HOCHSCHULE DER MEDIEN

Masterthesis

Entwicklung eines VST Plug-Ins für ein Twin-Mikrofon zur Synthese frequenzabhängiger Richtcharakteristiken bei der Postproduction

Daniel Schröfel (21796)

-

30. September 2011

Erstprüfer: Prof. Oliver Curdt (HdM Stuttgart)

Zweitprüfer: M.Eng. Matthias Domke (Microtech Gefell)

Hochschule der Medien, Stuttgart

Elektronische Medien, Medientechnik

Eidesstattliche Erklärung

Hiermit erkläre ich, Daniel Schröfel, an Eides statt, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Stuttgart, den 30. September 2011

_____

Daniel Schröfel

Abstract

By using the technology of twin microphones with two cardioid signals it is possible to manipulate a recording in a way to create arbitrary polar patterns. This thesis describes the development of a VST plug-in that provides the feature to create the directivity of the microphone very easily and intuitively using a graphical user interface.

With the plug-in developed in this thesis completely new polar patterns can be created, which were not possible so far. Because of an FIR filter and the usage of measured values of the microphone the plug-in enables to get closer to the 'ideal' polar patterns. Another feature of the plug-in is to set up the directivity of the microphone in dependency to the frequency.

The three main chapters of this thesis describe the principles, the process of development – from the graphical user interface to the signal processing – and an evaluation of a test recording. An important objective of the development is an intuitive handling of creating the directivity in dependency to the frequency.

Kurzfassung

Die Technologie der Twin-Mikrofone ermöglicht es, durch zwei getrennt aufgezeichnete Nierensignale im Nachhinein jede mögliche Richtcharakteristik nachzubilden. Diese Arbeit beschreibt die Entwicklung eines VST Plug-Ins, durch das die flexible Einstellung der Richtcharakteristik mithilfe einer intuitiven grafischen Benutzeroberfläche vereinfacht werden soll.

Durch das in dieser Arbeit entwickelte Plug-In sind sogar Zwischenstufen bzw. völlig neue Richtcharakteristiken möglich. Aufgrund einer FIR-Filterung und durch Einbezug von Messwerten des jeweiligen Mikrofons ist es möglich, der „idealen“ Richtcharakteristik näher zu kommen. Eine weitere Besonderheit des in dieser Arbeit entwickelten Plug-Ins soll die Einstellung der Richtcharakteristik in Abhängigkeit der Frequenz sein.

Diese Arbeit teilt sich in die Grundlagen, die Entwicklung des Plug-Ins – von der Benutzeroberfläche bis zur Signalverarbeitung – und die Bewertung einer Probeaufnahme. Besonderes Augenmerk wird auf eine intuitive Bedienung der frequenzabhängigen Einstellung der Richtcharakteristik gelegt.

Inhaltsverzeichnis

1 Einleitung 1

2 Grundlagen 4

2.1 Richtcharakteristiken 4

2.2 Audio Plug-Ins 8

2.3 Möglichkeiten der Audioprogrammierung 9

2.3.1 Audio Unit 9

2.3.2 Real Time Audiosuite 10

2.3.3 Virtual Studio Technology 10

2.3.4 Zusammenfassung 11

2.4 Twin-Mikrofon 12

2.4.1 Funktionsweise 12

2.4.2 Chancen 15

2.4.3 Plug-In 17

3 Implementierung eines VST Plug-Ins für ein Twin-Mikrofon 19

3.1 Ziele 19

3.2 Gestaltung und Programmierung der grafischen Benutzeroberfläche 22

3.2.1 Visuelle Herangehensweise 22

3.2.2 Schieberegler für konstante Synthese der Richtcharakteristik 24

3.2.3 Koordinatensystem für frequenzabhängige Synthese der Richtcharakteristik 26

3.2.4 Zusammenwirken von Schieberegler und Koordinatensystem 32

3.2.5 Bypassparameter und Nutzung der Preset-Funktionen 33

3.3 Implementierung der mathematischen Grundlagen zur Erzeugung der Filterkoeffizienten für die Einstellung von frequenzabhängiger Richtcharakteristik 35

3.3.1 Berechnung der Filterkoeffizienten aus der Benutzeroberfläche über den gesamten Frequenzbereich 35

3.3.2 Umwandlung der berechneten Koeffizienten vom Frequenzbereich in den Zeitbereich (IFFT) 40

3.4 Implementierung der (Echtzeit)bearbeitung der Audiosignale mit den Filterkoeffizienten (Faltungsalgorithmus) 41

3.4.1 Einbettung der Faltung in das Plug-In 41
3.4.2 Einfache Faltung und schnelle lineare Faltung 44
3.4.3 Implementierung der einfachen Faltung 46

3.5 Probleme 47

4 Probeaufnahme und abschließende Bewertung des Plug-Ins 51

4.1 Aufnahmesituation 51

4.2 Möglichkeiten der Einflussnahme auf die Aufnahme durch das Plug-In in der Postproduction 53

4.3 Bewertung 56

5 Ausblick 58

5.1 Performance 58

5.2 Grafische Benutzeroberfläche 60

5.3 Allgemein 61

6 Fazit 63

Literatur- und Quellenverzeichnis 65

Abbildungsverzeichnis 67

1 Einleitung

> *„Großmembranmikrofone mit umschaltbarer Charakteristik sind die „Arbeitspferde“ des Tonstudios, sie dürfen in keinem professionellen Studio fehlen“*[1]

schreibt Thomas Görne in seinem Buch „Mikrofone in Theorie und Praxis“. Großmembranmikrofone mit umschaltbarer Richtcharakteristik sind aufgrund ihrer Vielfältigkeit sehr gut in vielen Aufnahmesituationen einsetzbar und somit eine Art „Allzweckwaffe“ im professionellen Tonstudio. Die meisten Vertreter dieser Riege arbeiten mit einer elektrischen Summierung der Ausgangssignale der beiden Kapseln. So lassen sich die klassischen Richtcharakteristiken Kugel, breite Niere, Niere, Superniere und Acht im Mikrofon nachbilden und ein Tonmeister hat die Möglichkeit am Mikrofon zu entscheiden mit welcher Richtcharakteristik er aufnehmen möchte.

Diese grundlegende Theorie bietet durch das einzelne Ausführen beider Kapselsignale über zwei Kabel völlig neue Möglichkeiten. Schließt man die beiden Signale über separate Kabel an ein Mischpult an, hat man die Chance jede beliebige Richtcharakteristik nachzubilden. So sind seit einiger Zeit sogenannte Twin-Mikrofone am Markt vertreten, die genau dies ermöglichen. Dadurch muss man sich nicht schon beim Aufstellen des Mikrofons für eine Richtcharakteristik entscheiden, sondern kann diese stufenlos im Regieraum einstellen. Ein Vertreter des Twin-Mikrofons ist das *„UM 930 Twin“* des Unternehmens Microtech Gefell.

Befindet man sich heutzutage im Regieraum eines professionellen Tonstudios, fällt einem neben Mischpult, Abhörmonitoren und Effektgeräten vor allem eins ins Auge: ein Computer.
Der Computer wird im Tonstudio zur Speicherung und Verwaltung der Audiodaten genutzt. Allerdings bietet ein Computer in Kombination mit Audio-Schnittstellen und

[1] Görne, Thomas: Mikrofone in Theorie und Praxis (8. Auflage), Aachen 2007, S. 74.

spezieller Software noch weit größere Chancen. So schreibt Thomas Görne im Buch „Tontechnik“ zum Einsatz des Computers im Tonstudio:

> *„Als handelsüblicher „personal computer“ (PC) wird er mit geeigneter Software und geeigneten Audio-Schnittstellen zur digitalen Workstation (engl. digital audio workstation, DAW) und kann sämtliche Studiogeräte ersetzen, von Mehrspur-Bandmaschine und Mischpult über Hall, Effektgerät und Dynamik-Prozessor bis hin zu Synchronizer, Synthesizer und Sequenzer.“* [2]

Die vielfältigen Möglichkeiten, die der Einzug des Computers in moderne Tonstudios mit sich gebracht hat, sind besonders im Bezug auf Twin-Mikrofone interessant. Wenn man die zwei Signale einzeln aus dem Mikrofon führt und erst später in einem Programm zu einer neuen Richtcharakteristik summiert, wird auch dies schon durch Software realisiert. Da in einem geeigneten Audio-Programm auch Hall- und Effektgeräte durch Audio Plug-Ins ersetzt werden, bietet es sich an, einem Tonmeister durch ein Audio Plug-In weitere Funktionen bei der Nutzung eines Twin-Mikrofons zur Verfügung zu stellen.

Das Ziel dieser Arbeit ist es, dem Tonmeister die Einstellung der Richtcharakteristik in Abhängigkeit der Frequenz durch ein Audio Plug-In zu ermöglichen. Im ersten Teil dieser Arbeit wird auf die relevanten Parameter der verschiedenen Richtcharakteristiken sowie auf die Funktionsweise eines Twin-Mikrofons eingegangen. Außerdem werden die wichtigsten Möglichkeiten der Audioprogrammierung skizziert und die Entscheidung für ein VST3 Plug-In dargelegt.
Im Hauptteil der Arbeit wird die Entwicklung des Plug-Ins von der Gestaltung der Benutzeroberfläche über die Berechnung der Filterkoeffizienten bis hin zum Faltungsalgorithmus beschrieben. Am Schluss der Arbeit werden durch die Bewertung einer Probeaufnahme die Möglichkeiten, die durch die Nutzung des Plug-Ins entstehen, aufgezeigt und es wird ein abschließendes Fazit gezogen.

[2] Görne, Thomas: Tontechnik (2. Auflage), München 2008, S. 314.

Die besondere Herausforderung dieser Arbeit liegt in der Kombination der drei Hauptbereiche Tontechnik, Informatik und Mathematik. Für das Verständnis dieser Arbeit sind grundlegende Audio-Kenntnisse zu Richtcharakteristiken, Mikrofonbau (u.a. Görne, 2007) sowie die mathematischen Grundlagen zu komplexen Zahlen und zur Fouriertransformation (u.a. Stoer, 2005) hilfreich.

2 Grundlagen

2.1 Richtcharakteristiken

Elementar für diese Arbeit sind die verschiedenen Richtcharakteristiken und deren relevante Parameter. Ein grundlegendes Verständnis dieser wird vorausgesetzt. Folgende Parameter sind entscheidend.

Der relative Abstandsfaktor gibt an, wie viel weiter ein Mikrofon als ein Druckempfänger (Kugel) von der Schallquelle entfernt sein kann, um das gleiche Verhältnis von Direkt- und Diffusschall aufzuzeichnen. Der Faktor wird in dieser Arbeit beziffert, um einen Parameter zu haben, der den Unterschied der Richtcharakteristiken auf den ersten Blick ermöglicht. Ein weiterer für diese Arbeit besonders wichtiger Parameter ist die Dämpfung des Schalls bei 90 Grad, da dieser Parameter die Schnittstelle von Benutzeroberfläche und Hintergrundberechnungen darstellt. Außerdem wird die Dämpfung von rückwärtigem Schall (180 Grad) angegeben. Ein besonders interessanter Parameter ist die relative Phasenlage des Kapselsignals einer bestimmten Richtcharakteristik. Die relative Phasenlage wird in Abbildung 2.1 mit „+“ oder „-“ gekennzeichnet.

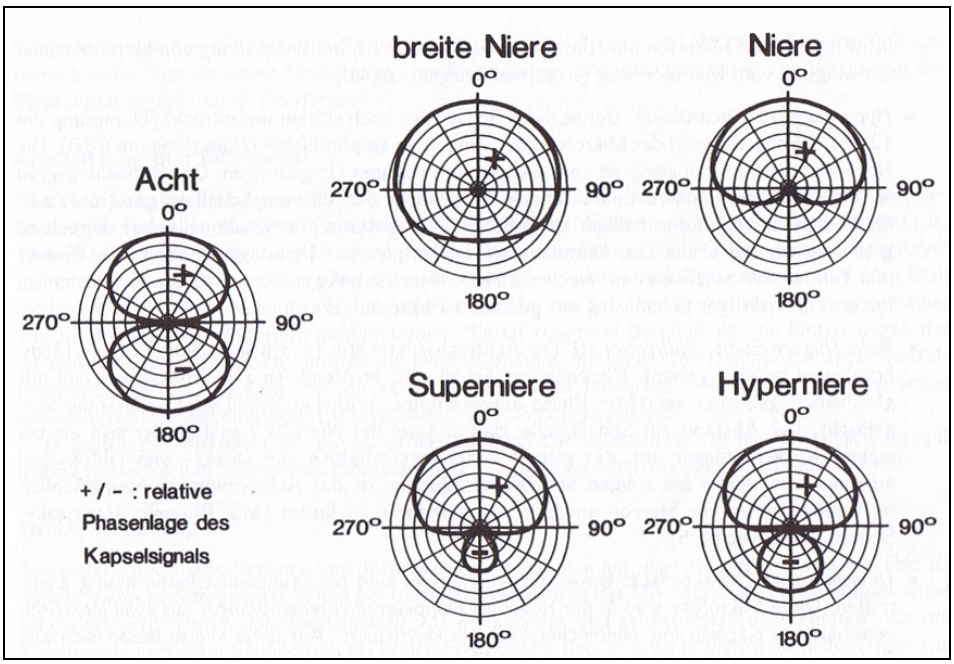

Abbildung 2.1: Mit Gradientenkapseln erreichbare Richtcharakteristiken (Görne, 2007)

Bevor genauer auf die virtuelle Kugelcharakteristik und die Auswirkungen der Richtcharakteristik bei Großmembrankapseln eingegangen wird, werden zunächst die Richtcharakteristiken ersten Grades kurz beschrieben und wichtige Parameter beziffert.

Breite Niere

Der relative Abstandsfaktor der breiten Niere ist 1.5, da rückwärtiger Schall um 10 dB und seitlicher Schall um 4 dB gedämpft werden. Die relative Phasenlage ist für alle Richtungen positiv.

Niere

Da rückwärtiger Schall theoretisch ausgeblendet wird und Schall bei 90 Grad um 6 dB gedämpft wird, ist der relative Abstandsfaktor 1.7. Die relative Phasenlage ist auch hier für alle Richtungen gleich.

Superniere

Die Superniere dämpft rückwärtigen Schall um ca. 12 dB und seitlichen Schall um ca. 9 dB. Rückwärtiger Schall wird mit gedrehter Phase aufgezeichnet. Der relative Abstandsfaktor ist 1.9.

Hyperniere

Während der Schall bei 90 Grad um 12 dB gedämpft wird, wird der Schall bei 180 Grad nur noch um 6 dB gedämpft. Eine Hyperniere kann doppelt so weit von der Schallquelle entfernt aufgestellt werden, um das gleiche Mischungsverhältnis von Direkt- und Diffusschall aufzunehmen wie ein Druckempfänger. Rückwärtiger Schall wird mit Phasendrehung aufgezeichnet.

Acht

Der relative Abstandsfaktor der Achtercharakteristik ist wie bei der Niere 1.7. Bei dieser Charakteristik wird der rückwärtige Schall mit gleichem Pegel und gedrehter Phase aufgenommen. Der Schall bei 90 Grad wird vollständig ausgeblendet.

Kugel

Als Kugel bezeichnet man in der Regel Druckempfänger, die den Schall aus allen Richtungen gleich aufnehmen. In dieser Arbeit ist meist die Rede von virtuellen Kugelcharakteristiken, die sich aus zwei Druckgradientenempfängern (Nieren) zusammensetzen. Im Folgenden soll kurz auf die Unterschiede von Druckempfängern und virtuellen Kugelcharakteristiken eingegangen werden.

Eine Eigenheit von Druckempfängern ist, dass sie hohe Frequenzen von hinten und an den Seiten ausblenden, aber gleichzeitig von vorne verstärken. So schreibt Hubert Henle:

> *„Mit zunehmender Frequenz wirkt das Mikrofon immer mehr als Hindernis für den Schall. Schallwellen mit hoher Frequenz, die von der Seite oder von hinten auf das Mikrofon treffen, werden nur teilweise oder überhaupt nicht um das Mikrofon herumgebeugt und damit ausgeblendet. Die Richtcharakteristik eines Druckempfängers wird damit bei hohen Frequenzen nierenförmig. Gleichzeitig werden hohe Frequenzen aus der Einfallsrichtung 0° von der Membran reflektiert. Dies ruft einen Druckstau vor dem Mikrofon hervor, was zu einer Anhebung des Höhenbereichs um bis zu 10 dB oder mehr führen kann.“* [3]

Während der diffusfeldentzerrte[4] Druckempfänger bei 90 Grad und 180 Grad einen Abfall der hohen Frequenzen zu verzeichnen hat, ist dieser bei der virtuellen Kugelcharakteristik nicht vorhanden. Allerdings reagiert die virtuelle Kugelcharakteristik wegen der zwei Druckgradientenempfänger auf Schallquellen im Nahfeld mit dem Nahbesprechungseffekt, der eine Bassanhebung zur Folge hat. Dies ist wiederum bei einem Druckempfänger nicht der Fall. Die virtuelle

[3] Henle, Hubert: Das Tonstudio Handbuch (5. Auflage), München 2001, S. 153.

[4] Bei Druckempfängern unterscheidet man zwischen Freifeld- und Diffusfeldentzerrung. Weitere Informationen hierzu finden sich in: Görne, Thomas: Mikrofone in Theorie und Praxis (8. Auflage), Aachen 2007, S. 39.

Kugelcharakteristik ist – im Gegensatz zum klassischen Druckempfänger – anfällig für Windgeräusche und Trittschall. Außerdem ist die Tiefbasswiedergabe bei einem klassischen Druckempfänger deutlich besser.

Das Mikrofon, für das das Plug-In in dieser Arbeit entwickelt wird, ist ein Großmembranmikrofon. Großmembranmikrofone haben besondere Eigenheiten im Bezug auf den Klang und die Richtcharakteristik. Thomas Görne schreibt im Buch „Mikrofone in Theorie und Praxis“ zum Unterschied zwischen Groß- und Kleinmembranmikrofonen:

> *„Großmembranmikrofone haben spezielle klangliche Eigenheiten, die sie für einige Anwendungen besonders attraktiv machen (harmonische Verzerrungen der Kapsel + nichtlinearer Frequenzgang = warmer Klang). Kleinmembranmikrofone sind deutlich billiger und kommen dem idealen Gradientenempfänger näher (frequenzunabhängige Richtcharakteristik = verfärbungsfreie Aufzeichnung von Diffusschall).“* [5]

Besonders im Bezug auf die Richtcharakteristik haben Großmembranmikrofone besondere Eigenheiten, dessen Hintergründe kurz erläutert werden sollen. Die Richtwirkung von Großmembranmikrofonen nimmt bei hohen Frequenzen zu, da der Membrandurchmesser mit ca. 2.5 cm relativ groß ist. Befindet sich die Wellenlänge einer Frequenz in einer ähnliche Größe wie der Membrandurchmesser (ca. 2.5 cm $\approx$ 13720 Hz) könnte diese bei Schalleinfall von 90 Grad öfter auf die Membran eintreffen. Im „Handbuch der Tonstudiotechnik“ von Michael Dickreiter steht hierzu:

> *„Senkrecht von vorn kommende Schallwellen treffen auf die gesamte Membranoberfläche gleichphasig auf. Schräg ankommende Schallwellen hingegen treffen mit unterschiedlichen Phasenlagen auf die einzelnen Membranzonen, was zu einer teilweisen Aufhebung der Membranauslenkung, also zu einer teilweisen Auslöschung des Signals führt. Ist der*

[5] Görne, Thomas: Mikrofone in Theorie und Praxis (8. Auflage), Aachen 2007, S. 176.

Membrandurchmesser gleich der Wellenlänge, so wird exakt seitlich auftreffender Schall ganz ausgelöscht.“ [6]

Es entstehen also Interferenzen und Auslöschungen, die zur Folge haben, dass hochfrequenter Schall von der Seite gedämpft aufgenommen wird. Daraus folgt, dass der aufgezeichnete Schall bei hohen Frequenzen bei 0 Grad deutlich größer ist und somit die Richtwirkung für diese Frequenzen ansteigt.
Während also bei einem Druckempfänger die Charakteristik zu hohen Frequenzen nierenförmig wird, nimmt die Charakteristik einer virtuellen Kugel, die sich aus einem Druckgradientenempfänger mit „großer“ Doppelmembran ergibt, in hohen Frequenzen eher eine Achtercharakteristik an (siehe Abbildung 2.2).[7]

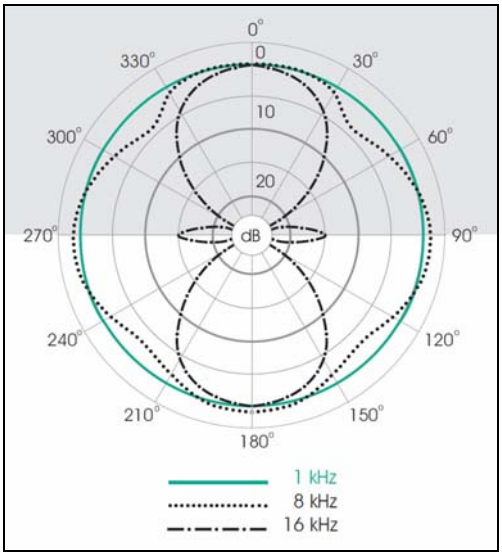

Abbildung 2.2: Polardiagramm für das "UM 930 Twin" des Unternehmens Microtech Gefell bei 1 kHz, 8 kHz und 16 kHz (Microtech Gefell)

2.2 Audio Plug-Ins

Der Einzug des Computers in die Tonstudios revolutionierte den Studioalltag enorm und brachte viele neue Möglichkeiten mit sich. Denn am Computer werden mittlerweile nicht nur die Aufnahme und der Schnitt der Audiosignale durchgeführt. Es ist heute auch möglich die Aufnahmen am Computer nachzubearbeiten und ein

[6] Dickreiter, Michael, et al.: Handbuch der Tonstudiotechnik (7. Auflage), München 2008, S. 128 ff.

