Technische Universität Berlin

Institut für Sprache und Kommunikation

Fachgebiet Audiokommunikation

Prof. Dr. Stefan Weinzierl

Bachelorarbeit

Ein Real Time Analyzer für Audiosignale im Tonstudio

eingereicht von
Alexander Jossifov
315714

betreut von
Prof. Dr. S. Weinzierl
Prof. Dr.-Ing. T. Sikora
Wilm Thoben
Andre Bartetzki

Berlin, 03.01.2013

Zusammenfassung

Zuverlässiges und hörgerechtes Metering ist für eine professionelle Studioumgebung unerlässlich. Die Überwachung und Beurteilung von Audiosignalen ermöglicht die optimale Ausnutzung der Audiosystemdynamik und vor allem die Balance und Aussteuerung verschiedener Audioinhalte zueinander. Um das zu ermöglichen werden Real-Time Analyzer benutzt, welche die Lautheit eines Eingangssignals an bestimmten Frequenzen über einen Frequenzbereich in Echtzeit messen. Das Ziel der Arbeit war es ein Real-Time Analyzer als UGen für die Audioprogrammiersprache SuperCollider zu entwickelt, der als Grundlage die Spezifikationen der DIN EN 61260 hat und eine Lautheitsanpassung beim Metering nach ITU-R BS.1770 vornimmt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

DFT	Discrete Fourier Transform
DSP	Digitaler Signalprozessor
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
OSC	Open Sound Control
RLB	Revised Low-frequency B-curve
RMS	Root Mean Square
RTA	Real-Time Analyzer
UGen	Unit-Generator

1 Einleitung

Zuverlässiges und hörgerechtes Metering ist für eine professionelle Studioumgebung unerlässlich. Die Überwachung und Beurteilung von Audiosignalen ermöglicht die optimale Ausnutzung der Audiosystemdynamik und vor allem die Balance und Aussteuerung verschiedener Audioinhalte zueinander.

Ein Teil der Studioumgebung stellt der Real Time Analyzer dar. Dabei handelt es sich um ein Spektrumanalysator, der die Lautheit eines Eingangssignals an bestimmten Frequenzen über einen Frequenzbereich in Echtzeit misst. Die Implementierung eines solchen RTAs kann sowohl analog als auch digital erfolgen. „Rein analoge RTA-Lösungen sind mittlerweile wegen des hohen Materialaufwands und der Genauigkeitsanforderungen [...] selten geworden.“ (Müller, 2008, S. 1093) Typischer ist der Einsatz von speziellen, digitalen Signalprozessoren (DSPs) oder reinen Softwarelösungen.

Der technische Fortschritt bietet die Möglichkeit die im Tonstudio eingesetzte Hardware durch äquivalente, preiswerte Software zu ersetzen. Diese Komponenten lassen sich über vordefinierte Schnittstellen beliebig in die Studioumgebung einbinden und vervielfältigen, was eine sehr flexible Arbeitsweise erlaubt. Ein großer Vorteil ist, dass die Entwicklung der digitalen Komponenten nicht die hohen Anforderungen der analogen Gegenstücke erfüllen müssen, da physikalische Einfluss- bzw. Störfaktoren bei Softwareanwendungen wegfallen.
Aber auch im privaten und Open Source Bereich steigt das Interesse für gute Lösungen zur Verarbeitung und Analyse von verschiedenen Audioinhalten, wie z.B. bei der Videobearbeitung und der Produktion von Musik.

Als Anstoß für diese Arbeit dient die Renovierung des elektronischen Studios der TU Berlin und der damit verbundenen Entwicklung einer Studiosoftware. Teil dieser Software soll ein für die Audioprogrammiersprache SuperCollider entwickelter Real Time Analyzer sein, der als Grundlage die Spezifikationen der DIN EN 61260 hat. Darüber hinaus soll eine Lautheitsanpassung beim Metering nach ITU-R BS.1770 erfolgen. Ziel der Arbeit ist es zu prüfen, ob ein RTA in SuperCollider implementiert werden kann, der die Anforderungen der Normen erfüllt und für professionelle Zwecke im Tonstudio eingesetzt werden kann. Zusätzlich soll festgestellt werden, ob die Performanz des speziell entwickelten UGens mit einem äquivalenten Pseudo-UGen, also einer Kombination schon vorhandener SuperCollider-Komponenten, vergleichbar oder sogar besser ist.

Die Arbeit gliedert sich in fünf Teile. Ich werde zunächst auf einige Grundlagen der digitalen Signalverarbeitung eingehen. Dabei liegt der Schwerpunkt auf digitale Filter, da sie der Grundbaustein für die Implementierung des Real Time Analyzers sind. Dann gehe ich auf die in der Arbeit verwendeten Normen ein und erläutere die für das Tool entscheidenden Richtlinien. Als nächstes gebe ich einen kleinen Überblick über die SuperCollider Umgebung und deren Funktionsweise, speziell in Hinsicht auf UGens. Der Hauptteil der Arbeit beschäftigt sich mit der Aufnahme der Anforderungen, der Implementierung der einzelnen Bestandteile des Tools, den damit verbundenen Designentscheidungen und dem

Test und der Evaluation des Tools. Zum Schluss werden die Ergebnisse der Arbeit noch einmal zusammengefasst und mögliche Verbesserungen vorgeschlagen.

2 Hauptteil

2.1 Digitale Signalverarbeitung

2.1.1 Grundlagen

Digitale Signalverarbeitung im Audiobereich ist die Verarbeitung zeit- und wertdiskreter Amplitudenwerte. Diese erhält man durch die Abtastung des analogen Signals in festen Zeitabständen. Dabei sind zwei Faktoren zu beachten: die Abtastrate, auch Samplerate oder Abtastfrequenz genannt, und die Sampletiefe. (Zölzer, 2002, 2008)

Die Samplerate bestimmt, wie oft ein Signal pro Sekunde abgetastet wird. Hier spielt das Nyquist-Shannon-Abtasttheorem eine wichtige Rolle, das besagt: „Ein abgetastetes Signal lässt sich ohne Informationsverlust rekonstruieren, wenn die Abtastfrequenz mehr als doppelt so hoch ist wie die höchste im Signal vorkommende Frequenz." (Lerch u. Weinzierl, 2008, S. 788) Das heißt, dass entweder die Samplerate mindestens das Doppelte der höchsten Frequenz des analogen Signals sein muss oder alle Frequenzen des Signals, die über der Hälfte der Samplefrequenz liegen, müssen mit Hilfe eines Filters gesperrt werden. Diese Abtastfrequenz wird auch als Nyquist-Frequenz bezeichnet.

Werden die Amplitudenwerte durch Binärzahlen dargestellt, wie es in EDV-Systemen üblich ist, bestimmt die Anzahl der Bits die Sampletiefe des digitalen Signals. Dies wird auch als Quantisierung bezeichnet. Wird zum Beispiel eine Wortbreite von 16 Bit für die Speicherung verwendet, sind 2^{16} = 65536 Quantisierungsstufen möglich d.h. es können Werte im Bereich von -32768 bis 32767 dargestellt werden. Das Signal wird auch oft als normierte Fließkommazahl zwischen -1.0 und 1.0 angegeben.

Für die Verarbeitung von digitalen Signalen gibt es drei Grundoperationen: die Gewichtung eines Signals, die Summation von gewichteten Signalen und die Verzögerung eines Signals um ganzzahlige Abtasttakte. Die Kombination dieser Grundoperationen wird auch als Filteroperation bezeichnet. (vgl. Zölzer, 2008, S. 816)

Eine Abbildung, die ein Eingangssignal mit Hilfe der Grundoperationen in ein Ausgangssignal überführt, wird als digitales System bezeichnet. Dieses System lässt sich durch die Impulsantwort beschreiben, welche angibt, wie sich das System bei der Eingabe eines sogenannten Einheitsimpulses verhält. Die Impulsantwort ist durch Gewichtungsfaktoren bzw. Koeffizienten definiert. (vgl. Zölzer, 2008, S. 817)

„Der Frequenzgang eines Systems beschreibt die Verstärkung und Phasenverschiebung des Systems über der Frequenz und wird über die zeitdiskrete Fourier-Transformation der Impulsantwort des Systems gemäß (15.6) berechnet. Hieraus wird der Betragsfrequenzgang (15.7) und der Phasengang (15.8) des Systems berechnet." (Zölzer, 2008, S. 820)

2.1.2 Filter

„Unter dem Begriff Filter versteht man ein System, welches nur bestimmte Frequenzkomponenten des Eingangssignals zum Ausgang des Systems durchlässt und die restlichen Frequenzkomponenten unterdrückt. Der Betragsfrequenzgang $|H(f)|$ eines digitalen Filters wird hierzu unterteilt in einen Durchlass- und einen Sperrbereich.“ (Zölzer, 2008, S. 823) „Filter werden zur Beeinflussung des Spektrums des Audiosignals eingesetzt, um eine Klangmodifikation zu erzielen.“ (Zölzer, 2008, S. 831)

Ein Filter lässt sich durch seine Grenzfrequenz und Ordnung beschreiben. Die Grenzfrequenz gibt an, ab welcher Frequenz der Betragsfrequenzgang abfallen soll. Die Ordnung gibt an, wie stark der Abfall pro Oktave ab der besagten Grenzfrequenz ist. Man spricht auch von der Flankensteilheit des Filters. Die Amplitude an der Grenzfrequenz beträgt $\sqrt{1/2}$ bzw. -3 dB.

Filtertypen

Es existieren mehrere Filtertypen, die nach ihrer Eigenschaft, Signale in bestimmten Frequenzbereichen zu sperren oder durch zu lassen, unterschieden werden. Beispiele dafür sind Tiefpass-, Hochpass- und Bandpass-Filter, um nur ein paar zu nennen. Die Filtertypen, die für diese Arbeit von Bedeutung sind, sind Hochpass-, Bandpass-, und Shelving-Filter (siehe Abb. 1).

Bei Hochpass-Filter liegt der Sperrbereich unter, und der Durchlassbereich über der Grenzfrequenz. Der Bandpass-Filter hat zwei Sperrbereiche, links und rechts vom Durchlassbereich, man Spricht auch vom Durchlassband. Dieser wird durch eine untere Bandeckfrequenz (größer Null) und eine endliche obere Bandeckfrequenz definiert. Es wird auch die Ordnung und die Bandmittenfrequenz des Filters angegeben. (DIN EN 61260, 2003) Shelving-Filter lassen das Signal bis zu der Shelving-Frequenz unverändert durch. Ab dieser Frequenz wird das Signal bis zu einer bestimmten Amplitude verstärkt oder gedämpft. Sie existieren sowohl als Low- als auch als High-Shelving-Varianten.

