STUDER
professional audio equipment

HOCHSCHULE DER MEDIEN

Entwicklung eines hybriden Hallalgorithmus zur Integration in ein Surround-Raumsimulationsverfahren

Bachelorarbeit im Studiengang Audiovisuelle Medien
Hochschule der Medien Stuttgart

vorgelegt von
Andreas Jäger
am 15. September 2008

Erstprüfer: Prof. Oliver Curdt
Zweitprüfer: Attila Karamustafaoglu

Abstract

Zur synthetischen Hallerzeugung werden zwei verschiedene Ansätze verfolgt: durch Faltung des Audiosignals mit einer Raumimpulsantwort kann das Ausklingverhalten eines realen Raums exakt reproduziert werden. Herkömmliche Hallalgorithmen verwenden dagegen rückgekoppelte Strukturen, um ressourcenschonend eine hohe Zahl von Reflexionen zu simulieren und so ebenfalls Hall zu erzeugen.

Hybride Hallalgorithmen kombinieren beide Verfahren, um effizient authentische Ergebnisse liefern zu können. Diese Arbeit dokumentiert die Entwicklung eines solchen Algorithmus, der kurze Faltungen zur Erzeugung der frühen Bestandteile des Schallfelds mit einer rückgekoppelten Struktur zur Berechnung des späten Halls kombiniert.

Dem Anwender steht im resultierenden Verfahren frei, die Nachhallzeit sowie Dämpfungen verschiedener Frequenzbereiche unabhängig von der verwendeten Raumimpulsantwort zu beeinflussen.

Durch Modifikation der zu Grunde liegenden Raumimpulsantwort wird der Algorithmus in ein bestehendes Verfahren zur Raumsimulation integriert.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst habe. Wörtlich oder sinngemäß übernommenes Gedankengut wurde im Text gekennzeichnet. Sämtliche verwendeten Quellen sind im Literaturverzeichnis nachgewiesen.

Stuttgart, den 15. September 2008

Andreas Jäger

Danksagung

Für die Möglichkeit, diese Arbeit zu erstellen, das in mich gesetzte Vertrauen und fortwährende Unterstützung danke ich ganz besonders

Prof. Oliver Curdt,

Attila Karamustafaoglu und nicht zuletzt

Lorenz Bucher.

Die Arbeit wäre auch nicht entstanden ohne

Marcel Babazadeh und

Stefan Ledergerber,

denen ich ebenfalls zu Dank verpflichtet bin.

Für die spontan überlassenen, dringend benötigten Raumimpulsantworten danke ich

Ralph Kessler und der Firma ***Pinguin.***

Für viele andere Dinge gilt mein Dank

meiner Familie und

Lisa.

Inhaltsverzeichnis

1 Einleitung

Im Bereich der Tonstudioausrüstung lässt sich bereits seit Jahren ein Zusammenwachsen ehemals getrennter Komponenten beobachten. Ein einzelner PC mit einer DAW[1]-Software und entsprechenden Plugins kann heute grundsätzlich alle Aufgaben übernehmen, für die früher Mischpult, Zuspieler und externe Signalprozessoren im Verbund eingesetzt wurden. Auch im High-End-Bereich vollzieht sich eine ähnliche Entwicklung hin zu multifunktionalen Geräten. Das bekannteste Beispiel hierfür ist wohl mit dem System 6000 das „Flaggschiff“ von TC Electronics, das sowohl die Aufgaben eines Hallgeräts als auch die eines Mastering-Signalprozessors übernehmen kann.

Die Firma Studer integrierte bereits vor mehreren Jahren in ihren Digitalmischpulten der Vista-Serie sowie deren Vorgänger, dem D950, mit dem sogenannten Virtual Surround Panning (VSP) einen Algorithmus, der zur räumlichen Positionierung von Schallquellen neben Amplituden- und Laufzeitunterschieden das Signal mit frühen Reflexionen und einer diffusen Hallfahne versah. Damit „wilderte“ VSP im Revier von externen Hallgeräten und bot durch die Kopplung von Panning und Raumsimulation Möglichkeiten, die denen herkömmlicher Hallgeräte überlegen waren. Leider musste VSP beim Umstieg auf eine neue Generation von DSP-Cores in den Vista-Systemen stark reduziert werden, so dass Reflexionen und Nachhall in aktuellen Mischpulten nicht mehr zur Verfügung stehen.

Derzeit laufen die Planungen für eine Neuauflage des vollständigen VSP in der nächsten Generation von DSP-Cores. Gerade auf dem Gebiet der Hallerzeugung hat sich seit der Entwicklung des Algorithmus jedoch eine rasante Entwicklung vollzogen. Besonders bemerkenswert ist dabei die Möglichkeit, mittels Faltung einem trockenen Signal den Klang eines beliebigen Raums exakt aufzuprägen. Soll dies aber für potentiell große Hallzeiten und gleichzeitig sehr ressourcenschonend geschehen – und das sind die Anforderungen an VSP – stößt man auch heute noch an die Grenzen des technisch Machbaren und Sinnvollen.

[1] Digital Audio Workstation

Hier kann ein hybrider Algorithmus Abhilfe schaffen, indem er einerseits durch Einbindung ressourcenschonender Teilalgorithmen für die späten Hallanteile die Anforderungen an den Rechner reduziert, andererseits aber die hohe subjektive Qualität eines Faltungshalls bewahrt. Einen solchen Algorithmus zu entwickeln, der im Zusammenspiel mit dem Amplituden- und Laufzeitpanning sowie der Reflexionserzeugung von VSP eine vollständige Raumsimulation bilden soll, ist das Ziel dieser Arbeit.

Im folgenden Kapitel soll zunächst ein Überblick über die nötigen Grundlagen von Raumakustik, Digitalfiltern und Hallalgorithmen gegeben werden. Die Kapitel 3 und 4 beschäftigen sich kurz mit dem vorhandenen VSP-Algorithmus sowie der dem Hybridalgorithmus zu Grunde liegenden Raumimpulsantwort.

Die eigentliche Entwicklungsarbeit wird in den Kapiteln 5 und 6 beschrieben. Kapitel 7 dokumentiert die entstandene Hybridstruktur. In Kapitel 8 erfolgt eine kurze Einschätzung des Ergebnisses. Die Arbeit schließt mit einem Ausblick auf mögliche Verbesserungen.

2 Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen der vorliegenden Arbeit. Dies umfasst einerseits die elementarsten Zusammenhänge der Raumakustik, andererseits die wichtigsten Fakten zu Digitalfiltern und Hallalgorithmen.

2.1 Raumakustik

Grundsätzlich entsteht das Schallfeld, das sich in Folge eines Schallsignals im Innern eines Raums ausbreitet (Abbildung 2.1), durch das Auftreten von Reflexionen dieses Signals. Deren Ausbildung basiert auf dem Zusammenspiel folgender physikalischer Phänomene:

Reflexion: trifft Schall auf eine genügend große Fläche, so wird er an ihr reflektiert. Ist die Einfallsrichtung des Schalls dabei senkrecht zur Fläche, wird genau entgegengesetzt seiner ursprünglichen Richtng zurück geworfen. Ist dies nicht der Fall oder ist die Fläche selbst gekrümmt, erfolgt die Reflexion in einem davon abhängigen Winkel [5, S. 143 f.].

Absorption: beim Auftreffen des Schalls auf die Reflexionfläche wird ein bestimmter Anteil der eingetroffenen Schallenergie absorbiert. Der Absorptionsgrad α ist abhängig von der Beschaffenheit der jeweiligen Fläche und in der Regel nicht für alle Frequenzen gleich [5, S. 147].

Beugung: ist die Fläche im Verhältnis zur Wellenlänge des Schalls klein, findet keine Reflexion statt. Stattdessen wird der Schall um die Fläche oder den Gegenstand gebeugt [5, S. 144].

Die ersten auftretenden Reflexionen werden als *frühe Reflexionen* bezeichnet. Ihre exakte Ausprägung (Verzögerung, Stärke, spektrale Komponenten und Einfallsrichtung) ist maßgeblich für den empfundenen Klang und beeinflusst beispielsweise die Räumlichkeit einer Darbeitung und die empfundene Größe eines Raums [4, S. 28] [5, S. 155ff.].

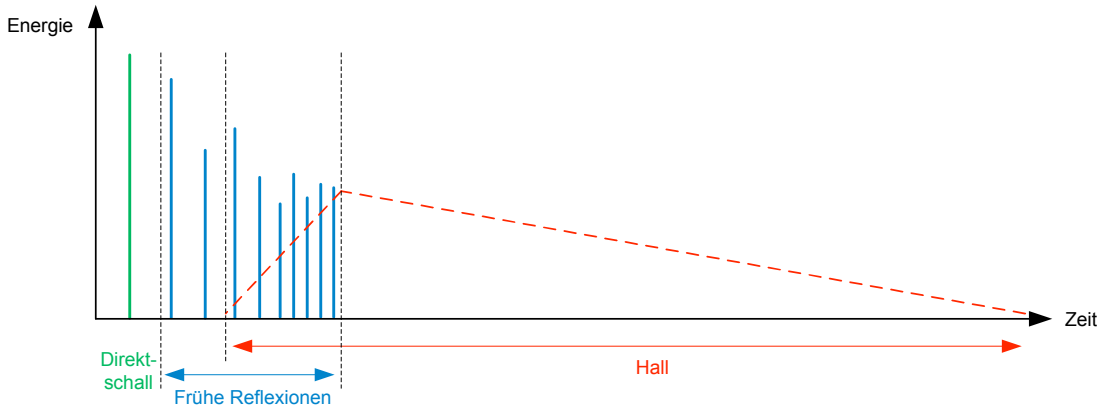


Abbildung 2.1: Schematischer Aufbau des Schallfelds

Mit zunehmender Zeit verdichten sich die Reflexionen sehr schnell, bis sie schließlich nicht mehr voneinander unterschieden werden können und zu einem einheitlichen Klang verschmelzen. Man spricht vom *Hall*. Durch die größere Zahl von Reflexionsvorgängen pro Zeiteinheit erfolgt seine Ausbildung in kleinen Räumen schneller als in Großen. Eine große Zahl von Streukörpern, Säulen etc. kann sie ebenfalls begünstigen [4, S. 25]. Damit einher gehen eine schnellere Abnahme der Energie des diffusen Schallfelds und damit kürzere Nachhallzeiten (s.u.) [4, S. 31], denn jede Reflexion geht auch mit einer teilweisen Absorption einher.

Der im Hinblick auf seine Akustik wichtigste Parameter eines Raums ist die sog. *Nachhallzeit* t_{60}. Sie ist definiert als die Zeit, in der der Schallpegel um 60 dB gegenüber seinem Anfangswert absinkt – was in etwa der Dynamik eines großen Orchesters entspricht [5, S. 148]. Sie beträgt in Opernhäusern ca. 1-1,5 Sekunden [5, S. 184f.], in großen Kirchen für bestimmte Frequenzen bis zu 13 Sekunden [5, S. 191].

Diese Frequenzabhängigkeit des Nachhalls ist im unterschiedlichen Absorptions-, Reflexions- und Beugungsverhalten von Oberflächen bezüglich verschiedener Frequenzanteile des Schalls begründet – denn unter Umständen werden tiefe Frequenzen eines Schallsignals noch gebeugt, hohe aber bereits reflektiert [5, S. 144]. Für hohe Frequenzen ist die Hallzeit dabei grundsätzlich kürzer als für niedrige. Dies liegt vor allem an der stärkeren Absorption hoher Frequenzen durch die Luft [4, S. 35].

2.2 Digitale Filter und Hallalgorithmen

Hallalgorithmen werden in erster Linie nach der ihnen zu Grunde liegenden Variante von Digitalfiltern unterschieden. Während ein Faltungshall auf dem Prinzip des FIR[1]-Filters beruht, nutzt ein algorithmischer Hall zur Erzeugung des späten Nachhalls IIR[2]-Filter. Da die Unterscheidung dieser beiden Arten von Digitalfiltern für die vorliegende Arbeit essentiell ist, sollen zunächst beide Varianten jeweils im Kontext der synthetischen Hallerzeugung erläutert werden. Während dies beim FIR-Filter bzw. Faltungshall in einem einzelnen Abschnitt erfolgt, macht beim algorithmischen Hall

- die größere Anzahl der ihm zu Grunde liegenden IIR-Filtern
- die grundsätzlich andere Denkweise hinter seiner Implementierung und schließlich
- die Vielfalt algorithmischer Verfahren zur Raumsimulation

eine getrennte Betrachtung all dieser Bereiche notwendig. Diese erfolgt in den Abschnitten 2.2.2 bis 2.2.4. Das Kapitel schließt mit einer Zusammenfassung der Vor- und Nachteile von FIR- und IIR-Verfahren aus Anwender- und Entwicklersicht, die schließlich zum Wunsch nach einer Hybridvariante führen.

Anmerkung:

Der Begriff „algorithmischer Hall" bezeichnet allgemein ein konventionelles, auf IIR-Filtern basierendes Verfahren zur Erzeugung künstlichen Halls. In dieser Arbeit werden zur besseren Unterscheidung der Bestandteile des Hybridverfahrens bei Bedarf die Bezeichnungen „IIR-Hallalgorithmus" für algorithmischen Hall bzw. „FIR-Hallalgorithmus" für Faltungshall verwendet. Der Begriff „Hallalgorithmus" meint in dieser Arbeit beide Grundformen.

2.2.1 FIR-Filter und Faltungshall

Jedes lineare zeitinvariante System[3] wird vollständig durch seine Impulsantwort[4] beschrieben. Kennt man die Impulsantwort eines solchen Systems, so kann man dessen

[1] Finite Impulse Response

[2] Infinite Impulse Response

[3] Ein System, dessen Ausgangssignal linear von der Amplitude des Eingangssignals abhängt und unabhängig vom Zeitpunkt des Eintreffens des Signals ist [18] [6, S. 89]. Das gilt im Audiobereich zum Beispiel für Equalizer, nicht aber für Regelverstärker.

[4] Das Ausgangssignal, das entsteht, wenn das System mit einem (theoretisch) unendlich kurzen Impuls (Dirac-Stoß) angeregt wird. Im digitalen Bereich ist dies eine einzelne „1" gefolgt von Nullen.

Wirkung auf ein beliebiges Signal durch Faltung des Signals mit ihr mathematisch korrekt nachbilden [6, S. 108 ff.].[5]

Auch ein ganzer Raum lässt sich akustisch als ein solches System sehen. Damit ist es möglich, durch Faltung eines Signals mit der Raumimpulsantwort dessen Akustik exakt zu reproduzieren, ohne seine inneren Zusammenhänge und Wirkungsweisen nachvollziehen zu müssen [10, S. 2].

Vom technischen Standpunkt unterscheidet sich ein FIR-Hallalgorithmus – also ein Faltungshall – nicht von jedem anderen FIR-Filter wie beispielsweise einem Tiefpass. Der Unterschied liegt lediglich in der Länge und Art der Impulsantwort [6, S. 262]. Denn während herkömmliche FIR-Filter mit einer errechneten Impulsantwort von einigen wenigen Punkten auskommen, benötigt ein Faltungshall eine vor Ort ausgemessene oder synthetisch erzeugte, äußerst lange Impulsantwort. Die Größe des benötigten Filters lässt sich berechnen als

$$\textit{Größe des Filters} = \textit{Raumimpulsantwort in Sekunden} \cdot \textit{Samplingfrequenz}$$

Damit bedeutet schon eine (typische) Dauer von 2 Sekunden bis zum vollständigen Verstummen des Halls bei einer Samplingfrequenz von 48kHz eine Faltungsgröße von 96000 Punkten und damit den enormen Aufwand von 96000 Multiplikationen und Additionen pro Sample des Audiosignals. Da der Rechenaufwand quadratisch von der Länge der Impulsantwort abhängt, erreicht auch moderne Hardware bei diesem Verfahren schnell ihre Grenzen.

Allerdings gibt es eine sehr effiziente Alternative: die sogenannte *Schnelle Faltung* beruht auf dem Grundsatz, dass eine Faltung zweier Signale im Zeitbereich einer Multiplikation ihrer Spektren entspricht [6, S. 312]. Ein Algorithmus, der sowohl Eingangssignal als auch Impulsantwort mittels einer FFT[6] in den Frequenzbereich überführt, dort ihre Spektren multipliziert und das Ergebnis wieder rücktransformiert, ist der oben genannten direkten Faltung schon ab einer Filtergröße von 30 bis 60 Samples überlegen [8, S. 13] [6, S. 311]. Erst durch diese Methode ist die Implementierung eines Faltungshalls in Echtzeit heute überhaupt möglich.

Grundsätzlich lässt sich auch auf anderen Wegen noch Rechenaufwand sparen: so kann beispielsweise die Auflösung der Impulsantwort im späteren Bereich des Halls redu-

[5] Eine sehr anschauliche Erklärung von linearen Systemen, ihren Eigenschaften und Kriterien sowie zur Faltung findet sich in [6], Kap. 5 - 7.

[6] Fast Fourier Transform

ziert werden [8, S. 67]. Dennoch bleibt der Faltungshall auch heute noch eine aufwändige Alternative.

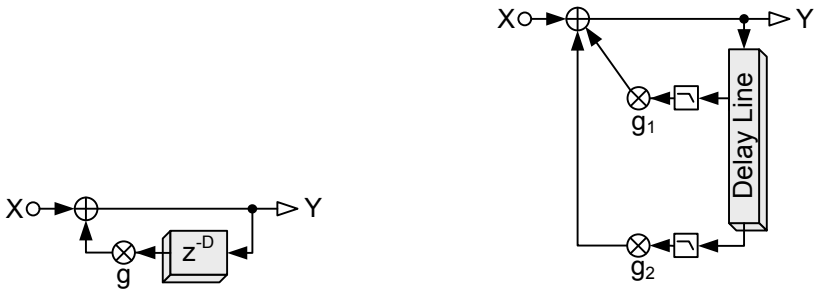
2.2.2 Konzept algorithmischen Halls

Wie erwähnt benötigt ein Faltungshall keinerlei Kenntniss über die physikalischen Zusammenhänge eines Raums, sondern betrachtet das System von außen als eine einzige „Black Box“. Die Philosophie hinter einem algorithmischen Hall ist eine völlig andere, denn er modelliert die Akustik eines Raums „von innen heraus“, indem er den Gesamtklang aus dem Zusammenwirken der zu Grunde liegenden Phänomene modelliert. Ziel bei seiner Entwicklung ist dabei, gute Näherungen für einzelne Teilaspekte der Raumakustik zu finden und so ein Modell zu schaffen, das zwar physikalisch nicht korrekt, subjektiv aber überzeugend und darüber hinaus in der Berechnung effizient ist.

Wichtigste Anforderung an einen solchen Algorithmus ist dabei neben der Generierung früher Reflexionen die Simulation des diffusen Reflexionsverhaltens eines Raums und damit die Erzeugung einer sehr großen Zahl von Echos. Hierfür kommen IIR-Filter zum Einsatz, die sich von FIR-Filtern erheblich unterscheiden. Denn während ein FIR-Filter nur aus dem Direktweg sowie Verzögerungs- und Skalierungselementen besteht, beeinflusst ein IIR-Filter das Eingangssignal selbst durch Rückkopplung des Ausgangs. Daher auch sein Name, denn in der Theorie hat ein solches Filter eine unendlich lange Impulsantwort.

Derartige Hallalgorithmen sind generell ressourcenschonender als ein Faltungshall. Dies hat zwei Gründe:

- der Entwickler kann selbst entscheiden, wie akkurat er bei der Nachbildung der einzelnen akustischen Teilphänomene vorgehen muss und will. Jeder algorithmische Hall macht sich dies zu Nutze, indem er zum Beispiel die wichtigen frühen Reflexionen wesentlich präziser abbildet als den diffusen Hall. Das ermöglicht eine ressourcenschonende Implementierung.
- IIR-Filter sind darüber hinaus grundsätzlich effizienter als FIR-Filter gleicher Wirkung [6, S. 320, 348]. Ihr Einsatz ist aber beim Faltungshall prinzipbedingt ausgeschlossen.



(a) einfache Form

(b) Variante mit zwei Feedbacks, jeweils mit Tiefpass

Abbildung 2.2: Kammfilter

2.2.3 IIR-Filter für algorithmischen Hall

Kammfilter

Ein Kammfilter besteht in seiner Grundform lediglich aus einem einzigen skalierten und verzögerten Rückführungspfad sowie einem Addierer (Abbildung 2.2(a))[7]. Für $g > 1$ ist das Filter instabil, eine Anregung bewirkt also ein Ansteigen des Ausgangssignals ins Unendliche. Gilt $g = 1$, bewirkt eine Anregung eine unendlich lange Reaktion. Damit muss in der Praxis immer $g < 1$ gelten.

