Transkription polyphoner Musik

Martin Schobert
<martin@schwingungsebene.de>

2008

Inhaltsverzeichnis

1 Einleitung

Auf der Internationalen Musikmesse in Frankfurt an Main im März 2008 stellte die Firma Celemony Software GmbH eine Technologie namens Direct Note Access vor. Die Fachpresse sah darin das Highlight der Messe. So titelte der Heise Verlag „Sensation auf der Musikmesse: Wave-Daten wie MIDI verändern“ [Hil08]. Zahlreiche Kommentare zu Blog-Einträgen und in Internet-Foren vermittelten durchweg den Eindruck, dass diese Technologie mit Begeisterung und Erstaunen aufgenommen wurde.

Aus der Sicht der Signalverarbeitung stecken in dem Konzept im wesentlichen zwei Problemfelder. Zum einen was man als Transkription bezeichnet und zum anderen die Auftrennung einer polyphonen Sound-Spur, so dass man für jede gespielte Note deren isolierten Klang erhält. Für beide Probleme existieren zahlreiche Lösungsansätze. Diese Ausarbeitung beschäftigt sich mit der Transkription von polyphonen Audiodaten und stellt ein Verfahren vor.

1.1 Celemony und Direct Note Access

Direct Note Access ist das Marketing-Schlagwort der Firma Celemony. Im Kern geht es darum, polyphones Audiomaterial auf der Ebene einzelner Noten verändern zu können. Die Akkorde eines Instrumentes rechnet eine Software auseinander, so dass jeder Ton des Akkords als Note vorliegt und in seinen Eigenschaften wie Tonhöhe, Timing und Dauer verändert werden kann. Die Noten kann man mit Hilfe der Maus modifizieren, indem man die „Blobs“, die Hüllkurven-Sinnbilder, verschiebt und streckt. Das Ergebnis ist sofort anhörbar. Als Ausgangsmaterial genügt ein Mixturklang in Form von Wave-Daten. Das Ergebnis soll klingen wie das Ausgangsmaterial – aber mit veränderten Noten.

Von der Firma Celemony gibt es bisher eine Reihe von Produkten auf dem Markt, die die Transkription monophoner Audiospuren ermöglichen. Monophon bedeutet, dass nicht mehrere Noten gleichzeitig erklingen. Das war bei der bereits existierenden Software Melodyne bisher eine Einschränkung. Im Herbst 2008 soll man die neue Technologie als Software unter dem Namen „Melodyne Plugin 2“ erwerben können.

Alle bisherigen Versuche, polyphone Musik abzuschreiben, unterlagen stets Einschränkungen. Der Begriff Direct Note Access ist offen hinsichtlich der In-

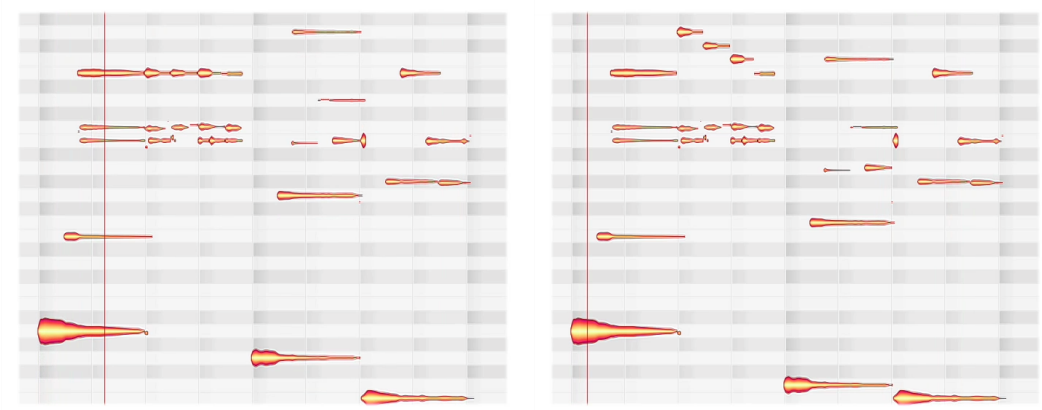

Abbildung 1.1: Auseinander gerechnete Gitarrenakkorde: links das Original, rechts veränderte Akkorde (Bildmaterial aus dem Demo-Video [Cel08])

terpretation. Und so ist es eine spannende Frage, in wie weit sich der Vorsatz, direkten Zugriff auf Noten zu ermöglichen, mit bestehenden Technologien umsetzen lässt. Das Demo-Video geht auf Einschränkungen nicht direkt ein. In dem Video werden Gitarrenspuren benutzt. [Cel08] Man kann deshalb vermuten, dass Instrumente mit harmonischen Obertönen besser geeignet wären. Technology Review gibt den Mann hinter Celemony, Peter Neubäcker, wie folgt wieder:

> Neubäcker is quick to admit that the software isn't perfect. It works well with clean, unprocessed instruments, such as acoustic guitar or piano. It will work on a heavily distorted electric guitar but may, for example, read some high-pitched harmonics as separate notes.
>
> Nor is the software quite the equivalent of a human ear, able to distinguish actual instruments from one another. Like an audio idiot savant, it can tease individual notes out of a complicated six-note chord. But feed it a recording in which two instruments are playing the same note–say, the trumpet and piano on an old Ellington record–and it will treat them as a single entity, rather than distinguishing their separate voices. [Bor08]

Ob die Einschränkungen bedeutsam sind, wird die Praxis zeigen. Mögliche Anwendungsbereiche der Software sind Notenkorrekturen bei bereits eingespielten Spuren und Umarbeitungen von Samples. Damit liegt in den meisten Fällen Ausgangsmaterial vor, das lediglich ein Instrument repräsentiert, so dass Anforderungen an das Audiomaterial bereits erfüllt sind. Wenn sich die Umsetzung von Direct Note Access tatsächlich als die „revolutionäre Tech-

nik“ [Hil08] herausstellt, wird sich auch automatisch der Bedarf ergeben, das Verfahren weiter zu optimieren – durch Celemony oder ggf. deren Konkurrenz.

Durch das Meldoyne Plugin 2 gehört Celemony zu den Pionieren der praktischen Lösung dieser Aufgabe. Andererseits haben sich auch andere mit dem Problem auseinandergesetzt und dazu Arbeiten veröffentlicht. Die Forschung geht bis in die 70er Jahre zurück. Seit etwa zehn Jahren findet sie verstärkt statt. Des Weiteren gab es bereits vor Celemony kommerzielle Umsetzungen, zumindest für den Teil der Transkription polyphoner Musik, beispielsweise Note Chaser von Peter Hutchinson[1] (1997), WIDI RecognitionSystem vom Music RecognitionTeam später Widisoft[2] (2001), AKoff Music Composer[3] (2001), AmazingMIDI von arakisoftware[4] (2003) und IntelliScore von Innovative Music Systems[5] (2004). [Kla04, S. 2]

Da die Software von Celemony jünger ist, kann man mutmaßen, dass sie irgendwie besser sein muss. Es ist davon auszugehen, dass sich die Autoren der Software mit alternativen Verfahren auseinandergesetzt haben, bevor sie ihre eigene Entwicklung begannen. Ferner gehört zum Konzept Direct Note Access nicht nur ein technisches Verfahren der Transkription, sondern auch die Zuordnung, welche Teile des Klangs eine einzelne Note ausmachen sowie die Interaktion mit den Noten. Eine aufgepeppte GUI für Notenbearbeitung erfüllt die Vorstellung von Direct Note Access mehr als ein WAV to MIDI .

Eine der spannenden Fragen ist, wie so etwas funktioniert. Die Firma Celemony hat ihr Verfahren zum Patent angemeldet, um die Erfindung zu schützen. Im Gegenzug muss die Firma in einer Patentschrift offenlegen, worin die Erfindung besteht. Vor Herbst 2009 wird es aber dazu laut Celemony keine technischen Publikationen geben.[6] Und ob sich aus der Offenlegung brauchbare Informationen ergeben ist nicht zuletzt eine Frage der Breitbandigkeit der Patentanmeldung. Im bereits zitierten Artikel der Technology Review gibt Neubäcker nur einen vagen Einblick in die Arbeitsweise:

> Neubäcker says that he spent time looking at spectral analyses, seeking “structures in the signal” that could identify a chord’s fundamental notes and link each of them with its own overtones. [Bor08]

[1] `http://www.tnsoptima.com/soundidea/notechaser.html`, abgerufen am 05. August 2008
[2] `http://www.widisoft.com/english/products.html`, abgerufen am 05. August 2008
[3] `http://www.akoff.com/music-composer.html`, abgerufen am 05. August 2008
[4] `http://www.pluto.dti.ne.jp/~araki/amazingmidi/`, abgerufen am 05. August 2008
[5] `http://www.intelliscore.net/`, abgerufen am 05. August 2008
[6] Antwort von Jörg Hüttner, Head of Product Support bei Celemony via Email vom 23. Juli 2008

Neubäcker nennt in dem Artikel von Technology Review die Stichworte „fundamental note“, also erste Harmonische und „overtones“. Da diese auch für andere Methoden relevant sind, sollen sie im Folgenden erläutert werden.

1.2 Grundschwingung und Obertöne

Wenn eine beidseitig eingespannte Saite, beispielsweise bei einem Klavier oder einer Gitarre, schwingt, treten an den Rändern Reflexionen auf und die Wellen überlagern sich, ähnlich Wasserwellen, die an einer Wand reflektieren. Bei bestimmten Frequenzen ergeben sich durch beidseitige Reflexion stationäre Schwingungsmuster, stehende Wellen genannt. Die stehenden Wellen entsprechen der Resonanzfrequenz der Saite. Die kleinste Resonanzfrequenz f_0 nennt man Grundschwingung oder erste Harmonische. Ganzzahligen Vielfache der Grundschwingung f_0 entsprechen Wellenlängen, die gleich der Länge der Saite sind. Die Resonanzfrequenz, die doppelt so hoch ist wie f_0, nennt man zweite Harmonische oder erster Oberton. Die Menge aller Resonanzfrequenzen bildet das Resonanzspektrum. Aber nicht jede Resonanzfrequenz ist auch eine Harmonische, sondern nur die Schwingungen, die ein ganzzahliges Vielfaches von f_0 sind.

Bei musikalischen Tönen bestimmt der Grundton meistens die empfundene Tonhöhe. Die Tonhöhe (engl. *pitch*) ist eine psychoakustische Größe. Der Grundton ist eine messbare physikalische Größe. Die Reihe der Obertöne bestimmt wesentlich die Klangfarbe. Der Klang eines Instrumentes hängt davon ab, welche Bauteile noch mitschwingen. Saiten- und Blasinstrumente erzeugen näherungsweise harmonische Obertonreihen. Glocken und Xylophone beinhalten z.B. auch nicht-harmonische Obertöne. Abbildung 1.2 zeigt das Spektrum eines Klaviertons und eines Thai-Gongs im Vergleich.

