Entwicklung einer Komponente zur Ermittlung regelmäßiger Betonungsmuster in einem Audiostream

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Informatiker

an der
Fachhochschule für Technik und Wirtschaft Berlin
Fachbereich Wirtschaftswissenschaften II
Studiengang Angewandte Informatik

1. Betreuer: Prof. Dr. -Ing. Thomas Jung

2. Betreuer: Dipl. Inf. Sebastian Oschatz

Eingereicht von Jens Röhner
Berlin, 2. August 2007

Inhaltsverzeichnis

1 Einführung

1.1 Motivation 7

1.2 Zielsetzung der Arbeit 7

1.3 Glossar 8

2 Grundlagen 13

2.1 Musikalische Parameter 13

2.1.1 Grundschlag 13

2.1.2 Takt 14

2.1.3 Timing 15

2.1.4 Offbeat 15

2.1.5 Synkope 15

2.1.6 Polyrhythmus 15

2.1.7 Perkussion 15

2.2 Psychoakustik 16

2.2.1 Frequenzgruppen 16

2.2.2 Dezibel 17

2.2.3 Weber-Fechner-Gesetz 17

2.2.4 Lautstärke 17

2.3 Signalverarbeitung 17

2.3.1 Digitales Audio 19

2.3.2 Diskrete-Fourier Transformation 19

2.3.3 Fensterfunktion 20

2.3.4 Hüllkurve 21

2.3.5 Digitale Filter 22

2.3.6 Kammfilter 23

2.3.7 Kompression 24

2.4 Beat-Tracking 25

2.4.1 Kognitive Prozesse 25

2.4.2 Music-Information-Retrieval 26

2.4.3 Eingabedaten 26

2.4.4 Anwendungen für MIR 27

2.4.5 Onsets 28

2.4.6 Inter-Onset-Intervall 28

2.4.7 Periodizität ... 29
2.5 VVVV ... 29
3 Stand der Technik ... 31
3.1 Vorgehensweise ... 31
3.2 Onset Detection – Akzentuierungen ermitteln ... 32
3.3 Periodizität – Tempo-Hypothesen aufstellen ... 34
3.4 Tracking - Positionen der Grundschläge ... 36
4 Anforderungen ... 38
4.1 Anwendungsfälle ... 38
4.2 Bedienbarkeit ... 38
4.3 Leistungsanforderungen ... 39
4.4 Implementierungsanforderungen ... 39
5 Analyse ... 40
5.1 Arbeitsmittel ... 40
5.1.1 VST ... 40
5.1.2 FFTW ... 41
5.1.3 Visual C++ 2005 Express Edition ... 41
5.1.4 Delphi und Borland Developer Studio 2006 ... 41
5.1.5 DirectShow ... 41
5.2 Theoretische Ansätze ... 42
5.3 Arbeitsverfahren ... 44
6 Design ... 45
6.1 Längen- und Zeiteinheiten der Ausgangsdaten ... 45
6.2 Vorverarbeitung ... 46
6.2.1 Monosignal bilden ... 46
6.2.2 Fensterfunktion anwenden ... 46
6.2.3 Überführen in die Frequenzdomäne ... 47
6.3 Onsets ermitteln ... 48
6.3.1 Unterteilung in Subbänder ... 48
6.3.2 Framebasierte Onset-Detection ... 49
6.3.3 RMS ... 49
6.3.4 Kompression ... 49
6.3.5 Tiefpassfilter ... 50
6.3.6 Differenz ... 51
6.3.7 Gewichtung ... 51
6.3.8 Kanäle bilden ... 52

6.4 Tempo-Hypothese aufstellen ... 53
6.4.1 Niedrige Werte verwerfen ... 53
6.4.2 Die Kammfilterbank ... 54
6.4.3 Resonanz auswerten und die Tempo-Hypothese bestimmen ... 55
6.4.4 Peakpicker ... 55
6.5 Positionen der Grundschläge ... 56
6.5.1 Agency ... 56
6.5.2 Aktionen der Agenten ... 56
6.5.3 Verwaltung der Agenten durch die Agency ... 58
6.5.4 Auswahl zwischen den Kanälen ... 59
7 Implementierung ... 60
7.1 Modellierung ... 60
7.2 Beattracker-Node ... 61
7.2.1 TMBeattrackerNode ... 61
7.2.2 TVSTHost und TVSTPlugin ... 62
7.3 VST-Plugin ... 63
7.4 Initialisierungsphase von Host und Plugin ... 63
7.5 Tracker ... 65
7.6 Die Verarbeitungskette ... 66
7.7 Die Klassen der Verarbeitungskette ... 67
7.7.1 FFT ... 67
7.7.2 OnsetDetection ... 67
7.7.3 Resonator ... 68
7.7.4 Interval ... 69
7.7.5 BeatDetection ... 70
7.7.6 Mixer ... 72
8 Test ... 72
8.1 Testaufbau Tempo-Extraktion ... 72
8.2 Testaufbau Positionen der Grundschläge ... 72
8.3 Testergebnisse ... 73
8.4 Kommentar ... 74
8.5 Auslastung ... 75
9 Ergebnis ... 76
9.1 Vergleich mit der Zielsetzung ... 76
9.1.1 Funktionalität im vvvv-System ... 76
9.1.2 Leistungsanforderungen ... 76

9.1.3 Bedienbarkeit 77
9.2 Ausblick 77
9.2 Analyse der Testergebnisse 78
9.3 Bewertung und Verbesserungsmöglichkeiten 78
9.4 Fazit 80

Anhang

Quellen 82
Abbildungsverzeichnis 83
Tabellen 84
Links 85

1 Einführung

1.1 Motivation

Grundschlag und Rhythmus gehören zu den wichtigsten Eigenschaften eines Musikstücks. Sie ordnen es zeitlich und schaffen Spannung. Menschen können diese Parameter, auch ohne musikalische Vorbildung, meist intuitiv erfassen. Computergestützte Algorithmen tun dies jedoch noch nicht in befriedigender Weise, so stellt die Untersuchung von Musik neben der Spracherkennung ein aktuelles Forschungsfeld der Audioanalyse dar. Eine korrekte, automatisierte Erkennung des Grundschlags eröffnet viele darauf aufbauende Möglichkeiten :

Suche und Schnitt von Sektionen, schnell und passgenau
Angleichung von zwei Musikstücken und eine nahtlose Überblendung
Erleichterung einer automatischen Transkription von Notenwerten
Synchronisation mit anderen Medien
Rhythmus-Erkennung und weiter eine Genre-Klassifikation um Musik-Datenbanken zu verwalten

1.2 Zielsetzung der Arbeit

Ziel dieser Arbeit ist die Ermittlung des Grundschlags in einem Musiksignal ohne Festlegung auf einen Stil oder verwendeten Instrumente. Außer den rohen Samplewerten werden keine zusätzlichen Informationen benötigt. Der Algorithmus arbeitet in Echtzeit und ist nicht auf die Eingabe von Parametern anhand der Benutzer angewiesen.
VVVV ist ein Software-System, mit dem die Generierung und Manipulation multimedialer Inhalte möglich ist. Die Arbeit wird als Komponente in das vvvv-System eingefügt. Es besteht die Möglichkeit ein visuelles oder akustisches Feedback-Signal auszugeben und damit die Ergebnisse der Analyse zu überprüfen.

1.3 Glossar

Analog Ein kontinuierlicher Vorgang der sich über einen kontinuierlichen Zeitraum erstreckt. Klänge der realen Welt sind analog.

ADSR Attack-Decay-Sustain-Release: Phasen eines musikalischen Tons (siehe Einschwingzeit).

Audiosignal Druckwellenschwankungen in Flüssigkeiten oder Gasen die in einem Frequenzbereich liegen, der durch die menschliche auditive Wahrnehmung verarbeitet werden kann (20-20000 Hertz).

Amplitude ist die maximale Auslenkung einer Schwingung

Bottom-up und Top-Down sind konträre Strategien zur Problemlösung. Bottom-Up bezeichnet dabei den Lösungsansatz, bei den kleinen Einheiten zu suchen und diese immer weiter zu größeren Einheiten zusammenzufügen. Die Top-Down-Strategie abstrahiert die Zusammenhänge und arbeitet sich davon ausgehend ins Detail. Bei der Kognition bedeutet Bottom-Up die direkte Verarbeitung aufgenommener Reize. Top-Down ordnet die Reize in einem bedeutungshaltigen Gesamtzusammenhang ein.

Clicktrack Mit einer Musik-Software erstellte regelmäßige Folge von kurzen Tönen die, ähnlich einem Metronom, einen Grundschlag repräsentieren.

Dezibel (dB) gibt das Verhältnis zweier Größen logarithmisch an. Der kleinste durch das menschliche Gehör wahrnehmbare Sinuston von 1000 Hz hat einen Schalldruckpegel von $2*10^{-5}\,Pa$, er wird als Bezugsgröße *(p0)* verwendet um die Intensität von Klangereignissen *(p)* anzugeben. Der Schalldruckpegel *L* in dB ergibt sich durch: *L = 20*log (p/p0).*
(Tabellen zu Dezibel siehe Anhang)

DirectShow Eine Multimedia-Schnittstelle unter Windows die das Verarbeiten von Audio- und Video-Datenströmen ermöglicht.

DLL (Dynamic-Link-Library) Ein fertiges Programm-Modul auf der Windows-Plattform, das

jedoch nicht selbstständig ausgeführt werden kann. Eine DLL ist eine Funktionsbibliothek und bietet ausführbaren Programmen (EXE) oder anderen DLLs Dienste an.

Dynamik Kennwert für den Verlauf der Lautstärke einer Folge von Klängen

Einschwingzeit Ein musikalischer Ton beginnt nicht abrupt, sondern benötigt eine bestimmte Zeit um sich aufzubauen. Die Einschwingzeit unterteilt sich in zwei Phasen: Bei der Attack-Phase steigt die Lautstärke des Tons stark und erreicht einen maximalen Wert. In der folgenden Decay-Phase fällt die Lautstärke wieder gering ab und wird auf einem bestimmten Level gehalten. Weitere Phasen sind: Sustain, das Halten eines Levels. Release, das Ausklingen, daher die relativ langsame Abnahme der Lautstärke eines Tons.

Frame Die gleichmäßige Unterteilung eines Stroms aus Audiosamples in Gruppen ergibt einzelne Frames (deutsch Rahmen).

Frequenzband Ein zusammenhängender Frequenzbereich.

Frequenzspektrum Eine beliebige periodische Wellenform lässt sich durch eine unendlich große Summe von überlagernden Sinusschwingungen erzeugen bzw. zerlegen. In einem Frequenz-spektrum werden die Amplituden der einzelnen Sinusschwingungen in einem Koordinatensystem (x= Frequenz, y= Amplitude) eingetragen.

Frequenz Die Anzahl von Ereignissen in einem bestimmten Zeitraum, zum Beispiel Grundschläge pro Minute oder Schwingungen pro Sekunde: f=1/T mit T=Schwingungsdauer (siehe auch Hertz).

Grundfrequenz siehe Ton

Harmonie Ein harmonischer Ton in der Musik ist eine Schwingung die sich nur aus einer Grundfrequenz und den Obertönen dazu zusammensetzt, im Frequenzspektrum als diskrete Linien zu sehen. Ein perkussiver Klang hat ein kontinuierliches Spektrum.

Hertz (Hz) Die Schwingungsdauer ist der Zeitraum in dem eine ganze Schwingung durchgeführt wird. Hertz ist die Kenngröße für die Frequenz der Schwingung: Hertz = Sekunde / Schwingungsdauer. kHz =1000 Hz.

IBI Inter-Beat-Intervall, der zeitliche Abstand zwischen zwei Grundschlägen.

IOI Inter-Onset-Intervall, der zeitliche Abstand zwischen zwei Onsets(siehe Beat-Tracking S.26). Die beiden Onsets müssen nicht angrenzen, sondern es können auch andere Onsets dazwischen liegen.

JND (just-noticeable-difference) Ein gerade noch wahrnehmbarer Unterschied in der Intensität eines Reizes (siehe auch Weber-Fechner-Gesetz).

Klang Zusammenfassung aller möglichen akustischen Ereignisse.

Metrik / metrische Ebene Der Grundschlag ist eine metrische Ebene. Tempo eines Musikstücks

Monophon Musik bei der zu jedem Zeitpunkt nur eine Ton erklingt.

MIDI (Music-Instrument-Digital-Interface) ein kompaktes Datenformat mit dem elektronische Musikinstrumente und Effektgeräte gesteuert und Daten dazwischen übertragen werden können. Die Geräte müssen über eine MIDI- Hardwareschnittstelle verfügen. In einer MIDI-Steuerinformation werden die drei Eigenschaften codiert: Frequenz, Intensität und Bezeichnung des Instrumentes. Bei dem Anschlag der Taste eines MIDI-Instrumentes werden folgende Informationen festgehalten: Zeitliche Beginn der Note und Verlauf des Drucks auf die Taste.

Periodizität Gleichmäßig wiederkehrende Eigenschaften.

Phase (Phasenwinkel) für eine Sinusschwingung ist dies der Winkelwert, den die Schwingung bei der Zeit t=0 hat.

Pascal (Pa) Einheit des Luftdrucks: $1\,Pascal\,(Pa) = 1\,N/m^2$ Die Kraft eines Newton verteilt auf einem Quadratmeter. Der Luftdruck liegt bei $10^5\,Pa$.

Polyphon Musik bei der mehrere Töne gleichzeitig erklingen. Ein Akkord auf einer Gitarre, bei der mehrere Saiten gleichzeitig schwingen ist ein polyphoner Klang.

Query-by-humming Eine gesummte oder anders erzeugte monophone Melodie dient bei bei

diesem Softwaresystem für eine Anfrage an eine Musik-Datenbank, mit dem Ziel der Zuordnung zu einem Musiktitel. Die Analyse bezieht sich auf melodische und rhythmische Merkmale.
Query-by-tapping Wie Query-by-humming, die Anfrage besteht jedoch aus dem geklopften Rhythmus, zum Beispiel über die Space-Taste.

RPF (Rhythm-Periodicity-Function) Eine Funktion die das periodische Auftreten von Betonungen in einem Musikstück beschreibt. Beispiele dafür ist die Ausgabe einer Kammfilterbank oder Autokorrelation.

Reiz oder Stimulus sind physikalische Eigenschaften, die über Sinnesorgane aufgenommen werden. Die Reizschwelle bezeichnet die Grenze ab der ein Reiz wahrgenommen wird. Bei Verstärkung des Reizes tritt ab einem bestimmten Grad Schmerz auf, die Schmerzschwelle. Die Reizschwelle ist für einen 1000 Hz Sinuston auf 0 dB festgelegt, die Schmerzschwelle liegt bei etwa 120 dB.

Rauschen Ein zufälliges Signal, dessen Verlauf nicht vorbestimmt werden kann. Ein frequenzbegrenztes Rauschen besitzt nur Frequenzanteile in einem bestimmten Bereich.

Resonanz Ein schwingungsfähiger Gegenstand zeigt eine Resonanz, wenn eine auf ihn übertragene Schwingung seiner Eigenfrequenz entspricht.

Schalldruck Schwankungen des Luftdrucks. Größenordnungen: Der Luftdruck liegt bei $10^5\,Pa$, ein normales Schallereignis hat etwa einen Schalldruck von $10^{-2}\,Pa$.

Schallintensität oder Energiestromdichte ist eine physikalische Größe, gilt also auch für Ultraschall: Schallintensität = Energiestrom / Empfängerfläche

Sample (Audio-) ist die Messung eines Schallereignis zu einem diskreten Zeitpunkt mit einer Zuordnung zu einem digitalen Wert.

SDK (Software-Developement-Kit) Quellcode, Programme und Dokumentation, die es Entwicklern ermöglicht Anwendungen für eine bestimmte Software zu entwickeln.

Signale sind physikalische Größen wie Licht-, Druck- oder Spannungsschwankungen, die als

Träger von Informationen dienen. Die Kontextabhängigen Merkmale eines Signals heißen Signalparameter. Signale sind zeitabhängig (befinden sich im Zeitbereich).
Sinusschwingung $x(t)=A0*\sin(\omega*t+\delta)$ Kreisfrequenz: $\omega=2\pi*f$ wobei f die Frequenz ist. Phasenwinkel: δ Amplitude: $A0$.

Sinusoide Schwingungen die durch die Winkelfunktionen Sinus oder Kosinus beschrieben werden können.

Ton ist eine harmonische Schwingung einer einzelnen Frequenz. In der Musik bezeichnet ein Ton den Grundton zusammen mit seinen Obertönen. Der Grundton oder Grundfrequenz ist die niedrigste Schwingung und die Obertöne sind jeweils ganzzahlige Vielfache davon. Über das Verhältnis der Obertöne ergibt sich die Klangfarbe.

Top-Down (siehe Bottom-Up)

WAV (für Wavelet engl. „kleine Welle“) Digitale Kopie des Schalldrucks zum Beispiel über ein Mikrofon aufgenommen. Eine Wav-Datei besteht aus zwei so genannten Chunks: Einem Format-Chunk der die Parameter über die Audiodaten enthält (Format-Typ, Anzahl der Kanäle, Samplerate, Byte-Rate, Bytes pro Sample, Anordnung der Bytes) und einem Daten-Chunk mit den Audiodaten**.**

Weber-Fechner-Gesetz besagt, dass eine logarithmische Zunahme der Reizintensität zu einer linearen Zunahme der Empfindungsstärke führt.

Winapi (Windows-Application-Programming-Interface) Programmierschnittstelle der Windows-Plattform mit der auf Funktionen des Betriebssystems zugegriffen werden kann.

µ-Law Bei dieser Kompression wird die Codierung an die logarithmische Wahrnehmung des Ohrs angepasst wird

2 Grundlagen

2.1 Musikalische Parameter

Musik ist eine durch den Menschen geplante und durchgeführte Erzeugung einer Abfolge von Klängen. Im Gegensatz zur Sprache, dient sie nicht primär, dazu Informationen zu übermitteln, sondern Empfinden auszudrücken und zu induzieren. Es werden folgende Eigenschaften unterschieden:

Rhythmus
Melodie
Harmonie
Klangfarbe

In dieser Arbeit geht es darum bestimmte Parameter über den *Rhythmus* eines Musikstücks zu ermitteln. Rhythmus bezeichnet die gesamte zeitliche Organisation eines Musikstücks, die sich durch die Abfolge von betonten und nicht betonten Zeitpunkten bildet. Der Rhythmus setzt sich hierarchisch, von unten nach oben geordnet, aus den folgenden Eigenschaften zusammen:

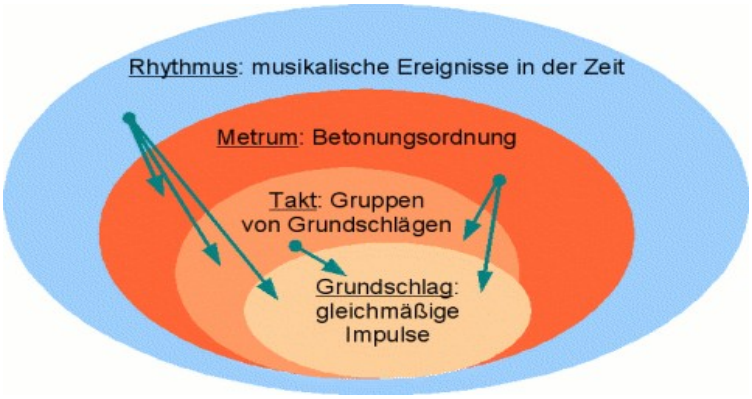


Abb.01: Hierarchischer Aufbau von Rhythmus

2.1.1 Grundschlag

Der Grundschlag oder *Beat* ist ein fundamentales Muster, das Musik in regelmäßige Zeitabschnitte unterteilt. Der Begriff kennzeichnet das einzelne Ereignis und die gesamte Folge über das

Musikstück hinweg. Der Grundschlag entspricht meist den Zeitpunkten, die durch das intuitive mitklopfen zu Musik bestimmt werden. Das *Tempo* eines Musikstückes ist die Anzahl der Grundschläge pro Minute und wird mit der Einheit *BPM* (Beats-Per-Minute) gemessen. Das Tempo kann sich auch innerhalb eines Musikstücks verändern. Eine andere Einheit dafür ist das *Inter-Beat-Intervall* (IBI), es ist gleich dem zeitlichen Abstand zwischen zwei Grundschlägen. Eine gerade noch durch den Menschen wahrnehmbare Änderung des Tempos liegt bei 4% [Gouy04]. Eine musikalische Note gibt ebenfalls eine relative zeitliche Dauer an. Die Dauer der musikalischen Noten wird durch das Inter-Beat Intervall bestimmt. Die ganze Note hat die längste Dauer. Die halbe Note dauert entsprechend halb so lang. Diese Aufteilung setzt sich weiter fort, bis zu der kleinsten üblichen Zeiteinheit: einer 64'tel Note.
Die Dauer einer dieser Noten ist dem Inter-Beat-Intervall zugeordnet. Meistens ist dies die Viertelnote: Bei einer Geschwindigkeit von 120 BPM ist ein IBI 0,5 Sekunden lang, daher ist auch die Viertelnote 0,5s lang. Die halbe Note 1s, die ganze Note 2s. Dementsprechend ist die Achtelnote 0,25s lang, die 16'tel Note 0,125s lang usw. .

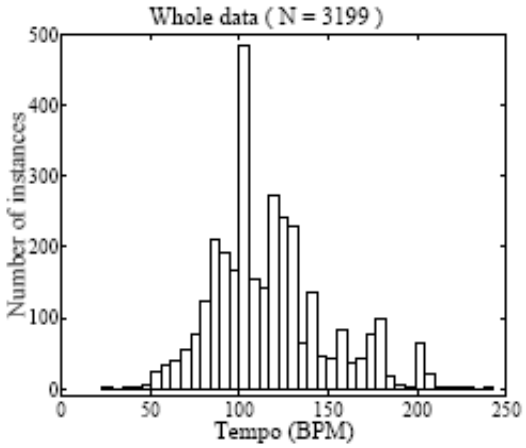


Abb.02: Verteilungen von Tempo bei 3199 Musikstücken

2.1.2 Takt

Über den Begriff Takt wird eine Folge von Grundschlägen zu einer Gruppe zusammengefasst. In der musikalischen Notation werden *Taktarten* als ein Bruch angegeben. Der Zähler gibt die Anzahl der Elemente in einer Gruppe an. Der Nenner definiert die zeitliche Dauer des einzelnen Elements über einen Notenwert. Mit der in der westlichen Musik häufig vorkommenden Taktart von 4/4 wird bestimmt, dass eine Gruppe die Anzahl vier hat (Zähler), wovon jedes der Gruppenelemente den Notenwert Viertelnote (Nenner) besitzt. Der ¾ -Takt, wie er bei einem Walzer verwendet wird,

bedeutet somit, dass immer drei Viertelnoten, die hinter einander kommen, in eine Gruppe gehören. Mit der *Zählzeit* wird die Folge der Grundschläge in einem Takt durchnummeriert. Die Zeiten dazwischen bekommen den Zusatz „und“. Beispiel für den 4/4-Takt : „1 und 2 und 3 und 4 und“.

2.1.3 Timing

Über das Timing wird angegeben, wann musikalische Ereignisse stattfinden. Ist das Timing exakt, so werden die Noten immer an den vorgesehenen Stellen gespielt. Dies ist eigentlich nur bei elektronisch erzeugter Musik der Fall, da Menschen immer ein bestimmtes Maß an Abweichung erzeugen. Die Abweichungen können auch einer Systematik unterliegen und Teil eines Stilmittels sein.

2.1.4 Offbeat

Offbeat bedeutet eine Betonung, die zwischen den Grundschlägen, also bei Verwendung von Zählzeiten, auf dem „und“ liegt. Es gibt Musikstile wie Reggae und Ska, die durchgängig diese Art der Akzentuierung nutzen.

2.1.5 Synkope

Eine Synkopierung bezeichnet die Betonung auf einer eigentlich als unbetont bestimmten Zählzeit. Durch den Bruch mit den Hörgewohnheiten, kann Spannung aufgebaut werden.

2.1.6 Polyrhythmus

Es wird von Polyrhythmus gesprochen, wenn mehrere Rhythmen gleicher zeitlicher Dauer übereinander geschichtet sind. Dies kommt beispielsweise häufig bei traditioneller lateinamerikanischer Musik vor.

2.1.7 Perkussion

Perkussive Instrumente bilden die Rhythmussektion in einer Gruppe von Klangerzeugern. Ein perkussives Instrument besitzt ein kontinuierliches Frequenzspektrum mit wenig harmonischen Anteilen. Dazu gehören unter anderen: Die *Bassdrum* oder *Große Trommel* liegt im tiefen

Frequenzbereich (unter 100 hz) und wird meistens über ein Pedal betätigt. Die *Snaredrum,* zu deutsch auch *Kleine Trommel* wird mit den Trommelstöcken angeschlagen und ist oftmals unten mit Metallkettchen, die ein schnarrendes Geräusch verursachen, bespannt.
Die perkussiven Instrumente können auch am PC oder Drumcomputer simuliert werden.

2.2 Psychoakustik

Das Forschungsfeld Psychoakustik beschäftigt sich mit der auditiven Wahrnehmung des Menschen. Es muss zwischen physikalisch messbaren Eigenschaften eines akustischen Signals und dem wahrgenommen Stimulus unterschieden werden, denn Wahrnehmung ist subjektiv.
Messungen mit präzis kalibrierten Geräten sind wiederholbar bei gleichem Ergebnis. Im Gegensatz dazu können Tests, die die Wahrnehmung eines Menschen betreffen, abweichende Ergebnisse liefern. Unterschiedliche Personen empfinden dasselbe akustische Ereignis verschieden, die Empfindung kann sogar bei derselben Person von einem Test zu einem anderen variieren.
Modelle der auditiven Wahrnehmung spielen eine wichtige Rolle bei dem Entwurf computergestützter Verfahren. Im Folgenden werden psychoakustische Eigenschaften, die für die vorliegende Arbeit von Relevanz sind und Einfluss auf den Entwurf von Beat-Tracking-Algorithmen haben, analysiert.

