Graz University of Technology

Master Thesis

Evaluation, Simulation and Implementation of a Multi-Channel Speech Enhancement System

using the Generalized Sidelobe Canceler and recent Multi-Channel Postfilter algorithms

conducted at the
Signal Processing and Speech Communications Laboratory
Graz University of Technology, Austria

in co-operation with
Commend International
Salzburg, Austria

by
Lukas Pfeifenberger, 0831870

Supervisors:
Assoc.Prof. Dipl.-Ing. Dr. Franz Pernkopf
Univ.-Prof. Dipl.-Ing. Dr.techn. Gernot Kubin

Assessors/Examiners:
Assoc.Prof. Dipl.-Ing. Dr. Franz Pernkopf

Graz, May 7, 2013

This work was funded by the Austrian Research Promotion Agency (FFG) under grant 839587.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

date	(signature)

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am	(Unterschrift)

Acknowledgement

First and foremost, I would like to thank the founding father of the company Commend, Peter Pablik, for believing in my work and providing me with the necessary support to conduct this project. I gratefully thank Franz Pernkopf for his guidance and motivation, and most importantly for supporting me with enthusiasm throughout my thesis. I appreciatively acknowledge Univ.-Prof. Dipl.-Ing. Dr.techn. Gernot Kubin for his immense knowledge about speech processing, and for pointing me towards the fruitful direction of multichannel speech enhancement in the very early stages of this work. My sincere thanks go to Franz Zotter for providing me with some nifty Matlab scripts, which proved to be valuable for daily use. Finally, I am obliged to Christian Feldbauer for all the time he spent with constructive and excessive discussions about my scientific interests during my course of studies.

Abstract

Speech intelligibility is a paramount issue in modern telecommunication systems. In many applications, background noise is the primary source of speech degradation. For this reason, noise reduction algorithms have been studied during the last three decades. However, when using a single microphone, the problem of achieving a sufficient level of noise reduction while maintaining speech quality still exists. It is well known that beamformers can surpass this limitation by exploiting the spatial information of the sound field. Superdirective beamformers, like the *Generalized Sidelobe Canceler* (GSC), combined with a multichannel postfilter gained significance in the application area of speech enhancement due to their robustness under real-world conditions. With the advent of both affordable and powerful embedded systems, using these structures also became economically feasible. For this reason, the company Commend International conducted a research project with the goal to implement a four channel speech enhancement system. The project covers three phases: Phase one introduces the GSC, several source location algorithms and multichannel postfilters. In the second phase, the performance of these algorithms is evaluated by using psychoacoustic measures like PESQ and PEASS. These experiments are done in Matlab, using three speech databases and a variety of different noise types. The third phase covers the implementation of the resulting multichannel speech enhancement system in C++ and a live demonstration of the resulting prototype. This thesis documents each phase and its results. Therefore, it focuses not only the theory of beamforming and multichannel postfilters, but also the necessary steps to build a prototype using state of the art software development tools.

Kurzfassung

Sprachverständlichkeit ist in modernen Telekommunikationssystemen von hoher Bedeutung. Da Hintergrundgeräusche die Hauptursache einer verminderten Sprachverständlichkeit darstellen, wird an Algorithmen zur Störgeräusch-Unterdrückung bereits seit drei Jahrzehnten geforscht. Besonders im einkanaligen Fall ist das Problem der Störgeräusch-Reduktion bei gleichbleibend hoher Sprachqualität noch immer ungelöst. Beamformer können dieses Limit umgehen, indem auch die räumliche Information des Schallfeldes genutzt wird. Der *Generalized Sidelobe Canceler* (GSC) erlangt dabei unter Verwendung eines mehrkanaligen Postfilters aufgrund seiner Robustheit erhöhte Aufmerksamkeit in diesem Anwendungsgebiet. Die heutige Verfügbarkeit von preisgünstigen und leistungsstarken embedded systems ermöglicht den ökonomisch sinnvollen Einsatz dieser Algorithmen. Aus diesem Grund wurde bei Commend International ein Forschungsprojekt gestartet, dessen Ziel die Umsetzung eines vierkanaligen Beamformers ist. Das Forschungsprojekt umfasst drei Phasen: Eine Theoriephase, in welcher die Grundlagen des GSC, sowie Algorithmen zur Quellenlokalisation und Postfilterung erläutert werden. Phase zwei umfasst die psychoakustische Bewertung dieser Algorithmen unter Verwendung von PESQ und PEASS. Diese Experimente wurden in Matlab unter Verwendung von drei Sprachdatenbanken und verschiedener Hintergrundgeräusche durchgeführt. In Phase drei wird die Implementation dieser Algorithmen in Form eines Prototypen in C++, sowie dessen Demonstration in Echtzeit beschrieben. Die vorliegende Arbeit dokumentiert den Verlauf und die Ergebnisse dieser drei Phasen. Daher liegt das Hauptaugenmerk nicht nur auf der Theorie von Beamformern und mehrkanaligen Postfiltern, sondern auch auf dem Entstehungsprozess des Prototypen unter Verwendung moderner Software-Entwurfsmuster.

Contents

1 Introduction

Today, the transmission of human speech is done in a vast number of applications ranging from telecommunication devices to human-machine interfaces. Applications like mobile phones, VoIP systems, intercoms, speech recognition systems or hearing aids can be found almost everywhere. Since we live in a noisy world, these systems are exposed to all kinds of environmental noise.

Speech enhancement is concerned with improving the intelligibility and the perceptual quality of a speech signal that has been degraded by additive noise. For example, using a cellular phone inside a car or at a shopping mall imposes all kinds of background noise on the voice communication. Therefore, an improvement in both intelligibility and quality is highly desired in almost every application. Also, speech recognition systems greatly benefit from speech enhancement, as additive noise causes the word error rate to increase. Speech enhancement in telecommunication systems improves the quality of speech at the receiving end, leaving the transmitting side unaffected. Thereby, the user at the receiving end experiences the speech quality of the transmitting device. This concept still puzzles some of the most prestigious companies.

In general terms, speech enhancement methods can be classified into two main categories. The first focuses on the utilization of a single microphone while the second deals with multiple microphones. Both rely on statistical properties of human speech production, while the latter also exploits the spatial information of the sound field. Therefore, both categories are treated as separate fields throughout literature [1], [2], [3]. However, the latter can be viewed as a generalized case of the former.

1.1 Single-Channel Speech Enhancement Methods

Most speech-related applications pick up sounds using only a single microphone. Therefore, the problem of *single-channel speech enhancement* attracted the most interest amongst researchers for the last three decades. However, the problem is still not solved satisfactory, as the mixing process of speech and noise cannot be easily undone.

A rich universe of algorithms has been conceived over the years: adaptive filtering using spectral subtraction or the Wiener filter, statistical-model based methods using the ML, MAP, MMSE or Bayesian estimators, or subspace methods using SVD or EVD, to mention just a few

examples. A good overview can be found in [3] and [4].

Due to its relatively low complexity, spectral subtraction is by far the most widely used approach. This method relies on the fact that human speech is sparse in both time and frequency. The noise spectrum can be estimated during periods where the speech signal is absent. A central assumption made is that the noise spectrum is more or less stationary or at least slowly changing compared to the speech spectrum. The noise spectrum estimate is then subtracted from the spectrum of the corrupted speech signal. Only the magnitude spectrum is affected by this process, the phase information is left unchanged. Some papers in the past claimed that the phase information has little influence on the speech quality [5], while more recent ones state the opposite [6], [7].

However, the phase influence is marginal and increases the SNR by less than 2dB [7]. Estimation errors of the noise spectrum have a more severe impact on speech quality. If the noisefloor estimate is too small, some parts of the noise fail to be subtracted. These noise remnants tend to be randomly distributed over time and frequency, causing a phenomenon known as *musical artifacts*. If the noise floor estimate is too big in the other hand, spectral components of the speech might get canceled with the noise. This results in a severely deteriorated speech quality, typical to almost every commercial implementation of this algorithm. It has already been shown that noise reduction and speech quality cannot be maintained at the same time using spectral subtraction [4]. There is rather a trade-off to be chosen, which may depend on the type of noise encoutered by the application.

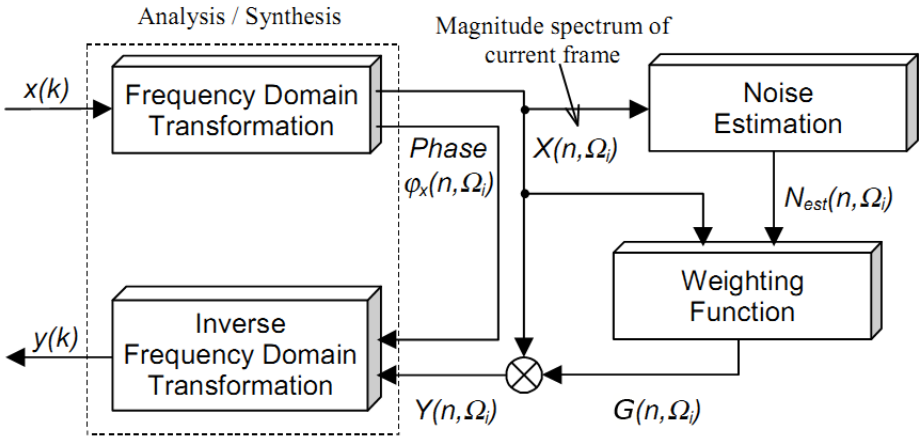

Figure 1.1: General form of the spectral subtraction algorithm. [8]

1.2 Multi-Channel Speech Enhancement Methods

When more than one microphone is available, not only the temporal information, but also the spatial features of the sound field can be utilized. This allows for richer speech enhancement concepts like *beamforming* and *source separation*. A beamformer exploits the fact that the speech and noise signals originate from different locations, which can be distinguished by the direction of the impinging sound waves. In source separation, the concept of *statistical independence* is used to distinguish between speech and noise, which are statistically independent in general. Both methods are addressed in this thesis, although the former attains more attention.

A beamformer can be thought of a spatial filter, which achieves speech enhancement by attenuating signals which do not originate from the same direction as the desired speech signal. Figure 1.2 illustrates this idea in form of a *beam pattern*. Because of the spatial information being used in this process, beamforming allows for a performance which surpasses the limitations of single channel speech enhancement methods. Conventional beamformers such as the *delay-and-sum* structure are designed to operate on a narrow frequency band, like in a radar application [9]. For speech signals, broadband or *superdirective* beamformers are used instead. They allow for two basic concepts: The first is used for blocking a signal that originates from a certain direction using the *null steering* technique. And the second one prefers a certain direction while suppressing all other directions at the same time using the adaptive *filter-and-sum* beamformer. Examples for such structures are the *Minimum Variance Distortionless Response* filter or the *Frost* beamformer. A comprehensive overview can be found in [9], [1], [10].

While these beamformers delivers excellent results in theory, in a practical implementation several robustness issues arise. Successfully steering the *beam* or *mainlobe* towards the desired speech source greatly depends on both the sensitivity and placement of the microphones being used. Implementations like the *generalized sidelobe canceler* alleviate these problems. However, spatial filtering alone cannot deliver sufficient noise suppression especially in diffuse noise fields. Therefore, a post-processing stage is used to further enhance the beamformer output. These so-called *postfilters* attracted a lot of scientific interest during the last decade. The earliest postfilters have been derived from the single-channel Wiener filter and operate as noise canceler on the output of a beamformer [2], [11]. Recent approaches also use the spatial information made available by the beamformer, and are therefore termed *multi-channel postfilters*. Examples are the *Transient Beam to Reference Ratio* [12], the *Direct to Diffuse Ratio* [13] or the *Multichannel Speech Presence Probability* [14].

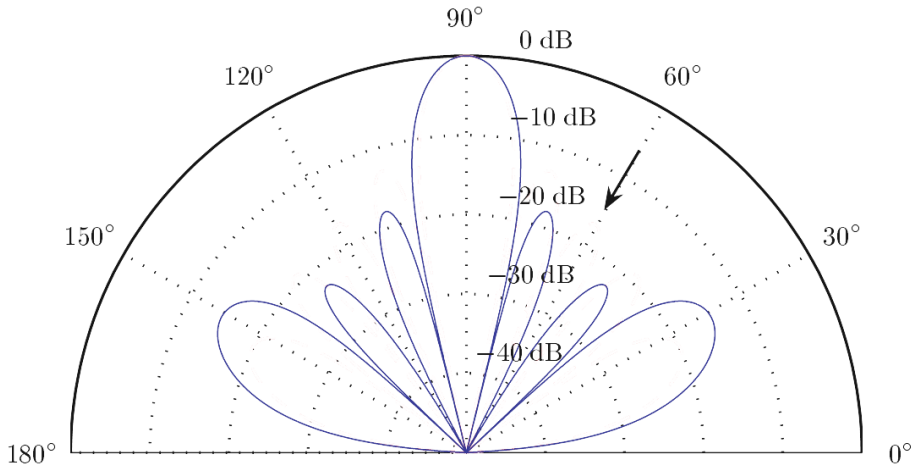

Figure 1.2: Beam pattern for a filter-and-sum *beamformer. The desired speech signal is located at 90˚, the mainlobe points toward this direction. Also, a noise source is located at 60˚, for which the beampattern shows a great attenutation due to* null steering *[9].*

1.3 Motivation and Problem Statement

During the last years embedded systems experienced a substantial increase in processing power combined with a decline in prices. As of today, the ARM Cortex-A9 MCU provides well over 4000Mflops, while complete single-board computers employing this chip start at around $40.

Therefore, multi-microphone approaches for speech enhancement gained an increasing interest in industrial and commercial applications. To advance their technological lead, the company Commend asked for an implementation of a multichannel speech enhancement algorithm running on such an embedded system. The task was defined as a research project funded by the Austrian Research Promotion Agency (FFG) for a year, yielding a prototype and this thesis as its final result. Therefore, this work not only illuminates the scientific aspects of multichannel speech enhancement systems, but also the design and implementation processes along the way.

The speech enhancement application at Commend requires a microphone array that is built into an intercom device. The physical dimensions of such a device allow for a linear array with a maximum aperture of 15cm. The array consists of up to four ECM or MEMS microphones, thus the microphones are spaced 5cm apart. A typical use case for such a device is an emergency telephone in a subway train station. Hence, the speaker is located close to the device while the ambient noise originates from farther away. For the prototype implementation, the *WandBoard* [15] using the Freescale i.MX6 Solo processor has been chosen as state of the art embedded system. It comprises a fully-featured single-board computer equipped with Gigabit LAN, USB OTG, SATA, HDMI and camera interfaces, and an audio codec. The four microphones are connected to the WandBoard using an USB soundcard.

Since the prototype is intended for a live demonstration, the vast range of speech enhancement algorithms is somewhat narrowed to the ones that operate in near real-time, which means using only small blocks of audio data at a time. Further, the used algorithms are constrained to a limited numerical complexity which allows them to be implemented on an embedded system. For its robustness and moderate complexity, the *generalized sidelobe canceler* has been chosen as broadband beamformer. To form a complete multichannel speech enhancement system, a postfilter and a source location algorithm are needed. Out of this field, the best algorithms in terms of performance, robustness and numerical complexity are evaluated and used for the final implementation.

1.4 Organization and Aim of this Work

According to the three phases of the research project, the thesis is organized in three main blocks, as shown in figure 1.3: In the *Theory* part, all concepts used in this work are presented in detail. In the *Experiments* part, the matlab experiments of the implemented methods are documented. In the *Prototype* part, the performance of C++ implementation is demonstrated.

The *Theory* phase spans the first four chapters, where chapter 2 introduces the sound wave propagation model, which is used to derive the basic principles of *direct sound fields* and *diffuse sound fields*. Their properties are used to define the signal model of the application, where the desired speech is located in the near field and the noise components are located in the far field. The signal model serves as a basis for the beamforming methods introduced in chapter 3. There, the GSC beamformer and its building blocks are described. Four different *blocking matrix* (BM) algorithms, and two adaption algorithms for the *adaptive interference canceler* (AIC) are introduced in this chapter. The beamformer needs to be steered towards the desired source. This task is termed *source location* and is covered in chapter 4. Five different source location algorithms are characterized. In an actual application, using a beamformer alone yields only a small amount of noise reduction. Additional performance is gained by using a multichannel postfilter. This relatively new field is illuminated in chapter 5 by presenting three recent approaches. In

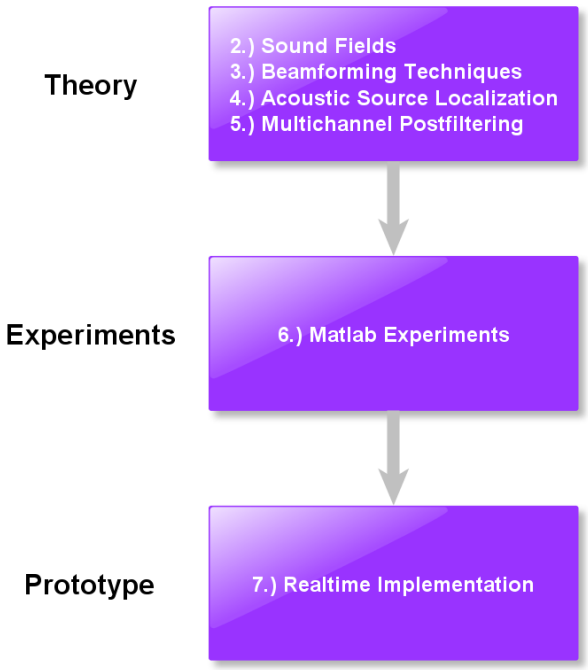

Figure 1.3: Organization of this thesis.

order to gain some savings in computational complexity, the principle and successful application of a gammatone filterbank is also shown.

The *Simulation* phase is covered in chapter 6. It deals with the evaluation of all these algorithms using three speech databases. This evaluation is done as a series of Matlab ®experiments, where the optimal combination of the algorithms is found by using two psychoacoustic quality measures. The first is the well-known PESQ score, which delivers a MOS-like score. The second one is the *Perceptual Evaluation Methods for Audio Source Separation* (PEASS) score, recently presented in [16] and [17].

The *Prototype* phase covers the real-time implementation of the optimal combination of the algorithms, documented in chapter 7. A C++ prototype has been implemented using open source tools. The verification and validation steps of the implementation compared to the Matlab simulation are shown. Finally, some performance results are presented for testing the prototype in a live scenario using noise and speech signals at the testing site at Commend headquarters.

2 Sound fields

In this chapter, wave propagation and sound fields are introduced to provide a basis for the challenges of the upcoming sections. Further, the process of measuring the relative room impulse response between two microphones is described in detail. And finally, a quick introduction to MIMO signal processing is given as a prerequisite for beamforming.

2.1 Wave Propagation

Sound waves are mechanical vibrations propagating through matter. They can be described in terms of a sound pressure field $p(\vec{\boldsymbol{r}}, t)$ and a sound velocity vector field $\vec{\boldsymbol{v}}(\vec{\boldsymbol{r}}, t)$. The sound pressure in pascals describes the pressure exerted at the medium, which is air in this case. And the sound velocity describes the velocity, at which air particles move as they transmit the sound wave. Both are functions of a spatial vector $\vec{\boldsymbol{r}} = [x, y, z]^T$ and time t.

A microphone can be thought of as a sensor, which measures sound pressure at a distinct location in space. To construct an accurate model for sound wave propagation, it is necessary to describe the sound pressure at any given point in space for a specific sound source.

Euler's equation describes the connection between sound pressure and sound velocity. The gradient of the sound pressure is proportional to the time derivative of the sound velocity [18], i.e.

$$\vec{\nabla} p = -\rho_0 \frac{\partial \vec{\boldsymbol{v}}}{\partial t}. \tag{2.1}$$

By using

$$\vec{\nabla} \vec{\boldsymbol{v}} = -\frac{1}{\rho_0 c^2} \frac{\partial p}{\partial t} \tag{2.2}$$

a second order differential equation, known as the acoustic wave equation, can be formulated

$$\Delta p = \vec{\nabla}^T \vec{\nabla} p = \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2}. \tag{2.3}$$

Where ρ_0 is the density of the medium at rest, and c denotes the speed of sound. In cartesian coordinates, the Laplacian represents

$$\Delta \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}. \tag{2.4}$$

A homogeneous solution for the differential equation in 2.3 can be found to be [18]

$$p(\vec{\boldsymbol{r}}, t) = \hat{p} \mathrm{e}^{j\vec{\boldsymbol{k}}^T \vec{\boldsymbol{r}} - j\omega t}, \tag{2.5}$$

where $\vec{\boldsymbol{k}} = [k_x, k_y, k_z]^T$ is the *wave number* for each dimension, and $\hat{p}$ is the amplitude. The wave number or propagation constant [19] measures wavelengths per meter, and is defined as

$$k = \frac{\omega}{c} = \frac{2\pi}{\lambda},$$

where $\omega = 2\pi f$, λ is the wavelength, and c is the speed of sound in air, being approximately $343 \frac{m}{s}$ at $20°C$.

Equation 2.5 can be recognized as a harmonic plane wave [19], traveling into the direction of $\vec{\boldsymbol{r}}$. Clearly, the source of such a plane wave is located infinitely far away. Therefore, this mode is only sufficient if the distance from the sound source to the microphone is large enough. Then, the microphone is said to be in the *farfield* of the source. For the *nearfield* however, a spherical wave model might be used instead. Spherical waves imply spherical surfaces of constant sound pressure. This means, that the wave fronts are concentric spheres originating from a very small point source, where this sound pressure is generated.

Using polar coordinates, equation 2.3 can be written as

$$\frac{\partial^2 p}{\partial r^2} + \frac{2}{r} \frac{\partial p}{\partial r} = \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2}, \tag{2.6}$$

with r being the radius from the origin. For the sound pressure of a spherical wave field, a simple solution to equation 2.6 is given in [19], i.e.

$$p(r, t) = \frac{j\omega\rho_0}{4\pi r} \hat{Q} \mathrm{e}^{j\omega t - jkr}, \tag{2.7}$$

where $\hat{Q}$ in $\left[\frac{m^3}{s}\right]$ is the peak *volume velocity* of the point source, at which matter (air) is expelled in order to generate the sound pressure $p(r, t)$. It is worth noting that the sound pressure diminishes with increasing distance from the source r in a linear manner. Further, the

sound velocity (or particle velocity) is given as

$$v(r,t) = \frac{p(r,t)}{\rho_0 c}\Big(1 + \frac{1}{jkr}\Big). \tag{2.8}$$

The ratio between $p(r,t)$ and $v(r,t)$ is known as the *impedance* of the medium (air), and equates to

$$\frac{p(r,t)}{v(r,t)} = \frac{\rho_0 c}{1 + \frac{1}{jkr}}. \tag{2.9}$$

In the farfield of the sound source, where $kr \gg$ this ratio tends asymptotically to $\rho_0 c$. This value is known as the *characteristic impedance*. For air at $20°C$, it is 416 $\left[\frac{Pa}{ms^{-1}}\right]$.

In the nearfield of the sound source, when either the distance r or the wave number k is small, the ratio is complex, meaning that sound pressure and sound velocity are not in phase any more.

2.2 Spatial Sampling of Sound Fields

The most general scenario, where spherical sound waves apply is given in figure 2.1. Here, M microphones are located in the nearfield of a single point sound source $\boldsymbol{s_m}$. For simplicity, the microphones are considered to be omnidirectional sound pressure receivers with perfect linearity and no additive system noise. In this scenario, each microphone is located at a distance $\|\boldsymbol{r_m}\|$ from an arbitrary coordinate origin $[\boldsymbol{e_x}, \boldsymbol{e_y}, \boldsymbol{e_z}]^T$. The point source is located at $\|\boldsymbol{r_s}\|$.

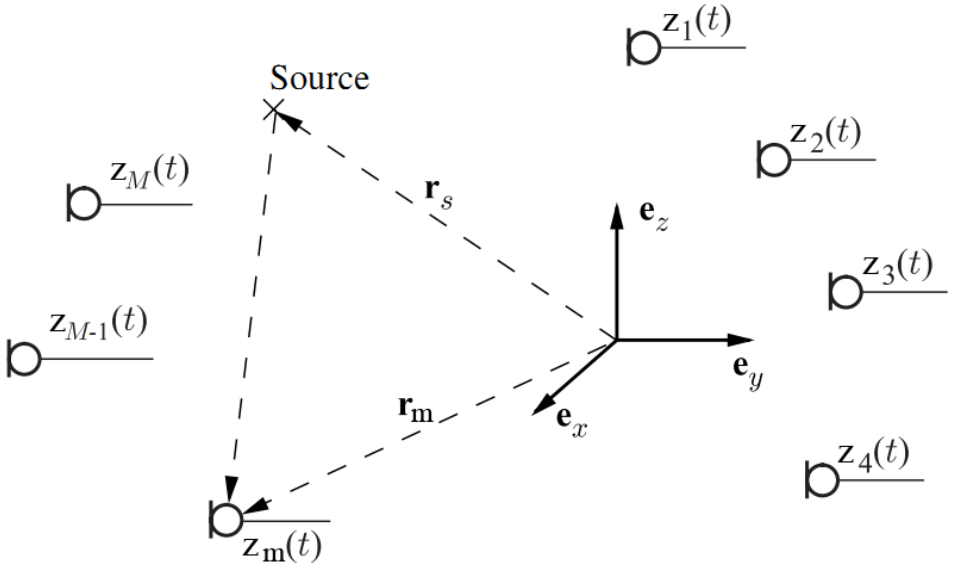

Figure 2.1: Spatial sampling of one source with M microphones, taken from [2].

According to the result from equation 2.7, the sound pressure at distance r is attenuated by $\frac{1}{r}$, and delayed in time by kr. Using $s(t)$ for the signal at the source, the output z_m of the m-th

microphone can be written as

$$z_m(t) = \frac{1}{\|\boldsymbol{r_m} - \boldsymbol{r_s}\|} s\Big(t - \frac{\|\boldsymbol{r_m} - \boldsymbol{r_s}\|}{c}\Big) \tag{2.10}$$

If the Fourier transform of $s(t)$ exists, equation 2.10 can be written as

$$Z_m(j\Omega) = \frac{1}{\|\boldsymbol{r_m} - \boldsymbol{r_s}\|} S(j\Omega) \mathrm{e}^{j2\pi f \frac{\|\boldsymbol{r_m} - \boldsymbol{r_s}\|}{c}}. \tag{2.11}$$

By combining the attenuation and the time delay into the frequency-dependent factor $A_m(j\Omega)$, equation 2.11 simplifies to

$$Z_m(j\Omega) = A_m(j\Omega) S(j\Omega). \tag{2.12}$$

Furthermore, introducing vector notation all M microphone signals can be expressed as

$$\boldsymbol{Z}(j\Omega) = \boldsymbol{A}(j\Omega) S(j\Omega), \tag{2.13}$$

with

$$\boldsymbol{Z}(j\Omega) = \begin{bmatrix} Z_1(j\Omega) & Z_2(j\Omega) & \cdots & Z_M(j\Omega) \end{bmatrix}^T,$$

and

$$\boldsymbol{A}(j\Omega) = \begin{bmatrix} A_1(j\Omega) & A_2(j\Omega) & \cdots & A_M(j\Omega) \end{bmatrix}^T.$$

Thereby, the vector $\boldsymbol{Z}(j\Omega)$ denotes the *spatial images* of the source signal $S(j\Omega)$. It contains the source signal, spatially sampled at the locations $\boldsymbol{r_1} \cdots \boldsymbol{r_M}$ for each frequency $j\Omega$.
For a real-world scenario although, this propagation model is too simple as it does not consider more complicated acoustic phenomena like reverberation and noise. Therefore, the model is modified to include these effects in the next chapter.

2.3 Signal Model

The given speech enhancement application assumes a microphone array that is built into an intercom device. Based on existing requirements, the following three restrictions are defined:

- The speaker is located close to the microphones when speaking, usually less than 0.5m.
- Due to the physical dimensions of an intercom device, the array aperture is small. Based on aforementioned constraints, the inter-microphone distance is 5cm.
- The ambient noise originates from outside the critical distance [19] of the enclosing room, i.e. the noise is modeled as diffuse sound.

Based on these conditions, the spherical wave propagation model from section 2.1 is extended to the following signal model: The user or speaker is modeled as the source signal $s(t)$, and is located

in a noisy and reverberant environment. Further, a microphone array of $m = 1 \dots M$ sensors is located close to the source. For simplicity, the array consists of linearly spaced microphones. Each microphone is modeled as omnidirectional sound pressure receiver with equal sensitivity over its entire frequency range. Each microphone picks up a signal $z_m(t)$ which contains a reverberated image of the source $s(t)$, and an additive noise component. Both the source and the noise signals are of non-stationary nature, since their content is unknown. Due to the small distance to the source origin, the microphones pick up mainly direct sound from the source signal with only little reverberation. On the contrary, the noise picked up by each microphone is mainly a diffuse sound, since we assume that its source is located outside the critical distance of the setup. Therefore, there is no distinct noise origin in this signal model [19]. The complete signal model is shown in figure 2.2.

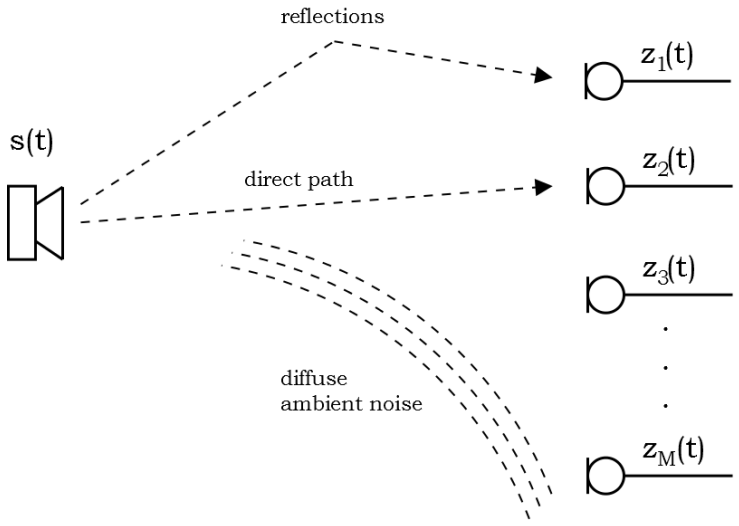

Figure 2.2: Signal model.

The acoustic path from the source to each microphone is known as *Acoustic Transfer Function* (ATF). It models the filter $a_m(t)$ which specifies the reverberated source signal received by the m-th microphone. The signal $z_m(t)$ received at each microphone is an additive mixture of the reverberated source signal and the interfering noise. The source and the interfering noise signals are assumed to be uncorrelated. This results in the following signal model

$$z_m(t) = a_m(t) * s(t) + n_m(t),$$

which can be expressed in the fourier-domain as

$$Z_m(j\Omega) = A_m(j\Omega)S(j\Omega) + N_m(j\Omega),$$

or in a more compact vector notation, covering all M microphones:

$$\boldsymbol{Z}(j\Omega) = \boldsymbol{A}(j\Omega)S(j\Omega) + \boldsymbol{N}(j\Omega). \tag{2.14}$$

The vector $\boldsymbol{A}$ contains the acoustic transfer function from the source towards the m-th microphone. Therefore it fully describes both the spatial layout and the frequency response from the source towards the array. The elements of $\boldsymbol{A}$ depict a spatial filter, while the dependency on $j\Omega$ indicates a frequency filter.

2.4 Acoustic Transfer Function

In the presented signal model, the acoustic paths from the source to the microphones are described as linear, time-invariant FIR filters

$$\boldsymbol{A}(j\Omega) = \begin{bmatrix} A_1(j\Omega) & A_2(j\Omega) & \cdots & A_M(j\Omega) \end{bmatrix}^T .$$

Each filter $A_m(j\Omega)$ depicts the room impulse response measured from the source's location to the m-th microphone. In this work, the room impulse response is referred to as *acoustic transfer function* (ATF).

A typical ATF can be quite long if the enclosing room has a long reverberation time. An example taken from [1] is shown in figure 2.3.

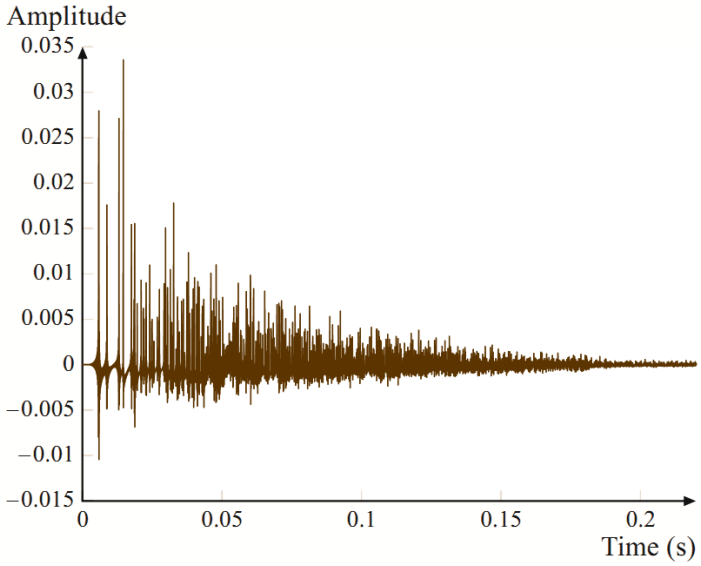

Figure 2.3: Typical ATF, taken from [1].

It can be seen that the ATF consists of two parts: The leftmost, highest spikes represent the direct sound propagation path along with early reflections from nearby objects. The second part consists of random reflections and late arrivals, which overlap to the so-called *tail*. Clearly, a long ATF consisting of thousands of taps is hard to estimate, especially when the excitation signal $s(t)$ is unknown.

2.5 Relative Transfer Function

A practical method to overcome the problem of estimating a long ATF is to use the *Relative Transfer Function* (RTF) between two microphones. According to [20], RTFs are much shorter than ATFs and hence easier to estimate. The RTFs $\tilde{\boldsymbol{A}}(j\Omega)$ are defined as the ratio between the ATFs and a reference ATF

$$\tilde{\boldsymbol{A}}(j\Omega) = \begin{bmatrix} 1 & \frac{A_2(j\Omega)}{A_1(j\Omega)} & \frac{A_3(j\Omega)}{A_1(j\Omega)} & \cdots & \frac{A_M(j\Omega)}{A_1(j\Omega)} \end{bmatrix}^T . \tag{2.15}$$

Here, the propagation path to the first microphone has been used as reference ATF.
There are two things to be considered when using RTFs: First, the ATFs may have zeros outside the unit circle, since they are not necessarily minimum phase. Hence, the RTFs are non-causal filters which have to be modeled with negative time delays to ensure stability. And this time delay needs to be long enough to allow the FIR assumption to be met.

Second, the RTFs do not model the propagation path to the source, but rather to the reference microphone. This means that a beamformer which uses the RTFs cannot estimate the source signal $s(t)$, but rather the reverberated source image at the first microphone $s_1(t) = a_1(t) * s(t)$. However, if the reverberation time of the setup is small, this is not a problem.

2.6 Spatial Coherence

The intended use case for this multichannel speech enhancement project is hands-free communication. Target applications would be emergency telephones, ticket booths and office intercoms, as well as toll stations or help points. In virtually all of these situations, the speaker is located in the direct vicinity of the device, whereas the interfering noise source is more remote. Therefore, the microphone array almost certainly encounters a directional sound field for the speaker, and a diffuse noise field. Directional sound impings only from a single direction or point source, whereas diffuse sound arrives from all directions equally strong, and has therefore no distinct origin. For applications such as car interiors or offices, the noise field can even be modeled by an ideal spherically isotropic noise field [21] [13] [1].

The spatial correlation of a sound field is characterized by the complex cross-coherence function. Between the i$^{\text{th}}$ and j$^{\text{th}}$ microphone, the coherence is given as

$$\gamma_{Z_i Z_j}(j\Omega) = \frac{\Phi_{Z_i Z_j}(j\Omega)}{\sqrt{\Phi_{Z_i Z_i}(j\Omega)\Phi_{Z_j Z_j}(j\Omega)}}, \tag{2.16}$$

where $\Phi_{Z_i Z_j}(j\Omega)$ denotes the cross-spectrum of $Z_i(j\Omega)$ and $Z_j(j\Omega)$. Assuming equal power levels at each microphone, the ideal spherical diffuse noise sound field features a purely real coherence function [19], determined as

$$\gamma_{N_i N_j}(j\Omega) = \frac{\sin(kd_{ij})}{kd_{ij}}, \tag{2.17}$$

where $k = \frac{\omega}{c}$ is the wave number and d_{ij} the distance between microphone i and j. In contrast, an ideal directional sound field is fully coherent towards the direction of the impinging sound waves. Hence, the coherence contains the direction of the source as phase information Θ_s, and is given as

$$\gamma_{S_i S_j}(j\Omega) = \mathrm{e}^{j\Theta_s \Omega}. \tag{2.18}$$

Assuming an ideal spherical diffuse sound field for noise and an ideal directional sound field for speech has the advantage of known spatial coherence functions for both cases. Otherwise the underlying correlation matrices for noise $\Phi_{NN}(j\Omega)$ and speech $\Phi_{SS}(j\Omega)$ have to be estimated, which is cumbersome in practice.

Another useful measure in spatial signal processing is the squared coherence [10]. It is a real

number between 0 and 1 and indicates the normalized correlation between the i-th and j-th microphone. It is given as

$$|\gamma_{Z_i Z_j}(j\Omega)|^2 = \frac{|\Phi_{Z_i Z_j}(j\Omega)|^2}{\Phi_{Z_i Z_i}(j\Omega)\Phi_{Z_j Z_j}(j\Omega)}. \tag{2.19}$$

Inserting into equations 2.17 and 2.18, the squared coherence of the ideal diffuse and the ideal directional sound fields turn out to be

$$|\gamma_{N_i N_j}|^2(j\Omega) = \frac{\sin(kd_{ij})^2}{k^2 d_{ij}^2}, \tag{2.20}$$

and

$$|\gamma_{S_i S_j}|^2(j\Omega) = 1, \tag{2.21}$$

respectively. Especially the result of equation 2.21 is expected, because an ideal directional sound field is also fully coherent.

2.7 ATF Measurement

Even though the assumptions made in section 2.3 are already experimentally proven in [20] and [19], it is advantageous to know to which extent they apply to a real-world scenario. Further, it is necessary to ensure that the simplifications made for the spatial coherence of the noise and speech sound fields in section 2.6 are valid for the given application. Therefore, the ATFs, RTFs and spatial coherences have been measured using the following setup: A linear microphone array and a measurement loudspeaker have been placed in a typical office room of about 4x6m. Hence the ATFs can be measured from the loudspeaker's position to the microphone array's position. The array consists of four *Audix TM1 Plus* measurement microphones, each 5cm apart. To measure the speaker's ATFs, the loudspeaker is placed at a distance of 0.5m in front of the array to attain mainly direct sound. To measure the noise ATFs, the speaker is placed outside the critical distance of 3m to get mainly diffuse sound. Figure 2.4 shows the measurement setup in detail. Green arrows depict direct sound, red ones diffuse sound.

Measuring the ATF is identical to measuring the room impulse response, which is a common task in acoustics engineering. A good overview is given in [22] and [23]. Most methods use a measurement pulse or waveform to excite the unknown system. The measured signal is the impulse response of the unknown system convoluted with the excitation waveform. By using a matched filter, the impulse response can be obtained through deconvolution. Commonly used excitation signals are linear or exponential sine sweeps, dirac-like pulses or pseudo-random sequences. For this task, a *maximum length sequence* (MLS) has been chosen as excitation waveform for the following reasons:

- The matched filter for the MLS is easily found, it is the time-reversed MLS.
- No additional low-pass or band-pass filtering is needed after deconvolution, since all frequencies are equally well excited.

- The crest factor for an MLS is 3dB lower than for a sine, hence a better SNR can be achieved.
- Generating an MLS is extremely simple using a linear feedback shift register (LFSR).

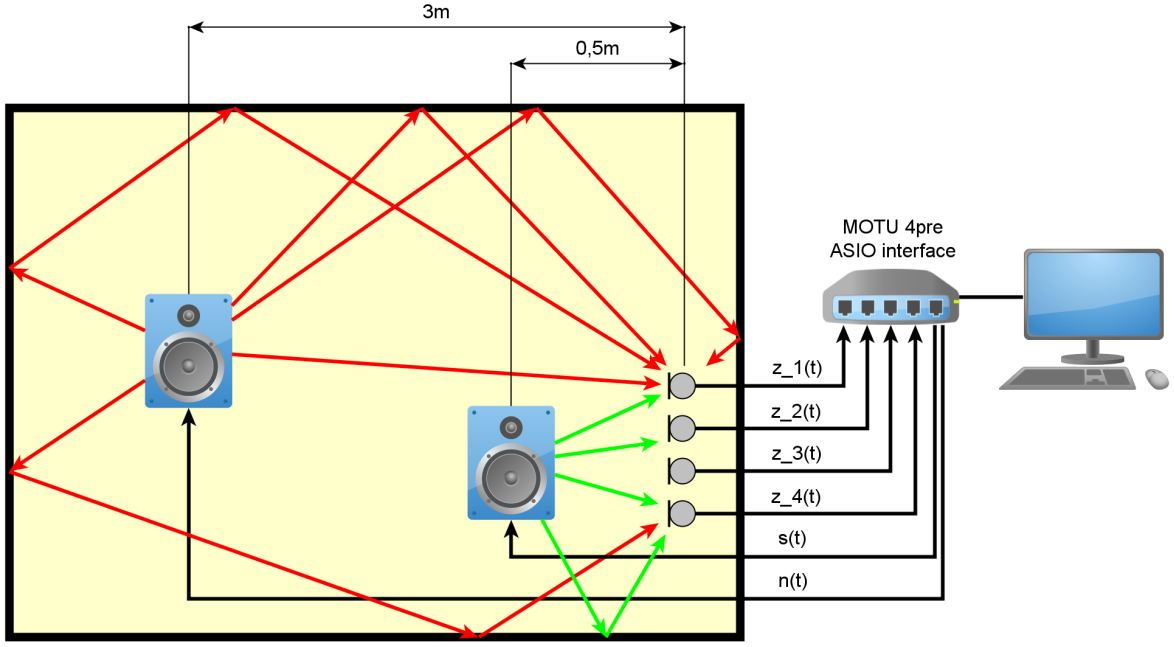

Figure 2.4: ATF measurement setup: The loudspeaker at the 3m position produces a diffuse sound field at the array, while the loudspeaker in the 0.5m position produces an almost ideal directional sound field with only little reverberation.

Measuring the impulse response of an unknown system with an MLS is described in [24]. The measuring procedure works as follows: The loudspeaker plays the MLS $S(j\Omega)$, the microphones pick up the signals

$$Z_m(j\Omega) = A_m(j\Omega)S(j\Omega).$$

Side effects of measurement noise are intentionally omitted, for simplicity. The recorded signal $Z_m(j\Omega)$ is then convolved with the time-reversed MLS to obtain an estimate of the ATF $A_m(j\Omega)$, which simplifies to a multiplication with the conjugated MLS in the frequency domain

$$\hat{A}_m(j\Omega) = Z_m(j\Omega)S^*(j\Omega).$$

One of the remarkable features of an MLS is the fact that its magnitude spectrum is 1 over the entire frequency range, and its phase spectrum is pseudo-random. Therefore, a convolution of the MLS $S(j\Omega)$ with its own time-reverse $S^*(j\Omega)$ yields a dirac pulse $\delta(0)$ at time-lag zero. This relation renders the estimate of the ATF to

$$\hat{A}_m(j\Omega) = A_m(j\Omega)S(j\Omega)S^*(j\Omega) = A_m(j\Omega).$$

For this experiment, a sampling frequency of 48kHz has been used. The echo tail is expected to be no longer than 0.5s, which requires the MLS to run for at least 1s in order to circumvent the

convolution effects of the FFT. Valid lengths of MLSs are $2^n - 1$. At the sampling frequency given, the nearest n would be 16. The generator polynomial used for the MLS is: $p_{16}(x) = x^{12} + x^3 + x + 1$.

Since this method requires a fully linear system, nonlinearities in the loudspeaker or the microphones distort the measured impulse response. The same is true for measurement noise. To avoid this, a highly linear measurement equipment has been used. For replaying the MLS, a *Geithain RL906* measurement loudspeaker is used. The ATFs are recorded using the four *Audix TM1 Plus* measurement microphones in a linear array.

Using the setup in figure 2.4, two sets of ATFs have been measured: First, the loudspeaker has been placed 3m away from the array. This setup simulates an ambient noise source located outside of the critical distance of the room. We obtain mainly a diffuse sound. Second, the loudspeaker has been placed 0.5m away from the array. This setup simulates a speaker close to the array, providing mainly directional sound. Figure 2.5 shows the measured ATF for the noise setup. Panel (a) shows the ATF in the time domain. It can be seen that the echo tail is almost 0.5s long. Panel (b) and (c) show the magnitude and phase spectra, respectively. Figure 2.6 shows the measured ATF for the directional sound setup. It can be seen that the ATF is shorter if the loudspeaker is placed closer to the array. This behavior is expected. It is interesting to note that the phase spectrum shown in panel (c) is almost linear, while the phase was randomly distributed for the noise ATF. This indicates that a single peak dominates the ATF. In fact, only little reverberation can be identified in this ATF. Figure 2.7 shows the calculated RTF between microphone 1 and 2 for the noise setup. The FFT length has been set to 100ms, leaving a 50ms window for both the causal and non-causal part of the RTF. Even though this is already a large window size the RTF does not decay, and thereby violates the FIR assumption. Since there is no distinct peak visible, this RTF contains mainly diffuse sound. Figure 2.8 shows the calculated RTF between microphone 1 and 2 for the directional sound setup. Here, the RTF quickly decays, as mentioned in [20]. There is a dominant peak observable at time lag 0s, indicating mainly directional sound in this ATF. The small peak at about 0.004s indicates an obstacle at about 1m distance. However, this peak is small enough to model the RTF as a delay-only filter. Figure 2.9 shows the squared coherence between microphone 1 and 2 for the noise setup. The squared coherence is similar to the squared coherence of the ideal diffuse sound field in equation 2.20. Especially for frequencies below 2000Hz a high coherence (and thus correlation) between the microphones can be observed. It can be concluded that this room can produce an almost ideal diffuse sound field. Figure 2.10 shows the squared coherence between microphone 1 and 2 for the directional sound setup. The squared coherence is strong throughout all frequencies. Again, this coincides with the ideal case of a directional sound field in equation 2.21.

This experiment has been repeated with two other office rooms of about 2.5x4m and 6x8m in size. Furthermore, the distance for the noise setup has been varied in the range of 2 to 6m. The results are similar to the ones presented above. From the shape of the observed RTFs and squared coherences it can be concluded, that modeling the speech signal as direct sound and the ambient noise as diffuse sound is valid for this scenario.

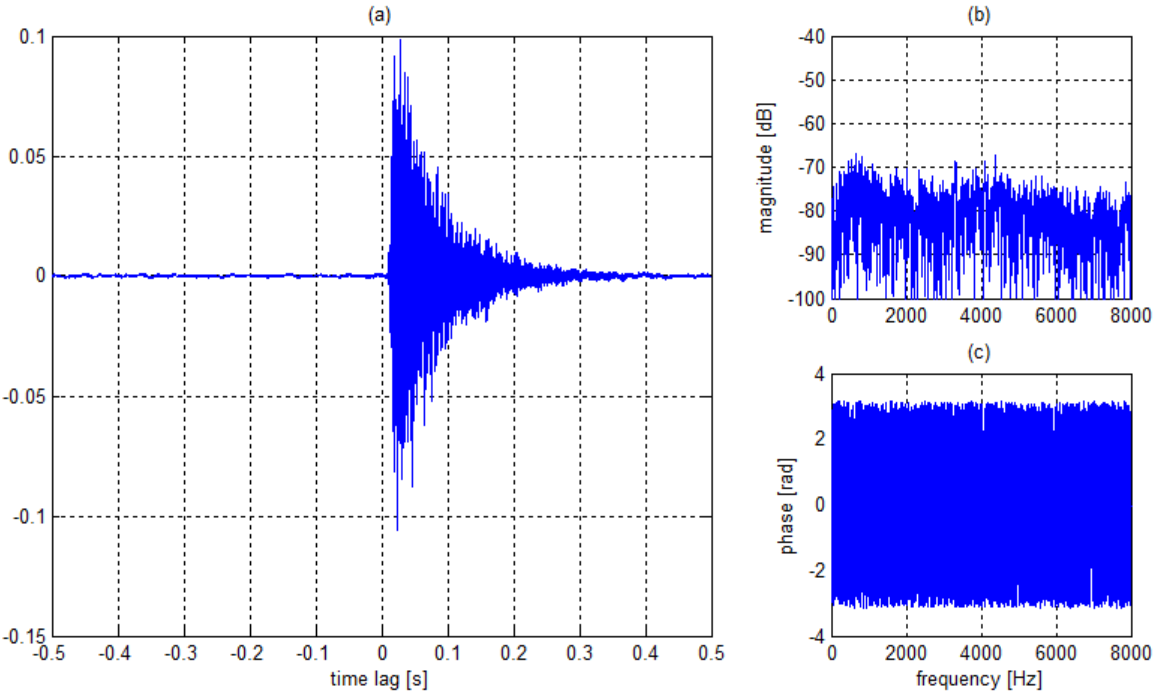

Figure 2.5: ATF measurement for mic1 from 3m distance. The EIR is over 0.3s long. (a) time domain, (b) magnitude spectrum, (c) phase spectrum.

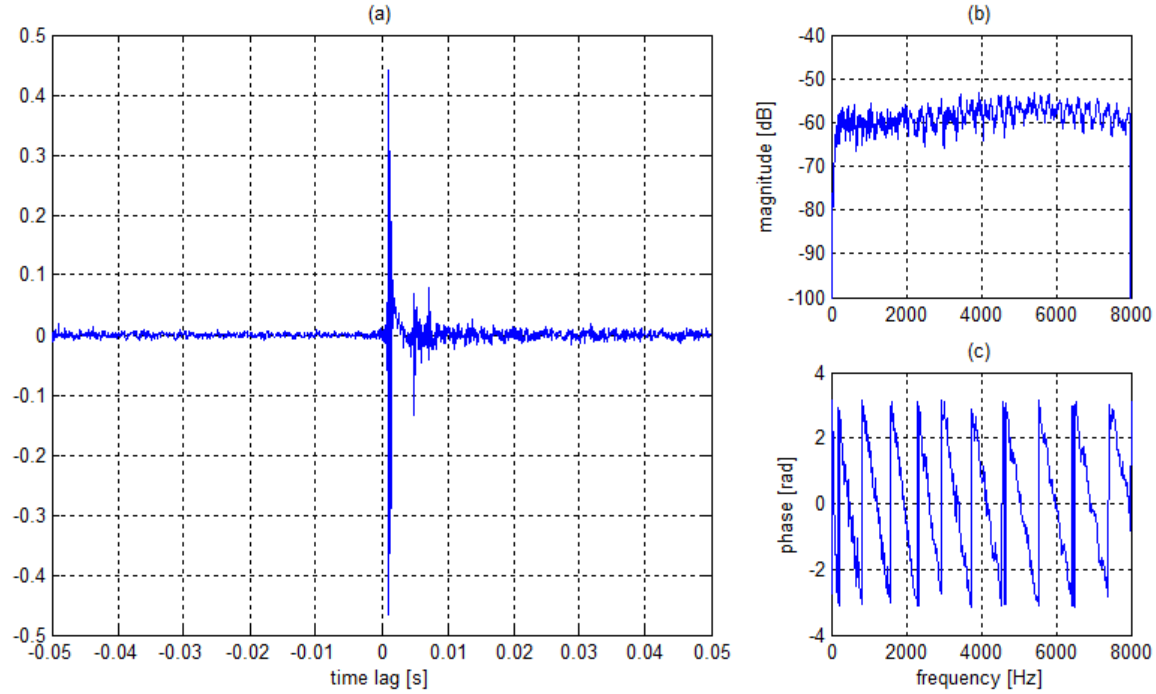

Figure 2.6: ATF measurement for mic1 from 0.5m distance. The small peaks at about 0.04s indicate some reverberation. (a) time domain, (b) magnitude spectrum, (c) phase spectrum.

