JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Inhaltsbasierte Ähnlichkeitsmetriken zur Navigation in Musiksammlungen

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Ingenieur

in der Studienrichtung

Informatik

Eingereicht von:
Klaus Seyerlehner, 0155548

Angefertigt am:
Institut für Computational Perception

Betreuung:
Univ.-Prof. Dr. Gerhard Widmer
Univ.Ass. Dipl.-Ing. Markus Schedl

Linz, Oktober 2006

Kurzfassung

Diese Arbeit befasst sich mit aktuellen Methoden zur automatischen Bestimmung von Ähnlichkeiten zwischen Musikstücken. Der Fokus liegt dabei auf so genannten inhaltsbasierten Methoden, die versuchen das digitale Audiosignal zu analysieren und mit anderen Audiosignalen zu vergleichen. Insbesondere werden zwei *State-of-the-Art* Ansätze im Detail beschrieben und die Bedeutung derartiger Methoden für die Navigation in und Visualisierung von Musiksammlungen diskutiert. Anschließend wird gezeigt, wie ein neuartiges Navigations- und Visualisierungssystem basierend auf automatisch berechneten Ähnlichkeiten funktionieren könnte, und eine prototypische Realisierung eines solchen Systems vorgestellt. Ein Ausblick auf mögliche Verbesserungen und Perspektiven für weitere Forschungen runden die Arbeit ab.

Abstract

This thesis investigates state of the art methods to automatically derive music similarity relations. To be precise, this work focuses on so-called content-based similarity measures, which try to analyse and compare the digital audio signals of music titles. Two current approaches are presented in detail and the importance of such methods for navigation in and visualization of music collections is discussed. Subsequently, a new navigation and visualization system based on automatic similarity measures is proposed. The evaluation of a prototypical implementation of the introduced system and a brief overview of possible improvements and some ideas for future research complete this thesis.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

In den letzten Jahren hat sich der Musikhandel gewaltig verändert. Vor allem die Fortschritte im Bereich der digitalen Audiokompression und das Aufkommen von Breitbandinternetanschlüssen haben zu diesem Wandel beigetragen. Während die Musikbranche in den vergangenen Jahren der Informationstechnologie die Schuld für sinkende Verkaufszahlen gab, hat nun ein Umdenkprozess begonnen. Das Internet hat sich für die Musikbranche von einem Netzwerk für Raubkopierer hin zu einem neuen Vertriebskanal entwickelt, und nach und nach erkennt die Musikindustrie das durch die Informationstechnologie eröffnete Potential.

Neben der kontinuierlichen Verbesserung der *Digital Right Management Systeme* (DRM) zum Schutz der digitalen Inhalte werden auch die Musikdatenbanken ständig erweitert. So verfügen Online-Musikplattformen wie *iTunes*[1] oder *Musicload*[2] über Musiksammlungen mit über einer Million Titel. Während die Verwaltung der enormen Datenmengen eine gelöste technische Herausforderung ist, ist die Frage, wie ein Kunde die riesigen Sammlungen auch effektiv nutzen kann, noch ungeklärt. Abzusehen ist aber, dass ohne effiziente *Content Management*-Techniken der Kunde in der Flut des Musikangebots unterzugehen droht. Systeme, die den Kunden bei der Navigation und Orientierung innerhalb des verfügbaren Angebots unterstützen, gewinnen daher an Bedeutung. Das Ziel dieser Arbeit ist die Entwicklung eines Ansatzes für die Realisierung eines solchen Systems, sowie die

[1] http://www.apple.com/itunes
[2] http://www.musicload.de

prototypische Umsetzung des entwickelten Ansatzes zur Anwendung auf private Musiksammlungen.

1.2 Ähnlichkeit von Musikstücken

Der Erfolg von Online Musikplattformen als neuem Vertriebskanal für die Musikbranche hängt besonders von der Akzeptanz der Plattform durch den Kunden ab. Ganz entscheidend ist daher die Entwicklung benutzerfreundlicher Systeme zur Navigation und Visualisierung der angebotenen Musiksammlungen. Wie könnte man aber ein solches System realisieren? Dazu gibt es unterschiedliche Ansätze. Die Grundidee ist aber zumeist gleich. Kennt man die Ähnlichkeitsbeziehung zwischen Musiktiteln, kann man basierend auf dieser Ähnlichkeitsrelation die gesamte Musiksammlung strukturieren und auch innerhalb dieser navigieren. *MSN Music*[3] empfiehlt zum Beispiel beim Kauf eines Titels automatisch weitere ähnliche Titel, die dem Kunden gefallen könnten. Die Ähnlichkeit von Musiktiteln wird dabei durch die Analyse des Einkaufsverhaltens der Kunden ermittelt.

Ein Ähnlichkeitsmaß für Musikstücke kann als eine **Basistechnologie** für eine Vielzahl von Anwendungen betrachtet werden. Neben Navigation und Visualisierung sind auch *Automatic Playlist Generation* und *Automatic Genre Classification* Anwendungsbereiche dieser Technologie. Aus mathematischer Sicht handelt es sich bei einem Ähnlichkeitsmaß um eine Metrik, die formal durch eine Abbildung $d : X \times X \rightarrow R$ definiert ist, wobei die Menge X die Menge aller Musikstücke ist. Folgende Beziehungen müssen für eine Metrik gelten:

- $d(x, x) = 0$ (identische Musikstücke haben den Abstand 0);
- $d(x, y) = 0$ folgt $x = y$ (nicht identische Punkte haben nicht Abstand 0);
- $d(x, y) = d(y, x)$ (Symmetrie);
- $d(x, y) \leq d(x, z) + d(z, y)$ (Dreiecksungleichung);

[3] http://music.msn.com

Um ein Ähnlichkeitsmaß zu realisieren, müssen also die einzelnen Distanzen $d_{ij} = d(s_i, s_j)$ mit $s_i, s_j \in X$ zwischen allen Musikstücken bestimmt werden. Genau an dieser Stelle setzen die unterschiedlichen Verfahren an. Wie bereits erwähnt, kann man zur Bestimmung der Distanzen zum Beispiel das Kaufverhalten analysieren. Diese Arbeit befasst sich aber nur mit so genannten *inhaltsbasierten Ähnlichkeitsmetriken*.

Unter inhaltsbasierten Ähnlichkeitsmetriken versteht man Verfahren, die auf der Analyse des Audiosignals beruhen. Durch Methoden der digitalen Signalverarbeitung versucht man Merkmale, die Rhythmus, Klangfarbe, Melodie oder Tempo charakterisieren, aus dem digitalen Audiosignal zu extrahieren. Die zu Grunde liegende Annahme ist, dass die wesentlichen Aspekte, die der Mensch zur Beurteilung von Ähnlichkeiten zwischen Musikstücken ins Kalkül zieht, in dem digitalen Audiosignal enthalten sind. Der Vorteil von inhaltsbasierten Ähnlichkeitsmetriken liegt klar auf der Hand. Ist bekannt, wie man die Distanzen aus den Audiosignalen bestimmen kann, kann die Bestimmung aller Distanzen zwischen allen Musikstücken automatisch durch ein Computersystem erfolgen.

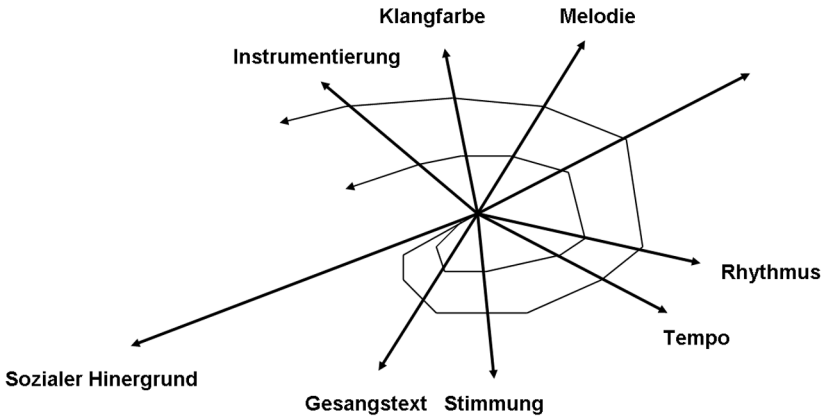


Abbildung 1.1: Dimensionen der Musikähnlichkeit (aus [24])

Ein grundlegendes Problem bei der Bestimmung von Distanzen zwischen Musiktiteln ist, dass die Ähnlichkeit von Musikstücken kein wohldefiniertes Konzept ist. Insbesondere ist die Wahrnehmung von Ähnlichkeiten sehr subjektiv und hat viele

Aspekte (siehe Abbildung 1.1): Tempo, Rhythmus, Harmonie, Melodie, Klangfarbe, sozialer Hintergrund, Gesangstext, sowie die Stimmung des Hörers haben Einfluss auf die Wahrnehmung. Dies wurde in [15] auch durch Experimente belegt. Dabei stellte sich heraus, dass die Teilnehmer an einem Klassifikationsexperiment selbst im Schnitt nur zu 76% über die Zugehörigkeit zu einem Genre einig sind. Damit ist klar, dass das Konzept der Ähnlichkeit von Musikstücken nicht präzise definiert und somit auch nicht präzise fassbar ist. Auch in [9] wird dargestellt, dass es im Bereich *Artist Similarity* keine absolute Wahrheit(*ground truth*) gibt, aber auch hier wird eine durchschnittliche Übereinstimmung von 85% festgestellt. Offensichtlich gibt es eine Art von gemeinsamem Ähnlichkeitsempfinden. Schreibt man die Abweichung vom gemeinsamen Ähnlichkeitsempfinden dem Individuum mit spezifischem kulturellem Hintergrund und aktueller Stimmungslage zu, dann bleibt die Hoffnung, dass genau jene Aspekte, die dem gemeinsamen Ähnlichkeitsempfinden zugeschrieben werden, im digitalen Audiosignal enthalten sind. Neben dieser These zeigen auch diverse Implementierungen von inhaltsbasierten Metriken, dass dieser Ansatz sinvoll ist.

Der nächste Abschnitt dieser Arbeit befasst sich mit den nötigen Grundlagen im Bereich Audiowahrnehmung und digitale Audioverarbeitung, um derartige inhaltsbasierte Metriken realisieren zu können.

2 Grundlagen

Dieses Kapitel stellt einige grundlegende Konzepte vor, die nötig sind, um die in Kapitel 3 vorgestellten inhaltsbasierten Ähnlichkeitsmaße beschreiben zu können. Zu Beginn wird auf die menschliche Wahrnehmung und deren mathematische Modellierung eingegangen, anschließend wird eine kurze Einführung verwendeter Konzepte der digitalen Signalverarbeitung gegeben.

2.1 Menschliche Wahrnehmung und psychoakustische Modelle

Eine ganz grundlegende Rolle in der Audioverarbeitung nimmt der Mensch als Empfänger von akustischen Botschaften — sei es nun Musik, Gesprochenes oder einfach nur Geräusche aus der Umgebung — ein. Das menschliche Gehör, das sich im Laufe der Evolution zu einem sensiblen Sinnesorgan entwickelt hat, bestimmt ganz grundlegend, welche akustischen Reize wir wahrnehmen und in welcher Form. Die menschliche Wahrnehmung und die mathematische Modellierung der menschlichen Wahrnehmung sind für die Realisierung von inhaltsbasierten Ähnlichkeitsmetriken von zentraler Bedeutung. Der folgende Abschnitt gibt einen kurzen Überblick über physikalische Grundlagen, die grundlegende Funktionsweise des Gehörs und die mathematische Modellierung der menschlichen Wahrnehmung.

2.1.1 Das Gehör

Aus physikalischer Sicht sind Musik und Gesprochenes nichts anderes als minimale Änderungen des Luftdrucks, die sich wellenförmig ausbreiten. Man spricht auch von Schallwellen. Das Ohr (siehe Abbildung 2.1) kann als Konverter, der diese Schallwellen in elektrische Signale zur Informationsübermittlung zwischen Nervenzellen umsetzt, betrachtet werden. Dabei besteht das Ohr grob aus drei Teilen (vgl. [7]):

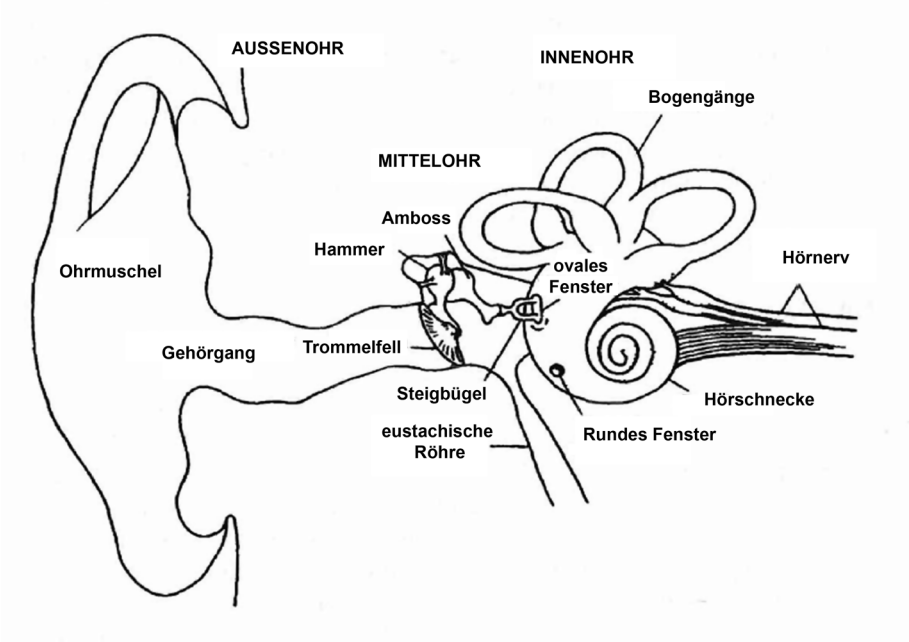


Abbildung 2.1: Das menschliche Ohr (aus [7])

1. Das Außenohr besteht aus der Ohrmuschel und dem Gehörgang, die gemeinsam für eine gute Übertragung der eintreffenden Schallwellen auf das Trommelfell sorgen. Das Trommelfell ist eine dünne, zweischichtige, elliptische Trennwand zwischen dem Außen- und dem Mittelohr. Eintreffende Schallwellen versetzen durch geringe Änderungen im Schalldruck und durch ihre Frequenz das Trommelfell in Schwingung.

2. Das Mittelohr besteht hauptsächlich aus den drei Gehörknöchelchen Hammer, Amboss und Steigbügel. Vom Außenohr werden die empfangenen Schwingungen in das Mittelohr übertragen, indem der Hammer ebenfalls in Schwingung versetzt wird. Der Hammer ist mit dem Amboss gelenkig verbunden und überträgt die Schwingungen auf den Amboss, der seinerseits die Schwingungen auf den Steigbügel, das dritte Glied in der Kette der Gehörknöchelchen im Mittelohr (oder Paukenhöhle) übermittelt. Die Platte des Steigbügels liegt im elastischen ovalen Fenster des Innenohres und gibt die Schwingungen und den Schalldruck an das Innenohr weiter. Durch die immer kleiner werdenden Kontaktflächen der Ohrknochenstruktur wird eine Verstärkung des Drucks auf das ovale Fenster des Innenohrs erreicht.

3. Das Innenohr besteht im Wesentlichen aus der Hörschnecke (*Cochlea*) und dem Gleichgewichtsorgan (*Labyrinth*). Erst in der Hörschnecke erfolgt die Umwandlung der von außen einwirkenden physikalischen Kräfte (Schallwellen) in biologische Signale (Nervenimpulse).

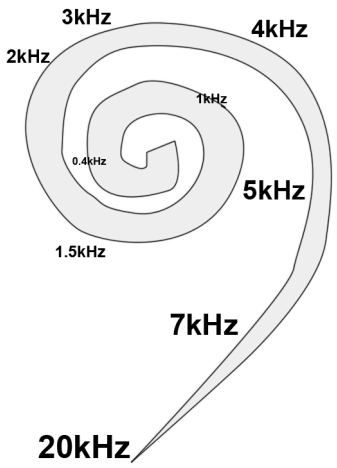


Abbildung 2.2: Die Hörschnecke

Dies ist natürlich der weitaus größte und schwierigste Teil der biologischen Sinnesarbeit. Spezielle Hörzellen führen diese Umwandlung durch und leiten die Nerven-

impulse über den Hörnerv ins Hirn. Die Hörschnecke an sich ist eine mit Flüssigkeit gefüllte Röhre, die durch das ovale Fenster in Schwingung versetzt wird. Die eigentlichen Hörzellen sitzen auf der Basilarmembran. Diese wird durch die schwingende Flüssigkeit aus der Ruhelage gebracht. Je höher die Schallfrequenz, desto näher liegt der Ort der maximalen Auslenkung (und mit ihr der Ort der Erregung der Sinneszellen des Cortiorgans) am Schneckeneingang (am ovalen Fenster). Dadurch werden bei hohen Frequenzen die Hörzellen nahe dem ovalen Fenster und bei tiefen Frequenzen die Nervenzellen in der Schneckenspitze gereizt. Folglich reagieren unterschiedliche Regionen der Hörschnecke und ihrer neuralen Rezeptoren auf unterschiedliche Frequenzen, wie in Abbildung 2.2 dargestellt.

Die Kenntnisse über den grundlegenden Hörvorgang erlauben es, einzelne Teile mathematisch zu modellieren, bzw. die Wahrnehmung des Menschen zu studieren. So beschäftigt sich die Psychoakustik (vgl. [38]) mit dem Zusammenhang zwischen den physikalischen Eigenschaften eines Schallsignals und den daraus resultierenden Hörempfindungen des Menschen.

2.1.2 Psychoakustische Modelle

Audiosignalverarbeitungsmethoden verwenden Modelle, die die Empfindsamkeit und Wahrnehmung des Gehörs repräsentieren. Dabei müssen die durch psychoakustische Modelle beschriebenen Zusammenhänge zwischen den physikalischen Reizen und den sinnespsychologischen Reaktionen durch Messungen eindeutig erfassbar sein. Ein psychoakustisches Modell kann dann einen solchen messbaren Zusammenhang beschreiben, indem der Zusammenhang zum Beispiel als Funktion modelliert wird. Beim Design eines psychoakustischen Modells wird das Gehör als signaltheoretisches Modell aufgefasst, das auf die Eingangsgröße Schallreiz mit der Ausgangsgröße Empfindung reagiert. Aufgrund des Vergleichs zwischen der genau definierten Eingangsgröße „Schallreiz“ und der gemessenen Ausgangsgröße „Empfindung“ kann das *Input/Output*-Verhalten des Gehörs studiert werden. Es liegen dann die Übertragungseigenschaften des Gehörs vor. Hieraus lassen sich wiederum allgemeine Modelle des Hörvorgangs ableiten. Aus praktischer Sicht sind

vor allem folgende Übertragungseigenschaften des menschlichen Gehörs von Interesse: die Lautstärkenwahrnehmung, die Tonhöhenwahrnehmung sowie die Frequenzauflösung und die zeitliche Auflösung. Konkrete Audiosignalverarbeitungsmethoden kombinieren einzelne Übertragungseigenschaften je nach Anwendungsgebiet zu einem Gesamtmodell. In Folge werden einige dieser Eigenschaften betrachtet.

2.1.3 Die Hörschwelle

Im direkten Zusammenhang mit der Hörschwelle steht auch die in der Akustik gebräuchliche Größe des Schallpegels. Während in der Physik der Schalldruck in *Pascal* $[N/m^2]$ gemessen wird, ist in der Akustik der Schallpegel, welcher das logarithmische Verhältnis zu Hörschwelle misst, gebräuchlich. Die Hörschwelle p_0 wurde mit $2 \times 10^{-5}\ N/m^2$ definiert. Die Umrechnung in *db-SPL* (Sound Pressure Level) erfolgt über folgende Formel:

$$SPL(p) = 20 \log_{10}(\frac{p}{p_0}) \tag{2.1}$$

Während die Hörschwelle p_0 der kleinste noch wahrnehmbare Schalldruck überhaupt ist, ist bekannt, dass im Allgemeinen die Hörschwelle von der Frequenz abhängt. Darum modelliert man die absolute Hörschwelle (*Absolute Threshold of Hearing*) als den minimalen Schallpegel in *db-SPL* (also relativ zu p_0) eines reinen Tons in einer störungsfreien Umgebung, der gerade noch von einem Menschen wahrgenommen werden kann. Die Abhängigkeit der Hörschwelle von der Frequenz f in *Hz* wird durch folgende Gleichung beschrieben:

$$ATH(f) = 3.64(f/1000)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}(f/1000)^4 \tag{2.2}$$

Die Gleichung, die die absolute Hörschwelle beschreibt (siehe Abbildung 2.3), wird auch als Terhardts Außen- und Mittelohrmodell bezeichnet. Am empfindlichsten ist das menschliche Gehör bei Frequenzen zwischen 3–4 *kHz*. Bei einem derartig niedrigen Schallpegel (ca. -5 *dB-SPL*) kann der Mensch andere Frequenzen nicht

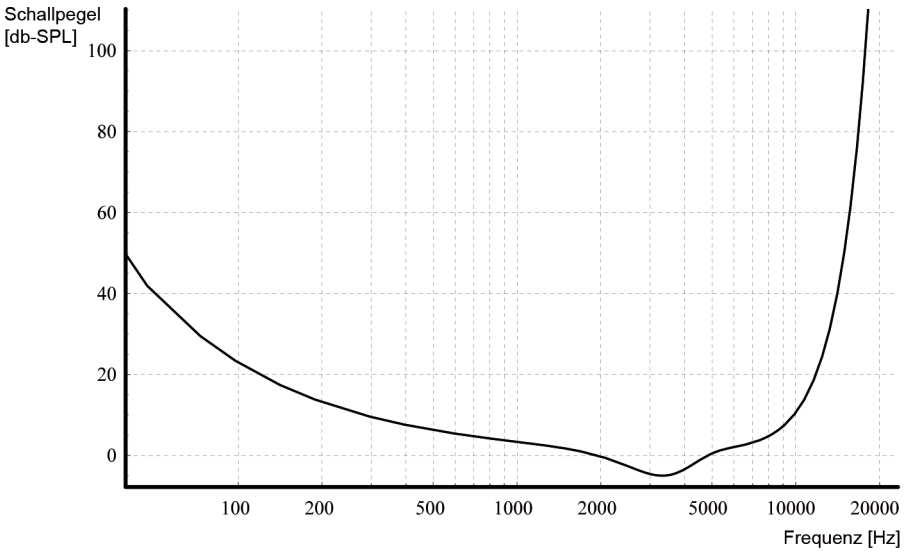


Abbildung 2.3: Die absolute Hörschwelle

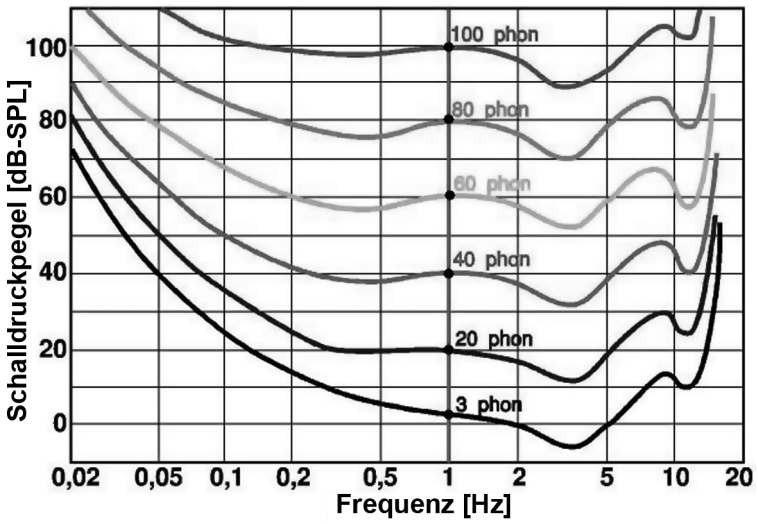
mehr wahrnehmen. Grundlegend gilt, dass zwei unterschiedliche Töne bei gleichem Schallpegel (*dB-SPL*) nicht gleich laut wahrgenommen werden. Diese Erkenntnis führt direkt zum Konzept der Lautheit, also der wahrgenommenen Lautstärke.

2.1.4 Lautheit

Das Lautstärkeempfinden des Menschen hängt stark von der Frequenz eines Tons ab. Experimentell wurde das mit Hilfe von Sinustönen unterschiedlicher Frequenz nachgewiesen, die trotz gleichen Schallpegels unterschiedlich laut wahrgenommen wurden. Aus diesen experimentell ermittelten Daten konnten Kurven gleicher Lautheit(vgl. [35]) ermittelt werden.

Die unterste Kurve in Abbildung 2.4 bei 3 *Phon* stellt die Hörschwelle dar. Als Referenzwert dient immer der in der Grafik dick hervorgehobene 1 *kHz* Ton. Alle anderen Töne, die gleich laut wie der entsprechende 1 *kHz* Ton wahrgenommen werden, liegen auf derselben Kurve. Die *Phon*-Skala ist also eine Lautheitsskala, die den Schallpegel des Referenztons angibt. Neben der *Phon*-Skala existiert auch

Abbildung 2.4: Kurven gleicher Lautstärke (aus [37])



noch eine zweite Lautheitsskala, die *Sone*-Skala. Die Lautheit in *Sone* kann nach Bladon und Lindblom [5] ausgehend vom Schallpegel in *dB-SPL* berechnet werden:

$$Sone(l_{dB-SPL}) = \begin{cases} 2^{(l-40)/10} & l \geq 40dB \\ (l/40)^{2.642} & l < 40dB \end{cases} \tag{2.3}$$

Die *Sone*-Skala ist von der Größe des Schallpegels unabhängiger und entspricht besser der menschlichen Intuition. So ist ein Ton mit 2 *Sone* doppelt so laut wie ein Ton mit 1 *Sone* Lautheit (vgl. [33]). Der mathematische Zusammenhang zwischen *Phon* und *Sone* ist wie folgt definiert:

$$Phon(l_{Sone}) = 40 + 10\log_2(l) \tag{2.4}$$

In der Praxis wird der korrekte Zusammenhang *F(Schalldruck, Frequenz)=Lautheit* jedoch nicht immer modelliert. Häufig wird einfach nur das Außen- und Mittelohrmodell nach Terhardt eingesetzt, das den Zusammenhang vereinfacht nur abhängig von der Frequenz darstellt.

