Technical Sciences
Oude Waalsdorperweg 63
2597 AK Den Haag
P.O. Box 96864
2509 JG The Hague
The Netherlands

www.tno.nl

T +31 88 866 10 00
F +31 70 328 09 61
infodesk@tno.nl

TNO report

TNO-DV 2012 S007

Real-time Beamforming and Sound Classification Parameter Generation in Public Environments

Date	1 February 2012
Author(s)	J.J.M. van de Sande
Number of pages	81 (incl. appendices)
Number of appendices	5
Assignor	TNO, Research Group Acoustics & Sonar
Project number	032.30960/01.04

Abstract

Undesired, human behavior in public environments is an increasing issue in today's society. The overload of security operators and law enforcement addresses the need for automatic detection of anomalous behavior. The EU-project ADABTS aims to facilitate the protection of EU citizens, property and infrastructure against threats of terrorism, crime and riots, by the automatic detection of abnormal human behaviour. At the Acoustics & Sonar department of TNO Defence and Safety, part of this problem is addressed by means of acoustical detection of anomalous events.

The approach is based on 'scanning' public environments by applying beamforming on the outputs of an acoustical sensor array and applying classification algorithms for detecting specific sources. In this Master's thesis, an initial step is taken with the development of a real-time beamforming system that delivers required sound parameters used in sound classification.

A number of different beamforming methods have been considered, differing in performance and computational complexity. Conventional methods like Delay and Sum (DAS), possibly combined with the use of static, frequency-invariant windows, lack spatial resolution at especially lower frequencies and are unable of coping with multiple interfering sources. Other methods provide an improved performance on the cost of increased complexity.

The method known as Minimum Variance Distortionless Response (MVDR) beamforming maintains a high spatial resolution at lower frequencies. Two main versions of this method are frequently used: a static (non-input-based) one and a dynamic (input-based) one. Static MVDR (SMVDR) is able to maintain performance at lower frequencies, but due to its static nature it does not add any extra value in multi-source environments. Dynamic MVDR (DMVDR), on the other hand, is partly capable of filtering away undesired coherent interferers and also has an improved spatial response in single-source environments. Its computational complexity, still, is an important bottleneck.

The search for less intensive beamforming methods leads to a way of adaptive beamforming. Beamformers in which static beamforming and dynamic filtering are split in two different parts, are able to alleviate complexity. However, the need for adding extra elements to account for target-signal cancellation in multi-source environments destroys the computational advantages, making it unsuitable.

The developed real-time application takes into account the intensive routines of DMVDR. Since the properties of the deployment platform are not known in advance, it is supplied with a mechanism for adapting to different and changing, available hardware resources such as available CPU-time, arithmetic units and memory. In this way it will always deliver the best possible solution, based on what the user is offering. Still, an extensive implementation process has led to a relatively fast execution of the algorithm.

The system is supplied with a user interface for controlling a number of parameters and for obtaining the first visual effects. Furthermore, it is provided with a user-friendly mechanism for calibrating the system for each possible deployment environment.

Samenvatting

Ongewenst, menselijk gedrag in publieke omgevingen is een toenemend probleem in de huidige maatschappij. Overbelasting van operators van beveiligingscentrales en wetshandhavers vergroot de vraag naar automatische detectie van afwijkend gedrag. Het EU-project ADABTS heeft als doel burgers, eigendommen en infrastructuur te beschermen tegen terroristische acties, criminaliteit en rellen, door middel van automatische detectie van afwijkend, menselijk gedrag. Op de afdeling Akoestiek & Sonar van TNO Defensie en Veiligheid, wordt een deel van dit probleem behandeld door middel van akoestische detectie van afwijkende gebeurtenissen.

De aanpak is gebaseerd op het akoestisch 'scannen' van publieke omgevingen door het toepassen van bundelvorming op de uitgangen van een akoestisch sensor array en het toepassen van geluidclassificatie algoritmen voor de detectie van specifieke bronnen. In deze Master thesis wordt een eerste stap genomen met de ontwikkeling van een real-time bundelvormingssysteem dat de eerste vereiste parameters levert voor de uiteindelijke geluidclassificatie.

Een aantal manieren van bundelvorming is beschouwd, elk verschillend in prestatie en rekencomplexiteit. De resolutie van standaard methoden als Delay and Sum (DAS), eventueel gecombineerd met een statisch, frequentie-invariant window, verslechtert vooral bij lagere frequenties. Bovendien presteren dergelijke methoden minder goed bij de aanwezigheid van een of meerdere stoorbronnen. Alternatieve methoden bieden voordelen ten koste van toenemende complexiteit.

Minimum Variance Distortionless Response (MVDR) bundelvorming behoudt een hoge resolutie bij lagere frequenties. Twee veelgebruikte versies van deze methode zijn een statische (niet input-afhankelijke) en dynamische (input-afhankelijke) versie. Static MVDR (SMVDR) behoudt de prestatie van conventionele bundelvormers bij hogere frequenties, ook bij lagere frequenties. Door de onafhankelijkheid van sensor input levert de methode echter geen toegevoegde waarde bij de aanwezigheid van stoorbronnen. Dynamic MVDR (DMVDR) is deels in staat om coherente stoorbronnen weg te filteren en heeft standaard een hogere resolutie bij de aanwezigheid van slechts één bron. De rekenintensiteit is echter iets waar rekening mee gehouden moet worden.

De zoektocht naar een minder complex bundelvormingsalgoritme leidt tot adaptieve bundelvorming. De bundelvormer wordt in dit geval gesplitst in een statische bundelvormer en een adaptief filter om stoorbronnen en ruis weg te filteren om zodoende de complexiteit te verlagen. Echter, de maatregelen die dienen te worden genomen om het wegfilteren van het gewenste bronsignaal te voorkomen, hebben een dusdanig negatief effect op de rekencomplexiteit dat dit algoritme geen toegevoegde waarde levert.

Het ontwikkelde real-time systeem houdt rekening met de rekenintensieve routines van DMVDR. Omdat de eigenschappen van het platform van het systeem van tevoren nog niet bekend zijn, is het voorzien van een mechanisme dat zich aanpast aan verschillende en veranderende, beschikbare hardware voorzieningen zoals beschikbare CPU-tijd, rekeneenheden en geheugen. Op deze manier levert het systeem altijd de best mogelijke oplossing, afhankelijk van wat de gebruiker aanbiedt. Ondanks dit, heeft een uitgebreid implementatieproces geleid tot een relatief snel en efficiënt algoritme.

De applicatie is voorzien van een gebruikersinterface voor het instellen van een aantal parameters en voor de weergave van de eerste visuele effecten. Bovendien is een gebruiksvriendelijk mechanisme toegevoegd voor de kalibratie van het systeem in iedere mogelijke omgeving.

Preface

Tomorrow morning at 08:45h sharp, it has been exactly three years ago I attended the very first course of my study at Delft University of Technology. An introduction in matrix operations formed the kick-off of university education. After a bridging program of one year and a two-year Master's study Embedded Systems, this Master's thesis lying in front of you, is the result of my final project at TNO Defence and Safety.

The Acoustics & Sonar department of TNO provided me the opportunity of being part of a very interesting and multi-disciplinary project. In the past 9 months I learned a lot: besides theory and practical skills, I also gained more experience with my personal characteristics. As with every project, things did not always run smoothly. Unexpected disappointments pass by, leading to progression-less periods and a faster and faster approaching deadline. In my opinion, perseverance and persistence are underestimated personal qualities that I think I improved the past months.

Of course, colleagues are there for support. Still, I would like to specially thank a number of them because of their pleasant and supportive cooperation.

First of all, I would like to thank my daily supervisor, Dr. ir. Arthur Berkhoff, for his guidance. His knowledge, meaningful comments and sincerity helped me in obtaining this result.

Secondly, Ir. Erwin Jansen, was a relief in taking care of peripheral project issues. It was very pleasant working with him and I always appreciated his interest in the progression I made.

I also would like to thank project leader Ing. Toon Beeks for keeping me on track in finding a right balance between project results and study related tasks.

Roommates are of great influence on experienced atmosphere and first aid in specific, practical issues. Ir. Derk Land provided both in a very positive way. Special thanks go out to him.

I would like to thank Ing. Frans Staats for helping me in performing the verification tests of the system and taking care of all the necessary provisions.

An acknowledgement goes out to Dr. ir. Peter Beerens for providing me some additional information on a few theoretical topics.

But maybe the most important persons not mentioned yet, gave me the opportunity of following my Master's study. Though lacking any possible technical background, my parents strongly encouraged me in studying and provided me the resources of taking this educational path. Without this support, there would be no thesis to read.

Finally, I would like to hearty thank all non-mentioned employees of TNO that have been helpful in any other way.

Last 9 months have been a pleasure. I sincerely hope you as a reader will enjoy reading this report and that the developed system will be of great value for TNO; now and in the future.

