
	3	3	89,0	13,6	8,4

Tabelle 8.4: Vergleich der SVM Parameter

8.3.1 Testdatensatz

Der zur Evaluierung der Ergebnisse der Genre - Klassifikation verwendete Datensatz enthielt jeweils 100 Musiktitel der Klassen *Rock*, *Pop*, *Klassik* und *Jazz/Blues/Folk* und 100 Sprachaufnahmen in der Klasse *Sprache*. Somit fand die Evaluierung auf 500 Audiodateien statt.

Zwischen Trainings- und Testdatensätzen gibt es keinerlei Überschneidung um aussagekräftige Ergebnisse zu erhalten.

8.3.2 Merkmale

In der Tabelle 8.5 werden die Erkennungsraten bei der Verwendung unterschiedlicher Anzahl und Kombination der berechneten Merkmale verglichen. Die Beschrei-

Merkmale	Erkennungsraten in %
Cepstral - Koeffizienten	81,2
Kurzzeitspektren	75,8
Spektrumsbeschreibung	85,6
Spektrumsbeschreibung und Cepstral - Koeffizienten	86,0
Alle	86,6

Tabelle 8.5: Vergleich unterschiedlicher Merkmale

bung der Merkmalsgruppen erfolgte in Kapitel 8.2.3. Aussagekräftiger sind hier die Konfusionsmatrizen der Klassifikation, da daraus die Fähigkeit der einzelnen Merkmale die einzelnen Klassen zu unterscheiden erkennbar ist.

		Rock	Pop	Klassik	Jazz/Blues/Folk	Sprache
Erkannt	Rock	**91**	27	0	4	0
	Pop	8	**61**	0	13	0
	Klassik	1	0	**95**	5	0
	Jazz	0	10	5	**75**	16
	Sprache	0	2	0	3	**84**
Gesamt		100	100	100	100	100
Erkennungsrate		81,2%				

Tabelle 8.6: Konfusionsmatrix unter Verwendung der Cepstral - Koeffizienten

Die alleinige Verwendung der 10 Cepstral - Koeffizienten ist nach Tabelle 8.6 für die Genre - Erkennung nicht ausreichend. Auch die Unterscheidung von Musik und Sprache bringt keine überzeugenden Ergebnisse. Die Klassifizierung von klassischer Musik gegen alle andere Genres gelingt dagegen recht gut. Hier äußert sich die Eigenschaft der Cepstral - Koeffizienten nur die Klangfarbe zu beschreiben, jedoch nicht den Rhythmus.

		Rock	Pop	Klassik	Jazz/Blues/Folk	Sprache
Erkannt	Rock	**93**	25	0	6	0
	Pop	4	**61**	0	8	1
	Klassik	1	1	**100**	9	0
	Jazz	2	10	0	**77**	2
	Sprache	0	3	0	3	**97**
Gesamt		100	100	100	100	100
Erkennungsrate		85,6%				

Tabelle 8.7: Konfusionsmatrix unter der Verwendung der Spektrumsbeschreibung

Die Merkmale zur Spektrumsbeschreibung, die auch die Merkmale zur Rhythmusbeschreibung beinhalten, liefern sehr viel bessere Erkennungsraten, siehe Tabelle 8.7, als die ausschliessliche Verwendung der Cepstral - Koeffizienten. Hier zeigt sich deutlich, dass es nicht ausreicht nur die spektrale Eigenschaften des Musiksignals zu berücksichtigen. Weitere Beschreibungen, wie die des Rhythmus und der zeitlichen Änderung scheinen hier die Erkennungsraten deutlich zu erhöhen.

Auffällig ist die relativ hohe Konfusion der Klassen Rock und Pop. Da die Grenze zwischen diesen Klassen fließend ist und auch der Mensch mit der Klassifizierung oft Schwierigkeiten hat, entspricht dies den Erwartungen.

Die alleinige Verwendung der rohen Frequenzinformation liefert erwartungsgemäß

		Rock	Pop	Klassik	Jazz/Blues/Folk	Sprache
Erkannt	Rock	**94**	25	0	7	0
	Pop	4	**65**	0	11	4
	Klassik	1	0	**96**	16	0
	Jazz	1	8	4	**65**	37
	Sprache	0	2	0	1	**59**
Gesamt		100	100	100	100	100
Erkennungsrate		75,8%				

Tabelle 8.8: Konfusionsmatrix unter Verwendung der Kurzzeitspektren

die schlechtesten Erkennungsraten, siehe Tabelle 8.8. Auffällig ist die schlechte Separierbarkeit von Sprache und Musik.

