Beiträge zum Entwurf und Vergleich von robusten Audiomerkmalen für das Musikretrieval

Diplomarbeit

Sebastian Kreuzer

Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III

11. Februar 2008

Inhaltsverzeichnis

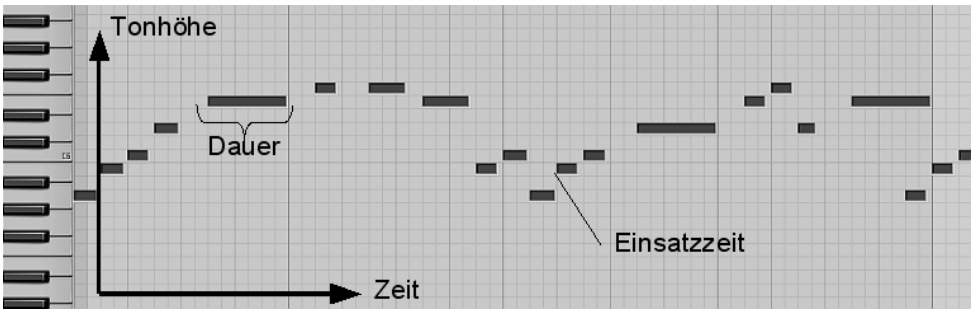
Kapitel 1

Einleitung und Überblick

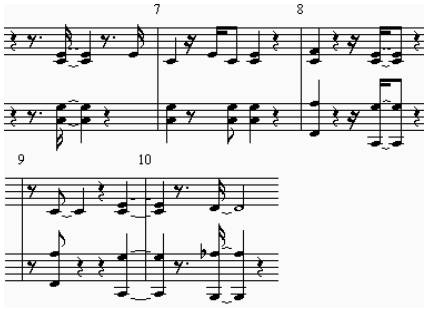
Beim *Audiomatching* handelt es sich um eine Aufgabe des inhaltsbasierten Suchens in Musikdatenbanken. Inhaltsbasiert bedeutet, dass direkt auf Rohdaten gearbeitet wird und keine vorher manuell hinzugefügten Zusatzinformationen oder Metadaten vorliegen. Die Suchaufgabe ist nach dem *„Query By Example" (QBE)*-Prinzip gestellt. Das heißt, es sollen zu einer gegebenen Anfrage, bei der es sich selbst um einen kurzen Abschnitt Musik in einem bestimmten Format handelt, ähnliche Abschnitte in der Musikdatenbank gefunden werden. Die gefundenen Abschnitte sollen sich jedoch auch in gewissen Eigenschaften von der Anfrage unterscheiden können, da in der Datenbank zum einen verschiedene Interpretationen eines Stückes enthalten sein können, die sich in Lautstärke, Tempo, Tonhöhe, Artikulation oder Besetzung unterscheiden. Zum anderen kann auch innerhalb eines Stückes ein Thema oder eine Passage mehrfach in modifizierter Form auftreten. Dennoch sollen all diese Variationen des Themas aus der Anfrage gefunden werden. Beethovens Fünfte Sinfonie enthält beispielsweise eine Wiederholung der Exposition und in der Reprise erscheint das Thema in abgewandelter Form zum dritten mal. Liegt nun als Anfrage ein typischerweise etwa 20 Sekunden langer Abschnitt, der das erste Auftreten des Themas enthält, vor, so sollen neben der exakt gleichen Passage auch die beiden abgewandelten Wiederholungen gefunden werden. Zusätzlich sollen diese Stellen auch in anderen Interpretationen als ähnlich erkannt werden. Von Beethovens Fünfter Sinfonie existiert z.B. eine Klaviertranskription von F. Liszt, in der die Themen auch dann gefunden werden sollten, wenn es sich bei der Anfrage um einen Abschnitt der Orchesterversion handelt.

1.1 Darstellungsformen von Musik

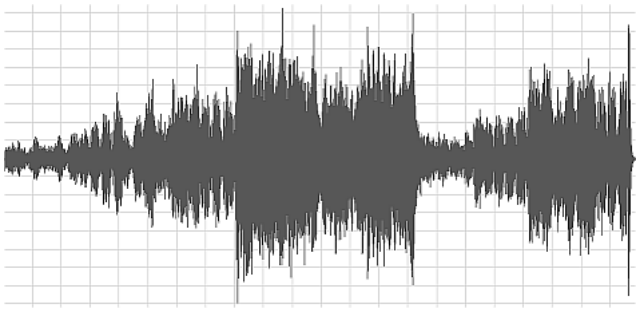
Musikstücke können in verschiedenen Formen gegeben sein, so z.B. aufgeschrieben als Partitur, im MIDI-Format oder auf Audio CDs. In der Partiturdarstellung, siehe 1.1(b), wird in Form von Noten, Pausen, Punktierungen und verschiedenen weiteren Symbolen die inhaltliche Abfolge eines Musikstückes für die einzelnen Instrumente wiedergegeben. Die symbolische Schreibweise kodiert Tonhöhen, Längen und Lautstärken, jedoch nur als Orientierung für die Musiker. Es wird Freiheit für Interpretationen gewährt, sodass sich ein



(a) MIDI-Daten in Pianorolldarstellung



(b) Ausschnitt einer Partitur



(c) PCM-Daten

Abbildung 1.1. verschiedene Darstellungsformen von Musik: 1.1(a) - MIDI-Datei in *Pianorolldarstellung*. Zu erkennen sind Einsatzzeit und Dauer jeder Note. Die Tonhöhe wird durch die Klaviertastatur am linken Rand dargestellt. 1.1(b) - Musik als Spielanweisung kodiert mit verschiedenen Symbolen für Tonhöhe, Länge, Pausen... 1.1(c) - PCM-Daten speichern Informationen des physikalischen Auftretens der Musik als Schallsignal.

Stück, wenn es von verschiedenen Personen oder auch mehrmals von einer Person gespielt wird, jedesmal anders anhört. Die Partiturdarstellung ist also inhaltlich orientiert jedoch unpräzise.

Beim *MIDI-Format (Musical Instrument Digital Interface)* werden für alle im Stück vorkommenden Instrumente Einsatzzeit, Tonhöhe, Tondauer und evtl. andere Parameter der gespielten Noten abgespeichert, wie in der Grafik 1.1(a) zu erkennen ist. Insbesondere enthält das Format, wie bei der Partitur, direkte Informationen über die Tonhöhen bzw. Noten und damit über die Melodie, im Gegensatz zur Partiturdarstellung ist das Musikstück jedoch sehr genau beschrieben. Weiteres zum MIDI-Format findet man in [17].

Diese Arbeit befasst sich jedoch ausschließlich mit Musik, die im *PCM-Format (Puls-Code-Modulation)* vorliegt, das auch bei Audio-CDs verwendet wird und daher weit verbreitet ist. Das PCM-Format ist an der physikalischen Erscheinung der Musik als Audiosignal orientiert. Abbildung 1.1(c) zeigt ein im PCM-Format gespeichertes Audiosignal.

Man erkennt, dass es sich um eine Schwingung handelt, die sich physikalisch z.B. als Spannungs- oder Luftdruckschwankung auswirkt. Im PCM-Format wird der Verlauf dieser Schwingung abgespeichert. Ein Musikstück wird auf diese Weise sehr genau dargestellt, jedoch sind keine direkten Informationen über enthaltene Noten und Melodien mehr vorhanden. Hier wird die Schwierigkeit des inhaltsbasierten Suchens ohne Metainformationen klar.

1.2 Grundsätzliches Vorgehen

Die Idee zur Lösung des Matchingproblems besteht darin, sowohl aus allen Stücken der Datenbank, als auch aus der Anfrage gewisse *Merkmale* zu extrahieren, auf deren Basis ein Vergleich von Anfrage und Datenbank erfolgt und die Auswahl der *Treffer*, also der als ähnlich erachteten Abschnitte, stattfindet. Die Merkmale müssen in gewisser Weise robust gegenüber bestimmten Abweichungen wie Lautstärke, Tempo und Klangfarbe sein. Das „in gewisser Weise“ wird hier erstmal nicht genauer spezifiziert, sondern soll im Verlauf dieser Arbeit näher behandelt werden. Neben einem geeigneten Algorithmus für das Vergleichen und das Auswählen der Treffer hängt nämlich der Erfolg des Matchingverfahrens und vor allem die Art der Treffer, die man erhält, hauptsächlich von den Merkmalen und deren Eigenschaften ab. In dieser Arbeit werden verschiedene Merkmale mit diversen Parametern anhand ihrer Matchingergebnisse bezüglich bestimmter Datenbanken und Anfragen verglichen.

1.3 Gliederung der Arbeit

In Kapitel 2 wird zunächst der Algorithmus vorgestellt, der bei allen Experimenten zum Einsatz kommt, um die Merkmale der Anfrage und der Musikdatenbank zu vergleichen. Das *Dynamic Time Warping* hat sich hier bewährt (siehe [10]), da es akzeptable Laufzeiteigenschaften aufweist und einen recht flexiblen und gegenüber zeitlichen Schwankungen robusten Vergleich von zeitabhängigen Folgen erlaubt. Es wird die Erzeugung der sogenannten *Matchingkurve* beschrieben, welche in den folgenden Kapiteln zum Vergleich und zur Erklärung von Eigenschaften verschiedener Merkmale dient.

Ab Kapitel 3 beginnt der Hauptteil, in dem verschiedene Merkmale im Detail vorgestellt und untersucht werden. Dabei wird auf deren Extraktion und verschiedene Parameter und Möglichkeiten der Modifikation eingegangen und ihre Eigenschaften in Bezug auf das Audiomatching untersucht. Auch zwischen den Merkmalen werden Vergleiche gezogen. Die Behandlung der Merkmale ist in drei Kapitel unterteilt. Kapitel 3 befasst sich mit solchen, die mittels einer Filterbank extrahiert werden. Dazu gehören die bekannten Pitch- und Chromamerkmale, die auf einer Zerlegung des Signals in Subbänder basieren, welche den einzelnen MIDI-Noten entsprechen. Desweiteren werden CENS-Merkmale, die aus einer statistischen Vergröberung der Chromamerkmale resultieren, betrachtet.

Kapitel 4 behandelt Merkmale, die mit Hilfe einer gefensterten Fouriertransformati-

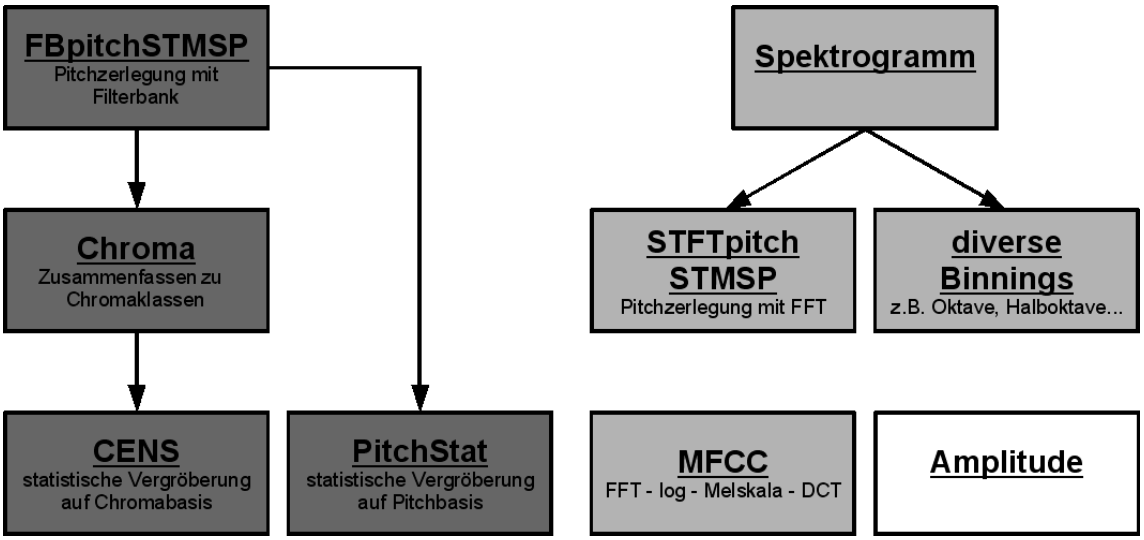


Abbildung 1.2. Übersicht der Merkmale, dunkelgrau: Filterbank basiert - hellgrau: durch Fouriertransformation - weiß: sonstige Merkmale

on generiert werden. Hier werden zunächst Spektralvektoren direkt als Merkmalsvektoren zum Audiomatching eingesetzt. Darauf folgt die Untersuchung von Approximationen der filterbankbasierten Merkmale durch die gefensterte Fouriertransformation. Neben der Nachbildung der Filterbänke werden auch einige andere Zusammenfassungen von Fourierkoeffizienten durchgeführt und ihre Eigenschaften als Merkmale untersucht, dazu gehören Oktav- und Halboktavzerlegungen. Desweiteren werden feinere Unterteilungen erzeugt und die Entwicklung des Ergebnis bei zunehmender Verfeinerung beobachtet.

Kapitel 5 befasst sich mit den ursprünglich aus der Sprachsignalverarbeitung stammenden MFCC-Merkmalen (von engl. *Mel Frequency Cepstral Coefficients*). Da diese Merkmale genauer untersucht werden, werden sie in einem eigenen Kapitel behandelt, obwohl sie auch zu den STFT-Merkmalen gehören. Es werden verschiedene Einsatzgebiete dieser Merkmale vorgestellt und die Matchingergebnisse unterschiedlicher Modifikationen der MFCC-Merkmale untersucht. Je nach Auswahl der Koeffizienten wird sich herausstellen, dass das Ergebnis sehr stark von der Amplitude beeinflusst ist. Um diese Tatsache näher zu untersuchen, werden spezielle Amplitudenmerkmale eingeführt. Am Ende des Kapitels erfolgt ein Vergleich von CENS-, MFCC- und Oktavmerkmalen anhand ihrer Treffer auf einer größeren Datenbank. Abbildung 1.2 zeigt eine Übersicht der behandelten Merkmale und deren Zusammenhang.

Im Anhang ist eine Auflistung der MATLAB-Funktionen zu finden, mit deren Hilfe die einzelnen vorgestellten Merkmale aus den Audiodaten extrahiert worden sind. Die einzelnen Parameter sind dort knapp kommentiert, da die Funktionsweise in der Beschreibung der entsprechenden Merkmale genau erläutert wird.

Kapitel 2

Das allgemeine Matchingverfahren

Ziel des Audiomatching ist, in einer Datenbank, die eine Kollektion von mehreren Musikstücken enthält, alle Passagen, die zu einer gegebenen Anfrage ähnlich sind, aufzufinden. Dazu werden die Stücke der Datenbank und die Anfrage in Folgen von Merkmalsvektoren umgewandelt. Jeder dieser Vektoren bezieht sich auf einen zeitlichen Abschnitt eines Musikstückes und hat die Aufgabe, gewisse charakteristische Eigenschaften dieser Passage hervorzuheben, gleichzeitig jedoch von anderen für das Matching störend wirkenden Eigenschaften zu abstrahieren. So sollen beispielsweise Abschnitte gefunden werden, die einer Passage der Datenbank ähnlich sind, sich jedoch in Parametern wie Lautstärke, Klangfarbe, Tempo oder Artikulation unterscheiden.

Typische Verfahren zum Audiomatching gehen in zwei Schritten vor: Zuerst werden Merkmale aus Datenbank und Anfrage extrahiert. Dabei geschieht die Merkmalsextraktion aus der Datenbank offline; die Anfrage wird, sobald sie vorliegt, online umgewandelt. Nachdem die benötigten Merkmale erzeugt worden sind, werden alle Merkmalsvektoren in der Datenbank durchlaufen, um sämtliche Abschnitte der enthaltenen Musikstücke mit der Merkmalsfolge der Anfrage zu vergleichen. Gegenüber leichten lokalen Temposchwankungen sind einige Merkmale invariant. Jedoch können bedingt durch ein anderes Grundtempo eines Stückes oder stärkere lokale Tempoänderungen die Anzahl von Vektoren in Anfrage und zugehöriger gewünschter Fundstelle voneinander abweichen. Da ein Merkmalsvektor im Audiomatching bei den meisten Merkmalstypen nur eine Sekunde oder weniger abdeckt, tritt dieser Fall sehr häufig ein. Man sucht also ein Verfahren, das auch unterschiedlich lange Merkmalsfolgen miteinander vergleichen kann und auf diese Weise Abschnitte in der Datenbank findet, die zur Anfrage ähnlich sind, aber andere Geschwindigkeiten oder einen anderen internen Geschwindigkeitsverlauf aufweisen.

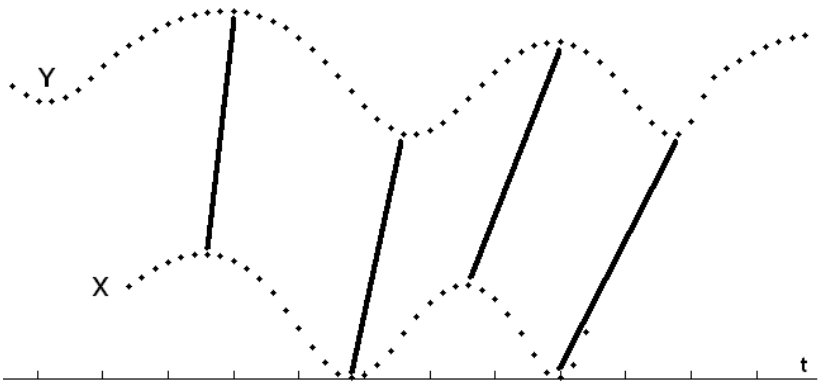


Abbildung 2.1. Darstellung zweier reellwertiger Folgen: Die Folge Y enthält einen Abschnitt, der ähnlich zur Folge X ist, sich jedoch in der Länge unterscheidet.

In Abschnitt 2.1 wird eine Variante des DTW-Algorithmus vorgestellt. Dieser Algorithmus wird in der gesamten Arbeit genutzt, um den Vergleich der Merkmalsfolge der Anfrage mit der Musikdatenbank vorzunehmen. Um Merkmale miteinander vergleichen zu können, benötigt der Algorithmus ein Abstandsmaß, welches jedem Paar von Merkmalsvektoren bestimmte Kosten zuordnet. Das eingesetzte Abstandsmaß wird in Abschnitt 2.2 behandelt. Schließlich wird in Abschnitt 2.3 die sogenannte *Matchingkurve* erklärt, die man als Ergebnis des DTW-Algorithmus erhält und beschrieben, wie die Treffer anhand der Kurve bestimmt werden.

2.1 Dynamic Time Warping

Der Algorithmus, der zum Vergleichen der Merkmalsfolgen eingesetzt wird, ist eine Variante des *Dynamic Time Warping (DTW)*, das sogenannte *Teilfolgen DTW*. Sei dazu $\mathcal{F}$ die Menge aller möglicher Merkmalsvektoren, die beim verwendeten Merkmalstyp auftreten können. Alle in der Datenbank enthaltenen Musikstücke werden nun in Folgen von Merkmalsvektoren umgewandelt und diese konkateniert, sodass man eine Sequenz $Y = (y_1, y_2, \ldots, y_M)$ der Länge M erhält, wobei $y_m \in \mathcal{F}$ für alle $m \in [1 : M]$ gilt. Die Start- und Endindizes der einzelnen Stücke werden in einer zusätzlichen Datenstruktur gespeichert, damit die Treffer am Ende wieder dem richtigen Stück zugeordnet werden können. Auch die Anfrage wird in eine Merkmalsfolge $X = (x_1, x_2, \ldots, x_N) \in \mathcal{F}^N$ umgewandelt. Nach der Merkmalsextraktion liegen also zwei Folgen vor, wobei die Folge der Datenbank deutlich länger ist, als die der Anfrage: M ist typischerweise wesentlich größer als N. Ziel ist es nun nicht nur, die Anfrage mit jeder zusammenhängenden Teilfolge der Länge N aus der Datenbank zu vergleichen, sondern auch unterschiedlich lange Abschnitte, die Ähnlichkeiten aufweisen, zu berücksichtigen. Die Grafik 2.1 zeigt beispielhaft zwei unterschiedlich lange Folgen (hier reellwertige). Bei geschickter Zuordnung der einzelnen Folgenglieder zueinander findet man in der Folge Y einen Abschnitt, der zwar eine andere Länge als X hat, jedoch sehr ähnlich zu X ist.

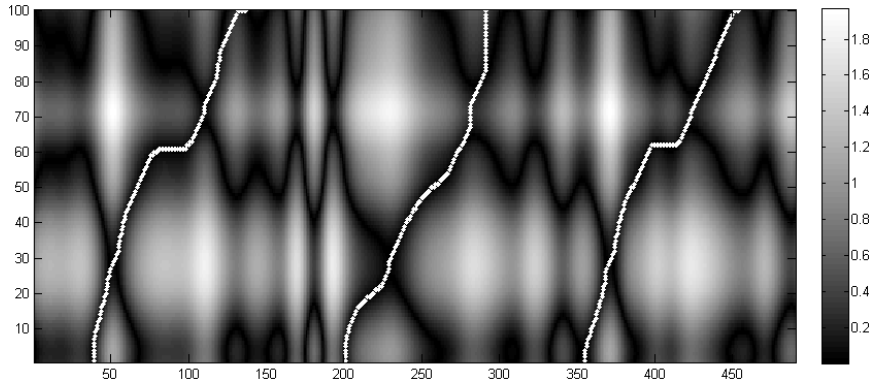


Abbildung 2.2. Drei Warpingpfade durch eine Kostenmatrix: Jeder Pfad verbindet die oberste Zeile mit der untersten, sodass die ganze Anfrage abgedeckt ist, d.h. einem Ausschnitt der Datenbank zugeordnet ist. Ziel des DTW-Algorithmus ist es, die Pfade durch „Täler“ mit minimalen Kosten zu legen.

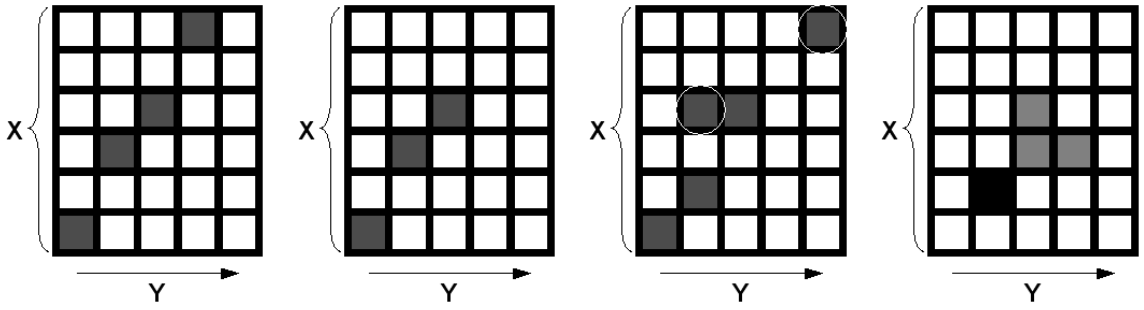
Um überhaupt Vergleiche zwischen zwei Merkmalsvektoren durchführen zu können, benötigt man ein *Abstandsmaß*, das durch eine Abbildung

$$c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0} \tag{2.1}$$

spezifiziert ist, die jedem Paar von Merkmalsvektoren einen Wert zuordnet, der ihre Ähnlichkeit repräsentiert. Im Allgemeinen fasst man das Abstandsmaß als Kostenfunktion auf, die bei geringer Ähnlichkeit große Werte (hohe Kosten) annimmt und bei großer Übereinstimmung niedrige Werte. In Abschnitt 2.2 wird das in dieser Arbeit verwendete Abstandsmaß eingeführt. Aus der Berechnung der Ähnlichkeiten zwischen allen Paaren (x_n, y_m) von Merkmalsvektoren aus den Folgen X und Y ergibt sich die *Kostenmatrix* $C \in \mathbb{R}^{N \times M}$ mit den Einträgen $C(n, m) := c(x_n, y_m)$. Die Idee hinter dem Matchingalgorithmus ist nun, Pfade wie in Abbildung 2.2 gezeigt durch die Matrix zu suchen, welche in irgendeiner Spalte bei x_1 beginnen und in einer fest vorgegebenen Spalte bei x_N enden, dabei soll der Pfad so gewählt werden, dass die Kosten minimal sind. Solch ein Pfad stellt eine Zuordnung zwischen der Anfrage und einer Sequenz der Datenbank dar, die eine andere Länge haben kann als die Anfrage. Addiert man die Kosten aller vom Pfad abgedeckten Paare von Merkmalsvektoren auf, so erhält man ein Maß für die Ähnlichkeit einer Teilfolge von Y zur Anfrage X. Erzeugt man für jedes y_m einen dort endenden Pfad, so ergeben sich die minimal möglichen Abstände zwischen Anfrage und allen Abschnitten der Datenbank.

Das Suchen der kostenminimalen Pfade geschieht mittels dynamischer Programmierung und liegt sowohl in Bezug auf Laufzeit, als auch auf Speicherkomplexität in $O(NM)$. Der Algorithmus wird detailliert und in allgemeinerem Kontext vorgestellt in [10]. Ein *Warpingpfad* ist eine Folge $p = (p_1, \ldots, p_L)$ mit $p_\ell = (n_\ell, m_\ell) \in [1 : N] \times [1 : M]$ für $\ell \in [1 : L]$, die den folgenden Bedingungen genügt:

1. $p_1 = (1, i)$ und $p_L = (N, j)$ mit $i, j \in [1 : M]$
2. *Schrittweite*: $p_{\ell+1} - p_\ell \in [(2, 1), (1, 2), (1, 1)]$ für alle $\ell \in [1 : L - 1]$



(a) ein gültiger Warpingpfad (b) X wird nicht abgedeckt (c) Schrittweitenbedingung verletzt (d) erreichbare Nachfolger

Abbildung 2.3. Die Bedingungen an einen Warpingpfad

Die erste Bedingung stellt sicher, dass jeder Warpingpfad die Anfrage X vom Anfang bis zum Ende abdeckt. Die zweite Bedingung verhindert, dass ein Pfad zeitlich rückwärts läuft und dass mehr als ein Vektor übersprungen wird. Grafik 2.3 veranschaulicht die Positionen, die ein Folgeglied eines Warpingpfades abhängig vom Vorgänger annehmen kann und zeigt Pfade, die durch die Einschränkungen verhindert werden. Eine naheliegende alternative Schrittweitenbedingung wäre $p_{\ell+1} - p_\ell \in [(1,0),(0,1),(1,1)]$. Sie wird nicht verwendet, weil so beliebig viele Vektoren von Y einem Element von X oder umgekehrt zugeordnet werden könnten.

Die *Gesamtkosten* c_p eines Warpingpfades p sind definiert durch

$$c_p := \sum_{\ell=1}^{L} c(x_{n_\ell}, y_{m_\ell}) \tag{2.2}$$

Sei mit $Y(a:b)$ die Teilfolge von $Y = (y_1, \ldots, y_a, \ldots, y_b, \ldots, y_M)$ bezeichnet, die mit Index a beginnt und mit b endet. Ein *optimaler Warpingpfad* zwischen X und der Sequenz $Y(a:b)$ ist der Pfad p^* mit den niedrigsten Gesamtkosten aller erlaubten Pfade zwischen den beiden Folgen. Die *DTW Distanz* zwischen X und $Y(a:b)$ ist dann definiert durch die Gesamtkosten von p^*:

$$\mathrm{DTW}(X, Y(a:b)) := c_{p^*} := \min\{c_p \mid p \text{ ist W.pfad zwischen } X \text{ und } Y(a:b)\} \tag{2.3}$$

Man möchte für sämtliche Indizes $b \in [1:M]$ ein $a \in [1:M]$ finden, sodass $\mathrm{DTW}(X, Y(a:b))$ minimal ist und hat damit einen optimalen Pfad, der in b endet. Dazu wird aus der Kostenmatrix C eine *akkumulierte Kostenmatrix* $D \in \mathbb{R}^{N \times M}$ erzeugt:

Algorithmus: AkkumulierteKostenMatrix

Eingabe: Kostenmatrix $C \in \mathbb{R}^{N \times M}$

Ausgabe: akkumulierte Kostenmatrix $D \in \mathbb{R}^{N \times M}$

Ablauf:

1. Setze $D(0, m) := \infty$ für $m \in [1 : M]$
2. Setze $D(1, m) := C(1, m)$ für $m \in [1 : M]$
3. Setze $D(n, 1) := \infty$ für $n \in [2 : N]$
4. Setze $D(n, 2) := \infty$ für $n \in [3 : N]$
5. für alle noch nicht gesetzten $D(n, m)$ setze rekursiv

$$D(n, m) := \min\{D(n-1, m-1), D(n-2, m-1), D(n-1, m-2)\} + C(n, m)$$

6. entferne die Hilfszeile 0

Die akkumulierte Kostenmatrix enthält so für jedes Paar $(n, m) \in [1 : N] \times [1 : M]$ die Kosten $\mathrm{DTW}(X(1 : n), Y(a_m : m))$ eines Warpingpfades, der die Bedingungen erfüllt und in einem Punkt a_m startet. Insbesondere enthält die „oberste“ Zeile der Matrix $D(N, m)$ mit $m \in [1 : M]$ damit die Kosten eines minimalen Warpingpfades, der die Anfrage X mit $Y(a_m : m)$ verbindet. Das Setzen von Einträgen der beiden ersten Spalten auf unendlich bewirkt, dass alle Felder, die gar nicht von einem gültigen Pfad erreicht werden können, auf unendlich gesetzt werden. Die temporäre Zeile 0 sorgt dafür, dass Zeile 2 berechnet werden kann, ohne dass das $n - 2$ zu Konflikten führt. In der Praxis werden die Kosten für den Schritt um zwei nach oben noch mit höheren Kosten belegt, damit der Algorithmus nicht dazu tendiert, sehr „steile“ Pfade zu erzeugen. Dies könnte daraus resultieren, dass kurze Pfade wegen der geringen Anzahl von Elementen auch häufig geringere Kosten haben.

Mit Hilfe der obersten Zeile der Matrix kann man also, wie in Abschnitt 2.3 beschrieben, zu X ähnliche Passagen in Y auffinden. Allerdings erhält man so nur die Endpunkte dieser Passagen. Der Startpunkt lässt sich aber sehr einfach in Linearzeit ermitteln:

Algorithmus: OptimalerWarpingPfad

Eingabe: akkumulierte Kostenmatrix $D \in \mathbb{R}^{N \times M}$, Endspalte E des Pfades

Ausgabe: Startspalte des Pfades

Ablauf: Der optimale Pfad $p^* = (p_1, \ldots, p_L)$ wird in umgekehrter Reihenfolge der Indizes berechnet, beginnend mit $p_L = (N, E)$. Sobald $n_\ell = 1$ für das aktuell berechnete $p_\ell = (n_\ell, m_\ell)$ gilt, ist der Anfang gefunden und es wird m_ℓ zurückgegeben. Ansonsten setze

$$p_{\ell-1} := \text{argmin}\{D(n_\ell - 1, m_\ell - 1), D(n_\ell - 2, m_\ell - 1), D(n_\ell - 1, m_\ell - 2)\}$$

und fahre rekursiv fort.
Falls „argmin“ nicht eindeutig ist, wähle das lexikografisch kleinste Paar.

2.2 Abstandsmaß

Damit der DTW-Algorithmus durchgeführt werden kann, muss eine Kostenfunktion gewählt werden, die jedem möglichen Paar von Vektoren eines Merkmalstyps einen Abstand zuordnet. Die Kostenfunktion soll Ähnlichkeiten herausarbeiten und zu großen Werten führen, wenn zwei Merkmalsvektoren sehr unterschiedlich sind, bzw. zu niedrigen Kosten, wenn es einen großen Grad an Übereinstimmung gibt. Für alle hier untersuchten Merkmalstypen soll das gleiche Abstandsmaß verwendet werden, damit deren Eigenschaften direkt miteinander verglichen werden können.

Das hier gewählte Abstandsmaß ist das *euklidische Skalarprodukt*, das für zwei Vektoren $x = (x_1, x_2, \ldots, x_n)^T$ und $y = (y_1, y_2, \ldots, y_n)^T$ definiert ist als

$$\langle x, y \rangle := x^T y = \sum_{i=1}^{n} x_i \cdot y_i \tag{2.4}$$

Um dieses Abstandsmaß anzuwenden, werden sämtliche in dieser Arbeit untersuchten Merkmale, bevor sie zum Einsatz kommen, mittels der L_2-Norm normiert. Dabei wird für jeden einzelnen Vektor $x = (x_1, x_2, \ldots, x_n)^T$ der Folge die Norm

$$\|x\|_2 := \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}} \tag{2.5}$$

berechnet und der Vektor komponentenweise durch den Wert geteilt. Die Normierung hat zum einen den Sinn, die Merkmale invariant gegenüber Lautstärke und damit auch Lautstärkeänderungen zu machen. Eine Passage, die genau wie die Anfrage gespielt ist, sich jedoch in der Lautstärke unterscheidet, wird auf diese Weise von den Merkmalen genauso

beurteilt, als wenn sie die gleiche Lautstärke hätte. Die andere entscheidende Motivation, die L_2-Norm zu benutzen, ist die Eigenschaft des euklidischen Skalarproduktes für zwei Vektoren x und y deren Winkel $\cos(\varphi) = \frac{x^T y}{\|x\|_2 \|y\|_2}$ zu bestimmen. Wenn die Vektoren beide L_2-normiert sind, erhält man also direkt den Winkel. Der Winkel gibt eine gute Auskunft über die Ähnlichkeit von zwei Vektoren, falls diese gleiche Länge haben. Wenn der Winkel gering ist, die Vektoren also eine vergleichsweise hohe Ähnlichkeit aufweisen, ist das Skalarprodukt nahe der Eins, während sich bei größer werdenden Winkeln die Werte Richtung Null bewegen. Da größere Ähnlichkeit jedoch geringere Kosten verursachen soll, wird das Ergebnis des Skalarproduktes noch von eins abgezogen. Letztendlich lautet das hier grundsätzlich verwendete Abstandsmaß für zwei Merkmalsvektoren $x = (x_1, x_2, \ldots, x_n)^T$ und $y = (y_1, y_2, \ldots, y_n)^T$ also

$$c(x, y) := 1 - \langle x, y \rangle = 1 - \sum_{i=1}^{n} x_i \cdot y_i \tag{2.6}$$

wobei sichergestellt wird, dass die Vektoren im L_2-Sinne normiert sind.

Bei der Normierung kann das Problem auftreten, dass alle Einträge eines Merkmalsvektors Null sind und damit auch die Norm, durch die geteilt werden soll, verschwindet. Zusätzlich würden durch das Normieren sehr leise Passagen, die nur noch aus Rauschen oder sonstigen nicht mehr relevanten Daten bestehen, verstärkt und könnten das Ergebnis verfälschen. Deshalb wird ein *Schwellwert* (engl. *threshold*) eingeführt, den die Norm überschreiten muss. Anderenfalls wird statt des normierten Vektors eine Gleichverteilung $x := (\frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}})^T$ für einen Vektor mit n Einträgen übernommen.

Die Schwellwerte sind durch Ausprobieren für die einzelnen Merkmale auf möglichst niedrige Werte festgelegt worden, sodass nur fast ganz stille Passagen auf Merkmalsebene durch Gleichverteilung ersetzt werden. Der Anteil der auf diese Weise ausgeblendeten Abschnitte ist so gering, dass man die Auswirkung auf die Matchingergebnisse vernachlässigen kann.

2.3 Erzeugung und Auswertung der Matchingkurve

Nach der Berechnung der Matrix D erhält man die Kosten von m Warpingpfaden. Die Kosten können also auch als Funktion in der Indexmenge der Datenbank $[1 : M]$ aufgefasst werden und führen dann zur *Matchingkurve*:

$$\Delta : [1 : M] \to \mathbb{R} \tag{2.7}$$

$$\Delta(m) := \frac{\mathrm{DTW}(X, Y(a_m : m))}{N} = \frac{D(N, m)}{N} \tag{2.8}$$

Wobei a_m der Anfangspunkt eines Warpingpfades mit minimalen Kosten nach m ist. Die Division durch N erfolgt, um den Wertebereich der Matchingkurve auf $[0, 1]$ zu normieren. Dies ist möglich, da ein Warpingpfad wegen der Schrittweitenbedingung höchstens die

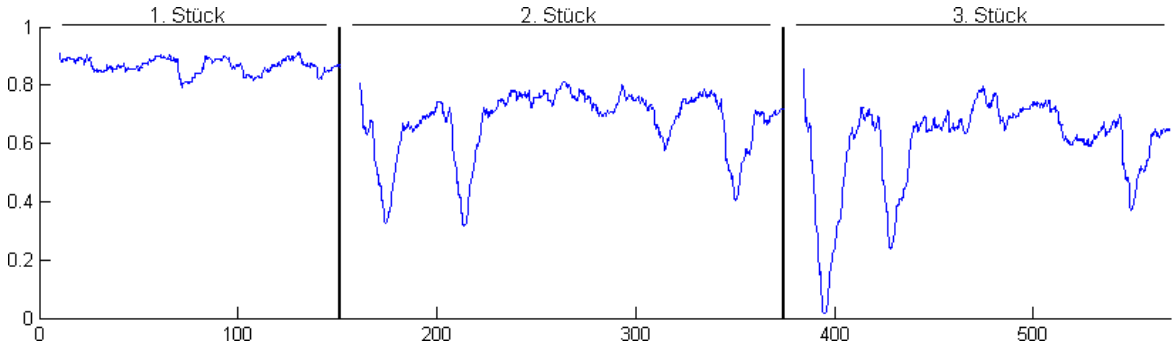


Abbildung 2.4. Eine Matchingkurve, die Y-Achse zeigt die Kosten, die X-Achse ist in Zeit in Sekunden umgerechnet und gibt die Position in der Datenbank an. Die senkrechten Linien trennen die einzelnen Stücke.

Länge N haben kann und das Abstandsmaß immer in $[0, 1]$ liegt (die MFCC-Merkmale bilden hier eine Ausnahme, der Wertebereich ist dann $[0, 2]$, siehe 5.3).

Bei einigen Merkmalen bietet es sich an, mehrere Durchläufe des Matchingalgorithmus mit jeweils leichten Modifikationen der Merkmalsvektoren hintereinander auszuführen. Beispielsweise eignen sich manche Merkmale dazu, Transpositionen der Tonhöhe durch Verschiebung der Vektoreinträge zu simulieren. Auf diese Weise erhält man K Matchingkurven Δ_k mit $k \in [1 : K]$, von denen an jedem Index jeweils der kleinste Funktionswert gewählt wird. Daraus resultiert wieder eine endgültige Kurve:

$$\Delta(m) := \min_{k \in [1:K]} \Delta_k(m) \tag{2.9}$$

In Abbildung 2.4 ist eine Matchingkurve zu erkennen. Die vertikalen Linien verdeutlichen Anfangs- bzw. Endpositionen der Stücke, die in der Datenbank konkateniert vorliegen. An den Anfängen der Stücke steigt die Kurve stark an und verschwindet irgendwann ganz. Dies liegt daran, dass zwischen den Stücken zwei Spalten der Matrix auf unendlich gesetzt werden. Der Matchingalgorithmus kann keinen Pfad durch diese Barriere legen, sodass die Pfade in der Nähe der Grenze immer mehr eingeengt werden und höhere Kosten verursachen. Nähert man sich der Grenze weiter, gibt es irgendwann keinen Warpingpfad mehr, der die Bedingungen erfüllt. Gleiches gilt natürlich auch für den linken Rand der gesamten Kurve. Auch gut zu sehen sind die verschiedenen lokalen Minima. Diese Peaks deuten Treffer an. Die Interpretation der Matchingkurve wird in den Kapiteln über Merkmalsextraktion genauer beschrieben.

Die endgültige Matchingkurve wird einerseits visualisiert und für Experimente genutzt, um die Eigenschaften verschiedener Merkmale zu untersuchen. Andererseits dient sie auch zum automatischen Auffinden der Trefferpassagen in den Musikstücken. Dazu wird das Minimum der gesamten Kurve bestimmt. Die Position dieses Minimums gibt den Index des Merkmalsvektors an, an dem die gefundene Passage in der Datenbank endet. Mit Hilfe der Matrix D kann der Warpingpfad rekonstruiert und der Startindex des Treffers ermittelt werden. Nun muss noch die Datenstruktur herangezogen werden, die die Positionen der Stücke innerhalb der Datenbank enthält. Nachdem der Index relativ zum Musikstück

bekannt ist, kann er in Zeit umgerechnet werden und die Position des Treffers in der Originaldatei lässt sich wiederfinden. Um weitere Treffer zu erhalten, muss das aktuelle Minimum aus der Matchingkurve entfernt werden, damit danach wieder ein neues gesucht werden kann. Zu diesem Zweck wird die letzte Fundstelle in der Matchingkurve auf unendlich gesetzt. Zusätzlich wird auch eine vorher festgelegte Umgebung des Treffers auf diese Weise entfernt, damit nicht der gleiche Treffer um einen Index verschoben wieder erscheint. Der ganze Vorgang – Minimum suchen, Position in Originaldatei berechnen, Treffer aus Matchingkurve entfernen – kann solange wiederholt werden, bis eine festgelegte Zahl von Treffern erreicht ist oder das nächste Minimum einen gewissen Schwellwert überschreitet.

Das Aussehen der Matchingkurve und die gewonnenen Treffer hängen von den verwendeten Merkmalen ab. Anhand der Matchingkurve sollen nun die Eigenschaften verschiedener Merkmale untersucht werden. Dabei werden kleine Testdatenbanken verwendet und Anfragen, zu denen die ähnlichen Passagen in der Datenbank bekannt sind. Diese zur Anfrage ähnlichen Abschnitte werden als erwartete oder gewünschte Treffer bezeichnet. Der Begriff der Ähnlichkeit ist natürlich nicht absolut zu definieren. In dieser Arbeit geht es um Abschnitte, die das gleiche Thema enthalten, jedoch bedingt durch andere Instrumentierung oder Spielweise anders klingen können. Einen Grenzfall stellt Beethovens Fünfte Sinfonie dar: Am Ende befindet sich das Thema der Exposition nur zur Hälfte, die andere Hälfte ist abgewandelt. Trotzdem soll diese Passage als ähnlich zum Thema der Exposition gelten.

Kapitel 3

Merkmalsextraktion mittels Filterbank

Das vorgestellte Matchingverfahren basiert auf Folgen von Merkmalsvektoren, welche zuerst aus den gegebenen Stücken in der Musikdatenbank und der Anfrage erzeugt werden müssen. Die Güte des Matchingverfahrens und die Eigenschaften der Treffer hängen maßgeblich von der Art der Merkmale ab. Der Hauptteil dieser Arbeit befasst sich mit unterschiedlichen Merkmalen und deren Parametrisierung, dabei werden die Auswirkungen auf die resultierenden Treffer untersucht.

Am Anfang liegen die einzelnen Musikstücke im PCM-Format vor, aus denen die Merkmale extrahiert werden. Das PCM-Format wird in Abschnitt 3.1 genauer erklärt. Dieses Kapitel widmet sich Merkmalen, die mit Hilfe von Filtern extrahiert werden, das dazu verwendete Konzept der Faltungsfilter wird in Abschnitt 3.2 kurz eingeführt.

In Abschnitt 3.3 geht es um filterbankbasierte Pitch- und Chromamerkmale. Zunächst wird deren Extraktionsvorgang behandelt. Darauf folgt eine Untersuchung der resultierenden Matchingkurven mit einer kurzen Einführung, welche Phänomene in den Kurven und den Visualisierungen der Merkmalsfolgen auftreten können. Auch die Veränderung der Ergebnisse bei unterschiedlicher Wahl der Parameter wird diskutiert.

Abschnitt 3.4 befasst sich mit Merkmalen, die aus einer statistischen Vergröberung der Pitch- oder Chromamerkmale hervorgehen. Die Vergröberung der Chromamerkmale wird als CENS bezeichnet. Auch hier wird der Extraktionsvorgang genau beschrieben und die Eigenschaften der Merkmale anhand von Matchingergebnissen untersucht.