2.2.1 Frequenzgruppen

Wie später gesehen werden kann, führt die Aufteilung des analysierten Frequenzbereichs in einzelne Segmente, die separat untersucht werden, zu einer Verbesserung der Ergebnisse. In der hier vorgestellten Arbeit erfolgt die Aufteilung nach Frequenzgruppen.
Das menschliche Gehör nimmt Frequenzen in dem Bereich von 20 bis 20.000 Hertz wahr. Dieser Umfang ist in 24 Frequenzgruppen eingeteilt, die auf die Basilarmembran abgebildet werden können. Die Basilarmembran befindet sich in der Cochlea (Gehörschnecke) im Innenohr, hier werden die mechanischen Reize des Schalls in neuronale Reize umgewandelt, die dann von der Hirnrinde verarbeitet werden. Die Basilarmembran ist am breiten Eingang der Cochlea dünn und weist eine hohe Resonanz für hohe Frequenzen auf. Zum sich verjüngendem Ende der Schnecke hin, wird die Membran dicker und reagiert stärker auf tiefe Frequenzen. Die Cochlea ist insgesamt 30 mm lang, wobei auf eine Frequenzgruppe etwa die Länge von 1.3 mm entfällt. Die Größe der Frequenzgruppen liegt bis 500 Hz bei einem Frequenzbereich von etwa 100 Hz, darüber nimmt sie

ca. um den Faktor 1,19 zu. Die Empfindlichkeit in den Frequenzgruppen ist unterschiedlich. Während sie bei sehr tiefen und hohen Frequenzen gering ist, liegt der empfindlichste Bereich weitgehend in der Frequenzregion in dem auch die gesprochene Sprache liegt (ca. 300Hz bis 4kHz). Mit dem Modell der Frequenzgruppen können zwei Klänge daraufhin überprüft werden, ob sie als Konsonant (daher wohlklingend), oder Dissonant empfunden werden. Dazu müssen die Grundfrequenzen sowie alle Obertöne darauf untersucht werden, ob sie innerhalb der gleichen Frequenzgruppe liegen, ist dies der Fall so ist das Ergebnis ein Klang der als rau, pulsierend und unangenehm empfunden wird. Die Auflösung der Frequenz ist jedoch höher als es das Modell der Frequenzgruppen zulässt; es wird angenommen, dass neben der räumlichen auch eine zeitliche Komponente die auditive Wahrnehmung beeinflusst.

2.2.2 Dezibel

Der Bereich, von der kleinsten wahrnehmbaren bis zur größten Schallintensität, der an der Schmerzgrenze liegt, ist sehr groß. Dezibel (dB) gibt das Verhältnis zweier Größen logarithmisch an. Der kleinste durch das menschliche Gehör wahrnehmbare Sinuston von 1000 Hz hat einen Schalldruckpegel von $2*10^{-5} Pa$, er wird als Bezugsgröße *(p0)* verwendet um die Intensität von Klangereignissen *(p)* anzugeben. Der *Schalldruckpegel L* in dB ergibt sich durch:

$$L = 20*\log (p/p0).$$

Ein Schalldruckpegel von 0 dB bedeutet demnach also nicht eine völlige Stille (dB-Tabellen siehe Anhang S.81).

2.2.3 Weber-Fechner-Gesetz

Das Weber-Fechner-Gesetz besagt, dass eine logarithmische Zunahme der Reizintensität zu einer linearen Zunahme der Empfindungsstärke führt. Diese Regel gilt beispielsweise für das empfundene Gewicht eines Gegenstands und auch für die auditive Wahrnehmung.

$$k = \frac{\Delta R}{R}$$

ΔR ist der Wert um den der Reiz R zunehmen muss um bewusst wahrgenommen zu werden,

das Verhältnis k bleibt gleich. Bei einem ohnehin schon hohen Level der Schallintensität (z.B. Maschinenlärm), kann eine geringe Zunahme, wie sie durch das zusätzliche Surren einer Fliege verursacht wird, nicht wahrgenommen werden. In einem stillen Zimmer wird die gleiche Zunahme aber registriert.

2.2.4 Lautstärke

Die psychoakustische Größe der Lautstärke wird definiert als die Einordnung eines akustischen Ereignisses auf einer Skala von leise bis laut. Sie steht in Bezug zu der physikalischen Größe Schallintensität. Es folgt ein Beispiel für den Unterschied zwischen Lautstärke und Intensität: Zwei Sinustöne mit dem gleichen dB-Wert, die nicht in der gleichen Frequenzgruppe liegen, werden einzeln und zusammen abgespielt. Im zweiten Fall werden sie als doppelt so laut empfunden, obwohl sich die Intensität nicht verdoppelt hat. Die wahrgenommene Lautstärke unterscheidet sich bei verschiedenen Frequenzen. Ein niederfrequentes Rauschen wird deutlich lauter empfunden als ein hochfrequentes. Auch die Länge eines Klangereignisses spielt eine Rolle. So ruft ein Sinuston von kurzer Dauer ein weniger starkes Empfinden hervor, als wenn er mit einer längeren Dauer abgespielt wird.

2.3 Signalverarbeitung

Die digitale Signalverarbeitung ist die rechnergestützte Verarbeitung digitalisierter Signale der realen Welt. Das Ziel ist die Modifikation oder Informationsgewinnung. Im Rahmen der hier entwickelten Komponente wird Signalverarbeitung dazu verwendet das Signal aus digitalem Audio in eine Form zu bringen, die die Auswertung auf höherer Ebene erlaubt. Dazu werden einige Techniken angewendet, die in den folgenden Abschnitten dargestellt werden.

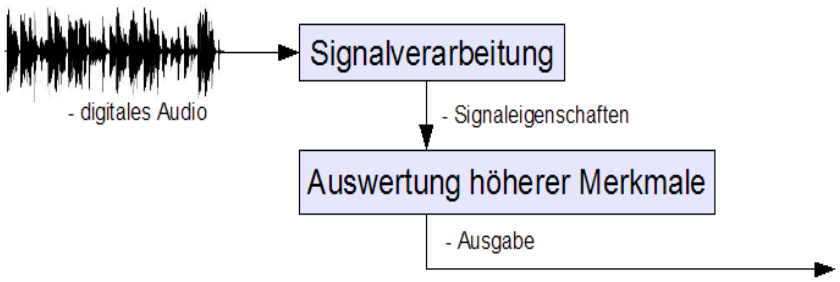


Abb.03: Die Signalverarbeitung bereitet Daten für die Auswertung höherer Merkmale vor

2.3.1 Digitales Audio

Digitales Audio sind Druckwellenschwankungen der Luft, die zu diskreten, gleichmäßigen Zeitpunkten abgetastet und in Zahlenwerte umgewandelt werden. Bei dieser Analog-Digital-Wandlung geht immer ein gewisses Maß an Information verloren. Die einzelnen Daten werden *Audiosample* genannt, sie haben einen Zeitpunkt und einen Wert in Dezibel. Folgen von Audiosamples werden zu *Frames* zusammengefasst,daher wird das gesamte Signal in Segmente einer bestimmten Länge, beispielsweise 1024 Samples zerteilt. Die Verarbeitung eines Audiosignals erfolgt *framebasiert.* Über die *Samplerate* wird angegeben wie viele Audiosamples pro Sekunde vorliegen, ein häufiger Wert dazu ist 44100.

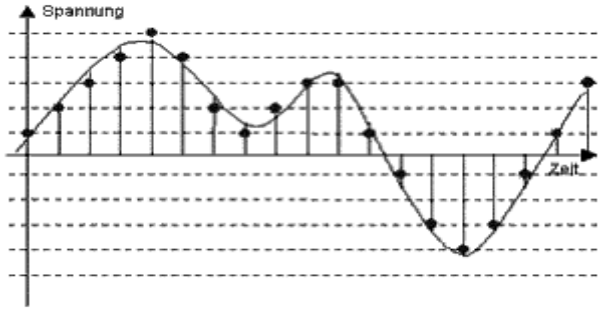


Abb.04: Ein Klang wird über ein Mikrophon in elektrische Spannung umgewandelt. Über einen ADC (Analog-Digital-Wandler) wird die Spannung zu diskreten Zeitpunkten abgetastet und einem festem Wert in dB zugeordnet.

2.3.2 Diskrete-Fourier-Transformation

Mit der Diskreten-Fourier-Transformation(DFT) kann ein beliebiges periodisches Signal in eine Summe von Sinus- und Kosinusfunktionen zerlegt werden, das Signal wird aus dem *Zeitbereich* in den *Frequenzbereich* überführt. Ein Beispiel für ein Signal im Zeitbereich ist die Wellenform eines Audiosignals.

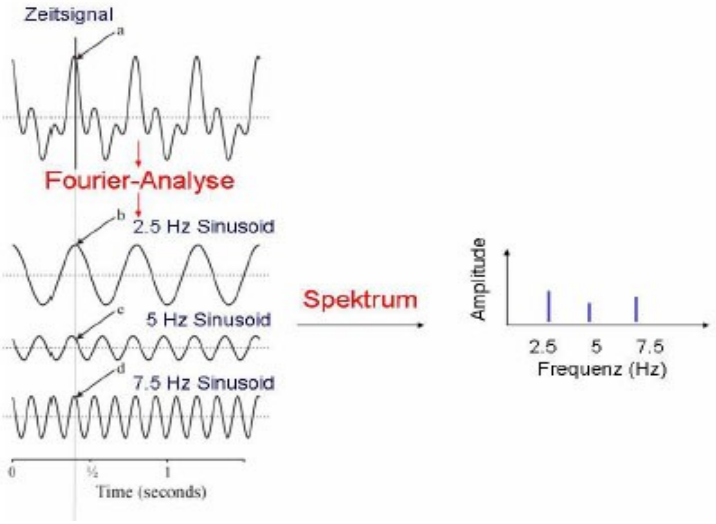


Abb.05: Fourier-Analyse (Erklärung siehe Text)

In der oberen Abbildung wird das periodische Signal im Zeitbereich (links oben) über die Fourier-Analyse in drei *Sinusoide* (Sinus- oder Kosinusfunktionen) zerlegt. Das Frequenzspektrum berechnet sich dann aus der *Amplitude* (Schwingungsweite) der einzelnen Sinusoide, diese werden als vertikale Balken in der Abbildung rechts angezeigt. Das Signal im Frequenzbereich heißt *Frequenzspektrum*. Im Prinzip wird bei der DFT das Signal mit einer Folge von Sinusoiden multipliziert. Das Ergebnis einer einzelnen Operation entspricht dem Anteil der Frequenz in dem Signal. Für ein Frame der Länger N müssen N^2 Operationen durchgeführt werden. Bei der digitalen Signalverarbeitung wird die *Schnelle-Fourier-Transformation(FFT)* verwendet. Die Berechnungen werden im komplexen Zahlenbereich durchgeführt und durch eine Teile-und-Herrsche Strategie, die auf schon berechnete Werte zurückgreifen kann, verringert sich der Rechenaufwand auf $N \log_2 N$. Viele Anwendungen der digitalen Signalerarbeitung sind dadurch erst möglich geworden.

2.3.3 Fensterfunktion

Die Fourier-Transformation zerlegt ein periodisches Signal, ein Audiosignal ist jedoch in den meisten Fällen nicht periodisch. Werden einzelne Frames verarbeitet so wird ein Signal an den Rändern abgeschnitten. Die Fourier-Transformation fasst das Signal als periodisch auf, mit einer Fortführung des Signals an der rechten Seite des Frames mit dem Signal an der linken Seite des Frames. Dadurch kommt es zu Stetigkeitssprüngen, dem *Leakage-Effect,* der zu einer starken Verfälschung der Frequenzanteile führen kann. Um diesen Effekt zu vermeiden, wird das Signal zur

Vorbereitung für eine Fourier-Transformation mit einer Fensterfunktion multipliziert, die die Werte an beiden Rändern eines Frames gegen Null gehen lässt. Damit sind stetige Übergänge gegeben, mit dem Nachteil einer Verringerung der spektralen Auflösung. Eine mögliche Fensterfunktion ist das *Hamm-Fenster*:

$$0.54 - 0.46\cos\left(\frac{2\pi n}{N}\right), n = \frac{-M}{2}, \ldots, \frac{M}{2}$$

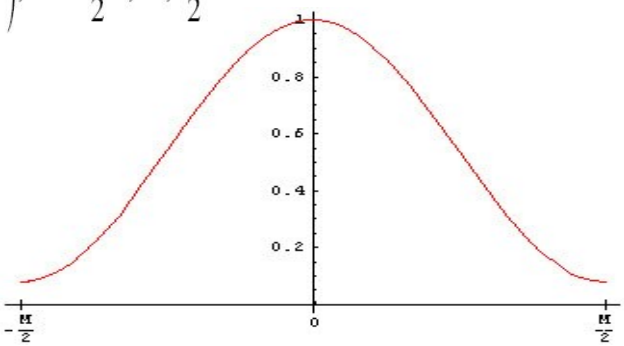


Abb.06: Hamm-Fensterfunktion

2.3.4 Hüllkurve

Mit der Hüllkurve kann die Dynamik eines Klangereignis erfasst werden. Die *Dynamik* beschreibt den zeitlichen Verlauf der Intensität eines Audiosignals. Sie wird gebildet, indem die positiven Spitzen der Wellenform miteinander verbunden werden. Dadurch kommt es auch zu einer starken Reduktion der Datenrate. Bei Audio-Synthesizer wird eine *modulierende* Hüllkurve eingesetzt, um den Klang einer Note zu manipulieren: Die Form der Welle wird an die Hüllkurve so angepasst, dass alle Werte unterhalb des vorgegebenen Verlaufs liegen. Dabei werden die Attack, Delay, Sustain und Release-Phasen (*ADSR-Hüllkurve*) unterschieden. Die *Attack*-Phase beschreibt das Einsetzen der Note.

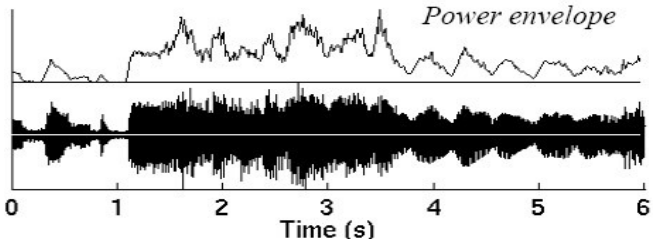


Abb 07: Die Hüllkurve ergibt sich durch die Verbindung der Spitzen in der Wellenform

2.3.5 Digitale Filter

Die typische Funktion von Filtern besteht darin, bestimmte Anteile in einem Signal zu unterdrücken. Während ein analoger Filter elektronische Komponenten wie Widerstand und Kondensator verwendet, wird die digitale Filterung auf einem Prozessor durchgeführt, wobei oft spezialisierte Signalprozessoren zur Anwendung kommen. Ein *Tiefpass-Filter* (engl. *Lowpass*) lässt Frequenzen, die sich unterhalb einer bestimmten *Grenzfrequenz* befinden durch, während Frequenzen die darüber liegen unterdrückt werden. Die Grenzfrequenz ist die Frequenz bei der die Amplitude auf 0.707 des Ursprungswerts fällt.

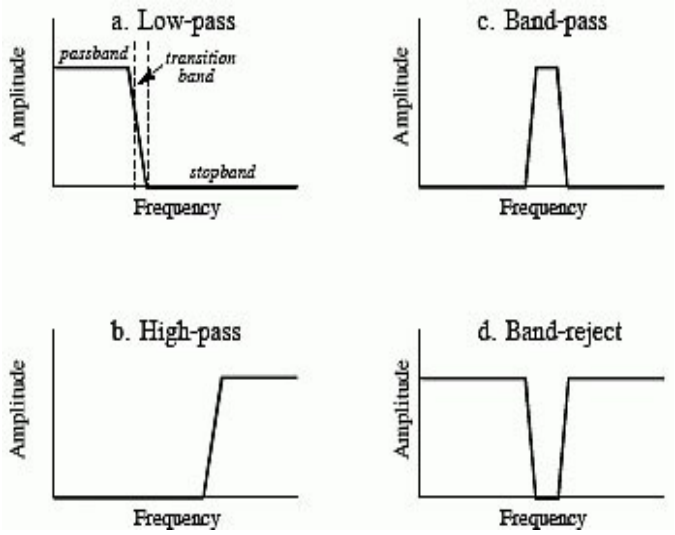


Abb.08: Filter (Erklärung siehe Text)

In der oberen Abbildung sind die Frequenzspektren von vier Filterarten dargestellt. Die Eingabe ist ein so genannter *Frequenzgang*. Dabei werden in ihrem Frequenzbereich kontinuierlich ansteigende Sinusoide durch die Filter geleitet und in einem Frequenzspektrum erfasst.
Der Tiefpass-Filter in der oberen rechten Ecke lässt sich über eine Umkehrung in einen *Hochpass-Filter* umwandeln. Durch Kombination dieser beiden Filterarten lässt sich ein *Bandpass*- oder *Bandsperre-Filter* bauen. Der Frequenzbereich der ungehindert durchgelassen wird heißt *Passband*, in dem *Stopband* dagegen werden die Frequenzen unterdrückt. Der Übergangsbereich dazwischen heißt *Transition-Band*. Filter unterscheiden sich auch durch ihre Verarbeitungsweise: Ein *Finite-Response-Filter(FIR)* berechnet die Ausgabe rein aus den Eingabedaten. Ein *Infinite-Response-Filter(IIR)* arbeitet rekursiv, er bezieht daher auch Ausgabewerte in die Berechnung mit ein, dies macht sie sehr schnell.
Ein Beispiel für den einen rekursiv arbeitenden Filter ist der Chebyshev-Filter, er wird in dieser

Arbeit verwendet: Chebyshev-Filter werden benutzt, um Bänder von Frequenzen voneinander zu trennen, ihr Vorteil liegt in einem kurzen Transition-Band. Ein Nachteil ist, dass es in dem Passband zu einer Verstärkung bestimmter Frequenzen kommen kann. Dies drückt sich durch eine *Welligkeit* (engl. *Ripple*) in dem Spektrum des Frequenzgangs aus. Eine präzisere Trennung von Frequenzen durch den Chebyshev-Filter mit einem minimalen Transition-Band hat eine erhöhte Welligkeit zur Folge.

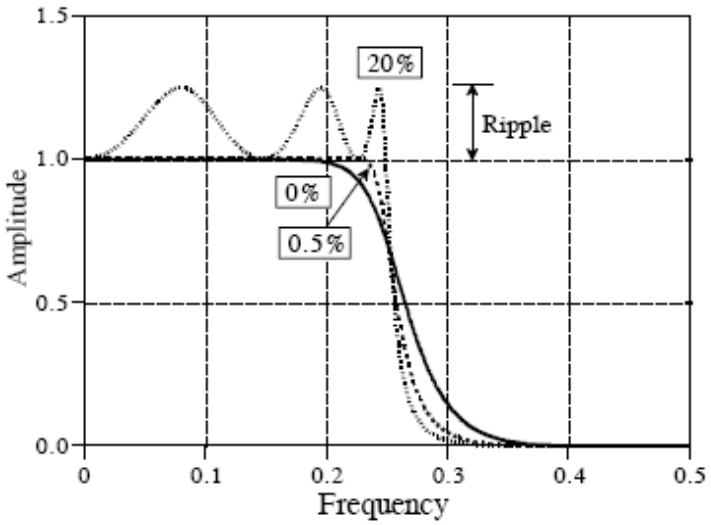


Abb.09: Frequenzgang dreier Chebyshev-Lowpass-Filter. Die Grenzfrequenz liegt bei etwa 0.25. Je kürzer das Transition-Band, um so größer die Welligkeit

2.3.6 Kammfilter

Kammfilter werden dazu benutzt in einem Signal periodisch auftretende Eigenschaften ausfindig zu machen. Dazu wird eine Addition des Signals mit einer zeitlich verzögerten Version von sich selber vorgenommen. Der Wert der *Verzögerung* oder *Delay* gibt die Periodendauer an, die in dem Signal deren Stärke in dem Signal ermittelt wird. Der Kammfilter hat einen periodischen Frequenzgang, dessen Stopband breit ist und dessen Passband den Zinken eines Kamms ähnelt, daher der Name. Es gibt zwei unterschiedliche Arten: Im *Feedforward-Ansatz* wird die Summe eines Samples durch Addition mit einem Sample gebildet, welches um den *Delay* zeitlich davor in dem gleichen Signal liegt. Der *Feedback-Ansatz* verwendet rekursiv Ausgabewerte. Er berechnet sich durch:

$$y[n]=x[n]+\alpha\, y[n-K]$$

Die Variable n gibt den aktuellen Zeitpunkt an, $x[n]$ ist das Eingabesignal und $y[n]$ das Ausgabesignal. K ist der Delay, α gibt die *Verstärkung* (engl. *Gain*) an.

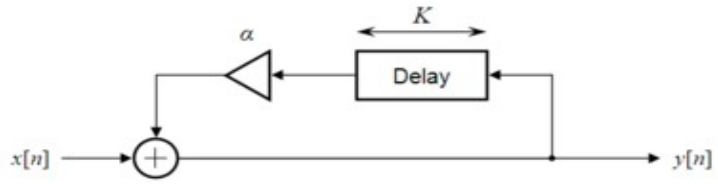


Abb. 10: Der Kammfilter in der Feedback-Version greift auf das um K verzögerte Ausgabesignal zurück

Bei einer Kamm-Filterbank werden mehrere Kamm-Filter nebeneinander verwendet, die unterschiedliche Delays haben. Durch Aufsummierung der Ausgabewerte für ein Signal ergibt sich die Energie in dem jeweiligen Filter. Eine Kamm-Filterbank gibt über die Energien der einzelnen Kamm-Filter die *Resonanz* auf verschiedene im Signal enthaltene Periodenlängen an.

2.3.7 Kompression

Mit der Technik der Kompression können akustische Signale in ihrem dynamischen Umfang beschränkt werden, dass bedeutet das niedrige Klangintensitäten verstärkt und hohe gedämpft werden. Die *μ-law* Kompression wird beispielsweise dazu verwendet, um akustische Signale über eine analoge Telefonleitung zu übertragen, dies führt zu einer Verbesserung der Tonqualität, das Signal wirkt insgesamt lauter. Bei digitaler Signalübertragung führt die Kompression eines akustischen Signals zusätzlich zu einer Verringerung der Datenrate, da der Wertebereich eingeschränkt wird. Der Wert von μ ist häufig auf 255 festgelegt.

$$s(x) = \frac{\ln(1+\mu * |x|)}{\ln(1+\mu)}, \mu = 255$$

Die logarithmische Verarbeitung eines akustischen Signals durch die Kompression entspricht in gewisser Weise der ebenfalls logarithmischen Verarbeitung durch die auditive Wahrnehmung.

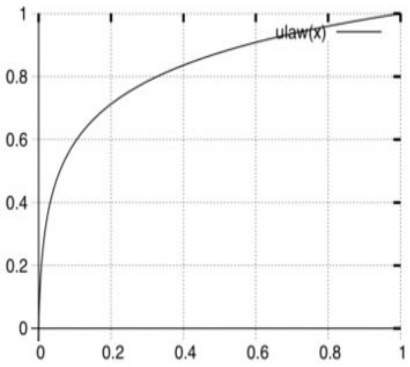


Abb.11: μ-law Kompression. Kleine Werte werden angehoben

2.4 Beat-Tracking

Beat-Tracking ist die Bestimmung der zeitlichen Positionen der Grundschläge in einem Musikstück durch ein Computerprogramm. Bei der *Tempo-Extraktion* wird die Geschwindigkeit eines Musikstücks ermittelt und in bpm angegeben. Eine Schätzung der Geschwindigkeit des Stücks wird hier *Tempo-Hypothese* genannt, sie ist Voraussetzung für das Beat-Tracking. Es gibt mehrere Anwendungsmöglichkeiten für das Beat-Tracking, sowohl direkt als auch indirekt.Es ist ein Beitrag zur Automatisierung der Erfassung und Darstellung von multimedialen Inhalten.

2.4.1 Kognitive Prozesse

Beim Hören von Musik entsteht in dem Zuhörer durch *Beat-Induktion* ein zeitlich gleichmäßiges Muster. Dies ist für Musiker eine essentielle Fähigkeit um eine Geschwindigkeit zu halten und sich mit anderen Musikern zu synchronisieren. Auch musikalisch ungeübte Menschen haben diese Fähigkeit. Für Honing [Hon99] scheint der Prozess der Beat-Induktion fundamental für die Verarbeitung und Kodierung zeitlicher Muster zu sein. Er sieht mehrere Gründe, wieso eine Modellierung der kognitiven Prozesse, die in einem Menschen dabei vor sich gehen, schwer am Rechner nachzuahmen sind: Der Prozess läuft nach einer Bottom-Up Strategie und sehr schnell ab, schon nach 5-10 Noten erkennt ein Mensch das Muster. Sobald das kognitive Muster angelegt ist, wird neues Material daraufhin überprüft, ob es zu dem bestehenden Rahmenwerk passt. Dies ist eine Top-Down Strategie. Über die Top-Down Strategie können auch Phänomene, wie die Zuordnung eines synkopierten Beat zu dem richtigen Grundschlag erklärt werden. Die Schwierigkeit besteht darin, die richtige Balance zwischen Stabilität und Reaktivität des Systems zu finden.

Bei Musik handelt es sich um Informationen die für Menschen bestimmt sind. Um wahrgenommene Eigenschaften darin zu ermitteln, können kognitive Modelle zum Vorbild für rechnergestützte Algorithmen dienen. Durch Testen dieser Verfahren können auch Schlüsse über die Qualität eines kognitiven Modells gezogen werden.