Entscheidend für den Klang des Kammfilters ist die Kombination aus Delay- und Skalierungswerten. Eine kurze Verzögerung bewirkt starke und sehr deutlich hörbare spektrale Verfärbungen, deren genaue Ausprägung von der gewählten Länge des Delays abhängt. Im Fall von Hallalgorithmen ist dies grundsätzlich unerwünscht. Ein langes Delay verursacht dagegen diskrete Echos. Diese sind zwar grundsätzlich gewollt, in ihrer typischerweise sehr regelmäßigen Ausprägung aber ebenfalls problematisch.

Die Skalierung des Feedbacks beeinflusst die Stärke dieser Effekte. Eine hohe Skalierung bedeutet also stärkere Verfärbungen oder langsamer abnehmende und damit insgesamt mehr erzeugte Echos, während eine niedrige Skalierung die Wirkung des Filters reduziert.

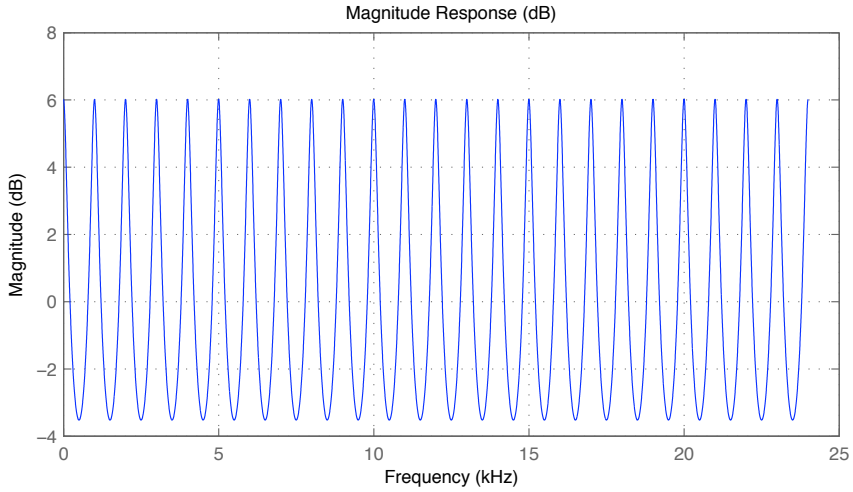
Durch Einsetzen eines Tiefpassfilters im Rückführungspfad kann die Absorption eines Raums hin zu hohen Frequenzen simuliert werden. Eine weitere Option ist die

[7] Kammfilter können auch über eine Feedforward-Struktur realisiert werden [16], sind in dieser Variante für die vorliegende Arbeit jedoch ohne Bedeutung.

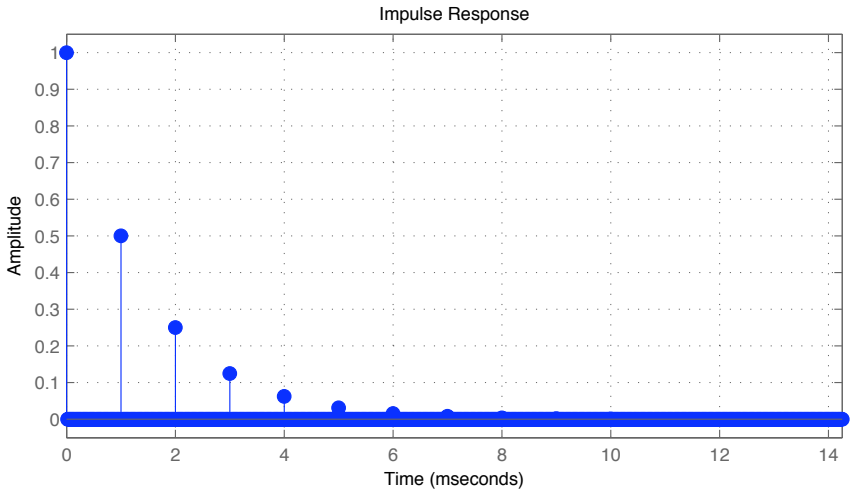
Verwendung mehrerer paralleler Rückführungspfade. Dabei erhält jeder der Feedback-Wege eine unterschiedliche Verzögerung (in der Praxis durch eine einzige lange Delay-Line mit mehreren Abgriffen realisiert) sowie eigene Skalierungs- und gegebenenfalls Tiefpass-Werte. Dies bedingt eine deutliche Erhöhung der erzeugten Wiederholungn, denn jedes Signal im Direktweg (und damit auch jede Wiederholung selbst) verursacht so viele weitere Echos, wie Feedback-Wege vorhanden sind[8]. Die Stabilität eines solchen Systems ist garantiert, so lange die Summe aller Skalierungswerte maximal 1 ist [8, S. 33].

Abbildung 2.3 zeigt exemplarisch den Frequenzgang sowie die Impulsantwort eines einfachen Kammfilters (D = 1 ms, g = 0,5). In Abbildung 2.4 ist analog das Verhalten eines Kammfilters mit zwei parallelen Rückkopplungswegen (D_1 = 1 ms, $D_2 = \frac{13}{48}$ ms; g_1 = 0,5, g_2 = 0,3) dargestellt.

[8] Eine Ausnahme bilden natürlich gleichzeitig auftretende Wiederholungen, die addiert werden

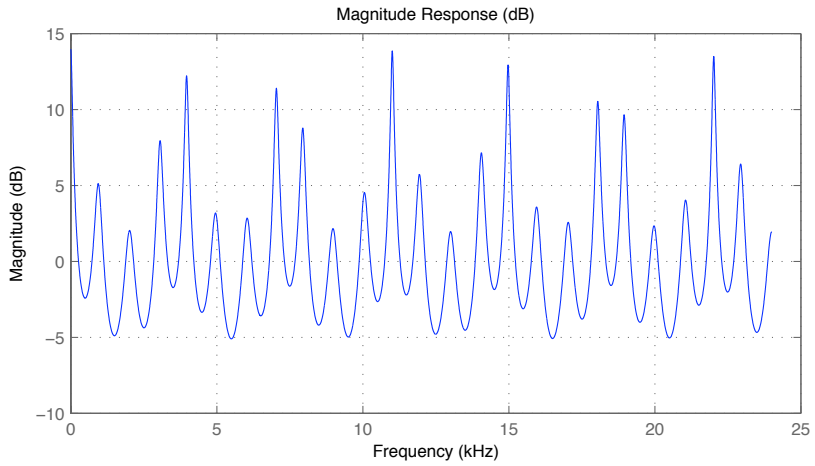


(a) Frequenzgang

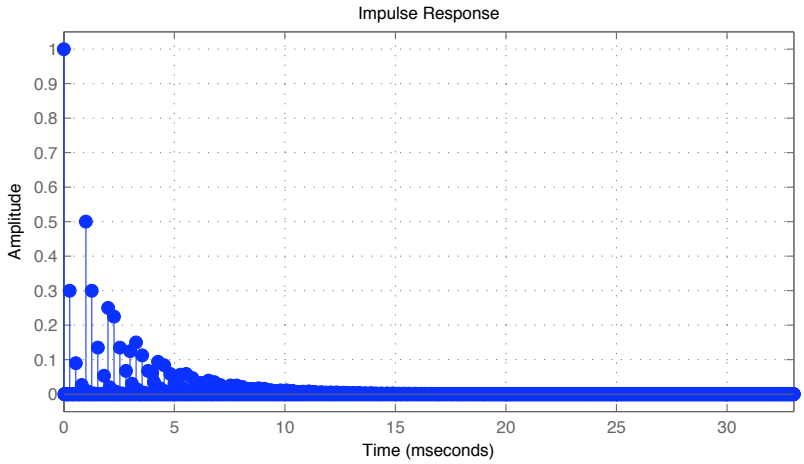


(b) Impulsantwort

Abbildung 2.3: Frequenzgang und Impulsantwort eines einfachen Kammfilters



(a) Frequenzgang



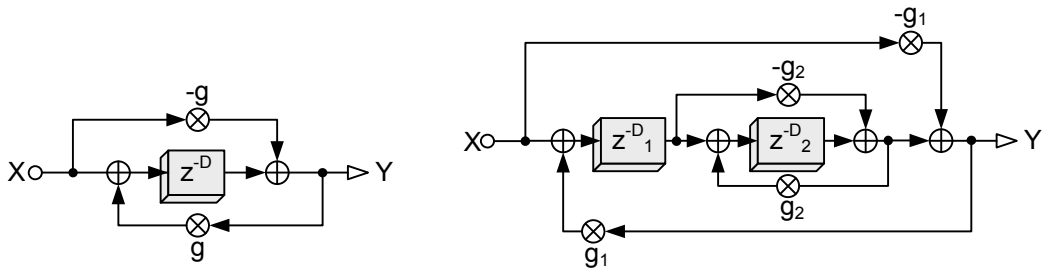
(b) Impulsantwort

Abbildung 2.4: Frequenzgang und Impulsantwort eines Kammfilters mit zwei parallelen Feedbacks

Allpass-Strukturen

Zwar lassen sich also mit einfachen Kammfiltern grundsätzlich Reflexionsvorgänge simulieren, die Wahl der passenden Parameter ist aber vor allem wegen der potentiell starken spektralen Verfärbungen äußerst problematisch. Doch mit sogenannten Allpassfiltern existieren Strukturen, die das gesamte Spektrum eines Signals gleichmäßig verstärken, jedoch eine frequenzabhängige Phasenverschiebung bewirken [15].

Der Aufbau eines einfachen digitalen Allpassfilters ist dem eines Kammfilters sehr ähn-



(a) einfache Form (b) veschachtelte Form

Abbildung 2.5: Allpässe

lich, jedoch verfügt es zusätzlich zum Feedback- auch über einen Feedforward-Weg, der mit dem negativen Wert der Feedback-Skalierung multipliziert wird (Abbildung 2.5(a)). Die Impulsantwort einer solchen Struktur ähnelt der eines Kammfilters (Abbildung 2.6(a)). Damit eignet sich ein solches Filter ebenfalls sehr gut zur Erzeugung von Echos, ohne aber die spektralen Probleme von Kammfiltern mit sich zu bringen. Wegen der Ausprägung ihrer Impulsantwort werden Allpässe teilweise auch als „Impulsexpander“ oder „Impulsdiffusoren“ bezeichnet [13, S. 19] und häufig auch in dieser Funktion eingesetzt.

Im Hinblick auf die klangliche Qualität eines Allpasses muss erwähnt werden, dass es nur unter sehr strengen Bedingungen wirklich verfärbungsfrei klingt. Voraussetzung hierfür ist, dass seine Impulsantwort weniger als ca. 10ms beträgt[9] oder am Eingang ein reiner Sinuston anliegt [13, S. 21]. Beides ist beim Einsatz in Hallalgorithmen in aller Regel nicht gegeben. Damit können auch Allpässe keineswegs bedenkenlos eingesetzt werden. Dennoch gelten sie als gute Alternative zum Kammfilter, da sich mit ihnen oft leichter gute Ergebniss erzielen lassen. Des Weiteren bieten sie im Gegensatz zu Kammfiltern die Möglichkeit, ihre Wirkung durch eine Reihenschaltung noch zu verstärken [3, S. 4].[10]

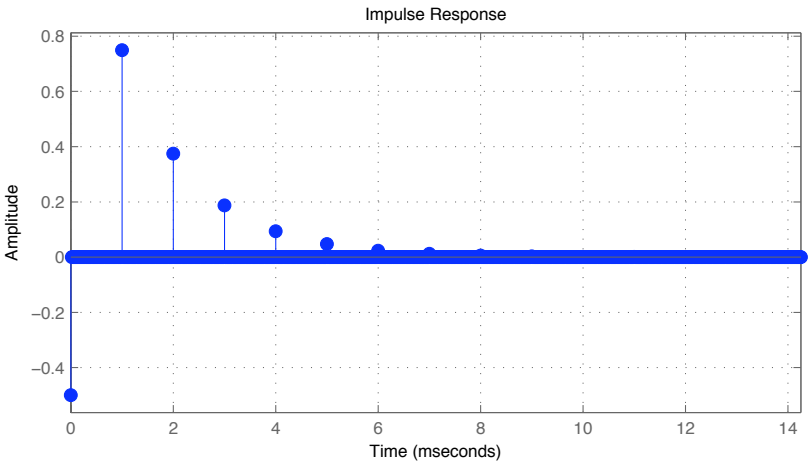
Eine wichtige Variation des Allpassfilters stellt der *Verschachtelte Allpass* dar (Abbildung 2.5(b)). Ein System, das aus ineinander liegenden Allpässen besteht, ist ebenfalls ein Allpass, hat aber je nach Wahl der Delay-Längen und Koeffizienten eine mitunter deutlich veränderte, dichtere Impulsantwort (Abbildung 2.6(b)). Als inneres Filter

[9] gemeint ist die effektive Länge der Impulsantwort bis zum letzten erzeugten Echo, denn es handelt sich ja um ein IIR-Filter

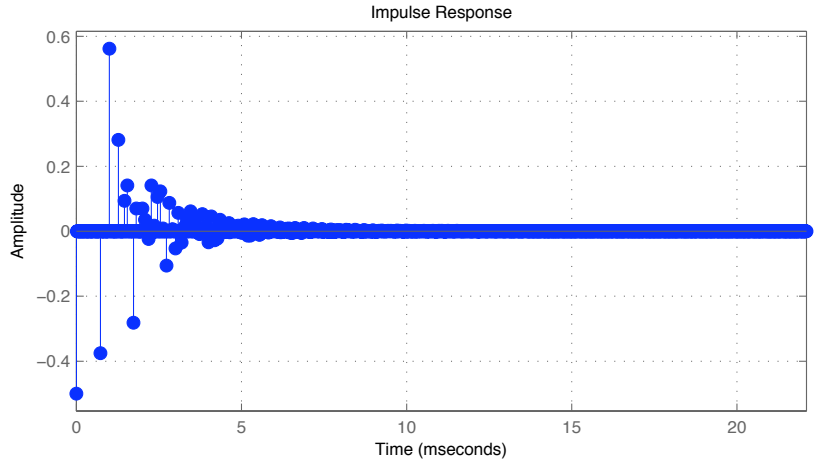
[10] Eine Reihenschaltung von Kammfiltern bedingt fast immer einen verschlechterten Klang, denn die spektralen Verfärbungen der einzelnen Kammfilter addieren sich. Damit erreichen nur Anteile, die in keinem der beteiligten Kammfilter ausgelöscht werden, das Ende der Schaltung [3, S. 4f.].

kommen hierfür auch Reihen von Allpässen oder wiederum verschachtelte Allpässe in Frage, wodurch sich die Anzahl der erzeugten Echos nochmals drastisch erhöhen lässt. Wie auch für einfache Kammfilter und einzelne Allpässe ist für jede dieser Strukturen Stabilität garantiert, falls alle $g < 1$ sind [2, S. 50].

Abbildung 2.6 zeigt die Impulsantworten eines einfachen Allpasses (D = 1 ms, g = 0,5) sowie eines verschachtelten Allpasses ($D_1 = \frac{35}{48}$ ms, $D_2 = \frac{13}{48}$ ms; $g_1 = g_2 = 0{,}5$). Da der Frequenzgang beider Filter 0 dB über das gesamte Spektrum beträgt, wurde auf seine Darstellung verzichtet.



(a) einfache Form



(b) verschachtelte Form

Abbildung 2.6: Impulsantworten von Allpässen

Allpässe lassen sich sehr effizient als Operatoren auf einer Delay Line implementieren

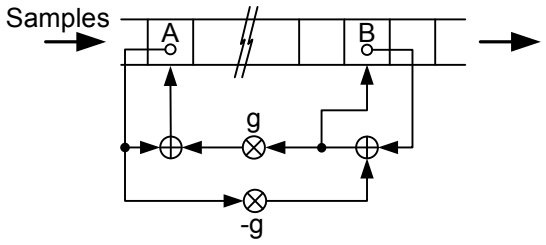


Abbildung 2.7: Allpass-Implementierung

(Abbildung 2.7). Dabei wird zunächst der Feedforward-, dann der Feedback-Weg nach folgenden Gleichungen berechnet:

$$\begin{aligned} B_{neu} &= B_{alt} + A_{alt} \cdot (-g) \\ A_{neu} &= A_{alt} + B_{neu} \cdot g \end{aligned}$$

[2, S. 50 f.] Damit ist eine sehr hohe Effizienz gewährleistet, wodurch sich Allpässe hervorragend als Grundstruktur für Hallalgorithmen eignen.

2.2.4 Algorithmische Verfahren zur Erzeugung späten Nachhalls

Schröder und Moorer Reverberator

Der vom heutigen Standpunkt erste bedeutende Hallalgorithmus wurde von Manfred Schröder im Jahr 1962 vorgestellt.[11] Schröders Verfahren (Abbildung 2.8) basiert auf einer Parallelschaltung von vier Kammfiltern. Deren typische Verfärbungen werden durch geschickte Wahl der Delay- und Skalierungswerte verhindert. Ergänzt wird die Schaltung durch zwei Allpässe am Ausgang [8, S. 29].

Moorers Modell stellt eine Weiterentwicklung von Schröders Entwurf dar. Seine Variante verwendet sechs parallele Kammfilter – jeweils mit einem Tiefpass im Feedback-Weg zur Simulation der Höhenabsorption – begnügt sich dafür jedoch mit der Verwendung eines einzigen Allpasses [8, S. 30].

Beide Modelle sind heute zwar veraltet, bilden aber dennoch den Grundstock für moderne Hallalgorithmen. Die heute allgemein anerkannten Grundideen wie die Simulation von Reflexionsvorgängen durch Rückkopplungen, die Glättung mit Allpässen oder

[11] Erscheinungsjahr des Papers [1], in dem er seinen Algorithmus vorstelle.

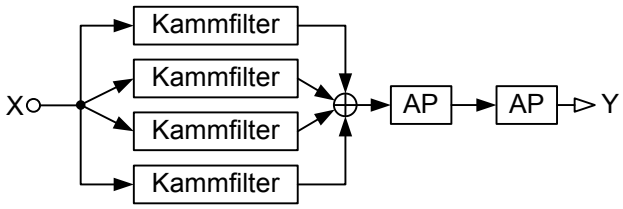


Abbildung 2.8: Schröder Reverberator

die Einbeziehung von Absorptionsvorgängen durch Einsatz zusätzlicher Filter sind schließlich schon in diesen frühen Varianten erkennbar. Mangels eines globalen Rückkopplungspfads gibt es jedoch keine für reale Räume typische Zunahme der Echos, was wohl das Hauptproblem bezüglich ihrer Realitätsnähe darstellt. Zusätzlich neigen beide Strukturen durch die relativ geringe Anzahl erzeugter Echos zu einem schlechten Ergebnis, wenn sie mit einem impulsartigen Signal angeregt werden [2, S. 15].

Gardner Reverberator

Der sogenannte Gardner Reverberator (Abbildung 2.9) basiert auf einer Reihenschaltung von (verschachtelten) Allpässen mit mehreren Abgriffen sowie einem globalen Rückkopplungspfad. Die mögliche Implementierung aller Allpässe auf einer einzigen Delay Line macht ihn dabei zu einer sehr ressourcenschonenenden Methode zur Hallerzeugung.

Die Kombination der Allpässe erzeugt eine sehr hohe Zahl von Echos, die sich darüber hinaus durch den globalen Rückführungspfad wie in einem realen Raum mit zunehmender Zeit noch erhöht. Die Dichte der Echos lässt sich dabei durch die Wahl der Allpasskoeffizienten beeinflussen [7, S. 2]. Damit hat eine derartige Struktur an sich optimale Voraussetzungen für den Einsatz in Hallalgorithmen.

Äußerst problematisch ist allerdings die Wahl der Parameter des Netzwerks. Denn wie Gardner selbst schreibt, kann eine einzige Struktur nicht alle benötigten Längen des Nachhalls abdecken, was verschiedene Ausprägungen des Algorithmus für unterschiedliche Hallzeiten nötig macht. Mangels mathematischer Methoden zur Bestimmung seiner Parameter basiert das Design der Struktur darüber hinaus letztlich auf Versuch und Irrtum [2, S. 55]. Ebenfalls problematisch ist die Beeinflussung der Nachhallzeit, denn sie hängt nicht linear von der Skalierung der Rückkopplung ab. Stattdessen muss die für eine bestimmte Hallzeit benötigte Skalierung aus gemessenen Werten

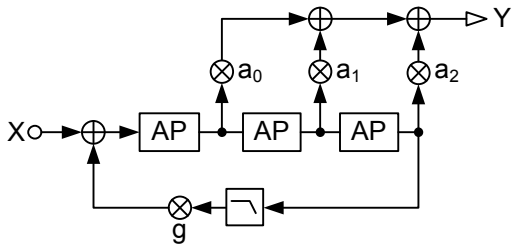


Abbildung 2.9: Gardner Reverberator

interpoliert werden [2, S. 55].

