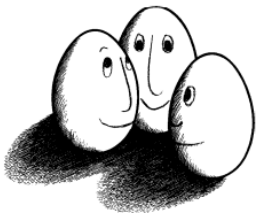
Bachelorarbeit

Entwicklung eines Annotierungswerkzeuges für Musikdaten

Niklas Reppel



Bachelorarbeit
Fakultät Informatik
Technische Universität Dortmund

Dortmund, 05. November 2011

Betreuer:

Prof. Dr. Katharina Morik
Dipl.-Inform. Marco Stolpe

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Ziel der vorliegenden Arbeit ist die Entwicklung eines Annotierungswerkzeuges für Musikdaten. Zweck dieses Werkzeugs ist die Unterstützung der vergleichenden Interpretationsforschung, d.h. dem wissenschaftlichem Vergleich unterschiedlicher Interpretationen eines Musikstücks bezüglich bestimmter Attribute wie z.B. Spieldynamik, Expressivität oder semantischen Konnotationen. Annotierung bedeutet hier, manuell einen bestimmten Abschnitt einer (Wave-) Audiodatei festzulegen (bzw. zu markieren) und diesen mit einer Annotation, d.h. einem bestimmten Attribut, zu belegen. Die Attribute sind bereits zuvor festgelegt. Von der annotierenden Person wird somit nur der Abschnitt markiert und die Annotation zugewiesen. Wenn im Folgenden von „Abschnitt" die Rede ist, bezieht sich dies auf einen Abschnitt in einer Audiodatei, d.h. einer einzelnen Interpretation eines Musikstücks, wohingegen „Stelle" sich auf die Partiturstelle, d.h. dem Abschnitt in einer übergeordneten, symbolischen Repräsentation (z.B. dem Notentext), bezieht.
Die Besonderheit des Werkzeuge ist, das die Annotierung „interpretationsübergreifend" erfolgen soll. Wurde eine Stelle in einer Interpretation annotiert, soll diese Annotation auch auf die selbe Stelle in anderen Interpretationen desselben Stücks angewendet werden. Hierfür wird ein entsprechender Algorithmus benötigt, der die annotierte Stelle in anderen Interpretationen wiederfindet und die Annotierung dort anwendet.
Da es im Hinblick auf den kognitiven Anspruch einfacher ist, einen Zeitpunkt

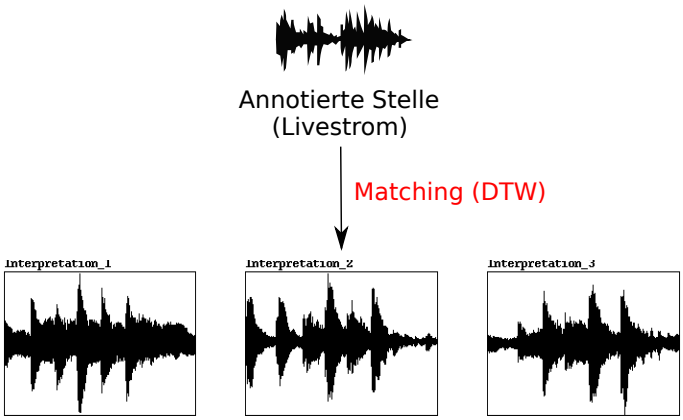


Abbildung 1: Interpretationsübergreifendes Annotieren

zu annotieren als einen Zeitraum mit Anfangs- und Endpunkt zu markieren, soll auch die Länge des annotierten Abschnitts vor der eigentlichen Annotierung festgelegt werden, so dass die Annotierung mit einem einzigen Klick erfolgen kann. Da die Grösse des Zeitraumes je nach Art und Tempo des Stücks variieren kann, sollte die Länge des Abschnitts einfach zu Konfigurieren sein. Dies impliziert, dass auch der Matching-Algorithmus mit Queries unterschiedlicher Länge funktionieren muss.
Dabei soll die Annotierung zunächst in Echtzeit auf einem Audiodatenstrom geschehen. Obwohl die Annotierung auf jeder Art von Audiodatenstrom erfolgen könnte, soll der Fokus auf live gespielten Interpretationen von Klavierstücken als Quelle liegen.
Nach erfolgter Annotierung soll die entsprechende Stelle aus jeder zuvor ange-

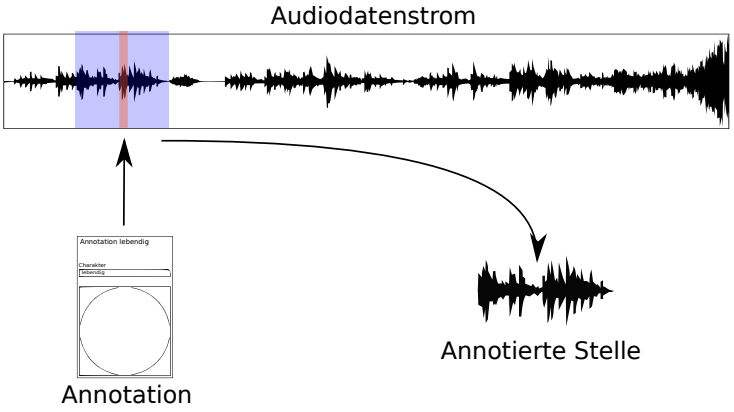


Abbildung 2: Annotierung aus dem Audiodatenstrom

gebenen Interpretation in Form einer kurzen Audiodatei verfügbar sein. So wird eine Basis an Vergleichsdaten geschaffen. Diese Daten können anschließend anhand verschiedener Merkmale bzgl. ihrer Gemeinsamkeiten und Unterschiede ausgewertet werden. Die Auswertung ist jedoch nicht Teil dieser Arbeit.
Ein Kernelemente der Arbeit ist also die Entwicklung einer grafischen Benutzerschnittstelle, die den Annotierungsprozess weitestgehend intuitiv ermöglicht. Die Schnittstelle sollte für die annotierende Person intuitiv und ohne grosse Vorkenntnisse zu bedienen sein. Eine Annotierung soll per Klick auf eine Schaltfläche getätigt werden können. Des weiteren sollen die Elemente gross genug sein, um auch auf einem Touchscreen zu funktionieren. Dies bedingt, das die Knöpfe (bzw. Tastfelder) ausreichend groß und optisch gut unterscheidbar sein müssen. Außerdem sollte das Werkzeug einfach um neue Annotations-Schaltflächen

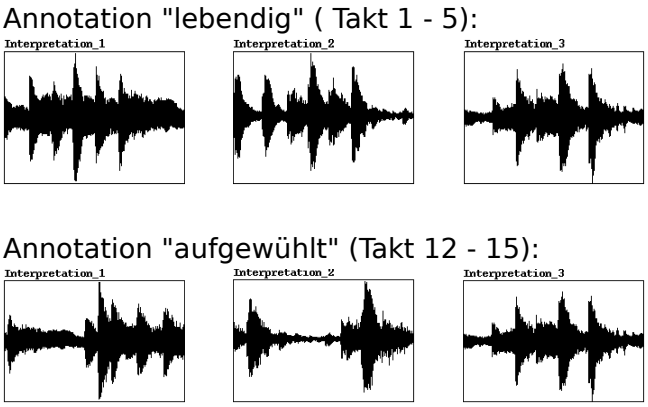


Abbildung 3: Vergleichsdaten

erweiterbar sein.
Das zweite Kernelement ist ein Matching- bzw. Suchverfahren, das die Annotierung über mehrere Interpretationen eines Musikstücks ermöglicht, d.h. die Stellen in anderen Interpretationen auffindet und extrahiert. Hierbei ist zu beachten, das es sich weniger um ein Information-Retrieval-Verfahren handelt denn um ein niederschwelliges Verfahren zur Suche ähnlicher Substrings. Kapitel zwei vermittelt hierzu zunächst Einblick in die verschiedenen Grundlagen sowie den Stand der Technik, auf dessen Basis das Matchingverfahren entwickelt werden soll. So werden zunächst einige grundlegende Techniken der digitalen Audiodatenverarbeitung dargelegt, sofern sie für das Verständnis der Arbeit wichtig sind. Anschließend werden grundlegende Techniken der Merkmalsextraktion aus Audiodaten detailliert behandelt, die ein wichtiger Vorverarbeitungsschritt für das Matchingverfahren sind. Auch den Ähnlichkeitsmassen für Audiodaten, die eine Teilmenge der Ähnlichkeitsmasse auf Zeitreihen bilden, kommt ein größerer Abschnitt zu. Insbesondere liegt hier ein Augenmerk auf der Dynamic Time Warping-Technik (kurz DTW), die als Suchverfahren im zu entwickelnden Werkzeug zum Einsatz kommen soll. Dabei wird zunächst (zum Verständnis) das klassische DTW vorgestellt, um anschließend die Varianten zur Online- und Offline-Substringsuche zu behandeln.
Vor der eigentlichen Implementierung sollen die unterschiedlichen Kombinationen aus DTW- und Merkmalsextraktionsverfahren im Hinblick auf ihre Tauglichkeit für das zu entwickelnde Werkzeug evaluiert werden. Dabei wird zum einen getestet, ob die manuell annotierten Abschnitte (in diesem Kontext „Queries" genannt) überhaupt und wenn, mit welcher Güte, d.h. mit welcher zeit-

lichen Abweichung vom erwarteten Wert, sie in den anderen Interpretationen gefunden werden. Eine detaillierte Beschreibung des Testverfahrens sowie die Testergebnisse finden sich im dritten Kapitel.
Im vierten Kapitel wird die Implementierung des Werkzeugs, insbesondere der grafischen Benutzerschnittstelle dokumentiert
Abschließend folgt eine Zusammenfassung der Ergebnisse sowie ein Ausblick auf zukünftige Arbeiten.

2 Audiodatenverarbeitung

2.1 Grundlagen

Im folgenden sollen einige grundlegende Techniken zur digitalen Audiodatenverarbeitung (Digital Sound Processing, kurz DSP) kurz dargelegt werden, sofern sie zum Verständnis der behandelten Algorithmen und Frameworks wichtig sind.

2.1.1 Datenformat

Die Audiodaten liegen im PCM-Format digitalisiert vor, d.h. das eingehende analoge Audiosignal wird in bestimmten Zeitabständen abgetastet und auf eine Dynamikstufe quantisiert (das Ergebnis eines Abtastpunkts wird in diesem Zusammenhang auch „Sample" genannt). Die Qualität der Digitalisierung wird von den Faktoren Samplerate, d.h. Abtastpunkte pro Sekunde (Einheit: Hz), und Auflösung, d.h. Dynamikstufen pro Abtastpunkt (Einheit: Bit), bestimmt. Ein Sample entspricht somit einer Gleitkommazahl, die die dynamische Auflösung in einer bestimmten Präzision, z.B. 16 Bit, wiedergibt. Der darstellbare Frequenzraum hängt von der Samplerate ab. Das Nyquist-Shannon-Abtasttheorem besagt (vereinfacht), das die höchste darstellbare Frequenz der Hälfte der Samplerate, also der halben Abtastfrequenz, entspricht (vgl. [16]). Bei 44100Hz Samplerate lassen sich demnach Frequenzen bis 22050Hz darstellen, was den menschlichen Hörbereich, der im Allgemeinen zwischen 16 und 20000Hz (vgl. [3]) liegt, abdeckt.
Die derzeit am häufigsten verwendete Qualität liegt bei 16 Bit Auflösung und 44100Hz Samplerate, d.h. 44100 16-Bit Float-Werte pro Sekunde (und Kanal). Dies entspricht der Qualität einer normalen Audio-CD, was sicher die Verbreitung dieses Formats erklärt. Zu beachten ist, das es sich hierbei um unkomprimierte Audiodaten handelt. Da im folgenden nur auf diesen gearbeitet wird,

werden komprimierte Audiodaten (z.B. im Mp3- oder Ogg Vorbis-Format) hier nicht näher behandelt.
Sind im zu digitalisierenden Signal Frequenzen enthalten, die höher als die höchste darstellbare Frequenz liegen, können Artefakte entstehen, wenn eine Schwingungsperiode vollständig zwischen zwei Abtastpunkten liegt. Artefakte sind ungewollte Störgeräusche, die bei der Umwandlung des digitalen Signals in ein analoges durch Fehlinterpretation solcher Abtastpunkte entstehen. Solche Nebeneffekte sind insbesondere vor Downsampling-Operationen, d.h. der Reduzierung der Samplerate, zu beachten. Vor solchen Operationen sollten die höheren Frequenzen mit einem Tiefpassfilter aus dem Signal herausgefiltert werden.
Es bleibt zu bemerken, das das so digitalisierte Audiosignal auch als Zeitreihe über Gleitkommazahlen aufgefasst werden kann.