The Hague, 1 February 2012

Jeroen van de Sande

Nomenclature

Symbols

$(.)_f$	Frequency-domain representation if not clear from context
$(.)_t$	Time-domain representation if not clear from context
$(.)^H$	Complex conjugate transpose
$(.)^T$	Transpose
$(.)^*$	Complex conjugate
$\|.\|$	Absolute value
$\|\|.\|\|$	Euclidian norm
$\underline{\boldsymbol{0}}_M$	Zero-vector of length M
a_n	Attenuation between source and sensor n
B	Number of beams
$\boldsymbol{B}$	Blocking matrix
$\boldsymbol{B}_f$	Blocking matrix at frequency index f
c	Speed of sound in [m/s]
$\underline{\boldsymbol{c}}$	Constraint vector
$\boldsymbol{C}$	Constraint matrix
$\boldsymbol{C}_f$	Constraint matrix at frequency index f
d	Inter-sensor distance in [m]
e	Noise plus interference
$\hat{e}$	Estimated noise plus interference
$E\{.\}$	Expectation operator
$\underline{\boldsymbol{f}}$	Response vector
F	Number of frequencies
F_c	Centre frequency in [Hz] of frequency band to which beamforming is applied
F_l	Minimum allowed frequency in [Hz] of signal
F_h	Maximum allowed frequency in [Hz] of signal
F_N	Nyquist frequency in [Hz]
F_s	Sampling frequency in [Hz]
F_T	Source/target frequency in [Hz]
J	Cost function
k	Frame index
K	Constant norm of NCAF filter
L	Filter length
m	Discrete time index
$n=1..N$	Number of sensors
p	Proportion of centre frequency defining difference between centre frequency and lowest frequency of a frequency bin.
P	Power
Q	Eigenvector matrix
$\underline{\boldsymbol{r}}_{ex}$	Correlation between scalar e and vector $\underline{\mathbf{x}}$
$r(t)$	Continuous-time source signal at time t
$r[m]$	Discrete-time source signal at time sample m
$\check{r}[m]$	Estimated, discrete-time source signal at time sample m
$\boldsymbol{R}_{xx}$	Signal autocorrelation matrix between sensors
$\boldsymbol{R}_{uu}$	Noise autocorrelation matrix between sensors
s	Frequency-domain source signal
$\hat{s}$	Estimated frequency-domain source signal

$\hat{s}_f$	Estimated frequency-domain source signal at frequency index f
$\underline{\boldsymbol{s}}_{c,f}$	Output vector of constrained FBF at frequency index f containing all sensors
$\underline{\boldsymbol{s}}_{b,f}$	Output vector of unconstrained ANC at frequency index f containing all sensors
T_s	Sampling period in [s]
$v_n(t)$	Continuous-time noise at sensor n at time t
$\underline{\boldsymbol{v}}$	Gradient vector or frequency-domain noise vector containing all sensors
$\underline{\hat{\boldsymbol{v}}}$	Estimated gradient vector
w_l	Filter coefficient of tap l
$\underline{\boldsymbol{w}}$	Weight/filter vector in time or frequency domain
$\underline{\boldsymbol{w}}_{c,f}$	Constrained weight/filter vector at frequency index f containing all sensors
$\underline{\boldsymbol{w}}_{u,f}$	Unconstrained weight/filter vector at frequency index f containing all sensors
$x_n(t)$	Continuous-time signal at sensor n at time t
$x_n[m]$	Discrete-time signal at sensor n at time sample m
$\underline{\boldsymbol{x}}$	Frequency-domain signal containing all sensors
$\underline{\boldsymbol{z}}_f$	Output vector of BM at frequency index f containing all sensors
z^{-L}	Delay of L samples
α	Filter adaptation constant
α_n	Filter weight of sensor n
$\Gamma_{VnVp}(e^{j\Omega})$	Noise coherence between sensor n and p at frequency $F = \Omega F_s$
$\boldsymbol{\Gamma}_{VV}$	Noise coherence matrix
$\delta(t)$	Value of Dirac-function at time t
ε	Error signal
θ_0	Azimuth viewing angle in [°] perpendicular to array axis
θ_s	Azimuth source angle in [°] perpendicular to array axis
λ	Lagrange multiplier; eigenvalue or wavelength, depending on context
Λ	Eigenvalue matrix
μ	Constant
ξ	Phase shift for look direction
ς	Phase shift for source direction
τ	Time delay constant
φ_n	Lower coefficient limit of CCAF filter
ϕ_0	Elevation viewing angle in [°] perpendicular to array axis
ϕ_s	Elevation source angle in [°] perpendicular to array axis
ψ_n	Upper coefficient limit of CCAF filter

Abbreviations

ABM	Adaptive Blocking Matrix
ADABTS	Automatic Detection of Abnormal Behaviour and Threats in crowded Spaces
AGSC	Adaptive Generalized Sidelobe Canceller
ALU	Arithmetic Logic Unit
ANC	Adaptive Noise Canceller
BM	Blocking Matrix
CCAF	Coefficient-constrained Adaptive Filter
CMMSE	Constrained Minimum Mean Square Error
CMSINR	Constrained Maximum Signal to Interference plus Noise Ratio
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DAS	Delay and Sum
DCT	Discrete Cosine Transform

DFT	Discrete Fourier Transform
DI	Directivity Index
DMVDR	Dynamic Minimum Variance Distortionless Response
FBF	Fixed Beamformer
FDAF	Frequency-domain Adaptive Filter
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GCC	GNU C Compiler
GJBF	Griffiths-Jim Beamformer
GPU	Graphical Processing Unit
GSC	Generalized Sidelobe Canceller
IFFT	Inverse Fast Fourier Transform
LCMV	Linear Constraint Minimum Variance
LE	Logic Element
LMS	Least Mean Squares
MFCC	Mel-frequency Cepstral Coefficient
MIC	Multiple Input Canceller
MMSE	Minimum Mean Square Error
MSE	Mean Square Error
MVDR	Minimum Variance Distortionless Response
NCAF	Norm-constrained Adaptive Filter
NLMS	Normalized Least Mean Squares
PC	Personal Computer
PSD	Power Spectral Density
RMS	Root-Mean-Square
SGD	Steepest Gradient Descent
SMVDR	Static Minimum Variance Distortionless Response
ULA	Uniform Linear Array
WNG	White Noise Gain

Contents

1 Introduction

1.1 Background

The ADABTS project, started in 2008, tries to address the increasing need for the protection of citizens and infrastructure against threats of terrorism and crime. Due to the increase of terrorism and aggression in general, security operators and law enforcement experience a growing inability of ensuring the safety of our society. Also verbal aggression in soccer stadiums is becoming an increasingly important issue nowadays. The ADABTS project, which stands for Automatic Detection of Abnormal Behavior and Threats in crowded Spaces, aims to automatically detect anomalous, human behaviour. The desired strength of the project is achieved by combining the added value of multiple audio-visual disciplines such as emotion and movement detection (video) as well as the detection of anomalous sounds (audio) such as breaking glass, gun shots and screaming.

1.2 Goal

The Acoustics & Sonar department of TNO Defence and Safety is involved in the acoustical part of the ADABTS project. Goal is to automatically detect specific, pre-defined, anomalous sounds. This Master's thesis forms the initial step by providing an initial set of acoustical parameters for (i) providing initial operator support and (ii) supporting the application of sound classification algorithms, by means of beamforming and parameter generation. The aim of this graduation project therefore can be defined as:

"Developing a real-time, portable, multi-beamforming system, capable of delivering sound classification parameters and preliminary operator support in public environments."

This goal can be specified with regard to some definitions and required parameters. The aim is to continuously obtain a complete acoustical image of the environment the system is deployed in. This means that a number of acoustical beams have to be created, as depicted in Figure 1-1, covering a complete 2D-grid in real-time. Real-time in this sense is defined as:

Real-time: the ability of the system to keep up with all incoming data without the build-up of an increasing input-output delay and with a maximum average input-output delay of 500 [ms].

The following, pre-selected, sound parameters have to be generated:

- Mel-frequency Cepstral Coefficients (MFCCs).
- Pitch frequencies.
- Root Mean Square-values (RMS-values).

These values have to be generated for each constructed beam. The easiest way of computing these parameters is by using the frequency domain representation of the signal. This means all beams are required to be (transformed to) a frequency domain representation. Appendix E describes the way in which the parameters are computed.

Furthermore, a requirement is that the user is able to select one specific 2D-point in the grid to which the user can listen, meaning that for this particular point the complete audio signal has to be reconstructed in time domain and played on the platform.

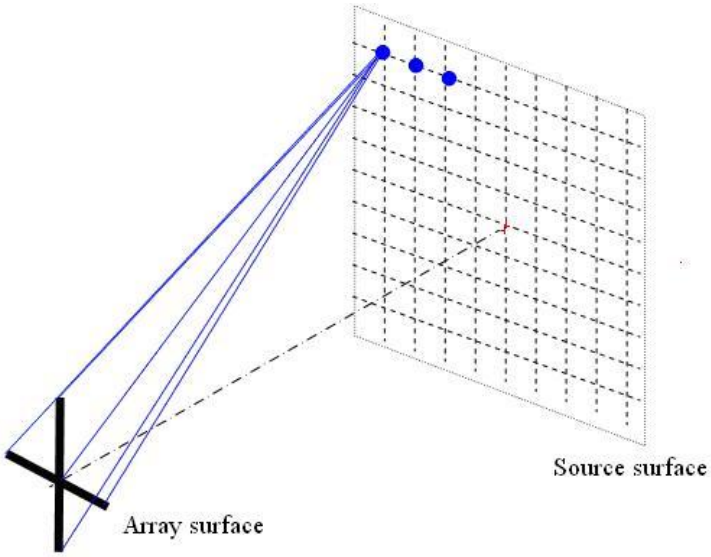


Figure 1-1 Overview of an acoustical cross-array constructing beams for covering the source surface.