2.1.5 Tonheit

Ähnlich wie bei der Lautstärke verhält es sich bei der Tonhöhe. Der Zusammenhang zwischen der Frequenz und der wahrgenommenen Tonhöhe (Tonheit) ist nicht linear.

$$Bark(f) = 13 \arctan(\frac{0.76f}{1000}) + 3.5 \arctan((\frac{f}{7500})^2) \tag{2.5}$$

Neben der *Bark*-Skala existiert noch die *Mel*-Skala, die im Wesentlichen den gleichen Zusammenhang modelliert.

$$Mel(f) = 2595(\log(1 + \frac{f}{700})/\log(10)) \tag{2.6}$$

Während die Tonhöhe niedriger Frequenzen fast linear wahrgenommen wird, beginnt ab etwa 500 *Hz* ein logarithmischer Zusammenhang. Man könnte auch sagen, dass sich die Frequenzauflösung des Gehörs bei hohen Tönen verschlechtert(vgl. Abbildung 2.5).

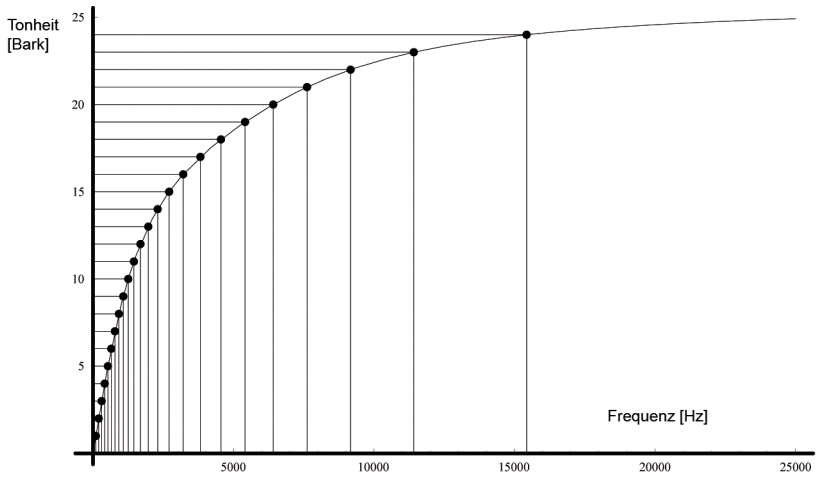


Abbildung 2.5: Die *Bark*-Skala und eingezeichnet die kritischen Bänder

2.1.6 Kritische Bänder

Das menschliche Gehör teilt den hörbaren Frequenzbereich in festgelegte Bereiche ein, die so genannten Frequenzgruppen (kritische Bänder). Die Bildung der Frequenzgruppen basiert auf der Umsetzung von Schall in Nervenimpulse im Innenohr. Es lassen sich 24 Frequenzgruppen feststellen. Lage und Breite der Frequenzgruppen legen den Schluss nahe, dass das menschliche Gehör die Basilarmembran des Innenohrs in ca. 24 gleichlange Abschnitte einteilt, für die jeweils die erzeugten Nervenimpulse gemeinsam ausgewertet werden. Dies entspricht aber genau der Definition der *Bark*-Skala. Ein *Bark* entspricht genau einem kritischen Band (siehe Abbildung 2.5). Natürlich lassen sich die Frequenzgruppen auch für die *Mel*-Skala definieren.

2.1.7 Maskierungseffekte

Die zeitliche Auflösung, sowie die Frequenzauflösung des menschlichen Gehörs sind beschränkt. Folglich ist es für die Modellierung der menschlichen Wahrnehmung wichtig auch diese Limitierungen einfließen zu lassen. Die eingeschränkte Wahrnehmungsfähigkeit des menschlichen Gehörs macht sich durch *Maskierungseffekte* bemerkbar.

Unter einem Maskierungseffekt versteht man, dass ein Signal durch das Auftreten eines weiteren lauteren Signals für den Menschen unhörbar wird. Ausschlaggebend dafür, ob ein Signal ein anderes verdeckt, sind die zeitliche Nähe und die Ähnlichkeit der Frequenz. Man unterscheidet daher *spectral masking* und *temporal masking*. Grundlegend gilt, je näher zwei Signale im Zeit- und Frequenzbereich liegen, umso stärker ist der Maskierungseffekt. Eine detaillierte Betrachtung von Maskierungseffekten gibt die Arbeit [20] von Painter und Spanias.

Für die hier vorgestellten Verfahren sind vor allem spektrale Maskierungseffekte interessant. Spektrale Maskierung ist eng mit dem Konzept der kritischen Bänder verbunden. Innerhalb eines Bandes maskiert ein Ton andere Töne fast gleichförmig. Der Maskierungseffekt ist aber nicht nur auf ein Band beschränkt, sondern kann

sich auch auf andere Bänder ausdehnen. Wie stark der Einfluss einer maskierenden Frequenz auf andere Frequenzen ist, wird durch eine *spreading function* modelliert. Eine *spreading function* liefert je einen Maskierungsschwellwert für alle benachbarten Frequenzen. Schroeder et al. [29] schlagen folgende gebräuchliche *spreading function* vor:

$$SF(\triangle z) = 15.81 + 7.5(\triangle z + 0.474) - 17.5\sqrt{1 + (\triangle z + 0.474)^2} \qquad (2.7)$$

Dabei ist $\triangle z = z_j - z_i$ die Differenz der Frequenzen z_i und z_j in *Bark*. Sind mehrere Töne gemeinsam aktiv, werden die einzelnen Schwellwertfunktionen meist additiv kombiniert und bilden eine gemeinsame Schwellwertfunktion. Nur jene Frequenzen, die im Originalsignal stärker als es die gemeinsame Schwellwertfunktion angibt vertreten sind, sind für den Menschen hörbar. Maskierungseffekte sind für die Modellierung des menschlichen Gehörs besonders im Bereich der Audiokompression von zentraler Bedeutung.

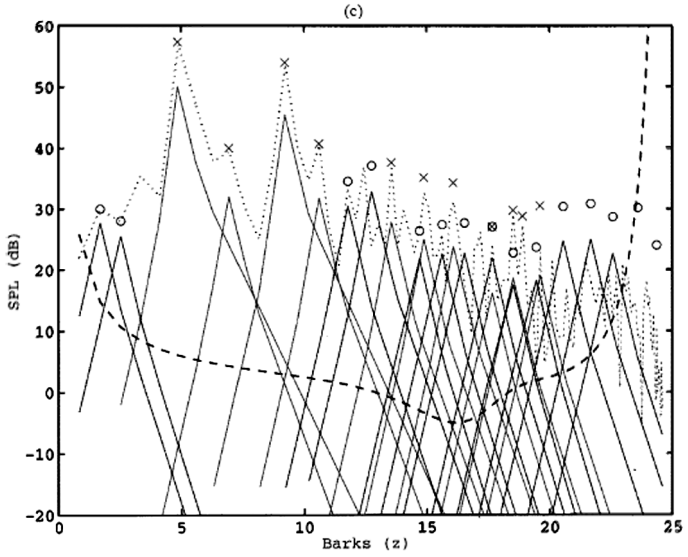


Abbildung 2.6: *spread functions* der maskierenden Frequenzen (aus [20])

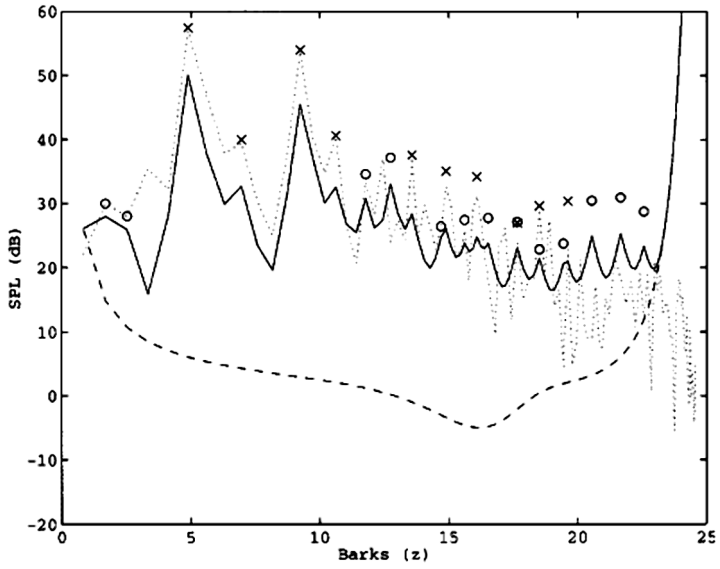


Abbildung 2.7: Resultierender gemeinsamer Maskierungsschwellwert (aus [20])

Abbildung 2.6 zeigt die *spreading function* für mehrere maskierende Töne und Geräusche. Abbildung 2.7 zeigt den gemeinsamen Maskierungsschwellwert.

Damit deckt dieser Abschnitt von der grundlegenden physikalischen und biologischen Seite und den gebräuchlichen Einheiten bis hin zu den wesentlichen psychoakustischen Eigenschaften des Gehörs das nötige Grundwissen ab. Der nächste Abschnitt befasst sich mit den nötigen Grundlagen aus dem Bereich der digitalen Signalverarbeitung.

2.2 Digitale Signalverarbeitung

Dieser Abschnitt gibt ein kurze Einführung in wichtige Konzepte der digitalen Signalverarbeitung, auf denen die in Kapitel 3 vorgestellten inhaltsbasierten Ähnlichkeitsmaße aufbauen. Diese Einführung vermittelt wirklich nur ganz fundamentale

Grundlagen und baut auf dem Standardwerk von Kent Steigliz [34] und dem Werk von Oppenheim und Schafer [19] auf. Für eine detailliertere Einführung in den Bereich der digitalen Signalverarbeitung sei an dieser Stelle auch auf [36, 32] verwiesen.

2.2.1 Abtasten und Quantisieren

Wie im Abschnitt 2.1.1 schon genau erläutert, ist Schall nichts anderes als kleine Änderungen des Luftdrucks. Diese Änderungen werden durch die Membran eines Mikrofons in ein elektrisches Signal $s(t)$ umgewandelt, wobei $s(t)$ die Auslenkung der Membran abhängig von der Zeit darstellt. Das Signal $s(t)$ ist ein analoges Signal, da sowohl die Zeit t als auch der Signalwert $s(t)$ reeller Natur sind. Computer können aber nur digitale Werte verarbeiten. Daher muss das analoge Signal $s(t)$ durch einen A/D Wandler in ein digitales Signal konvertiert werden. Dieser Umwandlungsprozess beinhaltet zwei Schritte: das **Abtasten** und das **Quantisieren**.

Abtasten

Zuerst muss das analoge Signal $s(t)$ in eine zeit-diskrete Form gebracht werden. Dies geschieht, indem in periodischen Intervallen der Länge T_s das Signal abgetastet wird (siehe Abbildung 2.8). Die Beziehung zwischen dem analogen, zeitkontinuierlichen Signal $s(t)$ mit $t \in \mathcal{R}$ und dem zeit-diskreten Signal $x(n)$ mit $n \in \mathcal{N}$, ist über die *Abtastperiode* T_s festgelegt.

$$x(n) = s(nT_s) \tag{2.8}$$

Die *Abtastrate* in *Hz* für das zeit-diskrete Signal ist durch

$$f_s = 1/T_s \tag{2.9}$$

definiert und bildet die *Grundfrequenz* f_s des digitalen Signals. Von zentraler Bedeutung, um ein analoges Signal korrekt durch ein digitales Signal darstellen zu können, ist, dass die Abtastfrequenz mindestens das Doppelte der höchsten im analogen Signal vorkommenden Frequenz sein muss (*Abtasttheorem nach Nyquist*). Alle Frequenzen oberhalb und unterhalb der Nyquist Frequenz $-f_s/2$ und $+f_s/2$ werden fälschlich wieder in dieses Basisband als Alias-Frequenzen abgebildet. Um diesen Aliaseffekt zu vermeiden, ist es ratsam, durch einen Bandpassfilter das analoge Eingangssignal auf das Basisband zu limitieren.

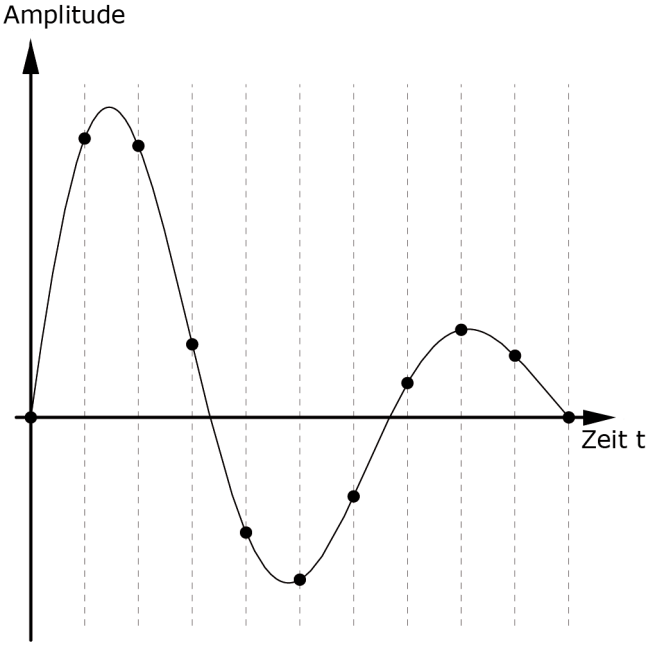


Abbildung 2.8: Konvertierung eines analogen in ein zeit-diskretes Signal (*Abtasten)*

Quantisieren

Im Anschluss müssen die Signalwerte des resultierenden zeit-diskreten Signals gerundet werden, sodass auch diese in binärer Form dargestellt werden können. Meist

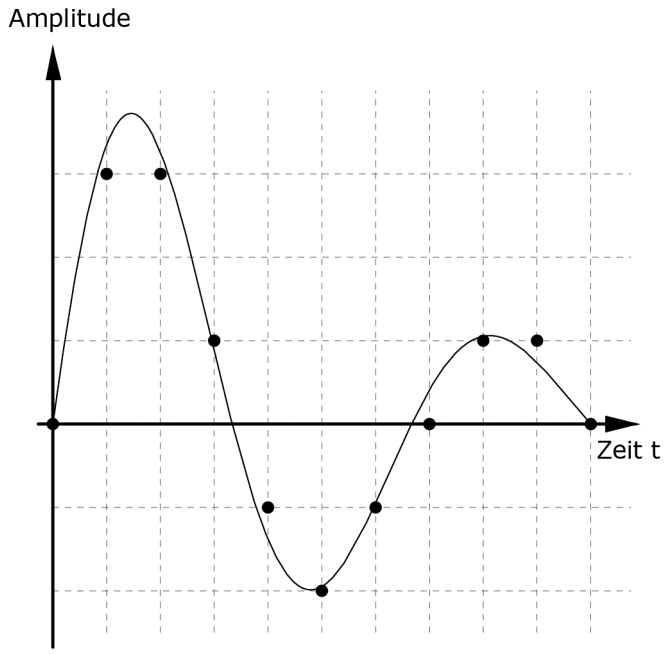


Abbildung 2.9: Konvertierung eines zeit-diskreten in ein digitales Signal (*Quantisiern*)

wird für die Darstellung eines Signalwertes, auch *Sample* genannt, ein *16 Bit Integer* verwendet. Die endliche Anzahl an Quantisierungsstufen, die mit *b Bits* darstellbar sind, begrenzen den repräsentierbaren Amplitudenbereich und bestimmen den *Quantisierungsfehler*, der durch das Runden entsteht (siehe Abbildung 2.9). Der Quantisierungsfehler wird als Verhältnis des Signalpegels zum Rauschpegel, der durch die Quantisierung entstandenen ist, angegeben. Als grobe Abschätzung für diesen *signal to noise ratio* (SNR) gilt, dass sich der SNR um 6 *dB* pro *Bit* erhöht.

Ein digitales Signal kann über einen D/A Wandler wieder in ein analoges Signal zurück gewandelt und danach über einen Lautsprecher ausgegeben werden. Dabei bietet das digitale Signal nicht nur den Vorteil, dass es verlustfrei gespeichert und vervielfältigt werden kann, sondern es kann auch durch einen Computer verarbeitet werden. Viele Methoden im Bereich der digitalen Audioverarbeitung werden

nicht im Zeitbereich, sondern im Frequenzbereich durchgeführt. Der Übergang vom Zeitbereich in den Frequenzbereich wird durch die *Fourier-Transformation* bewerkstelligt, auf die nun näher eingegangen wird.

2.2.2 Fourier-Transformation

Schon 1807 zeigte *Jean Baptiste Joseph Fourier*, dass jedes kontinuierliche periodische Signal als unendliche Summe von Sinus- und Cosinuswellen dargestellt werden kann. Abgeleitet von der Fourier-Transformation für kontinuierliche Funktionen existiert auch die diskrete Version, die als *diskrete Fourier-Transformation*, oder kurz DFT, bekannt ist. Die DFT ist eine lineare Abbildung und kann als Basiswechsel im Vektorraum betrachtet werden. Mit den diskreten Versionen der Sinus- und Cosinuswellen als Basis gelingt der Übergang vom Zeitbereich in den Frequenzbereich. Betrachtet man das digitale Signal $x(n)$, das die Samplesequenz $x_0, x_1, x_2, ..., x_{N-1}$ repräsentiert, dann erhält man das *Frequenzspektrum* $X(k)$ gemäß der diskreten Fourier-Transformation:

$$X(k) = \sum_{n=0}^{N-1} e^{-j\frac{2\pi}{N}nk} x(n) \tag{2.10}$$

Das Ergebnis der diskreten Fourier-Transformation sind die N komplexen *Fourier-Koeffizienten*. Der umgekehrte Weg vom Spektrum in den Zeitbereich wird durch die *inverse diskrete Fourier-Transformation* (iDFT) ermöglicht.

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}nk} X(k) \tag{2.11}$$

Hinsichtlich der Laufzeitkomplexität sind die DFT und die iDFT, wenn man sie direkt berechnet, wie jede allgemeine isomorphe lineare Abbildung, von der Komplexität $O(N^2)$. Wendet man jedoch das **divide and conquer** Prinzip an und nützt dabei die Periodizität der Sinuswellen aus, so kommt man zu einem effizienten Algorithmus zur Berechnung der DFT. Dieser schnelle Algorithmus, der

nur für digitale Signale der Länge 2^n ausgeführt werden kann, wird als *Fast-Fourier-Transformation* (FFT) bezeichnet und hat eine Laufzeitkomplexität von $O(N \log N)$. Auch für die iDFT gibt es mit der iFFT eine effiziente Realisierung.

Kurzzeit-Fourier-Transformation

Eine wesentlicher Nachteil der DFT ist, dass sie keine Informationen über den Zeitpunkt, an dem eine Frequenzkomponente auftritt, gibt. Das ist kein Problem bei stationären Signalen, lässt aber Raum für Verbesserungen, wenn es sich um nicht-stationäre Signale handelt. Signale, wie Sprache, Musik oder Bilder sind aus nicht-stationären (in Zeit und Raum variierenden) Ereignissen zusammengesetzt. So setzen Sprachsignale sich zum Beispiel aus Folgen von kurz andauernden Lauten zusammen, die man *Phoneme* nennt. Für die Durchführung einer Frequenzanalyse ist daher eine entsprechend hohe zeitliche und spektrale Auflösung wünschenswert, sodass man die spektralen Eigenschaften jedes elementaren Ereignisses im Eingangssignal unterscheiden kann. Um auch eine zeitliche Auflösung zu erhalten, wird das digitale Signal meist in kleine, sich überlappende Zeitfenster, auch *Frames* genannt, unterteilt (siehe Abbildung 2.10). Anschließend wird für jeden dieser Frames die DFT berechnet. Speziell für Sprache und Musik gilt, dass kurze Signalstücke (10 - 50 *ms*) annähernd stationär sind, wodurch streng genommen die Anwendung der DFT erst zulässig wird.

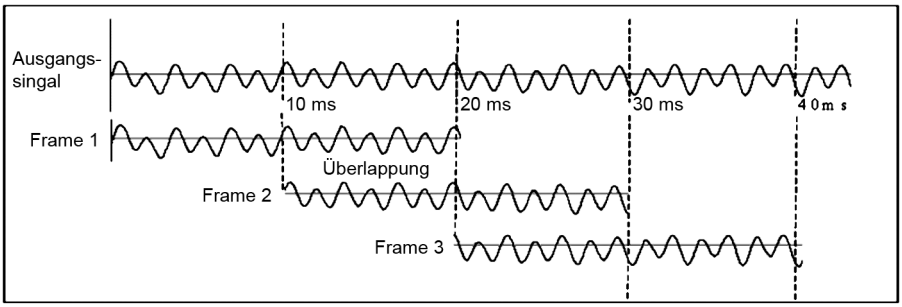


Abbildung 2.10: Einteilung in Frames durch eine rechteckige Fensterfunktion

Da man für jeden Frame ein Spektrum erhält, kann man die Veränderung der spektralen Zusammensetzung eines Signals mit der Zeit beobachten. Dieses Verfahren

wird als *Kurzzeit-Fourier-Transformation*(STFT) bezeichnet.

Bei der Anwendung der STFT gibt es einen grundlegenden *Trade-Off* zwischen der Auflösung im Zeit- und im Frequenzbereich. Die DFT nimmt als Eingabe ein Fenster von K Samples $[x(0), x(1), ..., x(K-1)]$ aus dem Zeitbereich mit einer Dauer von $\triangle T = KT_s$ und liefert N Koeffizienten $[X(0), X(1), ..., X(K-1)]$, die äquidistant zwischen 0 Hz und der Grundfrequenz f_s Hz verteilt sind. Damit ist die Frequenzauflösung $\triangle f$, der Frequenzabstand zwischen zwei Koeffizienten, gegeben durch die Gesamtbandbreite aufgeteilt auf die Anzahl der Koeffizienten $\triangle f = \frac{f_s - 0}{K}$. Aus dem Zusammenhang der Abtastfrequenz mit der Abtastperiode $f_s = \frac{1}{T_s}$ ergibt sich der indirekt proportionale Zusammenhang zwischen spektraler und zeitlicher Auflösung:

$$\triangle f = \frac{f_s - 0}{K} = \frac{f_s}{K} = \frac{1}{KT_s} = \frac{1}{\triangle T} \tag{2.12}$$

Aufgrund des *Unschärfeprinzips* können also nicht beide, die Zeit- und die Frequenzauflösung, gleichzeitig erhöht werden. Trotz dieses kleinen Nachteils ist die STFT in der digitalen Audioverarbeitung weit verbreitet. Die zur STFT zugehörige Visualisierung ist das *Spektrogramm* und stellt für jeden Frame das *Betragsspektrum* dar.

Betrags- und Phasenspektrum

Die DFT liefert als Resultat N komplexe Fourier-Koeffizienten. Anhand dieser Koeffizienten lassen sich für jede Frequenz der Betrag sowie die Phase bestimmen, und man erhält das *Betragsspektrum*, oder auch *Amplitudenspektrum* $M(k)$, und das *Phasenspektrum* $\phi(k)$ für jede Frequenz k wie folgt:

$$M(k) = |X(k)| \tag{2.13}$$

$$\phi(k) = \angle X(k) = \arctan \left| \frac{reX(k)}{imX(k)} \right| \tag{2.14}$$

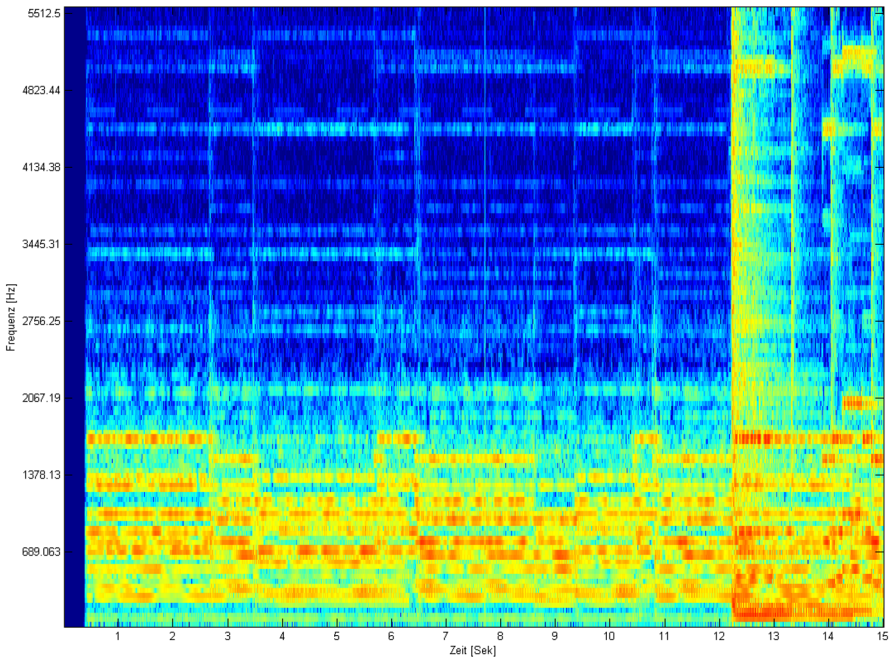


Abbildung 2.11: Spektrogramm hoher Zeitauflösung (256 Punkt FFT).

Das *Betragsspektrum* gibt an, wie stark eine Frequenz im Signal vertreten ist. Für die Darstellung der STFT eines Signals als Spektrogramm wird entweder das Betragsspektrum jedes Frames oder das *quadrierte Betragsspektrum* verwendet.

$$|X(f)|^2 = X(f)X^*(f) \tag{2.15}$$

Man beachte, dass $X^*(f)$ in Formel (2.15) die zu $X(f)$ konjugiert komplexe Zahl bezeichnet. Die Abbildungen 2.12 und 2.11 zeigen zwei Spektrogramme der ersten 15 Sekunden von *Dire Straits - Walk of Life*. In Abbildung 2.11 wurden kleine Fenster, je 256 Samples, verwendet, und man erhält eine gute zeitliche Auflösung. Für Abbildung 2.12 wurden — zur Demonstration des Unschärfeprinzips — Fenster mit je 2048 Samples gewählt, wodurch die Frequenzauflösung deutlich besser ist bei gleichzeitig schlechterer Zeitauflösung.