		Rock	Pop	Klassik	Jazz/Blues/Folk	Sprache
Erkannt	Rock	**96**	29	0	7	0
	Pop	4	**58**	0	8	0
	Klassik	0	1	**97**	6	0
	Jazz	0	11	3	**79**	0
	Sprache	0	1	0	0	**100**
Gesamt		100	100	100	100	100
Erkennungsrate		86,0%				

Tabelle 8.9: Konfusionsmatrix unter Verwendung der Cepstral - Koeffizienten und der Spektrumsbeschreibung

Durch die Kombination der Cepstral - Koeffizienten und der Spektrumsbeschreibung, die auch die Rhythmusbeschreibung beinhaltet, lassen sich die Erkennungsraten noch einmal erhöhen und die Nachteile der beiden Merkmalsgruppen ausgleichen, siehe Tabelle 8.9. Hier zeigt sich der Vorteil der Kombination von Merkmalen, die unterschiedliche Wahrnehmungen beschreiben. Die Erkennungsrate ist, obwohl nur 19 Merkmale verwendet werden, gut.

Die Verwendung aller 85 berechneten Merkmale kann das Klassifikationsergebnis nocheinmal steigern, siehe Tabelle 8.10. Die Erhöhung der Merkmalsanzahl steht allerdings in keinem Verhältnis zur Steigerung des Klassifikationsergebnisses.

		Rock	Pop	Klassik	Jazz/Blues/Folk	Sprache
Erkannt	Rock	**96**	26	0	7	0
	Pop	4	**63**	0	11	0
	Klassik	0	0	**99**	6	0
	Jazz	0	11	1	**76**	0
	Sprache	0	1	0	0	**100**
Gesamt		100	100	100	100	100
Erkennungsrate		86,6%				

Tabelle 8.10: Konfusionsmatrix unter der Verwendung aller Merkmale

8.3.3 Merkmalsselektion

Im vorangehenden Abschnitt wurde gezeigt, dass die berechneten Merkmale für die Klassifikation unterschiedlich gut geeignet sind und die Ergebnisse je nach verwendeten Gruppen von Merkmalen stark schwanken. Auffällig ist, das die schon durch die Verwendung von nur 19 Merkmalen das Ergebnis ähnlich gut sein kann, wie bei der Verwendung aller 85 Merkmale.

Aufgrund dieser Erkenntnisse erfolgte analog zu Kapitel 8.2.4 eine Selektion der Merkmale, die die höchsten Erkennungsraten ermöglichen.

Anzahl	Erkennungsrate in %
20	86,0
26	87,2
31	87,0
40	86,6
50	87,2
60	87,0
85	86,6

Tabelle 8.11: Vergleich der Erkennungsraten unterschiedlicher Anzahl selektierter Merkmale

In Tabelle 8.11 sind die Erkennungsraten unterschiedlicher Anzahl selektierter Merkmale dargestellt. Die höchste Erkennungsraten lassen sich durch die Verwendung von 26 und 50 Merkmalen erreichen. Aufgrund der geringeren Anzahl an Merkmalen ist die Verwendung von 26 Merkmalen als das beste Ergebnis anzusehen.

Bei diesen selektierten Merkmalen handelt es sich um:

- Mittelwert und Standardabweichung der Gesamtenergie
- Mittelwert der Subbandenergie 1, 3, 5, 6, 8, 10, 13, 14 und 17
- Standardabweichung der Subandenergie 0, 1 und 2
- Mittelwert und Standardabweichung des Spectral Centroids
- Mittelwert und Standardabweichung des Spectral Rolloff Faktors
- Wert des Maximums der AKF
- Cepstral - Koeffizienten mit Index 0, 2, 3, 4, 5, 6 und 7

8.3.4 Parameter des Klassifikators

Einen weiteren Einfluss auf das Ergebnis der Klassifikation haben die SVM Parameter. Die erreichten Erkennungsraten unterschiedlicher Parameterkombinationen sind in Tabelle 8.12 dargestellt. Die Vorstellung der Parameter erfolgte in Kapitel 4.1.