In the final system, the purpose is to make use of a 2D-cross array which is already present. Since the array is two-dimensional, only a surface can be scanned instead of a complete 3D-space, which must be taken into account in positioning. The array contains 24 equidistant microphones with 4 side branches, each consisting of 2 microphones (see appendix A) adding up to a total of 32 microphones. The microphones of the two main, linear branches are located at a distance of 0.13 [m] from each other. The array is supplied with a NIOS II development board connected to an Altera Stratix II FPGA (EP2S60F672C3N). The array contains 16 AD-converters, each connected to two microphones, which read out the 24-bit samples at a rate of 10 [kHz]. The FPGA buffers the data to packets of 10 [ms] after which the NIOS II board sends them over Ethernet to a connected PC. At the PC-side, an interface is present that simply writes all received characters to a file. A timeserver running on the PC takes care of synchronizing the array with the PC.

Because of the early start of the acoustical part of the total ADABTS project, not much is known about the final deployment platform yet. The only thing known is that the final system will be a non-real-time, windows-based, 32-bit system on which other parties may run their application or at least connect to. This must be born in mind. Further, interest of other companies outside the scope of this project gives reason for using this acoustical part of the ADABTS project as a stand-alone system. This will also be taken into account by means of providing dedicated user interfaces.

Due to the presence of an FPGA at the array, this might be a considerable option as an implementation platform. An assessment must be made to estimate the required and available amount of logic and memory, but also implementation time and flexibility.

1.3 Approach

The first step in the whole process is to carry out a study on beamforming. Beamforming is a thoroughly studied topic and is used in all kinds of different contexts. It is important to emphasize the fact that a large number of beams have to be constructed in a real-time fashion. A very advanced algorithm may yield impressive results but may not at all be suited in our context due to its computational complexity. The first part of this report will describe the principle of beamforming based on conventional Delay and Sum beamforming. Furthermore, other, more advanced algorithms will be described and evaluated based on their performance and complexity. Finally, after an evaluation, some constraints will be imposed derived from the particular structure of the array used in this project.

The second part of this report will mainly focus on turning the requirements of the project and the results of the beamforming evaluation in a working, real-time solution. A system overview will be given, together with more detailed decisions and descriptions of implemented mechanisms. Attention will also be given to user interfacing and some optimizations used to speed up executions.

2 Beamforming

Beamforming is a signal processing technique that provides the ability for a sensor (actuator) array to focus on a specific source (destination) with a particular angular position with respect to the array. The fundamental methodology is based on the constructive and destructive interference the array elements experience, caused by the difference in length of the propagation paths between the array elements and the source (destination), leading to spatial filtering. For this particular project, the case of a microphone array, focusing on a point in space from which specific sounds may be emitted, is considered.

In essence we can distinguish two approaches for beamforming: one approach in which the spatial filtering coefficients are computed statically, in advance, regardless of the sensor signals and one that dynamically determines its coefficients by taking into account the sensor signals.

First, the basic technique of beamforming will be discussed using the most general Delay and Sum-method. After this, other possible (dynamic) techniques are outlined and compared with each other after which a decision for a beamforming method is made, based on the properties of the different techniques and the requirements and constraints for this project.

In the following discussion, a uniform linear array (ULA) is considered, consisting of a line of *N* equidistant microphones, unless other indicated.

2.1 Static beamforming

2.1.1 Delay and Sum

Microphone arrays exploit the differences in travel times between source and microphones, as illustrated in Figure 2-1.

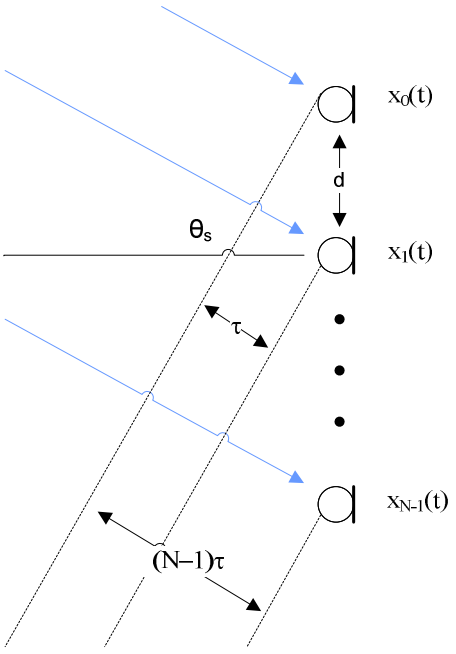


Figure 2-1 Travel time difference representation.

The continuous-time signal received at sensor *n=0,..,N-1* is a delayed and attenuated version of the original signal r(*t*) originated at the source, polluted by interference plus noise $v_n(t)$:

$$x_n(t) = a_n r(t - n\tau) + v_n(t) \tag{2.1}$$

In which a_n is the attenuation factor between the source and sensor *n* and time constant τ is defined as:

$$\tau = \frac{d}{c}\sin(\theta_s) \qquad [\text{s}] \tag{2.2}$$

With *c* the speed of sound in [m/s], *d* the distance between two neighbouring sensors in [m] and θ_s the angle between the source and the array axis in [°], as illustrated in Figure 2-1.

Assuming the signal is sampled with a sampling frequency of $F_s = \frac{1}{T_s}$ such that for each sensor we can write $x_n[m] = x_n(mT_s)$, Figure 2-2 depicts the structure of a conventional beamformer in which α_n are the beamformer weights per sensor *n*.

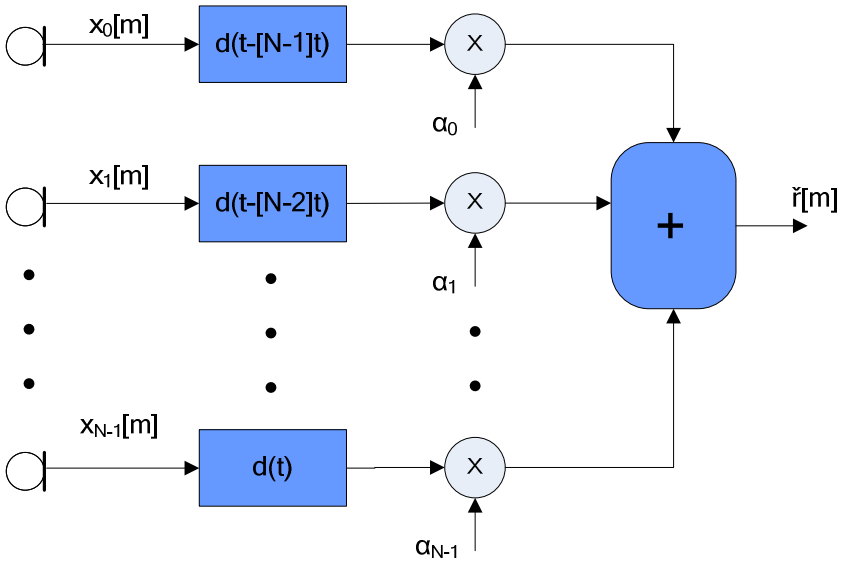


Figure 2-2 General structure beamformer.

This leads to the discrete-time output signal:

$$\breve{r}[m] = \sum_{n=0}^{N-1} \alpha_n x_n[m]\delta(t - [N - n - 1]\tau) \tag{2.3}$$

The signals x_n are time shifted, leading to a change in phase depending on the position of the sensor. Generalizing (2.3) with vector notation and translating time shifts in the time domain to phase shifts in the frequency domain, the frequency domain representation $\hat{s}[k]$ of frame *k* of the output signal $\hat{r}[m]$ of frequency F_c is defined as:

$$\hat{s}[k] = \underline{\mathbf{w}}^H \underline{\mathbf{x}}[k] \tag{2.4}$$

With [18]:

$$\underline{\mathbf{w}} = [\alpha_0, \alpha_1 e^{-j\xi}, \cdots, \alpha_{N-1} e^{-j(N-1)\xi}]^T \tag{2.5}$$

$$\underline{\mathbf{x}}[k] = [1, e^{-j\varsigma}, \cdots e^{-j(N-1)\varsigma}]^T s[k] + \underline{\mathbf{v}}[k] \tag{2.6}$$

$$\xi = \frac{2\pi F_c d}{c} \sin(\theta_0) \tag{2.7}$$

$$\varsigma = \frac{2\pi F_T d}{c} \sin(\theta_s) \tag{2.8}$$

In which $(.)^T$ denotes the transpose and $(.)^H$ the complex conjugate transpose. α_n is the amplitude weight for sensor *n*, F_c is the centre frequency in [Hz] of the frequency band considered, F_T the frequency in [Hz] of the source/target signal, θ_0 represents the angle in [°] from which a maximum gain is obtained, i.e. the 'look' direction and θ_s is the angle in [°] of the incoming waves from the source. The frequency domain representation of the time domain source signal $r[m]$ at centre frequency F_c is again defined by $s[k]$.