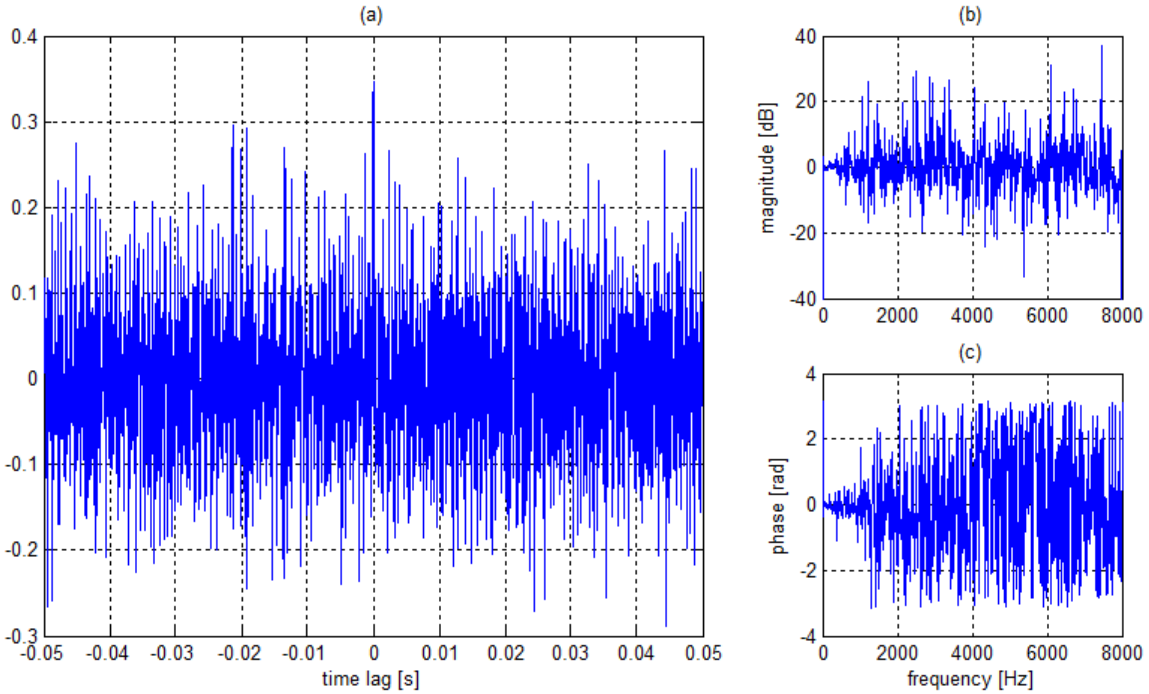

Figure 2.7: RTF between mic1 and 2 from 3m distance. The FIR constraint is clearly not met for this observation window. (a) time domain, (b) magnitude spectrum, (c) phase spectrum.

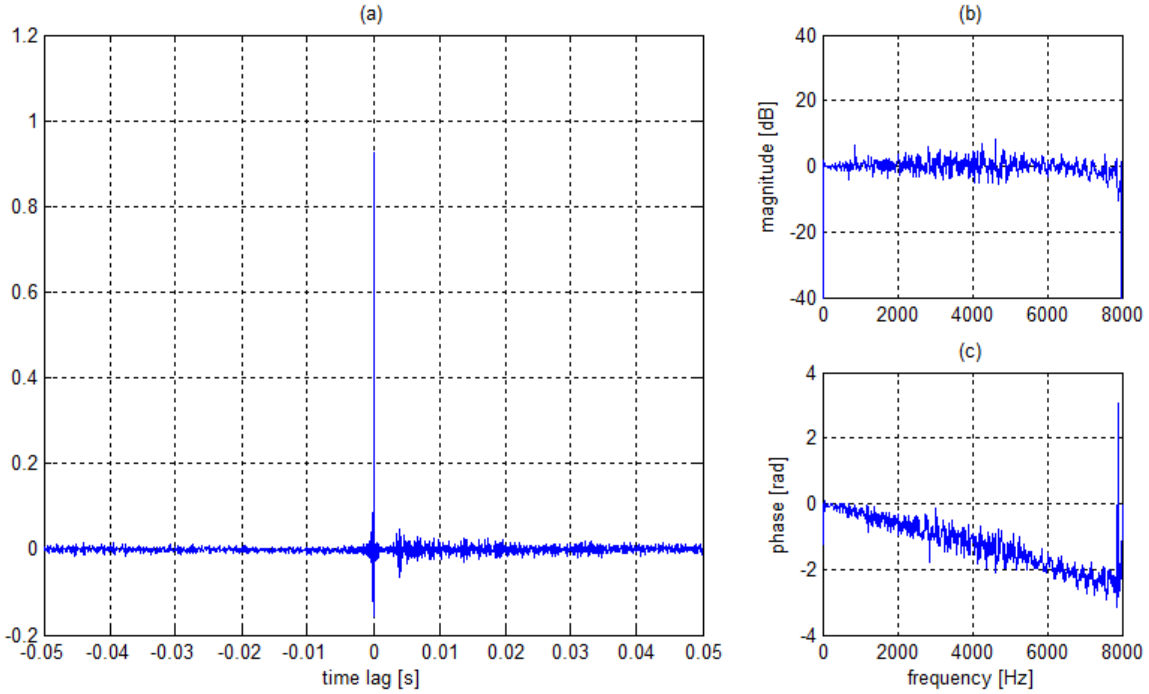

Figure 2.8: RTF between mic1 and 2 from 0.5m distance. The single, dominant peak indicates an almost perfect directional sound field. (a) time domain, (b) magnitude spectrum, (c) phase spectrum.

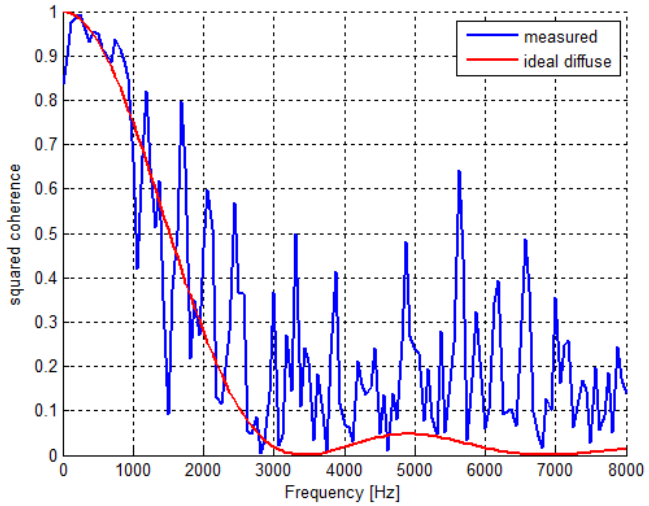

Figure 2.9: Coherence between mic1 and 2 from 3m distance.

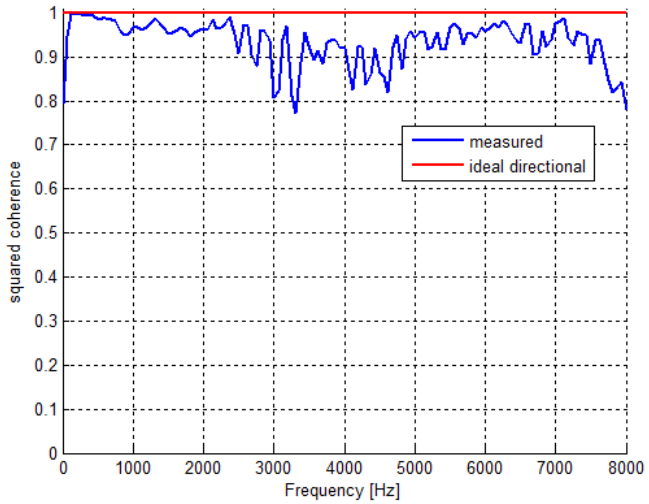

Figure 2.10: Coherence between mic1 and 2 from 0.5m distance.

2.8 MIMO Systems

Sound fields are both time and space dependent. With a single sensor, only temporal sampling can be done. When multiple sensors are available, also spatial sampling is possible. A beamformer can be thought of a digital filter which operates in both domains. The most general model for such a filter is a *multiple input – multiple output* or MIMO system. It models M' inputs $s_{m'}(t)$ and M outputs $z_m(t)$, where each input is connected with each output via a simple FIR filter $h_{mm'}$ with L taps. Also, an uncorrelated interference noise $n_m(t)$ is added to each of the outputs, as shown in figure 2.11. In this example, the system of coupled ATFs $h_{mm'}$ builds the MIMO system. The ATFs are the acoustic paths between the loudspeakers and the microphones. Therefore, the inputs of the MIMO system are the loudspeaker signals $s_{m'}(t)$ and its outputs are the microphone signals $z_m(t)$. This might sound confusing, but it is correct from the MIMO system's perspective.

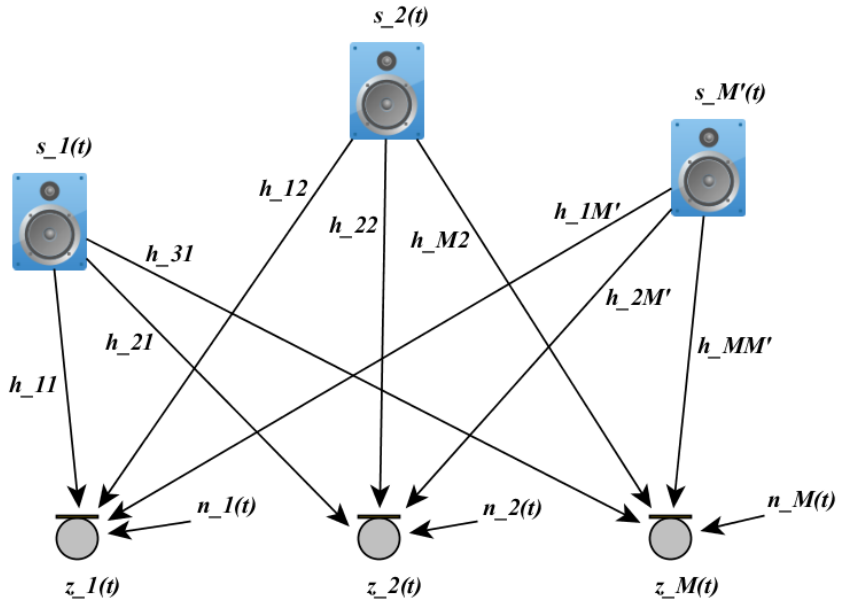

Figure 2.11: Acoustical MIMO system.

The interferences $n_m(t)$ could be ambient noise or sensor noise. To estimate the filters in this scenario, the well-known Wiener filter can be extended from the *single input – single output* or SISO case to the MIMO case [25]. By defining the filter coefficients of the ATFs $h_{mm'}$ and the sampled signal $s_{m'}(k)$ of the m-th sensor as vectors

$$\begin{aligned}\boldsymbol{h}_{mm'} &= \begin{bmatrix} h_{mm',0} & h_{mm',1} & \cdots & h_{mm',L-1} \end{bmatrix}^T \\ \boldsymbol{s_{m'}}(k) &= \begin{bmatrix} s_{m'}(k) & s_{m'}(k-1) & \cdots & s_{m'}(k-L+1) \end{bmatrix}^T, \end{aligned}$$

a single output $z_m(k)$ can be written as a MISO system

$$z_m(k) = \sum_{m'=1}^{M'} \boldsymbol{h}_{mm'}^T \boldsymbol{s}_{m'}(k) + n_m(k) = \boldsymbol{h}_{m:}^T \boldsymbol{s}(k) + n_m(k), \tag{2.22}$$

where the notation in $\boldsymbol{h}_{m:}$ is borrowed from Matlab and denotes a $M'L \times 1$ stacked column vector

$$\boldsymbol{h}_{m:} = \begin{bmatrix} \boldsymbol{h}_{m1'}^T & \boldsymbol{h}_{m2'}^T & \cdots & \boldsymbol{h}_{mM'}^T \end{bmatrix}^T.$$

Also, all M' blocks of input signals $s_{m'}$ are concatenated to a $M'L \times 1$ column vector

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{s}_1^T & \boldsymbol{s}_2^T & \cdots & \boldsymbol{s}_{M'}^T \end{bmatrix}^T.$$

By further combination of the filters $\boldsymbol{h}_{mm'}$ into a compact $M \times M'$ matrix

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{h}_{11}^T & \boldsymbol{h}_{12}^T & \cdots & \boldsymbol{h}_{1M'}^T \\ \boldsymbol{h}_{21}^T & \boldsymbol{h}_{22}^T & \cdots & \boldsymbol{h}_{2M'}^T \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{h}_{M1}^T & \boldsymbol{h}_{M2}^T & \cdots & \boldsymbol{h}_{MM'}^T \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}_{1:}^T \\ \boldsymbol{h}_{2:}^T \\ \vdots \\ \boldsymbol{h}_{M:}^T \end{bmatrix},$$

N MISO systems can be written as a single MIMO system

$$\boldsymbol{z}(k) = \boldsymbol{H}\boldsymbol{s}(k) + \boldsymbol{n}(k). \tag{2.23}$$

With the system estimate $\hat{\boldsymbol{H}}$, an error signal vector $\boldsymbol{e}(k)$ can be defined as

$$\boldsymbol{e}(k) = \boldsymbol{z}(k) - \hat{\boldsymbol{H}}\boldsymbol{s}(k) = \sum_{m=1}^{M} e_m(k), \tag{2.24}$$

consisting of M error signals

$$e_m(k) = z_m(k) - \hat{\boldsymbol{h}}_{m:}^T \boldsymbol{s}(k). \tag{2.25}$$

The MIMO mean squared error (MSE) or cost function is then defined as

$$J(\hat{\boldsymbol{H}}) = \mathbb{E}\{\boldsymbol{e}^T(k)\boldsymbol{e}(k)\} = \sum_{m=1}^{M} \mathbb{E}\{e_m^2(k)\} = \sum_{m=1}^{M} J_m(\hat{\boldsymbol{h}}_{m:}), \tag{2.26}$$

i.e. it can be decomposed into M independent cost functions [25]

$$\begin{aligned} J_m(\hat{\boldsymbol{h}}_{m:}) &= \mathbb{E}\{[z_m(k) - \hat{\boldsymbol{h}}_{m:}^T \boldsymbol{s}(k)][z_m(k) - \hat{\boldsymbol{h}}_{m:}^T \boldsymbol{s}(k)]^T\} \\ &= \mathbb{E}\{z_m(k) z_m^T(k) - z_m(k)\boldsymbol{s}^T(k)\hat{\boldsymbol{h}}_{m:} - \hat{\boldsymbol{h}}_{m:}^T \boldsymbol{s}(k) z_m^T(k) + \hat{\boldsymbol{h}}_{m:}^T \boldsymbol{s}(k)\boldsymbol{s}(k)^T \hat{\boldsymbol{h}}_{m:}\} \\ &= \sigma_{z_m}^2 - 2\hat{\boldsymbol{h}}_{m:}^T \boldsymbol{p}_{sz_m} + \hat{\boldsymbol{h}}_{m:}^T \boldsymbol{R}_{ss} \hat{\boldsymbol{h}}_{m:} \,, \end{aligned} \tag{2.27}$$

where $\boldsymbol{R}_{ss}$ is the $M'L \times M'L$ input signal covariance matrix

$$\boldsymbol{R}_{ss} = \mathbb{E}\{\boldsymbol{s}(k)\boldsymbol{s}(k)^T\} = \begin{bmatrix} \boldsymbol{R}_{s_1 s_1} & \boldsymbol{R}_{s_1 s_2} & \cdots & \boldsymbol{R}_{s_1 s_{M'}} \\ \boldsymbol{R}_{s_2 s_1} & \boldsymbol{R}_{s_2 s_2} & \cdots & \boldsymbol{R}_{s_2 s_{M'}} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{R}_{s_{M'} s_1} & \boldsymbol{R}_{s_{M'} s_2} & \cdots & \boldsymbol{R}_{s_{M'} s_{M'}} \end{bmatrix}, \tag{2.28}$$

which has a block Toeplitz structure. The symbol $\sigma_{z_m}^2$ denotes the variance of the output signal $z_m(k)$, and $\boldsymbol{p}_{sz_m}$ is the cross-correlation vector between all inputs $\boldsymbol{s}(k)$ and $z_n(k)$

$$\boldsymbol{p}_{sz_m} = \mathbb{E}\{\boldsymbol{s}(k) z(k)^T\}. \tag{2.29}$$

The optimal system estimate $\hat{\boldsymbol{H}}_{opt}$ with respect to the mean squared error (MSE) criterion is

found by evaluating the minimum of the cost function in equation 2.26 as shown in [26]. Taking the gradient of the convex cost function $J(\cdot)$ with respect to all $\hat{\boldsymbol{h}}_{mm'}^T$ and equating the result to zero yields

$$\nabla J = \frac{\partial J(\hat{\boldsymbol{H}})}{\partial \hat{\boldsymbol{H}}^T} = \sum_{m=1}^{M} \frac{\partial J_m(\hat{\boldsymbol{h}}_{m:})}{\partial \hat{\boldsymbol{h}}_{m:}^T} = \sum_{m=1}^{M} [-2\boldsymbol{p}_{sz_m} + 2\boldsymbol{R}_{ss}\hat{\boldsymbol{h}}_{m:}] \stackrel{!}{=} 0, \tag{2.30}$$

which can easily be identified as the sum of M independent MISO Wiener-Hopf equations. Thus, any MIMO Wiener filter can be decomposed into M independent MISO Wiener filters. The MSE-optimal MISO Wiener filter is given by solving equation 2.30, leading to

$$\hat{\boldsymbol{h}}_{m:,opt} = \boldsymbol{R}_{ss}^{-1}\boldsymbol{p}_{sz_m}. \tag{2.31}$$

This solution is not very practical, because it requires the inverse of $\boldsymbol{R}_{ss}$. Since this matrix is of size size $M'L \times M'L$, its inversion has a computational complexity of $\mathcal{O}(M'^3L^3)$. For long filters, where L reaches several thousand taps, this inversion virtually impossible. Therefore, adaptive algorithms like the normalized least mean squares (NLMS) algorithm or one of its many variants presented in [25] are used in general. This well-known SISO NLMS algorithm can easily be extended to the MISO case. The only difference is that M' filters $\hat{\boldsymbol{h}}_{m:,opt}$ are adapted simultaneously by using the common error signal $e_m(k)$ from equation 2.25. In [25], the MISO NLMS algorithm is given as

$$\hat{\boldsymbol{h}}_{m:}(k) = \hat{\boldsymbol{h}}_{m:}(k-1) + \mu\frac{\boldsymbol{s}(k)e_m(k)}{\boldsymbol{s}^T(k)\boldsymbol{s}(k) + \delta} \tag{2.32}$$

where $e_m(k)$ is the sum of the error signals from all M' filter estimates $\hat{\boldsymbol{h}}_{m:'}$ defined in equation 2.25. The *learning rate* μ and the regularization constant δ have to be chosen according to $0 < \mu < 2$, and $\delta > 0$.

The NLMS algorithm minimizes the MISO cost function $J_m(\hat{\boldsymbol{h}}_{m:})$, and thereby identifies M' filters $\hat{\boldsymbol{h}}_{m:}$ simultaneously. As shown, applying the algorithm to all M independent MISO systems independently is equivalent to minimizing the MIMO cost function $J(\hat{\boldsymbol{H}})$.
Because of the filter operations in equation 2.25 this algorithm has a quadratic complexity of $\mathcal{O}(M'^2L^2)$. While this is already much better than cubic complexity, it can even be further reduced when using the frequency domain representation of the MISO NLMS algorithm.

In the following, we examine the frequency domain (FD) solution of the MISO Wiener filter for two reasons: First, using the FD representation of FIR filters reduces the numerical complexity from $\mathcal{O}(M'^2L^2)$ to $\mathcal{O}(M'L\log(M'L))$. Second, non-causal filter structures are required when using RTFs, as shown. Basically, filters with negative time delays could also be designed using bulk delays, as shown in [27]. But then an extra delay is necessary each time a non-causal filter is used, and these delays sum up quickly when there are many signal processing stages like beamformer, postfilter, echo canceler and noise canceler.
Since low processing delays are important in real-time applications like hands-free telephony, recent approaches like the *Subband Feedback Controlled Generalized Sidelobe Canceler* [28] are using noncausal filters for all stages, i.e. the same delay is reused.

The signal model for a MISO system from equation 2.22 can easily be transformed into the frequency domain by using the overlap-add method and windowing. By defining a stride of L samples and an overlap factor of 50% (for simplicity) we obtain frames of length $2L$. A windowed

block of the output signal z_m is given as

$$\tilde{z}_m(k-l) = z_m(k-l)w(l), \tag{2.33}$$

where the window function

$w = \begin{bmatrix} w_0 & w_1 & \cdots & w_{2L-1} \end{bmatrix}^T$ minimizes aliasing which would otherwise occur due to circular convolution effects of the FFT. Most implementations use the hanning window. In vector notation, the $2L \times 2L$ DFT matrix $\boldsymbol{\mathcal{F}}$ defines the frequency domain representation of a block of the output signal z_m

$$Z_m(j\Omega) = \boldsymbol{\mathcal{F}}\tilde{z}_m. \tag{2.34}$$

The MISO system from equation 2.22 is transformed to the frequency domain as

$$\begin{aligned} Z_m(j\Omega) &= \sum_{m'=1}^{M'} H_{mm'}^H(j\Omega)S_{m'}(j\Omega) + N_m(j\Omega) \\ &= \boldsymbol{H}_{m:}^H(j\Omega)\boldsymbol{S}(j\Omega) + N_m(j\Omega). \end{aligned} \tag{2.35}$$

With the frequency-domain representation of the error signal in equation 2.25

$$E_m(j\Omega) = Z_m(j\Omega) - \boldsymbol{H}_{m:}^H(j\Omega)\boldsymbol{S}(j\Omega),$$

the cost function evaluates to

$$\begin{aligned} J_m(\hat{\boldsymbol{H}}_{m:}^H(j\Omega)) &= \mathbb{E}\{|E_m(j\Omega)|^2\} \\ &= \mathbb{E}\{[Z_m(j\Omega) - \hat{\boldsymbol{H}}_{m:}^H(j\Omega)\boldsymbol{S}(j\Omega)][Z_m(j\Omega) - \hat{\boldsymbol{H}}_{m:}^H(j\Omega)\boldsymbol{S}(j\Omega)]^H\} \\ &= \Phi_{z_m z_m}(j\Omega) - 2\hat{\boldsymbol{H}}_{m:}^H(j\Omega)\boldsymbol{\Phi}_{sz_m}(j\Omega) + \hat{\boldsymbol{H}}_{m:}^H(j\Omega)\boldsymbol{\Phi}_{ss}(j\Omega)\hat{\boldsymbol{H}}_{m:}(j\Omega), \end{aligned} \tag{2.36}$$

where $\boldsymbol{\Phi}_{ss}(j\Omega)$ is the $M' \times M'$ input signal power-spectral density matrix, and $\boldsymbol{\Phi}_{sz_m}(j\Omega)$ is the cross-spectral density vector between all inputs and the output signal $Z_m(j\Omega)$. And $\Phi_{z_m z_m}(j\Omega)$ is the power spectrum estimate of the output signal $Z_m(j\Omega)$. Taking the derivative of the above cost function and equating it to zero gives

$$\nabla J_m = \frac{\partial J_m(\hat{\boldsymbol{H}}_{m:,opt}(j\Omega))}{\partial \hat{\boldsymbol{H}}_{m:,opt}^H(j\Omega)} = -2\boldsymbol{\Phi}_{sz_m}(j\Omega) + 2\boldsymbol{\Phi}_{ss}(j\Omega)\hat{\boldsymbol{H}}_{m:,opt}(j\Omega) \stackrel{!}{=} 0. \tag{2.37}$$

The MSE-optimal solution for the noncausal MISO Wiener filter is

$$\hat{\boldsymbol{H}}_{m:,opt}(j\Omega) = \boldsymbol{\Phi}_{ss}^{-1}(j\Omega)\boldsymbol{\Phi}_{sz_m}(j\Omega). \tag{2.38}$$

Similar as in the time domain approach, M' filters are optimized in a single step. The complexity for inverting the matrix $\boldsymbol{\Phi}_{ss}$ is significantly reduced as its size is only $M' \times M'$. While equation 2.38 provides a direct solution for the filters $\hat{\boldsymbol{H}}_{m:}(j\Omega)$ there is also an adaptive approach in form

of the frequency domain multichannel NLMS algorithm [25], i.e.

$$\hat{\boldsymbol{H}}_{m:}(j\Omega,k) = \hat{\boldsymbol{H}}_{m:}(j\Omega,k-1) + \mu\frac{\boldsymbol{S}(j\Omega,k)E_m(j\Omega,k)}{\boldsymbol{S}(j\Omega,k)^H\boldsymbol{S}(j\Omega,k)+\delta}, \tag{2.39}$$

where k denotes the signal frame, and the parameters μ and δ are selected as for the time domain MISO NLMS in equation 2.32. The term $\boldsymbol{S}(j\Omega,k)^H\boldsymbol{S}(j\Omega,k)$ in the denominator represents the combined energy of all input signal frames $S_{m'}(j\Omega,k)$ and can be easily computed using the Frobenius norm $\boldsymbol{S}(j\Omega,k)^H\boldsymbol{S}(j\Omega,k) = \|\boldsymbol{S}(j\Omega,k)\|_2^2$.

When the inputs of a MIMO system are microphones in an array structure like in figure 2.11, a remarkable observation can be made: While each single filter $\hat{\boldsymbol{H}}_{mm'}(j\Omega)$ represents the frequency response of that channel, the $M \times M'$ filter coefficients for a single frequency $j\Omega$ form a spatial filter. Hence, both a frequency filter and a spatial filter are defined by the multichannel Wiener filter $\boldsymbol{H}$. Shaping the spatial response independently of the frequency response of that filter is the main goal of adaptive beamforming.

3 Beamforming Techniques

The reception of a specific signal under the presence of interfering noise can be greatly enhanced by exploiting the spatial properties of the sound field. A beamformer can be thought of as a multivariate filter that operates on the outputs of a sensor array. Originally, beamforming was developed for radar and sonar systems. While these systems only deal with narrowband signals, speech signals span several octaves. Further, the impinging speech signal is often highly reverberated due to reflections from obstacles such as walls. Then the speech signal seemingly originates from multiple directions at once.
Hence, the most important design goal of a beamformer is to "listen" into a specific direction with a given tolerance or *beamwidth* for all frequencies in the speech band. The spatial response of a beamformer is termed *beam pattern* or *directivity pattern*. The shape of the beam pattern is determined by the physical alignment of the array, which is measured in wavelengths [9]. To obtain the same beam pattern for each frequency, the array would have to be rescaled if the wavelength changes. Since this is physically impossible, the beam pattern greatly varies with frequency, depending on the beamformer architecture. In this chapter, the most important architectures, such as the *delay-and-sum* and the *filter-and-sum* beamformer are derived as a basis for the *general sidelobe canceler*.

3.1 Basic Principle

The aim of acoustic beamforming in general and multichannel speech enhancement in particular is to estimate a desired source signal which is corrupted by reverberation, additive noise or other unwanted sound sources. To accomplish this, the beamformer builds a spatial filter which listens into the general direction of the desired source, and thereby masks out the unwanted sounds. This basic principle is depicted in figure 3.1.

In the lower part of figure 3.1 a beamforming filter is shown in its most general form. In its frequency domain representation, the output $y(k)$ of the beamforming filter can be written as

$$Y(j\Omega) = \sum_{m=1}^{M} W_m(j\Omega) Z_m(j\Omega). \tag{3.1}$$

where $Z_m(j\Omega)$ is the signal received at the m-th microphone. It contains the reverberated source

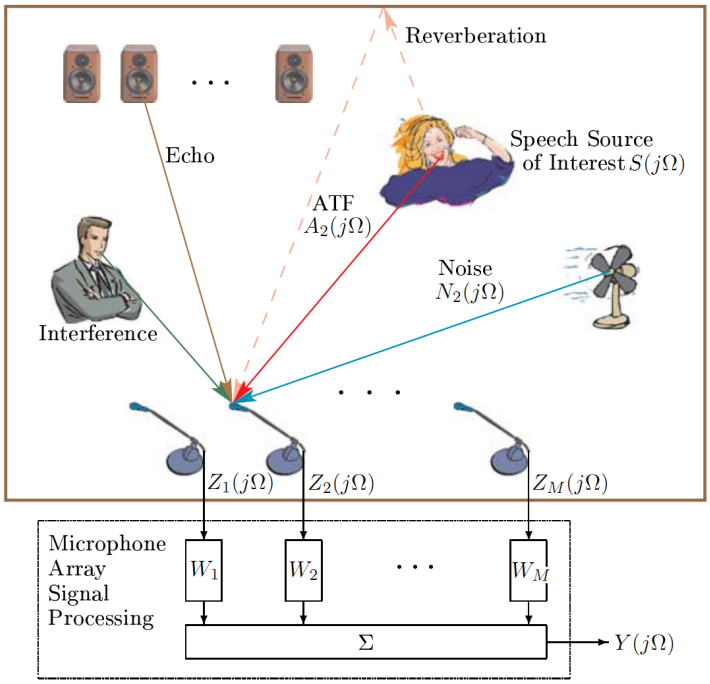

Figure 3.1: Basic principle of acoustic beamforming taken from [9].

signal $S(j\Omega)$ and the summation of all the unwanted signals denoted as $N_m(j\Omega)$

$$Z_m(j\Omega) = A_m(j\Omega)S(j\Omega) + N_m(j\Omega), \tag{3.2}$$

where $A_m(j\Omega)$ represents the ATF from the source signal to the m-th microphone. In vector notation, the microphone signals can be written as

$$\boldsymbol{Z}(j\Omega) = \boldsymbol{A}(j\Omega)S(j\Omega) + \boldsymbol{N}(j\Omega). \tag{3.3}$$

The beamformer output $Y(j\Omega)$ in Equation 3.1 is

$$Y(j\Omega) = \boldsymbol{W}^H(j\Omega)\boldsymbol{Z}(j\Omega), \tag{3.4}$$

using vector notation, where the beamforming filter is

$$\boldsymbol{W}(j\Omega) = \begin{bmatrix} W_1(j\Omega) & W_2(j\Omega) & \cdots & W_M(j\Omega) \end{bmatrix}^T.$$

This beamforming filter hast to perform two tasks: First, it has to point the so-called *looking direction* of the beamformer towards the source signal $S(j\Omega)$, in order to let it pass. Second, it should suppress the unwanted noises $N_m(j\Omega)$ as good as possible.

3.2 Performance Measures

3.2.1 Array Gain

The performance of a microphone array is often expressed as improvement of the SNR towards the desired source signal. This improvement is measured as the ratio of the SNR at the array output to the average SNR at the microphone signals. To evaluate the array gain of an arbitrary beamformer, the following procedure introduced in [2] may be used: Assuming that the desired speech and the interfering noise in equation 3.3 are mutually uncorrelated, the power spectral density of the input signals $\boldsymbol{Z}(j\Omega)$ evaluates to

$$\begin{aligned}\boldsymbol{\Phi}_{ZZ}(j\Omega) &= \mathbb{E}\{\boldsymbol{Z}(j\Omega)\boldsymbol{Z}^H(j\Omega)\} \\ &= \mathbb{E}\{[\boldsymbol{A}(j\Omega)S(j\Omega) + \boldsymbol{N}(j\Omega)][\boldsymbol{A}(j\Omega)S(j\Omega) + \boldsymbol{N}(j\Omega)]^H\} \\ &= \boldsymbol{A}(j\Omega)\Phi_{SS}(j\Omega)\boldsymbol{A}^H(j\Omega) + \boldsymbol{\Phi}_{NN}(j\Omega).\end{aligned} \tag{3.5}$$

The input SNR is defined as the ratio of the average power of the speech and the average power of the noise components

$$iSNR = \frac{\frac{1}{M}Tr(\boldsymbol{A}(j\Omega)\Phi_{SS}(j\Omega)\boldsymbol{A}^H(j\Omega))}{\frac{1}{M}Tr(\boldsymbol{\Phi_{NN}}(j\Omega))}, \tag{3.6}$$

where $Tr()$ is the trace operator. The SNR at the output $Y(j\Omega)$ can be derived by inspecting equation 3.4

$$\begin{aligned}\Phi_{YY}(j\Omega) &= \mathbb{E}\{Y(j\Omega)Y^*(j\Omega)\} \\ &= \mathbb{E}\{\boldsymbol{W}^H(j\Omega)\boldsymbol{Z}(j\Omega)\boldsymbol{Z}^H(j\Omega)\boldsymbol{W}(j\Omega)\} \\ &= \boldsymbol{W}^H(j\Omega)\boldsymbol{A}(j\Omega)\Phi_{SS}\boldsymbol{A}^H(j\Omega)\boldsymbol{W}(j\Omega) + \boldsymbol{W}^H(j\Omega)\boldsymbol{\Phi}_{NN}\boldsymbol{W}(j\Omega).\end{aligned} \tag{3.7}$$

The output SNR is defined as the ratio of the speech power and the noise power at the beamformer output

$$oSNR = \frac{\boldsymbol{W}^H(j\Omega)\boldsymbol{A}(j\Omega)\Phi_{SS}\boldsymbol{A}^H(j\Omega)\boldsymbol{W}(j\Omega)}{\boldsymbol{W}^H(j\Omega)\boldsymbol{\Phi}_{NN}\boldsymbol{W}(j\Omega)}. \tag{3.8}$$

The array gain is defined as the ratio of the output SNR and the input SNR [2]. It is given as

$$\mathcal{G}(j\Omega) = \frac{\boldsymbol{W}^H(j\Omega)\boldsymbol{A}(j\Omega)\Phi_{SS}\boldsymbol{A}^H(j\Omega)\boldsymbol{W}(j\Omega)}{\boldsymbol{W}^H(j\Omega)\boldsymbol{\Phi}_{NN}\boldsymbol{W}(j\Omega)} \frac{Tr(\boldsymbol{\Phi_{NN}}(j\Omega))}{Tr(\boldsymbol{A}(j\Omega)\Phi_{SS}\boldsymbol{A}^H(j\Omega))}. \tag{3.9}$$

If the noise signals at the microphones are completely uncorrelated, the cross-power spectra matrix $\boldsymbol{\Phi_{NN}}(j\Omega)$ turns out to be the diagonal matrix $\boldsymbol{\Phi_{NN}}(j\Omega) = \boldsymbol{\sigma}_n^2\boldsymbol{I}_{M\times M}$, where $\boldsymbol{I}_{M\times M}$ is the identity matrix. In this case, the array gain turn out to be

$$\mathcal{G}(j\Omega) = \frac{\Phi_{SS}\boldsymbol{A}(j\Omega)\boldsymbol{A}^H(j\Omega)\boldsymbol{W}(j\Omega)\boldsymbol{W}^H(j\Omega)}{\boldsymbol{\sigma}_n^2\boldsymbol{W}^H(j\Omega)\boldsymbol{W}(j\Omega)} \frac{\boldsymbol{\sigma}_n^2 M}{\Phi_{SS}\boldsymbol{A}(j\Omega)\boldsymbol{A}^H(j\Omega))} = M. \tag{3.10}$$

This result is the theoretical best-case scenario, where the noise signals at the microphones are completely uncorrelated. As already seen from the coherence measurements in section 2.7 the

correlation between the noise signals increases drastically towards lower frequencies. Hence, the array gain will be much lower than M for a real beamformer.

3.2.2 Directivity Pattern

In general the array gain depends on both frequency and looking direction of the beamformer, therefore the *directivity pattern* is a more practical tool for examining the performance of a beamformer. The directivity pattern is obtained by evaluating the spatial selectivity of the beamformer for each looking direction Θ over the entire frequency range. The directional impulse response power [2] of the beamforming filter $\boldsymbol{W}(j\Omega)$ is calculated as

$$\boldsymbol{\Psi}(j\Omega,\Theta) = \boldsymbol{W}^H(j\Omega)\boldsymbol{a}\boldsymbol{a}^H\boldsymbol{W}(j\Omega) = \left|\boldsymbol{W}^H(j\Omega)\boldsymbol{a}\right|^2, \tag{3.11}$$

where vector $\boldsymbol{a}$ resembles the looking direction of the beamformer

$$\boldsymbol{a} = \left[\mathrm{e}^{-j\Delta\tau_0\Omega f_s} \quad \mathrm{e}^{-j\Delta\tau_1\Omega f_s} \quad \dots \quad \mathrm{e}^{-j\Delta\tau_M\Omega f_s}\right]^T, \tag{3.12}$$

and

$\Delta\tau_m = \frac{\Delta d_m}{c}\cos(\Delta\alpha - \Theta)$ is the relative time delay between an arbitrary reference point and the m-th microphone. The distance Δd_m and the corresponding angle $\Delta\alpha$ are measured from that reference point. In practice, this reference point is chosen to be one of the microphones. The sample rate of the discrete system is denoted as f_s. In this work, microphone 1 has been selected to be the reference. Θ is the looking direction of the array and c is the speed of sound, which is approximately $343\frac{m}{s}$ at $20°C$.

3.2.3 Directivity Index

The directivity pattern characterizes the performance of the beamformer depending on a certain looking direction. If a more general measure is needed, it is advantageous to omit this angular dependency. This can be achieved by using the Directivity Index. $D(j\Omega)$ is defined as the ratio of the directivity pattern in looking direction and the directivity pattern averaged over all directions [2]

$$D(j\Omega) = \frac{\boldsymbol{\Psi}(j\Omega,\Theta)}{\frac{1}{2\pi}\int_A \boldsymbol{\Psi}(j\Omega,\Theta_a)da}. \tag{3.13}$$

For a flat geometry like a linear microphone array, A denotes the path along the unit circle, so that each possible looking direction is covered by the integral. For a three-dimensional array structure, A would have to be the surface of a unit sphere. According to [2], for an arbitrary beamforming filter $\boldsymbol{W}(j\Omega)$ and an ideal diffuse noise field the directivity index can be expressed as

$$D(j\Omega) = \frac{\left|\boldsymbol{W^H}(j\Omega)\boldsymbol{a}\right|^2}{\sum_{m=1}^{M}\sum_{m'=1}^{M} W_m(j\Omega)W_{m'}^*(j\Omega)\gamma_{n_m n_{m'}}(j\Omega)}, \tag{3.14}$$

where $\gamma_{n_m n_{m'}}(j\Omega)$ is the spatial coherence function between microphone m and m' in the ideal diffuse noise field, which is already given in equation 2.17.

3.3 Delay-and-Sum Beamformer

The Delay-and-Sum beamformer (DS-BF) is one of the simplest beamformer structures. As the name suggests, all microphone signals are time-shifted such that the signal of interest is coherent in all channels. After the summation stage, the signal of interest is constructively added, and the noise signals are destructively added. This holds, as long as the noise is non-coherent and does not originate from the same direction than the desired speech signal.

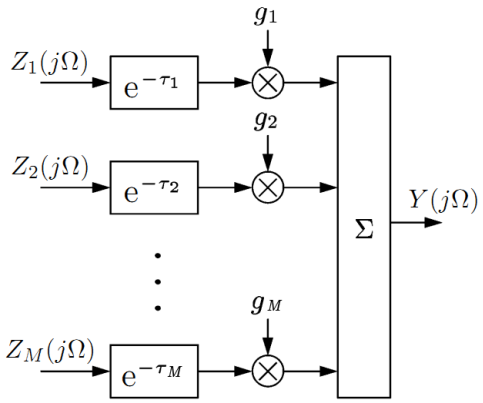

Figure 3.2: Structure of a delay-and-sum beamformer, taken from [29]. The frequency argument $j\Omega f_s$ in the delay lines has been omitted for readability.

With the signal model defined in section 2.3, the output of the DS-BF with M sensors can be written as

$$Y(j\Omega) = \sum_{m=1}^{M} g_m \mathrm{e}^{-j\Omega f_s \tau_m} Z_m(j\Omega), \tag{3.15}$$

where f_s is the sampling rate of the system. The delays τ_m and the weighing factors g_m form the spatial filter. The factor $\frac{1}{M}$ is incorporated into the weights to account for an unwanted amplification of the target signal. If equation 3.15 is expressed in vector notation, the delays and the weighing factors can be combined to form the spatial filter $\boldsymbol{W}(j\Omega)$

$$Y(j\Omega) = \boldsymbol{W}^H(j\Omega)\boldsymbol{Z}(j\Omega). \tag{3.16}$$

With the signal model defined in equation 2.14 the beamformer output $Y(j\Omega)$ becomes

$$\begin{aligned} Y(j\Omega) &= \boldsymbol{W}^H(j\Omega)\boldsymbol{A}(j\Omega)S(j\Omega) + \boldsymbol{W}^H(j\Omega)\boldsymbol{N}(j\Omega) \\ &= \hat{S}(j\Omega) + \hat{N}(j\Omega). \end{aligned} \tag{3.17}$$

The beamformer delivers an estimate of the desired source signal as $\hat{S}(j\Omega)$, accompanied by the modified noise signal $\hat{N}(j\Omega)$. If the spatial filter $\boldsymbol{W}(j\Omega)$ matches the ATFs $\boldsymbol{A}(j\Omega)$ exactly, all source signals $\hat{S}(j\Omega)$ are constructively aligned and the estimate of the speech signal becomes $\hat{S}(j\Omega) = S(j\Omega)$, because then $\boldsymbol{W}^H(j\Omega)\boldsymbol{A}(j\Omega) = \boldsymbol{A}^H(j\Omega)\boldsymbol{A}(j\Omega) = 1$. For the delay-and-sum beamformer, this condition can only be fulfilled if the ATFs consist of pure time delays. Physically, this is only possible in a perfect anechoic environment. Since $\boldsymbol{W}(j\Omega)$ points the beamformer into the direction of the desired signal, it is often called a *steering vector* [9]. Other

directions, especially those with strong interfering noise signals, should be suppressed as much as possible. These properties are best illustrated by the *directivity pattern* and the *directivity index*.

For simulation, a linear delay-and-sum beamforming array with $M = 4$ equally spaced microphones has been used. The inter-microphone distance $d = 5cm$. The desired signal impings from 0°, or *broadside direction*, and the noise signal has been simulated with an ideal diffuse sound field. The steering vector points at 0° and the sampling rate $f_s = 32kHz$.

Figure 3.3 shows the directivity pattern in (a) and the directivity index in (b). The directivity pattern has been calculated according to equation 3.11. The array gain $\mathbf{\Psi}(j\Omega, \Theta)$ is plotted in decibels, encoded by the colorbar to the right hand side of the plot in (a). The mainlobe at 0° is visible. However, the beamwidth varies greatly with frequency. While the beam only spans about 10° for high frequencies, there is no spatial selectivity for frequencies below 2kHz, since signals from all directions pass the beamformer unattenuated. For wavelengths smaller than the inter-microphone distance, spatial aliasing in the form of sidelobes occurs. For the given geometry, this happens for frequencies above $f = \frac{c}{\lambda} = 6860Hz$. In analogy to the sampling theorem, this form of aliasing is caused by spatial undersampling.

Making the array aperture (the biggest distance between any two microphones in the array) larger, the selectivity becomes better for lower frequencies. But the same time the sidelobes reduce towards lower frequencies as well, because the inter-microphone distance grows with the aperture. Hence, a careful tradeoff between these two phenomena has to be chosen for this beamformer structure.

Figure 3.3 (b) shows the directivity index for the same array according to equation 3.14 As already observable in the directivity pattern, the selectivity of this beamformer is poor for low frequencies. For higher frequencies, it converges to 6dB, which is in accordance to the array gain from equation 3.10 with $M = 4$ microphones.

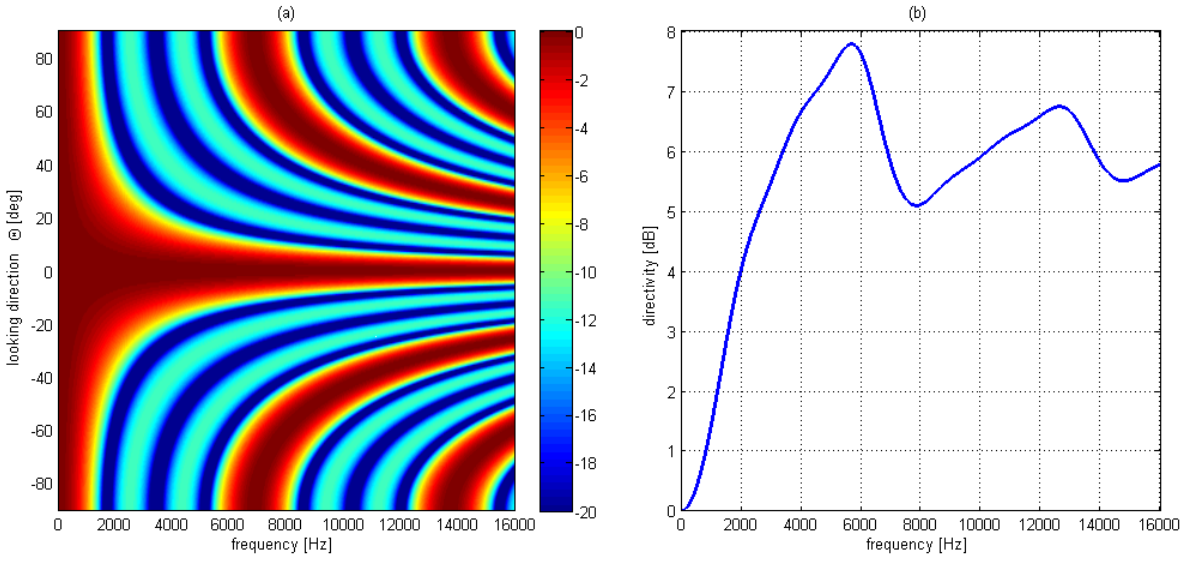

Figure 3.3: (a) Directivity pattern $\mathbf{\Psi}(j\Omega, \Theta)$ of a delay-and-sum beamformer with 4 microphones, all spaced 5cm apart, from broadside direction. (b) Directivity index $D(j\Omega)$ for the same array.

It is obvious that the delay-and-sum beamformer cannot completely remove interfering noise

signals. It rather low-pass filters the noise. Therefore it is of limited use in a practical application.

3.4 Filter-and-sum Beamformer

The filter-and-sum beamformer allows for a more flexible shaping of the directivity pattern. As shown in figure 3.4, each microphone signal is filtered before the summation. Thereby, for each frequency bin an independent narrowband beamformer is designed. With this structure the spatial response can be shaped for each frequency independently. Especially for low frequencies, where the wavelength is much larger than the array aperture, the directivity can be greatly improved.

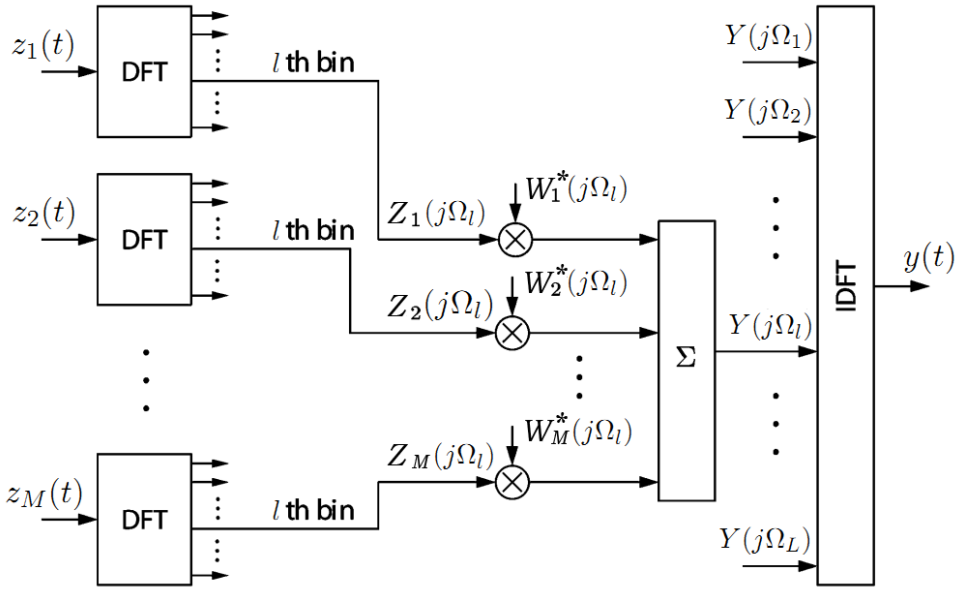

Figure 3.4: Structure of a filter-and-sum beamformer, taken from [29]. The FFT produces L non-redundant frequency bins.

The output of the filter-and-sum beamformer can be written as

$$Y(j\Omega) = \boldsymbol{W}^H(j\Omega)\boldsymbol{Z}(j\Omega), \tag{3.18}$$

where the matrix $\boldsymbol{W}^H(j\Omega)$ now represents M FIR filters instead of simple delays. By using this structure, the following design goal can be formulated: The output power of the noise should be minimized while preserving the desired signal without any distortions. This design is termed the *minimum variance distortionless response* (MVDR) beamformer [2]. The power spectral density of the output $Y(j\Omega)$ has the same structure as shown in equation 3.7. The noise power of the output equates to

$$\Phi_{\hat{N}\hat{N}}(j\Omega) = \boldsymbol{W}^H(j\Omega)\boldsymbol{\Phi}_{NN}(j\Omega)\boldsymbol{W}(j\Omega), \tag{3.19}$$

and is minimized by the MVDR beamformer. The *distortionless response* requirement is formu-

lated as unit gain in looking direction

$$\boldsymbol{W}^H(j\Omega)\boldsymbol{A}(j\Omega) = 1. \tag{3.20}$$

This constrained minimization problem is solved by using the method of Lagrange multipliers, with the Lagrange function given as

$$\mathcal{L}(\boldsymbol{W}(j\Omega), \lambda) = \Phi_{\hat{N}\hat{N}}(j\Omega) + \lambda(\boldsymbol{W}^H(j\Omega)\boldsymbol{A}(j\Omega) - 1). \tag{3.21}$$

Computing the gradient with respect to $\boldsymbol{W}^H(j\Omega)$ and λ and setting the result to zero gives the necessary conditions

(i): $2\boldsymbol{\Phi}_{nn}(j\Omega)\boldsymbol{W}(j\Omega) + \lambda\boldsymbol{A}(j\Omega) = 0$, and
(ii): $\boldsymbol{W}^H(j\Omega)\boldsymbol{A}(j\Omega) = \boldsymbol{F}(j\Omega) = 1.$

The optimal solution for $\boldsymbol{W}(j\Omega)$ is obtained by solving for λ and substituting into the first condition

$$\boldsymbol{W}_{MVDR}(j\Omega) = \frac{\boldsymbol{\Phi}_{NN}^{-1}(j\Omega)\boldsymbol{A}(j\Omega)}{\boldsymbol{A}^H(j\Omega)\boldsymbol{\Phi}_{NN}^{-1}(j\Omega)\boldsymbol{A}(j\Omega)}. \tag{3.22}$$

For the case of $M = 2$ microphones, a geometric interpretation of this constrained optimization problem is provided in figure 3.5. The noise power of the beamformer output $\boldsymbol{W}^H(j\Omega)\boldsymbol{\Phi}_{NN}(j\Omega)\boldsymbol{W}(j\Omega)$ from equation 3.19 is shown as ellipse around the origin, and the line touching it represents the constraint $\boldsymbol{W}^H(j\Omega)\boldsymbol{A}(j\Omega)$ from equation 3.20. The point where the two intersect minimizes the output power and meets the design constraint at the same time. Hence it depicts the optimal MVDR solution.