Kernel	Parameter	Erkennungsraten in %
Radial	C=1	84,8
	C=3	85,2
Polynomial	d=1, C=1	85,8
	d=1, C=3	86,0
	d=2, C=1	87,2
	d=2, C=3	87,2
	d=3, C=1	86,0

Tabelle 8.12: Vergleich unerschiedlicher Kernelparameter

Die höchsten Erkennungsraten lassen sich somit durch die Verwendung eines *Polynomial Kernels* des Exponenten $d = 2$ und der Komplexität $C = 1$ oder $C = 3$ erreichen.

Die Konfusionsmatrix der Genre - Klassifikation, Tabelle 8.13 spiegelt die Schwierigkeiten wieder, die auch ein ungeübter Mensch bei der Einteilung von Musik in bestimmte Genres hat. Sehr gut unterscheiden lassen sich klassische und nichtklassische Musik sowie Sprache und Musik. Größere Schwierigkeiten treten bei der Unterscheidung von Rock und Pop Musik auf. Da der Übergang zwischen diesen fließend und oft nicht eindeutig ist, entspricht dieses Ergebnis den Erwartungen.

		Rock	Pop	Klassik	Jazz/Blues/Folk	Sprache
Erkannt	Rock	**93**	22	0	7	0
	Pop	6	**67**	0	10	1
	Klassik	1	0	**100**	6	0
	Jazz	0	10	0	**77**	0
	Sprache	0	1	0	0	**99**
Gesamt		100	100	100	100	100
Erkennungsrate		87,2%				

Tabelle 8.13: Endergebnis der Genre - Klassifikation nach der Merkmalsselektion mit einem polynomial Kernel, $d = 3, C = 1$

Außerdem ist Pop - Musik ein sehr weit gefasstes Genre, das sich vielen Elementen anderer Musikstile bedient.

Die Ähnlichkeit der Klasse *Jazz/Blues/Folk* zu *Rock* und *Pop* lässt sich an den Ergebnissen erkennen.

8.4 Ergebnisse der inhaltsbasierten Ähnlichkeitssuche

Die Ergebnisse der subjektiven Ähnlichkeitssuche sind stark abhängig von dem zu Verfügung stehenden Testmaterial und dem sich daraus ergebenden geringsten Abstand zwischen zwei Liedern, da das System immer einen nächsten Nachbarn findet.

Anstatt die Ähnlichkeit des Anfrage- und des vom System gelieferten Ergebnisliedes zu bewerten wurden hier im Vorhinein Klassen ähnlicher Lieder definiert, damit der nötige Datensatz beschränkt und die Aufgabe in eine Klassifikation überführt. Es wird dann die Fähigkeit des Systems untersucht, diese Musikstücke mit einem *k Nearest Neighbor Klassifikator* in diese Klassen einzuteilen.

Es wurde also bei der Evaluierung des Systems nicht die tatsächliche Ähnlichkeit zweier Musikstücke bewertet, sondern die Fähigkeit des Systems, als ähnlich definierte Musikstücke in die gleiche Klasse einzuordnen.

Die Evaluierung erfolgte auf einem Datensatz mit 382 in sechs definierten *Ähnlichkeitsklassen*.

Das Ergebnis der Ähnlichkeitssuche ist allein abhängig von den verwendeten Merkmalen. Da der für jedes Lied berechnete Merkmalsvektor 170 Einträge enthält und

die Ergebnisse der Genre - Klassifikation in Kapitel 8.3 zeigen, dass sich die Erkennungsrate durch eine Selektion geeignetster Merkmale verbessern lässt, erfolgte auch hier eine Merkmalsselektion.

Diese wurde unter *WEKA* [WF05] mittels einer *Subset Evaluierung* der Merkmale durchgeführt. Durch ein Gradientenabstiegsverfahren wird die beste Untergruppe an Merkmale gesucht und zur Klassifikation verwendet.

Anzahl der Merkmale	Erkennungsrate in %
170	92,67
47	93,72

Tabelle 8.14: Ergebnisse der Merkmalsselektion zur Ähnlichkeitssuche

Die Erkennungsraten sind in Tabelle 8.14 dargestellt und konnten durch die Verwendung von 47 anstatt von 170 Merkmalen erhöht werden.

