Brandenburgische Technische Universität
Cottbus - Senftenberg

Lehrstuhl für Kommunikationstechnik
Fakultät 3

Bachelorarbeit

über das Thema

Sprecherlokalisierung mit einem 3D-Mikrofonfeld

Autor: Martin Birth
Matrikel-Nr.: 2934203
E-Mail: martin.birth@gmx.de

Prüfer: Prof. Dr.-Ing. habil. Matthias Wolff

Abgabedatum: 15. Oktober 2013

Brandenburgische
Technische Universität
Cottbus - Senftenberg

Lehrstuhl
Kommunikationstechnik

I Kurzfassung

Nicht direkt mit einem technisches Gerät verbunden zu sein, ist einer der erstrebsamsten Ziele der akustischen Quellenortung. Es fördert die naturgemäße Kommunikation zwischen Mensch und Maschine und erweitert den Bewegungsradius. Aus diesem Grund beschäftigt sich diese Bachelorarbeit mit der Lokalisierung von Sprechern in einem freien Raum. Für die Umsetzung wird das neugestaltete „SpeechLab“ des Lehrstuhls für Kommunikationstechnik zum Einsatz gebracht. Es beinhaltet zwei orthogonal zueinander angebrachte Mikrofonfelder mit jeweils 32 Mikrofonen. Die auch „Mikrofonarrays“ genannten Apparaturen sollen dazu genutzt werden, um mittels zeitlicher Verzögerung und Summierung der unterschiedlichen Audiosignale, eine Richtwirkung auf einen spezifischen Punkt zu erwirken. Solch eine Vorgehensweise wird als Delay-and-Sum-Beamforming bezeichnet. Durch die passive Auslenkung erfolgt eine Verstärkung der Signale in „Blickrichtung“, unter gleichzeitiger Verringerung der nebenstehenden Geräusche. Die genaue Ortung wird durch eine Abtastung von verschiedenen Punkten realisiert, die in einem fest definierten Raumkoordinatensystem bestimmt wurden.
Anschließend an die theoretischen Vorbetrachtungen wird eine eigene Implementation eines Java-basierten Audiointerfaces vorgestellt. Dieses stellt die Grundlage dar, für die Realisierung des Beamformers und des daraus resultierenden Suchalgorithmus zur Raumabtastung. Die grafische Visualisierung erfolgte schlussendlich in einer dreidimensionalen Benutzerschnittstelle, welche das „SpeechLab“ in dessen Aufbau repräsentieren soll.

Abstract

The localisation of an acoustic source is one of the most significant purposes to reduce the use of technical equipment. This furthers the natural communication between man and machine and expands their range of motion. For this reason these bachelor thesis deal with the localisation of a speaker in a free space. The redesigned „SpeechLab“ of the Chair of Communications Engineering is been used for the implementation. It includes two orthogonally mounted microphone arrays with 32 microphones. This equipment obtains a specific directivity by the use of delay and summation of the individual signals. This procedure is known as a delay-and-sum beamforming. The microhone array has a „viewing direction“, because of the passive deflection of the audio signals. There happens a simultaneous reduction in the noise field. The actual location is then performed by a sampling of different points in space, that is defined in a coordinate system.
After the theoretical considerations an own implentation will be presented on a Java based audio interface class. This provides the basis for the realization of the beamformer and the resulting search algorithm for space scanning. Finally there is a representation of a three-dimensional user interface, which is intended to represent the „SpeechLab”“ in the structure of the graphical visualization.

II Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Programmcode-Verzeichnis

V Tabellenverzeichnis

VI Abkürzungsverzeichnis

API Application Programming Interface
ASIO Audio Stream Input/Output
DSB Delay-and-Sum-Beamformer
dB Dezibel
dB FS Dezibel Full Scale
DLL Dynamic Link Library
GUI Graphical User Interface
IDE Integrated Development Environment
LCARS Library Computer Access and Retrieval System
RMS Root Mean Square (engl.)
SNR Signal to Noise Ratio
SPL Sound Pressure Level

1 Einleitung

In dem Informationsaustausch zwischen Mensch und Maschine erfolgt immerwährend eine Verbesserung der Interaktion. Es ist für uns zur Gewohnheit geworden, das Mikrofon in die Hand zu nehmen, es als Headset zu tragen oder gar das Handy an das Ohr zu halten. Ungeachtet dessen, ist es durch den direkten Kontakt mit der Technik, keine rein natürliche Kommunikation. Die Entwicklung hat indessen Fortschritte gemacht und so gibt nun Möglichkeiten solche technischen Geräte komplett entfallen zu lassen. Die akustische Quellenortung und die damit angeschlossene Verarbeitung von Audiosignalen soll hierfür eine große Rolle spielen.

Als Ausgangspunkt ist für uns der Schall ein grundlegender Informationsträger. Obgleich Sprache, Musik oder die normalen Geräusche des Alltages, der Mensch ist so konzipiert, dass er durch das Gehör, sowie der neuralen Verarbeitung, aus dem Schall Informationen herausfiltern kann. Auch wenn neben dem erwünschten Nutzschall Geräusche dabei sind, die eher störend auf uns wirken. Ein Individuum kann sich trotzdem ausgezeichnet auf eine bestimmte Schallquelle konzentrieren und so andere Störquellen ausblenden. Dieses selektive Hören wird als *Cocktailparty-Effekt* bezeichnet. Der Mensch kann infolgedessen ohne weitere Anstrengungen so eine Fähigkeit nutzen, um die Richtung und Entfernung zur Geräuschquelle zu bestimmen. Teile der Tierwelt können darüber hinaus sogar die Ohren direkt auf die Schallquellen ausrichten. Wie lässt sich so etwas jedoch technisch lösen?

Mit nur einem Mikrofon ist dies nicht so einfach zu realisieren. Aufgrund dessen wurden Mikrofonfelder entwickelt und sind schon mehr als 20 Jahre ein Schwerpunkt in der nachrichtentechnischen Forschung. Dabei bestehen solche Felder aus einer Vielzahl an Mikrofonen, die durch verschiedene Methoden synchronisiert werden. Durch die Signalkopplung erreicht man, dass deren Ausrichtung sich auf einen zentralen Punkt konzentriert. Dieses Verfahren wird für unzählige Anwendungsfälle genutzt. Zum Beispiel lässt sich so ein Sprecher in einem Fernsehstudio lokalisieren und die Kamera richtet sich automatisch auf ihn aus. Telefonfreisprecheinrichtungen, wenn mehrere Personen in einem Kraftfahrtzeug sitzen, können direkt den derzeitigen Sprecher anvisieren. Ein weiterer Anwendungsfall findet sich in der Industrie, wo mittels akustischer Felder die Ortung von Störquellen an Maschinen immer mehr an Bedeutung gewinnt. So können vibrierende Teile an Motoren schnell lokalisiert werden. Als letztes Beispiel wird Hintergrundrauschen vom einem Sprecher separiert, um so ein leistungsfähigeres Signal zu erhalten.

Ein ähnliches Szenario will der Lehrstuhl für Kommunikationstechnik um Prof. Dr.-Ing. habil. Matthias Wolff realisieren. Als auditives Teilprojekt soll im Rahmen dieser Bachelorarbeit eine automatische Sprecherlokalisation mit einem 3D-Mikrofonfeld entwickelt werden. Mittels eines Mikrofonfeldes soll ein Sprecher im Raum erkannt und alle Mikrofone automatisch auf ihn ausrichten werden. Die Verarbeitung der Schallquellen soll dabei in Echtzeit erfolgen. Das Ausgangssignal kann im Anschluss daran für weitere Projekte, wie die Verarbeitung in einem Spracherkenner verwendet werden.

Für die Durchführung des Projektes wurden die Räumlichkeiten des „SpeechLab“, der Brandenburgischen Technischen Universität Cottbus-Senftenberg, zur Verfügung gestellt. Das neu eingerichtete Labor umfasst zwei Sensorfelder mit jeweils 32 Mikrofonen, die orthogonal zueinander angebracht wurden (Abb. 1). Das Mikrofonfeld 1 ist um einen 72 Zoll Multitouchscreen[1] angebracht. Für das Mikrofonfel 2 wurde eine auf Rollen gelagerte Deckenkonstruktion gewählt. Wodurch es möglich ist die vertikale Position, unter Zuhilfenahme einen Schrittmotors, beliebig zu verändern.

Abbildung 1: SpeechLab-Labor der BTU Cottbus

Die vorrangigste Aufgabe ist die Entwicklung einer Audioschnittstelle, die alle Eingangssignale der Mikrofonfelder miteinander verbindet und somit ein Hardwareinterface für bereits vorhandene Softwareprojekte zur Verfügung stellt.

Die Ausrichtung der Mikrofone soll unter Zuhilfenahme eines „Delay-and-Sum-Beamformers“ (dt. Verzögerungs- und Summiernugsstrahlformer) erfolgen. Er ermöglicht es, das Feld auf einen beliebigen dreidimensionalen Punkt auszurichten, ohne auf dessen physikalische Eigenschaften Einfluss nehmen zu müssen. Das gesamte Konzept des „SpeechLab“ wurde von Dipl.-Ing. Thomas Fehér (TU Dresden) konzipiert, welcher sich mit der akustischen Quellenortung und Mikrofonarrays/Beamforming bereits ausgiebig in seiner Forschung beschäftigt hat. Die von ihm angestellten Berechnungen und Entwürfe stellen die Grundlage für die Anschaffungen der Mikrofonfelder und der Implementation der Algorithmen.

[1] Berührungsempfinderlicher Bildschirm

Die angeschafften Apparaturen sollen dazu genutzt werden, einen Sprecher an einer gewissen Position lokalisieren zu können. Dabei wird der zu messende Raum nacheinander abgetastet und der Punkt mit der höchsten schallabstrahlenden Energie als Sprecherposition genommen. Eine parallele Verarbeitung des Programm ist folglich realisierbar. Einerseits läuft eine fortwährende Raumabtastung und andererseits ist die Ausrichtung der Mikrofone auf die erkannte Sprecherposition möglich. Diese zwei unterschiedlichen Verarbeitungprozeduren von Audiodaten sind ein großer Vorteil von Beamformern. Die passive Ausrichtung der Mikrofone ist dafür nicht nur der alleinige Grund. Vielmehr erfolgt die gesamte audiotechnische Verarbeitung rein digital.

Alle notwendigen Implementierungsarbeiten erfolgen in dem bereits vorhandenen Java-„SpeechLab“-Projekt. Es beinhaltet unterschiedliche Hardware-, Sprach- und Audiokomponenten, die zum Steuern des Sprachlabors notwendig sind. Darüber hinaus verfügt es über eine visuellen Benutzerschnittstelle, die im Design der StarTrek^{TM}-Filme gehalten ist. Im Rahmen der erstellten Bachelorarbeit wird SpeechLab-Softwareprojekt um eine neue Komponente erweitert und ebnet somit Weg für viele neue interessante Funktionen.

2 Grundlagen

In der Ausarbeitung sollen die Eigenschaften von Schall und deren Verarbeitung nur eine untergeordnete Rolle spielen. Jedoch sind einige Basiselemente für das weitere Verständnis von Mikrofonen, Richtcharakteristiken und Beamformern notwendig. Aus diesem Grund erfolgen in folgenden Kapitel alle physikalischen und mathematischen Grundlagen, die für die Implementation der akustischen Lokalisierung notwendig sind.

2.1 Schall und Schallausbreitung

Der Schall ist immer der Ausgangspunkt bei der Verarbeitung von akustischen Signalen. Dabei wird er allgemein als erzwungene Schwingung flexibler Stoffe definiert. Benachbarte Moleküle eines Kommunikationsträgers werden durch äußere Krafteinwirkung so aus der Ruhelage gebracht, dass sich der Bewegungsimpuls, bei ausreichender Energiezufuhr, durch das Medium fortsetzt. So entsteht bei entsprechender Anregung des Systems eine Longitudinalwelle[2], die sich im Raum abhängig von der Zeit ausbreitet. Unterschieden werden drei uns bekannte Arten von Schall: Der Körperschall in festen Stoffen, den Flüssigkeitsschall in Flüssigkeiten und den Luftschall in Gasen. Ausschlaggebend ist für uns Letzterer, der für die Sprache und dessen Ausbreitung in Räumen verantwortlich ist.
Eine harmonische, lineare Schwingung, oder auch Welle, wird durch die Formel

$$a(t,x) = a_0 \cdot cos(\omega t - kx) \tag{2.1.1}$$

charakterisiert. Dabei beschreiben die Variablen x die Abhängigkeit vom Ort und t von der Zeit. Die Kreisfrequenz wird angrenzend mit $\omega = 2\pi f = 2\pi/T$ beschrieben, wobei T die Periodendauer, folglich der Abstand zwischen zwei gleichen Schwingungszuständen ist. Die Wellenanzahl k wird über den räumlichen Abstand, folglich der Wellenlänge λ (Lambda) dargelegt und mit $k = 2\pi/\lambda$ bestimmt. Die sich aus der Bindung des räumlichen und zeitlichen Verlaufs resultierende Ausbreitungsgeschwindigkeit c wird als Phasengeschwindigkeit bezeichnet. Für Luftschall wird diese als Schallgeschwindigkeit angegeben.
Die Geschwindigkeit ist immer abhängig vom Medium, welches sich durch den Druck p und die Dichte ρ (Rho), folglich auch der Temperatur, auszeichnet. Die Schallgeschwindigkeit in einem idealen Gas berechnet sich demnach mit

$$c_{ideal} = \sqrt{\kappa \frac{p}{\rho}} = \sqrt{\kappa \frac{RT}{M}}. \tag{2.1.2}$$

In der Formel beschreibt der Adiabatenexponent κ (Kappa) die spezifische Wärmekapazität der Luft ($\kappa_{Luft} = 1,4$), R entspricht der universellen Gaskonstante ($R_{Luft} = 8,3145 J/molK$) und die mittlere Molare Masse, für Stickstoff und Sauerstoff, wird mit M angegeben ($M = 0,02896 kg/mol$). Unter der Voraussetzung, dass R, κ und M sich im idealen Gas konstant verhalten, hängt die Schallgeschwindigkeit nur noch von der Temperatur ab und ist des Weiteren

[2] In Ausbreitungsrichtung schwingende physikalische Welle

auch nicht von Druck und Dichte des Gases abhängig. Wird von einer normalen Raumtemperatur ($T = 20°C = 293,15K$) ausgegangen und setzt alle Werte in die Gleichung (2.1.2) ein, erhalten wir für die Luftschallgeschwindigkeit folgenden Wert:

$$c_{ideal} \approx \sqrt{1,4 \cdot \frac{8,3145\frac{J}{molK} \cdot 293,15K}{0,02896\frac{kg}{mol}}} = 343,26\frac{m}{s}. \tag{2.1.3}$$

In einer reale Umgebung wird davon ausgegangen, dass die Luft durch Schmutzpartikel und Wassermoleküle (Luftfeuchtigkeit) verunreinigt ist. Das führt zum Ergebnis, dass die Schallgeschwindigkeit auf

$$c_s = 343\frac{m}{s} \tag{2.1.4}$$

gerundet wird.

Von einer punktuellen Schallquelle breitet sich der Schall im offenen Raum näherungsweise gleichmäßig und kugelförmig aus. Auch der Mensch, wenn er eine gewissen Entfernung zum Schallwandler besitzt, gehört zu dieser Kategorie an Schallquellen. Durch die weite Entfernung nimmt die Schallintensität (Leistung) und auch der Schalldruck zunehmend ab.

2.2 Effektivwert

Der Effektivwert, auch RMS (engl. root mean square) genannt, ist die wichtigste Einheit einer Schwingung. Er entspricht dem Energiegehalt und „deren mittleren Amplitude des gleichgerichteten Schwingungsverlaufs“ ([Gö08], S.24). Gebildet wird er als quadratischer Mittelwert von der Amplitude ξ (Xi), einer sich zeitlich verändernden physikalischen Größe. Zur Bestimmung des Effektivwertes wird die Quadratwurzel von allen quadrierten und summierten Werten eines Intervalls gebildet. Daraus ergibt sich die Formel:

$$\xi_{eff}(t) = \sqrt{\frac{1}{T}\int_0^T \xi^2(t)dt}. \tag{2.2.1}$$

Erfolgt die Verarbeitung von n Werten immer in einer Zeit T, erhält man näherungsweise

$$\xi_{eff} \approx \sqrt{\frac{1}{T}\sum_{i=1}^{n} \xi_i^2 \Delta T_i} = \sqrt{\frac{1}{T}(\xi_1^2\Delta T_1 + \xi_2^2\Delta T_2 + .. + \xi_n^2\Delta T_n)}. \tag{2.2.2}$$

Für konstante Perioden Δt mit $T = n \cdot \Delta t$ lässt sich die Gleichung (2.2.2) weiter vereinfachen:

$$\xi_{eff} \approx \sqrt{\frac{1}{n}\sum_{i=1}^{n} \xi_i^2} = \sqrt{\frac{1}{n}(\xi_1^2 + \xi_2^2 + ... + \xi_n^2)}. \tag{2.2.3}$$

Mit der Vereinfachung (2.2.3) kann folgend die Bestimmung des quadratischen Mittelwerts einer Schwingung und so deren effektiven Energiegehalt bestimmt werden.

2.3 Akustische und elektrische Pegel

Durch die Vorannahme, dass alle subjektiven Sinnesreize, von der empfundenen Stärke her, sich verhältnisgleich zum Logarithmus der objektiven Intensität der physikalischen Reize verhalten, eignet sich die logarithmische Darstellung optimal für den akustischen Pegel. Diese Gesetzmäßigkeit des Reizverhältnisses ist auf das Weber-Fechner-Gesetz zurückzuführen ([Gö08] S.31). Die näherungsweise Reproduktion einer Sinneswahrnehmung wird demnach durch das Verhältnis von zwei Reizen bestimmt und aus deren Quotient der Pegel (engl. Level) errechnet.

Als gebräuchliches Maß für die Stärke des Schalls, wird der Leistungspegel L_p, auch Schalldruckpegel (SPL) genannt, verwendet. Für die Berechnung wird der Effektivwert des Schalldrucks zu einem definierten Bezugswert ins Verhältnis gesetzt:

$$\begin{aligned} L_p &= 10 \log \left(\frac{p}{p_0} \right)^2 dB_{SPL} \\ &= 20 \log \frac{p}{p_0} dB_{SPL}. \end{aligned} \tag{2.3.1}$$

Der Referenzwert beträgt $p_0 = 2 \cdot 10^{-5} Pa$ und entspricht etwa dem kleinsten für den Menschen zu unterscheiden Schalldruck. Da es durch den Quotienten zu einer Auslöschung der Einheiten kommen würde, kam es zur Einführung der „Pseudo-Einheit“ Bel (zu Ehren von Alexander Graham Bell). Eine Angabe erfolgt jedoch in Dezibel [dB], da es sich um einen zehntel (dezi) Teil des Bels handelt.

In Analogie zum Schalldruckpegel verhält sich auch der Spannungspegel. Er wird korrespondierend mit

$$L_u = 20 \log \frac{u}{u_0} dB_u \tag{2.3.2}$$

berechnet. Für den deutschen Studiostandard ist hier eine Referenzspannung von $u_0 = 0,775V$ vorgesehen, ohne Angabe einer zugehörigen Impedanz. Dieser Bezugswert ist einer historischen Entwicklung zu schulden, bei der aus der leistungsbezogenden Telefontechnik die 0dB als $1mW$ an 600Ω definiert und folgender Wert ermittelt wurde:

$$U = \sqrt{P \cdot R} = \sqrt{0,001W \cdot 600\Omega} = 0,7746V. \tag{2.3.3}$$

Ein weiterer Pegel ist der digitale Pegel dB FS (engl. Dezibel full scale). Er wird überwiegend in digitalen tontechnischen Systemen verwendet. Da ein analoges Signal sowohl negative, als auch positive Wertebereiche hat, wird durch einen Gleichrichter alles auf den negativen Bereich ausgerichtet. Dabei wird die Bewertung des Pegels relativ zur maximalen Aussteuerbarkeit vorgenommen, die bei 0 dB FS liegt. Die Darstellung der dB FS-Skala erfolgt über die digitalisieren Spannungswerte.

Die Pegelwerte von dB FS haben jedoch einen entschiedenen Nachteil: Durch die Digitalisierung können die realen Peak-Werte über die Skala von 0 dB FS hinaus gehen. Beim Analogisieren tritt dieser Effekt durch einen D/A-Wandler auf. Infolgedessen entsteht das „Clipping“. Dabei überschreiten die Maximalwerte der Sinuswelle den Wert für die Vollaussteuerung.

2.4 Sampling

Um Geräusche, Klänge oder Töne aufzunehmen, muss man sich bewusst sein, dass es sich um analoge Signale handelt, die im Zeit- und Wertebereich kontinuierlich sind. Das Speichern einer Quelle benötigt jedoch diskrete Werte. Dies wird durch eine Digitalisierung erzielt. Dabei werden die analogen Werte in diskrete binäre Stufen transformiert. Ein Audiosample ist demnach nur ein Teil eines digitalisierten analogen Signales.
Das Digitalisieren von Audiosignalen geschieht in mehreren Stufen. In erster Instanz wird eine Tiefpassfilterung der analogen Quelle eingesetzt, um Artefakte, in Form von störenden Frequenzanteilen (Alias-Signal), zu vermeiden. Danach erfolgt eine zeitliche, konstante Abtastung unter zur Hilfenahme des Shannon/Nyquist–Abtasttheorems, was besagt das die Abtastfrequenz immer doppelt so groß sein muss, wie die maximale Frequenz des Signals. Angesichts dessen ist eine spätere Rekonstruktion des Signales erst möglich. Die gebräuchlichste Abtastfrequenz ist die $44,1 kHz$ und garantiert Frequenzen des Signals bis $22,05 kHz$.

