Algorithmen zur Synchronisation von Musikdaten im Partitur-, MIDI- und PCM-Format

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Vlora Arifi

aus

Prishtina

Bonn, den 1. August 2002

Algorithmen zur Synchronisation von Musikdaten im Partitur-, MIDI- und PCM-Format

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Vlora Arifi

aus

Prishtina

Bonn, den 1. August 2002

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

Mamit, për të gjitha që m'i dhuroi në jetë.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Digitalisierung und weltweite Vernetzung haben zu einer Flut von Daten höchst unterschiedlichen Inhalts und Formats geführt, die es durch gezielte multimediale Techniken sinnvoll zu nutzen gilt. Insbesondere sind Verfahren zur Indexierung, zum Browsing und Retrieval in diesen multimedialen Datenbeständen, durch die dem Benutzer die Daten erst zugänglich gemacht werden, von großer Bedeutung. Während für rein textuelle Dokumente in diesem Zusammenhang schon eine Reihe von Methoden entwickelt wurden, steht man bei multimedialen Dokumenten hier noch vor vielen ungelösten Problemen, was insbesondere an der Inhomogenität und Komplexität der Daten liegt.

Allein schon der Fall des Datentyps *Audio*, um den es in dieser Arbeit geht, ist äußerst komplex. Zur Beschreibung von Musik gibt es viele verschiedene Möglichkeiten, von denen die meisten im Hinblick auf bestimmte Verwendungszwecke entwickelt wurden. So wird z. B. ein Musikstück vollständig durch die Darstellung der *akustischen Wellenform* – oft kodiert im *PCM-Format* – beschrieben und kann über geeignete Ausgabegeräte, z. B. mit einem CD-Gerät, wiedergegeben werden. Die Wellenform hat allerdings den Nachteil, dass *inhaltsbezogene Informationen* – wie z. B. die dem Musikstück zugrundeliegenden Noten – praktisch nicht ablesbar sind. Darüber hinaus ist der hohe Speicherbedarf der PCM-Daten für viele Anwendungen nicht akzeptabel. Im Gegensatz hierzu wird ein Musikstück durch die *Partiturdarstellung* rein durch inhaltsbezogene Informationen wie Tonhöhen, Tondauern, Einsatzzeiten sowie Tempo- und Dynamikangaben beschrieben. Damit eignet sich diese Darstellungsform sowohl als Spielanleitung für Musiker als auch für die algorithmische Verarbeitung durch Computer. Hinzu kommt, dass diese parametrische Darstellungsform der Musik im Vergleich zum PCM-Format äußerst speichereffizient ist. Ziel einer digitalen Musikbibliothek (DMB) ist es nun, solch unterschiedliche Daten mit Hilfe multimedialer Techniken neuen Nutzungs- und Speicherformen zugänglich zu machen, wie z. B. weltweiter Zugriff, neuartige Retrieval- und Browsing-Techniken oder multimediale Weiterverarbeitung gefundener Dokumente etwa zu Lehr-, Forschungs- oder Präsentationszwecken.

Diese Aufgabenstellungen dienen als Ausgangspunkt für die vorliegende Arbeit, in der neue, für diese Ziele grundlegende Algorithmen entworfen, implementiert und getestet werden. Zum einen geht es um Algorithmen zur *Extraktion* von musikalisch relevanten Parametern, wie z. B. Einsatzzeiten oder Tonhöhen aus einer Wellenformdarstellung eines Musikstücks. Diese Informationen sind im Hinblick auf inhaltsbasiertes Retrieval und Browsing in PCM-Daten

von großer Bedeutung. Zum anderen werden Algorithmen zur *Synchronisation* verschiedener Darstellungen ein und desselben Musikstücks entworfen. Hierbei verstehen wir unter Synchronisation ein Verfahren, das zu einer bestimmten Position innerhalb einer Darstellung eines Musikstücks (z. B. in einem PCM-Datensatz) die entsprechende Stelle innerhalb einer anderen Darstellung (z. B. MIDI-Datensatz) bestimmen kann. Die Verlinkung verschiedener Datenformate ist ebenfalls für die inhaltliche Suche in multimedialen Datenbeständen grundlegend. Weitere Anwendungsbeispiele wären die synchrone Darstellung von Partiturinformation während der Wiedergabe einer Audio-CD oder die synchrone Darstellung der entsprechenden Partiturstellen bei einer Aufführung eines Klavierstücks zu pädagogischen Zwecken.

Hintergrund

Ein großes Problem bei multimedialen Datenbanken sind die verschiedenen Datenformate, die allein schon bei reinen Musikdaten anzutreffen sind. Gängige Datenformate sind Partiturdatenformate wie Score oder Capella, partiturnahe Datenformate wie MIDI sowie das wellenformbasierte PCM-Datenformat. Dabei kann das MIDI-Datenformat als eine Art Bindeglied zwischen den anderen Datenformaten angesehen werden, denn es kann auf der einen Seite zwar wesentliche Pariturinformationen enthalten, auf der anderen Seite aber auch Interpretationen des jeweiligen Musikstücks erfassen. Der Schwerpunkt dieser Arbeit ist die Entwicklung von *Synchronisationstechniken*, die zur automatischen Verlinkung von partiturnahen Daten (MIDI) und akustischen Wellenformdaten (PCM) eingesetzt werden können. Zur besseren Einordnung und Abgrenzung des Themas wollen wir im Folgenden auf verwandte Problemstellungen wie die Segmentierung, Extraktion, Musiktranskription und Synchronisation eingehen und anschließend einen Überblick zum Stand der Forschung geben.

In der Audiosignalverarbeitung versteht man unter der *Segmentierung* die zeitliche Unterteilung eines akustischen Signals in logisch zusammenhängende Bereiche. Die Segmentierung eines Musikstücks kann zum einen auf rein inhaltsbasierten Kriterien beruhen, wie z. B. die Unterteilung einer Sonate in einzelne Sätze, die Unterteilung eines Sonatensatzes in Exposition, Durchführung und Reprise und diese wiederum in musikalische Phrasen. Auch die Einteilung eines Musikstücks in Takte kann als zeitliche Segmentierung aufgefasst werden. Im Gegensatz hierzu geht es beim Segmentierungsproblem von Audiosignalen meist um die Zerlegung in elementare Bestandteile. Dabei handelt es sich im Fall von Sprachsignalen bei den Segmenten zum Beispiel um Phoneme oder im Fall von monophonen Musikstücken um die einzelnen Töne. Im letzteren Fall sind die Segmentgrenzen also gerade die Anfangs- und Endpunkte der Töne, die es zu bestimmen gilt [72]. In dieser Arbeit handelt es sich bei den Audiosignalen typischerweise um Aufnahmen polyphoner Musikstücke, die im PCM-Datenformat vorliegen. Selbst im Fall eines einzelnen Instruments – wir konzentrieren uns im Folgenden auf das Klavier – ist es bei polyphoner Musik nicht mehr offensichtlich, wie die Segmentgrenzen sinnvoll definiert werden können. Bei den entsprechenden Audiosignalen handelt es sich hierbei um komplexe Klanggemische, die sich aus neu hinzukommenden und gleichzeitig schon abklingenden Tönen und Akkorden zusammensetzen. Hierbei sei betont, dass schon einzelne Töne komplexe, sich aus den Partialtönen konstituierende Klänge darstellen. In Verbindung mit Resonanzeffekten führt die Überlagerung von verschiedenen Tönen zu überraschenden Phänomenen sowohl im Frequenz- als auch im Amplitudenverlauf der einzelnen Partialtöne (siehe auch Kapitel 2). Aufgrund dieser Phänomene ist ein allgemeinerer

Segmentierungsbegriff nötig. Segmentgrenzen werden definiert durch die Zeitpunkte, an denen abrupte Veränderungen im Spektrum wie auch in der Energie des Signals auftreten. Bei den Segmenten handelt es sich daher um Abschnitte, in denen sich das Audiosignal nur unwesentlich verändert und deshalb als quasiperiodisch angenommen werden kann (siehe auch Abschnitt 2.4). Die so definierten Segmentierungsgrenzen stellen somit Kandidaten für die Einsatzzeiten von Tönen dar.

Das Segmentierungproblem kann als Teilproblem des allgemeineren *Extraktionsproblems* angesehen werden. In der Audiosignalverarbeitung geht es bei der *Extraktion* um das „Herausziehen“ geeigneter Parameter aus einem Musiksignal, welche – quasi als Extrakt – den zugrundeliegenden musikalischen Gehalt beschreiben. Hierzu zählen neben den oben erwähnten Kandidaten für die Einsatzzeiten insbesondere auch spektrale Parameter zur Beschreibung der Tonhöhen sowie Parameter zur Beschreibung der Tondauern und Lautstärken. In diesem Zusammenhang können z. B. *Beat-* und *Tempotracking* als Spezialfälle des Extraktionsproblems angeführt werden, bei denen insbesondere zeitliche und dynamische Parameter geeignet interpretiert werden müssen [11, 35, 36, 84] und in manchen Fällen die spektralen wiederholten Strukturen [30]. Bei der *automatischen Begleitung* werden vor allem spektrale Parameter zur Bestimmung von Tonhöhen eingesetzt [19, 72], manchmal sogar in der Kombination mit Energieparametern [9]. Es sei an dieser Stelle erwähnt, dass die Bestimmung von Tonhöhen aus den entsprechenden spektralen Parametern bei polyphoner Musik ein meist noch ungelöstes Problem darstellt. Dies liegt unter anderem daran, dass sich in dem Klanggemisch Partialtöne überlagern und sich damit nicht mehr eindeutig den Einzeltönen zuordnen lassen. Mögliche Ansätze zur Lösung des Tonhöhenextraktionsproblems sind die Verwendung von Instrumentenmodellen und Notenschablonen [6, 64], oder die Verwendung von Zusatzinformation z. B. in Form von Partiturdaten [83]. Diese beiden Ansätze werden auch in dieser Arbeit eine wichtige Rolle spielen (siehe Kapitel 4).

Bei der *Musiktranskription* geht es grob gesprochen um die Übertragung einer Musikaufnahme in Notenschrift. Mit anderen Worten sollen aus einem Audiosignal die Partiturdaten des zugrundeliegenden Musikstücks – wie z. B. die einzelnen Töne, Instrumente, Taktart, Tempo- oder Lautstärkebezeichnungen – gewonnen werden. Damit stellen Transkription und Extraktion ähnliche Probleme dar, die sich aber in einem wesentlichen Punkt unterscheiden. Während es sich bei den Partiturdaten um „normierte“ Daten handelt, die dem Musiker als Spielanleitung viele interpretatorische Freiheiten wie lokale Temposchwankungen und Klangfarbenmanipulationen lassen, beinhalten die extrahierten Parameter eines Musiksignals implizit sowohl die Partiturinformationen als auch die spezifischen interpretatorischen Merkmale der jeweiligen Aufnahme. Damit bietet sich für das Transkriptionsproblem folgende Vorgehensweise an. In einem ersten Schritt werden geeignete musikalische Parameter extrahiert, aus denen Einsatzzeiten, Tonhöhen, Tondauern etc. bestimmt werden. In einem zweiten Schritt müssen diese Daten durch geeignete Quantisierungsverfahren „bereinigt“ und normiert werden. Somit kann die Extraktion als eine Vorstufe zur Transkription aufgefaßt werden. Stellt schon die Extraktion von Tonhöhen aus Aufnahmen polyphoner Musikstücke, wie oben erwähnt, ein weitgehend ungelöstes Problem dar, so gilt dies erst recht für die automatische Musiktranskription . Spezialfälle zur Musiktranskription werden u. a. in [10] behandelt.

Unter der *Synchronisation* versteht man ganz allgemein die zeitliche Abstimmung prinzipiell unabhängiger Prozesse, wenn diese in Wechselwirkung miteinander treten oder sich logisch bedingen. Zum Beispiel müssen in multimedialen Anwendungen mehrere Datenströme ver-

schiedener Formate wie Video, Audio und Text zur gleichzeitigen Wiedergabe synchronisiert werden. In dieser Arbeit verstehen wir unter dem *Synchronisationsproblem* die zeitliche Synchronisierung verschiedener Varianten eines Musikstücks, die als unabhängige Datenströme normalerweise unterschiedlichen Datenformats gegeben sind. Die Synchronisation wird in Form einer geeigneten Verlinkung realisiert. Hierbei handelt es sich z. B. bei dem einen Datenstrom um die CD-Aufnahme eines Musikstücks, bei dem anderen um die reinen Partiturdaten, kodiert in einem geeigneten Datenformat. In diesem Fall kann das Synchronisationsproblem in folgender Weise als eine zur Musiktranskription komplementäre Aufgabenstellung aufgefasst werden. Geht es bei der Musiktranskription um die Bestimmung der Partiturdaten aus einer gegebenen Musikaufnahme, setzt man beim Synchronisationsproblem das Vorliegen beider Datenströme voraus. Während es das Ziel der Transkription ist, die extrahierten Daten von den interpretatorischen Freiheiten (z. B. lokale Temposchwankungen) der vorliegenden Aufnahme zu bereinigen, geht es bei der Synchronisation um die Erfassung gerade dieser interpretatorischen Abweichungen der Musikaufnahme von der zugehörigen Partitur. Wir werden das Synchronisationsproblem in Kapitel 5 genau spezifizieren.

Die oben erwähnten Aufgaben der automatischen Segmentierung, Extraktion und Transkription polyphoner Musik sind bisher nur teilweise und mit einem eingeschränkten Erfolg gelöst. Wie wir in Kapitel 2 sehen werden, haben wir es mit einem sehr komplexen Problem zu tun, für dessen Lösung eine bedeutende Menge an Heuristiken notwendig sind. Die Idee, Modelle von Instrumentenklängen zu verwenden, wurde von mehreren Autoren verfolgt [6, 50, 64, 72]. Dabei sind diese Modelle je nach Anwendung in Form von Notenschablonen, neuronalen Netzen oder HMMs realisiert. Um eine instrumentenunabhängige Lösung zu erreichen, haben manche Autoren Lösungen, basierend auf Blackboard-Architekturen, vorgeschlagen, die eine spektrale Analyse mittels des sogenannten Log-Lag-Autokorrelogramms realisieren. Zugleich wird bei der Bestimmung von Akkorden Wissen aus der Kontrapunktlehre verwendet [51]. Dabei hat man sich auf eine bestimmte Musikrichtung, ein Instrument und eine feste Anzahl von Stimmen eingeschränkt. In einer aktuelleren Arbeit [43] wird sogar die Extraktion von Tonhöhen aus noch komplexeren Klängen, die von mehreren gleichzeitig spielenden Instrumenten stammen, untersucht. Hier werden weder Klangmodelle noch andere Heuristiken verwendet. Die Erkennung der Stimmen wird durch spektrale Glättung mittels Moving-Average-Techniken und Subtraktionsmethoden realisiert. Allerdings werden – so die Autoren – nur einige der wichtigsten Stimmen erkannt.

Nach dem heutigen Stand der Forschung folgt daraus, dass keines der bisher realisierten Systeme, selbst für eingeschränkte „Laborbedingungen“, hinreichend robuste bzw. fehlerfreie Ergebnisse liefert. Es ist zu bemerken, dass die bisher noch nicht untersuchte Kombination von Instrumentklangmodellen und musiktheoretischem Wissen eine vielversprechende, aber zugleich äußerst komplizierte Lösung darstellen könnte. Da aber die Extraktion der Musikparameter nur ein Teil des Synchronisationsproblems darstellt, haben wir diese Idee hier nicht weiter verfolgt.

Zuletzt wollen wir darauf hinweisen, dass die Begriffe der Extraktion und Transkription in der Literatur nicht sauber genug abgegrenzt sind. Oft wird die Extraktion der Einsatzzeiten und Tonhöhen mit der automatischen Musiktranskription gleichgesetzt, auch wenn sie, unserer Meinung nach, weit von der endgültigen Transkription entfernt liegt. Wir werden in dieser Arbeit vor allem den Begriff Extraktion bevorzugen. Beim Zitieren von anderen Arbeiten werden wir aber die von den Autoren verwendeten Bezeichnungen verwenden. Wir bemerken,

dass diese jedoch oft sehr vorsichtig zu „genießen“ sind.

Beitrag und Gliederung dieser Arbeit

Diese Arbeit beschäftigt sich mit dem Entwurf eines Gesamtsystems zur Zeitsynchronisation von Partitur-, MIDI- und PCM-Daten im Fall von polyphoner Musik. Der allgemeine Fall beliebiger Orchestermusik scheint mit den derzeitigen Methoden und technischen Möglichkeiten völlig außer Reichweite zu sein. Wir haben uns daher auf den Fall von Klaviermusik beschränkt, was aber aufgrund der Polyphonie immer noch eine große Herausforderung darstellt. Viele der in dieser Arbeit beschriebenen Methoden und entwickelten Algorithmen lassen sich aber gut auch auf andere Instrumentgruppen übertragen. Im Folgenden gehen wir auf die Gliederung dieser Arbeit ein und fassen die Inhalte der einzelnen Kapitel zusammen. Dabei wird auch auf den jeweiligen Beitrag der Arbeit eingegangen. Jedes Kapitel beginnt mit einer kurzen Übersicht und an gegebener Stelle wird der jeweilige Stand der Forschung erläutert.

Kapitel 2 geht auf die hier grundlegenden Aspekte der Musik ein. Hierzu zählen neben den verschiedenen Darstellungsformen mit den zugehörigen Datenformaten (Partiturformate, PCM-Format, MIDI-Format) auch Aspekte der Interpretation und Wahrnehmung (Psychoakustik) von Musik. Weiterhin werden die für das Extraktionsproblem grundlegenden Charakteristika des Klavierklangs und die sich daraus ergebenden Folgerungen diskutiert.

Kapitel 3 stellt die hier benötigten signaltheoretischen und mathematischen Grundlagen dar und legt damit die Bezeichnungskonventionen fest. Insbesondere werden die benötigten Signalräume, Operatoren und Signaltransformationen, wie z. B. die gefensterten Fouriertransformationen und Multiratenfilterbänke, eingeführt. Die Grundbausteine für spezielle Filterbankbäume, deren Struktur an die Frequenzcharakteristik das Klaviers angepasst ist und deren Auflösung die Klaviertöne zu trennen vermag, werden entworfen.

Kapitel 4 befasst sich mit der Extraktion musikalischer Merkmale aus PCM-Daten. Diese Parameter dienen als Ausgangsmaterial für das im nächsten Kapitel behandelte Synchronisationsproblem von PCM- und partiturähnlichen Daten. Daher steht weniger die vollständige Extraktion aller Partiturdaten im Vordergrund (wie es z. B. beim Transkriptionsproblem nötig wäre), sondern vielmehr die Extraktion einer hinreichend großen Menge an Parametern, die eine Synchronisation einer vorgegebenen Auflösung ermöglicht. Hierfür eignen sich insbesondere die Einsatzzeiten und Tonhöhen. Zur Bestimmung der Einsatzzeiten werden drei verschiedene Ansätze – basierend auf der autoregressiven Methode (LPC-Methode), einer amplitudenbasierten Methode und unter Verwendung von Novelity-Kurven – zur Segmentierung von Audiosignalen vorgestellt. Anhand der Ergebnisse zahlreicher Experimente werden die Vor- und Nachteile dieser Ansätze diskutiert. Anschließend geht es um die Extraktion von Tonhöhen. Hierbei werden unter Verwendung geeigneter Multiratenfilterbänke Spektralparameter berechnet, aus denen sich schließlich mit Hilfe von Notenschablonen die Tonhöhen bestimmen lassen. Hierfür wurde eine neuartige Strategie entwickelt, die im Gegensatz zu energiebasierten Strategien [6], [43] das Frequenzspektrum von unten nach oben durchkämmt, so dass auch leise, energiearme Töne besser erkannt werden können. Alle Algorithmen wurden in MATLAB implementiert und auf einem umfangreichen Datenmaterial getestet. Eine Diskussion einiger der Experimente und Ergebnisse runden das Kapitel ab.

Kapitel 5 diskutiert algorithmische Lösungen diverser Synchronisationsaufgaben. Ausgangspunkt ist ein Musikstück, das in unterschiedlichen Formaten (Partitur, MIDI, Wellenform- oder PCM-Darstellung) vorliegt. Demgemäß gibt es drei Arten von Synchronisationsproblemen nämlich

- PM: Synchronisation von **P**artitur und **M**IDI
- PW: Synchronisation von **P**artitur und **W**ellenformdarstellung
- MW: Synchronisation von **M**IDI und **W**ellenformdarstellung

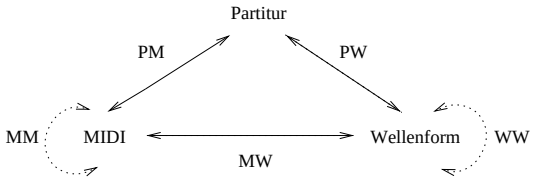


Abbildung 1.1: Verschiedene Synchronisationsaufgaben.

Wir beschränken uns auf polyphone Klaviermusik.

Zur PM-Synchronisation sind schon algorithmische Lösungen vorgeschlagen worden, siehe [24, 38, 48]. Allerdings bleibt eine experimentelle Verifikation dieser Algorithmen meist aus oder beschränkt sich auf einfache Beispiele.

Das PW-Synchronisationsproblem ist bisher kaum studiert worden. Es wird in dieser Arbeit unseres Wissens zum ersten Mal in solcher Allgemeinheit behandelt. Die besondere Schwierigkeit dieser Aufgabe liegt in der Tatsache begründet, dass die Notenereignisse wie Einsatzzeiten und Tonhöhen nicht explizit vorliegen, sondern erst extrahiert werden müssen. Die dabei unvermeidlichen Fehler müssen im Nachhinein auf geeignete Weise kontrolliert werden.

Als noch schwieriger gestaltet sich die Lösung des MW-Synchronisationsproblems, da die MIDI-Datei als eine weniger sichere Referenzdatei im Vergleich zur Partitur angesehen werden kann.

MM-Synchronisation bzw. WW-Synchronisation, d. h. die Synchronisation zweier MIDI-Versionen bzw. zweier PCM-Versionen ein und desselben Klavierstücks, werden hier nicht weiter betrachtet, könnten aber etwa über den Umweg

$$\begin{array}{lcll} PM_1 & \& & PM_2 & \longrightarrow & M_1M_2 \\ PW_1 & \& & PW_2 & \longrightarrow & W_1W_2 \end{array}$$

realisiert werden.

Der fünfte Kapitel beginnt mit einer genaueren Problemspezifikation und geht dann im ersten Abschnitt kurz auf den Stand der Forschung zu Synchronisationsproblem in der Musik ein. Die Abschnitte 5.2 bis 5.4 widmen sich den drei Synchronisationsaufgaben (PM, PW, MW). In allen drei Fällen werden die Daten vorverarbeitet: um die Synchronisation robust gegenüber unscharfen (impliziten) Notenereignissen in der Partitur (wie z. B. Triller, Arpeggien oder Verzierungen) zu machen, werden diese mittels sogenannter Fuzzy-Noten modelliert und gesondert behandelt. MIDI-Dateien müssen aus technischen Gründen sowie zur Steigerung der

Robustheit zeitquantisiert werden, während aus den PCM-Dateien mit den in Kapitel 5 beschriebenen Verfahren Kandidaten für Einsatzzeiten und Tonhöhen extrahiert werden. Dem schließt sich ebenfalls eine Zeitquantisierung an. Auf den so vorverarbeiteten Daten werden nun auf der Basis geeigneter Kostenfunktionen mittels dynamischer Programmierung kostenoptimale zeitliche Verlinkungen (Matchings) berechnet. Die Kostenfunktionen hängen von mehreren Parametern ab. Im fünften Abschnitt gehen wir näher auf die Parameterwahl ein. Unsere Verfahren wurden anhand zahlreicher Beispiele unterschiedlicher Komplexität getestet. Wir beschließen das Kapitel mit der Zusammenfassung der Testergebnisse und einem Resumé.

Danksagung

Diese Arbeit entstand in der Abteilung III des Instituts für Informatik an der Rheinischen Friedrich-Wilhelms-Universität. An dieser Stelle bedanke ich mich herzlich bei den zahlreichen am Gelingen dieser Arbeit direkt oder indirekt beteiligten Personen.

Hierzu zählen in erster Linie meine Eltern und meine Familie, die mich durch mein gesamtes Studium in vielerlei Hinsicht selbstlos unterstützt haben.

Ein Stipendium des Deutschen Akademischen Austauschdienstes ermöglichte mir, mein Promotionsstudium in Deutschland zu beginnen.

Viele Anregungen bekam ich durch sehr fruchtbare Diskussionen, die ich insbesondere mit Frank Kurth und Meinard Müller führen durfte.

Außerdem danke ich allen Mitarbeitern unserer Arbeitsgruppe für die offene und freundschaftliche Atmosphäre, insbesondere Andreas Beschorner, Roland Engelbrecht, Heiko Goeman, Frank Kurth, Dirk Meyer, Axel Mosig, Meinard Müller und Andreas Ribbrock.

Bei Lule Ahmedi, Naim Bajçinca, Burbuqe Hana, Mevlyde Kasumi, Arbana Metaj, Nathalia Peixoto und Suada Sultanić bedanke ich mich für ihre wertvolle Freundschaft. Sie sind für mich ein Beweis dafür, dass Entfernungen von mehreren tausend Kilometern wahren Beziehungen nicht hinderlich sind.

Herrn Hans-Georg Müller danke ich für die sprachliche Korrektur dieser Arbeit.

Bei Herrn Prof. Dr. R. Manthey bedanke ich mich für die Übernahme des Korreferats.

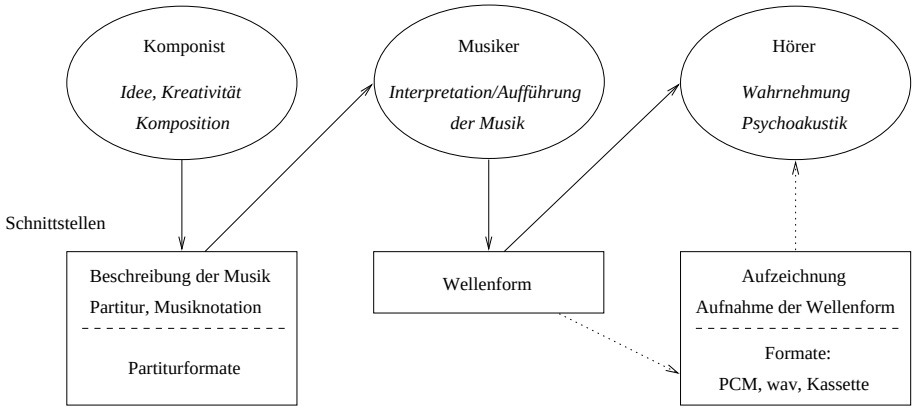
Zu besonders großem Dank bin ich meinem Doktorvater Prof. Dr. M. Clausen verpflichtet. Ohne seine unschätzbare, sowohl wissenschaftliche als auch moralische Unterstüzung, wäre diese Arbeit gar nicht erst entstanden.

Vlora Arifi, den 1. August 2002

Kapitel 2

Aspekte der Musik

Musik ist eine der ältesten menschlichen Kunst- und Kommunikationsformen. Der Begriff *Musik* leitet sich vom griechischen Wort *musiké* ab, worunter das griechische Altertum zunächst die musischen Künste *Dichtung, Musik und Tanz* als eine Einheit, dann die *Tonkunst* im Besonderen verstand [58]. Musik ist ein komplexes Geflecht verschiedener Elemente, angefangen bei der geistigen Idee bis hin zum akustischen Material. Diese Elemente stehen nicht wie Inhalt und Form nebeneinander, sondern verbinden sich in der Musik ganzheitlich. Im Folgenden gehen wir auf die für diese Arbeit benötigten Aspekte der Musik ein, die in der folgenden Abbildung schematisch dargestellt sind.



Ausgangspunkt der Musik, insbesondere bei der mehrstimmigen Musik des Abendlandes seit dem 12. Jahrhundert, ist oft die geistige Idee des Komponisten. Dieser fixiert seine Komposition mittels einer *Notenschrift*, durch die die Musik über verschiedene Parameter wie Tonhöhe und Tondauer der Noten beschrieben wird (Abschnitt 2.1). Unter Verwendung der Partiturinformation, ergänzt durch tieferes Wissen um den Gehalt und die Gestalt der Musik, realisiert der Musiker den Klang der Komposition. Da nicht alle Parameter der Musik durch die Partitur erfasst sind, bleiben dem Künstler dabei viele Freiheiten, was zu verschiedenen *Interpretationen* ein und desselben Musikstücks führt (Abschnitt 2.2). Über die *akustische Wellenform*, die Inhalt des Abschnitts 2.3 ist, erreicht die Musik den Hörer. Diese Wellenform kann durch Modelle beschrieben werden, die ihre quasiperiodische Natur berücksichtigen (siehe Abschnitt 2.4). Während oder nach der Aufzeichnung der akustischen Wellenform wird das gewonnene

Audiosignal zur Zweck der Verarbeitung und Abspeicherung in digitalen Medien geeignet abgetastet (Abschnitt 2.5). Die Auswahl der Abtastrate basiert auf zwei Kriterien: die spektrale Energiekonzentration des Signals und die selektive Natur des menschlichen Gehörs. Auf die *psychoakustischen Effekte*, die für die Wahrnehmung und das Hörempfinden eine wichtige Rolle spielen, wird in Abschnitt 2.6 eingegangen. In dieser Arbeit steht insbesondere vom Klavier erzeugte Musik im Mittelpunkt der Untersuchungen. Daher gehen wir in Abschnitt 2.7 gesondert auf die Charakteristika des Klavierklangs ein. Das MIDI-Datenformat, das als Standard zur Ansteuerung von digitalen Instrumenten entwickelt wurde, nimmt eine Art Zwischenstellung ein, da es sowohl reine Partiturinformation kodieren, aber auch interpretatorische Feinheiten erfassen kann. Damit eignet sich das MIDI-Format, auf das in Abschnitt 2.8 eingegangen wird, besonders gut für die Aufgaben, die in dieser Arbeit in Angriff genommen werden.

2.1 Musiknotation

Wie bereits in der Einleitung dieses Kapitels erwähnt, beschreibt der Komponist sein Musikstück unter Verwendung festgelegter Symbole und unter Einhaltung vorgegebener Regeln durch eine *Notenschrift* bzw. durch die *Partitur*, wodurch die Musik in einer lesbaren Form fixiert wird (vgl. [58]). Wir beschränken uns hier auf die klassische westliche Musiknotation, auf deren Elemente im Folgenden kurz eingegangen werden soll.

Das *Notensystem* besteht aus insgesamt fünf Notenlinien und eventuell zusätzlichen Hilfslinien, in die die Symbole für die Noten, Pausen, Vorzeichen, Akzente usw. eingetragen werden. Zur Festlegung der Tonhöhe dient der *Notenschlüssel*, der am Anfang des Notensystems steht. In der Klaviermusik wird neben dem G- oder *Violinschlüssel* für die tieferen Noten der F- oder *Bassschlüssel* verwendet. Andere Schlüsselarten, wie z. B. der Tenorschlüssel, der Altschlüssel, Mezzosopran- oder Sopranschlüssel, werden für bestimmte Instrumente je nach ihrer Tonlage gebraucht, sind aber für uns im Folgenden nicht von Bedeutung.

In ein solches mit dem entsprechenden Schlüssel versehenes Notensystem kann eine Komposition notiert werden. Die Tonhöhe einer Note ist dabei durch die vertikale Position des entsprechenden Notensymbols im Notensystem festgelegt, die Tondauer durch die Form des Symbols. Die Tondauer ist durch eine rationale Zahl beschreibbar, die das Verhältnis der Notendauern der verschiedenen Noten untereinander beschreibt. Das Gleiche gilt für die *Taktart*, durch die das Musikstück in periodisch wiederkehrende Gruppen von betonten und unbetonten Pulsschlägen gegliedert wird. Bei *polyphoner Musik*, d. h. mehrstimmiger Musik, werden die gleichzeitig zu spielenden Noten, z. B. die Noten eines Akkords, übereinander im Notensystem notiert. Wenn es um mehrere Stimmen oder Instrumente geht, werden diese in Form einer *Partitur* in übersichtlicher Schichtung jeder dieser Stimmen aufgezeichnet. Hierbei stehen alle gleichzeitig zu spielenden Noten und Pausen genau übereinander.

Die absoluten, in Sekunden ausdrückbaren zeitlichen Dauern der Noten bzw. Takte ergeben sich aus der Angabe einer mit dem sogenannten *Mälzelschen Metronom* (M. M.) bestimmbaren bzw. vorgegebenen Zahl, welche die Anzahl der Schläge einer Grundnotendauer pro Minute angibt. Meist ist das Tempo einer Komposition aber nicht exakt, sondern nur durch eine textuelle Tempobezeichnung, dem *Grundtempo*, angegeben. Begriffe wie *Allegro* oder *Moderato* sind eine vage Tempoangabe, die nur in Verbindung mit dem musikalischen Verlauf

des Stückes aussagekräftig ist (vgl. [39]). Darüber hinaus gibt es Bezeichnungen für *lokale Tempoänderungen* wie z. B. *accelerando*, *ritardando* oder *a tempo*, die eine Beschleunigung, Verlangsamung oder die Rückkehr in das Grundtempo fordern.

Ähnlich wie bei der Tempoangabe werden auch für die Beschreibung der *Dynamik* textuelle Bezeichnungen verwendet. Absolute *Lautstärkegrade* wie z. B. *pp*, *mf*, *f* oder *fff* gelten allgemein für den gesamten Abschnitt bis hin zur nächsten Dynamikangabe. Bezeichnungen wie z. B. *crescendo* oder *decrescendo*, oft auch durch Symbole auseinanderlaufender oder zusammenlaufender spitzer Klammern beschrieben, regeln den *lokalen Lautstärkeverlauf* der Komposition.

Verzierungen einer Melodie werden meist nicht explizit notiert, sondern nur durch Sonderzeichen oder kleingestochene Zusatznoten angedeutet. Als Beispiele können hier Arpeggio, Triller, Vorschlag, Doppelvorschlag, Nachschlag oder Tremolo angeführt werden.

In der neueren Zeit sind auf dem Markt Notationsprogramme wie *Capella*, *Finale*, *Score* und *Sibelius* erschienen. Sie unterstützen mehr oder weniger die Aufzeichnung der wichtigsten Notationssymbole und unterstützen (erleichtern) wesentlich die Arbeit der Komponisten. Das Notieren der Musik mittels PCs erlaubt, neben den bekannten Vorteilen der digitalen Datenverarbeitung wie z. B. das Korrigieren, Kopieren und Einfügen von Symbolen und Text, auch die Umwandlung der internen Datenformate in das verbreitete und von vielen Notationsprogrammen unterstützte MIDI-Format (siehe Abschnitt 2.8). Es ist leider zu bemerken, dass bei dieser Umwandlung viele Elemente, wie z. B. Tempoangaben, Gruppierungen und Verzierungszeichen, ignoriert werden.

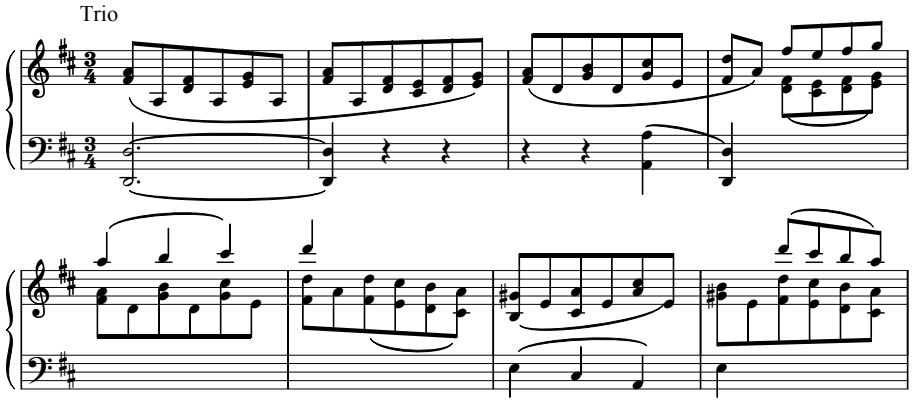


Abbildung 2.1: Die Notation der ersten acht Takte des Trios aus Mozarts Klaviersonate in A-Dur, Op. 331, erzeugt durch Capella.

2.2 Interpretation von Musik

Die im letzten Abschnitt beschriebe Musiknotation reicht selbst bei den einfachsten Musikstücken nicht dazu aus, diese exakt zu beschreiben. Die Partitur enthält zwar die wesentlichen inhaltlichen Parameter eines Musikstücks, dient aber letztendlich nur als Spielanleitung für den Musiker. Von ihm hängt es ab, wie er die Spielanleitung *interpretiert* und damit die von dem Komponisten erdachte Musik reproduziert. Der Musiker übernimmt damit nicht nur

die Rolle des puren „Klangerzeugers“, sondern die vielleicht noch wichtigere Rolle des *Interpreten*. Dabei hat er insbesondere bei der Organisation des Tempos, aber auch bezüglich der Tonhöhe und des Timbres große Freiheiten (vgl. [79]). Die neuere Musik ist meist ziemlich vollständig notiert, und es gibt sogar Interpretationen, die direkt vom Komponisten angeleitet werden. Bei der Musik früherer Jahrhunderte stoßen wir für unsere heutigen Begriffe jedoch auf eine ziemlich vage Notation bezüglich Tempo, Lautstärke und Artikulation.

Die vage Notation ist aber nur ein Grund für das Vorliegen unterschiedlicher Interpretationen ein- und desselben Musikstücks. Jede durch herkömmliche Musikinstrumente (d. h. nicht auf elektronische Weise) erzeugte Interpretation eines Musikstückes hat ihren eigenen Charakter. Einem Musiker wird es nicht gelingen, zwei identische Interpretationen zu realisieren. Hierfür sind u. a. die sogenannten *Mikrodeviationen* der Einsatzzeiten (vgl. [8]), auch als *Random Timing Noise* bezeichnet (vgl. [22]), als Grund zu nennen. Gerade diese kleinen lokalen Abweichungen der Noteneinsatzeiten von den metronomisch exakten Positionen machen das abstrakt notierte Musikstück erst zur Musik, verleihen der Musik Lebendigkeit und Ausdruckskraft. Man denke als Vergleich nur an das stupide und leblose Abspielen einer Partitur durch Notations- und Sequencer-Programme.

Die zeitlichen Mikrodeviationen werden auch zum Zweck der Klangfarbenmanipulation verwendet, so verleiht z. B. das leicht versetzte Abspielen der Töne eines Akkords dem Klang ein anderes Timbre. Weiterhin können sie eine Akzentuierungsfunktion haben wie z. B. im Fall der Melodieführung, wo der Melodieton eines Akkords manchmal zeitlich leicht abgesetzt gespielt wird (Asynchronizitäten von bis zu ca. 30 Millisekunden, siehe Abschnitt 2.6). Manchmal sind zeitliche Asynchronizitäten als eine Abspielmöglichkeit erlaubt, z. B. wenn die Töne des Akkords sehr weit voneinander entfernt liegen, so dass sie nicht alle für jeden Pianisten gleichzeitig greifbar sind und deshalb als Arpeggio gespielt werden. Zuletzt sind zeitliche Mikrodeviationen manchmal auch nur auf unbeabsichtigte Ungenauigkeiten des Musikers zurückzuführen.

Da das *Tempo* einen für diese Arbeit fundamentalen Aspekt der Musik darstellt, wollen wir im Folgenden hierauf genauer eingehen. Das Tempo eines Musikstücks ist im Allgemeinen sehr großen interpretationsbedingten Schwankungen unterworfen. Hierbei handelt es sich zum einen um Abweichungen im *Grundtempo*, die sozusagen auf einer globalen Ebene operieren und das gesamte Musikstück bzw. den gesamten Satz betreffen. Zum anderen treten erhebliche lokale Temposchwankungen innerhalb des jeweiligen Musikstückes auf. Es gibt zahlreiche Kompositionen, wo von verschiedenen Interpreten völlig unterschiedliche Grundtempi gewählt werden. Beethovens *Große Hammerklavier Sonate*, Op. 106 in B-Dur, ist ein sehr gutes Beispiel dafür. Der erste Satz der Sonate, bei dem Beethoven ein Tempo von 138 Halbenoten pro Minute vorgibt – ein praktisch nicht realisierbares Tempo – wurde von A. Schnabel und F. Gulda gelegentlich mit einem Tempo von ca. 126 - 132 Halben pro Minute gespielt, während Solomon und Ch. Rosen ein Tempo von ca. 120 Halben pro Minute wählten (vgl. [41]). Viele Pianisten spielen diesen Satz mit einem Tempo zwischen 104 und 116, manche sogar mit einem Tempo unter 90 Halben pro Minute. Die Gesamtspieldauer des ersten Satzes der Sonate beträgt somit abhängig vom Interpreten zwischen 8,75 (Schnabel) und 13 Minuten (Barenboim) – ein erheblicher Unterschied.

Ein weiteres häufiges Phänomen sind große Temposchwankungen innerhalb eines Musikstückes. Am Beispiel Beethovens *Grande Sonate Pathétique*, Opus 13, in c-moll wurden in [41] die Tem-

pi innerhalb des ersten Satzes für die drei in der Anzahl der Takte gleichlangen Abschnitte des ersten Themas, des Seitensatzes und der Schlussgruppe für verschiedene Interpreten untersucht. Die Ergebnisse zeigten, dass die Temposchwankungen insbesondere bei Interpreten der älteren Pianistenschule sehr groß waren (Abweichungen von bis zu 4 Sekunden zwischen den unterschiedlichen Abschnitten, was einer Temposchwankung von 8-10% entspricht), während die Abschnitte von den eher jüngeren Interpreten relativ gleichmäßig gespielt wurden, so dass die Schwankungen fast unbemerkbar waren (unter einer Sekunde bzw. 1-3%). Auch im Fall der *Waldsteinsonate*, Opus 53 in C-Dur, traten ähnliche Phänomene auf: Die ersten 12 Takte des Beginns der Sonate, verglichen mit den ersten 12 Takten des Seitenthemas, stehen in einem Ungleichheitsverhältnis von – je nach Interpreten – ca. 20:25 (Elly Ney), 15:19 (Gulda), 18:23 (Horowitz), 17:21 (Solomon) oder 18:24 (Gilels).

Die oben diskutierten Abweichungen im Grundtempo und die Temposchwankungen wirken sich nicht nur auf die Gesamtdauer des Musikstücks aus, sondern haben auch weitreichende Folgen für die relativen Tondauern, was zu lokalen Schwankungen der Einsatzzeiten führt. Wir konkretisieren diese Aussage anhand einiger Beispiele. Bei einem rascheren Grundtempo fällt im Allgemeinen ein *rubato*, also eine kontrollierte lokale Temposchwankung, wenig stark aus. Auch die Länge von sehr kurzen Noten wie Verzierungsnoten, insbesondere die des Typs *appoggiatura* (Vorschlag), verkürzen sich bei einem schnelleren Grundtempo meist nicht proportional zu den längeren Noten.

Ähnlich werden bei einem langsameren Grundtempo die Einsatzzeiten eines asynchron gespielten Akkords nicht dem langsameren Tempo entsprechend skaliert, da sie in solch einem Fall als Arpeggio klingen würden (vgl. [22]). Ein weiteres schönes Beispiel ist der Fall des Schlussritardandos, das in [23] in Abhängigkeit vom Grundtempo verschiedener Interpretationen ein und desselben Musikstücks untersucht wurde. Hierbei stellte sich heraus, dass das Schlussritardando relativ zum Grundtempo viel deutlicher ausfiel, wenn das Grundtempo schneller war.

Als ein letztes, hier erwähntes Beispiel für Uneindeutigkeiten der Musiknotation, die zu verschiedenen Interpretationen führen können, ist der Triller. Er bewirkt einen raschen Wechsel einer Hauptnote mit der kleinen oder großen Obersekunde (vgl. [58]) und kann sehr unterschiedlich gespielt werden, abhängig von der Epoche, dem Tempo und dem Interpreten. Einige mögliche Trillerexpansionen sind in Abbildung 5.2 illustriert.

Neben den zeitlichen Abweichungen bei verschiedenen Interpretationen, die in der Interpretationsfreiheit des Musikers begründet sind, tauchen bei vielen Einspielungen eines Musikstücks auch unabsichtliche Abweichungen von der Notation auf. Bei solchen Abweichungen handelt es sich z. B. um fehlende Töne, zusätzliche, in der Partitur nichtexistierende Töne oder eine „Kombination von beidem“ (vgl. [24]), nämlich falsch gespielten Tönen. Man spricht dann auch von *Einfüge-* (insertion), *Lösch-* (deletion) bzw. *Substitutionsfehlern* (vgl. [24]). Selbst den größten Interpreten gelingt es nicht, solche Fehler vollständig zu vermeiden. Dies bleibt jedoch dem menschlichen Ohr meist aufgrund der Wahrnehmungscharakteristika des menschlichen Ohres verborgen (siehe Abschnitt 2.6).

Am Beispiel des Tempoaspekts haben wir illustriert, dass die Notation eines Musikstücks dem Interpreten viele Freiheiten lässt. Die Notation ist nur eine grobe Beschreibung der wirklichen Musik und darf dieser nicht gleichgesetzt werden. Man könnte sagen, dass die Musiknotation einer Projektion der Musik auf einen äußerst kleinen Parameterraum entspricht. Die „Rekon-

struktion“ der Musik aus diesen unzureichenden Parametern kann damit nicht eindeutig sein und benötigt das Zusatzwissen des Interpreten. Eine tiefergehende Disskusion über die Interpretation von Musikstücken, auch hinsichtlich weiterer musikalischer Aspekte, liegt außerhalb des Rahmens dieser Arbeit. Wir verweisen auf die Literatur (siehe u. a. [47, 81]).

Zusammenfassend läßt sich sagen, dass bei der Interpretation insbesondere von klassischer Musik, viele Abweichungen vom Grundtempo auftreten, welche wiederum von Interpretation zu Interpretation völlig unterschiedlich ausfallen können. In Kapitel 5 werden wir den Begriff der *Tempokurve* formalisieren, der neben dem Grundtempo insbesondere auch die lokalen Temposchwankungen berücksichtigt. Die Bestimmung der Tempokurve einer Interpretation mittels eines automatischen Verfahrens stellt – wie wir in diesem Abschnitt plausibel gemacht haben – ein komplexes Problem dar. In Kapitel 5 gehen wir im Kontext des *Synchronisationsproblems* näher auf diese Problematik ein.

2.3 Wellenform und PCM-Format

Aus physikalischer Sicht erzeugt der Interpret durch seine Stimme oder unter Verwendung eines Musikinstruments ein *Audiosignal* in Form einer Luftdruckschwingung, die sich um ihre Quelle verbreitet und schließlich den Hörer erreicht. Graphisch kann eine solche Luftdruckschwingung als *Wellenform* dargestellt werden (Abbildung 2.2). Aus physikalischer Sicht stellt

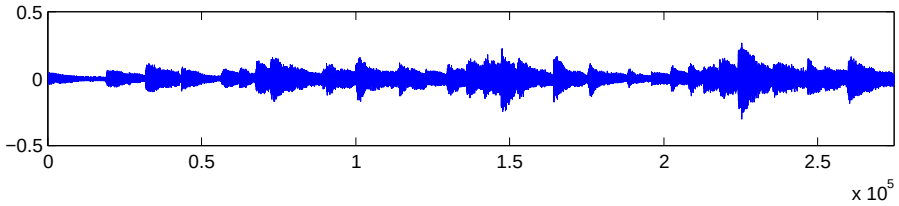


Abbildung 2.2: Die Wellenformdarstellung eines zwölfsekundigen Abschnitts des Trios aus Mozarts Klaviersonate in A-Dur, Op. 331 (CD-Aufnahme).

schon ein einzelner von einem Musikinstrument erzeugter Ton ein äußerst komplexes Klanggebilde dar, hier sei nur an die vorliegenden Obertöne und rauschartigen Komponenten erinnert. Zur besseren Abgrenzung zu einem reinen Sinuston sprechen wir in diesem Fall von einem *Musikton*, den wir uns auch als quasiperiodisches Audiosignal vorstellen können. In Abschnitt 2.4 diskutieren wir zwei mögliche mathematische Modellierungen von Musiktönen und ihren Wellenformen. Abschnitt 2.5 behandelt dann das für diese Arbeit wichtige *PCM-Format*, in dem ein Audiosignal in diskretisierter Form digital verarbeitet werden kann.

2.4 Modellierung von Musiktönen

Eine *reine Sinusschwingung* ist das durch die Amplitude A, Frequenz ω_0 und Phase θ definierte Signal

$$t \mapsto A \cdot sin(2\pi\omega_0 t + \theta).$$

Ein typischer Musikton stellt keine periodische Schwingung dar, besitzt aber eine gewisse quasiperiodische Natur. In seiner Sustain-Phase (siehe Abschnitt 2.7) ist es möglich, ihn als eine endliche Summe *harmonischer Sinusschwingungen* – das heißt reiner Sinusschwingungen, deren Perioden sich von einer Grundperiode nur um ein ganzzahliges Vielfaches unterscheiden – mit sich langsam verändernder Amplitude und Frequenz zu modellieren. Darüber hinaus enthält ein Musikton auch *nicht-periodische Komponenten* wie z. B. rauschartige Komponenten und sogenannte Transienten (siehe Abschnitt 2.7). In der folgenden mathematischen Modellierung eines Musiktons x werden diese nicht-harmonischen Komponenten, die meist Erscheinungen von kurzer Dauer und geringer Energie darstellen, vom quasiperiodischen Anteil des Musiktons getrennt und mit $\varepsilon(t)$ bezeichnet:

$$x(t) := \sum_{n=1}^{N} A_n(t) sin[2\pi n\omega_0 t + \theta_n(t)] + \varepsilon(t). \tag{2.1}$$

Hierbei sind $A_n(t)$ und $\theta_n(t)$, $n = 1, \ldots, N$ die zeitabhängigen Amplituden und Phasen der n-ten Komponente der Summe, wobei diese sich typischerweise nur sehr langsam in der Zeit verändern. Weiterhin bezeichne ω_0 die Grundfrequenz. Wir nehmen an, dass ein Musikton von endlicher Dauer ist, mit anderen Worten hat $A_n(t)$ einen kompakten Träger, gegeben durch ein Intervall $[a, b] \subset \mathbb{R}$, wobei a der Einsatzzeit und $b-a$ der Dauer des Klanges entspricht. Die Modellierung 2.1 wird auch *additive Synthese* genannt, weil hier die einzelnen Komponenten separat modelliert und schließlich addiert werden (vgl. [20], [71]).

Eine weitere sehr elegante Methode zur Modellierung von Musiktönen wird in [55] vorgestellt. Hier werden verschiedene Aspekte von Klängen parametrisiert, wobei die zugehörigen Parameterräume (physikalische, mathematische und interpretative Räume) untereinander in einer Beziehung stehen. Wir wollen im Folgenden kurz auf die physikalischen Aspekte dieser Modellierung eingehen. Zur Vereinfachung sollen hierbei bei der Modellierung nur die von der Schallquelle verursachten Schwingungen berücksichtigt werden. Im strengen Sinne müsste man auch die durch die Raumakustik bedingten Veränderungen der Schwingungen, welche wiederum von der Konstellation der in Schwingung versetzten Objekte im Raum abhängen, mitberücksichtigen. Eine formale Beschreibung all dieser Parameter in Abhängigkeit von Raum und Zeit wäre aber äußerst kompliziert und für unsere Zwecke nicht von wesentlicher Bedeutung.

Es sei zunächst x ein beliebiges Audiosignal wie z. B. die in Abbildung 2.2 dargestellte Wellenform. Wir betrachten die Folge der lokalen Maxima der Wellenform und verbinden diese durch einen Polygonzug. Dies definiert eine zeitabhängige Funktion, die wir mit H_x bezeichnen. Analog verfahren wir mit den lokalen Minima. Die beiden Polygonzüge definieren die sogenannte *Hüllkurve*, die in der Abbildung 2.3 dargestellt ist. Diese Grobdefinition soll ei-

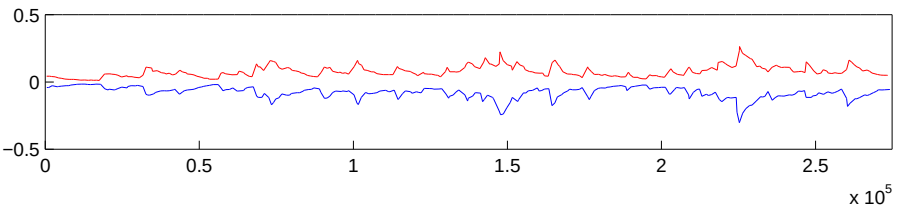


Abbildung 2.3: Hüllkurve der Wellenform von Abbildung 2.2

ne für unsere Zwecke ausreichende intuitive Vorstellung des Begriffs *Hüllkurve* geben. Die ADSR-Hüllkurven in Abschnitt 2.7 liefern weitere typische Beispiele.

Handelt es sich bei dem Audiosignal x um einen quasiperiodischen Musikton, so kann neben der Hüllkurve die Farbfunktion F_x definiert werden, die die harmonischen Informationen über x beinhaltet, also diejenigen Bestandteile, die für das Empfinden der Tonhöhe und der Klangfarbe (Timbre) des Musiktons verantwortlich sind. F_x ist nach Definition eine stetige, periodische Funktion $F_x : \mathbb{R} \to \mathbb{R}$ mit der Eigenschaft $F_x(t + T_0) = F_x(t)$, wobei T_0 die kleinstmögliche Periode bezeichnet. Eine solche Funktion kann durch eine Fourierreihe mit eindeutig bestimmten Koeffizienten α_r und δ_r dargestellt werden [27]:

$$F_x(t) = \alpha_0 + \sum_{r=1}^{\infty} \alpha_r \sin(2\pi r\omega_0 t + \delta_r),$$

wobei $\omega_0 := 1/T_0$ die Grundfrequenz des Musiktons ist. Der r-te Summand $\alpha_r \sin(2\pi r\omega_0 t+\delta_r)$ heißt die r-te *Partialschwingung* und wird auch als $(r-1)$-ter *Oberton* bezeichnet. In diesem Modell läßt sich der Musikton x wie folgt aus H_x und F_x rekonstruieren:

$$x(t) = H_x(t) \cdot F_x(t). \tag{2.2}$$

Die rauschartigen und transienten Komponenten sind in dieser Darstellung nicht miteinbezogen. Man könnte sie aber – so wie in (2.1) – getrennt vom quasiperiodischen Anteil des Klangs modellieren. Wir verweisen auf Abbildung 2.4 für eine typische Zerlegung eines abklingenden Musiktons in Hüll- und Farbkurve.

Bei einem Vergleich der Formeln (2.1) und (2.2) stellt man fest, dass die Amplitudenfunktionen $A_n(t)$ in der ersten Formel als Hüllkurven der entsprechenden Partialtöne interpretiert werden können. Die Modellierung in (2.1) ist damit wesentlich feiner und kommt der Realität auch näher, da die Amplitudenfunktionen $A_n(t)$ in der Praxis im Allgemeinen verschieden voneinander sind (siehe auch Abb. 2.9). Die Modellierung laut (2.2) ist aber gebräuchlicher, wenn man allgemein über die Amplitudenhüllkurve eines Audiosignals spricht, und sie ist wegen ihrer Kompaktheit in der Praxis leichter verwendbar, da Amplitudenparameter und Klangparameter besser separiert sind.

