LEHRSTUHL FÜR REALZEIT-COMPUTERSYSTEME
TECHNISCHE UNIVERSITÄT MÜNCHEN
UNIV.-PROF. DR.-ING. G. FÄRBER

Räumliche Ortung und Separation von Geräuschquellen im Bereich der mobilen Servicerobotik

Robin Gruber

Diplomarbeit

Räumliche Ortung und Separation von Geräuschquellen im Bereich der mobilen Servicerobotik

Diplomarbeit

Betreut durch den Lehrstuhl für Realzeit-Computersysteme
Technische Universität München
Prof. Dr.-Ing. Georg Färber

Ausgeführt am Deutschen Zentrum für Luft- und Raumfahrt
Institut für Robotik und Mechatronik

Betreuer: Dr. Gerhard Grunwald (DLR)
Dipl.–Ing. Stefan Blum (RCS)

Bearbeiter: Robin Gruber
Lindenschmistraße 29a
81371 München

Eingereicht im Dezember 2000

Danksagung

An dieser Stelle möchte ich mich bei Professor Färber bedanken, der meine externe Diplomarbeit ermöglicht hat.

Ebenso gebührt mein Dank meinen Betreuern Dr. Gerhard Grunwald und Dipl.–Ing. Stefan Blum, die für meine Probleme immer ein offenes Ohr hatten und mich sowohl bei technischen als auch logistischen Fragen unterstützten.

Zusammenfassung

Die vorliegende Arbeit stellt einen Algorithmus zur Ortung — im Sinne einer Raumrichtungsbestimmung — einer kontinuierlich aktiven Schallquelle (Sprache, Musik, Maschinengeräusche usw.) zusammen mit seiner praktischer Implementierung in Hard- und Software sowie Tests seiner Leistungsfähigkeit vor. Außerdem wird eine Möglichkeit zur Sprachaktivitätsdetektion und eine Form eines simulierten Richtmikrofons ebenfalls zusammen mit der praktischen Realisierung beschrieben.

Es wird dabei sowohl auf den entwickelten Algorithmus, den Aufbau der erforderlichen Hardware, die Funktionsweise des Ortungs- und aller Zusatzprogramme, als auch auf die Inbetriebnahme auf einem Linux-Rechner eingegangen.

Der Algorithmus — in dieser Arbeit als "Subtraktiver Steered Beamformer" bezeichnet — ist eine Abwandlung des bereits erfolgreich für Schallortung eingesetzten Steered Beamformers, mit dem Unterschied, dass die Differenz zweier Mikrofonsignale und nicht deren Summe für eine Ortung herangezogen wird. Damit werden die Vorteile des Steered Beamformers (z.B.: Möglichkeit zur Beschränkung des Suchwinkels, einfacher Algorithmus, leichte Zusammenfassung der Mikrofonsignale) um den Vorteil einer höheren Genauigkeit durch die Differenzbildung ergänzt.

Die hierfür entwickelte Hardware besteht aus vier tetraederförmig angeordneten omnidirektionalen Mikrofonen nebst Verstärkern und zwei handelsüblichen Soundkarten. Die in C++ verfasste Software bietet neben der Hauptfunktion Schallortung, Sprachaktivitätsdetektion und simuliertes Richtmikrofons auch noch zusätzliche Werkzeuge, wie ein Oszilloskop, einen Vierspurrekorder und eine Richtcharakteristiksimulation.

Inhaltsverzeichnis

Abbildungsverzeichnis ii

1. Einleitung 1

2. Methoden der Schallortung 3
2.1. Steered Beamformer . . . 3
2.1.1. Darstellung der Richtcharakteristik . . . 3
2.1.2. Prinzip eines simulierten Richtmikrofons . . . 5
2.1.3. Schallortung durch ein simuliertes Richtmikrofon . . . 10
2.2. Laufzeitdifferenzschätzung . . . 10
2.3. Subtraktiver Steered Beamformer . . . 13
2.3.1. Prinzip . . . 13
2.3.2. Verwendung mehrerer Mikrofonpaare für die Ortung . . . 15
2.3.3. Verwendung mehrerer Mikrofone als Richtmikrofon . . . 18
2.3.4. Realisierung . . . 19

3. Aufbau der Hardware 21
3.1. Mikrofonträger . . . 21
3.2. Mikrofone . . . 25
3.3. Vorverstärker . . . 26
3.4. Line-Verstärker . . . 29
3.5. Stromversorgung . . . 32
3.6. Soundkarten . . . 33

4. Funktionsweise der Software 35
4.1. Schnittstelle zum ALSA-Treiber . . . 36
4.2. Soundbibliothek . . . 37
4.2.1. Klasse `Sound` . . . 37
4.2.2. Klasse `SoundReceiver` . . . 38
4.3. Schallortung . . . 39
4.4. Sprachaktivitätsdetektion . . . 40
4.5. Simuliertes Richtmikrofon . . . 41
4.6. Zubehör . . . 42

4.6.1. Aussteuerungsanzeige . . . 42
4.6.2. Oszilloskop . . . 43
4.6.3. Mikrofonpositionsanzeige . . . 44
4.7. Hauptprogramm . . . 45
4.8. Weitere Programme . . . 45
4.8.1. Richtcharakteristiksimulation . . . 45
4.8.2. Vierspurrekorder . . . 47
4.8.3. Synchronitätsmessung . . . 47

5. Messungen und Ergebnisse **49**
5.1. Messung mit statischem Empfänger . . . 49
5.2. Messung mit bewegtem Empfänger . . . 51
5.3. Bewertung der Messungen . . . 51

6. Ausblick **55**

Literaturverzeichnis **59**

A. Bedienungsanleitung **61**
A.1. Inbetriebnahme . . . 61
A.2. Abgleich . . . 62
A.3. Betrieb . . . 62

B. Headerfiles **65**
B.1. Datei `lib/micpos.h` . . . 65
B.2. Datei `lib/soundlow.h` . . . 65
B.3. Datei `project/meter.h` . . . 66
B.4. Datei `project/minenergy.h` . . . 66
B.5. Datei `project/oszi.h` . . . 67
B.6. Datei `project/outputplugin.h` . . . 67
B.7. Datei `project/showmicpos.h` . . . 68
B.8. Datei `project/speechactivity.h` . . . 68

C. Datenblätter **69**
C.1. Kondensatormikrofon EB-5400 . . . 69
C.2. Dämpfer . . . 69
C.3. Verstärkertransistor BC 847 C . . . 71
C.4. Operationsverstärker TL 074 . . . 73
C.5. Spannungswandler NMA0505S . . . 74
C.6. NAND-Schmitt-Trigger CD4093 . . . 75

Abbildungsverzeichnis

2.1. Richtcharakteristik eines omnidirektionalen Empfängers 4
2.2. Zusammenfassen der Richcharakteristik bei verschiedenen Frequenzen in ein Diagramm 4
2.3. Prinzip eines Richtmikrofons (Wandler nicht eingezeichnet) 5
2.4. Addition zweier unverzögerter Mikrofonsignale 5
2.5. Richtcharakteristik eines Mikrofonpaars, dessen Signale addiert werden, bei einem Abstand von 200mm und einem Frequenzgemisch von 300-3300 Hz . 6
2.6. Addition zweier Mikrofonsignale mit einstellbarer Signalverzögerung 6
2.7. Richtcharakteristik bei zwei Mikrofonen im Abstand von 200mm bei einem Frequenzgemisch von 300-3300Hz, deren Signale um 50% der Signallaufzeit gegeneinander verzögert sind. 7
2.8. Richtcharakteristik bei Verzögerung der Mikrofonsignale um 100% der Laufzeit 8
2.9. Laufzeitunterschied zwischen einer Kugelwelle und einer ebenen Welle . . . 8
2.10. Berechnung der Verzögerung in Abhängigkeit von der Schwenkrichtung . . 9
2.11. Richtcharakteristik von zwei summierten Mikrofonen mit einem Abstand von 200mm bei 3kHz 11
2.12. Richtcharakteristik von zwei summierten Mikrofonen mit einem Abstand von 200mm bei 300Hz 11
2.13. Laufzeitdifferenzschätzung mit Hilfe eines adaptiven Filters 12
2.14. Prinzip des subtraktiven Beamformers 13
2.15. Richtcharakteristik des subtraktiven Beamformers bei einem Mikrofonabstand von 200mm und einem Frequenzgemisch von 300-3300Hz 14
2.16. Vektorfeld als Ergebnis einer räumlichen Schallortung mit zwei Mikrofonen 16
2.17. Bei Schalleinfall aus x-Richtung spielt nur die x-Komponente der Koordinaten eine Rolle 17
2.18. Schalleinfall aus beliebigen Richtungen kann durch Rotation auf Einfall aus x-Richtung transformiert werden. 17
2.19. Richtcharakteristik eines mit vier Mikrofonen aufgebauten Richtmikrofons bei einem Frequenzgemisch von 300-3300Hz 20

3.1. Tetraederförmige Anordnung der Einzelmikrofone 22
3.2. Praktische Realisierung des Mikrofonständers 22
3.3. Ausrichtung der Mikrofone 23
3.4. Masse-Feder-Tiefpass 23

3.5. Übertragungsfunktion eines Masse-Feder-Tiefpasses bei verschiedenen Dämpfungen . . . 24
3.6. Mechanisches Mikrofonersatzschaltbild . . . 26
3.7. Montierter Mikrofonvorverstärker . . . 27
3.8. Plan des Vorverstärkers . . . 28
3.9. Ersatzschaltbild des Kondensatormikrofons . . . 28
3.10. Schaltplan des Line-Verstärkers . . . 30
3.11. Prinzip des Line-Verstärkers . . . 31
3.12. 5-Volt-Versorgung . . . 32
3.13. Soundkartensynchronisation . . . 34
3.14. Zwei synchronisierte Soundkarten . . . 34

4.1. Zwischenspeicherung der Audiodaten in einem Ringspeicher . . . 37
4.2. Format der im Ringspeicher liegenden Audiodaten . . . 38
4.3. Sprachaktivitätsdetektion . . . 40
4.4. Aussteuerungsanzeige . . . 43
4.5. Oszilloskop . . . 44
4.6. Mikrofonpositionsanzeige . . . 44
4.7. Programm Micro . . . 46

5.1. Messung mit langsam rotierendem Empfänger bei Sprachsignal . . . 51
5.2. Messung mit langsam rotierendem Empfänger bei Musik . . . 52

6.1. Spektrum eines Sprachsignals (kontrastverstärkt) . . . 56
6.2. Spektrum eines Musiksignals (kontrastverstärkt) . . . 56

Kapitel 1

Einleitung

Autonome mobile Roboter gewinnen auch im Service-Bereich immer mehr an Bedeutung. Technische Schwierigkeiten bereitet jedoch immer noch die Mensch-Maschine-Kommunikation, vor allem mit diesbezüglich nicht besonders geschulten Personen.

Die von Menschen bevorzugte Kommunikationsform ist Sprache. Könnte eine Einflussnahme auf das Verhalten eines Service-Roboters verbal erfolgen, so würde das die Akzeptanz seitens des Benutzers erheblich steigern. Problematisch daran ist, dass elektronische Spracherkennungssysteme — wenn sie fahrzeugzentriert angebracht sind — nur einzelne Kommandos verstehen können, für eine Erkennung natürlicher Sprechweise aber auf ein Headset angewiesen sind, da Umgebungsgeräusche eine Erkennung sonst stark erschweren.

Eine Möglichkeit auch ohne Headset Spracherkennung zu ermöglichen besteht darin, ein Richtmikrofon für die Schallaufnahme zu verwenden, welches Umgebungsgeräusche dämpft. Dazu muss allerdings die Sprecherposition bekannt sein, um das Richtmikrofon auf die Schallquelle steuern zu können. Darüber hinaus sind einige Kommandos nur dann korrekt umzusetzen, wenn die Richtung zum Sprecher bekannt ist. Das gilt vor allem für örtlich relative Anweisungen, wie beispielsweise “Komm her!”

Aber auch für die Orientierung eines mobilen Roboters ist die Lokalisation von Schallquellen hilfreich. Das gilt vor allem bei der Navigation durch Gebäude, in denen sich auch Menschen bewegen. Zwei sich auf dem Gang unterhaltende Personen stellen schließlich auch ein Hindernis dar.

Ziel dieser Diplomarbeit ist die Entwicklung einer Mikrofon-Anordnung nebst computergestützter Signalverarbeitung, welche in der Lage ist, eine Geräuschquelle räumlich zu orten. Außerdem soll die Mikrofonanordung als simuliertes Richtmikrofon für beliebig einstellbare Raumrichtungen verwendbar sein, um auch auf größere Entfernung elektronische Spracherkennung einsetzen zu können.

Die gesamte Anordnung soll mit leicht erhältlichen und preisgünstigen Standardkomponenten konstruiert werden, die Soft- und Hardware auf einem Linux-PC lauffähig sein.

Außerdem soll die Anordnung zur Aufnahme der Geräusche in ihrer Grundfläche nicht mehr als 200x200mm und in ihrer Höhe nicht mehr als 400mm betragen, da der Platz auf der Plattform eines mobilen Roboters stark beschränkt ist.

Zusätzlich zu den üblichen Problemen einer Schallortung, wie Echos, Hall oder multiple Geräuschquellen, kommen noch die Anforderungen, die der Betrieb auf einem mobilen Roboter mit sich bringt. Dazu zählen unter anderem:

- Toleranz gegenüber Mobilität von Sprecher und Empfänger, welche eine Adaption der Raum-Akustik praktisch unmöglich macht.
- Unempfindlichkeit gegen Störungen, die durch den Roboter selbst erzeugt werden. Das können durch Motoren, Fahrwerk, Lüfter, Festplatten oder Anschlagknacken der Ultraschallortung verursachte Geräusche ebenso, wie Spannungsschwankungen in der Versorgungsspannung des Computers sein

In Kapitel 2 wird zunächst auf zwei gebräuchliche Methoden der Schallortung und anschließend auf den in dieser Arbeit entwickelten Ansatz eingegangen. Kapitel 3 stellt den Aufbau der verwendeten Hardware dar und Kapitel 4 befasst sich mit der Implementierung des vorgestellten Algorithmus und einigen Zusatzprogrammen.

Eine Anleitung zum Abgleich und zur Inbetriebnahme, sowie die Bedienungsanleitung finden sich schließlich in Anhang A.

Kapitel 2

Methoden der Schallortung

Die zwei folgenden Abschnitte beschreiben zunächst zwei bekannte Methoden der Schallortung. Anschließend wird der im Rahmen dieser Arbeit entwickelte Ansatz vorgestellt, welcher ebenso wie die beiden anderen Verfahren auf die Ortung kontinuierlicher Schallquellen ausgelegt ist.

Da alle drei Verfahren prinzipiell bereits mit zwei Mikrofonen funktionieren, werden sie auch zunächst — ohne Beschränkung der Allgemeinheit — anhand eines Mikrofonpaars erklärt, damit Diagramme und Formeln übersichtlich bleiben. Eine Methode zur Verknüpfung von Mikrofonpaaren wird anschließend in Abschnitt 2.3.2 erläutert.

2.1. Steered Beamformer

Das Funktionsprinzip eines Steered Beamformers besteht darin, ein virtuelles Richtmikrofon solange zu schwenken, bis die Richtung des maximalen Schallleistungseinfalls gefunden ist. In dieser Richtung wird dann die Quelle angenommen.

2.1.1. Darstellung der Richtcharakteristik

In den folgenden Abschnitten werden häufig Richtcharakteristikdiagramme zur Verdeutlichung eingesetzt. Daher soll hier kurz deren Prinzip erläutert werden.

Ein Richtcharakteristikdiagramm drückt die Abhängigkeit der Empfindlichkeit eines Empfängers von der Einfallsrichtung der Welle aus. Meistens werden kreisförmige Diagramme verwendet, bei denen der Einfallswinkel direkt eingetragen werden kann. Unter diesem Winkel wird dann ein Punkt dessen Abstand zum Mittelpunkt des Diagramms der Empfindlichkeit entspricht eingezeichnet.

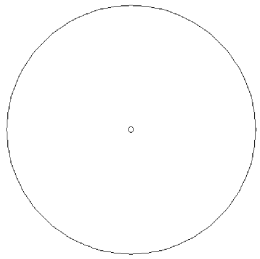


Abbildung 2.1.: Richtcharakteristik eines omnidirektionalen Empfängers

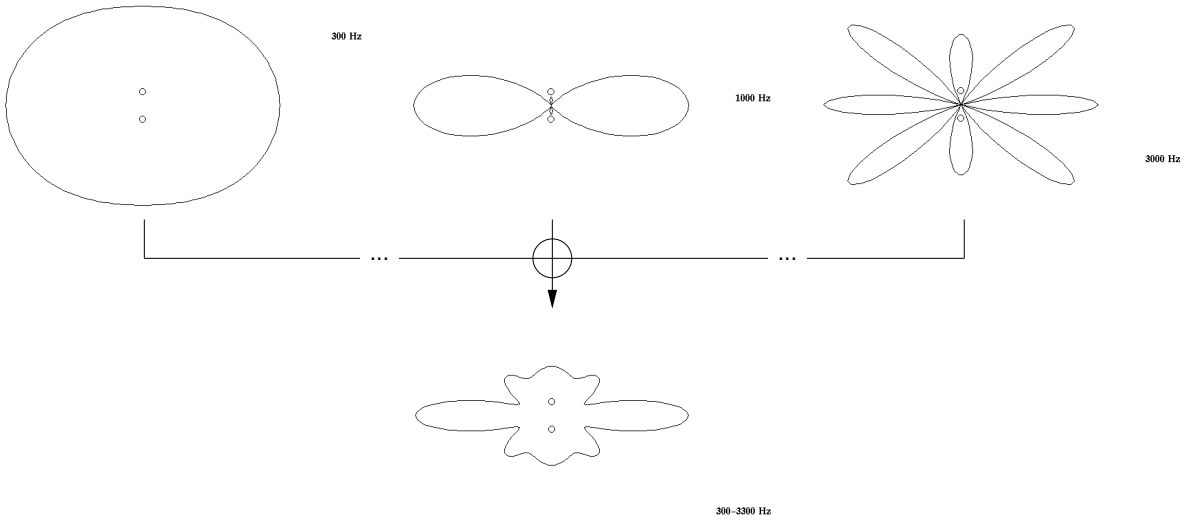


Abbildung 2.2.: Zusammenfassen der Richcharakteristik bei verschiedenen Frequenzen in ein Diagramm

Ist die Empfindlichkeit des Empfängers in alle Richtungen gleich, so zeigt das Diagramm einen Kreis (siehe Abb. 2.1). Bevorzugt er bestimmte Richtungen, so ergeben sich die bekannten “Keulen”.

Eine Abhängigkeit der Empfindlichkeit von der Einfallsrichtung wird durch konstruktive und destruktive Überlagerung der einfallenden Wellen erzeugt. Aus diesem Grund hängt die Richtcharakteristik — bei einer Empfängerkonfiguration — auch noch von der Frequenz ab.

Um nun nicht für jede Frequenz ein eigenes Diagramm erstellen zu müssen, wird in den folgenden Abschnitten meistens die Summe der Empfindlichkeiten für einen bestimmten Frequenzbereich in einem Diagramm zusammengefasst. Man kann sich das als die Richtcharakteristik des Empfängers für bandbegrenztes Rauschen des angegebenen Bereichs vorstellen. Abbildung 2.2 zeigt ein so entstandenes Diagramm.

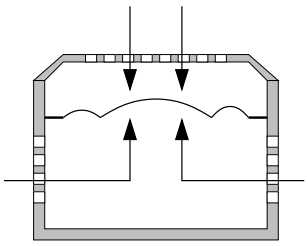


Abbildung 2.3.: Prinzip eines Richtmikrofons (Wandler nicht eingezeichnet)

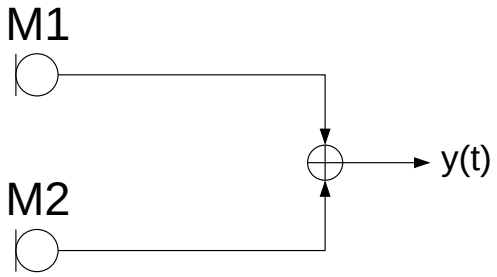


Abbildung 2.4.: Addition zweier unverzögerter Mikrofonsignale

2.1.2. Prinzip eines simulierten Richtmikrofons

Ein normales Richtmikrofon enthält üblicherweise eine Mikrofonkapsel, deren Membran über verschiedene Öffnungen im Gehäuse und entsprechend dimensionierte Verzögerungsleitungen mit Schallwellen der Umgebung versorgt wird (siehe Abb. 2.3). Diese Schallwellen überlagern sich dann konstruktiv oder destruktiv auf der Membran, sodass Schall aus bestimmten Richtungen verstärkt und aus anderen Richtungen bedämpft wird.

Das gleiche kann man natürlich auch dadurch realisieren, dass man mehrere omnidirektionale Einzelmikrofone im entsprechenden Abstand verwendet und deren Signale addiert bzw. subtrahiert.

Verwendet man zwei Mikrofone und addiert ihre Signale gemäß Abbildung 2.4, so ergibt sich eine Richtcharakteristik, welche Einfallsrichtungen senkrecht zur Verbindungsgeraden der Mikrofone bevorzugt, nämlich diejenigen, aus denen das Signal an beiden Mikrofonen gleichzeitig ankommt und sich so konstruktiv überlagert (siehe Abb. 2.5).[1]

Verzögert man das Signal von Mikrofon M1 gegenüber dem Signal von Mikrofon M2 nach Abbildung 2.6, so kann man — ohne die Mikrofone selbst zu bewegen — die Richtung maximaler Empfindlichkeit beeinflussen. Diese ist auch hier die Richtung, aus der ein Schallsignal kommen muss, damit es sich nach der Verzögerung bei der Addition konstruktiv überlagert.

[1] Die Richtcharakteristikdiagramme dieses Abschnitts sind der Übersichtlichkeit wegen zweidimensional gehalten. Die tatsächliche räumliche Empfindlichkeit kann man sich als Rotationskörper der zweidimensionalen Empfindlichkeitskurven um die Verbindungsgerade der beiden Mikrofone vorstellen.

