DIPLOMARBEIT

Konzeption und Realisierung der Software

zur Matrizierung einer Blumleinaufnahme in das 5.0 Format

von
Ralf Willner
Matr.Nr.14204

Erstprüfer: Prof. Oliver Curdt
Zweitprüfer: Prof.Uwe Schulz
Eine Diplomarbeit im Studiengang Audiovisuelle Medien
Fakultät Electronic Media
Hochschule der Medien, Februar 2008

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die hier vorliegende Arbeit selbständig und ohne andere als die angegebenen Quellen und Hilfsmittel ausgearbeitet habe. Die aus fremden Quellen direkt oder indirekt übernommenen Zitate und Gedanken sind als solche gekennzeichnet.

Stuttgart, den _____
Datum

_____
Unterschrift

Inhaltsverzeichnis

1 Einführung

1.1 Entwicklung und Geschichte

Die Welt ist Klang. In vielen Schöpfungsmythen unsere Erde beginnt das Leben und damit die Existenz des Seins durch Klang. *„Im Anfang ward das Wort“* ist im Johannesevangelium zu lesen. Die Silbe *„Om“*[1], der transzendenten Urklang indischer Religionen, aus dessen Vibrationen die Welt entstand, Kosmologen, welche im Urknall den Ausgangspunkt unseres Universums sehen. In einem japanischen Schöpfungsmythos wurde die Sonnengöttin Amaterasu aus ihrer Höhle durch Klänge einer Harfe gelockt und erleuchtete die Welt mit Licht.[2] Bereits *Pythagoras* hatte sich im 6. Jahrhundert v. Chr. mit Tonalität und Harmonie befasst und definierte das Verhältnis uns bekannter Intervalle. Ebenso verfasste *Johannes Kepler* im Jahre 1619 *„Harmonices Mundi libri V*“, worin das dritte Keplersche Gesetz beschrieben wird.[3] Danach ist das Verhältnis *d* der dritten Potenz der durchschnittlichen Entfernung eines Planeten von der Sonne, zum Quadrat seiner Umlaufzeit stets unveränderlich. Er glaubte, dass es eine musikalische Harmonie enthülle, die der Schöpfer im Sonnensystem verewige. Somit hatte Klang und Musik schon immer einen besonderen Stellenwert in unserem Leben. So wurde am 18. Juli 1877, von *Thomas Alva Edison*, der Phonograf erfunden. Dieser Phonograf war in der Lage Schallwellen auf eine staniolbeschichtete Walze aufzuzeichnen und wieder abzutasten[4]. Der Startschuss in die Klangreproduktion war gegeben. Die Aufnahmen waren vergleichbar schlecht und hatten einen geringen Rauschabstand. Durch Entwicklung der Elektronenröhre im Jahre 1904 konnten Schalltrichter durch Mikrofone und Schläuche mit Kabel ersetzt werden. Der auf dem elektrostatischen Prinzip beruhende Statophon-Lautsprecher wurde für die Tonwiedergabe bei Filmvorführungen entwickelt. Der Drang, Zuhörer in möglichst reale Klangräume zu transportieren, war groß. Dies konnte mit nur einem Mikrofon und/oder Lautsprecher nicht simuliert werden. 1933/34 experimentierte die *Bell Telephone Company* mit ersten

[1] http://de.wikipedia.org/wiki/Om

[2] http://www.mushroom-online.com/

[3] http://de.wikipedia.org/wiki/Johannes_Kepler

[4] Surround, Einführung in Mehrkanalton-Technik ab S.12

Mehrkanalaufzeichnungen. Die Premiere einer öffentlich aufgeführten Stereodarbietung wird allgemein Walt Disneys Film „*Fantasia*“ zugeschrieben, dieser beinhaltete 7 Tonspuren und wurde mit dreißig Lautsprechern wiedergegeben. 1950 begann man auch im Rundfunk stereofon zu übertragen und erst 1970 ging man dazu über, Zweikanalstereo im Fernsehen zu senden. Ungefähr zur gleichen Zeit wurde an der Quadrofonie gearbeitet. Ein Mehrkanalverfahren bei dem mit vier diskreten Kanälen gearbeitet wurden. Die Lautsprecher werden jeweils im *90°* Winkel zueinander angeordnet. Von *Pink Floyd* wurden Aufnahmen mit dieser Technik produziert, bei denen die einzelnen Schallquellen im Vierkanalspektrum verteilt wurden. Zur weiteren Verbesserung der Raumabbildung wurden zwei rückwärtige Kanäle hinzugefügt. Dieses Verfahren blieb aber glücklos. Im Kino folgten diverse Mehrkanalformate mit der Absicht den Zuschauer möglichst realistisch in das Geschehene zu versetzen. Der Center für die getreue Dialogortung wurde eingeführt. Vor allem *Dolby*[5] war führend in der Entwicklung neuer Konzepte und Formate. Worauf *George Lucas*[6] sein *THX* Konzept aufbaute, welches auch heute noch als einer der besten Tonstandards im Bereich Kinoton gilt[7].

[5] Dolby Labaratories Untrenehmen von Ray Dolby in den U.S.A gegründet

[6] George Walton Lucas Jr. (* 14. Mai 1944 in Modesto, Kalifornien, USA)

ist ein US-amerikanischer Produzent, Drehbuchautor und Regisseur

[7] Surround, Einführung in Mehrkanalton-Technik ab S.20

2 Konzept und Idee

Die Idee dieser Arbeit, beruht auf verschiedenen Konzepten zum einen auf einer Arbeit von *Thilo Thiede*, *Werner Schaller* und *Johannes Hensel*, welche die Aufnahme und Wiedergabe eines räumlichen Schallfeldes mittels verschiedener Mikrofonanordnungen beschreibt (Orthophonie), zum anderen dem Surround-Mikrofon der Firma *Soundfield* sowie einem diskreten B-Format-Mikrofon. Das Konzept dieser Diplomarbeit beruht auf der These, dass aus einer Blumlein- Stereoaufnahme, mit orthogonal ausgerichteten Achten, unendlich viele Achten mit der Drehung um den Winkel α, berechnet werden können. Dieser These soll mathematisch, grafisch sowie messtechnisch auf den Grund gegangen werden. Als Resultat soll eine Software entstehen, die ein Blumlein-Verfahren in ein 5.0 Format umgerechnet. Die in den kommenden Kapiteln folgenden Konzepte dienen zur Informationsgewinnung und geleiten den Leser in den Bereich der Mikrofonmatrizierung.

2.1 Das Prinzip der Orthophonie

Das erste Verfahren verwendet Mikrofone mit einer Richtcharakteristik 2.Ordnung. Eine Richtcharakteristik 2. Ordnung lässt sich durch eine zweifache Differenzbildung zwischen räumlich versetzten Elementarmikrofonen nullter Ordnung, also Empfängern mit Kugelcharakteristik, erzeugen (Abb. 1).

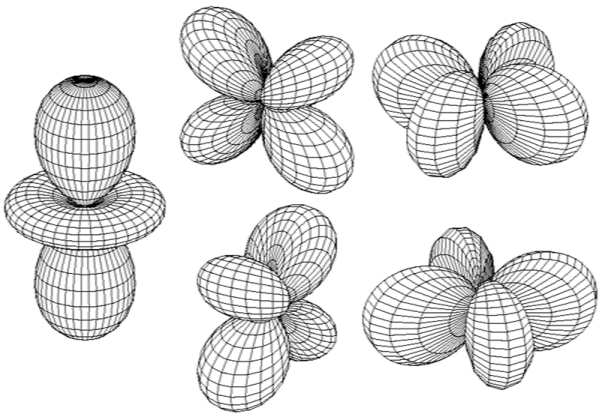

Abb.1 Die fünf Grundcharakteristiken zweiter Ordnung[8]

Diese Elementarmikrofone können dabei auf einer Geraden, in einer Ebene oder im Raum verteilt angeordnet sein (*Abb.2*). Das bedeutet, dass sich jede beliebig im Raum ausgerichtete

[8] http://elvera.nue.tu-berlin.de/files/0853Thiede1999.pdf S.3

Richtcharakteristik n-ter Ordnung mittels einer begrenzten Anzahl von aufgenommenen Signalen simulieren lässt (*Mikrofonarrays*)[9].

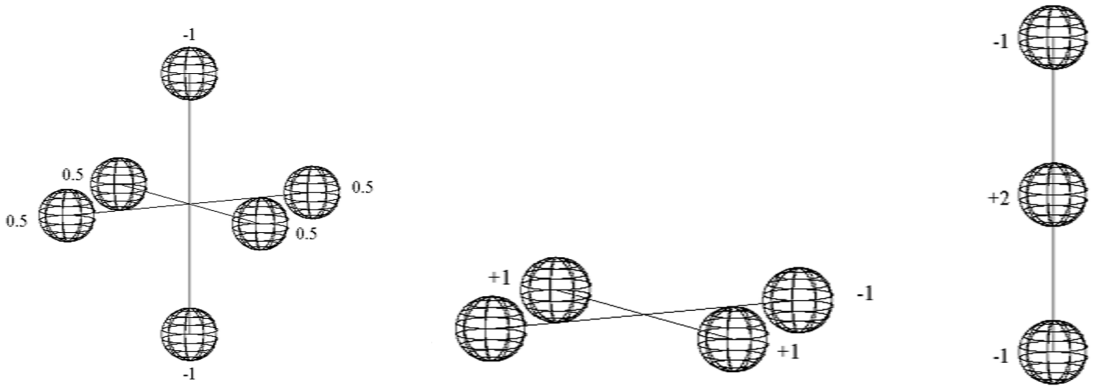

Abb. 2 räumliche Anordnung ***ebene Anordnung*** ***lineare Anordnung***

Daraus folgt natürlich das n-beliebige Lautsprecher mit den jeweiligen Richtcharakteristiken versorgt werden können und daher diese mathematisch hergeleitete Aufnahmetechnik aufwärtskompatibel, bis hin zu den Lautsprecherarrays der Wellenfrontsynthese[10] ist. Die Nachteile dieses Konzeptes sind, das Mikrofone zweiter Ordnung einen Hochpasscharakter besitzen, der durch einen entsprechenden Filter korrigiert werden müsste und dadurch den Rauschabstand verkleinern würde. Zusätzlich müsste die Anordnung höchst symmetrisch angeordnet sein.

2.2 Soundfield-Mikrofon

Das Soundfield-Verfahren basiert auf der Verwendung eines Spezialmikrofons. Dieses Mikrofon besteht aus vier Membranen, die in der Form eines Tetraeders angeordnet sind. Aus den Signalen der vier Kapseln wird das A-Format gewonnen, welches dann in vier neue Signale W, X, Y und Z matriziert wird. Das W-Signal entspricht einer Kugelcharakteristik, die anderen drei einer Achtcharackteristik. Diese vier Signale werden als B-Format bezeichnet. Das X-Signal entspricht einem Links-Rechts-Signal, das Y-Signal einer Tiefenkomponente, das Z-Signal der Höhe und das W-Signal einer im Zentrum platzierten Kugel.[11] Auch dieses Format ist aufwärtskompatibel und kann später bei der Bearbeitung

[9] http://elvera.nue.tu-berlin.de/files/0853Thiede1999.pdf S.5

[10] Die Wellenfrontsynthese (abgekürzt WFS) ist ein Verfahren, mit dem ein Schallfeld eines bestimmten Raumes realistisch nachgebildet werden soll

[11] Surround, Einführung in Mehrkanalton-Technik ab S.93

noch bis auf ein n-beliebiges Mehrkanalformat weiterverarbeitet werden, d.h. das Format muss zur Aufnahmezeit noch nicht bekannt sein. Man ist hier natürlich an einen Hersteller gebunden. Dieser Gedanke wurde in einem diskreten B-Format-Mikrofon weiterentwickelt. Nur dass auf das Z-Signal verzichtet wird und dadurch auf einen Kanal. Dieses Verfahren entspricht am ehesten dem in dieser Arbeit beschriebenen Konzept.

Abb.3 Soundfield-Mikrofon mit Tetraederkapsel[12]

2.3 Surrounder-Konzept

Mir der These aus mehreren Mikrofonen andere Mikrofone uns somit auch andere Richtcharakteristiken zu matrizieren, entstand der Grundgedanke der Software *„Surrounder 2to5“*. Aus zwei Kanälen einen Upmix auf das 5.0 Format zu berechnen klingt schon fast abenteuerlich, bereits bei Probeaufnahmen und ersten Versuche stellte sich heraus, dass dies trotzdem möglich ist. Das die Qualität natürlich schlechtere Ergebnisse liefern wird, als fünf gut positionierte Surroundmikrofone, war klar. Kein Toningenieur oder Tonmeister wird klassische Musik mit der Blumleintechnik als Hauptmikrofon aufnehmen, interessanter ist die Idee für denjenigen, der mit wenig Equipment gute Stereoaufnahmen fertigen will und diese eventuell auch noch brauchbar auf 5.0 upmixen will. Wenn man daher die Blumlein-Stereoaufnahmetechnik betrachtet, in der eine XY-Aufnahme in eine MS-Aufnahmen, und ebenso eine MS-Aufnahme wieder zurück in eine XY-Aufnahme berechnet werden kann, will man sich diesen Vorteil für andere Matrizierungsmöglichkeiten zu Nutze machen. Dieses Prinzip beruht auf zwei Mikrofonen mit Acht-Charakteristik die einander orthogonal angeordnet sind.

[12] http://www.soundfield.com/soundfield/soundfield.php

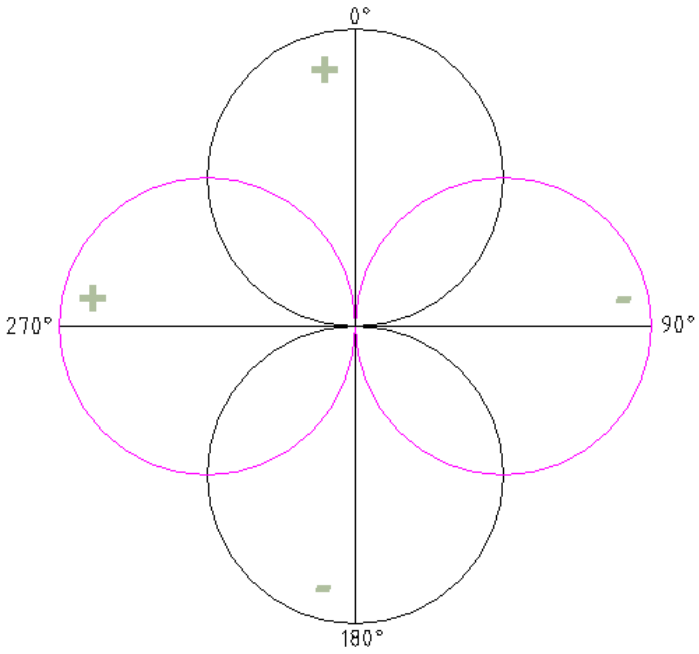

Abb.4 Blümleinstereofonie MS mit zwei Mikrofonen (schwarz/rosa) mit Phasigkeiten der Signale

Diese Achten nehmen Schall von vorne und von hinten auf, wobei der Schall von hinten die Mikrofonmembran gegenphasig auslenkt. Bei einem Schalleinfall von der Seite gibt es keine Druckunterschiede an der Membranvorder- oder Rückseite. Es gibt daher auch keine Membranauslenkung. Dieses Prinzip ermöglicht die MS-Technik. Ordnet man zwei Achten orthogonal an, bilden sie in der Summe die Charakteristik einer Kugel, nehmen also Schall aus allen Richtungen auf (*siehe Abb.4*). Für jede Richtung lässt sich nun ein Signal durch Addition der beiden Achten, mit dementsprechenden Faktoren, herstellen. Fünf Signale mit dem Winkel *+/-30°* (Links/Rechts), 0° (Center), *+/-105°-115°* (Hintenlinks/Hintenrechts) gemäss der *ITU-775* (siehe Kapitel 3.2) sollen dadurch erzeugt werden.

2.3.1 Mathematischer Beweis und die Acht aus zwei Kreisen?

Betrachtet man die Winkelfunktionen und den Einheitskreis genauer und wendet dieses Wissen auf diese Acht-Charakteristik an, die ja aus zwei Kreisen zusammengesetzt ist, lässt sich jeder x-beliebige Vektor *sa* mit dem Winkel α aus den Richtcharakteristiken der beiden Mikrofone ableiten (*Abb.5*). Der Anteil, der nach vorne gerichteten Acht, ist *sa1, sa2* beschreibt den Anteil der seitlichen Acht. Die hier benutzten Winkelfunktionen für *so=1* (Einheitskreis) lauten:

Gleichung (1)[13] $\cos\alpha = sa1$

Gleichung (2) $\sin\alpha = sa2$

[13] Handbuch der Tonstudiotechnik, Johannes Webers S.248

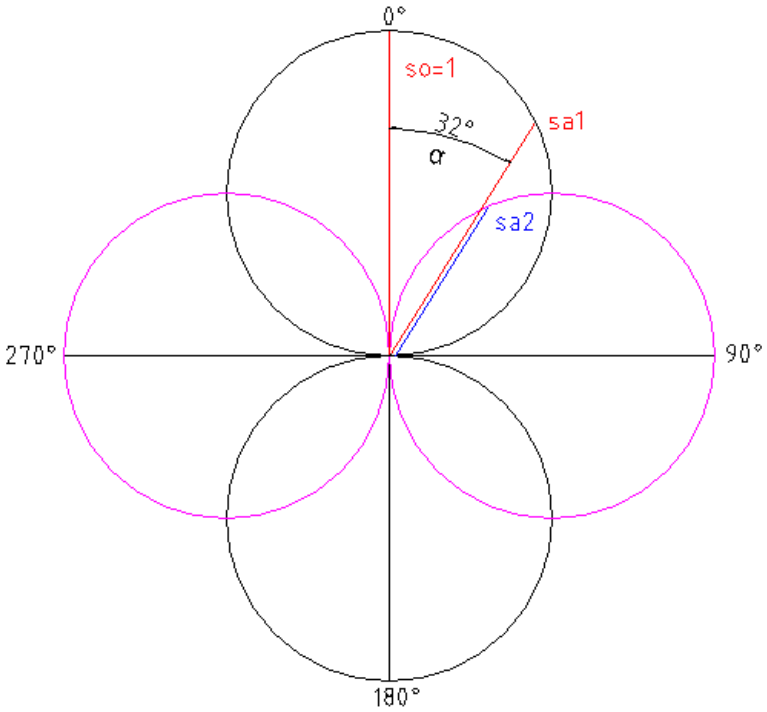

Abb. 5 Gewinnung des Vektoren für den Winkel α

Der Satz des Pythagoras besagt das;

Gleichung (3) $\sin^2\alpha + \cos^2\alpha = 1$

Setzt man nun Gleichung (1) und Gleichung (2) in Gleichung (3) erhält man eine neue Gleichung, die aus Anteilen der beiden Mikrofonsignale *sa1* und *sa2* im Quadrat ein neues Signal *sa*, berechnet, wobei der Verstärkungsfaktor 1 und damit die Verstärkung 0 dB, ist(*Gleichung 4*). Das bedeutet für jeden Winkel α lässt sich aus zwei Achtsignalen ein neues gerichtetes Signal gewinnen.

Gleichung (4) $sa1^2 + sa2^2 = 1$

Umgekehrt kann dies durch die MS-Matrizierung bewiesen werden. Hierbei wird aus dem Mittensignal M (*siehe Abb.5 schwarzes Mikrofon*) und dem Seitensignal S (*siehe Abb.5 rosa Mikrofon*) eine jeweils linke Acht und rechte Acht. Man kann sich diesen Vorgang als Drehung der beiden Mikrofone um 45° im Uhrzeigersinn vorstellen. Entscheidend dabei ist die Tatsache, dass Mikrofone mit Acht-Charakteristik den Schall von vorne mit normaler Phasenlage und den Schall von hinten gegenphasig übertragen. (*siehe Abb. 4*) Dies erklärt das Minuszeichen in Gleichung (6).