Als zweiten Schritt wird das abgetastete Signal quantisiert. Dabei wird der Wertebereich in N-diskrete Stufen aufgeteilt. Bei einer Quantisierung von 16-Bit würde dies $2^{16} = 65636$ Stufen bedeuten.

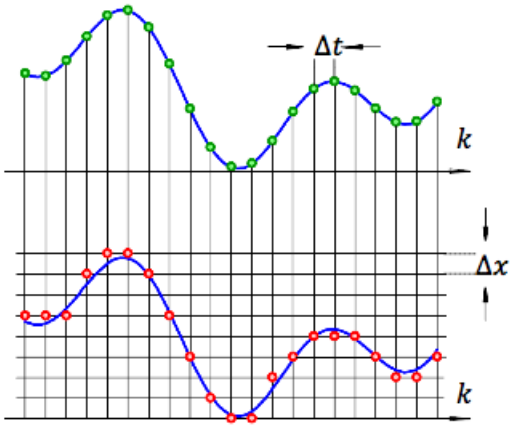


Abbildung 2: Abtastung und Quantisierung[3]

Jeder Abtastzeitpunkt des Audiomaterials kann so einem diskreten binären Wert zugeordnet werden. Problembehaftet ist jedoch dabei die Wahl der Quantisierungsstufen, wodurch es zu Verzerrungen (Quantisierungsrauschen) kommt und des Weiteren ein höherer Quantisierungsfehler entsteht. Um diese Fehler möglichst gering zu halten, ist von der Wahl von zu großen Stufen abzuraten.

[3] Quelle: [Lin11]

3 Mikrofone

Um Schallwellen in digitale Signale zu verwandeln, bedarf es eines Schallwanlders. Mittels solch einem elektroakustischen Wandler wird so der Schall in ein elektrisches Signal umgewandelt. Das geschieht jedoch meist indirekt durch eine akustisch-mechanische Wandlung, wobei der Schall eine mechanische Bewegung auslöst. Diese kinetische Veränderung kann daraufhin in eine elektrische Ausgangsgröße transformiert werden. Die am Ausgang des Luftschallmikrofons ausgegeben Spannung ist proportional dem Schalldruckverlauf. Für die physikalische Umsetzung haben sich zwei grundlegende Wandlertypen durchgesetzt: Dynamische Mikrofone und Kondensatormikrofone.

In dem ersten Fall sind die Schallwandler so konzipiert, dass ein Leiter in einem Magnetfeld bewegt wird und sich so eine Spannungsindizierung ergibt. Die Ausgangsspannung steht hierbei in Abhängigkeit von der magnetischen Flussdichte, der Membranschnelle und der Länge des Leiters. Ein entschiedener Vorteil der elektrodynamischen Mikrofone ist, dass sie keine Versorgungsspannung benötigen und gegen mechanische Einflüsse durch ihre Bauweise resistent sind. Nachteilig ist hingegen, dass durch die hohe Masse von Spule und Membran es zu hoher Ausbreitung von Schall im Korpus kommt. Dieser Körperschall sorgt für eine hohe Resonanz im Inneren und wirkt somit störend auf die Übertragungsfunktion, die im Nachgang eine Glättung mittels Tiefpass erfordert.

Für den Einsatz im Studiobereich hat sich daher die zweite Bauart, die Kondensatormikrofone bzw. elektrostatischen Wandler durchgesetzt. Diese brillieren von Natur aus durch einen linearen Frequenzgang.

3.1 Kondensatormikrofone

Kapazitive Mikrofone bilden mit einer elastischen und elektrisch leitenden Membran eine Elektrode eines Plattenkondensators. Die Gegenelektrode befindet sich fest montiert parallel dazu. Durch einfallenden Schall kommt es zur Auslenkung der Membran und somit zur Veränderung der Kapazität, die Einfluss auf das Ausgangssignal nimmt.
Die physikalischen Gesetzmäßigkeiten eines Kondensators mit den folgenden Formeln beschrieben:

$$U = \frac{Q}{C} \tag{3.1.1}$$

und

$$C = \epsilon_0 \epsilon_r \frac{A}{d} \tag{3.1.2}$$

und durch einsetzen von (3.1.2) in (3.1.1), ergibt sich

$$U = \frac{Q \cdot d}{\epsilon_0 \epsilon_r A}. \tag{3.1.3}$$

Hierbei sind ϵ_0 und ϵ_r Konstanten, die das Dielektrikum[4] zwischen den beiden Kondensatorplatten beschreiben. Ein weiterer Kondensator wird mit der üblichen Phantomspeisung von $U_0 = +48V$ versorgt und erzielt somit die Bereitstellung von der Ladung Q. Durch eine gleichbleibende Form der Kondensatoren bleibt die Oberfläche A ebenso konstant. Folglich wird klar, dass die Ausgangsspannung des Kondensators nur noch von dem Abstand d zwischen den beiden Platten abhängt. Dieser wird durch den einfallenden Schalldruck auf die bewegliche Membran beeinflusst. Im Vergleich zu dynamischen Schallwandlern erzielen Kondensatormikrofone eine höhere Empfindlichkeit und haben wegen ihrer kompakten Bauform auch einen linearen Frequenzgang.

3.2 Elektret-Kondensatormikrofon

Eine besondere Bauart der elektrostatischen Wandler stellen die Elektret-Kondensator- Mikrofone dar. Bei ihnen besteht die verwendete Membran aus einem permanent elektrisch geladenem Material. Elektretfolien bestehen aus einem Werkstoff, der als „Teflon“ bekannt ist, genauer gesagt Polytetrafluorethylen. Für die Herstellung wird eine Kunststofffolie mit Elektronen beschossen und in ihr dauerhaft gebunden. Diese wird als Membran auf die Kapsel aufgebracht. Durch eine elektrostatische Induktion entsteht eine Spannung, bekannt auch als Wandlervorspannung. Somit lässt sich die Polarisationsspannung des Kondensators auf mehr als $100V$ erhöhen, was zu einer höheren Empfindlichkeit der Mikrofonkapsel führt ([Gö08] S.252).

Die geringe Größe von Kondensatormikrofonen machen sie optimal für den Bau von Mikrofonfeldern und finden daher auch Einsatz im SpeechLab. Hierfür wurden jedoch Back-Elektret-Wandler verwendet, bei denen die Gegenelektrode mit der Elektretfolie beschichtet ist. In unserem Fall des SpeechLabs kommen die *MM1* Messmikrofone zum Einsatz, die sich durch einen linearen Frequenzgang und einer Kugelcharakteristik auszeichnen (Datenblatt im Anhang A).

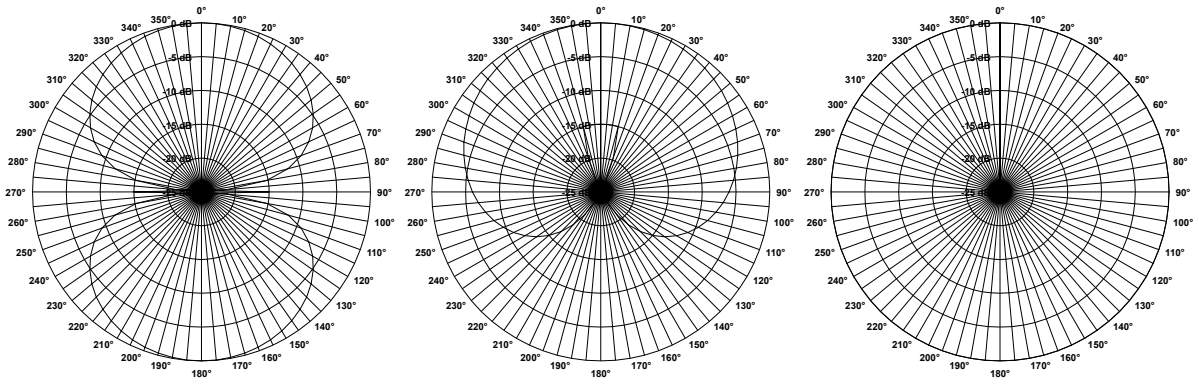
[4] nichtleitende, nichtmetallische Substanz

3.3 Mikrofonrichtcharakteristiken

Um einen Schallwandler eindeutig zu kennzeichnen reicht meistens der Frequenzgang alleine nicht aus, da die Empfindlichkeit der Schallaufnahme für jedes Mikrofon immer in Abhängigkeit vom Einfallswinkel ϕ und der Frequenz f steht. Der Winkel mit $\phi = 0°$ wird als Frontalrichtung benannt. Der Effektivwert für einen bestimmten Winkel $u(\phi)$ im Verhältnis mit dem für die Frontalrichtung, ergibt das Richtfaktor mit der Formel:

$$\Gamma(\phi) = \frac{u(\phi)}{u(0°)}. \tag{3.3.1}$$

Aus einer Erweiterung des Richtfaktors um die Frequenz, ergibt sich die entsprechende Richtfunktion $\psi(f, \phi)$. Die Ermittlung der jeweiligen Leistungswerte für eine Richtung erfolgt durch die Iteration über alle Winkel und Frequenzen. Diese Daten werden anschließend in einem Polardiagramm erfasst, was allgemein als Richtcharakteristik bekannt ist. Drei Arten von Charakteristiken haben sich hier durchgesetzt: Acht, Niere und Kugel (3).



(a) Acht (b) Niere (c) Kugel

Abbildung 3: Beispiel-Richtcharakteristiken von Mikrofonen[5]

[5]Erstellt mit: `http://www.sengpielaudio.com/EBS-AlleCharLinLog.xls`

3.4 Mikrofonfelder

Werden mehrere Sensoren zusammengefasst und synchronisiert, spricht man allgemein von einem Sensorarray (dt. Sensorenfeld). Diese Felder treten in unterschiedlichsten Dimensionen und Anordnungen auf. So können zum Beispiel mehrere Teleskope, Antennen oder, wie für unseren Fall, Mikrofone parallel betrieben werden. Eine nachstehende Signalverarbeitung konzentriert die verschiedenen Signale und verstärkt bzw. beeinflusst deren Richteigenschaften. Grundlegend vergleicht man die Arbeitsweise so mit einem Holspiegel, bei der die Schallquelle der Brennpunkt ist ([Sar04]). Wie die so entstandene Richtkeule (engl. beam) genau geformt wird, ist in den Kapiteln 5 und 5.1 zum Delay-and-Sum-Beamforming genau ausgeführt. Mikrofonarrays[6], lassen sich ein-, zwei- oder dreidimensional anordnen. Dabei hat die räumliche Anordnung einen entschiedenen Vorteil. So lässt sich jeder beliebiger Punkt im Raum anvisiert. Hingegen ist es mit anderen Arraykompositionen nur möglich in einem von ihnen ausgehenden Halbkreis die Ortung durchzuführen.

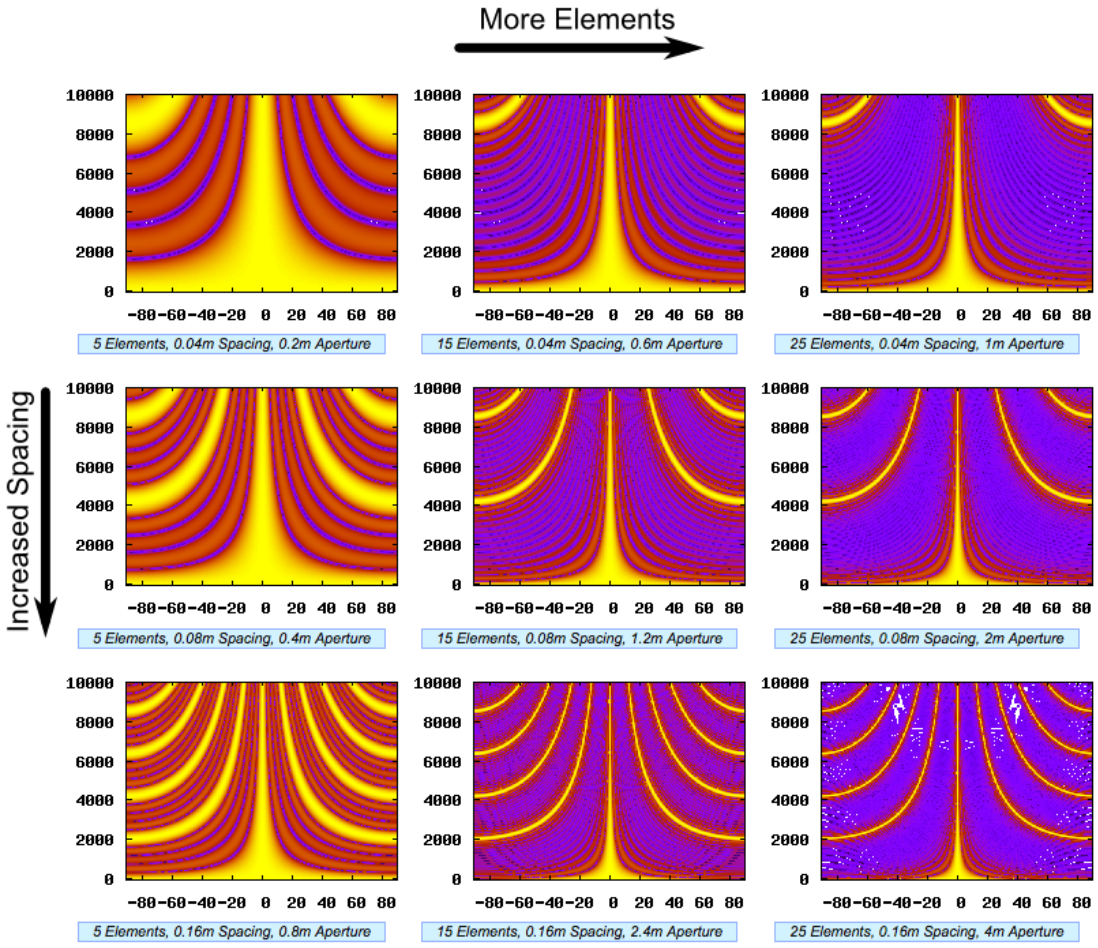


Abbildung 4: Winkel (x in Grad) und Frequenz (y in Hz) in Abhängigkeit von Mikrofonabständen und -mengen[7]

[6] In der Literatur meist „Mikrofonarray“ anstatt „Microphone-array“.
[7] Quelle: `http://www.labbookpages.co.uk/audio/beamforming/delaySum.html`

Wie in 3.3 beschrieben, hängt die Richtcharakteristik eines einzelnen Mikrofons von der Frequenz und dem Winkel ab. Für die Gesamtheit des Feldes kommen nun noch der Abstand und die Anzahl der Schallwandler hinzu. Beide Faktoren haben hohen Einfluss auf die Charakteristik, wie in Abbildung 4 ersichtlich ist. Mit zunehmender Anzahl an Elementen verdichtet sich die Hauptkeule, wobei gleichzeitig die Maxima der Nebenkeulen geringer werden. Wohingegen durch eine Vergrößerung des Abstands der Mikorfone es dazu kommt, dass die Nebenkeulenmaxima sich mehr ausprägen und dichter an die Hauptkeule stoßen ([Gre12]).

Für die Richtwirkung des Arrays ist daher eine ausgeglichene Anordnung von Vorteil. Für das SpeechLab wurden zwei verschiedene Mikrofonarrays entwickelt, die orthogonal zueinander angebracht wurden. Das erste Mikrofonfeld wurde um einen 72 Zoll Bildschirm herum angeordnet und besteht aus vier Gruppen, die sich wiederum aus Arraykompositionen ergeben (Abb. 5).

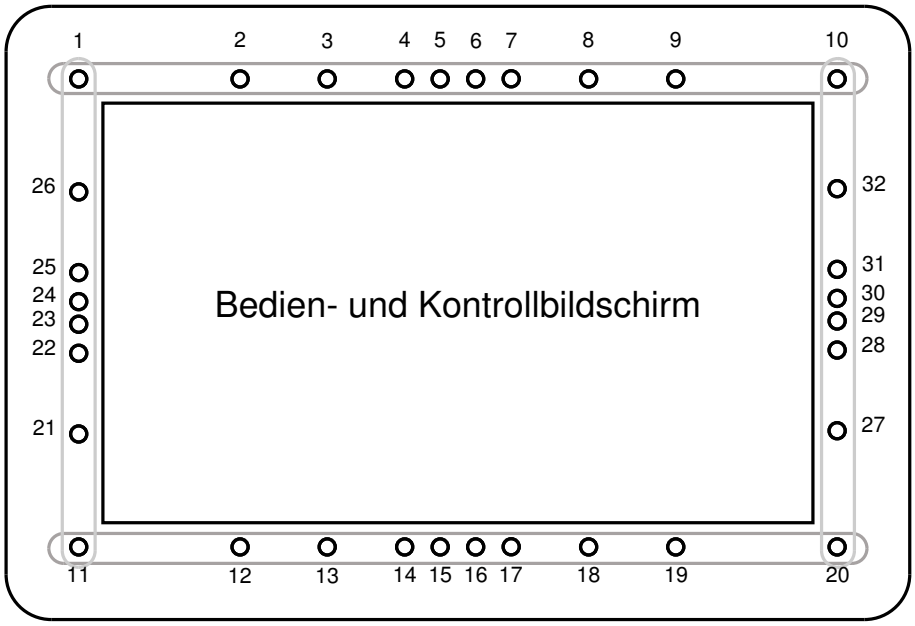
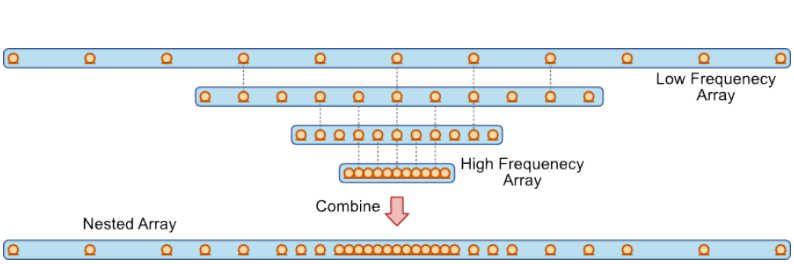
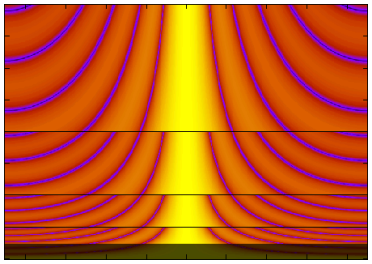


Abbildung 5: Bildschirm Array

Die einzelnen linearen Arrays besitzen per se einen gleichmäßigen Abstand, werden jedoch durch Überlagerung zu einem Mikrofonfeld zusammengefasst (Abb. 6). Grund dafür ist, dass mit steigendem Abstand der Mikrofone die Hauptkeule breiter wird, auf der Basis einer kleineren Frequenz. Das so entstandene Array behält durch die Überlagerung eine konstante Richtkeule, die zum Ende hin nicht wesentlich breiter wird ([Gre12]).



(a) Überlagerung linearer Arrays



(b) überlagerte Richtcharakterristik

Abbildung 6: Bsp. Arraykomposition (29 Sensoren, f=0..8kHz, Winkel -90 bis 90 Grad)[8]

Für das zweite, an der Decke hängende Mikrofonarray wurde eine andere Komposition gewählt. Die Wahl viel dabei auf eine spiralförmige Anordnung, deren Empfindlichkeit, in Abhängigkeit vom Winkel und der Frequenz, wesentlich höher ist. Die Intensität nimmt hierbei für die hohen Frequenzen deutlich zu. Begründet ist das durch die dichtere Anordnung der Mikrofone, die im Gegenzug zu dem anderen Array keinen Bildschirm im Zentrum besitzen. Ersichtlich wird dies auch auf dem Datenblatt im Anhang D.

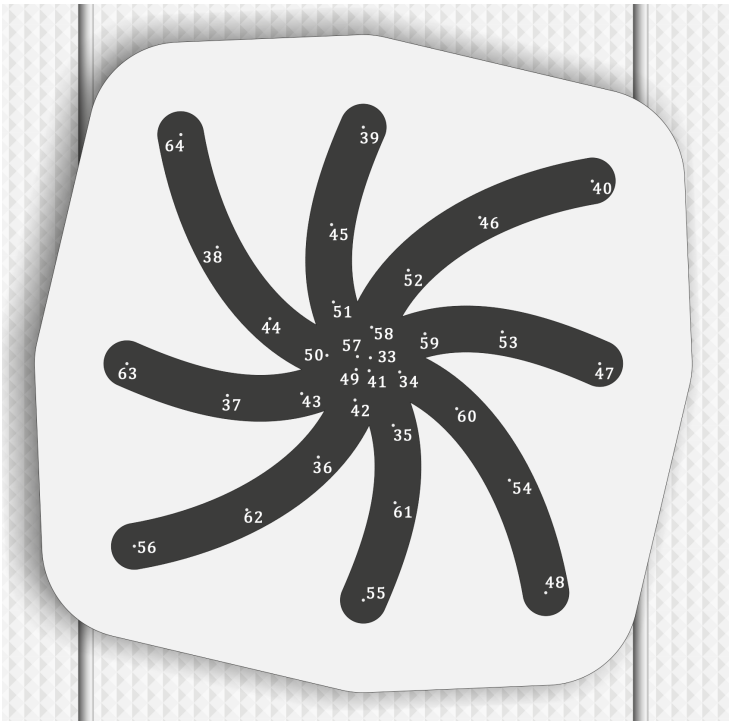


Abbildung 7: Deckenarray Array

[8]Quelle: `http://www.labbookpages.co.uk/audio/beamforming/composite.html`

3.5 Richtcharakteristik von Mikrofonfeldern

Jedes Mikrofonfeld besitzt nach seiner Ausrichtung eine bestimmte Abstrahlcharakteristik (engl. beam pattern). Diese entsteht durch die Wichtung der einzelnen Summanden im DSB-Algorithmuses. Der Beamformer formt somit für ihn charakteristische Keulen. Gekennzeichnet wird die Abstrahlcharakteristik durch die Hauptkeule (engl. main lobe) und mehreren Nebenkeulen (engl. side lobes), die durch Nullstellen voneinander getrennt sind (Abb. 8). Die „Blickrichtung“ der ausgerichteten Beamformer wird durch die Richtung der Hauptkeule definiert und hat an ihrer Spitze die maximale Empfindlichkeit.