Überlagerung der Obertöne

Bei der Überlagerung zweier Musiktöne x_1 und x_2 entsteht ein komplexer Klang, dessen Klangfarbe sowohl von den Amplitudenfunktionen der einzelnen Partialtöne als auch von ihren Phasen abhängt (siehe [74]). Wir gehen hierauf genauer ein und schreiben x_1, x_2 und $x := x_1 + x_2$ gemäß (2.1) als

$$\begin{array}{l} x_1(t) := \sum_{n=1}^{N} A_{1,n}(t) sin[2\pi n\omega_{1,0} t + \theta_{1,n}(t)] + \varepsilon_1(t) \\ x_2(t) := \sum_{n=1}^{N} A_{2,n}(t) sin[2\pi n\omega_{2,0} t + \theta_{2,n}(t)] + \varepsilon_2(t) \quad \text{sowie} \\ x(t) := \sum_{n=1}^{N} A_n(t) sin[2\pi n\omega_0 t + \theta_n(t)] + \varepsilon(t) \end{array}$$

Das Verhältnis der Grundfrequenzen der beiden Musiktöne sei durch eine natürliche Zahl beschreibbar, also $\omega_{2,0} = i \cdot \omega_{1,0}$ mit einem geeigneten $i \in \mathbb{N}$. Dann kommt es zu einer

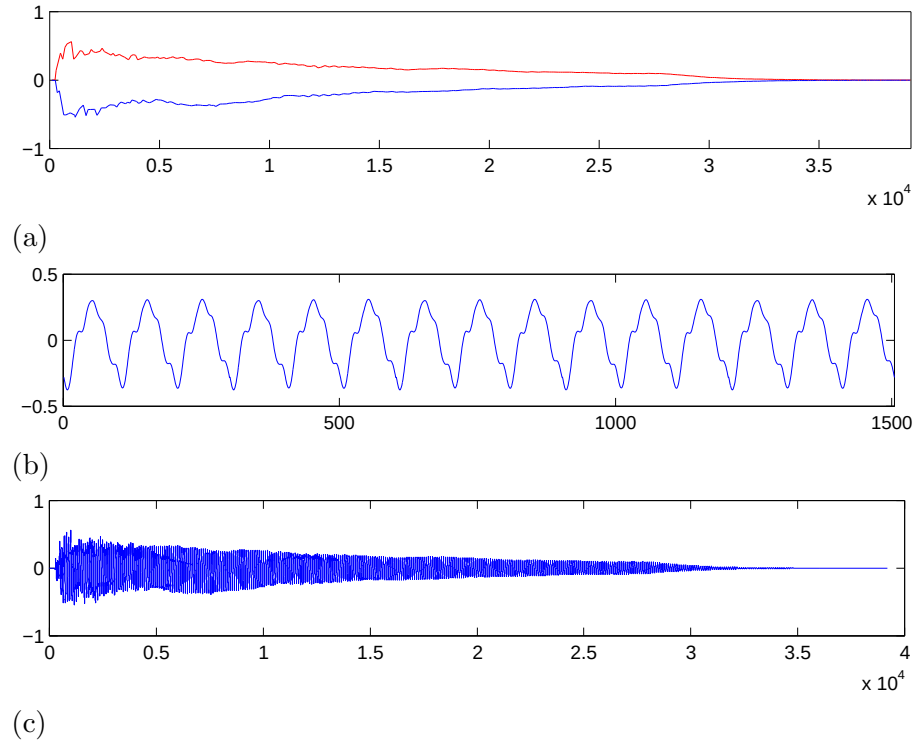


Abbildung 2.4: (a) die Hüllkurve H_x, (b) ein Ausschnitt aus der Farbfunktion F_x, (c) das Signal $x = H_x \cdot F_x$.

Überlagerung der einzelnen Partialtöne und für die resultierenden Amplituden gilt:

$$A_n^2(t) = \begin{cases} A_{1,n}^2(t) + A_{2,n/i}^2(t) + 2A_{1,n}(t)A_{2,n/i}(t)\cos[\theta_{2,n/i}(t) - \theta_{1,n}(t)] & \text{falls} \quad i|n \\ A_{1,n}^2(t) & \text{sonst.} \end{cases}$$

Aus der Formel folgt, dass die Amplitude sehr stark von der Phasendifferenz der Musiktöne abhängt. Dies zeigt, dass der spektrale Gehalt (siehe Kapitel 3) von überlagerten Klängen nicht einfach aus der Addition zweier Spektren gewonnen wird, sondern dass die zeitliche Komponente (hier ausgedrückt durch die Phasen), auch eine wichtige Rolle spielt. Falls man diese außer Betracht lässt, kommt es zu Ungenauigkeiten bei der Tonhöhenextraktion, die sich – wie wir in Kapitel 4 sehen werden – als *Oktavenfehler* manifestieren.

2.5 Diskretisierung und PCM-Format

Erst durch die Aufzeichnung und Speicherung kann Musik sowohl räumlich als auch zeitlich unabhängig reproduziert und damit einem breiten Publikum zugänglich gemacht werden. Die Aufzeichnungstechniken haben sich in den letzten hundert Jahren rasant entwickelt und wurden durch die *Digitalisierung* revolutioniert. Hierbei hat sich insbesondere mit der *PCM-Kodierung* (Pulse-Code-Modulation) eine grundlegende digitale Darstellungsform der Musik als Standard durchgesetzt, auf die wir wegen ihrer Relevanz auch für diese Arbeit im Folgenden kurz eingehen.

Bei der Aufzeichnung wird der Schalldruck eines Audiosignals in eine äquivalente elektrische Spannung transformiert. Bei der anschließenden Digitalisierung durch den A/D-Wandler wird das als elektrische Spannung vorliegende Signal zunächst zwecks Bandbegrenzung gefiltert, dann abgetastet und quantisiert. Das so vorverarbeitete Signal wird dann weiter kodiert, so dass es sich für eine Verarbeitung mittels Rechner oder für die Speicherung auf diversen digitalen Datenträgern (z. B. DAT, CD, MiniDisc) eignet.

Dieses in mehreren Schritten durchgeführte Digitalisierungsverfahren ist verlustbehaftet, d. h. es gehen Informationen verloren und die ursprüngliche akustische Wellenform kann nur noch approximativ aus den digitalen Daten rekonstruiert werden. Meist sind allerdings die Transformationsschritte so gewählt, dass die Informationsverluste nicht zu wahrnehmbaren Rekonstruktionsfehlern führen (siehe 2.6).

Abtastung

Zur Vermeidung sogenannter Aliasingeffekte muß ein kontinuierliches Signal vor der Abtastung bandbegrenzt werden. Bei gängigen Diskretisierungsverfahren wird daher das Ausgangssignal, das der analogen akustischen Wellenform entspricht, durch Filterung mit einem geeigneten Tiefpassfilter passend zum menschlichen Hörempfinden auf $0 - 20000$ Hz bandbegrenzt (siehe Abschnitt 2.6). Dieses bandbegrenzte kontinuierliche Signal, im Folgenden mit $x_c : \mathbb{R} \to \mathbb{C}$ bezeichnet, wird dann durch Abtastung an äquidistanten Zeitpunkten diskretisiert:

$$x(n) := x_c(nT), \quad \text{mit} \quad n \in \mathbb{Z},$$

wobei $T > 0$ die Periode der Abtastung und $x : \mathbb{Z} \to \mathbb{C}$ das resultierende diskretisierte Signal bezeichne.

Die Abtastperiode T bzw. die Abtastrate $\omega := 1/T$ wird durch das *Shannonsche Abtasttheorem* bestimmt. Es besagt, dass für ein stetiges, durch $\Omega > 0$ bandbegrenztes Signal die Abtastrate ω mindestens so groß wie die durch 2Ω definierte *Nyquist-Frequenz* sein muss, um eine vollständige Rekonstruktion des Signals aus diesen Abtastwerten zu garantieren. Theoretisch lässt sich dann das kontinuierliche Signal x_c mit Hilfe der Shannonschen Synthesefunktionen (sinc-Funktionen) durch folgende Interpolationsformel wiedergewinnen:

$$x_c(t) = \sum_{n=-\infty}^{\infty} x\left(\frac{n}{\omega}\right) \frac{sin\pi(\omega t - n)}{\pi(\omega t - n)}.$$

In der Praxis wird eine solche Synthese mittels Tiefpassfilterung mit analogen Filtern realisiert, deren Impulsantwort diese sinc-Synthesefunktionen approximieren (Genaueres über Filterung in Kapitel 3).

Einerseits sollte die Abtastrate zur Vermeidung von Aliasingeffekten oberhalb der Nyquistrate liegen. Andererseits ist auch eine allzu hohe Abtastrate nicht erwünscht, da diese sich sowohl erheblich auf den Speicherbedarf als auch die Berechnungskomplexität auswirkt. In der Praxis hat sich herausgestellt, dass es bei einer Bandbegrenzung von $\Omega = 20$ kHz zu i. A. nicht hörbaren Verlusten bei der A/D-Wandlung kommt, was eine Abtastrate von mindestens $\omega \geq 40$ kHz nahelegt. Standardwerte sind $\omega = 44,1$ kHz und $\omega = 48$ kHz. Die erstgenannte Abtastrate wird bei der Speicherung im CD-Format verwendet, die zweite entspricht der sogenannten *Studioqualität*. Im digitalen Rundfunk wird auch die Abtastrate $\omega = 32$ kHz verwendet.

Quantisierung und Kodierung

Die Abtastung eines kontinuierlichen Signals entspricht einer zeitlichen Diskretisierung. In einem weiteren Schritt müssen nun noch die im allgemeinen kontinuierlichen Abtastwerte diskretisiert werden. Man spricht hierbei auch von *Amplituden-Quantisierung*. Hierbei werden die unendlich vielen möglichen Amplitudenwerte mittels einer endlichen Anzahl von genau vorherbestimmten Quantisierungswerten dargestellt, was allerdings zu zusätzlichen Informationsverlusten führt. Die Größe des dadurch entstehenden Quantisierungsfehlers, auch *Quantisierungsrauschen* genannt, hängt von der Größe der Quantisierungsschritte ab. Diese werden in der Praxis so klein gewählt, dass der Quantisierungsfehler bzw. das sogenannte *Signal-zu-Quantisierungsrauschen*-Verhältnis unter einer vorherbestimmten *Maskierungsschwelle* liegt, die gewährleistet, dass das Quantisierungsrauschen nicht von Menschen wahrnehmbar ist (siehe Abschnitt 2.6).

In einem letzten Schritt wird das so digitalisierte Signal kodiert, wobei bei einer festen Anzahl q von Quantisierungsstufen (bei nicht-adaptiver Quantisierung) jeder Abtastwert mit Wörtern der Länge $\log_2(q)$ kodiert wird. Zum Beispiel werden bei CD-Qualität $q = 2^{16}$ Quantisierungsstufen verwendet, was bei der Kodierung zu Wörtern der Länge 16 Bit führt. Eine höhere Auflösung wird durch 20 und 24-Bit Quantisierung erreicht. Die letztere kommt u. a. beim DVD-Audioformat zum Einsatz und liefert einen dynamischen Bereich von 144 dB. Zusammen mit der Abtastrate von 192 kHz erreicht man so eine sehr hohe Audioqualität (vgl. [63], [66]).

Dieses Verfahren zur Diskretisierung kontinuierlicher Audiosignale ist unter dem Namen *Pulse Code Modulation* (PCM) bekannt. Die so kodierten Signale, auch *PCM-Signale* genannt, stellen in dieser Arbeit eines der grundlegenden Datenformate dar.

2.6 Psychoakustik

Die Musik ist als ein geistiges Produkt des Menschen nicht nur von kulturell bedingten „Einschränkungen“ beeinflusst, sondern ebenfalls sehr stark an die Physiologie des menschlichen Ohres angelehnt. Das menschliche Hörempfinden und die physikalische Struktur von Tönen als Summe von Partialtönen (Obertonreihe) diente durch die Epochen hindurch als Basis u. a. für die Modi und Tonsysteme (z. B. moll, Dur)[1]. Die Auswirkungen erstrecken sich in natürlicher Weise auf das Komponieren, den Instrumentenbau und die Interpretation von Musik, also auf das Musikleben in allen Bereichen.

Die Erkenntnisse über die psychoakustischen Phänomene (siehe Abschnitt 2.6.1) und hier vor allem die absolute Hörschwelle, die kritischen Bänder und die Maskierung werden im Bereich der Audiokodierung intensiv ausgenutzt. In Verbindung mit dem zeitlichen Auflösungsvermögen der menschlichen Hörwahrnehmung (siehe Abschnitt 2.6.2) werden sie hier im Hinblick auf Komposition, Interpretation und Empfinden von Musik betrachtet. Die Hervorhebung von Melodien z. B. durch besondere Dynamik oder zeitlich verschobenen Anschlag, auf die wir bereits in Abschnitt 2.2 eingegangen sind, wird hier in Zusammenhang mit dem Auflösungsvermögen des menschlichen Ohres und der sogenannten Rückwärtsmaskierung (sie-

[1] Die temperierte Stimmung und das aus ihr resultierende chromatische System stellen eine Modifikation der Naturintervallverhältnisse dar.

he Abschnitt 2.6.3) erklärt.

Aus diesem Abschnitt erhalten wir Erkenntnisse über die Interpretation von Musik und variierende Auslegungen von Partituren. Diese liefern uns auch Parametergrößen, die bei der zeitlichen Segmentierung (siehe Kapitel 4) Verwendung finden. Die Frequenzauflösung des menschlichen Gehörs wiederum motiviert ihrerseits die Anwendung der dazu passenden Multiraten-Filterbänke (siehe Kapitel 3).

2.6.1 Psychoakustik unter spektralen Aspekten

Das menschliche Gehör ist von selektiver Natur, da es nur Signale, deren Frequenzen etwa im Bereich vom 20 bis 20000 Hz liegen (vgl. [93]), empfangen kann. Es ist ein System, das aus mehreren Subsystemen besteht. Von diesen ist das Gehirn und dessen Verarbeitung von Audiodaten bis jetzt am wenigsten verstanden. Hier werden wir uns ausschließlich auf das Innenohr und seine Funktion konzentrieren.

Die im Innenohr lokalisierte *Cochlea* (Gehörschnecke) stellt einen mechanisch-elektrischen Wandler des Gehörs dar, der zugleich eine frequenzselektive Funktion ausübt. Die relativen Schwingungen zwischen der Basilar- und der Deckmembran, die auf ihrem Weg von der Ohrmuschel über Außen- und Mittel- bis zum Innenohr hin mehrfach modifiziert werden, werden von den 3500 Haarzellengruppen[2] des Cortischen Organs in Nervenimpulse umgewandelt und weiter durch den Hörnerv zum Gehirn geleitet (vgl. [3]). Die Basilarmembran spannt sich vom ovalen Fenster, das zwischen dem Mittel- und Innenohr liegt, bis zur Schneckenspitze aus (vgl. [58]). Für jede Frequenz gibt es eine bestimmte Position innerhalb der Basilarmembran, an welcher die Haarzellen mit maximaler Amplitude schwingen. Für hohe Frequenzen befinden sich diese Positionen näher am ovalen Fenster und für tiefe am anderen Ende der Membran. Eine derartige frequenzabhängige Unterteilung der Basilarmembran suggeriert, dass diese ungefähr wie ein Frequenzenanalysator funktioniert. Dieser Analysator ähnelt einer aus mehreren Frequenzbändern bestehenden Filterbank.

Wenn zwei Sinusschwingungen gleicher Amplitude und eng benachbarter Frequenzen gleichzeitig erklingen, ergibt sich ein pulsierendes Signal, wie in Abbildung 2.5 (c) dargestellt. Ist die Frequenzdifferenz gering, hören wir aufgrund der beschränkten Frequenzauflösung der Basilarmembran einen vibrierenden, etwas rauen Klang (in der Musiktheorie *dissonanter* Klang genannt). Erst ab einer genügend großen Differenz der Frequenzen werden die beiden Komponenten vom Gehör getrennt. Nahe beieinander liegende Frequenzen stimulieren also – wenn auch in unterschiedlicher Intensität – eine Gruppe von Haarzellen in der Basilarmembran, die zum gleichen *kritischen Band* gehören. Unterscheiden sich die Amplituden der beiden Töne erheblich, tritt der *Maskierungseffekt* ein – der leisere Ton, dessen Lautstärke unterhalb der *Maskierungsschwelle* liegt, wird abgedeckt. Diese Effekte werden sehr systematisch in der Implementierung perzeptueller Audiocodierer ausgenutzt. Ausführlichere Darstellungen des Themas „Maskierung“ und ihre Anwendbarkeit in der Audiocodierung findet man in [93] und [89].

Ein Frequenzband wird *kritisch* genannt, wenn zwei beliebige verschiedene Frequenzen innerhalb seiner Breite aufeinander wirken bzw. einander maskieren können. Die Breite der

[2] Eine Haarzellengruppe besteht aus einer inneren und 3 bis 4 äußeren Haarzellen (vgl. [58]), deren Funktionen noch nicht vollständig geklärt sind (vgl. [3]).

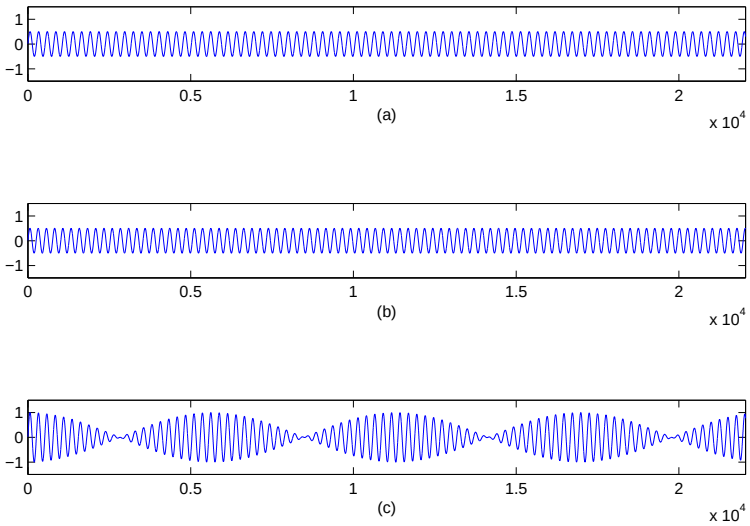


Abbildung 2.5: a) Abschnitt einer Sinusschwingung mit der Frequenz $f_1 = 83.40$ Hz (entspricht der Grundfrequenz von Ton E_3), b) Abschnitt einer Sinusschwingung mit der Frequenz $f_2 = 87.30$ Hz (entspricht der Grundfrequenz von Ton F_3), c) das durch Addition der Signale aus a) und b) resultierende pulsierende Signal.

kritischen Bänder können über folgendes psychoakustisches Phänomen bestimmt werden: Die Lautstärke zweier gleichzeitig klingender Töne, deren Frequenzen nahe beieinander liegen, wird geringer als die Summe ihrer tatsächlichen Lautstärken empfunden. Die Bandbreiten werden dann über die Frequenzbereiche definiert, in denen solche Phänomene auftreten.

Im Modell der sogenannten *Barkskala* wird das Frequenzspektrum in insgesamt 24 kritische Bänder unterteilt, deren Bandbreiten experimentell bestimmt wurden (vgl. [93]). Die kritischen Bänder für die Frequenzen unterhalb von 500 Hz haben eine Breite von ca. 100 Hz. Ab 500 Hz ergibt sich die Breite in etwa aus der Frequenzspanne dreier Halbtöne (vgl. [67]), bzw. aus ca. 20 % der mittleren Frequenz des Bands (vgl. [93]).

Abschließend gehen wir noch auf das Phänomen der *absoluten Hörschwelle* oder der *Ruhehörschwelle* ein. Die absolute Hörschwelle stellt den minimalen, von der Frequenz abhängigen Schalldruck dar, der gerade ausreichend ist, um wahrgenommen zu werden. Der Bereich zwischen 2 – 5 kHz ist derjenige, bei welchem die absolute Hörschwelle ihren kleinsten Wert annimmt, mit anderen Worten ist das Ohr in diesem Bereich am empfindlichsten. Die absolute Hörschwelle hängt stark vom Alter ab und variiert von Person zu Person. Sie dient als Ausgangspunkt zur Bestimmung der Maskierungsschwelle.

2.6.2 Psychoakustik unter zeitlichen Aspekten

In diesem Unterabschnitt werden wir kurz auf das Phänomen der *Asynchronizität* von Klangereignissen eingehen, welches sowohl die Seite des Rezipienten, als auch die des ausführenden Musikers betrifft. Hierbei geht es allerdings nicht um bewusste Grundkonzepte polyphoner Musik (Fugenprinzip, Canon, Klangfarbeneffekte etc.), sondern um relativ kleine zeitliche Differenzen bei mehrstimmiger Musik. Selbige können einerseits unbeabsichtigt und nicht ver-

meidbar sein, andererseits bewusste Mittel zur Betonung bestimmter musikalischer Aspekte (z. B. Verdeutlichung des Themas).

Vor der genauen Auseinandersetzung mit den beiden letztgenannten Aspekten wollen wir kurz einige physiologische Tatsachen darlegen. Experimente mit synthetisierten Tönen haben gezeigt, dass das menschliche Ohr imstande ist, Asynchronismen ab ca. 2 ms zu erkennen. Die Reihenfolge der aufeinander folgenden Ereignisse ist ab einer Zeitverschiebung von ca. 20 ms bemerkbar (vgl. [75], [34]). Weitere Untersuchungen haben gezeigt, dass dieser Schwellwert bei synthetisch erzeugten Musikklängen kleiner ist als bei „reellen“ Musikklängen wie z. B. bei Klaviertönen. Er steigt bei Intervallen von zwei Tönen auf über ca. 30 ms an (vgl. [34]), bei mehr als zwei Tönen sogar auf bis zu 70 ms (vgl. [25]). Weiterhin unterscheidet sich die physikalische Einsatzzeit einer Note von der perzeptuellen, d. h., die vom Menschen wahrgenommene Einsatzzeit ist im Vergleich zum tatsächlichen Beginn des Klangs zeitlich verzögert. Hierbei spielt auch die Anstiegsphase (siehe Abschnitt 2.7) des jeweilgen Klangerzeugers eine Rolle (vgl. [25]).

Wie schon in Abschnitt 2.2 erwähnt, stellen Asynchronismen bei der Ausführung mehrstimmiger Musik eine Alltäglichkeit dar. Dazu gehört z. B. das nichtgleichzeitige Abspielen von Tönen, die laut Partitur gleichzeitig erklingen sollen. Bei Ensembles oder Orchestern tritt dies häufig auf, weil hier mehrere Interpreten agieren, die nicht absolut synchron spielen können, auch wenn der allgemeine Eindruck anders ist[3]. Bezüglich dieser Problematik wurden bereits mehrere Studien durchgeführt. Rasch (vgl. [75]) zeigt, dass Asynchronismen bei kleinen Ensembles von insgesamt drei Instrumenten in einem Intevall von etwa 30 - 50 ms liegen, abhängig von den (Familien der) verwendeten Instrumente. Einer der Gründe mag die Anstiegsphase (siehe Abschnitt 2.7) der Instrumente sein, die z. B. bei den Streichinstrumenten Werte zwischen 50 - 200 ms annehmen kann. Rasch vermutet, dass der Asynchronismus solcher Instrumente, bei denen eine kurze und scharfe Anstiegsphase gegeben ist, stärker wahrgenommen wird. In seinen Untersuchungen stellt er darüber hinaus fest, dass auch das Spieltempo mit der Intensität zeitlicher Verschiebungen korreliert: je schneller das Tempo, desto synchroner die Instrumentalisten. Sehr große Differenzen wurden in Passagen wie z. B. Satzanfängen, Passagen mit sehr langen Noten in einer und sehr kurzen Noten in den anderen Stimmen, in accelerierenden oder retardierenden Momenten beobachtet und allgemein in allen rhythmisch komplexeren Fällen oder in solchen, die eine neue zeitliche Übereinstimmung erfordern. Diese können als nicht beabsichtigte Asynchronismen betrachtet werden.

Im Fall des Klaviers wurden ähnliche Analysen durchgeführt. Auch hier können die Asynchronismen zwei verschiedene Ursachen haben: die motorischen Einschränkungen und/oder Charakteristika des Klaviermechanismus (unbeabsichtigte Asynchronismen) und die bewusste asynchrone Abspaltung einzelner Töne eines Mehrklanges mit der Absicht der besonderen Betonung, z. B. der Explikation einer Melodielinie. Diese *Melodieführung* wurde lange Zeit allein dem vorzeitigen Abspielen jener Töne zugeordnet. Neuere Analysen ergeben aber, dass weitere Aspekte von Bedeutung sind. Goebl (vgl. [33]) folgert auf Grund seiner Messungen, die auch die Zeit zwischen dem Anschlag der Taste und dem Anschlag der Saite durch den Hammer berücksichtigen, dass die Melodieführung maßgeblich durch den Impuls des Tastenanschlags und die damit auftretenden, variierenden Kräfte bestimmt wird. Eine weitere Studie

[3]Das perzeptuelle Empfinden der Hörer ist nicht nur vom Auflösungsvermögen des menschlichen Ohres abhängig; die Aufmerksamkeit des Hörers spielt hier auch eine gewisse Rolle und ist meistens horizontal (Richtung der Melodielinien) anstatt vertikal (Richtung Differenzen der Einsatzzeiten) orientiert (vgl. [75]).

von Goebl und Parncutt (vgl. [34]) mit synthetischen und reellen Klavierklängen zeigt, dass die Reihenfolge asynchron gespielter Töne erst ab 30 ms hörbar ist und dass diese Differenz nicht reicht, um den ersten Ton als den „wichtigeren“ zu erkennen. In diesem Falle kann o. g. Impuls ausschlaggebend für die Erkennung einer Melodie sein.

Für uns sind – zusammenfassend – die Erkenntnisse wichtig, dass Asynchronismen zwischen den Tönen eines Klangereignisses bei reellen Musikaufführungen in der Regel zwischen 30 und 50 ms liegen und dass sie (zumindest unter experimentellen Bedingungen) erst ab 30 ms hörbar sind.

2.6.3 Wechselwirkungen zeitlicher und spektraler Aspekte der Psychoakustik

In diesem Unterabschnitt werden wir auf die Wechselwirkungen der zeitlichen und spektralen Aspekte der Psychoakustik und deren Einflüsse auf Interpretationen eingehen. Erstens stellt sich die Frage, welche Signale einen Maskierungseffekt bewirken können. Ist die Lautstärke das einzige Kriterium? Die Antwort finden wir wieder bei der Physiologie des Ohres. Die Schwingungen in der Basilarmembran verschwinden (räumlich gesehen) relativ direkt hinter der Position, an welcher sie ihre maximale Amplitude erreicht haben (vgl. [54]). Tiefe Frequenzen resonieren am hinteren Teil der Membran, hohe Frequenzen aktivieren die vorderen Bereiche. Hieraus folgt, dass tiefe Frequenzen als Maskierer hoher Frequenzen auftreten können, nicht jedoch umgekehrt. Rasch hat mit seinen Experimenten mit zwei Tönen verschiedener Höhen, Einsatzzeiten und Lautstärken versucht, die Effekte der Maskierung in der Wahrnehmung polyphoner Musik zu erklären (vgl. [74], [76]). Eine wichtige Erkenntnis, die aus diesen Studien folgt, ist, dass für die Wahrnehmung gleichzeitig klingender Töne sowohl die Frequenzen als auch die Einsatzzeiten eine Rolle spielen können.

Bei zwei oder mehreren gleichzeitig gespielten Tönen ist das Resultat ein komplexer Klang, der aus der Überlagerung der einzelnen Partialschwingungen besteht. Die Erkennung bzw. Differenzierung der Tonhöhen ergibt sich in diesem Fall ausschließlich als Resultat der Frequenzanalyse im Hörorgan. Das Ohr analysiert dabei die Verhältnisse zwischen den einzelnen Frequenzen, gruppiert sie basierend auf diesen und zieht daraus Schlussfolgerungen über die Höhen der gespielten Töne. Ein Ton wird maskiert, falls seine Partialschwingungen in der „Maskierungsumgebung“ des anderen Tons liegen und sein Schalldruckpegel um mindestens 20 dB niedriger als der der „potentiellen Maskierer“ ist.

Die spektralen Maskierungseffekte entfallen, falls der Maskierer um wenige Millisekunden verschoben ist. Die notwendige Zeitverschiebung ergibt sich aus der Rückwärtsmaskierung, die am „effektivsten“ in der Zeitspanne von $10 - 15$ ms vor der Einsatzzeit der Maskierer ist (vgl. [74]). Die Länge dieses Intervalls hängt zusätzlich von der Lautstärke der Töne ab. Bei einer Zeitverschiebung von 30 ms ist sogar ein relativer Schwellwert von -60 dB sogar ausreichend. Erklingt also ein leiseres Signal mindestens 30 ms vor einem lauteren, so wird es wahrgenommen, auch wenn sein Schalldruck 60 dB unterhalb des Schalldrucks des folgenden Signals ist, insofern es überhaupt oberhalb der absoluten Hörschwelle liegt.

Die Asynchronismen sind also eine Voraussetzung z. B. für die Trennung der Melodie von der Begleitung und die Verfolgung verschiedener Stimmen. Sie sind aber erst ab einem gewissen Wert (ca. $30-50$ ms) hörbar, was gerade ausreichend ist, um unerwünschte Maskierungseffekte

zu beseitigen und diese von Interpretationsagogik zu trennen. Weiterhin sind die zeitlichen Intervalle beim sequentiellen Spielen zweier Töne (Arpeggien und Glissandi sind hier nicht miteinbezogen) im allgemeinen größer als 50 ms. Dies stellt bei der Parameterwahl für unseren Segmentierungsalgorithmus (Kapitel 4) und für das Clustering von Mehrklängen (Kapitel 5) einen großen Vorteil dar. Wir können also annehmen, dass die minimale zeitliche Distanz zwischen zwei hintereinander gespielten Tönen bzw. Klängen weit über 50 ms liegt und alle Einsatzzeiten, die sich unterhalb dieser Grenze befinden, als gleichzeitig erklingende Menge von Tönen interpretiert werden.

2.7 Charakteristika des Klavierklangs

Wie in Kapitel 1 beschrieben, stellen die *Segmentierung* von Musiksignalen und die *Extraktion* von Tonhöhen wesentliche Teilaufgaben der Synchronisationsaufgabe dar. Für den Entwurf geeigneter Methoden und die Wahl geeigneter Parameter zur Bewältigung dieser Teilaufgaben sind neben den schon besprochenen musiktheoretischen, interpretatorischen und psychoakustischen Aspekten insbesondere auch die physikalischen Charakteristika der vorkommenden Musikklänge von fundamentaler Bedeutung. Diese hängen offensichtlich in starken Maße von den jeweiligen Musikinstrumenten ab. Wir konzentrieren uns in diesem Abschnitt auf den für uns relevanten Klavierklang.

2.7.1 ADSR-Hüllkurve

Die wie in Abschnitt 2.3 definierte Hüllkurve eines Musiktons kann grob in vier Abschnitte unterteilt werden: die *Anstieg-* oder ***A**ttack-Phase*, die *Abkling-* oder ***D**ecay-Phase*, die *Halte-* oder ***S**ustain-Phase* und schließlich die ***R**elease-Phase*. Daher bezeichnet man die Hüllkurve häufig auch als *ADSR-Hüllkurve*.

Im Fall des Klaviers resultiert die Anstiegphase aus dem Tastenanschlag. Während dieser Phase steigt die Lautstärke des Tons bis zu ihrem maximalen Wert an. In der sich anschließenden Abklingphase fällt diese bis zu einem bestimmten Amplitudenwert relativ schnell ab, um in der folgenden Haltephase weitgehend unverändert zu bleiben. Mit dem Loslassen der Taste beginnt die Release-Phase, während der die Lautstärke des Tons bis zum Nullpunkt abklingt. Die vier Phasen einer ADSR-Hüllkurve sind schematisch in Abbildung 2.6 dargestellt.

Dauer und Verlauf der einzelnen Phasen im ADSR-Modell sind stark instrumentenabhängig, bei manchen Musikinstrumenten fallen sogar ganze Phasen völlig weg. So hat z. B. ein Blasinstrument eine kürzere Release-Phase als ein Glockenspiel. Der Verlauf der verschiedenen Phasen in Abhängigkeit vom Instrument wird in Abbildung 2.7 illustriert, in der die ADSR-Hüllkurven des Kammertons a' (A_5 in MIDI-Notation, siehe Abschnitt 2.8) für sechs verschiedene Instrumente dargestellt sind. Dabei wurden zur Verdeutlichung des instrumentenspezifischen Verhaltens der Hüllkurve Instrumente verschiedener Instrumentengruppen ausgewählt.

Die Kenntnis der ADSR-Hüllkurven für spezifische Instrumente (*Instrument Patches*) findet unter anderem bei der elektronischen Synthetisierung von Klängen des jeweiligen Instrumententyps Verwendung – sowohl bei Hardware-Synthesizern wie Keyboards als auch bei den flexibleren Software-Synthesizern (vgl. [49], [77]).

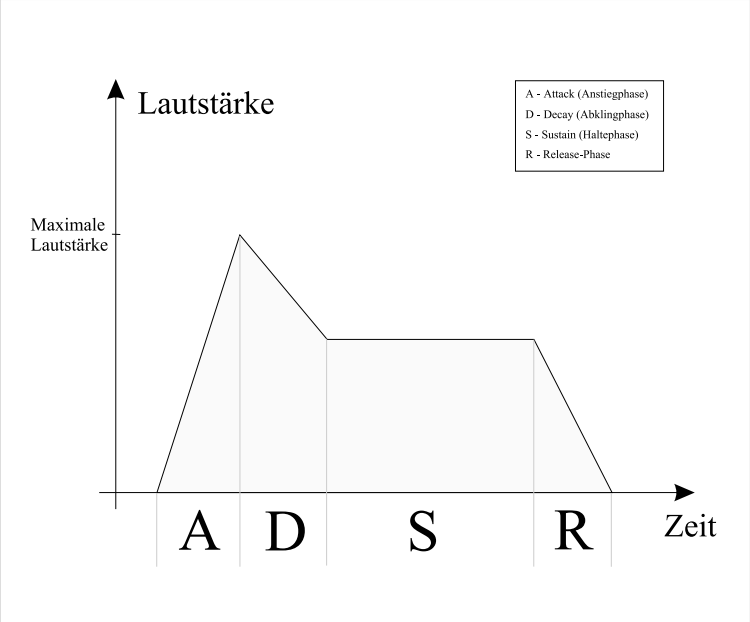


Abbildung 2.6: Typischer Verlauf einer ADSR-Hüllkurve.

2.7.2 Klaviermechanismus

Zum besseren Verständnis des Klavierklangs gehen wir kurz auf die Grundmechanismen dieses Instruments ein.

Die Klaviatur eines handelsüblichen Klaviers besteht aus 88 Tasten, wobei sich der Tonumfang auf etwas über sieben Oktaven (von A_1 bis C_9 in MIDI-Notation) erstreckt[4]. Die Anschlagmechanik des Klaviers ist äußerst komplex und besteht neben der Tastatur u. a. aus den Hämmern und Dämpfern. Schlägt man eine Taste an, so springt der zu ihr gehörige Hammer gegen eine Gleichklanggruppe von ein bis drei Saiten. Gleichzeitig hebt sich der Dämpfer der entsprechenden Taste von den Saiten, so dass diese frei schwingen können. Beim Loslassen der Taste senkt sich der Dämpfer wieder auf die Saiten und hält deren Schwingung an. Die Schwingung der Saiten wird über einen Steg auf den Resonanzboden übertragen und von dort als Schall in die Umgebung abgestrahlt. Während der Einschwingzeit strahlen auch die Saiten und der Steg einen geringen Teil des Gesamtklangs ab. Zu dem Gesamtmechanismus des Klaviers gehöhren auch die drei Pedale, mit denen verschiedene „Lautstärkemodulationsfunktionen“ gesteuert werden können. Weitere Einzelheiten über den Aufbau des Klaviers findet man in z. B. in [26], [4] oder [92].

Sowohl die Farbe als auch die Qualität des Klavierklangs hängen zu einem großen Teil vom Aufbau des jeweiligen Instruments ab. Aber auch der Interpret kann durch seine Spielweisen Einfluss auf die Klangfarbe nehmen (siehe Abschnitte 2.2 und 2.6). Wir beschreiben im Folgenden die für diese Arbeit wichtigsten Charakteristika des Klavierklangs und diskutieren verschiedene Parameter, die den Klang beeinflussen.

[4]Einige große Konzertflügel haben sogar bis zu 97 Tasten.

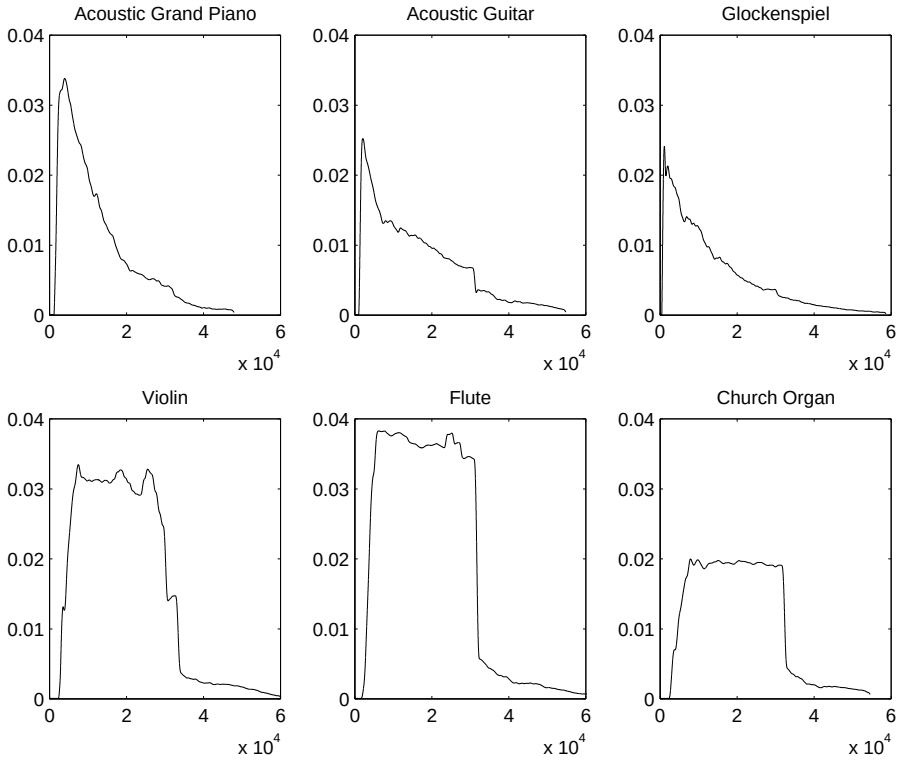


Abbildung 2.7: Die ADSR-Amplitudenhüllkurve des Kammertons a' (MIDI-Bezeichnung A_5), erzeugt von sechs verschiedenen Instrumenten.

2.7.3 Tastenanschlag, Dynamik und Schalldruckpegel

Die Kraft, mit der eine Taste des Klaviers angeschlagen wird, bestimmt die Schnelligkeit mit der der entsprechende Hammer aus seiner Ruheposition die Saiten erreicht und diese anschlägt. Hiervon wiederum hängt offensichtlich die Lautstärke des vom Klavier generierten Musiktons ab. Abhängig von der Kraft des Tastenanschlags wirkt der Hammerfilz auf der Saite unterschiedlich hart. In diesem Fall spricht man von der *dynamischen Härte des Hammers.* Für die Lautstärke ist darüber hinaus auch die *statische Härte* des Hammers von großer Bedeutung, die sich aus der Härte und Beschaffenheit des Hammerfilzes ergibt (vgl. [80], [26]).

Eine Veränderung der Lautstärke wirkt sich – wie sich aus einer spektralen Analyse ergibt – nicht gleichmäßig auf alle Spektralkomponenten aus. Mit anderen Worten ist das Spektrum eines lauten Klangs nicht einfach ein konstant verstärktes Spektrum des entsprechend leiseren Klangs. Bei zunehmender Lautstärke eines Musiktons ändert sich die Energie der unteren Spektralkomponenten nur wenig, während diese in den oberen Spektralkomponenten stark ansteigt. Hinzu kommen neue, höhere Obertöne, die in einer leiseren Variante des Musiktons nicht präsent sind. Somit können die hohen Obertöne als Träger der Lautstärke betrachtet werden. Die erhöhte Präsenz der Obertöne ist auch der Grund, warum laute Töne oft als „heller“ empfunden werden.

Unter der *Dynamik* versteht man den Gesamtumfang des Schalldruckpegels, von dem leisesten

bis zum lautesten spielbaren Ton. Messungen zeigen, dass die Dynamik des Klaviers ca. 30 dB bis 35 dB beträgt (vgl. [26]). Die Dynamik kann unter Verwendung des Dämpferpedals manipuliert und um ca. 1 dB gesenkt werden. Auch in diesem Fall treten gleichzeitig starke Veränderungen in der oben angesprochenen Spektralzusammensetzung auf.

Wir haben gesehen, dass Tastenanschlag, Lautstärke und Spektralzusammensetzung eines Tons in einer komplexen Beziehung stehen. Darüber hinaus hängen diese Aspekte wiederum von der jeweiligen Tonlage ab. Experimentelle Messungen zeigen, dass dieselben Anschlagsstärken bei verschiedenen Tönen zu völlig verschiedenen Schalldruckpegeln führen. Weiterhin stellt sich heraus, dass bei einer Zunahme der Anschlagsstärke der Schalldruckpegel bei verschiedenen Tönen nicht in gleichem Maße zunimmt. Für weitere Einzelheiten und die physikalischen Erklärungen verweisen wir auf [26].

Die Erkenntnisse dieses Abschnitts sind für die Wahl geeigneter Parameter für die Zeitsegmentierung und Tonhöhenextraktion von großer Bedeutung (siehe Kapitel 4). Sie legen nahe, bei der Segmentierung von Musikstücken *zeitadaptive* Schwellwert-Methoden einzusetzen. Zum anderen hängt die Spektralzusammensetzung der Musiktöne von der Dynamik ab, so dass beim Einsatz fest gewählter Schablonen für die einzelnen Musiktöne fehlerhafte Extraktionen insbesondere bei extremer Dynamik zu erwarten sind.

2.7.4 Die Abklingphase

In Abbildung 2.8 sind die in Abschnitt 2.4 definierten ADSR-Hüllkurven für verschiedene Klaviertöne dargestellt. Diese zeigen, dass die Haltephase im Fall des Klaviers nur schwach ausgeprägt ist bzw. völlig ausfällt. Darüber hinaus besteht in diesem Fall die Abklingphase aus zwei „Subphasen“, wobei die Abklingrate der ersten Subphase deutlich größer ist als die der zweiten. Dieses Phänomen lässt sich nach [26] und [92] u. a. wie folgt erklären. Eine einzelne Klaviersaite kann in zwei Richtungen schwingen: vertikal und parallel. Beide Schwingungen fallen exponenziell ab, aber die vertikale Schwingung verklingt schneller als die horizontale, die damit allmählich zur vorherrschenden Schwingung wird. Aus der vertikalen Schwingung entsteht somit der kurzlebige Sofortklang (erste Subphase der Abklingphase), aus der horizontalen der langlebige Nachklang (zweite Subphase der Abklingphase). Weiterhin entnimmt man der Abbildung, dass die Abklingrate der ersten Subphase bei steigender Tonhöhe zunimmt und kürzer wird. Abschließend sei bemerkt, dass die Abklingzeit – abhängig von der Tonlage und der Lautstärke – für das Absinken des Schalldruckpegels um 60 dB ca. 0,2 bis 50 Sekunden beträgt (vgl. [26]).

2.7.5 Inharmonizität

Es ist eine bekannte Tatsache, dass die nacheinander liegenden Oktaven viel angenehmer klingen, wenn sie in einem Verhältnis von knapp über $2:1$ gestimmt werden. Hierfür lassen sich sowohl physikalische als auch hörpsychologische Gründe anführen.

Die physikalischen Gründe liegen in der *Inharmonizität* der Obertöne eines Klaviertons. Die Obertöne liegen im allgemeinen nicht an den erwarteten harmonischen Positionen, sondern weichen von diesen leicht nach oben ab. Die tatsächlichen Positionen der Obertöne können nach [32] durch folgende Formel berechnet werden:

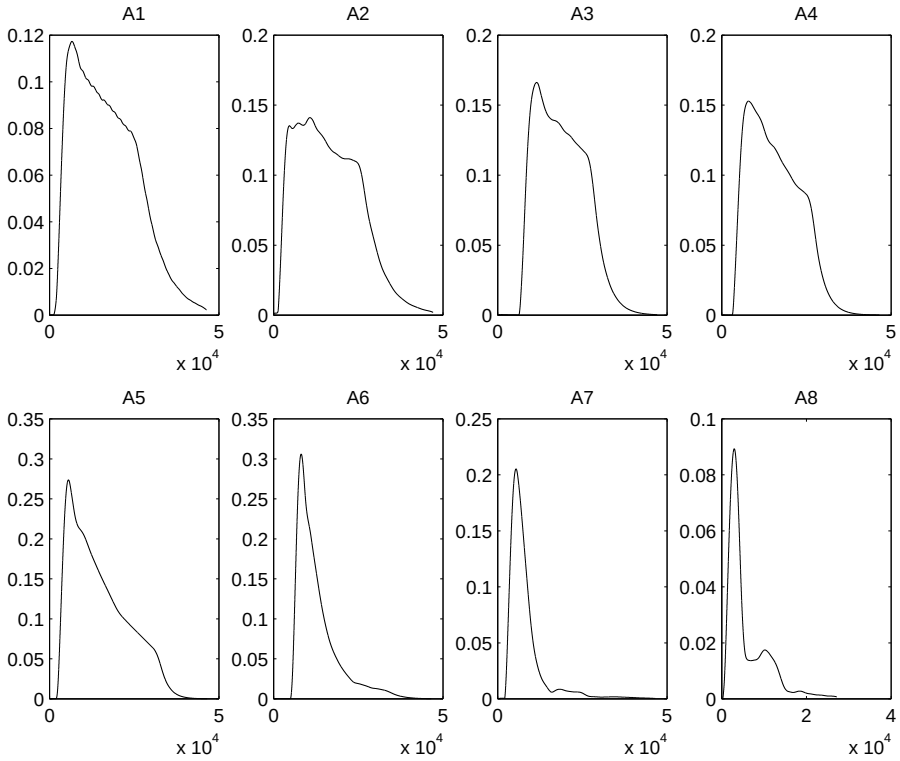


Abbildung 2.8: Die Amplitudenhüllkurven acht verschiedener Klaviertöne.

$$\omega_n = (n+1)\omega_0\sqrt{1+(n+1)^2B}, \tag{2.3}$$

wobei ω_0 die Grundfrequenz des Tons, $n \geq 1$ die Nummer des Obertons und B die Konstante der Inharmonizität bezeichnet. Die Inharmonizität insbesonderer hoher Obertöne ist aufgrund der quadratischen Abhängigkeit von praktischer Relevanz und muss bei der Tonhöhenextraktion berücksichtigt werden.

Weiterhin ist die Inharmonizitätskonstante B im Fall einer nicht umwickelten Klaviersaite proportional zum Durchmesser der Saite und reziprok zur Spannung der Saite sowie zur Saitenlänge [26]. Hieraus folgt, dass die Abweichung der Obertöne für hohe Töne viel größer ist als die für tiefe Töne. Allerdings ist dieses Phänomen für die Praxis nicht relevant, da für die hohen Töne nur die ersten Obertöne hörbar sind, so dass die starken Inharmonizitäten keine perzeptuellen Folgen haben.

Obige Überlegungen wurden durch Messungen mit einem Frequenzanalysator bestätigt. In [4] wurde u. a. untersucht, wie stark die Obertöne des niedrigsten Klaviertons A_1 von den idealen harmonischen Positionen abweichen. Es stellte sich heraus, dass z. B. der 16. Partialton einen Halbton, der 23. über einen Ganzton und der 33. Partialton über zwei Ganztöne über ihrer harmonischen Position liegen.

Neben den physikalischen gibt es auch hörpsychologische Gründe für die anfangs erwähnte „unreine“ Stimmung des Klaviers. Experimente haben gezeigt, dass ein Intervall musikalisch

als eine Oktave empfunden wird, wenn es ca. 10 Cent[5] bzw. 0,6% über dem Verhältnis 2 : 1 liegt (vgl. [26]). Diese Abweichung nimmt mit zunehmender Distanz zwischen den Intervallen zu.

2.7.6 Klangfarbe

Die *Klangfarbe*, auch als *Timbre* bezeichnet, ist diejenige Charakteristik eines Klanges, die ihn von anderen Klängen derselben Tonhöhe und Lautstärke unterscheidet. Die Empfindung der Klangfarbe hängt vom *Klangspektrum* oder vom *spektralen Gehalt* ab, also von Art, Anzahl und Intensität der einzelnen Partialtöne und ihren Amplitudenkurven. Die unterschiedlichen Klangfarben verschiedener Instrumente kommen durch die besondere Schwingungsform des jeweiligen Klangkörpers zustande. Dabei spielen nicht nur die unterschiedlichen Verläufe der Amplitudenkurven der Partialtöne eine Rolle, sondern auch instrumentenabhängige Unterschiede in der Anstiegsphase, der Dauer dieser Phase und der Transienten, die während dieser Phase auftauchen etc. (vgl. [40]). Im Fall des Klaviers werden solche rauschartigen Übergangskomponenten, die im Frequenzbereich zwischen 300 und 3000 Hz liegen, u. a. durch Vibrationen in der Hammermechanik erzeugt (vgl. [26]). Für Töne aus der Basslage liegen solche Komponenten nur 10 bis 20 dB unter dem Pegel des „Hauptklanges“ und sind damit von bedeutender Intensität. Die Übergangskomponenten sind daher bei der Modellierung der Instrumentenklangfarbe z. B. bei Synthesizern sehr wichtig. Es gibt mehrere Studien, die sich mit der Klangfarbe und einer hierauf basierenden Klassifikation von Instrumenten beschäftigen. Wir verweisen für weitere Details z. B. auf die Arbeit [53], bei der es um die automatische Erkennung von Musikinstrumenten geht.

Die oben erwähnten Charakteristika des Klavierklangs zusammen mit den Ergebnissen dieser Studien weisen darauf hin, dass die Klangfarbe von vielen Faktoren wie z. B. dem Instrumentenbau, der Dynamik, der Abspielweise und der Kombination der Töne im Akkord abhängt. Hinzu kommt noch die jeweilige Raumakustik und das komplexe Zusammenwirken all dieser Faktoren. Die Entwicklung von Verfahren zur Tonhöhenextraktion, die gegenüber all diesen Faktoren robust sind, stellt eine enorme Schwierigkeit dar.

2.7.7 Abklingverhalten der Partialtöne

Abschließend wollen wir noch auf ein wichtiges Phänomen eingehen, das in Abbildung 2.9 graphisch dargestellt ist. Hier sieht man deutlich, wie die einzelnen Partialtöne eines Klanges unterschiedliches Abklingverhalten besitzen. Wir stellen anhand dieses Beispiels einige allgemeine Beobachtungen zum Verhalten der verschiedenen Partialtöne zusammen:

(i) Die ADSR-Hüllkurven verschiedener Partialtöne eines Klangs unterscheiden sich wesentlich.

(ii) Der Grundton des Klangs kann bis zu 25 dB unterhalb des Lautstärkepegels der stärksten Spektralkomponente liegen. Er ist in diesem Fall kaum wahrnehmbar und trägt

[5] 1 Cent $\hat{=}$ 1/100 eines Halbtons bzw. 1/1200 der Oktave.

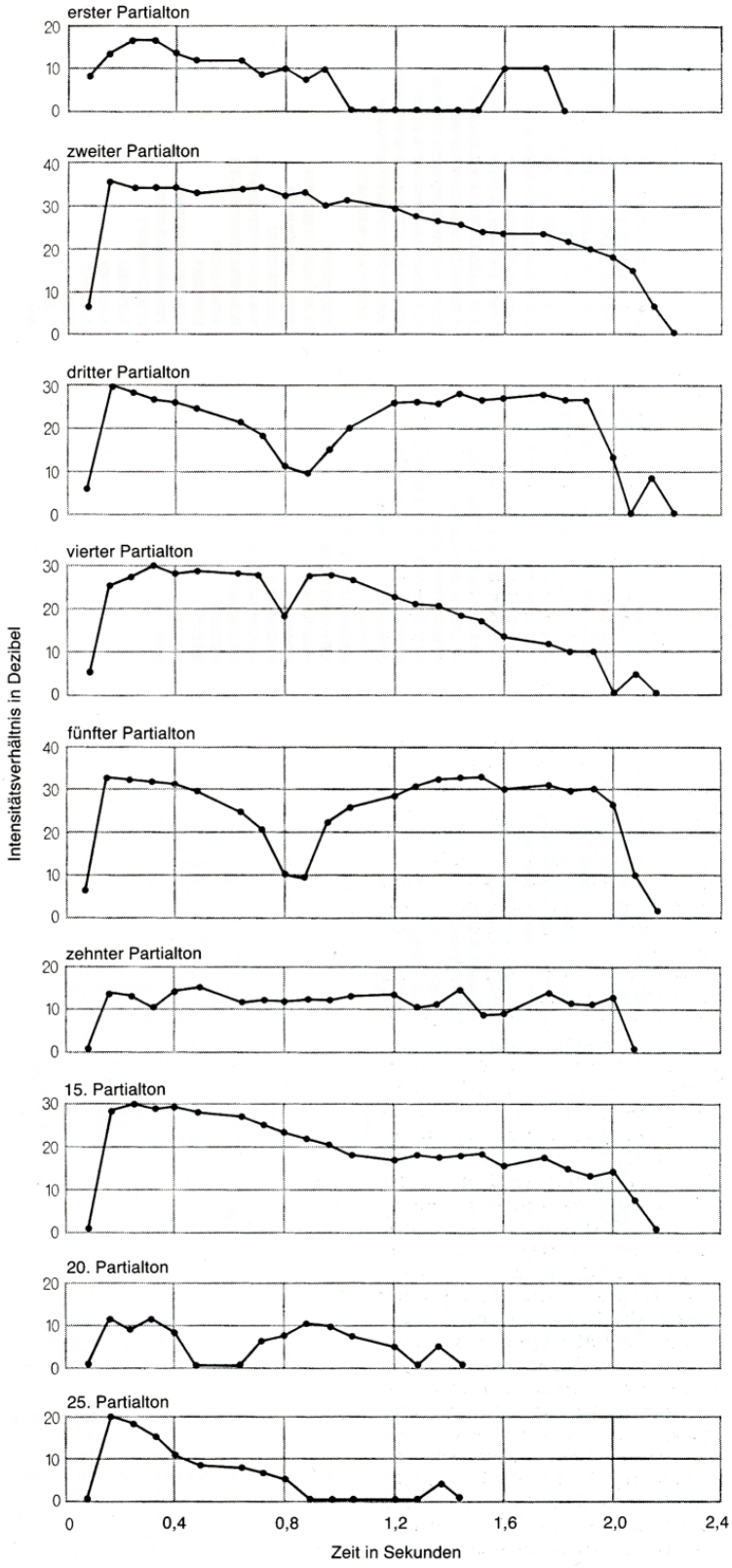


Abbildung 2.9: Abklingkurven für neun Partialtöne des Tons C2, erzeugt durch Messungen im Abstand von 80 ms (Abbildung entnommen aus [4]).

daher nicht wesentlich zur Tonhöhe des Klangs bei. Dieses Phänomen tritt insbesondere bei den Noten der zwei untersten Oktaven auf. Bei höheren Oktaven stellt der Grundton meist die dominante Komponente des Spektrums dar.