[7] Dickreiter, Michael, et al.: Handbuch der Tonstudiotechnik (7. Auflage), München 2008, S. 149.

Musikstück zu mischen. Im Extremfall bedeutet das, dass der Computer als komplettes virtuelles Studio[8] fungiert und alle Möglichkeiten eines klassischen Tonstudios in sich vereint. In den meisten Tonstudios findet man allerdings Mischformen des klassischen und des virtuellen Tonstudios. Auch wenn viele Aufgaben noch mit den klassischen Werkzeugen bzw. Effektgeräten des Tonmeisters ausgeführt werden, kann man einige dieser Aufgaben schon mit Audio Plug-Ins am Computer umsetzen.

Um den Computer als virtuelles Studio bzw. als Aufnahmegerät nutzen zu können, bedarf es natürlich noch einer bestimmten Software. Die hierfür verwendeten Programme werden im Allgemeinen als Sequenzer oder DAW (digital audio workstation) bezeichnet und im Verlauf dieser Arbeit Sequenzer-Software genannt. Vier der bekanntesten Programme der Sequenzer-Software sind *ProTools*, *Sequoia*, *Cubase* und *Logic*.

Wenn man also das klassische Tonstudio mit dem virtuellen Studio vergleicht, so kann man sagen, dass die Effektgeräte im klassischen Tonstudio im virtuellen Studio die Audio Plug-Ins sind. So wird ein Kompressor z.B. nicht mehr als Gerät eingeschliffen, sondern einfach durch ein Plug-In in der Sequenzer-Software realisiert. Mittlerweile finden sich die meisten elektronischen Geräte wie Kompressoren oder Limiter auch als Audio Plug-In im Sortiment der Hersteller. Des Weiteren entstand durch die Chance des virtuellen Studios ein neuer Markt, auf dem viele neue Audio Plug-Ins angeboten werden.

Da verschiedene Programme bzw. Hersteller für Sequenzer-Software am Markt vertreten sind, gibt es auch verschiedene Möglichkeiten der Audioprogrammierung, auf die im folgenden Kapitel genauer eingegangen wird.

2.3 Möglichkeiten der Audioprogrammierung

2.3.1 Audio Unit

Eine Audio Unit (AU) ist ein Audio Plug-In, das für das Betriebssystem Mac OS X geeignet bzw. entwickelt worden ist. Solch ein entwickeltes Plug-In ist speziell an die

[8] Als virtuelles Studio ist hier ein Computer gemeint, der alle elektronischen Geräte eines Tonstudios durch dementsprechende Software in sich vereint und somit ohne zusätzliche Geräte auskommt.

Mac-Technologie angepasst und läuft auch nur auf diesem Betriebssystem. Somit ist die Schnittstelle Audio Unit natürlich sehr eng verbunden mit der Sequenzer-Software, die speziell für Mac-Computer entwickelt wird.
Der schon genannte und wohl bekannteste Vertreter dieser Riege ist *Logic*. Audio Unit hat in *Logic*-Version 6 die VST-Schnittstelle des Unternehmens Steinberg abgelöst, die seitdem nicht mehr unterstützt wird.

2.3.2 Real Time Audiosuite

Die Real Time Audiosuite (RTAS) ist ein Format von Audio Plug-Ins und wurde vom Unternehmen Digidesign – mittlerweile Avid Technology – entwickelt. In diesem Fall ist das Format nicht direkt an ein Betriebssystem gebunden, sondern an eine Sequenzer-Software.
Avid Technology (ehemals Digidesign) ist das Unternehmen, welches hinter der Sequenzer-Software *ProTools* steht. Somit ist die Real Time Audiosuite an diese Software gekoppelt und grundsätzlich sind Plug-Ins auch nur mit einer Variante des Programms *ProTools* lauffähig. *ProTools* ist eine sehr weit verbreitete Seqenzer-Software und besonders in professionellen Tonstudios sehr beliebt.

2.3.3 Virtual Studio Technology

Die Schnittstelle Virtual Studio Technology (VST) ist die am weitesten verbreitete Schnittstelle auf allen Plattformen. Die Schnittstelle wurde vom Unternehmen Steinberg Media Technologies entwickelt und ist somit auf den ersten Blick speziell für die von Steinberg entwickelte Sequenzer-Software *Cubase* entwickelt worden. Allerdings laufen VST Plug-Ins auch mit anderer Sequenzer-Software.
Im Prinzip ist VST die Möglichkeit Audio Plug-Ins für Sequenzer-Software zu entwickeln, die mit dem Betriebssystem Microsoft Windows verknüpft ist. So laufen VST2 Plug-Ins nicht nur mit den Steinberg-Technologien *Cubase* und *Nuendo*, sondern u.a. auch mit *Magix Sequoia* .

Der über Jahre verbreitete Standard ist VST 2, der 1999 veröffentlicht wurde. Besonders Plug-Ins der Version VST 2.4 sind sehr weit verbreitet. Allerdings ist seit 2008 auch die VST 3 Norm veröffentlicht. Während es bei VST 2 relativ wenig

Vorgaben und Regeln gab und dem Entwickler sehr viel Freiraum gelassen wurde, ist VST 3 sehr viel strukturierter und strenger dem Entwickler gegenüber. Die wohl wichtigste Änderung ist die deutliche Trennung von Oberfläche und Prozessor in VST 3.

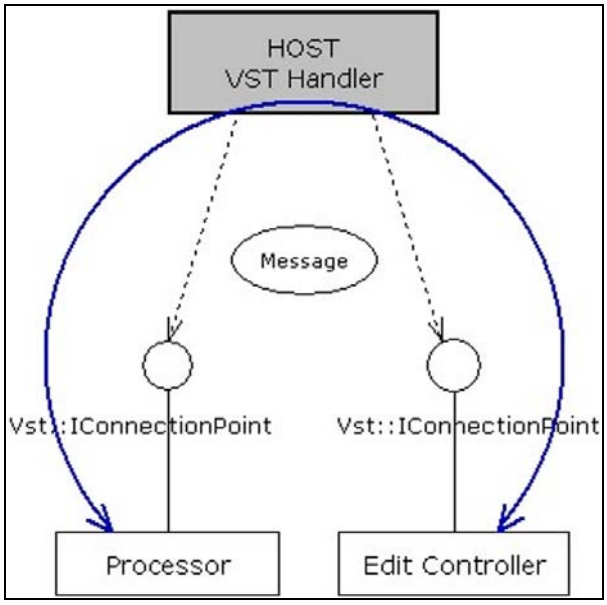

Abbildung 2.3: Kommunikation zwischen den zwei Komponenten Processor und Controller (Steinberg Media Technologies)

Außerdem gibt es im VST SDK 3.5 auch einen Wrapper, der es ermöglicht aus VST3 Plug-Ins VST2 Plug-Ins zu erstellen. VST Plug-Ins unterscheiden kategorisch Plug-Ins für virtuelle Instrumente (Midi-Instrumente) und Effekt-Plug-Ins (z.B. Equalizer).

2.3.4 Zusammenfassung

Nachdem nun auf die drei wichtigsten Möglichkeiten der Audio Plug-In Programmierung eingegangen wurde, soll in diesem Kapitel anhand einer Zusammenfassung die Entscheidung für die VST-Schnittstelle deutlich gemacht werden.

VST ist nicht nur die Technologie, die auf allen Plattformen am meisten verbreitet ist, sondern auch die Technologie, die am wenigsten eingeschränkt bei der Verwendung mit verschiedenen Programmen und Betriebssystemen ist. VST ist nicht an ein Betriebssystem oder an eine Sequenzer-Software gebunden, was die weite Verbreitung und eine vielseitige Nutzung möglich macht.

Auch wenn zurzeit immer noch sehr viele VST2 Plug-Ins entwickelt werden, festigt sich VST 3 mittlerweile als der neue Standard und besonders die großen Unternehmen veröffentlichen immer öfter auch VST 3 Varianten ihrer schon bekannten Plug-Ins. VST 3 soll laut Steinberg durch die Umstrukturierung und die klaren Vorgaben für den Entwickler auch deutlich stabiler laufen als VST 2. Nicht zuletzt wegen des Wrappers, der aus VST3 Plug-Ins VST2 Plug-Ins generieren kann, fiel die Entscheidung in dieser Arbeit zugunsten der Entwicklung eines VST3 Plug-Ins. Als Entwicklungsumgebung für VST3 Plug-Ins bietet sich Microsoft Visual Studio an, das in der Version Microsoft Visual Studio 2008 zusammen mit dem VST SDK 3.5.0 für diese Arbeit verwendet wurde.

2.4 Twin-Mikrofon

2.4.1 Funktionsweise

Die grundlegende Funktionsweise eines Twin-Mikrofons basiert auf der Technik der Studiomikrofone mit umschaltbarer Richtcharakteristik. Großmembrankondensatormikrofone mit umschaltbarer Richtcharakteristik sind nun schon seit vielen Jahren Bestandteil professioneller Tonstudios. Umschaltbare Kondensatormikrofone etablierten sich weltweit als Standard.[9]

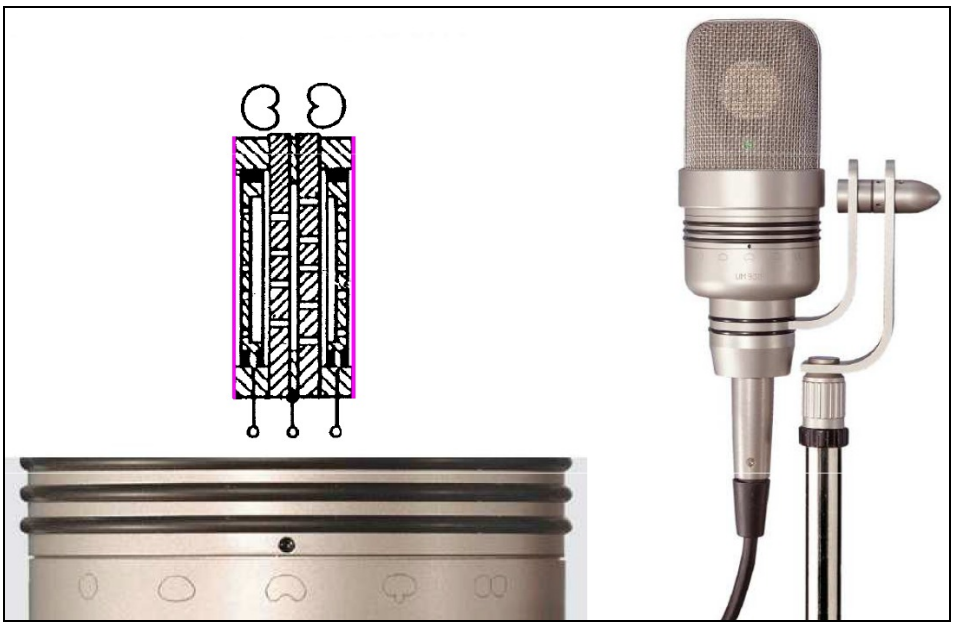

Abbildung 2.4: Aufbau von Mikrofonen mit umschaltbarer Richtcharakteristik (Microtech Gefell)

[9] Vgl. Görne, Thomas: Mikrofone in Theorie und Praxis (8. Auflage), Aachen 2007, S. 74.

In klassischen umschaltbaren Kondensatormikrofonen finden sich zwei Membranen eines Doppelgradientenempfängers, der sogenannten Braunmühl-Weber-Kapsel[10]. Die Membranen sind unabhängig angeschlossen und liefern jeweils ein Nierensignal. Durch Veränderung bzw. Umpolung der Vorspannung kann die Amplitude bzw. die Polarität der Kapseln verändert werden. Werden beide Membranen nicht verändert, addieren diese sich zu einer virtuellen Kugelcharakteristik. Wird die Polarität der hinteren Nierenkapsel gedreht, so ergibt sich eine Achtercharakteristik. Durch verschiedene Einstellungen von Amplitude und Polarität sind auch weitere Charakteristiken möglich.[11]

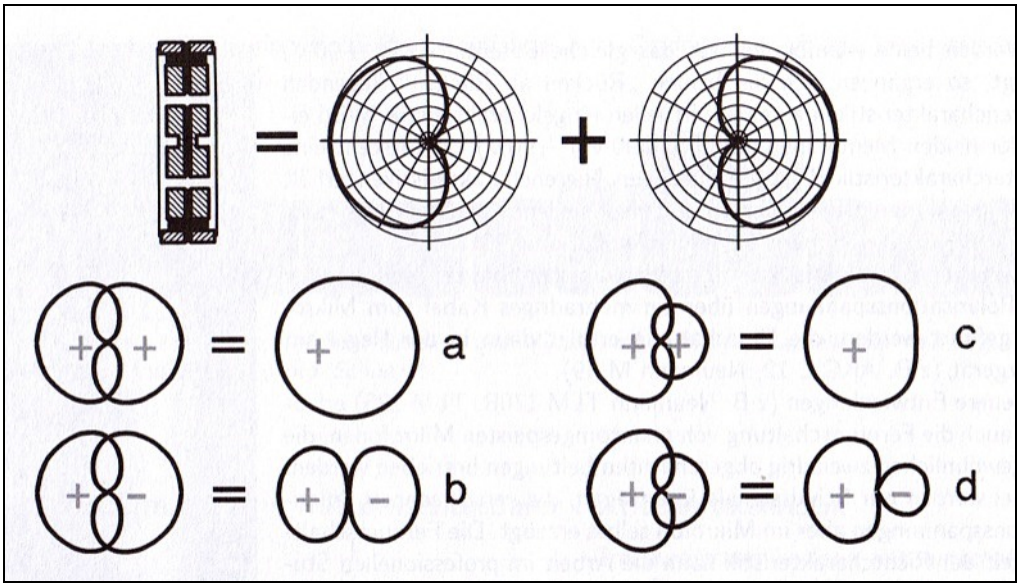

Abbildung 2.5: Funktionsweise einer elektrisch umschaltbaren Kapsel (Görne, 2007)

Spricht man im Zusammenhang mit umschaltbaren Mikrofonen oder Twin-Mikrofonen von einer Kugelcharakteristik handelt es sich meist um eine virtuelle Kugelcharakteristik eines Doppelgradientenempfängers und nicht um einen Druckempfänger.

[10] Diese Technik geht auf Arbeiten von Walter Weber und Hans Joachim von Braunmühl zurück. Weitere Informationen finden sich in: Görne, Thomas: Mikrofone in Theorie und Praxis (8. Auflage), Aachen 2007, S. 65.

[11] Vgl. Görne, Thomas: Mikrofone in Theorie und Praxis (8. Auflage), Aachen 2007, S. 74 ff.

Wenn man also zwei Nierensignale Rücken-an-Rücken zur Verfügung hat, kann man durch Veränderung der Amplitude und der Polarität des hinteren Signals und folgender Addition der beiden Kapselsignale jede Richtcharakteristik erster Ordnung nachbilden. Würde man die Signale einzeln aus dem Mikrofon führen und an ein Mischpult oder eine Sequenzer-Software anschließen, könnte man dort mit den zwei Kanälen jede beliebige Richtcharakteristik erster Ordnung erzeugen, indem man je nach Wunsch-Richtcharakteristik die Lautstärke des Signals der hinteren Niere anpasst und evtl. die Phase des Kanals dreht.

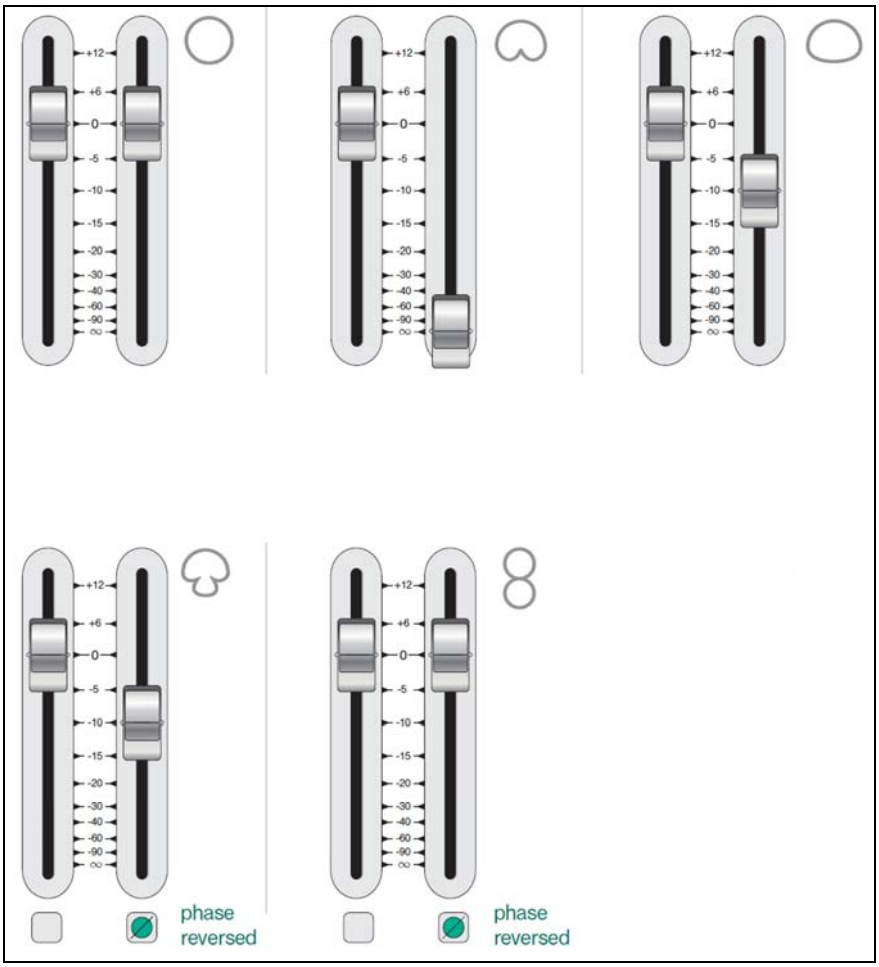

Abbildung 2.6: Einstellungen der Richtcharakteristik am Mischpult (Microtech Gefell)

Auf dieser Theorie basiert die Idee eines Kondensator-Mikrofons mit Doppelgradientenempfänger, das die Signale über zwei einzelne Kabel ausgibt. So wird ein Mikrofon mit Doppelmembrankapsel und zwei einzelnen Signalwegen für die zwei Nierenkapseln als „Twin-Mikrofon“ bezeichnet.

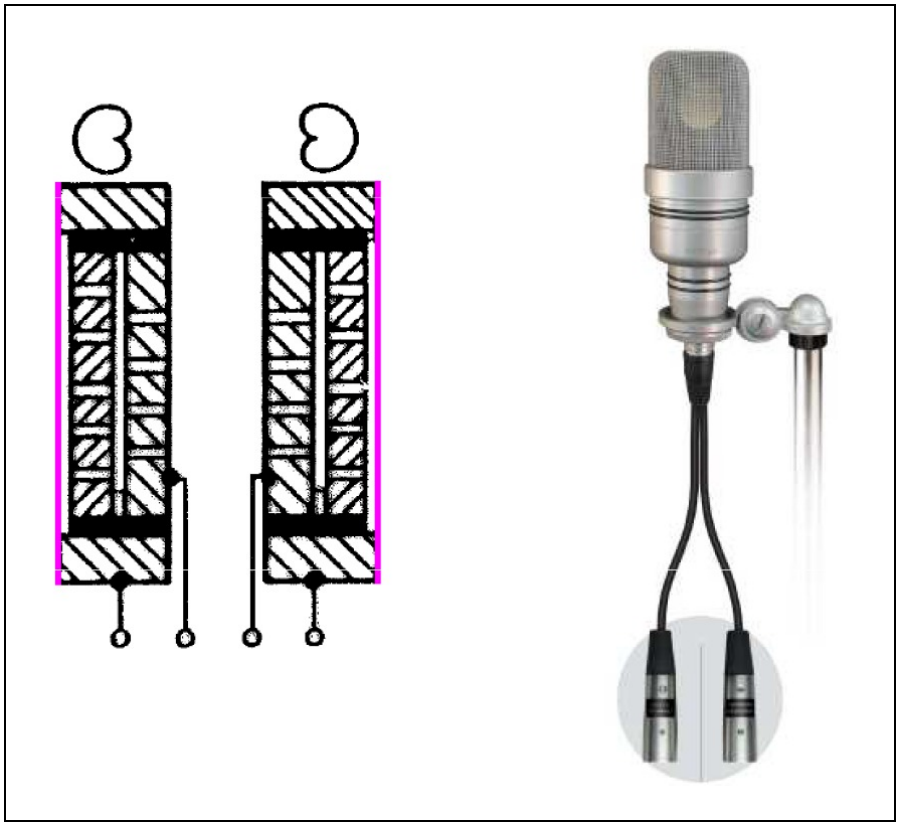

Abbildung 2.7: Aufbau von Twin-Mikrofonen (Microtech Gefell)

Die Entwicklung der Twin-Mikrofone – besonders im Hinblick auf die Weiterverarbeitung der zwei Signale – steht noch am Anfang und könnte in Zukunft viele neue Möglichkeiten und Chancen bieten.