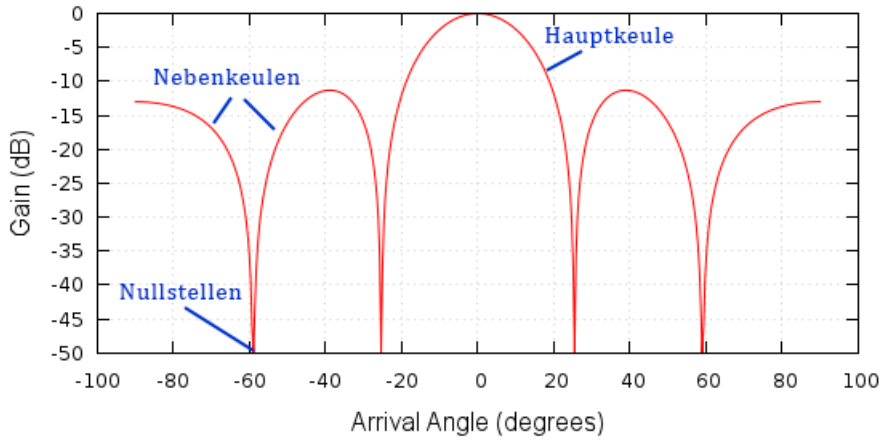


Abbildung 8: Beispiel Array-Richtcharakteristik (4 Sensorn, 0,2m Abstand, 1kHz)[9]

Zwei wichtige Faktoren begleiten die Charakteristik der Ausrichtung. Die Breite der Hauptkeule nimmt bei hohen Frequenzen zu, wobei gleichzeitig die Anzahl der Nebenkeulen steigt. Demnach ist die Richtcharakteristik frequenzabhängig (Vgl.[Dre99] S. 36). Aber auch die Anzahl der Mikrofonelemente spielt hierfür eine wichtige Rolle, wie bereits in Abschnitt 3.4 gezeigt.

Die genauen theoretischen Betrachtungen zur Berechnung der Mikrofonfeldrichtcharakterristik sind im Abschnitt 5.2 beschrieben.

[9] Quelle: `http://www.labbookpages.co.uk/audio/beamforming/delaySum.html`

4 Aufbau SpeechLab

Das SpeechLab, wie es in Abbildung 1 dargestellt ist, funktioniert nur, weil eine Reihe von Hardware- und Softwarekomponenten optimal aufeinander abgestimmt sind. Um zu verstehen, wie die Verarbeitungskette der Audiosignale aufgebaut ist, werden hier kurz alle zugehörigen Bestandteile vorgestellt.

4.1 Hardwarekomponenten

Hardwareseitig besteht das gesamte Mikrofonfeld aus zwei einzelnen Arrays, deren Komposition bereits auf Seite 11 vorgestellt wurden. Als Schallwandler wurden 64 *MM1* Messmikrofone der Firma *Beyerdynamic* verwendet, die in den Arrayplatten eingelassen wurden. Dadurch verändert sich die Kugelcharakteristik (siehe Datenblatt A) zu einer Halbkugel. Jedoch kommt es gleichzeitig zu einem angenommenen Anstieg von $+3dB$ des Frequenzganges, da durch das Einbetten in die Platte die Masse des Resonanzkörpers größer wird. Eine messtechnische Untersuchung erfolgt in Kapitel 8.
Darüber hinaus ist eine Spannungsversorgung von $+48V$ nötig, die von acht *Focusrite OctoPre MKII* Mikrofonvorverstärkern (engl. preamps) geliefert wird. Diese wandeln gleichzeitig die analogen Signale in das digitale ADAT-Format um. Die acht Kanäle werden optisch über TOSLink[10] übertragen. Alle Preamps werden in einem ADAT/MADI-Converter (*RME ADI-648*) konzentriert und gehen von dort aus optisch weiter an einen Industrie-PC, der die Verarbeitung vornimmt. Das zusammengefasste Signal wird dort von der Soundkarte (*RME Hammerfall DSP MADI*) entgegen genommen und dem Rechnersystem zur weiteren Verarbeitung zur Verfügung gestellt.

Um den Ablauf der Audioströme synchron zu halten dient die *Hammerfall*-Soundkarte gleichzeitig als Taktgeber für alle eben genannten Komponenten. Das garantiert ein echt paralleles Eintreffen der Sampleblöcke am Computersystem, ohne das es zu Verschiebungen kommt. Für eine spätere Wiedergabe von Audiodaten verfügt das System über zwei Hochleistungsregielautsprecher (*ME Geithein RL 904*), die sich durch einen linearen Frequenzgang und frei von Verfälschungen auszeichnen.

Die Gesamtübersicht über alle Hauptbestandteile der Hardware sind noch einmal in dem Schaubild 9 zusammengefasst.

[10] Optisches Übertragungskabel mit rechteckigen Steckern.

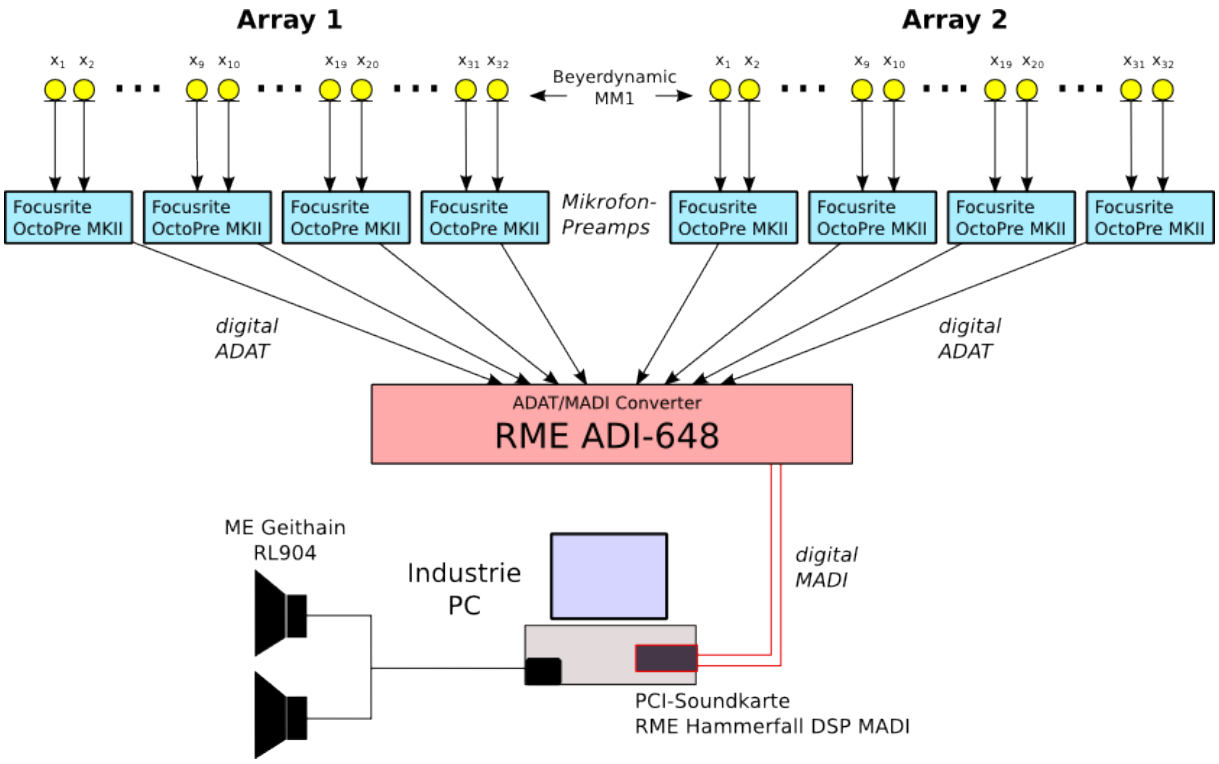


Abbildung 9: Schematische Darstellung der Hardwarekomponenten

4.2 Softwarekomponenten

Die softwareseitige Verarbeitung der Audiosignale läuft über einen Industrie-PC mit dem Betriebssystem *MS Windows 7 (64-Bit)* ab. Für die Programmierumgebung wird eine *Eclipse* IDE eingesetzt, die wiederum auf einer Java-Entwicklungsumgebung aufsetzt. Beide Systeme benutzen einen 32-Bit-Adresssatz zur Berechnung der Operationen.

Die Einbettung des Audioprogrammes erfolgt in dem Java-basierten „SpeechLab“-Projekt. Zu diesen gehören Teilgebiete wie Motorsteuerung, Spracherkenner oder die Regulierung der LED-Farben. Durch die objektorientierte Implementation sind alle Bestandteile überwiegend modular und damit äußerst skalierbar. Das garantiert eine flexible Arbeitsweise und Erweiterbarkeit für zukünftige Projekte.
Die Optik des Hauptprogrammes ist in einem LCARS „Look and Feel“ (dt. Aussehen und Handhabung) gehalten, was im Design der StarTrek™-Filme entwickelt wurde. Eine Konzipierung für sprach- und berührungsempfindliche Konsolen, macht dieses Look-and-Feel nicht nur optimal für den Einsatz im *Raumschiff Enterprise*, sonder auch für die Applikationen des SpeechLabs (Abbildung 10).

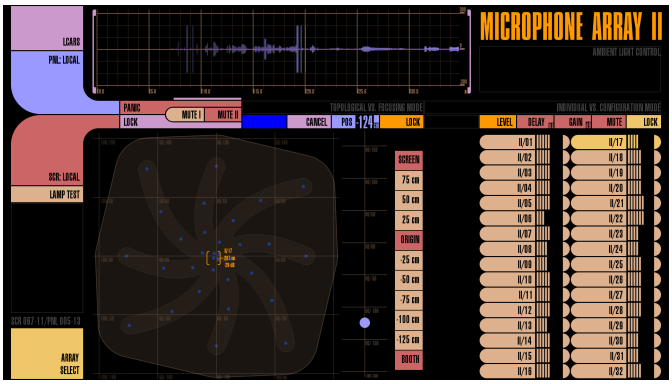


Abbildung 10: Microphonearray 2 in der LCARS-Benutzerkonsole

4.3 Raumkoordinatensystem

Da alle Mikrofone nur eine relative Lage auf ihren jeweiligen Platten haben, war es erforderlich ein einheitliches Raumkoordinatensystem zu schaffen. In dem System bekamen alle Mikrofone eine absolute Position in dem Messfeld, was für die Mikrofonarrays vorgesehenen war. Die grundlegende Fläche für die Messung umfasst ein Quadrat von 4,40x4,40m (Grundriss Anhang E). Das Zentrum des Quadrates befindet sich mittig vom Kontrollbildschirm, 2,20m in den Raum hinein und stellt den Ursprung des Koordinatensystems dar. Gleichzeitig befindet sich, in einer Höhe von 2,50m, die mittig angebrachte Deckenkonstruktion. An ihr ist das zweite Array befestigt, welches sich linear durch einen Schrittmotor steuern lässt. Der Bewegungsradius des Mikrofonfelds 2 beträgt 90cm zum Bildschirm hin und 152cm in die entgegengesetzte Richtung. Die sich so verändernde Y-Koordinaten, der Mikrofone an der Decke wird später in die Berechnungen einbezogen. In der perspektivisch korrekten Darstellung 11 ist der aufgespannte Raum veranschaulicht.

Ausgehend von dem konstruierten Kubus, kann nun die absolute Lage der Mikrofone bestimmt werden. Dafür erfolgte ein Heranziehen der berechneten Daten von Dipl.-Ing. Thomas Fehér (siehe Anhang C und D), die in dreidimensionale Koordinaten transformiert wurden. Für das Mikrofonfeld 1, was sich um den Bildschirm herum befindet, musste nur berücksichtigt werden, dass das zweidimensionale Array nun um eine dritte Dimension erweitert wurde. Das erfolgte, indem die alten Y-Werte zu den neuen Z-Koordianten umgewandelt wurden, unter der Berücksichtigung eines zusätzlichen Niveaus von 1,54m, die für die Anbringungshöhe des Bildschirms steht. Die Z-Koordinaten ergaben sich aus dem Abstand zum Messfeldzentrum, die von dort aus 2,20m entfernt sind. Die verbleibenden X-Koordinaten wurden bewahrt.

Das Array 2 an der Deckenkonstruktion erforderte mehr als die Transformation in die dritte Dimension. Und zwar war es hierfür notwendig die Schräglage zu berücksichtigen. Die Platte wurde aus akustischen Zwecken, etwas geneigt, um einer Schallreflexion direkt unterhalb zu entgehen. Der Winkelsatz

$$c = a.\frac{\sin(\gamma)}{\sin(\alpha)} \tag{4.3.1}$$

mit einem Winkel von $\alpha = 8,4°$ wurde hierfür angewendet, wobei $\gamma = 90°$ und $a = Y_{ALT} - 1,032m$ betragen. Das ergibt für die Z-Koordinate zusammen mit der Anbringungshöhe ein $z = c + 2,20m$. Für die X–Werte musste nur eine Spiegelung entlang der Y-Achse vorgenommen werden, da bei der Konstruktion der Platte von oben gebohrt wurde und somit eine andere Sicht entstanden ist. Die Y-Koordinaten bleiben konstant, unter der Prämisse, dass sich das Mikrofonfeld über dem Mittelpunkt befindet. Im Falle einer händischen oder motorisierten Bewegung des Feldes, wird durch den oberhalb an der Decke befestigen Laser die absolute Position der Arrayplatte neu bestimmt und somit die genaue Lage im Raum wiedergegeben. Dies wird für die Y-Koordinaten folglich berücksichtigt und dynamisch angepasst.

Alle dreidimensionalen Lagedaten der Mikrofone befinden sich im Anhang F und stellen hiermit die Grundlage für die Berechnung der Signalverzögerungen bzw. der einzelnen Abstände zu dem jeweiligen anvisierten Ziel.

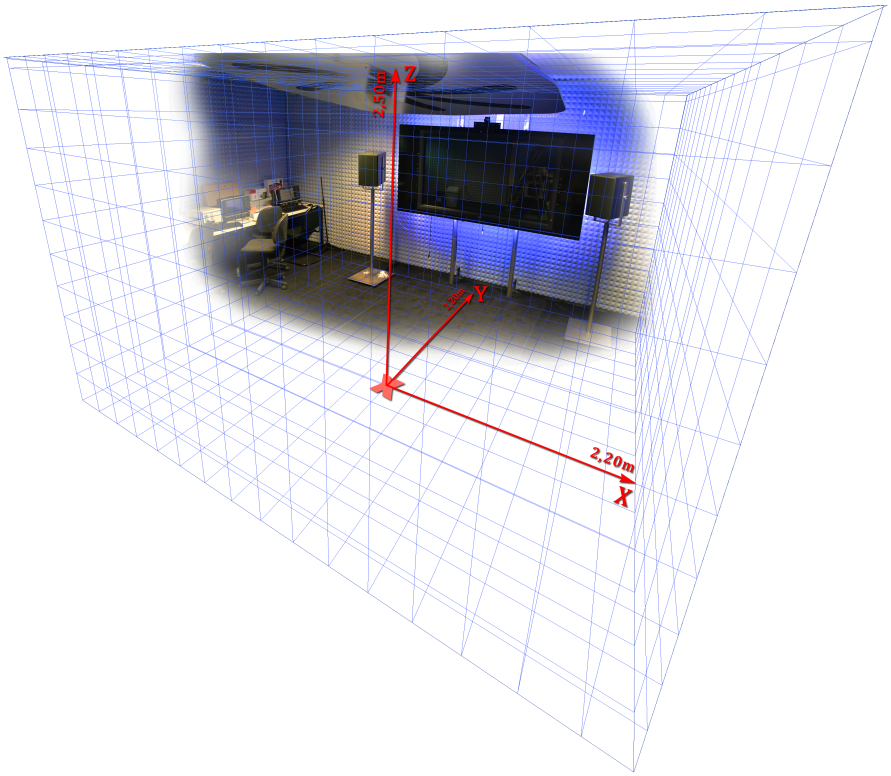


Abbildung 11: Visualisierung des Messfeldes der Arrays

5 Beamforming

Unter dem Begriff „Beamforming“ (dt. Strahlformung) wird allgemein das passive Auslenken eines Sensorfeldes (engl. sensor array) in eine fokussierte Richtung verstanden. Erste Aufzeichnungen über deren Anwendungen führen bis auf die frühen 80er Jahre des 20. Jahrhunderts zurück. In der nachrichtentechnischen Forschung nimmt es daher immer mehr an Bedeutung zu und wird in vielen Fachgebieten, wie bei dem Radar, Sonar, Seismologie, Kommunikation und Audiotechnik verwendet. Dabei wird diese Technik genutzt, um ein Signal zu Orten, dessen Einfallswinkel zu bestimmen oder gar ein spezifischen Sprecher aus einer Geräuschkulisse heraus zu verstärken, unter der Gleichzeitigen Reduktion der Nebengeräusche und Nachhallelemente (Vgl. [BW01]).

Mit Hilfe des Beamforming wird unter Einsatz eines Mikrofonarrays ein Strahl, durch räumliches Filtern der anliegenden Signale, gebildet. Es erfolgt dabei eine unterschiedliche Verzögerung der Kanäle, um so einen Strahl zu bilden, der sich auf eine beliebige Position lenken lässt. Die Auslenkung ist dabei rein passiv und es muss kein Einfluss auf physikalische Anordnung der Mikrofone genommen werden. Um jedoch die Synchronisation der Signale exakt durchzuführen, muss eingangs eine Bestimmung der Zeitunterschiede der einzelnen Mikrofone erfolgen. Das ist notwendig wegen den differenzierenden Entfernungen und Einfallswinkeln, wodurch der Schall auch ungleich an ihnen auftrifft. Die Anpassung der Verzögerungen (engl. delays) erfolgt dabei flexibel, um auch bewegende Ziele weiterhin anvisieren zu können. In einem zweiten Schritt werden alle verschobenen Signale zu einem Ausgangssignal summiert.

Beamforming ist sowohl im Frequenzbereich, als auch im Zeitbereich möglich. In dieser Arbeit wird jedoch nur auf die Betrachtungen im Zeitbereich eingegangen. Für die Erzeugung von Beamformer gibt es eine Vielzahl an verschiedenen Algorithmen und Vorgehensweisen. Grundlegend werden konventionelle und adaptive Beamformer unterschieden. Die als zweites genannte Beamformerklasse wird meist dazu genutzt, um die breitbandige Ausrichtung der Keulen so zu steuern, dass Störquellen direkt in den Nullstellen liegen. Darüber hinaus wird der Gewichtungsfaktor durch einen adaptiven Transversalfilter ergänzt (Vgl. [Dre99] S. 38). So eine Anordnung ermöglicht eine direkte Ausrichtung auf die Schallquelle. Das geschieht einschließlich einer Dämpfung nebenstehender Störquellen. Auf weitere Beamformeringverfahren, wie superdirektive oder Filter-Beamformer (Wiener, Frost oder Kalman Filter) wird an dieser Stelle nicht weiter eingegangen.
Für die Verarbeitung der Audiosignale im SpeechLab soll in erster Linie das „Delay-and-Sum-Beamformering“ (dt. Verzögerungs- und Summierungsstrahlrichten) im Blickpunkt stehen. Es ermöglicht alle Mikrofone auf einen expliziten Sprecher auszurichten. Durch de Justierung und die Eigenschaften der Hauptkeule verringert sich so der Signalrauschabstand (engl. signal to noise ratio, SNR), unter gleichzeitiger Dämpfung der Nebenkeulen.

5.1 Sum-and-Delay-Beamforming

Ausgehend davon, dass die Schallwellen nicht direkt aus der Querabrichtung auf ein Array mit N Mikrofone treffen, kommt es zu einem zeitlichen Versatz der eintreffenden Signale. Das lässt sich, wie bereits im Kapitel zuvor beschrieben, mit der weiteren Entfernung begründen, den der Schall zu einem entfernteren Mikrofon zurücklegen muss. Die einzelnen Delayglieder τ_i $(i = 1, 2, ..., N)$ werden dazu genutzt die Verzögerung so auszugleichen, dass die anliegenden Signale $x_i(t)$ danach kohärent sind. Die Wichtungsfunktion w_i sorgt im weiteren Verlauf für eine entsprechende Bewertung der unabhängigen Signale, um anschließend das zu einem Gesamtausgang $y(t)$ zusammenzufassen ([Gol04]). In der Literatur wird die Wichtungfunktion auch häufig als Fensterfunktion benannt. Die Abbildung 12 zeigt alle Zusammenhänge eines Delay-and-Sum-Beamformer DSB.