2.4.2 Music-Information-Retrieval

Beat-Tracking ist ein Teil eines relativ neuen Forschungsfeld in der Informatik des Music-Information-Retrieval (MIR), hierbei wird ein Musikstück auf bestimmte musikalische Eigenschaften untersucht:

Geschwindigkeit
Genre
Emotionaler Gehalt
Rhythmus
Melodie
Harmonie

Rhythmische Parameter spielen eine besonders wichtige Rolle, denn korrekt detektierte Grundschläge bilden das Fundament, auf dem weitere Ermittlungsschritte aufbauen können. Fast alle musikalischen Ereignisse beziehen sich auf den Grundschlag. Als Beispiel wären hier Akkordwechsel genannt.

2.4.3 Eingabedaten

Eingabedaten der MIR-Verfahren sind geschriebene Notenwerte, MIDI oder Audio. Notenwerte sind diskret und besitzen einen hohen Abstraktionsgrad, die Struktur ist anhand der Einteilung in Takte direkt ablesbar. MIDI-Daten geben den Zeitpunkt, Art und die Dauer eines musikalischen Ereignis an. Sie sind ebenfalls diskret und auf einem hohen Abstraktionsgrad. Audiodaten erscheinen für die Analyse als kontinuierlicher Strom (auch wenn es sich um diskrete Samples handelt) und haben einen niedrigen Abstraktionsgrad. Die metrische Struktur ist implizit enthalten, dass heißt die Daten müssen erst interpretiert werden.

2.4.4 Anwendungen für MIR

Durch die digitale Speicherung ist es möglich sehr große Datenbestände an Musik vorzuhalten. Von wirtschaftlichen Interesse sind hier Online-Shops, die Dateien zum kostenpflichtigen herunterladen bereitstellen. Die einzelnen Dateien müssen geordnet und beschrieben werden, dabei ist es bei der enormen Menge notwendig automatisierte Verfahren anzuwenden. Neue Arten der Anfrage durch den Nutzer sind in der Entwicklung, zum Beispiel Query-by-humming oder Query-by-tapping.

Transkription

Transkription beschreibt die Überführung von Musik, die als Audio vorliegt, in Notenwerte. Auf der Struktur die durch die Folge von Grundschlägen gegeben ist, können harmonische Töne und Akkorde besser ermittelt werden. Die so erzeugten Noten können z.B. als Lehrmittel dienen. Werden die gefunden Daten weiter in MIDI überführt, kann eine Übertragung von einem Klangerzeuger zum nächsten stattfinden.

Genre-Klassifizierung

Eine Klassifizierung von Musik in Genres, eventuell mit einer weiteren Verfeinerung betreffend des emotionalen Gehalts, beinhaltet als wichtige Parameter das rhythmische Grundgerüst und Tempo. Die zuverlässige Erkennung von Melodien und Akkorden in einem polyphonen Signal ist mit den heutigen technischen Verfahren noch lange nicht erreicht. Rhythmus-Eigenschaften sind im allgemeinen leichter zu ermitteln und können durch eine Kombination mit dem gefundenen harmonischen Gehalt eines Musikstücks, die Basis für eine Einteilung in Genres bilden. Interessant ist auch das Auffinden von gleichartigen Rhythmen für die Erstellung von Zuordnungslisten. Diese können bei der Suche in Datenbeständen als Alternative angezeigt werden oder die automatische Generierung von Abspiellisten unterstützen.

Synchronisierung

Eine Angleichung der Geschwindigkeit mit einer anderen Audioquelle, auf Basis der Grundschlag-Frequenz, ermöglicht die Überblendung zwischen Musikstücken und ein kontinuierliches Abspielen, wie es in manchen Genres, vor allem der elektronisch erzeugten Musik, üblich ist und heute zum größten Teil noch manuell vorgenommen wird. Was diese Arbeit betrifft, liegt das Ziel

vor allem in der Synchronisierung von Audio und visuellem Material, dies muss sich nicht auf Animationen und Videos beschränken, sondern kann auch konkrete physische Geräte wie Lichtmaschinen mit einbeziehen. Eine automatisierte Synchronisation bedeutet jedoch nicht zwangsläufig eine einfache Ersetzung des Menschen, sondern eröffnet die Möglichkeit, sich auf Interpretationen auf einer höheren Ebene zu konzentrieren.

Segmentierung

Eine Segmentierung einer Audiodatei auf Basis der Grundschläge führt zur Erleichterung der Editierung und ermöglicht das passgenaue Schneiden und Suchen.

2.4.5 Onsets

Ein Onset ist der Beginn eines musikalischen *Ereignisses.* Dies kann ein Klang sein, ein einzelner Ton oder ein Zusammenklang mehrerer Töne. Bei einer ADSR-Hüllkurve(siehe S.19) stimmt ein Onset mit der Attack-Phase überein. Diese Arbeit konzentriert sich auf die Ermittlung von Onsets perkussiver Instrumente. Charakteristisch dafür ist das relativ breite, kontinuierliche Frequenzspektrum des Klangs, sowie eine kurze Dauer bei hoher Energie. Ein Onset hat drei Eigenschaften: einen Zeitpunkt, die Stärke und einen Frequenzbereich. Ziel ist es Onsets zu finden, die auf dem Grundschlag liegen, für diesen Zustand gibt es Indizien: Starke Onsets, Onsets die im tiefen Frequenzbereich liegen oder Onsets bestimmter perkussiver Instrumente, wie zum Beispiel Schläge der Snaredrum.

2.4.6 Inter-Onset-Intervall

Ein Inter-Onset-Intervall (IOI) ist der zeitliche Abstand zwischen zwei Onsets. Dieses Paar muss nicht angrenzen, sondern es kann beliebig in einer Menge von Onsets gebildet werden. Das Inter-Beat Intervall ist meist in der Menge der IOIs enthalten.

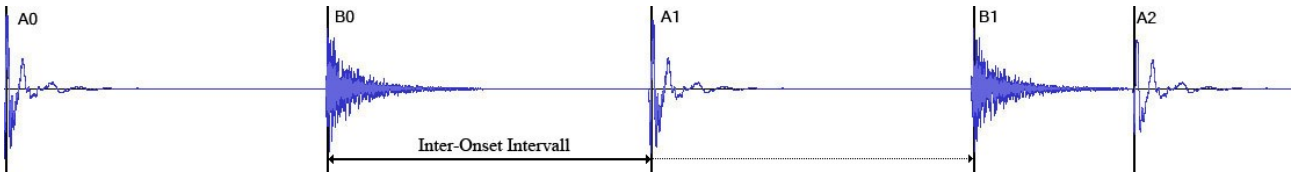


Abb.12: Angezeigt ist die Wellenform eines ganzen Taktes: A0,A1 und A2 sind Bassdrum-Schläge. B0 und B1 sind Snaredrum-Schläge. Zwischen Paaren von Onsets liegen Inter-Onset-Intervalle

2.4.7 Periodizität

Der Grundschlag ist ein gleichmäßig wiederkehrendes Ereignis. Die Anzahl der Grundschläge pro Minute entspricht der Frequenz, während das Inter-Beat-Intervall die Periodendauer darstellt. Für Musik die nur in Form von Audio vorliegt, ist die absolute Einteilung eines Tempos in richtig und falsch nicht möglich. Unterschiedliche Zuhörer können bei ein und demselben Musikstück verschiedene Grundschläge setzen, eine häufige Abweichung ist das verdoppeln oder halbieren der Frequenz. Des Weiteren kann sich das Tempo im Verlauf eines Musikstücks ändern, somit ist eine globale Tempoangabe nicht möglich .

2.5 VVVV

Das vvvv-System ermöglicht das Erstellen und die Manipulation von multimedialen Inhalten in Echtzeit über eine grafische Oberfläche. Einzelne Bausteine, *Nodes* genannt, werden mit Linien verbunden, über die Daten in eine Richtung gesendet werden können. Der so entstandene gerichtete Graph stellt den Programmcode dar. Die Eingabe kann über viele verschiedenartige Quellen vorgenommen werden, unter anderem: Maus, Video, Mikrofone und Touchpad. Möglichkeiten der Ausgabe sind beispielsweise Monitore, Beamer, Licht und Lautsprecher. Die Verarbeitung und Manipulation findet immer bei laufender Anwendung statt, sodass die Auswirkungen sofort zu sehen sind. Über die so genannten *Pins* an der Oberseite eines Nodes können ihm Daten zugeführt werden, an der Unterseite liegen die Pins die Daten ausgeben. Es können nur Pins verbunden werden, die den gleichen Datentyp haben. Wird mit der Maus über einen Pin gefahren, so werden alle Pins markiert, die damit verbunden werden können. Verbindungen die einen geschlossenen Kreislauf zur Folge hätten sind nicht möglich. Nodes können entweder über direkte Eingabe des Namen oder ein Auswahlmenü angelegt werden. Das vvvv-System läuft auf der Windows-Plattform und ist größtenteils in Delphi geschrieben. Für die Audioverarbeitung wird DirectShow verwendet.

In der folgenden Abbildung 13 ist ein vvvv-*Patch* zu sehen, bei dem ein Kreis erzeugt und über einen GDI-Renderer ausgegeben wird. In die beiden oberen Input-Nodes können Zahlenwerte eingegeben werden. Die Zahlenwerte werden über die Verbindungen an den Circle-Node gesendet.

Der obere der beiden Input-Nodes endet in einem Pin, der die Größe des Kreises bestimmt. Der andere Input-Node bestimmt über den Pin die Position des Kreises auf der x-Achse, sie ist hier um 0.43 nach rechts verschoben. Der Ausgabe-Pin des Circle-Nodes endet in dem Renderer-Node und bewirkt die grafische Ausgabe.

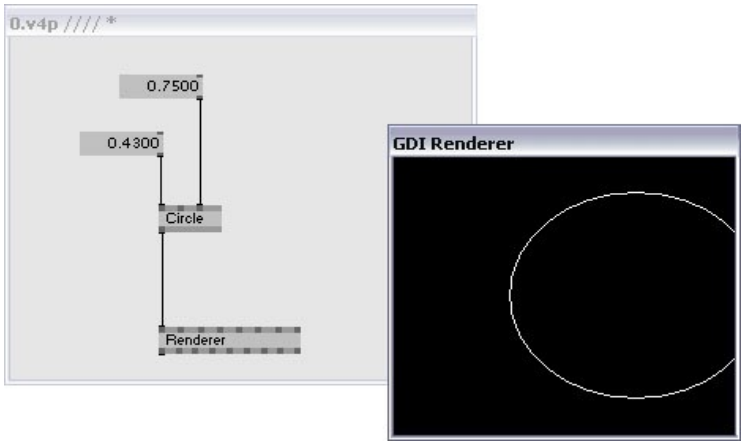


Abb.13: vvvv-Patch (Erklärung siehe Text)

Das Haupteinsatzgebiet von vvvv liegt bei größeren multimedialen Installationen. Dabei sind die Fähigkeiten zur Steuerung großflächiger Projektionen und der Interaktion über eine Vielzahl an Eingabequellen, wie Sensoren, Motion-Tracking oder Touchpads, von Vorteil. Interessant ist vvvv auch um Musik-Visualisierungen zu erstellen, da es Grafiken in Echtzeit generieren kann und über Möglichkeiten zur Audioanalyse verfügt.

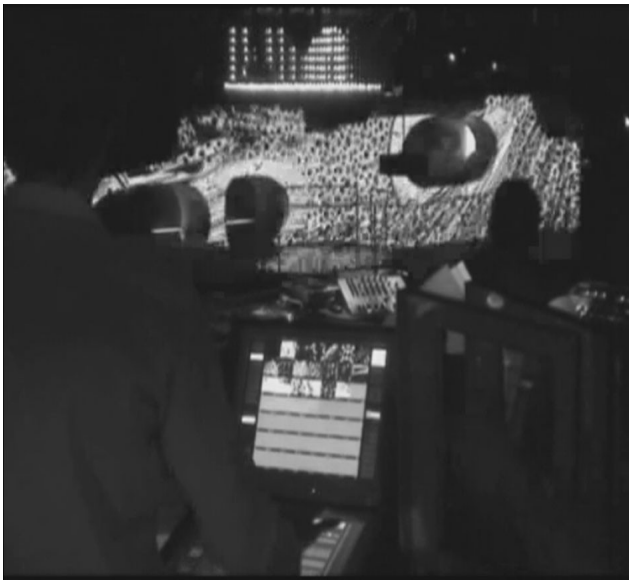


Abb.14: Feste Installation von Beamern und Monitoren in einem Club mit Steuerung durch das vvvv-System über ein Touch-Pad

3 Stand der Technik

Frühe Arbeiten die sich mit der Auswertung der rhythmischen Struktur von Musikstücken durch Computer befassen, entstanden in den 1980'er Jahren. Ein Beispiel ist die Arbeit von Schloss [Schlo85]: Sie hat die Transkription eines rein durch perkussive Instrumente erzeugten Audiosignals zum Inhalt. Onsets werden extrahiert, indem aus der Welle die Hüllenform gebildet wird. Diese wird mit Hilfe einer Regressionsgeraden auf Steigungen untersucht. Bei der 1995 entstanden Arbeit von Goto [Goto96] werden Musikstücke in Echtzeit auf rhythmische Eigenschaften untersucht. Dabei werden vordefinierte rhythmische Muster verwendet. Eine Neuerung ist es, mehrere Tempo-Hypothesen parallel zu verfolgen, dabei kommen so genannte *Multiple Agenten* zum Einsatz. Scheirer [Schei97] unterteilt den Frequenzbereich in fünf Teile und führt separat die Tempo-Extraktion aus. Dazu verwendet er eine Bank aus Kamm-Filtern. Heute besteht die Motivation für die Entwicklung von Verfahren zur Rhythmuserkennung in der großen Zunahme an digital gespeicherter Musik, die verwaltet werden muss. In diesem Bereich Tätige haben eine Plattform auf der *International Conference on Music Information Retrieval* , die jährlich abgehalten wird. Im Rahmen der Ismir oder der Organisation Mirex werden Wettbewerbe für Algorithmen abgehalten, die sich mit den Aufgaben des Music-Information- Retrieval befassen. Die Verfahren werden auf Basis von manuell gekennzeichneten Testdaten geprüft. Gewinner des letzten Mirex-Wettbewerbs zur Tempo-Extraktion ist das Verfahren von Klapuri [Klap04]. In dem Bereich Beat-Tracking konnte dagegen Beat Root von Dixon [Dix06] das beste Ergebnis erzielen.
Eine Abdeckung aller Arbeiten und Techniken zum Thema kann hier nicht erfolgen, es wird aber versucht, auf allgemeine Vorgehensweisen [Gouy04] einzugehen, die sich als zweckmäßig erwiesen haben.

3.1 Vorgehensweise

Das Verfahren des Beat-Tracking kann in drei Einzelschritte unterteilt werden. Der erste Schritt besteht darin, Onsets in dem Audiosignal zu detektieren. Im zweiten Schritt werden diese Eigenschaften auf ihre Periodizität hin untersucht. Daraus ergeben sich eine oder mehrere Tempo-Hypothesen. Im dritten Schritt wird der Audiodatenstrom in Hinblick auf die Tempo-Hypothese(n) untersucht, bei mehreren wird die erfolgreichste bestimmt. Das Ergebnis kann wiederum Auswirkungen auf die Aufstellung neuer Tempo-Hypothesen haben.

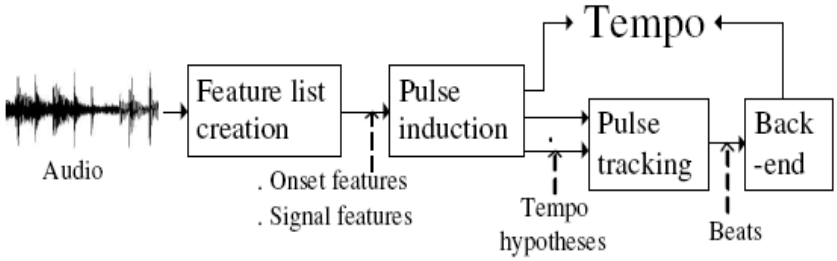


Abb.15: Aufbau eines Beat-Tracking-Systems. Auf Periodizität wird in dem „Pulse induction“ Teil untersucht, das Tempo ergibt sich aus der Ausgabe plus dem Tracking-Teil

Alle entwickelten Algorithmen zum Beat-Tracking haben diesen Aufbau, unterscheiden sich aber hinsichtlich der Methoden die in den einzelnen Phasen angewendet werden. Zwischen der dargestellten Vorgehensweise und dem kognitiven Modell der Beat-Induktion (siehe Kapitel 2.4.1) bestehen starke Übereinstimmungen. Im Folgenden werden die drei Phasen des Beat-Trackings ihrer Reihenfolge entsprechend näher erläutert.

3.2 Onset Detection – Akzentuierungen ermitteln

Bei der Onset-Detection geht es darum relevante Änderungen in dem Strom der Audiodaten zu ermitteln. Beat-Tracking konzentriert sich auf Onsets die wichtig in Hinblick auf den Rhythmus sind, darum stehen Änderungen in der Dynamik des Audiosignals im Mittelpunkt. Es gibt zwei Ansätze: Einmal die Hüllkurve (siehe Kapitel 2.3.4) der Wellenform zu bilden, und diese auf besonders große Steigungen zu untersuchen. Damit ist es möglich Anfänge einzelner Noten zu ermitteln.

Im zweiten Ansatz wird die Energie in einem Frame berechnet und jeweils die Differenz der Energie des Frames zu seinem Vorgänger gebildet. Ist sie positiv und ausreichend hoch, wird dieser Frame als Onset gewertet. Der framebasierte, abstrahierende Ansatz bedeutet weniger Rechenaufwand und scheint bei direktem Vergleich zu dem ersten Verfahren Vorteile bei der Untersuchung eines verzerrten Signals zu haben [Gouy04]. Um den Verlust an zeitlicher Genauigkeit zu verringern, können Frames überlappend verarbeitet werden. Gleich welcher der beiden Ansätze gewählt wird, letztendlich ist die Frage entscheidend, wie hoch der Wert sein muss, um als ein Onset zu gelten. Hier scheint eine Orientierung an der menschlichen Wahrnehmung die, wie in Webers Gesetz (siehe Kapitel 2.2.3) beschrieben, logarithmisch verläuft zu guten Ergebnissen zu führen [Klap04].

Die Unterteilung des gesamten Frequenzbereichs und die separate Durchführung der Onset-Detection, führt zur Verbesserung der Datenbasis auf dem die Tempo-Extraktion stattfinden kann . Dabei können Parallelen zu dem Modell der Kritischen Bänder (siehe Kapitel 2.2.1) gezogen werden. Für die Relevanz dieser Vorgehensweise spricht folgender Versuch:

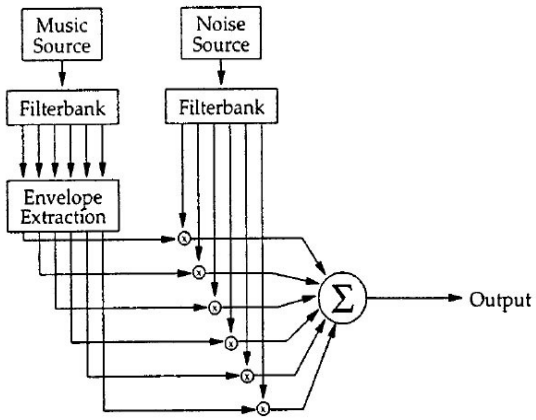


Abb.16: Versuch von Scheirer (Erklärung siehe Text)

Ein Musiksignal wird in sechs gleich große Frequenzbereiche unterteilt und in jedem Bereich die Dynamik berechnet. Die Dynamik jedes Bereichs moduliert die Dynamik eines Rauschen im gleichen Frequenzbereich. Aus der Summe der sechs Rauschen ergibt sich ein Audiosignal. Dieses Signal genügt, damit ein Zuhörer das ursprüngliche Musikstück identifizieren kann, da es den rhythmischen Gehalt des Musikstücks vollkommen widerspiegelt.
Scheirer[Schei97] sieht die Anordnung der Bänder nicht als ausschlaggebend. Die guten Ergebnisse die Klapuri[Klap04] mit seinem verfeinerten Modell erzielt sprechen aber dafür, dass die Art der Unterteilung doch wichtig ist. In seinem Verfahren werden in 36 Bänder separat Änderungen ermittelt und dann zu vier so genannten Kanälen zusammengeführt.

Abb.17: Aus der Wellenform werden Onsets entnommen und in einer metrischen Struktur untergebracht (Bögen oberhalb)

3.3 Periodizität – Tempo-Hypothesen aufstellen

Der zweite Arbeitsschritt besteht darin, eine Tempo-Hypothese aufbauend auf den entnommenen Eigenschaften aufzustellen. Dazu wird das Signal der Onsets auf Periodizität hin untersucht. Techniken der Signalverarbeitung, die sich dazu eignen sind Autokorrelation und Kamm-Filter.
Dabei wird untersucht, wo das Höchstmaß an Übereinstimmung zwischen dem Signal und einer zeitlich verschobenen Version von sich selber erreicht ist. Der Wert der Verschiebung (Delay) bestimmt die gesuchte Periodendauer. Es scheint keinen signifikanten Unterschied zu machen, ob die Autokorrelation oder eine Kamm-Filterbank verwendet wird [Gouy04].

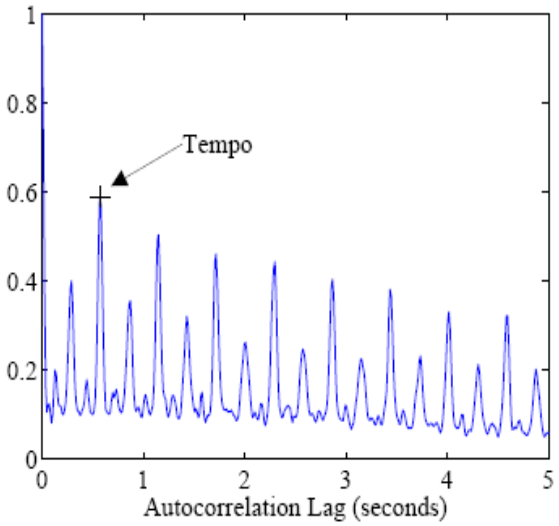


Abb.18: Autokorrelation

Die obere Abbildung zeigt die Ausgabe einer Autokorrelation. Auf der x-Achse sind die Delays eingetragen, die y-Achse zeigt die Stärke dazu. Die Wahl des *IBI-Peak*, daher des *Peak* (Spitzenwert), der die zeitliche Länge zwischen zwei Grundschlägen angibt, fällt hier auf das globale Maximum bei etwa 0.6 Sekunden und ist mit dem Kreuz gekennzeichnet. Dieser Wert stellt die Tempo-Hypothese für das Stück dar. Wie gesehen werden kann, treten Hochpunkte bei der Ausgabe der Autokorrelation auch bei Teilern und Vielfachen der eigentlichen Periode.
Eine andere weit verbreitete Technik ist das *Onset-Histogramm*. Dabei werden als erstes die Inter-Onset Intervalle ausgerechnet, indem Paare über die gesamte betrachtete Menge gebildet werden. Die IOIs werden mit der Summe ihrer Stärken in einem Histogramm abgelegt. Die Spitzenwerte stehen in einer Beziehung von Teiler und Vielfachen, wie auch bei dem Verfahren der Autokorrelation oder der Kamm-Filterung.

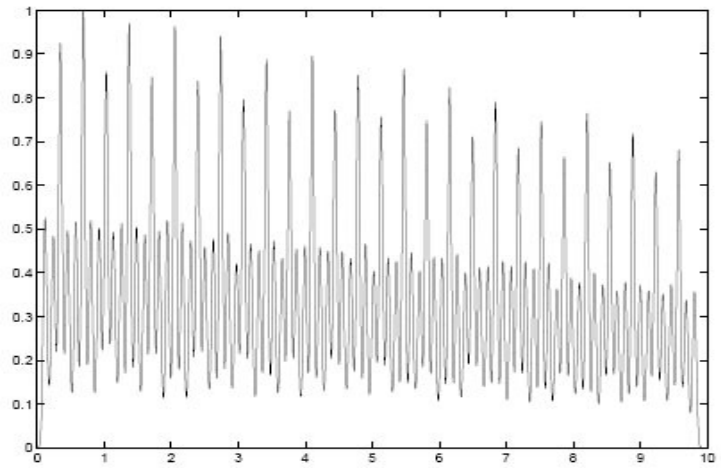


Abb.19: Ein Onset-Histogramm. Auf der x-Achse liegt die Zeit, die y-Achse die Dynamik

Das Onset-Histogramm, die Ausgabe der Kammfilterbank oder auch anderer verwandter Techniken, werden als *Rhythm-Periodicity-Function (RPF)* bezeichnet. Die Auswertung der RPF hat als Ziel die Bestimmung des Inter-Beat-Intervalls, jedoch kann daraus auch auf andere, längere oder kürzere metrische Einheiten geschlossen werden. Es muss beachtet werden, dass jede RPF nur eine Annäherung an die Realität ist. Die Verwendung von ganzen Zahlen bei den Delays der Kamm-Filterbank bedeutet beispielsweise eine Ungenauigkeit.

Nachdem die RPF ermittelt ist, werden zur Bestimmung des Inter-Beat-Intervall deren *Peaks* ausgewertet. Bei der Wahl des IBI-Peak gibt es unterschiedliche Strategien: Die einfachste besteht darin das globale Maximum in der RPF zu suchen. Eine andere Möglichkeit besteht darin, den maximalen Peak in einem bestimmten Bereich zu suchen, wobei die Auswahl dieses Bereiches anhand musikalischen Vorwissens über häufige Geschwindigkeiten geschieht. Erweiterte Ansätze beziehen in die Bewertung eines Peaks auch anderen Peaks mit ein.