3.1 Audiodaten im PCM-Format

Ein Audiosignal kann als Zeitfunktion $f : \mathbb{R} \to \mathbb{R}$ angesehen werden, die jedem Zeitpunkt einen Amplitudenwert zuordnet. Dieser Amplitudenwert repräsentiert dann z.B. den momentanen Luftdruck innerhalb einer Schallwelle oder die aktuelle Spannung in einem Lautsprecherkabel. Durch diese Darstellung ist das Audiosignal vollständig und exakt beschrieben. Damit das Signal digital verarbeitet werden kann, muss es sowohl im Zeitbereich

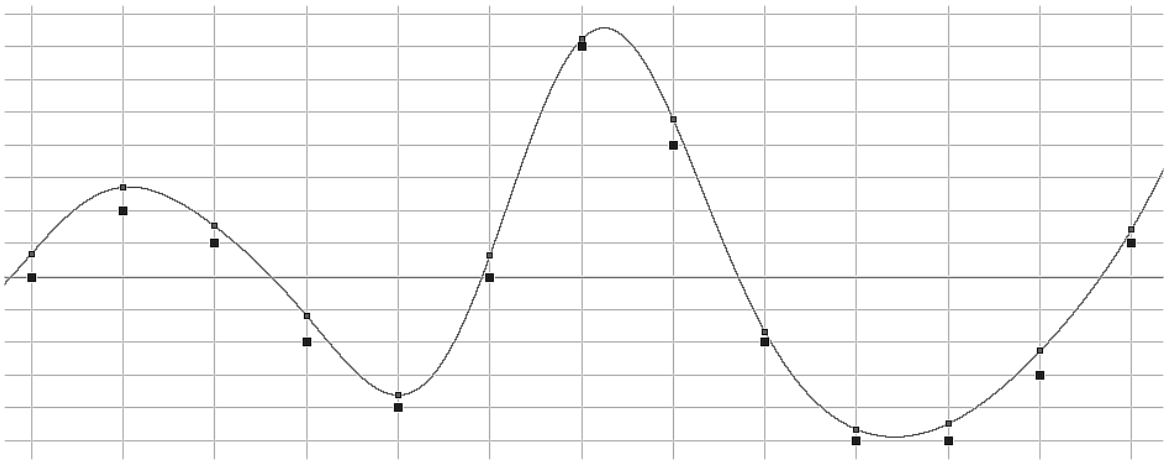


Abbildung 3.1. Audiosignal mit eingezeichneten Abtastpunkten. Jedes dieser Samples wird auf den darunter eingezeichneten Punkt gerundet.

als auch im Wertebereich diskretisiert werden.

Abtastung Die zeitliche Diskretisierung bezeichnet man als *Abtastung* oder *Sampling*. Für ein $T \in \mathbb{Q}^+$ überführt die T-Abtastung $x(n) := f(T \cdot n)$ die Funktion $f : \mathbb{R} \to \mathbb{R}$ in eine Folge $x : \mathbb{Z} \to \mathbb{R}$. Eine Abtastung ist also eine Auswertung der Zeitfunktion an äquidistanten Stellen. Die einzelnen Werte von x bezeichnet man als *Abtastwerte* oder *Samples*, T nennt man *Abtastperiode* und $\frac{1}{T}$ *Abtastrate*, welche häufig in Hertz (Hz) angegeben wird mit 1 Hz $\hat{=}$ 1 Abtastwert/Sekunde.

Quantisierung Anschließend muss noch der Wertebereich der Folge diskretisiert werden. Dieser Vorgang heißt *Quantisierung*. Dazu muss zunächst ein Intervall bestimmt werden, in dem alle Abtastwerte liegen. Im PCM-Format ist dieses Intervall (abhängig von der Bitrate) festgelegt. Dieses Intervall wird nun in endlich viele, gleich große[1] Intervalle aufgeteilt und jedes der Teilintervalle durch eine ganze Zahl kodiert. Jeder Wert der zuvor abgetasteten Funktion wird also abhängig vom Intervall, in dem er sich befindet, durch eine ganze Zahl dargestellt, sodass die Folge nur noch einen ganzzahligen Wertebereich hat. Die *Bitrate* bestimmt, wieviele Bits die kodierenden Zahlen lang sind und damit in wieviele Intervalle der Wertebereich partitioniert wird.

Das resultierende Signal ist von der Form $x : \mathbb{Z} \to \mathbb{Z}$, ein Beispiel zeigt Abbildung 3.1. Aus technischen Gründen, damit die Fouriertransformation immer definiert ist, legt man formal den Signalraum fest als

$$\ell^2(\mathbb{Z}) := \{x : \mathbb{Z} \to \mathbb{C} \mid \|x\|_2 < \infty\} \text{ mit } \|x\|_2 := \left(\sum_{k\in\mathbb{Z}} |x(k)|^2\right)^{\frac{1}{2}} \tag{3.1}$$

[1] Es existieren auch PCM-Arten, die mit unterschiedlichen Intervallbreiten arbeiten und logarithmisch quantisieren.

Die Einschränkung auf Funktionen mit endlicher Norm ist für alle in der Praxis auftretenden Audiosignale möglich, da diese nur endliche Amplitudenwerte haben können und zeitlich begrenzt sind, also endlich viele Abtastwerte besitzen. Die Erweiterung des Wertebereichs auf komplexe Zahlen ist rein mathematisch motiviert durch die Definition der Fouriertransformation. Bei allen hier auftretenden Audiosignalen ist der Imaginärteil gleich Null und der Realteil nimmt nur ganzzahlige Werte an.

Die entstandene Folge von diskreten Werten kann nun im Rechner z.B. als WAV-Datei abgespeichert werden. WAV-Dateien sind im Wesentlichen eine Sequenz von Abtastwerten mit fester Anzahl von Bits pro Wert, wobei auch mehrere Kanäle unterstützt werden. Die hier verwendete Musikdatenbank enthält WAV-Dateien in Mono mit einer Abtastrate von 22050 Hz und 16 bit pro Sample. Zu beachten ist noch, dass der Wertebereich der PCM-Daten häufig, so z.B. in MATLAB, auf $[-1, 1]$ skaliert wird und bei den Berechnungen damit rationale Zahlen mit Betrag zwischen 0 und 1 vorliegen.

Das PCM-Format enthält keine direkten Informationen über Frequenzen und Tonhöhen, sondern nur über den zeitlichen Amplitudenverlauf. Bei einigen der im Folgenden vorgestellten Verfahren zur Merkmalsextraktion wird versucht Informationen dieser Art zurückzugewinnen.

3.2 Faltungsfilter

Eine sehr häufig verwendete Strategie der Merkmalsextraktion ist, das Audiosignal in einzelne Frequenzbänder zu zerlegen, welche dann weiterverarbeitet werden. Diese Frequenzbänder können beispielsweise der westlichen Tonleiter mit zwölf Noten pro Oktave oder sonstigen musikalischen Einteilungen entsprechen oder aber gewissen Höreigenschaften nachgebildet sein. Aber auch andere Zerlegungen sind denkbar und sollen hier untersucht werden. Eine Möglichkeit der Zerlegung in Frequenzbänder stellen *Filter* dar, dieses Konzept wird zunächst kurz eingeführt.

Ein *Filter* ist eine Abbildung $T : I \to O$, die ein Eingangssignal $x \in I$ in ein Ausgangssignal $y \in O$ umwandelt. Als Signalraum wird hier nur der oben definierte Raum $\ell^2(\mathbb{Z})$ von Folgen zugelassen, es gilt also $I = O = \ell^2(\mathbb{Z})$. Bei den hier betrachteten Filtern handelt es sich um *lineare zeitinvariante* Filter (LTI-Filter von engl. *linear time-invariant*), die die folgenden Eigenschaften besitzen:

Linearität Für alle Signale $x, y \in \ell^2(\mathbb{Z})$ und $\lambda \in \mathbb{C}$ gilt $T[\lambda \cdot x + y] = \lambda \cdot T[x] + T[y]$.

Zeitinvarianz Für alle Signale $x \in \ell^2(\mathbb{Z})$ und $k \in \mathbb{Z}$ gilt $T[x^k] = (T[x])^k$,
wobei x^k das um k verschobene Signal ist: Für alle $n \in \mathbb{Z}$ sei $x^k(n) := x(n-k)$.

LTI-Filter werden durch ihre *Impulsantwort* eindeutig charakterisiert. Die Impulsantwort eines Filters

$$h := T[\delta] \quad \text{mit} \quad \delta(n) := \begin{cases} 1 & falls\ n = 0 \\ 0 & sonst \end{cases} \tag{3.2}$$

ist das Ausgangssignal, das man erhält, wenn das Filter auf den *Einheitsimpuls* δ angewendet wird. Ein Signal kann nun mit dem LTI-Filter bearbeitet werden, indem man es mit der Impulsantwort h des Filters faltet. Dabei ist die lineare *Faltung* zweier Folgen x, y definiert als

$$(x * y)(n) := \sum_{k \in \mathbb{Z}} x(k) \cdot y(n-k) \tag{3.3}$$

Ist die Impulsantwort h eines LTI-Filters bekannt, so kann das Ausgangssignal y aus dem Eingangssignal x mittels $y = C_h[x] := h * x$ berechnet werden.

Interessant für die Merkmalsextraktion werden Faltungsfilter durch ihre Eigenschaften im Frequenzbereich. Sei $\hat{x}$ die Fouriertransformierte des Signals x:

$$\hat{x}(\omega) := \sum_{n \in \mathbb{Z}} x(n) \cdot e^{2\pi i \omega n} \tag{3.4}$$

Dann heißt die Fouriertransformierte der Impulsantwort eines Filters $\hat{h}$ *Frequenzgang* des Filters. Zerlegt man $\hat{h}(\omega) := a(\omega) \cdot e^{2\pi i \varphi(\omega)}$ in Amplitude $a(\omega) := |\hat{h}(\omega)|$ und Phase $\varphi(\omega)$, so heißen a *Amplitudengang* und φ *Phasengang* des Filters. Synonym werden die Begriffe Frequenzantwort, Amplitudenantwort und Phasenantwort gebraucht. Auch durch den Frequenzgang ist ein LTI-Filter eindeutig charakterisiert. Nach dem Faltungssatz

$$\widehat{x * y} = \hat{x} \cdot \hat{y} \tag{3.5}$$

gilt, dass durch Transformation in den Frequenzbereich das Falten eines Signals mit der Impulsantwort in punktweise Multiplikation mit der Frequenzantwort überführt wird. Diese Eigenschaft ist wichtig, denn somit können einzelne Frequenzen des Signals durch Faltung im Zeitbereich unterschiedlich gewichtet werden. Insbesondere können auf diese Weise einzelne Frequenzen unterdrückt bzw. einzelne Frequenzbänder extrahiert werden. Es existieren Algorithmen, die aus einer gewünschten Frequenzantwort eine geeignete Impulsantwort generieren, siehe z.B. in [2], sodass man in der Praxis mit Hilfe von Faltungsfiltern Frequenzbandzerlegungen durchführt, um Merkmale zu extrahieren.

3.3 Pitchmerkmale und Chromamerkmale

Eine Idee, Merkmale für das Audiomatchingverfahren zu generieren, ist die Rückgewinnung von Informationen über enthaltene Tonhöhen und damit Noten aus dem Zeitsignal. Mit Hilfe verschiedener Filter, die jeweils Frequenzen bestimmter Noten extrahieren und die anderen Frequenzen ausblenden, ist eine solche Zerlegung des Signals möglich. Der genaue Vorgang zur Berechnung der Pitch-, Chroma- und auch CENS-Merkmale ist [10] entnommen worden.

3.3.1 Pitchzerlegung

Die Frequenzen der einzelnen Noten werden anhand der im MIDI-Standard festgelegten Werte bestimmt und entsprechen so der westlichen Oktavskala mit zwölf Halbtonschritten pro Oktave. Die *Zentrumsfrequenz* einer MIDI-Note $p \in [1:120]$ errechnet sich mittels

$$f_m(p) := 2^{\frac{p-69}{12}} \cdot 440 \text{ Hz} \tag{3.6}$$

wobei $p = 69$ den Kammerton a' darstellt, der folgerichtig eine Zentrumsfrequenz von 440 Hz hat. Die Zerlegung geschieht nun mit Hilfe einer *Filterbank*, d.h. es werden verschiedene Filter jeweils auf das ursprüngliche Signal angewendet und man erhält verschiedene Ergebnissignale. Die Filterbank für die Pitchzerlegung besteht aus *Bandpassfiltern* für jede MIDI-Note. Dabei haben Bandpassfilter die Eigenschaft, ein bestimmtes Frequenzintervall ungehindert passieren zu lassen und die anderen Frequenzen auszublenden. Neben der Zentrumsfrequenz der Note ist auch ihre *Bandbreite* zu beachten. Wegen der exponentiellen Entwicklung der Frequenzen und der immer größer werdenden Abstände der Zentrumsfrequenzen muss auch die Breite der Noten für höhere Werte wachsen, um das gesamte Spektrum abzudecken. Dieser Zusammenhang von Zentrumsfrequenz und Bandbreite wird vom *Q-Faktor* (von engl. *quality-factor*) erfasst, welcher definiert ist durch $Q := \frac{\text{Zentrumsfrequenz}}{\text{Bandbreite}}$. Konstruiert man nun die Bandpassfilter der Filterbank so, dass alle den gleichen Q-Faktor aufweisen, so erhält man eine der exponentiellen Skala angepasste Zerlegung.

Im Folgenden werden einige Details zu der Filterbank genannt, die zur Pitchzerlegung für die Experimente dieser Arbeit verwendet worden sind. Die Zentrumsfrequenzen entsprechen denen der MIDI-Noten, wobei 88 Bandpassfilter zur Filterbank gehören, die den MIDI-Noten von $p = 21$ bis $p = 108$ entsprechen, sodass die Tasten eines Klavieres nachgebildet werden. Tatsächlich kann über die Parameter „MIDI_min“ und „MIDI_max“ der Bereich, in dem die Filterbank arbeitet, noch eingeschränkt werden. Der Q-Faktor beträgt für alle Filter 25, so ergibt sich beispielsweise für die Note $a' \hat{=} (p = 69)$ eine Bandbreite von $w = \frac{f_m(p)}{Q} = \frac{440 \text{ Hz}}{25} = 17.6$ Hz. Damit werden die Grenzen für das Bandpassfilter nach Ausrichtung am Zentrum zu 431.2 Hz und 448.8 Hz. Auf beiden Seiten gibt es noch einen Übergangsbereich von jeweils der halben Breite des Filters, in dem Frequenzen bis zum Rand immer niedriger gewichtet werden. Genaueres dazu findet man in [10]. Mit Übergangsbereich deckt das Filter für a' die Frequenzen von $431.2 - 8.8 = 422.4$ Hz bis 457.6 Hz ab. Wie Abbildung 3.2 zeigt, überlappen sich die Übergangsbereiche zweier benachbarter Filter.

Je tiefer die Zentrumsfrequenz, desto schmaler wird das Frequenzband, das herausgefiltert werden soll. Aus technischen Gründen, da schmale Frequenzbänder numerisch schwieriger zu behandeln sind, wird das Signal der Filterbank in drei Sampleraten zur Verfügung gestellt. Dazu wird es mit entsprechenden Antialiasingfiltern bearbeitet und jeweils ein Downsampling von 22050 Hz auf 4410 Hz und auf 822 Hz durchgeführt. Tonhöhen von $p = 93$ und größer werden aus dem Signal mit 22050 Hz herausgefiltert. Von $p = 57$ bis $p = 92$ verwendet man das mit 4410 Hz und für die darunterliegenden wird das Signal mit 822 Hz Abtastrate als Eingabe benutzt. Die niederfrequenten Bereiche lassen sich so einfacher präzise bearbeiten.

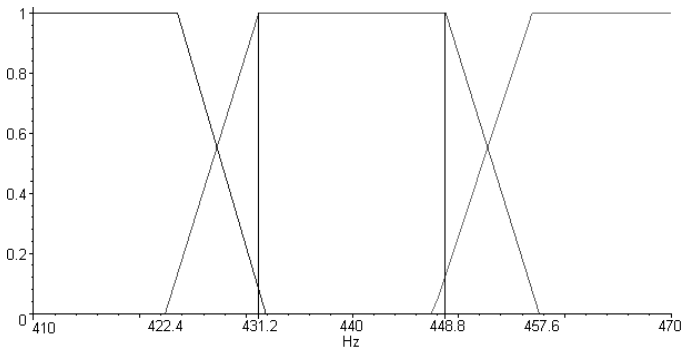


Abbildung 3.2. Amplitudenantwort eines Bandpassfilters. Die x-Achse zeigt die Frequenz, die y-Achse gibt den Faktor an, mit dem die Amplituden der Frequenzen des Signals multipliziert werden. Man erkennt den Durchlass- und den Übergangsbereich des Filters und die Überlappung mit den Nachbarfiltern.

3.3.2 Lokale Energieverteilung (STMSP)

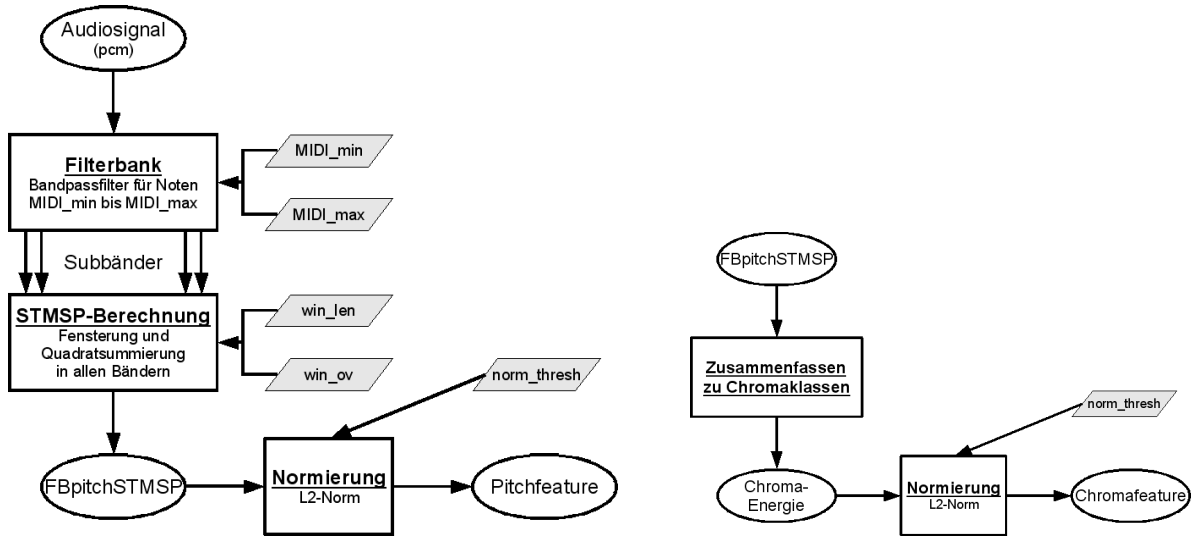
Die einzelnen Merkmalsvektoren sollen nun Informationen zur Energieverteilung auf die Subbänder zu bestimmten Zeitpunkten enthalten und damit Aufschluss über gespielte Noten geben. Dazu werden in allen Bändern lokale Energieverteilungen (*short-time mean-square power (STMSP)*) berechnet. Sei x eines der Subbänder und $w \in \mathbb{N}$ die festgelegte Breite des Intervalls, auf dem die lokale Energie ermittelt werden soll. Dann ist für jeden Zeitpunkt $n \in \mathbb{Z}$ des Signales x die lokale Energie (STMSP) definiert durch

$$\mathrm{stmsp}_{x,w}(n) := \sum_{k \in \left[n - \left\lfloor \frac{w}{2} \right\rfloor : n + \left\lfloor \frac{w}{2} \right\rfloor\right]} |x(k)|^2 \tag{3.7}$$

Der Vorgang stellt eine Fensterung mit einem Rechteckfenster der Länge w dar mit Bildung der Quadratsumme aller Werte innerhalb der Fenster. Diese Fensterung und Summenbildung kann auch als Faltung des quadrierten Signals mit dem Rechteckfenster aufgefasst werden. Das Ergebnis ist ein Signal gleicher Länge, da für jedes Sample des Signals auch ein Fenster vorliegt, das dort zentriert ist. Wegen der äußerst großen Überlappung der Fenster enthält das Ergebnis sehr viel Redundanz, die nicht mehr benötigt wird. Um die Datenmenge zu verringern, wird deshalb nur alle d Samples ein Fenster zentriert und ausgewertet, was einem *Downsampling* des aus der Faltung resultierenden Signals um den Faktor d entspricht. Nach der Energieberechnung für alle Subbänder x_i fasst man alle Energiewerte je Zeitfenster zu einem Merkmalsvektor zusammen. Bei einer Zerlegung in 88 Subbänder, einer Fensterbreite w und einem Downsamplingfaktor d lautet der n-te Vektor also

$$\left(\mathrm{stmsp}_{x_1,w}(d \cdot n), \mathrm{stmsp}_{x_2,w}(d \cdot n), \ldots, \mathrm{stmsp}_{x_{88},w}(d \cdot n)\right)^T \tag{3.8}$$

Die resultierende Folge von Merkmalsvektoren ist durch die Reduktion der Fensteranzahl kompakter als das Ursprungssignal, sodass man die berechneten Vektoren abspeichert, um eine erneute Durchführung des Filterns zu vermeiden. Außerdem dient die Pitchzerlegung nicht nur selbst als Merkmal, sondern wird auch als Ausgangspunkt für die Erzeugung einiger weiterer genutzt.



(a) Pitch (b) Chroma

Abbildung 3.3. Ablaufdiagramme - 3.3(a): Erzeugung der pitchSTMSP Daten und Pitchmerkmale mittels Filterbank. Dargestellt sind Daten (oval), Arbeitsschritte (rechteckig) und Parameter (raute). 3.3(b): Schritte, mit denen pitchSTMSP zu Chromamerkmalen weiterverarbeitet wird.

In Abbildung 3.3(a) ist der Ablauf der gesamten Extraktion von Pitchmerkmalen schematisch dargestellt. Der Parameter „win_len“ entspricht der oben eingeführten Fensterlänge w, während „win_ov“ die Überlappung zweier benachbarter Fenster angibt, welche intern in den Downsamplefaktor $d = \text{win\_len} - \text{win\_ov}$ umgerechnet wird. Man erkennt auch die abschließende Normierung, bevor das abgespeicherte Merkmal zum Audiomatching eingesetzt wird.

3.3.3 Berechnung der Chromamerkmale

Die Pitchmerkmale können noch weiter vergröbert werden, d.h. es kann weiter Informationsgehalt reduziert werden, um robust gegenüber bestimmten Änderungen zu werden und evtl. weitere Ähnlichkeiten von Passagen erkennen zu können. Eine Möglichkeit dazu ist, die Pitches in *Chromaklassen* zusammenzufassen. Eine Chromaklasse enthält alle Noten, die zu einem bestimmten Halbton modulo der Oktave gehören, für einen MIDI-Pitch p ist dies die Menge $\{q \in [1:120] \mid q \equiv p(\mod 12)\}$. Zum Beispiel gehören die Noten $C, c, c', c'' \ldots$ zu einer Chromaklasse. Zur Erzeugung der Chromamerkmale bedient man sich der zuvor berechneten noch nicht normierten Pitchmerkmale und addiert in jedem Vektor alle Werte, die zu einer Chromaklasse gehören, auf. Es resultieren Vektoren mit jeweils 12 Einträgen (einer für jeden Halbtonschritt) aus den zuvor 88-dimensionalen Merkmalsvektoren.

Dieser zusätzliche auf den Pitchmerkmalen aufbauende Schritt ist in 3.3(b) dargestellt. Wie üblich werden die Vektoren vor dem Matching L_2-normiert.

3.3.4 Verwendete Musikdaten

Bevor die Pitch- und Chromamerkmale getestet werden, sollen sämtliche in dieser Arbeit beispielhaft benutzten Audiomatchingszenarien, d.h. Anfragen und Datenbanken, vorgestellt werden. In Abbildung 3.4 sind die sechs verwendeten Testdatenbanken dargestellt. Grundsätzlich wird mit Stücken in Mono mit einer Abtastrate von 22050 Hz, 16 bit gearbeitet.

Die Bezeichnung „Datenbank“ ist für die kurzen Kollektionen von zwei oder drei Stücken sicherlich übertrieben. Allerdings soll die typische Terminologie der QBE-Aufgabe (zur Anfrage passende Positionen in einer Datenbank finden) beibehalten werden, sodass in der gesamten Arbeit die zu durchsuchende Musikansammlung, sei sie noch so kurz, als Datenbank bezeichnet wird. Am Ende werden einige Merkmale auch auf einer größeren Datenbank getestet, welche in Anhang B aufgelistet ist.

Mit Abstand am häufigsten wird die in 3.4(f) abgebildete Datenbank $\mathcal{D}^{Sho}$ verwendet. Ihre Eigenschaften werden zunächst kurz erläutert, während die Einzelheiten der seltener eingesetzten anderen Testdatenbanken bei ihrer Verwendung erklärt werden. Auf $\mathcal{D}^{Sho}$ wird immer $\mathcal{Q}^{Sho}$ angefragt. Das zweimal enthaltene Stück von Shostakovich ist nach dem Schema A_1, A_2, B, A_3, A_4 aufgebaut. Das Thema vom Anfang des Stückes erscheint also insgesamt viermal, was in der Datenbank somit zu acht Treffern führen sollte. Die einzelnen Wiederholungen klingen jedoch unterschiedlich, da A_1 von Streichern gespielt wird, A_2 von einer Klarinette und Holzbläsern, A_3 von einer Posaune und Blechbläsern und Teil A_4 vom gesamten Orchester. Diese Art des Aufbaus macht das Stück sehr interessant zum Testen von Merkmalen für das Audiomatching. Das Orgelstück von Bach befindet sich zusätzlich in der Datenbank, um zu erkennen, wie sich einzelne Merkmale bei Passagen verhalten, die nicht zu Treffern führen sollen, sondern im Gegenteil sehr unterschiedlich zur Anfrage sind.

3.3.5 Audiomatching mit Pitch- und Chromamerkmalen

Es wird nun ein Audiomatching mit den Pitchmerkmalen durchgeführt. Datenbank ist $\mathcal{D}^{Sho}$, Anfrage der 19-sekündiger Abschnitt $\mathcal{Q}^{Sho}$ aus der zweiten Interpretation, welcher pro Stück viermal in ähnlicher Form vorkommt. Zunächst werden, neben dem für die Filterbank immer geltenden Bereich von $p = 21$ bis $p = 108$, der Parameter „win_len“ auf 4410 Samples und „win_ov“ auf 2205 Samples eingestellt. Daraus ergibt sich, dass alle $4410 - 2205 = 2205$ Samples ein Fenster ausgewertet wird und damit $\frac{22050}{2205} = 10$ Vektoren pro Sekunde vorliegen. Allgemein ergibt sich somit folgende Formel für die Merkmalsrate, falls eine Nyquistfrequenz von F Hz vorliegt:

$$\frac{F}{\text{win\_len} - \text{win\_ov}} \tag{3.9}$$

Abbildung 3.5 stellt eine Visualisierung der Merkmalsfolge der Anfrage dar. Wegen der Merkmalsrate von 10 Hz enthält die Merkmalsfolge zu der 19 sekündigen Anfrage 190 Vektoren, deren Komponenten entlang der Y-Achse angeordnet sind. Man erkennt die

Komponist	Stück	Interpret / Dirigent	Position in DB (s)
J.S. Bach	BWV 565, Toccata	P. Cabrera	0-160
J.S. Bach	BWV 565, Toccata	W. Ruebsam	160-332

(a) $\mathcal{D}^{BachI}$

Komponist	Stück	Interpret / Dirigent	Position in DB (s)
J.S. Bach	BWV 565, Toccata	T. Koopman	0-151
J.S. Bach	BWV 565, Toccata	P. Cabrera	151-311

(b) $\mathcal{D}^{BachII}$

Komponist	Stück	Interpret / Dirigent	Position in DB (s)
L.v. Beethoven	5. Sinfonie, 1. Satz (Orchester)	L. Bernstein	0-519
L.v. Beethoven	5. Sinfonie, 1. Satz (Klavier)	K. Scherbakov	519-963

(c) $\mathcal{D}^{BeSin}$

Komponist	Stück	Interpret / Dirigent	Position in DB (s)
L.v. Beethoven	Sonate Nr. 17 „Der Sturm“, 1. Satz	D. Barenboim	0-526
L.v. Beethoven	Sonate Nr. 17 „Der Sturm“, 1. Satz	E.G. Gilels	526-1087

(d) $\mathcal{D}^{BeSo}$

Komponist	Stück	Interpret / Dirigent	Position in DB (s)
L.v. Beethoven	5. Sinfonie, 1. Satz	L. Bernstein	0-519
M. Ravel	Bolero	C. Abbado	519-1381

(e) $\mathcal{D}^{Ravel}$

Komponist	Stück	Interpret / Dirigent	Position in DB (s)
J.S. Bach	BWV 565, Toccata	T. Koopman	0-151
D.D. Shostakovich	Jazz Suite Nr. 2, 6. Satz (Walzer)	D. Yablonsky	151-344
D.D. Shostakovich	Jazz Suite Nr. 2, 6. Satz (Walzer)	R. Chailly	344-567

(f) $\mathcal{D}^{Sho}$

Komponist	Stück	Interpret / Dirigent	Ausschnitt (s)	Bezeichnung
J.S. Bach	BWV 565, Toccata	P. Cabrera	0-34	Bach
J.S. Bach	BWV 565, Toccata	P. Cabrera	23-34	BachK
J.S. Bach	BWV 565, Toccata	W. Ruebsam	29-46	BachT
L.v. Beethoven	5. Sinfonie, 1. Satz (Orchester)	L. Bernstein	0-21	BeSin
L.v. Beethoven	Sonate Nr. 17 „Der Sturm“, 1. Satz	D. Barenboim	69-93	BeSo
D.D. Shostakovich	Jazz Suite Nr. 2, 6. Satz (Walzer)	D. Yablonsky	0-19	Sho

(g) Anfragen $\mathcal{Q}^{Bezeichnung}$

Abbildung 3.4. Testdatenbanken, die in den Beispielen dieser Arbeit verwendet werden. Neben dem Namen der Stücke sind Komponist und Interpret/Dirigent der Version eingetragen, außerdem die zeitliche Position in der Datenbank. Tabelle 3.4(g) enthält die verwendeten Testanfragen.

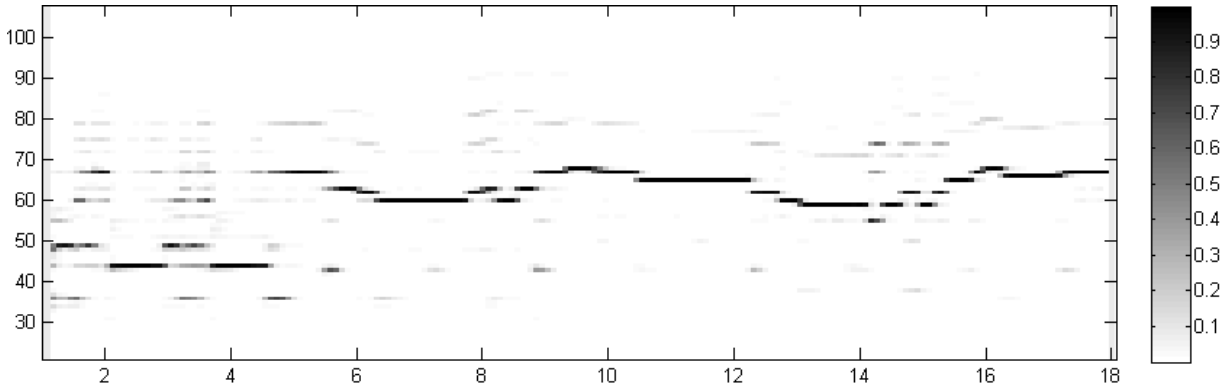


Abbildung 3.5. Visualisierung der Pitch-Merkmalsvektoren der Anfrage $\mathcal{Q}^{Sho}$. Die X-Achse zeigt die Zeit in Sekunden. An der Y-Achse sind die einzelnen MIDI-Noten aufgetragen. Die Energie pro Pitch und Zeitpunkt ist durch die Graustufe (siehe Balken) dargestellt.

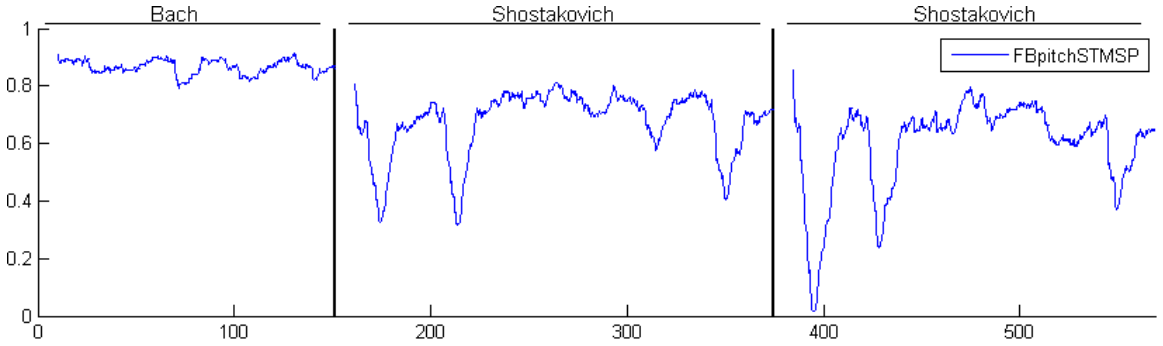
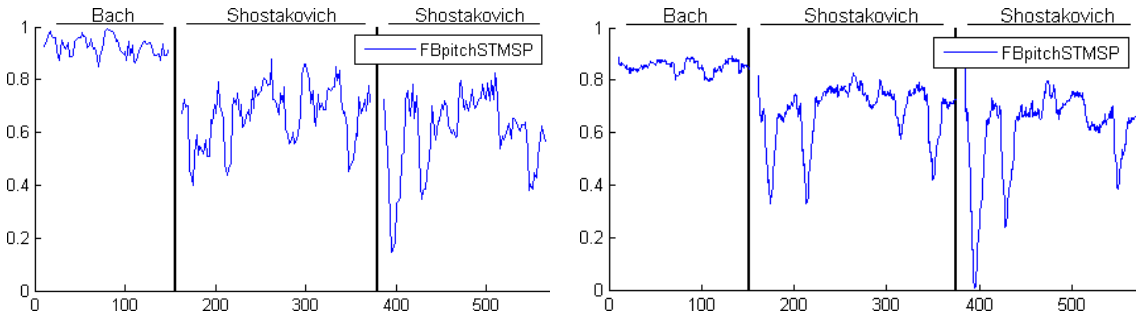


Abbildung 3.6. Visualisierung einer Matchingkurve vom Matching mit Pitchmerkmalen ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$). Die vertikalen Linien trennen die drei Stücke, die in der Datenbank konkateniert vorliegen. Erkennbar sind sieben deutliche lokale Minima der Kurve, hierbei handelt es sich um Treffer. Tatsächlich müssten acht Treffer vorliegen, da in den Stücken von Shostakovich jeweils vier zur Anfrage ähnliche Passagen vorliegen.

Energieverteilung pro Vektor auf die 88 Bänder, die Farbskala repräsentiert den Energieanteil. Da in der Passage relativ wenige Obertöne oder zusätzliche Klänge enthalten sind, lässt sich der Melodieverlauf recht klar erkennen. Am linken und rechten Rand sind hellgraue vertikale Streifen zu sehen. Hierbei handelt es sich um Merkmalsvektoren, die auf Gleichverteilung gesetzt worden sind, da die Gesamtlautstärke am Anfang und Ende der Anfrage unter den Schwellwert fällt.

In Abbildung 3.6 ist die Matchingkurve zu der Anfrage an die Datenbank zu erkennen. Sofort fallen sechs deutliche Peaks und ein etwas kleinerer auf. Bei allen diesen Peaks handelt es sich um korrekte Treffer, die tatsächlich das Thema aus der Anfrage beinhalten. Das absolute Minimum bei knapp 400 Sekunden lokalisiert den exakten Treffer, von dieser Position stammt die Anfrage. Eigentlich würde man bei einem exakten Match Kosten von Null erwarten, jedoch ist die Fensterung der Anfrage und der Datenbank nicht genau synchron: Bei der Anfrage wird das erste Fenster genau über dem ersten Sample zentriert.



(a) Merkmalsrate 0.5 Hz (b) Merkmalsrate 50 Hz

Abbildung 3.7. Ergebnis des Matchings ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$) mit gröber bzw. feiner aufgelösten Pitchmerkmalen. Die niedrigere Rate ist an der „eckigeren“ Kurve zu erkennen. Die Treffer bei der geringeren Merkmalsrate setzen sich nicht so deutlich vom Rest ab, siehe Text.

Dies ist zwar auch bei der Datenbank der Fall, allerdings liegt der Abschnitt, der zur Anfrage identisch ist, irgendwo in der Datenbank, sodass die Fenster etwas verschoben sind. Daraus ergeben sich sehr ähnliche, aber durch die Verschiebung minimal abweichende Merkmalsfolgen.

Der kleinere Peak im zweiten Stück gehört zu Abschnitt A_3, in dem das Thema von einer Posaune und Blechbläsern gespielt wird. Bedingt durch die deutlich andere Klangfarbe im Gegensatz zur Anfrage, die von Streichern gespielt wird, sind die Abstände der Pitchmerkmalsvektoren trotz gleicher Melodie verhältnismäßig groß. Die zugehörige Stelle im dritten Stück ist nicht mehr als Peak auszumachen. Andererseits würde nach Entfernen der sieben deutlichen Treffer diese Position zum Minimum und damit tatsächlich zum achten Treffer. Dennoch stößt man hier an die Grenzen der Pitchmerkmale. Die Matchingkurve nimmt im Stück von Bach generell höhere Werte an, was daran liegt, dass sich sowohl die Klangfarbe der Orgel als auch die Melodie stark vom Orchesterstück unterscheidet.

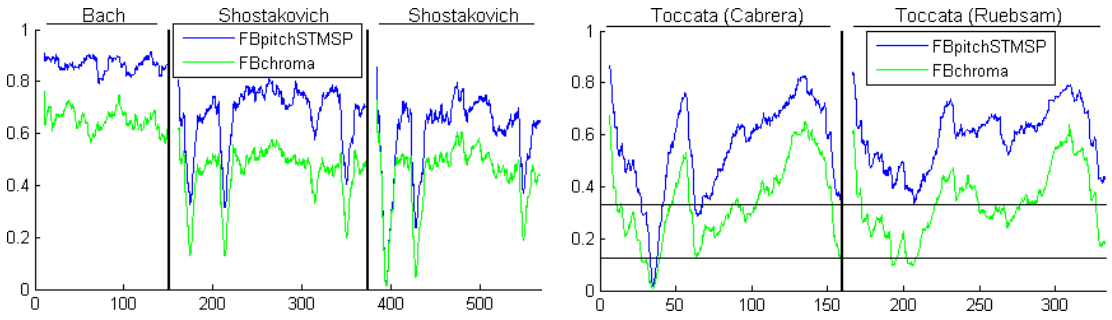
Im folgenden Abschnitt wird erörtert, wie sich Änderungen der Fensterbreite und der Merkmalsrate auf die Matchingkurve auswirken. Dazu wird erst eine Fensterbreite von 50000 Samples gewählt und eine Überlappung von 5900. Statt zehn Merkmalsvektoren pro Sekunde verbleibt, nach Formel (3.9), nur noch einer für zwei Sekunden. Dieses extreme Beispiel dient dazu, die Veränderungen des Ergebnisses zu zeigen, wenn die zeitliche Auflösung in eine bestimmte Richtung geändert wird. Abbildung 3.7(a) zeigt die resultierende Kurve. Einige der Peaks weisen höhere Kosten auf als bei der größeren Merkmalsrate. Dieses Phänomen beruht wie die Tatsache, dass der exakte Treffer keine Kosten von Null liefert, auf einer Verschiebung der Fenster in Anfrage und Datenbank: Bei der Verwendung von größeren Fenstern können die Fenster der Anfrage gegenüber den Fenstern der Datenbank weiter verschoben sein. Andererseits verläuft die Kurve an einigen Stellen niedriger als im Anfangsbeispiel und es existieren deutlich mehr kleinere Spitzen. Dies lässt sich dadurch erklären, dass einige Details, die die Anfrage und eine Passage im Musikstück unterscheiden, durch die Mittelung innerhalb der großen Fenster verschwinden. So führen im Groben ähnliche, jedoch in Einzelheiten unterschiedliche Abschnitte schneller zu nied-

rigeren Kosten. Erkennbar ist auch, dass die Kurve im Stück von Bach ihr Niveau kaum ändert, da hier auch keine gröberen Ähnlichkeiten zur Anfrage vorliegen. Insgesamt kann man sagen, dass bei größer werdenden Fensterbreiten der Kontrast zwischen Treffern und dem Rest der Kurve schwächer wird.

Nun wird die Fensterbreite reduziert und die Merkmalsrate erhöht. Die Fensterbreite wird auf 882 erniedrigt und mittels einer Überlappung von 441 Samples eine Merkmalsrate von 50 Vektoren pro Sekunde eingestellt. Die Zeit, die das Matching in Anspruch nimmt, steigt wegen der quadratischen Laufzeit des DTW-Algorithmus deutlich an. In Abbildung 3.7(b) ist die Matchingkurve zu sehen. Die Kosten des exakten Treffers unterscheiden sich erwartungsgemäß nur noch sehr geringfügig von der Null, denn die kleineren Fenster lassen nur noch geringe Verschiebungen zu. Ansonsten liegen keine größeren Unterschiede zum Matching mit einer Rate von zehn Merkmalen pro Sekunde vor. Insbesondere hat sich durch die höhere Auflösung keine Verbesserung der Treffer eingestellt, die Berechnungszeit ist jedoch drastisch gestiegen.

Allgemein haben sich Auflösungen im Bereich von zehn bis zu zwei Merkmalen pro Sekunde bewährt, siehe [10]. Um die Überlappung auf halber Fensterbreite zu belassen, wird von nun an, wie in [11], bei allen mittels Filterbank erzeugten Merkmalen, die auf Pitchmerkmalen basieren, von einer Fensterbreite von 4410 Samples und 2205 Überlappung ausgegangen. Dann folgen teilweise, je nach Merkmalstyp, noch weitere zeitliche Vergröberungen, die am Ende zu geringeren Raten führen.

Mit gleicher Datenbank ($\mathcal{D}^{Sho}$) und Anfrage ($\mathcal{Q}^{Sho}$) wird jetzt zum Vergleich ein Audiomatching mit Chromamerkmalen durchgeführt. Standardmäßig sind eine Fensterbreite von 4410, eine Überlappung von 2205 Samples und damit eine Auflösung von 10 Vektoren pro Sekunde eingestellt. Es ergibt sich die in Bild 3.8(a) sichtbare Matchingkurve, die große Ähnlichkeiten zu der Kurve der Pitchmerkmale aufweist. Generell bewegt sich das Ergebnis jedoch im Bereich niedrigerer Kosten, verglichen zum Ergebnis des Matchings mit den 88-dimensionalen Vektoren. Dieses Phänomen ist wieder damit zu erklären, dass durch die weitere Vergröberung, sprich das Zusammenfassen zu Chromaklassen, Details verloren gehen und deshalb mehr Ähnlichkeiten zwischen Merkmalsvektoren auftreten können. Mehr Aufschluss soll ein zweites Beispiel geben. In der Datenbank $\mathcal{D}^{BachI}$ befinden sich zwei Stücke von Bachs „Toccata“ auf zwei verschiedenen Orgeln gespielt. Deshalb ist die Klangfarbe und damit die Obertonstruktur der beiden Versionen sehr unterschiedlich. Anfrage ist $\mathcal{Q}^{BachK}$. Beide Matchingkurven in Abbildung 3.8(b) zeigen deutlich die Position, von der die Anfrage stammt. Die Kurve der Pitchmerkmale weist auch ihren zweiten Treffer im ersten Stück auf, obwohl die Anfrage dort nicht mehr vorkommt. Mit den Pitchmerkmalen wird also in diesem Fall eine höhere Ähnlichkeit auf Grund der Klangfarbe und nicht auf Grund der Melodie festgestellt, was hier zu einem falschen Ergebnis führt. Die Chromamerkmale sind robuster gegenüber der Obertonstruktur, da höhere Frequenzen, die Obertöne darstellen, mit entsprechenden tieferen Frequenzen zusammengefasst werden, siehe dazu auch [1]. Die zu den Chromamerkmalen gehörende Kurve weist ihren zweiten Treffer im zweiten Stück auf. Die Ablenkung durch die andere Klangfarbe ist also geringer als bei den Pitchmerkmalen. Allerdings findet man im zweiten Stück zwei fast gleiche Peaks direkt nebeneinander. Tatsächlich ist das niedrigere dieser beiden Minima nicht der korrekte



(a) Shostakovich ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$) (b) Bach ($\mathcal{D}^{BachI}$, $\mathcal{Q}^{BachK}$)

Abbildung 3.8. Audiomatching mit Pitch- und Chromamerkmalen: Wegen der stärkeren Vergröberung liefern die Chromamerkmale generell niedrigere Kosten. Im Stück von Shostakovich ändert sich beim Übergang zu Chromaklassen nichts an den Treffern, jedoch beim obertonreichen Orgelstück von Bach.

zweite Treffer, sondern das direkt daneben liegende. Die Chromamerkmale liefern somit zwar bessere Ergebnisse in Bezug auf Klangfarbe, genügen jedoch für dieses Beispiel auch nicht. Eine weitere Vergröberung der pitchbasierten Merkmale, die noch robuster gegenüber bestimmten Abweichungen ist, wird in Kapitel 3.4 vorgestellt.