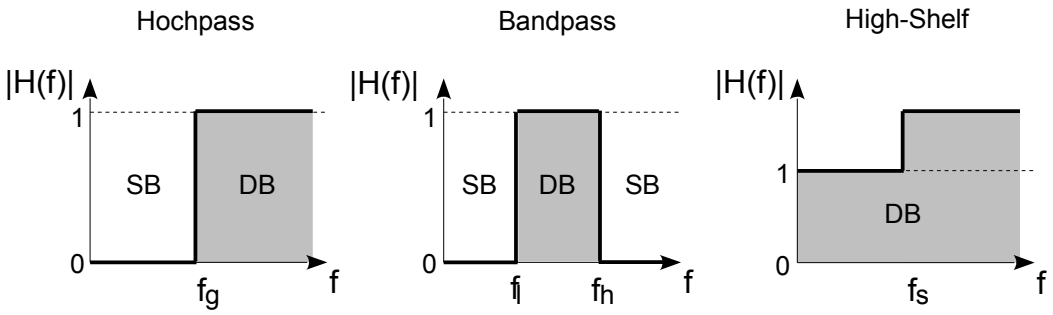

Abbildung 1 : Filtertypen

Filterdesign

Man unterscheidet bei digitalen Filter zwischen FIR-Filter, die eine endliche Impulsantwort haben, und IIR-Filter, die eine unendliche Impulsantwort haben. Die Wahl zwischen den beiden ist abhängig vom Anwendungszweck, wobei anzumerken ist, dass die Latenz des Eingangssignals (Laufzeit durch den Filter) und der Rechenaufwand bei IIR-Filter geringer sind als bei FIR-Filter. (vgl. Zölzer, 2008, S. 830)

Der Frequenzgang eines IIR-Filters wird mit

$$H(f) = \frac{\sum_{n=0}^{N-1} b(n) e^{-j2\pi nf/f_A}}{1 + \sum_{n=1}^{M} a(n) e^{-j2\pi nf/f_A}}$$

berechnet. Das Ausgangssignal wird durch die Differenzgleichung

$$y(n) = \sum_{k=0}^{N-1} b(k)x(n-k) - \sum_{k=1}^{M} a(k)y(n-k)$$

bestimmt. Die Gleichung für das Ausgangssignal eines IIR-Filters 2. Ordnung hat also die folgende Form:

$$y(n) = b_0 \cdot x(n) + b_1 \cdot x(n-1) + b_2 \cdot x(n-2) - a_1 \cdot y(n-1) - a_2 \cdot y(n-2)$$

Diese Form wird auch als *Direct Form I* bezeichnet. Alternativ kann man durch eine Umstellung der Gleichung das Ausgangssignal auch mit

$$\begin{aligned} w(n) &= x(n) - a_1 \cdot w(n-1) - a_2 \cdot w(n-2) \\ y(n) &= b_0 \cdot w(n) + b_1 \cdot w(n-1) + b_2 \cdot w(n-2) \end{aligned}$$

berechnen. Dies wird als *Direct Form II* bezeichnet und spielt für die Implementierung und Optimierung des Filters eine wichtige Rolle, da weniger Verzögerungswerte (Delay-Werte) zwischengespeichert werden müssen. (vgl. Boulanger u. Lazzarini, 2011, S. 485)

„Für viele Anwendungen reichen IIR-Filter 1. und 2. Ordnung aus. Für höhere Filterordnungen werden Filter 1./2. Ordnung in Serie geschaltet.“ (Zölzer, 2008, S. 830) Bei in Serie geschalteten Filter 2. Ordnung spricht man auch von *second-order sections* oder Biquad-Filter.

Es gibt verschiedene Umsetzungen dieser Filter, die alle unterschiedliche Eigenschaften haben. Beispiele dafür sind Chebyshev-, Cauer-/Elliptic- und Butterworth-Filter. Im Gegensatz zu dem Chebyshev- und Elliptic-Filter ist der Butterworth-Filter sowohl im Durchlass als auch im Sperrbereich eben und ist als IIR-Filter gut für den Einsatz in Softwareanwendungen geeignet.

Die Übertragungsfunktion der digitalen Filter wird üblicherweise aus dem analogen Gegenstück mit Hilfe der bilinearen Transformation hergeleitet.

Anwendungen

Mit Hilfe von Filter lassen sich eine Vielzahl von Audio-Effekte realisieren, die in professioneller Audioverarbeitungs-Software vorzufinden sind, wie z.B. Tremolo, Vibrato, Chorus, Flanger, Phaser, Limiter, Kompressor, Reverb, usw. Sie sind jedoch auch gut geeignet für Bewertungsfilter, Equalizer und Pegel-, Effektivwert- und Spitzenwertmessungen. (vgl. Zölzer, 2008, S. 830)

Auch Real Time Analyzer werden mit Hilfe von Filter verwirklicht. (siehe Müller, 2008, S. 1092) Dafür werden eine Reihe von Bandpassfilter parallel geschaltet, die eine konstante relative Bandbreite haben, d.h. die Filter liegen in einem bestimmten musikalischen Intervall zueinander, was üblicherweise eine Terz, Oktave, aber auch ein Halbtonschritt sein kann. Aus der Energie des gefilterten Signals, lässt sich die Lautheit des Audiosignals in dem vom Bandpassfilter definierten Frequenzbereich ermitteln. So lässt sich die spektrale Verteilung bzw. der „Anteil unterschiedlicher Frequenzen am Audiosignal“ darstellen. Der Vorteil einer solchen Umsetzung ist eine geringere Analyselatenz und eine Frequenzauflösung die „gehörangepasster“ ist als bei anderen Spektralanalyse-Methoden. (vgl. Weinzierl, 2006, S. 22)

Die Anforderungen an einen solchen Filtersatz werden in der DIN EN 61260 Norm spezifiziert.

2.2 Verwendete Normen

2.2.1 DIN EN 61260

Bei der DIN EN 61260 (2003) handelt es sich um eine internationale Norm, die Anforderungen an analoge und digitale Bandfilter stellt, die Teil eines Filtersatzes oder eines Spektrumanalysators sein sollen. Werden diese Anforderungen erfüllt, dürfen beliebig viele Bandfilter eingesetzt werden, die sich über jeden gewünschten Frequenzbereich erstrecken können.

Darüber hinaus werden Gleichungen für die Berechnung von Bandmittenfrequenzen und Bandeckfrequenzen aufgeführt, die abhängig vom Intervall der Filter sind. Die danach konstruierten Filter müssen die Anforderungen für die relative Dämpfung erfüllen, d.h. die Ordnung des Filters muss so gewählt werden, dass der Betragsfrequenzgang in den vorgegebenen Grenzen liegt. Dabei werden drei verschiedene Filterklassen (0 bis 2) unterschieden. Je kleiner die Zahl ist umso kleiner sind die Fehlergrenzen, in den der Betragsfrequenzgang liegen darf. In der Regel erfüllen digitale Butterworth Bandpassfilter 6. Ordnung die Anforderungen. Das entspricht einer Flankensteilheit von 18 dB/Oktave.

Weitere Richtlinien für die Bandfilter werden in Abschnitt 4 der Norm genannt. Einige sind auf digitale Filter jedoch nicht anwendbar, da es sich z.B. um physikalische Bedingungen an ein elektronisches Gerät handelt. Die Anforderungen, die für diese Arbeit von Bedeutung sind, beschränken sich auf die relative Dämpfung und die Echtzeitverarbeitung.

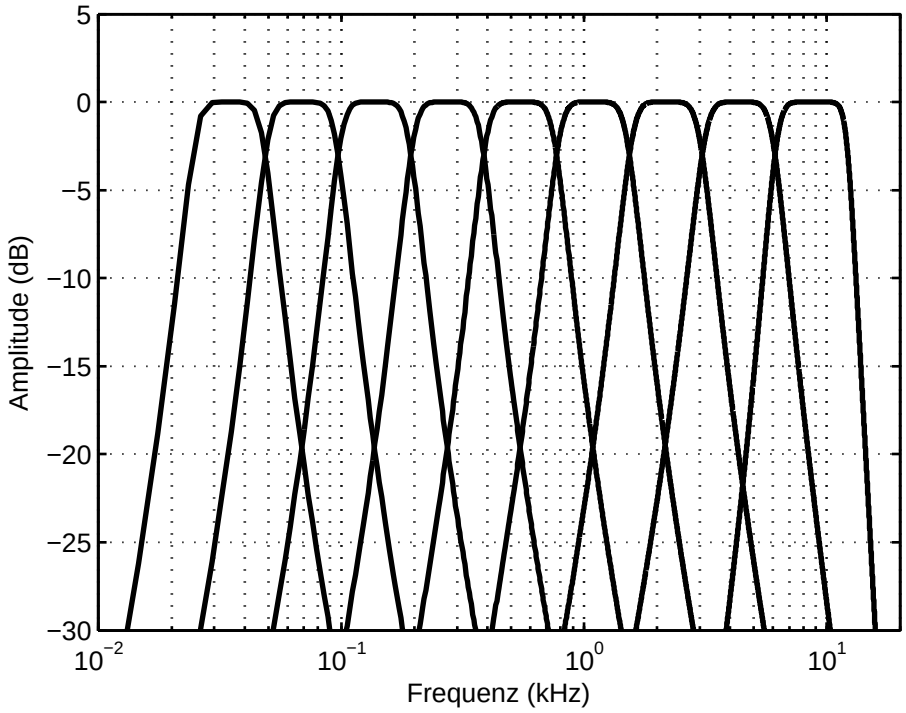

Abbildung 2 : Oktav-Filter, 44100 Hz Samplerate

2.2.2 ITU-R BS.1770

Bei der ITU-R BS.1770 (2006) handelt es sich um eine Norm zur Messung der subjektiven Lautheit und dem True-Peak eines oder mehrerer Audiosignale. Dabei werden Phänomene aus der Psychoakkustik berücksichtigt.

Die Norm ist für Anwendungen gedacht, in denen die Lautheit unterschiedlicher Audiosignale, wie sie im Radio- und Fernsehrundfunk vorkommen, gemessen und angepasst werden muss. Dabei unterscheidet man zwischen verschiedenen Audioinhalten, wie z.B. Sprache, Musik, Soundeffekte, oder eine Kombination aus den dreien, wobei der Wechsel zwischen diesen Inhalten einen starken Einfluss auf die wahrgenommene Lautheit haben kann. Speziell bei der Tonträgerproduktion ist eine verlässliche Lautheitsanpassung beim Mastering der einzelnen Titel, z.B. einer Musikkompilation, von großer Bedeutung. Die Norm ist jedoch im Allgemeinen nicht für die Messung der subjektiven Lautheit einzelner Töne geeigneten.

Es werden zwei Filter eingesetzt: ein sogenannter Pre-Filter (Abb. 3), der das Audiosignal ab etwa 1000 Hz verstärkt, und ein RLB-Filter (Abb. 4), der das Audiosignal unter 400 Hz dämpft. Die Aufgabe des Pre-Filters ist es für die akustischen Effekte aufzukommen, die der Kopf auf den einfallenden Schall hat. Der RLB-Filter ist ein sogenannter B-Bewertungsfilter, der die Schallempfindlichkeit des menschlichen Gehörs bei niedrigen Frequenzen nachbilden soll. Die Koeffizienten beider Filter werden für eine Samplerate von 48 kHz angegeben (siehe Tabelle 1 und 2).