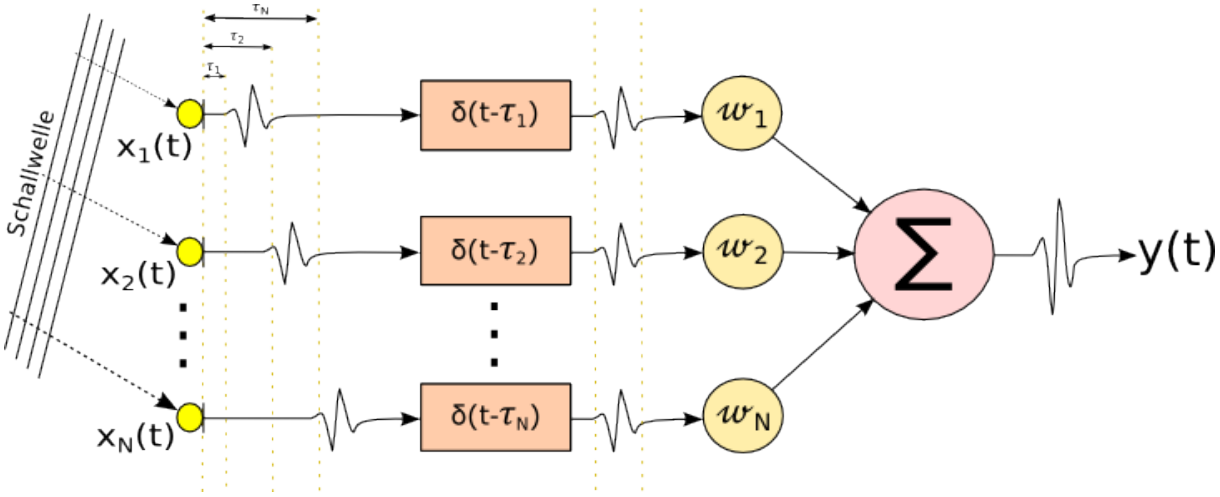


Abbildung 12: Blockschaltbild eines Delay-and-Sum-Beamformers

Ausgehend vom Schaubild 12 lässt sich eine Formel für das Ausgangssignal $y(t)$ aufstellen:

$$y(t) = \sum_{i=1}^{N} w_i \cdot x_i(t - \tau_i). \tag{5.1.1}$$

Durch die bereits erfolgte Digitalisierung der Eingangssignale von den Soundkarten wird die Delay $(t - \tau_i)$ durch eine Sampleverzögerung k_i substituiert:

$$y(k) = \sum_{i=1}^{N} w_i \cdot x_i(-k_i). \tag{5.1.2}$$

Die Delays k_i errechnen sich aus dem Abstand d_i von einer Zielkoordinate $s = (x_s, y_s, z_s)$ zu allen Mikrofonenpositionen $r_i = (x_i, y_i, z_i)$ $(i = 1, 2, ..., N)$. Dafür wird der Satz des Pythagoras im dreidimensionalen Raum angewendet:

$$d_i = \sqrt{(x_s - x_i)^2 + (y_s - y_i)^2 + (z_s - z_i)^2}. \tag{5.1.3}$$

Aus allen errechneten Abständen wird das Mikrofon mit der weitesten Entfernung d_{max} zum Punkt s ausgewählt. Es wird als Referenz angenommen, da bei ihm der Schall als letztes ankommt. Folglich ist eine Zurückhaltung aller Signale notwendig, bis der Schall auch an dem letzten Mikrofon eingetroffen ist. Daraus wird anschließend die Differenz des Gesamtabstand d_{max} mit den einzelnen Abständen d_i bestimmt. Zusammen mit der Schallgeschwindigkeit $c_s = 343\frac{m}{s}$ aus (2.1.4) und der Sampleabtastfrequenz $f_s = 44100Hz$ lässt sich so die Verzögerung für jeden Kanal bestimmen:

$$k_i = \frac{d_{max} - d_i}{c_s} \cdot f_s \tag{5.1.4}$$

Die einzelnen, verzögerten Signale k_i werden schlussendlich zu einem Gesamtausgangsignal $y(k)$ mit der Formel (5.1.2) zusammengefasst. Damit erfolgt im einfachsten Fall die Wichtung der Signale nach ihrer Entfernung mit w_i, um so den Verlust der Schallintensität auszugleichen. Alle errechneten Abstände für eine konkrete Zielposition werden schließlich in einem „steering"-Vektor (dt. Lenkvektor) zusammengefasst.

Für die Normalisierung des Ausgangs wird zusätzlich durch die Anzahl der verwendeten Mikrofone N geteilt:

$$y_{norm}(k) = \frac{y(k)}{N}. \tag{5.1.5}$$

Das Ausgangssignal von diesem Verfahren, wird nun dazu genutzt einen bestimmten Punkt des Raumes genau anzuvisieren.

DSBs die im komplexen, fouriertransformierten Bereich rechnen, können darüber hinaus noch genauer ihre Haupt- und Nebenkeulen variieren. Dabei hilft die Wichtungsfunktion, unter Verwendung der Phase und Amplitude, die Keulen optimal einzustellen. Eine Amplitudenwichtung wirkt sich dabei auf die Form des Beamformers aus. Wohingegen die Phase mehr auf die Richtung er Hauptkeule Einfluss hat ([Pap05] S.18). Um das zu realisieren muss jedoch die einfache Wichtungsfunktion $w_n(k) = 1/N$ durch ein komplexes Glied ausgetauscht werden, was in einem weiterführenden Projekt durchgeführt werden könnte.

5.2 Beampattern

Eine Möglichkeit die Beschaffenheit eines Beamformers zu bestimmen, ist dessen Richtcharakteristik (engl. beam pattern) zu errechnen. Daraus lässt sich die genauen Beschaffenheit der Hauptkeule bestimmen und die Lage der Nebenkeulen identifizieren. Die Berechnungen erfolgen für eine Schallrichtung, eine bestimmte Frequenz und einen spezifischen Winkel. Dafür sind der Azimutwinkel φ (horizonatal) und Elavationwinkel θ (vertikaler) notwendig (Vgl. [BW01] S.22). Eine Iteration über alle Winkel und Frequenzen ergibt folglich die Richtfunktion des Beamformers:

$$|H(\varphi,\theta)|\Big|_{dB} = -10log_{10}\left(\frac{|W^H d|^2}{W^H \Gamma_{vv}\Big|_{Wavefront} W}\right). \tag{5.2.1}$$

Dabei entspricht d dem Delay des jeweiligen Samples a_i $(i = 1, 2, ..., N)$, in Abhängigkeit von einer Frequenz Ω und der Phasenverzögerung τ_i $(i = 1, 2, ..., N)$.

$$d = \left[a_1 e^{(-j\Omega\tau_1)}, a_2 e^{(-j\Omega\tau_2)}, ..., a_N e^{(-j\Omega\tau_N)}\right] \tag{5.2.2}$$

Das Wichtungsglied in der Gleichung (5.2.1) setzt sich aus den komplex konjugierten Wichtungen $w_i^* = w_i(e^{J\Omega})$ mit $i = 1, 2, ..., N$ zusammen. Der Operator ^{H} bezeichnet dabei die Transformierung (Hermite Form).

$$W^H = [w_0^*, w_1^*, ..., w_N^*] = \left[w_0\left(e^{J\Omega}\right), w_1\left(e^{J\Omega}\right), ..., w_N\left(e^{J\Omega}\right)\right] \tag{5.2.3}$$

Die Schallwelle Γ_{vv} wird des weiteren, mit

$$\Gamma_{vv}\Big|_{Wavefront} = e^{(j\Omega\tau_{nm})} \tag{5.2.4}$$

berechnet. Die Laufzeitdifferenz τ_{nm} wird unter der Verwendung der Entfernung zwischen zwei Mikrofonen $l_{n,m}$ und der Winkel für den Azimut φ und der Elavation θ berechnet. Das ergibt unter der Verwendung der kartesischen Koordinaten:

$$\tau_{nm} = \frac{f_s}{c_s}\left(l_{x,nm} sin(\theta)cos(\varphi) + l_{y,nm} sin(\varphi)cos(\theta) + l_{z,nm} cos(\theta)\right). \tag{5.2.5}$$

Die Abstände berechnen sich aus $l_{x,nm} = x_n - x_m$, $l_{y,nm} = y_n - y_m$ und $l_{z,nm} = z_n - z_m$. Die Schallgeschwindigkeit kann mit $c_s = 343\frac{m}{s}$ aus (2.1.4) entnommen werden und die Samplefrequenz beträgt $f_s = 44100Hz$.

Eine Richtcharakterristik lässt sich sehr kompliziert plotten, was der Komplexität aus den zwei Winkeln und der Frequenz zu verschulden ist.

5.3 Sprecherlokalisierung mittels Beamformern

Die Lokalisation des Sprechers baut auf den gewählten Raumkoordinatensystem auf und durchläuft eine Reihe von Schritten die fortlaufend bearbeitet werden:

1. Einteilung des zu messenden Raumes in Abtastpunkte.
2. Aufteilung des Gitters in größere Raumblöcke.
3. Erstellung von Steering-Vektoren für alle Abtastpositionen.
4. Messung der großen Blöcke durch breiten Beamformer (wenige Mikrofone).
5. Maximum aus den der gemessenen Energiewerte ermitteln.
6. Block mit größten Energie feiner Abtasten.
7. Eventuelle Änderung der aktuellen Sprecherposition einleiten.
8. Bei Änderung der Lasermessdaten gehe zu 3.
9. Gehe zu 4.

Die ersten beiden Schritte erfolgen nur einmalig. Auf die dafür angelegten Koordinatenfelder der Abtastpunkte kann später in der Implementation dynamisch zugegriffen und so die entsprechende Position anvisiert werden. Der zu messende Raum orientiert sich am Raumkoordiantensystem. Er wurde aus diesem Zweck in Würfel unterteilt, die eine Abmessung von 20x20x20cm besitzen. Die Raumquadrate stellen gleichzeitig die Genauigkeit der Positionsbestimmung dar. Aus den Feldabmessungen aus Kapitel 4.3 und den gewählten Würfelabmessungen ergeben sich somit 23 Würfel in der Breite bzw. Länge und 12 in der Höhe. Das macht insgesamt 6348 mögliche und messbare Raumpositionen.

Angesichts der hohen Anzahl würde ein Durchlaufen aller Koordinaten zu lange dauern. Aus diesem Grund wurde eine Auswahl an Positionen getroffen, die größere Quadranten aufspannen sollen. Die Anvisierung erfolgt dabei mit einer geringeren Anzahl an Mikrofonen und somit auch einem breiteren Beamformer. Die Annahme, dass der Sprecher sich nicht in seiner Größe ändert, dient der Ausgangslage für die Ermittlung einer Höheneinheit. Daher muss nur eine bestimmte Ebene abgesucht werden. Eine Darstellung der aufgespannten Fläche für die zwölf Höheneinheiten befindet sich im Anhang 27.

Nach der Aufteilung des Raumes erfolgt die Berechnung der Steeringvektoren für alle Mikrofone und der jeweiligen Position. Das wird in Abhängigkeit von der derzeitigen Lage des variablen Mikrofonfeld 2 bestimmt. Durch die Steeringvektoren kann nun für jeden beliebigen Punkt eine zeitliche Verschiebung der Signale erfolgen und der Beamforming-Algorithmus angewendet werden. Die Messungen der Positionen an sich wird über die Bestimmung des effektiven Schallpegels am Ausgang des Beamformers durchgeführt. Es erfolgt daraufhin ein Vergleich aller in einem Intervall gemessenen Werte. Die größte Intensität wird folglich als Standort des Sprechers erkannt. Die Messung wird im Schritt vier und fünf für die größeren Raumblöcke auf verschiedenen Ebenen durchgeführt. Sollte die Ebeneneinheit sich nicht ändern, wird diese beibehalten. Eine zyklische Überprüfung erfolgt jedoch mit größerem Zeitintervall. Bei der Erkennung eines neuen Raumblocks, ändern sich auch unmittelbar die Abtastkoordinaten für die feinere Abtas-

tung (Schritt 6). Im Falle einer daraufhin neu erkannten Energiequelle, kann auch eine neue Sprecherposition an den parallel laufenden, beamforming-gestützten Audioausgabealgorithmus übergeben werden.
Angesichts der Tatsache, dass eine Veränderung der Position des Mikrofonfelds 2 möglich ist, muss ständig eine Überprüfung erfolgen. Im Falle einer Änderung ist eine Neuberechnung der Steeringvektoren erforderlich. Sollte das jedoch nicht nötig werden, geht der Algorithmus zum Schritt vier zurück und läuft abermals alle Positionen durch.

Mittels solch einem Ablaufs sollte es so möglich sein, die Sprecherposition effektiv zu bestimmen, ohne Übermäßig viele Resourcen in Form von Zeit und Rechenkapazität zu verbrauchen. Die genaue Umsetzung wird im Abschnitt 6.2 und 6.3 letztendlich genauer elaboriert.

6 Realisierung einer Sprecherlokalisation

6.1 Audioschnittstelle

Um mit der Entwicklung der Audioverarbeitung beginnen zu können, war es in erster Linie vorrangig eine adäquate Audioschnittstelle zu finden. Da die Implementationsarbeiten unter Java stattfinden sollten, lag es nahe ein passendes Interface (dt. Schnittstelle) auch unter der gleichen API zu entwerfen. Dabei viel die Wahl auf die frei verfügbare Audiobibliothek „JPortAudio“. Diese ist als „cross-platform, open-source“ (dt. plattformübergreifend, quelloffen) deklariert und bietet somit die Möglichkeit auf verschiedenen Betriebssystemen funktionsfähig zu sein. Trotzdem muss es mit Vorsicht behandelt werden, da es erst im Jahr 2012 entwickelt wurde und es sich noch in einem „alpha“-Entwicklungsstadium befindet.

JPortAudio ist eine native Java-Schnittstelle zu dem gleichsamen „PortAudio“, welches in 'C' bzw. 'C++' konzipiert wurde ([RB13]). Es ist im Grunde eine einheitliche API für eine Vielzahl an unterschiedlichen Betriebssystemen mit verschiedenen Audioschnittstellen (siehe Abb. 13).

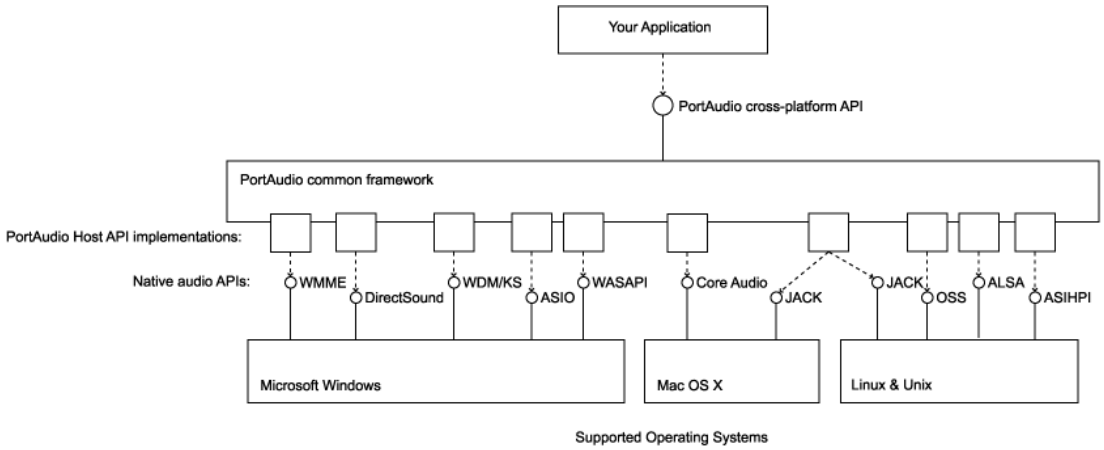


Abbildung 13: Modularer Aufbau von PortAudio[11]

PortAudio setzt unter Windows auf Schnittstellen wie WMME, DirectSound oder ASIO. Das „Audio Stream Input/Output“, kurz ASIO, ist ein ganz besonderes Audiotransfer-Protokoll. Es unterstützt eine synchrone Verarbeitung von Multichannel-Audiointerfaces und ist demnach optimal für den professionellen Studioeinsatz geeignet. Zusätzlich kennzeichnet dieses System eine sehr geringe Latenz (engl. Laufzeitverzögerung), die für Echtzeitanwendungen vom Vorteil sind. Die Lizenz an ASIO hält der Softwareentwickler *Steinberg*. Aufgrund dessen ist es kein direkter Teil von PortAudio und eine Einbindung in die Umgebung muss separat erfolgen. Unter der Benutzung einer Anleitung und verschiedenen Projektdateien lies sich so eine DLL erzeugen, die alle gewünschten Audioschnittstellen enthält und unter JPortAudio Anwendung findet.

Die Interface-Klasse arbeitet ausnahmslos mit blockierenden Schreib- und Leseoperationen. Das sperrende Zeitintervall, in dem die Soundkarte gelesen wird, regelt sich über die Größe des

[11] Quelle: `http://portaudio.com/docs/v19-doxydocs-dev/api_overview.html`

Lesenpuffers und die Latenz. Bei einer Puffergröße von 512 Sample beträgt die blockierende Zeit des System etwa 11,5ms. In dem folgenden Programmcodeauszug 1 ist ersichtlich, wie die Anbindung der JPortAudio-Klasse an ein eigenes Java-Programm aussieht. Hierbei erfolgt eine abwechselnde Füllung des Puffers (engl. buffer) mit einer konkreten Anzahl an Sample ($NUM\_FRAMES = 512$), die anschließend zur weiteren Nutzung zur Verfügung gestellt werden. Im Falle eines nicht rechtzeitigen Abholens der Audiodaten gibt die Methode „read“ einen Boole'scher Wert zurück, wodurch kenntlich gemacht wird, dass es zu einem Pufferüberlauf im Audiointerface gekommen ist.

```
1 if (inStream.read(inBuffer[bufferIndex], NUM_FRAMES))
2     LCARS.err("HDW", "Audio input error: - BUFFER OVERFLOW");
```

Listing 1: Einlesen der Audiodaten

Der Zugriff auf eine Audioschnittstelle verlangt jedoch im Vorfeld eine Auswahl des passenden Interfaces. Das geschieht über eine Initialisierung des Audioinputstreams. Dabei werden Parameter, wie Gerätenummer, Anzahl der Kanäle, Formatierung der Samples oder die Samplefrequenz festgelegt (Programmcodeabschnitt 2). Neben der Abtastfrequenz von 44100Hz, wurden für der Samplefortmatierung Float-Werte gewählt, die eine spätere rechnerische Verarbeitung erleichtern sollten.

```
1  private static void openInputStream(){
2    inParameters = new StreamParameters();
3    inParameters.device = device;
4    inParameters.channelCount = CHANNELS;
5    inParameters.sampleFormat = sample_param;
6    inParameters.suggestedLatency = di.defaultHighInputLatency;
7    inStream = PortAudio.openStream(inParameters, null, SAMPLERATE,
8        FRAMES_PER_BUFFER, FLAGS);
9
10   if (inStream.isActive() == false){
11     inStream.start();
12     System.out.println("... Input stream started");
13   }
14 }
```

Listing 2: Aktivierung des Audiostreams

Nach der Initialisierung und Abholung der Audiodaten, kann folgend nun mit der tatsächlichen Verarbeitung der Samples begonnen werden.

6.2 Beamforming Implementation

Die schwerpunktmäßige Beamformerimplementation erfolgte in der Java „Audio“-Klasse. Sie soll fertige Umsetzung des Delay-and-Sum-Beamformers und der automatische Quellenlokalisation beinhalten. Dabei wird hier auf die programmtechnische Implementation, des im Abschnitts 5.1 theoretisch beschriebenen DSB-Algorithmus, gesetzt.

Um mit der Audioverarbeitung beginnen zu können, müssen zuvor erst die Audioschnittstellen initialisiert werden (Programmcodeauszug 2). Im Anschluss erfolgt die Berechnung der Steeringvektoren unter Verwendung der Formeln (5.1.3) und (5.1.4). Sie werden zusammen mit den vorher bestimmten Raumkoordinatenpunkten in einem Feld abgelegt („Position3d[] positions = new Position3d[6348]“). Dieses Feldkonstrukt umfasst alle 3D Zielkoordinaten und die jeweiligen Verzögerungsvektoren der Mikrofone. Beide Schritte, „Audio“ und „Positions“ sind in der Abbildung 15 gelb veranschaulicht.

Wenn die beiden ersten Schritte erfolgt sind, kann mit dem Einlesen des Audiostreams begonnen werden. Die notwendigen Daten werden dafür in einem Audiopuffer mittels der „read()“-Methode abgelegt (Listing 1). Er Umfasst 512 Abtastwerte pro Audiokanal. Das entspricht bei 64 verwendeten Kanälen 32768 Samples, die in dem Puffer (engl. Buffer) passen. Somit lässt sich der synchronisierten Audioinput in 512 Rahmen (engl. frames) einteilen, die jeweils ein Sample von jedem Kanal beinhalten. Das entspricht für den ersten Frame, dass von allen Kanälen das erste Float-Sample enthalten ist. Durch die programmiertechnische Verarbeitung von Datenfeldern wird jedoch mit dem Index 0 begonnen zu zählen (Abb. 14).

Kanal-Nr.	0	1	2	...	63	0	1	2	...	63	...	0	1	2	...	63
	0	0	0	0	0	1	1	1	1	1	...	512	512	512	512	512
Sample-Nr.	0	1	2	...	63	64	65	66	...	127	...	32704	32705	32706	...	32767

Abbildung 14: Aufbau des Audiobuffers

Der Audiobuffer besteht aus einem mehrdimensionalen Feld, womit es möglich ist stets neue Daten einlesen zu können, ohne diese zu kopieren. Es erfolgt nur eine Verschiebung des Index („bufferIndex“) auf das nächste Feld. Somit ist es möglich schnell Audiodaten einzulesen und gleichzeitig verarbeiten zu können. Die zyklische Verschiebung des Index auf den sogenannten „inBuffer“ erfolgt zwischen vier durchgängig laufenden Threads (dt. Programmfäden). Sie sind in der Abbildung 15 blau dargestellt und arbeiten jeweils sperrend mit einer Semaphore, um die Abarbeitung der jeweiligen Aufgabe nicht zu beeinträchtigen. Gleichzeitig sind sie der Kern der Audioverarbeitung, da sie neben dem Einlesen und Verarbeiten auch die Aufgabe der Soundwiedergabe übernehmen.