Die allgemeingültige Formel lautet[14]:

Gleichung (5) $L = (M + S) * \frac{1}{\sqrt{2}}$

[14] Handbuch der Tonstudiotechnik, Michael Dickreiter Band I S. 291

Gleichung (6) $R = (M - S) * \frac{1}{\sqrt{2}}$

Für Beispiel gilt: $\alpha = 45°,\ M = sa1 = \cos 45° = \frac{1}{\sqrt{2}},\ S = sa2 = \sin 45° = \frac{1}{\sqrt{2}}$

$\cos^2 45° + \sin^2 45° = (1/\sqrt{2})^2 + (1/\sqrt{2})^2 = sa1^2 + sa2^2 = 1$

In dieser beschriebenen Arbeit werden also aus zwei Mikrofonsignalen neue Signale berechnet. Diese neuen Signale entsprechen Mikrofonen mit der Ausrichtung und dem Winkel α. Die dabei neu entstandenen Richtcharakteristiken besitzen wiederum die Form einer Acht. Für eine Berechnung einer XY-aufnahme wird zum Winkel α einfach 45° addiert. Die errechneten Faktoren im XY-Modus der fünf neuen Signale sehen wie folgt aus:

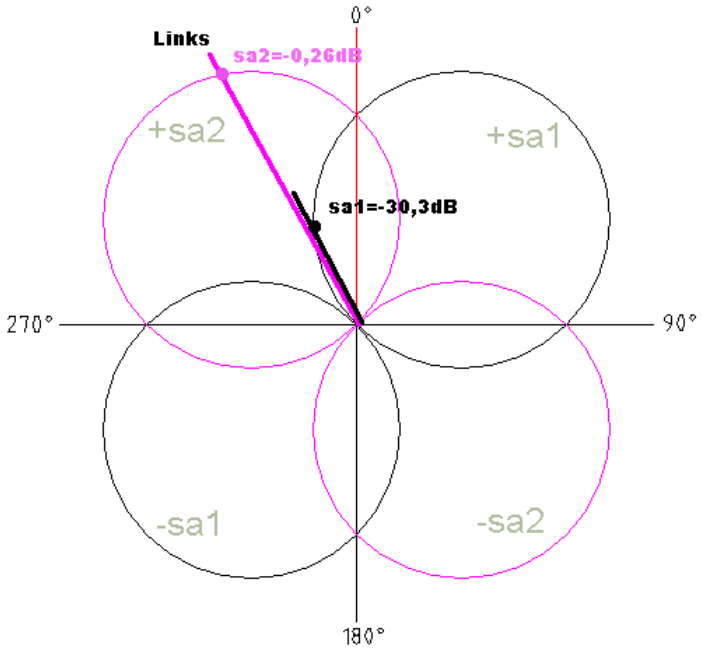

Abb. 6 Blümleinstereofonie XY im Vergleich zu MS um 45°gedreht mit Verstärkungsfaktoren für das Linkssignal mit Winkel -30°

Links:

$\sin^2(-30° + 45°) = \sin^2 5° = 0{,}030 \Rightarrow$ dies entspricht einem Pegel von *-30,3 dB*

$\cos^2(-30° + 45°) = \cos^2 5° = 0{,}97 \Rightarrow$ dies entspricht einem Pegel von *-0,26 dB*

Wie aus *Abb. 6* ersichtlich werden für das linke Signal die beiden phasenpostiven Anteile *(+sa1 ,+sa2)* der zwei Achten addiert. Für das neue Signal *sa* gilt daher:

$sa = 0{,}030 * sa1 + 0{,}97 * sa2$

HintenRechts:

$\sin^2(105° + 45°) = \sin^2 150° = 0{,}7449 \Rightarrow$ dies entspricht einem Pegel von -2,5 dB

$\cos^2(105° + 45°) = \cos^2 150° = 0{,}25 \Rightarrow$ dies entspricht einem Pegel von -12 dB

In diesem Fall wird von der rechten Acht das phasenpositive und von der linken Acht das phasennegative Signal aufaddiert:

$$sa = 0{,}25 * sa1 + 0{,}7499 * (-sa2)$$

Die Faktoren der restlichen drei Signale (Rechts, Center, Hintenlinks) ergeben sich durch das Einsetzen des Winkels α in die Gleichung 1,2 und 3. Die mathematische Herleitung geht davon aus, dass die Acht-Charakteristik aus zwei Kreisen besteht. Diese Annahme entspricht nicht der Realität. Die Kreise sind nicht rund, sondern eher eiförmig.

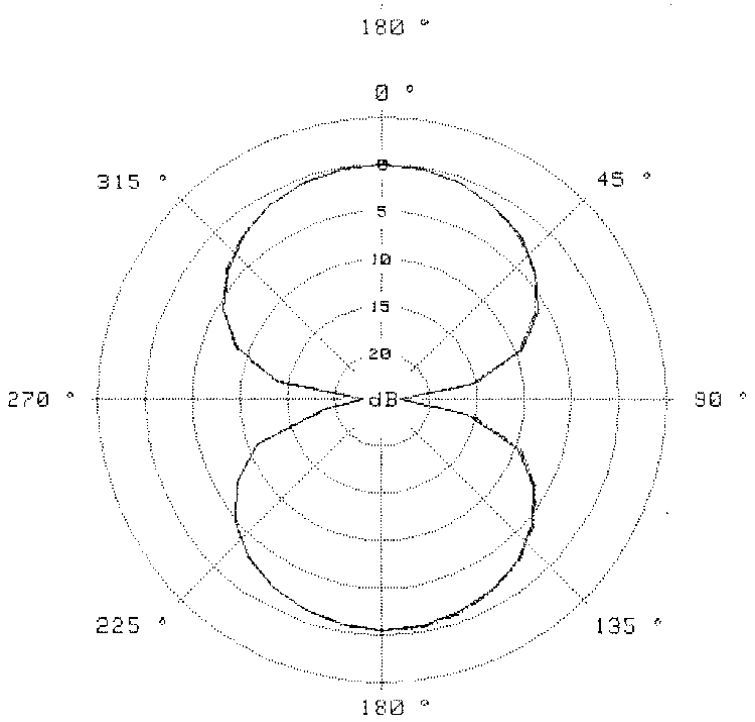

Abb7 Richtcharakteristik Neumann SM69 in Schalterstellung Acht (Messtechnik SWR Stuttgart)

Zur Überprüfung und Darstellung der mathematischen Herleitung, wurden zwei Achten im Messlabor des SWR Stuttgart, unter Leitung von Rainer Hofmann[15], getestet. Dazu wurde das Blumleinverfahren im XY-Modus aufgebaut. Beide Mikrofone wurden über ein Mischpult, mit den berechneten Faktoren, verstärkt und über die Mastersumme des Pults aufaddiert. Das Messsignal wurde dem Monoausgang des Mischpultes entnommen. Die fünf neuen Richtcharakteristiken Links, Rechts, Center, Hintenlinks sowie Hintenrechts wurden gemessen. Es sollte hiermit überprüft werden, ob aus einem Blumleinverfahren unendlich viele Achten mit dem Winkel α matriziert werden können. Eine These konnte belegt werden. Man kann unendlich viele neuen Achten aus diesem Verfahren gewinnen, nur der Winkel α

[15] Ingenieur der Nachrichtentechnik zuständig für Messtechnik und Messlabor beim SWR Stuttgart

war nicht ganz korrekt. Das Messdiagramm ergab einen anderen Winkel α für die berechneten Verstärkungsfaktoren, entsprach also nicht dem Winkel, welcher der Berechnung zu Grunde lag. Die gemessene Acht war bis zu 10° falsch gedreht. Für Links hätte die Hauptachse der Acht auf -30° liegen müssen. Das Ergebnis liegt bei -40° oder 320°.

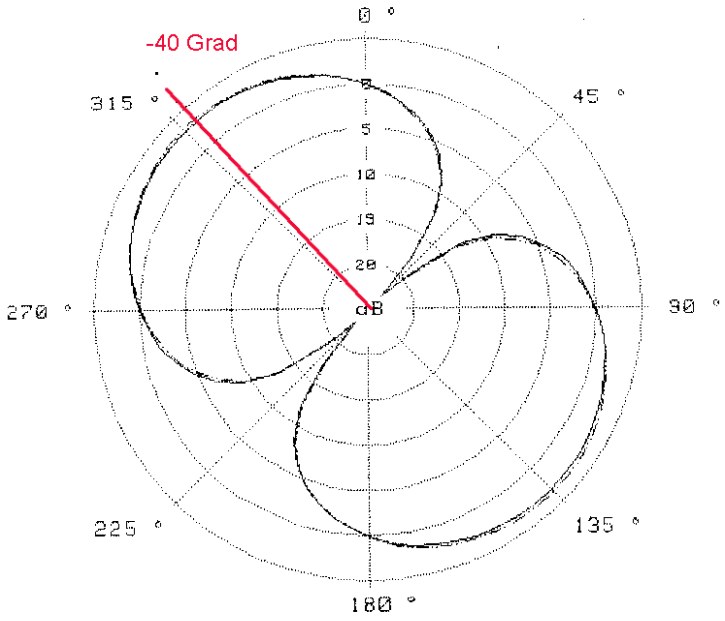

Abb.8 Richtcharakteristik aus XY-Verfahren matriziert für Links -30° (Messtechnik SWR Stuttgart)

Für Hintenlinks wurde am Mischpult die Faktoren für den Winkel α=-105° eingestellt. Auch hier hat sich ein anderer Winkel ergeben.

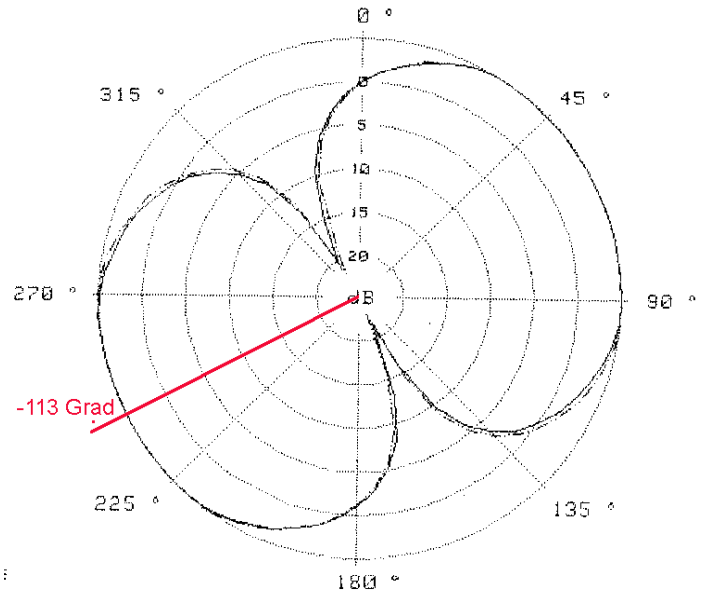

Abb. 9 Richtcharakteristik aus XY-Verfahren matriziert für Hintenlinks -105° (Messtechnik SWR Stuttgart)

Die Messtechnik lieferte andere Werte für den Winkel α, als die berechneten Faktoren hätten ergeben müssen. Diese Tatsache ist auf die Eiförmigkeit der Acht-Charakteristik zurück zu führen und wurde für diese Arbeit als nicht ausreichend betrachtet. Es musste eine andere Methode für die Gewinnung zuverlässiger Verstärkungsfaktoren gefunden werden.

2.3.2 Grafische Ermittlung der Verstärkungsfaktoren

Da Richtcharakteristiken, oder auch Polardiagramme, den Mikrofonpegel bezüglich der Einfallsrichtung des Schalls angeben. Kann einem Diagramm Pegelwerte für eine Richtung oder Winkel α entnommen werden. Betrachtet man ein Polardiagramm des Blumleinverfahrens im XY-Modus, sieht man, dass beide Mikrofone in der Summe eine Kugelcharakteristik ergeben. Wie im mathematischen Beleg, setzt sich ein Signal für jeden Winkel α aus jeweils zwei Komponenten der beiden Achten (*Abb. 5*) zusammen. Die jeweiligen Verstärkungsfaktoren werden schlicht dem Polardiagramm entnommen.

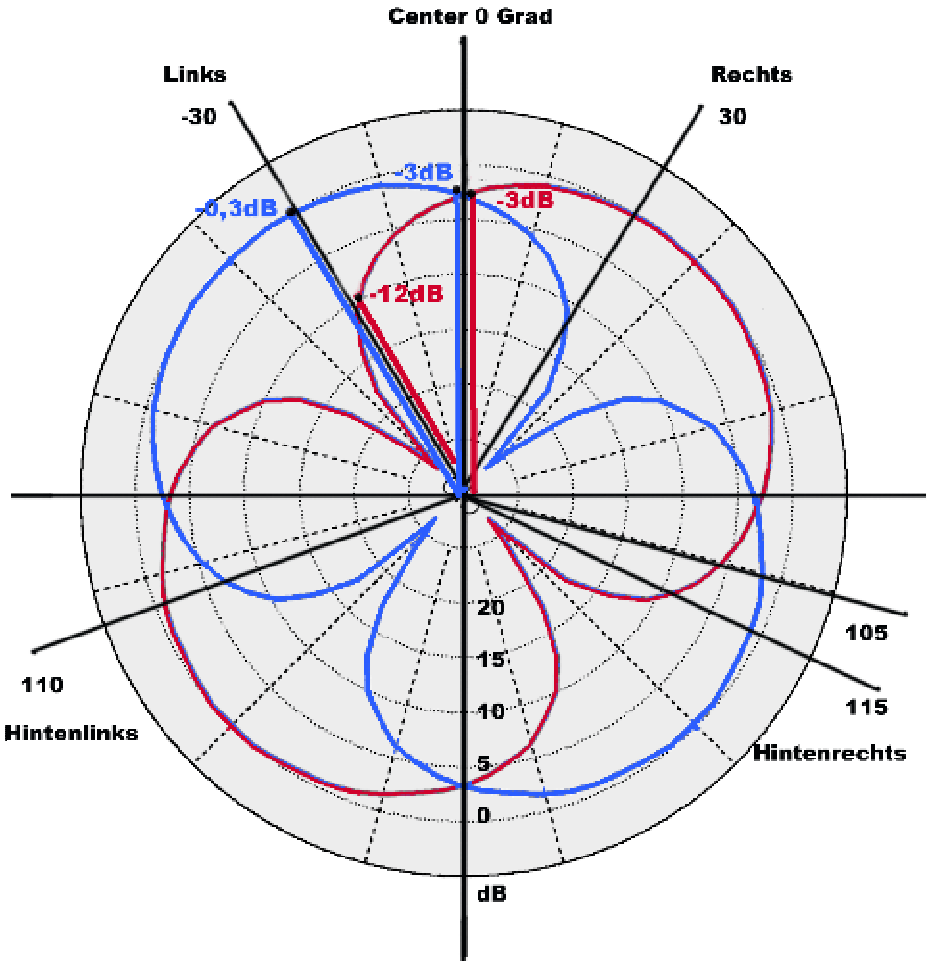

Abb. 10 XY-Blumlein-Verfahren im Polardiagramm mit den zugehörigen Winkeln für die ITU-775 Lautsprecherausrichtung

Auch die MS-Matrix lässt mittels *Abb.10* ablesen. Das Mittensignal, welches dem Centersignal entspricht, wird zu gleichen Teilen aus der linken und rechten Acht mit dem Pegel *-3 dB* gewonnnen. Der Pegel *-3 dB* entspricht dabei dem Verstärkungsfaktor $\frac{1}{\sqrt{2}}$. Die abgelesenen Pegelwerte für alle fünf Signale anteilig aus der linken Acht *(blau)* und rechten Acht *(rot)* ergeben folgende Werte:

Winkel	**linke Acht Phase +**	**rechte Acht Phase +**	**linke Acht Phase -**	**Rechte Acht Phase -**
Links	-0,3dB	-12dB		
Rechts	-12dB	-0,3 dB		
Center	-3dB	-3dB		
Hintenlinks				
105°	-6dB			-2dB
110°	-7dB			-1,2dB
115°	-8dB			-1dB
Hintenrechts				
105°		-6dB	-2dB	
110°		-7dB	-1,2dB	
115°		-8dB	-1dB	

Tabelle 1 Verstärkungsfaktoren der beiden Achten grafisch ermittelt anhand Abb.10

Sämtliche Verstärkungsfaktoren *A* können mit dem Pegel *D* und der Formel $A = 10^{\frac{D}{20}}$ berechnet werden. (*Tabelle 2*)

z.B. Verstärkungsfaktor Rechts mit der rechts Acht: $10^{\frac{-12dB}{20}} = 0{,}2511$

Winkel	linke Acht Phase +	rechte Acht Phase +	linke Acht Phase -	Rechte Acht Phase -
Links	0,9660	0,25		
Rechts	0,25	0,9660		
Center	0,7071	0,7071		
Hintenlinks				
105°	0,5			0,7943
110°	0,4667			0,8710
115°	0,3981			0,8912
Hintenrechts				
105°		0,5	0,7943	
110°		0,4667	0,8710	
115°		0,3981	0,8912	

Tabelle 2 Verstärkungsfaktoren der beiden Achten errechnet aus Tabelle 1

Testet man die die gleiche XY-Blumleinanordung mit den grafisch ermittelten Verstärkungsfaktoren, erhält man deutlich bessere Ergebnisse (*Abb. 11/12*). In der Diplomarbeit wurde für die Berechnung der Verstärkungsfaktoren, daher der grafischen Methode den Vorzug gewährt.

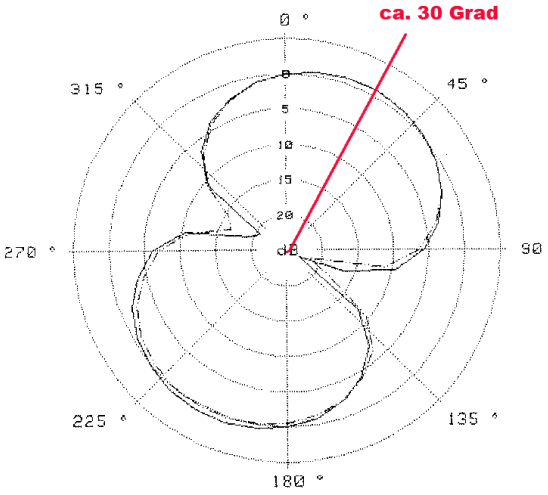

Abb.11 Messdiagramm Rechts anhand grafisch ermittelter Pegel

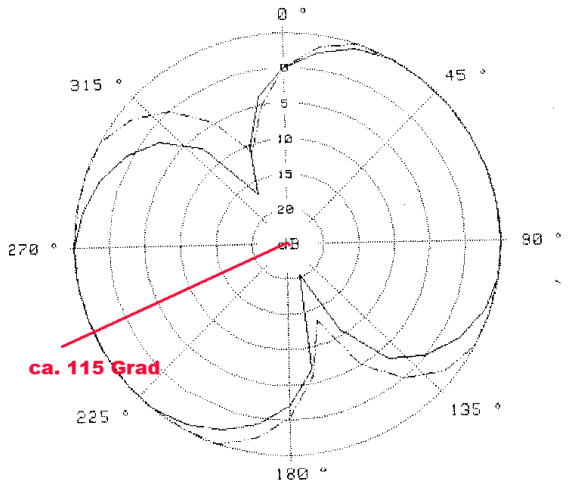

Abb.12 Messdiagramm Hintenlinks 115° anhand grafisch ermittelter Pegel

Alle Verstärkungsfaktoren der Software für die unterschiedlichen Winkel α können dem folgenden Javacode entnommen werden. Wobei der Winkel α der Variable GrafikInputs.wedge entspricht.

```
l[i]= (float) (0.9660*sampleLinks+(0.25*sampleRechts));
r[i]= (float) (0.9660*sampleRechts+(0.25*sampleLinks));
if(GrafikInputs.xy==true){
        c[i]= (float) ((sampleRechts+sampleLinks)*0.7071);
      }
if(GrafikInputs.wedge==105){
      hl[i]= (float) ((sampleLinks*0.5)-(sampleRechts*0.7943));
      hr[i]= (float) ((sampleRechts*0.5)-(sampleLinks*0.7943));

      }
if(GrafikInputs.wedge==110){
      hl[i]= (float) ((sampleLinks*0.4667)-(sampleRechts*0.8710));
      hr[i]= (float) ((sampleRechts*0.46675)-(sampleLinks*0.8710));
      }
if(GrafikInputs.wedge==115){
      hl[i]= (float) ((sampleLinks*0.3981)-(sampleRechts*0.8912));
      hr[i]= (float) ((sampleRechts*0.3981)-(sampleLinks*0.8912));
      }
```

Im Rahmen des Fortschreitens der Mehrkanaltechnik und dem Format 5.1 wird hier ebenfalls dieses Format als Grundlage verwendet. Wobei auf die Bearbeitung/Berechnung des LFE Kanal in der Diplomarbeit bewusst verzichtet wurde, da man dies in einem dementsprechenden Editor weitaus besser einstellen kann. Für die Diplomarbeit soll also ein 5.0 Standard gelten.