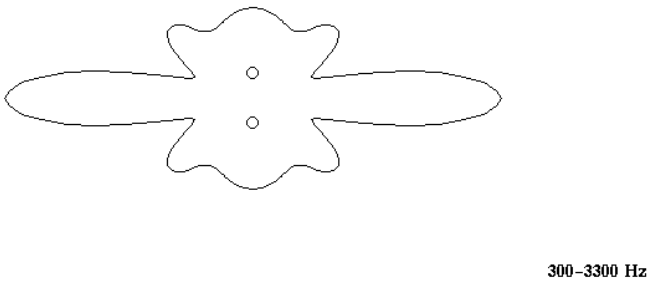


Abbildung 2.5.: Richtcharakteristik eines Mikrofonpaars, dessen Signale addiert werden, bei einem Abstand von 200mm und einem Frequenzgemisch von 300-3300 Hz

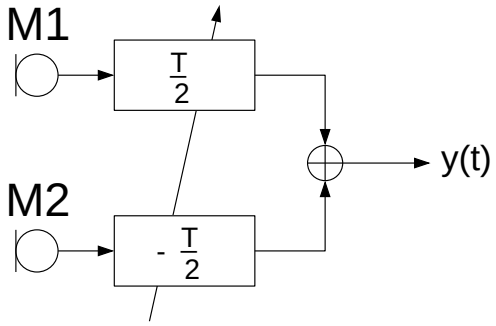


Abbildung 2.6.: Addition zweier Mikrofonsignale mit einstellbarer Signalverzögerung

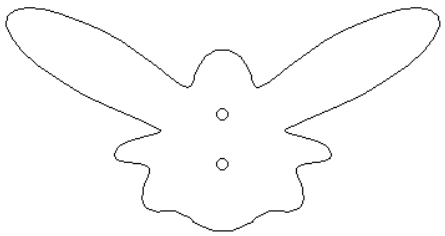


Abbildung 2.7.: Richtcharakteristik bei zwei Mikrofonen im Abstand von 200mm bei einem Frequenzgemisch von 300-3300Hz, deren Signale um 50% der Signallaufzeit gegeneinander verzögert sind.

Abbildung 2.7 zeigt die Richtcharakteristik bei einer Verzögerung des Signals von M1 gegenüber dem Signal von M2 um 50% der Schalllaufzeit für den Abstand zwischen beiden Mikrofonen. Bei Abbildung 2.8 beträgt diese Verzögerung 100%, was bei einer Schallgeschwindigkeit v_s und einem Mikrofonabstand d einer Zeit t von

$$t = \frac{d}{v_S} \tag{2.1}$$

entspricht.

Um die weiteren Berechnungen einfach zu halten, wird im folgenden die von einer Schallquelle ausgehenden Kugelwelle als ebene Welle genähert. Es soll nun gezeigt werden, dass der Fehler durch diese Näherung vernachlässigbar ist, wenn der Abstand zur Schallquelle a groß gegenüber dem Mikrofonabstand d ist.

Wie in Abbildung 2.9 dargestellt, kommt eine von der Quelle Q im Abstand a ausgesandte Kugelwelle bei den Mikrofonen M1 und M2 um die Laufzeit der Strecke ϵ später an, als eine gedachte ebene Welle.

Für den Streckenunterschied ϵ gilt:

$$\epsilon = l - a$$

und mit

$$l = \sqrt{a^2 + \frac{d^2}{4}}$$

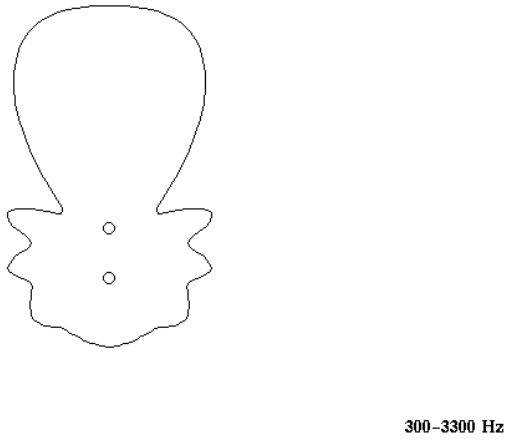


Abbildung 2.8.: Richtcharakteristik bei Verzögerung der Mikrofonsignale um 100% der Laufzeit

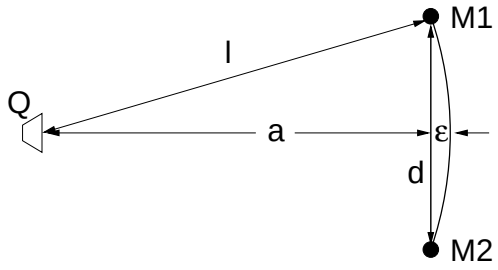


Abbildung 2.9.: Laufzeitunterschied zwischen einer Kugelwelle und einer ebenen Welle

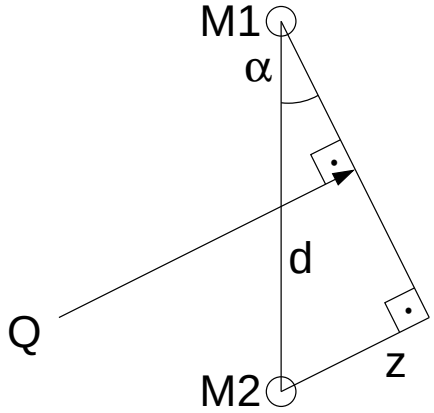


Abbildung 2.10.: Berechnung der Verzögerung in Abhängigkeit von der Schwenkrichtung

ist

$$\epsilon = \sqrt{a^2 + \frac{d^2}{4}} - a \tag{2.2}$$

Eine Entfernung zur Quelle von 1m und ein Mikrofonabstand von 0,2m ergibt ein ϵ zu 5,0mm, was bei einer Abtastrate von 48kHz weniger als die Schalllaufzeit zwischen zwei Abtastwerten ist.

Mit obiger Näherung kann nun die einzustellende Verzögerung T, welche für den Mikrofonschwenk in eine gewünschte Richtung benötigt wird nach Abbildung 2.10 berechnet werden. Dabei ist d der Abstand der Mikrofone, z der Streckenunterschied um den die Welle bei M2 früher ankommt als bei M1, v_S ist die Schallgeschwindigkeit. T ergibt sich damit zu:

$$T = \frac{z}{v_S} = \frac{d \sin \alpha}{v_S} \tag{2.3}$$

Der Winkel in Abhängigkeit der Verzögerung ist demnach:

$$\sin \alpha = \frac{T \cdot v_S}{d} \quad \Rightarrow \quad \alpha = \arcsin \frac{T \cdot v_S}{d} \tag{2.4}$$

Da die Zeitabstände T, um welche das Signal verzögert werden kann, aufgrund der Abtastung nur diskrete Werte annehmen können, ist die Winkelauflösung der Anordnung bereits durch Abtastrate T_a und Mikrofonabstand d begrenzt. Da durch die Abtastung ein Fehler von $\pm\frac{T_a}{2}$ entstehen kann, ist im günstigsten Fall — also bei kleinem $|\alpha|$ — die minimale Winkelunsicherheit $\Delta\alpha_{min}$:

$$\Delta\alpha_{min} = \arcsin\frac{\frac{T_a}{2}\cdot v_s}{d} - \arcsin\frac{-\frac{T_a}{2}\cdot v_s}{d} = 2\arcsin\frac{\frac{T_a}{2}\cdot v_s}{d} \tag{2.5}$$

Fällt die Schallwelle aus Richtung der Verbindungsgeraden zwischen den Mikrofonen ein, so führt das zu einer deutlich größeren Winkelunsicherheit $\Delta\alpha_{max}$.

$$\Delta\alpha_{max} = \pi - 2\arcsin\frac{d - \frac{T_a}{2}\cdot v_s}{d}$$

Diese Einfallsrichtung sollte nach Möglichkeit nicht für eine Ortung herangezogen werden.

2.1.3. Schallortung durch ein simuliertes Richtmikrofon

Um nun eine Richtungsbestimmung durchzuführen, werden die von den Mikrofone M1 und M2 gelieferten Signale abgetastet, sodass ein Audiodatensatz $x_{M1}[t]$ bzw. $x_{M2}[t]$ entsteht. Anschließend wird mit Gleichung 2.3 für jede mögliche Richtung α die erforderliche Verzögerung $T[\alpha]$ bestimmt und dann für jede dieser Richtungen die Energie $E[\alpha]$ über die Summe von je M Abtastwerten berechnet:

$$E[\alpha] = \sum_{t=-\frac{M}{2}}^{\frac{M}{2}} (x_{M1}[t - \frac{T[\alpha]}{2}] + x_{M2}[t + \frac{T[\alpha]}{2}])^2 \tag{2.6}$$

Die Richtung, welche die maximale Energie liefert ist die Richtung zur Schallquelle.

Nachteilig bei dieser Methode ist, dass ein solches durch mehrere Mikrofonen simuliertes Richtmikrofon bei höheren Frequenzen nicht nur einen, sondern mehrere Kegelmantel maximaler Empfindlichkeit ausbildet, wie Abbildung 2.11 zeigt. Bei niedrigen Frequenzen ist aber keine genaue Richtungsbestimmung möglich, wie aus Abbildung 2.12 ersichtlich ist.

Schwierigkeiten kann bei dieser Methode also das Spektrum der zu ortenden Schallquelle bereiten. Enthält es nur niederfrequente Anteile, so ist eine Ortung nur sehr ungenau möglich, da die Richtcharakteristik in diesem Fall kein deutliches Maximum bildet. Bei nur hohen Frequenzanteilen bildet die Richtcharakteristik zwar sehr scharfe Maxima aus, jedoch können sich Mehrdeutigkeiten ergeben. Am besten für eine Ortung geeignet ist ein möglichst “weißes” Signal, also ein Signal, dass Spektralanteile in allen Frequenzbereichen aufweist.

2.2. Laufzeitdifferenzschätzung

Eine andere Methode der Schallortung stellt die Laufzeitdifferenzschätzung dar. Dabei wird der Laufzeitunterschied der Mikrofonsignale direkt aus den Mikrofonsignalen selbst

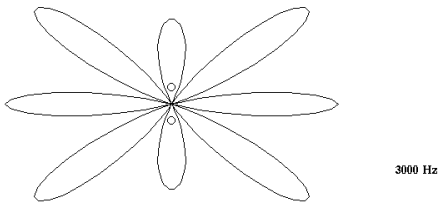


Abbildung 2.11.: Richtcharakteristik von zwei summierten Mikrofonen mit einem Abstand von 200mm bei 3kHz

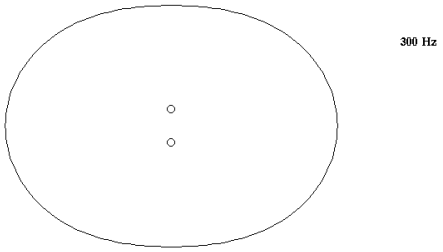


Abbildung 2.12.: Richtcharakteristik von zwei summierten Mikrofonen mit einem Abstand von 200mm bei 300Hz

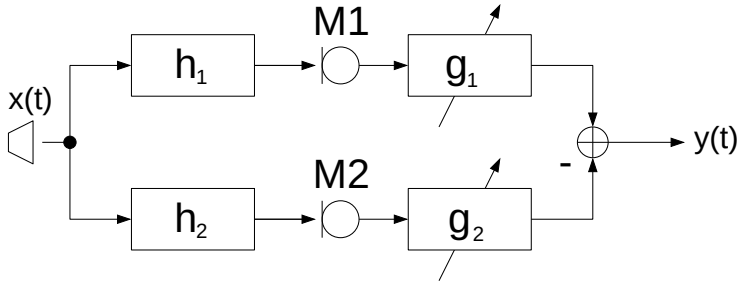


Abbildung 2.13.: Laufzeitdifferenzschätzung mit Hilfe eines adaptiven Filters

bestimmt. Hierzu werden Algorithmen wie die Kreuzkorrelation, die Phasenkreuzkorrelation — bei der nur die Phaseninformation korreliert wird — oder ein adaptives Filter verwendet.

Die Schätzung des Laufzeitunterschieds mit Hilfe eines adaptiven Filters basiert auf dem Modell, dass ein von einer Quelle ausgesandtes Signal $x(t)$ durch zwei — aufgrund des Mikrofonabstands verschiedene — Raumimpulsantworten gefaltet wird, bevor es zu den Empfängern gelangt (siehe Abb. 2.13).

Die so entstandenen Mikrofonsignale werden durch zwei diskrete Filter gefaltet und dann deren Differenz $y(t)$ gebildet. Beide Filterkoeffizientenvektoren werden dabei durch ein Gradientenabstiegsverfahren solange modifiziert, bis die Differenz $y(t)$ null ist. Es muss natürlich — beispielsweise durch Normieren des größten Koeffizienten auf eins — sichergestellt werden, dass sich nicht die triviale Lösung (beide Koeffizientenvektoren sind Nullvektoren) einstellt. Im nicht-trivialen Fall ist $y(t)$ aber nur dann null, wenn gilt (nach [1]) ($\star$ sei der Faltungsoperator):

$$y \star h_1 \star g_1 = y \star h_2 \star g_2$$

Damit ist (mit $y \neq 0$):

$$h_1 \star g_1 = h_2 \star g_2$$

Das ist der Fall, wenn:

$$h_1 = g_2 \quad und \quad h_2 = g_1$$

g_1 entspricht also am Ende des Gradientenabstiegsverfahrens der Raumimpulsantwort h_2 und g_2 entspricht h_1. Des weiteren wird bei dieser Methode angenommen, dass die jeweils größten Koeffizienten der beiden Filter dem direkten Ausbreitungspfad entsprechen, sodass über deren Abstand die Laufzeitdifferenz bestimmt werden kann.

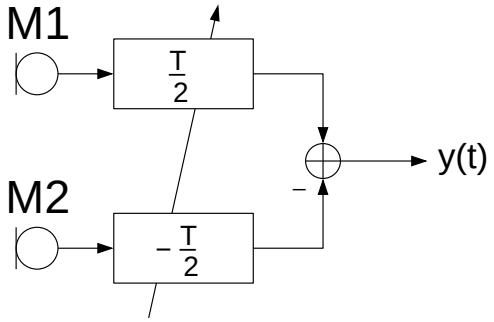


Abbildung 2.14.: Prinzip des subtraktiven Beamformers

Nachteilig bei dieser Methode ist, dass bei sich ständig ändernder Raumakustik — aufgrund der Positionsänderungen von Sender oder Empfänger — für jeden neu aufgenommenen Datensatz die komplette Iteration der Filter neu durchgeführt werden muss, was einen erheblichen Rechenzeitbedarf darstellt. In [1] wird bei der Ortung weißen Rauschens von etwa 620 Iterationsschritten ausgegangen.

Natürlich besteht bei dieser Methode — wie bei jedem Gradientenabstiegsverfahren — die Gefahr, dass die Iteration nicht das globale Minimum findet, sondern in einem lokalen einrastet, was dann sehr wahrscheinlich zu einem falschen Ergebnis führt.

2.3. Subtraktiver Steered Beamformer

Der folgende Abschnitt stellt den für diese Arbeit entwickelten Ansatz vor. Die Bezeichnung "Subtraktiver Steered Beamformer" wurde gewählt, um die Ähnlichkeit des Algorithmus zum normalen Steered Beamformer deutlich zu machen.

2.3.1. Prinzip

Betrachtet man die Methode der Laufzeitdifferenzschätzung mit Hilfe adaptiver Filter (siehe Abschnitt 2.2), so fällt auf, dass immer alle Koeffizienten berechnet werden, obwohl nur jeweils der betragsmäßig größte Filterkoeffizient pro Filter interessant ist.

Der Ansatz dieser Arbeit ersetzt daher die adaptiven Filter durch zwei Verzögerungsleitungen, wie beim Steered Beamformer (siehe Abschnitt 2.1) und verändert diese, bis die Differenz ein Betragsminimum aufweist (siehe Abb. 2.14). Auf diese Weise wird ein Raumbereich systematisch nach einer Geräuschquelle durchsucht.

Analog Gleichung 2.6 ergibt sich die Energie aus einem Datensatz in diesem Fall zu:

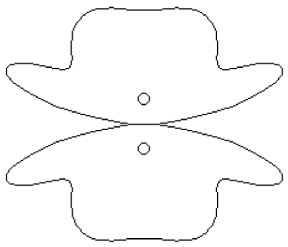


Abbildung 2.15.: Richtcharakteristik des subtraktiven Beamformers bei einem Mikrofonabstand von 200mm und einem Frequenzgemisch von 300-3300Hz

$$E[\alpha] = \sum_{t=-\frac{M}{2}}^{\frac{M}{2}} (x_{M1}[t - \frac{T[\alpha]}{2}] - x_{M2}[t + \frac{T[\alpha]}{2}])^2 \tag{2.7}$$

Alternativ könnte man anstatt der Summe der Differenzenquadrate auch die Summe der absoluten Differenzen berechnen, um bei der Implementierung Rechenzeit zu sparen:

$$E[\alpha] = \sum_{t=-\frac{M}{2}}^{\frac{M}{2}} \left| x_{M1}[t - \frac{T[\alpha]}{2}] - x_{M2}[t + \frac{T[\alpha]}{2}] \right| \tag{2.8}$$

Damit würden nicht die Differenzleistungen, sondern die Differenzamplituden berechnet, was eine etwas schlechtere Ortungsqualität zur Folge hätte (siehe Kapitel 5).

Die Methode des subtraktiven Beamformers hat gegenüber dem Steered Beamformer den Vorteil, dass das Minimum einer Differenzbildung erheblich steiler ist, als das Maximum einer Summation bei Annäherung des Suchwinkels an die tatsächliche Einfallsrichtung, wie die Richtcharakteristik in Abbildung 2.15 zeigt (vgl. Abb. 2.5).

Diese Methode bietet die Möglichkeit (wie beim Steered Beamformer auch) bei Bedarf nur ein bestimmten Bereich nach einer Schallquelle abzusuchen. Weiterhin ist von Vorteil, dass der Subtraktive Steered Beamformer immer das — für den abgesuchten Bereich — globale Minimum liefert, da alle möglichen Richtungen ausprobiert werden. Auch ist dieser Algorithmus weniger aufwändig als die Implementierung eines Gradientenabstiegsverfahrens. Außerdem benötigt der Algorithmus immer die gleiche Rechenzeit, was für Echtzeitanwendungen von großem Vorteil ist.

Anzumerken ist, dass die oben beschriebene Summe von Differenzquadraten eine starke mathematische Verwandtschaft zur Kreuzkorrelation besitzt. Gleichung 2.9 stellt eine verallgemeinerte Form von Gleichung 2.7 dar:

$$\sum_n (x[n] - x[n+a])^2 \tag{2.9}$$

$$= \sum_n (x[n]^2 - 2x[n]x[n+a] + x[n+a]^2) \tag{2.10}$$

$$= -2\sum_n x[n]x[n+a] + \sum_n x[n]^2 + \sum_n x[n+a]^2 \tag{2.11}$$

Für den Fall auf Energie normierter Korrelationsfenster werden die beiden letzten Terme jeweils zu eins. In diesem Fall ist die Summe der Differenzenquadrate bis auf diese Konstante und den Faktor von -2 zur Kreuzkorrelation

$$\sum_n x[n]x[n+a] \tag{2.12}$$

identisch.

Die sich dadurch ergebende Möglichkeit, die Berechnung auch im Frequenzbereich durchzuführen, ist allerdings für diese Arbeit nicht zweckmäßig (siehe Abschnitt 4.3).

2.3.2. Verwendung mehrerer Mikrofonpaare für die Ortung

Wie zu Beginn dieses Kapitels bereits angedeutet, beschränkten sich die bisherigen Betrachtungen auf die Verwendung von zwei Mikrofonen. Da die Ortungsqualitäten einer derartigen Anordnung — aufgrund der Mehrdeutigkeiten und der schlechten Winkelauflösung bei Schallquellen, die aus Richtung der Verbindungsgeraden der Mikrofone einfallen — für die Anforderung einer Ortung in alle Raumrichtungen bei weitem nicht ausreichen, wird im folgenden auf die Koppelung mehrerer Mikrofonpaare eingegangen.

Eine Möglichkeit besteht darin, mit jedem verwendeten Mikrofonpaar eine separate Ortung durchzuführen. Jede Einzelmessung produziert dabei — aufgrund der rotationssymmetrischen Richtcharakteristik — eine Vektorschar für die möglichen Positionen der Schallquelle in Form eines Kegelmantels (oder einer Kreisscheibe als Spezialfall) (siehe Abb. 2.16).

Um nun die tatsächliche Richtung zu bestimmen, werden die einzelnen Kegel der verschieden ausgerichteten Mikrofonpaare geschnitten, sodass nur noch die tatsächliche Richtung übrig bleibt. Für eine eindeutige Bestimmung sind mindestens zwei windschief oder drei orthogonal angeordnete Mikrofonpaare nötig, wofür mindestens vier Mikrofone erforderlich werden:

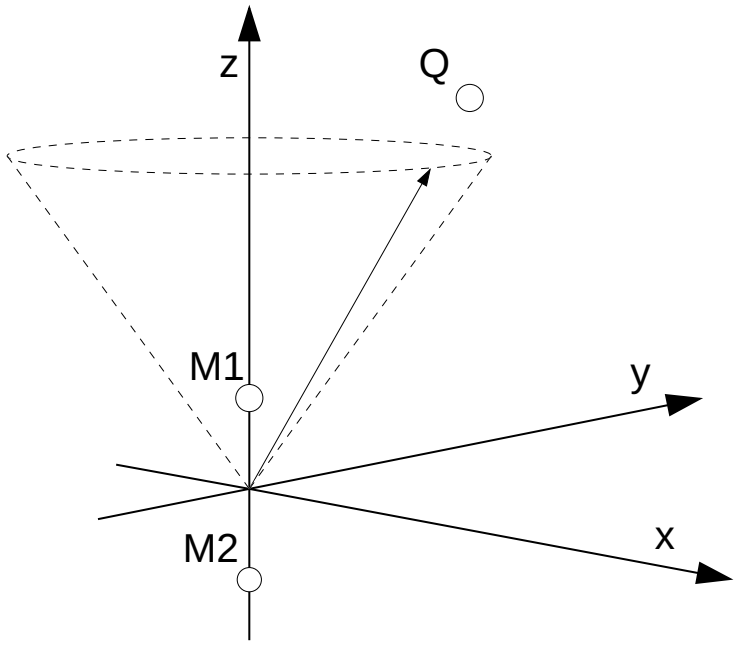


Abbildung 2.16.: Vektorfeld als Ergebnis einer räumlichen Schallortung mit zwei Mikrofonen

- Ein Mikrofon lässt keinerlei Richtungsbestimmung zu.
- Zwei Mikrofone liefern eine Richtungsbestimmung mit Mehrdeutigkeiten in Form eines Kegelmantels, da die Richtcharakteristik rotationssymmetrisch zur Verbindungsgeraden der Mikrofone ist.
- Drei Mikrofone erlauben keine Unterscheidung, von welcher Seite die Schallwelle auf die von den Mikrofonen aufgespannte Ebene trifft.

Problematisch an dieser Vorgehensweise ist allerdings, dass zum einen bei kleinen Messfehlern die Berechnung der Kegelschnitte sehr aufwendig wird und zum anderen bereits eine falsche Messung das Gesamtergebnis unbrauchbar macht.