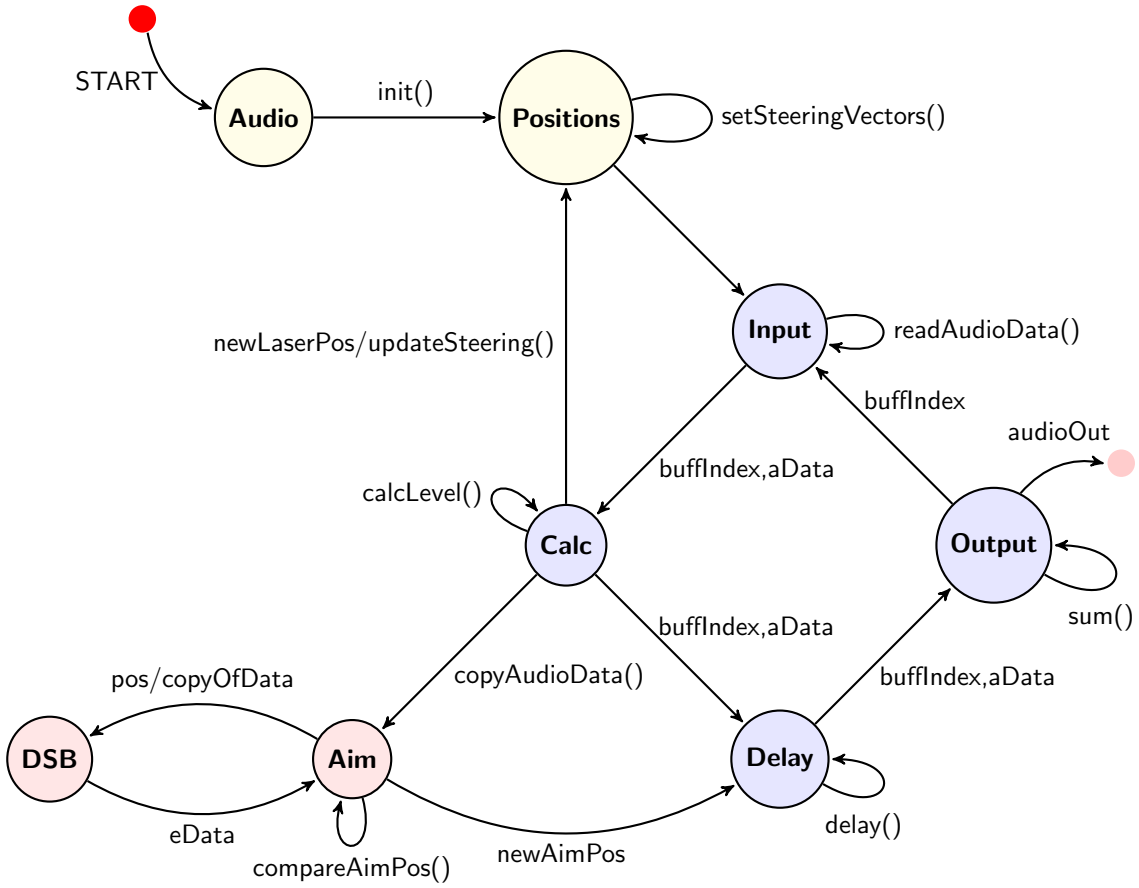


Abbildung 15: Zustandsdiagramm

Der Thread „Calc“ hat die Aufgabe die Audiodaten auf deren Pegel hin zu überprüfen. Dafür erfolgt die Bestimmung des Effektivwertes mittels der Formel (2.2.3). Auf dessen Basis kann daraufhin die Berechnung des Spannungspegel nach (2.3.2) vorgenommen werden. Zusätzlich wird jedoch noch der Abtastwert (dB FS) gespeichert und für das Anzeigen einer Frequenzkurve zur Verfügung gestellt. Nachfolgend werden in diesem Thread-Teil die Audiodaten für die Überprüfung der Sprecherposition in ein weiteren Puffer kopiert, um die Kernabarbeitung nicht zu unterbrechen. Eine zusätzliche Kontrolle soll überprüfen, ob das Mikrofonfeld 2 an der Decke bewegt worden ist. Daher wertet ein weiteres Interface die Hardwaredaten von Motor und Laser aus. Tritt der Fall ein, erfolgt ein Verwerfen der Audiodaten und eine Neuberechnung der Steeringvektoren. Die so entstehende akustische Pause bzw. Verzerrung wird toleriert, da ein gleichzeitiges Verschieben der Platte und Messen vorerst nicht angedacht wurde.

Der signifikanteste Teil der Kernmethoden ist der DSB-Teil. Er ist in zwei Komponenten gespalten. Zum einen erfolgt die Verschiebung der Samples im „Delay“-Abschnitt und zum anderen die Summierung im inneren des „Output“-Threads. Für die Verzögerung muss aus dem Positions3D-Feld der entsprechende Steeringvektor eingelesen werden. Die Variable, die für die Auswahl des Vektors zuständig ist, wird durch atomares Lesen und Schreiben geschützt und ist daher vom Typ „AtomicInteger“. Das garantiert, dass keine zwei Threads gleichzeitig die Variable verändern dürfen. In einem neu angelegt Feld werden schließlich die verschobenen Daten eingegliedert. Das erfolgt in Abhängigkeit ihres jeweiligen Delays. In dem Programmcodestück 3 ist die verschachtelte Verschiebung der Samples des einen „inBuffers“ auf den „delayBuffer“ genau zu sehen. Hierfür muss in jedem Schritt eine Neuberechnung der neuen Sampleposition auf der Grundlage der Steeringvektoren erfolgen.

```
1  for(int i=NUM_FRAMES * CHANNELS; i<delayBuffer.length; i++)
2    delayBuffer[i]=0f;
3
4  // delaying
5  for (int channel = 0; channel < CHANNELS; channel++)
6  {
7   int steer = (int) positions[atomicTargetPoint.get()].steering[1][channel];
8   for (int sample0 = channel, sample1 = channel + steer * CHANNELS;
9       (sample1 < NUM_FRAMES*CHANNELS*2) && (sample0 < NUM_FRAMES*CHANNELS);
            sample0 = sample0 + CHANNELS, sample1 = sample1 + CHANNELS)
10  {
11    delayBuffer[sample1]=inBuffer[bufferIndex][sample0];
12  }
13 }
14
15 // to pass the audio data for output
16 System.arraycopy(delayBuffer, 0, inBuffer[bufferIndex], 0, inBuffer[bufferIndex
       ].length);
17 // carry over for next buffer
18 System.arraycopy(delayBuffer, inBuffer[bufferIndex].length, delayBuffer, 0,
       inBuffer[bufferIndex].length);
```

Listing 3: Sampleverzögerung

Der so neu gefüllte Delay-Puffer wird im Anschluss zurück in den „inBuffer“ kopiert, um für die weitere Verarbeitung zur Verfügung zu stehen. Der durch die Verschiebung entstandene Übertrag muss zusätzlich für den nächsten Puffer übergeben werden.

Damit schlussendlich eine Wiedergabe der Audiodaten möglich ist, muss zuvor die Summierung der Samples auf einen Monokanal erfolgen. Das geschieht im „Output“-Thread unter der Verwendung von Gleichung (5.1.2). Dabei werden alle Kanäle eines Frames zusammengezogen und jeweils in Abhängigkeit ihrer Entfernung gewichtet. Übertragungswege die ausgeblendet, sprich gemuted (dt. dämpfen) wurden, entfallen bei der Summierung und werden übersprungen. Die

dafür notwendigen Informationen sind in dem Feld „activeChannel[]“ abgelegt. Infolgedessen, dass JavaSound (Standard Java Audio Ausgabe) nur Bytewerte für den Audioausgabestream akzeptiert, muss noch eine Konvertierung erfolgen. Dafür werden die Float zu Short gecastet (dt. geformt) und anschließend durch Shift-Operationen (Little-Endian) in Byte-Werte konvertiert.

```
1  for (int frame = 0, nout = 0; frame < NUM_FRAMES; frame++)
2  {
3   float sample = 0f;
4   short shortSample = 0;
5   float anzahlNullen = 0;
6   for (int nin = frame * CHANNELS, channel = 0; channel < CHANNELS; nin++,
        channel++)
7   {
8    if(activeChannel[channel]==false) continue;
9    if (inBuffer[bufferIndex][nin] == 0 ) anzahlNullen++;
10    sample += inBuffer[bufferIndex][nin]*channelGains[channel]*positions[
          atomicTargetPoint.get()].steering[0][channel];
11  }
12  sample =  sample / (activeChannels.get() - anzahlNullen);
13  shortSample = (short) ((sample / activeChannels.get()) * recgain);
14  outBuffer[nout++] = (byte) (shortSample & 0x00FF);
15  outBuffer[nout++] = (byte) ((shortSample & 0xFF00) >> 8);
16 }
```

Listing 4: Samplessummierung

Nachstehend kann man nun den Audiostream dazu benutzen, um einen Sprecher aufzuzeichnen, oder das gesamte Mikrofonarray an einen Spracherkenner koppeln. Unzählige Szenarien lassen sich nun mit so einer Audioverarbeitung realisieren.

6.3 Implementation der automatischen Lokalisierung

Die automatisch Lokalisierung ist angeknüpft an die Audioverarbeitung aus Kapitel 6.2. Wie bereits im Zustandsautomat (Abb. 15) angedeutet, erfolgen dafür zwei parallel laufende Schritte. Zum einen gibt es den „Aim“-Thread, der alle Audiodaten entgegen nimmt und die Steuerung des Positionsdaten übernimmt. Das heißt, hier entscheidet sich, mit wie vielen Mikrofonen die Messung durchgeführt werden soll und welche konkrete Position angesteuert wird. Wie in der Abfolge in Kapitel 5.3 beschrieben, laufen zwei unterschiedliche Instanzen des „Aim“-Threads. Die eine Entität übernimmt dabei, nur mit einer kleinen Anzahl an Mikrofonen, die Überprüfung der großen Abschnitte. Die Positions- und Messwertewerte liegen dafür in dem Feld „bigPositions“ . Die Untersuchung bzw. Messung führt die DSB-Methode durch (Codeausschnitt 5). Sie beinhaltet die beiden Schritte des Delay-and-Sum-Algorithmus, wie er bereits im Abschnitt 6.2 genauer beschrieben wurde. Die Implementation unterscheidet sich nur darin, dass am Ende keine Formatierung in Byte-Werte durchgeführt wird, sondern eine Berechnung der Energie nach den Formeln (2.3.2) und (2.2.2) erfolgt.

```
1 bigPositions[1][j] = dsb((int) bigPositions[0][j],bigBeamformer);
```

Listing 5: Aufruf des DSB mit „bigPositions“

Nachdem alle Energiewerte aus dem Array bestimmt sind, erfolgt ein Vergleich. Der Punkt, mit dem größten Energiewert, wir daraufhin an die zweite Entität weitergereicht. Dort durchlaufen alle Sampledaten im Anschluss noch einmal die Berechnung mit einem zweiten feineren Beamformer.
An dieser Stelle ist in der Implementation kam ein Schlüsselfaktor zum tragen. Und zwar ergeben sich hier keine eindeutigen Messdaten, die hätten an den zweiten Thread weitergereicht werden können. Die gemessenen Positionen sprangen stets unkontrolliert. Da auch kein konkretes Muster zu erkennen war, lässt sich das nur auf die Richtcharakteristik der Hauptkeule schlussfolgern. Daher wäre es hier in einer weiteren Untersuchung angebracht deren Eigenschaften und Lage der Nebenkeulen eindeutig bestimmt. Nur durch die Berechnung des Beampatterns, unter der Anwendung der Vorbetrachtungen aus 5.2, ließe sich so der Beamformer eindeutig verifizieren.
Darüber hinaus wurde in einer Messung der Mikrofone in der Holzplatte festgestellt, dass es zu Verzerrungen des Frequenzgangs in unterhalb von 200Hz und oberhalb von 3000Hz kommt. Der Bereich dazwischen sollten durch einen Bandpass herausgefiltert werden, wodurch die Energieanteile, die nicht für die menschliche Sprache zuständig sind auch herausgefiltert werden. Damit würde man eine genauere Messung erzielen können.

Um die Messdaten und Positionen schnell visualisieren zu können, wurde ein *Java2D*-Grafikboard entwickelt, welches die Messfelder aus der Positionsebene 27 enthält, deren Erstellung in Kapitel 5.3 vorgestellt wurde. Ein Screenshot (dt. Bildschirmfoto) befindet sich im Anhang K. Im Abschnitt 7.2 wird daraufhin eine adäquate dreidimensionale Lösung vorgestellt, die dieses anfängliche zweidimensionale Darstellung ersetzt und dem Benutzer auch mehr Interaktionsmöglichkeiten bietet.

6.4 Validierung der Beamformer Implementation

Eine Überprüfung der Beamformer-Implementation ist unerlässlich. Für diesen Zweck wurde ein Experiment durchgeführt, wobei die eingehenden und ausgehenden Audiodaten vor und hinter dem DSB für jeweils zehn Sekunden aufgenommen werden. Dabei wird das Array auf eine vorher bestimmte Position ausgelenkt. Damit es zu keiner Verfälschung durch fehlerhafte Verzögerungsdaten kommt, erfolgte eine Rechnerische Bestimmung die Verzögerung in einem Tabellenkalkulationsprogramm. Die Berechnung der Tabelle im Anhang 5 erfolgt unter Verwendung der Gleichungen (5.1.3) und (5.1.4) in Bezug auf die Mikrofonpositionsdaten F. Die kalkulierten Delays 5 sollten als Referenz dienen, um zu erkennen, wie lange jede Verzögerung erfolgt. Das Mikrofonarray 2 wurde dabei über den Nullpunkt platziert.

Als Audioquelle für die Aufnahme diente ein Impuls. Dieser ist ein kurzweiliger, akustischer Stoß in Form eines Händeklatschens. Das so entstehende schlagartige Signal garantiert, dass in jeder Audiospur direkt erkannt werden kann, wo das Signal beginnt. Das ermöglicht einen Vergleich der aufgezeichneten Spuren. Die Speicherung für die eingehenden Signale erfolgt in zwei WAV-Dateien mit jeweils 32 Kanälen. Hingegen benötigt die Ausgaben nur einen Monokanal. Nach der Aufnahme erfolgt ein Einlesen der Audiospuren in das Programm *Adobe Audition*. Dadurch, dass das Mikrofon mit der Positionsnummer 10 den größten Abstand von $d_{max} = d_{10} = 5,385m$ besitzt, mussten alle Audiodaten so lange verzögert werden, bis der Schall auch an ihn eingetroffen ist. Eine parallele Darstellung der Kanäle lässt sich für diesen Zweck schnell in *Adobe Audition* realisieren. Der Vergleich ist in der Darstellung 16 zu sehen (Auszug aus Screenshot J).

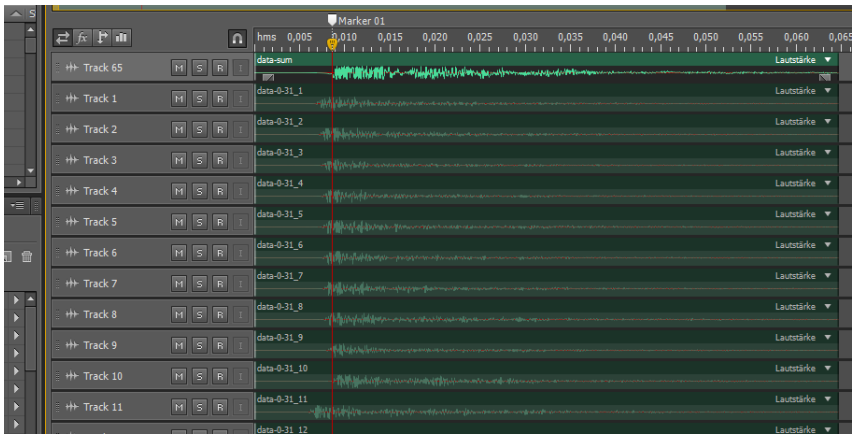


Abbildung 16: Ausschnitt aus Multitrackansicht

In der Gegenüberstellung ist ersichtlich, unter der Betrachtung des *Marker 1*, dass der summierte Ausgabesignal (*Track 65*) erst beginnt, wenn auch die Schallwellen das Mikrofon mit der Nummer 10 (*Track 10*) erreicht haben. Darüber hinaus sind die Audiodaten der Ausgabe auch in einem normalisierten Bereich und nicht übersteuert. Weiterführend lässt so ein Experiment jedoch keine Rückschlüsse auf die Beschaffenheit der Form des Beamformers. Es wird lediglich die Arbeitsweise des DSB-Algorithmuses überprüft. Daher wird empfohlen die genaue Richtcharakteristik nachstehend zu berechnen.

7 3D Visualisierung

7.1 Grundlegendes

Die Lage einer dreidimensionalen Koordinate im Raum lässt sich stets schwer vorstellen. Daher ist es nötig eine Visualisierung zu schaffen, die dem Benutzer genau anzeigt, wo sich ein Objekt oder ein Punkt befindet. Für diesen Zweck wurde eine dreidimensionale Umgebung geschaffen, die das SpeechLab repräsentieren soll. Erstellt wurde die Szenerie mit der Java 3D™ API. Sie bietet dabei alle Hilfsprogramme, die das Erstellen eines solchen Interfaces erleichtern. Die Arbeit erfolgt komplett objektorientiert, wodurch es dem Entwickler ermöglicht wird, in einer relativ kurzem Zeit eine komplett definierte Umgebung zu visualisieren ([Mik08]).

Java 3D™ arbeitet ein wenig anders als andere Programmierschnittstellen für Grafik-APIs. Durch einen übergebenen Szenengraph kann die Grafikhardware nicht direkt angesprochen werden, wie es bei OpenGL oder Direct3D üblich ist. Der Graph enthält alle Objekte der Szene mit einem entsprechendem Beleuchtungsmodell, das nötig ist, um Texturen oder Oberflächeneigenschaften, wie Rundungen, zu erkennen. Auf dieser Grundlage wird eine Graphhierarchie inklusive Abzweigungen erstellt, in der alle Gegenstände zu finden sind. Der Entwickler hat somit die Möglichkeit schneller und effektiver auf einzelne Objekte oder ganze Gruppen zuzugreifen und sie zu transformieren.
Die Gegenstände an sich enthalten dreidimensionalen Koordinaten mit X-,Y- und Z-Ebene. Hierbei steht die Z-Koordinate für die Entfernung von einem Punkt zu dem jeweiligen Betrachtungspunkt. Dabei kommt das „ray tracing“ (dt. Strahlverfolgung) entlang dieser Linie zum Einsatz. Der Algorithmus entscheidet welche Objekte wirklich sichtbar sind und jene die bei der Erstellung der Szene (engl. rendering) vernachlässigt werden können. Dafür erfolgt ein Vergleich aller Z-Koordinden von allen auf einer Linie sich befindlichen Objekten bzw. Punkten. Somit werden mögliche Verdeckungen ermittelt bzw. ausgeblendet.

Um nicht eigene Geometrieobjekte erstellen zu müssen, bringt Java 3D™ einfache Klassenobjekte von den grundlegende Formen mit. Diese sind Kugel (engl. sphere), Quader (engl. box), Kegel (engl. cone) und Zylinder (engl. cylinder) und lassen sich problemlos jeder Zeit durch Angabe von zusätzlichen Größenparametern erzeugen. Durch Objekte des Typ „Appearance“ (dt. Erscheinung) lassen sich zusätzlich eine Vielzahl an Methoden einsetzen, die auf das Aussehen der Objekte Einfluss nehmen. Neben Behandlungsweisen für Farben, Transparenz oder Texturen lassen sich auch Materialeigenschaften simulieren. Das Erstellen der Basisobjekte erfolgt auf Grundlage von polygonalen Netzen, die durch Färbung und Beleuchtung einen dreidimensionalen Eindruck hervorrufen. Um eigene Formen oder Objekte zu entwickeln, wird ein separates Polygonnetz erzeugt, das sich überwiegend aus Dreiecken zusammensetzt. Die Gruppierung erfolgt in einer Shape3D-Klasse. Auf so eine Art und Weise lässt sich jede beliebige Form erstellen und in die Szene einfügen.

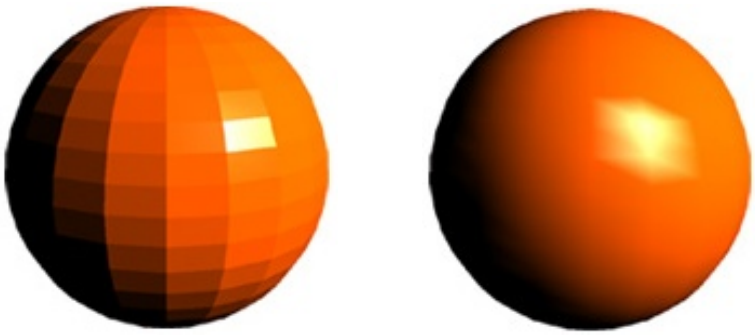
Um einsatzfähige Objekte zum Leben zu erwecken bietet die verwendete API eine Reihe von

dreidimensionalen Transformationsmöglichkeiten, die es dem Programmierer ermöglicht die Lage oder Größe des erzeugten Objekt im Raum zu verändern. So werden beispielhafte Verschiebungen von Punkt $p = (x, y, z)$ nach $p' = (x', y', z')$ durch Translationen vorgenommen:

$$\begin{vmatrix} x' \\ y' \\ z' \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{vmatrix} . \begin{vmatrix} x \\ y \\ z \\ 1 \end{vmatrix} \tag{7.1.1}$$

Die Steuerung weiterer Darstellungseffekte erfolgt über die Beleuchtungen. So setzt sich jeder gefärbte Pixel aus einer Reihe von verschiedenen Beleuchtungsmodellen zusammen. Aus ambienten, defusen und specularen Lichtanteilen wird so für jeden Bereich separat entschieden, in Abhängigkeit des Betrachtungswickels, wie deren Färbung aussehen soll.