Keine dieser Strategien führt jedoch mit Sicherheit zum IBI-Peak. Aus diesem Grund verwenden manche Verfahren mehrere starke Peaks als Tempo-Hypothese und lassen sie durch Multiple Agenten im Trackingteil überprüfen: Dixon [Dix06] verwendet zum Beispiel einen Algorithmus, bei dem in einem Onset-Histogramm Gruppen von IOIs gebildet werden, die sich um einen bestimmten Mittelwert bewegen. Dann wird eine Rangliste erstellt, die sich auf die Stärke der Mitglieder in der Gruppe bezieht. Dabei werden auch andere Gruppen mit einbezogen, die über einen Mittelwert verfügen, der einen Teiler oder ein Vielfaches beträgt, wobei ein gewisser Toleranzbereich eingerichtet ist. Am Ende werden mehrere Peaks gewählt und darüber Tempo-Hypothesen erstellt. Diese werden dann durch das Tracking auf ihre Qualität überprüft.

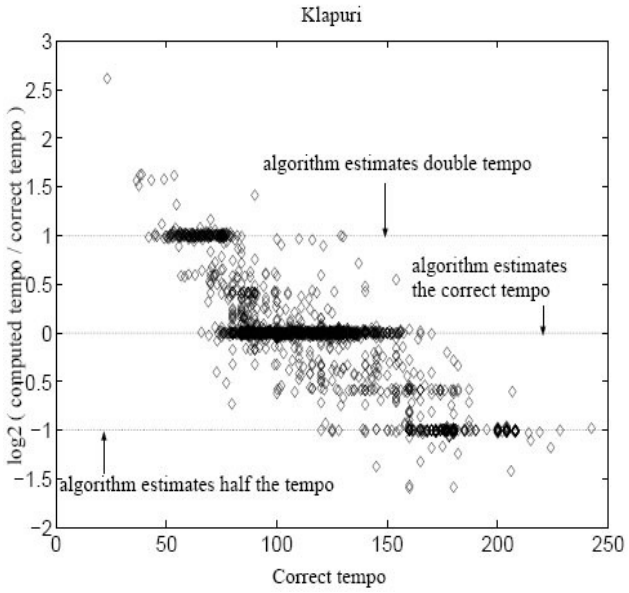


Abb.20: Halbierte und verdoppelte Geschwindigkeiten sind typische Ausgaben einer Tempo-Extraktion, hier des Verfahrens von Klapuri, dass im Vergleich sehr gut abgeschnitten hat.

3.4 Tracking - Positionen der Grundschläge

In dem ersten Schritt sind die Onsets ermittelt worden. Im zweiten Schritt sind Tempo-Hypothesen darüber aufgestellt worden, die sich als IBI ausdrücken. Im dritten Schritt werden die IBIs auf den Strom der Onsets angewendet. Die Anzahl der Grundschläge pro Minute (BPM) ist die Frequenz, dazu wird jetzt noch die Phase bestimmt. Die generelle Annahme ist, dass eine Korrelation zwischen den Grundschlägen und den Onsets im Audiostrom besteht. Bei Verwendung von mehreren Tempo-Hypothesen nebeneinander, können diese bewertet und die erfolgreichste bestimmt werden.

Gouyon und Dixon [Gouy06] sehen folgenden allgemeinen Aufbau eines Beat-Trackers: Er verfügt über einen Zustand und führt Aktionen aus, um sich dem Ziel, für den vorhandenen Audiostrom die beste Erklärung zu finden, anzunähern. Der Zustand setzt sich aus den Variablen IBI (der Tempo-Hypothese), Phase und einem Wert über die Wahrscheinlichkeit mit der die Hypothese richtig liegt, zusammen. Der Anfangszustand besteht in dem IBI und einer unbestimmten Phase und Wahrscheinlichkeit. Bei den Aktionen gibt es mehrere Strategien:

Eine mögliche Aktion ist es, ein Raster anzulegen indem die IBI-Variable mit einer Folge von ganzen Zahlen 0 bis n multipliziert wird. Dann wird überprüft, inwiefern das Raster mit den

tatsächlich vorhandenen Onsets korreliert, wobei deren Stärke mit in die Berechnung einfließt. Um das Raster flexibel zu halten kann ein Toleranzbereich um jeden Punkt eingerichtet werden. Wird bei der Korrelation ein bestimmter Grenzwert überschritten, wird dieser Zeitpunkt als Phasenbeginn gewertet.

Eine alternative Aktion dazu ist es, nur einen oder wenige Beats im voraus zu berechnen und jeweils ein Onset zu wählen, welches am nächsten zu den geschätzten Zeitpunkten liegt, wobei jedoch ein zulässiger Toleranzbereich nicht überschritten werden darf. Die Abweichungen von den berechneten Zeitpunkten können dazu verwendet werden, die IBI-Variable anzupassen. Kommen Multiple Agenten zum Einsatz, können diese Anhand der Stärke der Onsets, der Abweichungen der tatsächlichen Onsets von der Hypothese und ihrer Laufzeit bewertet werden. Eine Aktion aus dem Bereich der Mustererkennung beinhaltet Wissen über häufig auftretende Rhythmus-Muster. Diese werden mit dem Strom der Onsets verglichen. Diese Herangehensweise hat Vor- und Nachteile, da für bestimmte Genres bessere Ergebnisse erzielt werden können, dagegen andere, die aus dem Rahmen fallen schlechte Ausgaben erzeugen. Es ist hier zwischen einer Spezialisierung und einer Generalisierung abzuwägen.

Das Verfolgen der Beats in dem Strom aus Onsets ist immer mit einer Balance zwischen Stabilität und Reaktivität verbunden. Ein stabiles Verfahren passt die IBI-Variable wenig an, die Toleranzbereiche sind klein. Ein Vorteil liegt darin, dass zeitliche Phasen in denen die Onsets wenig mit dem Beat korrelieren, überbrückt werden können und später wieder angeknüpft werden kann. Der Nachteil besteht darin, schlecht auf Änderungen des Tempos zu reagieren und über eine lange Zeit hinweg falsche Ergebnisse zu liefern. Eine starke Reaktivität bedeutet sehr flexibel zu sein und einen großen Toleranzbereich einzurichten. Die IBI-Variable wird zügig angepasst. Der Vorteil liegt darin, schnell auf Änderungen zu reagieren. Der Nachteil ist, dass das Verfahren bei dieser Strategie schnell durch Onsets die nicht direkt auf dem Beat liegen, fehlgeleitet wird. Dies ist zum Beispiel durch die systematischen Abweichungen der Fall. Bei den beiden Strategien kann eine Parallele zu dem kognitiven Modell (siehe Kapitel 2.4.1) gezogen werden. Großer Stabilität ähnelt der Top-Down Strategie, große Reaktivität hingegen der Bottom-Up Strategie.

4 Anforderungen

Es ist eine Software-Komponente zu erstellen, die einen Strom von musikalischen Audiodaten in Echtzeit analysiert und die Positionen der Grundschläge darin detektiert. Das Ergebnis ist als Signal auszugeben. Die Komponente ist als ein VST-Plugin zu implementieren und in das vvvv-System zu integrieren.

4.1 Anwendungsfälle

Einsatzgebiet ist die Synchronisation eines visuellen Signals mit einem Musikstück. *Musik-Visualisierungen* werden heute vermehrt auf Veranstaltungen angewendet, um den Eindruck einer musikalischen Darbietung zu erhöhen. Visualisierungen können Videos, Animationen und Lichteffekte umfassen. Dabei geht es um die visuelle Interpretation von Gehörtem. Der Rhythmus spielt in Musik eine bedeutende Rolle, und somit wird häufig versucht beide Quellen zu synchronisieren. Diese Art der Interpretation ist die am intuitivsten vom Empfänger zu verstehende, stärker noch als eine Interpretation auf Basis des emotionalen Gehalts. Wird die Visualisierung durch eine Person gesteuert, so erfolgt die Synchronisation heute meistens über einen so genannten *Tap-Button*, mit dem der Grundschlag mitgekloft werden kann. Lichteffekte werden vorrangig über die Kopplung an ein Frequenzspektrum gesteuert. Bei dem Überschreiten eines Grenzwertes wird ein Ereignis ausgelöst. Selten werden spezialisierte Hardwarelösungen eingesetzt.

4.2 Bedienbarkeit

Die Komponente soll innerhalb des vvvv-Systems als regulärer Node verfügbar sein. Die Bedienung ist hierbei denkbar einfach: Ein Audiosignal wird oben eingeführt, das Feedback-Signal wird unten ausgegeben. Die direkte Dokumentation soll aus einem kurzen Text bestehen, der bei der Auswahl des Nodes im Menü angezeigt wird. Eine Referenz ist nach dem Standard für vvvv-Nodes zu verfassen und soll auf der Webseite verfügbar sein. Die Benutzung des Nodes innerhalb des vvvv-Systems soll keine zusätzlichen Kenntnisse voraussetzen.

4.3 Leistungsanforderungen

Die Leistungsanforderung an die Komponente ist die Verarbeitung in Echtzeit: In der laufenden Anwendung wird ein Strom an Samples übergeben und muss analysiert sein bevor das Audio ausgegeben wird, um gleichzeitig ein Feedback-Signal geben zu können. Die Beattracking-Komponente wird in den meisten Fällen im Verbund mit mehreren anderen vvvv-Nodes eingesetzt, die rechenintensive Operationen durchführen. Daher ist eine möglichst geringe Auslastung durch die Komponente wichtig. Die Ausgabe des Feedback-Signal kann sehr störend wirken, wenn sie an falschen Zeitpunkten gesetzt wird. Bei einem unsicheren Analyseergebnis soll daher kein Signal ausgegeben werden.

4.4 Implementierungsanforderungen

Die Komponente soll aus zwei Teilen bestehen: Einem VST-Plugin, das in Form einer dynamischen Programmbibliothek vorliegt. Einem vvvv-Node der das Plugin in das vvvv-System integriert wird. Das vvvv-System läuft auf der Windows-Plattform, auf Unix-basierten Systemen oder Mac ist es nicht verfügbar.

5 Analyse

Da sich die hier vorgestellte Arbeit aus mehreren Teilen zusammensetzt, müssen diese benannt werden: Unter *Beattracker-Komponente* sind beide Teile zusammengefasst. Mit *Beattracker-Node* ist der Teil bezeichnet, der in das vvvv-System integriert ist. *VST-Plugin* oder *Beattracker-Plugin* ist der vom vvvv-System unabhängige Teil, der in einem VST-Plugin realisiert ist.
Aus den Anforderungen ergibt sich ein Aufbau, der im folgenden dargestellt wird: Ein VST-Plugin hat auf der Windows-Plattform die Form einer Dynamic-Link-Library (DLL). Der Beattracker-Node ist ein regulärer vvvv-Node. Zum Laden des Plugins und der Kommunikation mit dem Plugin, ist ein VST-Host darin zu integrieren. Die Verarbeitung von Audio geschieht innerhalb des vvvv-Systems mit DirectShow. Aus diesem Grund ist der Beattracker-Knoten als ein DirectShow-Filter anzulegen, der Zugriff auf die rohen Audiodaten erhält.

5.1 Arbeitsmittel

Aus den Anforderungen ergeben sich die Arbeitsmittel die kurz besprochen werden:

5.1.1 VST

Cubase ist eine weit verbreitete professionelle Musik-Software und wird von der Firma Steinberg produziert. VST ist eine Schnittstelle, die dazu geschaffen ist fremde Komponenten zur Audioverarbeitung unter Cubase nutzen zu können, sie werden VST-Plugins genannt. Typisch sind Effekte wie Echo, Hall und Delay. Das VST-Plugin ist alleine nicht funktionstüchtig sondern benötigt einen VST-Host. Dieser liefert den Strom aus Audiosamples und empfängt ihn nach der Verarbeitung wieder. Der Host hat keine Informationen darüber, was das Plugin mit dem Audiostrom macht und kann auch nicht direkt auf die Verarbeitung Einfluss nehmen. Das Plugin kann jedoch bestimmte Funktionen bereitstellen um dem Host Eingaben zu ermöglichen. Ein VST-Plugin muss immer von der Basisklasse AudioEffectX abgeleitet sein, die in der VST-SDK definiert ist. Um bestimmte Funktionen zu unterstützen muss das Plugin nur die passenden virtuellen Funktionen dieser Basisklasse überschreiben.

5.1.2 FFTW

Bei der Verarbeitung muss das Audiosignal sehr oft in den Frequenzbereich überführt werden, dafür wird ein schnelles Verfahren benötigt. Die Bibliothek FFTW ist in C geschrieben, liegt auf der Windows-Plattform als DLL vor und berechnet die Schnelle-Fourier-Transformation. FFTW ist frei verfügbar und zeigt in Benchmarktests sehr gute Ergebnisse.

5.1.3 Visual C++ 2005 Express Edition

Visual C++ 2005 Express Edition ist eine Entwicklungsumgebung von Microsoft und ist kostenlos zu erhalten.

5.1.4 Delphi und Borland Developer Studio 2006

Delphi ist eine objektorientierte Programmiersprache, die auf Grundlage von Pascal entwickelt wurde. Mit Delphi ist es möglich auf viele Funktionalitäten der Windows-Plattform zuzugreifen, beispielsweise Winapi-Funktionen, DirectShow und COM-Objekte.

5.1.5 DirectShow

DirectShow ist eine Multimedia-Schnittstelle und Bestandteil der Microsoft-Plattform-SDK. DirectShow ermöglicht das Verarbeiten von Audio- und Video-Datenströmen einer großen Anzahl verschiedener Formate. Eine Kette für die einzelnen Schritte der Verarbeitung von Multimedia, wie Auslesen von Dateien, Splitten und Decodieren heißt bei DirectShow *Filtergraph*. Die einzelnen Glieder heißen dementsprechend *Filter*. Filter sind über Ein- und Ausgabepins miteinander verbunden. Eine Quelle ist der Start eines Filtergraphen, dieser Eingabefilter kann eine Datei oder beispielsweise ein Mikrofon sein. Die durchlaufenen Filter können verschiedenartige Operationen auf die Daten anwenden. Ausgabe kann wiederum eine Datei sein oder Lautsprecher und Monitor. Filter können in andere Filtergraphen eingefügt werden, dies macht DirectShow sehr flexibel. Die konkreten Instanzen von DirectShow-Filtern sind als *COM-Komponenten* realisiert. COM ist ein älteres Komponentenmodell von Microsoft, bei dem Clients Objekte einer COM-Komponente erzeugen. Die Clients können über einen Zeiger auf eine sprachunabhängige Schnittstelle Funktionen der Komponente nutzen, dabei erhält der Client nie den direkten Zugriff auf das Objekt. COM-Komponenten implementieren in der Regel mehrere Schnittstellen. Die Namen der

Schnittstellen beginnen mit I für Interface, z.B. IBaseFilter (ein grundlegendes Filter-Interface in DirectShow).

5.2 Theoretische Ansätze

In den Grundlagen wurde gezeigt, dass sich ein Verfahren zur Detektion von Grundschlägen in einzelne Schritte unterteilen lässt: Onset-Detection, Aufstellen einer Tempo-Hypothese und Ermittlung der konkreten Beat-. Da sich dieser Aufbau als vorteilhaft erwiesen hat und damit auch einzelne Stufen der VerarbeituPositionenng separat entwickelt und gekapselt werden können, dient er als Vorlage für die Entwicklung der Komponente. Damit bleibt zu klären wie die einzelnen Teilprobleme gelöst werden können. Im folgenden werden Alternativen aufgezeigt und es wird dargelegt, warum bestimmte Lösungswege gewählt wurden:

Bei der Onset-Detection gibt es die Möglichkeit einer direkten Analyse der Wellenform auf Steigungen oder eines framebasierten Ansatzes. Es wurde der framebasierte Ansatz gewählt, da die Vorteile gegenüber der Wellenform-Analyse überwiegen: Zusätzlich zu den von Gouyon (siehe Kapitel 3.2) dargestellten Vorteilen muss auch der erheblich geringere Rechenaufwand gesehen werden, womit auf die Leistungsanforderungen eingegangen wird. Die geringere zeitliche Auflösung kann durch kleine Blockgrößen und eventuell eine überlappende Verarbeitung ausgeglichen werden.
Wie in den Grundlagen dargestellt ist, soll eine Aufteilung des Frequenzbereichs in Segmente, die separat auf Onsets untersucht werden, Vorteile gegenüber der Verarbeitung des Spektrums als Ganzes bringen. Um dies Nachzuvollziehen wurde der Versuch von Scheirer wiederholt (siehe Kapitel 3.2). Tatsächlich ist das Signal erheblich besser zu erkennen, wenn es in mehrere Frequenzbereiche unterteilt wird. Damit stellt sich die Frage, wie die Aufteilung zu erfolgen hat. Grundsätzlich wird hier der Arbeit von Klapuri gefolgt mit dem Unterschied, dass die Segmentierung auf Basis von Frequenzgruppen (siehe Kapitel 2.2.1) geschieht. Damit wird versucht das Modell an der auditiven Wahrnehmung zu orientieren. Die Verarbeitung von 24 Subbändern bringt jedoch einen hohen Rechenaufwand mit sich. In Hinblick auf die Anforderung nach möglichst geringer Auslastung führt die Zusammenfassung zu vier Kanälen zu einem Geschwindigkeitsvorteil.

Bei der Aufstellung einer Tempo-Hypothese für das Onset-Signal bieten sich drei Techniken an: Ein Onset-Histogramm, die Autokorrelation und eine Kammfilterbank. Ein Onset-Histogramm erscheint als einfachster Ansatz, aber die große Anzahl an Peaks, wie sie zum Beispiel in Abbildung 16 Kapitel 3.3 gesehen werden kann, bedeutet dass noch weitere Verarbeitungsschritte durchlaufen werden müssen, um zu einem klaren Ergebnis kommen zu können. Daher wurde diese Möglichkeit nicht in Betracht gezogen. Wie Gouyon [Gouy04] darlegt, scheint es zwischen der Anwendung einer Autokorrelation und Kammfilterbank hinsichtlich der Qualität der Ergebnisse keine bedeutenden Unterschiede zu geben. So wurde, Klapuri und Scheirer folgend, die Kammfilterbank gewählt.
Die Auswertung der Ausgabe der Kammfilterbank wurde ohne ein konkretes Vorbild entwickelt, dazu wurden Tests mit einfachen Eingabedaten gemacht. Dies können Clicktracks sein oder kleine Drum-Tracks, wie sie zum Beispiel mit dem Tool Hydrogen erzeugt werden können. Der Vorteil ist, dass die Geschwindigkeit und der Aufbau bekannt sind. Es konnte festgestellt werden, dass an dem Wert in der Resonanz, der der Geschwindigkeit der Eingabe entspricht, ein Peak liegt. Dabei kann auch gesehen werden, dass an den Punkten, die einem Vielfachen des Abstands dieses Peak von Null entsprechen, wieder ein Peak liegt. Der Bereich des Tempo, in dem sich Musikstücke bewegen, kann eingeschränkt werden. Um diesen Bereich festzulegen wurden die BPM-Werte der Stücke aus dem Test von Gouyon[Gouy04] beachtet und eigene Tests gemacht. Die einfache Wahl des größten Peaks in dem festgelegten Bereich hat sich nicht als günstig erwiesen. Für die Bestimmung der Tempo-Hypothese werden Folgen von Peaks gebildet. Die Folgen werden nun untereinander verglichen und die stärkste stellt die Tempo-Hypothese.

Die nächste Stufe in der Verarbeitung ist, die Tempo-Hypothese auf das Onset-Signal anzuwenden und somit die konkreten Beat-Positionen zu ermitteln. Die verwendete Multiple Agenten Technik basiert grob gesehen auf einer vereinfachten Form des Verfahrens von Dixon[06]. Es bestehen jedoch große Unterschiede:
Die Verarbeitung findet in Echtzeit statt, um den gestellten Anforderungen zu genügen. Die Agenten sind in separaten Kanälen tätig, da in bestimmten Frequenzbereichen klarere Signale vorliegen als in anderen, in denen sich beispielsweise sehr viele Instrumente überlagern. Alle Agenten eines Kanals haben die gleiche Tempo-Hypothese, die über die Ausgabe der Kammfilterbank gesteuert wird. Dadurch sind Agenten leicht zu vergleichen und es müssen, dies in Hinblick auf eine geringe Auslastung, weniger parallel aktiv sein. Es wird nicht durchgängig ein Feedback-Signal gegeben, sondern versucht dies nur zu tun, wenn die Analyseergebnisse relativ eindeutig sind.

5.3 Arbeitsverfahren

Die Entwicklung der Beattracker-Komponente an sich kann unabhängig von dem vvvv-System erfolgen und ist dadurch leichter zu testen. Es ist möglich einen musikalischen Audiostrom in eine Form zu bringen, die sich gut grafisch darstellen lässt. Anhand der visuellen Repräsentation lassen sich gut periodisch auftretende Muster in dem Strom entdecken. Auf dieser Basis kann ein Verfahren entwickelt und überprüft werden. Doch dies geht nur bis zu einem bestimmten Punkt:
Ein visuelles Feedback-Signal reicht nicht aus um die Genauigkeit eines Algorithmus zu validieren, ein akustisches Signal ist dagegen sofort intuitiv erfassbar und wirkt störend bei falsch detektierten Beat-Positionen.

6 Design

Beschrieben werden die einzelnen Verarbeitungsschritte, die in der erstellten Arbeit durchlaufen werden. Im ersten Schritt wird aus dem Stereosignal ein Monosignal erzeugt. Dieses wird mit einer Fensterfunktion multipliziert und über die Schnelle-Fourier-Transformation in den Frequenzbereich überführt. Der Frequenzbereich wird nun in 24 Subbänder aufgeteilt und in jedem einzeln die Onsets ermittelt. Anschließend werden sechs angrenzende Subbänder zu einem Kanal zusammengefasst. Jeder der vier erhaltenen Kanäle wird durch eine Kammfilterbank geführt und in dem Signal die Periodizität ermittelt. Daraufhin wird die Ausgabe der Kammfilterbänke untersucht und einer der Werte als Tempo-Hypothese festgelegt. Die Tempo-Hypothese dient Multiplen Agenten dazu das Onset-Signal in einem Kanal zu parsen und die Positionen der Grundschläge festzulegen. Am Ende werden die Ergebnisse aus den vier Kanälen verglichen und das beste ausgewählt, um es als Feedbacksignal auszugeben.

6.1 Längen- und Zeiteinheiten der Ausgangsdaten

Für die Verarbeitung der Audiosignale gelten feste Längen- und Zeitwerte die durch den aufrufenden VST-Host festgelegt sind:

Samplerate	48000	
Feldlänge(Stereo)	2048	
Feldlänge(Mono)	1024	
Framerate (Samplerate / Feldlänge(Mono))	48000/1024	46,875
Zeit pro Frame (ms) (Sekunde (ms) / Framerate)	1000 / 46,875	21,33333
Länge des Onset-Signals = 1024 Frames	1024 * 21,33333s	21,84532992s

Die Samplerate liegt bei 48000 Samples pro Sekunde. Ein Sample setzt sich aus zwei 16-Bit Werten zusammen, einer für den linken Stereokanal der andere für den rechten Stereokanal. Bei einem Aufruf durch den VST-Host werden drei Parameter übergeben: Zwei gleichlange Felder mit Fließkommazahlen, das erste Feld enthält die ankommenden Audiodaten, in das andere werden die herausgehenden Audiodaten geschrieben. Der dritte Parameter gibt die Länge der Felder an, die bei 2048 liegt. Die Stereodaten liegen abwechselnd in dem Feld. Die ungeraden Indexwerte beinhalten

die Audiodaten für den linken Stereokanal, die geraden Indexwerte die Audiodaten für den rechten Stereokanal. Werden die beiden Kanäle zu einem zusammengefasst, ergibt das ein Monosignal mit der Länge 1024. Wird die Samplerate von 48000 durch die Anzahl der Monosamples pro Aufruf geteilt, ergibt sich eine Framerate von etwa 47 Frames pro Sekunde. Die Zeit pro Frame wird wie folgt berechnet: eine Sekunde geteilt durch die Framerate. Dies ergibt etwa 21 Millisekunden pro Frame. In der weiteren Verarbeitung werden Einheiten von 1024 Frames analysiert. Das ergibt einen Beobachtungszeitraum von knapp 22 Sekunden. Der zeitlich aktuellste Frame liegt auf dem Indexwert 0, der zeitlich älteste Frame liegt auf dem Indexwert 1023. Der VST-Host gibt den Takt der Verarbeitung vor und es wird immer ein neuer Frame auf den Indexwert 0 gesetzt, alle anderen werden um eins nach rechts verschoben und der älteste verworfen.

6.2 Vorverarbeitung

In der Vorverarbeitung wird aus dem Stereosignal ein Monosignal erzeugt. Dieses wird mit einer Fensterfunktion multipliziert und anschließend über die Schnelle-Fourier-Transformation in die Frequenzdomäne überführt.

6.2.1 Monosignal bilden

Der 2048 Indexwerte lange Vektor mit den Stereo-Audiodaten wird in ein Monosignal überführt. Dazu werden die zusammengehörigen, nebeneinander liegenden Werte aufaddiert und durch zwei geteilt. Das ergibt ein Feld der Länge 1024, das als Frame bezeichnet wird. Wenn nur einer der beiden Stereokanäle ausgewertet werden würde, könnte gerade der verwendet werden, der das für die Analysezwecke schlechtere Signal hat (z.B. wenn die perkussiven Instrumente auf dem ausgewerteten Kanal leise gemischt sind).

6.2.2 Fensterfunktion anwenden

Die einzelnen Frames werden nun mit dem Hamm-Fenster multipliziert. Damit werden die Leak-Effekte an den Rändern eines Blocks (siehe Kapitel 2.3.2) vermindert und die Frames für die Verarbeitung durch die Schnelle-Fourier-Transformation vorbereitet:

$$h(n) = s(n)\left[0.54 - 0.46\cos\left(\frac{2\pi n}{N}\right)\right], n = 0,\ldots, N$$

Der Vektor *s(n)* ist das Mono-Audiosignal, *n* der fortlaufende Index *0,....,N* mit *N = 1024*, *h(n)* der Ausgabevektor.