Deshalb wird in dieser Arbeit ein anderer Ansatz realisiert. Hierbei werden ebenfalls die für eine eindeutige Richtungsbestimmung erforderlichen vier Mikrofone verwendet, aber daraus sechs Mikrofonpaare gebildet. Diese sechs Mikrofonpaare werden nun wieder zu einem Richtmikrofon zusammengefasst und dann mit diesem eine Messung ausgeführt.

Dazu wird für jedes Mikrofonpaar die für den Schwenk in eine vorgegebene Raumrichtung erforderliche Verzögerungszeit berechnet, dann für jedes Paar die Differenzleistungsbestimmung nach Gleichung 2.7 durchgeführt und die Summe dieser Leistungen gebildet.

Die Leistungssumme wird für alle abzusuchenden Raumrichtungen berechnet und diejenige Richtung bestimmt, welche das Leistungsminimum liefert. Das ist auch die wahrscheinlichste Richtung zur Schallquelle.

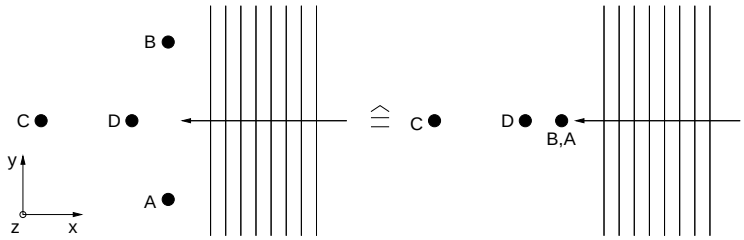


Abbildung 2.17.: Bei Schalleinfall aus x-Richtung spielt nur die x-Komponente der Koordinaten eine Rolle

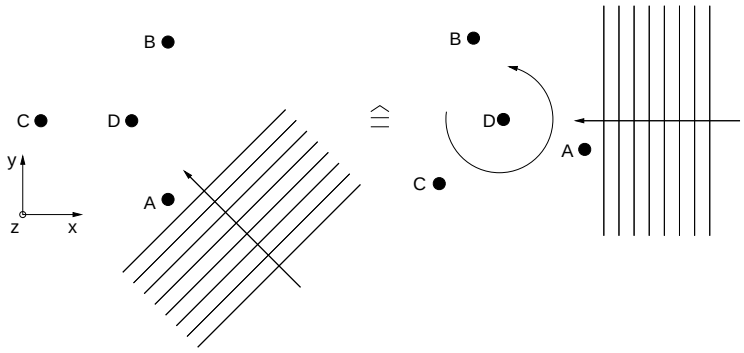


Abbildung 2.18.: Schalleinfall aus beliebigen Richtungen kann durch Rotation auf Einfall aus x-Richtung transformiert werden.

Der Vorteil bei der Verwendung von sechs Paaren anstatt vier ist, dass mehr Daten vorliegen, welche den Einfluss einer falschen Messung verringern.

Nun ist für sechs Mikrofonpaare die separate Berechnung der jeweiligen Verzögerung relativ aufwendig. Es müsste zunächst der Winkel zwischen der Verbindungsgeraden der Mikrofone und der Schallquelle bestimmt und dann für jedes Pärchen separat eine entsprechende Verzögerung eingestellt werden. Im folgenden soll daher gezeigt werden, dass es ausreichend ist, nur für jedes Einzelmikrofon eine Verzögerung zu berechnen.

Die Differenzleistung eines Mikrofonpaars ist Null, wenn die Schallwellenfront an beiden Mikrofonen nach einer Signalverzögerung gleichzeitig ankommt. Bei einer Schallwelle, die entgegen der Richtung der positiven x-Achse einfällt, ist das der Fall, wenn für die Verzögerung die Laufzeit entsprechend der x-Komponente der Mikrofonposition eingesetzt wird (siehe Abb. 2.17).

Schall der aus einer anderen Richtung einfällt, kann durch virtuelle Drehung der Mikrofonanordnung auf den gleichen Fall zurückgeführt werden (siehe Abb. 2.18).

Nachdem eine Ausrichtung in alle Raumrichtungen möglich sein soll, muss die Rotation sowohl um ϑ, als auch um φ erfolgen ($\vartheta = 0$ und $\varphi = 0$ entspreche einer Ausrichtung in positive x-Richtung), was einer Rotation zuerst um die y-, dann um die z-Achse entspricht . Zu beachten ist, dass die virtuelle Drehung der Mikrofonanordnung in die entgegengesetzte

Richtung einer Drehung zur Schallquelle erfolgen muss.

Der transformierte Koordinatenvektor ist dann (nach [8], S.317f):

$$\begin{bmatrix} x\prime \\ y\prime \\ z\prime \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\vartheta & 0 & -\sin\vartheta \\ 0 & 1 & 0 \\ \sin\vartheta & 0 & \cos\vartheta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.13}$$

Da in unserem Fall nur die x-Komponente des gedrehten Vektors von Belang ist, vereinfacht sich die Gleichung zu:

$$x\prime = \cos\varphi \cdot (x\cos\vartheta - z\sin\varphi) - y\sin\varphi \tag{2.14}$$

Es resultiert mit Gleichung 2.1 die erforderliche Signalverzögerung des n-ten Mikrofons:

$$T_n(\vartheta,\varphi) = \frac{x\prime_n}{v_S} = \frac{\cos\varphi \cdot (x_n\cos\vartheta - z_n\sin\varphi) - y_n\sin\varphi}{v_S} \tag{2.15}$$

Damit ergibt sich (mit Gleichung 2.7) für die Summe der Differenzenleistungen aller Mikrofonpaare bei vier Mikrofonen (M sei die Anzahl der zu verrechnenden Abtastwerte, x_n sind die Abtaswerte des n–ten Mikrofons, x_m die des m–ten Mikrofons):

$$E(\vartheta,\varphi) = \sum_{n=1}^{3}\sum_{m=n}^{4}\sum_{t=-\frac{M}{2}}^{\frac{M}{2}} (x_n[t+T_n(\vartheta,\varphi)] - x_m[t+T_m(\vartheta,\varphi)])^2 \tag{2.16}$$

Auch hier liefert die Suche nach dem Minimum bei einem Schwenk über das abzusuchende Gebiet die Einfallsrichtung.

2.3.3. Verwendung mehrerer Mikrofone als Richtmikrofon

Ist einmal die Richtung zum Sprecher ermittelt, so kann ein Richtmikrofon in diese Richtung geschwenkt werden, um Umgebungsgeräusche abzuschirmen und damit die Signalqualität zu verbessern.

Es bietet sich an, das Richtmikrofon durch die bereits vorhandenen Mikrofone zu realisieren. Das hat zum einen den Vorteil, dass keine bewegten Teile für den mechanischen Schwenk eines Richtmikrofons vorzusehen sind. Zum anderen wird das Signal-Rauschverhältnis im Gegensatz zu einem Mikrofon besser. Nimmt man das Mikrofon- und Verstärkerrauschen als unkorreliert an, so addieren sich bei der Summation der Mikrofonsignale nur die Rauschleistungen (siehe auch [6]). Beim Nutzsignal hingegen addieren sich die Amplituden.

Es ergibt sich also eine Verbesserung der Signal-Rauschleistung um den Faktor G zu (n sei die Anzahl der Mikrofone):

$$G = \frac{n^2}{n} = n \tag{2.17}$$

Für die Bildung eines Richtmikrofons aus omnidirektionalen Mikrofonen kommen analog zum Steered Beamformer (Abschnitt 2.1) und zum subtraktiven Steered Beamformer (Abschnitt 2.3) prinzipiell zwei Möglichkeiten in Frage:

- Eine subtraktive Verrechnung der Mikrofonsignale. Dabei werden die Signale verzögert und teils positiv teils negativ zum Gesamtsignal verrechnet, sodass sich eine Richtcharakteristik ausbildet, welche eine gewünschte Richtung bevorzugt und Schalleinfall aus anderen dämpft. Dabei nimmt man aber in Kauf, dass die Bandbreite des Signals beschnitten wird, weil tiefe Frequenzen an allen Mikrofonen ohne nennenswerte Phasenverschiebung ankommen und daher bei der Subtraktion eliminiert werden.
- Eine additive Verrechnung. Hier werden die Mikrofonsignale so verzögert und addiert, dass sich einfallende Schallwellen aus der gewünschten Richtung nach der Verzögerung konstruktiv überlagern. Das hat zwar den Vorteil, dass Geräusche aus der Vorzugsrichtung mit voller Bandbreite wiedergegeben werden, aber auch den Nachteil, dass die Dämpfung der anderen Einfallsrichtungen weniger stark ist.

Da das Richtmikrofon für eine nachgeschaltete Sprachverarbeitung vorgesehen ist, kann nur die additive Verrechnung verwendet werden, da nicht abzusehen ist, welchen Einfluss eine Bandbegrenzung des Signals auf die Erkennungsrate einer beliebigen elektronischen Spracherkennung hat.

Die Verzögerungen für die Mikrofonsignale sind so einzustellen, dass sich Signale aus der gewünschten Richtung nach der Verzögerung bei der Addition konstruktiv überlagern. Diese Bedingung wurde bereits in Abschnitt 2.3.2 ausführlich behandelt. Die erforderlichen Verzögerungen ergeben sich daher nach Gleichung 2.15.

Abbildung 2.19 zeigt schließlich das aus der Verwendung von vier Mikrofonen resultierende Richtcharakteristikdiagramm.

2.3.4. Realisierung

Für die praktische Realisierung der Schallortung waren aufgrund der beengten Platzverhältnisse wie sie auf der Plattform mobiler Roboter herrschen Gesamtabmessungen kleiner als 200 x 200 x 400mm gefordert, was nur einen maximalen Mikrofonabstand d von 200mm erlaubte.

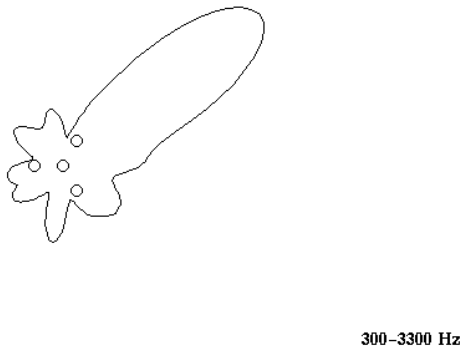


Abbildung 2.19.: Richtcharakteristik eines mit vier Mikrofonen aufgebauten Richtmikrofons bei einem Frequenzgemisch von 300-3300Hz

Bei einer Abtastrate T_a von 48kHz — was die maximale Abtastrate einer handelsüblichen Soundkarte darstellt — und einem Mikrofonabstand von 200mm bedeutet das nach Gleichung 2.5 eine Winkelunsicherheit $\Delta\alpha_m in$ im besten Fall von 2,0°.

Aufgrund der Forderung nach einer Low-Cost-Lösung und der Lauffähigkeit auf einem Linux-System kamen nur handelsübliche Standardsoundkarten in Frage, für die ein Treiber unter Linux zur Verfügung stand. Zum Zeitpunkt der Diplomarbeit waren keine Soundkarten mit mehr als zwei gleichzeitig zu benutzenden Eingängen erhältlich, welche diese Forderungen erfüllten. Das bedeutete, dass für je zwei Mikrofone eine Soundkarte zu verwenden war. Um nun Steckplätze und Rechenzeit zu sparen, wurde für die praktische Realisierung eine Beschränkung der Mikrofonanzahl auf das für eine räumliche Ortung erforderliche Minimum von vier Mikrofonen gewählt. Diese Anzahl hat darüber hinaus auch den Vorteil, dass sich bei einer Anordnung an Tetraederecken sechs Pärchen mit gleichem Abstand bilden lassen, welche eine möglichst gleichmäßige Empfindlichkeit in alle Raumrichtungen ermöglicht, da die jeweilige Richtcharakteristik aller sechs möglichen Mikrofonpaare gleich ist. Bei dieser Anzahl an Mikrofonen ergibt sich — beim Betrieb als Richtmikrofon — nach Gleichung 2.17 bereits eine Verbesserung des Signal-Rauschverhältnisses im Gegensatz zum Einsatz nur eines Mikrofons um den Faktor 4, also 6dB.

Kapitel 3

Aufbau der Hardware

Das folgende Kapitel beschreibt den praktischen Aufbau der unter Abschnitt 2.3.4 geforderten Mikrofonpositionierung, eine Schwingungsdämpfung zur Unterdrückung von Vibrationen vom Roboter, die verwendeten Mikrofone nebst Vorverstärkern, den Line-Verstärker zum Anschluss an die Soundkarten und schließlich die Soundkarten selbst zusammen mit der Schaltung zu deren Synchronisation.

3.1. Mikrofonträger

Um die geforderte Positionierung von vier Mikrofonen mit einem paarweisen Abstand von 200mm zu erreichen, wurde eine Anordnung nach Abbildung 3.1 aufgebaut, bei der die Mikrofone an den Ecken eines Tetraeders zu liegen kommen. Als Standfuß findet eine 100x100x5mm große und 530g schwere Messingplatte Verwendung, die mit vier Löchern für die Montage von Schwingungsdämpfern versehen ist.

Es ergeben sich folgende Abmessungen:

Die Gesamtlänge der Ausleger l beträgt 115mm, wobei die letzten 5mm durch die Vorverstärkerplatine gebildet werden. Die Länge des Hauptstabs von den Auslegern Z bis zur Spitze D ist 163mm, wobei auch hier wieder die letzten 5mm durch die Mikrofonplatine gebildet werden. Der Abstand der Ausleger von der Bodenplatte ist so gewählt, dass eine gute Beschallung aller Mikrofone erreicht wird. Er beträgt 180mm. Die Gesamthöhe der Anordnung beträgt damit 350mm (ohne Dämpfer).

Die Mikrofonvorverstärker sind an den Enden von Auslegern und Hauptstab so befestigt, dass alle Mikrofone nach oben zeigen (siehe Abbildung 3.3).

Ohne Dämpfungsmaßnahmen können beim Betrieb auf mobilen Robotern Vibrationen die Signalqualität negativ beeinflussen. Körperschall der bis zum Mikrofongehäuse übertragen wird, erzeugt durch die Trägheit der Membranmasse — welche einer Bewegung des

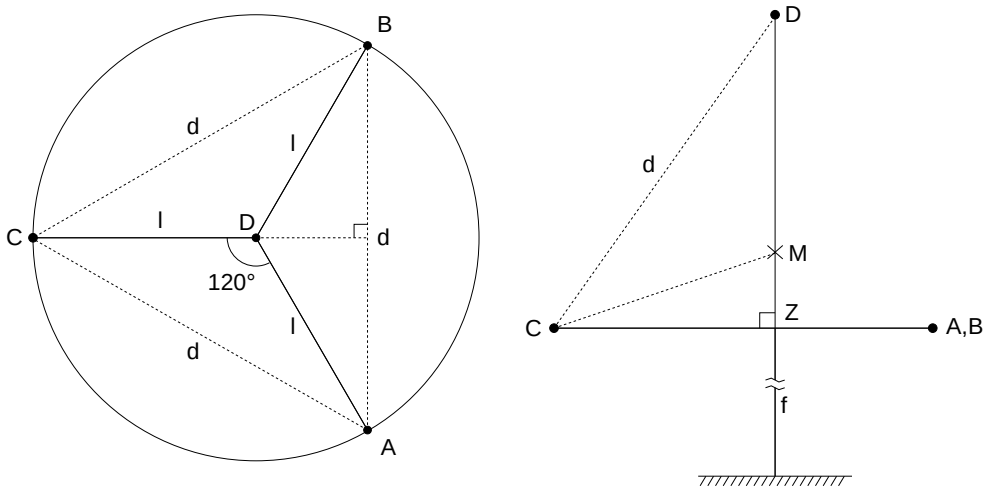


Abbildung 3.1.: Tetraederförmige Anordnung der Einzelmikrofone

Abbildung 3.2.: Praktische Realisierung des Mikrofonständers

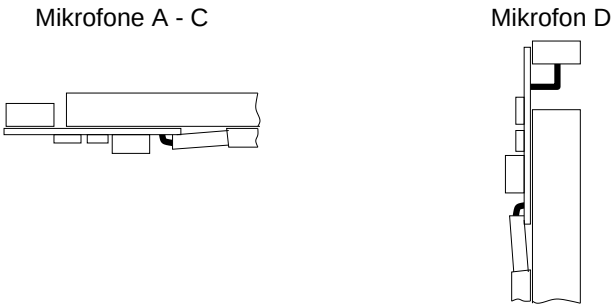


Abbildung 3.3.: Ausrichtung der Mikrofone

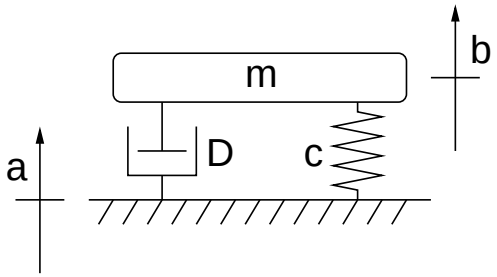


Abbildung 3.4.: Masse-Feder-Tiefpass

Gehäuses nicht schnell genug folgen kann — ein Störsignal. Aus diesem Grund dient die Grundplatte zusammen mit vier Gummidämpfern als Masse-Feder-Tiefpass für Körperschall, der die störenden höheren Frequenzanteile herausfiltert (siehe [7] Seite 541f). Niedrigere Frequenzanteile haben geringeren Einfluss auf die Mikrofone (siehe Abschnitt 3.2), weswegen auf eine Dämpfung dieses Anteils verzichtet wird. Abbildung 3.4 verdeutlicht das Prinzip.

F_D sei die Dämpferkraft, F_F die Federkraft und F_M die Kraft, welche die Masse beschleunigt. Es ergibt sich:

$$F_M = F_D + F_F \tag{3.1}$$

mit:

$$F_F = (a - b) \cdot c \tag{3.2}$$

$$F_D = \frac{d}{dt}(a - b) \cdot D \tag{3.3}$$

$$F_M = m \cdot \frac{d^2}{dt^2} b \tag{3.4}$$

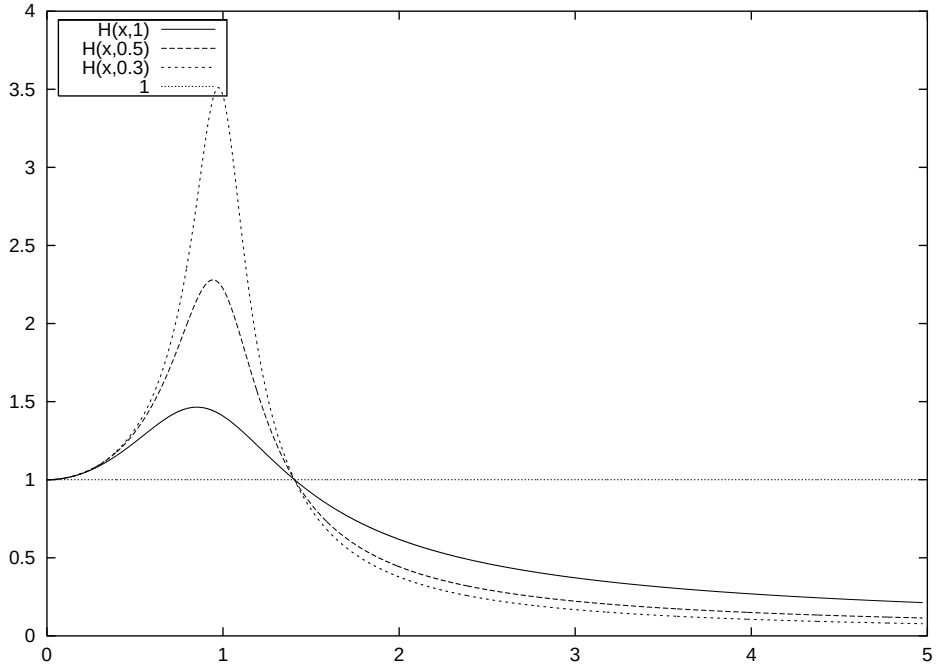


Abbildung 3.5.: Übertragungsfunktion eines Masse-Feder-Tiefpasses bei verschiedenen Dämpfungen

Unter der Annahme des eingeschwungenen Zustands und sinusförmiger Anregung kann zur Bestimmung der Übertragungsfunktion die komplexe Wechselstromrechnung verwendet werden. Setzt man dazu noch die Gleichungen 3.2, 3.3 und 3.4 in Gleichung 3.1 ein, so ist

$$m\omega^2 B = -j\omega DA + j\omega DB - Ac + Bc$$

Damit ergibt sich die Übertragungsfunktion $H(\omega)$ zu

$$H = \frac{B}{A} = \frac{j\omega D - c}{j\omega D + c - m\omega^2} \tag{3.5}$$

Unter der Annahme geringer Dämpfung liegt die Resonanzfrequenz der Anordnung bei:

$$\omega_0 = \sqrt{\frac{c}{m}} \tag{3.6}$$

Abbildung 3.5 zeigt den Verlauf der Übertragungsfunktion abhängig von der Dämpfung.

Es ist ersichtlich, dass eine Körperschalldämpfung erst bei Frequenzen oberhalb des $\sqrt{2}$-fachen der Resonanzfrequenz eintritt. In der Praxis wählt man daher die Resonanzfrequenz so, dass die niedrigste zu unterdrückende Schwingungsfrequenz um den Faktor vier darüber liegt.

Für die Masse-Feder-Körperschalldämpfung im Mikrofonfuß werden vier parallelgeschaltete Gummidämpfer (siehe Anhang C.2) verwendet.

Jeder dieser Dämpfer weist im Arbeitspunkt eine Federhärte von $140\frac{N}{cm}$ auf, sodass sich mit der Masse der Messingplatte von 0,53kg im Grundmodus (alle Dämpfer schwingen phasengleich) eine Resonanzfrequenz von

$$f_0 = \frac{1}{2\pi}\sqrt{\frac{c}{m}} = \frac{1}{2\pi}\sqrt{\frac{4 \cdot 14000\frac{N}{m}}{0.53kg}} = 51,7Hz$$

ergibt. Alle Frequenzanteile über etwa 200Hz werden also bereits stark bedämpft. Niedrigere Frequenzanteile haben aufgrund der geringen Größe der verwendeten Mikrofone kaum einen Einfluss (siehe Abschnitt 3.2).

Darüber hinaus dienen die Dämpfer auch zur Montage des Mikrofonträgers. Dazu wurden in die Deckplatte des Roboters vier Löcher gebohrt und mit M4 Gewinden versehen, in welche die Dämpfer geschraubt wurden. Darauf wurde nun die Grundplatte des Mikrofonträgers mit ihren vier Löchern (∅ 4,2mm) gesetzt und mit Hilfe von Beilagscheiben und Muttern befestigt.

3.2. Mikrofone

Die verwendeten Mikrofone sind druckempfangende Elektret-Kondensatormikrofone (siehe Anhang C.1). Sie weisen folgende Eigenschaften auf, die für eine Schallortung vorteilhaft sind.