Der Algorithmus, der in der Norm beschrieben wird, funktioniert nach dem Prinzip, dass beide Filter in Serie geschaltet und auf das Eingangssignal angewendet werden. Dann wird der quadratische Mittelwert des gefilterten Signals für jeden Kanal berechnet und je nach der Richting, von dem der Schall kommt, gewichtet. Die Gewichtung soll dabei für den Effekt aufkommen, dass „rückwärtig einfallender Schall [...] im Mittel lauter wahrgenommen wird als frontal einfallender Schall“. (Weinzierl, 2008a, S. 562) Zum Schluss wird die gewichtete Energie jedes Kanals aufsummiert und die Lautheit berechnet. Der Algorithmus ist für einen bis maximal fünf Kanäle bzw. Eingangssignale gedacht (left, right, center, left surround, right surround).

Seit der Veröffentlichung der ITU-R BS.1770 (2006) sind noch zwei weitere Versionen der Norm erschienen. Die Aktuellste ist 2012, in der der Algorithmus zur Berechnung der Lautheit mehrerer Kanäle um eine Gate-Komponente erweitert wird, die Signale unter einer bestimmten Lautheit bei der Summierung der Energie unterdrücken soll.

Der zweite Teil der Norm beschäftigt sich mit der Messung des sogenannten True-Peak Levels. Es wird ein Algorithmus zur Messung des Spitzenwertes vorgeschlagen, der Anomalien entgegnen soll, die bei der Angabe des Spitzenwertes anhand der maximalen, absoluten Samplewerte auftreten.

i	a_i	b_i
0	1.0	1.53512485958697
1	-1.69065929318241	-2.69169618940638
2	0.73248077421585	1.19839281085285

Tabelle 1 : Pre-Filter, Koeffizienten, 48 kHz Samplerate

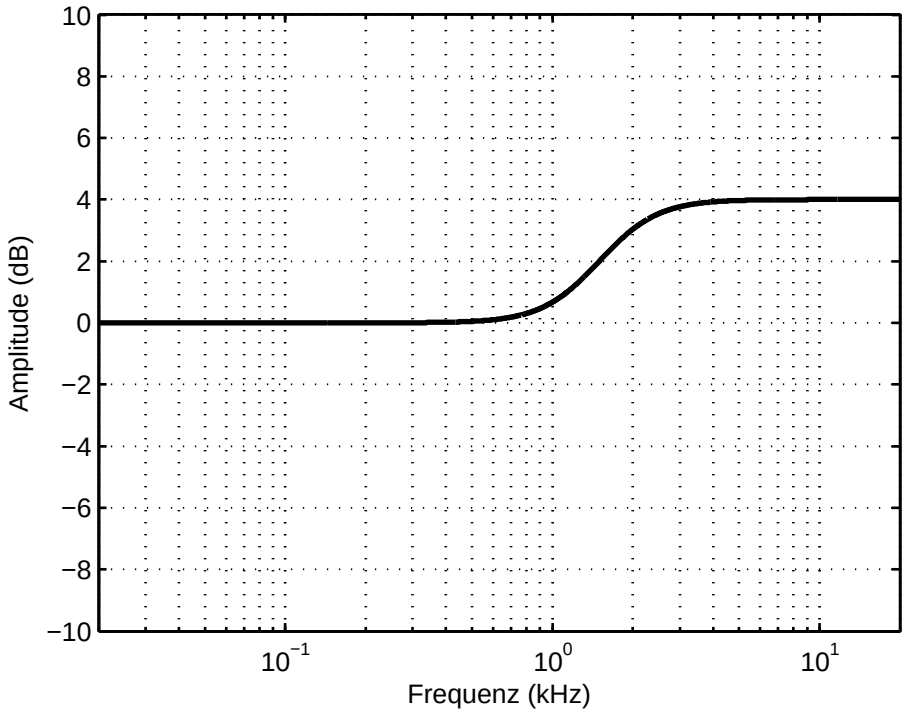

Abbildung 3 : Pre-Filter, Betragsfrequenzgang, 48 kHz Samplerate

i	a_i	b_i
0	1.0	1.0
1	-1.99004745483398	-2.0
2	0.99007225036621	1.0

Tabelle 2 : RLB-Filter, Koeffizienten, 48 kHz Samplerate

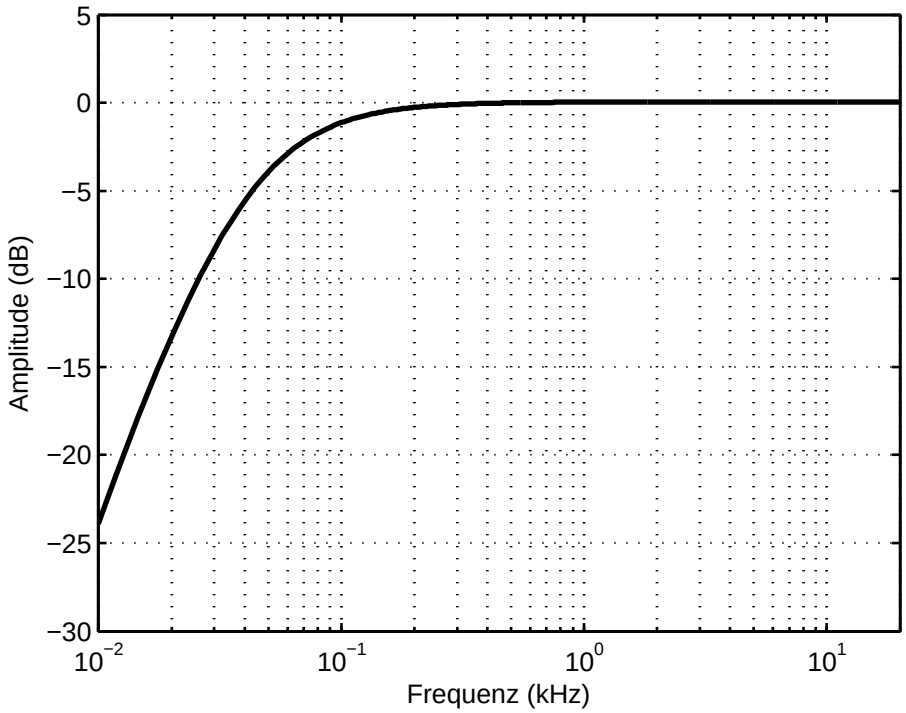

Abbildung 4 : RLB-Filter, Betragsfrequenzgang, 48 kHz Samplerate

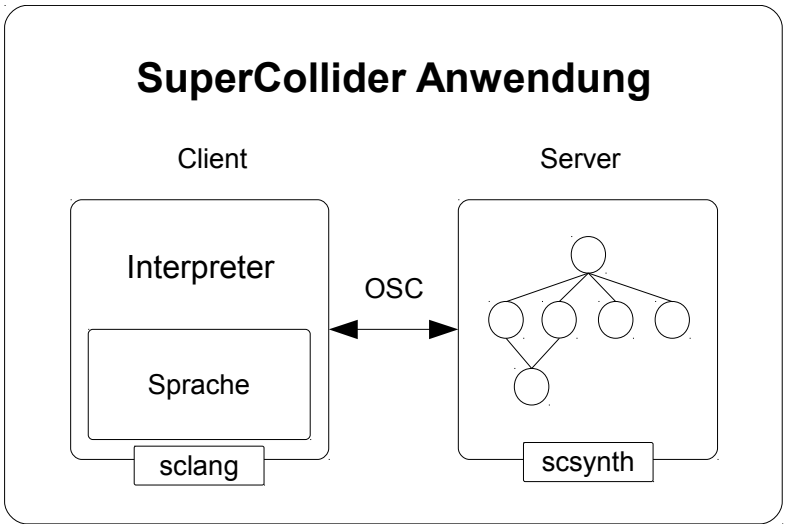

Abbildung 5 : SuperCollider Client-Server-Architektur (SCCode, 2012, Abb. 1)

2.3 SuperCollider

SupcerCollider[1] ist eine Open Source Programmierumgebung für *real time audio synthesis* und algorithmische Komposition. Sie basiert auf einer Client-Server-Architektur (Abb. 5). Der Server (scsynth) verwaltet die Synthesizer und den Synth-Graphen, der die Verbindung zwischen den einzelnen Komponenten und die Reihenfolge der Verarbeitung des Signals durch diese Komponenten angibt. Der Client umfasst die SuperCollider Sprache (sclang), die dem Benutzer die Möglichkeit gibt den Aufbau des Synth-Graphen zu beschreiben und die Funktionen, die darauf angewendet werden sollen. Client und Server kommunizieren über eine OSC-Schnittstelle miteinander.

2.3.1 UGens

Mit Hilfe von UGens (Parmenter, 2011; Stowell, 2011) lässt sich die Funktionalität von SuperCollider um zusätzliche Komponenten zur Audiogenerierung und Audioverarbeitung erweitern. Diese werden als Plug-Ins vom Server erkannt und können mit anderen UGens zu einen Graphen zusammen geschaltet werden. Die Eingaben und Ausgaben der UGens können sowohl Audiosignale als auch Steuerwerte sein.

Man unterscheidet in dem Zusammenhang auch zwischen zwei verschiedenen Raten: der Rate in der das Audiosignal übertragen wird (audio rate) und der Rate mit der die Parameter der UGens zur Laufzeit gesteuert werden können (control rate). Bei der Implementierung ist darauf zu achten, mit welcher Rate der UGen aufgerufen wird, weil sich abhängig davon die zu verarbeitende Blockgröße und die Ausgabe verändert.

SuperCollider erlaubt es sogenannte Pseudo-UGens in der SuperCollider Sprache zu definieren, auch SynthDefs genannt, bei denen schon vorhandene UGens zu einer neuen Einheit zusammengefasst werden. Bei diesem Vorgang entsteht jedoch ein gewisser Overhead, der die Pseudo-UGens für manche Anwendungen unpraktisch macht. Dieser ergibt

[1] Projektseite: http://supercollider.sourceforge.net/

sich durch die Verwaltung der zusätzlichen Graphen-Struktur und der einzelnen UGens, sowie die Kommunikation zwischen ihnen. Eine effizientere Möglichkeit stellt die Entwicklung neuer Plug-Ins in C++ dar, die nach der Implementierung wie normale UGens verwendet werden können.

Jeder UGen hat vier Funktionen, die vom Server aufgerufen werden: eine load-Methode, einen Konstruktor, einen Destruktor und eine next-Funktion. Die load-Methode wird aufgerufen, wenn der Server startet und die Plug-Ins geladen werden. Der Konstruktor wird aufgerufen sobald ein UGen instantiiert wird. Die eigentliche Funktionalität des UGens liegt in der next-Funktion, die periodisch aufgerufen wird und dem UGen das Eingangssignal, also die Samplewerte, in Form eines Input-Buffers zur Verfügung stellt. Diese Werte werden verarbeitet und als Ausgangssignal in den Output-Buffer geschrieben. Der Destruktor wird aufgerufen, wenn der UGen nicht mehr gebraucht wird und wird hauptsächlich dazu verwendet, um den vom UGen allozierten Speicher wieder frei zu geben.