2.1.2 Verarbeitung in Blöcken

Die Verarbeitung der oben beschriebenen Audiodaten wäre Sample für Sample (sprich, Gleitkommazahl für Gleitkommazahl) sehr rechenintensiv und schwierig umzusetzen. Die meisten Verfahren benötigten eine bestimmte Anzahl an gelesenen bzw. gepufferten Samples, auf denen sie Arbeiten können (z.B. sämtliche FFT-basierten Verfahren, vgl. nächster Abschnitt).
Deshalb ist die Denkweise verbreitet, die gepufferten Samples als *Block* aufzufassen. Digitale Audiodatenverarbeitung findet somit in den meisten Fällen in kurzen Abschnitten (Blöcken) gleicher Länge statt. Eine bestimmte Anzahl Samples (zumeist eine Zweierpotenz) wird zu einem Block zusammengefasst. Einige Verfahren liefern präzisere Ergebnisse, wenn sich die Blöcke überlappen, z.B. um einen halben oder dreiviertel Block. Diese Denkweise geht auf Arbeiten zur Verwendung der schnellen Fouriertransformation in den 1960er Jahren zurück (vgl. [21]). Alternativ lässt sich dies als eine (gleitende) Fensterung mit einer Rechteckfunktion formulieren.
Die so entstehenden Blöcke sind somit nichts anderes als Vektoren von Gleitkommazahlen. Die Zeitreihe von Gleitkommazahlen wird somit zu einer Zeitreihe von Vektoren.

2.1.3 FFT

Als Grundlage für viele Algorithmen in der digitalen Audioverarbeitung werden die Vektoren (Blöcke) zunächst mit Hilfe der schnellen Fouriertransformation

(Fast Fourier Transformation - FFT) vom Zeit- in den Frequenzraum transformiert, d.h. in einzelne Frequenzbänder zerlegt. So lässt sich die Energieverteilung über dem Frequenzspektrum analysieren (z.B. durch optisch durch Sonargramme, d.h. grafische Darstellungen der Energieverteilung). Soll ein Algorithmus ein „klingendes" Ergebnis haben, d.h. der Veränderung eines digitalen Audiosignals dienen, lässt sich die Transformation nach Manipulation der Energieverteilung auch Umkehren (IFFT - Inverse Fast Fourier Transformation). Da das Verfahren im folgenden jedoch nur zu analytischen Zwecken verwendet wird, wird die inverse Transformation hier nicht näher behandelt.

2.1.3.1 Transformation Da im Rahmen dieser Arbeit nur mit zeitdiskreten Signalen gearbeitet wird, soll die Darstellung auf die eindimensionale Diskrete Fourier-Transformation (DFT) beschränkt bleiben. Eine DFT transformiert eine Folge über $X = (x_0, x_1 ... x_n)$ in eine andere Folge $Y = (y_0, y_0 ... y_n)$ gleicher Länge mit $N \in \mathbb{N}$ und $x_n, y_n \in \mathbb{C}$ für alle $n \in [0 : N]$ und ist ist mathematisch definiert als:

$$y_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{k}{N} n}$$

Intuitiv wird die Folge vom Zeit- in den Frequenzraum transformiert, d.h. in der transformierten Folge entspricht jedes Element der Amplitude auf einem bestimmten Frequenzband. Quadriert man die Amplitude, erhält man die Energie auf dem jeweiligen Frequenzband.
Zur effizienten Berechnung sind verschiedene Verfahren entwickelt worden, die

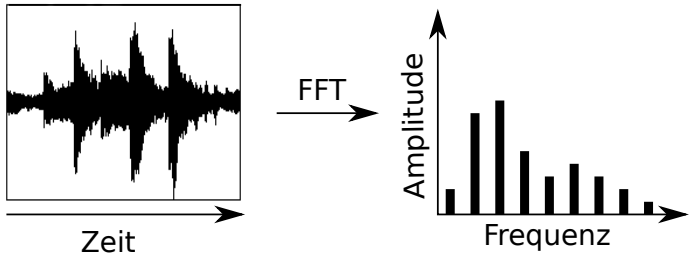


Abbildung 4: Fourier-Transformation

meist unter dem Begriff Schnelle Fourier-Transformation (Fast Fourier Transformation, FFT) subsumiert werden. Das bekannteste Verfahren wurde 1965 von Cooley und Tukey entdeckt (bzw. wiederentdeckt, nachdem die Idee schon um 1805 von Carl Friedrich Gauss entwickelt wurde, vgl. [15]). Es handelt sich

hierbei um einen Divide-and-Conquer-Algorithmus, der die Transformation der Folge X rekursiv in Transformationen von Teilfolgen aufteilt. Zumeist wird die Folge X der Länge N dabei in Teilfolgen der Länge $N/2$ geteilt. In diesem Fall ist die Transformation auf Folgen beschränkt, dessen Länge eine Zweierpotenz ist. Wohl nicht zuletzt aufgrund der Verbreitung dieses Verfahrens arbeiten viele Audioprogramme nicht heute mit Zweierpotenzen als Block- bzw. Puffergröße.
Es existieren inzwischen jedoch weitere Berechnungsverfahren, die nicht auf diese Restriktion angewiesen sind (vgl. [6]). Die meisten heute quelloffenen verfügbaren Implementierungen greifen je nach Eingabe automatisch auf die entsprechende Berechnungsmethode zurück.

2.1.3.2 Leakage-Effekt Ein bekanntes Problem der FFT ist der sogenannte *Leakage-Effekts*. An den Rändern eines Audioblocks entsteht (durch die Faltung mit einer Rechteckfunktion) ein gewisser Anteil an Rechteckschwingungen. Eine Rechteckschwingung ist auch als Gemisch vieler Frequenzbänder darstellbar. Bei einer Fouriertransformation „streuen" („leaken") diese Frequenzen, wodurch eine unsaubere Impulsantwort entsteht. Um dem entgegenzuwirken, wird der

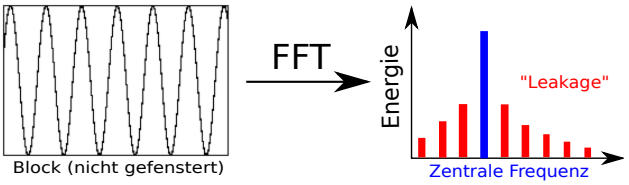


Abbildung 5: Leakage-Effekt

jeweilige Audioblock vor der Transformation mit einer (diskreten) Fensterfunktion multipliziert. Zumeist wird hier ein Von-Hann-Fenster (Bei Fensterbreite M definiert als: $w(n) = 1/2 * [1 + cos(2\pi n/M)]$ für $n \in [-M/2, +M/2]$) oder ein Hamming-Fenster ($w(n) = 0,54 + 0,46 * cos(2\pi n/M)$ für $n \in [-M/2, +M/2]$) verwendet. Während ersteres die Streuung „in die Breite" stärker unterdrückt, hat letzteres eine bessere Fokussierung des zentralen Frequenzbandes bei etwas breiterer Streuung zur Folge (vgl. [8]). Zwar kann der Leakage-Effekt dadurch nicht vollständig verhindert, jedoch deutlich reduziert werden.

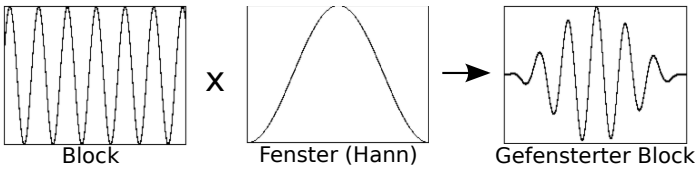


Abbildung 6: Faltung mit einer Fensterfunktion

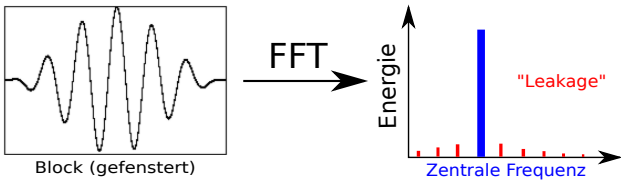


Abbildung 7: Reduzierter Leakage-Effekt

2.1.4 Filter

Anstatt mit einer FFT lassen sich Audiosignale auch mit Hilfe von Filtern zur Analyse in einzelne Frequenzbereiche zerlegen. Intuitiv dargestellt unterdrückt ein Filter bestimmte Frequenzbereiche eines Audiosignals und erhält den Rest (bzw. hebt ihn unter Umständen hervor). So lässt sich mit einer Verkettung von Bandpassfiltern, die jeweils nur einen schmalen Frequenzbereich passieren lassen, ein ganz ähnlicher Effekt erzielen wie mit einer Fouriertransformation. Mathematisch lassen sich (digitale) Filter, sofern sie gewissen Bedingungen bzgl. der Stabilität, Invarianz und Linearität gehorchen, als Faltung des zu filternden diskreten Signals x mit dem ebenfalls diskreten Filter-Signal y auffassen. Diese Faltung ist an einer Stelle $n \in \mathbb{Z}$ definiert als:

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

Da eine detaillierte Beschreibung der Filter-Thematik den Rahmen dieser Arbeit sprengen würde, sei hier auf die Literatur (z.B. Kapitel 2.3 in [15] oder Kapitel 8 in [16]) verwiesen.

2.2 Merkmalsextraktion

Für Audiomatching und -synchronisierungsaufgaben ist als Vorverarbeitung ein Merkmalsextraktionsverfahren nötig, das von den rohen Audiodaten abstrahieren und so das Matching sinnvoll ermöglicht. Bei den im folgenden präsentierten Verfahren handelt es sich um Merkmale, die für Synchronisierungsaufgaben und niederschwellige Suchaufgaben entwickelt wurden. Diese sind von semantisch orientierten Verfahren zu unterscheiden, die eher für Klassifizierungs- und Kategorisierungsverfahren (z.B. Genreklassifizierung) entwickelt worden sind. Bezüglich letzterer seien z.B. die Arbeiten über automatisch konstruierte Methodenbäume zur Merkmalsextraktion von Ingo Mierswa und Katharina Morik ([14]) genannt. Ebenfalls in diese Kategorie fallen die Tonhöhen-Histogramm-Darstellung von Tzanetakis et al. ([20]) oder die Performance-Worm-Plots von Simon Dixon ([4]). Da das in dieser Arbeit zu entwickelnde Werkzeug für niederschwellige Aufgben konzipiert ist, sei diesbezüglich nur auf die Literatur hingewiesen.
Idealerweise sollen die extrahierten Merkmale musikalisch sinnvoll sein. Ein intuitiver Ansatz wäre daher, die Audiodaten zunächst in symbolische Musikdaten (Noten, MIDI, MusicXML etc.) zu Transformieren und dort eine Substring-Suche durchzuführen. Ein solcher High-Level-Ansatz ist jedoch nicht unproblematisch. Die Transformation von Audiodaten in symbolische Musikdaten gilt, insbesondere bei polyphoner Musik, allgemein nicht unbedingt als zuverlässig (vgl. [4]) und wäre relativ Speicher- und Rechenintensiv. Dies ist insbesondere bei Online-Matching-Anwendungen von Nachteil. Hier gilt es also, Merkmale zu finden, die zum einen effizient und schnell berechenbar sind, zum anderen jedoch die Struktur der Musik (d.h. das Verhältnis von Frequenzspektrum, Tonsystem und menschlicher akustischer Wahrnehmung) berücksichtigen. Weiterhin muss das Verfahren robust genug sein, um mit stark differierenden Interpretationen eines Stücks fertig zu werden. Störfaktoren wie Rauschen oder Brummen sollten möglichst nicht ins Gewicht fallen.
Generell werden niederschwellige Ansätze bevorzugt, die das Audiosignal zunächst in einzelne Frequenzbänder zerlegen (bzw. in den Frequenzraum transformieren), um anschließend die Frequenzbänder auf verschiedene Arten zu aggregieren oder zu selektieren. Diese Repräsentationen sind also niederschwelliger als eine symbolische Repräsentation oder eine semantische Annotierung. Im folgenden sollen die wichtigsten Verfahrend detailliert beschrieben werde. Für eine weitere Übersicht sei auf Kapitel 3 in [15] verwiesen. Des weiteren finden sich in [4, 9, 13, 2] Ansätze, die im Folgenden detaillierter beschrieben werden.

2.2.1 Vorverarbeitung

Verschiedene Arten der Vorverarbeitung dienen zumeist dazu, weniger wichtige Informationen bereits vor der Merkmalsextraktion zu herauszufiltern. So kann das Audiosignal beispielsweise mit verschiedenen Filtern bearbeitet werden, um Frequenzbänder, die für die jeweilige Anwendung wenig Informationsgehalt haben, herauszufiltern. Generell sind Frequenzen über 12000Hz für die meisten Menschen kaum noch zu unterscheiden und sind auch für den musikalischen Informationsgehalt von geringerer Bedeutung (die Grundfrequenz des höchsten Ton eines Klaviers liegt bei ca. 4186Hz, d.h. erst dessen höhere Obertöne reichen in solche Höhen, vgl. [3]). Deshalb schlagen einige Verfahren (vgl. [15]) die Verwendung einer Samplerate von 22050Hz vor, d.h. Frequenzen über 11025Hz werden gar nicht gespeichert. Da Audiodaten jedoch weitaus häufiger mit einer Samplerate von 44100Hz (CD-Qualität) gespeichert vorliegen, wäre ein Downsampling-Schritt notwendig sowie, um Artefakte durch das Downsampling zu vermeiden, eine vorherige Tiefpassfilterung. Ein ähnlicher Effekt lässt sich auch erzielen, indem die höheren Frequenzbänder eines fouriertransformierten Signals verworfen werden.

2.2.2 Zerlegung in Frequenzbänder

Allen im Folgenden dargestellten Verfahren ist gemein, das das Audiosignal zunächst in den Frequenzraum transformiert wird, d.h. das Audiosignal wird über die Amplituden auf einer bestimmten Menge an Frequenzbändern repräsentiert. Während die meisten Verfahren auf die schnelle Fouriertransformation zurückgreifen, wird in [15] ein filterbasierter Ansatz vorgestellt.