Es gibt keinen einzelnen offensichtlichen Weg, den Grundton f_0 aus einem akustischen Signal zu berechnen. Anssi Klapuri hat sich in seiner Dissertation mit Musiktranskription beschäftigt und verschiedene Verfahren untersucht. Aus algorithmischer Sicht teilt er die Verfahren im wesentlichen in zwei Kategorien ein: Verfahren, die nach Partialtönen an den Stellen suchen, an denen man Harmonische erwartet und Verfahren, die Abstände zwischen den Harmonischen untersuchen. Die f_0-Abschätzer aus der letzten Kategorie, die also Intervalle untersuchen, sollen bei Inharmonitäten bessere Ergebnisse liefern. [Kla04, S. 23–26]

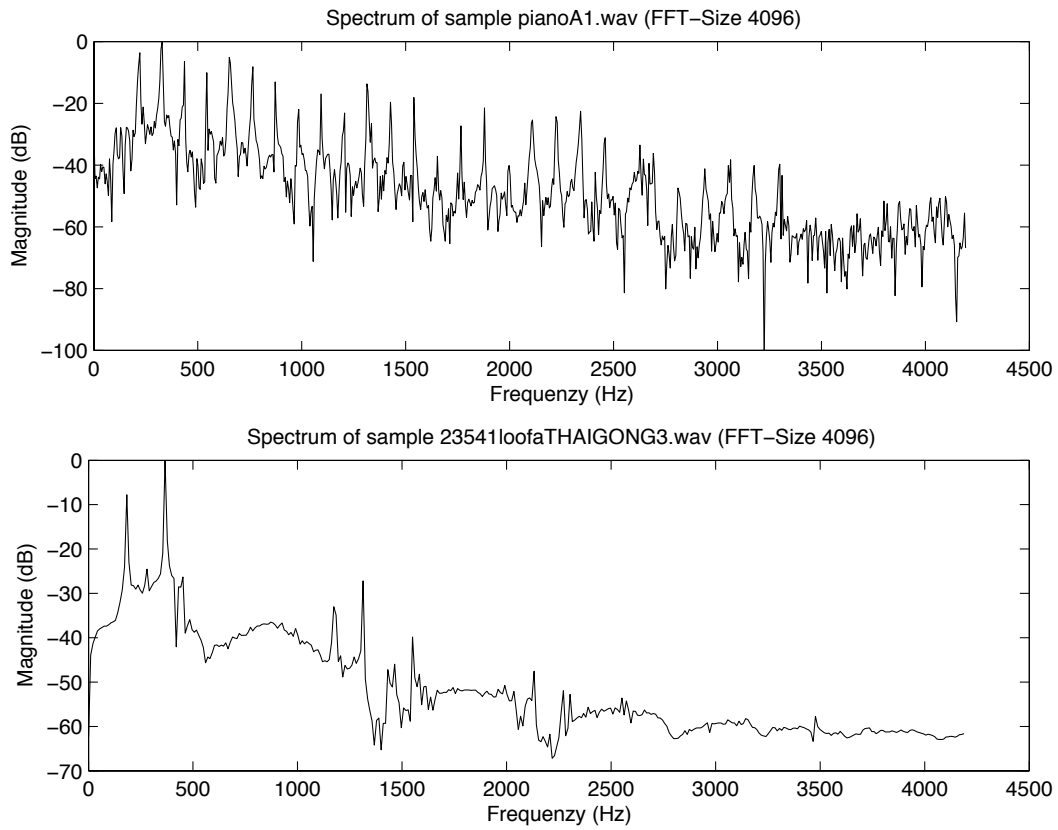

Abbildung 1.2: unten: Spektrum eines Thai-Gongs, oben: Spektrum eines Klaviertons mit $f_0 = 220$ Hz (durch Messung: $f_0 = 218$ Hz, $f_1 = 326$ Hz, $f_2 = 435$ Hz, $f_3 = 544$ Hz, ..., auch wenn der Dateiname vermuten lässt, dass es sich um den Kammerton mit 440 Hz handelt.)

Es gibt einen weiteren Ansatz, f_0 zu bestimmen. Die Hüllkurve des Audiosignals in der Zeit-Domäne beinhaltet Informationen über die Periodizität der Obertöne. In seiner Dissertation stellt Klapuri eine entsprechende Methode vor und erweitert sie, um mehrere Grundtöne für polyphone Audiodaten herauszufinden. In dem Paper „A Perceptually Motivated Multiple-F0 Estimation Method“ [Kla05] greift er das Verfahren erneut auf und stellt eine verbesserte und vereinfachte Version dessen vor. In [RK05] verwendet er die Methode zur f_0-Abschätzung, um es um wahrscheinlichkeitstheoretische Modelle zu erweitern.

Das zugrunde liegende Verfahren von Klapuri orientiert sich an Arbeiten der Wissenschaftler Ray Meddis und Lowel O´Mard, die Anfang der 1990er Jahre über Tonhöhenempfindungen veröffentlichten. Die Ursprünge gehen aber bis in die 50er Jahre zurück. [Kla04, S. 28]

2 Abschätzung einer Grundfrequenz

Die Methode, die hier betrachtet werden soll, ist „perceptually motivated“, d.h. sie basiert auf Theorien, wie sich die Wissenschaftler den auditiven Wahrnehmungsapparat vorstellen. Allerdings wird das Modell um so abstrakter, je mehr es in Richtung neuronale Verarbeitung geht. Die späteren Verarbeitungsschritte sind deshalb nicht unumstritten.

Dem Modell nach arbeitet die Hörschnecke wie eine Filterbank, bestehend aus mehreren Bandpassfiltern, die akustische Wellen in einzelne Kanäle auftrennt. Das Signal eines einzelnen Kanals repräsentiert die Schwingung der Basilarmembran an einer bestimmten Stelle in der Hörschnecke. Die Eigenschaften der Basilarmembran, etwa die Steifheit, ändern sich entlang der Hörschnecke, was die Frequenzauftrennung möglich macht. Im Innenohr wird die mechanische Schwingung der Basilarmembran mittels der Inneren Haarzellen in elektrische Signale umgewandelt. Diesen Konvertierungsvorgang nennt man Transduktion. Die Basilarmembran erregt aber nicht direkt die Inneren Haarzellen, sondern indirekt über die Äußeren Haarzellen.

Im Modell gibt es diese Transduktion auch. Sie wandelt die Signale aus der Filterbank in eine Repräsentation um, die die Wahrscheinlichkeit angibt, dass die Haarzellen „feuern“. Ursprünglich wurde die Transduktion durch Differentialgleichungen beschrieben. Es hat sich dann herausgestellt, dass man die Transduktion auch durch einen vereinfachten Prozess modellieren kann, der etwa einer Hüllkurvenextraktion entspricht. Was weiter im Gehirn passiert, ist Gegenstand der psychoakustischen und neurologischen Forschung, liegt aber bisher weitgehend im Dunkeln.

Die Hüllkurven der auditv gefilterten Signale beinhalten Informationen über den spektralen Abstand von Obertönen. Diese Informationen werden durch das hier vorgestellte Verfahren mittels praktischer Signalverarbeitung ausgewertet. Aber warum findet man den spektralen Abstand in der Hüllkurve?

Jeder Schwingung kann man eine Einhüllende zuordnen. Diese Hüllkurve ist eine gedachte Linie, die Minima bzw. Maxima einer Wellenform verbindet. Wenn man zwei Sinusschwingungen additiv mischt, dann entsteht eine neue Hüllkurve wie in Abbildung 2.1. Die Frequenz der Hüllkurve hängt vom Frequenzunterschied der beiden Sinusschwingungen ab – die Hüllkurvenfrequenz entspricht dem Doppelten des Frequenzunterschiedes. Wenn also die beiden Sinusschwingungen im Frequenzabstand Δf zueinander stehen, dann muss

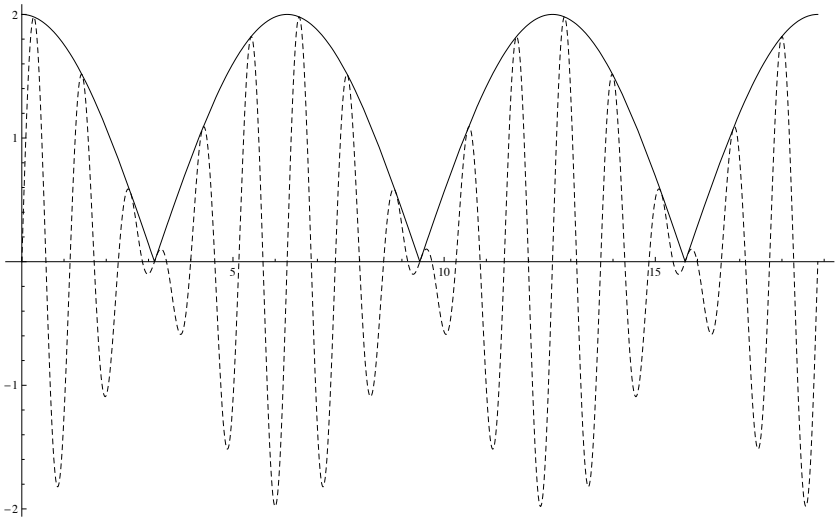

Abbildung 2.1: gestrichelte Kurve: $f(x) = sin(6x) + sin(5x)$; Hüllkurve: $h(x) = 2|cos(2x)|$

$2\Delta f$ als Frequenz im Spektrum der Hüllkurve enthalten sein. In einfachen Fällen wie in Abbildung 2.1 kann man Δf direkt aus dem Graph der Funktion $h(x)$ ablesen. Audiodaten beinhalten jedoch zahlreiche Partialtöne, so dass die Hüllkurve komplexere Formen annimmt.

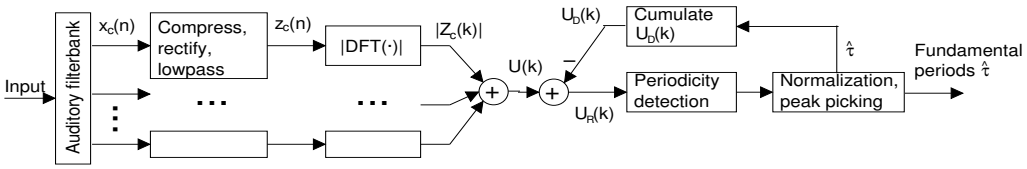

Abbildung 2.2: Das Verfahren im Überblick. Bildquelle: [Kla05]

Abbildung 2.2 zeigt das Blockschaltbild des Verfahrens nach [Kla05] im Überblick. In den folgenden Abschnitten sollen die einzelnen Schritte erläutert werden. Um die Arbeitsweise der einzelnen Schritte anschaulich zu gestalten, soll ein Beispielsignal verarbeitet werden. Das Ausgangssignal ist eine Wellenform mit den Frequenzen 500, 600, 700, 800, 900 und 1000 Hz, die mit gleicher Amplitude auftreten. Die Wellenform ist normalisiert, d.h. die Amplitudenwerte sind im Bereich -1 bis +1. Abbildung 2.3 zeigt das Betragsspektrum des Eingangssignals.

Streng genommen handelt es sich nicht um eine harmonische Obertonreihe. Man müsste die Frequenzen 100, 200, 300 und 400 Hz ergänzen. Es wird sich zeigen, dass für das gewählte Beispielsignal im Frequenzbereich bis 500 Hz „Störungen“ entstehen. Um diese in den graphischen Darstellungen besser hervorzuheben, wurden die tieferen Partialtöne ausgelassen.