(iii) Die höheren Partialtöne klingen schneller ab als die tieferen.

(iv) Beim Klavier klingen im letzten Teil der Abklingphase einige der Partialtöne verhältnismäßig schnell ab und verschwinden fast, um dann plötzlich wieder anzuschwellen bevor sie endgültig ausklingen.

Dieses Beispiel zeigt, welche zeitabhängigen Phänomene im Spektrum auftreten können, wenn ein einzelner Ton z. B. auf dem Klavier gespielt wird. Hinzu kommen noch die oben erwähnten Abhängigkeiten der Spektralzusammensetzung von der Härte des Hammers und der Dynamik. Dies zeigt die Komplexität des spektral-temporalen Musters eines Klangs, selbst schon beim Anschlag einer einzelnen Taste auf dem Klavier. Es ist also nicht zu erwarten, dass ein Klavierklang einheitlich durch spektrale Parameter zu beschreiben wäre.

2.8 MIDI-Format

Wie in der Einleitung dieses Kapitels angedeutet wurde, ist das MIDI-Format zwischen Partitur und Interpretation angesiedelt. Dieser Standard bietet die Möglichkeit, neben bestimmten Partiturinformationen, hierzu gehören insbesondere Tonhöhen, Einsatzzeiten und Tondauern, auch interpretatorische Feinheiten zu erfassen. Darüber hinaus ist die MIDI-Kodierung eines Musikstückes viel kompakter als sein PCM-Pendant. Damit eignet sich dieses Format sowohl für die Aufgabe der inhaltsbasierten Indexierung und des Retrievals in Digitalen Musikdatenbanken (vgl. [14]) als auch für die Kodierung der Extraktionsergebnisse (siehe Kapitel 4) und deren akustische Wiedergabe. Letzteres ist für eine subjektive Bewertung der Extraktionsergebnisse wichtig.

MIDI-Dateien bzw. das MIDI-Format werden in dieser Arbeit an vielen Stellen verwendet. Sie dienen zum einen als Quelle von Zusatzinformationen, die für eine Bereinigung der Extraktionsergebnisse bei ungenauer Schätzung der Einsatzzeiten oder Tonhöhen verwendet werden können. Die extrahierten Notenereignisse können wiederum im MIDI-Format kodiert und akustisch wiedergegeben werden. Schließlich dienen MIDI-Dateien dem Vergleich der Extraktionsergebnisse mit der entsprechenden Partitur(stelle).

Nach einer kurzen Einführung in MIDI gehen wir auf den MIDI-Standard ein und konzentrieren uns auf das Kommunikationsprotokoll und die für uns wichtigsten MIDI-Befehle. Zusätzlich wird der Aufbau der Standard-MIDI-Dateien als eine weitere, sehr wichtige Komponente des MIDI-Standards betrachtet. Am Ende des Abschnitts folgt eine kurze Zusammenfassung, in der wir die Vorteile des MIDI-Formats und seine Anwendbarkeit in dieser Arbeit motivieren.

2.8.1 Einführung in MIDI

MIDI steht für *Musical Instrument Digital Interface*, und wie dieser Name bereits andeutet, wurde es ursprünglich als eine digitale Schnittstelle für Musikinstrumente entworfen. Seit

1982 existiert hierfür ein Standard, der von allen großen Herstellern elektronischer Musikinstrumente akzeptiert wurde (vgl. [59]). Genauer gesagt, ist MIDI ein „Kommunikationsprotokoll“ , das für den unidirektionalen, asynchronen, seriellen Austausch der Daten zwischen elektronischen Musikinstrumenten und anderen MIDI-fähigen Geräten[6] dient. Dies geschieht mit einer Übertragungsrate von 31,25 kBaud (31250 bps). Diese Beschränkung ist durch die Hardware auferlegt. Die Daten sind dabei Befehle für Synthesizer, wie z. B. Informationen über Klangfarbe, Tonhöhe oder Anschlagsstärke. Sie werden von einem *MIDI-Controller* in Echtzeit generiert und zu einem *MIDI-Sound-Generator* weiter geleitet. Dieser Sound-Generator interpretiert die empfangenen Befehle und erzeugt daraus die gewünschten, synthetisierten Klänge. Die *MIDI-Instruktionen* können mittels *MIDI-Sequencer* gespeichert, bearbeitet, miteinander kombiniert und wiedergegeben werden. Da MIDI nur die Sequenz der abzuspielenden Instruktionen spezifiziert und nicht die Klänge selbst, sind MIDI-Dateien in der Regel sehr viel kürzer als die zugehörigen PCM-kodierten Audioaufnahmen desselben Musikstücks, was MIDI besonders attraktiv macht.

2.8.2 MIDI-Standard

Der MIDI-Standard definiert sowohl die Verkabelung der MIDI-fähigen Geräte als auch das Format der Daten, die zwischen diesen ausgetauscht werden. Er kann in insgesamt drei Komponenten unterteilt werden: das *Kommunikationsprotokoll*, die *Hardwareschnittstelle* und die *Standard MIDI-Dateien*. Hier werden wir aber nur auf die Softwarekomponente des MIDI-Standards etwas genauer eingehen.

2.8.2.1 Kommunikationsprotokoll

MIDI definiert sechzehn Kanäle, die für die Datenübertragung genutzt und durch jeweils eine Nummer gekennzeichnet werden. Die Befehlsfolge, die über einen gewünschten Kanal gesendet werden soll, enthält neben den Daten auch die entsprechende Kanalnummer. Über diese Kanäle werden gleichzeitig verschiedene Geräte bedient oder auch verschiedene Stimmen eines polyphonen Musikstücks logisch voneinander getrennt übertragen. An jedem Gerät können die Sende- und Empfangskanäle individuell eingestellt werden. Jedes Gerät beachtet dabei nur die an es selbst adressierten Pakete der Befehle und ignoriert die restlichen.

Ein MIDI-Befehl besteht aus einer Bytesequenz $(b_1, \ldots, b_n)$ mit $n \in \mathbb{N}$, wobei b_1 die Art des Befehls und eventuell die notwendige Zuordnung zu einem MIDI-Kanal enthält und *Statusbyte* genannt wird. Die Bytes $b_2, \ldots, b_n$ bilden die sogenannten *Datenbytes*. Bei einem Statusbyte ist das höchstwertige Bit gesetzt, bei einem Datenbyte hingegen nicht[7]. MIDI-Befehle werden in *System-* und *Instrumentenbefehle* unterteilt. Die Instrumentenbefehle beziehen sich immer auf einen bestimmten Kanal. Systembefehle hingegen sind an keinen Übertragungskanal gebunden und wirken sich auf das gesamte MIDI-System aus.

[6] Zu MIDI-fähigen Geräten zählen u. a. Synthesizer, Keyboard, Expander, Drumcomputer, Effektgeräte, Sequenzer, Sampler und sogar einige spezielle Gitarren (vgl. [49]).

[7] Die variable Befehlslänge erfordert zwar die explizite Unterscheidung zwischen Status- und Datenbytes, ermöglicht aber gleichzeitig eine effizientere Ausnutzung der ohnehin eingeschränkten Kapazität des Datenübertragungskanals.

Zur Gruppe der Systembefehle gehören *Echtzeit-Befehle*, *System-Common-Befehle* und *System-Exclusive-Befehle*. Zu den Instrumenten- oder Kanalbefehlen zählen *Notenbefehle* (*Voice Messages*) und *Steuerbefehle*. Im Folgenden werden wir aber nur auf die für uns interessanten Befehle eingehen.

Notenbefehle

Mit dem *Note-On*-Befehl (s, p, v) wird ein Ton eines angeschlossenen Gerätes eingeschaltet. Hierbei enthält das Statusbyte s die Kennung für den Note-On-Befehl und den Übertragungskanal[8]. Die Tonhöhe wird durch $p \in [0 : 127]$ angegeben. Der Schritt von einer Tonhöhe zur nachfolgenden entspricht einem Halbton. Somit können insgesammt 10.5 Oktaven dargestellt werden. Die Zahl 0 entspricht der Tonhöhe C_0 mit der Grundfrequenz $\omega_0 = 8.17$ Hz und die Zahl 127 der Tonhöhe G_{10} mit der Grundfrequenz $\omega_0 = 12573.85$ Hz. Die Gesamtbandbreite der Tonhöhen überschreitet also deutlich den Tonumfang existierender Musikinstrumente. Die unterste Oktave umfasst sogar vom Menschen nicht hörbare Tonhöhen. Der Tonumfang eines heutigen Klaviers wird mit den Zahlen zwischen 21 (A'' bzw. A_1 in MIDI-Notation) und 108 (c_5 bzw. C_9) kodiert.

Das letzte Byte $v \in [0 : 127]$ dieses Befehls kodiert die Geschwindigkeit, mit der die Taste eines Keyboards angeschlagen wird. Die Interpretation dieses Wertes ist vom Gerät abhängig. Einige Geräte steuern durch ihn die maximale Amplitude des Tons und andere die Bandbreite des Filters bzw. die Helligkeit des Klanges. Der Velocity-Wert $v = 0$ schaltet den entsprechenden Ton aus, sofern einer eingeschaltet war. Es gibt darüber hinaus einen *Note-Off* Befehl zum Ausschalten eines Tones, der ähnlich zum Note-On-Befehl aufgebaut ist. Hierbei wird das dritte Byte jedoch als die Geschwindigkeit, mit der die Taste losgelassen werden soll, also als *Ausschwingdynamik* interpretiert.

Die Notenbefehle stellen also die Grundbefehle über die Tonhöhe, Lautstärke, Einsatzzeit (der Zeitpunkt der Aktivierung eines Note-On Befehls) und Notendauer (das Zeitintervall zwischen Note-On und Note-Off) dar. Über zusätzliche *Steuerbefehle* lassen sich bestimmte Klangeffekte erreichen. Man kann durch sie eine feinere Auflösung der Tonhöhen erreichen (*Pitch Wheel Change*), Pedaleffekte simulieren (*Control Change*) und vieles mehr. Dies ermöglicht u. a., dass der generierte Klang naturgetreuer klingt, was in Kombination mit der Kompaktheit der Befehle und den resultierenden kleinen Dateien die eigentliche Stärke des MIDI-Standards ausmacht.

Systembefehle

Von den drei oben genannten Arten von Systembefehlen erwähnen wir hier nur kurz einige *Echtzeit-System-Befehle* und *System-Common-Befehle*. Die ersten dienen, wie ihr Name andeutet, zur Steuerung der Echtzeitanwendungen bzw. zur Synchronisation aller Systemmodule. Von diesen ist der *Timing-Clock*-Befehl wichtig, der einen regelmäßigen zeitlichen Puls liefert. 24 Timing-Clocks entsprechen einer Viertelnote[9]. Ebenso wie bei Taktschlägen ist die Rate der Timing-Befehle vom Tempo abhängig.

Die Echtzeit-Befehle *Start*, *Stop* und *Continue* werden von Sequenzerprogrammen generiert, um den Start, das Ende bzw. die Fortsetzung einer Wiedergabe anzustoßen.

[8] Auf die Beschreibung des Statusbytes wird im Folgenden verzichtet.

[9] Die zeitliche Auflösung modernerer Sequenzer ist mit 48 bis 480 Sequenzer-Ticks pro Viertelnote wesentlich höher (vgl. [62]).

2.8.2.2 Standard-MIDI-Dateien

MIDI-Befehle werden von Synthesizern in Echtzeit und in ihrer zeitlichen Reihenfolge ausgeführt, ohne dabei explizite Zeitangaben mitzuführen. Wenn aber die abgespielte Sequenz der MIDI-Befehle in eine Datei gespeichert werden soll, müssen die Zeiten explizit angegeben werden. Die Speicherung der MIDI-Daten wurde durch die Spezifikation der Standard-MIDI-Dateien standardisiert, was den Austausch von MIDI-Dateien zwischen verschiedenen Anwendungen ermöglicht.

Die Spezifikation der Standard-MIDI-Dateien definiert drei MIDI-Formate, von denen nur zwei in Sequenzerprogammen Verwendung gefunden haben. *Format 0* speichert alle Ereignisse (MIDI-Befehle) in einer einzigen Spur. Dies ergibt kleinere, kompaktere Dateien. *Format 1* verwendet mehrere Spuren und ermöglicht somit die Trennung der Stimmen in verschiedenen Spuren. Solcher Art aufgeteilten Dateien sind für die Nachbearbeitung mittels Sequenzerprogrammen leichter und anschaulicher zu verwenden.

MIDI-Dateien sind byteweise organisiert und werden zusätzlich in sogenannte *Chunks* unterteilt. Die ersten vier Bytes eines Chunks bestimmen den Chunk-Typ. In den folgenden vier Bytes wird die Länge des Chunks in Bytes angegeben. Es gibt zwei Typen von Chunks: *Header-Chunks* und *Track-Chunks* (Spuren). Der Header-Chunk, mit dem eine MIDI-Datei anfängt, enthält Informationen, die für die gesamte MIDI-Datei gültig sind. Hier wird das Dateiformat, die Spuranzahl sowie die Art der Zeitangabe (Ticks pro Viertelnote) angegeben.

Track-Chunks enthalten im Wesentlichen Noten-Befehle (*Events*) und zusätzliche *Metadaten*, wie z. B. *Tempo-*, *Takt-* und *Tonartangaben*. Letztere werden üblicherweise in der ersten Spur gespeichert. Falls sie sich im Laufe des Stückes ändern, wird für die jeweiligen Zeitpunkte ein entsprechendes `Meta`-Kommando gesetzt. Somit lassen sich insbesondere Temposchwankungen modellieren.

2.8.3 Zusammenfassung

Der MIDI-Standard ermöglicht eine partiturnahe Kodierung der Musikstücke, indem Tempo, Takt- und Tonart sowie Tonhöhen, Einsatzzeiten, Notenlängen (indirekt) und Dynamik explizit angegeben werden. Außerdem lässt sich die Taktunterteilung aus den MIDI-Daten berechnen. Ferner können die einzelnen Stimmen in getrennten Tracks abgelegt werden.

Über Partiturinformationen hinaus lassen sich Interpretationsfeinheiten wie Temposchwankungen und lokale Dynamik der einzelnen Töne explizit kodieren.

Diese Eigenschaften siedeln das MIDI-Format zwischen der Partitur und einer Interpretation eines Musikstücks an und machen es gerade wegen dieser Zwischenstellung und seiner Flexibilität für die Aufgabe der Synchronisation sehr attraktiv.

Von Vorteil ist auch die Anzahl MIDI-kodierter Musikstücke und die Eigenschaft existierender Notationsprogramme (wie z. B. *Capella*, *Sibelius*, *Finale*, *Score*), ihre Dateien als MIDI zu exportieren. Letztere Eigenschaft ermöglicht es, prinzipiell PCM-Dateien mit der Partitur zu synchronisieren, indem zunächst PCM mit MIDI und danach MIDI mit der Partitur synchronisiert wird.

Kapitel 3

Grundlagen der Signalverarbeitung

In diesem Kapitel fassen wir die benötigten signaltheoretischen Grundlagen zusammen und legen damit auch die Bezeichnungskonventionen fest.

Bei der Wellenform eines Musiksignals handelt es sich um eine Darstellung im Zeitbereich, an der man zeitlich verändernde Parameter wie z. B. Dynamik (Lautstärke, Energie) ablesen kann. Allerdings sind in dieser Darstellung spektrale Informationen wie die vorkommenden Tonhöhen oder die Frequenzen versteckt. Auch sind Zeitpunkte von musikalischen Ereignissen, die keine wesentlichen Amplitudenveränderungen bewirken (Hinzukommen leiser Noten), praktisch nicht aus der Wellenform ablesbar oder von Amplitudenmodulationen unterscheidbar. Der erste Schritt beim Extraktionsproblem polyphoner Musik besteht daher meistens in der Transformation des Musiksignals in einen neuen Merkmalsraum, der die quasiperiodische Natur dieser Signale mitberücksichtigt und eine einfachere Extraktion der Parameter erlaubt. (Diese Transformation wird im Folgenden auch als *Front-End-Transformation* bezeichnet).

Hier spielt insbesondere die *gefensterte Fouriertransformation* (Windowed Fourier Transform, WFT) eine wichtige Rolle, die das eindimensionale Musiksignal in einen zweidimensionalen *Zeit-Frequenz-Bereich* oder auch *Phasenraum* überführt. Nachdem wir die Signale und Signalräume sowie einige für uns wichtige Operatoren in Abschnitt 3.1 definiert haben, gehen wir in Abschnitt 3.2 auf Zeit-Frequenz-Analysemethoden ein und stellen dabei einige grundlegende Transformationen vor, und zwar die *kontinuierliche* und die *diskrete Fouriertransformation*, sowie die bereits erwähnte und für uns wichtige *gefensterte Fouriertransformation* (WFT).

Da die WFT eine lineare Unterteilung des Zeit-Frequenz-Bereichs bewirkt, eignet sie sich nur bedingt zur Analyse von harmonischer Musik. Hier sind Filterungsverfahren erwünscht, die das menschliche Hörempfinden mitberücksichtigen und das Musiksignal gemäß der kritischen Bänder zerlegt. Dies kann unter anderem durch den Entwurf geeigneter Multiraten-Filterbänke realisiert werden, deren mathematische Grundlagen wir in Abschnitt 3.3 beschreiben. Schließlich konstruieren wir zwei konkrete Multiraten-Filterbänke, die Grundbausteine für unsere baumartige Multiratenfilterbank sind, die wir in Abschnitt 4.3.1 verwenden.

3.1 Signale, Signalräume und Operatoren

Die von einem oder mehreren Instrumenten oder Sängern generierten Musiksignale, die hier auch als *Audiosignale* oder einfach nur *Signale* bezeichnet werden, sind von ihrer Natur endliche, analoge Signale, denen segmentweise ein quasiperiodisches Verhalten zugeschrieben werden kann. Solche endlichen, analogen Signale werden meist durch den kontinuierlichen Signalraum $L^2(\mathbb{R})$ modelliert. Hierzu definiert man für eine messbare Funktion $f : \mathbb{R} \to \mathbb{C}$ von endlicher Energie zunächst die sogenannte L^2-Norm durch

$$\|f\|_2 := \sqrt{\int_{\mathbb{R}} |f(t)|^2 dt}$$

und dann den Lebesgueraum $L^2(\mathbb{R})$ als

$$L^2(\mathbb{R}) := \{f : \mathbb{R} \to \mathbb{C} \mid f \text{ messbar und } \|f\|_2 < \infty\}.$$

Streng genommen besteht $L^2(\mathbb{R})$ aus Äquivalenzklassen von Funktionen, wobei $f \sim g$ gdw. $\|f - g\|_2 = 0$. Alle messbaren Funktionen mit kompaktem Träger gehören z. B. zu $L^2(\mathbb{R})$, was diesen Raum zur Modellierung von in der Praxis auftretenden Signalen nahelegt. Wir werden hier die Länge $\ell := |\min(\text{supp}(f)) - \max(\text{supp}(f))|$ des Trägers $supp(f)$ eines endlichen Signals häufig auch als *Signallänge* bezeichnen. Winkel und Orthogonalitäten kann man in L^2 mit dem Skalarprodukt

$$\langle f, g\rangle := \int_{\mathbb{R}} f(t)\overline{g(t)}dt$$

messen. Aus signaltheoretischer Sicht erlauben Skalarprodukte, Ähnlichkeiten im Signalgehalt zu messen. So wird bei der später erläuterten Fouriertransformation gemessen, wie groß der Signalanteil an bestimmten Sinus- und Cosinusschwingungen ist. Hier stehen Skalarprodukt und Norm im Übrigen durch $\|f\|_2^2 = \langle f, f\rangle$ in Zusammenhang. Die L^2–Norm eines Signals stellt aus physikalischer Sicht seine Energie dar. Dies ist übrigens auch der Grund dafür, warum man die Signale aus $L^2(\mathbb{R})$ als *Energiesignale* bezeichnet.

In der digitalen Audiosignalverarbeitung haben wir es ausschließlich mit diskreten Signalen zu tun, die durch Abtastung mit einer geeigneten Abtastrate aus den ursprünglichen analogen Signalen gewonnen wurden (siehe 2.5). Diskrete Signale werden mathematisch durch Abbildungen $\mathbb{Z} \to \mathbb{R}$ oder $\mathbb{Z} \to \mathbb{C}$ modelliert. Manchmal werden auch *endliche* Signale $I \to \mathbb{R}$, wobei $I \subset \mathbb{Z}$ ein endliches Intervall ist, benötigt. Zur Modellierung diskreter Signale geht man zu den ℓ^2–Räumen über, die das diskrete Analogon der L^2–Räume darstellen. Für ein Intervall $I \subseteq \mathbb{Z}$ definiert man

$$\ell^2(I) := \{x : I \to \mathbb{C} \mid \sum_{i \in I} |x(i)|^2 < \infty\}.$$

Ein wichtiges Beispiel ist $I = \mathbb{Z}$. Dieser Signalraum ist für unsere Betrachtungen sehr gut geeignet, weil alle endlichen Signale $x : J \to \mathbb{C}$, mit denen wir es in der Praxis zu tun haben, durch Fortsetzung mit Nullwerten im Bereich $\mathbb{Z} \setminus J$ in ihn eingebettet werden können. Diese Annahme erspart uns häufig die explizite Angabe von Definitionsbereichen. Außerdem können wichtige Operatoren sowie Fourierdarstellungen auf diesem Raum definiert werden.

Ein weiterer für uns wichtiger Raum ist der Raum aller absolut summierbaren Signale

$$\ell^1(\mathbb{Z}) := \{x : \mathbb{Z} \to \mathbb{C} \mid \sum_{i\in\mathbb{Z}} |x(i)| < \infty\}.$$

Die Signale aus dem Raum $\ell^2(\mathbb{Z})$ kann man mit $\ell^1(\mathbb{Z})$–Signalen *falten* oder *filtern*:

Satz/Definition 3.1.1 Sind $x \in \ell^2(\mathbb{Z})$ und $f \in \ell^1(\mathbb{Z})$, dann liegt die durch

$$f * x : \quad n \mapsto \sum_{i\in\mathbb{Z}} f(i)x(n-i)$$

definierte *lineare Faltung* oder *lineare Filterung* von x und f in $\ell^2(\mathbb{Z})$. □

Durch Faltung erhalten wir also aus $\ell^2(\mathbb{Z})$–Signalen neue $\ell^2(\mathbb{Z})$–Signale. Zur Beschreibung von Filterbänken werden wir noch einige andere Signaltransformationen einführen.

Definition 3.1.2 Eine Abbildung

$$F : \quad \ell^2(\mathbb{Z}) \to \ell^2(\mathbb{Z})$$

heißt *Operator*. □

Um die Operatoreigenschaft deutlicher zu kennzeichnen, schreiben wir im Falle der Faltung (engl.: *Convolution*) $C_f[x]$ anstatt $f * x$. C_f ist dann das *Operatorsymbol*. Zwei weitere grundlegende Operatoren sind

- punktweise Multiplikation mit $\lambda \in \mathbb{C}$,

  $$\mu_\lambda[x] : \; n \mapsto \lambda x(n)$$

  und

- Translation um $\ell \in \mathbb{Z}$ Abtastwerte

  $$T_\ell[x] : \; n \mapsto x(n+\ell).$$

Sollen mehrere Operatoren hintereinander ausgeführt werden, verwenden wir die übliche Kompositionsschreibweise. So bedeutet $O \circ P$, dass zunächst der Operator P und danach der Operator O ausgeführt wird.

3.2 Zeit–Frequenz–Analyse

Aus der Wellenformdarstellung von Audiosignalen können wir den Amplituden- bzw. Lautstärkeverlauf erkennen und in manchen Fällen an den Stellen, wo die Amplitude besonders abrupt ansteigt, auch die Positionen der Einsatzzeiten lokalisieren. Die Information über die Tonhöhen bleibt aber in der Wellenform versteckt. Um diese herauszufinden, müssen wir das Signal unter Berücksichtigung der Zeitachse im Frequenzbereich untersuchen.

Die Fourieranalysis bildet die Grundlage der klassischen Zeit–Frequenz–Analyse von Signalen. Durch sie wird die Darstellbarkeit von Funktionen des Signalraums $L^2(\mathbb{R})$ als eine Überlagerung von Schwingungen,

$$\hat{f}(\omega) := \mathcal{F}[f](\omega) := \lim_{N\to\infty} \int_{|t|<N} f(t)\exp(-2\pi i\omega t)dt,$$

und

$$f(t) = \int_{-\infty}^{\infty} \hat{f}(\omega)\exp(2\pi i\omega t)d\omega, \tag{3.1}$$

garantiert. Hierbei ist $\hat{f}$ die (kontinuierliche) *Fouriertransformierte von* f und die Darstellung (3.1) von $f \in L^2(\mathbb{R})$ ein nichttriviales Ergebnis der Fourieranalysis. Ist $\hat{f}(\omega) = 0$ für $|\omega| > \Omega$, so heißt das Signal f Ω-*bandbegrenzt*. Bilden $[\Omega_1, \Omega_2]$ und $[-\Omega_2, -\Omega_1]$ den Träger von $\hat{f}$, so heißt $\Omega_2 - \Omega_1$ die *Bandbreite* von f. Eine ausführliche Behandlung der kontinuierlichen Fourieranalysis findet sich in [27].

Ein wesentlicher Nachteil der Fouriertransformation ist, dass sie nur den mittleren Frequenzgehalt des *gesamten* Signals berechnet. Lokalisiert man das Signal vorab durch punktweise Multiplikation mit einer geeignet verschobenen Fensterfunktion um einen Zeitpunkt t, so kann auch der Frequenzgehalt um diesen Zeitpunkt t herum bestimmt werden. Formal definiert man hierzu die gefensterte Fouriertransformation (oder WFT von engl. *Windowed Fourier Transform*[1]).

Definition 3.2.1 Sei $g \in L^2(\mathbb{R})$ mit $\|g\|_2 \neq 0$, dann heißt g *Fensterfunktion* und für $f \in L^2(\mathbb{R})$ heißt die Abbildung

$$W_g[f] : (\omega, t) \mapsto \int_{-\infty}^{\infty} \bar{g}(u-t)f(u)\exp(-2\pi iu\omega)du$$

(g-) *gefensterte Fouriertransformierte* von f. □

Ist die Fensterfunktion g um den Zeitpunkt t lokalisiert, enthält $W_g[f](\omega, t)$ anschaulich gesprochen den Frequenzanteil im Bereich ω zum Zeitpunkt t der Funktion f. Eine in dieser Arbeit verwendete Fensterfunktion mit dieser Eigenschaft ist das Hann-Fenster g_N der Breite N,

$$g_N : \quad t \mapsto \begin{cases} \frac{1}{2}\left(1 - \cos\frac{2\pi t}{N}\right), & \text{falls } t \in [0, N], \\ 0 & \text{sonst.} \end{cases}$$

Diskrete Signale entstehen aus kontinuierlichen durch *Abtastung*. Wie wir bereits im zweiten Kapitel angedeutet haben, impliziert in diesem Zusammenhang das Abtasttheorem nach Shannon, dass bandbegrenzte Signale schon durch ihre Werte an einer diskreten Menge von Punkten darstellbar sind. Eine diskrete Version der WFT erhält man, indem man $W_g[f](\omega, t)$ auf einem diskreten Gitter von Punkten $(\omega, t) \in \mathbb{R}^2$ auswertet. Unter gewissen Anforderungen an dieses Gitter und unter Wahl geeigneter Fensterfunktionen kann dann sogar eine Signalrekonstruktion aus den Analysekoeffizienten durchgeführt werden, d. h., eine verlustfreie Signaldarstellung ist möglich.

[1] Die WFT wird auch als STFT (engl. *Short–Time Fourier Transform*) bezeichnet.

Betrachtet man endliche Intervalle, so kann man die diskrete Fourierentwicklung durch Summation approximieren. Für einen endlichen Signalausschnitt der Länge N, also für $x \in \mathbb{R}^N$ definiert man seine *diskrete Fouriertransformierte* (DFT) in Matrixform durch $D_N x$, wobei $D_N := \frac{1}{\sqrt{N}} \left(W_N^{kn}\right)_{0 \le k,n < N}$ die normalisierte N-reihige *DFT-Matrix* ist und $W_N := \exp\left(-\frac{2\pi i}{N}\right)$ eine primitive N–te Einheitswurzel. Die DFT für einen Signalblock der Länge N kann mittels schneller Algorithmen (FFT-Algorithmen) mit $O(N\log N)$ arithmetischen Operationen effizient ausgewertet werden [13].

Eine Version der diskreten gefensterten Fouriertransformation für endliche Intervalle erhält man in ähnlicher Weise durch die Wahl einer Fensterfunktion $g_N \in \mathbb{R}^N$ und einer Schrittweite $s \in \mathbb{N}$. Die Abbildung

$$w_{g,s}[x] : (k,n) \mapsto \sum_{\ell=0}^{N-1} g_N(\ell) T_{ns}[x](\ell) W_N^{\ell k} \tag{3.2}$$

heißt (g-) *gefensterte diskrete Fouriertransformierte* (WDFT) der Schrittweite s. Als g_N kann man die diskrete Version $g_N : \; n \mapsto \; \frac{1}{2}\left(1 - \; \cos\frac{2\pi n}{N-1}\right)$ des oben angegebenen Hann–Fensters wählen. Die diskrete gefensterte Fouriertransformierte $w_{g,s}[x]$ ist eine komplexwertige Abbildung mit Definitionsbereich $[0 : N-1] \times \mathbb{Z}$. Der Wert $w_{g,s}[x](k,n)$ gibt den Frequenzgehalt von x bezüglich der Frequenz k in einem Intervall der Breite N ab der Stelle ns an. Die zusätzliche Fensterung kostet "nur" weitere $O(N)$ Multiplikationen, d. h. auch die WDFT kann mit Aufwand $O(N\log N)$ approximativ bestimmt werden.

3.3 Kaskadierte Multiratenfilterbänke

Die Zeit–Frequenz–Analyse mit der gefensterten Fouriertransformation liefert eine feste Zeit– und Frequenz–Auflösung. Bei der Analyse von Audiosignalen ist dies jedoch nicht immer angemessen. Hier wäre es i. A. besser, Signalanteile niedriger Frequenzen mit einem breiteren Zeitfenster zu analysieren. Hingegen erfordern Signalanteile höherer Frequenzen eine feinere Zeitauflösung, da plötzliche Änderungen der Schwingungen sonst nicht erkannt werden können. Eine Form der Signalanalyse, die eine hierauf angepasste Zeit–Frequenz–Auflösung erlaubt, ist die *Wavelettransformation* [5]. Anders als bei der WFT, wo für alle Analysefrequenzen der gleiche Signalabschnitt untersucht wird, werden bei der Wavelettransformation verschiedene Frequenzen des Signals mit unterschiedlich skalierten Wavelets analysiert. Diese Transformation ermöglicht also eine ausreichende Auflösung sowohl im Zeit– als auch im Frequenzbereich. Ähnlich wie bei den Varianten der Fouriertransformation gibt es die kontinuierliche und diskrete Wavelettransformation. Aus Sicht der Signalverarbeitung kann man die diskrete Wavelettransformation als *digitale Multiratenfilterung* erklären [88].

Betrachtet man die Fouriertransformierte $D_N h$ eines Filters $h \in \mathbb{C}^N$, so geben die Frequenzkomponenten Aufschluss über die Art des Filters. Ist die Energie von $D_N[h] \downarrow [0 : N/2-1]$ konzentriert auf das Intervall $[0 : K]$, $K < N/2$, so spricht man von einem *Tiefpassfilter* mit einer Abschnittsfrequenz von $\Omega(K) := R \cdot K/N$ Hertz. Hierbei ist R die zugrundeliegende Abtastrate. Ist die Energie von $D_N h \downarrow [0 : N/2-1]$ konzentriert auf das Intervall $[K : N/2-1]$, $K > 0$, so spricht man von einem *Hochpassfilter* mit Abschnittsfrequenz von $\Omega(K)$. Ist die Energie des Bereichs $[0 : N/2-1]$ auf ein Intervall $[K_1, K_2]$, $0 \le K_1 < K_2 \le N/2-1$

konzentriert, so sprechen wir von einem *Bandpassfilter* mit *Durchlassbereich* $[\Omega(K_1), \Omega(K_2)]$.

Bei der Multiratenfilterung wird ein Signal anschaulich mit verschiedenen Faltungsfiltern gefiltert. Dabei deckt jeder Bandpassfilter einen bestimmten Frequenzbereich ab. Da die resultierenden Signale eine geringere Bandbreite vorweisen, kann deren Abtastrate reduziert werden. Dies wird durch folgende Multiratenoperatoren formalisiert:

- Downsampling um $M \in \mathbb{N}_{>0}$ Abtastwerte:

$$(\downarrow M)[x] : \; n \mapsto x(Mn).$$

- Upsampling um $M \in \mathbb{N}_{>0}$ Abtastwerte:

$$(\uparrow M)[x] : \; n \mapsto \begin{cases} x(n/M), & \text{falls } n \text{ Vielfaches von } M \text{ ist,} \\ 0 & \text{sonst.} \end{cases}$$

Für die bisher eingeführten Operatoren gelten die im folgenden Satz zusammengefassten Rechenregeln.

Satz 3.3.1 Seien $h \in \ell^1(\mathbb{Z})$, $\ell, n \in \mathbb{Z}$ und $M, N \in \mathbb{N}$, dann gelten:

(1) $T_\ell \circ C_h = C_h \circ T_\ell = C_{T_\ell[h]}$,

(2) $T_\ell \circ (\downarrow M) = (\downarrow M) \circ T_{M\ell}$,

(3) $T_\ell \circ T_n = T_{\ell+n}$,

(4) $C_h \circ (\downarrow M) = (\downarrow M) \circ C_{(\uparrow M)[h]}$

(5) $(\downarrow M) \circ (\downarrow N) = (\downarrow MN)$,

(6) $(\uparrow M) \circ (\uparrow N) = (\uparrow MN)$.

□

Beweis: Siehe [16, 86, 87, 89]. Aussage (4) ist die bekannte *1. Noble-Identität*.

Die nach Filterung und Abtastratenänderung entstehenden Signale $(\downarrow M) \circ C_h[x]$ heißen *Subbandsignale*. Um ein Signal in verschiedene Subbandsignale zu transformieren, kann entweder eine Filterbank mit vielen verschiedenen Filtern gewählt oder aber eine Kaskadierung von Filterbänken mit wenigen (z. B. jeweils zwei) Filtern vorgenommen werden. Letzteres hat den Vorteil, dass für verschiedene Frequenzbereiche verschiedene Abtastraten und somit verschiedene zeitliche Auflösungen gewählt werden können.

Das Konzept der kaskadierten Multiratenfilter wird nun kurz in Anlehnung an [46] formal eingeführt. Wir werden hier die kaskadierten Filterbänke durch Bäume beschreiben.

Definition 3.3.2 (Syntax eines Filterbankbaums) Ein *Filterbankbaum* (FBT) ist ein endlicher Wurzelbaum $\mathbf{T}$ mit Wurzel k_0, zusammen mit einer Funktion w, die jedem Knoten k von $\mathbf{T}$, $k \neq k_0$ ein Gewicht

$$w(k) := (p_k, M_k, h_k)$$

für positive ganze Zahlen p_k, M_k und $h_k \in \ell^2(\mathbb{Z})$ zuordnet. Hierbei sollen gelten:

- $0 \leq p_k < M_k$ und M_k ist der Ausgrad der Vaterknotens v von k, wobei typischerweise $M_k \geq 2$ gilt (mit Ausnahme der Wurzel k_0),
- für Kinderknoten $\ell \neq k$ von v ist $p_\ell \neq p_k$.

□

Anschaulich ist h_k der p_k-te Filter einer M_k-Band Filterbank. Für den Wurzelknoten definieren wir im Folgenden stets $w(k_0) := (0, 1, \delta)$, wobei $\delta : n \mapsto \delta_{n,0}$ die Kroneckerfunktion ist. Durch FBTs wollen wir kaskadierte Filterbänke beschreiben. Dies geschieht, indem wir jedem FBT eine Semantik in Form von Signalverarbeitungsoperatoren zuweisen.

Definition 3.3.3 (Semantik eines Filterbankbaums) Die Semantik eines FBT $(\mathbf{T}, w)$ ist ein Paar $(\mathbf{T}, W)$, wobei W jedem Knoten $k \in \mathbf{T}$ einen Operator $W(k) : \ell^2(\mathbb{Z}) \to \ell^2(\mathbb{Z})$ zuordnet. Für einen Knoten ℓ mit $w(\ell) = (p, M, h)$ definieren wir

$$\varphi_\ell := (\downarrow M) \circ C_h \circ T_p.$$

Die Verknüpfung dieser Elementaroperatoren entlang des Pfades $k_0 \to k_1 \to \ldots \to k_r = k$ in $\mathbf{T}$ liefert die Semantik

$$W(k) := \varphi_{k_r} \circ \ldots \circ \varphi_{k_0}, \tag{3.3}$$

des Knotens $k \in \mathbf{T}$. Als Konvention setzen wir $W(k_0) := \mathrm{id}_{\ell^2(\mathbb{Z})}$. □

Setzt man $q_0 := 1$ und $q_i := M_0 \cdots M_{i-1}$, so kann man (3.3) mit Hilfe von Satz 3.3.1 umformen in

$$W(k) = (\downarrow q_{r+1}) \circ C_{(\uparrow q_r)h_r} \circ \ldots \circ C_{(\uparrow q_1)h_1} \circ T_{p_r q_r + \ldots + p_1 q_1}. \tag{3.4}$$

Interpretiert man für ein Signal x das Signal $W(k)[x]$ als das Subbandsignal zum Knoten k, so gibt $W(k)$ Aufschluss über die Interpretation dieses Signals. Das Subbandsignal hat eine um q_{r+1} reduzierte Abtastrate und ist entsprechend dem Produktfilter $C_{(\uparrow q_r)h_r} \circ \ldots \circ C_{(\uparrow q_1)h_1}$ gefiltert. Die Translationskomponente $T_{p_r q_r + \ldots + p_1 q_1}$ bestimmt die Verschiebung des Subbands k im Verhältnis zum Eingangssignal. Diese Beziehung wird für die Überlegungen in Kapitel 4 grundlegend sein, wo für eine Signalkomponente des Eingangssignals die entsprechende Komponente innerhalb eines Subbandsignals zu finden ist.

Aufgrund der nach dem Satz von Riesz-Fischer bestehenden Isomorphie, die zwischen $\ell^2(\mathbb{Z})$ und den auf dem Einheitskreis T definierten Funktionen von $L^2(T)$ besteht, kann man sowohl diskrete Signale als auch Operatoren in den Frequenzbereich transformieren (vgl. [27]). Die z-Transformation

$$Z_T : \quad x \mapsto \left(z \mapsto \sum_{k \in \mathbb{Z}} x(k) z^{-k} \right)$$

überführt $\ell^2(\mathbb{Z})$-Signale x in $L^2(T)$-Signale $Z_T[x]$. Ist ζ ein linearer Operator auf $\ell^2(\mathbb{Z})$, dann ist $Z(\zeta) = Z_T \circ \zeta \circ Z_T^{-1}$ ein linearer Operator auf $L^2(T)$. Weiterhin ist die Abbildung $\zeta \mapsto Z(\zeta)$ ein Isomorphismus zwischen Algebren von Operatoren. Insbesondere gelten für lineare Operatoren ζ, ξ des $\ell^2(\mathbb{Z})$

$$Z(\zeta \circ \xi) = Z(\zeta) \circ Z(\xi), \text{ und } Z(\mathrm{id}_{\ell^2(\mathbb{Z})}) = \mathrm{id}_{L^2(T)}.$$

Die obigen $\ell^2(\mathbb{Z})$-Operatoren haben dann folgende Analoga im z-Bereich:

Satz 3.3.4 Sei $x \in \ell^2(\mathbb{Z})$ ein Signal mit z-Transformierter $X = Z_T[x]$, $H(z)$ die z-Transformierte des Filters $h \in \ell^1(\mathbb{Z})$, $M \in \mathbb{N}_{>0}$ und $\ell \in \mathbb{Z}$, dann gelten:

1. $Z((\downarrow M)) = \sum_{k=0}^{M-1} (\downarrow M)_{z,k}$, wobei $((\downarrow M)_{z,k} X)(z) := \frac{1}{M} X\left(z^{\frac{1}{M}} e^{\frac{2\pi i}{M} k}\right)$.
2. $Z(C_f) = \mu_F$, wobei F die z-Transformierte von $f \in \ell^2(\mathbb{Z})$ und $\mu_F : G \mapsto FG$ ist. Insbesondere gilt $Z(T_\ell) = \mu_{z^\ell}$.
3. $Z((\uparrow M)) = (\uparrow M)_z$, wobei $((\uparrow M)_z H)(z) := H(z^M)$.

□

Beweis: Siehe [16, 86, 65, 88].

Der Ausdruck (3.4) für die Semantik eines Subbandoperators im Zeitbereich kann mit Hilfe dieser Resultate umgeformt werden in eine z–Bereich–Semantik

$$W_z(k) = (\downarrow q_{r+1})_z \circ H_r(z^{q_r}) \circ \ldots \circ H_1(z) \circ z^{p_r q_r + \ldots + p_1 q_1}. \tag{3.5}$$

Nimmt man an, dass die Filter h_i Bandpassfilter sind, lässt dieser Ausdruck insbesondere eine Interpretation der Filtercharakteristik (Durchlassbereich) des zum Subband k gehörigen Produktfilters zu.

Im Fall $M = 2$ erhält man einfache Multiratenfilterbänke, die aus ein Analyse- und Synthese-Teil bestehen. Für unsere Anwendung benötigen wir als Grundbaustein nur eine *2-Band Multiraten-Analyse-Filterbank*.

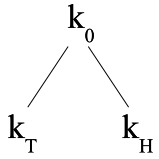


Abbildung 3.1: 2-Band Multiraten-Analyse-Filterbank.

Eine 2-Band Multiraten-Analyse-Filterbank ist ein FBT (siehe Abb. 3.1), wo $w(k_T) = (0, 2, h_0)$ und $w(k_H) = (1, 2, h_1)$ mit einem Tiefpassfilter h_0 und einem Hochpassfilter h_1.

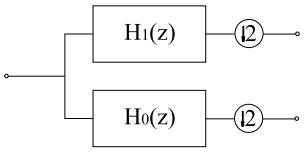


Abbildung 3.2: 2-Band Multiraten-Filterbank – Symbolik aus der Signalverarbeitung.

Erweitern wir diese einfache FB so, dass im Ausgang jedes Subbands eine gleiche 2-Band Multiraten-Analyse-FB folgt und wiederholen wir das gleiche insgesamt $S - 1$ mal, so erhalten wir einen sogenannten *vollständigen 2-Band Filterbankbaum der Höhe S*. In Abbildung 3.3 ist ein vollständiger dreistufiger Filterbankbaum dargestellt. Diese Struktur ermöglicht die

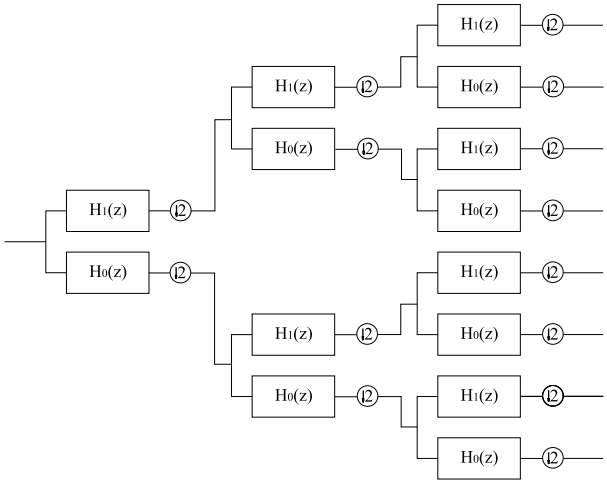


Abbildung 3.3: Vollständiger 2-Band Filterbankbaum der Höhe drei.

Zerlegung des Spektrums in insgesamt 2^S Subbänder der gleichen Bandbreite $\frac{R/2}{2^S}$. Für die eindeutige Trennung der Grundfrequenzen von Klaviertönen ab dem Ton G3 ($\omega_0 = 97.99$ Hz) würden wir bei einer Samplingrate von $R = 22050$ Hz einen vollständigen Filterbankbaum der Höhe $S = 11$ benötigen, was zur Zerlegung des Spektrums in insgesamt 2048 Subbänder führt. Dadurch wird die notwendige Frequenzauflösung für die tieferen Frequenzen erreicht. Gleichzeitig werden aber die hohen Frequenzen mit einer viel größeren Frequenzauflösung als nötig analysiert. Zusätzlich erhalten wir durch die starke Dezimierung in einigen Frequenzbereichen eine für unsere Anwendung untolerierbar geringe Zeitauflösung. Dieses Problem lässt sich durch etwas kompliziertere Filterbankbaumstrukturen lösen, mittels denen die gewünschte Zeit-Frequenz-Auflösung erzielt werden kann. Eine auf unser Problem angepasste Struktur eines Filterbankbaums wird in Unterabschnitt 4.3.1 angegeben.

Kapitel 4

Extraktion musikalischer Merkmale

In diesem Kapitel geht es um die *automatische Extraktion* von musikalisch relevanten Parametern eines Audiosignals, das als Wellenform im PCM-Format gegeben ist. Dabei setzen wir im Folgenden voraus, dass es sich bei dem Audiosignal um ein von einem Klavier eingespieltes polyphones Musikstück handelt. Hierbei sollen zum einen die Einsatzzeiten von musikalischen Ereignissen, d. h. Einsatzzeiten von neu hinzukommenden Tönen, und zum anderen die Tonhöhe dieser Töne bestimmt werden. Wie in der Einleitung dargestellt, kann dieses Problem als eine Erweiterung des *Segmentierungsproblems* und als Teilproblem der *Musiktranskription*, aufgefasst werden. In dieser Arbeit werden die extrahierten Merkmale insbesondere für die Lösung des *Synchronisationsproblems* (siehe Kapitel 5) benötigt. Hier steht also weniger die vollständige und fehlerfreie Extraktion aller Partiturdaten im Vordergrund, sondern es geht vielmehr um die Extraktion einer hinreichend großen Menge an Parametern, die eine Synchronisation einer vorgegebenen Auflösung ermöglicht. In Abschnitt 4.1 gehen wir im Detail auf das Extraktionsproblem ein und beschreiben den Stand der Forschung. In einem ersten Schritt geht es um die Bestimmung von möglichen Einsatzzeiten neu hinzukommender Töne. Hierbei wurden mehrere Verfahren, die grob in Zeitverfahren, Zeit-Energie-Verfahren und Zeit-Spektral-Verfahren eingeteilt werden können, implementiert und getestet (Abschnitt 4.2). Innerhalb der durch die Zeitpunkte definierten Segmente kommen keine neuen Töne hinzu, so dass das Signal in diesen Bereichen als quasiperiodisch angenommen werden kann. In einem zweiten Schritt wird nun das Frequenzspektrum der einzelnen Segmente untersucht. Für jedes einzelne Segment werden mögliche Tonhöhen geschätzt. Abschnitt 4.3 beschreibt ein Verfahren, das die Segmente unter Verwendung eines an die Frequenzcharakteristik des Klaviers angepassten Filterbankbaums in Subbänder zerlegt, aus denen mittels geeigneter Schablonen innerhalb eines iterativen Verfahrens die Tonhöhen abgelesen werden können. Schließlich werden in Abschnitt 4.4 das Gesamtsystem vorgestellt, die durchgeführten Experimente beschrieben und die erzielten Ergebnisse diskutiert.

Bei den hier vorgestellten Verfahren, insbesondere bei den Verfahren zur Tonhöhenextraktion, handelt es sich um off-line Verfahren (Verfahren, die nicht in Echtzeit arbeiten), die jedoch die Kenntnis an Partiturinformation nicht voraussetzen. In einem weiteren Schritt ist der Entwurf effizienterer Algorithmen geplant, die zu on-line Extraktionsverfahren führen. Hierbei soll untersucht werden, inwieweit sich die Partiturdaten als Zusatzinformation (z. B. zur Schätzung von Einsatzzeiten) einsetzen lassen.

4.1 Stand der Forschung

Die Aufgabenstellung der Extraktion und Musiktranskription wurden bereits in Kapitel 1 erläutert. Dabei geht es vor allem um die Erkennung von Notenparametern, wie z. B. Einsatzzeiten, Tonhöhen und Tondauern. Die ersten intensiveren Arbeiten aus dem Bereich der Musiksegmentierung und -transkription datieren aus den siebziger Jahren und benutzten hauptsächlich die Methode des *Pitch-Trackings*. Mittels dieser Methode folgt die Abgrenzung der Töne, d. h. die Bestimmung ihrer Einsatzzeiten und Dauern implizit aus dem Tracking der Grundfrequenzen dieser Töne. Es ging dabei um einstimmige (vgl. z. B. [68]) oder auch zweistimmige Musik [57], allerdings mit der Einschränkung bezüglich der Tonlage und begrenzten möglichen Kombinationen gleichzeitig gespielter Töne. Durch letzteres kann eine Überlappung der Partialtöne vermieden werden. Im Fall der einstimmigen Musik kann man die automatische Musiktranskription als ein in weiten Teilen gelöstes Problem ansehen. Wir werden, um einen Überblick über die verwendeten signaltheoretischen Methoden zu schaffen, kurz einige Verfahren für diesen einfacheren Fall der Extraktion erläutern.

McNab & al. beschreiben in [56] ein prototypisches System zur Melodietranskription von einstimmigem Gesang, das zur Suche in der Meldex–Bibliothek verwendet wird. Für die Tonhöhenerkennung wurde der Gold-Rabiner-Algorithmus, eine Zeitbereichs-Methode zur Bestimmung der Grundfrequenz eingesetzt. Die Autoren berichten von Transkriptionsfehlern von 11.73%. Neben klassischen Signalverarbeitungsmethoden werden in der neueren Zeit für die Segmentierung einstimmiger Musik häufig Hidden-Markov-Modelle (HMM) eingesetzt. Diese sonst in der Sprachsegmentierung häufig angewendete Technik hat insbesondere bei der Aufgabenstellung der automatischen Musikbegleitung Anwendung gefunden. So werden in [9] und [72] zwei Ansätze zur Analyse von Gesang bzw. Oboe vorgestellt. Es ist leider keine quantitative Evaluierung der Extraktionsergebnisse vorhanden.

Im Gegensatz zur einstimmigen Musik geht es bei mehrstimmiger Musik um komplexe PCM-Daten, die typischerweise durch Überlagerung einzelner Töne oder Stimmen verschiedener Quellen (Instrumente, Gesang etc.) entstanden sind. Wie zu erwarten, ist eine Extraktion von Partiturdaten bei polyphoner Musik nur sehr eingeschränkt möglich. Das zeigt auch die folgende Diskussion existierender Verfahren. Fast allen diesen Verfahren ist gemeinsam, dass die Detektion der Einsatzzeiten nur einen ersten Schritt darstellt. Daher werden zunächst einige für uns interessante Zeitsegmentierungsmethoden erwähnt. In [31] werden zwei verschiedene Zeitsegmentierungsansätze vorgestellt – die Amplituden-Thresholding-Methode und die autoregressive Segmentierung. Die erste Methode basiert, wie ihr Name schon andeutet, auf der Detektion der Einsatzzeiten aus der Amplitudenhüllkurve. Hier wird von Problemen mit schwer erkennbaren Einsatzzeiten bei Legato-Noten berichtet, sowie bei der Bestimmung eines geeigneten Schwellwerts, der niedrig genug ist, um alle Einsatzzeiten zu bestimmen, aber wiederum hoch genug, um Fehlschätzungen zu vermeiden. Für die zweite Methode der Zeitsegmentierung wird ein autoregressives Modell verwendet, das auf dem häufig in der Spracherkennung verwendeten LPC-Algorithmus basiert und neue Ereignisse (wie z. B. neu einsetzende Noten) durch den Vergleich der AR-Modelle für je zwei benachbarte Segmente erkennt. Eine modifizierte Variante dieses Algorithmus wurde auch in dieser Arbeit getestet (siehe Unterabschnitt 4.2.1).

Foote stellt in [29] Methoden zur automatischen zeitlichen Segmentierung von Audioaufnahmen vor. Er verwendet dabei einen sogenannten *Novelity Score*, der aus der Korrelation einer

Ähnlichkeitsmatrix mit einer geeigneten Kernfunktion entsteht. Die Dimensionen der Kernfunktion zusammen mit der Breite des Analysefensters bei der hierbei berechneten WFT, bestimmen, wie fein die Segmentierung gemacht werden muss. Aus so erhaltenen Novelity-Score-Kurven bzw. aus ihren Peaks kann man ablesen, wo die Positionen der neu aufgetretenen Ereignisse (Sprache nach Musik, neues Instrument, oder bei kleineren Auflösungen neue Noten bzw. Akkorde) auftreten. Die Methode wird außer in Retrieval-Systemen auch für die Analyse des Rhythmus angewendet [30]. In [28] wird ein Energieprofil bzw. Spektrogramm (ähnlich wie in Abschnitt 4.2.2 bzw. 4.2.3) berechnet, um einen Vergleich zwischen verschiedenen Audioaufnahmen (für den Zweck des Retrievals) mittels dynamischer Programmierung durchzuführen. Die zeitliche Auflösung bei der Erstellung der Energieprofile war dabei eine Sekunde. Sie muss allerdings je nach Anwendung angepasst werden.

Nun wollen wir einige aktuellere Arbeiten erwähnen, die sich mit der Aufgabe der Extraktion von musikalischen Merkmalen bzw. mit der automatischen Musiktranskription von polyphoner Musik beschäftigen. *Scheirer* hat im Rahmen seiner Magisterarbeit [83] ein System zur automatischen Extraktion der Interpretationsschwankungen unter Verwendung der entsprechenden Partitur entwickelt. Solch ein System extrahiert eher die Interpretationsfeinheiten, d. h. leichte Abweichungen von den vorgeschriebenen metronomischen Positionen. Um die Einsatzzeiten der erwarteten Töne zu bestimmen, verwendet er mehrere Methoden. Dabei untersucht er z. B. die Energie der hohen Frequenzen oder die Energie im Ausgang von Kammfiltern für bestimmte Signalabschnitte um die geschätzte Position der Einsatzzeiten herum. Das System ist aber, wie der Autor selbst feststellt, nicht im Stande, Interpretationen, bei denen bedeutende Temposchwankungen (wie z. B. accelerando) auftauchen, zu verfolgen.