- Geringe Größe
  Zur Vermeidung von Beugungseffekten, müssen die verwendeten Mikrofone klein gegenüber der Wellenlänge sein. Die Schallwellenlänge in Luft (bei Normalbedingungen) bei Frequenzen von 5kHz (obere Grenze des vokalen Sprachspektrums [9]) beträgt ca. 66mm; die Mikrofone haben einen Durchmesser von 5 und eine Höhe von 2,7mm, sodass diese Forderung erfüllt wird.
- Keine Richtwirkung
  Da der Schalleinfall bei der verwendeten Mikrofonanordnung (siehe Abschnitt 3.1) aus nahezu beliebigen Richtungen erfolgen kann, ist es erforderlich, dass die Mikrofone eine omnidirektionale Richtcharakteristik aufweisen, um gleiche Signalpegel auch bei verschiedenen Einfallsrichtungen zu garantieren. Ungleiche Signalpegel würden beim verwendeten Ortungsalgoritmus zu einem unscharfen Minimum und somit zu schlechteren Ergebnissen führen. Für den Fall, dass die Empfindlichkeit der als omnidirektional spezifizierten Mikrofone trotzdem nicht völlig unabhängig von der Einfallsrichtung ist, sind die vier Mikrofonkapseln gleich orientiert. Damit ist der Ein-

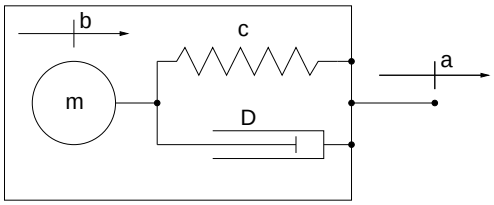


Abbildung 3.6.: Mechanisches Mikrofonersatzschaltbild

fallswinkel der Schallwellenfronten für alle vier Mikrofone bei einer als unendlich weit entfernt angenommenen Schallquelle (siehe Abschnitt 2.1.2) gleich .

- Körperschallunempfindlichkeit
  Der Einsatzbereich der Schallortung auf mobilen Robotersystemen bringt es mit sich, dass Vibrationen durch das Gehäuse und den Mikrofonträger auch zu den Mikrofonen geleitet werden. Höherfrequente Vibrationen (oberhalb 200Hz) werden bereits durch den Masse-Feder-Tiefpass im Mikrofonträger (siehe Abschnitt 3.1) unterdrückt. Tiefere Frequenzen führen aufgrund der niedrigen Membranmasse zu keinen nennenswerten Störungen. Abbildung 3.6 zeigt das mechanische Ersatzschaltbild eines Mikrofons, dabei ist m die Membranmasse, c die Membransteifigkeit und D die Dämpfung durch den Wandler. Es zeigt sich, dass Mikrofone mit niedriger Membranmasse weniger körperschallempfindlich sind (siehe [11]).

- Gutes Signal-Rauschverhältnis
  Das Signal-Rauschverhältnis der Mikrofone beträgt 38dB. Gemäß DIN 45590 bedeutet dies, dass bei einer Beschallung des Mikrofons mit einem Sinuston mit einer Frequenz von 1kHz und einem Schalldruck von 94dB eine Ausgangsspannung erzeugt wird, welche 38dB über der Ausgangsspannung des unbeschallten Mikrofons liegt. Das ist zwar keine Studioqualität (60dB SNR und mehr), stellt aber für ein Mikrofon dieser Größe einen guten Wert dar. Dieser Wert wird durch die Kombination der Mikrofone noch verbessert (siehe Abschnitt 2.1.2).

- Große Bandbreite
  Um die Anwendungsmöglichkeiten der Anordnung nicht von vornherein zu beschränken ist es erforderlich, dass die Mikrofone sowohl für Sprache, als auch für Musik oder andere Geräusche geeignet sind. Die hier verwendeten Typen haben einen sehr großen Dynamikbereich von 50-20000Hz, der auch Musikaufnahmen zulässt.

3.3. Vorverstärker

Eine sehr wichtige Rolle bei der Verarbeitung von Audiosignalen kommt den Signalverstärkern zu. Die verwendeten Mikrofone haben eine Empfindlichkeit von -65dB (0dB $\equiv$ 1V/μbar), was einer Ausgangsspannung von -69dBV (15,8μV) bei einem Schallpegel von

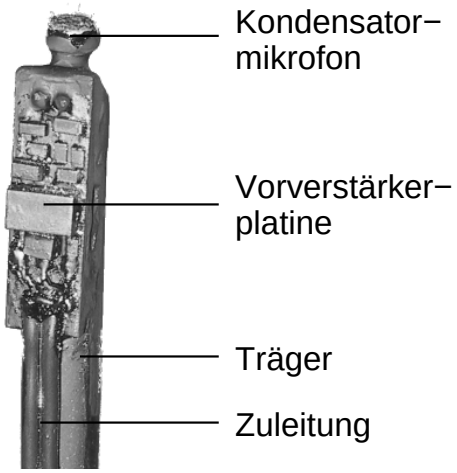


Abbildung 3.7.: Montierter Mikrofonvorverstärker

70dB (0dB $\equiv 20\mu$Pa) (laute Zimmerlautstärke) entspricht.

Will man dieses Mikrofonsignal auf den Pegel für einen Line-Eingang von 1 V_{SS} bringen, so ist eine Gesamtverstärkung von 69dB, also eine Spannungsverstärkung um den Faktor 2800 erforderlich.

Für ein gutes Signal/Rauschverhältnis des verstärkten Signals ist die Rauscharmut vor allem der ersten Verstärkerstufe von entscheidender Bedeutung, da ein hier zugesetztes Rauschsignal von allen folgenden Verstärkerstufen mitverstärkt wird.

Genauso können auch kleine elektromagnetische Einstreuungen auf dem Kabel vom Mikrofon zum Verstärker — beispielsweise durch nicht funkentstörte Geräte — schnell ein schwaches Signal unbrauchbar machen. Wird aber ein bereits verstärktes Signal übertragen, so haben Störungen weniger Einfluss. Aus diesem Grund wurde für jedes Mikrofon ein rauscharmer Vorverstärker vorgesehen, der in nächster Nähe zum Mikrofon angebracht ist (siehe Abb. 3.7).

Diese Vorverstärker erreichen mit nur einem Transistor eine Verstärkung von ca. 52dB und übernehmen außerdem die Stromversorgung der Mikrofonkapseln (siehe Abb. 3.8).

Die Stromversorgung des Verstärkers mit 5V erfolgt über die Anschlüsse Pin3 und Pin5. Diese bereits gut stabilisierte Betriebsspannung (siehe Abschnitt 3.4) wird zunächst durch einen Tiefpass gebildet aus R2 und C3 von letzten Störungen im Bereich über 100Hz befreit und anschließend über R1 dem internen Verstärker der Mikrofonkapsel zugeführt. Dieser Widerstand bildet zusammen mit dem internen Komponenten der Kapsel (Ersatzschaltbild der Kapsel siehe Abb. 3.9) einen A-Verstärker. Das Audiosignal wird über C1 ausgekoppelt und dem Verstärkertransistor zugeführt ([2], Seite 1986). Der Basisruhestrom wird durch R4 und R5 so eingestellt, dass sich am Kollektor die halbe Betriebsspannung $\frac{U_b}{2}$ einstellt, der Transistor also maximal ausgesteuert werden kann. Bei einer angenom-

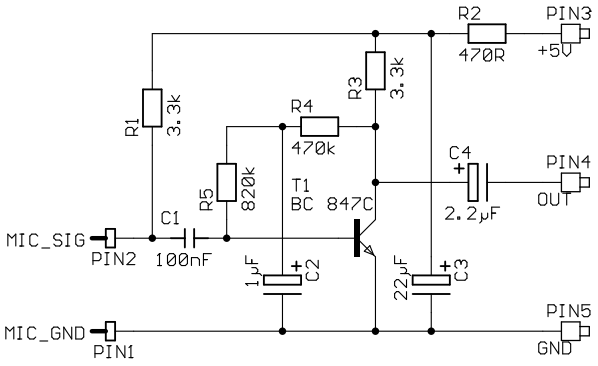


Abbildung 3.8.: Plan des Vorverstärkers

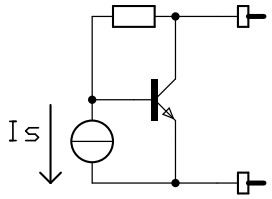


Abbildung 3.9.: Ersatzschaltbild des Kondensatormikrofons

menen Stromverstärkung von 420 (aufgrund der Serienstreuung der Transistoren kann die tatsächliche Stromverstärkung beim BC 847C zwischen 420 und 800 liegen) ist dazu für einen Kollektorstrom

$$I_C = \frac{\frac{U_b}{2}}{R_9}$$

ein Basisstrom von

$$I_B = \frac{I_C}{420}$$

erforderlich. Dieser wird durch die Gegenkopplungswiderstände R4 und R5 (C2 sei aufgrund des statischen Falles hier nicht berücksichtigt) bei Vernachlässigung des Spannungsabfalls in der Basis-Emitter-Diode zu

$$I_B = \frac{\frac{U_b}{2}}{R_4 + R_5}$$

eingestellt. Daraus ergibt sich für $R_4 + R_5$:

$$R_4 + R_5 = 420 \cdot R_9 = 1,39 M\Omega$$

C2 bildet mit R4 zusammen einen Tiefpass, der zum einen die Gegenkopplung tiefpassfiltert, um die maximale Transistorverstärkung zu erreichen und zum anderen verhindert,

dass Verstärkungsrauschen wieder der Basis zugeführt wird. Die Grenzfrequenz ist hierbei auf 2 Hz angesetzt. Damit ergeben sich mit C1 = 1μF für R4 ein Wert von 500k (470k) und für R5 890k (820k).

Das nun maximal verstärkte Signal wird über C4 ausgekoppelt, der zusammen mit der Eingangsimpedanz des Line-Verstärkers (siehe Abschnitt 3.4) zu einer unteren Grenzfrequenz des Vorverstärkers von 25Hz führt.

Um die hohe Verstärkung von mindestens 52dB mit nur einem Transistor zu erreichen, wird eine schwache Gegenkopplung verwendet. Diese hat jedoch zur Folge, dass der Verstärkungsfaktor der Schaltung stark von der Serienstreuung der Transistoren abhängt. Der daher erforderliche Abgleich wird zusammen mit dem für die Mikrofone notwendigen — die Mikrofonekapseln weisen ihrerseits bereits eine Serienstreuung der Empfindlichkeit von ±4dB auf — im Line-Verstärker (siehe Abschnitt 3.4) durchgeführt.

3.4. Line-Verstärker

Der Line-Verstärker hat die Aufgabe, die von den Vorverstärkern kommenden Signale von ca. 0.14 V_{SS} (-15.3dB) bei 70dB Schalldruck auf den standardisierten Line-Pegel von $1V_{SS}$ zu bringen, die Unterschiede in den Verstärkungsfaktoren der Mikrofone und der Vorverstärker auszugleichen und bei Bedarf ein Synchronisationssignal einzuspeisen, das die Messung eines Versatzes sowohl zwischen den beiden Soundkarten, als auch zwischen dem jeweils linken und rechten Kanal ermöglicht.

Abbildung 3.10 zeigt den Schaltplan.

Als Stromversorgung dienen gut geglättete 5V (siehe Abschnitt 3.5). Abgesichert durch die Sicherung F1 und mit einer Betriebsanzeige (R1, D1) versehen wird die Versorgungsspannung zunächst einem DC-DC-Wandler (siehe Anhang C.5) zugeführt, der daraus symmetrische ±5V generiert und eine Potenzialtrennung vornimmt. Letztere ist wichtig, da sonst Kriechströme über die Massen der Audioleitungen fließen würden, was Störsignale zur Folge hätte.

Die symmetrische Spannung wird nun noch durch Elektrolyt- und keramische Kondensatoren von nieder- und hochfrequenten Störungen befreit und dann den Gattern, den Operationsverstärkern und den 3,5mm Klinkenbuchsen zum Anschluss der Mikrofonvorverstärker zur Verfügung gestellt. Anzumerken ist, dass die Spitzen der Klinkenstecker die positive Stromversorgung erhalten, der Mittelanschluss das Signal liefert und der Masseanschluss die Audio- und Stromversorgungsmasse trägt. Diese Art der Beschaltung verhindert Kurzschlüsse beim Ein- und Ausstecken, sodass auch ein Umstecken bei angelegter Betriebsspannung (Hot Plugging) erlaubt ist.

Die eigentliche Verstärkung wird durch die vier Operationsverstärker des TL074 (rauscharmer Operationsverstärker für Audio-Anwendungen, siehe Anhang C.4) durchgeführt. Ab-

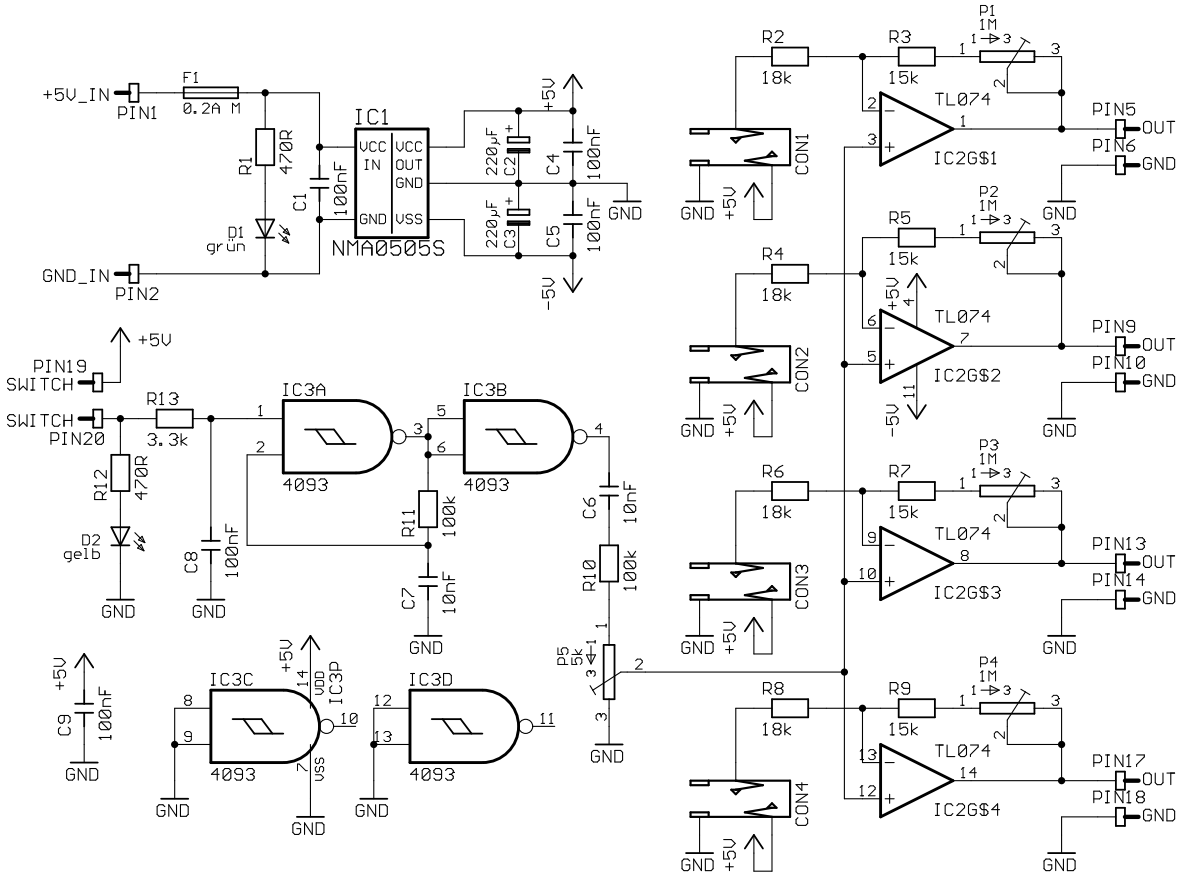


Abbildung 3.10.: Schaltplan des Line-Verstärkers

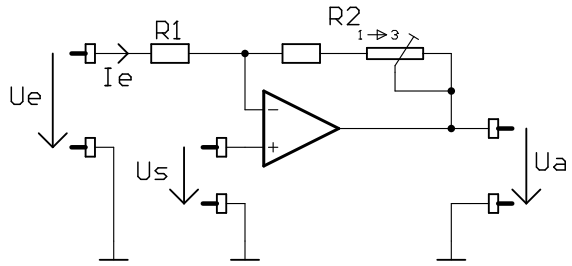


Abbildung 3.11.: Prinzip des Line-Verstärkers

bildung 3.11 zeigt das Prinzip.

U_e ist die Eingangsspannung, I_e der Eingangsstrom, U_a Ausgangsspannung und U_s das bei Bedarf eingespeiste Synchronisationssignal. Mit der idealisierten Annahme, die Differenzspannung zwischen den Operationsverstärkereingängen sei Null und es fließe kein Strom in die Eingänge, ergibt sich für die Ausgangsspannung:

$$U_a = U_e - I_e R_1 - I_e R_2$$

und mit

$$I_e = \frac{U_e - U_s}{R_1}$$

ist

$$U_a = U_s(1 + \frac{R_2}{R_1}) - U_e \frac{R_2}{R_1}$$

Da R2 zwischen 15kΩ und 1MΩ eingestellt werden kann, beträgt die mögliche Spannungsverstärkung zwischen 0,83 (-1.4dB) und 56,3 (35dB), sodass sich zusammen mit den Vorverstärkern eine Gesamtverstärkung zwischen 51dB und 87dB ergibt. Diese Abgleichmöglichkeit reicht auch aus, um die Serienstreuung der Transistoren (±3dB) und die der Mikrofone (±4dB) auszugleichen.

Das Synchronisationssignal wird von IC3 erzeugt. Der verwendete CD 4093 enthält vier NAND-Gatter mit Schmitt-Trigger-Verhalten. Der Oszillator kann über einen Schalter (Pins SWITCH) aktiviert werden, was dann durch das Leuchten von D2 angezeigt wird. Das Schaltsignal wird durch R13 und C8 von Störspitzen befreit und einem der beiden NAND-Eingänge zugeführt. Dadurch geht der Ausgang des Gatters auf LOW, C7 — dessen Spannung am anderen Eingang anliegt — wird über R11 entladen, bis die Triggerschwelle erreicht ist. Danach geht der Ausgang auf HIGH und lädt den Kondensator bis zur oberen Triggerschwelle, ab welcher erneut entladen wird.

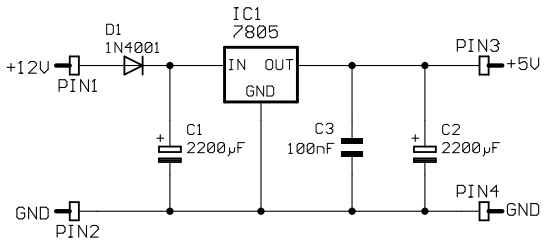


Abbildung 3.12.: 5-Volt-Versorgung

Das so entstehende Rechtecksignal (mit unsymmetrischem Tastverhältnis, da der Umschaltpunkt nicht in der Mitte der Versorgungsspannung liegt) wird durch ein weiteres Gatter geleitet, um Rückwirkungen auf den Oszillator zu verhindern. Anschließend wird es durch C6 gleichstromfrei gemacht, über einen einstellbaren Spannungsteiler auf die gewünschte Amplitude gebracht und mit Hilfe der Operationsverstärker auf das Audiosignal aufmoduliert. Es ist von Vorteil den gesamten Oszillator ein- und auszuschalten anstatt nur das Signal, da ein laufender Oszillator immer über die Versorgungsspannung in andere Schaltungsteile einstreuen würde, selbst wenn sein Ausgang abgetrennt ist.

Zu beachten ist, dass aufgrund der Operationsverstärkerschaltung das Einspeisen des Testsignals nur dann funktioniert, wenn die Mikrofone angeschlossen sind.

3.5. Stromversorgung

Als Stromversorgung für den Line-Verstärker (siehe Abschitt 3.4) sind 5V vorgesehen, wie sie auch einem Computernetzteil entnommen werden können. Häufig ist diese 5V Versorgung aber mit starken Störspitzen in allen Frequenzbereichen durchsetzt, sodass ein Betrieb des Verstärkers daran zu einer Verschlechterung der Signalqualität führt. Das war auch bei der vorliegenden Arbeit der Fall. Daher musste die 5V-Versorgung mit einer gesonderten Spannungsstabilisierungs- und -glättungsschaltung aus den 12 Volt des Computernetzteils erzeugt werden. Abbildung 3.12 zeigt die verwendete Schaltung.

Dabei laden die 12V des Computernetzteils über eine Diode den hochkapazitiven Elektrolytkondensator C1 auf, der eine Glättung der Spannung durchführt. Die Diode verhindert bei Spannungseinbrüchen der 12Volt-Versorgung, dass Ladung zurück in das Netzteil fließt, was den Glättungeffekt zunichte machen würde.

Die so gesiebten 12V werden dann durch den Spannungsregler IC1 auf 5 Volt stabilisiert. C3 verhindert ein Anschwingen der Schaltung und C2 verbessert nochmals das Regelverhalten von IC1.

Diese Schaltung darf *nicht* während des laufenden Betriebs eingesteckt werden, da dann durch das plötzliche Aufladen von C1 und dem damit verbundenen hohen Stromfluss ein

Spannungseinbruch in der Computerstromversorgung erzeugt würde, der zu Betriebsstörungen, Datenverlust oder gar einer Notabschaltung des Computernetzteils führen könnte.

3.6. Soundkarten

Um die Audiodaten in den PC einzulesen, wird für jedes Mikrofon ein AD-Wandler mit einer Abtastrate von 48kSamples/s bei 16 Bit Auflösung benötigt (siehe Abschnitt 2.3.4). Um der Forderung nach einer Low-Cost-Lösung gerecht zu werden, teure Spezialkomponenten zu vermeiden und eine Treiberunterstützung unter Linux zu gewährleisten, wird dabei auf handelsübliche Soundkarten zurückgegriffen.

Preisgünstige Vierkanalkarten, für die unter Linux Treiber zur Verfügung stehen, waren zum Zeitpunkt der Arbeit nicht verfügbar, sodass zwei Stereo-Soundkarten (Creative Labs Soundblaster PCI 128 mit CT 5110-Chipsatz) Verwendung fanden.