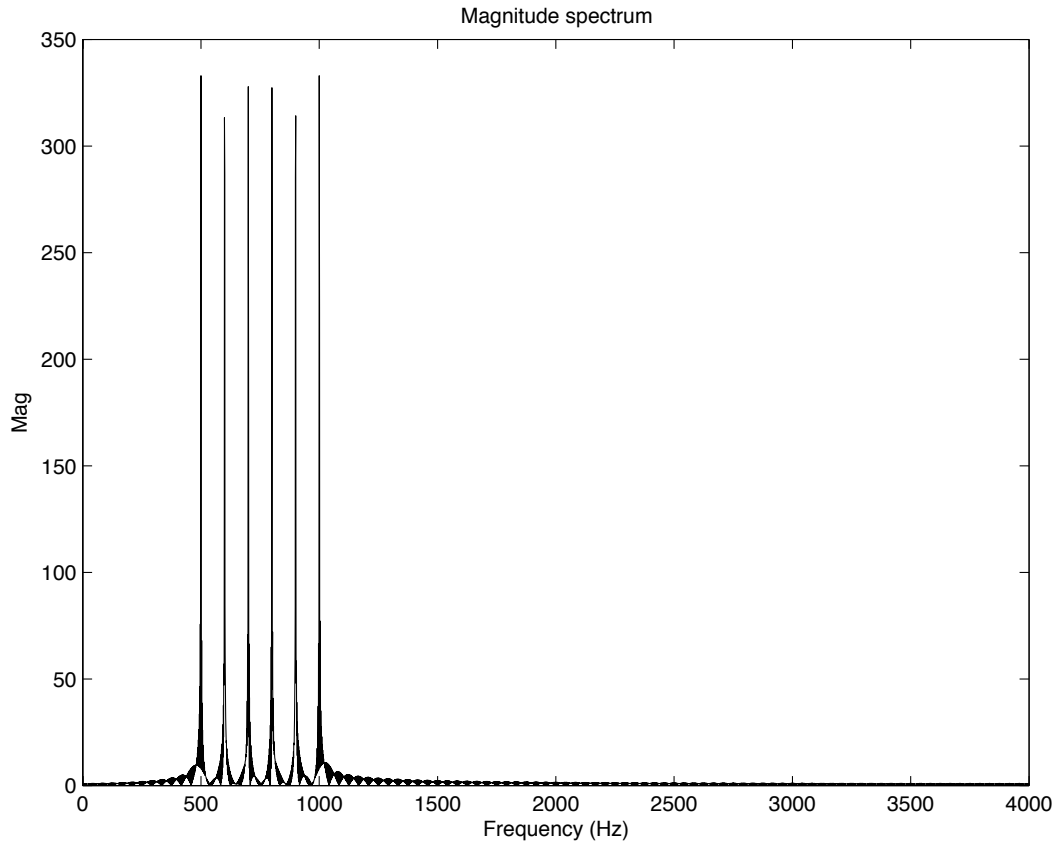

Abbildung 2.3: Betragsspektrum des Beispielsignals

2.1 Modellierung des auditven Filters

Es gibt eine ganze Reihe von Modellen wie auditve Filter funktionieren könnten. Die Daten dafür werden durch Untersuchungen des Hörapparates gewonnen – z.B. durch Tierexperimente, um die Ergebnisse dann auf den Menschen zu übertragen. Im Prinzip handelt es sich jeweils um einen Satz von Bandpassfiltern. Anssi Klapuri verwendet in [Kla05] die Implementation einer Patterson-Holdsworth-Filterbank von Malcolm Slaney [Slaa].

Slaney hat eine Toolbox für Matlab veröffentlicht, die populäre auditve Modelle implementiert [Slab]. Seine Toolbox wird hier verwendet, um die Filterkoeffizienten für die Bandpassfilter zu bestimmen.

Abbildung 2.4 zeigt beispielhaft das Frequenzverhalten einer Filterbank nach Patterson und Holdsworth für fünf Bandpässe. Man kann sehen, dass der Abstand der Mittenfrequenzen exponentiell wächst und es Überlappungen gibt. Der Abstand der Mittenfrequenzen für tiefere Frequenzen ist fast äquidistant und die Kanäle sind eng beieinander. Des Weiteren sind die Filter unterschiedlich breit.

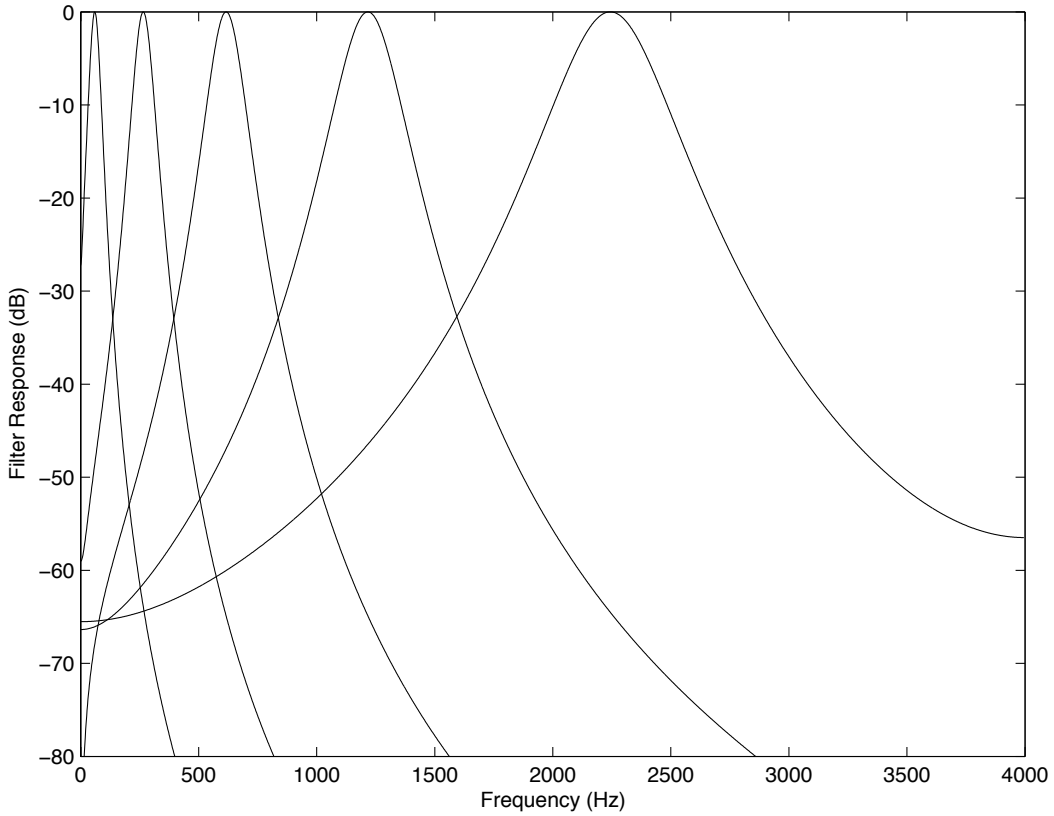

Abbildung 2.4: Frequenzverhalten der auditiven Filter

Das Ergebnis einer auditive Filterung des Beispielsignals ist für einen einzelnen Kanal in Abbildung 2.5 dargestellt. Die zugrunde liegende Filterbank besteht aus 50 Bandpassfiltern. Die Mittenfrequenz für den ausgewählten Kanal beträgt etwa 663 Hz. Mit zunehmendem Abstand von der Mittenfrequenz werden die Frequenzkomponenten abgeschwächt.

2.2 Modellierung des Cochleären Verstärkers

Das auditve System vermag es, sehr leise aber auch vergleichsweise laute Geräusche wahrzunehmen. Begriffe wie „leise“ und „laut“ sind Eigenschaften, die durch die Wahrnehmung bestimmt sind. Zugrunde liegt die psychoakustische Größe Lautheit. Dagegen ist Schalldruck eine physikalische Größe, die wie der Druck als Kraft pro Fläche definiert ist. Zwischen Lautheit und Schalldruck besteht keine Linearität.

Der Dynamikbereich, das Verhältnis zwischen dem kleinsten und größten Schalldruckpegel, beträgt beim menschlichen Ohr ungefähr 120 bis 140 dB. Die kleinste wahrnehmbare Schalldruckänderung, die Hörschwelle, beträgt etwa 20 μPa.

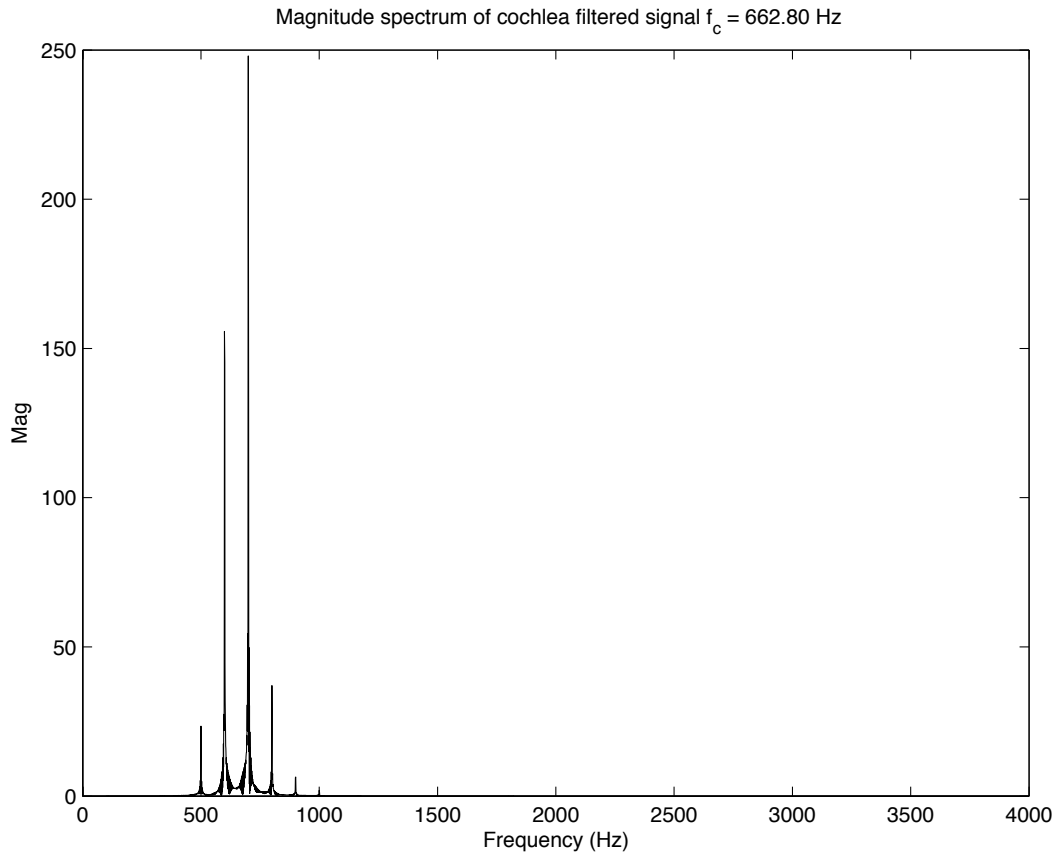

Abbildung 2.5: Spektrum des gefilterten Signals

Bei einem Schalldruck zwischen 20 und 200 Pa ist die Schmerzschwelle erreicht. Dieser Schalldruck ist gegenüber dem der Hörschwelle also um den Faktor eine Million bzw. zehn Millionen größer.

Im Innenohr gibt es eine Art Subsystem, das akustische Signale verstärken kann, sonst würde man leise Geräusche nicht wahrnehmen. Diese Funktion wird den Äußeren Haarzellen zugeschrieben. Sie arbeiten als Cochleärer Verstärker. Die Verstärkung ist nicht-linear. Geräusche mit geringem Schalldruck unterliegen höheren Verstärkungen als Geräusche mit hohem Schalldruck.

Im Modell geht es nicht um die Verstärkung des Signals, sondern um die Einschränkung der Dynamik. Dynamikeinschränkungen können technisch mit Kompressoren umgesetzt werden. In [Kla05] wird die nachstehende Formel verwendet. Der Parameter v ist mit 0.33 belegt.

$$FWC(x) = \begin{cases} x^v, & \text{wenn } x \geq 0 \\ -|x|^v, & \text{wenn } x < 0 \end{cases} \tag{2.1}$$

FWC steht für „full wave compression“. Dabei bedeutet „full wave“, dass

beide Halbwellen komprimiert werden. Die Kompression entspricht hier dem Ziehen der Kubikwurzel. Da negative Zahlen nicht als Radikand benutzt werden können, wird die Wurzel für negative x aus dem Betrag gezogen und das Vorzeichen nachträglich appliziert.