Martin hat in [51] und [52] ein System zur automatischen Transkription von einfacher mehrstimmiger Musik präsentiert, das auf einer Blackboard-Architektur basiert. Die Signalanalyse wurde mittels eines sogenannten Log-Lag-Korrelogramms durchgeführt, in der Hoffnung, durch solch einen Ansatz die Notwendigkeit der Instrumentenmodelle bei der Erkennung von Akkorden zu eliminieren. Das System ist fähig, Töne des Intervalls zwischen B_3 und A_5 zu erkennen. Die Experimente mit synthetisierten Klavierstücken, deren maximale Anzahl von Stimmen bis zur vier war, zeigten Probleme bei der Erkennung von Oktaven und lieferte insgesamt, neben den unerkannten, auch falsch geschätzte Noten. Systematische Angaben über die Transkriptionsergebnisse wurden leider nicht bekanntgegeben.

In [42] und [43] haben *Klapuri & al.* ein Verfahren zur automatischen Transkription von Musikaufzeichnungen vorgestellt. Die Einsatzzeiten werden aus den Amplitudenhüllkurven der Subbandsignale bestimmt, genauer aus der numerischen Ableitung der Logarithmen dieser Hüllkurven. Danach folgt die iterative Erkennung von Tonhöhen aus der Klangzusammensetzung. In jedem Iterationsschritt wird die dominante Tonhöhe bestimmt und ihr Spektrum mittels Moving-Average-Methoden geschätzt. Solch ein geschätztes Spektrum wird dann aus dem Gesamtspektrum subtrahiert und die gesamte Prozedur für das residuale Signal wiederholt, bis ein gewisser Schwellwert erreicht wird. Die Testergebnisse mit künstlich erzeugten Klangmischungen, die bis sechs verschiedene Noten und Timbres enthalten können, zeigen, dass zumindest einige der Tonhöhen einer Klangzusammensetzung korrekt erkannt werden. Die Bewertung der Experimente mit reellen Aufzeichnungen wurde hier leider nicht diskutiert. Aus den akustischen Beispielen, die auf der Web-Seite der Autoren angegeben sind [44]), kann man schließen, dass das Verfahren noch nicht „reif“ für eine automatische Transkription solcher komplexen Audioaufzeichnungen ist. Da aber diese Aufgabenstellung so anspruchsvoll

ist, verdient sie ohne Zweifel unsere Aufmerksamkeit.

Bobrek hat in [6], ähnlich wie in dieser Arbeit, für die Signalanalyse und Tonhöhenextraktion Multiratenfilterbänke und Notenschablonen verwendet. Bei der Extraktion hat er aber die Energievektoren bezüglich ihrer maximalen Komponenten normiert, was nach unserer Auffassung einen Kritikpunkt dieser Methode darstellt. Bei der Erkennung von Tonhöhen wird dann die Ähnlichkeit zwischen diesen Energievektoren und den Schablonen aus der Datenbank berechnet. Das globale Maximum bestimmt die erste Tonhöhe, deren Komponenten dann eliminiert werden. Der gesamte Ablauf wird wiederholt, bis es keine wichtigen Energiekomponenten mehr gibt. Die Fehlerrate wird für synthetische Aufnahmen als relativ klein angegeben. Das gilt jedoch nicht für reelle Aufnahmen. Auch hier ist eine systematische Bewertung der Ergebnisse, bis auf einige konkrete Beispiele, nicht vorhanden.

Marlot hat in [50] ein Verfahren zur Transkription mehrstimmiger Klaviermusik mittels neuronaler Netze vorgestellt. Das System erkennt nur die Einsatzzeiten und Tonhöhen. Die Notendauern und Lautstärken werden dabei nicht betrachtet. Für die Zeit-Frequenz-Transformation des Signals wurde hier die Korrelationsfunktion verwendet. Das Spektrum wurde logarithmisch in insgesamt 304 Frequenzbänder unterteilt. Für jede Note wurde ein feed-forward neuronales Netz (NN) trainiert (also insgesamt 88 NN für Noten zwischen A_1 und C_9). Der Autor hat das System mit aus MIDI-Daten stammenden Aufnahmen für verschiedene Klavier-Samples sowie in unterschiedlichen Polyphonien getestet. Er berichtet von Tonhöhendetektionen mit einer Genauigkeit von 94.6%. Die Anzahl der falsch detektierten Noten liegt zwischen 20 und 35% und steigt mit der Anzahl simultan gespielter Noten. Das System kann jedoch die Tonhöhen nur mit einer zeitlichen Verzögerung erkennen. Zudem werden die Einsatzzeiten meist als zu spät geschätzt, was das Verfahren ungeeignet für die Musiktranskription macht. Die Ergebnisse der Tests mit reellen Audioaufzeichnungen wurden in dieser Arbeit nicht erwähnt, außer dass die Notwendigkeit des Trainings mit solchen Aufnahmen als eine Verbesserungsmöglichkeit vorgeschlagen wurde.

Amazing MIDI, ein *PCM-to-MIDI*-Produkt von der japanischen Firma Araki [2], verwendet bei der Tonhöhenerkennung einen einzigen Klang als Ausgangspunkt. Der Vorteil ist, dass dieser „Muster-Klang“ vom Benutzer eingegeben werden kann. Da hier jedoch nur das Spektrum eines Tons die Spektren der Noten der gesamten Tonlage modelliert, ist zu erwarten (und die Testergebnisse bestätigen es), dass neben den korrekt erkannten Tonhöhen auch viele falsche auftauchen.

4.2 Zeit-Segmentierung

Für die Lösung des Extraktions-Problems gibt es mehrere verschiedene Sichtweisen und Ansätze. Einige, wie bereits im Abschnitt 4.1 erklärt, basieren auf Pitch-Tracking-Methoden, d. h. die Segmentierung und Extraktion werden gleichzeitig durchgeführt und basieren ausschließlich auf der „Verfolgung“ der Frequenzkomponenten. In unserem Verfahren stellt die *Zeitsegmentierung* einen ersten Schritt zur Extraktion relevanter Parameter (Einsatzzeiten, Tonhöhen usw.) dar, die für die Aufgabe der Synchronisation (siehe Kapitel 5) notwendig sind. Die Zeitsegmentierung wird durch die Bestimmung der zeitlichen Positionen innerhalb der PCM-Datei, die möglicherweise den Noten-Einsatzzeiten entsprechen, erreicht. Hierzu wurden zunächst Methoden für die Erkennung der Einsatzzeiten entwickelt und getestet, die

nur auf dem PCM-Signal basieren und bei der Einsatzzeitenextraktion keine Partiturvorkentnnisse verwenden.

Nach einer kurzen Zusammenfassung der relevanten physikalischen Phänomene der Anstiegsphase, die für die Detektion der Einsatzzeiten angesetzt werden können, werden die entwickelten Verfahren ausführlich beschrieben. Am Ende dieses Abschnitts werden die Testergebnisse diskutiert und die einzelnen Verfahren zusammen mit den entsprechenden Parameterwahlen bewertet.

Physikalische Phänomene der Anschlags-Phase

Für die zu segmentierenden PCM-Daten wurde der Begriff *Einsatzzeit* als der Zeitpunkt definiert, an dem ein neues Ereignis auftritt, wobei dieses Ereignis der Anstiegsgsphase des ADSR-Modells entspricht (siehe 2.7). Die Phänomene, die diese Phase begleiten, wie z. B. die Transienten, stellen zusammen mit den spektralen Hüllen der Töne die wichtigsten Signalcharakteristika dar, die die Klangfarbe der Instrumente bestimmen. Wir werden bekannte Eigenschaften dieser Phänomene verwenden, um die Anstiegsphase für unsere Algorithmen zur Zeitsegmentierung modellieren zu können. Die meisten der Phänomene wurden bereits ausführlicher in Abschnitt 2.7 beschrieben.

Aus der ADSR-Hüllkurve (Abbildung 2.6) ist ersichtlich, dass während der Anstiegsphase die Energie des Signals von Null bis zu einem maximalen Wert ansteigt, bevor die Energie wieder abklingt. Der Wert dieses Maximums hängt von der Lautstärke ab, die im Fall des Klaviers durch die Härte des Hammers bestimmt wird (siehe Abschnitt 2.7). Dies führt zu dem Schluss, dass die Peaks in der Amplitudenhüllkurve eine der gesuchten Eigenschaften sind, die sich für die Einsatzzeitenextraktion gut eignen. Dies gilt allerdings nur für einen einzelnen, isolierten Ton. Sobald wir es mit mehreren zeitlich benachbarten Tönen verschiedener Längen und Lautstärken zu tun haben, verzerrt sich dieses ideale Bild der Amplitudenhüllkurve derart, dass die tatsächlichen Peaks nicht mehr so einfach zu erkennen und von sogenannten Pseudo-Peaks zu unterscheiden sind, die dann zu Fehldetektionen führen können. Die Gründe dafür sind verschieden:

a) Die Amplitude mancher Partialtöne klingt allmählich ab, um dann vor dem endgültigen Ausklingen wieder anzusteigen, was sich wiederum in Schwingungen der Hüllkurve des Signals widerspiegelt.

b) Ein neubegonnener Ton, dessen Grundfrequenz nahe der des noch erklingenden Tons liegt, resultiert in einer Amplitudenmodulation des Signals. Das Phänomen kommt stärker zum Tragen, wenn die beiden Töne gleichzeitig abgespielt werden (siehe Abschnitt 2.6).

c) Legato abgespielte Töne dauern länger, was zur Überlappung der hintereinander gespielten Töne führt. Dieser Effekt ist noch bedeutender mit abnehmender Länge des Zeitintervalls zwischen den beiden Anschlägen (im Fall von sehr kurzen Noten, Trillern usw.). Die Verwendung des Haltepedals verstärkt diesen Effekt noch deutlicher.

d) Die Lautstärke des nächsten gespielten Tons spielt auch eine wichtige Rolle. Je leiser dieser Ton ist, um so kleiner ist der zugehörige Peak.

Die ersten zwei Punkte deuten an, dass es Schwingungen innerhalb der Hüllkurve gibt, die zu den oben erwähnten Fehldetektionen führen können. Da die Zeitintervalle zwischen solchen Peaks von beträchtlicher Größe sind, funktionieren Verfahren, die unter Zugrundelegung einer minimalen Distanz zwischen zwei Einsatzzeiten fehldetektierte Peaks entfernen, nicht ohne Weiteres. Wird einerseits diese minimale Distanz zu groß gewählt, eliminiert das Verfahren neben den Pseudo-Peaks auch solche, die zu den Noten eines Trillers oder zu sehr schnell gespielten Notenfolgen gehören. Wenn andererseits der Parameter für die minimale Distanz zu klein gewählt wird, werden möglicherweise Pseudo-Peaks nicht als solche erkannt. Die im Punkt (d) angedeutete Dynamik des Musikstücks, die im Fall des Klaviers etwa 30-35 dB erreicht, erschwert die Erkennung der richtigen Einsatzzeiten, so dass die Standard-Schwellwert-Technik nicht ausreichend ist. Statt dessen scheint eine adaptive Methode, die die zeitliche Entwicklung der Signaldynamik berücksichtigt, mehr zu versprechen.

Die Verwendung der Spektralinformation könnte das Verfahren weiter verbessern, da sich nämlich beim Beginn neuer Noten bzw. neuer Akkorde das Spektrum des Signals verändert. Dies geschieht auch, wenn die gleichen Töne wiederholt werden, da die Anstiegsgsphase durch Transienten charakterisiert wird. Dies sind rauschartige Komponenten, die durch den Hammer des Klaviers erzeugt werden, jedoch wieder sehr schnell abklingen. Solch eine Erkenntnis kann bei der Modellierung der Einsatzzeiten sehr hilfreich sein.

Die unten beschriebenen Verfahren nutzen die oben erwähnten Charakteristika teilweise oder ganz. Die Segmentierungsresultate geben Aufschluss über die Relevanz der Phänomene a) – d) bei der Einsatzzeitenmodellierung.

4.2.1 Autoregressive Segmentierung – LPC-Methode

Der LPC-Algorithmus (Linear Predictive Coding) wurde ursprünglich für die Kodierung von Sprachsignalen eingesetzt. Die erzielten guten Ergebnisse haben die Autoren von [31] motiviert, diesen für die Extraktion der Einsatzzeiten aus Audiodaten zu testen. Die Testergebnisse, die für den Fall der Flöte und des Cellos angegeben wurden, sind sehr vielversprechend und haben uns motiviert, denselben Algorithmus für den Fall des Klaviers zu testen.

Die detallierte Beschreibung des LPC-Algorithmus ist unter anderem in [21] oder in [45] zu finden. Hier wird nur die von uns modifizierte Variante des LPC-Algorithmus in Anlehnung an [31] beschrieben. Gegeben seien ein Signal $x : \mathbb{Z} \to \mathbb{R}$ und eine Fensterfunktion $g : \mathbb{Z} \to \mathbb{R}$ der Länge $N := |\min(\mathrm{supp}(g)) - \max(\mathrm{supp}(g))|$. Wir betrachten nun das gefensterte Signal $s = T_{-rN}[g] \cdot x$, wobei $r \in \mathbb{Z}$ ist. Aufgrund der LPC-Analyse ist zu erwarten, dass der aktuelle Abtastwert approximativ aus der Linearkombination der p vorherigen Abtastwerte vorhergesagt werden kann:

$$\hat{s}(n) := -\sum_{k=1}^{p} a(k)s(n-k).$$

Hier sind $a(1), \ldots, a(p)$ die LPC-Koeffizienten, und p steht für die Ordnung des LPC-Modells. Der Prädiktionsfehler, auch *Residualsignal* genannt, ist durch

$$u(n) = s(n) - \hat{s}(n) = s(n) + \sum_{k=1}^{p} a(k)s(n-k),$$

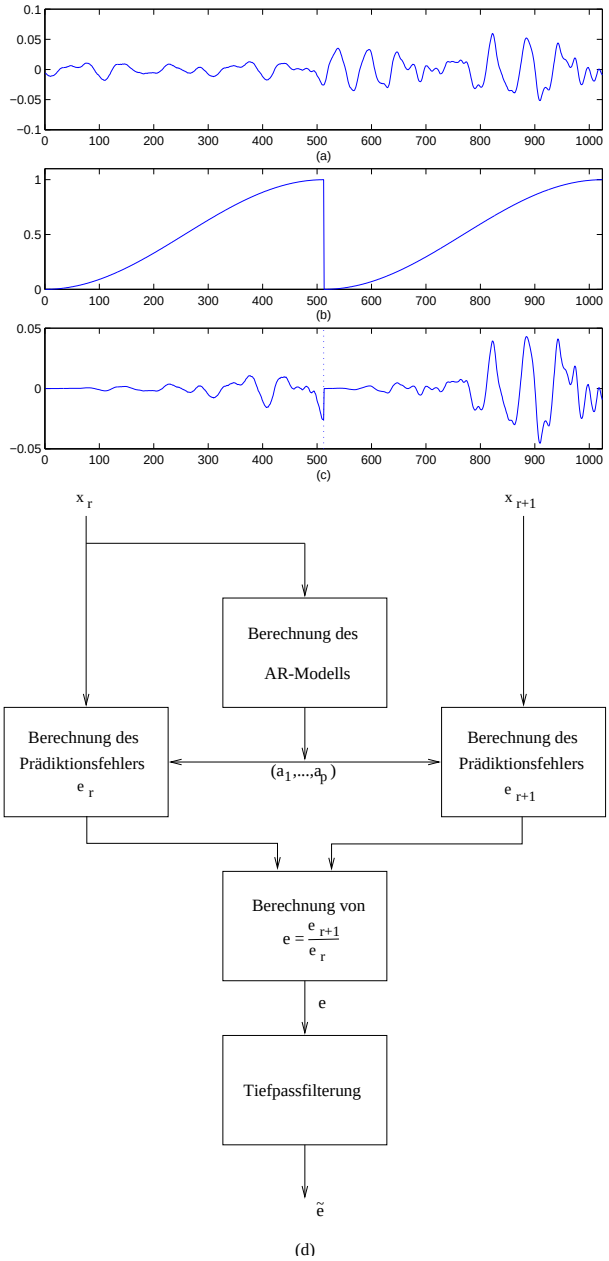


Abbildung 4.1: Schematische Beschreibung des autoregressiven LPC-Algorithmus. (a) Ein Abschnitt des Signals. (b) Die Fensterfunktionen. (c) Das durch die Fensterung modifizierte Signal, getrennt in zwei Segmente x_r und x_{r+1}, die weiter als Eingangsdaten des Algorithmus dienen. (d) Schema des LPC-Algorithmus.

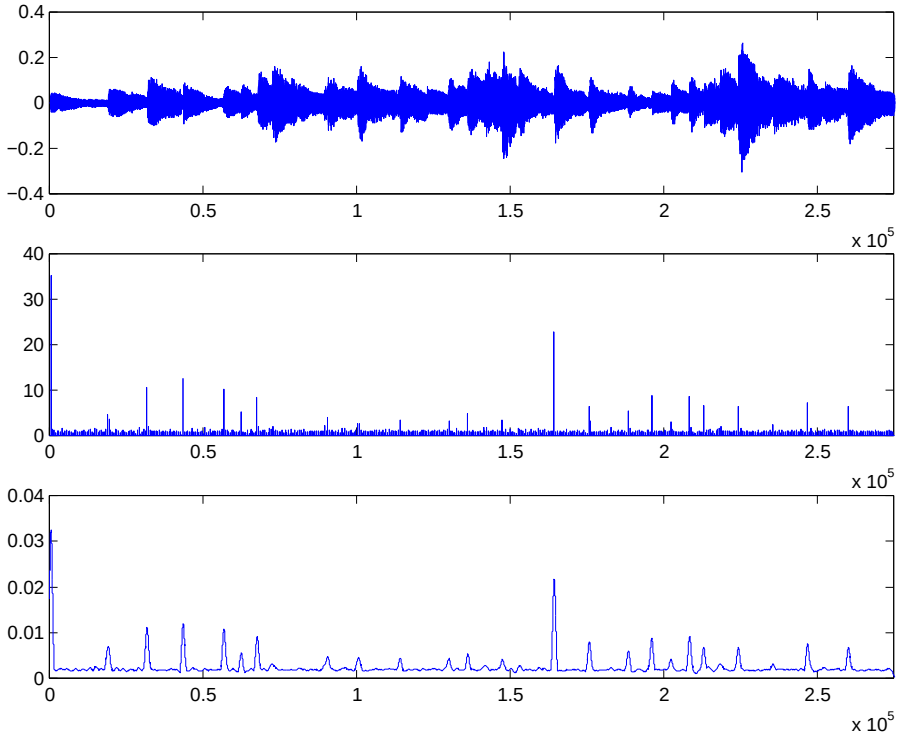


Abbildung 4.2: Extraktion der Einsatzzeiten mit der LPC-Methode (die ersten 12 Sekunden des Trios aus Mozarts Klaviersonate in A-Dur, Op. 331, CD-Aufnahme). Oben: Wellenformdarstellung. Mitte: Fehlerkoeffizienten. Unten: Geglättete Fehlerkoeffizienten. Die Peaks in den zwei untersten Abbildungen zeigen die möglichen Positionen der Einsatzzeiten.

gegeben. Der gesamte quadratische LPC-Fehler im Intervall $[n_1 : n_2]$ ist

$$e = \sum_{n=n_1}^{n_2} u(n)^2.$$

Die LPC-Koeffizienten aus dem analysierten Signal werden so bestimmt, dass der Prädiktionsfehler minimal ist (vgl. [70]).

Das Fenster g wird so gewählt, dass vergangene Werte „vergessen“ werden. Es kann eine exponentiell ansteigende Kurve sein, aber auch ein Cosinus-Fenster, wie z. B. das Hamming- oder Hann-Fenster. In unserem Fall wurde die erste Hälfte des Hamming-Fensters verwendet (siehe Abb. 4.1 (b)).

Der Ablauf des Algorithmus wird anhand der Abbildung 4.1 (d) beschrieben. Aus einem gefensterten Abschnitt des Signals werden die Prädiktionskoeffizienten $(a(1), \ldots, a(p))$ und der Prädiktionsfehler e_1 berechnet. Dieselben Prädiktionskoeffizienten werden verwendet, um die Abtastwerte aus dem benachbarten gefensterten Abschnitt des Signals zu prädizieren. Falls das nachfolgende Segment des Signals deutliche Änderungen im Vergleich zum vorherigen zeigt, ist zu erwarten, dass der Prädiktionsfehler e_2 im Vergleich zu e_1 sehr groß sein wird. Um diesen Effekt zu verstärken, bilden wir $e = e_2/e_1$. Die Prozedur wird für je zwei benachbarte Fenster wiederholt. Schließlich wird die Folge der Fehlerkoeffizienten e durch Tiefpassfilterung

geglättet.

Die Ergebnisse dieses Algorithmus für einen Ausschnitt aus Mozarts Klavier-Sonate in A-Dur, Op. 331, sind in Abb. 4.2 dargestellt. In der Abbildung zeigen die deutlichen Peaks Kandidaten für Stellen möglicher Einsatzzeiten der Noten an. Es wurden aber nicht alle Einsatzzeiten erkannt. Es handelt sich dabei um Stellen, an denen Töne vorkommen, die im Vergleich zu dem noch klingenden Akkord ganz leise abgespielt wurden. Die Erklärung für das nicht Detektieren liegt in dem nicht wesentlich geänderten Frequenzgehalt des Signals. Da dieser Algorithmus genau auf diesen Änderungen basiert, wurden die entsprechenden Stellen auch nicht erkannt. Aus demselben Grund folgt, dass der Algorithmus nicht robust gegenüber Wiederholungen von Noten bzw. Akkorden ist.

4.2.2 Amplitudenbasierte Methode

Ein weiterer Versuch, das Problem der Einsatzzeitendetektion zu lösen, basiert auf der Betrachtung der Signalenergie im Zeitbereich. Es ist bekannt, dass mit jeder gespielten Note bzw. jedem gespielten Akkord die Energie des Signals (lokal) ansteigt. Diese Energie fällt jedoch im Falle des Klaviers gleich nach dem Anschlag wieder ab, da hier die Sustain-Phase des ADSR-Modells nur schwach ausgeprägt ist (siehe Abb. 2.8). Diese Eigenschaft des Klavierklangs kann zum Zweck der Erkennung der Einsatzzeiten wie folgt ausgenutzt werden.

Für das gegebene Signal x und eine Fensterfunktion g der Länge N ist die Energie-Hüllkurve $e : \mathbb{Z} \to \mathbb{R}$, die für die Bestimmung der Einsatzzeiten verwendet wird, für $k \in \mathbb{Z}$ wie folgt definiert:

$$e(k) := \sum_{n \in \mathbb{Z}} |T_{-kN/2}[g_N](n)x(n)|^2. \tag{4.1}$$

Für das Beispiel aus 4.2.1 ist die so berechnete Energie-Hüllkurve in Abbildung 4.3 (c) dargestellt. Man erkennt, dass diese Kurve eine geglättete Variante des oberen (positiven) Teils der eigentlichen Hüllkurve der Wellenform ist (siehe Abb. 4.3 (b) bzw. (a)).

Um die Einsatzzeiten aus dieser Kurve zu extrahieren, muss man die Stellen, an denen die Energie deutlich ansteigt, d. h. e monoton steigend ist, bestimmen. Das sind eben die Stellen, an denen die Anstiegphase der Noten bzw. Akkorde beginnt. Dafür werden wir einen Operator $d : \ell^2(\mathbb{Z}) \to \ell^2(\mathbb{Z})$ einsetzen, der wie folgt definiert ist:

$$d[e] : k \mapsto \begin{cases} e_{k+1} - e_k, & \text{falls} \quad e_k < e_{k+1}, \\ 0 & \text{sonst.} \end{cases}$$

Eine solche Folge $d : \mathbb{Z} \to \mathbb{R}$ ist in Abb. 4.3 (d) zu sehen. Die Peaks der Kurve d liefern neben den richtigen Einsatzzeiten auch Fehldetektionen. Genau wie bei der LPC-Methode werden einige Einsatzzeiten nicht erkannt, da die entsprechenden Töne zu leise gespielt wurden. Daher ist der Energiebeitrag dieser Töne zu gering, um bei dieser Methode Einfluss zu haben. Die Robustheit dieser Methode bei Stücken mit großen Dynamikunterschieden, bei denen die Lautstärke der kurz hintereinander gespielten Töne sehr unterschiedlich ist, entspricht somit nicht unseren Anforderungen.

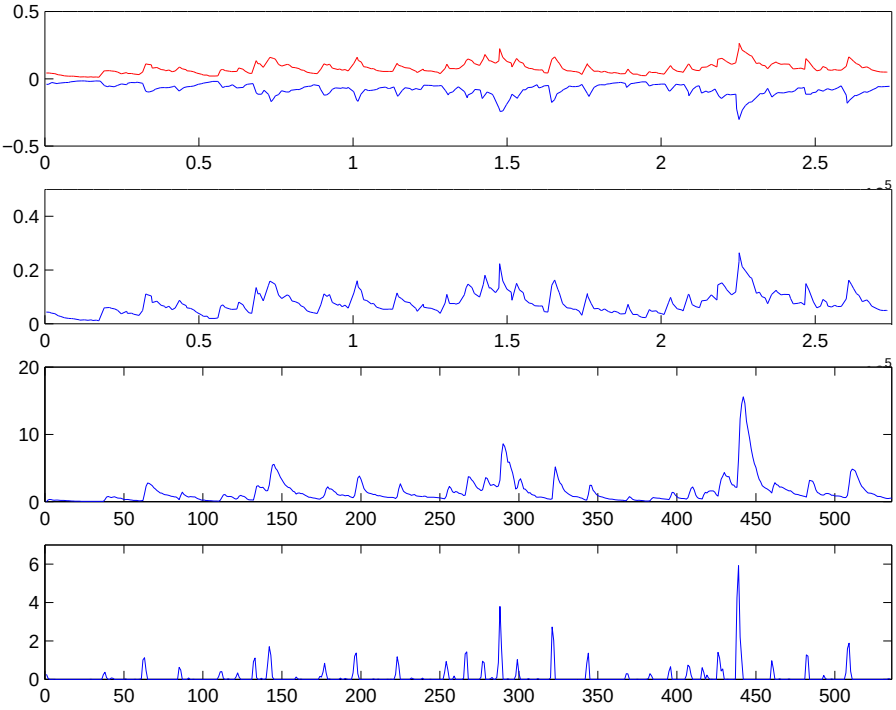


Abbildung 4.3: Extraktion der Einsatzzeiten mit der Energie-Hüllkurven-Methode (Signal aus der Abbildung 4.2). Von oben nach unten: Hüllkurve des Signals, oberer Teil der Hüllkurve, Energiehüllkurve, Folge der positiven Ableitungskoeffizienten.

4.2.3 Die Novelity-Kurven-Methode

Sei $\hat{X} = w_{g,N/2}[x]$ die durch die Formel (3.2) definierte gefensterte Fouriertransformierte des betrachteten Signals x. Eine Folge spektraler Vektoren

$$\tilde{X}(k,n) = \begin{pmatrix} |\hat{X}(k_0,n)| \\ \vdots \\ |\hat{X}(k_{N-1},n)| \end{pmatrix} \in \mathbb{R}^N$$

für aufeinanderfolgende $n \in \mathbb{Z}$ wird Spektrogramm genannt. Ein Spektrogramm stellt die Energieverteilung des Signals in der Zeit-Frequenz-Ebene dar. Das Spektrogramm unseres Beispiels ist in Abbildung 4.4 zu sehen, allerdings wurde genauer $20\log_{10}(\tilde{X}(k,n))$ dargestellt.

Aus dem ℓ^1-Abstand je zweier benachbarter Vektoren erhalten wir die durch

$$\mathcal{T}(n) := \sum_{j=0}^{N-1} |\tilde{X}(k_j, n+1) - \tilde{X}(k_j, n)|$$

definierte Novelity-Kurve $\mathcal{T} : \mathbb{Z} \to \mathbb{R}$. Die resultierende Novelity-Kurve zu Abb. 4.4 ist in Abbildung 4.5 dargestellt.

Wie in Abbildung 4.5 zu sehen ist, tauchen die mit den beiden oben beschriebenen Methoden nicht erkannten Einsatzzeiten (in Position 25, 55, 75 usw.) hier deutlich auf. Die Begründung

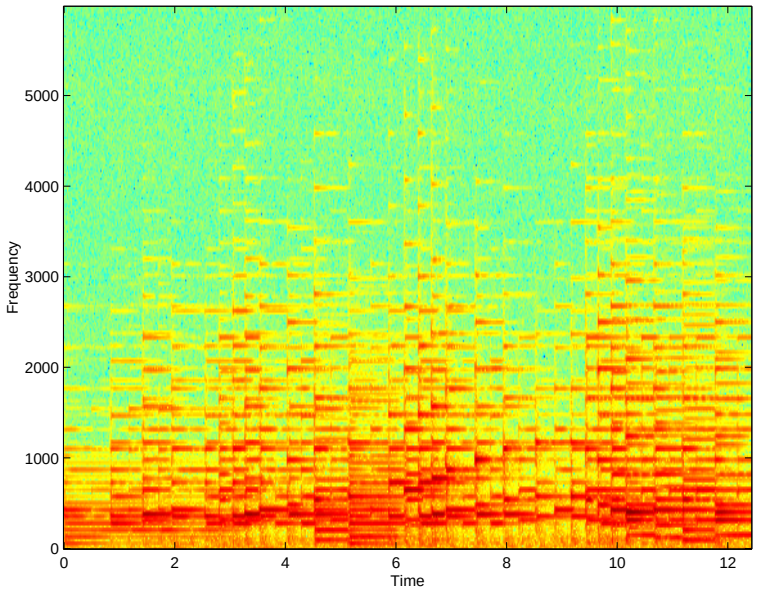


Abbildung 4.4: Das Spektrogramm des Signals aus Abbildung 4.2 (a).

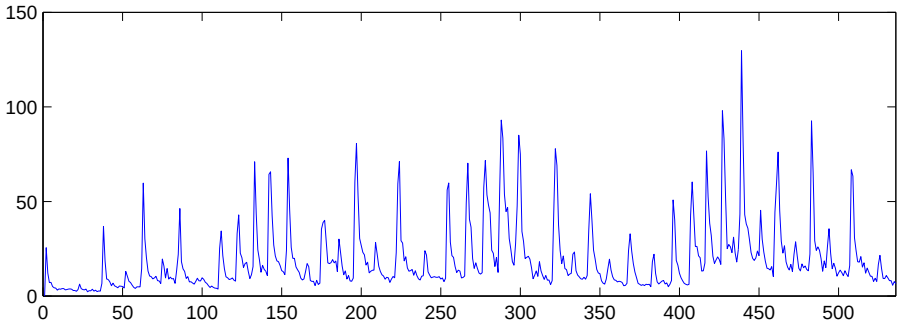


Abbildung 4.5: Novelity-Kurve der Wellenform aus 4.2 (a).

dafür liegt in der Tatsache, dass die Novelity-Kurve ein Maß sowohl der Energie- als auch der Frequenz-Änderungen ist.

4.2.4 Peak-Picking-Algorithmus

Auf die in den Unterabschnitten 4.2.1, 4.2.2 und 4.2.3 eingeführten Kurven e, d und $\mathcal{T}$ soll nun ein Peak-Picking-Algorithmus angewandt werden, um aus den Kandidaten die Einsatzzeiten zu schätzen.

Ein *Peak* wird hier als ein *echtes lokales Maximum* definiert, d. h. ein Peak bildet nicht nur den größten Wert innerhalb eines Fensters, sondern ist auch größer als eine vorgegebene Anzahl seiner Nachbarn. So kann sichergestellt werden, dass keine zwei Peaks, d. h. keine zwei benachbarten Einsatzzeiten, näher als das sogenannte minimale IOI (*inter-onset-interval*) zueinander liegen. Zusätzlich kann die Bedingung gestellt werden, dass der Peak größer als

ein Schwellwert sein muss, um als ein Einsatzzeitenkandidat ausgewählt zu werden. So können einige falsche, rauschartige Peaks eliminiert werden.

Im Folgenden wird dieser Algorithmus generisch für beliebige Kurven angegeben. Dabei bezeichne y eine beliebige der drei Kurven, e, d oder $\mathcal{T}$. Sei $g_q : \mathbb{Z} \to \mathbb{R}$ ein Rechteckfenster, dessen Länge $N_q := |\mathrm{supp}(g_q)|$ ein Parameter des Algorithmus ist. Es wird für die Erzeugung der Schwellwertkurve gebraucht. Die Schwellwertkurve E_{th} zu y ist eine Treppenfunktion, die für $k \in \mathbb{Z}$ im Intervall $[kN_q : (k+1)N_q - 1]$ den g_q-gewichteten Mittelwert

$$\frac{1}{N_q} \sum_{n \in \mathbb{N}} T_{-kN_q}[g_q](n) y(n)$$

besitzt. Für das obige Beispiel ist E_{th} in der Abbildungen 4.8 (a) und (b) mit gestrichelten Linien dargestellt.

Das Kriterium des minimalen IOI wird durch ein weiteres Rechteckfenster $g_{\mathrm{min}} : \mathbb{Z} \to \mathbb{R}$ der Länge $N_{\mathrm{min}} := |\mathrm{supp}(g_{\mathrm{min}})|$ eingesetzt. N_{min} ist hier also das minimale IOI.

Zur Extraktion der Peaks aus $\ell^2(\mathbb{Z})$-Signalen definieren wir einen Operator $P : \ell^2(\mathbb{Z}) \to \ell^2(\mathbb{Z})$. Für $y \in \ell^2(\mathbb{Z})$ ist $P[y]$ dann das Signal

$$r \mapsto \begin{cases} y(r), & \text{falls} \quad y(r) \geq E_{\mathrm{th}}(r) \\ & \qquad \wedge\, y(r) = \max\{T_{-kN_{\mathrm{min}}}[g_{\mathrm{min}}](n) y(n)\} \\ & \qquad \wedge\, \forall t \in [r - N_{\mathrm{min}} + 1 : r - 1] \cup [r + 1 : r + N_{\mathrm{min}} - 1] : y(r) > y(t) \\ 0 & \text{sonst.} \end{cases}$$

Die Abtastrate dieses Signals wird dann mit der des ursprünglichen Audiosignals x durch Upsampling um $N/2$ Werte (Schrittweite) angeglichen. Die Ergebnisse des Peak-Picking-Algorithmus für einige Signale werden im folgenden Abschnitt vorgestellt.

4.2.5 Parameterwahl für den Segmentierungsalgorithmus und Diskussion der Ergebnisse

Von den drei vorgestellten Segmentierungsmethoden ist die LPC-Methode zeitlich die aufwendigste. Da ihre Ergebnisse noch dazu sehr „verrauscht“ sind, was oft zur Detektion vieler falscher Peaks führt, haben wir in unserem Verfahren diese Methode erst im letzten Segmentierungsstadium lokal verwendet, und zwar um eine noch „feinere“ Schätzung der Einsatzzeiten zu erzielen. Die Ergebnisse zweier weiterer Kandidaten – der amplitudenbasierten Methode und der Novelity-Kurven-Methode – werden zunächst anhand unseres Beispiels verglichen. Um die Genauigkeit der Schätzung von Einsatzzeiten zu bewerten, wurden diese aus der Wellenform „von Hand“ bestimmt. Wir bezeichnen die Menge der erwarteten Einsatzzeiten mit $\{e_1, \ldots, e_\upsilon\}$. Aus der Menge der detektierten Peaks P suchen wir diejenigen Positionen p aus, die als „echte Einsatzzeitenkandidaten“ gelten, d. h. wir entfernen aus der Menge $\{p_1, \ldots, p_\kappa\}$ diejenige Elemente, die in der Tat zu keiner echten Einsatzzeit gehören. Die Differenzen der entsprechenden Einsatzzeitenpositionen aus den zwei so erhaltenen Mengen sind in Abbildung 4.6 dargestellt. Die durchgezogene Linie (Rauten) gehört zur Novelity-Kurven-Methode,

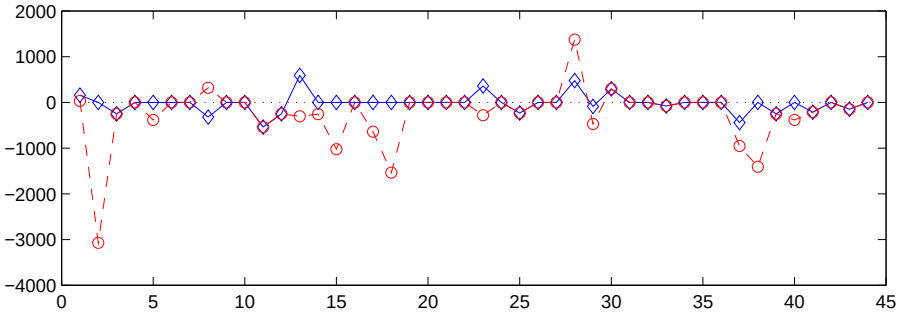


Abbildung 4.6: Abweichung der geschätzten Einsatzzeiten von den erwarteten Positionen. Die durchgezogene Linie entspricht der Novelity-Kurve und die gestrichelte der amplitudenbasierten Methode.

während die gestrichelte Linie die Ergebnisse der amplitudenbasierten Methode wiedergibt. Man bemerkt, dass die amplitudenbasierte Methode ungenauere Schätzungen liefert, wobei eine Standardabweichung von etwa 629 Samples, d. h. 28.5 ms, vorliegt. Bei der zweiten Methode liegt die Standardabweichung bei 199 Samples, d. h. bei ca. 9 ms, was ein deutlich besseres Ergebnis ist. Ähnliche Ergebnisse wurden auch bei anderen Beispielen erzielt, was wahrscheinlich daran liegt, dass die Novelity-Kurve sowohl die Energie- als auch die Frequenzinformationen berücksichtigt und deshalb auch die Einsatzzeiten der leisen oder wiederholten Noten/Akkorde erkennt. Deshalb werden wir ab jetzt nur diese Zeit-Segmentierungsmethode weiter verfolgen.

Nun erläutern wir die Wichtigkeit der richtigen Wahl der Parameter anhand einiger Beispiele. Der erste Parameter – die Länge N des Analysefensters – muss so ausgewählt werden, dass die Ergebnisse so wenig wie möglich durch falsche Einsatzzeiten-Schätzungen „verrauscht“ sind und möglichst alle korrekten Positionen liefern. In Abbildung 4.7 sehen wir von oben nach unten die Novelity-Kurven zu Fensterlängen N von 2048, 1024 und 512 Abtastwerten (entsprechend 92.9 ms, 46.4 ms und 23.2 ms). Man sieht, dass die Länge $N = 2048$ in diesem Beispiel, in dem nur Achtelnoten vorkommen (siehe Abb. 2.1), gerade ausreichend ist. Für kürzere Noten wäre diese Fensterlänge viel zu groß und würde mindestens zwei Einsatzzeiten zusammenfassen. Dies hätte die „Verschmierung“ der Einsatzzeiten und Fehldetektionen zur Folge. An der Abbildung zu $N = 512$ (unten) sieht man, dass eine solche Zeitauflösung viel zu fein wäre und extrem viele falsche Peaks liefern würde. Im allgemeinen Fall hat sich die Fensterlänge $N = 1024$ (Abb. 4.7, mitte) als gute Wahl erwiesen. Deshalb werden wir diese Länge als Standardfensterlänge für unseren Analysealgorithmus festlegen.

Die Wichtigkeit zweier weiterer Parameter des Segmentierungsalgorithmus, N_q (Länge des Fensters zur Berechnung der Schwellwertkurve) und $N_{\min}$ (minimales IOI), wird durch Tabelle 4.1 belegt. Hier sind die Gesamtanzahl der detektierten Einsatzzeiten κ, die Anzahl der falsch geschätzten Einsatzzeiten (f.) sowie die Anzahl der nicht detektierten (n. d.) Einsatzzeiten für verschiedene Kombinationen dieser Parameter angegeben.

Man sieht, wie die Anzahl der geschätzten Einsatzzeiten sich für verschiedene Kombinationen dieser Parameter ändert. Wenn man $N_q = 0$ setzt, d. h. keinen Schwellwert bei der Einsatzzeitendetektion verwendet und dazu die minimale IOI sehr klein wählt (46.4 ms), wird die Anzahl der detektierten Peaks sehr groß ($\kappa = 123$). Zwar werden hier alle $v = 44$ erwarteten

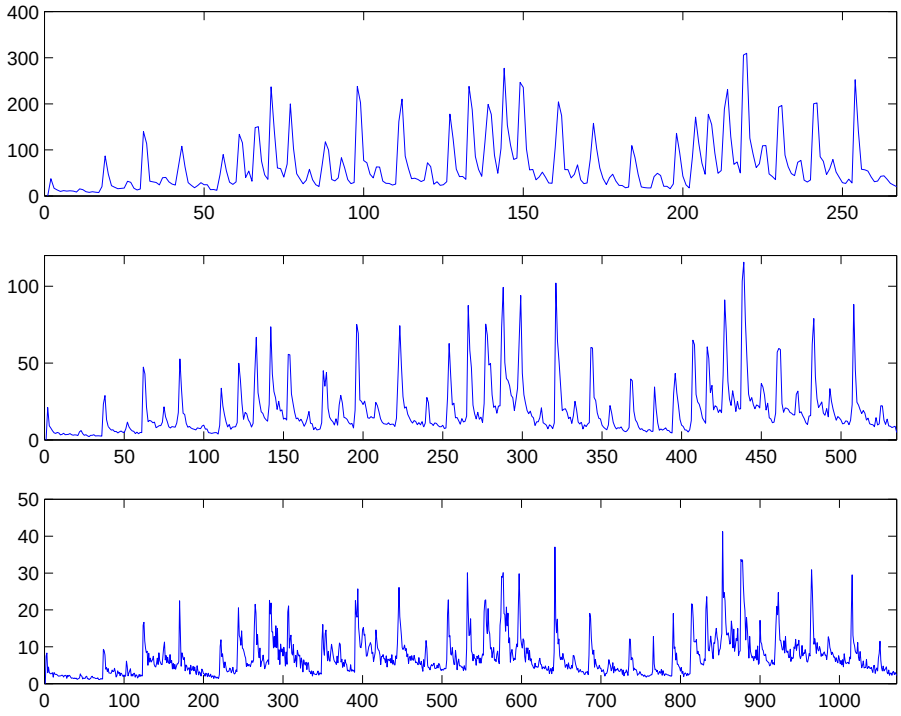


Abbildung 4.7: Novelity-Kurve berechnet mit Fenstern verschiedener Längen: oben $N = 2048$, Mitte $N = 1024$, unten $N = 512$.

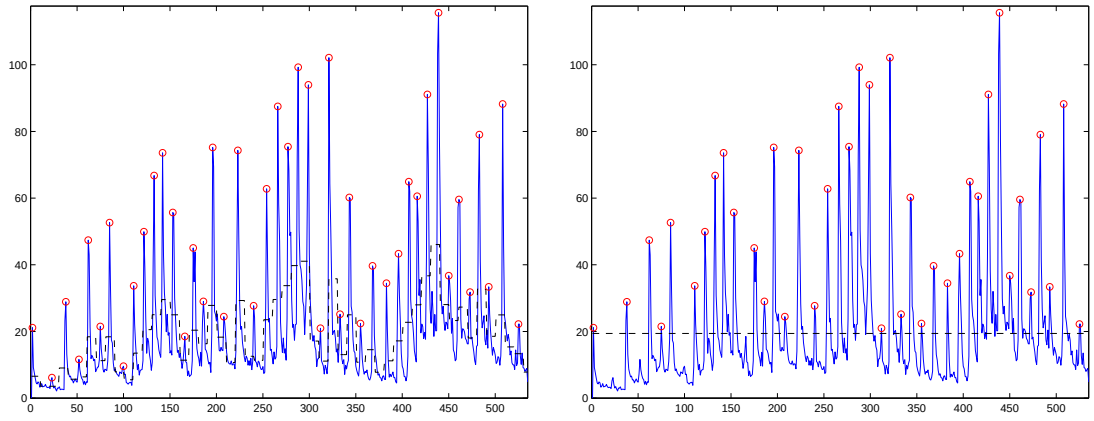
Einsatzzeitenpositionen detektiert, aber die Anzahl der falsch detektierten Einsatzzeitenkandidaten ist fast doppelt so groß wie v. Berechnet man den Schwellwert bezüglich der gesamten Signallänge (hier $N_q = 12.46$ Sekunden), sinkt die Anzahl der Einsatzzeitenkandidaten deutlich auf 60, aber 4 von den erwarteten Einsatzzeiten bleiben dabei nicht erkannt. Die optimale Kombination der Parameter in diesem Beispiel ist $N_q = 0.25$ s und $N_{\min} = 0.116$ s. Sie liefert alle erwarteten 44 Einsatzzeiten ohne eine einzige Fehldetektion (siehe Abb. 4.8 (a)).

Die Wahl der Parameter ist allerdings sehr stark von der Dynamik, der rhythmischen Struktur und dem Tempo des Musikstücks abhängig. So liefern z. B. die oben gewählten „optimalen“ Parameterwerte $N_q = 0.25$ s und $N_{\min} = 0.116$ s für den ca. 16 Sekunden langen Anfangsabschnitt einer Interpretation von Bachs *Aria con Variationi* viel schlechtere Ergebnisse. Es werden in diesem Fall insgesamt 45 Einsatzzeitenkandidaten detektiert, wovon 11 Fehldetektionen sind. Andererseits sind drei der erwarteten Einsatzeiten (Positionen 116, 423 und 497 in der Abbildung 4.9 (a)) gar nicht detektiert worden. Der betrachtete Abschnitt dieses Musikstücks ist durch einige Triller und eine ziemlich enge Dynamikspanne charakterisiert. Deshalb müssen wir bei der Wahl von $N_{\min}$ einen viel kleineren Wert von z. B. 92.9 ms wählen, was etwas kleiner als die Dauer der kürzesten Trillernoten ist. Für N_q können wir dann die Gesamtlänge des Signals wählen. Diese Parameterwahl ermöglicht die Detektion der erwarteten 37 Einsatzzeitenpositionen. Dabei liegen keine Fehldetektionen vor. Das Ergebnis ist in Abbildung 4.9 (b) zu sehen. In Abbildung 4.8 (b) haben wir die Segmentierungsergebnisse für diese Parameterkombination im Beispiel von Mozarts Trio dargestellt.

Die Experimente zeigen, dass die Werte $N_q = 1$ s und $N_{\min} = 116.1$ ms eine Art Kompro-

N_q [s]	$N_{\min}$ 46.4 ms			69.7 ms			92.9 ms			116.1 ms			185.8 ms		
	κ	f.	n.d.	κ	f.	n.d.	κ	f.	n.d.	κ	f.	n.d.	κ	f.	n.d.
0	123	79	0	72	28	0	50	6	0	46	2	0	43	0	1
0.25	66	22	0	53	9	0	46	2	0	44	0	0	43	0	1
1	56	15	3	46	5	3	41	0	3	41	0	3	40	0	4
12.46	60	20	4	44	4	4	40	0	4	40	0	4	39	0	5

Tabelle 4.1: Ergebnisse des Segmentierungsalgorithmus in Abhängigkeit der Parameter N_q und $N_{\min}$ (der Parameter N ist auf 1024 festgelegt). In den Spalten bezeichnet mit „κ“ kann man die Gesamtanzahl der detektierten Einsatzzeitenkandidaten für die entsprechende Kombination der Parameter ablesen, in den mit „f.“ bezeichneten die Anzahl der falsch geschätzten Einsatzzeiten. Die mit „n. d.“ bezeichneten Spalten enthalten die Anzahl der nicht detektierten Einsatzzeiten (Fehldetektionen).



(a) (b)

Abbildung 4.8: Peak-Peaking für Mozarts Trio und Parameterkombination (a) $N_q = 0.25$ s und $N_{\min} = 0.116$ s, (b) $N_q = 13$ s und $N_{\min} = 0.0929$ s.

misslösung für viele Musikstücke darstellen. In unseren hier betrachteten Beispielen erhalten wir im ersten Fall drei fehlende Einsatzzeiten bei Null Fehldetektionen und im zweiten Fall wieder drei fehlende Einsatzzeiten bei zwei Fehldetektionen.

4.3 Extraktion der Tonhöhen

Neben der Bestimmung von Einsatzzeiten von Noten in PCM-Dateien ist die Extraktion der Tonhöhen ein weiterer ganz wesentlicher Bestandteil der Segmentierung von Musiksignalen für Synchronisationszwecke. Zur Tonhöhenbestimmung werden Transformationen benötigt, die den Frequenzbereich in geeignete Teilbänder zerlegen. Da Musiksignale höchstens lokal stationär sind, müssen Transformationen verwendet werden, die neben der Frequenzdarstellung des Signals auch eine Zeitdarstellung liefern. Eine mögliche Zeit-Frequenz-Darstellung erhält man durch die gefensterte Fouriertransformation (WFT). Diese hat allerdings den

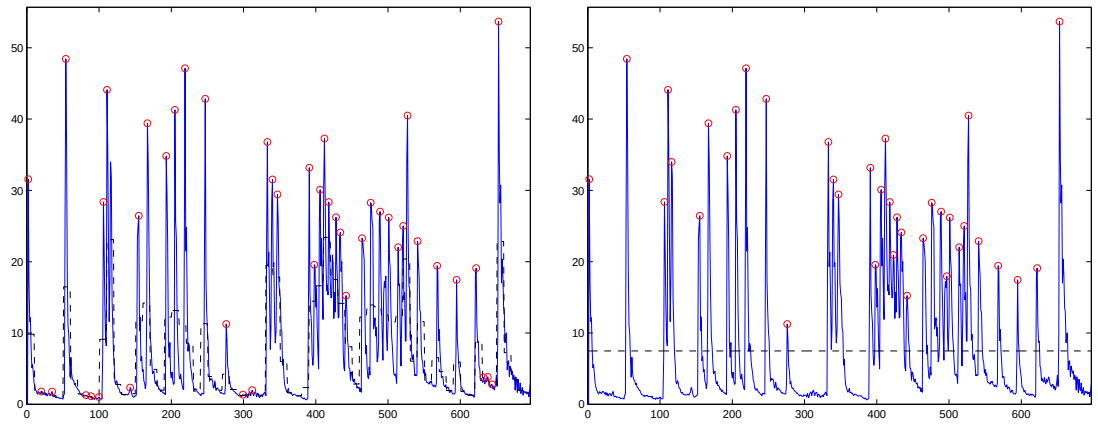


Abbildung 4.9: Peak-Peaking für Bachs Aria und Parameterkombination (a) $N_q = 0.25$ s und $N_{\min} = 0.116$ s, (b) $N_q = 13$ s und $N_{\min} = 0.0929$ s.

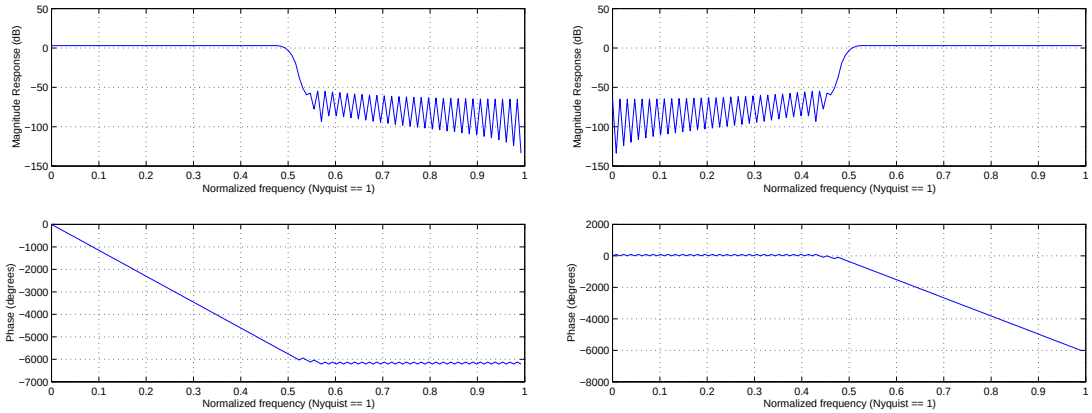
Nachteil, dass durch die gewählte Fensterbreite eine feste Skalierung sowohl im Zeit- als auch im Frequenzbereich eingeführt wird und zu einer Frequenzaufteilung in Bänder gleicher Breite führt. Da aber das menschliche Hörempfinden einen bezüglich der Frequenzen (also Tonhöhen) logarithmischen Charakter hat (Verdopplung der Frequenz entspricht einem Oktavintervall – siehe Abschnitt 2.6), ist eine logarithmische Aufteilung des Frequenzbereiches bei der Zeit-Frequenz-Darstellung sinnvoll. Daher wurden als Basis für unsere Analyse kaskadierte Multiratenfilterbänke verwendet, die eine Oktavbandzerlegung des Frequenzbereiches liefern (Unterabschnitt 4.3.1).

Die Tonhöhenerkennung erfolgt durch eine geeignete Interpretation der Koeffizienten innerhalb der einzelnen Subbänder. Wie im Abschnitt 2.7 erläutert wurde, hängen die Energieverhältnisse der Töne einer Obertonreihe stark von der Lage der Grundfrequenz, der Lautstärke und der Instrumentation ab. Deshalb ist das Problem der Tonhöhenextraktion für den Fall polyphoner Musik ohne weitere Zusatzkenntnisse praktisch nicht lösbar.

Für die chromatische Notenfolge des Klaviers verwenden wir Notenschablonen (engl.: *Templates*) . Diese stellen die Basis für das Template-Matching-Verfahren (Unterabschnitt 4.3.2) dar. Da diese Template-Datenbank nur einen Repräsentanten pro Note hat anstatt Schablonen für alle Notenlängen, Lautstärken, Akzentuierungen und anderen Abspielsweisen, die den zeitlichen Verlauf des Klangspektrums beeinflussen, sowie alle möglichen Typen von Klavieren, ist klar, dass im allgemeinen Fall eine fehlerfreie Extraktion der Tonhöhen nicht zu erwarten ist. In Unterabschnitt 4.3.4 werden diese Phänomene anhand konkreter Beispiele ausführlicher erklärt und mögliche Verbesserungen der Methode diskutiert.

4.3.1 Aufbau des Filterbankbaums

Zur Extraktion der Tonhöhe wird eine Transformation verwendet, die den Frequenzbereich in Bänder unterteilt, welche den Notenfrequenzen unserer westlichen temperierten Stimmung entsprechen. Die Grundfrequenzen der Noten sind dem menschlichen Hörsystem angepasst worden, das logarithmische Empfindlichkeit gegenüber Frequenz- bzw. Lautstärkeänderun-



(a) (b)

Abbildung 4.10: Frequenz- und Phasengang der: (a) Tiefpass- und (b) Hochpass-Filter.

gen zeigt. Die *temperierte Stimmung* teilt die Oktave logarithmisch in 12 Teile. Die Grundfrequenz der musikalischen Töne bezüglich dieser Unterteilung kann mit der folgenden Formel beschrieben werden:

$$\omega_0(p) = 2^{\frac{p-69}{12}} \cdot 440 \quad [\text{Hz}],$$

wobei $p \in [0:127]$ die MIDI-Tonhöhen repräsentiert. Die Grundfrequenz der Note mit der MIDI-Tonhöhe $p = 69$, d. h., die Grundfrequenz der Note A_5 (Kammerton a'), beträgt also $\omega_0(69) = 440$ Hz.

