

Diplomarbeit

**Tempoerkennung aus Audiosignalen
langsamer Musikstücke**

**Thorsten Deinert
17. Oktober 2010**

Betreuer:
Prof. Dr. Günter Rudolph
Dipl.-Inform. Igor Vatolkin

Fakultät für Informatik
Algorithm Engineering (Ls11)
Technische Universität Dortmund
<http://ls11-www.cs.tu-dortmund.de>

Inhaltsverzeichnis

1 Einführung	1
1.1 Aufbau der Arbeit	2
1.2 Musiktheoretische Grundlagen	2
1.2.1 Akzente	3
1.2.2 Metrische Hierarchie	4
1.3 Aufgaben der Tempoerkennung	6
1.4 Ziel der Arbeit	7
I Grundlagen der Tempoerkennung	9
2 Audio, Signalverarbeitung und Onset-Erkennung	11
2.1 Audiosignale	12
2.2 Zeit, Energie und Spektrum	14
2.2.1 Fouriertransformation	14
2.2.2 Spektrogramm	15
2.3 Onset-Erkennung	17
2.3.1 Transiente Onsets	18
3 Musikalische Akzentanalyse	21
3.1 Akzentanalyse auf einzelnen Frequenzbändern	22
3.2 Akzentanalyse auf Chromamerkmalen	25
3.2.1 Chroma	25
3.2.2 Einfacher Chromaakzent	26
3.2.3 Chromaanalyse durch Grundfrequenzschätzung	27
3.2.4 Grundfrequenzbasierter Chromaakzent	30
4 Verfahren zur Tempoerkennung	35
4.1 Regelbasierte Verfahren	36
4.2 Autokorrelation	36
4.2.1 Generalisierte Autokorrelation	38

4.3	Histogrammbasierte Verfahren	39
4.4	Oszillatoren	40
4.5	Agentensysteme	41
4.6	Probabilistische Verfahren	43
4.7	Regression	46
4.8	Vergleich der Verfahren	48
II	Tempoerkennung in der Praxis	51
5	Entscheidung für ein Verfahren	53
5.1	<i>k</i> -NN-Regression auf Chromaakzenten	53
5.2	Verallgemeinerung des Verfahrens	56
5.2.1	Akzentanalyse	56
5.2.2	Periodenschätzung	58
5.2.3	<i>k</i> -NN-Regression	58
6	Umsetzung des Verfahrens	61
6.1	Anforderungen	61
6.2	Referenzimplementierung mit Matlab	62
6.2.1	MIRtoolbox	62
6.2.2	Speicherbedarf	63
6.2.3	Laufzeit	64
6.3	Testframework „TempEx“	65
6.3.1	Verwendung des Programms	66
6.3.2	Laufzeit	69
6.3.3	Modul zur Erzeugung der Frames	69
6.3.4	Berechnung der Grundfrequenzstärken	72
6.3.5	Butterworth-Tiefpassfilter	74
6.4	Beispielanwendung „TempEst“	75
7	Ergebnisse	77
7.1	Trainingsdatenbank	77
7.2	Manuell segmentierte Ausschnitte	79
7.2.1	Klassifikation in Tempokategorien	80
7.2.2	Bereinigung des Spektrums	82
7.2.3	Exemplarische Untersuchung der Fehlklassifikationen	83
7.3	Vollständige Musikstücke	86
7.4	Willkürlich beschnittene Stücke	87
7.5	Auswirkung der Größe der Trainingsmenge	88

7.6 Klassifikation unbekannter Musikstücke	90
8 Fazit und Ausblick	93
A Musikdaten	99
A.1 Trainingsmenge	99
A.2 Unabhängige Testmenge	102
Abbildungsverzeichnis	105
Tabellenverzeichnis	108
Literaturverzeichnis	113
Erklärung	115

Kapitel 1

Einführung

Den meisten Menschen ist das Phänomen bekannt: Man hört Musik und reagiert spontan. Die Finger klopfen, der Fuß wippt oder der Kopf nickt im Takt der Musik. Ob das Stück bekannt ist oder nicht, spielt dabei keine wesentliche Rolle. Der Mensch ist also in der Lage, das Tempo und den Beat von Musik zu erkennen. Dabei ist *erkennen* vielleicht nicht die treffende Formulierung. Die Wenigsten machen sich bewusst Gedanken über das Tempo, den Rhythmus und den Takt eines Stücks, und haben dennoch ein Gespür dafür. Geübte Hörer können sich vielleicht darüber Gedanken machen, um was für einen Takt es sich handelt. Dennoch muss man keine musikalische Ausbildung haben, um das Tempo von Musik zu *empfinden*. So sind die meisten Menschen in der Lage, sich im Takt der Musik koordiniert zu bewegen: Menschen können tanzen.

Gerade die Dinge, die der Mensch zum großen Teil unterbewusst – also nebenbei – wahrnimmt, stellen die Informatik vor große Herausforderungen. Computer können die Musik nicht einfach auf sich wirken lassen, um dann den Beat im richtigen Tempo mitzuklopfen. Daher stellt sich die Aufgabe, zu analysieren, auf welche unmittelbaren Einflüsse Menschen reagieren, wenn sie den Beat der Musik fühlen. Aus welchen Informationen in der Musik entwickeln Menschen ihren Eindruck vom Tempo? Und worauf stützen sie sich, wenn sie in Erwartung des nächsten Beats mit dem Finger klopfen, dem Fuß wippen oder mit dem Kopf nicken?

In dieser Arbeit werden diese Fragen nicht grundlegend beantwortet, aber es werden Verfahren vorgestellt, die versuchen, dieses Phänomen abzubilden. Zunächst werden die musikalischen Grundlagen erläutert. Dann werden die verschiedenen Ideen und Ansätze betrachtet und es wird versucht, gemeinsame Ideen aufzuzeigen. Es zeigt sich, dass bestehende Verfahren gerade bei langsamem Liedern Schwierigkeiten haben - nicht nur das genaue Tempo zu schätzen, sondern überhaupt zu erkennen, dass sie langsam sind. Daher wird ein Verfahren gesucht, dass besonders bei langsamem Liedern eine gute Schätzung für das Tempo liefert.

1.1 Aufbau der Arbeit

Ein kleiner Überblick über den Aufbau der Arbeit soll den Einstieg in das Thema Tempoerkennung erleichtern. In diesem Kapitel folgt eine Betrachtung des Tempos aus musikalischer Sicht. Hierzu werden die Grundbegriffe wie Tempo, Schlag und Metrik definiert und in Beziehung zueinander gesetzt. Es werden also die musikalischen Grundlagen der Tempoerkennung gelegt. Außerdem wird gezeigt, dass die Tempoerkennung nur eine der Aufgaben in dem Bereich der Metrik darstellt. Das Problem der Tempoerkennung wird also vom unterbewussten, impliziten Erleben zu einer explizit definierten Aufgabe, die genau umrissen und zu ähnlichen oder weitergehenden Aufgaben abgegrenzt wird.

Die folgenden Kapitel bilden als erster Teil der Arbeit einen zentralen Einstieg in aktuelle Verfahrensweisen zur Tempoerkennung. Zunächst widmet sich Kapitel 2 den Audiosignalen als Art der Eingabe und liefert einen kleinen Einstieg in die Verarbeitung von Audiosignalen.

Klapuri *et al.* [32] fassen prägnant zusammen, welche drei zentralen Probleme gelöst werden müssen, um ein erfolgreiches System zur metrischen Analyse - und damit auch zur Tempoerkennung - aufzubauen:

1. Der Grad an musikalischer Betonung muss über die Zeit abgebildet werden - diskret oder kontinuierlich. Die diskrete Variante ist die Onset-Erkennung und wird in Kapitel 2 erläutert. Kapitel 3 stellt die kontinuierliche musikalische Akzentanalyse vor.
2. Die Perioden (und für andere Anwendungen auch die Phasen) der zugrundeliegenden metrischen Impulse müssen geschätzt werden. Die Methoden hierzu werden in Kapitel 4 diskutiert.
3. Das System muss aus der Schätzung der Perioden die richtige auswählen: Für die Temposchätzung diejenige, die zum Beat korrespondiert. Auch hier liefert das Kapitel 4 genauere Einblicke.

Anschließend wird die Entscheidung für ein Verfahren als Ausgangsbasis begründet, und eine Verallgemeinerung des Verfahrens wird beschrieben. Eine Umsetzung des Verfahrens als .NET-Projekt ermöglicht dann die Auswertung einer Testdatenbank mit Musikstücken. Schließlich werden Ideen aufgezeigt, mit denen die Genauigkeit des herausgearbeiteten Verfahrens noch weiter verbessert werden könnte.

1.2 Musiktheoretische Grundlagen

Ein eingehendes Verständnis der Tempoerkennung ist nur möglich, wenn vorher die musiktheoretischen Grundlagen gelegt und die verwendeten fachlichen Begriffe klar definiert

und voneinander abgegrenzt sind. Peter Benary versteht Musik als „Zeit-Kunst“ und sieht „ihre Zeitlichkeit, ihre Gebundenheit an ein zeitliches Nacheinander“ als „urtümlichste Seite der Musik“ an. Das *Tempo* ist – schon wortwörtlich¹ – ein Maß für diesen zeitlichen Charakter der Musik, wobei das „Hauptaugenmerk dem *Metrum*² als dem – im wörtlichen wie im übertragenen Sinn – Pulsschlag der Musik“ gilt [4, S. 7].

Während der Begriff Rhythmus noch allgemein bekannt ist, kann man sich nur wenig unter dem Begriff Metrum oder Metrik vorstellen. Die „**Rhythmik** ist die Lehre von den musikalischen Dauer-Verhältnissen“ [4, S. 9]. Sie beschreibt also, in welchem Verhältnis die Länge und Kürze einzelner Noten zueinander stehen. Die **Metrik** hingegen „ist die Lehre von den musikalischen Schwereverhältnissen“ [4, S. 9]. Sie erschließt sich dem Zuhörer unwillkürlich als regelmäßiges Muster von starken und schwachen Schlägen (oder englisch „*beats*“) aus dem Klang des MusikstückS. So beschreiben Lehrdahl und Jackendorff [37, S. 12] die Wirkung der Metrik und geben tiefere Einblicke in die Zusammenhänge der metrischen Struktur.

1.2.1 Akzente

Lehrdahl und Jackendorf definieren zunächst, was mit dem Begriff **Akzent** („*accent*“) gemeint ist. Sie teilen Akzente in drei Arten ein: phänomenale Akzente, strukturelle Akzente und metrische Akzente [37, S. 17f]. Für diese Arbeit sind phänomenale und metrische Akzente von zentraler Bedeutung.

Phänomenale Akzente Mit diesem Begriff („*phenomenal accent*“) werden Ereignisse an der musikalischen Oberfläche beschrieben, die einen bestimmten Moment im Fluss der Musik hervorheben oder betonen. Beispiele für diese hörbaren Akzente sind das Anschlagen einer neuen Note, lokale Betonungen, plötzliche Änderungen in Dynamik oder Klangfarbe („*timbre*“) oder Harmoniewchsel. Bei der Akzentanalyse, deren Verfahren in Kapitel 3 im Detail erläutert werden, besteht die Aufgabe genau darin, diese Akzente im Audiosignal möglichst präzise über die Zeit nachzuverfolgen.

Metrische Akzente Unter dem Begriff werden die Schläge aufgefasst, die in ihrem metrischen Kontext relativ stark sind. Genau dieser Akzent ist gemeint, wenn es darum geht, Schläge – oder das Tempo als deren Abstand – zu erkennen. Die genaue Bedeutung wird aber erst klar, wenn die Beziehung zu phänomenalen Akzenten hergestellt wird. Phänomenale Akzente im wahrgenommenen Audiosignal bilden die Grundlage, auf denen der Zuhörer versucht, ein regelmäßiges Muster metrischer Akzente zu extrapolieren. Metrische Akzente sind somit ein geistiges Konstrukt, die subjektive Schlussfolgerung eines metrischen Musters aus den Mustern der hörbaren Akzentuierung.

¹Der Begriff *Tempo* in der Musik ist ursprünglich italienisch und bedeutet übersetzt „Zeit“

²Metrum ist griechisch und heißt übersetzt „Maß“



Abbildung 1.1: Schläge und ihre Abstände

Schläge (und damit auch Tempo) zu erkennen bedeutet also, aus dem Audiosignal phenomenale Akzente als Hinweise zu extrahieren und daraus auf ein regelmäßiges Muster metrischer Akzente zu schließen.

1.2.2 Metrische Hierarchie

Die Bedeutung und Beziehung der Begriffe Schlag, Takt, Periodizität (Wiederholungsrate) und metrisches Muster lässt sich am Besten in der metrischen Hierarchie verdeutlichen.

Lehrdahl und Jackendorf erklären in [37, S. 18ff] anschaulich, wie ein Schlag zeitlich zu verstehen ist und wie sich metrische Muster aus Schlägen zusammensetzen. Wie Punkte in der Geometrie idealerweise eine unendlich kleine Ausdehnung haben, markiert ein **Schlag** einen *Zeitpunkt* unendlich kurzer Dauer. Ein Metronom klickt zum Zeitpunkt des Schlages, aber es hält den Ton nicht dauerhaft. Und genau wie der Abstand zweier Punkte eine messbare Ausdehnung hat, liegt zwischen zwei Beats ein messbares Zeitintervall, eine Zeitspanne oder Dauer.

Es wurde bereits erklärt, dass metrische Akzente ein regelmäßiges Muster bilden. Regelmäßigkeit bedeutet, dass die Schläge äquidistant in der Zeit sind, also die Zeitspannen zwischen ihnen idealerweise gleich sind. Die Metrik teilt also Schläge in gleiche Abstände ein. So können die Schläge in Abbildung 1.1(b) nicht metrisch genannt werden, da die Zeitspannen dazwischen nicht gleichmäßig sind. Dann lässt sich auch von **Periodizität** sprechen: Je kleiner die Zeitspanne zwischen den Schlägen, desto größer ist ihre Periodizität, also ihre Anzahl innerhalb einer Zeiteinheit.

Allerdings bilden auch die Schläge in Abbildung 1.1(a) noch kein metrisches Muster. Die Metrik beschreibt „musikalische Schwereverhältnisse“, aber die abgebildeten Schläge haben alle das gleiche Gewicht, es gibt keinen Wechsel zwischen schweren und leichten Schlägen. Es sind mehrere *Ebenen* von Schlägen in einer **metrischen Hierarchie** notwendig, um die Schwereverhältnisse der Metrik zu beschreiben. Die Schläge einer Ebene sind äquidistant und haben alle dasselbe Gewicht.

Die Abbildung 1.2(a) zeigt das Zusammenspiel der Ebenen am Beispiel des 4/4-Taktes. Dort sieht man 4 Ebenen von Schlägen eingezeichnet: Achtel-, Viertel-, halbe und ganze Noten. Die Gewichtung der jeweiligen Zählzeiten in der Metrik ergibt sich durch die Summierung der Schläge über die Ebenen. Daher sind die Zählzeiten 1 und 3 stärker betont als 2 und 4, und der erste Schlag des Taktes ist stärker betont als der dritte.

Zeit	1	2	3	4	1	2	3	4
$\frac{1}{4}$	●	●	●	●	●	●	●	●
$\frac{1}{2}$	●	●	●	●	●	●	●	●
$\frac{1}{4}$	●		●		●		●	
$\frac{1}{8}$	●				●			
Gewicht	4	1	2	1	3	1	2	1

(a) 4/4-Takt: Die Viertel sind die Grundsäume, und die Zeitspannen zwischen den Ebenen unterscheiden sich immer um Faktor zwei.

Zeit	1	2	3	1	2	3
$\frac{1}{4}$	●	●	●	●	●	●
$\frac{1}{2}$	●	●	●	●	●	●
$\frac{1}{4}$	●			●		
$\frac{1}{8}$						
Gewicht	3	1	2	1	2	1

(b) 3/4-Takt: Auch hier bilden die Viertel den Grundsäum. Während zu den Achteln der Faktor zwei liegt, bilden drei Viertel einen Takt (erst Faktor 2, dann 3).

Zeit	1	2	3	4	5	6	1	2	3	4	5	6
$\frac{1}{4}$	●	●	●	●	●	●	●	●	●	●	●	●
$\frac{1}{2}$	●			●			●		●		●	
$\frac{1}{4}$	●						●			●		
$\frac{1}{8}$												
Gewicht	3	1	1	2	1	1	3	1	1	2	1	1

(c) 6/8-Takt: Der Grundsäum besteht aus den Achteln. Jeweils drei Achteln werden zur nächsten Ebene zusammengefasst, und zwei punktierte Viertel ergeben einen Takt (erst Faktor 3, dann 2).

Abbildung 1.2: Die verschiedenen Ebenen der metrischen Hierarchie am Beispiel unterschiedlicher Taktarten. Hervorgehoben wird jeweils die Ebene der Grundsäume, also der *Tactus* oder Beat.

Die Abbildung 1.2 zeigt insgesamt, dass die Beats auch auf den höheren Ebenen gleiche Abstände haben. Deshalb spricht man vom „metrischen Raster“, in dem die Periodizitäten von einer Ebene zur nächsten verstärkt werden, und die Ebenen können durch die Länge der Zeitspannen zwischen den Beats - etwa durch Notenlängen - angegeben werden. In der klassischen westlichen tonalen Musik ist die Zeitspanne einer Ebene immer das zwei- oder dreifache der Zeitspanne der darunterliegenden Ebene. Damit stellt sich die Metrik für einen 3/4-Takt und einen 6/8-Takt gemäß den Abbildungen 1.2(b) und 1.2(c) dar.

Tactus, Tatum, Takt und Tempo

Die Begriffe Tatum, Takt und Tactus bezeichnen spezielle Ebenen der metrischen Hierarchie. Unter dem Begriff **Tatum** versteht man die kleinste (und damit schnellste) in dem Stück wahrgenommene metrische Ebene. Der Begriff Takt wird oft als Synonym für Metrum verwendet. In den folgenden Kapiteln bezeichnet **Takt** („measure“) aber die Gruppierung von Notenwerten gleicher Zählzeit, wobei diese in der Regel dem Tactus entspricht. In Abbildung 1.2(a) sind die vier Zählzeiten des 4/4-Takts zu erkennen. Ein Verfahren, dass in diesem Fall den Takt schätzt, versucht also, die Schläge der Ebene der ganzen Noten zu bestimmen.

Die verschiedenen Ebenen der metrischen Hierarchie werden jedoch nicht gleich stark wahrgenommen. Der Zuhörer konzentriert sich besonders auf eine mittlere Ebene, in denen

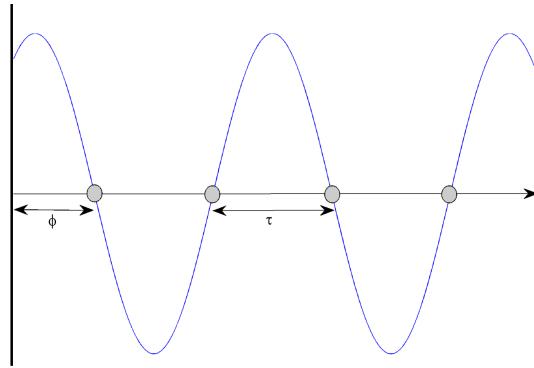


Abbildung 1.3: Grundschläge, mit einer imaginären Sinusschwingung abgeglichen. Der Abstand $\tau = \frac{1}{f}$ ist die Periodendauer des Tempos f .

der Abstand der Schläge moderat ist [37, S. 21]³. Die Schläge dieser Ebene erscheinen dem Hörer als regelmäßiger Puls, der sich durch das Musikstück zieht. Diese Ebene wird daher als Grundpuls, Grundsenschlag oder **Tactus** bezeichnet. Der englische Begriff **Beat** wird oft in diesem Sinn verwendet, und so bezieht sich die Beaterkennung auf das Erkennen des Grundschlags.

Der Tactus als hervorstechene metrische Ebene bestimmt das wahrgenommene **Tempo** der Musik, und ist also genau das, was Menschen mitklopfen oder der Dirigent eines Orchesters dirigiert. Ist im Folgenden von Tempo in **Bpm** („beats per minute“) die Rede, bezieht sich dies auf die Anzahl der Grundschläge in der Minute, und übliche Tempowerte liegen zwischen 40 Bpm und 240 Bpm.

1.3 Aufgaben der Tempoerkennung

Im Bereich der musikalischen Metrik finden sich einige interessante Aufgaben. Zuerst wäre da die **Tempoerkennung** (in der Literatur oft als „tempo estimation“ oder „tempo induction“ bezeichnet). Die Tempoerkennung beschäftigt sich ausschließlich mit dem Abstand der Grundschläge. Man kann sich die Grundschläge eines Stücks als Sinusschwingung

$$y(x) = \sin(\pi f x + \phi) \quad (1.1)$$

vorstellen, die genau bei jedem Grundsenschlag eine Nullstelle hat. In der Tempoerkennung geht es dann darum, das Tempo, respektive die Frequenz f der Grundschläge, zu schätzen.

Möchte man nicht nur die Frequenz der Grundschläge, sondern auch ihre genaue zeitliche Position wissen, so muss man zu f auch die Anfangsphase ϕ ermitteln. Dann spricht man von der **Beaterkennung** („beat detection“). Das ist genau die Aufgabe, die Menschen - wie in der Einleitung beschrieben - unterbewusst lösen, wenn sie beginnen im Takt der

³Lehrdahl und Jackendorff schreiben an dieser Stelle genaugenommen von „ein (oder zwei) Ebenen“.

Musik mitzuklopfen. Sie erfühlen f und ϕ der Beats und schätzen dann vorab, wann der nächste Beat sein wird.

Offensichtlich geht es sowohl bei der Tempo- als auch bei der Beaterkennung nur um eine Ebene der metrischen Hierarchie: die der Grundschlüsse. Eine weitere Aufgabe besteht darin, die komplette metrische Hierarchie zu analysieren und auf Takt oder Rhythmus zu schließen.

Bei den verschiedenen Aufgaben kann man differenzieren, welche Daten bei der Erkennung verfügbar sind. Bei der Analyse während der Wiedergabe von Musik ist nur das bekannt, was schon wiedergegeben ist. Aufgrund der Beobachtungen in der Vergangenheit soll das aktuelle Tempo geschätzt oder der nächste Beat vorhergesagt werden. Das Verfahren wird dann **kausal** („*causal*“) genannt. Beispielsweise können so Lichteffekte mit der Musik automatisch beatsynchron erzeugt werden. Für diesen Einsatz müssen die verwendeten Verfahren auch echtzeitfähig sein.

Bei anderen Anwendungen liegen alle Daten schon zu Beginn vor, etwa wenn das Tempo von verschiedenen Musikstücken einer Musikbibliothek geschätzt werden soll. Diese Anwendungen bezeichnet man als **nichtkausal** („*non-causal*“).

1.4 Ziel der Arbeit

Diese Diplomarbeit beschränkt sich auf das Problem der Tempoerkennung bei vollständig vorhandenen Musikdaten. In Kapitel 4 werden einige bekannte Verfahren vorgestellt.

Die Genauigkeit der meisten Programme zur Tempoerkennung lässt bei langsamer Musik zu wünschen übrig. Das geschätzte Tempo liegt häufig nahe dem Doppelten des wirklichen Tempos des Stücks. Durch die schon betrachteten musiktheoretischen Grundlagen ist dieses Phänomen leicht zu erklären: Das Verfahren ist nicht auf Ebene der Grundschlüsse, sondern auf einer benachbarten metrischen Ebene gelandet, und hat beispielsweise bei einem 4/4-Takt die Achtel als Tactus erkannt.

Ziel dieser Arbeit ist, zu untersuchen, wie auch für langsame Musik das Tempo ausreichend genau geschätzt werden kann. Im Blickpunkt liegt hierbei nicht so sehr, möglichst genau den Wert des tatsächlichen Tempos zu treffen, sondern Sprünge zu höher- oder tiefergelegenen metrischen Ebenen – also das Phänomen der Verdopplung oder Halbierung – zu unterbinden. Ernüchternd ist dabei aber die schon in Abschnitt 1.2.2 von Lehrdahl und Jackendorf verwendete Formulierung, dass der Zuhörer sich auf „ein (oder zwei)“ [37, S. 21, im Original nicht kursiv] metrische Ebenen konzentriert. Die Entscheidung für eine metrische Ebene als Tactus ist somit auch für den menschlichen Hörer allein aus dem Gehörten nicht immer eindeutig möglich. Somit kann auch von einem Verfahren keine vollständige Präzision erwartet werden. Es soll aber untersucht werden, wodurch sich die metrischen Fehlinterpretationen des vorgestellten Verfahrens auszeichnen.

Teil I

Grundlagen der Tempoerkennung

Kapitel 2

Audio, Signalverarbeitung und Onset-Erkennung

An dieser Stelle wird auf die Art der Eingabedaten näher eingegangen: Audiodaten. Gesucht ist ein Verfahren, dass das Tempo aus Audiosignalen erkennt. In seinem Kapitel über Beaterkennung und Metrumanalyse nennt Stephen Hainsworth die Art der Eingabe „die erste und wichtigste Unterscheidung“ [25, S. 102] der Verfahren zur Tempoerkennung. Die Eingabe steht am Anfang der Verfahren, daher ist eine Betrachtung der Implikation von Audiosignalen als Eingabe im Vergleich zu symbolischen Daten grundlegend für ein genaues Verständnis der Idee, die hinter verschiedenen Verfahren steht.

Historisch betrachtet, arbeiten die ersten Verfahren zur Tempo- und Beaterkennung nicht auf Audiosignalen, sondern auf symbolischen Daten. Symbolisch bedeutet hier, dass als Eingabedaten beispielsweise die Zeitpunkte gegeben sind, an denen Noten beginnen – gewöhnlich in der gewünschten Quantisierung. MIDI-Daten sind eine symbolische Eingabe – sie beschreiben die genauen Zeiten, zu denen eine Note beginnt (bzw. eine Taste in einer bestimmten Intensität gedrückt wird) und endet (die Taste wird losgelassen).

Symbolische Eingabedaten sind also keine unmittelbar akustisch wahrnehmbaren Signale. MIDI-Daten beispielsweise sind vielmehr eine diskrete Abfolge von Ereignissen, die beschreiben, wie ein Tongenerator (Synthesizer) ein Audiosignal erzeugen kann. Daraus ergibt sich, dass die Datenmenge relativ gering ist – die zur (Re-)Produktion der richtigen Klangfarbe erforderlichen Daten befinden sich beim Synthesizer. Verfahren, die symbolische Daten verarbeiten, benötigen daher deutlich weniger Rechenleistung. Die entscheidendere Implikation ist, dass in den symbolischen Eingabedaten der exakte Zeitpunkt explizit definiert ist. MIDI-Daten beschreiben ferner genau die Note und deren Intensität sowie andere Merkmale, die nachher dem hörbaren Sound zugrunde liegen. Das bedeutet, dass sich die Verfahren auf die symbolischen Eingabedaten verlassen können – sie beschreiben explizit, was wann geschieht.

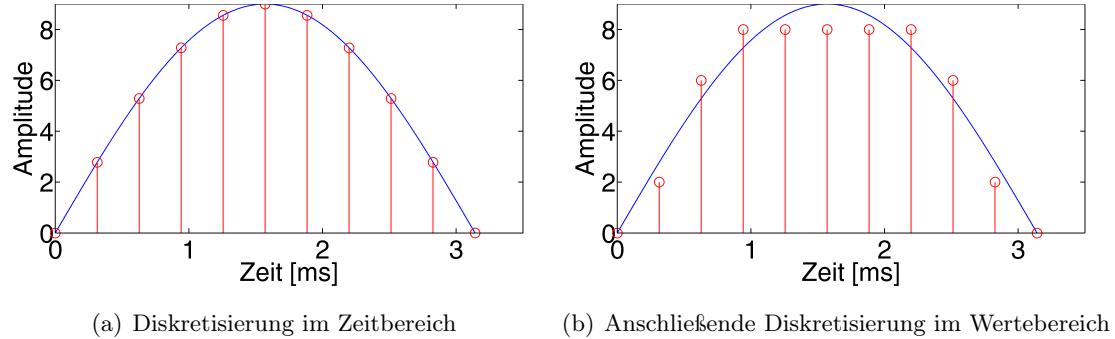


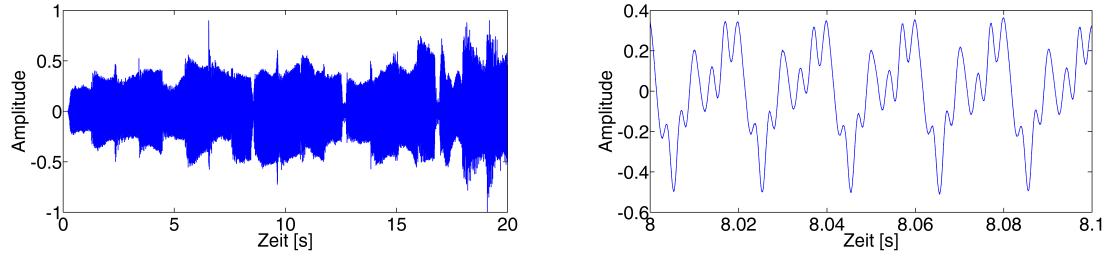
Abbildung 2.1: Diskretisierung des analogen Audiosignals

In der Praxis begegnet man jedoch meist akustischen Signalen und nicht deren symbolischer Repräsentation. Menschen hören akustische Signale mit den Ohren und lesen nicht die Noten eines Liedes und erzeugen es dann im Gehirn. Im Unterschied zu diskreten symbolischen Daten beschreibt ein akustisches Signal (Audiosignal) den Schalldruck kontinuierlich in der Zeit und damit den wahrgenommenen Klang. Im Audiosignal sind keine expliziten Informationen über Startzeiten oder Tonhöhen von Noten vorhanden. Die besondere Schwierigkeit bei Audiosignalen liegt also darin, dem Signal Informationen über die Merkmale zu entlocken, aus denen das Tempo ermittelt werden kann. Bei der Tempoerkennung sollten diese Merkmale so präzise wie möglich Zeitpunkte extrahieren, zu denen in der Musik etwas von Bedeutung geschieht - beispielsweise eine neue Note angespielt wird. Welche Methode dafür auch zum Einsatz kommt, das Verfahren muss berücksichtigen, dass die Daten immer eine gewisse Ungenauigkeit und Unsicherheit in sich tragen. Im folgenden werden Audiosignale näher betrachtet und untersucht, wie aus dem Audiosignal zur Tempoerkennung relevante Merkmale extrahiert werden können.

2.1 Audiosignale

Audiosignale beschreiben Töne und Klänge als (elektrisches) Signal und transportieren akustische Informationen. Bei einem *analogen* Audiosignal ist die Spannung proportional zum Schalldruck. Viele Bereiche der Audioverarbeitung arbeiten heute aber nicht mehr auf analogen Audiosignalen. So sind die Audiodaten auf einer CD in *digitaler* Form gespeichert. Um analoge Signale digital zu repräsentieren, muss auf zwei Dimensionen diskretisiert werden:

Diskretisierung im Zeitbereich Sind Audiosignale im Grunde eine kontinuierliche Funktion über die Zeit, so müssen sie zur Verarbeitung im Computer zeitlich diskretisiert werden. Hierzu wird das analoge Signal in bestimmten Zeitintervallen abgetastet, und der gemessene Wert wird aufgezeichnet. Die **Abtastrate** ist hierbei ein entscheidender Faktor für die Qualität des digitalen Signals: Sie bestimmt, welche Frequenzen



(a) Das gesamte Audiosignal mit einer Länge von etwa 20 Sekunden

(b) Ausschnitt von 100 ms Länge aus dem Audiosignal

Abbildung 2.2: Audiosignal im Zeitbereich

des analogen Signals korrekt wiedergegeben werden können. Im Abtasttheorem beschreiben Nyquist und Shannon, dass bei einer Abtastrate f_s nur Frequenzen bis $f_s/2$ korrekt reproduziert werden. Höhere Frequenzen können so genannte Alias-Effekte erzeugen - also „Phantomschwingungen“ bei tieferen Frequenzen, die ursprünglich nicht vorhanden waren. Eine Tiefpassfilterung vor der Abtastung verhindert dieses Phänomen. Eine CD arbeitet beispielsweise mit einer Abtastrate von 44,1 kHz, also werden Frequenzen bis zu einer Höhe von 22050 Hz wiedergegeben.

Diskretisierung im Wertebereich Auch der Wertebereich muss digital dargestellt werden. Die elektrische Spannung (entspricht dem Schalldruck) wird zur Abtastzeit t (analog) gemessen und dann auf den nächstgelegenen Wert diskretisiert. Bei digitalen Audiosignalen wird die Genauigkeit im Wertebereich durch die Anzahl der Bits pro Abtastung („*bits per sample*“) angegeben. Bei CDs werden hier 16 Bit pro Sample benutzt, sodass $2^{16} = 65536$ diskrete Werte verfügbar sind, um die verschiedenen Nuancen des Schalldrucks zur Zeit t darzustellen.

Auch wenn ein digitales Audiosignal sowohl im Zeit- als auch im Wertebereich diskretisiert ist, werden Audiosignale hier als kontinuierliche Signale behandelt. Das liegt an den hohen Anforderungen an die Auflösung, die das menschliche Gehör an Audiosignale stellt. Es reagiert sehr sensibel auf Zeitverschiebungen oder fehlerhafte Wiedergaben - deutlich sensibler als das Auge. Daraus ergibt sich auch die große Datenmenge von Audiosignalen. Ein vierminütiges Musikstück ($l = 240$ s) auf einer CD hat - bei einer Abtastrate $f_s = 44100$ Hz und $b = 16$ Bits pro Sample eine Größe von

$$l \cdot f_s \cdot b = 240[\text{s}] \cdot 44100[\text{Hz}] \cdot 16[\text{Bit}] = 20,187[\text{MB}]$$

pro Kanal - im Stereoformat also 40,37 Megabyte.

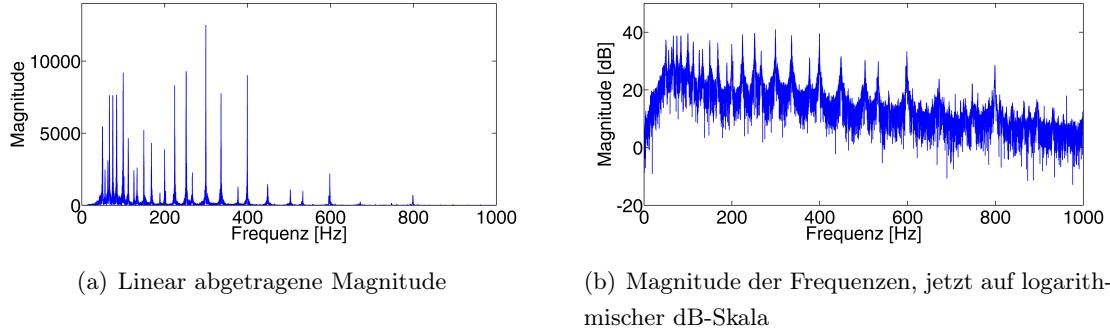


Abbildung 2.3: Spektrum des gesamten Audiosignals, eingeschränkt auf den Bereich unter 1 kHz

2.2 Zeit, Energie und Spektrum

Es gibt zwei grundlegende Bereiche, in denen Audiosignale repräsentiert und verstanden werden können [11]. Der erste Bereich ist der **Zeitbereich**, in dem Audiosignale *aufgenommen* und *wiedergegeben* werden. Hier wird der zur jeweiligen Zeit gemessene Schalldruck dargestellt. Der Schalldruck kann mit einem Mikrofon gemessen und mit einem Lautsprecher wieder erzeugt werden. Allerdings vermittelt die reine Zeitdarstellung nur einen Eindruck von der Entwicklung der Lautstärke des Signals, wie der Abbildung 2.2 zu entnehmen ist.

Der zweite Bereich, der **Frequenzbereich** oder das **Spektrum**, bietet sich zur *Analyse* des Audiosignals an. In Abbildung 2.2(b) kann man die periodische Veränderung im Schalldruck zwar gut erkennen, aber die Frequenzen sind nicht ersichtlich. Im Spektrum wird hingegen deutlich, welche Frequenz wie stark im Audiosignal vorhanden ist. Einen Ausschnitt aus dem Frequenzspektrum zeigt die Abbildung 2.3. Da die Tonhöhe von Noten in der westlichen Musik die Grundfrequenz des Tons repräsentiert, gewinnt man aus dem Frequenzspektrum einen Eindruck über die Töne, aus denen sich das Audiosignal zusammensetzt [11].

2.2.1 Fouriertransformation

Die **Fouriertransformation (FT)** ist das mathematische Mittel, um ein Audiosignal vom Zeit- in den Frequenzbereich umzuwandeln. In der Praxis sind Audiosignale wie oben definiert diskrete, abgetastete Zeitsignale. Sei $x(n)$ mit $n = 0, \dots, N-1$ ein diskretes Signal mit der Länge N , wobei n die Zeit des Samples angibt. Die **diskrete Fouriertransformation (DFT)** ist dann definiert als

$$\text{DFT}_x(k) = X(k) = \sum_{n=0}^{N-1} x(n) e^{-i \frac{2\pi}{N} kn} \quad (2.1)$$

und liefert komplexe Werte für die diskreten Frequenzen k mit $k = 0, \dots, N-1$. Die Umwandlung vom Frequenzbereich in den Zeitbereich führt die **inverse Fouriertrans-**

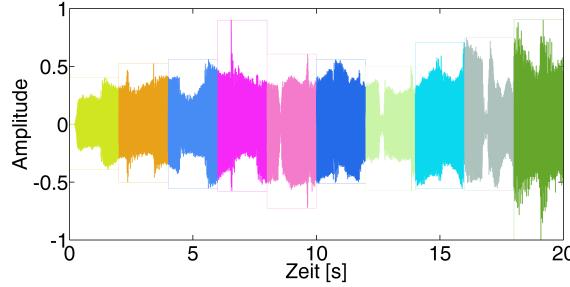


Abbildung 2.4: Die Einteilung des Audiosignals in Fenster (Frames) von zwei Sekunden Länge. Auf eine Überlappung der Fenster wurde für die Darstellung verzichtet.