Das gleichzeitige Einlesen von zwei unabhängigen Soundkarten ist zwar unter Linux mit Hilfe des ALSA-Treibers [3] möglich, jedoch weisen die Quarze der Karten normalerweise einen leichten Frequenzunterschied auf. Im vorliegenden Fall betrug dieser — gemessen mit Hilfe des Sychronsignalgenerators (siehe Abschnitt 3.4) und der Synchronitätsmessung (siehe Abschnitt 4.8.3) — ca. 0.0016%. Diese nur scheinbar sehr geringe Abweichung führte dazu, dass die Signale der beiden Karten trotz synchronen Starts innerhalb von 100s bei einer Abtastrate von 48kHz um knapp 77 Abtastwerte gegeneinander versetzt waren. (Ein 180°-Schwenk der Geräuschquelle bei dem verwendeten Mikrofonabstand von 200mm erzeugt nur einen Versatz von 58 Abtastwerten).

Ein häufiger Neustart der Karten ist nicht akzeptabel, da durch ihn etliche Werte verloren gehen würden, eine nachgeschaltete Sprachverarbeitung aber kontinuierliche Daten benötigt. Auch eine vorherige softwaremäßige Kalibrierung ist nicht zweckmäßig, da sich damit — auf Kosten einer höheren Prozessorlast für das interpolierte Einfügen von Abtastwerten — der Fehler zwar verkleinern, aber durch die variablen Umweltbedingungen (Schwankungen in Temperatur oder Versorgungsspannung), welche zu einem Drift der Quarze führen, nie ganz eliminieren lassen würde.

Ebenso ist eine kontinuierliche Kalibrierung unzweckmäßig, da dazu auf die Eingangssignale ein Synchronisationssignal gelegt werden müsste, das bei der Verarbeitung im Rechner wieder umständlich zu entfernen wäre.

So bleibt nur die Möglichkeit der Synchronisation auf Hardwareseite. Dabei werden durch eine kleine Modifikation der Schaltung beide Soundkartenquarze gekoppelt (siehe Abb. 3.13).

Der normale Quarzoszillator der Soundkarte besteht aus einem 24MHz-Quarz, zwei Widerständen und zwei Kondensatoren. Die Anregung des Quarzes findet über den 1kΩ Widerstand R2, die Auskoppelung des Signals über C2 statt. Zur Synchronisation wird der

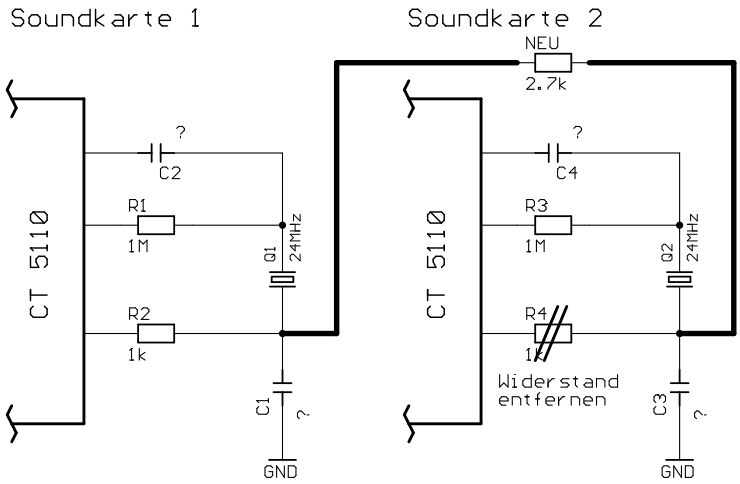


Abbildung 3.13.: Soundkartensynchronisation

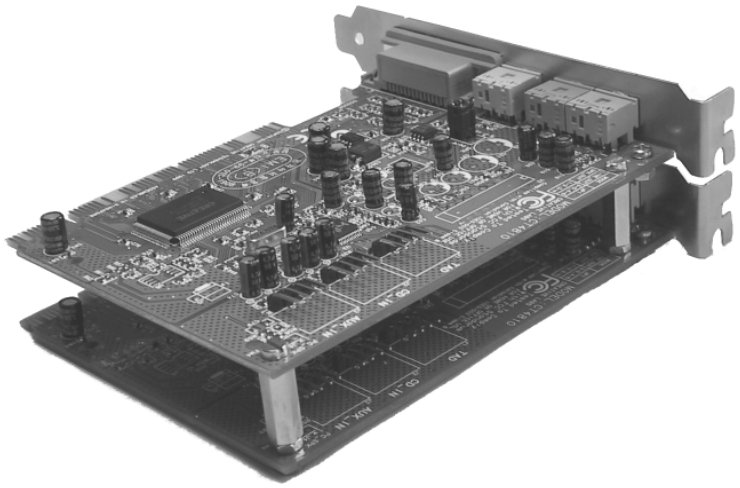


Abbildung 3.14.: Zwei synchronisierte Soundkarten

Anregungswiderstand der zweiten Karte entfernt und der Quarz über einen 2,7kΩ Widerstand und ein möglichst kurzes Kabel — um die kapazitive Belastung der Schaltung klein zu halten — mit dem ersten Quarz verbunden. Auf diese Art regt nun der Oszillator der ersten Karte auch den Quarz der zweiten Karte an, der trotz der leichten Verstimmung noch gut schwingt.

Andere Varianten der Verschaltung, wie etwa die direkte Koppelung von Oszillatorausgang der Karte 1 und -eingang der Karte 2 haben sich als nicht funktionsfähig erwiesen.

Um eine mechanische Belastung des Verbindungskabels auszuschließen, wurden beide Karten mit Abstandsbolzen im Abstand zweier PCI-Steckplätze aufeinander geschraubt (siehe Abb. 3.14).

Kapitel 4

Funktionsweise der Software

Das im folgenden vorgestellte Softwarepaket enthält bis auf den Soundkartentreiber alle Programme, die für die Realisierung einer Schallortung und eines automatisch gesteuerten Richtmikrofons mit der in Kapitel 3 angegebenen Hardware unter Linux erforderlich sind, sowie einige Werkzeuge zur Kalibrierung und für die Visualisierung. Das gesamte Programm ist in C++ geschrieben und verwendet Qt [10] als Bibliothek für die grafische Ausgabe unter XFree86. Der Quellcode selbst ist aufgrund seiner Länge von über 4000 Zeilen dieser Arbeit nicht beigefügt, kann aber von [5] bezogen werden.

Als Soundkartentreiber findet der ALSA-Treiber (Advanced Linux Sound Architecture) [3] Verwendung. Diesem Treiber wurde gegenüber dem üblicherweise auf Linuxsystemen verwendeten OSS-Treiber (Open Sound Source) der Vorzug gegeben, da er über ein leistungsfähigeres Programmierinterface verfügt, welches beispielsweise den synchronen Samplestart von zwei Soundkarten überhaupt erst ermöglicht. Da der ALSA-Treiber auch OSS-kompatibel ist, stellt seine Verwendung auch bei anderen Programmen, die auf eine OSS-Schnittstelle angewiesen sind kein Problem dar.

Für die Benutzung der SoundBlaster PCI 128-Karten mit CT5110-Chipsatz ist eine Treiberversion von mindestens 0.5.9b empfehlenswert.[1]

Obwohl die vorliegende Arbeit nur mit vier Mikrofonen arbeitet, ist die Funktion der Software nicht auf vier Kanäle beschränkt, sondern kann — eine entsprechende Anzahl an Soundkarten vorausgesetzt — beliebig erweitert werden. Dazu werden die Mikrofonpositionen und deren Anzahl in die Headerdatei `lib/micpos.h` eingetragen und das Programm neu compiliert.

Wird aufgrund von Änderungen eine kontinuierliche Ausgabe von Statusmeldungen gewünscht, so findet sich in jeder Programmdatei eine Compilerdirektive `#define DEBUG(x)`,

[1] Versionen älter als 0.5.8 erkennen die Soundkarte nicht. Zwischen Version 0.5.8 und 0.5.9b kann es Schwierigkeiten bei der Initialisierung der Karte auf langsamen Rechnern (200MHz oder weniger) geben.

die durch Ergänzen zu `#define DEBUG(x) x` die Ausgabe einiger Statusmeldungen zur Laufzeit aktiviert.

Für eine Compilierung des Programms gibt man im Verzeichnis `projekt make` ein. Da das Programm Qt verwendet, ist darauf zu achten, dass Qt installiert und die Umgebungsvariable `$QTDIR` korrekt gesetzt ist. [2]

Die folgenden Erläuterungen sollen einen Überblick über die Funktion und die Bedienung der Programme liefern. Eine detailliertere Beschreibung stellen die Kommentare im Quellcode selbst dar.

4.1. Schnittstelle zum ALSA-Treiber

Das Interface des ALSA-Treibers lässt durch seine Komplexität zwar sehr viele Ein-griffs-mög-lich-kei-ten zu, dafür müssen — im Gegensatz zum OSS-Treiber — auch etliche Parameter eingestellt werden, bis das Einlesen von Audiodateien beginnen kann. Aus diesem Grund wurde die Klasse `SoundCard` (zu finden in der Datei `lib/soundlow.cc/.h`) erstellt, welche die ALSA-Schnittstelle vereinfacht. Da die Schallortung keine Audioausgabe benötigt, beschränkt sich die Klasse auf das Einlesen. Auch wird nur ein Datenformat benötigt, sodass die Klasse immer das Datenformat 16 Bit vorzeichenbehaftete Integer-Zahl verwendet.

Das Interface ist daher sehr einfach gehalten:

Zunächst gibt es den Konstruktor `Soundcard()`, dem die Anzahl der zu verwendenden Soundkarten, sowie die Abtastrate und die Zahl der pro Lesezugriff zu übertragenden Abtastwerte übergeben werden. Dieser nimmt dann die vollständige Initialisierung der Soundkarten vor, startet aber noch nicht das Einlesen.

Der Beginn des Einlesens wird mit der Methode `go()` festgelegt. Wird sie aufgerufen, so werden alle verwendeten Soundkarten synchron gestartet.

Die Daten können dann mit der Methode `read()` gelesen werden. Sie übergibt bereits ein Feld vom Typ vorzeichenbehafteter 16 Bit Integer, sodass keine Typkonvertierung erforderlich wird. Die Funktion kehrt erst zurück, wenn genug Daten eingelesen wurden, um den Puffer vollständig zu füllen. Außerdem übernimmt diese Funktion auch die Fehlerbehandlung, welche sogar einen entstehenden Überlauf des Soundkartenpuffers beheben kann. Ein solcher Überlauf tritt dann auf, wenn der benutzte Computer die entstehende Datenmenge nicht in Echtzeit verarbeiten kann. In diesem Fall wird das Einlesen angehalten, alle noch nicht gelesenen Abtastwerte vernichtet, um etwas Zeit zu gewinnen und das Einlesen anschließend wieder synchron gestartet.

Der Aufruf des Destruktors `~Soundcard()` stoppt das Einlesen der Daten.

[2] Das Programm kann sowohl mit Qt Version 1.2.x, als auch mit Version 2.1.x übersetzt werden.

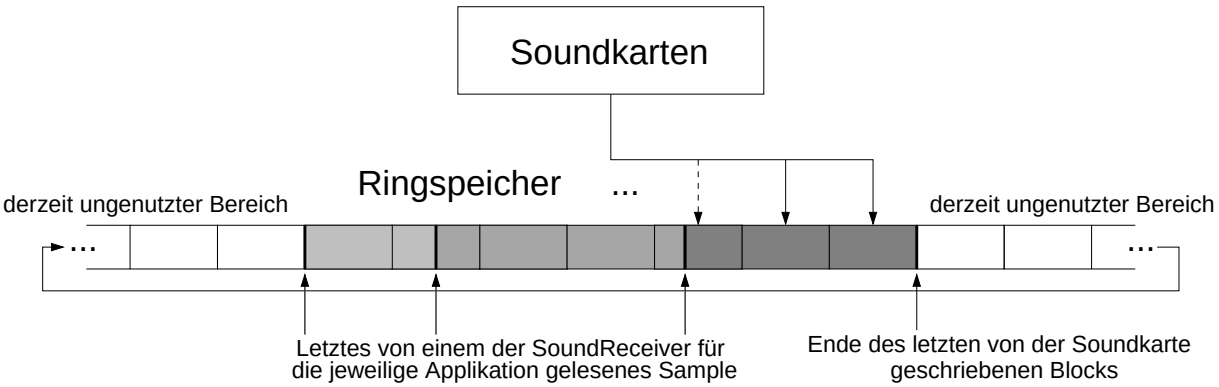


Abbildung 4.1.: Zwischenspeicherung der Audiodaten in einem Ringspeicher

4.2. Soundbibliothek

Das Herzstück des Programms stellt die Soundbibliothek `sndctrl` dar. Sie hat die Aufgabe, die Abtastwerte von der Soundkarte zu lesen sobald sie vorliegen und sie anschließend an alle Unterobjekte (Applikationen) in individuellen Fensterbreiten und -abständen oder -überlappungen, sowie einer für jeden Kanal separat einstellbaren Verzögerung weiterzuleiten. Die Soundbibliothek besteht aus zwei Klassen. Die Klasse `Sound` übernimmt zentral das Einlesen der Daten oder ihre Simulation, wogegen die Klasse `SoundReceiver` separat für jede Applikation die Verteilung der Daten übernimmt (siehe Abb. 4.1).

4.2.1. Klasse `Sound`

Zur Initialisierung wird der Konstruktor `Sound()` mit den Parametern für die Anzahl der gewünschten Audiokanäle, die gewünschte Abtastrate und dem Simulationsmodus aufgerufen. Letzterer bestimmt ob die Sounddaten von den Soundkarten (`MODE_NORMAL`) oder aus einer Datei gelesen (`MODE_FILEIO`) oder ob simulierte Daten verwendet werden sollen (`MODE_SIMULATION`).

Der Konstruktor belegt zunächst Speicher für den Sounddaten-Fifo und initialisiert die Datenquelle. Im Normalmodus wird mit Hilfe des `Soundcard`-Objekts die Soundkarte initialisiert. Sollen die Daten aus einer bzw. mehreren Dateien gelesen werden, so werden diese geöffnet. Wird der Simulationsmodus gewählt, so wird ein Fenster geöffnet, in dem der Einfallswinkel der simulierten Schallquelle eingestellt werden kann.

Zum Start der Datenaquisition wird `startSampling()` aufgerufen. Diese Methode startet das Abtasten durch die Soundkarten und installiert einen Zeitgeber, der genau dann auslöst, wenn die Soundkarten eine Pufferlänge gefüllt haben. Das ist erforderlich, da das Lesen der Sounddaten per Polling erfolgt.

Läuft der Zeitgeber ab, so wird die Methode `timerEvent()` aufgerufen. Diese liest die

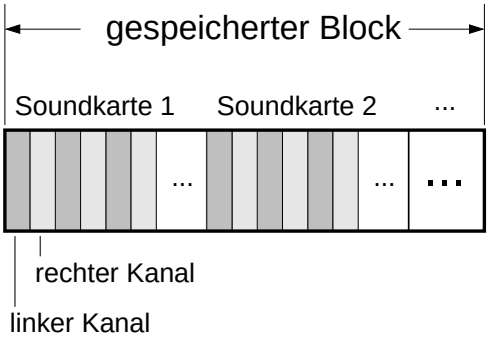


Abbildung 4.2.: Format der im Ringspeicher liegenden Audiodaten

Daten aus der Soundkarte, den Dateien oder dem Simulationspuffer und schiebt diese in den Sounddaten-Fifo. Der Ringspeicher trägt die Daten noch in genau der Struktur, wie sie von den Soundkarten kommen, daher enthält jeder Datenblock zunächst den linken und rechten Kanal von Karte eins, danach den linken und rechten Kanal von Karte zwei usw. (siehe Abb. 4.2). Nach dem Füllen des Puffers wird das Signal `newDataReady()` ausgelöst, auf das die einzelnen `SoundReceiver`-Slots ansprechen.

4.2.2. Klasse `SoundReceiver`

Jede eigenständige Sound-Applikation generiert ein `SoundReceiver`-Objekt. Dabei wird dem Konstruktor ein Pointer auf das zu verwendende `Sound`-Objekt und die Anzahl der angeforderten Kanäle übergeben. Letzteres dient nur dazu, falls weniger Kanäle benötigt werden, als das `Sound`-Objekt gerade verarbeitet.

Der Konstruktor belegt zunächst Speicher für den Datenblock, welcher später der Applikation übergeben werden soll. Anschließend meldet der Konstruktor seinen Slot beim dem ihm übergebenen Sound-Objekt an. Die Breite des Datenblocks kann mit `setFrameSize()` und der Abstand in dem die Blöcke aus dem Audiodatenstrom extrahiert werden sollen mit `setStepSize()` eingestellt werden. Dabei kann der Abstand der Blöcke so gewählt werden, dass sowohl Lücken als auch Überlappungen möglich sind. Die Fensterung der Daten erfolgt mit einem Rechteckfenster.

Darüber hinaus können noch kanalspezifische Verzögerungen durch die Methode `setDelay()` definiert werden.

Sobald nun neue Daten von den Soundkarten vorliegen, werden vom `Sound`-Objekt alle `SoundReceiver`-Objekte durch ein Signal benachrichtigt. Diese überprüfen separat, ob die nun im Fifo angesammelte Datenmenge für die Blocklänge ihrer Applikation ausreichend ist und übergeben diese dann — auch in mehreren kleineren Blöcken, falls die eingelesene Datenmenge einem Vielfachen der Applikations-Blocklänge enspricht — ebenfalls per Signal-Slot-Mechanismus an ihre Applikation.

4.3. Schallortung

Die Schallortung durch die Klasse `Minenergy` (zu finden in der Datei `minenergy.cc/.h`) funktioniert nach dem Prinzip des subtraktiven Beamformers (siehe Abschnitt 2.3) und arbeitet mit einer Fensterbreite von 170ms. Außerdem werden die zu verarbeitetnden Audiosignale um ein ganzes Fenster verzögert, damit die Sprachaktivitätsdetektion genug Zeit erhält, einen Aktivitätsstatus zu ermitteln, bevor eine Ortung erfolgt.

Dem Konstruktor `Minenergy()` werden die Parameter für die horizontale und vertikale Suchwinkelöffnung, sowie das zu verwendende `Sound`-Objekt und ein Verbose-Flag übergeben. Ist das Verbose-Flag gesetzt, so wird die jeweils gefundene Richtung in einem Fenster grafisch dargestellt.

Bei der Initialisierung des `Minenergy`-Objekts wird neben den Einstellungen für das Fenster auch ein `SoundReceiver`-Objekt erzeugt, das für die Bereitstellung der Audiodaten sorgt.

Liegen neue Daten vor, so wird die Methode für den `newDataReady()`-Slot ausgeführt, welche die eigentliche Schallortung durchführt. Dazu wird mit einer Auflösung von 4° für ϑ und φ der gewünschte Bereich abgesucht. Für jeden so abzusuchenden Raumwinkel wird dazu einmal die Funktion `energy()` ausgeführt und derjenige Raumwinkel mit der geringsten Energie als die Richtung zur Schallquelle angenommen. Der so bestimmte Raumwinkel wird durch Senden eines `sweepTo()`-Signals anderen Applikationen zur Verfügung gestellt.

Im Prinzip wäre es — wenn auch umständlich — möglich, anstatt der Differenzbildung auch eine Korrelation (siehe Gleichung 2.11) durchzuführen und diese — anstatt wie hier im Zeitbereich — im Frequenzbereich durchzuführen.

Für die Berechnung wäre eine Fouriertransformation erforderlich. Die Fouriertransformation als FFT implementiert besitzt eine algorithmische Komplexität (N sei die Fensterbreite, n ein Proportionalitätsfaktor) von $N \cdot \ln N \cdot n$ wohingegen die Differenzbildung im Zeitbereich einen Rechenzeitbedarf von (hier sei m der Proportionalitätsfaktor) $N \cdot 90 \cdot m$ aufweist. Geht man nun von der von der Schallortung verwendeten Fensterbreite von 8192 Abtastwerten aus, so ergibt sich, dass die Berechnung im Zeitbereich solange schneller ist, wie keine FFT zur Verfügung steht, deren n besser als $10 \cdot m$ ist, was für die vorliegende Arbeit nicht der Fall ist.

Die Funktion `energy()` selbst bestimmt zunächst aus der Mikrofongeometrie und der Schallgeschwindigkeit die für einen virtuellen Schwenk des Mikrofons in die gewünschte Richtung notwendigen Signalverzögerungen für die einzelnen Kanäle (siehe Gleichung 2.14). Danach werden für jedes mögliche Mikrofonpaar die Differenzenergie über einen Datensatz bestimmt und diese über alle Mikrofonpaare summiert. Für den für diese Arbeit üblichen Fall von vier Mikrofonen steht eine beschleunigte Routine zur Verfügung.

Die Gesamtenergie wird auf die Anzahl der Mikrofonpaare und die Datensatzlänge normiert zurückgeliefert.

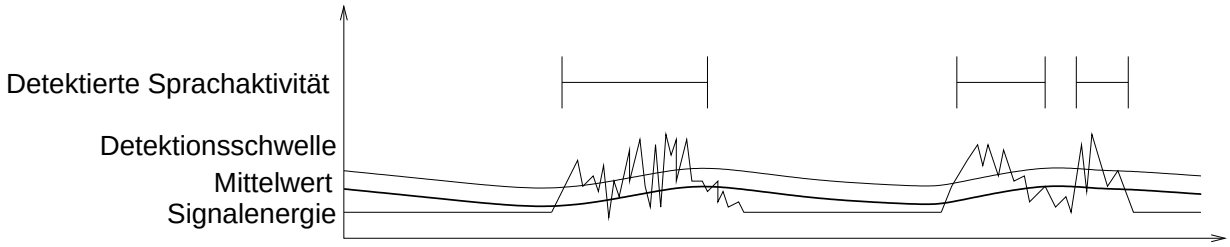


Abbildung 4.3.: Sprachaktivitätsdetektion

4.4. Sprachaktivitätsdetektion

Bei der Ortung eines Sprechers kann das Problem entstehen, dass in Sprachpausen die Schallortung durch Detektion von Hintergrundgeräuschen oder Verstärkerrauschen unsinnige Werte liefert, welche, falls ein simuliertes Richtmikrofon oder eine Videokamera damit gesteuert werden zu unmotivierten Schwenks führen würden. Auch bei einer nachgeschalteten Spracherkennung oder einer akustischen Robotersteuerung kann es wichtig sein zu wissen, wann Sprache vorhanden ist und wann nicht.

Eine einfache Möglichkeit eine Sprachdetektion zu realisieren basiert auf der Annahme, dass ein Sprachsignal starken Dynamikschwankungen unterworfen ist, im Gegensatz zu Hintergrundgeräuschen, welche meistens stationären Charakter haben.

Um nun Signale mit starken Dynamikschwankungen zu detektieren, wird die momentane Signalenergie mit dem gleitenden Mittelwert der letzten Signalenergien verglichen. Überschreitet die momentane Energie die tiefpassgefilterte um etwa 40%, so ist davon auszugehen, dass Sprache präsent ist (siehe Abb. 4.3).