Based on above formulas, the total gain of the array in the look-direction equals $\sum_{n=0}^{N-1} \alpha_n$. For the particular case of Delay and Sum (DAS) beamforming, the amplitude weights α_n all equal $\frac{1}{N}$. In other words: only a phase shift per sensor is performed to focus on the look direction θ_0 . Other beamforming methods, assign different amplitude weights to each sensor, leading to an improved spatial response. A few of those methods will be discussed in the coming paragraphs.

2.1.1.1 Aliasing

Beamforming is subject to two types of aliasing: frequency and spatial aliasing. Frequency aliasing occurs when the highest frequency F_h of the signal is larger than or equal to half the sample frequency F_s [21], such that this constraint to avoid frequency aliasing can be defined as:

$$F_h < \frac{1}{2} F_s = F_N$$

With F_N representing the Nyquist frequency.

Spatial aliasing, on the other hand, occurs when the inter-sensor distance *d* is larger than half the minimum wavelength $\lambda_{\min}$ that is evaluated, leading to the constraint [22]:

$$d < \frac{\lambda_{\min}}{2} = \frac{c}{2F_h}$$

Figure 2-3 depicts the beam responses as polar plots of a two-sensor linear array with varying sensor distances in relation to the wavelength to show the influence of spatial aliasing. When obeying the spatial aliasing constraint (left image), no sidelobes appear. As λ decreases with respect to the sensor distance, more and more sidelobes appear.

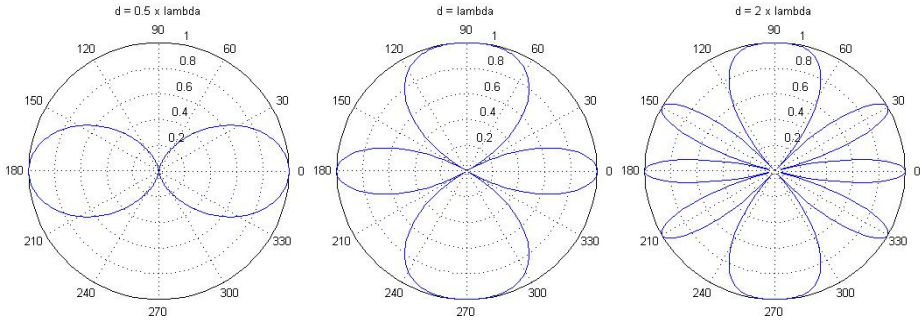


Figure 2-3 Polar plots of spatial aliasing with different ratios of sensor distance and wavelength.

2.1.1.2 Far-field assumption

In the above elucidation, the source is considered to be in the 'far-field': the distance between source and array is large enough for assuming the waves impinging on the array to be planar opposed to spherical. This is, however, not per definition correct.

Planar waves are waves of which the wave fronts are normal to the direction of propagation [12], so in this case parallel with respect to every sensor in the array. A point-source in essence generates spherical waves, but when it is located far enough from the array, the waves may be considered planar on arrival at the sensors [1]. Reconstructing the original signal by using only the angle between the source and the center of the array for every sensor then will suffice.

Figure 2-4 gives a visual impression of this matter by means of an image of the wavefield of a sine source operating at different frequencies and at different distances from an array consisting of 12 equidistant (0.13 [m]) sensors.

It is clear that a larger source-array distance and a lower source frequency will lead to less influence caused by the planar wave assumption. To finalize this discussion, the actual influence of the planar wave assumption on non-planar waves in terms of the spatial response is simulated and shown in Figure 2-5. The planar wave plot shows the response when the source is assumed to emit waves that are planar on arrival at the array. The spherical wave plot on its turn correctly assumes these waves to be spherical.

Ignoring spatial aliasing, it is easy to see that for a small source-array distance the assumption will have a large effect on the spatial response at angles near the source direction. However, at the operating distance of 12-16 [m] applied in this concerning project, the effect is almost negligible.

A question that may arise is, why not assume all waves to be spherical? The problem in this case is that the source distances have to be known in advance, which is not per definition the case. Furthermore, it reduces generality when source distances for every look direction have to be incorporated per application site.

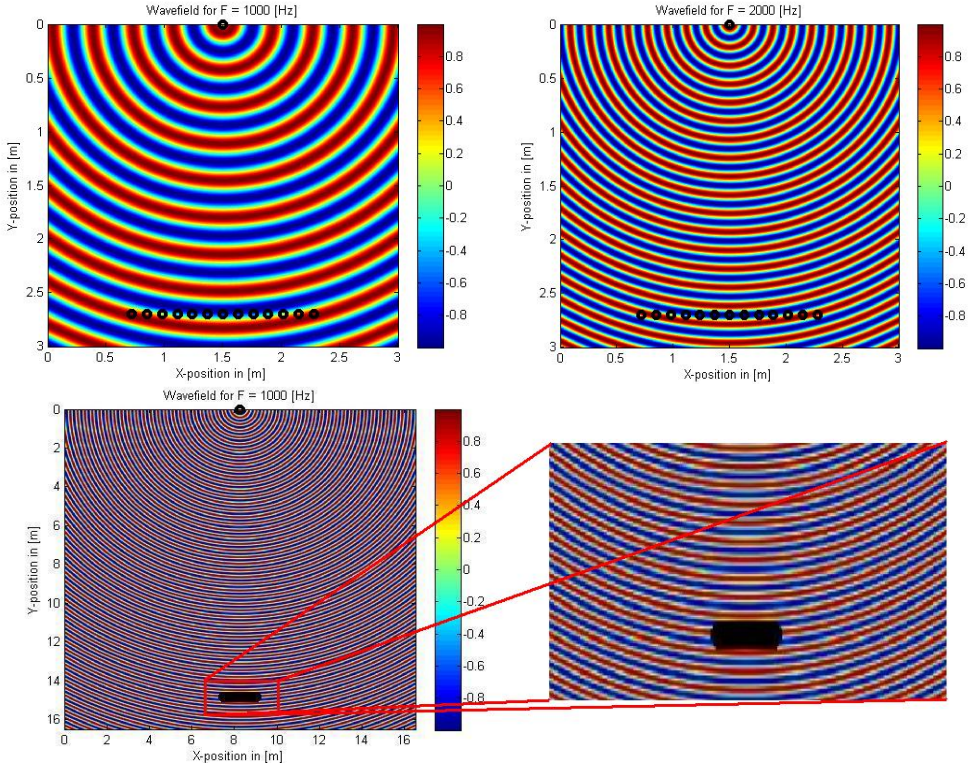


Figure 2-4 Wavefield of sine source. Upper left: F = 1000 [Hz], distance = 2,75 [m]. Upper right: F = 2000 [Hz], distance = 2,75 [m]. Below: F = 1000 [Hz], distance = 15 [m].

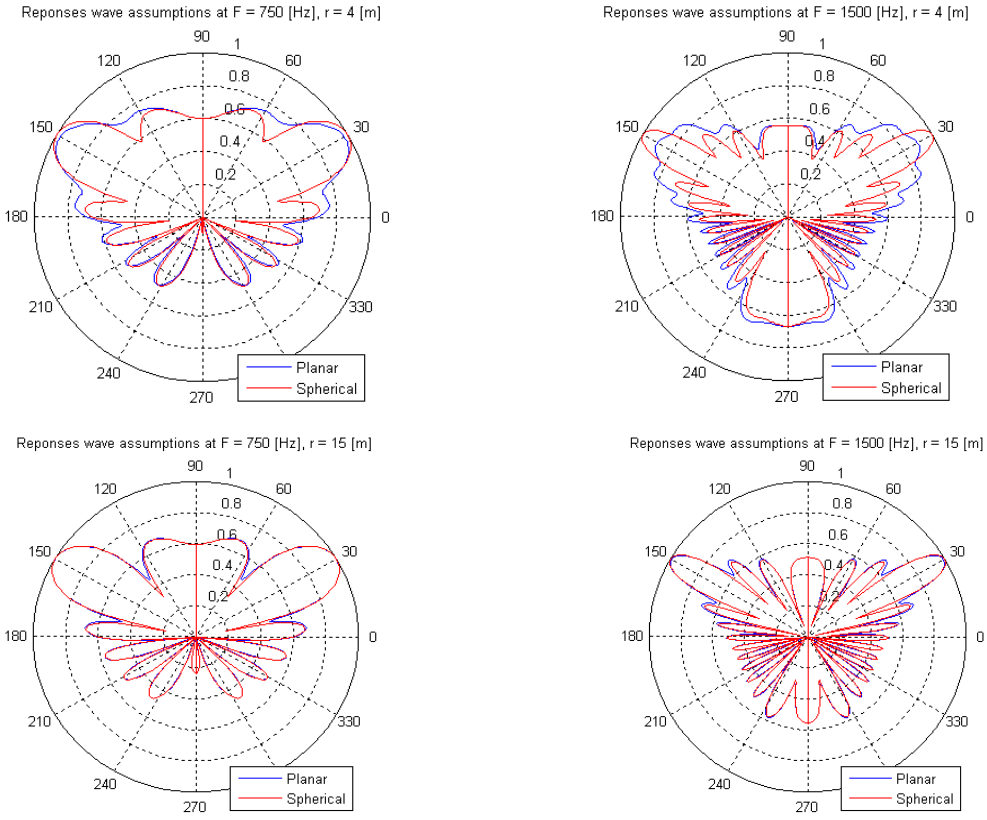


Figure 2-5 Responses different wave assumptions. Upper left: F = 750 [Hz], distance = 4 [m]. Upper right: F = 1500 [Hz], distance = 4 [m]. Lower left: F = 750 [Hz], distance = 15 [m]. Lower right: F = 1500 [Hz], distance = 15 [m].