Bei der Tonhöhenextraktion mittels Filterbänken würden wir intuitiv nach einer Lösung streben, die die Grundfrequenzen der Töne möglichst in verschiedene Subbänder verteilt. Damit würden wir die Interpretation der Filterkoeffizienten und deren Zuordnung zu den entsprechenden Tonhöhen vereinfachen. Um eine hierzu geeignete Zerlegung des Frequenzbereiches zu erhalten, muss der Qualitätsfaktor[1] der Filter mindestens den Wert

$$Q_{\min} = \frac{\omega_0(p)}{\omega_0(p+1) - \omega_0(p)} = \frac{1}{2^{\frac{1}{12}} - 1} = 16.72$$

haben. Filter mit solch einem Q-Faktor ermöglichen allerdings nur die Trennung von Grundfrequenzen für die Töne einer einzigen Oktave. Wenn der Tonumfang jedoch mehrere Oktaven umfasst, muss diese untere Schranke noch erhöht werden, um zu ermöglichen, dass auch die sehr nah beieinander liegenden Obertöne verschiedener Noten getrennt werden können.

Ausgehend von der Oktavbandzerlegung einer geeignet gewählten Filterbank (die einen konstanten Q-Faktor von $Q = 1.5$ liefert) werden die einzelnen Oktavbänder in weitere Subbänder zerlegt, bis alle Halbtöne im Frequenzbereich zwischen $\omega_0(38) = 73.41$ Hz (entspricht dem D_3 in MIDI-Notation) und 11025 Hz separiert sind, d. h. Grundfrequenzen verschiedener Halbtöne liegen in verschiedenen Subbändern. Ähnlich wie in [7] wurde dies durch eine Multiraten-Filterbank, deren Baumstruktur aus einem vollständigen Binärbaum der Höhe $r = 6$ besteht, welcher auf weiteren 5 Stufen bzgl. der 2^{r-1} Tiefpassbänder verfeinert ist (siehe Abb.

[1] Der Qualitätsfaktor Q (Q-Faktor) ist als das Verhältnis der Mittenfrequenz ω_c und der Bandbreite (ω_l, ω_h) eines Filters definiert, also

$$Q = \frac{\omega_c}{\omega_h - \omega_l}.$$

4.11), realisiert. Dabei wurden orthogonale FIR-Filter der Länge N = 128 benutzt, deren Frequenzübertragungsverhalten in Abbildung 4.10 dargestellt ist. Diese Struktur führt auf eine Zerlegung in 224 Subbänder mit einem Qualitätsfaktor zwischen $Q_{\min} = 16.72$ und $Q = 31.5$, was deutlich oberhalb der gewünschten Untergrenze liegt. Da in den tiefen Frequenzen zur Trennung der Halbtöne eine Zerlegung in kleinere Bandbreiten notwendig ist, müssen hierfür Filter großer Länge eingesetzt werden, was aber eine schlechte Zeitauflösung nach sich zieht. Die oben beschriebene Filterbankstruktur (Abb. 4.11) wurde als Kompromiss zwischen den widersprüchlichen Forderungen der guten Zeit- *und* Frequenzauflösung gewählt. Diese Filterbank wurde in MATLAB implementiert und sowohl auf synthetisch erzeugte als auch auf natürliche Musiksignale im PCM-Format angewendet und getestet.

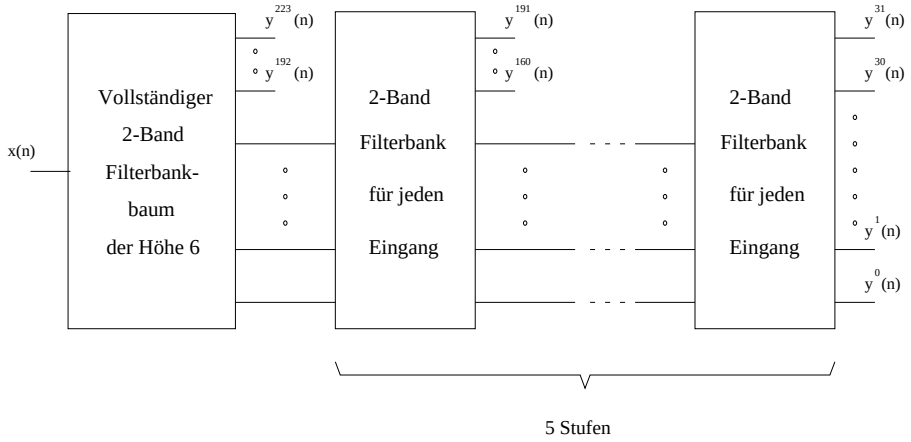


Abbildung 4.11: Aufbau des Filterbankbaums.

4.3.2 Tonhöhenextraktion mittels Schablonen

Tonhöhenextraktion im Fall polyphoner Musik stellt ein sehr kompliziertes Problem dar. Bei monophoner Musik würde es i. A. ausreichen, dass nur die Grundfrequenz, also die niedrigste Komponente im Spektrum, oder im Fall fehlender Grundtöne das Verhältnis der Obertöne (siehe [68]) erkannt wird. Das Problem wird wesentlich komplizierter, sobald wir es mit zweistimmiger Musik zu tun haben. Das liegt an der Überlagerung der Obertöne von Noten, die gleichzeitig klingen (siehe 2.4). Diese Überlagerung kann zum einen partiell geschehen, d. h., nur ein Teil der Obertöne einer Note überlappt sich mit den Komponenten (Grund- und Obertöne) von anderen Noten. Im Fall eines Oktavintervalls kann aber andererseits auch eine totale Überlagerung der Frequenzkomponenten vorliegen. Dies begründet das Bedürfnis für die Verwendung zusätzlicher Informationen, wie z. B. der Amplitudenverhältnisse der Notenpartialtöne.

Einige bekannte Transkriptionssysteme wurden in Abschnitt 4.1 beschrieben. Sie stellen verschiedene Ansätze zur Lösung ähnlicher Probleme dar. Trotzdem ist allen eines gemeinsam: die Modellierung von Noten ist ein zentraler Punkt. Einige Methoden verwenden ein einziges Notenmodell pro Instrument, was sehr oft zu einer fehlerhaften Transkription führt. Dies spiegelt die bekannte Diskrepanz zwischen einfachen und zeitlich effizienten Lösungen einerseits und der Qualität der Ergebnisse andererseits wider (z. B. auf dem Markt erhältliche Produkte

[1], [2]). Andere Autoren verwenden teilweise mehrschichtige Modelle für jeden einzelnen Ton. Der hier beschriebene Ansatz ähnelt teilweise der Tonhöhenextraktionsmethode von Bobrek [6]. Beide Methoden verwenden Notenschablonen und setzen bei der Tonhöhenerkennung eine Subtraktionsmethode ein. Der wesentliche Unterschied zwischen beiden Algorithmen liegt aber in verschiedenen Ansätzen, wie die Normierung und die Subtraktion genau durchgeführt werden.

Es folgen nun Überlegungen zur Wahl der richtigen Parameter für die Erzeugung von Notenschablonen.

Erzeugung der Schablonen

Wie in Abschnitt 2.7 erläutert, verändert sich das Klangspektrum der Töne während des zeitlichen Ablaufs. Nach ihrer größten Abweichung aus der Ruheposition, die die Klaviersaite erst eine gewisse Zeit erreicht, nachdem der Hammer sie getroffen hat, fällt die Amplitude dieser Schwingungen exponentiell mit der Zeit ab. Allmählich fällt auch die Lautstärke des Tons, den sie erzeugt hat, ab (siehe Abbildung 2.8). Auch wenn die Lautstärke des Klangs mit der Zeit perzeptuell gleichmäßig schwächer wird, ist das Verhältnis der Partialtöne als Funktion der Zeit völlig anders. Wie in Abbildung 2.9 zu erkennen ist, verändert sich die Abklingkurve jedes einzelnen Partialtons zeitlich. Dies geschieht i. A. für jeden Partialton unterschiedlich ([4]).

Die für die Tonhöhenextraktion wichtigsten Fakten über das Klangspektrum von Klaviertönen können wie folgt zusammengefasst werden (vgl. auch Abschnitt 2.7):

(a) Die Amplitude der Grundfrequenz eines Tons, insbesondere eines aus der Bass-Lage, kann 20-25 dB unterhalb des Pegels der stärksten Komponenten liegen.

(b) Die eigentlichen Träger der Lautstärke der Klaviertöne sind die höheren Obertöne (siehe Abb. 4.12).

(c) Das Dämpfer-Pedal verringert nicht nur die Lautstärke des Klangs, sondern auch seine spektrale Struktur.

(d) Die Partialtöne besitzen ganz unterschiedliche Abklingzeiten. Manche werden zwischenzeitlich sogar wieder lauter, bevor sie sich abschwächen. Die Toleranzgrenzen der Abklingzeit, also der Zeit zwischen dem ersten Lautstärkemaximum und dem Verstummen des Tones, liegen zwischen zwei und 5.5 Sekunden (vgl.[4]).

(e) Die höheren Obertöne des Klaviers weichen von den idealharmonisch erwarteten Positionen ab. Diese Abweichung steigt mit der Nummer n des Obertons und ist proportional zu n^2. Diese Inharmonizität steigt auch mit der abnehmenden Länge der Klaviersaite.

Die erste obige Beobachtung führt zu dem Schluss, dass die Normierung der Schablone anhand der stärksten Komponente nicht sinnvoll ist, da diese nicht unbedingt die Grundfrequenz repräsentiert (siehe unten Abb. 4.15). Der Punkt (e) weist darauf hin, dass die Positionen höherer Obertöne im Fall des Klaviers nicht als Vielfache des Grundtons zu erwarten sind. In diesem Fall wäre eine explizite Bestimmung der Positionen der Obertöne erwünscht. Die

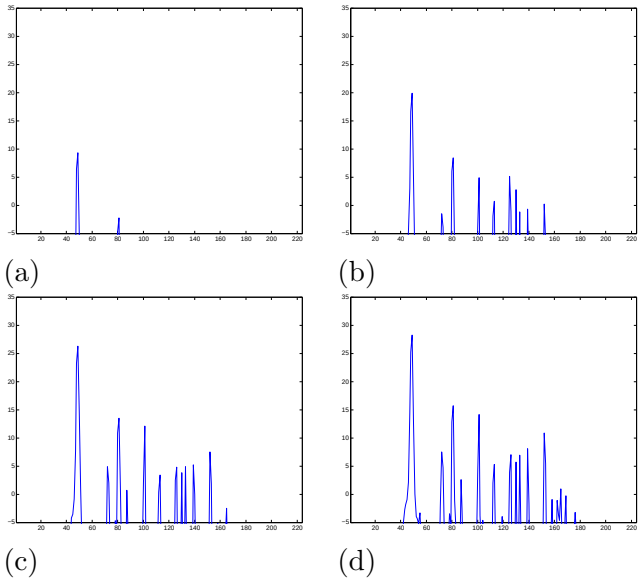
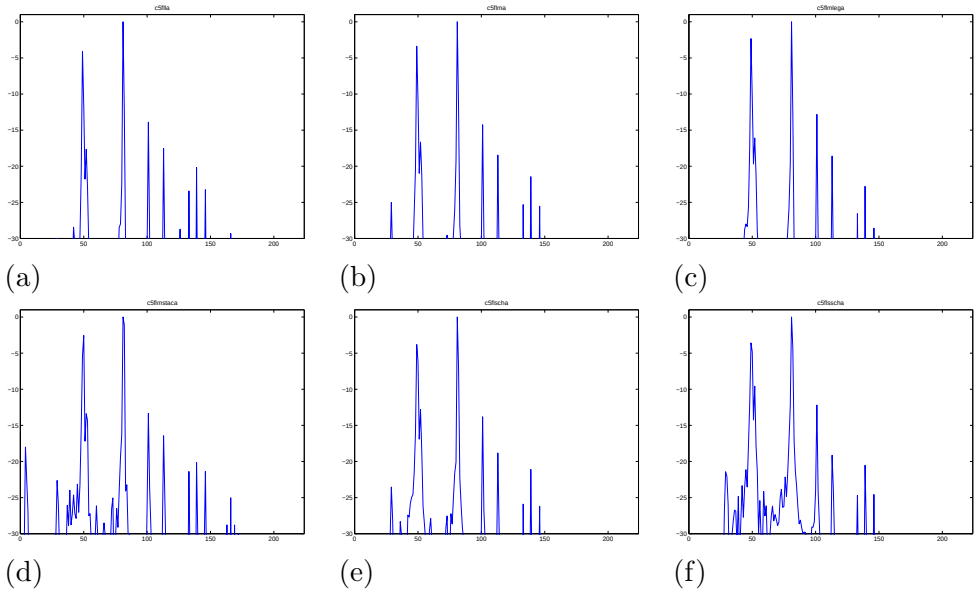


Abbildung 4.12: Schablonen für die Tonhöhe c_5, erzeugt mit Yamahas Gran-Touch E-Piano. Beispiel für die Dynamik: (a) piano, (b) mezzoforte, (c) forte, (d) fortissimo.

Inharmonizität der Obertöne ist jedoch sowohl vom Instrument als auch von jedem einzelnen Ton abhängig (siehe z. B. [32] und [78] für einige Beispiele). Wir müssten also die genaue Grundfrequenz der Töne und deren Inharmonizitätskoeffizienten für jedes Klavier zur Verfügung haben, was für solch einen Ansatz praktisch nicht realisierbar ist. Also müssen wir unsere Analyse auf bestimmte, uns zur Verfügung stehende Klaviere einschränken, mit der Gewissheit, dass es zu Abweichungen der Positionen der Obertöne kommen kann. Dies kann trotz der Verwendung von Schablonen Tonhöhenextraktionsfehler zur Folge haben. Diese Extraktionsfehler tauchen auch als Folge unterschiedlicher Klangspektren von Tönen, die von verschiedenen Klavieren erzeugt werden, auf (vergleiche die Abbildungen 4.13 und 4.14). Die Punkte (b) und (c) zeigen, dass es keine universelle Notenschablone auch nur bezüglich eines einzigen Klaviers geben kann, da die Amplitude der Komponenten stark von der Lautstärke, Benutzung des Pedals und den Notenlängen abhängig ist. Man könnte natürlich mehrere Schablonen pro Tonhöhe für alle möglichen Kombinationen dieser verschiedenen Parameter erzeugen, was aber zu einer riesigen Schablonendatenbank führen würde. In einer anderen Erweiterung könnten die Schablonen explizit die zeitliche Evolution der Frequenzkomponenten widerspiegeln. In diesem Fall würden wir für jede Tonhöhe und für jede Kombination der oben erwähnten Parameter eine Zeit-Frequenz-Darstellung als “Schablone” speichern. Eine dritte Möglichkeit wäre die Modellierung aller Phasen des ADSR-Modells (ähnlich wie in [9]), in welchem Fall wir für jede Abspielweise und jede Phase einer Note eine Schablone hätten.

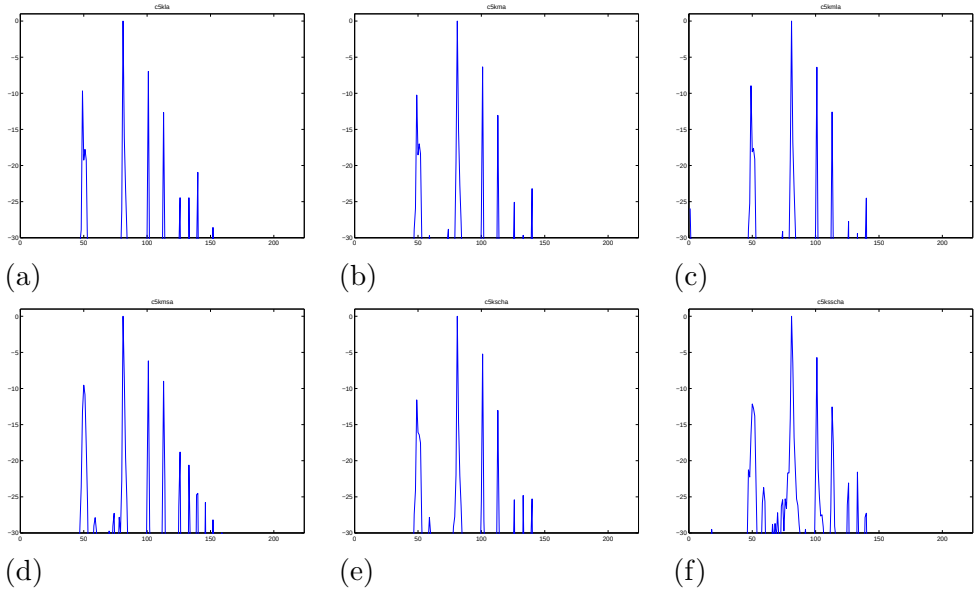
Zwar klingen diese Erweiterungen vielversprechender als die Verwendung einer einzigen Schablone pro Note. Wenn wir jedoch ein System entwickeln wollen, das zusätzlich unabhängig von der Bauart des Instruments ist, müssen wir mehrere Schablonen pro Note verwenden, was zu einer viel komplizierteren Extraktionsmethode führt. Da wir es aber nicht mit dem Problem der Transkription, sondern mit der Synchronisation von PCM-Daten und Partitur-Daten bzw. MIDI-Daten zu tun haben, steuern wir eine sinnvolle Kompromisslösung zwischen der Komplexität der Methode und der Qualität der Ergebnisse an.



(a) (b) (c)

(d) (e) (f)

Abbildung 4.13: Schablonen für die Tonhöhe c_5, erzeugt mit einem Steinway-Flügel. Spielweise: (a) langsam, (b) mittelschnell, (c) mittelschnell, legato, (d) mittelschnell, staccato, (e) schnell, (f) schnell, staccato.



(a) (b) (c)

(d) (e) (f)

Abbildung 4.14: Schablonen für die Tonhöhe c_5, erzeugt mit einem Schimmel-Klavier. Spielweise: (a) langsam, (b) mittelschnell, (c) mittelschnell, legato, (d) mittelschnell, staccato, (e) schnell, (f) schnell, staccato.

Zur Erzeugung der Schablonen wird die Subbandtransformation für jeden einzelnen Ton eines der untersuchten Instrumente[2] durchgeführt. Die Verhältnisse der Energien der einzelnen Obertöne in den Subbändern ergeben die Schablonen. Eine Schablone ist also ein Vektor $S \in \mathbb{R}^{\kappa}$ ($\kappa = 224$ in unserem konkreten Fall). In Abbildung 4.15 ist zu sehen, wie diese Schablone für die Tonhöhen c_2 bis c_5 im Fall des Yamaha GranTouch E-Pianos aussehen. Man erkennt, wie unregelmäßig die Verhältnisse der Obertöne für verschiedene Tonhöhen sind. Diese Unregelmäßigkeiten bestätigen unsere obige Vermutung, dass ein einziges Modell für die gesamte Tonlage nicht ausreichend ist. Daher verwenden wir in unserem Template-Matching-Algorithmus für jede Note eine eigene Schablone.

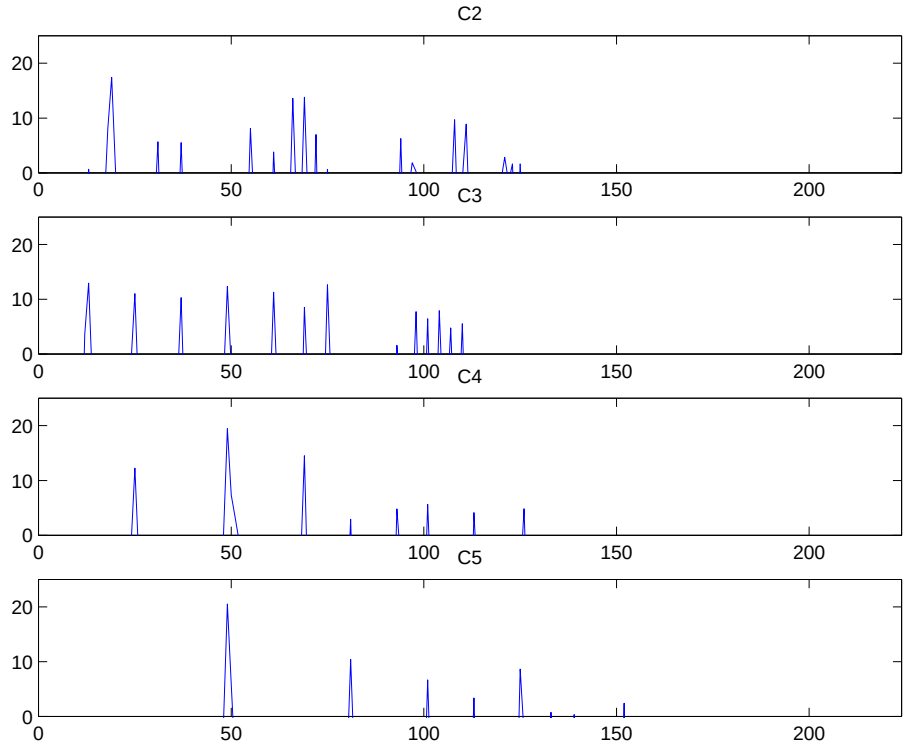


Abbildung 4.15: Schablone für Tonhöhen C_2 bis C_5 (Die x-Achse stellt die Subbandnummern und die y-Achse die Energie der Subbandkomponenten dar). Nur die signifikanten Komponenten wurden gezeigt. Man kann feststellen, dass u. a. der Grundton des Tons C_2 fehlt und der erste Oberton sehr schwach ist.

4.3.3 Template-Matching-Methode für die Tonhöhenextraktion

Die Template-Matching-Methode basiert auf einem Vergleich der Verteilung der Subbandenergie eines untersuchten Zeitintervalls mit den Notenschablonen. Das Zeitintervall ist dabei ein Intervall, das zwischen zwei benachbarten detektierten Einsatzzeiten liegt. Da diese Einsatzzeiten aus dem ungefilterten Signal geschätzt wurden, müssen die von der Filterbank eingeführten zeitlichen Verzögerungen kompensiert werden. Da es sich hier um eine Multi-

[2]Es wurden insgesamt drei verschiedene Klaviere getestet: Yamahas GranTouch E-Piano, ein Steinway-Flügel und ein Schimmel-Klavier.

ratenfilterbank handelt, die im allgemeinen Fall auch Filter verschiedener Längen enthält, ist die zeitliche Verzögerung für jedes Subband verschieden. Wir gehen nun näher auf die Berechnung dieser Verzögerungen und die Erzeugung der Energievektoren ein, bevor wir den gesamten Tonhöhenextraktionsalgorithmus beschreiben.

Erzeugung der Energievektoren

Der Träger des in 4.2.4 geschätzten Peaksignals $P : \mathbb{Z} \to \mathbb{R}$ liefert eine Menge von Einsatzzeiten $\{p_1, \ldots, p_P\} := \text{supp}(P)$. Da diese alle verschieden sind, wählen wir eine aufsteigende Nummerierung $p_1 < \ldots < p_P$. Für jede Einsatzzeit $p \in \text{supp}(P)$ eines Eingangssignals $x : \mathbb{Z} \to \mathbb{R}$ wollen wir nun die zugehörige Position im k-ten Subbandsignal bestimmen. Mit der Verwendung von Operatoren und der Semantik von Filterbank-Bäumen, die in Abschnitt 3.3 definiert sind, können wir aus der inversen Z-Transformation des Signals im Ausgang des Subbands k, dessen Semantik durch $W(k) := \varphi_{k_r} \circ \cdots \circ \varphi_{k_0}$ definiert ist, die zeitliche Verzögerung der Einsatzzeiten wie folgt berechnen. Es gilt

$$W(k) = (\downarrow q_{r+1}) \circ C_{(\uparrow q_r)h_r} \circ \cdots \circ C_{(\uparrow q_1)h_1} \circ T_{l_a},$$

wobei $q_0 := 1$, $q_i := D_0 \cdots D_{i-1}$ und $l_a := l_r q_r + \cdots + l_1 q_1$ sind [3]. Aus

$$Y_a^{(k)}(z) = W_z(k) \circ X(z) = \left[(\downarrow q_{r+1})_z \circ H_z(k) \circ z^{l_a} \right] \circ X(z) = \cdots$$

$$= \left[\sum_{s=0}^{q_{r+1}-1} \frac{1}{q_{r+1}} Y\left(z^{\frac{1}{q_{r+1}}} e^{\frac{2\pi i}{q_{r+1}} s} \right) \right] z^{\frac{l_a}{q_{r+1}}}$$

folgt

$$y_a^{(k)}(n) = Z^{-1}\left[Y_a^{(k)}(z) \right] = \cdots = \frac{1}{q_{r+1}} \sum_{s=0}^{q_{r+1}-1} Z^{-1}\left[Y\left(z^{\frac{1}{q_{r+1}}} e^{\frac{2\pi i}{q_{r+1}} s} \right) z^{\frac{l_a}{q_{r+1}}} \right] = \cdots = y(nq_{r+1} + l_a),$$

wobei $y(n) = Z^{-1}\left(H_z(k) X(z) \right)$.

Die Abbildung der Einsatzzeitenpositionen p in jedes Subband k ist jetzt für unseren Spezialfall mit dem Downsampling-Faktor $D_0 = 1$, $D_1 = \cdots = D_r = D$ und $l_1 = \cdots = l_r = l$ leicht aus der Phase der y_a, nämlich $p = nq_{r+1} + l_a$, zu berechnen. Also wird die im Signal x betrachtete Einsatzzeit p im k-ten Subband an der Position

$$\rho(k, p) := n = \frac{p - l_a}{q_{r+1}} = \cdots = \frac{p + l}{D^r} - l$$

auftauchen. Diese Abbildung hängt von der Tiefe r des Baums und der Länge der verwendeten Filter $L = 2 \cdot l$ ab. Die zeitliche Verzögerung wird in unserem Fall nur einmal für die jeweilige Subbandgruppe berechnet, da diese Gruppen jeweils aus Filtern gleicher Längen bestehen.

[3] Zur Erinnerung: $(\downarrow D)$ stellt hier den Downsampling-Operator, C_h den Faltungsoperator und T_ℓ den Translationsoperator dar.

Jetzt können wir die Energie-Vektoren $E_i \in \mathbb{R}_{\geq 0}^{N_{\mathrm{sb}}}$ für jedes Zeit-Intervall zwischen zwei geschätzten Einsatzzeiten p_i und p_{i+1} durch die Formel

$$E_i^{(k)} = \sum_{j=\rho(k,p_i)}^{\rho(k,p_{i+1})-1} \left[y_j^{(k)}\right]^2,$$

bestimmen.

Template-Matching Algorithmus

Die in Abschnitt 4.2 beschriebene Zeitsegmentierung liefert die obige Folge $p_1, \ldots, p_P$ von geschätzten Einsatzzeiten. Auf dieser Basis werden, wie gerade beschrieben, die Energievektoren E_i erzeugt. Jedem zeitlichen Intervall $[p_i, p_{i+1} - 1]$, $1 \leq i \leq P$, entspricht somit ein Energievektor. Der Template-Matching Algorithmus wird iterativ für jedes Zeitintervall bzw. für jeden Energievektor E_i durchgeführt. Er wird durch den unten gegebenen Pseudo-Code beschrieben.

Algorithmus: Extraktion von Tonhöhen mittels Schablonen

Input:

E_i	–	der Energievektor.
$a \in (0, 1]$	–	Parameter zur Bestimmung der niedrigsten bedeutenden Energiekomponente in E_i.
$b \in (0, 1]$	–	Parameter zur Bestimmung der leisesten akzeptierbaren Note im Akkord.

Initialisiere:

$E' := E_i$		
$\Pi_i = \emptyset$	//	Menge der detektierten Tonhöhen
$\mathcal{E}_i = \emptyset$	//	Menge der detektierten Tonenergien

Start.

1. Finde die Position k der niedrigsten „bedeutenden“ Komponente des Vektors E', die die Bedingung
$$E'(k) \geq a \cdot \frac{1}{|\mathrm{supp}(E')|} \|E'\|_1,$$
erfüllt.

2. Teste, ob es eine Notenschablone S_k gibt, die dieser Komponente zugeordnet werden kann, d. h., deren erste Komponente im selben Subband k liegt[4].

   (i) Falls ein geeignetes S_k existiert, setze

   a) $\check{E}' := \frac{1}{E'(k)} \cdot E'$, (Normierung des Energievektors)

[4]Die Struktur des Filterbankbaums ist so gewählt, dass ab der Tonhöhe 38 jedem Grundton ein Subband entspricht, das er mit keinen weiteren Grundtönen teilen muss.

b) für $j = 1 : |\mathrm{supp}(E')|$

$$\Delta(j) = \begin{cases} \check{E}'(j) - S_k(j), & \text{falls} \quad \check{E}'(j) - S_k(j) > 0, \\ 0 & \text{sonst.} \end{cases}$$

c) $\varepsilon := \|E'\|_1 - \|\Delta \cdot E'(k)\|_1$, (Energie der Note)

d) $\pi :=$ Tonhöhe zur Schablone S_k,

e) $E' = \Delta \cdot E'(k)$, („Rücknormierung“)

(ii) sonst erkläre die Komponente k als rauschartig und setze $E'(k) := 0$.

3. (i) Falls $\frac{\varepsilon}{\|E_i\|_1} \geq b$, akzeptiere die detektierte Note (π, ε) und aktualisiere $\Pi_i = \Pi_i \cup \{\pi\}$ und $\mathcal{E}_i = \mathcal{E}_i \cup \{\varepsilon\}$,

(ii) sonst berücksichtige sie nicht mehr weiter, da sie zu „leise“ ist.

4. Gehe zu 1., bis $\|E'\|_1 \leq b \cdot \|E_i\|_1$ wird.

Ende.

Output: Menge der detektierten Tonhöhen Π_i und ihrer Energien $\mathcal{E}_i$.

Für den Schritt 2. (ii) gibt es folgende Erweiterungsmöglichkeit. Es ist oft ungünstig, eine Komponente einfach zu eliminieren, insbesondere wenn diese eine signifikante Energie besitzt. Es gibt nämlich Fälle, bei denen die Grundfrequenzen am Rande eines Subbands liegen und wegen der Überlappung der benachbarten Subbänder (*Aliasing*) zwischen diesen verteilt werden können. Dies kann durch leichte Abweichungen in der Stimmung der Instrumente verstärkt werden. Der Algorithmus überprüft, ob in den benachbarten Subbändern eine Grundfrequenz zu erwarten ist. Wenn dies der Fall ist, dann wählt man aus den beiden diejenige Schablone, die den kleinsten Fehler liefert (kleinste resultierende Differenzenergie E' in Schritt 2.(i)(e))[5]. Man könnte das Problem alternativ auch zu lösen versuchen, indem man die exakte Frequenz-Position der Komponente per FFT bestimmt und anhand der Nachbarschaftsrelationen das wahrscheinlichere Subband und folglich die entsprechende Schablone direkt auswählt. Das würde aber den Algorithmus unnötig verkomplizieren und die Berechnungskosten erhöhen, weshalb wir diese Variante ausgeschlossen haben.

Nun noch einige Kommentare zur gewählten Normierung der Energievektoren. Aus den in Unterabschnitt 4.3.2 aufgelisteten Eigenschaften wurde bereits geschlossen, dass die größte Komponente nicht immer den Grundton einer Note darstellt. Im Hinblick auf häufige Komponentenüberlagerungen in polyphoner Musik musste ein Ansatz entwickelt werden, der auf eindeutig zuzuordnenden Komponenten basiert. Dies sind solche Komponenten, die nicht zu mehreren Noten zugeordnet werden können, sondern nur zu einer einzigen Note gehören. Deshalb beginnt der vorliegende Algorithmus für die Tonhöhenextraktion mit der niedrigsten signifikanten Komponente des Energievektors (Schritt 1.) und normiert diesen Vektor auf der Basis der Energie gerade dieser Komponente, bevor die Schablone gewählt wird. So erhalten wir eine korrektere Schätzung der Energie des extrahierten Tons unabhängig von der Dynamik im Kontext des Akkords, wo verschiedene Noten mit ganz unterschiedlichen Lautstärken gespielt werden können. Somit besteht keine Gefahr, dass die stärksten Komponenten (Obertöne) gleich eliminiert werden, was im Fall der Normierung und Extraktion, auf der Basis der stärksten Komponente des Spektrums (vgl. [6]) zu erwarten wäre.

[5] Ähnlich werden auch die tieferen Tonhöhen behandelt, deren Grundtöne wegen der niedrigeren Frequenzauflösung in den unteren Subbändern sich das selbe Subband teilen.

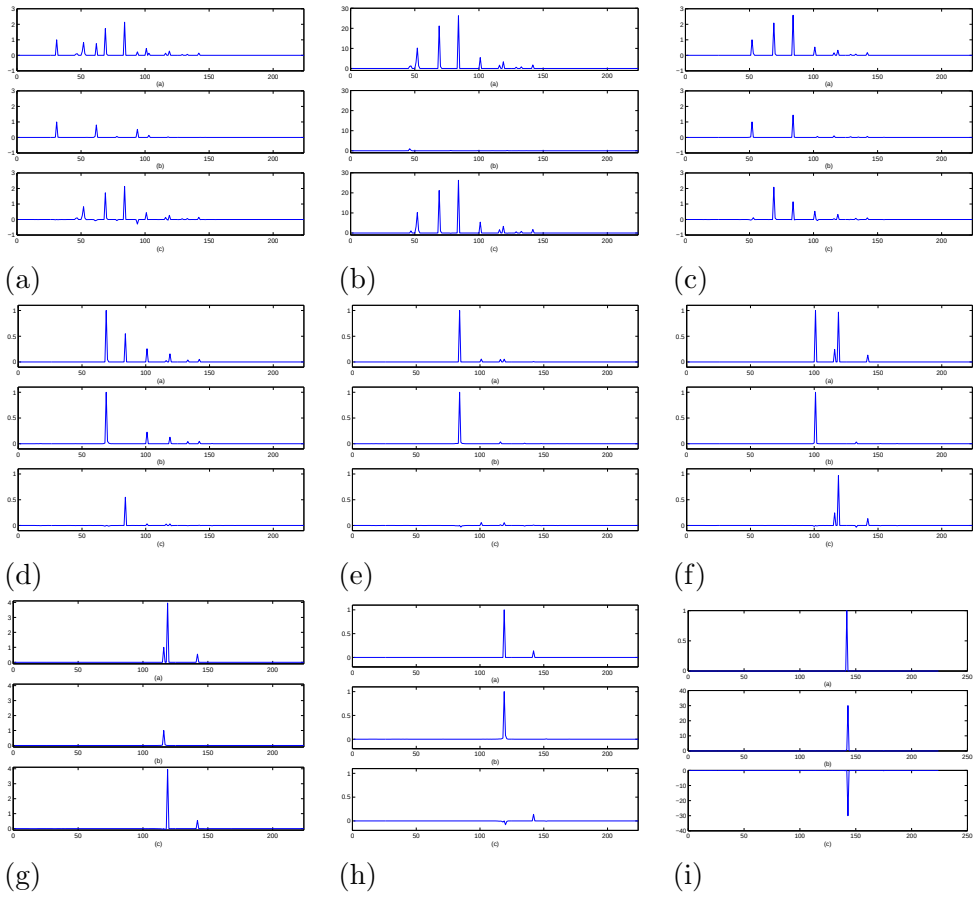


Abbildung 4.16: Bestimmung der Tonhöhen mittels Schablonen. Abb. (a) zeigt einen typischen Iterationsschritt: vom aktuellen Energievektor (oben) wird die „linkeste“ Schablone (Mitte) abgezogen und führt zum neuen Energievektor (unten), der den Ausgangspunkt des nächsten Iterationsschritt (hier Abb. (b) oben) bildet. Im ersten Iterationsschritt (Abb. (a)) wurde die Tonhöhe 52 erkannt, im zweiten Iterationsschritt (Abb. (b)) wurde mangels Energie keine Tonhöhe extrahiert, im dritten (Abb. (c)) die Tonhöhe 61, im vierten (Abb. (d)) die Tonhöhe 67 und im fünften (Abb. (e)) die Tonhöhe 73. Die weiteren Schritte haben keine energiereichen Töne mehr geliefert, auch wenn der Algorithmus dies aus der übriggebliebenen Gesamtenergie nicht direkt schließen kann.

Abbildung 4.16 zeigt die zeitliche Entwicklung der Subbandenergien bei der Tonhöhenextraktion mittels Schablonen. In insgesamt neun Iterationsschritten sind vier Tonhöhen mit den MIDI-Bezeichnungen 52 (im ersten Iterationsschritt), 61 (im dritten Iterationsschritt), 67 (im vierten Iterationsschritt) und 73 (im fünften Iterationsschritt) erkannt worden. Vergleicht man das Analyseergebnis mit den Partiturinformationen, so stellt man fest, dass die Töne 61 und 67 neu angespielte, gesuchte Töne sind und dass 52 einen nachklingenden Ton darstellt, der zu einem früheren Zeitpunkt gespielt wurde. Dabei stellt der Ton 73 nur einen Oberton des Tons 61 dar, was wir im Folgenden als *Oktavfehler* bezeichnen.

Die Problematik der Erkennung von Oktavintervallen wegen der Überlagerung von Partialtönen wurde bereits im Abschnitt 2.4 erläutert. Durch die Einführung von Tonhöhenschablonen haben wir versucht, dieses Problem in den Griff zu bekommen. Da aber durch diese Schablonen nicht alle möglichen Spielweisen, Instrumente und Lautstärken „abgedeckt“ werden können, tauchen zwei verschiedene Fälle auf:

- Ein Oktavintervall wurde gespielt. Hier kann es passieren, dass während der Erkennung des tieferen Tons der Grundton des hohen Tons „eliminiert“ wird. Der Algorithmus nimmt nun an, dass der erste Oberton von hinreichender Energie ein Grundton sein muss. Dadurch ergibt sich ein *Intervallfehler*.
- Kein Oktavintervall wurde gespielt. Da aber der normierte Energievektor nicht identisch mit der Schablone ist, kann es passieren, dass die Obertöne nicht ganz eliminiert werden. Falls die Energie dieser „Differenzkomponenten“ ausreichend ist, werden durch den Algorithmus weitere Tonhöhen oberhalb des tatsächlich gespielten Tons „erkannt“. Diese stellen ebenfalls Intervallfehler dar.

Die Oktavfehler sind ein Spezialfall der Intervallfehler. Da Oktavfehler bei der Extraktion am häufigsten beobachtet werden, haben wir im nächsten Kapitel bei der Synchronisation von Partitur- bzw. MIDI-Dateien mit PCM-Dateien insbesondere diese Art von Extraktionsfehlern mitberücksichtigt.

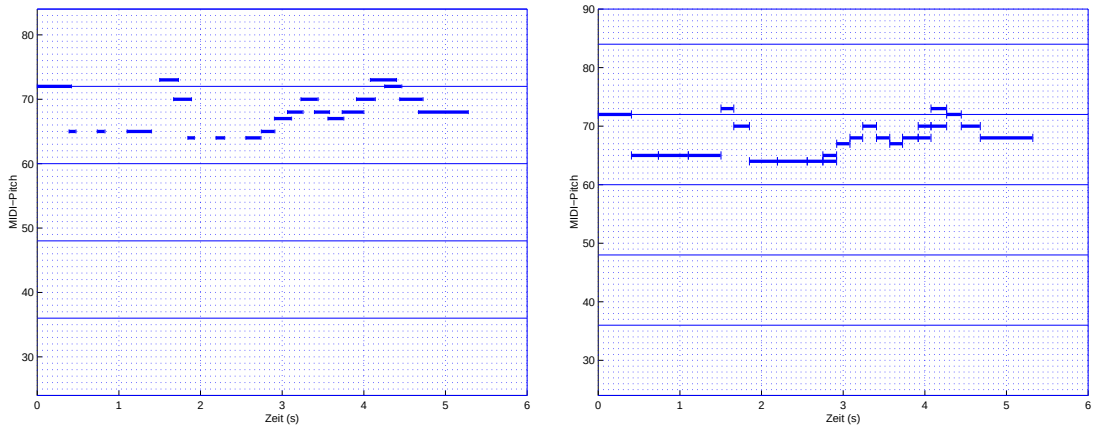
4.3.4 Parameterwahl für den Tonhöhenextraktionsalgorithmus und Diskussion der Ergebnisse

Aus unserer Sammlung von zahlreichen Testdaten haben wir für die Gütebewertung des Tonhöhenextraktionsalgorithmus eine Reihe typischer Beispiele ausgesucht, die in Tabelle 4.2 aufgelistet sind. Hier handelt es sich sowohl um monophone als auch um polyphone Musikabschnitte von 2.5 bis 14 Sekunden Dauer. Die Tests sollten zeigen, wie der Algorithmus sich bei der Tonhöhenextraktion bei Musikstücken verschiedener Polyphonien, Aufzeichnungsarten bzw. Instrumente sowie unter der Verwendung verschiedener Schablonensätze verhält.

Im ersten Beispiel handelt es sich um einen monophonen Abschnitt, der auf einem Yamaha *Gran-Touch* E-Piano gespielt wurde. Der optimale Energieparameter b, mit dem die minimale Energie des Tons relativ zur Energie des Intervalls gesteuert wird (Schritt 3(i) im Tonhöhenextraktionsalgorithmus), liegt, wie bei monophonen Abschnitten auch zu erwarten ist, zwischen $0.1 - 0.4$. Dies bedeutet, dass die „Empfindlichkeit“ des Algorithmus bei den monophonen Musikabschnitten eher etwas „kleiner“ sein muss, da pro Intervall höchstens

M.	Musikstück	Art der Aufnahme	phys. Dauer	max. Polyphonie
1.	Monophon 1	E-Piano	5.3 s	1
2.	C-Dur Kadenz	E-Piano	6.5 s	6
3.	Bach/Var. 21	E-Piano	13.3 s	3
4.	Bach/Var. 21	Steinway-Flügel	13.3 s	3
5.	Mozart/Trio	CD-Aufnahme	12.5 s	4
6.	Synth. Akkorde	Capella/Cakewalk	2.5 s	10

Tabelle 4.2: Teststücke zur Gütebewertung des Tonhöhenextraktionsalgorithmus.



(a) (b)

Abbildung 4.17: Klavierwalzendarstellung des ersten Beispiels aus der Tabelle 4.2. (a) Die zugehörige MIDI-Datei, (b) die extrahierten Ergebnisse.

eine Tonhöhe zu erwarten ist. Allerdings darf man den Wert von b nicht beliebig erhöhen, um sicher zu gehen, dass alle Töne erkannt werden. Aus Tabelle 4.3 geht hervor, dass die optimale Tonhöhenextraktion für $b = 0.3$ (entspricht 30% der Energie des Intervalls) erreicht wurde. Alle 21 erwarteten Töne, d.h. 100%, wurden richtig erkannt. Drei der Töne klingen auch im folgenden Intervall nach. Falsche Töne oder Intervallfehler kommen hier nicht vor. Die Klavierwalzendarstellung der Ergebnisse ist in Abbildung 4.17 (b) zu sehen. Ein Vergleich mit Abbildung 4.17 (a), der Klavierwalzendarstellung der zugehörigen (gleichzeitig aufgezeichneten) MIDI-Datei, zeigt bis auf Notendauern eine weitgehende Übereinstimmung. Steigert man den Parameter b auf 0.4, so erhält man nur 19 der 21 Tonhöhen (90.48%) in ihren erwarteten Positionen. Ein Ton wird erst später und einer gar nicht detektiert. Wählt man $b = 0.1$, so werden neben den 21 korrekten und 12 nachklingenden Tönen, weiterhin noch 3 falsche Töne extrahiert. Ein noch kleinerer Parameterwert führt zu weiteren Fehldetektionen.

Im zweiten Beispiel handelt es sich um eine C-Dur Kadenz (insgesamt vier sechsstimmige Akkorde) gespielt auf dem E-Piano. Die Ergebnisse des Tonhöheextraktionsalgorithmus für verschiedene Werte von b befinden sich in den Zeilen 5 bis 8 in der Tabelle 4.3. Für großes $b = 0.1$ wurden in allen vier Intervallen nur drei der jeweils sechs Töne extrahiert. Zwei weitere Töne stehen mit den detektierten in einem Oktavverhältnis, und der Algorithmus konnte sie nicht erkennen. In jedem Intervall blieb ein Ton unerkannt. Senken wir den Wert des Parameters b, so erhalten wir immer bessere Ergebnisse. Optimal ist die Wahl $b = 0.02$,

Nr.	M.	b	Sch.	Partiturseite				extrahierte Daten				
				Korr.	O. F.	n. d.	s. d.	Korr.	O. F.	s. d.	n. T.	f.
1.	1	0.4	EP	90.48	0.00	4.76	4.76	90.48	0.00	4.76	4.76	0.00
2.	1	0.3	EP	100.00	0.00	0.00	0.00	87.5	0.00	0.00	12.5	0.00
3.	1	0.1	EP	100.00	0.00	0.00	0.00	58.33	0.00	0.00	33.33	8.33
4.	1	0.04	EP	100.00	0.00	0.00	0.00	41.18	0.00	0.00	31.37	27.45
5.	2	0.1	EP	50.00	33.33	16.67	0.00	85.71	14.29	0.00	0.00	0.00
6.	2	0.04	EP	58.33	29.17	12.50	0.00	77.78	16.67	0.00	0.00	5.56
7.	2	0.02	EP	75.00	25.00	0.00	0.00	78.26	17.39	0.00	0.00	4.35
8.	2	0.01	EP	75.00	25.00	0.00	0.00	75.00	16.67	0.00	0.00	8.33
9.	3	0.3	EP	60.38	0.00	32.08	7.55	82.05	0.00	10.26	7.69	0.00
10.	3	0.1	EP	100.00	0.00	0.00	0.00	65.43	1.23	0.00	32.10	1.23
11.	3	0.04	EP	100.00	0.00	0.00	0.00	47.75	8.11	0.00	40.54	3.60
12.	3	0.1	SF	94.34	1.89	3.77	0.00	62.50	3.75	0.00	32.50	1.25
13.	4	0.04	SF	96.23	1.89	1.89	0.00	49.04	5.77	0.00	41.35	3.85
14.	4	0.1	EP	77.36	11.32	5.66	5.66	47.67	18.60	3.49	25.58	4.65
15.	4	0.04	EP	94.34	5.66	0.00	0.00	39.06	16.41	0.00	25.58	35.16
16.	4	0.1	SF	84.91	7.55	3.77	3.77	56.25	11.25	2.50	28.75	1.25
17.	4	0.04	SF	92.45	5.66	1.89	0.00	43.75	8.04	0.00	36.61	11.61
18.	5	0.1	EP	75.56	10.00	11.11	3.33	61.82	6.36	2.73	24.55	4.55
19.	5	0.04	EP	91.11	4.44	3.33	1.11	48.81	4.76	0.60	40.48	5.36
20.	5	0.1	SF	70.00	11.11	13.33	5.56	59.43	6.60	4.72	24.53	4.72
21.	5	0.04	SF	81.11	6.67	7.78	4.44	47.40	7.14	2.60	37.66	5.19
22.	6	0.01	EP	77.42	19.35	3.23	0.00	48.00	34.00	0.00	10.00	8.00

Tabelle 4.3: Ergebnisse des Tonhöhenextraktionsalgorithmus für die Musikstücke $M \in [1:6]$ aus der Tabelle 4.2 in Abhängigkeit vom Parameter „b“ (minimale relative Energie) und der Schablonen „Sch.“. „Korr.“ steht für korrekt extrahierten Töne, „O. F.“ für Oktavfehler, „n. d.“ für nicht detektierte Töne, „s. d.“ für die in einem späteren Intervall detektierten Töne, „n. T.“ für nachklingende Töne und Oktavfehler und schließlich „f.“ für falsch extrahierte Töne oder andere Intervallfehler, die aber keine Oktavfehler darstellen. Diese Angaben sind in Prozent angegeben und wurden sowohl relativ zur Gesamtanzahl der Töne in der Partitur als auch relativ zur Gesamtanzahl der extrahierten Töne berechnet.

wobei in zwei Intervallen je fünf Töne erkannt werden und in zwei anderen je vier. Die weiteren Töne stehen (wieder) in einem Oktavverhältnis mit einem der erkannten Töne. In diesem Fall blieben keine Töne unerkannt.

Die Beispiele 3, 4 und 5 führen zu ähnlichen Schlussfolgerungen: je mehr Töne gleichzeitig einsetzen desto kleiner muss der Wert für den Parameter b sein. Dies garantiert, dass eine größere Anzahl der zu erwartenden Tonhöhen erkannt wird. Auf der anderen Seite führen kleine Werte für b leichter zu Oktavfehlern und Fehldetektierung von (nachklingenden) Noten. Zur Illustration der obigen Aussagen wurde in Abbildung 4.18 (a) die Klavierwalzendarstellung einer MIDI-Datei entsprechend dem dritten Beispiel aus Tabelle 4.2 dargestellt, zusammen mit den Klavierwalzendarstellungen der resultierenden Extraktionsergebnisse für $b = 0.3$ (Abb. 4.18 (b)), $b = 0.1$ (Abb. 4.18 (c)) und $b = 0.04$ (Abb. 4.18 (d)). In Abbildung 4.19 sind die Klavierwalzendarstellung einer MIDI-Datei sowie die extrahierten Ergebnisse von Mozarts Trio

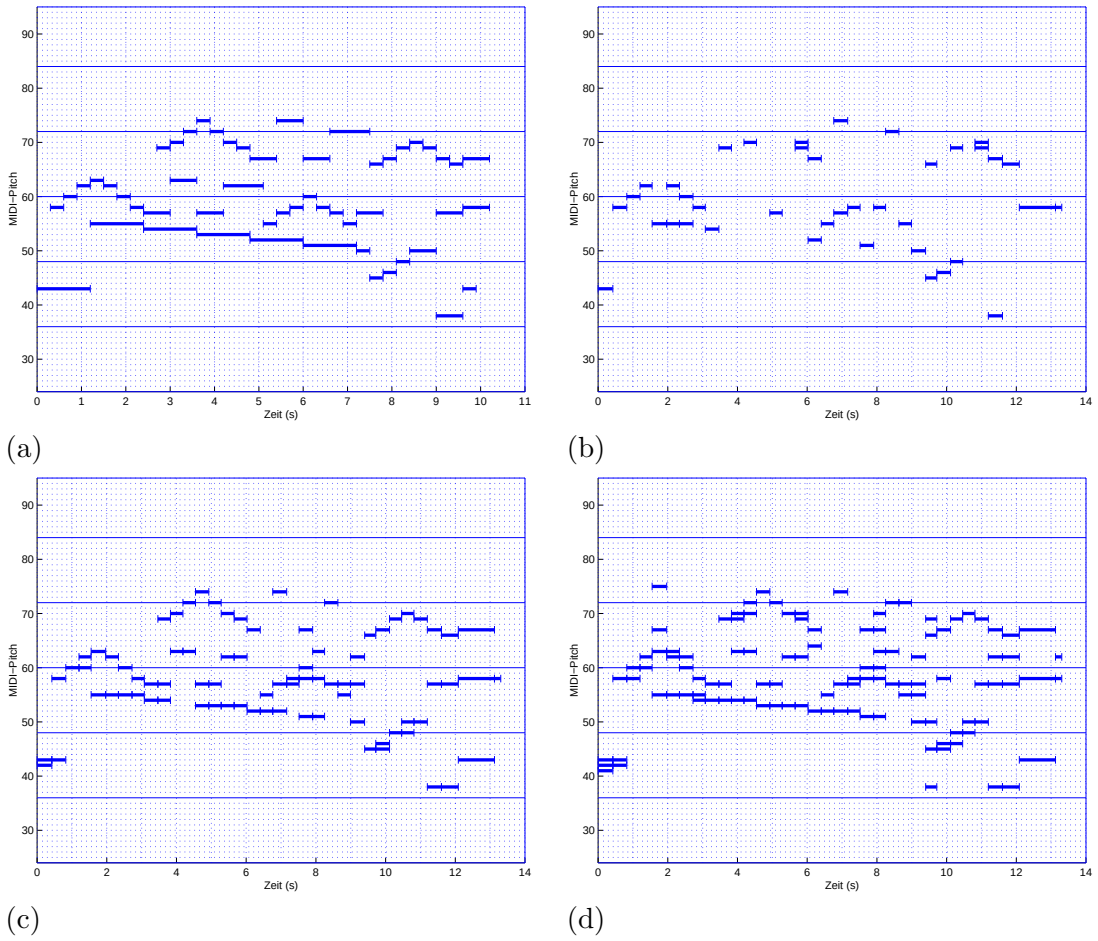
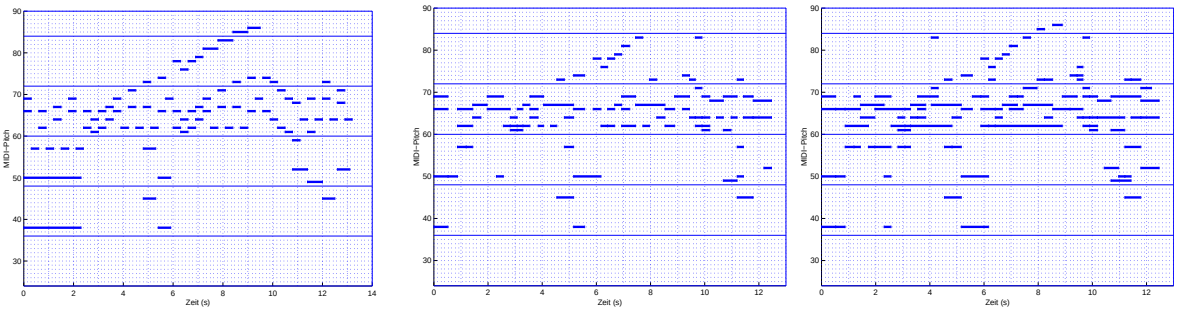


Abbildung 4.18: Klavierwalzendarstellung des dritten Beispiels (M = 3) aus der Tabelle 4.2: (a) MIDI-Datei, (b) die extrahierten Ergebnisse für $b = 0.3$, (c) die extrahierten Ergebnisse für $b = 0.1$, (d) die extrahierten Ergebnisse für $b = 0.04$. Die Notendauern wurden dabei nicht geschätzt.

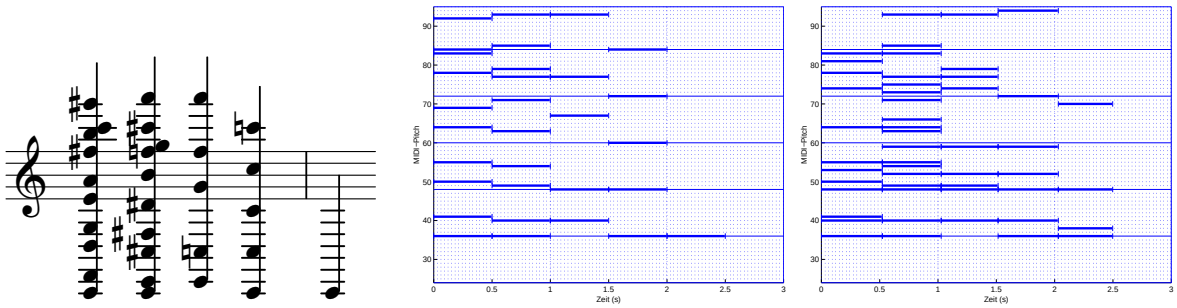
(Musikstück $M = 5$) für Parameterwerte $b = 0.1$ und $b = 0.04$ dargestellt worden. Die Verwendung der Tonhöhenschablonen eines Steinway-Flügels hat sich nur im dritten Beispiel für $b = 0.1$ als sinvoller herausgestellt. Auch andere, hier nicht aufgeführte Experimente zeigen ähnliche Ergebnisse. So können wir schliessen, dass die E-Piano-Schablonen für unsere aus verschiedenen Quellen gewonnenen Testdaten gut verwendbar sind.