Die Klasse `SpeechActivity` (zu finden in den Dateien `speechactivity.cc/.h`), übernimmt die Aufgabe der Sprachaktivitätsdetektion. Dem Konstruktor `SpeechActivity()` wird nur das verwendete `Sound`-Objekt und ein Verbose-Flag übergeben. Ist letzteres gesetzt, so wird ein kleines Fenster eingeblendet, welches vorhandene Sprachaktivität anzeigt. Auch das `SpeechActivity`–Objekt generiert bei der Initialisierung sein eigenes `SoundReceiver`–Objekt, welches es mit den benötigten Audiodaten versorgt.

Sobald neue Daten vorliegen, wird die Slot-Methode newDataReady() aufgerufen, welche zunächst die gesamte Signalenergie eines Datensatzes über alle Kanäle bildet. Liegt diese mindestens 40% und einen zusätzlichen Offset von 400.000 bit^2 über der tiefpassgefilterten, so wird ein Timer gestartet und ein Signal emittiert, das anderen Applikationen Sprachaktivität anzeigt. Läuft der Timer ab, ohne dass nochmal eine so hohe Signalenergie aufgetreten ist, so wird ein Signal emittiert, welches ein Ende der Sprachaktivität anzeigt.

Anschließend wird die aktuelle Energie E_M durch ein IIR-Filter der tiefpassgefilterten Energie E_{DS} hinzugefügt:

$$E_{DS\,neu} = (E_{DS\,alt} \cdot (40 - 1) + E_M)/40$$

Der Zahlenwert von 40 wurde empirisch ermittelt und führt zu einer konservativen Detektion.

4.5. Simuliertes Richtmikrofon

Das simulierte Richtmikrofon hat die Aufgabe, die Richtung der maximalen Empfindlichkeit auf ein vorgegebenes Ziel zu schwenken, um Störgeräusche aus anderen Richtungen zu unterdrücken.

Dies wird von der Klasse `OutputPlugin` (zu finden in `outputplugin.cc/.h`) übernommen. Bei der Initialisierung der Klasse werden dem Konstruktor `OutputPlugin()` ein Flag für die Aktivierung des Messmodus, die Abtastrate für die Ausgabe, ein Verbose-Flag und das zu verwendende `Sound`-Objekt übergeben.

Wird das Verbose-Flag gesetzt, so wird die Richtung in die das Mikrofon gerade gerichtet ist, in einem Fenster grafisch angezeigt. Ist der Messmodus aktiviert, so wird die aktuelle Mikrofonausrichtung auch noch als Zahlenwert auf Standard Error ausgegeben. Da eine dem Programm nachgeschaltete Spracherkennung unter Umständen nicht mit der Abtastrate zurechtkommt, mit der das Programm arbeitet, bietet die `OutputPlugin`-Klasse auch die Möglichkeit, einen Ausgangsdatenstrom zu liefern, dessen Abtastrate beliebig herabgesetzt werden kann.

Der Konstruktor initialisiert alle Variablen und generiert ein `SoundReceiver`-Objekt, welches für die Bereitstellung der Audiodaten sorgt.

Die Klasse besitzt darüber hinaus einen Slot `speechActivityChanged()` über welchen sie von Sprachaktivität in Kenntnis gesetzt werden kann.

Das Schwenken des Mikrofons wird durch das Empfangen eines `sweepTo()`-Signals ausgelöst, was auf zwei Arten erfolgen kann. Entweder wird die Schallortung dafür verwendet und dessen `sweepTo()`-Signal mit dem `sweepTo()`-Slot des simulierten Richtmikrofons verbunden, oder es wird eine manuelle Steuerung benutzt, bei der der Benutzer die Raumwinkel manuell einstellen kann. Eine solche manuelle Steuerung wird durch die Klasse `SweepControl` (zu finden in `outputplugin.cc/.h`) realisiert. Auch sie generiert ein `sweepTo()`-Signal, welches zur Einstellung der Mikrofonrichtung benutzt werden kann.

Wird also die Funktion, welche dem `sweepTo()`-Slot zugeordent ist aufgerufen, so wird zunächst überprüft, ob Sprachaktivität vorliegt. Ist das der Falls, so berechnet die Funktion aus der Mikrofongeometrie und der Schallgeschwindigkeit die für den Mikrofonschwenk erforderlichen Signalverzögerungen, welche dazu führen, dass ein Signal aus der gewünschten

Richtung, wenn es mit den Verzögerungen beaufschlagt wird, bei allen Mikrofonen gleichzeitig ankommt. Dies führt zu einem Schwenk der Richtung der maximalen Empfindlichkeit (siehe Abschnitt 2.3.3). Die berechneten Verzögerungen werden dann im `SoundReceiver`-Objekt eingestellt, welches diese dann beim Liefern von Audiodaten anwendet.

Liegen neue Audiodaten vor, so wird die Funktion `newDataReady()` aufgerufen. Diese summiert zunächst alle Audiokanäle, die vom `SoundReceiver`-Objekt bereits mit den für einen Schwenk erfordelichen Verzögerungen versehen sind, sodass ein Mono-Signal entsteht, welches bevorzugt Schallanteile aus der eingestellten Mikrofonrichtung enthält. Dieser Datenblock liegt aber noch in der Abtastrate mit welcher das `Sound`-Objekt arbeitet vor. Um diese herabzusetzen, wird für jeden Abtastwert der Ausgangsdaten entsprechend dem Verhälnis von neuer zu alter Abtastrate über mehrere Abtastwerte der Eingangsdaten gemittelt. Diese Ausgangsdaten werden auf `Standard Out` ausgegeben, sodass sie einfach mit Hilfe einer Pipe, also der Redirektion der Ausgabe einem anderen Programm zugeführt werden können.

4.6. Zubehör

Das Programm `soundsource` verfügt neben den Objekten zur Schallortung, für das Richtmikrofon und für die Sprachaktivitätsdetektion auch noch über etwas Zubehör, welches Abgleich und Betrieb der Hardware erleichtert, wie eine Aussteuerungsanzeige, ein Oszilloskop und eine Darstellung der Mikrofonpositionen.

4.6.1. Aussteuerungsanzeige

Die Aussteuerungsanzeige wird realisiert durch die Klasse `Meter` (zu finden in `meter.cc/.h`) (siehe Abb. 4.4). Bei der Initialisierung des Objekts wird nur ein Zeiger auf das zu verwendende `Sound`-Objekt übergeben. Der Konstruktor erstellt dann ein neues Fenster, welches entsprechend der Anzahl der verwendeten Mikrofone Aussteuerungsbalken anzeigt und ein neues `SoundReceiver`-Objekt generiert, welches für die Bereitstellung der Audiodaten sorgt.

Liegen neue Daten vor, so wird die Signalenergie eines Datensatzes berechnet, diese auf die Datensatzlänge normiert und für eine übersichtlicheren Darstellung logarithmiert. Steigende Pegel werden sofort berücksichtigt, fallende werden tiefpassgefiltert dargestellt um eine gefälligere Anzeige zu erreichen.

Entsprechend werden dann für jeden Kanal horizontale Linien in dem Pegel entsprechenden Farben in ein Pixmap gezeichnet. Sind alle Linien aller Kanäle fertig, so wird mit `repaint()` die Methode `paintEvent()` aufgerufen, welche das Pixmap schließlich darstellt.

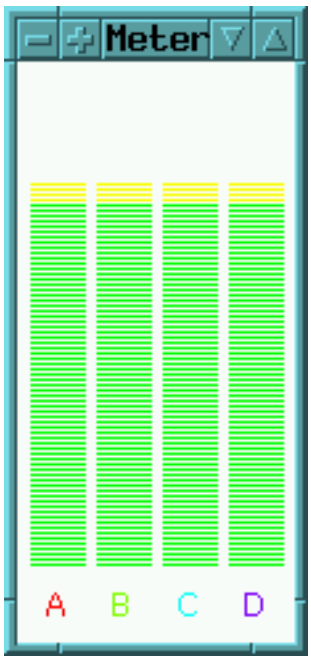


Abbildung 4.4.: Aussteuerungsanzeige

4.6.2. Oszilloskop

Ein weiteres praktisches Tool stellt das Oszilloskop realisiert durch die Klasse `Oszi` (zu finden in `oszi.cc/.h`) dar (siehe Abb. 4.5).

Bei der Initialisierung des Objekts wird dem Konstruktor nur das zu verwendende `Sound`-Objekt übergeben. Dieser generiert ein Fenster, in welchem die Wellenzüge für die verwendeten Kanäle gezeichnet werden sollen. Außerdem wird ein Pixmap erzeugt, damit eine schnellere Darstellung der Kurven möglich wird.

Auch hier sorgt ein `SoundReceiver`-Objekt wieder für die Bereitstellung der Sounddaten.

Liegen neue Daten vor, so wird der Inhalt des Pixmaps zunächst gelöscht und eine horizontale Linie als Nulllinie eingezeichnet. Anschließend wird Kanal A nach dem ersten von negativ nach positiv erfolgendem Nulldurchgang durchsucht und dieser Zeitpunkt als Triggerposition für alle Kanäle verwendet. Damit wird ein stehendes Bild erreicht.

Nun werden für alle Kanäle mit unterschiedlichen Farben die Abtastwerte in das Pixmap eingetragen, wobei benachbarte Punkte mit einer Linie verbunden werden.

Sind alle Punkte eingetragen, so wird mit `repaint()` ein `paintEvent()` ausgelöst, welcher das Pixmap auf den Bildschirm kopiert.

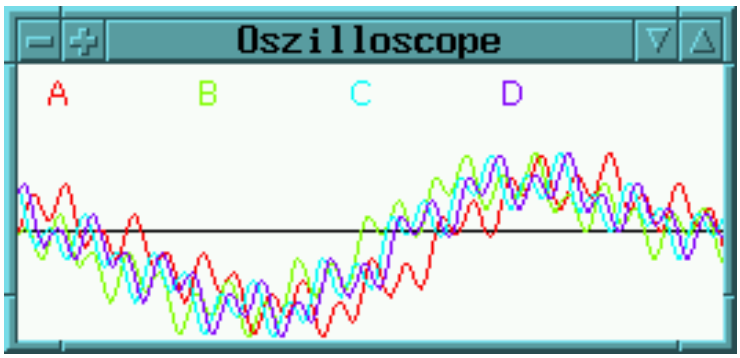


Abbildung 4.5.: Oszilloskop

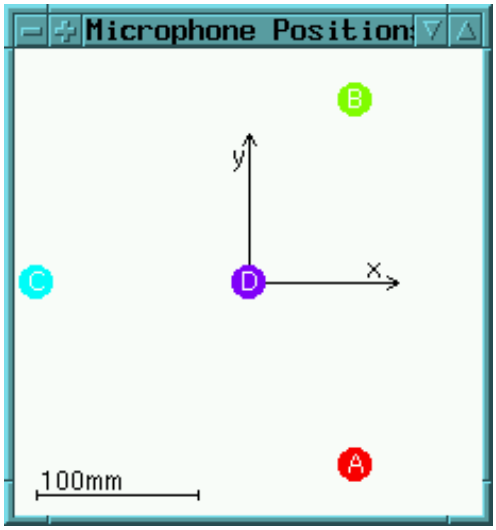


Abbildung 4.6.: Mikrofonpositionsanzeige

4.6.3. Mikrofonpositionsanzeige

Zur besseren Übersicht über die Bezeichnung der Mikrofone wurde ein weiteres Werkzeug programmiert, welches die in der Datei `micpos.h` angegebenen Mikrofonpositionen in einem maßstabsgetreuen Diagramm abbildet (siehe Abb. 4.6). Als Perspektive wurde dabei der Blick auf die x-y-Ebene gewählt. Die dargestellten Mikrofonbezeichnungen (A,B,C,...) und deren Farben entsprechen den Bezeichnungen und Farben wie sie auch das Oszilloskop und die Aussteuerungsanzeige verwenden.

Das Koordinatensystem ist dabei das gleiche, wie es auch bei den Darstellungen des simulierten Richtmikrofons und (falls $\vartheta = 0$ gewählt ist) der Schallortung verwendet wird.

4.7. Hauptprogramm

Das Hauptprogramm (`main.cc`) hat die Funktion, die Kommandozeilenparameter zu parsen, gegebenenfalls eine kurze Übersicht über die Befehle auszugeben und die Objekte des Programms nach den Angaben der Kommandozeile zu starten.

4.8. Weitere Programme

Zusätzlich zu dem Programm `Soundsource`, welches die Funktionen der Schallortung übernimmt gibt es noch ein paar zusätzliche eigenständige Programme, die für den Abgleich, den Betrieb oder die Visualisierung verwendet werden können.

4.8.1. Richtcharakteristiksimulation

Zur Bestimmung der Richtcharakteristik von Mikrofonanordnungen kann das Programm `micro` (zu finden unter `simulations/main.cc/.h`) verwendet werden (siehe Abb. 4.7). Das Programm verwendet dabei ebenfalls die Mikrofonpositionen, welche in `micpos.h` eingetragen sind.

Nach dem Starten werden in der Mitte des Fensters die Mikrofone eingezeichnet. Die farbigen Kurven geben die normierte Empfindlichkeit der Mikrofonanordnung (maßstabsgetreu in Bezug auf die Signalleistung) in Abhängigkeit der Richtung zur Schallquelle an. Die Zuordnung zwischen Farbe und Frequenz kann der Tabelle auf der rechten Seite des Fensters entnommen werden. Die schwarze Linie stellt dabei die Leistungssumme über alle Frequenzen zwischen 300 und 3300 Hz dar, was dem wichtigsten Teil des Sprachspektrums entspricht.

Am unteren Rand findet sich ein Scrollbalken, mit dem der Faktor der Signalverzögerung für die Mikrofone eingestellt werden kann. Ein Faktor von 100% bedeutet dabei, dass die Verzögerung der Schallaufzeit zwischen den Mikrofonen entspricht. Der Scrollbalken am rechten Rand dient dazu, die Richtung der maximalen Empfindlichkeit zu schwenken.

Mit den Knöpfen am oberen Rand des Fensters kann das zum jeweiligen Mikrofon gehörende Signal invertiert werden.

Um die Diagramme zu erstellen, wird von Programm für jede der angegebenen Frequenzen eine in 3m Entfernung angenommene virtuelle Schallquelle in 2°-Schritten um die Mitte der virtuellen Mikrofonpositionen geführt. Für jeden Winkelschritt werden nun die Abstände zwischen der Schallquelle und den einzelnen Mikrofonen und damit die Signallaufzeiten bestimmt. Nun wird je ein Sinus- und ein Cosinus-Signal der gewünschten Frequenz mit den durch die Laufzeiten zu den einzelnen Mikrofon entstehenden Verzögerungen beaufschlagt. Anschließend werden alle Sinus- und Cosinus-Anteile der so bestimmten virtuellen

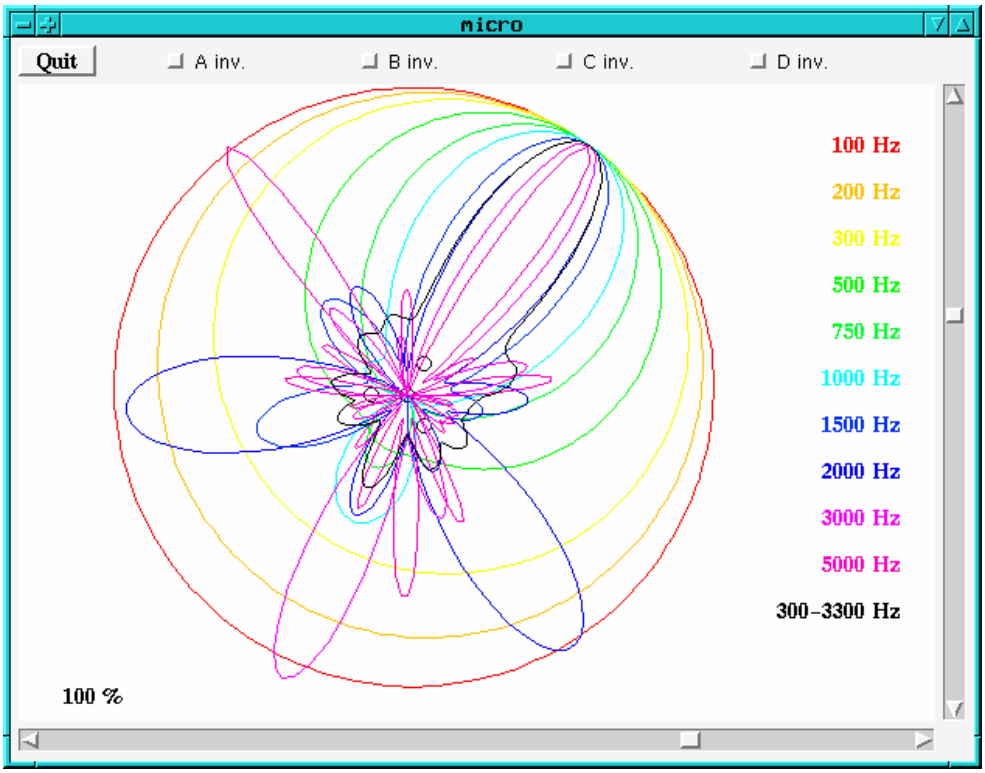


Abbildung 4.7.: Programm Micro

Mikrofonsignale quadriert und addiert. Die so ermittelte Signalleistung wird noch auf die Mikrofonanzahl normiert und in das Diagramm eingetragen.

4.8.2. Vierspurrekorder

Das Programm `soundsource` verfügt über die Möglichkeit, seine Audiodaten statt von den Soundkarten auch aus Dateien zu lesen. Diese können beispielsweise mit dem Vierspurrekorder `quadrec` erzeugt werden.

Nachdem der Soundkartenmixer auf die richtigen Werte gesetzt ist (z.B. mit Alsamixer) wird das Programm mit dem gewünschten Namen für die Ausgabedateien als Kommandozeilenparameter gestartet. Dieses schreibt dann die mit einer Abtastrate von 48kHz eingelesenen Audiodaten in vier Dateien (`name01.raw` ... `name04.raw`) auf die Festplatte. Es sollte bei dieser Art des Harddiskrecordings genug Speicherplatz zur Verfügung stehen, da die produzierte Datenmenge von 375kB/s beispielsweise auch 200MB freien Platz in 9 Minuten füllt!

Der Vierspurrekorder benutzt ebenfalls die Low-Level-Soundklasse `SoundCard` um die Daten von den Soundkarten zu lesen, sortiert die Abtastwerte nach den einzelnen Kanälen und schreibt diese dann auf die Platte.

4.8.3. Synchronitätsmessung

Werden als Audiodatenquelle mehrere Soundkarten verwendet, so müssen diese synchronisiert werden, um den Frequenzunterschied der Oszillatorquarze zu eliminieren (siehe Abschnitt 3.6). Der Erfolg dieser Maßnahmen kann entweder mit dem Oszilloskop des `soundsource`-Programms (siehe Abschnitt 4.6.2) oder mit der Synchronitätsmessung überprüft werden.

Wählt man letztere Methode, so schaltet man den Synchronsignalgenerator auf dem Line-Verstärker (siehe Abschnitt 3.4) ein und startet das Programm.

Es wird nun fortlaufend der Versatz zwischen Kanal A und C, A und B, sowie C und D angezeigt. Bleibt der Wert für den Versatz zwischen Kanal A und C konstant, so arbeiten beide Soundkarten synchron. Konstante Werte, die betragsmäßig kleiner gleich zwei sind, beeinflussen die Genauigkeit der Ortung nicht nennenswert.[3]

Das Programm kann durch Drücken von `Strg-C` beendet werden.

[3] Die Werte für den Versatz A-B und C-D sollten in jedem Fall konstant sein, andernfalls bedeutet das, dass der rechte und linke Kanal einer Soundkarte nicht synchron abgetastet wird. Interessanterweise wurde bei den verwendeten Soundkarten ein Versatz zwischen dem rechten und dem linken Kanal von ein bis zwei Abtastintervallen gemessen.

Kapitel 5

Messungen und Ergebnisse

Um die Leistungsfähigkeit der Schallortung zu ermitteln wurden Messreihen mit verschiedenen Konfigurationen aufgenommen. Die Messungen fanden in einem etwa $20m^2$ großen Raum statt, der außer einem Teppichboden keinerlei schallschluckendes Mobiliar enthielt und somit einen auch bereits deutlich hörbaren Nachhall erzeugte.

Als Störgeräuschquellen dienten einige PCs, die mit ihren Lüftern und Festplatten einen bereits auffälligen Hintergrundgeräuschpegel verursachten. Das zu ortende Geräusch wurde — der Reproduzierbarkeit wegen — mit Hilfe eines Lausprechers erzeugt, wobei die Schallortung kontinuierlich die horizontale Schallquellenposition ermittelte. Die Lautstärke wurde so gewählt, dass die Sprache in einer Entfernung von 4m noch gut verständlich war. Der Empfänger wurde in mindestens 1.5m Abstand zu einer Wand, in gleicher Höhe wie der Lausprecher aufgestellt. Bei den Messungen wurde stets die Sprachaktivitätsdetektion benutzt, da ansonsten bei dem relativ lauten Hintergrundgeräusch die Art des Sprechens — mehr oder weniger Pausen — die Messergebnisse stark beeinflusst hätte.

5.1. Messung mit statischem Empfänger

Für jede zu vermessene Entfernungs-Winkel-Kombination wurden jeweils vier Messreihen zu je 200 Werten aufgezeichnet. Messwerte welche um mehr als einen Abtastwert, also $\pm 4°$ von der tatsächlichen Position abwichen, wurden als falsch gewertet. Von den vier Messreihen wurde die schlechteste und die beste gestrichen und aus den anderen beiden der Mittelwert gebildet. Abhängig von der Entfernung und dem Winkel zur Schallquelle, sowie der Art des Geräusches (Sprache oder Musik) ergeben sich folgende Trefferquoten:

Entfernung zur Quelle: 1m

Art des Geräuschs: Sprache

0°	90°	180°
90%	77%	80%

Zum Vergleich: Richtungsbestimmung durch die Summe absoluter Differenzen anstatt durch die Summe der Differenzenquadrate (siehe Abschnitt 2.3.2):

0°
82%

Art des Geräuschs: Musik

0°	90°	180°
94%	90%	75%

Entfernung zur Quelle: 2m, Art des Geräuschs: Sprache

0°
59%

Entfernung zur Quelle: 4m, Art des Geräuschs: Sprache

0°
27%

Zum Vergleich: Bei einer geforderten Genauigkeit von ±4° würde die Trefferquote bei gleichverteilten, zufälligen Richtungsbestimmungen 3,3% betragen.

Eine Ortung von impulsartigen Signalen, wie Klatschen, Schnippen oder Klopfen war praktisch nicht möglich.