2.2.2.1 Zerlegung mit FFT Die gängigste und weithin bekannte und in mannigfaltigen Implementierungen verfügbare Methode zur Transformation und Zerlegung eines Audiosignals ist die schnelle Fourier-Transformation, auf die im vorherigen Kapitel bereits eingegangen wurde. Problematisch ist, das sich die Frequenzbänder des FFT-Resultats linear über das Frequenzspektrum verteilen, während sich die Grundfrequenzen der Töne (bezogen auf die gleichstufig temperierte Stimmung der westlichen Musik) exponentiell (bezüglich der Oktave) über das Frequenzspektrum erstrecken.
Ein weiteres Problem ist, das die Frequenzbänder des FFT-Resultats gerade im Bassbereich oft weiter auseinander liegen als die Grundfrequenzen der Töne (vgl. [15]). Bei einer Blockgrösse von 2048 Samples beträgt die Breite der Frequenzbänder des Transformierten Blocks 10.77Hz. Der Abstand von E3

(164,814Hz) und F3 (174,614Hz) beträgt schon nur noch 9,7Hz. Ab diesem Bereich ist eine Abbildung und somit eine Aggregation auf musikalisch sinnvolle Frequenzbänder also kaum mehr möglich ist.
Zudem wird die Zerlegung mit kleinerer Blockgrösse gröber und unmusikalischer. Möchte man dennoch ein FFT-basiertes Verfahren nutzen, bietet sich eine heuristische Lösung (siehe *Heuristischer Ansatz*) sowie die Verwendung von Blockgrössen ≥ 2048 an.

2.2.2.2 Zerlegung mit Bandpassfiltern

Um den Problemen der FFT zu begegnen, bietet sich als Alternative die Zerlegung in Frequenzbänder durch eine Bandpassfilterbank (d.h. eine Verkettung von Bandpass-Filtern) an. Bei diesem Ansatz (beschrieben in [15]) wird das Audiosignal in 88 Teilsignale (Subbänder) zerlegt. Die zentralen Frequenzen der 88 (schmalbandigen) Filter entsprechen den Grundfrequenzen der 88 Töne eines Konzertflügels.
Das Verfahren arbeitet mit variablen Sampling-Frequenzen. In einem Vorverarbeitungsschritt wird das Signal zunächst in Bassbereich (bis 441Hz), Mittenbereich (bis 2205Hz) und Höhenbereich (bis 11025Hz) zerlegt, die jeweils mit eigener Sampling-Frequenz gespeichert (882Hz, 4410Hz und 22050Hz) werden. Hierzu wird das Originalsignal stufenweise mit einem entsprechenden Tiefpass-Filter gefiltert, um Artefakte zu vermeiden, und anschließend um den entsprechenden Faktor (5x) downgesampled. So wird letztlich der Speicheraufwand reduziert, da unwichtige bzw. redundante Information verworfen wird. Im anschließenden Schritt werden die eigentlichen Bandpassfilter angewendet. Die Bandbreite wird der Tonhöhe entsprechend angepasst (tiefere Töne $\rightarrow$ schmaleres Frequenzband, höhere Töne $\rightarrow$ breiteres Frequenzband).
Nachteil dieses Verfahrens ist zum einen der höhere Speicheraufwand, da jedes Subband einen bestimmten Speicherbedarf hat, der nur unwesentlich geringer als der des (downgesampleten) Originalsignals ist. Durch den zusätzlichen Vorverarbeitungsschritt ist der Aufwand des Verfahrens insgesamt höher. Zudem kommt in der Originalarbeit ein Verfahren zum Einsatz, das als *forward-backward-filter* bekannt ist, um Phasenverschiebungen bzw. -verzerrungen zu vermeiden. Dieses Verfahren setzt jedoch die Kenntnis des gesamten Ausgangssignals voraus und ist somit (in seiner Grundform) nicht für datenstromorientierte Anwendungen geeignet.

2.2.3 Merkmale

2.2.3.1 Tonhöhenbasierte Merkmale

Tonhöhenbasierte Merkmale basieren auf der Messung der Verteilung der Schallenergie über das Frequenzspektrum

(weshalb im Folgenden auch die Bezeichnung *lokale Energie(-verteilung)* bzw. *lokale Amplitude* verwendet wird). Anders formuliert wird die Amplitude oder Energie auf jenen Frequenzbändern gemessen, die im Kontext des gleichstufig temperierten westlichen Tonsystems sinnvoll sind. Diese können Tonhöhen (bzw. Grundfrequenzen der jeweiligen Töne) sein, die den 88 Tasten eines Konzertflügels entsprechen, oder auch eine Abbildung auf MIDI-Tonhöhen (vgl. [9]).
Bei einem mit Hilfe einer Fourier-Transformation zerlegten Signal werden die FT-Komponenten auf die jeweils am nächsten liegende Tonhöhe (d.h. die Grundfrequenz des entsprechenden Tons) aufaddiert und somit die (lineare) Frequenzbandeinteilung der FT auf die (logarithmische) Frequenzbandeinteilung des westlichen Tonsystems abgebildet (vgl.[4]). Möchte man die Energie messen, werden die Frequenzbänder zunächst quadriert. Im Bassbereich treten die oben beschriebenen Probleme auf.
Bei einem mit einer Filterbank zerlegten Signal (vgl. [15]) wird die lokale Energie (Short Time Mean Square Power) auf den jeweiligen Subbändern in bestimmten Zeitabständen gemessen, d.h. auf jedem Subband-Signal x wird alle d Samples ein (Rechteck-)Fenster der Grösse w wie folgt ausgewertet:

$$\sum_{k \in [n - \lfloor \frac{w}{2} \rfloor : n + \lfloor \frac{w}{2} \rfloor]} |x(k)|^2$$

Über alle Subbänder ergibt sich also ein Feature-Vektor der Dimension $n = 88$, entsprechend den Tönen auf einem Konzertflügel. Wenn zu erwarten ist, das nicht das volle Spektrum verwendet wird, kann der Frequenzbereich und damit die Dimension des Vektors entsprechend eingeschränkt werden.
Ein solcher Merkmalsvektor sollte nicht mit einer Abbildung auf *Noten* und damit einer Transformation in symbolische Musikdaten verwechselt werden. Eine Note hat neben einer Tonhöhe immer eine Dauer. Da die Amplitude bzw. Energie in regelmäßigen Zeitabständen gemessen wird, findet die Dauer einer einzelnen Note hier keine Repräsentation (im Gegensatz zum Note-On/Note-Off bei MIDI-Daten). Zum anderen Besteht ein Ton nicht nur aus seiner Grundfrequenz, sondern auch aus Obertönen. In den oben beschriebenen Verfahren entspricht ein Element des Zielvektors jedoch nicht einem *Ton* mit sämtlichen Obertönen, sondern nur einer *Tonhöhe*. Durch die Obertöne würde sich also auch ein einzelner Ton über mehrere Elemente des Zielvektors erstrecken. Die so gewonnene Repräsentation lässt sich also zwischen den reinen Audiodaten und einer symbolischen Repräsentation ansiedeln, ist jedoch weitaus niederschwelliger als tatsächliche Noten oder MIDI.

2.2.3.2 Chroma-Merkmale Sollen die Merkmale robust gegenüber sehr unterschiedlichen Varianten eines Stücks (z.B. zur Synchronisierung von Klavierauszug und Orchesterfassung eines Stückes) sein, reicht eine Abbildung auf Tonhöhen nicht unbedingt aus. Um die Robustheit zu erhöhen, können die in einem vorherigen Schritt gewonnenen Tonhöhenmerkmale weiter auf *Tonhöhenklassen* aggregiert werden. *Tonhöhenklasse*, auch *Chroma*, heisst in diesem Zusammenhang, das die 12 Tonhöhen einer chromatischen Tonleiter unabhängig von der Oktave zusammengefasst werden, d.h. C0,C1,...,C8 gehören sämtlich zur Chroma-Klasse C. Zur Wahrnehmung von Tonhöhe und Chroma sei auf die Arbeit von R. Shepard verwiesen ([19]). Wichtige Arbeiten zur chromabasierten Merkmalsgewinnung finden sich bei Bartsch et al. ([2]) sowie in [15, 20, 9]. Einen Chroma-Merkmalsvektor erhält man, in dem man die jeweiligen Töne eines Pitch-Vektors (siehe vorheriges Kapitel) über alle Oktaven aufaddiert. So ergibt sich ein Chroma-Vektor der Dimension $n = 12$. Aufgrund der Aufad-

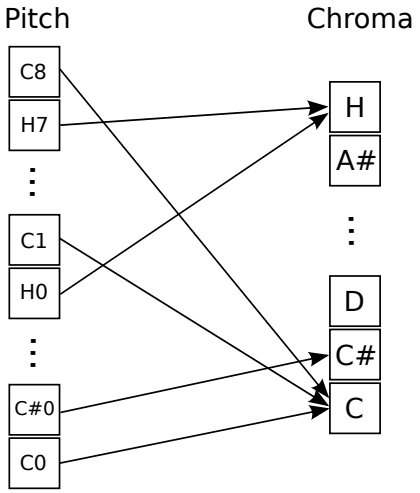


Abbildung 8: Pitch und Chroma

dierung bietet es sich hier an, den Chroma-Vektor zu normalisieren (vgl. [15]), z.B: anhand der l_1-Norm des Chroma-Vektors v, d.h.

$$v_{norm} = v/\|v\|_1, \|v\|_1 = \sum_{i=1}^{12} |v(i)|$$

Ein Beispiel für ein (erweitertes) Chroma-basiertes Verfahren ist das im Folgenden vorgestellte CNES-Verfahren.

2.2.3.3 MFCCs Mel Frequency Cepstral Coefficients (wie beschrieben von B. Logan in [13], vgl. auch [9]) sind an der Mel-Skala orientierte Merkmale. Die Mel-Skala (Einheit: Mel) verhält sich gegenüber der Hertz-Skala im Bassbereich linear, darüber hinaus jedoch logarithmisch und ist somit eher an der Frequenzwahrnehmung des menschlichen Ohres orientiert (vgl. [3]). Somit ist die Mel-Skala eine Maßeinheit für die *wahrgenommene* Tonhöhe.
Das Verfahren basiert auf einer FFT. Zunächst wird der Logarithmus der spektralen Komponenten berechnet. Danach werden die FFT-Komponenten anhand der Mel-Skala aggregiert (ähnlich wie bei der Lokalen Energieverteilung auf Halbtöne aggregiert wird). Schließlich werden die Komponenten mit einer diskreten Cosinus-Transformation dekorreliert.
Diese Art Merkmal orientiert sich an der Klangfarbe und ist damit eher für Spracherkennung oder nicht-musikalische Geräusche ausgelegt. Bei musikalischen Anwendungen lieferte dieses Merkmal in der Vergangenheit im Vergleich zu Pitch- und Chroma-Merkmalen eher schlechte Ergebnisse (vgl. [2]). Jedoch fliesst die Idee auch in den im folgenden beschriebenen heuristischen Ansatz ein, der den Bassbereich linear abbildet.

2.2.3.4 Heuristischer Ansatz Dieser Ansatz (beschrieben in [4]) ist prinzipiell ein tonhöhenbasiertes Verfahren, das die Fourier-Transformation zur Frequenzbandzerlegung verwendet. Um die beschriebenen Nachteile eines FFT-basierten Verfahrens zu kompensieren (im Bassbereich unter einer bestimmten Frequenz liegen die FFT-Frequenzbänder weiter auseinander als die Halbtöne und sind somit nicht eindeutig auf diese abbildbar), wird das Frequenzspektrum in drei Intervalle zerlegt. Das erste Intervall enthält den Bassbereich bis zu jener Frequenz, ab der die FFT-Frequenzbänder eindeutig auf Tonhöhen abbildbar sind. Bis zu dieser Grenze werden die Elemente des FFT-Vektors eins zu eins in den Zielvektor übernommen. Das zweite Intervall umfasst das sonstige „musikalische Spektrum", d.h. die Frequenzen bis zu einer Frequenz, ab welcher die Frequenzbänder weniger Informationsgehalt besitzen, weil sie innerhalb des musikalischen Spektrums kaum vorkommen (extreme Höhen). Innerhalb dieses Intervalls werden die FFT-Elemente in Halbtonschritten auf Tonhöhen abgebildet bzw. aufaddiert. Alle FFT-Elemente über der oberen Grenzfrequenz werden auf das letzte Element des Zielvektors abgebildet (alternativ können diese Frequenzen auch weggelassen werden).
Im Originaltext ist das Verfahren mit einer festen Blockgrösse (2048 Samples) und einem fest dimensionierten Zielvektor (n = 84) angegeben, die untere Frequenzgrenze ist bei 300Hz angesiedelt, die obere bei 12.5kHz. Mit steigender Blockgrösse und damit feinerer Zerlegung wird der Bassbereich präziser abgebildet. Somit kann die untere Grenzfrequenz bei grösseren Blockgrössen

Abbildung 9: Abbildungsverfahren

entsprechend tiefer angesiedelt werden. Des weiteren nähert sich das Verfahren (durch den linearen Bassbereich), ähnlich den MFCCs, der menschlichen Frequenzwahrnehmung an.