2.1.1.3 Narrowband assumption

As discussed in paragraph *2.1.1 Delay and Sum*, a source signal can be spatially filtered by computing the weight coefficients based on phase shifting the sensor outputs. This phase shift is dependent on the frequency that is considered. A weight vector designed for signals with a frequency of, for example, 500 [Hz] will apply different, incorrect phase shifts for signals with a frequency of 2000 [Hz]. This would mean that for every frequency a different weight vector has to be computed. However, this would tremendously increase the computational complexity of the system and therefore is undoable. Still, we can limit the number of weight vectors by using one weight vector for a small range of frequencies; a narrowband frequency bin such that the introduced phase error is acceptable.

To visualize this, again a plot is given (Figure 2-6); this time of the spatial response of an array with a source present at an angle of 40 [°] which exerts a signal of a particular frequency F_l. One spatial response is computed using a weight vector designed for the frequency F_l. The other response is computed using a weight vector designed for the centre frequency F_C of the frequency bin to which the frequency F_l belongs, with the relation:

$$F_l = F_c - F_c \times p \qquad \text{[Hz]} \tag{2.9}$$

With p begin a scalar value representing the proportion of the centre frequency that defines the bin width. Plots are given for p-values of 0.05 and 0.12.

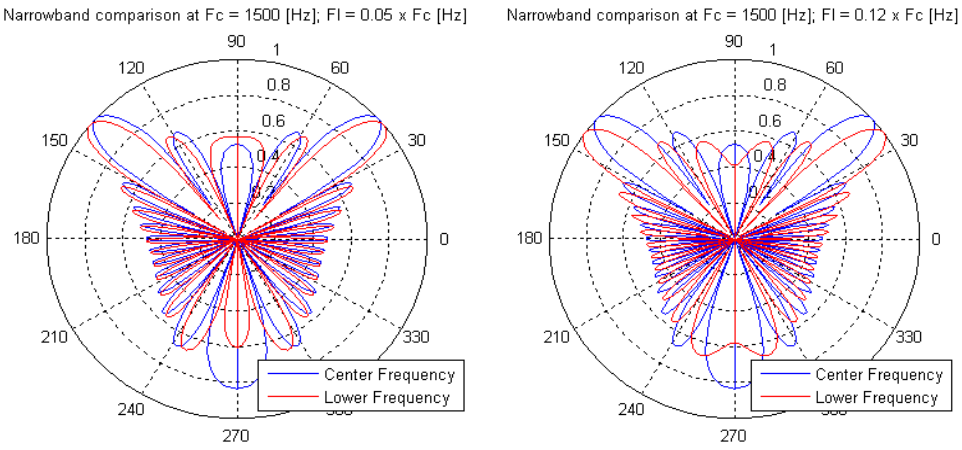


Figure 2-6 Influence of frequency bin widths on dislocated spatial response. Left: p = 0.05. Right: p = 0.10.

It is clear the look direction is dislocated with a few degrees when using the same steering vector for multiple frequencies. This angle shift of the main beam of a signal with a centre frequency proportion of p, can be calculated directly by using the formula for the delay vector of the signal. Recall Eq. (2.4) to (2.8). In the formula for ς, the value for the source/target frequency, F_T, which represents F_l in (2.9), is replaced by $(1-p)F_c$. By equating the steering phase shift and the source phase shift, the angle difference of the main beam for the frequency F_l can be determined:

$$\xi = \frac{2\pi F_c d}{c}\sin(\theta_0) = \frac{2\pi(1-p)F_c d}{c}\sin(\theta_s) = \varsigma \tag{2.10}$$

This leads to an angle difference for the main beam of:

$$\Delta\theta = \arcsin\left(\frac{1}{1-p}\sin\theta_0\right) - \theta_0 \qquad [°] \tag{2.11}$$

So the angle difference depends only on the focus angle and the proportion between the bin limits and the centre frequency of the bin. In practice this means that the absolute frequency bin width can be enlarged at higher frequencies. Furthermore, for smaller focus angles, wider and thus fewer bins can be used to calculate the weight vectors spanning the whole frequency range we would like to consider. This can save calculations when beamforming is performed in frequency domain.

Finally, to visualize the above, a plot is given that relates the proportion p to the beam angle deviation in [°] for focus angles of 0 to 90 [°].

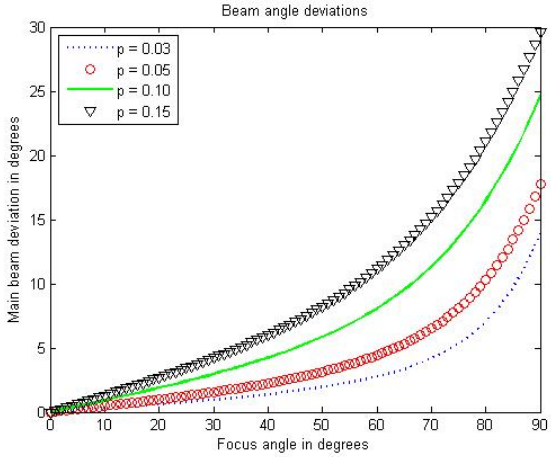


Figure 2-7 Angle deviations of main beam related to p.

2.1.2 *Fixed windowing*

Opposed to the conventional DAS beamformer, methods exist to assign a value to the amplitude weight α_n of sensor n. Some methods define weights dependent on frequency. Others, like the weight windows discussed in this paragraph, are frequency-invariant. These fixed windows all have their advantages and disadvantages concerning main lobe width, sidelobe height etc. There is no window that possesses the best properties for all criteria. Therefore it is good practice to formulate some criteria to which the frequency response should be optimized for this particular application.

As the project description already indicates, the system will be deployed in crowded spaces with interferers from many unknown directions. To this extent, the use of a spatial filter, possessing a frequency response containing very large attenuation at specific points with the additional disadvantage of having one or more larger sidelobes, is useless. As the position of strong interferers is unknown, noise preferably should be suppressed evenly in all directions, which makes the presence of deep nulls at random look directions useless. These degrees of filtering freedom can come to better use for overall suppression.

Furthermore, the desire for a small main lobe width still is present to achieve high spatial resolution. But as mentioned before, every optimization comes with its compromises, which requires finding a balance between these two properties.

Taking above requirements into consideration, three windowing methods are selected to compare with each other and with the DAS beamforming method. Figure 2-8 shows the frequency responses of a simulated beamformer using a sinc-, Dolph-Chebyshev- [19], Kaiser-Bessel- [23] and rectangular DAS-window at two frequencies. To obtain a better overview, the responses are plotted in a square plot instead of the circular, polar plot.

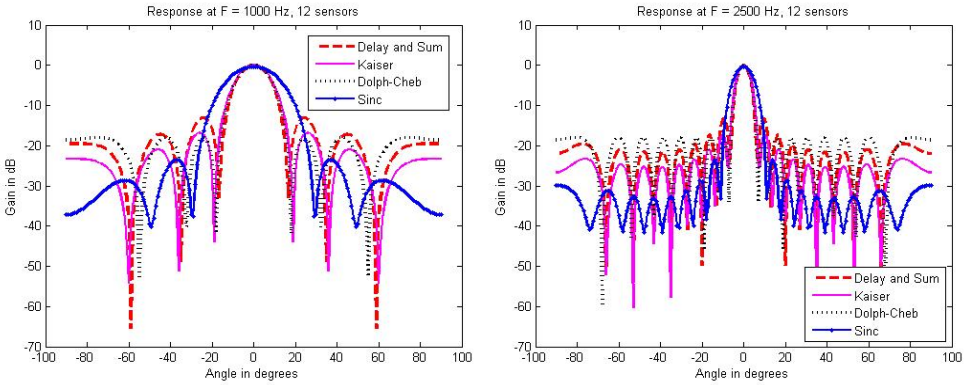


Figure 2-8 Reponses of different fixed window beamformers. Left: F=1000 [Hz]. Right: F = 2500 [Hz]. (12 sensors, look angle = 0 [°]).

It is clear from the plots that applying the sinc-window does result in stronger attenuation of sources from other directions but also leads to a much wider main lobe. The Kaiser-window performs better than the rectangular DAS-window relating to lower sidelobes while having only a slightly larger main lobe width. The Dolph-Chebyshev-window, finally, provides sidelobes with equal height and also only a slightly broadened main lobe, making it useful in this particular beamforming application.