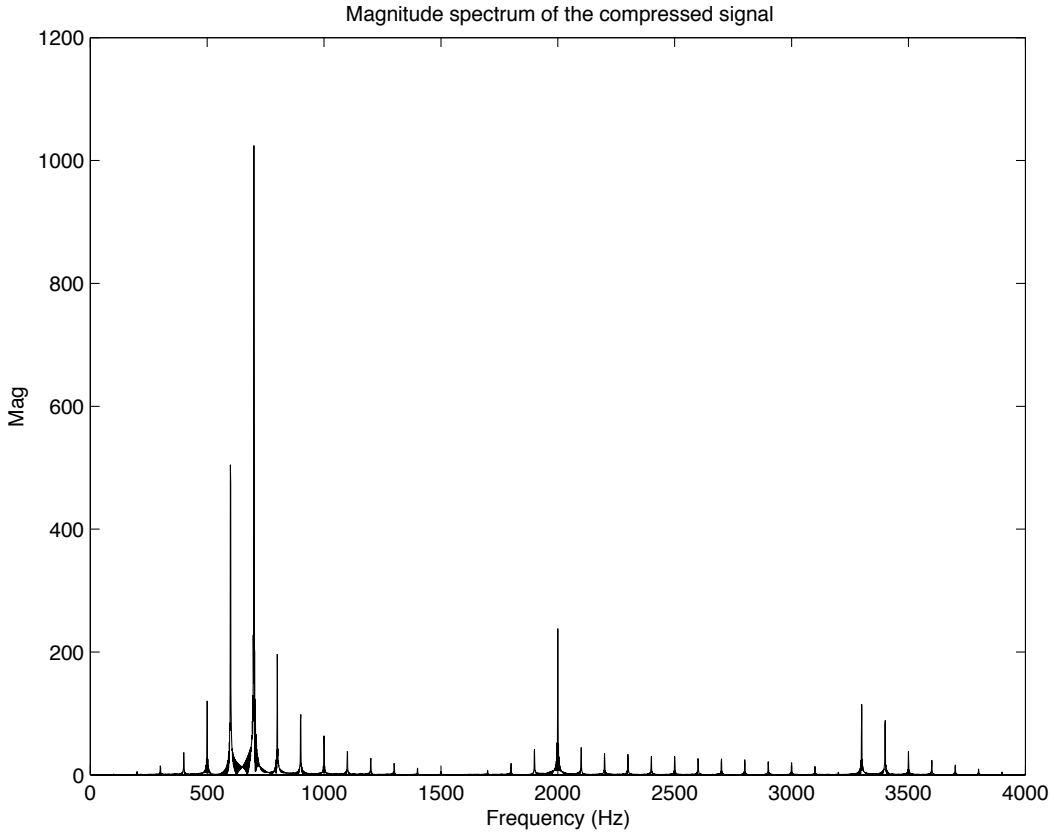

Abbildung 2.6: Spektrum des komprimierten Signals

Abbildung 2.6 zeigt das Spektrum nach der Kompression. Im Vergleich mit Abbildung 2.5 sieht man, dass die Kompression Störungen erzeugt. Die Mittenfrequenz des auditiven Bandpassfilters beträgt hier $f_c = 662.8$ Hz. Die Störungen treten um ungerade Vielfache von f_c herum auf, also beispielsweise etwa um 1988 Hz und 3314 Hz.

2.3 Modellierung der Transduktion

Die Inneren Haarzellen übernehmen die Aufgabe die Transduktion, also die Umwandlung von mechanischen Bewegungen der Basilarmembran in elektrische Signale. Die Funktion der Inneren Haarzellen kann man vereinfacht als Hüllkurvendetektor modellieren.

Die additive Überlagerung zweier Sinusschwingungen entspricht einer Amplitudenmodulation. Das wird aus nachstehender Gleichung sichtbar. Der Übergang von der linken zur rechten Seite der Gleichung ergibt sich direkt aus der Anwendung der Additionstheoreme wie sie im jedem Tafelwerk stehen:

$$sin(2\pi f_1 t) + sin(2\pi f_2 t) = 2 * \underbrace{cos\left(2\pi \frac{f_1 - f_2}{2} t\right)}_{\text{Basisbandsignal}} * \underbrace{sin\left(2\pi \frac{f_1 + f_2}{2} t\right)}_{\text{Trägerschwingung}} \quad (2.2)$$

Im Produkt auf der rechten Seite in Gleichung 2.2 entspricht der Sinus-Term der Trägerschwingung, der Kosinus-Term dem aufmodulierten Basisbandsignal. Das Basisbandsignal entspricht der Hüllkurve und die erhält man wie bei einem einfachen AM-Radiogerät durch Gleichrichtung und Tiefpassfilterung. Die Gleichrichtung, die man in der Elektrotechnik mit einer Diode umsetzen würde, kann man mathematisch wie folgt beschreiben:

$$HWR(x) = \begin{cases} x, & \text{wenn } x \geq 0 \\ 0, & \text{wenn } x < 0 \end{cases} \quad (2.3)$$

Die Gleichrichtung ist ein „Kern-Element“ des Verfahrens. [Kla04, S. 37] Sie ist eine nicht-lineare Operation und erzeugt gewünschte und nicht gewünschte Veränderungen im Spektrum des Signals:

Dazu sei noch einmal auf Abbildung 2.1 verwiesen. Die Abbildung zeigt die Trägerschwingung mit dem aufmodulierten Basisbandsignal. Das Gesamtsignal, also $f(x)$, ist die gestrichelte Linie. Die Hüllkurve $h(x)$ ist als durchgehende Linie dargestellt. Tatsächlich gibt es zwei Hüllkurven – für die untere Halbwelle und für die obere Halbwelle. In Abbildung 2.1 ist lediglich die Hüllkurve der oberen Halbwelle dargestellt. Beide Hüllkurven beinhalten die gleiche Information. Die eine Hüllkurve ergibt sich aus der Spiegelung der anderen Hüllkurve an der x-Achse.

Aber nur, weil das menschliche Auge die Hüllkurve $h(x)$ in Abbildung 2.1 imaginieren kann, bedeutet das jedoch nicht, dass die Frequenz der Hüllkurve auch als Frequenz im Spektrum von $f(x)$ vorkommt. Die Frequenz, die die Hüllkurve ausmacht, soll aber ermittelt werden. Wäre die Frequenz der Hüllkurve f_H im Spektrum der Wellenform von $f(x)$, so würde es genügen, das Signal $f(x)$ so zu filtern, dass alle Frequenzen oberhalb von f_H herausgefiltert werden. Da aber die Frequenz der Hüllkurve f_H sich nicht im Spektrum der Wellenform von $f(x)$ befindet, muss die Frequenz der Hüllkurve in das Spektrum hineinge-

bracht werden. Die Gleichrichtung der Wellenform $f(x)$ soll das aber bewirken – sie induziert das Spektrum der Hüllkurve in das Spektrum des Signals $f(x)$.

Die im vorherigen Absatz aufgestellt Behauptung, dass sich die Frequenz der Hüllkurve nicht im Spektrum von $f(x)$ befindet, kann man aber auch anders betrachten.

Angenommen, wir haben eine Sinusschwingung mit einer Frequenz von 23 Hz in der Zeit-Domäne. Wenn man das Signal Fourier-analysiert, also in die Frequenz-Domäne überführt, erhält man ein Spektrum, bei dem sich für die Frequenz von 23 Hz ein Peak ergibt – wie man das eben erwartet. Das Spektrum kann man nun aber auch anders betrachten, und zwar so, dass auch alle anderen Frequenzen im Spektrum auftreten, nur eben mit dem Betrag null. Die Komponente beispielsweise mit der Frequenz 42 Hz, die zu dem Spektrum (nicht) beiträgt, könnte man so beschreiben:

$$f_{nil}(x) = 0 = 0 * sin(2\pi 42x) = sin(2\pi 42x) - sin(2\pi 42x) \tag{2.4}$$

Nach Gleichung 2.4 tritt die Schwingung mit der Frequenz 42 Hz mit der Amplitude null auf – oder arithmetisch equivalent als Mischung aus Schwingung und Gegenschwingung.

Eingangs wurde beschrieben, dass es zwei Hüllkurven gibt, also für die obere bzw. untere Halbwelle, jeweils mit der Hüllkurvenfrequenz f_H. Die Hüllkurven kann man als Spiegelbilder betrachten mit der x-Achse als Spiegelachse. Ordnet man der oberen Hüllkurve die Funktion $h(x) = sin(2\pi f_H)$ zu, muss man der unteren Hüllkurve, da sie ja spiegelbildlich ist, die Funktion $h'(x) = -sin(2\pi f_H)$ zuordnen.

Die Hüllkurvenfrequenz f_H kommt also doch in $f(x)$ vor. Messbar ist sie allerdings erst, wenn man den Gegenschall entfernt, z.B. indem man eine der Halbwellen anschneidet. Die Gleichrichtung bewirkt dies. Dadurch, dass die untere Hüllkurve $h'(x) = -sin(2\pi f_H)$ abgeschnitten ist und nur die obere Hüllkurve in der Zeit-Domäne übrig bleibt, ist die obere Hüllkurve auch im Spektrum messbar. Das Spektrum der oberen Hüllkurve und das Spektrum des Signals überlagern sich im Ergebnis.

Bei echten Audiodaten gibt es nicht nur eine einzige Frequenz im Spektrum der Hüllkurve, sondern es überlagern sich zahlreiche Frequenzen. Alle diese niederfrequenten Schwingungen werden entsprechend der vorangestellten Überlegungen in das Spektrum induziert.

Abbildung 2.7 zeigt das Ergebnis der Gleichrichtung. Bei 100 Hz sieht man im Spektrum einen Peak. Dieser kennzeichnet einen der Oberton-Abstände im Spektrum des Ausgangssignals. Bei 0 Hz gibt es ebenfalls einen Peak mit einem Betrag von über 800, den man allerdings nicht sieht, da er genau auf der y-Achse liegt. Es ist zu beachten, dass ein Peak bei 0 Hz auch in den folgenden Graphen stets vorhanden ist.

Durch die Gleichrichtung entstehen „Ecken“ bei jedem Nulldurchgang der Wellenform, die sich im Spektrum als Störungen bemerkbar machen – so z.B. der Peak bei 1300 Hz, der betragsmäßig größer ist als der Peak bei der gleichen Frequenz in Abbildung 2.6.

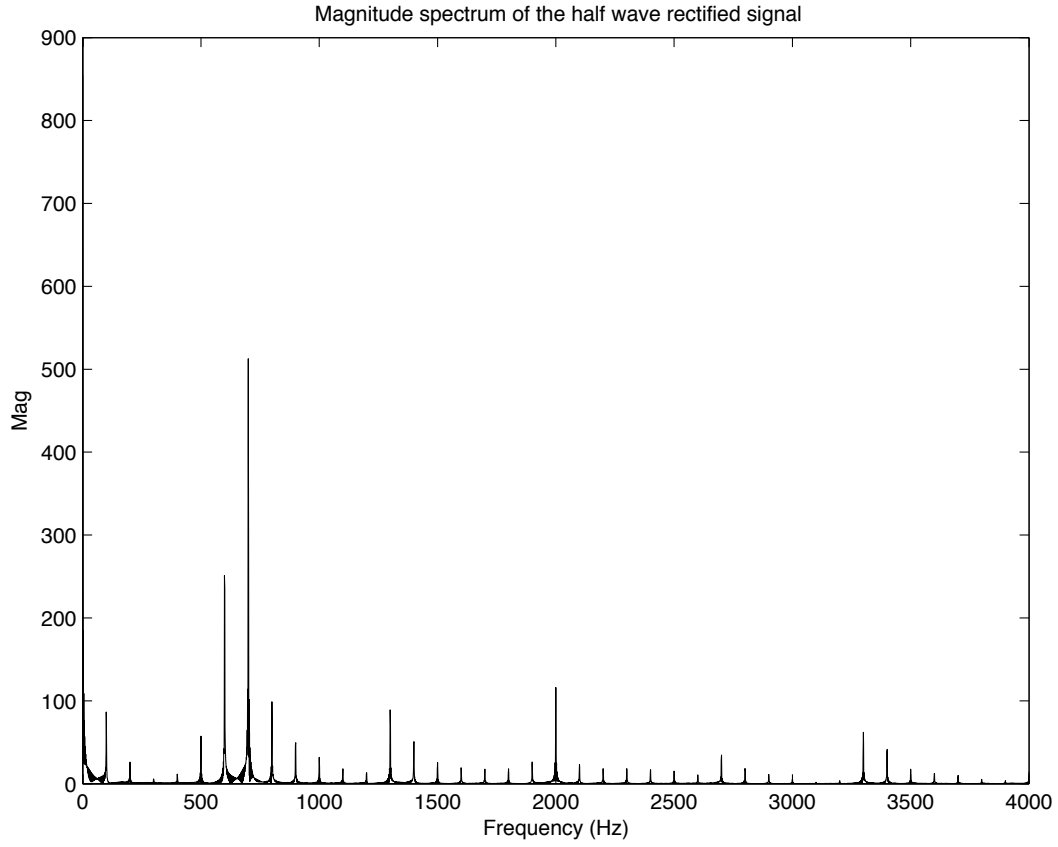

Abbildung 2.7: Spektrum des gleichgerichteten Signals

Anschliessend wird das Signal tiefpassgefiltert, so dass Frequenzkomponenten oberhalb von 1 kHz unterdrückt werden. Siehe Abbildung 2.8.