2.4.2 Chancen

Das in dieser Arbeit relevante Mikrofon *„UM 930 Twin“* ermöglicht es nicht nur, zwei Nieren Rücken-an-Rücken auszugeben, sondern gibt immer die vordere Niere aus, während die zweite Richtcharakteristik durch klassische Umschalttechnik frei wählbar ist. Die anwählbaren Richtcharakteristiken des Umschaltringes sind Kugel, breite Niere, Superniere, Acht und Niere (hinten). So hat man die Möglichkeit zwei unterschiedliche Richtcharakteristiken aufzuzeichnen und kann somit z.B. mit nur einem Mikrofon durch Aufzeichnung einer Kugel und einer Niere ein Straus-Paket[12] nachbilden. Möchte man zwischen allen Charakteristiken frei wählen können, sollte man natürlich neben

[12] Der Diplom-Tonmeister Volker Straus hat sich durch eine Kombination von Kugel und Niere in einem „Paket“ bei Aufnahmen eine breite Niere generiert.

der vorderen Niere auch die hintere Niere aufzeichnen, sodass die Richtcharakteristik am Pult oder in der Sequenzer-Software frei wählbar ist. Dadurch sind sogar Zwischenstufen möglich. Da auch die durch den Umschaltring generierte Kugel lediglich durch Summierung der beiden Nierensignale generiert wird, macht die Nutzung des Mikrofons als Twin-Mikrofon mit zwei Ausgängen nur Sinn, wenn man die zwei Nierensignale ausgibt, da man mit diesen „alle" Charakteristiken nachbilden kann. Die Chance nachträglich zu entscheiden, welche Richtcharakteristik die perfekte für eine bestimmte Aufnahmesituation ist, sowie die Vielseitigkeit durch die freie Wahl der Richtcharakteristik sind sicher zwei elementare Vorteile, die für die Nutzung eines Twin-Mikrofons sprechen.

Die Möglichkeit, die Richtcharakteristik vom Arbeitsplatz des Tonmeisters im Prinzip wie mit einer Fernbedienung zu variieren, bringt einen erheblichen Vorteil mit sich. So schreibt Thomas Görne in seinem Buch „Mikrofone in Theorie und Praxis":

> *„Die Fernumschaltbarkeit der Richtcharakteristik kann die Arbeit im professionellen Tonstudio wesentlich erleichtern."* [13]

Die Nutzung eines Twin-Mikrofons kann den Arbeitsalltag vereinfachen und besonders im Bezug auf die Weiterverarbeitung der beiden Signale durch ein elektronisches Gerät oder ein Software-Plug-In können sich weitere Vorteile ergeben. So können nicht nur die Richtcharakteristiken nachgebildet werden, wie dies bei einem umschaltbaren Mikrofon der Fall ist, sondern man kann z.B. durch eine FIR-Filterung[14] der gewählten Richtcharakteristik deutlich näher kommen als das bis dato der Fall war. Eine weitere Innovation könnte die Einstellung der Richtcharakteristik in Abhängigkeit der Frequenz sein.

[13] Görne, Thomas: Mikrofone in Theorie und Praxis (8. Auflage), Aachen 2007, S. 75.

[14] Ein FIR-Filter (finite impulse response filter) ist ein Filter mit endlicher Impulsantwort.

2.4.3 Plug-In

Im Hinblick auf die Weiterverarbeitung der beiden Signale eines Twin-Mikrofons und den bereits genannten Chancen drängt es sich auf, die mögliche Verwendung eines Audio Plug-Ins für die Einstellung der Richtcharakteristik genauer zu erörtern.
Das Plug-In würde aus den beiden Signalen des Twin-Mikrofons ein neues Signal mit der gewählten Richtcharakteristik erstellen. Dies bringt auf der einen Seite den enormen Vorteil mit sich, dass man unabhängig von externen Geräten ist und allein durch ein Software-Plug-In während der Aufnahme oder im Postprocessing die Richtcharakteristik einstellen kann.

Auf der anderen Seite birgt dies natürlich die Chance noch weitere Funktionen in das Plug-In einzugliedern, die den Tonmeister bei seiner Arbeit unterstützen.
Eine Möglichkeit wäre, durch eine FIR-Filterung und den beiden Signalen der „idealen“ Richtcharakteristik näher zu kommen, denn kein Mikrofon ist in der Lage, die Richtcharakteristik über den gesamten Frequenzbereich perfekt wie aus dem Lehrbuch abzubilden. Durch die FIR-Filterung könnte man der Richtcharakteristik erheblich näher kommen, wie die Bilder in Abbildung 2.8 zeigen

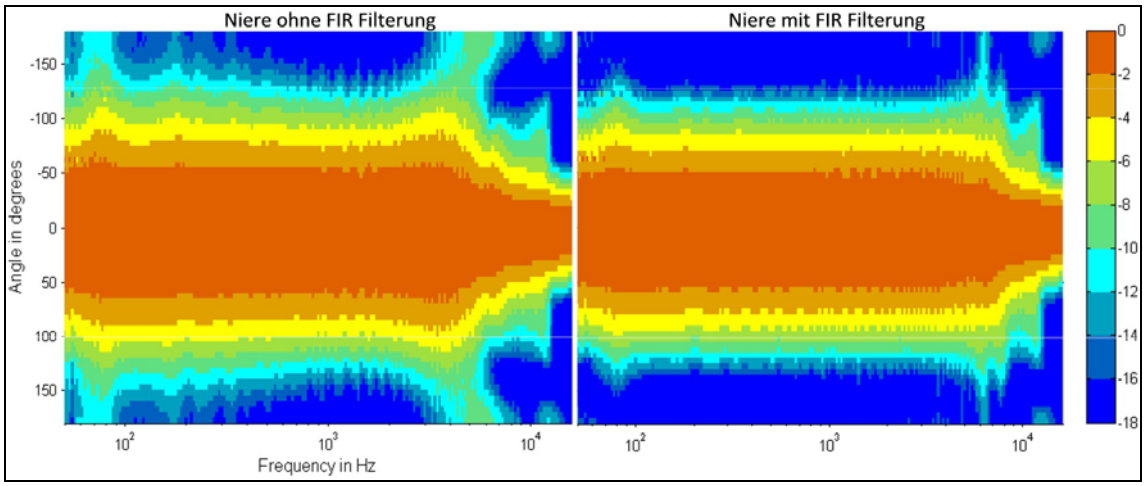

Abbildung 2.8: 2D Abbildung der Richtcharakteristik einer Niere ohne FIR-Filterung und mit FIR-Filterung (Microtech Gefell)

Eine weitere sehr interessante Möglichkeit wäre die Einstellung der Richtcharakteristik in Abhängigkeit der Frequenz durch das Plug-In zu realisieren. So könnte man z.B.

verschiedene Richtcharakteristiken für bestimmte Frequenzen einstellen, wie Abbildung 2.9 zeigt.

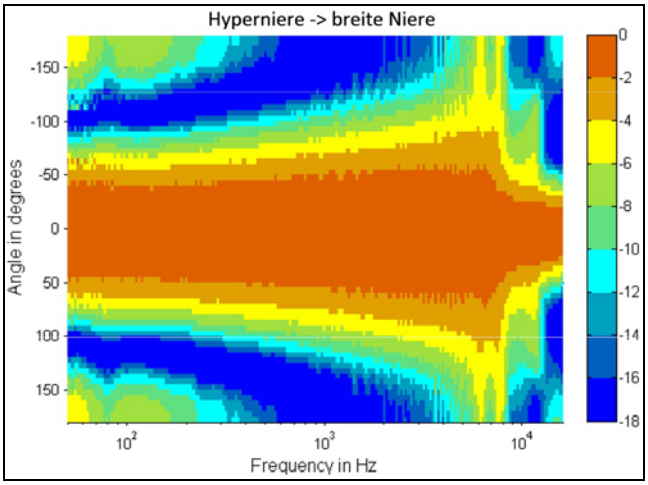

Abbildung 2.9: : 2D Abbildung eines Übergangs der Richtcharakteristiken Hyperniere zur breiten Niere mit FIR-Filterung (Microtech Gefell)

Da heutzutage nahezu alle professionellen Aufnahmen sowohl im Tonstudio als auch außerhalb mit dem Computer gemacht werden, hat man durch ein Software-Plug-In die Chance, einem Tonmeister einen gesamten Werkzeugsatz zum erworbenen Twin-Mikrofon mitzugeben, ohne dass dieser ein externes Gerät benötigt.

3 Implementierung eines VST Plug-Ins für ein Twin-Mikrofon

3.1 Ziele

Das Ziel dieser Arbeit ist die Entwicklung eines VST Plug-Ins für ein Twin-Mikrofon zur Synthese frequenzabhängiger Richtcharakteristiken bei der Postproduction. Es soll also während der Postproduction einer Aufnahme möglich sein, die Richtcharakteristik des verwendeten Mikrofons einzustellen. So muss sich der Tonmeister bei Außenaufnahmen (z.B. Konzertmitschnitt) nicht schon im Vorfeld für eine spezielle Richtcharakteristik entscheiden, sondern kann dies in der gewohnten Atmosphäre des Tonstudios während der Nachbearbeitung bzw. während des Mischprozesses tun. Die Entscheidung für oder gegen eine bestimmte Richtcharakteristik könnte also in den Prozess der Postproduction eingegliedert werden und muss nicht schon im Planungsprozess erfolgen.
Großmembranmikrofone mit umschaltbarer Charakteristik finden in Tonstudios sehr häufig Verwendung. Durch das Plug-In in Verwendung mit dem Twin-Mikrofon kann die Charakteristik am Aufnahmerechner einfach variiert werden und es bedarf keiner Umstellung mehr am Mikrofon selbst. So muss der Tonmeister nicht den Aufnahmeraum betreten, um die Richtcharakteristik zu verändern, sondern kann dies einfach vom Platz aus entscheiden, ohne dass die Musiker etwas davon mitbekommen. Dies erleichtert die Arbeit des Tonmeisters enorm. Außerdem wird die intime Athmosphäre der Musiker im Aufnahmeraum nicht gestört.

Eine weitere Chance des während dieser Arbeit entwickelten Plug-Ins soll sein, dass die Richtcharakteristik nicht nur für den gesamten Frequenzbereich konstant verändert werden kann, sondern dass die Richtcharakteristik in Abhängigkeit der Frequenz wählbar ist. So sollte es z.B. möglich sein, dass man als Richtcharakteristik eine Kugel verwendet, aber für eine bestimmte Resonanzfrequenz des Raumes die Richtcharakteristik einengt. Durch die Synthese frequenzabhängiger Richtcharakteristiken ist man somit noch flexibler und es ergeben sich gar neue Chancen und Möglichkeiten der Einflussnahme auf ein aufgenommenes Signal.

Das Mikrofon, für das in dieser Arbeit ein Plug-In entwickelt wird, ist das *„UM 930 Twin“* des Unternehmens Microtech Gefell.

Abbildung 3.1: Das Mikrofon "UM 930 Twin“ des Unternehmens Microtech Gefell (Microtech Gefell)

Hierbei handelt es sich um ein Kondensatormikrofon mit Großmembrandualkapsel. Das Mikrofon gibt zwei verschiedene Signale über je ein Kabel aus. Ein Signal ist grundsätzlich die vordere Niere und das zweite Signal ist durch klassische Umschalttechnik frei wählbar. Der Umschaltring muss, um das Plug-In später nutzen zu können, auf Niere gestellt sein. Denn, um die Richtcharakteristik bei der Postproduction frei wählen zu können, muss man die beiden Nierensignale (Niere vorne, Niere hinten) aufzeichnen, sodass beim Mikrofon die beiden Nierensymbole übereinander stehen (siehe Abbildung 3.2). Im Folgenden wird deshalb nur noch von zwei Nierensignalen gesprochen.

Da man diese zwei Signale der vorderen und hinteren Niere benötigt, um mithilfe des Plug-Ins eine neue Richtcharakteristik zu generieren, braucht man also in der Sequenzer-Software eine Spur mit zwei Kanälen. In diesem Zusammenhang bietet sich

eine Stereogruppe an. Es sollte immer gewährleistet sein, dass sich die vordere Niere auf dem linken Kanal der Stereospur und die hintere Niere auf dem rechten Kanal der Stereospur befinden.

Abbildung 3.2: Stellung des Schaltringes, um die vordere und die hintere Niere aufzuzeichnen, sowie die Beschriftung der Kabel (Microtech Gefell)

Hierzu sollte man die aufgezeichneten Monosignale jeweils nach links (vordere Niere) und nach rechts (hintere Niere) pannen und auf eine Stereogruppe routen. Auf dieser Gruppe sollte dann das Plug-In aktiv werden und die gewünschte Richtcharakteristik in Abhängigkeit von der Frequenz wiedergeben. Falls man direkt in einer Stereospur aufgenommen hat, sollte das Plug-In natürlich auch auf dieser funktionieren, sofern sich die Signale auf der richtigen Kanalseite befinden.

Um dieses Plug-In zu implementieren, bedarf es einer strukturellen Aufteilung in drei wichtige Abschnitte. Die Benutzeroberfläche, die Berechnung der Filterkoeffizienten und schließlich die Programmierung des Faltungsalgorithmus werden zu den drei wichtigsten Meilensteinen dieser Arbeit.
Die Benutzeroberfläche soll möglichst intuitiv und einfach zu bedienen sein. Es sollen keine weiteren Erläuterungen nötig sein. Ein Tonmeister, der das Mikrofon verwendet hat, sollte beim ersten Blick die Möglichkeiten des Programms und die Bedienung der Oberfläche erkennen. So sollten die zwei Funktionen: konstante sowie frequenzabhängige Synthese der Richtcharakteristik sowie ihre Bedienung sogleich sichtbar und intuitiv durchführbar sein.

Aus diesen visuellen Einstellungen der Benutzeroberfläche sollen dann die Eingabewerte über den gesamten Frequenzbereich generiert werden. Aus diesen gewonnenen Werten sollen durch Funktionen und verschiedene mathematische Berechnungen nun die entscheidenden Filterkoeffizienten, die der Faltungsalgorithmus benötigt, berechnet werden.

Nachdem die Filterkoeffizienten berechnet worden sind, müssen diese noch auf die eingehenden Audiosignale angewandt werden. So wird also aus den zwei Eingangssignalen und den berechneten Filterkoeffizienten ein neues Audiosignal mithilfe eines Faltungsalgorithmus erstellt.

Das Ziel dieser Arbeit ist es, diese mathematischen Berechnungen in Echtzeit durchzuführen, sodass man die Beeinflussung des Audiosignals direkt akustisch wahrnehmen kann. So kann der Tonmeister direkt die Auswirkungen seiner Einstellungen hören und entscheiden, welche Einstellung für seinen Anwendungszweck die perfekte ist.

In den nun folgenden Kapiteln wird die Umsetzung dieser drei wichtigen Schritte detailliert erläutert. Begonnen bei der Benutzeroberfläche über die Berechnung der Filterkoeffizienten bis zum Faltungsalgorithmus kann man in den Kapiteln die Funktionsweise des Plug-Ins von der Benutzereingabe bis zur Signalverarbeitung verfolgen.

3.2 Gestaltung und Programmierung der grafischen Benutzeroberfläche

3.2.1 Visuelle Herangehensweise

Zu Beginn der Gestaltung der Benutzeroberfläche wurde ein grundlegendes Konzept erstellt, wie die Oberfläche strukturiert sein sollte und welche Erwartungen diese erfüllen sollte. Die grundlegende Idee der Gestaltung der Benutzeroberfläche war, diese visuell möglichst überschaubar und intuitiv zu gestalten. Ein Tonmeister, dem die Funktionen und Möglichkeiten des Mikrofons *„UM 930 Twin“* bekannt sind, sollte sich in der Oberfläche des Plug-Ins direkt zurechtfinden und schon nach wenigen Minuten wissen, wie er seine Anforderungen mithilfe des Plug-Ins umsetzen kann.

Zur Umsetzung der grafischen Gestaltung und der Steuerelemente wurde *VST GUI 4* verwendet. *VST GUI 4* ist ein Satz verschiedener Klassen bzw. Steuerelemente für die

Programmierung von Benutzeroberflächen von Audio Plug-Ins wie VST oder Audio Unit, insbesondere aber natürlich für VST.

Die grafische Aufteilung des Plug-Ins teilt sich im Prinzip in drei Teile, denn in der Oberfläche sollen die zwei Funktionen – konstante sowie frequenzabhängige Synthese der Richtcharakteristik – des Plug-Ins, auf den ersten Blick getrennt wahrnehmbar sein. Außerdem gibt es in der Sequenzer-Software noch eine Art Kopfzeile, in der man das Plug-In an- und ausschalten kann, das Plug-In auf Bypass stellen kann und Presets speichern und laden kann. Diese Zeile ist visuell schon vorhanden, allerdings müssen die Funktionen bezüglich des Bypass-Parameters bzw. der Presets im Hintergrund implementiert werden, damit sie nutzbar sind.

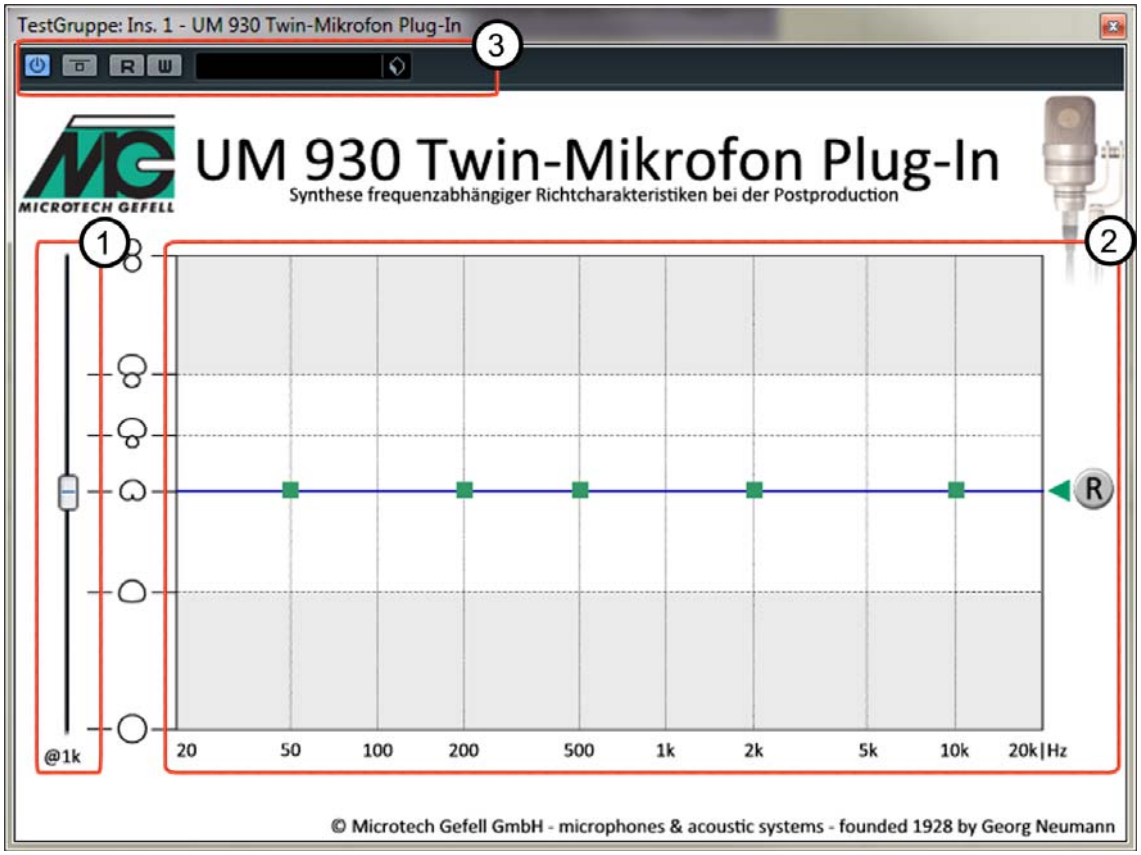

Abbildung 3.3: Visuelle Darstellung der drei Teile der Benutzeroberfläche: Konstante (1) sowie frequenzabhängige (2) Einstellung der Richtcharakteristik und die Kopfzeile der Sequenzer-Software (3)

Für die zwei Varianten – konstante sowie frequenzabhängige Synthese der Richtcharakteristik – des Plug-Ins wurden ein Schieberegler sowie ein Koordinatensystem verwendet. Die zwei Bedienelemente stehen direkt nebeneinander und somit visuell in Zusammenhang, aber sind je nach Anforderung des Tonmeisters an das Plug-In auch getrennt wahrnehmbar. Schon in Abbildung 3.3 wird deutlich, dass man sich nicht für eine der zwei Varianten entscheiden muss, sondern immer beide Varianten möglich sind. Der Schieberegler fungiert in der Benutzeroberfläche nur als Fernbedienung für alle Punkte im Koordinatensystem, um dem Tonmeister eine konstante Einstellung der Richtcharakteristik über alle Frequenzen zu vereinfachen. Daraus ergibt sich: Nur die Punkte und die Kurve im Koordinatensystem sind relevant für die im Hintergrund durchgeführten Berechnungen.

Auf die genaue Programmierung des Schiebereglers und des Koordinatensystems wird in den folgenden Kapiteln eingegangen. Da sich während der Programmierung noch interessante Möglichkeiten im Bezug auf das Zusammenspiel der beiden Elemente in der Oberfläche ergaben, finden sich dazu noch weitere Ausführungen, bevor dann auf die Kopfzeile der Sequenzer-Software eingegangen wird.

3.2.2 Schieberegler für konstante Synthese der Richtcharakteristik

Für die konstante Einstellung der Richtcharakteristik wurde ein Schieberegler (VST: Slider) verwendet, der in der *VST GUI 4* auch vorgefertigt als Steuerelement zu finden war. Neben dem Schieberegler finden sich die Symbole der verschiedenen Richtcharakteristiken (Acht, Hyperniere, Superniere, Niere, breite Niere, Kugel). Der mathematische Bezug zur Richtcharakteristik bzw. die mathematische Einheit des Schiebereglers ist der relative Empfang bei 90 Grad der jeweiligen Richtcharakteristik. Da der Regler Werte von 0 bis 1 zurück gibt, wird hier nicht der relative Empfang in dB angeben, sondern der relative Empfang bei 90 Grad als prozentualer Wert von 0 bis 1. Der relative Empfang bei 90 Grad ist z.B. 1 bei der Kugel und 0 bei der Acht. Alle weiteren Angaben sind der folgenden Tabelle zu entnehmen.