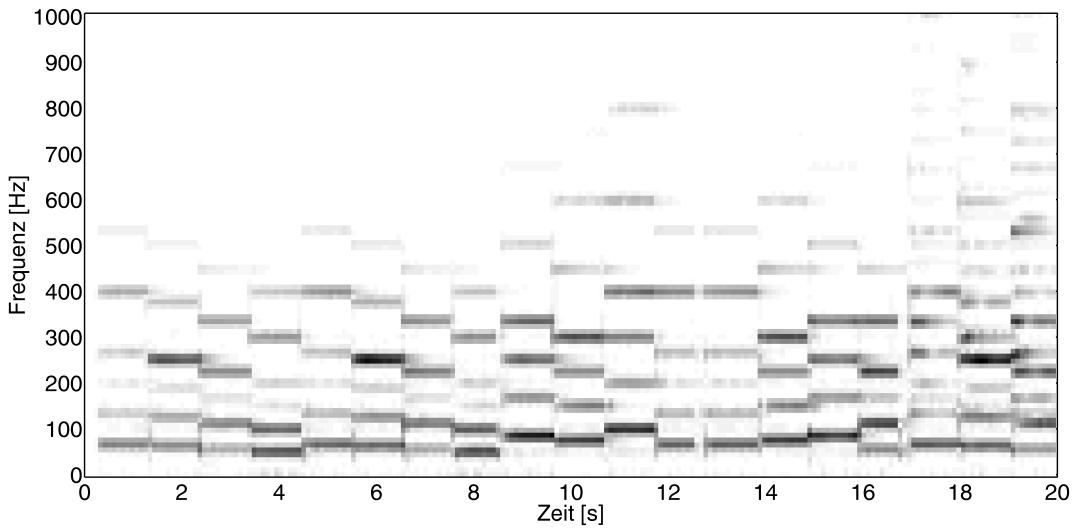


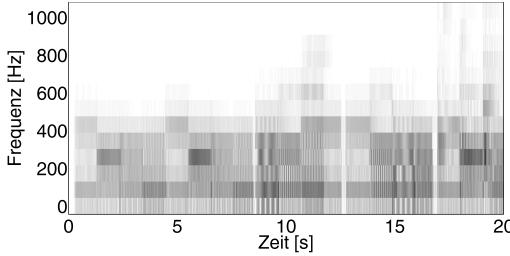
Abbildung 2.5: Spektrogramm mit einer Fenstergröße von 4096 Samples (93 ms)

formation (IFT) durch, und im Fall von einem diskreten, komplexwertigen Spektrum $X(k)$ mit $k = 0, \dots, N - 1$ ist die **inverse diskrete Fouriertransformation (IDFT)** wie folgt definiert:

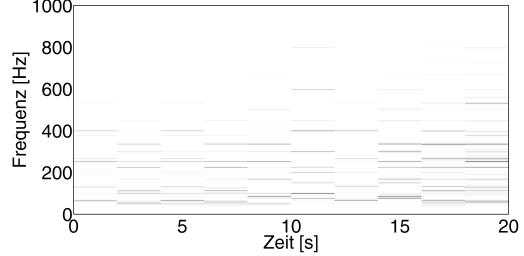
$$\text{IDFT}_X(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{i \frac{2\pi}{N} kn} = x(n) \quad (2.2)$$

2.2.2 Spektrogramm

Will man das Tempo von Audiosignalen bestimmen, hilft eine Umwandlung des gesamten Signals in das Frequenzspektrum nicht weiter. Offensichtlich werden Informationen sowohl aus dem Frequenzbereich als auch aus dem Zeitbereich benötigt. Präziser möchte man oft die Entwicklung von Merkmalen des Frequenzspektrums über die Zeit hinweg verfolgen. Zu diesem Zweck gibt es eine Menge von Zeit-Frequenz-Repräsentationen, und nach Davy sind Noten auch nichts anderes als eine speziell codierte Zeit-/Frequenz-Repräsentation: Sie beschreiben die Zeit, die Höhe (Frequenz) und die Dauer der zu spielenden Töne [25].



(a) Eine Fenstergröße von 512 Samples sorgt für gute Auflösung der Zeit, aber die Frequenzen sind grob.



(b) Bei einer Fenstergröße von 2 Sekunden ist die Frequenz sehr genau abzulesen, aber die Zeit ist ungenau.

Abbildung 2.6: Das Spektrogramm stellt immer einen Kompromiss zwischen guter Auflösung der Zeit- oder der Frequenzachse dar.

Die Basis von Zeit-Frequenz-Repräsentationen sind **Fenster** („*Frames*“), ein Konzept, dass sich durch viele Algorithmen zur Audioverarbeitung zieht. Das Signal wird dazu in der Praxis in Abschnitte einer bestimmten Länge und Position eingeteilt und über eine Fensterfunktion w erzeugt. Fensterfunktionen sind im Normalfall positiv und symmetrisch und haben begrenzten Support: Es gibt ein W , so dass für alle $t \geq W : w(t) = 0$ gilt. Dann wäre der Frame mit der Position t_0 und Fensterfunktion w gegeben durch

$$s_{t_0}^w(t) = x(t_0 + t)w(t), \quad t = 0, \dots, W - 1 \quad (2.3)$$

In der Verarbeitung von Audiosignalen finden gewöhnlich Gauss-, Hamming-, von-Hann- oder Rechteckfenster Verwendung und die Frames haben eine Dauer von 20ms bis 100ms. Wendet man auf diese Frames die DFT an, erhält man die Kurzzeit-Fouriertransformation **STFT** („*short-time Fourier transform*“):

$$\text{STFT}_x^w(t, f) = \text{DFT } s_{t_0}^w(f) \quad (2.4)$$

Die Leistung der Frequenzkomponenten der STFT bildet das **Spektrogramm** SP_x^w als bekannteste Zeit-Frequenz-Darstellung des Audiosignals:

$$SP_x^w(t, f) = |\text{STFT}_x^w(t, f)|^2 \quad (2.5)$$

Das Spektrogramm stellt immer einen Kompromiss zwischen guter Auflösung im Zeitbereich (viele, kleine Frames) und guter Auflösung im Frequenzbereich (lange Frames) dar. Dies wird deutlich, wenn man mit dem Shannon-Theorem die Nyquist-Frequenz bestimmt. Wenn k_s die Samplingrate des Signals ist, ist $k_s/2$ die Nyquist-Frequenz und gibt an, welche Frequenzen aus dem Signal rekonstruiert werden können. Bei höheren Frequenzen können aufgrund der Abtastung Alias-Effekte auftreten. Parallel dazu sind die Ergebnisse der DFT symmetrisch, sodass bei einer geraden Framegröße N genau $N/2$ informative Werte vor-

liegen. Bei einer Abtastrate des Signals von k_s Hz und einer Framegröße N ergeben sich folgende Auflösungen im Zeit- und Frequenzbereich:

$$rest = \frac{N}{k_s} [\text{s}] \quad (2.6)$$

$$res_f = \frac{k_s/2}{N/2} = \frac{k_s}{N} [\text{Hz}] \quad (2.7)$$

Gehen wir von der Abtastfrequenz einer CD von $k_s = 44100$ Hz aus, ergeben sich für Frames der Länge $N_1 = 1024$ und $N_2 = 4096$ also die Auflösungen $rest_1 = \frac{1024}{44100 \text{ Hz}} \approx 0,023$ s = 23 ms mit einer Auflösung im Frequenzbereich von $res_{f1} = \frac{44100 \text{ Hz}}{1024} \approx 43$ Hz für N_1 und $rest_2 = \frac{4096}{44100 \text{ Hz}} \approx 0,093$ s = 93 ms mit einer Auflösung im Frequenzbereich von $res_{f2} = \frac{44100 \text{ Hz}}{4096} \approx 10,8$ Hz für N_2 . Da die Frequenzen der DFT linear verteilt sind, ist gerade im Bereich tiefer Töne auf die Auflösung im Frequenzbereich zu achten.

2.3 Onset-Erkennung

Viele Verfahren zur Tempoerkennung bestimmen das Tempo nicht direkt aus dem Eingangssignal. Vielmehr besteht der erste Schritt darin, aus dem Audiosignal Merkmale zu extrahieren, aus denen sich das Tempo leichter bestimmen lässt. Eine Methode zur Vorverarbeitung der Audiodaten ist die Onset-Erkennung. Während das Tempo als ein kontinuierliches, veränderliches Signal angesehen werden kann, liegen ihm doch diskrete musikalische Ereignisse zu Grunde: die gespielten Noten. Viele der Verfahren zur Tempoerkennung beruhen auf symbolischen Daten, die die Startzeiten („onset times“) der Noten darstellen. [25] Ziel der Onset-Erkennung ist daher, aus dem Audiosignal diskrete Startwerte der Noten zu erkennen. Diese können dann mit bestehenden symbolischen Verfahren verarbeitet werden. Die Genauigkeit der Onset-Erkennung beeinflusst damit in hohem Maße die Genauigkeit des gesamten Prozesses der Tempoerkennung.

Die Beschränkung auf die Betrachtung der Startzeiten der Noten ergibt sich daraus, dass sich das Ende einer gespielten Note, und damit auch die Notenlänge, oft nicht genau erkennen lässt. Dies liegt an den verschiedenen möglichen Spielweisen (legato, staccato), an der Raumakustik (Nachhall) und am Ausschwingen des Tons bei dem jeweiligen Instrument.

In diesem Zusammenhang gilt es zwei Kategorien von Schallquellen zu unterscheiden. *Harmonische* Klänge werden gewöhnlich als Noten erkannt und haben eine erkennbare Tonhöhe. *Perkussive* Klänge, etwa vom Schlagzeug, erzeugen hingegen ein breites Frequenzspektrum und sind daher eher als Klangwolke zu erkennen. Sie sind an einem deutlichen Anstieg der Signalenergie (einem „Transient“) zu erkennen. Die meisten „harmonischen“ Instrumente haben auch einen sprunghaften Anstieg der Signal. Daher ist der Bereich transienter Onset-Erkennung gut erforscht.

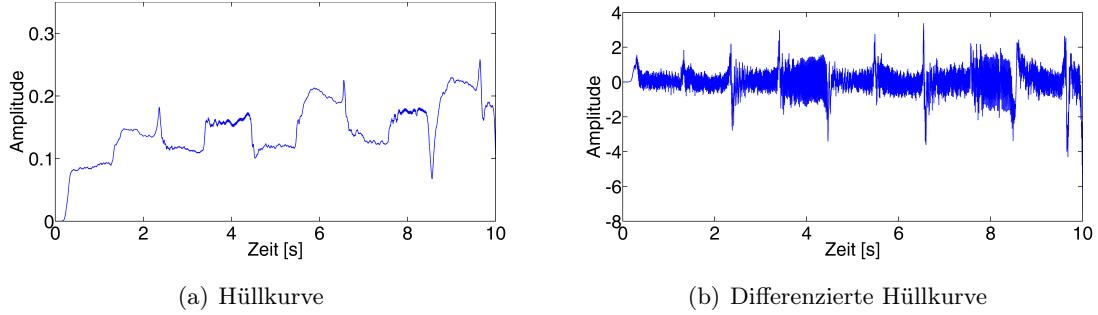


Abbildung 2.7: Die Hüllkurve des Audiosignals

2.3.1 Transiente Onsets

Transiente Onsets, etwa vom Schlagzeug oder durch Instrumente, bei denen Noten mit deutlichem Energieanstieg angeschlagen werden – zum Beispiel Klavier oder Gitarre – sind leicht an der Energiedichte zu erkennen. Die Hüllkurve wird durch die Funktion $E_j(n)$ beschrieben, die die Leistungen der Frequenzen aus einem gewünschten Frequenzbereich für jeden Frame aufsummiert:

$$E_j(n) = \sum_{k \in \kappa_j} |\text{STFT}_x^w(n, k)|^2 \quad (2.8)$$

Bei der Analyse ist die typische Größe der Frames etwa 20 ms mit einer Überlappung von 50% bis 75%. Es können verschiedene Frequenzbänder j betrachtet werden, die unterschiedlich aufgeteilt werden. Interessant ist in jedem Fall der tieffrequente Bereich von 20-200 Hz, während im mittleren Bereich von 200 Hz bis 15 kHz die meisten harmonischen Informationen zu finden sind. Der Bereich über 15 kHz enthält keine harmonischen, aber starke transiente Informationen. Daher gibt es viele Möglichkeiten, das Frequenzspektrum aufzusplitten, etwa eine Aufteilung in 5-10 Teilbänder, die gleichmäßig auf einer logarithmischen Frequenzskala verteilt sind [25].

Die Aussagekraft der Transients lässt sich erhöhen, indem $D_j(i)$, der Gradient von $E_j(i)$ betrachtet wird. Dieser Ansatz erkennt den Beginn der Transients und wird häufig verwendet. Klapuri *et al.* differenzieren in [32] die Energiedichte mit

$$D_j(n) = E_j(n) - E_j(n - 1) \quad (2.9)$$

und berechnet damit die lineare Regression auf zwei Beobachtungen. Für drei Beobachtungen ergibt sich der Gradient als lineare Regression mit

$$D_j(n) = \frac{E_j(n + 1) - E_j(n - 1)}{3}. \quad (2.10)$$

Die Onsets kann man nun an den Maxima von $D_j(n)$ erkennen. Die Schwierigkeit ist hier, die richtigen Maxima zu identifizieren, indem man diejenigen verwirft, die bestimmten Kriterien nicht genügen:

- Maxima mit zu geringer Energie (Beispielsweise mindestens das Doppelte des lokalen Mittels von E_j über 1.5 Sekunden)
- Maxima mit einem höheren Peak in der lokalen Umgebung (Dixon's Timing Kriterium) - daher aus psychoakustischer Sicht nicht relevant

Sind mehrere Teilbänder j vorhanden, können die Gradienten $D_j(n)$ vorher gleichgerichtet und summiert werden [25].

Kapitel 3

Musikalische Akzentanalyse

In der Praxis hat sich gezeigt, dass bessere Ergebnisse erzielt werden, wenn die musikalische Betonung nicht diskret als Onsets, sondern als kontinuierliches Merkmal über die Frames gemessen wird [23]. Eronen nennt das Erzeugen dieses kontinuierlichen Merkmals „musikalische Akzentanalyse“ (*musical accent analysis*) [16, S. 43]. Das erinnert an die Definition des phänomenalen Akzents im Abschnitt 1.2.1 der musikalischen Grundlagen, und genau dieser phänomenale Akzent soll abgebildet werden. Das Ziel ist, Zeitpunkte des Audiosignals hervorzuheben, die markant sind. Den markanten Stellen im Audiosignal liegen diskrete Tonereignisse zugrunde, beispielsweise

- die Startzeiten der Noten,
- plötzliche Energiesprünge,
- Veränderungen der Klangfarbe oder
- Harmoniewechsel, die bei gebunden gespielten Noten nicht unbedingt mit Energieschüben einhergehen.

Die Methoden zur Merkmalsextraktion stammen vornehmlich aus der Signalverarbeitung und umfassen zeitliche und spektrale Merkmale. In [32] und [17] verwenden Eronen und Klapuri verschiedene Spektralmerkmale zur Akzentanalyse. Dabei wird das Signal in Frequenzbänder zerlegt und dann bestimmt, wo Veränderungen stattfinden. Zur Trennung des Signals kommen verschiedene Techniken zum Einsatz:

- Diskrete Fouriertransformation
- Filterbänke
- Chroma-Analysierer

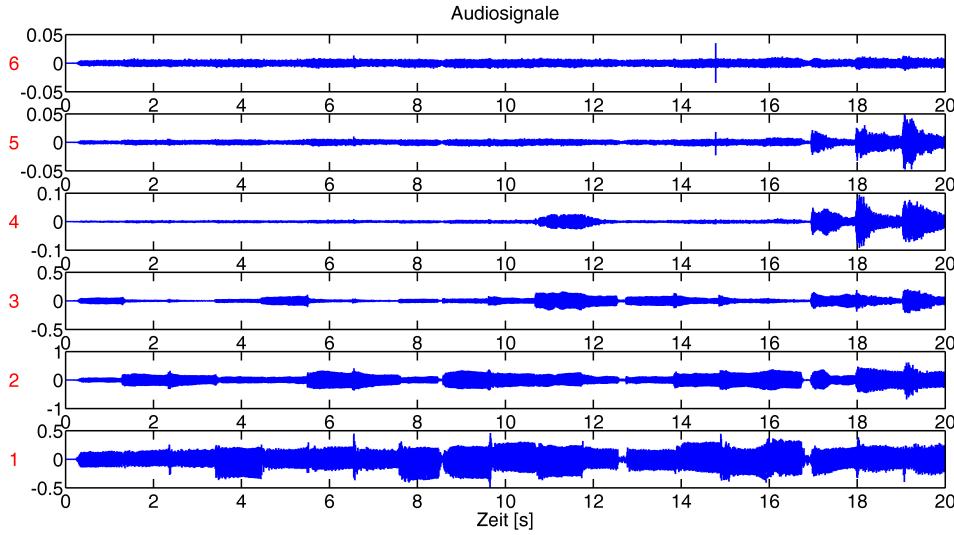


Abbildung 3.1: Frequenzbänder eines Audiosignals nach der Filterbank

3.1 Akzentanalyse auf einzelnen Frequenzbändern

Grundlegend für das Arbeiten mit einzelnen Frequenzbändern ist die Beobachtung Scheirers, dass sich einige Vereinfachungen am Audiosignal vornehmen lassen, ohne die Wahrnehmung des Rhythmus zu verändern. Scheirer extrahiert auf den einzelnen Bändern nur die Hüllkurve und moduliert damit dann ein weißes Rauschen. Wenn das Rauschen auf den einzelnen Bändern moduliert und anschließend das modulierte Rauschen über die Bänder summiert wird, bleibt der rythmische Eindruck erhalten - kehrt man die Reihenfolge um und summiert zunächst die Hüllkurven und moduliert damit das Eingangssignal, geht die rhythmische Wirkung verloren [45].

Klapuri *et al.* beschreiben, welche Überlegungen der bandweisen Berechnung des Akzentsignals vorausgehen. Scheirer verwendet in der oben genannten Arbeit lediglich sechs logarithmisch verteilte Frequenzbänder (separiert an den Frequenzen 200 Hz, 400 Hz, 800 Hz, 1600 Hz und 3200 Hz). Hiermit lassen sich Akzente in Musikstücken mit deutlichen Beats gut erkennen, da diese deutliche Energiesprünge erzeugen. In ruhigeren Stücken kommen die Akzente jedoch oft durch Harmoniewechsel zustande. Um Harmoniewechsel oder die Anfänge gebunden gespielter Noten erkennen zu können ist jedoch eine Auflösung in etwa 40 Teilbänder nötig. Bei so vielen Teilbändern ist es aber nicht mehr möglich, die Periodizität genau genug abzuschätzen. Die Hüllkurven der einzelnen Bänder sind dann eventuell zu schmal, um metrische Intervalle - geschweige denn die korrekten - erkennen zu können [32].

Eine Möglichkeit, dieses Dilemma zu umgehen, zeigen Goto und Muraoka auf [21]. Hier werden die Energiesprünge auf den Bändern berechnet, aber über vorgegebene Frequenzbereiche aufsummiert, bevor das Tempo geschätzt wird. Klapuri *et al.* sehen hierin den



Abbildung 3.2: Aufbau der Akzentextraktion aus den Audiodaten nach [32]

Vorteil, Harmoniewchsel auf schmalen Bändern zu erkennen, aber die Analyse der Periodizität auf breiteren Frequenzbändern durchzuführen und betrachtet daher sein Verfahren als Kompromiss zwischen Schreier und Goto *et al.* [32]. Da die Akzentanalyse einen fundamentalen Bestandteil dieser Arbeit ausmacht, soll hier das Verfahren zur Berechnung des bandweisen Akzentsignals beschrieben und erläutert.

Das Audiosignal liegt zunächst mit einer Samplingrate von 44100 Hz in 16 Bit vor und wird normalisiert. Das Signal wird in Frames von je 23 ms aufgeteilt, das entspricht 1024 Samples pro Frame (mit $\frac{1024}{44100} \approx 0.023$ ms). Mit 50% Überlappung (512 Samples) resultiert hieraus eine zeitliche Auflösung der aufeinanderfolgenden Frames von etwa 11,5 ms oder $\frac{44100}{512} \approx 86$ Hz.

Die Frames werden mit einer von-Hann- bzw. Hanning-Fensterfunktion berechnet, um unerwünschte Effekte bei der Fouriertransformation zu minimieren. Die Fensterfunktion ist wie folgt definiert:

$$w(n) = 0,5 \left(1 - \cos \left(\frac{2\pi n}{N-1} \right) \right) \quad (3.1)$$

Jeder Frame wird nun mittels diskreter Fouriertransformation in den Spektralbereich transformiert. Bei den oben angegebenen Daten beträgt die Auflösung im Frequenzbereich dann $\frac{44100}{1024} \approx 43$ Hz. Dann werden 36 Dreiecksfilter berechnet, die gleichmäßig auf einer *critical-band*-Skala zwischen 50 Hz und 20 kHz verteilt sind. Die Leistung des Bandes b in Frame k wird dann berechnet und in $x_b(k)$ festgehalten, wobei k der Index des jeweiligen Frames und $b = 1, 2, \dots, b_{max} = 36$ der Index des Bandes ist.

Das menschliche Gehör nimmt Unterschiede in der Intensität nicht absolut wahr, sondern relativ zur Intensität des Signals. In Anlehnung an diese Beobachtung normalisieren Klapuri *et al.* die Leistungsschwankungen mit der Leistung [32]. Dies führt zu folgender Normierung:

$$y_b(k) = \frac{\ln(1 + \mu x_b(k))}{\ln(1 + \mu)} \quad (3.2)$$

Dabei bestimmt μ den Kompressionsgrad zwischen annähernd linearer ($\mu < 0,1$) und annähernd logarithmischer Transformation ($\mu > 10^4$). Die Autoren verwenden im folgenden $\mu = 100$.

Die normalisierten Leistungen $y_b(k)$ werden auf die doppelte Auflösung interpoliert, indem einfach Nullen zwischen die Werte gesetzt werden. Somit verdoppelt sich die zeitliche Auflösung der ursprünglichen Frames und die Samplingrate ist nun $f_r = 2 \cdot 86$ Hz = 172 Hz. Das interpolierte Signal wird durch einen Butterworth-Tiefpassfilter sechster Ordnung mit kritischer Frequenz $f_{LP} = 10$ Hz geglättet. Das interpolierte, geglättete Signal sei nun $z_b(n)$.

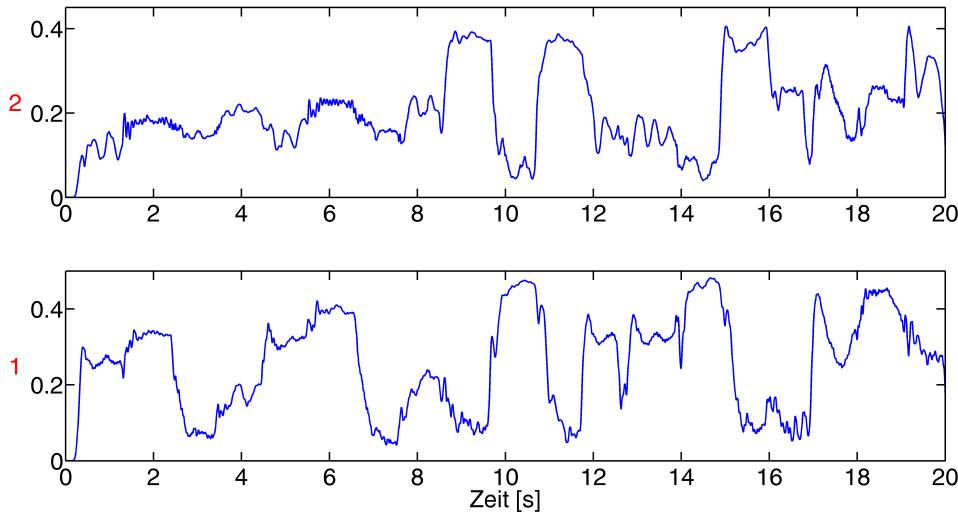


Abbildung 3.3: Die Leistungen der Bänder nach logarithmischer Normalisierung. Dargestellt werden zwei der 36 Frequenzbänder.

Nun folgt die Differenzierung des Signals, da die Momente betont werden sollen, in denen die Leistung steigt, und die Gleichrichtung („half-wave rectification“, HWR) des differenzierten Signals, die $z'_b(n)$ genannt wird:

$$z'_b = \text{HWR}(z_b(n) - z_b(n-1)) \quad (3.3)$$

wobei die Gleichrichtung HWR lediglich negative Werte auf 0 abschneidet:

$$\text{HWR}(x) = \max(x, 0) \quad (3.4)$$

Als abschließende Berechnung auf den einzelnen Bändern wird ein gewichteter Durchschnitt von $z_b(n)$ und $z'_b(n)$ gebildet,

$$u_b(n) = (1 - \lambda)z_b(n) + \lambda \frac{f_r}{f_{LP}} z'_b(n) \quad (3.5)$$

bei dem λ die Gewichtung des Differenzials im Verhältnis zum Signal $z_b(n)$ beschreibt und laut Aussage der Autoren im Bereich $0,6 \leq \lambda \leq 1,0$ gute Ergebnisse liefert. Der zusätzliche Faktor $\frac{f_r}{f_{LP}}$ soll ausgleichen, dass das Differenzial eines tiefpassgefiltertes Signals nur kleine Amplituden aufweist.

Wie oben ausgeführt, kombinieren Klapuri *et al.* die Ergebnisse auf den einzelnen, schmalen Bändern zu Akzentsignalen auf wenigen, breiten Frequenzbändern. Es werden jeweils m_0 benachbarte Bänder linear summiert, und so ergeben sich $c_0 = \lceil \frac{b_0}{m_0} \rceil$ Akzentsignale $v_c(n)$ auf den unterschiedlichen Frequenzbereichen $c = 1, \dots, c_0$

$$v_c(n) = \sum_{b=(c-1)m_0+1}^{cm_0} u_b(n) \quad (3.6)$$

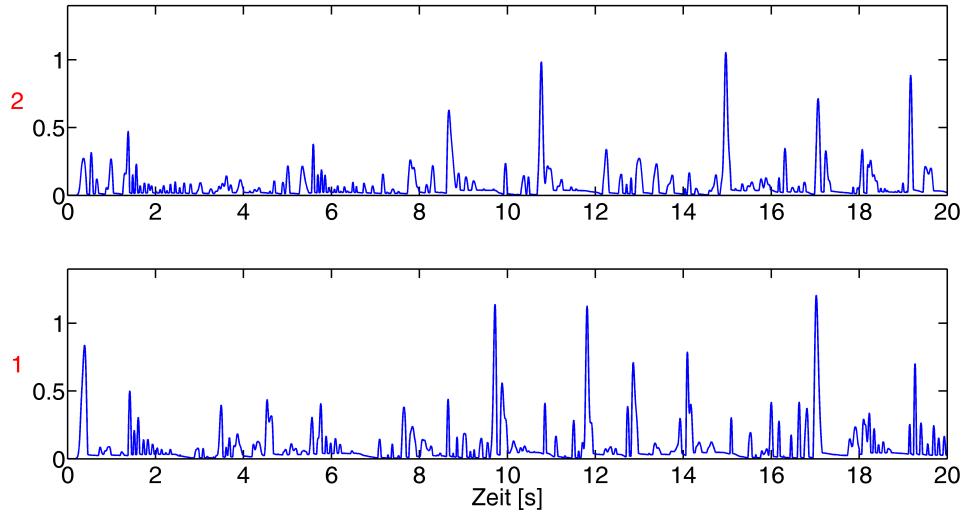


Abbildung 3.4: Die Bänder nach der Berechnung der gewichteten Summe $u_b(n)$ der Leistung und der differenzierten Leistung.

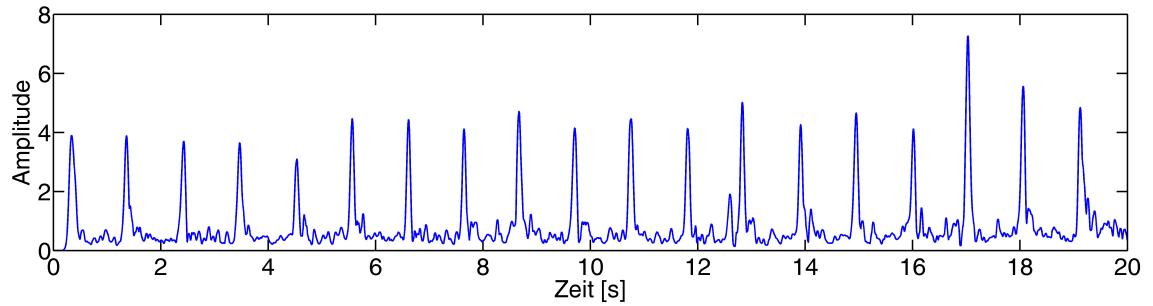


Abbildung 3.5: Die Summe der Bänder zeigt deutlich die Akzentuierung über die Zeit.

Die Autoren verwenden das vorgestellte Verfahren zur Akzentanalyse mit $b_0 = 36$ und $m_0 = 9$. Die ursprünglich 36 Bänder werden zur weiteren Verarbeitung auf 4 Akzentsignale reduziert. Auf die Weiterverarbeitung zur Tempoerkennung in [32] wird später eingegangen.

3.2 Akzentanalyse auf Chromamerkmalen

Klapuri *et al.* haben in obigem Ansatz viele enge Frequenzbänder genutzt, um Harmoniewechsel in den musikalischen Akzent einfließen zu lassen. Da Harmoniewechsel gerade bei der Tempoerkennung langsamer Musik eine so wichtige Rolle spielen, ist es naheliegend, das *Chroma* auf den einzelnen Frames zu untersuchen.

3.2.1 Chroma

Das Chroma beschreibt die zwölf Halbtonklassen (A, A#, B, C, C#, D, D#, E, F, F#, G, G#) der tonalen Musik. Nach Shepard [50] unterscheidet der Mensch bei der Wahrneh-

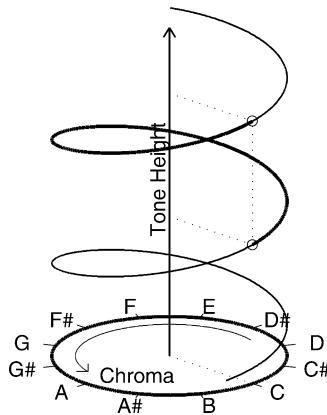


Abbildung 3.6: Die Shepard-Helix beschreibt die Wahrnehmung der Tonhöhe, entnommen aus [3, S. 97]. Die Tonhöhe steigt vertikal über den Chromaklassen an.

mung eines Tons Tonhöhe und **Chroma**. Die Abbildung 3.6 illustriert den Zusammenhang zwischen Tonhöhe und Chroma. Steigt die Tonhöhe, wie dargestellt, von C auf C', werden auf dem Weg entlang der Helix alle Halbtonklassen durchlaufen.

Die Chromaanalyse ordnet im *Chromavektor* den Chromaklassen die Energie zu, mit der sie in dem betreffenden Spektrum vorhanden sind. Die Tonhöhe einer Note wird in ihre Grundfrequenz f umgerechnet. Im MIDI-Format wird die Tonhöhe einer Note durch die Notennummer dargestellt, einer Zahl zwischen 0 und 127, bei der 69 dem Kammerton A entspricht. Geht man nun von einer gleichmäßig temperierten Stimmung aus, errechnet sich die Grundfrequenz f_n für jede MIDI-Notennummer n durch

$$f_n = 2^{\frac{n-69}{12}} f_A \quad (3.7)$$

mit $f_A = 440$ Hz als klassischer Grundfrequenz für den Kammerton A. Umgekehrt liefert

$$n = 69 + \text{round}\left(12 \log\left(\frac{f}{f_A}\right) / \log(2)\right) \quad (3.8)$$

die Notennummer n einer Frequenz f , wobei *round* auf die nächste ganze Zahl runden [16, S. 58].

Hat man die Energie für die Töne über mehrere Oktaven bestimmt, werden äquivalente Halbtöne der Oktaven zusammengerechnet. Die Beschränkung auf die zwölf Halbtöne als Chromaklassen ist nicht zwingend. Die Halbtöne können zur Steigerung der Auflösung im Chromabereich weiter unterteilt werden. Wird beispielsweise jeder Halbton in drei Segmente unterteilt, ergeben sich 36 Chromaklassen zu einer Oktave.

3.2.2 Einfacher Chromaakzent

Eronen beschreibt zunächst eine simple Methode zur Berechnung der Chromamerkmale: Nach einer STFT wird die Energie jeder Frequenz des Spektrums auf die zu der Frequenz

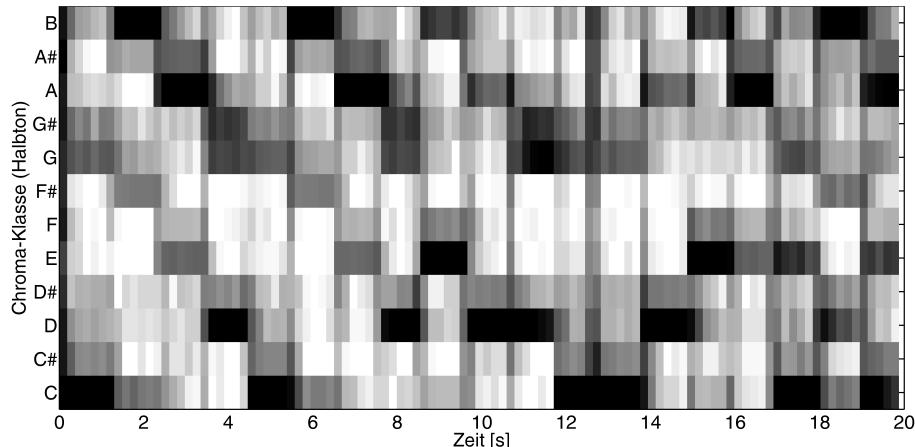


Abbildung 3.7: Das Chromagramm stellt dar, wie stark die Töne zur jeweiligen Zeit im Frequenzspektrum auftreten. Die Framelänge beträgt hier 8192 Samples (186 ms).

korrespondierende Halbtonklasse addiert. Hierzu wird ein relativ großes Analysefenster benötigt, um die Frequenzen der tiefen Töne genau genug aufzulösen [16, S. 58]. Eine Länge von 186 ms liefert beispielsweise eine Genauigkeit von 5,38 Hz.

Eronen und Klapuri vergleichen in [17] die Genauigkeit dieses simplen Chromaakkzents bei der Tempoerkennung mit der frequenzbandbasierten Akzentanalyse aus [32]. Die Chromavektoren werden ähnlich über die Zeit interpoliert, tiefpassgefiltert und gewichtet differenziert wie die Frequenzbänder. Es zeigt sich jedoch, dass die Tempoerkennung mit dem simplen Chromaakzent signifikant schlechter ist.

3.2.3 Chromaanalyse durch Grundfrequenzschätzung

An dieser Stelle soll gezeigt werden, was damit gemeint ist, dass die Tonhöhe einer Note zu ihrer Grundfrequenz korrespondiert. Klapuri definiert die **Tonhöhe** („*pitch*“) als ein durch unsere Sinne wahrnehmbares Attribut, das es ermöglicht, Sounds auf einer Frequenzskala von tief bis hoch einzurordnen. Genauer gesagt definiert sich die Tonhöhe als die Frequenz einer Sinuskurve, die vom menschlichen Zuhörer mit dem Sound in (wortwörtlichen) Einklang gebracht wird. Die **Grundfrequenz** (**F0**, „*fundamental frequency*“) ist die messbare physikalische Größe der Tonhöhe und nur für (annähernd) periodische Sounds definiert. Für diese Sounds ist F0 definiert als das Inverse der Periodendauer und steht in engem Zusammenhang mit der empfundenen Tonhöhe [30, S. 8].

Klänge mit wahrnehmbarer Tonhöhe

Natürliche musikalische Klänge bestehen gewöhnlich aus einer Vielzahl von Frequenzkomponenten. Wie oben beschrieben, beschreibt die Grundfrequenz F0 die wahrgenommene Tonhöhe und ist somit die dominierende Frequenz. Die relative Amplitude der Obertonbestandteile und ihre zeitliche Entwicklung bestimmen hingegen die Klangfarbe, das *Timbre*,

des Sounds. Untersucht man die Frequenzen der Obertonbestandteile und ihr Verhältnis zur Grundfrequenz, kann man Instrumente in zwei Gruppen von Klängen einteilen:

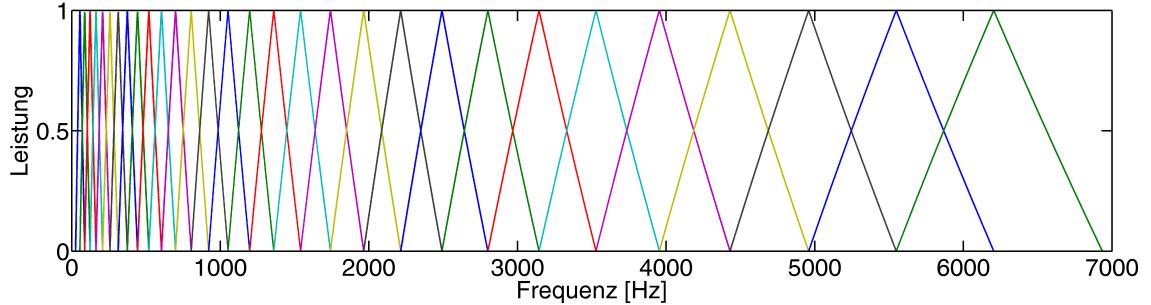
- Harmonische Klänge
 - Chordophone: Streicher (Violine...), Zupfinstrumente (Gitarre...), Anschlag (Klavier)
 - Aerophone: sämtliche Holzbläser (Luft, Blatt, Doppelrohr) und Blechbläser (mit und ohne Ventile)
- Nicht harmonische Klänge
 - Idiophone: Schlaginstrumente (Xylophon, Glockenspiel), Gezupft (Kalimba)
 - Membranophone: Pauke

Harmonische Klänge prägen oft die langsame Musik und haben eine besondere spektrale Struktur. Die dominanten Frequenzkomponenten, **harmonische Obertöne** (*harmonics*) genannt, treten annähernd äquidistant auf. Der Abstand zwischen den harmonischen Obertönen entspricht in etwa der Grundfrequenz, die als das Inverse der Periodendauer definiert ist. Bei einem vollkommen harmonischen Klang würde man die Obertöne genau an den ganzzahligen Vielfachen der Grundfrequenz finden. In der realen Welt sind harmonische Klänge aber nicht immer völlig harmonisch, da reale Chordophone von der Steifigkeit von materiellen Saiten betroffen sind. Diese Disharmonie zeigt sich beispielsweise bei Klavieren.¹

Wesentlich ist, dass die Bestandteile harmonischer Klänge nicht als einzelne Töne wahrgenommen werden, sondern als ein Ton mit einer Tonhöhe, F0, und einer charakteristischen Klangfarbe, die sich aus den Obertönen zusammensetzt. Daher ist es für eine genaue Analyse des Chromas wichtig zu entscheiden, ob lokale Maxima im Frequenzspektrum die Grundfrequenz eines Tons darstellen oder dessen harmonische Obertöne [29, S. 231f].