Bei Chromamerkmalen bietet es sich an, die Komponenten der einzelnen Merkmalsvektoren zyklisch zu verschieben und mit den verschobenen Versionen erneut Matchingkurven zu erzeugen. Die generelle Idee wird in [5] vorgestellt und in [10] auf das Audiomatching übertragen. Bei einer Verschiebung um einen Halbtonschritt wird z.B. der Energieverlauf des Bandes c auf $c\#$ übertragen und h wird zu c. Enthält die Datenbank eine zur Anfrage ähnliche Passage, jedoch um einen Halbton nach oben transponiert, so ergibt ein Audiomatching mit einer um eins nach oben verschobenen Anfrage auf der unveränderten Datenbank niedrige Kosten für diese Passage. Führt man das Audiomatching jeweils für alle zwölf möglichen zyklischen Verschiebungen der Anfrage durch und ermittelt dann in jedem Punkt das Minimum der zwölf resultierenden Matchingkurven, so ergibt sich eine neue Kurve, die zusätzlich invariant gegenüber Transpositionen um eine beliebige Zahl von Halbtonschritten ist. Abbildung 3.9 zeigt zwei Matchingkurven stammend von der Datenbank $\mathcal{D}^{BeSo}$ mit zwei Interpretationen von Beethovens „Klaviersonate Nr. 17“. Anfrage ist $\mathcal{Q}^{BeSo}$. Die Kurve der zyklisch verschobenen Merkmale weist drei korrekte Treffer pro Stück auf, alle gewünschten Abschnitte werden damit gefunden. Wird keine Verschiebung durchgeführt, erhält man nur zwei Treffer pro Stück. Der jeweils in nur einer Kurve vorhandene Peak zeigt eine Position in der Reprise an, in der das Thema transponiert gespielt wird. Mit Hilfe des zyklischen Verschiebens kann die transponierte Stelle gefunden werden, während ohne Verschieben keine Ähnlichkeit erkannt wird. Abbildung 3.10 zeigt die Merkmale der Anfrage und eines Abschnitts der Reprise. Deutlich zu sehen ist die Ähnlichkeit im Melodie- und Harmonieverlauf, jedoch sind die einzelnen Chromabänder verschoben.

Auch auf Pitchmerkmalen, die nicht zu Chroma zusammengefasst sind, kann ein Verschieben durchgeführt werden. Allerdings ergibt hier ein zyklisches Verschieben keinen Sinn. Stattdessen kann man die einzelnen Komponenten nach oben oder unten schieben und frei

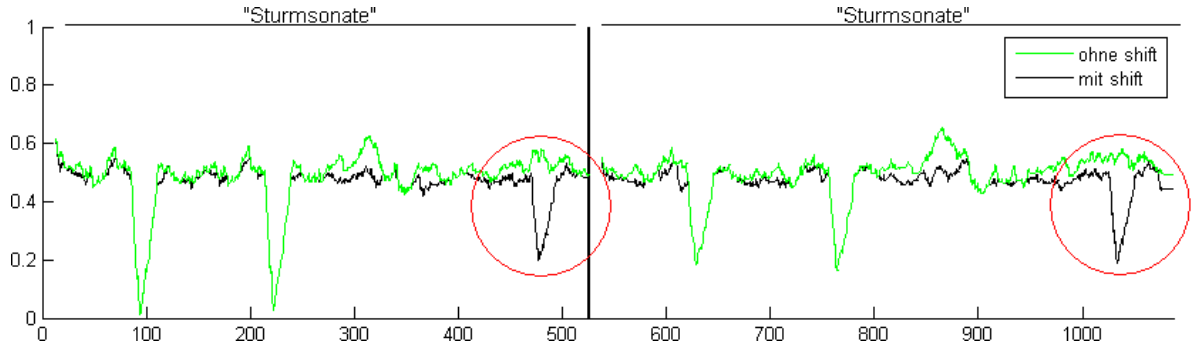
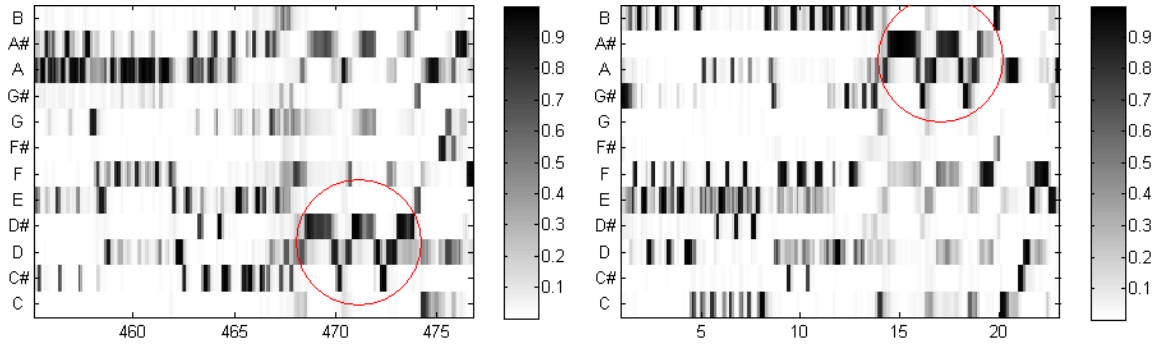


Abbildung 3.9. Zwei Matchingergebnisse ($\mathcal{D}^{BeSo}$, $\mathcal{Q}^{BeSo}$) mit gleichen Chromamerkmalen, einmal mit und einmal ohne zyklische Verschiebung. Markiert sind die Treffer, die nur mit Verschieben auftreten. Zu beachten ist, dass die hellere Kurve die dunklere überlagert und deshalb einige Teile der dunklen Kurve nicht zu sehen sind, da sie exakt wie die helle verlaufen.



(a) vergrößerter Abschnitt Datenbank $\mathcal{D}^{BeSo}$ (b) Anfrage $\mathcal{Q}^{BeSo}$

Abbildung 3.10. Chromamerkmale von Anfrage und eines Abschnittes der Datenbank. Der Abschnitt ist ähnlich zur Anfrage, jedoch transponiert, was man besonders gut am eingekreisten Teil erkennen kann.

werdende Pitchbänder im oberen bzw. unteren Bereich mit Null auffüllen. Führt man den DTW-Algorithmus jeweils für Verschiebungen um bis zu elf nach oben und nach unten durch, so können Transpositionen im Rahmen einer Oktave berücksichtigt werden. Beim Beispiel der Klaviersonate erzielt man auf diese Weise sehr ähnliche Ergebnisse wie beim zyklischen Verschieben der Chromamerkmale.

3.4 CENS-Merkmale

Sowohl die Pitch- als auch die Chromamerkmale stoßen bei Abschnitt A_3 des Stückes von Shostakovich an ihre Grenzen. Und wie das Beispiel mit den Versionen von „Toccata“ zeigt, werden Passagen mit noch größerer Abweichung, die jedoch immer noch das gleiche Thema enthalten, gar nicht mehr gefunden. Wünschenswert sind daher Merkmale, die noch robuster gegenüber lokalen Tempoänderungen und anderer Klangfarbe sind.

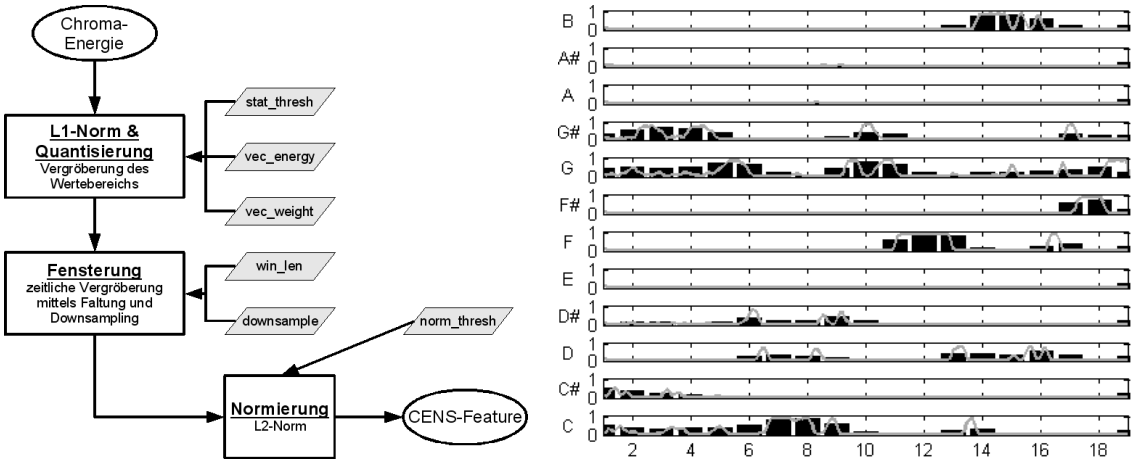
3.4.1 Berechnung der CENS-Merkmale

Zu diesem Zweck werden jetzt die Chromamerkmale sowohl im Werte- als auch im Zeitbereich weiter vergröbert. Daraus erhält man die *CENS-Merkmale (Chroma Energy distribution Normalized Statistics)*, die in [11] vorgestellt werden. Zunächst erfolgt eine Normierung der noch nicht L_2-normierten Chromavektoren mittels der L_1-Norm $\|x\|_1 := \sum_{i=1}^{n} |x_i|$ für einen Vektor $x = (x_1, x_2, \ldots, x_n)^T$. Diese Norm hat den Sinn, absolute Lautstärkeinformationen aus den Merkmalsvektoren zu entfernen und stattdessen nur noch die Verteilung der Energie auf die einzelnen Chromaklassen hervorzuheben. Außerdem wird dadurch der nächste Schritt ermöglicht, welcher eine Quantisierung des Wertebereiches darstellt. Wie in [11] beschrieben wird in jedem normalisierten Chromavektor $v = (v_1, v_2, \ldots, v_{12})^T$ jeder Eintrag v_i wie folgt vergröbert:

$$v_i' := \begin{cases} 4 & falls\ 0.4 < v_i \leq 1 \\ 3 & falls\ 0.2 < v_i \leq 0.4 \\ 2 & falls\ 0.1 < v_i \leq 0.2 \\ 1 & falls\ 0.05 < v_i \leq 0.1 \\ 0 & falls\ v_i \leq 0.05 \end{cases} \tag{3.10}$$

Die Abstufung ist logarithmisch gewählt, um eine Annäherung an die menschlichen Wahrnehmung der Lautstärke zu erzielen, welche ebenfalls logarithmisch ist. Näheres zu Extraktionsschritten, die durch die Eigenschaften des Gehörs motiviert sind, findet man in Kapitel 5 über die MFCC-Merkmale.

Nun folgt noch eine zeitliche Vergröberung. Dazu werden die quantisierten Merkmalsvektoren komponentenweise mit einem Hannfenster der Länge 41 gefaltet und danach ein Downsampling um den Faktor 10 durchgeführt. Dies entspricht wieder wie bei der Berechnung der Pitchmerkmale einer Fensterung und Summierung der gewichteten Werte innerhalb jedes Fensters, siehe Abschnitt 3.3.2. In diesem Fall haben die Fenster eine Breite von 41 und eine Überlappung von 31. Bedingt durch das Downsampling reduziert sich die Merkmalsrate um den Faktor 10. Startet man also mit den typischen Chromamerkmalen mit Fensterbreite 4410 und Überlappung 2205 auf 22050 Hz WAV-Dateien, d.h. einer Merkmalsrate von 10 pro Sekunde, so liegt bei den CENS-Merkmalen nur noch eine Rate von einem Merkmal pro Sekunde vor. Schließlich werden die einzelnen Merkmalsvektoren wie gewohnt L_2-normiert. Der Vorgang der Gewinnung von CENS- aus den Chromamerkmalen ist in Abbildung 3.11(a) schematisch dargestellt. Die Grafik zeigt fünf Parameter, die zusätzlich zu den Parametern der Chromaextraktion vorhanden sind. „stat_thresh“ stellt einen Schwellwert dar, der bei der L_1-Normierung benötigt wird, um zu kleine Werte auszuschließen, genauso wie der Schwellwert in der abschließenden L_2-Normierung. Mit den Parametern „vec_energy“ und „vec_weight“ lassen sich die Abstufung und die Bewertung der Quantisierung einstellen, welche hier generell wie in Formel (3.10) festgelegt ist. Schließlich werden die Breite und Überlappung der Hannfenster über „win_len“ und „downsample“ parametrisiert. Auch diese Werte sind bei allen folgenden Experimenten auf 41 bzw. 10 fixiert.



(a) Berechnung von CENS-Merkmalen (b) Vergleich Chroma-CENS

Abbildung 3.11. Vergröberung der Chromamerkmale zu CENS-Merkmalen: 3.11(a) zeigt den Ablauf der weiteren Vergröberung von Chroma-Energievektoren zu CENS-Merkmalen. 3.11(b) visualisiert Chromamerkmale (durchgezogene Kurve) und die niedriger aufgelösten CENS-Merkmale (Balkendiagramm).

In Abbildung 3.11(b) sind CENS-Merkmalsvektoren und die Chromamerkmale, aus denen sie erzeugt worden sind, visualisiert. Es handelt sich um die bekannte 19 sekündige Anfrage aus dem Stück von Shostakovich. Zum besseren Vergleich der beiden Merkmalstypen wird diesmal eine andere Darstellung als bisher gewählt. Die schwarzen Balken geben die Komponenten der CENS-Vektoren wieder. Deutlich erkennt man, dass es sich um 19 Vektoren mit je zwölf Einträgen handelt. Damit liegt pro Sekunde ein CENS-Merkmalsvektor vor. Die hellere Kurve stammt von den Chromamerkmalen, man erkennt die höhere Merkmalsrate, nämlich 10 pro Sekunde, und die zeitliche Vergröberung der CENS-Merkmale.

Die Gewinnung der CENS-Merkmale geschieht also in zwei Schritten, zum einen der zeitlich genauen Bestimmung der Energieverteilung auf die zwölf Chromaklassen, zum anderen einer statistischen Auswertung des Ergebnisses über längere Zeitintervalle. Eine Wahl von größeren Fenstern direkt auf STMSP-Ebene führt zu ungenauen Merkmalen, wie in 3.3.5 gezeigt. Durch die Aufteilung in zwei Stufen werden auch lokale Informationen aufgenommen und dann statistisch ausgewertet. Durch die Quantisierung des Wertebereiches werden die CENS-Merkmale zusätzlich robuster gegenüber Rauschen, das in der Anschlagsphase von Noten auftreten kann, siehe auch [11].

Zusätzlich zu den chromabasierten CENS-Merkmalen wird die statistische Vergröberung mit gleichen Parametern auch noch direkt auf die 88-dimensionalen Pitchmerkmale angewendet. Die so entstehenden Merkmale werden im Folgenden FBpitchStat-Merkmale (von Statistik) genannt.

Abbildung 3.12. Vergleich dreier Merkmale anhand der Matchingkurven ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$)

3.4.2 Audiomatching mit CENS-Merkmalen

Abbildung 3.12 zeigt die drei Kurven, die aus dem Matching mit Chroma-, FBpitchStat- und CENS-Merkmalen resultieren. Das Matchingszenario besteht aus der gewohnten Datenbank $\mathcal{D}^{Sho}$ und der Anfrage $\mathcal{Q}^{Sho}$. Bemerkenswert ist, dass die Kurven der statistisch vergröberten Merkmale nicht ganz bis zu den Begrenzungslinien reichen. Dies liegt an der unterschiedlichen Merkmalsrate: Bei reinen Chromamerkmalen liegen zehn Vektoren pro Sekunde vor, bei den beiden anderen nur einer pro Sekunde.

Beim Betrachten fällt auf, dass sich die drei Kurven im Wesentlichen auf drei verschiedenen Ebenen befinden, besonders im Stück „Toccata“ wird das deutlich. Die CENS-Merkmale tendieren dazu, generell niedrigere Kosten zu erzeugen, während die nicht zu Chroma zusammengefassten Merkmale höhere Kosten liefern. Die niedrigeren Kosten der CENS-Merkmale sind nicht direkt durch die geringere Zahl von Merkmalsvektoren zu erklären, denn beim Erzeugen der Kurven wird, wie in Kapitel 2.3 definiert, durch die Anzahl der Vektoren in der Anfrage geteilt. Die Tatsache, dass eine höhere Merkmalsrate zu längeren Warpingpfaden führt, wird also auf Kostenebene beim Erzeugen der Matchingkurve ausgeglichen. Indirekt besteht jedoch ein Zusammenhang, da die niedrigere Merkmalsrate ja durch zeitliche Vergröberung entstanden ist und somit gewisse Details, die die Anfrage von einem Abschnitt der Datenbank unterscheiden, verloren gehen. Genauso verschwinden durch die Quantisierung Unterschiede im Wertebereich, sodass bei CENS also geringere Abstände auftreten. Interessant ist auch, dass die statistisch vergröberten Pitchmerkmale höhere Kosten liefern als die reinen Chromamerkmale. Offensichtlich führt die Reduktion auf 12-dimensionale Vektoren zumindest für dieses Beispiel zu mehr Verlust von Information, die Unterschiede herausstellt. Besonders in „Toccata“, welches keinen Treffer enthält, sieht man, dass die Chromamerkmale eine größere Ähnlichkeit anzeigen als die 88-dimensionalen vergröberten Pitchmerkmale.

Nun werden die Treffer, die das Matching mit den drei Merkmalen jeweils liefert, verglichen. Die bekannten sieben Peaks der Chromamerkmale sind auch in den Matchingkurven

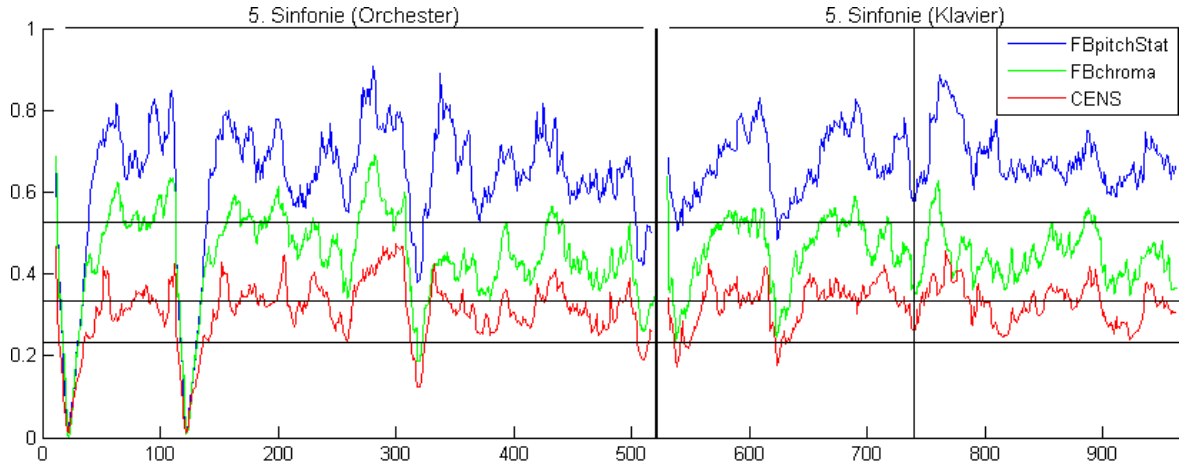


Abbildung 3.13. „Beethovens Fünfte“ als Orchester- und Klavierversion ($\mathcal{D}^{BeSin}$). Die Anfrage $\mathcal{Q}^{BeSin}$ stammt aus der Orchesterversion. Bei keinem der drei gezeigten Merkmale können alle vier ähnlichen Passagen in der Klavierversion gefunden werden.

der beiden statistisch vergröberten Merkmale als Treffer zu erkennen. Teil A_3 der zweiten Interpretation von Shostakovich, der bei den Chromamerkmalen nicht als Treffer auftaucht, setzt sich bei den FBpitchStat-Merkmalen minimal ab und wird zum achten Treffer. In der Matchingkurve der CENS-Merkmale ist sogar ein recht deutlicher Peak an dieser Position zu erkennen. Durch die statistische Vergröberung werden die Merkmale also robuster gegenüber Abweichungen, sodass klanglich unterschiedliche Passagen, die jedoch das gleiche Thema beinhalten, eher gefunden werden können. Kombiniert mit dem Zusammenfassen zu Chromaklassen wird der Treffer noch deutlicher. Andererseits sinken auch die Kosten für Abschnitte, die nicht zu den gewünschten Treffern gehören, wie z.B. das gesamte Stück „Toccata“, das keine Treffer für die Anfrage aus dem Stück von Shostakovich liefern soll. Die Matchingkurve der CENS-Merkmale nimmt hier Werte an, die nur noch minimal größer sind als einige korrekte Treffer. Stärkere Vergröberung macht Merkmale also sensibler für gewünschte Treffer mit größerer Abweichung in Parametern wie Tempo, Tempoänderung, Lautstärke, Artikulation und Klangfarbe, jedoch mit dem Preis, dass falsche Treffer wahrscheinlicher werden.

Zum Abschluss der filterbankbasierten Merkmale folgt noch eine härtere Matchingaufgabe. Die Datenbank $\mathcal{D}^{BeSin}$ enthält eine Orchesterversion und eine Klaviertranskription von Beethovens Fünfter Sinfonie. Bei der Anfrage $\mathcal{Q}^{BeSin}$ handelt es sich um die ersten 21 Sekunden des Orchesterstückes. In beiden Versionen kommt das in der Anfrage enthaltene Thema viermal vor. Am Ende beider Stücke ist jedoch nur die Hälfte des Themas zur Anfrage ähnlich, die zweite Hälfte ist verändert. Die Matchingkurven sind in Abbildung 3.13 dargestellt. Es handelt sich wieder um die Merkmale Chroma, FBpitchStat und CENS. Die ersten sechs Treffer aller drei Merkmale sind korrekt. Dazu gehören immer die vier Positionen aus der Orchesterversion und die Exposition bzw. deren Wiederholung im Klavierstück, welche zumindest vom Aufbau her noch der Anfrage ähnlich sind. Der siebte Treffer ist bei jedem Merkmal falsch. Bei der Reprise der Klavierversion zeigen die drei

Kurven zwar ein kleines lokales Minimum, jedoch folgt diese Position in den Trefferlisten aller drei Merkmale deutlich später. Die vierte Position lässt sich gar nicht mehr als Treffer erkennen.

Die beiden Versionen unterscheiden sich zum einen sehr stark in der Klangfarbe, zum anderen ist aber auch der Aufbau des vielschichtigen Orchesterstückes anders als die nur auf dem Klavier gespielte Version. Trotzdem führt die Ähnlichkeit von Melodie und Harmonie auch im Klavierstück noch zu Treffern bei der Exposition, welche aus dem Orchesterstück als Anfrage gewählt worden ist. Die Reprise in der Klavierversion, die das Thema in modifizierter Form enthält, führt nicht mehr zu genügend geringen Kosten, um in der Trefferliste weit genug nach vorne zu gelangen. Schließlich wird der letzte Abschnitt, der auch nur zur Hälfte mit dem Thema der Anfrage übereinstimmt, gar nicht mehr als ähnlich erkannt. Hier stellt sich jedoch die Frage, was man intuitiv überhaupt noch als ähnlich gelten lassen will, d.h. welche Passagen man nach welchen Kriterien auswählt, die später als Treffer gewünscht sind. Denn tatsächlich stimmen hier Klangfarbe, Tempo, Lautstärke und Artikulation nicht mit der Anfrage überein und auch der Melodieverlauf nur in der ersten Hälfte.

Diese Matchingaufgabe zeigt, dass je nach Anforderung und Vorstellung vom intuitiven Ähnlichkeitsbegriff unterschiedliche Merkmale an ihre Grenzen stoßen. Es macht also Sinn, Merkmale mit unterschiedlichen Eigenschaften zu erzeugen und zu untersuchen. Im nächsten Kapitel werden basierend auf einer gefensterten Fouriertransformation verschiedene Merkmale erstellt und anhand ihrer Matchingkurven untersucht.

Kapitel 4

Merkmalsextraktion mittels STFT

Die bisherige Strategie der Merkmalsextraktion besteht aus einer Filterung und anschließender Energieberechnung innerhalb von Zeitfenstern für jedes Band. Ein anderer Ansatz ergibt sich aus der gefensterten Fouriertransformation. Hierbei werden zuerst Zeitfenster gebildet und diese jeweils mittels einer DFT transformiert. Die einzelnen Koeffizienten pro Zeitfenster werden dann zu Bändern zusammengefasst. Grundsätzlich führen beide Ansätze zu ähnlichen Ergebnissen. Jedoch haben die Strategien unterschiedliche Vor- und Nachteile. So sind Filterbänke präziser, die Spektrogramme lassen sich jedoch flexibler weiterverarbeiten durch einfaches Addieren von einzelnen Koeffizienten, ohne dass eine neue Filterbank erzeugt werden muss.

Nach einem kurzen Überblick über die gefensterte Fouriertransformation in Abschnitt 4.1 wird in Abschnitt 4.2 zunächst ein Audiomatching direkt mit Spektralvektoren durchgeführt und die Ergebnisse werden untersucht. In Abschnitt 4.3 wird dann die angenäherte Pitchzerlegung mit Fouriertransformation behandelt und die Ergebnisse von Pitchzerlegungen mittels Filterbank und mittels STFT verglichen. Abschnitt 4.4 befasst sich mit unterschiedlichen Zerlegungen des Spektrums auf Basis der von der STFT-Pitchzerlegung stammenden Technik. Hier werden Oktavbinnings mit und ohne Überlappung der einzelnen Bins betrachtet. Dann wird eine Verfeinerung auf Halboktaven und kleinere Frequenzabschnitte durchgeführt und die Entwicklung der Matchingkurven untersucht. Die unterschiedlichen Binnings werden anhand ihrer Treffer auf einer umfangreicheren Datenbank verglichen.

4.1 Gefensterte Fouriertransformation

Die *gefensterte Fouriertransformation* (engl. *Short Time Fourier Transform (STFT)*) ermöglicht es, innerhalb bestimmter Zeitintervalle die Frequenzverteilung eines Signals zu berechnen.

Die *diskrete Fouriertransformation (DFT)* eines Signals $x = (x_1, x_2, \ldots, x_L)^T$ mit $x_i \in \mathbb{C}$ für alle $i \in [1 : L]$ ist definiert als Multiplikation des Signalvektors mit der quadratischen

DFT-Matrix passender Größe:

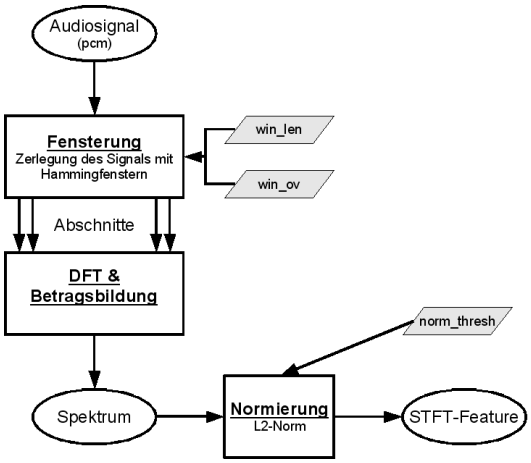
$$\hat{x} := \mathrm{DFT}_L \cdot x := \left(\Omega_L^{kj}\right)_{0 \le j,k < L} \cdot x \tag{4.1}$$

wobei $\Omega_L := e^{\frac{-2\pi i}{L}}$ die primitive L-te Einheitswurzel ist. Ausgeschrieben lautet die DFT-Matrix der Größe L damit:

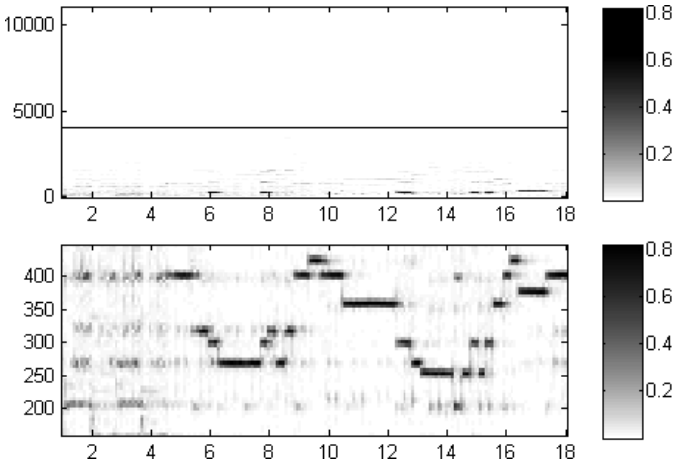
$$\mathrm{DFT}_L = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{\frac{-2\pi i}{L}} & \cdots & e^{\frac{-2\pi i(L-1)}{L}} \\ 1 & e^{\frac{-4\pi i}{L}} & \cdots & e^{\frac{-4\pi i(L-1)}{L}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\frac{-2\pi i(L-1)}{L}} & \cdots & e^{\frac{-2\pi i(L-1)^2}{L}} \end{pmatrix}$$

Das resultierende Signal $\hat{x}$ ist komplex und hat ebenfalls Länge L. Die einzelnen Vektorkomponenten $\hat{x}(i)$ heißen *Fourierkoeffizienten* und geben die Phase und Amplitude an, mit der eine Kosinusschwingung bestimmter Frequenz im Signal x vorkommt. Die Fourierkoeffizienten stellen nur eine Approximation des Spektrums dar, siehe [10]. Bei einer Zerlegung eines Signals mit Hilfe der DFT muss also mit einer gewissen Ungenauigkeit gegenüber der Zerlegung mit Filterbank gerechnet werden. Das Ergebnis $\hat{x}$ ist symmetrisch, d.h. es gilt für alle $i \in [1:L] : \hat{x}(i) = \hat{x}(L+1-i)$. Deshalb resultieren $\left\lceil \frac{L}{2} \right\rceil$ nützliche Koeffizienten, die Aufschluss über die Energie bestimmter Frequenzkomponenten des Signals x geben. Die Koeffizienten verteilen sich gleichmäßig über einen Bereich von 0 Hz bis zur Nyquistfrequenz des Signals. Die genaue Formel zur Umrechnung des Index eines Koeffizienten in die zugehörige Frequenz, ist in Kapitel 4.3.1 zu finden, wo es darauf ankommt, die Amplitude bestimmter Frequenzen zu ermitteln.

Würde man das gesamte Signal mit einer Matrix transformieren (was schon wegen der Berechnungskomplexität schwierig wäre), so wäre jegliche Zeitinformation in der Phase versteckt und man könnte ohne Weiteres keine Aussage mehr über den zeitlichen Ablauf des Signals machen. Da der zeitliche Ablauf jedoch für das Audiomatching wichtig ist, wird das Signal zunächst mit Hilfe von Fenstern in einzelne kürzere Abschnitte zerlegt. Jeder Abschnitt wird nun getrennt mittels der DFT transformiert und man erhält pro Fenster einen Vektor mit Koeffizienten, einen sogenannten *Spektralvektor*. Zusammen ergeben diese Vektoren das *Spektrogramm* des Signals. Die Komponenten des Spektrogramms sind nach der Fouriertransformation komplex. Da die Zeitinformation durch die Fensterung berücksichtigt wird, wird keine Phaseninformation mehr benötigt. Deshalb wird von allen Vektoreinträgen der Betrag berechnet und die Koeffizienten damit reell gemacht. Die Spektralvektoren geben jetzt nur noch die Energieverteilung auf die einzelnen Frequenzen pro Fenster an. Abbildung 4.1(a) zeigt den Berechnungsvorgang schematisch. Das Ergebnis ist also eine Folge von Vektoren, sodass es sich anbietet, mit diesen Spektralvektoren direkt ein Audiomatching durchzuführen.



(a) Berechnung des Spektrogramms



(b) Spektrogramm der Anfrage; unten vergrößerter Ausschnitt

Abbildung 4.1. Schematische Darstellung der STFT-Berechnung und Visualisierung des Spektrogramms der Anfrage $\mathcal{Q}^{Sho}$, die horizontale Linie markiert die Frequenz, bis zu der die Filterbank arbeitet.

4.2 Audiomatching mit Spektralvektoren

Zuerst müssen die Fensterbreite, die Überlappung und die Fensterfunktion festgelegt werden. Gewählt wird ein Hammingfenster der Breite 4096 und eine Überlappung von 1891. Genau wie die Pitchmerkmale bei den Experimenten mit Filterbank werden die Spektralvektoren als Grundlage für weitere Merkmale genutzt, sodass man sie abspeichert, um eine erneute Berechnung zu sparen. Die hier gewählte Fensterbreite, Fensterfunktion und Überlappung gelten also auch für die folgenden STFT-basierten Merkmale. Die Fensterbreite wird, im Gegensatz zu den filterbankbasierten Merkmalen, als Zweierpotenz gewählt, da der verwendete schnelle Algorithmus zur Berechnung der DFT so optimal ausgenutzt wird, siehe [4]. Die Breite von 4096 ermöglicht Vergleiche zwischen den verschiedenen Merkmalstypen, da sie der 4410 nahe liegt. Mit einer Überlappung von 1891 erhält man aus Formel (3.9) die von der Pitchzerlegung gewohnte Merkmalsrate von 10 Vektoren pro Sekunde.

Bild 4.1(b) zeigt das Spektrogramm der Anfrage $\mathcal{Q}^{Sho}$. Im oberen Diagramm sind die Vektoren vollständig zu sehen, sie decken Frequenzen bis zur Nyquistfrequenz von 11025 Hz ab. Man erkennt, dass nur ein geringer Bereich in den unteren Frequenzen Werte größer Null annimmt. Dies hängt mit der Klangfarbe und damit der Obertonstruktur des Stückes zusammen, denn die Grundtöne liegen im Allgemeinen relativ niedrig. Die horizontale Linie markiert die Frequenz von 4186 Hz, das ist die Zentrumsfrequenz der MIDI-Note 108, also die höchste Note, die bei der Pitchzerlegung mit Filterbank berücksichtigt wird. Bei der hier gewählten Datenbank liegen keine Frequenzen über dieser Schranke, bei anderen Stücken kann dies durchaus der Fall sein, sodass beim Audiomatching mit Spektralvektoren mehr Obertöne in die Bewertung einfließen können als beim Matching mit Pitchmerkmalen. Das untere Diagramm von Abbildung 4.1(b) stellt einen bezüglich der Frequenzachse

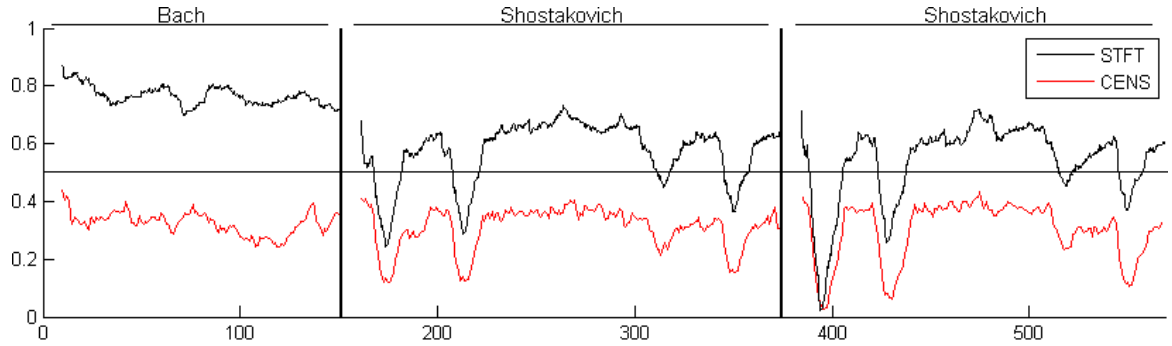
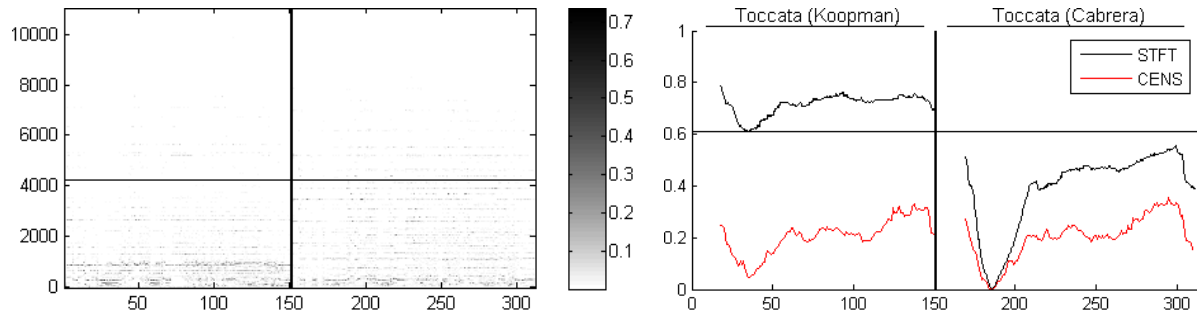


Abbildung 4.2. Matchingkurve ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$): Vergleich von STFT- mit CENS-Merkmalen, beide liefern die gleichen Treffer.

vergrößerten Ausschnitt dar. Hier ist wieder, wie bei den Pitchmerkmalen, die Melodie zu erkennen. Allerdings stellt hier eine Vektorkomponente nicht eine Note dar, sondern jede Note verteilt sich über mehrere Einträge.

Die Anfrage $\mathcal{Q}^{Sho}$ auf der Datenbank $\mathcal{D}^{Sho}$ liefert die Kurve aus Abbildung 4.2. Zum Vergleich ist auch die Kurve der CENS-Merkmale eingezeichnet, da diese für das Beispiel zu korrekten und gut erkennbaren Ergebnissen führt und man leicht die Positionen der korrekten Treffer sehen kann. Zuerst fällt wieder auf, dass sich die Kurven auf unterschiedlichen Ebenen befinden. Die STFT-Merkmale ergeben generell höhere Kosten, da sie zum einen im Frequenzbereich genauer aufgelöst und nicht zu Chroma zusammengefasst sind und zum anderen keiner statistischen Vergröberung unterzogen worden sind, sodass wesentlich mehr Details vorhanden sind, die zu Unterschieden führen können. Wenn man die Ergebnisse betrachtet, fällt auf, dass 8 korrekte und deutlich erkennbare Treffer vorliegen. Der zur Teil A_3 gehörende Treffer ist in beiden Versionen sogar deutlicher abzugrenzen als bei den Pitchmerkmalen. Offensichtlich stört die höhere Auflösung im Frequenzbereich das Ergebnis in diesem Beispiel nicht, sondern führt im Gegenteil zu einer besseren Trennung der Treffer von den nicht passenden Abschnitten.

Je höher die Frequenz ist, desto mehr Fourierkoeffizienten liegen in der Umgebung der Zentrumsfrequenz einer MIDI-Note, diese Tatsache wird in Abschnitt 4.3.1 genauer thematisiert. Als Konsequenz nimmt die Auflösung der Spektralvektoren relativ zu den Pitchmerkmalen gesehen bei höheren Frequenzen zu. Falls sich zwei Passagen also deutlich in der Klangfarbe und damit der Obertonstruktur unterscheiden, sollte die höhere Frequenzauflösung in den oberen Bereichen größere Kosten erzeugen. Um dieses Verhalten zu überprüfen, wird ein Audiomatching auf zwei Versionen von „Toccata“ durchgeführt mit Datenbank $\mathcal{D}^{BachII}$ und Anfrage $\mathcal{Q}^{Bach}$. Die beiden Orgelstücke sind obertonreich und unterscheiden sich stark in der Klangfarbe, da sie von zwei verschiedenen Orgeln stammen. Abbildung 4.3(a) zeigt die Merkmalsvektoren. Man sieht, dass die höheren Frequenzen stärker vertreten sind als in der Datenbank mit den Stücken von Shostakovich. Die Grenze von 4186 Hz wird diesmal überschritten, damit sind auch Frequenzen vorhanden, die bei der Filterbankzerlegung gar nicht berücksichtigt werden. Die zugehörige Matchingkurve ist in Bild 4.3(b) zu erkennen. Zusätzlich ist auch die CENS-Kurve eingezeichnet, welche



(a) Spektrogramm (b) Matchingkurve ($\mathcal{D}^{BachII}$, $\mathcal{Q}^{Bach}$)

Abbildung 4.3. Audiomatching auf zwei Versionen von „Toccata“, die beiden Stücke sind auf unterschiedlichen Orgeln gespielt worden und unterscheiden sich in der Klangfarbe. Die Kurve der CENS-Merkmale zeigt deutliche Treffer. Die im Frequenzbereich höher aufgelösten STFT-Vektoren sind weit von einem Treffer im ersten Stück entfernt.

die beiden tatsächlichen Vorkommen der Anfrage deutlich markiert. Die Kurve der Spektralvektoren liegt jedoch in der ersten Version des Stückes in jedem Punkt höher als der höchste Punkt des zweiten Teils. Die andere Klangfarbe der Orgel bewirkt, dass durchgehend hohe Kosten entstehen und auch die Stelle, die melodisch ähnlich ist, nur relativ zum Stück niedrige Kosten liefert, jedoch nicht insgesamt.

Ein entscheidender Nachteil der Spektralvektoren ist ihr Speicherplatzbedarf. Mit den hier gewählten Parametern hat eine Merkmalsdatei etwa die 3.5-fache Größe der WAV-Datei, aus der sie extrahiert worden ist. Beim Abspeichern der Spektralvektoren für eine umfangreiche Musikdatenbank ergäben sich so gewaltige Datenmengen, während man im Allgemeinen eher an Merkmalen interessiert ist, die weniger Speicherplatz als die WAV-Dateien benötigen. Nun sollen die Spektralvektoren jedoch weiterverarbeitet werden.

4.3 Approximation von Filterbänken

Bei den in Kapitel 3 vorgestellten Merkmalen wird das Signal mit Hilfe einer Filterbank in verschiedene, den MIDI-Noten entsprechende, Subbänder zerlegt. Alternativ kann stattdessen auch eine gefensterte Fouriertransformation durchgeführt werden. Die Fourierkoeffizienten werden so zu Bins zusammengefasst, dass mit jedem Bin eine Note korrespondiert. Die Ergebnisse des Audiomatchings mit Merkmalen, die mit Hilfe einer Filterbank gewonnen werden, sollen nun verglichen werden mit den Ergebnissen der mittels STFT generierten Merkmale.

4.3.1 Pitchzerlegung

Zuerst wird eine gefensterte Fouriertransformation durchgeführt. Fensterbreite und Überlappung sollen so gewählt werden, dass ein Vergleich mit den durch Filterbank gewonnenen Merkmalen möglich ist. Man erhält ausgehend vom Signal x eine Folge von Vektoren

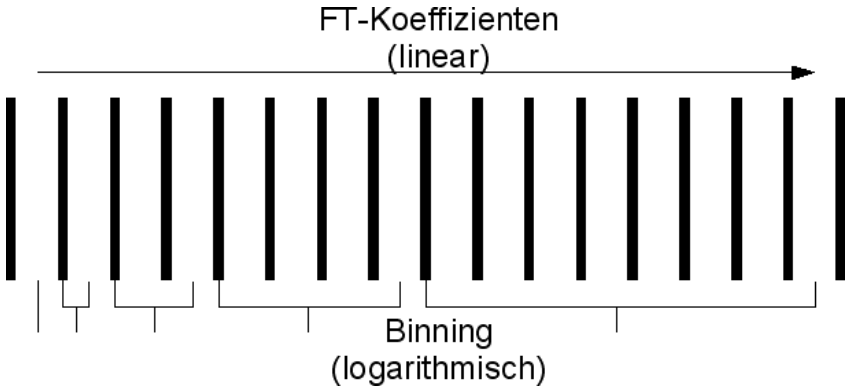


Abbildung 4.4. Fourierkoeffizienten werden zu Pitches zusammengefasst, dabei kann es vorkommen, dass für einen Pitch kein Koeffizient vorliegt.