Um jedoch nicht nur ebene Formen darzustellen, werden auch noch eine Reihe von Interpolationsverfahren benötigt, die für Realisierung einer sauberen und/oder runden Oberfläche verantwortlich sind. Für diesen Zweck wurden verschiedene *shading methods* (dt. Schattierungsverfahren) entwickelt. Die Java 3D™ API bietet hier das „FLAT-SHADING“ und „GAURAUD-SHADING“ an. Im Vergleich von Darstellung 17 wird hierbei ersichtlich, dass das zweite Schattierungsverfahren realitätsnäher erscheint und somit auch eher Anwendung findet.



(a) Flat (b) Gouraud

Abbildung 17: Vergleich Shading[12]

[12]Quelle: `http://www.dma.ufg.ac.at/app/link/Grundlagen%3A3D-Grafik/module/9728?step=all`

7.2 Realisierung eines eigenen 3D-Szenengraphes

Die eigene Realisierung und Umsetzung der SpeechLab-Szene setzt auf einem Szenengraph auf, der in einem „JFrame“, der Java „Swing“-Klasse, eingebettet ist. Er dient als grafische Benutzeroberfläche (engl. graphical user interface, GUI) und bietet im Wesentlichen alle Interaktionsmöglichkeit mit dem Programm. Die entwickelte GUI-Klasse trägt dabei den Namen „Room3D“.

Die entstandene Szene enthält im Großen und Ganzen nur vier Objekte: Einen Cube (dt. Würfel), eine Box, ein Cylinder und eine Sphere.

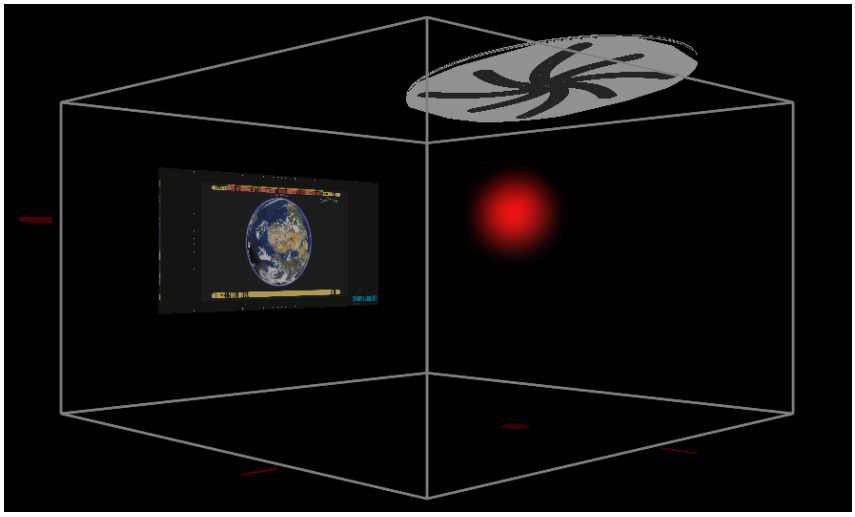


Abbildung 18: Dreidimensionale Visualisierung des SpeechLab

Der Cube soll den Raum, der für die Messungen steht an sich repräsentieren und ist daher vollkommen durchsichtig, um nicht unnötige Verdeckungen durch Flächen oder Linien entstehen zu lassen. Für die Erstellung des Cubes wurde auf die von Daniel Selman entwickelte Klasse „CuboidTest“ zurückgegriffen, die es ermöglichte einen transparenten Würfel zu erzeugen, der keinerlei Querstreben von den enthaltenen Polygonnetzen besitzt. Die verwendete Box wurde auf eine fast ebene Fläche zusammengestaucht und mit einem halbtransparenten Bild des originalen TV-Bildschirms und dem „Mikofonfeld 1“ versehen. So entsteht die Illusion eines echten Gerätes. Eine ähnliche Verarbeitung der *Appeareance* erfolgte auch für das „Mikrofonfeld 2“, welches sich an der Decke befindet. Auch dieses wurde halbdurchsichtig mit einer Textur versehen, die dem Originalem gleich kommt. Jedoch wurde für das Gebilde die Cylinder-Klasse verwendet und ebenso auf eine flache Ebene reduziert. Um nun den Messpunkt mit der höchsten Schallenergie im Raum anzuzeigen wurde eine Sphere verwendet, die durch ganz bestimmte Eigenschaften von außen nach innen an Intensität zunimmt und daher im Kern die größte Punktdichte besitzt. Für diesen Zweck wurde von Michael N. Jacobs in dem Artikel „Star Trek Technology for Java3D“ ein Verfahren beschrieben, um ausgehend von einem Sphere-Objekt ein der Sonne ähnliches Gebilde zu generieren, was den Eindruck von Lichtstrahlen erzeugt.

Beschrieben wurde das im Teil „Implicit Surfaces“ (dt. impliziert Oberflächen). Als Ausgangsmaterial fungiert eine implizite Darstellung einer eindeutigen Sphere. Der Alpha-Kanal einer Textur wird dafür auf ein Objekt gelegt, welches somit eine Struktur erhält. Eine Interpolation der Umgebungsfarbe auf die Vertex- und Texturnormalen, mittels der „MODULATE“-Funktion der Texturattributen, sorgt zusätzlich zu einer Verstärkung des Effektes. Auch wenn die runde Grundfläche der Kugel so als Scheibe zu erkennen ist, werden mit den Vertexnormalen[13] zusammen die Illusion einer leuchtenden Kugel geschaffen. Deren Steuerung des Erscheinungsbild erfolgt unter Zuhilfenahme von Transparenzeffekten. Die Abbildung 19 dient als Veranschaulichung.

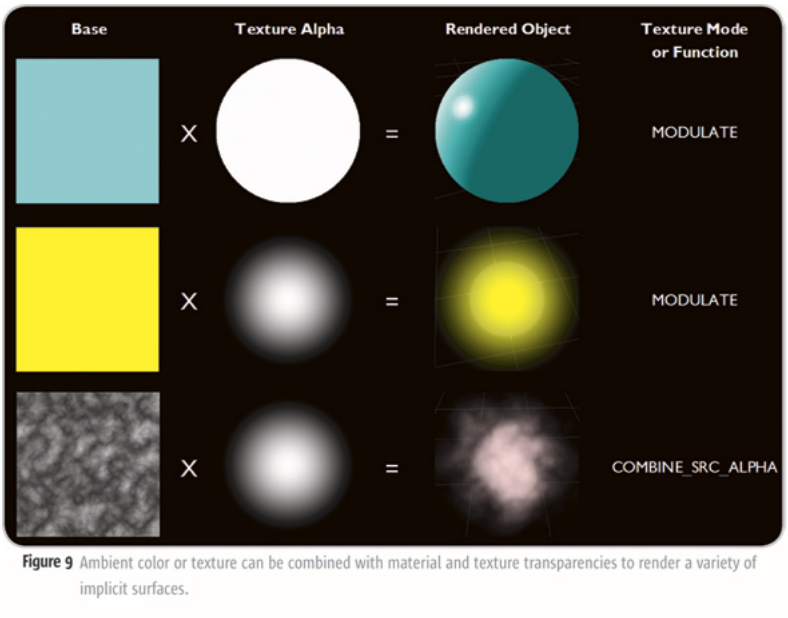


Abbildung 19: Darstellungsmöglichkeiten mit „implicit surface “[14]

Durch die verschiedenen halbdurchsichtigen Objekte in der Szene, ist eine Sortierung notwendig geworden. Die folgende Parameterangabe verbessert somit die Tiefenwirkung der Elemente:

```
1 view.setTransparencySortingPolicy(View.TRANSPARENCY_SORT_GEOMETRY);
```

Listing 6: Sortierung der Transparenzen

Mit der Sortierung werden transparente Objekte von hinten nach vorne arrangiert. Das garantiert eine Verdunkelung entfernter Pixel und eine Zunahme des Effektes für räumliche Tiefe.

Durch die Einbindung der Transformierungsgruppe (engl. tranform group) „rotationGroup“ wird eine perspektivisch korrekte Sicht von allen Seiten auf die Objektgruppe gewährleistet und globale Translationen somit ermöglicht. Die Rotationsgruppe beinhaltet alle gezeichneten Formen, die sich durch Maussteuerung beliebig drehen und wenden lassen.

[13] Normalen auf den Eckpunkten der Polygone
[14] Quelle: Figur 9, [Jac08]

Um die Darstellung annähernd realistisch zu gestalten, wurde zum Schluss ein Beleuchtungsmodell gewählt, bei dem das ambiente Licht gleichmäßig auf allen Texturen mit einem leichten Grauton verteilt. Demgegenüber steht ein intensives, direktes Licht, was alle Farbanteile komplett enthält und aus der Richtung

```
1 Vector3f lightDirection = new Vector3f(4.0f, -7.0f, -12.0f);
```

Listing 7: Beleuchtungsvektor

strahlt. Das Zusammenspiel aus diesen beiden Komponenten sorgt dafür, dass der Inhalt der Szene bestmöglich zu sehen ist.

Für die Repräsentation der eingehenden Positionsdaten wurde die Methode „updatePoint()“ geschaffen, die als Schnittstelle die Audio.getPostion3D() benutzt. Sie läuft als Thread und nimmt alle 50ms neue Messwerte entgegen, um sie folglich auf dem Szenegraphen darzustellen. Die grafische Visualisierung wird im Anschluss an dieses Teilprojekt in die LCARS-Umgebung des SpeechLab mit eingebunden und die Messerwerte in Textform als Schriftlayer über die tatsächlich grafische Oberfläche eingebettet.

8 Messung der Mikrofone

Durch die Einfassung der Mikrofone in einer laminierten Pressspanplatte, war es notwendig geworden, eine Messung Vorzunehmen. Es musste überprüfen werden, wie sich das Einlassen in die Platte auf den Frequenzgang auswirkt. Gemessen wurde mit der Software *EASERA* von der Firma *AFMG Technologies GmbH* und der Durchführungsort war der reflexionsarme Raum des Lehrstuhls für Kommunikationstechnik.

Im Ersten Schritt wurden fünf zufällig ausgewählte Mikrofone miteinander verglichen, um das mit dem besten Frequenzgang auszuwählen zu können. Die Mikrofone waren:

Nr. Messreihe	**Array Nr.**	**Mic-Nr.**	**Serien-Nr.**	**Pegeleindruck**
M0003_S01_R01	1	13	2144	normal
M0004_S01_R01	1	04	2168	zu laut
M0005_S01_R01	1	26	2137	zu leise
M0006_S01_R01	2	55	2183	normal
M0007_S01_R01	2	36 (alt)	2197	defekt

Tabelle 1: Mikrofonauswahl

Bevor mit der Messung jedoch begonnen konnte, musste eine Kalibirerung der *MM1*-Mikrofone erfolgen. Aus diesem Grund wurde ein passgenauer Adapter für den Schallkalibrator\Pistonphon *Brüel & Kjær4231* angefertigt. Verwendet wurde Kalibrierungfrequenz von $1kHz$, die jedoch nicht Einfluss die vollständige Übertragungsfunktion des Mikrofons nimmt. Es erfolgt demzufolge nur eine Angleichung des Schalldrucks für die $1kHz$-Frequenz. Die Sensitivität von $15,0mV/Pa$ wurde aus dem Datenblatt A entnommen. Anschließend erfolgte eine Kalibrierung der Soundkarte durch das Aufnahmesystem. Das erfolgt durch Bildung einer akustischen Rückkopplung, bei dem der Eingang mit dem Ausgang verbunden wird. Positioniert wurden die Mikrofone genau einen Meter vor dem Monitorlautsprecher. Eine separate Kalibrierung des verwendete Regielautsprecher *Geithein ME RL906* war für dieses Teilexperiment nicht notwendig, da es hier nur in erster Instanz um einen Vergleich der Mikrofone untereinander ging.

Für die Messung wurde ein „log-sweep“ mit 21,8s bei 48,0 kHz Abtastfrequenz ausgewählt. Als Signalquelle diente ein 1/f-Rauschen, welches auch als Rosa Rauschen bekannt ist. Es beinhaltet eine normalverteilte Amplitude, dessen Energiedichte umgekehrt proportional zur Frequenz (1/f) abnimmt ([IW13a]).

Das Diagramm in der Abbildung 20 zeigt die Messergebnisse in dBSPL für die verglichenen Mikrofone der Tabelle 1.

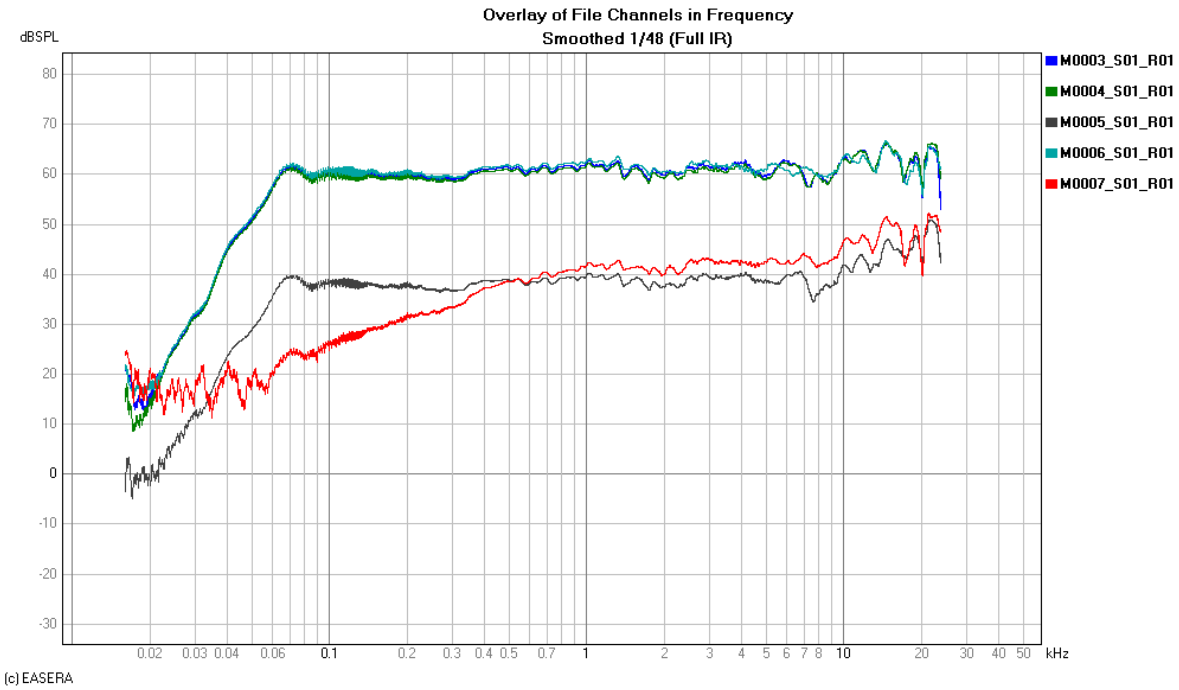


Abbildung 20: Messergebnisse der MM1 Mirkofone

Die vorher angenommenen Pegeleindrücke (siehe Tabelle 1) bestätigten sich größtenteils. So fiel das Mikrofon 26, mit der Seriennummer 2137, negativ durch einen zu geringen Frequenzgang auf. Dieser bewegte sich 20dB unterhalb der anderen Frequenzverläufe. Aber auch das vorher als defekt angenommene Mikrofon, kristallisierte sich als nicht tauglich heraus. Alle Frequenzen unterhalb der 500Hz Marke wiesen hier einen starkes Gefälle auf.
Einen positiven Eindruck machten hingegen die drei anderen Mikrofone. Von denen sogar das Mikrofon mit der Nummer 55 einen gekürzten Mikrofonhals aufwies, der nach der Messung zu Folge keinen Einfluss auf deren Frequenzgang nimmt. Als optimal für den zweiten Schritt der Messung stellte sich der Schallwandler mit der Nummer 13 heraus. Über ihn wurde auch schon im Vorfeld ein optimaler Pegel angenommenen. Auch im Vergleich mit dem geglätteten Frequenzgang des mikrofonspezifischen Datenblatt B, wird hierbei klar, dass das genau den Herstellerangaben entspricht.

Für die Arraykonstruktionen wurde vermutet, dass es durch die Einlassung der Mikrofone in die Holzplatten, zu einem Hub des Frequenzganges um 3dB kommt. Das wurde im zweiten Schritt mit 3 verschiedenen Messungen überprüft:

Nr.	**Nr. Messreihe**	**Aufstellung**	**Plattengröße** in cm	**Foto**
1	N0001_S01_R01	Freifeld	-	Abb. I28
2	N0002_S01_R01	in kleiner Holzplatte	17x40x2	Abb. I 29
3	N0003_S01_R01	in großer Holzplatte	100x100x2	Abb. I 30

Tabelle 2: Messung von Mikrofon in Holzplatte

Es wurden für den Versuchsaufbau zwei verschiedenen Holzplatten mit gleicher Stärke angefertigt, um zu überprüfen, ob die größere Fläche auch zu mehr Reflexionen führen kann. Der Sensitivitätswert des ausgewählte Mikrofon 13, musste auf $14,7 mV/Pa$ angepasst werden (siehe Datenblatt B).
Wie in der Abbildung 21 ersichtlich ist, kam es durch den alleinige Einbau, in einer kleinen Holzplatte, zu einem deutlichen Hub des Frequenzgangs (grüne Kurve). Der Einbau hier hatte dabei überwiegend Einfluss auf den Frequenzbereich zwischen 400 und 6000Hz.
Durch den Einbau in einer noch Größeren Platte, der den Dimensionen des Mikroffeldes 2 entspricht, wurde dieser Effekt zusätzlich verstärkt. In eine große Holzplatte kam es zu einem Anstieg des Frequenzgangs um bis zu 10dB FS. Mit der zunehmenden Masse, lässt sich das auch auf einen größeren Resonanzkörper zurückführen. Darüber hinaus kommt es zu deutlich mehr akustischen Reflexionen auf der Platte, die der Schallwandler ebenso mit erfasst. Das wird besonders in der hohen Frequenzlage der Messung *N0003_S01_R01* ersichtlich. Hier kommt es zu besonderes großen Verzerrungen. Daher sollte der Einsatz eines Bandpasses für die Verarbeitung der Audiodaten in Erwägung gezogen werden. So ein Filter würde alle mittlere Frequenzen passieren lassen, wobei die Höhen und Tiefen gedämpft werden. Das hätte folglich keinen Einfluss auf die Stimmlage, da ein normaler Sprecher in einem Frequenzband zwischen 300Hz und 3400Hz liegt ([IW13b]).

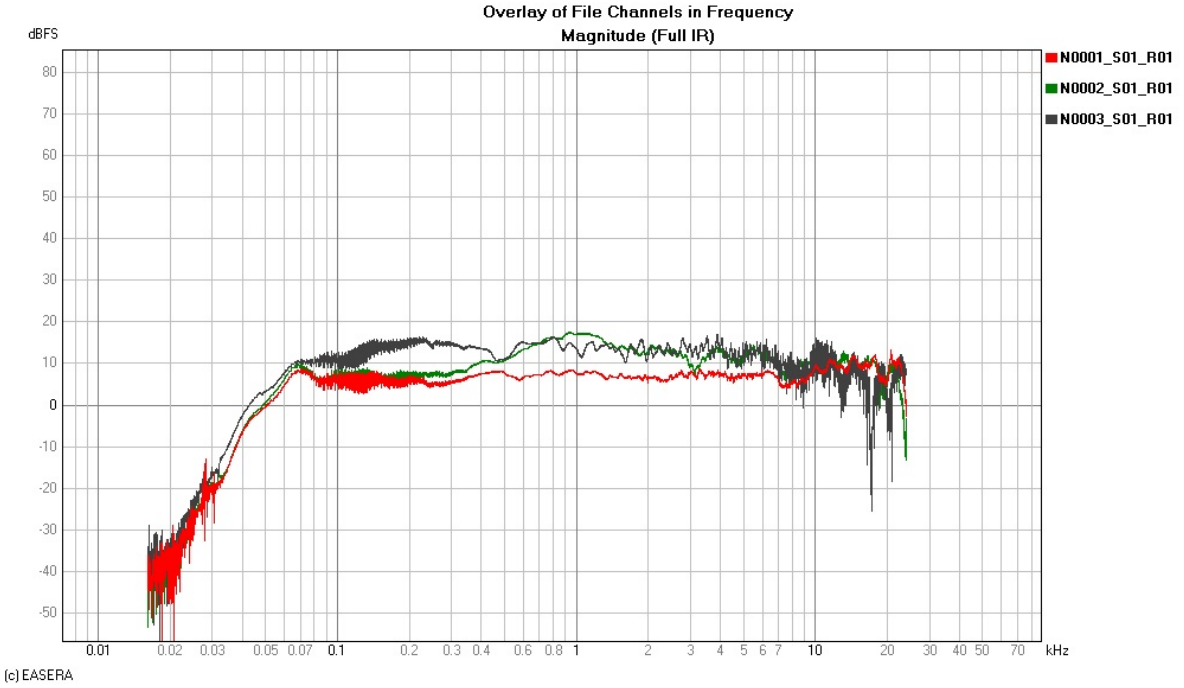


Abbildung 21: Messergebnisse Mikrofone in den Holzplatten

9 Zusammenfassung

In erster Instanz sollte zu dieser Bachelorarbeit ein Audiointerface entwickelt werden. Die primäre Aufgabe lag dabei auf der Entwicklung einer Audioschnittstelle, welche den Echtzeitanforderungen an das Systems gerecht wird. Im Vorfeld wurden viele verschiedene Implementationen getestet und analysiert. Eine ausreichende Leistungsfähigkeit, zur synchronen Unterstützung von mehrkanaligen Audiosystemen, konnte jedoch nur die „open-source“ Softwareschnittstele *jPortAudio* liefern. Dieses befindet sich gegenwärtig im Alpha-Entwicklungsstatus. In dem derzeitigen Zustand läuft das System damit, unter den vorhandenen Testbedingungen, äußerst stabil. Eine Optimierung der Schnittstelle wäre jedoch empfehlenswert, auf Grund der hohen CPU-Last von 50 Prozent beim einlesen der Audiodaten. Weitere Realisierungen von parallel arbeitenden Systemen wie Spracherkenner, grafische Schnittstellen oder Videoverarbeitung würden sich dadurch als schwierig erweisen.