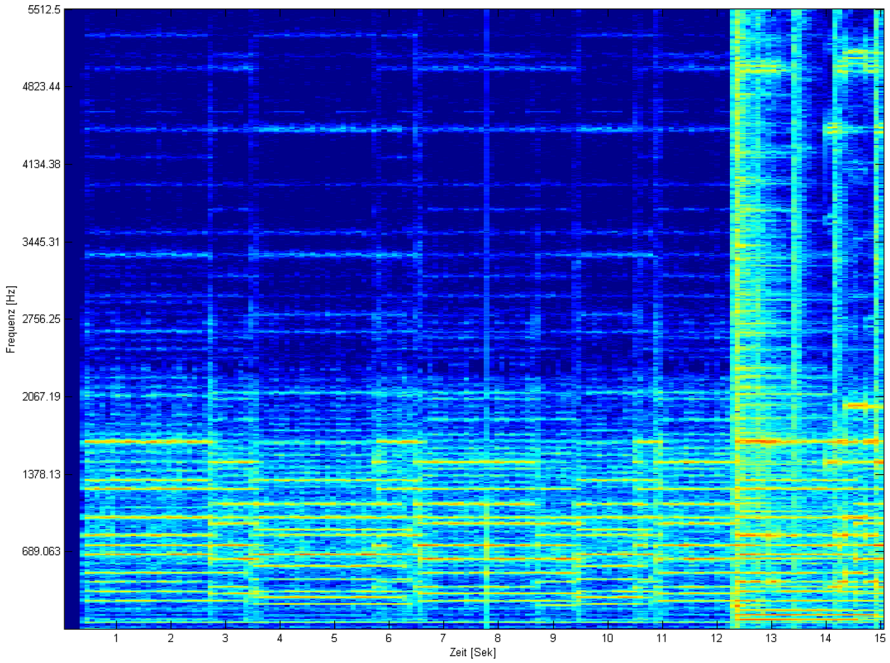


Abbildung 2.12: Spektrogramm mit hoher Frequenzauflösung (2048 Punkt FFT).

2.2.3 Fensterfunktionen

Bei der STFT wird das digitale Audiosignal in kurze *Fenster* oder *Frames* unterteilt (siehe Abbildung 2.10), wobei meistens die einzelnen Fenster einander überlappen. Das „Ausschneiden“ der Fenster aus dem digitalen Audiosignal wird durch eine Fensterfunktion realisiert. Die Fensterfunktion $w(n)$ ist innerhalb des Frames größer 0 und für alle anderen Teile des digitalen Signals gleich 0.

$$x_w(n) = x(n)w(n) \tag{2.16}$$

Nach der Fensterung wird dann bei der STFT für das gefensterte Signal $x_w(n)$ die DFT berechnet, wobei die Wahl der Fensterfunktion das Ergebnis der DFT erheblich beeinflussen kann. Der Einfluss der Fensterfunktion auf das Ergebnis der DFT lässt sich durch das *Fenstertheorem* der Fourier-Transformation für zeitkontinuierliche Signale erklären und wirkt sich direkt auf die DFT aus, da die

DFT einen Spezialfall der Fourier-Transformation darstellt. Das Fenstertheorem lässt sich nach [19] für das zeit-kontinuierliche Signal $x(t)$ und die zugehörige kontinuierliche Fensterfunktion $w(t)$ wie folgt definieren:

Seien $X(e^{j\omega})$ und $X(e^{j\omega})$ die zu $x(t)$ und $w(t)$ gehörigen Fourier-Transformierten

$$x(t) \overset{\mathcal{F}}{\leftrightarrow} X(e^{j\omega}) \tag{2.17}$$

$$w(t) \overset{\mathcal{F}}{\leftrightarrow} X(e^{j\omega}) \tag{2.18}$$

und gilt außerdem

$$x_w(t) = x(t)w(t) \tag{2.19}$$

dann folgt hieraus

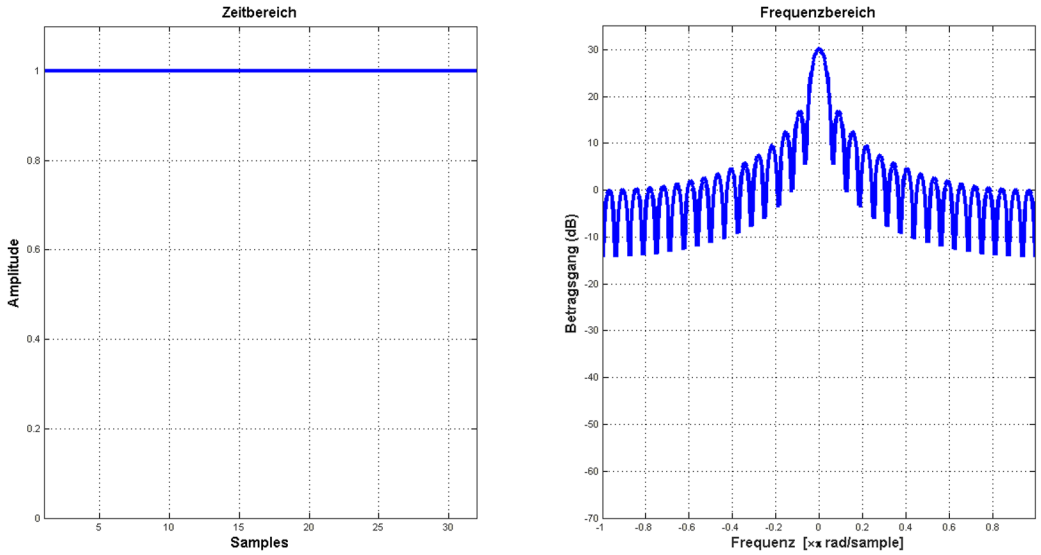
$$X_w(e^{j\omega}) = \frac{1}{2\pi} \int_{\pi}^{\pi} X(e^{j\theta})W(e^{j\theta})d\theta \tag{2.20}$$

Die Gleichung (2.20) stellt eine periodische Faltung - d.h. eine Faltung von zwei periodischen Funktionen - dar, wobei sich die Integrationsgrenzen nur über eine Periode erstrecken. Die Fensterung im Zeitbereich entspricht einer Faltung der Fourier-Transformierten des Eingangssignals mit der Fourier-Transformierten der Fensterfunktion.

$$X_w(e^{j\omega}) = X(e^{j\omega}) * W(e^{j\omega}) \tag{2.21}$$

Für die Analyse von Signalen durch die DFT bedeutet das, dass Impulse im Spektrum durch die Faltung mit der Fourier-Transformierten der Fensterfunktion verwischt bzw. verbreitert werden und somit deren exakte Frequenzen nicht mehr so genau zu bestimmen sind. Die Art der Verschmierung im Spektrum wird durch $W(e^{j\omega})$ bestimmt, und dem zufolge ist die Wahl einer geeigneten Fensterfunktion von zentraler Bedeutung.

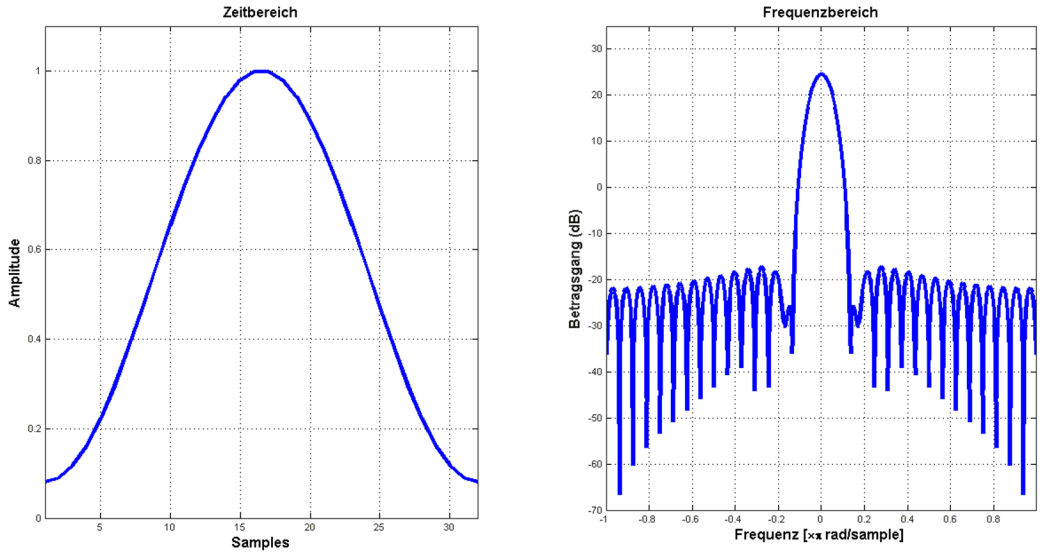
Die intuitivste Fensterfunktion ist die Rechtecksfunktion (2.22). Vergleicht man aber die Rechtecksfunktion mit anderen Fensterfunktionen, so stellt sich schnell heraus, dass die Rechtecksfunktion keine gute Wahl ist. Allgemein gilt, je näher die transformierte Fensterfunktion, genau genommen der Betragsgang $|W(e^{j\omega})|$, an einem idealen Puls ist (enger Hauptkolben, minimale Nebenkolben), desto näher ist das Ergebnis der DFT am originalen Signalspektrum. Eine wesentlich bessere Wahl für eine Fensterfunktion ist zum Beispiel die *Hamming* Funktion (2.23).



$$w(n) = \begin{cases} 1 & n \geq 0 \wedge n < N \\ 0 & \text{sonst} \end{cases} \tag{2.22}$$

Abbildung 2.13: Rechteckiges Fenster für eine Fensterlänge von 64 Samples.

Wie man in den Abbildungen 2.13 und 2.14 gut erkennen kann, hebt sich bei der Hamming Fensterfunktion der etwas breitere Hauptkolben bei 24 *dB* von den Nebenkolben, die bei etwa -20 *dB* liegen, deutlich ab, während bei der rechteckigen Fensterfunktion die Nebenkolben mit 0 bis 15 *dB* starke Beiträge liefern und so das Spektrum verfälschen. Leider besitzt die Hamming-Funktion einen breiteren Hauptkolben, wodurch Frequenzen, die nahe beieinander liegen, nicht mehr voneinander unterschieden werden können. Die wesentliche Erkenntnis aus diesem Abschnitt über Fensterfunktionen ist, dass mit der STFT aufgrund der Fensterung



$$w(n) = 0.53836 - 0.46164 \cos(\frac{2\pi n}{N-1}) \tag{2.23}$$

Abbildung 2.14: Hamming Fenster für eine Fensterlänge von 64 Samples.

das tatsächliche Spektrum nicht korrekt bestimmt werden kann, aber durch eine geeignete Wahl der Fensterfunktion bestimmte Effekte der Fensterung reduziert werden können.

Aufbauend auf den vorgestellten Grundlagen geht das nächste Kapitel nun auf inhaltsbasierte Ansätze zur automatischen Bestimmung von Ähnlichkeiten ein und stellt zwei aktuelle Ansätze vor.

3 Inhaltsbasierte Ähnlichkeitsmetriken

Ein gängiger Ansatz, um Ähnlichkeiten zwischen Musiktiteln zu bestimmen, ist der Versuch, direkt digitale Audiosignale zu analysieren und Eigenschaften der Signale zum Vergleich heranzuziehen. Derartige Verfahren nennt man inhaltsbasierte Ähnlichkeitsmetriken (*content based similarity measures*), und man bezeichnet den Vorgang der Analyse als *Feature Extraction Process*, und die durch die Analyse erhaltenen charakteristischen Eigenschaften als *Features*.

3.1 Low-Level Audio Features

Viele unterschiedliche Audio Features wurden von der *Speech Recognition*, der *Instrument Recognition* und *Music Information Retrieval* Gemeinschaft vorgeschlagen, um Audiosignale zu beschreiben. Nach [16] und [26] lassen sich die Merkmale grob in folgende Kategorien einteilen:

- **Temporal Features**
  Eigenschaften, die aus dem gesamten Signal extrahiert werden, ohne das Signal in Frames zu unterteilen;
  (z.B.: *log attack time*, *temporal decrease*, *effective duration*)
- **Energy Features**
  Eigenschaften, die die Energie des Signals beschreiben;
  (z.B.: *total energy*, *harmonic energy*, *noise part energy*)

- **Spectral Features**
  Eigenschaften, die spektrale Information über das Signal liefern;
  (z.B.: *spectral centroid*, *spectral spread*, *spectral skewness*)

- **Harmonic Features**
  Eigenschaften, die aus der sinusförmigen, harmonischen Modellierung des Signals berechnet werden;
  (z.B.: *fundamental frequency*, *inharmonicity*, *odd-to-even ratio*)

- **Perceptual Features**
  Eigenschaften, die ein Modell des menschlichen Gehörs zu Berechnung verwenden;
  (z.B.: *mel frequency cepstral coefficients*, *loudness*, *sharpness*)

Spezifisch für die Musikverarbeitung ist, dass diese *Low-Level Features* Information über die Zeit enthalten müssen, da sich Musiksignale mit fortlaufender Zeit stark ändern (siehe Abschnitt 2.2.2). Damit sind hier nur jene Low-Level Features von Bedeutung, die sinnvoll auf den Frames eines Musiksignals definiert sind, und man erhält durch die Merkmalsextraktion pro Frame genau ein Low-Level Feature, welches die Signaleigenschaften zu einem bestimmten Zeitpunkt beschreibt.

3.2 Allgemeine Vorgehensweise

Der Weg ausgehend von den Low-Level Features hin zu einem Distanzmaß wird in dieser Arbeit angelehnt an die Vorgehensweise im Bereich *Computer Vision* als strukturierter Prozess betrachtet. Der Prozess gliedert sich in drei Schritte: **Merkmalsextraktion**, **Modellgenerierung** und **Distanzberechnung**.

Die **Merkmalsextraktion** liefert als Resultat eine Menge an Low-Level Features. Aufbauend auf den extrahierten Merkmalen wird in der **Modellgenerieungsphase** ein Gesamtmodell für einen Musiktitel erstellt. Die Repräsentation eines Musikstücks durch ein Gesamtmodell ist ein wichtiger Schritt hin zur Ähnlichkeitsbestimmung. Ausgehend von den einzelnen Bewertungen der Signaleigenschaften

zu unterschiedlichen Zeitpunkten durch ein Low-Level Feature, muss hin zu einem Gesamtmodell abstrahiert werden. Erst basierend auf dem Gesamtmodell können dann im Rahmen der **Distanzberechnung** sinnvoll Distanzen zwischen den Modellen und damit Distanzen zwischen Musiktiteln bestimmt werden.

Als günstig hat sich bei der Modellbildung erwiesen, sich auf einen Aspekt musikalischer Ähnlichkeit (siehe Abbildung 1.1), wie Rhythmus, Tempo, Melodie oder Klangfarbe zu konzentrieren. So fokussieren die in Folge vorgestellten *Fluctuation Patterns* auf den Rhythmus, während das zweite *State-of-the-Art* Verfahren, *Timbre Distribution*, versucht die klangfarblichen Eigenschaften eines Titels zusammen zu fassen.

3.3 Fluctuation Patterns

Wie man dem Namen bereits entnehmen kann, versucht man mit Hilfe der *Fluctuation Patterns* (FP) die rhythmische Struktur eines Musiktitels in einem Modell zu erfassen.

3.3.1 Merkmalsextraktion

Der komplexe Extraktionsprozess für die FPs nach Elias Pampalk [21] kann grob in drei Verarbeitungsstufen untergliedert werden: **Vorverarbeitung**, **Psychoakustische Verarbeitung** und **Extraktion der rhythmischen Struktur**. Einen Überblick des Extraktionsprozesses gibt Abbildung 3.1, in der die unterschiedlichen Verarbeitungsstufen farblich hervorgehoben sind.

Vorverarbeitung

Die Vorverarbeitung führt im Wesentlichen eine Reduktion der Ausgangsdaten durch. Da die meisten Musikstücke in Form von komprimierten Audiodateien (z.B.:

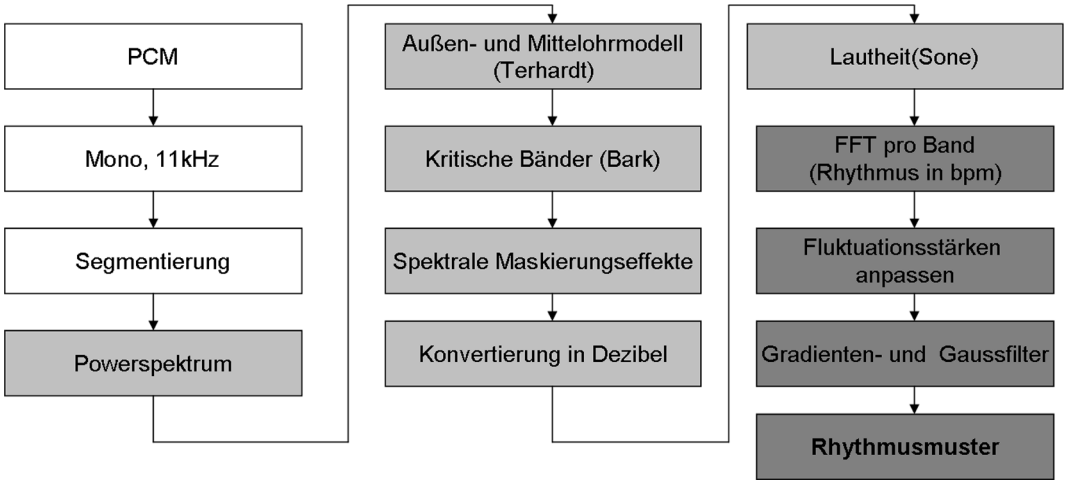


Abbildung 3.1: Überblick *Fluctuation Patterns*

mp3-Dateien) vorliegen, ist als erstes eine Decodierung nötig, die das digitale Audiosignal aus dem komprimierten *Bitstream* rekonstruiert **(a)**. Danach findet die eigentliche Reduktion statt, indem der Audiostream (meist **44 *kHz* stereo**) in einen **11 *kHz* mono** Stream konvertiert wird. Anschließend wird das Audiosignal noch in 6 Sekunden Segmente unterteilt, wobei nur jedes dritte Segment weiterverarbeitet wird. Insgesamt findet eine Reduktion der Daten um den Faktor 24 statt.

Psychoakustische Verarbeitung

Aus Abschnitt 2.1.6 ist bekannt, dass das menschliche Gehör akustische Reize in Frequenzgruppen wahrnimmt. Um die Lautstärkenschwankungen während der 6 Sekunden eines Segments innerhalb der kritischen Bänder bestimmen zu können, wird eine STFT (siehe 2.2.2) durchgeführt und für jeden Frame das quadrierte Betragsspektrum berechnet **(b)**. Als Fensterfunktion kommt eine Hamming-Fensterfunktion (2.23) der Länge 256 zum Einsatz, wobei die einzelnen Frames sich zu 50% überlappen.

Das quadrierte Betragsspektrum jedes Frames wird entsprechend dem Außen- und Mittelohrmodell von Terhardt (2.2) gewichtet **(c)**, um die frequenzabhängige Laut-

stärkenwahrnehmung zu modellieren. Die Einteilung in kritische Bänder erfolgt entsprechend der *Bark*-Skala (2.5), indem das quadrierte Betragsspektrum innerhalb je eines kritischen Bands — entspricht genau einem Bark — aufsummiert wird **(d)**. Insgesamt werden 20 kritische Bänder berücksichtigt.

Anschließend versucht man spektrale Maskierungseffekte auszunützen. Dazu wird die *spreading function* nach Zwicker (2.7) mit jedem Frame — bestehend aus den 20 kritischen Bändern — gefaltet **(e)**. Dieser Schritt scheint nicht sehr sinnvoll. Einerseits macht nur die Berechnung eines Schwellwerts ohne den Vergleich mit dem tatsächlichen Spektrum keinen Sinn, da nicht ermittelt wird, welche Frequenzen unhörbar sind, andererseits — geht man von *mp3*-Dateien als Audioformat der Ausgangsdaten aus — wurden die unhörbaren Frequenzen schon im Zuge der Codierung aus dem Spektrum entfernt. Diese Annahme wird auch durch die experimentellen Erkenntnisse von Lidy und Rauber [14] untermauert. In ihren Experimenten wurde der Einfluss der einzelnen psychoakustischen Transformationen auf die Qualität einer Gerne Klassifizierungsaufgabe geprüft. Dabei stellte sich heraus, dass die Faltung mit der *spreading function* einen negativen Einfluss auf die Qualität der automatischen Klassifizierung anhand der *Fluctuation Patterns* hat, und es ist zu überlegen, diesen Schritt der psychoakustischen Vorverarbeitung bei der Berechnung der *Fluctuation Patterns* wegzulassen.

Obwohl für jeden Frame schon die frequenzabhängige Lautstärkenanpassung und die Gruppierung in die Frequenzgruppen durchgeführt wurde, ist die Einheit der Lautstärke noch immer *Pascal*. Um die Lautstärkenwahrnehmung zu modellieren, wird eine Umrechnung in *db-SPL* (2.1) und anschließend in *Sone* (2.3) durchgeführt **(f)**. Nach diesen Schritten ist die psychoakustische Vorverarbeitung abgeschlossen. Das Ergebnis ist eine 20×512 Matrix, die die Lautstärkenwahrnehmung alle 11.6 *ms* in den 20 kritischen Bändern beschreibt. Abbildung 3.2 stellt alle Schritte der psychoakustischen Verarbeitung **(a)-(f)** für eines der 6 Sekunden Segmente aus *Dire Straits - Walk of Live* dar.

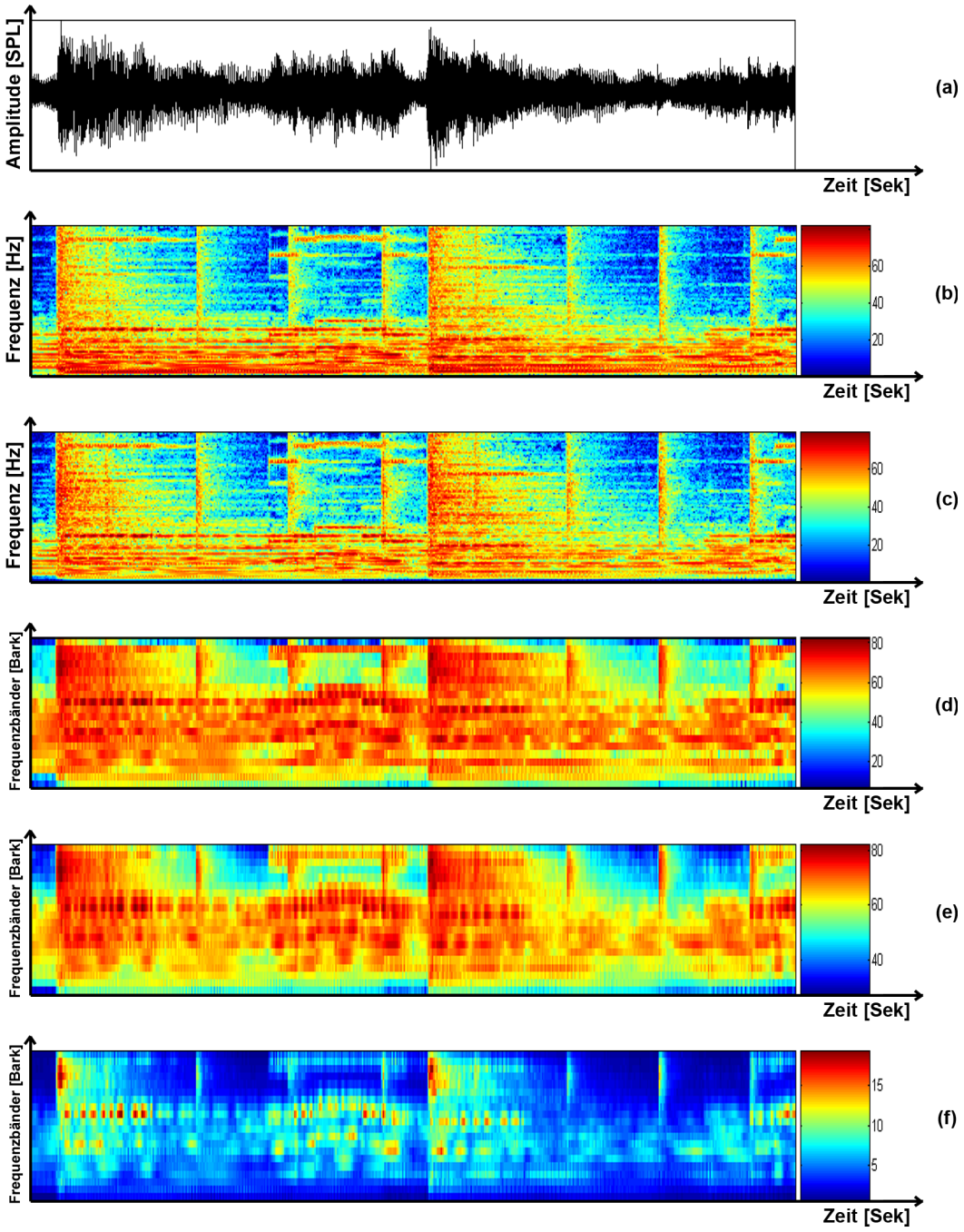


Abbildung 3.2: Visualisierung der psychoakustischen Verarbeitung

Extraktion der rhythmischen Struktur

Die psychoakustische Vorverarbeitung liefert die Änderung der Lautstärke innerhalb der 6 Sekunden Segmente in jedem kritischen Band mit der Zeit. Vor allem die periodischen Änderungen der Lautstärke charakterisieren den Rhythmus innerhalb eines Frequenzbandes. Periodische Änderungen wiederum können vortrefflich mittels der Fourier-Transformation erkannt werden. So erhält man durch erneute Anwendung der FFT auf jedes kritische Band eine Darstellung der rhythmischen Struktur pro Band **(g)**. Die FFT der Länge 512 mit einer rechteckigen Fensterfunktion (2.22) liefert eine 20×61 Matrix, die die Stärke der vorhandenen periodischen Lautstärkevariationen pro kritischem Band (Rhythmus) charakterisiert. Auch Rhythmen werden vom Menschen abhängig von ihrem Tempo unterschiedlich stark wahrgenommen (vgl. [38]). Daher findet eine Anpassung der wahrgenommenen Rhythmusstärke abhängig vom Tempo statt **(h)**. Dieser Einfluss wird mathematisch durch eine Gewichtung mit der so genannten Schwankungsstärke (3.1) nach [10] modelliert.