2.2 Dynamic beamforming

The discussed methods of static beamforming do not take into account the sound environment. This makes them on the one hand computationally non-intensive because all spatial filter coefficients can be computed in advance. On the other hand, they lack performance referring to spatial resolution at lower frequencies due to a wider main lobe width and, in the presence of strong interferers, due to a constant spatial response. Dynamic beamforming aims to overcome both these shortcomings by using the sensor outputs in the computation of the weight vectors and therefore obtain time-varying (dynamic) filter responses. Because of the additional computational complexity involved with these methods; the necessity of a large number of beams and the real-time requirement for the system, from now on a rough estimation of the computational complexity will be incorporated at the description of each method. This will be done by indicating the required number of real additions, multiplications and divisions per frame of samples.

2.2.1 Minimum Variance Distortionless Response

The proposed method(s) for frequency-invariant weight vector computation all possess an important disadvantage: the frequency responses at low frequencies (<1000 [Hz][1]) experience significant deterioration compared to higher frequencies. A method that tries to intercept this problem is the Minimum Variance Distortionless Response (MVDR) beamforming method, also known as superdirective beamforming. This paragraph will

[1] Frequency related to linear array structure used in the examples.

describe two types of MVDR beamforming: (input-based) dynamic MVDR (DMVDR) and (non-input-based) static MVDR (SMVDR). Though this paragraph is about dynamic beamforming, SMVDR is discussed to point out the strength of DMVDR and to use as a step-up to a robust version of DMVDR. Furthermore, in the case of processing limitations it could be used as a back-up method.

2.2.1.1 *Dynamic MVDR*

The MVDR beamformer aims to minimize the total output power, while maintaining unit gain in the look direction, leading to a constrained minimization problem. First, the equation for the weight coefficients will be derived in this paragraph after which the method will be compared to the conventional DAS beamforming method by means of simulating responses in frequency domain.

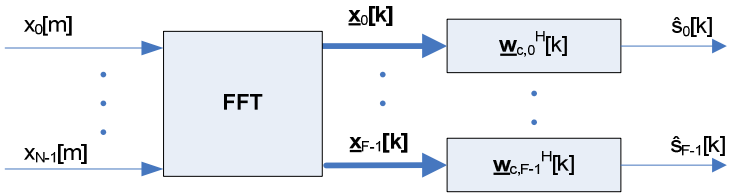


Figure 2-9 System representation of the beamforming filter.

To derive the expression to determine the constrained weight vector $\underline{\mathbf{w}}_c$, first consider Figure 2-9 which is a simple representation of our system. In this figure, $\hat{s}_f[k]$ represents the estimated signal of frame *k* and frequency index *f* from the position that is focused on. We would like to minimize the average output power in order to reduce the contributions of interferers and noise from other directions. To keep a better overview, frame and frequency indices are discarded from now on, without loss of generality. The total output power per frequency bin *f* and frame *k* is defined as:

$$P = E\left\{|\hat{s}|^2\right\} = E\left\{\left|\underline{\mathbf{w}}_c{}^H \underline{\mathbf{x}}\right|^2\right\} = E\left\{(\underline{\mathbf{w}}_c{}^H \underline{\mathbf{x}})(\underline{\mathbf{x}}^H \underline{\mathbf{w}}_c)\right\} = \underline{\mathbf{w}}_c{}^H \mathbf{R}_{\mathbf{x}\mathbf{x}^H} \underline{\mathbf{w}} \tag{2.12}$$

When minimizing this equation by setting the derivative equal to zero, we would obtain a vector $\underline{\mathbf{w}}_c$ containing only zeros, which in essence is correct as the output power is minimized. However, as mentioned before, MVDR beamforming is designated to preserve unit gain in the look direction. Hence, a constraint has to be included in the minimization problem.

Let the constraint angle vector be defined as [3, 18]
$\underline{\mathbf{c}} = \left(1, e^{-j\xi_0} \cdots e^{-j(N-1)\xi_0}\right)^T$ containing the phase shifts per sensor for the desired look direction. To obtain unit gain in the look direction, the constraint can be defined as [3]:

$$\underline{\mathbf{c}}^H \underline{\mathbf{w}}_c = 1 \tag{2.13}$$

Leading to the final definition of the constrained minimization problem [18, 24]:

$$\min_{\underline{\mathbf{w}}_c}\{P\} = \min_{\underline{\mathbf{w}}_c}\left\{\underline{\mathbf{w}}_c{}^H \mathbf{R}_{\mathbf{x}\mathbf{x}^H} \underline{\mathbf{w}}_c\right\} \quad \text{subject to } \underline{\mathbf{c}}^H \underline{\mathbf{w}}_c = 1 \tag{2.14}$$

To solve the above equation, Lagrange multipliers are used. Lagrange multipliers are designated to perform a minimization, subject to one or several constraints. When including the constraints in the equation for the output power, we obtain [24]:

$$P = \underline{\mathbf{w}}_c{}^H \mathbf{R}_{\mathbf{xx}^H} \underline{\mathbf{w}}_c + \lambda(\underline{\mathbf{c}}^H \underline{\mathbf{w}}_c - 1) \tag{2.15}$$

In which λ is the concerning Lagrange multiplier. When setting the derivative of (2.15) to $\underline{\mathbf{w}}_c$ equal to zero, the optimal, constrained weight vector is obtained:

$$\frac{dP}{d\underline{\mathbf{w}}_c} = \mathbf{R}_{\mathbf{xx}^H} \underline{\mathbf{w}}_c + \underline{\mathbf{c}}\lambda = \underline{\mathbf{0}} \tag{2.16}$$

$$\underline{\mathbf{w}}_c = -\mathbf{R}^{-1}_{\mathbf{xx}^H} \underline{\mathbf{c}}\lambda \tag{2.17}$$

Using Eq. (2.13), both sides of Eq. (2.17) can be multiplied on the left with $\underline{\mathbf{c}}^H$ after which the left-hand side can be set equal to 1. The equation for λ then is obtained from Eq. (2.17). The final optimal value for the constrained weight vector is determined by substitution of λ in Eq. (2.17) [3, 18, 24, 25]:

$$\underline{\mathbf{w}}_c = \mathbf{R}_{\mathbf{xx}^H}{}^{-1} \underline{\mathbf{c}}(\underline{\mathbf{c}}^H \mathbf{R}^{-1}_{\mathbf{xx}^H} \underline{\mathbf{c}})^{-1} \tag{2.18}$$

What is visible immediately is that the above equation needs the inverse of the autocorrelation matrix of the input signal. This involves two important issues. (i) The autocorrelation matrix must be full rank in order for inversion to be possible. As the input signal is not deterministic, an invertible autocorrelation matrix is not per definition guaranteed. (ii) Furthermore, one of the limitations of the beamforming array is that the used microphones are low-cost and unstable. The matrix inversion makes the spatial filter very sensitive to small fluctuations of the output signals of the microphones. This has a negative influence on the constructed output signal of the beamformer.

These two issues may be a burden for making use of this autocorrelation matrix. Further in this report this possibility will be reconsidered. For now, an alternative, *static* solution of MVDR (SMVDR) will be described.

2.2.1.2 Static MVDR

With static MVDR (SMVDR), the autocorrelation matrix is constructed by making use of the noise coherence function of the sound field, which is defined as [3]:

$$\Gamma_{\mathbf{VV}} = \begin{pmatrix} 1 & \Gamma_{V_0V_1} & \Gamma_{V_0V_2} & \cdots & \Gamma_{V_0V_{N-1}} \\ \Gamma_{V_1V_0} & 1 & \Gamma_{V_1V_2} & \ddots & \Gamma_{V_1V_{N-1}} \\ \Gamma_{V_2V_0} & \Gamma_{V_2V_1} & \ddots & \ddots & \Gamma_{V_2V_{N-1}} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \Gamma_{V_{N-1}V_0} & \Gamma_{V_{N-1}V_1} & \Gamma_{V_{N-1}V_2} & \cdots & 1 \end{pmatrix} \tag{2.19}$$

In which the noise coherence between sensors *n* and *p* is defined as [3, 25]:

$$\Gamma_{V_nV_p}(e^{j\Omega}) = \frac{\phi_{V_nV_p}(e^{j\Omega})}{\sqrt{\phi_{V_nV_n}(e^{j\Omega})\phi_{V_pV_p}(e^{j\Omega})}} \tag{2.20}$$

$\phi_{V_nV_p}(e^{j\Omega})$ represents the Power Spectral Density (PSD) of the noise coherence between sensors n and p for $\Omega = [0\text{-}2\pi]$. As the pure noise signal is not present by itself and the accuracy of the microphones is still involved, instead of using Eq. (2.20), an assumption can be made about the environment. Assume measurements are performed in a diffuse noise field: all microphones receive equal-variance and random-phase noise signals from all directions [2]. The noise coherence between two sensors then can be written as [3, 4, 25]:

$$\Gamma_{V_nV_p}(e^{j\Omega})\Big|_{Diffuse} = \frac{\sin(\Omega F_s l_{n,p}/c)}{\Omega F_s l_{n,p}/c} \tag{2.21}$$

In which $l_{n,p}$ is the distance between the sensors n and p in [m], c the speed of sound in [m/s] and $F = \Omega F_s$ the frequency of interest in [Hz]. The definition of the weight vector then becomes [3, 4, 25]:

$$\underline{\mathbf{w}}_c = \Gamma_{\mathbf{VV}}{}^{-1}\underline{\mathbf{c}}(\underline{\mathbf{c}}^H\Gamma_{\mathbf{VV}}{}^{-1}\underline{\mathbf{c}})^{-1} \tag{2.22}$$

The performance of the SMVDR beamformer can be compared with that of the traditional DAS beamformer, observing multiple different characteristics. One of them is the generated beampattern for a particular look direction. Figure 2-10 shows the responses for two frequencies for both methods simulating one (linear) branch of the used array, with 12 sensors placed as depicted in appendix A.