Die Wellenformen der komprimierten, gleichgerichteten, tiefpassgefilterten auditiven Kanäle $z_c(n)$ werden anschliessend mittels Diskreter Fourier-Transformation in die Frequenzdomäne überführt, so dass für jeden Kanal c ein Betragsspektrum $|Z_c(k)|$ wie in Abbildung 2.9 entsteht.

$$Z_c(k) = DFT(z_c(n)) \tag{2.5}$$

Die Spektren der einzelnen Kanäle werden zu einem Summenspektrum $U(k)$ zusammenaddiert:

$$U(k) = \sum_c |Z_c(k)| \tag{2.6}$$

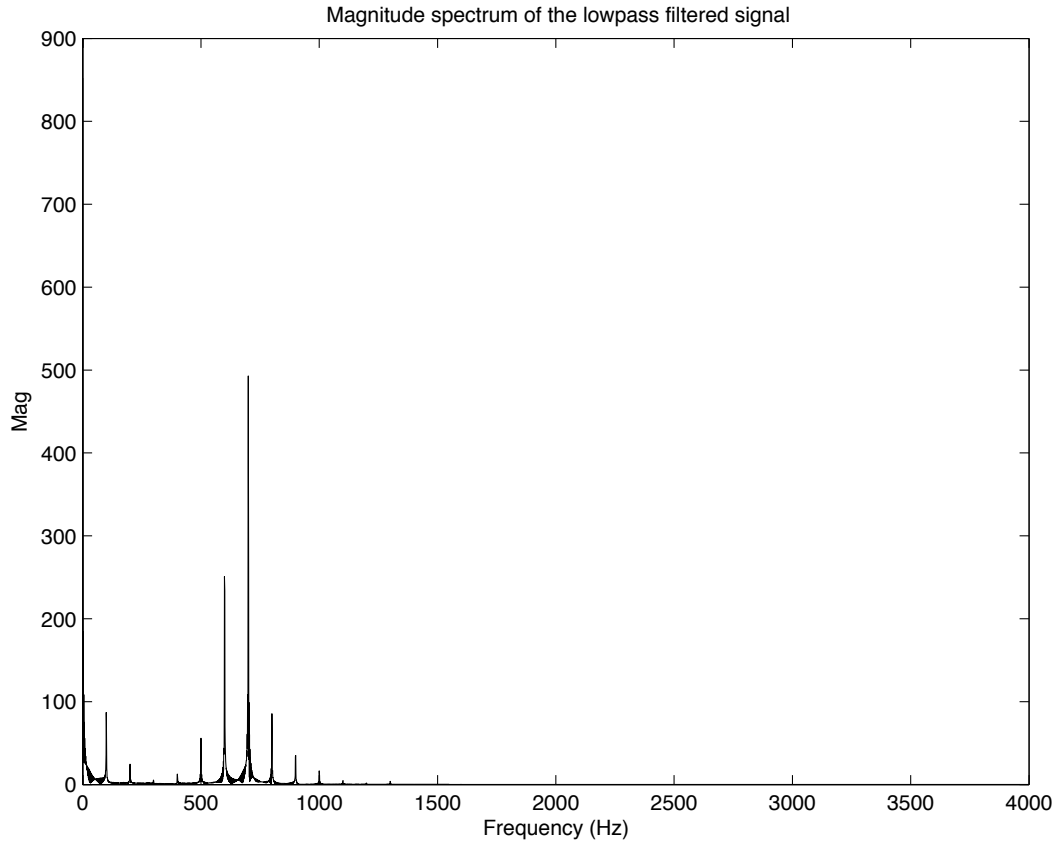

Abbildung 2.8: Spektrum des tiefpassgefilterten Signals

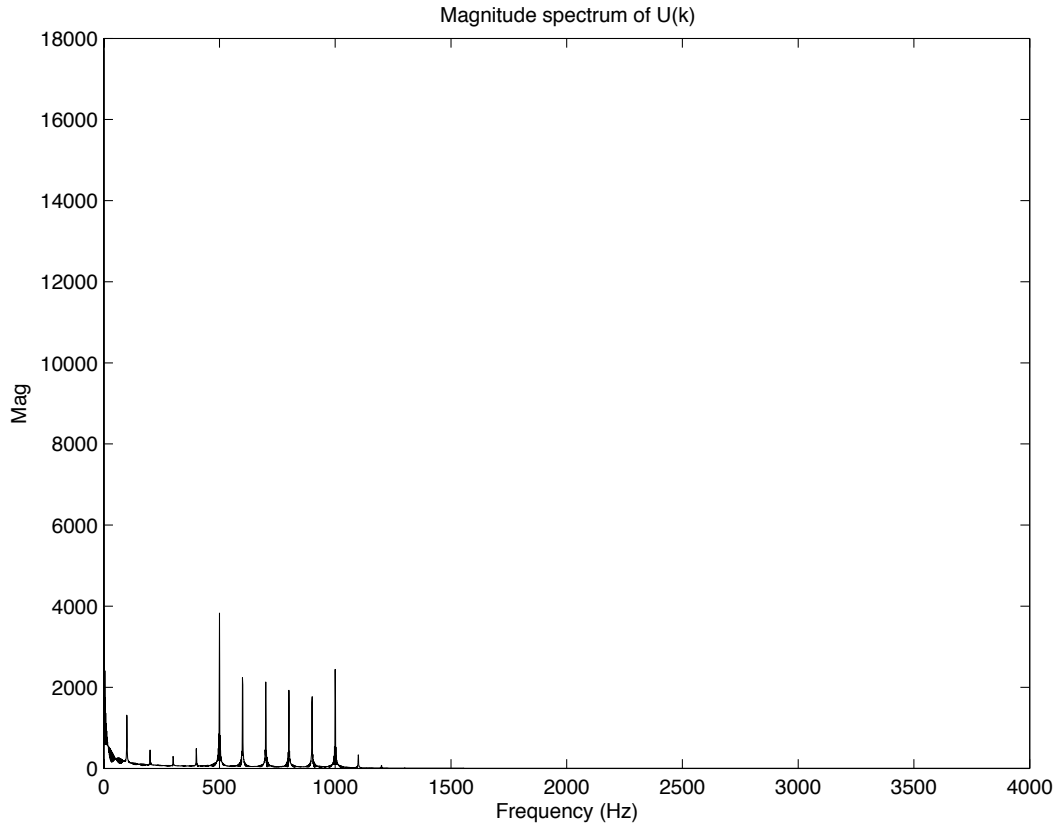

Abbildung 2.9: Summenspektrum $U(k)$

Abbildung 2.9 zeigt so ein Summenspektrum. Es fällt die Besonderheit auf, dass der Peak für 500 Hz größer ist als die anderen Peaks, abgesehen vom Peak bei 0 Hz. Im nächsten Schritt erfolgt die Periodizitätsanalyse, für die dieser Peak bei 500 Hz eine tragende Rolle spielt, denn er entspricht der Frequenz f_0 des Grundtons.

2.4 Periodizitätsanalyse

Die Periodizitätsanalyse findet in der Frequenz-Domäne statt. Die Analyse operiert auf dem diskreten Summenspektrum $U(k)$. Das Summenspektrum kann man als Vektor mit K Komponenten auffassen. Eine Komponente $U(k)$ mit $k = 0, \ldots, K-1$ steht dabei nicht für eine einzelne Frequenz, sondern für einen schmalen Frequenzbereich. Für jedes k gibt $U(k)$ an, wie stark die Komponenten zum Spektrum beiträgt.

Für die Periodizitätsanalyse relevant sind dabei nicht alle Frequenzen, sondern nur der niederfrequente Teil des Spektrums, beispielsweise bis 1 kHz. In diesem Frequenzbereich befinden sich die die kleineren Differenzfrequenzen aus den Hüllkurven der auditiven Kanäle.

Da nur ein kleiner Teil des Spektrums die gesuchte Information beinhaltet, kann man die höherfrequenten Anteile im Spektrum unterdrücken. Das ist für die polyphone Transkription im Zusammenhang mit dem vorgestellten Algorithmus notwendig, denn der Algorithmus sucht Stellen im Spektrum heraus, die ein Maximum sind. Überlagerungseffekte, genauer konstruktive Interferenzen von nicht harmonischen Partialtönen, können im Spektrum Maxima erzeugen, die aber nicht als Grundton in Frage kommen.

$$H_{LP}(k) \quad = \frac{1}{0.108 f_s k/K + 24.7} \tag{2.7}$$

Gleichung 2.7 repräsentiert eine Gewichtung der Frequenzen. Der Graph für $H_{LP}(k)$ ist in Abbildung 2.10 zu sehen. Wie man daraus ablesen kann, ist H_{LP} monoton fallend. Technisch betrachtet, ist H_{LP} ein Tiefpassfilter in der Frequenz-Domäne, allerdings steht weniger die Filterung im Vordergrund, sondern die Gewichtung in Abhängigkeit von der Äquivalentrechteck-Bandbreite[1].

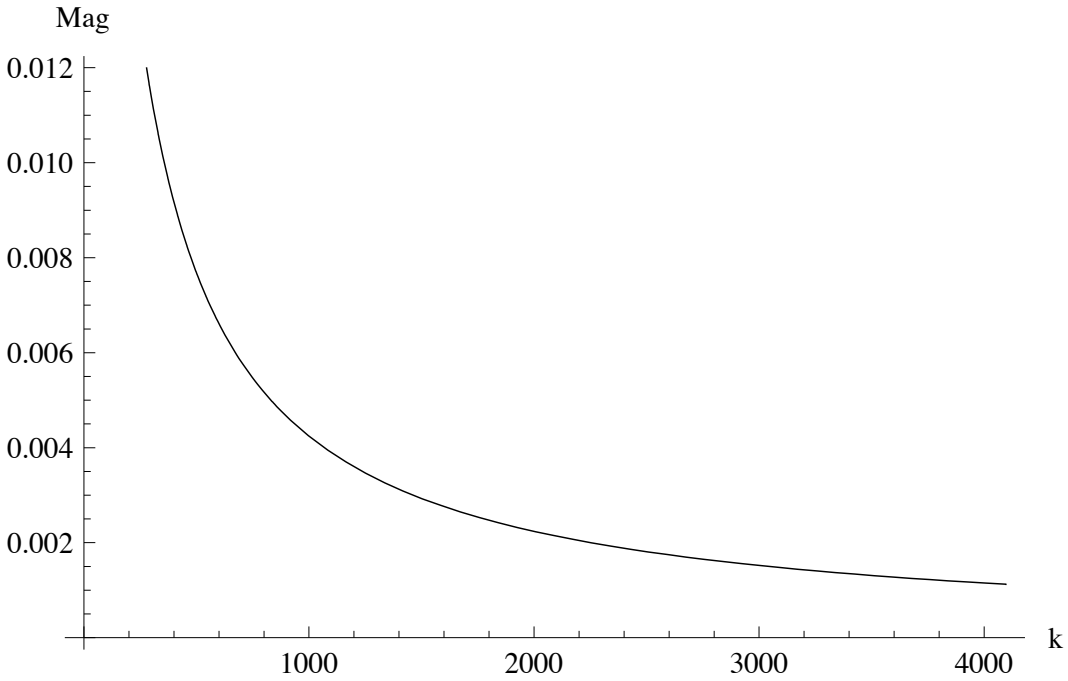

Abbildung 2.10: Graph der Funktion $H_{LP}(k)$ mit $K = 4096$

[1]Wikipedia-Artikel zur ERB-Skala, `http://de.wikipedia.org/w/index.php?title=ERB-Skala&oldid=50194592`, abgerufen am 31. August 2008

Das Summenspektrum hat K Komponenten. $H_{LP}(k)$ wird komponentenweise mit dem Summenspektrum $U(k)$ multipliziert. Das dadurch entstehende Spektrum ist in Abbildung 2.11 dargestellt. In Abbildung 2.11 sieht man für Vielfache von 100 Hz kleine Peaks.