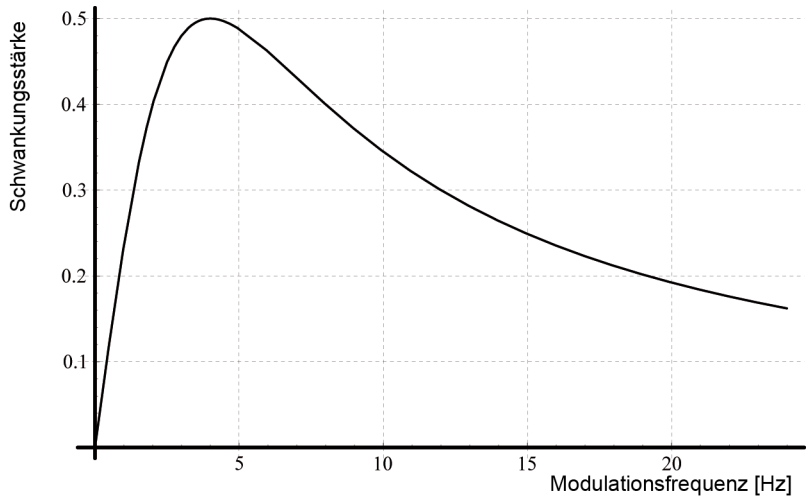


Abbildung 3.3: Die Schwankungsstärke abhängig von der Modulationsfrequenz

$$FluctuationStrength(f) = \frac{1}{\frac{f}{4} + \frac{4}{f}} \tag{3.1}$$

Wie in Abbildung 3.3 gut ersichtlich ist, werden Rhythmen mit einer Modulationsfrequenz von 4 *Hz* — das entspricht einem Tempo von etwa 240 Schlägen pro Minute — am stärksten wahrgenommen, während sehr niedrige Modulationsfrequenzen vom Menschen kaum mehr als Rhythmen interpretiert werden und daher eine sehr geringe Gewichtung aufweisen.

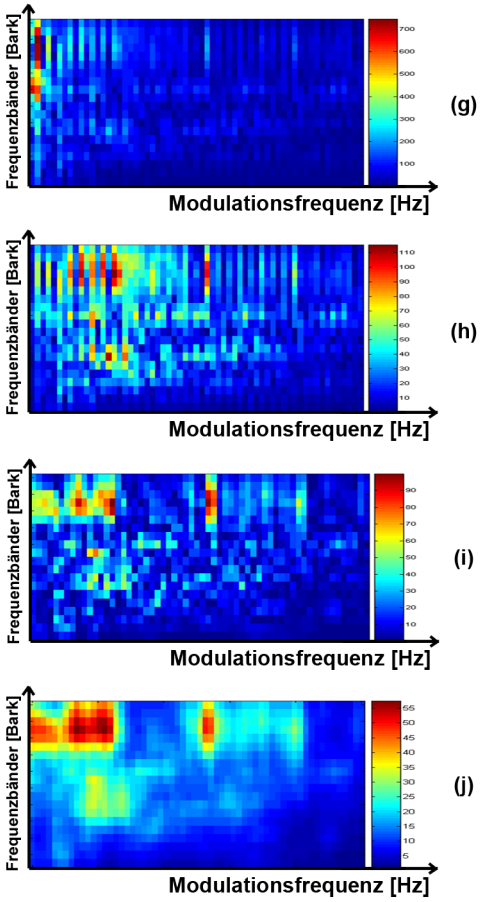


Abbildung 3.4: Visualisierung der Rhythmusextraktion

Abschließend wird noch ein Gaußfilter **(i)** und ein Gradientenfilter **(j)** angewandt, um bestimmte rhythmische Muster zu verstärken und unwichtige Information zu reduzieren. Der Gaußfilter erhöht die Ähnlichkeit von Rhythmusmustern, die geringfügige Unterschiede im Tempo der Rhythmen oder starke rhythmische Ähnlichkeiten in unterschiedlichen kritischen Bändern aufweisen. Als Resultat der Merk-

malsextraktion erhält man pro Segment eine 20×60 Matrix, die die rhythmische Struktur der 6 Sekunden eines Segments repräsentiert. Die Extraktion der rhythmischen Struktur für das 6 Sekunden Segment aus *Dire Straits - Walk of Live* wird in Abbildung 3.4 veranschaulicht.

3.3.2 Modellgenerierung

Für einen Musiktitel hat man nach der Merkmalsextraktion eine Menge an Rhythmusmustern $FP^1, FP^2, ..., FP^N$, je ein Rhythmusmuster pro Segment. Diese müssen nun zu einem Gesamtmodell für das zu repräsentierende Musikstück verschmolzen werden. Jedes der einzelnen Rhythmusmuster besteht aus einer 20×60 Matrix, je eine der zwanzig Zeilen für ein kritisches Band und sechzig Spalten für Tempos von 0 bis 600 *bpm* (*beats per minute*). Das Gesamtmodell FP^{total} ist wiederum eine derartige Rhythmusmatrix, wobei die einzelnen Elemente der Gesamtmatrix als der Median über die entsprechenden Elemente aller einzelnen Rhythmusmuster definiert ist.

$$fp_{ij}^{total} = Median(fp_{ij}^1, fp_{ij}^2, ..., fp_{ij}^N) \tag{3.2}$$

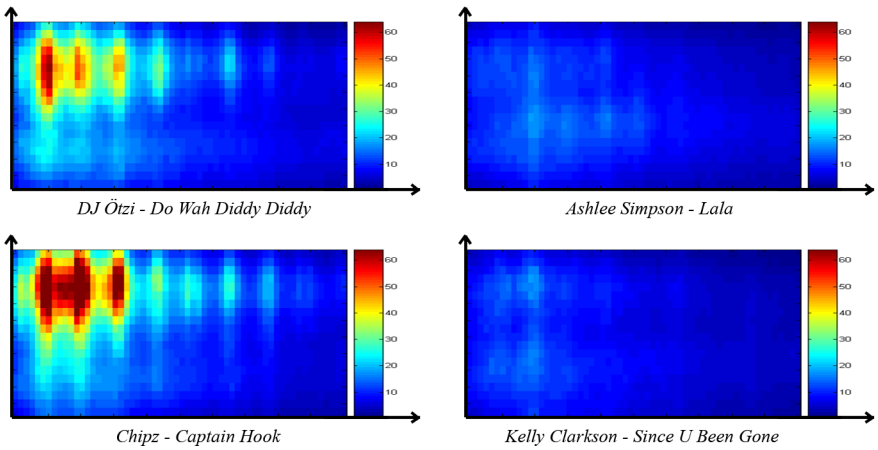


Abbildung 3.5: Vergleich der *Fluctuation Patterns* unterschiedlicher Musiktitel

Abbildung 3.5 zeigt die Gesamtmodelle von vier Musiktiteln. Deutlich zu erkennen sind die rhythmischen Ähnlichkeiten zwischen den übereinander angeordneten Titeln, und die Unterschiede nebeneinander platzierter Visualisierungen.

3.3.3 Distanzberechnung

Diese Rhythmusmatrix, die nun das gesamte Musikstück repräsentiert, nennt man *Fluctuation Pattern*. Die Distanz zwischen zwei *Fluctuation Patterns* A und B unterschiedlicher Lieder wird einfach durch die euklidische Distanz bestimmt.

$$D(A, B) = \sqrt{\sum_i \sum_j (a_{ij} - b_{ij})^2} \tag{3.3}$$

3.3.4 Zusammenfassung

Insgesamt betrachtet ist dieser Ansatz, die rhythmische Struktur eines gesamten Musikstücks zu erfassen, sehr innovativ. Eine der großen Stärken des Verfahrens liegt in der Ausnützung von psychoakustischen Eigenschaften. Sowohl der Extraktionsprozess als auch die Distanzberechnung zwischen Musikstücken lassen sich effizient implementieren. Das ist gerade für große Musiksammlungen ein wesentliches Kriterium. Zahlreiche Evaluierungen [23, 14] zeigen, dass die *Flucutation Patterns* rhythmische Ähnlichkeitsbeziehungen erfassen können, wenngleich der in Folge beschriebene klangfarben basierte Ansatz von Jean-Julien Aucouturier dem menschlichen Ähnlichkeitsempfinden näher kommt.

3.4 Timbre Distribution

Das hier als *Timbre Distribution* bezeichnete Verfahren wurde von Jean-Julien Aucouturier [1, 2, 3] entwickelt und versucht im Gegensatz zu den schon vorgestellten *Fluctuation Patterns* nicht den Rhythmus eines Musikstücks zu charakterisieren, sondern dessen klangfarbliche Charakteristiken zu beschreiben. Einen

Gesamtüberblick ausgehend vom Signal, über die Merkmalsextraktion und die Modellgenerierung bis hin zur Distanzberechnung gibt Abbildung 3.6.

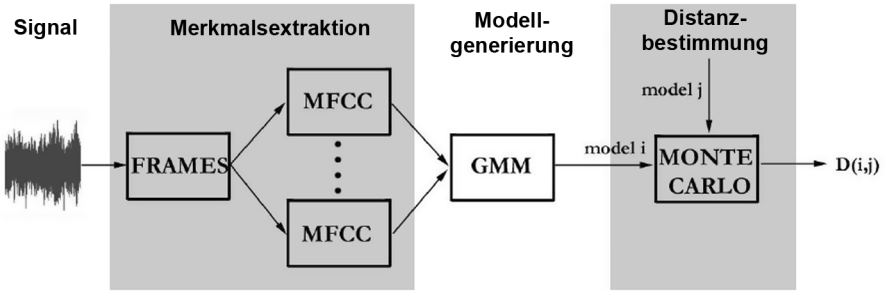


Abbildung 3.6: Überblick *Timbre Distribution* (aus [3])

3.4.1 Merkmalsextraktion

Zur Berechnung der klangfarblichen Eigenschaften eines Signals über die Zeit hinweg, greift Aucouturier auf den klassischen Segmentierungsansatz zurück, bei dem das Audiosignal in Frames unterteilt wird. Für jeden Frame wird dann die **spektrale Hülle** (*spectral envelope*) bestimmt. Der Grund für das Interesse an der spektralen Hülle ist, dass Forschungen [31] im Bereich der automatischen Instrumentenerkennung ergeben haben, dass gerade die spektrale Hülle einen großen Teil der Klangfarbe eines Instruments erklärt. Was genau man unter der spektralen Hülle versteht, soll nun geklärt werden.

Spectral Envelope

Die spektrale Hülle ist eine Kurve im Frequenz-Betragsraum (siehe Abbildung 3.7), die die Spitzen des Kurzzeitspektrums einhüllt. Für jeden Frame des Kurzzeitspektrums existiert eine spektrale Hülle, die die spektralen Eigenschaften zu einem Zeitpunkt charakterisiert.

Die spektrale Hülle liegt eng am Betragsspektrum (siehe Abschnitt 2.2.2) und verbindet dessen Spitzen. Dabei sollte die spektrale Hülle aber nicht zu stark schwin-

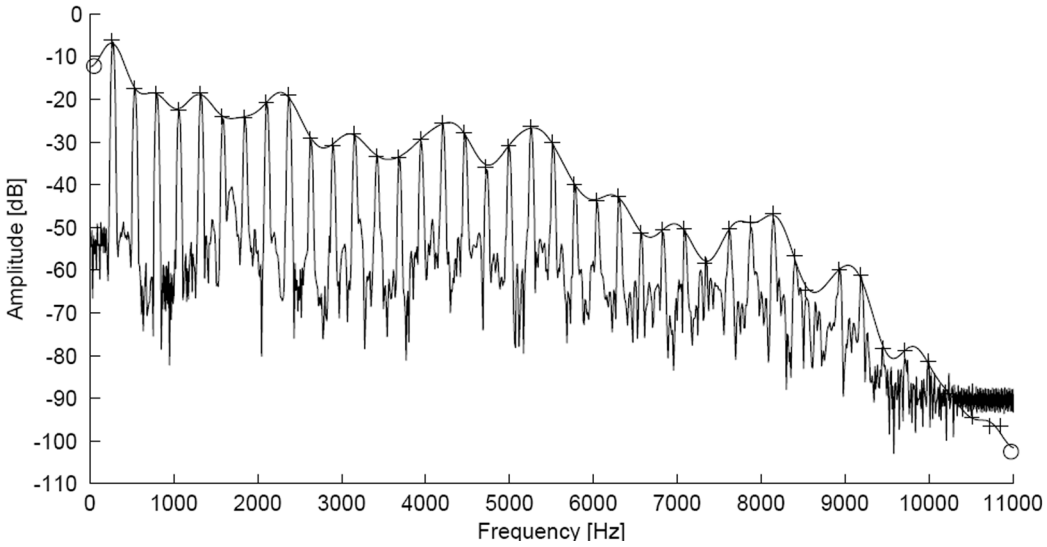


Abbildung 3.7: Die spektrale Hülle einer Violine (aus [30])

gen, sondern eine allgemeine Beschreibung der Verteilung der Energie des Signals bezüglich der Frequenz wiedergeben. Eine weitere wünschenswerte Eigenschaft ist, die spektrale Hülle durch eine stetige Funktion zu modellieren, so dass es keine Sprungstellen gibt.

Mel Frequency Cepstrum Coefficients

Prinzipiell gibt es mehrere Verfahren, um die spektrale Hülle eines Frames zu berechnen. Einer der ersten Ansätze basierte auf *Linear Predicting Codes*. Eine andere weit verbreitete Möglichkeit die spektrale Hülle zu schätzen und darzustellen ist es, die so genannten *Mel Frequency Cepstrum Coeffizients* zu berechnen. Ausgangspunkt für die Berechnung bildet das *Log-Spektrum*, das als Logarithmus des Betragsspektrums definiert ist (vgl. [19]).

$$\log(M(k)) = \log|X(k)| \tag{3.4}$$

Das *Cepstrum* ist als das invers Fourier-Transformierte des Log-Spektrums definiert und berechnet sich wie folgt:

$$c_n = \frac{1}{N} \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}nk} \log |X(k)| \tag{3.5}$$

Führt man vor der Berechnung des Cepstrums noch eine Anpassung an die psychoakustische *Mel*-Skala (nach 2.6) durch, so erhält man statt des Cepstrums das *Mel-Cepstrum*. Die Koeffizienten c_n aus (3.5) werden dann als *Mel Frequency Cepstrum Coefficients*, oder kurz MFCCs, bezeichnet. Die Koeffizienten niedriger Ordnung beschreiben die langsamen Änderungen der spektralen Hülle, wogegen die Koeffizienten hoher Ordnung schnelle Änderungen der Hülle modellieren. Verwendet man nur die ersten M Koeffizienten zur Repräsentation der spektralen Hülle, so erhält man eine geglättete Approximation des *Mel-Cepstrums*. Jean-Julien Aucouturier zeigte in seinen Experimenten [3], dass für die Musikverarbeitung die ersten 20 Koeffizienten eine gute Schätzung der spektralen Hülle erlauben.

Mit der Berechnung der MFCCs verfügt man über eine Möglichkeit, die Klangfarbe jedes Frames zu bestimmen. Um jedoch zu einem Gesamtmodell für ein Musikstück zu kommen, ist eine Generalisierung über alle Frames nötig.

3.4.2 Modellgenerierung

Die Verallgemeinerung zu einem Gesamtmodell wird durch die Schätzung der Verteilung der MFCC-Vektoren bewerkstelligt. Daher auch der hier eingeführte Name *Timbre Distribution*. Der Extraktionsprozess liefert für jeden Frame einen MFCC-Vektor der Dimension 20, welcher als Punkt in einem Vektorraum gesehen werden kann. Die Verteilung der Punkte im Vektorraum wird durch ein *Gaussian Mixture Model*[1] (GMM) angenähert. Ein Gaussian Mixture Model stellt eine n-dimensionale Verteilung als Kombination von k n-dimensionalen Normalverteilungen dar. Die Anzahl k der verwendeten Normalverteilungen, auch Komponenten

[1] Details zu *Gaussian Mixture Models* entnehme man [17].

genannt, bestimmt, wie genau eine Verteilung durch eine GMM modelliert werden kann. Abbildung 3.8 soll dem Leser eine Vorstellung des Verfahrens geben.

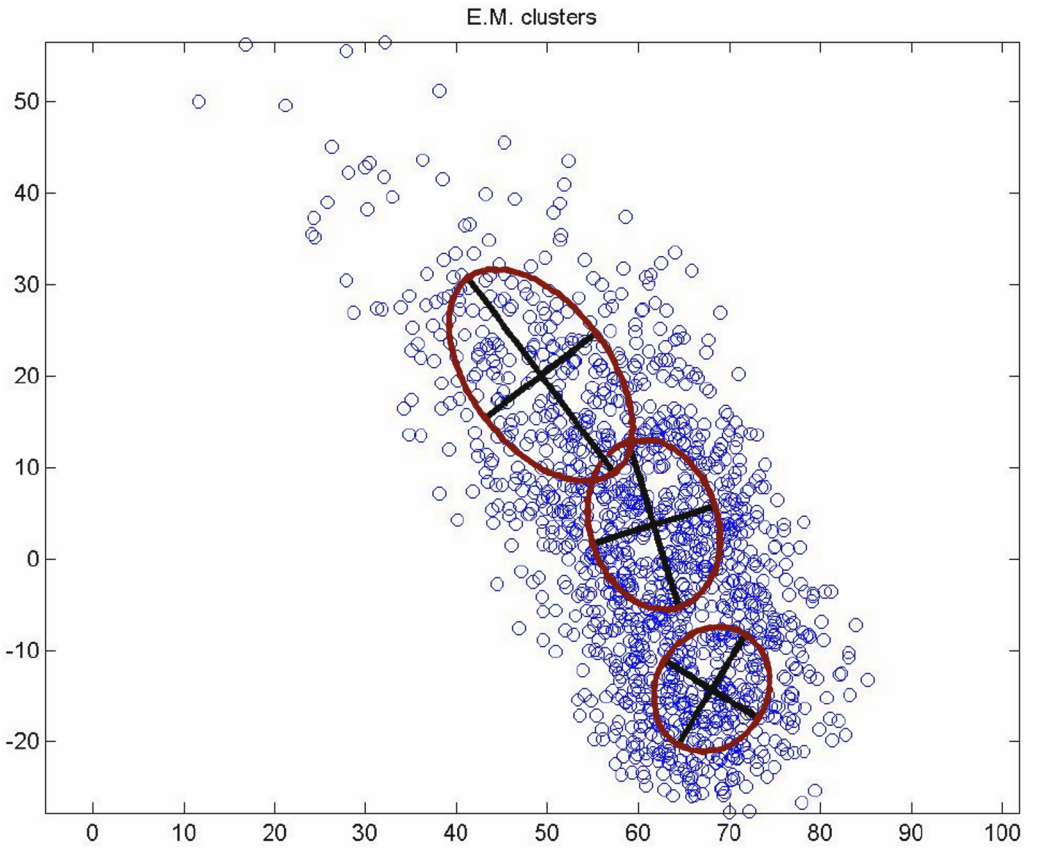


Abbildung 3.8: Eine Verteilung von MFCC-Vektoren modelliert durch ein GMM mit 3 Komponenten (aus [2])

Die Wahrscheinlichkeit, einen Vektor $\vec{x}$ unter der Wahrscheinlichkeitsverteilung, die durch ein GMM modelliert wird, zu beobachten, ist gegeben durch:

$$p(\vec{x}) = \sum_{i=1}^{k} P(C = i)p(\vec{x}|C = i) \tag{3.6}$$

wobei $C = i$ die i-te Komponente identifiziert und $p(\vec{x}|C = i)$ eine n-dimensionale Normalverteilung mit dem Erwartungswert $\vec{\mu}_i$ und der Kovarianzmatrix Σ_i ist.

$$p(\vec{x}|C=i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^T \Sigma_i^{-1}(\vec{x}-\vec{\mu_i})} \tag{3.7}$$

Zum Schätzen der Parameter $\vec{\mu}_i$, Σ_i und $P(C=i)$ eines GMM aus einer vorhandenen Stichprobe, wie in unserem Fall der MFCC-Vektoren, versucht man nach dem **Maximum Likelihood Prinzip** jenes Modell zu wählen, das am wahrscheinlichsten die Daten generiert hat. Während für einfache parametrische Verteilungen nach dem Maximum Likelihood Prinzip die Parameter der Verteilung analytisch bestimmt werden können, erfolgt die Schätzung der Parameter für ein GMM schrittweise. Der Trainingsalgorithmus eines GMM ist allgemein unter dem Namen *Expectation-Maximization Algorithm* bekannt und ist im Wesentlichen ein Gradientenabstiegsverfahren. Für eine genaue Beschreibung des EM-Algorithmus und eine entsprechende mathematische Herleitung sei auf [17] verwiesen.

3.4.3 Distanzberechnung

Die Schätzung der Verteilung der MFCC-Vektoren durch ein GMM liefert ein kompaktes Gesamtmodell für die klangfarblichen Eigenschaften eines Musiktitels. Zur Distanzmessung zwischen zwei Gesamtmodellen A und B wird ein **Monte Carlo**-Ansatz verwendet. Aus den Verteilungen von A und B kann je eine Stichprobe (S^A und S^B) an MFCC Vektoren generiert werden. Für jede Stichprobe wird die Wahrscheinlichkeit, jene Vektoren sowohl unter dem eigenen Modell als auch unter dem anderen Modell zu beobachten, berechnet. Das symmetrische und normalisierte Distanzmaß wird dann nach folgender Formel aus [3] ermittelt:

$$\begin{aligned} D(A,B) = &\sum_{i=1}^{i=DSR} \log P(S_i^A|A) + \sum_{i=1}^{i=DSR} \log P(S_i^B|B) \\ &- \sum_{i=1}^{i=DSR} \log P(S_i^A|B) - \sum_{i=1}^{i=DSR} \log P(S_i^B|A) \end{aligned} \tag{3.8}$$

Die Genauigkeit der Distanzberechnung hängt klar von der Anzahl der Elemente in der Stichprobe ab, die als *Distance Sample Rate* (DSR) bezeichnet wird.

3.4.4 Zusammenfassung

Der Ansatz von Aucouturier orientiert sich an einem klassischen Ansatz zur Mustererkennung. Interessant ist jedoch die Idee, ein Musikstück einfach als Verteilung von MFCC-Vektoren zu betrachten, wodurch man eine zeitunabhängige Repräsentation erhält. Obwohl die Qualität der Ähnlichkeitsbestimmung aufgrund von klangfarblichen Eigenschaften recht überzeugend ist, ist wohl der größte Nachteil, die lange Rechenzeit. Die aktuelle Forschungsarbeit von Elias Pampalk [25] verspricht aber Abhilfe im Bezug auf lange Rechenzeiten. Vereinfachungen des erzeugten Modells — es kommt nur eine einzige Komponente zum Einsatz — bewirken eine Verschnellerung bei der Modellgenerierung circa um den Faktor 833 und bei der Distanzberechnung um den Faktor 4000, während die Qualität des Ähnlichkeitsmaßes nach [18] unverändert bleibt.

3.5 Implementierung

Beide hier vorgestellten Verfahren wurden im Rahmen der Diplomarbeit implementiert und in das Institutsprojekt **CoMIRVA**[2] (*Collection of Music Information Retrieval and Visualization Applications*) integriert. Ein komplettes Framework basierend auf der *Java Sound API* wurde entwickelt, das neben den beiden Verfahren — als Beispielimplementierungen — alle nötigen Funktionen, die für ein effizientes Prototyping und einfache Implementierung weiterer Verfahren nötig sind, zur Verfügung stellt.

Aufbauend auf **CoMIRVA**, im Speziellen auf den Audioverarbeitungsteil, wurde ein System zur ähnlichkeitsbasierten Navigation und zur Visualisierung von Musiksammlungen entwickelt, das den Namen **Music Browser** trägt. Das nächste Kapitel geht auf die neu entwickelten Konzepte und Algorithmen, die zur Navigation und für die Visualisierung im **Music Browser** eingesetzt werden, im Detail ein.

[2] http://www.cp.jku.at/comirva

4 Navigation und Visualisierung

Musiksammlungen mit mehreren tausend Musiktiteln sind heutzutage völlig selbstverständlich, wenngleich die Besitzer meist keinerlei Überblick über ihre Sammlung haben. Abhilfe sollen hier moderne Anwendungen schaffen, die es dem Benutzer ermöglichen, seine Sammlung effizient zu nutzen.

Wesentlich für den Benutzer einer solchen Anwendung ist, dass er sich innerhalb seiner Musiksammlung orientieren kann. Eine sehr intuitive Art, den Benutzer bei der Orientierung zu unterstützen ist es, die Struktur der gesamten Musiksammlung als Landkarte zu visualisieren. Diese Idee scheint vielversprechend und wurde schon in mehreren Ansätzen untersucht [22, 11]. Neben der Möglichkeit der Orientierung ist es aber ebenso wichtig, dass der Benutzer sich innerhalb seiner Musiksammlung bewegen kann. Die Nutzung der Kartenmetapher für Musiklandschaften ist in Hinblick auf die Navigation eher begrenzt. Vergleichbar mit einem Straßennavigationssystem benötigt man für die lokale Navigation eine hochauflösende Straßenkarte. Während der Navigation innerhalb der Straßenkarte verliert man aber schnell die Orientierung im globalen Kontext. Man kann zwar feststellen, welches die nächste Seitenstraße ist, nicht aber, ob man Richtung Wien oder Richtung Salzburg fährt.

Eine Lösung für dieses Problem könnte der hier vorgestellte Ansatz der *mehrstufigen Navigation* sein. Eine grobe globale Karte unterstützt den Benutzer bei der Orientierung innerhalb der gesamten Musiksammlung, während eine sehr feine Darstellung der aktuellen Umgebung die Möglichkeit gibt, tatsächlich gezielt in der Sammlung zu navigieren. Die globale und die lokale Darstellung sollten dabei verknüpft, und die aktuelle Position innerhalb des lokalen Ausschnitts in der globalen Darstellung erkennbar sein.