Richtcharakteristik	**Symbol**	**Relativer Empfang bei 90°** *(Richtungsmaß in dB)*	**Relativer Empfang bei 90°** *(Richtungsfaktor von 0 bis 1)*
Acht		- ∞	0
Hyperniere		- 12	0,37
Superniere		- 8,6	0,25
Niere		- 6	0,50
Breite Niere		- 3	0,71
Kugel		0	1

Tabelle 3.1: Symbole der Richtcharakteristiken in der Benutzeroberfläche mit den zugehörigen Einheiten[15]

Der Schieberegler befindet sich direkt links neben dem Koordinatensystem und teilt sich mit diesem auch die Beschriftung der y-Achse. Der Schieberegler ermöglicht es, alle Punkte im Koordinatensystem gleichzeitig zu verschieben. So ist es möglich nach dem Öffnen des Plug-Ins alle Punkte konstant zu verschieben und somit eine konstante Richtcharakteristik über alle Frequenzen zu erzeugen. Außerdem kann man den Regler so auch als Voreinstellung benutzen, bevor man Details in Abhängigkeit der Frequenz festlegt. Alle weiteren Informationen zum Zusammenspiel zwischen dem Regler und dem Koordinatensystem finden sich in Kapitel 3.2.4.

[15] Vgl. Dickreiter, Michael, et al.: Handbuch der Tonstudiotechnik (7. Auflage), München 2008, S. 123 ff.

3.2.3 Koordinatensystem für frequenzabhängige Synthese der Richtcharakteristik

Für die Einstellung der Richtcharakteristik in Abhängigkeit der Frequenz konnte kein Steuerelement gefunden werden, denn hierfür wird ein Koordinatensystem benötigt, welches nicht vorgefertigt in der *VST GUI 4* vorhanden ist. Somit wurde ein neues Steuerelement in der Klasse *CRCPoints* programmiert. Dieses sollte es ermöglichen, mithilfe von bewegbaren Punkten einen flexiblen Graf zu erstellen, der die Richtcharakteristik in Abhängigkeit der Frequenz angibt.

An der x-Achse des Koordinatensystems befinden sich die Frequenzen von 20 bis 20000 Hz in gewohnt logarithmischer Darstellung, wie es ein Tonmeister von einem klassischen Equalizer Plug-In gewohnt ist. An der y-Achse befinden sich die Symbole der verschiedenen Richtcharakteristiken (Acht, Hyperniere, Superniere, Niere, breite Niere, Kugel), damit sofort intuitiv erkennbar ist, was am Koordinatensystem bzw. beim Schieberegler eingestellt wird. Wenn die Einheit der x-Achse also in Hz (20 - 20000) angegeben ist, so ist die Einheit der y-Achse sowie des Schiebereglers der relative Empfang der jeweiligen Richtcharakteristik bei 90 Grad, der hier als prozentualer Wert von 0 bis 1 dargestellt wird (siehe Tabelle 3.1).

Das Zusammenwirken bzw. die Darstellung der Punkte in Bezug auf die erzeugte Kurve ist nicht verwandt mit der Darstellung der Punkte und deren Auswirkungen auf die Kurve eines Equalizer Plug-Ins, da hier nicht mehrere Parameter für einen Punkt zur Verfügung stehen.

In dem in dieser Arbeit entwickelten Plug-In steht die visuelle Bearbeitung der Punkte im Koordinatensystem im Vordergrund. So ist die Kurve von allen fünf Punkten bzw. deren Position abhängig und ist somit eher mit der Bearbeitung einer Lautstärkekurve verwandt als mit der Bearbeitung der Punkte einer Equalizerkurve.

In diesem Zusammenhang ist die Anzahl der Punkte sehr wichtig. Nach anfänglichen Überlegungen fiel die Entscheidung auf fünf Punkte, denn mit fünf Punkten besteht die Möglichkeit zwei Funktionen des Plug-Ins zu kombinieren. Auf der einen Seite hat man die Chance einen weichen Übergang zweier Richtcharakteristiken zu erzeugen. Auf der anderen Seite kann man nun zusätzlich noch die Richtcharakteristik für eine bestimmte

Frequenz einengen. So hat man z.B. bei einer Orchesteraufnahme die Möglichkeit, der Aufnahme durch eine Kugel in den tiefen Frequenzen eine gute Räumlichkeit zu verleihen und gleichzeitig durch einen Übergang zu einer Niere trotzdem die Ortbarkeit der einzelnen Instrumente möglich zu machen. Hierbei bietet sich ein Übergang zur Niere bei ca. 500 - 800 Hz an, da zwischen 500 und 2000 Hz im Bereich maximaler Gehörempfindlichkeit die Ortungsschärfe besonders gut ist.[16] Gleichzeitig könnte die Richtcharakteristik für eine bestimmte negativ auffallende Resonanzfrequenz des Raumes noch weiter eingeengt werden. Fünf Punkte in dem Koordinatensystem sollten niemanden visuell überfordern und ermöglichen gleichzeitig eine sehr individuelle Bedienung mit vielen Chancen und Möglichkeiten.

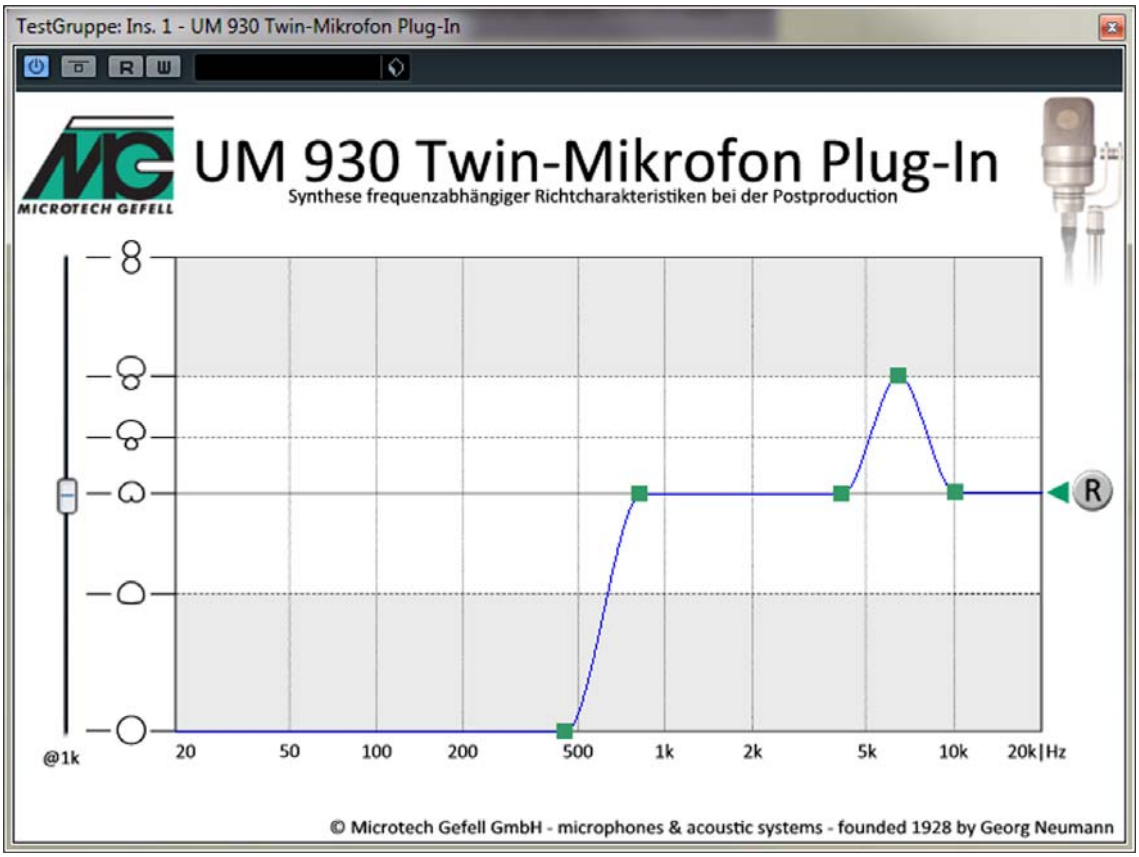

Abbildung 3.4: Darstellung einer möglichen Kurve in der Benutzeroberfläche mit einem weichen Übergang von Kugel zur Niere und Einengung einer bestimmten Resonanzfrequenz

[16] Vgl. Görne, Thomas: Mikrofone in Theorie und Praxis (8. Auflage), Aachen 2007, S. 28.

Eine weitere wichtige Entscheidung im Hinblick auf die Position der Punkte ist, wie frei der Nutzer diese bewegen können soll bzw. wie viel Spielraum er haben darf, damit die Kurve, die durch die fünf Punkte gezeichnet wird, einen Sinn ergibt. Jeder Punkt kann einfach durch einen Mausklick und die folgende Bewegung (*onMousClick()* und *onMousMove()*-Methoden) im Koordinatensystem bewegt werden. Als erstes ist es elementar wichtig, dass er die Punkte nur mit der Maus bewegen kann, solange er sich mit dieser im Koordinatensystem befindet. Sobald er das Koordinatensystem mit dem Mauszeiger also verlässt, verliert er auch den Zugriff auf den Punkt. Zwei weitere wichtige Kriterien sind, dass die Punkte nicht aneinander vorbeibewegt werden können und nicht zu dicht aneinander liegen. Somit kann man einen Punkt nur auf 35 Pixel an seine Nachbarpunkte heran bewegen. Ein weiteres wichtiges Kriterium ist, dass man den letzten Punkt nicht über 10 kHz bewegen kann. Dies hat Gründe, die in der Generierung der Filterkoeffizienten für die Faltung zu finden sind und somit genauer in Kapitel 3.3 erläutert werden.

Nachdem die grundlegende Funktion des Koordinatensystems und die Möglichkeiten, wie man die Punkte verschieben und positionieren kann, nun erklärt sind, fehlt nun noch die grafische Berechnung der Kurve, die durch die Punkte verlaufen soll. Diese ist aus zwei Gründen sehr wichtig. Erstens repräsentiert die Kurve visuell exakt das, was das Plug-In umsetzen soll, und ist somit für den Nutzer der wichtigste Bestandteil der Oberfläche. Zweitens wird genau diese mathematische Grundlage der Graphenberechnung verwendet, um im Hintergrund die Filterkoeffizienten zu berechnen. Somit ist die Kurve bzw. deren mathematische Grundlage nicht nur das Herzstück der Benutzeroberfläche sondern gleichzeitig auch die Schnittstelle zwischen *Controller* und *Processor* in der VST-Programmierung.
Bei der Berechnung der Kurve unterscheiden sich zwei Bereiche. Vom linken und rechten Rand des Koordinatensystems in den ersten bzw. letzten Punkt werden einfache vertikale Geraden gezeichnet. Der interessante und wichtige Teil der Kurvenberechnung spielt sich zwischen den Punkten ab, da hier weiche Kurven gezeichnet und berechnet werden sollen. In dieser Arbeit wurden zwei unterschiedliche Möglichkeiten für die Kurvenberechnung programmiert und gegenübergestellt. Beide Varianten bringen Vor- und Nachteile mit sich. Bei beiden

Möglichkeiten ergeben sich vier Teilfunktionen zwischen den fünf Punkten, die berechnet werden müssen. Im Folgenden wird auf die mathematischen Hintergründe der beiden Varianten eingegangen.

Die erste Möglichkeit der Berechnung der Kurve ist die Darstellung mit cubischen Splinefunktionen[17]. In diesem Fall sind die fünf Punkte durch vier Polynome dritten Gerades verbunden, wobei die Kurven, die in den Punkten aufeinander treffen, in den Punkten nicht nur den gleichen Funktionswert haben, sondern auch in der ersten und zweiten Ableitung übereinstimmen. Dadurch entsteht eine „glatte Kurve“. Für den ersten und letzten Punkt ist die Steigung auf Null gesetzt, damit diese mit der Steigung der Anfangs- bzw. Endgeraden übereinstimmt. Daraus ergibt sich auch dort ein weicher Übergang. Die Berechnung der Kurve erfolgte durch einen Algorithmus der Spline-Interpolation[18]. Ein weiterer wichtiger Aspekt ist, dass die Kurve ausbrechen kann. Deshalb muss die Kurve am oberen und unteren Rand des Koordinatensystems abgefangen werden. Sollte die Kurve außerhalb des Koordinatensystems verlaufen, werden für diesen Bereich ihre Funktionswerte entlang einer Geraden am Rand des Koordinatensystems gesetzt.

Der Vorteil dieser Variante ist, dass die Kurve sehr organisch wirkt, denn mit jeder Aktion reagiert die Kurve und bleibt immer eine „glatte Kurve“. Der Nachteil dieser Variante ist, dass die Punkte nicht zwingend Extrem- bzw. Sattelpunkte sind und somit der höchste und niedrigste Punkt nicht zwingend in den Extrempunkten der Kurve liegen. Somit sind die Punkte nur Werkzeuge, um die Kurve zu erzeugen, und die Position eines Punktes sagt nicht konkret etwas aus, wie man es von einem Equalizer gewohnt ist, bei dem ein Punkt immer für eine angegebene Frequenz Extrempunkt ist.

[17] Weitere Informationen, die als Grundlage für diese Arbeit dienten, finden sich in: Stoer, Josef: Numerische Mathematik 1 (9. Auflage), Heidelberg 2005, S. 104.

[18] Weitere Informationen, die als Grundlage für diese Arbeit dienten, finden sich in: Stoer, Josef: Numerische Mathematik 1 (9. Auflage), Heidelberg 2005, S. 108.

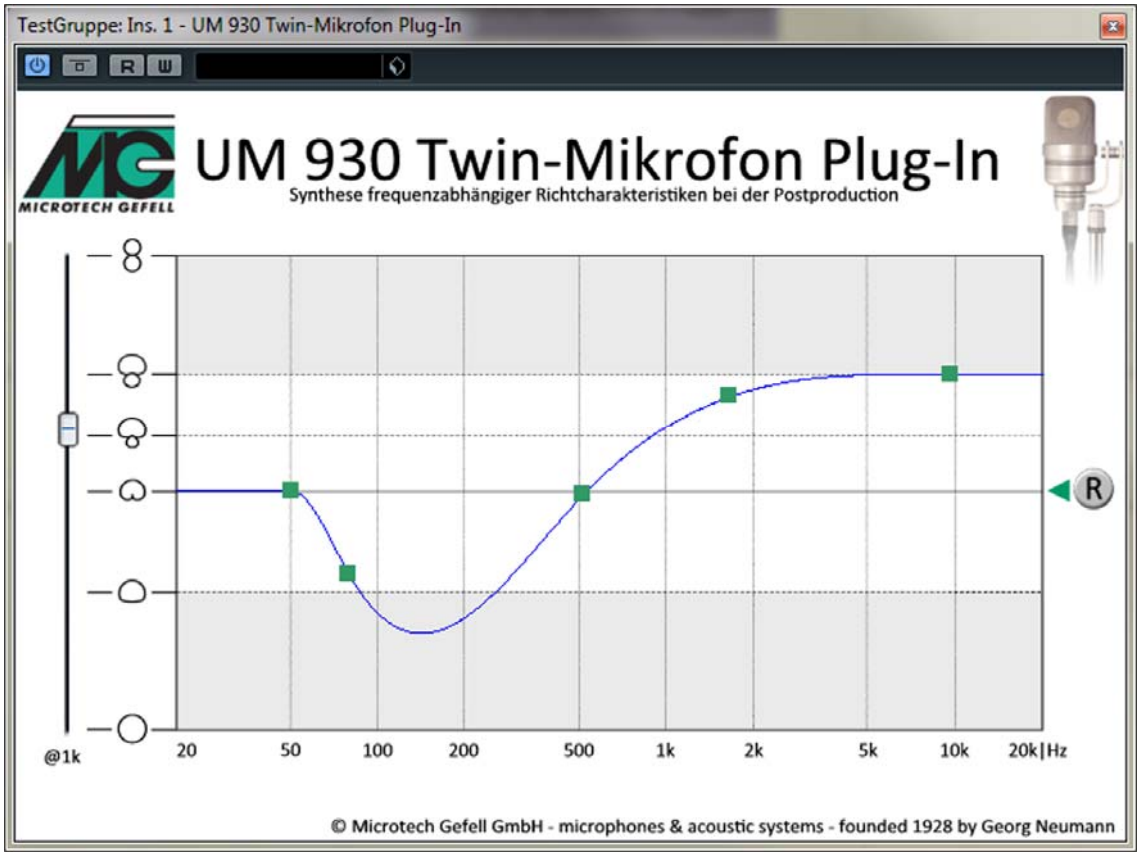

Abbildung 3.5: Darstellung einer Kurve mit cubischen Splinefunktionen in der Benutzeroberfläche

Bei der zweiten Möglichkeit der Berechnung der Kurve wird dafür gesorgt, dass die Steigung in den Punkten Null ist. Dadurch wird erreicht, dass die Punkte in den Extrempunkten der Kurve liegen oder als Sattelpunkt dienen. Somit werden der höchste und niedrigste Punkt zu Maxima bzw. Minima. Auch hier handelt es sich um Funktionen dritten Grades, die allein dadurch bestimmt werden, dass die Funktionswerte am Anfang und am Ende gegeben sind sowie die Anfangs- und Endsteigung Null ist. Aus diesem Grund lassen sich diese Funktionen einfach bestimmen. Allerdings sind die Kurven nicht mehr „glatt“, weil sie in den Punkten in der zweiten Ableitung nicht mehr übereinstimmen.

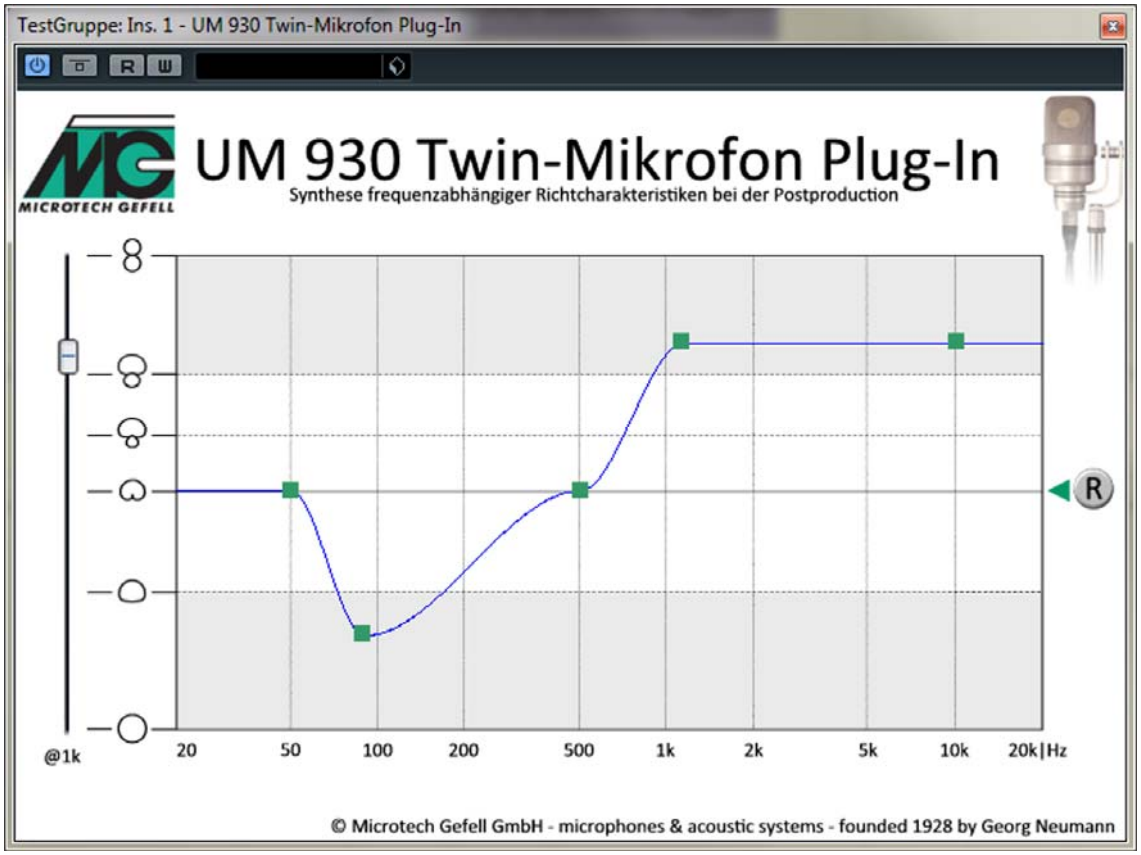

Abbildung 3.6: Darstellung einer Kurve mit einer Funktion dritten Gerades in der Benutzeroberfläche

Bei der zweiten Variante wirkt die Kurve nicht mehr so organisch und variabel, da sich die einzelnen Funktionen zwischen den Punkten nicht mehr beeinflussen. Allerdings sind die Punkte nun aussagekräftiger, da diese nun immer Extrem- oder Sattelpunkte sind, und so müsste die Bedienung dem Tonmeister vertrauter vorkommen.