Die Prozessorauslastung bei obigen Messungen betrug auf einem Pentium III mit 866MHz Taktfrequenz etwa 45%. Auch die Verwendung absoluter Differenzen anstatt Differenzenquadraten für die Berechnung zeigte keine signifikante Änderung in der Prozessorlast. Bei dieser Auslastung konnten während der gesammten Messung keine Pufferüberläufe beobachtet werden.

Die Verbesserung des Signal-Rauschverhältnisses durch den Einsatz von vier anstatt einem Mikrofon konnte in Ermangelung eines schalltoten Raums nicht quantitativ bestimmt werden, jedoch ist sie beim direkten Vergleich zwischen dem Signal eines Mikrofons und dem Ausgangssignal des gesteuerten Richtmikrofons deutlich hörbar.

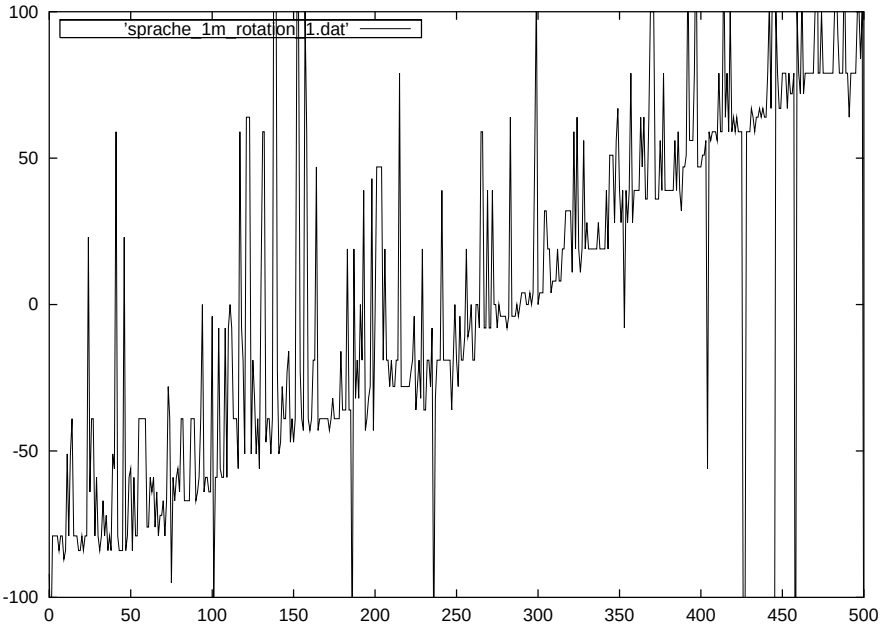


Abbildung 5.1.: Messung mit langsam rotierendem Empfänger bei Sprachsignal

5.2. Messung mit bewegtem Empfänger

Nach den Messungen, bei denen jeweils Schallquelle und Empfänger statisch waren, wurden unter den gleichen Bedingungen zwei weitere Messreihen aufgenommen, bei denen der Empfänger relativ zum Sender langsam um 180° (von -90° bis +90°) gedreht wurde. Es wurde je eine Messreihe für Sprache (siehe Abb. 5.1) und Musik (siehe Abb. 5.2) aufgenommen.[1]

5.3. Bewertung der Messungen

Wie sich aus obigen Messungen zeigt, ist das von mir entwickelte Verfahren gut dazu geeignet, auch in verhallten Räumen eine Ortung aktiver Geräuschquellen bis etwa zwei Meter Entfernung durchzuführen. Bei größeren Abständen wird die Trefferquote allerdings zunehmend schlechter. Das liegt daran, dass Reflektionen ab dieser Entfernung einen erheblich größeren Anteil am einfallenden Signal darstellen und somit zunehmend eine Ortung der Echos stattfindet. Das Problem ließ sich nachvolziehen, indem versucht wurde, eine manuelle Ortung durchzuführen. Dazu wurde auf einen geschlossenen Kopfhörer seitenrichtig das Signal der Mikrofone A und B direkt eingespielt (aufgrund des gleichen Abstandes der

[1] Die Beschriftung der Abszisse bezeichnet natürlich die Nummer des Messwerts, nicht den tatsächlichen Winkel.

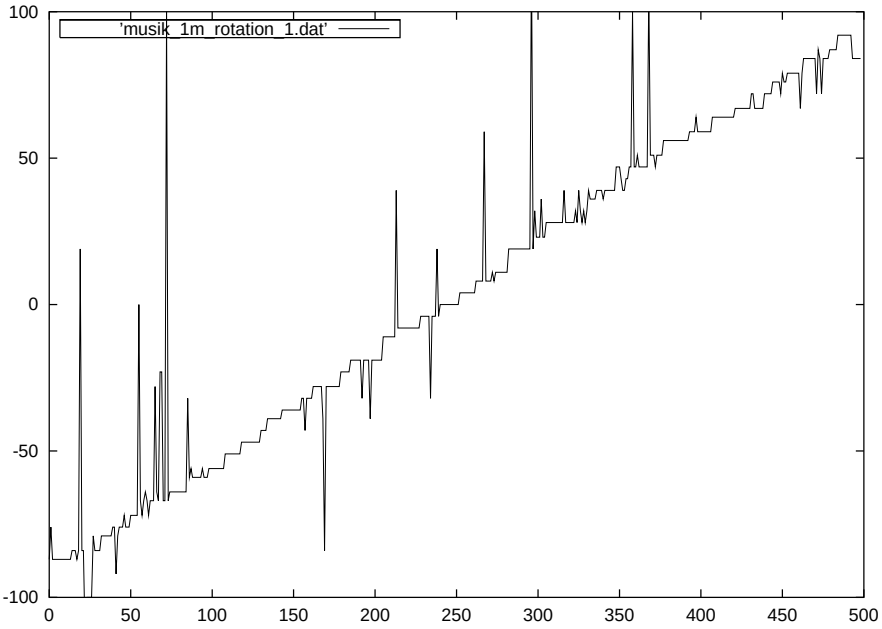


Abbildung 5.2.: Messung mit langsam rotierendem Empfänger bei Musik

Mikrofone, wie dem der menschlichen Ohren ist das zulässig) und damit die Ortung eines Geräuschs im Raum versucht, was ohne die Verknüpfung des Höreindrucks mit anderen Sinneseindrücken bereits erhebliche Schwierigkeiten bereitete.

Es stellte sich ebenfalls heraus, dass die Ortung von kontinuierlichen Signalen wie Musik besser funktioniert, als die von Sprache, da es bei nicht-kontinuierlichen Signalen mehr Stellen mit geringer Lautstärke gibt, an denen Störsignale das Geräusch übertönen können. Auch die Verwendung der Sprachaktivitätsdetektion verbessert in stark verhallten Räumen die Ortungsqualität nur wenig, da sie natürlich nicht entscheiden kann, ob das eintreffende, lautere Signal über den direkten Ausbreitungsweg beim Mikrofon ankommt, oder ob es durch die zufällig lautere Summe einiger Echos entsteht.

Eine Ortung impulsartiger Signale, wie Klatschen, Knacken oder Schnippen ist — zumindest in nicht-schallgedämmten Räumen — mit dieser Methode praktisch nicht möglich. Das erklärt sich, wenn man bedenkt, dass die verwendete Fensterbreite der Ortung 170ms beträgt. Nimmt man nun einen Abstand der Wände des Raumes zu 5m an, so bedeutet das, dass ein impulsartiges Geräusch zusammen mit mehr als zwanzig (!) aus verschiedenen Richtungen einfallenden Echos in einem Fenster wiederzufinden ist. Da das Programm nicht erkennen kann, welches das Signal des direkten Ausbreitungswegs ist, versucht es, eine Richtung zu finden, die für das gesamte einfallende Signal passend ist, was natürlich zum Scheitern verurteilt ist. Eine Methode, die das Problem umgeht, wird in Kapitel 6 vorgeschlagen.

Die in Abschitt 2.3.1 vorgeschlagene Modifikation des Ortungsalgorithmus zur Senkung des

Rechenzeitbedarfs wurde ebenfalls ausgemessen. Sie führte allerdings neben einer leichten Verschlechterung der Ortungsqualität zu keiner signifikanten Senkung des Rechenzeitbedarfs.

Kapitel 6

Ausblick

Ein Nachteil der verwendeten Methode besteht im immer noch relativ hohen Rechenzeitbedarf, der durch die paarweise Differenzbildung der eingelesenen Werte verursacht wird. Abhilfe könnte hier die Verwendung der bei Prozessoren des 80x86er Typs neuerer Bauart vorhandenen sogenannten Multimedia Extension (MMX) sein. Diese Prozessoreinheit beschleunigt einige Berechnungen, wie das Differenzenquadrat, die Fourier- oder die Cosinustransformation erheblich, sodass auch der Ortungsalgorithmus — je nach Realisierung (Frequenz- oder Zeitbereich) — mindestens 75% schneller zu berechnen sein sollte.

Eine andere Möglichkeit zur Senkung der Rechenleistung bestünde darin, den Abstand der Mikrofone zu erhöhen, sodass schon bei niedrigerer Abtastrate die gleiche Winkelauflösung erreicht wird. Das hat dann natürlich zur Folge, dass bei gleicher Fensterbreite seltener Richtungsinformationen erzeugt werden.

Die Genauigkeit der Anordnung sollte sich weiter erhöhen lassen, indem für die Ortung mehr als vier Mikrofone verwendet werden.[1] Ebenfalls könnte der Einsatz eines Filters, welcher die Bandbreite des einfallenden Signals beschneidet, möglicherweise die Ortungsqualität weiter verbessern.

Natürlich ist der in dieser Arbeit verwendete Algorithmus nicht in der Lage, gleichzeitig mehrere Geräuschquellen zu orten, wohingegen das menschliche Ohr sehr wohl zu derartigem in der Lage ist. Der Grund dafür liegt in der Funktionsweise des Ohres, das einfallende Signale zunächst in den Frequenzbereich transformiert und dann dort auswertet. Durch den zeitlich versetzten Beginn der Signale oder durch Erkennen des Geräuschs und Zuordnen der Obertöne zu dem jeweilen Geräusch können verschiendene Schallereignisse wieder voneinander getrennt werden. Abbildung 6.1 zeigt das Spektrum eines Sprachsignals und Abbildung 6.2 zeigt das Spektrum eines Musiksignals (klassische Musik). Dabei wird der Betrag der Spektralkoeffizienten als Grauwert, die Zeit auf der Abszisse (1 Segment = 1 s) und die Frequenz auf der Ordinate (1 Segment = 500Hz) dargestellt.

[1] Die entwickelte Software ist bereits für die Verwendung von mehr als vier Mikrofonen ausgelegt.

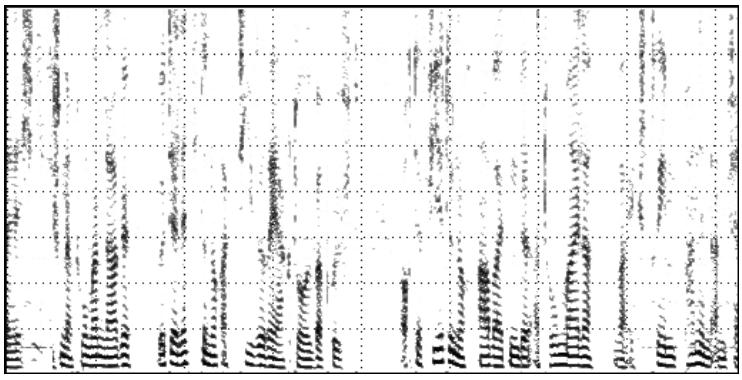


Abbildung 6.1.: Spektrum eines Sprachsignals (kontrastverstärkt)

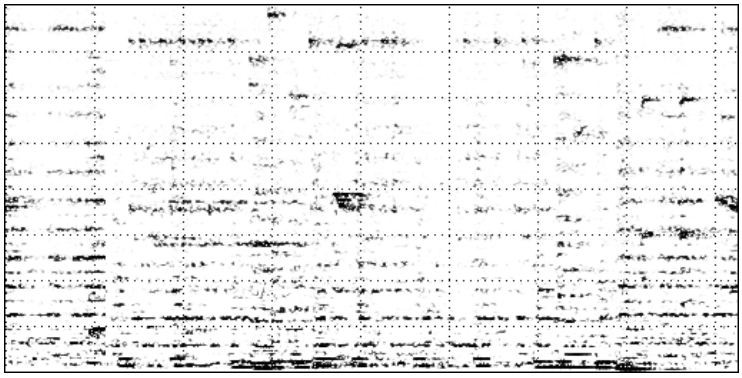


Abbildung 6.2.: Spektrum eines Musiksignals (kontrastverstärkt)

Um nun aber auch noch den Versatz eines an zwei oder mehreren Mikrofonen ankommenden Geräuschs zu bestimmen, wäre — neben der bereits sehr genauen Frequenzauflösung, die für die Trennung erforderlich ist — auch noch eine sehr feine Zeitauflösung nötig, die praktisch nur durch eine Berechnung der Fouriertransformation nach jedem eingelesenen Abtastwert möglich wäre. Das ist derzeit aber aufgrund der extrem hohen Rechenleistung unpraktikabel. Sobald die erforderlichen Prozessoren zur Verfügung stehen, ist dieser Methode sicherlich der Vorzug zu geben.

Bei Anwesenheit einer lauten Störquelle kann es vorkommen, dass die Ortung zwischen der Störquelle und dem Sprecher wechselt, je nachdem welche der beiden gerade lauter erscheint. Durch Einsatz einer Kamera, welche die in Frage kommenden Richtungen absucht und einen Gesicht mit Mundbewegungen detektiert, ließe sich auch in diesem Fall eindeutig der tatsächliche Sprecher identifizieren.

Mit der in dieser Arbeit vorgestellten Anordnung ist nur eine Richtungs-, aber keine Positionsbestimmung der Schallquelle möglich. Sollte dies aber gewünscht sein, so kann man die Position durch Einsatz einer zweiten Anordnung, die gegenüber der ersten etwas versetzt ist, triangulieren.

Bei Anwesenheit von lauten und instationären Hintergrundgeräuschen wird auch ohne Sprecher eine Detektion von Geräuschquellen stattfinden. Möglicherweise wird dann eine nachgeschaltete Spracherkennung fälschlicherweise im Hintergrundgeräusch Steuerkommandos erkennen, was zu ungewünschten Aktionen des Roboters führen würde. Um das zu umgehen, wäre der Sprachaktivitätsdetektor so einzustellen, dass er nur auf sehr laute und impulsartige Geräusche anspricht. Dann könnte man den mobilen Roboter durch Klatschen “aufwecken” und anschließend akustisch Steuerkommandos eingeben, ohne dass es zu Fehlinterpretationen kommen kann.

Literaturverzeichnis

[1] R. Barthel. Ein robustes akustisches Quellenortungsverfahren für multimediale Anwendungen in verhallter Umgebung. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2000.

[2] diverse. *Professionelle Schaltungstechnik Band 9.* Franzis', 1992.

[3] ALSA Driver. Advanced Linux Sound Architecture. http://www.alsa-project.org.

[4] W. Eggert, R. Lehnert, B. Leopold, W. Röhrer, A. Schief, K. Schubert, and A. Senner. *Tabellenbuch Metall.* Verlag Europa-Lehrmittel, 1952.

[5] R. Gruber. Robins Nest. http://www.robinsnest.de.

[6] J. Hagenauer. *Manuskript zur Vorlesung Nachrichtentechnik 1.* Lehrstuhl für Nachrichtentechnik, 1996.

[7] E. Hering, R.Martin, and M.Stohrer. *Physik für Ingenieure.* Springer-Verlag, 1997.

[8] H. Stöcker. *Taschenbuch mathematischer Formeln und moderner Verfahren.* Verlag Harri Deutsch, 1993.

[9] E. Terhardt. *Skriptum zur Vorlesung: Elektroakustik und Audiokommunikation.* Fachschaft Elektro- und Informationstechnik, 1992.

[10] Trollsoft. http://www.trollsoft.com.

[11] E. Zwicker and M. Zollnet. *Elektroakustik.* Springer-Verlag, 1997.

Anhang A

Bedienungsanleitung

A.1. Inbetriebnahme

Um die Mikrofone und den Line-Verstärker mit einem Linux-PC zu verbinden, werden zunächst die beiden hardwaremäßig synchronisierten Soundkarten (siehe Abschnitt 3.6) in zwei benachbarte PCI-Steckplätze gesteckt, der Line-Verstärker (siehe Abschnitt 3.10) in die Line-In-Eingänge der Soundkarten gesteckt und die Mikrofone mit dem Line-Verstärker verbunden. Anschließend wird der ALSA-Treiber zusammen mit der ALSA-Library und den ALSA-Utilities installiert [ALSA] und die für die Karten erforderlichen Parameter in die Datei `etc/modules.conf` eingetragen. Üblicherweise sieht das wie folgt aus:

```
alias char-major-14 soundcore

alias char-major-116 snd alias snd-card-0 snd-card-ens1371

alias sound-slot-0 snd-card-0

alias sound-service-0-0 snd-mixer-oss

alias sound-service-0-1 snd-seq-oss

alias sound-service-0-3 snd-pcm-oss

alias sound-service-0-8 snd-seq-oss

alias sound-service-0-12 snd-pcm-oss
```

Werden mehrere Soundkarten benutzt, so folgen weitere Einträge, welche `card-1` bzw. `card-2` etc. und damit auch `sound-service-1-...` bzw. `sound-service-2-...` etc. konfigurieren.

Nach einem Neustart des Rechners wird die Installation (z.B. mit `arecord` und `aplay`) überprüft. Eventuell muss noch mit einem Mixer (z.B. `alsamixer`) die Aufnahmequelle auf

Line-In gestellt, die Stummschaltung aufgehoben und die Eingangsempfindlichkeiten auf gleiches Niveau gesetzt werden. Danach startet man das Programms `soundsource` (siehe Kapitel 4) mit den Parametern `-tlp` . Nun kann man anhand der Aussteuerungsanzeige und des Diagramms mit den Mikrofonpositionen die richtige Zuordnung der Mikrofone überprüfen. Stimmt sie nicht, so wird entsprechend umgesteckt. Nun kann der Abgleich erfolgen.

A.2. Abgleich

Der Abgleich der Anordnung wird sowohl nach dem ersten Aufbau, nach Austausch von Baugruppen oder Modifikation der Schaltung, als auch nach einem Umstecken der Mikrofone erforderlich.

Dazu wird die Mikrofonanordnung gleichmäßig mit einem kontinuierlichen Geräusch beschallt, das in etwa den Schalldruck des später zu ortenden Signals besitzt. Das läßt sich beispielsweise durch Aufstellen der Anordnung in einem stark verhallten Raum und Beschallen mit einer akustischen Rauschquelle erreichen. Nun startet man das Programm `soundsource` mit den Parametern `-tlp` und justiert mit den Spindeltrimmern auf der Line-Verstärker-Platine (siehe Abschnitt 3.10) die Balken der Aussteuerungsanzeige auf gleiche Höhe im gelben Bereich. Eventuell muss hierzu der Mixer der Soundkarte (z.B. mit `alsamixer`) nachgestellt werden, wenn der Pegel nicht allein durch Verdrehen der Spindeltrimmer erreicht werden kann. Diese Mixereinstellung ist in jedem Fall zu speichern, sodass sie bei einem Neustart des Rechners wieder vorliegt. Damit ist der Abgleich abgeschlossen.

A.3. Betrieb

Das Programm `soundsource` besitzt einige Kommandozeilenparameter über welche die verschiedenene Funktionen ausgewählt und konfiguriert werden können. Das Programm arbeitet immer mit der in `micpos.h` definierten und fest eincompilierten Anzahl an Audioeingängen. Wird das Programm ohne Parameter aufgerufen, so wird eine Liste der möglichen Parameter angezeigt. Diese sind im einzelnen:

`-m[nsf <name>]` Definiert den Betriebsmodus. `n` stellt die Soundkarten als Datenquelle ein, `s` aktiviert den Simulationsmodus, bei dem ein zusätzliches Fenster eingeblendet wird, über das die Position der simulierten Schallquelle eingestellt werden kann und mit `f <name>` werden die Audiodaten aus Dateien gelesen. Die Dateien für die einzelnen Kanäle müssen Raw-Format besitzen und aus vorzeichenbehaftete 16 Bit Integer-Werten bestehen. Die Namen der Dateien werden zu `name01.raw` ... `name99.raw` angenommen. Das ist auch das Format, indem der Verspurrekorder `quadrec` (siehe Abschnitt 4.8.2) seine Dateien schreibt. Die Standardeinstellung ist der Betrieb mit Soundkarten.

`-r <rate>` Legt die Abtastrate in Hz fest, mit der die Daten von den Soundkarten, aus einer Datei oder in der Simulation gelesen werden. Wird dieser Parameter nicht angegeben, so wird eine Abtastrate von 48000 Hz angenommen.

`-o` Zeigt das Oszilloskop (siehe Abschnitt 4.6.2). Dieses stellt den aktuellen Signalverlauf aller verwendeten Kanäle in der jeweiligen Mikrofonfarbe dar. Die Triggerung erfolgt immer auf den ersten negativ-zu-positiven Nulldurchgang von Kanal A. Da das Oszilloskop eine nicht vernachlässigbare Menge an Rechenzeit benötigt wird es standardmäßig nicht angezeigt.

`-l` Schaltet die Aussteuerungsanzeige (siehe Abschnitt 4.6.1) ein. Diese zeigt für jeden verwendeten Kanal die logarithmierte Signalleistung an. Erreicht der Balken den roten Bereich, so ist die Signalleistung bereits so hoch, dass der Wertebereich der Soundkarten erschöpft ist und das Signal abgeschnitten wird. Das führt zu teilweise erheblichen Verzerrungen des Signals und sollte nach Möglichkeit im Betrieb vermieden werden. Aufgrund der logarithmierten Darstellung sollte bei angeschlossenem Verstärker bereits das Verstärkerrauschen einen sichtbaren Pegel erzeugen.

`-p` Zeigt die Mikrofonpositionen (siehe Abschnitt 4.6.3) an. Dabei werden die in in `micpos.h` angegebenen und fest eincompilierten Mikrofonpositionen maßstabsgetreu dargestellt. Die Perspektive zeigt den Blick auf die x-y-Ebene. Jedes Mikrofon wird dabei mit seiner Bezeichnung (A..Z) und seiner Farbe dargestellt, wie sie auch von der Aussteuerungsanzeige und dem Oszilloskop verwendet werden.

`-t` Aktiviert den Testmodus. Dabei wird die Schallortung deaktiviert und ein eventuell eingeschaltetes simuliertes Richtmikrofon erhält eine manuelle Richtungskontrolle.