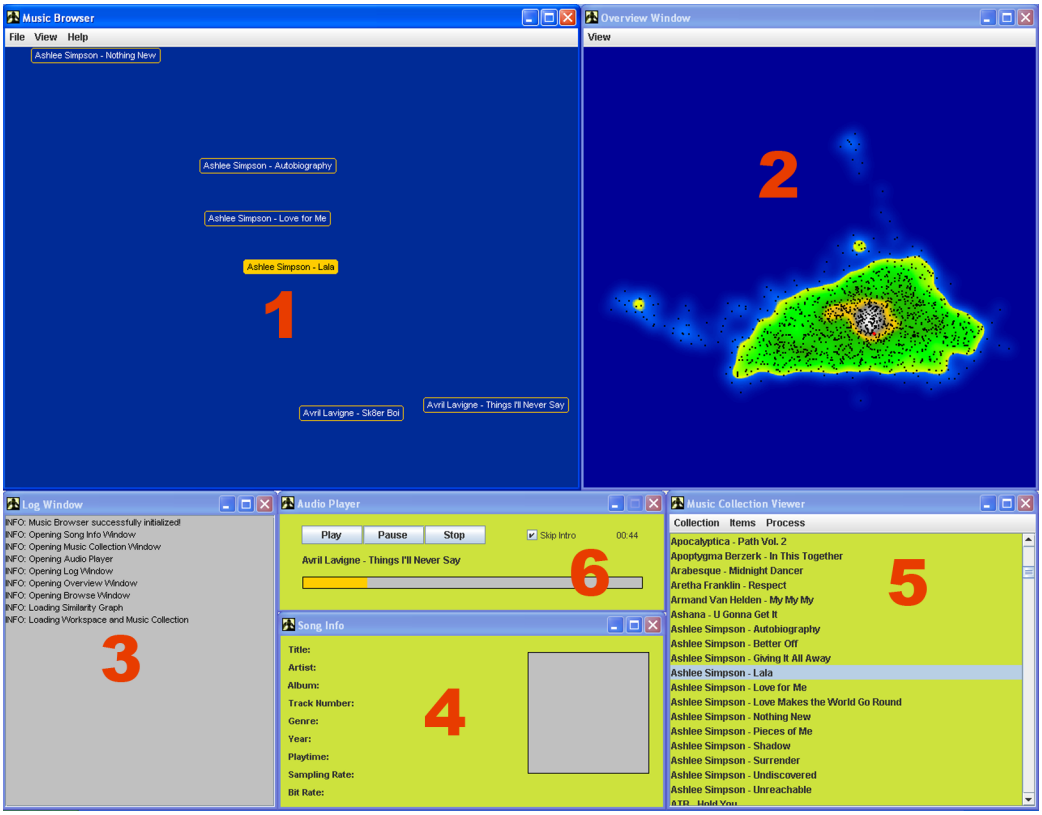


Abbildung 4.1: Prototyp eines mehrstufigen Navigationssystems: **Music Browser**

Die Idee der mehrstufigen Navigation wurde in Form eines Prototyps mit dem Namen **Music Browser** umgesetzt. Der **Music Browser** ist eine integrierte Musikverwaltungsumgebung. Neben der Verwaltung von privaten Musiksammlungen **(5)**, der Navigationsunterstützung **(1)** und der Visualisierung der Musiksammlung, **(2)** runden ein Audioplayer **(6)**, eine Informationsfenster[1] zum aktuell fokussierten Musiktitel **(4)** und ein Nachrichtenfenster **(3)** die Musikverwalungsumgebung ab. Dank des **Music Browsers** kann der Ansatz der mehrstufigen Navigation nun auch praktisch evaluiert werden. Eine kurze Zusammenfassung der gewonnenen Erkenntnisse und erste Meinungen von Testnutzern, sowie einen Ausblick auf Verbesserungsmöglichkeiten findet man im letzten Kapitel der Arbeit.

[1] Der eingesetzte *Cover Crawler* zur automatischen Suche nach Albumbildern wurde von Markus Schedl nach [28] implementiert.

Die folgenden Abschnitte beschäftigen sich mit dem grundlegenden Modell, auf dem die Navigation und Visualisierung im *Music Browser* beruht. Es wird auch im Detail beschrieben, wie die globale Karte erzeugt und die lokale Navigation realisiert wurde.

4.1 Der Ähnlichkeitsgraph

Als Basisdatenstruktur verwendet der **Music Browser** einen Ähnlichkeitsgraphen. Jeder Knoten im Ähnlichkeitsgraphen repräsentiert ein Musikstück, und die Kanten des Graphen verbinden jedes Musikstück mit den k ähnlichsten Musikstücken. Welche Musiktitel die k ähnlichsten Titel sind, wird anhand der Distanzen d_{ij} bestimmt. Diese wiederum lassen sich für je zwei Musiktitel S_i und S_j durch eine inhaltsbasierte Ähnlichkeitsmetrik $d_{ij} = D(S_i, S_j)$ berechnen (siehe Kapitel 3). Ein solcher Graph wird allgemein als k-Nachbarschaftsgraph (*k-nearest neighbor graph*) bezeichnet (vgl. [12]). Die Verwendung des k-Nachbarschaftsgraphen als Basisdatenstruktur ist durch mehrere Überlegungen motiviert.

Inhaltsbasierte Ähnlichkeitsmetriken erlauben nur die Distanzbestimmung $d_{ij} = D(S_i, S_j)$ für zwei Musikstücke S_i und S_j. Die Interpretation eines Musikstücks als Punkt oder Vektor in einem Raum nur basierend auf den Distanzen d_{ij} ist nicht so einfach möglich. Folglich ist es sinnvoll, eine distanzbasierte Datenstruktur einzusetzten, wie etwa den k-Nachbarschaftsgraphen, bei dem die Distanzen leicht durch die Länge der Kanten modelliert werden können.

Ein weiteres Motiv für die Verwendung des k-Nachbarschaftsgraphen ist die Existenz effizienter Algorithmen zur Berechnung bzw. Annäherung des Ähnlichkeitsgraphen (vgl. [12]). Damit ist der Nachbarschaftsgraph auch in Bezug auf die Laufzeitkomplexität eine gute Wahl. Gerade auch die Speicherkomplexität des Nachbarschaftsgraphen ist mit $O(N)$ aufgrund der durch die Konstante k limitierten Nachbarschaftsbeziehungen ein entscheidendes Kriterium. Speziell im Hinblick auf die Anwendung auf grössere Musiksammlungen, die — wie es heute üblich ist — oft über 1.5 Millionen Titel beinhalten, sind Laufzeit und Speicherkomplexitätsüberlegungen von entscheidender Bedeutung. Als Beispiel sei hier erwähnt,

dass selbst die Konstruktion der Distanzmatrix, die alle Distanzen d_{ij} zwischen allen Musiktiteln beinhaltet, für eine Größenordnung von 1.5 Millionen Titeln hinsichtlich der Speicherkomplexität mit $O(N^2)$ kaum mehr verwaltbar ist. Würde man für die Speicherung einer Distanz den Datentyp *Integer (4 Bytes)* verwenden, so wäre die Distanzmatrix ca. 8.12 *Terabyte* groß. Das ist eine Größenordnung, die selbst für Großrechner nicht mehr bewältigbar ist, da die Distanzmatrix aus Effizienzgründen im Hauptspeicher gehalten werden sollte.

Neben diesen theoretischen Überlegungen ist der k-Nachbarschaftsgraph als Basis für die lokale Navigation eine geradezu ideale Datenstruktur (vgl. [27]). Ausgehend vom aktuellen Musikstück kann man sehr schnell die nächsten k Nachbarn bestimmen, sowie deren nächste Nachbarn. Allgemein betrachtet kann man also durch das Ausschneiden eines *Subgraphen* ausgehend vom aktuellen Musiktitel sehr effizient die lokale Nachbarschaft ermitteln. Dazu kommt, dass die Qualität der lokalen Nachbarschaft automatisch mit der Größe der Musiksammlung steigt, da der k-Nachbarschaftsgraph per Definition immer die nähesten Titel verbindet. Die aktuelle Implementierung im **Music Browser** verwendet eine leicht modifizierte Version des k-Nachbarschaftsgraphen. Wieso und wie dieser Graph erzeugt wird, beschreibt der nächste Abschnitt.

4.1.1 Konstruktion des Ähnlichkeitsgraphen

Da der Ähnlichkeitsgraph auch als Basis für das Navigationsmodell dienen soll, muss man sich speziell auch Gedanken über die Erreichbarkeit von Knoten im Graphen machen, schließlich soll man ja die gesamte Musiksammlung durchstöbern können.

Die Kanten des in dieser Arbeit verwendeten Nachbarschaftsgraphen sind nicht gerichtet. Das bedeutet, dass die durch die Kanten modellierten Ähnlichkeitsbeziehungen immer bidirektional sind und man über eine Kante im Graphen immer in beide Richtungen navigieren kann. Damit ist garantiert, dass jeder Knoten zumindest von k anderen Knoten aus erreicht werden kann. Zusätzlich erleichtert dieser Ansatz die Erreichbarkeitsüberlgegungen erheblich. **Genau dann, wenn**

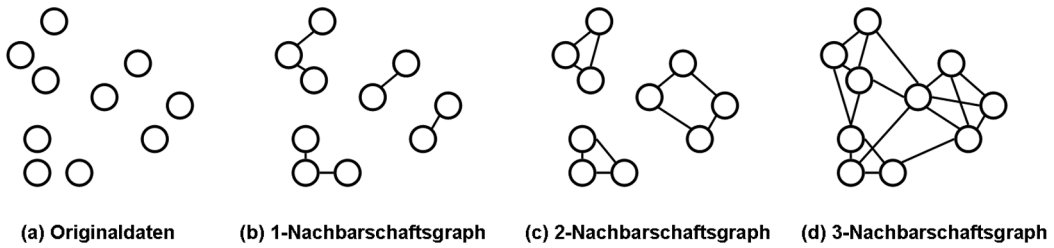


Abbildung 4.2: k-Nachbarschaftsgraph

der Ähnlichkeitsgraph zusammenhängend ist, sind alle Knoten im Graphen von jedem beliebigen Knoten aus erreichbar.

Nun gibt es leider keine Garantie, dass ein Nachbarschaftsgraph der Ordnung k auch tatsächlich zusammenhängend ist. Beobachtet man aber die Änderung der Struktur eines Nachbarschaftsgraphen abhängig von seiner Ordnung, so fällt auf, dass die Anzahl der *Cluster* mit steigender Ordnung schnell abnimmt.Abbildung 4.2 veranschaulicht diese Eigenschaft. Während die Originaldaten **(a)** natürlich aus 10 Clustern bestehen, sind es in **(b)** nur mehr 4, in **(c)** nur mehr 3 und ab der Kantenordnung 3 **(c)** ist der Graph bereits zusammenhängend. Aus dieser Beobachtung abgeleitet wurde ein Algorithmus erdacht, der den Nachbarschaftsgraphen schrittweise konstruiert und garantiert, dass der Nachbarschaftsgraph zusammenhängend ist.

Die Grundidee ist immer gleichzeitig alle Kanten der Ordnung k in einem Schritt in den Graphen einzufügen, wie in Abbildung 4.2. Eine Kante ist von der Ordnung k, wenn sie den aktuell betrachteten Knoten mit dem k nächsten Knoten in seiner Nachbarschaft verbindet. Diese Definition ist also relativ zum betrachteten Knoten, und man muss jeden Knoten des Graphen einzeln betrachten, um die zugehörige Kante der Ordnung k einfügen zu können. Die Konstruktion eines Ähnlichkeitsgraphen der Ordnung $\mathbf{K}$ benötigt demzufolge $\mathbf{K}$ Schritte. Während jedes Schrittes führt man Buch über die aufgetretenen Verschmelzungen von Clustern in der Struktur des Graphen und aktualisiert die Anzahl der Cluster am Ende jedes Schritts. So ist es einfach nach $\mathbf{K}$ Schritten zu prüfen, ob noch mehr als ein Cluster vorhanden ist. Gibt es mehr als einen Cluster, so wird ein weiterer Schritt

durchgeführt, wobei nur mehr jene Kanten eingefügt werden, die zur Verschmelzung zweier Cluster in diesem Schritt führen. Den letzten Schritt wiederholt man so lange, bis nur mehr ein Cluster vorhanden ist und damit garantiert ist, dass der Graph zusammenhängend ist. Man beachte, dass aufgrund der Buchführung über die aktuell vorhandenen Cluster, die Prüfung, ob mehr als ein Cluster vorhanden ist, sehr effizient durchgeführt werden kann. Algorithmus 1 gibt den Pseudocode für dieses Verfahren wieder.

Für eine Menge $M = node_1, ..., node_N$ von **N** Knoten soll der Ähnlichkeitsgraph SG der Ordnung **K** erzeugt werden. Um den Algorithmus kompakt darstellen zu können wird angenommen, dass folgende Methoden zur Verfügung stehen:

- **getKNearestNode**($node_i$, k)
  Liefert für den Knoten $node_i$ den k nächsten Knoten in seiner Nachbarschaft. Dies kann anhand der Distanzen d_{ij} bestimmt werden.
- **addEdge**($node_i$, $node_j$)
  Fügt in den Graphen eine ungerichtete Kante zwischen Knoten i und Knoten j ein, falls diese noch nicht vorhanden ist.
- **getCluster**($node_i$)
  Liefert den Cluster, zu dem der Knoten $node_i$ gehört.
- **sameCluster**($node_i$, $node_j$)
  Liefert einen boolschen Wert, der angibt, ob zwei Knoten im selben Cluster liegen oder nicht.
- **rememberToMerge**($cluster_i$, $cluster_j$)
  Merkt zwei Cluster zur Verschmelzung zu einem Cluster vor.
- **mergeClusters**()
  Alle vorgemerkten Cluster werden entsprechend verschmolzen und die Anzahl der verbleibenden Cluster wird zurückgegeben.

Ein wichtiger Aspekt des Algorithmus ist, dass das tatsächliche Zusammenführen der Cluster erst nach der **for**-Schleife erledigt wird. Führen mehrere Kanten der

Algorithm 1 Graph Construction Algorithm

```
1: clusters ← N
2: k ← 1
3: while clusters > 1 do                          //till there is only one cluster
4:    for i = 0 to N do                           //insert all edges of order k
5:       kNearest ← getKNearestNode(node_i, k)
6:       if (k < K) ∨ ¬ sameCluster(node_i, kNearest) then
7:          addEdge(node_i, kNearest)
8:          rememberToMerge(getCluster(node_i), getCluster(kNearest))
9:       end if
10:   end for
11:   clusters ← mergeClusters()
12:   k ← k + 1
13: end while
```

Ordnung k zum Verschmelzen zweier Cluster, so werden alle diese Kanten eingefügt, nicht nur die erste Kante, die diese zwei Cluster verschmelzen würde. Dies führt dazu, dass einzelne *Subcluster* nicht nur durch eine Kante, sondern durch mehrere Kanten verknüpft sind, und erleichtert so die Navigation über Clustergrenzen hinweg.

Das Ergebnis der Konstruktion des Ähnlichkeitsgraphen ist ein k-Nachbarschaftsgraph, der durch das Einfügen von zusätzlichen Kanten so erweitert wurde, dass er zusammenhängend ist. Dadurch ist gewährleistet, dass man beim Navigieren tatsächlich alle Knoten im Graphen erreichen kann. Die Garantie, dass der Graph zusammenhängt, erweist sich auch im nächsten Schritt, der Generierung einer globalen Karte, als hilfreich.

4.2 Globale Darstellung

Die globale Darstellung soll die Musiksammlung in einer Art Karte darstellen. Da der Nachbarschaftsgraph als Grundlage für die Navigation dient, und die glo-

bale Karte und die lokale Navigation miteinander verknüpft sein sollen, ist eine intuitive Lösung die Darstellung des Ähnlichkeitsgraphen auf einer Ebene. Der Ähnlichkeitsgraph ist im Allgemeinen aber hochdimensional. Um ihn auf einer Ebene darstellen zu können, ist es daher nötig eine Dimensionsreduktion durchzuführen. Unter einer Dimensionsreduktion versteht man ein Verfahren, das eine Projektion der hochdimensionalen Daten auf einen niedrigdimensionalen Raum (meist eine Ebene) so erzeugt, dass die wesentlichen Struktureigenschaften im hochdimensionalen Raum auch im niedrigdimensionalen Raum erhalten bleiben. Im **Music Browser** wird zur Dimensionsreduktion ein modifizierter *Multidimensional Scaling* (MDS) Ansatz verwendet.

4.2.1 Multidimensional Scaling

MDS[2] ist ein Verfahren, um für eine Menge von **N** Objekten mit den Distanzen $\Delta = \{d_{ij}|i,j = 1,...,N\}$ eine Konfiguration $P = \{\vec{p}_1,...,\vec{p}_N\}$ zu finden, sodass der Punkt $\vec{p}_i$ das Objekt o_i in einem niedrigdimensionalen Raum darstellt. Die Distanzen δ_{ij} im niedrigdimensionalen Raum zwischen zwei Objekten o_i und o_j lassen sich demnach wie folgt bestimmen:

$$\delta_{ij} = ||\vec{p}_i - \vec{p}_j|| \tag{4.1}$$

Beschränkt man sich auf eine Projektion auf eine Ebene — wie es für die Generierung einer Karte sinnvoll ist — dann vereinfacht sich der Abstand zwischen zwei Objekten mit $\vec{p}_i = \begin{pmatrix} p_{ix} \\ p_{iy} \end{pmatrix}$ und $\vec{p}_j = \begin{pmatrix} p_{jx} \\ p_{jy} \end{pmatrix}$ zu:

$$\delta_{ij} = ||\vec{p_i} - \vec{p_j}|| = \sqrt{(p_{ix} - p_{jx})^2 + (p_{iy} - p_{jy})^2} \tag{4.2}$$

Die grundlegende Idee hinter MDS Verfahren ist, dass die Distanzen δ_{ij} zwischen zwei Objekten in der Ebene die tatsächlichen Distanzen d_{ij} im hochdimensionalen Raum möglichst gut annähern sollen.

[2] Dieser Abschnitt basiert auf den Arbeiten [4, 6, 8, 13].

$$\forall i \forall j : \delta_{ij} \approx d_{ij} \tag{4.3}$$

Einleuchtend ist natürlich, dass die Distanzen δ_{ij} nie exakt die Distanzen d_{ij} darstellen können, da für eine exakte Darstellung der Distanzen in einer Ebene die Datenpunkte im hochdimensionalen Raum ebenfalls auf einer Ebene liegen müssten, was im Allgemeinen naturgemäß nicht der Fall ist. Demzufolge kann man aber den Fehler, der aus einer Approximation resultiert, quantifizieren,

$$e_{ij} = |\delta_{ij} - d_{ij}| \tag{4.4}$$

und es lässt sich für die gesamte Konfiguration eine Bewertungsfunktion, häufig mit *Stress-* oder *Loss*-Funktion bezeichnet, definieren, die den Gesamtapproximationsfehler angibt:

$$Loss(P) = \sum_{i=0}^{N} \sum_{j=0}^{N} e_{ij}^2 \tag{4.5}$$

Ausgehend von einer zufälligen Konfiguration kann durch die Optimierung bezüglich der *Loss*-Funktion eine Konfiguration gefunden werden, die die Struktur der Daten im hochdimensionalen Raum gut annähert. Häufig verwendet man zur Optimierung die generalisierte **Newton-Raphson Methode** und leitet die *Loss*-Funktion nach den Parametern ab. Es lässt sich dann für jeden Punkt $\vec{p}_i$ der Gradient bestimmen, und man kann die Konfiguration verbessern, indem man schrittweise für jedes Objekt o_i den Punkt $\vec{p}_i$ in Richtung des steilsten Abstiegs (des negativen Gradienten) verschiebt, solange bis die Qualität der Konfiguration bei einem der lokalen Minima konvergiert. Grundsätzlich können aber zur Optimierung hinsichtlich der *Loss*-Funktion auch andere Optimierungsverfahren wie **genetische Algorithmen**, **Tabu-Suche** oder **Simulated Annealing** eingesetzt werden. Der nächste Abschnitt beschreibt die Modifikationen der Bewertungsfunktion im **Music Browser**, die in Anlehnung an die klassische Bewertungsfunktion definiert wurde.

4.2.2 Modifikation der Bewertungsfunktion

Im Gegensatz zum klassischen MDS Verfahren, das alle Distanzen d_{ij} zwischen allen N Objekten berücksichtigt, soll für die Erstellung der Karte im **Music Browser** nur der Ähnlichkeitsgraph in einer Ebene dargestellt werden. Das bedeutet, dass für jeden Knoten $node_i$ des Graphen nur die Kanten im Graphen als Distanzen zu berücksichtigen sind. Dadurch ändert sich natürlich die *Loss*-Funktion, und es reduziert sich der Rechenaufwand erheblich. Darüber hinaus beeinflusst die *Loss*-Funktion das Ergebnis der Positionierung der Knoten des Graphen in der Ebene ganz grundlegend. Eine zentrale Forderung an die Ausrichtung des Graphen in der Ebene ist, dass die Nachbarn eines Knoten $node_i$ in der Nähe des Knotens $node_i$ platziert werden sollen. Diese Forderung fließt in die modifizierte *Loss*-Funktion ein.

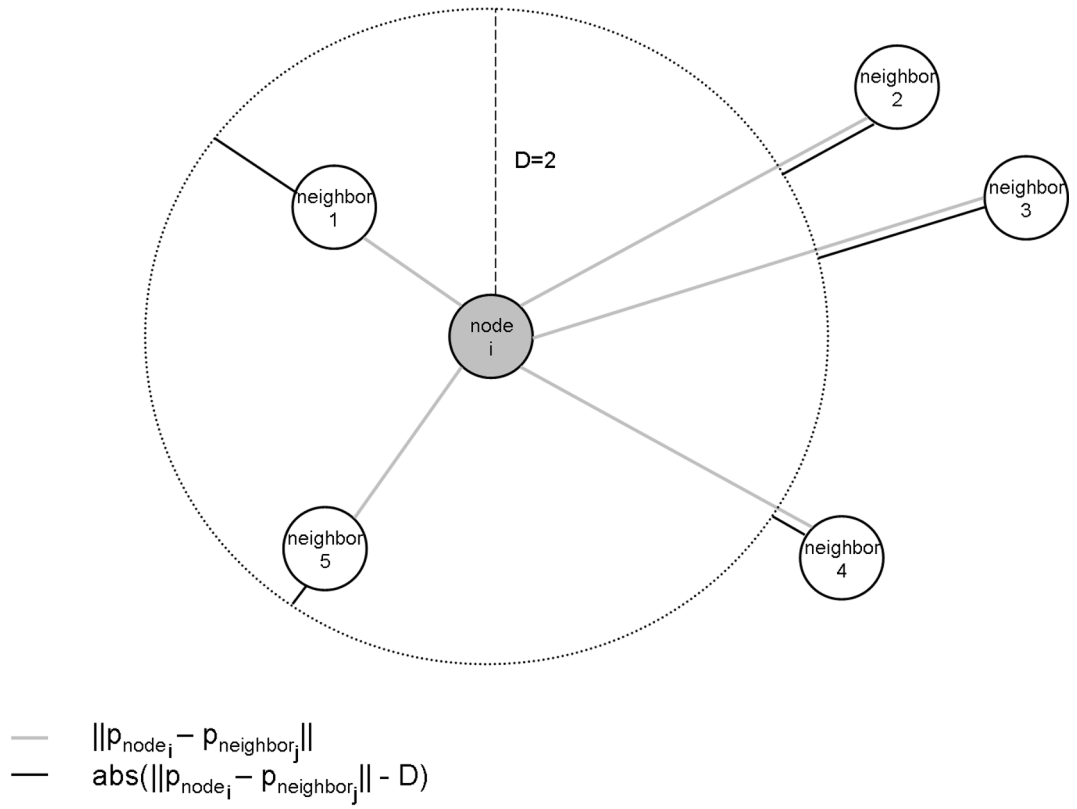


Abbildung 4.3: Darstellung des Knotenfehlers e_{node_i}

Sei $|node_i|$ die Anzahl der Kanten vom Knoten $node_i$ zu seinen Nachbarn $neighbor_j$, dann definiert sich der Fehler eines Knoten wie folgt:

$$e_{node_i} = \sum_{j}^{|node_i|} w_{node_i,neighbor_j} abs(||\vec{p}_{neighbor_j} - \vec{p}_{node_i}|| - D) \tag{4.6}$$

Man beachte, dass der Fehler nicht mehr auf den einzelnen Distanzen d_{ij} sondern pro Knoten $node_i$ definiert ist. Die Nachbarknoten $neighbor_j$ sind relativ zum Knoten $node_i$ zu sehen. Anstelle der Kantenlängen $d_{node_i neighbor_j} = D(S_{node_i}, S_{neighbor_j})$ tritt eine Konstante D. Die Nachbarknoten $neighbor_j$ sollen also in der Nähe des Knoten $node_i$ liegen, wobei *„Nähe“* durch die Konstante D festgelegt ist, die den geforderten Abstand zum Knoten $node_i$ angibt. Für den **Music Browser** wird die Konstante mit $D = 2$ definiert. Schlussendlich gibt es für jede Kante vom Knoten $node_i$ zu seinen Nachbarn $neighbor_j$ noch ein Gewicht $w_{node_i,neighbor_j}$ das wie folgt definiert ist:

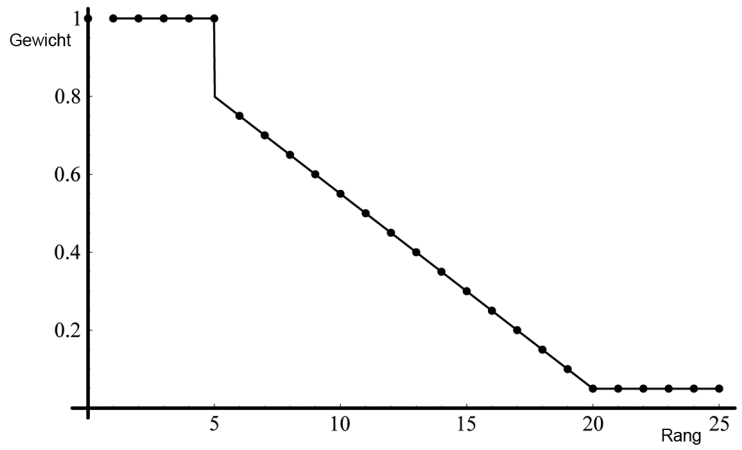


Abbildung 4.4: Darstellung des Gewichts $w_{node_i,neighbor_j}$ mit $K = 5$ und $E = 20$

$$w_{node_i,neighbor_j} = \begin{cases} 1 & \text{falls } \mathbf{rank}(neighbor_j) < K \\ \frac{E - \mathbf{rank}(neighbor_j)}{E} & \text{falls } K < \mathbf{rank}(neighbor_j) < E \\ 0.05 & \text{falls } \mathbf{rank}(neighbor_j) > E \end{cases} \tag{4.7}$$

Die Funktion **rank**($neighbor_j$) liefert den Rang des Knoten $neighbor_j$, wenn man alle Knoten ihrer Distanz zum Knoten $node_i$ nach ordnen würde, und kann leicht schon bei der Konstruktion des Ähnlichkeitsgraphen berechnet werden. Insgesamt definiert das Gewicht also, wie stark der Fehler einer Verbindung $abs(||\vec{p}_{neighbor_j} - \vec{p}_{node_i}|| - D)$ im Graphen beim Platzieren der Knoten in der Ebene berücksichtigt wird. Je größer der Rang ist, umso weniger wird der MDS Ansatz die Verbindung berücksichtigen. Man beachte, dass K die Ordnung des Ähnlichkeitsgraphen ist und damit die k-nächsten Nachbarn alle Gewicht 1 haben. Verbindungen höheren Rangs, die dadurch entstehen, dass die Verbindungen ungerichtet sind oder eingefügt wurden, damit der Ähnlichkeitsgraph zusammenhängend ist, verlieren mit steigendem Rang an Bedeutung. Ab Rang E, mit $E = 20$ für den **Music Browser**, werden alle Verbindungen gleichermaßen mit einem niedrigen Gewicht von 0.05 einbezogen (siehe Abbildung 4.4). Die *Loss*-Funktion, die den Gesamtfehler einer Konfiguration bestimmt, ist demnach wie folgt festgelegt:

$$Loss(P) = \sum_{i=0}^{N} e_{node_i} \tag{4.8}$$

Damit wurde eine modifizierte Bewertungsfunktion definiert. Auch zahlreiche andere Bewertungsfunktionen wurden im Rahmen der Implementierung des Prototyps getestet. Unter den getesteten Varianten hat sich die obig definierte Bewertungsfunktion anhand einfacher, qualitativer Evaluierungen — wie sie auch in Kapitel 5 durchgeführt werden — durchgesetzt.