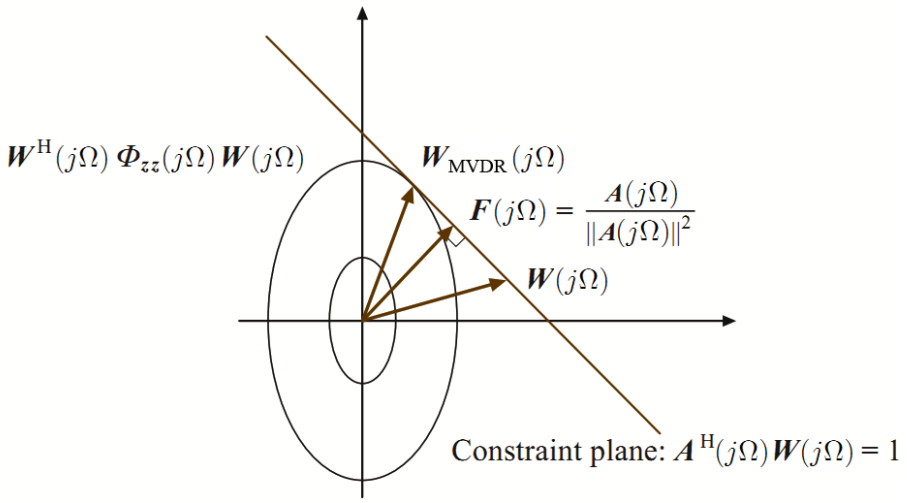

Figure 3.5: Constrained minimization, taken from [1].

To compare the performance of this beamformer to the delay-and-sum structure, the same array geometry with $M = 4$ equally spaced microphones has been used for simulation. Again, the noise is an ideal diffuse sound field. Figure 3.6 shows the directivity pattern in (a) and the

directivity index in (b). The directivity pattern shows an improved spatial selectivity for lower frequencies, and the directivity index is greater than 3.5dB for all frequencies.

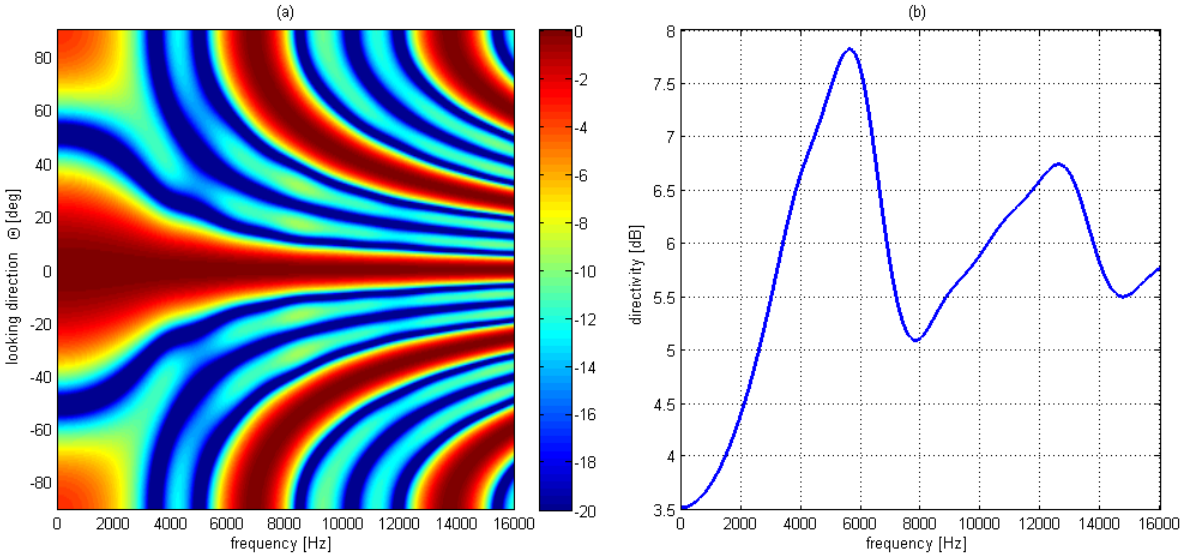

Figure 3.6: (a) Directivity pattern $\boldsymbol{\Psi}(j\Omega, \Theta)$ of a MVDR beamformer with 4 microphones, all spaced 5cm apart, with speech impinging from 0°, and an ideal diffuse noise field. (b) Directivity index $D(j\Omega)$ for the same array.

However, the capabilities of this beamformer can be best demonstrated using a directional noise field. Suppose there is a distinct noise source located at 20°, and the speech signal impings from 0°, as before. Then, equation 3.22 produces a beamformer that shows no distortion for the speech signal, i.e. the gain is 1 for all frequencies at 0°. The noise source on the other hand, is perfectly suppressed for all frequencies. This suppression can be regarded as a *null* in the response of the spatial filter $\boldsymbol{W}_{MVDR}(j\Omega)$. Therefore, the MVDR filter automatically steers a null towards the noise source. With M microphones, a maximum of $M-1$ spatial nulls can be constructed. The directivity pattern in figure 3.7 (a) demonstrates this null steering.

It can be seen that the MVDR beamformer perfectly suppresses the noise signal originating from 20°, while keeping the desired signal located at 0°. However, sounds from other directions do not seem to be suppressed, but amplified for frequencies below 2kHz, as the directivity index in panel (b) indicates. This is because there is no other constraint given except for the speech at 0° and the noise at 20°.

In an actual implementation, several problems arise when using the MVDR beamformer:

- The ATFs $\boldsymbol{A}(j\Omega)$ need to be known or estimated. Since the ATFs may be long FIR filters, it is advantageous to estimate the RTFs $\hat{\boldsymbol{A}}(j\Omega)$ instead. If there is not much reverberation, it may even be sufficient to estimate the general direction of the source signal. This task is known as *direction of arrival* (DOA) estimation.
- The cross-power spectral density matrix $\boldsymbol{\Phi_{NN}}(j\Omega)$ of the noise signals at the microphones is not known a priori and its estimation is not trivial.
- The matrix $\boldsymbol{\Phi_{NN}}(j\Omega)$ must be invertible and well-conditioned. For low frequencies, this is generally not the case because the correlation between the noise signals is high. As a

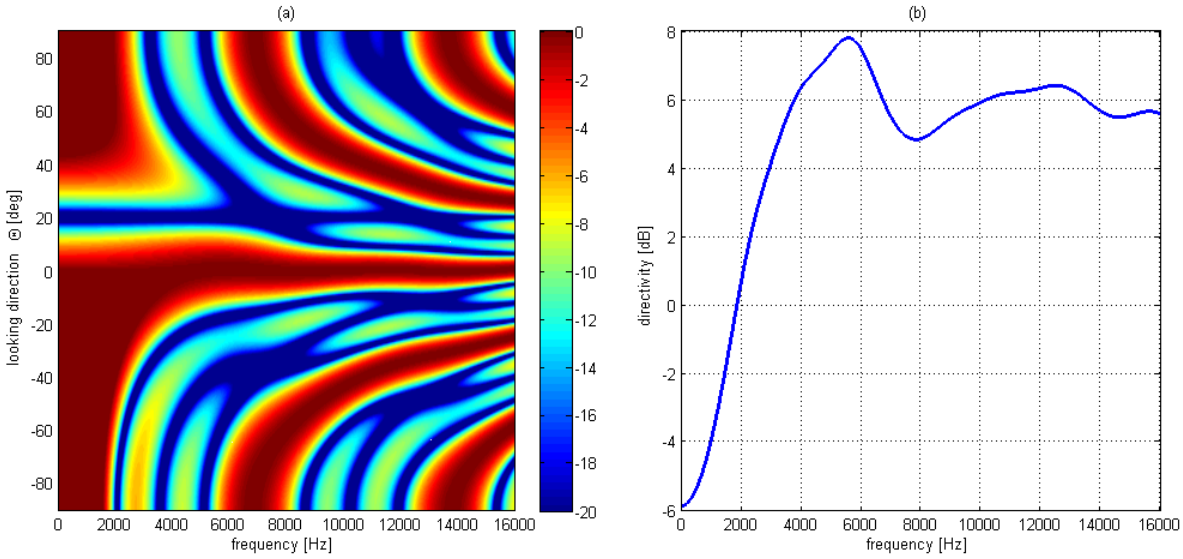

Figure 3.7: (a) Directivity pattern $\boldsymbol{\Psi}(j\Omega, \Theta)$ of a MVDR beamformer with 4 microphones, all spaced 5cm apart, with speech impinging from 0°, and noise impinging from 20°. (b) Directivity index $D(j\Omega)$ for the same array.

consequence, low-frequency filter coefficients can get arbitrarily large. This also implies an arbitrarily high amplification of sensor noise or ambient low-frequency sounds.

To overcome the estimation and inversion of $\boldsymbol{\Phi_{NN}}(j\Omega)$, an adaptive algorithm has been proposed by *Frost* [9]. Its frequency domain realization is presented in [1] and is given as

$$\boldsymbol{W}(k+1, j\Omega) = \boldsymbol{P}(j\Omega)[\boldsymbol{W}(k, j\Omega) - \mu \boldsymbol{Z}(k, j\Omega) Y^*(k, j\Omega)] + \boldsymbol{F}(j\Omega), \tag{3.23}$$

where k denotes the frame index, μ is the learning rate, and $\boldsymbol{P}(j\Omega)$ is the projection matrix to the null space of $\boldsymbol{A}(j\Omega)$ [30], i.e.

$$\boldsymbol{P}^H(j\Omega)\boldsymbol{A}(j\Omega) \stackrel{!}{=} \boldsymbol{0}_{M\times 1}.$$

This is fulfilled when

$$\boldsymbol{P}(j\Omega) = \boldsymbol{I} - \frac{\boldsymbol{A}^H(j\Omega)\boldsymbol{A}(j\Omega)}{\|\boldsymbol{A}(j\Omega)\|_2^2}. \tag{3.24}$$

As shown in figure 3.5, the vector $\boldsymbol{F}(j\Omega)$ is perpendicular to the unity gain constraint defined in equation 3.20. It resembles a fixed filter-and-sum beamformer where the output is normalized to 1

$$\boldsymbol{F}(j\Omega) = \frac{\boldsymbol{A}(j\Omega)}{\|\boldsymbol{A}(j\Omega)\|_2^2}. \tag{3.25}$$

While the estimation of $\boldsymbol{\Phi_{NN}}(j\Omega)$ is avoided with Frost's algorithm, the susceptibility to sensor noise still remains. Additionally, the width of the mainlobe is small for higher frequencies and cannot be controlled, which makes the beamformer sensitive to positioning errors of the

microphones and estimation errors of the source location. These robustness issues make the Frost algorithm unsuitable for practical applications.

3.5 Generalized Sidelobe Canceler

To increase the robustness of the *Frost* beamformer, *Griffiths* and *Jim* [9] considered to split the optimal MVDR filter $\boldsymbol{W}_{MVDR}(j\Omega)$ from equation 3.22 into two independent filters operating in mutually orthogonal subspaces [30]. The first filter coherently aligns the desired speech signal, and the second filter constructs a reference of the noise signals, which is then subtracted to obtain a good speech estimate. This design is termed the *generalized sidelobe canceler* (GSC). It is defined as

$$\boldsymbol{W}_{MVDR}(j\Omega) = \boldsymbol{F}(j\Omega) - \boldsymbol{B}(j\Omega)\boldsymbol{H}(j\Omega). \tag{3.26}$$

It consists of three main blocks: THe *Fixed Beamformer* $\boldsymbol{F}(j\Omega)$, the *Blocking Matrix* $\boldsymbol{B}(j\Omega)$ and the *Adaptive Interference Canceler* $\boldsymbol{H}(j\Omega)$. An overview of the GSC structure is given by the block diagram in figure 3.8.

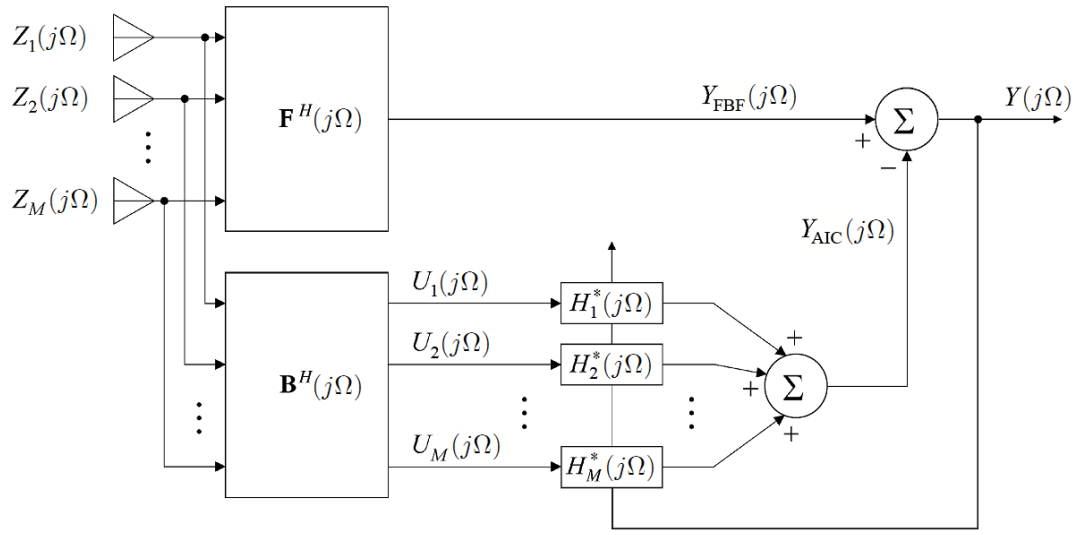

Figure 3.8: Structure of the generalized sidelobe canceler [1].

The fixed beamformer $\boldsymbol{F}(j\Omega)$ in equation 3.26 is essentially a filter-and-sum beamforming structure, and has already been defined in equation 3.25. As $\boldsymbol{F}(j\Omega)$ relies on the ATFs of the desired speech source, the fixed beamformer aligns the desired speech signal of each microphone so that they are constructively added. Its output is given as

$$Y_{FBF}(j\Omega) = \boldsymbol{F}^H(j\Omega)\boldsymbol{Z}(j\Omega). \tag{3.27}$$

The ATFs used for the fixed beamformer can often replaced by simple delays, that represent the direction of the desired sound source [1]. This is possible, if the direct path to the source's location is stronger than the more complex reflection paths. ATFs that represent pure time

delays are given as

$$\boldsymbol{A}(j\Omega) = \begin{bmatrix} \mathrm{e}^{-j\Omega f_s \tau_1} & \mathrm{e}^{-j\Omega f_s \tau_2} & \dots & \mathrm{e}^{-j\Omega f_s \tau_M} \end{bmatrix}^T. \tag{3.28}$$

Using these ATFs, the fixed beamformer $\boldsymbol{F}(j\Omega)$ turns into a delay-and-sum beamformer

$$\boldsymbol{F}(j\Omega) = \boldsymbol{A}^H(j\Omega). \tag{3.29}$$

Like the other beamforming structures, the fixed beamformer has a low spatial selectivity for low frequencies. Hence it contains much of the unwanted signals in lower frequencies. At higher frequencies, the sidelobes of the beamformer causes some of the unwanted signals to leak through, as explained in figure 3.3. Since it is cumbersome to eliminate all of these noise sources, the GSC rather masks out the desired source signal to get a good reference of all unwanted signals. This is done by the second term $\boldsymbol{B}(j\Omega)\boldsymbol{H}(j\Omega)$ in equation 3.26. This noise-only signal is then subtracted from the fixed beamformer's output, as equation 3.26 already suggests.

The $M \times M$ matrix $\boldsymbol{B}(j\Omega)$ has the same structure as the projection matrix in equation 3.24. Its columns span the null space of $\boldsymbol{A}(j\Omega)$, so that

$$\boldsymbol{B}^H(j\Omega)\boldsymbol{A}(j\Omega) \stackrel{!}{=} \boldsymbol{0}_{1\times M}. \tag{3.30}$$

Geometrically, this constraint can be interpreted as follows: A signal that propagates along the acoustic paths described by $\boldsymbol{A}(j\Omega)$ gets completely canceled when filtered with $\boldsymbol{B}(j\Omega)$. And since $\boldsymbol{A}(j\Omega)$ describes the ATFs to the desired speech source location, the matrix $\boldsymbol{B}(j\Omega)$ blocks the speech signal. Hence, it is called *blocking matrix* (BM).

The output of the blocking matrix is termed the *noise references* $\boldsymbol{U}(j\Omega)$, as all speech components get effectively canceled and only noise components remain as

$$\boldsymbol{U}(j\Omega) = \boldsymbol{B}^H(j\Omega)\boldsymbol{Z}(j\Omega) = \begin{bmatrix} U_1(j\Omega) & U_2(j\Omega) & \cdots & U_M(j\Omega) \end{bmatrix}^T. \tag{3.31}$$

The vector $\boldsymbol{H}(j\Omega)$ contains M adaptive filters, whose purpose is to align the noise references $\boldsymbol{U}(j\Omega)$ to the noise components in the fixed beamformer's output $Y_{FBF}(j\Omega)$. The filters in $\boldsymbol{H}(j\Omega)$ are termed the *Adaptive Interference Canceler* (AIC). Using the noise references, the output of the AIC equates to

$$Y_{AIC}(j\Omega) = \boldsymbol{H}^H(j\Omega)\boldsymbol{U}(j\Omega) = \boldsymbol{H}^H(j\Omega)\boldsymbol{B}^H(j\Omega)\boldsymbol{Z}(j\Omega). \tag{3.32}$$

Combining the fixed beamformer from equation 3.27, the blocking matrix and the adaptive interference canceler in equation 3.32, the structure of the generalized sidelobe canceler emerges as

$$Y(j\Omega) = Y_{FBF}(j\Omega) - Y_{AIC}(j\Omega) = \Big[\boldsymbol{F}^H(j\Omega) - \boldsymbol{H}^H(j\Omega)\boldsymbol{B}^H(j\Omega)\Big]\boldsymbol{Z}(j\Omega). \tag{3.33}$$

While this beamformer cannot surpass the performance of the MVDR beamformer, it avoids most of the robustness problems as it splits the task effectively in three parts:

- The fixed beamformer $\boldsymbol{F}(j\Omega)$ coherently aligns the desired signal.
- The blocking matrix $\boldsymbol{B}(j\Omega)$ blocks the desired signal resulting in the noise references $\boldsymbol{U}(j\Omega)$.
- The adaptive interference canceler $\boldsymbol{H}(j\Omega)$ eliminates the noise components that leak through the sidelobes of the fixed beamformer.

The adaptive AIC filters in $\boldsymbol{H}(j\Omega)$ are determined by minimizing the noise power at the beamformer output. This minimization task can be regarded as a multichannel Wiener filter optimization problem, which has been introduced in section 2.8. In the following, the instantaneous Wiener solution and the adaptive version using the NLMS algorithm is specified.

Using equation 3.33 and 3.32, the power-spectral density (PSD) at the beamformer output calculates as

$$\begin{aligned}\Phi_{YY}(j\Omega) &= \mathbb{E}\left\{\left[Y_{FBF}(j\Omega) - \boldsymbol{H}^H(j\Omega)\boldsymbol{U}(j\Omega)\right]\left[Y_{FBF}(j\Omega) - \boldsymbol{H}^H(j\Omega)\boldsymbol{U}(j\Omega)\right]^H\right\} \\ &= \Phi_{Y_{FBF}Y_{FBF}}(j\Omega) - \boldsymbol{\Phi}_{Y_{FBF}U}(j\Omega)\boldsymbol{H}(j\Omega) - \boldsymbol{H}^H(j\Omega)\boldsymbol{\Phi}_{UY_{FBF}}(j\Omega) \\ &\quad + \boldsymbol{H}^H(j\Omega)\boldsymbol{\Phi}_{UU}(j\Omega)\boldsymbol{H}(j\Omega).\end{aligned} \tag{3.34}$$

It is used as multichannel cost function of the multichannel Wiener filter problem. The optimal result in a MSE sense is found by taking the first derivative with respect to $\boldsymbol{H}^H(j\Omega)$ and equating the result to zero

$$\frac{\partial \Phi_{YY}(j\Omega)}{\partial \boldsymbol{H}^H(j\Omega)} = -2\boldsymbol{\Phi}_{UY_{FBF}}(j\Omega) + 2\boldsymbol{\Phi}_{UU}(j\Omega)\boldsymbol{H}(j\Omega) \stackrel{!}{=} 0. \tag{3.35}$$

Solving for $\boldsymbol{H}(j\Omega)$ yields the AIC filters

$$\boldsymbol{H}(j\Omega) = \boldsymbol{\Phi}_{UU}^{-1}(j\Omega)\boldsymbol{\Phi}_{UY_{FBF}}(j\Omega). \tag{3.36}$$

These filters are generally non-causal. They model the spatial correlation of the noise signal amongst the microphones, and therefore contain both positive and negative time delays. The matrices $\boldsymbol{\Phi}_{UU}(j\Omega)$ and $\boldsymbol{\Phi}_{UY_{FBF}}(j\Omega)$ represent the PSD of the noise references and the cross-PSD between the noise references and the fixed beamformer output, respectively. Both terms are given as

$$\begin{aligned}\boldsymbol{\Phi}_{UU}(j\Omega) &= \mathbb{E}\{\boldsymbol{U}(j\Omega)\boldsymbol{U}^H(j\Omega)\} \\ \boldsymbol{\Phi}_{UY_{FBF}}(j\Omega) &= \mathbb{E}\{\boldsymbol{U}(j\Omega)Y_{FBF}^*(j\Omega)\}.\end{aligned} \tag{3.37}$$

In an actual implementation, they can be estimated by using first-order recursive averaging, i.e.

$$\begin{aligned}\boldsymbol{\Phi}_{UU}(k, j\Omega) &= \boldsymbol{\Phi}_{UU}(k-1, j\Omega)\alpha + (1-\alpha)\boldsymbol{U}(k, j\Omega)\boldsymbol{U}^H(k, j\Omega) \\ \boldsymbol{\Phi}_{UY_{FBF}}(k, j\Omega) &= \boldsymbol{\Phi}_{UY_{FBF}}(k-1, j\Omega)\alpha + (1-\alpha)\boldsymbol{U}(k, j\Omega)Y_{FBF}^*(k, j\Omega),\end{aligned} \tag{3.38}$$

where k is the frame index, and α is a smoothing parameter, chosen to be in the range of $0 < \alpha < 1$. Considering equation 3.36, a problem arises with the inversion of $\boldsymbol{\Phi}_{UU}(j\Omega)$. Its estimate using equation 3.37 can be ill-conditioned, thus imposing numerical instability to the

matrix inversion. As a countermeasure, a diagonal loading term $\epsilon \boldsymbol{I}_{M\times M}$ with a small constant ϵ may be added. But this decreases the quality of the inverse when the matrix only has small elements, and thus distract the AIC filters when the input signal power is low.

The matrix inverse can also be replaced by $\frac{1}{\Phi_{UU}(k,j\Omega)}$, which is given by

$$\Phi_{UU}(k, j\Omega) = \Phi_{UU}(k-1, j\Omega)\alpha + (1-\alpha)\|\boldsymbol{U}(k, j\Omega)\|_2^2. \tag{3.39}$$

Equation 3.39 is the time-averaged normalization term of the multichannel NLMS algorithm given in equation 2.39. By incorporating this term, equation 3.36 changes into

$$\boldsymbol{H}(j\Omega) = \frac{\boldsymbol{\Phi}_{UY_{FBF}}(j\Omega)}{\Phi_{UU}(k, j\Omega)}. \tag{3.40}$$

Stable AIC filter are guaranteed, if the normalization term is larger than the largest eigenvalue λ_{max} of $\boldsymbol{\Phi}_{UU}(j\Omega)$ [26]. A proof can be found by

$$\|\boldsymbol{U}(k, j\Omega)\|_2^2 = \text{trace}\left(\boldsymbol{\Phi}_{UU}(j\Omega)\right) = \sum_{m=1}^{M} \lambda_m > \lambda_{max}. \tag{3.41}$$

With equation 3.40 an instantaneous estimate of the AIC filters is given, it only depends of the PSD estimates in equation 3.38 and 3.39. By using the multichannel NLMS algorithm introduced in chapter 2.8, the filters $\boldsymbol{H}(j\Omega)$ can also be determined adaptively [20] with

$$\boldsymbol{H}(k+1, j\Omega) = \boldsymbol{H}(k, j\Omega) + \frac{\mu}{P_{est}(k, j\Omega)}\boldsymbol{U}(k, j\Omega)Y^*(k, j\Omega), \tag{3.42}$$

where k is the frame number and μ the learning rate of the NLMS algorithm. The normalization term $P_{est}(k, j\Omega)$ of the summed power spectral density of the inputs $\boldsymbol{Z}(j\Omega)$ is determined by recursive averaging, similar to equation 3.39

$$P_{est}(k, j\Omega) = P_{est}(k-1, j\Omega)\alpha + (1-\alpha)\|\boldsymbol{Z}(k, j\Omega)\|_2^2. \tag{3.43}$$

Basically, the calculation of the filters $\boldsymbol{H}(j\Omega)$ can only be performed in the absence of the desired speech signal. Therefore, a *voice activity detector* (VAD) is needed. Obtaining an accurate VAD is a non-trivial task, and still subject to research. Fortunately, the VAD for this application does not require high accuracy, as the cross-power spectral densities are long-term averages. In the experiments section, both the instantaneous and the adaptive AIC filters are used for comparison.

3.6 Constructing a Blocking Matrix

What remains to be defined is the blocking matrix $\boldsymbol{B}(j\Omega)$. Its purpose is to remove the speech signal to get an estimate of the noise signals. Only if the blocking matrix perfectly blocks all speech components, $\boldsymbol{U}(j\Omega)$ contains clean noise estimates. In practice, some leakage of the desired signal into the noise reference branch is unavoidable. This so-called *target leakage* causes the AIC to cancel out some desired speech components. This immediately degrades the overall

performance of the GSC structure. Building a good blocking matrix has been the subject of numerous papers. The most influential ones are presented in the following.

3.6.1 Eigenspace Blocking Matrix

The blocking matrix spans the null space [31] of $\boldsymbol{A}(j\Omega)$ in order to generate noise references $\boldsymbol{U}(j\Omega)$ that are orthogonal to the output of the fixed beamformer $\boldsymbol{F}(j\Omega)$. The simplest blocking matrix which fullfils that condition was already given as projection matrix in equation 3.24 for the Frost beamformer. This $M \times M$ blocking matrix is

$$\boldsymbol{B}^H(j\Omega) = \boldsymbol{I} - \frac{\boldsymbol{A}^H(j\Omega)\boldsymbol{A}(j\Omega)}{\|\boldsymbol{A}(j\Omega)\|_2^2}, \tag{3.44}$$

where $\boldsymbol{I}$ is the $M \times M$ identity matrix. It can be easily verified that this blocking matrix fulfills the constraint $\boldsymbol{B}^H(j\Omega)\boldsymbol{A}(j\Omega) \stackrel{!}{=} \boldsymbol{0}_{1\times M}$, and forms a null space of $\boldsymbol{A}(j\Omega)$.
Using this blocking matrix involves M^2 complex multiplications per frequency bin. This is by far the most complex part of the GSC. Further, the full ATFs $\boldsymbol{A}(j\Omega)$ need to be estimated.

3.6.2 Generalized Eigenvector Blocking Matrix

A more sophisticated blocking matrix is based on a beamforming filter, which tries to maximize the output SNR as presented in [32]. According to the signal model defined in equation 2.14, the power spectral density at the microphones calculates to

$$\boldsymbol{\Phi}_{ZZ}(j\Omega) = \boldsymbol{A}(j\Omega)\Phi_{SS}(j\Omega)\boldsymbol{A}^H(j\Omega) + \boldsymbol{\Phi}_{NN}(j\Omega).$$

Then, the SNR at the output of an arbitrary beamforming filter $\boldsymbol{W}(j\Omega)$ is given as

$$oSNR(j\Omega) = \frac{\boldsymbol{W}(j\Omega)^H\boldsymbol{\Phi}_{ZZ}(j\Omega)\boldsymbol{W}(j\Omega)}{\boldsymbol{W}(j\Omega)^H\boldsymbol{\Phi}_{NN}(j\Omega)\boldsymbol{W}(j\Omega)} - 1. \tag{3.45}$$

The beamforming filter $\boldsymbol{W}(j\Omega)$ maximizing this SNR is equivalent to the eigenvector corresponding to the largest eigenvalue of $\boldsymbol{\Phi}_{NN}^{-1}(j\Omega)\boldsymbol{\Phi}_{ZZ}(j\Omega)$. In [32], this principal eigenvector is further derived as

$$\boldsymbol{W}(j\Omega) = \zeta(j\Omega) \cdot \boldsymbol{\Phi}_{NN}^{-1}(j\Omega)\boldsymbol{A}(j\Omega), \tag{3.46}$$

where $\zeta(j\Omega)$ is an arbitrary complex constant. The same authors use this result in [33] to construct a projection into the orthogonal complement of $\boldsymbol{A}(j\Omega)$. This projection has already been defined in equation 3.30, it ensures the orthogonality between the noise references $\boldsymbol{U}(j\Omega)$ and the beamformer output $Y(j\Omega)$

$$\mathbb{E}\{\boldsymbol{U}(j\Omega)Y(j\Omega)\} \stackrel{!}{=} \boldsymbol{0}_{1\times M}, \tag{3.47}$$

with

$$\boldsymbol{U}(j\Omega) = \boldsymbol{B}^H(j\Omega)\boldsymbol{Z}(j\Omega),$$

and

$$Y(j\Omega) = \boldsymbol{W}^H(j\Omega)\boldsymbol{Z}(j\Omega).$$

Using 3.30 and 3.47, the generalized eigenvector blocking matrix $\boldsymbol{B}(j\Omega)$ equates to [33]

$$\boldsymbol{B}(j\Omega) = \boldsymbol{I} - \frac{\boldsymbol{A}(j\Omega)\boldsymbol{A}^H(j\Omega)\boldsymbol{\Phi}_{NN}^{-1}(j\Omega)}{\boldsymbol{A}^h(j\Omega)\boldsymbol{\Phi}_{NN}^{-1}(j\Omega)\boldsymbol{A}(j\Omega)}. \tag{3.48}$$

It can be easily verified that this blocking matrix fulfills the constraint $\boldsymbol{B}^H(j\Omega)\boldsymbol{A}(j\Omega) \stackrel{!}{=} \boldsymbol{0}_{1\times M}$, and forms a null space of $\boldsymbol{A}(j\Omega)$. Also, its performance is good when used on real-world data. The drawback is, that the noise correlation matrix $\boldsymbol{\Phi}_{NN}(j\Omega)$ has to be known, which is hard to estimate in a real-time application.

3.6.3 Adaptive Blocking Matrix

The main goal of the blocking matrix is to block the desired speech signal as good as possible. This can also be achieved with an adaptive filter or adaptive blocking matrix (ABM) instead of a fixed blocking matrix. This concept has first been presented in [27], and has the advantage that the ATFs are found adaptively and do not need to be separately estimated. Also, the numerical complexity reduces from M^2 filter operations to M filter operations. The basic principle is shown in figure 3.9.

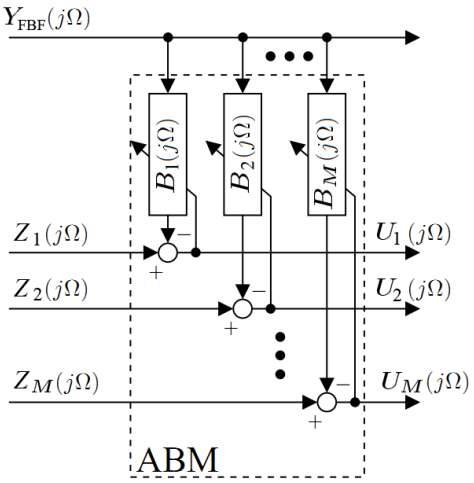

Figure 3.9: Structure of an adaptive blocking matrix.

Each filter $B_m(j\Omega)$ is a SISO Wiener filter [34], and found by minimizing the energy at each noise reference $U_m(j\Omega)$. This energy is given as

$$\begin{aligned}\Phi_{U_mU_m}(j\Omega) &= \mathbb{E}\{U_m(j\Omega)U_m^*(j\Omega)\}\\ &= \Big[Z_m(j\Omega) - Y_{FBF}(j\Omega)B_m(j\Omega)\Big]\Big[Z_m(j\Omega) - Y_{FBF}(j\Omega)B_m(j\Omega)\Big]^*\\ &= \Phi_{Z_mZ_m}(j\Omega) - \Phi_{Z_mY_{FBF}}(j\Omega)B_m^*(j\Omega) - \Phi_{Y_{FBF}Z_m}(j\Omega)B_m(j\Omega)\\ &+ B_m(j\Omega)\Phi_{Y_{FBF}Y_{FBF}}(j\Omega)B_m^*(j\Omega).\end{aligned} \tag{3.49}$$

Taking the derivative of $\Phi_{U_m U_m}(j\Omega)$ with respect to $B_m^*(j\Omega)$ and equating it to zero

$$\frac{\partial \Phi_{U_m U_m}(j\Omega)}{\partial B_m^*(j\Omega)} = -2\Phi_{Z_m Y_{FBF}}(j\Omega) + 2B_m(j\Omega)\Phi_{Y_{FBF}Y_{FBF}}(j\Omega) \stackrel{!}{=} 0, \tag{3.50}$$

gives the Wiener solution for the optimal adaptive filters $B_{m,OPT}(j\Omega)$

$$B_{m,OPT}(j\Omega) = \frac{\Phi_{Z_m Y_{FBF}}(j\Omega)}{\Phi_{Y_{FBF}Y_{FBF}}(j\Omega)}. \tag{3.51}$$

These filters can also be found recursively using the SISO NLMS algorithm

$$B_m(k+1, j\Omega) \overset{FIR}{\leftarrow} B_m(k, j\Omega) + \frac{\mu Y_{FBF}(j\Omega) U_m(j\Omega)}{\Phi_{Y_{FBF}Y_{FBF}}(j\Omega) + \delta}. \tag{3.52}$$

Again, k denotes the frame index, μ the NLMS learning rate and δ a small regularization constant.
While this ABM might be a good solution in theory, in practice several problems arise: The adaption algorithm can only be executed in the presence of a desired speech signal, while at the same time there must not be any interfering noise. This may be practically impossible for an application in a noisy environment. Also, reverberation in the ATFs might divert the adaptive filters $B_m(j\Omega)$ from the speaker's location, which is why several constraints have been imposed on the NLMS algorithm in the original paper [27].
Despite the theoretical benefits of this solution, simulations using real-world data showed a bad performance with this type of blocking matrix. This may be due to the contrary adaption scheme of the BM and the AIC: The AIC filters may only be updated during speech absence, and the BM filters only during speech presence. This way, the noise references $\boldsymbol{U}(j\Omega)$ keep changing, but the AIC cannot react in time to that change.

3.6.4 Sparse Blocking Matrix

The authors in [31] try to reduce the complexity of the eigenspace blocking matrix while maintaining its performance at the same time. First, it is shown that the constraint $\boldsymbol{B}^H(j\Omega)\boldsymbol{A}(j\Omega) \stackrel{!}{=} \boldsymbol{0}_{1\times M}$ limits the rank of the blocking matrix to

$$rank\{\boldsymbol{B}(j\Omega)\} \leq M - 1.$$

This means that the size of the blocking matrix can be reduced to $M \times M - 1$. This has a great impact on the complexity of the entire GSC beamformer, since both the number of noise references in $\boldsymbol{U}(j\Omega)$ and the number of AIC filters in $\boldsymbol{H}(j\Omega)$ are reduced to $M - 1$ as well.

When using RTFs $\frac{\boldsymbol{A}(j\Omega)}{A_1(j\Omega)}$ instead of ATFs $\boldsymbol{A}(j\Omega)$, the blocking matrix can be further simplified to a sparse matrix. It is shown, that a *sparse* blocking matrix performs equally well as the

eigenspace blocking matrix. The sparse BM is defined as [31]

$$\boldsymbol{B}(j\Omega) = \begin{bmatrix} -\frac{A_2^*}{A_1^*} & -\frac{A_3^*}{A_1^*} & \cdots & -\frac{A_M^*}{A_1^*} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{3.53}$$

It can be easily verified that this blocking matrix also fulfills the constraint $\boldsymbol{B}^H(j\Omega)\boldsymbol{A}(j\Omega) \stackrel{!}{=} \boldsymbol{0}_{1\times M-1}$, and forms a null space of $\boldsymbol{A}(j\Omega)$. Hence, orthogonality between $\boldsymbol{F}(j\Omega)$ and $\boldsymbol{B}(j\Omega)$ is also ensured. In a graphical interpretation, this blocking matrix steers $M-1$ spatial nulls towards the direction of the speech signal [9].

When using simple time delays instead of the RTFs, the blocking matrix becomes even simpler

$$\boldsymbol{B}(j\Omega) = \begin{bmatrix} -\mathrm{e}^{-j\Omega f_s \tau_2} & -\mathrm{e}^{-j\Omega f_s \tau_3} & \cdots & -\mathrm{e}^{-j\Omega f_s \tau_M} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{3.54}$$

but then the risk of undermodelling the acoustic paths from the source to the microphones arises, causing reverberations of the desired speech signal to leak through the blocking matrix. On the other hand, estimation errors in the RTFs may cause more leakage than this undermodelling. Whether it is preferable to estimate the RTFs, or to use delay-only models is experimentally verified in section 6.

3.7 Design Considerations

The MVDR beamforming filter provides good noise suppression and low speech distortions at the same time. In contrast to direct approaches like the Frost algorithm, the GSC is robust enough for a practical implementation. This is possible because it splits the MVDR constraint into a fixed beamformer, and a blocking matrix with an AIC. This allows for separate target tracking and noise removal, and hence a greater stability. However, the GSC cannot exceed the performance of the MVDR filter, as it is merely another formulation.

In a real application scenario, the GSC shows some deficiencies:

(i): Since the GSC belongs to the class of MVDR beamformers, it also has a poor spatial selectivity for low frequencies, especially when the array aperture is small. This effect can be seen in figure 3.6. As a result, the noise reduction performance is bad for low frequencies and has to be compensated by a postprocessing stage. This topic is covered in chapter 5.

(ii): The RTFs need to be estimated, and estimation errors cause a phenomenon known as target leakage in the blocking matrix. This causes speech components to appear in the noise references, which are then subtracted by the AIC. Hence the overall speech quality decreases.

(iii): Simple mechanical problems can also cause target leakage. If the blocking matrix is built upon a predefined array geometry, sensor misalignment due to assembly tolerances cannot be neglected. Also, the speed of sound c is temperature dependent. To avoid this, both the fixed beamformer and the blocking matrix need to be adaptive in a certain sense, otherwise the overall performance greatly depends on assembly accuracy of the microphone array.

However, the GSC produces a non-distorted and enhanced estimate of the speech signal, and also provides a good estimate of the interfering noise in case of low target leakage. Since the user's distance to the microphone array is assumed to be low in this application, the sound field will be mostly directional. Therefore target leakage is expected to be minimal once the general direction of the speaker is known. The same assumption was made for other multichannel speech enhancement systems like [35], [36] or [12].

To be able to deal with these problems, the following conditions need to be fulfilled:

- The mainlobe of the fixed beamformer is aligned towards the desired speech source, so that the largest part of the speech energy passes the fixed beamformer, and not the blocking matrix.
- The speaker is close enough to the array to attain a low target leakage, so that the GSC does not cancel out the speech signal.
- The AIC adapts quickly enough to provide an accurate estimate $Y_{AIC}(j\Omega)$ of the unwanted signals. This estimate can then be used in the postprocessing stage.
- The postprocessing stage has to compensate for the poor spatial selectivity of the GSC at low frequencies.

The first condition is subject to *source location*, which is covered in chapter 4. The second condition has to be ensured by the application. The third condition is dependent of the spatial characteristics of the noise field. For example, the AIC has to adapt quickly enough to cancel out moving noise sources like passing-by cars. The adaption speed can be controlled with the NLMS algorithm. And the fourth condition is subject to *multichannel postfiltering*, which is covered in chapter 5.

4 Acoustic Source Localization

4.1 Problem Formulation

As a basis for the fixed beamformer and the blocking matrix of the GSC, the ATFs or RTFs of the source location are needed. Another option is to use delay-only paths to the source. Both possibilities are investigated in this section.

As explained in chapter 2, the ATFs contain the acoustic path to the true location of the desired source, whilst the RTFs contain this path with respect to a reference microphone. Using the latter has the benefit of having shorter FIR-filters, which makes the estimation task easier. Furthermore, estimating the true ATFs requires to use a dereverberation algorithm algorithm, which introduces additional complexity. From a practical viewpoint, removing the reverberation of the user's voice is not necessary in this application.

4.2 Estimating the Relative Transfer Function

4.2.1 Weighted Least Squares

One of the earliest RTF estimation techniques was proposed by Shalvi and Weinstein [37]. It is basically a system identification method that assumes stationary noise signals, an instationary speech signal, and slowly time-varying RTFs being subject to the FIR assumption. As mentioned previously, the FIR assumption is valid for the given setup, despite the fact that a RTF is the ratio of two ATFs.

According to the system model from equation 2.14, the cross-PSD between the first and the m-th microphone can be written as

$$\Phi_{Z_1 Z_m}(j\Omega) = \tilde{A}_m(j\Omega)\Phi_{Z_m Z_m}(j\Omega) + \Phi_{N_m Z_m}(j\Omega) + \epsilon(j\Omega), \tag{4.1}$$

where $\epsilon(j\Omega)$ represents the error between the real cross-PSD $\Phi_{N_m Z_m}(j\Omega)$ and its estimate

$$\epsilon(j\Omega) = \hat{\Phi}_{N_m Z_m}(j\Omega) - \Phi_{N_m Z_m}(j\Omega),$$

and the RTF $\tilde{A}_m(j\Omega)$ between the m-th and the first microphone. By using an observation window of K frames, equation 4.1 can be expressed as

$$\begin{bmatrix} \Phi^1_{Z_1Z_m}(j\Omega) \\ \Phi^2_{Z_1Z_m}(j\Omega) \\ \vdots \\ \Phi^K_{Z_1Z_m}(j\Omega) \end{bmatrix} = \begin{bmatrix} \Phi^1_{Z_mZ_m}(j\Omega) & 1 \\ \Phi^2_{Z_mZ_m}(j\Omega) & 1 \\ \vdots & \vdots \\ \Phi^K_{Z_mZ_m}(j\Omega) & 1 \end{bmatrix} \begin{bmatrix} \tilde{A}_m(j\Omega) \\ \Phi_{N_mZ_m}(j\Omega) \end{bmatrix} + \begin{bmatrix} \epsilon^1(j\Omega) \\ \epsilon^2(j\Omega) \\ \vdots \\ \epsilon^K(j\Omega) \end{bmatrix}. \tag{4.2}$$

The weighted least squares (WLS) solution of $\tilde{A}_m(j\Omega)$ [37] is obtained by

$$\hat{\tilde{A}}_m(j\Omega) = \frac{\left\langle \hat{\Phi}_{N_mZ_m}(j\Omega)\hat{\Phi}_{Z_mZ_m}(j\Omega) \right\rangle - \left\langle \hat{\Phi}_{N_mZ_m}(j\Omega) \right\rangle \left\langle \hat{\Phi}_{Z_mZ_m}(j\Omega) \right\rangle}{\left\langle \hat{\Phi}^2_{Z_mZ_m}(j\Omega) \right\rangle - \left\langle \hat{\Phi}_{Z_mZ_m}(j\Omega) \right\rangle^2}, \tag{4.3}$$

with the averaging function

$$\left\langle \Phi(j\Omega) \right\rangle = \frac{\sum_{k=1}^{K} W_k \Phi^k}{\sum_{k=1}^{K} W_k}, \tag{4.4}$$

where W_k is the k-th value of a weighing function. In [37], a window function that minimizes the covariance of the solution is used. However, the method has several drawbacks when speech is used as excitation signal for the estimation of $\hat{\tilde{A}}_m(j\Omega)$:

Speech is sparse in both time and frequency, resulting in a lot of observations $\hat{\Phi}_{Z_mZ_m}(j\Omega)$ and $\hat{\Phi}_{N_mZ_m}(j\Omega)$ not contributing new information to the averaging process defined in equation 4.4. Therefore, the observation interval has to be very long in order to produce good results. But the RTFs $\hat{\tilde{A}}_m(j\Omega)$ are assumed to be constant over this entire observation interval. This is not be the case when the speaker is moving. Hence, the tracking capabilityis limited and unusable in a real-time application.
To overcome these problems, Cohen modified this approach in [38], where only frames containing speech are used in a real-time algorithm. Therefore, a speech presence probability estimator, presented in [39], is used.

First, the cross-PSD of the noisy observations are calculated recursively with

$$\hat{\Phi}_{Z_1Z_m}(k, j\Omega) = \alpha_s \hat{\Phi}_{Z_1Z_m}(k-1, j\Omega) + (1-\alpha_s) Z_1(k, j\Omega) Z_m^*(k, j\Omega), \tag{4.5}$$

where α_s is a smoothing time-constant with $0 \leq \alpha_s < 1$. The periodogram of the desired speech signal at each microphone m is estimated as

$$\hat{\Phi}_{S_mS_m}(k, j\Omega) = \alpha_s \hat{\Phi}_{S_mS_m}(k-1, j\Omega) + (1-\alpha_s) \left[G^{p(k,j\Omega)}(k, j\Omega) |Z_m(k, j\Omega)| \right]^2, \tag{4.6}$$

where $p(k, j\Omega)$ is a time and frequency-dependent speech probability factor, and $G(k, j\Omega)$ is a real-valued gain factor for suppressing bins containing only noise. Both are obtained using the *improved minima-controlled recursive averaging* (IMCRA) noise power estimation method

in [39]. Next, the cross-PSD of the noise-only signals are estimated with

$$\hat{\Phi}_{N_1N_m}(k,j\Omega) = \tilde{\alpha}_n(k,j\Omega)\hat{\Phi}_{N_1N_m}(k-1,j\Omega) + (1-\tilde{\alpha}_n(k,j\Omega))Z_1(k,j\Omega)Z_m^*(k,j\Omega). \quad (4.7)$$

The smoothing constant $\tilde{\alpha}_n(k,j\Omega)$ is given as

$$\tilde{\alpha}_n(k,j\Omega) = \alpha_n + (1-\alpha_n)p(k,j\Omega). \quad (4.8)$$

With these PSDs, the RTFs are recursively estimated using

$$\hat{\tilde{A}}_m(k,j\Omega) = \hat{\tilde{A}}_m(k-1,j\Omega) + \mu\hat{I}(k,j\Omega)\hat{\Phi}^{-1}_{S_mS_m}(k,j\Omega)\hat{\epsilon}(k,j\Omega), \quad (4.9)$$

where

$$\hat{I}(k,j\Omega) = \begin{cases} 1, & \text{if } p(k,j\Omega) \geq p_0 \\ 0, & \text{otherwise} \end{cases} \quad (4.10)$$

represents a frequency-dependent *voice activity detector* (VAD), returning a binary decision whether a frequency bin contains speech or noise. The value p_0 is a predetermined constant $0 \leq p_0 < 1$ and controls the trade-off between speech detection and false alarm probabilities. The estimation error $\hat{\epsilon}(k,j\Omega)$ is given by

$$\hat{\epsilon}(k,j\Omega) = \hat{\Phi}_{Z_1Z_m}(k,j\Omega) - \hat{\Phi}_{N_1N_m}(k,j\Omega) - \hat{\tilde{A}}_m(k-1,j\Omega)\hat{\Phi}_{S_mS_m}(k,j\Omega). \quad (4.11)$$

The update of the RTF $\hat{\tilde{A}}_m(k,j\Omega)$ is only carried out if a desired signal energy is present, indicated by $\hat{I}(k,j\Omega) \neq 0$. This update procedure has to be carried out for all M microphones. Experimentally obtained values for α_s and α_n are given by $\alpha_s = \alpha_n = 0.85$ [40].

4.2.2 Independent Component Analysis

The RTFs are used to construct the blocking matrix, whose purpose it is to produce the noise references $\boldsymbol{U}(j\Omega)$ of the GSC. As shown, they should be free of the desired speech signal and only contain the noise components. Therefore, generating these noise references can also be regarded as a *blind source separation* (BSS) task. In [41], the blocking matrix is replaced by an *Independent Component Analysis* (ICA), which generates the noise references based on the principle of *mutual statistical independence*.

The least-squares optimization algorithms used so far (NLMS and WLS) rely on correlations. For example, uncorrelatedness between speech and noise has been assumed so far. Considering random variables, correlations can easily be recognized as second order moments. The k-th moment of a random variable x is defined as [42]

$$m'_k(x) = \mathbb{E}\{x^k\} = \int_{-\infty}^{\infty} \xi^k p_x(\xi) d\xi. \quad (4.12)$$

For Gaussian distributions, only the first and second order moments exist, i.e. all moments $m'_k = 0$ for $k > 2$. However, speech is usually modeled as Laplace distribution [1], therefore it is

beneficial to exploit the higher order moments as well. Instead of relying only on correlations, the *independent component analysis* (ICA) uses the richer concept of statistical independence. A set of random variables $x_1, \cdots, x_M$ is said to be independent, iff their joint density is equal to the product of their marginal densities

$$p_{x_1,x_2,\cdots,x_m}(x_1, x_2, \cdots, x_m) = \prod_{m=1}^{M} p_{x_m}(x_m). \tag{4.13}$$

For separating M sources, at least M sensors are needed. In the context of BSS, both the speech and the unwanted noise signals are regarded as *sources*. For simplicity, assume M sources $S_m(j\Omega)$ and M observations (microphone signals) $Z_m(j\Omega)$, both considered as random variables. All sources are said to be latent variables, as they cannot be observed directly. In accordance with the signal model defined in section 2.3, only convoluted mixtures of the sources can be observed at the microphones. In the frequency domain, these convolutions can be written as multiplications in form of a MIMO system

$$\boldsymbol{Z}(j\Omega) = \boldsymbol{A}(j\Omega)\boldsymbol{S}(j\Omega), \tag{4.14}$$

where $\boldsymbol{A}(j\Omega)$ is an unknown $M \times M$ mixing matrix, containing the ATFs from all sources to all microphones. The source vector $\boldsymbol{S}(j\Omega) = [S_1(j\Omega), \cdots, S_M(j\Omega)]^T$ contains the M sources and $\boldsymbol{Z}(j\Omega) = [Z_1(j\Omega), \cdots, Z_M(j\Omega)]^T$ is the observation vector. In contrast to the signal model used so far, this definition assumes more than one source signal $S(j\Omega)$. That is because the ICA cannot distinguish between noise and speech sources, therefore the noise is also regarded as a source. A straightforward approach in blind source separation to undo the mixing process is using an unmixing matrix $\boldsymbol{W} = \boldsymbol{A}^{-1}$ to obtain $\boldsymbol{Y}(j\Omega)$ estimates of the sources

$$\boldsymbol{Y}(j\Omega) = \boldsymbol{W}(j\Omega)\boldsymbol{Z}(j\Omega). \tag{4.15}$$

In [43], the unmixing matrix is estimated in the frequency domain by an ICA algorithm, as illustrated in figure 4.1 for $M = 2$ components. The unmiximg matrix $\boldsymbol{W}(j\Omega)$ is optimized so that the outputs $Y_1(j\Omega)$ and $Y_2(j\Omega)$ are mutually independent.

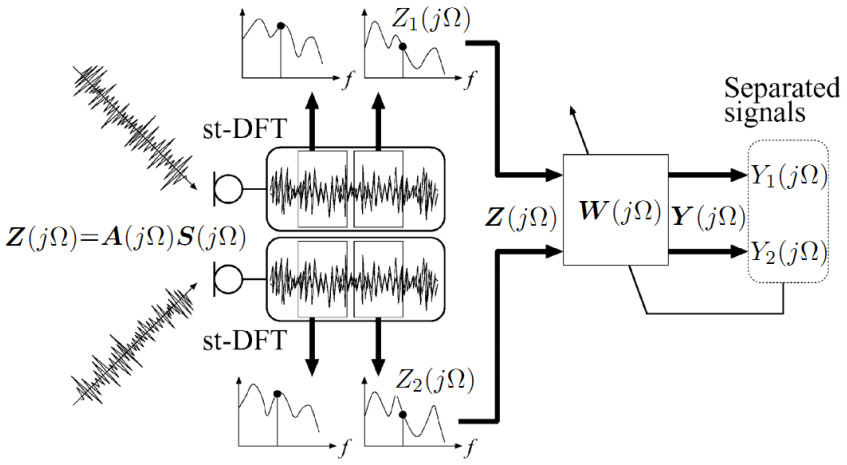

Figure 4.1: Schematic principle of the FD-based ICA, taken from [43].