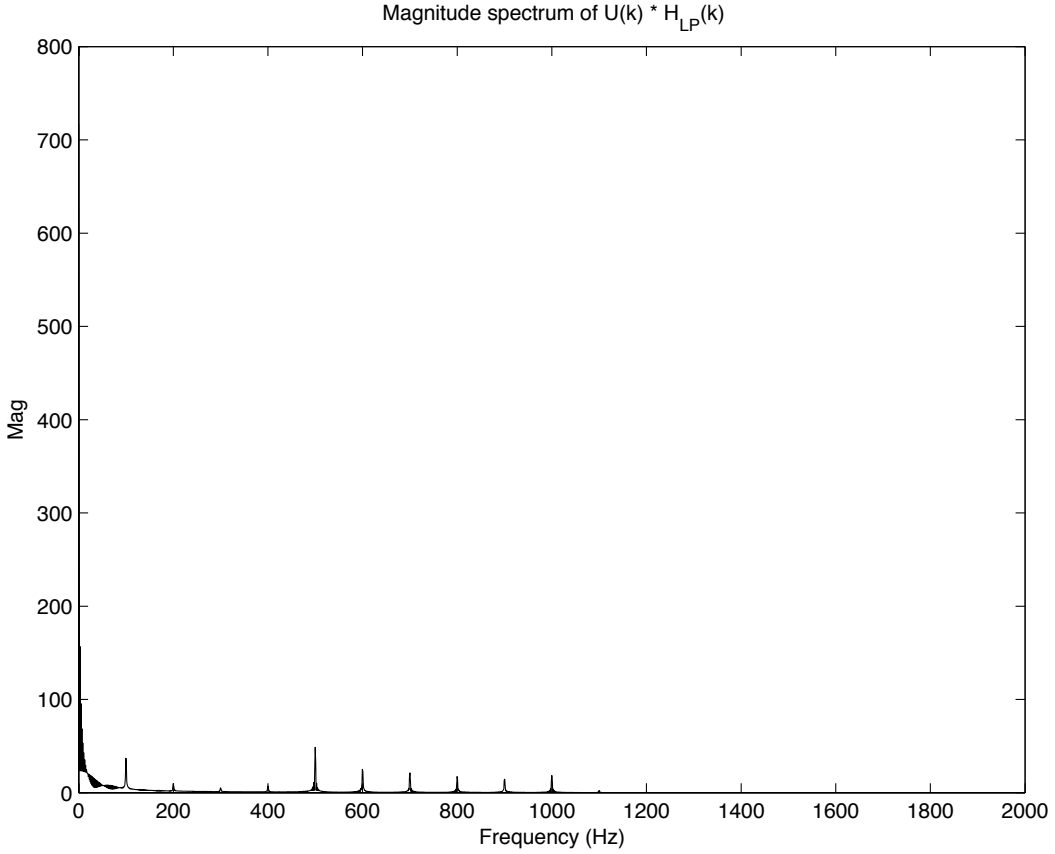

Abbildung 2.11: $H_{LP}(k) * U(k)$

Eine Funktion die den wesentlichen Teil der Suche ausmacht, ist in Gleichung 2.8 wiedergegeben. Sie gibt eine Gewichtung an, ähnlich einer Wahrscheinlichkeit, dass ein f_0-Kandidat auch Grundton ist. Der f_0-Kandidat, für den die Gewichtung ermittelt werden soll, wird nicht als Frequenzparameter an die Funktion λ übergeben, sondern als Schwingungsperiode τ mit $\tau = f_s/f$. Die Variable f_s bezeichnet die Abtastrate.

$$\lambda(\tau) = \frac{f_s}{\tau} \sum_{p=1}^{\tau/2} \max_{k \in \kappa_{p,\tau}} \left(H_{LP}(k) * U(k) \right) \tag{2.8}$$

Die Suche nach dem Maximum erfolgt in Gleichung 2.8 nicht überall im Spektrum, sondern nur in schmalen Bereichen um Stellen herum, an denen sich harmonische Obertöne befinden könnten. Diese schmalen Bereiche um den p-ten Oberton herum sind durch die Intervallgrenzen $k^A_{p,\tau}$ und $k^B_{p,\tau}$ bestimmt:

$$k^A_{p,\tau} = \frac{pK}{\tau + \Delta\tau/2} + 1 \tag{2.9}$$

$$k^B_{p,\tau} = \max\left(\frac{pK}{\tau - \Delta\tau/2}, k^A_{p,\tau}\right) \tag{2.10}$$

Der Term f_s/τ in Gleichung 2.8 gehört zur Gewichtung $H_{LP}(k)$ und ist lediglich ausgeklammert.

Die Berechnung für ein einzelnes $\lambda(\tau)$ ist nachstehend als Beispielrechnung angegeben[2]. Das Beispiel soll verdeutlichen, dass Maxima in spektralen Bereichen möglicher harmonischer Obertöne gesucht und zu einer Gesamtgewichtung aufaddiert werden.

$$\begin{aligned}
\tau &= 16 && \text{// entspricht 500 Hz}\\
K &= 4096\\
f_s &= 8000\\
\lambda(16) &= \tfrac{f_s}{16} * (\max_{k\in[249\ldots264]} (H_{LP}(k) * U(k)) + && \text{// 243 \ldots 257 Hz}\\
&\quad \max_{k\in[497\ldots529]} (H_{LP}(k) * U(k)) + && \text{// 485 \ldots 517 Hz}\\
&\quad \max_{k\in[746\ldots793]} (H_{LP}(k) * U(k)) + && \text{// 728 \ldots 774 Hz}\\
&\quad \max_{k\in[994\ldots1057]} (H_{LP}(k) * U(k)) + && \text{// 971 \ldots 1032 Hz}\\
&\quad \max_{k\in[1242\ldots1321]} (H_{LP}(k) * U(k)) + && \text{// 1213 \ldots 1290 Hz}\\
&\quad \max_{k\in[1490\ldots1586]} (H_{LP}(k) * U(k)) + && \text{// 1455 \ldots 1549 Hz}\\
&\quad \max_{k\in[1739\ldots1850]} (H_{LP}(k) * U(k)) + && \text{// 1698 \ldots 1807 Hz}\\
&\quad \max_{k\in[1987\ldots2114]} (H_{LP}(k) * U(k))) && \text{// 1940 \ldots 2064 Hz}
\end{aligned}$$

Die Berechnung erfolgt analog für alle τ im Bereich von 1 bis f_s/f_{min}, entsprechend einem Frequenzbereich von f_{min} bis zur Abtastrate[3] f_s. Eine Darstellung des Graphen von $\lambda(\tau)$ für das Beispielspektrum ist Abbildung 2.12 zu sehen.

[2] Die Einheiten wurden weggelassen.

[3] Es würde tatsächlich auch die Nyquist-Frequenz $f_s/2$ als Grenzfrequenz genügen, was einem τ von zwei entspricht.

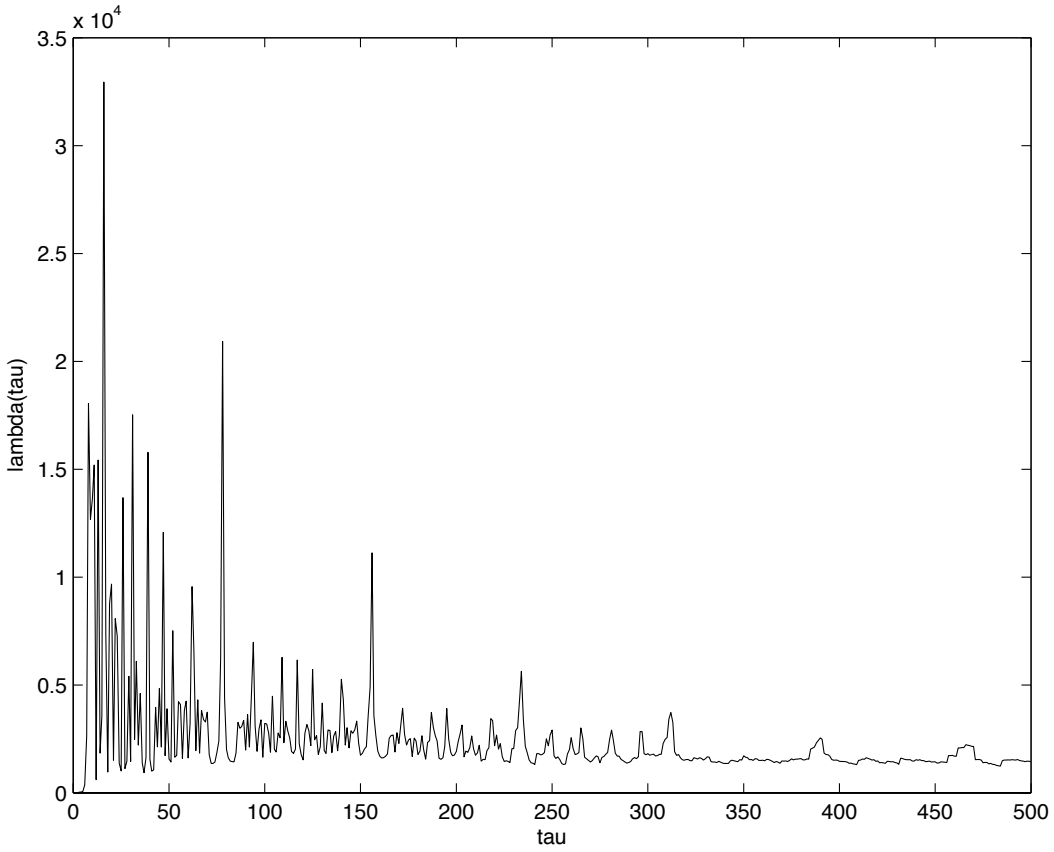

Abbildung 2.12: Graph von $\lambda(\tau)$

2.5 Normalisierung

Anschließend werden die Werte $\lambda(\tau)$ in Abhängigkeit von der korrespondierenden Frequenz ($f = f_s/\tau$) gewichtet, um weder zu hohe noch zu tiefe f_0-Frequenzen zu ermitteln. Als geeigneten Wert für b gibt Klapuri -0.04 an.

$$\Lambda(\tau) = \left(1 + b * ln\left(\frac{f_s}{\tau}\right)\right) * \lambda(\tau) \tag{2.11}$$

Abbildung 2.13 zeigt beispielhaft den Graphen für $\Lambda(\tau)$.