4.2.3 Optimierungsalgorithmus

Der Optimierungsalgorithmus der im **Music Browser** zum Einsatz kommt, ist einem Gradientenabstiegsverfahren nachempfunden. Prinzipiell besteht der Algorithmus nur aus drei Schritten:

1. Platziere die Knoten des Graphen zufällig in der Ebene.

2. Korrigiere die Position jedes Knoten $node_i$ der Reihenfolge nach von Knoten $node_0$ bis Knoten $node_{N-1}$ ein wenig, sodass sich der Gesamtfehler reduziert.

3. Wiederhole Schritt 2, solange, bis es keinen Knoten mehr gibt, dessen Position man korrigieren kann, sodass sich der Gesamtfehler verringert.

Der Korrekturvektor c_i der Position p_{node_i} eines Knoten $node_i$ ist entsprechend der *Loss*-Funktion definiert.

$$\vec{c}_{node_i} = \sum_{j}^{|node_i|} \overbrace{\frac{\vec{p}_{node_i} - \vec{p}_{neighbor_j}}{||\vec{p}_{node_i} - \vec{p}_{neighbor_j}||}}^{\text{Richtung}} \overbrace{w_{node_i,neighbor_j}}^{\text{Gewicht}} \overbrace{(||\vec{p}_{node_i} - \vec{p}_{neighbor_j}|| - D)}^{\text{Fehler}} \quad (4.9)$$

Der Einfluss jedes Nachbarn auf den Korrekturvektor wird durch die **Richtung**, in die der Nachbar $neighbor_j$ den Knoten $node_i$ zieht, die Größe des durch ihn erzeugten **Fehlers** und sein **Gewicht** bestimmt. Um die Konvergenz des Algorithmus zu unterstützen, wird die Länge eines Korrekturvektors auf $l_{max} = 0.05$ beschränkt. Das hat zur Folge, dass sich die Position eines $\vec{p}_{node_i}$ Knotens $node_i$ pro Durchlauf nur geringfügig ändert.

$$\vec{c}_{node_i} = \begin{cases} \vec{c}_{node_i} & ||\vec{c}_{node_i}|| \leq l_{max} \\ \frac{l_{max}}{||\vec{c}_{node_i}||}\vec{c}_{node_i} & ||\vec{c}_{node_i}|| > l_{max} \end{cases} \quad (4.10)$$

Algorithmus 2 beschreibt den Optimierungsalgorithmus als Pseudocode. Wiederum werden einige Methoden definiert:

- **randomPoint()**

  Liefert einen zufälligen Punkt $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ in der Ebene mit $x_1, x_2 \in [-500, 500]$[3].

[3]Das um 0 symmetrische Intervall könnte prinzipiell beliebig gewählt werden (z.B.: $[-1, 1]$). Aus praktischen Gründen spezielle beim Debuggen, hat sich aber das Intervall $[-500, 500]$ durchgesetzt.

- **getError(node$_i$)**
  Diese Methode berechnet für den Knoten $node_i$ den Fehler e_{node_i} nach Formel (4.6).

- **getCorrection(node$_i$)**
  Berechnet den Korrekturvektor $\vec{c}_{node_i}$ nach Formel (4.9) und Formel (4.10);

Algorithm 2 Optimization Algorithm

```
1: unchangedSince ← 0
2: for i = 0 to N do                                   //place nodes randomly
3:    p⃗_node_i ← randomPoint()
4: end for
5: i ← 1
6: while unchangedSince < N do  //as long as we can improve the total error
7:    e_node_i ← getError(node_i)
8:    c⃗_node_i ← getCorrection(node_i)
9:    p⃗_node_i ← p⃗_node_i + c⃗_node_i          //change the position of the current node
10:   if e_node_i > getError(node_i) then        //have we yield an improvement?
11:      unchangedSince ← 0
12:   else                                                    //no improvement ...
13:      p⃗_node_i ← p⃗_node_i − c⃗_node_i          //... so revert the change of the position
14:      unchangedSince ← unchangedSince + 1
15:   end if
16:   if i == N+1 then          //a modulo counter specifying the node index
17:      i ← 1
18:   else
19:      i ← i + 1
20:   end if
21: end while
```

Ein interessanter Aspekt des eingesetzten Optimierungsalgorithmus ist, dass aufgrund der Tatsache, dass der Ähnlichkeitsgraph zusammenhängend ist, die einzelnen Subcluster nicht mehr lose, sondern verbunden sind. Die Verbindungen zwischen den Subclustern sorgen dafür, dass sich die Subcluster während des MDS

Verfahrens entsprechend den Verbindungen anziehen. Das wiederum bedeutet, dass jeder Subcluster vom Algorithmus in der Nähe von möglichst ähnlichen Subclustern entsprechend der Struktur im Gesamtgraphen platziert wird. Wäre der Graph nicht zusammenhängend, wäre damit auch keinerlei Ähnlichkeitsbeziehung zwischen den unterschiedlichen Teilgraphen definiert, und sie könnten unabhängig von einander in unterschiedliche Regionen der Ebene konvergieren. Ist die Distanz zwischen den Teilgraphen ein Vielfaches von deren Ausdehnung, so würde man in einer Überblickskarte die einzelnen Teilgraphen nur mehr als Punkte, oder sehr kleine Inseln, wahrnehmen. Dieser Effekt kann bei einem zusammenhängenden Graphen nicht auftreten. Nun fehlt nur mehr ein kleiner Schritt hin zu einer Übersichtskarte. Im Rahmen der Kartenerzeugung wird ausgehend von dem platzierten Ähnlichkeitsgraphen eine Musiklandschaft erzeugt.

4.2.4 Kartenerzeugung

Das Ergebnis des MDS Ansatzes ist eine Positionierung der Musiktitel auf einer Ebene. Ausgehend von dieser Punktmenge in der Ebene soll nun ein Karte, die Musiklandschaft, erzeugt werden. Dazu wird ein Verfahren mit dem Namen *Kernel Density Estimation* verwendet. Jedem der N Musiktitel, repräsentiert durch einen Punkt $\vec{p_i}$ in der Ebene wird eine so genannte *Kernel*-Funktion $K_i(\vec{x})$ zugewiesen. Durch das Aufsummieren der einzelnen *Kernel*-Funktionen erhält man eine Gesamtverteilung, die man für jeden Punkt $\vec{x}$ in der Ebene auswerten kann:

$$p(\vec{x}) = \frac{1}{N} \sum_{i=1}^{N} K_i(\vec{x}) \tag{4.11}$$

Bei der hier verwendeten *Kernel*-Funktion handelt es sich um eine 2-dimensionale Normalverteilung mit der Varianz σ^2 und dem Erwartungswert p_i.

$$K_i(\vec{x}) = \begin{cases} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2\sigma^2}(\vec{x}-\vec{p_i})^T(\vec{x}-\vec{p_i})} & \text{falls } \frac{\sqrt{(\vec{x}-\vec{p_i})^T(\vec{x}-\vec{p_i})}}{\sigma} < 3 \\ 0 & \text{sonst} \end{cases} \tag{4.12}$$

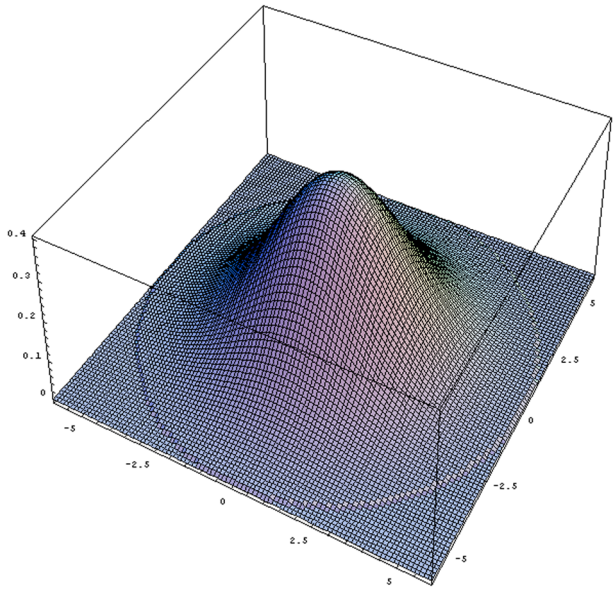


Abbildung 4.5: Eine zweidimensionale, begrenzte Normalverteilung

Eine *Kernel*-Funktion nach Gleichung (4.12) hat ab einem bestimmten Radius, abhängig von der Varianz, immer den Funktionswert 0. Durch diese Limitierung wird der Einfluss einer *Kernel*-Funktion auf einen bestimmten Radius reduziert, was den Rechenaufwand erheblich verringert, ohne das Ergebnis wesentlich zu beeinflussen, wenngleich es sich streng genommen nicht mehr um eine Verteilungsfunktion handelt, da das Volumen unter der *Kernel*-Funktion nun nicht mehr exakt 1 ist. Abbildung 4.5 zeigt die in (4.12) definierte *Kernel*-Funktion für $\sigma = 2$.

Die Gesamtverteilung, die man für jeden Punkt $\vec{x}$ in der Ebene auswerten kann, gibt somit eine Schätzung der Dichte. Stellt man nun die Ebene als Grafik dar, so kann man jedem Punkt der Grafik einen Farbwert, der proportional zur Dichte ist, zuweisen, und es entsteht eine Karte.

Abhängig von der für die *Kernel*-Funktionen verwendeten Standardabweichung ergeben sich unterschiedliche Karten. Abbildungen 4.6 und 4.7 zeigen die gleiche Konfiguration P eines Ähnlichkeitsgraphen, dargestellt als Karte mit unterschiedlichem Parameter σ.

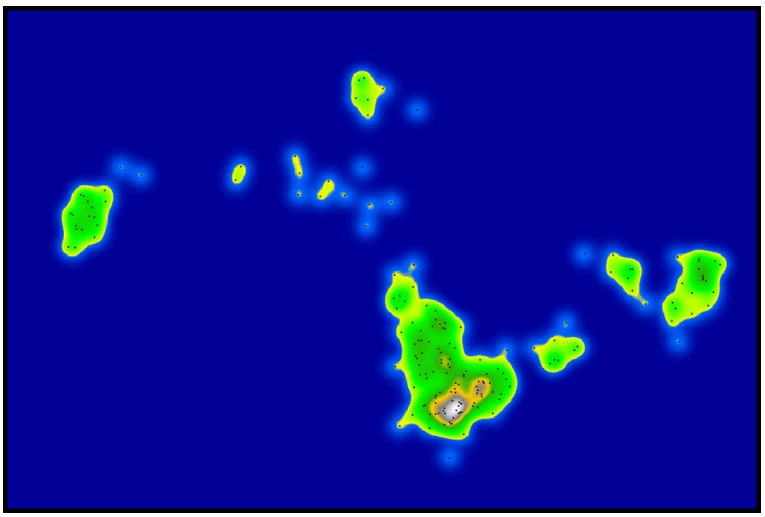


Abbildung 4.6: Globale Karte mit $\sigma \approx 10$

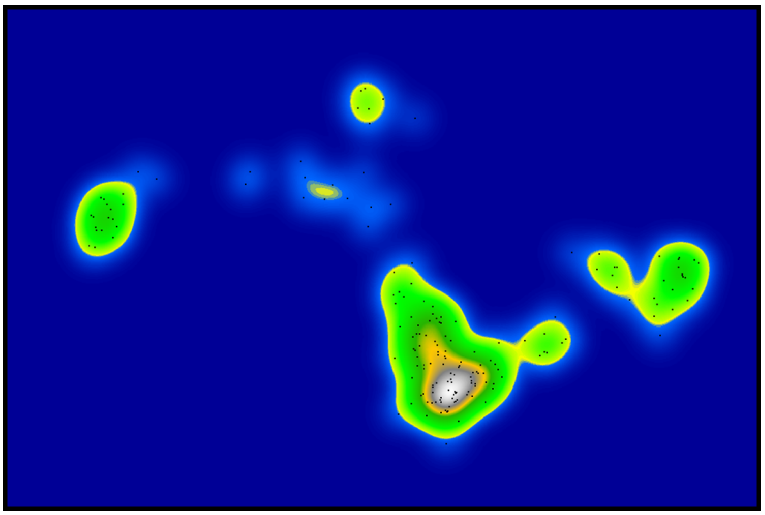


Abbildung 4.7: Globale Karte mit $\sigma \approx 20$

4.3 Lokale Navigation

Die lokale Navigation soll es ermöglichen, von einem ausgewählten Musiktitel ausgehend zu ähnlichen Musiktiteln zu gelangen. Um lokal navigieren zu können, benötigt man daher ein ausgewähltes Musikstück. Zu Beginn kann diese Auswahl im **Music Browser** sowohl über die globale Karte, als auch über die das Verwaltungsfenster für die Musiksammlung geschehen (siehe Abbildung 4.1). Entsprechend dem ausgewählten Musiktitel soll dann die lokale Nachbarschaft im Navigationsfenster zur weiteren Navigation angeboten werden. Abbildung 4.8 zeigt als ausgewählten Musiktitel *Avril Lavigne - Sk8er Boi* und dessen lokale Nachbarschaft. Jeder der im Navigationsfenster dargestellten Titel kann nun wiederum durch einen Mausklick ausgewählt werden, und wird so selbst zum Zentrum der Darstellung. Auf diese Art und Weise kann ausgehend vom aktuell ausgewählten Titel die gesamte Musiksammlung durchstöbert werden.

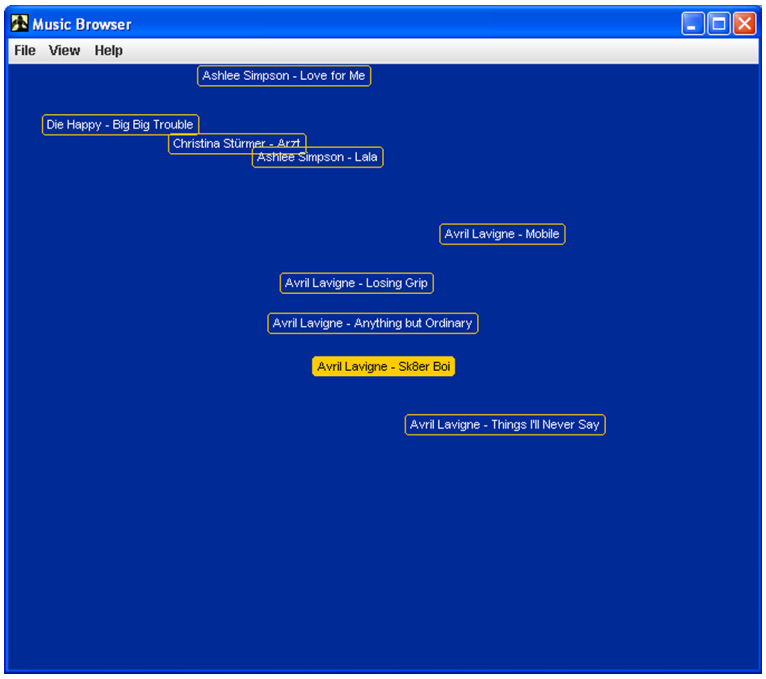


Abbildung 4.8: Darstellung der lokalen Nachbarschaft von *Avril Lavigne - Sk8er Boi*

Das Navigationsfenster sollte aber nicht nur eine Navigations- und Darstellungsmöglichkeit bieten, sondern gleichzeitg ein Detailausschnitt der globalen Karte sein und daher die Orientierung der globalen Karte beibehalten. Um die gewünschte Darstellung der lokalen Nachbarschaft mit entsprechender globaler Ausrichtung zu erhalten, wird in zwei Schritten vorgegangen. Zuerst wird aus dem Ähnlichkeitsgraphen, der als Gesamtmodell dient, ein Teilgraph ausgeschnitten, und zwar so, dass die Nachbarschaft des aktuell ausgewählten Musiktitels im Subgraphen enthalten ist. Anschließend wird dann der extrahierte Subgraph in einem Fenster mit entsprechender Ausrichtung visualisiert.

4.3.1 Extraktion der lokalen Nachbarschaft

Die Extraktion der lokalen Nachbarschaft LN gestaltet sich überraschend einfach. Für den aktuell ausgewählten Musiktitel — repräsentiert durch den Knoten $node_{cur}$ im Ähnlichkeitsgraphen — werden alle seine direkten Nachbarknoten $neighbor_j$ extrahiert und auch der Knoten $node_{cur}$ selbst. Zusätzlich wird noch ein oberes Limit L für die darzustellenden lokalen Nachbarn eingeführt. Wurden mehr als L Knoten extrahiert, so werden die ersten L Knoten entsprechend ihrem Rang $\mathbf{rank}(neighbor_j)$ ausgewählt und bilden geordnet nach ihrem Rang die Lokale Nachbarschaft $LN = LN_1, LN_2, ..., LN_{|LN|}$. Damit ist LN_1 jener Knoten, der $node_{cur}$ am nächsten ist und LN_2 der zweit nächste Knoten, usw. Nachdem die lokale Nachbarschaft nun ermittelt werden kann, muss sie nur mehr in einem Fenster visualisiert werden.

4.3.2 Darstellung der lokalen Nachbarschaft

Auch die Darstellung der lokalen Nachbarschaft ist relativ einfach. Das aktuell ausgewählte Musikstück $node_{cur}$ wird im Ursprung der lokalen Navigationsebene platziert. Um die Orientierung zu erhalten, werden die Richtungsvektoren $\vec{g}_i$ anhand der Positionen in der globalen Karte für alle Knoten in der lokalen Nachbarschaft bestimmt:

$$\forall_{i=1,\ldots,|LN|} : \vec{g}_i = \frac{\vec{p}_i - \vec{p}_{node_{cur}}}{||\vec{p}_i - \vec{p}_{node_{cur}}||} \tag{4.13}$$

Die lokale Position $\vec{lp}_i$ eines Musikstücks LN_i in der Navigationsebene erhält man, wenn der Richtungsvektor entsprechend der Reihung in der lokalen Nachbarschaft gestaucht wird:

$$\vec{lp}_i = \frac{|LN| - i}{|LN|}\vec{g}_i \tag{4.14}$$

Auf diese Art wird über den Richtungsvektor, der aus der globalen Karte berechnet wurde, die Orientierung erhalten, während die Distanz zum ausgewählten Musikstück $node_{cur}$ über den Rang innerhalb der lokalen Nachbarschaft einfließt.

Mit der Visualisierung der lokalen Nachbarschaft ist die Beschreibung der Algorithmen, die im **Music Browser** zum Einsatz kommen, komplett. Im nächsten Kapitel, *Qualitative Evaluierung*, werden die in dieser Arbeit vorgestellten inhaltsbasierten Ähnlichkeitsmetriken und der vorgeschlagene Navigationsansatz analysiert, und deren Schwächen und Stärken anhand von Beispielen illustriert.

5 Qualitative Evaluierung

Im Mittelpunkt dieses Kapitels steht die Analyse des vorgestellten Ansatzes. Es sollen Stärken und Schwächen des Verfahrens anhand von Beispielen aufgezeigt werden und Unterschiede zwischen den beiden vorgestellten inhaltsbasierten Ähnlichkeitsmetriken diskutiert werden. Darüber hinaus soll dem Leser die Funktionsweise des Systems anhand der Beispiele näher gebracht werden.

5.1 Datenbasis

Besonders wichtig für jede Evaluierung sind natürlich die Ausgangsdaten. Wünschenswert wäre eine Musiksammlung, die einige hundert Musiktitel unterschiedlicher Genres beinhaltet, so dass die Musiksammlung einen repräsentativen Ausschnitt des verfügbaren Musikangebots darstellt. Erstaunlich ist, dass in einem so aktiven Forschungsbereich, wie dem des *Music Information Retrievals*, derartige Datenbasen kaum verfügbar und nur schwer zugänglich sind. Dies hat vor allem lizenzrechtliche Gründe. Mangels Testdaten aus kommerziellen Sammlungen, bleibt nur der Ausweg über frei verfügbare Musiksammlungen. Die MAGNATUNE[1] Sammlung ist eine solche frei verfügbare Sammlung, die speziell für nicht-kommerzielle Studienprojekte lizenziert ist, und diese bietet neben einer Einteilung in Genres qualitativ ansprechende Musiktitel. Aus dieser Musiksammlung wurde eine Untermenge an Musiktiteln ausgewählt, indem von jedem Album zufällig ein Musiktitel in die Testsammlung (**Sammlung A**) aufgenommen wurde. Die insgesamt 6 Genres der MAGNATUNE Sammlung sind in der Testsammlung mit der in Tabelle

[1] http://www.magnatune.com

5.1 angegebenen Häufigkeit vertreten. Anhand der **Sammlung A** soll vor allem geprüft werden, wie gut die Struktur einer Musiksammlung durch die globale Karte erfasst werden kann. Im Optimalfall sollte man sechs abgegrenzte Regionen ausmachen können, die die einzelnen Genres repräsentieren. Die Anordnung der Regionen sollte dabei so sein, dass ähnliche Genres nahe beieinander liegen. Auch die Häufigkeit eines Genres sollte aus der globalen Karte abgelesen werden können.

Genre	Häufigkeit
Classic	66
Electronic	44
Jazz & Blues	9
Metal & Punk	19
Pop & Rock	34
World	34
Gesamt	216

Die zweite Testsammlung, **Sammlung B**, ist eine Privatsammlung mit einem Umfang von 1389 Musiktiteln. Hier sind keine exakten Genre-Einteilungen bekannt, aber die Sammlung ist, wie viele Privatsammlungen, relativ homogen und besteht im Wesentlichen aus den Genres *Rock & Pop*, *Dance*, *Electronic*, *Rap* und *Hip-Hop*, eben all jenen Genres, die weitläufig als *Mainstream* bezeichnet werden und fließend ineinander übergehen. Viele bekannte Künstler sind in dieser Sammlung enthalten, was das Auffinden korrekter und fehlerhafter Ähnlichkeitsbeziehungen erleichtert. Darüber hinaus sind Beispiele mit bekannten Künstlern wesentlich aussagekräftiger, da die meisten Menschen schnell eine Musikrichtung mit einem bekannten Künstler assoziieren können. Hinsichtlich der Struktur der globalen Karte ist keine klare Aufteilung in unterschiedliche Regionen zu erwarten, da die enthaltenen Genres stark verwandt sind und nicht klar getrennt werden können. Zu erwarten ist eine große zusammenhängende Musiklandschaft, innerhalb der es einen fließenden Übergang zwischen den enthaltenen Genres gibt.

Die hier skizzierten Idealvorstellungen sollen in Folge helfen, die Resultate des **Music Browser** zu bewerten und Probleme aufzudecken. In einem ersten Schritt wer-

den die unterschiedlichen Resultate abhängig vom verwendeten Ähnlichkeitsmaß (siehe Kapitel 3) untersucht und Stärken und Schwächen aufgezeigt.

5.2 Vergleich der Ähnlichkeitsmetriken

In Kapitel 3 wurden zwei inhaltsbasierte Ähnlichkeitsmetriken — *Fluctuation Patterns* und *Timbre Distribution* — vorgestellt, die beide als Basis für das Navigationsmodell des **Music Browsers** eingesetzt werden können. Das verwendete Ähnlichkeitsmaß hat folglich einen entscheidenden Einfluss auf das Navigationssystem, bestehend aus dem lokaler Navigationsfenster und der globalen Überblickskarte. Anders herum betrachtet, erlaubt ein Vergleich der Navigationseigenschaften des Systems auch Rückschlüsse auf die Qualität des Ähnlichkeitsmaßes. Einige auffällige Unterschiede des Systems abhängig vom Ähnlichkeitsmaß werden nun beschrieben.