$(\hat{x}_n)_{n\in[1:N]}$, die für jedes Fenster die Fourierkoeffizienten beinhaltet, dabei gibt N die Zahl der Fenster an, in die das Signal aufgeteilt ist. Die Zahl der Koeffizienten pro Vektor entspricht der Fensterbreite in Samples. Wie bereits erwähnt sind wegen der Symmetrie der aus der DFT resultierenden Vektoren nur die Hälfte der Koeffizienten relevant, sodass man bei einer Fensterbreite von L mit $\lceil\frac{L}{2}\rceil$-dimensionalen Vektoren arbeitet. Die einzelnen Koeffizienten geben nun die Energieverteilung von 0 Hz bis zur Nyquistfrequenz an äquidistanten Stellen wieder. Bei einer Nyquistfrequenz von F Hz repräsentiert der k-te Koeffizient ($k \in \left[0 : \lceil\frac{L}{2}\rceil - 1\right]$) also eine Frequenz von

$$f_c(k) := \frac{2k \cdot F}{w}^1 \tag{4.2}$$

Bei einer Abtastrate von 22050 Hz und damit einer Nyquistfrequenz von 11025 Hz ergibt sich bei einer Fensterbreite von 4096 Samples beispielsweise für den 512. Koeffizienten eine Frequenz von 2756.25 Hz. Zusammen mit der Frequenzformel für die MIDI-Noten $f_m(p) = 2^{\frac{p-69}{12}} \cdot 440$ Hz kann jetzt jeder der Fourierkoeffizienten einem Pitch zugeordnet werden. Es werden für jeden Vektor $\hat{x}_n$ des Spektrogramms und für jeden Pitch p die Quadrate der Fourierkoeffizienten, die zu diesem Pitch gehören, aufaddiert:

$$\sum_{\{k|f_m(p-0.5)\leq f_c(k)<f_m(p+0.5)\}} |\hat{x}_n(k)|^2 \tag{4.3}$$

Auf diese Weise erhält man wieder Merkmalsvektoren, mit denen später Matchingaufgaben durchgeführt werden. In Abbildung 4.4 ist das Zusammenfassen der Koeffizienten veranschaulicht. Man erkennt auch das Problem, dass für manche Noten kein Koeffizient vorliegt. Dies liegt daran, dass bei tiefer werdenden Frequenzen die Frequenzbänder der Noten immer schmaler werden und damit weniger Fourierkoeffizienten pro Note vorliegen.

[1] Der Index c gibt an, dass mit der Formel die Frequenz eines Fourierkoeffizienten berechnet wird. Im Gegensatz dazu wird bei der Formel für die Berechnung der Frequenz aus einer MIDI-Nummer der Index m verwendet.

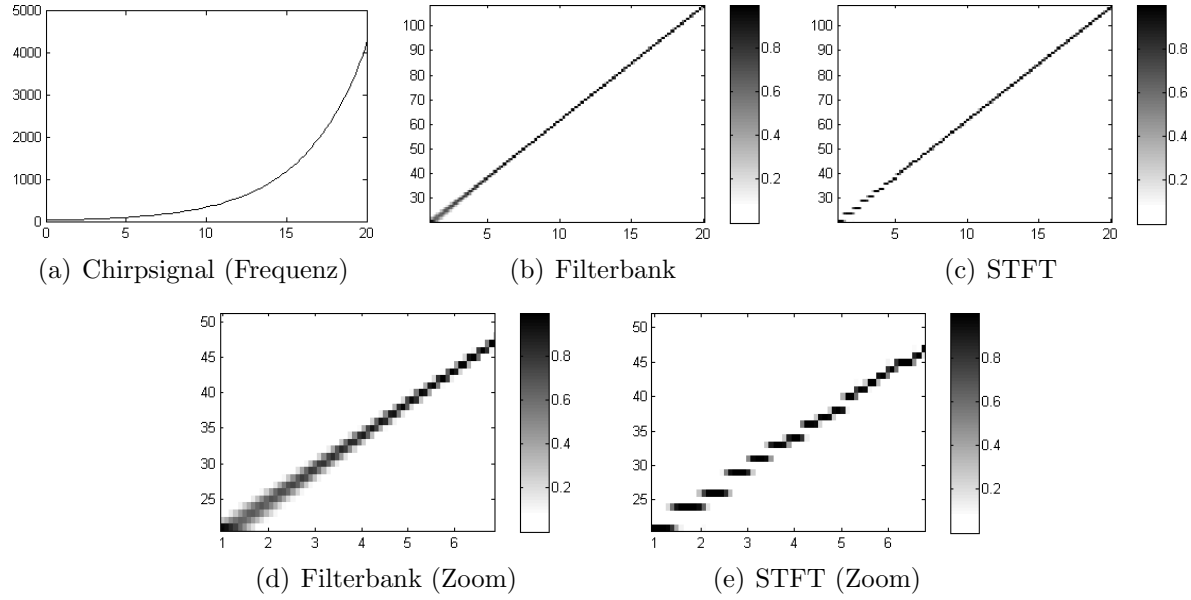
In manchen Fällen kommt es vor, dass für einen Pitch p kein Koeffizient k existiert mit $f_m(p-0.5) \leq f_c(k) < f_m(p+0.5)$. Die entsprechende Note ist dann in allen Merkmalsvektoren mit Null belegt, egal welche Frequenzverteilung vorliegt. Es handelt sich hier also um den nächsten Unterschied zu den Pitchmerkmalen, die mit Filterbank erzeugt werden, neben der Tatsache, dass die DFT nur eine Approximation ist.

Einen weiteren Unterschied stellen die Grenzen zwischen zwei Pitches dar. Hier ist das arithmetische Mittel zwischen zwei MIDI-Pitch-Nummern gewählt worden, was auf Frequenzebene wegen $2^{\frac{a+b}{2}} = \sqrt{2^{a+b}} = \sqrt{2^a \cdot 2^b}$ dem geometrischen Mittel entspricht. Die Zentrumsfrequenz liegt hier also im geometrischen Mittel des Intervalls, während sich bei der Zerlegung mit Filterbank die Zentrumsfrequenz im arithmetischen Mittel befindet. Die aus den Spektralvektoren resultierenden Merkmale werden im Folgenden mit den filterbankbasierten Pitchmerkmalen verglichen.

4.3.2 Audiomatching mit STFT-Pitchmerkmalen

Für die folgenden Experimente wird die Fensterbreite auf 4096 Samples gesetzt und die Überlappung auf 1891, woraus nach Formel (3.9) die übliche Rate von zehn Merkmalsvektoren pro Sekunde resultiert. Um die Unterschiede besser sehen zu können, wird zunächst von einem Chirpsignal eine Pitchzerlegung mittels Filterbank und eine per Fouriertransformation durchgeführt. Das Chirpsignal startet bei einer Frequenz von 26 Hz und reicht bis 4300 Hz (siehe Abbildung 4.5(a)), es deckt damit recht genau den Bereich der MIDI-Noten 21 bis 108 ab. Weiterhin steigt die Frequenz des Signals exponentiell, sodass man in den Merkmalsdiagrammen 4.5 eine lineare Entwicklung beobachten kann. Die Abbildungen 4.5(b) und 4.5(c) zeigen die Ergebnisse. Beide Verfahren ordnen den aktuellen Pitch einer MIDI-Note zu, sodass sich aufsteigende Folgen von Noten ergeben, die sich in den beiden Diagrammen sehr ähneln. In den Abbildungen 4.5(d) und 4.5(e) sind vergrößerte Ausschnitte des niedrigen Frequenzbereiches zu sehen. Bei beiden Arten der Zerlegung fällt eine Überlappung der einzelnen Noten auf. Diese lässt sich durch die Überlappung der Fenster bei STMSP-Berechnung und der gefensterten Fouriertransformation erklären. Zusätzlich kann es bei STFT zu Überlappungen kommen, weil innerhalb eines Fensters wegen der steigenden Frequenz mehrere Fourierkoeffizienten ungleich Null werden können. Ein weiterer Grund für Überlappungen bei der Filterbankversion sind die Überschneidungen der Frequenzantworten der verwendeten Filter. Weitere (aber deutlich schwächere) Überlappungen treten bei der STFT-Version durch Artefakte der DFT, die ja nur eine Approximation ist, auf. Deutlich zu erkennen ist auch das oben angesprochene Problem der fehlenden Fourierkoeffizienten für manche MIDI-Noten. Bei der Zerlegung mit STFT wird in den Merkmalsvektoren bei diesen MIDI-Noten nie ein Wert größer Null auftreten. Musikstücke, die Noten in den entsprechenden Frequenzbereichen enthalten, können auf diese Weise zu fehlerhaftem Verhalten der STFT-Pitchmerkmale führen.

Nun wird ein Audiomatching durchgeführt. Als Datenbank wird $\mathcal{D}^{BachII}$ mit zwei Versionen von „Toccata“ gewählt, da das Orgelstück einen relativ großen Frequenzumfang hat, Anfrage ist $\mathcal{Q}^{Bach}$. Die Merkmale aus den Abbildungen 4.6(a) und 4.6(b) sehen im Groben ähnlich aus. Bei genauerem Hinsehen erkennt man jedoch leichte Unterschiede. So ist



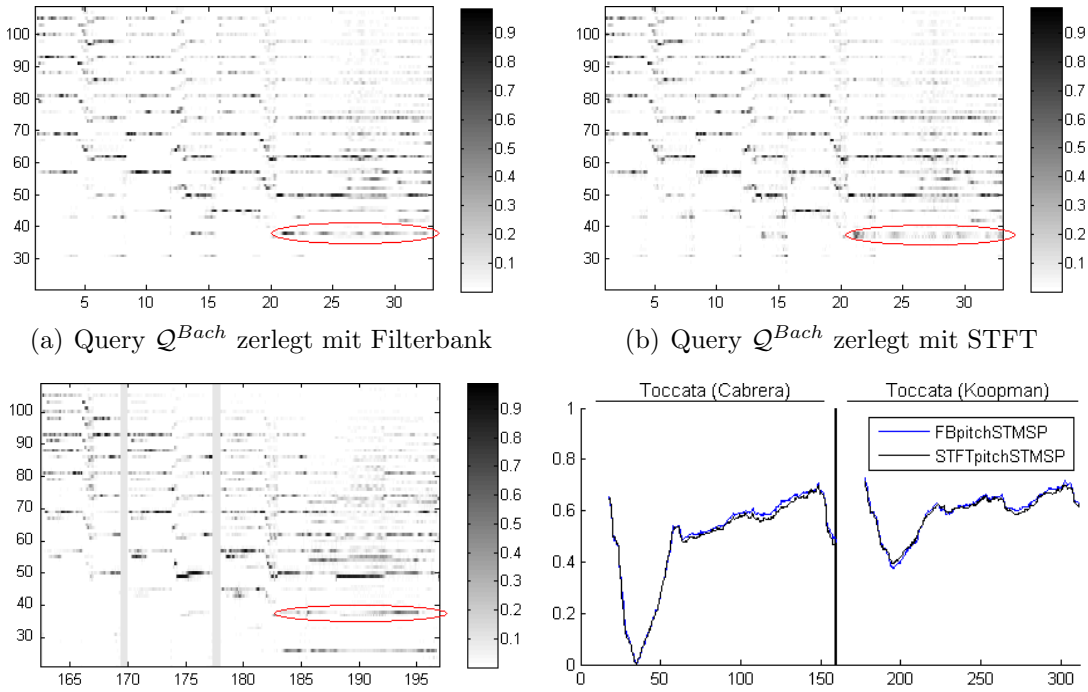
(a) Chirpsignal (Frequenz) (b) Filterbank (c) STFT

(d) Filterbank (Zoom) (e) STFT (Zoom)

Abbildung 4.5. Pitchzerlegung eines Chirpsignals mit Filterbank und mit Fouriertransformation. In der vergrößerten Darstellung sind Unterschiede zwischen den Ergebnissen zu erkennen, insbesondere weist die Zerlegung mit STFT Lücken auf. Diese entstehen, weil in den Frequenzbändern mancher MIDI-Noten keine Fourierkoeffizienten liegen.

bei den filterbankbasierten Merkmalen an der markierten Stelle ein längeres Band einer MIDI-Note zu sehen. Bei der Zerlegung mit Spektralvektoren zerfällt dieses Band auf zwei MIDI-Noten. Ursache ist die Wahl der Grenzfrequenz zwischen zwei Noten. Die Energie der betreffenden Note befindet sich vollständig in einem Band der Filterbank, jedoch überschreiten manche der vorhandenen Frequenzen die tiefer liegende obere Pitchgrenze der Zerlegung mittels STFT. Dieses Phänomen ist in Grafik 4.7 veranschaulicht. Das gleiche Phänomen tritt jedoch auch in der Datenbank auf, siehe Abbildung 4.6(c), sodass dieser Unterschied auf Ebene der Matchingkurve teilweise kompensiert werden sollte. Außerdem nähern sich die Pitchgrenzen bei höheren Frequenzen einander an, sodass das Phänomen nur in niedrigen Frequenzbereichen auftritt. Abbildung 4.6(d) zeigt die Matchingkurven der Pitchmerkmale, die mit Filterbank bzw. mit STFT erzeugt worden sind. Die Kurven sind sehr ähnlich, man erkennt nur wenige sehr geringe Abweichungen. Offenbar bleiben die wesentlichen Eigenschaften der Pitchmerkmale auch bei der Approximation erhalten.

Insgesamt sind beim Betrachten der Merkmalsvektoren einige kleinere Abweichungen von den Resultaten der Filterbank zu erkennen. Eine Zerlegung mittels STFT bleibt damit nur eine Approximation für die Filterbank. Da sich jedoch auch bei allen anderen im weiteren Verlauf vorkommenden Beispielen gezeigt hat, dass die Abweichungen auf Ebene der Matchingkurven sehr gering sind, werden Spektralvektoren im Folgenden als flexibler Ausgangspunkt für unterschiedliche Binnings verwendet. So können zum Experimentieren sehr schnell und einfach verschiedene Zusammenfassungen durchgeführt werden, wenn die Spektralvektoren einmal abgespeichert vorliegen.



(a) Query $\mathcal{Q}^{Bach}$ zerlegt mit Filterbank

(b) Query $\mathcal{Q}^{Bach}$ zerlegt mit STFT

(c) Abschnitt (2. Stück) Datenbank $\mathcal{D}^{BachII}$ zerlegt mit Filterbank

(d) Matchingkurven ($\mathcal{D}^{BachII}$, $\mathcal{Q}^{Bach}$)

Abbildung 4.6. Vergleich der Matchingkurven von Pitchmerkmalen, die mit Filterbank und mit STFT generiert worden sind. In den Merkmalen sind leichte Unterschiede zu erkennen. Die Kurven sind nicht identisch aber sehr ähnlich.

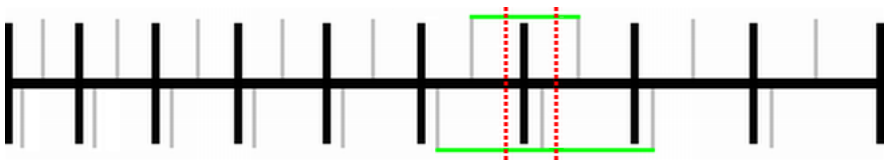


Abbildung 4.7. Die Grenzen zwischen den Pitches bei Zerlegung mit Filterbank (obere Hälfte) und mit STFT sind grau dargestellt, die Zentrumsfrequenzen durch dicke schwarze Balken. Es kann sein, dass sich zwei Frequenzen (gestrichelte Linien), die im Signal vertreten sind, nach Zerlegung mit STFT in einem Bin befinden und nach Filterung in zweien, bzw. umgekehrt. Das Beispiel ist zur Verdeutlichung übertrieben, tatsächlich nähern sich die Grenzen bei höheren Frequenzen einander.

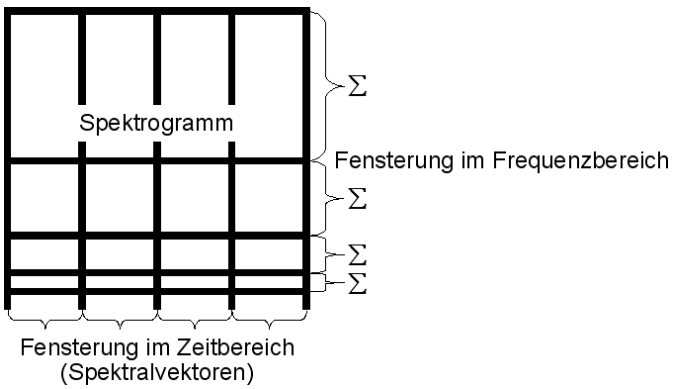


Abbildung 4.8. Fensterung im Frequenzbereich: Die Koeffizienten der Spektralvektoren werden in Fenster aufgeteilt und innerhalb jedes Fensters summiert.

4.4 Verschiedene Binnings

Die Fourierkoeffizienten können nicht nur zu MIDI-Pitches, sondern auch auf beliebige andere Art und Weise zusammengefasst werden. Als Verallgemeinerung zu den Pitchmerkmalen ist ein Konzept entwickelt worden, mit welchem ausgehend von Spektralvektoren mit geringem Aufwand unterschiedliche Binnings produziert und getestet werden können. Alle im Folgenden untersuchten Binnings sind an der logarithmischen Wahrnehmung der Tonhöhe orientiert, die sich in der exponentiellen Verteilung der Noten auf die Frequenzen niederschlägt. Die Anzahl der Fourierkoeffizienten, die zusammengefasst werden, wird also mit steigender Frequenz exponentiell zunehmen. Die untersuchten Binnings sind noch grundsätzlich an der MIDI-Pitch-Struktur orientiert.

4.4.1 Berechnung der Binnings

Das Zusammenfassen von Koeffizienten innerhalb eines Spektralvektors kann auch als eine Fensterung des Vektors mit anschließender Addition aller Werte pro Fenster aufgefasst werden. Nach der gefensterten Fouriertransformation wird also eine Fensterung aller Spektralvektoren durchgeführt, siehe Abbildung 4.8. Daher stammt auch der Name „spectralwindow“ des Programms zur Berechnung aller in diesem Kapitel vorgestellten Binnings, siehe Anhang A.5. Bei den folgenden Erklärungen ist immer zu beachten, dass sich die erwähnten Fenster nicht mehr über Samples erstrecken, sondern über Frequenzbereiche.

Anhand der Parameter, die das Programm übernimmt, werden das allgemeine Vorgehen und die einzelnen Berechnungsschritte erläutert. Grundsätzlich geht es bei der Parametrisierung darum, die Start- und Endfrequenzen der einzelnen Frequenzfenster festzulegen. Danach fasst der Algorithmus die Koeffizienten entsprechend zusammen. Zur intuitiveren Bedienung werden bei frequenzbezogenen Parametern nicht die Frequenzen in Hertz angegeben, sondern MIDI-Pitch-Nummern. Die Umrechnung in Frequenzen erfolgt programmintern. Durch Angabe von gebrochenen Pitchwerten könnte auch auf diese Weise jede beliebige Frequenz gewählt werden, was jedoch sehr umständlich wäre. Da im Fol-

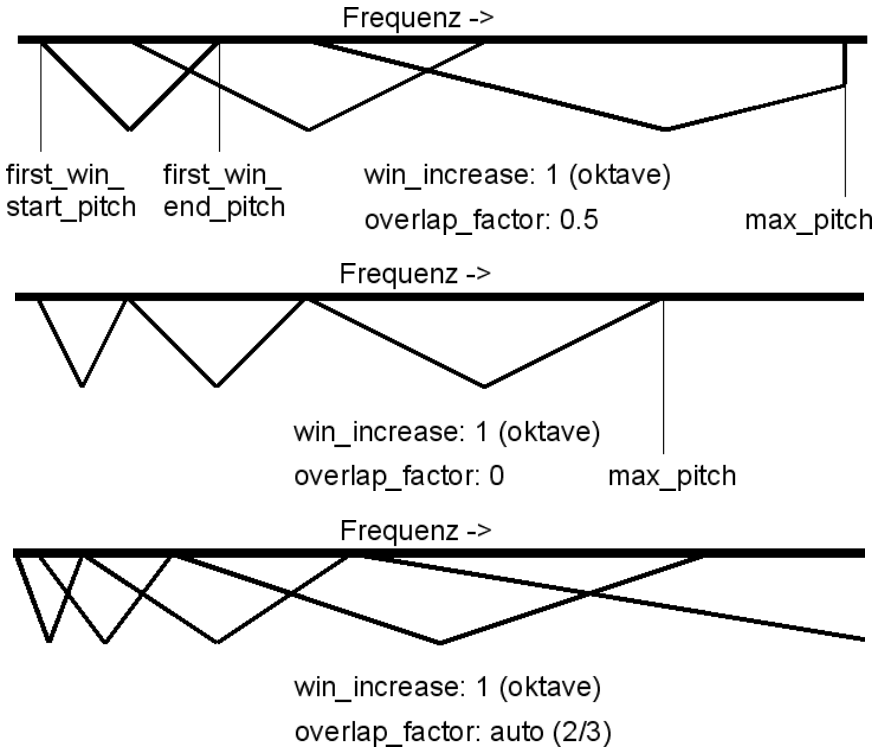


Abbildung 4.9. Bestimmung der Fenstergrenzen abhängig von fünf Parametern, die Grenzen des ersten Fensters, Zunahme der Fenstergröße, Überlappung und Obergrenze des gesamten Binnings festlegen. Hier sind zur übersichtlicheren Darstellung Dreiecksfenster abgebildet. Die Fensterfunktion ist jedoch einstellbar, häufig werden Rechteckfenster benutzt.

genden Binnings betrachtet werden, die an der MIDI-Noten-Struktur orientiert sind, stellt dies aber keine Einschränkung dar.

Die Start- und Endfrequenzen, bzw. Pitches, des „ersten“ Fensters, also des Fensters mit dem niedrigsten Startwert, werden explizit angegeben. „first_win_start_pitch“ stellt die untere Grenze dar. Bevor der Wert mit der Formel (3.6) in eine Frequenz umgerechnet wird, wird er noch um 0.5 verringert. So erhält man, wie bei der Approximation der Filterbank, das geometrische Mittel von der gegebenen MIDI-Note und der darunterliegenden Note, also die untere Grenzfrequenz der angegebenen Note. Dies hat den Sinn, dass das Fenster die Startnote noch komplett enthält und nicht erst bei ihrer Zentrumsfrequenz beginnt. „first_win_end_pitch“ gibt die obere Grenze des ersten Fensters an. Dieser Wert wird erst um 0.5 erhöht und dann mittels Formel (3.6) umgerechnet. Analog zur Verringerung der Startfrequenz hat hier die Erhöhung den Sinn, dass das Fenster den Pitch noch komplett einschließt. Gibt man also sowohl für den Start- als auch für den Endpitch den gleichen Wert an, so umfasst das erste Fenster genau eine MIDI-Note nach der Konvention des geometrischen Mittels als Grenze, wie in Abschnitt 4.3.1.

Ausgehend vom ersten Fenster werden nun unter Berücksichtigung zweier weiterer Parameter die Start- und Endfrequenzen aller anderen Fenster berechnet. Prinzipiell sieht das Programm eine Überlappung der Fenster vor, ist diese jedoch auf Null eingestellt, so

entspricht die Startfrequenz eines Fensters der Endfrequenz des vorhergehenden Fensters. Mit dem Parameter „overlap_factor“ wird die Überlappung anteilig zum Vorgängerfenster eingestellt. Wird der Faktor z.B. auf 0.5 gesetzt, so beginnt jedes Fenster in der (arithmetischen) Mitte des vorhergehenden Fensters, siehe Bild 4.9(a). Durch Angabe eines negativen Überlappungsfaktors könnte man auch einen Abstand zwischen den Fenstern erzielen. Der Grund für die anteilige und nicht absolute Bestimmung der Überlappung ist die variable Fensterlänge. Wie bereits erwähnt soll die Fensterung der exponentiellen Zunahme der Zentrumsfrequenzen und der Breiten der MIDI-Pitches angepasst sein. Dies wird dadurch erreicht, dass die Länge jedes Fensters mit einem bestimmten Faktor multipliziert wird, um die Länge des nächsten Fensters zu berechnen. Mit Hilfe des Parameters „win_increase“ wird dieser Faktor eingestellt, er ergibt sich aus $2^{\text{win\_increase}}$. Das Potenzieren hat wieder den Sinn, die Werte der Parameter an die Struktur der MIDI-Noten anzupassen. Wählt man „win_increase“ als 1, dann wird die Länge des Fensters jedesmal verdoppelt, was einer Oktave entspricht. Bei 0 erhält man eine konstante Fensterlänge. Durch Angabe von negativen Werten könnte man auch eine Verkleinerung der Fenster bewirken.

Gibt man den Parameter „overlap_factor“ gar nicht an, wird die Überlappung automatisch so gewählt, dass jedes Fenster gerade das übernächste berührt und ein Fenster dazwischen liegt, das beide überlagert, siehe Abbildung 4.9(c). Sei $v := 2^{\text{win\_increase}}$ der berechnete Vergrößerungsfaktor zum nächsten Fenster, so wird der Überlappungsfaktor, falls er nicht angegeben ist, auf folgende Weise bestimmt:

$$\text{overlap\_factor} := \frac{v}{v+1} \tag{4.4}$$

Startet ein Fenster mit Breite b am Punkt s, so reicht es bis $s+b$ und das nächste Fenster beginnt bei $s+b-\frac{v}{v+1}\cdot b$ und hat Breite $v\cdot b$. Damit endet dieses Fenster bei $s+b-\frac{v}{v+1}\cdot b+v\cdot b$ und das wiederum nächste Fenster beginnt bei

$$s+b-\frac{v}{v+1}\cdot b+v\cdot b-\frac{v}{v+1}\cdot vb = s+b\cdot\left(1-v\cdot\left(\frac{1}{v+1}-1+\frac{v}{v+1}\right)\right) = s+b$$

Das erste Fenster endet also genau dort, wo das übernächste beginnt.

Der Parameter „max_pitch“ legt eine obere Grenze fest, kein Fenster überschreitet diesen Wert. Damit legt der Parameter indirekt die Anzahl der Fenster fest. Beginnt ein Fenster vor „max_pitch“, überschreitet jedoch diesen Wert, so wird es an der Grenze abgeschnitten. Ergäbe sich bei einer bestimmten Einstellung beispielsweise ein Fenster, das von $p = 100$ bis $p = 110$ reicht und ist „max_pitch“ auf 108 gesetzt, so würde daraus ein Fenster mit dem Bereich $p = 100$ bis $p = 108$. In Grafik 4.9(a) ist dieses Abschneiden zu erkennen. Sobald ein Fenster den Wert erreicht oder überschritten hat, wird kein weiteres mehr erzeugt. Auch dies ist in der Grafik zu sehen: Wegen der Überlappung müsste eigentlich in der Mitte des letzten Fensters noch ein weiteres beginnen, dies ist nicht der Fall, weil bereits ein Fenster die Grenze erreicht hat. Bei den hier vorgestellten Binnings ist die obere Grenze immer so gewählt, dass dort gerade die berechnete Endfrequenz eines Fensters liegt, so kommt es nicht zum Abschneiden, siehe Abbildung 4.9(b). Der Wert „max_pitch“ wird vor der

Umrechnung in Frequenz um 0.5 erhöht, damit die Obergrenze des letzten Fensters auf der Grenzfrequenz und nicht der Zentrumsfrequenz einer MIDI-Note liegt.

Schließlich kann über den Parameter „window“ der Typ der Fensterfunktion gewählt werden, z.B. *Rechteckfenster*, *Dreiecksfenster* oder *Hammingfenster*. Die Fensterfunktion bestimmt, wie die einzelnen Koeffizienten innerhalb der Fenster gewichtet werden.

Das Zusammenfassen geschieht nun auf folgende Weise: Zunächst werden für ein bestimmtes Fenster die Quadrate der zugehörigen Koeffizienten aus allen Spektralvektoren in eine Matrix geschrieben. Sei $(\hat{x}_n)_{n\in[1:N]}$ die Folge der N Spektralvektoren und decke ein Fenster die Koeffizienten von Index a bis e ab, dann lautet die Matrix $\left(|\hat{x}_n(i)|^2\right)_{n\in[1:N],i\in[a:e]}$. In den Zeilen der Matrix stehen die Koeffizienten pro Spektralvektor. Durch eine Multiplikation der Matrix mit einem Vektor können die einzelnen Koeffizienten gewichtet summiert werden. Sei $\mathfrak{W} = (\mathfrak{w}_a, \mathfrak{w}_{a+1}, \ldots, \mathfrak{w}_e)$ ein Vektor, der ein Fenster realisiert, welches von Koeffizient a bis e reicht, d.h. die einzelnen Einträge des Vektors entsprechen den Werten der Fensterfunktion. Indem man für alle Fenster die Matrixmultiplikation

$$\left(|\hat{x}_n(i)|^2\right)_{n\in[1:N],i\in[a:e]} \cdot \mathfrak{W}^T \tag{4.5}$$

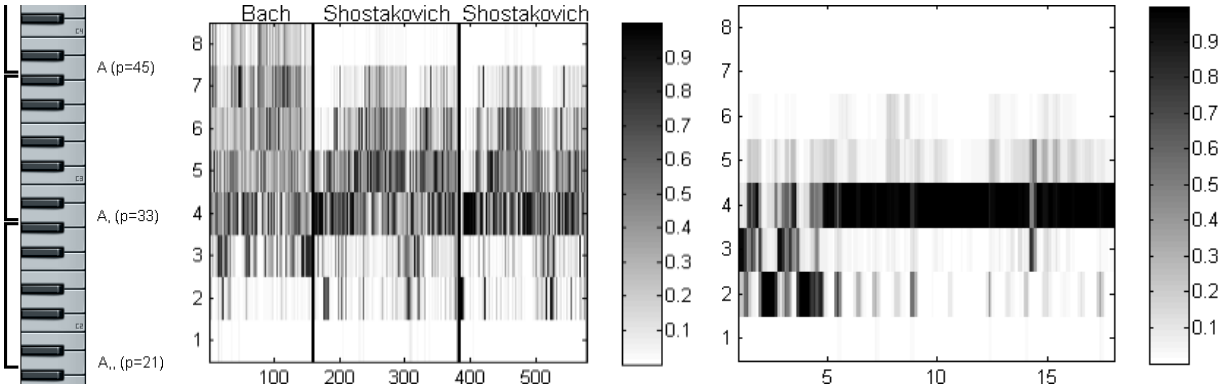
durchführt, erhält man das gesamte Binning.

Beispielhaft werden jetzt die Einstellungen genannt, mit denen das Pitchbinning aus Abschnitt 4.3.1 realisiert worden ist. Als Fensterfunktion „window“ kommt das Rechteckfenster zum Einsatz, da sämtliche Koeffizienten, die zu einem Pitch aufaddiert werden, gleich gewichtet werden sollen. „first_win_start_pitch“ und „first_win_end_pitch“ werden beide auf 21 festgelegt, so deckt das erste Fenster genau die MIDI-Note 21 ab, wie es in 4.3.1 beschrieben worden ist. Da sich die einzelnen Pitches nicht überlappen, wird „overlap_factor“ auf 0 gesetzt, zwei benachbarte Fenster berühren sich so gerade in den Endpunkten. Nach der Frequenzformel für die MIDI-Pitches nimmt die Frequenz bei jeder Note um den Faktor $2^{\frac{1}{12}}$ zu. Dies entspricht einer Unterteilung der Oktave in zwölf Schritte mit gleichmäßiger exponentieller Zunahme. Setzt man „win_increase“ auf $\frac{1}{12}$, wird genau dieser Vergrößerungsfaktor zum nächsten Fenster erreicht. Der höchste Pitch, der berücksichtigt werden soll, ist $p = 108$, deshalb wird „max_pitch“ auf 108 gesetzt. Mit diesen Einstellungen wird also genau die vorgestellte Pitchzerlegung mittels STFT umgesetzt.

Im Folgenden werden verschiedene Binnings erzeugt und untersucht. Die Fensterbreite der zugrundeliegenden Spektralvektoren beträgt immer 4096 Samples und die Überlappung 1891. Damit liegen wie üblich 10 Merkmalsvektoren pro Sekunde vor.

4.4.2 Oktavzerlegung

Es sollen nun gesamte Oktaven zusammengefasst werden, dabei stellt sich die Frage, über welchen Notenbereich sich eine Oktave erstrecken soll. Zunächst startet die unterste Oktave beim MIDI-Pitch 21, also beim a. Die Oktaven reichen so immer von einem a bis zum nächsten $g\#$, siehe Abbildung 4.10(a). Damit ergeben sich folgende Einstellungen für das Programm: „first_win_start_pitch“ = 21 und „first_win_end_pitch“ = 32, dies entspricht der untersten Oktave. „win_increase“ wird auf 1 gesetzt, sodass sich die Fensterlänge jedesmal



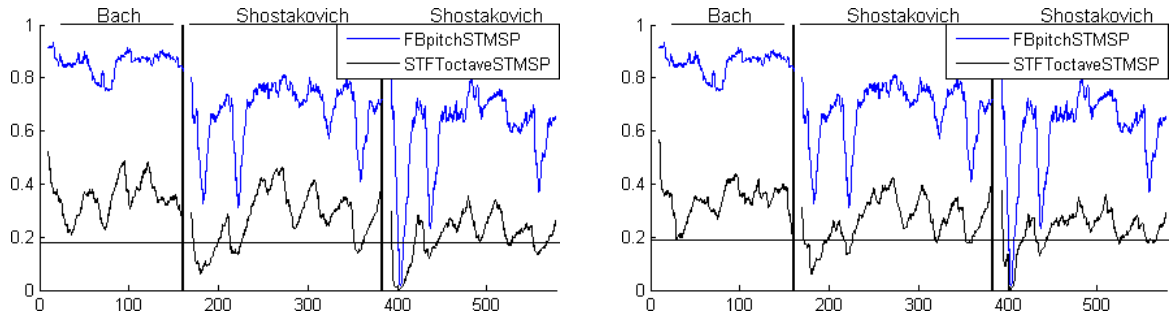
(a) Oktavzerlegung (b) Merkmale der Datenbank $\mathcal{D}^{Sho}$ (c) Merkmale der Anfrage $\mathcal{Q}^{Sho}$

Abbildung 4.10. Oktavzerlegung mit Oktaven von a bis $g\#$

verdoppelt und man zusammen mit der gewählten Überlappung von Null eine Oktavzerlegung erhält. Die Obergrenze wird mit $p = 116$ etwas höher gesetzt als üblich, damit es nicht zum Abschneiden kommt und es sich auch beim letzten Fenster um eine vollständige Oktave handelt. Als Fensterfunktion wird ein Rechteckfenster gewählt.

Abbildung 4.10(b) zeigt die Merkmale der Datenbank $\mathcal{D}^{Sho}$. In Grafik 4.10(c) ist die Merkmalsfolge der Anfrage $\mathcal{Q}^{Sho}$ zu sehen. Durch die sehr grobe Einteilung geht viel Struktur verloren. Der sonst gut erkennbare Melodieverlauf liegt fast vollständig in einer Oktave und produziert ein langes, nicht weiter differenziertes Band, das etwa von Sekunde 5 bis zum Ende reicht und nur minimal unterbrochen wird. Beim Betrachten der Merkmalsfolge der Datenbank fällt auf, dass dennoch gewisse Informationen erhalten bleiben. Das Stück von Bach zeigt eine deutlich andere Verteilung der Energie auf die Oktaven als die Stücke von Shostakovich. Da das Orgelstück obertonreicher ist als das Orchesterstück, sind die beiden oberen Oktaven in „Toccata“ wesentlich stärker vertreten. Im Gegensatz dazu sehen sich die beiden Versionen von Shostakovich nach dem groben Binning und der damit verbundenen Vernachlässigung vieler Details sehr ähnlich.

Die Matchingkurve ist in Abbildung 4.11(a) dargestellt. Zum Vergleich ist auch die Kurve der Pitchmerkmale eingezeichnet. Die Matchingkurve der Oktavmerkmale verläuft auf niedrigem Niveau und weist viele lokale Peaks auch an Positionen ohne Treffer auf. Die generell niedrigen Kosten im Vergleich zu den Pitchmerkmalen sind durch den großen Informationsverlust beim Zusammenfassen ganzer Oktaven zu erklären. Wie in einigen Beispielen von stark vergröberten Merkmalen zuvor gehen Details verloren, die Unterschiede zwischen Datenbank und Anfrage herausstellen würden. Genauso ist auch die große Zahl zusätzlicher Peaks zu erklären. Im vorliegenden konkreten Beispiel kann dieses Phänomen genauer betrachtet werden: Die meisten Merkmalsvektoren der Anfrage, siehe Bild 4.10(c), enthalten in der vierten Komponente einen sehr großen Eintrag und die anderen Einträge sind gering. Die Struktur, die innerhalb der Oktave liegt und zu dem hohen Anteil der Oktave geführt hat, ist nicht mehr vorhanden. Wenn nun in der Datenbank an einer Stelle auch der größte Anteil in der vierten Komponente liegt, egal wie die Struktur innerhalb



(a) Oktave a - g# (b) Oktave c - h

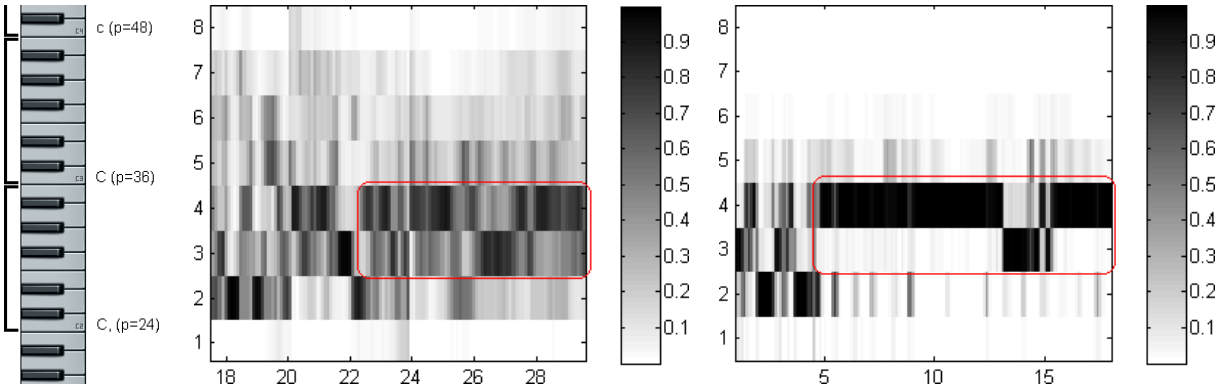
Abbildung 4.11. Matchingkurven vom Matching mit Oktavmerkmalen ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$). Die Kurven ändern sich, wenn der Notenbereich, über den sich die Oktaven erstrecken, verändert wird. In 4.11(b) gibt es acht korrekte Treffer, in 4.11(a) nur sieben. Zum Vergleich ist die Kurve vom Matching mit Pitchmerkmalen eingezeichnet.

der Oktave auch aussieht, dann sinken die Kosten und man erhält, je nachdem wie lang diese Passage ist, ein mehr oder weniger tiefes lokales Minimum.

Trotz der sehr starken Vergröberung und der großen Zahl von Peaks sind die ersten sieben Treffer von insgesamt acht gewünschten Treffern korrekt. Obwohl viel an Information verloren gegangen ist, kann zumindest bei diesem Beispiel noch eine Unterscheidung von passenden und nicht passenden Abschnitten getroffen werden. Allerdings setzen sich die Treffer auch nicht so deutlich ab wie z.B. bei den Pitchmerkmalen.

Bisher sind die Oktaven willkürlich auf den Bereich von a bis $g\#$ festgelegt worden. Da eine Oktave viele Koeffizienten abdeckt, kann ein Verschieben der Grenzen zu einem deutlich anderen Binning führen und sollte einen Unterschied auf Matchingebene bewirken. Um dies zu untersuchen, werden die Oktaven auf einen Bereich von c bis h eingestellt. Dazu ändert man Start- und Endpitch des ersten Fensters auf $p = 24$ bzw. $p = 35$ und erhöht den maximalen Pitch auf $p = 119$, damit die letzte betrachtete Oktave vollständig ist und die Anzahl von acht Oktaven erhalten bleibt. Die anderen Parameter bleiben unverändert. Datenbank und Anfrage sind weiterhin $\mathcal{D}^{Sho}$ bzw. $\mathcal{Q}^{Sho}$.

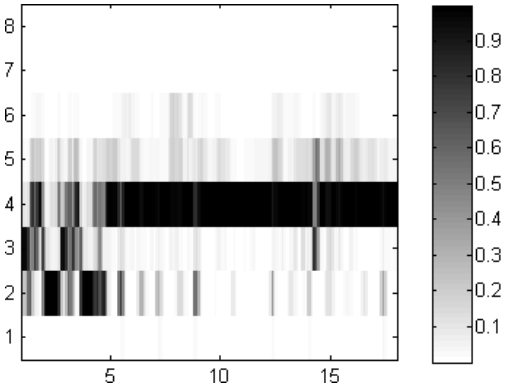
In Abbildung 4.11(b) ist die Matchingkurve dargestellt. In diesem Fall sind alle acht Treffer korrekt. Die Verschiebung der Oktavgrenzen hat also einen deutlichen Unterschied bewirkt. Genauer zu sehen ist dieser Unterschied beim direkten Vergleich der Merkmalsfolge nach der Zerlegung von a bis $g\#$ (siehe 4.12(c)) mit der Merkmalsfolge nach der Zerlegung von c bis h (siehe 4.12(d)). Während der Melodieverlauf ab der fünften Sekunde der Anfrage fast vollständig in einer Oktave zwischen a und $g\#$ liegt, unterschreitet er die Untergrenze der höher angesetzten Zerlegung von c bis h an einigen Stellen. Dadurch erhält die Merkmalsfolge mehr Struktur, die zur stärkeren Abgrenzung von falschen Passagen führt, weshalb sich ähnliche Abschnitte deutlicher absetzen können. Einige falsche Peaks weisen höhere Kosten auf als bei der vorherigen Zerlegung, so z.B. zwei der drei erkennbaren Spitzen im Stück von Bach. Auffälligerweise sind die Kosten des ersten Peaks im Stück von Bach jedoch etwas gesunken. Abbildung 4.12(b) zeigt einen Ausschnitt der Merkmalsfolge,



(a) Oktavzerlegung

(b) Merkmale, Ausschnitt „Toccata“ aus $\mathcal{D}^{Sho}$

(c) Merkmale der Anfrage $\mathcal{Q}^{Sho}$ (Zerlegung c-h)

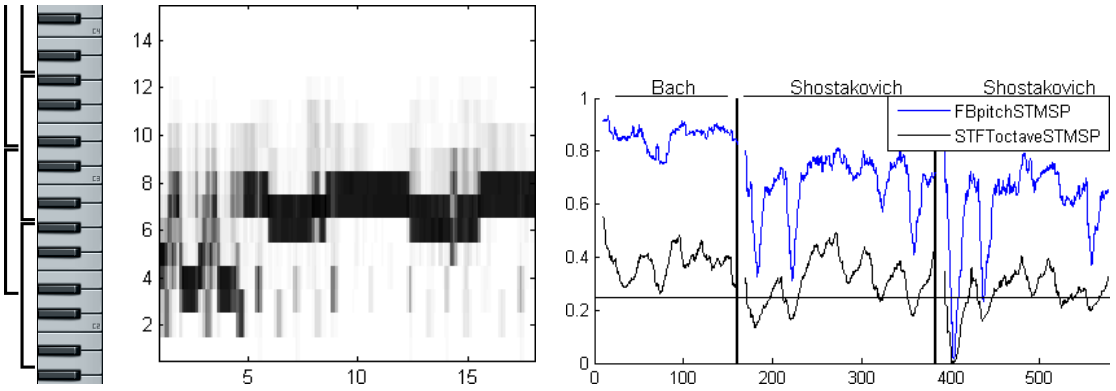


(d) Merkmale der Anfrage $\mathcal{Q}^{Sho}$ (Zerlegung a-$g\#$)

Abbildung 4.12. Veranschaulichung des Binnings und der Merkmalsfolgen von Datenbank und Anfrage bei Oktavzerlegung mit Oktaven von c bis h. Zum Vergleich ist in 4.12(d) noch einmal die Merkmalsfolge der Anfrage bei Zerlegung in Oktaven von a bis $g\#$ dargestellt.

der etwa den Bereich des falschen Peaks abdeckt. Man erkennt im eingekreisten Bereich einen Verlauf von hohen Einträgen zwischen Komponente 3 und 4, der dem Verlauf in der Anfrage ähnlich ist. Die Länge dieses Abschnitts der Datenbank stimmt zwar nicht mit der Länge der Anfrage überein, dies wird jedoch teilweise durch den DTW-Algorithmus kompensiert, sodass relativ geringe Kosten und damit der Peak zustandekommen. Die hier vorliegende Ähnlichkeit der Merkmalsvektoren ist rein zufällig. Je stärker ein Merkmal vergröbert ist und je weniger Struktur des Ausgangsstückes erhalten bleibt, desto größer wird die Gefahr, dass unterschiedliche Passagen zufällig zu ähnlichen Merkmalen führen. In diesem Fall sind jedoch die Kosten der tatsächlich ähnlichen Passagen noch geringer und alle acht Treffer sind korrekt.