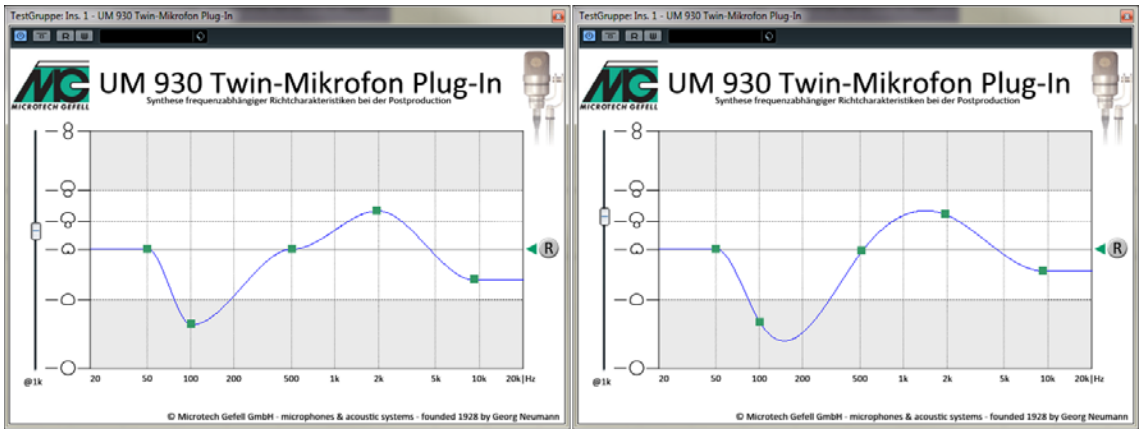

Abbildung 3.7: Vergleich zweier ähnlicher Kurven mit den verschiedenen Darstellungsformen in der Benutzeroberfläche

Nach einiger Recherche und Überlegung wurde sich für die zweite Variante entschieden, da sich dadurch ein Tonmeister direkt in seinem Element befinden und sofort mit der Kurve umgehen können sollte. Ihm sollte die Bedienung der Punkte schon aus anderen Umfeldern bekannt sein und er kann somit sofort intuitiv mit der Arbeit beginnen. Bei der ersten Variante müsste vermutlich nach Anfangsschwierigkeiten erst eine neue Herangehensweise der Bedienung verstanden werden. Sollte dies auch nur wenige Minuten in Anspruch nehmen, so würde es dem Plug-In schon im ersten wichtigen Eindruck Attraktivität rauben.

3.2.4 Zusammenwirken von Schieberegler und Koordinatensystem

Während der Entwicklung des Plug-Ins rückte das Zusammenspiel zwischen Schieberegler und Koordinatensystem immer weiter in den Vordergrund. Wenn man nach dem Öffnen des Plug-Ins mit dem Schieberegler die Punkte zuerst konstant verschieben kann, warum sollte man den Schieberegler im späteren Arbeitsprozess nicht mehr verwenden können. Denn auch wenn man die Kurve schon individuell gestaltet hat und diese für die speziellen Anforderungen gut geeignet ist, kann immer noch die Situation eintreten, dass man den Verlauf zwar beibehalten möchte, aber die gesamte Kurve nach oben oder unten verschieben möchte.

Wenn der Schieberegler die Kurve im späteren Arbeitsprozess noch editieren soll, brauchen die Kurve im Koordinatensystem und der Regler eine Art Synchronisation bzw. einen Synchronisationspunkt. Ansonsten würde z.B. die Möglichkeit bestehen, dass man den Schieberegler zu Beginn auf „Acht“ stellt und die Punkte aber im Laufe des Arbeitsprozesses im unteren Drittel des Koordinatensystems anordnet. Wenn man diese nun alle gleichmäßig nach oben schieben will, ist das nun nicht mehr möglich, da sich der Regler schon an seiner oberen Grenze befindet.

Aus diesem Grund wurde also entschieden, dass, wenn sich die Kurve bewegt, der Schieberegler ebenfalls mitverschoben wird. Anders herum war dies ohnehin schon der Fall. Nach längerem Überlegen mit welcher Position der Kurve man den Schieberegler synchronisiert, wurde festgelegt die Position der Kurve bei 1 kHz zu verwenden. Da die meisten Mikrofonhersteller die Richtcharakteristik für ihre Mikrofone bei 1 kHz angeben, ist dies ein logisch gewählter Wert, der den meisten

Nutzern sinnvoll erscheinen sollte. Wird nun die Kurve bewegt, wird der Schieberegler auf die Höhe gesetzt, die die Kurve an der Position der Frequenz 1 kHz hat.

An dieser Stelle soll noch auf einen weiteren Fall eingegangen werden. Nun ist es möglich die Punkte durch den Schieberegler an den oberen bzw. unteren Rand des Koordinatensystems zu bewegen. Sollte der Schieberegler allerdings noch weiter bewegt werden, bleiben die Punkte am Rand des Koordinatensystems hängen und die Kurve wird bei weiterer Bewegung in gleicher Richtung zusammengeschoben, da die Punkte das Koordinatensystem nicht verlassen können. Dies trifft sowohl auf den oberen als auch auf den unteren Rand zu.

Rechts neben dem Koordinatensystem befindet sich noch der Reset-Button auf den an dieser Stelle nur kurz eingegangen werden soll. Der Reset-Button hat sowohl auf die Punkte im Koordinatensystem als auch auf den Schieberegler Zugriff. Wird der Button betätigt, so werden Schieberegler und Punkte, wie der grüne Pfeil visuell andeutet, auf die vertikale Mitte gesetzt und der Tonmeister hat wieder eine Grundlage für eine neue Einstellung. Horizontal werden die Punkte nicht zurück gesetzt, sodass die gewählten Frequenzen erhalten bleiben. Die Position sowie die visuelle Darstellung des Buttons wurden so gewählt, dass die Funktion des Buttons möglichst leicht zu durchschauen ist.

3.2.5 Bypassparameter und Nutzung der Preset-Funktionen

Ein ebenfalls wichtiger Bestandteil der Benutzeroberfläche ist die Kopfzeile über dem eigentlichen Plug-In, die allerdings von der Sequenzer-Software abhängt und auch je nach Programm unterschiedlich aussehen kann.

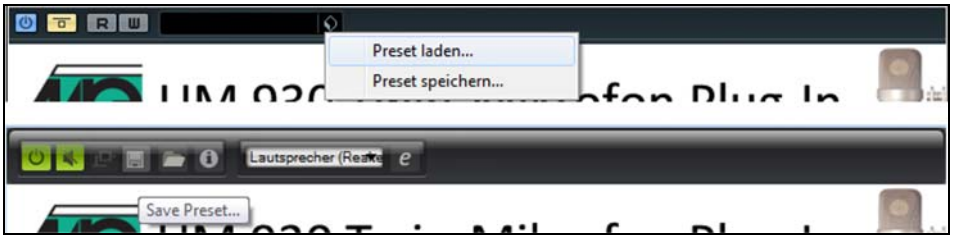

Abbildung 3.8: Kopfzeilen aus zwei verschiedenen Programmen: Steinberg Cubase 6 (oben) und der VST TestHost (unten)

Diese Kopfzeile wird grafisch nicht vom Entwickler programmiert bzw. gestaltet, aber deren Funktionen müssen im Hintergrund implementiert werden. Neben dem An/Aus Knopf, der grundsätzlich ohne Mitarbeit des Plug-In-Entwicklers funktioniert, finden sich dort noch der Bypass-Knopf sowie die Möglichkeit Presets für das Plug-In zu laden und zu speichern.
Die Aktivierung des Bypass-Knopfes ist im Prinzip relativ einfach. Für die Kommunikation zwischen *Controller* und *Processor* werden in der VST-Programmierung *Parameter* verwendet. In dem in dieser Arbeit entwickelten Plug-In werden z.B. *Parameter* für den Schieberegler und die x- und y-Positionen der Punkte verwendet. Um den Bypass-Knopf zu aktivieren, muss man lediglich einen weiteren *Parameter* erstellen und den *Flag* auf *„kIsBypass“* setzen.

Um dem Nutzer zu ermöglichen, Presets des Plug-Ins zu speichern und zu laden, bedarf es etwas mehr Arbeit. Ein Preset ist im Prinzip nicht mehr als eine Datei, in der die Werte der *Parameter* gespeichert werden. Die eben erwähnten *Parameter* müssen nun in die Preset-Datei gespeichert werden, um bei Aufruf des Presets aus dieser auch wieder geladen werden zu können. Hierfür ist es notwendig in der *Processor*-Klasse die Methoden *getState* und *setState* zu überschreiben. In diesen beiden Methoden werden dann die Parameter in die Datei geschrieben bzw. aus der Datei gelesen.
So werden die Daten in die Preset-Datei geschrieben und auf *Processor*-Seite auch wieder ausgelesen. Allerdings wird sich beim Laden eines Presets in der Oberfläche noch nichts regen. Hierfür gibt es in der *Controller*-Klasse noch die Methode *setComponentState*, die die Preset-Datei ausliest, die Parameter neu setzt und dadurch auch die Oberfläche aktualisiert.
Mit diesen Änderungen im Programmcode sind also auch die Bypass-Funktion sowie die Möglichkeiten Presets zu speichern und zu laden für den Tonmeister nutzbar.

3.3 Implementierung der mathematischen Grundlagen zur Erzeugung der Filterkoeffizienten für die Einstellung von frequenzabhängiger Richtcharakteristik

3.3.1 Berechnung der Filterkoeffizienten aus der Benutzeroberfläche über den gesamten Frequenzbereich

Nachdem der Tonmeister durch die grafische Benutzeroberfläche nun alle Möglichkeiten hat, die Kurve für seinen speziellen Zweck zu beeinflussen, ist nun die Weiterverarbeitung der visuellen Kurve von besonderer Bedeutung. Aus der Kurve müssen nun Funktionswerte generiert werden, die dann in weiteren mathematischen Funktionen und Berechnungen zu den Filterkoeffizienten für den Faltungsalgorithmus werden.

In diesem Kapitel soll nun zuerst auf die Berechnung der Übertragungsfunktion des FIR-Filters über den kompletten Frequenzbereich eingegangen werden. Die Berechnung der Übertragungsfunktion des FIR-Filters ist der erste Schritt, um zu den Koeffizienten für den Faltungsalgorithmus zu gelangen.

Das neue Ausgangssignal des Plug-Ins ergibt sich aus einer Addition des vorderen und hinteren Nierensignals, wobei das hintere Nierensignal mit den berechneten Koeffizienten vorher gefaltet werden muss. In diesem und den zwei folgenden Kapiteln kann man die Berechnung bis zu den finalen Faltungskoeffizienten verfolgen.

Für diese und auch die weiteren Berechnungen werden mathematische Funktionen bzw. Messwerte des Mikrofons *„UM 930 Twin“* benötigt, die als Daten-Dateien[19] vorliegen müssen. Diese Dateien werden vom Unternehmen Microtech Gefell mit dem Plug-In mitgeliefert. Der Ordner mit den Dateien muss sich im gleichen Ordner wie die „.exe-Datei“ der verwendeten Sequenzer-Software befinden.

Für die weiteren Berechnungen im Hintergrund ist die Kurve im Koordinatensystem und deren Berechnung von großer Bedeutung. Denn für die Kurve müssen nun die Funktionswerte berechnet werden. Es werden Funktionswerte für den

[19] Diese Daten-Dateien beziehen sich auf eine Samplingrate von 48 kHz. Somit ist das Plug-In in der ersten Version nur mit dieser Samplingrate einwandfrei lauffähig.

Frequenzbereich von 0 Hz bis 24 kHz berechnet. Die Anzahl der berechneten Funktionswerte müssen einer 2er-Potenz entsprechen, da dies nachher für die in Kapitel 3.3.3 beschriebene IFFT Vorraussetzung ist. Im Laufe der Entwicklung dieses Plug-Ins wurde sich für eine 2er-Potenz von 2^{13} entschieden. Da der erste Wert bei 0 Hz ist, aber der Wert bei 24 kHz ebenfalls Bestandteil der Berechnung sein soll, wird im weiteren Verlauf immer von 2^{13}+1 Werten gesprochen.
Im Folgenden soll die komplexe Formel, die für die Berechnung der Übertragungsfunktion des FIR-Filters (H_r) verwendet wurde, genauer erläutert werden.

$$H_r(f) = \frac{\Gamma(\Theta = 90°, f) \cdot S_f(\Theta = 0°, f) - S_f(\Theta = 90°, f)}{S_r(\Theta = 90°, f) - \Gamma(\Theta = 90°, f) \cdot S_r(\Theta = 0°, f)}$$

Formel 3.1: FIR-Übertragungsfunktion zur Beeinflussung der Richtcharakteristik (Domke, 2010)

Bevor auf die Herleitung der Formel sowie ihre detaillierte Erläuterung eingegangen wird, ist im Folgenden die Bedeutung der verwendeten Formelzeichen aufgelistet.

$H_r(f)$	Übertragungsfunktion des FIR-Filters
$\Gamma(\Theta, f)$	Richtungsfaktor aus der Benutzeroberfläche
$S_f(\Theta, f)$	Übertragungsfunktion der vorderen Kapsel (*front*)
$S_r(\Theta, f)$	Übertragungsfunktion der hinteren Kapsel (*rear*)

Bei allen Werten in dieser Formel handelt es sich um komplexe Zahlen[20] und somit handelt es sich um eine komplexe Berechnung. Die Übertragungsfunktion des FIR-Filters über den Frequenzbereich von 0 Hz bis 24 kHz ist das Ergebnis dieser Formel, welches für die weiteren Berechnungen benötigt wird. Die Einstellungen des Tonmeisters aus der Benutzeroberfläche finden sich in dieser Formel im

[20] Die Darstellung einer Schwingung mit komplexen Zahlen spielt in dieser Arbeit eine wichtige Rolle (besonders in der in Kapitel 3.3.3 beschriebenen IFFT). Eine gute Erklärung hierzu findet sich in: Görne, Thomas: Tontechnik (2. Auflage), München 2008, S. 25. ff.

Richtungsfaktor wieder. Die Übertragungsfunktionen der vorderen und hinteren Kapsel bei 0 Grad und 90 Grad werden beim Start des Plug-Ins aus den schon erwähnten Dateien eingelesen.[21]

Nun soll noch auf die Herleitung der Formel eingegangen werden. Hierfür benötigen wir drei weitere Formeln, die im Folgenden beschrieben werden.

$$\Gamma(\Theta, f) \quad = \quad \frac{G_m(\Theta, f)}{G_m(\Theta = 0°, f)}$$

Formel 3.2: Berechnung des Richtungsfaktors für eine bestimmte Frequenz in Abhängigkeit der Übertragungsfunktion (G_m) des gesamten Mikrofons (Domke, 2010)

In dieser Formel ist zu sehen, dass man den Richtungsfaktor eines Mikrofons für eine bestimmte Frequenz aus der Übertragungsfunktion des gesamten Mikrofons (G_m) herleiten kann. Als nächstes wird die Übertragungsfunktion des gesamten Mikrofons genauer beschrieben.

$$G_m(\Theta = 90°, f) = S_f(\Theta = 90°, f) + S_r(\Theta = 90°, f) \cdot H_r(f)$$

$$G_m(\Theta = 0°, f) = S_f(\Theta = 0°, f) + S_r(\Theta = 0°, f) \cdot H_r(f)$$

Formel 3.3: Berechnung der Übertragungsfunktion des gesamten Mikrofons bei 0 Grad und 90 Grad (Domke, 2010)

Die zwei Formeln für 0 Grad bzw. 90 Grad sind im Prinzip sehr einfach zu erklären, da sie widerspiegeln, was das Plug-In im Endeffekt macht. Es addiert die hintere Niere (S_r), die vorher mit den errechneten FIR-Koeffizienten (H_r) gefaltet worden ist, mit der vorderen Niere (S_f). Daraus ergibt sich dann die neue Richtcharakteristik. Setzt man

[21] Durch das Einlesen von Messwerten der jeweiligen Mikrofonkapseln verliert der Begriff „matched Pair“ für das *„UM 930 Twin“* an Bedeutung, da die Berechnungen die Messwerte der einzelnen Kapseln einbeziehen.

diese beiden Formeln für 90 Grad und 0 Grad jetzt in die Formel 3.2 ein und formt um, so erhält man die eigentliche Formel 3.1, die im Plug-In Verwendung findet.

Die Formel 3.1 wurde in zwei Formeln umgewandelt, mit denen man den Real- und Imaginärteil direkt ausrechnen kann. Das hat den Vorteil, dass man ohne Hilfsklassen oder Umrechnungen[22] mit einfachen Rechenoperationen zum Ergebnis kommt.
Der Eingangswert, den der Benutzer am Bildschirm verändert, ist der Richtungsfaktor bei 90 Grad, der sich aus der Kurve in der Benutzeroberfläche ableitet. Das Ergebnis ist die Übertragungsfunktion des FIR-Filters über den gesamten Frequenzbereich. Das Ergebnis sind also $2^{13}+1$ komplexe Zahlen.

An dieser Stelle soll noch einmal auf die komplexen Zahlen eingegangen und ein Bezug zur Tontechnik hergestellt werden.
Komplexe Zahlen kann man als *„Real- und Imaginärteil“* oder als *„Betrag und Phase“* darstellen. Wichtig für die Erklärung der komplexen Ergebnisse der Übertragungsfunktion des FIR-Filters ist aber die Darstellung mit *„Betrag und Phase“*. Vereinfacht kann man sagen, dass der *Betrag* der komplexen Zahl die Amplitude der hinteren Niere und die *Phase* die relative Phasenlage des hinteren Kapselsignals für einen Wert der Übertragungsfunktion des FIR-Filters angibt. Während die *Phase* vereinfacht ausgedrückt entweder 0 oder $-\pi$ ist, wird der *Betrag* als Wert von 0 bis 1 dargestellt. Einer der komplexen Ergebniswerte bildet also die Stellung des Mischpultfaders aus Abbildung 2.6 ab und wird in Tabelle 3.2 noch einmal beziffert. In der in dieser Arbeit verwendeten Übertragungsfunktion des FIR-Filters kann die Phase natürlich auch Werte zwischen 0 und $-\pi$ annehmen.

[22] Hiermit sind Umrechnungen von Real- und Imaginärteil in Betrag und Phase und umgekehrt gemeint. Mehr dazu findet sich in Kapitel 3.3.2.

Richtcharakteristik	**Symbol**	**Stellung des Mischpultfaders** *(dB)*	*Betrag* (H_r)	**Phasen-drehung**	*Phase* (H_r)
Acht		0	1	ja	$-\pi$
Superniere		- 10	0,32	ja	$-\pi$
Niere		$-\infty$	0	nein	0
Breite Niere		- 10	0,32	nein	0
Kugel		0	1	nein	0

Tabelle 3.2: Bezug eines komplexen Wertes der Übertragungsfunktion des FIR-Filters auf den Mischpultfader für das hintere Kapselsignal

3.3.2 Umwandlung der berechneten Koeffizienten vom Frequenzbereich in den Zeitbereich (IFFT)

Nachdem durch das Postprocessing bei hohen und tiefen Frequenzen nun die Übertragungsfunktion des FIR-Filters perfektioniert und überarbeitet wurde, ist der nächste elementare Schritt die Umwandlung vom Frequenz- in den Zeitbereich der gesamten Übertragungsfunktion.
Will man ein Signal vom Zeitbereich in den Frequenzbereich umwandeln, verwendet man eine Fouriertransformation. Da in diesem Fall genau das Gegenteil erreicht werden soll, nämlich die Umformung in den Zeitbereich, benötigt man eine inverse Fouriertransformation. Da die diskrete Fouriertransformation (DFT) sowie die inverse diskrete Fouriertransformation (IDFT) sehr viele mathematische Berechnungen und somit Zeit benötigen, sind diese nicht prädestiniert für ein Echtzeit-Plug-In.

Die schnelle Fouriertransformation (FFT) und die inverse schnelle Fouriertransformation (IFFT[23]) schaffen es, eine Summe der Länge $N = 2^m$ durch Aufteilung in gerade und ungerade Summanden auf eine Summe der Länge N/2 zurückzuführen. Durch diese Rekursion benötigen sie nicht so viele Berechnungen wie die DFT oder die IDFT. Im Bereich von Echtzeit-Plug-Ins bzw. in allen Bereichen, wo es auf Schnelligkeit und somit die Anzahl der benötigten Berechnungen ankommt, setzt man deshalb auf die FFT sowie die IFFT. So schreibt Prof. Dr. Josef Stoer:

> *„Die direkte Auswertung aller N Summen erfordert $O(N^2)$ Multiplikationen, so daß sie für sehr großes N völlig ungeeignet ist. Es ist deshalb von größter praktischer Bedeutung, dass von Cooley und Tukey (1965) erstmals ein Verfahren gefunden wurde, dass zur Berechnung aller Summen lediglich O(N log N) Multiplikationen benötigt. Verfahren dieser Art sind unter dem Namen schnelle Fouriertransformationen und FFT-Verfahren bekannt.“* [24]

[23] Eine genaue Erläuterung zur FFT und IFFT findet sich in: Stoer, Josef: Numerische Mathematik 1 (9. Auflage), Heidelberg 2005, S. 88 ff.

[24] Stoer, Josef: Numerische Mathematik 1 (9. Auflage), Heidelberg 2005, S. 88.

Da sich die Anzahl der Eingangswerte für die in dieser Arbeit durchgeführte IFFT auf 2^{14} = 16384 belaufen und sich in späteren Versionen des Plug-Ins evtl. noch erhöhen werden, kann man hier von einem großen N sprechen. Somit wird in diesem Plug-In eine IFFT verwendet, um die berechneten Werte in den Zeitbereich zu wandeln.