Die Software liefert, aus einer mit MS oder XY aufgenommenen Stereodatei, fünf neue Audiodateien. Diese fünf Dateien werden, durch die entsprechenden Winkel α, auf fünf Lautsprecher verteilt, die von der *ITU-775*[16] abgeleitet sind (*Abb.13*).

[16] siehe Kapitel 3.1

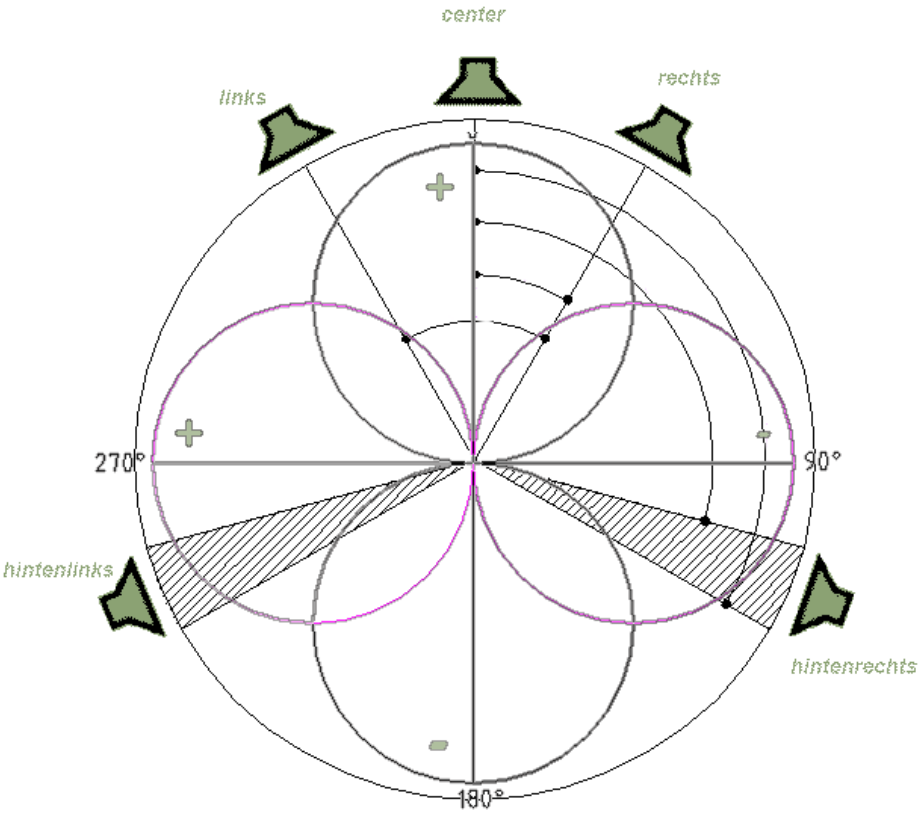

Abb.13 Blümleinanordnung MS mit Lautsprecheranordnung nach ITU-775

3 Normen, Empfehlungen

Surround, was im deutschen mit Raumklang übersetzt werden kann, ist aus den Kinos nicht mehr wegzudenken. Auch im Bereich Home-Entertainment ist dieser Prozess nicht zu bremsen. Für ca. 200 Euro können brauchbare Lautsprechersysteme im Fachhandel erworben werden. Teilweise werden Filme im Fernsehen oder Konzerte und Features im Radioim Mehrkanalformat übertragen. Unser Auge strebt nach HDTV[17] unser Ohr nach Raumklang. Normen und Empfehlungen sind daher im Bereich Surround äußerst wichtig, ermöglichen sie doch die gemeinsamen Plattformen, welche für die Produzierenden und Konsumierenden betreten werden können. Besonders einheitliche Wiedergabesituationen scheinen erstrebenswert, lassen sich aber konsumerseitig nur schwer durchsetzen. Der perfekt klingende Mix im Studio wird im Wohnzimmer stets anders klingen, allein schon notwendige bauakustische Maßnahmen übersteigen im Regelfall die private Portokasse. Daher sollen Normen und Empfehlungen, allen Beteiligten dienen und uns kompatible Formate liefern, zugleich aber nicht als starres Element einer Produktionskette gesehen werden, da dies jede Weiterentwicklung bremsen wird. Es wird hier die *ITU18-R- BS 775*[13] und die *EBU19 R 90-2000*[14] besprochen, die beide als Grundlage dieser Arbeit dienen sollen.

3.1 ITU-R-BS 775 (ITU 775)

Diese Empfehlung ist vor allem für Tonstudios, aber auch für den Heimbereich gedacht. Darin wird die Schaffung optimaler Hörbedingungen der Regieräume für Bild- und Tonwiedergabe beschrieben. Auf der Basis der *ITU 775* wurde eine Empfehlung unter dem Titel *„Hörbedingungen und Wiedergabeanordnungen für die Mehrkanal-Stereofonie„* des

[17] High Definition Television, hochauflösende Fernsehnorm

[18] Die Internationale Fernmeldeunion (Kürzel ITU; englisch International Telecommunication Union; mit Sitz in Genf ist eine Unterorganisation der Vereinten Nationen und die einzige Organisation, die sich offiziell und weltweit mit technischen Aspekten der Telekommunikation beschäftigt

[19] Die Europäische Rundfunkunion – englisch: European Broadcasting Union (EBU); ist ein Zusammenschluss von derzeit 75 Fernseh- und Rundfunkanstalten in 56 Ländern Europas, Nordafrikas und des Nahen Ostens. Sitz der EBU ist Genf.

Surround-Sound-Forum (SSF)[20] ins Deutsche verfasst. Bezüglich der Hörräume sollen folgende Richtlinien kurz genannt werden:

Grundfläche:40 m² (>25 m² im Heimbereich)

Raumvolumen:>300 m³

Nachhallzeit: 0,2-0,4 s (zwischen 200 Hz und 2,5 kHz)

FrüheReflexionen: -10 dB zum Direktschall

Nachhall: keine Flatterechos, keine Klangfärbungen

Grundgeräusch: 10 dB

Das wichtigste Element der *ITU 775* ist sicherlich die optimale Lautsprecherposition:

Höhe der Lautsprecher:1,2 m (0,9-1,4 m im Heimbereich)

Winkel LC bzw. CR: 30° (C bei 0°, daraus erschließt sich für LR=60°)

Winkel LSC bzw. CRS: 100-120°

Neigung der Lautsprecher: L/C/R=0°, LS/RS=15° nach unten

Abstand zu Reflexionsflächen: 1 m (>0,5 m im Heimbereich)

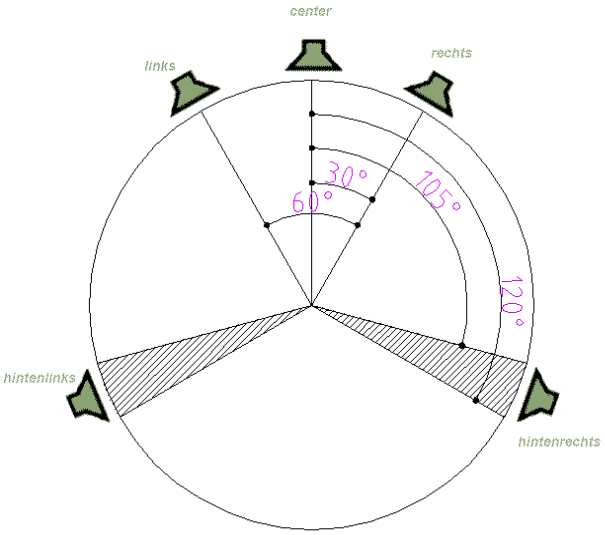

Abb.14 Lautsprecheraufstellung nach ITU 775

Dieser 3/2-Standard kann um einen optionalen LFE-Kanal („*Low Frequency Effect*“ bei *Dolby* oder „*Low Frequency Enhancement*“ bei *DTS*) erweitert werden. Welcher meist als Subbasskanal mit der oberen Grenzfrequenz bei *80 Hz* bzw. *120 Hz* benutzt wird. Die

[20] Das"SURROUND-SOUND-FORUM" ist eine interdisziplinäre und überregionale Arbeitsgemeinschaft, die1996 anlässlich der 19.Tonmeistertagung vom Verband DeutscherTonmeister(VDT) gegründetwurde.

Wiedergabe erfolgt über einen Subwoofer[21], dessen Position nicht festgelegt ist, da die Richtungswahrnehmung mit sinkender Frequenz abnimmt. Bereits ab *300 Hz* können keine Intensitätsdifferenzen mehr an den Ohren wahrgenommen werden. Hier wird die Richtung nur noch durch die unterschiedlichen Laufzeiten abgeleitet.[22] Für den Pegel des LFE-Kanals wird eine Anhebung um *10 dB* empfohlen, um dadurch die richtige Durchsetzungskraft zu bekommen. Da der LFE-Kanal nur optional ist gilt für die Produktion zu beachten, dass alle wichtige Inhalte auch ohne diesen Kanal übertragen werden können und daher nur anteilig auf diesem liegen. Um den Sweetspot zu vergrößern, kann man die Surroundkanäle dekorrelieren und diese Signale dann über mehrere Lautsprecher wiedergeben. Besonders in größeren Räumen, wie z.B. Kinosälen, kann dies sehr nützlich sein.

Um den Austausch und die Archivierung der Mehrkanalaufnahmen zu verbessern wurde von der *ITU*, die Verteilung der Kanäle auf ein 8 Spurmedium definiert:

Spur	Signal	Bemerkung
1	Links	
2	Rechts	
3	Center	
4	LFE	optionales Tiefbass- und Effektsignal (alternativ Kommentare oder monofones Surround-Signal MS)
5	LS(Links surround)	alternativ Monosurround MS (-3dB)
6	RS(Rechts Surround)	alternativ Monosurround MS (-3dB)
7	frei verfügbar	linkes Signal einer separaten Stereomischung

[21] Lautprecher, der für tieffrequente Signale ausgelegt ist

[22] vgl. Handbuch der Tonstudiotechnik , Michael Dickreiter, S.119

8	frei verfügbar	rechtes Signal einer separaten Stereomischung

Tabelle 3 Spurbelegung der Surroundkanäle auf ein Achtspurmedium[23]

3.2 Surround-Sound-Bewertung (EBU R 90-2000)

Diese Empfehlung der European Broadcasting Union beschreibt eine Terminologie für den Bereich Surround. Da das Hören und Klangempfinden meist subjektiv ist, sollen hier allgemeingültige Parameter definiert werden, die gut verständlich sind und leicht nachvollzogen werden können. Hier wird in Hauptparameter unterschieden und diese weiter in Unterparameter unterteilt.

Hauptparameter	Unterparameter	Allg.Beschreibung
1.Qualität des vorderen Klangbilds	Klangverteilung Klangstabilität Breite des Klangbildes Lokalisationsschärfe	weit/eng präzise/unpräzise stabil/unstabil lokalisierbar/nicht lokalisierbar
2.Seitliche und hintere Klangqualität	Klangverteilung Klangstablität Lokalisationsschärfe Homogentität des Raumklanges	stabil/unstabil lokalisierbar/nicht lokalisierbar
3.Raumeindruck	Nachhall Akustische Balance Erkennbare Raumgröße Räumliche Tiefe	Raum verhallt/trocken direkt/indirekt großer Raum/kleiner Raum

[23] Surround Einführung in Mehrkanal-Technik S.44

4.Transparenz	Definition der Klangquelle Zeitliche Bestimmung	Deutlichkeit klar/trüb
5.Verteilung der Klangquellen	vordere/hintere Lautheits-Verteilung direkte/indirekte Lautheitsverteilung Dynamikumfang	Klangquelle zu laut/zu leise
6.Klangfarbe	Timbre Klangfarbe Klangfarbe des Nachhalls	dröhnend/scharf neutral/dumpf/grell dunkel/hell
7.Frei von Rauschen und Störungen	Rauschen Verzerrungen Codierungsfehler	Bitfehler, Elektrische Störungen Akustische Störungen, Verzerrungen, Umgebungslärm Codierungsartefakte
8.Haupteindruck	Durchschnitt der subjektiven Gewichtung aller Parameter	

Tabelle 4 Haupt- und Unterparameter der EBU R 90-2000

4 Mehrkanalformate

Mehrkanalformate müssen natürlich genauso gespeichert und abgespielt werden, wie Mono- oder Stereoaufnahmen. Da Mehrkanalaufnahmen natürlich mehr Kapazität und Bandbreite bei der Übertragung benötigen, können sie verlustbehaftet oder bei ausreichender Kapazität und Bandbreite verlustfrei codiert werden.

4.1 Verlustbehaftete Mehrkanalcodierung

Da sich digitale Speichermedien nun seit Jahren zum Standard entwickelt haben, und die Kapazitäten immer größer werden aber dennoch nicht begrenzt sind, wird Audiomaterial, immer noch häufig, verlustbehaftet codiert, insbesondere bei der Kombination mit Videodaten. Inwieweit sich die einzelnen verlustbehafteten Codierungsformate unterscheiden soll hier kurz erläutert werden.

4.1.1 Dolby Surround und Dolby Surround ProLogic

Dolby Surround ist technisch gesehen ein analoges Matrixübertragungsformat. Ein Centerkanal sowie ein frequenzbeschränkter, rückwärtiger Monokanal werden in ein Stereosignal codiert. Bei der Codierung wird der Centerkanal mit *-3 dB* dem linken und rechten Signal beigemischt. Bei der Wiedergabe können dann die phasenkohärenten Signale wieder auf dem Centerlautsprecher wiedergegeben werden. Der Pegel des Surroundkanals wird ebenfalls um *3 dB* verringert, und durchläuft dann einen Bandpassfilter sowie ein Rauschminderungssystem. Dieses Signal wird jetzt mit $\pm$ *90°* phasengedreht und wiederum dem linken und rechten Kanälen aufaddiert (Abb. 15). Dieser Unterschied der Phasen wird beim Dekodieren erkannt und der Surroundkanal kann wieder gewonnen werden. Auf den Hauptkanälen löscht sich dieser durch die Gegenphasikeit aus. *Dolby Surround ProLogic* ist dabei eine Weiterentwicklung von *Dolby Surround* und verbessert die Kanaltrennung bei der Wiedergabe. Durch größere Kapazitäten der Speichermedien und größere Bandbreiten der Übertragung hat diese Codierung mittlerweile an Bedeutung verloren.

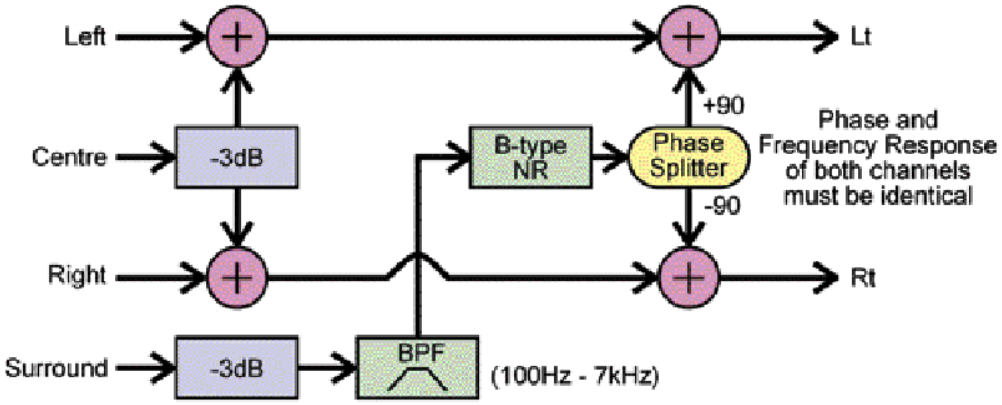

Abb. 15 Matrizierung Dolby Surround[24]

4.1.2 MPEG 2 Audio

MPEG 2 Audio ist das passende Audioformat für *MPEG2*, das Standardformat bei DVD-Video. *MPEG 2 Audio* ist eine verlustbehaftete Kompression und wurde 1992 eingeführt und verwendet insgesamt 818 Patente.[25] Bis zu acht Kanäle können hierbei diskret und datenreduziert übertragen werden. Wie *MPEG 1* ist auch *MPEG 2 Audio* in verschiedene Layer unterteilt, die jeweils abwärtskompatibel zu *MPEG 1* sind. Die Kompressionsrate liegt bei 12:1, was bei sechs Kanälen eine Datenrate mit *400 kB/s* bedeutet. *MPEG 2 AAC* ist eine Weiterentwicklung von *MPEG 2 Audio*, bei der die Bandbreite um weitere 50 % reduziert wurde. Beide sind der Internationale ISO-Standard für Mehrkanal-Audiocodierung und in Ländern mit *PAL*[26]-Norm bilden sie die Grundlage für DVD-Video.

4.1.3 AC3 (Dolby Digital)

Wieder ein Format, das von den *Dolby-Laboratories* entwickelt wurde. *AC3* oder auch *Dolby Digital* umfasst insgesamt sechs Kanäle, die unter 5.1 zusammengefasst werden. Die ersten fünf Kanäle übertragen das komplette Frequenzspektrum von *20 Hz – 20 kHz*. Der LFE überträgt nur Tieftoneffekte bis *120 Hz*. Dieses Format ist für Sampleraten bis zu *48 kHz* und einer Bittiefe bis zu *20 Bit* ausgelegt. Das mit einem speziellen Perceptual-Coding-Algorithmus abgetastete Signal, wird mit einem Kompressionsverhältnis von 10:1 datenreduziert. Diese Reduktion ist den Algorithmen von *MP3* und *Vorbis* ähnlich und ist

[24] http://www.soundonsound.com

[25] http://de.wikipedia.org/wiki/MPEG-2#MPEG-2_Audio

[26] Fernsehstandard in diversen Ländern, insb. Europa

somit auch verlustbehaftet. Die Datenrate bei *Dolby Digital* 2/0 (Stereo) beträgt durchschnittlich *192 kBit/s*, für *Dolby Digital* 5/1 *384-448 kBit/s*, jeweils bei *48 kHz* und *16 Bit*. Ein zusätzliches Feature von *Dolby Digital*, sind die im Bitstrom enthaltenen Metadaten (bezeichnet als BSI für Bit Stream Information). Zum einen kann hierdurch die Dynamik der Tonwiedergabe verändert werden, zum anderen werden Parameter und Daten für einen optionale Downmix von 5/1 zu 2/0 übertragen. Im Kino hat *Dolby Digital* das Lichtonverfahren *Dolby Stereo SR* abgelöst. Im Heimbereich konnte *Dolby Digital* das schlechtere Format *Dolby Surround* verdrängen. Auch im Rundfunk nimmt *Dolby Digital* eine immer zunehmende Rolle ein. Voraussetzung dafür ist der Empfang der Programme über *DVB* bzw. *DAB* (Digital Decoder)[27]. *AC3* oder *Dolby Digital* ist das meistverwandte Codierformat und gehört zum DVD-Standard.