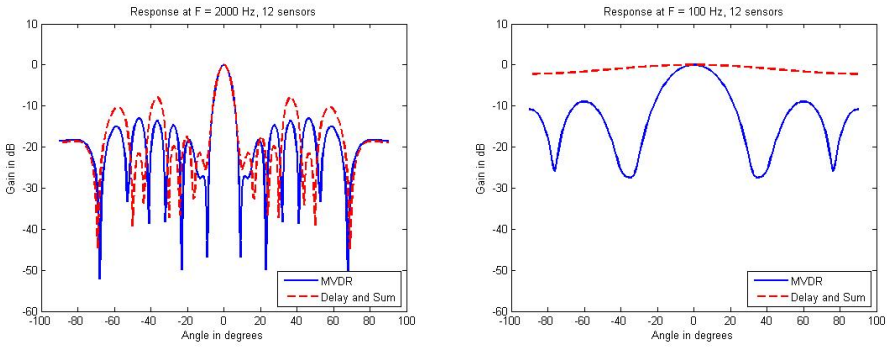


Figure 2-10 Plots of reponses SMVDR and DS beamformer. Left: F=2000 [Hz]. Right: F=100 [Hz]. (12 sensors, look angle = 0 [°])

It is clear that the SMVDR beamformer provides a better spatial response compared to the DAS beamformer: the highest sidelobes are attenuated more and especially at lower frequencies it keeps its performance, though still deteriorated.

To gain a better overview of the advantages of the SMVDR beamformer over a continuous frequency range, Figure 2-11 provides more insight. The spatial responses for all frequencies in the range of F = 0-3000 [Hz] are computed and plotted with the colour indicating the attenuation.

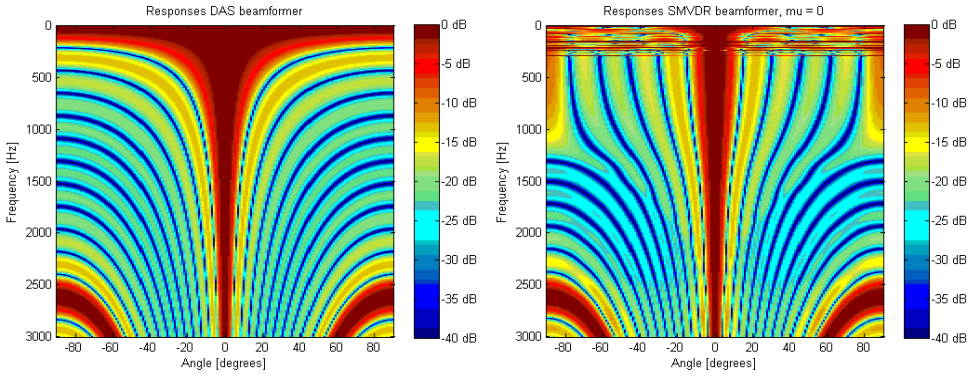


Figure 2-11 Plots of attenuation over whole frequency range. Left: DAS beamformer. Right: SMVDR beamformer. (12 sensors, viewing angle = 0 [°]).

From the plots it is clear that the SMVDR beamformer has a better resolution for frequencies below 1000 [Hz]. However, for frequencies below approximately 400 [Hz], a highly distorted response is observable. The explanation for this lies in the inversion of the coherence matrix: for low frequencies the correlation between the sensors becomes higher. This makes the columns of the coherence matrix become (almost) linearly dependent, leading to an (almost) non-invertible matrix. As a result of this, the entries of the inverse - and therewith the weight coefficients - take on extremely high values, leading to extreme amplifications of spatially uncorrelated signals, i.e. white noise. A measure for this particular property is the White Noise Gain (WNG) which is shown for both beamforming methods (Figure 2-12) and is defined as [26]:

$$WNG(e^{j\Omega}) = 10\log_{10}\left(\frac{\left|\underline{\mathbf{w}}_c{}^H\,\underline{\mathbf{c}}\right|^2}{\underline{\mathbf{w}}_c{}^H\,\underline{\mathbf{w}}_c}\right) \quad [\text{dB}] \tag{2.23}$$

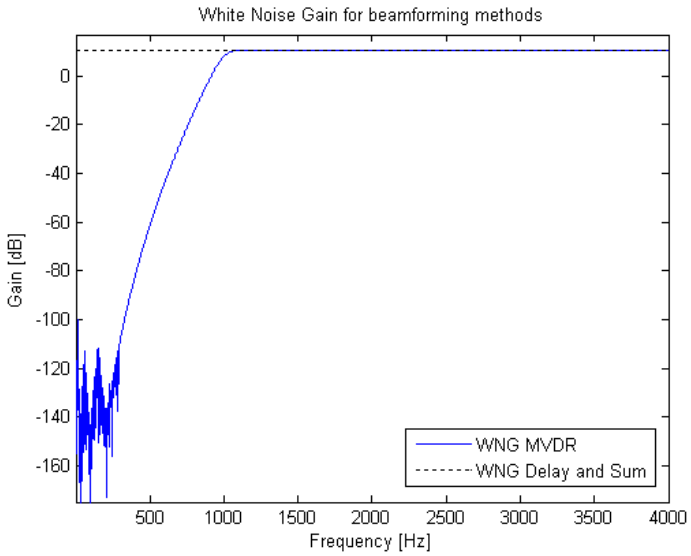


Figure 2-12 White Noise Gain for DAS and MVDR.

The above figure confirms the expectation: frequencies up to 400 [Hz] and even higher experience a low WNG. Note that the DAS beamformer has a constant WNG due to the fact that sensor weights are uniform and independent of frequency.

A measure to partly intercept this problem is adding a constant μ to the diagonal of the noise coherence matrix. In this way, the columns (and rows) reduce their interdependency which results in lower valued entries in the inverse matrix and along with that: lower valued weight coefficients. The equation for the constrained weight vector then becomes [3, 25]:

$$\underline{\mathbf{w}}_c = (\Gamma_{\mathbf{vv}} + \mu\mathbf{I})^{-1}\underline{\mathbf{c}}(\underline{\mathbf{c}}^H(\Gamma_{\mathbf{vv}} + \mu\mathbf{I})\underline{\mathbf{c}})^{-1} \tag{2.24}$$

The disadvantage of this adjustment is the degradation of the performance of the beamformer. This can be evaluated by assessing the Directivity Index (DI) which is defined as [3]:

$$DI(e^{j\Omega}) = 10\log_{10}\left(\frac{\left|\underline{\mathbf{w}}_c{}^H\underline{\mathbf{c}}\right|^2}{\underline{\mathbf{w}}_c{}^H\Gamma_{\mathbf{vv}}\big|_{Diffuse}\underline{\mathbf{w}}_c}\right) \quad [\text{dB}] \tag{2.25}$$

Figure 2-13 shows the relation between the constant μ and the WNG and DI.
What is clear is that an improvement of the WNG leads to a deterioration of the DI. A balance between these two properties gives a reasonable solution.

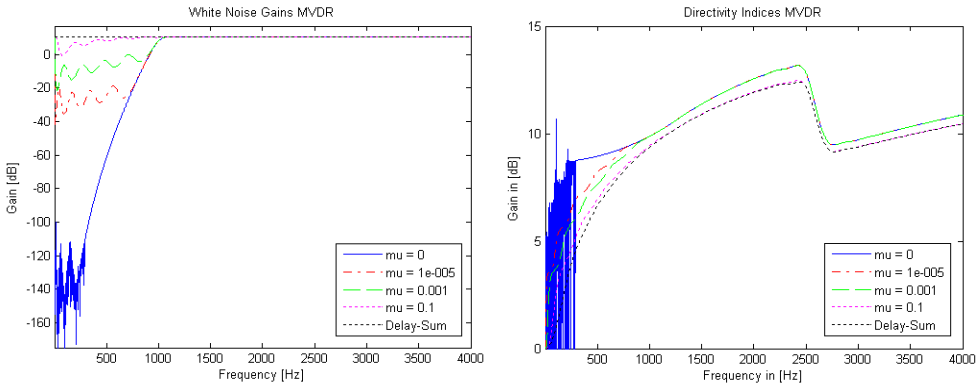


Figure 2-13 Relation between μ and White Noise Gain (l) and Directivity Index (r) SMVDR.

To end this discussion, Figure 2-14 shows the full response of the SMVDR beamformer for μ=0.001.