Diese Erkennungsraten besagen nicht, dass das System mit einer bestimmten Wahrscheinlichkeit ähnliche Lieder finden kann, sondern dass aus dem vordefinierten Datensatz dieser Prozentsatz an Liedern in die richtige Klasse eingeteilt werden konnte. Eine Erhöhung dieser Erkennungsrate stellt somit auch eine Verbesserung der Ergebnisse der Suche nach ähnlichen Liedern dar und die quantitativen Ergebnisse spiegeln den subjektiven Eindruck der Ergebnisse der Ähnlichkeitssuche wieder.

Bei den selektierten Merkmalen handelt es sich um:

- Mittelwert und Standardabweichung des Mittelwerts der Frameenergie
- Mittelwert des Mittelwertes der Subbänder 2, 3, 4, 5, 6 und 23
- Mittelwert der Standardabweichung der Subbänder 0, 6, 20 und 21
- Standardabweichung des Mittelwerts der Subbänder 1, 3, 5, 7, 10 und 30
- Standardabweichung des Standardabweichung 2, 7, 8, 11, 23 und 27
- Mittelwert und Standardabweichung des Mittelwerts und der Standardabweichung des Spectral Centroids und des Spectral Flux
- Mittelwert der Standardabweichung und Standardabweichung des Mittelwerts des Spectral Rolloff
- Mittelwert und Standardabweichung der Low Energy Rate

- Mittelwert des AKF Wertes und der Verzögerung
- Mittelwert der Cepstral - Koeffizienten mit Index 0, 2, 3, 5, 7 und 9
- Standardabweichung der Cepstral - Koeffizienten mit Index 0, 1 und 6

Kapitel 9

Zusammenfassung und Ausblick

In dieser Arbeit wurden Möglichkeiten zur Strukturierung und Organisation großer Datenbanken vorgestellt und Anwendungen zur Intergration in den *Foobar2000* - Audio - Player entwickelt, die diese Funktionen bereitstellen.

Das Problem der Fehleranfälligkeit der textbasierten Suche anhand der zu einem Titel gespeicherten Meta - Information konnte durch die Verwendung der *Levenstein Distance* und dem Festlegen einer Ähnlichkeitsschwelle zwischen Anfrage und Referenz wirkungsvoll verringert werden. Die Erweiterung der *Foobar 2000 Datenbanksuche* um diesen Suchfilter ermöglicht eine schnelle und effektive Anwendung.

Zur inhaltsbasierten Suche nach Liedern mit bestimmten musikalischen Eigenschaften wurde sowohl eine Klassifikation in mehrere Klassen anhand verschiedener Kriterien als auch eine Suche nach zueinander ähnlichen Musikstücken entwickelt. Die für diese Mustererkennungs- bzw. Mustervergleichsaufgabe nötige Frequenzinformation konnten direkt aus dem Bitstrom der mp3 - Dateien gewonnen werden. Hieraus wurden Merkmale berechnet, die für die Wahrnehmung von Musik wichtigen Eigenschaften Klangfarbe, Tonhöhe und Rhythmus beschreiben und anhand dieser erfolgte die Klassifikation mit einer *Support Vector Machine*. Zur Bestimmung der Ähnlichkeit von Musikstücken wurde eine *Nearest Neighbor* Suche verwendet.

Auch diese Funktionen wurden in Foobar 2000 integriert und bieten dadurch eine einfache Anwendung. Exemplarische für viele weitere Anwendungen wurde hier die Klassifikation der Musik in fünf Genre sowie in Live- und Studio-Aufnahmen vorgenommen.

Die Integration in den Audio Player und die Speicherung dieser so gewonnenen Information als Meta - Information ermöglicht das Anzeigen, die Suche und Sortierung anhand dieser inhaltsbasierten Einteilungen. Um die Subjektivität der Genre

Einteilung zu umgehen entstand im Laufe der Arbeit die Idee diese vom Nutzer selbst vornehmen zu lassen und die Möglichkeit zu implementieren den Klassifikator mit vorgegebenen Liedern selbst zu trainieren. Desweiteren wurden einige Funktionen implementiert, die es dem Nutzer ermöglichen eigene Bewertungen von Liedern vorzunehmen.