In the following, the frequency argument $j\Omega$ is omitted for better readability. A very basic

approach for maximizing independence between the outputs Y_m is given by the concept of *maximum likelihood estimation* (MLE) [42]. Inserting equation 4.14 into 4.13 gives the joint density of the observations

$$p_Z(\boldsymbol{Z}) = |\det \boldsymbol{W}| \prod_{m=1}^{M} p_m(S_m) = |\det \boldsymbol{W}| \prod_{m=1}^{M} p_m(w_m^T \boldsymbol{Z}), \tag{4.16}$$

where $[w_1, \cdots, w_M]^T$ are the columns of the unmixing matrix $\boldsymbol{W}$. Assuming $\boldsymbol{Z}(1), \cdots, \boldsymbol{Z}(K)$ i.i.d. samples of the observation vector, the likelihood function is obtained as the product evaluated at these K points

$$L(\boldsymbol{W}) = \prod_{k=1}^{K} \prod_{m=1}^{M} p_m(w_m^T \boldsymbol{Z}(k)) |\det \boldsymbol{W}|. \tag{4.17}$$

For a large sample size K, numerical stability is ensured by taking the logarithm of the likelihood function, which turns the products into sums. The MLE-optimal solution maximizes equation 4.17 and is found by

$$\frac{\partial}{\partial \boldsymbol{W}} \ln(p_Z(\boldsymbol{Z}|\boldsymbol{W})) \Big|_{\boldsymbol{W}=\boldsymbol{W}_{MLE}} \stackrel{!}{=} 0. \tag{4.18}$$

In [41], a *natural gradient* algorithm to iteratively solve equation 4.18 is used. It is given by

$$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} + \mu \big[\boldsymbol{I} - \big\langle \Phi(\boldsymbol{Z}) \boldsymbol{Z}^H \big\rangle_K\big] \boldsymbol{W}_{t-1}, \tag{4.19}$$

where μ is the learning rate, $\langle \cdot \rangle_K$ denotes the averaging operator over K samples, and $\Phi(\cdot)$ is a nonlinear contrast function, given as

$$\Phi(\boldsymbol{Z}) = \tanh(\mathrm{Re}\{\boldsymbol{Z}\}) + j \tanh(\mathrm{Im}\{\boldsymbol{Z}\}). \tag{4.20}$$

Apart from the MLE approach, many other ICA algorithms exist [42]. Inherent to all of them is the permutation and scaling ambiguity on the columns of the unmixing matrix $\boldsymbol{W}$. Since the ICA is performed on frequency domain data $\boldsymbol{Z}(j\Omega)$, the permutation and scaling problem applies to each frequency bin. The scaling problem has been solved in [41] by normalizing the unmixing matrix to

$$\boldsymbol{W}_t = \mathrm{diag}(\boldsymbol{W}_t^{-1}) \boldsymbol{W}_t, \tag{4.21}$$

where *diag* denotes the diagonal matrix. When each source $S_1 \cdots S_M$ originates from a different direction, the outputs of the ICA $Y_1 \cdots Y_M$ can be reordered according to their *Direction Of Arrival* (DOA), which effectively solves the permutation problem [43]. The DOA estimation is introduced in the next chapter. The numerical complexity of the MLE search and the potentially long observation window K are the most challenging issues in successfully applying the ICA. Also, a VAD might be necessary in order to adapt $\boldsymbol{W}$ only during speech presence.

4.3 Estimating the Direction Of Arrival

The most simple means to model a RTF is to use a single delay and magnitude instead of a full FIR filter. This way, the complexity of estimating the entire RTF boils down to a much simpler *Direction Of Arrival* (DOA) estimation. Also, estimating a single number from lots of input data is generally more robust, as measurement errors may level out. As a downside, reverberations cannot be modeled anymore. A single delay is tantamount to imposing an ideal direct sound field on the impinging speech signal. As seen in section 2.7 this constraint is only valid if the distance of the speaker to the microphone array is small.
The basic principle of DOA estimation is illustrated in figure 4.2 for a linear array with an inter-microphone distance d. For the assumed near-field scenario, it can be seen that the DOA angle Θ is different for each microphone.

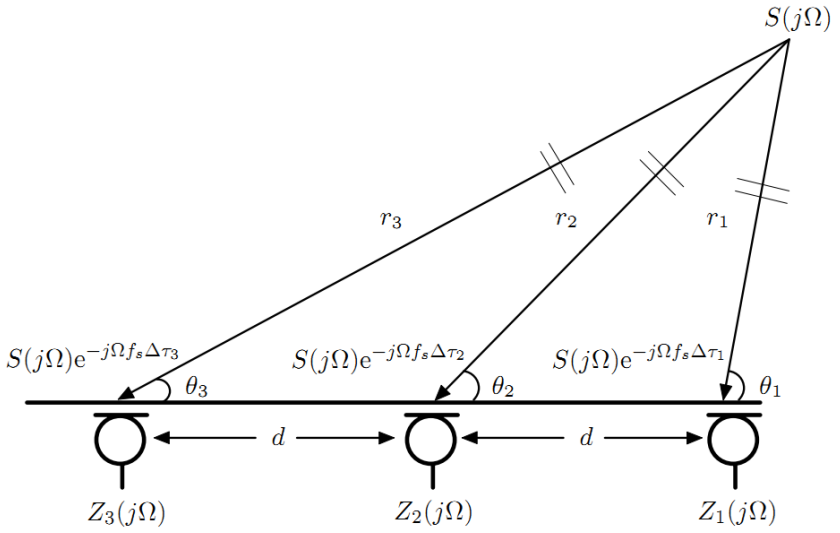

Figure 4.2: DOA principle, taken from [9]. The distance from the source $S(j\Omega)$ to the m^{th} microphone is given as $r_m = c\tau_m$, with c being the speed of sound.

Remembering equation 3.28 from chapter 3.5, ATFs using pure delays are given as

$$\boldsymbol{A}(j\Omega) = \begin{bmatrix} \mathrm{e}^{-j\Omega f_s \tau_1} & \mathrm{e}^{-j\Omega f_s \tau_2} & \dots & \mathrm{e}^{-j\Omega f_s \tau_M} \end{bmatrix}^T. \tag{4.22}$$

As mentioned, only RTFs can be estimated. When using the first microphone as reference again, the RTFs are obtained by dividing each element of equation 4.22 by the first element $\mathrm{e}^{-j\Omega f_s \tau_1}$, resulting in

$$\begin{aligned} \tilde{\boldsymbol{A}}(j\Omega) &= \begin{bmatrix} 1 & \mathrm{e}^{-j\Omega f_s (\tau_2 - \tau_1)} & \dots & \mathrm{e}^{-j\Omega f_s (\tau_M - \tau_1)} \end{bmatrix}^T \\ &= \begin{bmatrix} 1 & \mathrm{e}^{-j\Omega f_s \Delta\tau_2} & \dots & \mathrm{e}^{-j\Omega f_s \Delta\tau_M} \end{bmatrix}^T. \end{aligned} \tag{4.23}$$

Using these RTFs, the the signal model defined in equation 2.14 simplifies to

$$Z_m(j\Omega) = S(j\Omega)\mathrm{e}^{-j\Omega f_s \Delta\tau_m} + N_m(j\Omega). \tag{4.24}$$

Benesty mentions a variety of DOA estimation techniques in [9], all being based on equation

4.24. In the following, principles using *generalized cross correlation* (GCC) and Eigenvectors of the input PSD matrix are introduced.

4.3.1 Smoothed Coherence Transform

The *Smoothed Coherence Transform* (SCOT) is based upon the cross-correlation between the microphone input signal $Z_m(j\Omega)$ and the reference microphone $Z_1(j\Omega)$. Using the simplified signal model from equation 4.24, the cross-correlation can be written as

$$\Phi_{Z_m Z_1}(j\Omega) = \mathbb{E}\{Z_m(j\Omega)Z_1^*(j\Omega)\} = \Phi_{SS}(j\Omega)\mathrm{e}^{-j\Omega f_s \Delta\tau_m}, \tag{4.25}$$

where uncorrelated speech and noise signals are assumed. Also, the noise signals at the first and the m-th microphone are assumed to be uncorrelated. For low frequencies, this assumption does not hold in reality, as observed from the ATF measurements in section 2.7. There is a strong correlation $\Phi_{N_m N_1}(j\Omega)$ for low frequencies, shadowing the desired phase information in equation 4.25. Therefore, low frequencies contribute no information to the DOA estimation process. However, this is not a problem since the DOA is estimated using correlations throughout all frequencies.
In order to overcome fluctuating power levels $\Phi_{SS}(j\Omega)$ of the desired speech signal, the SCOT pre-whitens the microphone signals before their cross-spectrum is computed [9]. It is given as

$$\begin{aligned}\Phi_{Z_m Z_1}^{SCOT}(j\Omega) &= \frac{\mathbb{E}\{Z_m(j\Omega)Z_1^*(j\Omega)\}}{\sqrt{\mathbb{E}\{|Z_m(j\Omega)|^2\}\mathbb{E}\{|Z_1(j\Omega)|^2\}}} \\ &= \frac{\Phi_{SS}(j\Omega)\mathrm{e}^{-j\Omega f_s \Delta\tau_m}}{\sqrt{(\Phi_{SS}(j\Omega)+\Phi_{N_m N_m}(j\Omega))(\Phi_{SS}(j\Omega)+\Phi_{N_1 N_1}(j\Omega))}}.\end{aligned} \tag{4.26}$$

Assuming the same noise power level $\Phi_{NN}(j\Omega)$ at each microphone, equation 4.26 simplifies to

$$\Phi_{Z_m Z_1}^{SCOT}(j\Omega) = \mathrm{e}^{-j\Omega f_s \Delta\tau_m}\Big[\frac{iSNR}{1+iSNR}\Big], \tag{4.27}$$

with the input SNR

$$iSNR = \frac{\Phi_{SS}(j\Omega)}{\Phi_{NN}(j\Omega)}$$

at each microphone. Obviously the cross-spectrum is scaled with the input SNR. If it is large, $\Phi_{Z_m Z_1}^{SCOT}(j\Omega)$ can be approximated as

$$\Phi_{Z_m Z_1}^{SCOT}(j\Omega) \approx \mathrm{e}^{-j\Omega f_s \Delta\tau_m}. \tag{4.28}$$

This simplification is identified as a Dirac pulse in time domain, occurring at time lag $\Delta\tau_m$

$$r_{z_m z_1}^{SCOT}(t) = \delta(t - \Delta\tau_m). \tag{4.29}$$

The time lag $\Delta\tau_m$ resembles the relative *time difference of arrival* of the speech signal between the m-th and the first microphone. Using a finite sampling rate f_s, the inverse FFT of equation

4.28 yields a sinc-pulse due to the sampling theorem

$$r_{z_m z_1}^{SCOT}(k) = \frac{\sin(\pi(k - f_s \Delta\tau_m))}{\pi(k - f_s \Delta\tau_m)}, \tag{4.30}$$

where k is the sample index. For a sampling rate of $f_s = 16kHz$ and an inter-microphone distance $d = 5cm$, the maximum possible time lag is given by $K = \lceil f_s \frac{d}{c} \rceil = 3$ samples. A resolution of only 3 integer samples allows for $2K + 1 = 7$ DOA values, which might be too coarse for speaker tracking. Therefore, $\Delta\tau_m$ has to be estimated in sub-sample space. This can be done by fitting a parabola or a Gaussian to the measured data points $r_{z_m z_1}^{SCOT}(k)$. However, for small array apertures a sinc kernel is recommended for greater accuracy [44]. This sinc kernel is given as

$$\tilde{r}_{z_m z_1}^{SCOT}(k) = \frac{\sin(\pi(k - f_s \Delta\tilde{\tau}_m))}{\pi(k - f_s \Delta\tilde{\tau}_m)}, \tag{4.31}$$

where $\Delta\tilde{\tau}_m$ denotes a predefined delay. The optimal delay in a MSE sense is determined by a quadratic cost function

$$J(\Delta\tilde{\tau}_m) = \frac{1}{\frac{1}{2K}\sum_{k=-K}^{K} \left| r_{z_m z_1}^{SCOT}(k) - \tilde{r}_{z_m z_1}^{SCOT}(k) \right|^2}. \tag{4.32}$$

The sum from $-K$ to K samples covers all possible DOAs from $-90°$ to $+90°$, as shown in figure 4.2. Maximizing this cost function identifys the most accurate $\Delta\tilde{\tau}_m$ by using

$$\Delta\hat{\tilde{\tau}}_m = \arg\max_{\Delta\tilde{\tau}_m} J(\Delta\tilde{\tau}_m). \tag{4.33}$$

Once the optimal time lag has been determined, the corresponding DOA angle Θ_m for the linear array shown in figure 4.2 can be computed with

$$\Theta_m = \sin^{-1}\Big(\frac{\Delta\hat{\tilde{\tau}}_m \cdot c}{(m-1)d}\Big), \tag{4.34}$$

where $(m-1)d$ may be recognized as the absolute distance between the first and the m-th microphone. Obviously, the valid range for $\Delta\tilde{\tau}_m$ extends from $-(m-1)d$ to $(m-1)d$. Equation 4.33 can be realized by dividing this range into a linear grid of I time delays. Then, the i-th time delay is given as $\Delta\tilde{\tau}_{m,i} = \frac{(m-1)d}{c}\left(\frac{2i}{I} - 1\right)$.

For the array in figure 4.2, where $d = 5cm$ and the search grid is divided into $I = 50$ equidistant chunks, the grid size for the second microphone ($m = 2$) is given as $d_{grid} = \frac{2(m-1)d}{I} = 2mm$. With mechanical tolerances like microphone placing in mind, this value appears to be a reasonable choice. Also, the computational cost for this grid search over 50 elements is still relatively small. Further, the delay values $\Delta\tilde{\tau}_{m,i}$ can be precomputed and stored in a table, and the grid search does not have to be executed for every frame of data. A few times per second is fairly enough, depending on the expected movement of the speaker in front of the array. A practical example using real speech data is given in figure 4.3. The cost function of equation 4.33 is plotted over a search grid of $I = 50$ possible time delays. In accordance to the speech arriving at $\Theta_m = 45°$ at the array, the cost function shows a maximum at the corresponding

angle. The speech data was generated using one of the ATFs measured in chapter 2.7. Using proper scaling, $J(\Delta\tilde{\tau}_m)$ can be used as *Speech Presence Probability* (SPP) for plotting.

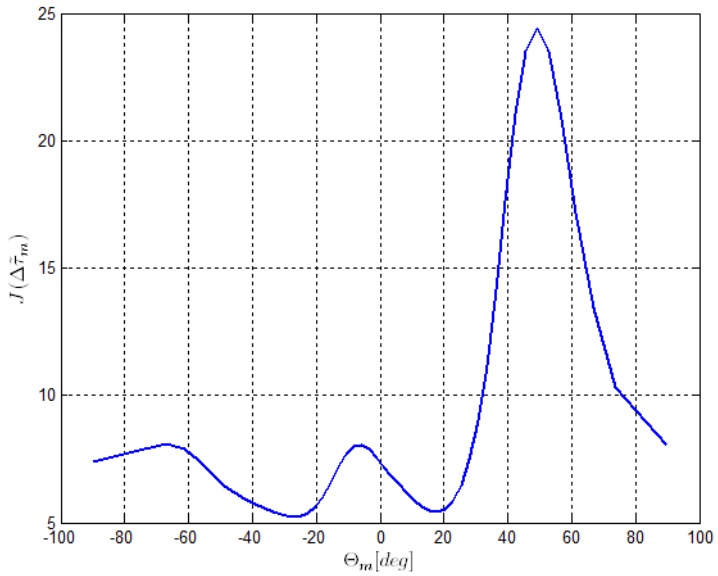

Figure 4.3: Cost function of equation 4.33 for speech arriving at $\Theta_m = 45°$, $d = 5cm$ and $I = 50$.

4.3.2 Phase Transform

Considering equation 4.27, it becomes clear that the DOA information is merely contained in the phase term rather than in the amplitude. The *Phase Transform* (PHAT) [9] modifies the cross correlation in equation 4.26 to

$$\Phi^{PHAT}_{Z_m Z_1}(j\Omega) = \frac{\Phi_{Z_m Z_1}(j\Omega)}{|\Phi_{Z_m Z_1}(j\Omega)|} = \frac{\Phi_{SS}(j\Omega)\mathrm{e}^{-j\Omega f_s \Delta\tau_m}}{|\Phi_{SS}(j\Omega)|} = \mathrm{e}^{-j\Omega f_s \Delta\tau_m}. \tag{4.35}$$

Unlike the SCOT, the PHAT does not rely on the input SNR, as may be seen from equation 4.35. Therefore, the PHAT performs better in low SNR conditions [9]. The remaining DOA estimation procedure is the same as for the SCOT, from equation 4.29 onwards.

4.3.3 Multiple Signal Classification

Another means to estimate the DOA is to use the eigenvectors of the spatial correlation matrix $\boldsymbol{\Phi}_{ZZ}(j\Omega)$. These types of algorithms are known as *Multiple Signal Classification* (MUSIC) [9]. Again, each time delay between the m-th and the first microphone is estimated separately.

By writing the signal model of equation 4.24 for the first and the m-th microphone in vector notation, we get

$$\boldsymbol{Z}_m(j\Omega) = \boldsymbol{\varsigma}(j\Omega, \Delta\tau_m)S(j\Omega) + \boldsymbol{N}_m(j\Omega), \tag{4.36}$$

where

$$\begin{aligned}\boldsymbol{Z}_m(j\Omega) &= \begin{bmatrix} Z_m(j\Omega) & Z_1(j\Omega) \end{bmatrix}^T \\ \boldsymbol{\varsigma}(j\Omega, \Delta\tau_m) &= \begin{bmatrix} \mathrm{e}^{-j\Omega f_s \Delta\tau_m} & 1 \end{bmatrix}^T \\ \boldsymbol{N}_m(j\Omega) &= \begin{bmatrix} N_m(j\Omega) & N_1(j\Omega) \end{bmatrix}^T.\end{aligned}$$

The spatial correlation matrix of $\boldsymbol{Z}_m(j\Omega)$ can be written as

$$\Phi_{Z_m Z_1} = \Phi_{SS}\boldsymbol{\varsigma}(\Delta\tau_m)\boldsymbol{\varsigma}^H(\Delta\tau_m) + \Phi_{NN}\boldsymbol{I}_{2\times 2}, \tag{4.37}$$

where the frequency argument $j\Omega$ has been omitted for enhanced readability. The noise is assumed to be uncorrelated between the microphones and of same power Φ_{NN}, similar to the PHAT and SCOT algorithms. An eigenvalue decomposition of $\Phi_{Z_m Z_1}$ gives

$$\Phi_{Z_m Z_1} = \boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^H, \tag{4.38}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}\begin{bmatrix}\lambda_1 & \lambda_2\end{bmatrix}$ are the eigenvalues of $\boldsymbol{\Phi}_{Z_m Z_1}$. $\lambda_1 = 2\Phi_{SS} + \Phi_{NN}$ and $\lambda_2 = \Phi_{NN}$. It is worth noting that $\lambda_1 > \lambda_2$. The associated eigenvectors are given as $\boldsymbol{E} = \begin{bmatrix}\boldsymbol{e}_1 & \boldsymbol{e}_2\end{bmatrix}$.

From equation 4.36 it can be easily seen that the matrix $\Phi_{SS}\boldsymbol{\varsigma}(\Delta\tau_m)\boldsymbol{\varsigma}^H(\Delta\tau_m)$ is positive semi-definite and of rank 1. Its only non-zero eigenvalue is $2\Phi_{SS}$. Therefore, the second eigenvector $\boldsymbol{e}_2$ is subject to

$$\Phi_{Z_m Z_1}\boldsymbol{e}_2 = \lambda_2\boldsymbol{e}_2 = \Phi_{NN}\boldsymbol{e}_2. \tag{4.39}$$

Inserting into equation 4.37 gives

$$\Phi_{Z_m Z_1}\boldsymbol{e}_2 = \left[\Phi_{SS}\boldsymbol{\varsigma}(\Delta\tau_m)\boldsymbol{\varsigma}^H(\Delta\tau_m) + \Phi_{NN}\boldsymbol{I}_{2\times 2}\right]\boldsymbol{e}_2. \tag{4.40}$$

Considering both equations 4.39 and 4.40, it is obvious that

$$\Phi_{SS}\boldsymbol{\varsigma}(\Delta\tau_m)\boldsymbol{\varsigma}^H(\Delta\tau_m)\boldsymbol{e}_2 = \boldsymbol{0}_{1\times 2} \tag{4.41}$$

must hold. This result can be further simplified to

$$\boldsymbol{e}_2^H\boldsymbol{\varsigma}(\Delta\tau_m) = 0. \tag{4.42}$$

This equation indicates that the eigenvector associated with the smaller eigenvalue of $\Phi_{Z_m Z_1}$ is orthogonal to the vector $\boldsymbol{\varsigma}(\Delta\tau_m)$ [9], which contains the actual DOA information. Exploiting this observation, the MUSIC cost function can be written as

$$J(\Delta\tilde{\tau}_m) = \sum_{\Omega=0}^{2\pi f_s} \frac{1}{\left|\boldsymbol{e}_2^H(j\Omega)\boldsymbol{\varsigma}(\Delta\tilde{\tau}_m, j\Omega)\right|^2}, \tag{4.43}$$

where averaging over all frequencies ensures that the global, optimal time lag $\Delta\hat{\tilde{\tau}}_m$ is detected,

i.e.

$$\Delta\hat{\tilde{\tau}}_m = \arg\max_{\Delta\tilde{\tau}_m} J(\Delta\tilde{\tau}_m). \tag{4.44}$$

Similar to SCOT and PHAT, a search range containing possible realizations of $\varsigma(\Delta\tilde{\tau}_m, j\Omega)$ has to be defined. The computational cost for MUSIC is significantly larger than for the other two, since an eigenvector decomposition of $\mathbf{\Phi}_{Z_m Z_1}(j\Omega)$ needs to be done for each frequency. By averaging the cost function over all frequencies, the peak occuring for the optimal delay might be less well-defined [9]. Therefore, also a wideband modification of the MUSIC algorithm is presented in [9]. But it is performed in time domain, which clearly results in an even higher computational cost.

Similar to the SCOT algorithm, an example showing the cost function of equation 4.43 over a set of 50 possible angles Θ_m is given in figure 4.4. Again, the speech signal arrives at $\Theta_m = 45°$ at the microphone array. The highest peak in the cost function correctly identifies the true DOA angle. The peak is sharper than for the SCOT algorithm, yet also a local maximum appears at about 25°.

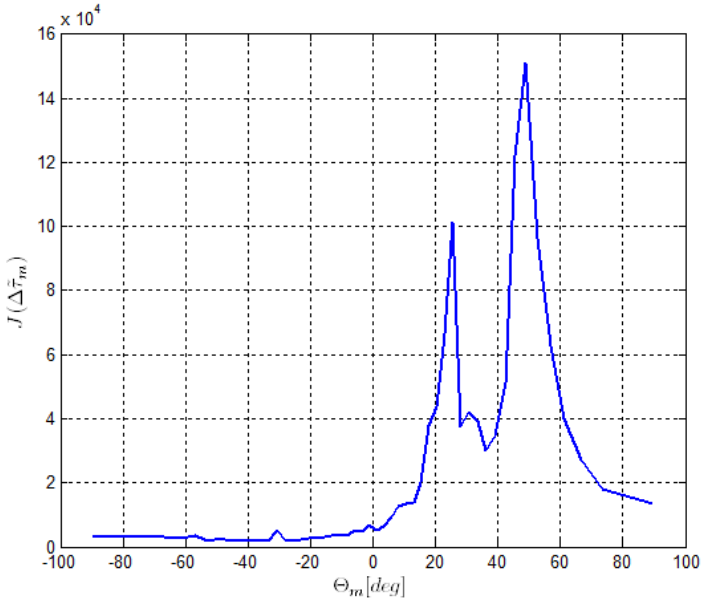

Figure 4.4: Cost function of equation 4.43 visualized for speech arriving at $\Theta_m = 45°$, $d = 5cm$ and $I = 50$.

4.3.4 Magnitude Estimation

When using DOAs for RTF approximation, the main direction of the source signal can be estimated using one of the three algorithms above. However, the amplitude spectrum of the DOA is modeled as 1, since $|e^{-j\Omega f_s \tau_m}| = 1$ for all frequencies. This imposes that the signals at the microphones have equal power at all frequencies at all times. In a practical implementation, this constraint can never be fulfilled since the microphone sensitivity varies depending on manufacturing tolerances. Low-cost electret microphones show a variance of about 6dB throughout the usable frequency range. Even when using measurement-grade equipment, the variance is still several dB. Furthermore, when the speaker is close to the array, the nearest microphone receives the loudest portion of the sound. According to equation 2.7, sound pressure increases

linearly with decreasing distance, therefore the true magnitude spectrum of the RTFs cannot be neglected.

According the definition of RTFs from equation 2.15, the magnitude ratio between the ATFs of the m-th and the first microphone has to be determined. A good approximation has been found experimentally by taking the square root the ratio of the PSD estimates of the microphone signals, i.e.

$$|\tilde{A}_m(j\Omega)| \approx \sqrt{\frac{\Phi_{Z_m Z_m}(j\Omega)}{\Phi_{Z_1 Z_1}(j\Omega)}} = \sqrt{\frac{\Phi_{SS}(j\Omega)|A_m(j\Omega)|^2 + \Phi_{N_m N_m}(j\Omega)}{\Phi_{SS}(j\Omega)|A_1(j\Omega)|^2 + \Phi_{N_1 N_1}(j\Omega)}}. \tag{4.45}$$

If the additive noise PSDs in equation 4.45 are small, this estimate is close to the true ATF magnitude ratio. Otherwise, the result has to be clamped to stay within meaningful bounds, for example $\pm 12dB$. Further, this magnitude estimation procedure is restricted to the presence of a speech signal. This can be detected using a VAD. The final estimate for the RTF is obtained by combining the magnitude estimate with the DOA estimate from one of the above algorithms, i.e.

$$\tilde{A}_m(j\Omega) = \mathrm{e}^{-j\Omega f_s \Delta\hat{\tau}_m} \sqrt{\frac{\Phi_{Z_m Z_m}(j\Omega)}{\Phi_{Z_1 Z_1}(j\Omega)}}. \tag{4.46}$$

This type of RTF combines the robustness of a DOA as single number criterion with the simplicity of a RTF magnitude estimation. It has a clear advantage over the more complex estimation of the full RTFs from chapter 4.2.2.

4.4 Voice Activity Detection

All of the RTF and DOA algorithms introduced so far may only be carried out in the presence of the desired speech signal. Clearly, without the speech signal the location of the speaker cannot be estimated. Therefore, a *Voice Activity Detector* (VAD) may be used to decide whether the desires speech signal is present or not. Using a VAD in speech enhancement applications is still a controversy, as false decisions have a devastating impact on the overall speech quality. However, using a VAD for the source location task has no direct influence on speech quality. Therefore, a certain percentage of false decisions of the VAD is negligible.

Utilizing the results from the ATF measurements in chapter 2.7, a good VAD can be built based upon spatial coherence [10]: The mean squared coherence is large during speech activity, and low during speech absence. This is especially true for high frequencies, as shown in figures 2.10 and 2.9, respectively.

Using the definition of the squared coherence in equation 2.19, the average over an entire frequency range can be formulated as

$$\Gamma_{Z_i Z_j} = \frac{1}{\Omega_1 - \Omega_0} \sum_{\Omega=\Omega_0}^{\Omega_1} |\gamma_{Z_i Z_j}(j\Omega)|^2, \tag{4.47}$$

which is known as *Mean Squared Coherence* (MSC). Especially the lower bound $\Omega_0 = 2\pi f_0$ has to be chosen carefully, in order to avoid frequencies that also have a high coherence in the

absence of speech. A good choice for Ω_0 is determined by locating the first zero of the MSC for the ideal diffuse noise sound field from equation 2.20, illustrated in figure 2.9. It is found by $\Omega_0 = \frac{f_0}{f_s} = \frac{c}{2d \cdot f_s}$. For the given array geometry in figure 2.9 of $d = 5cm$, this lower frequency is at $f_0 = 3430Hz$. The upper bound Ω_1 is only limited by the frequency range of human speech production. Any two microphones i and j of the microphone array can be used to evaluate equation 4.47.

5 Multichannel Postfiltering

5.1 Postfiltering Concepts

Unlike traditional speech enhancement methods, beamforming imposes only little distortion on the speech signal. Therefore, a significant improvement in speech quality can be achieved. However, when the noise field is spatially incoherent or diffuse the noise reduction capabilities of a beamformer are insufficient. An example has been given for the MVDR beamformer in section 3.4. In illustration 3.6, it can be seen that the noise reduction performance is poor for low frequencies. Therefore, additional noise canceling is required. Usually, this is done in form of a *postfilter*, which operates on the output of the beamformer. The earliest postfilters have been derived by multichannel Wiener filtering. The optimal MISO Wiener solution for the signal model given in section 2.3 is derived in [2] and [11]. It is given as

$$\boldsymbol{W} = \frac{\boldsymbol{\Phi}_{NN}^{-1}\boldsymbol{A}}{\boldsymbol{A}^H\boldsymbol{\Phi}_{NN}^{-1}\boldsymbol{A}} \cdot \frac{\Phi_{SS}}{\Phi_{SS} + \left[\boldsymbol{A}^H\boldsymbol{\Phi}_{NN}^{-1}\boldsymbol{A}\right]^{-1}}, \tag{5.1}$$

where the frequency dependency $j\Omega$ has been omitted. The first term can be recognized as the MVDR beamformer from section 3.4, and the second term resembles a single channel Wiener filter for noise suppression. Based on this result, a single channel Wiener filter seems to be sufficient to enhance the overall performance. However, after a decade of research in the field of acoustic beamforming, a multichannel postfilter has shown to be superior: Single channel noise reduction filters exhibit a low performance when the noise is highly non-stationary, but this type of noise dominates in most practical use cases. Multichannel noise reduction filters surpass this limit by using the spatial information made available by the beamformer. Thereby the postfilter does not depend on the statistics of the noise signal, but rather on the spatial selectivity of the beamformer.

In this chapter, three of the most prominent multichannel postfilters are introduced: The *Transient Beam to Reference Ratio* presented by Cohen [36], the *Direct to Diffuse Ratio* method from [13] and Benesty's *Multichannel Speech Presence Probability* presented in [14]. Some of the methods are based on single channel noise suppression. Therefore, single channel methods are introduced first.

5.2 Single-Channel Speech Enhancement

Single channel speech enhancement algorithms have a rich history for almost thirty years. Most methods are based on the estimation of the noise power, followed by spectral subtraction. The most prominent concepts for estimating the noise power are the *Minimum Statistics* approach [45] and the *Improved Minima Controlled Recursive Averaging* (IMCRA) [40]. *Ephraim* and *Malah* proposed the *minimum mean-squared error logarithmic spectral-amplitude* (MMSE-LSA) estimator for spectral enhancement [46]. Many speech enhancement algorithms are based on this work or one of its many modifications. In the following, these noise estimation and spectral subtraction concepts are briefly introduced.

5.2.1 Minimum Statistics

In [45], Martin introduced the *Minimum Statistics* noise spectrum estimator. It uses optimal smoothing of the noisy input signal power to estimate its noisefloor. As a benefit, it does not need a VAD, but relies completely on the signal statistics instead.

The signal model for most single channel speech enhancement algorithms assumes an additive superposition of a speech signal $S(j\Omega)$ and a noise signal $N(j\Omega)$, i.e.

$$Y(j\Omega) = S(j\Omega) + N(j\Omega). \tag{5.2}$$

Again, speech and noise are assumed to be uncorrelated. Thus, the *power spectrum density* (PSD) of the noisy input calculates to

$$\sigma_y^2(j\Omega) = \sigma_s^2(j\Omega) + \sigma_n^2(j\Omega). \tag{5.3}$$

Using the signal model in equation 5.2, an estimate for the true noise PSD $\sigma_n^2(j\Omega)$ is given by the following smoothing operation

$$P(k, j\Omega) = P(k-1, j\Omega)\alpha(k, j\Omega) + (1 - \alpha(k, j\Omega))|Y(k, j\Omega)|^2, \tag{5.4}$$

where k denotes the frame number. The main idea of the minimum statistics approach is to derive an optimal smoothing parameter $\alpha(k, j\Omega)$, so that $P(k, j\Omega)$ is as close as possible to the true noise PSD $\sigma_n^2(j\Omega)$. This is achieved by minimizing the conditional MSE

$$\mathbb{E}\Big\{ \big(P(k, j\Omega) - \sigma_n^2(k, j\Omega)\big)^2 \Big| P(k-1, j\Omega) \Big\}. \tag{5.5}$$

Setting the first derivative with respect to $\alpha(k, j\Omega)$ to zero yields the optimal time and frequency dependent smoothing parameter

$$\alpha_{OPT}(k, j\Omega) = \frac{1}{1 + \left(\frac{P(k-1, j\Omega)}{\sigma_n^2(k, j\Omega)} - 1 \right)^2}. \tag{5.6}$$

Since $P(k, j\Omega)$ is a smoothed version of the input $|Y(j\Omega)|^2$, the actual noisefloor $\sigma_n^2(j\Omega)$ is smaller during speech activity. Hence, the estimator $P(k, j\Omega)$ is biased. An unbiased estimator

is given by

$$\hat{\sigma}_n^2(k, j\Omega) = P_{min}(k, j\Omega) B_{min}(D, Q_{eq}(k, j\Omega)), \tag{5.7}$$

where $P_{min}(k, j\Omega)$ denotes the smallest value within a window of D recent values of $P(k, j\Omega)$. The function $B_{min}(D, Q_{eq}(k, j\Omega))$ depends on this window length and the inverse normalized variance of $P(k, j\Omega)$, given as

$$Q_{eq}(k, j\Omega) = \frac{2\sigma_n^4(j\Omega)}{\mathrm{var}\{P(k, j\Omega)\}}. \tag{5.8}$$

Due to the minimum search in equation 5.7, the noise estimate $\hat{\sigma}_n^2(k, j\Omega)$ lags D frames behind $P(k, j\Omega)$. Therefore, the estimation accuracy heavily depends on D. Making it too small causes $\hat{\sigma}_n^2(k, j\Omega)$ to closely follow $P(k, j\Omega)$, and thereby speech components may be recognized as noise. Making it too large on the other hand prevents $\hat{\sigma}_n^2(k, j\Omega)$ to follow the true noise PSD quickly enough, hence transient noises may not be suppressed anymore. The overall computational complexity is rather large, therefore the minimum search can also be done iteratively. Further implementation details can be found in [45].

5.2.2 Improved Minima-Controlled Recursive Averaging

In [40], Cohen proposed the *Improved Minima Controlled Recursive Averaging* or IMCRA noise spectrum estimator. Like with the Minimum Statistics approach, an explicit VAD to distinguish noise from speech is not needed. Instead, a time and frequency dependent *speech presence probability* is employed to control a smoothing factor. This factor is used to estimate the noise power from the noisy input signal. The speech presence probability itself is estimated using two recursive averaging stages.

Using the same signal model as in equation 5.2, the first stage is given by

$$P(k, j\Omega) = P(k-1, j\Omega)\alpha_s + (1-\alpha_s)|Y(k, j\Omega)|^2. \tag{5.9}$$

From $P(k, j\Omega)$, the following two SNRs are calculated

$$\gamma_{min}(k, j\Omega) = \frac{|Y(k, j\Omega)|^2}{B_{min} P_{min}(k, j\Omega)}; \quad \zeta(k, j\Omega) = \frac{P(k, j\Omega)}{B_{min} P_{min}(k, j\Omega)}, \tag{5.10}$$

where $P_{min}(k, j\Omega)$ is the minimum of the recent D frames of $P(k, j\Omega)$, similar to Minimum Statistics. B_{min} is a predefined constant depending on D. The second recursive averaging stage is defined similar to the first one

$$\tilde{P}(k, j\Omega) = \tilde{P}(k-1, j\Omega)\alpha_s + (1-\alpha_s)|Y(k, j\Omega)|^2, \tag{5.11}$$

but it is only updated if $\gamma_{min}(k, j\Omega) < \gamma_0$ and $\zeta(k, j\Omega) < \zeta_0$. This way, frequency bins that may contain speech are excluded from the second smoothing stage. Again, two SNRs are calculated

from the result

$$\tilde{\gamma}_{min}(k,j\Omega) = \frac{|Y(k,j\Omega)|^2}{B_{min}\tilde{P}_{min}(k,j\Omega)}; \quad \tilde{\zeta}(k,j\Omega) = \frac{P(k,j\Omega)}{B_{min}\tilde{P}_{min}(k,j\Omega)}, \tag{5.12}$$

where $\tilde{P}_{min}(k,j\Omega)$ is the minimum of the recent D frames of $\tilde{P}(k,j\Omega)$, similar to the first smoothing stage. Next, the *a-priori* speech absence probability estimator is defined as

$$q(k,j\Omega) = \begin{cases} 1, & \text{if } \tilde{\gamma}_{min}(k,j\Omega) \leq 1 \text{ and } \tilde{\zeta}(k,j\Omega) < \zeta_0 \\ \frac{\gamma_1 - \tilde{\gamma}_{min}(k,j\Omega)}{\gamma_1 - 1}, & \text{if } 1 < \tilde{\gamma}_{min}(k,j\Omega) < \gamma_1 \text{ and } \tilde{\zeta}(k,j\Omega) < \zeta_0 \\ 0, & \text{otherwise} \end{cases} \tag{5.13}$$

Based on a Gaussian statistical model [40], the *speech presence probability* is derived

$$p(k,j\Omega) = \left[1 + \frac{q(k,j\Omega)}{1-q(k,j\Omega)}\big(1+\xi(k,j\Omega)\big)\mathrm{e}^{-\nu(k,j\Omega)}\right]^{-1}, \tag{5.14}$$

with the *a-priori* SNR and the *a-posteriori* SNR, respectively

$$\xi(k,j\Omega) = \frac{\sigma_s^2(k,j\Omega)}{\sigma_n^2(k,j\Omega)}; \quad \gamma(k,j\Omega) = \frac{|Y(k,j\Omega)|^2}{\sigma_n^2(k,j\Omega)}, \tag{5.15}$$

and

$$\nu(k,j\Omega) = \gamma(k,j\Omega)\frac{\xi(k,j\Omega)}{1+\xi(k,j\Omega)}. \tag{5.16}$$

Finally, a recursive averaging operation is used to estimate the noise PSD

$$\hat{\sigma}_n^2(k,j\Omega) = \hat{\sigma}_n^2(k-1,j\Omega)\tilde{\alpha}(k,j\Omega) + (1-\tilde{\alpha}(k,j\Omega))|Y(k,j\Omega)|^2, \tag{5.17}$$

where the smoothing factor $\tilde{\alpha}(k,j\Omega)$ is controlled by the speech presence probability using

$$\tilde{\alpha}(k,j\Omega) = \alpha + (1-\alpha)p(k,j\Omega). \tag{5.18}$$

Both the IMCRA and Minimum Statistics noise estimators have a similar performance and numerical complexity. The window length D can be halved for the IMCRA, as the minimum search of the first averaging stage serves as input for the second one. Even though, the same considerations on D apply. In [40], the recursive averaging stages also employ smoothing over frequency to improve the estimation accuracy. In the experiments this has been found to be unnecessary. To reduce the computational cost, this smoothing has been removed. Further implementation details can be found in [40].

5.2.3 Minimum Mean Squared Error Log-Spectral Amplitude estimator

The estimated noise PSD is used for subtracting the noise amplitude from the amplitude spectrum of the input signal, thereby leaving the phase spectrum unmodified. This technique is

known as *Spectral Subtraction*, using a non-causal Wiener filter. Numerous modifications to this Wiener filter have been prepared, probably the most influential one is the *Minimum Mean-Squared Error Logarithmic Spectral-Amplitude* (MMSE-LSA) estimator [46].
First, the algorithm calculates an *a-priori* SNR from the power of the k-th noisy input signal frame $|Y(k, j\Omega)|^2$ and the estimate of the noise power $\hat{\sigma}_n^2(k, j\Omega)$

$$\gamma(k, j\Omega) = \frac{|Y(k, j\Omega)|^2}{\hat{\sigma}_n^2(k, j\Omega)}. \tag{5.19}$$

Then, a *decision directed a-priori* SNR is calculated recursively by using

$$\xi(k, j\Omega) = G^2(k-1, j\Omega)\gamma(k-1, j\Omega)\alpha + (1-\alpha)\max\big(\gamma(k, j\Omega) - 1, 0\big), \tag{5.20}$$

where α is a smoothing factor which controls the trade-off between noise removal and speech quality. The MMSE-LSA gain function is given as

$$G(k, j\Omega) = \frac{\xi(k, j\Omega)}{1 + \xi(k, j\Omega)} \exp\left(\frac{1}{2}\int_{\nu(k,j\Omega)}^{\infty} \frac{\mathrm{e}^{-x}}{x} dx\right), \tag{5.21}$$

with

$$\nu(k, j\Omega) = \gamma(k, j\Omega)\frac{\xi(k, j\Omega)}{1 + \xi(k, j\Omega)}. \tag{5.22}$$

Finally, the enhanced output signal is obtained by filtering the input signal $Y(k, j\Omega)$ with the gain function

$$X(k, j\Omega) = G(k, j\Omega)Y(k, j\Omega). \tag{5.23}$$

Since the gain function in equation 5.21 is real-valued, it represents a non-causal Wiener filter. In real-time applications, such filters can only be used with windowing and overlap-adding the consecutive signal frames. This technique has already been used for realizing the beamforming filters, as described in detail in section 2.8. Therefore, no additional FFT operations are necessary when using a single-channel postfilter.

A major problem of spectral subtraction is a phenomenon known as *musical noise*. Caused by estimation errors in the noisefloor energy, the noise is not attenuated equally over frequency and time. Depending on the statistics of the noise, this effect produces audible tones which are randomly distributed over all frequencies. In order to alleviate this problem, several modifications to the decision-directed a-priori SNR formula in 5.20 exist, like exploiting the correlation properties of speech [47], averaging the a-priori SNR over multiple frames [48], or averaging it over frequency [49].

However, there is a drawback when using Wiener filtering for noise removal; improving the noise reduction performance results in decreasing speech quality, and vice versa [3].

5.2.4 Optimally-Modified Log-Spectral Amplitude estimator

Another widely used improvement of the MMSE-LSA algorithm is its *Optimally Modified* extension (OM-MMSE-LSA) proposed by Cohen in [49]. The MMSE-LSA gain function from equation 5.21 is modified to

$$G_1(k, j\Omega) = G^{p(k,j\Omega)}(k, j\Omega) G_{min}^{1-p(k,j\Omega)}. \tag{5.24}$$

This modification prevents speech components from being eliminated, if they have a low gain value $G(k, j\Omega)$ and a high speech presence probability $p(k, j\Omega)$ at the same time. This happens especially for frequencies with a low SNR. The factor G_{min} allows for a lower limit on signal attenuation. It avoids a complete removal of the output signal during the absence of speech, and thereby increases the perceptive signal quality [3]. Usually it is set to around -15dB.

5.3 Multi-Channel Postfilter

Even though single channel speech enhancement methods prevail for decades, their performance is still poor when the SNR is low, or the interfering noise is non-stationary. Multichannel postfilters surpass these limits by using the spatial information made available by the beamformer. Thereby, the performance of multichannel speech enhancement systems does not rely on noise statistics, but rather on spatial selectivity. In this chapter, three recently published postfilter concepts are introduced: The *Transient Beam to Reference Ratio* [12], the *Direct to Diffuse Ratio* [13] and the *Multichannel Speech Presence Probability* [14].

5.3.1 Transient Beam to Reference Ratio

In [12], the IMCRA noise estimator benefits from incorporating the noise reference provided by the generalized sidelobe canceler from section 3.5. The postfilter algorithm is known as *Transient Beam to Reference Ratio* (TBRR), which indicates whether a transient component is related to the desired speech or the interfering noise. Figure 5.1 shows the block diagram of the proposed multichannel postfilter.

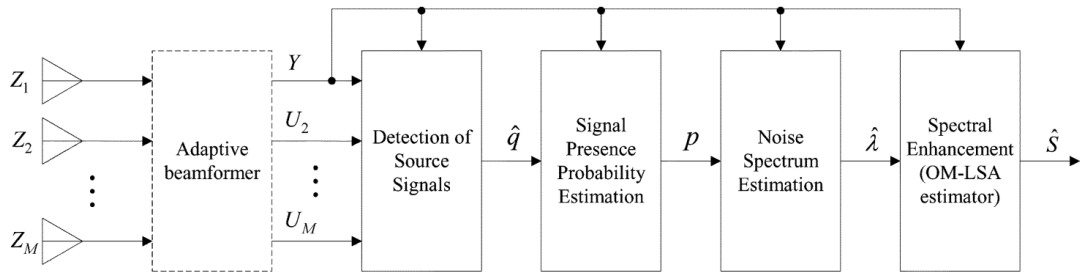

Figure 5.1: Block diagram of the TBRR postfilter, as presented in [12].

The postfilter performs the following tasks: First, two operators for estimating the peaks and the noise floor of a signal are defined. The TBRR is given as their ratio. Then, this ratio is used to derive *a-priori speech absence probability* $q(k, j\Omega)$, which serves as input for the IMCRA algorithm.

The ratio between the transient power at the beamformer output $Y(k, j\Omega)$ and the noise references $\boldsymbol{U}(k, j\Omega)$ is expressed as the *Transient Beam to Reference Ratio* (TBRR). It is calculated

as follows

$$\psi(k, j\Omega) = \frac{\max\{\mathcal{S}Y(k, j\Omega) - \mathcal{M}Y(k, j\Omega), 0\}}{\max\left\{\{\mathcal{S}\big(U_m(k, j\Omega)\big) - \mathcal{M}\big(U_m(k, j\Omega)\big)\}_{m=1}^{M}, \epsilon\right\}}, \tag{5.25}$$

where the operator $\mathcal{S}(\cdot)$ smoothes the PSD, and provides an estimate of the power-envelope of that signal [12]. It is equivalent to the first-order recursive averaging filter from equation 5.9, used in the IMCRA algorithm. The operator $\mathcal{M}(\cdot)$ estimates the noise floor PSD by using either one of the single-channel noisefloor estimation methods presented in chapter 5.2. The symbol ϵ is a small positive constant to avoid a division by zero in the absence of instationarities.

When the beamformer is steered towards the desired signal, and the noise is uncorrelated with the speech, then the TBRR is greater than one for speech transients and smaller than one for noise transients. While a transient in the speech signal is desired, a noise transient should be suppressed as much as possible. To accomplish this, the *a-priori speech absence probability* $q(k, j\Omega)$ has been heuristically defined as

$$q(k, j\Omega) = \begin{cases} 1 & \text{if } \gamma(k, j\Omega) \leq \gamma_{low} \text{ or } \psi(k, j\Omega) \leq \psi_{low} \\ \max\left\{\frac{\gamma_{high} - \gamma(k, j\Omega)}{\gamma_{high} - \gamma_{low}}, \right. & \\ \left. \frac{\psi_{high} - \psi(k, j\Omega)}{\psi_{high} - \psi_{low}}, 0\right\} & \text{otherwise.} \end{cases} \tag{5.26}$$

The *a-posteriori* SNR $\gamma(k, j\Omega)$ has already been defined in equation 5.15. The constants $\gamma_{low}, \gamma_{high}$ and ψ_{low}, ψ_{high} define the endpoints of a linear function to map $\psi(k, j\Omega)$ and $\gamma(k, j\Omega)$ to the range $[0, \cdots, 1]$. They are defined in [36] and [35]. Note that the denominator of equation 5.25 requires M instances of the IMCRA algorithm to be executed on each of the noise references $U_m(k, j\Omega)$. Therefore, the computational load is excessive when many microphones are used. However, instead of using the IMCRA algorithm for $\mathcal{M}(\cdot)$, Minimum Statistics or any other noisefloor estimation algorithm can be used as well.

The noise canceler consists of the last two blocks appearing in figure 5.1: The noisefloor PSD $\hat{\sigma}_n^2(k, j\Omega)$ at the beamformer output is estimated using equations 5.14 - 5.18 from the IMCRA algorithm, which is then used for the OM-MMSE-LSA filter presented in chapter 5.2.4.

5.3.2 Direct to Diffuse Ratio

The *Direct to Diffuse Ratio* (DDR) [13] uses the complex spatial coherence function to distinguish between direct and diffuse sound components. Therefore, this approach perfectly qualifies as a postfilter given the acoustic conditions described in section 2.3.
If we assume a perfectly directional sound field for the desired speaker signal, the signal model defined in equation 2.14 simplifies to

$$Z_m(j\Omega) = S(j\Omega)\mathrm{e}^{-j\Omega f_s \Delta\tau_m} + N_m(j\Omega). \tag{5.27}$$

Hence, the speech-ATFs simply turn into time delays. The same model has been used for the DOA estimation in section 4.3. Based on this model, the direct to diffuse sound ratio is derived in [50]. However, in this paper the DDR is termed *Signal to Reverberation Ratio* or (SRR).

Using the signal model in equation 5.27, the PSD of the m^{th} microphone is given as

$$\Phi_{Z_m Z_m}(j\Omega) = \mathbb{E}\{|Z_m(j\Omega)|^2\} = \Phi_{SS}(j\Omega) + \Phi_{NN}(j\Omega), \tag{5.28}$$

where $\Phi_{SS}(j\Omega)$ and $\Phi_{NN}(j\Omega)$ are the PSDs of the signal and the noise, which are assumed to be equal at all microphones [50]. If we further assume a perfect diffuse noise sound field, the spatial PSD between the m-th and the m'-th microphone is given as

$$\begin{aligned}\Phi_{Z_m Z_{m'}}(j\Omega) &= \mathbb{E}\{Z_m(j\Omega) Z^*_{m'}(j\Omega)\} \\ &= \Phi_{SS}(j\Omega)\mathrm{e}^{-j\Omega f_s \Delta\tau_{mm'}} + \Phi_{NN}(j\Omega)\gamma_{N_m N_{m'}}(j\Omega),\end{aligned} \tag{5.29}$$

where $\gamma_{N_m N_{m'}}(j\Omega)$ may be recognized as the spatial coherence of the ideal diffuse sound field given in equation 2.17. The relative time delay between the m^{th} and the m'^{th} microphone is denoted as $\Delta\tau_{mm'}$. This quantity can be determined using one of the DOA estimation algorithms presented in chapter 4.3. Since the speech is assumed to be directional, and the noise to be diffuse, the ratio of their PSDs gives the DDR

$$\Gamma_{DDR}(j\Omega) = \frac{\Phi_{SS}(j\Omega)}{\Phi_{NN}(j\Omega)}. \tag{5.30}$$

Using the definition of the coherence in equation 2.16, the spatial coherence between the m-th and the m'-th microphone can be directly measured from the microphone signals. It is given as

$$\gamma_{Z_m Z_{m'}}(j\Omega) = \frac{\Phi_{Z_m Z_{m'}}(j\Omega)}{\sqrt{\Phi_{Z_m Z_m}(j\Omega)\Phi_{Z_{m'} Z_{m'}}(j\Omega)}}. \tag{5.31}$$

By inserting equation 5.28 and 5.29 into 5.31, the coherence can be expressed as

$$\begin{aligned}\gamma_{Z_m Z_{m'}}(j\Omega) &= \frac{\Phi_{SS}(j\Omega)\mathrm{e}^{-j\Omega f_s \Delta\tau_{mm'}} + \Phi_{NN}(j\Omega)\gamma_{N_m N_{m'}}(j\Omega)}{\Phi_{SS}(j\Omega) + \Phi_{NN}(j\Omega)} \\ &= \frac{\Gamma_{DDR}(j\Omega)\mathrm{e}^{-j\Omega f_s \Delta\tau_{mm'}} + \gamma_{N_m N_{m'}}(j\Omega)}{\Gamma_{DDR}(j\Omega) + 1}.\end{aligned} \tag{5.32}$$

Solving for the DDR gives

$$\Gamma_{DDR}(j\Omega) = \mathrm{Re}\left\{\frac{\gamma_{N_m N_{m'}}(j\Omega) - \gamma_{Z_m Z_{m'}}(j\Omega)}{\gamma_{Z_m Z_{m'}}(j\Omega) - \mathrm{e}^{-j\Omega f_s \Delta\tau_{mm'}}}\right\}, \tag{5.33}$$

where only the real part contains the desired information [50]. In [13], the DDR is converted into a speech absence probability using the following mapping function

$$q(k, j\Omega) = \frac{10^{\gamma_0\gamma_s/10}}{10^{\gamma_0\gamma_s/10} + \Gamma_{DDR}^{\gamma_s}(j\Omega)}, \tag{5.34}$$

where γ_0 controls the offset along the Γ_{DDR} axis, and γ_s defines the steepness of the transition between speech absence and presence. When using more than 2 microphones, the average of the

DDRs between each pair can be used before calculating the speech absence probability. This makes the algorithm less prone to estimation errors. Similar to the TBRR, the noisefloor PSD at the beamformer output $\hat{\sigma}_n^2(k, j\Omega)$ is estimated using equations 5.14 - 5.18 from the IMCRA algorithm, which is then used for the OM-MMSE-LSA filter presented in chapter 5.2.4. Compared to TBRR, the computational cost is fairly small. A similar approach to the DDR can be found in [21].