Die Eingangswerte für die IFFT, die aus den Kapiteln 3.3.1 und 3.3.2 hervorgehen, sind komplexe Werte. Da das nun gewonnene Zeitsignal ein reelles Signal ist, wird nur der Realteil des komplexen Ergebnisses weiter berücksichtigt und der durch Rundungsfehler erstandene sehr geringe Imaginärteil wird vernachlässigt.
Dieser wird im Anschluss an die IFFT noch mit einer Funktion, die ebenfalls als Datei des Unternehmens Microtech Gefell vorliegt, multipliziert. Aus den nun errechneten Werten können nun die Koeffizienten für den Faltungsalgorithmus entnommen werden.
In der ersten Version des Plug-Ins wurde sich für eine Faltung mit 511 Koeffizienten entschieden. Die ersten 255 der 511 Koeffizienten für die Faltung sind die 255 letzten der Werte aus der IFFT. Die noch fehlenden 256 Koeffizienten sind die ersten 256 Werte des Ergebnisses der IFFT.
Die Multiplikation mit den Werten aus der oben erwähnten Datei nach der IFFT sorgt dafür, dass der Wert an der Stelle 256 den größten Einfluss hat und die Werte zum Rand hin ausgeblendet werden und somit weniger oder gar kein Einfluss haben. Mit den nun gewonnenen Koeffizienten wird der Faltungsalgorithmus durchgeführt, der im folgenden Kapitel beschrieben wird.

3.4 Implementierung der (Echtzeit)bearbeitung der Audiosignale mit den Filterkoeffizienten (Faltungsalgorithmus)

3.4.1 Einbettung der Faltung in das Plug-In

Bevor nun auf die Möglichkeiten und letztlich auf die Implementierung des Faltungsalgorithmus eingegangen wird, soll an dieser Stelle noch einmal deutlich gemacht werden, wo der Faltungsalgorithmus im Gesamtzusammenhang des Plug-Ins eingebettet wird.

Das Plug-In erzeugt also durch die FIR-Koeffizienten des Mikrofonherstellers Microtech Gefell und die zwei Audiosignale des Mikrofons *„UM 930 Twin“* und durch die Berechnung der Filterkoeffizienten und schließlich eines Faltungsalgorithmus im Postprocessing das neue Audiosignal. Das Plug-In berechnet also aus der Benutzeroberfläche die Filterkoeffizienten immer dann, wenn sich die Kurve in der Benutzeroberfläche verändert hat. Der Faltungsalgorithmus wird allerdings ständig durchgeführt, damit immer aus den zwei Eingangssignalen das neue Audiosignal errechnet und ausgegeben wird.

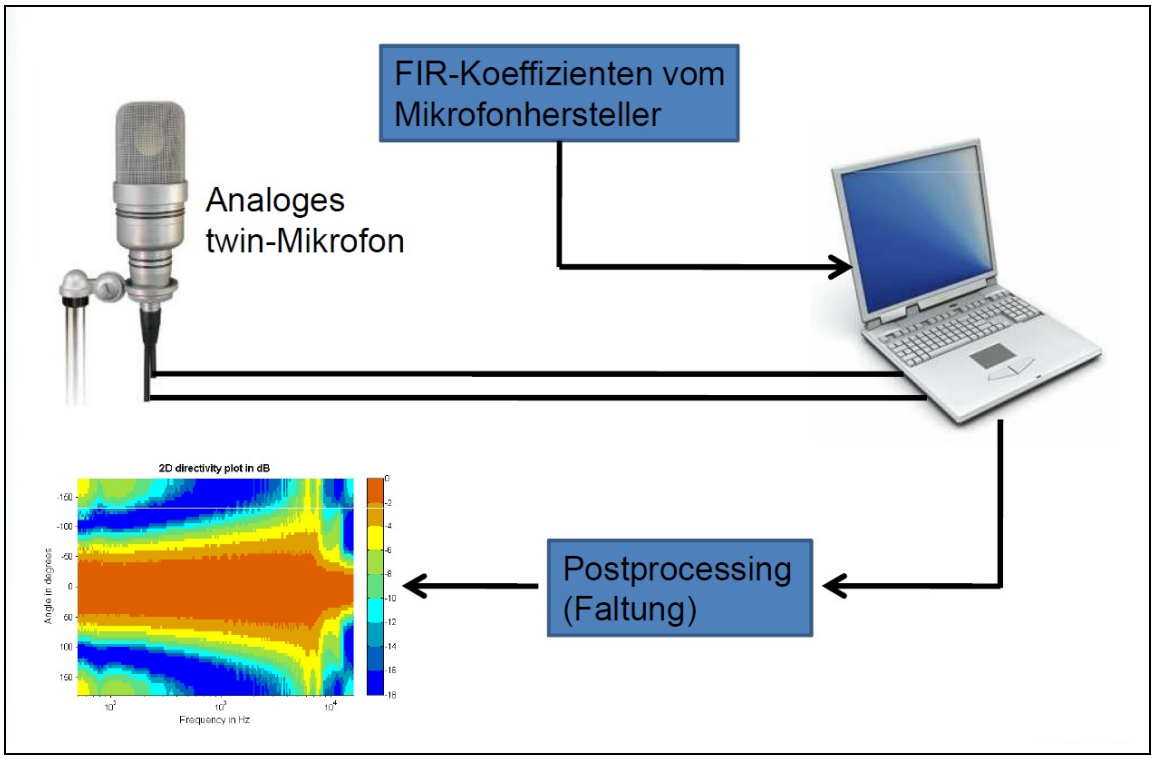

Abbildung 3.9: Arbeitsweise mit einem Twin-Mikrofon von der Aufnahme bis zur Synthese der Richtcharakteristik in Abhängigkeit der Frequenz in der Postproduction (Microtech Gefell)

Der wohl wichtigste Bestandteil bei der Implementierung eines VST Plug-Ins ist die *process*-Funktion. Diese findet sich in der Processor-Klasse. Die *process*-Funktion wird vom Framework in regelmäßigen Abständen aufgerufen und ist das Herzstück des Plug-Ins. Dieser Funktion werden Daten mitgegeben, die unter anderem Parameteränderungen und die zu bearbeitenden Samples der Eingangskanäle enthalten.

In diesem Plug-In unterteilt sich die *process*-Funktion in drei elementare Teile. Die Änderung der *Parameter*, die Berechnung der Filterkoeffizienten und schließlich die Faltung werden nacheinander abgearbeitet.
Zuerst wird geprüft, ob sich *Parameter* geändert haben. Für jede *Parameter*-Änderung wird die jeweilige Variable in der *Processor*-Klasse geändert. Sollte sich in diesem Teil mindestens eine *Parameter*-Änderung ereignet haben, müssen die Koeffizienten für die Faltung und somit alle in Kapitel 3.3 beschriebenen Berechnungen durchgeführt werden. Im zweiten Teil der Funktion werden also je nachdem, ob es eine Parameteränderung gab, die Koeffizienten neu bestimmt.
Der dritte und entscheidende Teil für das Ausgangssignal der Funktion ist der Faltungsalgorithmus, der in den nächsten beiden Kapiteln detaillierter erläutert wird. Grundsätzlich wird der Faltungsalgorithmus im Gegensatz zur Berechnung der Koeffizienten immer durchgeführt, aber auch hier gibt es Ausnahmen. Befindet sich das Plug-In im Bypass-Modus soll natürlich nicht gefaltet werden, sondern das Ausgangssignal gleich dem Eingangssignal sein. Ein weiterer wichtiger Faktor, damit das Plug-In fehlerfrei läuft, ist, dass es auf einer Stereospur liegt und die Spur somit genau zwei Eingangskanäle besitzt. Außerdem müssen die Dateien, die für die Berechnungen benötigt werden, korrekt auf der Festplatte abgelegt worden sein, sodass sie fehlerfrei in das Plug-In eingelesen werden konnten. Sollte sich das Plug-In also im Bypass-Modus befinden oder einer der beiden genannten Faktoren nicht eintreffen, wird keine Faltung vorgenommen. Stattdessen werden die Ausgangskanäle gleich der Eingangskanäle gesetzt.

Eine weitere Innovation in VST 3 sind die sogenannten *„SilenceFlags“*, die angeben, ob auf der Spur ein Eingangssignal anliegt. Wenn kein Eingangssignal auf einer Spur anliegt, arbeitet das Plug-In nicht und somit werden keine Ressourcen verbraucht. Das bedeutet, dass keinerlei Berechnungen durchgeführt werden und somit der Prozessor und die Soundkarte des Rechners entlastet werden. Es werden einfach die Ausgangskanäle gleich der Eingangskanäle gesetzt. Dies ist gerade bei modernen Musikproduktionen sehr sinnvoll, da hier häufig mit etwa 100 Spuren gearbeitet wird, aber auf diesen Spuren immer nur bruchstückhaft Audiomaterial liegt. Deshalb wurde

diese neue Möglichkeit in diesem Plug-In auch schon in der ersten Version implementiert.

3.4.2 Einfache Faltung und schnelle lineare Faltung

Im Folgenden soll auf die grundlegende Theorie der Faltung und die zwei für diese Arbeit möglichen Implementierungen – einfache Faltung und schnelle lineare Faltung – eingegangen werden.

In einem VST Plug-In wird die Faltung immer auf ein Sample angewandt. Wie in Kapitel 3.3.3 schon beschrieben, wird die Faltung mit 511 Koeffizienten durchgeführt, d.h. ein neues Sample ergibt sich aus 511 Koeffizienten und 511 Samples. Bei der Faltung in diesem Plug-In handelt es sich um eine Summe aus 511 Multiplikationen von Koeffizienten und Samples nach dem in Formel 3.4 beschriebenen Schema.

$$y(n) = \sum_{m=0}^{n} h(n-m)x(m)$$

Formel 3.4: Summenformel für die einfache Faltung (Ifeachor; Jervis, 1993)

Das in Formel 3.4 beschriebene Schema ist das Schema der einfachen Faltung. Dieses Schema bringt eine Latenz von 256 Samples mit sich, die in der Methode *„uint32 PLUGIN_API getlatencySamples()“* angegeben werden muss. Allerdings benötigt dieses Schema für eine sehr hohe Anzahl der Koeffizienten sehr viele Multiplikationen und sorgt somit für eine sehr hohe Auslastung des Prozessors und der Soundkarte des Computers. Für eine Länge der Summenformel von 511 würde dies noch zu keinem gravierenden Problem führen.

Die zweite Möglichkeit eine Faltung zu realisieren ist die schnelle lineare Faltung. Das Schema der schnellen linearen Faltung, was in Formel 3.5 zu sehen ist, erfordert Fouriertransformationen.

$$x_1(l) \otimes x_2(r) = F_D^{-1}[X_1(k)X_2(k)]$$

Formel 3.5: Formel für die schnelle lineare Faltung (Ifeachor; Jervis, 1993)

Die Formelzeichen $x_1(l)$ und $x_2(r)$ stehen für zwei periodische Abfolgen der Länge N. $X_1(k)$ ist die Fouriertransformierte von $x_1(l)$ und $X_2(k)$ ist die Fouriertransformierte von $x_1(r)$. F_D^{-1} steht für eine inverse Fouriertransformation. Daraus folgt, dass beide Abfolgen zuerst fouriertransformiert werden müssen, bevor sie dann multipliziert werden. Von diesem Ergebnis muss noch die inverse Fouriertransformation gebildet werden. Aus jeweils 511 Eingangswerten würden also auch 511 neue Samples berechnet. Deshalb muss vor der Berechnung auf 511 Samples gewartet werden und das Plug-In hätte somit eine Latenz von 511 Samples.
Nutzt man hierfür die Algorithmen der DFT und IDFT birgt das schon bei der relativ geringen Anzahl von 511 Faltungskoeffizienten das Problem, dass das Plug-In nicht mehr in Echtzeit lauffähig wäre. Die diskreten Fouriertransformationen sind dafür zu rechenintensiv. Nutzt man allerdings die Algorithmen der FFT und IFFT für die schnelle lineare Faltung, würde diese schon für ein N von 128 schneller als die einfache Faltung sein und bei einem N von 1024 wäre die schnelle lineare Faltung sogar zehn mal so schnell.[25]
In der ersten Version des Plug-Ins soll mit 511 Koeffizienten in die Faltung gegangen werden. Da dies allerdings keiner 2er-Potenz (2^9 = 512) entspricht, kann an dieser Stelle nicht mit den Algorithmen der schnellen Fouriertransformation gearbeitet werden.

Aus diesem Grund und wegen der geringeren Latenz bei der einfachen Faltung wird in der ersten Version des Plug-Ins die einfache Faltung implementiert. Weitere Informationen zur Performance – besonders im Hinblick auf zukünftige Versionen – sind im Kapitel 5.1 zu finden.
Im folgenden Kapitel soll nun die Signalverarbeitung im Detail erläutert werden. Außerdem soll die Implementierung des Algorithmus der einfachen Faltung beschrieben werden.

[25] Ifeachor, Emmanuel; Jervis, Barrier: Digital Signal Processing – A practical Approach, Great Britain 1993, S. 224.

3.4.3 Implementierung der einfachen Faltung

Bei der Programmierung eines VST Plug-Ins werden der *process*-Funktion also die Audiodaten mitgegeben, die bearbeitet werden müssen. Die *process*-Funktion erhält eine gewisse Anzahl Samples, für die sie dann die gewünschte Veränderung der Audiodaten vornimmt.

Der in dieser Arbeit beschriebene Faltungsalgorithmus bezieht sich nur auf die Faltung der berechneten Koeffizienten mit dem hinteren Nierensignal (rechts auf der Stereospur). Das nun gefaltete Signal wird schließlich mit dem um 256 Samples verzögerten vorderen Nierensignal addiert, um so die gewünschte Richtcharakteristik zu erzeugen.

In diesem Plug-In wird die Faltung für jedes Sample neu berechnet. Die in Formel 3.4 dargestellte mathematische Grundlage wird für jedes Sample wieder neu durchlaufen. Wie in Kapitel 3.4.2 beschrieben, gelingt dies mit einer geringen Latenz von 256 Samples, da die einfache Faltung direkt mit dem ersten Sample beginnt und erst nach und nach die weiteren hinzuzieht. So muss nicht erst auf einen Block Samples der Größe 511 gewartet werden, wie dies bei der schnellen linearen Faltung der Fall wäre, sondern es kann sofort für jedes Sample auch ein neues berechnet werden. Allerdings ist aufgrund der Koeffizienten erst nach 256 Samples ein Signal zu hören, woraus sich die Latenz ergibt.

Das bedeutet die beiden Eingangssignale der vorderen und hinteren Niere werden vor der Berechnung temporär gespeichert. Während das vordere Nierensignal vorerst vernachlässigt wird, wird mit dem hinteren Nierensignal und den berechneten Filterkoeffizienten durch den Faltungsalgorithmus nun ein neues Signal errechnet. Danach wird dieses neu gewonnene Signal zu dem verzögerten Signal der vorderen Niere addiert. Das so gewonnene Ausgangssignal wird auf beide Ausgangskanäle (links, rechts) gelegt und somit sind die Einstellungen der Benutzeroberfläche nun auch für den Nutzer hörbar. Trotz der Nutzung einer Stereospur ergibt sich dann durch das Plug-In logischerweise ein Monosignal.

Die 511 Filterkoeffizienten werden in einem Array gespeichert und, wie schon erwähnt, nur bei Änderung der Benutzeroberfläche neu berechnet. Diesem Array muss

also ein Array mit 511 Samples gegenüberstehen, um die Formel 3.4 zu realisieren. So wird zuerst ein Array der Größe 511 erstellt und mit „0“ initialisiert. Das erste Sample wird dann an die erste Position gesetzt und somit neu berechnet. Für das zweite und alle folgenden Samples werden alle Samples um eine Position weiter geschoben und das aktuelle Sample auf die erste Position des Arrays gesetzt.

$$y(0) = h(0)x(0)$$
$$y(1) = h(1)x(0) + h(0)x(1)$$
$$y(2) = h(2)x(0) + h(1)x(1) + h(0)x(2)$$

Formel 3.6: Formeln für die ersten drei Schritte der einfachen Faltung (Ifeachor; Jervis, 1993)

In der Formel 3.6 sieht man gut, wie die ersten drei Samples neu berechnet werden. Während das erste Sample nur durch „eine“ Multiplikation bestimmt wird, braucht das dritte Sample schon drei Multiplikationen. Im Programmcode sind es für jeden Vorgang natürlich 511 Multiplikationen, da die in Formel 3.6 nicht aufgeführten Multiplikationen durch die Initialisierung des Arrays „0“ ergeben.
Die Samples werden immer eine Position weiter geschoben und das neue Sample kommt vorne hinzu. Die Koeffizienten bleiben stehen und so werden die Samples an den Koeffizienten vorbei geschoben und in jedem Schritt wird ein neues Sample berechnet, was durch Addition mit dem linken Eingangssample das neue Ausgangssample ergibt.

In diesem Kapitel konnte also nachvollzogen werden, wie durch die Benutzereingabe des Tonmeisters neue Filterkoeffizienten berechnet werden und wie diese schließlich auf ein Sample des Signals angewandt werden.

3.5 Probleme

In diesem Kapitel soll nun auf Probleme, die während der Implementierung auftraten, eingegangen werden. Es soll dargelegt werden, wo sich während des Entwicklungsprozesses größere Hürden befanden. Außerdem wird aufgezeigt, ob und wie diese gelöst werden konnten.

Mit Beginn der Arbeit traten Probleme bei der Einarbeitung in die VST-Programmierung auf. Es fiel deutlich schwerer als erwartet, ein Template-Projekt in der Entwicklungsumgebung – in diesem Fall Visual Studio 2008 – fehlerfrei zu erstellen. Besonders die Einbindung der *VST GUI 4* in das Projekt bereitete größere Startschwierigkeiten, da Beispielprojekte des VST SDK 3.5.0 zumeist noch die ältere VST *GUI SF* nutzen. So fehlte Anschauungsmaterial zur Nutzung der *VST GUI 4*.
Das Hauptproblem während der Einarbeitung und zu Anfang der Implementierung war, dass es im Prinzip noch recht wenig Entwickler von VST3 Plug-Ins gibt. Dadurch gibt es zu VST 3 kaum Tutorials oder anderes Anschauungsmaterial. Die wichtigste Quelle war die Dokumentation im VST SDK 3.5. Die Dokumentation erklärt die Struktur von VST3 Plug-Ins und den Unterschied zu VST2 Plug-Ins theoretisch sehr gut und man bekommt einen strukturierten Überblick. Allerdings fiel es zuerst schwer, die theoretischen Ausführungen in der Entwicklungsumgebung in die Praxis umzusetzen.

Außerdem ergaben sich Probleme bei der Implementierung der mathematischen Funktionen. Die Entwicklung des Plug-Ins seitens des Mikrofonherstellers Microtech Gefell fand in dem Mathematikprogramm „Matlab“ statt. Die Realisierung der mathematischen Funktionen in C++ mit seinen geringen mathematischen Fähigkeiten erforderte eine Einarbeitung in die Theorie der komplexen Zahlen sowie der Fouriertransformationen. So wurde die Mathematik neben der Informatik und der Tontechnik zu einem größeren Bestandteil als zu Beginn erwartet.

Im Kapitel 3.1 wurde als ein primäres Ziel festgelegt, dass das in dieser Arbeit entwickelte Plug-In in Echtzeit lauffähig sein soll. Da in dem in dieser Arbeit entwickelten Plug-In eine IFFT und eine einfache Faltung durchgeführt werden, war schon zu Beginn der Entwicklung klar, dass die Lauffähigkeit in Echtzeit eine der größten Herausforderungen darstellen würde.
Die einfache Faltung mit einer verhältnismäßig geringen Anzahl von 511 Koeffizienten verbraucht zwar relativ viel Prozessorlast, aber verhindert nicht, dass das Plug-In in Echtzeit lauffähig ist. Für eine größere Anzahl von Koeffizienten sollte allerdings unbedingt auf die schnelle lineare Faltung mit schnellen Fouriertransformationen gesetzt werden. Mehr dazu findet sich im Ausblick im Kapitel 5.1.

Das aber eigentliche Problem, um die Lauffähigkeit des Plug-Ins in Echtzeit zu gewährleisten, ist die IFFT. Ob das Plug-In nun in Echtzeit lauffähig ist, wird allein dadurch bestimmt, mit wie vielen Eingangswerten man die IFFT berechnet. Natürlich führt im Prinzip die Kombination der einfachen Faltung und der IFFT dazu, dass es zu Problemen im Echtzeitbetrieb kommen kann. Allerdings ist es die IFFT, die zwar weniger Prozessorlast in Anspruch nimmt als die einfache Faltung, aber im Gegenzug die ASIO[26]-Auslastung der Soundkarte an ihre Grenzen bringt.
Zu Beginn der Arbeit wurde noch mit Messwerten von 2^{18} = 262144 gearbeitet. Bei dieser Anzahl von Werten sind alle Host-Programme (Sequenzer-Software) beim Start des Plug-Ins sofort abgestürzt. Erst bei geringeren Zweierpotenzen von ca. 2^{16} = 65536 stürzten die Host-Programme nicht mehr ab.[27]
Nach Gesprächen mit dem Unternehmen Microtech Gefell einigte man sich auf eine Anzahl von 2^{14} = 16384 Werten als Eingang der IFFT. Durch diese Werte ist das Plug-In auf der einen Seite lauffähig in einer Sequenzer-Software, auf der anderen Seite ist es mit ein paar Einschränkungen auch in Echtzeit lauffähig. Führt man das Plug-In in einer Sequenzersoftware auf einem normalen PC aus, so wird man feststellen, dass bei Betätigung eines Steuerelements der Benutzeroberfläche die Audiodaten evtl. fehlerhaft abgespielt werden. Mit Beendigung der Benutzerinteraktion stellt sich dieses Phänomen wieder ein. Das bedeutet somit, dass bei Ausführung der Berechnung neuer Koeffizienten und somit auch der IFFT Probleme auftreten.