2.2.3.5 Selektion Statt sämtliche Frequenzbänder eines fouriertransformierten Audioblocks auf bestimmte Töne zu aggregieren, ist es ebenso denkbar, bestimmte Bänder zu selektieren und die restlichen zu Verwerfen. So könnte man nur die Frequenz selektieren, die der Grundfrequenz eines Tones am nächsten liegt, was jedoch gegenüber kleineren Stimmungsschwankungen wenig robust wäre. Sinnvoller wäre es also, die Frequenzbänder mit einem bestimmten Schwellwert, zum Beispiel 10Hz, um die Grundfrequenz eines Tones herum zu selektieren.

2.2.3.6 Anschlags-Merkmale Bei dieser Erweiterung (vgl. [15, 4]) wird nicht die (absolute) lokale Amplitude/Energie pro Zeiteinheit auf den jeweiligen Frequenzbändern gemessen, sondern der Energiezuwachs. So wird der Anschlag eines Tones das eigentliche Feature. Als Nebeneffekt fallen kontinuierliche Störfaktoren weniger ins Gewicht. Dies lässt sich als zusätzlicher Operator sowohl mit FFT-basierten als auch mit filterbasierten Pitch- und Chroma-Verfahren kombinieren. Für einen Feature-Vektor F der Dimension n an einer Stelle t im Signal x wird also die *half-wave rectified first order difference* für jedes Frequenzband f berechnet:

$$F_x^{'}(t,f) = max(F_x(t,f) - F_x(t-1,f), 0)$$

Ein Problem dieses Verfahrens ist die Unterscheidung zwischen tatsächlichen Anschlägen eines Tons und eventuellen Nebengeräuschen (z.B. Pedalgeräusche eines Klaviers). Eine grobe Unterscheidung kann ein Schwellwert liefern, d.h.

es werden nur Anschläge, die eine gewisses Energieniveau überschreiten, in Betracht gezogen.

2.2.3.7 Quantisierung Je nach gewählter Repräsentation ist die dynamische Auflösung der Merkmalsvektoren 16 oder sogar 32 Bit. Um die Merkmale robuster gegenüber dynamischen Schwankungen zu machen, können die Merkmalsvektoren *quantisiert* werden (vgl. [15]). Hierzu werden die gemessenen Energie- oder Amplitudenwerte auf eine bestimmte, wesentlich geringere Anzahl von Dynamikstufen abgebildet. Quantisierung lässt sich sowohl bei Pitch- wie auch bei Chroma-basierten Verfahren verwenden. Die Quantisierungsfunktion muss in Abhängigkeit vom Verfahren gewählt werden (Energie, Amplitude dB oder RMS, normalisiert/nicht-normalisiert etc.). Ein Beispiel für eine Quantisierungsfunktion findet sich in der Beschreibung des CNES-Verfahrens.

2.2.3.8 Kombiniertes Chroma Um den Problemen der FFT im Bassbereich zu begegnen und gleichzeitig den Grad der Robustheit zu erhöhen, kann das Chroma-Verfahren auch, analog zum bereits dargestellten heuristischen Ansatz, dergestalt angepasst werden, das der Bassbereich bis zu jener Frequenz, ab der die Frequenzen auf Halbtöne abbildbar sind, linear zu übertragen. Ab dieser Grenze werden die FFT-Komponenten bis zu einer oberen Grenzfrequenz auf Chroma-Bänder aggregiert (statt auf Halbtöne).

2.2.3.9 CNES Dieses in [15] vorstellte Verfahren (*Chroma Energy Normalized Statistics*, kurz CNES) verkettet einige der zuvor vorgestellten Verfahren. Zunächst wird der Tonhöhen-Vektor (Dimension $n = 88$, gewonnen durch filterbankbasierte Subband-Zerlegung) zu einem Chroma-Vektor v aufaddiert. Anschließend wird der Chroma-Vektor mit der l_1-Norm von v, d.h. $\|v\|_1 = \sum_{i=1}^{12} |v(i)|$ normalisiert und quantisiert. Die Quantisierungsfunktion $\tau : [0, 1] \rightarrow [0, 1, 2, 3, 4]$ lautet:

$$\tau(a) = \begin{cases} 0, \text{wenn } 0 \leq a < 0.05 \\ 1, \text{wenn } 0.05 \leq a < 0.1 \\ 2, \text{wenn } 0.1 \leq a < 0.2 \\ 3, \text{wenn } 0.2 \leq a < 0.4 \\ 4, \text{wenn } 0.4 \leq a \leq 1 \end{cases}$$

Die so entstehende Zeitreihe über Vektoren wird nun Vektor für Vektor mit ein einer Hann-Funktion der Länge **w** gefaltet. So stellt jeder Vektor nicht die

Chroma-Verteilung (bzw. quantisierte Chroma-Verteilung) dar, sondern ein gewichtetes Mittel über einen bestimmten Zeitraum. Anschließend wird die Reihe um einem Faktor **d** downgesampled, d.h. nur jeder **d**-te Vektor wird ausgewertet, die anderen Verworfen. Die Vektoren der so entstehenden Zeitreihe werden erneut bzgl. der Euklidischen Norm normalisiert. Bei einer initialen Blockgrösse von 200ms (8820 Samples bei einer Samplingrate von 44100Hz), einer Überlappung von einem halben Block, einer Fenstergrösse $\mathbf{w} = 41$ und einem Downsampling-Faktor $\mathbf{d} = 10$ erhält man so einen CNES-Feature-Vektor pro Sekunde Audiomaterial, der ca. 4410ms abdeckt. Es lässt sich also Feststellen, das das CNES-Verfahren um einiges stärker vom ursprünglichen Audiomaterial abstrahiert, als die bisher vorgestellten Verfahren. Zudem ist es in der oben genannten Konfiguration für Sequenzen größerer Länge ausgelegt. So ist ein Feature pro Sekunde für Anwendungen mit sehr kurzen Audio-Queries (Länge < 2s) viel zu grob.
Über die initiale Blockgrösse, die Fenstergrösse **w** und den Downsampling-Faktor **d** lässt sich die Granularität des Verfahrens jedoch dem jeweiligen Zweck anpassen.
Alternativ zum Originaltext lassen sich die zugrundeliegenden Chroma-Merkmale auch aus einem der oben vorgestellten, FFT-basierten Verfahren gewinnen.

2.3 Ähnlichkeitsmasse

Da es sich, wie bereits dargelegt, bei Audiodaten im Wesentlichen um Zeitreihen handelt, können die Ähnlichkeitsmasse, die für Zeitreihen im Allgemeinen entwickelt wurden, auch Anwendung bei der Ähnlichkeitsberechnung von Audiodaten finden.

2.3.1 Lineare Ähnlichkeitsmasse

Seien $X = (x_1, ...x_N)$ und $Y = (y_1, ...y_M)$ mit $N, M \in \mathbb{N}$ zwei Zeitreihen, so ist ein lineares Ähnlichkeitsmass bzw. ein lineares Alignment ein Mass, welches zum Zeitpunkt t jeweils x_t und y_t direkt (linear) aligniert und die Distanz anhand einer Metrik berechnet. Als Beispiel wäre hier die *Summe der quadrierten Differenzen* (wie beschrieben in [1]) zu nennen, die als $\mathcal{D}(X, Y) = (\sum_{t=0}^{n-1} |x_t - y_t|^2)^{1/2}$ definiert ist. Nach Parseval's Theorem ist dieses Distanzmass invariant gegenüber einer Transformation von Zeit- in den Frequenzraum. Das Problem unterschiedlich langer Sequenzen lässt sich hier schon erahnen. Weiterhin ergibt sich jedoch das Problem, das das direkte Alignment wenig robust gegenüber kleineren Streckungen und Verschiebungen ist (vgl. [10]).

Gerade bezogen auf die Synchronisierung von Audiodaten, wo sich einzelne Interpretationen von Stücken bzgl. Tempo und Dynamik sehr unterscheiden können, wird darum eher auf nicht-lineare Verfahren (wie das im folgenden beschriebene Dynamic Time Warping) zurückgegriffen.

2.3.2 Dynamic Time Warping

Um den oben beschriebenen Problemen eines linearen Distanzmasses zu begegnen, wurde in den 1970er-Jahren im Kontext der Spracherkennung die *Dynamic Time Warping*, kurz *DTW* bekannt bzw. entwickelt. Als grundlegende Arbeit sei hier der Text von Sakoe und Chiba ([17]) genannt. Als klassisches Übersichtswerk gilt weiterhin die Arbeit „Fundamentals of Speech Recognition" von Rabiner und Juang. Ein ebenfalls sehr umfassendes Übersichtswerk neueren Datums wurde von Meinhard Müller verfasst (Kapitel 4 in [15]).
Ziel dieser Technik ist es, ein (nicht-lineares) Mass zu finden, das Ähnlichkeit zwischen Zeitreihen besser wiedergibt als beispielsweise die euklidische Distanz. Weiterhin lässt sich DTW auch zur Synchronisierung von Audiodaten verwenden (vgl. [5], [15]).
Dynamic Time Warping (wie beschrieben in [15]) beschreibt sowohl ein Ali-

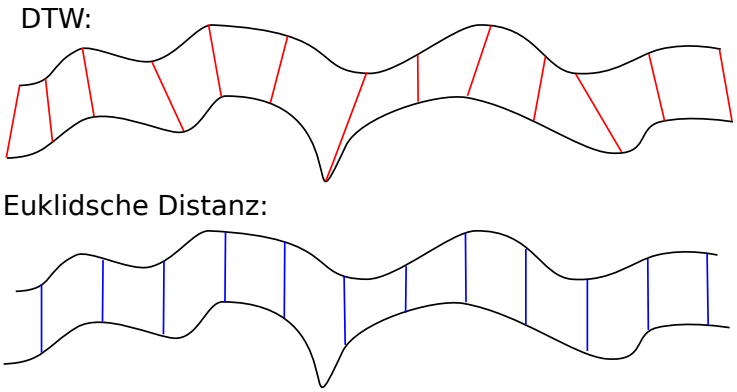


Abbildung 10: Dynamic Time Warping

gnment als auch ein Distanz- bzw. Ähnlichkeitsmass zwischen zwei Zeitreihen $X = (x_1, ...x_N)$, $Y = (y_1, ...y_M)$ mit $N, M \in \mathbb{N}$ und $x_n, y_m \in \mathcal{F}$ für alle $n \in [0 : N]$, $m \in [0 : M]$. Der Algorithmus zur Berechnung von DTW basiert auf Dynamischer Programmierung. Als Distanzmass ist DTW somit nahe

verwandt mit Ähnlichkeitsmassen für Strings, so zum Beispiel der Levenshtein-Distanz bzw. der Editierdistanz.

Definition 1 (Featurespace $\mathcal{F}$)**.** *Raum der Elemente, über welchem die Zeitreihen definiert sind.*

Im einfachsten Fall ist ein Element einer Zeitreihe ein numerischer Wert aus $\mathbb{R}$, d.h. $\mathcal{F} = \mathbb{R}$. Im Kontext des Audio-Matching sind die Zeitreihen zumeist (resultieren aus der Merkmalsextraktion) über Vektoren aus $\mathbb{R}^n$, $n \in \mathbb{N}$ definiert, d.h. $\mathcal{F} = \mathbb{R}^n$
Für den Vergleich einzelner Elemente aus $\mathcal{F}$ wird zunächst eine *Lokale Kostenfunktion* über $\mathcal{F}$ benötigt.

Definition 2 (Lokale Kostenfunktion c)**.** $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}$ *für die gilt:*

- *Definitheit:* $c(i, j) = 0 \iff i = j$
- *Symmetrie:* $c(i, j) = c(j, i)$
- *Dreiecksungleichung* $c(i, k) \leq c(i, j) + c(j, k)$

für alle $i, j, k \in \mathcal{F}$

Die tatsächliche Ausprägung von c hängt von der Beschaffenheit von $\mathcal{F}$ ab. Möglichkeiten wären beispielsweise die Euklidische Distanz oder die Manhattan-Distanz. Erstere böte sich insbesondere bei Zeitreihen über Vektoren an.
Mit Dynamic Time Warping kann ein *Warping-Pfad*, d.h. eine Zuordnung (Alignment) der Elemente aus X und Y, berechnet werden.

Definition 3 (Warping-Pfad)**.** *Sequenz von Indexpaaren* $p = (p_1, ..., p_L)$ *mit* $p_l = (n_l, m_l) \in [1 : N] \times [1 : M]$ *für* $l \in [1 : L]$ *die folgende Bedingungen erfüllt:*

- *Grenzbedingung:* $p_1 = (1, 1), p_l = (N, M)$
- *Monotonie:* $n_1 \leq n_2 \leq ... \leq n_L$, $m_1 \leq m_2 \leq ... \leq m_L$
- *Schrittgrösse:* $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1))\}$

Die Gesamtkosten $c_p(X, Y)$ eines Warping-Pfades p zwischen X und Y betragen demnach:

$$\sum_{l=1}^{L} c(x_{n_l}, y_{m_l})$$

Der *optimale Warping-Pfad* p^* ist der Warping-Pfad mit den geringsten Gesamtkosten aus allen möglichen Warping-Pfaden.

Über die Kosten des optimalen Warping-Pfades ist zudem das *Distanzmass* Dynamic Time Warping $DTW(X,Y)$ definiert.