4.1.4 DTS (Digital Theater System)

DTS wurde 1990 in Kalifornien entwickelt und erstmals in dem Film *„Jurassic Park“* verwendet. *DTS* kann sieben Kanäle mit einer Samplerate bis zu 96 kHz und einer Bittiefe bis zu *24 Bit* wiedergeben. Die Kompressionsraten variieren und liegen zwischen 3:1 und 6:1, was bei *48 kHz* und *16 Bit* einer Datenrate von *754 kBit/s-1,5 Mbit/s* entspricht. *DTS* wird als Konkurrenz zu *Dolby Digital* gesehen, wobei ein Vorteil des *DTS*-Formats die höhere Samplerate und höhere Bittiefe ist. Auch die Wiedergabequalität von *DTS* ist durch die geringere Kompression besser als bei *Dolby Digital*. Bei *96kHz* werden die Daten in einen Core- und einen Extension-Strom aufgeteilt, deren Verhältnis frei wählbar ist. Der Core-Strom enthält die *48 kHz* Informationen, der Extension-Strom die zusätzlichen *96 kHz* Information.[28] Das 6/1 Format ist durch eine spezielle Matrix abwärtskompatibel bis 1/0. Eine Neuheit des DTS-Formats ist ein zusätzlicher Effektkanal in dem Steuerungsdaten für Nebelmaschinen, Stroboskope etc. enthalten sein können.[29]

[27] Digitale Standards für digitales Fernsehen (DVB) und digitaler Hörfunk (DAB)

[28] Surround Einführung in Mehrkanal-Technik S.144

[29] http://de.wikipedia.org/wiki/DTS

4.2 Verlustfreie Codierformate

4.2.1 PCM (Puls Code Modulation)

Die *PCM* ist ein verlustfreies und unkomprimiertes Codierungsverfahren, in welchem analoge Signale in binäre umgewandelt werden. Es ist neben *MPEG 2 Audio* und *AC3* einer der Hauptstandards für DVD-Video, und erlaubt eine Aufzeichnung bzw. Wiedergabe von bis zu acht Kanälen in *24 Bit* und *96 kHz*. Bei einer Stereoaufnahme kann die Samplerate sogar bis *192 kHz* erhöht werden. Dem analogen Signal werden Proben entnommen. Den Proben wird eine binäre Zahl zugeordnet. Die Häufigkeit der Proben oder Abtastung pro Sekunde wird Samplerate genannt, die Auflösung oder Rasterbreite in Y-Richtung wird mit der Bittiefe definiert.

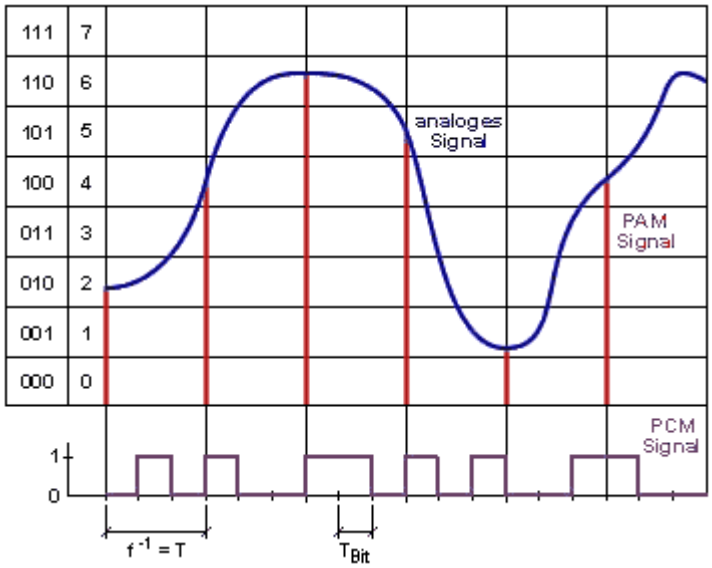

Abb. 16 Abtastung eines analogen Signals mit 3 Bit und Samplerate1/T

Bekannte Audioformate wie *wav* oder *aiff* gehören zu den *PCM* Verfahren. Auch beim *DSP* (Digital Signal Processing) wird dieses Verfahren genutzt. Jede abgetastete Binärzahl wird in eine Dezimalzahl umgewandelt und kann mit mathematischen Rechenoperationen manipuliert werden (z.B. Filteralgorithmen, Hallalgorithmen, Delay). In gleicher Weise funktioniert die *DSP* des Surrounder 2to5, hier wird, wie mittlerweile in Audioprogrammen üblich, mit *32 Bit* Fließkomma gearbeitet. Je höher die Auflösung des gewandelten Signal desto geringer sind die Fehler, die im neuen Signal durch Algorithmen entstehen. Auch wenn die CD mit 44*,1 kHz* und *16 Bit* wiedergegeben wird, sollte bei einer Aufnahme zumindest eine Bittiefe von *24 Bit* angestrebt werden. Dadurch wird das auftretende Quantisierungsrauschen minimiert. Durch jedes zusätzlich verwendetes Bit kann der Abstand zum

Quantisierungsrauschen halbiert, oder in Pegeln ausgedrückt, um *6 dB* verringert werden.[30] Die *PCM* ist durch ihre hohe Qualität ein bevorzugtes Format für hochwertige Aufnahmen.

4.2.2 MLP (Meridian Lossless Packing)

Meridian Audio entwickelte diese Mehrkanalcodierung und ließ diese Technik von *Dolby* vermarkten. Dieses Verfahren arbeitet mit einer verlustfreien Kompression, die eine Komprimierung von bis zu 2:1 erreicht. Die Datenrate richtet sich nach der jeweiligen Anwendung und liegt zwischen *1,4 und 9,2 MBit/s.* Es können hier acht Kanäle mit bis zu *192 kHz* und *24 Bit* codiert werden. Die Dekodierung des Signals benötigt relativ wenig Rechenleistung und lässt sich leicht implementieren, daher wird dieses Format selbst von einigen DVD-Kombiplayern unterstützt. Zusätzlich besteht eine Fehlerkorrektur die Fehler bis *2 ms* korrigieren kann und die Möglichkeit Metadaten in den Datenstrom einzubinden.[31] Die Wiedergabequalität steht der von der *PCM* um nichts nach und besticht durch die Halbierung des Speicherbedarfs einer solchen Aufnahme.[32]

4.2.3 DSD (Direct Stream Digital)

DSD wurde von Sony entwickelt und auf der *SACD* (Super Audio CD) verwendet. Das Audiosignal wird hier mit einer Samplerate von *2,8 MHz* und einer der Bittiefe 1 abgetastet. Dabei entsteht ein sehr hohes Quantisierungsrauschen, welches durch Noise Shaping[33] in nicht hörbare Frequenzbereiche verschoben wird. Der tatsächliche Dynamikumfang der *SACD*, beträgt *120 dB*, die höchste übertragbare Frequenz ist *100 kHz*. Trotz dieser guten Werte ist dieses Format nicht unumstritten, da es nur von wenigen Studiogeräten unterstützt wird und einfache Rechenoperationen, wie z.B.Crossfades sehr rechenintensiv.

[30] Handbuch der Tonstudiotechnik Band II S.281

[31] http://de.wikipedia.org/wiki/Meridian_Lossless_Packing

[32] Surround Einführung in Mehrkanal-Technik S.147

[33] Quantisierungsrauschen wird in nicht hörbaren Frequenzbereich verlagert

5 Mikrofonierung

Es gibt verschiedenste Mikrofonierungstechniken über die zahlreiche Arbeiten gefertigt wurden. Hier werden Stereomikrofonierungstechniken beschrieben werden, da diese die Grundlage der Mehrkanalaufnahmen bilden. Beide Verfahren dienen zum Verständnis dieser Arbeit.

5.1 Stereomikrofonierung

5.1.1 AB-Stereofonie

Dieses Zweikanal-Stereo-Verfahren beruht auf Laufzeitunterschieden zwischen den beiden Mikrofonen. Daraus folgend werden die Mikrofone im Abstand von *40-80 cm* oder mehr aufgestellt. Die Mikrofone werden frontal nach vorne ausgerichtet und der Abstand ergibt den Aufnahmebereich. Bei Laufzeiten von *1-1,5 ms* bilden sich Phantomschallquellen zwischen den Wiedergabelautsprecher aus. Ab ca. *2 ms*, was einer Wegdifferenz von *68 cm* entspricht, werden die Schallquellen getrennt wahrgenommen.[34] Für diese Technik spricht die ausgezeichnete räumliche Abbildung, hingegen leiden die Lokalisationsschärfe einzelner Schallquellen ebenso wie die Monokompatibilität.

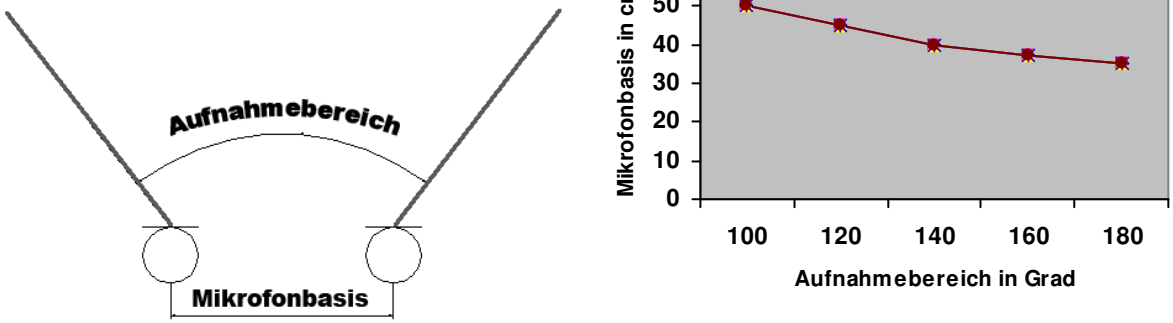

Abb.17 links: Aufnahembereich AB-Anordnung
rechts: Abhängigkeit Mikrofonbasis zum Aufnahmebereich

[34] Handbuch der Tonstudiotechnik, Johannes Webers S. 185

5.1.2 XY-Stereofonie

Diese Technik wird auch Koinzidenz-Verfahren genannt, bei dem sich die zwei Mikrofone auf der gleichen Achse befinden. Hier werden zwei gerichtete Mikrofone mit einem Öffnungswinkel α platziert, sodass hier der Öffnungswinkel und der daraus resultierende Versatzwinkel, den Aufnahmebereich definiert. An den Mikrofonen entstehen bei seitlichem Einfallswinkel Intensitätsunterschiede. Erst bei Schallpegeldifferenzen von mehr als *16 dB* identifiziert der Hörer die Schallquelle aus der Richtung des Lautsprechers. Diese Aufnahmen besitzen eine sehr gute Lokalisationsschärfe dafür eine schlechtere räumliche Tiefe.

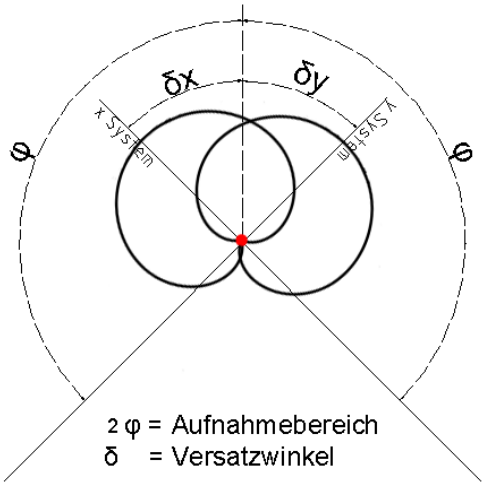

Abb.18 XY-Anordnung mit Aufnahmebereich und Versatzwinkel

5.1.3 Blumleinstereofonie

Der 1903 in London geborene *Alan Dower Blumlein* entwickelte dieses spezielle Aufnahmeverfahren, dass die Grundlage dieser Arbeit bildet. Das Prinzip ist eine Abänderung der XY-Stereofonie, welche ebenfalls zu den Intensitätsstereofonien gehört. Es werden zwei Mikrofone mit Achtcharakteristik orthogonal zueinander angeordnet. Die Achten nehmen Signale von vorne und von hinten in gleichen Teilen auf, nur dass die hinteren Signale phasengedreht sind (*Abb.19*). Da seitlicher Schall gleichphasig auf die Membran auftrifft, kann hier kein Druckunterschied und daher auch kein Signale entstehen. Ein Vorteil der Achten ist, dass die Richtcharakteristik relativ frequenzunabhängig ist und daher für Matrizierungstechniken vernachlässigt werden kann. Meist wirken jedoch Aufnahmen mit Achten in den Bässen etwas schwächer als Aufnahmen mit Druckempfängern.

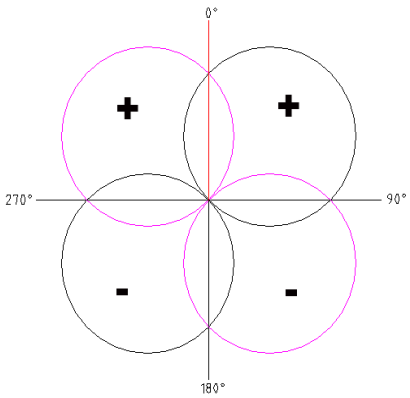

Abb.19 XY-Blumleinstereofonie mit Phasen + und -

5.2 Mehrkanalmikrofinierung

5.2.1 INA 3/INA 5 und ASM 5

INA steht für Ideale Nierenanordnung und wurde von *Ulf Herrmann* und *Volker Henkels* entwickelt. Beim *INA 3*-Verfahren werden drei Nieren verwendet um die Bereiche Links-Center sowie Rechts-Center separat aufzunehmen. Diese Bereiche dürfen sich nicht überschneiden, da es sonst zu Doppelabbildungen kommen kann, müssen sich aber an der C-Achse berühren. Der Aufnahmewinkel zwischen Links und Rechts muss null sein, damit L-R nicht als Stereobasis in Erscheinung tritt. Wird dieses Verfahren dann über ein *ITU 775* gerechtes Lautsprechersystem wiedergegeben, bilden sich Phantomschallquellen aus, die eine korrekte Lokalisation ermöglichen. Fügt man diesem System noch zwei Nieren für die Surroundkanäle hinzu, entsteht das *INA 5*-Verfahren. Diese Aufnahmetechnik ist ein gemischtes Verfahren, d.h. hier entstehen Laufzeitendifferenzen und Íntensitätsunterschiede. Die Praktische Umsetzung der *INA 5* findet in der *ASM 5* (Adjustable Surround Microphone) statt. Die Mikrofonabstände und Mikrofonwinkel sind frei einstellbar, teilweise können auch die Richtcharakteristiken per Fernsteuerung verändert werden.[35] Verschiedene Firmen bieten *ASM 5*- Systeme an, z.B. Brauner oder *Microtech Gefell*.

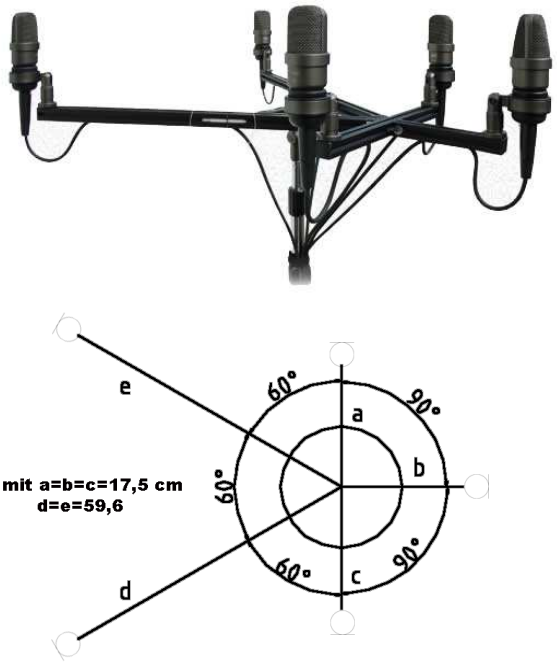

Abb.19 ASM 5 der Firma Microtech Gefell

[35] Surround, Einführung in Mehrkanalton-Technik ab S.79

5.2.2 IRT Kreuz

Dieses Verfahren geht auf *Günther Theile* zurück und ist ein Vierkanal-Aufnahmenverfahren. Hier wird jeweils in den Ecken eines Quadrats mit *20-25 cm* Kantenlänge ein Mikrofon platziert. Die Diagonalen des Rechteckes bilden die Mikrofonachsen, daraus folgt dass die Mikrofone jeweils um *90°* versetzt werden. Es werden hierbei Nierenmikrofone verwendet. Man kann dieses System auch als eine nach vorne und eine nach hinten gerichtete Stereoaufnahme betrachten.

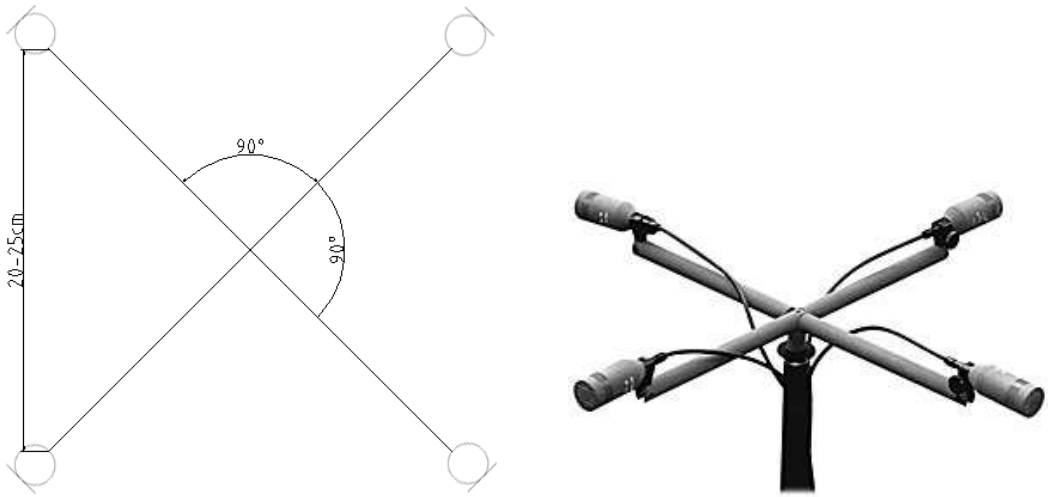

Abb. 20 IRT-Kreuz

5.2.3 Surround-Kugelflächenmikrofon

Hier wird ein Kugelflächenmikrofon verwendet das sich in der Praxis im Bereich Surround bereits mehrfach bewähren konnte. [36] Dieses Kugelflächenmikrofon wird um zwei seitlich angebrachte Mikrofone mit Achtcharakteristik erweitert. Diese Mikrofone werden jeweils nach vorne und nach hinten ausgerichtet, sodass hier zwei seitlich angeordnete MS-Aufnahmen entstehen. Dieses System kann Aufgrund des großen Aufnahmebereichs *120°* näher an eine Schallquelle positioniert werden. Durch Matrizierungsverfahren kann hier sogar eine Aufnahme im 5/1-Format erzeugt werden. Für die fünf Lautsprecher ergeben sich fünf virtuelle oder berechnete Mikrofone, deren Richtcharakteristik von Kugel über Niere zur Acht durchgestimmt werden können. Dieses Aufnahmesystem eignet sich für gerichtete Atmo-Aufnahmen, bei denen die Lokalisation erhalten bleiben soll. Besonders durch die

[36] Surround, Einführung in Mehrkanalton-Technik ab S.81

veränderbaren Richtcharakteristiken der virtuellen Mikrofone überzeugt dieses System durch gute Aufnahmequalität.

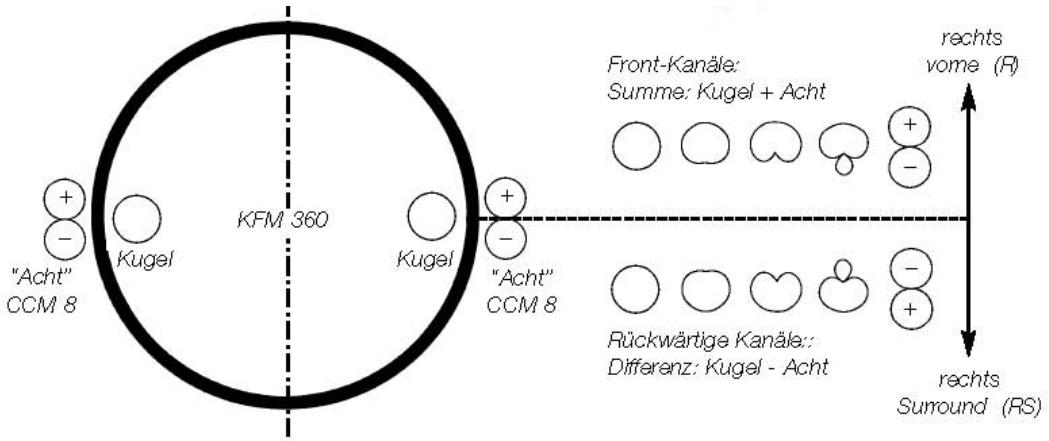

Abb.21 .Schoeps KFM 360 Kugelflächenmikrofon mit zwei Achten[37]

5.2.4 Doppelte-MS-Stereofonie

Bei diesem koinzidenten Verfahren wird eine MS-Technik um einen rückwärtigen Bereich erweitert. Es wird dabei ein zweites M-Mikrofon angebracht und nach hinten ausgerichtet. Das hintere MS-System liefert die Signale für die Surroundlautsprecher. Abhängig von den Charakteristiken der beiden M-Mikrofone kann dabei ein Aufnahmebereich von bis zu 360° entstehen. Daher kann die doppelte MS-Technik genau wie die das übliche MS-Verfahren nachbearbeitet werden. Das vordere M Signal kann für einen Frontkanal genommen werden es entsteht daher eine 5/0 kompatibel Anordnung.