Das vorhergehende Beispiel hat gezeigt, dass die Wahl der Oktavgrenzen bei der Oktavzerlegung entscheidenden Einfluss auf Matchingkurve und Treffer hat. Ein fester Melodieverlauf kann vollständig innerhalb einer bestimmten Oktave liegen, jedoch bei anderer



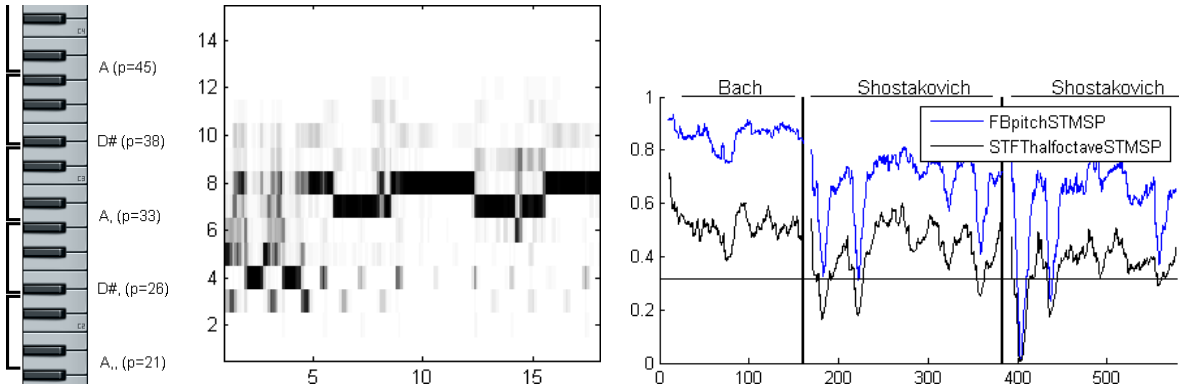
(a) Binning (b) Merkmale der Anfrage $\mathcal{Q}^{Sho}$ (c) Matchingkurve ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$)

Abbildung 4.13. Oktavzerlegung mit überlappenden Oktaven. Die Intervalle sind a bis $g\#$ und $d\#$ bis d.

Wahl der Oktavgrenzen diese überschreiten und ein charakteristisches Muster erzeugen, das gänzlich verloren geht, wenn der gesamte Verlauf zu einem Bin zusammengefasst wird. Da die am besten geeigneten Grenzen bei jedem Stück anders sein können, ist es nicht sinnvoll, sich einfach auf bestimmte Grenzen festzulegen. Um das Problem abzuschwächen, dabei jedoch weiterhin mit einer Zerlegung in ganze Oktaven zu arbeiten, wird nun mit einer Überlappung der Frequenzfenster gearbeitet. Dazu werden wieder die Randfrequenzen des ersten Fensters auf $p = 21$ und $p = 32$ gesetzt, sodass die erste Oktave von a bis $g\#$ reicht. Die Überlappung wird automatisch so bestimmt, dass jedes Fenster gerade das übernächste berührt, siehe Formel (4.4). „win_increase“ wird auf $\frac{1}{2}$ gesetzt, damit ergibt sich ein Vergrößerungsfaktor von $2^{\frac{1}{2}} = \sqrt{2}$. Die Fensterbreite verdoppelt sich also alle zwei Fenster. Da jedes Fenster das übernächste berührt, ergibt sich auf diese Weise eine Oktavzerlegung mit Oktaven von a bis $g\#$, die mit anderen Fenstern überlagert ist, welche jeweils die obere Hälfte einer Oktave und die untere Hälfte der nächsthöheren Oktave abdecken, siehe Abbildung 4.13(a).

Die Idee des Verfahrens besteht darin, trotz der großen Fenster einen Teil der Struktur der Stücke beizubehalten, die bei den vorherigen nicht überlappten Oktavzerlegungen zu einem großen Teil in einem Bin verschwunden ist. Durch die Überlappung werden die Bereiche, die nicht weiter differenziert werden, d.h. in einem Bin liegen, verkleinert. Die Bereiche, in denen die gesamte innere Struktur verloren geht, umfassen nur noch sechs MIDI-Pitches statt zwölf.

In Abbildung 4.13(b) ist die Merkmalsfolge der Anfrage dargestellt. Man sieht, dass der Abschnitt ab der fünften Sekunde, der bei der nicht überlappenden Oktavzerlegung von a bis $g\#$ überwiegend ein einzelnes Band gebildet hat (Abbildung 4.12(d)), nun eine Struktur enthält, die sich über drei Komponenten der Merkmalsvektoren erstreckt. Es bleibt also mehr Information erhalten, die beim Identifizieren der ähnlichen Passagen helfen könnte, obwohl bei der Zerlegung für das unterste Fenster die Grenzen a bis $g\#$ gewählt worden



(a) Binning (b) Merkmale der Anfrage (c) Matchingkurve ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$)

Abbildung 4.14. Halboktavzerlegung. Dargestellt sind eine Veranschaulichung des Binnings, die Merkmalsfolge der Anfrage $\mathcal{Q}^{Sho}$ und die Matchingkurve auf der Datenbank $\mathcal{D}^{Sho}$.

sind, die sich bei der reinen Oktavzerlegung für dieses Beispiel als ungünstig herausgestellt haben. Die Matchingkurve in Bild 4.4.2 zeigt, dass die acht ersten Treffer genau mit den acht ähnlichen Passagen übereinstimmen. Das Ergebnis hat sich durch die Überlappung also auch für die ungünstigeren Grenzen verbessert.

Die Überlappung stellt eine Möglichkeit dar, die großen Fenster beizubehalten und trotzdem nicht zuviel Struktur zu zerstören. Stattdessen kann auch mit kleineren Fenstern gearbeitet werden. Dies soll im nächsten Abschnitt untersucht werden.

4.4.3 Halboktavzerlegung und feinere Zerlegungen

Das Spektrum soll nun in Halboktaven zerlegt werden, d.h. ein Bin umfasst nicht mehr zwölf, sondern nur noch sechs MIDI-Pitches. Dazu wird das erste Fenster auf $p = 21$ bis $p = 26$ festgelegt. Es enthält damit die sechs Noten a bis d. „win_increase“ wird auf $\frac{1}{2}$ gesetzt, damit ist der Vergrößerungsfaktor $\sqrt{2}$. Mit diesen Werten und einer Überlappung von Null wird erreicht, dass jeweils zwei aufeinanderfolgende Fenster zusammen eine Oktave abdecken, wobei jedes der beiden Fenster genau sechs MIDI-Pitches enthält, siehe 4.14(a). Die Datenbank ist wie üblich $\mathcal{D}^{Sho}$, Anfrage ist dazu passend $\mathcal{Q}^{Sho}$. Die Merkmalsvektoren der Anfrage sind in Abbildung 4.14(b) visualisiert. Der Bereich, der bei der Oktavzerlegung überwiegend zu einem Band geführt hat, ist wegen der schmaleren Bins etwas differenzierter und bildet eine Struktur, die über zwei Vektorkomponenten verläuft. Weiterhin erkennt man aber auch deutliche Unterschiede zu den Merkmalen der überlappenden Oktavzerlegung. Die Halboktavzerlegung stellt also ein weiteres Merkmal mit anderen Eigenschaften dar. Abbildung 4.14(c) zeigt die Matchingkurve. Es liegen sechs korrekte Treffer vor, das ist einer weniger als bei der Zerlegung in Oktaven von a bis $g\#$, vergleiche Abbildung 4.11(a). An diesem einen Beispiel kann man natürlich nicht sehen, welche der vorgestellten groben Zerlegungen die besten Resultate liefert. Jedoch fällt auf, dass die Oktavzerlegung mit verschiedenen Grenzen, mit und ohne Überlappung und auch die Halboktavzerlegung

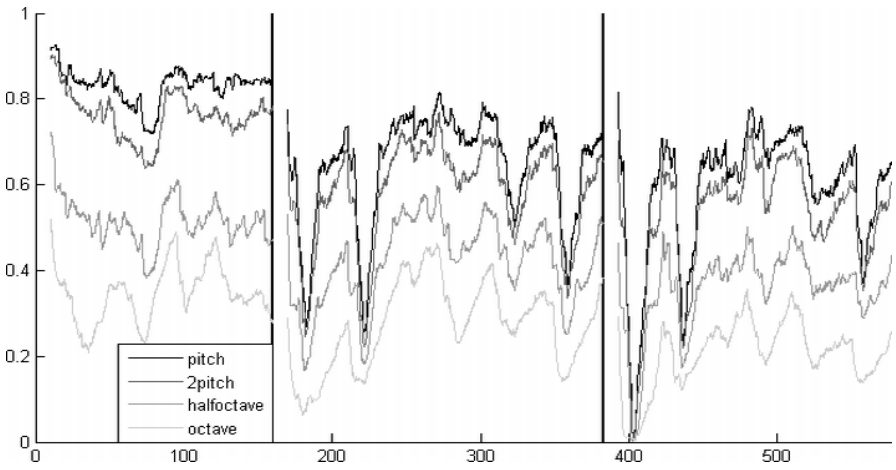


Abbildung 4.15. Matchingkurven ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$) von vier schrittweise verfeinerten Binnings. Dargestellt sind die Kurven von Oktav- und Halboktavzerlegung, von einer Zerlegung, die jeweils zwei Pitches zusammenfasst (2pitch) und von einer Pitchzerlegung. Je feiner die Zerlegung ist, desto höher verläuft die Kurve im Schnitt, siehe Text.

allesamt unterschiedliche Eigenschaften haben, sodass es sich lohnt, diese Merkmale auf größeren Datenbanken zu testen. In Abschnitt 4.4.4 wird ein genauerer Vergleich zwischen Oktavzerlegung mit Überlappung und Halboktavzerlegung durchgeführt.

Nun soll untersucht werden, wie sich die Matchingkurve ändert, wenn die Größe der Bins schrittweise reduziert wird. Datenbank und Anfrage sind weiterhin $\mathcal{D}^{Sho}$ bzw. $\mathcal{Q}^{Sho}$. Es werden verschiedene Binnings generiert, wobei jedoch immer das unterste Fenster bei $p = 21$ beginnt. Abbildung 4.15 zeigt Matchingkurven von verschiedenen Merkmalen. Eingezeichnet sind die Kurven der bekannten Oktav- und Halboktavzerlegung. Zusätzlich sind eine Zerlegung mit doppelter Pitchbreite und eine Pitchzerlegung erstellt worden. Die Pitchzerlegung geschieht mit einem „win_increase“-Wert von $\frac{1}{12}$, bei der Zerlegung in doppelte Pitches wird $\frac{1}{6}$ gewählt. Je feiner die Zerlegung im Frequenzbereich aufgelöst ist, desto höher ist der durchschnittliche Verlauf der zugehörigen Matchingkurve. Dies ist sehr gut im Stück von Bach zu erkennen, wird aber auch in den beiden anderen Stücken deutlich. Die Begründung hierfür ist wieder, dass Merkmale, die viele Details enthalten, höhere Kosten verursachen als Merkmale, bei denen viel Information in einzelnen Bins verschwindet. Bei kleineren Fenstern bleibt mehr Struktur erhalten, die zu Unterschieden und höheren Kosten führt. Aus dem gleichen Grund geht die Zahl und Intensität der falschen Peaks zurück, wenn die Fenster kleiner werden. Dies ist ebenfalls besonders gut im Stück von Bach zu erkennen, aber auch in den beiden anderen Stücken. Bei korrekten Peaks können durch das Verfeinern zwei verschiedene Phänomene auftreten, welche sich beide gut an den Abschnitten A_3 der Stücke von Shostakovich beobachten lassen. Der Abschnitt ist von einer Posaune und Blechbläsern gespielt und unterscheidet sich deshalb deutlich in Klangfarbe und Artikulation von der Anfrage aus Teil A_1, welcher von Streichern gespielt wird. Im ersten Stück von Shostakovich setzt sich der Peak bei zunehmender Verfeinerung stärker von der Umgebung ab. Die Kosten der falschen Abschnitte steigen wegen der höhe-

ren Auflösung an, während in diesem Fall die Kosten für den ähnlichen Abschnitt weniger stark ansteigen. Da A_3 sich jedoch auch von der Anfrage unterscheidet, hätten die Kosten bei höherer Auflösung für diesen Teil auch stärker steigen können. Dies ist der Fall im zweiten Stück von Shostakovich in der Datenbank. Hier grenzt sich der Peak in der Kurve der Pitchzerlegung schwächer ab als in der Kurve der Oktavzerlegung.

Es lässt sich also keine direkte Aussage treffen, ob feinere oder gröbere Zerlegungen bessere Ergebnisse liefern, da bei feineren Auflösungen zwar nicht gewünschte Passagen schneller zu hohen Kosten führen, jedoch auch ähnliche Abschnitte, die gewisse Unterschiede zur Anfrage aufweisen, nicht mehr herausragen. Im nächsten Abschnitt werden drei der Merkmale auf einer größeren Datenbank getestet.

4.4.4 Vergleich verschiedener Binnings

Bisher sind verschiedene Eigenschaften von Merkmalen und auftretende Phänomene an einzelnen Beispielen erläutert worden. Um die Güte verschiedener Merkmale in Bezug auf das Audiomatching besser beurteilen zu können, werden nun Experimente auf einer größeren Musikdatenbank durchgeführt. Die Datenbank enthält etwa 20 Stunden klassische Musik in Form von Orchester- und Klavierstücken von 14 Komponisten, unter anderem Bach, Beethoven, Mozart und Vivaldi, siehe Anhang B. Darin sind auch alle in bisherigen Beispielen verwendeten Stücke enthalten. Für jedes zu testende Merkmal werden drei Anfragen durchgeführt, dabei handelt es sich um Abschnitte, die aus den bisherigen Beispielen bekannt sind, nämlich $\mathcal{Q}^{BeSin}$, $\mathcal{Q}^{Sho}$ und die Sekunden 29 bis 46 aus der Ruebsam Version von Bachs Toccata, diese Anfrage wird mit $\mathcal{Q}^{BachT}$ bezeichnet.

Da der Speicherbedarf des DTW-Algorithmus hoch ist, wird die Datenbank in vier etwa gleichgroße Blöcke unterteilt. Auf jedem Block wird getrennt ein Audiomatching mit den Anfragen durchgeführt und danach die Matchingkurven, die aus den einzelnen Blöcken resultieren, konkateniert. Auf diese Weise können auch große Datenbanken verarbeitet werden und das Verfahren könnte parallelisiert werden.

Zur Auswertung der Matchingergebnisse kommen *Precision-Recall-Diagramme* zum Einsatz, eine umfangreichere Einführung dazu findet man in [3]. Diese Diagramme eignen sich besonders gut, wenn man als Ergebnis zu einer Anfrage eine sortierte Trefferliste erhält, wie es bei der Minimabestimmung der Matchingkurve der Fall ist. Wie in Abschnitt 2.3 beschrieben werden die Minima der Matchingkurve nacheinander bestimmt und es resultiert eine nach Kosten sortierte Liste. Liegt zu einer Anfrage q nach dem Audiomatching mit einem bestimmten Merkmal eine Trefferliste T_q vor und ist die Menge der tatsächlich gewünschten (relevanten) Treffer R_q zur Anfrage bekannt, so können *Precision* p_q und *Recall* r_q berechnet werden:

$$p_q = \frac{|T_q \cap R_q|}{|T_q|} \tag{4.6}$$

$$r_q = \frac{|T_q \cap R_q|}{|R_q|} \tag{4.7}$$

Dabei gibt die Precision das Verhältnis von der Anzahl der relevanten Treffer zur Gesamtzahl der erhaltenen Treffer an, während der Recall das Verhältnis von der Anzahl der relevanten Treffer zur Zahl der gewünschten Treffer beschreibt. Die Precision stellt also die Güte der Trefferliste in Bezug auf die Anzahl fehlerhafter Einträge dar und der Recall die Güte in Bezug auf Vollständigkeit der Trefferliste. Betrachtet man nur die ersten $i \in [1 : |T_q|]$ Einträge der Trefferliste, so definiert man die Precision- und Recallwerte entsprechend für die eingeschränkte Treffermenge $T_{q,i}$:

$$p_{q,i} = \frac{|T_{q,i} \cap R_q|}{|T_{q,i}|} = \frac{|T_{q,i} \cap R_q|}{i} \tag{4.8}$$

$$r_{q,i} = \frac{|T_{q,i} \cap R_q|}{|R_q|} \tag{4.9}$$

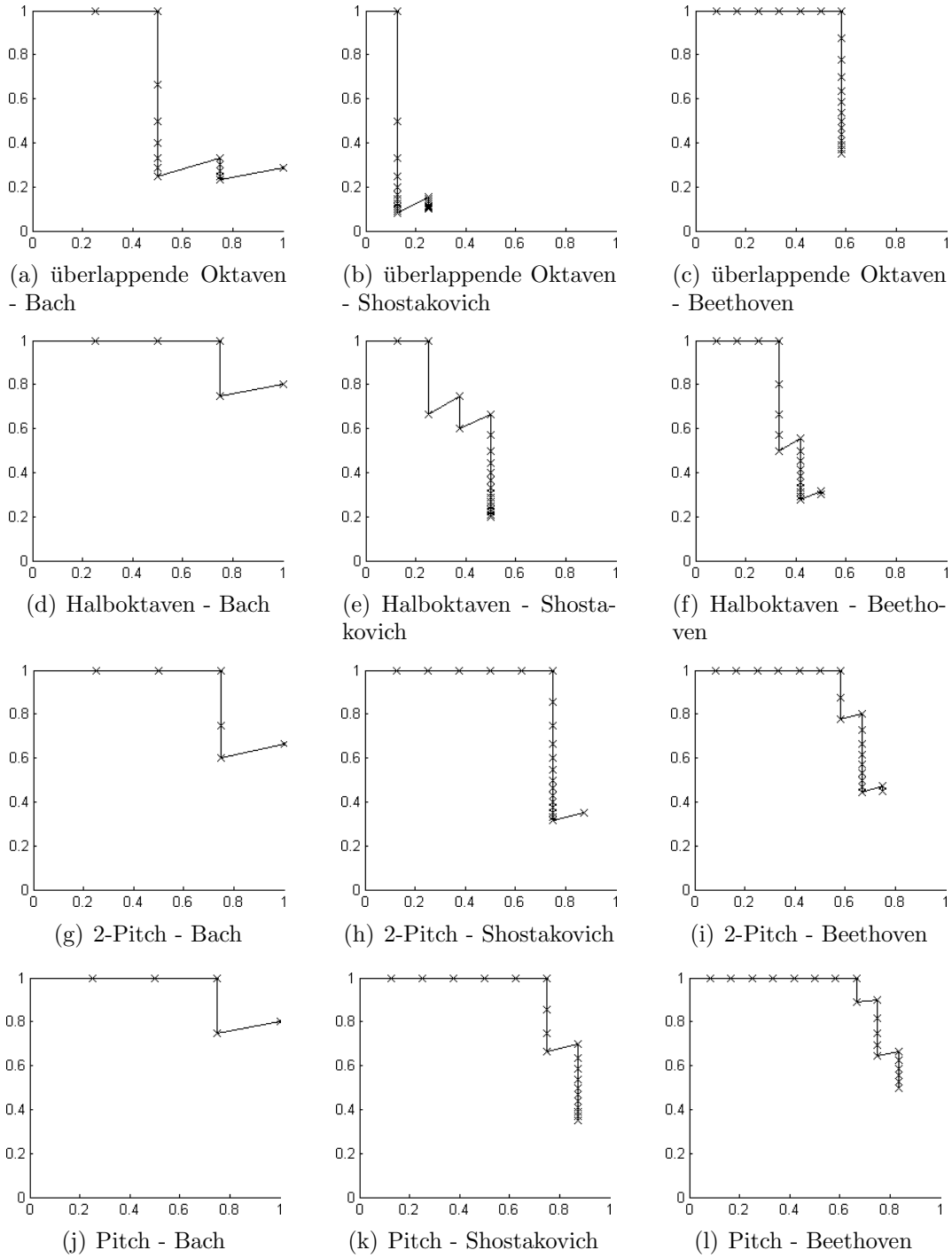
Bei schrittweiser Vergrößerung der betrachteten Trefferliste ergeben sich Folgen von Precision- und Recallwerten. Trägt man diese Folgen gegeneinander ab, erhält man das Precision-Recall-Diagramm:

$$\{(r_{q,i}, p_{q,i}) \mid i \in [1 : |T_q|]\} \tag{4.10}$$

Die Diagramme erlauben es, den Verlauf der Precision bei steigendem Recall zu untersuchen. Steigenden Recall erhält man, wenn die betrachtete Liste verlängert wird und damit mehr Funde berücksichtigt werden. Im Allgemeinen enthält eine längere Trefferliste jedoch auch mehr falsche Einträge, sodass die Precision in den meisten Fällen bei steigendem Recall sinkt.

Untersucht werden nun die überlappende Oktavzerlegung, die Halboktavzerlegung, die Zerlegung in Bänder mit der Breite von zwei Pitches und die Pitchzerlegung. Zu jedem der vier Merkmale wird für jede der drei genannten Anfragen ein Diagramm erstellt. Abbildung 4.4.4 zeigt die zwölf resultierenden Precision-Recall Diagramme. Die Trefferliste zu den 2pitch-Merkmalen ist in Anhang C beispielhaft für die Anfrage aus dem Stück von Shostakovich zu finden. Alle Trefferlisten sind auf 20 Einträge begrenzt. Falls sich bei einer Anfrage die gewünschten Treffer nicht alle unter den ersten 20 der Liste befinden, wird der Recall von 1 nicht erreicht. In diesem Fall bricht die Kurve vorher ab. Je früher eine Kurve abbricht, desto geringer ist also der Anteil der gewünschten Treffer unter den ersten 20 der Liste. Umgekehrt weist eine bis zum Recall von 1 reichende Kurve darauf hin, dass innerhalb der ersten 20 Treffer alle gewünschten Passagen vorkommen. Zu beachten ist, dass die Stücke, aus denen die Anfragen stammen, in der Datenbank enthalten sind und damit der erste Treffer immer korrekt ist.

Für die Zahl der erwarteten Treffer zu den Anfragen gilt $|R_{\mathcal{Q}^{BachT}}| = 4$, $|R_{\mathcal{Q}^{Sho}}| = 8$ und $|R_{\mathcal{Q}^{BeSin}}| = 12$. Bei allen vier Merkmalen sind alle gewünschten Treffer zur Anfrage $\mathcal{Q}^{BachT}$ unter den ersten 20 Einträgen der Trefferliste zu finden. Die Kurven zur Halboktavzerlegung, zur 2pitch-Zerlegung und zur Pitchzerlegung weisen bis zum Recallwert von 0.75 eine Precision von eins auf. Dies bedeutet, dass $\frac{3}{4}$ der erwarteten Treffer, bei 4 erwarteten also 3, ohne Unterbrechung durch falsche Funde beginnend mit Position eins in der Trefferliste auftreten, d.h. Treffer eins bis drei sind korrekt. Das Diagramm zur überlappenden Oktavzerlegung nimmt nur bis 0.5 den Wert eins an, hier ist also der dritte Treffer nicht korrekt.



(a) überlappende Oktaven - Bach
(b) überlappende Oktaven - Shostakovich
(c) überlappende Oktaven - Beethoven
(d) Halboktaven - Bach
(e) Halboktaven - Shostakovich
(f) Halboktaven - Beethoven
(g) 2-Pitch - Bach
(h) 2-Pitch - Shostakovich
(i) 2-Pitch - Beethoven
(j) Pitch - Bach
(k) Pitch - Shostakovich
(l) Pitch - Beethoven

Abbildung 4.16. Precision Recall Diagramme für jeweils drei Anfragen bei vier verwendeten Merkmalen. Die X-Achse gibt den Recall an, die Y-Achse die Precision. Die Zahl der tatsächlich relevanten Treffer zur Anfrage beträgt bei Bach 4, bei Shostakovich 8 und bei Beethoven 12. Überlappende Oktavzerlegung und Halboktavzerlegung zeigen wechselhafte Ergebnisse, während sich die feineren 2pitch- und pitch-Merkmale absetzen.

Auch die folgenden Werte dieses Diagramms sind geringer als bei den Diagrammen zu den drei anderen Merkmalen, sodass die überlappende Oktavzerlegung bei dieser Anfrage eindeutig schlechter abgeschnitten hat. Die zu Halboktav- und 2pitch-Zerlegung gehörenden Diagramme fallen an gleicher Position von der Eins ab, jedoch sinkt das von der 2pitch-Zerlegung auf einen niedrigeren Wert, d.h. die Zahl der falschen Funde bis zum nächsten korrekten Treffer ist höher als bei der Halboktavzerlegung. Das Resultat der Halboktavzerlegung ist bei der Anfrage $\mathcal{Q}^{BachT}$ damit etwas besser als das der 2pitch-Zerlegung. Die Pitchzerlegung führt exakt zum gleichen Diagramm wie die Halboktavzerlegung.

Bei keinem der Merkmale befinden sich alle acht ähnlichen Abschnitte zur Anfrage $\mathcal{Q}^{Sho}$ unter den ersten 20 Treffern. An der Länge der Kurve (Zahl der korrekten Treffer unter den ersten 20), der Länge des Abschnitts an dem die Kurve eins ist (Zahl der korrekten Treffer ohne Unterbrechung beginnend von Platz eins) und den Werten des weiteren Kurvenverlaufs sieht man sehr schnell, dass die überlappende Oktavzerlegung ein schlechtes Ergebnis liefert, die Halboktavzerlegung besser abschneidet und die 2pitch-Zerlegung noch einmal deutlich bessere Resultate abgibt. Die Kurve der Pitchzerlegung verläuft zunächst ähnlich wie die der 2pitch-Zerlegung, fällt dann jedoch weniger stark ab. Damit liefert die Pitchzerlegung für diese Anfrage das beste Ergebnis.

Auch zur Anfrage $\mathcal{Q}^{BeSin}$ werden bei keinem der Merkmale alle gewünschten Abschnitte unter den ersten 20 gelistet. Vier der zwölf erwarteten Treffer befinden sich jedoch auch in der bereits als kritisch erwähnten Klaviertranskription von Beethovens Fünfter Sinfonie. Diesmal ergibt die überlappende Oktavzerlegung gut erkennbar ein besseres Ergebnis als die Halboktavzerlegung. Denn bei der Halboktavzerlegung ist die Zahl der korrekten Treffer niedriger und zwischen den richtigen befinden sich einige falsche Funde. Die 2pitch-Zerlegung erzielt, wenn auch nur geringfügig, bessere Resultate. Die Pitchzerlegung zeigt aber auch hier ein noch besseres Ergebnis.

Insgesamt wechseln sich überlappende Oktavzerlegung und Halboktavzerlegung in der Güte der Ergebnisse ab. Die 2pitch-Zerlegung zeigt, bis auf eine geringe Ausnahme bei der Anfrage $\mathcal{Q}^{BachT}$, deutlich bessere Ergebnisse. Letztendlich weist jedoch das feinste der untersuchten Binnings, nämlich die Pitchzerlegung, bei allen drei Anfragen das beste Ergebnis auf.

Kapitel 5

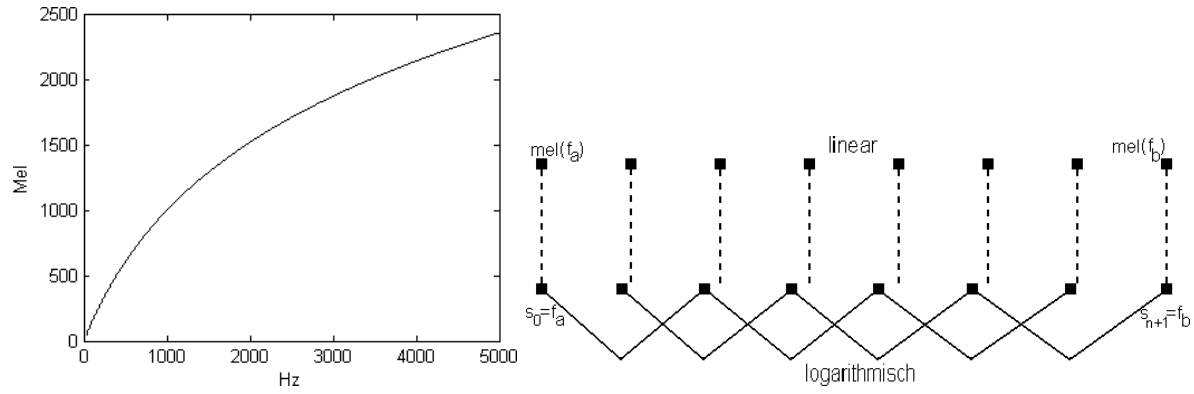
MFCC-Merkmale

Alle bisher vorgestellten Merkmale basieren auf einer Zerlegung des Spektrums, die an der MIDI-Notenstruktur, also an der westlichen, temperierten Stimmung orientiert ist. Ein anderer Ansatz ist, die menschliche Wahrnehmung der Lautstärke und Tonhöhe zu betrachten und das Audiosignal entsprechend psychoakustischer Erkenntnisse zu Merkmalen zu verarbeiten. Diese Strategie kommt bei den *MFCC-Merkmalen (Mel Frequency Cepstral Coefficients)* zum Einsatz. Die Merkmale stammen aus der Sprachsignalverarbeitung und sind dort stark verbreitet, werden aber zunehmend für die Verarbeitung von Musikdaten genutzt, siehe [7].

In Abschnitt 5.1 werden die einzelnen Schritte der Berechnung der Merkmale erläutert, dabei wird auch schon auf die Mel-Skala als Maß für die subjektive Wahrnehmung der Tonhöhe eingegangen. In Abschnitt 5.2 werden verschiedene Ergebnisse zu den MFCC-Merkmalen vorgestellt und der Sinn der abschließenden DCT bei der Merkmalsberechnung geklärt. Darauf folgt in Abschnitt 5.3 eine erste Untersuchung der Merkmale anhand von Audiomatchingszenarien. Dabei wird sich herausstellen, dass die Merkmale sehr stark auf die Lautstärke der Audiosignale reagieren. Um dies weiter zu untersuchen, werden in Abschnitt 5.4 *Amplitudenmerkmale* eingeführt, die ausschließlich von der Lautstärke abhängen. Abschnitt 5.5 enthält einen Vergleich der Matchingkurven von MFCC- und Amplitudenmerkmalen und behandelt modifizierte MFCC-Merkmale, die nicht mehr vom Lautstärkeverlauf abhängig sind. In Abschnitt 5.6 werden Konzepte der MFCC-Merkmale – nämlich das Logarithmieren und die DCT – auch auf Pitchmerkmale angewendet. Schließlich werden im letzten Abschnitt 5.7 MFCC-Merkmale mit CENS-Merkmalen und der 2pitch-Zerlegung auf einer größeren Testdatenbank verglichen.

5.1 Berechnung der MFCC-Merkmale

Zur Berechnung der MFCC-Merkmale wird die „MA Toolbox für MATLAB“ verwendet, die auf [12] frei verfügbar ist. Damit die Namen von Parametern und die grundsätzliche Schnittstelle konsistent zu den anderen Funktionen zur Merkmalsextraktion bleiben, wird eine Wrapper-Funktion benutzt, die die entsprechende Funktion der Toolbox aufruft, im



(a) Melskala (b) Binning entsprechend der Melskala

Abbildung 5.1. Darstellung der Melskala in Abhängigkeit von der Frequenz und Visualisierung des Mel-Binnings: Zunächst wird der Frequenz-Wertebereich $[f_a, f_b]$ in Mel umgerechnet und der Mel-Bereich linear eingeteilt. Die dabei entstehenden $n + 2$ äquidistanten Stützstellen werden wieder in Hz umgerechnet und ergeben so die Grenzen des logarithmischen Binnings, welches mit Dreiecksfiltern realisiert wird.

Audiomatchingprogramm aber genau wie alle bisherigen Merkmalsextraktoren eingebunden werden kann. Im Folgenden werden die einzelnen Berechnungsschritte zur Extraktion der Merkmale und die Parameter der Wrapper-Funktion erklärt.

Die MFCC-Merkmale sind STFT-basiert, d.h. den ersten Schritt stellt die Berechnung des Spektrogramms des Audiosignals dar, wie in Kapitel 4. Fensterbreite und Überlappung werden wie üblich über die Parameter „win_len“ bzw. „win_ov“ eingestellt.

Die Koeffizienten der einzelnen Spektralvektoren werden nun zu Bins zusammengefasst. Die Anordnung dieser Bins ist an der sogenannten *Mel-Skala* orientiert, die durch die Wahrnehmung der Tonhöhe durch den Menschen motiviert ist. Für die Mel-Skala gibt es verschiedene Definitionen, die aus Experimenten zur subjektiven Wahrnehmung der Tonhöhe hervorgegangen sind. Hier wird die Definition von Stevens & Volkmann, siehe [16], verwendet. Die Mel-Werte ergeben sich nach dieser Definition aus der Frequenz f in Hertz mittels

$$\mathrm{Mel}(f) := 1127.01048 \cdot \ln\left(\frac{f}{700} + 1\right) \tag{5.1}$$

Die Formel ist empirisch ermittelt worden, sodass eine Frequenz, die subjektiv als doppelt so hoch wahrgenommen wird als eine andere, auch den doppelten Mel-Wert erhält. Der Vorfaktor ist eine Normierung, damit eine Frequenz von 1000 Hz einem Wert von 1000 Mel entspricht, wobei dieser Bezugspunkt willkürlich gewählt ist. Abbildung 5.1(a) zeigt die Melskala in Abhängigkeit von der Frequenz. Betrachtet man die Y-Achse als sogenannte *Tonheit*, d.h. die subjektiv wahrgenommene Tonhöhe, so erkennt man, dass bei linear steigender Frequenz die Tonheit nur logarithmisch zunimmt. Es besteht also ein Zusammenhang zu den MIDI-Noten, denn auch die Pitch-Nummer nimmt bei linear steigender Frequenz logarithmisch zu. So ist also auch die Tonleiterstruktur grundsätzlich an der menschlichen Wahrnehmung der Tonhöhe ausgerichtet.

Neben der Berechnung der Mel-Werte aus Frequenzen wird auch die Umrechnung von Mel in Hertz bei der Erzeugung der Merkmale benötigt:

$$f = \text{Mel}^{-1}(m) = 700 \cdot \left(\exp\left(\frac{m}{1127.01048}\right) - 1\right) \tag{5.2}$$

Der Parameter „mel_filt_bank“ enthält drei Werte, die die Struktur der Filterbank[1] festlegen, nämlich die kleinste berücksichtigte Frequenz f_a in Hertz, die größte Frequenz f_b und die Anzahl der Filter n. Die Zerlegung des Spektrums geschieht mit Dreiecksfiltern. Die Anfangs- und Endpositionen und die Zentren der Filter sollen entsprechend der Mel-Skala im Spektrum verteilt sein. Dazu werden $n + 2$ Stützstellen äquidistant auf das Intervall $[\text{Mel}(f_a), \text{Mel}(f_b)]$ verteilt, damit hat man den Mel-Bereich linear unterteilt, siehe oberen Teil von Abbildung 5.1(b). Dann werden die einzelnen Stützstellen mit der invertierten Mel-Formel (5.2) in Frequenz in Hertz umgerechnet. Die Stützstellen $(s_i)_{i\in[0:n+1]}$ der Filter in Hertz ergeben sich also aus

$$s_i := \text{Mel}^{-1}\left(\text{Mel}(f_a) + \frac{\text{Mel}(f_b) - \text{Mel}(f_a)}{n+1} \cdot i\right) \tag{5.3}$$

Auf jedem s_i mit $i \in [1 : n]$ wird nun ein Zentrum eines Dreiecksfilters platziert, wobei jedes Filter von s_{i-1} bis s_{i+1} reicht, siehe Abbildung 5.1(b). Man erhält auf diese Weise n überlappende Filter, die an der Mel-Skala orientiert sind, sich also entsprechend der subjektiven Wahrnehmung der Tonhöhe auf den angegebenen Frequenzbereich verteilen.

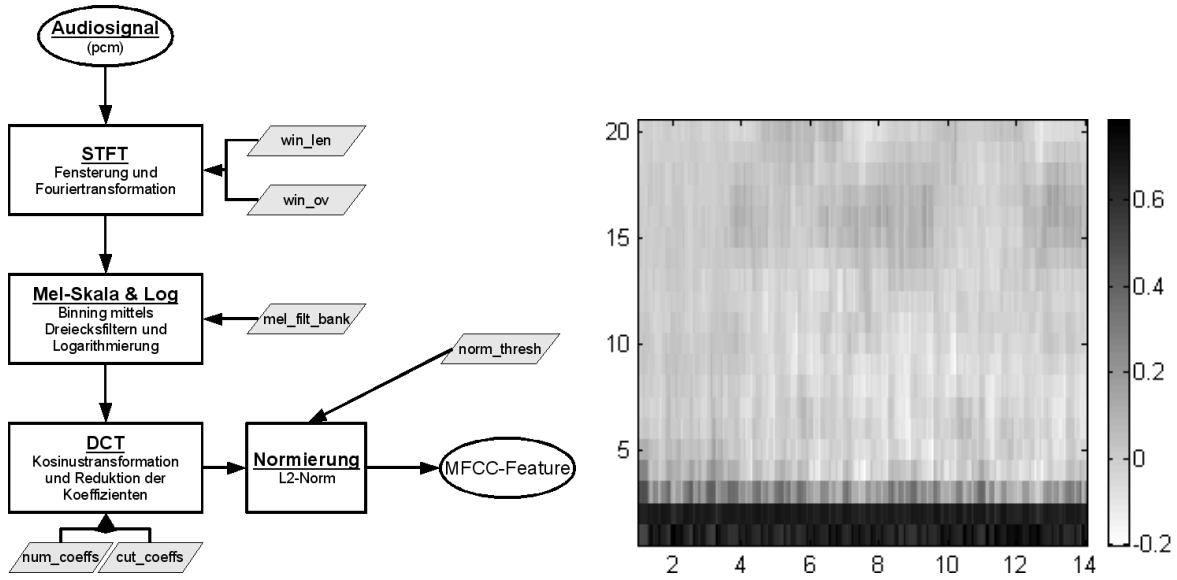
Analog zum Vorgehen in Kapitel 4 werden die passenden DFT-Koeffizienten zu den Frequenzgrenzen der Filter bestimmt und für jedes Filter die Quadrate der zugehörigen Koeffizienten aus den einzelnen Spektralvektoren gewichtet summiert. Es resultieren Merkmalsvektoren mit n Komponenten. Durch die Anordnung der Filter wird die menschliche Wahrnehmung der Tonhöhe berücksichtigt. Auch die Wahrnehmung der Lautstärke ist nicht linear, sondern logarithmisch. Um auch diese Eigenschaft des Gehörs einzubeziehen, wird von allen Einträgen der Merkmalsvektoren der Logarithmus zur Basis zehn berechnet.

Abschließend wird eine *Diskrete Kosinus Transformation (DCT)* durchgeführt. Es existieren vier Varianten der DCT, siehe [13]. Hier wird die *DCT-II* verwendet, diese wird ähnlich der DFT mittels Matrixmultiplikation realisiert. Die DCT-Matrix der Größe L ist definiert als

$$DCT_L := \left(\frac{2}{L}\right)^{\frac{1}{2}} \cdot \left(\kappa_j \cos\left(\frac{j(k+\frac{1}{2})\pi}{L}\right)\right)_{0\le j,k<L} \tag{5.4}$$

wobei $\kappa_0 = \frac{1}{\sqrt{2}}$ gilt und für alle $j \in [1 : L-1] : \kappa_j = 1$. Jeder der Merkmalsvektoren wird mit der Matrix multipliziert und man erhält neue Vektoren gleicher Länge. Für die endgültigen Merkmalsvektoren wird nur die über den Parameter „num_coeffs“ gewählte Zahl von Koeffizienten übernommen, die höheren Koeffizienten werden abgeschnitten. In Abschnitt

[1] Wie in der Literatur üblich wird hier im Zusammenhang mit MFCC-Merkmalen von Filtern gesprochen. Die Filter stellen Fenster dar, innerhalb derer die Koeffizienten der Spektralvektoren gewichtet summiert werden und so ein Binning ergeben. Der Begriff Filter entspricht hier also dem des Frequenzfensters aus Abschnitt 4.4.



(a) Ablauf der MFCC-Extraktion (b) normierte MFCC Merkmalsvektoren

Abbildung 5.2. Schematische Darstellung der MFCC-Merkmalsextraktion und Visualisierung der normierten MFCC-Merkmalsvektoren der Anfrage $\mathcal{Q}^{Sho}$.

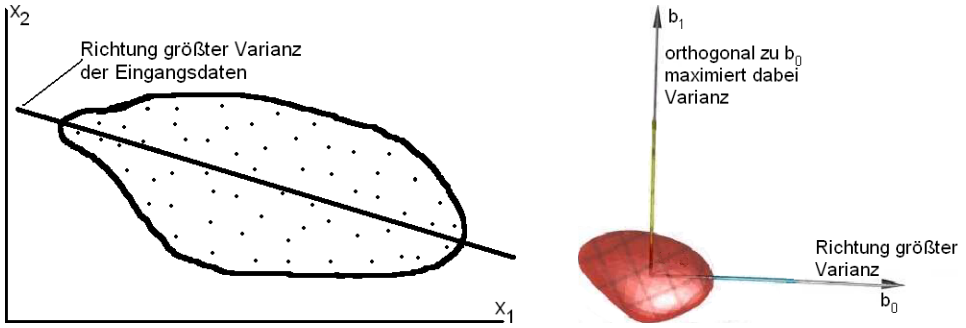
5.5 werden auch MFCC-Merkmale betrachtet, bei denen die niedrigen Koeffizienten entfernt worden sind. Der Parameter „cut_coeffs“ legt die Zahl der Koeffizienten fest, die vom kleinsten Index beginnend abgeschnitten werden.

Vor dem Einsatz zum Audiomatching werden die Merkmalsvektoren L_2-normiert. Zu beachten ist, dass die original MFCC-Merkmale nicht normiert werden, das Normieren dient hier zur Anwendung des immer verwendeten Kostenmaßes per Skalarprodukt. Der gesamte Ablauf der Merkmalsextraktion ist schematisch in Abbildung 5.2(a) dargestellt.

5.2 Allgemeines zu MFCC-Merkmalen

Die DCT am Ende der Merkmalsberechnung dient zur *statistischen Dekorrelation* der Komponenten der Merkmalsvektoren und soll eine Reduktion der Koeffizienten ohne entscheidenden Informationsverlust ermöglichen.

Zunächst wird der Vorgang zum Erreichen der statistischen Dekorrelation kurz erläutert. Am Anfang liegt eine Menge von Vektoren mit fester Anzahl von Komponenten vor, z.B. die einzelnen Vektoren einer Merkmalsfolge. Das Ziel ist, eine orthonormale Basis zu finden, bezüglich derer die Eingangsvektoren eindeutig dargestellt werden können, und die Eingangsvektoren dann als Linearkombination dieser Basisvektoren darzustellen. Nach der Transformation werden die Eingabevektoren also durch die Koeffizienten der Linearkombination festgelegt. Es soll nun nicht eine beliebige orthonormale Basis gefunden werden, sondern die *Varianz* der Eingangsdaten „in Richtung“ eines Basisvektors soll bei kleinen Koeffizienten möglichst groß sein und mit zunehmendem Koeffizient kleiner werden. Für



(a) 2-dimensionale Beispieldaten (b) 3-dimensionale Beispieldaten

Abbildung 5.3. 5.3(a): Grafisch veranschaulichte Menge 2-dimensionaler Eingabedaten. Die Richtung des ersten Basisvektors (Richtung größter Varianz) ist eingezeichnet. 5.3(b): Darstellung 3-dimensionaler Eingansdaten und der beiden ersten (orthogonalen) Basisvektoren.

einen Vektor $x = (x_1, x_2, \ldots, x_n)^T$ sei $\bar{x} := \frac{1}{n}\sum_{i=1}^{n} x_i$ der Mittelwert oder *Erwartungswert* des Vektors, dann stellt die Varianz

$$\mathrm{Var}(x) := \frac{1}{n}(x - \bar{x})^T(x - \bar{x}) \tag{5.5}$$

ein Maß für die Abweichung der Vektoreinträge vom Mittelwert dar. Der erste Basisvektor b_0 soll nun entsprechend der größten Varianz der Eingangsdaten ausgerichtet sein, siehe Abbildung 5.3(a). Falls p Eingabevektoren mit jeweils n Komponenten vorliegen, so sei X die $n \times p$ Matrix, die in ihren Spalten die Eingabevektoren enthält. b_0 soll nun so bestimmt werden, dass $\mathrm{Var}(b_0^T X)$ maximal wird, außerdem soll $b_0^T b_0 = 1$ gelten, da eine orthonormale Basis angestrebt wird. Dann wird der nächste Basisvektor b_1 bestimmt, sodass unter den Nebenbedingungen $b_1^T b_1 = 1$ und $b_1^T b_0 = 0$ die Varianz $\mathrm{Var}(b_1^T X)$ maximal wird. b_1 ist also ein zu b_0 orthogonaler Einheitsvektor, der in Richtung möglichst großer Varianz der Eingabedaten zeigt. Abbildung 5.3(b) zeigt dreidimensionale Eingabedaten und die beiden ersten Basisvektoren, die orthogonal zueinander sind und in Richtung größter Varianz zeigen. Auf diese Weise werden bei zunehmender Zahl von Nebenbedingungen weitere Vektoren konstruiert, bis die Basis mit n Vektoren vollständig ist. Dieser Aufbau der Basis kann mit der *Hauptkomponentenanalyse*, siehe [6], oder der weiter unten angesprochenen *Karhunen-Loeve-Transformation* realisiert werden.