While the assumption of a diffuse noise sound field holds for most real application scenarios, the sound field from the speech signal might not be perfectly directional due to reverberations. Therefore the DDR might be wrong for frequencies containing strong speaker echoes. As a result, the estimated noisefloor PSD $\hat{\sigma}_n^2(k, j\Omega)$ might be underestimated for these frequencies. A straightforward approach to fix this issue is to run an additional single channel noise PSD estimation algorithm like IMCRA or Minimum Statistics on the beamformer output. Then, the greater noise estimate can simply be selected by

$$\hat{\sigma}_n^2(k, j\Omega) = \max\left(\hat{\sigma}_{n,DDR}^2(k, j\Omega), \hat{\sigma}_{n,IMCRA}^2(k, j\Omega)\right). \tag{5.35}$$

However, this fix has to be used with great care. If the noise estimator removes too much information from the desired speech signal, the primary benefit from the beamformer might be lost again. In the experiments, the bias compensation factor B_min of both noise estimators has been lowered to avoid this from happening.

5.3.3 Multichannel Speech Presence Probability

The last postfilter concept is based on a *Multichannel Speech Presence Probability* (MCSPP), presented in [14]. The motivation is to generalize the already optimal single channel noise reduction Wiener filter from chapter 5.2 to the multichannel case. Basically, the algorithm is very similar to IMCRA, except for being extended to operate with multiple microphones.

Considering the original signal in equation 2.14, the spatial PSDs at the microphones is given by:

$$\boldsymbol{\Phi}_{ZZ}(j\Omega) = \boldsymbol{\Phi}_{SS}(j\Omega) + \boldsymbol{\Phi}_{NN}(j\Omega), \tag{5.36}$$

where $\boldsymbol{\Phi}_{SS}$ and $\boldsymbol{\Phi}_{NN}$ are the unknown speech and noise PSDs at the microphones, respectively. In [51], the single-channel *a posteriori* SNR from equation 5.15 is defined in a multi-channel fashion as

$$\tilde{\gamma}(k, j\Omega) = \text{trace}\left[\boldsymbol{\Phi}_{NN}^{-1}(k, j\Omega)\boldsymbol{\Phi}_{ZZ}(j\Omega)\right]. \tag{5.37}$$

The multichannel *a priori* SNR is calculated analogously with

$$\xi(k, j\Omega) = \text{trace}\left[\boldsymbol{\Phi}_{NN}^{-1}(k, j\Omega)\boldsymbol{\Phi}_{SS}(j\Omega)\right] = \tilde{\gamma}(k, j\Omega) - M. \tag{5.38}$$

By using the PSDs a long-term average is obtained. An instantaneous *a posteriori* SNR is obtained by using

$$\gamma(k, j\Omega) = \boldsymbol{Z}^H(k, j\Omega)\boldsymbol{\Phi}_{NN}^{-1}(k, j\Omega)\boldsymbol{Z}(k, j\Omega). \tag{5.39}$$

Moreover, a third quantity $\beta(k, j\Omega) = \gamma(k, j\Omega)\xi(k, j\Omega)$ is defined by using the matrix inversion lemma[51]. It is given as

$$\beta(k, j\Omega) = \boldsymbol{Z}^H(k, j\Omega)\boldsymbol{\Phi}_{NN}^{-1}(k, j\Omega)\boldsymbol{\Phi}_{SS}(j\Omega)\boldsymbol{\Phi}_{NN}^{-1}(k, j\Omega)\boldsymbol{Z}(k, j\Omega). \tag{5.40}$$

In chapter 5.2.2, the IMCRA algorithm uses a single-channel *speech presence probability* $p(k, j\Omega)$. A straightforward generalization to the multi-channel case can be found by inserting equations 5.38 and 5.40 into 5.14, i.e.

$$p(k, j\Omega) = \left[1 + \frac{q(k, j\Omega)}{1 - q(k, j\Omega)} \left[1 + \xi(k, j\Omega)\right] \exp\left[-\frac{\beta(k, j\Omega)}{1 + \xi(k, j\Omega)}\right]\right]^{-1}. \tag{5.41}$$

Similar to the IMCRA algorithm, the *speech absence probability* $q(k, j\Omega)$ is defined as

$$q(k, j\Omega) = \begin{cases} 1, & \text{if } \tilde{\gamma}(k, j\Omega) < M \text{ and } \gamma(k, j\Omega) < \gamma_0 \\ \frac{\tilde{\gamma}_0 - \tilde{\gamma}(k, j\Omega)}{\tilde{\gamma}_0 - M}, & \text{if } M \leq \tilde{\gamma}(k, j\Omega) < \tilde{\gamma}_0 \text{ and } \gamma(k, j\Omega) < \gamma_0 \\ 0, & \text{otherwise} \end{cases} \tag{5.42}$$

Also, smoothing in frequency is performed to increase the robustness against outliers. So far, the MCSPP solely depends on the noise PSD at the microphones $\boldsymbol{\Phi}_{NN}(j\Omega)$. As this matrix is generally unknown, a two-stage iterative estimation by using the speech presence probability is suggested by [14].

If the MCSPP is used as a postfilter, this iterative estimation of $\boldsymbol{\Phi}_{NN}(j\Omega)$ is not necessary. A good noise estimate for this noise PSD can be obtained by using the output $Y(k, j\omega)$ of the GSC beamformer from equation 3.33: The beamformer's output is an estimate of the speech component at the reference microphone. By using the RTFs, this estimate can be back-projected to each of the microphones with $\boldsymbol{A}(k, j\Omega)Y(k, j\omega)$. According to the signal model from equation 2.14, the noise component at each microphone is then found by subtraction

$$\hat{\boldsymbol{N}}(k, j\Omega) = \boldsymbol{Z}(k, j\Omega) - \boldsymbol{A}(k, j\Omega)Y(k, j\omega), \tag{5.43}$$

An estimate of the noise PSD can be obtained by recursive averaging

$$\boldsymbol{\Phi}_{\hat{N}\hat{N}}(k, j\Omega) = \boldsymbol{\Phi}_{\hat{N}\hat{N}}(k-1, j\Omega)\alpha + (1-\alpha)\hat{\boldsymbol{N}}(k, j\Omega)\hat{\boldsymbol{N}}^H(k, j\Omega). \tag{5.44}$$

As for the TBRR and the DDR, the noisefloor PSD at the beamformer output $\hat{\sigma}_n^2(k, j\Omega)$ is estimated using equations 5.14 - 5.18 from the IMCRA algorithm, which is then used for the OM-MMSE-LSA filter presented in chapter 5.2.4. The computational cost of MCSPP is the smallest of these three postfilters, as no additional noisefloor estimator such as IMCRA is needed. Even though, using equations 5.37 and 5.40 requires matrix multiplications and inversions, which might be more complex than the entire GSC beamformer if many microphones M are used. In the Matlab experiments, the matrix inversion of equation 5.38 has been heuristically replaced with the inverse of the trace of $\boldsymbol{\Phi}_{\hat{N}\hat{N}}(k, j\Omega)$. The performance of all three postfilters is compared in section 6.

5.4 Psychoacoustics

The *multichannel speech enhancement* system (MCSE) consists of the GSC beamformer and a postfilter. Usually, the postfilter involves either matrix inversions or several instances of a noisefloor estimation algorithm, therefore it has a higher computational complexity than the GSC. In the context of a real-time application scnario, the aim is to reduce the computational cost of the postfilter.

5.4.1 Auditory Masking

A widely used method to reduce the computational complexity of speech enhancement algorithms is to exploit the properties of the human auditory system. A remarkable property is knwown as auditory masking. Masking is described as the loudness of a test sound necessary to be barely audible in the presence of a masking sound [52]. This masking sound can either be a pure sinusodial tone, or a narrow-band noise signal. Given a masking signal, the so-called *masking curves* are obtained by sweeping the test-tone over the entire frequency range, measuring its necessary loudness to be audible to human listeners. Figure 5.2 shows these masking patterns for a sinusodial masker at 1kHz at five different volumes. It can be seen that the masking threshold is both frequency and volume dependent. From this observation it can be concluded, that a residual noise signal from the postfilter is inaudible if a louder speech component is present at nearby frequency. The figure uses a frequency scale which has been converted to the *equivalent rectangular bandwidth* (ERB).

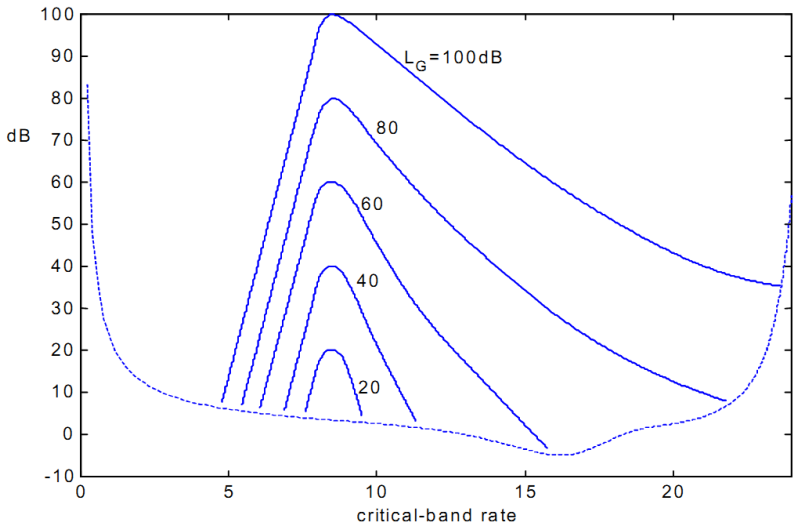

Figure 5.2: Masking curves for a pure test tone of 1kHz, taken from [53]. The dashed line denotes the hearing threshold.

Masking also occurs in time domain. For example, if a loud and a soft signal are played shortly one after another, the soft one is masked because the ear's sensitivity is still adjusted to the loud one. This effect even works the other way around, where the loud signal appears after the soft one. This is possible because the human brain needs a certain amount of time to perceive and process the acoustic information. In [52], masking effects are described in detail.

Another observation can be made when measuring the perceived loudness as function of the number of simultaneously played test-tones. The test-tones are equally spaced in frequency, usually very close to each other. The tones are also played at the same volume, so that the

perceived loudness increases with the number of tones being played. Using human listeners, it can be observed that the overall loudness does not increase, once a certain number of test-tones is exceeded. The frequency span covered by these test-tones is known as the *Critical Bandwidth* [52].

As with the masking curves, the critical bandwidth is frequency dependent and rises semi-logarithmically with frequency. An approximation for the critical bandwidth as function of frequency has been given in [54]. It is given as *equivalent rectangular bandwidth* (ERB), which can be thought of bandwidth of a bandpass filter used by the human auditory system to group the perceived loudness. The formula is given as

$$f_{ERB} = 11.17268 \log\left(\frac{1 + 46.06538 f_{Hz}}{14678.49 + f_{Hz}}\right). \tag{5.45}$$

In figure 5.3 the ERB scale is illustrated over a linear frequency range. Due to the logarithm in equation 5.45, the ERB increases slowly at higher frequencies, allowing to map the entire frequency range up to 16kHz into only 36 bands.

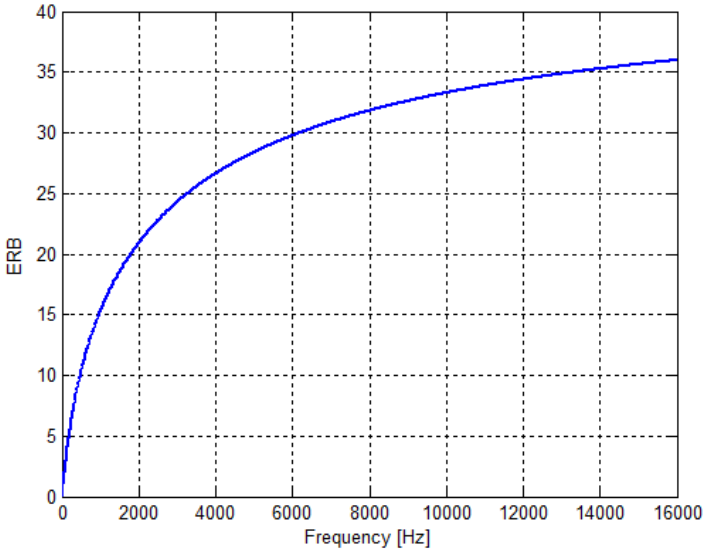

Figure 5.3: Equivalent rectangular bandwidth, as per equation 5.45.

When using a filterbank, where the bandwidths of the single filters are adjusted according to equation 5.45, a linear frequency scale can be converted to much less ERB bands. If for example a window length of 32ms at $f_s = 16$kHz is used, the FFT would produce 257 non-redundant linearly spaced frequency bands, which can be converted into 32 ERB bands. Thereby the numerical complexity for algorithms operating in the frequency domain can be reduced by a factor of 8.

5.4.2 Simplified Gammatone Filterbank

In order to exploit the properties of the human auditory system, a filterbank is used where the filters model the frequency response of the basilar membrane to a certain extent. They have to account for both the critical bandwidth and the masking phenomenon [52]. A widely used type of filterbank for that purpose is the gammatone filterbank. It allows to control the steepness

and bandwidth by using the following formula [55]

$$g(t) = a \cdot t^{n-1} \mathrm{e}^{-2\pi b \cdot t} \cos(2\pi f_c \cdot t + \phi), \tag{5.46}$$

where the parameters a is the filter amplitude, n the filter order which controls the steepness, f_c the center frequency, b the filter bandwidth and ϕ the phase of the carrier. A design goal in using a gammatone filterbank is the reconstruction of the original signal. This is simply done by adding the output of the filters. To avoid unwanted gaps or bumps in the overall frequency response, the parameters of the filters must be chosen such that the sum of all filters exhibits a smooth frequency response. An example is shown in figure 5.4.

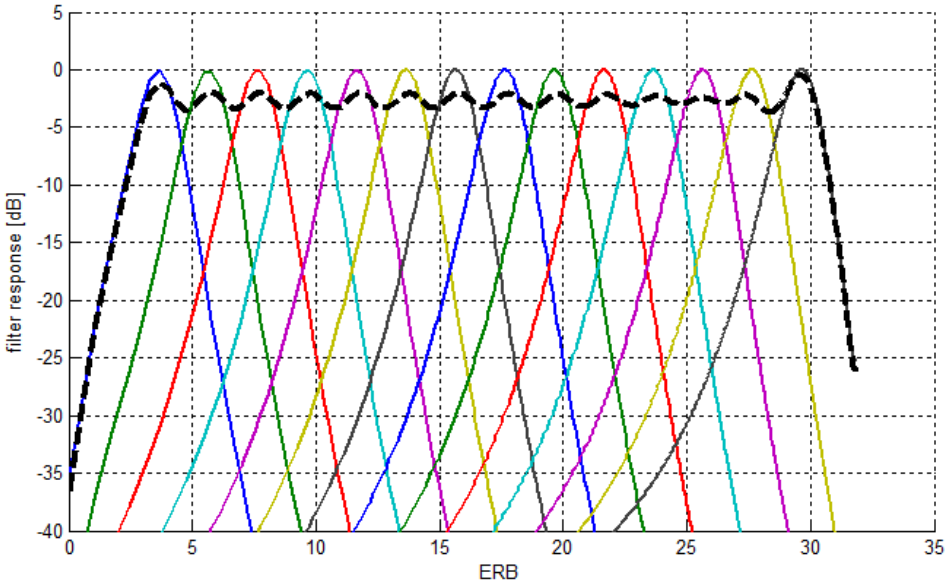

Figure 5.4: Frequency response of a gammatone filterbank with 14 filters, using $a = 1$, $n = 2$, $b = 1.5ERB$ and a spacing of 2 ERB between the filters. The dashed line in black shows the frequency response of the sum of all filters.

Approaches such as [56] suggest to put the filterbank in front of the entire GSC beamformer and postfilter, in order to gain the largest benefit in computational cost. However, reconstructing the signal phase by inverse gammatone filters is not trivial. The phase has to be estimated after applying filter operations such as the GSC filters. To avoid this problem, the filterbank is only used in the postfilter. All three presented postfilter algorithms only use the magnitude spectrum of the beamformer output $Y(j\Omega)$ or the noise references $\boldsymbol{U}(j\Omega)$. It is therefore sufficient to apply the magnitude of the gammatone filterbank to the magnitude of the postfilter's input signals and simply dropping the phase. As a result, the postfilter does not modify the phase of the enhanced speech signal. Hence, both the decomposition and synthesis using the gammatone filterbank can be done efficiently in magnitude domain.

Furthermore, we observed that the steepness of the gammatone filters can be increased without noticeable loss in perceived quality. This allows for a drastic simplification of the entire filterbank: Instead of applying the filter transfer function shown in figure 5.4, it is sufficient to simply add the magnitudes of the FFT bins falling within the bandwidth b. Decomposition of

a signal $Y(j\Omega)$ into the k-th ERB band then becomes

$$Y_{ERB}(k) = \sum_{\Omega=\Omega_0}^{\Omega_1} Y_{Hz}(j\Omega), \tag{5.47}$$

where Ω_0 and Ω_1 mark the boundaries of the ERB band. Synthesis is performed by dividing the ERB bin to all FFT bins that fall within that band, i.e.

$$Y_{Hz}(j\Omega) = \frac{1}{\Omega_1 - \Omega_0} Y_{ERB}(k) \tag{5.48}$$

This mapping is shown in figure 5.5. It transforms 257 FFT bins to the ERB scale using 24 bands. A filled rectangle denotes a set of frequency bins combined into the k^{th} ERB bin used in equation 5.47.

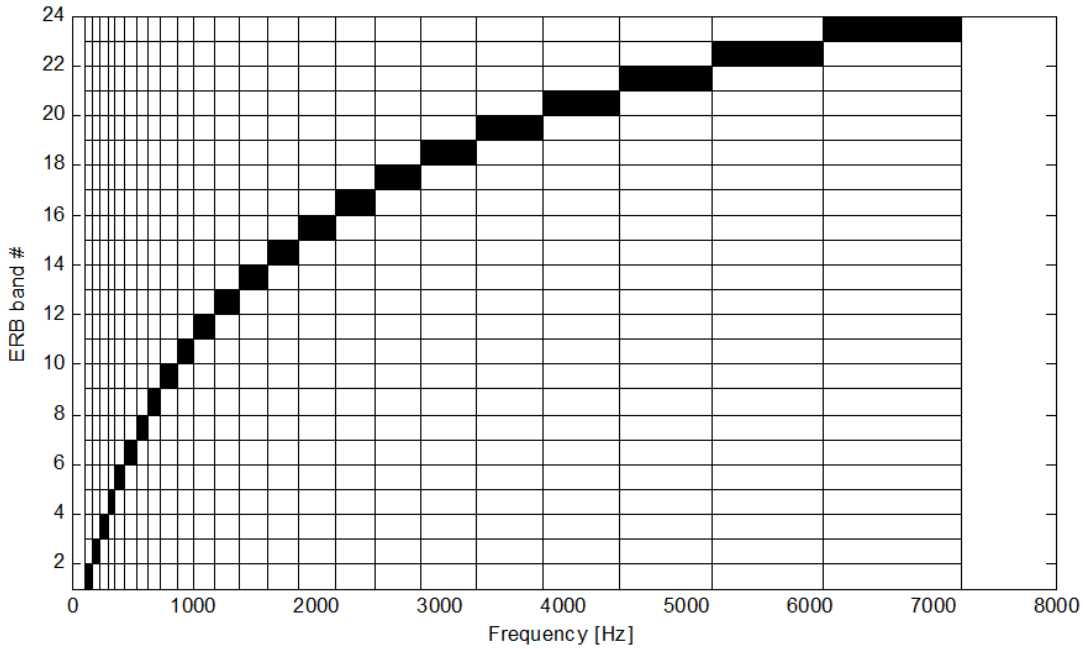

Figure 5.5: Mapping of the simplified gammatone filterbank for 24 bands, a sampling rate of $f_s = 16kHz$ and 32ms FFT length.

While this simplified filterbank saves about 90% of the computational cost of the entire post-filter, its effect on the perceived speech quality is negligible. This is empirically shown in the experiments in section 6.

6 Matlab Experiments

In the previous chapters, the main building blocks of a multichannel speech enhancement (MCSE) system based on a beamformer and a multichannel postfilter have been introduced. Many different algorithms were presented to solve the problems in each domain. In this chapter, the optimal combination of the algorithms is determined using the given microphone array setup, based on different speech quality measures.

6.1 Quality Assessment

To evaluate the overall speech quality of the enhanced signal after the beamformer and the postfilter, a technical measure such as the SNR is insufficient. More sophisticated measures are advantageous. While using the *Mean Optinion Score* (MOS) is adequate, its evaluation requires a full-fledged listening test. This requires a lot of time and dozens of test persons, being trained for the listening tests. Therefore, automated measures that mimic the human impression of speech quality and noise annoyance are used instead.

6.1.1 Signal Blocking Factor

With the means of estimating both the RTF and the DOA, it is advantageous to have some performance measures in order to evaluate and compare the different algorithms. The most intuitive measure would be the output SNR of the GSC beamformer. If the estimated RTF is close to the true RTF, the SNR is maximized. However, this measure is impractical as the SNR improvement of the beamformer alone can be very small, depending on the spatial correlation of noise sound field.

A more sophisticated measure evaluates the accuracy of the DOA and RTF estimates by examining the noise references $\boldsymbol{U}(j\Omega)$ of the blocking matrix [57]. It is known as *Signal Blocking Factor* (SBF), and it evaluates the target leakage in the blocking matrix. Using target leakage as a performance measure is valid, since the accuracy of the RTFs directly influences the blocking

matrix. The SBF is given as

$$\text{SBF} = 10\log_{10}\frac{\mathbb{E}\left\{\sum_{m=1}^{M}|S_m(j\Omega)|^2\right\}}{\mathbb{E}\left\{\sum_{m=1}^{M}|U_m(j\Omega)|^2\right\}}, \tag{6.1}$$

where $S_m(j\Omega)$ is the speech component at the m-th microphone, and $U_m(j\Omega)$ is the m-th noise reference at the output of the blocking matrix. Clearly, measuring the signal blocking factor is only useful in the absence of additive noise at the microphones. In this case, $U_m(j\Omega)$ only carries the portion of the speech signal, which is leaking through the blocking matrix. In the experiments section, this measure is used to determine the optimal DOA and RTF estimation algorithm.

6.1.2 Perceptual Evaluation of Speech Quality

Probably the most widely used measure for speech quality is the *Perceptual Evaluation of Speech Quality* (PESQ). It is standardized as ITU-T recommendation P.862 released in 2001. The algorithm takes a degraded signal and compares it to the noise-free reference of that signal. Both signals must be time-aligned to ensure proper operation. The PESQ algorithm allows for two filter options. Narrow-band and Wide-band, which can be chosen depending on the application. After filtering the signals are split into frames which are time-aligned to account for jitter that may occur in VoIP networks. Then, an auditory transform is applied to the signal to convert it from frequency domain to loudness domain. To account for the distortions that may be perceived by a human listener, the frequency and loudness representation of the signals are subtracted and accumulated over time. These distortions are weighed depending on whether it was caused by additive noise or missing speech components. Finally, the result is converted into a MOS score, where a score of 1 denotes bad quality and 5 excellent quality.

While PESQ was never explicitly intended to evaluate speech enhancement algorithms, it is used in most publications in that field. The PESQ score for speech enhancement algorithms is generally too low, especially when the algorithm only modifies the magnitude spectrum of the speech signal. Therefore the PESQ score does not reflect the perceived impression of speech quality.

6.1.3 PEASS

A more elaborated means to evaluate the quality of a speech enhancement algorithm has been provided by the *Perceptual Evaluation methods for Audio Source Separation* Toolkit in [16] and [17]. It was developed for quality assessment of audio source separation algorithms. The multichannel speech enhancement algorithm depicted by the beamformer and the postfilter can be seen as a source separation task, i.e. the enhanced speech signal at the output is an estimate of the speech source at the reference microphone.

PEASS takes the enhanced speech signal $X(j\Omega)$, and the clean speech and noise components $\boldsymbol{A}(j\Omega)S(j\Omega)$ and $\boldsymbol{N}(j\Omega)$ as inputs. After applying a filterbank, the signals are decomposed into a target component $\hat{s}_j$ and three distortion components corresponding to target distortion e_j^{target}, additive interferences e_j^{interf}, and processing artifacts e_j^{artif} such as musical noise.

Then, a perceptual salience measure (PEMO-Q) is used to predict a feature vector consisting of four terms: a global feature q_j^{global} that estimates the overall quality of the speech estimate,

and the features q_j^{target}, q_j^{interf} and q_j^{artif} that estimate the quality with respect to the distortion components. Each feature is in the range from 0 to 1. By using a neural network trained on a large set of test sounds, these feature are then mapped into four scores, the *Overall Perceptual Score* (OPS), the *Target Perceptual Score* (TPS), the *Interference Perceptual Score* (IPS) and the *Artifact Perceptual Score* (APS). Each score ranges from 0 to 100 for improved human readability, where a larger number indicates a higher perceived quality.

PEASS has been provided as a Toolbox for Matlab ®. Because of its complexity, it is rather slow. Therefore some of its functionality has been written as MEX-functions for the sake of speed. Both PEASS and PESQ are used to evaluate the perceptual speech quality of the beamformer and the postfilter in the experiments.

6.2 Experimental Setup

The multichannel speech enhancement system composed of the GSC and the postfilter are evaluated under realistic conditions, therefore actual multichannel speech and noise recordings are used. All three performance measures (PEASS, PESQ, SBF) require the ground truth of all components in order to produce results. Therefore, both the speech and the noise signals have been recorded separately. Before being presented to the algorithms, the recordings are added to produce the noisy speech inputs $\boldsymbol{Z}(j\Omega)$.

6.2.1 Acquiring Speech Data

In order to test the multichannel speech enhancement system against a significant amount of speech data, both English and German speech databases have been used. For American English, the TIMIT speech corpus [58] is often used for the evaluation of speech enhancement algorithms. For German, both the Kiel Corpus of Read Speech (KCORS) [59] and the Kiel Corpus of Spontaneous Speech (KCOSS) [60] are employed. All three databases provide 16kHz speech recordings from both male and female speakers. Apart from short read sentences in TIMIT and KCORS, the KCOSS corpus also provides longer sentences containing fluent speech from conversations. Therefore, the latter also provides a richer dynamic in volume and duration of the different utterances.
Since all the speech data is recorded in mono, they have to be converted to multichannel signals in order to be useful for the beamformer. In section 2.7, the measurement of the ATFs for typical office rooms is described in detail. This procedure is repeated in three different office rooms of about 2.5x4m and 6x8m in size, with the measurement loudspeaker located approximately 0.5m away from the microphone array. The angle of the loudspeaker is set to four different positions: 0°, 20°45°and 70°relative to the broadside of the array. As described in chapter 2.7, the linear array consisting of 4 microphones with an inter-microphone distance of 5cm is used for recording of the ATFs. From these 3 rooms and 4 positions, 12 different ATFs are available. The multichannel speech data is generated by convolving the speech files with the ATFs using Matlab ®[61]. All three speech databases provide approximately 9hrs of spoken speech that can be used with the ATFs. For comparing the results of the RTF estimation methods presented in chapter 4, the RTFs are constructed from the measured ATFs by using equation 2.15.

6.2.2 Acquiring Noise Data

To obtain useful noise data the approach of convolving mono files with ATFs is not very practical because the noise sound field is diffuse and hence the ATFs are very long and hard to measure. A better way is to use stereo noise samples and replay them in the aforementioned office rooms.

The microphone array is used to record the noise with four channels. Hence, the noise recordings are more or less diffuse, depending on the location of the loudspeaker and the microphone array in the room. As noise data 56 recordings have been used from various sources. The complete list is provided in section 9.

6.3 Matlab Implementation

In the previus chapters, several algorithms for RTF and DOA estimation, BM construction and multichannel postfiltering have been presented. To compare all approaches, they have been implemented in Matlab as *MCSE framework*. To simulate the same real-time conditions as in chapter 7, the code has been designed to work with consecutive frames of data. Doing so, the Matlab implementation shows the same behavior as the C++ implementation, which allows for easy validation and verification of the C++ code against the Matlab implementation.

In figure 6.1 an overview of the MCSE system is given. Orange blocks denote the generation of the simulation data, which is constructed from the speech databases, the ATF measurements and the noise recordings as described in chapter 6.2. Using the signal model from equation 2.14, the simulation data can be generated by convolving the speech data with the ATFs, followed by adding the noise data. Since the array used for the recordings consists of four microphones, a maximum of four channels can be used as test data. The performance of the multichannel speech enhancement system should be evaluated for different noise levels, therefore the volume of the noise is adjusted prior to the addition. The *Signal to Interference Ratio* (SIR) controls the power ratio between the Speech and Noise samples. It can be adjusted in decibel as a simulation parameter. The blue block denotes the source localization task, where four RTF estimation algorithms of chapter 4 have been implemented. The ICA algorithm has not been included in the simulation framework for a number of reasons: First, it did not converge to a useful solution in approximately 25 % of all cases. Also, solving the permutation problem based on the DOA proved to be unreliable because the coherence is also high for the noise component at low frequencies. Further, the ICA only works in batch mode, meaning that a block of several seconds of audio is processed at a time. Thus, changes in the RTF are only recognized after a delay of several seconds. The green blocks depict the GSC beamformer. It consists of a fixed beamformer, a blocking matrix and an adaptive interference canceler. The fixed beamformer has been implemented as shown in chapter 3.5. For the BM, four structures have been mentioned in section 3.6. Although, the *Adaptive Blocking Matrix* and the *Generalied Eigenvector Blocking Matrix* structures are not used due to their lack of robustness for low SNRs. The other two structures (*Eigenspace* and *Sparse*) are more robust because they rely only on the DOA estimation. Furthermore, the numerical complexity is fairly negligible. For the AIC, two different structures are specified in chapter 3.5. Both the Wiener solution and the NLMS algorithm have been implemented. The purple block shows the multichannel postfilter. All three different multichannel postfilters (MCPF) from chapter 5 have been implemented. The TBRR and DDR based postfilter require a noisefloor estimator (NE). Therefore, the IMCRA and Minimum Statistics algorithms from chapter 5.2 have been implemented. The third postfilter, the MCSPP does not need a noisefloor estimator at all. The yellow blocks in figure 6.1 denote the evaluation of the results, based on the PEASS and PESQ speech quality measures, as well as on the SBF, introduced in chapter 6.1.

The execution of a specific combination of all algorithms depends on a set of simulation parameters, stored in a Matlab-struct. An example of this setup struct is given by the following lines of code:

```
       doa_type: {SCOT|PHAT|MUSIC}
       rtf_type: {DOA|real|CohenWLS}
        bm_type: {sparse|eigenspace}
       aic_type: {Wiener|NLMS}
        ne_type: {Minimum Statistics|IMCRA}
      mcpf_type: {TBRR|DDR|MCSPP}
speech_database: {KCORS|KCOSS|TIMIT}
     speech_ATF: {B208_0deg.wav|B208_20deg.wav|B208_45deg.wav|B208_70deg.wav}
      noisefile: Noise Samples\Roadnoise_4ch.wav
         Nbands: {0,48,36,24,12}
          Nmics: {1,2,3,4}
 use_filterbank: {yes|no}
     target_SIR: {-20,-15,-10,-5,0,5,10,15,20}
```

Prior to an experiment, a combination of the parameters in the brackets has to be selected.

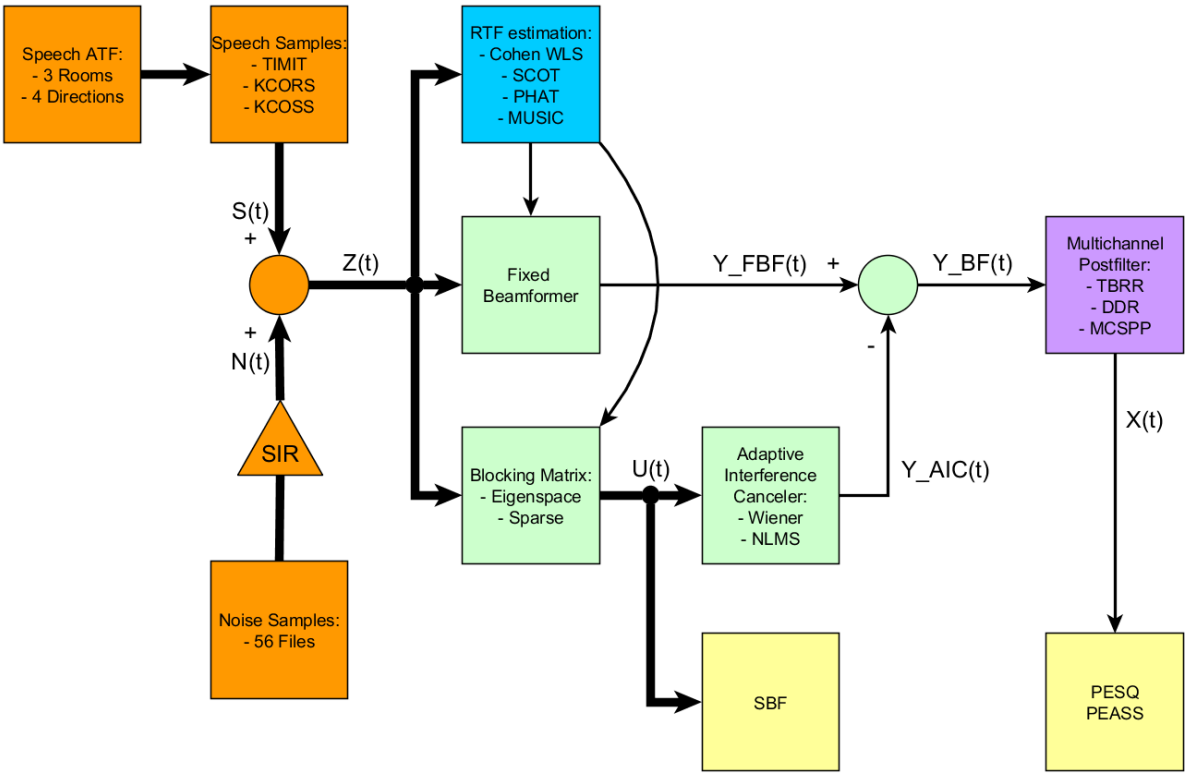

Figure 6.1: Matlab implementation of the entire MCSE system.

6.4 Simulation Testbench

To be able to automatically execute all of the simulation scenarios described in the previous section, a simulation testbench has been written using Matlab. By using object oriented programming with Matlab-structs, the MCSE implementation in figure 6.1 can also be re-used for this testbench. It simulates all meaningful combinations of parameters shown in the setup struct. A specific combination of simulation parameters is termed a *simulation scenario*. Each simulation scenario is tested against a fixed SIR setting ranging from -20dB to +20dB in 5dB

steps, resulting in 9 different SIR levels. For each SIR, 1 hour of audio data is used. The audio material consists of a predefined combination of speech data, ATFs and noise data. Hence, each combination of algorithms is tested against the same set of input data, which allows for an easy comparison. This simulation data is divided into 60 small chunks of 1 minute duration, otherwise the PEASS algorithm would run out of memory. Each of these small simulations is called an *episode*. 60 episodes are are combined as *batch*. In figure 6.2 an overview of the simulation testbench is given in form of an UML activity diagram.

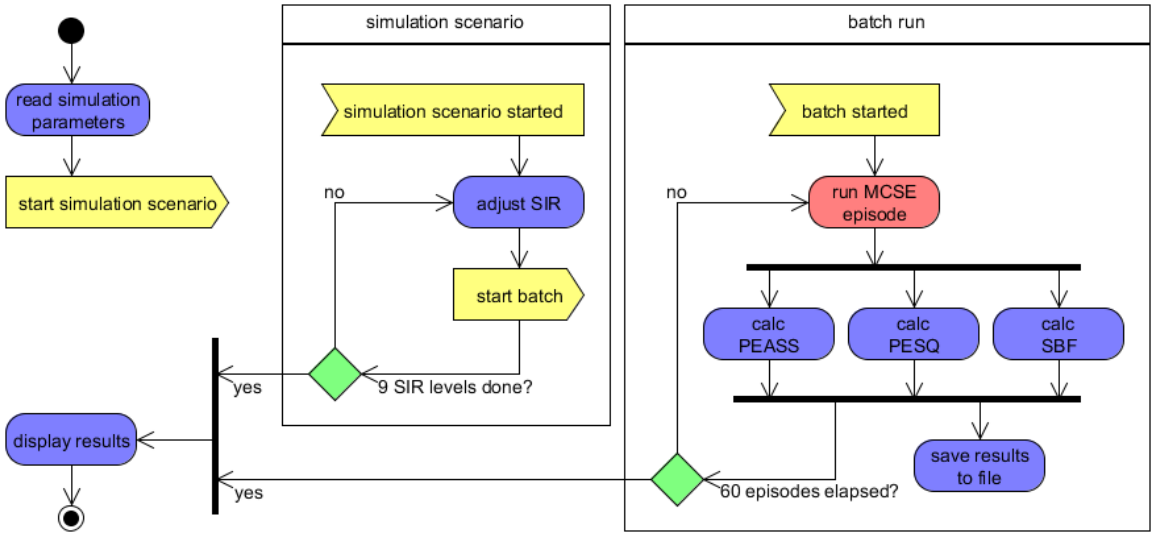

Figure 6.2: Activity diagram of the MCSE testbench.

6.5 Simulation Scenarios

To reduce the vast combinatorial space of all simulation parameters, a set of 24 meaningful simulation scenarios has been conceived. A single scenario simulates 9 hours of audio, which requires approximately 32 hours to complete on a Core i7 CPU @ 3.4Ghz and 8Gbyte of main memory, depending on the algorithms being tested. This accounts for a total simulation time of 768 hours or 24 days. To speed up this process, a second identical machine was used in parallel. For a better overview, these 24 simulation scenarios have been grouped into the following five blocks:

6.5.1 Scenario 1: RTF Estimation

The blue block in figure 6.1 lists one RTF algorithm and three DOA algorithms. These four algorithms are compared against the true RTF, which is constructed from the ATFs used for generating the multichannel speech signals in chapter 6.2.1. These five simulation scenarios are evaluated against each other using the SBF as performance measure:

RTF estimation method
RTF: Cohen WLS
DOA: SCOT
DOA: PHAT
DOA: MUSIC
true RTF

6.5.2 Scenario 2: BM and AIC Structure

The beamformer in figure 6.1 shows four algorithms, two for constructing a BM and two for the AIC. Based on the optimal RTF estimation method from the first scenario, each of the two BM structures is evaluated against the two AIC structures. As performance measure the PEASS and PESQ algorithms are used:

BM and AIC structure
BM: Sparse + AIC: Wiener
BM: Sparse + AIC: NLMS
BM: Eigenspace + AIC: Wiener
BM: Eigenspace + AIC: NLMS

6.5.3 Scenario 3: Postfilter Algorithm

Three multichannel postfilters (MCPF) in figure 6.1 are compared against the two available noisefloor estimation methods (NE): IMCRA and Minimum Statistics. The MCSPP postfilter algorithm does not need a noisefloor estimation, thus five combinations remain. Evaluation is based on the outcome of the second test scenario and on the PEASS and PESQ speech quality measures:

Postfilter algorithm
NE: Minimum Statistics, PF:TBRR
NE: IMCRA, PF:TBRR
NE: Minimum Statistics, PF:DDR
NE: IMCRA, PF:DDR
MCSPP

6.5.4 Scenario 4: Effect of the Filterbank

In chapter 5.4, the concept of using a filterbank in order to reduce the numerical complexity of the postfilter is presented. In this simulation scenario, the simplified gammatone filterbank is applied to the best postfilter determined in the third scenario. The influence of the filterbank in terms of PEASS and PESQ is evaluated as a function of the number of bands:

Effecto of the Filterbank
none used
12 bands
24 bands
36 bands
48 bands

6.5.5 Scenario 5: Number of Microphones

The last test is done to empirically demonstrate that multichannel speech enhancement has a significant advantage over single channel speech enhancement in terms of speech quality. Again,

PEASS and PESQ are used as performance measures. In this last test scenario, the number of microphones is varied from 1 to 4. For using one microphone, the *Minimum Statistics* noisefloor estimation method, combined with the OM-MMSE-LSA algorithm from chapter 5.2 is used as a standard single channel speech enhancement algorithm. For comparison, the theoretical performance maximum of using the true RTFs from all four microphones is also evaluated:

Number of Microphones
1
2
3
4
true RTF

6.6 Simulation Results

The first simulation scenario, the RTF estimation, leads to the following results. In figure 6.3 the SBF is plotted against the 9 SIR levels. It can be seen that the estimation of the full RTFs using Cohen's WLS method from section 4.2.1 has a rather low performance in terms of the SBF. The same result can be observed for the MUSIC algorithm. Better results are obtained by the SCOT or PHAT DOA estimation methods, which are almost equally well suited for this task. Another benefit is that these two methods have a much smaller computational cost than the other two. The brown plot is obtained when the RTF estimation module is bypassed and the FBF and the BM are constructed using the real RTFs, as explained for the first scenario and illustrated in figure 6.1. It gives the upper performance limit for the algorithms. From this plot, it can be seen that the SCOT method is slightly better than PHAT for low SIR levels. Hence, the SCOT method is chosen as DOA estimation algorithm in the final MCSE system.

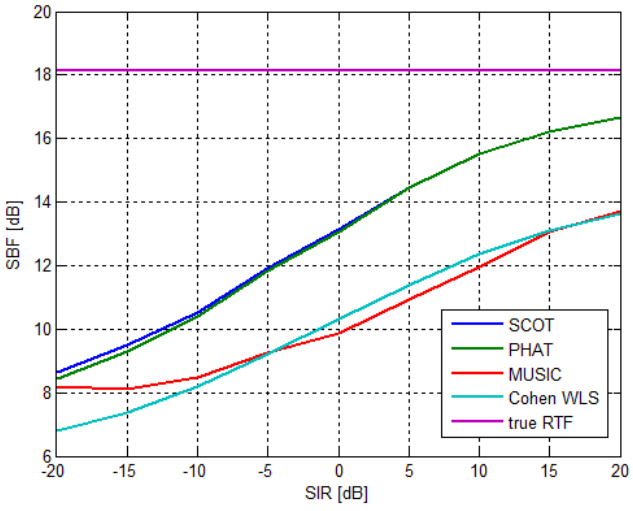

Figure 6.3: SBF results for the first simulation scenario.

Using the SCOT method from the first scenario, the two blocking matrices have been evaluated against the two AIC algorithms in the second scenario. The resulting PEASS plots are shown in figure 6.4. Panel (a) shows the overall perceptual score (OPS), as explained in section 6.1.3. It can be observed that the combination of an eigenspace BM and the NLMS algorithm for

the AIC filters gives the highest score. Panel (b) shows the target-related perceptual score (TPS), which is also the highest for this combination. Panel (c) shows the interference-related perceptual score (IPS), where all four methods are more or less equal. Panel (d) shows the artifact-related perceptual score (APS), where again the combination of the sparse BM and the NLMS algorithm is performing best for large SNR conditions. Hence, this setting is used for the GSC beamformer in the final MCSE system.

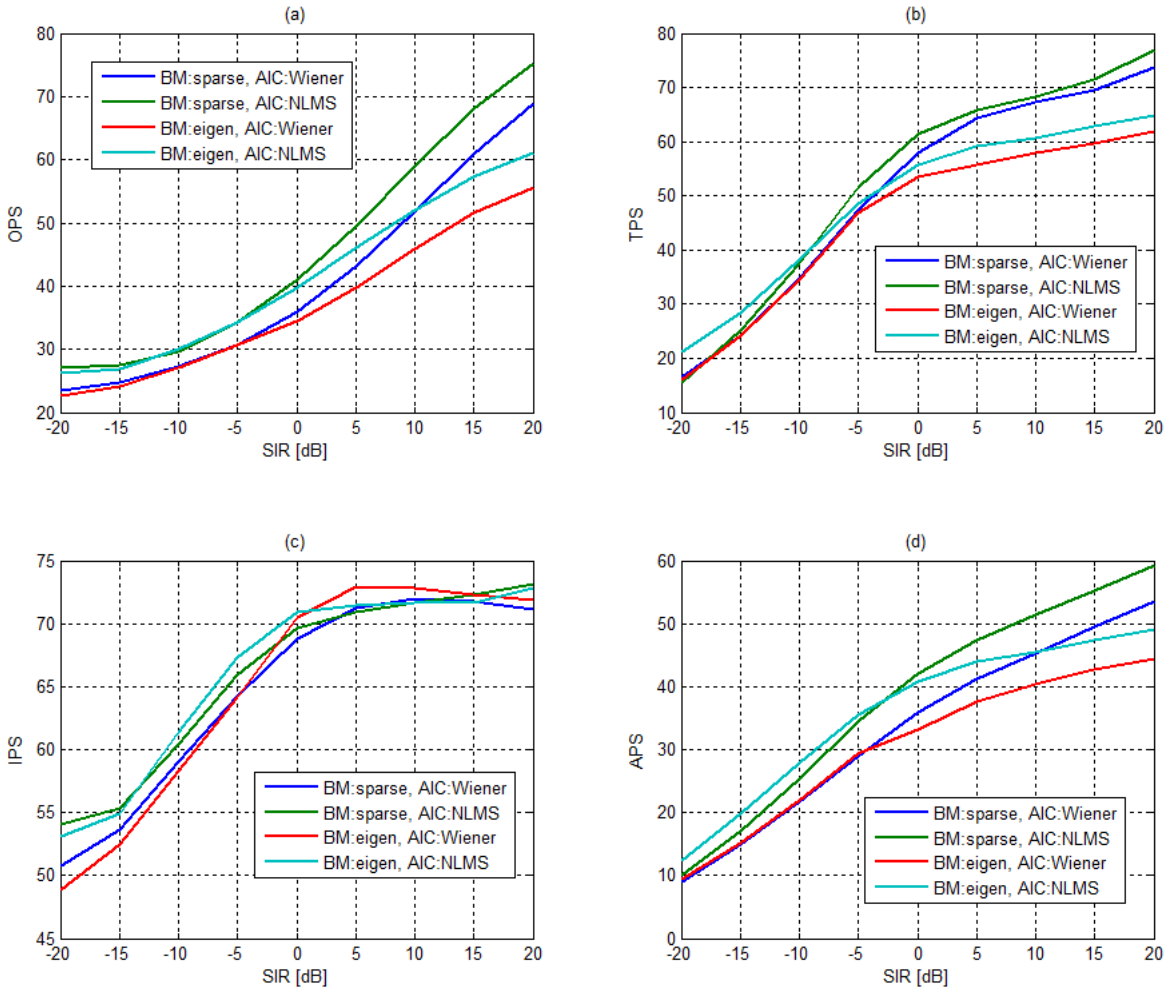

Figure 6.4: PEASS results for the second simulation scenario. (a) OPS, (b) TPS, (c) IPS, (d) APS.

Figure 6.5 shows the PESQ results for the second scenario. Again, the aforementioned combination achieves the highest MOS. For an unknown reason, the MOS rises below SIR levels of -10dB, where it should go further towards a score of 1. This rather unexpected behavior could not be resolved and may be another indicator that the PESQ measure is not well suited for evaluating speech enhancement algorithms.

The simulation results for the third scenario are shown in figure 6.6. Here, the best multichannel postfilter depending on two noisefloor estimation methods has to be determined. The MCSPP postfilter has the highest score in the OPS, TPS and APS categories. Only the IPS appears to be larger when using the TBRR method. For SIR levels above 5dB, the MCSPP is even the worst method. The influence on whether choosing the IMCRA or the Minimum Statistics noisefloor estimator is rather negligible. However, since the MCSPP postfilter is also the one with the least computational cost, it is chosen as postfilter method for the final MCSE algorithm.

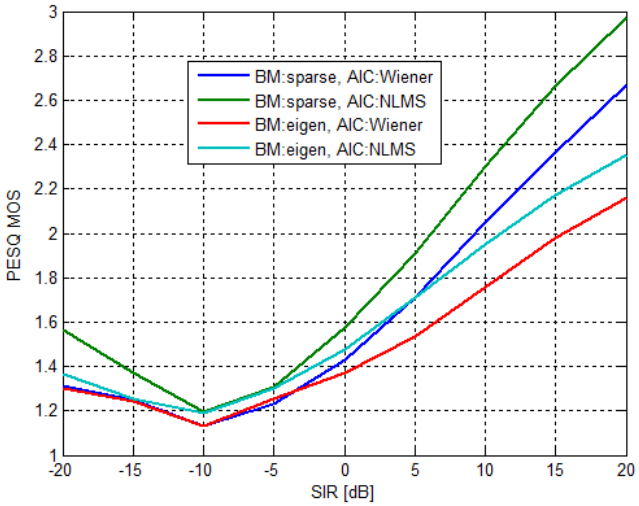

Figure 6.5: PESQ results for the second simulation scenario.

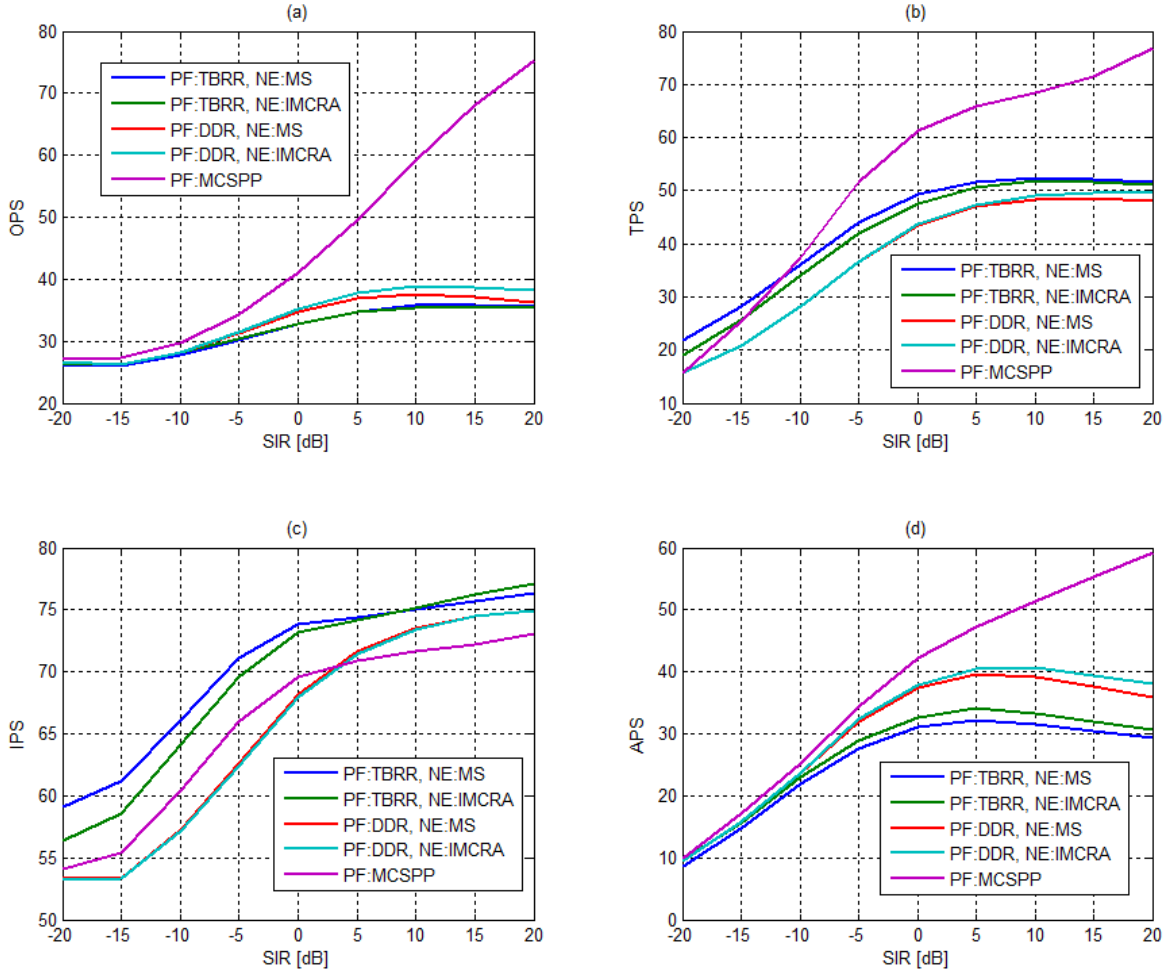

Figure 6.6: PEASS results for the third simulation scenario. (a) OPS, (b) TPS, (c) IPS, (d) APS.