Wie schon erwähnt bezieht sich dies auf die Ausführung des Plug-Ins auf einem normalen Rechner. Verwendet man stattdessen einen Audiorechner – insbesondere mit spezieller Audio-Soundkarte[28] – so hat man die Möglichkeit den Puffer der Soundkarte möglichst hoch einzustellen. Stellt man diesen Puffer auf den Höchstwert, läuft das Plug-In fehlerfrei in Echtzeit. In der Praxis wird der Puffer während einer

[26] ASIO (Audio Stream Input/Output) ist ein von Steinberg entwickeltes Audiotransferprotokoll.

[27] Die Angaben beziehen sich auf Tests mit dem Rechner, mit dem das Plug-In entwickelt wurde, somit einem normalen Laptop ohne spezielle Audio-Soundkarte.

[28] In diesem Fall wurde auf einem speziellen Audiorechner mit einer RME Hammerfall DSP Soundkarte getestet, dessen Puffergröße größtmöglich auf 4096 eingestellt war.

Aufnahme möglichst gering eingestellt. Während des Postprocessings wird der Puffer größtmöglich eingestellt, um bei der Verwendung und Berechnung von Audio Plug-Ins möglichst viel ASIO-Power zur Verfügung zu haben.

Also lässt sich abschließend sagen, dass das Plug-In in professionellen Audio-Umgebungen problemlos in Echtzeit lauffähig ist, sofern man den Puffer der Audio-Soundkarte hoch genug einstellt.

4 Probeaufnahme und abschließende Bewertung des Plug-Ins

4.1 Aufnahmesituation

Im Rahmen dieser Arbeit wurden zwei Probeaufnahmen mit dem Mikrofon *„UM 930 Twin“* durchgeführt. Es wurde ein Holzbläser-Trio mit Querflöte, Klarinette und Saxophon und ein Jazz-Trio mit Piano, Schlagzeug und Basssaxophon aufgezeichnet. Für die folgenden Kapitel wurde ein Ausschnitt der Aufnahme des Jazz-Trios mit dem entwickelten Plug-In bearbeitet.

Bevor in den nächsten Kapiteln auf die Einflussnahme des Plug-Ins auf die Aufnahme genauer eingegangen wird, sollen im Folgenden noch die Eckdaten der Aufnahme und die Aufnahmesituation genauer beschrieben werden.
Aufgenommen wurde das Stück *„Cyberrobot attacks“* des Stuttgarter Jazz-Trios *„New Thing Trio“*. Das *„New Thing Trio“* setzt sich aus Christoph Beck (Basssaxophon), Henry Kasper (Piano) und Markus Zink (Schlagzeug) zusammen. Das in dieser Arbeit relevante Stück wurde von Markus Zink geschrieben.

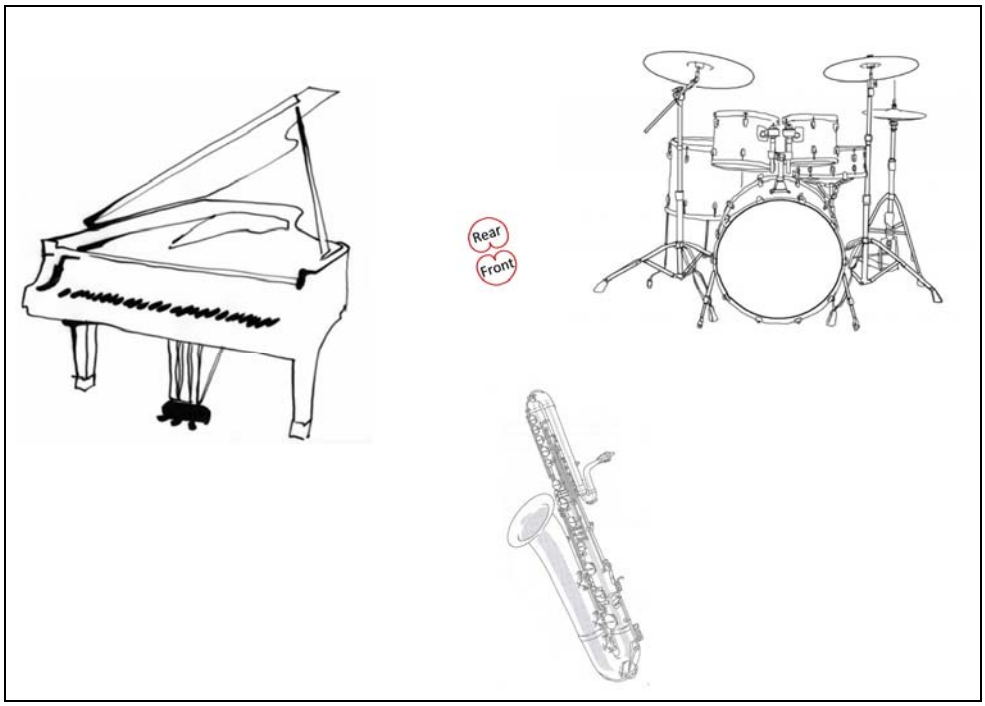

Abbildung 4.1: Skizzenhafte Aufstellung um das Twin-Mikrofon, dass durch zwei Nierensymbole dargestellt ist

Das Mikrofon wurde, wie in Abbildung 4.1 zu sehen ist, im Prinzip in der Mitte der Musiker platziert. Das Basssaxophon befindet sich frontal vor der vorderen Niere, sodass sich Piano und Schlagzeug links und rechts neben dem Mikrofon befinden. Diese Aufstellung wurde bewusst gewählt, um im Postprocessing die Möglichkeit zu haben, das Verhältnis der Signale von „Piano und Schlagzeug“ und „Basssaxophon“ durch die Richtcharakteristik zu variieren. Denn, wenn das Basssaxophon im oben genannten Stück spielt, nimmt es meist die Rolle des Solisten ein und das Piano und das Schlagzeug dienen als Rhythmusgruppe.

Die Aufnahme wurde im Stuttgarter Jazz-Club „Bix“ gemacht. Während der Aufnahme war kein Publikum anwesend. Die Bühne des „Bix“ eignet sich sehr gut für eine solche Aufnahme, da sie sehr wenige Raumreflexionen zurückwirft und somit sehr trocken klingt. Dadurch hat man im Postprocessing noch alle Möglichkeiten, begonnen bei dem in dieser Arbeit erstellten Plug-In bis hin zu Halleffekten, um auf das Aufnahmesignal Einfluss zu nehmen.

Abbildung 4.2: Foto während der Aufnahme im Stuttgarter Jazzclub „Bix“ von Christoph Beck (Basssaxophon), Henry Kasper (Piano) und Markus Zink (Schlagzeug)

4.2 Möglichkeiten der Einflussnahme auf die Aufnahme durch das Plug-In in der Postproduction

Anhand des im vorherigen Kapitel erwähnten Stückes des Stuttgarter Jazz-Trios *„New Thing Trio“* soll nun auf die Möglichkeiten der Einflussnahme durch das Plug-In auf die Aufnahme eingegangen werden. Da eine Analyse des kompletten Musikstücks zu umfangreich für diese Arbeit wäre, soll im Folgenden besonders auf den Schlussteil eingegangen werden. Dieser wird durch ein Pianosolo eingeleitet. Das Schlagzeug und das Basssaxophon steigen ruhig ein und die drei Musiker steigern sich in ein lautes *„Tutti“* mit dem das Stück dann endet. Deshalb eignet sich besonders dieser Ausschnitt gut, um eine Einstellung des Plug-Ins zu finden, die auch für das komplette Stück verwendet werden kann. Die Bearbeitung der Aufnahme wurde mit der Sequenzer-Software *Cubase 5* durchgeführt.

Wenn man am Anfang versucht, die Chancen und Grenzen des Plug-Ins zu testen, indem man den Schieberegler und somit die Richtcharakteristik konstant von Kugel bis Acht variiert, ist eindrucksvoll zu hören, welche Chancen das Plug-In mit sich bringt. Ist der Regler auf Kugel gestellt, wirkt es als höre man ein Hauptmikrofon, das den Gesamtklang aller Musiker gut eingefangen hat. Stellt man den Schieberegler auf Acht, klingt es allerdings nach einem Stützmikrofon des Basssaxophons.

Zu Beginn der Bearbeitung wurde zuerst mithilfe des Schiebereglers die konstante Richtcharakteristik gesucht, bei der die Balance der einzelnen Instrumente untereinander am besten wirkte. Da das Mikrofon in der Mitte der drei Musiker stand und diese in einem Halbkreis um das Mikrofon platziert waren, ist es logisch das hierfür eine Richtcharakteristik unterhalb der Niere am nächsten lag. So wurde die konstante Gerade in etwa zwischen Kugel und breiter Niere angeordnet, wie in Abbildung 4.3 zu sehen ist.
Durch diese Einstellung ordnet sich das Basssaxophon leicht über den anderen beiden Instrumenten an, wobei der Flügel noch gut zu hören ist und das Schlagzeug nicht zu laut wirkt.

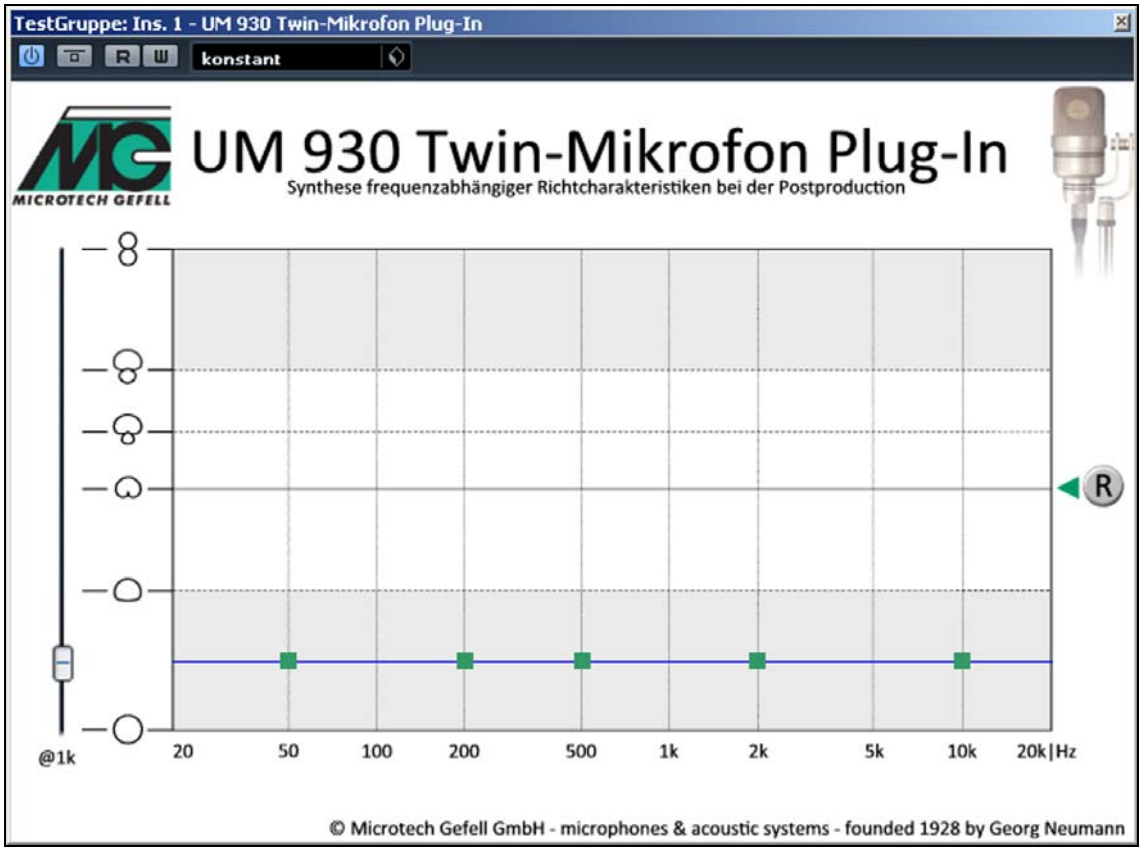

Abbildung 4.3: Konstante Einstellung der Richtcharakteristik für die Aufnahme des Stuttgarter "New Thing Trio"

Nachdem nun über den gesamten Frequenzbereich eine konstante Richtcharakteristik gefunden wurde, sollten noch die Möglichkeiten der frequenzabhängigen Synthese der Richtcharakteristik genauer untersucht werden.

Eine Schwierigkeit dieser Aufnahme ist, dass das Schlagzeug sowie der Flügel sich beide seitlich neben dem Mikrofon befinden, denn in einigen Fällen ist das Signal des Schlagzeugs deutlich lauter als das des Flügels. Durch konstante Einstellung der Richtcharakteristik kann man allerdings nur beide in etwa gleichermaßen dämpfen. Durch die frequenzabhängige Synthese der Richtcharakteristik soll nun eine gute Balance zwischen Piano und Schlagzeug erreicht werden. Außerdem soll das Basssaxophon etwas direkter und präsenter wirken. Die Grundlage bildet die zu Beginn gefundene konstante Richtcharakteristik.

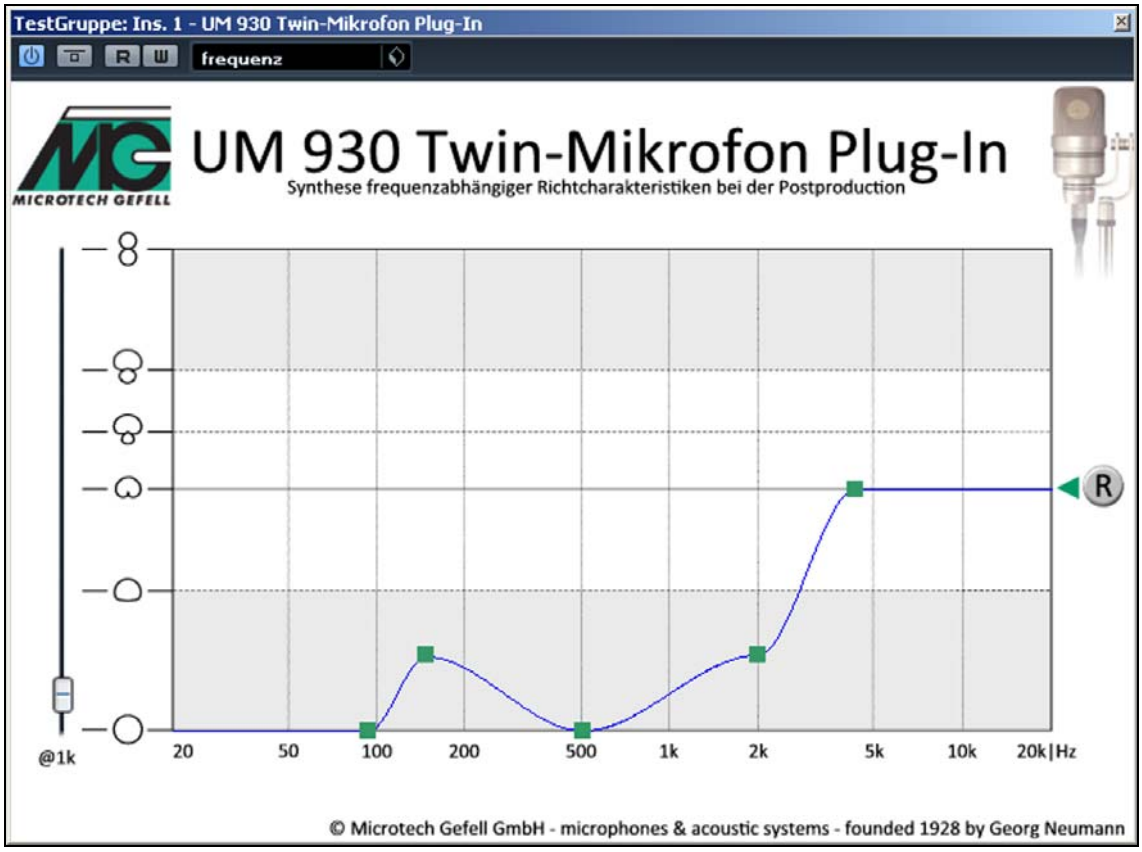

Abbildung 4.4: Frequenzabhängige Einstellung der Richtcharakteristik für die Aufnahme des Stuttgarter "New Thing Trio"

Unter 100 Hz wird die Richtcharakteristik Kugel verwendet. Das hat den Vorteil, dass die Bassdrum des Schlagzeugs noch deutlicher zu hören ist und ein etwas wärmerer Klang erzeugt wird.

Da das Piano bisher meist sehr schlank bzw. dünn klang und im Verhältnis zu den anderen Instrumenten etwas mehr Substanz benötigte, wird die Richtcharakteristik bei 500 Hz mit einer weichen Kurve ebenfalls auf Kugel gestellt.

Ab 5 kHz aufwärts wird die Richtcharakteristik durch einen weichen Übergang zur Niere etwas mehr eingeschränkt. Zum einen wird dadurch der Anschlag der Snaredrum etwas zurückgenommen und das Verhältnis von Bassdrum zu Snaredrum klingt nun insgesamt ausgewogener. Zum anderen klingt das Basssaxophon nun etwas präsenter und direkter. Allerdings verändert sich die Klangfarbe des Basssaxophons ein wenig. Das Schlagzeug wirkt indirekter, aber die Balance von Piano und Schlagzeug wurde verbessert.

Auf den ersten Blick sieht die nun generierte Kurve durchaus unkonventionell aus, aber der akustische Höreindruck zeigt deutlich, welche Chancen das *„UM 930 Twin“* in Kombination mit dem in dieser Arbeit entwickelten Plug-In mit sich bringt. Auch, wenn die in dieser Arbeit erstellte Aufnahmesituation vielleicht keinem klassischen Anwendungsfall – wenn es einen solchen für ein Twin-Mikrofon überhaupt gibt – entsprach, so konnte doch das Potential der Technologie aufgezeigt werden.

4.3 Bewertung

Nachdem nun die Möglichkeiten der Einflussnahme des Plug-Ins auf eine Aufnahme an einem praktischen Beispiel dargelegt wurden, soll nun noch einmal auf die Zukunftschancen und Anwendungsmöglichkeiten des in dieser Arbeit entwickelten Plug-In eingegangen werden. Außerdem soll das erzielte Ergebnis bewertet werden.
Grundsätzlich ist zu sagen, dass sich die Nutzung des Twin-Mikrofons und die Anwendung des Plug-Ins im Postprocessing gelohnt haben. Durch eine geschickte Platzierung des Mikrofons wurde eine sehr flexible Ausgangslage für das Postprocessing geschaffen. Die Suche nach der perfekten Richtcharakteristik landete nicht bei einer der klassischen Richtcharakteristiken, sondern genau zwischen Kugel und breiter Niere. Nur dadurch war die Balance der einzelnen Instrumente bei einer Aufnahme mit nur einem Mikrofon in dieser Qualität möglich.

Die Einstellung der Richtcharakteristik in Abhängigkeit der Frequenz ermöglichte es, weitere Details mit dem Plug-In zu bearbeiten. So konnte man in diesem praktischen Beispiel die Dämpfung bestimmter Richtungen und somit Instrumente in Abhängigkeit der Frequenz wählen und einen noch besseren Klang erstellen. Man kann also schon in dieser frühen Phase des Postprocessings – wenn auch nur in begrenztem Maß – in den Klang und die Räumlichkeit eines Signals eingreifen, ohne einen Equalizer oder ein Hall Plug-In verwenden zu müssen. Dadurch ergibt sich mehr Flexibilität in der klanglichen Gestaltung.
Achtet man besonders auf das Basssaxophon kann man bei unterschiedlichen Einstellungen der Richtcharakteristik eine Veränderung der Klangfarbe feststellen. Daraus ergibt sich, dass mit dem Mikrofon durch Veränderung der Richtcharakteristik verschiedene Klangfarben möglich sind. Somit muss die „Wärme“ oder die „Präsenz“

einer Aufnahme nicht schon bei der Mikrofonwahl entschieden werden oder durch Effektgeräte erzeugt werden, sondern könnte zum Teil auch durch die Wahl der Richtcharakteristik variiert werden.

Durch die in diesem Kapitel genannten Möglichkeiten und Chancen ergeben sich verschiedenste Einsatzgebiete für das Mikrofon in Kombination mit dem Plug-In. So wäre das *„UM 930 Twin“* als Duo gut als Hauptmikrofon bei einer Orchesteraufnahme einsetzbar. Besonders wenn die akustische Umgebung im Vorfeld nicht bekannt ist und somit die Entscheidung zwischen zwei Nieren oder zwei Kugeln fallen müsste, wäre man mit dem *„UM 930 Twin“* deutlich flexibler.
Außerdem wurde während der Nachbearbeitung der Probeaufnahme deutlich, dass sich das Twin-Mikrofon auch sehr gut als Stützmikrofon eignen würde. Besonders bei Solisten – deren Abschattung vom Gesamtorchester wichtig ist – könnte man in der Nachbearbeitung deutlich flexibler sein.
Ein weiteres interessantes Anwendungsfeld könnten Sprach- und Gesangsaufnahmen im Tonstudio sein. Hier werden ohnehin häufig umschaltbare Kondensatormikrofone eingesetzt. Besonders, wenn mehrere Musiker gleichzeitig im gleichen Raum aufnehmen, würde man bezüglich der Abschattung der einzelnen Schallquellen untereinander mehr Möglichkeiten haben.
In diesem Kapitel wurden die Funktionalität und die damit verbundenen Chancen und Möglichkeiten des Plug-In aufgezeigt. Aus den Ergebnissen der Probeaufnahme konnten außerdem mögliche Anwendungsgebiete hergeleitet werden.