Für zwei Vektoren $x = (x_1, x_2, \ldots, x_n)^T$ und $y = (y_1, y_2, \ldots, y_n)^T$ stellt der *Korrelationskoeffizient*

$$\mathrm{Kor}(x, y) := \frac{\frac{1}{n}(x - \bar{x})^T(y - \bar{y})}{\sqrt{\mathrm{Var}(x)\mathrm{Var}(y)}} \tag{5.6}$$

ein Maß für den Zusammenhang der beiden Vektoren dar. Die Werte des Koeffizienten liegen immer in $[-1, 1]$. Ein Wert von 1 bedeutet, dass bei Komponenten, die in x große Werte haben, auch in y große Werte auftreten und umgekehrt für kleine Werte. Bei einem Korrelationskoeffizient von -1 ist ein gegenläufiges Verhalten zu beobachten. Wenn

$\mathrm{Kor}(x, y) = 0$ gilt, so verhalten sich x und y unabhängig, man sagt, x und y sind *unkorreliert*. Die Basisvektoren, die nach dem oben beschriebenen Verfahren erzeugt werden, sind paarweise unkorreliert. Daher stammt der Name Dekorrelation, wenn man eine Menge von Eingabevektoren einer solchen Basistransformation unterzieht. Man spricht auch davon, dass die Koeffizienten dekorreliert werden, da sie sich nach der Transformation auf unkorrelierte Basisvektoren beziehen.

Stellt man alle Eingangsvektoren nacheinander als Linearkombination der Basisvektoren dar, so durchläuft der unterste Koeffizient den größten Wertebereich, weil die Abweichung (Varianz) der Eingangsdaten bezüglich des ersten Basisvektors am größten ist. Bei höheren Koeffizienten wird der Bereich geringer, sodass letztendlich eine Sortierung nach Informationsgehalt vorliegt: Beim Entfernen des untersten Koeffizienten geht die Information über das vergleichsweise große Intervall verloren, in dem sich der Koeffizient befinden kann. Entfernt man den höchsten Koeffizient, so wird in den meisten Fällen[2] weniger Information verloren gehen. In [13] wird gezeigt, dass nach der Transformation der mittlere quadratische Fehler, der beim Entfernen der höchsten Koeffizienten auftritt, minimal ist.

Der Sinn der Dekorrelation ist, die Dimension der Eingabedaten und damit auch die Datenmenge zu verringern, während der dabei auftretende Informationsverlust minimiert wird. Falls der Raum der Eingangsdaten eine niedrigere Dimension hat als die Anzahl der Einträge pro Vektor (z.B. wenn die Vektoren 3 Einträge haben und alle in einer Ebene liegen), dann werden höhere Koeffizienten Null und können eingespart werden, ohne dass überhaupt Information verloren geht. Dies ist bei realen Anwendungen selten der Fall, dennoch können häufig die höheren Koeffizienten entfernt werden, ohne dass der Informationsverlust für die Anwendung störend wirkt. Beispielsweise beruhen Verfahren zur verlustbehafteten Kompression von Audio- und Bilddaten auf diesem Konzept.

Die beschriebene Dekorrelation wird exakt durch die *Karhunen-Loeve-Transformation (KLT)* realisiert, siehe [13]. Allerdings hängen die Basisvektoren der KL-Darstellung von den Eingangsdaten ab und sind vor der Transformation nicht bekannt. Zur Bestimmung der KL-Basis müsste das gesamte Signal herangezogen werden, was bei Echtzeitanwendungen unmöglich ist. Aber auch für offline-Berechnungen, wie z.B. die Merkmalsextraktion aus der Datenbank, ist die Karhunen-Loeve-Transformation wegen der aufwändigen Bestimmung der Basis ungeeignet. Es hat sich jedoch gezeigt, dass die DCT für Sprachsignale eine gute Näherung der KLT darstellt. Die DCT transformiert immer in eine feste Basis, nur abhängig von der Länge, und kann durch eine einfache Matrixmultiplikation realisiert werden, siehe Formel (5.4). Wie in [13] gezeigt sind DCT-Matrizen orthogonal, d.h. die Umkehrung der Transformation kann durch Multiplikation mit der transponierten Matrix realisiert werden. Abbildung 5.4 zeigt beispielhaft die acht Basisvektoren der DCT_8. Ein Signal wird also durch die Transformation als Linearkombination dieser Vektoren dargestellt. Der Koeffizient null bezieht sich dabei auf einen Vektor mit konstanten Einträgen und gibt damit einen Offset an. Bezogen auf die konkrete Anwendung der MFCC-Merkmale bedeu-

[2] Es existieren auch Fälle, bei denen die Varianz aller bestimmten Basisvektoren gleich ist, z.B. wenn alle Daten auf einer Hyperkugel liegen. In realen Anwendungen hat man es jedoch mit abfallendem Informationsgehalt zutun.

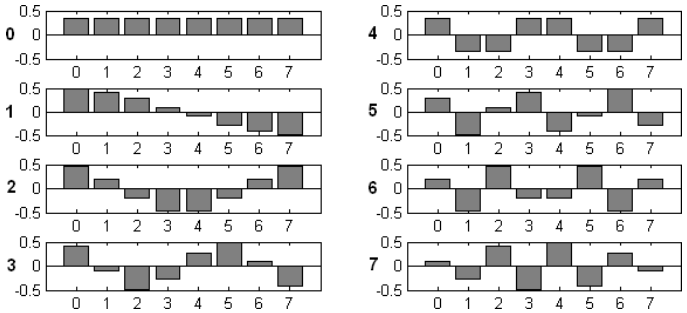


Abbildung 5.4. Die acht Basisvektoren der DCT_8.

tet dies, dass der Koeffizient mit Index null Information über die Gesamtenergie innerhalb eines Zeitfensters enthält. In den Abschnitten 5.3 und 5.5 wird diese Eigenschaft näher betrachtet.

Die gute Approximation der KLT durch die DCT innerhalb der Sprachsignalverarbeitung ist bereits bekannt, siehe [8]. In [7] wird die Übertragbarkeit der MFCC-Merkmale auf Musikdaten überprüft, insbesondere wird dabei auf die DCT als Approximation der KLT für Musik eingegangen. Experimentell ist die Annäherung auch für Musik akzeptiert worden. Allerdings sind die Experimente ausschließlich auf einer Musikdatenbank mit Popmusik getestet worden und unabhängig von konkreten Anwendungen der MFCC-Merkmale. In dieser Arbeit sollen die MFCC-Merkmale mit abschließender DCT speziell im Hinblick auf ihre Eigenschaften im Audiomatching mit klassischer Musik untersucht werden.

Der Sinn, die MFCC-Merkmale einer DCT zu unterziehen, ist einerseits eine Reduktion der Datenmenge. Bei großen Musikdatenbanken müssen riesige Datenmengen gespeichert und verarbeitet werden, sodass für praktische Anwendungen kompakte Merkmale erwünscht sind. Eine andere Motivation ist eine „Glättung“ der einzelnen Merkmalsvektoren. Formel (5.4) und auch die Visualisierung der Basisvektoren 5.4 zeigen, dass die höheren Koeffizienten mit schneller schwingenden Basisvektoren korrespondieren. Entfernt man einen höheren Koeffizient, so verschwinden schnellere Schwankungen innerhalb der Merkmalsvektoren. Diese Eigenschaft wird in der Sprachsignalverarbeitung ausgenutzt, siehe [15]. Aber auch beim Audiomatching könnte die spektrale Glättung, genau wie die zeitliche Glättung der CENS-Merkmale, zu robusteren Merkmalen führen.

In [14] werden die MFCC-Merkmale aus einem anderen Blickwinkel betrachtet. Hier wird versucht mit Hilfe der Merkmale einen akustischen „Wahrnehmungsraum“ zu definieren. Gemeint ist damit, die Klangfarbe über eine gewisse Anzahl von Parametern eindeutig zu beschreiben, analog zum RGB-Modell für Farben. Im RGB-Modell liegen drei Parameter vor, die jeweils eine Komponente zur Gesamtfarbe beisteuern, wobei diese Komponenten eine Art „wahrgenommene Orthogonalität“ aufweisen, da beim Ändern einer der drei Grundfarben die anderen unberührt bleiben und damit eine Unabhängigkeit der Komponenten vorliegt. Nach der DCT sind die Komponenten der MFCC-Merkmale (annähernd) statistisch unabhängig. In [14] wird untersucht, ob die Unabhängigkeit auch für die akustische Wahrnehmung gilt. Dazu werden basierend auf vorherigen Erkenntnissen MFCC-Merkmale mit 13 Koeffizienten untersucht. Es werden bestimmte Koeffizienten auf null und andere auf

positive Werte gesetzt und die Merkmale synthetisiert, also hörbar gemacht. Dann werden nach genauen Vorgaben verschiedene Testpersonen nach der Ähnlichkeit von vorgespielten synthetisierten MFCC-Merkmalen befragt und die Ergebnisse statistisch ausgewertet. Das Experiment wird ausdrücklich als erster Schritt in diese Forschungsrichtung beschrieben, da nur eine sehr kleine Teilmenge von möglichen Merkmalen getestet worden ist und subjektive Angaben von nur zehn Personen vorliegen. Dennoch zeichnet sich ab, dass die statistische Unkorreliertheit der MFCC-Koeffizienten auch in etwa einer akustischen Orthogonalität entspricht. Auch die Anzahl von 13 Koeffizienten scheint geeignet, um einen Wahrnehmungsraum für die Klangfarbe zu definieren.

5.3 Audiomatching mit MFCC-Merkmalen

Nun wird ein Audiomatching mit MFCC-Merkmalen auf der Datenbank $\mathcal{D}^{Sho}$ mit der Anfrage $\mathcal{Q}^{Sho}$ durchgeführt. Wie bei den anderen STFT-Merkmalen üblich wird eine Fensterbreite von 4096 Samples gewählt und eine Überlappung von 1891 also eine Rate von 10 Merkmalsvektoren pro Sekunde. Auch andere Fensterbreiten und zeitliche Auflösungen könnten ausprobiert werden. Die hier verwendeten Werte haben sich in Versuchen als günstig erwiesen und sollen innerhalb dieser Arbeit immer als Ausgangspunkt verwendet werden. Die anderen Parameter werden für das erste Experiment auf den Standardwerten der Toolbox belassen. Damit wird der Frequenzbereich von 20 Hz bis 11050 Hz betrachtet und in 40 Bänder zerlegt. Die Zahl der Koeffizienten wird nach der DCT auf die Hälfte, also auf 20 reduziert. Diese Merkmale werden im Folgenden mit MFCC_{20}^{40} bezeichnet, der hochgestellte Wert gibt also die Zahl der Bins an und der tiefgestellte die Anzahl der benutzten DCT-Koeffizienten. Da die Merkmale grundsätzlich L_2-normiert eingesetzt werden, kann auf ein nachgestelltes L_2 in der Bezeichnung verzichtet werden.

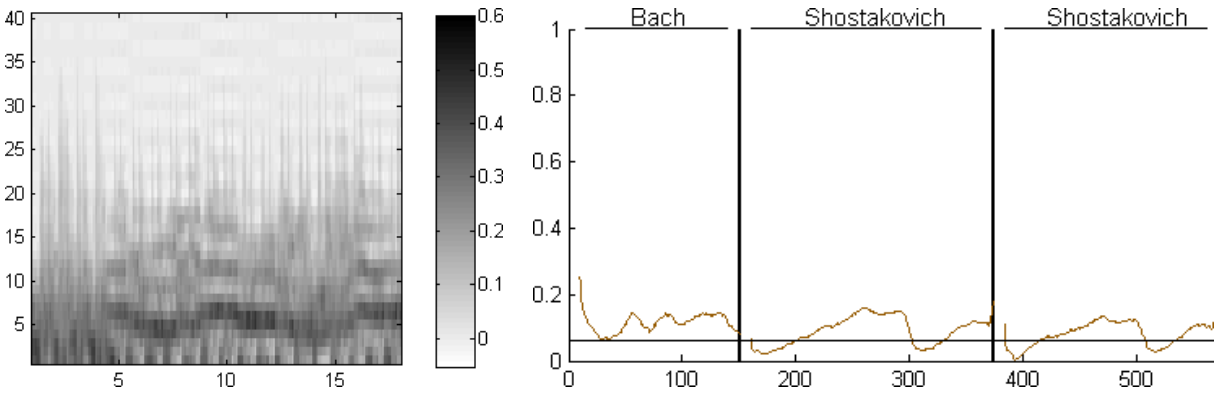
Abbildung 5.2(b) zeigt die Merkmalsfolge der Anfrage. Bei allen bisher betrachteten Merkmalen ist in der Visualisierung eine deutliche Struktur erkennbar, die Hinweise auf enthaltene Frequenzen gibt. Nach der DCT ist eine solche anschauliche Struktur nicht mehr vorhanden. Stattdessen sieht man, dass die beiden niedrigsten Koeffizienten die Merkmalsvektoren dominieren. Fast der gesamte Anteil der Energie liegt in diesen beiden Komponenten und für die anderen Einträge bleibt wenig Spielraum. Dieses Ungleichgewicht wird sich später negativ auf die Matchingergebnisse auswirken.

Einen weiteren Unterschied zu allen bisherigen Merkmalen stellt der Wertebereich der einzelnen Vektorkomponenten dar. Die bisher betrachteten Merkmalsvektoren enthalten nur positive Einträge, sodass alle Komponenten nach der Normierung in $[0, 1]$ liegen. Nach Anwenden des Logarithmus und der DCT liegen jedoch auch negative Werte vor, sodass der Wertebereich der Komponenten der MFCC-Merkmalsvektoren $[-1, 1]$ umfassen kann. Als Konsequenz kann das Skalarprodukt in der Kostenfunktion Werte aus $[-1, 1]$ liefern, anstatt nur Werte zwischen 0 und 1. Die Gesamtkostenfunktion (2.6) kann damit Werte zwischen 0 und 2 annehmen, sodass der maximale Wertebereich der Matchingkurve bei MFCC-Merkmalen $[0, 2]$ beträgt anstatt $[0, 1]$. Die Kostenfunktion erzeugt genau dann Werte größer 1, wenn das Skalarprodukt negativ ist. Dazu müssen positive Werte eines

Merkmalsvektors mit negativen aus dem anderen multipliziert werden. Damit die Kostenfunktion für negative Skalarprodukte sinnvoll bleibt, müssen negative Werte in einem Merkmalsvektor einen inhaltlichen Unterschied zu positiven darstellen und der Unterschied muss beim Absinken des negativen Wertes deutlicher werden. D.h. ein stärker negativ werdender Wert gegenüber einem positiven muss einen größer werdenden Unterschied des zugehörigen Musikabschnittes repräsentieren. Der Logarithmus erfüllt diese Bedingung als streng monoton steigende Funktion: Größere Energie in einem Bin führt zu höheren Werten auch im Intervall $(0, 1]$, welches auf negative Werte abgebildet wird. Negative Koeffizienten nach der DCT bedeuten, dass einer der Basisvektoren mit negativem Vorzeichen in die Linearkombination eingeht. Dies deutet offensichtlich auf eine komplett andere Struktur der Merkmalsvektoren vor der DCT hin, als wenn der Basisvektor mit positivem Vorzeichen addiert würde. Liefert die Kostenfunktion also Werte größer 1, so bedeutet dies auch einen entsprechend großen Unterschied der durch die Merkmalsvektoren repräsentierten Musikabschnitte. Die Matchingkurve kann also wie bisher interpretiert werden. Die meisten beobachteten Werte werden aber weiterhin im Intervall $[0, 1]$ liegen.

Zur besseren Veranschaulichung werden *rekonstruierte* MFCC-Merkmale (Bezeichnung nach dem Programm aus der MA-Toolbox) visualisiert. Dazu wird mit den fertigen, bereits normierten Merkmalsvektoren eine inverse DCT durchgeführt. Dies geschieht durch Multiplikation mit der transponierten DCT-Matrix, da diese orthogonal ist, siehe 5.2. Die DCT-Matrix ist nach Definition quadratisch, die Zahl der Koeffizienten muss also unverändert sein, damit die inverse DCT durchgeführt werden kann. Falls Koeffizienten entfernt worden sind, müssen diese vor der Multiplikation mit der inversen DCT-Matrix mit Nullen aufgefüllt werden. Das Ersetzen durch Nullen ist sinnvoll, da die inverse DCT das Signal als Linearkombination der Basisvektoren darstellt und abgeschnittene Koeffizienten zum Verlust von Basisvektoren führen. Da die rekonstruierte Version nur zur Visualisierung dient, wird keine abschließende Normierung mehr durchgeführt. Abbildung 5.5(a) zeigt die rekonstruierte Version der MFCC_{20}^{40}-Anfragemerkmale von $\mathcal{Q}^{Sho}$. Die einzelnen Vektoren haben jetzt wieder 40 Einträge, die zu den 40 Bins gehören. Eine grobe Verteilung der Energie auf die Frequenzen ist jetzt wieder erkennbar. So sieht man beispielsweise, dass die höheren Frequenzen der Anfrage schwächer vertreten sind und erkennt eine Struktur etwa bei Koeffizient 5 wieder, die an die Zerlegung in Halboktaven, siehe Abbildung 4.14(b), erinnert. Bei der schwach zu erkennenden gitterartigen Struktur im Hintergrund handelt es sich um Artefakte, die von der Reduktion der DCT-Koeffizienten stammen. Auch das verwischte Aussehen ist auf die Reduktion zurückzuführen. Im weiteren Verlauf werden Merkmalsfolgen in der anschaulicheren rekonstruierten Version dargestellt.

Die Matchingkurve ist in Abbildung 5.5(b) zu sehen. Die Kurve verläuft auf niedrigem Niveau, weil die Merkmalsvektoren sehr hohe Einträge in den beiden unteren Komponenten haben und die restlichen Komponenten entsprechend geringe Werte enthalten. Deshalb sind die Skalarprodukte nahe eins und damit die Kosten gering. Nur die ersten vier Treffer sind korrekt: Jeweils der Teil A_1, aus dem auch die Anfrage stammt, und Teil A_3 in den beiden Stücken von Shostakovich. Treffer Nummer fünf liegt im Stück von Bach. An den vier verbleibenden erwarteten Trefferpositionen A_2 und A_4 in beiden Stücken, befinden sich nichteinmal schwache Peaks. Auch sind die korrekten Peaks bis auf die Position des



(a) Merkmale nach rückgängig gemachter DCT

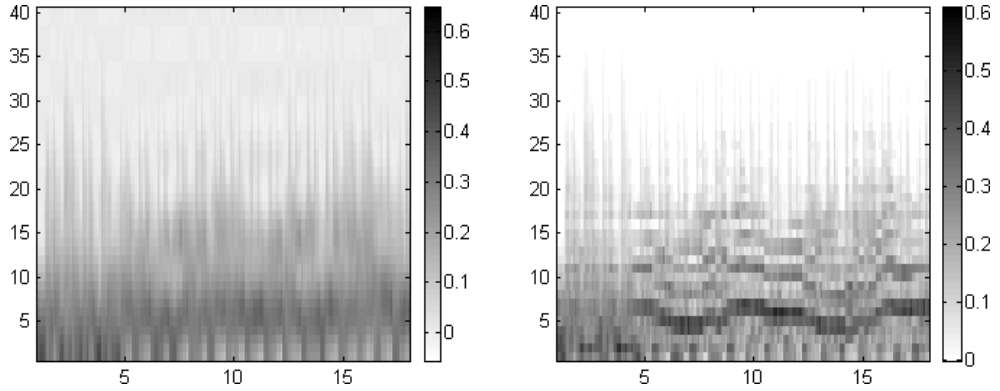
(b) Matchingkurve ($\mathcal{D}^{Sho}$, $\mathcal{Q}^{Sho}$)

Abbildung 5.5. Visualisierung der MFCC_{20}^{40}-Merkmale der Anfrage $\mathcal{Q}^{Sho}$ in „rekonstruierter" Form, d.h. auf die bereits normierten MFCC-Merkmale ist eine inverse DCT angewendet worden, um eine bessere Darstellung der Energieverteilung auf die Bins zu erzielen. 5.5(b) zeigt die Matchingkurve, die bei der Anfrage an die Datenbank $\mathcal{D}^{Sho}$ entsteht.

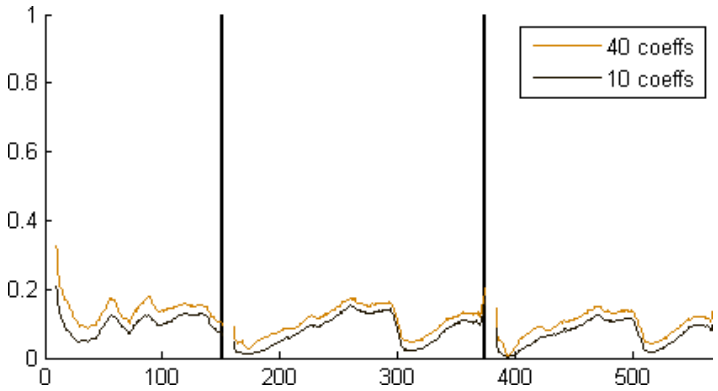
exakten Ausschnitts relativ breit und unpräzise. Mit den gewählten Parametern scheinen die MFCC-Merkmale also keine brauchbaren Ergebnisse in Bezug auf das Audiomatching zu liefern.

Die Parameter sollen nun verändert werden. Zunächst wird eine Reduktion der Koeffizienten nach der DCT auf 10 gewählt, die Bezeichnung der Merkmale lautet dann MFCC_{10}^{40}. Anfrage $\mathcal{Q}^{Sho}$ und Datenbank $\mathcal{D}^{Sho}$ des letzten Experiments werden weiterhin verwendet. In Abbildung 5.6(a) sind die Merkmalsvektoren der Anfrage in rekonstruierter Version dargestellt. Es ist sehr viel Information verloren gegangen, da nur ein Viertel der ursprünglichen Koeffizienten verwendet wird. Das Bild sieht stark verwischt aus und die Strukturen, die bei 20 Koeffizienten erkennbar gewesen sind, sind kaum mehr auszumachen. Dennoch zeigt die in Abbildung 5.6(c) sichtbare Matchingkurve kaum Unterschiede gegenüber der Kurve bei 20 Koeffizienten.

Nun wird die Zahl der Koeffizienten auf 40 festgelegt, es erfolgt also keine Reduktion nach der DCT. Die rekonstruierten MFCC_{40}^{40}-Merkmale der Anfrage sind in Abbildung 5.6(b) zu sehen. Das Bild ist wesentlich deutlicher, die Verteilung der Energie auf die einzelnen Bins ist nun klar erkennbar. Da keine Koeffizienten entfernt worden sind, liegen auch keine Artefakte mehr vor. Zu sehen sind damit die Merkmalsvektoren im Zustand direkt nach dem Binning und der Logarithmierung des Wertebereichs, die DCT ist vollkommen aufgehoben. Die Matchingkurve, siehe Abbildung 5.6(c), unterscheidet sich jedoch nur unwesentlich von den Kurven, die mit MFCC-Merkmalen mit 20 oder sogar mit nur 10 Koeffizienten erzeugt worden sind. Insbesondere gibt es bei vier von acht gewünschten Treffern nach wie vor keine entsprechende Reaktion der Kurve. Die DCT erfüllt hier also durchaus den Zweck, dass man die Datenmenge auf ein gewisses Maß reduzieren kann, ohne dass größere Änderungen auftreten. Diese Tatsache kann bei praktischer Anwendung ausgenutzt werden, um die Datenmenge der Merkmale, die für eine ganze Musikdatenbank



(a) rekonstruierte $MFCC_{10}^{40}$-Merkmale (b) rekonstruierte $MFCC_{40}^{40}$-Merkmale

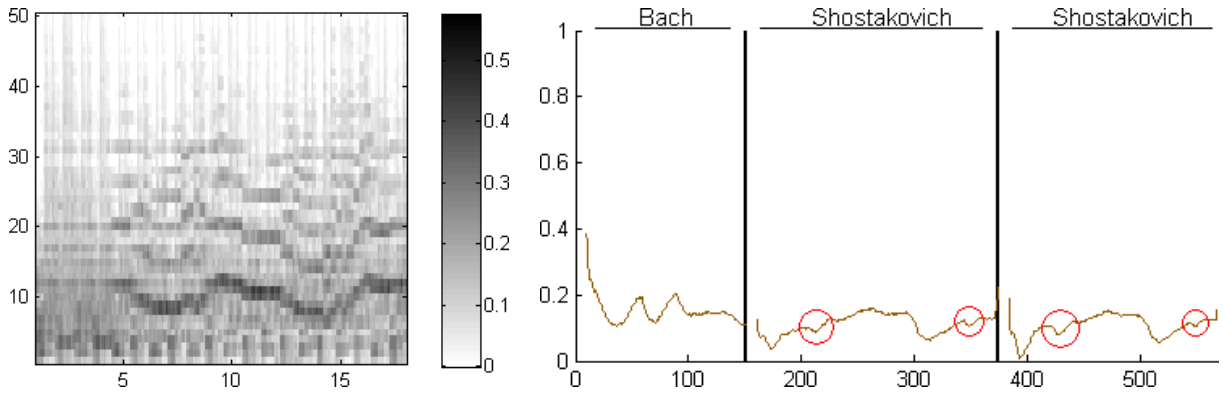


(c) Matchingkurven

Abbildung 5.6. Rekonstruierte MFCC-Merkmalsvektoren mit 40 Bins mit verschiedener Koeffizientenanzahl (Anfrage $\mathcal{Q}^{Sho}$). 5.6(c): Matchingkurven der beiden Merkmalsversionen bei Anfrage von $\mathcal{Q}^{Sho}$ auf $\mathcal{D}^{Sho}$.

abgespeichert und verarbeitet werden müssen, zu verringern. Es hat sich gezeigt, dass die Ergebnisse bei Nutzung aller höheren Koeffizienten minimal besser sind, insbesondere also keine Verbesserung durch Abschneiden höherer Koeffizienten eintritt, sodass zunächst bei den weiteren Untersuchungen kein Abschneiden der höheren Einträge mehr erfolgt. Am Ende von Abschnitt 5.5 wird diese Möglichkeit jedoch noch einmal aufgegriffen und untersucht.

Bisher ist der Bereich von 20 Hz bis 11050 Hz, also das ganze vorhandene Spektrum, betrachtet worden, wie beim Matching mit Spektralvektoren. Bei anderen Binnings, wie z.B. Pitch- oder Oktavzerlegung, wird nur ein Bereich, der in etwa dem Umfang einer Klaviertastatur entspricht, berücksichtigt. Jetzt soll auch der Bereich der MFCC-Merkmale entsprechend eingeschränkt werden. Der Frequenzbereich der Filterbank für die MFCC-Merkmale wird auf 27 Hz bis 4100 Hz gesetzt, was dem Bereich der Pitchzerlegung von $p = 21$ bis $p = 108$ nahe kommt. Zusätzlich wird die Anzahl der Filter auf 50 erhöht, dies hat sich in Experimenten als vorteilhaft erwiesen. Die Zahl der Koeffizienten wird nach der DCT auf 50 belassen. Die Bezeichnung dieser Merkmale lautet somit MFCC_{50}^{50},



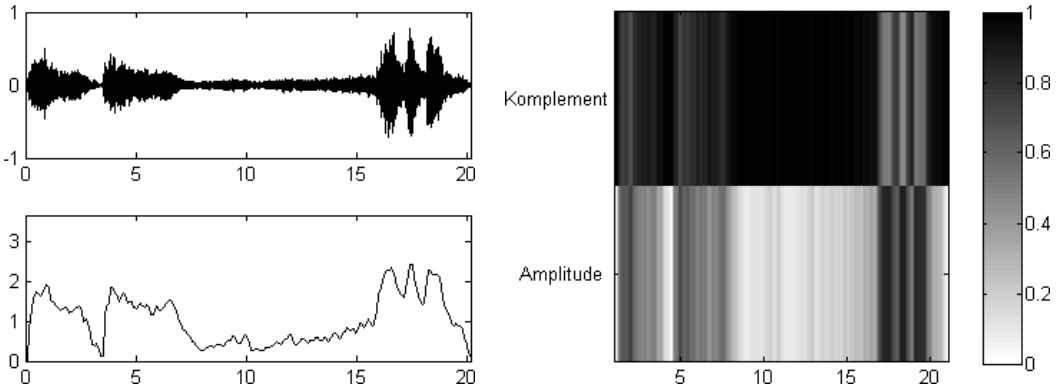
(a) rekonstruierte $MFCC_{50}^{50}$-Merkmale der Anfrage $\mathcal{Q}^{Sho}$

(b) Matchingkurve ($\mathcal{Q}^{Sho}$, $\mathcal{D}^{Sho}$)

Abbildung 5.7. $MFCC_{50}^{50}$-Merkmale, deren Binning etwa auf den Bereich der MIDI-Noten $p = 21$ bis $p = 108$ beschränkt ist.

wobei der betrachtete Frequenzbereich aus Gründen der Übersichtlichkeit nicht in der Nomenklatur auftaucht, sondern jeweils im Text angegeben wird (der hier gewählte Bereich wird, bis auf eine Ausnahme, von nun an immer verwendet). Die rekonstruierten MFCC_{50}^{50}-Merkmale der Anfrage $\mathcal{Q}^{Sho}$ sind in Bild 5.7(a) dargestellt. Gut erkennbar ist, dass der abgedeckte Frequenzbereich kleiner und die Frequenzauflösung höher ist, da mehr Fenster auf einem kleineren Intervall vorhanden sind. Abbildung 5.7(b) zeigt die Matchingkurve. An den Positionen der vier bisher gar nicht erkannten ähnlichen Passagen liegen nun leichte Peaks vor (in der Grafik eingekreist). Diese Minima sind jedoch sehr schwach und führen wegen der Breite der Peaks bei Abschnitt A_3 nicht zu Treffern. Die grundsätzlichen negativen Eigenschaften – insgesamt sehr niedrig verlaufende Kurve sowie schwache und breite Peaks – sind also weiterhin vorhanden. Dennoch hat sich durch die Einschränkung auf den Frequenzbereich, der auch bei den meisten anderen Merkmalen üblich ist, und durch die Erhöhung der Auflösung eine leichte Verbesserung eingestellt, denn vorher haben sich in der Matchingkurve an einigen zur Anfrage ähnlichen Stellen gar keine Anzeichen von lokalen Minima gezeigt.

Es stellt sich die Frage, warum die Abschnitte A_1 und A_3 relativ deutlich wiedererkannt werden, die beiden anderen Vorkommen von ähnlichen Abschnitten in beiden Stücken je nach Einstellung aber nur sehr schwach oder gar nicht zu niedrigeren Kosten führen. Wie in Abschnitt 5.2 bemerkt, stellt der Koeffizient null nach der DCT die Gesamtenergie dar. Da dieser Koeffizient sehr dominant ist gegenüber den meisten anderen Einträgen, könnte es sein, dass die Matchingkurve sehr stark vom Energieverlauf und damit vom Lautstärkeverlauf der Stücke abhängt. In der Tat ist Abschnitt A_1, aus dem die Anfrage stammt, relativ leise. Dies gilt auch für den Teil A_3, welcher ebenfalls einen gut erkennbaren Peak in der Matchingkurve liefert. A_2 und A_4 sind jedoch deutlich lauter und führen nicht zu Treffern. Dieses Verhalten soll genauer untersucht werden, dazu werden Merkmale eingeführt, die ausschließlich auf den Energieverlauf der Stücke reagieren.



(a) Signal (oben) & LSTMSP (b) Merkmale der Anfrage $\mathcal{Q}^{BeSin}$

Abbildung 5.8. Darstellung der Anfrage $\mathcal{Q}^{BeSin}$ als zeitliches Signal, als daraus berechnete LSTMSP-Kurve und schließlich als Merkmalsfolge, welche durch Zusammensetzen der LSTMSP-Werte mit ihrem Komplement (siehe Text) entsteht.

5.4 Amplitudenmerkmale

Die Pitchmerkmale werden erzeugt, indem das Signal in Bänder zerlegt wird und in jedem Band lokale Energien berechnet werden. Die Pitchmerkmale geben also den Energie- und damit den Lautstärkeverlauf für jedes Band an. Die Idee ist nun, das Signal nicht erst zu zerlegen, sondern direkt auf den PCM-Daten die STMSP-Berechnung durchzuführen. Auf diese Weise erhält man Informationen über den Lautstärkeverlauf des ganzen Signals.

Die lokale Energie wird wie in Abschnitt 3.3.2 berechnet durch Faltung des punktweise quadrierten Signals mit einem Rechteckfenster und anschließendem Downsampling. Formal sind die STMSP-Werte weiterhin definiert durch Formel (3.7), wobei x jedoch das gesamte Signal und kein Subband mehr ist. Wie bei der Berechnung der MFCC-Merkmale soll die logarithmische Lautstärkewahrnehmung des Menschen berücksichtigt werden. Dazu werden bei einem Signal x der Länge $N \in \mathbb{N}$ und einer Fensterlänge $w \in \mathbb{N}$ für alle $n \in [1 : N]$ die STMSP-Werte logarithmiert:

$$\text{lstmsp}_{x,w}(n) := \log_{10}(\text{stmsp}_{x,w}(n) + 1) \tag{5.7}$$

Die Addition der 1 verhindert, dass negative Werte auftreten und sorgt dafür, dass Energiewerte von Null auch nach der Umrechnung noch Null ergeben.

Im oberen Teil von Abbildung 5.8(a) ist ein 21-sekündiger Abschnitt aus Beethovens Fünfter Sinfonie, nämlich Anfrage $\mathcal{Q}^{BeSin}$, als zeitliches Signal dargestellt. Die PCM-Daten liegen, wie in Abschnitt 3.1 erläutert, im Wertebereich $[-1, 1]$. Darunter sind die logarithmierten STMSP-Werte zu sehen. Man erkennt, dass in Bereichen, in denen vermehrt hohe Amplituden im Signal auftreten, auch die Energiekurve entsprechend höhere Werte annimmt. Wegen der logarithmischen Darstellung reagiert die Kurve bei geringeren Amplituden stärker auf kleinere Änderungen als bei höheren Amplituden. Die Fensterbreite bei der Berechnung der STMSP-Werte beträgt 4410. Der maximale Energiewert, der innerhalb

eines Fensters auftreten kann, ist damit 4410, nämlich wenn alle Signalwerte im Fenster Betrag 1 haben. Durch Anwendung des Logarithmus, siehe Formel (5.7), wird daraus etwa 3.6445. Dies ist der größte Wert, den die Energiekurve annehmen kann, dadurch ist die Y-Achsenbeschriftung in der Abbildung motiviert.

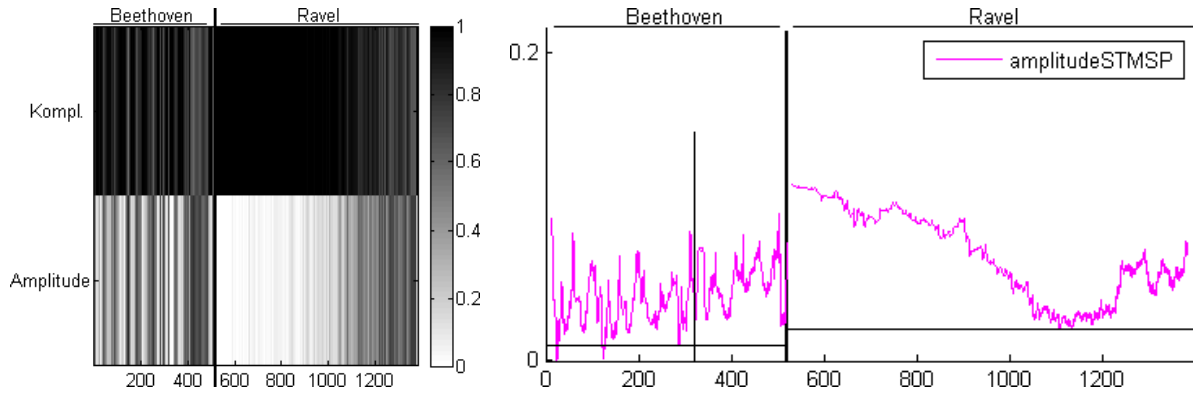
Möchte man die resultierenden Werte als Merkmale verwenden, ergibt sich ein Problem. Das definierte $\text{lstmsp}_{x,w}$ stellt nur eine Folge von einzelnen logarithmierten Energiewerten dar. Soll das bisherige Matchingverfahren mit dem euklidischen Skalarprodukt als Abstandsmaß weiterhin verwendet werden, benötigt man eine Folge von Vektoren, die man L_2-normieren kann. Das Problem wird durch eine künstliche Erweiterung der Werte zu Vektoren mit zwei Einträgen gelöst. Wie bereits festgestellt beträgt der maximal mögliche logarithmierte Wert bei einer Fensterbreite von w

$$m := \log_{10}(w + 1) \tag{5.8}$$

Zu jedem STMSP-Wert s der Folge wird das Komplement $m - s$ berechnet und beide Werte zu einem Merkmalsvektor $(s, m - s)^T$ zusammengesetzt. Nach der Normierung stellen die Merkmalsvektoren anschaulich gesprochen also Vektoren der Länge 1 in der euklidischen Ebene dar, deren Winkel zu $(0, 1)^T$ mit steigender Energie des Signals zunehmen. Das Skalarprodukt bestimmt den Winkel zwischen zwei Merkmalsvektoren, der wiederum von der Energie der Signale abhängt, sodass man letztendlich bei größeren Energieunterschieden auch höhere Kosten erhält.

Die Merkmalsvektoren zu dem in Abbildung 5.8(a) gezeigten Abschnitt von Beethovens Fünfter Sinfonie sind in 5.8(b) dargestellt. Die Fensterbreite beträgt 4410 und die Überlappung 2205, woraus die übliche Merkmalsrate von 10 resultiert. Die Komponente „Amplitude“ enthält die STMSP-Daten, man erkennt den Verlauf der Energiekurve wieder. Das Komplement nimmt überwiegend Werte nahe eins an. Dies liegt daran, dass die Energie des Signals in den meisten Fällen deutlich unter dem Maximalwert liegt, welcher den theoretisch größten Wert darstellt, jedoch in der Praxis von einem Musikstück nicht erreicht wird. Hier zeichnet sich ein Nachteil des Verfahrens ab: In den meisten Fällen wird das Komplement den größten Anteil haben, was den Bereich, in dem sich die meisten Merkmalsvektoren von Datenbank und Anfrage befinden, einschränkt und zu weniger differenzierten Ergebnissen führt.

Für das Audiomatching wird die Datenbank $\mathcal{D}^{Ravel}$ mit einer Orchesterversion von Beethovens Fünfter Sinfonie und dem Stück „Bolero“ von M. Ravel verwendet, Anfrage ist $\mathcal{Q}^{BeSin}$. Die Merkmalsfolge der Datenbank ist in Abbildung 5.9(a) zu sehen. Die Sinfonie enthält Abschnitte größerer Lautstärke, die sich mit leiseren Passagen abwechseln, während „Bolero“ eine durchgehende Zunahme der Lautstärke vom Anfang bis zum Ende aufweist. Dieses Verhalten der Lautstärke ist an den Merkmalsvektoren zu erkennen. In Abbildung 5.9(b) ist die Matchingkurve visualisiert. Die Abbildung ist bezüglich der Y-Achse um den Faktor fünf vergrößert, da die gesamte Kurve niedrige Werte annimmt und sich kaum über den Wert 0.1 erhebt. Die niedrigen Kosten resultieren einerseits wieder aus der starken Vergröberung und dem Verlust aller Information außer der Lautstärke. Zusätzlich besitzen die meisten Merkmalsvektoren von Anfrage und Datenbank, wie oben erwähnt, einen großen



(a) Merkmale der Datenbank $\mathcal{D}^{Ravel}$

(b) Matchingkurve ($\mathcal{D}^{Ravel}$, $\mathcal{Q}^{BeSin}$)

Abbildung 5.9. Audiomatching mit Amplitudenmerkmalen: Links sind die Merkmalsvektoren zu sehen. Die meisten Merkmalsvektoren sind einander ähnlich, da das Komplement den größten Anteil hat. Aus vielen ähnlichen Merkmalsvektoren resultiert eine sehr niedrig verlaufende Matchingkurve. Der Wertebereich der Kurve ist vergrößert dargestellt.

Eintrag in der Komponente, die das Komplement enthält. Deshalb ergeben die meisten Paare von Merkmalsvektoren relativ hohe Skalarprodukte und damit niedrige Kosten.

Nur die beiden ersten Treffer sind korrekt. Der dritte Treffer ist ein deutlicher Peak, liegt jedoch nicht an einer Position, die das Thema der Anfrage enthält, sondern an einer Stelle, die nur eine ähnliche Lautstärke hat wie die Anfrage. Umgekehrt stimmt die Lautstärke des dritten Abschnitts (in der Abbildung 5.9(b) durch eine senkrechte Linie markiert), der zur Anfrage ähnlich ist, nicht mit der Lautstärke der Anfrage überein. Deshalb erhält man hier recht hohe Kosten und einen sehr schwachen Peak.

Bemerkenswert ist auch der Verlauf der Kurve in „Bolero“. Das Stück beginnt deutlich leiser, als die Anfrage aus der Sinfonie im Durchschnitt ist. Deshalb sind die Kosten zu Beginn vergleichsweise hoch. Die Lautstärke steigt im Verlauf des Stückes immer weiter an und wird der Lautstärke der Anfrage immer ähnlicher, sodass die Kosten sinken, bis in etwa die Lautstärke der Anfrage erreicht ist. Natürlich ist der Lautstärkeverlauf der Anfrage anders als im Stück Bolero, sodass die Kosten nicht Null werden. Aber allein die größere Ähnlichkeit der durchschnittlichen Lautstärke führt zu geringeren Kosten als in der Umgebung. Schließlich wird das Stück von Ravel noch lauter und die Kosten steigen wieder an.

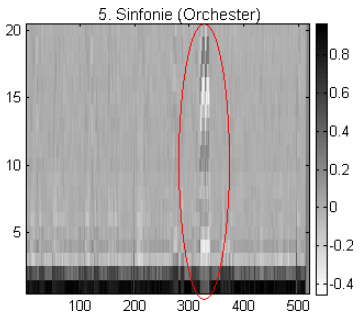
Merkmale, die nur auf die Lautstärke reagieren, sind für das Audiomatching unbrauchbar. Liegt beispielsweise das gleiche Stück in zwei unterschiedlichen Gesamtlautstärken vor, so wird keine Ähnlichkeit erkannt. Tatsächlich wird in den meisten Fällen versucht, die Merkmale invariant gegenüber der Lautstärke zu machen, so z.B. bei den CENS-Merkmalen durch die Normierung vor der Quantisierung oder allgemein durch Normieren der Merkmale vor dem Einsatz, wobei die Amplitudenmerkmale durch die Art der Konstruktion trotz Normierung die Lautstärkeinformation behalten. Im Folgenden sollen die Matchingkurven von Amplituden- und MFCC-Merkmalen verglichen werden, um die Abhängigkeit der MFCC-Merkmale von der Lautstärke zu untersuchen.