Die Ergebnisse der Genre - Klassifikation spiegeln die Schwierigkeiten wieder, die auch ein ungeübter Mensch bei der Einteilung von Musik in Genre hat, da deren Übergänge fließend sind und sich viele Stücke nicht eindeutig klassifizieren lassen. Sehr gut gelingt die Unterteilung in klassische und nicht-klassische Musik, sowie die Unterscheidung von Musik und Sprache. Die Unterscheidung von Rock und Pop Musik ist aufgrund der Gemeinsamkeiten dieser Genre weniger eindeutig.

Das entwickelte System kann leicht um neue Funktionen erweitert werden. Besonders die Erweiterung um weitere Klassifikationsaufgaben wäre denkbar. Eine Möglichkeit wäre die Klassifikation der Musik nach vorhandenen Instrumenten oder Rhythmus und Tempo, die gleichzeitig als *Highlevel Feature* Verwendung finden könnten. Eine weitere Verbesserung der Ergebnisse könnte durch die stärkere Anpassung an die Wahrnehmung des Menschen erfolgen. Hierzu wären psychoakustische Merkmale auf Basis der Bark Skala, sowie die Verwendung eines Lautheitsmodells denkbar.

Die Genre Klassifikation könnte um eine größere Anzahl an Klassen erweitert werden. Kurze Tests ergaben, dass die stufenweise Klassifikation über einen Baum eine gute Möglichkeit darstellen könnte.

Die Gewinnung des für die inhaltsbasierte Ähnlichkeitssuche verwendeten Merkmalsvektors aus den sekundenweise berechneten Merkmalsvektoren beruht auf dem sehr einfachen Verfahren der Mittelwertsbildung. Hier ließen sich Verbesserungen der Ergebnisse durch die Modellierung der Verteilung der Merkmale eines Liedes erreichen. Hierzu wäre ein *Gaussian Mixture Model (GMM)* denkbar.

Das entwickelte System stellt einen funktionsfähigen Ansatz dar, die Verwaltung großer Musikdatenbanken zu vereinfachen, die Suche in diesen zu erleichtern und anhand musikalischer Eigenschaften durchzuführen, bietet aber auch genug Raum für Erweiterungen und Verbesserungen der Funktionen.

Anhang A

Entwickelte Anwendungen

A.1 Installationsanweisungen

Die entwickelten Plugins benötigen einen installierten Foobar 2000 Audioplayer in der Version 0.8.3 unter Windows.

Datenbank Die interne Datenbankverwaltung von Foobar muss in den PREFERENCES unter DATABASE aktiviert sein. Diese speichert Informationen über jeden Titel und bildet die Grundlage für die entwickelten Systeme.

User Interface Außerdem ist es zu empfehlen unter DISPLAY COLUMNS UI als User Interface Module auzuwählen. Dieses bietet gegenüber dem Standard - Interface den Vorteil einer frei konfigurierbaren Spaltenansicht.

Das Entwickelte System führt zur Beschreibung der Lieder folgende neuen Tags ein:

- AUTOGENRE
- AUTOLIVE
- RATING
- SIMILARITY

Zur einfachen Organisation können Columns UI neue Spalten hinzugefügt werden, die diese Tags anzeigen. Neben der Anzeige für jeden Titel kann so auch nach diesen Informationen sortiert und gesucht werden.

Zur Installation der entwickelten Funktionen werden die Plugins in den Ordner COMPONENTS im Foobar Installationsverzeichnis kopiert, die Registrierung erfolgt automatisch.

Foo_DBSearch_Lev Das Plugin *Foo_dbsearch_lev* zur Levenstein Suche in der Datenbank benötigt zusätzlich das Plugin *foo_dbsearch*, da es einen Suchfilter für diese Standard - Suchfunktion von Foobar 2000 bereitstellt.

Foo_Search_Dupli Die Duplikatensuche über die Levenstein Distance hat keine weiteren Vorraussetungen.

Foo_Classification Für die Klassifikation ist die *fftw3.dll* des FFTW Projektes [FJ05] nötig. Diese muss im Foobar 2000 Installationsordner liegen und ist zur Berechnung der DCT der Cepstral Koeffizienten nötig. Zusätzlich müssen die trainierten Klassifikator Modell Dateien der *LIBSVM SVM Library* [CL01] im Installationsordner liegen. Für die Genre Klassifikation ist dies die *genre.model* für die Live Klassifikation die *live.model*.