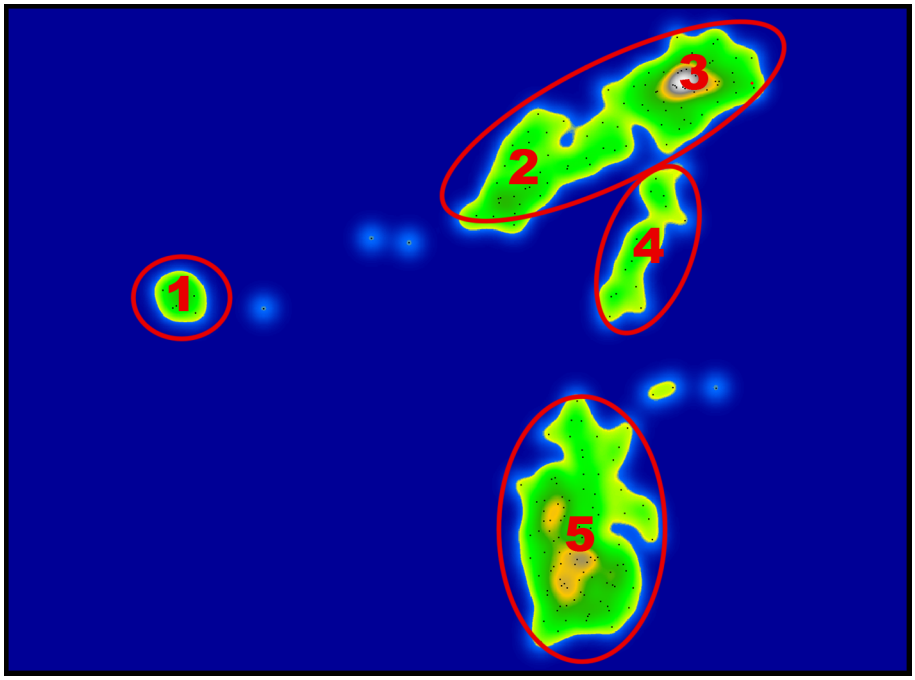


Abbildung 5.1: **Sammlung A**: Darstellung basierend auf den *Fluctuation Pattern*

Als erstes wird die Struktur der globalen Karte untersucht. Abbildung 5.1 zeigt eine Karte der **Sammlung A**, der das Ähnlichkeitsmaß *Fluctuation Patterns* zugrunde liegt. Für die **Sammlung A** sind sechs Genres definiert, die in der Überblickskarte erkennbar sein sollten. Die Überblickskarte in Abbildung 5.1 ist in fünf sinnvoll unterscheidbare Regionen eingeteilt. Interessant ist, dass die fünf Regionen nicht den einzelnen Genres entsprechen. Obwohl *Region 1* keinem Genre zugeordnet werden kann, bildet diese Region eine klar erkennbare homogene Gruppe. Im Wesentlichen handelt es sich um Musikstücke, die nahezu keine rhythmische Struktur aufweisen, sehr leise sind und mystisch klingen. Damit hat das Ähnlichkeitsmaß eine interessante Gruppe von Musiktiteln entdeckt, die zwar nicht durch ein Genre definiert ist, aber einen wesentlichen Strukturaspekt der Musiksammlung darstellt. Das Genre *Classic* verteilt sich auf *Region 2* und *Region 3*, wobei ein Übergang von eher langsamen Stücken (*Region 2*) zu schnelleren Stücken (*Region 3*) beobachtet werden kann. *Region 4* ist eher eine heterogene Gruppe. Eventuell könnte man das Genre *World* zuordnen — eindeutig ist diese Zuordnung jedoch nicht. Die restlichen Genres (*Rock & Pop*, *Metal & Punk*, *Jazz & Bues* und *Electronic*) finden sich in *Region 5*. Eine sinnvolle Strukturierung der Region ist nicht wirklich auszumachen.

Zum direkten Vergleich wurde auch mit dem Ähnlichkeitsmaß *Timbre Distribution* eine Überblickskarte für **Sammlung A** generiert. Auf den ersten Blick fällt auf, dass die einzelnen Regionen nicht so klar abgegrenzt sind, wie man es von den *Fluctuation Patterns* gewohnt ist. *Region 1* kann man klar dem Genre *Punk & Metal* zuordnen, und diese geht über in *Region 2* vom Genre *Electronic*. *Region 3* ist die heterogenste Region. Hier findet man vorwiegend Musiktitel aus den Genres *Rock & Pop*, *Jazz & Blues*, sowie *World*. Dabei lässt sich eine Strukturierung ausgehend von *Pop & Rock* oben bis hin zu *Jazz & Blues* unten beobachten. Im Hinblick auf das Genre *World* ist zu beobachten, dass es offensichtlich nicht klar von anderen Genres getrennt werden kann. So ist dieses Genre zwar vermehrt in der *Region 3* anzutreffen, aber Titel dieses Genres findet man auf der gesamten Karte verstreut. Die *Region 4* und die *Region 5* beinhalten klassische Musiktitel und umschließen gemeinsam die *Punk & Metal* Region, was dadruch erklärt werden kann, dass in *Region 4* vor allem klassische Stücke mit Zupfinstrumenten beheimatet sind.

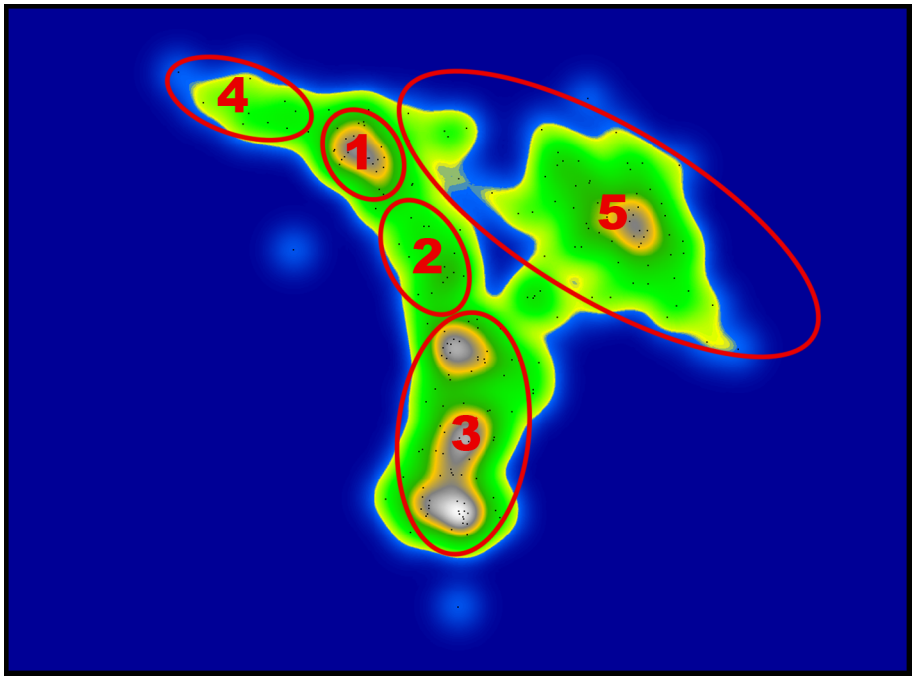


Abbildung 5.2: **Sammlung A**: Darstellung basierend auf dem Ähnlichkeitsmaß *Timbre Distribution*

Im Mittelpunkt des ersten Vergleichs steht die Frage, in wie weit ein Ähnlichkeitsmaß in der Lage ist, einzelne Untergruppen einer Musiksammlung zu trennen, was anhand von **Sammlung A** überprüft wurde. Im direkten Vergleich der Ähnlichkeitsmetriken fallen folgende Unterschiede auf:

- Grundlegend kann durch die *Fluctuation Patterns* hauptsächlich das Genre *Classic* von den anderen Genres getrennt und sogar eine sprechende Strukturierung dieses Genres erreicht werden. Eine klare Trennung der anderen Genres scheint jedoch nicht wirklich möglich.
- Das *Timbre Distribution* Verfahren kann immerhin vier der sechs Genres trennen. Auch die fließenden Übergänge machen Sinn und verbinden eher verwandte Genres.
- Überraschender Weise wurde durch *Fluctuation Patterns* eine sehr interes-

sante, klar abgegrenzte Gruppe entdeckt, die ganz offensichtlich eine Struktureigenschaft der **Sammlung A** sein sollte. Durch das *Timbre Distribution* Verfahren konnte diese Gruppe nicht entdeckt werden.

Insgesamt betrachtet ist das *Timbre Distribution* Verfahren hinsichtlich der Erkennung der globalen Struktur einer Musiksammlung dem *Fluctuation Pattern* Verfahren vorzuziehen. Nun soll aber noch analysiert werden, wie gut die Strukturierung einer homogenen Sammlung, wie es die **Sammlung B** ist, abhängig vom Ähnlichkeitsmaß gelingt. Dazu wurde wiederum sowohl für die *Fluctuation Patterns* (siehe Abbildung 5.3) als auch für das *Timbre Distribution* Verfahren (siehe Abbildung 5.4) eine globale Karte generiert.

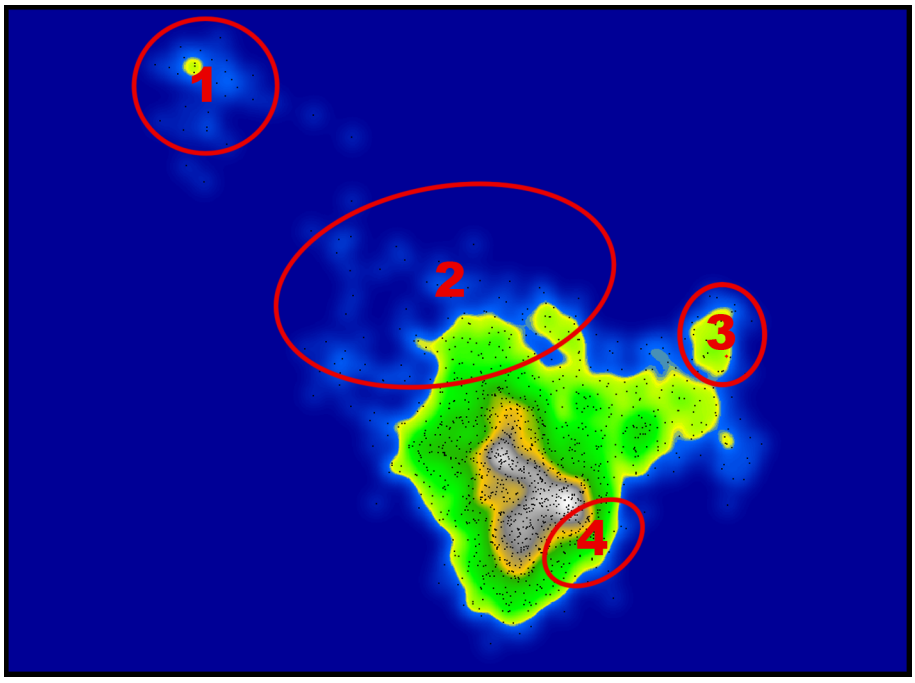


Abbildung 5.3: **Sammlung B**: Darstellung basierend auf dem Ähnlichkeitsmaß *Fluctuation Pattern*

Dabei stellt man fest, dass beide Verfahren die globale Struktur, wie erwartet, als eine zusammenhängende Struktur erfassen. Gerade bei den *Fluctuation Patterns*

lassen sich aber interessante Regionen (siehe Abbildung 5.3) innerhalb der Struktur feststellen. Wiederum lässt sich eine überraschend klar abgegrenzte Gruppe in *Region 1* identifizieren. Musiktitel mit starkem Bass und einer starken rhythmischen Struktur wie etwa *DJ Ötzi - Do Wah Diddy Diddy* oder *Vengaboys - We Like to Party!* finden sich hier. *Region 2* deckt das Genre *Dance* und *Electronic* ab. *Region 3* hingegen erfasst ebenfalls Titel, die eine klare rhythmische Struktur ausweisen, aber dem Genre *Rap* bzw. *Hip-Hop* zugeordnet werden können. Interpreten wie *Eminem* oder *50 Cent* sind hier anzutreffen. Offensichtlich werden durch die *Fluctuation Patterns* Musiktitel mit einer klaren rhythmischen Struktur gut erkannt, und es entsteht eine zum Teil recht klar erkennbare Ordnung innerhalb der homogenen Sammlung.

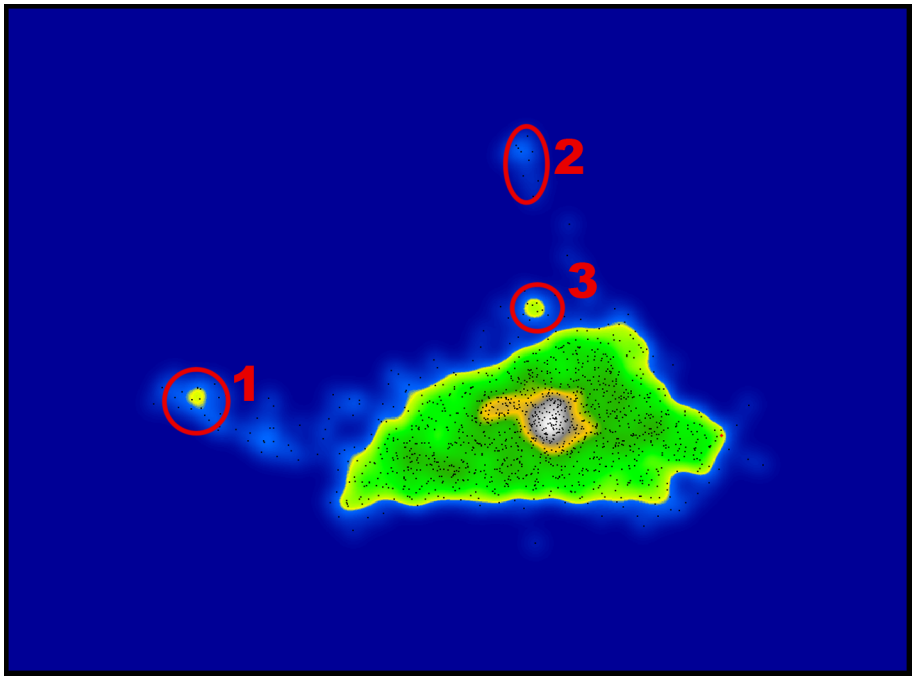


Abbildung 5.4: **Sammlung B**: Darstellung basierend auf dem Ähnlichkeitsmaß *Timbre Distribution*

Im Gegensatz dazu ist eine derartige Strukturierung bei dem *Timbre Distribution* Verfahren nicht zu erkennen. Den in Abbildung 5.4 eingezeichneten Regionen kann hier eindeutig ein bestimmter Interpret zugeordnet werden. So befinden sich etwa

in *Region 1* Lieder von *Bon Jovi*, in *Region 2* Lieder von *ABBA* und in *Region 3* Lieder von den *Flippers*. Auch im lokalen Navigationsfenster bemerkt man oft diese Gruppierung nach Interpreten, während eine derartige Häufung eines einzigen Interpreten im lokalen Navigationsfenster bei den *Fluctuation Patterns* nie zu beobachten ist. Abbildung 5.5 zeigt das lokale Navigationsfenster für den Titel *ABBA - Chiquitita*. Dieser Effekt lässt sich auch leicht erklären, denn die einzelnen Titel eines Interpreten sind oft äußerst ähnlich. Dennoch bilden sich kaum grobe Strukturen vergleichbar zu den *Fluctuation Patterns* aus.

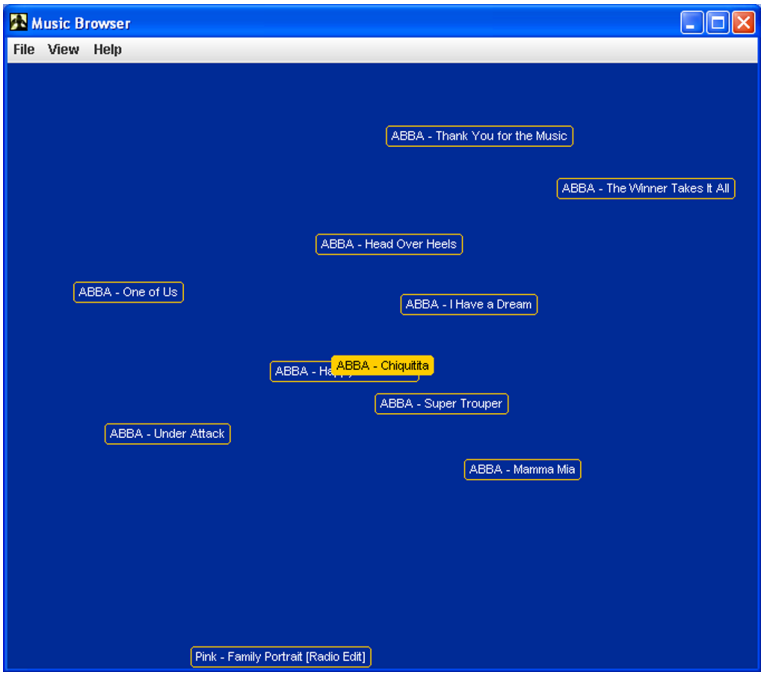


Abbildung 5.5: **Sammlung B**: Navigationsfenster mit *ABBA - Chiquitita* als aktuell ausgewähltem Musikstück

Homogene Sammlungen werden anscheinend durch die *Fluctuation Patterns*, sofern Titel mit klaren rhythmischen Strukturen vorhanden sind, besser organisiert, während das *Timbre Distribution* Verfahren Untergruppen, die durch einzelne Interpreten gebildet werden, besser identifizieren kann.

Im Endeffekt ist es schwierig zu beurteilen, welches der beiden Ähnlichkeitsmaße

nun das qualitativ Bessere ist. Ein Grund dafür dürfte sein, dass die zwei Ähnlichkeitsmaße versuchen, unterschiedliche Dimensionen der musikalischen Ähnlichkeit zu quantifizieren und damit auch für unterschiedliche Anwendungsgebiete besser bzw. schlechter geeignet sind. Deutlich wird jedoch, dass keines der beiden Verfahren optimal für die Anwendung in einem Navigationssystem für Musiksammlungen geeignet ist. Erst durch eine Verbesserung der Qualität der Ähnlichkeitsbewertung, zum Beispiel durch Kombination der Verfahren oder durch Erweiterung um Informationen aus dem Web, werden Navigationssysteme ähnlich dem **Music Browser** alltagstauglich machen. Der nächste Abschnitt konzentriert sich nun auf die Evaluierung des Navigationsansatzes.

5.3 Evaluierung des Navigationsansatzes

Die Beurteilung des Navigationsansatzes ist problematisch. Wie im letzten Abschnitt erörtert wurde, hat die zugrunde liegende Ähnlichkeitsmetrik einen maßgeblichen Einfluss auf das Navigationssystem. Einige Eigenschaften des Navigationssystems sind jedoch auch inhärent mit dem Navigationsansatz verknüpft. Genau jene Eigenschaften werden untersucht und mit Beispielen illustriert.

Als erstes ist festzustellen, ob der vorgestellte MDS Ansatz prinzipiell in der Lage ist, die Struktur des hochdimensionalen Ähnlichkeitsgraphen auf eine Ebene zu projizieren, sodass die wichtigen Struktureigenschaften des hochdimensionalen Graphen erhalten bleiben. Um diese Frage zu beantworten, wurden weitere Überblickskarten der **Sammlung A** erzeugt. Abbildung 5.6 zeigt eine davon, und man kann leicht die fünf in Abbildung 5.1 erkennbaren Regionen — die Nummerierungen beider Abbildungen sind ident — auch in Abbildung 5.6 ausmachen. Zahlreiche Wiederholungen zeigen, dass immer eine derartige Gruppierung zu beobachten ist. Speziell *Region 1* aus Abbildung 5.1 ist eindeutig eine kompakte zusammengehörige Gruppe. Die klar abgegrenzte Darstellung dieser Gruppe in allen generierten Karten weist darauf hin, dass das modifizierte MDS Verfahren tatsächlich die Struktur des hochdimensionalen Ähnlichkeitsgraphen in einer Ebene reproduzieren kann. Diese Beobachtung bestätigt, dass auch der modifizierte

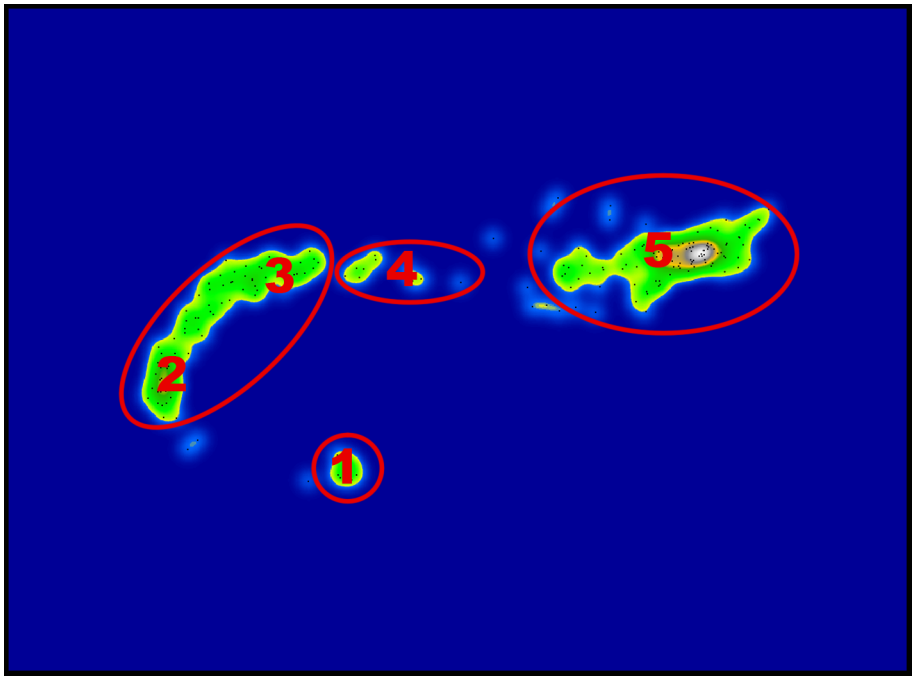


Abbildung 5.6: **Sammlung A**: Darstellung basierend auf dem Ähnlichkeitsmaß *Fluctuation Pattern*

MDS Ansatz in der Lage ist, eine strukturerhaltende Projektion hochdimensionaler Daten auf eine Ebene durchzuführen.

Nachdem plausibel argumentiert werden konnte, dass der MDS Ansatz prinzipiell funktioniert, sollen nun aber auch Probleme dieses Ansatzes diskutiert werden. Wie aus Abschnitt 4.2 bekannt ist, ist eine Dimensionsreduktion immer mit Fehlern verbunden. Da stellt sich natürlich automatisch die Frage, ob und wie sich diese Fehler beim dem vorgeschlagenen Navigationsansatz auswirken.

Tatsächlich machen sich die Fehler der Dimensionsreduktion bei der Navigation bemerkbar, indem man gelegentlich während der lokalen Navigation größere *„Sprünge“* in der globalen Karte beobachten kann. Ein Beispiel dieses unerwünschten Verhaltens gibt Abbildung 5.7. Obwohl Titel *Scooter - Shake That!* und *Daddy DJ - Daddy DJ* im hochdimensionalen Raum eng bei einander liegen, kann diese Nachbarschaftsbeziehung in der Ebene nicht korrekt wiedergegeben werden, da

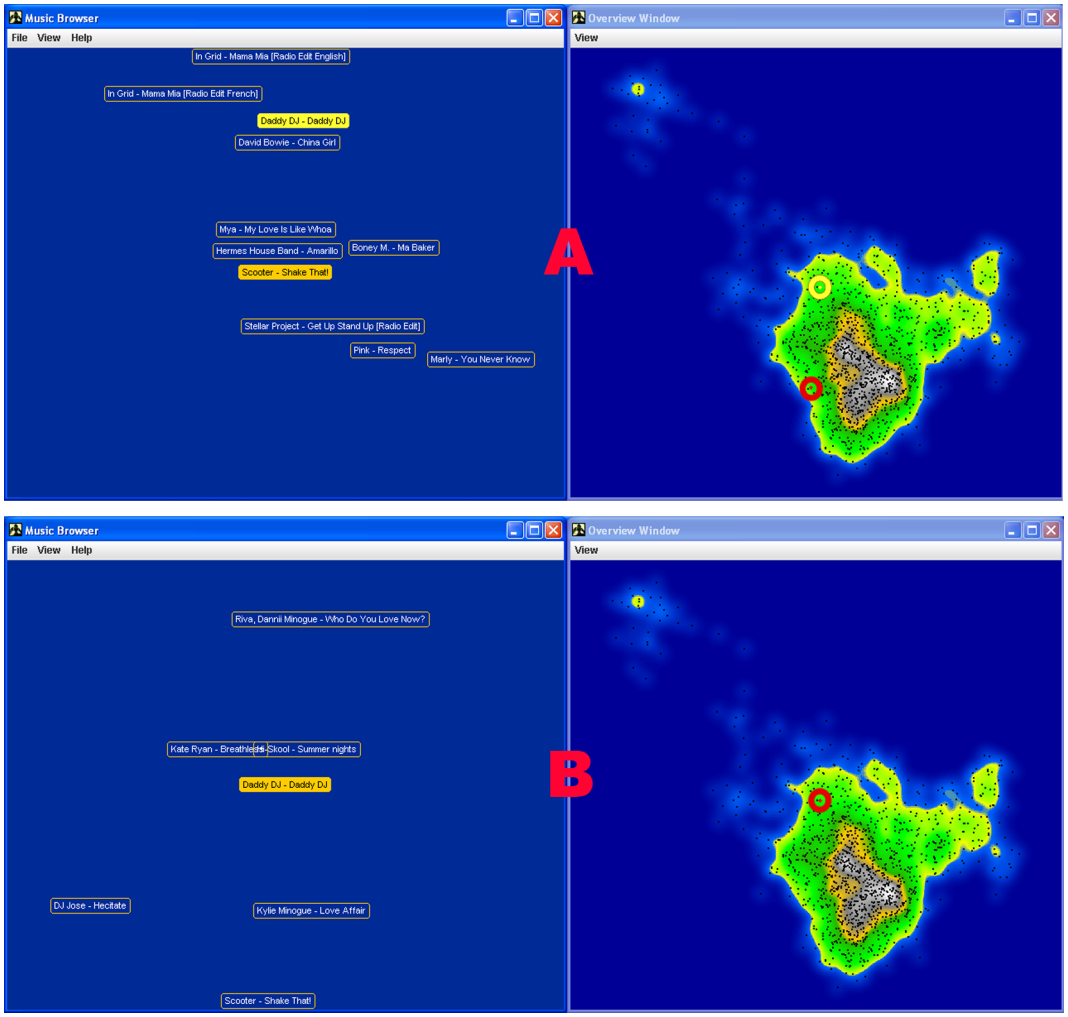


Abbildung 5.7: Veranschaulichung eines *„Sprungs“* in der globalen Karte während eines Navigationsvorgangs

die anderen Nachbarschaftsbeziehungen eine Platzierung in unterschiedlichen Regionen fordern. Das Resultat ist ein großer Sprung in der globalen Karte, obwohl im Navigationsfenster eigentlich zu einem lokalen Nachbarn navigiert wurde, von dem man erwarten würde, dass er auf der globalen Karte in der Nähe des aktuell ausgewählten Titels liegt. Abbildung 5.7 zeigt in der **Darstellung A** die lokale Nachbarschaft von *Scooter - Shake That!* mit dem Titel *Daddy DJ - Daddy DJ* als Navigationsziel. Gleichzeitig sind aktuelle Position und Ziel auch in der globalen

Karte durch Markierungen hervorgehoben. Die Distanz zwischen den beiden Titeln ist für Nachbarn doch erheblich. **Darstellung B** zeigt das Navigationssystem, nachdem der Navigationsschritt durchgeführt wurde.