Definition 4 (Dynamic Time Warping Distanz)**.** $DTW(X,Y) = c_{p^*}(X,Y) = min\{c_p|\ p\ ist\ ein\ Warping\text{-}Pfad\}$

Wird nur die DTW-Distanz zweier Zeitreihen benötigt, so genügt das Berechnen der *akkumulierten Kostenmatrix* D.

Definition 5 (Akkumulierte Kostenmatrix)**.** *Akkumulierte Kostenmatrix* $D \in \mathbb{R}^{M \times N}$

- $D(n,1) = \sum_{k=1}^{n} c(x_k, y_1)$ *für* $n \in [1:N]$
- $D(1,m) = \sum_{k=1}^{m} c(x_1, y_k)$ *für* $m \in [1:M]$
- $D(n,m) = min\{D(n-1,m-1), D(n-1,m), D(n,m-1)\} + c(x_n, y_m)$

Die Distanz $DTW(X,Y)$ lässt sich von $D(N,M)$ ablesen.
Der entsprechende optimale Warping-Pfad, d.h. das eigentliche Alignment, kann durch Backtracking auf D berechnet werden:
Beginn bei $p_l = (n,m)$,

$$p_i - 1 = \begin{cases} (1, m-1), \text{wenn } n = 1 \\ (n-1, 1), \text{wenn } m = 1 \\ argmin(D(n-1,m-1), D(n,m-1), D(n-1,m)), \text{ sonst.} \end{cases}$$

Sollte im dritten Fall *argmin* nicht eindeutig sein, wird das lexikografisch kleinste Indexpaar gewählt.
Die Laufzeit des Algorithmus beträgt $O(NM)$, wie sich am Aufbau der gesamten Matrix leicht herleiten lässt. Wird nur die Distanz benötigt, kann die Matrix nur zeilen- oder spaltenweise berechnet werden und hätte so einen Speicherbedarf von $O(N)$ bzw $O(M)$. Für die Berechnung eines optimalen Warping-Pfades wird jedoch die vollständige Matrix benötigt und der Speicherbedarf wächst auf $O(NM)$. Die Laufzeit ist in beiden Fällen gleich.
Es gilt zu beachten, das der optimale Warping-Pfad nicht unbedingt eindeutig ist. Weiterhin ist $DTW(X,Y)$ zwar ein Distanzmass, jedoch keine Metrik, da die Dreiecksungleichung hierfür im Allgemeinen nicht gültig ist. Dies mag man sich an folgendem Beispiel (zitiert aus [15]) verdeutlichen:
Seien ein Featurespace $\mathcal{F} = \{\alpha, \beta, \gamma\}$ und eine Metrik $c : \mathcal{F} \times \mathcal{F} \to \{0,1\}$ definiert als:

$$c(x,y) = \begin{cases} 0, \text{wenn } x = y \\ 1, \text{wenn } x \neq y \end{cases}$$

Des weiteren seien $X = (\alpha, \beta, \gamma)$, $Y = (\alpha, \beta, \beta, \gamma)$ und $Z = (\alpha, \gamma, \gamma)$ drei Sequenzen über $\mathcal{F}$. Die DTW-Distanzen dieser Sequenzen belaufen sich demnach wie folgt: $DTW(X, Y) = 0$, $DTW(X, Z) = 1$ und $DTW(Y, Z) = 2$. Somit ist $DTW(Y, Z) > DTW(X, Y) + DTW(X, Z)$ und die Dreiecksungleichung nicht erfüllt.
Des weiteren zeigt sich hier, das der optimale Warping-Pfad nicht eindeutig ist. So sind $p^1 = ((1,1),(2,2),(3,2),(4,3))$, $p^2 = ((1,1),(2,2),(3,2),(4,3))$ und $p^3 = ((1,1),(2,1),(3,3),(4,3))$ gleichermassen optimale Warping-Pfade zwischen Y und Z, deren Kosten sich auf 2 belaufen.

2.3.2.1 Indizierung mit DTW Obwohl DTW nicht der Dreiecksungleichung gehorcht und somit keine Metrik ist, wurden Verfahren entwickelt, um Zeitreihen anhand der DTW-Distanz zu indizieren und so Information Retrieval auf grossen Datenmengen zu ermöglichen bzw. zu beschleunigen. Hier sei Insbesondere die Arbeiten von Eamonn Keogh et al. ([10, 11]) genannt.
Da die Aufgabe des zu entwickelnden Werkzeugs jedoch eher eine (feinteilige) Suche ähnlicher Subsequenzen auf einer relativ kleinen Datenmenge ist denn ein großflächiges (und gröberes) Retrieval, sei an dieser Stelle nur auf die Literatur verwiesen.

2.4 Subsequenz-Suche

Im Folgenden sei $X = (x_1, ..., x_N)$ eine Query-Sequenz und $Y = (y_1, ..., y_M)$ eine Datenbank-Sequenz, d.h. es gilt: $N, M \in \mathbb{N}$ und $x_n, y_m \in \mathcal{F}$ für alle $n \in [0 : N]$, $m \in [0 : M]$. Zusätzlich sei $M \ll N$.
Ziel ist es nun, eine Subsequenz (eigentlich Substring)

$$Y(a^* : b^*) := (y_{a^*}, y_{a^*+1}, ..., y_{b^*})$$

mit der minimalen Distanz (z.B. DTW-Distanz) aller Substrings aus Y zu finden (beziehungsweise alle Substrings aus Y, deren Distanzen unter einem bestimmten Schwellwert liegen, insbesondere falls man auch mehrfach auftretende Substrings innerhalb der Datenbank-Sequenz finden möchte).

2.4.1 Subsequence Dynamic Time Warping

In seiner ursprünglichen Form ist DTW nicht für die Suche nach ähnlichen Subsequenzen geeignet. Beim Berechnen der vollständigen akkumulierten Kos-

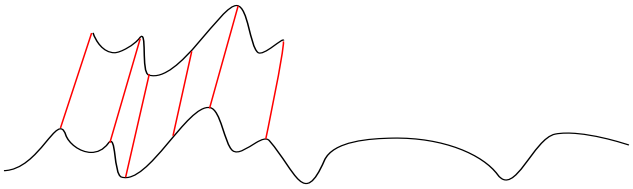


Abbildung 11: Subsequence Dynamic Time Warping

tenmatrix würden die nicht-passenden Regionen, die einen der Query ähnlichen Substring innerhalb der Datenbanksequenz umgeben, die DTW-Distanz stark erhöhen. Die Grenzbedingung des Warping-Pfades, d.h. die Forderung, der Warping-Pfad müsse die Sequenzen vollständig abdecken, steht hier ebenso im Wege. Der Algorithmus lässt sich jedoch mit geringfügigen Modifikationen entsprechend anpassen.
Die im Folgenden beschriebene Variante des Offline Subsequence Dynamic Time Warping ist der Beschreibung in [15] entlehnt und setzt die Kenntnis der kompletten Datenbanksequenz voraus. Die Online-Variante (beschrieben von Sakurai, Faloutsos und Yamamoro in [18]), die darauf ausgelegt ist, die Datenbanksequenz in Form eines Datenstroms zu bearbeiten, wurde in der Notation angeglichen.

2.4.1.1 Offline

Für die Offline-Subsequenz-Suche muss lediglich die Akkumulierte Kostenmatrix geringfügig anders initialisiert werden:

Definition 6 (Akkumulierte Kostenmatrix für Subsequenz-Suche)**.** *Akkumulierte Kostenmatrix* $D \in \mathbb{R}^{M \times N}$

- $D(n,1) = \sum_{k=1}^{n} c(x_k, y_1)$ *für* $n \in [1:N]$
- $D(1,m) = c(x_1, y_m)$ *für* $m \in [1:M]$
- $D(n,m) = min\{D(n-1,m-1), D(n-1,m), D(n,m-1)\} + c(x_n, y_m)$

Alternativ lässt sich dies auch über eine erweiterte Akkumulierte Kostenmatrix formulieren:

Definition 7 (Erweiterte Akkumulierte Kostenmatrix für Subsequenz-Suche)**.** *Akkumulierte Kostenmatrix* $D \in \mathbb{R}^{M \times N}$

- $D(n,0) = \infty$ *für* $n \in [0:N]$
- $D(0,m) = 0$ *für* $m \in [0:M]$

- $D(n,m) = min\{D(n-1,m-1), D(n-1,m), D(n,m-1)\} + c(x_n, y_m)$ *für* $n \in [0:N]$, $m \in [0:M]$

Auf diese Weise führen die nicht zusammenpassenden Sequenzen vor und hinter der (passenden) Query-Sequenz nicht zu einer Erhöhung der DTW-Distanz zwischen Query und eventueller Subsequenz.
Um den der Query entsprechenden Substring nun aufzufinden, wird wiederum Backtracking angewandt. Der Startpunkt für das Backtracking ist nun $b^* := argmin(N,b)$, $b \in [1:M]$, d.h. das Element der obersten Zeile mit dem niedrigsten Distanzwert, und entspricht gleichzeitig dem End-Index der Subsequenz $Y(a^* : b^*)$.
Der Warping-Pfad $p^* = (p_1, ...p_L)$ beginnt also bei $p_L = (N, b^*)$. Das Backtracking folgt nun derselben Rekursion wie beim allgemeinen DTW, mit Ausnahme der Endbedingung: Sobald Index $n = 1$ ist, d.h. $p_1 = (1, a^*)$, wird das Backtracking beendet. So fliessen die unpassenden Bereiche vor und nach der gefundenen Subsequenz nicht mit in Distanzberechnung ein.
Offensichtlich fällt die Grenzbedingung für einen Subsequenz-Warping-Pfad weg, Schrittgrösse und Monotonie bleiben jedoch erhalten. Möchte man alle Subsequenzen, deren DTW-Distanz einen bestimmten Schwellwert τ unterschreitet, auffinden, kann dies wie folgt geschehen:

1. Erstelle Liste der Menge der möglichen Startpunkte

$$P = \{(N,b) | D(N,b) < \tau\}$$

2. Bereinige die Liste, um doppelte Suchergebnisse zu vermeiden
3. Berechne Subsequenz-Warping-Pfade (wie oben beschrieben) durch Backtracking von den Startpunkten der bereinigten Liste.

Der zweite Schritt kann durch ein heuristisches Verfahren gelöst werden. Beispielsweise kann überprüft werden, ob die Punkte in einer bestimmten (direkten) Umgebung zu einem möglichen Startpunkt $D(N,b)$ ebenfalls in P enthalten sind. Trifft dies zu, so können diese entfernt werden. Hierbei wäre es ratsam, die Liste zunächst aufsteigend bzgl. der Distanz zu sortieren, so dass die möglichen Startpunkte mit der niedrigsten Distanz gewählt werden.

2.4.1.2 Online Ist die Datenbanksequenz nicht vollständig bekannt, z.B. da sie als Datenstrom vorliegt, wäre die Anwendung des im vorigen Kapitel beschriebenen Subsequence-DTW-Algorithmus eventuell problematisch. Zwar könnte man die akkumulierte Kostenmatrix für die Subsequenz-Suche Stück für

Stück mit jedem einfließenden Datenstrom-Element aufbauen und ab einem bestimmten Schwellwert das Backtracking durchführen. Allerdings könnte dies, wenn durchweg die komplette Matrix gespeichert wird, zu einem erheblichen Speicheraufwand (allgemein: $O(NM)$) führen, bzw. lässt sich dieser schlecht einschätzen, wenn die Länge des Datenstroms unbekannt und nicht beschränkt ist. Zudem ist unsicher, ob sich die Distanz, wenn sie sich einmal unterhalb des Schwellwertes befindet, mit einem weiteren eingehenden Datenstrom-Element nicht noch verringert und somit eventuell unvollständig zurückgegeben wird. Das mit SPRING betitelte Verfahren ([18]) begegnet diesen Problemen.
Anstatt die komplette Matrix vorzuhalten, werden nur die jeweils letzte Spalte und die aktuelle Spalte der Matrix berechnet. Da ein Backtracking dadurch unmöglich wird, werden Start- und Endindex (t_s und t_e) einer möglichen Subsequenz sowie die aktuell gefundene niedrigste Distanz (d_{min}) isoliert gespeichert und ständig aktualisiert.
Zudem wird der Startindex für jede mögliche Subsequenz vorwärts berechnet. Die Initialisierung erfolgt prinzipiell wie bei der erweiterten akkumulierten Kostenmatrix zur Subsequenz-Suche. Auch die Berechnung der Distanzwerte folgt derselben Rekursion. Zusätzlich zum Distanzwert wird der Startindex anhand der Rekursion vorwärts-berechnet:

$$S(n,m) = \begin{cases} S(n,m-1), & \text{wenn } D(n,m-1) \text{ min.} \\ S(n-1,m), & \text{wenn } D(n-1,m) \text{ min.} \\ S(n-1,m-1) & \text{wenn } D(n-1,m-1) \text{ min.} \end{cases}$$

Die Aktualisierung von d_{min}, t_s und t_e erfolgt, wenn am Index t $D(N,t)$ sowohl kleiner als der Schwellwert τ als auch das aktuelle d_{min} unterschreitet. Ist dies der Fall, wird $d_{min} = D(N,t)$, $t_s = S(N,t)$ und $t_e = t$ gesetzt. Grundvoraussetzung für das Auffinden einer Subsequenz ist ist auch hier, das d_{min} den Schwellwert τ unterschreitet. Um Doubletten zu vermeiden, wird eine Subsequenz jedoch erst dann als gefunden gemeldet, wenn der Wert zum Index t in $D(N,t) > d_{min}$ ist, d.h. ein Minimum überwunden hat, oder der aktuelle Startwert $S(N,t) > t_e$ ist, d.h. eine neue mögliche Subsequenz beginnt. Wird eine Subsequenz gefunden, wird d_{min} zurückgesetzt (d.h. $d_{min} = \infty$). Zudem werden, um Überlappungen zu vermeiden, die Werte $D(n,t) = \infty$ gesetzt, falls $S(n,t) \leq t_e$ ist. Somit wird die Berechnung dort quasi „abgeschnitten".