Foo_Similarity_Search Die Ähnlichkeitssuche benötigt zusätzlich die Datei *ANN.dll* des Projektes *ANN: A Library for Approximate Nearest Neighbor Searching* [fANNS] im Installationsordner, die die Nearest Neigbor Suche bereitstellt.

Foo_Genre_Train Die online Trainingsfunktion des Genre Klassifikators benötigt ebenfalls *fftw3.dll* zur Merkmalsberechnung, sowie für das Training des Klassifikators die Datei *svmtrain.exe* der *LIBSVM SVM Library* [CL01] im Foobar Installationsordner.

Foo_Rating Dieses Plugin hat keine weiteren Vorraussetzungen.

Foo_Copy Zur Installation dieses Plugins genügt es die Datei in den *Components* Ordner zu kopieren.

Foo_Evaluation Dieses Plugin hat keine weiteren Vorraussetzungen.

A.2 Bedienungsübersicht

Foo_DBSearch_Lev Die Datenbanksuche über die Levenstein Distance steht in der *Database Search* als Schuchfilter zu Verfügung. Hierzu wird die Datenbank Suche über COMPONENTS –> IMS –> DATABASE SEARCH aufgerufen. Als Filter lässt sich dann, neben den Standardsuchfiltern auch *match Levenstein* auswählen.

Foo_Classification Die Klassifikation der Lieder lässt sich über mehrere Wege starten. Im Menü COMPONENTS –> IMS –> CLASSIFICATION lässt sich die Genre und Live Klassifikation für jedes in der Datenbank gespeicherte Lied starten und auch wieder beenden. Das Anhalten der Berechnung ist wichtig, wenn Foobar während der Berechnung geschlossen werden soll, da ansonsten die schon berechneten Ergebnisse verloren gehen. Zur Kontrolle wird in einem Fenster der Status der Berechnung angezeigt. Die zweite Möglichkeit ist es Lieder in der Playlist zu markieren und über das Kontextmenü IMS –> CLASSIFICATION die Klassifikation dieses markierten Liedes nach Live und/oder Genre starten.

Die Ergebnisse der Klassifikation werden als Tags *AUTOLIVE* und *AUTOGENRE* gespeichert und stehen somit foobar-weit zur Organisation zu Verfügung. Zusätzlich speichert diese den für die Ähnlichkeitssuche nötigen Merkmalsvektor unter dem Tag *SIMILARITY* ab.

Foo_Similarity_Search Die Suche nach ähnlichen Titeln lässt sich über den Playlist Kontextmenüeintrag IMS –> FIND SIMILAR SONGS für den markierten Titel starten. Im darauffolgenden Fenster kann die Anzahl der zu suchenden ähnlichen Titel eingestellt und die Suche gestartet werden. Die Ergebnisse werden sortiert nach Ähnlichkeit in der Liste angezeigt und können direkt als neue Playlist exportiert werden. Diese Playlist setzt sich zusammen aus dem Namen des Anfragetitels und dem Zusatz *similar songs*. Für die Suche ist es notwendig, dass vorher die Klassifikation ausgeführt worden ist, da diese den erforderlichen Merkmalsvektor im Tag *SMILARITY* speichert. Foo_Similarity_Search berechnet, um eine schnelle Suche zu ermöglichen keine Merkmale, sondern liest nur diese gespeicherten Vektoren aus und berechnet die Abstände.

Foo_search_dupli Die Suche nach Duplikaten über die Levenstein Distance in der Datenbank wird über das Hauptmenü COMPONENTS –> IMS –> FIND DUPLICATES gestartet. Im sich öffnenden Fenster kann werden nach dem Start die gefundenen Paare angezeigt. Abhängig von der Größe der Datenbank und der damit nötigen Vergleichsoperationen kann die Suche einige Zeit in Anspruch nehmen.

Foo_Genre_Training Über die Funktion zum erstellen eigener Trainingsmodelle für die Genre Klassifikation lassen sich Lieder den einzelnen Klassen zuordnen. Hierzu wir im Kontextmenü der Playlist das Menü IMS –> GENRE TRAINING bereitgestellt, mit den jeweiligen Einträgen für jede Klasse. So lassen sich den Listen Lieder zuordnen und nach dem Starten des Trainings werden die Merkmale berechnet und anschliessend der Klassifikator trainiert.