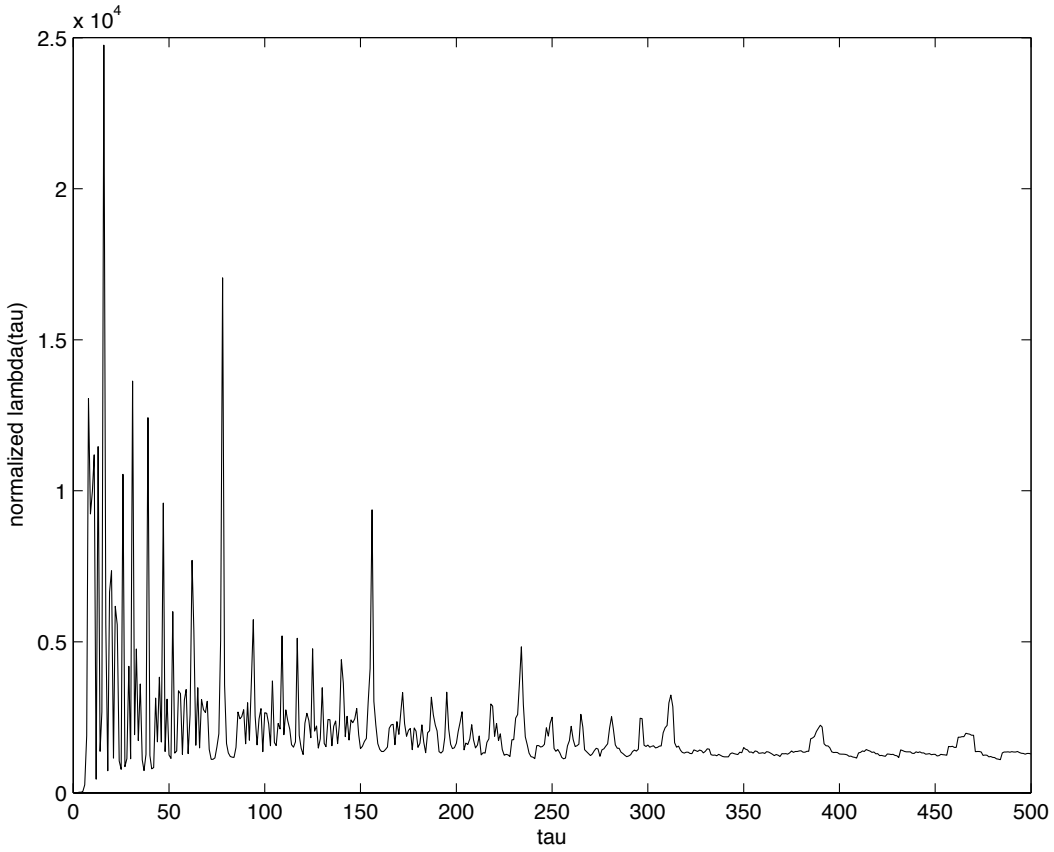

Abbildung 2.13: Graph von $\Lambda(\tau)$

2.6 Bestimmung von f_0

Um aus $\Lambda(\tau)$ den Grundton zu ermitteln, muss lediglich die Stelle τ gesucht werden, für die $\Lambda(\tau)$ maximal ist. Da τ eine Schwingungsdauer darstellt, muss τ ferner in eine Frequenz umgerechnet werden:

$$f_0 = \frac{f_s}{\text{argmax}(\Lambda(\tau))} \tag{2.12}$$

Das Maximum für $\Lambda(\tau)$ tritt an der Stelle $\tau = 16$ auf. Unter Anwendung von Gleichung 2.12 ergibt sich die Frequenz des Grundtons mit 500 Hz.

3 Abschätzung mehrerer Grundfrequenzen

Im letzten Kapitel wurde gezeigt, wie man aus einem Spektrum mit Harmonischen den Grundton f_0 ermitteln kann. Aufbauend auf dieser Idee, konstruiert Klapuri eine Erweiterung, die es ermöglicht, das Verfahren auf polyphone Audiodaten anwenden zu können. Dieses Verfahren arbeitet iterativ. In einer Programmschleife ermittelt es einen Grundton f_0, errechnet dazu ein passendes Obertonspektrum, entfernt es aus dem Gesamtspektrum und führt das übrig bleibende Spektrum erneut der Auswertung zu. Der Algorithmus im Pseudocode gestaltet sich wie folgt:

Algorithmus 1 Abschätzung der Grundtöne und deren Stärke

Require: $U, max\_iterarations$
1: $K \Leftarrow |U|$
2: $U_R \Leftarrow U$ // Restspektrum
3: $U_D \Leftarrow [0, \ldots, 0]$ // erkanntes Spektrum
4: $d \Leftarrow 0.5$
5: $F \Leftarrow [0, \ldots, 0]$ // erkannte Grundtöne
6: $S \Leftarrow [0, \ldots, 0]$ // deren Stärke
7: **for** $i = 1$ to $max\_iterarations$ **do**
8: $[f_0, s] = calc\_f_0(U_R)$
9: $U_{cancel} = calc\_cancelation\_spectrum(f_0, U_R)$
10: **for** $k = 1$ to K **do**
11: $U_D(k) = U_D(k) - U_{cancel}(k)$
12: $U_R(k) = max(0, U(k) - d * U_D(k)$
13: **end for**
14: $F[i] \Leftarrow f_0$
15: $S[i] \Leftarrow s$
16: **end for**
17: $return\ [F, S]$

Im Algorithmus wird die Funktion *calc_cancelation_spectrum*() benutzt, um das Obertonspektrum U_{cancel} passend zum Grundton zu berechnen. Die Implementation der Funktion hat große Ähnlichkeit mit der Funktion, die Gleichung 2.8 realisiert. Nur werden keine Maxima zu einem $\lambda(\tau)$ aufaddiert. sondern die Peaks im Restspektrum an den Stellen zwischen $k^A_{p,\tau}$ und $k^B_{p,\tau}$ werden in das Spektrum U_{cancel} übertragen. Die übertragenen Peaks werden gewichtet – mit $f_s/\tau * H_{LP}(k)$.

4 Implementation und Testlauf

Um mit dem vorgestellten Ansatz polyphone Musik transkribieren zu können, ist nun lediglich eine weitere Erweiterung notwendig. Es fehlt noch eine Schleife, die die diskrete Wellenform in einzelne Frames zerlegt und für jedes Frame den im letzten Kapitel beschriebenen Algorithmus anwendet. Als geeignete Länge empfiehlt Klapuri 93 ms lange Frames, um die Fehlerrate gering zu halten. Diese Zahl wurde experimentell ermittelt. [Kla05]

Klapuri vergleicht die Fehlerraten seines Verfahrens in Abhängigkeit vom Grad der Polyphonie und der Frame-Länge mit einer seiner früheren Arbeiten und mit einem Verfahren der Autoren Tolonen und Karjalainen. [Kla05] Er kommt zu dem Ergebnis, dass die Fehlerraten geringer sind als bei den anderen Verfahren. Ferner hängt die Fehlerrate stark vom Grad der Polyphonie ab.

Ein fairer Vergleich gestaltet sich aber schwierig, da die Wahl der Versuchsbedingungen einen wesentlichen Einfluss auf die Qualität der einzelnen Verfahren hat. Eine Evaluation der Qualität soll in dieser Ausarbeitung deshalb nicht erst versucht werden, da dafür das gesteckte Zeitbudget nicht ausreichen würde. Stattdessen soll das Verfahren umgesetzt und damit experimentiert werden.

Artist	**Album**	**Titel**
Yann Tiersen	Le fabuleux destin d'Amélie Poulain	Comptine d'un autre été: l'Apres Midi [Tie]
Apocalyptica	Inquisition Symphony	Nothing else matters
Balanescu-Quartett	Possessed	Autobahn
Balanescu-Quartett	Possessed	Das Model
Carlos Santana	-	Samba Pa Ti
Christian Morgenstern	Hawaii Blue	Outro eins
Metrovavan	Retrofitting	Astronomical Twilight
J. S. Bach / Scottish Chamber Orchestra	-	Air 'on the G string' (BWV1068)
Boards of Canada	Music Has The Right To Children	Kaini Industries

Der Code für das Transkriptionssystem ist in Matlab implementiert. Mit dem Matlab-Code wurden dann die in der Tabelle aufgelisteten Titel transkribiert. Das Matlab-Programm verarbeitet die Audiodaten, die als WAV-Datei vorliegen müssen und generiert eine Textdatei in der die Grundtöne als MIDI-Notennummern und deren Stärke pro Frame gespeichert sind.

Die gewonnen Daten sollen nun weiterverarbeitet werden. Doch da trennen sich algorithmische Welten auf, denn die Art der Weiterverarbeitung hängt von Ziel der Transkription ab. Um das zu verdeutlichen, seien ein paar Problemfelder genannt.

Wenn das Ziel der Transkription die Erstellung einer Notenschrift ist wie z.B. in [Tie], dann sind Fragen nach der Tonart relevant, nach dem ob eine Note eine Viertel- oder eine Achtelnote ist und welcher Takt vorliegt, ob Noten betont sind und dergleichen. Auch für eine Transkription in eine MIDI-Datei sind diese Fragen z.T. relevant. Dafür bedarf es einer Analyse der gewonnenen Daten. Und zwar derart, dass für einen Grundton über die transkribierten Frames hinweg verfolgt wird, inwiefern es sich um die Phasen Attack, Decay, Sustain oder Release (ADSR) handelt bzw. in einer Vereinfachung um Attack und Release. In [RK05] werden für die Erkennung von Notenanschlägen und von Pausen zusätzliche Modelle bemüht.

Bei einem Software-System wie dem Melodyne Plugin 2 ist ein Mechanismus möglicherweise nicht zwangsläufig relevant, der diese Phasen erkennt. Denn setzt man voraus, das es eine separierende Komponente gibt, die bestehendes Soundmaterial auseinander rechnet, so ist in dem getrennten Soundmaterial auch die Information über die ADSR-Phasen enthalten. Somit vereinfacht sich sogar das weitere Verfahren.

Hier soll es lediglich darum gehen, die transkribierten Stücke anschliessend wieder in Audiodateien zurückzuwandeln, damit man sich das Ergebnis anhören kann. In der Synthese gibt es kein Obertonspektrum. Alle Töne sind durch Sinus-Oszillatoren beschrieben, die in der Frequenz des Grundtones schwingen. Dadurch klingt das synthetisierte Stück dumpf. Durch die nicht modellierten Anschläge mangelt es bisweilen auch an klanglicher Klarheit. Das menschliche Gehör ist aber durchaus in der Lage, darin falsche Töne zu erkennen.

Für diese Art der Synthese ist es nicht notwendig, die ADSR-Parameter zu kennen. Die Phasen Attack und Release werden über die gemessene Stärke des Grundtons gesteuert. Dabei ist es nicht wesentlich, entscheiden zu können, ob ein Grundton tatsächlich präsent ist oder nicht. Die Stärke des Grundtons

wirkt sich einfach auf dessen Amplitude in der Synthese aus. Ein Perl-Skript führt die Synthese durch[1]. Die synthetisierten Soundbeispiele wurden anschliessend in das MP3-Format umgewandelt und unter der folgenden Adresse publiziert:

`http://www.schwingungsebene.de/polymt/sound-examples/`

Beim Anhören der Soundbeispiele stellt man fest, dass das Verfahren erstaunlich robust ist. So zeigt die Transkription von Air 'on the G string' in einer Orchester-Version zwar Schwächen in den höheren Tönen, die bisweilen etwas „wackelig“ erscheinen, die Töne sind aber bei holistischem Hören nachvollziehbar. Diese „wackeligen“ hohen Töne deuten darauf hin, dass sie über die Frames hinweg mitunter nicht erkannt werden. In ähnlicher Form tritt das Problem auch bei den anderen Titeln auf. Möglicherweise kann man das reparieren, indem man die Cut-off-Frequenz des Tiefpassfilters nach der Gleichrichtung anpasst.

Beim Anhören der Soundbeispiele über Kopfhörer bemerkt man (möglicherweise) tiefe, kratzige Geräusche. Wie dieses zustande kommen, konnte nicht vollständig geklärt werden.

[1] Das Generieren größerer Mengen Audiodaten mittels Matlab hat sich als problematisch erwiesen. Die Java-VM ist dann mehrfach mit Ausnahmefehlern (und zwar NullPointerExceptions) abgestürzt. Leider gibt es keinen eleganten Mechanismus, nicht die gesamten Daten der Wellenform im Speicher zu halten, ausser man bindet externe Funktionalität mittels C-Bibliotheken ein.
Deshalb wurde zunächst ein Perl-Skript geschrieben, das die Textdatei mit den Transkriptionsdaten verarbeitet und ein ChucK-Skript generiert und mittels Aufruf der ChucK-Umgebung abspielt. Die benutzte Version von ChucK 1.2.0.8 (dracula) erlaubt noch kein Lesen und Schreiben von Dateien, jedoch besteht die Möglichkeit, parallel zum Abspielen eine Art Harddisk-Recorder zu verwenden. Leider ist das nicht ganz frei von Problemen. Wenn die CPU-Geschwindigkeit nicht ausreicht, um die Audiodaten in „Echtzeit“ abzuspielen, kommt es zu Aussetzern, die sich in der Aufnahme bemerkbar machen. Das ChucK-Skript benutzt für jede MIDI-Note einen Sinus-Oszillator, für den die Amplitude entsprechend der Daten aus dem Transkriptionsvorgang gesteuert wird. Das Perl-Skript nimmt noch ein paar Korrekturen vor: Zu laute Töne werden auf eine Maximallautstärke zurückgestellt und im ChucK-Skript werden nicht alle Oszillatoren aktiviert, um Rechenzeit zu sparen. Leider hat sich auch ChucK an dieser Stelle als problematisch erwiesen. Bei einem größeren zweidimensionalem Feld mit etwa 6000 Zeilen für jedes Frame und 128 Spalten meldete ChucK einen Fehler „memory exhausted“, wahrscheinlich weil nicht genug Speicher für den Syntaxbaum verfügbar ist. Bei geeigneter Speicherung würde das Feld im RAM nicht mehr als drei MB belegen. Eine Recherche, wie man einem ChucK-Prozess mehr Speicher zuweisen kann, verlief ergebnislos.
Das Perl-Skript wurde daraufhin so modifiziert, dass die komplette Synthese in Perl erfolgt. Das ist leider extrem langsam. Eine Implementation in der Sprache C, vielleicht sogar mit vorberechneten Sinuswerten, wäre voraussichtlich wenigstens um den Faktor 200 schneller.