2.5 Audio-Synchronisierung

Verwandt mit der Subsequenz-Suche auf Audiodaten ist auch das synchrone Abspielen von Musikstücken. Mit MATCH (Dixon, [5]) und dem SyncPlayer

(Müller, [15]) sind bereits zwei Programme bzw. Frameworks zum synchronen Abspielen von Musikstücken bekannt, die beide auf Dynamic Time Warping und den oben genannten Merkmalsextraktionsverfahren basieren. In [15] findet sich des Weiteren eine reichhaltige Übersicht über die Synchronisierungs-Thematik. Ein Ansatz zur DTW-basierten Synchronisierung von Audio- und MIDI-Daten findet sich in [9].

3 Experimentelle Evaluierung

Die oben beschriebenen Verfahren zur Merkmalsextraktion und zur Subsequenz-Suche wurden hinsichtlich ihrer Güte evaluiert, um das für die Anwendung am besten geeignete Verfahren zu finden.
Hierzu wurden die oben beschriebenen Merkmalsextraktions- und Suchverfahren in unterschiedlichen Konfigurationen getestet. Verfahren, die von vornherein nicht vielversprechend waren (z.B. MFCCs), wurden aussen vor gelassen.

3.1 Getestete Verfahren

Da es sich bei dem zu entwickelnden Werkzeug letztlich um eine Subsequenzsuche auf Audiodaten handelt, wurde die Online- und Offline Variante des Subsequence Dynamic Time Warping in Kombination mit verschiedenen Merkmalsextraktionsverfahren getestet. Die Merkmalsextraktionsverfahren wurden auf FFT-Basis, d.h. so Implementiert, das sie sowohl für Online- als auch für Offline-Anwendungen geeignet sind. Als lokale Kostenfunktion dient die Euklidische Distanz, als Fensterfunktion wurde aufgrund der vergleichsweise schmalbandigen Impulsantwort (bzw. guten Fokussierung) das Hamming-Fenster gewählt. Da die Verfahren auf FFT-Basis implementiert wurden, wurde bei allen Verfahren die Idee übernommen, den Bassbereich bis zu der Frequenz, ab der sich die FFT-Komponenten auf Halbtöne abbilden lassen, linear abzubilden. Hier wurden zwei Varianten getestet: bei der ersten Variante wurde die untere Grenzfrequenz unabhängig von der Blockgrösse bei 300Hz angesetzt (vgl. MFCCs). Bei der zweiten Variante wurde die untere Grenzfrequenz in Abhängigkeit von der Blockgrösse gesetzt, um so lediglich die Abbildbarkeit auf Halbtöne zu gewährleisten.

- Heuristischer Ansatz (Lokale Amplitude)
- Heuristischer Ansatz (Lokale Energie)
- Heuristischer Ansatz, Anschlag (Lokale Amplitude)

- Heuristischer Ansatz, Anschlag (Lokale Energie)
- Kombiniertes Chroma
- Kombiniertes Chroma, Anschlag
- Kombiniertes Chroma, quantisiert
- CNES (modifiziert)

Die Verfahren wurden mit den Blockgrössen 2048, 4096, 8192 und 16346 getestet. Bei Blockgrössen $\geq$ 16384 (deren Länge über einer halben Sekunde läge) wäre die Präzision bei kurzen Queries (1-2 Sekunden) a priori zu gering. Weiterhin wurde bei den ersten vier Verfahren getestet, inwiefern das verwerfen von bestimmten FFT-Komponenten zu besseren Ergebnissen führt. Das CNES-Verfahren wurde, in Abweichung von der Originalarbeit, auf Basis des kombinierten Chromas (d.h. auch auf FFT-Basis, nicht auf Basis einer filterbasierten Zerlegung) implementiert. So wurde auf einfache Weise die Online-Fähigkeit hergestellt. Zusätzlich wurden hier unterschiedliche Fenstergrössen für das Gewichtungsfenster (5,7,11) und verschiedene Downsampling-Raten (3x, 5x) getestet. Die Konfiguration der Originalarbeit (Fenstergrösse 10, 10x downgesampled) wurde aufgrund der sehr geringen Feature-Frequenz (1 Feature-Vektor pro Sekunde) nicht getestest, da dies bei Query-Längen von unter 2 Sekunden a priori wenig Sinn macht. Da das CNES-Verfahren im Originaltext mit einer Blockgrösse von 200ms bei einem halben Block Überlappung angegeben ist, wurde diese Konfiguration auch in Betracht gezogen.

3.2 Testdaten

Als Testdaten liegen drei Interpretationen des Molto Moderato aus Schuberts Klaviersonate in B-Dur (D960) vor. Die Interpretationen unterscheiden sich nicht nur bezüglich der Phrasierung und Dynamik sowie der Aufnahmequalität (Hall, Mikrofonierung, Rauschen etc.) bisweilen erheblich voneinander. Einige Interpreten (z.B. Brendel) neigen dazu, die (leicht variierte) Wiederholung der Exposition (d.h. des ersten Abschnittes des Stücks) auszulassen. Somit unterscheiden sich die Aufnahmen nicht nur klanglich und spieldynamisch, sondern auch strukturell.
An einigen Stellen sind Störungen zu hören, die aus dem Digitalisierungsprozess stammen können oder von einem unpräzisen Plattenspieler, der dem Klang nach eventuell als Quelle gedient haben könnte. Insofern sind die vorliegenden Daten ein anspruchsvoller Test für die gefundenen Methoden.

3.2.1 Datenbanksequenzen

Als Datenbanksequenzen dienen die vollständigen Interpretationen. Stille am Anfang und am Ende wurden entfernt. Zudem wurden die Interpretationen normalisiert, d.h. anhand der Maximalpegel angeglichen.

3.2.2 Queries

Bestimmte Stellen wurden manuell, also nach Optik und Gehör, mit Hilfe eines Audiobearbeitungsprogramm aus jeder (normalisierten) Interpretation herausgeschnitten. Diese Stellen dienen als Queries für den Matching-Algorithmus. Für das Qualitätskriterium (siehe *Qualitätskriterium*) werden genaue Start- und Endposition (in Samples) der Query notiert. Bei Queries, die sich innerhalb des Stückes wiederholen, werden auch die Start- und Endpunkte der Wiederholungen gespeichert, da diese ebenfalls gefunden werden sollten. Um die Verlässlichkeit bezüglich Wiederholungen zu Testen, wurde darauf geachtet, Stellen mit Wiederholungen als Queries zu wählen.
Die ausgewählten Queries lassen sich in 3 Kategorien einteilen:

- längere musikalisch sinnvolle Abschnitte (l. mus.)
- kurze willkürliche Abschnitte (k. ran.)
- längere willkürliche Abschnitte (l. ran.)

„Musikalisch Sinnvoll" bedeutet in diesem Fall, das die Abschnitte einen formal sinnvollen Abschnitt umfassen (z.B ein vollständiges Thema oder eine isolierte Phrase, zumindest aber vollständige Töne). Da die sich aus der Annotierung ergebenden Abschnitte jedoch keinerlei Bezug zum formalen Aufbau des Stückes haben müssen (ein Benutzer kann jederzeit eine Annotation klicken), werden auch willkürliche Abschnitte verschiedener Länge in Betracht gezogen. Willkürlich heisst in hier, das die Abschnitte auch mitten in einem Ton anfangen können oder viel Stille enthalten. Längere Abschnitte dauern ungefähr 5 bis 15 Sekunden, kurze Abschnitte 1 bis 3 Sekunden. Für jede Kategorie wurden drei Queries ausgewählt. Weiterhin wurde bei der Auswahl der Queries auf größtmögliche dynamische Vielfalt geachtet. Da die Wiederholung einer willkürlichen Sequenz schwer nach Gehör einzugrenzen ist, wird davon ausgegangen, das diese jeweils nur einmal vorkommen.
Tabelle 1 gibt eine Übersicht über die Queries und deren Vorkommen innerhalb der Interpretationen. Die Zeitangaben beziehen sich auf die Datenbanksequenzen, bei denen die Stille am Anfang entfernt wurde. Die Zeitangaben

Tabelle 1: Query-Tabelle

	Interpretation 1			Interpretation 2			Interpretation 3		
l. mus.	#	Start	Ende	#	Start	Ende	#	Start	Ende
Query 1	1	00:00:00	00:10:09	1	00:00:00	00:14:02	1	00:00:00	00:18:26
	2	00:26:10	00:35:26	2	00:40:11	00:58:19	2	00:38:24	00:52:29
	3	08:41:18	08:51:13	3	05:36:07	05:53:00	3	06:40:02	06:53:08
	4	09:05:21	09:15:07	4	16:12:11	06:29:24	4	07:14:24	07:28:16
	-	-	-	5	15:31:17	15:49:25	5	17:03:15	17:16:14
	-	-	-	6	16:09:28	16:26:19	6	17:37:05	17:50:16
Query 2	1	04:12:00	04:16:16	1	04:33:00	04:38:21	1	05:23:04	05:30:19
	2	12:58:18	13:03:27	2	09:59:12	10:05:26	2	11:52:16	12:00:02
	-	-	-	3	20:09:16	20:14:11	3	22:24:11	22:31:16
Query 3	1	04:05:13	04:16:18	1	04:27:00	04:32:23	1	05:16:18	05:22:25
	2	12:52:12	13:03:17	2	09:53:01	09:59:12	2	11:46:12	11:52:16
	-	-	-	3	20:02:06	20:08:16	3	22:18:07	22:24:11
k. ran.	#	Start	Ende	#	Start	Ende	#	Start	Ende
Query 1	-	01:38:03	01:39:00	-	02:11:09	02:12:05	-	02:11:22	02:12:22
Query 2	-	04:26:17	04:30:06	-	04:53:16	04:56:07	-	05:47:15	05:51:11
Query 3	-	06:50:23	06:53:27	-	13:07:08	13:11:14	-	14:48:28	14:52:17
l. ran.	#	Start	Ende	#	Start	Ende	#	Start	Ende
Query 1	-	10:41:29	10:47:25	-	07:55:06	08:00:16	-	09:07:06	09:14:27
Query 2	-	12:20:21	12:27:22	-	19:35:22	19:40:21	-	21:45:04	21:52:15
Query 3	-	01:19:11	01:28:21	-	01:52:04	02:00:11	-	01:48:26	01:59:08

sind der Übersichtlichkeit halber nicht in Samples sondern im Format *Minute:Sekunde:Hundertstel* angegeben.

3.3 Testmethode

Die Suchalgorithmen wurden mit einer diagonalen Suche getestet. Sämtliche Queries (sprich alle Interpretationen aller Queries) wurden jeweils auf allen Sequenzen gesucht. Da die Qualität des Dynamic Time Warping massiv vom Schwellwert abhängt, wurde zunächst ein Durchlauf zur Kalibrierung der Schwellwerte durchgeführt. Anschließend wurden sämtliche Queries in sämtlichen Datenbanksequenzen mit jedem der oben angegebenen Verfahren gesucht.

3.3.1 Kalibrierung des Schwellwertes

Da die Verfahren mit unterschiedlichen Vektorgrössen und -inhalten arbeiten, ist der Schwellwert von Verfahren zu Verfahren unterschiedlich. Weiterhin kann

mit steigender Länge der Queries die Distanz insgesamt grösser ausfallen.
Je nach Merkmalsextraktionsverfahren und Blockgrösse haben die Ergebnisvektoren unterschiedliche Dimensionen, was dazu führt, das sich auch hier andere Distanzwertbereiche ergeben. Um somit eine Vergleichbarkeit zu Schaffen (bzw. die Präzision nicht unnötig zu verschlechtern), muss also zunächst eine Kalibrierung des Schwellwertes stattfinden. Zur Kalibrierung der Schwellwerte wurde zunächst ein diagonaler Durchlauf über die Queries der Kategorie „lang, musikalisch" mit einem einheitlichen, sehr hohen Schwellwert (500000) durchgeführt. Die höchste Distanz, mit der eine Query gefunden wurde, wurde im eigentlichen Test als Schwellwert verwendet. Da der Online-Algorithmus weniger sensibel bzgl. des Schwellwertes ist, wurde der Kalibrierungsdurchlauf nur mit dem Online-Algorithmus durchgeführt. Das Ergebnis der Kalibrierung wurde für die Offline-Variante mit den selben Parametern bezüglich der Merkmalsextraktion übernommen. Weil die Distanzwerte hauptsächlich von Query-Länge und Merkmalsextraktion (die die Dimensionalität des Merkmalsvektors bestimmt) abhängen, wurden die Kalibrierungsergebnisse für die Offline-Variante übernommen.