Feedback Delay Networks

Feedback Delay Networks (FDNs) folgen einem grundlegend anderen Ansatz als die bisher vorgestellten Strukturen. Während diese auf die Erzeugung vieler Echos im Direktweg setzen und über maximal einen Rückkopplungspfad verfügen, bauen FDNs auf einigen parallelen Delays sowie einer Rückkopplung aller Kanäle untereinander auf. Pro Kanal und Durchgang entsteht damit zwar nur ein einziges Echo, die Vielzahl der Rückkopplungen lässt ihre Zahl aber dennoch sehr hoch werden. Abbildung 2.10 zeigt exemplarisch ein einfaches FDN. Die genaue Anordnung von Delays, Filtern und Matrix ist jedoch von der konkreten Implementierung abhängig.

Die Rückkopplung der Kanäle untereinander wird durch eine Matrixmultiplikation berechnet. In Frage kommen hier in erster Linie verlustfreie Matrizen, denn bei ihrem Einsatz ist die Hallzeit eines FDN unendlich, wenn nirgends zusätzliche Dämpfungen eingebaut werden. Dieser Ansatz empfiehlt sich, denn durch Kopplung der Delays mit Filtern zur Dämpfung (s.u.) lässt sich so die gewünschte Nachhallzeit präzise variieren [13, S. 31].

In der Praxis wird häufig eine sog. Hadamard Matrix eingesetzt, die eine verlustfreie Matrix darstellt. Eine solche Matrix der Größe $n = 2$ lautet

$$H_2 = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

Matrizen höherer Ordnung ergeben sich durch rekursive Einbettung. Damit ist

$$H_4 = \frac{1}{\sqrt{2}} \cdot \begin{bmatrix} H_2 & H_2 \\ -H_2 & H_2 \end{bmatrix} = \frac{1}{2} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \text{ [13, S. 32]}$$

Die Beeinflussung der Nachhallzeit in einem solchen verlustfreien FDN geschieht durch eine Dämpfung in jedem Kanal, die in einem logarithmischen Zusammenhang mit der Länge des jeweiligen Delays steht. Ist die Dämpfung frequenzabhängig, können so auch verschiedene Hallzeiten für unterschiedliche Spektralbereiche realisiert werden. Die für eine gewünschte Hallzeit bei einer bestimmten Frequenz benötigte Dämpfung des Loop Filters lässt sich in Abhängigkeit von Länge des jeweiligen Loop Delays D nach folgender Formel berechnen:

$$20 \cdot \lg \left| H \cdot \left(e^{j\omega\tau} \right) \right| = -60 \cdot \frac{D}{f_s \cdot t_{60}(\omega)} \tag{2.1}$$

[13, S. 42]. Im einfacheren Fall einer für alle Frequenzen gleichen Hallzeit ergibt sich damit folgender Zusammenhang:

$$20 \cdot \lg a = -60 \cdot \frac{D}{f_s \cdot t_{60}} \tag{2.2}$$

$$a = 10^{-\frac{3D}{f_s \cdot t_{60}}} \tag{2.3}$$

Zur Wahl der Delaylängen gibt es keine präzisen mathematischen Vorschriften. Als Faustregeln können jedoch gelten:

- Die einzelnen Delays sollten teilerfremd sein
- Bezüglich der Summe der Delaylängen sollte gelten:

  $$D_{ges} \geq 0,15 \cdot t_{60} \cdot f_s$$

  Beispielsweise sollte damit für eine Nachhallzeit von 1s und eine Samplingfrequenz von 50kHz die Summe der Delays mindestens 7500 Samples betragen [13, S. 38f.].

Auch wenn damit FDNs mathematisch wesentlich präziser beschrieben werden können als alle anderen vorgestellten Strukturen, ist besonders die Wahl der Delays nicht unproblematisch. Es gibt also durchaus „schlecht klingende“ FDNs. In der Praxis zeigt sich außerdem, dass ein gutes FDN über mindestens 8 Kanäle verfügen sollte.

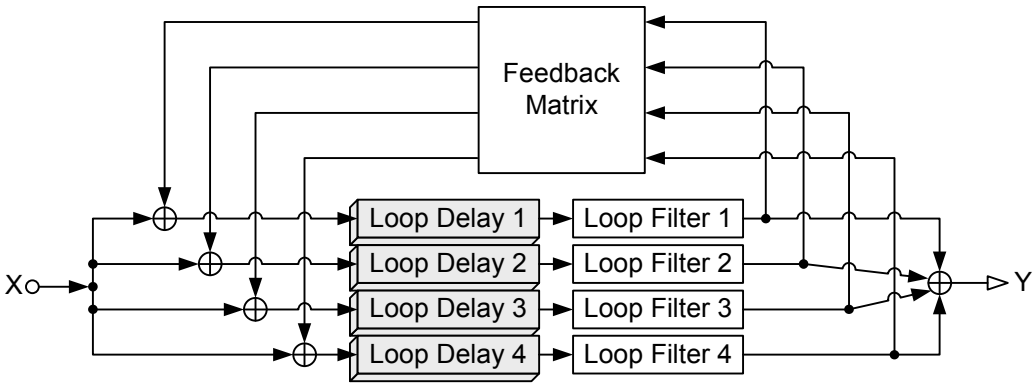


Abbildung 2.10: einfaches Feedback Delay Network

2.2.5 Vorteile eines Hybrid-Algorithmus

Über die Frage nach dem richtigen Grundalgorithmus für eine bestimmte Anwendung entscheiden heute zwei Kriterien: klangliche Anforderungen einerseits, Entwicklungs- und Rechenaufwand andererseits. Je nach Einsatzzweck und Prioritäten wird man sich für eine der Varianten entscheiden. Jedoch gibt es im Hinblick auf beide Punkte durchaus Argumente für eine hybride Variante.

Aus Anwendersicht

Betrachtet man die Anforderungen des Nutzers an einen Hallalgorithmus, so gibt es zwei typische Anwendungsfälle, die sich jeweils in einem bevorzugten Algorithmus niederschlagen.

Möchte man einen bestehenden Raum möglichst detailgetreu und exakt nachbilden, kennt man darüber hinaus seine Impulsantwort und ist mit ihr (im Wesentlichen) zufrieden, wird ein Faltungshall unschlagbare Ergebnisse liefern.

Möchte man maximale Kontrolle über einen virutellen Raum, so wird ein algorithmischer Hall überlegen sein, denn grundsätzlich lassen sich all seine Parameter beliebig verändern. Darüber hinaus ist auch nicht in allen Fällen maximale Realitätsnähe gewünscht. Auch hier ist ein algorithmischer Hall die bessere Alternative.

In der Mitte zwischen diesen beiden Extremen liegen schließlich aus Sicht des Anwenders die Vorteile eines hybriden Algorithmus: denn er kann gleichzeitig einen glaubwürdigen Raum liefern und dennoch in bestimmten Bereichen hohe Flexibilität bieten.

Möchte man zum Beispiel eine Hallzeit erreichen, die über der des Originalraums liegt, kommt ein Faltungshall schnell an seine Grenzen. Damit schlägt ein Hybridalgorithmus die Brücke zwischen beiden Verfahren.

Entwicklungs- und Berechnungsaufwand

Bedenkt man jedoch mit Sicht auf Aufwand und Ertrag bei Entwicklung und Einsatz die Vor- und Nachteile beider Formen von Hallalgorithmen, wird deutlich, warum ein Hybridalgorithmus grundsätzlich Sinn macht:

Denn die Berechnung eines Faltungshalls ist extrem aufwändig, seine Konzeption aber verhältnismäßig einfach, wenn auch der mathematische Aufwand hinter den nötigen Optimierungen bei der Berechnung nicht zu unterschätzen ist. IIR-Algorithmen andererseits haben eher moderate Anforderungen an die Berechnung, benötigen aber immensen Aufwand in der Entwicklung, um klanglich überzeugen zu können.

Gleichzeitig sind die späten Hallanteile bezüglich der Notwendigkeit einer detailgetreuen Nachbildung unkritischer als die frühen, denn sie sind ohnehin diffus und bringen wenig neue Rauminformation. Darüber hinaus manifestiert sich der Klangcharakter eines Raums schon im frühen Bereich des Nachhalls – beispielsweise im Hinblick auf das Abklingverhalten einzelner Frequenzbereiche.

Es müsste also möglich sein, die Qualität eines Faltungshalls auch dann zu erhalten, wenn man einen „intelligenteren“ Algorithmus einsetzt als die „Brute Force Methode“ einer Faltung über die gesamte Hallzeit. Dabei müsste der Entwicklungsaufwand dennoch erheblich geringer sein als für einen gleichwertigen IIR-Algorithmus, denn die anspruchsvollsten Teile bleiben bei der Konzeption des IIR-Teilalgorithmus außen vor.

Und genau das zu erreichen, ist die Motivation der vorliegenden Entwicklung.

3 Der bestehende Algorithmus

Bevor die eigentliche Entwicklung des Algorithmus beschrieben wird, erfolgt zunächst ein kurzer Blick auf den bestehenden VSP-Algorithmus. Besonders wird dabei natürlich auf den späten Nachhall eingegangen, denn ihn zu verbessern ist ja der Sinn dieser Arbeit.

3.1 Panning

Selbstverständlich bietet VSP wie jedes Surround-Panning einige Parameter zur Verteilung eines Signals auf die einzelnen Kanäle. Diese umfassen *LFE Level*, *Left/Right & Front/Back Pan*, *Center Percentage* sowie *Divergence* („Übersprechen“ der Kanäle untereinander) [9, Parameters 3-30f].

Neben diesen Parametern, die alle Leistungspanning und damit einer XY-Mikrofonierung entsprechen, wird auch die Möglichkeit zur Simulation weiterer Mikrofonierungstechniken durch Integration von Laufzeitdifferenzen gegeben. Die vorkonfigurierten Varianten sind eine virtuelle AB und ORTF-Anordnung. Die Option *User* erlaubt darüber hinaus die Simulation weiterer Varianten durch Festlegung von Richtcharakteristiken und Abständen des virtuellen Surround-Hauptmikrofons [9, Parameters 3-49ff].

Zusätzlich besteht die Möglichkeit, durch Integration einer Pseudo-HRTF[1] eine Positionierung auch in extremen Seitenpositionen zu realisieren. Eine abgeschwächte Variante dessen ist der Pan Mode *Sphere*, der ein Kugelflächenmikrofon simuliert [9, Parameters 3-44f].

[1] HRTF: Head Related Transfer Function. Die (eigentlich individuelle, aber in diesem Fall generell approximierte) Übertragungsfunktion des menschlichen Kopfes bezüglich eines Signals aus einer bestimmten Richtung. Sie beinhaltet beispielsweise Abschattungseffekte des Kopfes und der Ohrmuscheln bei Schalleinfall von den Seiten oder von hinten sowie entsprechende Phasenverschiebungen [17].

3.2 Frühe Reflexionen

VSP verfügt darüber hinaus über die Option, dem Direktsignal zusätzlich frühe Reflexionen hinzuzufügen. Deren genaue Ausprägung lässt sich über verschiedene Parameter beeinflussen. Sie umfassen unter anderem

- Größe des virtuellen Raums
- Absorptionsverhalten seiner Begrenzungsflächen
- Abstand der virtuellen Schallquelle

Die Wahl der Parameter beeinflusst dabei die genaue Ausprägung der erzeugten Reflexionen nach psychoakustischen Grundsätzen und führt so zu einem veränderten räumlichen Eindruck des Schallereignisses [9, Parameters 3-45].

Wie auch das eigentliche Panning werden die frühen Reflexionen für jeden VSP-Kanal einzeln berechnet.

3.3 Hall

3.3.1 Verfahren

Schließlich umfasst VSP auch ein Verfahren zur Erzeugung einer diffusen Hallfahne. Realisiert ist dieses über vier Hall-Busse, die in den VSP-Kanälen ähnlich Aux-Send-Wegen zur Verfügung stehen. Damit kann ein einziger Hall-Kanal von mehreren VSP-Kanälen gemeinsam genutzt werden.

Der Hallalgorithmus besteht im Wesentlichen aus einem Feedback Delay Network, wie es in Abschnitt 2.2.4 beschrieben wurde. Aus ihm werden die vier Kanäle L, R, LS, RS[2] durch Abgriff verschiedener Ausgangssigale des FDN gewonnen. Für den Center- sowie den LFE-Kanal wird kein diffuser Nachhall berechnet. Zur Beeinflussung stehen dem Anwender neben dem Pegel des Halls lediglich die Parameter *High Frequency Absorption*, *Low Frequency Absorption*[3], *Front/Rear Balance* und *Reverb Time* zur Verfügung [9, Parameters 3-46].

[2] L = Left, R = Right, LS = Left Surround, RS = Right Surround

[3] Bei beiden Parametern entspricht ein negativer Wert einer Dämpfung des jeweiligen Bereichs, was auch in dieser Arbeit übernommen wurde

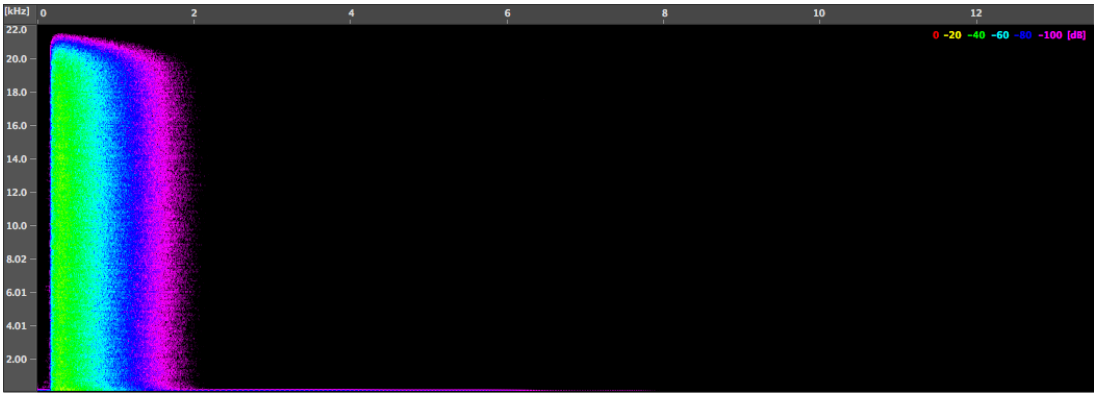


Abbildung 3.1: VSP-Hall – Nachhallzeit 0,5 s; High Absorption 0 dB; Low Absorption 0 dB

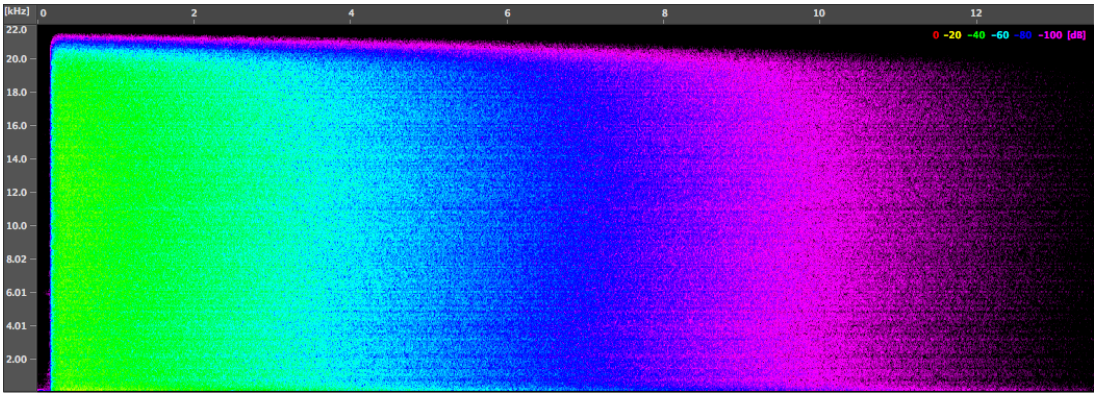
Die Abbildungen 3.1 und 3.2 zeigen Spektrogramme der Impulsantworten des linken Kanals bei unterschiedlicher Wahl der einzelnen Parameter zur Veranschaulichung von deren Auswirkung auf den erzeugten Nachhall.

3.3.2 Subjektive Mängel

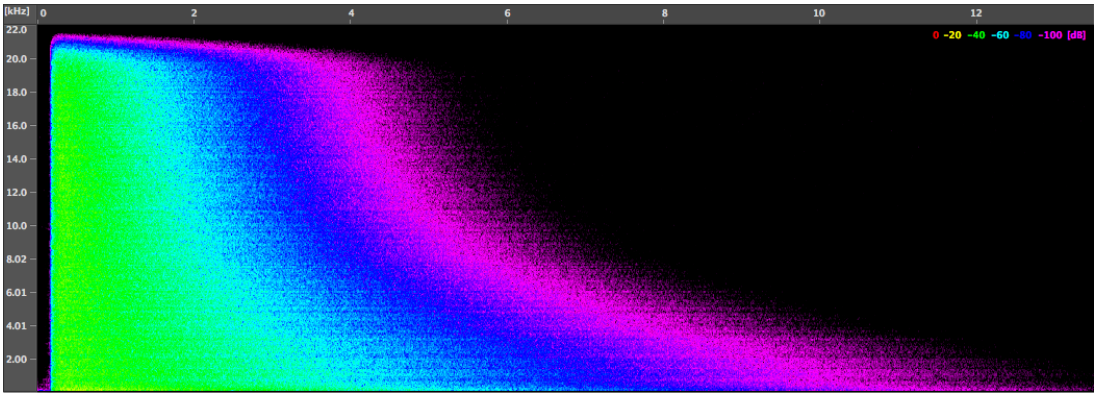
Die Hörbeispiele 1 bis 4 geben einen Eindruck vom Klang des Halls bei verschiedenen Nachhallzeiten und Eingangssignalen. Zwar hat der vorhandene Hallalgorithmus ohne Frage auch viele positive Eigenschaften, diese sollen jedoch in der folgenden kurzen Betrachtung außen vor bleiben.

Ein erstes augenfälliges Problem der vorliegenden Implementierung ist die sehr hohe Verzögerung bis zum Beginn des diffusen Nachhalls. Darauf folgend erreicht das Signal jedoch sehr schnell sein Maximum, so dass besonders bei perkussiven Signalen ein recht unnatürlicher Gesamteindruck entsteht (Hörbeispiel 7). Jedoch scheint es sich hierbei um einen Implementierungsfehler und nicht um ein konzeptionelles Problem zu handeln, so dass dieses Problem vermutlich mit verhältnismäßig geringem Aufwand zu beheben wäre.

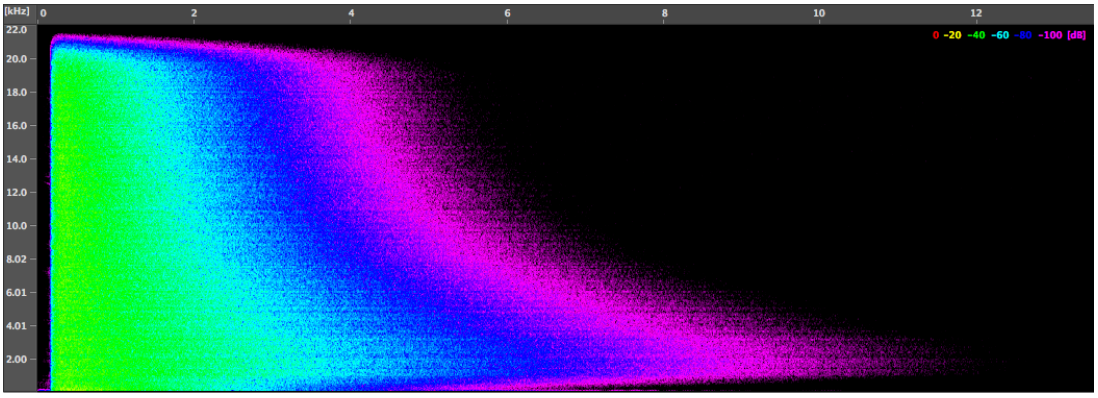
Wesentlich problematischer ist die starke Neigung des Algorithmus zur Überbetonung hoher Frequenzen. Ein solches Verhalten ist beispielsweise bei Snare Drum, Gitarre oder Trompete zu bemerken (Hörbeispiel 4). Der entstehende Klang ist höchst unnatürlich. Abhilfe schaffen nur extreme Einstellungen der Höhenabsorption, was jedoch bei längeren Hallzeiten mit einem dumpfen Gesamtklang einher geht (Hörbeispiele 5 und 6). Für derartige Anwendungen ist der Algorithmus damit ungeeignet.