Leider lässt sich das Problem der Sprünge nicht lösen und ist inhärent mit jeder Art der Dimensionsreduktion verbunden. Die einzige Möglichkeit solche Sprünge zu vermeiden, ist Ähnlichkeitsbeziehungen, die zu Sprüngen führen zu ignorieren, also im lokalen Navigationsfenster nicht darzustellen. Dies wiederum bedeutet aber auf der einen Seite, dass vorhandene Ähnlichkeitsbeziehungen nicht mehr dargestellt werden, und auf der anderen Seite, dass damit auch die Navigation eingeschränkt wird und eventuell Musiktitel nicht mehr erreichbar sind. Abhilfe für dieses Problem kann nur über eine Verbesserung der Qualität der Ähnlichkeitsmetriken erfolgen, denn es ist in Wahrheit nicht besonders plausibel, dass es Ähnlichkeitsbeziehungen zwischen Musiktiteln aus völlig unterschiedlichen Regionen gibt. Wahrscheinlicher ist, dass es sich schlicht um eine falsch bestimmte Ähnlichkeitsbeziehung handelt.

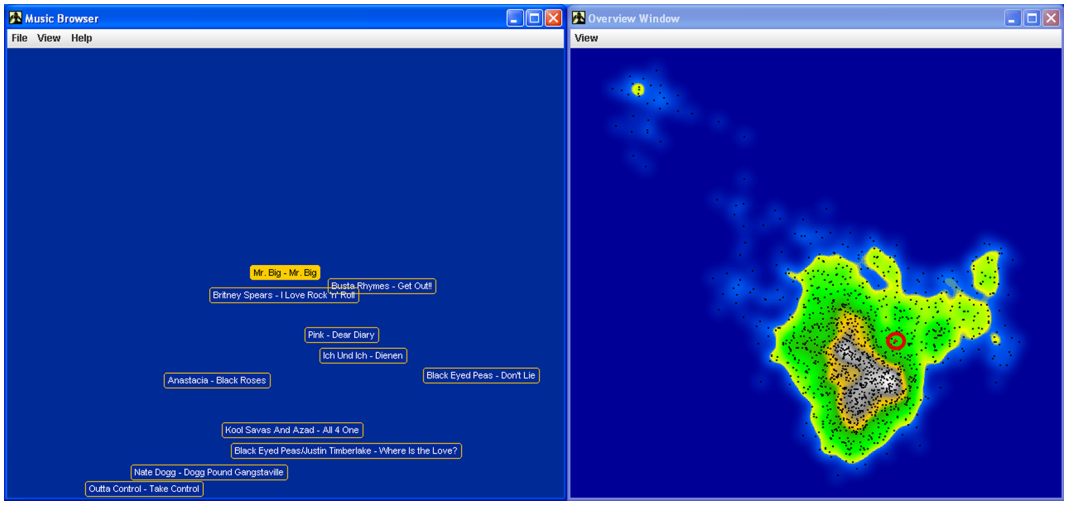


Abbildung 5.8: Eine Einschränkung des Navigationssystems: Eine Navigation in nördliche Richtung ist in dieser Situation nicht möglich.

Neben den Sprüngen gibt es noch ein zweites Phänomen, das im Zusammenhang mit dem vorgeschlagenen Navigationsansatz auftreten kann. So ist es mitunter

möglich, dass man, während man im lokalen Navigationsfenster die Musiksammlung erforscht, an einen Punkt gelangt, der es nicht mehr zulässt, in eine bestimmte Richtung weiter zu gehen, obwohl in der globalen Karte in dieser Richtung noch weitere Musiktitel eingezeichnet sind. Ein Beispiel für eine solche Navigationssituation ist in Abbildung 5.8 dargestellt. Ausgehend vom aktuellen Musiktitel *Mr. Big - Mr. Big* ist es nicht möglich, weiter in nördliche Richtung zu wandern, obwohl sich offensichtlich in der globalen Karte nördlich des gewählten Titels noch weitere Musikstücke befinden. Auf den ersten Blick ist dieses Phänomen erstaunlich, lässt sich aber dank der Kartenmetapher schnell erklären. Man stelle sich eine Situation in einer Straßenkarte vor, bei der der Reisende von einem kleinen *Dorf A* zum nördlich gelegenen Nachbardorf *Dorf B* gelangen will. Nun gibt es leider zwischen *Dorf A* und *Dorf B* keine direkte Straßenverbindung. Der Reisende ist damit gezwungen einen Umweg zu fahren, zum Beispiel zuerst etwas westlich und dann erst nördlich, bevor er zum *Dorf B* gelangen kann. Die gleiche Situation kann auch innerhalb der Musiklandschaft auftreten. Es kann vorkommen, dass man von einem Musiktitel aus nicht direkt weiter in eine gewünschte Richtung navigieren kann, sondern einen Umweg über andere Musiktitel nehmen muss. Für den Benutzer ist dieses Verhalten aber nicht intuitiv nachvollziehbar, da in der Musiklandschaft ja keine „Straßen“ — also die Ähnlichkeitsbeziehungen der Musiktitel — eingezeichnet sind. Zum Abschluss des Kapitels wird auf ein letztes kleines Manko des Systems verwiesen, das mit der Farbwahrnehmung des Menschen in Zusammenhang steht.

In Abschnitt 4.2.4 wurde erklärt, wie die globale Karte durch *Kernel Density Estimation* entsteht. Dabei wird jedem Punkt der Karte ein Farbwert entsprechend der Verteilungsdichte der Musiktitel zugewiesen. Das bedeutet, dass proportional der Dichte ein entsprechender Wert einer Farbpalette verwendet wird. Die verwendete *Island*-Farbpalette sorgt dafür, dass man die schönen Inselstrukturen im Meer wahrnehmen kann. Obwohl diese Farbpalette wunderschön anzusehen ist, ist sie aus technischer Sicht nicht besonders gut geeignet, denn der Mensch nimmt den Farbverlauf der *Island*-Farbpalette stark nicht-linear mit der Größe des Farbwerts wahr. Dies führt zu einem falschen Eindruck der tatsächlichen Dichte.

Abbildung 5.9 zeigt die gleiche globale Karte wie Abbildung 5.1, nur mit einer

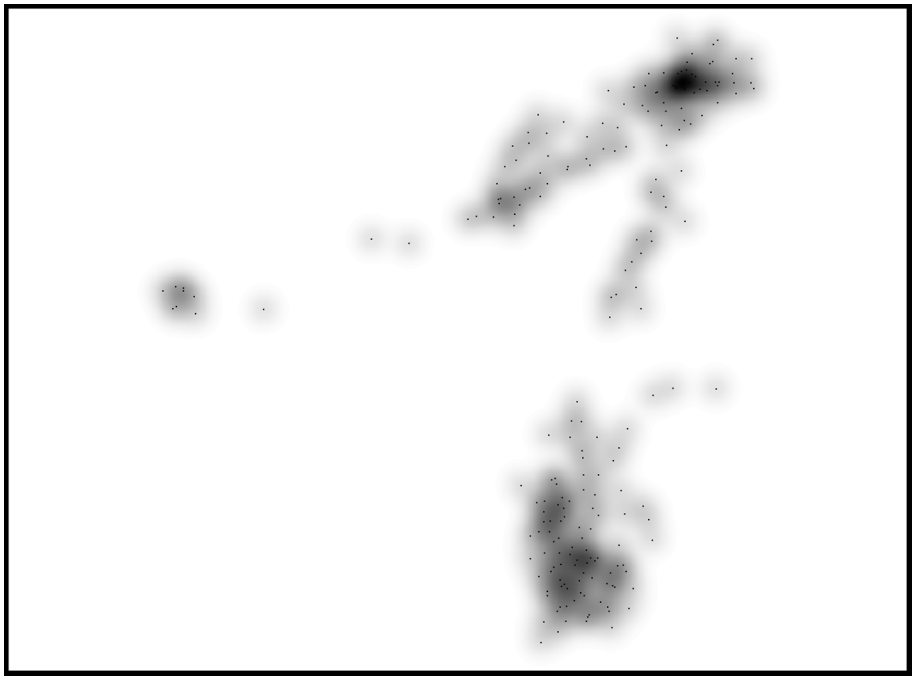


Abbildung 5.9: Einfluss der Farbpalette (vgl. Abbildung 5.1)

grauen Farbpalette, die einen linearen Farbverlauf aufweist. Speziell die Abgrenzung der *Region 3* von *Region 4* in Abbildung 5.1 kann man in Abbildung 5.9 nicht mehr so einfach ausmachen. Dennoch ist festzuhalten, dass die nicht-lineare Farbwahrnehmung nicht unbedingt negativ zu bewerten ist. Schließlich kann nicht definitiv entschieden werden, ob der Einfluss der *Island*-Farbpalette hilfreich ist und die menschliche Wahrnehmung einfach beim Auffinden der Gruppierungen unterstützt, oder ob durch diese Farbpalette das Ergebnis schlicht verfälscht wird. Zumindest sollte man sich bei einer Analyse bewusst sein, dass auch die Farbpalette das Ergebnis beeinflusst. [2]

Mit der Diskussion über die nicht-lineare Farbwahrnehmung schließt dieses Kapitel über die qualitative Evaluierung ab. Der Vergleich der Ähnlichkeitsmaße hat kein eindeutiges Ergebnis zu Gunsten eines Ähnlichkeitsmaßes geliefert, und die Analyse des Navigationssystems hat dessen prinzipielle Tauglichkeit bestätigt, aber auch

[2] Daher bietet der **Music Browser** auch die Möglichkeit, die Insellandschaft auszublenden, so dass nur die Platzierung der Musiktitel in der Ebene dargestellt wird.

Probleme des Ansatzes skizziert. Das letzte Kapitel dieser Arbeit konzentriert sich nun auf eine Zusammenfassung der durch den Prototyp gewonnenen Erkenntnisse und gibt zum Abschluss noch einen Ausblick auf Verbesserungsmöglichkeiten und Perspektiven für zukünftige Forschungen.

6 Schlussfolgerungen und Perspektiven

Die Zielsetzung dieser Arbeit war es, einen Prototyp zu entwickeln, der ein neuartiges Navigations- und Visualisierungsmodell umsetzt und dabei die Anwendung inhaltbasierter Ähnlichkeitsmetriken demonstriert. Im Speziellen sollte der Prototyp dazu dienen zu evaluieren, ob der grundlegende Ansatz der *mehrstufigen Navigation* für die Navigation in Musiksammlungen Sinn macht, oder eben nicht. Im Rahmen dieser Arbeit wurde daher auch eine Prüfung der Brauchbarkeit dieses Konzepts anhand des Prototyps durchgeführt. Der nächste Abschnitt fasst kurz die Ergebnisse dieser Analyse zusammen.

6.1 Benutzertest

In einer ersten Evaluierungsphase haben mehrere Testpersonen aus meinem persönlichen Umfeld den **Music Browser** ausprobiert und wurden dabei von mir eingewiesen und beobachtet. Obwohl diese Art der Evaluierung natürlich sehr subjektiv ist, kann man doch schon erste Erkenntnisse aus den Rückmeldungen ziehen. Die Rückmeldungen der Testpersonen lassen sich in einigen wesentlichen Punkten zusammenfassen:

- Grundsätzlich waren alle Benutzer in der Lage, innerhalb kürzester Zeit das Navigationssystem zu bedienen. Der Zusammenhang zwischen der lokalen und der globalen Karte war allen Testnutzern sehr schnell klar.

- Ein Hauptkritikpunkt der Testnutzer war, dass im lokalen Navigationsfenster Musiktitel, die ihrer Meinung nach offensichtlich in keiner Ähnlichkeitsbeziehung zueinander stehen, gemeinsam dargestellt wurden. Dies deutet auf Unzulänglichkeiten bei der automatischen Bestimmung von Ähnichkeiten zwischen Musiktiteln hin.

- Einigen Benutzern fiel auf, dass das Navigieren in der lokalen Karte mitunter zu Sprüngen in der globalen Karte führen kann. Dieses überraschende Verhalten irritiert offensichtlich manche Nutzer, ist aber aus technischer Sicht natürlich kaum zu vermeiden, da nicht alle Zusammenhänge der hochdimensionalen Daten korrekt in einer Ebene wiedergegeben werden können.

- Überraschend war, dass einige Testnutzer mit Vorliebe in der globalen Karte herumspielten, um einen Eindruck der unterschiedlichen Regionen zu bekommen. Das ist insbesondere interessant, da eigentlich zu Beginn der Arbeit nicht geplant war, dass man die globale Karte auch zum Navigieren verwenden kann. Benutzer finden offensichtlich derartige Visualisierungen in Kombination mit interaktiven Elementen besonders reizvoll.

Ergänzend muss man noch festhalten, dass alle Nutzer grundlegend von der Idee eines Navigationssystems für Musiksammlungen begeistert waren und sich vorstellen konnten, ein solches System in Zukunft einzusetzten.

6.2 Schlussfolgerungen

Vor allem aus der Beobachtung, dass die Testnutzer das Navigationssystem in sehr kurzer Zeit begriffen haben, kann man ableiten, dass der *mehrstufige Navigationsansatz* durchaus ein möglicher Lösungsweg ist, um Navigation in großen Musiksammlungen zu realisieren. Speziell die Übersichtskarte spricht offensichtlich den Spieltrieb vieler Nutzer an, und fordert geradezu heraus, die verschiedenen Regionen der Musiksammlung zu erkunden.

Weniger zufriedenstellend sind die Rückmeldungen, die die Qualität der automatischen Ähnlichkeitsbestimmung betreffen. Gerade kleine Sammlungen neigen natürlich aufgrund der Tatsache, dass einfach zu manchen Titeln keine ähnlichen Musiktitel in der Sammlung vorhanden sind, dazu, falsche Ähnlichkeitsbeziehungen zu fördern. Auf der anderen Seite zeigen manche Darstellungen im lokalen Navigationsfenster aber auch klar die Unzulänglichkeiten der Ähnlichkeitsmetriken auf. So ist unter den 5 ähnlichsten Musiktiteln zu *The Offspring - Defy You* kein einziger Titel aus den Genres *Rock* bzw. *Hard Rock* zu finden, was schlichtweg nur als unzulänglich bezeichnet werden kann. Insgesamt betrachtet muss man leider eingestehen, dass die hier vorgestellten inhaltsbasierten Ähnlichkeitsmetriken weder qualitativ noch von der Rechengeschwindigkeit her praxistauglich sind. Weitere aktuelle Forschungen in diesem Bereich [25] versprechen aber Verbesserungen und geben damit die Hoffnung, dass paxistaugliche inhaltsbasierte Ähnlichkeitsmetriken schon in naher Zukunft verfügbar sind.

Im Hinblick auf irritierende Sprünge in der globalen Karte beim lokalen Navigieren kann man nur feststellen, dass offensichtlich der MDS Ansatz nicht optimal zur Projektion eines hochdimensionalen Ähnlichkeitsgraphen auf eine Ebene geeignet ist. Ein wesentlicher Nachteil des MDS Ansatzes gerade im Bezug auf den Ähnlichkeitsgraphen ist, dass ähnliche Musiktitel zwar in eine Region gezogen werden, nicht als ähnlich klassifizierte Titel aus der Region aber nicht verdrängt werden. So können zwei an sich nicht ähnliche Subcluster in ein und derselben Region platziert werden, was natürlich nicht wünschenswert ist. Hier gibt es Raum für Verbesserungen, die in Folge noch kurz angeschnitten werden sollen.

6.3 Perspektiven

Wie durch den Prototyp deutlich wird, ist insbesondere der aktuelle MDS Ansatz zur Darstellung des Graphen in der Ebene verbesserungswürdig. Der Nachteil des MDS Ansatzes wird noch deutlicher, wenn man bedenkt, dass das Verfahren nicht besonders effizient hinsichtlich der Laufzeit ist. Daher wird vorgeschlagen, grundlegend auch andere Verfahren (zum Beispiel *Self-Organizing Maps* basierte

Ansätze) zu prüfen, die distanzbasierte Daten in einen niedrigdimensionalen Raum projizieren können. Neben dem Vorschlag das MDS Verfahren zu ersetzen, bleibt natürlich auch die Möglichkeit offen, das MDS Verfahren zu verbessern. So könnte die zufällige Platzierung der Knoten zu Beginn des MDS Algorithmus durch eine informierte Initialisierung ersetzt werden, was den Optimierungsaufwand erheblich reduzieren würde. Zusätzlich können unterschiedliche Optimierungsalgorithmen, wie etwa **Tabu-Suche** und **Simulated Annealing** getestet werden. Eventuell macht es Sinn, eine diskrete Version des MDS Ansatzes zu entwickeln, bei der es möglich ist, die Qualität einer Lösung auch nach dem Auftreten von nicht als ähnlich klassifizierten Knoten in der lokalen Nachbarschaft zu bewerten.

Allerdings ist der MDS Ansatz nicht die einzige Schwäche des Testsystems. Auch die verwendeten inhaltsbasierten Ähnlichkeitsmetriken haben sich als problematisch erwiesen. So können zwar prinzipiell interessante Ähnlichkeitsbeziehungen extrahiert werden, aber damit geht leider auch eine Vielzahl von inkorrekten Ähnlichkeitsbeziehungen einher. Die Verbesserung inhaltsbasierter Ähnlichkeitsmetriken ist ein aktuelles Forschungsthema, mit dem sich viele Forschungsgruppen befassen. Dennoch scheint eine qualitative Verbesserung derartiger Verfahren problematisch. Das Hauptproblem im Zusammenhang mit dem Ähnlichkeitsgraphen ist, dass falsche Ähnlichkeitsbeziehungen unterschiedlichste Musiktitel aus unterschiedlichen Teilbereichen des Graphen verbinden und damit die Dimensionalität des Graphen erheblich wächst, was wiederum zu Problemen bei der Dimensionsreduktion führt. Daher der Vorschlag, falsche Ähnlichkeitsbeziehungen zu erkennen und zu eliminieren. Erkennen könnte man falsche Ähnlichkeitsbeziehungen eventuell anhand der Struktur des Graphen. Aber auch anhand der Distanzmatrix könnte man falsche Ähnlichkeitsbeziehungen erkennen. Nur jene Distanzen die zum Beispiel kleiner als 95% der Distanzen in der gesamten Distanzmatrix sind, könnte man als gültige Ähnlichkeitsbeziehungen akzeptieren. Unabhängig von den angeführten Beispielen ist leicht zu erkennen, dass es im Bereich inhaltsbasierter Ähnlichkeitsmetriken noch viel Spielraum und Anlass für weitere Forschungen gibt.

Gerade durch Verbesserungen in diesen zwei Bereichen könnte man den Prototyp entscheidend weiterentwickeln, aber auch andere Möglichkeiten der Realisie-

rung von mehrstufigen Navigationssystemen sollten ins Kalkül gezogen werden. Insgesamt bleibt der Bereich des *Music Information Retrievals* ein bewegtes und interessantes Forschungsfeld, und man darf auf weitere Ergebnisse in dieser Forschungsrichtung gespannt sein.

Abbildungsverzeichnis

7 Literaturverzeichnis

[1] J. J. Aucouturier, F. Pachet. Music Similarity Measures: What's the Use?, In *Proc. of the 3rd International Symposium on Music Information Retrieval*, 2002.

[2] J. J. Aucouturier, F. Pachet. Finding Songs That Sound the Same, In *Proc. 1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA-2002)*, 2002.

[3] J. J. Aucouturier, F. Pachet. Improving Timbre Similarity: How high's the sky?, In *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[4] K. Backhaus, B. Erichson, W. Plinke, R. Weiber. *Multivariate Analysemethoden*, Springer-Verlag, 2003.

[5] R. Bladon, B. Lindblom. Modeling the judgment of vowel quality differences. In *The Journal of the Acoustical Society of America*, 69(5):1414-22, 1981.

[6] I. Borg, P. J. F. Groenen. *Modern Multidimensional Scaling Theory and Applications*. Springer-Verlag, 1997.

[7] C. R. Cave. *Perceptual Modelling for Low-Rate Audio Coding*. Master thesis at the McGill University, 2002.

[8] T. F. Cox, M. A. A. Cox. *Multidimensional Scaling (CRC Monographs on Statistics & Applied Probability)*, 2nd Edition, Chapman & Hall, 2000.

[9] D. P. W. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *Proc. International Conference on Music Information Retrieval (ISMIR)*, 2002.

[10] H. Fastl. Fluctuation Strength and Temporal Masking Patterns of Amplitude-Modulated Broad-Band Noise. In *Hearing Research 8*, 59–69, 1982.

[11] D. Gleich, M. Rasmussen, K. Lang, L. Zhukov. The World of Music: SDP layout of high dimensional data, Interactive Poster at the *IEEE Symposium on Information Visualization (InfoVis 2005)*, 2005.

[12] G. Karypis, E. H. Han, and V. Kumar. CHAMELEON: A Hierarchical Clustering Algorithm using Dynamic Modeling, In *IEEE Computer: Special Issue on Data Analysis and Mining*, 32(8):68–75, 1999.

[13] J. A. Lee, A. Lendasse, M. Verleysen. Curvilinear Distance Analysis versus Isomap. In *Proceedings of the 10th European Symposium on Artificial Neural Networks (ESANN'2002)*, 85-192, 2002.

[14] T. Lidy, A. Rauber. Evaluation of Feature Extractors and Psycho-Acoustic Transformations for Music Genre Classification. In *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR'05)*, London, 34-41, 2005.

[15] S. Lippens, J.P. Martens, and T. De Mulder. A comparison of human and automatic musical genre classification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages iv–233– iv–236, 2004.

[16] A.A. Livshin, X. Rodet. Musical Instrument Identification in Continuous Recordings. In *Proc. of the 7th Int. Conference on Digital Audio Effects*, Naples, Italy, October 5-8, 2004.

[17] S.Y. Kung, S.H. Lin, M.W. Mak. *Biometric Authentication: A Machine Learning Approach*. Prentice Hall, 2004.

[18] M. I. Mandel, D. P.W. Ellis. Song-Level Features and Support Vector Machines for Music Classification. In *Proc.of the 6th International Conference on Music Information Retrieval (ISMIR 05)*, London, 2005.

[19] A. V. Oppenheim, R. W. Schafer. *Zeitdiskrete Signalverarbeitung*, R. Oldenbourg Verlag, 2.Auflage, 1995.

[20] T. Painter, A. Spanias. Perceptual Coding of Digital Audio. In *Proc. of the IEEE*, Vol.88, No. 4, 2000.

[21] E. Pampalk, A. Rauber, D. Merkl. Using Psycho-Acoustic Models and Self-Organizing Maps to Create a Hierarchical Structuring of Music by Sound Similarity. In *Proc. Int. Symposium on Music Information Retrieval* (ISMIR), Paris, France, 2002.

[22] E. Pampalk, S. Dixon, G. Widmer. Exploring Music Collections by Browsing Different Views. In *Proc. of the 4th International Conference on Music Information Retrieval (ISMIR'03)*, Washington, D.C., USA, October 2003.

[23] E. Pampalk, S. Dixon, G. Wimer. On The Evaluation of Perceptual Similarity Measures for Music. In *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*, London, 2003.

[24] E. Pampalk. *Tutorial: Music Similarity*. Presentation at the ISMIR 05, London, 2005. Available online [21.06.05]:

`http://ismir2005.ismir.net/documents/pampalk_ismir_05_similarity_tutorial_c.pdf`

[25] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Technische Universität Wien, 2006.

[26] G. Peeters. A large set of audio features for sound description (similarity and classification) in the CUIDADO project[1], 2003.

[1] http://recherche.ircam.fr/equipes/analyse-synthese/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf

[27] J. C. Platt. Fast embedding of sparse music similarity graphs. In *Advances in Neural Information Processing Systems*, volume 16, pages 571–578, 2004.

[28] M. Schedl, P. Knees, T. Pohle, G. Widmer. Towards Automatic Retrieval of Album Covers, In *Proceedings of the 28th European Conference on Information Retrieval (ECIR'06)*, 2006.

[29] M.R. Schroeder, B.S. Atal, J.L. Hall. Optimizing digital speech coders by exploiting masking properties of the human ear. *Journal of the Acoustical Society of America*, 66:1647-1652, 1979.

[30] D. Schwarz. *Spectral Envelopes in Sound Analysis and Synthesis*, Diplomarbeit am Institut de la Recherche et Coordination Acoustique/Musique (IRCAM) und am Institut für Informatik (Universität Stuttgart), 1998.

[31] D. Schwarz, X. Rodet. Spectral Estimaion and Representation for Sound Analysis-Synthesis. In *Proc. ICMC*, 1999.

[32] B. A. Shenoi. *Introduction to digital signal procesing and filter design.* Wiley-Interscience, 2005.

[33] E. Skovenborg, S. H. Nielsen. Evaluation of Different Loudness Models with Music and Speech Material, In *Proc. of the 117th AES convention*, San Francisco, 2004.

[34] K. Steiglitz. *Digital Signal Processing Primer - With Applications to Digital Audio and Computer Music.* Addison Wesley, 1996.

[35] T. Thiede. *Perceptual Audio Quality Assessment using a Non-Linear Filter Bank.* PhD thesis, Technischen, Universität Berlin, 1999.

[36] S. V. Vaseghi. *Advanced Digital Signal Processing and Noise Reduction.* John Wiley & Sons, Second Edition, 2000.

[37] Wikipedia - *Die freie Enzyklopädie.* Available online [21.06.05]:

`http://de.wikipedia.org`

[38] E. Zwicker, H. Fastl. *Psychoacoustics: Facts and Models*. Springer, 1999.

Klaus Seyerlehner

Staatsangehörigkeit: Österreich

Geburtsdatum: 2. Oktober 1981

Geburtsort: Linz

Ausbildung

- 1988 – 1990 Öffentliche Volksschule 9 in Linz (4020)
- 1990 – 1992 Öffentliche Volksschule 48 in Linz (4020)
- 1992 – 2000 2.Bundesgymnasium Linz, Khevenhüllerstr.1
- 2000 - 2001 Wehrdienst
- **Matura** mit **gutem Erfolg** bestanden (Fachbereichsarbeit aus Informatik)
- Führerscheine **A** und **B**
- Studium der **Informatik** seit September 2001

„Ich erkläre an Eides statt, dass ich die vorliegende Diplom- bzw. Magisterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.“

Linz, am ....