3.3.2 Qualitätskriterium

Um die Verfahren bezüglich ihrer Qualität einschätzen zu können, wurden (trotz der relativ kleinen Datenmenge) zunächst Precision:

$$\frac{\#\{\{gefundene\ Sequenzen\} \cap \{korrekt\ gefundene\ Sequenzen\}\}}{\#\{gefundene\ Sequenzen\}}$$

d.h. die Anzahl der korrekt gefundenen Sequenzen im Verhältnis zur Menge aller gefundenen Sequenzen, Recall:

$$\frac{\#\{\{gefundene\ Sequenzen\} \cap \{korrekt\ gefundene\ Sequenzen\}\}}{\#\{Queries\}}$$

d.h. die Anzahl der korrekt gefundenen Sequenzen im Verhältnis zur Menge aller Queries sowie der F-Score:

$$2 * \frac{Precision * Recall}{Precision + Recall}$$

d.h. das Verhältnis von Precision und Recall anhand der diagonalen Suche ermittelt. Damit ein Verfahren praktisch einsetzbar ist, sollte sich insbesondere die Menge der falschen Positiven in Grenzen halten. Sollte der Aufwand für manuelle Nacharbeiten (sprich, aussortieren von *false positives*) zu hoch sein, ist

ein Verfahren nicht praxistauglich. Für die Tests wird eine Query als gefunden gemeldet, wenn Start- und Endpunkt nicht mehr als eine Sekunde vom manuell ermittelten Start- und Endpunkt entfernt sind. Wird eine Query innerhalb einer Datenbanksequenz aufgefunden, werden die vom Programm bestimmten Start- und Endpunkte der gefundenen Sequenz mit den zuvor notierten Start- und Endpunkten der Query (innerhalb der aktuell bearbeiteten Interpretation, inkl. Wiederholungen) verglichen.
Ein weiteres wichtiges Mass für die Qualität gilt hier die (zeitliche) Abweichung des Start- und Endpunkts des vom Matching-Algorithmus gefundenen Abschnitts zum (manuell bestimmten) Start- und Endpunkt der Query bzw. einer möglichen Wiederholung, d.h. den erwarteten Werten. Als „Zeitmass" dient hier der Abstand in Samples. Um den Vergleich der Verfahren zu vereinfachen, wird das arithmetische Mittel der Start- und Endpunkt-Abweichungen als Vergleichswert herangezogen. Dieser sollte möglichst nicht über einer Sekunde liegen, da sonst die weitere Auswertung im Rahmen der Interpretationsforschung verfälscht werden könnte.

3.4 Testumgebung

Zum Testen wurde eine einfache Standalone-Umgebung auf Java-Basis implementiert. Zum Einlesen der Audiodateien dient die in die Java-Technologie integrierte Java-Sound-API, für die Fouriertransformation wurde die Jtransforms-Bibliothek verwendet ([22]).

3.5 Ergebnisse

3.5.1 Schwellwertproblematik

Während der Tests stellte sich eine Problematik bezüglich des Schwellwertes heraus, anhand dessen eine Subsequenz als gefunden gemeldet wird. So ist es durchaus möglich, sowohl den Offline- wie auch den Online-DTW-Algorithmus derart zu kalibrieren, das eine bestimmte Query in der Datenbanksequenz, aus der sie ursprünglich stammt, mit allen Wiederholungen und ohne Falschmeldungen gefunden wird.
Jedoch sind die Ergebnisse mit diesem Schwellwert schon für eine andere Query aus der gleichen Datei häufig kaum zu gebrauchen.
So ergeben sich mit den folgenden Parametern

Verfahren	Merkmal	Blockgrösse	Overlap	Schwellwert
OnlineDTW	Kombi-Chroma (quantisiert)	4096	2	2084

folgende Ergebnisse (Qu1-In1 = Query 1 aus Interpretation 1 etc.):

Query	Db-Sequenz	Gefunden	Precision	Recall	F-Score
Qu1-In1	Interpretation 1	4	1	1	1
Qu2-In1	Interpretation 1	809	0	0	0
Qu1-In2	Interpretation 2	0	0	0	0
Zusammen	Zusammen	813	0,0049	1	0,0098

Die Schwellwerte so zu kalibrieren, das sie auf unterschiedlichen Datenbanksequenzen und unterschiedlichen Queries gleichermaßen funktionieren, ist mit den gegebenen Verfahren also anscheinend nicht möglich. Insbesondere der Offline-DTW-Algorithmus stellte sich als besonders sensibel gegenüber der Schwellwerteinstellung heraus. So kann sowohl ein zu hoher wie auch ein zu niedriger Schwellwert dazu führen, das bisweilen gar keine Ergebnisse geliefert werden.

3.5.2 Testergebnisse

Die Tabellen 3 - 6 präsentieren jeweils die besten 10 Verfahren bezüglich der Kenngrössen F-Measure, Precision, Recall und zeitlicher Abweichung (in Millisekunden).

Tabelle 2: Legende

Abkürzung	Verfahren
PA	Lokale Amplitude (Pitch Amplitude)
PAO	Lokale Amplitude Anschlag (Pitch Amplitude Onset)
PE	Lokale Energie (Pitch Energy)
PEO	Lokale Energie Anschlag(Pitch Energy Onset)
CC	Kombiniertes Chroma (Combined Chroma)
CCO	Kombiniertes Chroma Anschlag (Combined Chroma Onset)
CCQ	Kombiniertes Chroma Quantisiert (Combined Chroma Quantized)
CNES_d^w	Chroma Normalized Energy Statistics (Downsamplingfaktor d, Fenstergrösse w)
(sel.), (adap.)	selektiv, adaptiv

Tabelle 3: F-Measure

DTW	Merkmal	Blockgr.	Ü.-lap.	F-Measure:	Precision:	Recall:	Abw. (ms):
Offline	CCQ	8192	0	0,0304	0,0279	0,033	844
Offline	CC	2048	0	0,0231	0,0204	0,027	914
Offline	$CNES_3^5$	4096	0	0,0204	0,0165	0,027	1192
Offline	$CNES_3^5$	4096	0	0,0204	0,0165	0,027	1192
Offline	$CNES_3^5$	4410	0	0,0158	0,0131	0,020	1199
Offline	$CNES_3^5$	4410	0	0,0158	0,0131	0,020	1199
Offline	CC	8192	0	0,0154	0,0108	0,027	1091
Offline	$CNES_3^5$	8820	2	0,0151	0,0121	0,020	811
Offline	$CNES_3^5$	8820	2	0,0151	0,0121	0,020	811
Offline	$CNES_3^7$	8192	0	0,0149	0,0119	0,020	1189

Tabelle 4: Precision

DTW	Merkmal	Blockgr.	Ü.-lap.	F-Measure:	Precision:	Recall:	Abw. (ms):
Offline	CCQ	8192	0	0,0304	0,0279	0,033	844
Offline	CCQ	2048	0	0,0231	0,0204	0,027	914
Offline	$CNES_3^5$	4096	0	0,0204	0,0165	0,027	1192
Offline	$CNES_3^5$	4096	0	0,0204	0,0165	0,027	1192
Offline	$CNES_3^5$	4410	0	0,0158	0,0131	0,020	1199
Offline	$CNES_3^5$	4410	0	0,0158	0,0131	0,020	1199
Offline	$CNES_3^5$	8820	2	0,0151	0,0121	0,020	811
Offline	$CNES_3^5$	8820	2	0,0151	0,0121	0,020	811
Offline	$CNES_3^7$	8192	0	0,0149	0,0119	0,020	1189
Offline	$CNES_3^7$	8192	0	0,0149	0,0119	0,020	1189

Tabelle 5: Recall

DTW	Merkmal	Blockgr.	Ü.-lap.	F-Measure:	Precision:	Recall:	Abw.(ms):
Online	CCO	2048	2	0,0005	0,0002	0,440	878
Online	$CNES_3^{11}$	4096	0	0,0021	0,0010	0,433	822
Online	$CNES_3^{11}$	4096	0	0,0021	0,0010	0,433	822
Online	$CNES_3^7$	4096	0	0,0021	0,0011	0,420	759
Online	$CNES_3^7$	4096	0	0,0021	0,0011	0,420	759
Online	CC	8192	0	0,0018	0,0009	0,420	748
Online	$CNES_3^{11}$	4410	2	0,0016	0,0008	0,420	620
Online	$CNES_3^{11}$	4410	2	0,0016	0,0008	0,420	620
Online	CC	16384	2	0,0017	0,0008	0,413	737
Online	CC	2048	0	0,0012	0,0006	0,413	573

Tabelle 6: Zeitliche Abweichung

DTW	Merkmal	Blockgr.	Ü.-lap.	F-Measure:	Precision:	Recall:	Abw. (ms):
Offline	CC	2048	2	0,0036	0,0020	0,0200	97
Offline	PE (adap.)	2048	2	0,0028	0,0015	0,0200	105
Offline	PE	2048	2	0,0019	0,0010	0,0133	118
Offline	PE (adap.)	16384	2	0,0010	0,0006	0,0067	234
Offline	PE	16384	2	0,0010	0,0005	0,0067	234
Offline	PE (sel.)	16384	2	0,0005	0,0003	0,0067	234
Offline	PE (sel., adap.)	16384	2	0,0005	0,0003	0,0067	234
Offline	PEO (adap.)	4096	2	0,0009	0,0005	0,0133	290
Offline	PEO (adap.)	4096	2	0,0009	0,0005	0,0133	290
Offline	PE	4096	2	0,0008	0,0004	0,0133	350

3.6 Test-Fazit

Es stellte sich heraus, das weder die Offline- Noch die Online-Variante des Subsequence Dynamic Time Warping in Kombination mit den bekannten Merkmalsextraktionsverfahren für diese Anwendung sinnvoll sind. Es kann nicht einmal sicher gestellt werden, das Subsequenzen innerhalb der gleichen Interpretation zuverlässig aufgefunden werden. Die höchsten erzielten Recall-Werte lagen unter 0.5, während die Menge der "false positives", die bei einer entsprechenden Kalibrierung des Schwellwertes gemeldet werden, die Ergebnisse nahezu unbrauchbar machen. Angesichts der Tatsache, das der Algorithmus auf ähnlicheren Daten (gleiches Instrument, gleicher Klang der Aufnahme) durchaus korrekte Ergebnisse lieferte, mag man dies auf die massiven Unterschiede der Interpretationen zurückführen. Zudem sind die Verfahren nur bedingt für Anwendungen mit einer Vielzahl an sehr unterschiedlichen Queries geeignet, unter anderem, da die Distanzwerte von der Länge abhängen und somit die Kalibrierung der Schwellwerte schlecht möglich ist. Somit ist das Verfahren für eine Anwendung mit beliebig langen Queries kaum zu gebrauchen.
Ansonsten lässt sich bezüglich der Verfahren feststellen, das eine stärkere Aggregierung (oder Abstrahierung) von Vorteil ist. Bei allen Parametern bis auf die zeitliche Abweichung finden sich die stark aggregierenden CNES-Verfahren in der Überzahl, während die ebenfalls stark aggregierenden chromabasierten Verfahren an der Spitze stehen. Des weiteren unterscheiden sich die Offline- und Online-Variante des Subsequence Dynamic Time Warping stark bezüglich ihrer Präzision. Insbesondere beim Recall liegt die Online-Variante weit vorn, allerdings bei wesentlich geringerer Präzision, was sich auch auf die F-Measure-Werte auswirkt, bei denen die Offline-Variante vorn steht (allerdings bei schlechten Recall-Werten).

4 Implementierung

Da für die Subsequenz-Suche letztlich kein adäquates Verfahren gefunden wurde, ist die Implementierung eher exemplarischer Natur. Dennoch sollen im Folgenden die Ideen bezüglich der GUI und des Arbeitsablaufes dargelegt werden.

4.1 Aufbau

Die Anwendung folgt einem modularen Aufbau. Die Matcher-Anwendung wurde (analog zur Testumgebung) in Java umgesetzt und funktioniert auch als Standalone-Programm über Kommandozeilensteuerung. Für die Fouriertransformation wurde die JTransforms-Bibliothek verwendet ([22]). Für die Umsetzung der GUI wird die Programmierumgebung „Pure Data" verwendet. Die Anbindung zwischen GUI und Matcher erfolgt über die PDJ-Bibliothek und die ProcessBuilder-Klasse des Java-Frameworks. Derart modular aufgebaut, lassen sich die einzelnen Komponenten leicht durch eventuelle zukünftige Versionen ersetzen.
Die Einbindung in eine Audioinfrastruktur (Audio-Eingänge für die Live-Interpretation etc.) erfolgt über das Pure Data-Framework, die Ein- und Ausgabe der Audiodateien erfolgt über Ablage in entsprechenden (konfigurierbaren) Ordnern.

4.2 Verwendete Frameworks und Bibliotheken

4.2.1 Pure Data

Pure Data (kurz: PD) ist eine Entwicklungsumgebung für elektronische Musik im speziellen und Audiodatenverarbeitung im Allgemeinen. Das Programmierparadigma von Pure Data lässt sich zwischen Objekt- und Datenstromorientiert ansiedeln. Die Programmierung erfolgt grafisch. Entwickelt wurde und wird es ursprünglich vom Max-MSP-Entwickler Miller Puckette, wobei sich eine sehr aktive Entwicklergemeinde gebildet hat, die sich für unzählige Verbesserungen und Erweiterungen verantwortlich zeichnet. Pure Data seinerseits ist in C und Tcl/Tk programmiert, ist quelloffen und unterliegt einer freien Lizenz (SIBSD). Es ermöglicht eine flexible Anbindung an verschiedenste Audiotreiber und Soundkarten und fügt sich gut in die Audioarchitekturen aller gängigen Betriebssysteme ein.