SuperCollider verwendet einen sogenannten *buffer coloring* Algorithmus, um eine effizientere Nutzung der Zwischenspeicher zu erzielen. Für den UGen bedeutet das, dass der übergebene Input-Buffer gleichzeitig auch der Output-Buffer sein kann. Das Schreiben von Werten in den Output-Buffer kann also dazu führen, dass die Werte des Input-Buffers überschrieben werden und bei wiederholtem Auslesen nicht dem ursprünglichem Eingangssignal entsprechen. Für Filteroperationen benutzt man deshalb eigene Buffer, um Teiles des Eingangs- und Ausgangssignal zwischenzuspeichern. (Stowell, 2011, S. 702)

Bei der Entwicklung von UGens ist auch die sogenannte *multichannel expansion* zu beachten. Wenn man dem UGen mehrere Eingangssignale als Parameter übergibt, dann werden beim Server, abhängig von der Anzahl der Kanäle, automatisch mehrere Instanzen des UGens erzeugt, die parallel zueinander in den Synth-Graphen eingefügt werden. Um das zu umgehen, muss man die Pointer zu den Input-Buffern als Objekt auffassen, dann dereferenziert man das Objekt beim Server und übergibt die Pointer als ein Array von Parametern an den UGen. Auf diese Weise kann man eine beliebige Anzahl an Eingangssignale verarbeiten, was z.B. für die Messung der Lautheit mehrkanaliger Audiosignale, wie sie in der ITU-R BS.1770 beschrieben wird, notwendig wäre.
Das UGen, das in dieser Arbeit entwickelt wird, berechnet jedoch keine Summenlautheit mehrkanaliger Audiosignale. Falls der Anwender das wünscht, ist es ihm überlassen, ob er sie vorher aufsummieren möchte.

Zusätzlich besitzt SuperCollider über einen sogenannten *real-time memory pool*, mit dessen Hilfe eine effizientere Speicherverwaltung beim Server erfolgen soll. Speicher wird mit *RTAlloc* alloziert und mit *RTFree* wieder freigegeben. (Stowell, 2011, S. 707)

2.3.2 Open Sound Control

Open Sound Control[2] ist ein Nachrichten-basiertes Protokoll für die Kommunikation zwischen Computer, Sound Synthesizer und Multimedia Geräte. Es ist für Client-Server-Architekturen gedacht und wird in der Anwendungsschicht implementiert, unabhängig vom darunter liegenden Transportprotokoll. Es hat viele verschiedene Anwendungen und Implementierungen wie z.B. für die Abbildung nicht musikalische Parameter von Sensoren oder Eingabegeräten unterschiedlicher Art auf elektronische Musikinstrumente, LAN-unterstützte Musikaufführungen, Mehrnutzerbetrieb eines Sound Servers oder der Schachtelung anderer Protokolle in OSC.

Wie schon erwähnt benutzt auch SuperCollider die OSC-Schnittstelle für die Client-Server-Kommunikation. Die OSC-Nachrichten werden über UDP oder TCP verschickt. Damit lassen sich beispielsweise konkrete Komponenten des Synth-Graphen ansprechen und deren Parameter zur Laufzeit verändern, aber auch der Synth-Graph selbst kann modifiziert und um neue Synths erweitert werden. Es ist auch möglich Daten über die Schnittstelle zu übertragen, die Server-seitig in Buffer gespeichert werden. Im Fall des RTAs können z.B. die Analysedaten auf diese Weise zum Client übertragen werden, der sie dann grafisch darstellt. Mit der OSC-Schnittstelle können auch andere Anwendungen, unabhängig von der SuperCollider Sprache, mit dem Server kommunizieren. (vgl. Wright u. a., 2003)

[2] Projektseite: http://opensoundcontrol.org/

2.4 Durchführung

Zunächst werden die Anforderungen an das Tool beschrieben und funktionale Anforderungen davon abgeleitet. Danach wird die Implementierung der einzelnen Tools zur Berechnung der im UGen verwendeten Filter näher erläutert und auf die Dateistruktur zur Speicherung der Filterkoeffizienten eingegangen. Zum Schluss gehe ich auf den Aufbau und der Funktionsweise des UGens ein.

2.4.1 Anforderungen

Die folgenden Anforderungen an das Tool ergeben sich aus der Aufgabenstellung der Bachelorarbeit:

- Das Real Time Analyzer Tool soll als UGen für die Audioprogrammiersprache SuperCollider entwickelt werden.
- Für die Spezifikation des Tools soll DIN EN 61260 benutzt werden. Zudem soll eine Lautheitsanpassung beim Metering berücksichtigt werden nach ITU-R BS.1770.
- Es sollen Oktav-, Terzband und Halbtonschritt-Filter benutzt werden, um die Auflösung des Meterings einstellen zu können.
- Die Filterkoeffizienten sollen für die Sampleraten 44.1 / 48 / 88.2 / 96 / 176.4 / 192 KHz berechnet werden, damit das Tool Audiosignale mit unterschiedlichen Sampleraten verarbeiten kann.
- Für die RMS-Berechnung sollen die Intergrationszeiten einstellbar sein.
- Die OSC Schnittstelle soll so angepasst werden, dass der UGen effektiv in einem Tonstudio eingesetzt werden kann.
- Es soll eine Peak-Level Berechnung vorgenommen werden.
- Sowohl die RMS-Werte als auch die Peak-Werte sollen eine Abfallrate (Decay) haben, der mit dB pro Sekunde angegeben wird.

Aus diesen Anforderungen lassen sich folgende funktionale Anforderungen an das Tool ableiten:

- Der UGen soll fünf verschiedene Parameter haben: die Samplerate (44.1 / 48 / 88.2 / 96 / 176.4 / 192 KHz), das Filterintervall (Oktave, Terz, Halbtonschritt), die Integrationszeit in Sekunden, die Abfallrate für die RMS- und Peak-Level-Spitzenwerte in dB pro Sekunde.
- Es werden Filtersätze für alle Kombinationen von Sampleraten und Intervalle nach DIN EN 61260 und ITU-R BS.1770 berechnet. Das Ausgabesignal der Filter muss den Anforderungen der Normen entsprechen.
- Für die Lautheitsanpassung soll der Pre-Filter und RLB-Filter auf das Audiosignal angewendet werden.
- Alle Filter werden als Biquad-Filter (second-order sections) und in der Direct Form II implementiert.
- Es soll der quadratische Mittelwert (RMS) des gefilterten Signals berechnet werden. Bei jedem Aufruf der next-Methode wird über das ganze Integrationsfenster der quadratische Mittelwert berechnet.
- Das Tool muss komplett in der SuperCollider Umgebung integriert und von *sclang* aufrufbar sein. Dazu muss der UGen implementiert und eine Class-Datei angelegt werden.
- Für die Übertragung der Analysedaten des RTA wird ein spezieller Buffer alloziert, auf den sowohl der Client als auch das UGen zugreifen kann.
- Alle Buffer, die in dem UGen für die Filter und der RMS-Berechnung verwendet werden, werden als Ring-Buffer implementiert, die sich nach außen hin wie Stacks verhalten.
- Das Peak-Level wird vom gefilterten Ausgangssignal bestimmt. Dabei wird der maximale Absolutwert des Signals bestimmt.
- Der UGen soll so effizient wie möglich implementiert werden, sodass eine möglichst niedrige CPU-Last erzielt wird.

2.4.2 Implementierung

Das Tool ist so aufgebaut, dass das Audiosignal durch eine Reihe von Filter geschickt und verarbeitet wird. Dabei wird zunächst eine Lautheitsanpassung vorgenommen (ITU-R BS.1770), dann wird das Spektrum auf einen bestimmten Frequenzbereich eingeschränkt (DIN EN 61260), der quadratische Mittelwert (RMS) und das Peak-Level berechnet und zum Schluss die Lautheit im jeweiligen Filterbereich ausgegeben. Das wird für alle Filter eines bestimmten Gesamtfrequenzbereichs (z.B. 20 Hz bis 22050 Hz) gemacht. Die Filter werden durch ihre Koeffizienten angegeben, die dem Tool zur Laufzeit zur Verfügung stehen.

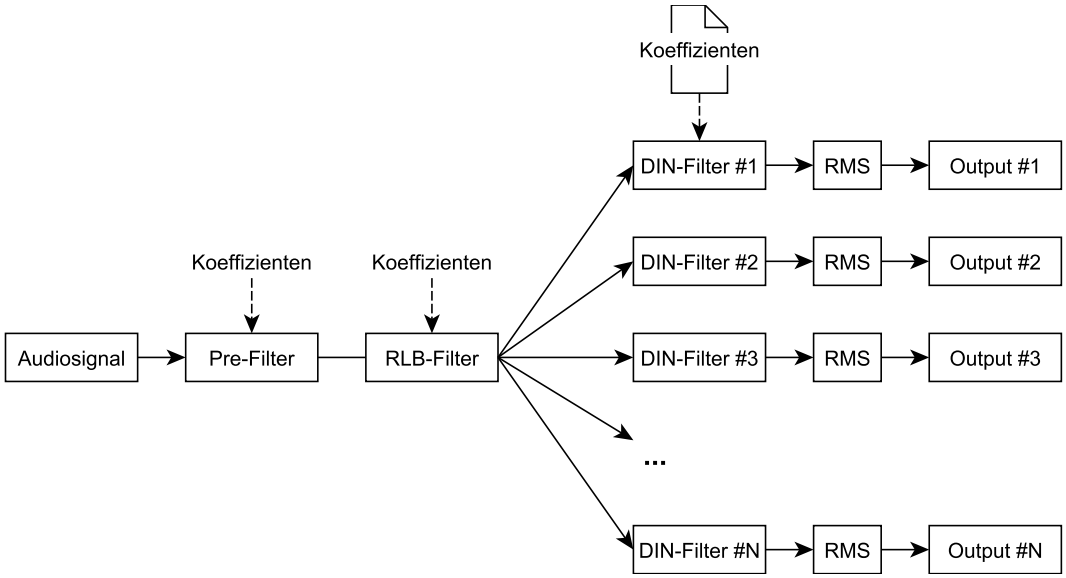

Abbildung 6 : Funktionsweise des Tools

Da es eine begrenzte Anzahl an Filtereinstellungen gibt und die Filter zur Laufzeit nicht verändert werden, bietet es sich an die Koeffizienten der DIN EN 61260 Filter im Voraus zu berechnen und zu exportieren, um sie später im Tool integrieren zu können. Darüber hinaus mussten die in der ITU-R BS.1770 Norm beschriebenen Filter rekonstruiert und ihre Koeffizienten für die verschiedenen Sampleraten berechnet werden, da sie dort nur für die Samplerate 48 kHz vorliegen. Beides wurde mit Hilfe von MATLAB[3] bewerkstelligt. Die verwendeten Skripte werden in den nächsten Kapiteln genauer beschrieben.