Schließlich wollen wir auf das sechste Beispiel aus der Tabelle 4.2, das speziell für das Testen von Tonhöhenextraktionsverfahren konstruiert wurde, kurz eingehen. Dieses Beispiel wurde mit Hilfe des Notationsprogramms *Capella* konstruiert und in eine MIDI-Datei umgewandelt, deren Wiedergabe dann aufgezeichnet wurde. Das Teststück enthält 4 Akkorde und endet mit einem einzigen Ton. Die ersten zwei Akkorde bestehen aus 10 Tönen, die insgesamt fünf Oktaven umfassen wobei je zwei Töne zu einer Oktave gehören. Die weiteren zwei Akkorde bestehen aus jeweils fünf Tönen, wobei einer der Akkorde nur Töne von C_3 bis C_8, die miteinander in einem Oktavverhältnis stehen, enthält. Es ist anzunehmen, dass dieses Beispiel konstruiert und insbesondere von einem einzelnen Pianisten gar nicht spielbar ist (siehe Abb.



(a) (b) (c)

Abbildung 4.19: Klavierwalzendarstellung des fünften Beispiels (M = 5) aus der Tabelle 4.2: (a) MIDI-Datei, (b) die extrahierten Ergebnisse für $b = 0.1$, (c) die extrahierten Ergebnisse für $b = 0.04$.



(a) (b) (c)

Abbildung 4.20: Visualisierung des sechsten Beispiels (M = 6) aus der Tabelle 4.2: (a) Partiturdarstellung, (b) Klavierwalzendarstellung der erzeugten MIDI-Datei, (c) die extrahierten Ergebnisse für $b = 0.01$.

4.20). Die Tonhöhenextraktionsergebisse sind in Tabelle 4.4 dargestellt, wobei in den Zeilen 1 bis 5 die Ergebnisse für die einzelnen zeitlichen Intervalle angegeben sind. Aus der Tabelle geht hervor, dass die Extraktion des zweiten Akkords die erfolgreichste war: es wurden insgesamt neun Töne richtig und der zehnte als Oktavfehler erkannt. Für den vierten Akkord, bei dem alle fünf Töne im Oktavverhältnis miteinander stehen, wurden nur drei Töne richtig erkannt. Bei den Experimenten, bei denen alle vier Akkorde getrennt voneinander aufgezeichnet wurden (Zeilen 6 bis 9 der gleichen Tabelle) erhält man bezüglich der Anzahl der richtig extrahierten Töne ähnliche Ergebnisse. Da hier aber die Wirkung der nachklingenden Töne (und eventuell der Aliasing-Effekte der Filterung) fehlt, ist die Situation leicht anders. So wurden z. B. im Fall des isolierten vierten Akkords aus dem Beispiel 6 vier anstatt wie zuvor drei Töne erkannt.

Die letzten in Tabelle 4.4 aufgeführten Experimente wurden mit dem Parameterwert $b = 0.01$ durchgeführt und zeigen nur die Ergebnisse des Algorithmus für hoch komplexe künstlich erzeugte Akkorde. Im Fall von reellen Aufzeichnungen ist nicht zu erwarten, dass der Algorithmus solche guten Ergebnisse liefern wird. Man muss dabei insbesondere die maßgebende Rolle der Dynamik bei der Tonhöhenextraktion bedenken, die in echten Interpretationen wohl präsent ist, aber in diesem konstruierten Beispiel völlig ignoriert wurde.

Nr.	Akk.	Poly.	Partiturseite			extrahierte Daten			
			Korr.	O. F.	n. d.	Korr.	O. F.	n. T.	f.
1.	1	10	7	2	1	7	5	0	0
2.	2	10	9	1	0	9	9	0	0
3.	3	5	4	1	0	4	2	2	1
4.	4	5	3	2	0	3	0	3	1
5.	5	1	1	0	0	1	1	1	1
6.	1	10	7	1	2	7	4	0	0
7.	2	10	9	0	1	9	3	0	1
8.	3	5	4	0	1	4	1	0	3
9.	4	5	4	1	0	4	0	0	2

Tabelle 4.4: Ergebnisse des Tonhöhenextraktionsalgorithmus für die jeweiligen Akkorde des Musikstücks $M = 6$ aus der Tabelle 4.2. „Akk.“ steht für Akkordnummer (?), „Poly“ für die Anzahl der gleichzeitig einsetzenden Töne, „Korr.“ für korrekt extrahierte Töne, „O. F.“ für Oktavfehler, „n. d.“ für nicht detektierte Töne, „n. T.“ für nachklingende Töne und Oktavfehler, und schließlich „f.“ für falsch extrahierte Töne oder andere Intervallfehler, die aber keine Oktavfehler darstellen. Die Angaben bezeichnen hier die genauen Anzahlen von Tönen.

4.4 Diskussion des Gesamtsystems

Das in dieser Arbeit entwickelte Extraktionsverfahren ist in Abbildung 4.21 schematisch dargestellt. Die meisten Module des Systems wurden bereits in den vorherigen Abschnitten beschrieben. Wir werden hier kurz auf die noch unbesprochenen Module (in der Abbildung mit gestrichelten Linien dargestellt) eingehen.

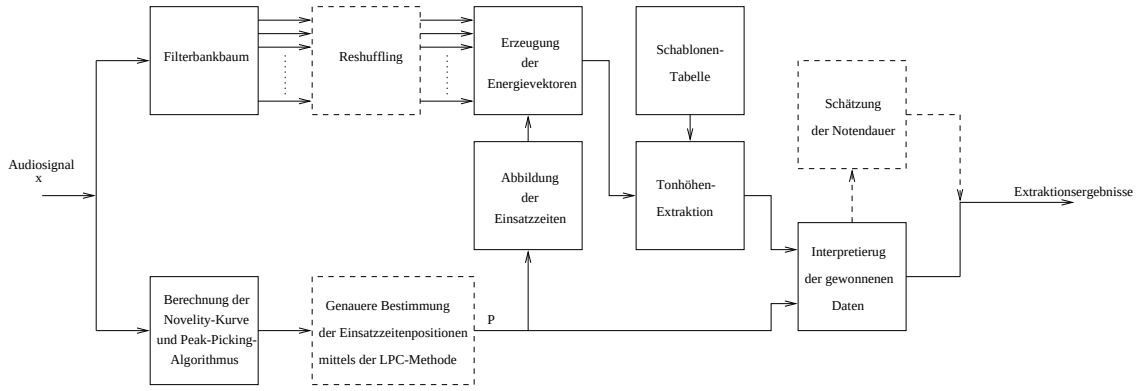


Abbildung 4.21: Schematische Darstellung des Gesamtsystems.

Wegen der Fensterung des Signals bei der Berechnung der Novelity-Kurve, repräsentiert (in unserem Fall) jeder Punkt dieser Kurve genau $N/2$ Abtastwerte dieses Signals. Für die als optimal ermittelte Fensterlänge von $N = 46.4$ ms würde jeder solche Punkt aus der Kurve einem $N/2 = 23.3$ ms langen Signalabschnitt entsprechen. Da im Fall der Audiosignale solch ein Wert von beträchtlicher Größe ist, ist eine feinere, genauere Schätzung der Einsatzzeitenpositionen erwünscht. Diese haben wir lokal mit Hilfe der LPC-Methode durchgeführt und so geschätzte Einsatzzeiten $p \in \{p_1, \ldots, p_P\} = \mathrm{supp}(P)$ erhalten. So können wir im besten Fall,

wenn die Einsatzzeit tatsächlich innerhalb der Segmentgrenzen liegt, diese mit einer Genauigkeit von 5.8 ms (Länge des hier verwendeten Fensters) schätzen. Falls die Einsatzzeit jedoch in der unmittelbaren Nähe der linken Segmentgrenze liegt, tauchen Fehler auf, die zwischen 5.8 und 23.2 ms liegen können.

Ein weiteres bisher nicht besprochenes Problem ist das Phänomen des *Shufflings* von Subbandsignalen, das wegen der Dezimierung des Signals im Ausgang des Hochpass-Filters in einer 2-Band Filterbank auftritt. Die Dezimierung wirkt auf das Hochpass-Signal so, dass sie es im unteren Frequenzbereich verschiebt und gleichzeitig die Frequenzachse invertiert (vgl. [16]). Filtern wir nun dieses Hochpass-Signal wieder mit einer 2-Band Filterbank, erhalten wir im Ausgang der Filterbank zwei Subbandsignale in umgekehrter Reihenfolge – das Tiefpass-Filter ergibt das Signal, das in Wirklichkeit das Hochpass-Signal repräsentiert, während im Ausgang des Hochpass-Filters das Tiefpass-Signal auftritt. Wegen dieses Phänomens entspricht die Reihenfolge der Subbänder im Ausgang des kaskadierten Multiraten-Filterbankbaums nicht mehr der natürlichen, aufsteigenden Ordnung. Deshalb muss vor der weiteren Bearbeitung eine „Neuordnung“ der Subbandsignale (Modul „Reshuffling“ in der Abbildung) durchgeführt werden.

Der Tonhöhenextraktionsalgorithmus, den wir in diesem Kapitel vorgestellt haben, bestimmt mit einer eingeschränkten Genauigkeit (siehe Unterabschnitt 4.3.4), für jedes Zeitintervall zwischen zwei geschätzten Einsatzzeiten die Tonhöhen der Noten, die Träger eines gewissen Anteils der Intervallenergie sind. Dabei wird angenommen, dass die extrahierte Tonhöhe im gesamten Intervall präsent ist, was jedoch in Wirklichkeit nicht unbedingt der Fall ist. Um die tatsächliche Dauer eines Tons zu bestimmen, müsste man z. B. den Grundton und die Obertöne dieses Musiktons „verfolgen“ können, was aber nur im Fall monophoner Musik (unter gewissen Einschränkungen) erreicht werden kann. In monophoner Musik kann eine Note (lt. Partitur) höchstens bis zum Einsatzzeitpunkt einer anderen Note dauern. In Wirklichkeit kann es aber passieren, dass eine Note für eine gewisse Abklingzeit nachklingt, die u. a. von ihrer Tonhöhe abhängig ist (siehe Unterabschnitt 2.7.4). Neben den unterschiedlichen Abklingzeiten der Töne treten bei polyphoner Musik auch andere der in Kapitel 2 behandelten Phänomene auf, die die Bestimmung von Tondauern erschweren. Zwei von diesen sind die Überlagerung der Partialtöne und die Amplitudenmodulation.

Im Rahmen dieser Arbeit haben wir einige einfachere Lösungen getestet, mittels derer aus den bereits extrahierten Parametern – Einsatzzeiten, Tonhöhen und Energien der Töne – die Notendauern grob bestimmt werden können. Nach der Berechnung des Quotienten zwischen der Tonenergie und Intervallänge, werden verbunden:

1. alle hintereinander vorkommenden Töne gleicher Tonhöhe, bei denen dieser Quotient im Vergleich zu dem ersten Ton aus der „Kette“, abnimmt,
2. alle hintereinander vorkommenden Töne gleicher Tonhöhe, bei denen dieser Quotient im Vergleich zu dem Ton im vorhergehenden Intervall abnimmt.

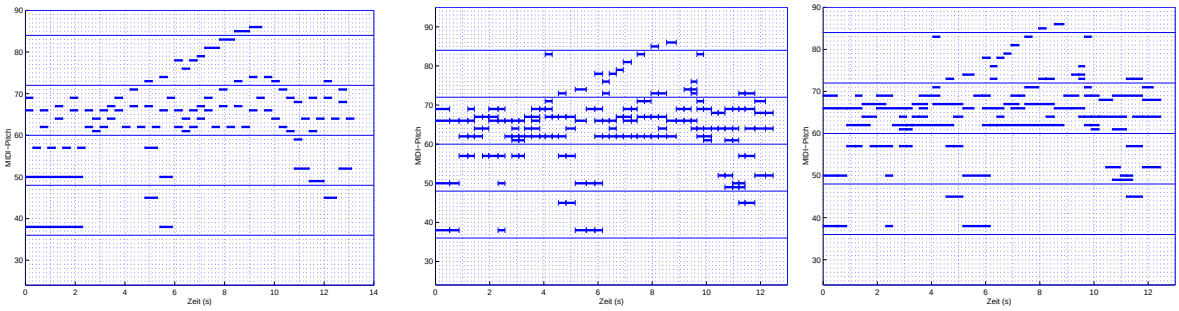
Diese Lösungen liefern nur teilweise gute Ergebnisse. Es passiert oft, dass neu hinzukommende Töne nicht als solche erkannt werden, und andererseits, dass die „Kette“ der tatsächlich nachklingenden Töne vorzeitig unterbrochen wird und dadurch ein nachklingender Ton als ein neugespielter Ton interpretiert wird. Der Fall eines kurzzeitig nichtdetektierten nachklingenden Tons kann mit dieser Methode (und vermutlich mit anderen Methoden auch) nicht

erkannt werden. Wir haben hier auch mit einer Modellierung der Abklingzeiten experimentiert, die die Abhängigkeit dieser von der Tonhöhe berücksichtigt (vgl. [26]). Dieses Modell wurde dann dazu eingesetzt, um die Abklingkurve der Töne (approximativ) zu modellieren und dadurch einen von Tonhöhe und Distanz abhängigen Energieschwellwert zu bestimmen, der bei der Bestimmung von Notendauern als Kriterium eingesetzt wird. Die Methode verbindet zwar eine große Anzahl von tatsächlich nachklingenden Tönen, verursacht aber dabei auch die oben erwähnten Fehler. In der ersten Zeile der unten angegebenen Tabelle sind für das fünfte Musikstück aus Tabelle 4.2 die Ergebnisse des Tonhöhenextraktionsalgorithmus gegeben (hier in Form von Anzahlen von korrekt detektierten Tönen, Anzahlen von Oktavfehlern usw.) In der zweiten Zeile sind die analogen Angaben, die aus der Verwendung der Methode zur Tondauerndetektion resultieren, dargestellt. Man kann deutlich erkennen, wie die Anzahl der nachklingenden Töne (n. T.) von 68 auf nur 3 gesenkt wird, jedoch auf Kosten der korrekt erkannten Töne. Es wurden nämlich 20 richtig erkannte Töne als nachklingende Töne eingeordnet. Zwei von diesen wurden dann als Oktavfehler interpretiert.

Nr.	M.	b	Sch.	Partiturseite				extrahierte Daten				
				Korr.	O. F.	n. d.	s. d.	Korr.	O. F.	s. d.	n. T.	f.
1.	5	0.04	EP	82	4	3	1	82	8	1	68	9
2.	5	0.04	EP	62	6	21	1	62	7	1	3	7

Tabelle 4.5: Ergebnisse des Tonhöhenextraktionsalgorithmus für das fünfte Musikstück ($M = 5$) aus Tabelle 4.2, gewonnen durch $b = 0.04$ und E-Piano Schablonen. Die erste Zeile der Tabelle enthält die „reinen“ extrahierten Daten. Die zweite Zeile enthält die nach Einsetzung der Tondauerndetektionsmethode erhaltenen Daten.

Eine präzisere Modellierung von Abklingzeiten und Abklingkurven der Töne könnte vielleicht die Ergebnisse weiter verbessern. Auf jeden Fall bleibt aber die Leistungsfähigkeit einer solchen Methode stark von den Ergebnissen der Einsatzzeiten- und Tonhöhenextraktionsalgorithmen abhängig. In Abb. 4.22 (a) ist die Klavierwalzendarstellung von einer zu diesem Musikstück zugehörigen MIDI-Datei gezeigt und neben ihr die Klavierwalzendarstellung der Extraktionsergebnisse für $b = 0.04$ ohne Tondauerndetektion (Abb. 4.22 (b)) bzw. mit Tondauerndetektion (Abb. 4.22 (c)).



(a) (b) (c)

Abbildung 4.22: Klavierwalzendarstellung des fünften Beispiels (M = 5) aus Tabelle 4.2: (a) MIDI-Datei, (b) die extrahierten Ergebnisse für $b = 0.04$ ohne Tondauerndetektion, (c) die extrahierten Ergebnisse für $b = 0.04$ mit Tondauerndetektion.

Die durch das hier entwickelte System gewonnenen Extraktionsergebnisse stellen, trotz der hier berichteten Probleme, eine gute Grundlage für die Ausführung der Synchronisation von Klavieraufzeichnungen mit den zugehörigen Dateien in einem partiturähnlichen Format (siehe Kapitel 5) dar. Die Aufgabe einer genaueren Extraktion von Tonhöhen und Tondauern, die in Richtung einer Lösung des Transkriptionsproblems führten, haben wir hier nicht weiter verfolgt. Die Verbesserungen des Systems sowohl was die Qualität der Extraktionsergebnisse als auch die Steigerung der Recheneffizienz (insbesondere bei der Filterung des Signals) betrifft, wäre hier die nächste Herausforderung. Dazu gehören weitere intensive Tests sowie die Entwicklung eines Moduls für die automatische Bewertung der Extraktionsergebnisse. Eine mögliche Lösung für die zweite Aufgabe könnte durch Kombination von Partiturdaten und des im nächsten Kapitel beschriebenen Synchronisationsverfahrens erzielt werden.

Kapitel 5

Synchronisation

Zur Einführung in die Thematik dieses Kapitels gehen wir von einer digitalen Musikbibliothek aus, in der Dokumentensammlungen von Musikstücken in unterschiedlichsten Ausprägungen und Formaten vorliegen. Zu den gängigsten Formaten zählen partiturnahe Formate wie Capella oder Score, das MIDI-Format sowie diverse (un-)komprimierte Wellenformdarstellungen wie WAV oder MP3. Es gibt zahlreiche Gründe, die in verschiedenen Formaten und in unterschiedlichen Interpretationen vorliegenden Versionen eines Musikstücks miteinander zu verlinken. Dies kann einerseits ein grobes Verlinken sein, das nur einen Hinweis darauf gibt, dass z. B. dieses Stück auch noch in anderen Interpretationen vorliegt, andererseits – und das ist die Thematik des vorliegenden Kapitels – kann man unter Verlinkung eine recht genaue zeitliche Synchronisation der Varianten des Stücks verstehen. Wir kommen kurz auf mögliche Anwendungen einer solchen Synchronisation zu sprechen:

- Musikwissenschaftler können auf der Basis solcher Synchronisationen u. a. vergleichende Tempostudien über verschiedene Interpretationen durchführen.
- Die zeitliche Verlinkung von Partitur und Interpretation ermöglicht eine Partiturlesehilfe beim Verfolgen der Interpretation, indem die aktuellen Noten hervorgehoben werden.
- Bei der partiturbasierten Suche nach Musikpassagen kann man im Erfolgsfall die Passage in einer geeigneten oder genehmen Interpretation direkt anspringen, um so eine qualitativ hochwertige Kostprobe zu bekommen.

Bevor wir Synchronisationsalgorithmen vorstellen, haben wir vorab eine genauere Problemspezifikation vorzunehmen. Zu diesem Zweck gehen wir auf die Begriffe Tempo, Zeitfluss und Synchronisation in Anlehnung an [55] nochmals genauer ein. Danach formulieren wir die Synchronisationsaufgabe.

Wenn am Anfang eines Musikstücks in der Partitur z. B. “M.M. Viertelnote = 100” steht, so handelt es sich hier um die Spielanweisung, dass das Stück zunächst mit einem gleichmäßigen Tempo von 100 Viertelnoten pro Minute zu spielen ist. Statt “Viertelnote” könnte man auch andere Grundeinheiten wie z. B. “Achtelnote” oder “Halbe Note” zugrunde legen. Allgemein spezifizieren wir ein *gleichmäßiges Tempo* T durch die Formel:

$$T = \frac{\#\text{Pulse}}{\text{Minute}} \ .$$

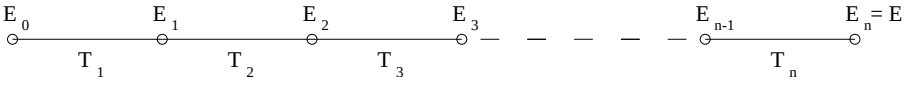
Auch ohne derartige Tempoangaben kann man jeder metrischen Position innerhalb eines beliebigen Taktes eines Musikstücks eine *musikalische Einsatzzeit* zuordnen: Gibt es pro Takt p Pulse und handelt es sich bei der in Rede stehenden Position um den k-ten Puls im s-ten Takt, so hat diese Position die *musikalische Einsatzzeit*

$$E = p \cdot (s-1) + k.$$

Sind nun E_0 und E musikalische Einsatzzeiten mit $E \geq E_0$, so ergibt sich bei gleichmäßigem Tempo T die zu E gehörige *physikalische Einsatzzeit* $e(E)$ relativ zur physikalischen Einsatzzeit von E_0 durch die Formel:

$$e(E) = e(E_0) + \frac{E - E_0}{T}. \tag{5.1}$$

In der Realität wird selten durchweg ein gleichmäßiges Tempo gespielt werden. Leichte Temposchwankungen sowie von der Partitur geforderte Tempoänderungen sind zu berücksichtigen. Realitätsnäher ist der Fall des stückweise gleichmäßigen Tempos. Nehmen wir an, dass $E_0 < E_1 < \ldots < E_n = E$ musikalische Einsatzzeiten sind, so dass das Tempo zwischen E_{i-1} und E_i gleichmäßig gleich T_i ist:



Dann ergibt sich die physikalische Einsatzzeit von E durch die Formel:

$$e(E) = e(E_0) + \sum_{i=1}^{n} \frac{E_i - E_{i-1}}{T_i}.$$

Bei immer feiner werdender Unterteilung kann man $\sum_{i=1}^{n}(E_i - E_{i-1})/T_i$ als Riemann-Summe auffassen. Dabei ist T_i letztlich als momentanes Tempo zu interpretieren. Unter geeigneten Annahmen (strenge Monotonie sowie stetige Differenzierbarkeit der Funktion der physikalischen Einsatzzeiten) ergibt sich dann der *Zeitfluss* zum musikalischen Zeitpunkt $E \geq E_0$ auf der Basis der Tempofunktion T zu

$$e(E) = e(E_0) + \int_{E_0}^{E} \frac{dx}{T(x)}.$$

Demnach ist das *Tempo* in E bei gegebenem Zeitfluss e der Kehrwert

$$T(E) = \left(\frac{de}{dE}(E)\right)^{-1}.$$

Wir illustrieren die gerade eingeführten Begriffe an zwei typischen Beispielen. Falls $T(x) = T$ konstant ist, liegt der Fall des gleichmäßigen Tempos vor und wir erhalten wie oben die Formel $e(E) = e(E_0) + (E - E_0)/T$. Ist andererseits $T(x) = \alpha \cdot (x - E_0) + T_0$ (für ein $\alpha \neq 0$, eine Konstante $T_0 > 0$ und alle $x \in [E_0, E]$), so ist

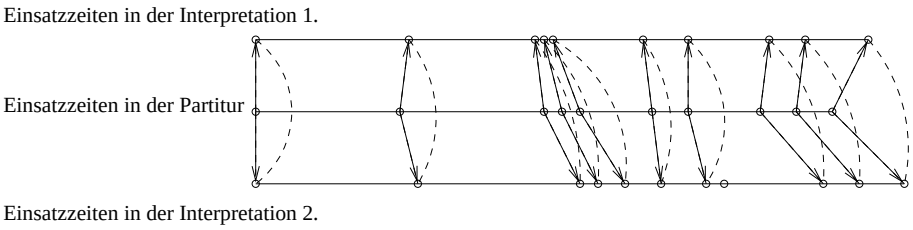
$$e(E) = e(E_0) + \frac{1}{\alpha} \ln \frac{\alpha(E - E_0) + T_0}{T_0}.$$

Das heißt, für $\alpha > 0$ liegt ab der musikalischen Einsatzzeit E_0 ein gleichmäßiges *Accelerando*, also eine Tempobeschleunigung vor, während sich für $\alpha < 0$ ein gleichmäßiges *Ritardando* ergibt.

Sind nun die Tempofunktionen T_1 und T_2 zweier Interpretationen eines Musikstücks bekannt, so führt dies auf zwei Zeitflüsse

$$e_1(E) = e_1(E_0) + \int_{E_0}^{E} \frac{dx}{T_1(x)} \quad \text{und} \quad e_2(E) = e_2(E_0) + \int_{E_0}^{E} \frac{dx}{T_2(x)}.$$

Die (ideale) *Synchronisation* dieser beiden Interpretationen besteht nun darin, für jede musikalische Einsatzzeit E der Partitur den Zeitpunkt $e_1(E)$ der ersten mit dem Zeitpunkt $e_2(E)$ der zweiten Interpretation zu verlinken. Dies wird in folgendem Schaubild illustriert.



Die Praxis ist leider sehr weit von dieser idealisierten Sichtweise entfernt. Dies beginnt schon mit den oft nur vage gegebenen Tempoangaben in einer Partitur. Nicht selten ist anstelle eines exakten Tempos z. B. nur “Allegro” angegeben, wodurch nur eine grobe Tempovorgabe gemacht wird. Des weiteren werden (positive oder negative) Tempobeschleunigungen nur angedeutet, indem die Stelle oder manchmal der Bereich, in dem das Tempo beschleunigt werden soll, markiert ist. Der genauere Verlauf des Tempos wird nicht spezifiziert und ist damit stark interpretationsabhängig. Folglich gibt es i. A. keine kanonische Tempofunktion, die man einer Partitur zuordnen könnte. Jede Tempofunktion basiert stets auf einer konkreten Interpretation der Partitur.

Vor diesem Hintergrund können wir nun verschiedene Synchronisationsaufgaben spezifizieren. Die Aufgaben sind nach steigendem Schwierigkeitsgrad aufgelistet.

1. Partitur und MIDI-Einspielung seien gegeben. Hier läuft die Synchronisationsaufgabe auf die Berechnung der Tempofunktion der MIDI-Einspielung hinaus. Diese Aufgabe ist vergleichsweise einfach, da die Extraktion der Notenereignisse aus der MIDI-Datei keine Hürde darstellt. Trotzdem werden wir auch bei dieser Aufgabe auf heikle Probleme stoßen. Diese werden vor allem dadurch verursacht, dass eine Partitur an manchen Stellen nur vage Spielanleitungen gibt, wie etwa im Fall von barocken Verzierungen. Mit dem Konzept der Fuzzy-Note werden wir diese Vagheit in zufriedenstellender Weise in den Griff bekommen.

2. Partitur und WAV-Einspielung seien gegeben. Die Synchronisationsaufgabe besteht hier in der Berechnung der Tempofunktion der WAV-Einspielung. Diese Aufgabe ist schwieriger als die erste, da die Extraktion der Notenereignisse aus der WAV-Datei, wie das

letzte Kapitel gezeigt hat, erhebliche Probleme macht und auch zu falschen Extraktionsresultaten führt. Diese fehlerbehaftete Grundlage macht ein kostenoptimales imperfektes Matching notwendig. Durch den Einsatz geeigneter Kostenfunktionen wird versucht, den analysebedingten Extraktionsfehlern entgegenzuwirken.

3. Partitur sowie MIDI- und WAV-Einspielung seien gegeben. MIDI- und WAV-Interpretation sind zu synchronisieren. Dies wird durch Kombination der Lösungen der beiden ersten Aufgaben erreicht.

4. Es seien nur MIDI- und WAV-Interpretation gegeben. Partiturkenntnisse liegen nicht vor oder sollen nicht benutzt werden. Bei dieser Aufgabe fehlt die Partitur als "sichere" Referenz. Hier nimmt MIDI provisorisch die Stelle der Partitur ein. Das weitere Vorgehen ist ähnlich wie bei der zweiten Aufgabe; wesentlicher Unterschied ist allerdings das Fehlen der musikalischen Einsatzzeiten.

Bevor wir in den folgenden Abschnitten der Reihe nach auf die einzelnen Aufgabenstellungen eingehen, geben wir im nächsten Abschnitt einen Kurzüberblick über den Stand der Forschung.

5.1 Stand der Forschung zur Synchronisation in der Musik

Es gibt eine Reihe von Synchronisationsaufgaben in der Musik, die schon seit geraumer Zeit bearbeitet werden, die aber mehr oder weniger stark von der in dieser Arbeit zugrundeliegenden Zielsetzung abweichen. Da teilweise ähnliche Ideen zur Lösung dieser Aufgaben benutzt wurden, stellen wir kurz einige der Arbeiten vor.

Beim Problem der *automatischen Begleitung einer Interpretation* geht es darum, auf Partiturbasis ein System zu entwickeln, das in der Lage ist, die Interpretation einer Solostimme der Partitur in Echtzeit (d. h. mit einer vertretbaren Zeitverzögerung von höchstens 40 ms) zu begleiten. Ein erster einfacher Begleiter wurde von Vercoe in [90] vorgestellt. Konkret geht es um die Begleitung eines solistischen Querflötenparts. Da allerdings die Tonhöhenextraktion in Echtzeit Probleme bereitet, arbeitet Vercoe mit Sensoren an der Flöte. Dadurch wird das Tonhöhenextraktionsproblem wesentlich vereinfacht. Die Synchronisation geschieht durch ein Mitverfolgen der Flötenstimme.

Ein weiteres Verfahren zur automatischen Begleitung wurde von *Dannenberg* [17] vorgestellt. Dannenberg geht von einer einstimmigen Solostimme und von einstimmiger Begleitung aus. Das Synchronisationsproblem reduziert er auf ein LCS-Problem (longest common subsequence), das er mittels gefensterter dynamischer Programmierung (suboptimal) löst. Das gesamte Verfahren wurde auch für Trompetensolo auf einem 8-bit Microcomputer System implementiert, wobei die Tonhöhe in Echtzeit detektiert wird.

In einer späteren Arbeit von *Dannenberg* und *Mukaino* (vgl. [19]) wurde dieses System auf polyphone Musik erweitert. Die neue Version erlaubt auch die Behandlung von Verzierungen und Glissandi. Gleichzeitig wurde das System so erweitert, dass in den kritischen, d. h. unklaren Stellen, so lange mehrere Matching-Hypothesen verfolgt werden, bis eine sinnvolle Entscheidung getroffen werden kann. Die auch in dieser Arbeit verwendete Idee, modulo Oktaven zu rechnen, wurde dort zur Effizienzsteigerung eingeführt. Das Verfahren wurde mit

Daten aus einem MIDI-fähigen Keyboard getestet. Allerdings fehlen konkrete Angaben über die Matching-Ergebnisse. *Grubb* und *Dannenberg* haben 1996 ein ähnliches Verfahren für automatische Begleitung im Fall mehrerer Musikinstrumente patentiert (vgl. [18]), allerdings fehlt noch die Realisierung eines Prototypen. Dafür aber ist Voraussetzung, dass für jedes Instrument das Signal separat vorliegt.

Raphael hat einen automatischen Begleiter für Oboe, was wieder ein einstimmiges Soloinstrument ist, entwickelt. Für die Erkennung der Noten verwendet er Hidden Markov Modelle (HMM) (vgl. [72]) und für das Matching zur Begleitungssteuerung Bayes-Netze (vgl. [73]). Der automatische Begleiter kann durch Training hinsichtlich Prädiktion verbessert werden.

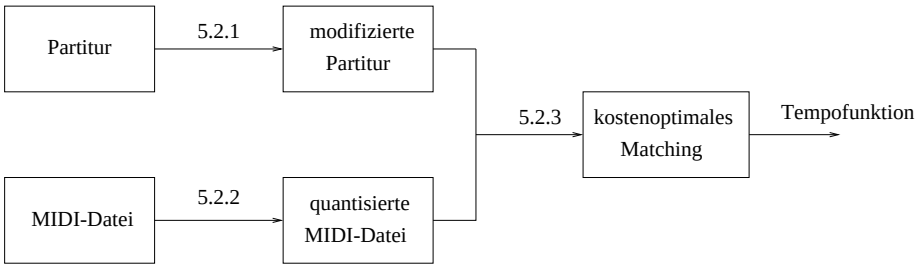
Eine weitere Methode zur automatischen Begleitung von Gesang wurde in [8] vorgestellt. Auch diese verwendet genauso wie das oben beschriebene Verfahren HMMs für die Modellierung der Töne. Die Notenmodelle bestehen aus drei Zuständen (je einen für die Anstieg-, Halte- und Abstiegphase). Neben den Notenmodellen gibt es auch Modelle für Rauschen und Pausen. Eine Nächste-Nachbarsuche der gewonnenen Feature-Vektoren mit den vorberechneten Vektoren aus dem Kodebuch führt dann zur Tonhöhenbestimmung. Leider enthält der Artikel keine Angaben, wie das Matching vorgenommen werden soll.

Der von *Honing* entwickelte *Strict-Matcher* (vgl. [24, 37]) behandelt das Problem der Synchronisation von Partitur- und MIDI-Daten. Der Algorithmus arbeitet mit aktuellen Zeitfenstern in der Partitur und versucht die Noten innerhalb des Fensters mit den gerade gespielten Noten der MIDI-Interpretation unter strikter Wahrung der zeitlichen Reihenfolge zu matchen. Der Algorithmus ist sehr anfällig gegenüber leichten Spielfehlern.

Large stellt in [48] ein Verfahren vor, das es erlaubt, die Verspieler bei Interpretationen zu lokalisieren und ggf. zu interpretieren. Large arbeitet mit einer Kostentabelle, die sowohl von den (einstimmigen!) Partiturdaten als auch von den MIDI-Daten abhängt. Anhand dieser Tabelle ermittelt er dann mittels dynamischer Programmierung einen besten Match und analysiert die Art der aufgetretenen Fehler.

5.2 Synchronisation von Partitur und MIDI

Das folgende Schaubild gibt einen ersten Überblick über diesen Abschnitt. In Unterabschnitt 5.2.1 wird die Partitur insbesondere in Bezug auf die Behandlung von Sondernoten (Triller, Vorschlag, Arpeggio) vorverarbeitet. Unterabschnitt 5.2.2 geht auf die zeitliche Quantisierung von MIDI ein. Nach diesen Vorverarbeitungsschritten stellen wir in Unterabschnitt 5.2.3 ein Kostenmaß vor, anhand dessen mittels dynamischer Programmierung ein kostenminimales Matching berechnet wird, das zur Synchronisation zwischen Partitur und MIDI führt.

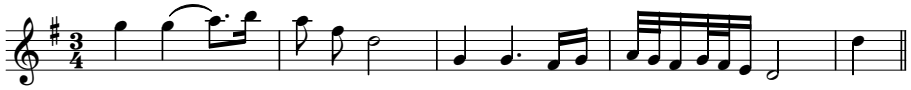


5.2.1 Partiturvorverarbeitung

Wir gehen davon aus, dass die gesamten Partiturinformationen elektronisch vorliegen. Unser erstes Ziel ist es, jedem Notenobjekt der Partitur eine musikalische Einsatzzeit zuzuordnen. Wir unterscheiden zwei Arten von Notenobjekten: explizite und implizite. Bei den expliziten Objekten sind alle Notenparameter vollständig gegeben. Die musikalische Einsatzzeit von expliziten Noten ist einfach anzugeben: Gibt es pro Takt p Pulse und ist die in Rede stehende Note zum k-ten "Puls" im s-ten Takt zu spielen, wobei k auch rational sein darf, so hat diese Note die *musikalische Einsatzzeit*

$$E = p \cdot (s-1) + k.$$

Beispiel 5.2.1 Das folgende Melodiefragment besteht nur aus expliziten Noten.



Wir geben zu dieser Notenfolge aus 19 Noten die entsprechende Folge $E_1, \ldots, E_{19}$ von musikalischen Einsatzzeiten an.

i-te Note	1	2	3	4	5	6	7	8	9	10	11
E_i	1	2	3	$3\frac{3}{4}$	4	$4\frac{1}{2}$	5	7	8	$9\frac{1}{2}$	$9\frac{3}{4}$

i-te Note	12	13	14	15	16	17	18	19
E_i	10	$10\frac{1}{8}$	$10\frac{1}{4}$	$10\frac{1}{2}$	$10\frac{5}{8}$	$10\frac{3}{4}$	11	13

□

Implizite Notenobjekte bestehen aus Noten oder Notengruppen, die mit Sonderzeichen wie Triller, Vorschlag oder Arpeggio versehen sind. Typischerweise handelt es sich hier nur um Abkürzungen für komplexere Notenkonstellationen, wie die Beispiele in Abb. 5.2 illustrieren. Diese Konstellationen sind also durch die Abkürzung nur implizit gegeben. Allerdings ist die Rückübersetzung dieser Abkürzung nicht immer eindeutig, sondern hängt u. a. von der Epoche oder vom Interpreten ab. Um diesen verschiedenen Alternativen gerecht zu werden, benutzen wir hier das Konzept der Fuzzy-Note. Eine *Fuzzy-Note* besteht aus einer endlichen Menge von alternativen Noten, die jeweils durch ihre Einsatzzeit, ihre Tonhöhe und Tondauer beschrieben sind. Ein implizites Notenobjekt, das neben den Sonderzeichen (Triller, Vorschlag, Arpeggio) durch eine oder mehrere Hauptnoten mit gleicher Einsatzzeit sowie gleicher Dauer beschrieben ist, wird nun ersetzt durch eine Fuzzy-Note: Diese übernimmt die Einsatzzeit und Dauer der Hauptnoten und zieht alle potenziellen Tonhöhen dieses Notenobjekts als Alternativen in Betracht. Anhand einiger Verzierungsformen verdeutlichen wir das prinzipielle Vorgehen.

Mordent ist eine Verzierungsart, die einen Wechsel mit der unteren Nebennote fordert. Abbildung 5.1 zeigt links das implizite Notenobjekt (Viertelnote d mit Mordent), rechts ein entsprechendes explizites Notenobjekt, in dem die Tonhöhen d und c vorkommen. Die zugehörige Fuzzy-Note besteht aus zwei Viertelnoten der Tonhöhen d und c mit gleicher Einsatzzeit.

Triller sind Verzierungsarten, die einen schnellen Wechsel einer Hauptnote mit der kleinen oder großen Ober- und (eventuell) Untersekunde auslösen. Die Abbildung 5.2 zeigt einige Trillerarten und mögliche Explikationen.

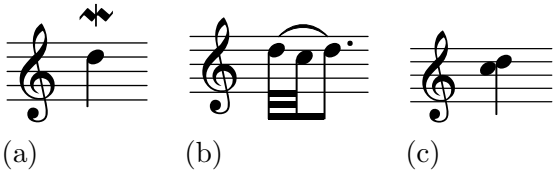


Abbildung 5.1: Mordent, (a) die Bezeichnung, (b) die Interpretationsart, (c) die entsprechende Fuzzy-Note.

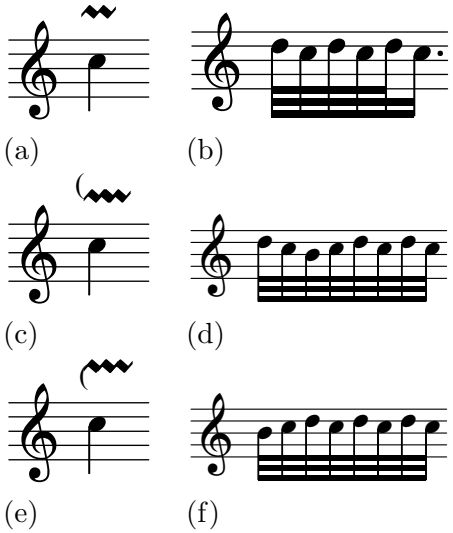


Abbildung 5.2: Triller.

Da die Aufführungspraxis gezeigt hat, dass oft auch sowohl die leitereigene Obersekunde wie auch die leitereigene Untersekunde beim Trillern benutzt wird, hat es sich als sinnvoll herausgestellt, bei allen Trillerformen grundsätzlich in der zugehörigen Fuzzy-Note neben der Tonhöhe der Hauptnote sowohl die Ober- als auch die Untersekunde mitzuberücksichtigen. Dies führt daher, bezogen auf obige Trillerbeispiele, zu folgender Fuzzy-Note:

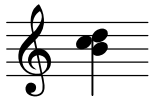


Abbildung 5.3: Triller – Fuzzy-Note.

Vorschlag ist die Bezeichnung für eine oder mehrere Verzierungsnoten, die einer Hauptnote vorausgehen. Der Vorschlag fällt entweder in die Zeit der Hauptnote oder in die Zeit der vorhergehenden Note. Des weiteren sind je nach Epoche und Aufführungspraxis lange und kurze Vorschläge zu unterscheiden. Abbildung 5.4 zeigt einige Vorschlagsbeispiele mit alternativ möglichen Explikationen.

Diese diffuse Situation mit den sehr unterschiedlichen Spielvarianten lösen wir auf, indem wir die Fuzzy-Note mit der Einsatzzeit und Dauer der Hauptnote versehen und die Tonhöhen von Haupt- und Vorschlagsnoten übernehmen. Bei der Synchronisation hat diese Festlegung den

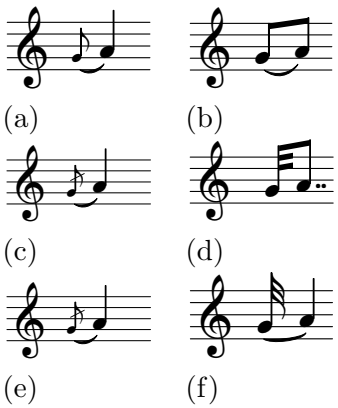


Abbildung 5.4: Vorschlag.

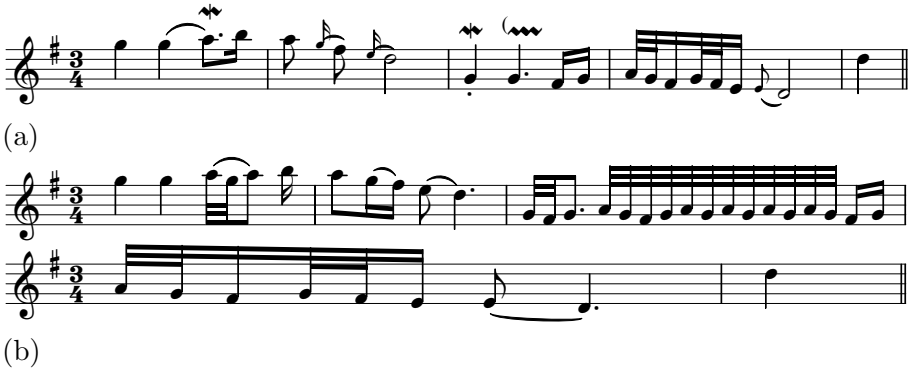


Abbildung 5.5: Beginn der *Aria con Variazioni* von J. S. Bach, BWV 988, (a) die ersten vier Takte der rechten Hand, (b) eine mögliche Expansion der Partitur.

Vorteil, dass sowohl in dem Fall, wo die Vorschlagsnote "auf Schlag" gespielt wird, als auch in dem Fall, wo die Hauptnote auf Schlag gespielt wird, die Voraussetzungen zur korrekten Synchronisation gegeben sind.

Mit **Arpeggio** bezeichnet man einen nach Harfenspielart gebrochenen Akkord, d. h. die Akkordtöne kommen der Reihe nach kurz hintereinander zum Einsatz und hören zum gleichen Zeitpunkt auf. Typischerweise spielt man die Töne des Akkords nach aufsteigender Tonhöhe. Es gibt aber auch die entgegengesetzte Spielweise. Die Übersetzung eines Arpeggios ist trivial: Das Arpeggiozeichen bleibt einfach unberücksichtigt und der gesamte Akkord wird zu einer Fuzzy-Note umgewandelt.

Wir illustrieren unser Vorgehen an der Originalfassung des obigen Beispiels. Es handelt sich um den Beginn der *Aria con Variazioni* von J. S. Bach, BWV 988. Wir zeigen in Abbildung 5.5 einerseits die ersten vier Takte der rechten Hand (inkl. der Verzierungszeichen) und präsentieren andererseits eine mögliche Expansion dieser Takte.

Abschließend zeigen wir die ersten drei Zeilen der *Aria con Variazioni* in konventioneller Notenschrift (Abbildung 5.6), dann in Klavierwalzendarstellung eine mögliche Explikation (Abbildung 5.7) und schließlich zum Vergleich die Klavierwalzendarstellung von der zur Synchronisation benutzten modifizierten Darstellung (Abbildung 5.8).

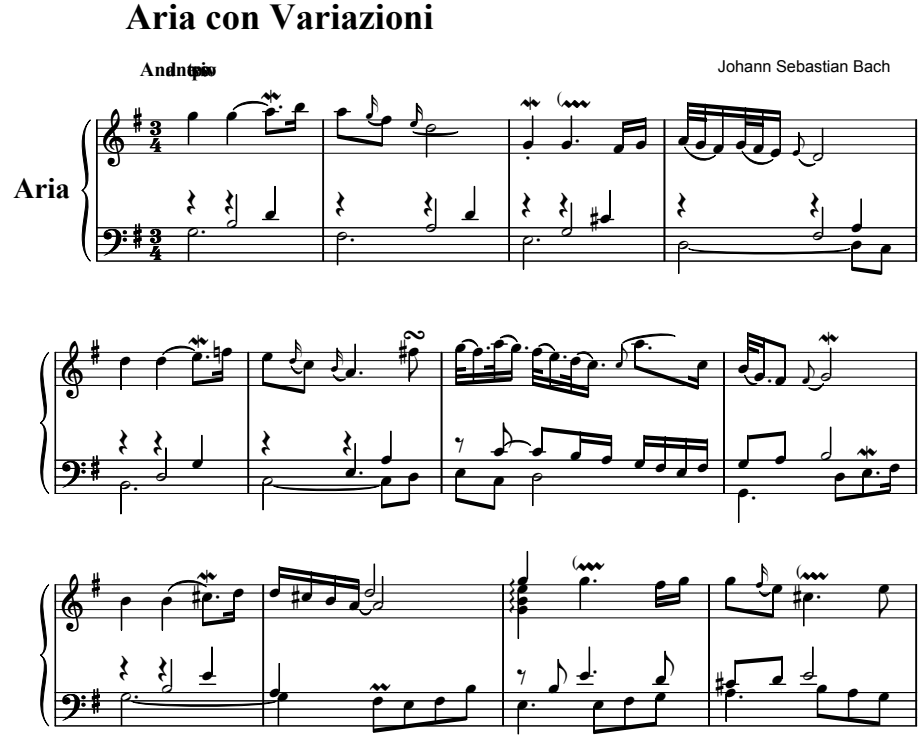


Abbildung 5.6: Drei ersten Zeilen der *Aria con Variazioni* von J. S. Bach, BWV 988.

Die wie oben beschriebene Explikation einer Partitur wollen wir im Folgenden als *modifizierte Partitur* bezeichnen. Aus mathematischer Sicht kann man eine solche modifizierte Partitur unter Vernachlässigung der Notendauern ansehen als endliche Menge P von Tripeln, die explizite sowie implizite Notenobjekte beschreiben. Ein explizites (bzw. implizites) Objekt wird dabei durch ein Tripel $(e, H, 0)$ (bzw. $(e, H, 1)$) angegeben, wobei $e \in \mathbb{Q}$ die musikalische Einsatzzeit ist und $\emptyset \neq H \subseteq [0:127]$ die Menge der Tonhöhen bezeichnet, die zum Zeitpunkt e einsetzen (expliziter Fall) bzw. als Alternative zum Einsatz bereit stehen (impliziter Fall). Anhand der letzten Komponente dieser Tripel kann man also implizite von expliziten Noten unterscheiden. Indem man die rationalen Einsatzzeiten mit Hilfe des Hauptnenners darstellt, kann man o. B. d. A. voraussetzen, dass alle musikalischen Einsatzzeiten sogar ganzzahlig sind. Insgesamt ist also die modifizierte Partitur beschreibbar als endliche Teilmenge

$$P \subset \mathbb{Z} \times 2^{[0:127]} \times \{0, 1\}.$$

Mit MIDI-Dokumenten kann man ähnlich verfahren. Da es aber in MIDI nur explizite Notenobjekte gibt, ist also ein MIDI-Dokument beschreibbar als endliche Teilmenge

$$M \subset \mathbb{Z} \times 2^{[0:127]} \times \{0\}.$$

Auf den ersten Blick könnte man aufgrund der beiden letzten Formelzeilen erwarten, dass wir jetzt unmittelbar mit dem Synchronisieren starten können. Da es aber bei MIDI aus verschiedenen Gründen Schwierigkeiten mit der *Gleichzeitigkeit* von Notenereignissen gibt, ist diese Gleichzeitigkeit durch geeignete Quantisierung nachträglich "wiederherzustellen". Damit beschäftigt sich der folgende Unterabschnitt.

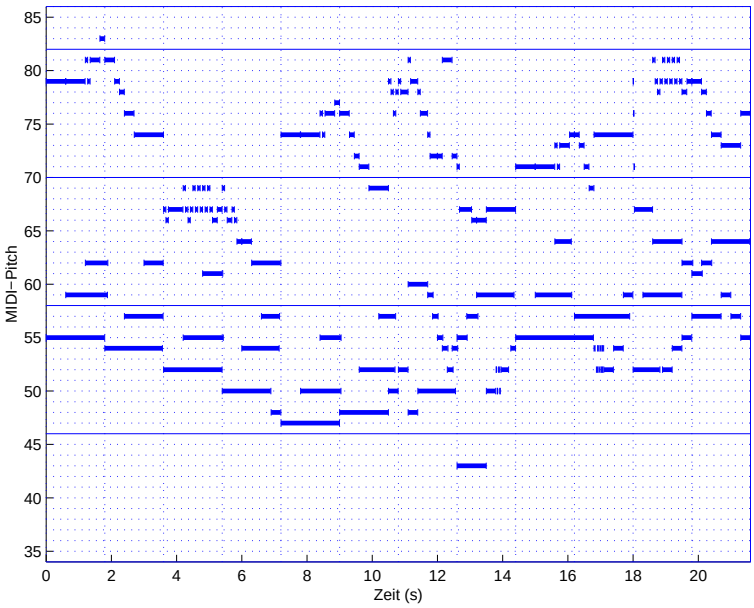


Abbildung 5.7: Klavierwalzendarstellung einer MIDI-Einspielung des Beginns der Aria.

5.2.2 Quantisierungen

Ein Großteil der MIDI-Dokumente basiert auf Einspielungen über ein MIDI-fähiges Keyboard. Das birgt gleich mehrere Quellen für Asynchronismen insbesondere bei Akkorden: Einerseits kann es sich um Einspielungsungenauigkeiten handeln, andererseits kann von MIDI bei einer derartigen Einspielung nur eine Note pro Einsatzzeit protokolliert werden. Daher ist vor der eigentlichen Synchronisation eine Zeitquantisierung erforderlich. Auf Zeitquantisierung gehen wir auch im Hinblick auf die später vorzunehmende Quantisierung bei PCM-Dokumenten nun zunächst in einem allgemeineren Rahmen ein.

Es sei $\Delta \geq 0$ eine fest vorgegebene Quantisierungskonstante. Jeder endlichen Teilmenge P von $\mathbb{Z} \times 2^{[0:127]} \times \{0,1\}$ ordnen wir eine Folge $Q_\Delta(P)$ von Tripeln (e, P_0, P_1) zu, wobei e eine quantisierte Einsatzzeit und P_0 bzw. P_1 eine Menge von expliziten bzw. impliziten Tonhöhen, die zum Zeitpunkt e beginnen, bezeichnet. Zur induktiven Definition dieser Folge benutzen wir die erste Projektion $\pi_1(P) := \{e \mid \exists H, t : (e, H, t) \in P\}$. Zunächst werden die quantisierten Einsatzzeiten eingeführt, indem wir mit der kleinsten Einsatzzeit

$$e_1 := \min \pi_1(P)$$

in P beginnen und für $i \geq 2$

$$e_i := \min[\pi_1(P) \cap (e_{i-1} + \Delta, \infty)]$$

setzen, sofern der Durchschnitt nicht leer ist. Ist der Durchschnitt für $i = p + 1$ zum ersten Mal leer, so definieren wir $e_{p+1} := \infty$. Diesen quantisierten Einsatzzeiten $e_1 < e_2 < \ldots < e_p$ ordnen wir jetzt Tonhöhenmengen $P_{01}, \ldots, P_{0p}$ bzw. $P_{11}, \ldots, P_{1p}$ zu durch die Festsetzung:

$$P_{0i} := \bigcup P_0 \quad \text{bzw.} \quad P_{1i} := \bigcup P_1,$$

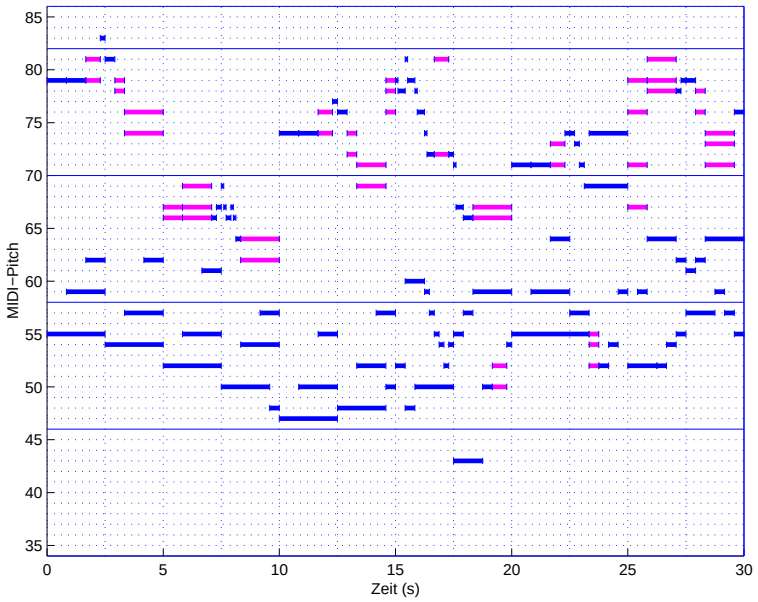


Abbildung 5.8: Klavierwalzendarstellung der in MIDI konvertierten modifizierten Partitur des Ariabeginns.

wobei die Vereinigung über alle P_0 bzw. P_1 gebildet wird, zu denen ein $(e, P_0, 0) \in P$ bzw. $(e, P_1, 1) \in P$ existiert, so dass $e \in [e_i, e_{i+1})$. Manche der Mengen P_{0i} oder P_{1j} können durchaus leer sein. Es ist aber stets $P_{0i} \cup P_{1i}$ nicht leer. Wir verabreden noch, dass die Quantisierungen auf die Gesamtdauer $\ell(P)$ von P keinen Einfluss haben sollen.