Wie in Kapitel 6.4 gezeigt wurde, funktioniert der Delay-and-Sum-Algorithmus optimal. Eine Nutzung für weitere Projekte, um das Mikrofonfeld herum, ist damit möglich. Eine weitere Optimierung des Suchalgorithmus sowie dessen Implementation sollte jedoch auch nicht kategorisch ausgeschlossen werden. Somit ist eine Berechnung der genauen Richtcharakteristik des jeweilig eingesetzten Beamformers unerlässlich, um so die genaue Beschaffenheit der Richtkeulen zu ermitteln. Ebenfalls haben die Messungen der Mikrofone in der Holzplatte ergeben, dass der Einsatz eines Bandpassfilters notwendig ist. Eine Ausblendung nicht benötigter Frequenzbereiche und eine damit verbundene Senkung der Fehlerquote wären eine vorteilhafte Folge. Die Effizienz des Suchalgorithmus kann des Weiteren, durch die oben genannten Modifikationen, maximiert werden und somit eine annehmbare Fehlerquote erlangen.
Eine eindeutige Lokalisierung des Sprecher im Raum des *SpeechLabs* ist daher theoretisch möglich. Eine Erhöhung der Rechenleistung würde zudem die Option bieten, die Anzahl der parallel laufenden Beamformer drastisch zu erhöhen. Dies hätte wiederum auf die Genauigkeit der Positionsbestimmung Einfluss.

Ein weiterer Schritt zur Verbesserung des DSB-Algorithmus wäre der Austausch der Kernimplementation gegen eine andere Beamformingtechnologie, beispielsweise ein adaptiven Beamformer. Damit wäre es möglich stationäre Störquellen, wie das Geräusch der im Raum angebrachten Klimaanlage vollständig auszublenden. Das geschieht, in dem die Nullstellen zwischen den Nebenkeulen sich direkt an der Position der Klimaanlage befinden. Die dafür notwendige Implementation läuft jedoch nicht im Zeitbereich. Eine Transformierung der Signale in den Frequenzbereich wäre hier die Lösung.

Für die grafische Benutzerschnittstelle wurden zwei unterschiedliche Darstellungen entwickelt. Die einfache Darstellung der Messdaten wurde in einer vollwertigen zweidimensionalen GUI realisiert. Eine verbesserte Visualisierung wird durch das konträre dreidimensionale, interaktive Benutzerinterface geboten. Dieses lässt dem Benutzer viel Platz für zusätzliche kreative Erweiterungen und passt somit optimal in das Gesamtbild der LCARS-Systems.

Schlussendlich wird mit dieser Arbeit ein Grundstein für viele neuartige Anwendungen gelegt, die das Ausgangssignal des Mikrofonfeldes verwenden. Darüber hinaus bietet diese Art der Signalverarbeitung neue Möglichkeiten, welche mit einem personengebundenen Mikrofon nicht realisierbar wären. Es ist nun eine Basis geschaffen worden, die weitere Forschungen innerhalb der Mensch-Maschine-Kommunikation ermöglicht.

Alle in dieser Bachelorarbeit entworfenen Programmcodes sind auf der beiliegenden CD enthalten.

10 Quellenverzeichnis

[BW01] Brandstein, Prof. M. ; Ward, Dr. D.: *Microphon Arrays, Signal Processing Techniques and Applications*. Springer, 2001

[Dre99] Drews, Martin: *Mikrofonarrays und mehrkanalige Signalverarbeitung zur Verbesserung gestörter Sprache*. Microfilm-Vorlesungsskripte, 1999

[Gol04] Goldenbaum, Mario: *Time-Delay-and-Sum-Beamforming an den gemessenen Sensordaten eines Experimental-Sediment-Sonars*, Hochschule Bremen, Diplomarbeit, 2004

[Gre12] Greensted, Andrew: *Delay Sum Beamforming*. `http://www.labbookpages.co.uk/audio/beamforming/delaySum.html`. Version: 2012. – Zugriff: 03.06.2013

[Gö08] Görne, Thomas: *Tontechnik*. Hanser, 2008

[IW13a] IT-Wissen: *Rosa Rauschen*. `http://www.itwissen.info/definition/lexikon/Rosa-Rauschen-pink-noise.html`. Version: 2013. – Zugriff: 07.10.2013

[IW13b] IT-Wissen: *Sprachfrequenz*. `http://www.itwissen.info/definition/lexikon/Sprachfrequenz-VF-voice-frequency.html`. Version: 2013. – Zugriff: 10.10.2013

[Jac08] Jacobs, Michael N.: *Star Trek Technology for Java3D*. `http://mikejacobs.ulitzer.com/node/99792/mobile`. Version: 2008. – Zugriff: 22.09.2013

[Lin11] Lindemann, Jens: *Übungsskript 4 - Analyse von Sprachsignalen*. 2011. – Lehrstuhl Kommunikationstechnik

[Mik08] Mikhalenko, Peter: *So lassen sich mit Java-3D Grafikanwendungen erstellen*. `http://www.zdnet.de/39190724/so-lassen-sich-mit-java-3d-grafikanwendungen-erstellen/?ModPagespeed=noscript`. Version: 2008. – Zugriff: 23.09.2013

[Pap05] Pape, Lutz: *Vergleich Robuster Mikrofonarrays*, Universität für Musik und darstellenden Kunst, Graz, Diplomarbeit, 2005

[RB13] Ross, Bencina ; Burk, Phil: *So lassen sich mit Java-3D Grafikanwendungen erstellen*. `http://portaudio.com/docs/v19-doxydocs-dev/api_overview.html`. Version: 2013. – Zugriff: 23.04.2013

[Sar04] Sarradj, Ennes: Mikrofonarray mit Nahfeld-Beamforming / Gesellschaft für Akustikforschung Dresden mbH. 2004. – Forschungsbericht

Anhang

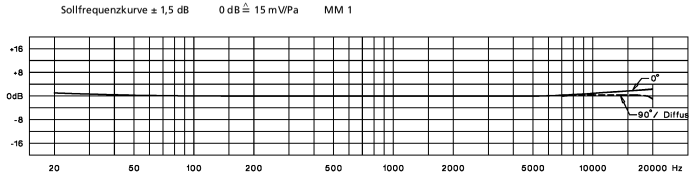
A MM1 Datenblatt

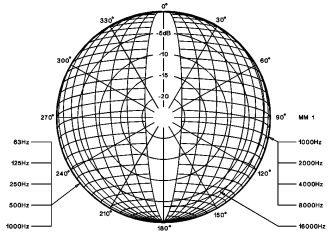
TECHNISCHE DATEN

Wandlerprinzip	Kondensator (Back-Elektret)
Arbeitsprinzip	Druckempfänger
Übertragungsbereich	20 - 20.000 Hz
	(50 - 16.000 Hz ± 1,5 dB)
Richtcharakteristik	Kugel, diffusfeldentzerrt
Feldleerlaufübertragungsfaktor bei f = 1 kHz	15 mV/Pa (≙ -36,5 dBV) ± 1 dB
Nennimpedanz	330 Ω
Nennabschlußimpedanz	≥ 2,2 kΩ
Max. Grenzschalldruckpegel bei	
f = 1 kHz, k = 1%, R_L = 2,2 kΩ	128 dB_{SPL}
Geräuschspannungsabstand bezogen auf 1 Pa	> 57 dB
A-bewerteter Äquivalentschalldruckpegel	ca. 28 dB(A)
Spannungsversorgung	12 - 48 V Phantomspeisung
Stromaufnahme	ca. 3,4 mA
Ausgang	transformatorsymmetrisch
Anschlußstecker	3-pol. XLR
Abmessungen	Länge: 133 mm
	Schaftdurchmesser: 19/9 mm
	Kopfdurchmesser: 9 mm
Gewicht ohne Kabel	88 g

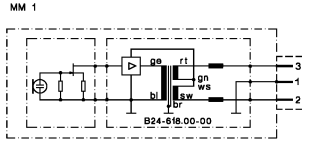
FREQUENZGANG & RICHTCHARAKTERISTIK

Die abgebildete Sollfrequenzkurve (± 1,5 dB) und das Richtdiagramm entsprechen den für dieses Mikrofon typischen Originalmeßwerten.





SCHALTBILD



Abbildungen nicht vertragsbindend. Änderungen vorbehalten. Printed in Germany.

beyerdynamic))))

MM 1

Messmikrofon

Best.-Nr. 449.350

MERKMALE

- Linearer Frequenzgang
- Kugelcharakteristik
- Kalibrierter Feldleerlaufübertragungsfaktor
- Schlankes und robustes Ganzmetallgehäuse

ANWENDUNGEN

Das MM 1 wurde zum Einmessen von ELA- und PA-Anlagen entwickelt. In Verbindung mit einem Spektrum-Analyser können neben Frequenzgangmessungen auch die Schalldruckpegel von Lautsprechern überwacht werden. Bei der Untersuchung von Schallsignalen in Forschung, Entwicklung und in der Industrie leistet das MM 1 wertvolle Dienste.
Dank des schlanken Gehäuses mit der abgesetzten Mikrofonkapsel ist der Einfluß auf das Schallfeld sehr gering, so daß ein Druckstau bei hohen Frequenzen weitgehend vermieden wird. Durch den linearen Frequenzgang wird eine naturgetreue Wiedergabe erreicht.

ZUBEHÖR – OPTIONAL

GST 400	Mikrofonstativ, 3/8", Höhe 0,90 - 1,65 m, mit Schwenkarm G 400	Best.-Nr. 421.294
GST 500	Mikrofonstativ, 3/8", Höhe 0,85 - 1,60 m, mit ausziehbarem Schwenkarm G 500	Best.-Nr. 406.252
ST 400	Mikrofonstativ, 3/8", Höhe 0,90 - 1,65 m	Best.-Nr. 421.286
ST 500	Mikrofonstativ, 3/8", Höhe 0,85 - 1,60 m	Best.-Nr. 406.643
WS 10	Windschutz, Farbe: anthrazit	Best.-Nr. 403.008

Germany
Theresienstr. 8
D-74072 Heilbronn
Tel. +49 (0)7131 / 617-0
Fax +49 (0)7131 / 617-224
E-mail: info@beyerdynamic.de
Internet: www.beyerdynamic.de

United States
56 Central Ave.
Farmingdale, NY 11735
Tel. +1 (631) 293-3200
Fax +1 (631) 293-3288
E-mail: salesUSA@beyerdynamic.com
Internet: www.beyerdynamic.com

Great Britain
17 Albert Drive
Burgess Hill RH15 9TN
Tel. +44 (0)1444 / 258258
Fax +44 (0)1444 / 258444
E-mail: sales@beyerdynamic.co.uk
Internet: www.beyerdynamic.co.uk

[15]Quelle: Beyerdynamic `http://www.beyerdynamic.de/shop/mm-1.html`

B MM1 Datenblatt (Serien-Nr. 2144)

Quelle: MM1 Beyerdynamic Datenblatt aus Mikrofonverpackung

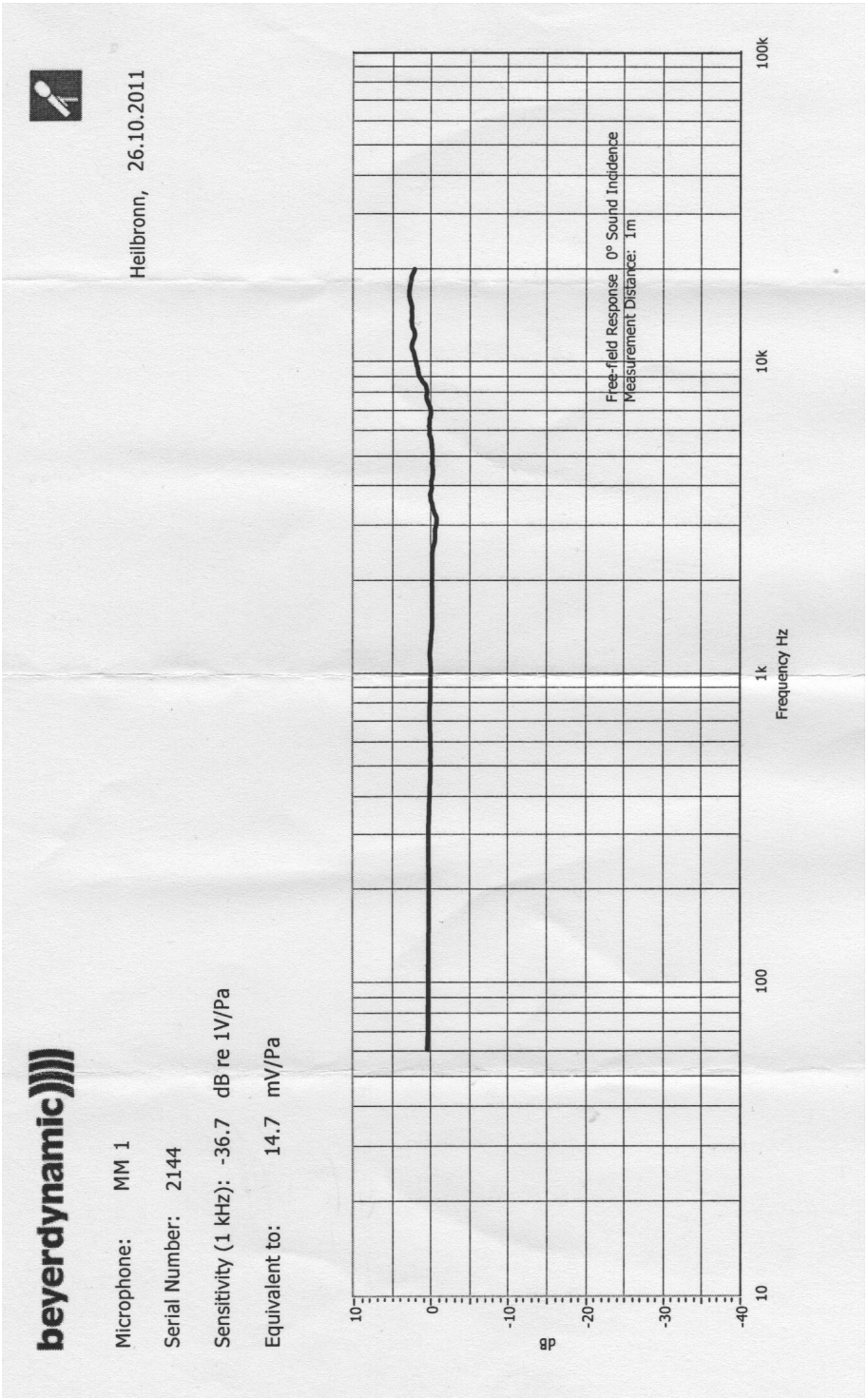


Abbildung 23

C Mikrofonfeld 1 Datenblatt

Quelle: Dipl.-Ing. Thomas Fehér (TU Dresden)

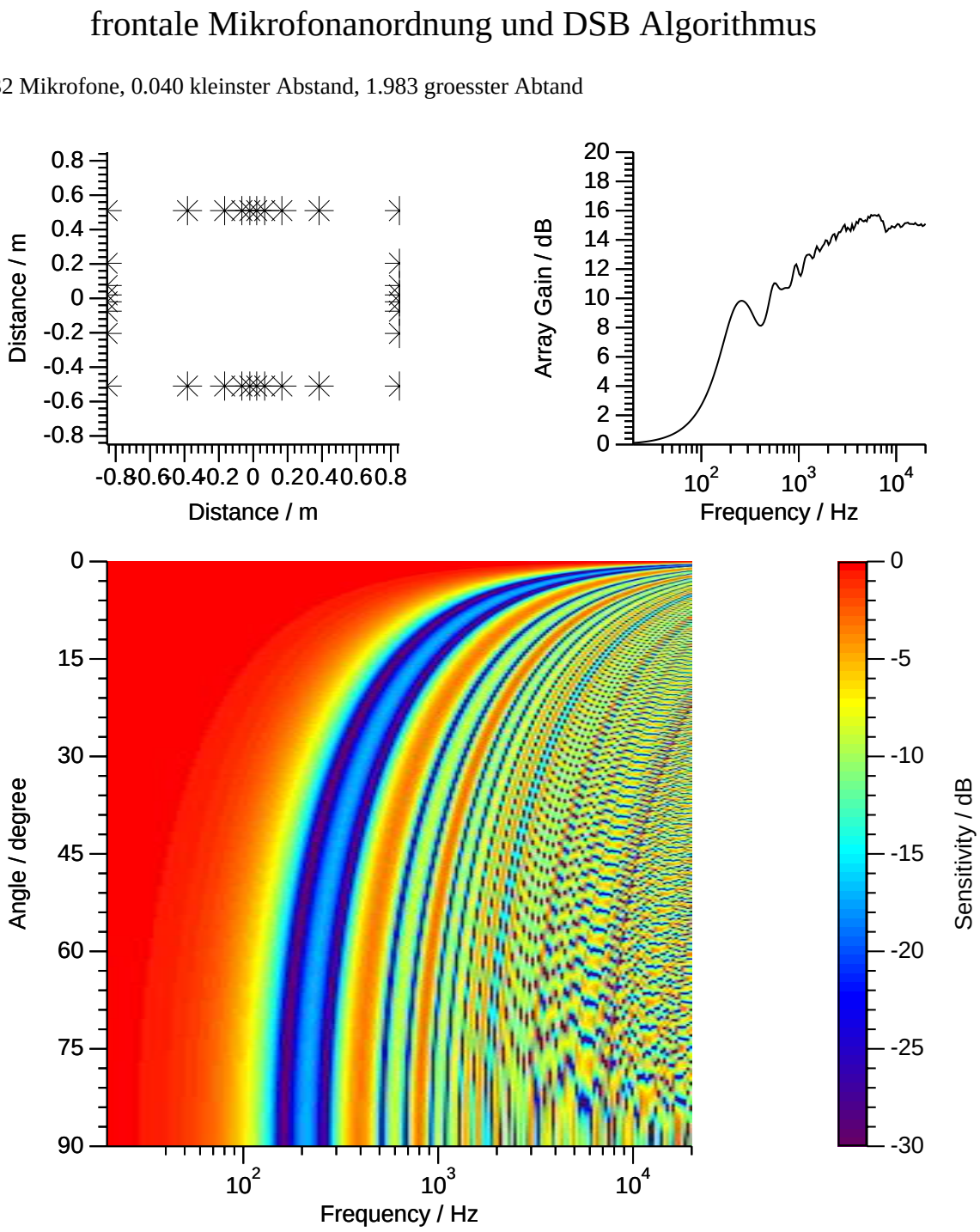


Abbildung 24

D Mikrofonfeld 2 Datenblatt

Quelle: Dipl.-Ing. Thomas Fehér (TU Dresden)