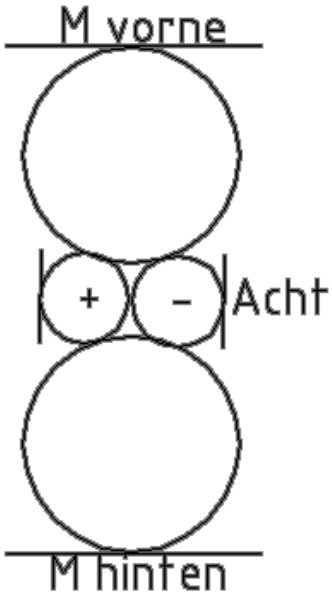

Abb.22 doppelte MS-Stereofonie

5.2.5 Soundfield

Dieses Prinzip wurde bereits im Kapitel 2.2 besprochen und soll der Richtigkeit halber an dieser Stelle mit eingereiht werden.

[37] http://www.schoeps.de/D-2004/kfm360.html

5.2.6 MST (Mitte-Seite-Tiefe)

MST ist dem doppelte MS-Verfahren sehr ähnlich, hierbei wird mit einer Blumleinstereofonie und einer Kugel gearbeitet, wie bereits in früheren Kapitel besprochen, können die Achten des Blumleinsystem durch die *Gerzon*-Matrizierung frei gedreht werden. Addiert man eine Acht mit einer Kugel entsteht eine Niere. Daraus folgend lassen sich für jeden Winkel α eine gerichtete Niere herstellen, die dann dem entsprechenden Lautsprecher zugeteilt werden kann.

5.2.7 Andere Mikrofonierungstechniken

Speziell in diesem sich noch entwickelndem Bereich gibt's es verschiedenste Rezepte und Techniken, die Tontechniker, Toningenieure und Tonmeister für sich entwickelt haben und hier nicht alle aufgelistet werden können. Hier sollen kurz die Namen der bekanntesten Verfahren genannt werden, *OCT* von *Günther Theile*, *Decca-Tree* und *Fukuda Tree*[38].

[38] Surround, Einführung in Mehrkanalton-Technik ab S.96-96

6 Aufbau und Funktionen des Surrounders

6.1 Kurze Programmbeschreibung

Dieses Programm berechnet aus zwei Kanälen ein 5/0 Signal, oder 3/2 Signal wie es in anderen Literaturen bezeichnet wird. Es werden dazu fünf virtuelle Mikrofone mit Acht-Charakteristik und Ausrichtungswinkel α errechnet. Jedem dieser Mikrofone wird eine Audiodatei zugeordnet. In diesem Fall jeweils eine für Links, Center, Rechts, Hintenlinks und zuletzt Hintenrechts. Die Aufnahme muss in der Blumleintechnik aufgenommen worden sein, entweder im XY- oder MS-Verfahren. Dieses Material wird im Programm geöffnet. Jetzt kann man mit dem Graphical User Interface[39] GUI die nötigen Einstellungen vornehmen. Die Einstellungsmöglichkeiten sollen hier nur kurz erläutert werden und zum allgemeinen Verständnis beitragen:

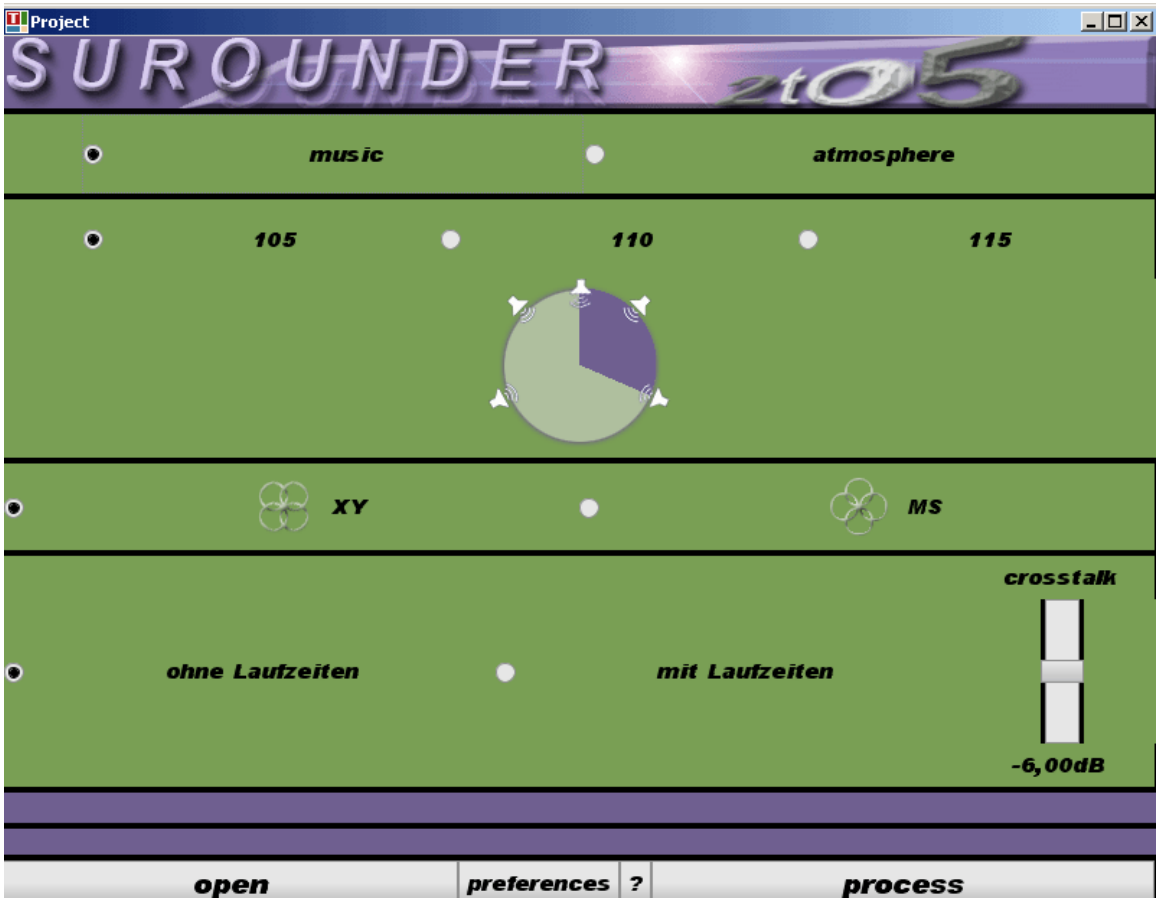

Abb.23 Screenshot Surrounder 2to5

[39] Bedienoberfläche eines Programmes

Music: Frequenzen für die Hintenlokalsiation (*1,2 kHz und 12 kHz*) werden abgesenkt, das Klangbild in der Front bleibt somit stabiler, ist vor allem bei Musik erwünscht
Atmosphere: hier werden keine Filter eingesetzt, entspricht eher einer Atmo-Aufnahme z.B für Film, Dokumentationen

105/110/115°: Die Norm *ITU-775* definiert die Winkel der hinteren Lautsprecher. Der User stellt hier den Winkel der hinteren Lautsprecher ein

XY/MS: Diese Einstellung definiert das Aufnahmeverfahren des Eingangmaterials, entweder XY oder MS

Laufzeiten mit/ohne: Im Laufzeitenmodus wird aus dem Blumleinverfahren, das nur mit Intensitäten und dadurch Pegeldifferenzen funktioniert, eine virtuelle *ASM 5*-Technik mit fünf Achten/Nieren generiert (*Kapitel 6.6 S.50*)

Open: öffnet eine Audiodatei wird dann im unteren Bereich der GUI angezeigt

Help: Kurzbeschreibung des Programms

Preferences: hier wird ein Speicherort auf einer Partition für temporäre Dateien anlegen

Process: Dateien werden in ausgewähltem Ordner gespeichert

6.2 Java

Java gilt als eine der objektorientierten Sprachen. Sie wurde von *Sun Microsystems* entwickelt und ist weiter auf dem Vormarsch. So kann die Sprache für Datenbanken, Internet mit *Java*-Applets und Servlets, Applikation für Handy und PDA[40] sowie Dokumenten- und Informationsaustausch auf *XML*-Basis verwendet werden und wie man hier sehen wird, in eingeschränktem Maße, auch für Multimedia-Anwendungen. Dazu muss der Entwickler zunächst eine Entwicklungsumgebung von Sun auf dem Rechner installiert haben. Der Surrounder 2to5 wurde mit der aktuellen Entwicklungsumgebung *JDK 1.6* entworfen. Das

[40] kleiner tragbarer Computer ohne Tastatur

Java Programm, auch der Surrounder, wird vom Programmierer normalerweise in verschiedenen Klassen unterteilt. Diese Klassen, mit ihrem Quelltext haben die Dateiendung *„.java“* und werden vom Java-Compiler in Java-Byte-Code umgewandelt. Die neuen Byte-Code-Dateien haben jetzt *„.class“*-Dateiendung. Dieser Byte-Code enthält noch keine maschinenspezifischen Anteile und kann daher noch nicht direkt ausgeführt werden sondern benötigt eine Java Virtual Machine *JVM*, welche auf dem Rechner installiert sein muss.

Die *JVM* führt dabei folgende Schritte durch:[41]

- Der Byte-Code wird geladen (Byte-Code-Loader), sucht und lädt auch ggf. weitere im Programm benötigte Klassen aus Bibliotheken (Klassenbibliotheken und *API´*s)

- Der Byte-Code wird auf Verstöße gegen die Java-Sprachregeln untersucht (Byte-Code-Verfizierung)

- Abschließend wird der Byte-Code interpretiert und als maschinenspezifischer Code auf dem System ausgeführt

Der Entwickler hat die Möglichkeit sich der jeweiligen *API* (Application Programming Interface) zu bedienen oder Open-Source-Packages von anderen Entwicklern zu benutzen. Die *GUI* dieses Programmes wurde mit dem Open-Source-Package *Thinlet*[42] entwickelt (*Kapitel 8*). Im Allgemeinen stellen diese Open-Source-Packages bereits viele nützliche Klassen und Methoden zur Verfügung, des weiteren lassen sich Packages leicht einbinden und nutzen.

Folgende Klassen wurden für die Diplomarbeit verwendet und werden in den folgenden Unterkapiteln erläutert:

6.2.1 Projekt

Projekt ist die Hauptklasse inklusive der main-Methode und ebenfalls eine Subklasse von *Thinlet*. Hier wird das Graphical User Interface *GUI* erzeugt und Teil gesteuert, Methoden die von der *GUI* gerufen werden, werden hier abgefangen. Sämtliche Elemente der GUI werden hier verwaltet. Der Pfad des temporären Ordners wird direkt in der main-Methode aus der

[41] Java 2 Grundlagen und Einführung RRZN, S. 19

[42] siehe Kapitel 8

Datei *hilfsDatei.bin* ausgelesen und wieder neu geschrieben. Falls noch kein Laufwerksbuchstabe gesetzt wurde, fängt der catch-Block die Exception auf und legt den Pfad mit „c:“ fest.

```
try {
            Processing.laufWerksBuchstabe=fu.getLaufwerksbuchstabe();
} catch (RuntimeException e) {
            e.printStackTrace();
            Processing.laufWerksBuchstabe="c";
      }}
hilfsFileII =new File(Processing.laufWerksBuchstabe+":\\hilfsDatei.bin");
```

6.2.2 Steuerung

Die Klasse Steuerung wird von der Klasse Thread abgeleitet. Die Klassenmethoden run() und ablauf() werden aus der Projectklasse, durch die starten()-Methode gerufen, welche ihrerseits durch das Drücken des Process-Button im Programm ausgelöst wird.

```
public void starten(){
            remove(dialogProgress);

            if(fu.getDiskSpace(openFile,hilfsFileII)){
               st=new Steuerung();
               st.ablauf();
               st.start();
            }
```

// der Button process ruft die Methode starten() in der Klasse Project

Die run()-Methode startet das *DSP* in der Processing Klasse, die ablauf()-Methode blendet die Progressbar ein. Es laufen somit zwei Prozesse nebeneinander ab.

```
public void run(){
          GrafikInputs gi=new GrafikInputs();
          FileUtility fu = new FileUtility();
          sampleRate= fu.getSampleRate();
          AudioFileFormat aff = fu.getAudioFileFormat(Project.openFile);
          AudioFormat af = fu.getAudioFormat(Project.openFile);
          bytesPerFrame=af.getFrameSize();
          processing.process(Project.openFile,12288);
      }
public void ablauf(){
            Project.p.progress();

      }
```

6.2.3 Processing

In dieser Klasse steckt der eigentliche Algorithmus, auch der komplette In- und Output befindet sich hier. Zuerst wird aus dem File f ein AudioInputStream erzeugt.

```
AudioInputStream aisI;
```

```
aisI=AudioSystem.getAudioInputStream(f);
```

Der AudioInputStream wird solange in einen Bytearray eingelesen bis die read()-Methode den Wert *„-1“* liefert. Der Bytearray hat die Länge bufferLength und ermöglicht ein schleifenweises Abarbeiten des AudioInputStreams. Dies erhöht die Schnelligkeit der Software und wird im Allgemeinen auch als Puffer bezeichnet.

```
byte [] buffer = new byte[bufferLength];
```

Der Bytearray buffer wird mit der convert()-Methode der Klasse Conversion in ein Fliesskomma-Array umgewandelt. Zur besseren Genauigkeit wird mit 32 Bit Fliesskomma gerechnet.

```
float[] fa=new float[];

fa = c.convertByteArray(buffer,fu.getAudioFormat(f),
                                              fu.getAudioFileFormat(f));
```

Die im Fliesskomma-Array fa enthaltenen Samplewerte des linken und rechten Kanal werden erneut in einer Schleife ausgelesen und der mixingXY()-Methode der Klasse Mixing übergeben. Dort werden aus den zwei Zahlen fünf Neue matriziert und in die fünf statischen Fliesskomma-Arrays gespeichert. Diese bilden die Grundlage der fünf Surroundkanäle, die am Ende des Algorithmus in die Dateien geschrieben werden.

```
m.MixingXY(d,f,j,schleifen,sampleLinks,sampleRechts);
```

Die fünf Fliesskomma-Arrays werden wiederum in fünf statische Byte-Arrays umgewandelt. Dies geschieht in der Klasse Reconversion mit der Methode FloatToByte().

```
l=rc.FloatToByte(linksResult,fu.getNewAudioFormat(f));
```

Um eine Arbeitspeicherproblematik zu vermeiden werden die Byte-Arrays in temporäre Dateien auf der Festplatte zwischen gespeichert.

```
tmpLinks = File.createTempFile("tmpLinks","tm",new
File(Project.tempPfad+":\\surounder_temp"));
```

Dafür werden zusätzlich ein FileOutputStream sowie ein FileInputStream benötigt. Der FileOutputStream füttert die temporäre Datei, der FileInputStream liest die Daten, nachdem der Hauptalgorithmus beendet ist, aus und gibt sie an den AudioInputstream weiter. Die Klasse AudioInputStream liefert eine write()-Methode, mit der die Audiodatei geschrieben wird.

```
FileOutputStream fopsLinks = null;
fopsLinks = new FileOutputStream(tmpLinks);
fopsLinks.write(l); // ByteArray wird in FileOutputStream gelesen
FileInputStream  fipsLinks =null;
```

```
fipsLinks = new FileInputStream(tmpLinks); //temporäre Daten werden
´                                          //ausgelesen
AudioInputStream loutlaisLinks = new AudioInputStream(fipsLinks,
fu.getNewAudioFormat(f),aisI.getFrameLength());
Project.outputLinks=fu.getTargetFile(Project.savePfad+"_Links."+fu.getFileT
ype(f)); //Enddatei mit gewünschtem Pfad wird erzeugt
int   lnWrittenBytesLinks  = AudioSystem.write
(loutlaisLinks,fu.getFileTyp(f),Project.outputLinks);//Datei wird geschrieb
```

Am Schluss müssen nur noch die temporären Dateien gelöscht werden.

```
tmpLinks.delete();
```

6.2.4 Conversion

Um die von der Audiodatei gelieferten Bytes in Samples zu packen wurde diese Klasse geschrieben. Hier wird ebenfalls das Format des Audioinputs abgeklärt und auf die interne Fliesskommaberechnung formatiert. Die Methode convertByteArray() liefert als Rückgabewert den benötigten Fliesskomma-Array. Mit einer switch/case-Logik wird die Bitbreite des Audiomaterials unterschieden. Durch das Schieben mit Bitoperatoren werden die Bytes in Float umgewandelt. Dabei werden zusätzlich die Attribute signed/unsigned sowie bigendian/littleendian überprüft und berücksichtigt.

```
//16bit Fall
case 16 : {
                  float [] fa = new float[ba.length/2];
                  //System.out.println("16bit-case");
                  // zuerst bigEndian
                  if(af.isBigEndian()){
                        // abfrage signed
                        if (ec == AudioFormat.Encoding.PCM_SIGNED){
                            for(int i=0;i<ba.length-1;i+=2){
                                 fa[j] = (  (ba[i + 0] << 8) |
                                            (ba[i + 1] & 0xFF) )
                                    / 32768.0F;
                              j++;
                             }
                           return fa;
                    }
                        // abfrage unsigned
                        else {
                              for(int i=0;i<ba.length-1;i+=2){
                                fa[j] = (((ba[i + 0] << 8) |
                                          (ba[i + 1] & 0xFF)) - 32768)
                                           / 32768.0F;
                                 j++;
                             }
                              return fa;
                        }
                  }
                  // jetzt littleEndian
                  else  if(!af.isBigEndian()){
                        // zuerst signed
                          if (ec == AudioFormat.Encoding.PCM_SIGNED){
                                  for(int i=0; i<ba.length-1;i+=2){
```

```
                                fa[j] =  (  (ba[i + 0] & 0xFF) |
                                           (ba[i + 1] << 8) )
                                           / 32768.0F;
                                           j++;
                                }
                            return fa;
                        }
                      // jetzt unsigned
                        else{
                          for(int i=0; i<ba.length-1;i+=2){
                                fa[j] =(( (ba[i + 0] & 0xFF) |
                                          (ba[i + 1] << 8) ) - 32768)
                                           / 32768.0F;
                                j++;
                                }
                           return fa;
                         }
                    }
                   // falls weder bigNedian oder littleEndien
                   else{
                     System.out.println("weder little noch bigEndian");
                     fa = null;
                     p.exception("Format wird nicht unterstützt");
                     return fa;
                   }
          }
```

6.2.5 Reconversion

Der Fliesskomma-Array wird wieder in einen Byte-Array umgewandelt.

```
//16bit Fall
case 16 : {

         byte[] buffer = new byte[fa.length*2];
                 if(naf.isBigEndian()){
                 for (int i=0;i<fa.length;i++){
                       float  fSample =  fa[i];
                       fSample = Math.min(1.0F, Math.max(-1.0F, fSample));
                       int sample = Math.round(fSample * 32767.0F);
                       byte low = (byte) ( (sample >> 8) & 0xFF);
                       byte high = (byte) (sample & 0xFF);
                       // high byte
                       buffer[j+0] = low;
                        // low byte
                       buffer[j+1] = high;
                       j+=2;
                       }
                       return buffer;
                 }
                 // jetzt littleEndian
                 else  if(!naf.isBigEndian()){
                 for (int i=0;i<fa.length;i++){
                       float fSample =fa[i];
                       fSample = Math.min(1.0F, Math.max(-1.0F, fSample));
                        //    scaling and conversion to integer
                       int nSample = Math.round(fSample * 32767.0F);
                       byte high = (byte) ( (nSample >> 8) & 0xFF);
                       byte low = (byte) (nSample & 0xFF);
                       // high byte
                       buffer[j + 0] = low;
                       // low byte
```

```
                    buffer[j + 1] = high;
                    j+=2;
                    }
                    return buffer;
```