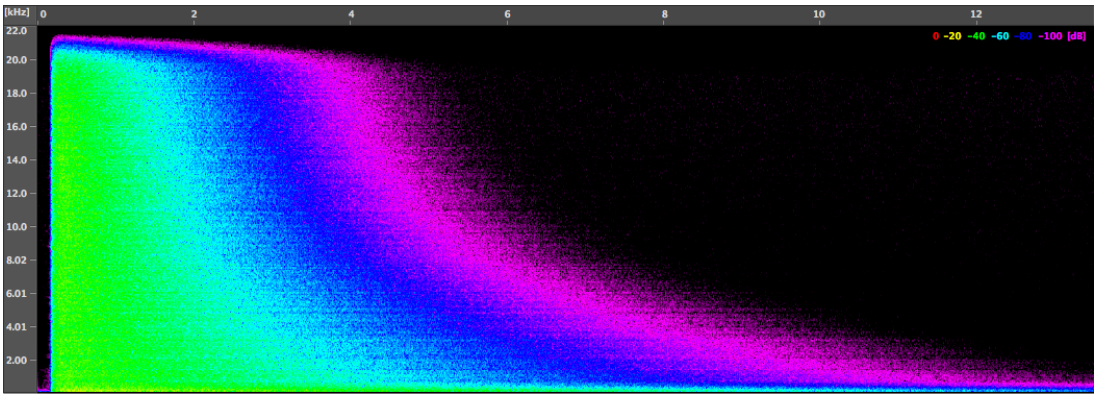
(a) High Absorption 0 dB; Low Absorption 0 dB



(b) Hight Absorption -6 dB; Low Absorption 0 dB



(c) High Absorption -6 dB; Low Absorption -6 dB



(d) High Absorption -6 dB; Low Absorption +6 dB

Abbildung 3.2: VSP-Hall – Nachhallzeit 7,9 s

4 Verwendete Raumimpulsantwort

Dem Faltungsteil des Hybridalgorithmus lag die HDIR[1] eines Konzertsaals im 5.0-Format zu Grunde. Die einzelnen Impulsantworten wurden mit einem Lautsprecher in der Center-Position erstellt, die virtuelle Schallquelle befindet sich also mittig auf der Bühne. Abbildung 4.1 zeigt exemplarisch ein Spektrogramm der Impulsantwort des Center-Kanals.

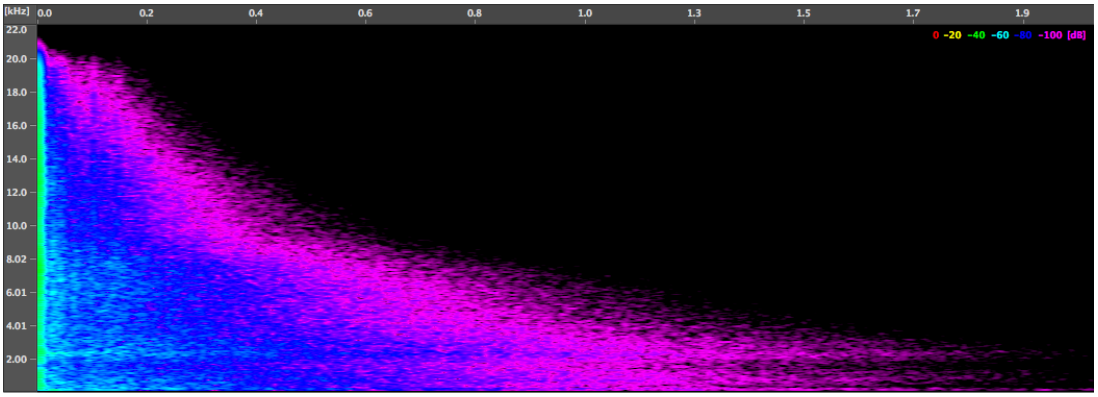


Abbildung 4.1: Spektrogramm der verwendeten Raumimpulsantwort (Center-Kanal)

[1]High Definition Impulse Response. Eine mit der Software *HDIR Creator* der Firma Pinguin erstellte, hochauflösende Raumimpulsantwort [10]. Bei der Erstellung dieser Impulsantworten erlaubt der HDIR Creator Manipulationen der Impulsantwort von spektralen Korrekturen bis zur Abbildung der Richtcharakterisitk von Solisten oder Instrumenten [12, S. 18 f].

5 Konzeption des Algorithmus

5.1 Problematik und Vorgehen

Die große Schwierigkeit bei der Entwicklung war, dass ein Hallalgorithmus in der Theorie zwar relativ simpel, seine vielen Varianten bezüglich verwendeter Teilalgorithmen und deren jeweiliger Parameter jedoch völlig unüberschaubar sind. Letztlich sind es außerdem die exakten Werte der einzelnen Parameter, die gute Algorithmen von durchschnittlichen abheben – mehr als generelle Richtwerte wird man also bezüglich keines Algorithmus in Erfahrung bringen können, denn kein Hersteller gibt die Details seiner Implementierung preis.

Im Fall eines fehlgeschlagenen Versuchs stellte sich damit immer die Frage, ob der Algorithmus grundsätzlich einem falschen Ansatz folgte oder lediglich ungünstige Werte der Parameter gewählt worden waren. Diese Frage eindeutig zu klären, war bis auf wenige offensichtliche Fälle schlicht ausgeschlossen. Ein großer Teil der Arbeit bestand folglich darin, die auftretenden Fehlversuche einschätzen zu lernen. Das machte es nötig, die einzelnen Parameter des betrachteten Algorithmus zu variieren – und das sowohl unabhängig voneinander als auch in Kombination.

Grundsätzliche Variationen des Algorithmus umfassten:

- Länge und Art der Impulsantwort:
  natürlich sollte die Faltung möglichst kurz gehalten werden, um die Effizienz des Gesamtalgorithmus hoch zu halten. Jedoch war unklar, wie lang die Faltung tatsächlich sein muss, um die relevante Rauminformation zu transportieren.
- Mögliche Segmentierung der Impulsantwort:
  eine weitere Variante war die Möglichkeit, nur einen Teil des gefalteten Signals in das IIR-Netzwerk zu schicken.
- Wahl der IIR-Struktur:
  Dies war natürlich die offensichtlichste Variable bei der Entwicklung des Algo-

rithmus. Besonders hier war eine Abwägung zwischen einem untauglichen Verfahren und schlecht gewählten Werten ausgesprochen schwierig.

Dieses Kapitel dokumentiert die Grundkonzeption des Hybridalgorithmus. Diese Arbeit lässt sich wiederum in zwei Teile untergliedern: zunächst wurden elementare Hallalgorithmen in einer sehr einfachen Verbindung mit einer kurzen Faltung getestet, um ihre grundsätzliche Eignung für ein hybrides Verfahren zu bestimmen (Abschnitt 5.2). Die dabei erlangten Erkenntnisse waren maßgeblich für den zweiten Schritt: die Konzeption der schließlich verwendeten Hybridstruktur, die in Abschnitt 5.3 beschrieben wird.

Mit der Konzeption ging die Implementierung der jeweiligen Filter und Algorithmen in MATLAB und C einher. Details zum Zusammenspiel der verwendeten Software, zur Versuchsdurchführung sowie zum verwendeten PC und Studio finden sich in Anhang A. Das Konzept zur Implementierung der Filter wird in Anhang B.1 beschrieben.

5.2 Erste Versuche

Um den Berechnungsaufwand gering zu halten und die Variablen bei der Entwicklung zu reduzieren, wurde zunächst auf eine Surround-Implementierung des Algorithmus verzichtet. Stattdessen wurden die ersten Untersuchungen nur für jeweils einen Kanal durchgeführt.

5.2.1 IIR-Hallalgorithmen im Verbund mit einer kurzen Faltung

Da wie oben beschrieben die Anforderungen an den IIR-Algorithmus völlig unklar waren, wurde zunächst eine ganze Reihe von Grundstrukturen implementiert und im Verbund mit einer relativ kurzen Faltung getestet. Da auch die benötigte Länge der Faltung unklar war, wurden verschiedene Ausschnitte der Impulsantwort mit Größen in einem Bereich zwischen 200ms und einer Sekunde verwendet, die teilweise zusätzlich mit einem Hamming-Fenster versehen wurden. Als Testsignale dienten während dieser Phase Schläge von Bass Drum, Snare Drum sowie bei Bedarf ein Dirac-Stoß zur Visualisiserung der Impulsantwort.

Während dieser ersten Tests wurde der Aufbau sehr einfach gehalten: das Eingangssignal wurde zunächst mit dem oben beschriebenen Segment der realen Impulsantwort gefaltet. Das gefaltete Signal wurde daraufhin entsprechend einer Reihenschaltung in

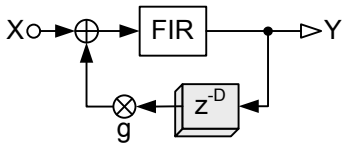


Abbildung 5.1: Rückkopplung vor die Faltung

die jeweilige IIR-Struktur geschickt.[1] Während der jeweiligen Versuche wurde nun versucht, die Ergebnisse durch Variation einzelner Parameter zu verbessern.

Rückkopplung vor die Faltung

Der erste Ansatz einer Kombination aus FIR- und IIR-Struktur bestand in einer Rückkopplung vor die Faltung, um deren effektive Länge zu erhöhen. Diese Variante wurde in diversen Konfigurationen getestet.

Eine recht einfache Konfiguration benutzte dabei lediglich ein skaliertes Delay im Rückführungspfad (Abbildung 5.1). Das damit erreichte Ergebnis war durch starke Resonanzen, die sowohl von der Impulsantwort als auch von der gewählten Delaylänge abhängig waren, jedoch sehr schlecht. Als Alternative wurde das Delay auf verschiedene Arten zeitlich variiert[2], um die Systemresonanzen zu verändern und damit unauffälliger werden zu lassen. Zwar konnten die störenden Resonanzen mit diesem Verfahren tatsächlich verringert werden, die ebenfalls auftretenden, allenfalls schwer zu eliminierenden Artefakte[3] führten jedoch dazu, dass auch dieser Ansatz schnell verworfen wurde.

Denn mit der Verwendung einer Allpass-Sektion im Rückführungsweg stand eine vielversprechende Alternative zur Verfügung. Damit konnte ein Ergebnis erzielt werden, das die störenden Resonanzen nicht mehr aufwies und wesentlich stabiler wurde. Als dies jedoch erreicht war, wurde der systematische Fehler im Verfahren schnell offensichtlich:

Denn jede Rückkopplung vor die Faltung führt zwangsweise dazu, dass der individu-

[1] Eine Ausnahme bildete die Rückkopplung vor die Faltung die als ein einziges IIR-Filter implementiert wurde

[2] zum Einsatz kam sowohl ein zufälliger Abgriff innerhalb eines bestimmten zeitlichen Bereichs als auch Modulationen der Delaylänge mit einem Sinus, Dreieck und Sägezahn.

[3] der zufällige Abgriff führte zu einem hörbaren Rauschen, die Modulationen in allen wirkungsvollen Varianten je nach Modulationsfrequenz zu Pitch Shifting oder der Entstehung von harmonischen Obertönen der Modulationsfrequenz.

elle Klangcharakter der Raumimpulsantwort dem Signal mehrmals „aufgeprägt“ und damit mit jedem Durchlauf stärker wird. Ist diese Klangfarbe charakteristisch – und das ist im Fall des zu entwickelnden Algorithmus ja ausdrücklich gewünscht – ist das Resultat schlicht nicht zu gebrauchen, denn der Effekt ist sehr deutlich hörbar und wirkt höchst unnatürlich.

Damit schied die Variante einer Rückkopplung vor die Faltung aus.

Schröder und Moorer Reverberator

Eine frühe Arbeit zur Entwicklung hybrider Hallalgorithmen [8] setzt zur Erzeugung des späten Hallteils einen einfachen Moorer Reverberator ein. Die dabei erreichten Ergebnisse sind durchaus ermutigend. Jedoch findet in der Arbeit keine Untersuchung von alternativen IIR-Methoden statt. Die Frage, ob es dem Moorer Reverberator überlegene Alternativen gibt, bleibt also offen. Dennoch scheint ein Schröder oder Moorer Reverberator zumindest ein guter Ausgangspunkt für die Erzeugung des späten Halls zu sein.

Zunächst wurde die Faltung in Verbindung mit einem Moorer Reverberator getestet. Es zeigte sich jedoch, dass eine derartige Struktur auch im Verbund mit einer Faltung zu den gleichen Problemen neigt, die ihr auch im alleinigen Einsatz anhaften: denn die erzielten Ergebnisse waren zwar keineswegs schlecht, kamen aber nicht über ein gewisses Grundniveau hinaus – besonders kurz vor Abklingen des Nachhalls traten die gleichen Probleme auf, die den Moorer Reverberator heute auch im alleinigen Einsatz zu einer veralteten Methode machen.

Als Variante wurden mehr Allpässe am Ausgang eingesetzt, was aber keine nennenswerte Verbesserung bewirkte. Weitere Variationen im Algorithmus – Möglichkeiten wären z.B. ein globales Feedback oder der Einsatz von deutlich mehr Kammfiltern gewesen – wurden allerdings außen vor gelassen, denn es konnte davon ausgegangen werden, dass sich beim Einsatz neuerer, dem Moorer Reverberator überlegener IIR-Algorithmen auch im Verbund mit einer Faltung bessere Ergebnisse erzielen lassen.

Gardner Reverberator und Feedback Delay Network

Da der Gardner Reverberator nur recht moderate Anforderungen an den Rechner stellt und nach allgemeiner Meinung durch seine recht neutrale Allpass-Struktur dennoch gute Ergebnisse erzielen kann, wurde er als nächster implementiert.

Die Ergebnisse fielen je nach Wahl der Parameter der einzelnen Skalierungen, Allpässe usw. höchst unterschiedlich aus. Dennoch konnte keine überzeugende Konfiguration gefunden werden – die in Gardners Thesis vorgeschlagenen Werte [2, S. 56] konnten ebenfalls nicht überzeugen.

Wie sich heraus stellte, ist die Wahl der zahlreichen Parameter bei diesem Algorithmus tatsächlich mehr eine Kunst als eine Wissenschaft, denn die komplexen Zusammenhänge lassen sich kaum nachvollziehen. Da darüber hinaus nicht mit Gewissheit gesagt werden konnte, ob sich ein solcher Algorithmus überhaupt für diesen sehr speziellen Einsatzfall eignet, musste sein alleiniger Einsatz schließlich verworfen werden.

Ähnliches galt auch beim Feedback Delay Network. Denn auch hier gibt es eine sehr hohe Zahl von verschiedenen Ausprägungen. Mögliche Varianten umfassen die verwendete Feedback-Matrix und ihre Position innerhalb des FDN sowie die genaue Ausführung der Loop Delays und Filter.

Auch hier war die Aufgabe, im Rahmen dieser Arbeit eine einzige Struktur zu finden, die alle Anforderungen des Algorithmus erfüllte, nicht zu lösen. Damit schied auch diese Kombination aus.

5.2.2 Funktionale Unterteilung des IIR-Teilalgorithmus

Da keine Standardstruktur dazu geeignet war, die Berechnung des Halls aus dem Ausgangssignal der Faltung vollständig zu übernehmen, wurde beschlossen, eine eigene, an die Bedürfnisse des Gesamtalgorithmus angepasste Struktur zu verwenden. Sie sollte aus zwei Teilen mit voneinander getrennten Aufgaben bestehen:

1. Vorverarbeitung des Eingangssignals
2. Erzeugung des späten Nachhalls

Während dann die Erzeugung des späten Halls im Idealfall auch in der Hybridvariante mittels eines Standardverfahrens erfolgen könnte, übernähme die Vorverarbeitung alle notwendigen Veränderungen an dessen Eingangssignal. Ihre Aufgaben wären damit:

- Eliminierung problematischer Anteile des gefalteten Signals. Im Wesentlichen beträfe dies
  - frühe Reflexionen
  - spektrale Unterschiede zwischen Beginn und Ende der Raumimpulsantwort
  - kurzzeitige Effekte in der Impulsantwort wie Schwebungen

- Ermöglichung von unauffälligen Überlappungen des FIR-Ausgangssignals im IIR-Teil
- Schaffung eines fließenden Übergangs zwischen frühem Hall aus der Faltung und spätem Hall aus der IIR-Struktur

5.2.3 Diskrete Mono-Vorverarbeitungsstufe

Der erste Ansatz war, die Vorverbeitung des Signals unabhängig von den anderen Bestandteilen des Algorithmus und für jeden Kanal einzeln durchzuführen. Dabei kam eine Reihe unterschiedlicher Strukturen zum Einsatz. Diese waren

- eine Delay Line mit mehreren Abgriffen
- Kaskaden diverser Allpass-Strukturen
- ein FDN auf Basis einer 4×4 Hadamard-Matrix

sowie Kombinationen dieser Bestandteile. Sämtliche Versuche, die Vorverarbeitung auf diese Art unabhängig vom Rest der Berechnung vorznehmen, hatten jedoch unbrauchbare Ergebnisse zur Folge. Denn während für das FDN keine Konfiguration gefunden werden konnte, die das Signal in der gewünschten Art und Weise beeinflusst hätte, brachten effektive Konfigurationen von Delay Line und Allpässen in der Regel recht hohen Rechenaufwand mit sich, benötigen viel Speicher und hatten darüber hinaus Nebenwirkungen, die den Einsatz einer jeden solchen Struktur ausschlossen. Im Fall der Delay Line mit mehreren Abgriffen waren dies Kammfiltereffekte, im Fall der Allpass-Kombinationen starkes Nachschwingen und ebenfalls spektrale Verfärbungen.

Der Gedanke einer diskreten Vorverarbeitungsstufe auf Basis der kompletten Faltung musste damit ebenfalls verworfen werden. Stattdessen wurde nun auch die Faltung selbst in die Überlegungen einbezogen. Diese Veränderung des Algorithmus ging mit einer Ausweitung der Betrachtung auf *alle* Kanäle einher und wird deswegen zusammen mit dem neu entstandenen Surround-Algorithmus im nächsten Abschnitt beschrieben.

5.3 Der Surround-Algorithmus

Nachdem alle bisherigen Implementierungen wenig Erfolg versprechend waren, wurde nach einer grundsätzlich neuen Herangehensweise gesucht. Der ab diesem Zeitpunkt verfolgte neue Ansatz basierte auf folgender Überlegung:

Die Klang eines Raumes ist durch Wechselwirkungen in seinem Inneren bestimmt. Diese müssten sich auch in den jeweiligen Surround-Raumimpulsantworten niederschlagen. Ein Surround-Hybridalgorithmus sollte sich das zu Nutze machen können, indem er den späten Hall eines Kanals nicht nur aus dessen früher Impulsantwort errechnet, sondern sie in Wechselwirkung zum Nachhall der anderen Kanäle setzt. Einerseits müsste dies dem realen Verhalten des Raumes zumindest nahe kommen, andererseits könnten statistische Effekte problematische Bestandteile wie Schwebungen abschwächen. Damit wäre einerseits das Ergebnis besser, andererseits die Implementierung einfacher.

Diesem Gedanken folgend, wurde die Betrachtung ab diesem Zeitpunkt von einem auf vier Kanäle erweitert. Wie auch beim bisherigen VSP-Algorithmus blieb der Center-Kanal außen vor. Diese Einschränkung schien sinnvoll, da

- der Center-Kanal für eine präzise Abbildung in der Mitte des Panoramas eingesetzt wird. Der zu erzeugende diffuse Hall benötigt aber keine präzise räumliche Abbildung
- auch der vorhandene Algorithmus nur vier Kanäle umfasst
- die eventuelle Verwendung eines globalen Feedback Delay Networks nach dem Beispiel des bisher verwendeten Algorithmus erheblich erleichtert wurde[4]

Aus den Überlegungen zur Einbeziehung der Faltung und Ausdehnung der Betrachtung auf vier Kanäle ging der schließlich verwendete Algorithmus recht schnell hervor. Seine Struktur wird in diesem Abschnitt beschrieben.

Anmerkung:

Die folgenden Abschnitte stellen den Endstand der MATLAB-Entwicklung dar. Die vorgestellte Struktur wurde in weiten Teilen auch im endgültigen Algorithmus beibehalten, jedoch im Zuge der in Kapitel 6 beschriebenen Feinabstimmung noch variiert.