5.2.3 Synchronisation als Matching-Problem

Zur Synchronisation von Partitur und MIDI werden wir im Folgenden von der zur modifizierten Partitur gehörenden endlichen Teilmenge P von $\mathbb{Z} \times 2^{[0:127]} \times \{0, 1\}$ die 0-Quantisierung

$$Q_0(P) = [(p_1, P_{01}, P_{11}), \ldots, (p_p, P_{0p}, P_{1p})]$$

zugrundelegen, während wir für geeignetes $\Delta \geq 0$ die Δ-Quantisierung

$$Q_\Delta(M) = [(m_1, M_{01}, \emptyset), \ldots, (m_m, M_{0m}, \emptyset)]$$

der zum MIDI-Dokument gehörenden endlichen Teilmenge M von $\mathbb{Z} \times 2^{[0:127]} \times \{0\}$ betrachten. Neben diesen Quantisierungen von Partitur- und MIDI-Dateien werden wir später auch noch Γ-Quantisierungen (für geeignetes $\Gamma \geq 0$) von Wellenformdarstellungen (etwa einer PCM-Datei) zur Synchronisation heranziehen. Schon jetzt wollen wir darauf hinweisen, dass wir dabei folgende systematische Bezeichnungsweise zugrundelegen:

$$\begin{array}{lrcl} \text{Partitur:} & Q_0(P) & = & [(p_i, P_{0i}, P_{1i})]_{i=1}^{p} \\ \text{MIDI:} & Q_\Delta(M) & = & [(m_j, M_{0j}, \emptyset)]_{j=1}^{m} \\ \text{Wellenform:} & Q_\Gamma(W) & = & [(w_k, W_{0k}, \emptyset)]_{k=1}^{w}. \end{array}$$

In der 0-Quantisierung der **P**artitur gibt es also p Einsatzzeiten $p_1, \ldots, p_p$, in der Δ-Quantisierung der **M**IDI-Datei gibt es m Einsatzzeiten $m_1, \ldots, m_m$, schließlich gibt es in der Γ-Quantisierung der **W**ellenformdarstellung genau w Einsatzzeiten $w_1, \ldots, w_w$. Des weiteren bezeichnen P_{0i}, M_{0i} und W_{0i} die Tonhöhenmengen zu expliziten Noten in der quantisierten Version der Partitur-, MIDI- bzw. Wellenformdatei, die zum i-ten Zeitpunkt einsetzen. Nur in der Partitur kann es auch noch zu diesem Zeitpunkt implizite Noten geben. Diese werden in der Menge P_{1i} zusammengefasst.

Auf der Basis von $Q_0(P)$ und $Q_\Delta(M)$ soll nun eine Synchronisation zwischen Partitur- und MIDI-Daten vorgenommen werden. Da wir zwei Versionen ein und desselben Musikstücks miteinander vergleichen und synchronisieren wollen, macht es Sinn, dass wir als Beginn beider Versionen den Zeitpunkt Null wählen, d. h. wir nehmen ab jetzt immer an, dass $p_1 = 0$ und $m_1 = 0$ gilt.

Unser Ziel ist es nun, eine Teilverlinkung der Zeitpunkte $p_1, \ldots, p_p$ mit den Zeitpunkten $m_1, \ldots, m_m$ so vorzunehmen, dass möglichst große Übereinstimmung erzielt wird. Dies präzisieren wir im Folgenden durch die Suche nach kostenminimalen Matchings zwischen Partitur und MIDI.

Definition 5.2.2 Unter einem *Partitur-MIDI-Match* (PM-Match) von $Q_0(P)$ und $Q_\Delta(M)$ verstehen wir eine partielle Abbildung $\mu\colon [1:p] \to [1:m]$, die auf ihrem Definitionsbereich streng monoton steigt und die für alle $i \in \mathrm{Def}(\mu)$ stets

$$(P_{0i} \cup P_{1i}) \cap M_{0\mu(i)} \neq \emptyset$$

erfüllt. □

Diese Definition bedarf einiger Erläuterungen. Die Tatsache, dass es manchmal in $Q_0(P)$ bzw. $Q_\Delta(M)$ Ereignisse ohne Gegenereignis gibt, wird durch die Forderung berücksichtigt, dass μ eine partielle und keine totale Funktion sein muss. Die Monotonie von μ spiegelt die Forderung der Zeittreue wider: Erklingt ein Ton in $Q_0(P)$ früher als ein zweiter Ton, so soll dies auch für die μ-Bilder dieser Töne gelten. Die Forderung $(P_{0i} \cup P_{1i}) \cap M_{0\mu(i)} \neq \emptyset$ verhindert, dass Zeitpunkte verlinkt werden, die tonhöhenmäßig nichts miteinander zu tun haben.

Naturgemäß gibt es zu $Q_0(P)$ und $Q_\Delta(M)$ viele verschiedene PM-Matches. Mit Hilfe geeigneter Kostenfunktionen werden wir nun unterschiedliche Matches miteinander vergleichen können. Ziel ist letztlich die effiziente Berechnung eines kostenminimalen PM-Matches.

Im Folgenden arbeiten wir anstelle von μ äquivalenterweise mit seinem Graphen

$$\mu \equiv \mathrm{Graph}(\mu) := \{(i_1, j_1) < \cdots < (i_\ell, j_\ell)\},$$

wobei

$$i_a \in I = \{i_1 < \cdots < i_\ell\} \subseteq [1:p] \quad \text{und} \quad j_a \in J = \{j_1 < \cdots < j_\ell\} \subseteq [1:m].$$

Statt $(i,j) \in \mathrm{Graph}(\mu)$ schreiben wir ab jetzt $(i,j) \in \mu$. Ferner sei $|\mu| := |\mathrm{Graph}(\mu)| = \ell$.

In der folgenden Definition ordnen wir jedem PM-Match μ Kosten zu. Dabei benutzen wir einen Parametersatz $\pi := (\alpha, \beta, \gamma, \delta, \zeta, \Delta) \in \mathbb{R}^6_{\geq 0}$ aus sechs zunächst beliebigen reellen Parametern, die später noch näher spezifiziert werden.

Definition 5.2.3 Zum Parametersatz $\pi := (\alpha, \beta, \gamma, \delta, \zeta, \Delta) \in \mathbb{R}^6_{\geq 0}$ seien die PM-Kosten des PM-Matches μ zwischen der 0-Quantisierung $Q_0(P)$ einer Partitur und der Δ-Quantisierung $Q_\Delta(M)$ eines zugehörigen MIDI-Dokuments wie folgt definiert:

$$
\begin{aligned}
C_\pi^{\mathrm{PM}}(\mu|P,M) \;:=\; & \alpha \cdot \sum_{(i,j)\in\mu} \left(|P_{0i} \setminus M_{0j}| + \left\lceil \frac{|P_{1i}|}{|P_{1i} \cup M_{0j}|} \right\rceil - \left\lceil \frac{|P_{1i} \cap M_{0j}|}{|P_{1i} \cup M_{0j}|} \right\rceil \right) + \\
& \beta \cdot \sum_{(i,j)\in\mu} \left(|M_{0j} \setminus (P_{0i} \cup P_{1i})| + |M_{0j} \cap P_{1i}| - \left\lceil \frac{|P_{1i} \cap M_{0j}|}{|P_{1i} \cup M_{0j}|} \right\rceil \right) + \\
& \gamma \cdot \sum_{k\notin \mathrm{Def}(\mu)} \left(|P_{0k}| + \left\lceil \frac{|P_{1k}|}{|P_{0k} \cup P_{1k}|} \right\rceil \right) + \qquad (5.2) \\
& \delta \cdot \sum_{t\notin \mathrm{Bild}(\mu)} |M_{0t}| + \\
& \zeta \cdot \sum_{(i,j)\in\mu} \left| p_i - m_j \cdot \frac{\ell(P)}{\ell(M)} \right|.
\end{aligned}
$$

□

Auch diese Definition bedarf einiger Erläuterungen. Die zu α gehörige Summe stellt die Kosten für die nichtgematchten Töne dar, die von expliziten und impliziten Notenereignissen der Partitur stammen. Genauer misst die Kardinalität $|P_{0i} \setminus M_{0j}|$ die Kosten, die durch die Differenz der Menge aller expliziten Noten zur i-ten Einsatzzeit in der Partitur im Vergleich zur Menge der expliziten Noten zur j-ten quantisierten Einsatzzeit in der MIDI-Version des Musikstücks entstehen. In Bezug auf die Differenz des zweiten und dritten Terms in der ersten Klammer bemerken wir zunächst, dass wegen $(i,j) \in \mu$ insbesondere M_{0j} nicht leer ist, also im Nenner jeweils eine von Null verschiedene Kardinalität steht. Des weiteren nimmt wegen $P_{1i} \cap M_{0j} \subseteq P_{1i} \subseteq P_{1i} \cup M_{0j}$ die Differenz der beiden Aufrundungen nur die Werte 0 oder 1 an. Die Differenz ist genau dann gleich 1, wenn zur i-ten Einsatzzeit in der Partitur eine implizite Note vorkommt, die zur j-ten Einsatzzeit in der MIDI-Fassung kein Pendant hat. Die zu β gehörige Summe misst durch den ersten Term die Kosten für die falsch extrahierten Töne, die nicht in $P_{0i} \cup P_{1i}$ liegen. Die beiden nächsten Terme garantieren, dass von jeder impliziten Note der Partitur im Fall $M_{0j} \cap P_{1i} \neq \emptyset$ nur *ein* Match berücksichtigt wird, was sich bei den Kosten im Term $|M_{0j} \cap P_{1i}| - 1$ widerspiegelt. Die zu γ gehörige Summe nimmt alle Einsatzzeiten der Partitur, die nicht zum Match μ gehören, und zählt im ersten Term in der Klammer die Anzahl der expliziten Noten zur k-ten Einsatzzeit, $k \notin \mathrm{Def}(\mu)$, während der zweite Term anzeigt, ob zu dieser Einsatzzeit in der Partitur eine implizite Note vorkommt (Term = 1) oder nicht (Term = 0). Dabei wird durch den 0/1-wertigen Term $\lceil |P_{1k}|/|P_{0k} \cup P_{1k}| \rceil$ berücksichtigt, dass die Tonhöhen in P_{1k} lediglich Alternativen darstellen[1]. Die zu δ gehörige Summe stellt die Kosten für die Akkordereignisse der MIDI-Datei in Rechnung, die kein Pendant auf der Partitur-Seite haben. Der letzte Summand $\sum_{(i,j)\in\mu} \left| p_i - m_j \cdot \frac{\ell(P)}{\ell(M)} \right|$ ist eine Art justierter ℓ^1-Abstand des Vektorpaares $(p_i, m_j)_{(i,j)\in\mu}$.

Im Folgenden halten wir den Parametersatz π sowie die Quantisierungen $Q_0(P)$ und $Q_\Delta(M)$

[1] Wir gehen hier vereinfachend davon aus, dass pro Einsatzzeit in der Partitur nur eine Sondernote vorkommt. Bei mehreren Sondernoten müsste noch die entsprechende Anzahl miteinbezogen werden.

fest und schreiben für die zum PM-Match μ gehörigen Kosten kurz $C^{\mathrm{PM}}(\mu)$. Ist nun $(i,j) \in \mu$, so ist auch $\mu' := \mu \setminus \{(i,j)\}$ ein PM-Match, und für die Kostendifferenz $C^{\mathrm{PM}}(\mu) - C^{\mathrm{PM}}(\mu')$ ergibt sich nach leichter Rechnung:

$$\begin{aligned} C^{\mathrm{PM}}(\mu) - C^{\mathrm{PM}}(\mu') \;=\; & \alpha \cdot \left(|P_{0i} \setminus M_{0j}| + \left\lceil \frac{|P_{1i}|}{|P_{1i} \cup M_{0j}|} \right\rceil - \left\lceil \frac{|P_{1i} \cap M_{0j}|}{|P_{1i} \cup M_{0j}|} \right\rceil \right) + \\ & \beta \cdot \left(|M_{0j} \setminus (P_{0i} \cup P_{1i})| + |M_{0j} \cap P_{1i}| - \left\lceil \frac{|P_{1i} \cap M_{0j}|}{|P_{1i} \cup M_{0j}|} \right\rceil \right) - \\ & \gamma \cdot \left(|P_{0i}| + \left\lceil \frac{|P_{1i}|}{|P_{0i} \cup P_{1i}|} \right\rceil \right) - \delta \cdot |M_{0j}| + \zeta \cdot \left| p_i - m_j \cdot \frac{\ell(P)}{\ell(M)} \right|. \end{aligned} \tag{5.3}$$

Aufgrund der letzten Formel kann man nun einen kostenminimalen PM-Match μ mittels dynamischer Programmierung bestimmen, indem man eine Matrix $C = (c_{ij})$ mit $i \in [0:p]$ und $j \in [0:m]$ einführt, die durch $c_{0j} = c_{i0} := C^{\mathrm{PM}}(\emptyset)$ initialisiert wird und an der Position $(i,j) \in [1:p] \times [1:m]$ die minimalen Kosten eines PM-Matches μ verzeichnet, wobei der Graph von μ Teilmenge von $[1:i] \times [1:j]$ ist. Gemäß dieser Definition sind dann c_{pm} die minimalen Kosten eines globalen PM-Matches. Nach obiger Formel ist für $(i,j) \in [1:p] \times [1:m]$:

$$c_{ij} = \min\{c_{i,j-1}, c_{i-1,j}, c_{i-1,j-1} + d_{ij}^{\mathrm{PM}}\},$$

wobei

$$d_{ij}^{\mathrm{PM}} := \begin{cases} \text{rechte Seite von Glg. 5.3,} & \text{falls } (P_{0i} \cup P_{1i}) \cap M_{0j} \neq \emptyset \\ 0 & \text{sonst.} \end{cases} \tag{5.4}$$

Die Berechnung der Matrix C ist nachfolgend im Pseudocode angegeben:

```
SCORE-MIDI-SYNCHRONIZATION-COST(π,Q_0(P),Q_Δ(M))
1   p := length[Q_0(P)]
2   m := length[Q_Δ(M)]
3   for i := 0 to p
4      do c[i,0] := C_π^PM(∅|P,M) (according to Eq. 5.2)
5   for j := 1 to ρ
6      do c[0,j] := C_π^PM(∅|P,M) (according to Eq. 5.2)
7   for i := 1 to p
8      do for j := 1 to m
9         do compute d^PM[i,j] (according to Eq. 5.4)
10           if c[i-1,j-1] + d^PM[i,j] < min{c[i-1,j], c[i,j-1]}
11              then c[i,j] := c[i-1,j-1] + d^PM[i,j]
12           else if c[i-1,j] < c[i,j-1]
13                 then c[i,j] := c[i-1,j]
14                 else c[i,j] := c[i,j-1]
15  return C = (c[i,j])
```

Mit Hilfe der so berechneten Matrix C kann jetzt leicht ein kostenminimaler PM-Match mit folgender Prozedur berechnet werden:

```
SCORE-MIDI-SYNCHRONIZATION(C,p,m)
1   i := p, j := m, PM-Match := ∅
2   while (i > 0) & (j > 0)
3      do if c[i, j] = c[i, j − 1]
4            then j := j − 1
5            else if c[i, j] = c[i − 1, j]
6               then i := i − 1
7            else PM-Match := PM-Match ∪ {(i, j)}, i := i − 1, j := j − 1
8   return PM-Match
```

Beispiel 5.2.4 Für den ersten Takt der Aria geben wir nun die Folgen der quantisierten Ereignisse an. Dabei wurde für die Partitur eine 0-Quantisierung durchgeführt, während für die MIDI-Abspielung $\Delta = 50$ ms gesetzt wurde, da dieser Wert einen sinnvollen Schwellwert zwischen den Asynchronismen der Akkorde (30 - 50 ms, siehe Unterabschnitt 2.6.2) und den kürzesten Notenfolgen (etwas unter 100 ms) darstellt. Die musikalischen Einsatzzeiten, gegeben wie in Beispiel 5.2.1, wurden bezüglich eines gleichmäßigen Tempos T „Viertelnote = 60 M.M.“ gemäß der Formel 5.1 in physikalischen Einsatzzeiten p_i (in Sekunden) umgerechnet. Aus Gründen der Anschaulichkeit, haben wir hier auch die Einsatzzeiten m_j, die im Fall einer MIDI-Datei in Ticks gegeben sind, in Sekunden umgerechnet. In der Tabelle sind die quantisierten Ereignisse von beiden Dateien so positioniert, dass sie sich tatsächlich entsprechen.

$Q_0(P)$				$Q_\Delta(M)$			
i	p_i	P_{0i}	P_{1i}	j	m_j	M_{0j}	M_{1j}
1	0	{55, 79}	∅	1	0	{55, 79}	∅
2	1	{59, 79}	∅	2	1.33	{59, 79}	∅
3	2	{62}	{79, 81}	3	2.66	{62, 81}	∅
				4	2.83	{79}	∅
				5	3	{81}	∅
4	2.75	{83}	∅	6	3.66	{83 }	∅

Der entsprechende Teil der Matrix C zur Kostenberechnung für die „Abschnitte“ der Länge $\ell(P) = 3$ s und $\ell(M) = 4$ s bezüglich des Parametervektors $\pi = (1, 1, 1, 1, 200\,\text{ms}, 50\,\text{ms})$ sieht wie folgt aus:

	0	1	2	3	4	5	6
0	16.0000	16.0000	16.0000	16.0000	16.0000	16.0000	16.0000
1	16.0000	**12.0000**	12.0000	12.0000	12.0000	12.0000	12.0000
2	16.0000	12.0000	**8.1035**	8.1035	8.1035	8.1035	8.1035
3	16.0000	12.0000	8.1035	**4.4385**	4.4385	4.4385	4.4385
4	16.0000	12.0000	8.1035	4.4385	4.4385	4.4385	**2.5009**

Das Ablesen eines globalen Matches beginnt an dem rechten unteren Element der Tabelle, in unserem Beispiel an der Position $(4, 6)$. Da alle drei benachbarten Werte größer als der Wert $c(4, 6) = 2.5009$ sind, wird diese Position als zum Match gehörig deklariert. Zeile 7 in

der Prozedur `SCORE-MIDI-SYNCHRONISATION` liefert $(3, 5)$ als neue Position. In den nächsten zwei Iterationsschritten kommen die Zeilen 3 und 4 der Prozedur zum Einsatz. Es kommt kein neuer Matchbeitrag hinzu, sondern es wird nur die Position $(3, 4)$ und dann die Position $(3, 3)$ angesteuert. Erst ab der Position $(3, 3)$ werden neue Beiträge zum Match detektiert, und zwar an den Positionen $(3, 3), (2, 2)$ und $(1, 1)$. Damit ist die Synchronisation zwischen beiden Dateien beendet. Das Matching-Ergebnis lautet also: PM-Match $= [(1, 1), (2, 2), (3, 3), (4, 6)]$ und ist genau die erwartete Folge. Man sieht sofort, wie der Algorithmus die Fuzzy-Note (3. Ereignis im $Q_0(P)$) mit der ersten der drei zu dem Triller gehörigen Ereignisse (3., 4. und 5. Ereignis im $Q_\Delta(M)$) verlinkt bzw. synchronisiert und alle drei Ereignisse dieser Fuzzy-Note zuordnet. □

Die Kostenberechnung eines optimalen PM-Matches wird in Abbildung 5.9 anhand einer 3D-Graphik illustriert. Dabei stellt die rechte Koordinatenachse die Indizes j für die Zeitpunkte m_j, $1 \leq j \leq m$, $m = 40$, dar. Der mittleren, horizontalen Koordinatenachse entsprechen die Inidizes i für die Zeitpunkte p_i, $1 \leq i \leq p$, $p = 23$. Der vertikalen Koordinate können dann die Werte c_{ij} der Matrix C entnommen werden, wobei der Parametervektor π geeignet gewählt wurde.

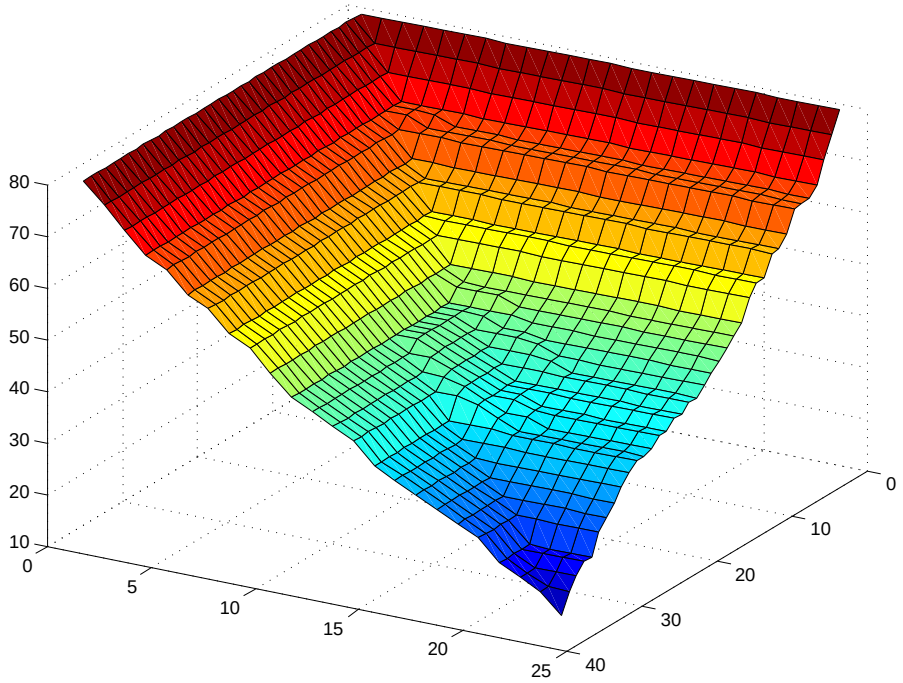


Abbildung 5.9: Die Kostenberechnung eines optimalen PM-Matches für die ersten vier Takte der *Aria con Variazioni* von J. S. Bach.

5.3 Synchronisation von Partitur und PCM

Zur Synchronisation von Partitur und Wellenformdarstellung[2] werden wir im Folgenden von der zur modifizierten Partitur gehörenden endlichen Teilmenge P von $\mathbb{Z} \times 2^{[0:127]} \times \{0,1\}$ die 0-Quantisierung

$$Q_0(P) = [(p_1, P_{01}, P_{11}), \ldots, (p_p, P_{0p}, P_{1p})]$$

zugrundelegen, während wir für geeignetes $\Gamma \geq 0$ die Γ-Quantisierung

$$Q_\Gamma(W) = [(w_1, W_{01}, \emptyset), \ldots, (w_w, W_{0w}, \emptyset)]$$

der zum PCM-Dokument gehörenden endlichen Teilmenge W von $\mathbb{Z} \times 2^{[0:127]} \times \{0\}$ betrachten. Auf dieser Basis soll nun eine Synchronisation vorgenommen werden. Wieder gehen wir o. B. d. A. davon aus, dass $p_1 = 0$ und $w_1 = 0$ gilt.

Im Gegensatz zur Diskussion von PM-Matches werden wir in diesem Fall zur teilweisen Auslöschung von Oktavfehlern, die bei der Analyse von Wellenformen auftreten können (siehe Unterabschnitt 4.3.2), mit Modulo-12-Versionen der ursprünglichen Tonhöhenmengen arbeiten:

$$\begin{aligned} P'_{0i} &:= P_{0i} \bmod 12 = \{x \bmod 12 \,|\, x \in P_{0i}\}, \\ P'_{1i} &:= P_{1i} \bmod 12 = \{x \bmod 12 \,|\, x \in P_{1i}\}, \\ W'_{0j} &:= W_{0j} \bmod 12 = \{x \bmod 12 \,|\, x \in W_{0j}\}. \end{aligned}$$

Die Teilverlinkung der Zeitpunkte $p_1, \ldots, p_p$ mit den Zeitpunkten $w_1, \ldots, w_w$ kann nun wie im Fall der Partitur-MIDI-Synchronisation vorgenommen werden. Analog zur Definition 5.2.2 verstehen wir unter einem *Partitur-PCM-Match* (PW-Match) von $Q_0(P)$ und $Q_\Gamma(W)$ eine partielle Abbildung $\mu\colon [1:p] \to [1:w]$, die auf ihrem Definitionsbereich streng monoton steigt und die für alle $i \in \mathrm{Def}(\mu)$ stets $(P'_{0i} \cup P'_{1i}) \cap W'_{0\mu(i)} \neq \emptyset$ erfüllt.

Neben den Oktavfehlern treten bedingt durch die Obertöne natürlich auch noch andere Fehler bei der Tonhöhenextraktion auf. Diese werden aber hier nicht weiter verfolgt. Die Erläuterungen nach der Definition von PM-Matches gelten nach entsprechenden Modifikationen auch für PW-Matches.

Die PW-Kosten des PW-Matches μ zwischen der 0-Quantisierung $Q_0(P)$ einer Partitur und der Γ-Quantisierung $Q_\Gamma(W)$ eines zugehörigen PCM-Dokuments bezüglich eines Parametersatzes π werden durch die Kostenfunktion $C_\pi^{\mathrm{PW}}(\mu|P, W)$ festgelegt. Zu deren Definition ersetzen wir einfach in Definition 5.2.3 die Symbole P_{0i} durch P'_{0i}, P_{1i} durch P'_{1i}, M_{0j} durch W'_{0j}und $m_j \cdot \ell(P)/\ell(M)$ durch $w_j \cdot \ell(M)/\ell(W)$. Mit anderen Worten, die PW-Kostenfunktion unterscheidet sich von der PM-Kostenfunktion nur in der Benutzung von Modulo-12-Versionen der Tonhöhenmengen. Damit lassen sich auch in völliger Analogie zur Partitur-MIDI-Synchronisation die zwei Algorithmen

`SCORE-PCM-SYNCHRONIZATION-COST(`π`,`$Q_0(P)$`,`$Q_\Gamma(W)$`)`

und

[2] Im Folgenden sprechen wir statt von Wellenform kurz von PCM.

`SCORE-PCM-SYNCHRONIZATION(`C`,`p`,`w`)`

angeben, über die ein kostenminimaler PW-Match berechnet werden kann.

Beispiel 5.3.1 Wir illustrieren die PW-Synchronisation wieder anhand des Beispiels der Aria. Hierbei wurden zwei Versionen der ersten $4\frac{1}{3}$ Takte dieses Musikstücks gewählt, eine Version in Partiturformat von „physikalische Dauer“ $\ell(W) = 13$ Sekunden (die Umrechnung der musikalischen in die physikalischen Einsatzzeiten wurde bereits im Beispiel 5.2.4 erklärt) und eine PCM-Version, bestehend aus $\ell(W) = 357876$ Samples bzw. 16.23 Sekunden[3]. Aus den im Beispiel 5.2.4 bereits erklärten Gründen arbeiten wir wieder mit der 0-Quantisierung der Partitur, während für die PCM-Version die Quantisierungskonstante Γ auf 50 ms gesetzt wurde. Da hier nur das prinzipielle Vorgehen illustriert werden soll, beschränken wir uns in der folgenden Diskussion auf den zweiten Takt. Hier tauchen u. a. zwei Vorschläge auf, die auf Partiturseite durch Fuzzy-Noten beschrieben werden. Die Folgen der quantisierten Ereignisse $Q_0(P)$ und $Q_\Gamma(W)$ sind in der folgenden Tabelle zeilenweise dargestellt, wobei aus Gründen der Übersichtlichkeit die zu verlinkenden Ereignisse in jeweils derselben Zeile stehen. (Diese Zuordnung ist ja gerade das Ziel der Synchronisation.)

$Q_0(P)$				$Q_\Gamma(W)$			
i	p_i	P_{0i}	P_{1i}	k	w_k	W_{0k}	W_{1k}
5	3	{54, 81}	∅	7	3.86	{54, 81}	∅
6	3.5	∅	{78, 79}	8	4.47	{79}	∅
				9	4.75	{66, 78}	∅
7	4	{57}	{74, 76}	10	5.06	{57, 66, 76}	∅
				11	5.71	{57, 74}	∅
8	5	{62}	∅	12	6.39	{57, 62}	∅

Dieses Beispiel illustriert zwei der Phänomene, die typischerweise auf Seiten der PCM-Daten auftreten. Zum einen erscheint der in Position 10 extrahierte Ton 57 ebenfalls in den sich anschließenden Positionen 11 und 12. Dieses Phänomen kann durch das Weiter- und Ausklingen des Tons 57 erklärt werden. Hierdurch wird der Ton während seiner physikalischen Gesamtdauer durch den Extraktionsalgorithmus immer wieder bei neu hinzukommenden Notenereignissen als „neu“ erkannt. Zum anderen taucht in Position 9 ein Oktavfehler auf (Ton 66, der durch den gerade gespielten Ton 78 und möglicherweise den abklingenden Ton 54 aus der Position 7 „verursacht wurde“), der sich sogar noch auf die nächste Position 10 auswirkt. Durch die Modulo-12-Versionen der Folgen, die in der folgenden Tabelle angegeben sind, können zumindest die Oktavfehler behoben werden.

[3] In der Tabelle wurden aus Gründen der Anschaulichkeit die in Samples gegebenen Einsatzzeiten w_k in Sekunden umgerechnet.

$Q_0(P)$				$Q_\Gamma(W)$			
i	p_i	P_{0i}	P_{1i}	k	w_k	W_{0k}	W_{1k}
5	3	{6, 9}	$\emptyset$	7	3.86	{6, 9}	$\emptyset$
6	3.5	$\emptyset$	{6, 7}	8	4.47	{7}	$\emptyset$
				9	4.75	{6}	$\emptyset$
7	4	{9}	{2, 4}	10	5.06	{4, 6, 9}	$\emptyset$
				11	5.71	{2, 9}	$\emptyset$
8	5	{2}	$\emptyset$	12	6.39	{2, 9}	$\emptyset$

Der entsprechende Teil der Matrix C zur Kostenberechnung sieht wie folgt aus (hierbei wurde $\pi = (1, 1, 1, 1, 90\text{ ms}, 50\text{ ms})$ gewählt):

	6	7	8	9	10	11	12
4	114.8476	114.8476	114.8476	114.8476	114.8476	114.8476	114.8476
5	114.8476	**111.8700**	111.8700	111.8700	111.8700	111.8700	111.8700
6	114.8476	111.8700	**110.8132**	110.8132	110.8132	110.8132	110.8132
7	114.8476	111.8700	110.8132	110.8132	**107.4704**	107.4704	107.4704
8	114.8476	111.8700	110.8132	110.8132	107.4704	107.4704	**106.7956**

Aus der Tabelle berechnet sich nun analog zu Beispiel 5.2.4 der globale Match PW-Match = [(5, 7), (6, 8), (7, 10), (8, 12)]. Das entspricht genau der erwarteten Folge von Matching-Paaren. Der Algorithmus hat für beide Vorschlagsnoten der Partitur (Position 6 bzw. 7 in $Q_0(P)$) die Anfangsposition in der entprechenden PCM-Version (Position 8 bzw. 10 in $Q_\Gamma(W)$) richtig erkannt. □

5.4 Synchronisation von MIDI und PCM

Auch die Synchronisation von MIDI und PCM kann nun durch leichte Modifikationen der vorherigen Verfahren bewerkstelligt werden. Im Folgenden werden wir von der zur MIDI-Datei gehörenden endlichen Teilmenge M von $\mathbb{Z} \times 2^{[0:127]} \times \{0\}$ für geeignetes $\Delta \geq 0$ die Δ-Quantisierung

$$Q_\Delta(M) = [(m_1, M_{01}, \emptyset), \ldots, (m_m, M_{0m}, \emptyset)]$$

zugrundelegen, während wir für geeignetes $\Gamma \geq 0$ die Γ-Quantisierung

$$Q_\Gamma(W) = [(w_1, W_{01}, \emptyset), \ldots, (w_w, W_{0w}, \emptyset)]$$

der zum PCM-Dokument gehörenden endlichen Teilmenge W von $\mathbb{Z} \times 2^{[0:127]} \times \{0\}$ betrachten. Auf dieser Basis soll nun eine Synchronisation vorgenommen werden, wobei wir wieder o. B. d. A. davon ausgehen dürfen, dass $m_1 = 0$ und $w_1 = 0$ gilt. Darüber hinaus arbeiten wir auch hier, zur Verminderung von Analysefehlern, mit Modulo-12-Versionen der Tonhöhenmengen:

$$M'_{0i} := M_{0i} \bmod 12 \quad \text{und} \quad W'_{0j} := W_{0j} \bmod 12.$$

Ziel ist es, eine optimale Teilverlinkung der Zeitpunkte $m_1, \ldots, m_m$ mit den Zeitpunkten $w_1, \ldots, w_w$ vorzunehmen. Analog zu Definition 5.2.2 verstehen wir unter einem *MIDI-PCM-Match* (MW-Match) von $Q_\Delta(M)$ und $Q_\Gamma(W)$ eine partielle Abbildung $\mu\colon [1:m] \to [1:w]$,

die auf ihrem Definitionsbereich streng monoton steigt und die für alle $i \in \mathrm{Def}(\mu)$ stets $M'_{0i} \cap W'_{0\mu(i)} \neq \emptyset$ erfüllt.

Zur Angabe einer geeigneten Kostenfunktion benutzen wir wieder einen Parametersatz $\pi := (\alpha, \beta, \gamma, \delta, \zeta, \Delta, \Gamma) \in \mathbb{R}^7_{\geq 0}$, der diesmal aus sieben zunächst beliebigen reellen Parametern, die später noch näher spezifiziert werden, besteht. Da bei der MIDI-PCM-Synchronisation keine impliziten Notenobjekte zu berücksichtigen sind, vereinfacht sich die Definition der Kostenfunktion im Vergleich zur Definition 5.2.3 wie folgt:

$$\begin{aligned} C_\pi^{\mathrm{MW}}(\mu|M,W) \quad &:= \textstyle\sum_{(i,j)\in\mu} \left\{ \alpha \cdot \left| M'_{0i} \setminus W'_{0j} \right| + \beta \cdot \left| W'_{0j} \setminus M'_{0i} \right| \right\} \\ &\quad + \gamma \cdot \textstyle\sum_{k \notin \mathrm{Def}(\mu)} |M'_{0k}| + \delta \cdot \sum_{t \notin \mathrm{Bild}(\mu)} |W'_{0t}| \\ &\quad + \zeta \cdot \textstyle\sum_{(i,j)\in\mu} \left| m_i - w_j \cdot \frac{\ell(M)}{\ell(W)} \right|. \end{aligned} \tag{5.5}$$

Analog zu Formel 5.3 ergibt sich hieraus:

$$\begin{aligned} C^{\mathrm{MW}}(\mu) - C^{\mathrm{MW}}(\mu') \quad &= \quad \alpha \cdot |M'_{0i} \setminus W'_{0j}| + \beta \cdot |W'_{0j} \setminus M'_{0i}| \\ &\quad - \gamma \cdot |M'_{0i}| - \delta \cdot |W'_{0j}| + \zeta \cdot \left| m_i - w_j \cdot \frac{\ell(M)}{\ell(W)} \right|. \end{aligned} \tag{5.6}$$

Die Bestimmung eines kostenminimalen MW-Matches μ mittels dynamischer Programmierung kann nun wieder in völliger Analogie zum Fall des PM-Matches durchgeführt werden. Die hierfür benötigten Algorithmen bezeichnen wir mit

`MIDI-PCM-SYNCHRONIZATION-COST(`π`,`$Q_\Delta(M)$`,`$Q_\Gamma(W)$`)`

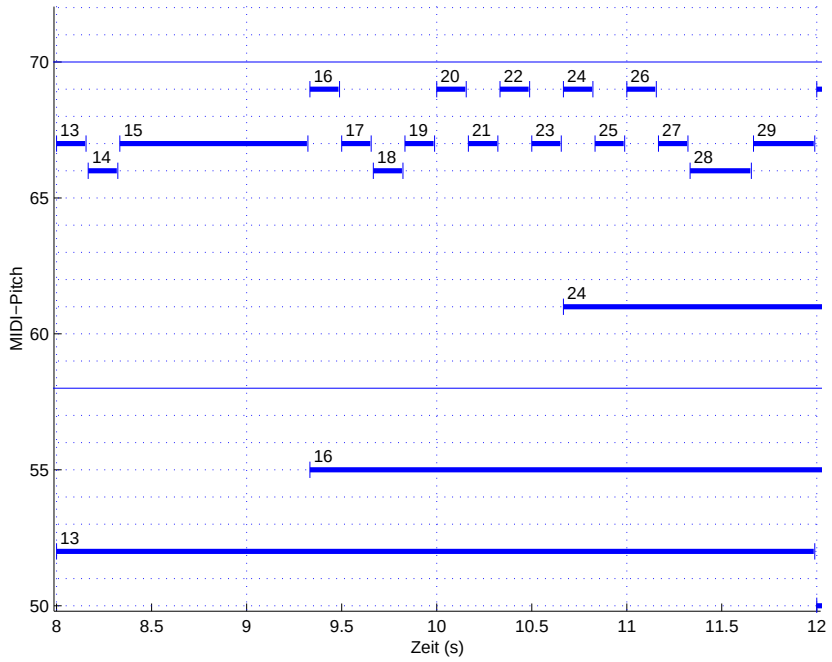
und

`MIDI-PCM-SYNCHRONIZATION(`C`,`m`,`w`)`.

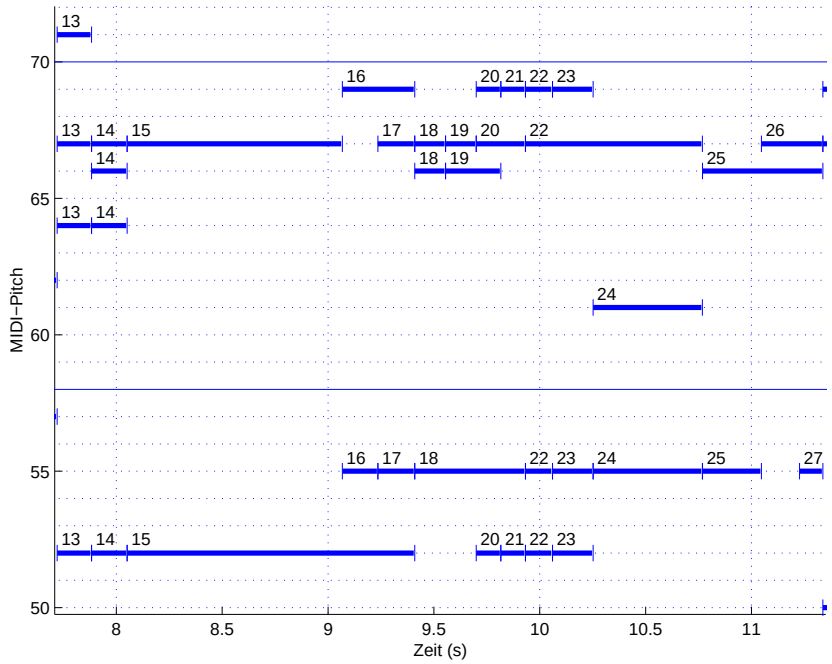
Beispiel 5.4.1 Für die Illustration des MW-Matchings haben wir den dritten Takt der *Aria* gewählt. Dieser ist vor allem deshalb interessant, weil er zwei Sondernoten enthält – einen Mordent und einen Triller (Doppelschlag). Insbesondere ist zu erwarten, dass die Realisation des Trillers stark vom Interpreten abhängig ist (vergleiche hierzu z. B. auch die Abbildungen 5.10 (a) und (b)). Das Problem der Synchronisation solcher Passagen wird noch zusätzlich durch die falsch extrahierten Noten auf Seiten der PCM-Version, bei der es sich um eine von einem Pianisten interpretierte Einspielung handelt, erschwert[4]. Hierbei handelt es sich unter anderem um zusätzliche erkannte Notenereignisse, hervorgerufen durch nachklingende Noten (wie z. B. Tonhöhe 67 in der Position 14, 18 und 20, Abb. 5.10 (b)) sowie Intervallfehler (wie z. B. die Tonhöhen 64 und 71 in der Position 13) oder einfach um Fehler des Extraktionsalgorithmus. So werden z. B. der ausklingende Ton der Tonhöhe 52 zwischen den Positionen 18 und 20 (siehe Abb. 5.10 (b)) und ebenso der Ton der Tonhöhe 55 zwischen den Positionen 26 und 27 nicht erkannt. Dabei ist es meist schwierig zu sagen, ob die Abweichungen der extrahierten Daten von den erwarteten Notenereignissen durch Extraktionsfehlern oder durch Interpretationsungenauigkeiten begründet sind.

Auf der einen Seite bleiben durch den „Modulo-12-Mechanismus“ die Oktavfehler zwar unberücksichtigt, was zu einem robusteren MW-Matching-Algorithmus führt. Auf der anderen

[4] Passagen mit Trillern oder Folgen von sehr kurzen Noten sowie Passagen reicher Harmonie und komplexer Polyphonie sind bei der Extraktion zweifellos die schwierigsten.



(a)



(b)

Abbildung 5.10: Klavierwalzendarstellung des dritten Takts: (a) einer MIDI-Version der *Aria*, (b) einer interpretierten PCM-Version der *Aria*.

Seite kann es aber auch passieren, dass gerade diese Modulo-12-Berechnungen die Ursache falscher Matches ist. Dieser Sachverhalt wird auch durch das folgende Beispiel illustriert. Wieder wurde der Parametervektor $\pi = (1, 1, 1, 1, 1\,\mathrm{s}, 50\,\mathrm{ms}, 50\,\mathrm{ms})$ gewählt. In der folgenden Tabelle sind die Folgen $Q_\Delta(M)$ und $Q_\Gamma(W)$ dargestellt, allerdings sind die Elemente der Folgen entsprechend dem Ergebnis des MW-Matching-Algorithmus zugeordnet (und nicht der „idealen“ Zuordnung entsprechend wie in den Beispielen 5.2.4 und 5.3.1). Ebenfalls wurden in Hinblick auf die folgende Diskussion die „richtigen“ Tonhöhen dargestellt –, also genau diejenigen der Abbildungen 5.10 (a) und 5.10 (b) – auch wenn für den Matching-Algorithmus die Modulo-12-Versionen der Folgen herangezogen wurden.

$Q_\Delta(M)$				$Q_\Gamma(W)$			
i	m_i	M_{0i}	M_{1i}	j	w_j	W_{0j}	W_{1j}
13	8.00	{52, 67}	∅	13	7.72	{52, 64, 67, 71}	∅
14	8.16	{66}	∅	14	7.88	{52, 64, 66, 67}	∅
15	8.33	{67}	∅	15	8.05	{52, 67}	∅
16	9.33	{55, 69}	∅	16	9.06	{55, 69}	∅
17	9.50	{67}	∅	17	9.23	{55, 67}	∅
18	9.66	{66}	∅				
19	9.83	{67}	∅	18	9.40	{55, 66, 67}	∅
20	10.00	{69}	∅				
21	10.16	{67}	∅	19	9.55	{66, 67}	∅
22	10.33	{69}	∅	20	9.70	{52, 67, 69}	∅
23	10.50	{67}	∅				
24	10.66	{61, 69}	∅	21	9.81	{52, 69}	∅
25	10.83	{67}	∅	22	9.93	{52, 55, 67, 69}	∅
26	11.00	{69}	∅	23	10.06	{52, 55, 69}	∅
27	11.16	{67}	∅	24	10.25	{55, 61}	∅
28	11.33	{66}	∅	25	10.76	{55, 66}	∅
29	11.66	{67}	∅	26	11.04	{67}	∅

Aufgrund eines Vergleichs der beiden Klavierwalzendarstellungen (Abbildungen 5.10(a) und 5.10(b)) sind offenbar die Matches $[(13, 13), (16, 16), (24, 24), (28, 25), (29, 26)]$ zu erwarten. Aus obiger Tabelle kann abgelesen werden, dass der MW-Matching-Algorithmus zwar die Notenereignisse $[(13, 13), (16, 16), (28, 25), (29, 26)]$ „richtig“ synchronisiert, allerdings bei dem Paar $(24, 24)$ scheitert und stattdessen die Zuordnung $(24, 21)$ liefert. Diese fehlerhafte Zuordnung ist offensichtlich darin begründet, dass sich gerade in diesem Zeitabschnitt die Notenereignisse der MIDI-Version von den extrahierten Notenereignissen der PCM-Version wesentlich unterscheiden. Allerdings ist diese fehlerhafte Verlinkung nur lokaler Natur und tritt vor allem in Abschnitten mit schnellen, nicht klar definierten oder stark interpretationsabhängigen Notenfolgen auf. Auf lange Sicht „renkt“ der Algorithmus sich aber wieder ein und findet in Passagen mit wohldefinierten Notenereignissen den Weg zur „richtigen“ Synchronisation zurück (z. B. ab dem Matching-Paar $(28, 25)$). In Hinblick auf die in der Einleitung dieses Kapitels angesprochenen Anwendungen ist aber gerade diese globale Verlinkung relevant; kleine lokale Abweichungen und die sich daraus ergebenden Synchronisationsfehler liegen in der Natur des Problems und sind kaum vermeidbar.

5.5 Wahl des Parametersatzes

In diesem Abschnitt gehen wir auf die Wahl des Parametersatzes $\pi = (\alpha, \beta, \gamma, \delta, \zeta, \Delta, (\Gamma))$ ein, durch den sowohl die Größe der Quantisierungsschritte als auch die Gewichtung der Kostenfunktionen festgelegt werden. Für die Interpretation der einzelnen Komponenten von π verweisen wir auf die Diskussion nach Definition 5.2.3.

Wie schon in den vorherigen Abschnitten erwähnt, haben wir meist die Quantisierungsparameter Δ und Γ auf 50 ms festgelegt, da dieser Wert einen sinnvollen Schwellwert zwischen den Asynchronismen der Akkorde (30 - 50 ms, siehe Unterabschnitt 2.6.2) und den kürzesten Notenfolgen (etwas unter 100 ms) darstellt. Dieser Wert für die Quantisierungsparameter Δ und Γ hat sich auch in unseren Experimenten als besonders sinnvoll erwiesen.

Durch die Parameter α und β werden diejenigen Kosten gewichtet, die durch die symmetrische Differenz der Tonhöhenmengen bezüglich eines Matches zweier Zeitpunkte entstehen. Hingegen gewichten die Parameter γ und δ die Kosten derjenigen Notenereignisse der beiden zu verlinkenden Dateien, für die kein Pendant in der jeweils anderen Datei gefunden wird. Wir diskutieren erst einmal die Parameterwahl $\alpha = \beta = \gamma = \delta = 1$ und $\zeta = 0$. Anschaulich gesprochen, führen in diesem Fall die jeweiligen Synchronisationsalgorithmen zu demjenigen Match der beiden zu synchronisierenden Dateien, der einfach die Gesamtanzahl aller Elemente in den zeitlich nicht gematchten Tonhöhenmengen und in den symmetrischen Differenzen der zeitlich gematchten Tonhöhenmengen minimiert. Will man z. B. die Synchronisation gegenüber Interpretations- bzw. Extraktionsfehler toleranter machen, so kann dies durch ein entsprechend kleiner gewähltes β gewährleistet werden. Wir wollen aber im folgenden auf solche Feinheiten nicht näher eingehen. Sowohl aufgrund theoretischer Überlegungen als auch aufgrund experimenteller Ergebnisse hat sich die Parameterwahl $\alpha = \beta = \gamma = \delta = 1$ für die meisten Synchronisationsaufgaben als sinnvoll herausgestellt. Allerdings hat sich in den Experimenten auch gezeigt, dass große zeitliche Abweichungen zwischen den gematchten Zeitpunkten der beiden Dateien meistens auf Synchronisationsfehler zurückgeführt werden können. Um bei einer Synchronisation solche Abweichungen kontrollieren zu können, wurde der mit ζ gewichtete Term in die Kostenfunktion mitaufgenommen (siehe Formel 5.2). Im Folgenden gehen wir näher auf die Wahl des Parameters ζ ein, von dem – wie die Experimente zeigen – wesentlich die Güte der Synchronisationsergebnisse abhängt.

Wir besprechen zuerst den Spezialfall des PM-Matchings und gehen davon aus, dass es sich auf Seite der MIDI-Datei um eine partiturartige, uninterpretierte Version handelt, bei der weder falsche Noten noch zeitliche Temposchwankungen auftauchen. In diesem Fall gilt dann $p = m$ und $p_i - m_i \cdot \frac{\ell(P)}{\ell(M)} = 0$, so dass der Term $\zeta \cdot \left| p_i - m_j \cdot \frac{\ell(P)}{\ell(M)} \right|$, den wir im Folgenden auch als *die zeitliche Komponente der Kostenfunktion* bezeichnen, für $i = j$ immer verschwindet. Wird ein Notenobjekt zur Einsatzzeit p_i in der Partitur mit dem Notenobjekt zur Einsatzzeit m_j in der MIDI-Datei gematcht, dann nimmt die zeitliche Komponente für das Paar (i, j) mit $i \neq j$ einen echt positiven Wert an. Damit erhöhen sich die Kosten bei einem solchen zeitlich abweichenden Match, selbst in dem Fall, dass die Tonhöhenmengen der beiden Notenobjekte übereinstimmen sollten. Mit anderen Worten werden durch den mit ζ gewichteten Term relative zeitliche Schwankungen bestraft.

In der Praxis sieht die Sache aber anders aus. Bei interpretierten Versionen hat man im Allgemeinen kein gleichmässiges Tempo mehr, sondern man hat neben den notenmäßigen

Abweichungen von der Partitur und eventuellen Extraktionsfehlern auch mit vielen lokalen Temposchwankungen zu kämpfen. Durch die zeitliche Komponente werden diese lokalen Schwankungen bestraft, selbst wenn bei einem Match alle Noten und Tonhöhen perfekt übereinstimmen sollten. Liegen extreme Temposchwankungen, wie z. B. accelerando oder ritardando vor, so kann es passieren, dass falsche Notenobjekte gematcht werden oder Notenobjekte ungematcht bleiben, wenn der Parameter ζ zu groß gewählt wird. Liegen aber auf der anderen Seite viele falsche Notenereignisse vor – z. B. aufgrund schlechter Extraktionsergebnisse –, so wird durch die zeitliche Komponente verhindert, dass es bei der Synchronisation zweier Dateien zu zu großen relativen zeitlichen Abweichungen kommt. Dem Parameter ζ kommt somit die Rolle zu, einen Kompromiss zwischen diesen beiden sich widerstreitenden Prinzipien herzustellen.

Nach Definition gilt für einen Match (i, j) die Bedingung $(P_{0i} \cup P_{1i}) \cap M_{0j} \neq \emptyset$. Aus Formel 5.3 und 5.4 ergibt sich durch eine kleine Rechnung

$$d_{ij} = -2 \cdot \left(|P_{0i} \cap M_{0j}| + \left\lceil \frac{|P_{1i} \cap M_{0j}|}{|P_{0i} \cup P_{1i}|} \right\rceil \right) + \zeta \cdot \left| p_i - m_j \cdot \frac{\ell(P)}{\ell(M)} \right|.$$

Nimmt man zunächst an, dass die Tonhöhenmengen zum Match (i, j) in genau einem Element übereinstimmen, also $|P_{0i} \cap M_{0j}| + \left\lceil \frac{|P_{1i} \cap M_{0j}|}{|P_{0i} \cup P_{1i}|} \right\rceil = 1$ gilt, ergibt sich daraus

$$d_{ij} = -2 + \zeta \cdot \left| p_i - m_j \cdot \frac{\ell(P)}{\ell(M)} \right|.$$

Damit ein solcher korrekter Match zweier Notenereignisse im Fall monophoner Passagen zu einer Abnahme der Kostenfunktion führt ($d_{ij} < 0$), sollte die zeitliche Komponente einen Wert kleiner als 2 annehmen. Gleichzeitig soll aber verhindert werden, dass zeitlich zu weit auseinanderliegende Notenobjekte verlinkt werden. Diese Forderung kann aber allein durch eine geeignete Wahl von ζ nicht erfüllt werden. Zur Lösung dieses Problems – bei gleichzeitiger Verringerung des Rechenaufwands – schlagen wir vor, nur die Kosten für diejenigen Einatzzeiten-Paare (p_i, m_j) zu berechnen, die der Bedingung

$$\left| p_i - m_j \cdot \frac{\ell(P)}{\ell(M)} \right| \leq \frac{2}{\zeta}$$

genügen. Somit werden nur die Kosten derjenigen Paare, die sich um die Hauptdiagonale der Kostentabelle herum befinden, berechnet, was den Algorithmus wesentlich effizienter macht. Diesselbe Strategie wird auch im Fall des PW- und des MW-Matchings angewendet. In Abbildung 5.11 ist z. B. der „Streifen“ um die Hauptdiagonale zur Berechnung eines optimalen MW-Matches zum nachfolgenden Beispiel 5.5.1 dargestellt.

Die Komplexität des Algorithmus, wenn auch deutlich durch diese Maßnahme gesenkt, bleibt weiterhin abhängig von dem jeweiligen Musikstück bzw. seinen Realisationen, die zu synchronisieren sind. Ebenso hängt die Wahl des Parameters ζ von dem Musikstück und seiner jeweiligen individuellen Interpretation ab. Bevor wir auf die automatische Bestimmung dieses Parameters eingehen, benötigen wir einige Notationen. Wir beziehen uns im Folgenden wieder auf den PM-Fall. Für den PW- und MW-Fall wird dann völlig analog verfahren. Aus einem Match μ berechnet sich die Tempofunktion T als stückweise konstante Funktion durch $T : [p_{i_1} : p_{i_\ell}[\to \mathbb{R}$, $T(t) := \frac{p_{i_{k+1}} - p_{i_k}}{m_{j_{k+1}} - m_{j_k}}$, falls $t \in [p_{i_k} : p_{i_{k+1}}[$, $1 \leq k < \ell$. (Siehe auch Einleitung

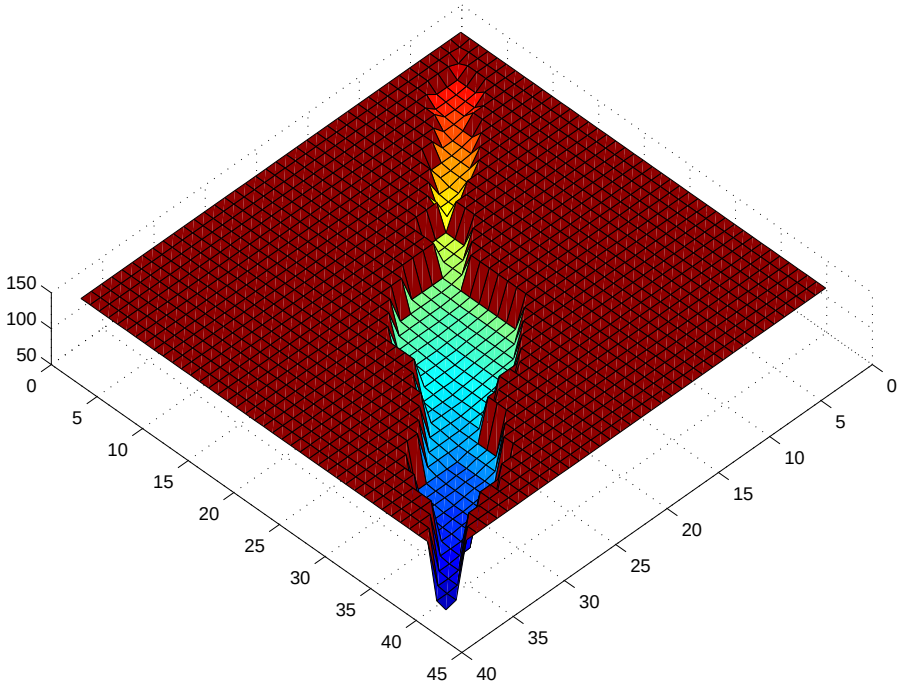


Abbildung 5.11: Effiziente Kostenberechnung eines optimalen MW-Matches für die ersten $4\frac{1}{3}$ Takte der *Aria con Variazioni* durch Beschränkung der Berechnungen auf einen „Streifen“ um die Hauptdiagonale.

zu diesem Kapitel. Im MW-Fall müsste man hier streng genommen von einer relativen Tempofunktion sprechen, da es sich in diesem Fall bei beiden Dateien um interpretierte Versionen des Musikstücks handelt.) Mit σ bezeichnen wir die Standardabweichung von T bezüglich des gleichmäßigen Durchschnittstempos. Wir bestimmen nun ζ durch folgende Vorgehensweise.