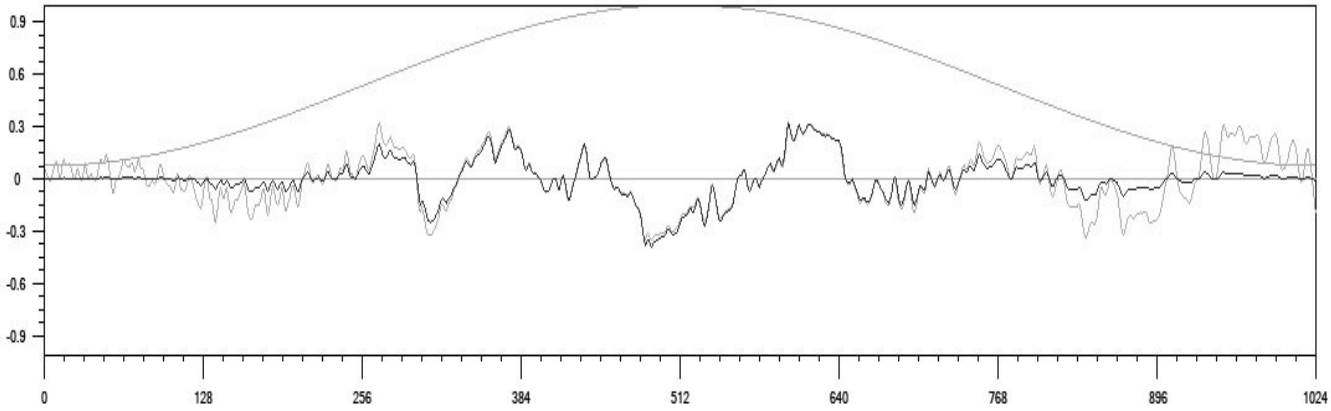


Abb.18: Die hellgraue Linie ist das Mono-Audiosignal s( ), Es wird mit der Hamm-Fensterfunktion multipliziert deren bogenartiger Verlauf in der oberen Hälfte zu sehen ist. Die Ausgabe besteht in der dunklen Linie h( ). An ihr kann die Verkleinerung der Amplituden zu den Rändern hin gesehen werden.

Um eine separate Untersuchung in verschiedenen Frequenzbereichen durchführen zu können, die zu einer Verbesserung der Analysewerte führt (siehe Kapitel 3.2), muss das Audiosignal transformiert und aufgeteilt werden.

6.2.3 Überführen in die Frequenzdomäne

Die Frames sind nun vorbereitet, um über die Schnelle-Fourier-Transformation in die Frequenzdomäne überführt zu werden. Dazu wird die Bibliothek FFTW verwendet. Die Funktion der Bibliothek bekommt einen Frame mit den Audiodaten im Zeitbereich und gibt zwei Vektoren aus. In dem einen Vektor steht der reale Teil, in dem anderen der imaginäre Teil. Die Magnituden der einzelnen Frequenzen ergeben sich durch:

$$m(n)=\sqrt{real(n)*real(n)+imag(n)*imag(n)}/N$$

Wobei N = 1024 und n = 0,..,N und m( ) die Magnituden sind. In dem Vektor *m( )* mit der Länge *1024* liegen nun auf den Indexnummern von *0,...,512* die positiven und auf den Indexnummern von *512,...,1024* die Magnituden der negativen Frequenzen vor.

6.3 Onsets ermitteln

In diesem Verarbeitungsschritt werden Akzentuierungen anhand plötzlicher, positiver Veränderungen in der Dynamik des Signals ermittelt.

6.3.1 Unterteilung in Subbänder

Der positive Frequenzbereich wird entsprechend den Frequenzgruppen (siehe Kapitel 2.2.1) in 24 Sektionen, hier Subbänder genannt,aufgeteilt. Sie durchlaufen die folgenden Verarbeitungsschritte einzeln, bis sechs von ihnen zu einem so genannten Kanal zusammengefasst werden. Die folgende Tabelle besitzt drei Spalten. Die erste Spalte gibt die Indexnummer der Subbänder *b(0,...,23)* an. In der zweiten Spalte steht der zugeordnete Frequenzbereich in Hertz. In der letzten Spalte stehen die Indexnummern auf dem Vektor m( ) mit den Magnituden, die dem Subband zugeordnet sind. Die erste Indexnummer eines Subbands b wird im folgenden mit *b0* die letzte mit *b1* bezeichnet, so ist also *b0(8) = 20* und *b1(8)=23*. Wenn in den Formeln der folgenden Verarbeitungsschritte einer Variablen der Buchstabe b angehängt wird (z.B. $r_b(n)$), dann um kenntlich zu machen, dass die Rechnung für alle Subbänder *b(0,...,24)* durchgeführt wird.

b( )	Hertz	m( )
0	0-100	0-2
1	100-200	2-4
2	200-300	4-6
3	300-400	6-8
4	400-510	8-11
5	510-630	11-14
6	630-770	14-17
7	770-920	17-20
8	920-1080	20-23
9	1080-1270	23-28
10	1270-1480	28-33
11	1480-1720	33-39

b( )	Hertz	m( )
12	1720-2000	39-46
13	2000-2320	46-53
14	2320-2700	53-62
15	2700-3150	62-72
16	3150-3700	72-85
17	3700-4400	85-102
18	4400-5300	102-122
19	5300-6400	122-148
20	6400-7700	148-178
21	7700-9500	178-220
22	9500-12000	220-278
23	12000-15500	278-512

6.3.2 Framebasierte Onset-Detection

Die Verarbeitung erfolgt framebasiert, denn eine direkte Analyse der Wellenform auf Steigungen hat hinsichtlich des Rechenaufwands und der Resistenz gegenüber Verzerrungen des Signals Nachteile (siehe Kapitel 3.2). Es wird die Folge von 1024 Frames betrachtet, was einem Zeitraum von etwa 22 Sekunden entspricht. Dieser Zeitraum ist ausreichend groß für eine Analyse auf Betonungsmuster und Periodizität, und doch noch klein genug, um auf Änderungen in der Geschwindigkeit eingehen zu können.

6.3.3 RMS

In dem ersten Verarbeitungsschritt der Onset-Detection wird für jedes der 24 Subbänder der Root-Mean-Square, also die Dynamik, für das aktuelle Frame berechnet:

$$r_b(n)=\sqrt{\frac{1}{b_1-b_0}\sum_{i=b_0}^{i=b_1} m(i)^2}$$

$r_b(n)$ ist der rms-Wert für das aktuelle Frame in dem Subband *b,* mit *b=0,..,23*. Die Subtraktion von *b1* mit *b0* ergibt die Anzahl der Werte die auf dem Vektor *m(b0,...,b1)* aufsummiert werden.

6.3.4 Kompression

Die Rms-Werte werden nun mit der µ-law Kompression (siehe Kapitel 2.3.7) verarbeitet, der Wertebereich der Onsets liegt zwischen 0 und 1. Der Wert 0 bedeutet Stille, 1 bedeutet die Obergrenze an Intensität des Audiosignals. Sie liegt für 16-Bit digitales Audio bei 96 dB. Durch die Kompression wird das Signal an die Lautstärken-Empfindung angepasst, wie es in dem Gesetz von Weber-Fechner bestimmt ist (siehe Kapitel 2.2.3) .

$$c_b(n)=\frac{\ln(1+\mu\, r_b(n))}{\ln(1+\mu)}$$

Der Vektor $c_b(n)$ mit *n=0,...,1024* ist das komprimierte Signal jeweils für die Subbänder *b(0,...,24)*. Die Variable μ *= 100*. Bei der grafischen Ausgabe lassen sich schon einzelne Bereiche mit einer hohen Intensität ausmachen.

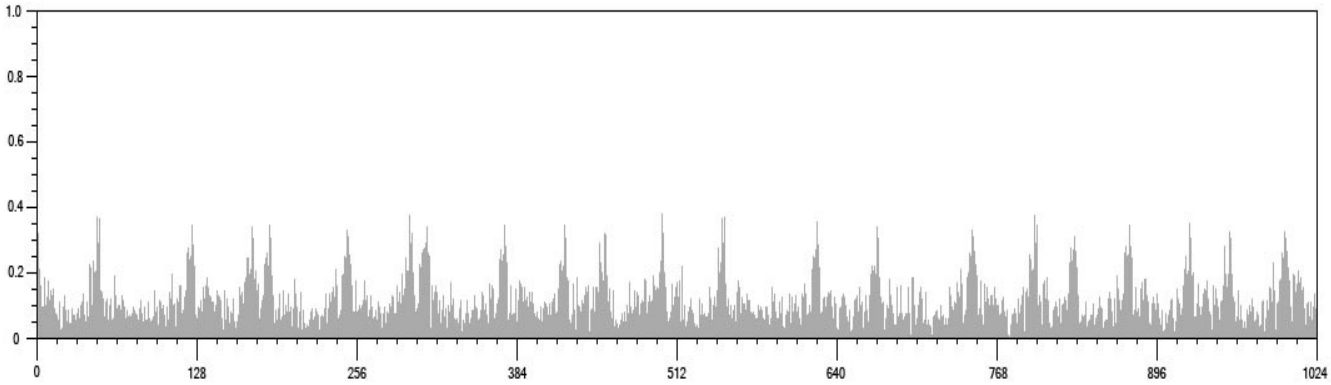


Abb.19: Das 1024-Indexwerte lange Onset-Signal nach der Kompression

6.3.5 Tiefpassfilter

Mit einem Chebyshev-Tiefpassfilter (siehe Kapitel 2.3.5) wird das komprimierte Signal nun gefiltert um es zu glätten. Die Grenzfrequenz liegt bei 10 Hz, alle Frequenzen darüber werden stark gedämpft. Betonungsmuster die einen kleineren zeitlichen Abstand als 0.1 Sekunden haben, fließen also nicht in die Berechnungen mit ein. Sie sind für eine Beatdetection auch nicht relevant und gelten als Rauschen.

$$l_b(n)=\alpha_0 c_b(n)+\alpha_1 c_b(n-1)+\alpha_2 c_b(n-2)-\beta_1 l_b(n-1)-\beta_2 l_b(n-2)$$

fc

$\alpha_0 = \ 0.06372802$
$\alpha_1 = \ 0.1274560$
$\alpha_2 = \ 0.06372802$
$\beta_1 = \ 1.194365$
$\beta_2 = -0.4492774$

Der Filter arbeitet rekursiv indem er den Ausgabevektor des Tiefpassfilters $l_b()$ mit in die Berechnungen einbezieht und erreicht dadurch eine hohe Geschwindigkeit. Der Chebyshev-Filter verwendet 3 Pole, die Filterkoeffizienten *fc* sind $\alpha_{0,}\alpha_{1,}\alpha_{2,}\beta_{0,}\beta_1$ .In dem folgendem Diagramm können nun die Betonungen als lokale Spitzen klarer erkannt werden.

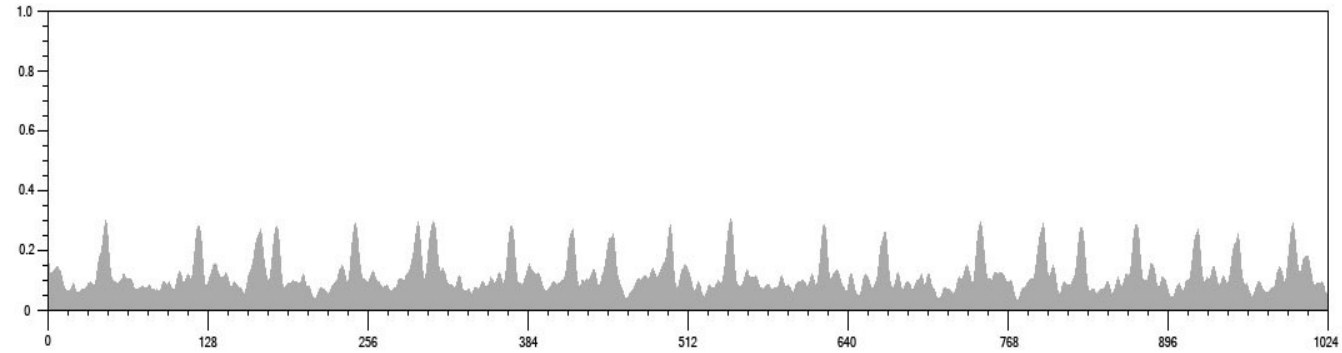


Abb.20: Onset-Signal nach der Glättung mit einem Lowpass-Filter mit Grenzfrequenz von 10Hz

6.3.6 Differenz

Als nächster Schritt wird in dem gefilterten Signal die Differenz des aktuellen Frames zu seinem Vorgänger berechnet. Die Anwendung eines Einweggleichrichters (Half-Wave-Rectifier) bedeutet einfach, dass negative Ergebnisse auf Null gesetzt werden.

$$d_b(n) = HWR\left[l_b(n) - l_b(n-1)\right]$$

Durch die Berechnung der Differenz treten abrupte Änderungen in dem Signal zu Tage, wobei durch die Einweggleichrichtung nur der Anstieg berücksichtigt wird. Die Spitzen entsprechen somit der Definition eines Onsets.

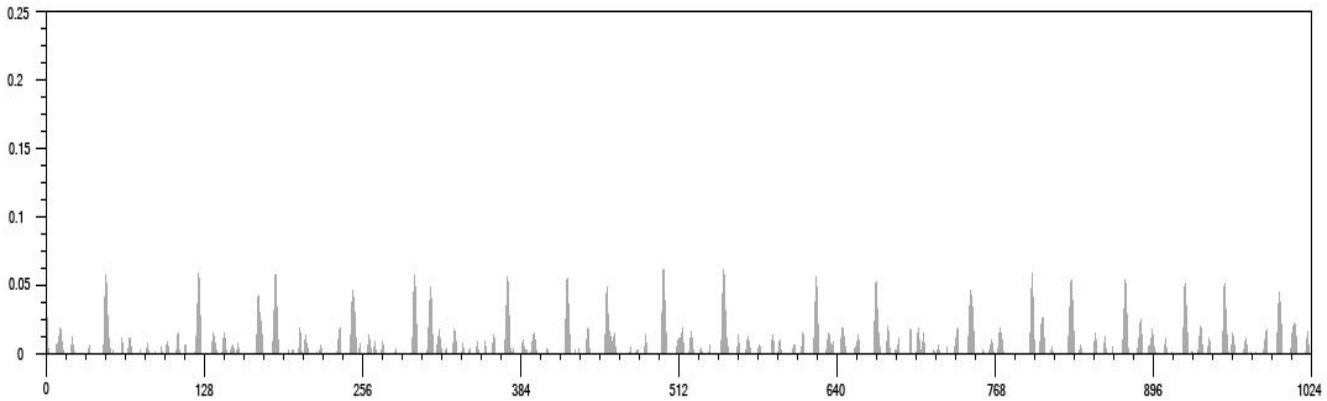


Abb.21: Onset-Signal nach der Bildung der Differenz und einem Einweggleichrichter

6.3.7 Gewichtung

Durch die Gewichtung des Tiefpass-Signals $l_b()$ mit dem Signal aus der Differenz $d_b()$ treten die Akzentuierungen noch deutlicher hervor.

$$w_b(n)=(1-\lambda)l_b(n)+\lambda\frac{f_r}{f_{LP}}d_b(n)$$

Ausgabevektor $w_b()$ ist das gewichtete Signal. Die Variable f_r bezeichnet die Framerate, daher hier etwa 86 Frames pro Sekunde, f_{LP} ist die Grenzfrequenz des Tiefpassfilters, die bei 10 Hz liegt. Spitzen treten nun klar hervor.

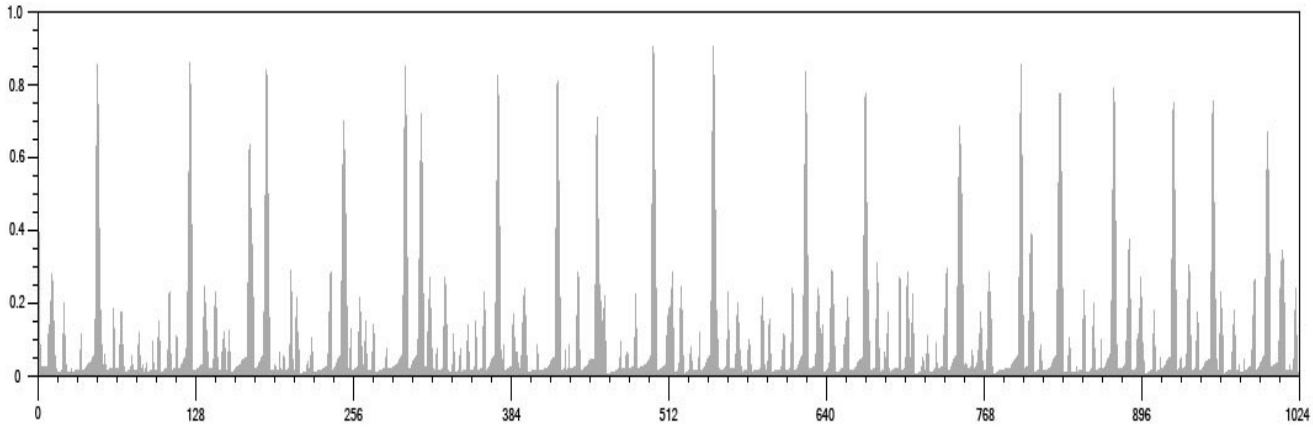


Abb.22: Onset-Signal nach der Normalisierung

Dargestellt wird das Subband 0, daher der tiefste Frequenzbereich mit Frequenzen bis 100 Hz. In diesem Bereich liegen die Bassinstrumente. Perkussive Instrumente haben besonders deutliche Onsets. Wird der Abschnitt in dem Musikstück, welcher in dem Diagramm zu sehen ist, verlangsamt abgespielt, so kann tatsächlich festgestellt werden, dass die Spitzenwerte mit einem Schlag der Bassdrum übereinstimmen. Bei einer Auswahl der oben gefundenen Spitzenwerte von Hand ergibt sich folgendes Muster:

Abb.23: Peaks mit der Hand ausgewählt. Die Punkte entsprechen dem Einsatz der Bassdrum in dem Testsignal.

6.3.8 Kanäle bilden

Die Analyse in den einzelnen Subbändern ist nun abgeschlossen und jeweils sechs von ihnen werden zu einem Kanal zusammengefasst. Dies ergibt vier Kanäle in denen die folgenden Verarbeitungsschritte einzeln durchgeführt werden.

$$v_c(n)=\left[\sum_{b=c*6}^{(c+1)*6} w_b(n)\right], c=0,\ldots,4$$

Mit c werden die vier Kanäle bezeichnet, für jeden Kanal *c* gibt es einen Vektor $v_c()$ mit dem Onset-Signal. In $w_b()$ steht das gewichtete Signal für ein Subband *b*. Auf $w_b()$ wird an der Stelle mit der Indexnummer *n* über die Subbänder *b = c*6* bis *b=(c+1)*6* die Summe gebildet. Für Kanal *c=2* wäre das z.B. *b=12,...,18*.

6.4 Tempo-Hypothese aufstellen

Nachdem in den vier Kanälen ein Onset-Signal über 1024 Samples mit einer zeitlichen Länge von etwa 22 Sekunden bestimmt wurde, wird nun eine Tempo-Hypothese darüber aufgestellt. Dazu wird eine Kammfilterbank verwendet die periodisch auftretende Eigenschaften im Bereich von Null bis vier Sekunden detektiert.

6.4.1 Niedrige Werte verwerfen

In folgendem Verarbeitungsschritt wird das Signal in einem Kanal darauf vorbereitet, um durch einen Kammfilter gefiltert zu werden. Da auf eine Regelmäßigkeit von Betonungen untersucht wird, erhöht es die Qualität der Ausgabe eines Kammfilters, wenn unbetonte Sektionen abgeschnitten werden. Dazu wird der Mittelwert über die 1024 Samples gebildet und mit einem Faktor von 1,4 multipliziert, Samples deren Wert unter dem Ergebnis liegen gelten als unbetont und werden auf Null gesetzt.

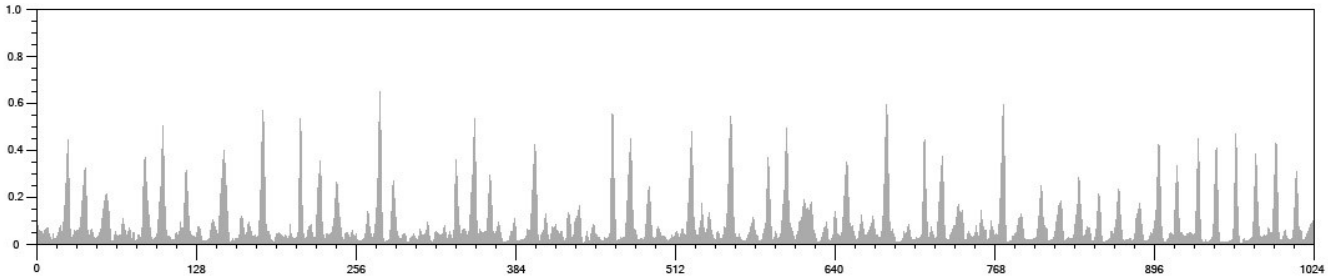


Abb.24: 6 angrenzende Subbänder sind zu einem Kanal zusammengeführt worden. Dies ist Kanal c=0 für die tiefsten Frequenzen.

Der Grenzwert wird als hellgraue, horizontale Linie dargestellt. Die Samples deren Wert darüber liegen sind dunkel eingefärbt. Einzelne Samples liegen unterhalb des Grenzwertes und gelten

trotzdem als betont. Dies liegt daran, dass der Grenzwert fortlaufend bei jedem neuen Frame (Index 0) berechnet wird, der hier angezeigte Grenzwert also nicht mit dem übereinstimmen muss der vorher (Index 1-1024) die Werte gefiltert hat.

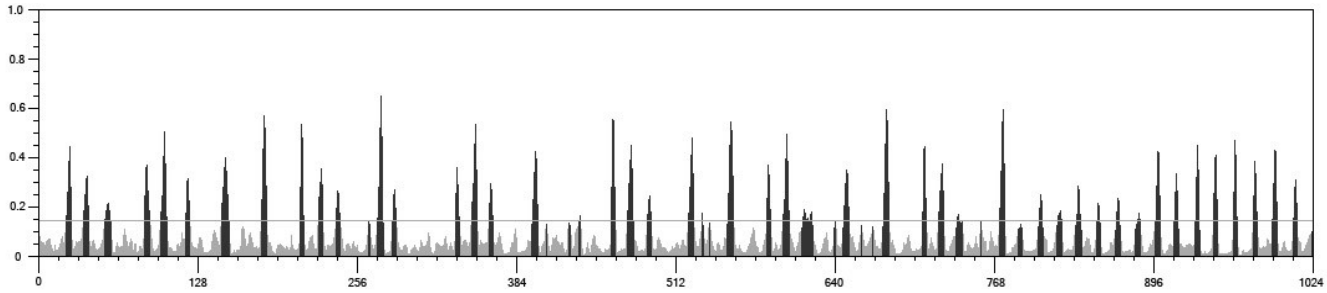


Abb.25: Die Onsets in einem Kanal werden von „Rauschen“ getrennt.

6.4.2 Die Kammfilterbank

Mit einer Kammfilterbank, kann ein Signal auf Periodizität hin untersucht werden. Die Ausgabe der Kammfilterbank weist eine *Resonanz* bei den Delay-Werten auf, die der Periodenlänge des Signals entsprechen. Die Ausgabe errechnet sich nach:

$$r_c(\tau, n) = r_c(\tau, n-\tau) + v_c(n)$$

Der Eingabevektor sind die Onset-Signale aus den vier Kanälen *vc* mit *c=0..3*. Der Ausgabevektor für die Kanäle hier Resonanz genannt, steht in *rc* wieder mit *c=0..3*. Es werden alle Indexwerte auf dem Onset-Signalen durchlaufen mit *n=0..1023*. Der Wert τ ist der Delay, er geht von $\tau = 0..188$, das sind etwa 4 Sekunden. Das heißt mit der Kammfilterbank wird auf periodisch auftretende Eigenschaften überprüft, deren Periodendauer zwischen 0 und 4 Sekunden lang ist.

Ein Beispiel für die Kammfilterung bei dem das Eingangssignal aus einem einzigen Impuls besteht. In Abbildung 26 wird ein Click gezeigt. In der folgenden Abbildung sieht man die Ausgabe bei einem Delaywert von 26. Da alle nachfolgenden Eingaben gleich Null sind, kann gesehen werden,wie ein einzelner Ausschlag, ähnlich einem Echo, eine nach und nach abfallende Version von sich hinter sich herzieht.