¹Klapuri führt in [29], worauf diese Abhandlung über harmonische Klänge beruht, genauer aus, wie die harmonischen Obertöne für Chordophone berechnet werden können und geht auch auf die spektrale Struktur von nicht harmonischen Klängen und deren F0-Schätzung ein. Interessant ist am Rande, dass die Prinzipien zur Schätzung der Grundfrequenz eines Klangs ganz ähnliche Methoden aufgreifen wie die Temposchätzung. Das Problem scheint gewisse Ähnlichkeiten aufzuweisen: Es geht um die Schätzung der dominanten Periode.

Unterschiedlich ist hier lediglich die Skalierung der zeitlichen Dimension: Die Grundfrequenz umfasst Bereiche von vielleicht 60 Hz bis 8 kHz. Ein Beispiel: Bei $F_0 = 290$ Hz dauert die Periode 3,4 ms und wird in einem Ausschnitt von 30 ms deutlich sichtbar. Das Tempo eines Musikstücks liegt gewöhnlich zwischen 45 und 240 Bpm. Das entspricht einem Bereich 0,75 Hz bis 4 Hz. Ein Tempo von 80 Bpm (= $1,\overline{3}$ Hz) würde mit einer Periodendauer von 750 ms korrespondieren. Ins Verhältnis zur F0-Periode gesetzt müsste der Ausschnitt also 6,6 Sekunden lang sein um ein ähnliches Bild zu geben. Bei weiterem Interesse zu diesem Thema sei die Lektüre von [29] empfohlen.

Abbildung 3.8: Antworten der Filterbank $H_b(k)$

Grundfrequenzschätzung als Summe über harmonische Bestandteile

Eine Möglichkeit, das Chroma von harmonischen Klängen genauer abzuschätzen besteht darin, die harmonischen Obertöne gewichtet zu summieren. Die Stärke oder *salience* eines Kandidaten τ für die F0-Periode lässt sich bei einer Abtastrate f_s durch

$$s(\tau) = \sum_{m=1}^M g(\tau, m) |Y(f_{\tau,m})| \quad (3.9)$$

abschätzen. Hierbei bezeichnet $f_{\tau,m} = m f_s / \tau$ die Frequenz des m -ten harmonischen Obertons des F0-Kandidaten f_s / τ und $Y(f)$ ist das bereinigte Frequenzspektrum eines Frames. Das Gewicht des m -ten Obertons wird durch die Funktion $g(\tau, m)$ in Abhängigkeit von τ approximiert.

Bereinigung des Spektrums Als Vorbereitung wird das Frequenzspektrum von Klangfärbungen bereinigt. Dieser als „whitening“ bezeichnete Schritt macht die Schätzung unabhängiger von der Art der Klangquelle, indem die Standardabweichung der spektralen Energie geschätzt und das Spektrum invers gefiltert wird.

Klapuri [31] verwendet zur Bereinigung eine einfache und allgemein verwendbare Methode im Frequenzbereich:

1. Die DFT $X(k)$ wird für das Signal $x(n)$ auf einem Frame mit Hanning-Fenster berechnet, der zuvor mit Nullen auf die doppelte Länge erweitert wurde.
2. Eine Bandpass-Filterbank wird auf den Frequenzbereich angewendet. Die Filterbank besteht aus Dreieckfiltern, deren Zentrum c_b die Frequenz des kritischen Bandes $b+1$ ist:

$$c_b = 229 \cdot (10^{(b+1)/21.4} - 1) \quad (3.10)$$

Bei c_b ist die Filterantwort $H_b(c_b) = 1$ und fällt in beiden Richtungen linear zu $H_b(c_{b-1}) = H_b(c_{b+1}) = 0$ ab. Außerhalb dieses Bereichs gilt $H_b = 0$ und es ergibt sich das in der Grafik 3.8 dargestellte Ergebnis für $b = 1, \dots, 30$.

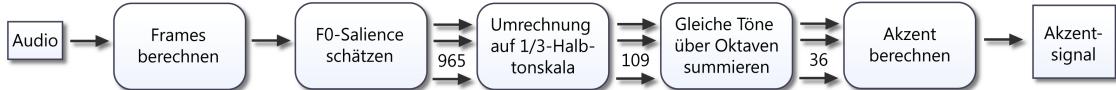


Abbildung 3.9: Aufbau der Extraktion des Chromaakzents aus den Audiodaten nach [17]

3. Die Standardabweichungen σ_b auf den Teilbändern werden berechnet:

$$\sigma_b = \left(\frac{1}{K} \sum_k H_b(k) |X(k)|^2 \right)^{\frac{1}{2}} \quad (3.11)$$

4. Für jedes Band wird der Kompressionskoeffizient $\gamma_b = \sigma_b^{\nu-1}$ mit $\nu = 0,33$ bestimmt, der die Stärke der Bereinigung angibt.
5. Das bereinigte Spektrum $Y(k) = \gamma(k)X(k)$ wird berechnet, indem das ursprüngliche Spektrum mit den Kompressionskoeffizienten gewichtet wird. Hierzu wird lineal zwischen zwei Mittelfrequenzen interpoliert, um für alle Frequenzen k Kompressionskoeffizienten zu erhalten.

Berechnung der Stärke eines Kandidaten Zur praktischen Anwendung approximiert man $s(\tau)$ auf einem diskreten, bereinigten Spektrum $Y(k)$ durch

$$\hat{s}(\tau) = \sum_{m=1}^M g(\tau, m) \max_{k \in \kappa_{\tau, m}} |Y(k)|. \quad (3.12)$$

Hierbei ist $\kappa_{\tau, m}$ die Menge der Frequenzkomponenten, die zu dem m -ten harmonischen Oberton von τ gehören:

$$\kappa_{\tau, m} = [\text{round}\left(\frac{mK}{\tau + \Delta\tau/2}\right), \dots, \text{round}\left(\frac{mK}{\tau - \Delta\tau/2}\right)], \quad (3.13)$$

wobei auch $\text{round}(x)$ wieder das Runden auf die nächste ganze Zahl beschreibt und $\Delta\tau$ bestimmt, wie viele umliegende Frequenzkomponenten einbezogen werden. $\Delta\tau = 0,5$ ist in der Regel ausreichend [31].

Klapuri zeigt, dass sich als Gewichtung $g(\tau, m)$ für die harmonischen Anteile von F0 die Funktion

$$g(\tau, m) = \frac{f_s/\tau + \alpha}{m f_s/\tau + \beta} \quad (3.14)$$

eignet. Die Parameter sind mit $\alpha = 52$ Hz und $\beta = 320$ Hz für Analysefenster von 93 ms Länge angegeben [31].

3.2.4 Grundfrequenzbasierter Chromaakzent

Aus den Grundfrequenzkandidaten berechnen Eronen und Klapuri gemäß [17] Chroma-merkmale und verarbeiten sie zu einem musikalischen Akzent. Das Audiosignal liegt in

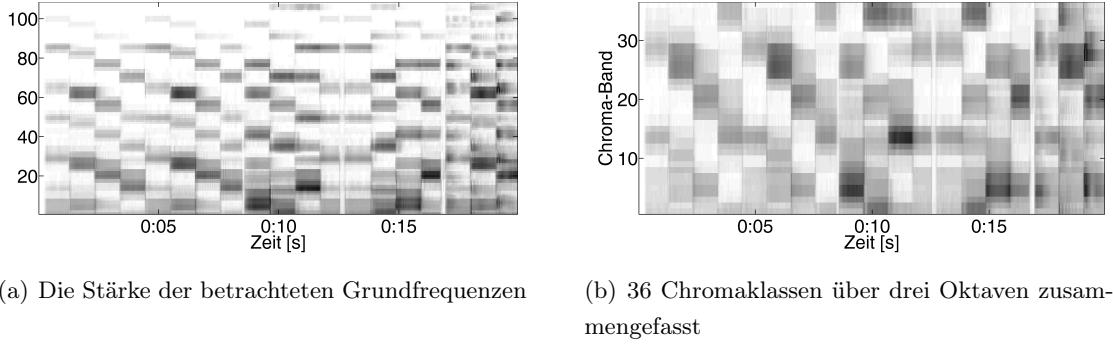


Abbildung 3.10: Grundfrequenzbasierte Chromaanalyse im Zeitverlauf

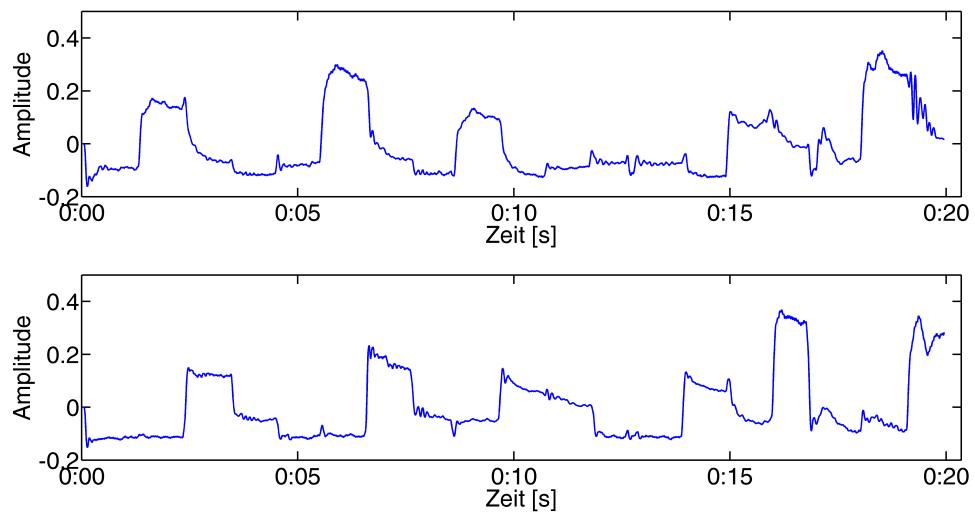


Abbildung 3.11: Die interpolierten Signale $z_b(n)$; dargestellt sind $b = 25$ (oben) und $b = 20$ (unten).

44,1 kHz mit 16 Bit pro Sample vor und wird in Frames von je 93 ms (also 4096 Samples) und 50% Überlappung eingeteilt.

Auf den Frames wird die Stärke für jeden Grundfrequenz als gewichtete Summe der Amplituden der harmonischen Obertöne berechnet. Die verwendeten Grundfrequenzen liegen zwischen 80 und 640 Hz (also über drei Oktaven) und werden mit einer Auflösung von drei Werten pro Halbton (36 Werte pro Oktave) transformiert. Die Abbildung 3.10(a) stellt die Stärke der Tonhöhen dar.

Die Werte gleicher Chromaklassen werden über die drei Oktaven aufsummiert, so dass ein 36-dimensionalen Chromavektor $x_b(k)$ entsteht. Der Index $b = 1, \dots, b_0 = 36$ steht für die Chromaklasse, und k ist der Index des Frames. Das Ergebnis ist in Abbildung 3.10(b) visualisiert. Anschließend wird die Matrix $x_b(k)$ normalisiert, indem der Mittelwert entfernt und die Standardabweichung jedes Chromakoeffizienten über die Zeit normalisiert wird. Man erhält die normalisierte Chromamatrix $\hat{x}_b(k)$.

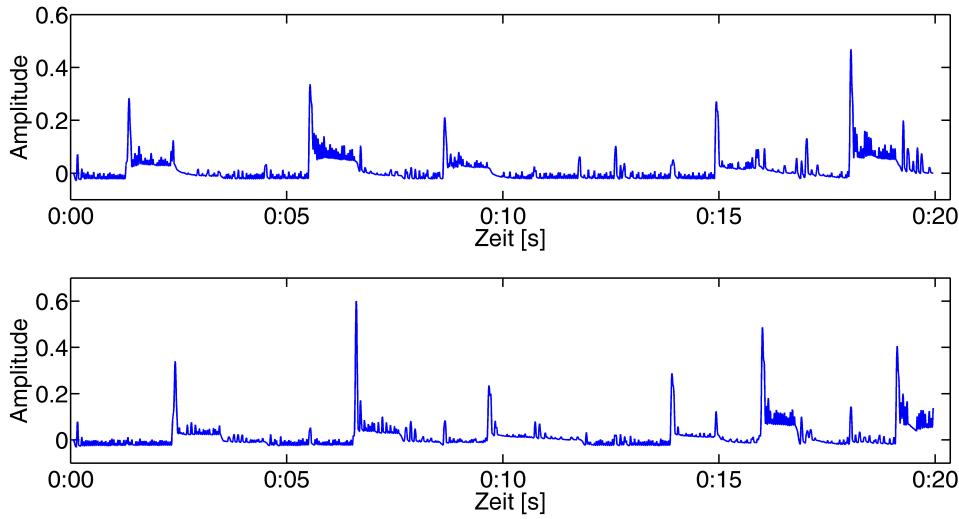


Abbildung 3.12: Gewichtet differenzierte Signale $u_b(n)$

Die weitere Verarbeitung der Chromaklassen zu einem Akzentsignal orientiert sich an der Akzentberechnung auf Frequenzbändern [32] (Abschnitt 3.1 in diesem Kapitel). Die Hüllkurven der Chromaklassen werden durch eingefügte Nullen auf das Achtfache gestreckt. Dadurch wird auch hier eine Abtastrate f_r von 172 Hz erreicht. Dann folgt eine Tiefpassfilterung durch einen Butterworth-Filter sechster Ordnung mit $f_{LP} = 10$ Hz als Grenzfrequenz. Das gefilterte und interpolierte Signal sei nun $z_b(n)$ und ist für zwei Chromaklassen in Abbildung 3.11 veranschaulicht.

Als nächstes wird $z_b(n)$ gleichgerichtet und gewichtet differenziert durch

$$z'_b(n) = \text{HWR}(z_b(n) - z_b(n - 1)), \quad (3.15)$$

und $\text{HWR}(x) = \max(x, 0)$ setzt auch hier negative Werte einfach auf Null. Anschließend können $z_b(n)$ und $z'_b(n)$ gewichtet werden zu

$$u_b(n) = (1 - \lambda)z_b(n) + \lambda \frac{f_r}{f_{LP}} z'_b(n). \quad (3.16)$$

Auch hier bestimmt $0 \leq \lambda \leq 1$ das Verhältnis zwischen $z_b(n)$ und seiner Ableitung $z'_b(n)$ und f_r/f_{LP} gleicht die kleinen Werte der Ableitung aus, die der Tiefpassfilterung geschuldet werden.

Schlussendlich werden alle $u_b(n)$ linear gemittelt und ergeben das in Abbildung 3.13 illustrierte Akzentsignal

$$a(n) = \frac{1}{b_0} \sum_{b=1}^{b_0} u_b(n), \quad (3.17)$$

das den Grad an musikalischem Akzent als Funktion in der Zeit beschreiben soll. Im folgenden Kapitel wird erklärt, wie in [17] aus diesem Akzentsignal die Periodizität und das Tempo bestimmt wird.

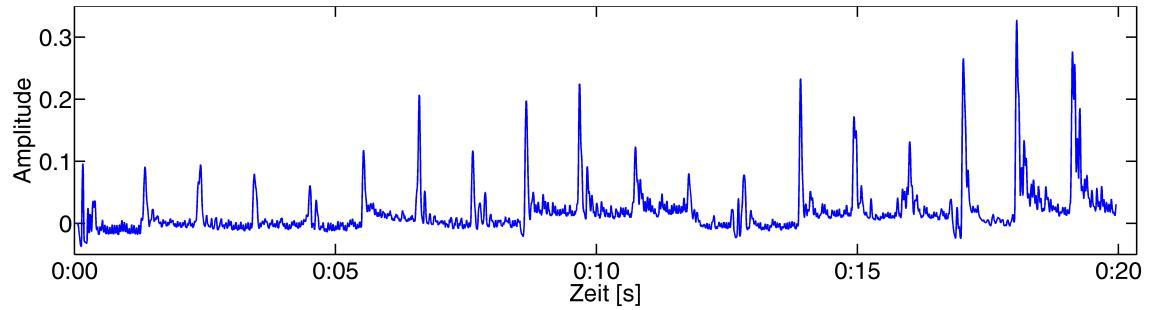


Abbildung 3.13: Der Mittelwert aller Chromaklassen bildet das Akzentsignal $a(n)$

Eronen und Klapuri vergleichen die Genauigkeit der Tempoerkennung auf Grundlage dieses Merkmals mit der frequenzbandbasierten Akzentanalyse [32] und erzielen damit bessere Ergebnisse. Diese sind signifikant besser als bei einer Analyse mit einfachem Chromaakzent und liefern aufgrund der erkannten Harmoniewechsel gute Ergebnisse bei langsamer Musik [17].

Dieses Kapitel hat einige Methoden aufgeführt, wie aus Audiosignalen die musikalische Betonung – der „*phenomenal accent*“ – kontinuierlich in der Zeit analysiert werden kann. Genau wie bei der Onset-Analyse hat die Genauigkeit großen Einfluss auf die Tempoerkennung, auf die nun eingegangen werden soll.

Kapitel 4

Verfahren zur Tempoerkennung

Nachdem bis hierher die musikalischen Grundlagen von Tempo und Metrik gelegt, das Problem der Tempoerkennung klar definiert und ein Einstieg in die Signalverarbeitung gegeben wurde, werden nun einige bekannte Verfahren zur Tempoerkennung vorgestellt. Stephen Hainsworth liefert in [25] einen Überblick über verschiedene Verfahren in diesem Aufgabenbereich - auch Verfahren für Aufgaben, die über die reine Tempoerkennung hinausgehen. Eronen gibt einen ähnlichen Überblick und ergänzt die von Hainsworth dargestellten Verfahren [16, Kapitel 3, S. 45ff]. Ansatz und Zielsetzung hinter den Verfahren sind sehr unterschiedlich. Hainsworth gruppiert daher die Verfahren nach folgenden Aspekten:

- Art der Eingabedaten
- Zielsetzung des Verfahrens
- Algorithmischer Ansatz, die Idee
- Kausalität
- Art der Ausgabe

Die Art der Eingabedaten – Audiosignale – und die Methoden, mit denen aus ihnen Merkmale erzeugt werden können, wurden bereits in den Kapiteln 2 und 3 behandelt.

Die Zielsetzung des Verfahrens legt fest, was mit dem Verfahren modelliert werden soll. So sind insbesondere frühe Arbeiten musikpsychologisch geprägt in dem Versuch, die menschliche Wahrnehmung von Musik nachzuahmen. Andere Verfahren stützen sich direkt auf das Signal und nicht so sehr auf die menschliche Wahrnehmung. Im Zusammenhang mit der Tempoerkennung bedeutet dies, dass die Zielsetzung bei den psychoakustischen Verfahren das Erkennen des *gefühlt* Tempos ist – das Tempo, das der *Zuhörer* subjektiv wahrnimmt. Im Gegensatz dazu haben die anderen Verfahren das Ziel, das objektiv *gewollte* Tempo zu erkennen – das Tempo, mit dem der *Komponist* ein Stück gekennzeichnet hat.

Wenn Hainsworth nach der kausalen Operation („*causal operation*“, Hainsworth stützt sich auf [47]) unterscheidet, bezieht er sich darauf, worauf sich die Schätzung der Metrik

in dem Modell stützt. Wie bereits erwähnt, basiert die Schätzung des Metrums zur Zeit t bei einem kausalen Modell nur auf Daten (x_1, \dots, x_t) , die bereits vergangen oder gerade aktuell sind. Nicht-Kausale Modelle dagegen können sich bei ihren Schätzungen auch auf zukünftige Daten stützen – ihnen ist das komplette Audiosignal des Musikstücks bekannt.

Die Art der Ausgabe, der „*output*“ des Verfahrens, wird von Hainsworth in die bereits im Abschnitt 1.3 dargelegten Aufgaben der Tempoerkennung unterteilt. Diese drei Aspekte der Gruppierung sind bereits durch die Zielsetzung dieser Diplomarbeit festgelegt: Gesucht ist ein Verfahren, das das objektive Tempo eines komplett vorliegenden Musikstücks möglichst genau erkennt. Damit bleibt die Frage offen, welche algorithmischen Ansätze bisher für die Metrikerkennung eingesetzt wurden.

4.1 Regelbasierte Verfahren

Regelbasierte Verfahren sind die ersten Verfahren, die in der Tempoerkennung und Metrikanalyse angewendet wurden, insbesondere als Computer noch nicht in der Lage waren, aufwändige Algorithmen auf Audiodaten auszuführen. Daher sind regelbasierte Verfahren oft simpel und beschreiben sinnvolle musiktheoretische Regeln. In der Tabelle 4.1 werden einige Arbeiten aufgeführt. Steedman beschreibt eines der ersten Modelle der Rhythmusanalyse auf symbolischen Daten und versucht, die rhythmische Struktur von Bachs „Wohltemperiertem Klavier“ zu bestimmen. Longuet-Higgins und Lee verwenden zur Beat- und Metrikbestimmung psychologisch begründete Regeln. Parncutt entwickelt ein Modell für phänomenale Akzente und wendet darauf Beaterkennung an. Außerdem bevorzugt er explizit die mittlere Tempoebenen [25].

Da die hier angesprochenen regelbasierten Verfahren auf symbolischen Daten arbeiten, eignen sie sich nur bedingt zur Tempoerkennung aus Audiosignalen und sind hier der Vollständigkeit halber aufgeführt.

4.2 Autokorrelation

Autokorrelation ist eine Methode zur Bestimmung von Wiederholungen - und deren Abstand (Periode) - in Daten und wurde daher in etlichen Studien eingesetzt [25]. Sie liefert eine Schätzung für die Periode, mit der sich das Signal wiederholt und ist auf einem Frame der Länge W und dem Akzentsignal $a(n)$ definiert als

$$\rho(\tau) = \sum_{n=0}^{W-1} a(n)a(n-\tau), \quad (4.1)$$

wobei $0 \leq \tau \leq W - 1$ die Verschiebung („*lag*“) angibt [16, S. 45]. Die Autokorrelation liefert hohe Werte $\rho(\tau)$ für diejenigen Verschiebungen τ , die der Periodendauer im Signal entsprechen. An diesen Stellen ist das Signal sich selbst ähnlich. Daher korrespondiert

Ansatz	Autor und Jahr [Zitat]	Eingabe	Ausgabe
regelbasiert	Parncutt 1994 [40] Eck 2000 [13]	S S	Beat, Takt Tempo
Autokorrelation	Brown 1993 [5] Seppänen <i>et al.</i> 2006 [48] Alonso <i>et al.</i> 2007 [2] Davies & Plumley 2007 [10] Ellis 2007 [15] Peeters 2007 [41]	S A A A A A	Tempo, Takt Tatum, Beat Tempo Beat Beat Tatum, Beat, Takt
Oszilliernde Filter	Large 1994 [33] McAuley 1995 [38] Scheirer 1998 [45] Toivainen 1998 [54] Eck 2001 [14]	S S A S A	Beat Beat Beat Beat Beat
Histogramm	Seppänen 2001 [47] Wang und Vilermo 2001 [56] Gouyon <i>et al.</i> 2002 [22] Jensen und Andersen 2003 [28] Uhle und Herre 2003 [55]	A A A A A	Tatum, Beat Beat Tatum Beat Tatum, Tempo, Takt
Multiple Agenten	Allen und Dannenberg 1990 [1] Rosenthal 1992 [44] Goto <i>et al.</i> 1999 [18], 2001 [19] Dixon 2007 [12]	S S A A	Beat Beat, Takt Beat, Takt Beat
Probabilistisch	Laroche 2001 [34], 2003 [35] Cemgil <i>et al.</i> 2000-03 [7], [8], [9] Raphael 2001 [43] Hainsworth und Macleod 2003 [26] Klapuri <i>et al.</i> 2006 [32] Shiu und Kuo 2008 [51]	A A A/S A A A	Beat Beat Transkription Beat Tatum, Beat, Takt Beat
Regression	Seyerlehner <i>et al.</i> 2007 [49] Eronen und Klapuri 2010 [17]	A A	Tempo Tempo

Tabelle 4.1: Übersicht über einige Ansätze zur Tempo- und Metrikerkennung, basierend auf [25, S. 104] und [16, S. 47]. Bei der Eingabe steht „A“ für Audiosignale und „S“ für symbolische Daten.

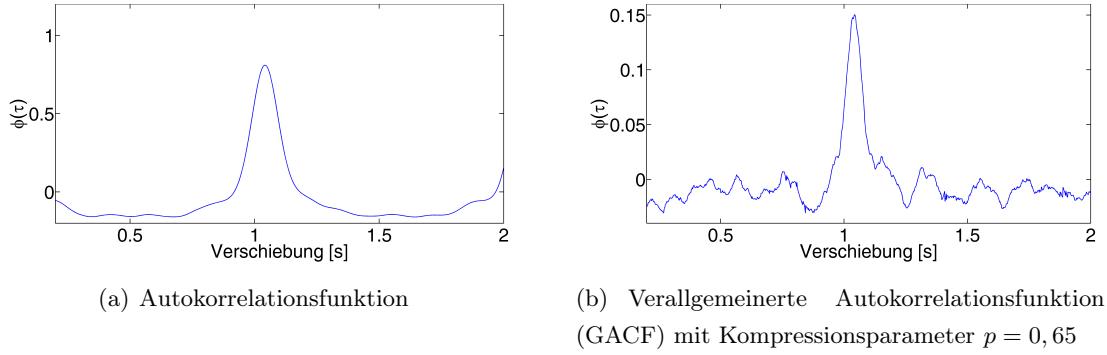


Abbildung 4.1: Die Autokorrelation bestimmt die Periodizität eines Signals. Die Verschiebung τ wird hier auf dem Bereich von 0,2 s bis 2 s dargestellt.

die Verschiebung τ , an der $\rho(\tau)$ maximal wird, mit der Zeitspanne der Periodendauer auf einer metrischen Ebene. Nicht immer ist der Beat, sondern kann auch eine andere metrische Ebene sein, insbesondere bei ausgeprägtem Tatum [25]. Das erklärt, warum dann ganzzahlige Vielfache des wirklichen Tempos geschätzt werden.

Die Autokorrelation kann nur Informationen über die mögliche Periode des Signals liefern. Man kann mittels Autokorrelation das Tempo bestimmen – will man jedoch die genaue zeitliche Position der Beats ermitteln, ist eine Schätzung der Phase erforderlich. Daher schätzt Ellis die Periode der Beats zuerst mittels Autokorrelation, und verwendet dann zur Bestimmung der einzelnen Beats dynamische Programmierung. [15]. Ist man jedoch nur am Tempo interessiert, bietet die Autokorrelation eine gute und leicht verständliche Möglichkeit, die Periode der Beats zu bestimmen.

4.2.1 Generalisierte Autokorrelation

In dem Zusammenhang ist interessant, dass die Autokorrelation die Kreuzkorrelation eines Signals mit sich selbst darstellt. Die Fouriertransformation hat die Eigenschaft, eine Kreuzkorrelation (\otimes) im Zeitbereich auf eine Multiplikation im Frequenzbereich abzubilden. Sei $A(k) = \text{DFT}_a(k)$ die Frequenzdarstellung von $a(n)$. Dann entspricht die Multiplikation $A^*(k) \cdot A(k) = |A(k)|^2$ im Frequenzbereich der Berechnung von $\rho(\tau) = a^*(n) \otimes (n)$ im Zeitbereich. Hierbei steht der * für *komplex konjugiert* und kann im reellen Zeitbereich für a^* vernachlässigt werden [6, S. 57]. Die inverse Fouriertransformation überträgt das Ergebnis abschließend in den Zeitbereich.

Die **generalisierte Autokorrelation (GACF)**, „generalized autocorrelation function“ verallgemeinert diese Tatsache und definiert sich durch $|A(k)|^p$, sodass sich für $p = 2$ die bekannte Autokorrelation ergibt. Eronen und Klapuri verwenden die GACF, um die Periode der Akzentsignale zu schätzen [17] und definieren sie als

$$\rho_m(\tau) = \text{IDFT}(|\text{DFT}(a_m)|^p). \quad (4.2)$$

Das m steht dabei für den m -ten Frame und a_m ist der Eingangsvektor, der bei einem Frame der Länge W mit Nullen auf die Länge $2W$ erweitert wird:

$$a_m = [a((m-1)W), \dots, a(mW-1), 0, \dots, 0] \quad (4.3)$$

Dies ist erforderlich, da die DFT nur $W/2$ relevante Werte liefert. Über den Parameter p lässt sich die Kompression im Frequenzbereich beeinflussen. Schaut man sich die Vorgehensweise genauer an, sieht man, dass Eronen und Klapuri die Periodenanalyse auf die grundfrequenzbasierte Frequenzanalyse abstimmen. Für diese Akzentmerkmale liefert $p = 0.65$ die besten Ergebnisse [17].

Das Ziel ist, einen einzigen Wert für das Tempo des gesamten Musikstücks zu schätzen. Daher wird aus den Periodenvektoren der Frames ein einziger Periodenvektor $\rho_{med}(\tau)$ berechnet, der die Periodenverteilung des Stücks repräsentiert. $\rho_{med}(\tau)$ wird dabei als punktweiser Median der Periodenvektoren über die Zeit ermittelt. Die Voraussetzung ist nahezu konstantes Tempo des betrachteten Musikstücks. Anschließend wird $\rho_{med}(\tau)$ normalisiert, da mit ansteigendem τ für die Analyse im Frame weniger Werte zur Verfügung stehen:

$$\hat{\rho}_{med}(\tau) = \frac{1}{N - \tau} \rho_{med}(\tau) \quad (4.4)$$

Der normalisierte Periodenvektor wird in [17] auf Verschiebungen zwischen 0,06 s und 2,2 s beschränkt, das sind umgerechnet 1000 Bpm und 27,3 Bpm. Bei einer Abtastrate von $f_r = 172$ Hz¹ entsprechen 0,06 s in etwa 10 Samples und 2,2 s 374 Samples. Eine Fenstergröße von $W = 512$ genügt, um den gewünschten Rahmen an Verzögerungen abzudecken.

Der so beschränkten Vektor wird zu $s(\tau)$ normalisiert. Dabei wird der Mittelwert entfernt und die Standardabweichung vereinheitlicht, um die Periodenvektoren verschiedener Stücke vergleichen zu können.

4.3 Histogrammbasierte Verfahren

Histogrammbasierte Verfahren ähneln in ihrer Idee den autokorrelationsbasierten Verfahren. Das Histogramm ist das Pendant der Autokorrelation – auf diskreten Eingabedaten statt auf kontinuierlichen Signalen. Daher geht diesen Verfahren eine Onset-Erkennung voraus (siehe Abschnitt 2.3), die diskrete Onsets aus dem Audiosignal extrahiert. [25]

Die Abstände zwischen den Onsets, die so genannten „*inter-onset intervals*“ (IOI), bilden die Grundlage der Histogramme. Die Abstände von aufeinander folgenden Onsets werden als Intervalle erster Ordnung („*first-order intervals*“) bezeichnet und korrespondieren mit dem Tatum als kleinstem musikalischen Puls. Zur Bestimmung höherer metrischer Ebenen wie der Tempoebene untersucht man daher Intervalle jeglicher Ordnung (*all-order intervals*) und bezieht damit die Abstände von weiter entfernten Onsets in die Betrachtung ein [25].

¹Die Abtastrate ist im Abschnitt 3.2.4 über die chromabasierte Akzentanalyse beschrieben.

Seppänen [46] definiert Inter-Onset-Intervalle o_i mit $i = 1, 2, \dots$ und unterteilt die Zeitachse der IOI in J Bereiche.

$$h(j) = \text{count}(i, |o_i - u(j + 0, 5)| < 0.5u) \quad (4.5)$$

ist dann die Anzahl der IOIs, die in den Bereich j fallen, und u steht für die Breite des Bereichs [25]. Das Histogramm nutzt Seppänen, um das Tatum des Signals zu extrahieren. Darauf aufbauend berechnet Seppänen weitere Merkmale, die dann als Eingabe einer Mustererkennung zur Bestimmung höherer metrischer Ebenen wie Beat und Takt dienen.

Gouyon *et al.* [22] verwenden eine Kombination aus Onset-Erkennung und IOI-Histogramm und bestimmen das Tatum als den höchsten Peak – *tick* genannt. Darauf bauen sie verschiedene Anwendungen auf [25]. Auch Wang und Vilermo [56], Uhle und Herre [55] sowie Jensen und Andersen [28] setzen histogrammbasierte Variationen für verschiedene Anwendungen ein.

4.4 Oszillatoren

Es gibt zwei unterschiedliche Ansätze, mittels Oszillatoren die Metrik zu analysieren [25]. Der erste Ansatz versucht, einen adaptiven Oszillator mit dem Eingangssignal zu stimulieren und in der Frequenz des Beats schwingen zu lassen. Vorreiter auf dem Gebiet ist die Arbeit von Large [33], der auf symbolischen Eingangsdaten den Beat von Klavierstücken nachvollziehen will. Die Schwingungsperiode und Phase des Oszillators wird dabei durch Impulse beeinflusst, die mit den Onset-Zeiten der Noten korrespondieren.

McAuley [38] und Toiviainen [54] präsentieren ähnliche Modelle, wobei Toiviainen eine live gespielten Performance interaktiv begleiten will [16].

Der zweite und vielleicht bedeutendere Ansatz ist die Verwendung einer Bank von Kammfilter-Resonatoren, deren feste Verzögerungen den Bereich der zu erwartenden Beatperioden abdecken. Er geht auf Scheirer [45] zurück, der eine der ersten erfolgreichen Methoden zur Beaterkennung aus Audiosignalen implementiert [25]. Scheirer splittet das Eingangssignal in sechs Teilbänder und berechnet die Leistung auf den Bändern über die Zeit. Anschließend wird für jedes Teilband eine Filterbank aus 150 Kammfiltern berechnet, die logarithmisch zwischen 60 und 240 Bpm verteilt sind. Der Filter r^* , der am deutlichsten auf das Signal anspringt, beschreibt dann die Periode des Tempos – oder ein Vielfaches des Tempos [16, S. 49f]. Vom internen Zustand der Verzögerung von r^* kann auf die Phase der Beats geschlossen werden, daher ist das Verfahren zum kausalen Beat-Tracking gut geeignet. Nur die Komplexität der Berechnung – im Beispiel Scheirers 6 Bänder zu je 150 Resonatoren – steht Echtzeitanforderungen entgegen.

Ein Kammfilter mit Verzögerung τ berechnet sich für die Eingabe $v_c(n)$ auf Teilband c durch

$$r_c(\tau, n) = \alpha_\tau r_c(\tau, n - \tau) + (1 - \alpha_\tau)v_c(n). \quad (4.6)$$

Hierbei ist $r_c(\tau, n - \tau)$ das Feedback mit der Verzögerung τ – die wieder eingespielte Ausgabe des Kammfilters vor τ Zeiteinheiten. Die Gewichtung $\alpha_\tau = 0.5^{\tau/T_0}$ beschreibt die Verstärkung des Feedbacks und nimmt exponentiell ab. Die Verstärkung fällt im Zeitraum T_0 auf die Hälfte ab, und Scheirer verwendet hierfür umgerechnet 1,5-2 s [45].

Klapuri *et al.* greifen das Verfahren von Scheirer auf und setzen die Halbwertzeit auf 3s, also $T_0 = 3[\text{s}] \cdot f_r$. So kann schnell genug auf Tempoänderungen reagiert werden. Gleichzeitig werden Periodenlängen von bis zu vier Sekunden erkannt [32]. Verwendet werden für τ Werte zwischen 1 und τ_{max} , wobei $\tau_{max} = 688$ einer Verzögerung von vier Sekunden entspricht.

Die momentane Energie jedes Kammfilters im Teilband c zur Zeit n berechnet sich durch

$$\hat{r}_c(\tau, n) = \frac{1}{\tau} \sum_{i=n-\tau+1}^n r_c(\tau, i)^2, \quad (4.7)$$

und wird danach in Bezug auf die Energie des Eingangssignals und die gesamte Leistung des Kammfilters zu $s_c(\tau, n)$ normiert. Die Funktion

$$s(\tau, n) = \sum_{c=1}^{c_0} s_c(\tau, n) \quad (4.8)$$

stellt die Auffälligkeit der unterschiedlichen metrischen Ebenen zur Zeit n dar und ergibt sich aus c_0 Teilbändern.