6.2.6 DateiAuswahl

FileOpener und FileSafer aus dem Swing Container werden erzeugt. Für das Anzeigen spezieller Dateiformate können im FileOpener Filter gesetzt werden.

```
return f.isDirectory() ||
          f.getName().toLowerCase().endsWith( ".wav" )||
          f.getName().toLowerCase().endsWith( ".aif" )||
`         f.getName().toLowerCase().endsWith( ".aiff" );
```

Das Erzeugen eines Fileopener oder FileSafer geschieht durch folgenden Code. Die getSelectedFile()- Methode liefert die ausgewählte Datei. Hingegen die setSelectedFile legt den Speicherort fest.

```
public void Dateispeichern(){
        JFileChooser fc= new JFileChooser();
        fc.showSaveDialog(null);
        fc.setDialogType(JFileChooser.SAVE_DIALOG);
        fc.setSelectedFile(Project.saveFile);
        }
```

6.2.7 Delay

Die Delay-Klasse ist prinzipiell nur eine Hilfkslasse die aus der Processing-Klasse ausgelagert wurde. Das Hauptsignal sowie die vier verzögerten (mit Laufzeiten) Signale werden aufaddiert.

```
public class Delay {
      float setDelayArray(float a,float b,float c, float d,float e){
            return a+b+c+d+e;
      }
}
```

6.2.8 FileUtility

Diese Klasse ist ebenfalls eine Hilfsklasse und liefert nützliche Methoden die an verschiedensten Stellen der Software benötigt werden. Z.B. die Methode getSampleRate() liefert die Samplefrequenz als Fiesskommazahl.

```
public float getSampleRate(){
            AudioFormat af=getAudioFormat(Project.openFile);
            float SR= af.getFrameRate();
            return SR;

      }
```

6.2.9 GrafikInputs

Die Voreinstellungen der *GUI* werden hier gespeichert sowie deren Veränderungen während der Programmlaufzeit. Die Voreinstellungen des Programms werden als Klassenvariablen gesetzt.

```
public class GrafikInputs {
      static boolean uebersprechen=false;
      static boolean musicalMaterial=true;
      static int wedge=105;
      static boolean xy=true;
      static boolean laufzeiten=false;
```

Für jede Variable gibt es eine setter/getter Methode.

```
boolean getUebersprechen(){
            return uebersprechen;
      }
double setUebersprechen(int n){
            double m=(100-n)/100.0;
            uebersprechenDB=m;
            return uebersprechenDB;
```

6.2.10 IIR

Eine Klasse für rekursive Filter mit unendlicher Impulsantwort (*siehe Kapitel 7.2*). Die Filterfaktoren `ABandStopHigh` und `BBandStopHigh` werden in Arrays gespeichert. Die Filterberechnung Berechnung erfolgt über die Instanzenmethode `float iirBandStopHigh()`, die das gefilterte Sample zurückgibt.

```
ABandStopHigh=new double[]{
                              0.84420591335262296000,
                           -2.40837891972029050000,
                           3.40608784899173810000,
                           -2.40837891972029050000,
                           0.84420591335262296000
                        };
BBandStopHigh=new double[]{
                              1.00000000000000000000,
                           -2.59108115924437280000,
                           3.32359208506061910000,
                           -2.14974627917628960000,
                           0.69059892321034932000
                        };
float iirBandStopHigh(float NewSample) {
          //output samples
          int n;
          filterLaenge= ABandStopHigh.length;
          //shift the old samples
          for(n=BBandStopHigh.length-1; n>0; n--) {
             u[n] = u[n-1];
             r[n] = r[n-1];
          }
```

```
            //Calculate the new output
            u[0] = NewSample;
            r[0] = (float) (ABandStopHigh[0] * u[0]);
            for(n=1; n<=BBandStopHigh.length-1; n++)
                r[0] += ABandStopHigh[n] * u[n] - BBandStopHigh[n] * r[n];

            return r[0];
        }
```

6.2.11 Mixing

In dieser Klasse werden die Eingangs-Fliesskommazahlen entsprechend der Matrix in Ausgangs-Fliesskommazahlen umgewandelt. Außerdem ist hier eine MS- Matrix enthalten. Auch der eingestellte Winkel für die hinteren Lautsprecher findet hier seine Berücksichtigung. In diesem Beispiel wird der Hintenlinks und Hintenrechts-Array in einer Schleife bis i= lenght-1 berechnet

```
if(GrafikInputs.wedge==105){
              hl[i]=   (float)  ((sampleLinks*0.5128)-(sampleRechts*0.7943));
              hr[i]=   (float)   ((sampleRechts*0.5128)-(sampleLinks*0.7943));
```

6.3 Input/Output und die Organisation einer Audiodatei

Als Quellmaterial oder Input der Software kann nur eine Stereodatei benutzt werden. Java unterstützt dabei sowohl *WAV*-Dateien als auch *AIFF*-Dateien. Eine Bittiefe von *8* bis *24 Bit* und Samplingfreuqenzen bis *96 kHz* können mit dem Surrounder berechnet werden. Ausgegeben werden fünf Monodateien, welche das gleiche Format wie das Quellmaterial besitzen. D.h. es können keine Formatkonvertierungen durchgeführt werden.

6.3.1 Pulscodemodulation PCM

Die Pulscodemodulation beschreibt ein Verfahren bei dem ein analoges Signal in ein binäres und somit digitales Signal umgewandelt wird. Unter Berücksichtung des Shannon´schem Theorem wird das analoge Signal dabei in zeitlich gleichen Abständen mit einer Samplingfrequenz f abgetastet. Die Abtastwerte werden nun binär gespeichert. Die dabei entstehenden Rundungsfehler werden als Quantsierungsrauschen bezeichnet, da sie auch als Rauschen wahrgenommen werden.

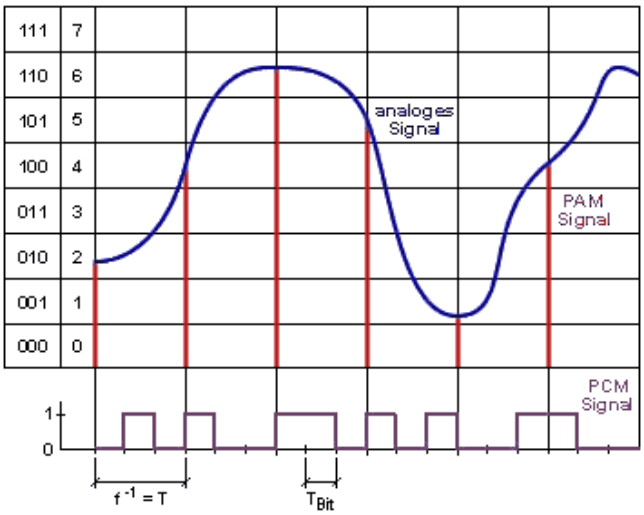

Abb24. Pulscodemodulation[43]

6.3.2 WAV/AIFF und die Attribute der Audiodateien

Wichtige Eigenschaften oder Attribute der Audiodateien werden im *Header* gespeichert und sind dadurch für den Programmierer, oder noch wichtiger, für das Programm zugänglich. Der *Header* ist ein Teil der Audiodatei und wird dabei wieder in so genannten *Chunks* untergliedert. In jedem *Chunk* sind verschiedenste Attribute oder Eigenschaften codiert:

-Samplefrequenz

-Samplegröße

-Anzahl der Kanäle

-Anzahl der Sampleframes

-Audioformat (WAV/AIFF)

-Dateigröße

-signed/unsigned

All diese Attribute können mit den Javaklassen AudioFormat und AudioFileFormat ausgelesen werden. z.B. Die Methode `public float getSampleRate()` liefert die Samplerate als Flieskommazahl.

```
public float getSampleRate(){
            AudioFormat af=getAudioFormat(Project.openFile);
            float SR= af.getFrameRate();
            return SR;
```

[43] http://de.wikipedia.org/wiki/Bild:Pcm.png

6.3.3 Samplefrequenz/Samplerate

Die Samplefrequenz oder Samplerate steckt in der Variable `Steuerung.sampleRate` wird vor allem für die Filter benötigt. Die Filterfaktoren sind von der Samplerate abhängig und sehen im Programmcode folgendermaßen aus:

```
if(Steuerung.sampleRate==44100){
            ACoef=new double[]{
                        0.01340799061903569600,
                         0.0000000000000000000,
                         -0.02681598123807139200,
                         0.0000000000000000000,
                         0.01340799061903569600
          };
            BCoef=new double[]{
                        1.0000000000000000000,
                         -3.64377973881270110000,
                         5.00893962214101410000,
                         -3.08216307959984490000,
                         0.71717534951999551000
               };
```

6.3.4 Samplegröße

Diese Information liefert die Samplegröße in Bits. Das Programm kann Dateien bis *24 Bit* verarbeiten. Höhere Bitanzahlen werden von *Java* derzeit noch nicht unterstützt.

6.3.5 little-endian/big-endian

Es gibt verschiedene Möglichkeiten die Bits in einer Datei zu organisieren. Häufig werden das *WAV*- oder das *AIFF*-Format verwendet, beide gehen auf das Prinzip der Pulscodemodulation zurück und gehören daher zu den linearen Verfahren. Da die abgetasteten Werte meist aus mehr wie *8 Bit* bestehen müssen sie in mehrere Bytes aufgeteilt werden. Eine Audiofile, welches mit *16 Bit* abgetastet wurde benötigt daher zwei Gruppen in Bytegröße. Die erste Gruppe von Bit 0-7, die zweite Gruppe von Bit 8-15. Ob der abgetastete Wert jetzt zuerst in die erste und dann in zweite Gruppe wird oder genau umgekehrt gespeichert, wird entweder als little-endian (*WAV*-Datei) oder big-endian (*AIFF*-Datei) bezeichnet.

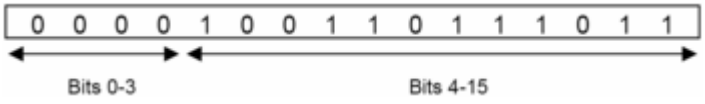

Abb.25 Beispiel der Bitanordnung eines 12 Bitsamples little-endian[44]

[44] http://de.wikipedia.org/wiki/Wav

6.3.6 signed/unsigned

Als weiteres Merkmal wird bei Audiodateien der Zustand signed/unsigned unterschieden. Dieses Attribut muss vom Programm überprüft werden und wird bei der Umwandlung der Bytes in eine Fliesskommazahl berücksichtigt.

```
if(ec == AudioFormat.Encoding.PCM_SIGNED){
```

Signed beschreibt, dass bei einer 8 Bit Zahl die Werte von 0 bis 255 gespeichert werden, bei unsigned werden die Werte von -128 bis 127 gespeichert. D.h. bei signed wird mit 0 begonnen, bei unsigned bildet die 0 die Mitte.

Samplegrösse	Attribut	Kleinster Wert	Min. Wert	Max. Wert
8 Bit	unsigned	0	128	255
	signed	-128	0	127
n-Bit	unsigned	0	2^{n-1}	$2^{n}-1$
	signed	$-(2^{n-1})$	0	$2^{n-1}-1$

Tabelle 5 signed/unsigned einer 8/n- Bitzahl

6.4 Der AudioInputstream

Die Klasse Audiosystem bietet viele Unterklassen sowie Methoden für das Medium Audio. Eine davon ist der AudioInputStream, dieser wird mit der geöffneten Audiodatei gefüttert und dann solange in den Bytearray „buffer“ ausgelesen, bis die read-Methode die Zahl -1 liefert und die komplette Datei ausgelesen wurde.

```
AudioInputStream aisI;
aisI=AudioSystem.getAudioInputStream(f);
bytesread = aisI.read(buffer);
```

Auch beim Speichern des Audiomaterials wird der AudioInputStream benötigt. Die berechneten und veränderten Werte werden in einen FileInputStream gepackt. Dieser wiederum wird als Objekt im Konstruktor des AudioInpurtStream benötigt. Genauso wie die Attribute des Audioquellmaterials.

```
AudioInputStream loutlaisLinks = new AudioInputStream
(fipsLinks,  fu.getNewAudioFormat(f),aisI.getFrameLength());
```

Dieser AudioInputstream wird über die write-Methode der Klasse Audiosystem in ein File geschrieben. Als Argument benötigt diese Methode zusätzlich noch den Filetyp sowie die Datei, in welche geschrieben werden soll.

```
int   lnWrittenBytesLinks  = AudioSystem.write
(loutlaisLinks,fu.getFileTyp(f),Project.outputLinks);
```

6.4.1 Arbeitsspeicherverwaltung durch temporäres Verzeichnis

Da nicht nur kleine Audiodateien bearbeitet werden sollen, sondern auch längeres Material, werden die Bytes nicht im Arbeitspeicher abgelegt sondern direkt in fünf temporäre Dateien geschrieben. Bereits eine Monodatei in 60-minütiger Länge sowie *24 Bit* und *44,1 kHz* benötigt circa *476 MB*. Da dies für alle 5 Kanäle der Surroundanordnung gilt, entsteht hier bereits ein Speicherbedarf von *2,4 GB*. Diese Kapazität an Daten lagert man sinnvollerweise natürlich nicht im Arbeitspeicher ab. Der Anwender der Software Surrounder hat die Möglichkeit die Daten auf einer Partition zwischenzuspeichern. Auf dieser Partition wird automatisch der Ordner „*surounder_temp*“ angelegt in dem die gespeicherten Dateien nach der Berechnung wieder gelöscht werden. Der Laufwerksbuchstabe wird ebenso in einer Datei auf der Partition gespeichert und bei jedem Neustart von dort ausgelesen, sodass das temporäre Verzeichnis erhalten bleiben kann.

```
tempFileAlt=new File(fu.getLaufwerksbuchstabe()+":\\surounder_temp");
tempFileAlt.mkdir();
tempPfad=fu.getLaufwerksbuchstabe();
```

6.4.2 Der Programmpunkt Laufzeiten und das Übersprechen

Da der Sweet-Spot bei der Matrizierung mit Intenstitätsdifferenzen relativ klein ausfällt, soll dieser durch simuliertes Übersprechen der jeweils anderen vier Kanäle vergrößert werden. Eine Dekorrelation der Signale wird auch im Kino zur Vergrößerung des Sweet-Spots benutzt. Der Modus Laufzeiten leitet sich von der *ASM 5*-Aufnahmetechnik ab und soll das Übersprechen der einzelnen Mikrofone simulieren. In dieser Aufnahmetechnik sind die Abstände und die Winkel der Mikrofone zueinander wie folgt definiert.

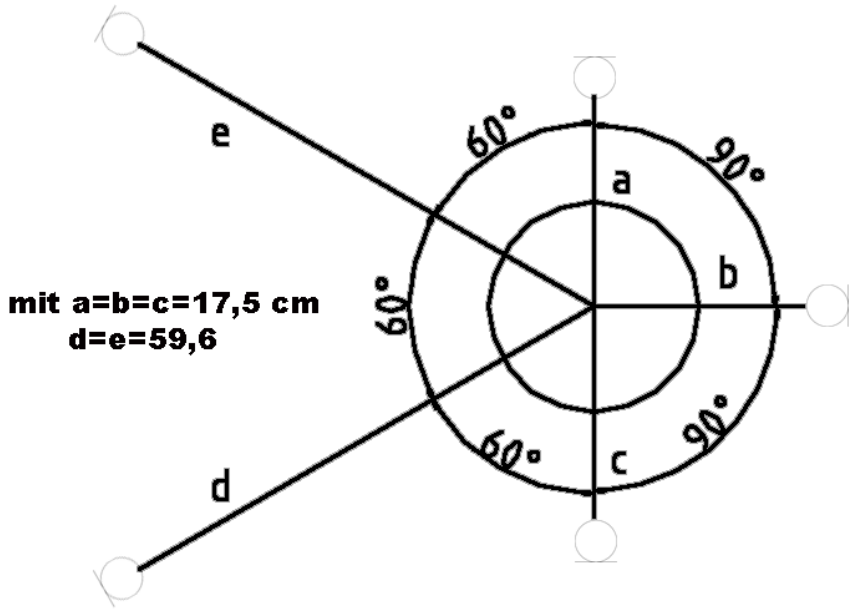

Abb.26 ASM 5-Anordnung gemäß ITU-775

Bei der *ASM 5*-Technik entstehen die Richtungsinformationen durch die Laufzeiten der Mikrofone zueinander und den Intensitätsdifferenzen. Geht man von Mikrofonen mit nierenförmigem Polardiagramm aus, wird der Schall aus unterschiedlichen Richtungen unterschiedlich an der Kapsel wahrgenommen. Dieser Unterschied ist zusätzlich frequenzabhängig und wird im Polardiagramm deutlich.

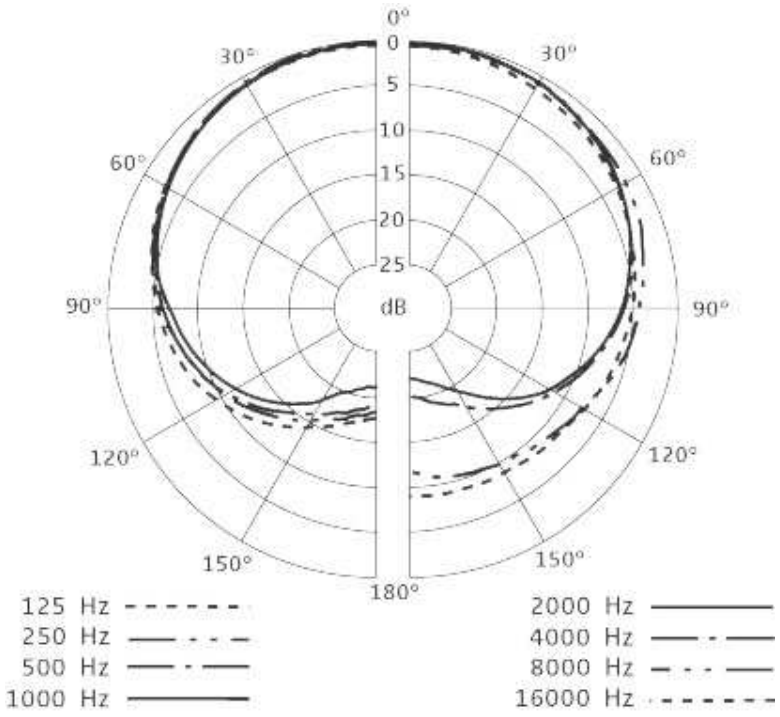

***Abb. 27 Polardiagramm einer Niere*[45]**

[45] http://www.sennheiser.com/

Ein Mikrofon mit Nieren-Charakteristik nimmt Schall von vorne mit *0 dB* hingegen Schall mit der Einfallsrichtung *120°* um ca. *10 dB* gedämpft auf. Betrachtet man das Mikrofon für den hinteren linken Kanal (*Abb. 27*) wird es den Schall von hinten links mit 0 dB aufnehmen, den Schall von vorne um ca. *20 dB* gedämpft und den Schall von vorne links mit ca.*15 dB* Dämpfung.