Figure 6.7 shows the PESQ results for the third simulation scenario. The MCSPP multichannel postfilter also has the highest MOS score. But again, the MOS rises below -10dB for an unknown reason.

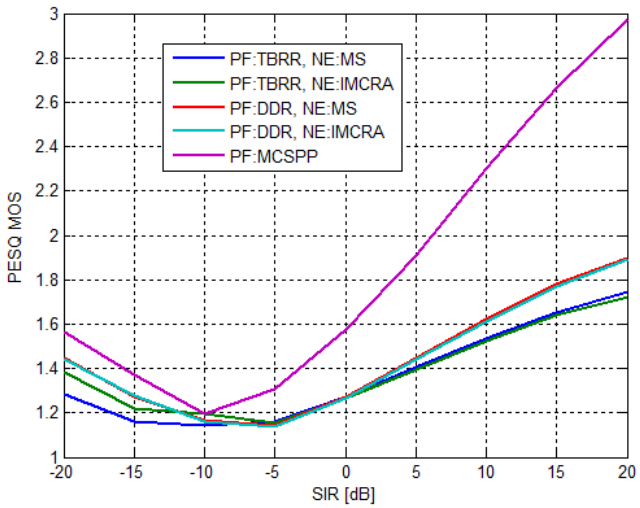

Figure 6.7: PESQ results for the third simulation scenario.

In the fourth simulation scenario, the influence of using the filterbank from chapter 5.4.2 is examined in terms of the PEASS scores. The best performing algorithms of the first three scenarios are used for the fourth one. From observing the PEASS results in figure 6.8 it can be learned that the influence is rather minimal compared to using no filterbank at all. As expected, the more bands used the better the overall score (OPS) gets. The other three scores do not give a distinct discrimination criterion. Even though the benefit of using more bands is almost vanishing, 48 bands are used for the final MCSE system, just to be on the safe side. Moreover, using 48 bands instead of 257 FFT bins when using $f_s = 16kHz$ and a framelength of $32ms$ already gives a fivefold reduced computational cost.

Also the PEQS results for the fourth simulation scenario in figure 6.9 are very close together, thereby not allowing for a unique conclusion to be drawn.

The last simulation scenario compares the influence of the number of microphones used in the speech enhancement system. Figure 6.10 gives the PEASS scores as a function of the input SIR. The multichannel speech enhancement system performs better in terms of the OPS than the single channel speech enhancement system throughout all SIR levels. This is expected, since the multichannel system can also use the spatial information, while the single channel system only relies on signal statistics. From the OPS it can also be seen that using more than two microphones is only of benefit when the SIR is low. For higher SIRs it is not important whether two or four microphones are used. This finding is of great importance for the implementation, since using more microphones also means higher cost both in hardware and software terms. While these costs clearly double, the benefit in terms of achievable speech quality is rather negligible. For SIRs above 10dB, using 2 or 3 microphones gives better results than using 4 or the true RTF information, which is probably an error. The TPS is rather similar for a different number of microphones, even for the single channel algorithm. The IPS follows smoothly the expected behavior: The more microphones are used, the more interfering noise components can be removed. For the true RTF information, this score is high throughout all SIR levels. The APS is somewhat difficult to interpret: The true RTF features the worst score while using a single microphone gives the highest score for low SIR levels, followed by two microphones. It seems as if this score was somehow inverted. However, listening to the resulting enhanced speech

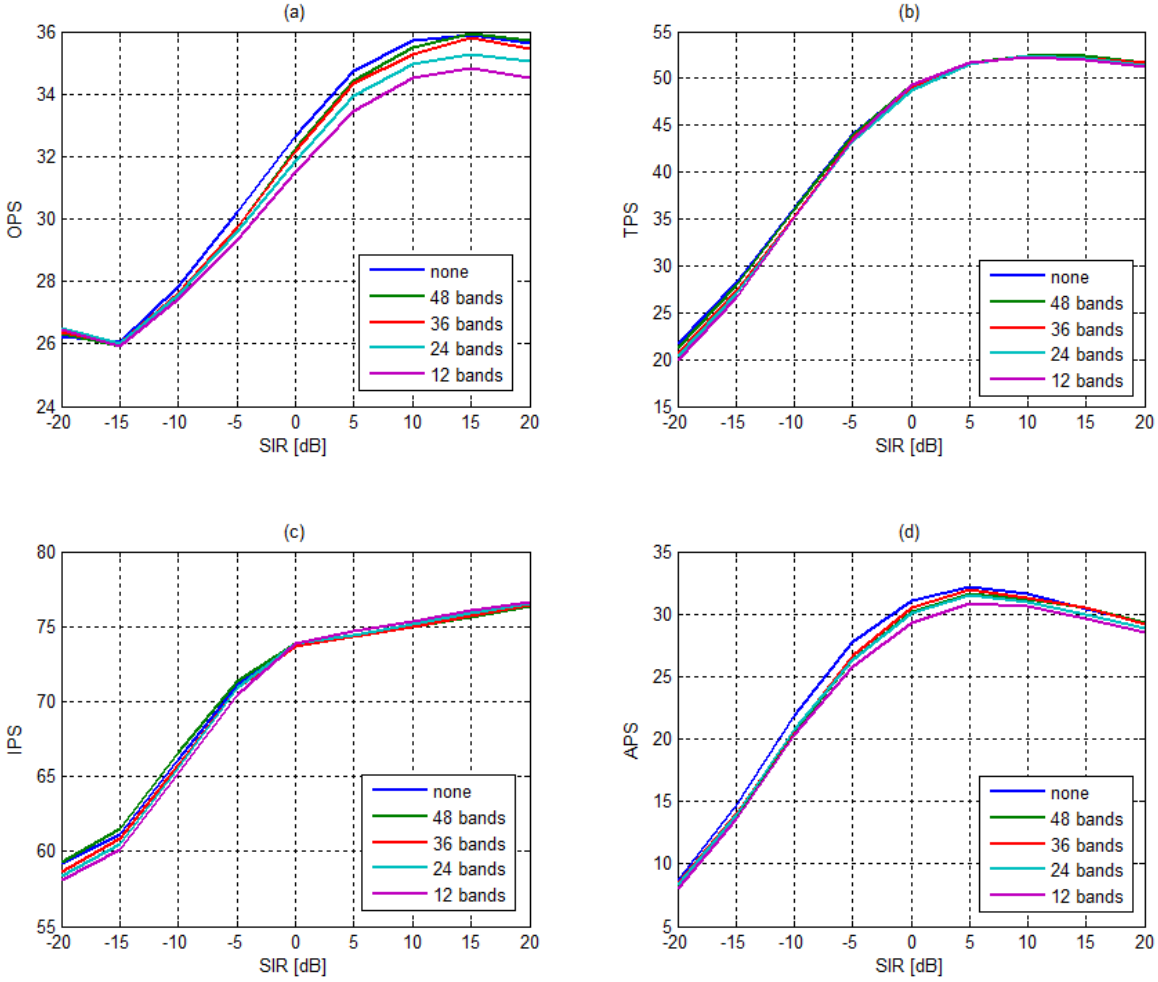

Figure 6.8: PEASS results for the fourth simulation scenario. (a) OPS, (b) TPS, (c) IPS, (d) APS.

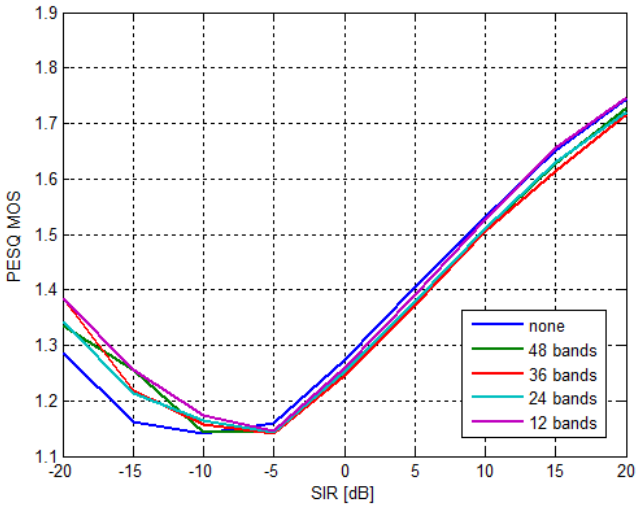

Figure 6.9: PESQ results for the fourth simulation scenario.

signals gives the correct impression of less musical artifacts when using more microphones.

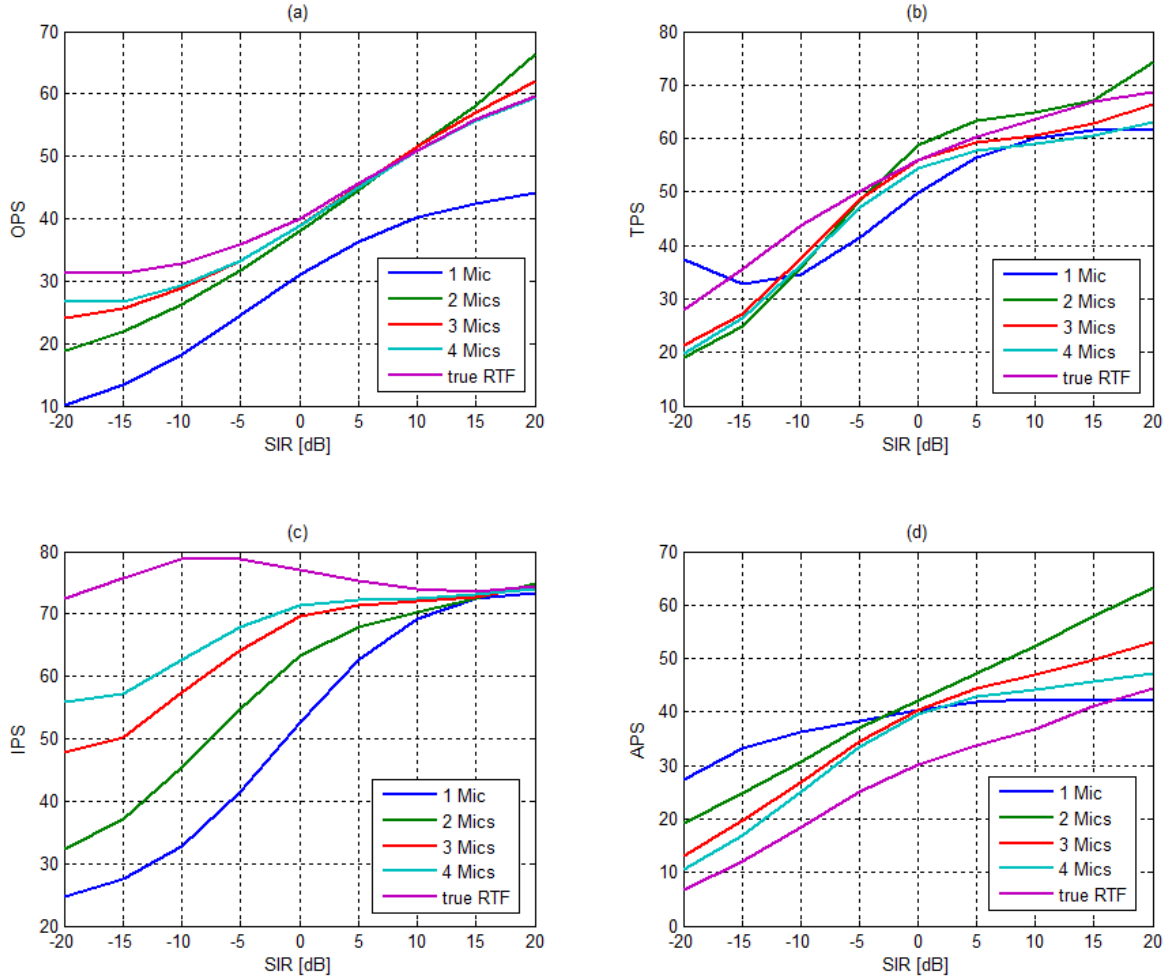

Figure 6.10: PEASS results for the fifth simulation scenario.

The PESQ results for the fifth simulation scenario shows the expected behavior of a rising MOS when using more microphones. Still, the MOS rises for SIR levels below -10dB. It seems that the PESQ measure is inaccurate for low SNR levels.

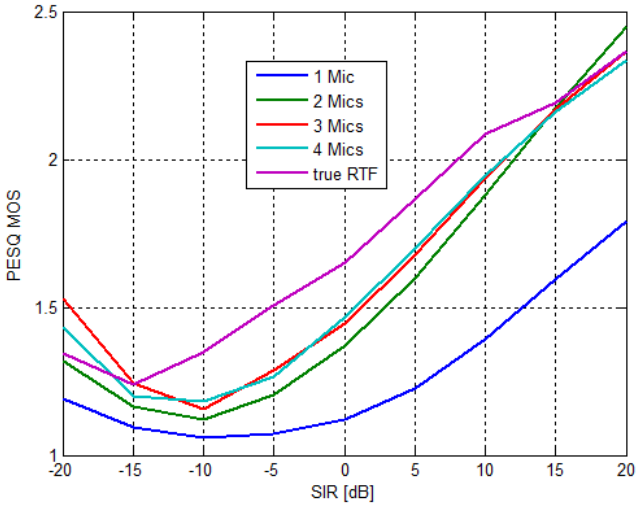

Figure 6.11: PESQ results for the fifth simulation scenario.

6.7 Performance of the best Combination

Based on the performance results of the five test scenarios, the optimal combination of RTF estimation, blocking matrix structure, AIC structure and postfilter algorithm has been determined. This combination provides the building blocks of the final MCSE algorithm, illustrated in figure 6.12. The RTF estimation is performed by a DOA estimate using the SCOT algorithm, combined with the magnitude estimation from section 4.3.4. As blocking matrix the sparse structure from chapter 3.6.4 is used. The AIC filters are adapted by the NLMS algorithm, as presented in chapter 3.5. And the best postfilter is the MCSPP from section 5.3.3, which has been combined with a simplified filterbank using 48 bands. This reduces the computational cost by a factor of 5 while maintaining the performance of the postfilter.

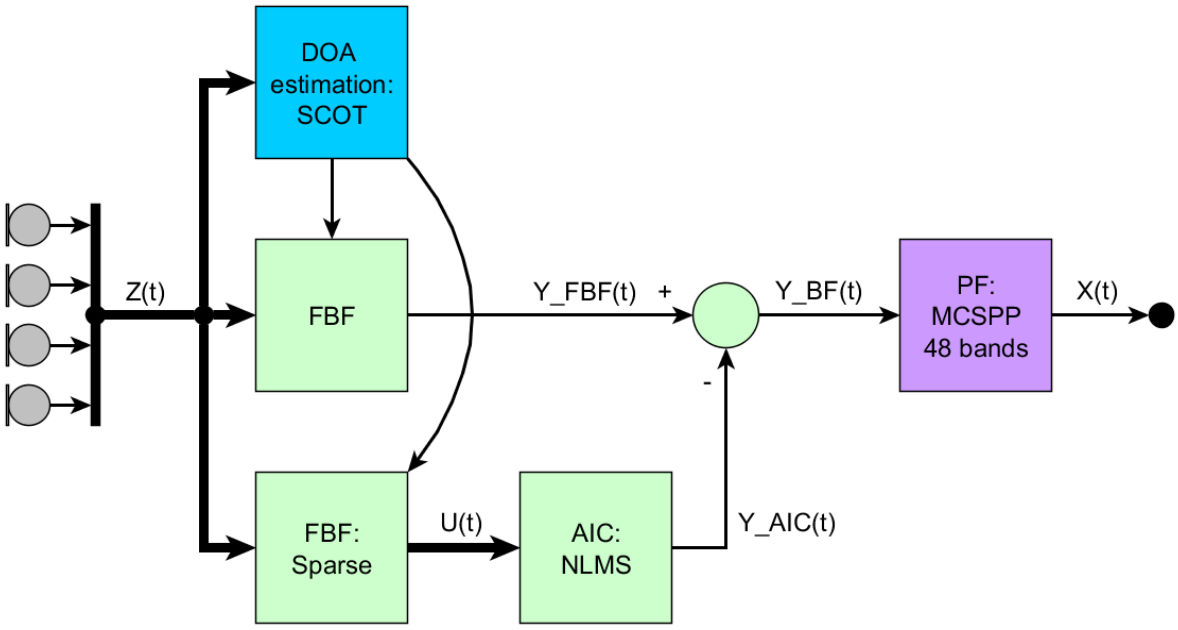

Figure 6.12: Final MCSE algorithm.

The performance of the final MCSE system is presented by observing the most important signals of the RTF estimation module, the GSC beamformer and the MCSPP postfilter. Apart from the determined parameters of the simulation scenarios, the following settings have been chosen to give a representative testcase for the MCSE algorithm:

```
       doa_type: PHAT
       rtf_type: DOA
        bm_type: sparse
       aic_type: NLMS
      mcpf_type: MCSPP
speech_database: KCOSS
     speech_ATF: B208_20deg.wav
      noisefile: Noise Samples\Roadnoise_4ch.wav
         Nbands: 48
          Nmics: 4
 use_filterbank: yes
     target_SIR: 0
             fs: 16000
        samples: 240000
```

This setup uses 15 seconds of audio using the KCOSS speech database and a set of ATFs recorded from an impinging speech signal angle of 20°, together with a multichannel noise sample recorded at a busy street. The SIR is set to 0dB, hence speech and noise are of equal power.

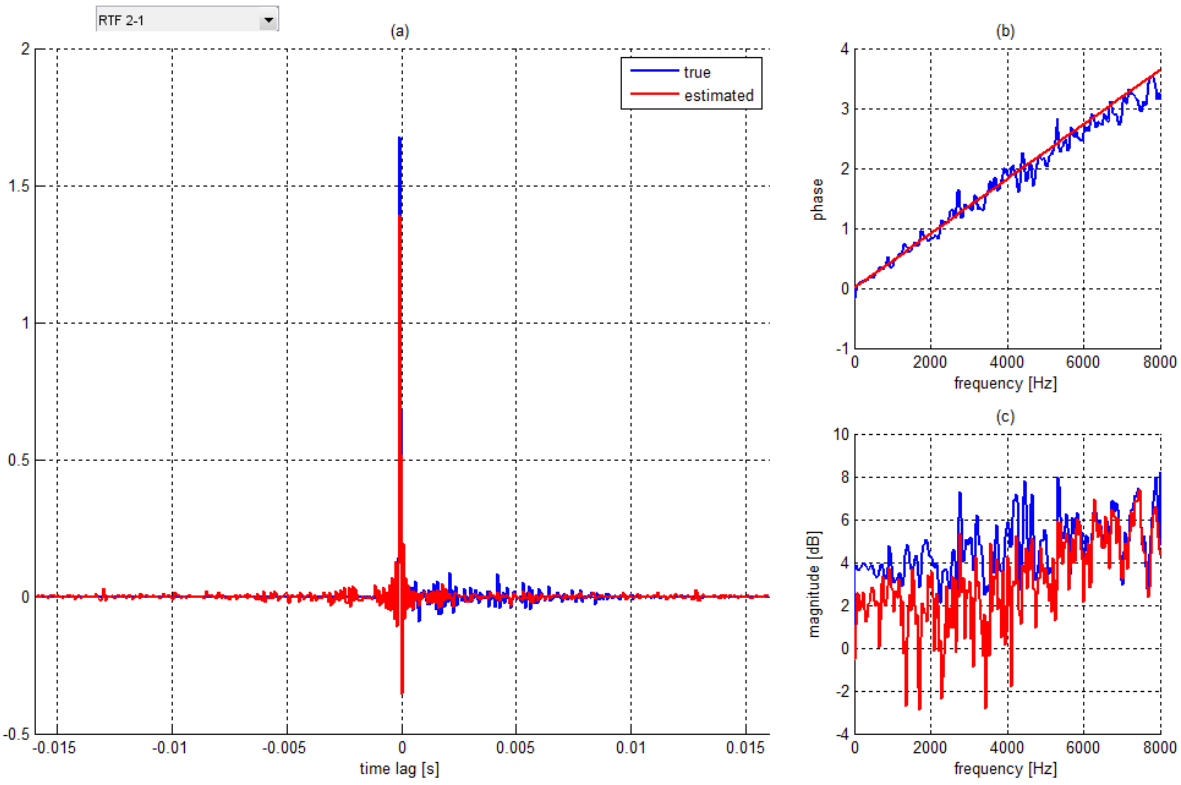

Figure 6.13: (a) DOA estimate of the RTF $\tilde{A}_2(j\Omega)$ by using the SCOT algorithm. (b) phase spectrum. (c) magnitude spectrum

First, the performance of the RTF estimate is studied. In figure 6.13 the resulting RTF from the DOA estimate achieved by the SCOT algorithm is shown. In panel (b) the estimated delay

$e^{-j\Omega f_s \Delta\hat{\tau}_m}$ is plotted as a function of the frequency. The red line indicates the estimate, and the blue one the true unwrapped angle of the RTF to be estimated. Panel (c) shows the magnitude which has been estimated using the algorithm of section 4.3.4. Estimation errors occur mainly in low frequencies, due to the high content of additive noise components. Magnitude and phase together give the RTF estimate. Panel (a) shows its IFFT. It may be observed that the RTF forms a non-causal FIR filter. Due to the frame length of 32ms, delays up to $\pm 16ms$ can be measured. This is more than sufficient for a speaker standing at a distance of 0.5m to the array. It can be seen that this RTF estimate adequately matches the true RTF.

Figure 6.14 presents the speech presence probability (SPP) of the SCOT algorithm evaluated for 50 different target angles over time, calculated from the coherence between the first two microphones. The SPP is obtained by taking the inverse of the cost function defined in equation 4.32. It can be seen that the correct DOA is found to be at about +20°. However, some outliers do exist around -30°. They can be easily rejected by defining a maximal speed of target movement. In other words, the speaker cannot move from +20°to -30°within less than 1 second.

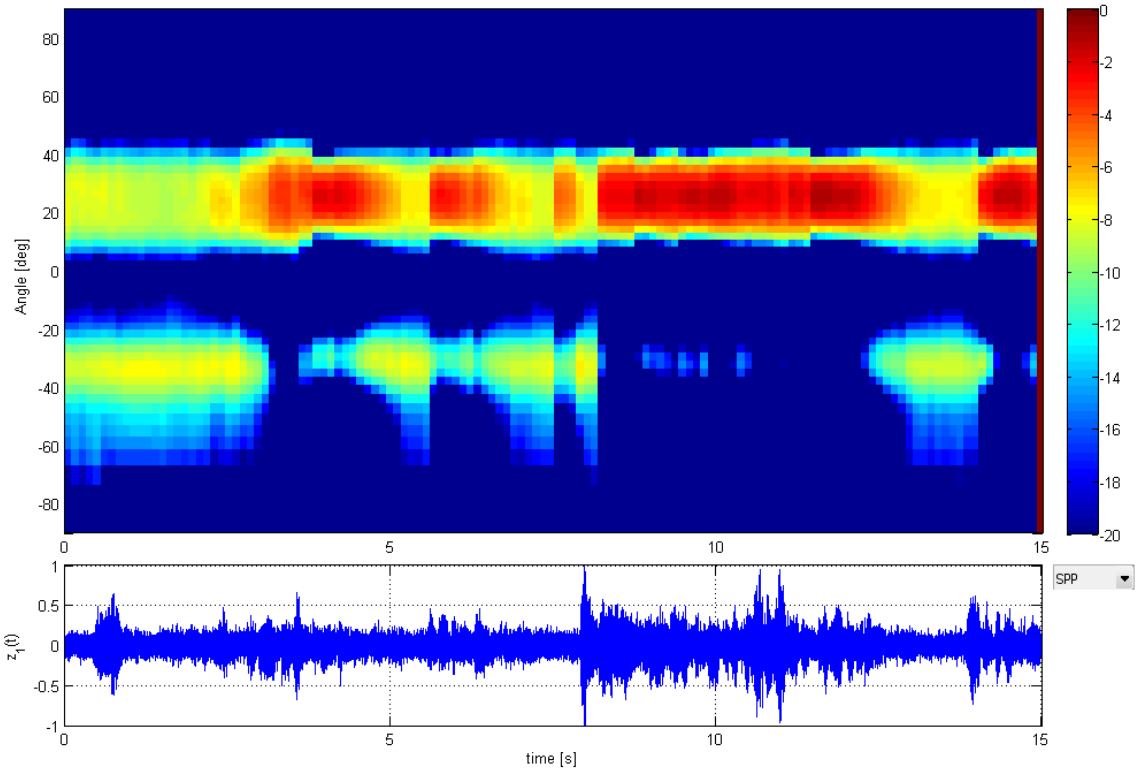

Figure 6.14: Speech presence probability of the SCOT algorithm, calculated from the coherence between microphone 1 and 2, drawn for impinging speech angles from -90 ˚to +90 ˚. The lower plot shows the waveform of $z_1(t)$.

From the RTF estimate, the fixed beamformer (FBF) and the blocking matrix (BM) are calculated. Then, the GSC adapts the AIC filters using the NLMS algorithm to produce the beamformer output. The following figures show the input signal $Z_1(j\Omega)$, the signal after the FBF $Y_{FBF}(j\Omega)$ and the output signal of the beamformer $Y(j\Omega)$ in figures 6.15, 6.16 and 6.17, respectively. It can be seen that the FBF mainly reduces the noise at higher frequencies. This is in accordance to the directivity pattern of the delay-and-sum beamformer illustrated in figure 3.3, where a noteworthy directivity can only be observed for higher frequencies. At the output of the GSC, the signal $Y(j\Omega)$ also features some noise reduction at low frequencies, according

to the MVDR constraint.

The beamformer output also depends on the AIC and the BM, whose output signals are studied in the following. Figure 6.18 shows the first noise reference $U_1(j\Omega)$ at the output of the blocking matrix. It can be seen, that the noise reference is almost entirely free of the speech signal, hence leakage is low. This is due to the close vicinity of the speaker to the array. This is a mandatory condition for this MCSE system. In figure 6.19 the output of the adaptive interference canceler $Y_{AIC}(j\Omega)$ is shown. The multichannel NLMS algorithm matches the output of the BM to the noise components of $Y_{FBF}(j\Omega)$, as explained in section 3.5.

To further enhance the beamformer output shown in figure 6.17, the MCSPP postfilter is employed. It estimates the noisefloor PSD at the beamformer output as explained in chapter 5.3.3. This PSD is shown as spectrogram in figure 6.20. Clearly visible is the logarithmic layout of the 48 filterbank bands being used. Based on this signal, the multichannel speech presence probability $p(k, j\Omega)$ is calculated and shown in figure 6.21. It is used to obtain the OM-MMSE-LSA gain function $G(k, j\Omega)$ from chapter 5.2.4. This gain function is illustrated in figure 6.22. It is limited to a maximal attenuation of $G_{min} = -15dB$, to keep the signal from being canceled down to zero in the case of speech absence. The gain is applied to the magnitude spectrum of the beamformer output $Y(j\Omega)$ to achieve further reduction of the unwanted noise signal. This enhanced signal is the final output of the MCSE system and termed $X(j\Omega)$. It is shown in figure 6.23. It can be seen that the transient noise components visible in the noise reference $U_1(j\Omega)$ in figure 6.18 have almost completely vanished. Such a performance cannot be achieved by the single-channel methods. This is possible because of the exploitation of the spatial information of the noise sound field by the beamformer. For comparison, the ground truth of the speech component $S_1(j\Omega)$ of the first microphone is shown in figure 6.24.

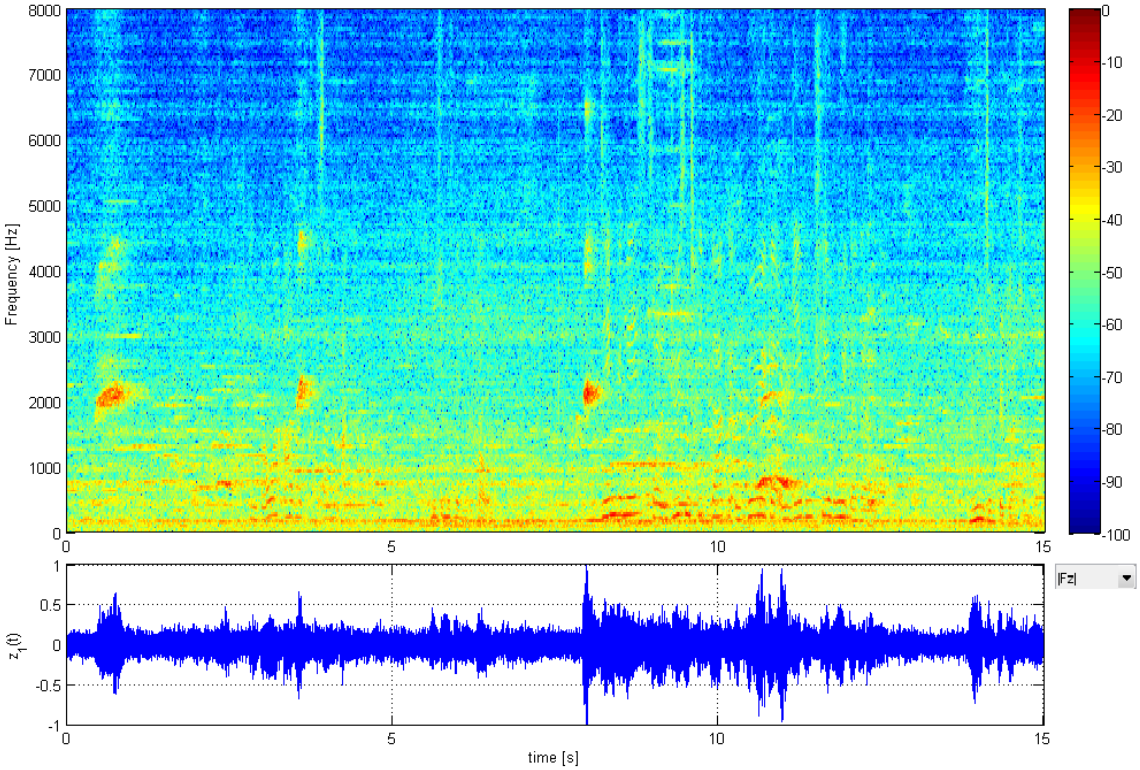

Figure 6.15: Spectrogram and waveform of the input signal $Z_1(j\Omega)$.

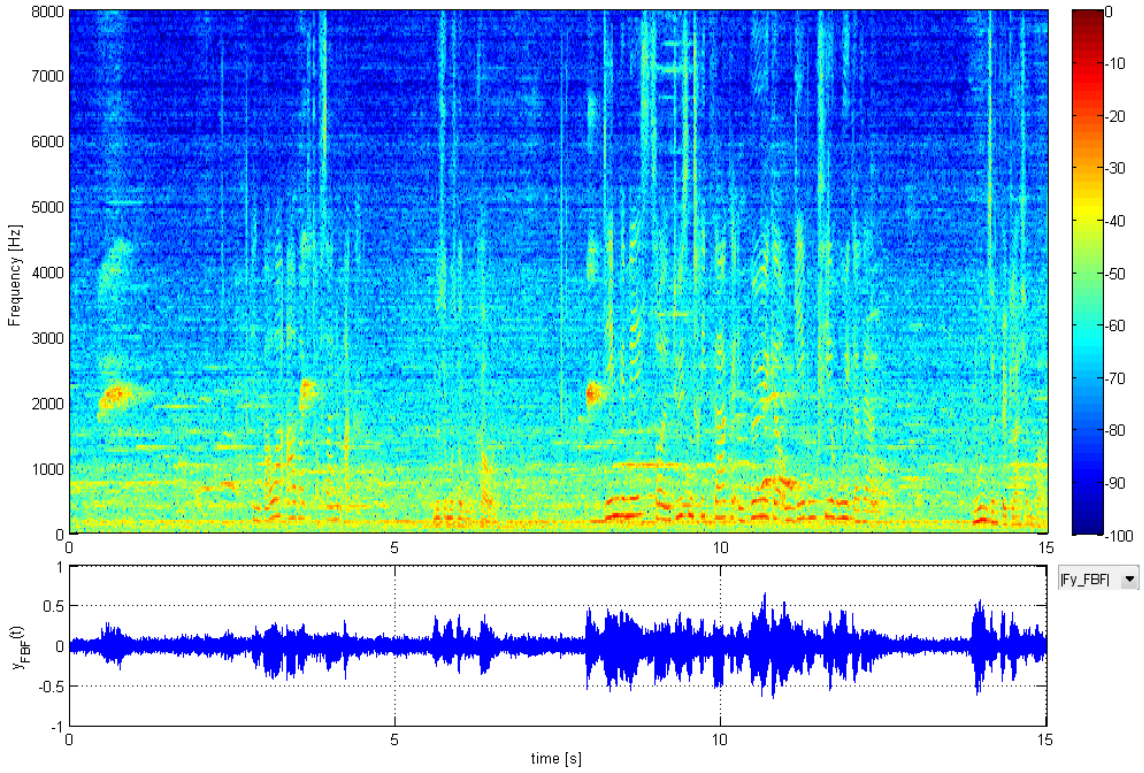

Figure 6.16: Spectrogram and waveform after the fixed beamformer $Y_{FBF}(j\Omega)$.

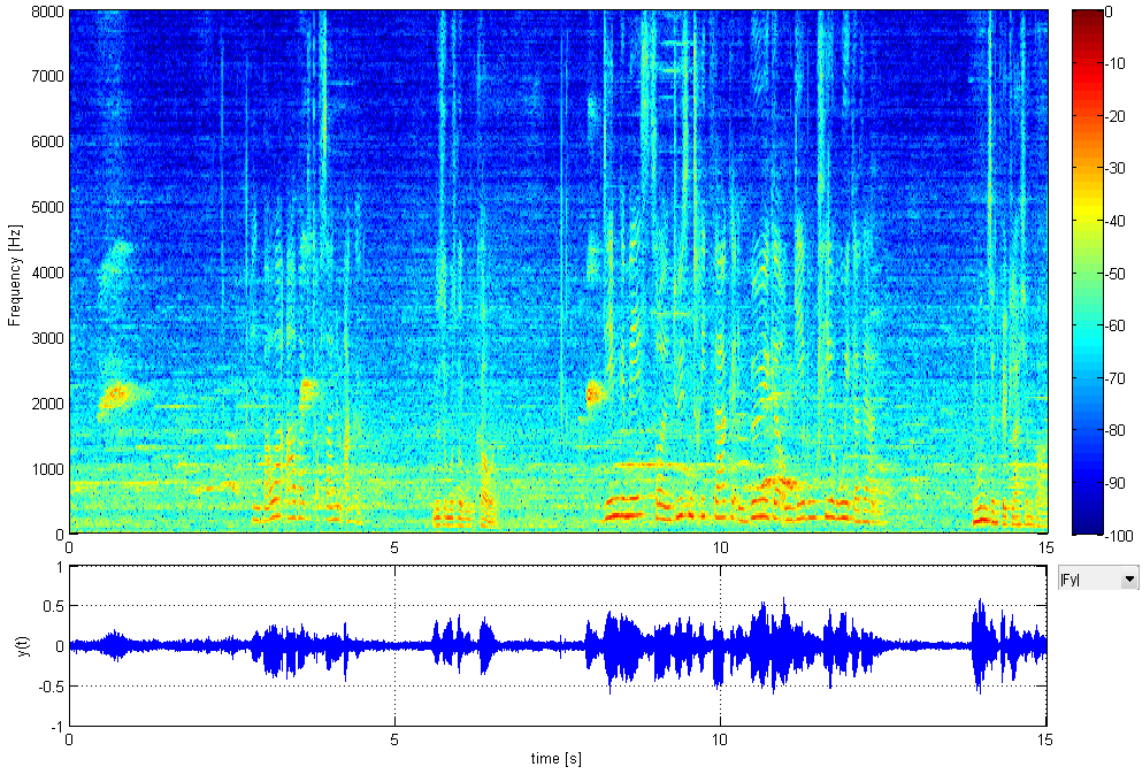

Figure 6.17: Spectrogram and waveform at the beamformer output $Y(j\Omega)$.

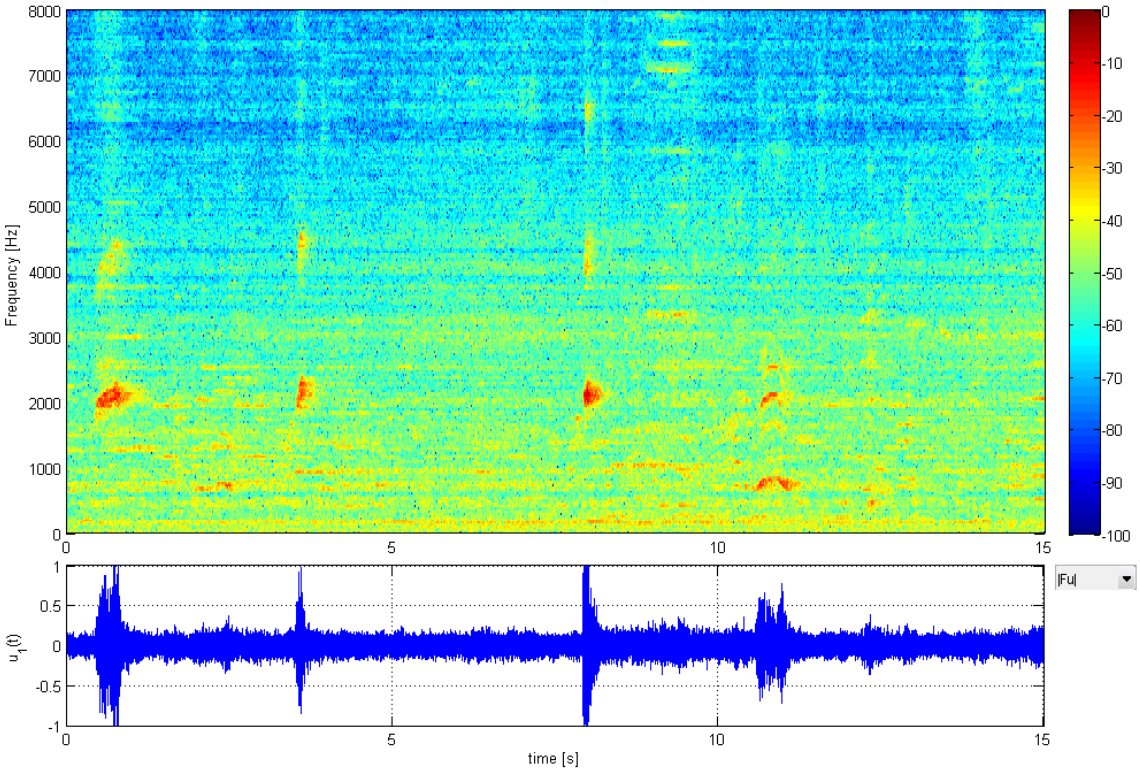

Figure 6.18: Spectrogram and waveform at the blocking matrix $U_1(j\Omega)$.

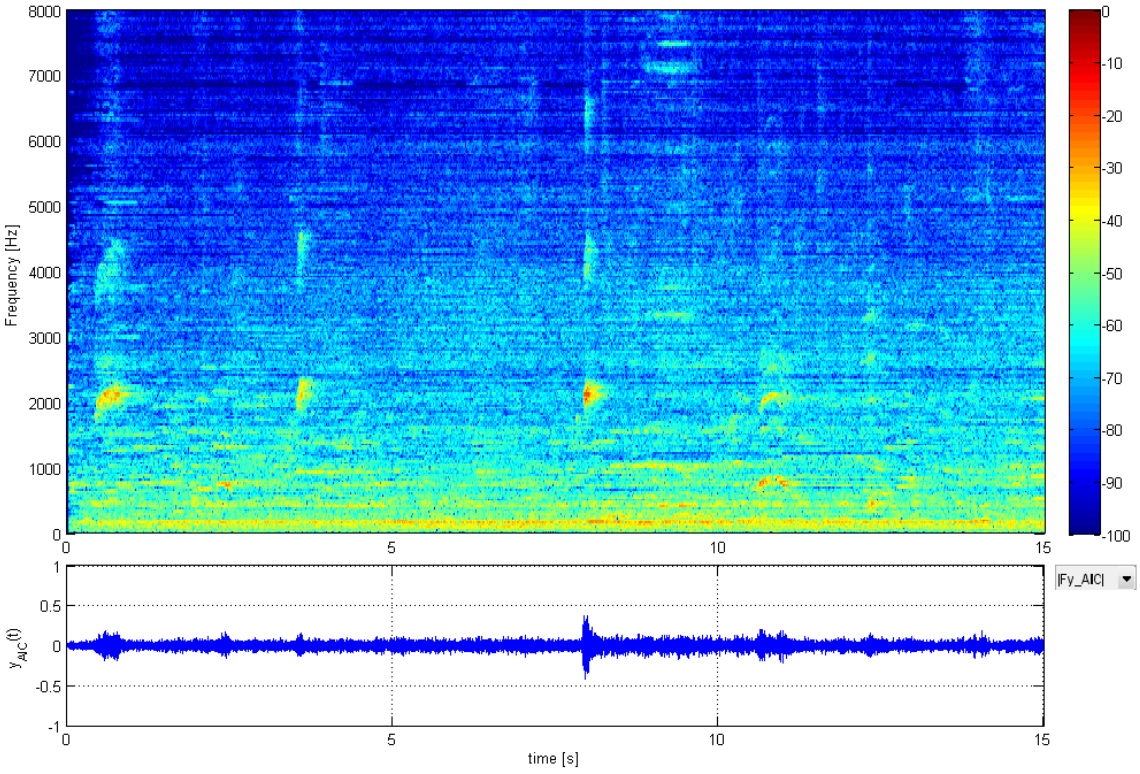

Figure 6.19: Spectrogram and waveform at the adaptive interference canceler $Y_{AIC}(j\Omega)$.

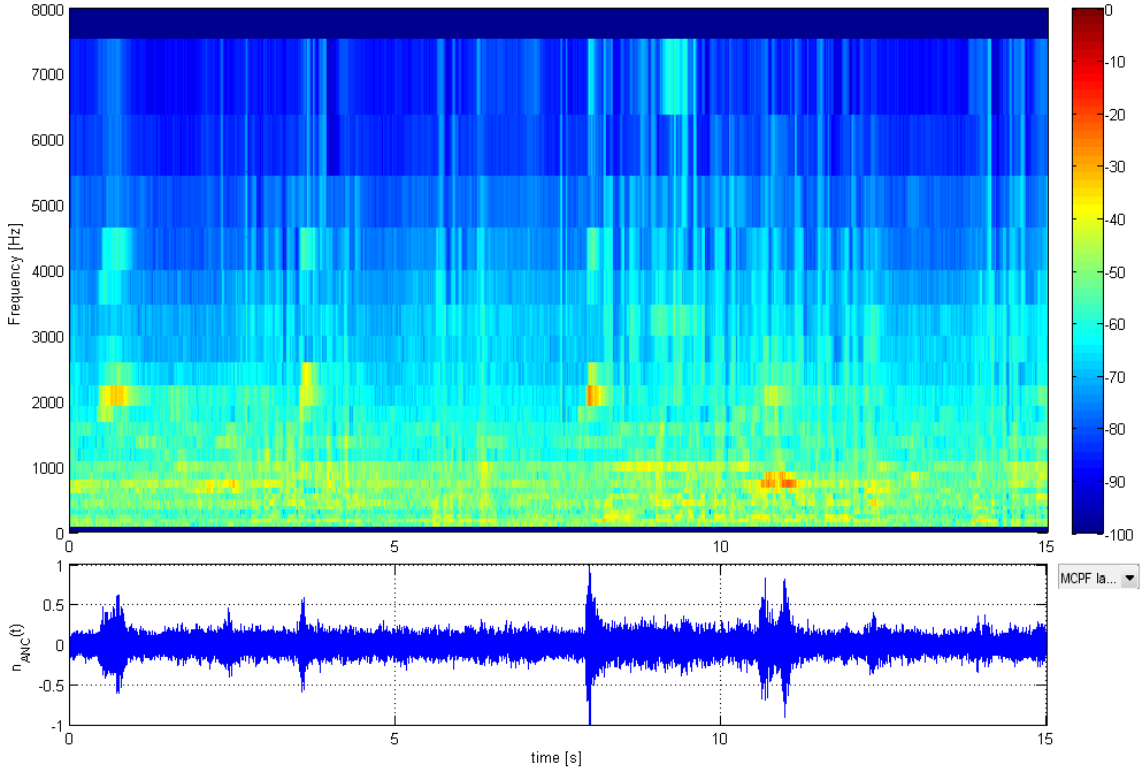

Figure 6.20: Spectrogram of the estimated noisefloor PSD $\hat{\sigma}_n^2(k, j\Omega)$ at the beamformer output. The lower plot shows the waveform of the corresponding noise component at the first microphone $\hat{n}_(t)$, calculated with equation 5.43.

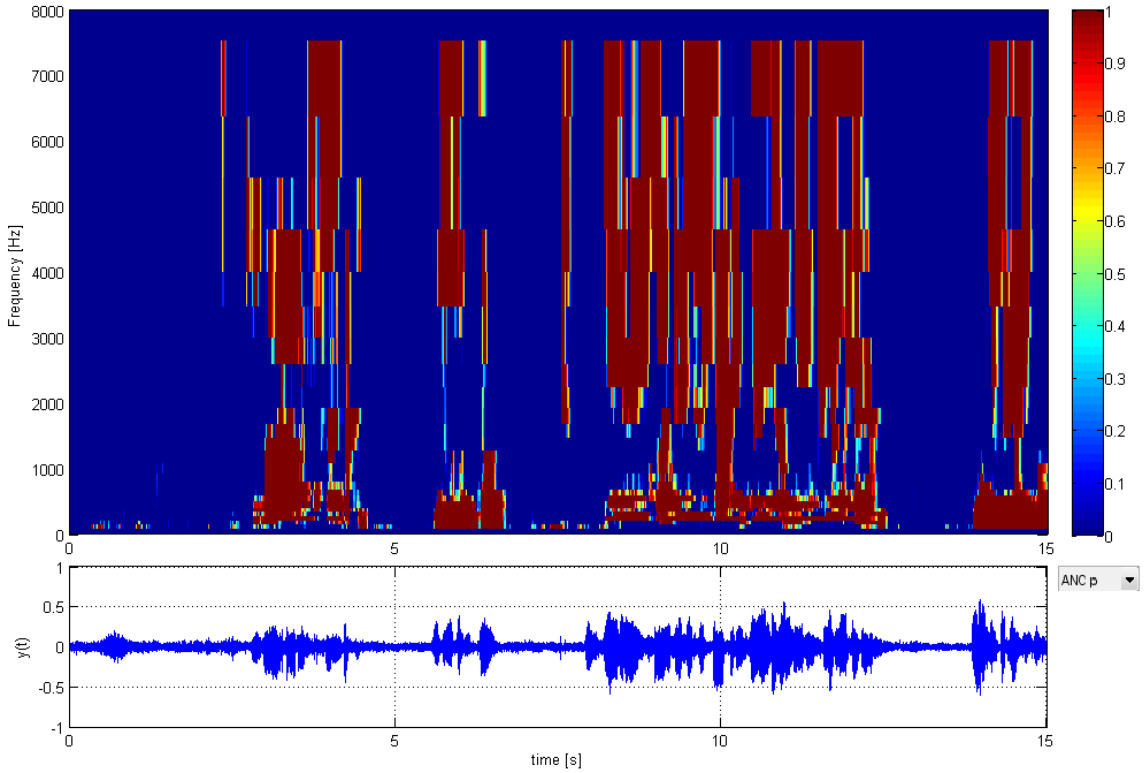

Figure 6.21: Spectrogram of the multichannel speech probability $p(k, j\Omega)$. The waveform below shows the output of the beamformer $y_(t)$.

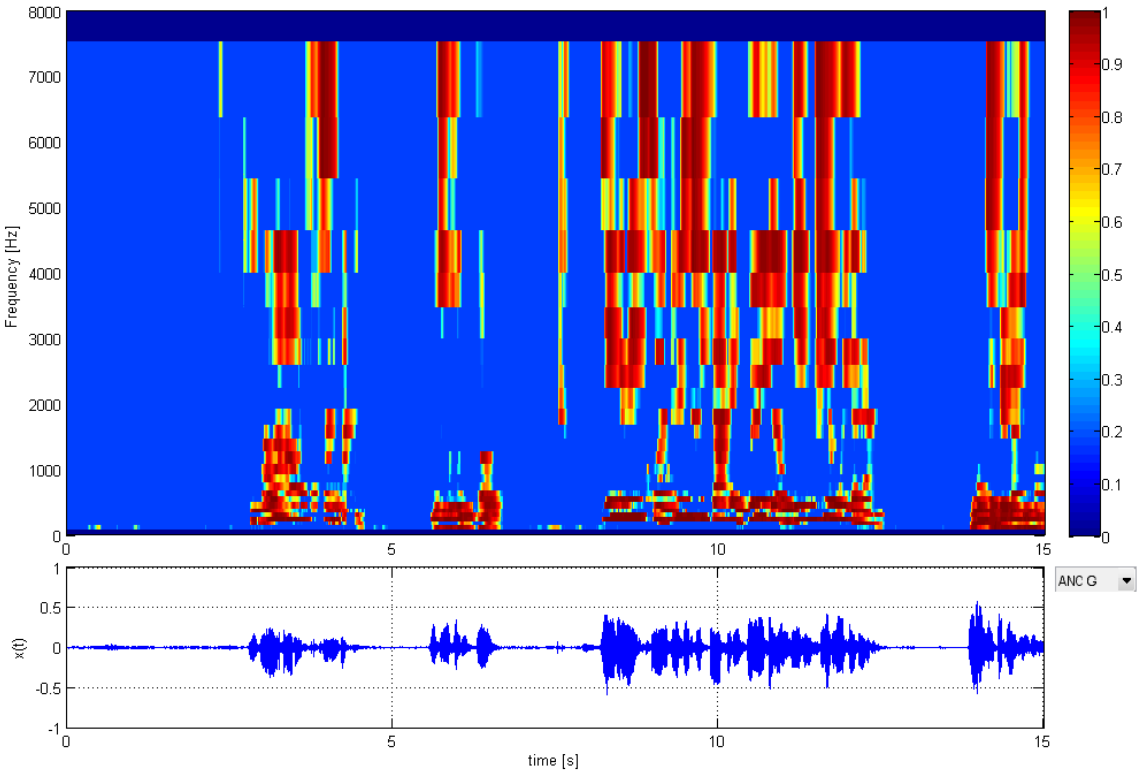

Figure 6.22: Spectrogram of the gain function $G(k, j\Omega)$. The waveform below shows the output of the post-filter $x(t)$.