5.5 Modifikation der MFCC-Merkmale

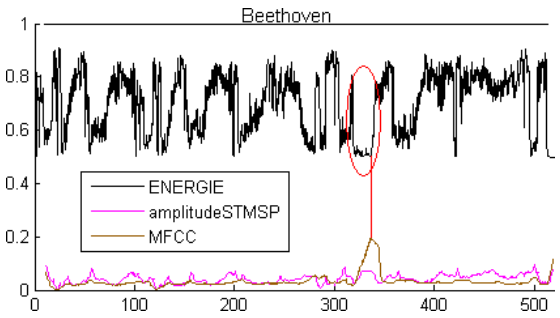
Zunächst enthält die Datenbank nur eine Version von Beethovens Fünfter Sinfonie. Anfrage ist der bekannte Abschnitt $\mathcal{Q}^{BeSin}$ aus der Exposition. Es wird jeweils ein Audiomatching mit Amplitudenmerkmalen und mit MFCC_{20}^{40}-Merkmalen durchgeführt. Die Zahl der Bins und Koeffizienten wird wieder reduziert (50 Koeffizienten haben sich ja als etwas besser herausgestellt), um den Effekt etwas deutlicher zu machen, er tritt aber auch bei einer größeren Zahl von Koeffizienten auf. Abbildung 5.10(a) zeigt die MFCC-Merkmalsfolge der fünften Sinfonie in nicht rekonstruierter Form. An den meisten Stellen sehen die Merkmale recht gleichmäßig ohne besondere Struktur aus und der unterste Koeffizient macht den größten Anteil aus. Bei etwa 330 Sekunden wird der gleichmäßige Verlauf jedoch unterbrochen. Der unterste Koeffizient nimmt hier einen geringeren Wert an. Da dieser Koeffizient die Gesamtenergie des Signals innerhalb der Zeitfenster angibt, muss an dieser Stelle also eine deutlich geringere Lautstärke als an den anderen Positionen vorliegen. Tatsächlich liegt hier ein Oboensolo vor, welches deutlich leiser ist, als der vom gesamten Orchester gespielte Rest des Stückes. Da der Koeffizient null an dieser Stelle nicht mehr dominant ist, können sich die anderen Komponenten nach der Normierung stärker durchsetzen und zur Strukturbildung der betroffenen Merkmalsvektoren deutlicher beitragen. Besonders leise Passagen heben sich also auf Merkmalsebene von den meisten anderen Abschnitten deutlich ab. Dieses Phänomen wirkt sich auch auf die in Abbildung 5.10(b) gezeigte Matchingkurve aus. Eingezeichnet sind neben der MFCC-Matchingkurve auch die der Amplitudenmerkmale (amplitudeSTMSP) und die Energiekurve[3], siehe Formel (5.7), die den Energieverlauf des Stückes angibt. Wie am sehr einheitlichen Aussehen der MFCC-Merkmalsfolge schon erkennbar, verläuft die Matchingkurve überwiegend auf gleicher Ebene. Diese Ebene ist sehr niedrig, da auch die Anfrage aus dem einheitlichen Teil vom Beginn des Stückes bis Sekunde 21 stammt. Eine Ausnahme bildet ein nach oben gerichteter Peak etwa bei Sekunde 330. Dies ist genau die Stelle, an der die Merkmalsfolge eine veränderte Struktur aufweist, was erwartungsgemäß zu höheren Kosten führt. Beim Betrachten der Energiekurve erkennt man, dass das Stück dort über einen Zeitraum von einigen Sekunden sehr leise ist. Dies stellt, wie bereits erwähnt, die Ursache für die anders aufgebauten Merkmalsvektoren in dieser Passage dar. Das Maximum des Peaks liegt genau am Ende der leisen Passage, weil sich die Matchingkurve auf die Endpunkte der DTW-Pfade bezieht und somit die Kosten, die zu dem Peak führen, von den zeitlich davor liegenden Merkmalsvektoren stammen. Der Energieverlauf ist jedoch auch an anderen Stellen nicht konstant, sondern schwankt stark. Trotzdem bleibt der Energiekoeffizient, also der Koeffizient null der Merkmalsvektoren, vergleichsweise hoch, sodass er nach der Normierung den weitaus größten Anteil hat. Nur eine sehr leise Passage kann also zu einer deutlichen Strukturveränderung der Merkmalsvektoren führen.

Die Matchingkurve der Amplitudenmerkmale verläuft auf ähnlichem Niveau, weist jedoch nicht einen so deutlichen Peak bei Sekunde 330 auf. Dieses Phänomen ist darauf

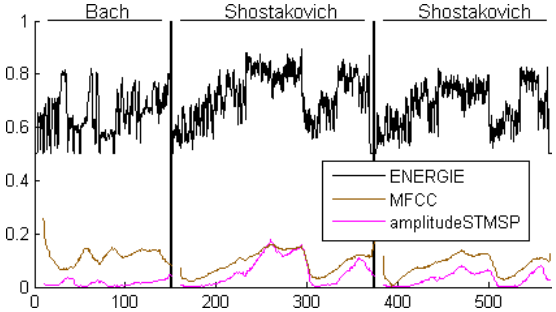
[3]Der Wertebereich der Energiekurve (LSTMSP) liegt zwischen 0 und 3.6. Hier ist zur besseren Übersicht die Kurve skaliert und verschoben dargestellt. Sie soll als Anhaltspunkt für den Verlauf der Lautstärke dienen. Genaue Werte sind hier nicht wichtig.



(a) Merkmale extrahiert aus der 5. Sinfonie



(b) Matchingkurve ($\mathcal{Q}^{BeSin}$ angefragt auf der 5. Sinfonie)



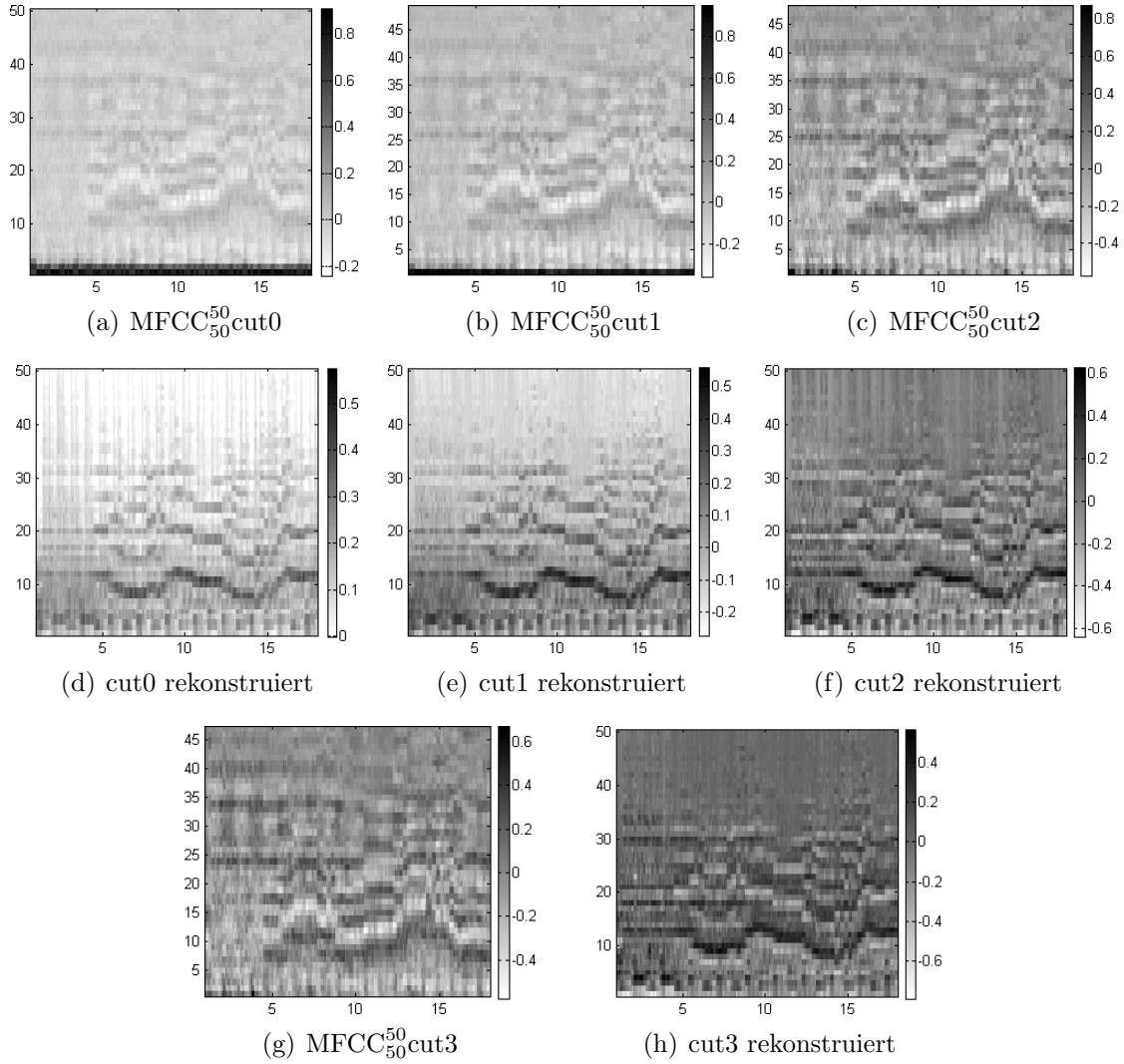
(c) Matchingkurve ($\mathcal{Q}^{Sho}$, $\mathcal{D}^{Sho}$)

Abbildung 5.10. Vergleich von MFCC_{20}^{40}-Merkmalen mit Amplitudenmerkmalen. An einigen Stellen ist eine deutliche Abhängigkeit der MFCC-Merkmale von der Lautstärke zu beobachten.

zurückzuführen, dass der Wertebereich der Einträge der MFCC-Merkmalsvektoren negative Werte umfasst. Die Kosten können sich daher in einem größeren Bereich bewegen und negative Einträge eines Merkmalsvektors, die auf positive eines anderen Vektors treffen, können beim Skalarprodukt schnell zu vergleichsweise hohen Kosten führen.

Abbildung 5.10(c) zeigt die Matchingkurve der MFCC_{20}^{40}-Merkmale auf der Datenbank $\mathcal{D}^{Sho}$ mit Anfrage $\mathcal{Q}^{Sho}$. Da die Anfrage leise ist, weist die Kurve der Amplitudenmerkmale bei lauten Abschnitten hohe Kosten auf und bei leiseren Abschnitten niedrige. Dieses Verhalten ist gut zu erkennen, wenn man die Matchingkurve mit der Energiekurve vergleicht. In den Stücken von Shostakovich zeigt die Kurve der MFCC-Merkmale nicht nur die gleiche Tendenz, sondern auch die Form und teilweise sogar die absoluten Werte der Kurve sind ähnlich zu denen der Amplitudenmerkmale. Im Stück von Bach sind aber auch Abweichungen zu sehen. Die MFCC-Merkmale mit den hier gewählten Parametern sind damit nicht rein von der Amplitude bestimmt, jedoch wegen des sehr großen Anteils des untersten Koeffizienten sehr stark von der Gesamtenergie abhängig. Deshalb können im Shostakovich Beispiel die lauter gespielten Versionen des Themas aus der Anfrage nicht gefunden werden.

Ein Versuch, dieses Problem zu beheben, besteht darin, den untersten Koeffizient zu entfernen, genauso wie bisher höhere Koeffizienten entfernt worden sind. Die Bezeichnung für MFCC-Merkmale wird erweitert, um diese Modifikation im Namen aufzuneh-



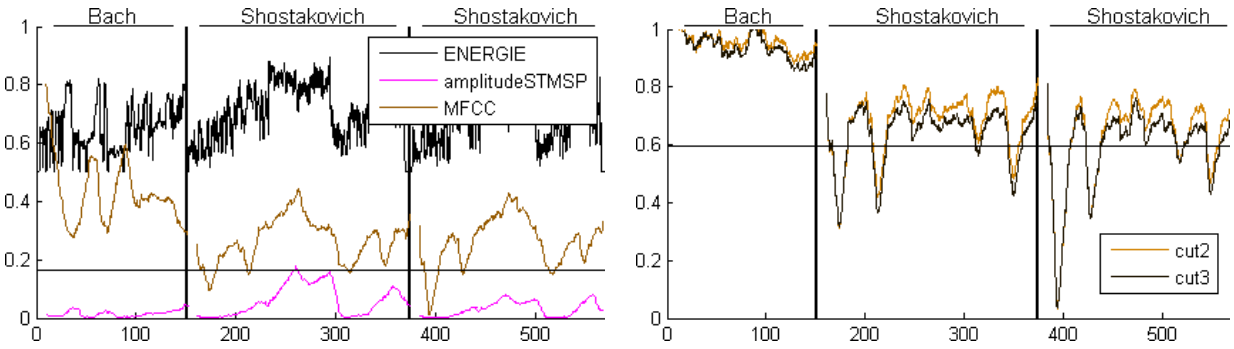
(a) MFCC_{50}^{50}cut0 (b) MFCC_{50}^{50}cut1 (c) MFCC_{50}^{50}cut2

(d) cut0 rekonstruiert (e) cut1 rekonstruiert (f) cut2 rekonstruiert

(g) MFCC_{50}^{50}cut3 (h) cut3 rekonstruiert

Abbildung 5.11. Merkmalsvektoren der Anfrage $\mathcal{Q}^{Sho}$. Dargestellt sind MFCC-Merkmale mit abgeschnittenen unteren Koeffizienten in original und rekonstruierter Form.

men: MFCC_c^bcut n bedeutet, dass ein Binning mit b Fenstern durchgeführt wird, nach der DCT die Zahl der Koeffizienten durch Abschneiden von hohen Koeffizienten auf c reduziert wird und dann noch die n niedrigsten Koeffizienten entfernt werden, also letztendlich $c - n$ Koeffizienten verbleiben. Falls, wie bisher, der cut-Wert nicht angegeben wird, ist immer cut0 gemeint, d.h. kein unterer Koeffizient wird entfernt. In Abbildung 5.11(b) ist die MFCC_{50}^{50}cut1-Merkmalsfolge (d.h. der Koeffizient mit Index null ist abgeschnitten) zu $\mathcal{Q}^{Sho}$ dargestellt und darunter die rekonstruierte Version. Die Struktur der rekonstruierten Merkmale ist die gleiche wie in Abbildung 5.11(a), wo keine Koeffizienten abgeschnitten worden sind. Die absoluten Werte haben sich jedoch geändert, weil die Energieinformation mit dem nullten Koeffizient verloren gegangen ist. Die Matchingkurve ist in Bild 5.12(a) visualisiert. Das durchschnittliche Niveau der Kurve ist deutlich höher als in den bisherigen



(a) Matchingkurve cut1 (b) Matchingkurven cut2 und cut3

Abbildung 5.12. Matchingkurven ($\mathcal{Q}^{Sho}$, $\mathcal{D}^{Sho}$) bei Verwendung von MFCC_{50}^{50}cut-Merkmalen. Bei cut1 ist noch eine gewisse Abhängigkeit von der Lautstärke zu erkennen, bei größerer Anzahl von abgeschnittenen unteren Koeffizienten nicht mehr, stattdessen verbessert sich das Ergebnis deutlich. In Abbildung 5.12(b) im Stück von Bach ist zu erkennen, dass die Kurve beim Matching mit MFCC-Merkmalen Werte größer eins annehmen kann.

Versuchen zu MFCC-Merkmalen, da die Merkmale wegen des fehlenden dominanten Koeffizienten mehr Struktur enthalten, die zu Unterschieden und höheren Kosten führen kann. An den acht zur Anfrage ähnlichen Positionen sind jetzt Peaks gut erkennbar und die ersten sechs Treffer sind korrekt. Das Entfernen des Energiekoeffizienten hat also deutliche Verbesserungen bewirkt. Die lokalen Minima setzen sich jedoch recht schwach von der Umgebung ab und überraschender Weise ist an einigen Stellen immer noch eine gewisse Abhängigkeit von der Lautstärke zu erkennen. Dies hängt mit dem ebenfalls dominanten ersten Koeffizient, siehe Abbildung 5.11(b), zusammen. Betrachtet man die DCT-Basisvektoren aus Abbildung 5.4, so erkennt man, dass der erste Koeffizient zu einem nach oben abfallenden Basisvektor gehört. Bei klassischer Musik sind typischerweise niedrige Frequenzen relativ stark vertreten und nach oben wird die Energie im Allgemeinen geringer, wie man an den Spektrogrammbeispielen in dieser Arbeit gut erkennen kann. Die Mel-Vektoren vor der DCT tendieren also dazu, hohe untere Koeffizienten und geringer werdende höhere zu enthalten. Die abfallende Tendenz muss durch die Linearkombination der Basisvektoren dargestellt werden, dadurch wird der erste Basisvektor besonders begünstigt und der Koeffizient Nummer eins meist stark gewichtet. Die Werte des Koeffizienten sinken, wenn in den unteren Frequenzen weniger Energie enthalten ist. Dann liegt in den meisten Fällen eine leisere Passage vor, da selten nur in höheren Frequenzen viel Energie enthalten ist. Deshalb ist immer noch eine gewisse Abhängigkeit der Merkmale von der Lautstärke gegeben.

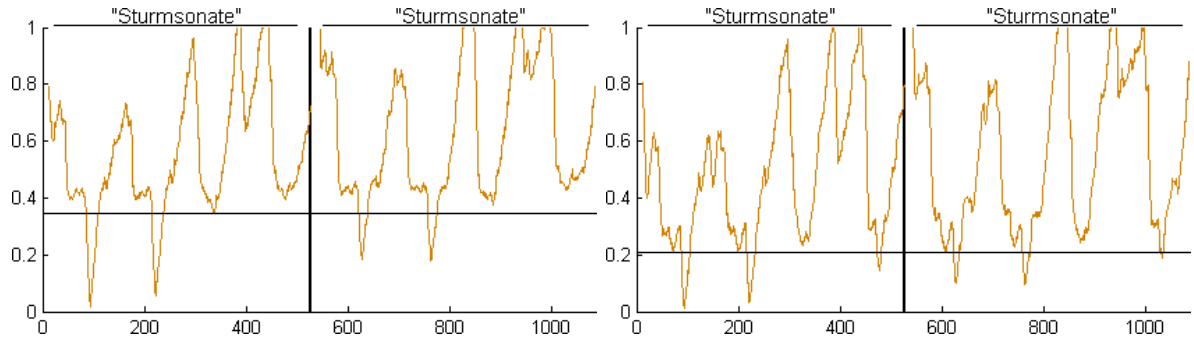
Um die störende Dominanz des ersten Koeffizienten und die verbliebene Lautstärkeabhängigkeit zu beseitigen, soll nun auch der Koeffizient eins entfernt werden. Zunächst scheint dieses Vorgehen der Idee der DCT zu widersprechen, die ja gerade die meiste Information in die unteren Koeffizienten verschiebt, damit die oberen, die weniger Information enthalten, abgeschnitten werden können. Aber auch bei den bisherigen Merkmalen wird bewusst auf Information verzichtet, um robust gegenüber Abweichungen zu sein. Das Abschneiden der für den Informationsgehalt eigentlich besonders wichtigen unteren Koef-

fizienten dient also auch diesem Zweck. Die MFCC_{50}^{50}cut2-Merkmalsfolge ist in Abbildung 5.11(c) dargestellt, darunter ist die rekonstruierte Version abgebildet. Die Koeffizienten der rekonstruierten Version fallen nicht mehr nach oben hin ab, wie es bei den bisher betrachteten MFCC-Merkmalen deutlich der Fall gewesen ist. Die Begründung stellt der fehlende erste Basisvektor dar, der die nach oben abfallende Tendenz repräsentiert und, wie im letzten Absatz bemerkt, meist stark gewichtet ist. Zum Vergleich werden auch noch Merkmale mit drei abgeschnittenen Koeffizienten generiert (MFCC_{50}^{50}cut3), sie sind in den Abbildungen 5.11(g) und 5.11(h) zu sehen. Der Wertebereich der rekonstruierten Version wird zwar nicht kleiner, die meisten Einträge haben jetzt jedoch recht ähnliche Werte. Daran wird der Informationsverlust deutlich.

Abbildung 5.12(b) zeigt die Matchingkurven (immer noch Anfrage $\mathcal{Q}^{Sho}$ auf $\mathcal{D}^{Sho}$) der MFCC-Merkmale mit zwei bzw. drei entfernten unteren Koeffizienten. Zunächst einmal fällt auf, dass beide Kurven ziemlich ähnlich sind, insbesondere wird durch das Abschneiden des dritten Koeffizienten bei diesem Beispiel kein weiterer Vorteil mehr erzielt. Die Ebene, auf der sich die Kurven befinden, ist noch einmal deutlich höher als bei der Version mit einem entfernten Koeffizient. Im Stück von Bach überschreitet die Kurve sogar den Wert 1, was wie bereits erwähnt wegen der vorhandenen negativen Einträge der MFCC-Merkmalsvektoren möglich ist. Beide Kurven zeigen acht korrekte Treffer, wobei die meisten Peaks deutlicher abgegrenzt sind als bei den vorherigen MFCC-Merkmalen. Das Abschneiden von zwei bzw. drei Koeffizienten hat also eine entscheidende Verbesserung gebracht. Das Entfernen weiterer Koeffizienten ändert zunächst wenig an den Kurven und führt bei weiterer Reduktion irgendwann zu schlechteren Ergebnissen.

Im Allgemeinen haben sich relativ präzise Merkmale mit recht vielen Bins ohne Glättung durch Entfernen von höheren Koeffizienten als günstig erwiesen. In speziellen Situationen kann sich jedoch eine stärkere Vergröberung positiv auswirken. Abbildung 5.13(a) zeigt die Matchingkurve, die bei Anfrage von $\mathcal{Q}^{BeSo}$ an die Datenbank $\mathcal{D}^{BeSo}$ unter Verwendung von MFCC_{50}^{50}cut2-Merkmalen entsteht. Beide in der Datenbank enthaltenen Versionen von Beethovens „Klaviersonate Nr. 17“ enthalten das Thema aus der Exposition dreimal, beim dritten Vorkommen jedoch transponiert. Die Matchingkurve weist pro Stück zwei korrekte Treffer auf, die transponierte Stelle wird nicht gefunden. In Abbildung 5.13(b) ist die Matchingkurve bei gleicher Datenbank und Anfrage unter Verwendung von MFCC_{13}^{40}cut2-Merkmalen dargestellt. Zusätzlich ist der Frequenzbereich wieder bis 11025 Hz vergrößert, um die Breite der einzelnen Bins weiter zu erhöhen. Nun sind alle sechs Treffer korrekt. Obwohl die Komponenten der Merkmalsvektoren nicht wie in Abschnitt 3.3.5 verschoben worden sind, wird auch die transponierte Version des Themas wiedergefunden. Da die einzelnen Bins keinen MIDI-Pitches entsprechen, würde ein Verschieben auch keinen Sinn ergeben, um transponierte Passagen aufzufinden.

Ein Grund dafür, dass die Ähnlichkeit der Anfrage zum transponierten Teil erkannt wird, ist die Größe der Bins. Ein Teil der Noten aus der Anfrage, die sich in einem bestimmen Bin befinden, liegt auch in der transponierten Version noch im gleichen Frequenzfenster. So weisen die Merkmalsvektoren der transponierten Version eine ähnliche Struktur wie die der Anfrage auf. Auch wenn nur die Anzahl der Bins reduziert und damit deren Breite erhöht wird, ist schon ein Peak an der transponierten Stelle zu erkennen, der jedoch nicht



(a) Matchingkurve $\text{MFCC}_{50}^{50}\text{cut2}$ (b) Matchingkurve $\text{MFCC}_{13}^{40}\text{cut2}$

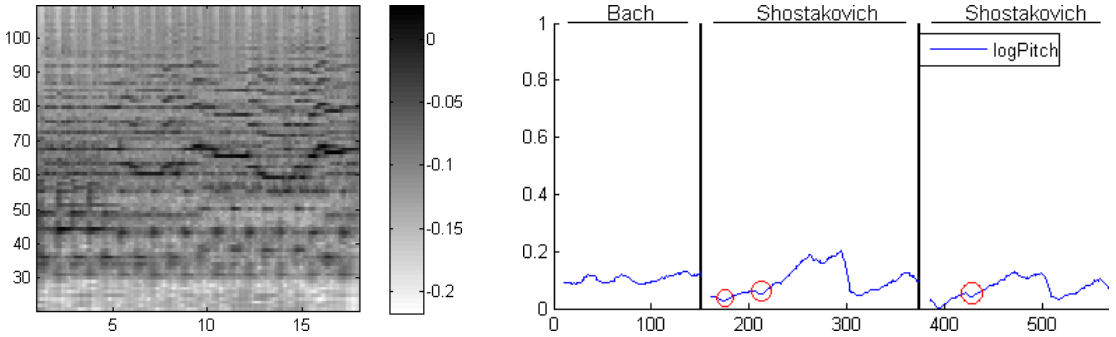
Abbildung 5.13. Matchingkurven bei Anfrage von $\mathcal{Q}^{BeSo}$ an $\mathcal{D}^{BeSo}$. Die beiden Versionen der Beethovensonate enthalten das Thema der Anfrage je dreimal, davon jedoch einmal in transponierter Form. Bei den stärker vergröberten MFCC-Merkmalen sind die Peaks zwar nicht so deutlich, allerdings kann auch die transponierte Stelle gefunden werden, obwohl keine Verschiebung der Vektorkomponenten durchgeführt wird.

deutlich genug ist, um zum Treffer zu führen. Mit der zusätzlichen Vergröberung durch das Entfernen einer großen Anzahl von DCT-Komponenten wird dann der Treffer erreicht. Für diesen speziellen Fall ist also eine stärker vergröberte Form der MFCC-Merkmale besser geeignet. Allerdings fällt beim Betrachten der anderen Treffer auf, dass die Peaks nicht mehr so deutlich sind, wie bei den genaueren $\text{MFCC}_{50}^{50}\text{cut2}$-Merkmalen. Andererseits wird deutlich, dass auch eine starke Reduktion der DCT-Koeffizienten das Ergebnis nicht zerstören muss, sodass man in der Praxis durchaus eine Platzersparnis durch Verzicht auf Koeffizienten erreichen kann, ohne entscheidend schlechtere Ergebnisse zu erhalten.

In kleineren Experimenten hat sich, abgesehen vom oben erwähnten Spezialfall, eine Filterbank mit Bereich etwa von MIDI-Pitch $p = 21$ bis $p = 108$ und 50 Filtern bewährt, wobei nach der DCT die beiden unteren Koeffizienten abgeschnitten werden, jedoch keine höheren Koeffizienten, d.h. die Version $\text{MFCC}_{50}^{50}\text{cut2}$ hat sich bewährt. Im folgenden Test verschiedener Merkmale auf einer größeren Datenbank, siehe Abschnitt 5.7, werden MFCC-Merkmale mit diesen Parametern verwendet. Die MFCC-Merkmale bieten jedoch noch viele Möglichkeiten der Modifikation und evtl. Verbesserung. Beispielsweise könnten andere Fensterbreiten und Überlappungen beim Zerlegen des Signals vor der DFT ausprobiert werden oder eine andere Anzahl von Filtern bzw. eine andere Auswahl der Koeffizienten nach der DCT. Auch eine unterschiedliche Gewichtung der Koeffizienten könnte versucht werden, z.B. Abschwächung der dominanten Einträge anstatt komplettes Entfernen.

5.6 Logarithmierte Pitchmerkmale

Die MFCC-Merkmale basieren auf einer Zerlegung des Spektrums anhand der Melskala, welche dann weiterverarbeitet wird. Statt des Melbinnings wird nun eine Pitchzerlegung als Ausgangspunkt erzeugt und diese logarithmiert und einer DCT unterzogen. Diese neuen



(a) logPitch-Merkmale von $\mathcal{Q}^{Sho}$ (b) Matchingkurve bei Anfrage an $\mathcal{D}^{Sho}$

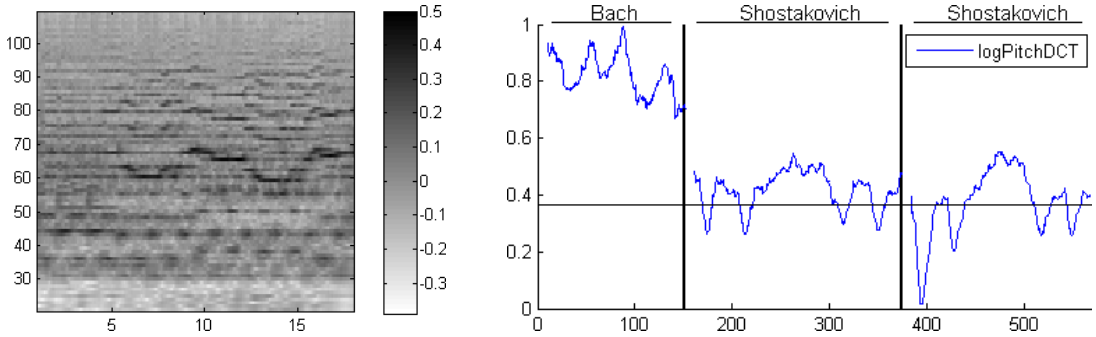
Abbildung 5.14. Die Matchingkurve der logarithmierten Pitch-Merkmale ist sehr ähnlich zu der Kurve der MFCC_{50}^{50}-Merkmale. Insbesondere liegt eine starke Abhängigkeit von der Lautstärke vor.

Merkmale werden kurz vorgestellt, jedoch nicht mehr im Detail untersucht.

Zunächst wird eine Pitchzerlegung im Bereich $p = 21$ bis $p = 108$ durchgeführt und die STMSP-Merkmalsvektoren gebildet. Von jedem einzelnen Vektoreintrag wird, wie bei den MFCC-Merkmalen, der Logarithmus zur Basis 10 bestimmt. In einem ersten Experiment wird diese Merkmalsfolge L_2-normiert und ohne DCT zum Audiomatching eingesetzt. Abbildung 5.14(a) zeigt die Merkmalsfolge der Anfrage $\mathcal{Q}^{Sho}$ und 5.14(b) die Matchingkurve bei Anfrage an die Datenbank $\mathcal{D}^{Sho}$. Es ist eine große Ähnlichkeit zur Kurve der MFCC_{50}^{50}-Merkmale, siehe Abbildung 5.7(b), zu erkennen. Insbesondere liegt also weiterhin eine starke Abhängigkeit von der Lautstärke vor. Schwächere lokale Minima sind zur Verdeutlichung eingekreist. Wie bei den MFCC-Merkmalen befinden sich diese Minima zwar an Positionen zur Anfrage ähnlicher Abschnitte, allerdings genügen sie meist nicht, um zu Treffern zu führen.

Analog zur Berechnung der MFCC-Merkmale wird nun nach der Logarithmierung der einzelnen Werte eine DCT durchgeführt. Auch für die logPitch-Merkmale hat sich gezeigt, dass die Lautstärkeabhängigkeit durch Entfernen der beiden niedrigsten Koeffizienten beseitigt werden kann. Nach abschließender L_2-Normierung unterscheiden sich die logarithmierten Pitchmerkmale nur noch durch die Anzahl und die Anordnung der Bins von den MFCC-Merkmalen, sodass entsprechend die Bezeichnung $\text{logPitch}_{88}^{88}$cut2 verwendet wird. Abbildung 5.15(a) zeigt die Merkmalsfolge der Anfrage $\mathcal{Q}^{Sho}$ in rekonstruierter Form. Die grundsätzliche Struktur hat sich gegenüber den Merkmalen ohne DCT nicht verändert, jedoch der Wertebereich. An der Matchingkurve in Abbildung 5.15(b) erkennt man, dass dadurch die Lautstärkeabhängigkeit beseitigt ist. Alle acht gewünschten Treffer werden auch tatsächlich gefunden, wobei die einzelnen Peaks recht deutlich abgesetzt sind.

Dieses kleine Beispiel zeigt, dass die bei MFCC-Merkmalen eingesetzte Logarithmierung und die DCT auch bei Anwendung auf ein Pitchbinning zu sinnvollen Ergebnissen führen kann. Die recht deutlichen Treffer weisen darauf hin, dass weitere Untersuchungen der logPitch-Merkmale interessante und brauchbare Ergebnisse liefern könnten.



(a) rekonstruierte Merkmale der Anfrage $\mathcal{Q}^{Sho}$

(b) Matchingkurve bei Anfrage an $\mathcal{D}^{Sho}$

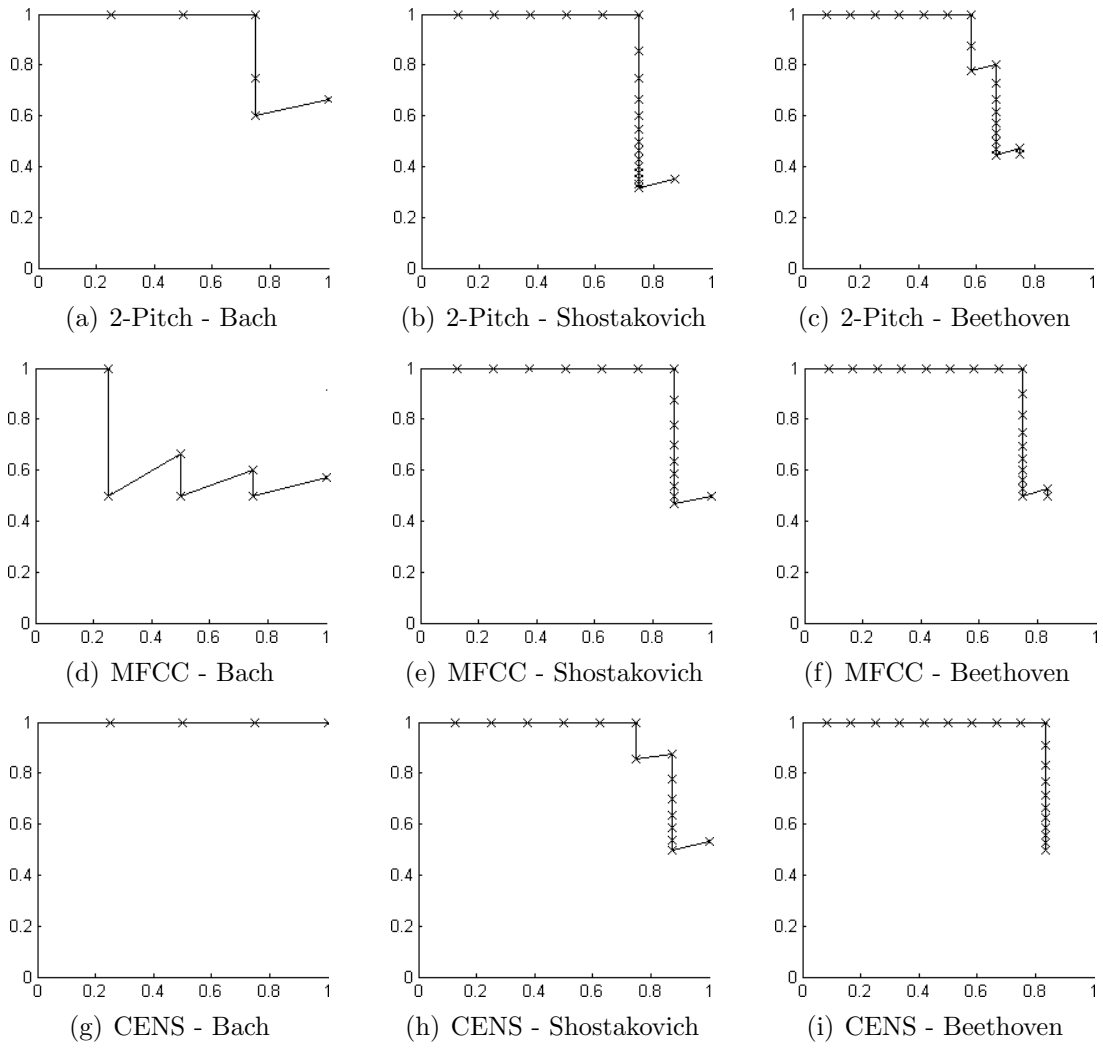
Abbildung 5.15. Die Matchingkurve der logPitch$^{88}_{88}$cut2-Merkmale weist acht deutliche korrekte Treffer auf.

5.7 Vergleich verschiedener Merkmale

Zum Abschluss sollen CENS-, MFCC$^{50}_{50}$cut2- und 2pitch-Merkmale auf einer größeren Testdatenbank verglichen werden. Zunächst werden Precision-Recall-Diagramme erstellt. Berechnung und Interpretation der Diagramme werden in Abschnitt 4.4.4 beschrieben. Die etwa 20-stündige Testdatenbank, siehe Anhang B, und die drei Anfragen ($\mathcal{Q}^{BachT}$, $\mathcal{Q}^{BeSin}$, $\mathcal{Q}^{Sho}$) sind ebenfalls die gleichen wie beim Test in Abschnitt 4.4.4. Die Trefferlisten zur Anfrage aus dem Stück von Shostakovich sind in Anhang C zu finden, Abbildung 5.16 zeigt die Precision-Recall-Diagramme. Bei den Anfragen $\mathcal{Q}^{BachT}$ und $\mathcal{Q}^{BeSin}$ zeigen die CENS-Merkmale deutlich die besten Ergebnisse, wobei jedoch die Klavierversion von Beethovens Fünfter Sinfonie verhindert, dass mit irgendeinem Merkmal alle gewünschten Passagen unter den ersten 20 Treffern auftauchen. Zur Anfrage $\mathcal{Q}^{Sho}$ werden von MFCC- und CENS-Merkmalen alle gewünschten Abschnitte gefunden, von den 2pitch-Merkmalen jedoch nicht. Das Diagramm der MFCC-Merkmale zeigt am Ende höhere Precisionwerte als das der CENS-Merkmale, sodass bei dieser Anfrage MFCC etwas vorne liegt. Bei der Anfrage $\mathcal{Q}^{BachT}$ stellt sich jedoch eine Schwäche der MFCC-Merkmale heraus, hier ist das Ergebnis deutlich schlechter als das der CENS- und 2pitch-Merkmale.

Insgesamt schneiden die CENS-Merkmale bei den gewählten drei Anfrage am besten ab. Die MFCC-Merkmale liefern bei zwei Anfragen gute Ergebnisse, zeigen bei der dritten Anfrage jedoch eine Schwachstelle. In einem nächsten Experiment soll eine größere Zahl von Anfragen zum Einsatz kommen, damit das Ergebnis nicht von der willkürlichen Auswahl der Anfragen abhängt.

Aus der Testdatenbank werden zufällig 30 Abschnitte mit Längen zwischen 16 und 21 Sekunden als Anfragen gewählt. Für jedes der drei Merkmale wird mit sämtlichen Anfragen ein Audiomatching auf der Datenbank durchgeführt. Die Trefferlisten werden auf fünf Einträge begrenzt und damit auch die Zahl der maximal möglichen gewünschten Treffer. Da die Anfragen zufällig gewählt sind, enthalten sie meist keine vollständigen markanten Themen, die innerhalb eines Stückes wiederholt werden, sodass die Anzahl der tatsächli-



(a) 2-Pitch - Bach (b) 2-Pitch - Shostakovich (c) 2-Pitch - Beethoven

(d) MFCC - Bach (e) MFCC - Shostakovich (f) MFCC - Beethoven

(g) CENS - Bach (h) CENS - Shostakovich (i) CENS - Beethoven

Abbildung 5.16. Precision Recall Diagramme für CENS-, MFCC- und 2pitch-Merkmale jeweils für die drei Anfragen $\mathcal{Q}^{BachT}$, $\mathcal{Q}^{BeSin}$ und $\mathcal{Q}^{Sho}$.

chen Vorkommen von ähnlichen Abschnitten nur selten die Fünf überschreitet. Durch die Begrenzung wird verhindert, dass durch Anfragen, die doch mehr als fünf Treffer liefern, einen größeren Einfluss auf das Gesamtergebnis haben als die anderen Anfragen. Zu jeder Anfrage wird die Zahl der ähnlichen Vorkommen in der Datenbank bestimmt (maximal fünf) und nach dem Audiomatching für jedes Merkmal die Anzahl der zusammenhängenden korrekten Treffer beginnend mit Platz eins der Trefferliste ermittelt. Stimmen bei einem Merkmal beispielsweise die Einträge eins, zwei und vier der Trefferliste, so wird dies als zwei korrekte Treffer gewertet.

Abbildung 5.17 zeigt die Ergebnistabelle. Es sind nur 23 Einträge enthalten, da Anfragen, die zu keinen weiteren ähnlichen Passagen in der Datenbank führen und damit bei jedem Merkmal zu genau einem Treffer führen würden, entfernt worden sind. Im Gesamtergebnis liegen die 2pitch-Merkmale deutlich zurück. Die Prozentwerte von CENS- und

Anfrage	Vorkommen	CENS	MFCC	2pitch
Bach_BWV565_toccataOrgan_Ruebsam_22050_mono_query_29_46	**4**	4	1	3
Beethoven_op002_no1_3_Sonate_1_Barenboim_22050_mono_query_117-137	**3**	2	3	3
Beethoven_op036_3_symphony_2_kegel_22050_mono_query_141-161	**4**	4	2	2
Beethoven_op055_3_symphony_3_blomstedt_22050_mono_query_7-27	**2**	2	2	2
Beethoven_op110_2_sonata_31_levine_22050_mono_query_31-50	**2**	2	2	2
Beethoven_op110_3_sonata_31_levine_22050_mono_query_126-144	**2**	2	2	2
Brahms_hungarian_dances_03_scholz_22050_mono_query_40-58	**3**	2	3	3
Brahms_hungarian_dances_06_scholz_22050_mono_query_2-22	**5**	5	5	5
Brahms_hungarian_dances_19_scholz_22050_mono_query_6-26	**2**	2	2	2
Burgmueller_op100_19_David_22050_mono_query_12-29	**3**	2	3	3
Dvorak_op095_1_symphony_9_francis_22050_mono_query_576-594	**3**	3	3	3
Dvorak_op095_1_symphony_9_maazel_22050_mono_query_372-389	**5**	5	5	5
Dvorak_op095_2_symphony_9_maazel_22050_mono_query_546-565	**5**	5	3	2
Dvorak_op095_3_symphony_9_adolph_22050_mono_query_198-217	**5**	5	5	5
Mendelssohn_op021_7_wedding_levine_22050_mono_query_69-90	**4**	2	4	2
Mozart_kv130_1_symphony_18_lizzio_22050_mono_query_58-78	**2**	2	2	2
Mozart_kv130_3_symphony_18_lizzio_22050_mono_query_45-63	**2**	2	2	2
Mozart_kv134_3_symphony_21_lizzio_22050_mono_query_115-132	**3**	3	3	3
Orff_carmina_burana_07_ormandy_22050_mono_query_35-53	**4**	4	4	4
Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono_query_0_19	**5**	5	5	5
Vivaldi_RV269_1_Spring_Carmirelli_22050_mono_query_86-106	**3**	2	3	1
Vivaldi_rv293_3_autumn_mae_22050_mono_query_36-53	**5**	3	5	2
Vivaldi_rv297_2_winter_mae_22050_mono_query_65-85	**5**	5	5	4
Gesamt:	**81**	73	74	67
		90%	**91%**	**83%**

Abbildung 5.17. Ergebnis des Tests mit mehreren Anfragen: Zu jeder Anfrage ist die Zahl der Vorkommen ähnlicher Abschnitte in der Datenbank angegeben (begrenzt auf 5) und die Zahl der durchgehenden korrekten Treffer für die drei untersuchten Merkmale.

MFCC-Merkmalen sind sich sehr ähnlich. Man sollte sich von dem einen Prozentpunkt, den die MFCC-Merkmale bei diesem Experiment mehr erreicht haben, jedoch nicht zu dem Trugschluss verleiten lassen, die MFCC-Merkmale seien besser geeignet. Denn die Precision-Recall-Diagramme haben bereits Schwächen von MFCC gegenüber CENS offengelegt. Beim genaueren Betrachten der Tabelle fällt auf, dass bei unterschiedlichen Anfragen mal CENS und mal MFCC bessere Resultate aufweisen. Zum Beispiel ist das Ergebnis der MFCC-Merkmale bei der Anfrage aus dem Stück „Toccata“ von Bach sehr schwach, wie bereits im vorherigen Experiment zu sehen. Auch bei anderen Anfragen zeigen sich solche Schwächen wie z.B. bei Beethovens Zweiter Sinfonie. Andererseits ist das Ergebnis der MFCC-Merkmale bei einigen Anfragen leicht besser, sodass sich insgesamt ein ausgeglichenes Bild ergibt. Die meisten dieser Anfragen, bei denen MFCC besser abschneidet, bestehen aus einem Teil, der nur einmal in einem Stück vorkommt und enthalten zusätzlich noch den Anfang oder das Ende eines Themas, welches im Stück wiederholt wird. Da die Anfragen zufällig ausgeschnitten werden, tritt dieses Phänomen in mehreren Anfragen auf, unter anderem z.B. bei der Anfrage aus dem Stück von Mendelssohn. Falls der Anteil des wiederholten Themas mehr als die Hälfte der Anfrage ausmacht, sollen alle Vorkommen des Themas als gewünschte Treffer gelten, obwohl die Anfrage einen Abschnitt enthält, der nicht an den wiederholten Positionen vorkommt. Diese Art der Trefferauswahl ist motiviert durch den Schluss von Beethovens Fünfter Sinfonie, der das Thema auch nur etwa

zur Hälfte enthält und danach abgewandelt ist, jedoch als Treffer erwartet wird.