Bei der Matrizierung der Surroundersoftware werden fünf neue Mikrofone mit der gleichen Richtcharaketeristik aber anderer Ausrichtung berechnet. Die Software Surrounder kann im Modus Laufzeiten die fünf virtuellen Mikrofone gemäß der *ASM 5*-Anorndnung positionieren, dadurch entstehen zwischen den Mikrofonen natürlich unterschiedliche Laufzeiten. Bei einer *ASM 5* Aufnahmen wird das vordere linke Mikrofon auch Signale aus Richtung der anderen vier Mikrofone aufnehmen, eben nur verzögert. Geht man in diesem Fall von Nierenmikrofonen aus, wie es auch der *Surrounder 2to5* macht, wird das vordere linke Mikrofon die Signale aus allen fünf Richtungen aufzeichnen. Gemäß der Nieren-Charakteristik werden die Pegel gemäß der Richtung unterschiedlich sein zusätzlich wird seitlich auftreffender Schall in den Höhen bedämpft aufgezeichnet. Genau dies geschieht in der Software. Im Laufzeiten-Modus werden einem Signal die anderen vier Signale mit einstellbarem Pegel und Laufzeiten zugemischt. Dabei ergeben die Abstände des einen Mikrofons zu den jeweils Anderen natürlich unterschiedliche Schallverzögerungen und Intensitäten. Der Anwender kann den Übersprechpegel im Modus Laufzeiten selbst einstellen. Der eingestellte Pegelwert gibt den kleinsten Pegelunterschied des Mikrofons zu einem der vier anderen Mikrofone an. Wird der Wert auf *-6 dB* eingestellt, wird z. B. dem linken Signal das Signal mit dem höchsten Übersprechpegel mit *-6 dB* zugemischt (für diesen Fall der Centerkanal, da dieser dem Linken am nächsten liegt)(*Abb.28*), die drei anderen Signale werden anteilig zu den *-6 dB* mit Faktoren von *0,5-0,9* dazu gemischt. Die Faktoren wurden bezüglich des Polardiagramms der allgemeinen Niere grafisch ermittelt. Hier ein Beispiel des linken Kanals im Javacode.

```
linksResult[i]=d.setDelayArray(m.l[i],0.9f*x*mII.c[i],0.8f*x*mIII.r[i]
,                                  0.6f*x*mIII.hl[i],0.5f*mIIII.hr[i]);
```

Die Laufzeiten zwischen den Mikrofonen, welche der Surrounder verwendet, wurden aus der *ASM 5*-Technik abgeleitet. Um die Leistung nicht unnötig zu beinträchtigen verwendet der Surrounder nur drei unterschiedliche Laufzeiten, nicht sechs wie es dem *ASM 5*-Prinzip entsprechen müsste (*Abb.28*). Die Verzögerung durch Laufzeiten ist mit der skip()-Methode realisiert. Es werden vier AudioInputStreams erzeugt. Einen Stream für die Matrizierung, Die

anderen Streams ließen sich mit der skip()-Methode um die jeweiligen Laufzeitenwerte verzögern. Diese verzögerten Streams werden je nach eingestelltem Pegelwert dem Signal, als Übersprechen, zugefügt. Z.B. die Variable `gip.getLaufzeitenLong` liefert die Anzahl der Bytes die im AudioInputStream übersprungen werden sollen.

```
long lo=aisI.skip(gip.getLaufzeitenLong());
long n=aisII.skip(gip.getLaufzeitenBack());
long m=aisIII.skip(gip.getLaufzeitenFront());
```

Verbindet man alle Mikrofone einer ASM 5-Anordnung mittels Geraden, dienen diese Geraden als Strecken. Hat man eine festgelegte Geschwindigkeit, in diesem Fall die Schallgeschwindigkeit $c = 340\ m/s$, ergibt sich daraus die Laufzeit zwischen den Mikrofonen. Die einzelnen Strecken s lassen sich rechnerisch wie folgt ermitteln. Die Laufzeiten t wurden mit der Formel $t = \frac{s}{c}$ berechnet

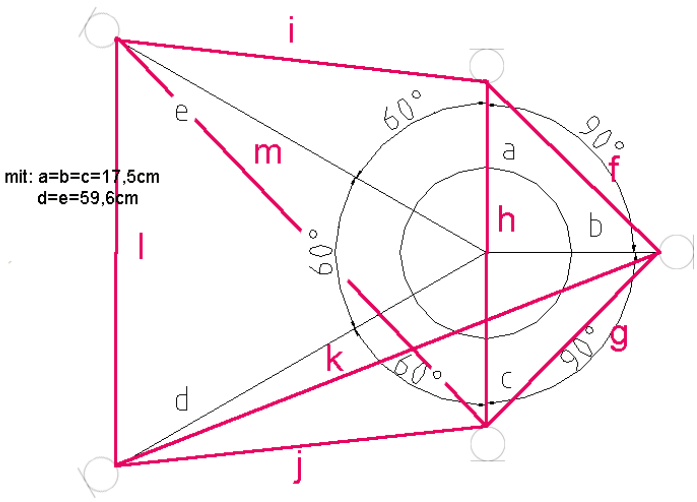

Abb.28 ASM 5-Verfahren mit den sechs möglichen Laufzeiten f/g, h, i/j, m, k, l

$$f = g = \sqrt{a^2 + b^2} = \sqrt{c^2 + b^2} = \sqrt{17{,}5^2 cm + 17{,}5^2 cm} = 24{,}7cm$$

das entspricht bei einer Schallgeschwindigkeit von *340 m/s* einer Laufzeit von ***0,72 ms***

$$h = \sqrt{f^2 + g2} = 34{,}9cm$$

das entspricht bei einer Schallgeschwindigkeit von *340 m/s* einer Laufzeit von ***1,02 ms***

$$i = j = \sqrt{c^2 + d^2 - 2cd * \cos 60°} = 53{,}0cm$$

das entspricht bei einer Schallgeschwindigkeit von *340 m/s* einer Laufzeit von ***1,55 ms***

$$l = \sqrt{d^2 + e^2 - 2ed * \cos 60°} = 59{,}6cm$$

das entspricht bei einer Schallgeschwindigkeit von *340 m/s* einer Laufzeit von ***1,72 ms***

$$k = \sqrt{j^2 + g^2 - 2\,jg * \cos(90° + 60°)} = 75{,}4cm$$

das entspricht bei einer Schallgeschwindigkeit von *340 m/s* einer Laufzeit von ***2,21 ms***

$$m = \sqrt{c^2 + e^2 - 2ec * \cos(60° + 60°)} = 68{,}9cm$$

das entspricht bei einer Schallgeschwindigkeit von *340 m/s* einer Laufzeit von ***2,02 ms***

Wenn man sich alle sechs Laufzeiten betrachtet sind jeweils zwei Werte nah beieinander, die Software verwendet daher nur drei Werte und kann dadurch einiges an Rechenleistung einsparen.

Was bisher noch nicht berücksichtigt wurde ist natürlich die Tatsache, dass alle matrizierten Signale sehr ähnlich und/oder phasengedreht sind. Würde man im Modus Laufzeiten alle fünf Signale mit den gegebenen Laufzeiten mischen würden sich tiefe Frequenzen aufaddieren und bis *T/2* zum Teil wieder auslöschen, bei den phasengedrehten Signalen genau umgekehrt. Bei einer Laufzeit von *T/2=1ms* wäre dies bei der Frequenz *f= 250Hz*. Daher werden alle vier Signale, die zugemischt werden sollen, bis *300Hz* gefiltert. Wie besprochen, werden auch die hohen Frequenzen bedämpft, zum einen weil die hohen Frequenzen eher für das Richtungswahrnehmen zuständig sind und dadurch die Richtung des Primärsignals verfälschen würden, zum anderen weil sich die, hier gewählte, Nierencharakteristik zu den hohen Frequenzen hin verschmälert und die hohen Frequenzen zur Seite schwächer abbildet. Diese höhen- und tiefenbedämpfte Filterkurve lässt sich auch als Bandpass realisieren. In der Software wurde ein Bandpass verwendet in dem die unter und die obere Grenzfrequenz zur Berechnung eingeben wurde. Die Filterkurve wurde mit dem Programm *Winfilter*[46] entwickelt und sie wie folgt aus:

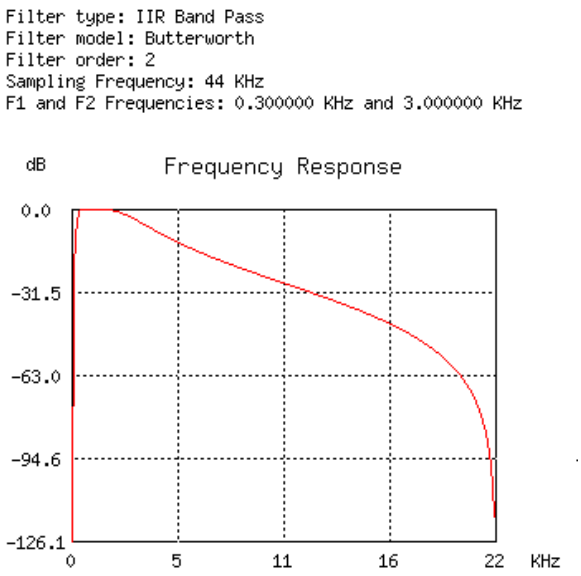

Abb.29BandpassButterworthfilter

[46] Kapitel 7.4 Entwerfen eines Filters mit Winfilter

7 Digitale Filter

Digitale Filter haben prinzipiell die gleichen Aufgaben wie analoge Filter und dienen zur Manipulation eines Signals. Es werden Frequenzen oder Frequenzbereiche gesperrt oder durchgelassen. Analoge Filter können passiv mit Spulen, Widerständen und Kondensatoren oder aktiv mit Operationsverstärker realisiert werden.

Abb. 30 passive Tiefpassschaltung mit Filterkurve[47]

Digitale Filter hingegen werden mit Logikbausteinen oder in einem Programm mit Signalprozessor gefertigt. Dazu werden drei Komponenten benötigt ein Verzögerungsglied ein Multiplikator und ein Addierer.

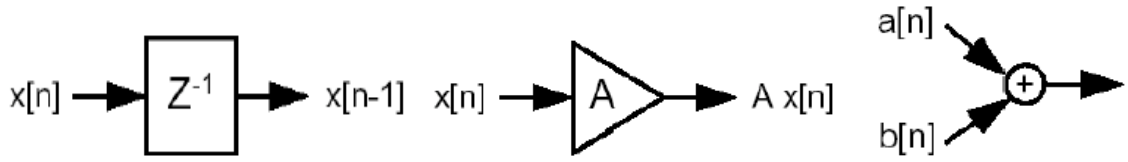

Abb. 31 Verzögerungsglied, Multiplikator und Addierer[48]

Es werden keine kontinuierlichen Werte sondern zeitdiskrete Werte verarbeitet. Ein zeitdiskretes Signal besteht in der zeitlich periodischen Abfolge nur aus einzelnen Impulsen, welchen den Signalverlauf über die Zeit darstellen, den jeweiligen Abtastwerten (*siehe Kapitel 4.2.1 und 6.3.1*)[49]. Vorteile digitaler Filter sind die hohen Genauigkeiten, die lineare Phase und die einfache Umsetzung. Die Filter werden in vier grundlegende Funktionsbereiche untergliedert in Bandsperre, Bandpass, Tiefpass und Hochpass. Außerdem wird die rekursive und nicht rekursive Variante unterschieden.

[47] http://de.wikipedia.org/wiki/Tiefpass

[48] http://www.sigterm.de/projects/digifil/html/node7.html

[49] http://de.wikipedia.org/wiki/Digitales_Filter

7.1 FIR - nicht rekursive Filter (Finite Impulse Response)

FIR-Filter berechnen das Ausgangssignal aus mehreren, teilweise zwischengespeicherten Werten des Eingangssignals. Eine Rückführung vom Ausgang gibt es nicht. Sie besitzen eine endliche Impulsantwort[50]. Aus diesem Grunde sind diese Filter unbedingt stabil.

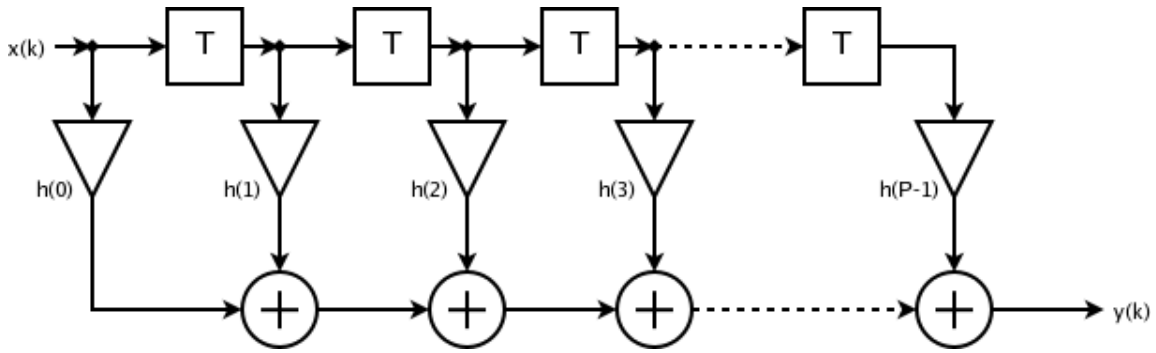

Abb.32 Abbildung eines Schaltbildes für einen FIR-Filter mit Addierer, Multiplikator, Verzögerung

Die Differenzgleichung eines FIR-Filter lautet:

$$y[k] = \sum_{i=0}^{n} \beta_i \cdot u_{k-i}$$

Die umgeformte Übertragungsfunktion:

$$H(z) = \sum_{i=0}^{n} \beta_i \cdot z^{-i}$$

Durch Umformen lässt sich die Übertragungsfunktion, mit βi als den Filterkoeffizienten, ableiten. Ein FIR-Filter dieser Form ist daher in der z-Ebene immer durch *n* Nullstellen und einen n-fachen Pol im Ursprung bei *0* gekennzeichnet. Der Frequenzgang wird daher bei einem FIR-Filter ausschließlich durch die Nullstellen in der z-Ebene bestimmt.[51]

Eigenschaften der FIR-Filter:
-FIR-Filter sind immer stabil, da die einzige n-fache Polstelle der Übertragungsfunktion immer im Ursprung und somit innerhalb des Einheitskreises liegt
-Die Gleichspannungsverstärkung ist gleich der Summe aller Filterkoeffizienten.

[50] Die Impulsantwort oder Gewichtsfunktion ist das Ausgangssignal eines eindimensionalen, linearen, zeitinvarianten, dynamischen Systems

[51] http://de.wikipedia.org/wiki/Filter_mit_endlicher_Impulsantwort

7.2 IIR rekursive Filter (infinite impulse response)

IIR-Filter verwenden zusätzlich zu den Eingangswerten auch zwischengespeicherte Werte des Ausgangssignals. Durch die Rückführung von Ausgangswerten wird die Stabilität von IIR-Filtern beeinträchtigt, was es schwieriger macht stabile Filter zu entwerfen. Dafür kann ein FIR-Filter durch ein IIR-Filter, mit gleicher Filkterkurve, mit weitaus weniger Rechenaufwand ersetzt werden. Besonders Übertragungsfunktionen analoger Filtertechnik lassen sich mit IIR-Filtern bewerkstelligen, wie z.B. Besselfilter, Buttwerworthfilter, Cauerfilter und Tschebyschefffilter.

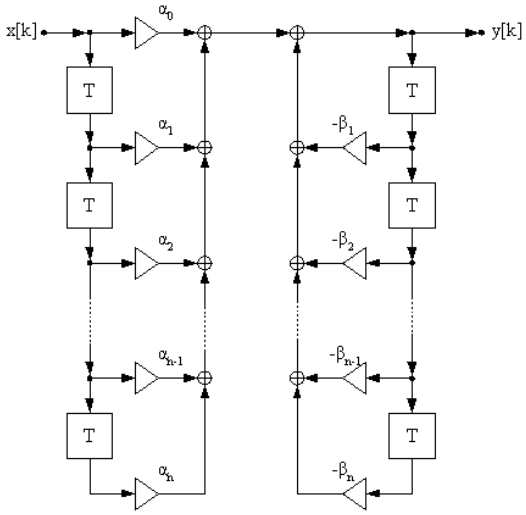

Abb. 33 IIR Filterstruktur mit den zwei Faktoren α, β und Rekursivität

Im zeitdiskreten Fall ist ein digitales Filter durch eine betragssummierbare Folge *S[n]* definiert, wobei jedem zeitdiskreten Signal *x[n]* ein Signal *y[n]=(S*x)[n]* zugeordnet wird, d. h.

$$y[n] = \sum_{k=-\infty}^{\infty} S[k]x[n-k]$$

daraus folgt für die Systemfunktion:

$$H(z) = \frac{a_2 \cdot z^{-}2 + a_1 \cdot z^{-}1 + a_0}{b_2 \cdot z^{-}2 + b_1 \cdot z^{-}1 + 1}$$

Im Allgemeinen sind zur Auswertung eines digitalen IIR-Filters unendlich viele Berechnungsschritte für jedes Glied *y[n]* erforderlich. Im Spezialfall eines rekursiven Systems gibt es jedoch auch eine endliche Darstellung, welche jedoch in der Ausführung der Berechnung eine unendliche Ein- und Ausschwingphase verlangen würde.

7.3 IIR vs. FIR

Die Vorteile eines IIR-Filter gegenüber einem FIR-Filter liegen zum einen darin, dass man, um einen vergleichbaren Filtereffekt zu erreichen, die erforderliche Ordnung deutlich niedriger ansetzen kann als für ein FIR-Filter. Diese Eigenschaft hat schließlich zur Folge, dass Rechenaufwand und Speicherbedarf eines IIR-Filters geringer ausfallen werden als bei einem entsprechenden FIR-Filter. Ein FIR-Filter hat jedoch einige Vorteile. Er ist es immer stabil, da er ohne Rückkopplungszweig auskommt. IIR-Filter können, wie auch analoge Filter, unter bestimmten Bedingungen zu Schwingen anfangen. Durch begrenzte Rechengenauigkeit auftretende Rundungsfehler sind ohne Rückkopplungszweig weniger problematisch. Auch sind linearer Phasenverlauf sowie konstante Gruppenlaufzeit bei einem FIR-Filter gewährleistet.

Eigenschaft	FIR	IIR
Rückkopplung	keine, endliche Impulsantwort	vorhanden, eventuell unendlich lange Impulsantwort
Komplexität	mehr Multiplikationen und Additionen nötig, Struktur einfacher	weniger Multiplikationen und Additionen nötig
Lineare Phase	ja	nein, nur annährend
Stabilität	ja	nur bei richtiger Wahl der Koeffizienten
Design	zum Teil einfach, ansonsten spezielle Algorithmen	Algorithmen für analoge Filter werden verwendet

Tabelle 6 Vergleich IIR- und FIR-Filter

7.4 Entwerfen eines Filter mit Winfilter

Winfilter ist eine Open-Source-Software mit der IIR- oder FIR-Filter entworfen werden können.[52] Man kann zwischen verschiedenen Filtermodellen (*Butterworthfilter*, *Schebyschefffilte*r und *Besselfilter*) und Filtertypen (Bandpass, Bandsperre, Hochpass und Tiefpass) wählen. Es wird die Samplefrequenz angegeben, die Filterordnung sowie die zu filternde Frequenz. Als Ergebnis liefert dieses Programm die Filterkurve, Phasenlage des Filters, Gruppenlaufzeiten, Polstellen und die Impulsantwort.