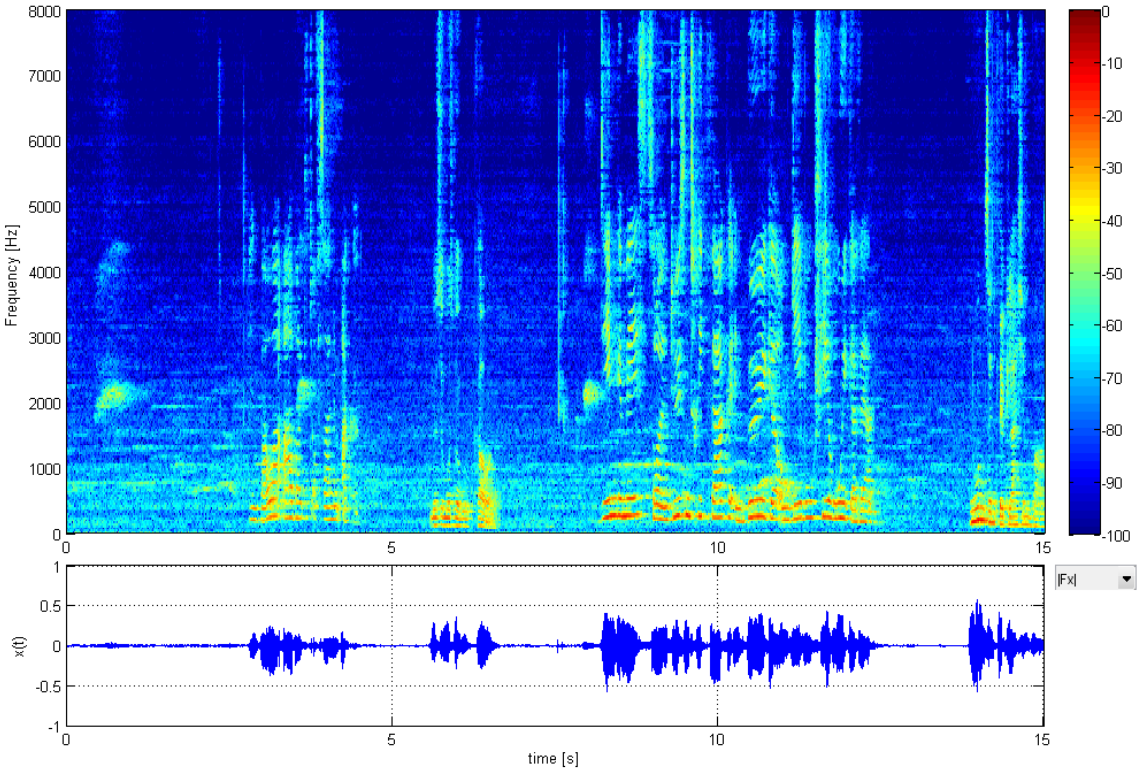

Figure 6.23: Spectrogram and waveform at the output of the MCSPP postfilter $X(j\Omega)$.

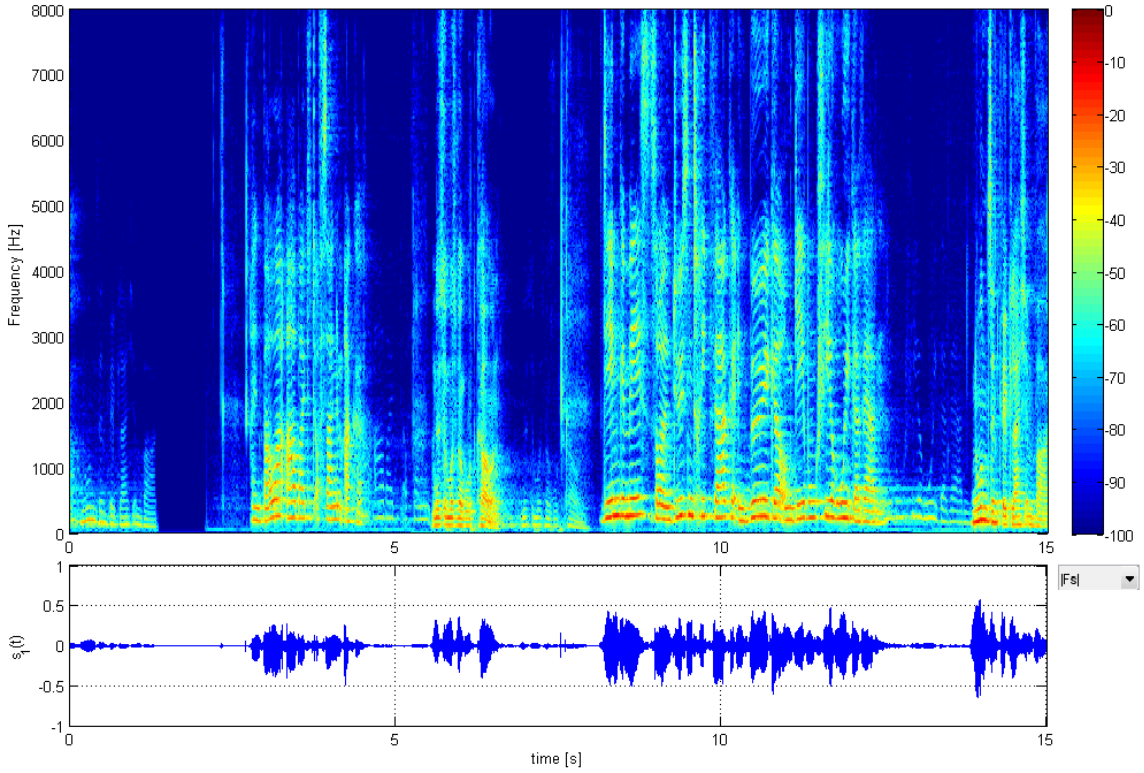

Figure 6.24: Spectrogram and waveform of the speech component at the first microphone $S_1(j\Omega)$.

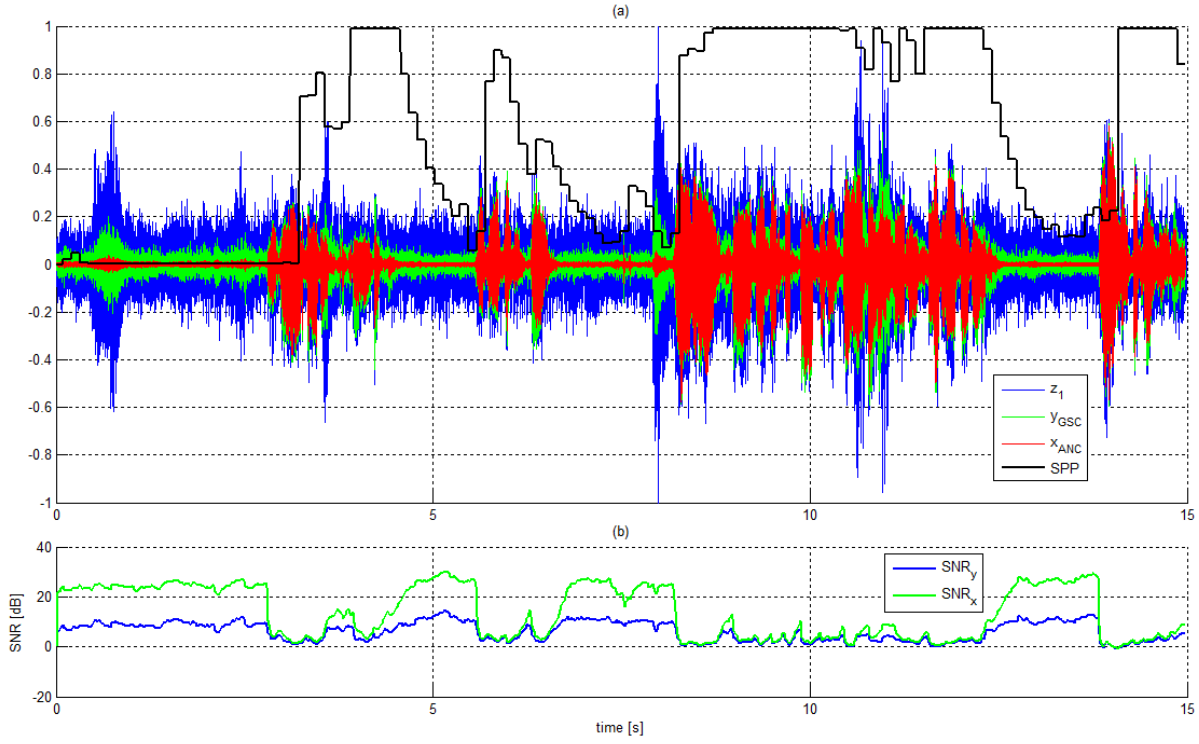

Figure 6.25: All relevant signals of the MCSE system combined in a single plot. (a) input signal $z_1(t)$ in blue, the beamformer output $y(t)$ in green, the output of the multichannel postfilter $x(t)$ in red and the speech presence probability (SPP) obtained from the VAD using equation 4.47 in black. (b) SNR per frame of the beamformer output in blue, and the SNR of the postfilter output in green.

7 Realtime Implementation

In this chapter, the process of building a real-time capable prototype of the final MCSE algorithm is documented. First, both the hardware and software prerequisites are going to be defined. Then, the implementation is described by using rapid prototyping. And finally, further steps on how to port this implementation to an embedded target are presented.

7.1 Prototype Implementation

All the building blocks of the final MCSE algorithm in figure 6.12 have been written in Matlab, as described in section 6.4. This code can only operate on offline audio data, due to the slow execution speed of Matlab scripts. Therefore, a fast and real-time capable implementation has to be performed, which serves as a proof of concept for the MCSE algorithm under realistic conditions. As programming language for the prototype C++ has been chosen for a number of reasons: It is about 1000 times faster than Matlab code. Also, numerous libraries exist for calculating the FFT or performing matrix operations with complex numbers. Further, frameworks for connecting audio hardware and passing audio data to and from a sound device is readily available and well tested. And finally, the code can be developed on a PC and ported to a different hardware platform, like the WandBoard featuring the state of the art ARM Cortex A9 CPU [15].

7.1.1 Hardware Requirements

For the prototype implementation, four microphone signals are required. Built-in soundcards have at most two line inputs and a single microphone preamp. Therefore an external sound interface is employed. The *MOTU 4pre* interface has four digitally adjustable microphone preamps and an ASIO interface, which allows for both a small and constant hardware delay. As microphones, the four *Audix TM1 Plus* measurement microphones from the ATF measurements in section 2.7 have been re-used. These microphones definitely have a much smaller variance in sensitivity compared to low-cost electret condenser microphones (ECMs). However, that is secondary because the magnitude estimation from chapter 4.3.4 takes care of these variances anyway. In an actual embedded system micro-electromechanical (MEMS) microphones are used, which also have low manufacturing tolerances. The hardware setup being used can be seen in figure 7.1

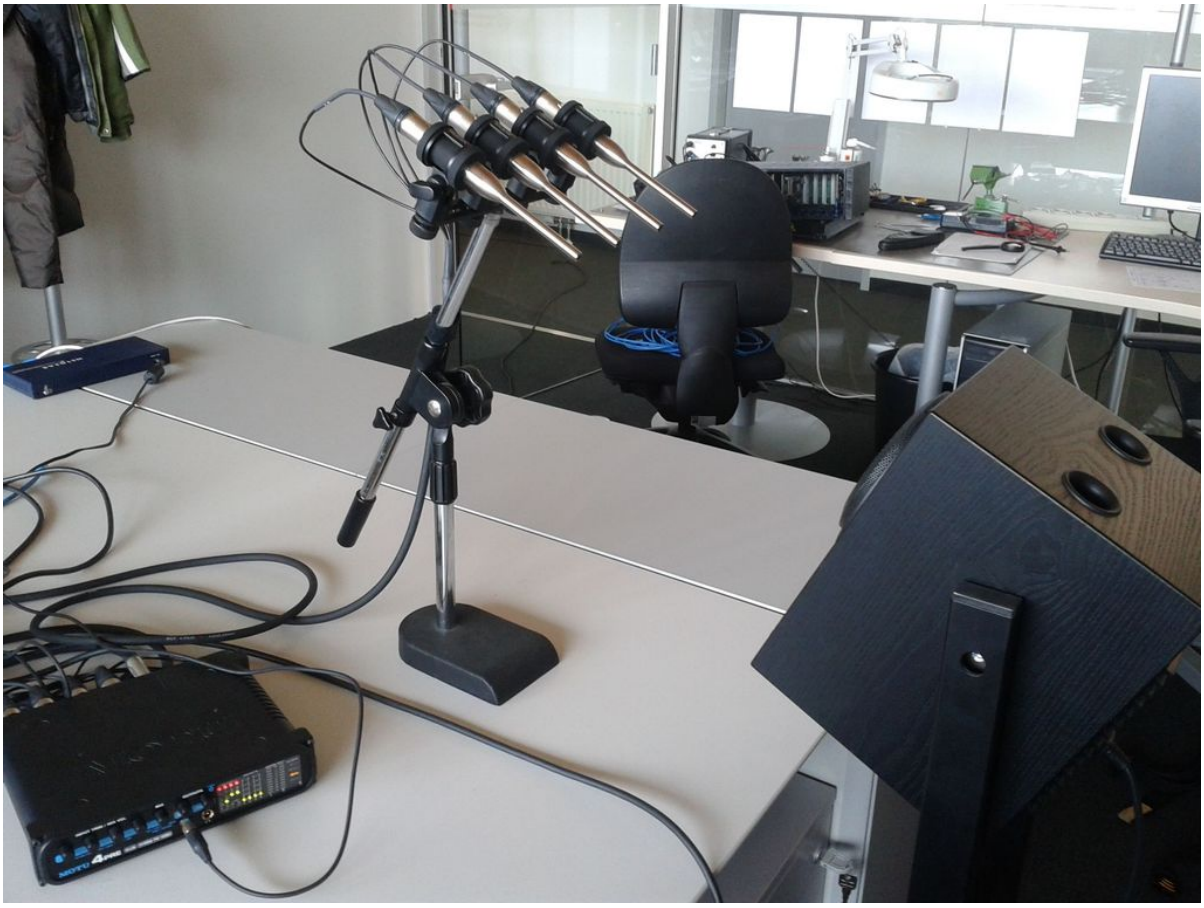

Figure 7.1: Hardware used for the prototype: The four Audix TM1 Plus microphones can be recognized in a linear array combination. In the lower left corner is the MOTU 4pre ASIO interface. The lower right corner shows the Geithain RL906 measurement loudspeaker used for recording the ATFs.

7.1.2 Software Requirements

To cut the implementation complexity of the prototype to a bare minimum, available software packages should be used wherever possible. Further, the code is be written in plain C++ in order to run on all kinds of operating systems and platforms. However, the ASIO interface is only available for Windows, therefore the prototype has been compiled for Windows. Apart from the audio driver, the complete MCSE implementation is required to be fully interchangeable between underlying operating systems and processor architectures. Therefore, only cross-platform libraries and frameworks have been used:

- The PortAudio framework [62] is used for simultaneous playback and recording of multiple audio streams, using ALSA, ASIO or DirectSound sound drivers. In order to work with the MOTU interface, ASIO support has been compiled into PortAudio using the ASIO Software Development Kit [63] provided by Steinberg Media Technologies.

- The FFTW library is used for the efficient calculation of the real-valued, single dimension FFTs and IFFTs needed for the MCSE algorithm. FFTW has been developed by Matteo Frigo and Steven G. Johnson at the MIT [64], and is available for many different platforms.

- The Eigen Matrix Library [65] simplifies all kinds of matrix operations using complex elements to a huge extent. Employing Eigen allows for a Matlab-like programming style when working with matrices of arbitrary sizes and data types.

- The RTF estimation task does not need to be executed in realtime, unlike the GSC and the postfilter. In fact, it is sufficient to estimate the RTF once every 0.10s, since the speaker is not expected to move very fast in front of the microphone array. Therefore, the RTF estimation is executed in a separate thread. To make threading fully compatible amongst Windows and Linux, the Posix standard has been used. For Windows, a wrapper is available; the Posix Threads for Windows API [66].

7.2 Rapid Prototyping

The effort for the C++ implementation should be as small as possible, and the verification and validation phases needs to be fast and efficient. Usually, software testing consumes about 80 % of the entire implementation time and requires comprehensive unit tests to be conceived and written. This step can be dramatically reduced in time by using the concept of *Rapid Prototyping* in Matlab.

Since a Matlab implementation of the entire MCSE code is readily available and tested, it is unnecessary to freshly implement everything from scratch in C++, including the tests. The code is re-written with the Eigen library and verified against the Matlab simulations which are known to work correctly. This verification phase is done using the Matlab Engine, an extension which allows to call Matlab functions within C-code. It works like the well known MEX-functions. Using these mechanisms, the entire porting to C++ took only about two weeks.

7.2.1 C++ Implementation

In figure 7.2 an activity diagram of the MCSE system has been created using UML. The code consists of three main blocks: An audio callback, the GSC and postfilter code and the source location thread. The audio callback is provided by the sound hardware and delivers frames of 16ms of audio data in near real-time. That is, with a delay of 15-25ms depending on the ASIO driver. The 16ms frames are concatenated to 32ms blocks with 50% overlap. These blocks are weighted with a hanning window and then converted to the frequency domain, before being passed to the algorithms. The GSC and postfilter block depicts the main part of the implementation, and does the exact same calculations as their pendants in Matlab, only in real-time at the same rate at which the audio frames from the soundcard arrive. Even though the MCSPP postfilter method showed to be superior, all three postfilters have been implemented in C++ in order to be able to listen to the output signal they generate. The third block does the RTF estimation in a separate thread. Its execution is triggered by the speech presence probability, which is calculated by the VAD. The RTF estimation is done once every 100ms to reduce the computational load on the system.

Since the implementation can handle up to four microphone inputs, the DOA estimation has to be done up to three times in parallel. This is because the DOA is estimated between the first and all other microphones, as explained. Also, the noisefloor estimation has to be done up to five times, depending on the postfilter method being used. Therefore, the Minimum Statistics method has been implemented. Because of these multiple instances, each module used in the MCSE implementation has been defined as a class with multiple methods. All classes, their methods and their mutual dependencies have been illustrated in figure 7.3 as class diagram in UML. The red block depicts the PortAudio API, which provides a callback function named *Pa_AudioCallback()*. It serves as entry point for processing the audio frames received from and

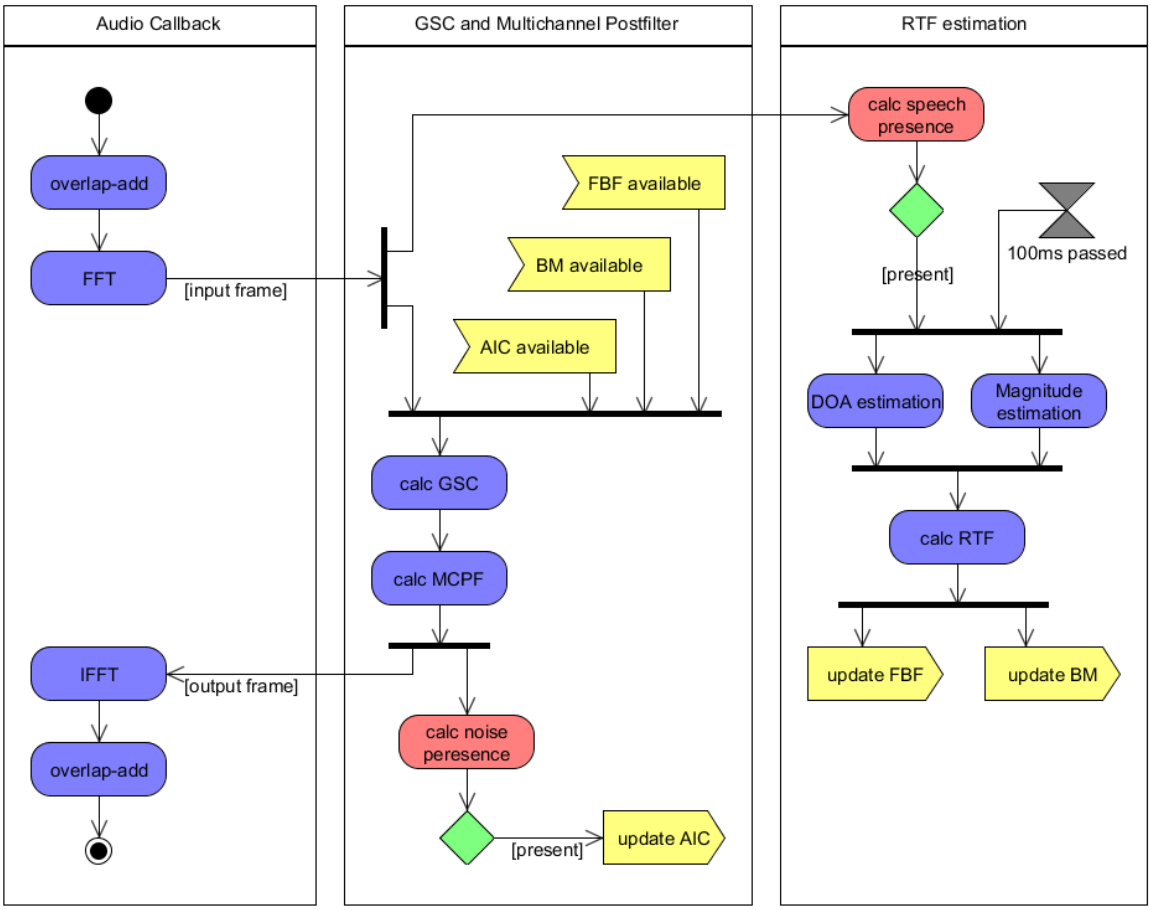

Figure 7.2: Activity diagram of the real-time MCSE implementation.

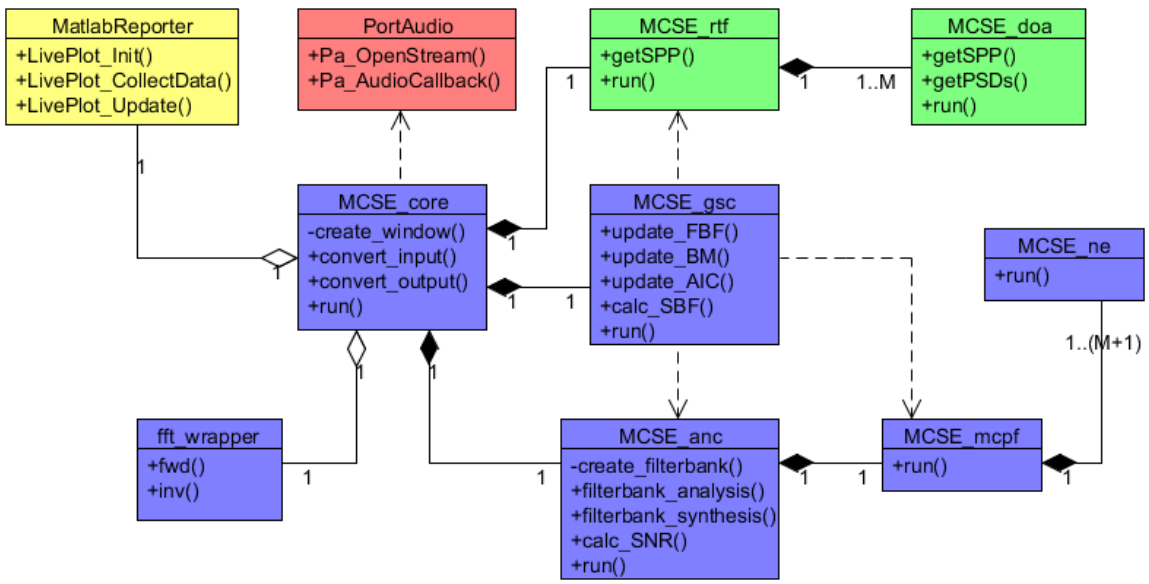

Figure 7.3: Class diagram of the real-time MCSE implementation.

sent back to the audio interface. This callback has an instance of the *MCSE_core* class, which instantiates of the following modules:

- The *MCSE_rtf* class instantiates multiple *MCSE_doa* modules, which are used to construct the $M-1$ RTFs $\tilde{A}_m(j\Omega)$.
- The *MCSE_gsc* class depends on the *MCSE_anc* and *MCSE_mcpf* instances.
- The *MCSE_anc* class subtracts the noise PSD estimate obtained by the *MCSE_mcpf* module. This module has one or more *MCSE_ne* instances, depending on the postfilter method being used.
- A FFT wrapper is used to calculate the FFT and IFFT from the audio frames, using the FFTW library.
- A *MatlabReporter* instance is used to verify the C++ implementation against the Matlab simulations from chapter 6, and for viewing information about the algorithm in realtime using the Matlab Engine.

As indicated in the activity diagram in figure 7.2, the *MCSE_rtf* and *MCSE_doa* instances run inside a thread, being executed every 100ms. All blue instances are executed at the same rate at which the audio frames are handed over by PortAudio. The timing is the same as for the Matlab experiments; frames of 32ms length are processed at a time. They overlap by 50 %, hence every 16ms a new data set is to be processed. All coding has been done using Microsoft Visual C++ 2010 Express.

7.2.2 Code verification Against the Simulation using the Matlab Engine

Verification of the C++ code is performed against the Matlab simulations from chapter 6 using the Matlab Engine ®[67]. The engine provides a modest but powerful set of functions to open a Matlab console within a C program at runtime, to exchange variables between Matlab and the C program and to execute Matlab commands and scripts from the C program. The most important functions are:

```
1  Engine *engOpen(const char *startcmd);
2  int engClose(Engine *ep);
3  mxArray *engGetVariable(Engine *ep, const char *name);
4  int engPutVariable(Engine *ep, const char *name, const mxArray *pm);
5  int engEvalString(Engine *ep, const char *string);
```

They are used by the yellow instance shown in figure 7.3, the *MatlabReporter* class. It handles the conversion from C++ floating point arrays to MEX-Structures and does all the communication with Matlab that is needed to compare each module with the Matlab simulation.

The *MCSE_core* instance shown in figure 7.3 depicts the base class of the MCSE implementation. Analogous to the Matlab code, it accepts 16ms frames of audio data as inputs. These frames can either originate from the soundcard, or from any other source. In the verification phase, the audio frames are received from Matlab via the *MatlabReporter* instance. This class communicates with Matlab using the Matlab Engine. Verification of the C++ code is thus done by receiving a frame of the microphone signals, calculating the MCSE algorithms in C++ and Matlab, sending the results from the C++ implementation back to Matlab and compare them against the simulation results. All this is done via the following code snippet:

```
1  while( ac->isRunning )
2  {
3    //pick up audio frames from matlab
4    mr->EvalString("MCSE_testbench_singlestep");
5    mr->GetRealMatrix(ac->ptrBUFz(), "BUFz");
6
7    //run the DOA estimation, GSC, MCPF and ANC modules
8    ac->run();
9
10   //send variables back to matlab for comparison
11   mr->PutCplxVector(ac->gsc->ptrFy_aic(), "Fy_aic");
12   mr->EvalString("MCSE_testbench_viewresult(GSC.Fy_aic,Fy_aic)");
13 }
```

The argument `MCSE_testbench_singlestep()` of the `EvalString()` command is a small Matlab script that invokes the Matlab simulation from the chapter 6.4. This way, the code being written for simulation can be re-used for verification of the C++ implementation without any changes. The simulation is done on a frame by frame basis. The current frame containing the four microphone signals is named `BUFz`. This block is downloaded to the C++ implementation using the command `GetRealMatrix()`, and stored into a floating point array. By calling the `run()` method of the *MCSE_core* instance, the RTF estimation, GSC beamformer and postfilter calculations are performed for that data frame, just as it is done in the Matlab simulation being executed in parallel. The results for each frame are then sent back to Matlab using the function `PutCplxVector()`. In the example above, the current frame of $Y_{AIC}(j\Omega)$ is being sent. The call to `EvalString()` invokes a small Matlab script named `MCSE_testbench_viewresult()` which compares the C++ data against the Matlab simulation.

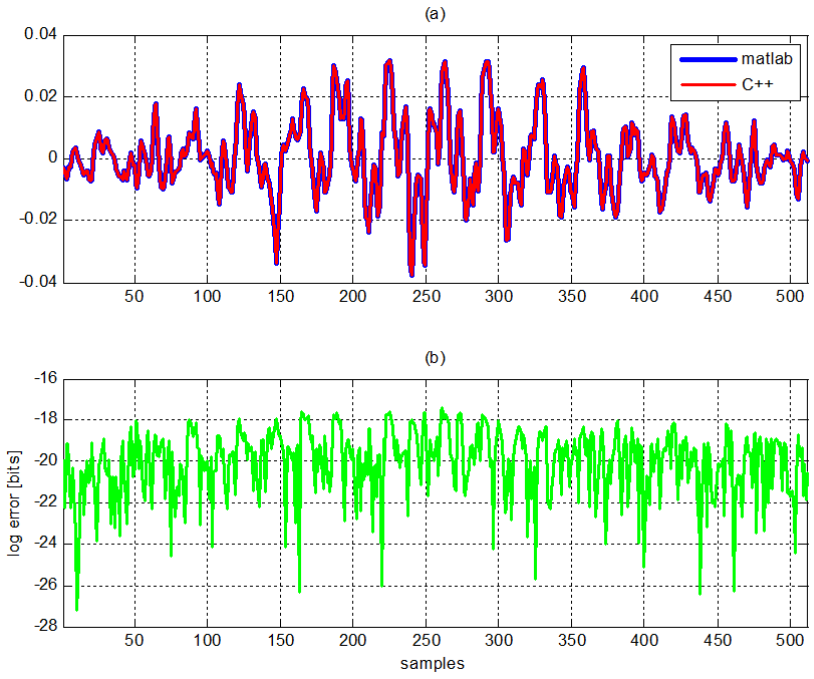

Figure 7.4: Code verification by comparing the results for $Y_{AIC}(j\Omega)$ from the C++ implementation with the Matlab simulation.

For the current frame of $Y_{AIC}(j\Omega)$, this plot is given in figure 7.4. In panel (a) the Matlab and C++ output of the IFFT of $Y_{AIC}(j\Omega)$ for the current frame are plotted on top of each other. Panel (b) shows the absolute error between the two, plotted on a binary log scale to get the error in bits. Since floating point numbers are used in the C++ implementation, the mantissa

is 23bits long. Since the ADCs in the sound interface deliver 16 bit audio data, an absolute error of 15 bits at any point in the processing chain is tolerable. This can be easily checked automatically for every signal.

Using this mechanism, an automated test was generated for each and every signal in the implementation, for a total of 60 minutes of audio data taken from a batch run from the simulation scenarios described in chapter 6.5. This way, the entire implementation can be verified with practically no effort.

7.2.3 Live Performance of the MCSE algorithm

After completion of the verification stage, the implementation is known to produce the same result as the Matlab simulation. Testing the implementation with live audio data from the soundcard is easily done by switching the audio source from the Matlab simulation data to the PortAudio driver, which provides a convenient callback function to deliver the audio frames:

```
1  //This routine will be called by PortAudio when audio is needed.
2  static int Pa_AudioCallback( const void *inputBuffer, void *outputBuffer,
       unsigned long framesPerBuffer, const PaStreamCallbackTimeInfo* timeInfo,
       PaStreamCallbackFlags statusFlags, void *userData )
3  {
4      MCSE_core *ac = (MCSE_core *)userData;
5      Map<ArrayXXf> inp((float *)inputBuffer,NIN,BLOCKSIZE);
6      Map<ArrayXXf> out((float *)outputBuffer,NOUT,BLOCKSIZE);
7
8      //read four mic channels
9      ac->ptrBUFin() = inp;
10
11     //run the DOA estimation, GSC, MCPF and ANC modules
12     ac->run();
13
14     //play result as stereo signal
15     out.row(0) = ac->ptrBUFout();
16     out.row(1) = ac->ptrBUFout();
17
18     mr->LivePlot_CollectData();
19     return paContinue;
20 }
```

The callback passes the audio frames over to the implementation, and passes the result back to the sound driver. Hence, the enhanced speech signal can be listened to via the soundcard's output channels. Also, the *MatlabReporter* instance shown in figure 7.3 is used to view some performance plots in real-time. This is done by collecting relevant data from the algorithms with the function `LivePlot_CollectData()`. This data is sent to Matlab for displaying with the following code:

```
1  ac->isRunning = true;
2  do
3  {
4    mr->LivePlot_Update();
5  }
6  while( ac->isRunning );
```

The function `LivePlot_Update()` continuously updates the *LivePlot* Matlab script. This script invokes a figure with two plots being refreshed in real-time. The figure is accompanied by Matlab-based *User Interface Control Objects* (uicontrol) which allow for simple user interactions like selecting which data set is being displayed, or selecting the number of microphones being used by the C++ implementation.

The first live scenario in figure 7.5 displays the Speech Presence Probability (SPP) between the first and the second microphone, obtained by the SCOT algorithm in the DOA estimation

module. The upper plot in figure 7.5 displays the last 10 seconds of the SPP over the possible range of impinging angles from -90°to +90°. The range of the SPP is color-coded so that blue corresponds to 0 and red corresponds to 1. The shape of the SPP was produced by uttering a constant sibilant sound while moving sideways in front of the microphone array in figure 7.1. The lower plot in figure 7.5 shows the maximum of the SPP amongst all possible angles. The user controls at the right allow to select the output signal being played by the soundcard. This way, the user can listen to the unmodified signal of the first microphone F_z, the output of the AIC Fy_{aic}, the output of the postfilter F_x, or the difference $F_n = F_z - F_x$. Also, the number of microphones being used can be selected during run-time. If one microphone is selected, the beamformer and postfilter code is bypassed and a standard single-channel speech enhancement algorithm is used instead. This algorithm uses the Minimum Statistics noisefloor estimator combined with the OM-MMSE-LSA spectral subtraction method from the simulations.

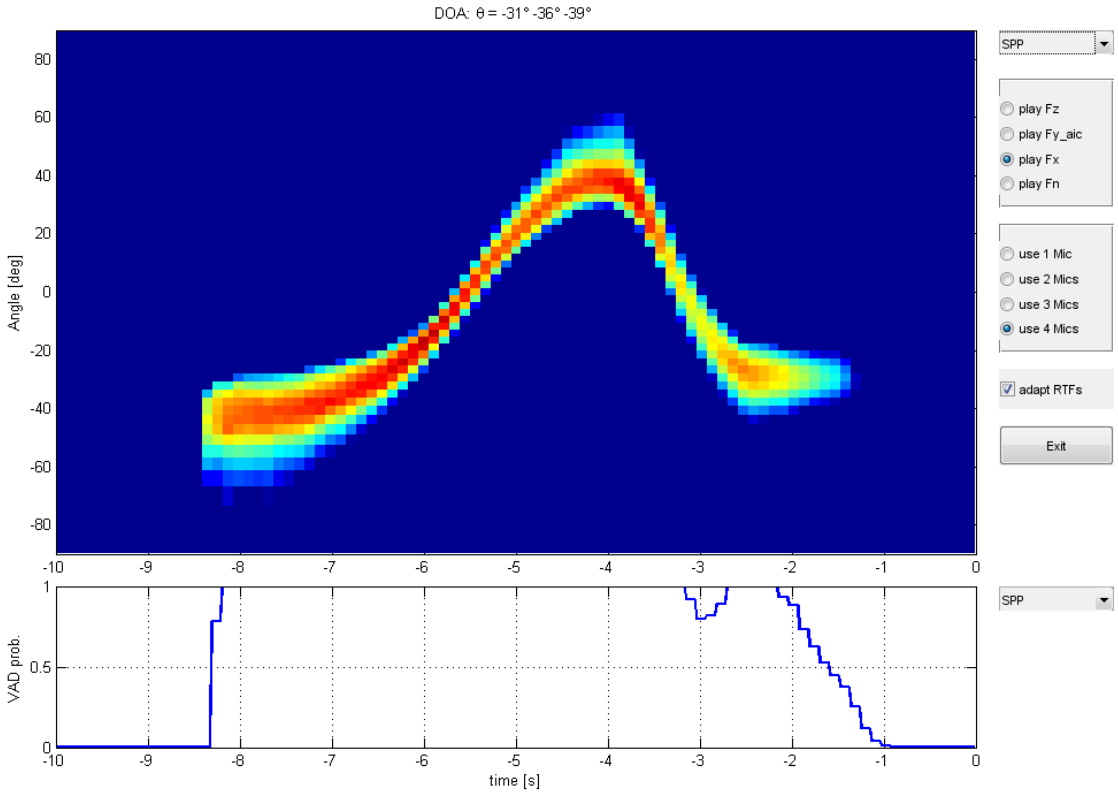

Figure 7.5: Speech Presence Probability for a speaker uttering a constant sibilant sound while moving sideways in front of the microphone array.

The next scenario in figure 7.6 shows the same plot, but this time music is played from two speakers located approximately 2m to the left of the microphone array in a standard office room. It can be seen that there is no distinct maximum in the SPP plot. In fact, the highest values do not exceed 0.5. Caused by the reverberations of the room, there is almost no direct sound field anymore. Thus there is no clear direction that can be identified. This observation confirms the initial assumption of direct and diffuse sound fields for the speaker and the noise, respectively.

As outlined in the Activity Diagram in figure 7.2, the RTFs are estimated once every 100ms. If the VAD detects a directional sound over a significant frequency range, the DOA information is updated. Each microphone estimates its own DOA with respect to the first one. Strictly speaking, the DOA is no direction, but rather a time delay between impinging sound waves at the first and at the m-th microphone. Hence, the possible location of the sound source lies on a hyperbola. The asymptote of this hyperbola depicts the actual DOA. Figure 7.7 shows these

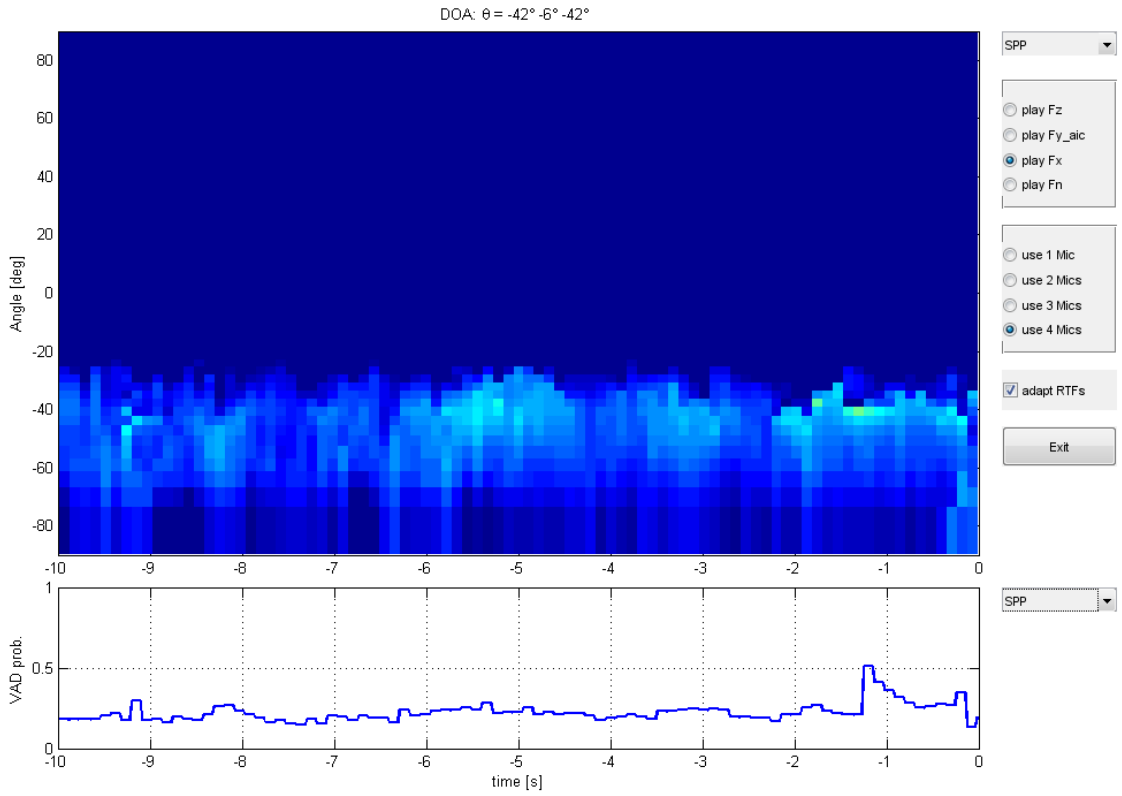

Figure 7.6: Speech Presence Probability for a set of speakers located 2m to the left of the microphone array.

hyperbolas for $m = 4$ microphones. Their intersect point is the most probable location of the sound source. According to the figure, the speaker was located approx. 35cm in front of the array. At the very top of the figure, the DOA angles are given as numeric values. The plot located at the bottom shows the SBF from equation 6.1. Since the true speech signal $S_m(j\Omega)$ is not available, the following approximation using the enhanced signal at the postfilter output $X(j\Omega)$, and the RTF estimate $\hat{\tilde{A}}_m(j\Omega)$ is used instead.

$$\hat{\text{SBF}} = 10\log_{10}\frac{\mathbb{E}\left\{\sum_{m=1}^{M}|Z_m(j\Omega)|^2\right\}}{\mathbb{E}\left\{\sum_{m=1}^{M}|Z_m(j\Omega)-\hat{\tilde{A}}_m(j\Omega)X(j\Omega)|^2\right\}}, \tag{7.1}$$

The peak in the SBF of figure 7.7 is caused by a short utterance to trigger the VAD.

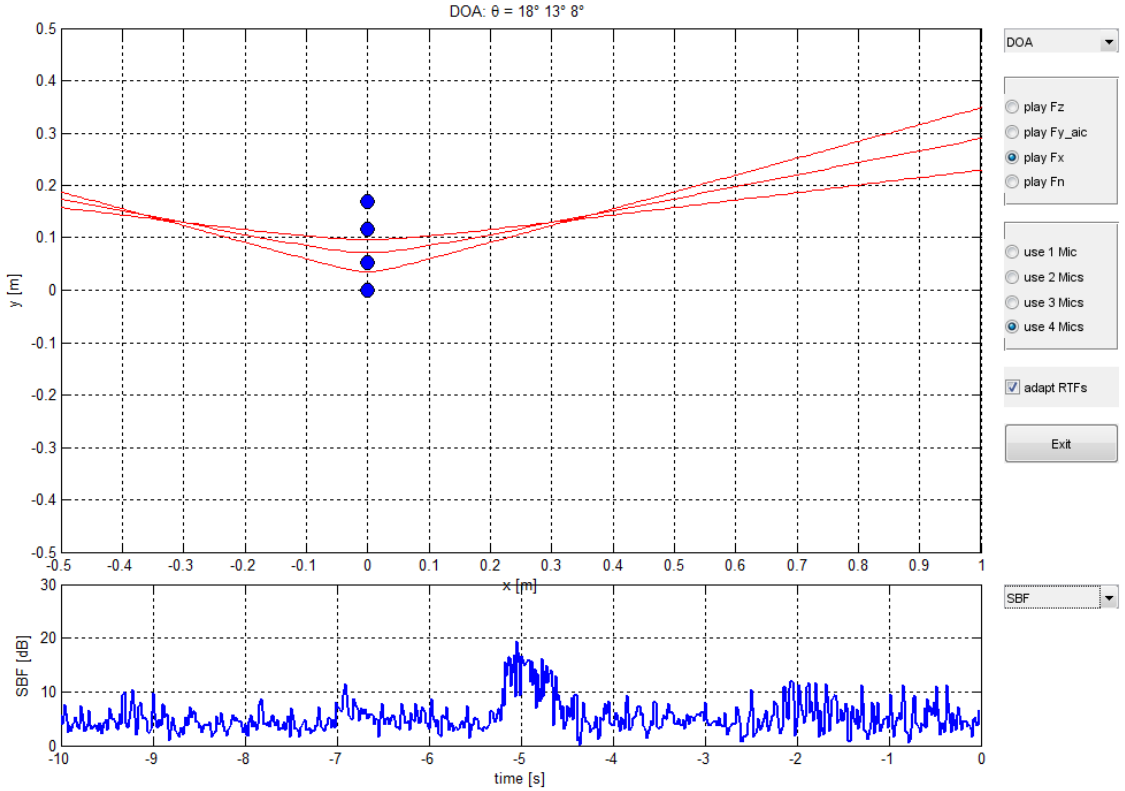

Figure 7.7: DOA hyperbolas drawn for the four-element microphone array with $d = 5cm$ and impinging angles $\Theta_2 = 18°$, $\Theta_3 = 13°$, $\Theta_4 = 8°$. The intersect point is the actual location of the speaker. The lower plot shows the Signal Blocking Factor.

The next live scenario involves speaking into the array from a distance of 30cm, while the pair of loudspeakers located 2m to the left of the microphone array is playing music. The upper plot in figure 7.8 shows the spectrogram of the first microphone signal $Z_1(j\Omega)$. The utterance is *"das ist ein test - eins - zwei - drei"*. The lower plot shows the SBF. The fact that the SBF rises to about 15dB indicates that the blocking matrix successfully suppresses the speech signal. As pointed out, this is a mandatory condition for the GSC to successfully suppress the noise while leaving the speech signal undistorted.

Figure 7.9 shows the spectrogram of the enhanced signal $X(j\Omega)$ at the output of the multichannel postfilter. From the SNR plotted below, it can be seen that the background noise is reduced by approximately 20dB. Furthermore, the amount of noise reduction is equal over the entire frequency range. This is one reason why the MCSPP postfilter received the highest PEASS score. The other postfilters showed low performance towards lower frequencies. One reason for this behavior is the high correlation of the noise signal at low frequencies. The speech pattern in figure 7.8 and 7.9 is almost identical, indicating that the speech signal is preserved during the entire processing chain. If the user moves further away from the microphone array, the sound waves from the speech signal turn from a direct sound field into a diffuse sound field. Then, the speech signal starts getting attenuated for distances greater than approximately 50cm. Interestingly, the speech does not get distorted in any way, it sounds more distant and damped until it subsides into the background noise.

Figure 7.10 shows the difference between the signal at the reference microphone and the output of the multichannel postfilter $\hat{N}(j\Omega) = Z_1(j\Omega) - X(j\Omega)$. This differential signal contains only the noise, and eventually some of the desired speech signal leaking through the blocking matrix.

The less speech this differential signal contains, the higher the quality of the speech estimate at the postfilter output. This is also the reason why the user can chose to listen to this signal with the radio button labeled *"play Fn"*. The human auditory system perceives even the faintest fragments of speech, which cannot be recognized by looking at the spectrogram. For example, the typical pattern of speech is almost invisible in the spectrogram of $\hat{N}(j\Omega)$. However, listening to this signal reveals very faint and distant speech. This is due to the fact that the beamformer extracts only the direct sound, which is the speech impinging at the array via the direct path. The speech that arrives as reverberation at the array is part of the diffuse sound field, and hence audible in the differential signal. The fact that the speech in the differential signal is so faint indicates that the prototype achieves a high speech quality and noise reduction at the same time.

Figure 7.11 shows the real-valued gain function $G(j\Omega)$, which is calculated by the multichannel postfilter to obtain the enhanced signal $X(j\Omega)$. The postfilter operates on a filterbank with 48 bands, like the simulation. The upper figure in this plot shows the gain over time and frequency, where blue corresponds to a value of 0 and red to a value of 1. It can be seen that there is no musical noise or other artifacts produced by the postfilter. Also, from listening to the signal one may find the speech quality to be quite high and without noticeable distortions. In contrast to most single channel speech enhancement methods, both the attenuation of the background noise and the speech quality are good at the same time. The lower plot shows the SNR again.

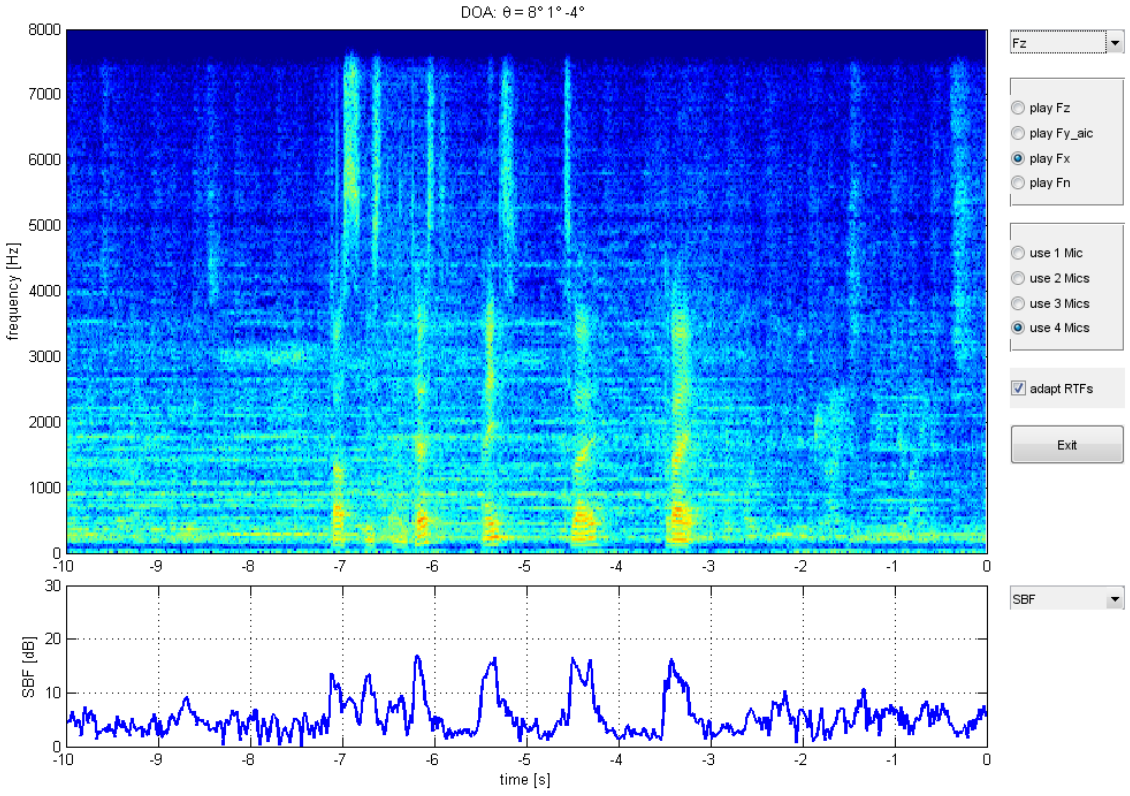

Figure 7.8: Spectrogram of the first microphone signal $Z_1(j\Omega)$. The lower plot shows the Signal Blocking Factor.

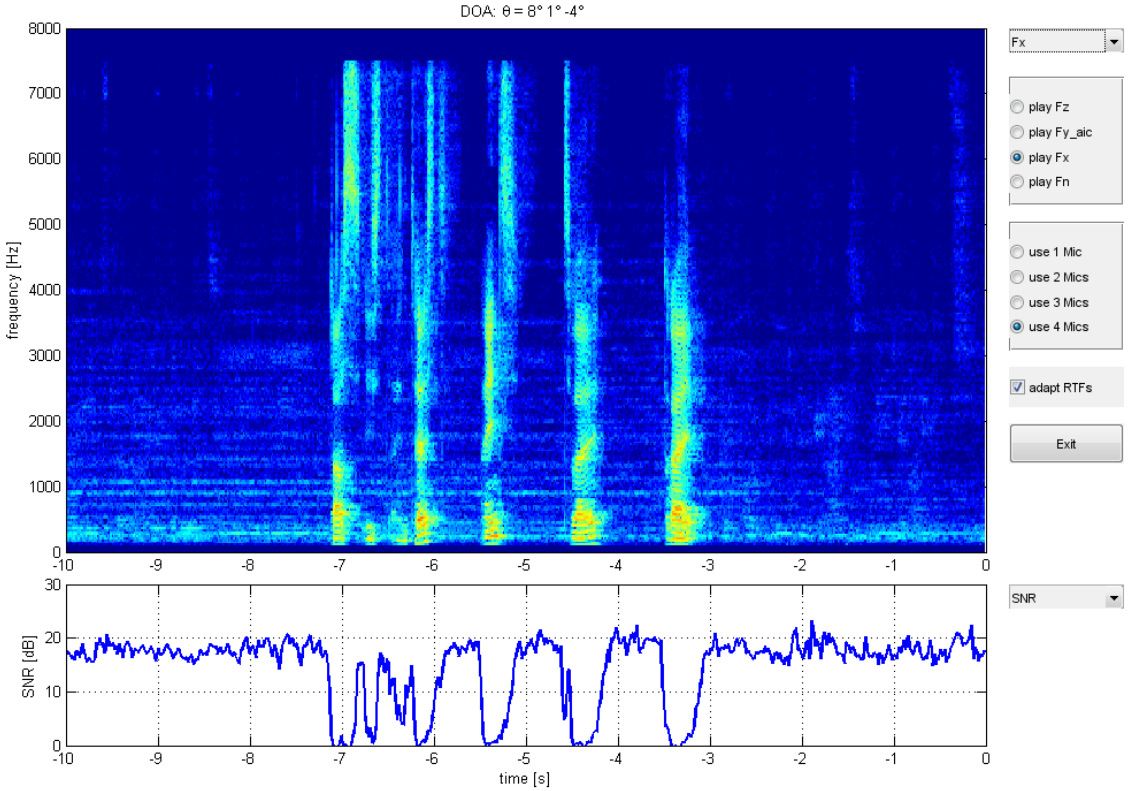

Figure 7.9: Spectrogram of the output of the multichannel postfilter $X(j\Omega)$. The lower plot shows the Signal-to-Noise Ratio.