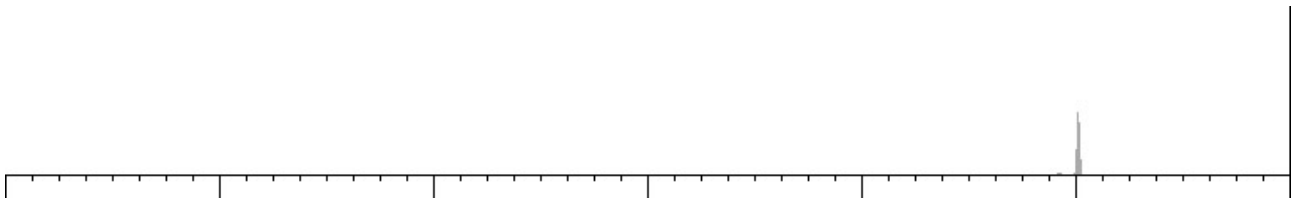


Abb.26: Ein Click

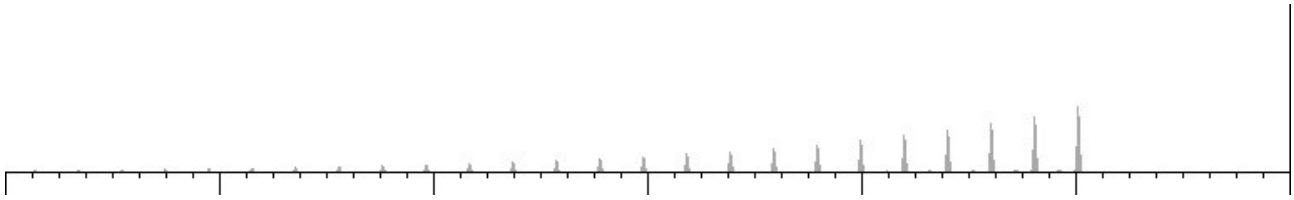


Abb.27: „Echo“ des Clicks

Folgt ein weiterer Click in einem Abstand eines vielfachen von 26, so addiert er sich zu dem Echo, genauso wie sein eigenes Echo. Die Amplitudenwerte steigen, das Signal zeigt eine erhöhte Resonanz. Auf diese Weise ist es möglich die Abstände von regelmäßigen Onsets zu ermitteln.

6.4.3 Resonanz auswerten und die Tempo-Hypothese bestimmen

In der Resonanz kann abgelesen werden, wie stark periodische Eigenschaften in dem betrachteten Zeitraum von 22 Sekunden sind (siehe Längen- und Zeiteinheiten 6.1), die ein Intervall von Null bis vier Sekunden haben. Ein Grundschlag ist eine periodisch auftretende Eigenschaft, und bewirkt einen hohen Wert in der Resonanz bei dem Delaywert der seinem Intervall entspricht. In den nächsten Schritten wird dieses Intervall in der Resonanz gesucht und damit die Tempo-Hypothese bestimmt.

6.4.4 Peakpicker

Das Peakpicker-Verfahren dient dazu Spitzen in einem Vektor zu detektieren, es wird für die Auswertung der Resonanzen und der Onsets verwendet. Das Verfahren bildet auf einem Vektor einen Durchschnittswert und setzt alle Indexnummern deren Wert darunter liegt auf Null. Damit bilden sich kleine Ballungen von überdurchschnittlich hohen Werten. Das Verfahren läuft rekursiv für jedes dieser Ansammlungen weiter, bis eine bestimmte Mindestanzahl an Stellen in einer Ballung unterschritten wird, dann wird darin der größte Wert als Peak identifiziert.

6.5 Positionen der Grundschläge

Nachdem für jeden Kanal eine Tempo-Hypothese aufgestellt wurde, wird die Technik der Multiplen Agenten angewendet, um die konkreten zeitlichen Positionen der Grundschläge zu bestimmen. Agenten bekommen mit der Tempo-Hypothese im Prinzip eine Grundschlag-Frequenz zugewiesen und haben dafür jeweils eine unterschiedliche *Phasen-Hypothese*. Während ihrer Aktivität überprüfen sie die Phasen-Hypothese und diejenige des stärksten Agenten gibt die Positionen der Grundschläge vor.

6.5.1 Agency

Eine Agency verwaltet die Agenten, die in einem Kanal tätig sind. Ein Agent hat den Zustand aktiv oder nicht aktiv. Je weniger Agenten gleichzeitig aktiv sind, um so weniger Auslastung. Die maximale Anzahl der Agenten liegt bei 16. Diese Zahl hat sich als groß genug erwiesen, um mit ziemlicher Sicherheit alle Onsets erfassen zu können. Tatsächlich sind selten mehr als 8 Agenten parallel aktiv. Die Ausgangsbasis der Agency ist der 1024 Indexwerte umfassende Onset-Vektor ihres zugeordneten Kanals und die Tempo-Hypothese die zwischen 16 und 64 liegt.
Der erste Arbeitsschritt besteht darin, in dem Onset-Vektor die Peaks zu bestimmen und alle anderen Werte auf Null zu setzen. Dazu wird das oben beschriebene Peakpicking-Verfahren verwendet. Auf dem erhaltenen Vektor wird überprüft, ob an der Stelle $n=64$ ein Peak vorhanden ist. Bei Beginn der Verarbeitung, wenn die Summe der Onset-Energie in dem Onset-Vektor noch gering ist, werden viele Onsets von dem Peakpicker-Verfahren als Peak bestimmt deren Wert eigentlich niedrig ist und die unrelevant sind. Um dies zu vermeiden, wurde mit $n=64$ eine etwas verzögerte Stelle ausgewählt. Ist ein Peak auf $n=64$ enthalten, so wird ein Agent gestartet und der Zeitpunkt, der durch diesen Indexwert beschrieben ist, gilt als dessen Phasen-Hypothese.

6.5.2 Aktionen der Agenten

Ein Agent überprüft den Strom aus Onsetwerten auf die Stärke einer bestimmten Phase hin. Seine Aktion besteht darin Onsets einzusammeln, die dieser Phase entsprechen. Über die Stärke der eingesammelten Onsets bestimmt sich der Wert des Agenten. Werden die Werte der gesamten Agenten, die in einem Kanal tätig sind betrachtet, kann auf die tatsächliche Phase des Grundschlags geschlossen werden.

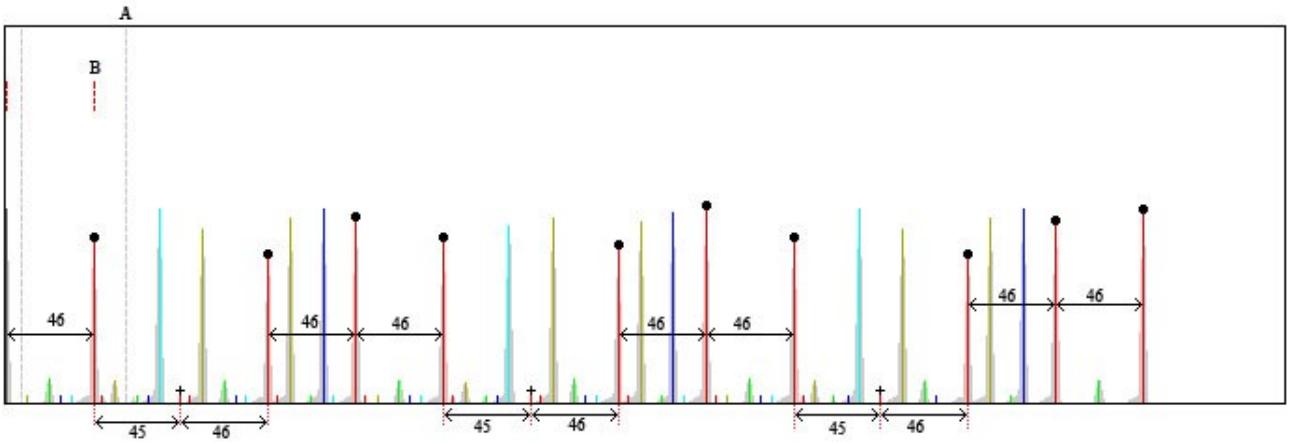


Abb.28: Das Onset-Signal ist hellgrau. Die Peaks sind bunt, jede Farbe gehört zu einem Agenten. Die kleinen bunten Striche am Boden markieren Stützpunkte die nicht valide sind.

Vorgehen im einzelnen: Das Sample an der Stelle *n=64* auf dem Onset-Signal wird als Startpunkt des Agenten gesetzt, es ist in der oberen Abbildung mit *A* markiert. Die Indexnummern auf dem Vektor, die zu einem Agenten gehören, werden im folgenden als *Stützpunkte* bezeichnet. Ausgangbasis ist immer der letzte Stützpunkt. Der Takt der Verarbeitung wird durch die hereinkommenden neuen Samples vorgegeben, wobei sich der gesamte Vektor immer um eins nach rechts verschiebt, und das neuste Sample an die Stelle 0 gesetzt wird. Von dem letzten Stützpunkt ausgehend wird die Indexnummer, die um die Tempo-Hypothese vor (oder zeitlich nach) ihm liegt daraufhin überprüft, ob sie einen Peak enthält. Auf dem oben dargestellten Vektor läuft die Suche nach einem neuen Stützpunkt immer von rechts nach links. Ist dies der Fall, so wird der neue Stützpunkt in der Liste des Agenten eingetragen, falls nicht, wird in der Umgebung von +/- zwei Samples um diesen Punkt nach einem Peak gesucht und ebenfalls als neuer Stützpunkt gewählt, falls er einen Peak enthält. Die Positionen der Stützpunkte werden bei jedem Verarbeitungsschritt um eins nach rechts verschoben, so dass diejenigen, die weiter als 1023 Samples, daher etwa 22 Sekunden von dem aktuellsten Sample entfernt sind, gelöscht werden. Es wird zwischen so genannten *validen* und *nicht validen* Stützpunkten unterschieden, auf validen Punkten liegt ein Peak. Als nicht valide Stützpunkte gelten Indexnummern die weder selbst noch im Toleranzbereich einen Peak enthalten. Sie werden trotzdem gesetzt, da damit kurze Zeiten überbrückt werden können, in denen das Musikstück keine oder keine passenden Beats enthält. Die Abweichungen von den Entfernungen, welche durch die Tempo-Hypothese gesetzt werden, werden nicht notiert, da sich eine Veränderung der Tempo-Hypothese nur über die Analyse der Resonanzen gesteuert wird. Der Wert eines Agenten ergibt sich über die Onsetstärken die er eingesammelt hat. Jeder Agent besitzt einen kurzfristigen Wert, welcher sich über die Summe der Peaks bildet, die auf dem Onset-Vektor liegen, und einen längerfristigen Wert.

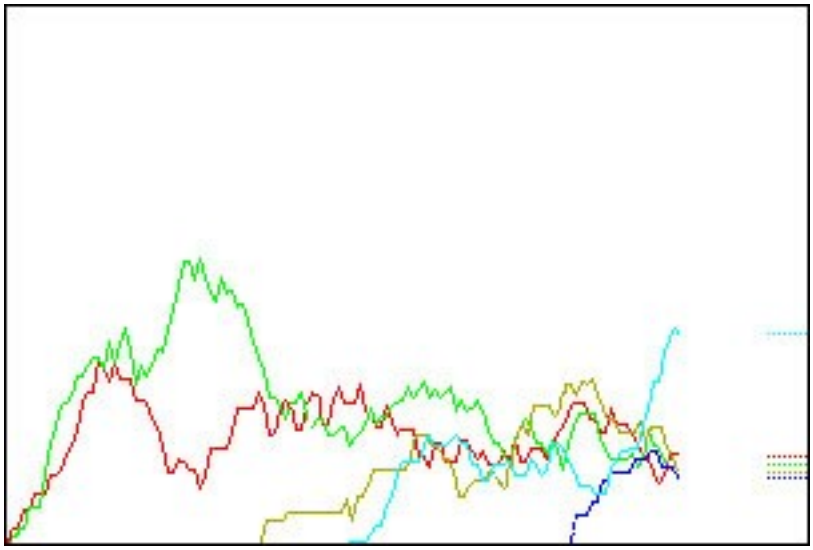


Abb.29: Stärkeverlauf von Agenten. Die x-Achse ist die Zeit, auf der y-Achse ist die Stärke eingetragen. Im Lauf der Zeit sind neue Agenten hinzugekommen z.B. Agent-Türkis

Wenn bei einem Agenten der letzte valide Stützpunkt vergeht, also über das 1023'te Sample hinweg geschoben wird, wird der Agent beendet. Um zu bestimmen, ob in dem Moment ein Grundschlag stattfindet oder nicht bildet der Agent den Mittelwert über den Intervallen zwischen seinen Stützpunkten. Dann wird überprüft, ob zwischen dem letzten validen Stützpunkt und dem Indexwert Null genau dieser Abstand oder ein Vielfaches davon vorliegt. Ist dies der Fall, so setzt er diesen Zeitpunkt als Beat-Position fest. Falls der letzte valide Stützpunkt zu viele Intervalllängen von der Nullstelle entfernt ist, wird durch den Agenten kein Beat-Signal gegeben, da sich die Ungenauigkeiten schnell aufaddieren.

6.5.3 Verwaltung der Agenten durch die Agency

Eine Agency startet, beendet und bewertet Agenten. Sie besitzt einen Vektor, in den die Agenten ihre Stützpunkte eintragen. Rückt ein Peak auf dem Onset-Vektor auf die Stelle *n=64* vor, wird überprüft ob dieser Peak schon durch einen aktiven Agenten eingesammelt wurde. Ein neuer Agent wird nur gestartet, falls dies nicht der Fall ist.Um nicht zu viele Agenten gleichzeitig aktiv werden zu lassen, wird pro Takt überprüft, ob sich zwei Agenten irgendwo auf dem Feld überschneiden. Dies kann geschehen, wenn sich die Phasen der beiden über die Verschiebung, welche bei wiederholter Ausnutzung des Toleranzbereichs, eintritt angleichen. Da beide jeweils ab diesem Schnittpunkt die gleichen Punkte setzen werden, macht es keinen Sinn, sie parallel weiterlaufen zu lassen. Aus diesem Grund findet ein Vergleich auf Basis der kurzfristigen Stärke statt und der

Schwächere wird auf unaktiv gesetzt. Am Ende eines Taktes wird über einen einfachen Vergleich der längerfristigen Energie über die Menge der aktiven Agenten der Top-Agent bestimmt. Der Top-Agent gibt das Feedback-Signal für diesen Kanal vor. Zu einem unschönen Höreindruck kommt es wenn sich zwei Agenten abwechseln und sich der eine zum Beispiel auf dem Offbeat, der andere auf dem Beat befindet. Darum wir das Signal für die Wiedergabe des Feedbacks für 1,5 Sekunden ausgesetzt. Die Phasenverschiebung wirkt so nicht mehr störend.

6.5.4 Auswahl zwischen den Kanälen

Die Agencies, bei denen jeweils eine für einen Kanal zuständig ist, haben ebenso wie die Agenten einen Wert. Dieser Wert berechnet sich über die Anzahl der Takte, bei denen kein Wechsel des Top-Agenten stattgefunden hat. Die Anzahl dieser Takte wird mit er Tempo-Hypothese multipliziert, da sonst Kanäle die eine kurze Intervalllänge verarbeiten Übergewicht hätten. Bei der Zusammenführung der Ergebnisse aus den Kanälen und der Wahl der Agency, die das Feedback-Signal liefert, wird die Agency mit dem höchsten Wert bevorzugt, denn ist ein Agent über einen langen Zeitraum Top-Agent, ist dies ein Indiz dafür, dass seine Phasen-Hypothese richtig ist.

7 Implementierung

7.1 Modellierung

Durch die Implementierungsanforderungen und die in der Analyse dargestellten Verarbeitungsschritte ergibt sich ein natürlicher Aufbau der Komponente, der durch weitere Modellierung noch verfeinert werden kann. Die erste Trennlinie kann zwischen VST-Plugin und dem darauf zugreifenden vvvv-Node gezogen werden. Hier verläuft auch die Grenze der Codierung mit C++ und Delphi.

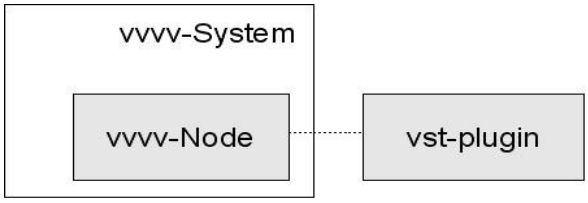


Abb.21: Die Komponente setzt sich vvvv-Node und VST-Plugin zusammen

Das Aufgabengebiet des vvvv-Node gliedert sich weiter auf in: Integration in das vvvv-System, daher der Node-Funktionalität. Kommunikation mit dem Plugin, also dem VST-Host.
Die Node-Funktionalität bedeutet die Anmeldung des Nodes in dem vvvv-System und die Möglichkeit der Integration darin, über Verbindungen seiner Pins mit anderen Nodes. Weiterhin die Entgegennahme von Audiosamples, deren Übergabe an den VST-Host und Ausgabe der Analyseergebnisse. Der Aufgabenbereich des VST-Hosts besteht darin, das Plugin aufzurufen, die Audiosamples zu übermitteln und das Ergebnis der Auswertung anzunehmen. Der Vorteil dieser Art der Aufteilung ist, dass der VST-Host leicht für die Entwicklung eines weiteren vvvv-Nodes dienen kann, der die Einbindung beliebiger VST-Plugins wie Filter oder Synthesizer in das vvvv-System ermöglicht.

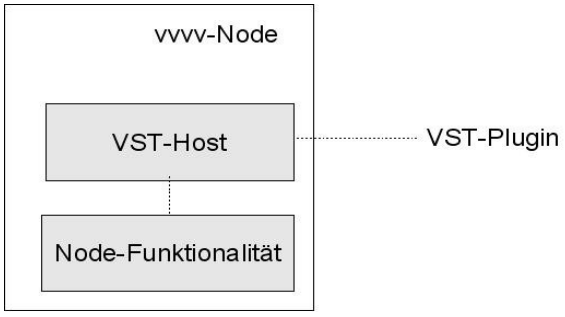


Abb.22: Unterteilung des vvvv-Nodes

Der Entwurf des VST-Plugin orientiert sich an Verarbeitungsketten, wie sie für multimediale Inhalte verwendet werden. Ein Beispiel dafür ist ein Filtergraph von DirectShow. Das Ideal des Entwurfs besteht darin, für jeden Verarbeitungsschritt ein gekapseltes Objekt vorliegen zu haben, dass genau einen Eingabewert und einen Ausgabewert hat.
Auch die Aufgaben des VST-Plugins können in zwei Bereiche unterteilt werden: Zum einen die eigentliche VST-Funktionalität, daher die Kommunikation mit dem Host. Zum anderen die Analyse der Audiodaten und die Generierung eines Feedback-Signals.

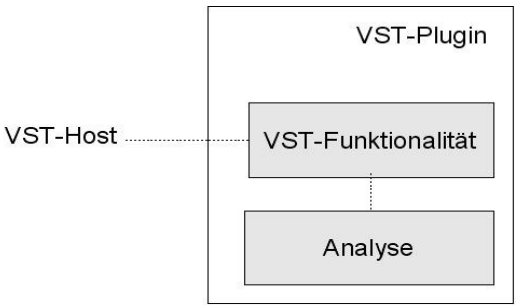


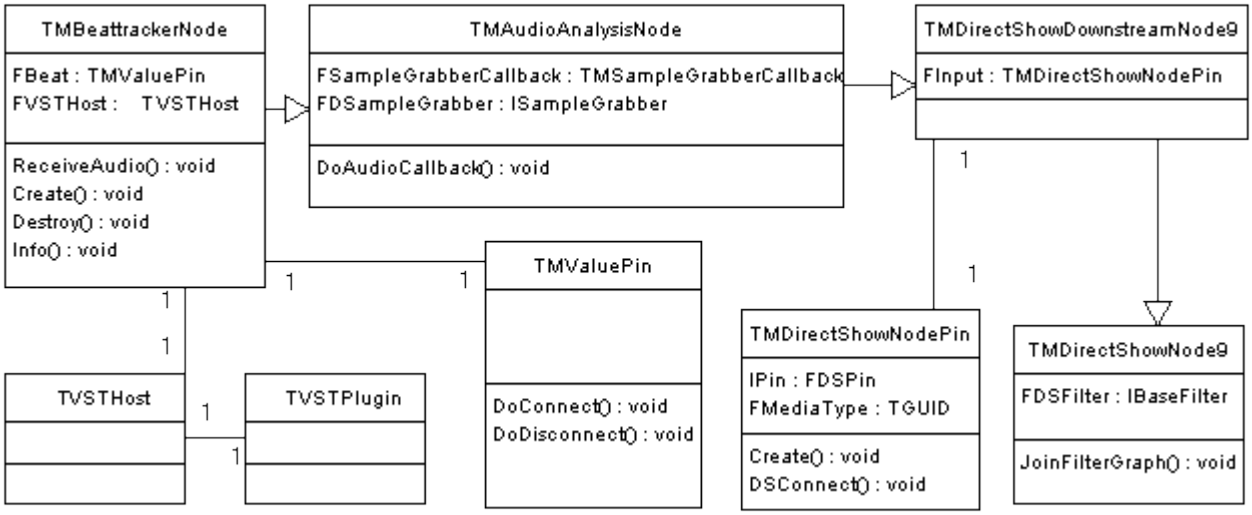
Abb.23: Unterteilung des VST-Plugins

7.2 Beattracker-Node

7.2.1 TMBeattrackerNode

Der Beattracker-Node besteht aus einem Objekt der Klasse TMBeattrackerNode. Die Klasse ist von TMAudioAnalysisNode, die schon im vvvv-System vorhanden ist, abgeleitet und erhält über die Vererbungshierarchie die Funktionalität, als regulärer Node innerhalb des Systems verwendbar zu sein. Die dargestellt Vererbungshierarchie beschränkt sich auf den Teil, der für die Audioverarbeitung relevant ist (von unten nach oben):
TMDirectShowNode kann über das Interface IBaseFilter in einen DirectShow-Filtergraphen integriert werden. TMDirectShowDownstreamNode9 besitzt mit FInput ein Objekt der Klasse TMDirectShowNodePin, das über das Interface IPin Datenströme verschiedener Medientypen einer DirectShow-Quelle annehmen kann. Mit TMAudioAnalysisNode erfolgt eine Spezialisierung auf den Medientyp Audio und über das Interface ISampleGrabber kann auf die rohen Audiosamples zugegriffen werden, die in dem Filtergraphen fließen.
Weiterhin erbt TMBeattrackerNode unter anderem die Fähigkeit sich am System anzumelden, damit hat der User die Möglichkeit einen Beattracker-Node über ein Auswahlmenü in die

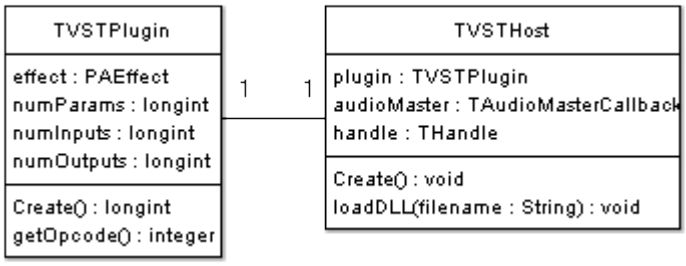
Anwendung einzufügen. Diverse Ein- und Ausgabepins können angelegt werden.



TMBeattrackerNode hat einen Input-Pin FInput geerbt, über den Audiodaten angenommen werden können. Der Output-Pin FBeat ist vom Typ TMValuePin und kann die booleschen Werte 0 für *true* und 1 für *false* ausgeben. Er gibt an, ob in dem aktuellen Zeitpunkt ein Beat stattfindet oder nicht. Das Attribut FVSTHost ist vom Typ TVSTHost und darüber kann Verbindung mit dem VST-Plugin aufgenommen werden.

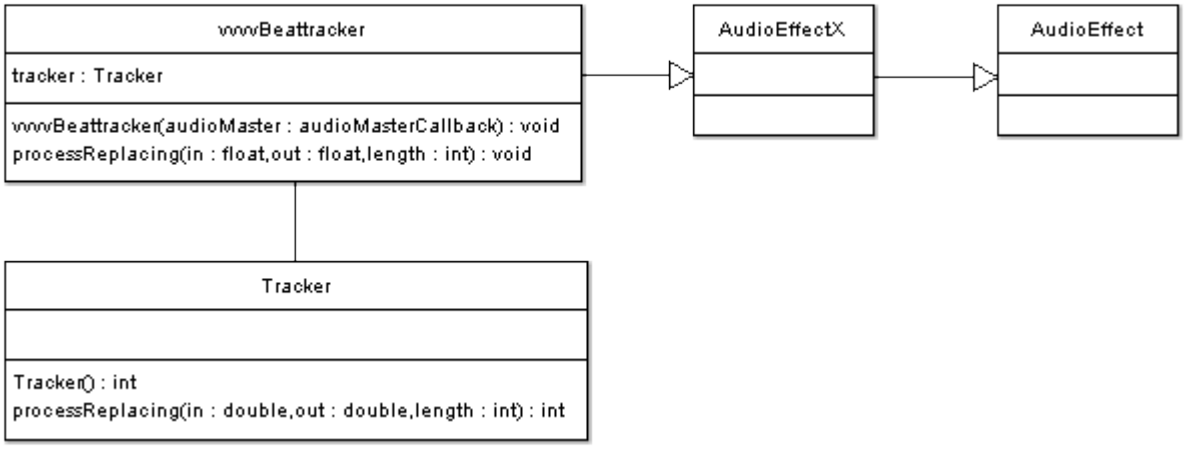
7.2.2 TVSTHost und TVSTPlugin

Die Klasse TVSTHost implementiert die VST-Host Funktionalität: Ein VST-Plugin in Form einer DLL kann aufgerufen und ihm Audiodaten gesendet werden. Über eine Callback-Funktion (über einen Zeiger kann diese Funktion aufgerufen werden) kann der Host Daten vom Plugin empfangen. Bei dem Laden wird das Objekt *plugin* der Klasse TVSTPlugin initialisiert, dass als Schnittstelle zwischen dem VST-Host und dem VST-Plugin fungiert. Der Aufruf des VST-Plugins läuft in der Funktion loadDLL() ab und wird weiter unten besprochen.



7.3 VST-Plugin

Die Klasse vvvvBeattracker implementiert die Beattracker-Funktionalität in einem VST-Plugin. Ein VST-Plugin ist immer von der der Klasse AudioEffectX abgeleitet, die wiederum von AudioEffect abgeleitet ist. Minimale Anforderungen an ein VST-Plugin: Implementation der Callback-Funktion createEffectInstance(), in der das Plugin-Objekt erzeugt wird, sowie processReplacing(), in der die Audioverarbeitung stattfindet.