Offenbar sind die MFCC-Merkmale robuster gegenüber solchen zusätzlich zum ähnlichen Teil vorliegenden Störstellen, die zwar aus dem gleichen Stück stammen und mit den selben Instrumenten gespielt sind, jedoch eine andere Melodie enthalten. Dies zeichnet sich auch dadurch ab, dass unter den falschen Treffern, die sich in den Tabellen der MFCC-Merkmale weit oben befinden, relativ häufig Abschnitte aus dem korrekten Stück anzutreffen sind. Die zufällige Wahl der Anfragen stellt auf diese Weise eine Stärke der MFCC-Merkmale heraus, die sich in etwa mit den Schwächen der Merkmale ausgleicht, sodass man ähnliche Ergebnisse wie bei den CENS-Merkmalen erhält. Betrachtet man dieses Ergebnis zusammen mit den Precision-Recall-Diagrammen, so sind die CENS-Merkmale etwas besser einzustufen als die MFCC-Merkmale. Deutlich schlechter schneiden die 2pitch-Merkmale ab.

Wenngleich man die hohe Trefferzahl im letzten Experiment nicht überbewerten sollte, zeigt sich doch, dass die MFCC-Merkmale ein hohes Potential in Bezug auf das Audiomatching haben und weiter untersucht werden sollten. Die am Ende von Abschnitt 5.5 vorgeschlagenen weiteren Modifikationen der Merkmale bieten sich als Grundlage zukünftiger Untersuchungen an, und Experimente mit anderen, größeren Testdatenbanken und anderen Verfahren zur Auswertung könnten interessante neue Ergebnisse liefern.

Anhang A

MATLAB-Code

In diesem Teil des Anhangs ist der Code der Programme aufgelistet, mit welchen die Merkmale, die in dieser Arbeit vorgestellt werden, extrahiert worden sind. Es handelt sich ausschließlich um MATLAB-Programme, siehe [9] für die MATLAB-Homepage. Jedes Programm erhält als Eingabe entweder einen PCM-Datenstrom oder eine zuvor extrahierte Merkmalsfolge, die weiterverarbeitet werden soll. Zusätzlich liegen einige Parameter vor, die die Merkmalsextraktion beeinflussen. Häufig wird auch das struct „sideinfo“ übergeben, das Metainformationen, wie z.B. die Namen und Verzeichnisse der Audiodateien oder die Abtastrate des PCM-Datenstroms enthält. Die Ausgabe aller gelisteten Funktionen ist eine Folge von Merkmalsvektoren, wobei es sich per Konvention immer um zweidimensionale Arrays handelt. Zu beachten ist, dass die Vektorkomponenten der einzelnen Merkmalsvektoren entlang der zweiten Dimension eingetragen sind und die zeitliche Abfolge der Vektoren parallel zur ersten verläuft. feature(2,3) gibt somit den dritten Eintrag des zweiten Merkmalsvektors wieder. Die Kommentare in den Programmköpfen enthalten alle Parameter der jeweiligen MATLAB-Funktion. Hinter den Parametern sind Defaultwerte angegeben, falls vorhanden.

Nach dem Code für die L_2-Normierungsfunktion werden die Funktionen in der Reihenfolge, mit welcher die entsprechenden Merkmale in der Arbeit vorgestellt werden, aufgelistet. In A.8 folgt noch ein beispielhafter Programmaufruf.

A.1 Normierung

Bevor ein Merkmal für das Audiomatching eingesetzt wird, werden alle Merkmalsvektoren im L_2-Sinne normiert, siehe dazu 2.2. Die folgende Funktion führt die Normierung für alle eingegebenen Merkmalsvektoren durch.

```
function feature = L2(feature, threshold)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: L2
% Date: 20.10.2007
% Programmer: Sebastian Kreuzer
%
```

```
% Input:
%         feature          Folge von Merkmalsvektoren
%
%         threshold = 0    Schwellwert: wenn Norm kleiner => Gleichverteilung
%
% Output:
%         feature          Folge von L2-normierten Merkmalsvektoren
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

unit = 1/sqrt(size(feature,2)); %Gleichverteilung entsprechender Länge

%normiere alle Merkmalsvektoren getrennt
for i=1:size(feature,1)
    n = norm(feature(i,:),2);
    if n > threshold
         feature(i,:) = feature(i,:)/n;
    else
         feature(i,:) = unit;
    end
end
```

A.2 Filterbank

Alle in Kapitel 3 vorgestellten Merkmale basieren auf einer Pitchzerlegung mittels Filterbank. Das folgende Programm führt die Filterung und die anschließende Berechnung der lokalen Energieverteilung durch. Zusätzlich können die Pitchmerkmale noch zu Chromamerkmalen weiterberechnet werden.

Eingabe sind, neben den PCM-Daten, gewünschte Fensterbreite und Überlappung, kleinster und größter berücksichtigter MIDI-Pitch und die Option, ob Pitch- oder Chromamerkmale generiert werden sollen. Das Programm kann bei einem Aufruf Merkmalsfolgen unterschiedlicher Auflösung erzeugen. Deshalb können als Fensterbreite und Überlappung Felder mit mehreren gewünschten Werten angegeben werden (Felder müssen gleiche Länge haben). Ausgabe ist ein „Cellarray“, das die Folgen von pitchSTMSP- bzw- Chromavektoren in den unterschiedlichen Auflösungen enthält.

Die Pitch- bzw. Chromavektoren werden nicht normiert, da sie auch zur Weiterberechnung zu anderen Merkmalen benutzt werden. Falls man ein Audiomatching direkt mit Pitch- oder Chromamerkmalen durchführen möchte, müssen die einzelnen Vektoren zuvor L_2-normiert werden.

```
function [f_pitch_energy_cell,sideinfo] =
                                audio_to_FBpitchSTMSP(f_audio,parameter,sideinfo)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: audio_to_STMSP
% Date: 12.12.2005
% Programmer: Meinard Mueller
% modified by Sebastian Kreuzer
%
% Input:
```

```
%           f_audio % PCM-Datenstrom
%
%           parameter
%               midi_min =  21 %kleinste berücksichtige MIDI-Note
%               midi_max = 108 %größte berücksichtige MIDI-Note
%               win_len = 4410  %Fensterbreite für STMSP
%               win_ov =  2205  %Überlappung für STMSP
%               chroma =     0  %1 - fasse zu Chromaklassen zusammen
%               fs =     22050  %Abtastrate von f_audio
%
%           benutzte Filterbank muss gespeichert sein als
%               MIDI_FB_ellip_pitch_60_96_22050_Q25.mat
%
% Output:
%           f_pitch_cell        %Folge von Merkmalsvektoren
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Vorbereitung
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Fensterlänge und Überlappung
win_len =  parameter.win_len;
win_ov  =  parameter.win_ov;

%lade Filterbank
load MIDI_FB_ellip_pitch_60_96_22050_Q25.mat

%genutzte Samplerate abhängig vom Pitch
fs_pitch = zeros(1,128);
fs_pitch(21:59) = 882;
fs_pitch(60:95) = 4410;
fs_pitch(96:120) = 22050;

fs_index = zeros(1,128);
fs_index(21:59) = 3;
fs_index(60:95) = 2;
fs_index(96:120) = 1;

%Anzahl der Samples im Signal
wavsize = size(f_audio,1);

%Stelle Signal in 3 Sampleraten zur Verfügung
pcm_ds = cell(3,1);
pcm_ds{1} = f_audio;
pcm_ds{2} = resample(pcm_ds{1},1,5,100);
pcm_ds{3} = resample(pcm_ds{2},1,5,100);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Berechne Fenstergrenzen (seg_pcm_start, seg_pcm_stop)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
num_window = length(win_len);
f_pitch_energy = cell(num_window,1);
seg_pcm_num = cell(num_window,1);
seg_pcm_start = cell(num_window,1);
seg_pcm_stop = cell(num_window,1);
for w=1:num_window;
    step_size = win_len(w)-win_ov(w);

    %zentriere erstes Fenster um erstes Sample
    group_delay = round(win_len(w)/2);
    seg_pcm_start{w} = [1 1:step_size:wavsize]';

    seg_pcm_stop{w} = min(seg_pcm_start{w}+win_len(w),wavsize);
    seg_pcm_stop{w}(1) = min(group_delay,wavsize);
    seg_pcm_num{w} = size(seg_pcm_start{w},1);
    f_pitch_energy{w} = zeros(seg_pcm_num{w},120);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Berechne lokale Energien in jedem Band
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for p=parameter.midi_min:parameter.midi_max
    index = fs_index(p);

    %Filterung
    f_filtfilt = filtfilt(h(p).b, h(p).a, pcm_ds{index});
    f_square = f_filtfilt.^2;

    %STMSP-Berechnung
    for w=1:length(win_len)
     %gleicht unterschiedliche Sampleraten aus
        factor = (parameter.fs/fs_pitch(p));
        for k=1:seg_pcm_num{w}
            start = ceil((seg_pcm_start{w}(k)/parameter.wav.fs)*fs_pitch(p));
            stop = floor((seg_pcm_stop{w}(k)/parameter.wav.fs)*fs_pitch(p));
            f_pitch_energy{w}(k,p)=sum(f_square(start:stop))*factor;
        end
    end
end
fprintf('\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% falls gewünscht, fasse zu Chromaklassen zusammen
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if parameter.chroma == 1
    for w=1:num_window;
        f_chroma_energy = zeros(seg_pcm_num{w},12);
        for p=parameter.midi_min:parameter.midi_max
            chroma = mod(p,12)+1;
            f_chroma_energy(:,chroma)
```

```
            = f_chroma_energy(:,chroma)+f_pitch_energy{w}(:,p);
        end
        f_pitch_energy{w} = f_chroma_energy;
    end
end

f_pitch_energy_cell = f_pitch_energy;
```

A.3 CENS

Das folgende Programm dient zur Berechnung der CENS-, bzw. FBpitchStat-Merkmale, siehe Abschnitt 3.4. Eingabe können Chromamerkmale sein, die dann zu CENS weiterverarbeitet werden, oder pitch-Merkmale, aus denen FBpitchStat-Merkmale erzeugt werden.

Der Parameter „stat_thresh“ stellt den Schwellwert der anfänglichen L_1-Norm dar. Über das Array „vec_energy“ werden die Quantisierungsstufen festgelegt und über das gleichlange Array „vec_weight“ die Werte, die den einzelnen Stufen zugewiesen werden. Die Parameter „stat_window_length_array“ und „stat_downsample_array“ geben Fensterbreite und Downsamplefaktor für die zeitliche Vergröberung an. Da das Programm verschiedene zeitliche Auflösungen in einem Arbeitsgang berechnen kann, können die beiden zuletzt genannten Parameter mit Arrays gleicher Länge gefüllt werden, die die verschiedenen Auflösungen enthalten.

```
function [f_CENS_cell,sideinfo] = STMSP_to_CENS(f_pitch,parameter,sideinfo)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: STMSP_to_CENS
% Date: 12.07.2007
% Programmer: Meinard Mueller, ported by Sebastian Ewert
% modified by Sebastian Kreuzer
%
% Input:
%         f_pitch   %pitch- oder Chroma-Merkmale
%
%         parameter
%             vec_energy = [40 20 10 5]/100 %Quantisierungsstufen
%             vec_weight =     [ 1 1 1 1]/4 %Quantisierungswerte
%             stat_thresh =           0.001 %Schwellwert für L1-Norm
%             stat_window_length_array = 41 %Breite des Stat.Fensters
%             stat_downsample_array =    10 %Downsmplfaktor. nach Fenster.
%             norm_threshold =         0.04 %Schwellwert für L2-Norm
%
% Output:
%         f_CENS_cell                        %Folge von Merkmalsvektoren
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

seg_num = size(f_pitch,1);
pitch_num = size(f_pitch,2);

%L1-Norm der Pitch- bzw. Chromavektoren
```

```
f_pitch_energy_distr = zeros(seg_num,pitch_num);
for k=1:seg_num
   if sum(f_pitch(k,:)>parameter.stat_thresh)>0
        seg_energy_square = sum(f_pitch(k,:));
        f_pitch_energy_distr(k,:) = ((f_pitch(k,:))/seg_energy_square);
    end
end

num_stat_window = length(parameter.stat_window_length_array);
f_CENS_cell = cell(num_stat_window,1);
for w=1:num_stat_window
    stat_window_length = parameter.stat_window_length_array(w);
    stat_downsample = parameter.stat_downsample_array(w);
    stat_window = hanning(stat_window_length);
    stat_window = stat_window/sum(stat_window);

    %Quantisierung der einzelnen Komponenten
    f_stat_help = zeros(seg_num,pitch_num);
    for n=1:length(parameter.vec_energy)
        f_stat_help = f_stat_help +
        (f_pitch_energy_distr>parameter.vec_energy(n))*parameter.vec_weight(n);
    end

    %zeitliche Vergröberung mittels Faltung mit Hannfenster
    f_pitch_energy_stat = upfirdn(f_stat_help,stat_window,1,stat_downsample);
    stat_num = ceil(seg_num/stat_downsample);
    cut = floor((stat_window_length-1)/(2*stat_downsample));
    f_pitch_energy_stat = f_pitch_energy_stat((1+cut:stat_num+cut),:);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % L2-Normierung
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    f_CENS_cell{w} = L2(f_pitch_energy_stat,parameter.norm_threshold);
end
```

A.4 Short Time Fourier Transform

Dieses Programm dient dazu, Spektralvektoren mit verschiedenen Fensterbreiten und Überlappungen zu berechnen. Eingabe sind ein PCM-Datenstrom und Felder mit den gewünschten Fensterbreiten und Überlappungen (Felder müssen gleiche Länge haben). Ausgabe ist ein „Cellarray“, das die Folgen von Spektralvektoren in den unterschiedlichen Auflösungen enthält. Um das erste Fenster auf dem ersten Sample der Audioeingabe zu zentrieren und weil der MATLAB-interne Befehl „spectrogram“ unnötig viel Speicher verbraucht, wird eine eigene Routine zur Fensterung benutzt, die auf den MATLAB Befehl „fft“ zurückgreift.

Die Spektralvektoren werden nicht normiert, da sie für die Weiterberechnung zu anderen Merkmalen bestimmt sind. Falls man ein Audiomatching direkt mit Spektralvektoren durchführen möchte, müssen die einzelnen Vektoren zuvor L_2-normiert werden.

```
function [f_stft_cell,sideinfo] = audio_to_STFT(f_audio,parameter,sideinfo)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: audio_to_STFT
% Date: 01.06.2007
% Programmer: Sebastian Kreuzer
%
% berechnet STFT-Merkmale (short-time fourier transform)
% es können mehrere win_len/win_ov Werte angegeben werden
% => es wird ein Cellarray mit Merkmalsfolgen für jede Auflösung erzeugt
%
% Input:
%            f_audio                  %PCM-Datenstrom
%
%            parameter
%               win_len =    4096     %Array mit Fensterbreiten in Samples
%               win_ov =     1891     %Array mit Überlappungen der Fenster
%               window = @hamming     %Fensterfunktion
%               fs =        22050     %Abtastrate von f_audio
%
% Output:
%            f_stft_cell              %Folge von Merkmalsvektoren
%            sideinfo.STFT.win_res
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

win_len = parameter.win_len;
win_ov = parameter.win_ov;
sideinfo.STFT.win_res = parameter.fs./(win_len-win_ov);

num_window = length(win_len);       %Anzahl der gewünschten Auflösungen
f_stft_cell = cell(num_window,1);
wav_size = length(f_audio);         %Länge des PCM-Datenstroms

for w=1:num_window

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Vorberechnung einiger benötigter Werte
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    win = window(parameter.window,win_len(w));
    num_coeffs = floor(win_len(w))/2+1;
    first_win = floor(win_len(w)/2);
    step_size = win_len(w)-win_ov(w);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Berechnung des Spektrogramms
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %Speicher für Spektrogramm reservieren
    num_frames = ceil((wav_size-win_len(w)+first_win)/step_size);
    f_stft_cell{w} = zeros(num_frames,num_coeffs);

    %zentriere erstes Fenster (frame) auf dem Beginn der Audiodaten
```

```
    frame = (1:win_len)-first_win;
    x = [zeros(first_win,1); f_audio(1:(win_len(w)-first_win))].*win;

    i = 1;
    while frame(end) < wav_size
        ft = abs(fft(x,win_len(w)))';   %Amplitudenspektrum
        f_stft_cell{w}(i,:) = ft(1:num_coeffs);
        frame = frame+step_size;
        if frame(end) > wav_size
           frame = frame(1):wav_size;
           win = win(1:length(frame));
        end
        x = f_audio(frame).*win;
        i = i+1;
    end

    %berechne letztes (unvollständiges) Fenster
    ft = abs(fft(x,win_len(w)))';   %Amplitudenspektrum
    f_stft_cell{w}(i,:) = ft(1:num_coeffs);

end
```

A.5 Approximation von Filterbänken und verschiedene Binnings

Das folgende Programm ist eingesetzt worden, um die in Abschnitt 4.3 beschriebene Pitchzerlegung basierend auf STFT durchzuführen. Wegen des allgemeinen Aufbaus können aber auch alle in 4.4 vorgestellten Binnings damit simuliert werden, wie z.B. eine Oktavzerlegung. Das Prinzip beruht auf einer Fensterung der einzelnen Spektralvektoren, siehe 4.4 für eine genaue Erklärung. Start- und Endfrequenz des ersten Fensters werden festgelegt und mit Hilfe eines Vergrößerungsfaktors eine exponentielle Zunahme der Fenstergröße eingestellt. Auch eine Überlappung anteilig zur Fenstergröße ist möglich. Zusätzlich kann die Fensterfunktion eingestellt werden.

```
function [f_spectralwindow,sideinfo] =
 STFT_to_spectralwindowL2(f_stft,parameter,sideinfo)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: STFT_to_spectralwindowL2
% Date: 15.07.2007
% Programmer: Sebastian Kreuzer
%
% Input:
%         stft                            %Folge von Spektralvektoren
%
%         parameter
%             window =          @rectwin %Form des spektralen Fensters
%             win_increase =           1 %Zunahme der Fenstergröße
%             overlap_factor              %Überlappung anteilig zur Breite
%             first_win_start_pitch = 21 %untere Grenze des ersten Fensters
```

```
%             first_win_end_pitch =   32 %obere Grenze des ersten Fensters
%             max_pitch =            116 %obere Grenze des letzten Fensters
%             norm_threshold =      0.04 %Schwellwert für L2-Norm
%             fs =                 22050 %Abtastrate der Ausgangs-PCM-Daten
%
% Output:
%         f_spectralwindow               %Folge von Merkmalsvektoren
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Vorberechnung einiger benötigter Werte
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nyquist = floor(parameter.fs/2); %Nyquistfrequenz des Eingangssignals in Hz
num_coeffs = size(f_stft,2);        %Anzahl der Fourierkoeffizienten
freq_step = nyquist/(num_coeffs-1); %Frequenzschritt pro Koeffizient

%untere Grenze des ersten Fensters in Hz
start_freq = 2^((parameter.first_win_start_pitch-69.5)/12)*440;

%Länge des ersten Fensters in Hz
len = 2^((parameter.first_win_end_pitch-68.5)/12)*440 - start_freq;

%obere Grenze des letzten Fensters in Hz
max_freq = min(nyquist, 2^((parameter.max_pitch-68.5)/12)*440);

%Faktor mit dem die Länge eines Fensters multipliziert wird, um die Länge
%des nächsten Fensters zu berechnen
factor = 2^parameter.win_increase;

%wenn "overlap_factor" nicht gegeben: berechne Überlappung so, dass jedes
%Fenster gerade das übernächste berührt
if ~isfield(parameter,'overlap_factor')
    parameter.overlap_factor = factor/(factor+1);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Berechnung der Anfangs- und Endkoeffizienten der Fenster
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%"freq_range" <- Anfangs- und Endfrequenzen der Fenster in Hz
freq_range(1,:) = [start_freq start_freq+len]; %1. Fenster
i = 1;
while freq_range(i,2) < max_freq
    i = i+1;
    %Anfangsfrequenz = Ende des letzten Fensters minus Überlappung
    freq_range(i,1) = freq_range(i-1,2) - len*parameter.overlap_factor;
    len = len*factor;                           %neue Fensterlänge
    freq_range(i,2) = freq_range(i,1) + len;   %Endfrequenz
```

```
end
freq_range(i,2) = max_freq;                    %ggf. abschneiden

num_windows = size(freq_range,1);              %Anzahl der Fenster

%"coeff_borders" <- Anfangs- und Endkoeffizienten der Fenster
coeff_borders(:,1) = min(ceil(freq_range(:,1)/freq_step+1), num_coeffs+1);
coeff_borders(:,2) = min(floor(freq_range(:,2)/freq_step+1), num_coeffs);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Quadratsummierung der Koeffizienten innerhalb der Fenster
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

f_stft = f_stft.^2; %Summiere Quadrate des Signals

f_spectralwindow = zeros(size(f_stft,1),num_windows);
for w=1:num_windows
    n = coeff_borders(w,2) - coeff_borders(w,1) + 1;
    if n > 0
        %Matrix-Multiplikation mit dem Fenster (als Vektor) realisiert Summierung
        f_spectralwindow(:,w) =
         f_stft(:,coeff_borders(w,1):coeff_borders(w,2))*window(parameter.window,n);
    end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% L2-Normierung
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f_spectralwindow = L2(f_spectralwindow, parameter.norm_threshold);
```

A.6 MFCC-Merkmale

Die MFCC-Merkmale aus Kapitel 5 werden mit folgendem Programm berechnet, welches die „MA Toolbox“ für MATLAB benutzt. Eingabe sind neben dem PCM-Datenstrom die Parameter „win_len“ und „win_ov“, die die Fensterbreite und Überlappung der zuerst durchgeführten Spektrogrammberechnung festsetzen. Die Mel-Filterbank wird über die drei Werte in „mel_filt_bank“ festgelegt, wobei der erste Wert die unterste berücksichtigte Frequenz angibt, der zweite die oberste Frequenz und der dritte Wert die Anzahl der Filter. Mit „num_coeffs“ wählt man die Zahl der Koeffizienten, die nach der DCT erhalten bleiben und schließlich mit „cut_coeffs“ die Anzahl der Koeffizienten, die vom niedrigsten beginnend abgeschnitten werden.

```
function [f_mfcc_cell,sideinfo] = audio_to_MFCCL2(f_audio,parameter,sideinfo)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: audio_to_MFCCL2
% Date: 16.05.2007
```

```
% Programmer: Sebastian Kreuzer
%
% Input:
%        f_audio                          %PCM-Datenstrom
%
%        parameter
%            win_len =             4096  %STFT Fensterbreite
%            win_ov =              1891  %STFT Überlappung der Fenster
%            mel_filt_bank = [20 16000 40])  %Mel-Filterbank
%            num_coeffs =            20  %Zahl der DCT Koeffizienten
%            cut_coeffs =   0   %Zahl abzuschneidender niedriger Koeffizienten
%            norm_threshold =      0.04  %Schwellwert für L2-Norm
%            fs =                 22050  %Abtastrate von f_audio
%
% Output:
%        f_mfcc_cell                      %Folge von Merkmalsvektoren
%
%        sideinfo.MFCCL2.win_res
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

win_len = parameter.win_len;
win_ov  = parameter.win_ov;
win_res = parameter.fs./(win_len-win_ov);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setzen der Parameter für MA Toolbox
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p.fs              = parameter.fs;
p.num_ceps_coeffs = parameter.num_coeffs;
p.mel_filt_bank   = parameter.mel_filt_bank;
if parameter.cut_coeffs > 0
    p.use_first_coeff = 0;
else
    parameter.cut_coeffs = 1;
    p.use_first_coeff = 1;
end

num_window = length(win_len);
f_mfcc_cell = cell(num_window,1);

for w=1:num_window
    %Passe aktuelle Fensterlänge und Überlappung für Toolbox an
    p.fft_size        = win_len(w);
    p.hopsize         = win_len(w)-win_ov(w);

    %Berechnung der Merkmale mit der Funktion der Toolbox
    [f_mfcc_cell{w}]  = ma_mfcc(f_audio, p)';

    %Abschneiden von niedrigen Koeffizienten entsprechend "cut_coeffs"
    f_mfcc_cell{w}    = f_mfcc_cell{w}(:,parameter.cut_coeffs:end);
```

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % L2 Normierung
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    f_mfcc_cell{w} = L2(f_mfcc_cell{w}, parameter.norm_threshold);
end

sideinfo.MFCCL2.win_res = win_res;
```

A.7 Amplitudenmerkmale

Mit diesem Programm werden die in Abschnitt 5.4 vorgestellten Amplitudenmerkmale berechnet. Die Parameter „win_len“ und „win_ov“ bestimmen Fensterbreite und Überlappung bei der STMSP-Berechnung. Mit „max_factor“ kann der Maximalwert zur Komplementbildung heruntergesetzt werden.

```
function [f_amplitudeSTMSPL2,sideinfo] =
                        audio_to_amplitudeSTMSPL2(f_audio,parameter,sideinfo)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: audio_to_amplitudeSTMSPL2
% Date: 01.09.2007
% Programmer: Sebastian Kreuzer
%
% Input:
%        f_audio                          %PCM-Datenstrom
%
%        parameter
%           win_len =              4410  %Fensterbreite
%           win_ov =               2205  %Überlappung
%           max_factor =              1  %Verkleinerung des Maximalwertes
%           norm_threshold =          0  %Schwellwert für L2-Norm
%           fs =                  22050  %Abtastrate von f_audio
%
% Output:
%        f_amplitudeSTMSPL2               %Folge von Merkmalsvektoren
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Fensterlänge und Fensterfunktion (Rechteck)
win_len = parameter.win_len;
win = window(@rectwin,win_len);

downsample = win_len-parameter.win_ov;

f_square = f_audio.^2; %Quadrate sollen summiert werden

%Fensterung & Summe, realisiert mittels Faltung mit Fenster & Downsampling
amp = upfirdn(f_square,win,1,downsample);

amp = log10(amp+1); %Bestimme log der Energie; +1 => keine negativen Werte
```

```
%Maximalwert zur Komplementbildung
maxval = log10(win_len+1)*parameter.max_factor;

%Merkmalsvektoren <- Energie & Komplement
f_amplitudeSTMSPL2 = [amp maxval-amp];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% L2-Normierung
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
f_amplitudeSTMSPL2 = L2(f_amplitudeSTMSPL2, parameter.norm_threshold);
```

A.8 Beispiel für einen Programmaufruf

Am Beispiel der Funktionen „audio_to_STFT“ und „STFT_to_spectralwindowL2“ wird ein typischer Programmaufruf zur Merkmalsextraktion demonstriert. Es werden Pitchmerkmale berechnet und die Merkmalsfolge geplottet.

```
audiofile = 'Bach_BWV565_toccataOrgan_Koopman_22050_mono.wav';

%Einlesen der Audiodaten
[f_audio,fs] = wavread(audiofile);

sideinfo = [];                  %Feld für Zusatzinformationen

%parameter.win_len =    4096    %nutze Defaultwerte des Programms
%parameter.win_ov =     1891    %d.h. Parameter werden nicht angegeben
%parameter.window = @hamming

parameter.fs = fs;              %setze Abtastrate des PCM-Datenstroms

%Erzeugung der Spektralvektoren
[f_stft_cell,sideinfo] = audio_to_STFT(f_audio,parameter,sideinfo);
f_stft = f_stft_cell{1};

%parameter.window =        @rectwin %nutze Defaultwerte des Programms
%parameter.norm_threshold =    0.04

parameter.win_increase =        1/12;
parameter.overlap_factor =         0;
parameter.first_win_start_pitch = 21;
parameter.first_win_end_pitch =   21;
parameter.max_pitch =            108;

%Erzeugung der Pitchmerkmale aus den Spektralvektoren
[f_pitch,sideinfo] = STFT_to_spectralwindowL2(f_stft,parameter,sideinfo);

imagesc(f_pitch');
```

Anhang B

Testdatenbank

In der folgenden Tabelle sind die Musikstücke der Testdatenbank aufgelistet, auf welcher die Experimente aus den Abschnitten 4.4.4 und 5.7 durchgeführt worden sind. Die Tabelle enthält die Namen der Stücke oder, falls sämtliche Stücke aus einem Ordner enthalten sind, die Namen der Ordner. Zur Unterscheidung enden Ordnernamen mit „/“, zusätzlich ist die Anzahl der Stücke pro Ordner angegeben.

Einzelstücke / Ordner	#
Bach\Bach_BWV565_22050_mono/	7
Bach\Bach_BWV787_801_Sinfonia_Schiff_22050_mono/	15
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op021_1_symphony_1_blomstedt_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op021_2_symphony_1_blomstedt_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op021_3_symphony_1_blomstedt_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op021_4_symphony_1_blomstedt_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op036_1_symphony_2_kegel_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op036_1_symphony_2_liszt_piano_scherbakov_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op036_2_symphony_2_kegel_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op036_2_symphony_2_liszt_piano_scherbakov_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op036_3_symphony_2_kegel_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op036_3_symphony_2_liszt_piano_scherbakov_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op036_4_symphony_2_kegel_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op036_4_symphony_2_liszt_piano_scherbakov_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op055_1_symphony_3_blomstedt_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op055_2_symphony_3_blomstedt_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op055_3_symphony_3_blomstedt_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op067_1_symphony_5_karajan_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_misc_22050_mono/Beethoven_op067_1_symphony_5_liszt_piano_scherbakov_22050_mono.wav	1
Beethoven\Beethoven_Symphonies_Bernstein_22050_mono/Beethoven_op067_1_symphony_5_bernstein_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_Barenboim_22050_mono/Beethoven_op002_no1_1_Sonate_1_Barenboim_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_Barenboim_22050_mono/Beethoven_op002_no1_2_Sonate_1_Barenboim_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_Barenboim_22050_mono/Beethoven_op002_no1_3_Sonate_1_Barenboim_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_Barenboim_22050_mono/Beethoven_op002_no1_4_Sonate_1_Barenboim_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_Barenboim_22050_mono/Beethoven_op002_no2_1_Sonate_2_Barenboim_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_Barenboim_22050_mono/Beethoven_op002_no2_2_Sonate_2_Barenboim_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_Barenboim_22050_mono/Beethoven_op002_no2_3_Sonate_2_Barenboim_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_Barenboim_22050_mono/Beethoven_op002_no2_4_Sonate_2_Barenboim_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_Barenboim_22050_mono/Beethoven_op031_no2_1_Sonate_17_Sturm_Barenboim_22050_mono.wav	1
Beethoven\Beethoven_Sonaten_misc_22050_mono/	24
brahms\Brahms_Hungarian_Dances_Scholz_22050_mono/	21
brahms\Brahms_Hungarian_Dances_misc_22050_mono/	2

Abbildung B.1. Testdatenbank Teil 1

Burgmueller\Burgmueller_op100_David_22050_mono/	25
Chopin\Chopin_op25_Etueden_Varsi_22050_mono/	12
Dvorak/	12
Grieg/	3
Mendelssohn/	8
Mozart\Mozart_serenades_misc/	6
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv128_1_symphony_16_kantschieder_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv128_2_symphony_16_kantschieder_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv128_3_symphony_16_kantschieder_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv130_1_symphony_18_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv130_2_symphony_18_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv130_3_symphony_18_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv130_4_symphony_18_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv134_1_symphony_21_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv134_2_symphony_21_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv134_3_symphony_21_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv134_4_symphony_21_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv162_1_symphony_22_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv162_2_symphony_22_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv162_3_symphony_22_lizzio_22050_mono.wav	1
Mozart\Mozart_symphonies_misc_22050_mono/Mozart_kv182_1_symphony_24_kantschieder_22050_mono.wav	1
orff/Orff_carmina_burana_01_jochum_22050_mono.wav	1
orff/Orff_carmina_burana_01_ormandy_22050_mono.wav	1
orff/Orff_carmina_burana_02_jochum_22050_mono.wav	1
orff/Orff_carmina_burana_02_ormandy_22050_mono.wav	1
orff/Orff_carmina_burana_03_jochum_22050_mono.wav	1
orff/Orff_carmina_burana_03_ormandy_22050_mono.wav	1
orff/Orff_carmina_burana_04_jochum_22050_mono.wav	1
orff/Orff_carmina_burana_04_ormandy_22050_mono.wav	1
orff/Orff_carmina_burana_05_jochum_22050_mono.wav	1
orff/Orff_carmina_burana_05_ormandy_22050_mono.wav	1
orff/Orff_carmina_burana_06_jochum_22050_mono.wav	1
orff/Orff_carmina_burana_06_ormandy_22050_mono.wav	1
orff/Orff_carmina_burana_07_jochum_22050_mono.wav	1
orff/Orff_carmina_burana_07_ormandy_22050_mono.wav	1
orff/Orff_carmina_burana_08_jochum_22050_mono.wav	1
ravel/	2
shostakovich/	2
vivaldi\Vivaldi_Four_Seasons_Carmirelli_22050_mono/	12
vivaldi\Vivaldi_Four_Seasons_Abbado_22050_mono/	12
vivaldi\Vivaldi_Four_Seasons_Mae_20050_Mono/	12
wagner/	3
Gesamtzahl:	**235**

Abbildung B.2. Testdatenbank Teil 2

Anhang C

Trefferlisten

In diesem Teil des Anhangs sind die Trefferlisten der Testszenarien aus Abschnitt 5.7 für die Anfrage aus dem Stück von Shostakovich abgebildet. Aus diesen Listen sind die Precision-Recall-Diagramme erzeugt worden. Die Tabellen können vom Audiomatchingprogramm automatisch im HTML-Format generiert werden. Das Programm erhält als Eingabe die Stücke der Datenbank und die Anfrage. Es werden automatisch entweder die vorhandenen Merkmale zu den Musikstücken geladen oder, falls sie noch nicht existieren, neu berechnet und für die Zukunft abgespeichert. Danach wird der in Abschnitt 2.1 beschriebene DTW-Algorithmus ausgeführt und die Matchingkurve erzeugt. Nun wird nach der in 2.3 vorgestellten Methode die Kurve ausgewertet, d.h. es werden lokale Minima bestimmt (die Anzahl und evtl. eine obere Kostenschranke sind vorgegeben) und aus den Positionen der Minima in der Matchingkurve die Positionen der Treffer innerhalb der WAV-Dateien berechnet. Schließlich werden die gefundenen Audioabschnitte in eigene kleine WAV-Dateien kopiert und in der erzeugten Tabelle verlinkt, damit man beim Betrachten der Tabelle jederzeit die als Treffer identifizierten Passagen anhören kann.

Jede Tabelle enthält in den Spalten den Rang des Treffers innerhalb der Liste, den Wert der Matchingkurve am Peak, der als Treffer erkannt worden ist (also die Kosten), den Namen der WAV-Datei, in dem der Treffer vorliegt und die Position innerhalb der Datei in Sekunden. Beim Namen handelt es sich jeweils um den erwähnten Link zum herauskopierten Audioabschnitt.

Rank	*Distance (Delta min)*	*File name*	*Match position (in seconds)*
1	0.024748	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	1 - 19
2	0.063581	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	35 - 53
3	0.102452	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	156 - 174
4	0.120847	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	4 - 22
5	0.125163	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	43 - 61
6	0.155863	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	180 - 198
7	0.191596	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op055_2_symphony_3_blomstedt_22050_mono.wav	639 - 657
8	0.205005	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	141 - 159
9	0.209147	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op055_2_symphony_3_blomstedt_22050_mono.wav	470 - 488
10	0.212671	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op067_1_symphony_5_liszt_piano_scherbakov_22050_mono.wav	94 - 112
11	0.213129	vivaldi\Vivaldi_Four_Seasons_Mae_20050_Mono\Vivaldi_rv297_3_winter_mae_22050_mono.wav	87 - 105
12	0.216258	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op036_2_symphony_2_liszt_piano_scherbakov_22050_mono.wav	296 - 314
13	0.223744	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op067_1_symphony_5_liszt_piano_scherbakov_22050_mono.wav	11 - 29
14	0.224873	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op055_2_symphony_3_blomstedt_22050_mono.wav	875 - 893
15	0.229868	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	123 - 141
16	0.231649	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op067_1_symphony_5_liszt_piano_scherbakov_22050_mono.wav	253 - 271
17	0.234658	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op067_1_symphony_5_liszt_piano_scherbakov_22050_mono.wav	399 - 417
18	0.236641	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op067_1_symphony_5_liszt_piano_scherbakov_22050_mono.wav	423 - 441
19	0.236849	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op055_2_symphony_3_blomstedt_22050_mono.wav	36 - 54
20	0.240967	Beethoven\Beethoven_Symphonies_misc_22050_mono\Beethoven_op055_2_symphony_3_blomstedt_22050_mono.wav	519 - 537

Abbildung C.1. Trefferlisten zur Anfrage „Shostakovich“ - CENS-Merkmale

Rank	*Distance (Delta min)*	*File name*	*Match position (in seconds)*
1	0.033313	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	0 - 18
2	0.311710	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	3 - 21
3	0.354702	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	33 - 51
4	0.419441	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	42 - 60
5	0.467686	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	155 - 173
6	0.484983	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	179 - 197
7	0.540805	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	124 - 142
8	0.573784	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op031_no2_2_Sonate_17_Sturm_Pollini_22050_mono.wav	159 - 176
9	0.588260	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op110_3_sonata_31_boehm_pollini_22050_mono.wav	196 - 214
10	0.594544	Mozart\Mozart_symphonies_misc_22050_mono\Mozart_kv162_1_symphony_22_lizzio_22050_mono.wav	146 - 164
11	0.594933	Dvorak\Dvorak_op095_3_symphony_9_adolph_22050_mono.wav	419 - 437
12	0.595023	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op106_3_sonata_29_shoji_22050_mono.wav	41 - 59
13	0.598161	ravel\Ravel_bolero_abbado_22050_mono.wav	201 - 219
14	0.598676	Dvorak\Dvorak_op095_4_symphony_9_adolph_22050_mono.wav	597 - 615
15	0.601370	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op031_no2_2_Sonate_17_Sturm_Gilels_22050_mono.wav	185 - 203
16	0.604300	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	144 - 162
17	0.605375	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op106_3_sonata_29_shoji_22050_mono.wav	21 - 38
18	0.606928	Mendelssohn\Mendelssohn_op064_2_violinconcerto_e_mutter_karajan_22050_mono.wav	547 - 565
19	0.608726	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op110_3_sonata_31_levine_22050_mono.wav	535 - 553
20	0.611794	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op110_1_sonata_31_levine_22050_mono.wav	194 - 212

Abbildung C.2. Trefferlisten zur Anfrage „Shostakovich“ - MFCC-Merkmale

Rank	*Distance (Delta min)*	*File name*	*Match position (in seconds)*
1	0.010592	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	0 - 18
2	0.217442	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	34 - 52
3	0.221638	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	42 - 60
4	0.245713	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	3 - 21
5	0.340471	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	179 - 197
6	0.348743	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Yablonsky_22050_mono.wav	155 - 173
7	0.382720	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op110_3_sonata_31_levine_22050_mono.wav	265 - 283
8	0.398324	Chopin\Chopin_op25_Etueden_Varsi_22050_mono\Chopin_op25_12_Etueden_Varsi_22050_mono.wav	43 - 61
9	0.415565	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op110_3_sonata_31_boehm_pollini_22050_mono.wav	199 - 217
10	0.422589	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op110_3_sonata_31_levine_22050_mono.wav	223 - 241
11	0.429870	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op110_3_sonata_31_boehm_pollini_22050_mono.wav	236 - 254
12	0.433031	Chopin\Chopin_op25_Etueden_Varsi_22050_mono\Chopin_op25_12_Etueden_Varsi_22050_mono.wav	0 - 18
13	0.435438	Burgmueller\Burgmueller_op100_David_22050_mono\Burgmueller_op100_09_David_22050_mono.wav	1 - 19
14	0.437464	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op106_3_sonata_29_zechlin_22050_mono.wav	31 - 49
15	0.450874	vivaldi\Vivaldi_Four_Seasons_Carmirelli_22050_mono\Vivaldi_RV293_1_Autumn_Carmirelli_22050_mono.wav	275 - 293
16	0.452800	Beethoven\Beethoven_Sonaten_misc_22050_mono\Beethoven_op110_3_sonata_31_levine_22050_mono.wav	541 - 559
17	0.453364	Mozart\Mozart_serenades_misc\Mozart_kv525_2_serenade_13_pitamic_22050_mono.wav	248 - 266
18	0.455720	Mendelssohn\Mendelssohn_op064_3_violinconcerto_e_mintz_abbado_22050_mono.wav	252 - 270
19	0.455997	Chopin\Chopin_op25_Etueden_Varsi_22050_mono\Chopin_op25_12_Etueden_Varsi_22050_mono.wav	72 - 90
20	0.456192	shostakovich\Shostakovich_JazzSuite2_6_Waltz2_Tchibo_22050_mono.wav	144 - 162

Abbildung C.3. Trefferlisten zur Anfrage „Shostakovich“ - 2pitch-Zerlegung

Danksagung

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

- Der gesamten Arbeitsgruppe von *Prof. Dr. Michael Clausen* für die wunderbare Betreuung. Insbesondere bedanke ich mich bei meinem Betreuer *Dr. Meinard Müller* für die regelmäßigen, sehr aufschlussreichen Treffen während meiner Einarbeitungsphase und die sehr gute Betreuung auch nach seinem Wechsel zum MPII Saarbrücken.
- *Dominik Bors*, *Johannes Hagen* und *Jochen Welle* für gründliches Korrekturlesen und viele hilfreiche Vorschläge zur Verbesserung von Sprache und Formatierung.
- *Daniel Wolff* für umfangreiche und brauchbare Literaturvorschläge.

Erklärung

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

Bonn, den 11. Februar 2008

Sebastian Kreuzer

Literaturverzeichnis

[1] M. A. Bartsch and G. H. Wakefield, *Audio thumbnailing of popular music using chroma-based representations*, IEEE Transactions on Multimedia, 7 (2005), pp. 96–104.

[2] A. E. Çetin, Ömer N. Gerek, and Y. Yardimci, *Equiripple FIR filter design by the FFT algorithm*, IEEE Signal Processing Magazine, (1997).

[3] M. Clausen and M. Müller, *Inhaltsbasiertes Multimediaretrieval*, 2007.

[4] J. W. Cooley and J. W. Tukey, *An algorithm for the machine computation of the complex fourier serie*, Mathematics of Computation, 19 (1965), pp. 297–301.

[5] M. Goto, *A chorus-section detecting method for musical audio signals*, 2003.

[6] I. T. Jolliffe, *Principal Component Analysis*, Springer, 2002.

[7] B. Logan, *Mel frequency cepstral coefficients for music modeling*, in Proceedings of the First International Symposium on Music Information Retrieval (ISMIR), Plymouth, Massachusetts, oct 2000.

[8] Marhav and C.-H. Lee, *On the asymptotic statistical behavior of empirical ceptsral coefficients*, IEEE Transactions on Signal Processing, 41 (1993).

[9] Matlab, *The MathWorks Deutschland*. `http://www.mathworks.de`, 2006.

[10] M. Müller, *Information Retrieval for Music and Motion*, Springer, 2007.

[11] M. Müller, F. Kurth, and M. Clausen, *Audio matching via chroma-based statistical features*, in ISMIR, 2005, pp. 288–295.

[12] E. Pampalk, *MA Toolbox for Matlab*. `http://www.ofai.at/~elias.pampalk/ma/index.html`, 2003-2006.

[13] K. R. Rao and P. Yip, *Discrete cosine transform: algorithms, advantages, applications*, Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[14] H. Terasawa, M. Slaney, and J. Berger, *The thirteen colors of timbre*, in Applications of Signal Processing to Audio and Acoustics, 2005. IEEE Workshop on, 2005, pp. 323– 326.

[15] P. Vary, U. Heute, and W. Hess, *Digitale Sprachsignalverarbeitung*, Teubner B.G. GmbH, 1998.

[16] J. Volkmann and S. S. Stevens, *A scale for the measurement of the psychological magnitude pitch*, The Journal of the Acoustical Society of America, 8 (1937), p. 208.

[17] ZeM College - Institut für Elektronische Musik, *MIDI Kompendium.* `http://www.zem-college.de/midi/index.htm`.