5 Ausblick

5.1 Performance

Die Performance des Plug-Ins wurde in dieser Arbeit nun schon öfter erwähnt. Performance meint in diesem Zusammenhang, inwiefern das Plug-In den Prozessor sowie die Soundkarte beansprucht. Ob das Plug-In in Echtzeit lauffähig ist, hängt allein von der Performance der verwendeten Algorithmen ab. Zudem muss man davon ausgehen, dass ein Tonmeister in einem Projekt nicht nur dieses eine Plug-In verwenden wird. In der Regel sind mehrere Plug-Ins gleichzeitig aktiv. Deshalb wird die Performance des Plug-Ins besonders bei der Weiterentwicklung und in zukünftigen Versionen eine wichtige Rolle spielen.
Die zwei für die Performance besonders relevanten Algorithmen in diesem Plug-In sind die IFFT und der Algorithmus der einfachen Faltung. Im Folgenden soll zuerst auf möglichen Verbesserungen der IFFT eingegangen werden, bevor dann in Zukunft mögliche Änderungen im Faltungsalgorithmus erläutert werden.

Wie im Laufe der Arbeit erwähnt wurde, wird bewusst auf eine inverse schnelle Fouriertransformation gesetzt. Dieser Algorithmus ist sehr gut implementiert und das Plug-In funktioniert in professionellen Audioumgebungen auch reibungslos. Allerdings geht der Algorithmus davon aus, dass er mit komplexen Zahlen rechnet. In diesem Fall startet die IFFT die Berechnung auch mit komplexen Werten. Allerdings entsteht durch die IFFT ein reelles Signal.
Es gibt Möglichkeiten, wie man schnelle Fouriertransformationen für reelle Signale optimieren kann. Sollte die Anzahl der Eingangswerte in Zukunft noch erhöht werden, wäre sicher interessant zu untersuchen, ob Verbesserungen für die hier verwendete IFFT möglich sind.
Eine weitere Option wäre, die Berechnung der Filterkoeffizienten mit einer Anzahl von 2^{14} Werten durchzuführen, bis man die ideale Einstellung gefunden hat. Durch Betätigung eines Buttons wird die Benutzeroberfläche eingefroren und im Hintergrund werden mit einer noch höheren 2er-Potenz einmalig die Koeffizienten mit mehr Eingangswerten ausgerechnet. So könnte ein Tonmeister in guter Audio-Qualität und

in Echtzeit entscheiden, was in seiner Situation die perfekte Einstellung ist. Hat er diese gefunden, so kann er den Button betätigen. Dann werden seine Einstellungen noch einmal mit nahezu perfekter Qualität berechnet. Nach dem Lösen des Buttons kann wieder normal weiter gearbeitet werden. Hierfür bietet sich ein *„COnOffButton“* aus der *VST GUI 4* an. Das in dieser Arbeit entwickelte Plug-In läuft sehr gut in Verbindung mit professioneller Audio-Hardware bei einer Verwendung von 2^{14} Werten. Sollte in der Zukunft eine noch größere 2er-Potenz angestrebt werden, sollten die erwähnten Möglichkeiten genauer untersucht werden.

Der zweite entscheidende Faktor für die Performance des Plug-Ins ist der Faltungsalgorithmus. In der ersten Version des Plug-Ins bot sich bei einer Koeffizientenanzahl von 511 noch eine einfache Faltung an. Im Bereich von 511 Koeffizienten ist dieser Algorithmus von der Performance her gut. Würde man in zukünftigen Versionen des Plug-Ins allerdings eine größere Koeffizientenzahl anstreben, könnte dieser Algorithmus an seine Grenzen stoßen.
Der zweite schon in dieser Arbeit erläuterte Algorithmus ist der der schnellen linearen Faltung. Da dieser Algorithmus – wie in Kapitel 3.4.2 beschrieben – allerdings schnelle Fouriertransformationen erfordert, müsste die Anzahl der Koeffizienten einer 2er-Potenz entsprechen. Momentan ist die Mathematik so gedacht, dass die Faltung mit 511 Koeffizienten durchgeführt wird. Deshalb sollte bei einer Faltung mit mehr Koeffizienten überlegt werden, ob man mit beispielsweise 1024 Koeffizienten eine schnelle lineare Faltung vornimmt, statt mit 511 eine einfache Faltung zu machen. Das könnte langfristig Vorteile für die Weiterentwicklung und die Flexibilität des Plug-Ins haben. So könnte ein Tonmeister in der Benutzeroberfläche selbst die Anzahl der Faltungskoeffizienten und somit die Latenz bestimmen.

Bei der schnellen linearen Faltung würde sich natürlich eine größere Latenz ergeben, da hier erst immer auf eine gewisse Anzahl Samples gewartet werden muss, bevor gefaltet werden kann. Bei 1024 Koeffizienten hat man also 1024 Samples Latenz. Dies kann in der Regel die Sequenzer-Software ausgleichen, sofern man die Anzahl der Samples in der Methode *„uint32 PLUGIN_API getlatencySamples()“* zurückgibt.

5.2 Grafische Benutzeroberfläche

Wie in Kapitel 3.2 beschrieben, soll die grafische Benutzeroberfläche sehr intuitiv und überschaubar wirken. Die Bedienung des Plug-Ins sollte einem Nutzer des *„UM 930 Twin“* schon auf den ersten Blick verständlich vorkommen. Obwohl die Benutzeroberfläche diese Anforderungen sehr gut erfüllt, sind im Laufe der Entwicklung noch einige Ideen aufgetaucht, die im Rahmen dieser Arbeit aus Zeitgründen nicht verwirklicht werden konnten.

Im vorherigen Kapitel wurden zur Steigerung der Performance zwei Möglichkeiten aufgezeigt, die auch auf die Oberfläche Auswirkungen hätten: Zum einen der bereits erwähnte Button, um die Oberfläche einzufrieren, und zum anderen wurde erwähnt, dass der Tonmeister die Anzahl der Koeffizienten für die Faltung selbst bestimmen könnte. Hierzu könnte man ein *„COptionMenu“* aus der *VST GUI 4* nutzen.

Abgesehen von den möglichen performancebedingten Änderungen der Oberfläche sind noch weitere Aspekte interessant. Es könnte z.B. für den Gesamteindruck der Oberfläche hilfreich sein, noch einige VU-Meter zu haben, die anzeigen, wie viel Pegel die beiden Eingänge liefern und wie groß der Pegel des Ausgangssignals ist. In diesem Plug-In bietet sich eine Anzahl von drei VU-Metern – für die beiden Eingangskanäle und das Mono-Ausgangssignal – an. Eine Vertauschung der beiden Nierensignale könnte dadurch schneller erkannt werden, da das vordere Nierensignal in vielen Fällen einen höheren Pegel haben müsste. Zur Darstellung eines VU-Meters stellt *die VST GUI 4* das Steuerelement *„CVuMeter“* zur Verfügung.

Da einigen Nutzern möglicherweise die Verbindung zwischen Schieberegler und der Kurve im Koordinatensystem nicht auf den ersten Blick ersichtlich ist, könnte man beim Betätigen des Schiebereglers eine Markierung einblenden, die kenntlich macht, dass sich dieser auf der Höhe der Kurve bei 1 kHz befindet. So könnten z.B. nur während der Betätigung des Schiebereglers zwei Geraden eingeblendet werden, die jeweils vom Punkt, an dem die Kurve 1 kHz schneidet, zu den Achsen des Koordinatensystems verlaufen. Eine senkrechte Gerade von der x-Achse zum Punkt bei 1 kHz und eine vertikale Gerade von der y-Achse zum Punkt bei 1 kHz. Die Linien

sollten sehr dünn und eher matt bzw. fast transparent erscheinen. Sie sollten nur sehr verhalten den Zusammenhang andeuten, ohne sich in den Vordergrund zu drängen.
Eine weitere Besonderheit des Plug-Ins könnte sein, dass mit ihm schon Presets mitgeliefert werden, in denen Einstellungen für bestimmte Situationen vorgenommen worden sind. So könnte ein Preset z.B. „Warme Bässe und direkte Höhen“ heißen. Die Presets könnten nicht nur wie gewohnt über die Presetfunktionen geladen werden, sondern evtl. auch über ein weiteres *„OptionMenu“*. Eine solche Funktion könnte dem Anwender den Einstieg in die Nutzung des Plug-Ins enorm erleichtern.

Auswählbare Presets, die visuelle Darstellung der Synchronisation zwischen Schieberegler und Kurve sowie einige VU-Meter könnten die Oberfläche komplettieren und für einen noch besseren ersten Gesamteindruck sorgen.

5.3 Allgemein

Nachdem nun speziell auf das in dieser Arbeit entwickelte Plug-In und mögliche Weiterentwicklungen in Bezug auf Performance und Benutzeroberfläche eingegangen wurde, soll in diesem Kapitel zum Schluss noch ein größerer Bogen gespannt werden. Denn neben der Synthese frequenzabhängiger Richtcharakteristiken ist langfristig noch die Einstellung eines Wunschamplitudenfrequenzgangs geplant.

Die Hauptfunktion des Plug-Ins – die Einstellung der Richtcharakteristik in Abhängigkeit der Frequenz – wurde implementiert und ist auch für einen Tonmeister schon gut nutzbar. Allerdings bringt die Berechnung der Filterkoeffizienten nach dem in dieser Arbeit beschriebenen Schema noch einen weiteren Vorteil mit sich. Es würde sich langfristig auch eine zweite Funktion integrieren lassen. Durch weitere mathematische Funktionen, die in die Berechnung der Filterkoeffizienten einfließen, ist es möglich, einen Wunschamplitudenfrequenzgang einzustellen. Das bedeutet, für das Mikrofon kann ein Frequenzgang im Stile eines Equalizers eingestellt werden. Für diese Möglichkeit bedarf es natürlich eines zweiten Koordinatensystems.
Da die Benutzeroberfläche bzw. die Fenstergröße des Plug-Ins ohnehin schon relativ groß ist, würde es sich nicht anbieten noch ein Koordinatensystem neben oder unter das bereits erstellte zu platzieren. Alternativ würde sich anbieten, dem Nutzer die

Möglichkeit zu geben das Koordinatensystem zu wechseln, indem man sich z.B. durch Klick auf einen Button oder eine Checkbox für ein Koordinatensystem entscheidet. Beim Start des Plug-Ins wäre somit nur das in dieser Arbeit erstellte Koordinatensystem zu sehen. Allerdings hat ein Tonmeister jederzeit die Chance die Ansicht zu wechseln und somit entweder die Richtcharakteristik in Abhängigkeit der Frequenz oder den Wunschamplitudenfrequenzgang einzustellen.

Zu Beginn der Arbeit wurde schon erwähnt, dass es im VST SDK 3.5.0 einen Wrapper gibt, der es ermöglicht aus einem VST3 Plug-In ein VST2 Plug-In zu erstellen. Außerdem existieren auch weitere Wrapper und Programme, die es ermöglichen VST Plug-Ins in andere Formate zu überführen. Möglicherweise bietet sich dadurch die Chance, dass Plug-In für noch mehr Plattformen nutzbar zu machen.

Am Ende dieser Arbeit steht also ein VST3 Plug-In zur Synthese frequenzabhängiger Richtcharakteristiken, welches für einen Tonmeister sofort einsetzbar ist. Zukünftige Änderungen und Weiterentwicklungen können unter Berücksichtigung der in diesem Kapitel genannten Faktoren effizienter umgesetzt werden.

6 Fazit

Es ist im Rahmen dieser Arbeit gelungen ein VST3 Plug-In zu entwickeln, das einem Tonmeister nach der Verwendung des Mikrofons *„UM 930 Twin“* die frequenzabhängige Synthese der Richtcharakteristik ermöglicht.

Zu Beginn der Arbeit wurden die relevanten Grundlagen zu Richtcharakteristiken und Audio Plug-Ins erläutert. Außerdem wurden zu Beginn der Arbeit die Idee und die Funktionsweise eines Twin-Mikrofons dargelegt.
Basierend auf diesen Grundlagen und der Idee, die frequenzabhängige Einstellung der Richtcharakteristik durch ein VST3 Plug-In zu realisieren, wurde im Kapitel 3 die Implementierung des in dieser Arbeit entwickelten Plug-Ins beschrieben. Die Umsetzung der drei Meilensteine Gestaltung der Benutzeroberfläche, Berechnung der Filterkoeffizienten und schließlich die Realisierung des Faltungsalgorithmus wurden im Hauptkapitel dieser Arbeit detailliert aufgezeigt.

Durch die in Kapitel 4 diskutierte Probeaufnahme konnten die Funktionsweise und die Möglichkeiten des Plug-Ins eindrucksvoll hörbar gemacht werden. Dieses praktische Beispiel ermöglichte es, die Einflussnahme des Plug-Ins auf eine Aufnahme darzulegen und in Zukunft mögliche Einsatzgebiete zu erkennen. Zum Ende der Arbeit wurden in Kapitel 5 interessante Faktoren für die Weiterentwicklung des Plug-Ins erwähnt.

Das in dieser Arbeit entwickelte Plug-In eröffnet einem Tonmeister völlig neue Chancen und Möglichkeiten vom Planungsprozess einer Aufnahme bis zur Nachbearbeitung einer Aufnahme. Das entscheidende Schlagwort, welches in dieser Arbeit schon öfter gefallen ist, heißt „Flexibilität“. Ein Twin-Mikrofon in Kombination mit einem Audio Plug-In bringt viele Chancen und Einsatzmöglichkeiten mit sich, sodass es sehr universell einsetzbar ist. Die Möglichkeit an der Sequenzer-Software zu entscheiden, wie die Richtcharakteristik in Abhängigkeit der Frequenz für die jeweilige Aufnahme sein soll, kann die Planung der Mikrofonierung einer Aufnahme vereinfachen.

Die Flexibilität und die universellen Einsatzmöglichkeiten sorgten schon für die große Verbreitung von klassisch umschaltbaren Kondensatormikrofonen. Diese Faktoren wurden durch den Einsatz des in dieser Arbeit einwickelten Plug-In noch weiter verbessert. Neben der Möglichkeit Zwischenstufen bzw. gar neuartige Richtcharakteristiken einzustellen ist nun auch die frequenzabhängige Synthese der Richtcharakteristik möglich.

Besonders im Hinblick auf die Entwicklung des Computers zur Digital Audio Workstation (DAW) könnten Twin-Mikrofone in Kombination mit einem Audio Plug-In zu den *modernen* „Arbeitspferden“ im Tonstudio der Zukunft werden.

Literatur- und Quellenverzeichnis

Achilles, Dietmar: *Die Fourier-Transformation in der Signalverarbeitung. Kontinuierliche und diskrete Verfahren der Praxis*. Berlin Heidelberg New York Tokyo 1985.

Black Duck Software: *ifft - In-place radix 2 decimation in time inverse FFT*. Abgerufen am 01.08.2011 von http://www.koders.com /cpp/fidA424C18889D6FD61058A51D460FC5F35FE52093E.aspx?s=ifft

Bremm, Peter: *Das digitale Tonstudio. Praktische Hilfe zur digitalen Tonstudiotechnik*. Bergkirchen 2004.

Brigham, E. Oran: *FFT. Schnelle Fourier-Transformation*. 3. Auflage, München 1987

Dickreiter, M.; Dittl, V.; Hoeg, W.; Wöhr, M.: *Handbuch der Tonstudiotechnik*. *Band 1*. 7. Auflage, München 2008.

Dickreiter, M.; Dittl, V.; Hoeg, W.; Wöhr, M.: *Handbuch der Tonstudiotechnik*. *Band 2*. 7. Auflage, München 2008.

Dickreiter, Michael: *Mikrofon-Aufnahmetechnik*. 3. Auflage, Stuttgart 2002.

Domke, Matthias: *Synthese von Richtcharakteristiken mit einem twin-Mikrofon.* Vortrag bei der Tonmeistertagung. Leipzig 2010.

Görne, Thomas: *Mikrofone in Theorie und Praxis*. 8. Auflage, Aachen 2007.

Görne, Thomas: *Signale und Systeme in Zeit und Frequenzbereich*. Hamburg 2008. Abgerufen am 12.07.2011 von http://www.mt.haw-hamburg.de/home/goerne/mat/zeitfunktion_frequenzfunktion.pdf

Görne, Thomas: *Synthese von Gradienten-Richtcharakteristiken*. Hamburg 2008. Abgerufen am 12.07.2011 von http://www.mt.haw-hamburg.de/home/goerne/mat/gradienten_synthese.pdf

Görne, Thomas: *Tontechnik*. 2. Auflage, München 2008.

Henle, Hubert: *Das Tonstudio Handbuch*. 5. Auflage, München 2001.

Ifeachor, Emmanuel; Jervis, Barrier: *Digital Signal Processing - A practical Approach*. Great Britain 1993.

Kammeyer, Karl-Dirk; Kroschel Kristian: *Digital Signal-Verarbeitung*. 6. Auflage, Wiesbaden 2006.

Kress, Dieter; Kaufhold, Benno: *Signale und Systeme verstehen und vertiefen. Denken und Arbeiten im Zeit und Frequenzbereich*. Wiesbaden 2010.

Microtech Gefell: *UM 930 twin Prospekt*. Abgerufen am 16.06.2011 von http://www.microtechgefell.de/dmdocuments/UM%20930%20twin_deu.pdf

Steinberg Media Technologies: *Documentation.* VST Software Development Kit Version 3.5.0, 2011.

Steinberg: *Cubase VST. Die Referenz*. Bonn 1999.

Stoer, Josef: *Numerische Mathematik 1*. 9. Auflage, Heidelberg 2005.

Stroustrup, Bjarne: *Die C++ Programmiersprache*. 3. Auflage, Bonn 1998.

von Grüningen, Daniel Ch.: *Digitale Signalverarbeitung mit einer Einführung in die kontinuierlichen Signale und Systeme*. 3. Auflage, München Wien 2004.

Watkinson, John: *The Art of Digital Audio*. 3. Auflage, Oxford 2001.

Weinzierl, Stefan: *Handbuch der Audiotechnik*. Berlin Heidelberg 2008.

Wolf, Jürgen: *C++ von A bis Z. Das umfassende Handbuch*. Bonn 2006

Zölner, Udo: *Digitale Audiosignal-Verarbeitung*. 3. Auflage, Wiesbaden 2005.

Abbildungsverzeichnis

Abbildung 2.1: Mit Gradientenkapseln erreichbare Richtcharakteristiken (Görne, 2007)..... 4

Abbildung 2.2: Polardiagramm für das "UM 930 Twin" des Unternehmens Microtech Gefell bei 1 kHz, 8 kHz und 16 kHz (Microtech Gefell)..... 8

Abbildung 2.3: Kommunikation zwischen den zwei Komponenten Processor und Controller (Steinberg Media Technologies)..... 11

Abbildung 2.4: Aufbau von Mikrofonen mit umschaltbarer Richtcharakteristik (Microtech Gefell) 12

Abbildung 2.5: Funktionsweise einer elektrisch umschaltbaren Kapsel (Görne, 2007)..... 13

Abbildung 2.6: Einstellungen der Richtcharakteristik am Mischpult (Microtech Gefell)..... 14

Abbildung 2.7: Aufbau von Twin-Mikrofonen (Microtech Gefell) 15

Abbildung 2.8: 2D Abbildung der Richtcharakteristik einer Niere ohne FIR-Filterung und mit FIR-Filterung (Microtech Gefell) 17

Abbildung 2.9: : 2D Abbildung eines Übergangs der Richtcharakteristiken Hyperniere zur breiten Niere mit FIR-Filterung (Microtech Gefell)..... 18

Abbildung 3.1: Das Mikrofon "UM 930 Twin“ des Unternehmens Microtech Gefell (Microtech Gefell) ... 20

Abbildung 3.2: Stellung des Schaltringes, um die vordere und die hintere Niere aufzuzeichnen, sowie die Beschriftung der Kabel (Microtech Gefell)..... 21

Abbildung 3.3: Visuelle Darstellung der drei Teile der Benutzeroberfläche: Konstante (1) sowie frequenzabhängige (2) Einstellung der Richtcharakteristik und die Kopfzeile der Sequenzer-Software (3) 23

Abbildung 3.4: Darstellung einer möglichen Kurve in der Benutzeroberfläche mit einem weichen Übergang von Kugel zur Niere und Einengung einer bestimmten Resonanzfrequenz..... 27

Abbildung 3.5: Darstellung einer Kurve mit cubischen Splinefunktionen in der Benutzeroberfläche 30

Abbildung 3.6: Darstellung einer Kurve mit einer Funktion dritten Gerades in der Benutzeroberfläche... 31

Abbildung 3.7: Vergleich zweier ähnlicher Kurven mit den verschiedenen Darstellungsformen in der Benutzeroberfläche..... 31

Abbildung 3.8: Kopfzeilen aus zwei verschiedenen Programmen: Steinberg Cubase 6 (oben) und der VST TestHost (unten) 33

Abbildung 3.9: FIR-Übertragungsfunktion: Postprocessing bei hohen Frequenzen (Microtech Gefell) ..... ***Fehler! Textmarke nicht definiert.***

Abbildung 3.10: Arbeitsweise mit einem Twin-Mikrofon von der Aufnahme bis zur Synthese der Richtcharakteristik in Abhängigkeit der Frequenz in der Postproduction (Microtech Gefell) *42*

Abbildung 4.1: Skizzenhafte Aufstellung um das Twin-Mikrofon, dass durch zwei Nierensymbole dargestellt ist *51*

Abbildung 4.2: Foto während der Aufnahme im Stuttgarter Jazzclub „Bix“ von Christoph Beck (Basssaxophon), Henry Kasper (Piano) und Markus Zink (Schlagzeug) *52*

Abbildung 4.3: Konstante Einstellung der Richtcharakteristik für die Aufnahme des Stuttgarter "New Thing Trio" *54*

Abbildung 4.4: Frequenzabhängige Einstellung der Richtcharakteristik für die Aufnahme des Stuttgarter "New Thing Trio"..... *55*