Das genaue Vorgehen bei der Wahl und Kombination der Teilstrukturen basierte im Wesentlichen auf Versuch und Irrtum und folgte keinem festen Schema. Auf eine genaue Auflistung der einzelnen Varianten wird daher verzichtet.

[4] die einem FDN zu Grunde liegenden Matrizen haben in aller Regel eine Dimension von 2^n. Dies kommt einer 4-kanaligen Variante sehr entgegen, erschwert eine 5-kanalige aber erheblich.

5.3.1 Modifizierte Faltung

Da die Vorverarbeitungsstufe alleine nicht in der Lage war, das Ausgangssignal der Faltung für die Berechnung des späten Nachhalls aufzubereiten, wurde wie erwähnt auch die Faltung selbst mit in die Überlegungen zur Erzeugung eines geeigneten Signals einbezogen und entsprechend modifiziert.

Jede Faltung wurde zunächst in zwei Bereiche aufgeteilt, die unabhängig voneinander berechnet wurden. Der erste Teil, der den frühen Teil des Schallfelds erzeugte (ca. 300 ms), wurde dabei nicht mehr dem IIR-Netzwerk zugeführt. Nur das Ausgangssignal des zweiten Teils (ebenfalls ca. 300 ms) lag noch der Berechnung des IIR-Anteils zu Grunde. Damit konnten sowohl die großen spektralen Unterschiede zwischen Beginn und Ende der Faltung als auch die frühen Reflexionen bei der Berechnung des späten Nachhalls außen vor bleiben.

Zusätzlich wurden auch die den Faltungen zu Grunde liegenden Impulsantworten leicht modifiziert. Um einen möglichst unhörbaren Übergang zwischen erster und zweiter Faltung sowie zwischen deren Wiederholungen im IIR-Netzwerk zu ermöglichen, wurde eine kurze Überlappung (entsprechend einer Kreuzblende) der beiden Impulsantworten von ca. 80 ms integriert. Darüber hinaus wurde der zweite Teil der Impulsantwort durch eine Cosinus-Blende von 200ms augeblendet. Damit waren die vorher deutlich hörbaren Anfangs- und Endpunkte der Impulsantworten weitgehend eliminiert. Darüber hinaus war diese Methode äußerst effizient, denn die Modifikation der Impulsantworten musste nur ein einziges Mal erfolgen und beeinträchtigte die Leistungsfähigkeit des Hallalgorithmus in keiner Weise. Abbildung 5.2 zeigt schematisch die Gewinnung der zwei Teilimpulsantworten eines Kanals aus dessen Raumimpulsantwort.

5.3.2 Neue Vorverarbeitungsstufe

Die der Faltung folgende neue Vorverarbeitungsstufe setzte sich aus mehreren Bestandteilen zusammen, die in Abbildung 5.3 dargestellt sind. Der größte Teil der Vorverarbeitung wurde bereits für alle Kanäle gemeinsam berechnet. Die einzelnen Schritte waren:

Verzögerung des Ausgangssignals der Faltung

Das Ausgangssignal der zweiten Faltung eines jeden Kanals wurde zunächst um einen Wert verzögert, der in etwa einem Fünftel der Länge der zweiten Faltung entsprach.

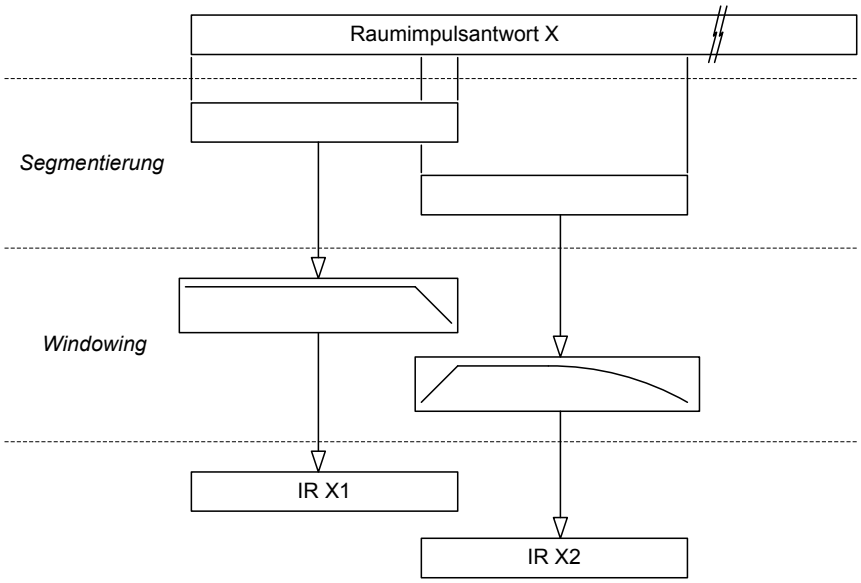


Abbildung 5.2: Gewinnung der Teilimpulsantworten

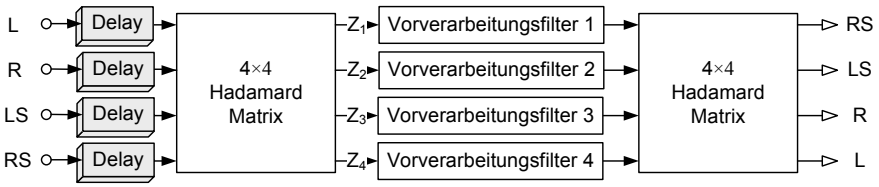


Abbildung 5.3: Vorverarbeitungsstufe

Dieser Schritt wurde damit als letzter noch kanalspezifisch ausgeführt. Die genauen Verzögerungswerte der Kanäle unterschieden sich dabei geringfügig, um eine zeitliche „Unschärfe“ hinzuzufügen, die wiederum dem flüssigen Übergang zwischen den Teilsignalen dienen sollte.

Matrizierung

Hier begann die gemeinsame Verarbeitung. Die vier Kanäle wurden mittels Multiplikation mit einer 4×4 Hadamard-Matrix für den nächsten Verarbeitungsschritt neu verteilt. Jedes Eingangssignal ging damit in jedes neu enstandene Zwischensignal mit halber Amplitude (aber teilweise umgekehrtem Vorzeichen) ein, wie in Gleichung 5.1 bis 5.4 zu sehen ist. Damit war sicher gestellt, dass die folgende Verarbeitung alle Kanäle auf einmal und in gleichem Maße beeinflusste.

$$Z_1 = 0,5L - 0,5R - 0,5LS + 0,5RS \tag{5.1}$$

$$Z_2 = 0,5L + 0,5R - 0,5LS - 0,5RS \tag{5.2}$$

$$Z_3 = 0,5L - 0,5R + 0,5LS - 0,5RS \tag{5.3}$$

$$Z_4 = 0,5L + 0,5R + 0,5LS + 0,5RS \tag{5.4}$$

Vorverarbeitungsfilter

Die erlangten Zwischensignale wurden jeweils einer Filterstruktur zugeführt, um jedem Eingangssignal in Anlehnung an die akustischen Verhältnisse in einem realen Raum mehrere diffuse und in ihrem Spektrum beschnittene Echos hinzuzufügen. In der MATLAB-Implementierung setzten sich diese Strukturen zusammen aus

- Skalierung
- Butterworth-Tiefpassfilter 1. Ordnung
- Delayline
- Allpass-Sektion

Die genaue Struktur dieser Filter wurde jedoch im Rahmen Feinabstimmung des Algorithmus noch stark angepasst.

Erneute Matrizierung

Wie sich zeigte, waren die Ergebnisse des vorgestellten Verfahrens besser, wenn die verarbeiteten Zwischensignale nochmals durch die gleiche Hadamard-Matrix geschickt und in umgekehrter Reihenfolge abgegriffen wurden[5].

Das Ausgangssignal der Vorverarbeitung wurde zum Direktsignal addiert und darüber hinaus dem folgenden Feedback Delay Network zugeführt. Jedoch wurde die erneute Matrizierung und Addition in der praktischen Umsetzung nicht einzeln ausgeführt. Stattdessen wurden die Zwischensignale direkt ins FDN zur Berechnung des späten Nachhalls geleitet, das beide Schritte mit übernahm.

5.3.3 Später Nachhall

Wie erhofft konnte für die Berechnung des späten Halls tatsächlich eine nur leicht angepasste Standardstruktur in Form eines FDN auf Basis einer 8×8 Hadamard Matrix verwendet werden (Abbildung 5.4(a)).

Die vorgenommene Anpassung resultierte im Wesentlichen aus der oben erwähnten Mitverarbeitung der Zwischensignale aus der Vorverarbeitungsstufe. Denn wie sich heraus stellte, konnte eine spezielle Konfiguration gefunden werden, die gleichzeitig die zweite Matrizierung der vorverarbeiteten Signale vornahm und die Matrixmultiplikation des späten Nachhalls ausführte. Dies wurde erreicht durch:

- Skalierung der Zwischensignale um Faktor $\frac{1}{\sqrt{2}}$ [6]
- Anlegen der Zwischensignale und Abgriff der Ausgangssignale an der Matrix nach folgendem Schema (Abbildung 5.4(b)):

$$
\begin{aligned}
In_1 = In_5 = Z_1 \qquad & \qquad L = Out_1 + Out_8 \\
In_2 = In_6 = Z_2 \qquad & \qquad R = Out_2 + Out_7 \\
In_3 = In_7 = Z_3 \qquad & \qquad LS = Out_3 + Out_6 \\
In_4 = In_8 = Z_4 \qquad & \qquad RS = Out_4 + Out_5
\end{aligned}
$$

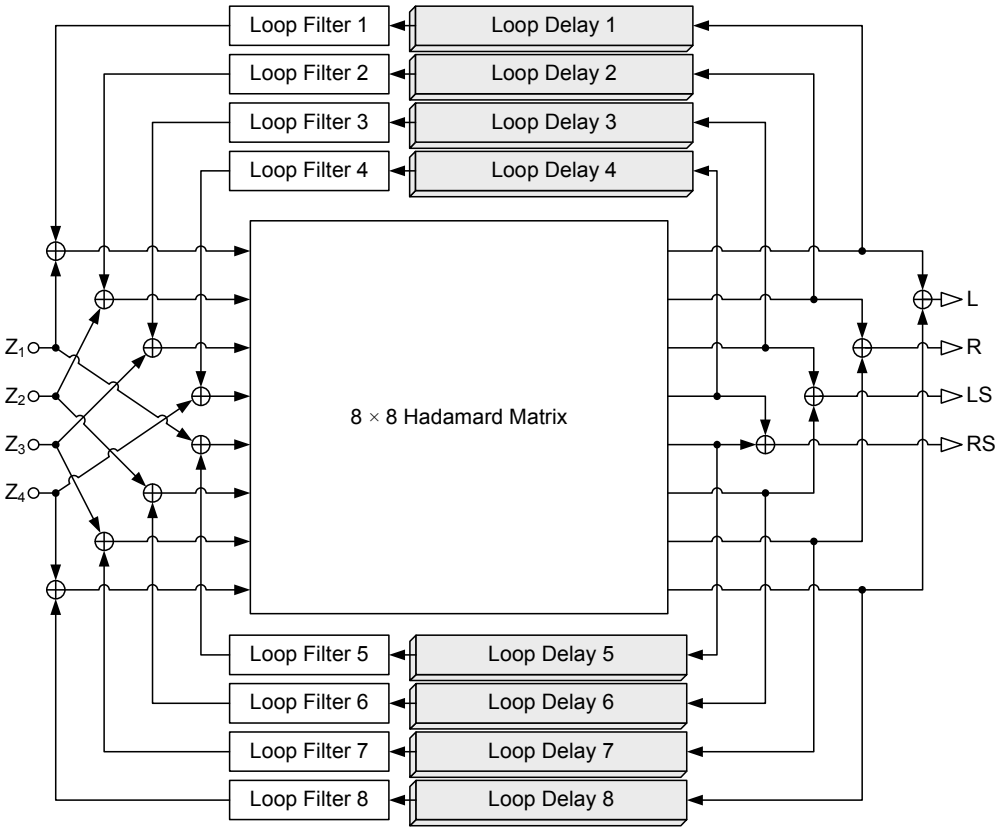
[5] Die Verwendung der transponierten Matrix für die zweite Multiplikation und Abgriff in der richtigen Reihenfolge wären natürlich ebenso möglich gewesen (die Phasendrehung der Kanäle 2 und 3 wäre sogar ausgeblieben). Da die Berechnung letztlich aber ohnehin nicht diskret erfolgte, wurde dieser Ansatz nicht weiter verfolgt.

[6] Die Skalierung ist bedingt durch die höhere Amplitude am Ausgang der erklärten Schaltung im Vergleich zur Multiplikation mit einer 4×4 Hadamard-Matrix.

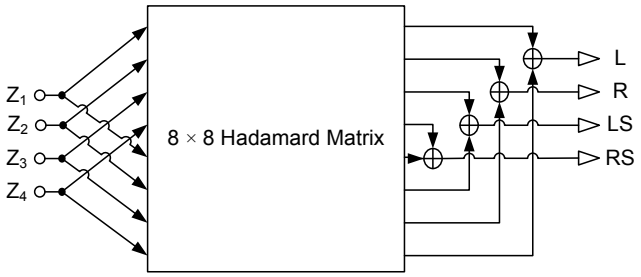
- Verlegen der Loop Delays und Filter in den Feedback-Weg

Diese Zuordnung war zwar nicht die einzige Möglichkeit zur Erlangung der gewünschten erneut matrizierten Zwischensignale, erwies sich jedoch als vorteilhaft beim Übergang von der Faltung zum FDN.

Die Längen der Delays wurden zunächst aus dem vorhandenen Algorithmus übernommen. Als Loop Filter kam eine Reihenschaltung eines Tief- und Hochpass erster Ordnung zum Einsatz. Deren Grenzfrequenz konnte frei eingestellt werden und war für alle Loop Filter identisch. Die Länge des Nachhalls wurde durch eine zusätzliche Dämpfung der Feedbacks gemäß Gleichung 2.3 bestimmt, jedoch wurde zuvor die vor dem Eintritt ins FDN durch die Faltungen erreichte Nachhallzeit von der gewünschten Dauer abgezogen.



(a) modifiziertes FDN



(b) Zuordnung der Zwischen- und Ausgangssignale (Darstellung ohne Feedbacks)

Abbildung 5.4: Erzeugung des späten Halls

6 Feinabstimmung, Integration in VSP

6.1 VST-Portierung

Zunächst wurde der vorhandene Algorithmus als VST-Plugin implementiert. Denn die MATLAB-Implementierung war zwar effizient und sehr flexibel, hatte jedoch mit fehlender Echtzeitunterstützung und geringer Portabilität zwei große Nachteile. Während letztere noch recht unproblematisch war, war die Möglichkeit der Veränderung der Parameter des Algorithmus zur Laufzeit für die vorzunehmende Feinabstimmung des Verfahrens nahezu unerlässlich.

Darüber hinaus war auch der Algorithmus zur Erzeugung der frühen Reflexionen zwar als Version für die DSPs im Mischpult und als VST-Plugin, nicht aber als MATLAB-Code implementiert. Ein Test beider Algorithmen im Verbund wurde damit durch die Portierung wesentlich erleichtert.

Details zur Implementierung des Plugins – insbesondere bezüglich der Manipulation der einzelnen Filtergruppen zur Laufzeit – finden sich in Anhang B.2.

6.2 Abstimmung des Hybridverfahrens

Die Feinabstimmung des Algorithmus erfolgte durch Hörtests seines Verhaltens bei verschiedenen Eingangssignalen und Filterparametern. Wurde dabei eine Schwachstelle bemerkt, wurde der Algorithmus jeweils an einer oder mehreren Stellen modifiziert. Deshalb erfolgt die Dokumentation dieses Vorgangs ebenfalls an Hand einer Beschreibung des jeweiligen Problems und der Maßnahmen zu seiner Lösung.

6.2.1 Hallspektrum und Ausklingverhalten

Den weitaus größten und schwierigsten Teil der Anpassungsarbeit machten Probleme beim Übergang vom frühen Hall aus der Faltung zum späten Hall aus dem FDN nötig. Es galt, zwei Hauptprobleme zu lösen:

1. Hohe spektrale Anteile, die am Ende der Faltung bereits verklungen waren, tauchten zu Beginn des späten Halls erneut auf
2. Einige spektrale Anteile des Halls aus der Faltung waren im späten Nachhall nicht mehr vertreten, andere waren zu stark. Überbetont waren vor allem die hohen Mitten, während tiefe Frequenzen völlig ausgelöscht wurden

Die vorgenommenen Veränderungen betrafen zunächst das Zusammenspiel zwischen Faltung und Vorverarbeitungsfilter. Während die Veränderungen an der Faltung eher klein waren, war bei den Vorverarbeitungsfiltern sehr viel Anpassungsarbeit zu leisten. Schließlich mussten auch die Delays im FDN verändert werden, um der neuen Situation Rechnung zu tragen.

Modifikation der Faltungen

Die Faltungen konnten geringfügig verkürzt werden, ohne die Ergebnisse zu verschlechtern. Darüber hinaus konnte ihre Überlappung ebenfalls leicht reduziert werden. Beides war natürlich im Sinne einer effizienten Verarbeitung.

Veränderung der Vorverarbeitungsfilter

Die Veränderung der Vorverarbeitungsfilter stellte den aufwändigsten Teil der Abstimmungsarbeit dar. Besonders in den tiefen Frequenzen kam es bei vielen Varianten zu Auslöschungen. Eine alternative Konfiguration, die durch Einfügen von Bandpässen die einzelnen Frequenzbereiche des Signals jeweils nur einer Filtergruppe zuordnete, führte ebenfalls zur schlechten Ergebnissen und wurde deshalb verworfen.

Die schließlich verwendete Variante ähnelt der schon in der MATLAB-Version verwendeten Konfiguration. Jedoch wurde der größte Teil der Verarbeitung von den Filtergruppen 1 und 4 übernommen, die durch entsprechende Hoch- und Tiefpässe auf getrennte Spektralbereiche wirkten. Die Vorverarbeitungsfilter 2 und 3 waren in der finalen Variante nur für eine weitere Diffusion der Signale zuständig und deshalb wesentlich geringer skaliert. Auch sie enthielten neben einem Tiefpass zum Entfernen zu hoher spektraler Anteile einen Hochpass, um Auslöschungen der aus Filtergruppe 4 stammenden tiefen Frequenzen zu verhindern.

Wegfall der Vorverarbeitungsdelays

Wie sich zeigte, waren die Vorverarbeitungsdelays unnötig und konnten eingespart werden. Zwar wurden einige kurze Delays in die Vorverarbeitungsfilter integriert, insgesamt waren die Ergebnisse aber ohne die ursprünglich angedachte pauschale Verzögerung des Ausgangs der Faltung besser. Besonders positiv wirkte sich dies bei den spektralen Problemen zu Beginn des späten Nachhalls aus.

Veränderung der Delays im FDN

Auch im Bereich des aus dem FDN stammenden späten Halls stellten sich durch auftretende Resonanzen Probleme ein. Aus diesem Grund wurde die Reihenfolge und die genauen Werte der Delays variiert.

6.2.2 Loop Filter

Die Implementierung der Loop Filter als Reihenschaltung von Hoch- und Tiefpass erwies sich als ungünstig, da eine Veränderung der Grenzfrequenzen schnell einen recht unnatürlichen Klang zur Folge hatte und die Möglichkeiten zur Beeinflussung des Ausklingsverhaltens deshalb sehr gering waren.

Deshalb wurde zunächst auf die Loop Filter des alten Hallalgorithmus zurückgegriffen. Diese basierten auf zwei Shelving Filtern für hohe und tiefe Frequenzen. Speziell im Bereich der Höhendämpfung traten damit aber die gleichen Probleme auf wie schon beim bisherigen Verfahren, denn eine geringe Dämpfung führte zu einem zu höhenlastigen Spektrum, eine hohe Dämpfung lies dagegen den Klang schnell dumpf wirken.

Deshalb wurde beschlossen, auf eine Kombination beider Varianten zu setzen. Die schließlich verwendeten Filter bestanden aus einer Reihenschaltung der alten Shelving Filter mit einem nicht durch den Anwender zu beeinflussenden Tiefpass 1. Ordnung mit einer Grenzfrequenz von 9000 Hz. Jedoch wurden die Eckfrequenzen der Shelving Filter verschoben, um der neuen Situation Rechnung zu tragen und eine sinnvolle Beeinflussung durch den Benutzer zu erlauben. Die Eckfrequenz des Low Shelf wurde auf 300 Hz, die des High Shelf auf 2800 Hz festgelegt.