4.5 Agentensysteme

Die Idee bei diesem Ansatz besteht darin, verschiedene Agenten ihre eigene Hypothese zum Beat und Tempo unabhängig voneinander verfolgen zu lassen [25]. Jeder Agent wird danach bewertet, wie gut seine Hypothese mit den beobachteten Daten übereinstimmt. Unpassende Hypothesen können verworfen werden, während zutreffende Hypothesen aufgeteilt und weiter verfeinert werden können. Dieser Ansatz bietet hervorragende Möglichkeiten zur kausalen Beatverfolgung, da man zu jeder Zeit analysieren kann, welcher Agent die höchste Bewertung erhält. Allerdings zeigt sich das Verfahren etwas sprunghaft, wenn es zwischen zwei Hypothesen wechselt. Möchte man nur das Tempo des Signals wissen, kann man am Ende die Hypothese mit der höchsten Wertung übernehmen.

Erste Ansätze dieser Art stammen von Allen und Dannenberg [1] und Rosenthal [44] und basieren auf symbolischen Daten. Goto und Dixon haben später Agentensysteme vorgestellt, die auf Audiodaten operieren.

Goto hat verschiedene Verfahren zur Beatverfolgung publiziert [21], [18], [19]. Diesen Verfahren gemein ist der grundsätzliche Prozess: Zuerst werden Onset-Komponenten auf unterschiedlichen Frequenzbereichen von Onset-Findern bestimmt. Diese werden dann an Agenten weitergegeben, die auf den Onset-Komponenten die Intervalle zwischen den Beats

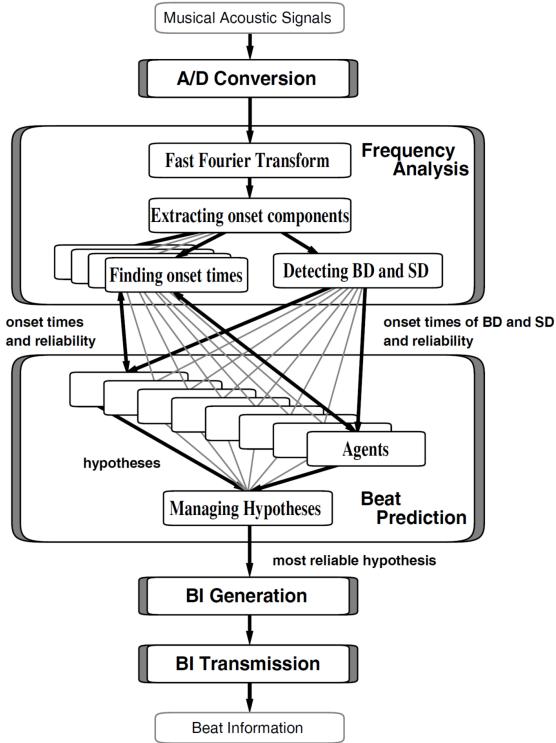


Abbildung 4.2: Übersicht des Beattracking-Systems (BTS) nach Goto, entnommen aus [20, S. 367]

(IBI für „*inter-beat interval*“) bestimmen und eine Prognose für den nächsten Beat liefern. Anfänglich beschreibt Goto das Verfahren mit 15 Onset-Findern, die Onsets an 15 Paare von Agenten übermitteln. Die beiden Agenten eines Paars vertreten die gleiche Hypothese für das IBI und das Tempo, aber der zweite Agent ist um die Hälfte versetzt. So sollen robustere Ergebnisse erzeugt werden, wenn anfänglich ein Beat falsch klassifiziert wird. Zur Berechnung ihrer Hypothese bestimmen die Agenten das Histogramm auf den Inter-Onset-Intervallen der Onsets (siehe Abschnitt 4.3). Außerdem versuchen sie die Art des Beats zu bestimmen, indem sie ein bestimmtes Schlagmuster für das Schlagzeug zugrunde legen [20], [21].

In späteren Arbeiten untersucht Goto Audiosignale, in denen das Schlagzeug keine bedeutende Rolle spielt und nutzt Harmoniewechsel als Basis der Onseterkennung [18]. Er stellt später ein Verfahren vor, das beide Ideen zusammenführt und sowohl Harmoniewechsel als auch – sofern vorhanden – Schlagmuster zur Klassifikation der Beats benutzt. Hierbei verwendet er Autokorrelation zur Bestimmung der vergangenen IBI und eine Kreuzkorrelation zur Vorhersage des nächsten IBI. [19]

Dixon hat ein Verfahren namens BeatRoot entwickelt [12]. Es basiert auf einer Agentenarchitektur, die Hypothesen darstellt, erweitert, bewertet und die lokalen Tempohypothesen entsprechend den Onsetzeiten und deren Intensität anpasst [25]. Während die

ersten Versionen auf MIDI-Daten gute Ergebnisse liefern, verwendet Dixon nun eine auf Spektralmerkmalen basierende Onset-Erkennung, die das Verfahren zur Verarbeitung von Audiodaten befähigt [16, S. 51].

4.6 Probabilistische Verfahren

Gemäß Hainsworth liegen Agentensystemen und probabilistischen Ansätzen ähnliche interne Modelle zugrunde. Allerdings betrachten sich die einzelnen Agenten unabhängig von den anderen, während probabilistische Modelle Verteilungen aller Parameter verwalten und optimieren, um zur besten Hypothese zu gelangen [25]. Probabilistische Verfahren definieren explizit ein Modell des Prozesses, der dem Metrum zugrunde liegt. Ziel der Verfahren ist die Schätzung der Parameter dieses Prozesses.

Die Idee ist, dass dem Metrum ein Prozess zugrunde liegt, der eine Sequenz von Zuständen (q_0, q_1, q_2, \dots) durchläuft und dabei eine Sequenz von Beobachtungen (s_0, s_1, s_2, \dots) erzeugt. Die Beobachtungen liegen als Onset-Zeiten oder Periodenvektoren vor. Cemgil und Kappen haben hierzu ein lineares, dynamisches System zur Beatverfolgung beschrieben [9], das von Shui und Kuo [51] sowie von Hainsworth und Macleod [26] verwendet wird. Da die Formulierungen in den Arbeiten und in dem Überblick von Hainsworth [25] und Eronen [16, S. 51ff] nicht einheitlich sind, wird das System hier in einer Schreibweise vorgestellt, die möglichst konsistent mit dem Rest des Kapitels ist.

Der Beat-Prozess ist als lineares, dynamisches System modelliert:

$$q_{n+1} = \Phi(n+1|n)q_n + \epsilon_n \quad (4.9)$$

$$s_n = M(n)q_n + v_n \quad (4.10)$$

Hier ist q_n die Zustandsvariable zum Zeitpunkt n und s_n ist die Beobachtung. Die Terme ϵ_n und v_n modellieren das Rauschen (den Fehler) des internen Prozesses, beziehungsweise den Messfehler der Beobachtung. Ein Zustand q_n des Systems definiert sich durch

$$q_n = [\phi_n, \tau_n]^T. \quad (4.11)$$

ϕ_n beschreibt die Phase – die genaue zeitliche Position – und τ_n die Periode des Beats zur Zeit n . Die Prognose für die Zeit des nächsten Beats lässt sich damit abschätzen durch

$$\phi_{n+1} = \phi_n + \tau_n, \quad (4.12)$$

wobei die Periode konstant und somit $\tau_{k+1} = \tau_n$ ist. Als Übergangsmatrix der Zustände ergibt sich daraus

$$\Phi(n+1|n) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \quad (4.13)$$

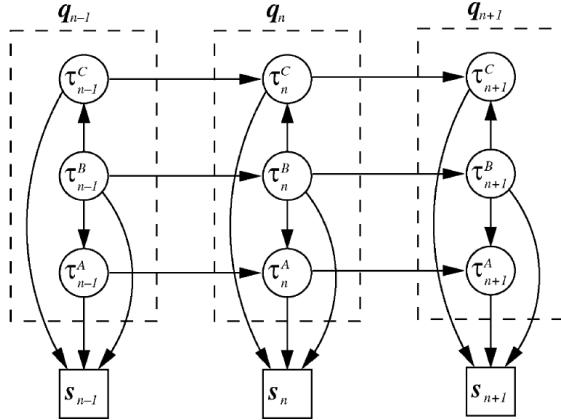


Abbildung 4.3: Probabilistisches Hidden-Markov-Modell zur Tempoerkennung, entnommen aus [32, S. 347]

Beobachtet man nur die Onsets, ist

$$M(n) = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad (4.14)$$

denn dann ist die Beobachtung s_n der n -te beobachtete Onset. Der Onset ist im Modell als ϕ_n in q_n definiert. Den Beat verfolgen kann man nun, indem man die Sequenz der internen Zustände bei gegebenen Beobachtungen in der Zeit von 0 bis n schätzt. Shiu und Kuo setzen hierfür Kalman-Filter ein [51], während Hainsworth und Macleod auf die statistische Methode der Partikel-Filterung zurückgreifen [16, S. 52].

Während die oben genannten Modelle auf diskreten Onsets beruhen, entwickeln Klapuri *et al.* in [32] ein probabilistisches Modell auf kontinuierlichen Daten. Die Beobachtungen beruhen einer Kammfilterbank, die in Abschnitt 4.4 erläutert wurde. Hierbei wird der Metrik-Prozess als verstecktes Markovmodell (HMM für „hidden Markov model“) beschrieben, das in der Abbildung 4.3 visualisiert ist und näher erläutert werden soll.

Die sichtbaren Beobachtungen sind die augenblicklichen Energien der Kammfilter $s(\tau, n)$ zur Zeit n und werden mit s_n bezeichnet. Sie werden durch den Zustand q_n eines unsichtbaren Prozesses des HMM erzeugt. Hierbei setzt sich ein Zustand q_n aus drei Variablen zusammen:

τ_n^A , die Tatum-Periode

τ_n^B , die Beat-Periode

τ_n^C , die Takt-Periode

Klapuri *et al.* nennen diesen Prozess ein zeithomogenes Markovmodell erster Ordnung, und definieren:

$P(q_1)$ als initiale Verteilung der Zustände

$P(q_n|q_{n-1})$ als Zustandsübergangsmatrix (hierbei wird die Wahrscheinlichkeit angegeben, mit der man bei gegebenem Zustand q_{n-1} im nächsten Zeitschritt zu q_n übergeht)

$p(s_n|q_n)$ als Verteilung der Beobachtungen s_n für einen gegebenen Zustand q_n

Damit lässt sich die gemeinsame Verteilung einer Zustandsfolge $Q = (q_1, q_2, \dots, q_N)$ und einer Folge von Beobachtungen $O = (s_1, s_2, \dots, s_N)$ beschreiben:

$$p(Q, O) = P(q_1)p(s_1|q_1) \prod_{n=2}^N P(q_n|q_{n-1})p(s_n|q_n) \quad (4.15)$$

Klapuri et al. zerlegen $P(q_n|q_{n-1})$ und unterstellen einige Annahmen zur Abhängigkeit der Variablen, die für unterschiedliche metrische Ebenen des Prozesses stehen:

$$P(q_n|q_{n-1}) = P(\tau_n^B|q_{n-1})P(\tau_n^A|\tau_n^B, q_{n-1})P(\tau_n^C|\tau_n^A, \tau_n^B, q_{n-1}) \quad (4.16)$$

Musikalisch sinnvoll ist die Annahme, dass das Tatum keine zusätzlichen Informationen für die Taktperiode liefert, wenn der Beat bekannt ist:

$$P(\tau_n^C|\tau_n^A, \tau_n^B, q_{n-1}) = P(\tau_n^C|\tau_n^B, q_{n-1}) \quad (4.17)$$

Weiter wird angenommen, dass der Beat zum Zeitpunkt n , τ_n^B , nur vom Beat τ_{n-1}^B abhängt. Die beiden anderen Variablen von q_{n-1} haben keinen Einfluss auf den Beat. Setzt man diese Annahmen bezüglich des vorhergehenden Zustands für Tatum und Takt bei gegebenem τ_n^B analog um, so kann man $P(q_n|q_{n-1})$ schreiben als

$$P(q_n|q_{n-1}) = P(\tau_n^B|\tau_{n-1}^B)P(\tau_n^A|\tau_n^B, \tau_{n-1}^A)P(\tau_n^C|\tau_n^B, \tau_{n-1}^C). \quad (4.18)$$

Unter denselben Annahmen gilt

$$P(q_1) = P(\tau_1^B)P(\tau_1^A|\tau_1^B)P(\tau_1^C|\tau_1^B), \quad (4.19)$$

und die bedingten Wahrscheinlichkeiten modellieren die Abhängigkeiten von Tatum und Takt vom Beat. Die optimale Folge von Zuständen bei gegebenen Beobachtungen s_n wird durch den Viterbi-Algorithmus ermittelt, der die verschiedenen metrischen Ebenen in der Modellierung in ihrer Gesamtheit optimiert.

Bei den probabilistischen Verfahren ist auch der Ansatz von Raphael [43] erwähnenswert, der die simultane Entwicklung von drei Prozessen in einem HMM abbildet:

Rhythmus Ein unsichtbarer Prozess mit Zuständen S_0, S_1, \dots, S_N , der die Position einer Note innerhalb eines Taktes modelliert

Tempo Ein unsichtbarer Prozess, der die Verbindung zwischen den Notenlängen des unsichtbaren Rhythmusprozesses und den beobachteten Notenlängen darstellt. Er schätzt die Länge eines Taktes in Sekunden in den Zuständen T_0, T_1, \dots, T_N .

Beobachtung Beobachten lassen sich die internen Prozesse an einer Folge von Notenlängen, die auf den Inter-Offset-Intervallen von MIDI-Daten beruhen. Die Beobachtungen sind abhängig vom unsichtbaren internen Zustand der beiden obigen Prozesse. Die Abhängigkeit beschreibt Rafael durch

$$Y_n = l(S_{n-1}, S_n)T_n + \epsilon_n, \quad (4.20)$$

wobei $l(S_{n-1}, S_n)$ den Abstand zweier Noten – und damit die Länge der Note S_{n-1} – misst, T_n die Dauer des Taktes zur Zeit n angibt und ϵ_n wie üblich einen Fehlerterm beschreibt.

Führt man vorher eine Onset-Erkennung der Audiodaten aus, lassen sich die IOI der erkannten Onsets als Beobachtung nutzen [16, S. 53].

4.7 Regression

Diese Kategorie von auf Regression beruhenden Verfahren findet sich noch nicht in Hainsworths Überblick über Verfahren zur Tempoerkennung [25], sondern wird erst von Eronen [16, S. 53f] hinzugefügt. Sie geht auf die Arbeit von Seyerlehner *et al.* [49] zurück, die eine k -Nearest Neighbor (**k -NN**) Klassifikation vorschlagen als Alternative zur fehleranfälligen Bestimmung des Tempos durch die Wahl des korrekten Maximums der Periodenfunktionen. Grundlegend ist die Beobachtung, dass ein ähnliches Tempo zu ähnlichen Periodenfunktionen führt, wie die Abbildung 4.4 darstellt². Daher suchen sie zu einem unbekannten Stück die k ähnlichsten Periodenfunktionen der Trainingsdaten und geben das Tempo an, dass am häufigsten auftritt.

Obwohl das Verfahren, so wie es von Seierlehner umgesetzt wurde, keine signifikanten Erfolge erzielte, greifen Eronen und Klapuri die Idee auf und entwickeln einige Verbesserungen [17]. Daher wird dieser Ansatz hier detailliert vorgestellt.

Bei gegebener Periodenbeobachtung $s(\tau)$ soll das Tempo T aus einem kontinuierlichen Wertebereich geschätzt werden.³ Mittels k -NN Regression wird das Tempo als gewichteter Median der Tempi der nächsten Nachbarn berechnet⁴. Zur Berechnung der Distanzen zu den Trainingsbeispielen wird die Euklidische Distanz berechnet.

Als erste Verbesserung beschreibt [17] einen Resampling-Schritt. Motiviert ist das Resampling dadurch, dass Distanzmaße wie die Euklidische Distanz empfindlich reagieren, wenn die Peaks der Beobachtung $s(\tau)$ nicht genau genug zu einem Trainingsbeispiel

²Die Abbildung ist ein kleiner Vorgriff auf Kapitel 7.

³Die Berechnung von $s(\tau)$ wird in Abschnitt 4.2.1 unter dem Stichwort „Generalisierte Autokorrelation“ dargestellt.

⁴In [17] wird ausgeführt, dass das Ergebnis der Klassifikation normalerweise durch die Bildung des Durchschnitts der k nächsten Nachbarn gebildet wird. Der gewichtete Median liefert aber deutlich bessere Ergebnisse als der gewichtete Mittelwert.

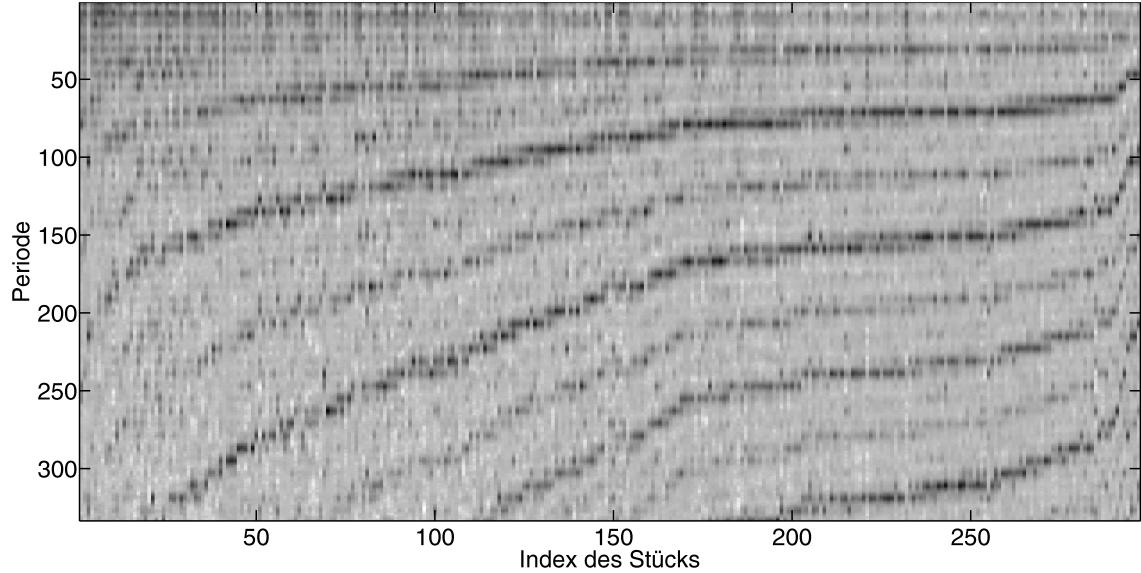


Abbildung 4.4: Das ansteigende Tempo der Stücke zeigt sich an den Maxima der Periodenfunktionen.

passen. Daher wird die Beobachtung $s(\tau)$ gestaucht (oder gedehnt), um einem Trainingsbeispiel mit anderem Tempo ähnlicher zu sein.

Aus einem Beobachtungsvektor $s(\tau)$ mit dem unbekannten Tempo T werden die gestauchten Testvektoren $s_r(\tau)$ erzeugt, wobei r das Verhältnis angibt, um dass $s(\tau)$ gestaucht wurde. Zu einem Testvektor $s_r(\tau)$ korrespondiert dann das unbekannte Tempo T/r . Verwendet werden 15 Werte für r , die sich gleichmäßig zwischen 0,87 und 1,15 aufteilen. Damit werden für ein Stück mit dem Tempo $T = 120$ Bpm gestauchte Testvektoren erzeugt, deren Tempi im Bereich von 104 bis 138 Bpm liegen.

Danach wird die Euklidische Distanz zwischen den Testvektoren $s_r(\tau)$ und den Trainingsvektoren $t_m(\tau)$ berechnet. Sind M Trainingsdaten vorhanden, so ergeben sich die Distanzen zu den Trainingsvektoren mit $m = 1, \dots, M$ als:

$$d(m, r) = \sqrt{\sum_{\tau} (t_m(\tau) - s_r(\tau))^2} \quad (4.21)$$

Mit dieser Distanzmatrix d wird für jedes Trainingsbeispiel t_m bestimmt, welches Verhältnis $\hat{r}(m)$ beim Resampling zur kleinsten Distanz $d(m)$ zwischen einem Testvektor und t_m führt. Berechnet werden daher

$$d(m) = \min_r d(m, r) \quad (4.22)$$

$$\hat{r}(m) = \operatorname{argmin}_r d(m, r) \quad (4.23)$$

Die k nächsten Nachbarn der Beobachtung $s(\tau)$ sind die Trainingsbeispiele $t_{m_i}(\tau)$ mit $i = 1, \dots, k$, die zu den k kleinsten Werten $d(m_i)$ gehören. Das Tempo $T(m_i)$, mit dem

der i -te nächste Nachbar t_{m_i} gekennzeichnet ist, ist eine Schätzung für das Tempo des Testvektors $s_{\hat{r}(m_i)}(\tau)$. Da das unbekannte Tempo des Testvektors T/r ist, ergibt sich das geschätzte Tempo der ursprünglichen Beobachtung $s(\tau)$ für den i -ten Nachbarn t_{m_i} durch

$$\hat{T}(i) = T(m_i)\hat{r}(m_i) \quad (4.24)$$

An dieser Stelle liegen k Schätzungen $\hat{T}(i)$ für das Tempo T der Beobachtung $s(\tau)$ vor. Die finale Schätzung des Tempos wird durch den gewichteten Median der geschätzten Tempi $\hat{T}(i)$ mit den Gewichten ω_i für die k nächsten Nachbarn berechnet:

$$\omega_i = \frac{\exp(-\gamma d(m_i))}{\sum_{i=1}^k \exp(-\gamma d(m_i))} \quad (4.25)$$

Durch die Gewichtung haben die Trainingsbeispiele, die näher am Testvektor liegen, größeren Einfluss auf das geschätzte Tempo. Der Parameter γ bestimmt, wie stark das Gewicht mit ansteigender Distanz d abnimmt, und Eronen und Klapuri erzielen mit $\gamma = 40$ die besten Resultate.

4.8 Vergleich der Verfahren

Erinnert man sich an die Einführung (Kapitel 1.1) zurück, muss ein System zur Tempoerkennung drei Probleme lösen. Die Verfahren in diesem Kapitel gehen zwei unterschiedliche Probleme an: das *Messen* und die *Interpretation* der Perioden.

Autokorrelation, Histogramm und Oszillatoren berechnen auf unterschiedliche Weise die Perioden der metrischen Impulse, die in den Eingabedaten „versteckt“ sind. Sie alle erzeugen im Prinzip einen Vektor, der angibt, wie stark eine Verzögerung τ (man kann τ genausogut Verschiebung oder IOI nennen, prinzipiell sind die Begriffe hier synonym für die Periode) in dem Signal auftritt. Klapuri *et al.* merken an, dass unterschiedliche Periodenschätzer nach gründlicher Optimierung ähnlich gute Ergebnisse liefern. Sie kommen zu dem Schluss, dass die Frage nach dem Messen der Periode im Vergleich zur Merkmalsextraktion und anschließenden Interpretation nur eine untergeordnete Rolle spielt [32, S. 345]. So bilden diese drei Verfahren also die grundlegenden Werkzeuge zur Messung von Perioden, die in der einen oder anderen Form Bestandteil eines Systems zur Tempoerkennung sind.

Die Frage der Interpretation bleibt damit aber offen, denn nicht immer korrespondiert der höchste Wert zu dem τ , das das Tempo des Signals angibt. Das Problem besteht also darin, aus den Perioden die richtige auszuwählen. Zunächst müssen nur solche Perioden betrachtet werden, die zu einem musikalisch sinnvollen Tempo (z.B. zwischen 45 Bpm und 240 Bpm) korrespondieren. Doch auch in diesem Bereich kann es mehr als einen dominanten Wert geben. Man könnte im Vorfeld untersuchen, wie sich im Allgemeinen Musikstücke auf die Tempi verteilen und die Ergebnisse der Periodenschätzung damit gewichten. Agentensysteme versuchen hier, mehrere Hypothesen unabhängig voneinander parallel zu verfolgen,

während probabilistische Verfahren Modelle für den internen Tempoprozess aufstellen, um so musikalische Zusammenhänge explizit zu modellieren. Einen interessanten Ansatz zur Interpretation liefern Eronen und Klapuri mit der überarbeiteten k -NN-Regression, mit der sie die Interpretation der Perioden in das Feld der Lernverfahren führen.

Vergleich der Ergebnisse

Welches Verfahren liefert die besten Ergebnisse? Vergleicht man die unterschiedlichen Verfahren, kommt man um diese Frage nicht herum. So vergleicht Hainsworth in [25, S. 124ff] die Genauigkeit seines eigenen Verfahrens [24] mit den Verfahren von Scheirer [45] und Klapuri *et al.* [32]. Auf einer Datenbank von 222 Musikstücken von etwa je einer Minute Länge kommt er zu dem Ergebnis, dass [32] die besten Ergebnisse liefert.

Eine umfangreiche Untersuchung zur Genauigkeit der Tempoerkennung unternehmen Gouyon *et al.* [23]. Sie untersuchen 12 Verfahren auf einer gemeinsamen Musikdatenbank von 3199 Stücken. Das Maß an Genauigkeit ist die relative Anzahl der Stücke, für die das geschätzte Tempo höchstens 4% vom richtigen Tempo abweicht. Auch hier erzielt das Verfahren von Klapuri *et al.* die besten Ergebnisse. Außerdem wird darauf hingewiesen, dass bei den meisten Fehlern das geschätzte Tempo das Doppelte oder die Hälfte des tatsächlichen Tempos ist. Interessant ist, dass sich bei Klapuris und Scheirers Verfahren eine Tendenz zur Präferenz eines mittleres Tempo zeigt. Die Fehler entstehen, weil langsame Stücke doppelt und schnelle Stücke halb so schnell geschätzt werden. Alle anderen Verfahren zeigen hingegen die Tendenz, die Geschwindigkeit zu schnell einzuschätzen [23].

Um die Leistungsfähigkeit von regressionsbasierten Verfahren einordnen zu können, vergleichen Eronen und Klapuri ihr Verfahren [17] mit [32], dem von Ellis [15] und dem von Sepännen *et al.* [48]. Auf einer Testdatenbank mit 355 Musikstücken erreicht ihr Verfahren eine Genauigkeit⁵ von 79%, eine signifikante Verbesserung zu dem Verfahren aus [32], das mit 71% den zweiten Rang einnimmt. Somit erscheinen Lernverfahren als eine vielversprechende Idee für die Tempoerkennung.

⁵Hier ist wie bei Gouyon *et al.* eine Abweichung von 4% erlaubt.

Teil II

Tempoerkennung in der Praxis

Kapitel 5

Entscheidung für ein Verfahren

Nach dem langen Einstieg in die diversen Grundlagen wird nun das Problem der Tempoerkennung in der Praxis angegangen. Die Auswahl eines Verfahrens als Ausgangsbasis bildet dabei den ersten zentralen Schritt. Deshalb soll die Entscheidung auch gut fundiert sein, wozu die grundsätzliche Betrachtung der Verfahren in Kapitel 4 befähigt.

5.1 k -NN-Regression auf Chromaakzenten

Die praktische Umsetzung des Verfahrens wird sich auf den Ansatz der k -NN-Regression von Eronen und Klapuri [17] stützen. Dieser Überblick führt die Bereiche zusammen, die bereits in den Grundlagen detailliert abgehandelt wurden.

Chromabasierte Akzentanalyse Aus den Audiodaten werden zunächst die Chromamerkmale unter Berücksichtigung der Grundfrequenzen extrahiert. Aus der Entwicklung der einzelnen Chromamerkmale über die Zeit wird dann das Akzentsignal bestimmt (Abbildung 5.2(a)).

Generalisierte Autokorrelation Auf dem Akzentsignal wird auf Fenstern die generalisierte Autokorrelationsfunktion (GACF) berechnet, und der punktweise Median über

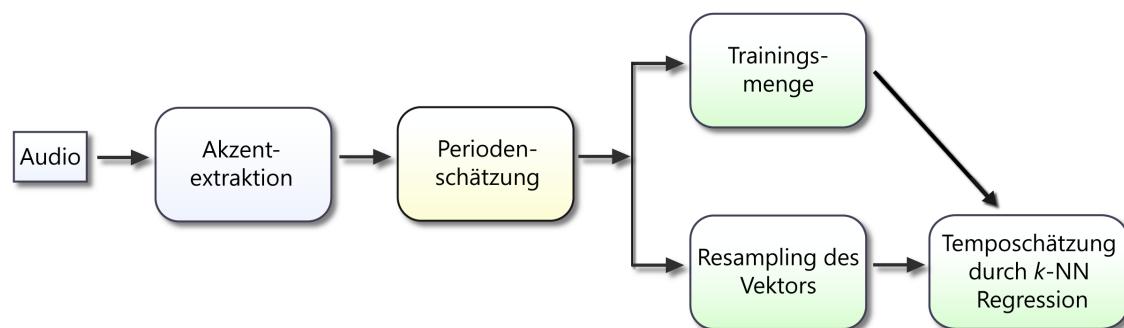
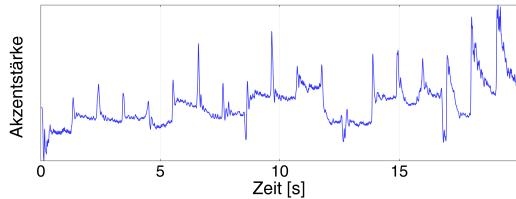
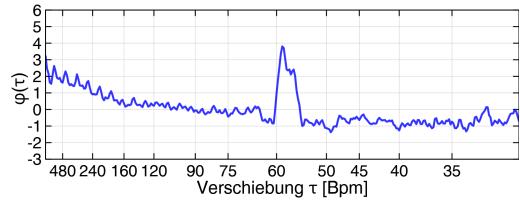


Abbildung 5.1: Überblick über das Verfahren von Eronen und Klapuri [17]



(a) Akzentsignal: Chromaakzent



(b) Periodenvektor: Die GACF des Chromaakzents

Abbildung 5.2: Das Akzentsignal und der Periodenvektor für das Verfahren von Eronen und Klapuri [17]

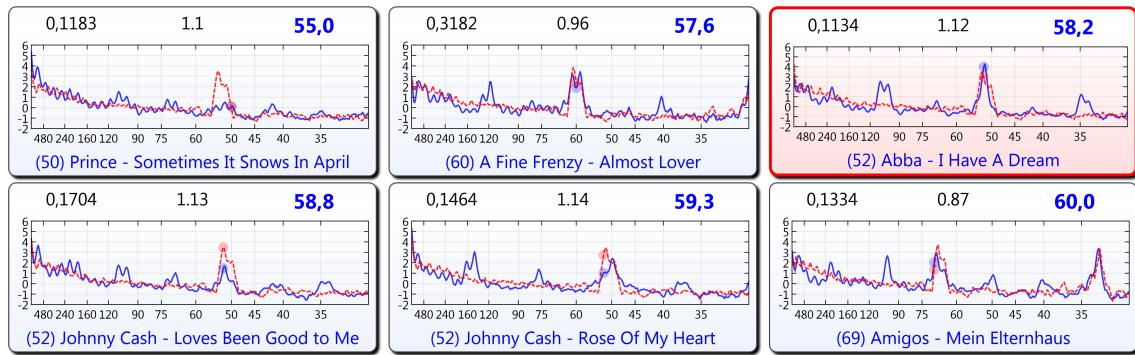


Abbildung 5.3: Klassifikation: Die $k = 6$ nächsten Nachbarn, in ansteigendem Tempo. Dargestellt ist jeweils der Periodenvektor des Trainingsbeispiels (blau) und der Periodenvektor des zu klassifizierenden Stücks nach dem Resampling (rot gestrichelt). Der Median bestimmt das geschätzte Tempo und ist hervorgehoben.

die Fenster als repräsentativer Periodenvektor des gesamten Musikstücks wird ermittelt (Abbildung 5.2(b)).

k -NN-Regression Aus dem Periodenvektor wird durch ein Lernverfahren auf einer Trainingsmenge das Tempo als gewichteter Median der gefundenen Nachbarn klassifiziert (Abbildung 5.3).

Bereits in Kapitel 4.8 hat ein Vergleich der Verfahren gezeigt, dass das Verfahren - nach Aussage der Autoren - gute Ergebnisse erzielt. Schon deshalb bildet es einen guten Ausgangspunkt für die praktische Umsetzung. Die folgenden Aspekte untermauern die Entscheidung für das Verfahren.

Tempoerkennung als explizites Ziel Im Gegensatz zu vielen in Kapitel 4 vorgestellten Verfahren besteht das Ziel von Eronen und Klapuri in der „Temposchätzung für Musikstücke mit annähernd konstantem Tempo“ [17, S. 50]. Damit entspricht das Ziel genau dem Anspruch dieser Arbeit.

	langsam	mittel	schnell
langsam	76%	16%	8%
mittel	4%	96%	0%
schnell	28%	14%	58%

Tabelle 5.1: Kategoriefehler bei der Klassifizierung in die Tempokategorien langsam (0 bis 90 Bpm), mittel (90 bis 130 Bpm) und schnell (ab 130 Bpm) für die k -NN-Regression, entnommen aus [17, S. 55]. Die Zeilen stehen für die tatsächliche Kategorie, die Spalten für die Schätzung.

	langsam	mittel	schnell
langsam	60%	30%	10%
mittel	1%	99%	0%
schnell	32%	24%	44%

Tabelle 5.2: Kategoriefehler bei der Klassifizierung in Tempokategorien für die Methode von Klapuri *et. al.* [32], entnommen aus [17, S. 55]. Die Zeilen stehen für die tatsächliche Kategorie, die Spalten für die Schätzung.

Verständlichkeit Die in [17] eingesetzten Methoden sind in den einzelnen Schritten leicht nachvollziehbar. Gerade die chromabasierte Merkmalsextraktion ist musikalisch plausibel, da sie Harmoniewchsel erkennt, die vor allem bei langsamer Musik eine große Rolle spielen.

Einige Verfahren benötigen explizite Annahmen über die Verteilung der Tempi, um die Interpretation der Periodenschätzung zu verbessern. Beispielsweise schätzen probabilistische Verfahren, wie das in [32], *a priori* die Verteilung der möglichen Tempi. Das k -NN-Verfahren stellt die Verteilung implizit durch die Tempoverteilung der Trainingsbeispiele bereit. Diese lässt sich bei Bedarf leicht an andere Bedürfnisse anpassen.

Klare Trennung der Probleme Bei diesem Ansatz sind die drei Problembereiche der Tempoerkennung klar erkennbar und voneinander getrennt. Eine klare Aufteilung ist aus softwaretechnischer Sicht vorteilhaft. Der Ansatz definiert klare Schnittstellen zwischen den einzelnen Schritten. Das bildet die Grundlage für eine Modularisierung des Verfahrens. Die speziellen Schritte lassen sich also durch andere Verfahren austauschen, die die gleiche Schnittstelle unterstützen. In [17] wird das genutzt, um unterschiedliche Verfahren zur Extraktion der Akzentmerkmale aus dem Audiosignal zu vergleichen. Analog ist das für die Periodenschätzung und die Klassifikation möglich.

Gute Klassifikationsergebnisse auf langsamen Musikstücken Die guten Ergebnisse liefern die Grundvoraussetzung für die weiteren Arbeiten. Die k -NN-Regression zeichnet sich aber besonders durch eine gute Klassifikation der langsamen Musikstücke aus. Die

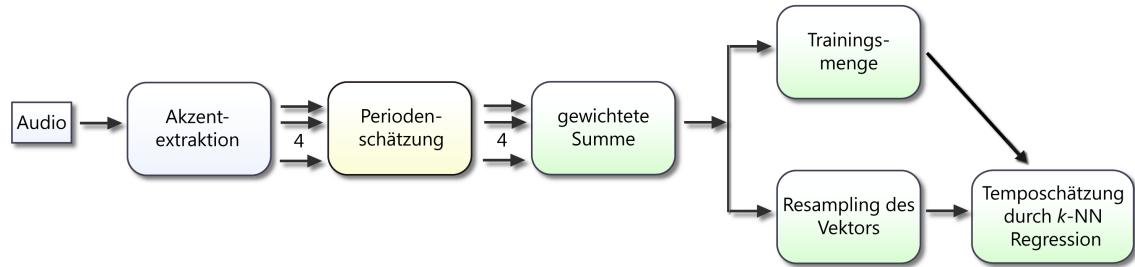


Abbildung 5.4: Übersicht des verallgemeinerten Verfahrens. Mehrere Akzente werden verarbeitet und vor der Klassifikation zusammengeführt.

Tabellen 5.1 und 5.2 stammen aus [17] und zeigen in der Konfusionsmatrix, welche Fehler die Verfahren bei der Schätzung von Tempokategorien machen. Vergleicht man die Daten miteinander, stellt man fest, dass durch die Regression langsame Musikstücke häufiger korrekt klassifiziert werden als bei [32], dass bevorzugt ein mittleres Tempo schätzt.

Die Tabelle zeigt jedoch relativ schlechte Klassifikationsraten bei schnellen Musikstücken: Etwa ein Drittel der schnellen Musikstücke werden langsam geschätzt. Das ist umso erstaunlicher, als viele kommerzielle Programme zur Tempoerkennung auf schnelle Musik oft besser erkennen als langsame. Deshalb wird das Verfahren aus [17] nun verallgemeinert.

5.2 Verallgemeinerung des Verfahrens

Bei langsamer Musik ist es erfolgversprechend, die Harmoniewechsel zu untersuchen. Schnelle Musik hingegen hat – insbesondere wenn es sich um Tanz- oder Clubmusik handelt – häufig ausgeprägte Muster im Bass- oder Perkussionsbereich. Daher soll das Verfahren von Eronen und Klapuri hier etwas verallgemeinert werden, indem nicht lediglich *ein* musikalischer Akzent berechnet wird, sondern gleich *mehrere*. Inspiriert ist diese Idee durch die Arbeiten von Scheirer [45] und Klapuri *et al.* [32], wo die Periodenschätzung auf mehreren Bändern separat durchgeführt und erst anschließend das Tempo geschätzt wird. Eronen und Klapuri überlegen am Ende ihres Artikels schon, ob man die Verfahren nicht kombinieren kann [17].