[52] http://www.winfilter.20m.com/

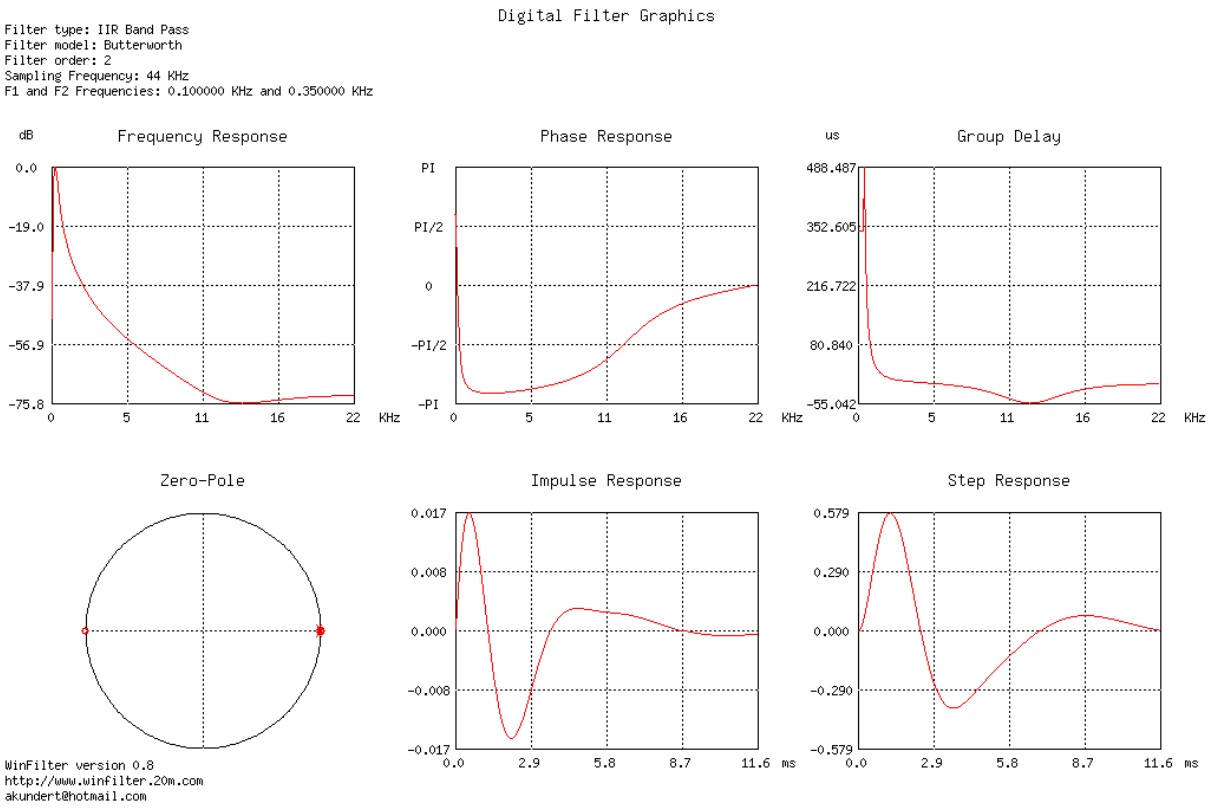

Abb. 34 Winfilter IIR Bandpass

Zusätzlich kann man sich die Filterfaktoren mit dem zugehörigen Code in *C* oder *VHDL* ausgeben lassen. Dieser Code kann sehr leicht in *Java* übersetzt werden. Die Implementierung eines IIR-Filter sieht wie folgt aus:

```
float iirBandpass(float NewSample) {
      int n;
      filterLaenge= BCoef.length;


      //shift the old samples
      for(n=BCoef.length-1; n>0; n--) {
            x[n] = x[n-1];
            y[n] = y[n-1];
          }

      //Calculate the new output
      x[0] = NewSample;
      y[0] = (float) (ACoef[0] * x[0]);
      for(n=1; n<=BCoef.length-1; n++)
             y[0] += ACoef[n] * x[n] - BCoef[n] * y[n];
             return y[0];
      }
```

In der Software Surrounder wurden IIR-Filter des Typs *Butterworth* verwendet, da diese mit niedrigeren Ordnung, genauerer Filterkurve (ohne Kammfiltereffekte) und erhöhter Performance entwickelt werden konnten. Speziell bei höheren Sampleraten war der

Unterschied, bereits in den Filterkurven zu sehen. Zur besseren Genauigkeit du Vermeiden eines Aufschwingens sind die Filterfaktoren ebenfalls *32 Bit* Fliesskommazahlen. Als Beispiel soll der gleiche Filter wie in *Abb.34* als FIR-Filter implementiert werden. Hier muss bereits mit 32 Filterfaktoren gerechnet werden, außerdem ist die Filterkurve alles andere als linear. Der Kammfiltereffekt ist deutlich zu erkennen.

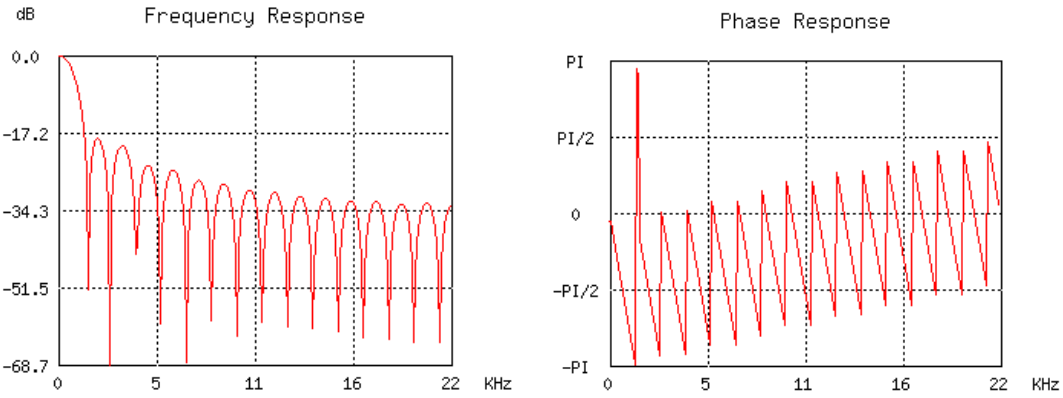

Abb. 35 FIR Filter

8 GUI- Die Grafikprogrammierung mit Thinlet[53]

Der Entwurf eines Grafical User Interface (*GUI*) kann auf mehreren Wegen beschritten werden. Es werden *API*`s plattformunabhängig sowie plattformabhängige angeboten. Zur ersteren Variante zählen *AWT* und *SWING*, zur letzteren *SWT*. Das Abstract Window Toolkit (*AWT*) ist Bestandteil der Java Foundation Classes mit dessen *API* plattformunabhängig und somit Komponenten des Betriebssystems mit eingebunden werden. *SWING*, das von *Sun Microsystems* entwickelt wurde, ist eine Weiterentwicklung von AWT und bietet zusätzliche Möglichkeiten wie z.B. Drag & Drop, Look & Feels, der Gebrauch von Tastenkombinationen und Tooltips. Das Standard Widget Toolkit (*SWT*) ist ebenfalls eine Bibliothek für die Erstellung grafischer Oberflächen und wurde von *IBM* für die Entwicklungsumgebung Eclipse entwickelt. Diese Schnittstelle wurde hauptsächlich für Windows entworfen und kann somit nicht auf allen Betriebsystemen fehlerfrei ausgeführt werden. Zusätzlich muss SWT auf dem benutzten System verfügbar sein.

Eine Alternative hierzu bietet *Thinlet*. Mit *Thinlet* kann schnell und unproblematisch eine GUI erstellt und programmiert werden. *Thinlet* ist ein Open Source Produkt und wurde in dieser Diplomarbeit verwendet. Es versteht sich als eine Alternative zu *SWING* und *SWT* und kommt im Gegensatz zu *SWING* mit weitaus weniger Resourcen und einer alten Java Runtime Environment *JRE* aus. Die *GUI* wird als *XML* beschrieben. Dies ermöglicht einen schnellen Aufbau der grafischen Oberflächen. Der Source-Code bleibt übersichtlich und elegant. Der eher umständliche Aufbau und die Anordnung der *AWT* oder *SWING* Komponenten wird durch das Wegfallen des Layout-Managers und des Schreibens der Listener erleichtert. Um mit der Thinlet-Biblitothek arbeiten zu können, muss die Datei thinlet.jar importiert werden. Diese wird beim Compilieren und beim Ausführen des Programms benötigt. Eine *JAR*-Datei ist ein Archiv mit verschiedenen Klassen und dient zur Erweiterung eines Programms oder Implementierung einer neuen Schnittstelle. Das thinlet.jar enthält Klassen, mit denen die so genannten Components hergestellt werden können, welche gleichbedeutend zu den Komponenten in *AWT* oder *SWING* sind. Die Components sind die Bestandteile oder Elemente der *GUI*. So werden Buttons, Labels, Dialoge, Checkboxes, Textareas etc. als Components betrachtet. Jede Component besitzt Eigenschaften und Attribute, die im Rahmen der Interaktion mit Methoden verändert und gesetzt werden können.

[53] http://www.thinlet.com

8.1 Interaktion zwischen GUI und dem Javacode

Die Hauptklasse Project der Diplomarbeit leitet sich von *Thinlet* ab, und parst die *XML*-Konfiguration und startet mittels FrameLauncher die *GUI*. Hier ein Beispiel des Exception[54]-Dialogs in *XML* geschrieben (*siehe Abb.35*).

```
<dialog  text="" modal="true" columns="1" background="#6F5F90" >
<label icon="exception.gif"  />
<panel  halign="center" >
<label text="" name="label" font="Arial Black 12 italic bold"
halign="fill"/>
</panel>
<button text="close" action="closeException()" font="Arial Black 14 italic
bold"/>
</dialog>
```

Abb. 35 Screenshot des Exception-Dialogs mit dem String „ AudioFormat wird nicht unterstützt“

Im Konstruktor der Klasse Project werden alle *XML*-Dateien geparst auch der Exception-Dialog. Die `"index.xml"` ist die eigentliche GUI und wird im Konstruktor mit der Methode `add(this.Programm)` gleich in der main-Methode gerufen und am Bildschirm angezeigt.

```
public Project()  {
            try {
                  this.Programm = parse("index.xml");
                  add(this.Programm);
                  this.dialogHelp = parse("help.xml");
                  this.dialogException=parse("exception.xml");
                  this.dialogProgress=parse("progress.xml");
                  }
```

Erst durch die add()-Methode wird der Dialog auf dem Bildschirm angezeigt. Allen Dialogen wurde in der Project-Klasse zwei Methoden gewidmet. Eine Methode zum Anzeigen `public void exception()` inklusive der add()-Methode, und eine Methode

[54] Exceptions bieten Möglichkeiten bei Programmfehlern zu reagieren um den Absturz zu verhindern

`public void closeException()` inklusive der remove()-Methode um den Dialog zu schließen.

```
public  void exception(String s){
             remove(dialogProgress);
             Object Panel = find(this.dialogException, "label");
             // die Meldung ausgeben
             setString(Panel, "text", s);
             // Dialog anzeigen                                          `
`            add(this.dialogException);

public void  closeException() {
             remove(dialogException);
             }
```

Anhand des Beispieles des Exception-Dialogs kann man auch die Interaktion zwischen der GUI und des eigentlichen Javaprogramms beschreiben. In den *XML*-Dateien können bei ausgeführten Aktionen einer Component mit „action“ ein Methodenaufruf im Javaprogramm gestartet werden. (*siehe S. 60 Javacode Zeile 7*). Zum Beispiel ruft das Drücken des Buttons mit dem Text „close“ die Methode closeException() in der Project-Klasse.

```
<button text="close" action="closeException()" font="Arial Black 14
```

Die closeException-Methode wiederum ruft die remove()-Methode aus dem Thinlet-Package und schließt den Dialog.

```
public void  closeException() {
             remove(dialogException);
```

Umgekehrt soll auch die *GUI* auf das Javaprogramm reagieren können. Auch hier dient der Exception-Dialog wieder als gutes Beispiel. Jede Exception oder Fehler die im Javaprogramm entstehen, soll einen Exception-Dialog mit Text auf dem Bildschirm anzeigen und den Anwender über den Fehler informieren. Der Text wird mit dem Label erzeugt. Ein Attribut oder Eigenschaft dieses Labels ist Text, dieser kann mit der setString()-Methode aus dem *Thinlet*-Package gesetzt und verändert werden. Die setString()-Methode benötigt dazu die Component, dessen Attribut verändert werden soll, der Name des Attributs und der Wert, der gesetzt werden soll.

```
setString(Label "text", “Audiodatei wird nicht unterstützt“ );
```

Um die Component zu finden, muss diese eindeutig durch das Attribut „name“ definiert sein. In diesem Fall heißt die Component „label“.

```
<label text="" name="label" font="Arial Black 12 italic bold"
```

Die find()-Methode des Thinlet-Package liefert diese Component als Object zurück. Dafür benötigt sie die XML-Datei sowie den Namen der Component.

```
Object Label = find(this.dialogException, "label");
```

So können sämtliche Components einer *GUI* an das Javaprogramm gekoppelt werden. Thinlet liefert bereits eine ganze Menge Components, Attribute und Methoden, die in einer *API* zusammengefasst wurden und dadurch einen schnellen Einstieg ermöglichen. Eine schöne Beschreibung findet man im Internet unter http://thinlet.sourceforge.net/home.html.

9 Versuchsdurchführung im Übertragungswagen

Der Algorithmus der Surroundersoftware wurde im Ü16 des SWR Stuttgart nachgebildet und Testpersonen vorgespielt. Dazu wurden verschiedenste Aufnahmen in Blumleinstereofonie sowie eine Musikprodukion im Einzelmikrofonverfahren verwendet. Die Signale wurden in die Audiosoftware *Sequoia* importiert und über die Auspielwege matriziert. Auch das Mischen der einzelnen Kanäle mit den jeweils vier Verzögerten (Übersprechen) wurde über diese Auspielwege realisiert. Die fünf Ausgänge (Links, Center, Rechts, Hintenlinks, Hintenrechts) wurden auf das Mischpult des Ü-Wagen geroutet und von dort direkt per Steckfeld auf die Surroundlautsprecher geleitet. Als Mischpult wurde das *mc66²* der Firma *Lawo* verwendet. Der Filter für die Abschwächung der Frequenzen *1,2 kHz* und *12 kHz* der hinteren Kanäle wurde direkt am Pult eingerichtet und konnte per Knopfdruck ein und ausgeschaltet werden. Für den Test wurde die Terminologie der *EBU R 90-2000* benutzt. Im Rahmen der Diplomarbeit wurden 10 Mitarbeiter der SWR Stuttgart Außenübertragung Hörfunk getestet, die alle Erfahrung in Mehrkanalaufnahmetechniken sammeln durften und daher in dieser doch noch recht neuen Welt als zunächst geschult betrachtet werden können. 10 Testpersonen ist für eine wissenschaftliche Aussage natürlich zu wenig und soll daher auch nur eingeschränkt orientierend sein. Zusätzlich ist die Abhörsituation für Mehrkanalaufnahmen im Ü-Wagen nicht als optimal zu betrachten. Der Sweet-Spot ist durch den begrenzten Raum im Ü-Wagen nur sehr klein und maximal für eine Person eingerichtet.
Die Laufzeiten für das Übersprechen wurde auf *3ms*, *10ms* und *20ms* und *40ms* festgelegt, was bei einer Schallgeschwindigkeit von *340m/s* Wegdifferenzen von *1m, 3,4m, 6,8m* und *13,6m* entspricht. Die Probanden sollten den jeweiligen Übersprechpegel auf das Optimum bezüglich Räumlichkeit einstellen und wurden nicht über die eingestellten Werte oder genaueren Details informiert. Als Tonmaterial diente ein Ausschnitt des klassischen Trios Triology, aufgenommen im Schwetzinger Schloss, Mitschnitt einer Intensivübergabe im Krankenhaus, das Lied „Georgia“ vom Autoren selbst produziert, Aufnahme von Strassengeräusche einer Bundestrasse und Aufnahmen einer kreisförmigen Bewegung mit einem Schellenkranz. Da die Ergebnisse doch sehr unterschiedlich ausfielen sollen hier nur die Tendenzen erläutert werden. Zum Verständnis und zur Bedienung der Software helfen sie allemal.

- die Absenkung *1,2 kHz* und *12 kHz* der hinteren Lautsprecher bewirkte bei allen Testpersonen ein präziseres Klangbild der Front, und wurde für Musik positiv betrachtet
- bei unbekannten Schallquellen (z.B. kreisförmige Bewegung mit Schellenkranz um die Mikrofone herum) war die Lokalisationsschärfe unpräzise und wurde sprunghaft eingestuft, bei bekannten Schallquellen (z.B. vorbeifahrende Autos) war die Lokalisationsschärfe präziser
- das Übersprechen erhöht die Räumlichkeit und verschlechtert die Lokalisationsschärfe
- je kürzer die Laufzeit der übersprechenden Kanäle, desto höher wurde der Übersprechpegel eingestellt, genaue Pegel sollen hier nicht genannt werden, da das Übersprechen nicht genau dem Übersprechen der Software Surrounder entsprach und die Ergebnisse im Gesamten zu unterschiedlich waren
- auch die Matrizierung der Stereomischung mit Einzelmikrofonverfahren und Panoramapotentiometer wurde als angenehm empfunden, der Übersprechpegel wurde tendenziell um *3dB* niedriger als beim den Blumleinverfahren eingestellt.
- Bei Sprache wurde der Übersprechpegel eher niedriger gewählt

10 Fazit

Das Prinzip, aus einem Blumleinverfahren Achten mit beliebiger Ausrichtung herzustellen, funktioniert. Durch die fehlenden Laufzeiten dieses Verfahren leidet jedoch die räumliche Abbildung. Die Software Surrounder gleicht dieses im Laufzeitenmodus aus. Dadurch wird die Räumlichkeit erhöht, kann aber sicherlich nicht mit echten Laufzeitenmikrofonierungs-Techniken mithalten. Die Lokalisationsschärfe, hierbei, funktioniert eher bei Geräuschen die dem Hörer bekannt sind und derer Bewegung nachvollzogen werden kann. Für Mitschnitte im Film- und Videobereich ist dieses Konzept sicherlich interessant. Das Blumleinverfahren liefert gute Stereoaufnahmen, bei denen die Stereobreite im nach hinein, durch das Mischungsverhältnis der Seiten/Mittenanteile verändert werden kann. Eine Verbreiterung der Stereobreite für das 16:9 Format kann unter Umständen nützlich sein. Solch eine Aufnahme in ein 5.0 Format um zu wandeln ist eine zusätzliche Option die überzeugen kann. Die eher negativen Seiten sollen nicht verschwiegen werden. So sind Druckgradientenmikrofone[55] empfindlicher für Wind- und Griffgeräusche. Die dadurch benötigten Windkörbe verändern die Richtcharakteristik der Mikrofone und können eine genaue Matrizierung erschweren. Was als großer Nachteil betrachtet werden darf, ist die Tatsache, dass die Software nicht in Echtzeit funktioniert. Einstellungsänderungen der Software können nicht sofort gehört werden, die Ausgangsdateien müssen zuerst in einen Editor importiert und dann dort überprüft werden. Dieses Problem entsteht durch das Fehlen einer funktionierenden ASIO[56]-Schnittstelle für Java. Ob es mittlerweile funktionierende Open- Source ASIO-Interfaces gibt ist dem Autoren nicht bekannt, soll aber auch nicht verneint werden.
Im Großen und Ganzen hat die Recherche, die Mitschnitte und die Arbeit sehr viel Spaß bereitet und ist von allen Beteiligten, als äußerst interessant betrachtet worden.

55 Mikrofone die auf Druckunterschiede reagieren, z.B. Acht, Niere, Superniere, Richtrohr

56 Audio Stream Input/Output (ASIO) ist ein von Steinberg entwickeltes, plattformübergreifendes, mehrkanalfähiges Audiotransfer-Protokoll.

11 Abkürzungsverzeichnis

AC3: ist ein Mehrkanal-Tonsystem der Firma Dolby

aiff: Audio Interchange File Format ein Containerformat für Audiodaten von Apple entwickelt

API: Application Programming Interface, englisch für Programmierschnittstelle in der Informatik

ASM 5: Adjustable Surround Microphone wurde nach ITU 775 entwickelt

EBU: European Broadcasting Union

DTS: Mehrkanal-Tonsystem der gleichnamigen kalifornischen Firma Digital Theater Systems

GUI: Graphical User Interface, Bedienoberfläche einer Software

FIR: finite impulse response filter, Filter mit endlicher Impulsantwort

IIR: infinite impulse response filter , Filter mit unendlicher Impulsantwort

INA*:* Ideale Nierenanordnung von Günther Theile entwickelt

IRT: Institut für Rundfunktechnik

ITU: International Telecommunication Union

JDK: Java Development Kit, Entwicklnugsumgebung für Java Programmierung

MPEG: Moving Picture Experts Group

PCM: Pulscodemodulation, Wandlertechnik von analog nach digital

THX: von George Lucas entwickelter Standard für Mehrkanalton in Kinos

wav: ein Containerformat für Audiodaten von Windows entwickelt

XML: Extensible Markup Language, wird für den Austausch von Daten zwischen unterschiedlichen IT-Systemen eingesetzt

12 CD-Inhalt

- Software Surrounder 2to5 mit Windows-Installer
- Sourcecode der Softrare in Java
- XY-Mitschnitt des Trios Triologie im Schwetzingern Schloss
- XY-Mitschnitt des Klavierkonzert von Oli Mustonen im Schwetzingern Schloss
- MS-Aufnahme Übergabe auf Intensivstation
- MS-Aufnahme einer Bundesstrasse
- MS-Aufnahme Ralle und Hille klampfen in der Kirche Epfendorf