Um Rundungsfehler oder Ungenauigkeiten bei der Berechnung so stark wie möglich einzugrenzen, wird ausschließlich *double*-Präzision für die Speicherung des Audiosignals, der Koeffizienten und der Delay-Werte verwendet. Ebenso werden die exportierten Koeffizienten als Gleitkommazahlen mit 15 Nachkommastellen, Mantisse und Exponent gespeichert.

Die Filterfunktionen werden in der *Transposed Direct Form II* implementiert. Das hat den Vorteil, dass durch die Umstellung der Gleichung nur noch die Hälfte an Delay-Werte für die Berechnung des Ausgangssignals gespeichert werden müssen. Damit wird neben

[3] Seite: http://www.mathworks.de/products/matlab/

der Speicherersparnis auch weniger Rechenleistung für Lese- und Schreibzugriffe auf den Speicher verwendet.

Bei frühen Tests wurden Ungenauigkeiten bei den Koeffizienten von Filtern mit einer niedrigen Bandmittenfrequenz (etwa 30 Hz) festgestellt, welche die Filter instabil und die Messung unmöglich gemacht haben. Um diese Ungenauigkeit zu senken, werden die Filter in sogenannten *second-order sections* bzw. als Biquad-Filter implementiert, d.h. dass sie aus mehreren Filter zweiter Ordnung zusammengesetzt werden. Oppenheim u. Schafer (1999) beschreiben die Effekte, die durch die Quantisierung der Koeffizienten und Rundungsfehler entstehen können.

2.4.3 Aufbau des DIN EN 61260 Tools

Für die Berechnung der Filter nach DIN EN 61260 (2003) wurde ein MATLAB Skript geschrieben, das die Filterkoeffizienten für alle möglichen Kombinationen aus Sampleraten und Intervalle ermittelt, d.h. es werden Koeffizienten für insgesamt 18 verschiedene Filtersätze berechnet. Diese Filtersätze enthalten Filter mit einer Bandmittenfrequenz von 20 Hz, die niedrigste Frequenz im hörbaren Bereich, bis zur Nyquist-Frequenz, also $Samplerate/2$ Hz. Abhängig von der Samplerate und dem Intervall können das zwischen neun bis mehrere hundert Filter pro Filtersatz sein.

Das Skript besteht im Wesentlichen aus vier Funktionen:

- *ebmFreq* berechnet die exakte Bandmittenfrequenz nach den Gleichungen (3) und (4) aus der Norm.
- *beFreq* berechnet die Bandeckfrequenzen des jeweiligen Filters nach den Gleichungen (5) und (6).
- *relDaemOm* berechnet die normierten Frequenzen nach den Gleichungen (10) und (11), die zur Prüfung der relativen Dämpfung verwendet werden.
- *isValidFilter* überprüft, ob der Betragsfrequenzgang an allen normierten Frequenzen zwischen der kleinsten und größten relativen Dämpfung der ausgewählten Filterklasse liegt. Für die Bestimmung der Grenzen wurden die Werte aus Tabelle 1 (S.13) der Norm verwendet. Ich habe mich für die Filterklasse 0 entschieden, da sie die kleinsten Fehlergrenzen hat.

Für die Berechnung der digitalen Butterworth Bandpass-Filter wurde die MATLAB-Funktion *butter* verwendet. Sie nimmt als Eingabeparameter die Ordnung des Filters, die Bandeckfrequenzen und den Filtertyp an. Zunächst wird die Bandmittenfrequenz des Filters berechnet und die Bandeckfrequenzen ermittelt. Damit wird ein Filter 2. Ordnung erstellt, der mit *isValidFilter* überprüft wird. Entspricht er nicht den Anforderungen für die relative Dämpfung, wird die Ordnung so lange erhöht, bis einer gefunden wird oder die Ordnung einen bestimmten maximalen Wert erreicht. Wird ein Filter gefunden, werden seine Koeffizienten in eine Datei gespeichert, auf die das Tool später zugreifen kann.

2.4.4 Struktur der Filter-Datei

Die Struktur der Datei beinhaltet die für das Tool wichtigsten Informationen für die Erstellung der Filter und die Berechnung des Ausgangssignals. Die Werte werden als Text abgespeichert und sind sowohl durch Leerzeichen als auch durch Zeilenumbrüchen inhaltlich getrennt. Die Struktur eines Filtersatzes für einen Filter mit der Samplerate `sr` und der Bandbreitenkennzahl `bk` sieht folgendermaßen aus:

```
sr bk
length error
n0   sections low mid high gain: g
     b(0,0) b(0,1) b(0,2) a(0,1) a(0,2)
     b(1,0) b(1,1) b(1,2) a(1,1) a(1,2)
     [...]
     b(m,0) b(m,1) b(m,2) a(m,1) a(m,2)
n1   sections low mid high
     b(0,0) b(0,1) b(0,2) a(0,1) a(0,2)
     b(1,0) b(1,1) b(1,2) a(1,1) a(1,2)
     [...]
     b(m,0) b(m,1) b(m,2) a(m,1) a(m,2)
[...]
```

`sections` gibt die Anzahl der *second-order sections* an. Multipliziert man sie mal zwei, erhält man die Ordnung des Filters. `low` und `high` sind die Bandeckfrequenzen, `mid` die exakte Bandmittenfrequenz, `b` sind die Zählerkoeffizienten und `a` die Nennerkoeffizienten der Übertragungsfunkton, `n` ist der Index des jeweiligen Filters, abhängig von der Referenzfrequenz (1000 Hz), und `g` ist der Faktor um den das Ausgangssignal nach der Filterung verstärkt bzw. gedämpft wird. `length` und `error` werden nur für das Parsen der Daten benutzt.

Die Struktur wiederholt sich für jede Filtereinstellung. Die Koeffizienten werden als Gleitkommazahlen mit bis zu 15 Nachkommastellen, Mantisse und Exponent abgebildet, um Rundungsfehler zu vermeiden und den Datentyp *double* möglichst voll auszunutzen.

2.4.5 Aufbau das ITU-R BS.1770 Tools

Für die Berechnung der Keoffzienten des Pre-Filters und RLB-Filters wurde ebenso MATLAB verwendet. Da die Filterkoeffizienten nur für die Samplerate 48 kHz angegeben sind, mussten die Filter für die übrigen Sampleraten anhand der vorgegebenen Filterdesigns rekonstruiert werden. Die Norm gibt vor, dass die Filter in der *Direct Form II* zu implementieren sind (vgl. Boulanger u. Lazzarini, 2011, S. 485).

Leunam (2011) und Brandstätter benutzen zwei unterschiedliche Ansätze zur Rekonstruktion, wobei Leunam's Koeffizienten für eine Samplerate von 48 kHz exakt den Vorgaben entsprechen. Im Rahmen dieser Arbeit habe ich jedoch einen eigenen Ansatz für

die Rekonstruktion versucht, da kleine Abweichungen bei den Koeffizienten nicht entscheidend für den Algorithmus sind. (ITU-R BS.1770, 2006, S. 4)

Bei dem Design des Pre-Filters handelt es sich um einen High-Shelving-Filter, der das Signal ab etwa 4000 Hz um 4 dB verstärkt. Für die Rekonstruktion wurde das *filterbuilder* Tool von MATLAB verwendet, das einen Shelving-Filter anhand der Verstärkung, der Ordnung und der Frequenz am Zentrum des Anstiegs erstellt. Die Frequenz f_c muss folglich bei 2 dB gesucht werden. Dazu wurde der Algorithmus verwendet, der in Kapitel 2.4.6 beschrieben wird ($f_c = 1500,834195202461Hz$).

Die damit berechneten Koeffizienten sind in Tabelle 3 aufgeführt. Den Betragsfrequenzgang sieht man in Abbildung 7. Die Koeffizienten stimmen bis drei Stellen nach dem Komma mit den Vorgaben überein und der Betragsfrequenzgang ist identisch.

Bei der Rekonstruktion des RLB-Filters bin ich zunächst davon ausgegangen, dass es sich um einen einfachen Hochpass-Filter zweiter Ordnung handelt, wie es auch in der Norm spezifiziert wird. Vergleicht man jedoch den angegebenen Betragsfrequenzgang mit dem eines Hochpass-Filters zweiter Ordnung, dann fällt auf, dass sie sich in ihrer Flankensteilheit unterscheiden. Eine Hintereinanderschaltung zweier Butterworth Hochpass-Filter erster Ordnung hat schließlich zum gewünschten Betragsfrequenzgang geführt.

Als nächstes habe ich die Grenzfrequenz angepasst, wobei zwei Punkte zu beachten waren. Die individuelle Grenzfrequenz der Filter liegt bei $\sqrt{1/4}$ bzw. -6.0206 dB. Mit Hilfe von MATLAB wurde ermittelt, dass die maximale Amplitude des vorgegebenen Filters bei 0.043277146260814 dB liegt, d.h. die Grenzfrequenz f_c muss bei einer Amplitude von -5.977322767018810 dB gesucht werden. Auch hier wurde der Algorithmus aus 2.4.6 verwendet ($f_c = 38.110551990990892Hz$). Die Koeffizienten ließen sich dann durch eine Multiplikation der Übertragungsfunktionen beider Filter ermitteln (siehe Tabelle 4). Sie stimmen den Vorgaben mit bis zu sieben Stellen nach dem Komma überein. Der Betragsfrequenzgang ist auch hier identisch.