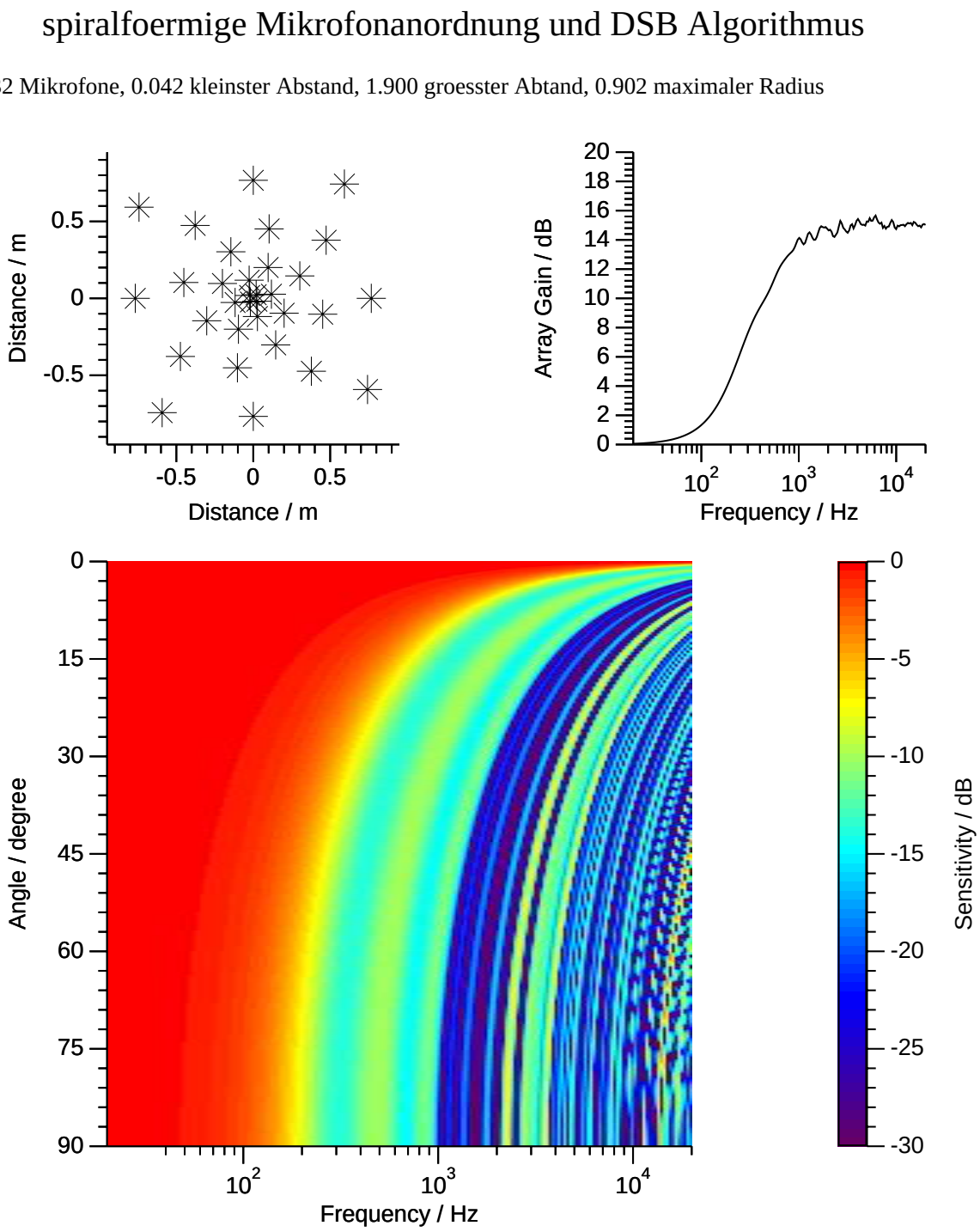


Abbildung 25

E SpeechLab Grundriss

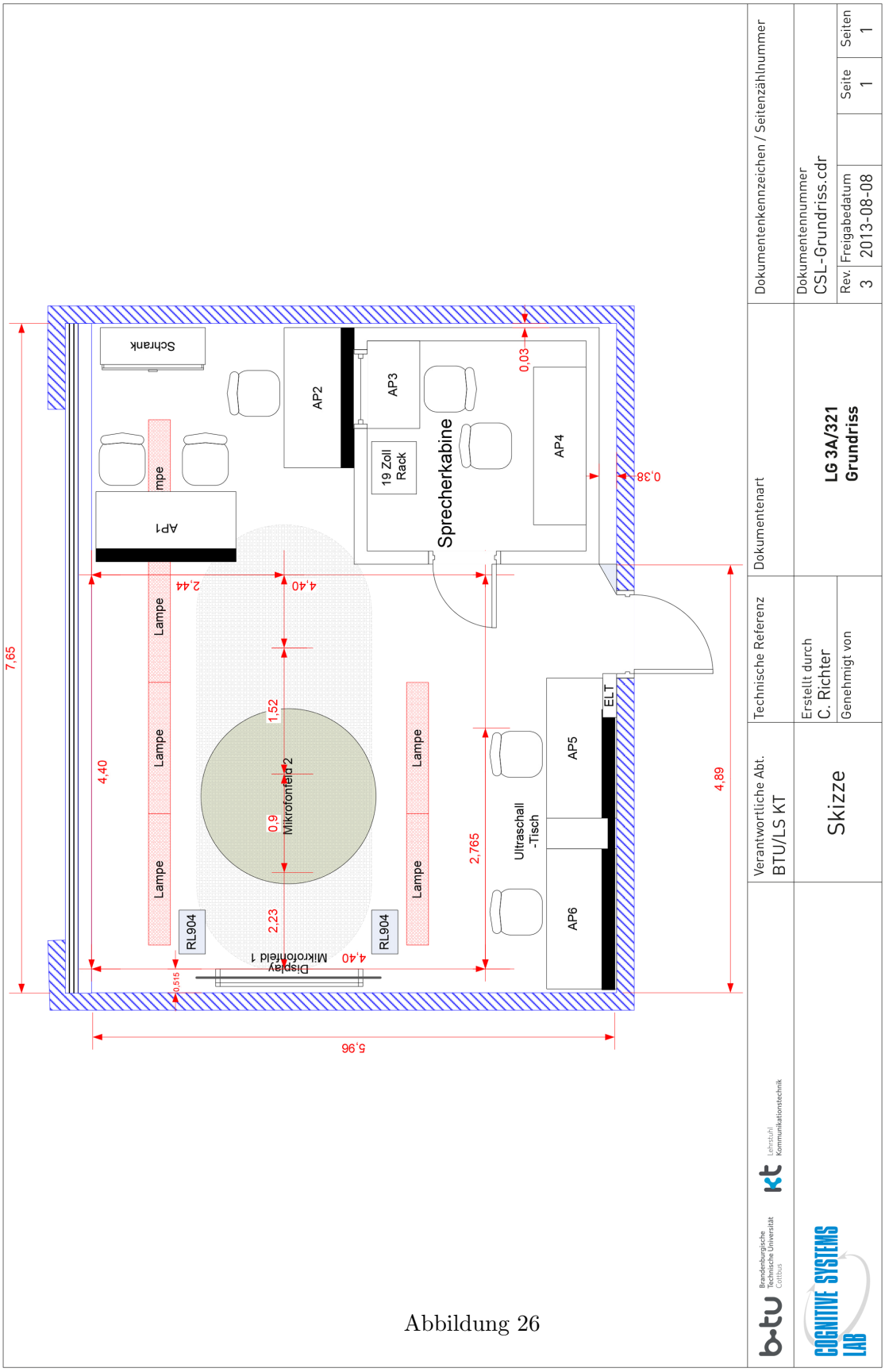


Abbildung 26

F 3D Positionsdaten der Mikrofone

Nr	X in m	Y in m	Z in m
1	-0,850	2,200	2,050
2	-0,383	2,200	2,050
3	-0,176	2,200	2,050
4	-0,066	2,200	2,050
5	-0,020	2,200	2,050
6	0,020	2,200	2,050
7	0,066	2,200	2,050
8	0,167	2,200	2,050
9	0,383	2,200	2,050
10	0,850	2,200	2,050
11	-0,850	2,200	1,030
12	-0,383	2,200	1,030
13	-0,167	2,200	1,030
14	-0,066	2,200	1,030
15	-0,020	2,200	1,030
16	0,020	2,200	1,030
17	0,066	2,200	1,030
18	0,167	2,200	1,030
19	0,383	2,200	1,030
20	0,850	2,200	1,030
21	-0,850	2,200	1,336
22	-0,850	2,200	1,465
23	-0,850	2,200	1,520
24	-0,850	2,200	1,560
25	-0,850	2,200	1,615
26	-0,850	2,200	1,744
27	0,850	2,200	1,336
28	0,850	2,200	1,465
29	0,850	2,200	1,520
30	0,850	2,200	1,560
31	0,850	2,200	1,615
32	0,850	2,200	1,744

Tabelle 3: Positionen Mikrofonfeld 1 (TV)

Nr	X in m	Y in m	Z in m
33	0,023	0,019	2,296
34	0,118	-0,027	2,301
35	0,097	-0,200	2,317
36	-0,146	-0,303	2,327
37	-0,451	-0,103	2,308
38	-0,474	0,378	2,262
39	0,000	0,767	2,225
40	0,743	0,592	2,242
41	0,019	-0,023	2,300
42	-0,027	-0,118	2,309
43	-0,200	-0,097	2,307
44	-0,303	0,146	2,284
45	-0,103	0,451	2,255
46	0,378	0,474	2,253
47	0,767	0,000	2,298
48	0,592	-0,743	2,369
49	-0,023	-0,019	2,300
50	-0,118	0,027	2296
51	-0,097	0,200	2,279
52	0,146	0,303	2,269
53	0,451	0,103	2,288
54	0,474	-0,378	2,334
55	0,000	-0,767	2,371
56	-0,743	-0,592	2,355
57	-0,019	0,023	2,296
58	0,027	0,118	2,287
59	0,200	0,097	2,289
60	0,303	-0,146	2,312
61	0,103	-0,451	2,341
62	-0,378	-0,474	2,343
63	-0,767	0,000	2,298
64	-0,592	0,743	2,227

Tabelle 4: Positionen Mikrofonfeld 2 (Decke)

G Verzögerungsdaten Beispiel

Ziel-Position

Nr	**x** in m	**y** in m	**z** in m
1	-2,000	-2,000	0,250

Verzögerung

Mic-Nr	**d_i** in m	**k_i** in Sample	**k_i** in Sample	**Mic-Nr**	**d_i** in m	**k_i** in Sample	**k_i** in Sample
1	4,7120	86,46969649	86	33	3,5152	240,1322492	240
2	4,8471	69,1127972	69	34	3,5474	235,9927044	236
3	4,9234	59,31977001	59	35	3,4512	248,3480984	248
4	4,9619	54,37791623	54	36	3,2605	272,8287382	273
5	4,9800	52,05265309	52	37	3,1990	280,7361916	281
6	4,9960	49,99333093	50	38	3,4688	246,0884453	246
7	5,0148	47,58260647	48	39	3,9443	185,031428	185
8	5,0573	42,13239782	42	40	4,2673	143,5597447	144
9	5,1535	29,77380746	30	41	3,4913	243,2041542	243
10	5,3854	0	0	42	3,4170	252,7405148	253
11	4,4239	123,4553381	123	43	3,3308	263,8048656	264
12	4,5676	105,0028201	105	44	3,4093	253,7254893	254
13	4,6485	94,62063399	95	45	3,6915	217,4949723	217
14	4,6892	89,38896778	89	46	3,9734	181,2985549	181
15	4,7084	86,92902805	87	47	3,9814	180,274305	180
16	4,7253	84,75131919	85	48	3,5761	232,3166918	232
17	4,7452	82,20305256	82	49	3,4692	246,0368707	246
18	4,7900	76,44597391	76	50	3,4402	249,7565459	250
19	4,8915	63,41182782	63	51	3,5467	236,0899185	236
20	5,1353	32,11732927	32	52	3,7399	211,2784937	211
21	4,4880	115,2286225	115	53	3,8190	201,1188096	201
22	4,5209	110,9980678	111	54	3,6187	226,836724	227
23	4,5360	109,0604399	109	55	3,1654	285,0469034	285
24	4,5474	107,6016443	108	56	2,8269	328,5044714	329
25	4,5635	105,5282213	106	57	3,4933	242,9455088	243
26	4,6038	100,362558	100	58	3,5699	233,1134545	233
27	5,1906	25,0171226	25	59	3,6599	221,5535626	222
28	5,2191	21,35583879	21	60	3,6046	228,6492473	229
29	5,2322	19,67670865	20	61	3,3459	261,8751578	262
30	5,2420	18,41161358	18	62	3,0564	299,0435909	299
31	5,2560	16,61215669	17	63	3,1170	291,2661511	291
32	5,2910	12,12225153	12	64	3,6629	221,1640963	221
				d_max	5,385		

Tabelle 5: Beispiel Delaydaten nach den Formeln (5.1.3) und (5.1.4)

H Messebene

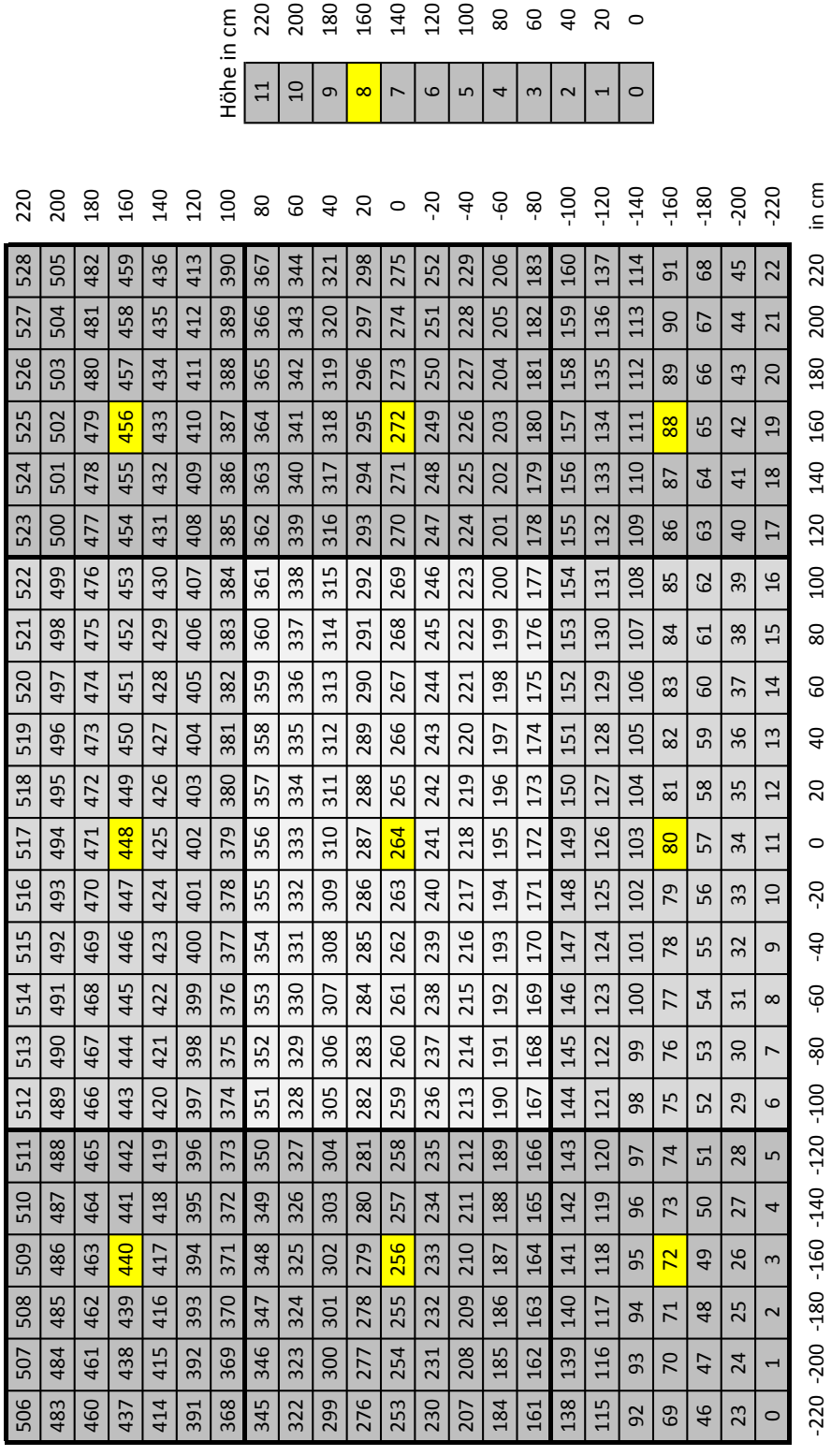


Abbildung 27: Positionsnummern der Messfeldebene

I Fotografien

Abbildung 28: Freifeldmessung

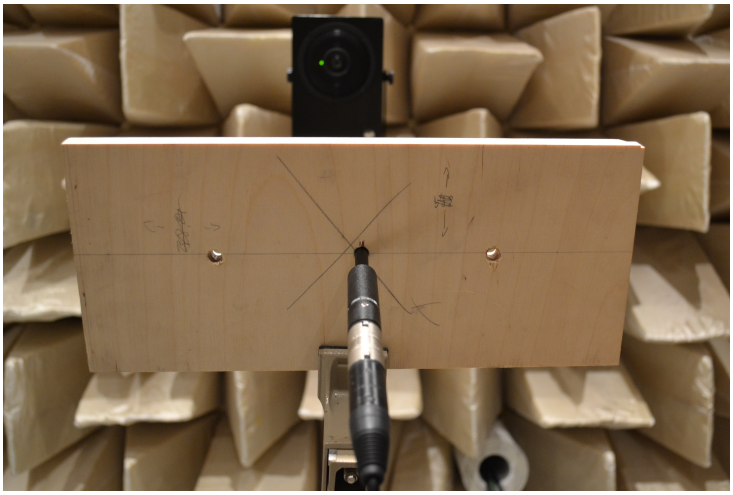


Abbildung 29: Messung in kleiner Holzplatte (17x40x2cm)

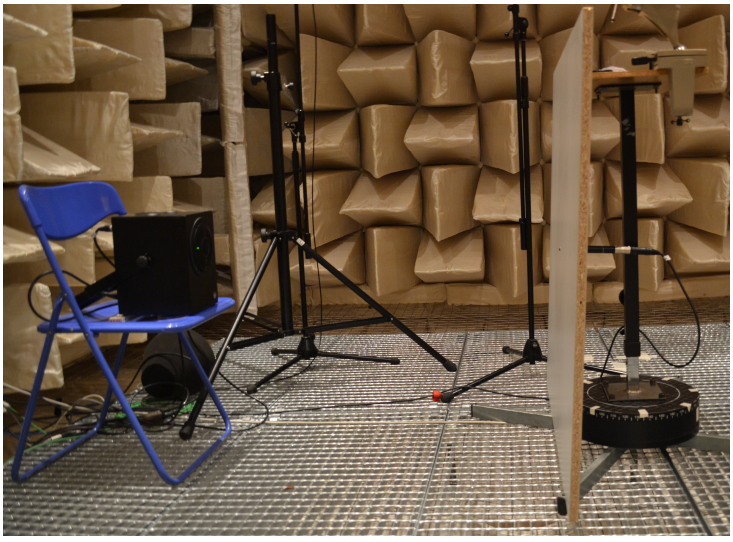


Abbildung 30: Messung in großer Holzplatte (100x100x2cm)

J Screenshots 1

Abbildung 31: Multitrackansicht aus Adobe Audition

K Screenshots 2

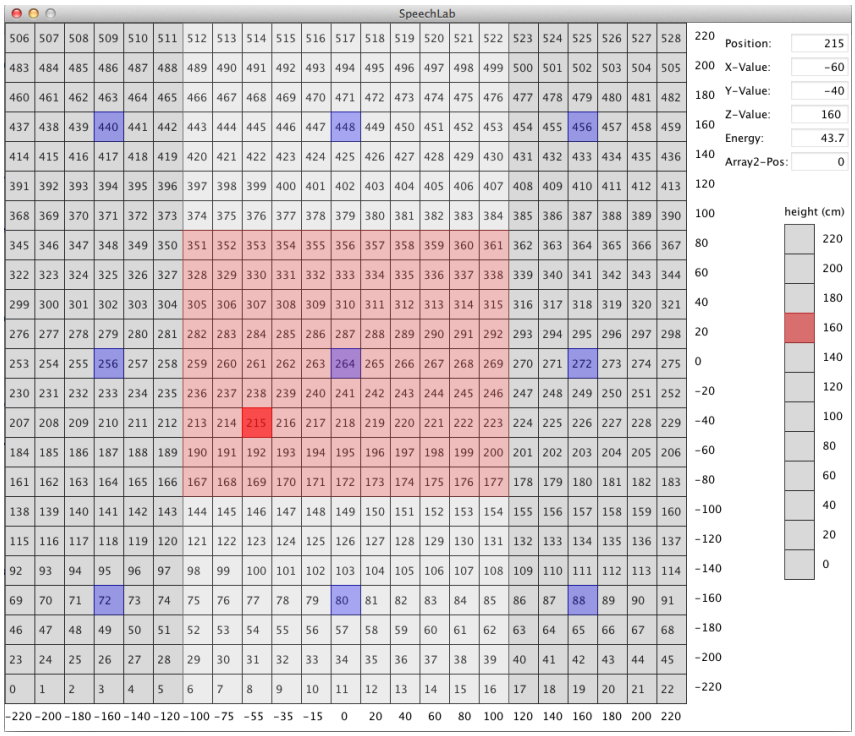


Abbildung 32: Screenshot 2D GUI

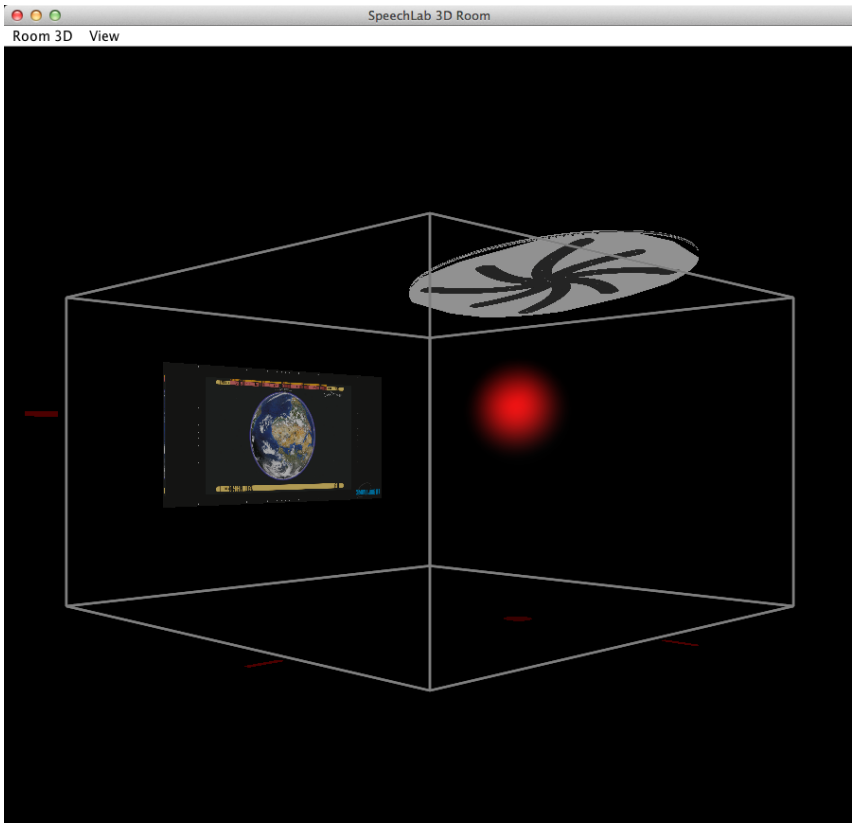


Abbildung 33: Screenshot vom Room3D GUI

Selbstständigkeitserklätung

Der Verfasser erklärt, dass er die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Ort, Datum: Cottbus, den 15. Oktober 2013 .....

(Unterschrift)