Es lassen die vorgebenen Genre neu trainieren, als auch sechs eigenen Genre festlegen. Das Trainingsmodell wird unter dem Dateinamen USER_GENRE.MODEL im Foobar Installationsordner gespeichert. Dieses Modell wird bei der Klassifikation bevorzugt gegenüber dem Standardmodell GENRE.MODEL verwendet. Die neu erstellten Namen der sechs Klassen werden in der Datei GENRE_TRAIN.INI gespeichert. Zur Wiederherstellung der Standard - Genre - Klassifikation bietet das Fenster von Foo_Genre_Train den Button RESET TO DEFAULT. Diese Funktion löscht sowohl das trainierte Modell, als auch die Namensgebung der Klassen. Somit wird wieder die Standard - Klassifikaion verwendet.

Foo_Rating Dieses Plugin bietet im Playlist Kontextmenü IMS –> RATE ein Wert zwischen 0 und 5 ausgewählt werden. Dieser Wert wird als Tag *Rating* gespeichert und bietet somit die Möglichkeit anhand dieser persönlichen Bewertung die Datenbank zu sortieren.

Foo_Copy Dieses Plugin kann sowohl im Hauptmenü über COMPONENTS –> IMS –> SEND PLAYLIST TO EXTERNAL DEVICE als auch im Kontextmenü der Playlist unter IMS –> SEND PLAYLIST TO EXTERNAL DEVICE aufgerufen werden. Im sich öffnenden Fenster wird die Größe der aktuellen Playlist angezeigt, und es lassen sich die gewünschte Zielgröße sowie der Zielordner angeben. Nach dem Starten des Kopiervorgangs werden alle Lieder der aktuellen Abspielliste kopiert, bis die angegebene Zielgröße erreicht ist.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AF04] J.-J. Aucouturier and Pachet F. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[AHH+03] Eric Allamanche, Jurgen Herre, Oliver Hellmuth, Thorsten Kastner, and Christian Ertel. A multiple feature model for musical similarity retrieval. In *Proceedings of the Fourth International Conference on Music Information Retrieval: ISMIR 2003*, pages 217–218, 2003.

[alMLIAD] amp11-lite MPEG1 Layer III Audio Dekoder. `http://www.piware.de/projects.shtml`.

[AML04] P. Ahrendt, A. Meng, and J. Larsen. Decision time horizon for music genre classification using short time features. In *EUSIPCO*, pages 1293–1296, Vienna, Austria, sep 2004.

[aMLIAD] amp11 MPEG1 Layer III Audio Dekoder. `http://www.ph.tum.de/~nbeisert/amp11.html`.

[AP02] Jean-Julien Aucouturier and Francois Pachet. Music similarity measures: What's the use? In *Proceedings of the 3rd International Symposium on Music Information Retrieval, IRCAM*, October 2002.

[AP03] J.J Aucouturier and F. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, 32(1), 2003.

[CF02] M. Cooper and J. Foote. Automatic music summarization via similarity analysis. In *Proc. Int. Symposium on Music Information Retrieval, ISMIR*, pages 81–85, Paris, France, 2002.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[DNS01] H. Deshpande, U. Nam, and R. Singh. Classification of music signals in the visual domain. In *Proceedings of the COST-G6 Conference on Digital Audio Effects*, Limerick, Ireland, 2001.

[fANNS] ANN: A Library for Approximate Nearest Neighbor Searching. `www.cs.umd.edu/~mount/ANN/`.

[FJ05] Matteo Frigo and Steven G. Johnson. The design and implementation of fftw3. In *Proceedings of the IEEE*, volume 93(2), pages 216–231, 2005.

[For] Foobar 2000 Development Forum. `http://www.hydrogenaudio.org/forums/index.php?showforum=34`.

[Hac00] Scot Hacker. *MP3: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2000.

[HCL01] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. *Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan*, 2001.

[JLZ02] D.-N. Jiang, L. Lu, and H.-J. Zhang. Music type classification by spectral contrast features. In *Proc. IEEE International Conference on Multimedia and Expo*, pages 113–116, Lausanne, Switzerland, August 2002.

[Kap02] André Kappes. Die Audiokodierung mp3. *Proseminar Redundanz, Fehlertoleranz und Kompression, Fakultät für Informatik, Universität Karlsruhe*, 2002.