5.2.1 Akzentanalyse

Im Prinzip ermöglicht die modulare Gestaltung des Verfahrens, aus den Audiodaten beliebige musikalische Akzente zu berechnen. Wichtig ist nur, dass ein kontinuierliches Akzentsignal erzeugt wird, da die generalisierte Autokorrelationsfunktion auf kontinuierlichen Daten arbeitet. Andernfalls müsste die Periodenschätzung entsprechend angepasst werden. In dieser Arbeit werden vier Akzentsignale berechnet und zur Klassifikation eingesetzt.

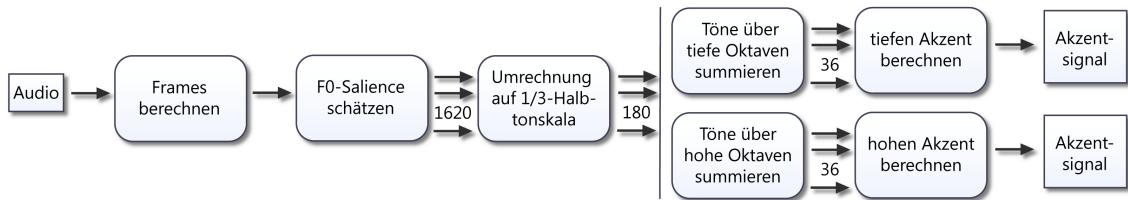


Abbildung 5.5: Aufbau der Chromaextraktion aus den Audiodaten

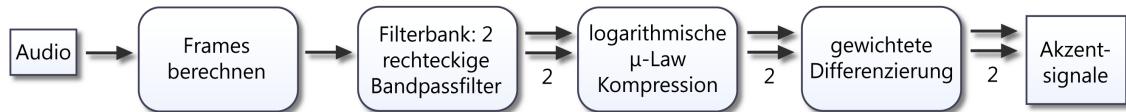


Abbildung 5.6: Aufbau der Energieakzentextraktion aus den Audiodaten

Chromabasierte Akzentsignale

Die Chromaanalyse berechnet wie in [17] Grundfrequenzkandidaten, aber die Frequenzen liegen hier logarithmisch verteilt zwischen 80 Hz und 2560 Hz. Über die fünf Oktaven liegen dann pro Halbton drei Werte vor (also 36 Tonschritte pro Oktave), die verwendet werden, um zwei unterschiedliche chromabasierte Akzente zu berechnen:

Tiefer Chromaakzent Für den tiefen Akzent werden die jeweils 36 Tonschritte der ersten drei Oktaven – von 80 Hz bis 640 Hz – zu einem Chromavektor zusammengefasst. Der tiefe Akzent soll die Harmoniewechsel der Begleitung beschreiben.

Hoher Chromaakzent Der hohe Akzent kombiniert die oberen drei Oktaven und berücksichtigt damit Frequenzen von 320 Hz bis 2560 Hz. Die untere Oktave (320 Hz bis 640 Hz) geht nur zu einem geringeren Anteil in den Akzent ein. Damit soll die Charakteristik der Melodie beschrieben werden.

Die Berechnung der beiden Akzente unterscheidet sich bei genauer Betrachtung nur im Aufsummieren über die Oktaven, wodurch zwei unterschiedliche Chromamatrizen entstehen. Aus diesen wird dann das jeweilige Akzentsignal berechnet, wie in Abschnitt 3.2.4 beschrieben.

Energieakzente auf Frequenzbereichen

Um genauere Informationen über Wiederholungen bei Bass und Perkussion zu erhalten, werden Energieakzente als zusätzliche Akzentkomponenten aus dem Audiosignal berechnet. Die Berechnung auf den Frequenzbändern orientiert sich an der bandweisen Akzentberechnung aus Abschnitt 3.1. Die Energie auf dem Frequenzband wird über die Zeit normiert und gewichtet differenziert. Da Harmoniewechsel nicht eingefangen werden müssen, wird das Audiosignal nur in zwei grobe Frequenzbänder zerlegt, analysiert und zu folgenden Akzenten verarbeitet:

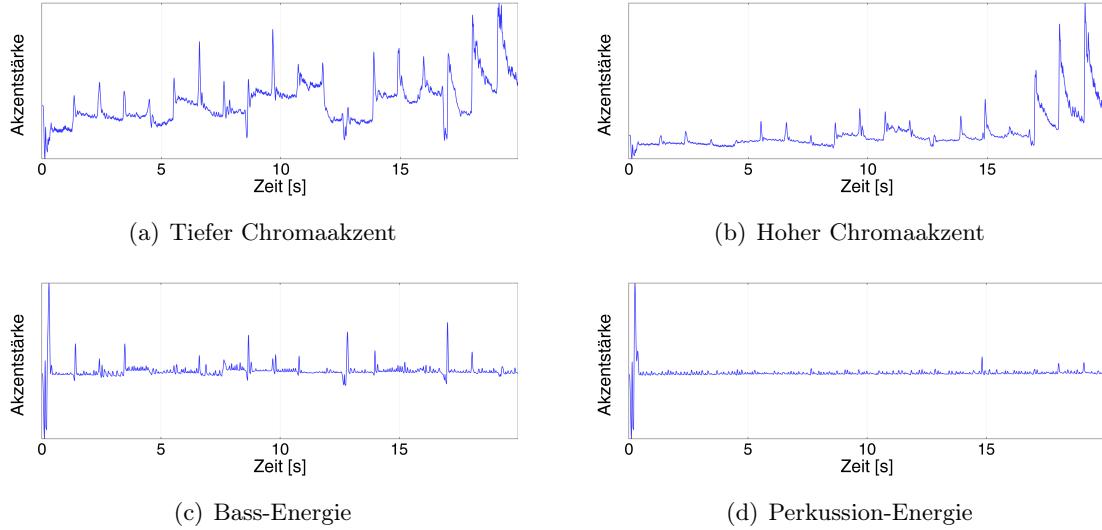


Abbildung 5.7: Die vier für Akzente, die für die Klassifikation berechnet werden

Bass-Energie Der Akzent berechnet sich aus der Energie auf den Frequenzen unterhalb von 110 Hz. Er beschreibt die Energie im Bassbereich und ist somit bei Stücken mit intensivem Schlagzeug ein guter Indikator.

Perkussion-Energie Der Akzent versucht das Einsetzen von Perkussionsinstrumenten zu beschreiben und betrachtet Frequenzen oberhalb der Melodielinie zwischen 3,2 kHz und 18 kHz.

Nachdem das Spektrum auf die entsprechenden Frequenzen eingeschränkt ist, werden die Energieakzente parallel weiterverarbeitet.

5.2.2 Periodenschätzung

Die Periodenschätzung erfolgt, genau wie in Abschnitt 4.2.1 beschrieben, mittels generalisierter Autokorrelationsfunktion und wird daher hier nicht nochmal im Detail erklärt. Die mittleren Perioden werden für jeden Akzent separat mit unterschiedlichen Werten p_c und p_e als Parameter berechnet. Hierbei beschreibt p_c die Kompression auf den Chromaakzenten und p_e die Kompression auf den Energieakzenten. Das ermöglicht die separate Optimierung der Periodenschätzung für die unterschiedlichen Akzentkategorien. In den Berechnungen bei dieser Arbeit hat sich, wie schon in [17], $p_c = 0,65$ für die Chromaakzente bewährt, und mit $p_e = 1,4$ für die Energieakzente wurden gute Ergebnisse erreicht.

5.2.3 k -NN-Regression

Die Regression wird grundsätzlich so durchgeführt wie in Abschnitt 4.7 vorgestellt. Da jedoch nun vier Periodenvektoren vorliegen, ergeben sich zwei grundsätzlich verschiedene Modelle der Klassifikation:

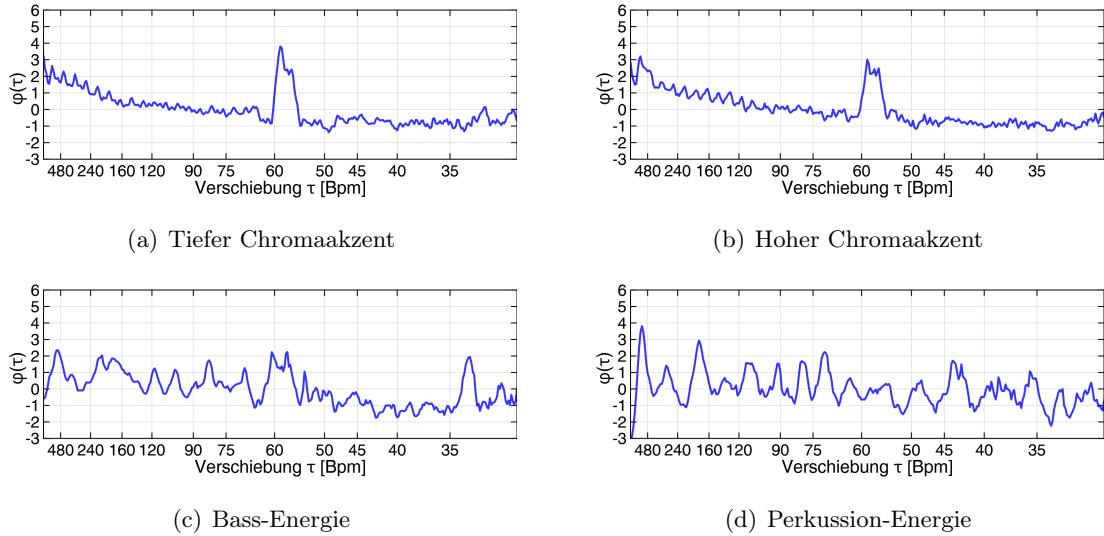


Abbildung 5.8: Die Periodenvektoren für die vier für Akzente

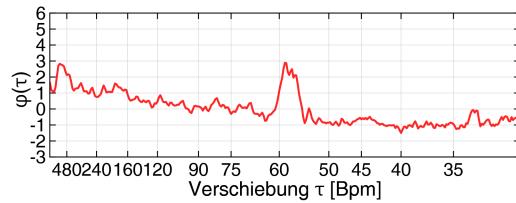


Abbildung 5.9: Die gewichtete Summe der vier Periodenvektoren der Akzente. Die Gewichtung beträgen [0,35 0,20 0,30 0,15].

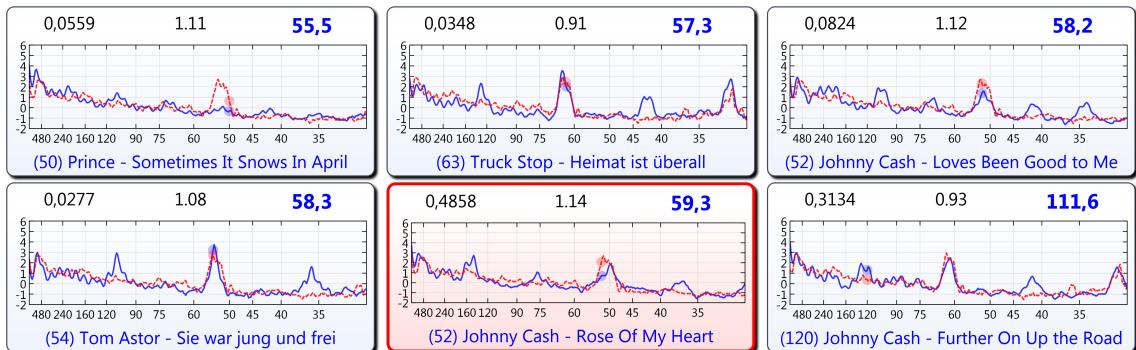


Abbildung 5.10: Klassifikation der gewichteten Periodenvektoren: Die $k = 6$ nächsten Nachbarn. Das Gewicht eines Nachbarn ist oben links angegeben. In der Mitte ist das Verhältnis des Resamplings, und rechts das resultierende Tempo.

Kombination der Periodenvektoren Eine Möglichkeit ist, die Periodenvektoren *vor* der Klassifikation geschickt zusammenzufassen. Die Vektoren beschreiben die Periodenschätzungen auf den unterschiedlichen Akzenten. In dieser Arbeit werden die Vektoren gewichtet gemittelt, um einem einzigen Repräsentanten für die Klassifikation zu erhalten.

Kombination der Klassifikationsergebnisse Die alternative Möglichkeit klassifiziert zuerst die vier Periodenvektoren einzeln. Erst *danach* wird das Tempo durch die Kombination der Klassifikationsergebnisse bestimmt. Da hierbei die Regression mehrfach ausgeführt werden muss, ist der Rechenaufwand höher. Interessant ist die Frage, wie die Ergebnisse geschickt zu dem geschätzten Tempo kombiniert werden können.

Diese Arbeit beschränkt sich auf das erste Modelle. Es soll auf einer Trainingsdatenbank getestet und mit dem Verfahren von Eronen und Klapuri aus [17] verglichen werden.

Kapitel 6

Umsetzung des Verfahrens

Bevor man die im vorigen Kapitel vorgestellten Ideen darstellen und vergleichen kann, müssen die Ergebnisse berechnet werden können. Daher werden hier einige Aspekte der Umsetzung dargestellt.

6.1 Anforderungen

Vor Beginn der Umsetzung müssen die Anforderungen an das resultierende Programm genauer untersucht werden. Um eine vernünftige Analyse des Verfahrens und der Ergebnisse zu ermöglichen, muss das Programm:

Musikstücke der Trainingsdatenbank einlesen Die Trainingsdatenbank muss leicht um neue Beispiele erweitert werden können. Für die Analyse wäre es interessant, die Stücke kommentieren zu können, um besonders schlecht oder gut klassifizierte Stücke auf spezielle Merkmale zu untersuchen.

Akzentsignale und Periodenvektoren aus Musikstücken extrahieren Die beiden ersten Schritte des Verfahrens können zusammengefasst werden, da für die Klassifikation die Akzentsignale nicht mehr benötigt werden.

Die Klassifikation benötigt aber die Periodenvektoren aller Stücke der Trainingsdatenbank, daher sollten diese vorab berechnet und gespeichert werden können. Außerdem müssen bei der Klassifikation eines unbekannten Stücks die Periodenvektoren mit den gleichen Einstellungen berechnet werden wie die Daten der Trainingsdatenbank, um vergleichbare Ergebnisse zu erhalten. Das Programm sollte die Parameter speichern, mit denen die Trainingsdaten berechnet wurden.

Die Berechnung der Akzentsignale und der Periodenvektoren stellt einige Anforderungen an die Möglichkeiten zur Signalverarbeitung. So müssen

- Audiosignale eingelesen werden

- Signale in Frames eingeteilt werden
- Spektrogramme berechnet werden - hierfür wird die Fouriertransformation benötigt
- Butterworth-Filter sechster Ordnung berechnet werden
- Die generalisierte Autokorrelationsfunktion implementiert werden - dies benötigt zusätzlich die inverse Fouriertransformation.

Bei der Wahl einer Programmierumgebung zur Umsetzung ist zu berücksichtigen, mit welchem Aufwand die Signalberechnungen zu realisieren sind.

Klassifikation von Musikstücken durchführen Die Temposchätzung muss auf Grundlage der Trainingsbeispiele eine Klassifikation durchzuführen. Zum Vergleich mit [17] soll ein Leave-One-Out auf der Trainingsdatenbank durchgeführt werden. Für unbekannte Musikstücke werden zuerst die Periodenvektoren aus dem Audiosignal erzeugt, die dann mit den Trainingsdaten klassifiziert werden.

Eine Visualisierung der Klassifikation soll das Verständnis der Klassifikation vertiefen und Ideen zur weiteren Verbesserung liefern. Dargestellt werden sollen die k nächsten Nachbarn und die Ähnlichkeiten ihres Periodenvektors zum klassifizierten Stück.

6.2 Referenzimplementierung mit Matlab

Matlab stellt von Haus aus eine ganze Reihe von Funktionen zur Verfügung, die die Signalverarbeitung unterstützen. So kann Matlab Audiodateien lesen und abspielen, und stellt unter anderem die FFT („*fast fourier transformation*“) und IFFT („*inverse fast fourier transformation*“) zur Verfügung. Viele Operationen können vektorbasiert durchgeführt werden, was die Implementierung der Verfahren erleichtert.

Ziel war es hierbei nicht, eine performante Anwendung zu erstellen, die alle Anforderungen des letzten Kapitels erfüllt. Vielmehr soll das Verfahren überhaupt so weit umgesetzt werden, dass eine Audiodatei geladen wird und Akzentsignale, Periodenvektoren und das geschätzte Tempo berechnet werden. Matlab bietet darüber hinaus vielfältige Möglichkeiten zur grafischen Darstellung der Daten. So lassen sich Zwischenergebnisse betrachten und verifizieren.

6.2.1 MIRtoolbox

Eine bedeutende Hilfe beim Einstieg in die Signalverarbeitung stellt MIRtoolbox¹ dar. MIRtoolbox enthält viele in Matlab geschriebene Funktionen zur Extraktion musikalischer

¹Die MIRtoolbox ist aktuell in der Version 1.3 unter <https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox> erhältlich. Dort findet man auch die Dokumentation.

Objekt	Speicherbedarf
Audio	189,34 MB
Frames	387,66 MB
Spektrum	284,02 MB
Audio (Matlab Array)	94,67 MB

Tabelle 6.1: Speicherbedarf von MIRtoolbox. Die geladene Audiodatei hat auf der Festplatte eine Größe von etwa 23,67 MB und ist 4:41 Minuten lang, also eine typische Audiodatei der Trainingsmenge.

Merkmale aus Audiodaten und ist als ein modulares Framework realisiert, dass der Benutzer relativ leicht benutzen und parametrisieren kann [36].

Das Audiosignal lässt sich durch Filterbänke in verschiedene Kanäle aufteilen oder in Frames zerlegen, und die meisten Merkmale lassen sich sowohl für Multi-Channel-Audiosignale als auch für Frames auswerten. Ein weiterer Vorzug ist die größere Vielfalt an unterstützten Audioformaten. MIRtoolbox nutzt die grafischen Möglichkeiten von Matlab für eine gute Visualisierung der verschiedenen Daten. Die Abbildungen im ersten Teil dieser Arbeit stammen daher größtenteils aus MIRtoolbox.

Listing 6.1: Frame und Spektrogramm mit der MIRtoolbox

```
audio = miraudio('..\sylver_30s.wav');
frame = mirframe(audio, 'Length', 4096, 'sp', 'Hop', 2048, 'sp');
spectrogram = mirspectrum(frame);
```

Die Frames und das Spektrogramm eines Musikstücks lassen sich mit MIRtoolbox elegant berechnen. Zur Verarbeitung des Spektrogramms zum Chromaakzent muss auf die Rohdaten des Spektrogramms zurückgegriffen werden. Dafür werden die Daten des Spektrogramms als Matrix herausgezogen, die dann mit allen von Matlab bereitgestellten Funktionen manipuliert werden kann:

```
tmp = get(spectrogram, 'Data');
SP = tmp{1,1}{1,1};
```

Die Variable `SP` ist vom Typ `<2048x652 double>`. Die Spalten beschreiben die einzelnen Frames, die Zeilen stehen für die Frequenzen des Spektrums.

6.2.2 Speicherbedarf

Auch wenn es bequem ist, das Spektrogramm mit der MIRtoolbox zu berechnen, spricht die Praxis gegen ihre Verwendung für die Berechnung der Akzente. Die Toolbox ist speicherhungrig, wie die Tabelle 6.1 belegt. Für die drei Objekte werden über 860 MB Arbeitsspeicher benötigt, das für die praktische Arbeit mit einer Trainingsmenge von etwa 100 Stücken diskutabel.

Allerdings ist schon die Rohdatenmenge bei Audiosignalen hoch. Die reinen Daten des Audiosignals belegen bereits knapp 95 MB als Array. Der Unterschied zu den gespeicherten 23,67 MB ergibt sich durch die benötigten Bytes pro Sample: In der Datei sind 16 Bit pro Sample gespeichert. Die Samples werden jedoch als Gleitkommawerte doppelter Genauigkeit in Matlab interpretiert und belegen jeweils volle 8 Byte.

Den Speicherbedarf für die Daten muss man also hinnehmen. MIRtoolbox berechnet die Frames aber schon beim Erstellen *vollständig* für das ganze Musikstück - und genauso das komplette Spektrogramm. Es genügt jedoch, die Chromavektoren (und genauso die Energie auf den Frequenzbändern) *sequenziell* auf dem Spektrum eines einzelnen Frames zu berechnen.

Listing 6.2: Manuelles Erzeugen von Frames in Matlab

```
% Anzahl der Frames
n = ceil((length(audio)-frame_size)/hop);
for (t=1:n)
    % Framedaten berechnen. ACHTUNG: Matlab indiziert Arrays ab 1!
    frame = audio( (t-1)*hop+1 : (t-1)*hop+frame_size ) .* window;

    % Das Spektrum dieses Frames berechnen.
    spectrum = fft(frame);
    % Chromavektoren berechnen...
end
```

Die sequenzielle Berechnung der Frames ohne die Toolbox gestaltet sich in Matlab nicht schwierig. Damit ist die Implementierung der Frames dann aber ein Teil des Verfahrens, ganz im Gegensatz zur vorbildlichen Eleganz und Wiederverwertbarkeit mit MIRtoolbox.

6.2.3 Laufzeit

Ein weiteres Problem ist die Laufzeit des Verfahrens mit Matlab. Die Extraktion des Chromaakzents ist rechenintensiv. Die Energieakzente werden dabei als Nebenprodukt aus dem gleichen Spektrum erzeugt, sodass das Verfahren zwar Zeit spart, aber Flexibilität einbüßt. Die Parameter sind damit für beide Akzentarten die gleichen, und auch die Autokorrelationsfunktionen werden für alle Akzente mit denselben Parametern ausgerechnet. Dennoch ist die Berechnung der Akzente und Periodenvektoren für ein Musikstück weit davon entfernt, echtzeitfähig zu sein. Beispielsweise werden die Akzente und Perioden für ein Stück mit 4:41 Minuten Länge in 08:58 Minuten berechnet². Auf die Berechnung der ganzen Testdatenbank (knapp 100 Stücke) wurde daher aus Zeitgründen verzichtet. Statt dessen wurden Ausschnitte der Stücke analysiert.

²Sämtliche Angaben zur Laufzeit wurden auf dem Computer des Autors mit Intel Core i7-920-Prozessor (Quadcore), 6 GB Arbeitsspeicher und Windows 7 Professional 64 Bit als Betriebssystem gemessen.

6.3 Testframework „TempEx“

Da die Berechnungen mit Matlab so lange dauern und die Arbeit mit den Trainingsdaten unhandlich ist, wurde für die Arbeit das Testframework „TempEx“ (für „tempo explorer“) implementiert. Das Framework beruht auf einer Datenbank, die folgende Bestandteile beinhaltet:

- Trainingsbeispiele
- Merkmale, Merkmalswertdefinitionen und die Zuordnung zu den Beispielen
- Testläufe mit bestimmten Parametern
- Ergebnisse der Testläufe auf den Beispielen

Die Programmierung des Frameworks und des Verfahrens zur Tempoerkennung erfolgt in F# und ist in eine eigene Klassenbibliothek gekapselt. F# ist eine funktionalen Programmiersprache für Microsofts .NET-Framework, die zusätzlich zum funktionalen Paradigma auch die anderen großen Programmierparadigmen – iterative und objektorientierte Programmierung – unterstützt³. Die grafische Benutzeroberfläche basiert auf der WPF-Bibliothek⁴ und ist in C# und XAML umgesetzt, weil die Unterstützung in Visual Studio für C# ausgereifter ist. Außerdem zeigt sich hier die relativ nahtlose Integration unterschiedlicher Programmiersprachen innerhalb des .NET-Frameworks.

Das Testframework basiert auf folgenden Technologien, die allesamt frei zugänglich sind:

.NET-Framework Microsofts .NET-Framework⁵ bildet die Grundlage der Entwicklung des Testframeworks. Es stellt, ähnlich wie Java, eine verwaltete Laufzeitumgebung („virtual machine“) sowie eine umfangreiche Klassenbibliothek zur Verfügung. Auch wenn .NET konzeptionell auf verschiedene Plattformen portiert werden kann, findet es in der Praxis hauptsächlich auf Windows-Systemen Verwendung. Es steht aber über das Mono-Projekt⁶ auch unter Linux zur Verfügung. Dafür bietet .NET die Integration verschiedener Programmiersprachen in einer Umgebung, was hier mit C# und F# demonstriert werden soll. Die Implementierung basiert auf dem vollständigen .NET Framework 4.

³Eine kleine Einführung in die funktionale Programmierung mit F# findet sich online im MSDN-Magazin unter [39] oder in der „F# Survival Guide“ [42]. Bücher zur Programmiersprache stehen vorzugsweise in englischer Sprache zur Verfügung, wie „Programming F#“ [52], „Expert F# 2.0“ [53] oder „F# for Scientists“ [27].

⁴Nähere Informationen zur Windows Presentation Foundation (WPF) liefert das „.NET Framework Developer Center“ unter <http://msdn.microsoft.com/de-de/netframework/aa663326.aspx>.

⁵Es existiert eine Fülle von Literatur über das .NET-Framework. Einen ersten Überblick bietet das „.NET Framework Developer Center“ in Microsofts MSDN unter <http://msdn.microsoft.com/de-de/netframework/default.aspx>.

⁶Siehe <http://www.mono-project.com>

Microsoft Sql Server Compact Die Datenbank ist mit Microsoft Sql Server Compact⁷ als eingebettete Datenbank umgesetzt, sodass kein Datenbankserver zur Verfügung stehen muss. Die Datenbankdatei kann einfach eingeladen werden. Alternativ kann die Datenbank auch eingebettet auf SQLite oder einen vollwertigen Datenbankserver wie MySql, Oracle oder Microsofts SQL-Server betrieben werden.

NAudio NAudio⁸ ist eine Open-Source Audio- und MIDI-Bibliothek. Sie wird in der Version 1.3 verwendet, um Audiodateien einzulesen.

Math.NET Numerics Math.NET⁹ ist eine mathematische Open-Source-Bibliothek für die .NET-Plattform. Math.NET Numerics stellt Methoden und Algorithmen zur Unterstützung numerischer Berechnungen zur Verfügung. Sie wird hier verwendet, um Fouriertransformationen zu berechnen.

F# Klassenbibliothek Leider beinhaltet das .NET-Framework nicht die von F# benötigte Klassenbibliothek `FSharp.Core.dll` in der Version 2.0.0.0 und 4.0.0.0. Daher muss F# auf dem System installiert sein¹⁰.

F# Power Pack Das F# Power Pack¹¹ ist eine Sammlung von Bibliotheken und Tools für F#. Sie enthält unter anderem Unterstützung für komplexe Zahlen, Vektoren und Matrizen und wird in der Version 2.0.0.0 verwendet.

6.3.1 Verwendung des Programms

Die Abbildungen auf diesen Seiten zeigen exemplarisch die Verwendung des Programms. Abbildung 6.1 zeigt TempEx mit geladener Trainingsdatenbank. Hier können die Trainingsbeispiele beschrieben und mit Merkmalen versehen werden. Die Datenbank kann um einzelne Audiodateien oder ganze Verzeichnisse erweitert werden. Hier kann auch der Beat auf der Tastatur mitgeklopft werden, um das Tempo des Beispiels manuell zu schätzen.

Ist eine Trainingsdatenbank geladen, können Testläufe erstellt und berechnet werden. Ein Beispiel hierfür ist in Abbildung 6.2 dargestellt, die TempEx bei der rechenintensiven Berechnung der Periodenvektoren für die gesamte Trainingsmenge zeigt.

⁷Nähere Informationen zum Sql Server Compact finden sich unter <http://www.microsoft.com/germany/sql/2008/compact.mspx>

⁸Das Projekt findet sich unter <http://naudio.codeplex.com>

⁹Math.NET stellt unter <http://www.mathdotnet.com/About.aspx> unter anderem das Numerics-Projekt bereit.

¹⁰Die Klassenbibliotheken für F# gibt es noch nicht als eigenständiges Redistributable. Daher muss derzeit das komplette *F# August 2010 Community Technology Preview* mit dem Compiler und Tools installiert sein. Microsoft stellt das CTP unter der Adresse <http://www.microsoft.com/downloads/en/details.aspx?FamilyID=f8c623ae-aef6-4a06-a185-05f59be47d67> zur Verfügung. Die F#-Version entspricht genau derjenigen, die ab Visual Studio 2010 Professional mitgeliefert wird.

¹¹bereitgestellt unter <http://fsharppowerpack.codeplex.com>

6.3. TESTFRAMEWORK „TEMPEX“

67

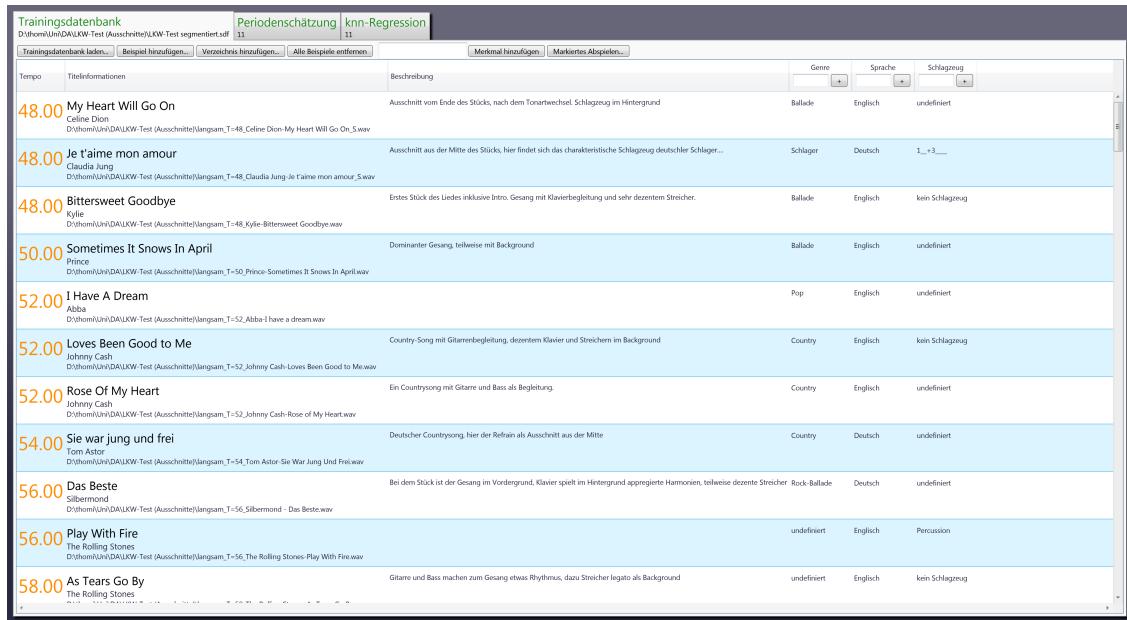


Abbildung 6.1: TempEx mit geladener Trainingsdatenbank

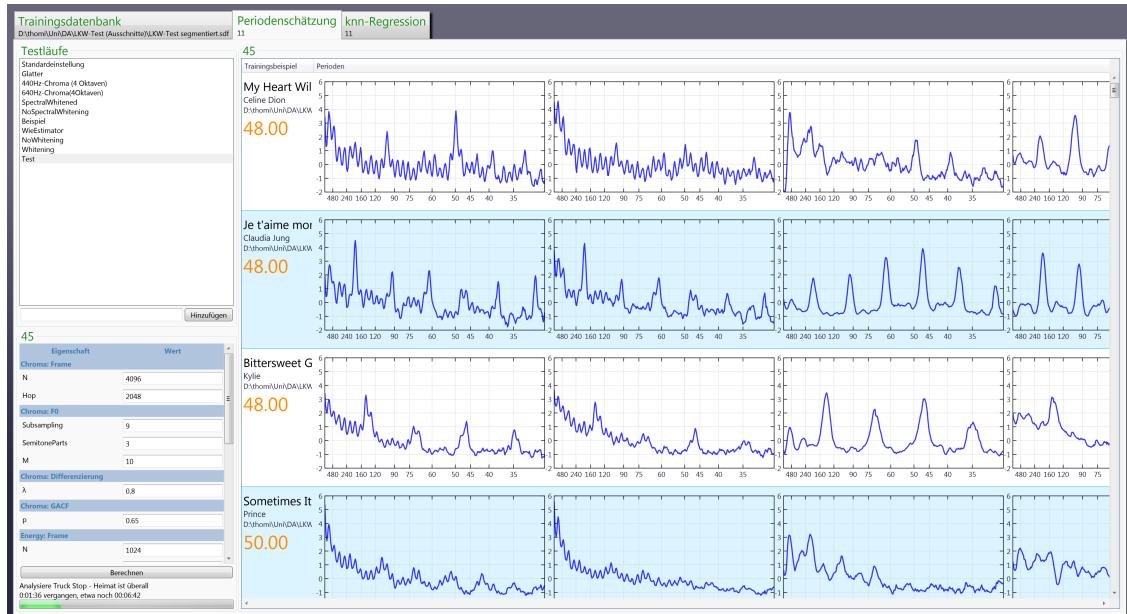


Abbildung 6.2: TempEx berechnet die Periodenvektoren der einzelnen Akzente auf der Trainingsdatenbank. Die Einstellungen für die Berechnungen finden sich im linken unteren Bereich, abhängig vom darüber gewählten Testlauf.

Trainingsdatenbank	Periodenschätzung	knn-Regression
D:\thoma\Uni\DA\UKW-Test (Ausschnitte)\UKW-Test segmentiert.pdf	11	11
90	# Trainingsbeispiel	
120	216 Claudia Jung - Amore amore	
Segmentiert	267 Clueso - Chicago	63 126,0 100,0% 1,000 Langsam Schnell
für Whittening optimieren	291 Europe - The Final Countdown	84 163,4 97,0% 0,978 Langsam Schnell
NoWhitening	316 Cornelian Harmonists - Veronika der Lenz ist da	116 58,0 50,0% -1,000 Mittel Langsam
k-NN Regressionsinstellungen	317 Chuck Berry - Roll Over Beethoven	141 70,6 49,9% -0,998 Schnell Langsam
k	294 Robbie Williams - Old Before I Die	176 88,8 49,5% -0,987 Schnell Langsam
y	308 Culture Candelabro - E'y DJ	116 58,7 49,4% -0,982 Mittel Langsam
Median Performance: 59,5%	309 Katy Perry - I Kissed A Girl	130 66,0 49,2% -0,978 Schnell Langsam
Median Mittelwert: 7,6%	296 Johnny Cash - Further On Up The Road	126 64,1 49,1% -0,975 Schnell Langsam
Median Std: 18,43%	219 Claudia Jung - Je T'aime mon amour	120 61,7 48,6% -0,959 Schnell Langsam
Zeige Klassifikation:	219 Claudia Jung - Je T'aime mon amour	48 66,0 37,5% 0,459 Langsam Langsam
Unbekanntes Stück klassifizieren	268 Johnny Cash - A Legend In My Time	84 59,3 29,4% -0,503 Langsam Langsam
Regression exportieren...	237 Johnny Cash - Help Me	63 57,0 9,5% -0,244 Langsam Langsam
Tabelle exportieren...	288 Johnny Cash - Like The 309	112 104,4 6,6% -0,101 Mittel Mittel
Bilder exportieren...	311 Daughtry - Feels Like Tonight	132 124,7 5,5% -0,082 Schnell Schnell
Gewichtete Optimierung...	221 Prince - Sometimes It Snows In April	50 52,5 5,0% 0,071 Langsam Langsam
langsam	295 Die Ärzte - Nichts in der Welt	120 126,0 5,0% 0,070 Schnell Schnell
mittel	298 Udo Jürgens - Mit 65 Jahren	120 126,0 5,0% 0,070 Schnell Schnell
schnell	301 Claudia Jung - Stromme Signale	126 120,0 4,8% -0,070 Schnell Schnell
	229 Timbaland feat. One Republic - Apologize	58 60,7 4,7% 0,066 Langsam Langsam
	259 Gunter Gabriel - Wie einmal tief im Keller saß	80 83,7 4,7% 0,066 Langsam Langsam
	218 Colbie Dion - My Heart Will Go On	48 49,9 4,0% 0,057 Langsam Langsam
	270 Madonna - The Power of Goodbye	84 87,4 4,0% 0,057 Langsam Langsam
	309 Fettes Brot - Endebben	130 124,8 4,0% -0,059 Schnell Schnell
	286 Spice Girls - Wannabe	108 112,0 3,7% 0,052 Mittel Mittel
	261 Patrick Lindner - Jedes Herz braucht eine Heimat	80 77,0 3,7% -0,054 Langsam Langsam
	242 Maria Mena All This Time Pick-Me - Up Song	66 68,4 3,6% 0,052 Langsam Langsam
	277 Claudia & alex - Ein par Takte Sandra	90 93,2 3,6% 0,051 Mittel Mittel

Abbildung 6.3: Die Klassifikationsergebnisse für einen gewählten Testlauf. Die Einstellungen für k und γ werden nach der Klassifikation auf die optimalen Werte voreingestellt.