Die durch den Benutzer einstellbaren Bereiche der Verstärkungs- bzw. Dämpfungswerte der Loop Filter wurden ebenfalls angepasst. Statt einem Bereich von -40 bis 0 dB für

die Höhenabsorption und -20 bis +20dB für die Tiefenabsorption wurde der wählbare Bereich jeweils auf -10 bis +10 dB eingeschränkt, da bei Verwendung größerer oder kleinerer Werte die spektralen Veränderungen generell zu einem unnatürlichen Ergebnis führten und derart große Korrekturen darüber hinaus nicht mehr nötig waren.

6.2.3 Skalierung des späten Nachhalls

Bezüglich des Pegels des späten Nachhalls traten zwei Probleme auf:

1. das Verhältnis der Lautstärken von frühem und spätem Hall war nicht optimal
2. die erzeugten Hallzeiten waren wesentlich niedriger als die eingestellten Werte

Zunächst wurde am FDN eine weitere, von der Impulsantwort abhängige Skalierung eingeführt. Diese wurde iterativ ermittelt und beim Laden der Impulsantwort aus einer Textdatei eingelesen.

Zusätzlich mussten die durch das FDN erreichten Hallzeiten drastisch erhöht werden. Dafür wurden die im Plugin eingestellten Hallzeiten vor Berechnung der Koeffizienten der Loop Filter verdoppelt, was zu Ergebnissen in der richtigen Größenordnung führte. Eine genauere Anpassung war aus zeitlichen Gründen leider nicht mehr möglich.

6.3 Integration in den bestehenden Algorithmus

Die Zusammenführung der beiden Teilalgorithmen verlief parallel zur Feinabstimmung. Dennoch soll sie hier als getrennter Punkt betrachtet werden, da sich die Aufgabenstellungen beider Vorgänge deutlich unterschieden.

6.3.1 Panning

Zunächst musste überlegt werden, ob im Nachhall eine Panning-Option nötig war und zur Verfügung gestellt werden konnte. Deshalb wurde am Ausgang des Algorithmus eine Skalierung integriert, um ein Leistungspanning des Halls zu ermöglichen.

Sie bestand aus der Summe einer frei wählbaren „Grundskalierung“ und einem richtungsabhängigen, von der aktuellen Position der Schallquelle bestimmten zweiten Summanden. Wurde das Signal genau auf einen Lautsprecher gepannt, so wurde sein Signal mit 1 skaliert, alle anderen Signale lediglich mit der Grundskalierung. Befand sich die

Schallquelle zwischen zwei Lautsprechern, so wurde die Skalierung der Signale beider Lautsprecher gemäß gängigem Leistungspanning durchgeführt.

Obwohl durch dieses Verfahren sehr gute Ergebnisse erzielt werden konnten – eine Grundskalierung von 0,6 und ein entsprechender variabler Anteil von 0,4 waren sehr überzeugend – erwies sich das Konzept schließlich als untauglich, da es für die nötige gleichzeitige Berechnung des Halls für mehrere VSP-Signale nicht geeignet war.

Eine Skalierung der Eingangssignale des Algorithmus war ebenfalls keine Möglichkeit, denn mit Einsetzen des Halls aus dem FDN glichen sich die Amplituden aller Kanäle sofort an, was einen völlig unnatürlichen Klangcharakter bewirkte.

Schließlich wurde auf die Panning-Option verzichtet, da sich zeigte, dass für Schallquellen aus nahezu allen Richtungen auch ohne Skalierung des Halls die räumliche Abbildung stabil war.

6.3.2 Eingangssignal des Hallalgorithmus

Weitere Überlegungen betrafen das zu verwendende Eingangssignal des Hallalgorithmus. Grundsätzliche Alternativen waren die Verwendung des Direktsignals, der frühen Reflexionen sowie eine Kombination beider.

Wie sich zeigte, konnte die Verwendung der frühen Reflexionen das Ergebnis jedoch nicht verbessern. In den meisten Fällen waren die Ergebnisse gleichwertig, bei ungünstiger Wahl der Parameter im Reflexions-Algorithmus teilweise aber auch schlechter. Auch eine teilweise oder vollständige Zuordnung der Reflexionen eines Kanals auf einen anderen Eingangskanal des Hallalgorithmus bewirkte kein besseres Ergebnis. Aus diesem Grund wurde von einer Verwendung der künstlich erzeugten Reflexionen als Eingangssignal schließlich Abstand genommen. Stattdessen blieb die Wahl bei der Zuführung des Mono-Direktsignals auf alle Kanäle.

6.3.3 Manipulation der Impulsantworten und Verzögerung

Die dem Algorithmus zu Grunde liegende Raumimpulsantwort musste zur Einbindung in VSP leicht angepasst werden. Dies betraf insbesondere die Entfernung des Direktsignals. Hierzu wurde zunächst ein Bereich von ca. 20 ms am Anfang der Impulsantwort jedes Kanals entfernt, der deutlich sichtbar das eintreffende Direktsignal enthielt[1]. Daraufhin wurde die Impulsantwort über einen sehr kurzen Bereich von eben-

[1] Natürlich wurden dabei die Impulsantworten aller Kanäle um den gleichen Wert verkürzt

falls 20 ms eingeblendet. Entsprechend der Länge des entfernten Abschnitts der Raumimpulsantwort wurde der Hybridalgorithms gegenüber der Generierung der frühen Reflexionen um 20 ms verzögert.

Damit beginnt der Hallalgorithmus zwar nahezu zeitgleich mit dem Algorithmus zur Reflexionerzeugung und enthält selbst ebenfalls noch frühe Reflexionen (siehe Abbildung 7.5). Dies scheint aber dem Gesamtalgorithmus nicht zu schaden, sondern bewirkt im Gegenteil ein homogeneres Klangbild.

7 Der finale Algorithmus

7.1 Blockdiagramme

In den Abbildungen 7.1 bis 7.3 ist der finale Algorithmus in Form von Blockdiagrammen dargestellt.

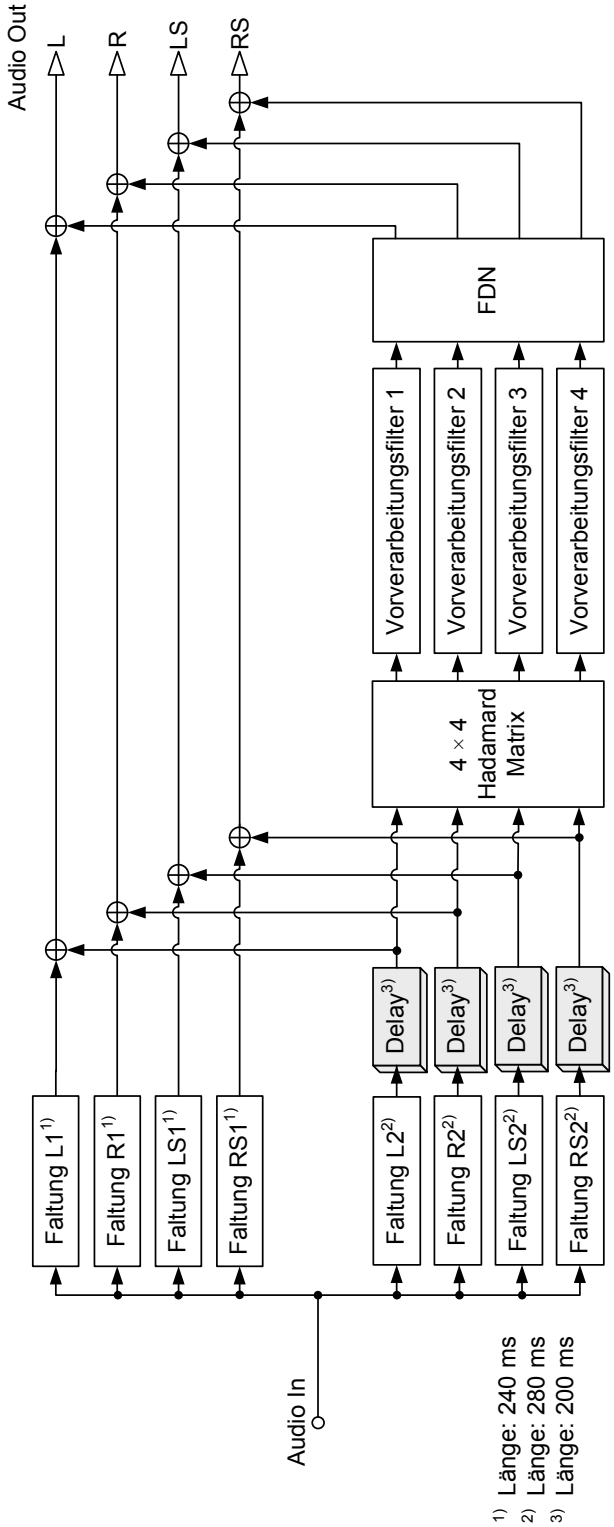


Abbildung 7.1: Überblick über den Hybridalgorithmus

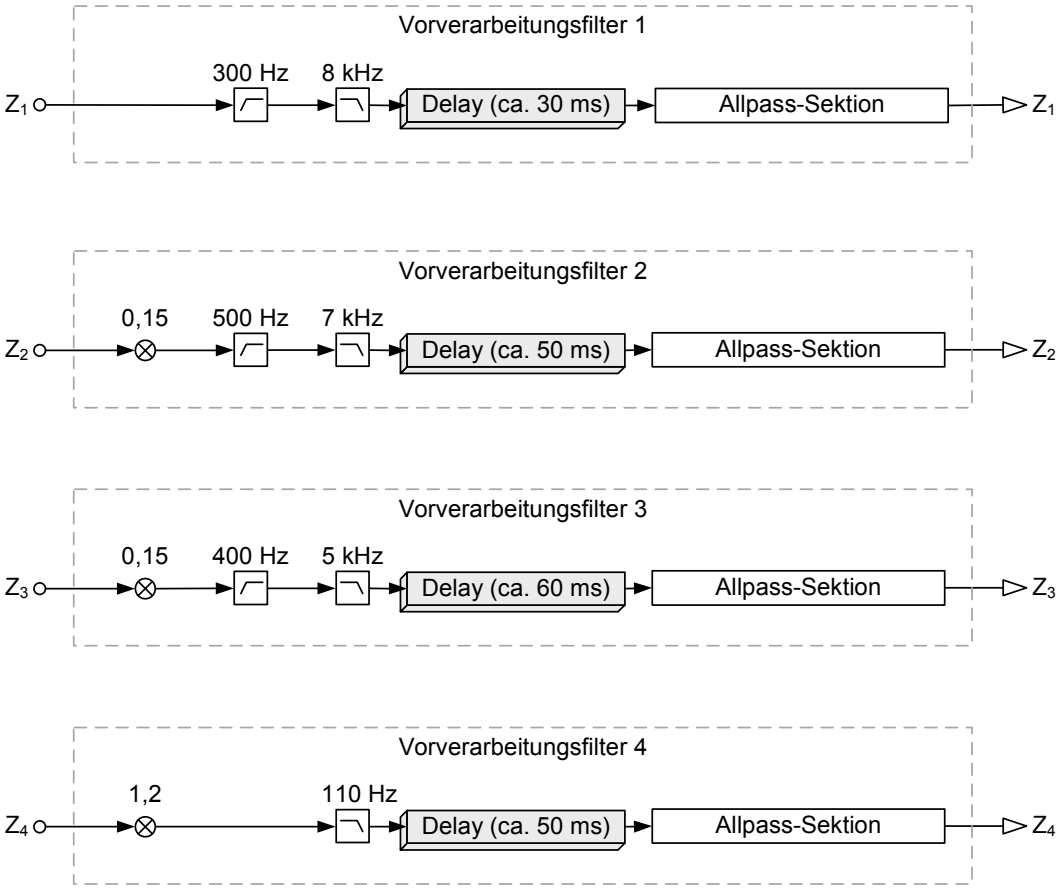


Abbildung 7.2: Vorverarbeitungsfilter

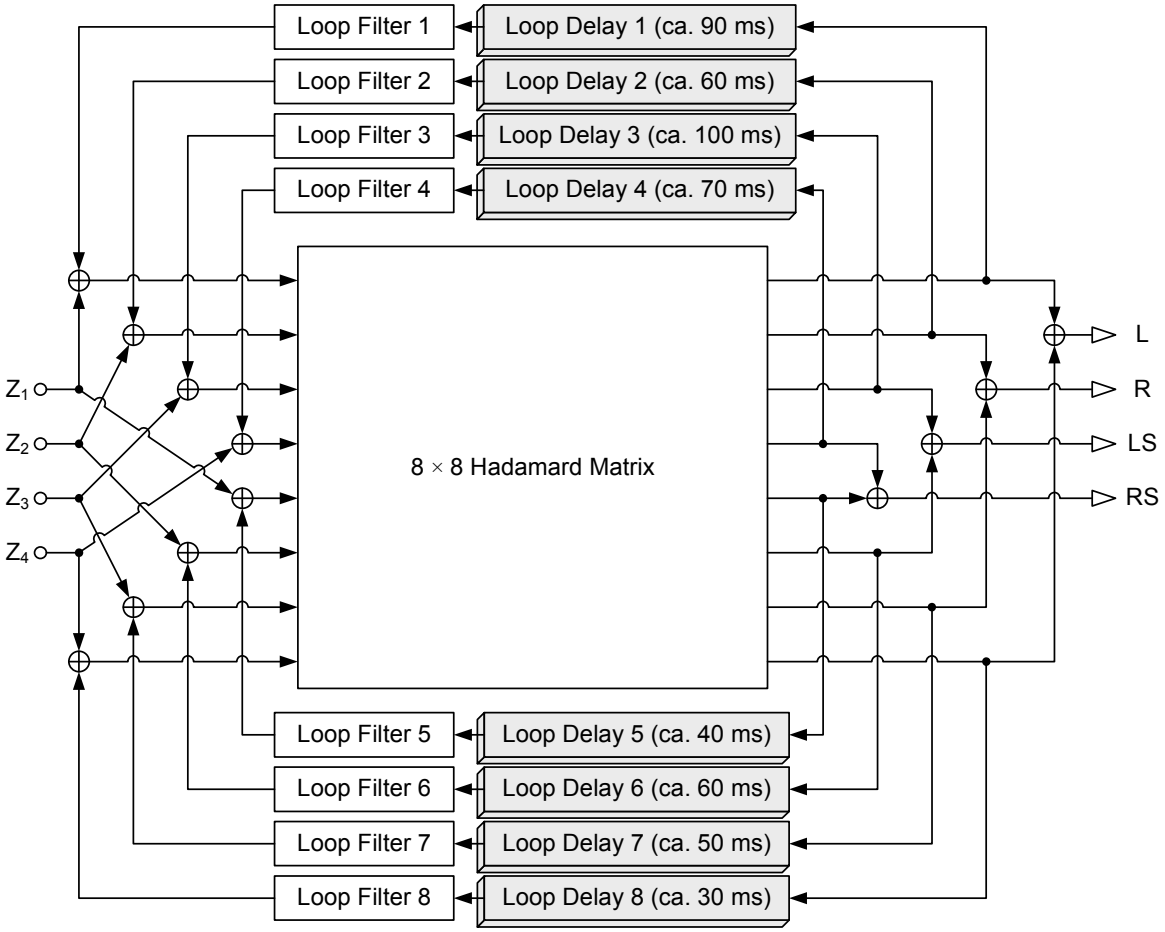


Abbildung 7.3: FDN

7.2 Beiträge der Teilalgorithmen

Die Beiträge der Teilalgorithmen zum erzeugten Schallfeld sind schematisch in Abbildung 7.4 dargestellt. Abbildung 7.5 zeigt die Anteile der Teilalgorithmen an einer real gemessenen Impulsantwort des linken Kanals.

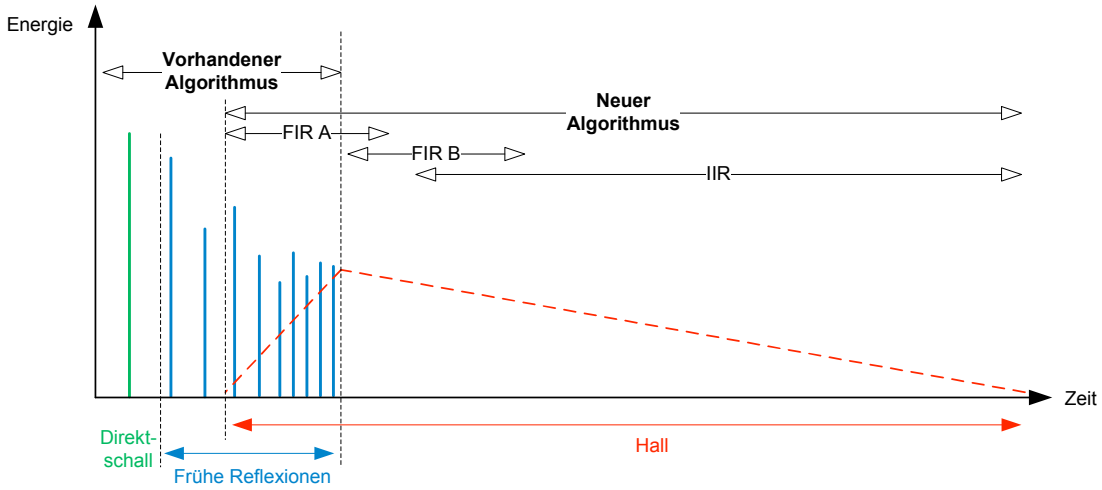


Abbildung 7.4: Beiträge der Teilalgorithmen zum erzeugten Schallfeld

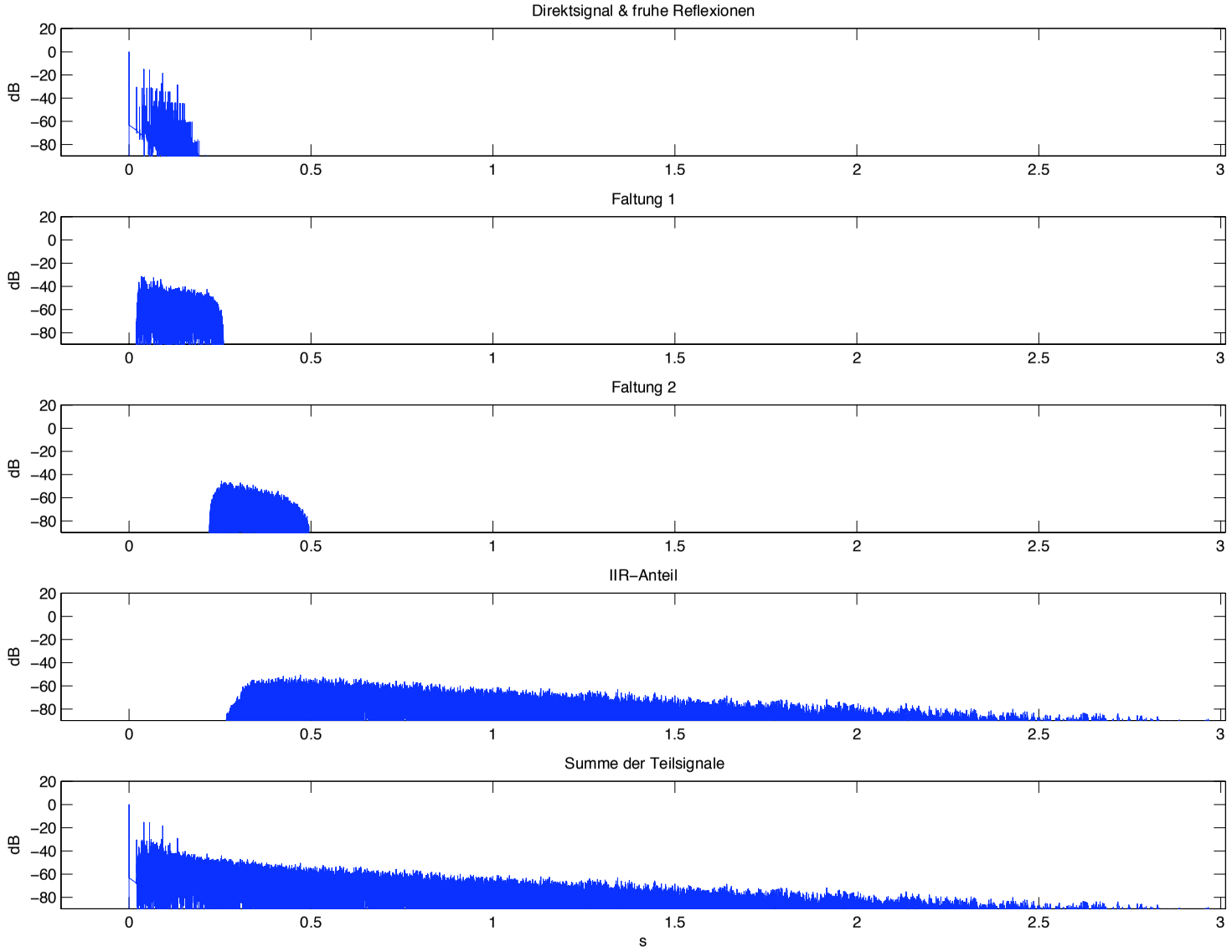
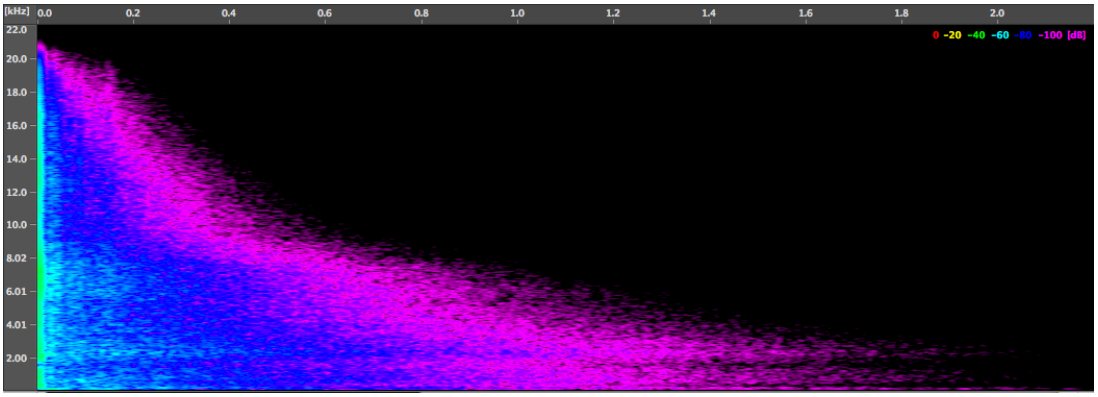