Darüber hinaus wurden alle Filter mit den Ergebnissen aus Leunam (2011) gegen geprüft. Der Unterschied in der Genauigkeit und die identischen Betragsfrequenzgänge bleiben über alle Sampleraten gleich. Alle Koeffizienten der rekonstruierten Filter sind im Anhang noch einmal komplett aufgelistet.

i	a_i	b_i
0	1.0	1.535155944422577
1	-1.690529840892640	-2.691547631881319
2	0.732378149458886	1.198239996024988

Tabelle 3 : Rekonstruierter Pre-Filter, Koeffizienten, 48 kHz Samplerate

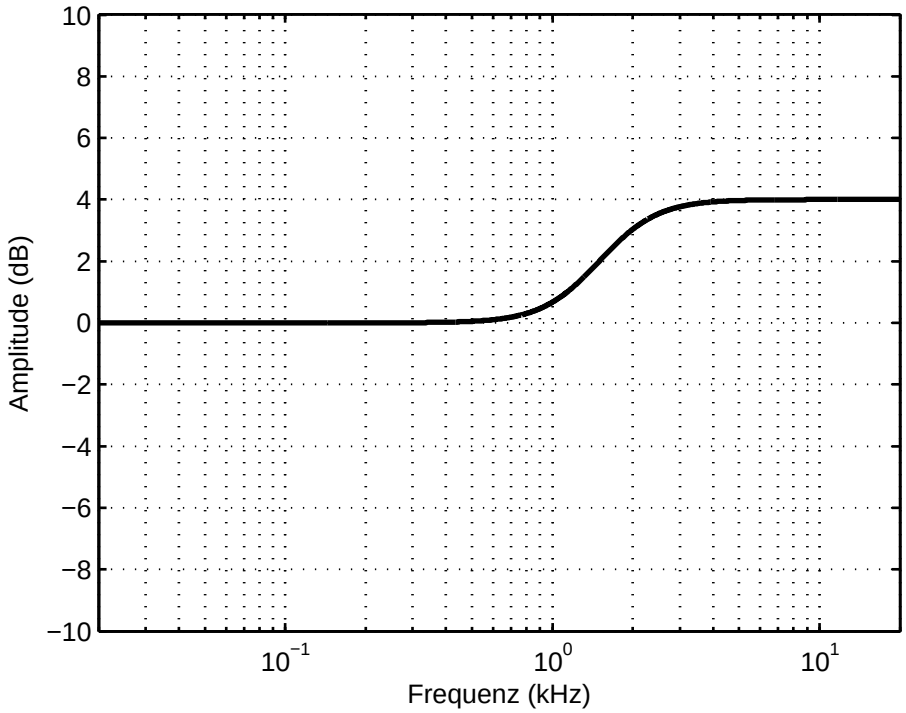

Abbildung 7 : Rekonstruierter Pre-Filter, Betragsfrequenzgang, 48 kHz Samplerate

i	a_i	b_i
0	1.0	1.0
1	-1.990047485034218	-2.0
2	0.990072248172754	1.0

Tabelle 4 : Rekonstruierter RLB-Filter, Koeffizienten, 48 kHz Samplerate

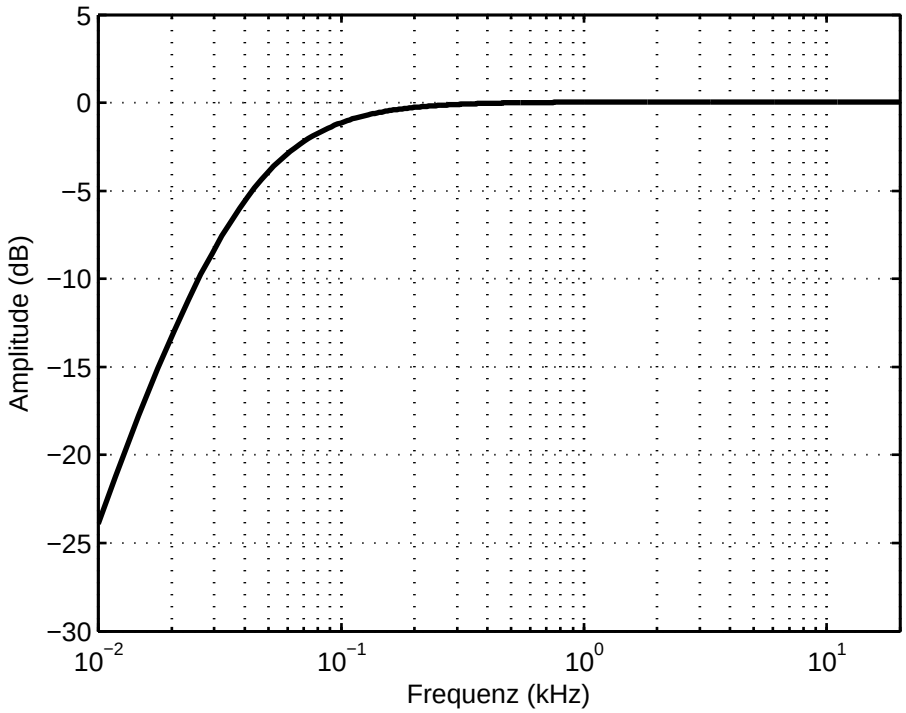

Abbildung 8 : Rekonstruierter RLB-Filter, Betragsfrequenzgang, 48 kHz Samplerate

2.4.6 Algorithmus zur Berechnung der Grenzfrequenz

Für die Rekonstruktion der Filter war es nötig die Grenzfrequenzen zu ermitteln. Dafür habe ich ein Skript geschrieben (getFreqAtMag), das nach dem Intervallhalbierungsverfahren die Frequenz an einer Amplitude des Betragsfrequenzganges bis zu einer bestimmten Genauigkeit approximiert. Dabei beginnt die Funktion bei einer Startfrequenz a_0 und nähert sich der gesuchten Frequenz in mehreren Schritten, indem es a_0 halbiert und das Ergebnis, abhängig von der ermittelten Amplitude, zu a_0 addiert oder subtrahiert. Ist beispielsweise die Amplitude an der Frequenz 12000 Hz zu hoch, wird als nächstes 6000 Hz, dann 3000 Hz und dann 1500 Hz überprüft. Ist sie bei 1500 Hz zu klein, wird im nächsten Schritt die Amplitude bei 2250 Hz berechnet, usw. Für die Berechnung der Amplitude an einer bestimmten Frequenz, wird die MATLAB-Funktion *freqz* benutzt.

$$a_x = \begin{cases} 12000 & , x = 0 \\ a_{x-1} + \frac{a_{x-1}}{2} & , freqz(a_{x-1}) < Amp_{Ziel} \\ a_{x-1} - \frac{a_{x-1}}{2} & , freqz(a_{x-1}) > Amp_{Ziel} \\ a_x & , freqz(a_{x-1}) = Amp_{Ziel} \\ & \quad \vee |Amp_{Ziel} - freqz(a_{x-1})| < \alpha \\ & \quad \vee x > 1000 \end{cases}$$

Es werden so viele Iterationen berechnet, bis entweder die gegebene Amplitude, ein Wert im Bereich α der gegebenen Amplitude oder eine obere Grenze an Schritte erreicht ist. Für die beiden Filter der ITU-R BS.1770 Norm wurde das Intervall von 0 Hz bis 12000 Hz gewählt. Tests haben gezeigt, dass etwa 1000 Iterationen ausreichend sind, um eine brauchbare Approximation zu bekommen. Mehr als 1000 zu berechnen, hatte keinen wesentlichen Effekt auf die Filterkoeffizienten. Bei starken Abweichungen lag der Fehler eher am Filterdesign oder an der Amplitude, an der die Frequenz gesucht wurde.

Dieser numerische Ansatz ist nur bei stetig steigenden Funktionen sinnvoll, d.h. er kann für den Pre-Filter und RLB-Filter benutzt werden, nicht aber für den Bandpassfilter, der zwei Eckfrequenzen hat.

2.4.7 Aufbau des UGens

Für die Implementierung des UGens wurde als Vorlage die Programmstruktur von Abbildung 25.1 in Stowell (2011) verwendet und um zusätzliche Funktionen erweitert.

Damit der Client den UGen aufrufen kann, wurde eine Class-Datei beim Server angelegt, die den Namen des UGens, die Parameter und die verwendeten Raten definiert.

Parameter

Das UGen hat die folgenden Parameter: das Intervall der Filter im Filtersatz, die Samplerate des Audiosignals, die Integrationszeit für die Berechnung des quadratischen Mittelwerts und die Abfallrate (Decay) für die RMS- und Peak-Level-Spitzenwerte.

Die Samplerate wird durch das Marco `SAMPLERATE` vom UGen ausgelesen, sie muss also nicht explizit als Parameter übergeben werden. Das Intervall für den Filtersatz kann entweder eine Oktave, eine Terz oder ein Halbtonschritt sein. Die drei Einstellungen werden durch die Zahlen 0 bis 2 kodiert, wobei 0 für Oktave, 1 für Terz und 2 für Halbtonschritt steht. Bei jeder anderen Eingabe wird standardmäßig eine Oktave als Intervall angenommen. Die Integrationszeit wird in Sekunden angegeben und kann alle Werte ab inkl. 0 Sekunden betragen. Der Decay wird in dB pro Sekunde angegeben. Falls er 0 dB/s beträgt, können die gespeicherten Werte nur noch durch Höhere überschrieben werden. Das entspricht einem Peak-Hold und wird verwendet, um sich die Spitzenwerte zu einem bestimmten Zeitpunkt in Ruhe angucken zu können. Die Decay-Parameter sind die einzigen, die zur Laufzeit über die OSC-Schnittstelle veränderbar sind bzw. an denen sich der UGen anpasst.

Der UGen *struct* verwaltet die Parameter der UGen Instanz, die Filter-Buffer, den RMS-Buffer und die Arrays für das Zwischenspeichern der Ausgabewerte. Darüber hinaus werden Informationen zum RMS-Fenster gespeichert.

Funktion

Die Filterkoeffizienten werden beim Laden des UGens, durch Aufruf der load-Methode, aus der Filter-Datei ausgelesen und auf dem Server gespeichert. Die Arrays mit den Koeffizienten sind global für alle Instanzen des UGens zugänglich, die sich abhängig von ihren Parametern den jeweiligen Filtersatz raus suchen. Da auf die Koeffizienten nur per Leseoperationen zugegriffen wird, ist eine parallele Nutzung weitgehend unproblematisch.

Die Konstruktor-Funktion alloziert die Buffer, die für die Delay-Werte der Filterfunktion und der RMS-Berechnung benötigt werden. Dabei werden an Stelle der C++ üblichen Funktionen *new* und *delete*, die von SuperCollider bereitgestellten Funktionen *RTAlloc* und *RTFree* zur Verwaltung des Speichers benutzt, um Gebrauch von dem *real-time memory pool* zu machen und so die Effizienz des UGens zu steigern. Auf diesem Speicher kann nur über die jeweilige UGen-Instanz zugegriffen werden. Man könnte theoretisch auch die Input- und Output-Buffer des UGens verwenden, um die Delay-Werte für die Filterfunktion zu bekommen, wegen dem *buffer coloring* muss man jedoch eigene Zwischenspeicher benutzen.

Für die Implementierung der Filterfunktion und der Optimierung für Fließkommazahlen, habe ich die Transposed Direct Form II verwendet (Oppenheim u. Schafer, 1999, S. 364). Das Ausgangssignal wird mit der folgenden Gleichung berechnet:

$$
\begin{aligned}
y(n) &= b_0 \cdot x(n) + v_1(n-1) \\
v_1(n) &= a_1 \cdot y(n) + b_1 \cdot x(n) + v_2(n-1) \\
v_2(n) &= a_2 \cdot y(n) + b_2 \cdot x(n)
\end{aligned}
$$

Da der UGen nur mit Direct Form II Biquad-Filter arbeitet, speichert jeder Buffer nur zwei Delay-Werte. Die Anzahl der Buffer pro Filter hängt von der Anzahl der *second-order sections* ab, die in einem Verhältnis von 1:2 zur Ordnung des Filters liegt. Hat der

Filter also die Ordnung sechs, dann werden drei Biquad-Filter benutzt und insgesamt sechs Delay-Werte gespeichert. Es wird in jedem Fall nur so viel Speicher dynamisch alloziert, wie vom Filter benötigt wird. Gleichzeitig passt sich die Filteroperation auf Filter mit verschiedenen Ordnungen an.