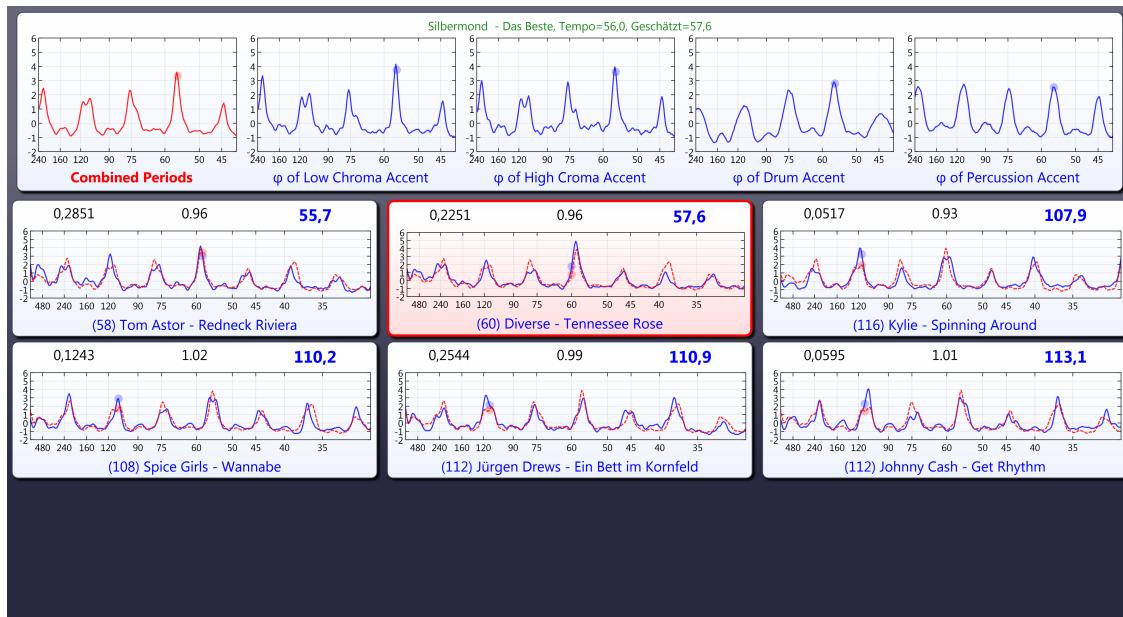


Abbildung 6.4: Die Informationsansicht zu einem Stück in TempEx. Die berechneten Periodenvektoren und gewichtete Summe werden dargestellt. Darunter sieht man die Nachbarn der Klassifikation mit einem Vergleich der Trainingsbeispiele (blau) mit dem Periodenvektor des zu schätzenden Stücks (rot gestrichelt) nach dem Resampling. Der Median bestimmt schließlich das Tempo und wird daher gekennzeichnet.

Schritt	Rechenzeit	Anteil an der Gesamtzeit
Energieakzente	0:44.9	36%
Chromaakzente	1:07.6	54%
Berechnung der Perioden	0:12.2	10%
Gesamtlaufzeit	2:04.7	100%

Tabelle 6.2: Rechenzeiten der Verfahrensschritte auf einer 30-minütigen Audiodatei

Sind die Periodenvektoren für die Trainingsdaten berechnet, kann eine Leave-One-Out-Klassifikation der Trainingsdatenbank durchgeführt werden. Die Genauigkeit, die Kategoriefehler und die Werte für die Stücke werden tabellarisch dargestellt (Abbildung 6.3). Für genauere Untersuchungen steht zu jedem Stück die in Abbildung 6.4 gezeigte Detailinformation zur Verfügung.

6.3.2 Laufzeit

Neben der Integration der oben beschriebenen Funktionen war die Reduzierung der Laufzeit ein Ziel der Umsetzung des Verfahrens in ein eigenständiges Programm. Die Berechnungen können sicherlich noch optimiert werden, sind aber erheblich schneller als in der Implementierung mit Matlab. Die Berechnung der Periodenvektoren für das gleiche, 4:41 Minuten lange Musikstück dauert 20,96 Sekunden, und die komplette Trainingsdatenbank lässt sich in 28:04 Minuten berechnen.

Anders als in der Matlab-Implementierung werden die Energieakzente eigenständig berechnet und erzielen durch kleinere Frames eine höhere zeitliche Auflösung. Da die Frames und Spektren für die Energieakzente erneut berechnet werden müssen, nehmen die Energieakzente nun einen erheblichen Teil der Rechenzeit in Anspruch, wie der Tabelle 6.2 zu entnehmen ist. Zukünftige Optimierungen sollten daher die Berechnung des Energieakzents einschließen.

Die verbleibenden Abschnitte zeigen Details der Implementierung. Hierbei soll die Wahl von F# als Programmiersprache zur Signalverarbeitung anhand von drei Beispielen begründet werden. Sie zeigen, dass sich die Signaldaten elegant funktional verarbeiten lassen und sich bei Bedarf iterativer Code einflechten lässt.

6.3.3 Modul zur Erzeugung der Frames

In Abschnitt 6.2.2 wurde gezeigt, dass die Arbeit auf Frames mit MIRtoolbox elegant ist, aber viel Speicher belegt. Da die Werte nur sequenziell benötigt werden, wurden die Frames statt dessen in Matlab erzeugt und nacheinander durchlaufen. Gleichzeitig geht die Eleganz und Wiederverwendbarkeit einer gekapselten Frameberechnung verloren.

In F# können **Sequenzen**¹² definiert werden, bei denen die einzelnen Elemente bei Bedarf berechnet werden. So kann man beispielsweise die theoretisch unendliche Sequenz `let N = {0.0 .. Double.PositiveInfinity}` definieren, ohne sie zu berechnen. Die Elemente der Sequenz werden erst berechnet, wenn der sequenzielle Zugriff erfolgt. F# bietet im Modul `seq` einige Funktionen an, mit denen Sequenzen bequem manipuliert und erzeugt werden können. Die wichtigsten zum Verständnis des folgenden Abschnitts sind:

- `Seq.map f s` wendet eine Funktion `f: (T->U)` auf jedes Element einer Sequenz `s` des Typs `T` an und erzeugt damit eine neue Sequenz vom Typ `U`.
- `Seq.init n f` erzeugt eine Sequenz mit `n` Elementen des Typs `T`, die mit der Funktion `f:(int->T)` für den übergebenen Index erzeugt werden.

Für die Arbeit mit Arrays und Listen stehen in den Modulen `Array` und `List` ähnliche Funktionen zur Verfügung.

Mit Sequenzen lassen sich die Frames eines Audiosignals elegant definieren. Hierzu wurde eine Klasse `Frame` entwickelt, die neben Informationen über das zugrundeliegende Signal eine Sequenz von Frames erstellt. Die einzelnen Frames der Sequenz sind als Signale implementiert, und werden aus einem Signal erzeugt.

Listing 6.3: Erzeugen einer Sequenz von Frames in F#

```
let fromSignal length hop window (s:Signal) =
    let n = s.Length - length - 1
    let w = window length
    let buffer = Array.create length 0.0
    let frames = seq { for offset in 0 .. hop .. n ->
        Signal(sprintf "Frame %d/%d" (offset/hop + 1) (n/hop+1),
            buffer |> Array.InPlace.init length
            (fun i -> s.[offset+i] * w i),
            s.SamplingRate, offset)}
    Frame(s, frames, length, hop)
```

Hier wird für jede Verschiebung des Frames zwischen 0 und `n` im Abstand `hop` ein Element der Sequenz generiert. Jedes Element ist ein Signal, das einen Namen erhält (z.B. „Frame 1/43“), die Abtastrate des Audiosignals übernimmt und sich seine Verschiebung merkt. `Array.create` erzeugt ein Array der Länge des Frames als Puffer, damit nicht für die Daten jedes Frames neuer Speicher auf dem Heap angefordert und nachher durch die Garbage-Collection freigegeben werden muss. Der Puffer wird durch die Funktion `Array.InPlace.init` mit den Werten des Audiosignals an der entsprechenden Position gefüllt, die dabei mit der Fensterfunktion multipliziert werden¹³. Das verwendete Modul `Array.InPlace` ist eine selbstdefinierte Erweiterung zur direkten Manipulation der Werte eines

¹²Als Referenz für die Programmiersprache F# sei auf Microsofts MSDN verwiesen. Sequenzen werden dort unter <http://msdn.microsoft.com/de-de/library/dd233209.aspx> behandelt.

¹³

Arrays in der gewohnten Syntax¹⁴. Das Objekt der Klasse `Frame`, das am Ende erstellt wird, definiert also eine Sequenz der Frames des Signals als einzelne, kleine Signale. Berechnet werden die einzelnen Frames dann sequenziell bei Bedarf.

Listing 6.4: Verwendung des Frame-Moduls

```
let audio = Audio.FromFile("../sylver_30s.wav")
let rms =
    audio
    |> Frame.fromSignal 4096 2048 Signal.Windows.hann
    |> Seq.map Signal.rms

let result = rms |> Seq.toArray
```

Das Frame-Modul kapselt nun die Erzeugung von Frames aus dem Audiosignal, und lässt sich elegant *und* effizient anwenden. Die erste Zeile liest das Audiosignal aus einer Datei in den Bezeichner `audio` ein. Dann werden die Frames auf dem Audiosignal definiert, und anschließend wird auf jedem Element der Sequenz – sprich auf jedem Frame – der RMS-Wert des Framesignals berechnet. Genau genommen erfolgt die Berechnung aber erst in der letzten Zeile, wo aus der Sequenz – die nur noch aus einem RMS-Wert pro Frame besteht – ein Array erzeugt wird.

Erwähnenswert ist hierbei der Pipeline-Operator. Man würde die Summe einer Sequenz `s` normalerweise mit `Seq.sum s` berechnen. Der Pipeline-Operator ermöglicht es, `s |> Seq.sum` zu schreiben, also zuerst den Parameter anzugeben. Damit lassen sich Funktionsaufrufe aneinanderfügen und die Daten werden wie durch eine Pipeline verarbeitet. Gerade bei der Signalverarbeitung, die ja immer wieder Funktionen auf Signale und Frames anwendet,

Der Parameter `window` ist eine Fensterfunktion vom Typ `int -> int -> float`. Einfach ausgedrückt, erhält die Funktion die Länge des Frames und den Index auf dem Frame als ganze Zahlen und berechnet daraus den Wert der Fensterfunktion als reelle Zahl. Da alle Frames die gleiche Länge haben, wird `let w = window length` eine neue Funktion von `int -> float` erstellt. Die Länge ist damit fixiert, und `w` erwartet nur noch den Index auf dem Frame als Argument. Diese Technik ist als Currying bekannt und typisch für die funktionale Programmierung.

¹⁴Einer der Grundpfeiler der (reinen) funktionalen Programmierung ist die Anforderung, dass Berechnungen einen neuen Wert zurückliefern, anstatt den übergebenen Wert zu überschreiben. Werden neben der Berechnung des neuen Wertes weitere Änderungen am System durchgeführt (beispielsweise die Ausgabe auf der Konsole, die Veränderung eines Wertes, oder das Schreiben in eine Datenbank), spricht man von Seiteneffekten der Funktion. Idealerweise haben Funktionen keine Seiteneffekte, und einige Programmiersprachen wie Haskell verbieten Seiteneffekte, wenn diese nicht explizit gekennzeichnet werden. Dadurch werden funktionale Programme robuster, klarer und auch einfacher parallelisierbar. F# gestattet Seiteneffekte in Funktionen, und die im `Array.InPlace`-Modul definierten Funktionen überschreiben als Seiteneffekt die Werte des übergebenen Arrays, das am Ende einfach zurückgegeben wird. Mit Bedacht eingesetzt – wie im Beispiel des ohnehin sequenziellen Framings – lässt sich damit der Arbeitsspeicher effizienter nutzen, und die Verwendung eines eigenen Moduls kennzeichnet explizit die Stellen, in denen Seiteneffekte durch direkte Veränderungen an Arrays auftreten.

führt diese Schreibweise zu lesbarem Code. Ohne Pipelining müsste die obige Deklaration von `rms` im Code wie folgt geschrieben werden:

```
let frame = Frame.fromSignal 4096 2048 Signal.Windows.hann audio
let rms = Seq.map Signal.rms frame
```

Das Frame-Modul wird für die Erzeugung sämtlicher Frames des Verfahrens benutzt: Für die Berechnung der Chroma- und Energieakzente ebenso wie für die generalisierten Autokorrelationsfunktion. Die Chromaberechnung wird genau wie im obigen Beispiel sequenziell für jeden Frame ausgeführt, und nur der berechnete Chromavektor wird in der Zeile einer Matrix festgehalten. Details zur Chromaanalyse zeigt der nächste Abschnitt, wo der Fokus auf die Analyse des Spektrums eines einzelnen Frames gerichtet wird.

6.3.4 Berechnung der Grundfrequenzstärken

Listing 6.5: Berechnung der Grundfrequenzstärken in F#

```
let analyzeChroma (Y:float[]) =
    let a = 52.0<Hz>
    let b = 320.0<Hz>
    let d = 0.5/2.0
    let K = float Y.Length
    let fs = frame.SamplingRate

    let F0salience f0 =
        let t = fs / f0
        let g m = (f0 + a) / (m*f0 + b)
        let kappa m =
            Y.[roundI (m*K / (t + d)) .. roundI(m*K / (t - d))]

        seq { for m in {1.0 .. M} ->
                g m * Array.max (kappa m) }
        |> Seq.sum

    let maxToneSalience f0 =
        Seq.init (int semitones)
            (fun k -> float k/(subsampling*semitones))
        |> Utility.createEqualTemp f0
        |> Seq.map F0salience
        |> Seq.max

    frequencies
    |> Array.Parallel.map maxToneSalience
```

Die hier dargestellte Funktion `analyzeChroma` bekommt das Spektrum `y` als Eingabe und berechnet darauf die Stärke aller Frequenzen `frequencies`. Dazu werden zunächst einige Werte deklariert. Augenfällig ist die Definition `let a=52.0<Hz>`. Hier wird ausgenutzt, dass

F# Werte mit einer Maßeinheit typisieren kann. Zur Laufzeit werden die Werte zwar nicht umgerechnet, aber schon bei der Kompilierung wird überprüft, ob eine Zuweisung oder Berechnung die richtige Maßeinheit hat. So endet die Addition $52.0\text{Hz} + 1.0$ wegen der unterschiedlichen Maßeinheiten in einem Kompilierfehler. Interessant ist, dass der Compiler die Typisierungen auch umrechnet:

```
let x = 1.0<s> * 52.0<Hz>
```

berechnet für x – wenig erstaunlich – den Wert 52. Der Typ von x ist jedoch wieder `float`, da Hz für 1/s steht und sich die Maßeinheiten somit aufheben. Ein praktisches Feature von F#, das gerade für technische Berechnungen helfen kann, Laufzeitfehler durch falsche Zuweisungen zu vermeiden, und deshalb hier erwähnt werden soll.

Die interne Funktion `F0salience` berechnet die Stärke eines F0-Kandidaten f_0 (natürlich vom Typ `float<Hz>`, da es sich um eine Frequenz handelt). Definiert werden die Periode t und die Funktionen g und κ_m , die das Gewicht des m -ten harmonischen Obertons von t berechnen, bzw. seine Frequenzbestandteile aus dem Spektrum ausschneiden. Danach erfolgt die Summierung der M harmonischen Bestandteile, nachdem für alle Werte der Sequenz $\{1.0 \dots M\}$ das Gewicht für den jeweiligen Bestandteil mit dem größten Wert im Frequenzbereich des Obertons multipliziert wurde. Die Summe ist dann die Stärke des F0-Kandidaten.

Die Funktion `maxToneSalience` ist eher technischer Natur. Sie erzeugt zu jeder Frequenz f_0 eine Anzahl von Zwischenfrequenzen, die sich logarithmisch bis zur nächsten Drittelf-Halbtonfrequenz verteilen. Für jede dieser Zwischenfrequenzen (die erste ist f_0 selbst) wird mit `F0salience` die Stärke ausgerechnet und nur der höchste Wert zurückgegeben.

Die Berechnung wird durch die letzten beiden Zeilen initiiert:

```
freqencies
|> Array.Parallel.map maxToneSalience
```

Hier wird für jede der Frequenzen der Drittelf-Halbtöne – ein vorher initialisiertes Array von `float<Hz>`-Werten – die Grundfrequenzstärke berechnet. Der einfache Aufruf von `Array.Parallel.map` statt dem schon bekannten `Array.map` führt die Berechnungen für die Elemente von `freqencies` parallel durch. Das beschleunigt gerade bei aktuellen Multicore-Prozessoren die Berechnung des Akzents. F# ermöglicht die Parallelisierung von Arrayberechnungen auf diese einfache Art, da `map` in funktionaler Manier ein neues Array erzeugt. Die oft mit der Parallelisierung von Algorithmen einhergehenden Synchronisierungsprobleme werden bei (rein) funktionaler Programmierung entschärft, da die einmal deklarierten Werte nicht mehr geändert werden können.

6.3.5 Butterworth-Tiefpassfilter

Im Gegensatz zur Matlab-Umgebung steht der Testumgebung kein Framework zur Berechnung von IIR-Filtern („Infinite-Impulse-Response“) zur Verfügung. Die Umsetzung eines Butterworth-Tiefpassfilters für beliebige Ordnungen und Trennfrequenzen ist komplex, aber für das Verfahren wird nur eine Konfiguration des Butterworth-Filters benötigt: Ein Tiefpassfilter sechster Ordnung mit einer Trennfrequenz $f_{LP} = 10$ Hz bei einer Samplinrate des Akzentsignals $f_r = 172,625$ Hz. Tony Fisher stellt ein Skript zur Verfügung¹⁵, das die Butterworth-Parameter für eine gegebene Ordnung n , Trennfrequenz f_{LP} und Samplingrate f_r berechnet. Zudem generiert es den C-Code für die Berechnung.

Listing 6.6: Butterworth-Filterung in F#

```
let butterworth6thOrder (b:float[]) signal =
    let NZEROS = 6
    let NPOLES = 6
    let GAIN = 5.125482511e+04
    let xv = Array.zeroCreate (NZEROS + 1) // Eingabefenster
    let yv = Array.zeroCreate (NPOLES + 1) // Ergebnishistorie

    signal |> Signal.InPlace.map (fun value =>
        // Fenster weiterschieben
        xv.[0] <- xv.[1]; xv.[1] <- xv.[2]; xv.[2] <- xv.[3]
        xv.[3] <- xv.[4]; xv.[4] <- xv.[5]; xv.[5] <- xv.[6]
        xv.[6] <- value / GAIN

        // Ergebnisse weiterschieben, aktuellen Wert berechnen
        yv.[0] <- yv.[1]; yv.[1] <- yv.[2]; yv.[2] <- yv.[3]
        yv.[3] <- yv.[4]; yv.[4] <- yv.[5]; yv.[5] <- yv.[6]
        yv.[6] <- (xv.[0] + xv.[6]) + 6.0 * (xv.[1] + xv.[5])
            + 15.0 * (xv.[2] + xv.[4])
            + 20.0 * xv.[3]
            + (b.[0] * yv.[0]) + (b.[1] * yv.[1])
            + (b.[2] * yv.[2]) + (b.[3] * yv.[3])
            + (b.[4] * yv.[4]) + (b.[5] * yv.[5]);
        yv.[6])

```

Die Umsetzung in F# ist ein gutes Beispiel für die Integration von iterativem Code. Die wesentlichen Unterschiede zum Original sind die Syntax für Zuweisungen (`<- statt =`) und Array-Indizierungen (`.[] statt []`). Außerdem wurden die Butterworth-Parameter in das Array `b` gekapselt - nur dieses Array müsste bei geänderter Trennfrequenz oder Abtastrate angepasst werden.

¹⁵Tony Fisher war im Informatikfachbereich der Universität von York, England, tätig und verstarb am 29. Februar 2000. Das Skript kann nach wie vor unter <http://www-users.cs.york.ac.uk/~fisher/mkfilter/trad.html> abgerufen werden.

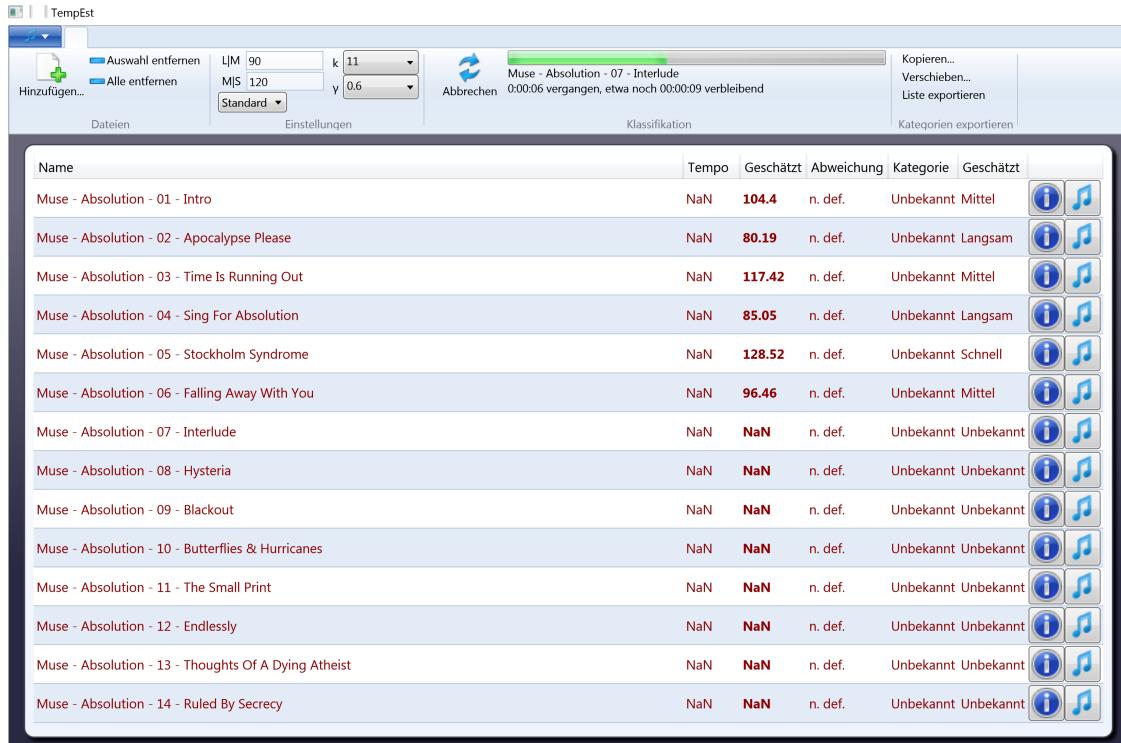


Abbildung 6.5: TempEst während der Berechnung der Tempi eines Albums

Die Signalwerte werden hier durch `Signal.InPlace.map` direkt im Signal verändert, da das ungefilterte Eingangssignal nicht mehr benötigt wird. Durch die Verwendung von `Signal.map` kann das bei Bedarf geändert werden. Es würde dann ein neues Signal erstellt und das Eingangssignal bliebe unverändert.

6.4 Beispielanwendung „TempEst“

Auf Basis der vorgestellten Technologie ist – zusätzlich zur Testumgebung für die Trainingsdaten – die Beispielanwendung „TempEst“ (für „tempo estimator“) zur Schätzung des Tempos beliebiger Audiodateien entstanden.

Aus der Testumgebung können die Einstellungen der Regression zusammen mit den relevanten Daten der Trainingsbeispiele in eine XML-Datei exportiert werden. Sie enthält das optimale k und γ für die Regression und jedes Trainingsbeispiel mit Namen, Tempo und kombiniertem Periodenvektor. Die Audiodaten der Trainingsbeispiele sind an dieser Stelle nicht mehr notwendig, da die Temposchätzung nur auf die obigen Daten angewiesen ist.

TempEst kann Audiodateien in eine Liste aufnehmen und dann das Tempo der Stücke schätzen. Dazu müssen zunächst die Akzente und Perioden aus den Audiodaten extrahiert werden. Die gemeinsam genutzte Signalverarbeitungsbibliothek stellt dafür alle Mittel be-

reit. Das Tempo wird dann aufgrund der Trainingsbeispiele der gespeicherten Regression geschätzt. Für genauere Informationen über die Klassifizierung stehen – wie schon in TempEx – die berechneten Perioden der Akzente und die Nachbarn der Klassifikation zu jedem berechneten Stück zur Verfügung.

Kapitel 7

Ergebnisse

Im vorigen Kapitel wurde die Umsetzung der Verfahrensidee näher dargestellt. Hat sich der Aufwand gelohnt? Die Rechenzeit in akzeptablen Größen zu halten mag aus praktischer Sicht interessant sein. Viel drängender für eine theoretische Betrachtung ist aber die Frage, ob das Verfahren gute Ergebnisse berechnet – ob es sich überhaupt lohnt, mehrere Akzente parallel zu betrachten.

Dieser Frage wird im vorliegenden Kapitel auf den Grund gegangen. Anhand einer Trainingsdatenbank wird die Genauigkeit des vorgestellten Verfahrens untersucht und mit dem Verfahren von Eronen und Klapuri verglichen.

7.1 Trainingsdatenbank

Die Trainingsdatenbank basiert auf 98 Musikstücken verschiedener Genres aus dem Bereich der gängigen Popmusik. Hierzu gehören Rock, Pop, Schlager, Country, Balladen und einige Volksmusikstücke. Ein relativ großer Anteil der Stücke ist deutschsprachig. Die Musikbeispiele wurden vom Musikfachbereich zur Verfügung gestellt und mit den richtigen Tempi gekennzeichnet. Eine vollständige Liste der Stücke findet sich im Anhang.

Aus den Musikstücken wurden für die Arbeit drei Trainingsmengen erzeugt:

- Die vollständigen Musikstücke bilden eine Trainingsmenge. Problematisch ist einerseits die Menge der Daten für die Berechnungszeit der Akzente. Außerdem besteht ein Musikstück aus unterschiedlichen Teilen, die grundverschieden sein können¹ und das Ergebnis eventuell negativ beeinflussen.
- Eine weitere Trainingsmenge bilden 98 Segmente, die der Autor aus den Musikstücken manuell ausgeschnitten hat. Die Segmente sollen den Charakter des Stücks treffen und haben eine individuelle Länge von 30 bis 90 Sekunden. Dies ermöglicht eine

¹Ein extremes Beispiel hierfür wäre etwa „Music“ von John Miles

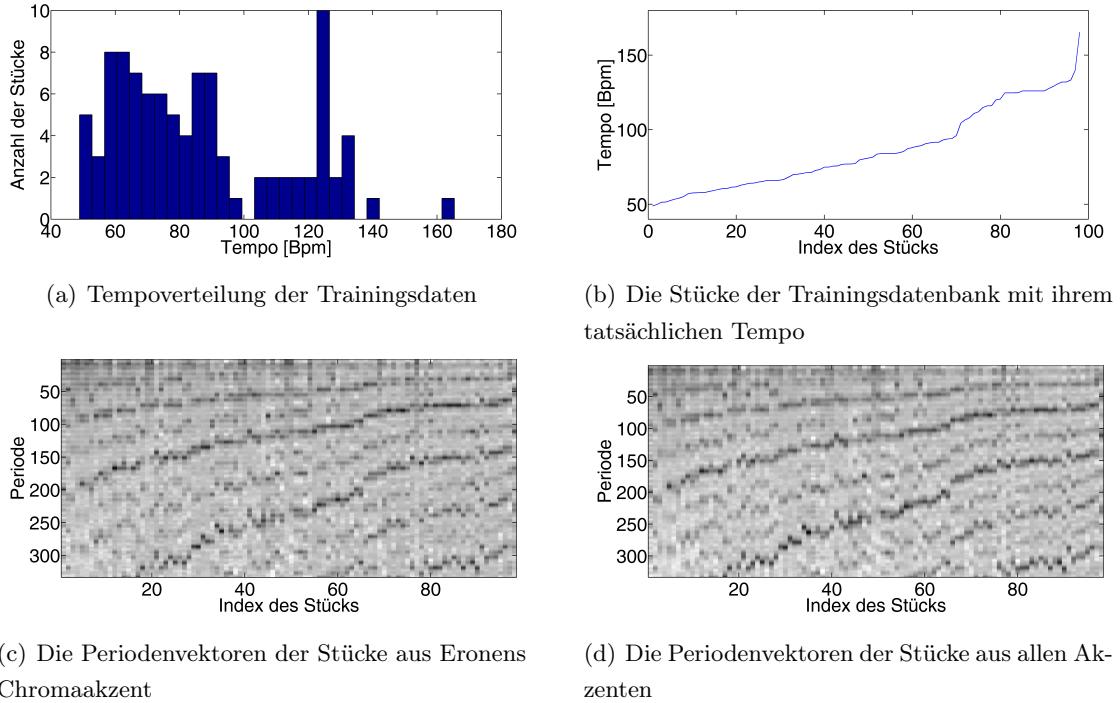


Abbildung 7.1: Die Tempoverteilung der Trainingsdaten und die berechneten Periodenvektoren. Die Stücke sind mit aufsteigendem Tempo angeordnet.

schnellere Berechnung der Akzente. Gleichzeitig wird vermutet, dass die Beschränkung auf charakteristische Segmente zu besseren Klassifikationsergebnissen führt als die Verwendung der kompletten Musikstücke. Daher bildet diese Menge den Kern der folgenden Klassifikation.

- Als zusätzliche Vergleichsmenge wird aus den Musikstücken ein Bereich von 30 Sekunden zur Berechnung der Akzente geladen. Dieser Bereich beginnt willkürlich bei der ersten Minute des Musikstücks. Es soll überprüft werden, ob eine vorhergehende Segmentierung wie bei der zweiten Trainingsmenge sinnvoll ist und wie das willkürliche Beschneiden das Ergebnis im Vergleich zur Berechnung des gesamten Audiosignals beeinflusst.

Tempoverteilung und Größe der Trainingsmenge Wichtige Indikatoren für die Einschätzung der Qualität der Ergebnisse sind die Tempoverteilung und die Größe der Trainingsmenge. Eronen und Klapuri stellen fest, dass sich die Genauigkeit auf seiner Trainingsdatenbank mit 335 Stücken bei einer Reduzierung auf 248 signifikant² verschlechtert, obwohl nur 71 Stücke erforderlich sind, um eine Genauigkeit von über 70% zu erzielen [17, S. 55].

²Die genauen Werte liefern sie nicht, aber der Abbildung lässt sich entnehmen, dass die Genauigkeit von 79% auf etwa 75% sinkt.

Genauigkeit	ϕ Abweichung	Gewichtung	k	γ	Bereinigung
79,6%	7,84%	[0,36 0,26 0,28 0,10]	6	1,15	
74,5%	11,61%	[0,28 0,22 0,35 0,15]	7	0,75	•
71,4%	12,71%	Eronen u. Klapuri [1 0 0 0]	6	0,45	
67,4%	15,48%	Eronen u. Klapuri [1 0 0 0]	11	0,15	•

Tabelle 7.1: Genauigkeit der Klassifikation von 98 manuell erzeugten Segmenten

Damit sind bei 98 Trainingsdaten ordentliche Ergebnisse zu erwarten, wenn „die Referenzbeispiele den Tempobereich gleichmäßig überspannen“ [17, S. 55]. Die Abbildung 7.1 verdeutlicht ein Manko der vorliegenden Trainingsdatenbank, da der Bereich zwischen 90 Bpm bis 110 Bpm völlig unterrepräsentiert ist und nur zwei schnelle Stücke (mit Tempo über 140 Bpm) vorhanden sind.

7.2 Manuell segmentierte Ausschnitte

Die Ergebnisse werden, wie schon in [17], durch eine Leave-One-Out Kreuzvalidierung berechnet. Man nimmt jedes Stück einmal aus der Trainingsmenge und lässt das Tempo des Stücks auf den verbleibenden Trainingsdaten klassifizieren. Das Stück ist korrekt klassifiziert, wenn das geschätzte Tempo innerhalb eines Toleranzbereichs vom wirklichen Tempo (oft „*ground truth*“ genannt) abweicht. Die Abweichung wird berechnet durch

$$\frac{|T_{est} - T_{ann}|}{T_{ann}}, \quad (7.1)$$

wobei T_{est} das geschätzte und T_{ann} das wirkliche Tempo darstellt. Der Toleranzbereich ist, wie in [32], [23] und [17], auf 4% festgelegt.

Zunächst werden die Akzente und Perioden aller Trainingsstücke berechnet, damit diese Daten für die Klassifikation zur Verfügung stehen. Die Periodenvektoren der Akzente werden gewichtet zu einem repräsentativen Periodenvektor aufsummiert. Die Gewichtung wurde für die unterschiedlichen Trainingsmengen optimiert³. Die Parameter der Akzentanalyse und Periodenschätzung sind in Kapitel 5.2 beschrieben und bleiben konstant. Ausgenommen ist die Untersuchung der Bereinigung des Spektrums (Abschnitt 7.2.2).

Anschließend wird für jedes Stück eine k -NN-Regression auf den anderen Trainingsdaten durchgeführt. Dabei werden für k Werte zwischen 3 und 15 eingesetzt und Gewichte γ in Schritten von 0,05 zwischen 0,1 und 2,0 angenommen. Die Kombination von k und γ , die die relativen Abweichung im Mittel minimiert, ist in der Tabelle 7.1 aufgeführt.

Die Klassifikation auf dem gewichtet summierten Periodenvektor erkennt 78 von 98 Stücken korrekt innerhalb der gegebenen Toleranz. Dies entspricht der in der Tabelle 7.1

³Für die Optimierung wurden systematisch verschiedene Gewichtungen der Akzente durchgerechnet. Es wurde die Gewichtung gewählt, die die gemittelte relative Abweichung auf der Trainingsmenge minimiert.

	langsam	mittel	schnell
langsam	96,4%	1,8%	1,8%
mittel	12,9%	87,1%	0,0%
schnell	30,0%	20,0%	50,0%

Tabelle 7.2: Konfusionsmatrix: Betrachtung aller Akzente für die Kategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 130 Bpm) und schnell (ab 130 Bpm).

	langsam	mittel	schnell
langsam	91,2%	8,8%	0,0%
mittel	22,6%	77,4%	0,0%
schnell	20,0%	40,0%	40,0%

Tabelle 7.3: Konfusionsmatrix: Tiefer Chromaakzent für die Tempokategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 130 Bpm) und schnell (ab 130 Bpm).

aufgeführten Genauigkeit von 79,6%. Die besten Klassifikationsergebnisse werden mit $k = 6$ betrachteten Nachbarn und der Gewichtung $\gamma = 1,15$ erzielt. Welche Parameter für die Temposchätzung auf einer unabhängigen Menge von Musikstücken bessere Ergebnisse liefert, muss durch Tests gezeigt werden, da diese Werte für die Trainingsmenge optimiert sind.

Wird nur der tiefe Chromaakzent verwendet, werden 70 Stücke – das entspricht 71,4% – korrekt erkannt. Die Periodenvektoren des Akzents können aus der berechneten Trainingsmenge übernommen werden, da das tiefe Chroma in diesem Verfahren genau den Akzent aus [17] darstellt. Das beste Ergebnis liefert hier auch eine Betrachtung der $k = 6$ nächsten Nachbarn. Das Ergebnis liegt ungefähr in dem Bereich, den man aufgrund der Beschaffenheit der Trainingsmenge erwarten kann.

Vergleicht man die Ergebnisse der Verfahren in Tabelle 7.1, gelangt man zu dem Schluss, dass die Berücksichtigung verschiedener Akzente bei der Klassifikation positive Auswirkungen auf die Genauigkeit der Ergebnisse hat und somit einen guten Ansatz zur Verbesserung der Tempoerkennung durch Klassifikation darstellt. Neben der besseren Genauigkeit ist auch die durchschnittliche relative Abweichung signifikant geringer. Weichen bei dem Verfahren aus [17] noch 21 Stücke um mehr als 10% vom tatsächlichen Tempo ab, sind es mit der hier vorgestellten Kombination nur 11 Stücke. Drei weitere Aspekte der Ergebnisse werden nun genauer diskutiert.

7.2.1 Klassifikation in Tempokategorien

Oft stellt sich die Aufgabe, nicht das Tempo als Zahl zu schätzen, sondern ein Musikstück in eine Tempokategorie einzurordnen. Wenn man vorher die Tempokategorien definiert,

ist es leicht, das berechnete Tempo in eine Kategorie einzuordnen. Hier werden zunächst die in [17] beschriebenen Tempokategorien verwendet, um eine gewisse Vergleichbarkeit zu erzielen: Langsam (0 bis 90 Bpm), mittel (90 bis 130 Bpm) und schnell (ab 130 Bpm).

Die Fehlklassifikationsraten der Kategorien („*confusion matrix*“) liefern wertvolle Informationen über die Art der Fehler und die Schwachstellen des Verfahrens. Die Tabelle 7.2 zeigt die Genauigkeit bzw. die Fehlklassifikationen auf den einzelnen Kategorien für den gewichteten Periodenvektor der Akzente. Die Tabelle 7.3 zeigt die Kategoriefehler des Verfahrens nach [17] im Vergleich. Auch bei der Klassifikation in Tempokategorien erhöht die gemeinsame Klassifikation unterschiedlicher Akzente die Genauigkeit.

Für die Interpretation der Fehlklassifikationen muss man sich die Ergebnisse im Detail ansehen. Auffällig sind zunächst die sehr guten Werte für langsame Musikstücke, die zu über 96% richtig eingeschätzt werden. Da nur 2% der langsamen Musikstücke als schnell klassifiziert werden, könnte man meinen, dass das Phänomen der Tempoverdoppelung im Prinzip nicht auftritt. Betrachtet man die einzelnen Ergebnisse, bei denen langsame Lieder falsch klassifiziert wurden, stellt man jedoch fest, dass in allen Fällen das geschätzte Tempo eine Verdoppelung des wirklichen Tempos darstellt. So wird Claudia Jungs „Amore Amore“ mit 126 Bpm klassifiziert, obwohl es mit 63 Bpm angegeben ist. Durch das sehr langsame Tempo und die Einteilung der Kategorien bleiben solche Lieder bei einer Verdoppelung des Tempos in der mittleren Kategorie.

Das soll deutlich machen, dass die Fehlklassifikationsraten auf den Kategorien bei einem Verfahren, dass auf kontinuierlichen Tempowerten rechnet, mit Sorgfalt interpretiert werden müssen. Zum einen werden teilweise große Abweichungen als korrekt toleriert, sofern das wirkliche und geschätzte Tempo innerhalb der Grenzen derselben Kategorie bleibt. Das Beispiel in der Trainingsmenge hierfür ist „Je t'aime mon amour“, wiederum von Claudia Jung, das mit 66 Bpm statt 48 Bpm (37,5% Abweichung) immer noch in die Kategorie „langsam“ fällt.