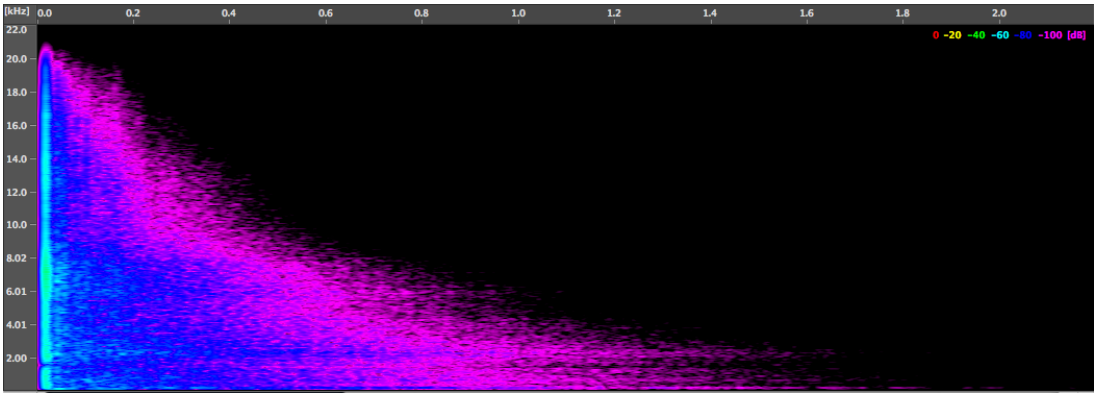
Abbildung 7.5: Bestandteile der Impulsantwort des Algorithmus

7.3 Spektrogramme

Abbildung 7.6(a) zeigt das Spektrogramm der Original-Impulsantwort des linken Kanals. Abbildung 7.6(b) zeigt das Spektrogramm der Impulsantwort des linken Kanals des Hybridalgorithmus bei einer eingestellten Nachhallzeit von 1,2 Sekunden.



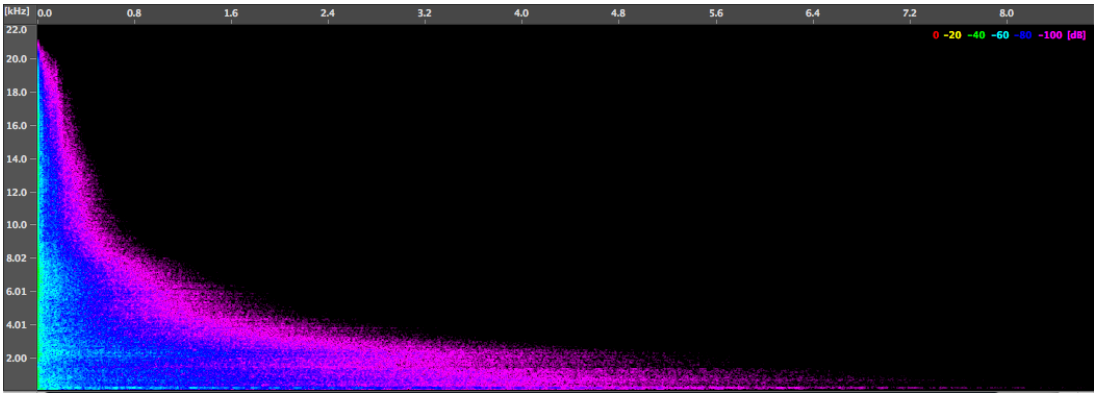
(a) Original-Impulsantwort



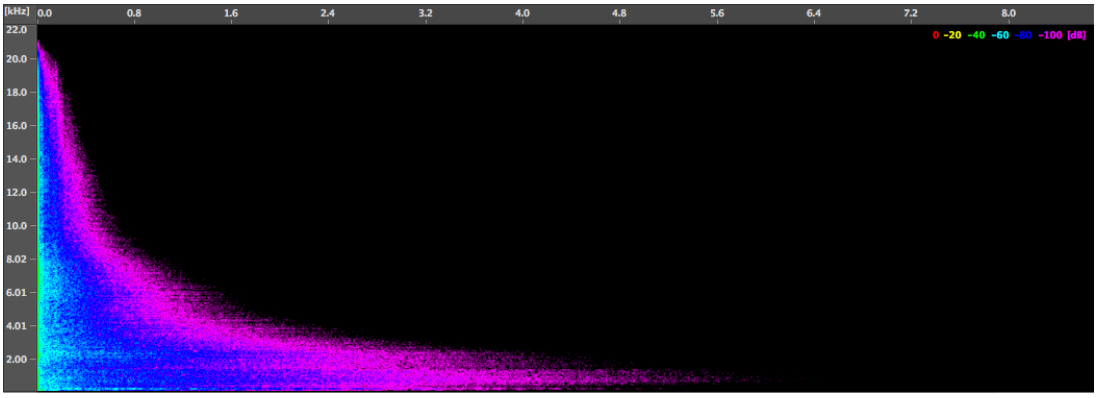
(b) Hybridalgorithmus mit Nachhallzeit 1,2 s

Abbildung 7.6: Spekrogramme der Originalimpulsantwort und einer Impulsantwort des Hybridalgorithmus

Abbildung 7.7(a) zeigt das Spektrogramm der Impulsantwort des linken Kanals des Hybridalgorithmus bei einer stark verlängerten Nachhhallzeit (eingestellter Wert: 6,0 Sekunden). Abbildung 7.7(b) zeigt die Impulsantwort bei einer Höhen- und Tiefendämpfung von jeweils -6 dB.



(a) High Absorption 0 dB, Low Absorption 0 dB



(b) High Absorption -6 dB, Low Absorption -6 dB

Abbildung 7.7: Spektrogramme des Hybridalgorithmus, eingestellte Nachhallzeit 6,0s

8 Beurteilung des Ergebnisses

8.1 Berechnungsaufwand

Da die Länge der benötigten Faltungen im Verhältnis zu einem reinen Faltungshall erheblich reduziert werden konnte, kann zunächst von einer deutlich erhöhten Leistungsfähigkeit besonders bei langen Hallzeiten ausgegangen werden. Der konstante Berechnungsaufwand für alle Hallzeiten stellt einen weiteren Vorteil dar. Die ohnehin benötigten Delays der einzelnen Faltungen ermöglichen darüber hinaus durch eine Erhöhung der Blockgrößen der Faltungen eine deutlich effizientere Implementierung als bei strikten Echtzeitanforderungen [8, S. 14].

Da jedoch die derzeitige Implementierung des Algorithmus noch in keiner Weise optimiert ist, sind konkrete Aussagen über die tatsächlichen Anforderungen des Algorithmus nicht zu treffen. Als Anhaltspunkt kann lediglich gelten, dass die aktuelle Implementierung auf dem Test-PC mit einem 2x3 GHz- Prozessor eine Belastung von ca. 40% auf nur einem der Prozessorkerne verursachte (jedoch bei erheblichen, durch die Faltung bedingten Latenzen). Die Liste der Unzulänglichkeiten der gegenwärtigen Implementierung ist jedoch lang. So findet beispielsweise die gesamte Verarbeitung im IIR-Teil Sample für Sample statt, obwohl große Teile für einen ganzen Block von Samples geschehen könnten. Auch eine verbesserte Implementierung der Vorverarbeitungsfilter könnte Vorteile bringen, denn die Berechnung der Gesamtfilter als Reihenschaltung von Einzelfiltern erscheint besonders bezüglich der Allpässe als sehr unvorteilhaft.

Es kann also davon ausgegangen werden, dass noch erhebliches Potential zur Verbesserung der Leistungsfähigkeit des Algorithmus besteht und eine optimierte Implementierung die benötigten Ressourcen wesentlich reduzieren könnte.

8.2 Klangliche Qualität, Beeinflussung durch den Anwender

Da im Rahmen der Arbeit keine ausgiebigen Hörtests außerhalb eines kleinen Personenkreises stattfinden konnten, können auch bezüglich der Klangqualität des Algorithmsu an dieser Stelle allenfalls generelle Aussagen getroffen werden. Die dieser Arbeit beiliegenden Hörbeispiele stellen jedoch zumindest einen kleinen Querschnitt des Verhaltens des Algorithmus bei verschiedenen Einstellungen dar. Eine Auflistung der Hörbeispiele und der jeweils gewählten Parameter findet sich in Anhang C. Dabei wird besonders verwiesen auf

- Beispiele 11 bis 14 zum generellen Klang des Algorithmus
- Beispiele 15 bis 17 zu den Möglichkeiten der Beeinflussung des Hallspektrums durch den Anwender
- Beispiele 18 und 19 zu realisierbaren Nachhallzeiten

Auch ohne größere Versuche lässt sich dennoch sicher festhalten, dass der Algorithmus mit seinem deutlich besseren Verhalten bei Signalen mit hohen Frequenzanteilen (Vergleich: Beispiele 4 und 14) einen Hauptkritikpunkt am alten Verfahren behebt. Auch scheint sein erheblich besseres Verhalten bei Anregung mit perkussiven Signalen außer Frage zu stehen.

Dennoch sollen einige Unzulänglichkeiten der gegenwärtigen Implementierung ebenfalls nicht unerwähnt bleiben:

- das Ausklingverhalten des Algorithmus ist ohne Zweifel noch verbesserungsfähig, was besonders bei sehr langen Hallzeiten auffällt (Beispiel 19).
- in den Beispielen mit der Trompete als Eingangssignal treten besonders zu Beginn unangenehme Resonanzen im Bereich von ca. 2,2 kHz auf, die durch steilflankiges Filtern behoben werden können. Ob dies am Eingangssignal selbst, der verwendeten Impulsantwort oder dem IIR-Algorithmus liegt, ließ sich aus zeitlichen Gründen nicht mehr zweifelsfrei überprüfen. Zwar tritt das Problem auch mit dem alten Algorithmus auf und die problematischen Frequenzen sind auch beim Eingangssignal teilweise sehr stark vertreten, dennoch scheint der neue Algorithmus hier ein etwas schlechteres Verhalten an den Tag zu legen.
- Erst gegen Ende der Arbeit wurden Probleme in der Korrelation des erzeugten Halls erkannt. In der vorliegenden Variante neigen hochfrequente Signale zu einer teilweisen Korrelation, tieffrequente Signale verhalten sich dem entgegengesetzt. Zwar konnten durch verändertes Abgreifen der Ausgangssignale am FDN

dekorrelierte Hallsignale gewonnen werden, jedoch war der Übergang von der Faltung zum FDN bei dieser Variante sehr schlecht. Für eine erneute Anpassung oder weitere Untersuchungen fehlte die Zeit.

9 Schlussbetrachtung

Durch das vorgestellte hybride Verfahren kann die Länge der für eine Hallberechnung benötigten Faltung erheblich reduziert werden. Damit stellt der Algorithmus für bestimmte Einsatzszenarien eine ressourcenschonende Alternative zum Faltunghall dar. Dies gilt besonders, da anders als bei einem reinen Faltungsalgorithmus der Berechnungaufwand für alle gewünschten Hallzeiten gleich und die Anforderungen an den Rechner damit konstant sind. Jedoch müssen einige Einschränkungen beachtet werden:

- Da keine Analyse des Ausklingverhaltens der Raumimpulsantwort vorgenommen wird, kann der Algorithmus nicht automatisch das reale Verhalten des Raums nachbilden. Stattdessen liegt dessen Festlegung im Aufgabenbereich des Anwenders. Dies kann jedoch durchaus auch als Vorteil gesehen werden, denn die daraus resultierende Variabilität des Nachhalls ist einem einfachen Faltungshall überlegen. Die nötige manuelle Skalierung des späten Nachhalls bei Integration einer neuen Impulsantwort ist dagegen als ein klarer Nachteil des aktuellen Verfahrens anzusehen.
- Durch die Struktur der momentan verwendeten Loop Filter ist die Beeinflussung des Ausklingverhaltens momentan nur recht pauschal möglich.
- Grundsätzlich besteht bei der Einbindung einer neuen Impulsantwort keine Garantie für ein gutes Verhalten des Algorithmus. Dennoch verlief die versuchsweise Einbindung anderer Impulsantworten generell zufrieden stellend.

Die Möglichkeiten der Weiterentwicklung liegen damit auf der Hand: eine Analyse der Impulsantworten und entsprechende automatische Anpassung von Skalierungen und Loop Filtern könnte eine Art „Low Cost Faltungshall“ ermöglichen.

Vor allen Dingen liegt jedoch großes Potential in der genauen Abstimmung des Zusammenspiels der Teilalgorithmen und des IIR-Algorithmus selbst. Hier konnte im Rahmen dieser Arbeit lediglich auf Basis von Versuch und Irrtum und ohne besondere Vorkenntnisse gearbeitet werden. Wie in Abschnitt 8.2 erläutert wurde, hat die aktuelle

Implementierung hier noch einige Mängel. Unter Einsatz entsprechender Ressourcen sind die erzielten Ergebnisse zweifellos zu übertreffen.

Dennoch ist geplant, den Algorithmus nach den notwendigen Überarbeitungen in das Virtual Surround Panning zu integrieren und damit in einem Produkt für den professionellen Markt einzusetzen. Der Erfolg des Experiments steht damit außer Frage.

A Verwendete Software, Versuchsdurchführung und -aufbau

Kern der Konzeption des Algorithmus war die Software MATLAB sowie die Signal Processing Toolbox, beides Produkte der Firma The MathWorks. Allerdings kam an verschiedenen Stellen nur eine Berechnung Sample für Sample in Frage, was wegen der hohen Zahl von benötigten Schleifen letztlich eine Implementierung in MATLAB sehr ineffizient gemacht hätte. Aus diesem Grund wurden die Processing-Funktionen schon sehr früh in C implementiert. Da MATLAB C-Routinen einbinden kann, konnte so eine effiziente Berechnung auch aus dessen sehr komfortabler Umgebung realisisert werden. Gleichzeitig war sowohl bezüglich der verwendeten Grundstrukturen als auch der aktuellen Filterparameter eine sehr hohe Flexibilität gewährleistet.

Während der ersten Entwicklungsphase wurden in MATLAB vor jedem Durchlauf die jeweiligen Filterkoeffizienten des gewünschten Gesamtfilters errechnet und zusammen mit dem Eingangssignal an die C-Funktion übergeben. Diese nahm die Berechnung vor und gab nach deren Abschluss das resultierende Signal an MATLAB zurück. Dort konnte es grafisch dargestellt, analysiert und (in Mono oder Stereo) angehört werden. Da MATLAB jedoch keine Surround-Wiedergabe unterstützt, wurden bei Bedarf die resultierenden Signale in eine mehrkanalige wav-Datei geschrieben und danach aus der Software Audiomulch [14] abgespielt.

Die VST-Portierung des Algorithmus wurde in Microsoft Visual Studio durchgeführt. Nach der Portierung war eine direkte Einbindung des Plugins in Audiomulch und damit eine Verarbeitung und Surround-Wiedergabe in Echtzeit möglich. Ein weiterer Vorteil Audiomulchs bestand in der Möglichkeit der freien Verbindung der Processing-Blöcke untereinander und die Kombination mit integrierten Elementen wie Delays und Skalierungen. Abbildung A.1 zeigt einen Screenshot von Audiomulch aus der späten Phase der Entwicklung, als die Teilalgorithmen zur Generierung der Reflexionen sowie der Hallerzeugung in Audiomulch eingebunden werden konnten. Die Beeinflussung der Filter des Plugins erfolgte dabei wie in Anhang B.2 beschrieben.

Die Berechnung wurde auf einem handelsüblichen PC ausgeführt, der über einen 2x3-GHz-Prozessor sowie 1 GB Arbeitsspeicher verfügte.

Während der Grobkonzeption des Algorithmus wurde die integrierte Soundkarte des Rechners zur Wiedergabe sowie Kopfhörer als Abhöre verwendet. Als die Anpassung an die Surround-Wiedergabe und klangliche Details von Interesse waren, wurde im Studio 7 der Firma Studer in Regensdorf gearbeitet. Die Ausgabe aus dem Rechner erfolgte dort über ein RME Fireface 800, das via ADAT an ein Vista 7 Mischpult angeschlossen wurde. Da dieses Pult noch über den ursprünglichen VSP-Algorithmus verfügt, konnte so auch ein Vergleich des aktuellen Stands mit dem alten Algorithmus erfolgen.

Als Abhöre dienten JBL LSR 6332 Passivmonitore an Crown Reference 1 Endstufen. Der PC selbst befand sich zur Vermeidung von Störgeräuschen in einem Nebenraum und wurde aus dem Studio lediglich ferngesteuert.

Zur Messung und Visualisierung der Impulsantworten wurde die primär zur einfachen Erstellung von Raumimpulsantworten konzipierte Software Impulse Response Utility von Apple eingesetzt, da sie sich als sehr benutzerfreundlich und für die vorgenommenen Messungen völlig ausreichend erwies.