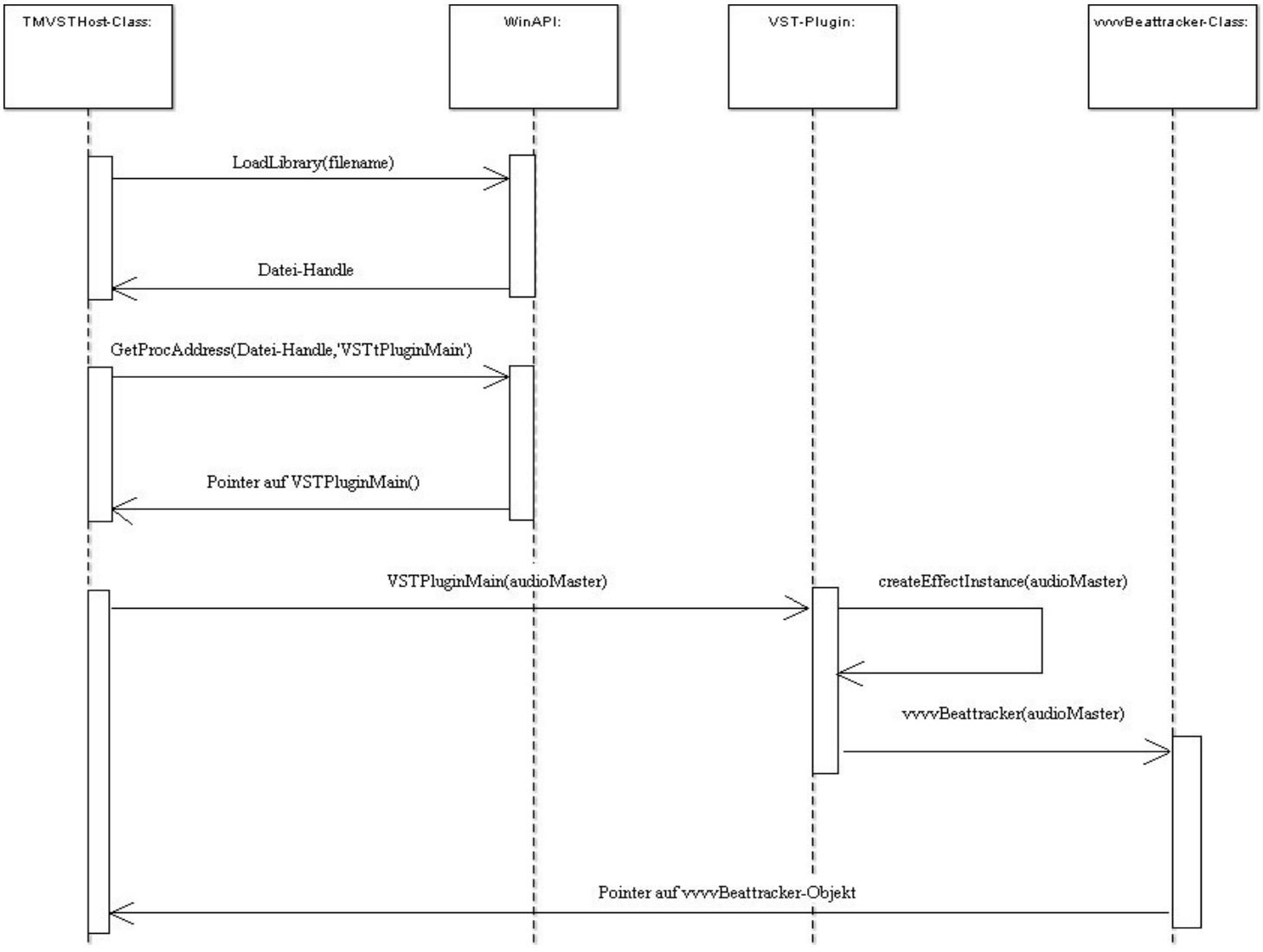


Die Klasse vvvvBeattracker besitzt zwei Funktionen: Der Konstruktor vvvvBeattracker() bekommt mit *audioMaster* einen Zeiger auf die Callback-Funktion des VST-Hosts. Über den Aufruf dieser Funktion können Nachrichten zum Host geschickt werden. Die zweite Funktion ist processReplacing(), die drei Parameter übergeben bekommt. Der erste Parameter ist ein doppelter Zeiger auf ein Feld mit den Audiosamples der Eingabe. Der zweite Parameter ist ebenfalls ein doppelter Zeiger, in den durch ihn bezeichneten Speicherbereich werden die Audiosamples der Ausgabe geschrieben. Der dritte Parameter gibt die Länge der Felder an.
Die Analyse der Audiosamples ist der Klasse Tracker ausgelagert, von der vvvvBeattracker eine Instanz besitzt. Somit fungiert vvvvBeatttracker im Prinzip als Schnittstelle zwischen dem VST-Host und der Verarbeitung in dem Tracker-Objekt.

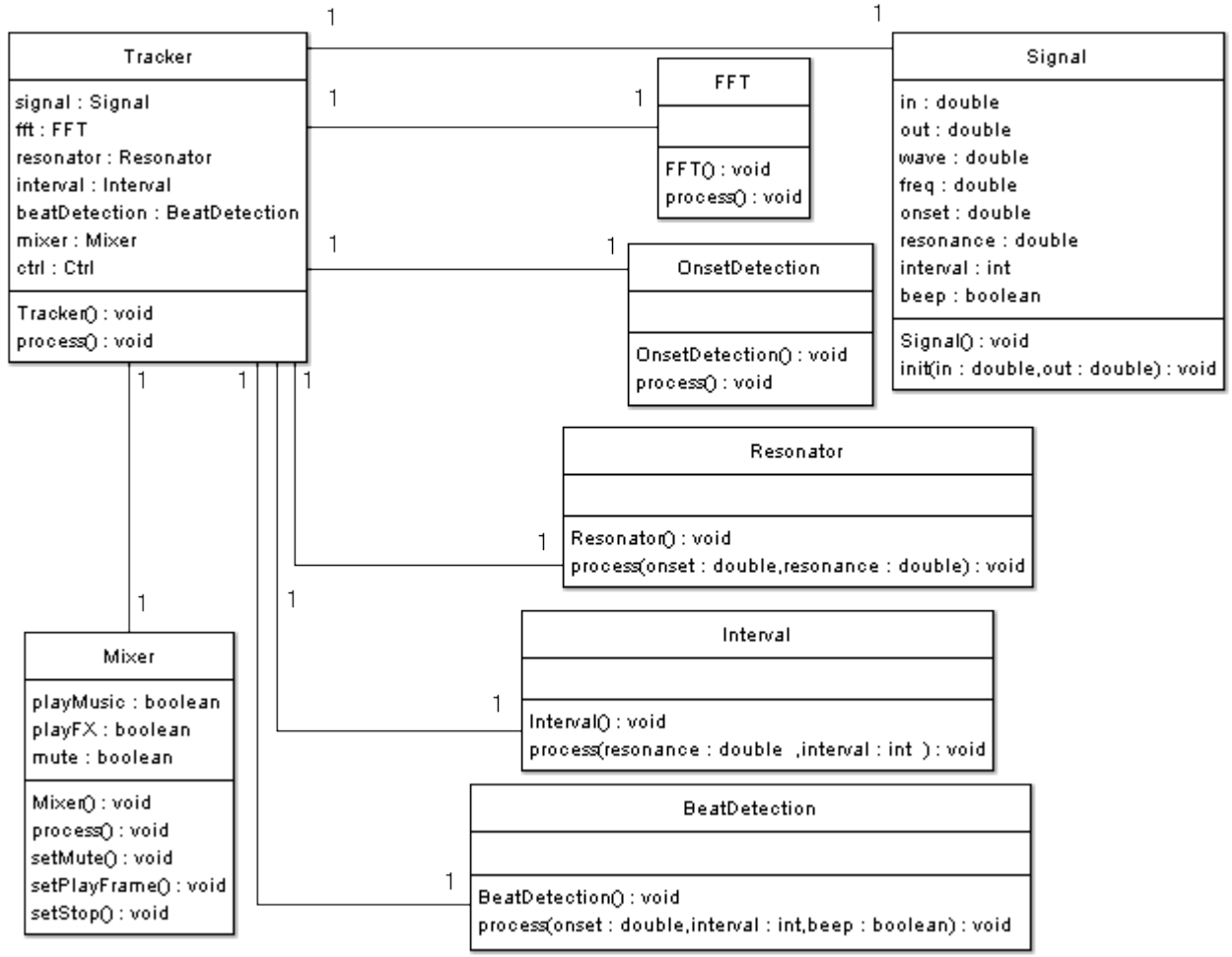
7.4 Initialisierungsphase von Host und Plugin

Zum Laden des Plugins aus dem Beattracker-Node im vvvv-System wird die Funktion loadDLL() der Klasse TVSTHost aufgerufen, wobei der String „vvvvBeattracker.dll“ als Parameter *filename*

übergeben wird. Mit *filename* wird nun die Winapi-Funktion LoadLibrary() aufgerufen, die ein Handle (hier ein Zeiger auf einen Speicherbereich) auf die DLL zurück gibt. Danach folgt der Aufruf der Winapi-Funktion GetProcAddress() mit dem Handle als Parameter und dem String 'VSTPluginMain', was der standardmäßige Name der DLL-Hauptfunktion eines VST-Plugins ist. Rückgabewert von GetProcAddress() ist ein Zeiger auf die Funktion VSTPluginMain() in dem Plugin. Über diesen Zeiger wird nun die Funktion aufgerufen und als Parameter der Zeiger *audioMaster* auf die Callback-Funktion *audioMasterCallback* übergeben. In VSTPluginMain() wird die Funktion createEffectInstance() aufgerufen, in der das vvvvBeattracker-Objekt initialisiert wird. Bei dem Aufruf des Konstruktors wird der Callback-Zeiger *audioMaster* übergeben, womit das Plugin jetzt die Möglichkeit besitzt Nachrichten an den Host zu senden. Rückgabewert von createEffectInstance() und somit auch VSTPluginMain() ist der Zeiger auf das neu angelegte vvvvBeattracker-Objekt, der in TVSTPlugin festgehalten wird.



Der Informationsaustausch zwischen VST-Plugin und VST-Host kann über die in den Basisklassen definierten Strukturen und Funktionen stattfinden. Ein Beispiel: Das vvvvBeattracker-Plugin ruft in dem Konstruktor die Funktion numParams() mit dem Integer-Wert 2 auf und gibt damit an, dass das Plugin eine Stereo-Eingabe verarbeiten kann. Die Basisklasse AudioEffect hat ein Objekt der Struktur *aeffect* und setzt in der Funktion numParams() darauf den Integer-Wert der ebenfalls *numParams* heißt auf Zwei. Auf diese Struktur kann auch der Host zugreifen und somit seine Aufrufe entsprechend mit einem Stereosignal tätigen. In den Basisklassen werden sehr viele Funktionen definiert, die einen großen Spielraum für die Kommunikation zwischen Host und Plugin geben.



7.5 Tracker

Tracker ist die Hauptklasse der Analyse und verwaltet die einzelnen Glieder der Verarbeitungskette. Die Klasse vvvvBeattracker besitzt eine Instanz von Tracker und gibt, wenn neue Audiodaten vom

Host übermittelt wurden, diese als ein Monosignal durch Aufruf der Funktion process() an Tracker weiter. Der Rückgabewert ist entweder *true*, falls in dem übergebenen Frame ein Beat detektiert wurde, oder *false*. In der Funktion process() von Tracker wird eine Verarbeitungskette durchlaufen, die einzelnen Stufen sind in eigenen Objekten gekapselt.
Der Aufruf der einzelnen Glieder erfolgt über die Funktion process(), die jedes der Objekte besitzt. Parameter sind die Eingabedaten an erster Stelle und die Ausgabedaten an zweiter Stelle. Die Daten der laufenden Verarbeitung werden von Tracker in einem Objekt der Klasse Signal in mehreren Arrays gehalten.

7.6 Die Verarbeitungskette

Die Organisation der Verarbeitung in einer Kette, bei dem die einzelnen Glieder nach außen hin gekapselt sind, hat den Vorteil, dass die Teile leicht verändert oder ausgewechselt werden können.

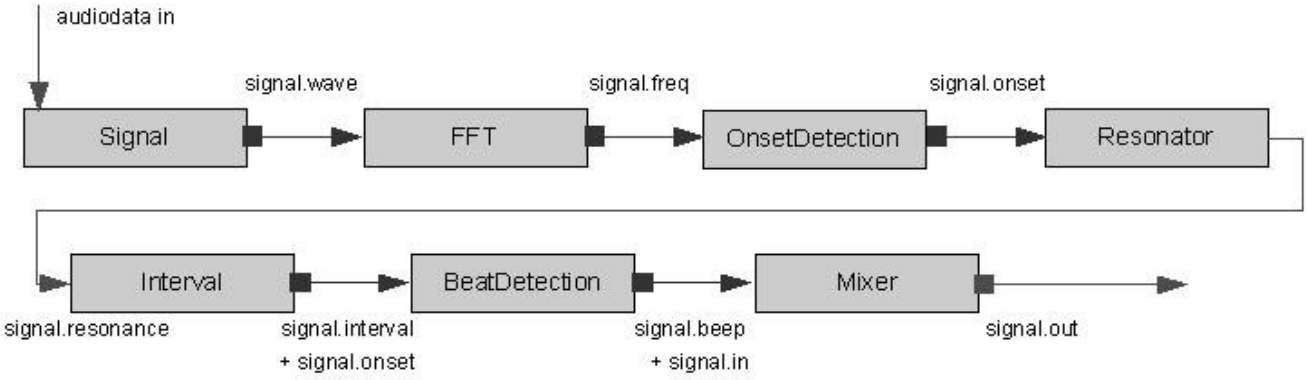
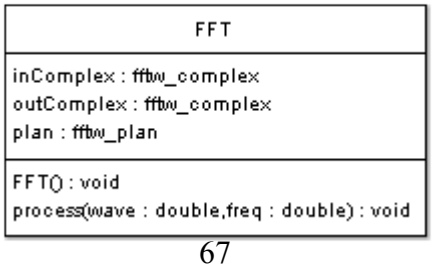


Abb.24: Verarbeitungskette des Beattracker-Plugins mit den Parametern

Das Objekt der Klasse Signal kopiert die Audiosamples, die Tracker als Parameter übergeben bekommen hat. Dann wird die Funktion process() des Objekts fft aufgerufen. Dort wird die Fourier-Transformation durchgeführt. Die Eingabedaten stehen in dem Parameter signal.wave. Die Ausgabedaten werden in den Parameter signal.freq geschrieben. Im nächsten Schritt bekommt OnsetDetection das Frequenzspektrum in signal.freq und gibt das Onset-Signal in den vier Feldern von signal.onset aus, dabei steht ein Feld für einen Kanal. In Resonator wird signal.onset auf Periodizität untersucht und das Ergebnis in signal.resonance ausgegeben. In Interval, dem nächsten Glied der Kette wird die Tempo-Hypothese basierend auf signal.resonance aufgestellt und in signal.interval ausgegeben. Die Berechnungen die in Interval durchgeführt werden sind recht aufwendig und Veränderungen in den Ausgaben der Kammfilterbank gehen relativ langsam vor sich. Daher hat es sich als ausreichend erwiesen die Tempo-Hypothese nur alle zehn Takte neu zu berechnen, dies führt zu einem starken Rückgang der Auslastung. BeatDetection bekommt die Onset-Signale aus den vier Kanälen in signal.onset und signal.interval. Ausgabewert ist signal.beep, er gibt an, ob zum aktuellen Zeitpunkt ein Grundschlag stattfindet oder nicht. Mixer, dem letzten Glied in der Kette, wird signal.wave, signal.out und signal.beep übergeben. Steht die boolesche Variable signal.beep auf true, so wird signal.wave mit einem Feedback-Signal gemischt und über signal.out ausgegeben.

7.7 Die Klassen der Verarbeitungskette

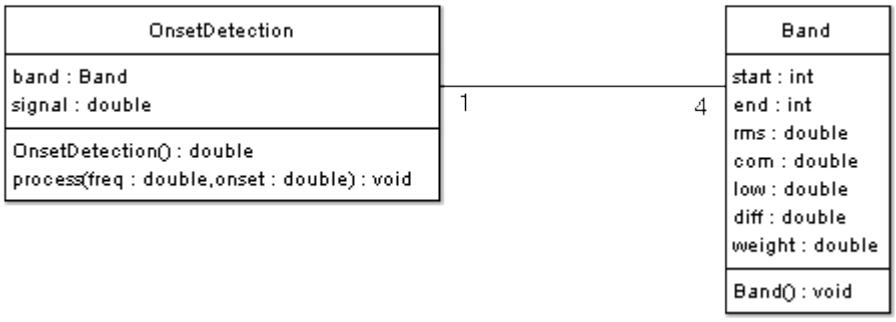


7.7.1 FFT

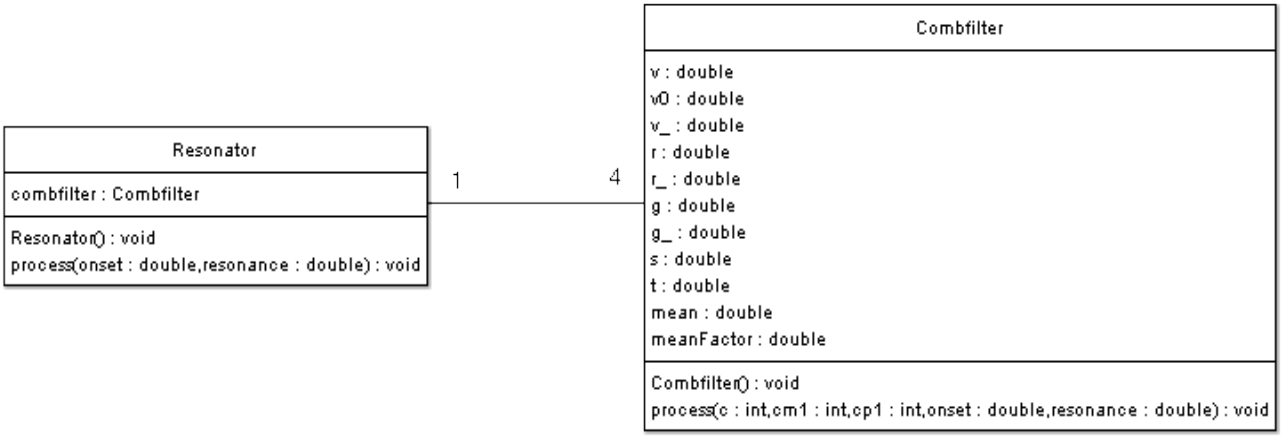
Die Fourier-Transformation wird mit Hilfe der Bibliothek FFTW durchgeführt. Bei der Initialisierungsphase wird die Funktion fftw_plan_dft_1d() aufgerufen, deren Aufgabe es ist eine optimierte Verarbeitungsvorschrift zu berechnen, die in der Variable *plan* gehalten wird. Die beiden Felder *inComplex* und *outComplex* sind zweidimensional (Imaginär- und Realteil) und nehmen double Werte auf. Bei dem Aufruf von process() werden die übergebenen Audiosamples in den Realteil von *inComplex* geschrieben, während der Imaginärteil auf Null gesetzt wird. Dann wird die Bibliotheksfunktion fftw_execute() aufgerufen und mit *plan* die Verarbeitungsvorschrift für eine eindimensionale Diskrete-Fourier-Transformation übergeben. Die Ausgabe der Funktion wird in *outComplex* geschrieben, in das Frequenzspektrum umgewandelt und über signal.freq ausgegeben.

7.7.2 OnsetDetection

Das OnsetDetection-Objekt bekommt von Tracker die Frequenzen des aktuellen Frames in signal.freq und gibt das Onset-Signal der vier Kanäle in signal.onset aus. In dem Arbeitsschritt der Onset-Detection wird der gesamte Frequenzbereich in 24 Subbänder geteilt. Die Daten dazu werden in 24 Objekten der Struktur *Band* gehalten. Sie haben mit *start* und *end* eine Beschreibung des ersten und letzten Indexwertes auf dem Feld signal.freq, auf dem sie arbeiten.

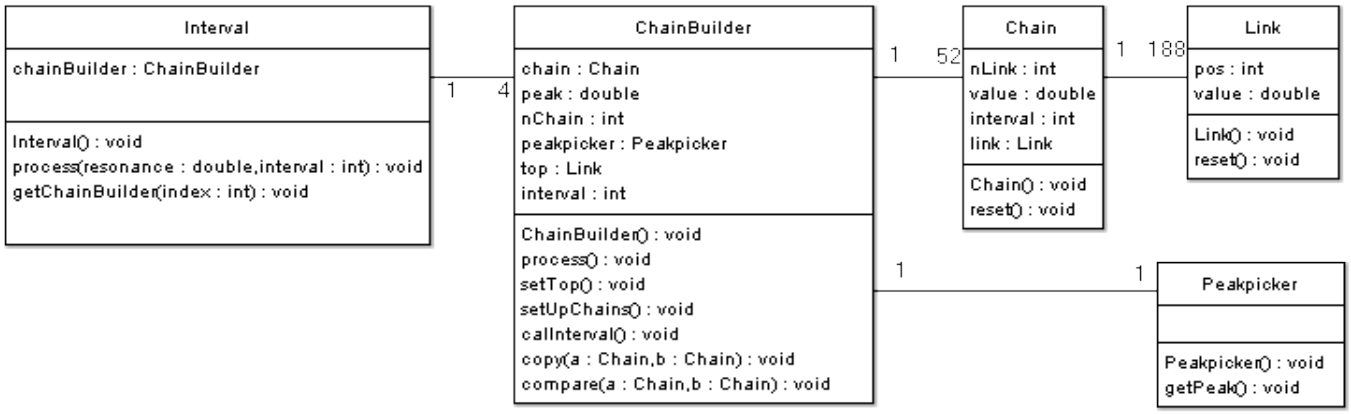


Weitere Felder von Band: Der Real-Mean-Square in *rms*. Das komprimierte Signal in *com*. Das lowpass-gefilterte Signal in *low*. Die Differenz in *diff* und das gewichtete Signal in *weight*. OnsetDetection hält eine Zählervariable *count* vor, die bei 0 beginnend pro Aufruf von process() inkrementiert wird und nach 1023 wieder bei 0 beginnt. Auf dem durch *count* bezeichneten Indexwert werden nun jeweils in den Band-Objekten die Felder mit den aktuellen Wert eines Frames belegt. Mit dieser Art der Verarbeitung kann sehr viel Kopierarbeit vermieden werden. Am Ende von der Funktion process() werden sechs angrenzende Bänder aufaddiert und bilden damit den aktuellen Onsetwert für einen Kanal.



7.7.3 Resonator

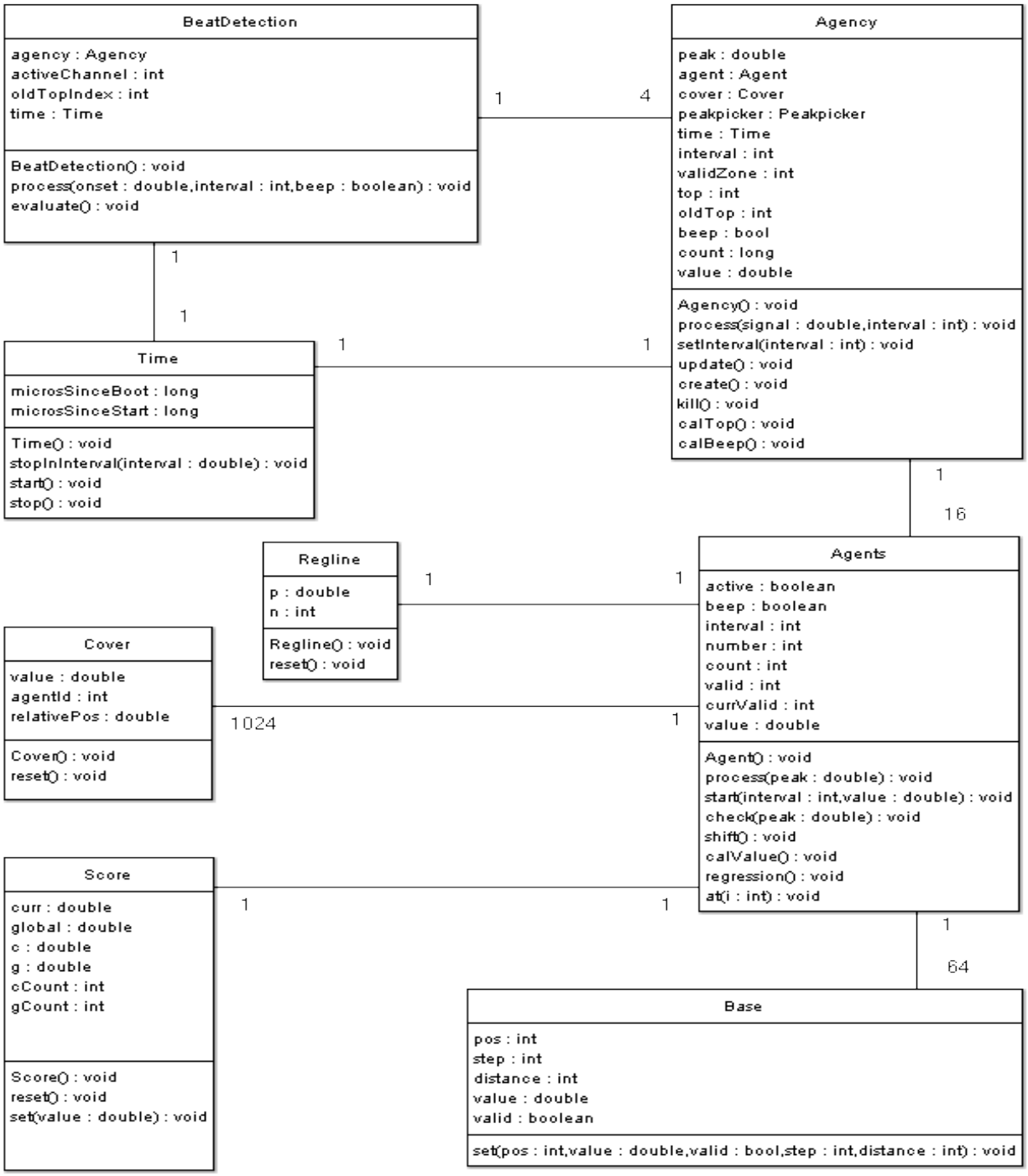
Um die Resonanzen auf die verschiedenen Periodenlängen zu berechnen, verwendet die Resonator-Klasse Objekte der Struktur Combfilter. Ein Objekt stellt dabei eine Kammfilterbank für jeweils einen Kanal dar. Die Verarbeitung wird auch hier durch eine Zählervariable gesteuert, die dem Combfilter als Parameter *c*, bei dem Aufruf von process() übergeben wird.