Andererseits wandern Stücke über die Kategoriegrenzen, die nur eine geringe Abweichung vom wirklichen Tempo aufweisen. Beispiele hierfür sind die beiden schnellen Stücke, bei denen ein mittleres Tempo geschätzt wird. „Feels Like Tonight“ von Daughtry mit 132 Bpm wird auf 127,7 Bpm geschätzt, die Abweichung ist damit 5,5% und die Toleranzgrenze für korrekt erkannte Stücke wurde nur knapp verfehlt. „Erdbeben“ von Fettes Brot wird mit einer Abweichung von nur 4% sogar korrekt erkannt, liegt aber mit 130 Bpm genau auf der Kategoriegrenze. Jede noch so kleine Abweichung nach unten hätte das Stück in die falsche Kategorie eingeordnet. Weiche Kategoriegrenzen könnten hier eine sinnvolle Lösung darstellen.

Vor diesem Hintergrund soll nun genauer auf schnelle Musik eingegangen werden, bei der die Tabellen auf viele Fehler hindeuten. Neben der erwähnten Starrheit ist die Einteilung der Tempokategorien an sich problematisch. Von 98 Stücken sind nur 10 schnell, und damit hoffnungslos unterrepräsentiert. Um eine etwas objektivere Bewertung der Genauig-

	langsam	mittel	schnell
langsam	96,5%	0,0%	3,5%
mittel	11,1%	88,9%	0,0%
schnell	21,7%	0,0%	78,3%

Tabelle 7.4: Konfusionsmatrix: Klassifizierung auf allen Akzenten in die Kategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).

	langsam	mittel	schnell
langsam	91,2%	0,0%	8,8%
mittel	33,3%	66,7%	0,0%
schnell	13,0%	4,4%	82,6%

Tabelle 7.5: Konfusionsmatrix: Tiefer Chromaakzent mit den Tempokategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).

keit bei schnellerer Musik zu erhalten, wird im folgenden die Grenze zwischen mittlerem und schnellem Tempo auf 120 heruntergestuft. Damit stehen dort 23 Stücke zur Verfügung. Die Tabelle 7.4 illustriert die Kategoriefehler mit der neuen Einteilung, und schnelle Musik befindet sich in etwa auf dem Niveau der anderen Kategorien. Ein Vergleich mit der Tabelle 7.5 zeigt, dass die Ergebnisse durch die Verwendung mehrerer Akzente auf langsamem und mittleren Stücken an Genauigkeit gewinnen, schnelle Stücke aber zur einer Halbierung des Tempos neigen.

7.2.2 Bereinigung des Spektrums

Auf eine interessante Beobachtung wurde schon zu Beginn von Kapitel 7.2 angespielt. Die Gewichtung ist für die bereinigten und unbereinigten Akzente jeweils separat optimiert, um einen fairen Vergleich zu ermöglichen. Erstaunlich ist: Beide Verfahren werden um 4-5% genauer, wenn das Spektrum vor der Chromaanalyse *nicht* bereinigt wird. Analog sinkt die relative Abweichung, wenn auf die Bereinigung verzichtet wird.

	langsam	mittel	schnell
langsam	89,5%	5,3%	5,3%
mittel	16,7%	83,3%	0,0%
schnell	21,7%	0,0%	78,3%

Tabelle 7.6: Konfusionsmatrix: Alle vier Akzente unter Bereinigung des Spektrums. Tempokategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).

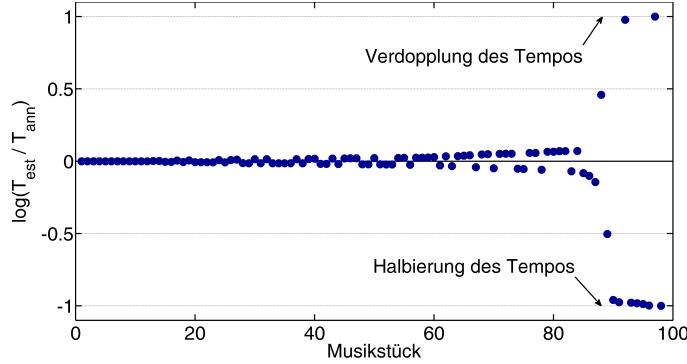


Abbildung 7.2: Der logarithmische Fehler verdeutlicht Verdoppelungen und Halbierungen des Tempos.

Betrachtet man nun die Kategoriefehler und vergleicht die Tabellen des vorigen Abschnitts mit der Tabelle 7.6, werden auch Kategorien durch die aktivierte Bereinigung schlechter erkannt. Ein weiteres Indiz dafür, dass die Bereinigung des Spektrums die Temposchätzung negativ beeinflusst.

Auf die Bereinigung wurde zunächst verzichtet, da sie Zeit kostet und die Ergebnisse der Berechnung nur schwer validierbar sind. Es sollte also eine mögliche Fehlerquelle bei den ersten Berechnungen ausgeschlossen werden. Es zeigt sich jedoch, dass die Ergebnisse durchweg besser sind – *und* sich effizienter berechnen lassen – wenn die Bereinigung bewusst deaktiviert wird.

Erklären lässt sich diese Beobachtung nur schwer. Das Ziel der Chromaanalyse ist hier ein anderes als in Klapuris Artikel [31], wo eine oder mehrere Grundfrequenzen möglichst exakt bestimmt werden sollen. Dort wurde die Bereinigung durchgeführt, um die Unabhängigkeit der Schätzung von der Klangfarbe bestimmter harmonischer Schallquellen zu gewährleisten. Das Ziel der Chromaanalyse zur Tempoerkennung hingegen ist, möglichst genau Harmoniewechsel zu erkennen, nicht die genauen Frequenzen.

Die Unterschiede zeigen sich bei vollständigen Audiodaten (Tabelle 7.7) weniger ausgeprägt als bei kurzen Ausschnitten (Tabelle 7.9) und segmentierten Stücken. Man kann bei letzteren davon ausgehen, dass der Klangcharakter des Stücks relativ konstant ist. Die Berücksichtigung dieses Charakters scheint die Klassifikation positiv zu beeinflussen.

7.2.3 Exemplarische Untersuchung der Fehlklassifikationen

Wie schon ausgeführt, weichen bei dem vorgestellten Verfahren nur 11 Stücke um mehr als 10% vom tatsächlichen Tempo ab. Die Art der Fehler lässt sich durch ein logarithmisches Fehlermaß verdeutlichen, wie es in [23] verwendet wird:

$$e = \log\left(\frac{T_{est}}{T_{ann}}\right) \quad (7.2)$$

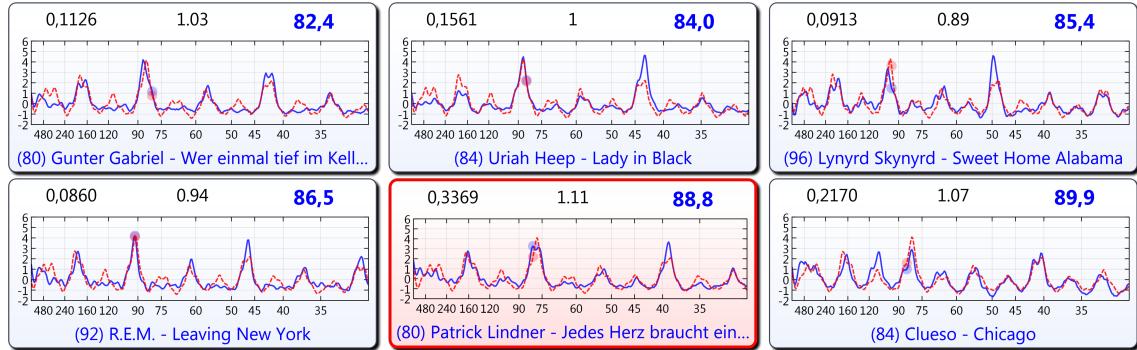


Abbildung 7.3: Die Klassifikation von „Roll Over Beethoven“, entnommen aus TempEx. Die Nachbarn sind nach aufsteigendem Tempo angeordnet, der Median wird umrahmt. Der Periodenvektor des Nachbarn ist blau dargestellt, der zu klassifizierende Vektor ist rot gestrichelt.

Hierbei bezeichnet T_{ann} das tatsächliche und T_{est} das vom Verfahren geschätzte Tempo. Bei $e = 1$ hat das Verfahren das doppelte des tatsächlichen Tempos geschätzt, bei $e = -1$ die Hälfte. Die Grafik 7.2 zeigt den Fehler e für die Stücke der Trainingsmenge an. Die Stücke sind aufsteigend nach $|e|$ angeordnet. Auffällig ist, dass das Tempo bei schnellen Stücken häufiger halbiert wird als bei langsamen verdoppelt. Das führt zu der ersten Ursache der Fehlklassifikationen des Verfahrens: Es sind zu wenig schnelle Stücke in der Trainingsmenge.

Qualität der Trainingsmenge Der Erfolg der k -NN-Regression hängt davon ab, ob die Trainingsmenge Stücke enthält, die dem zu klassifizierenden Stück ähneln. Insbesondere müssen auch Trainingsbeispiele mit ähnlichem Tempo vorhanden sein, da auch das Resampling die Trainingsvektoren nicht um einen beliebig großen Faktor ausdehnt. Die Trainingsmenge enthält mit „Roll Over Beethoven“ ein Beispiel, dass die Forderung nach einer ausgeglichenen Tempoverteilung unterstreicht. Mit 176 Bpm ist es das schnellste Stück der Datenbank. Selbst ein Resampling mit dem Faktor 0,87 auf etwa 153 Bpm liegt noch außerhalb der Tempi, die von der Datenbank abgedeckt werden (das zweitschnellste Stück ist mit 141 Bpm gekennzeichnet). Es ist damit auf dieser Trainingsmenge *unmöglich*, zu einer genauen Schätzung des Tempos für „Roll Over Beethoven“ zu kommen. Daher ist das geschätzte Tempo von etwa 88 Bpm sogar die musikalisch plausibelste Schätzung, die auf der Trainingsmenge möglich ist⁴. Alle gefundenen Nachbarn liegen in diesem Tempobereich, und die Abbildung 7.3 zeigt die Informationen, die TempEx zu der Klassifikation liefert.

Mehrdeutige Temposchätzung Das zweite untersuchte Beispiel ist „Amore Amore“ von Claudia Jung und gehört zu den wenigen Stücken, bei denen das Tempo verdoppelt

⁴Interessant ist am Rande, dass „Roll Over Beethoven“ auf einer größeren Trainingsdatenmenge korrekt mit 177,6 Bpm geschätzt wird. In dieser Menge finden sich einige ähnlich schnelle Stücke, und sie dient am Ende dieses Kapitels als Vergleich.

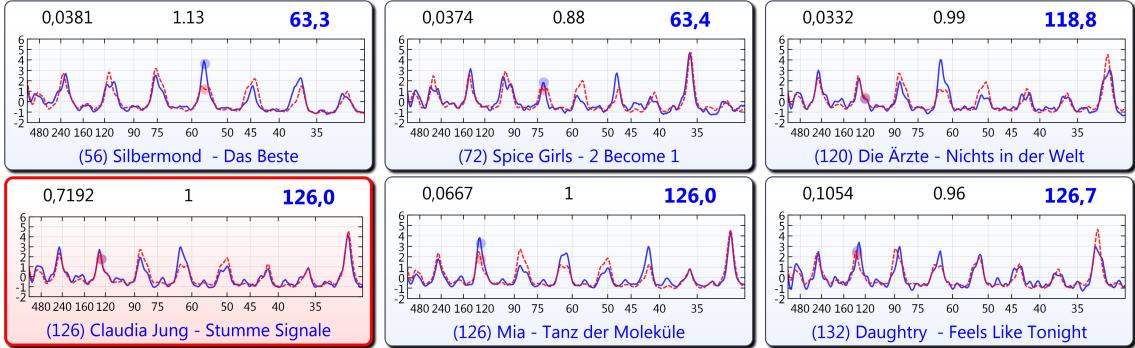


Abbildung 7.4: Die Klassifikation von „Amore Amore“ in TempEx. Links oben sieht man das Gewicht bei der Berechnung des Median (0,7192 für „Stumme Signale“).

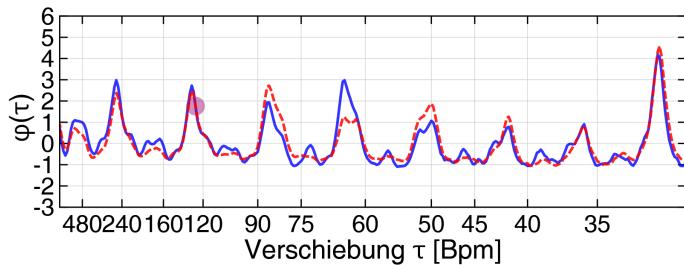


Abbildung 7.5: Die Periodenvektoren von „Amore Amore“ (rot gestrichelt) und „Stumme Signale“ (blau) im Vergleich

wird. Die Übersicht in TempEx (Abbildung 7.4 macht deutlich, dass die Klassifikation auf „Stumme Signale“ als Trainingsbeispiel zurückgeht. Das Gewicht des Beispiels ist mit etwa 0,72 sehr hoch, die anderen Stücke haben praktisch keinen Einfluss auf das Ergebnis. Die Ähnlichkeit der Periodenvektoren ist in der Abbildung 7.5 noch deutlicher zu erkennen.

Die Ähnlichkeit der beiden Stücke beschränkt sich aber nicht auf ihre Periodenvektoren. Sie sind nicht nur von der gleichen Sängerin, sondern klingen auch sehr ähnlich. Der Autor dieser Arbeit hat unwillkürlich im gleichen Tempo weitergeklopft, wenn er „Amore Amore“ nach „Stumme Signale“ gehört hat. Andere Personen gaben unterschiedliche Meinungen darüber ab, ob eines der beiden Stücke schneller ist – und wenn ja, welches. Auch Menschen sind sich manchmal nicht einig darüber, was die richtige metrische Ebene für das Tempo ist. Die Folge ist, dass eine Fehlklassifikation in solchen Fällen nicht am Verfahren liegt, denn es findet ja die Ähnlichkeit der beiden Stücke auf bemerkenswerte Weise. Das gekennzeichnete Tempo ist hier diskutabel und stellt somit die zweite Fehlerquelle dar.

Verfahrensfehler Die dritte Quelle von Fehlern ist das Verfahren selbst. Als Beispiel hierfür soll „I Kissed A Girl“ von Katy Perry dienen. Das Tempo ist mit 126 Bpm zweifelsfrei richtig annotiert, aber das Stücks wird auf etwa 64 Bpm geschätzt. Betrachtet man die Klassifikation in TempEx genauer (Abbildung 7.6), haben fünf der sechs Nachbarn ein

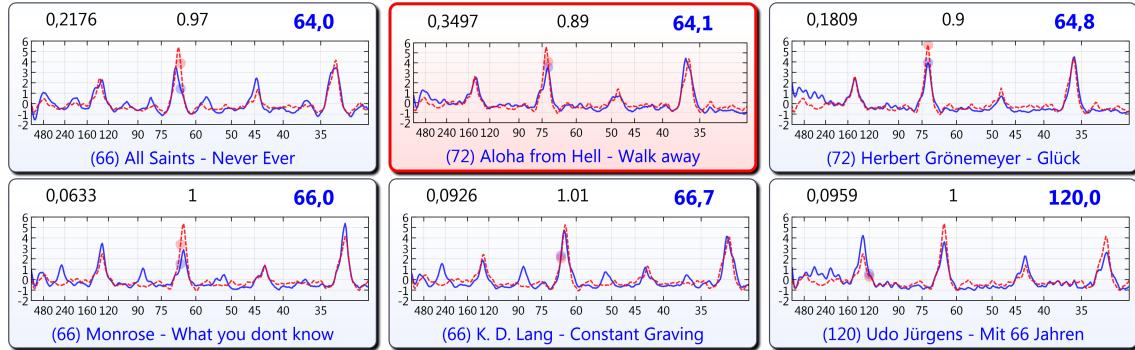


Abbildung 7.6: Die Klassifikation von „I Kissed a Girl“ in TempEx. Dieses schnelle Stück findet fast nur langsame Nachbarn.

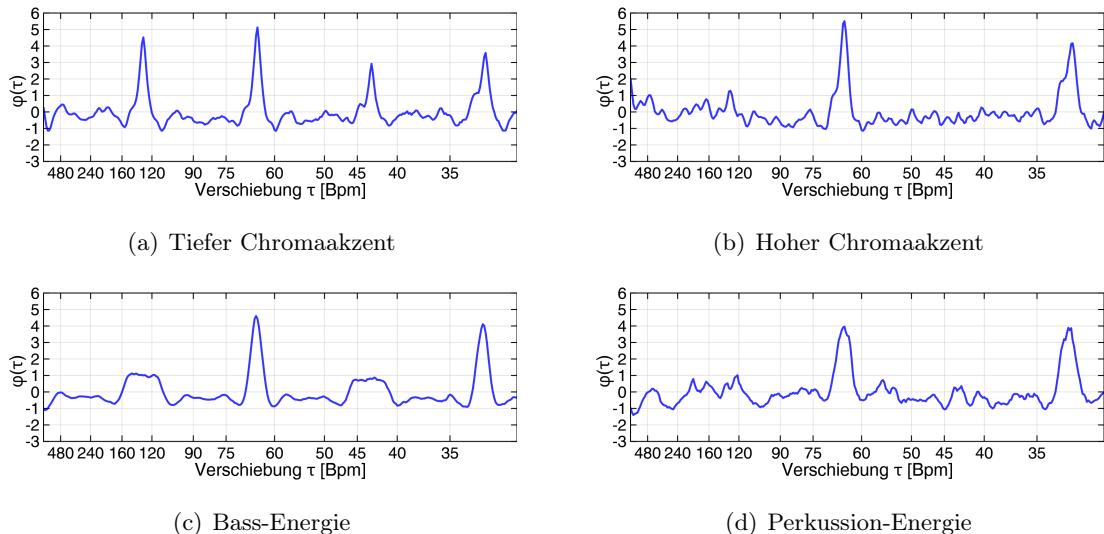


Abbildung 7.7: Die Periodenvektoren für die vier Akzente von „I Kissed A Girl“

langses Tempo. Nur Udo Jürgens versucht die Klassifikation zu retten, aber dafür ist das Stück zu schwach gewichtet.

Der Fehler resultiert daraus, dass das Verfahren einen für das schnelle Tempo untypischen Periodenvektor erzeugt. In Abbildung 7.7 sind die Periodenvektoren der einzelnen Akzente zu sehen. Die Periodenvektoren aller Akzente zeigen eindeutige Maxima bei etwa 65 Bpm. Das Regressionsverfahren soll zwar die Interpretation des Periodenvektors übernehmen, aber es versagt für Katy Perry auf der betrachteten Trainingsmenge.

7.3 Vollständige Musikstücke

Die Ergebnisse auf der segmentierten Trainingsmenge sollen nun mit der Analyse des ganzen Liedes verglichen werden. Es soll abgeschätzt werden, welche Audiodaten man am

Genauigkeit	ϕ Abweichung	Gewichtung	k	γ	Bereinigung
80,6%	8,75%	[0,50 0,10 0,30 0,10]	3	0,30	
78,6%	10,21%	[0,26 0,10 0,40 0,24]	7	0,10	•
72,5%	13,13%	Eronen u. Klapuri [1 0 0 0]	11	0,85	
69,4%	12,91%	Eronen u. Klapuri [1 0 0 0]	7	0,10	•

Tabelle 7.7: Genauigkeit der Klassifikation von 98 vollständigen Audiodateien

	langsam	mittel	schnell
langsam	94,7%	1,8%	3,5%
mittel	11,1%	88,9%	0,0%
schnell	21,7%	0,0%	78,3%

Tabelle 7.8: Konfusionsmatrix: Alle Akzente bei 98 vollständigen Stücken mit den Kategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).

besten für die Berechnung unbekannter Stücke heranzieht. Lohnt es sich, die Stücke vorher zu segmentieren? Oder ist es besser, die gesamten Stücke zu analysieren?

Zur Beantwortung dieser Fragen wird die Trainingsmenge von 98 Musikstücken in voller Länge analysiert und klassifiziert. Wie für die segmentierten Trainingsdaten wird eine Leave-One-Out Kreuzvalidierung durchgeführt, und das Tempo der Stücke darf höchstens 4% vom wirklichen Tempo abweichen, um als korrekt angesehen zu werden.

Die Ergebnisse der Klassifikation sind in Tabelle 7.7 dargestellt. Es zeigt sich, dass die Ergebnisse auf den vollständigen Musikstücken nicht signifikant von den Ergebnissen auf den Segmenten abweichen. Obwohl die durchschnittliche relative Abweichung bei den unbereinigten Akzenten schlechter ist, wurden sogar mehr Stücke korrekt klassifiziert. Die Klassifikation in die Kategorien liefert ebenfalls vergleichbar gute Ergebnisse.

7.4 Willkürlich beschnittene Stücke

Die dritte Trainingsmenge besteht aus 98 Ausschnitten, die willkürlich auf 30 Sekunden beschnitten wurden. Nach den überraschenden Ergebnissen bei den vollständigen Audi-

Genauigkeit	ϕ Abweichung	Gewichtung	k	γ	Bereinigung
82,7%	7,57%	[0,38 0,15 0,32 0,15]	11	0,85	
78,6%	10,75%	[0,35 0,15 0,30 0,20]	3	0,60	•
72,5%	10,56%	Eronen u. Klapuri [1 0 0 0]	12	0,30	
67,4%	14,02%	Eronen u. Klapuri [1 0 0 0]	10	0,80	•

Tabelle 7.9: Genauigkeit der Klassifikation auf 98 Ausschnitten mit 30 Sekunden Länge

	langsam	mittel	schnell
langsam	94,7%	1,8%	3,5%
mittel	0,0%	100,0%	0,0%
schnell	26,1%	0,0%	73,9%

Tabelle 7.10: Konfusionsmatrix: Alle Akzente bei 98 Ausschnitten. Die Tempokategorien sind langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).

Genauigkeit	ϕ Abweichung	Gewichtung	k	γ
85,5%	7,75%	[0,20 0,20 0,35 0,25]	3	0,85
79,1%	11,48%	Eronen u. Klapuri [1 0 0 0]	3	0,15

Tabelle 7.11: Genauigkeit der Klassifikation von Ausschnitten aus 297 Stücken

odaten zeigt auch die Leave-One-Out Klassifikation auf den Ausschnitten erstaunlich gute Ergebnisse. Gemäß der Tabelle 7.9 werden auf der Trainingsdatenbank bei der Analyse willkürlicher Ausschnitte bessere Ergebnisse erzielt als bei den manuell ausgeschnittenen repräsentativen Segmenten. Die Genauigkeit des kombinierten Verfahrens steigt auf annähernd 83% und ist auf dieser Trainingsmenge um 10% höher als beim einzelnen Chromaakzent – unabhängig davon, ob das Spektrum vorher bereinigt wurde.

Ob eine vorausgehende Segmentierung der Stücke sinnvoll ist, kann hier nicht abschließend beurteilt werden. Dazu müssten umfangreichere Untersuchungen durchgeführt werden⁵. Es wird auf der Trainingsmenge aber deutlich, dass die Klassifikation auf relativ kurzen Ausschnitten der Musikstücke erstaunlich gute Ergebnisse liefert – bessere Ergebnisse als die Analyse der vollständigen Musikstücke, und das unter Reduzierung der Rechenzeit.

7.5 Auswirkung der Größe der Trainingsmenge

Kurz vor Abschluss der Arbeit bestand noch die Möglichkeit, eine Leave-One-Out-Klassifikation auf einer bedeutend größeren Testdatenbank durchzuführen. Die Datenbank besteht aus 297 Ausschnitten von wiederum 30 Sekunden Länge, und das Tempo verteilt sich wie in Abbildung 7.8 dargestellt.

Die Durchführung der Klassifikation folgt dem Muster der vorangegangenen Tests und führt zur Tabelle 7.11. Wie erwartet steigt die Genauigkeit des Verfahrens durch die größere Menge an Vergleichsdaten, wie die Tabelle 7.11 erkennen lässt. Bei über 85% der Stücke liegen die Abweichung des geschätzten Tempos bei höchstens 4%. Auch dieser Test bestätigt, dass die Betrachtung mehrerer Akzente Vorteile bringt, denn nur mit dem tiefen

⁵Außerdem zweifelt der Autor nach dieser Analyse an seiner Fähigkeit, repräsentative Segmente in Musikstücken zu finden.

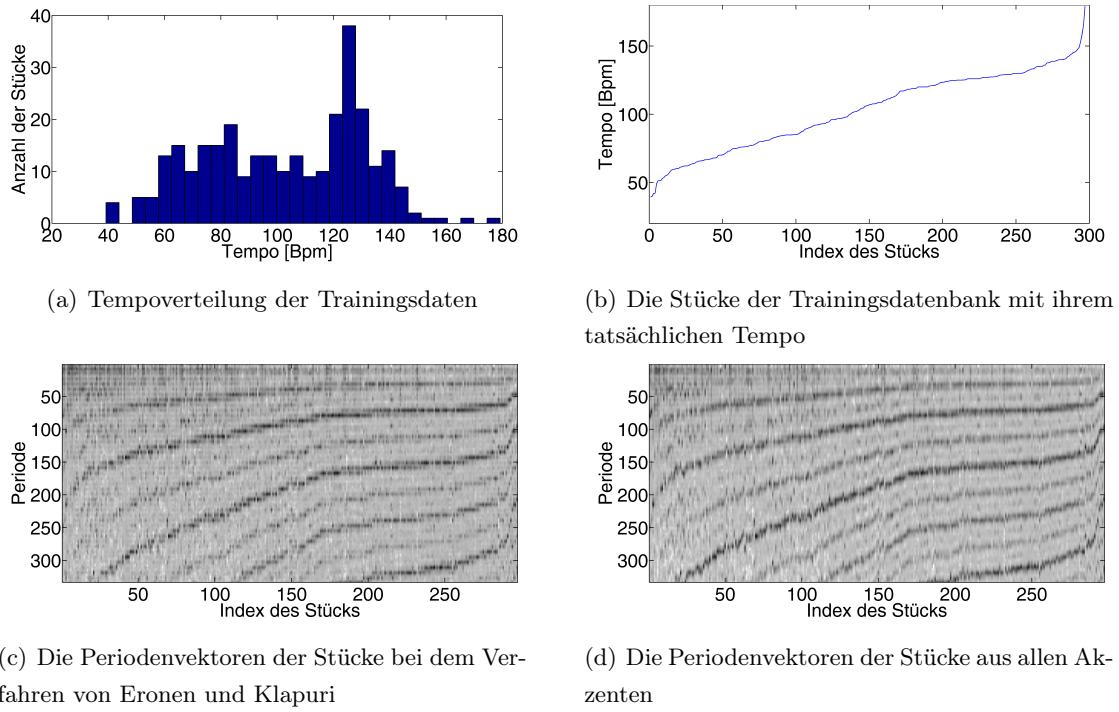


Abbildung 7.8: Die Tempoverteilung der Trainingsdaten und die Periodenvektoren der vergrößerten Trainingsdatenbank

	langsam	mittel	schnell
langsam	87,6%	6,2%	6,2%
mittel	5,8%	91,3%	2,9%
schnell	4,3%	3,5%	92,2%

Tabelle 7.12: Konfusionsmatrix: Alle Akzente bei 297 Ausschnitten. Die Tempokategorien sind langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).

Trainingsmenge	Genauigkeit	ϕ Abweichung	k	γ
98 manuelle Segmente	66,7%	15,00%	6	1,15
98 ganze Stücke	73,3%	7,11%	3	0,30
98 Ausschnitte	66,7%	17,20%	11	0,85
297 Ausschnitte	66,7%	19,04%	3	0,85
Vergleich: 297 Ausschnitte	93,3%	8,19%	11	0,60

Tabelle 7.13: Genauigkeit der Klassifikation unbekannter Musikstücke auf unterschiedlichen Trainingsmengen

Chromaakzent werden etwa 79% der Stücke korrekt klassifiziert. Auf eine Bereinigung des Spektrums wurde an dieser Stelle verzichtet.

Die Kategoriefehler auf der großen Trainingsmenge sind relativ ausgewogen, wie der Tabelle 7.12 zu entnehmen ist. Sie zeigen aber auch, dass die Art der Fehler stark von der Trainingsmenge abhängt: Langsame Stücke werden hier häufiger falsch eingeschätzt als schnelle Stücke, ganz im Gegensatz zur kleineren Trainingsdatenbank. Die Trainingsmenge nimmt demnach starken Einfluss auf die Qualität der Ergebnisse.

7.6 Klassifikation unbekannter Musikstücke

Die besten Optimierungen auf den Trainingsdaten sind nutzlos, wenn sie in der Praxis zur Klassifikation unbekannter Musikstücke ungeeignet sind. Daher wird zum Abschluss des Kapitels überprüft, wie das Verfahren auf 15 unbekannten Musikstücken abschneidet. Unbekannt bedeutet hier, dass die Klassifikationsparameter und die Gewichtung der Periodenvektoren nicht für diese Stücke optimiert sind.

Als Trainingsmenge für die Klassifikation werden alle vier vorgestellten Trainingsmengen gegeneinander getestet. Die optimale Gewichtung der Periodenvektoren und Klassifikationsparameter wird aus dem jeweiligen Leave-One-Out Test übernommen, und das Spektrum wird grundsätzlich nicht bereinigt. Die Stücke finden sich im Anhang A.2.

Lässt man die letzte Zeile der Tabelle 7.13 außer Acht, erscheinen die Ergebnisse zunächst ernüchternd. Es werden nur zwei Drittel der Stücke korrekt klassifiziert, die mittlere Abweichung nimmt bei größerer Trainingsmenge sogar zu, und die besten Ergebnisse liefert die Betrachtung vollständiger Stücke.

Die letzte Zeile erklärt diese Beobachtung. Hier wurden das Tempo wieder mithilfe der großen Trainingsmenge geschätzt, aber es wurden andere Klassifikationsparameter verwendet. Die Periodenvektoren wurden mit der Gewichtung [0,35 0,20 0,30 0,15] summiert, und die Anzahl der betrachteten Nachbarn auf $k = 11$ erhöht, um der größeren Menge an Beispieldaten Rechnung zu tragen. Mit diesen Einstellungen wird nur ein Stück – „In Too

Deep“ von Genesis – nicht korrekt auf 52 Bpm geschätzt. Das geschätzte Tempo von 103,8 Bpm ist eine Verdoppelung des Tempos.

Die Schlussfolgerung ist, dass sich gute Werte einer optimierten Klassifikation der Trainingsmenge nicht einfach auf unbekannte Stücke übertragen lassen. Das Verfahren kann mit den richtigen Einstellungen sehr gute Klassifikationsergebnisse liefern. Die richtigen Einstellungen für einen breiten Praxiseinsatz müssen aber noch ermittelt werden.

Das hier vorgestellte Verfahren, bei dem verschiedene Akzente vor der Klassifikation kombiniert werden, liefert also durchweg bessere Ergebnisse als die Betrachtung eines einzelnen Chromaakzents. Es zeigt sich, dass die Genauigkeit von der Qualität der Trainingsmenge abhängt. Die Trainingsmenge muss für einen praktischen Einsatz des Verfahrens gut geplant und unter Umständen manuell segmentiert sein. Die Klassifikationsparameter für unbekannte Stücke sind dann für die jeweiligen Anforderungen und die Trainingsmenge herauszufinden.

Außerdem erzielt das Verfahren schon mit relativ kleinen Trainingsmengen gute Ergebnisse. Die Verbesserung der Genauigkeit durch das Betrachten mehrerer Akzente ist bei der kleineren Trainingsmenge sogar auffälliger als bei dem zuletzt durchgeföhrten Test auf der vergrößerten Datenbank.

Kapitel 8

Fazit und Ausblick

„Fassen wir alles zusammen, so kommen wir zu dem Ergebnis...“ [58, Pre. 12:13a]

In dieser Diplomarbeit wurde das Problem der Tempoerkennung grundlegend definiert. Anschließend wurde eingehend betrachtet, was Audiosignale sind, wie sie verarbeitet werden und wie aus Audiosignalen auf unterschiedliche Weise ein Maß für die musikalische Akzentuierung berechnet werden kann.

Großer Wert wurde darauf gelegt, die Ideen aufzuzeigen, die verschiedenen Verfahren zur Tempo- und Beaterkennung zugrunde liegen. Die betrachteten Verfahren stellen den aktuellen wissenschaftlichen Stand im Feld der Metrikanalyse dar. Der erste Teil dieser Arbeit bildet damit einen umfassenden Einstieg in diesen Bereich und liefert viele Grundlagen für das Verständnis der angeführten Arbeiten.

Des Weiteren wurde untersucht, welches Verfahren als optimale Grundlage für die weiteren Untersuchungen dienen kann, und die *k*-NN-Regression wurde gewählt. Das Verfahren wurde unter der Hypothese verallgemeinert, durch die Einbeziehung mehrerer musikalisch motivierter Akzente die Klassifikation genauer und ausgewogener gestalten zu können.

Umgesetzt wurde das verallgemeinerte Verfahren zunächst in einer (nicht mehr vollständigen) Referenzimplementierung in Matlab. Anschließend wurde eine Testumgebung auf Basis des .NET-Frameworks in den Programmiersprachen F# und C# erstellt. Diese Umgebung sorgt für akzeptable Rechenzeiten und ermöglicht die systematische Untersuchung verschiedener Einstellungen der Extraktion der Akzentsignale, der Berechnung der Periodenvektoren und der Klassifikation. Die Ergebnisse lassen sich visuell darstellen, um so das Verständnis für das Problemfeld der Tempoerkennung zu vergrößern. Außerdem ist das Programm „TempEst“ entstanden, dass auf den Modulen der Testumgebung aufsetzt, und daher ohne großen Aufwand auf Seiten der Programmierung die Schätzung des Tempos und der Tempokategorie von Audiodateien ermöglicht.

Die Ergebnisse auf der Trainingsdatenbank lassen folgende Schlussfolgerungen zu:

Mehrere Akzente betrachten Die mit der Testumgebung berechneten Ergebnisse bestätigen die Hypothese, dass die gemeinsame Betrachtung unterschiedlicher Akzente

bessere Klassifikationsergebnisse liefert. Die Genauigkeit des Verfahrens ist mit mehreren Akzenten durchweg besser als mit dem tiefen Chroma allein, und auf kurzen Ausschnitten wird Unterschied mit über 10% Zuwachs an Genauigkeit am deutlichsten. Der Ansatz, mehrere Akzente gewichtet aufsummiert zu klassifizieren, weist also in die richtige Richtung.

Bereinigung des Spektrums unterlassen Auf allen durchgeführten Tests hat die Bereinigung des Spektrums vor der Chromaberechnung die Genauigkeit des Verfahrens beeinträchtigt. Daher wird empfohlen, das Spektrum bei der Chromaberechnung nicht zu bereinigen. Als Bonus wird dadurch die Rechenzeit und der Speicherbedarf des Verfahrens verringert.

Verdoppelungen und Halbierungen des Tempos Das Problem, die richtige metrische Ebene zu erkennen, ist noch immer nicht abschließend gelöst. Drei Erklärungen für dieses Phänomen wurden an Beispielen näher untersucht: Es gibt Musikstücke, bei denen das richtige Tempo schon für Menschen schwer zu schätzen ist. Dann spielt die Ausgewogenheit der Trainingsdaten bei dem vorgestellten Verfahren eine entscheidende Rolle. Und obwohl das Verfahren in den meisten Fällen gute Ergebnisse erzielt, eliminiert es auch nicht alle Fehler dieser Art. Auf der Trainingsdatenbank hat das Verfahren eher die Tendenz, das Tempo bei schnellen Stücken zu halbieren als bei langsamen Stücken zu verdoppeln.

Nach diesem Blick zurück folgt nun der Ausblick, in Form einer Sammlung von Ideen und Möglichkeiten, die das Verfahrens weiter verbessern können und die Tempoerkennung noch tiefgehender untersuchen.

Auswahl der Akzente In dieser Arbeit wurde das Verfahren aus [17] verallgemeinert, um auf beliebigen kontinuierlichen Akzentsignalen das Tempo zu bestimmen. Die hier aufgezeigten Chroma- und Energieakzente sind nur eine Möglichkeit, um musikalisch sinnvolle Werte aus dem Audiosignal zu extrahieren, die bei der Schätzung des Tempos eine Rolle spielen können. Hier wäre eine Vielzahl anderer Akzente denkbar – Eronen und Klapuri testen ja verschiedene Akzente einzeln gegeneinander, und es wäre sicherlich spannend, weitere oder andere Akzete miteinander zu kombinieren. Dazu soll diese Arbeit motivieren.

Berechnung der Periodenvektoren Eine weitere Möglichkeit wäre, zu testen, ob die generalisierte Autokorrelation zur Berechnung der Periodenvektoren wirklich das Mittel der Wahl darstellt. Da hier kontinuierliche Daten vorliegen, könnte man untersuchen, wie sich die Ergebnisse verändern, wenn beispielsweise Kammfilterbänke benutzt werden, um die Tempovektoren zu ermitteln.

Gewichtung der Periodenvektoren der Akzente In dieser Arbeit wurden die Periodenvektoren gewichtet aufsummiert, um einen einzigen Periodenvektor für die Klassifikation zu ermöglichen. Einerseits kann man mehr Intelligenz in die Berechnung des repräsentativen Periodenvektors für das Stück stecken. So könnten die einzelnen Ergebnisse der generalisierten Autokorrelation analysiert werden, und aufgrund der Analyse könnte entschieden werden, mit welchem Verfahren das Tempo des Stücks bestimmt werden soll. Wenn die Analyse bereits auf ein eindeutiges Tempo hindeutet, könnte das Tempo vielleicht direkt aus den Periodenvektoren bestimmt werden, ohne eine Klassifikation durchzuführen. Auch die Gewichtung der Akzente könnte sich an Eigenschaften der Periodenvektoren orientieren, statt für eine Trainingsmenge optimiert zu werden.