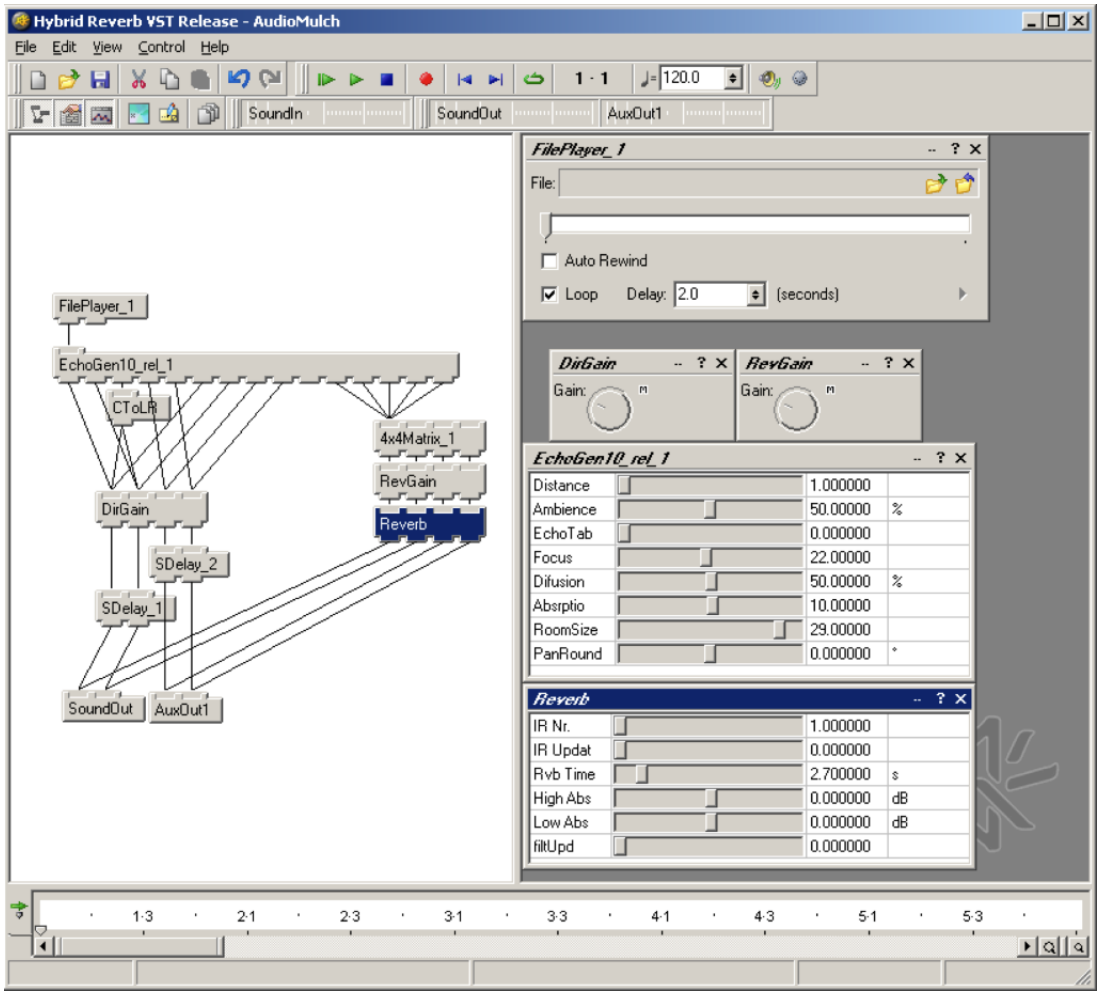


Abbildung A.1: Screenshot Audiomulch

B Aspekte der Implementierung

B.1 Filter

Für die zu implementierenden Filter musste ein eigenes, angepasstes Konzept entwickelt werden. Dies resultierte aus folgenden Anforderungen

1. Freie Filtergröße, denn sie war für alle Konfigurationen, Programmdurchläufe und Filter potentiell verschieden
2. Effiziente Berechnung sehr großer Filter mit einer Vielzahl von Koeffizienten mit Wert 0, wie sie bei Delays, Kammfiltern und Allpässen auftreten

Daraus resultierte folgende Implementierung:

Jedes Filter enthielt einen (FIR) bzw. zwei (IIR) sog. „Filter Processors“, die für die eingangs- und ausgangsseite Verarbeitung zuständig waren. Bei der Initialisierung wurde einem solchen Filter Processor lediglich ein Array beliebiger Länge mit seinen Koeffizienten übergeben.

Daraufhin wurden folgende Schritte ausgeführt:

1. Ein Ringspeicher entsprechender Größe wurde alloziert
2. Alle Koeffizienten $\neq 0$ wurden in einem Array variabler Größe abgelegt. Koeffizienten mit Wert 0 wurden schon bei der Initialierung verworfen.
3. Der Filter Processor enthielt darüber für jeden Koeffizienten einen Zeiger auf eine Position in seinem Ringspeicher. Dieser zeigte auf das jeweilige, zum Koeffizienten gehörige Sample. Zusätzlich wurde ein Zeiger auf den aktuellen Schreibpunkt gehalten.

Bei der Verarbeitung eines Samples fielen daher nur die folgenden Arbeitsschritte an:

- Eingangsseite
    1. schreiben des Eingangssamples an die entsprechende Stelle im Ringspeicher

2. initialisieren des Ausgangssamples mit Wert 0
3. Für jeden Filterkoeffizienten
   - Multiplikation mit dem jeweiligen gespeicherten Eingangssample im Ringspeicher
   - Addition des Ergebnisses zum Ausgangssample
4. Dekrementieren aller Zeiger auf den Ringspeicher, gegebenenfalls Sprung zum Ende des Speichers

- Ausgangsseite (entfällt beim FIR-Filter)

1. Für jeden Filterkoeffizienten
   - Multiplikation mit dem jeweiligen gespeicherten Ausgangssample im Ringspeicher
   - Subtraktion des Produkts vom aktuellen Ausgangssample
2. Schreiben des neuen Ausgangssamples an die entsprechende Stelle im Ringspeicher
3. Dekrementieren der Zeiger auf den Ringspeicher analog zur Eingangsseite

Damit spielte lediglich die Anzahl der Filterkoeffizienten, deren Wert ungleich 0 war, eine Rolle für den Berechnungsaufwand des Filters. Die Länge eines Delays war dagegen beispielsweise völlig unerheblich (abgesehen vom höheren Speicherbedarf).

Nachfolgend sind zur Veranschaulichung die Initialisierungs- und Verarbeitungsfunktion eines IIR-Filters sowie die entsprechenden FilterProcessor-Hilfsfunktionen aufgelistet.

```
#define EPS                             1e-22f

/*-------------------------------------------------------------------------------
        Structs
-------------------------------------------------------------------------------*/

struct filterProcessorStruct{
    float * memStart, * writePtr, * coeffs, ** coeffsPtrsOnMem;
    int memSize, nCoeffs;
};
typedef struct filterProcessorStruct FilterProcessor;

struct IIRfilterStruct{
    FilterProcessor inputSide, outputSide;
};
typedef struct IIRfilterStruct IIRfilter;
```

```
/*-------------------------------------------------------------------------------
        IIR filter initialization and processing
-------------------------------------------------------------------------------*/

void initIIRfilter(IIRfilter * filt, float * numerator, int numLength, float *
    denominator, int denomLength){
    initFilterProcessor(&(filt->inputSide), numerator, numLength);
    initFilterProcessor(&(filt->outputSide), denominator, denomLength);
    return;
}

inline void doIIRfiltering(IIRfilter * filt, float * sampleToProcess){
    doInputSideFilterProcessing(&(filt->inputSide), sampleToProcess);
    doOutputSideFilterProcessing(&(filt->outputSide), sampleToProcess);
    return;
}

/*-------------------------------------------------------------------------------
       FilterProcessor initializiation and processing
-------------------------------------------------------------------------------*/

void initFilterProcessor(FilterProcessor * filt, float * coeffs, int filterLength){
    float * tempCoeffs, ** tempCoeffsPtrsOnMem;
    int i;
    filt->nCoeffs = 0;
    filt->memSize = filterLength;

    //create temporary arrays for ALL coefficient pointers and coefficients
    tempCoeffs = (float *)calloc(filterLength, sizeof(float));
    tempCoeffsPtrsOnMem = (float **)calloc(filterLength, sizeof(float*));

    //allocate filter processor's memory on heap
    filt->memStart = (float *)calloc(filterLength, sizeof(float));
    filt->writePtr = filt->memStart;

    //copy non-zero coefficients to temp array. Create pointer to coefficient's memory
        position for every non-zero coefficient.
    for(i = 0; i < filterLength; i++){

        //ignore coefficients with value 0
        if (0 == coeffs[i])
            continue;

        tempCoeffs[filt->nCoeffs] = coeffs[i];
        tempCoeffsPtrsOnMem[filt->nCoeffs] = filt->memStart+i;
        filt->nCoeffs++;
    }

    //allocate heap space for final arrays. Copy values from temporary arrays
    filt->coeffsPtrsOnMem = (float **)calloc(filt->nCoeffs, sizeof(float *));
```

```
    filt->coeffs = (float *)calloc(filt->nCoeffs, sizeof(float));

    for(i = 0; i < filt->nCoeffs; i++){
        filt->coeffsPtrsOnMem[i] = tempCoeffsPtrsOnMem[i];
        filt->coeffs[i] = tempCoeffs[i];
    }

    //deallocate temporary arrays
    free(tempCoeffsPtrsOnMem);
    free(tempCoeffs);

    return;
}

void doInputSideFilterProcessing(FilterProcessor * filt, float * sampleToProcess){
    int i;

    //write new sample to memory. Addition of EPS is used to prevent processor from
        denormalizing
    *filt->writePtr = * sampleToProcess + EPS;

    //initialize output sample with small value
    *sampleToProcess = EPS;

    //add scaled sample from memory for every non-zero filter coefficient
    for(i = 0; i < filt->nCoeffs; i++)
        *sampleToProcess += filt->coeffs[i] * *(filt->coeffsPtrsOnMem[i]);

    //update filter pointers for next cycle
    decrementFilterProcessorMemPtrs(filt);

    return;
}

void doOutputSideFilterProcessing(FilterProcessor * filt, float * sampleToProcess){
    int i;

    //subtract scaled sample from memory for every non-zero filter coefficient (
        ignoring first output coefficient)
    for(i = 1; i < filt->nCoeffs; i++)
        *sampleToProcess -= filt->coeffs[i] * (*(filt->coeffsPtrsOnMem[i]) + EPS);

    //write new output value to memory
    *filt->writePtr = * sampleToProcess;
    //update filter pointers for next cycle
    decrementFilterProcessorMemPtrs(filt);
    return;
}

void decrementFilterProcessorMemPtrs(FilterProcessor * filt){
    int i;
```

```
    float * memEndPtr = filt->memStart+filt->memSize-1;
    float ** coeffsPtrsOnMemPtr = filt->coeffsPtrsOnMem;
    //if writePointer is already pointing to first position on memory, set it to last
        memory position.
    if(filt->writePtr == filt->memStart)
        filt->writePtr = memEndPtr;
    else
        //just decrement
        filt->writePtr--;

    //do the same for every coefficient pointer
    for(i = 0; i < filt->nCoeffs; i++){
        if((*coeffsPtrsOnMemPtr) == filt->memStart)
            *coeffsPtrsOnMemPtr = memEndPtr;
        else
            (*coeffsPtrsOnMemPtr)--;
        coeffsPtrsOnMemPtr++;
    }
    return;
}
```

B.2 VST-Plugin

Oberstes Ziel der Portierung war eine möglichst umfangreiche Beeinflussung der einzelnen Filtergruppen zur Laufzeit, deren Umsetzung im Folgenden beschrieben wird. Auf die Audioverarbeitung wird nur eingegangen, falls Änderungen gegenüber der MATLAB-Implementierung nötig waren.

B.2.1 Faltungen

Der einzige beeinflussbare Parameter der Faltungen war die ihnen zu Grunde liegende Impulsantwort. Die acht Teil-Impulsantworten wurden in MATLAB aus den vollständigen Raumimpulsantworten extrahiert, mit den nötigen Ein- und Ausblendungen versehen und jeweils Sample für Sample in eine Textdatei geschrieben. Diese wurde dann aus dem laufenden Plugin eingelesen. Da das Plugin aus mehreren Raumimpulsantworten wählen konnte, war auch ein Wechsel des zu Grunde liegenden Raums zur Laufzeit problemlos möglich. Auch eine beliebige Veränderung der Impulsantwort in MATLAB war machbar. Die neuen Teilimpulsantworten konnten in die jeweiligen Dateien geschrieben und direkt ins Plugin eingelesen werden.

In der MATLAB-Implementierung wurde die integrierte Funktion zur schnellen Faltung eingesetzt und erst die Ausgangssignale der Faltungen an die Processing-Funktion

übergeben. Im Plugin wurde für die Faltungen die Intel Integrated Performance Primitives Bibliothek eingesetzt. Sie gilt als sehr effizient und war bereits zuvor in einem ähnlichen Projekt der Harman Consumer Group – wie auch Studer ein Unternehmen im Harman-Konzern – verwendet worden. Der für die Einbindung notwendige Programmcode konnte daher relativ einfach von einem Kollegen angepasst und danach in das Plugin eingebunden werden.

Während die anderen Bestandteile der Audioverarbeitung nach wie vor Sample für Sample erfolgten, wurde die Faltung jeweils für größere Blöcke von Samples ausgeführt.

B.2.2 Delay des zweiten Faltungsteils, Vorverarbeitungsdelay und -filter

Diese Filter wurden über externe Dateien beeinflusst. Am aufwändigsten war dies für die Vorverarbeitungsfilter, denn sie sollten (wie auch in der MATLAB-Version) sowohl in der Anzahl der Koeffizienten als auch deren Wert völlig frei sein. Deshalb wurde folgendes Vorgehen gewählt:

- Die jeweiligen Filterkoeffizienten wurden in MATLAB mit den bereits vorher verwendeten Funktionen berechnet
- aus MATLAB wurde für jeden Filter eine Textdatei erzeugt, die dessen aktuelle Koeffizienten enthielt. Die erste Zeile enthielt zunächst die Anzahl von Zählerkoeffizienten, dann folgten deren genaue Werte. Gleiches galt für die zweite Zeile, die die Nennerkoeffizienten enthielt (ein FIR-Filter enthielt dementsprechend nur eine Zeile).

Ein einfacher Kammfilter, dessen Rückweg aus einem Delay von 10 Samples und einer Skalierung von 0,5 besteht, wäre damit durch eine Textdatei folgenden Inhalts repräsentiert worden:

```
1 1
11 0 0 0 0 0 0 0 0 0 0 0.5
```

Die grafische Oberfläche des Plugins bot die Möglichkeit, die letzten in MATLAB erzeugten Dateien einlesen zu lassen und entsprechende neue Filter zu erzeugen. Die Aktualisierung konnte damit aus dem laufenden Plugin geschehen.

Für die Delays wurde genauso vorgegangen. Zwar wäre auch eine deutlich einfachere Variante vorstellbar gewesen, aber da die vorgestellte Methode bereits implementiert war und fehlerfrei funktionierte, wurde ihr der Vorzug gegeben.

B.2.3 Loop Delays und Filter

Während die Loop Delays bereits zur Kompilierzeit festgelegt wurden und ansonsten nicht beeinflussbar waren, mussten die Loop Filter nach den aktuell in der grafischen Oberfläche eingestellten Werten für Hallzeit, High Cut und Low Cut berechnet werden. Sie wurden zunächst wie auch in der MATLAB-Version als eine Reihenschaltung von Hoch- und Tiefpassfilter erster Ordnung implementiert. Dies geschah folgendermaßen:

- Berechnung der Übertragungsfunkion der einzelnen Filter nach folgenden Formeln [11, S. 40]:

$$\begin{aligned} H(z) &= \frac{1}{2}(1 \pm A(z)) \qquad (LP/HP\ +/-) \\ A(z) &= \frac{c + z^{-1}}{1 + cz^{-1}} \\ c &= \frac{\tan(\pi f_c / f_s) - 1}{\tan(\pi f_c / f_s) + 1} \end{aligned}$$

- Faltung der Übertragungsfunktionen beider Filter zur Erzeugung eines neuen Filters als Reihenschaltung von Hoch- und Tiefpass [6, S. 133]:

$$H(z)_{ges} = H(z)_{HP} * H(z)_{LP}$$

- Multiplikation der Zählerkoeffizienten mit dem nach Gleichung 2.1 aus der aktuellen Nachhallzeit und dem jeweiligen Loop Delay berechneten Faktor (entspricht einer weiteren Faltung der Übertragungsfunktionen von Loop Filter und Skalierung und damit wiederum einer Reihenschaltung)

C Hörbeispiele

Die CD enthält Hörbeispiele des alten und des neuen Hallalgorithmus, jeweils in Kombination mit dem VSP-Reflexionsalgorithmus. Jedem Hörbeispiel entspricht ein Ordner, in dem sich die Signale der Surround-Kanäle L, R, C, LS, RS als Mono-Dateien mit einer Abtastrate von 48 kHz befinden.

Die Parameter der Reflexionserzeugung wurden jeweils möglichst neutral gewählt und sind für alle Hörbeispiele gleich. Das Signal befindet sich im Panorama genau mittig. Da weder Reflexionen noch Hall diesen Kanal beeinflussen, enthält der Center-Kanal aller Aufnahmen daher das unbearbeitete Direktsignal.

Zusätzlich enthält die CD eine Projektdatei der Software Magix Sequoia, in dem die Hörbeispiele angeordnet wurden.

Die einzelnen Hörbeispiele sind:

Nr.	Hallalgorithmus	Signal	Hallzeit (s)	High Abs (dB)	Low Abs (dB)
1	VSP alt	Gesang	1,0	-6	0
2	VSP alt	Trompete	1,0	-6	0
3	VSP alt	Gesang	3,0	-6	0
4	VSP alt	Trompete	3,0	-6	0
5	VSP alt	Trompete	3,0	-12	0
6	VSP alt	Trompete	3,0	-18	0
7	VSP alt	Snare Drum	1,5	-6	0
11	Hybrid	Gesang	1,2	0	0
12	Hybrid	Trompete	1,2	0	0
13	Hybrid	Gesang	5,3	0	0
14	Hybrid	Trompete	5,3	0	0
15	Hybrid	Gesang	3,2	0	0
16	Hybrid	Gesang	3,2	+10	-10
17	Hybrid	Gesang	3,2	-10	+10
18	Hybrid	Trompete	0,5	0	0
19	Hybrid	Trompete	8,0	0	0

Die Hörbeispiele 1 bis 4 sowie 11 bis 14 sollen einen Eindruck über den Klang beider Algorithmen bei verschiedenen Eingangssignalen und Hallzeiten vermitteln. Um einen Vergleich der Verfahren zu ermöglichen, wurden für beide Algorithmen bezüglich der Dämpfungswerte die vorgesehenen Standard-Einstellungen verwendet. Die eingestellten Werte der Hallzeiten wurden ebenfalls unterschiedlich gewählt, da der alte Algorithmus bei gleichen Einstellungen einen erheblich längeren Nachhall erzeugt.

Die weiteren Beispiele zum alten Algorithmus sollen dessen subjektive Probleme hervor heben. Dies sind in den Beispielen 4 bis 6 die starke Höhenbetonung und das aus der nötigen starken Höhendämpfung resultierende dumpfe Ausklingverhalten sowie in Beispiel 7 das ungünstige Verhalten bei impulsartiger Anregung (siehe Abschnitt 3.3.2).

Die weiteren Beispiel zum Hybridalgorithmus sollen einen Überblick über die Möglichkeiten zur Einflussnahme auf das Ausklingverhalten geben. In den Beispielen 16 bis 17 wurden deshalb – nach einer neutralen Einstellung in Beispiel 15 – zur Veränderung des Spektrums des Halls die Werte für Höhen- und Tiefendämpfung höchstmöglich verändert. Die Beispiele 18 und 19 zeigen mit 0,5 und 8 Sekunden eingestellter Nachhallzeit ebenfalls extreme Konfigurationen.

Literaturverzeichnis

[1] Manfred Schröder: *Natural Sounding Artificial Reverberation,* J. Audio Engineering Society, Vol. 10, No 3, 1962

[2] William Gardner: *The Virtual Acoustic Room,* Master Thesis, Massachusetts Institute of Technology, Cambridge 1982

[3] David Griesinger: *Practical Processors and Programs for Digital Reverberation*, proceedings of the AES 7th International Conference, 1989
Verwendete Version des Dokuments: http://www.davidgriesinger.com/prctpro.pdf, Zugriff am 14. September 2008

[4] Michael Dickreiter: *Handbuch der Tonstudiotechnik, Band 1,* K.G. Saur, 6. Auflage, München 1997

[5] Jürgen Meyer: *Akustik und musikalische Aufführungspraxis,* Verlag Erwin Bochinsky, 4. Auflage, Frankfurt am Main 1999

[6] Steven W. Smith: *The Scientist And Engineer's Guide To Digital Signal Processing,* California Technical Publishing, 2. Auflage, San Diego 1999

[7] Luke Dahl, Jean-Marc Jot: *A Reverberator Based On Absorbent All-Pass Filters,* proceedings of the COST G-6 Conference On Digital Audio Effects, 2000

[8] Sean Browne: *Hybrid Reverberation Algorithm Using Truncated Impulse Response Convolution and Recursive Filtering*, Master Thesis, University of Miami, Miami 2001

[9] Studer Professional Audio: *Vista Digital Mixing System Manual,* Stand vom 5. August 2003

[10] Ralph Kessler: *An Optimized Method For Capturing Multidimensional „Acoustic Fingerprints“,* AES 118th Conference, 2005

[11] Udo Zölzer (Editor)*: Digital Audio Effects*, John Wiley and Sons, Chichester 2007

[12] Werners Iländer: *Mehrkanalmusikaufnahmen mit Faltungshall,* Diplomarbeit, Hochschule der Medien Stuttgart, 2008

[13] Julius O. Smith: *Artificitial Reverberation and Spatialization*, Lecture Notes MUS420/EE367A Lecture 3, Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, 2008

[14] http://www.audiomulch.com, Zugriff am 14. September 2008

[15] http://de.wikipedia.org/wiki/Allpassfilter, Zugriff am 14. September 2008

[16] http://en.wikipedia.org/wiki/Comb_filter, Zugriff am 14. September 2008

[17] http://en.wikipedia.org/wiki/Head-related_transfer_function, Zugriff am 14. September 2008

[18] http://de.wikipedia.org/wiki/LZI-System, Zugriff am 14. September 2008