Auch die Werte für die RMS-Berechnung werden in separate Buffer zwischengespeichert, die das Integrationsfenster repräsentieren. Da sich die Werte in diesem Fenster zeitlich parallel zur Verarbeitung des Eingangssignals verschieben, wird für die Speicherung ein Ring-Buffer benutzt. Die Größe dieses Buffers hängt von der angegebenen Integrationszeit und der Samplerate ab. Beträgt die Integrationszeit beispielsweise 3 ms, dann muss der Buffer bei einer Samplerate von 44100 Hz genau 133 Werte speichern können. Auf diese Weise kann das Integrationsfenster beliebig groß sein, also auch mehrere Sampleblöcke umfassen. Der quadratische Mittelwert wird dann mit

$$RMS = \sqrt{\frac{y^2(0) + y^2(1) + \ldots + y^2(n)}{N}}$$

gebildet. (vgl. Weinzierl, 2008b, S. 10)

Der Algorithmus für die RMS-Berechnung ist so implementiert, dass er für jedes verarbeitete Sample das quadrat des gefilterten Ausgangssignals in den Ring-Buffer schreibt und gleichzeitig die Summe des Zählers der RMS-Gleichung berechnet, indem er den ältesten Wert im Ring-Buffer von der laufenden Summe subtrahiert und den jüngsten aufaddiert. So bleibt für die eigentliche RMS-Berechnung nur noch die Division und das Wurzelziehen.

Der Peak-Level des Signals entspricht dem maximalen Wert des Betrags aller gefilterten Samplewerte in dem übertragenen Sampleblock. Es werden sowohl von der RMS-Berechnung als auch vom Peak-Level die Spitzenwerte gespeichert. Falls der letzte Spitzenwert kleiner ist, als der gerade berechnete, wird er mit dem neuen Wert überschrieben. Ansonsten wird der letzte Spitzenwert um x dB pro Sekunde vermindert.

Ausgabe

Für die Ausgabe muss von *sclang* speziell ein Buffer alloziert werden, dessen Pointer an den UGen übergeben wird. Auf diese Weise können Server und Client beliebig auf denselben Speicher zugreifen, ohne spezielle Anfragen aneinander senden zu müssen. Im Hintergrund werden natürlich immer noch OSC-Nachrichten verschickt. Der Buffer wird nur zur Übertragung der Werte benutzt d.h. ein Überschreiben der Werte (z.B. vom Client) hat keinen Einfluss auf die Berechnung des Ausgangssignals. Die berechneten Spitzenwerte werden am Ende der next-Funktion in diesen Buffer geschrieben. An erste Stelle steht die Anzahl der Bänder, die der Filtersath hat. Dami weiß der Anwender, wie viele Werte er aus dem Buffer raus zu lesen hat. Dann folgen die RMS-Werte und danach die Peak-Level Werte. So kann der Anwender selber bestimmten, welche Daten er beim Client verwenden möchte.

2.5 Evaluation

Das UGen wurde auf der Linux Distribution Debian 6 64-bit entwickelt und getestet. Als Sound Server wurde JACK/ALSA mit Echtzeitverarbeitung verwendet. Es wurde der Quelltext von SuperCollider 3.4.4 benutzt, inklusive der zu der Zeit vorliegenden Plug-Ins, um das UGen zu kompilieren. Diese Konfiguration unterstützt jedoch nur die Sampleraten 44.1 kHz, 48 kHz und 96 kHz.

Für die Evaluation muss getestet werden, ob das gefilterte Ausgangssignal der einzelnen Butterworth Bandpass-Filter in den Grenzen der relativen Dämpfung für die Filterklasse 0 liegt. Das Signal wird an den normierten Frequenzen entsprechend der Norm geprüft. Als Eingangssignal wird ein Sinusoid mit maximaler Amplitude (± 1.0) verwendet und auf jeder Testfrequenz für eine Sekunde gehalten. Die Integrationszeit wird mit 1 ms und der Decay mit 60.0 dB pro Sekunde angegeben. Da der quadratische Mittelwert des Signals betrachtet wird, muss die Ausgabe der RTA-Analyse mit dem Scheitelfaktor für Sinus-förmige Signale multipliziert werden, um die tatsächlichen Spitzenwerte des Signals zu erhalten. (vgl. Weinzierl, 2008b, S. 14)

Aus den Ergebnissen der Tests lässt sich erkennen, dass bei Filter mit einer niedrigen Bandmittenfrequenz die Lautheit an den Testfrequenzen nicht in den Grenzen der relativen Dämpfung liegt. Dabei kann es sich um Abweichungen von 0.5 bis 2 dB handeln. Erst ab einer bestimmten Bandmittenfrequenz entsprechen die Filter den Vorgaben. Davor ist entweder die Flankensteilheit nicht in jedem Fall richtig, das Ausgangssignal übersteigt zur Bandmitte hin die obere Grenze von 0.15 dB oder beides. Sporadisch gibt es Werte die in den Grenzen liegen, aber das ändert sich von Test zu Test, woraus man schließen kann, dass diese Werte schwanken. In den gesammelten Testdaten treten auch Fehler bei Testfrequenzen auf, die über der Nyquist-Frequenz liegen, was jedoch mit dem Sampling-Theorem zu erklären ist. Damit sind diese Fehler für die Evaluation nicht weiter ausschlaggebend.

Die Oktav-Filter mit der Samplerate 44100 Hz entsprechen erst ab dem Index -1 (der Bandmittenfrequenz 501 Hz) den Anforderungen der Norm d.h. von 9 Filter sind 4 mangelhaft. Im Fall der Terz-Filter sind von insgesamt 29 Filter 9 mangelhaft, wobei die Tests ab dem Index -7 (der Bandmittenfrequenz 199 Hz) für alle Testfrequenzen positiv verlaufen. Für Halbtonschritt-Filter liegt die Grenze zwischen Fehlerhaften und Norm-gerechten Filtern beim Index -17 (386 Hz).

Die häufigsten Fehler entstehen zur Bandmitte hin, wobei die Abweichung vom Soll-Wert größer wird je niedriger die Testfrequenz ist. Dasselbe gilt auch für Filter mit den Sampleraten 88200 Hz und 96000 Hz. Dort verschieben sich natürlich die Grenzen etwas. Eine Erhöhung der Filterklasse würde in einzelnen Fällen zu besseren Testergebnissen führen, aber das wäre keine klare Lösung.

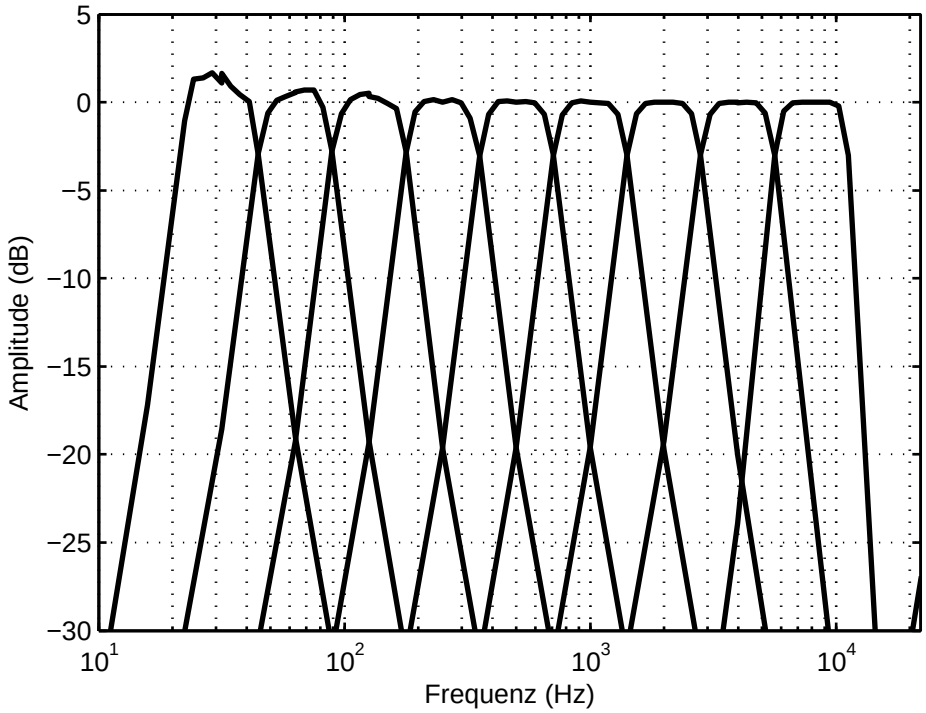

Abbildung 9 : Testwerte des Oktav-Filtersatzes, Samplerate 44100 Hz

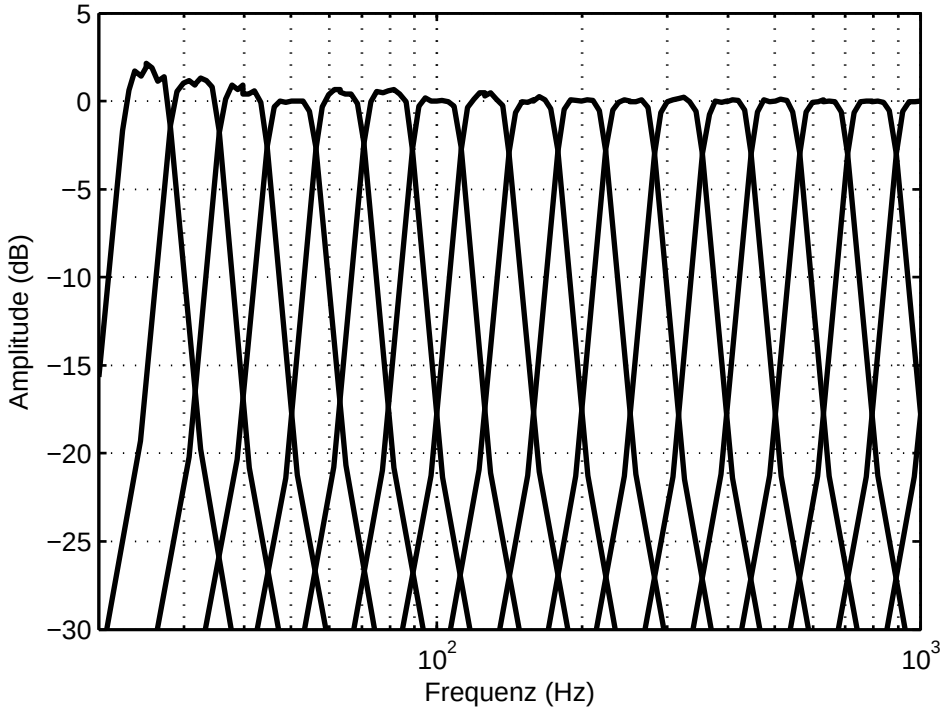

Abbildung 10 : Ausschnitt aus dem Terz-Filtersatz, Samplerate 44100 Hz

Der Grund für diese starken Schwankungen im Bereich der Bandmittenfrequenz ließ sich darauf zurückführen, dass das UGen eine statische Integrationszeit für alle Filter benutzt. Bei einem Testdurchlauf der 44100 Hz Terz-Filter mit einer Integrationszeit von 100 ms sind alle Tests, bis auf eine Testfrequenzen, positiv verlaufen. Das Problem entsteht durch die niedrige Frequenz der Sinus-Funktion im direkten Bezug zum Integrationsfenster, was den Effekt hat, dass die gemessene Lautheit anfängt die Sinus-Funktion abzubilden. Um das zu lösen, muss die Integrationszeit variabel zur Bandmittenfrequenz des Filters gewählt werden. Hier liegt ein Grundlegender Fehler bei dem Design des UGens vor.