Andererseits kann untersucht werden, ob es sinnvoll ist, die Akzente erst nach der Klassifikation zusammenzuführen, wie bereits in Kapitel 5.2 angedeutet. Die Ergebnisse der Autokorrelationen stehen prinzipiell auch nach den Temposchätzungen der einzelnen Akzente zur Verfügung und können eine Entscheidung für ein Tempo unterstützen.

Alternative Klassifikationsverfahren Betritt man den Bereich der Klassifikationsverfahren, so zählen k -NN-Verfahren zu den einfachen und anschaulichen Verfahren zur überwachten Klassifikation von Daten. Die Statistik und Informatik stellt aber einen großen Fundus an Klassifikationsverfahren zur Verfügung. Es könnte untersucht werden, ob nicht andere Verfahren wie Random Forests oder die Stützvektormethode (SVM, für „*support vector machine*“) Alternativen zu der hier verwendeten Regression darstellen. Besonders interessant ist diese Überlegung für das Problem der Klassifikation in Tempokategorien, da hier nur zwischen wenigen Klassen entschieden werden muss.

Optimierung Die Ergebnisse des hier vorgestellten Verfahrens sind zwar für die Trainingsdatenbank in Bezug auf die Gewichtung der Periodenvektoren und der Parameter k und γ der gewichteten Regression optimiert, aber die Einstellungen der Chroma- und Energieakzente wurden nicht systematisch auf optimale Klassifikationsergebnisse angepasst. Hier wurden lediglich einige sinnvolle Annahmen getroffen, die sich an den Aussagen in den Arbeiten [17] und [32] orientieren, auf denen die Berechnung der Akzente beruht. Eine systematische Optimierung der Parameter der Akzentextraktion und der Periodenschätzung auf einer ausreichend großen und gut verteilten Datenbank könnte die Ergebnisse noch aussagekräftiger machen.

Offen geblieben ist auch die Frage, welche Parameter der Klassifikation für *unbekannte* Musikstücke die besten Ergebnisse erzielen. Die Optimierung auf der Trainingsdatenbank ermöglicht den Vergleich der Genauigkeit verschiedener Verfahren. Eine gute Klassifikation unbekannter Musikstücke erfordert in der Praxis jedoch andere Einstellungen der Klassifikationsparameter.

Anwendung auf andere Genres Die hier verwendete Trainingsmenge deckt viele Bereiche der populären Musik ab. Der Klassik- oder Jazzbereich wurde jedoch überhaupt nicht betreten. Es wäre daher interessant, herauszufinden, wie sich das vorgestellte Verfahren bei diesen Genres verhält und wie die Trainingsmengen aussehen könnten.

Fehlklassifikationen auf musikalische Eigenschaften untersuchen In Bezug auf die Verdoppelung oder Halbierung des Tempos könnte das Verständnis durch eine eingehende Untersuchung der musikalischen Eigenschaften der falsch erkannten Stücke vertieft werden. Eine Untersuchung zum Einfluss des Taktes oder der dominierenden Instrumente auf die Klassifikation könnte beispielsweise helfen, Fehlerquellen zu identifizieren.

Abschließend sei gesagt, dass die Arbeit nur eine Möglichkeit aufzeigt, aus unterschiedlichen musikalischen Akzenten das Tempo von Musik zu klassifizieren. Langsame Musikstücke profitieren von der Konzentration auf Harmoniewchsel, während gleichzeitig die Energiesprünge von schneller Musik mit Schlagzeug berücksichtigt werden. Das führt zu guten Ergebnissen, und wirft – was vielleicht noch wichtiger ist – eine Menge neuer, interessanter Fragen auf. Die Zukunft wird zeigen, ob diese Fragen das Potenzial haben, die Tempoerkennung weiter voranzutreiben.

Persönliches Nachwort

Ich liebe Musik, und daher habe ich die Gelegenheit sehr geschätzt, mich beruflich mit dem Thema Informatik und Musik auseinandersetzen zu können. Ich hoffe, dass die Begeisterung für das Thema etwas durchklingt und die Arbeit so zu weiteren Untersuchungen auf dem Gebiet ermuntert.

Bei dem ersten Teil, den Grundlagen der Tempoerkennung, habe ich mich stark an den Ausführungen von Manuel Davy, Stephen Hainsworth und Anssi Klapuri in dem Buch „Signal Processing Methods for Music Transcription“ orientiert, sowie an den Arbeiten von Anssi Klapuri und Antti Eronen. Sie und die anderen Autoren der Arbeiten, auf die hier verwiesen wird, sind wirkliche Experten in diesem Bereich, und das macht es oft schwierig, die Thematik einfacher zu erläutern und präziser darzustellen als sie es getan haben.

Da ich beruflich seit einigen Jahren mit dem .NET-Framework arbeite und es für so manche Aufgabe zu schätzen gelernt habe, wollte ich - spätestens nach der Diplomarbeit - eine .NET-Version des resultierenden Verfahrens schreiben. Während ich mich dann im Frühjahr in das Thema eingelesen habe, bin ich zufällig über ein Video gestolpert, dass die Möglichkeiten und Konzepte der funktionalen Programmiersprache F# vorstellt, und ich hatte das Gefühl, dass diese Sprache genau das Richtige für technische Berechnungen auf großen Datenmengen, wie bei der Signalverarbeitung, ist. Und so hoffe ich, abseits der ausgetretenen Pfade der universitären Standardsprache Java und der oft verwendeten Implementierungen in Matlab und C(++), einen etwas anderen, funktionalen, aber dennoch interessanten Blick auf die Signalverarbeitung geworfen zu haben.

Schließlich war es für mich faszinierend, die Zusammenhänge von Musik und Tempo zu ergründen, und je tiefer man in die Verfahren einsteigt, desto komplexer wird der Problembereich. Umso erstaunter und dankbarer bin ich deshalb dafür, dass dem Menschen die Fähigkeit der Tempoerkennung gegeben ist, ohne dabei je einen Gedanken an Fouriertransformationen, zeithomogene Markovmodelle oder gewichtete Regressionen zu verschwenden. Wer auch immer dem Menschen das Verfahren zur Tempoerkennung mitgegeben hat: Ich würde gerne den Quellcode dazu lesen – denn das ist der Stein der Weisen, an dem sich sämtliche Verfahren zur Tempoerkennung messen müssen.

„Ich danke dir dafür, dass ich wunderbar gemacht bin“ [57, Ps. 139:14]

Anhang A

Musikdaten

A.1 Trainingsmenge

Die Trainingsmenge besteht aus 98 Musikstücken. Die hier aufgeführten Klassifikationsergebnisse beruhen auf den manuell erstellten Segmenten, die eingehend in Kapitel 7 besprochen werden. Die Spalten K_{ann} und K_{est} stehen für die tatsächliche und die geschätzte Tempokategorie der Stücke. Die Tempokategorien sind langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm). T_{ann} und T_{est} bezeichnen das tatsächliche und das geschätzte Tempo der Stücke. Die Spalten „Abw.“ und „Log“ sind die Abweichung bzw. der logarithmische Fehler, die in Kapitel 7 definiert wurden.

Name	K_{ann}	K_{est}	T_{ann}	T_{est}	Abw.	Log
Celine Dion - My Heart Will Go On	1	1	48	49,9	4,0%	0,057
Claudia Jung - Je t'aime mon amour	1	1	48	66,0	37,5%	0,459
Kylie - Bittersweet Goodbye	1	1	48	48,9	1,8%	0,026
Prince - Sometimes It Snows In April	1	1	50	52,5	5,0%	0,071
Abba - I Have A Dream	1	1	52	53,4	2,7%	0,038
Johnny Cash - Loves Been Good to Me	1	1	52	51,5	1,0%	-0,014
Johnny Cash - Rose Of My Heart	1	1	52	51,4	1,2%	-0,018
Tom Astor - Sie war jung und frei	1	1	54	54,1	0,1%	0,002
Silbermond - Das Beste	1	1	56	57,6	2,9%	0,041
The Rolling Stones - Play With Fire	1	1	56	55,1	1,6%	-0,023
The Rolling Stones - As Tears Go By	1	1	58	57,7	0,6%	-0,008
Timbaland feat. One Republic - Apologize	1	1	58	60,7	4,7%	0,066
Tom Astor - Redneck Riviera	1	1	58	60,0	3,4%	0,049
A Fine Frenzy - Almost Lover	1	1	60	61,5	2,5%	0,035
Die Ladiner - Wahre Liebe ein Leben lang	1	1	60	60,6	1,0%	0,014

Name	<i>K_{ann}</i>	<i>K_{est}</i>	<i>T_{ann}</i>	<i>T_{est}</i>	Abw.	Log
Diverse - Tennessee Rose	1	1	60	58,0	3,3%	-0,049
Amigos - Dann kam ein Engel	1	1	63	63,4	0,6%	0,008
Claudia Jung - Amore amore	1	s	63	126,0	100,0%	1,000
Johnny Cash - Help Me	1	1	63	57,0	9,5%	-0,144
Truck Stop - Heimat ist überall	1	1	63	62,8	0,3%	-0,005
All Saints - Never Ever	1	1	66	66,2	0,4%	0,005
Claudia Jung - Mittsommernacht	1	1	66	64,9	1,7%	-0,025
K. D. Lang - Constant Graving	1	1	66	66,0	0,0%	0,000
Maria Mena-All This Time Pick-Me - Up Song	1	1	66	68,4	3,6%	0,052
Monrose - What you dont know	1	1	66	64,2	2,8%	-0,041
Pink - Dear Mr. President	1	1	66	65,3	1,0%	-0,014
Percy Sledge - When A Man Loves A Woman	1	1	68	66,9	1,6%	-0,023
Amigos - Mein Elternhaus	1	1	69	70,0	1,4%	0,020
Fady Maalouf - Amazed	1	1	69	70,2	1,7%	0,025
Xavier Naidoo - Abschied nehmen	1	1	69	71,3	3,3%	0,047
Aloha from Hell - Walk away	1	1	72	72,7	1,0%	0,014
Herbert Grönemeyer - Glück	1	1	72	71,3	1,0%	-0,014
Spice Girls - 2 Become 1	1	1	72	73,3	1,8%	0,025
Incubus - Love Hurts	1	1	76	77,3	1,7%	0,024
Justin Timberlake - What Goes Around... Comes Around	1	1	76	74,9	1,5%	-0,021
Kevin Michael - Ain't Got You	1	1	76	75,6	0,5%	-0,008
Silbermond - Symphonie	1	1	76	76,8	1,1%	0,015
Söhne Mannheims - Das hat die Welt noch nicht gesehen	1	1	76	74,9	1,5%	-0,021
Udo Jürgens - Griechischer Wein	1	1	76	75,7	0,4%	-0,006
Truck Stop - Sag Mami Goodbye	1	1	78	77,0	1,2%	-0,018
Gunter Gabriel - Wer einmal tief im Keller saß	1	1	80	83,7	4,7%	0,066
Johnny Cash - Give me love to rose	1	1	80	81,0	1,2%	0,017
Patrick Lindner - Jedes Herz braucht eine Heimat	1	1	80	77,0	3,7%	-0,054
Queen - We Will Rock You	1	1	80	80,4	0,5%	0,006
Rihanna - Rehab	1	1	80	79,8	0,2%	-0,004
Johnny Cash - God's Gonna Cut You Down	1	1	82	81,6	0,5%	-0,007
Spice Girls - Viva Forever	1	1	84	84,0	0,0%	0,000
Clueso - Chicago	1	s	84	165,4	97,0%	0,978
Johnny Cash - A Legend In My Time	1	1	84	59,3	29,4%	-0,503
Katie Melua - If you were a Sailboat	1	1	84	84,0	0,0%	0,000
Madonna - The Power of Goodbye	1	1	84	87,4	4,0%	0,057

Name	<i>K_{ann}</i>	<i>K_{est}</i>	<i>T_{ann}</i>	<i>T_{est}</i>	Abw.	Log
Natasha Bedingfield - Soulmate	1	1	84	84,0	0,0%	0,000
Ne-Yo - Go On Girl	1	1	84	84,0	0,0%	0,000
Trucker Bill & Country Band - Please Release Me	1	1	84	84,6	0,8%	0,011
Uriah Heep - Lady in Black	1	1	84	85,4	1,6%	0,023
Beyonce - If I were a boy	1	1	88	88,0	0,0%	0,000
Mühlenhof Musikanten - Jeder Mensch geht seinen Weg	1	1	88	89,3	1,5%	0,021
Claudia & alexx - Ein par Takte Samba	m	m	90	93,2	3,6%	0,051
Catain Cooook und seine singenden Saxophone - Ich denk so gern an meine Mutter	m	m	92	91,6	0,5%	-0,007
Juli - Regen und Meer	m	m	92	91,6	0,5%	-0,007
Maria Mena - Just Hold Me	m	m	92	91,2	0,9%	-0,013
R.E.M. - Leaving New York	m	m	92	90,6	1,5%	-0,021
Wir sind Helden - Denkmal	m	m	92	93,8	2,0%	0,029
Lynyrd Skynyrd - Sweet Home Alabama	m	m	96	96,1	0,1%	0,002
Stefan Mross - Echte Freunde	m	m	96	94,1	2,0%	-0,029
Gunter Gabriel - Ein Stückchen pro Tag	m	m	108	106,6	1,3%	-0,019
Spice Girls - Wannabe	m	m	108	112,0	3,7%	0,052
Johnny Cash - Get Rhythm	m	m	112	108,0	3,6%	-0,052
Johnny Cash - Like The 309	m	m	112	104,4	6,8%	-0,101
Jürgen Drews - Ein Bett im Kornfeld	m	m	112	110,9	1,0%	-0,014
Abba - Gimme Gimme Gimme	m	m	116	116,0	0,0%	0,000
Europe - The Final Countdown	m	l	116	58,0	50,0%	-1,000
Kylie - Spinning Around	m	m	116	116,0	0,0%	0,000
Pussycat Dolls - When I Grow Up	m	m	116	114,8	1,0%	-0,014
Robie Williams - Old Before I Die	m	l	116	58,7	49,4%	-0,982
Die Ärzte - Nichts in der Welt	s	s	120	126,0	5,0%	0,070
Johnny Cash - Further On Up the Road	s	l	120	61,7	48,6%	-0,959
Mark Medlock - Summer Love	s	s	120	120,6	0,5%	0,008
Udo Jürgens - Mit 66 Jahren	s	s	120	126,0	5,0%	0,070
Robie Williams - Let me Entertain You	s	s	123	124,7	1,4%	0,020
Abba - Voulez-Vous	s	s	126	126,0	0,0%	0,000
Claudia Jung - Stumme Signale	s	s	126	120,0	4,8%	-0,070
Katy Perry - I Kissed A Girl	s	l	126	64,1	49,1%	-0,975
Kiss - I was made for loving you	s	s	126	124,7	1,0%	-0,014
Manowar - Fighting The World	s	s	126	127,6	1,3%	0,018
Marianne Rosenberg - Er gehört zu mir	s	s	126	129,0	2,4%	0,034
Mia - Tanz der Moleküle	s	s	126	126,0	0,0%	0,000

Name	K_{ann}	K_{est}	T_{ann}	T_{est}	Abw.	Log
Gunter Gabriel - Freund aller Fahrer	s	s	129	126,0	2,3%	-0,034
Culcha Candela - Ey DJ	s	l	130	66,0	49,2%	-0,978
Fettes Brot - Erdbeben	s	s	130	124,8	4,0%	-0,059
Boney - Ich hab noch viel vor in meinem Leben	s	s	132	133,3	1,0%	0,014
Daughtry - Feels Like Tonight	s	s	132	124,7	5,5%	-0,082
Dj Ötzi - Cheerio	s	s	132	132,0	0,0%	0,000
Peter Fox - Alles neu	s	s	132	132,0	0,0%	0,000
Spencer Davis Group - Somebody help me	s	s	132	130,7	1,0%	-0,014
Cascada - Because The Night	s	s	138	139,9	1,3%	0,019
Comedian Harmonists - Veronika der Lenz ist da	s	l	141	70,6	49,9%	-0,998
Chuck Berry - Roll Over Beethoven	s	l	176	88,8	49,5%	-0,987

A.2 Unabhängige Testmenge

Die unabhängige Testmenge besteht aus 15 Musikstücken, die nicht in der Trainingsmenge vorhanden sind. Sie kommen aus ähnlichen Genres wie die Trainingsmenge, enthalten also keine Klassik oder Jazz. Die Ergebnisse in dieser Tabelle beziehen sich auf die große Trainingsmenge mit einer Gewichtung von [0,35 0,20 0,30 0,15], $k = 11$ und $\gamma = 0,6$. K_{ann} , K_{est} , T_{ann} und T_{est} sind wie oben die tatsächliche und geschätzte Kategorie bzw. das Tempo, und „Abw.“ ist die Abweichung.

Name	K_{ann}	K_{est}	T_{ann}	T_{est}	Abw.
ABBA - Waterloo	s	s	144	146,0	1,4%
Black Eyed Peas - I Got A Feeling	s	s	126	128,4	1,9%
Bon Jovi - Always	l	l	72	69,7	3,2%
Bon Jovi - Lie To Me.	l	l	80	80,2	0,2%
Boss Hoss - Close	m	m	96	94,8	1,3%
BossHoss1	s	s	132	129,6	1,8%
genesis - Hold On My Heart	l	l	84	85,5	1,8%
Genesis - In Too Deep	l	m	52	103,8	99,6%
Ich und Ich - Wenn ich tot bin	l	l	56	55,7	0,6%
Johnny Cash - Cool Water	l	l	58	57,0	1,7%
Matthias Reim Vermiss Dich	l	l	69	68,3	1,0%
Matthias Reim-Bastian	l	l	88	87,6	0,5%
Rihanna - Take A Bow	l	l	84	81,6	2,9%
Rolling Stones - Jumping Jack Flash	s	s	138	142,6	3,3%
Wolfgang Petry -Augen zu und durch	s	s	132	129,9	1,6%

Abbildungsverzeichnis

1.1	Schläge und ihre Abstände	4
1.2	Die verschiedenen Ebenen der metrischen Hierarchie am Beispiel unterschiedlicher Taktarten. Hervorgehoben wird jeweils die Ebene der Grundschläge, also der <i>Tactus</i> oder Beat.	5
1.3	Grundschläge, mit einer imaginärer Sinusschwingung abgeglichen. Der Abstand $\tau = \frac{1}{f}$ ist die Periodendauer des Tempos f	6
2.1	Diskretisierung des analogen Audiosignals	12
2.2	Audiosignal im Zeitbereich	13
2.3	Spektrum des gesamten Audiosignals, eingeschränkt auf den Bereich unter 1 kHz	14
2.4	Die Einteilung des Audiosignals in Fenster (Frames) von zwei Sekunden Länge. Auf eine Überlappung der Fenster wurde für die Darstellung verzichtet.	15
2.5	Spektrogramm mit einer Fenstergröße von 4096 Samples (93 ms)	15
2.6	Das Spektrogramm stellt immer einen Kompromiss zwischen guter Auflösung der Zeit- oder der Frequenzachse dar.	16
2.7	Die Hüllkurve des Audiosignals	18
3.1	Frequenzbänder eines Audiosignals nach der Filterbank	22
3.2	Aufbau der Akzentextraktion aus den Audiodaten nach [32]	23
3.3	Die Leistungen der Bänder nach logarithmischer Normalisierung. Dargestellt werden zwei der 36 Frequenzbänder.	24
3.4	Die Bänder nach der Berechnung der gewichteten Summe $u_b(n)$ der Leistung und der differenzierten Leistung.	25
3.5	Die Summe der Bänder zeigt deutlich die Akzentuierung über die Zeit. . . .	25
3.6	Die Shepard-Helix beschreibt die Wahrnehmung der Tonhöhe, entnommen aus [3, S. 97]. Die Tonhöhe steigt vertikal über den Chromaklassen an. . . .	26
3.7	Das Chromagramm stellt dar, wie stark die Töne zur jeweiligen Zeit im Frequenzspektrum auftreten. Die Framelänge beträgt hier 8192 Samples (186 ms).	27

3.8	Antworten der Filterbank $H_b(k)$	29
3.9	Aufbau der Extraktion des Chromaakkzents aus den Audiodaten nach [17] . .	30
3.10	Grundfrequenzbasierte Chromaanalyse im Zeitverlauf	31
3.11	Die interpolierten Signale $z_b(n)$; dargestellt sind $b = 25$ (oben) und $b = 20$ (unten).	31
3.12	Gewichtet differenzierte Signale $u_b(n)$	32
3.13	Der Mittelwert aller Chromaklassen bildet das Akzentsignal $a(n)$	33
4.1	Die Autokorrelation bestimmt die Periodizität eines Signals. Die Verschiebung τ wird hier auf dem Bereich von 0,2 s bis 2 s dargestellt.	38
4.2	Übersicht des Beattracking-Systems (BTS) nach Goto, entnommen aus [20, S. 367]	42
4.3	Probabilistisches Hidden-Markov-Modell zur Tempoerkennung, entnommen aus [32, S. 347]	44
4.4	Das ansteigende Tempo der Stücke zeigt sich an den Maxima der Periodenfunktionen.	47
5.1	Überblick über das Verfahren von Eronen und Klapuri [17]	53
5.2	Das Akzentsignal und der Periodenvektor für das Verfahren von Eronen und Klapuri [17]	54
5.3	Klassifikation: Die $k = 6$ nächsten Nachbarn, in ansteigendem Tempo. Dargestellt ist jeweils der Periodenvektor des Trainingsbeispiels (blau) und der Periodenvektor des zu klassifizierenden Stücks nach dem Resampling (rot gestrichelt). Der Median bestimmt das geschätzte Tempo und ist hervorgehoben.	54
5.4	Übersicht des verallgemeinerten Verfahrens. Mehrere Akzente werden verarbeitet und vor der Klassifikation zusammengeführt.	56
5.5	Aufbau der Chromaextraktion aus den Audiodaten	57
5.6	Aufbau der Energieakzentextraktion aus den Audiodaten	57
5.7	Die vier für Akzente, die für die Klassifikation berechnet werden	58
5.8	Die Periodenvektoren für die vier für Akzente	59
5.9	Die gewichtete Summe der vier Periodenvektoren der Akzente. Die Gewichtung betragen [0,35 0,20 0,30 0,15].	59
5.10	Klassifikation der gewichteten Periodenvektoren: Die $k = 6$ nächsten Nachbarn. Das Gewicht eines Nachbarn ist oben links angegeben. In der Mitte ist das Verhältnis des Resamplings, und rechts das resultierende Tempo.	59
6.1	TempEx mit geladener Trainingsdatenbank	67

6.2	TempEx berechnet die Periodenvektoren der einzelnen Akzente auf der Trainingsdatenbank. Die Einstellungen für die Berechnungen finden sich im linken unteren Bereich, abhängig vom darüber gewählten Testlauf.	67
6.3	Die Klassifikationsergebnisse für einen gewählten Testlauf. Die Einstellungen für k und γ werden nach der Klassifikation auf die optimalen Werte voreingestellt.	68
6.4	Die Informationsansicht zu einem Stück in TempEx. Die berechneten Periodenvektoren und gewichtete Summe werden dargestellt. Darunter sieht man die Nachbarn der Klassifikation mit einem Vergleich der Trainingsbeispiele (blau) mit dem Periodenvektor des zu schätzenden Stücks (rot gestrichelt) nach dem Resampling. Der Median bestimmt schließlich das Tempo und wird daher gekennzeichnet.	68
6.5	TempEst während der Berechnung der Tempi eines Albums	75
7.1	Die Tempoverteilung der Trainingsdaten und die berechneten Periodenvektoren. Die Stücke sind mit aufsteigendem Tempo angeordnet.	78
7.2	Der logarithmische Fehler verdeutlicht Verdoppelungen und Halbierungen des Tempos.	83
7.3	Die Klassifikation von „Roll Over Beethoven“, entnommen aus TempEx. Die Nachbarn sind nach aufsteigendem Tempo angeordnet, der Median wird umrahmt. Der Periodenvektor des Nachbarn ist blau dargestellt, der zu klassifizierende Vektor ist rot gestrichelt.	84
7.4	Die Klassifikation von „Amore Amore“ in TempEx. Links oben sieht man das Gewicht bei der Berechnung des Median (0,7192 für „Stumme Signale“).	85
7.5	Die Periodenvektoren von „Amore Amore“ (rot gestrichelt) und „Stumme Signale“ (blau) im Vergleich	85
7.6	Die Klassifikation von „I Kissed a Girl“ in TempEx. Dieses schnelle Stück findet fast nur langsame Nachbarn.	86
7.7	Die Periodenvektoren für die vier Akzente von „I Kissed A Girl“	86
7.8	Die Tempoverteilung der Trainingsdaten und die Periodenvektoren der vergrößerten Trainingsdatenbank	89

Tabellenverzeichnis

4.1 Übersicht über einige Ansätze zur Tempo- und Metrikerkennung, basierend auf [25, S. 104] und [16, S. 47]. Bei der Eingabe steht „A“ für Audiosignale und „S“ für symbolische Daten.	37
5.1 Kategoriefehler bei der Klassifizierung in die Tempokategorien langsam (0 bis 90 Bpm), mittel (90 bis 130 Bpm) und schnell (ab 130 Bpm) für die k -NN-Regression, entnommen aus [17, S. 55]. Die Zeilen stehen für die tatsächliche Kategorie, die Spalten für die Schätzung.	55
5.2 Kategoriefehler bei der Klassifizierung in Tempokategorien für die Methode von Klapuri <i>et. al.</i> [32], entnommen aus [17, S. 55]. Die Zeilen stehen für die tatsächliche Kategorie, die Spalten für die Schätzung.	55
6.1 Speicherbedarf von MIRtoolbox. Die geladene Audiodatei hat auf der Festplatte eine Größe von etwa 23,67 MB und ist 4:41 Minuten lang, also eine typische Audiodatei der Trainingsmenge.	63
6.2 Rechenzeiten der Verfahrensschritte auf einer 30-minütigen Audiodatei . . .	69
7.1 Genauigkeit der Klassifikation von 98 manuell erzeugten Segmenten	79
7.2 Konfusionsmatrix: Betrachtung aller Akzente für die Kategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 130 Bpm) und schnell (ab 130 Bpm).	80
7.3 Konfusionsmatrix: Tiefer Chromaakzent für die Tempokategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 130 Bpm) und schnell (ab 130 Bpm).	80
7.4 Konfusionsmatrix: Klassifizierung auf allen Akzenten in die Kategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm). . .	82
7.5 Konfusionsmatrix: Tiefer Chromaakzent mit den Tempokategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm). . .	82
7.6 Konfusionsmatrix: Alle vier Akzente unter Bereinigung des Spektrums. Tempokategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).	82
7.7 Genauigkeit der Klassifikation von 98 vollständigen Audiodateien	87

7.8 Konfusionsmatrix: Alle Akzente bei 98 vollständigen Stücken mit den Kategorien: Langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).	87
7.9 Genauigkeit der Klassifikation auf 98 Ausschnitten mit 30 Sekunden Länge	87
7.10 Konfusionsmatrix: Alle Akzente bei 98 Ausschnitten. Die Tempokategorien sind langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).	88
7.11 Genauigkeit der Klassifikation von Ausschnitten aus 297 Stücken	88
7.12 Konfusionsmatrix: Alle Akzente bei 297 Ausschnitten. Die Tempokategorien sind langsam (0 bis 90 Bpm), mittel (90 bis 120 Bpm) und schnell (ab 120 Bpm).	89
7.13 Genauigkeit der Klassifikation unbekannter Musikstücke auf unterschiedlichen Trainingsmengen	90

Literaturverzeichnis

- [1] ALLEN, P. E. und R. B. DANNENBERG: *Tracking musical beats in real time*. In: *International Computer Music Conference (ICMC)*, Seiten 140–143, Glasgow, Schottland, 1990.
- [2] ALONSO, M., G. RICHARD und B. DAVID: *Accurate tempo estimation based on harmonic+noise decomposition*. EURASIP Journal on Advances in Signal Processing, 2007.
- [3] BARTSCH, A. und G. H. WAKEFIELD: *Audio thumbnailing of popular music using chroma-based representations*. IEEE Transactions on Multimedia, 7(1):96–104, 2005.
- [4] BENARY, P.: *Rhythmik und Metrik - Eine praktische Anleitung*. Musikverlag Hans Gerig, Köln, 1967.
- [5] BROWN, J. C.: *Determination of the meter of musical scores by autocorrelation*. Journal of the Acoustical Society of America, 94(4):1953–1957, 1993.
- [6] BUTZ, T.: *Fouriertransformation für Fußgänger*. Teubner, Wiesbaden, 2007.
5., durchgesehene Auflage.
- [7] CEMGIL, A. T., P. DESAIN und B. KAPPEN: *Rhythm quantization for transcription*. Computer Music Journal, 24(2):60–76, 2000.
- [8] CEMGIL, A. T. und B. KAPPEN: *Tempo tracking and rhythm quantization by sequential Monte Carlo*. In: *Neural Information Processing Systems*, Vancouver, Kanada, 2001.
- [9] CEMGIL, A. T. und B. KAPPEN: *Monte carlo methods for tempo tracking and rhythm quantization*. Journal of Artificial Intelligence Research, 18:45–81, 2003.
- [10] DAVIES, M. E. und M. D. PLUMBLEY: *Context-dependent beat tracking of musical audio*. IEEE Transactions on Audio, Speech, and Language Processing, Seiten 1009–1020, März 2007.

- [11] DAVY, M.: *An introduction to statistical signal processing and spectrum estimation.* In: Klapuri, A. und M. DAVY (Herausgeber): *Signal Processing Methods for Music Transcription*, Kapitel 1, Seiten 21–64. Springer Science+Business Media LLC, New York, 2006.
- [12] DIXON, S.: *Evaluation of the audio beat tracking system beatroot.* Journal of New Music Research, 30(1):39–50, 2007.
- [13] ECK, D.: *A positive-evidence model for classifying rythmical patterns.* Technischer Bericht IDSIA-09-00, IDSIA, Lugano, Switzerland, 2000.
- [14] ECK, D.: *A network of relaxation oscillators that finds downbeats in rhythms.* Technischer Bericht IDSIA-06-01, Dalle Molle Institute for Artificial Intelligence, 2001.
- [15] ELLIS, D. P.: *Beat tracking by dynamic programming.* Journal of New Music Research, 36(1):51–60, 2007.
- [16] ERONEN, A.: *Signal Processing Methods for Audio Classification and Music Content Analysis.* Doktorarbeit, Tampere University of Technology, Finnland, 2009.
- [17] ERONEN, A. und A. Klapuri: *Music tempo estimation using k-NN regression.* IEEE Transactions on Audio, Speech, and Language Processing, 18(1):50–57, 2010.
- [18] GOTO, M.: *Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions.* Speech Communication, 27(3-4):311–335, 1999.
- [19] GOTO, M.: *An audio-based real-time beat tracking system for music with or without drum sounds.* Journal of New Music Research, 30(2):159–171, 2001.
- [20] GOTO, M. und Y. MURAOKA: *A beat tracking system four acoustic signals in music.* In: *ACM International Conference on Multimedia*, Seiten 365–372, San Francisco CA, U.S.A., 1994.
- [21] GOTO, M. und Y. MURAOKA: *A real-time beat tracking system for audio signals.* In: *International Computer Music Conference*, Seiten 171–174, Tokyo, Japan, 1995.
- [22] GOUYON, F., P. HERRERA und P. CANO: *Pulse-dependent analysis of percussive music.* In: *22nd Internatinal Audio Engineering Society Conference*, Espoo, Finnland, 2002.
- [23] GOUYON, F. ET AL.: *An experimental comparison of audio temo induction algorithms.* IEEE Transactions on Audio, Speech, and Language Processing, 14(5):1832–1844, 2006.
- [24] HAINSWORTH, S. W.: *Techniques for the Automated Analysis of Musical Audio.* Doktorarbeit, Department of Engineering, University of Cambridge, 2004.

- [25] HAINSWORTH, S. W.: *Beat tracking and musical metre analysis*. In: Klapuri, A. und M. DAVY (Herausgeber): *Signal Processing Methods for Music Transcription*, Kapitel 4, Seiten 101–130. Springer Science+Business Media LLC, New York, 2006.
- [26] HAINSWORTH, S. W. und M. D. MACLEOD: *Particle filtering applied to musical tempo tracking*. Journal of Applied Signal Processing, 15:2385–2395, 2004.
- [27] HARROP, J.: *F# for Scientists*. John Wiley & Sons, Hoboken NJ, U.S.A., 2008.
- [28] JENSEN, K. und T.H. ANDERSEN: *Beat estimation on the beat*. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz NY, U.S.A., 2003.
- [29] Klapuri, A.: *Auditory model-based methods for multiple fundamental frequency estimation*. In: Klapuri, A. und M. DAVY (Herausgeber): *Signal Processing Methods for Music Transcription*, Kapitel 8, Seiten 229–267. Springer Science+Business Media LLC, New York, 2006.
- [30] Klapuri, A.: *Introduction*. In: Klapuri, A. und M. DAVY (Herausgeber): *Signal Processing Methods for Music Transcription*, Kapitel 1, Seiten 3–20. Springer Science+Business Media LLC, New York, 2006.
- [31] Klapuri, A.: *Multiple fundamental frequency estimation by summing harmonic amplitudes*. In: *7th International Conference on Music Information Retrieval (ISMIR-2006)*, Victoria, Kanada, 2006.
- [32] Klapuri, A., A. ERONEN und J. ASTOLA: *Analysis of the meter of acoustic music signals*. IEEE Transactions on Audio, Speech, and Language Processing, 14(1):342–355, 2006.
- [33] LARGE, E. W.: *Beat tracking with a nonlinear oscillator*. In: *International Joint Conference on Artificial Intelligence*, Seiten 24–31, Stockholm, Schweden, 1994.
- [34] LAROCHE, J.: *Estimating tempo, swing and beat locations in audio recordings*. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Seiten 135–138, 2001.
- [35] LAROCHE, J.: *Efficient tempo and beat tracking in audio recordings*. Journal of the Audio Engineering Society, 51(4):226–233, 2003.
- [36] LARTILLOT, O. und P. TOIVIAINEN: *A matlab toolbox for musical feature extraction from audio*. In: *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07)*, Seiten 1–8, Bordeaux, Frankreich, September 2007.

- [37] LERDAHL, F. und R. JACKENDOFF: *A Generative Theory of Tonal Music*. MIT Press, Cambridge MA, U.S.A., 1983.
- [38] McAULEY, J. D.: *Perception of Time as Phase: Towards an Adaptive-Oscillator Model of Rhythmic Pattern Processing*. Doktorarbeit, Computer Science and Cognitive Science, Univ. Indiana, 1995.
- [39] NEWARD, T.: *Einführung in F#*. MSDN Magazin, 2008.
- [40] PARNCUTT, R.: *A perceptual model of pulse salience and metrical accent in musical rhythms*. Music Perception, 11(4):409–464, 1994.
- [41] PEETERS, G.: *Template-based estimation of time-varying tempo*. EURASIP Journal on Advances in Signal Processing, 2007(1):158–171, 2007.
- [42] PUOPOLI, J. und S. SQUIRES: *F# Survival Guide*. <http://www.ctocorner.com/fsharp/book>, 15.09.2010.
- [43] RAPHAEL, C.: *Automated rhythm transcription*. In: *2nd Annual International Symposium on Music Information Retrieval*, Bloomington IN, U.S.A., 2001.
- [44] ROSENTHAL, D. F.: *Machine Rhythm: Computer Emulation of Human Rhythm Perception*. Doktorarbeit, Massachussettes Institute of Technology, 1992.
- [45] SCHEIRER, E. D.: *Tempo and beat analysis of acoustical musical signals*. Journal of the Acoustical Society of America, 103(1):588–601, 1998.
- [46] SEPPÄNEN, J.: *Tatum grid analysis of musical signals*. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Seiten 131–134, New Paltz NY, U.S.A., 2001.
- [47] SEPPÄNEN, J.: *Computational Models of Musical Meter Estimation*. Diplomarbeit, Tampere University of Technology, Tampere, Finnland, August 2001.
- [48] SEPPÄNEN, J., A. ERONEN und J. HIIPAKKA: *Joint beat and tatum tracking from music signals*. In: *7th International Conference on Music Information Retrieval ISMIR 2006*, Victoria, Kanada, Oktober 2006.
- [49] SEYERLEHNER, K., G. WIDMER und D. SCHNITZER: *From rhythm patterns to perceived tempo*. In: *8th International Conference on Music Information Retrieval (ISMIR-07)*, Wien, Österreich, 2007.
- [50] SHEPARD, R. N.: *Circularity in judgements of relative pitch*. Journal of the Acoustical Society of America, 36(12):2346–2353, 1964.

- [51] SHIU, Y. und C.-C. J. KUO: *Musical beat tracking via kalman filtering and noisy measurements selection*. In: *IEEE International Symp. Circ. and Syst.*, Seiten 3250–3253, Mai 2008.
- [52] SMITH, C.: *Programming F#*. O'Reilly & Associates, Beijing, 2009.
- [53] SYME, D., A. GRANICZ und A. CISTERNINO: *Expert F# 2.0*. Apress, New York, U.S.A., 2010.
- [54] TOIVIAINEN, P.: *An interactive MIDI accompanist*. Computer Music Journal, 22(4):63–75, 1998.
- [55] UHLE, C. und J. HERRE: *Estimation of tempo, micro time and time signature from percussive music*. In: *International Conference on Digital Audio Effects*, London, U.K., 2003.
- [56] WANG, Y. und M. VILERMO: *A compressed domain beat detector using mp3 audio bitstreams*. In: *ACM International Multimedia Conference*, Seiten 194—202, Ottawa, Kanada, 2001.
- [57] *Die Bibel, nach der Übersetzung Martin Luthers*. Deutsche Bibelgesellschaft, Stuttgart, 1984.
- [58] *Gute Nachricht Bibel*. Deutsche Bibelgesellschaft, Stuttgart, 1997.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 17. Oktober 2010

Thorsten Deinert