1. Berechne für die Menge $\{\tau_1 = 0.15, \tau_2 = 0.3, \tau_3 = 0.5, \tau_4 = 1, \tau_5 = 2, \tau_6 = 3\}$ von typischen Zeitabweichungen, die sich in unseren Experimenten als sinnvoll herausgestellt hat, die zugehörigen Parameter $\zeta_i = 2/\tau_i$ $(i \in [1:6])$.

2. Führe nun für jeden Parameter $\zeta = \zeta_i$ $(i \in [1:6])$ den Synchronisationsalgorithmus durch und bestimme dabei die Kardinalität $|\mu|$ des Matches μ, die Tempofunktion T und die zugehörige Standardabweichung σ.

3. Wähle denjenigen Parameter ζ, der bezüglich eines noch festzulegenden Kriteriums, das von $|\mu|$ und σ abhängt, den besten Match erzeugt.

Die folgenden beiden Kriterien haben sich sowohl aus theoretischer als auch aus experimenteller Sicht als sinnvoll herausgestellt. Nach dem ersten Kriterium wird dasjenige ζ gewählt, das zu einem längsten Match der Länge $|\mu|$ führt, wobei bei mehreren Kandidaten die niedrigere Standardabweichung σ maßgebend ist. Bleiben immer noch mehrere Kandidaten zur Auswahl, so wird unter diesen das größte ζ gewählt. Beim zweiten Kriterium wird das Verhältnis $|\mu|/\sigma$ maximiert, was einem Kompromiss zwischen einer großen Anzahl an Matchingbeiträgen und

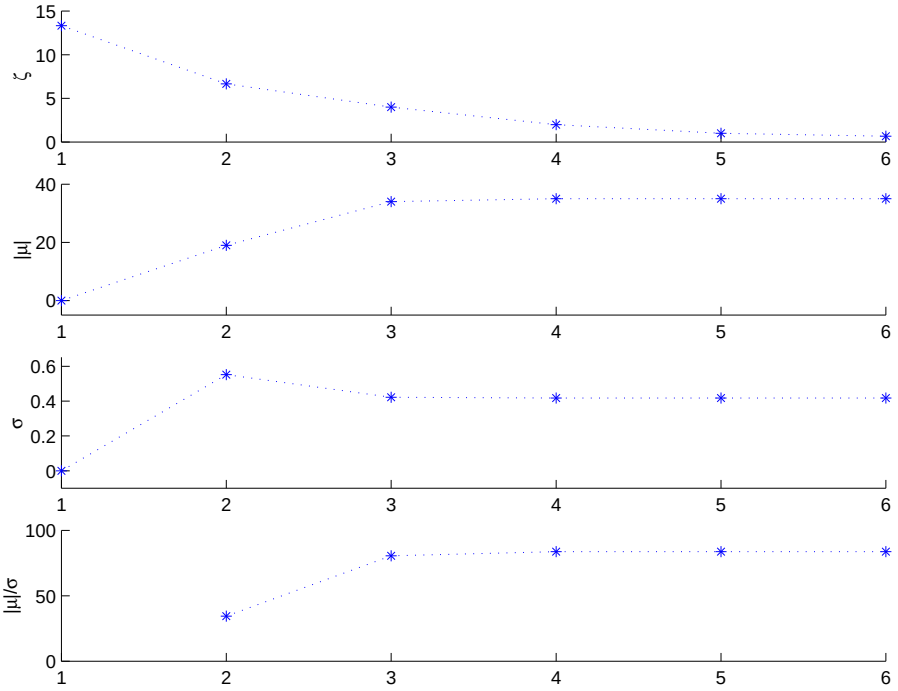


Abbildung 5.12: Automatische Wahl des Parameters ζ. Von oben nach unten: Gewichtsparameter ζ, Kardinalität des Matches μ, Standardabweichung σ der zu vorgegebenem ζ gehörigen Tempofunktion, Verhältnis $|\mu|/\sigma$.

einer niedrigen Standardabweichung der geschätzten Tempofunktion entspricht. Auch hier wird bei mehreren optimalen Kandidaten das größte ζ gewählt.

Beispiel 5.5.1 Wir illustrieren das MW-Synchronisationsverfahren anhand der ersten $4\frac{1}{3}$ Takte der *Aria*. In Abbildung 5.12 sind von oben nach unten die sechs verschiedenen Werte von ζ, die zugehörigen Kardinalitäten $|\mu|$, die Standardabweichungen $|\sigma|$ und die Quotienten $|\mu|/\sigma$ dargestellt. Den x-Achsen entsprechen dabei die Indizes i für die Werte ζ_i, $i = 1, \ldots, 6$. Wie zu erwarten ist, nimmt für fallende Werte von ζ die Anzahl $|\mu|$ der gematchten Zeitpunkte zu, da bei fallendem ζ relative Zeitabweichungen weniger stark ins Gewicht fallen. Nach beiden Kriterien führen die Parameter ζ_4, ζ_5 und ζ_6 zu optimalen Matching-Ergebnissen, was schliesslich zur Wahl $\zeta = \zeta_4$ führt.

Eine genauere Untersuchung des Synchronisationsergebnisses zeigt, dass von insgesamt $m = 40$ quantisierten Ereignissen der MIDI-Datei $Q_\Delta(M)$ 28 Ereignisse (70 %) richtig und 11 Ereignisse (27.5%) gar nicht verlinkt wurden. Ein weiteres verlinktes Ereignis gehört zu den Trillerereignissen, bei denen schon der Begriff einer „korrekten“ Verlinkung aufgrund der unterschiedlichen Interpretationen dieser impliziten Noten problematisch ist.

Abbildung 5.13 (a) zeigt die durch den optimalen Match μ, also das Synchronisationsergebnis definierte (relative) Tempofunktion T. (Aus optischen Gründen wurde in der Abbildung für jedes Intervall $[p_{i_1} : p_{i_\ell})$ der entsprechende Wert von T auf diesem Intervall nur durch einen Punkt über dem entsprechenden Intervallmittelpunkt dargestellt. Diese Punkte wurden dann durch eine Linienzug verbunden.) In Abb. 5.14 (a) ist der Zeitfluss der Synchronisation

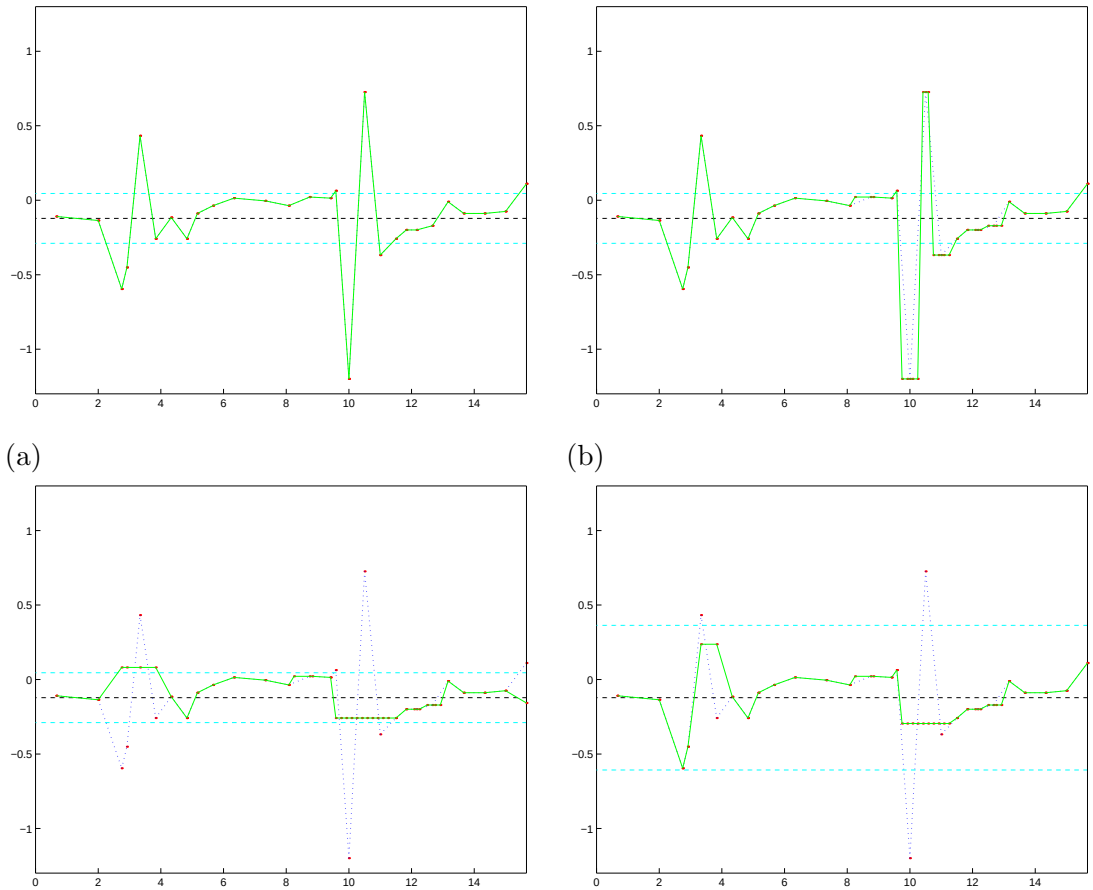
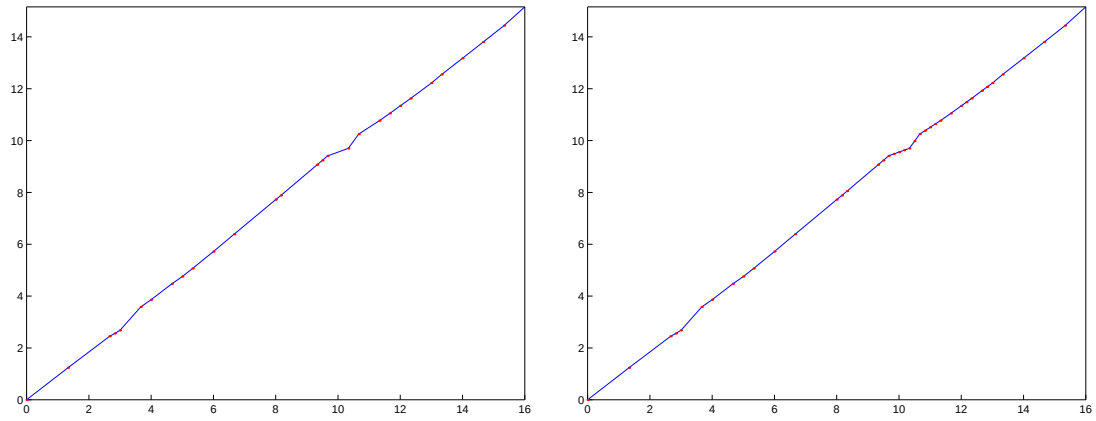
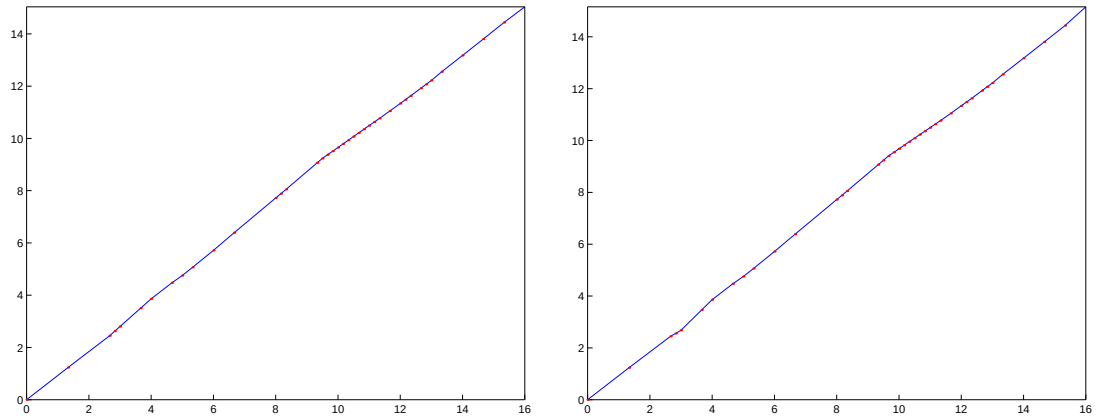


Abbildung 5.13: Tempofunktionen (x-Achse als Zeitachse, y-Achse gibt den Logarithmus des momentanen Tempos an): (a) die fehlenden Matches sind nicht interpoliert, (b) die fehlenden Matches sind interpoliert, (c) Ausreißerbeseitigung durch Schwellwertverfahren (Schwellwert = α-winsorisierte Standardabweichung) mit anschließender Interpolation, (d) Ausreißerbeseitigung durch Schwellwertverfahren (Schwellwert vorab datenunabhängig gewählt) mit anschließender Interpolation.



(a) (b)



(c) (d)

Abbildung 5.14: Zeitfluss der synchronisierten Einsatzzeiten in der MIDI-Datei (in der x-Achse) und PCM-Datei (y-Achse): (a) – (d) entspricht (a) – (d) in Abb. 5.13.

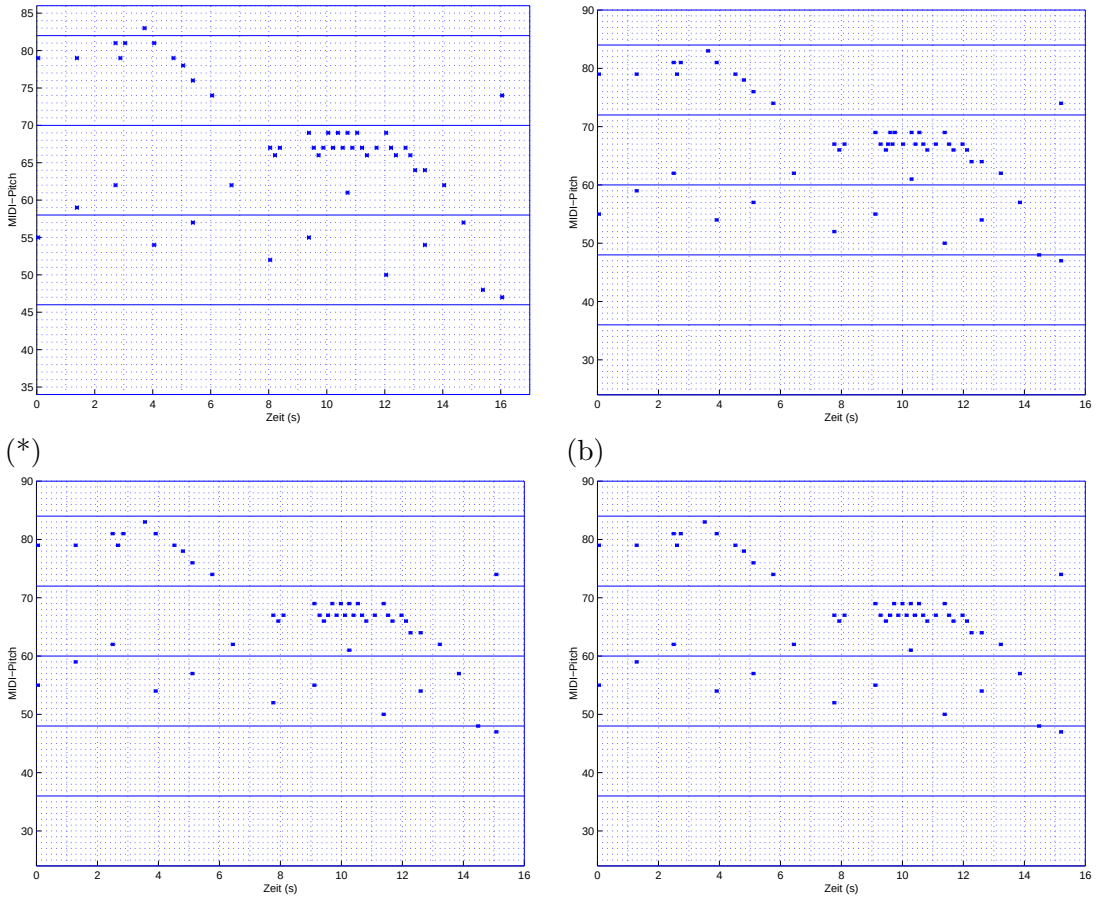


Abbildung 5.15: Reduzierte Klavierwalzendarstellung der MIDI-Datei: (a) nichtsynchronisiert, (b) synchronisiert mittels einfacher Interpolation, (c) synchronisiert mittels einfacher Interpolation nach Entfernung der Ausreißer durch α-winsorisierte Standardabweichung, (d) synchronisiert mittels einfacher Interpolation nach Entfernung der Ausreißer mittels eines eingegebenen Schwellwerts.

dargestellt. Hierbei sind auf der x-Achse die MIDI-Einsatzzeiten eingetragen und auf der y-Achse die entsprechenden durch den Extraktionsalgorithmus berechneten Einsatzzeiten der PCM-Datei. (Bei einem gleichmäßigen Tempo der beiden Versionen würde der Zeitfluss eine Gerade darstellen.)

Die Abbildung 5.15 (*) entspricht nicht den vorherigen Abbildungen 5.13 (a) und 5.14 (a), sondern bezieht sich nur auf die MIDI-Datei bei der vorgenommenen MW-Synchronisation. Abgebildet ist die auf Einsatzeiten und Tonhöhen reduzierte Klavierwalzendarstellung der MIDI-Einspielung. Hierbei wurde die Darstellung der Notenlängen unterdrückt, um so die Effekte bei der Synchronisation besser hervortreten zu lassen. Wir sprechen im Folgenden auch einfach von einer reduzierten Klavierwalzendarstellung. Bei den Bereichen mit den stärksten zeitlichen Abweichungen bei der Synchronisation handelt es sich um die Zeitintervalle von 2.5 – 3 Sekunden (Mordent) und 9 – 12 Sekunden (Triller), sowie um den letzten Akkord in der zeitlichen Position von 16 Sekunden.

In Abb. 5.13 (b) ist neben der Tempofunktion aus (a) (punktierte Kurve) auch die durch Interpolation entstandene neue Tempofunktion (durchgezogene Linie) dargestellt. Der entsprechende neue Zeitfluss ist in Abb. 5.14 (b) zu sehen. Weiterhin zeigt Abb. 5.15 (b) die zugehörige reduzierte Klavierwalzendarstellung der entsprechend synchronisierten MIDI-Datei.

Abb. 5.13 (c) zeigt die modifizierte Tempofunktion (durchgezogene Linie), die mittels Eliminierung von „Ausreißern“ berechnet wurde. Hierbei wurde die α-winsorisierte Standardabweichung der Tempofunktion berechnet, die in beiden Richtungen um den α-winsorisierten Mittelwert zwei Schwellwerte bestimmen. Alle „Ausreißer“, die außerhalb dieser Grenzen liegen, wurden zuerst eliminiert. Anschließend wurde dann das gewöhnliche Interpolationsverfahren durchgeführt. Der resultierende Zeitfluss ist in Abb. 5.14 (c) und die reduzierte Klavierwalzendarstellung der so synchronisierten Datei in Abb. 5.15 (c) dargestellt.

Die Abbildungen 5.13 (d), 5.14 (d) und 5.15 (d) zeigen die entsprechenden Ergebnisse für die Korrektur der Tempofunktion mittels eines festgelegten Schwellwerts für das Tempoverhältnis, gegeben durch den Wert 1.4. (Der Wert 1 entspricht hierbei zwei tempomäßig identischen Interpretationen. Der Wert 2 für das Tempoverhältnis gibt an, dass eine der Interpretationen doppelt so schnell wie die andere gespielt ist usw.).

Zusammenfassend lässt sich für dieses Beispiel sagen, dass der MW-Algorithmus ohne Korrektur der Ausreißer die beiden Dateien bis auf die kritische Zeitspanne mit dem Triller erfolgreich synchronisiert. Die Anwendung der ersten Schwellwertmethode behebt das Problem der Synchronisation im Bereich des Trillers, modifiziert aber unerwünschterweise die Einsatzzeiten des Mordents. Die Korrektur der Tempofunktion durch einen vorgegebenen Schwellwert, der deutlich über der α-winsorisierten Standardabweichung liegt, liefert schließlich ein sehr gutes Synchronisationsergebnis. Im Hinblick auf die in der Einleitung dargestellten Anwendungen sind aber kleine lokale Abweichungen und Fehler bei der Synchronisation, wie z. B. bei Trillerpassagen, akzeptabel, solange die globale Synchronisation gewährleistet ist. In diesem Sinne leistet der MW-Algorithmus sogar mehr als gefordert. Erzeugt er doch in diesem Beispiel aus der mechanischen MIDI-Abspielung der ersten Takte der Aria eine abspielbare MIDI-Version, die nun wesentlich der realen Interpretation ähnelt. Nur die stark verzerrte Trillerpassage zwischen der neunten und elften Sekunde (im Fall (b)) trüben den Kunstgenuss.

5.6 Weitere Ergebnisse und Diskussion der Experimente

Zum Test der verschiedenen Synchronisationsalgorithmen wurden zum einen Abschnitte aus handelsüblichen CD-Aufnahmen und MIDI-Aufnahmen verwendet. Zum anderen wurden speziell für diese Arbeit zahlreiche Aufnahmen von Klavierstücken in unterschiedlichen Versionen und auf verschiedenen Instrumenten (MIDI-Klavier, Schimmel-Klavier, Steinway-Flügel) gemacht. Hierbei wurden teilweise auch Versionen mit extremen Temposchwankungen, rhythmischen Verfremdungen und absichtlichen Fehlern erzeugt. Typischerweise handelt es sich bei den Testaufnahmen um polyphone Klavierstücke, bei denen neben komplexen Notenkombinationen wie z. B. den Verzierungen auch lokale, interpretatorisch bedingte Temposchwankungen auftreten. Für die Tests wurden Musikabschnitte einer Länge von 10 Sekunden bis 60 Sekunden ausgewählt.

Im Folgenden haben wir stellvertretend für dieses ganze Paket an Testdaten fünf typische Beispiele ausgewählt und diskutieren die jeweiligen Synchronisationsergebisse. Hierbei handelt es sich um die ersten Takte der *Aria con Variazioni* aus den Golbergvariationen (BWV 988) von J. S. Bach, den Anfang der 22. Variation des gleichen Werks, die ersten Takte des Trios aus W. A. Mozarts Klaviersonate KV 331 A-Dur „Alla Turca“ und schließlich die ersten Abschnitte der Etüde Nr. 2 (*Arabeske*) und der Etüde Nr. 1 (*Aufrichtigkeit*) Op. 100 von F. Burgmüller. Die Art der Aufnahme sowie die genaue musikalische und physikalische Dauer der einzelnen Abschnitte zeigt folgende Tabelle (T1 = Takt 1):

Nr.	Musikstück	Art der Aufnahme	musikalische Dauer	physikalische Dauer
1.	Bach/Aria	Steinway-Flügel	T1 – T4$\frac{1}{3}$	16.23 s
2.	Bach/Var. 22	E-Piano	T1 – T8	10.75 s
3.	Mozart/Trio	CD-Aufnahme	T1 – T7$\frac{1}{3}$	12.46 s
4.	Burgmüller/Etüde	Steinway-Flügel	T1 – T10, m.W.	15.65 s
5.	Burgmüller/Etüde	Steinway-Flügel	T1 – T8, m.W.	37 s

In der folgenden Tabelle findet man einen Überblick über Matching-Ergebnisse für die fünf ausgewählten Musikstücke. Hierbei wurden beide der in Abschnitt 5.5 beschriebenen Kriterien bei der Wahl für ζ berücksichtigt: das erste Wahlkriterium, das zum längsten Match der Länge $|\mu|$ mit minimaler Standardabweichung σ führt, und das zweite Wahlkriterium, welches den Quotienten $|\mu|/\sigma$ minimiert. Die Spalten $|\mu|/p$ (beim PM- und PW-Matching) bzw. $|\mu|/m$ (beim MW-Matching) zeigen das Verhältnis der Kardinalität des optimalen Matches μ und der Anzahl der quantisierten Partitur- bzw. MIDI-Ereignisse (p bzw. m). Da die Synchronisationsverfahren für sechs verschiedene Werte von ζ durchgeführt werden, führt dies zu einem sechsfachen Rechenaufwand (600 %) bei der Berechnung der Kostentabellen. Die Spalte *Recheneffizienz* zeigt, inwieweit dieser Rechenaufwand unter Verwendung des „Streifenverfahren“ (siehe Abb. 5.11) gesenkt werden kann. So werden z. B. im ersten Fall im Schnitt weniger als ein Sechstel aller Einträge der Kostentabellen berechnet, was zu einem Gesamtaufwand von 97.88 % (im Vergleich zu 600 %) führt, was eine wesentliche Effizienzsteigerung darstellt.

Nr.	1. Wahlkriterium			2. Wahlkriterium			Rechen-Effizienz	opt. Komb.	
	ζ	$\lvert\mu\rvert/p$ oder $\lvert\mu\rvert/m$	σ	ζ	$\lvert\mu\rvert/p$ oder $\lvert\mu\rvert/m$	σ		WK	SM.
1.	ζ_4	29/40	0.21	ζ_4	29/40	0.21	97.88%	1, 2	2
2.	ζ_2	33/35	0.12	ζ_2	33/35	0.12	131.41%	1, 2	0, 2
3.	ζ_3	37/44	0.22	ζ_3	37/44	0.22	93.88%	1, 2	1, 2
4.	ζ_4	63/68	0.39	ζ_4	63/68	0.39	77.61 %	1, 2	2
5.	ζ_5	112/122	0.55	ζ_5	112/122	0.55	42.86 %	1, 2	2

Wir gehen nun der Reihe nach auf die einzelnen Beispiele ein und diskutieren die erzielten Synchronisationsergebnisse – insbesondere auch in Abhängigkeit von der jeweiligen Parameterwahl. Das Beispiel der Aria wurde bereits sehr ausführlich im vorigen Abschnitt besprochen. Da sich in diesem Beispiel die lokalen zeitlichen Temposchwankungen im Rahmen halten, stellt sich für beide Kriterien bei der Wahl von ζ der Wert $\zeta = \zeta_4$ (entspricht einer Abweichung von 1 Sekunde) als optimal heraus. In der Tabelle steht dabei in der vorletzten Spalte WK = 1 abkürzend für das erste und WK = 2 für das zweite Wahlkriterium. Wie schon erwähnt, stellen bei der Synchronisation der Aria die Verzierungen die größten Schwierigkeiten dar. In Abhängigkeit von der Schwellwertmethode ergibt sich in diesen Bereichen eine unterschiedliche Interpolation der nicht-gematchten (oder impliziten) Notenereignisse. In der Tabelle steht in der letzten Spalte SM = 0 dafür, dass keine Detektion und Eliminierung der Ausreißer bei der Synchronisation stattgefunden hat. SM = 1 bedeutet, dass die α-winsorisierte Standardabweichung als Schwellwert eingesetzt wurde, und SM = 2 bedeutet, dass ein festgelegter Tempofaktor als Schwellwert eingesetzt wurde. Im Beispiel der Aria lieferten die beiden Methoden SM = 1 und SM = 2 ähnlich gute Ergebnisse. Die mechanisch erzeugte MIDI-Datei klingt nach der Synchronisation der realen Interpretation sehr nah und das Synchronisationsergebnis kann – bezüglich eines subjektiven Bewertungsmaßstabes – als erfolgreich angesehen werden.

Auch im zweiten Beispiel wurde durch den automatischen MW-Matchingalgorithmus eine sehr gute Synchronisation der zu verlinkenden Dateien erzielt. Den Erfolg kann man zum einen auf die guten Extraktionsergenisse zum anderen auf die nur geringen Temposchwankungen der Interpretation zurückführen. Für beide Wahlkriterien stellte sich $\zeta = \zeta_3$ als optimal heraus. In diesem Beispiel haben die Interpolationen zu SM = 0 (keine Schwellwertmethode) und SM = 2 zu ähnlich guten Erbenissen bezüglich einer subjektiven Bewertung geführt.

Das dritte Beispiel stellt eine Interpretation der Mozartsonate von R. Castro dar, bei der im Trio ziemlich große lokale Temposchwankungen auftreten und insbesondere der Anfang des Trios extrem verzögert gespielt wurde. Bei der Synchronisation ohne Ausreißerkorrektur (SM = 0) kam es insbesondere am Anfang des Trios zu fehlerhaften Matchingergebnissen. Allerdings „renkte“ sich der MW-Algorithmus nach einigen Noten ein und erzielte im nachfolgenden Teil eine akzeptable Synchronisation. Durch den Einsatz von Schwellwertmethoden (SM = 1 und SM = 2) wurden die Matchingergebnisse deutlich verbessert. Die synchronisierte MIDI-Datei ähnelte nun tempomäßig bis auf die zweite Einsatzzeit sehr stark der realen Interpretation.

Das vierte und das fünfte Beispiel wurden rhythmisch stark verfremdet, um so zu testen, wie der PW- bzw. MW-Matchingalgorithmus auf Extrembeispiele reagiert. Bei den beiden Beispielen handelt es sich um den ersten mit Wiederholung gespielten Abschnitt der jewei-

ligen Etüde. Hierbei wurde in Beispiel 4 das Tempo kontinuierlich erhöht, gegen Ende des Abschnitts wieder verlangsamt und in der Wiederholung ähnlich verfahren. Beim Matching traten einige Fehler auf, die zu einem Teil auf eine „unreine“ Extraktion zurückzuführen waren. Mittels eines konstanten Schwellwerts (SM = 2) konnte das Matchingergebnis verbessert werden, so dass insbesondere die erste Hälfte des Abschnitts relativ gut synchronisiert wurde. In der zweite Hälfte sind einige lokale „Holpereien“ im Tempoverlauf der synchronisierten MIDI-Datei zu hören. Der Grund liegt vermutlich in einer „unglücklichen“ Kombination der Extraktionsfehler und der starken Tempoverzögerung in der PCM-Version über ein längeres Zeitintervall hinweg.

Im fünften Beispiel handelt es sich bei der Klaviereinspielung (der PCM-Version) um eine durch Punktierung rhythmisch stark verfremdete Version des Originalmusikstücks. Der MW-Algorithmus erzielte nicht nur ein – global gesehen – gutes Synchronisationsergebnis, sondern konnte meistens auch die rhythmischen Verfremdungen einfangen, auch wenn er an manchen Stellen lokal scheiterte. Aufgrund der extremen lokalen Verzerrungen wurden die besten Matchingergebnisse diesmal für die Wahl $\zeta = \zeta_5$ erzielt, was der zeitlichen Abweichungen von 2 Sekunden entspricht. Die Beseitigung von „Ausreißern“ mit der Schwellwertmethode $SM = 1$ hat sich in diesem Beispiel als völlig ungeeignet erwiesen, da die durch die Rhythmisierung verfremdeten Notenereignisse nicht mehr von den durch ein fehlerhaftes Verfahren verursachten Ausreißern unterschieden werden konnten. In diesem Beispiel führte die Schwellwertmethode $SM = 2$ mit einem großen Schwellwert von 2.5 (der natürlich von der speziellen Rhythmisierung in diesem konkreten Beispiel abhängt) zum besten Synchronisationsergebnis.

Zusammenfassend lässt sich hinsichtlich der Parameterwahl und der Effizienz folgendes Fazit ziehen:

- Bei der Synchronisation kommt es vorweigend zu lokalen Zeitabweichungen, die sich zwischen 0.3 und einer Sekunde bewegen. Diese Tatsache könnte man zur weiteren Effizienzsteigerung benutzen, indem man die Werte für τ und ζ weiter einschränkt.

- Bei starken Rhythmus- oder Temposchwankungen erweist sich die α-winsorisierte Standardabweichung als Schwellwert (SM = 1) als ungeeignet, da hier die zeitlich verschobenen Notenereignisse von den Ausreißern nicht mehr unterscheidbar sind.

- Das erste Kriterium (WK = 1) und das zweite Kriterium (WK =2) bei der Wahl von ζ liefern in etwa gleich gute Synchronisationsergebnisse.

- Je länger die zu synchronisierenden Musikabschnitte sind, desto effizienter ist das „Streifenverfahren“ gegenüber der Berechnung der kompletten Kostentabellen.

5.7 Resumé und Ausblick

Wir beschließen das Kapitel mit einem Resumé und einem kleinen Ausblick. Ziel dieser Arbeit war es, Verfahren zur automatischen Synchronisation verschiedener Versionen eines Musikstücks in unterschiedlichen Formaten (Partitur, MIDI, PCM) zu entwickeln und einen Prototypen zu implementieren und zu testen. Die Komplexität dieser Aufgabenstellung ist

insbesondere auf die folgenden zwei Schwierigkeiten zurückzuführen. Zum einen ist typischerweise eines der zu synchronisierenden Musikstücke als CD-Aufnahme in Wellenformdarstellung gegeben (z. B. dem PCM-Dateiformat). Um solche Daten mit partiturähnlichen Daten überhaupt vergleichbar und algorithmisch zugänglich zu machen, müssen in einem Extraktionsschritt erst einmal aus den PCM-Daten Notenereignisse extrahiert werden. Eine vollständige Extraktion der Notenereignisse – im Sinne der Musiktranskription – stellt insbesondere für polyphone Musik eine noch ungelöste Aufgabe dar. In unserem Fall begnügen wir uns auf die Extraktion einer hinreichend großen Menge an Notenereignissen, die zumindest eine Synchronisation gewährleisten soll. Allerdings hat man im Allgemeinen mit vielen Extraktionsfehlern und Extraktionsungenauigkeiten zu kämpfen. Die zweite Schwierigkeit besteht darin, dass es in den konkreten Musikaufnahmen nicht nur durch die interpretatorischen Freiheiten zu lokalen Temposchwankungen sondern machmal auch zu erheblichen Abweichungen vom eigentlichen Notentext kommt - man denke hier nur an Trillerereignisse, Arpeggien oder falsch und ungenau gespielte Noten. Der Matching-Algorithmus muss also einerseits robust gegenüber den falsch extrahierten Notenereignissen sein, andererseits soll er aber sensibel gegenüber den lokalen Temposchwankungen sein, die es ja gerade zu erfassen gilt. Darüber hinaus soll sich aber der Matching-Algorithmus nicht durch interpretatorisch bedingte Abweichungen vom Notentext zu sehr „verwirren“ lassen.

Es ist nur zu offensichtlich, dass diese Anforderungen widersprüchlicher Natur sind. Der von uns entwickelte MW-Matchingalgorithmus versucht nun einen Mittelweg zu gehen. Durch Einführung des Konzepts der Fuzzy-Note wurden interpretatorische Abweichungen, die der Notentext bei Trillern oder Arpeggien zulässt, modelliert. Durch die dynamische Programmierung wurde eine Robustheit des MW-Matchings gegenüber den Extraktionsfehlern und den falsch oder ungenau gespielten Noten erzielt. Hierbei ist eine wesentliche Leistung dieser Arbeit die Bereitstellung eines geeigneten Matching-Begriffs und die Definition einer sinnvollen Kostenfunktion. Hierbei stellt die Kostenfunktion sicher, dass auf der einen Seite möglichst viele der Notenereignisse gematcht werden (Maximierung von $|\mu|$). Allerdings verhindert auf der anderen Seite die zeitliche Komponente der Kostenfunktion, dass es bei den Matches zu zu großen und damit unwahrscheinlichen relativen zeitlichen Verschiebungen bei der Synchronisation der beiden Datenströme kommt. Mit anderen Worten, es wird auf einen Match „verzichtet“, wenn dieser zu extremen lokalen Zeitverzerrungen führen sollte. Die Freiheiten, die man bei der Wahl des Parametervektors π der Kostenfunktion hat, sowie die verschiedenen Kriterien bei der Bestimmung der optimalen Parameteres ζ und unterschiedlichen Schwellenwertverfahren lassen genügend Raum zu Experimenten. Eine automatische Bestimmung dieser Parameter wurde ansatzweise im letzten Abschnitt diskutiert.

Das von uns entwickelte Verfahren zur Synchronisation liefert im Hinblick auf die in der Einleitung dieses Kapitels beschriebenen Anwendungen gute Ergebnisse. Zwar kommt es aufgrund der oben beschriebenen Extraktionsfehler öfters zu falschen Matchingergebnissen, allerdings handelt es sich hierbei meist um zeitlich „lokalisierte“ Fehler. Selbst in Extremsituationen, wie z. B. den diskutierten Trillerpassagen und den rhythmischen Verfremdungen, stabilisiert sich der Synchronisationsalgorithmus wieder nach kurzer Zeit und erzielt „global“ gesehen immer noch eine akzeptable Synchronisation und Verlinkung, die bei Anwendungen wie der Partiturlesehilfe oder partiturbasierten Suche in Musikpassagen ausreichend ist.

Unser Prototyp wurde in MATLAB implementiert und für zahlreiche Musikbeispiele unterschiedlichen Datenformats getestet. Hierbei handelte es sich bei den Testdaten typischerweise

um Abschnitte von Klavierstücken der Länge 10 bis 60 Sekunden. Hinsichtlich der Laufzeit stellt der Extraktionsschritt der Notenereignisse den Flaschenhals unseres Verfahrens dar. Die extrem langen Filter der kaskadierten Multiratenfilterbänke führen zu langen Laufzeiten. Die Laufzeiten der Matching-Algorithmen sind insbesondere auch durch den Einsatz des „Streifenverfahren“ vergleichsweise gering. Für die Synchronisation längerer Musikabschnitte oder ganzer Musikstücke empfiehlt sich folgendes „Top-Down“-Verfahren. In einem ersten Schritt werden die beiden zu verlinkenden Dateien nach „sicheren“ Notenereignissen durchsucht, wie zeitlich isolierte Töne (z. B. Einsätze nach längere Pausen), Akkorde oder in einer Tonlage isolierte Töne, die sich spektral von ihrer Umgebung abheben und daher leicht und sicher extrahiert werden können. Diese „sicheren“ Notenereignisse können dann verlinkt werden und stellen die Stützpfeiler einer Synchronisation dar. Bei den Passagen zwischen diesen Stützpfeiler handelt es sich dann um kürzere Musikabschnitte, die auf die zuvor beschriebene Weise synchronisiert und verlinkt werden können.

Wie die Diskussion der Ergebnisse in den Abschnitten 5.5 und 5.6 zeigt, geht das von uns vorgestellte Synchronisationsverfahren über die in der Einleitung dieses Kapitels formulierten Anforderungen hinaus. In vielen der behandelten Beispiele gewährleistete das MW-Matching nicht nur eine gute zeitliche Synchronisation, sondern war sogar in der Lage, kleine interpretatorische Temposchwankungen zu erfassen. Um das Synchronisationsergebnis auch akustisch nachvollziehbar zu machen, wurde z. B. beim MW-Matching eine mechanisch klingende, uninterpretierte MIDI-Version entsprechend der Synchronisation mit einer realen Interpretation (vorliegend als CD-Aufnahme) zeitlich modifiziert. Auf diese Weise wurde eine abspielbare MIDI-Version erzeugt, die nun im zeitlichen Verlauf dieser realen Interpretation ähnelte und der auf diese Weise förmlich Leben eingeflößt wurde. In einem weiteren Schritt könnte man nun versuchen, auch die Notenlängen und den dynamischen Verlauf der realen Aufnahme zu extrahieren, um mit diesen Daten die MIDI-Version weiter zu verfeinern und zu einer lebendigen Interpretation zu machen. Eine weitere Vision ist, mit Hilfe der extrahierten Daten und der durch den MW-Matchingalgorithmus hergestellten Synchronisation unter Zuhilfenahme von statistischen Daten den jeweiligen Interpreten der Aufnahme automatisch zu schätzen.

Literaturverzeichnis

[1] AKoff Sound Labs: *Akoff Musik Composer.* http://www.akoff.com.

[2] Araki Software: *AmazingMIDI.* http://www.pluto.dti.ne.jp/ araki/amazingmidi/

[3] Ballou, G.: *Handbook for Sound Engineers.* Focal Press, 1998.

[4] Blackham, E.D.: *Klaviere.* Die Physik der Musikinstrumente, 2. Auflage, Spectrum, Akademischer Verlag, 1998.

[5] Blatter, C.: *Wavelets – Eine Einführung.* Vieweg 1998.

[6] Bobrek, M.: *Polyphonic Music Segmentation using Wavelet Based Tree-Structured Filter Banks with Improved Time-Frequency Resolution.* PhD Dissertation, August 1996.

[7] Bobrek, M., Koch, D.: *Music Signal Segmentation Using Tree-Structured Filter Banks.* Journal of Audio Engineering Society, Vol. 46, No. 5, May 1998.

[8] Canazza, S., Roda, A., Orio, N.: *A Parametric Model of Expressiveness in Musical Performance Based on Perceptual and Acoustic Analysis.* ICMC Proceredings, 1999.

[9] Cano, P., Loscos, A., Bonda, J.: *Score-Performance Matching using HMMs.* ICMC Proceedings 1999, pp.441-444.

[10] Cemgil, A. T., Desain, P., Kappen, B.: *Rhythm Quantisation for Transcription.* Computer Music Journal, 24(2), Summer 2000, pp. 60-76.

[11] Cemgil, A. T., Kappen, B., Desain, P., Honing, H.: *On tempo tracking: Tempogram Representation and Kalman filtering.* ICMC Proceedings 2000, pp. 352-355.

[12] Clausen, M.: *Schnelle problemsensitive Indexerstellung zur effizienten Mustersuche in grossen Datenbanken.* Manuskript, 2000.

[13] Clausen, M., Baum, U.: *Fast Fourier Transforms.* BI Wissenschaftsverlag 1993.

[14] Clausen, M., Engelbrecht, R., Meyer, D., Schmitz, J.: *PROMS: A Web-based Tool for Searching in Polyphonic Music.* Proceedings Intl. Symp. on Music Information Retrieval 2000, Plymouth, M.A., USA.

[15] Cormen, T. H., Leiserson, C. E., Rivest, R. L.: *Introduction to Algorithms.* MIT Press, 1990.

[16] Crochiere, R. E., Rabiner, L. R.: *Multirate Digital Signal Processing.* Prentice-Hall Signal Processing Series, 1983.

[17] Dannenberg, R. B.: *An On-Line Algorithm for Real-Time Accompaniment.* Proceedings of ICMC, 1984.

[18] Dannenberg, et al.: *Automated musical accompaniment with multiple input sensors.* US Patent #5521324, 1994. http://www.uspto.gov/patft/index.html

[19] Dannenberg, R. B., Mukaino, H.: *New Techniques for Enhanced Quality of Computer Accompaniment.* Proceedings of ICMC, 1988.

[20] De Poli, G., Piccialli, A., Roads, C.: *Representations of Musical Signals.* MIT Press, 1991.

[21] Deller, J. R., Proakis, J.G., Hansen, J.H.L.: *Discrete-Time Processing of Speech Signals.* Prentice Hall, 1993.

[22] Desain, P., Honing, H.: *Tempo curves considered harmful.* Time in contemporary musical thought, J. D. Kramer (ed.), Contemporary Music Review, 7(2), 1993, pp. 123-138. http://www.nici.kun.nl/mmm/papers/dh-93-f-1.html

[23] Desain, P., Honing, H.: *Physical motion as a metaphor for timing in music: the final ritard.* ICMA Proceedings, 1996. http://www.nici.kun.nl/mmm/papers/dh-96-e.html

[24] Desain, P., Honing, H., Heijink, H.: *Robust Score-Performance Matching: Taking Advantage of Structural Information.* ICMC Proceedings 1997, pp. 377-340. (oder http://stephanus2.socsci.kun.nl/mmm/ppers/dhh-97-a.html)

[25] Dixon, S.: *Automatic Extraction of Tempo and Beat from Expressive Performances.* Journal of New Music Research, 30, 1, 2001. http://www.ai.univie.ac.at/ simon/

[26] Fletcher, N. H., Rossing, T. D.: *The Physics of Musical Instruments.* Springer-Verlag, 1991.

[27] Folland, J. B.: *Real Analysis.* John Wiley & Sons, 1984.

[28] Foote, J.: *ARTHUR: Retrieving Orchestral Music by Long-Term Structure.* Proceedings of the International Symposium on Music Information Retrieval, Plymouth, Massachusetts, October 2000.

[29] Foote, J.: *Automatic Audio Segmentation Using a Measure of Audio Novelity.* Proceedings of IEEE International Conference on Multimedia and Expo, vol. I, 2000, pp. 452-455. http://www.fxpal.com/people/foote/papers/allpapers.html

[30] Foote, J., Uchihashi, S.: *The Beat Spectrum: A New Approach To Rhythm Analysis.* In Proc. International Conference on Multimedia and Expo (ICME) 2001 http://www.fxpal.com/people/foote/papers/allpapers.html

[31] Foster, S., Schloss, W.A., Rockmore, A.J.: *Toward an Intelligent Editor of Digital Audio: Signal Processing Methods.* Computer Music Journal, Vol. 6, No. 1, Spring 1982.

[32] Galembo, A., Askenfeld, A.: *Signal Representation and Estimation of Spectral Parameters by Inharmonic Comb Filters with Application to the Piano.* IEEE Transactions on Speech and Audio Processing, Vol. 7, No. 2, March 1999.

[33] Goebl, W.: *Skilled Piano Performance: Melody Lead Caused by Dynamic Differentiation.* In C. Woods & G. Luck & R. Brochard & F. Seddon & J. A. Sloboda (Eds.), Proceedings of the 6th international conference on music perception and cognition (pp. 1165-1176). Keele, UK: Keele University Department of Psychology, 2000 (http://www.ai.univie.ac.at/ wernerg/).

[34] Goebl, W., Parncutt, R.: *Perception of onset asynchronies: Acoustic piano versus synthesized complex versus pure tones.* Meeting of the Society for Music Perception and Cognition (SMPC2001), 9-11 August 2001, pp. 21-22. (http://www.ai.univie.ac.at/ wernerg/).

[35] Goto, M., Muraoka, Y.: *A Real-time Beat Tracking System for Audio Signals.* Proceedings of the 1995 International Computer Music Conference, pp.171-174, September 1995. http://staff.aist.go.jp/m.goto/PROJ/bts.html

[36] Goto, M: *An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds.* Journal of New Music Research, Vol.30, No.2, pp.159-171, June 2001. http://staff.aist.go.jp/m.goto/PROJ/bts.html

[37] Heijink, H.: *Matching Scores and Performances.* Master Theses.

[38] Heijink, H., Desain, P., Honing, H., Windsdor, L.: *Make Me a Match: An Evaluation of Different Approaches to Score-Performance Matching.* Computer Music Journal, Volume 24, Issue 1, Spring 2000.

[39] Hempel, C.: *Allgemeine Musiklehre.* Atlantis-Schott, Serie Musik, 1997.

[40] Kahrs, M., Brandenburg, K.: *Applications of Digital Signal Processing to Audio and Acoustics.* The Kluwer International International Series in Engineering and Computer Science, 1998.

[41] Kaiser, J.: *Beethovens 32 Klaviersonaten und ihre Interpreten.* Fischer Taschenbuch Verlag, 1984.

[42] Klapuri, A., Virtanen, T., Holm, J.: *Robust Multipitch Estimation for the Analysis and Manipulation of Polyphonic Musical Signals.* In Proc. COST-G6 Conference of Digital Audio Effects, DAFx-00, Verona, Italy, 2000. http://www.cs.tut.fi/ klap/iiro/

[43] Klapuri, A., Virtanen,T., Eronen, A., Seppänen, J.: *Automatic Transcription of Musical Recordings.* Consistent&Reliable Acoustic Cues Workshop, CRAC-01, Aalborg, Denmark, September 2001. http://www.cs.tut.fi/ klap/iiro/

[44] Klapuri, A.: *Web-Seite.* http://www.cs.tut.fi/ klap/iiro/index.html

[45] Kleijn, W. B., Paliwal, K.K.: *Speech Coding and Synthesis.* Elsevier, 1995.

[46] Kurth, F., Clausen, M.: *Adaptive FBT and M-Band Wavelet Packet Algorithms in Audio Signal Processing.* TSP-Journal, Vol. 49, No. 2, 1999, pp. 549-554.

[47] Kühn, C.: *Formlehre der Musik.* Münchrn (dtv) und Kassel (Bärenreiter) 1998, 5. Aufl.

[48] Large, E. W.: *Dynamic programming for the analysis of serial behaviours.* Behaviour Research Methods, Instruments, & Computers. 1993.

[49] Lehmann, E., Riethmüller, T., Straßburg, H.: *Das SoundBlaster-Profibuch.* Addison-Wesley, 1993.

[50] Marlot, M.: *A comparision of feed forward neural network architectures for piano music transcription.* ICMC Proceedings 1999.

[51] Martin, K. D.: *Automatic Transcription od Simple Polyphonic Music: Robust Front End Processing.* M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 399. Presented at the Third Meeting of the Acoustical Societies of America and Japan, December 1996.

[52] martin, K. D.: *A Blackboard System for Automatic Transcription of Simple Polyphonic Music.* M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 385.

[53] Martin, K. D.: *Sound-Source Recognition: A Theory and Computational Model.* PhD Dissertation, June, 1999.

[54] Mathews, M.: *The Ear and How It Works.* In Music, Cognition and Computerized Sound, edited by P. R. Cook, The MIT Press 1999.

[55] Mazzola, G: *Geometrie der Töne. Elemente der Mathematischen Musikteorie.* Birkhäuser Verlag, 1990.

[56] McNab, R. J., Smith, L. A., and Witten, I. H.: *Signal Processing for Melody Transcription.* Proceedings of the 19^{th} Australasian Computer Science Conference, Melbourne, Australia, January 31 - February 2, 1996.

[57] Moorer, J. A.: *On the Segmentation and Analysis of Continuous Musical Sound by Digital Computer.* PhD thesis, Department of Music, Standford University, Standford, CA, May, 1975.

[58] Michels, U.: *dtv-Atlas Musik.* Deutscher Taschenbuch Verlag, 1997.

[59] *MIDI Manufacturers Associacion Web-Seite.* http://www.midi.org

[60] *MiDiLiB Web-Seite.* http://www-mmdb.iai.uni-bonn.de/

[61] Nikias, C. L.: *Riding the New Integrated Media System Wave.* The Past, Present and Future of Multimedia Signal Processing. IEEE Signal Processing Magazine, July 1997.

[62] Noll, J.: *Musik-Programierung.* Addison-Wesley, 1994.

[63] Noll, P.: *MPEG Digital Audio Coding.* IEEE Signal Processing Magazine, September 1997.

[64] Ortiz-Berenguer, L. I., Casajús-Quirós, F. J.: *Pattern recognition of piano chords based on physical model.* AES Convention Paper, Presented at the 112th Convention, 2002 May 10-13, Munich, Germany

[65] Oppenheim, A.V, Willsky, A.S.: *Signale und Systeme.* VCH Verlagsgesellschaft, 1989

[66] *DVD-Audio explained.* Panasonic Web-Seite http://www.panasonic.com/

[67] Parncutt, R.: *Harmony: A Psychoacoustical Approach.* Springer Verlag, 1989.

[68] Piszczalski, M., Galler, B. A.: *Predicting musical pitch from component frequency ratios.* Journal of Acustic Society of America, Vol. 66, No. 3, September 1979.

[69] Pohlmann, K. C.: *Principles of Digital Audio.* McGraw-Hill, Inc., 1995.

[70] Proakis, J. G., Manolakis, D. G.: *Digital Signal Processing.* Prentice Hall, 1996.

[71] Quatieri, T. F., McAulay, R. J.: *Audio Signal Processing Based on Sinusoidal Analysis/Synthesis.* [40], pp. 343-416.

[72] Raphael, C.: *Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models.* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 4, April 1999.

[73] Raphael, C.: *A Probabilistic Expert System for Automatic Musical Accompaniment.* Jour. of Comp. and Graph. Stats. Vol. 10 No. 3, pp. 487-512, 2001. http://fafner.math.umass.edu/papers/index.html

[74] Rasch, R. A.: *The Perception of Simultaneous Notes as in Polyphonic Music.* ACUSTICA, Vol. 40, 1978.

[75] Rasch, R. A.: *Synchronization in Performed Ensemble Music.* ACUSTICA, Vol. 43, 1979.

[76] Rasch, R. A., Plomp, R.: *The Perception of Musical Tones.* The Psychology of Music, Academic Press, 1982.

[77] Roads, C.: *The Computer Music Tutorial.* MIT Press, 1996.

[78] Rocchesso, D., Scalcon, F.: *Bandwidth of Percieved Inharmonicity for Physical Modeling of Dispersive Strings.* IEEE Transactions on Speech and Audio Processing, Vol. 7, No. 5, September 1999.

[79] Rothgeb, J.: *Simulating Musical Skills by Digital Computer.* Machine Models of Musik, ed. Schwanauer, S. M. und Levitt, D.A., MIT Press, 1993.

[80] Russel, D. A.: *The piano hammer as a nonlinear spring.* http://www.kettering.edu/ drussell/Piano/NonlinearHammer.html

[81] Sauter, F.: *Die tonale Musik. Anatomie der musikalischen Ästhetik.* Verlag Franz Sauter, Hamburg 2000.

[82] Sakoe H., Chiba, S.: *Dynamic Programming Algorithm Optimization for Spoken Word Recognition.* IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-26, No. 1, February 1978.

[83] Scheirer, E. D.: *Extracting Expressive Performance Information from Recorded Music.* Unpublished M. S. thesis, MIT Media Laboratory, 1995 http://web.media.mit.edu/ eds/thesis.pdf

[84] Scheirer, E. D.: *Tempo and Beat Analysis of Acoustic Musical Signals.* J. Acoust. Soc. Am. 103 (1) (Jan 1998), pp. 588-601. http://web.media.mit.edu/ eds/beat.pdf

[85] *Synchronized Multimedia.* http://www.w3.org/AudioVideo/

[86] Strang, G., Nguyen, T.: *Wavelets and Filter Banks.* Wellesley-Cambrige Press, 1996

[87] Vaidyanathan, P. P.: *Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial.* Proc. IEEE, Vol. 78, No. 1, pp. 56-93, Jan. 1990.

[88] Vaidyanathan, P. P.: *Multirate systems and filter banks.* Prentice Hall, 1993

[89] Veldhuis, R., Kohlrausch, A.: *Waveform Coding and Auditory Masking.* Elsevier, 1995, ch. 11, pp. 397-431.

[90] Vercoe, B.: *The Synthetic Performer in the Context of Live Performance.* Proceedings of ICMC, 1984.

[91] Watkinson, J.: *The Art of Digital Audio.* Oxford: Focal Press, 1994.

[92] Weinreich, G.: *Gekoppelte Schwingungen von Klaviersaiten.* Die Physik der Musikinstrumente, 2. Auflage, Spectrum, Akademischer Verlag, 1998.

[93] Zwicker, E., Fastl, H.: *Psychoacoustics.* Springer, 1990.