Für den Test der ITU-R BS.1770 Filter wurde die Testfrequenz in Halbtonschritten inkrementiert. Der Testablauf selbst blieb gleich. Das Ergebnis des Tests wird in Abb. 11 dargestellt, aus dem sich der RLB- und Pre-Filter rauslesen lässt. Auch hier sind die Schwankungen durch die Integrationszeit zu erklären.

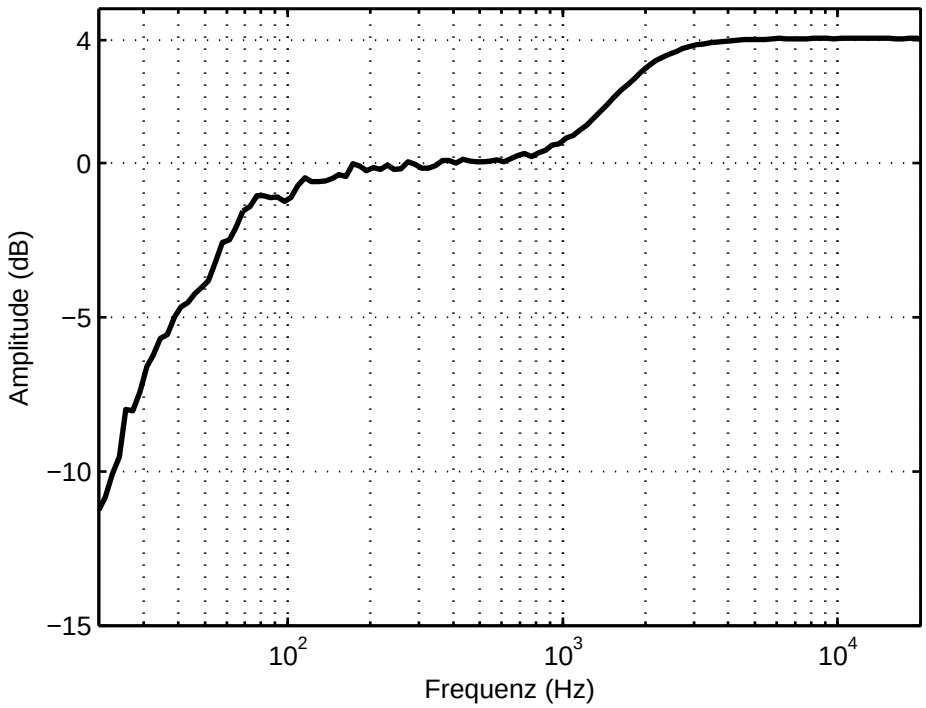

Abbildung 11 : Pre-Filter und RLB-Filter Ausgabe im Bereich von 20 Hz bis 20 kHz, Samplerate 44100 Hz

Zusätzlich wurde die CPU-Auslastung von SuperCollider während der Tests aufgezeichnet.

	Leerlauf	Oktave	Terz	Halbton
44100 Hz	3%	5% (9)	10% (29)	40% (121)
48000 Hz	3%	5.5% (10)	10.5% (30)	43% (123)
96000 Hz	5%	11% (11)	25%(33)	95% (135)

Tabelle 5 : UGen Auslastung bei 128 Blockgröße mit $\pm 1.0\%$ Abweichung

3 Fazit

Leider können in dieser Arbeit keine festen Aussagen über die in der Einleitung gesetzten Ziele getroffen werden, da das UGen in seiner jetzigen Form kein oder nur ein teilweise zuverlässiges Metering von Audioinhalten zulässt. Das hat in erster Linie weniger mit der Implementierung der Filter zu tun, als mit den ungenauen Messergebnissen, die durch die interne, statische Integrationszeit des Tools entsteht. Die Filter, die nicht davon betroffen sind, entsprechen jedoch den Vorgaben der Norm. Auch die Lautheitsanpassung erfolgt erwartungsgemäß.

Die Tests wurden nur auf einer ganz bestimmten Konfiguration von Hardware und Software durchgeführt, die nicht für die Audioverarbeitung optimiert ist. Die Evaluation des UGens in einer richtigen Studioumgebung war im Rahmen dieser Arbeit nicht möglich, deshalb sind die Ergebnisse nicht repräsentativ für den Einsatz des Tools in einem Tonstudio. An der Stelle müssen noch weitere Tests durchgeführt werden, um festzustellen, ob sich die Filter auch bei niedrigen Frequenzen stabil verhalten, wenn sich die Samplerate deutlich erhöht.

Literatur

[Boulanger u. Lazzarini 2011] BOULANGER, R. (Hrsg.) ; LAZZARINI, V. (Hrsg.): *The Audio Programming Book*. Cambridge, MA : MIT Press, 2011

[Brandstätter] BRANDSTÄTTER, U: *Implementing Methods for Automatic Volume Equalization*. – `http://www.cp.jku.at/people/seyerlehner/supervised/brandstaetter_project.pdf`

[DIN EN 61260 2003] Norm EN 61260 März 2003. *Bandfilter für Oktaven und Bruchteile von Oktaven*

[ITU-R BS.1770 2006] Norm BS.1770 2006. *Algorithms to measure audio programme loudness an true-peak audio level*

[Lerch u. Weinzierl 2008] *Kapitel* Digitale Audiotechnik: Grundlagen. In: LERCH, A. ; WEINZIERL, S.: *Handbuch der Audiotechnik*. Berlin : Springer, 2008, S. 785–811

[Leunam 2011] LEUNAM, V.: *ITU-R BS.1770-1 filter specifications*. January 2011. – `http://scribd.com/doc/49991813/ITU-R-BS-1770-1-filters`

[Müller 2008] *Kapitel* Messtechnik. In: MÜLLER, S.: *Handbuch der Audiotechnik*. Berlin : Springer, 2008, S. 1087–1169

[Oppenheim u. Schafer 1999] OPPENHEIM, A. ; SCHAFER, R.: *Discrete-Time Signal Processing*. 2. Edition. Upper Saddle River, NJ : Prentice Hall, 1999

[Parmenter 2011] *Kapitel* 2: The Unit Generator. In: PARMENTER, J.: *The SuperCollider Book*. Cambridge, MA : MIT Press, 2011, S. 55–80

[SCCode 2012] *Client versus Server Architecture and Operations*. December 2012. – `http://doc.sccode.org/Guides/ClientVsServer.html`

[Stowell 2011] *Kapitel* Writing Unit Generator Plug-ins. In: STOWELL, D.: *The SuperCollider Book*. Cambridge, MA : MIT Press, 2011, S. 691–720

[Weinzierl 2006] WEINZIERL, S.: *Kommunikationswissenschaft II: Audiosignale und Kontrollinstrumente*. 2006. – `http://www.ak.tu-berlin.de/fileadmin/a0135/Unterrichtsmaterial/Radio100K_SS2008/kwIIskript.pdf`

[Weinzierl 2008a] *Kapitel* Aufnahmeverfahren. In: WEINZIERL, S.: *Handbuch der Audiotechnik*. Berlin : Springer, 2008, S. 551–607

[Weinzierl 2008b] *Kapitel* Grundlagen. In: WEINZIERL, S.: *Handbuch der Audiotechnik*. Berlin : Springer, 2008, S. 1–39

[Wright u. a. 2003] WRIGHT, M. ; FREED, A. ; MOMENI, A.: OpenSound Control: State of the Art 2003. In: *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03)*. Montreal, Canada, 2003, S. 153–159

[Zölzer 2002] ZÖLZER, U. (Hrsg.): *DAFX - Digital Audio Effects*. 1. Edition. Hoboken, NJ : John Wiley & Sons, 2002

[Zölzer 2008] *Kapitel* Digitale Signalverarbeitung, Filter und Effekte. In: ZÖLZER, U.: *Handbuch der Audiotechnik*. Berlin : Springer, 2008, S. 813–848

A Anhang

Alle MATLAB-Skripte, die Filter-Datei und der Quelltext des UGens liegen der Arbeit als CD-ROM bei.

Samplerate	i	a_i	b_i
Pre-Filter Koeffizienten			
44100 Hz	0	1.0	1.530880422010828
	1	-1.663565955257725	-2.650891852219996
	2	0.712523593016214	1.168969067967657
48000 Hz	0	1.0	1.535155944422577
	1	-1.690529840892640	-2.691547631881319
	2	0.732378149458886	1.198239996024988
88200 Hz	0	1.0	1.557529940424599
	1	-1.830748536798077	-2.905414624574983
	2	0.843994481683384	1.361130629035691
96000 Hz	0	1.0	1.559728983978739
	1	-1.844445689161430	-2.926540864510010
	2	0.855700557353563	1.378066748723405
176400 Hz	0	1.0	1.571134377856920
	1	-1.915226969831173	-3.036439120366919
	2	0.918676946259591	1.468754718938417
192000 Hz	0	1.0	1.572246924318443
	1	-1.922107381433912	-3.047189422656538
	2	0.925029346174441	1.477864463078623
RLB-Filter Koeffizienten			
44100 Hz	0	1.0	1.0
	1	-1.989169709593363	-2.0
	2	0.989199033390936	1.0
48000 Hz	0	1.0	1.0
	1	-1.990047485034218	-2.0
	2	0.990072248172754	1.0
88200 Hz	0	1.0	1.0
	1	-1.994577523887741	-2.0
	2	0.994584874699538	1.0
96000 Hz	0	1.0	1.0
	1	-1.995017551761315	-2.0
	2	0.995023757958928	1.0
176400 Hz	0	1.0	1.0
	1	-1.997286924243459	-2.0
	2	0.997288764438474	1.0
192000 Hz	0	1.0	1.0
	1	-1.997507224333063	-2.0
	2	0.997508777815694	1.0

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit selbstständig und ohne unerlaubte Hilfe angefertigt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommen Stellen als solche kenntlich gemacht habe.

Berlin, 03.01.2013