[Log00] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, Plymouth, MA, USA, 2000.

[LOL03] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–289, New York, NY, USA, 2003. ACM Press.

[LR04] S. Leitich and A. Rauber. Information retrieval in digital libraries of music. In *Proceedings of the 6th Russian Conference on Digital Libraries (RCDL2004)*, Pushchino, Russia, 2004.

[LS01] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *Proc. IEEE International Conference on Multimedia and Expo, ICME*, Tokyo, Japan, August 2001.

[MA97] D. Mount and S. Arya. Ann: A library for approximate nearest neighbor searching. *CGC 2nd Annual Fall Workshop on Computational Geometry*, 1997.

[Mod04] Daniel Modrow. Ausgewählte Verfahren zur Messung der Ähnlichkeit von Symbolfolgen. *Diplomarbeit, Lehrstuhl für Mensch - Maschine - Kommunikation, Technische Universität München*, 2004.

[Nam01] Unjung Nam. Automatic music style classification: Towards the detection of perceptually similar music. *Music Department, Stanford University, USA*, May 8, 2001.

[Nav01] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1), March 2001.

[Pan95] Davis Pan. A tutorial to mpeg/audio compression. *IEEE Multimedia Journal*, Summer 1995.

[PG99] D. Perrott and R. O. Gjerdingen. Scanning the dial: An exploration of factors in the identification of musical style. *Research Notes. Department of Music, Northwestern University, Illinois, USA.*, 1999.

[Pla] Foobar 2000 Audio Player. `http://www.foobar2000.com`.

[Pye99] D. Pye. Content-based methods for the management of digital music. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages IV–2437–2440, Phoenix, AR, USA, 1999.

[Rig02a] G. Rigoll. Versuchsmanuskript zum Praktikum Mensch - Maschine - Kommunikation, 5. Auflage. *Lehrstuhl für Mensch - Maschine - Kommunikation, Technische Universität München*, Oktober 2002.

[Rig02b] Gerhard Rigoll. Kurzmanuskript zur Vorlesung Signaldarstellung. *Lehrstuhl für Mensch - Maschine - Kommunikation, Technische Universität München*, September 2002.

[Sea] Foobar2000 Database Search. `http://foosion.foobar2000.org/#dbsearch`.

[Shl94] Seymour Shlien. A guide to mpeg-1 audio standard. *In IEEE Transactions on Broadcasting*, 40(4):206–218, December 1994.

[Tag] Foobar2000 Wiki: Tagz. `http://wiki.hydrogenaudio.org/index.php?title=Foobar2000:Tagz`.

[TC00] G. Tzanetakis and P. Cook. Sound analysis using mpeg compressed audio. In *Proc. Int. Conf. on Audio, Speech and Signal Processing, ICASSP*, volume 10, Istanbul, Turkey, 2000.

[TC02] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. In *Proc. IEEE Transaction on Speech and Audio Processing*, pages 293–302, Lausanne, Switzerland, 2002.

[TEC01] G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals. In *Proc. Int. Symposium on Music Inform. Retriev. (ISMIR)*, pages 205–210, Bloomington, IN, USA, October 2001.

[tec01] Information technology. Coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbit/s - part 3: Audio, 1993-08-01.

[Tuta] Foobar 2000 Software Development Tutorial. `http://www.hydrogenaudio.org/forums/index.php?showtopic=42886`.

[Tutb] Playlist Tree Tutorial. `http://foobar.bowron.us/Tutorial/`.

[UI] Columns UI. `http://music.morbo.org/columns.php`.

[WF05] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2nd edition edition, San Francisco 2005.

[Wüs06] Ludwig Wüstner. Visuelle Lokalisation und Klassifikation mittels AdaBoost auf Basis von Haar - und Gabor - Wavelets. *Diplomarbeit, Lehrstuhl für Mensch - Maschine - Kommunikation, Technische Universität München*, 2006.

[XMS+03] Changsheng Xu, Namunu C Maddage, Xi Shao, Fang Cao, and Qi Tian. Musical genre classification using support vector machines. *In IEEE International Conference on Acoustics, Speech and Signal Processing*, 2003.

[ZF90] E. Zwicker and H. Fastl. *Psychoacoustics, Facts and Models.* Springer Verlag, Heidelberg, 1990.