Pure Data verfügt über eine umfangreiche Bibliothek an Objekten zur Audiobearbeitung sowie zum Erstellen entsprechender Kontrollstrukturen. Zudem kann es in verschiedenen Programmiersprachen (C, Python, Java über PDJ) erweitert werden. Eine umfassende Beschreibung der Möglichkeiten von PD findet sich in [12] oder [16].

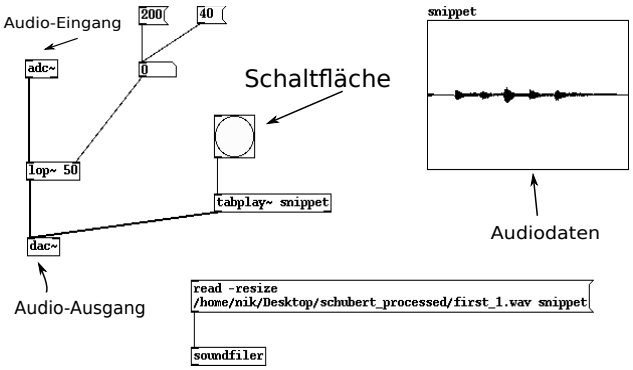


Abbildung 12: Pure Data Beispiel

4.2.1.1 PD GUI Editor Die Grundlage der zu entwickelnden grafischen Benutzerschnittstelle ist der GUI Editor von Pure Data. Mit diesem lassen sich Audioanwendungen in enger Interaktion zwischen DSP und GUI entwickeln. Die Programmierung erfolgt über das Verknüpfen verschiedener Objekte mit Kontroll- und Audiodatenströmen (vgl. Abb. 12). Durch Zusammenfassen verschiedener Objekte zu Abstraktionen lassen sich wiederverwendbare Einheiten schaffen, auf denen das Annotierungswerkzeug aufbaut.

4.2.1.2 PDJ PDJ ist eine Erweiterung für Pure Data. Mit PDJ ist es möglich, das an sich in C geschriebene Pure Data mit in Java implementierten Objekten zu erweitern ([7]).

4.3 Arbeitsablauf

Der Arbeitsablauf beginnt mit der Konfiguration des gewünschten Zeitraumes für die Annotierung (d.h., dem Zeitraum, der um die annotierte Stelle herum

aus dem Livestrom extrahiert wird) sowie der Konfiguration von des Ausgabe-Ordners für sämtliche extrahierte Stellen (sowohl aus der Live-Interpretation wie auch aus den gespeicherten Interpretationen) und des Datenbankordners, der die anderen Interpretationen des Stücks enthält. Die Konfiguration des Zeitraumes ist notwendig, da die die Annotierung mit einem einzelnen Klick erfolgt. Das Markieren von Anfang und Ende eines Bereiches würde die kognitive Anforderung deutlich erhöhen. Beispielsweise könnten parallel Abschnitte entstehen, oder das Beenden eines Abschnitts könnte übersehen werden. Der Zeitraum für den zu extrahierenden Abschnitt setzt sich aus Vorlauf, d.h. der Bereich vor dem markierten Zeitpunkt, und Nachlauf, d.h. dem Bereich nach dem markierten Zeitpunkt zusammen. Die annotierten Stellen werden als Audiodatei im Ausgabe-Ordner gespeichert und in allen Dateien gesucht, welche sich im Datenbankordner befinden.
Sind Interpret und Annotator bereit, startet der Annotator den Annorierungsvorgang durch Klick auf eine Start-Taste. Ein Statusfeld zeigt den aktuellen Arbeitsschritt an.
Der Vorgang des Annotierens findet über Tastfelder statt. Um eine Stelle zu markieren, tippt die annotierende Person das entsprechende Feld.
Ist der Annotierungsvorgang beendet, klickt die annotierende Person auf eine Stop-Taste, was die Annotierung abschliesst und den Suchvorgang in den anderen Interpretationen auslöst. Ist der Suchvorgang beendet, wird dies über das Statusfeld mitgeteilt. Die Zuordnung von Interpretation und Annotation der extrahierten Abschnitte erfolgt jeweils über die Dateinamen nach dem Schema *Ursprungsdateiname-Annotation-Nr.*

4.4 Benutzerschnittstelle

4.4.1 Elemente

4.4.1.1 Info-Feld Hier können die Informationen über das Stück sowie den aktuellen Interpreten eingegeben werden.

4.4.1.2 Settings-Feld Hier können Vor- und Nachlauf (d.h. der Zeitraum, der aus dem Audiostrom um den annotierten Zeitpunkt herum extrahiert wird) eingestellt werden. Eine aus dem Livestrom extrahierte Query wird in allen Dateien gesucht, die sich im Datenbank-Ordner befinden. Ausserdem kann der Ausgabeordner, in dem die extrahierten Sequenzen gespeichert werden, konfiguriert werden.

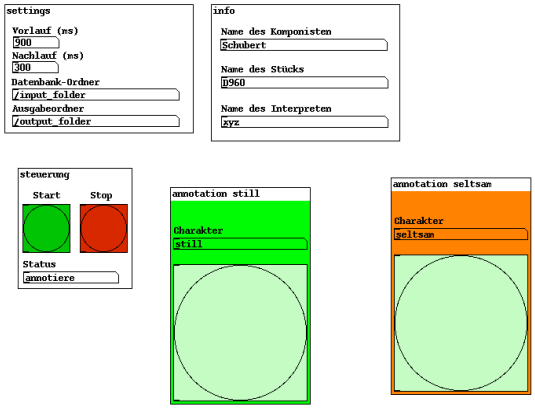


Abbildung 13: Beispiel GUI

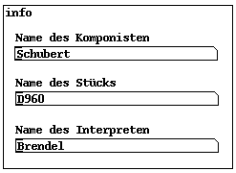


Abbildung 14: Info-Feld

4.4.1.3 Steuerungsfeld Durch Klick auf „Start" wird der Annotierungsvorgang gestartet und die notwendigen Parameter initialisiert. Durch Klick auf „Stop" wird der Annotierungsprozess beendet und der Matching-Prozess, der die annotierten Stellen in den anderen Interpretationen aufsucht, wird gestartet. Das Status-Feld gibt Rückmeldung über den Arbeitsschritt, in dem sich der Benutzer gerade befindet.

4.4.1.4 Annotation Die Annotation ist das eigentliche Kernelement des Programms. Durch Klicken oder Tasten der Annotation wird der entsprechende Abschnitt aus dem Audiostrom extrahiert. Von dieser Schaltfläche können beliebig viele hinzugefügt werden. Zur einfachen optischen Unterscheidung können die Schaltflächen mit unterschiedlichen Farben versehen werden.

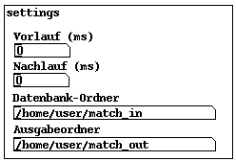


Abbildung 15: Settings-Feld

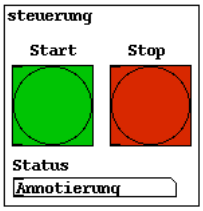


Abbildung 16: Steuerungs-Feld

5 Zusammenfassung und Ausblick

Mit dem Ziel, interpretationsübergreifende Annotierung von Musikdaten zu ermöglichen, wurden im ersten Kapitel zunächst Anforderungen an ein Werkzeug, welches dies ermöglicht, dargelegt. Auf den Grundlagen zur Merkmalsextraktion aus Audiodaten und des Subsequence Dynamic Time Warping, die im zweiten Kapitel nebst allgemeinen Grundlagen der digitalen Audiodatenverarbeitung dargestellt wurden, wurde ein entsprechendes Suchverfahren implementiert.
Um das Verfahren zu Testen und eine gute Parameterkonfiguration empirisch

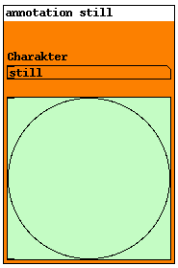


Abbildung 17: Annotations-Schaltfläche

zu ermitteln, wurde zunächst ein Datensatz auf Basis dreier (sehr unterschiedlicher) Interpretationen des Molto Moderato aus Schuberts Klaviersonate in B-Dur (D960) erstellt. Die Auswahl der Abschnitte erfolgte mit den Ziel, ein möglichst breites Spektrum an Phrasierung und Dynamik abzudecken. Auf Basis dieses Datensatzes wurden die verschiedenen Kombinationen aus Merkmalsextraktion und DTW-Varianten anhand einer diagonalen Suche (d.h. alle Abschnitte wurde auf allen Interpretationen gesucht) evaluiert.
Leider stellte sich heraus, das die anvisierte Suchmethode auf Dynamic Time Warping Basis nicht die gewünschten Ergebnisse lieferte. Zu gross sind die Unterschiede in Spieldynamik und Tempo der unterschiedlichen Interpretationen, als das die Kombination aus niederschwelliger Merkmalsextratkion und Subsequence-DTW dem gerecht wurden. Selbst stark abstrahierende Verfahren wie CNES brachten hier kaum Vorteile. Insbesondere liess sich der Schwellwert für die Ähnlichkeit einer Subsequenz nicht so kalibrieren, das die Menge an *false positives* im praktikablen Rahmen blieb.
Dennoch wurden auf Basis von Pure Data eine grafische Benutzerschnittstelle und ein Arbeitsablauf für den Annotierungsprozess entwickelt und im vierten Kapitel dokumentiert.

5.1 Ausblick

Da sich Subsequence Dynamic Time Warping nun nicht als erfolgreich herausstellte, bleibt die Frage nach Alternativen. Nachsortieren der Ergebnisse ist aufgrund der schlechten Recall-Werte (nicht einmal die Hälfte der Queries wurden gefunden) wenig vielversprechend. Da Synchronisierung (d.h. das synchrone Abspielen) mit Dynamic Time Warping (auch online, vgl [4]) bereits erfolgreich gelöst wurde, könnte man die verschiedenen Interpretationen zunächst mit der live gespielten Interpretation synchronisieren und die Subsequenzen anhand der Indizes extrahieren. Gerade bezogen auf die Schubert-Interpretationen, die dieser Arbeit als Testdaten zugrunde lagen, stellt sich jedoch das Problem, das die Interpreten mit der Anzahl der Wiederholungen einen freien Umgang pflegen (z.B. Brendel). An diesem Punkt scheitert eine rein DTW-basierte Synchronisierung (so getestet mit dem MATCH-Player, vgl. [5]). Somit wäre für diesen Ansatz eine wesentlich aufwändigere Partitur-Synchronisierung vonnöten, die Sprungmarken im Notentext differenziert je nach Interpretation erkennt und behandelt. Für eine langfristige Weiterentwicklung wäre neben einem funktionierenden Suchverfahren auch eine optische Partitursynchronisierung auf Tcl/Tk-Basis, die sich in Pure Data integrieren lässt, wünschenswert.

Literatur

[1] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms*, volume 730. Springer Berlin / Heidelberg, 1993.

[2] Mark A. Bartsch and Gregory H. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proceedings of the 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.

[3] David Butler. *Musician's Guide to Perception and Cognition.* Schirmer, New York, 1992.

[4] Simon Dixon. Live tracking of musical performances using on-line time warping. In *Proc. of the 8th Int. Conference on Digital Audio Effects (DAFx'05)*, 2005.

[5] Simon Dixon and Gerhard Widmer. Match: A music alignment toolchest. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, London, GB, 2005.

[6] P. Duhamel and M. Vetterli. Fast fourier transforms: A tutorial review and a state of the art. *Signal Processing*, 19, 1990.

[7] Pascal Gauthier. *PDJ: Java Interface for Pure Data.* http://www.leson666.com/software/pdj/.

[8] Frederic J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. In *Proceedings of the IEEE*, volume 66, 1978.

[9] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Proceedings of the 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[10] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7, 2005.

[11] Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 285–289, New York, NY, USA, 2000. ACM.

[12] Johannes Kreidler. *Loadbang - Programming Electronic Music in Pd.* Wolke, Hofheim, 2009.

[13] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the 1st International Conference on Music Information Retrieval (ISMIR 2000)*, 2000.

[14] Ingo Mierswa and Katharina Morik. Automatic feature extraction for classifying audio data. *Machine Learning*, 58, 2005.

[15] Meinhard Müller. *Information Retrieval for Music and Motion*. Springer, 2007.

[16] Miller Puckette. *Theory and Technique of Electronic Music*. World Scientific Publishing Co., 2007.

[17] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. In *IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-26, NO.1*, 1978.

[18] Yasushi Sakurai, Christos Faloutsos, and Masashi Yamamuro. Stream monitoring under the time warping distance. In *2007 IEEE 23rd International Conference on Data Engineering*, 2007.

[19] Roger N. Shepard. Circularity in judgements of relative pitch. *Journal of the Acoustical Society of America*, 36:2346–2353, 1964.

[20] George Tzanetakis, Andrey Ermolinsky, and Perry Cook. Pitch histograms in audio and symbolic music information retrieval. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2003)*, 2002.

[21] Peter D. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15, 1967.

[22] Piotr Wendykier. *JTransforms*.
http://sites.google.com/piotrwendykier/software/jtransforms.