7.7.4 Interval

Die Aufgabe der Klasse Interval ist es, in den Resonanzen der vier Kanäle jeweils die Tempo-Hypothese zu bestimmen. Dazu verwendet Interval Objekte der ChainBuilder-Klasse, die die eigentliche Aufgabe für einen Kanal übernehmen. Interval dient damit im Prinzip als Schnittstelle, die Veränderung der Anzahl der Kanäle wäre damit leicht durchführbar. Ein ChainBuilder-Objekt bestimmt Peaks in den Resonanzen seines Kanals mit Hilfe eines Objektes der Peakpicker-Struktur.

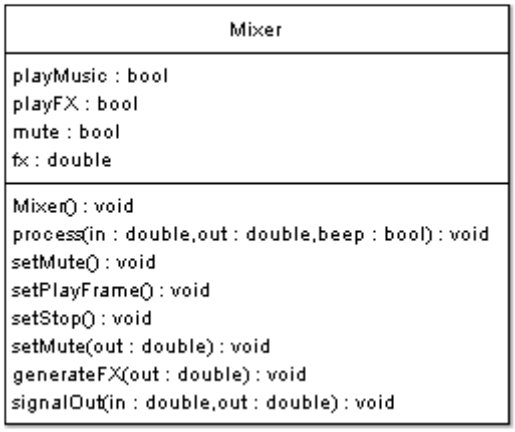
Im ersten Schritt werden in ChainBuilder.setupChains() Folgen von Peaks gebildet. Eine Folge besteht aus einem Objekt der Klasse Chain. Dieses wiederum hat für jedes Glied, daher jeden Peak, ein Objekt der Struktur Link. Nachdem die Ketten aufgebaut wurden, werden sie nun miteinander verglichen, um eine Reihenfolge zu bilden. Dies geschieht in der Funktion ChainBuilder.sortChains() in dem ein Selection-Sort (Sortieralgorithmus) auf Basis der Stärke der Peaks, die zu der Folge gehören, durchgeführt wird. Der Indexwert des ersten Peaks in der Folge bestimmt die Tempo-Hypothese in dem jeweiligen Kanal.



7.7.5 BeatDetection

In der Klasse BeatDetection findet die Ermittlung der Grundschlag-Positionen statt. Ausgangsdaten dazu werden in den Parametern von process() geliefert. Es sind die Onset-Signale aus den vier Kanälen die in signal.onset stehen. Die vier Tempo-Hypothesen darüber stehen in signal.interval. Falls ein Beat in dem übergebenen Frame detektiert wird, gibt die Funktion *true* zurück, ansonsten *false*. Die Klasse BeatDetection nutzt für die Verarbeitung in den einzelnen Kanälen Objekte der

Klasse Agency. Eine Agency besitzt 16 Agenten die verschieden Phasen-Hypothesen für das Onset-Signal haben. Die Agenten sind jedoch nicht immer alle parallel aktiv. Über Objekte der Klasse Time können BeatDetection und Agency verhindern, dass der zeitliche Abstand zwischen zwei Feedback-Signalen zu kurz wird. Die Cover-Objekte einer Agency dienen dazu festzuhalten, welche Peaks auf dem Onset-Signal durch Agenten belegt sind. Die Agenten halten Daten in Objekten verschiedener Strukturen: In Base stehen die Angaben zu den Stützpunkten. In Score befindet sich der kurzfristige und längerfristige Wert des Agenten. Regline enthält Informationen über die validen Stützpunkte eines Agenten und den Mittelwert der Intervalle dazwischen.



7.7.6 Mixer

Das letzte Glied in der Verarbeitungskette ist Mixer, hier wird das Feedback-Signal generiert. Mixer bekommt über process() die Audiodaten des Frames in '*in*' übergeben und kopiert diese in '*out*'. Je nachdem ob der dritte Parameter *beep* true oder false ist, wird ein Sinuston in '*out*' gemischt. Innerhalb von vvvv kommt Mixer nicht zum Einsatz, in vvvv kann beispielsweise über den Anschluss eines beep-Nodes an dem Ausgabe-Pin des Beattracker-Nodes ein Feedback-Signal ausgegeben werden.

8 Test

Der Test des Algorithmus besteht aus zwei Teilen: In dem ersten Teil wird die Tempo-Extraktion überprüft. Der zweite Teil bezieht sich auf die Positionen der Grundschläge. Bei der Auswahl der Lieder wurde ein breites Spektrum der Popmusik abgedeckt, wobei ein Schwerpunkt auf Stile gelegt wird, für die erfahrungsgemäß Musik-Visualisierungen erstellt werden. Innerhalb der Stile wurde versucht möglichst zufällig Stücke auszuwählen.

8.1 Testaufbau Tempo-Extraktion

Ein Testlauf besteht darin, einen halb-minütigen Auszug eines Musikstücks zu einem repräsentativen Zeitpunkt zu untersuchen. Dieser ist meist der Anfang des Stücks, es kann aber auch ein etwas späterer Zeitpunkt gewählt werden, falls der Anfang aus einem Intro, beispielsweise mit gesprochener Sprache oder einer Geräuschkulisse, besteht. Die Ausgabe der Komponente besteht in der globalen Tempo-Hypothese in BPM umgerechnet. Global bedeutet hier die Tempo-Hypothese des Kanals, der auch das Feedback-Signal liefert. Der Vergleichswert ergibt sich dadurch, dass in dem gleichen Zeitraum die Grundschläge durch Betätigung der Space-Taste bestimmt werden. Über die Intervalle wird ein Mittelwert gebildet und ebenfalls in BPM umgerechnet. Bewertet wurde wie in dem Test von Gouyon [Gouy04]: Doppelte und halbe Geschwindigkeiten gelten als richtig. Abweichungen die sich im Bereich der JND für Geschwindigkeiten bewegen, die bei 4% liegt, werden auch als richtig gewertet.

8.2 Testaufbau Positionen der Grundschläge

Ein Test auf die Positionen der Grundschläge der genaue Zahlenwerte als Ergebnis hat, ist sehr aufwendig und konnte in dem verfügbaren Zeitrahmen nicht durchgeführt werden. Darum besteht der Notenwert der sich ergeben hat aus einer Einschätzung. Auch wenn das Testergebnis damit subjektiv ist, spiegelt es doch recht gut die Qualität der Analyseergebnisse wieder, denn ein falsch gesetztes Feedback-Signal wirkt sofort erheblich störend. Die Qualität des Feedback-Signals wurde mit Noten bewertet. Stücke bei denen die Tempo-Extraktion nicht erfolgreich verlaufen ist, wurden nicht getestet.

*Stücke deren Rhythmuspart mit einem elektronischen, programmierten Klangerzeuger erstellt wurde.

Noten zu den Beats (gibt an welche Ereignisse erlaubt sind, um die jeweilige Note zu erzielen):

1 : Durchgehender Grundschlag einer Geschwindigkeit

2 : kurze Pausen

3 : Tempowechsel, Pausen

4 : Tempowechsel, Pausen und kleinere Fehler

5 : Keine oder fehlerhafte Beats

8.4 Kommentar

Anhand der Testergebnisse kann gesehen werden, dass die Tempo-Extraktion gut funktioniert. Etwa 80% der Geschwindigkeiten liegen im richtigen Bereich. Davon war etwa eine Hälfte doppelt oder halb so schnell wie der Vergleichswert, die andere Hälfte stimmt mit dem Vergleichswert überein. Das Beat-Tracking funktioniert jedoch nur bei etwa 50% der getesteten Musikstücke (Noten 1-4). Deutlich kann jedoch gesehen werden, dass die Qualität der Auswertung von oben nach unten zunimmt, wobei nach unten hin der Bereich liegt, der erfahrungsgemäß für Musik-Visualisierungen in Frage kommt. In der Menge der Musik, deren Rhythmus durch ein programmiertes, elektronisches Instrument erzeugt wurde schneidet die Auswertung gut ab: Von 28 Stücken funktionieren 21, 16 davon gut bis sehr gut.

8.5 Auslastung

Die CPU-Auslastung auf einem Intel T2250 Dual-Core mit 1.73 Ghz und Windows XP Service Pack 2 bewegt sich um die 10%. Es treten kurze Spitzenwerte auf, die bei 22% Auslastung liegen können.

9 Ergebnis

9.1 Vergleich mit der Zielsetzung

Ist die Zielsetzung erreicht, für einen Strom von musikalischen Audiodaten, ohne Beschränkung hinsichtlich der Instrumente und des Stils, die Grundschläge zu detektieren? Dies ist sicher nicht der Fall, doch dazu ist auch kein anderer heute existierender Algorithmus fähig. Das Feld der Music-Information-Retrieval im allgemeinen und die Grundschlagerkennung im speziellen sind aktuelle Felder der Forschung und werden es aller Voraussicht nach auch noch länger bleiben.
Kann das Verfahren für eine Musik-Visualisierung eingesetzt werden? Musik deren Rhythmus mit elektronischen, programmierten Klangerzeugern generiert wurde, wobei auch Sampler darunter fallen (Ausschnitte des Klangs realer Instrumente werden genommen, manipuliert und neu

angeordnet), schneiden in dem Test gut ab. Für diese Musik werden auch am häufigsten mit Computern generierte Musik-Visualisierungen erstellt.

9.1.1 Funktionalität im vvvv-System

Die Beattracker-Komponente kann innerhalb des vvvv-Systems als Node eingefügt werden, über den Eingabe-Pin Audiodaten erhalten und ein Feedback-Signal am Ausgabe-Pin ausgeben. Über den integrierten VST-Host kann das Plugin geladen und Daten damit ausgetauscht werden. Dabei besteht jedoch das Problem, dass die Audioverarbeitung in dem DirectShow-Filtergraphen, der in vvvv verwendet wird, mit sehr großen Sample-Blöcken (11025 Samples) arbeitet, die unregelmäßig geliefert werden. Diese Blöcke können zwar in kleinere Frames zerlegt und dafür die Positionen der Grundschläge ermittelt werden, aber da es nicht klar ist, wann die Frames tatsächlich über die Soundkarte ausgegeben werden (wegen den Unregelmäßigkeiten), kann bisher noch kein passendes Feedback-Signal ausgegeben werden. Verschiedene Möglichkeiten das Problem zu lösen sind angedacht, konnten aber bis zum Zeitpunkt der Erstellung dieses Textes noch nicht realisiert werden.

9.1.2 Leistungsanforderungen

Das Verfahren arbeitet in Echtzeit und mit etwa 10% Auslastung für die Verarbeitung lassen sich gut noch andere vvvv-Nodes daneben betreiben.

9.1.3 **Bedienbarkeit**:

Die Bedienung und Einbindung der Beattracker-Komponente in einem vvvv-Node funktioniert so wie gefordert: Den Node im Auswahlmenü wählen, bei dem Eingabepin an der Oberseite eine Audioquelle anschließen und unten an dem Ausgabepin das Feedback-Signal entgegennehmen.

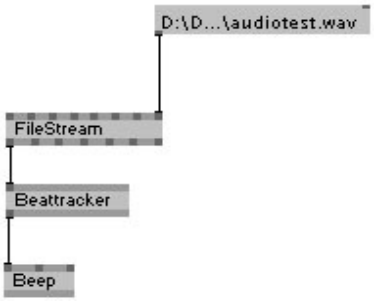


Abb.25: Akustisches Feedback-Signal über einen beep-Node

9.2 Ausblick

VVVV-Nodes lassen sich einfach erweitern, indem ihnen weitere Ein- und Ausgabe-Pins hinzugefügt werden. Dem Anwender könnte die Möglichkeit gegeben werden, die Frequenz der Grundschläge zu beeinflussen, die durch das Feedback-Signal ausgegeben werden.
Mit Hilfe zweier Eingabe-Pins könnte ein Tempobereich festgelegt werden. Fällt die Frequenz des detektierten Grundschlags in diesen Bereich, wird das Feedback-Signal normal ausgegeben. Falls nicht wird überprüft, ob ein ganzzahliges Vielfaches oder Teiler in diesen Bereich fällt und die Frequenz entsprechend angepasst. Ein elegantere Möglichkeit ist die Vorgabe der Frequenz und Phase durch einen Tap-Button (beispielsweise die Space-Taste wie im Test). Dieser kann über einen einfachen Pin realisiert werden, der einen booleschen Wert entgegen nimmt. Das Verfahren kann nun dazu dienen die Phaseninformation immer aktuell zu halten. Wenn der Anwender das Tempo ändern möchte, beispielsweise auf die Hälfte, gibt er dies einfach durch eine neue Eingabe vor.
Anwender könnten den Wunsch haben ein Feedback-Signal zu verwenden, das nicht den regelmäßigen Grundschlag widerspiegelt, sondern den Rhythmus. Dies ist leicht umzusetzen, indem die Onset-Signale aus den Kanälen an vier Pins ausgegeben werden. Der Vorteil ist, dass die Onset-Signale nicht „falsch“ liegen. Bei Tests mit einem akustischen Feedback-Signal konnte festgestellt werden, dass die Onsets immer zu dem Rhythmus des Musikstücks passen.

Geplant ist den VST-Host so zu erweitern, dass er als Host für beliebige VST-Plugins dienen kann. In einem anderen Node integriert kann somit auf die Vielzahl von Plugins zur Manipulation und Generierung von Audio zugegriffen werden.

9.2 Analyse der Testergebnisse

Warum unterscheiden sich die Ergebnisse der Analyse von elektronischer und nicht-elektronischer Musik so stark? Als Gründe dafür werden angenommen:

Unklares Signal
Abweichungen
Komplexität

In natürlicher Musik überlagern sich die Signale stärker. Ein Beispiel hierfür ist Rockmusik, deren Betonung auf einer verzerrten Gitarre liegt. Das Level der Dynamik ist hoch und umfasst einen großen Frequenzbereich, dadurch treten andere Signale, auch perkussive, nicht klar hervor.

Menschen produzieren beim Spiel der Instrumente immer Abweichungen, dies kann auch gewollt und Teil eines Stilmittels sein. Für die Auswertung bedeuten diese Abweichungen jedoch eine erhebliche Erschwerung.

Es gibt Musik die rhythmisch sehr komplex ist. Manche Musikstücke aus dem Jazz, Latin oder der experimentellen Musik fallen darunter. Ein Beispiel für ein Musikstück aus dem Test, dass schwer auszuwerten ist, obwohl es einen starken perkussiven Anteil hat, ist aus dem Latin-Bereich (Redencion von Cachaito aus dem Test). Der rhythmische Aufbau ist komplex und doch gibt es ein Instrument, dass meist genau zu den Zeitpunkten des Grundschlags zum Einsatz kommt. Es ist eine Art Ratsche, sie ist leise und doch unter den anderen, lauteren Instrumenten zu vernehmen. Ein Mensch kann die Einsätze dieses Instrumentes nutzen, um sein kognitives Modell über den Grundschlag an diesem regelmäßigen Signal auszurichten. Doch für ein rechnergestütztes Verfahren ist es schwierig dieses Signal unter der Überlagerung zu bestimmen. Es ist das aus der Sprachverarbeitung bekannte Cocktail-Party-Problem, bei dem ein Sprecher durch die auditive Wahrnehmung aus einer Vielzahl von Sprechern heraus gefiltert wird.

9.3 Bewertung und Verbesserungsmöglichkeiten

Ein Vorteil bei dem Verfahren des Beat-Tracking ist, dass es gut in einzelne Schritte aufgeteilt werden kann, die sich individuell analysieren und eventuell verbessern lassen.

Die Detektion der Onsets kann überprüft werden, indem der Audiostrom Frame für Frame

abgespielt wird, wobei ein Frame solange in einer Schleife über die Soundkarte ausgegeben wird, bis weiter geschaltet wird. Akzentuierungen können hierbei gut wahrgenommen werden. Resultiert ein Peak aus einer Akzentuierung, so kann der Onset als detektiert gelten. Bei Tests konnte die Korrelation der Dynamik mit den resultierenden Peaks festgestellt werden. Zu genaueren Ergebnissen könnte eine Erhöhung der zeitlichen Auflösung führen. Da in dem Verfahren auf Basis von Frames gearbeitet wird, könnten diese überlappend verarbeitet werden. Natürlich würde eine Erhöhung der Auslastung damit einhergehen, darum wäre darüber nachzudenken, bei dem vvvv-Node zwei Eingabe-Pins anzubieten, die mit unterschiedlicher Auflösung arbeiten.

Die Kammfilterbank und die Tempo-Extraktion kann durch die Ergebnisse des im vorigen Kapitel besprochenen Tests validiert werden. Dabei fiel das Ergebnis mit etwa 80% richtig eingeschätzter Stücke gut aus. Die Vorgehensweise, Folgen von Peaks zusammenzufassen und davon die stärkste Folge zu berechnen, scheint in die richtige Richtung zu gehen. Auch hier könnte die Erhöhung der zeitlichen Auflösung noch zu genaueren Ergebnissen führen. Auch wenn doppelte oder halbierte Grundschlag-Frequenzen nicht falsch sind, wäre es besser, genau das Tempo zu treffen, dass die meisten Menschen für ein Stück angeben würden. Allein aus der Resonanz der Kammfilterbank auf dieses Tempo zu schließen, wird als schwierig bis unmöglich eingeschätzt.

Mit dem Onset-Signal und einer richtigen Tempo-Hypothese können die Positionen der Grundschläge bestimmt werden. Wenn die eben gemachten Annahmen über die Qualität der Auswertung richtig sind, müsste hier der Schwachpunkt des Verfahrens liegen.
Die Arbeitsweise des Verfahrens besteht darin, auf der metrischen Ebene der Grundschläge regelmäßige Betonungen zu ermitteln. Eine häufige Fehlerquelle ist das Fehlen dieser Betonungen über einen längeren Zeitraum. Theoretisch könnte das Feedback-Signal trotzdem unendlich weitergeführt werden, da die Frequenz und Phase des Grundschlags ermittelt wurden. Praktisch summieren sich kleinere Fehler aber schnell auf und die Phase driftet bald in einen als Fehler wahrgenommenen Bereich ab.
Die Überlegung ist, auf mehreren metrischen Ebenen gleichzeitig zu arbeiten. Dabei werden Onsets in ein Raster eingefügt, dass sich selber aktualisiert. In Bereichen des Musikstücks, die kein klares Signal auf der Ebene des Grundschlags haben, können dann auch Onsets einer niedrigeren metrischen Ebene dazu dienen die Phaseninformation aktuell zu halten.

9.3 Fazit

Das hier vorgestellte Verfahren zur Ermittlung der Grundschläge ist in der Praxis einsetzbar, aber sicherlich kann in den einzelnen Teilen noch viel verbessert werden. Doch durch die Struktur der Anwendung ergibt sich die Möglichkeit separat an den Bausteinen zu arbeiten, Tests durchzuführen und direkt Feedback dazu zu erhalten.

Anhang

Quellen

[Gouy04] Fabien Gouyon
„An experimental comparison of audio tempo induction algorithms“
www.iua.upf.edu/mtg/publications/9d0455-IEEETSAP-Gouyon.pdf
Stand 31.7.2007
Erscheinungsjahr 2004

[Dix06] Simon Dixon
„MIREX 2006 Audio Beat Tracking Evaluation: BeatRoot“
http://www.music-ir.org/evaluation/MIREX/2006_abstracts/BT_dixon.pdf
Stand 31.7.2007
Erscheinungsjahr 2006

[Hon99] Henkjan Honing, Peter Desain
„Computational Models of Beat Induction: The Rule-Based Approach“
http://www.nici.ru.nl/mmm/papers/dh-100/dh-100.pdf
Stand 31.7.2007
Journal of New Music Research
Erscheinungsjahr 1999,Vol. 28, No. 1, pp. 29-42

[Goto96] M. Goto, Y. Muraoka
„Beat Tracking based on Multipleagent Architecture — A Real-time Beat Tracking System for Audio Signals“
http://libra.msra.cn/paperdetail.aspx?id=259532
Stand 31.7.2007
Second International Conference on Multiagent Systems (pp. 103–110)
Erscheinungsjahr 1996

[Schlo85] W. Andrew Schloss
„On The Automatic Transcription of percussive music“

http://ccrma.stanford.edu/STANM/stanms/stanm27/stanm27.pdf
Stand 31.7.2007
Erscheinungsjahr 1985

[Klap04] Anssi P. Klapuri, Antti J. Eronen, and Jaakko T. Astola
„Analysis of the Meter of Acoustic Musical Signals“
http://www.cs.tut.fi/sgn/arg/klap/sapmeter.pdf
Stand 31.7.2007
Erscheinungsjahr 2004

[Gouy06] Fabien Gouyon, Simon Dixon
„Computational Rhythm Description“
http://www.ofai.at/~fabien.gouyon/slides-GouyonDixon-ISMIR06.pdf
Auf der 7th International Conference on Music Information Retrieval
Stand 31.7.2007
Erscheinungsjahr 2006

[Schei97] Eric D. Scheirer
„Tempo and beat analysis of acoustic musical signals“
http://www.iro.umontreal.ca/~pift6080/documents/papers/scheirer_jasa.pdf
Stand 31.7.2007
Erscheinungsjahr 1997

Abbildungsverzeichnis

01 Hierarchischer Aufbau von Rhythmus
http://de.wikipedia.org/rhythmus

02 Tempoverteilungen bei 3199 Musikstücken
aus [Gouy04]

03 Signalverarbeitung
erstellt mit OpenOffice-Draw
04 Spannungsverlauf
http://medienlab.fh-wedel.de
05 Fourier-Analyse
Krol,Mzyk „Extraction of Rhythmical Features of Audio Signals“ 2005
06 Hamm-Fensterfunktion
http://de.wikipedia.org/fensterfunktion
07 Hüllkurve
A. Klapuri „Sound event detection and rhythmic parsing of music signals“ 2004
08 Filter
http://www.dspguide.com/
09 Frequenzgang dreier Lowpass-Chebyshev-Filter
http://www.dspguide.com/
10 Aufbau eines Feedback-Kammfilters
http://en.wikipedia.org/combfilter
11 M-Law Kompression
http://es.wikipedia.org/ulaw
12 Inter-Onset-Intervalle
erstellt mit Audacity
13 vvvv-Patch
erstellt mit vvvv
14 Installation im einem Club
http://meso.net/main.php
15 Aufbau eins Beat-Tracking-System
Aus [Gouy06]
16 Versuch von Scheirer
aus[Schei97]
17 Wellenform mit Onsets
aus [Gouy06]
18 Autokorrelation
aus [Gouy06]
19 Onset-Histogramm
aus [Gouy06]
20 Ausgabe einer Tempo-Extraktion
aus [Gouy04]

Tabellen

Dezibel

Verhältnis	dB
1.000000 : 1	- 0.00
0.316000 : 1	-10.00
0.100000 : 1	-20.00
0.010000 : 1	-40.00
0.001000 : 1	-60.00
1.122018 : 1	+ 1.00
1.258925 : 1	+ 2.00
2.000000 : 1	+ 6.02
4.000000 : 1	+12.04
8.000000 : 1	+18.06

Quelle: http://www.hifi-selbstbau.de/text.php?id=27&s=read

Schalldruckpegel

10 dB : Atmen, raschelndes Blatt

20 dB : Ticken einer Armbanduhr

30 dB : Flüstern

40 dB : leise Musik

45 dB : übliche Geräusche in der Wohnung

50 dB : Regen, Kühlschrankgeräusche

55 dB : normales Gespräch

60 dB : Nähmaschine, Gruppengespräch

65 dB : Kantinenlärm

70 dB : Fernseher, Schreien, Rasenmäher

75 dB : Verkehrslärm

80 dB : Telefonläuten, Presslufthammer

90 dB : Lastwagen

100 dB : Ghettoblaster

110 dB : Diskomusik, Symphoniekonzert, Motorsäge, Autohupe

120 dB : Kettensäge, Presslufthammer, Gewitterdonner

130 dB : Autorennen, Düsenjäger

Quelle:http://www.welt.de/print-welt/article334313/Vom_Ticken_der_Uhr_bis_zum_Presslufthammer_-_Geraeusche_in_Dezibel.html

Links

vvvv	Das vvvv-Multimedia-Framework
	http://vvvv.org/tiki-index.php
VST	VST-SDK und Dokumentation
	http://ygrabit.steinberg.de/~ygrabit/public_html/index.html
Ismir	International Conference on Music Information Retrieval
	http://ismir2006.ismir.net/
Mirex	Music Information Retrieval Evaluation eXchange
	http://www.music-ir.org/mirexwiki/index.php/Main_Page
Audacity	kostenloses Audiobearbeitungsprogramm
	http://www.audacity.de/
Hydrogen	kostenlose Drum-Machine
	http://www.hydrogen-music.org/

Erklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Jens Röhner / Berlin den 31.Juli 2007