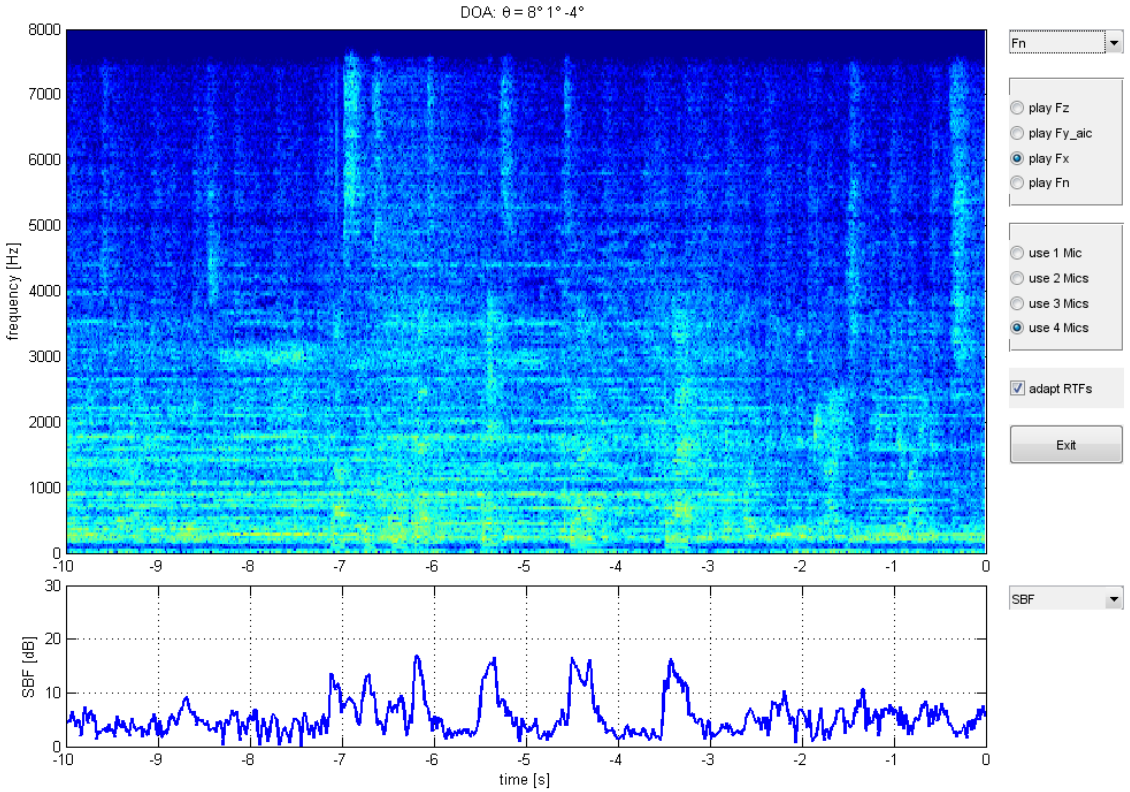

Figure 7.10: Spectrogram of the difference signal $\hat{N}(j\Omega) = Z_1(j\Omega) - X(j\Omega)$. The lower plot shows the Signal Blocking Factor.

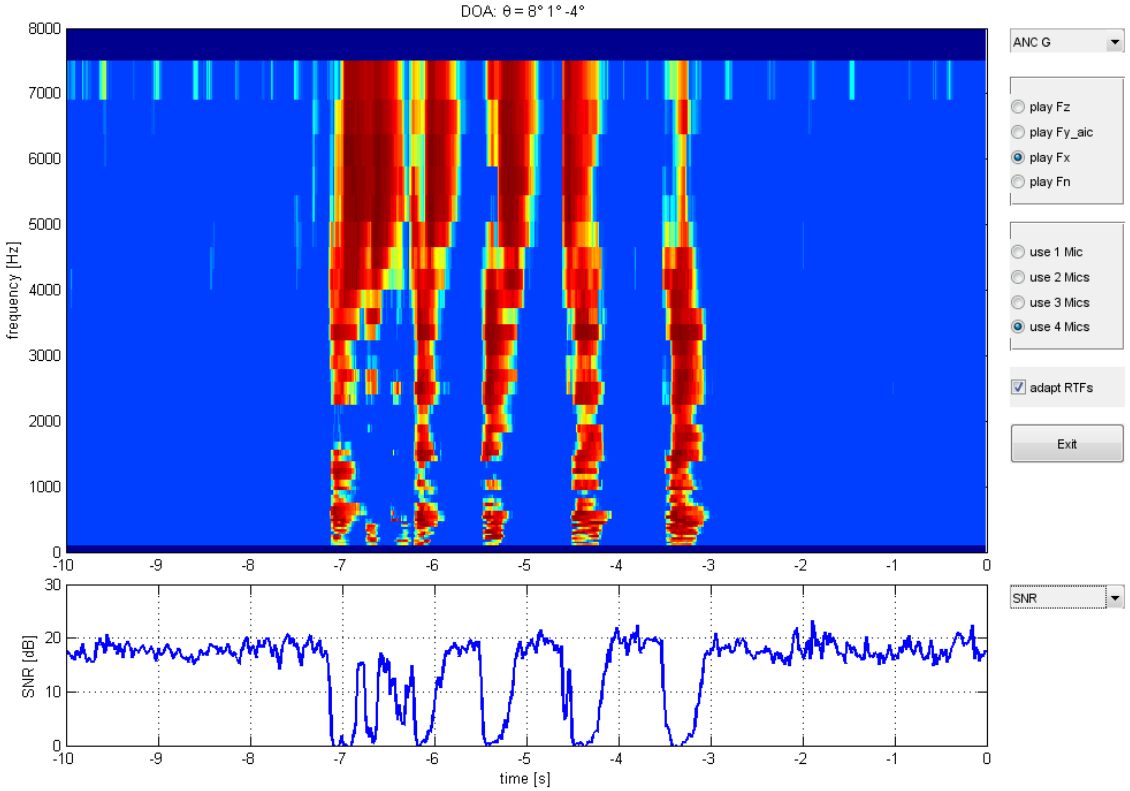

Figure 7.11: Gain function $G(j\Omega)$ of the postfilter drawn over time and frequency. The lower plot shows the Signal-to-Noise Ratio.

7.3 Porting to an embedded platform

The final objective of this multichannel speech enhancement project is to port the prototype implementation to an embedded platform. While this process is not part of the thesis anymore, the most important steps in doing so are sketched in the following.

The technology roadmap of the company Commend includes a new series of intercom devices using state of the art embedded systems. Today, such systems are affordable even on a low budget, and therefore the prototype should be able to run on such a device. For example, full-featured evaluation Kits including the ARM Cortex-A9 MCU, Ethernet and USB ports as well as an on-board sound chip are already available for $40. Thanks to well-established standards in modern software development, all of the libraries and frameworks used for the prototype can also be compiled for the ARM target. For example, the FFTW library supports the NEON FPU of the ARM Corex devices. The NEON feature allows four FPU operations to be executed in parallel, thereby yielding a fourfold floating-point performance. On the WandBoard, which has an ARM Cortex-A9 CPU being clocked at 800MHz, FFTW achieves the following performance when executing a one dimensional 512 point FFT and IFFT:

```
Problem: orf512, setup: 15.77 s, time: 16.93 us, ''mflops'': 680.54
Problem: orb512, setup: 16.31 s, time: 19.19 us, ''mflops'': 600.24
```

When using four microphones, four FFTs and one IFFT are required for each 16ms frame. Hence, FFTW achieves the 184-fold performance required. Also, the Eigen framework has built-in NEON support. This makes cross-compilation of the prototype fairly easy. The following CFLAGS have been used with GCC 4.6.3:

```
CFLAGS="-O3 -mfpu=neon -ftree-vectorize -mfloat-abi=softfp -ffast-math
    -funsafe-math-optimizations -fsingle-precision-constant"
```

A small benchmark program has been written, which uses a multichannel WAV-file instead of the soundcard as input. This way, the performance of the code can be evaluated by comparing its execution time against the duration of the WAV-file. Executing the benchmark on the BeagleBoard XM gives the following output for 2 microphones:

```
Reading file: MCSE_test_2ch.wav
Fs = 16000Hz
channels = 2
samples = 240000 per channel
duration = 15s
frame size = 256 samples
Using 937 frames for simulation

created MCSE_core with fs=16000, BlockLength=0.016, Nmics=2
created plan for a 512 point DFT

CPU time = 3.17 seconds.
CPU load = 21.1333%.
```

And with 4 microphones:

```
Reading file: MCSE_test_4ch.wav
Fs = 16000Hz
channels = 4
samples = 240000 per channel
duration = 15s
frame size = 256 samples
Using 937 frames for simulation

created MCSE_core with fs=16000, BlockLength=0.016, Nmics=4
created plan for a 512 point DFT

CPU time = 5.27 seconds.
CPU load = 35.1333%.
```

Without any further optimization, the code achieves almost three times the required performance when using four microphones. Although, when considering real-time operation of the code, some delays caused by the audio driver (ALSA) are to be considered. Also, the surrounding framework for audio compression and transmission to another client has to be considered. And finally, for the sake of stability of the entire embedded system, it is a good idea to keep the total CPU load below 50%. Therefore it might be advantageous to optimize the code by hand a bit further, though.

8 Conclusion and Future Work

In this thesis, an in-depth overview of state of the art *multichannel speech enhancement* (MCSE) systems based on acoustic beamforming and multichannel postfilters has been given. The first part of the work focused on the mathematical framework of superdirective beamformers with special attention to the GSC beamformer, followed by four different blocking matrices. Then, five algorithms for source locating have been characterized, where it was shown that estimating the DOA alone is more robust than estimating the entire RTF towards the speech source. Also, three different multichannel postfilter algorithms have been introduced.

Part two of this thesis focused on the selection of a subset of these algorithms used for implementation. The selection process was based upon the psychoacoustic motivated performance measures PESQ and PEASS, but also on implementation issues like real-time capability and of course numerical complexity. For this selection process, an extensive set of Matlab experiments using the TIMIT, KCOSS and KCORS speech databases has been created. Based on the outcome of these experiments, the following algorithms are used for the prototype implementation:

Due to its robustness and performance, the GSC has been chosen as beamformer. The speaker-dependent RTFs are modeled as simple time delays because the speaker is always close to the array. Therefore, a simple DOA estimation using the SCOT algorithm is sufficient for this purpose. Further, the magnitude estimation from section 4.3.4 is used to account for the amplitude differences between the microphone signals. The AIC filters in the GSC are adapted by the NLMS algorithm. As multichannel postfilter, the MCSPP is used for both its simplicity and outstanding speech quality. Finally, the postfilter is combined with the simplified filterbank from chapter 5.4.2 using 48 bands. The filterbank reduces the computational cost in the postfilter by a factor of 5 while maintaining its performance.

The last part of this work, is devoted to the real-time implementation of the Matlab-code, using the optimal combination of algorithms identified in part two. Here, implementation-related issues have been documented to provide an insight into the basic concepts of rapid prototyping and modern software development. Finally, the thesis concludes with a performance demonstration of the prototype running on a standard PC. The achieved performance approves the initial assumption of the diffuse noise sound field and the direct sound field for speech. Even in a standard office room, the background noise is almost an ideal diffuse sound field, while the speech signal is mostly directional when the user is speaking from a distance shorter than 50cm

into the array. The initial statement, that multichannel speech enhancement is superior to the well-established single channel methods has been verified in both theory and practice.

Further steps at the company Commend include the casting of the prototype into a product. Therefore, the Austrian Research Promotion Agency extended this research project to a second year. Within this time, the following steps should be accomplished:

- Continue research on the Direct to Diffuse Ratio (DDR) from chapter 5.3.2. The algorithm is a promising candidate for a robust DOA estimator that does not need a VAD.
- Continue research on the Multichannel Speech Presence Probability (MCSPP) from chapter 5.3.3. The modifications made so far can further be optimized to increase the performance in low SNR conditions.
- Incorporate an *Acoustic Echo Canceler* (AEC) into the MCSE system, so that it can be used for bidirectional communication. In [68], three basic concepts are mentioned: A cascade of a multichannel AEC and the GSC, the GSC followed by a single-channel AEC, and a merged AEC and GSC design. All three concepts have pros and cons regarding performance and numerical complexity. Thus, the integration of an AEC into the prototype is a future research topic.
- Incorporate Digital Audio Watermarking techniques into the MCSE system. Research on this topic has already been concluded and allows for numerous applications in VoIP telephony.
- Port the implementation to an embedded platform using a state of the art MCU, like the ARM Cortex-A9.
- Select a suited embedded platform, microphones, loudspeakers and a housing to build a stand-alone prototype.
- Use an audio server like the Jack Audio Connection Kit to use the MCSE system in conjunction with standard VoIP software.
- Merge the business strategies of the company with the new capabilities of this device, and develop a roadmap for its applications.

9 Listings

In this chapter, the most relevant code snippets from the Matlab experiments and the prototype implementation are presented for documentation:

Noise Samples used in the simulations:

```
[01] "Noise Samples\Abfüllanlage2_4ch.wav"
[02] "Noise Samples\Abfüllanlage3_4ch.wav"
[03] "Noise Samples\Abfüllanlage_4ch.wav"
[04] "Noise Samples\Abriss3_4ch.wav"
[05] "Noise Samples\Abriss_4ch.wav"
[06] "Noise Samples\Autobahn_4ch.wav"
[07] "Noise Samples\Bagger_4ch.wav"
[08] "Noise Samples\Barcelona airport gate 39_3_4ch.wav"
[09] "Noise Samples\Barcelona airport gate 39_4ch.wav"
[10] "Noise Samples\Boiler_4ch.wav"
[11] "Noise Samples\Crossing Hankyu_4ch.wav"
[12] "Noise Samples\Cygnus X - The Orange Theme3_4ch.wav"
[13] "Noise Samples\Destroyer Operations Room_4ch.wav"
[14] "Noise Samples\Diffuse_speech1_4ch.wav"
[15] "Noise Samples\Diffuse_speech2_4ch.wav"
[16] "Noise Samples\Diffuse_speech3_4ch.wav"
[17] "Noise Samples\Diffuse_speech4_4ch.wav"
[18] "Noise Samples\Diffuse_speech5_4ch.wav"
[19] "Noise Samples\Diffuse_speech6_4ch.wav"
[20] "Noise Samples\Factory3_4ch.wav"
[21] "Noise Samples\Factory_4ch.wav"
[22] "Noise Samples\Fluglärm_4ch.wav"
[23] "Noise Samples\Güterzüge_4ch.wav"
[24] "Noise Samples\Kiev train station_4ch.wav"
[25] "Noise Samples\Kompressor_4ch.wav"
[26] "Noise Samples\London Tube3_4ch.wav"
[27] "Noise Samples\London Tube_4ch.wav"
[28] "Noise Samples\Manhattan_4ch.wav"
[29] "Noise Samples\Narita Airport_4ch.wav"
[30] "Noise Samples\New York City rescue van on Times Square3_4ch.wav"
[31] "Noise Samples\Pendulum - Blood Sugar3_4ch.wav"
[32] "Noise Samples\Pendulum - Tarantula3_4ch.wav"
[33] "Noise Samples\Penn station newark_4ch.wav"
[34] "Noise Samples\Roadnoise2_4ch.wav"
[35] "Noise Samples\Roadnoise3_4ch.wav"
[36] "Noise Samples\Roadnoise_4ch.wav"
[37] "Noise Samples\Schiffsmotor_4ch.wav"
[38] "Noise Samples\Shinkansen2_4ch.wav"
[39] "Noise Samples\Shinkansen3_4ch.wav"
[40] "Noise Samples\Shinkansen_4ch.wav"
[41] "Noise Samples\Stadtverkehr2_4ch.wav"
```

```
[42] "Noise Samples\Stadtverkehr3_4ch.wav"
[43] "Noise Samples\Stadtverkehr_4ch.wav"
[44] "Noise Samples\Stahlwerk_4ch.wav"
[45] "Noise Samples\Staubsauger_4ch.wav"
[46] "Noise Samples\Toronto Hilton lounge3_4ch.wav"
[47] "Noise Samples\Toronto Hilton lounge_4ch.wav"
[48] "Noise Samples\Traffic Chaos_4ch.wav"
[49] "Noise Samples\Traffic Noise in India3_4ch.wav"
[50] "Noise Samples\Traffic Noise in India_4ch.wav"
[51] "Noise Samples\Tsurumai_4ch.wav"
[52] "Noise Samples\Vintage Cars_4ch.wav"
[53] "Noise Samples\Yamanote Line Shinjuku3_4ch.wav"
[54] "Noise Samples\Yamanote Line Shinjuku_4ch.wav"
[55] "Noise Samples\lg-jfk-airport_4ch.wav"
[56] "Noise Samples\tottenhamcourtroad_4ch.wav"
```

Calculation of the directivity pattern in Matlab:

```
%calc directivity pattern
for j=1:Nangles
    x = micpos(:,1) - micpos(1,1);          %mic 1 = reference mic
    y = micpos(:,2) - micpos(1,2);
    d = sqrt(x.^2+y.^2);                    %distance mic(1) to all others
    a = atan2(y,x);
    tau = d.*cos(a-avect(j)*pi/180)./c;
    for k=1:Nbins
        A = exp(-sqrt(-1)*tau*2*pi*fvect(k));
        Psi(j,k) = W(:,k)'*A;
    end
end

HH = 20*log10(abs(Psi));
HH = min(max(HH,-20),0);
```

Calculation of the directivity index in Matlab:

```
%calc directivity index
DI = zeros(1,Nbins);
for i=1:Nmics
    for j=1:Nmics
        DI = DI + W(i,:).*conj(W(j,:)).*squeeze(C_nn(i,j,:)).';
    end
end
for k=1:Nbins
    DI(k) = abs(W(:,k)'*As(:,k)).^2 ./ max(real(DI(k)), 1e-6);
end
```

Calculation of the coherence of an ideal diffuse sound field in Matlab:

```
%calc coherence for an ideal diffuse sound field
C_nn = zeros(Nmics,Nmics,Nbins);
for i=1:Nmics
    for j=1:Nmics
        x = micpos(i,1) - micpos(j,1);
        y = micpos(i,2) - micpos(j,2);
        md = sqrt(x.^2+y.^2);
        tau = md/c;
        C_nn(i,j,:) = sinc(2*fvect*tau);
    end
end
```

Calculation of the MVDR filter weights for the directional sound field in figure 3.7:

```
%calc Phi_nn for a directional sound field
theta_n = 20;
for k=1:Nbins
    x = micpos(:,1) - micpos(1,1);          %mic 1 = reference mic
    y = micpos(:,2) - micpos(1,2);
    d = sqrt(x.^2+y.^2);                    %distance mic(1) to all other mics
```

```
7       a = atan2(y,x);
8       tau = d.*cos(a-theta_n*pi/180)/c;
9       An(:,k) = exp(-sqrt(-1)*tau*2*pi*fvect(k));
10      Phi_nn(:,:,k) = An(:,k)*An(:,k)';
11  end
12
13  %calc MVDR filter weights W
14  for k=1:Nbins
15      ni = inv(Phi_nn(:,:,k) + I*1e-6);
16      W(:,k) = ni*As(:,k) / (As(:,k)'*ni*As(:,k));
17  end
```

Calculation of the GSC beamformer in Matlab:

```
1   for k=1:AC.Nbins
2       %fixed beamformer FBF
3       GSC.F(:,k) = RTF.As(:,k) / norm(RTF.As(:,k))^2;
4
5       %apply FBF
6       GSC.Fy_fbf(k) = GSC.F(:,k)'*AC.Fz(:,k);
7
8       %apply BM
9       GSC.Fu(:,k) = GSC.B(:,:,k)'*AC.Fz(:,k);
10
11      %apply AIC filters
12      GSC.Fy_aic(k) = GSC.H(:,k)'*GSC.Fu(:,k);
13
14      %calc GSC output
15      GSC.Fy(k) = GSC.Fy_fbf(k) - GSC.Fy_aic(k);
16
17      %calc overall GSC filter weights
18      GSC.W(:,k) = GSC.F(:,k) - GSC.B(:,:,k)*GSC.H(:,k);
19
20      %AIC update as per NLMS
21      GSC.Szz(k) = GSC.Szz(k)*GSC.alpha + (1-GSC.alpha)*norm(AC.Fz(:,k))^2;
22      GSC.H(:,k) = GSC.H(:,k) + 0.1*GSC.Fu(:,k)*conj(GSC.Fy(k))/max(GSC.Szz(k)
            ,0.1);
23  end
```

SCOT Algorithm for estimating the DOA in Matlab:

```
1   %size of the DOA search range
2   DOA.Nangles = 50;
3   %provide search range tau in [s] for the distance from mic 1 to mic <DOA.m>
4   DOA.tau_range = linspace(-AC.mic_dist(DOA.m),AC.mic_dist(DOA.m),DOA.Nangles) /
        AC.c;
5   %calc corresponding theta range
6   DOA.theta_range = asin(DOA.tau_range*AC.c/(AC.mic_dist(DOA.m) + 1e-6));
7
8   DOA.psi = zeros(1,AC.Nbins);
9   %max search range for tau in integer samples
10  DOA.samples_max = ceil(max(DOA.tau_range*AC.fs));
11  %tau search range in integer samples
12  DOA.samples_index = (-DOA.samples_max:DOA.samples_max)';
13
14  %recursive cross-PSD estimation
15  for k=1:AC.Nbins
16      RTF.Phi_zz(:,:,k) = RTF.Phi_zz(:,:,k)*RTF.alpha +
            (1-RTF.alpha)*AC.Fz(:,k)*AC.Fz(:,k)';
17  end
18
19  %get PSDs from the RTF module
20  Szmz1 = squeeze(RTF.Phi_zz(DOA.m,1,:))';
21  Szmzm = real(squeeze(RTF.Phi_zz(DOA.m,DOA.m,:)))';
22  Sz1z1 = real(squeeze(RTF.Phi_zz(1,1,:)))';
23
24  %SCOT function
25  DOA.psi = Szmz1./(sqrt(Szmzm.*Sz1z1) + 1e-6);
26
27  r = rifft(DOA.psi)';
28  %crop r to range -samples_max...+samples_max
```

```
r = circshift(r,[DOA.samples_max 0]);
r = r(1:DOA.samples_max*2+1);

%search tau via MSE
for a=1:DOA.Nangles
    r_ref = sinc(DOA.samples_index+DOA.tau_range(a)*AC.fs);
    DOA.J(a) = 1./max( mean((r-r_ref).^2) , 1e-6);
end
[val DOA.index] = max(DOA.J);

%get tau and theta
tau = DOA.tau_range(DOA.index);
theta = DOA.theta_range(DOA.index);
```

MUSIC Algorithm for estimating the DOA in Matlab:

```
%narrowband MUSIC as in Microphone Array Signal Processing Jacob Benesty 2008
DOA.J = zeros(DOA.Nangles,1);
for a=1:DOA.Nangles
    for k=1:AC.Nbins
        Phi = [Sz1z1(k), Szmz1(k)'; Szmz1(k), Szmzm(k)];
        [V,D] = eig(Phi);
        [val index] = min(diag(D));
        %eigenvector(2)'*sigma
        DOA.J(a) = DOA.J(a) + 1/abs(V(:,index)'*[1; DOA.sigma(a,k)]).^2;
    end
end
[val DOA.index] = max(DOA.J);
```

TBRR Postfilter in Matlab:

```
%calc noisefloors
MCPF.NE{1} = calc_NE(AC,MCPF.NE{1},Fs_hat);
for m=1:AC.Nmics
    MCPF.NE{m+1} = calc_NE(AC,MCPF.NE{m+1},MCPF.Fn(m,:));
end

%calc TBRR
tmp1 = max( MCPF.NE{1}.S - MCPF.NE{1}.lambda , 0);
tmp2 = zeros(1,AC.Nbands);
for m=1:AC.Nmics;
    tmp2 = max( MCPF.NE{m+1}.S - MCPF.NE{m+1}.lambda , tmp2);
end
MCPF.psi = tmp1./(tmp2 + 1e-6);

%speech absence probability
index = (MCPF.psi < MCPF.psi_low);
tmp2 = (MCPF.psi_high - MCPF.psi) / (MCPF.psi_high-MCPF.psi_low);
MCPF.q = max(tmp2,0);
MCPF.q(index) = 1;

%noisefloor estimation
MCPF.lambda = MCPF.Y .* MCPF.q;
```

MCSPP Postfilter in Matlab

```
for k=1:AC.Nbins

    AW = RTF.As(:,k)*GSC.W(:,k)';
    I = eye(AC.Nmics);
    Cnn = MCPF.Cnn(:,:,k);
    tmps0 = real(trace( AW*Cnn*AW' ));
    tmpn0 = real(trace( (I-AW)*Cnn*(I-AW)' ));
    MCPF.Phi_ss0(k) = MCPF.Phi_ss0(k)*MCPF.alpha0 + (1-MCPF.alpha0)*tmps0;
    MCPF.Phi_nn0(k) = MCPF.Phi_nn0(k)*MCPF.alpha0 + (1-MCPF.alpha0)*tmpn0;

    tmps = sum(abs(MCPF.Fs(:,k)).^2);
    tmpn = sum(abs(MCPF.Fn(:,k)).^2);
    MCPF.Phi_ss(k) = MCPF.Phi_ss(k)*MCPF.alpha + (1-MCPF.alpha)*tmps;
    MCPF.Phi_nn(k) = MCPF.Phi_nn(k)*MCPF.alpha + (1-MCPF.alpha)*tmpn;
```

```
15
16 end
17
18 tmps = MCPF.Phi_ss.*MCPF.Phi_nn0;
19 tmpn = MCPF.Phi_nn.*MCPF.Phi_ss0;
20 gamma = tmps ./ (tmpn + 1e-6);
21
22 MCPF.zeta = max(gamma-1,0);
23 MCPF.q = 1./(1+MCPF.zeta);
24 MCPF.lambda = MCPF.Y.*MCPF.q;
```

DDR Postfilter in Matlab:

```
1 MCPF.Sz1z1 = MCPF.Sz1z1*MCPF.alpha + (1-MCPF.alpha)* abs(AC.Fz(1,:)).^2;
2 MCPF.Sz2z2 = MCPF.Sz2z2*MCPF.alpha + (1-MCPF.alpha)* abs(AC.Fz(2,:)).^2;
3 MCPF.Sz1z2 = MCPF.Sz1z2*MCPF.alpha + (1-MCPF.alpha)*
      AC.Fz(1,:).*conj(AC.Fz(2,:));
4 MCPF.Cz1z2 = MCPF.Sz1z2 ./ sqrt(MCPF.Sz1z1.*MCPF.Sz2z2 + 1e-6);
5
6 tau = AC.mic_dist(2)/AC.c;
7 MCPF.Cn1n2 = sinc(2*AC.fvect*tau);
8
9 sigma = conj(RTF.As(2,:))./abs(RTF.As(2,:));
10 ddr = (MCPF.Cn1n2 - MCPF.Cz1z2) ./ (MCPF.Cz1z2 - sigma);
11 ddr = max(real(ddr),0);
12 q = 1./(1+ddr);
13
14 %speech absence probability
15 MCPF.q = filterbank_analysis(AC, q );
16
17 %noisefloor estimation
18 MCPF.lambda = MCPF.Y .* MCPF.q;
19 MCPF.NE = calc_NE(AC,MCPF.NE, Fs_hat);
20 MCPF.lambda = max(MCPF.lambda,MCPF.NE.lambda);
```

OM-MMSE-LSA spectral subtraction algorithm in Matlab:

```
1     %input PSD
2     ANC.Y = filterbank_analysis(AC, abs(Fy).^2 );
3     ANC.lambda = filterbank_analysis(AC, lambda );
4
5     ANC.gamma = ANC.Y./(ANC.lambda + 1e-6);
6     ANC.zeta = ANC.zeta*ANC.alpha_zeta + (1-ANC.alpha_zeta)*max(ANC.gamma-1,0);
7
8     %zeta smoothing, locally
9     ANC.zeta_local = 10*log10(rolling_mean(ANC.zeta,ANC.b_local));
10    ANC.P_local = (ANC.zeta_local-ANC.zeta_lo)./(ANC.zeta_hi-ANC.zeta_lo);
11    ANC.P_local(ANC.zeta_local < ANC.zeta_lo) = 0;
12    ANC.P_local(ANC.zeta_local > ANC.zeta_hi) = 1;
13
14    %zeta smoothing, globally
15    ANC.zeta_global = 10*log10(rolling_mean(ANC.zeta,ANC.b_global));
16    ANC.P_global = (ANC.zeta_global-ANC.zeta_lo)./(ANC.zeta_hi-ANC.zeta_lo);
17    ANC.P_global(ANC.zeta_global < ANC.zeta_lo) = 0;
18    ANC.P_global(ANC.zeta_global > ANC.zeta_hi) = 1;
19
20    %speech absence probability
21    ANC.q = 1-ANC.P_global.*ANC.P_local;
22
23    %speech presence probability
24    ANC.p = (1 + (ANC.q./(1-ANC.q)) .* (1+ANC.xi) .* exp(-ANC.nu) ).^-1;
25
26    %decision directed a-priori SNR estimation
27    ANC.xi = ANC.xi_old.*ANC.alpha + (1-ANC.alpha).*max(ANC.gamma-1,0);
28
29    %MMSE-LSA
30    tmp = ANC.xi./(1+ANC.xi);
31    ANC.nu = tmp.*ANC.gamma + 1e-6;
32    ANC.G0 = tmp.*exp(0.5*expint(ANC.nu));
33    ANC.xi_old = ANC.G0.^2.*ANC.gamma;
34
```

```
35      %speech presence probability
36      ANC.p = (1 + (ANC.q./(1-ANC.q)) .* (1+ANC.xi) .* exp(-ANC.nu) ).^-1;
37
38      %OM-LSA
39      ANC.G = ANC.G0.^ANC.p .* ANC.G_min.^(1-ANC.p);
40
41      %calc output signal
42      ANC.Fx = filterbank_synthesis(AC,ANC.G).*Fy;
```

Main function of the realtime-implementation in C++:

```
1   int main(void)
2   {
3     //init MCSE_core
4     aSetup.fs = FS;
5     aSetup.BlockLength = BLOCKLENGTH;
6     aSetup.Nmics = NIN;
7     aSetup.micpos = ArrayXXf::Zero(4,2);
8     aSetup.micpos << 0, 0.000,
9                      0, 0.053,
10                     0, 0.117,
11                     0, 0.169;
12    ac = new MCSE_core(&aSetup);
13    ac->isRunning = true;
14
15    mr = new MatlabReporter();
16    mr->EvalString("clear all; close all; clc");
17    mr->LivePlot_Init(ac);
18
19    pthread_create(&threadID_core, NULL, core_thread, (void *)ac);
20    cout << "starting core thread.\n";
21    pthread_create(&threadID_rtf, NULL, rtf_thread, (void *)ac);
22    cout << "starting rtf thread.\n";
23
24    cout << "starting PortAudio\n";
25    try
26    {
27      Pa_Initialize();
28      cout << "PortAudio version number = " << Pa_GetVersion() << "PortAudio
            version text = " << Pa_GetVersionText() << endl;
29
30      int numDevices = Pa_GetDeviceCount();
31      int motu_devicenum = -1;          //device number of the MOTU interface
32      if( numDevices < 0 )
33      {
34        cout << "ERROR: Pa_GetDeviceCount returned:" << numDevices << endl;
35        throw paDeviceUnavailable;
36      }
37
38      cout << "Number of devices = " << numDevices << endl;
39      for(int i=0; i<numDevices; i++ )
40      {
41        string device(Pa_GetDeviceInfo(i)->name);
42        cout << "\nDevice #" << i << ": " << device.data() << endl;
43        string motu("MOTU");
44        if (device.find(motu) != string::npos)
45        {
46          motu_devicenum = i;
47          cout << "\nFound MOTU Device: " << device.data() << endl;
48        }
49      }
50      if (motu_devicenum == -1)
51      {
52        cout << "ERROR: MOTU device was not found\n";
53        throw paDeviceUnavailable;
54      }
55
56      inputParameters.channelCount = NIN;        //record NIN channels
57      inputParameters.device = motu_devicenum;
58      inputParameters.hostApiSpecificStreamInfo = NULL;
59      inputParameters.sampleFormat = paFloat32;
```

```
60      inputParameters.suggestedLatency =
              Pa_GetDeviceInfo(motu_devicenum)->defaultLowInputLatency ;
61      inputParameters.hostApiSpecificStreamInfo = NULL; //See your specific
              host's API docs for info on using this field
62
63      outputParameters.channelCount = NOUT;         //output channels
64      outputParameters.device = motu_devicenum;
65      outputParameters.hostApiSpecificStreamInfo = NULL;
66      outputParameters.sampleFormat = paFloat32;
67      outputParameters.suggestedLatency =
              Pa_GetDeviceInfo(motu_devicenum)->defaultLowOutputLatency ;
68      outputParameters.hostApiSpecificStreamInfo = NULL;  //See your specific
              host's API docs for info on using this field
69
70      err = Pa_OpenStream( &stream, &inputParameters, &outputParameters,
              FS_SYSTEM, BLOCKSIZE_SYSTEM, paNoFlag,  Pa_AudioCallback, ac );
71      if( err != paNoError ) throw err;
72
73      err = Pa_StartStream( stream );
74      if( err != paNoError ) throw err;
75      ac->startTime = Pa_GetStreamTime(stream);
76      ac->isRunning = true;
77      do
78      {
79        mr->LivePlot_Update();
80      }
81      while( ac->isRunning );
82
83      err = Pa_CloseStream( stream );
84      if( err != paNoError ) throw err;
85    }
86    catch (PaError err)
87    {
88      Pa_Terminate();
89      cout << "An error occured while using the portaudio stream\n";
90      cout << "Error number: " << err << endl;
91      cout << "Error message: " << Pa_GetErrorText(err) << endl;
92      system("pause");
93      return err;
94    }
95
96    pthread_mutex_lock(&mut_core);
97    pthread_cond_signal(&cond_core);
98    pthread_mutex_unlock(&mut_core);
99    pthread_join(threadID_core, NULL);
100   printf("Waiting for core_thread to join.\n");
101
102   pthread_mutex_lock(&mut_rtf);
103   pthread_cond_signal(&cond_rtf);
104   pthread_mutex_unlock(&mut_rtf);
105   pthread_join(threadID_rtf, NULL);
106   printf("Waiting for rtf_thread to join.\n");
107
108   Pa_Terminate();
109   delete ac;
110   printf("Test finished.\n");
111   fileptr.close();
112   return 0;
113 }
```

Bibliography

[1] J. Benesty, M. M. Sondhi, and Y. Huang, *Springer Handbook of Speech Processing.* Berlin–Heidelberg–New York: Springer, 2008.

[2] P. Vary and R. Martin, *Digital Speech Transmission.* West Sussex: Wiley, 2006.

[3] P. C. Loizou, *Speech Enhancement: Theory and Practice.* Boca Raton: CRC Press, 2007.

[4] J. Benesty, S. Makino, and J. Chen, *Speech Enhancement.* Berlin–Heidelberg–New York: Springer, 2005.

[5] D. L. Wang and J. S. Lim, "The unimportance of phase in speech enhancement," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 30, no. 4, Aug. 1982.

[6] P. Mowlaee and R. Martin, "On phase importance in parameter estimation for single-channel source separation," *International Workshop on Acoustic Signal Enhancement in Aachen*, Sep. 2012.

[7] T. Gerkmann, M. Krawczyk, and R. Rehr, "Phase estimation in speech enhancement - unimportant important or impossible?" *IEEE 27-th Convention of Electrical and Electronics Engineers in Israel*, Nov. 2012.

[8] S. Schmitt, M. Sandrock, and J. Cronemeyer, "Single channel noise reduction for hands free operation in automotive environments," *AES 112-th Convention, Munich, Germany*, May 2002.

[9] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing.* Berlin–Heidelberg–New York: Springer, 2008.

[10] Y. A. Huang and J. Benesty, *Audio Signal Processing For Next-Generation Multimedia Communication Systems.* Boston: Kluwer Academic Publishers, 2004.

[11] R. Zelinski, "A microphone array with adaptive post-filtering for noise reduction in reverberant rooms," *International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 2578–2581, Apr. 1988.

[12] S. Gannot and I. Cohen, "Speech enhancement based on the general transfer function gsc and postfiltering," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 6, Nov. 2004.

[13] M. Taseska and E. A. Habets, "Mmse-based blind source extraction in diffuse noise fields using a complex coherence-based a priori sap estimator," *International Workshop on Acoustic Signal Enhancement*, Sep. 2012.

[14] M. Souden, J. Chen, J. Benesty, and S. Affes, "An integrated solution for online multichannel noise tracking and reduction," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 7, Sep. 2011.

[15] "Wandboard," Website, available online at http://http://www.wandboard.org/; visited on November 19th 2012.

[16] E. Vincent, "Improved perceptual metrics for the evaluation of audio source separation," INRIA, Centre de Rennes, Bretange Atlantique, Campus de Beaulieu, 35042 Rennes Cedex, France, Tech. Rep., 2011.

[17] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, "Subjective and objective quality assessment of audio source separation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 7, Sep. 2011.

[18] E. G. Williams, *Fourier acoustics - sound radiation and nearfield acoustical holography.* London: Academic Press, 1999.

[19] H. Kuttruff, *Room Acoustics*, 5th ed. London–New York: Spoon Press, 2009.

[20] S. Gannot, D. Burshtein, and E. Weinstein, "Signal enhancement using beamforming and nonstationarity with applications to speech," *IEEE Transactions on Signal Processing*, vol. 49, no. 8, Aug. 2001.

[21] I. A. McCowan and H. Bourlard, "Microphone array post-filter based on noise field coherence," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 11, no. 6, Nov. 2003.

[22] G. Stan, J. Embrechts, and D. Archambeau, "Comparison of different impulse response measurement techniques," Sound and Image Department, University of Liege, Belgium, Tech. Rep., 2002.

[23] M. Holters, T. Corbach, and U. Zölzer, "Impulse response measurement techniques and their applicability in the real world," *Proceedings of the 12th Int. Conference on Digital Audio Effects, Como, Italy*, Sep. 2009.

[24] T. D. Rossing, *Springer Handbook of Acoustics.* Berlin–Heidelberg–New York: Springer, 2007.

[25] Y. Huang, J. Benesty, and J. Chen, *Acoustic MIMO Signal Processing.* Berlin–Heidelberg–New York: Springer, 2006.

[26] S. Haykin, *Adaptive Filter Theory*, 4th ed. New Jersey: Prentice Hall, 2002.

[27] O. Hoshuyama, A. Sugiyama, and A. Hirano, "A robust adaptive beamformer for microphone arrays with a blocking matrix using constrained adaptive filters," *IEEE Transactions on Signal Processing*, vol. 47, no. 10, Oct. 1999.

[28] K. Li, Q. Fu, and Y. Yan, "A subband feedback controlled generalized sidelobe canceller in frequency domain with multi-channel postfilter," *2nd International Workshop on Intelligent Systems and Applications*, pp. 1–4, May 2010.

[29] J. Benesty, J. Chen, Y. Huang, and J. Dmochowski, "On microphone-array beamforming from a mimo acoustic signal processing perspective," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 15, no. 3, Mar. 2007.

[30] S. Gannot, "Array processing of nonstationary signals with application to speech," Tel-Aviv University, Tech. Rep., 2000.

[31] M. G. Shmulik, S. Gannot, and I. Cohen, "A sparse blocking matrix for multiple constraints gsc beamformer," *IEEE International Conference on Acoustics, Speech and Signal Processing*, Mar. 2012.

[32] E. Warsitz and R. Haeb-Umbach, "Blind acoustic beamforming based on generalized eigenvalue decomposition," *IEEE Transactions on audio, speech, and language processing*, vol. 15, no. 5, Jul. 2007.

[33] E. Warsitz, A. Krueger, and R. Haeb-Umbach, "Speech enhancement with a new generalized eigenvector blocking matrix for application in a generalized sidelobe canceller," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 73–76, May 2008.

[34] W. Herbordt and W. Kellermann, "Analysis of blocking matrices for generalized sidelobe cancellers for non-stationary broadband signals," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, May 2002.

[35] I. Cohen, "Analysis of two-channel generalized sidelobe canceller (gsc) with post-filtering," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, Nov. 2003.

[36] I.Cohen, "Multichannel post-filtering in nonstationary noise environments," *IEEE Transactions on Signal Processing*, vol. 52, no. 5, May 2004.

[37] E. Weinstein and O. Shalvi, "System identification using nonstationary signals," *IEEE Transactions on Signal Processing*, vol. 44, no. 8, Aug. 1996.

[38] I. Cohen, "Relative transfer function identification using speech signals," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 5, Sep. 2004.

[39] I. Cohen and B. Berdugo, "Speech enhancement for non-stationary noise environments," Lamar Signal Processing Ltd., Israel, Tech. Rep., 2001.

[40] I. Cohen, "Noise spectrum estimation in adverse environments: Improved minima controlled recursive averaging," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 5, Sep. 2003.

[41] J. Li, Q. Fu, and Y. Yan, "An approach of adaptive blocking matrix based on frequency domain independent component analysis in generalized sidelobe canceller," *IEEE 10th International Conference on Signal Processing*, pp. 231–234, Oct. 2010.

[42] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York: John Wiley, 2001.

[43] H.Saruwatari, T. Kawamura, T. Nishikawa, A. Lee, and K. Shikano, "Blind source separation based on a fast-convergence algorithm combining ica and beamforming," *IEEE Transactions on Audio, Speech and Language processing*, vol. 14, no. 2, Mar. 2006.

[44] B. Qin, H. Zhang, Q. Fu, and Y. Yan, "Subsample time delay estimation via improved gcc phat algorithm," *9th International Conference on Signal Processing*, pp. 2579–2582, Dec. 2008.

[45] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 5, Jul. 2001.

[46] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 2, Apr. 1985.

[47] Y. A. Huang and J. Benesty, "A multi-frame approach to the frequency-domain single-channel noise reduction problem," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 20, no. 4, May 2012.

[48] I. Cohen, "Relaxed statistical model for speech enhancement and a priori snr estimation," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, Sep. 2005.

[49] ——, "Optimal speech enhancement under signal presence uncertainty using log-spectral amplitude estimator," *IEEE Signal Processing Letters*, vol. 9, no. 4, Apr. 2002.

[50] O. Thiergart, G. D. Galdo, and E. A. P. Habets, "Signal-to-reverberant ratio estimation based on the complex spatial coherence between omnidirectional microphones," Int. Audio Labs. Erlangen, Erlangen, Germany, Tech. Rep., 2012.

[51] M. Souden, J. Chen, J. Benesty, and S. Affes, "Gaussian model-based multichannel speech presence probability," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 5, Jul. 2010.

[52] H. Fastl and E. Zwicker, *Psychoacoustics: Facts and Models.* Berlin–Heidelberg–New York: Springer, 1999.

[53] L. Lin, E. Ambikairajah, and W. Holmes, "Auditory filter bank design using masking curves," School of Electrical Engineering and Telecommunications, University of New South Wales, Australia, Tech. Rep., 2003.

[54] B.C.J.Moore and B.R.Glasberg, "Suggested formula for calculating auditory-filter bandwidth and excitation patterns," Department of Experimental Psychology, University of Cambridge, Cambridge CB2 3EB, Tech. Rep., 1983.

[55] J. Holdsworth, I. Nimmo-Smith, R. Patterson, and P. Rice, "Implementing a gamamtone filterbank," Cambridge Electronic Design, Science Park, Cambridge, Tech. Rep., 1988.

[56] B. Huang, C. Zhu, W. Fan, Y. Tao, and Q. Zeng, "Microphone array speech enhancement based on filter bank generalized sidelobe canceller," College of Information and Communications, Guilin University of Electronic Technology, Tech. Rep., 2009.

[57] R. Talmon, I. Cohen, and S. Gannot, "Relative transfer function identification using convolutive transfer function approximation," *IEEE Transactions on audio, speech, and language processing*, vol. 17, no. 4, May 2009.

[58] "Timit acoustic-phonetic continuous speech corpus," Website, available online at http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1; visited on January 16th 2013.

[59] "The kiel corpus of read speech vol. 1," Website, available online at http://www.ipds.uni-kiel.de/forschung/kielcorpus.en.html; visited on January 16th 2013.

[60] "The kiel corpus of spontaneous speech vol. 1-3," Website, available online at http://www.ipds.uni-kiel.de/forschung/kielcorpus.en.html; visited on January 16th 2013.

[61] "Matlab r2010b," Website, available online at http://www.mathworks.de/products/matlab/; visited on January 16th 2013.

[62] R. Bencina and P. Burk, "Portaudio v19," Website, available online at http://www.portaudio.com; visited on November 19th 2012.

[63] "Asio sdk 2.2," Website, available online at http://www.steinberg.net/en/company/developer.html; visited on November 19th 2012.

[64] M. Frigo and S. G. Johnson, "Fftw 3.3.2," Website, available online at http://www.fftw.org; visited on November 19th 2012.

[65] G. Guennebaud, B. Jacob *et al.*, "Eigen 3.1.2," Website, available online at http://eigen.tuxfamily.org; visited on November 19th 2012.

[66] R. Johnson, "Pthreads win32 2.9.1," Website, available online at http://www.sourceware.org/pthreads-win32; visited on November 19th 2012.

[67] "Matlab engine," Website, available online at http://www.mathworks.de/de/help/matlab/calling-matlab-engine-from-c-c-and-fortran-programs.html; visited on November 19th 2012.

[68] G. Reuven, S. Gannot, and I. Cohen, "Joint noise reduction and acoustic echo cancellation using the transfer-function generalized sidelobe canceller," Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa 32000, Israel, Tech. Rep., Jan. 2006.

[69] A. Marti, M. Cobos, and J. J. Lopez, "A real-time sound source localization and enhancement system using distributed microphones," *130th Audio Engineering Society Convention*, May 2011.

[70] D. N. Zotkin and R. Duraiswami, "Accelerated speech source localization via a hierarchical search of steered response power," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 5, Sep. 2004.

[71] E. Warsitz and R. Haeb-Umbach, "Acoustic filter-and-sum beamforming by adaptive principal component analysis," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, Mar. 2005.

[72] I. Cohen, "Identification of speech source coupling between sensors in reverberant noisy environments," *IEEE Signal Processing Letters*, vol. 11, no. 7, Jul. 2004.

[73] L. Ying, L. Jianping, and Z. Yiwen, "Minimum entropy-based acoustic source localization with laplace distribution," *10th International Conference on Signal Processing*, pp. 498–501, Oct. 2010.

[74] C. Jingdong, J. Benesty, and H. Yiteng, "Robust time delay estimation exploiting redundancy among multiple microphones," *IEEE Transactions on Speech and Audio Processing*, pp. 549–557, Nov. 2003.

[75] D. L. Maskell and G. S. Woods, "The estimation of subsample time delay of arrival in the discrete-time measurement of phase delay," *IEEE Transactions on Instrumentation and Measurement*, vol. 48, no. 6, Dec. 1999.

[76] N. Yousefian and P. C. Loizou, "A dual-microphone speech enhancement algorithm based on the coherence function," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 599–609, Dec. 2011.

[77] H. Sawada, R. Mukai, S. Araki, and S. Makino, "A robust and precise method for solving the permutation problem of frequency-domain blind source separation," *IEEE Transactions on Speech and Audio Processing*, vol. 12, no. 5, Sep. 2004.

[78] S. Y. Low and S. Nordholm, "A robust multichannel speech enhancement method based on decorrelation," *IEEE International Symposium on Circuits and Systems*, pp. 2875–2878, Jul. 2005.

[79] J. Benesty, J. Chen, and Y. Huang, "Binaural noise reduction in the time domain with a stereo setup," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 8, Nov. 2011.

[80] S. G. Osgouei and M. Geravanchizadeh, "Dual-channel speech enhancement based on a hybrid particle swarm optimization algorithm," *5th International Symposium on Telecommunications*, pp. 873–877, Mar. 2011.

[81] E. Weinstein, M. Feder, and A. V. Oppenheim, "Multi-channel signal separation by decorrelation," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 4, Oct. 1993.

[82] B. Zamani, M. Rahmani, and A. Akbari, "Residual noise control for coherence based dual microphone speech enhancement," *IEEE International Conference on Computer and Electrical Engineering*, pp. 601–605, Dec. 2008.

[83] R. Talmon, I. Cohen, and S. Gannot, "Convolutive transfer function generalized sidelobe canceler," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 17, no. 7, Sep. 2009.

[84] T. Lotter and P. Vary, "Dual-channel speech enhancement by superdirective beamforming," Institute of Communication Systems and Data Processing, RWTH Aachen University, Tech. Rep., 2005.

[85] W. Herbordt and W. Kellermann, "Efficient frequency-domain realization of robust generalized, sidelobe cancellers," *IEEE Fourth Workshop on Multimedia Signal Processing*, pp. 377–382, Aug. 2002.

[86] W. Herbordt, H. Buchner, S. Nakamura, and W. Kellermann, "Multichannel bin-wise robust frequency-domain adaptive filtering and its application to adaptive beamforming," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 4, May 2007.

[87] J. Bitzer, K. U. Simmer, and K. Kammeyer, "Multi-microphone noise reduction by postfilter and superdirective beamformer," University of Bremen, FB-1, Dept. of Telecommunications, Tech. Rep., 1999.

[88] K. Lae-Hoon, M. Hasegawa-Johnson, and S. Koeng-Mo, "Generalized optimal multimicrophone speech enhancement using sequential minimum variance distortionless response(mvdr) beamforming and postfiltering," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, May 2006.

[89] S. Y. Jeong, K. Kim, and J. H. Jeong, "Adaptive noise power spectrum estimation for compact dual channel speech enhancement," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1630–1633, Mar. 2010.

[90] C. S. Lin and C. Kyriakakis, "Multi-frequency noise removal based on reinforcement learning," *115th Audio Engineering Society Convention*, Oct. 2003.

[91] I. Cohen, "On speech enhancement under signal presence uncertainty," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. 661–664, Aug. 2002.

[92] ——, "On the decision-directed estimation approach of ephraim and malah," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, May 2004.

[93] A. Petrovsky, M. Parfieniuk, and K. Bielawski, "Psychoacoustically motivated nonuniform cosine modulated polyphase filter bank," Department of Real Time Systems, Bialystok Technical University, Poland, Tech. Rep., 2002.

[94] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 16, no. 1, Jan. 2008.

[95] ——, "Subjective comparison and evaluation of speech enhancement algorithms," Department of Electrical Engineering, The University of Texas at Dallas, USA, Tech. Rep., 2006.

[96] M. Brandstein and D. Ward, *Microphone Arrays*. Berlin–Heidelberg–New York: Springer, 2001.