Literatur

[Bor08] John Borland. "Digital Sound Separator. New software can modify the individual notes of a recorded chord". In: *Technology Review* (16. Apr. 2008). URL: http://www.technologyreview.com/printer_friendly_article.aspx?id=20606&channel=infotech&section= (besucht am 05. 08. 2008).

[Cel08] Celemony Software GmbH. *Demo-Video zu Direct Note Access*. 2008. URL: http://assets.celemony.com/de/mac/Vorschau_DNA_mac_large.mov (besucht am 05. 08. 2008).

[Hil08] Ulrich Hilgefort. "Sensation auf der Musikmesse: Wave-Daten wie MIDI verändern [Update]". In: *Heise-Newsticker* (2008). URL: http://www.heise.de/newsticker/meldung/104962 (besucht am 05. 08. 2008).

[Kla04] Anssi Klapuri. "Signal processing methods for the automatic transcription of music". Diss. Tampere University of Technology, 2004. URL: http://citeseer.ist.psu.edu/klapuri04signal.html (besucht am 05. 08. 2008).

[Kla05] Anssi Klapuri. "A Perceptually Motivated Multiple-F0 Estimation Method". In: *in Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 2005. S. 291–294. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.74.1152 (besucht am 05. 08. 2008).

[RK05] Matti P. Ryynänen und Anssi Klapuri. "Polyphonic music transcription using note event modeling". In: *in Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. 2005. URL: http://www.cs.tut.fi/~mryynane/ryynklapuri_polytrans_final.pdf (besucht am 05. 08. 2008).

[Slaa] Malcolm Slaney. *An efficient implementation of the Patterson-Holdsworth auditory filterbank*. Technical report #35, Apple Computer Co. URL: http://citeseer.ist.psu.edu/slaney93efficient.html (besucht am 05. 08. 2008).

[Slab] Malcolm Slaney. *Auditory Toolbox*. Technical Report #1998-010, Interval Research Corporation. URL: http://cobweb.ecn.purdue.edu/~malcolm/interval/1998-010/ (besucht am 05. 08. 2008).

[Tie] Yann Tiersen. *Noten des Stücks Comptine d'un autre été: l'Apres Midi von Michael Jordan*. URL: `http://music.glowingpixel.com/transcriptions/pdf/Yann%20Tiersen%20-%20Comptine%20autre%20ete.pdf` (besucht am 05. 08. 2008).

A Quelltexte

Die Quelltexte sind unter der GNU Lesser General Public License 3 veröffentlicht.

Für die Nutzung des Matlab-Codes ist des Weiteren noch die Auditory Toolbox von Malcolm Slaney notwendig. Die kann man unter der in [Slab] genannten Adresse herunterladen.

Die folgenden Listings stellen die relevanten Teile der Implementation dar. Der vollständige Matlab-Code ist unter der folgenden Adresse verfügbar.

`http://www.schwingungsebene.de/polymt/polymt-src.tgz`

toolbox_test.m

```
path(path, '../auditory_toolbox');
figure(gcf);
clf(gcf);
clc;

max_f0 = 10;

run_transcription('../samples/christian_morgenstern_outro_eins.wav', max_f0);
run_transcription('../samples/santana_samba_pa_ti.wav', max_f0);
run_transcription('../samples/metrovavan_astronomical_twilight.wav', max_f0);
run_transcription('../samples/bach_air.wav', max_f0);
run_transcription('../samples/tiersen_track4.wav', max_f0);
run_transcription('../samples/apocalyptica_nothing_else_matters.wav', max_f0);
run_transcription('../samples/balanescu_quartett_autobahn.wav', max_f0);
run_transcription('../samples/balanescu_quartett_model.wav', max_f0);
run_transcription('../samples/boc_kaini_industries.wav', max_f0);
```

run_transcription.m

```
function run_transcription(infile, max_f0)

%
% define parameters
%
min_freq = 60;
max_freq = 2200;
num_channels = 100;
lp_f_min = 1000; % lowpass filter freq
fft_size = 8*1024;

outfile = sprintf('%s.transcription.%d', infile, max_f0);

% read audio data from file
```

```
[input_waveform_complete, sample_rate] = wavread(infile);


%
% mix down to mono
%
if size(input_waveform_complete,2) == 2 % stereo -> mix down
    input_waveform_complete = input_waveform_complete(:,1) + input_waveform_complete(:,2);
end

frame_duration = 0.092;
frame_length = round(frame_duration * sample_rate);

% normalize and zero padding
input_waveform_complete = input_waveform_complete / max(abs(input_waveform_complete));
input_waveform_complete = [input_waveform_complete' zeros(1, frame_length)];



%
% compute filter coefficients for a bank of Gammatone filters
%
fcoefs = MakeERBFilters(sample_rate, num_channels, min_freq);

%
% main loop
%

offset = 1;
frame_counter = 0;
f_tab = [];
s_tab = [];

while offset + frame_length < length(input_waveform_complete)

    disp(sprintf('frame %d of %d', frame_counter, 2*round(length(input_waveform_complete)/
        frame_length) ));

    input_waveform = input_waveform_complete(offset:round(offset+frame_length));
    frame_counter = frame_counter + 1;
    offset = offset + round(frame_length/2);

    U_k = calc_summary_magnitude_spectrum(sample_rate, input_waveform, ...
        min_freq, num_channels, fcoefs, ...
        lp_f_min, ...
        fft_size);

    U_k = [U_k zeros(1, length(U_k))];

    % step 1
    residual_sms = U_k;
    spectrum_detected = zeros(1,length(U_k));

    f0_list = [];
    s_list = [];
    I = max_f0;

    while( I > 0)
        % step 2
```

```
            [f0,s] = estimate_f0(residual_sms, sample_rate, min_freq, max_freq);
74          f0_list = [f0_list f0];
            s_list = [s_list s];

            % step 3/4
            subtraction_spec = calc_cancelation_spectrum(f0, residual_sms, sample_rate, min_freq,
                max_freq);
79          spectrum_detected = spectrum_detected + subtraction_spec;

            % step 5
            d = 0.5;
            residual_sms = max(0, U_k - d * spectrum_detected);
84
            I = I - 1;
        end

        f_tab = [f_tab; f0_list];
89      s_tab = [s_tab; s_list];
    end

    %
    % store results
94  %

    disp(sprintf('writing results to %s', outfile));
    fd = fopen(outfile,'w');
    max_s = max(max(s_tab));
99  for i=1:size(f_tab,1)
        for j=1:size(f_tab,2)
            n = freq_to_midi_note(f_tab(i,j));
            v = round(100 * s_tab(i,j) /max_s);
            fprintf(fd, 'n%d v%d, ', n, v);
104     end
        fprintf(fd, '\n');
    end
    fclose(fd);
```

calc_summary_magnitude_spectrum.m

```
    function U_k = calc_summary_magnitude_spectrum(sample_rate, waveform_frame, ...
2       min_freq, num_channels, fcoefs, ...
        lp_f_min, ...
        fft_size)

        nyquist_freq = sample_rate / 2;
7
        % filter the waveform and store multiple waveform outputs within a matrix
        coch = ERBFilterBank(waveform_frame, fcoefs);
        erb_center_freqs = ERBSpace(min_freq, nyquist_freq, num_channels);

12      U_k = zeros(1, fft_size/2);

        cochlea_sum = zeros(1, fft_size/2);

        for j=1:size(coch,1)
17
          col = 1;
```

```
        c = coch(size(coch,1)-j+1,:);
        coch_spec = abs(fft(c, fft_size));
        cochlea_sum = cochlea_sum + coch_spec(1:fft_size/2);
22      fc = erb_center_freqs(length(erb_center_freqs)-j+1);

        %disp(sprintf('calculating waveform for auditory channel %d with f_c = %.2f Hz', j, fc)
            );
        %wavwrite(c, sample_rate, sprintf('cochlea_filtered_at_%5.2f.wav', fc));

27      % compress waveform
        c = fwc(c);

        % half-wave-rectification
        c = max(c, 0);
32
        % lowpass filter data from c
        c = filter(fir1(32, lp_f_min/nyquist_freq), 1, c);

        % sum up magnitude spectra
37      c2 = [c zeros(1, length(c))];
        U_c_k = abs(fft(c2, fft_size));
        U_k = U_k + U_c_k(1:length(U_k));
    end
```

calc_cancelation_spectrum.m

```
function subtraction_spec = calc_cancelation_spectrum(f0, residual_sms, sample_rate, min_freq
    , max_freq)

    K = length(residual_sms);
4   K_half = K/2;

    subtraction_spec = zeros(1, length(residual_sms));

    tau = sample_rate / f0;
9   delta_tau = 1;

    p = 1;
    while p <= 20 && p*(K/tau + delta_tau/2) < K_half

14      k0 = floor(p*K / (tau + delta_tau/2)) + 1;
        k1 = max(round(p*K/(tau - delta_tau/2)), k0);
        k = k0:k1; % list of values for k

        weighting = (sample_rate/tau) ./ (0.108 * sample_rate * k / K + 24.7);
19      subtraction_spec(k) =  weighting * residual_sms(k1 - k0 + 1);

        p = p + 1;
    end
```

estimate_f0.m

```
1   function [f0, s] = estimate_f0(spectrum, sample_rate, min_freq, max_freq)
```

```
min_tau = 1;
max_tau = sample_rate/min_freq;
delta_tau = 1;
lambda1 = [];

for tau = min_tau:delta_tau:max_tau
    s = salience(tau, sample_rate, spectrum, delta_tau);
    s = (1 - 0.04 * log(sample_rate/tau)) * s;
    lambda1 = [lambda1 s];
end

%lambda1;
%clf;
%plot(lambda1);

f0 = sample_rate / (argmax(lambda1));
s = max(lambda1);
```

fwc.m

```
function compressed_signal = fwc(waveform)

compressed_signal = [];
v = 0.33;

for i=1:length(waveform)
    if waveform(i) >= 0
        compressed_signal(i) = waveform(i)^v;
    else
        compressed_signal(i) = -((-waveform(i))^v);
    end

end
```

salience.m

```
function lambda = salience(tau, sample_rate, U_k, delta_tau)

U_k_new = U_k; %[U_k zeros(1, length(U_k))];
K = length(U_k_new);
K_half = K / 2;

my_sum = 0;

p = 1;
while p <= 20 && p*(K/tau + delta_tau/2) < K_half

    k0 = floor(p*K / (tau + delta_tau/2)) + 1;
    k1 = max(round(p*K/(tau - delta_tau/2)), k0);

    % debug
    %if tau == 16
    %    [k0 k1]  % k * sample_rate / K
    %end
```

```
        k = k0:k1; % list of values for k
22      H_LP = 1 ./ (0.108 * sample_rate * k / K + 24.7);
        my_sum = my_sum + max(H_LP .* U_k_new(k));
        p = p + 1;
27  end
    lambda = my_sum * sample_rate / tau;
```

argmax.m

```
1   function at_pos = argmax(vector)
    [max_val, at_pos] = max(vector);
```