`-b <rate>` Aktiviert das simulierte Richtmikrofon. Die Steuerinformation für das Mikrofon kommt entweder von der Schallortung und wird nur übernommen, wenn eine Sprachaktivität vorliegt oder im Testmodus von einer manuellen Steuerung. Die Ausgabedatenrate kann zwischen 8000Hz und der mit `-r` eingestellten Abtastrate liegen. Die Audiodaten des simulierten Richtmikrofons werden auf `Standard Out` geschrieben. Ihr Format ist Raw, vorzeichenbehaftete 16 Bit-Integer. Ist der ausführliche Modus aktiviert (`-c`), so wird ein Diagramm angezeigt, das mit einem Vektor die aktuelle Richtung zur Schallquelle darstellt. Dabei wird in der Darstellung nur der φ-Anteil der Richtung berücksichtigt.

`-h <angle>` Legt den horizontalen Suchwinkel (φ) fest. Dabei überstreicht die Schallortung ausgehend von der x-Achse als 0° in 4° Schritten einen Bereich von ±angle° um die z-Achse. Wird kein Wert angegeben, so werden 180° angenommen, also ein Abtasten der gesamten x-y-Ebene.

`-v <angle>` Legt den vertikalen Suchwinkel (ϑ) fest. Dabei überstreicht die Schallortung ausgehend von der x-y-Ebene als 0° in 4° Schritten einen Bereich von ±angle° um die x-y-Ebene. Diese Option kann mit der -h-Option kombiniert werden, sodass Raumausschnitte abgesucht werden können. Wird hier ein Winkel größer als null angegeben, so zeigt die grafische Ausgabe bei eingeschaltetem Verbose-Modus einen Kreis, der die Schallquelle darstellt. Seine horizontale Auslekung entspricht der Schallquellenposition in φ-Richtung, seine vertikale Auslenkung entspricht der Schallquellenposition in ϑ-Richtung. Das Diagramm ist auf den maximalen Suchwinkel normiert. Ist der vertikale Suchwinkel null, so zeigt das Diagramm die Richtung zur Schallquelle als Vektor entsprechend der Darstellung der Mikrofonpositionsanzeige. Der Standardwert für diese Einstellung ist 0°.

`-c` Aktiviert den ausführlichen Modus. Für Sprachortung, Sprachaktivitätsdetektion und simuliertes Richtmikrofon werden die zugehörigen Diagramme angezeigt.

`-a` Aktiviert den Messmodus. Dabei wird die φ-Richtung, in welche das simulierte Richtmikrofon zeigt im Gradmaß für eine spätere Verwertung auf `Standard Error` ausgegeben.

Anhang B

Headerfiles

In diesem Kaptiel sind die Schnittstellendeklarationen aller Programmmodule in Form der Headerdateien des Haptprogramms zusammengefasst. Dabei sind nur die als `public` deklarierten Schnittstellen und Slots, sowie die Signals angegeben.

B.1. Datei `lib/micpos.h`

```
typedef struct {
    double x,y,z;
} Tripel;

#define MICNUM 4
Tripel micpos[MICNUM]={{57.8,100.0,-40.9},{57.8,-100.0,-40.9},
                       {-115.7,0.0,-40.9},{0,0,122.5}};
```

B.2. Datei `lib/soundlow.h`

```
class SoundCard
{
 public:
    SoundCard(int numberOfCards, int desiredRate, int desiredBufferSize);
    ~SoundCard();
    void go();
```

```
    void read(signed short *data);
};
```

B.3. Datei project/meter.h

```
class Meter : public QFrame
{
Q_OBJECT
 public:
    Meter(Sound * soundApp, QWidget * parent=NULL, const char * name=NULL);
    ~Meter();
 public slots:
    void newDataReady(int channels, int frameSize, signed short** data);
};
```

B.4. Datei project/minenergy.h

```
class Minenergy : public QFrame
{
Q_OBJECT
public:
    Minenergy(int horizontalAngle, int verticalAngle, bool verboseMode,
              Sound * soundApp, QWidget * parent=NULL, char * name=NULL);
    ~Minenergy();
public slots:
    void newDataReady(int channels, int frameSize, signed short** data);
signals:
    void sweepTo(double theta, double phi);private:
    SoundReceiver * receiver;
    QPixmap * pm;
    int sampleRate;
    int pos;
    double t,p;
    int hAngle, vAngle;
    bool verbose;
```

```
};
```

B.5. Datei project/oszi.h

```
class Oszi : public QFrame
{
Q_OBJECT
 public:
    Oszi(Sound * soundApp, QWidget * parent=NULL, char * name=NULL);
    ~Oszi();
 public slots:
    void newDataReady(int channels, int frameSize, signed short** data);
};
```

B.6. Datei project/outputplugin.h

```
class SweepControl : public QLabel
{
Q_OBJECT
 public:
    SweepControl(QWidget * parent=NULL, const char * name=NULL);
    ~SweepControl();
 public slots:
    void phiBarChanged(int p);
    void thetaBarChanged(int t);
 signals:
    void sweepTo(double theta, double phi);
};

class OutputPlugin : public QFrame
{
Q_OBJECT
 public:
    OutputPlugin(bool measmode, int newRate, bool verbosemode,
                 Sound * soundApp);
```

```
    ~OutputPlugin();
 public slots:
    void newDataReady(int channels, int frameSize, signed short** data);
    void sweepTo(double theta, double phi);
    void speechActivityChanged(bool active);
};
```

B.7. Datei project/showmicpos.h

```
class ShowMicPos : public QFrame
{
Q_OBJECT
 public:
    ShowMicPos(QWidget * parent=NULL, const char * name=NULL);
    ~ShowMicPos();
};
```

B.8. Datei project/speechactivity.h

```
class SpeechActivity : public QFrame
{
Q_OBJECT
 public:
    SpeechActivity(bool verbosemode, Sound * soundApp, QWidget * parent=NULL,
                const char * name=NULL);
    ~SpeechActivity();
    void paintEvent(QPaintEvent *);
 public slots:
    void newDataReady(int channels, int frameSize, signed short** data);
 signals:
    void speechActivityChanged(bool);
};
```

Anhang C

Datenblätter

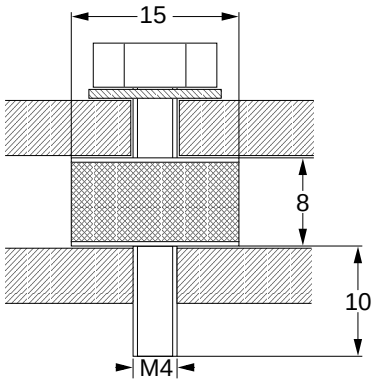
C.1. Kondensatormikrofon EB-5400

Die verwendeten Mikrofone sind Elektret-Kondensatormikrofone der Firma Teisco Sound Research LTD (http://www.teisco.com). Aufgrund der schlechten Lesbarkeit des Datenblattes, welches eine Reproduktion verhindert, seinen hier nur die wichtigsten Kenndaten angegeben:

Abmessungen:	zylindrisch, Ø 5 x 2,7 mm
Empfindlichkeit:	-65dB ±4dB (0dB=1V/μBar @ 1kHz)
Übertragungsbereich:	50 - 20.000 Hz
Ausgangswiderstand:	2kΩ max. (bei R_L= 2,2kΩ)
Richtcharakteristik:	omnidirektional
Betriebsspannungsbereich:	1 - 10 V (3V)
Betriebsstrom:	250μA max.
Maximaler Schalldruck:	110dB
Signal-Rauschabstand:	besser als 38dB

C.2. Dämpfer

Die verwendeten Dämpfer sind Schwingmetall-Puffer. Sie haben einen Durchmesser von 15mm, eine Höhe von 8mm und weisen eine Härte von 40° Shore (siehe [4], Seite 147) auf .


15
8
10
M4

C.3. Verstärkertransistor BC 847 C

NPN general purpose transistors BC846; BC847; BC848

FEATURES

- Low current (max. 100 mA)
- Low voltage (max. 65 V).

APPLICATIONS

- General purpose switching and amplification.

DESCRIPTION

NPN transistor in a SOT23 plastic package.
PNP complements: BC856; BC857; BC858.

MARKING

TYPE NUMBER	MARKING CODE	TYPE NUMBER	MARKING CODE
BC846	1Dp	BC847C	1Gp
BC846A	1Ap	BC848	1Mp
BC846B	1Bp	BC848A	1Jp
BC847	1Hp	BC848B	1Kp
BC847A	1Ep	BC848C	1Lp
BC847B	1Fp		

PINNING

PIN	DESCRIPTION
1	base
2	emitter
3	collector

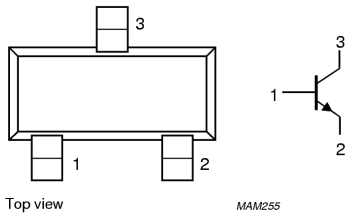


Fig.1 Simplified outline (SOT23) and symbol.

QUICK REFERENCE DATA

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter			
	BC846		–	80	V
	BC847		–	50	V
	BC848		–	30	V
V_{CEO}	collector-emitter voltage	open base			
	BC846		–	65	V
	BC847		–	45	V
	BC848		–	30	V
I_{CM}	peak collector current		–	200	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C	–	250	mW
h_{FE}	DC current gain	I_C = 2 mA; V_{CE} = 5 V			
	BC846		110	450	
	BC847; BC848		110	800	
f_T	transition frequency	I_C = 10 mA; V_{CE} = 5 V; f = 100 MHz	100	–	MHz

CHARACTERISTICS

T_j = 25 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
I_{CBO}	collector cut-off current	I_E = 0; V_{CB} = 30 V	–	–	15	nA
		I_E = 0; V_{CB} = 30 V; T_j = 150 °C	–	–	5	µA
I_{EBO}	emitter cut-off current	I_C = 0; V_{EB} = 5 V	–	–	100	nA
h_{FE}	DC current gain	I_C = 10 µA; V_{CE} = 5 V; see Figs 2, 3 and 4				
	BC846A; BC847A; BC848A		–	90	–	
	BC846B; BC847B; BC848B		–	150	–	
	BC847C; BC848C		–	270	–	
h_{FE}	DC current gain	I_C = 2 mA; V_{CE} = 5 V; see Figs 2, 3 and 4				
	BC846		110	–	450	
	BC847; BC848		110	–	800	
	BC846A;BC847A; BC848A		110	180	220	
	BC846B; BC847B; BC848B		200	290	450	
	BC847C; BC848C		420	520	800	
V_{CEsat}	collector-emitter saturation voltage	I_C = 10 mA; I_B = 0.5 mA	–	90	250	mV
		I_C = 100 mA; I_B = 5 mA	–	200	600	mV
V_{BEsat}	base-emitter saturation voltage	I_C = 10 mA; I_B = 0.5 mA; note 1	–	700	–	mV
		I_C = 100 mA; I_B = 5 mA; note 1	–	900	–	mV
V_{BE}	base-emitter voltage	I_C = 2 mA; V_{CE} = 5 V; note 2	580	660	700	mV
		I_C = 10 mA; V_{CE} = 5 V; note 2	–	–	770	mV
C_c	collector capacitance	I_E = i_e = 0; V_{CB} = 10 V; f = 1 MHz;	–	2.5	–	pF
f_T	transition frequency	I_C = 10 mA; V_{CE} = 5 V; f = 100 MHz;	100	–	–	MHz
F	noise figure	I_C = 200 µA; V_{CE} = 5 V; R_S = 2 kΩ; f = 1 kHz; B = 200 Hz	–	2	10	dB

Notes

1. V_{BEsat} decreases by about 1.7 mV/K with increasing temperature.
2. V_{BE} decreases by about 2 mV/K with increasing temperature.

C.4. Operationsverstärker TL 074

TL071, TL071A, TL071B, TL072
TL072A, TL072B, TL074, TL074A, TL074B
LOW-NOISE JFET-INPUT OPERATIONAL AMPLIFIERS

SLOS080D – SEPTEMBER 1978 – REVISED AUGUST 1996

- **Low Power Consumption**
- **Wide Common-Mode and Differential Voltage Ranges**
- **Low Input Bias and Offset Currents**
- **Output Short-Circuit Protection**
- **Low Total Harmonic Distortion 0.003% Typ**
- **Low Noise V_n = 18 nV/$\sqrt{Hz}$ Typ at f = 1 kHz**
- **High Input Impedance . . . JFET Input Stage**
- **Internal Frequency Compensation**
- **Latch-Up-Free Operation**
- **High Slew Rate . . . 13 V/μs Typ**
- **Common-Mode Input Voltage Range Includes V_{CC+}**

description

The JFET-input operational amplifiers in the TL07_ series are designed as low-noise versions of the TL08_ series amplifiers with low input bias and offset currents and fast slew rate. The low harmonic distortion and low noise make the TL07_ series ideally suited for high-fidelity and audio preamplifier applications. Each amplifier features JFET inputs (for high input impedance) coupled with bipolar output stages integrated on a single monolithic chip.

The C-suffix devices are characterized for operation from 0°C to 70°C. The I-suffix devices are characterized for operation from −40°C to 85°C. The M-suffix devices are characterized for operation over the full military temperature range of −55°C to 125°C.

electrical characteristics, $V_{CC\pm}$ = ±15 V (unless otherwise noted)

PARAMETER		TEST CONDITIONS†	T_A‡	TL071C TL072C TL074C			TL071AC TL072AC TL074AC			TL071BC TL072BC TL074BC			TL071I TL072I TL074I			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
V_{IO}	Input offset voltage	V_O = 0, R_S = 50 Ω	25°C		3	10		3	6		2	3		3	6	mV
			Full range			13			7.5			5			8	
α_{VIO}	Temperature coefficient of input offset voltage	V_O = 0, R_S = 50 Ω	Full range		18			18			18			18		μV/°C
I_{IO}	Input offset current	V_O = 0	25°C		5	100		5	100		5	100		5	100	pA
			Full range			10			2			2			2	nA
I_{IB}	Input bias current§	V_O = 0	25°C		65	200		65	200		65	200		65	200	pA
			Full range			7			7			7			20	nA
V_{ICR}	Common-mode input voltage range		25°C	±11	−12 to 15		±11	−12 to 15		±11	−12 to 15		±11	−12 to 15		V
V_{OM}	Maximum peak output voltage swing	R_L = 10 kΩ	25°C	±12	±13.5		±12	±13.5		±12	±13.5		±12	±13.5		V
		R_L ≥ 10 kΩ	Full range	±12			±12			±12			±12			
		R_L ≥ 2 kΩ		±10			±10			±10			±10			
A_{VD}	Large-signal differential voltage amplification	V_O = ±10 V, R_L ≥ 2 kΩ	25°C	25	200		50	200		50	200		50	200		V/mV
			Full range	15			25			25			25			
B_1	Unity-gain bandwidth		25°C		3			3			3			3		MHz
r_i	Input resistance		25°C		10^{12}			10^{12}			10^{12}			10^{12}		Ω
CMRR	Common-mode rejection ratio	V_{IC} = V_{ICR}min, V_O = 0, R_S = 50 Ω	25°C	70	100		75	100		75	100		75	100		dB
k_{SVR}	Supply-voltage rejection ratio ($\Delta V_{CC\pm}/\Delta V_{IO}$)	V_{CC} = ±9 V to ±15 V, V_O = 0, R_S = 50 Ω	25°C	70	100		80	100		80	100		80	100		dB
I_{CC}	Supply current (each amplifier)	V_O = 0, No load	25°C		1.4	2.5		1.4	2.5		1.4	2.5		1.4	2.5	mA
V_{O1}/V_{O2}	Crosstalk attenuation	A_{VD} = 100	25°C		120			120			120			120		dB

† All characteristics are measured under open-loop conditions with zero common-mode voltage unless otherwise specified.
‡ Full range is T_A = 0°C to 70°C for TL07_C,TL07_AC, TL07_BC and is T_A = −40°C to 85°C for TL07_I.
§ Input bias currents of a FET-input operational amplifier are normal junction reverse currents, which are temperature sensitive as shown in Figure 4. Pulse techniques must be used that maintain the junction temperature as close to the ambient temperature as possible.

C.5. Spannungswandler NMA0505S

newport components

NMA SERIES

Isolated 1W Dual Output

features

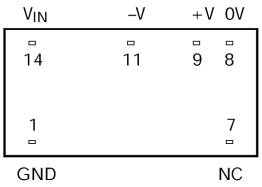
- Dual Output from a Single Input Rail
- Pin Compatible with NMH
- Industry Standard Pinout
- Power Sharing on Output
- 1kVDC Isolation
- SIP & DIP Package Styles
- Efficiency to 80%
- Power Density 0.85W/cm^3
- 3.3V, 5V, 12V, 24V & 48V Input
- 5V, 9V, 12V and 15V Output
- Footprint from 1.17cm^2
- UL 94V-0 Package Material
- No Heatsink Required
- Internal SMD Construction
- Toroidal Magnetics
- Fully Encapsulated
- No External Components Required
- MTTF up to 2.1 Million hours
- PCB Mounting
- Custom Solutions Available

description

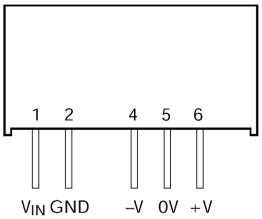
The NMA series of DC-DC converters are the standard building blocks for on-board distributed power systems. They are ideally suited to providing dual rail supplies on primarily digital boards with the added benefit of galvanic isolation to reduce switching noise. All of the rated power may be drawn from a single pin provided the total load does not exceed 1 Watt.

pin connections

14 Pin DIP (top view)



7 Pin SIP



C.6. NAND-Schmitt-Trigger CD4093

MOTOROLA
SEMICONDUCTOR TECHNICAL DATA

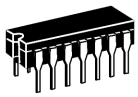
MC14077B — See Page 6-160
MC14078B, MC14081B, MC14082B — See Page 6-5

MC14093B

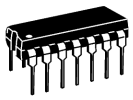
Quad 2-Input "NAND" Schmitt Trigger

The MC14093B Schmitt trigger is constructed with MOS P–channel and N–channel enhancement mode devices in a single monolithic structure. These devices find primary use where low power dissipation and/or high noise immunity is desired. The MC14093B may be used in place of the MC14011B quad 2–input NAND gate for enhanced noise immunity or to "square up" slowly changing waveforms.

- Supply Voltage Range = 3.0 Vdc to 18 Vdc
- Capable of Driving Two Low–Power TTL Loads or One Low–Power Schottky TTL Load Over the Rated Temperature Range
- Triple Diode Protection on All Inputs
- Pin–for–Pin Compatible with CD4093
- Can be Used to Replace MC14011B
- Independent Schmitt–Trigger at each Input



L SUFFIX
CERAMIC
CASE 632



P SUFFIX
PLASTIC
CASE 646

D SUFFIX
SOIC
CASE 751A

ORDERING INFORMATION

MC14XXXBCP	Plastic
MC14XXXBCL	Ceramic
MC14XXXBD	SOIC

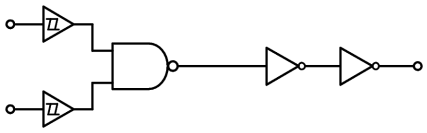
T_A = – 55° to 125°C for all packages.

MAXIMUM RATINGS* (Voltages Referenced to V_{SS})

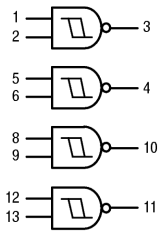
Symbol	Parameter	Value	Unit
V_{DD}	DC Supply Voltage	– 0.5 to + 18.0	V
V_{in}, V_{out}	Input or Output Voltage (DC or Transient)	– 0.5 to V_{DD} + 0.5	V
I_{in}, I_{out}	Input or Output Current (DC or Transient), per Pin	± 10	mA
P_D	Power Dissipation, per Package†	500	mW
T_{stg}	Storage Temperature	– 65 to + 150	°C
T_L	Lead Temperature (8–Second Soldering)	260	°C

* Maximum Ratings are those values beyond which damage to the device may occur.
†Temperature Derating:
Plastic "P and D/DW" Packages: – 7.0 mW/°C From 65°C To 125°C
Ceramic "L" Packages: – 12 mW/°C From 100°C To 125°C

EQUIVALENT CIRCUIT SCHEMATIC
(1/4 OF CIRCUIT SHOWN)



LOGIC DIAGRAM



V_{DD} = PIN 14
V_{SS} = PIN 7

This device contains protection circuitry to guard against damage due to high static voltages or electric fields. However, precautions must be taken to avoid applications of any voltage higher than maximum rated voltages to this high-impedance circuit. For proper operation, V_{in} and V_{out} should be constrained to the range $V_{SS} \le (V_{in}$ or $V_{out}) \le V_{DD}$.
Unused inputs must always be tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}). Unused outputs must be left open.

REV 3
1/94
© Motorola, Inc. 1995

SWITCHING CHARACTERISTICS (C_L = 50 pF, T_A = 25°C)

Characteristic	Symbol	V_{DD} Vdc	Min	Typ #	Max	Unit
Output Rise Time	t_{TLH}	5.0 10 15	— — —	100 50 40	200 100 80	ns
Output Fall Time	t_{THL}	5.0 10 15	— — —	100 50 40	200 100 80	ns
Propagation Delay Time	t_{PLH}, t_{PHL}	5.0 10 15	— — —	125 50 40	250 100 80	ns

#Data labeled "Typ" is not to be used for design purposes but is intended as an indication of the IC's potential performance.

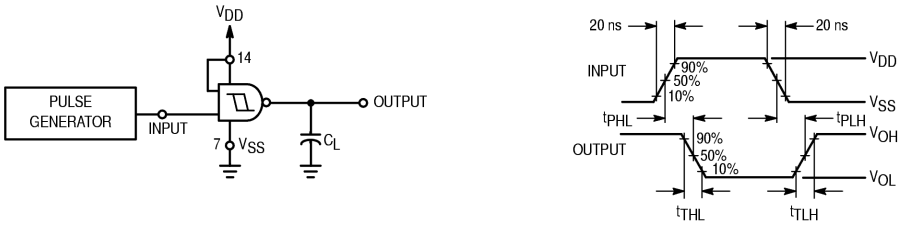
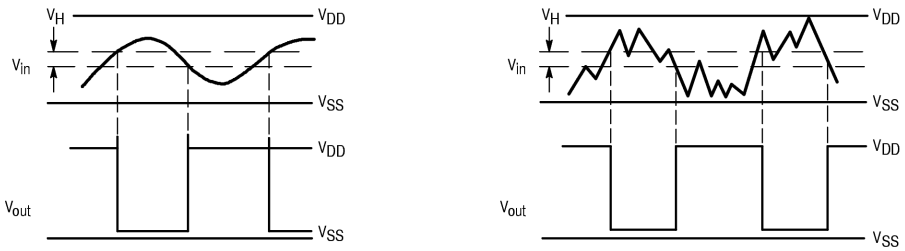


Figure 1. Switching Time Test Circuit and Waveforms



(a) Schmitt Triggers will square up inputs with slow rise and fall times.

(b) A Schmitt trigger offers maximum noise immunity in gate applications.

Figure 2. Typical Schmitt Trigger Applications