Though the SMVDR beamformer is an improvement referring to the conventional beamforming methods, the use of an assumption of the noise field possesses some important disadvantages for this particular application. As the final system will be applied in crowded spaces, the assumption of a diffuse noise field will not always hold. A substantial amount of interferers may be present, which is not taken into account in weight vector computation. To illustrate this, Figure 2-15 shows plots of spatial filter responses of SMVDR and DMVDR beamformers, when equal-strength interferers are present at -20 and 60 [°] and when creating a beam for an angle of 20 [°]. In all three plots the relation in strength between interference and noise variance differ.

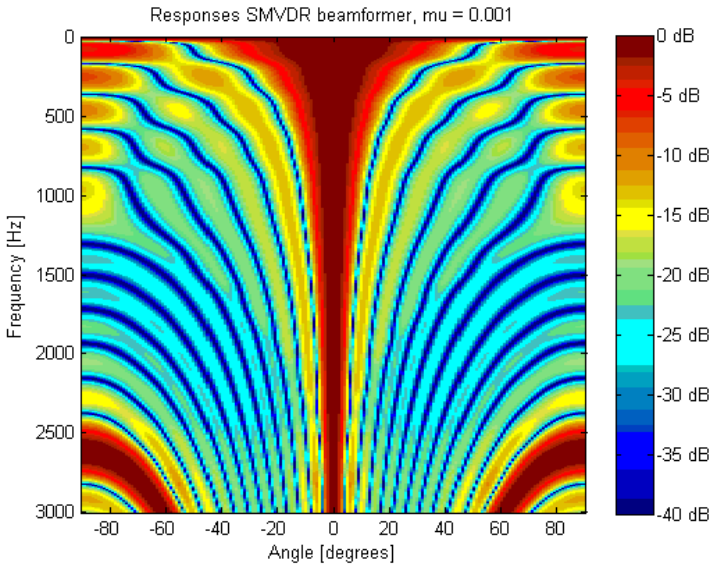


Figure 2-14 Plot of SMVDR response over whole frequency range (12 sensors, μ = 0.001, look direction = 0 [°]).

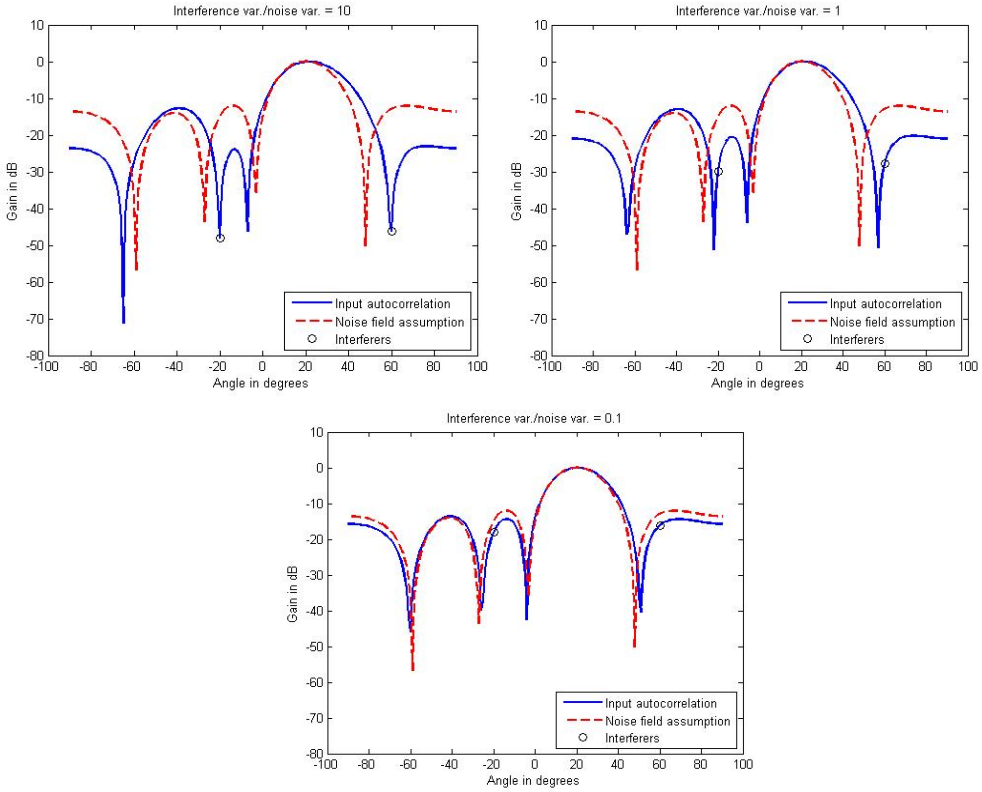


Figure 2-15 Plots of beamresponses for noise field assumption and actual input autocorrelation with decreasing interference variance to noise variance relations.

When using a noise field assumption, the spatial filter is designed solely to suppress white noise, regardless of the presence of interferers. When white noise is present as the main interference source, like in the last plot, this is a reasonable solution. However, when one or more strong interferers distort the source signal, the obtained results worsen significantly.

2.2.1.3 *Robust Dynamic MVDR*

The results of the comparison between the static and input-based MVDR beamforming methods indicate the need for the input-based approach. It may be possible to cope with its pitfalls mentioned earlier: non-deterministic input signals and the combination of noisy sensors. Namely, the same scheme for increasing white noise gain when computing filter coefficients by making a noise field assumption as discussed before, can be applied to the problems of input-based MVDR beamforming as well. Loading the main diagonal with a constant, increases invertibility regardless of sensor inputs and therewith also decreases the influence of interdeviant sensor noise levels due to its generation of lower absolute filter coefficients. Figure 2-16 shows the improvement of input-based robust DMVDR beamforming as opposed to conventional DAS beamforming. As opposed to Figure 2-15, now a complete, linear grid scan of 180 [°] is performed for locating sources. The left figure shows the measured signal strength at each point when one source is located at -20 [°]. In the right figure, an extra, coherent source is added at 40 [°].

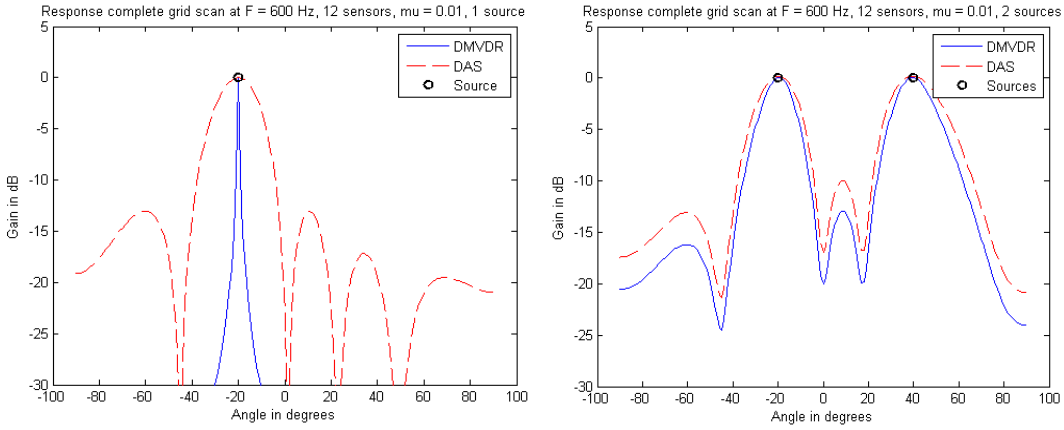


Figure 2-16 Responses DMVDR and DAS with 1 source (left) and 2 coherent sources (right).

It is clear DMVDR outperforms DAS, though it must be admitted that in the case of multiple coherent sources, its performance also degrades.

2.2.1.4 *Computational complexity*

To finish the description of this method, an estimate will be made on the computational cost of DMVDR beamforming.

First, the final equation for the weight vector can be combined with its multiplication with the frequency representation of the sensor signals. By doing this, we can see the complexity can be reduced a bit more. As indicated in the problem description, we only need the power of the beamformed signals per frequency bin.

Using Eq. (2.18), the output power can be written as:

$$P = \left(\underline{\mathbf{w}}_c{}^H \underline{\mathbf{x}}\right)\left(\underline{\mathbf{w}}_c{}^H \underline{\mathbf{x}}\right)^H = \frac{\underline{\mathbf{c}}^H \mathbf{R}_{\mathbf{xx}^H}{}^{-1} \underline{\mathbf{x}}\underline{\mathbf{x}}^H \mathbf{R}_{\mathbf{xx}^H}{}^{-1} \underline{\mathbf{c}}}{\left(\underline{\mathbf{c}}^H \mathbf{R}^{-1}_{\mathbf{xx}^H} \underline{\mathbf{c}}\right)\left(\underline{\mathbf{c}}^H \mathbf{R}^{-1}_{\mathbf{xx}^H} \underline{\mathbf{c}}\right)} \tag{2.26}$$

This can be rewritten to:

$$P = \frac{\underline{\mathbf{c}}^H \mathbf{R}_{\mathbf{xx}^H}{}^{-1} \underline{\mathbf{c}}}{\left(\underline{\mathbf{c}}^H \mathbf{R}^{-1}_{\mathbf{xx}^H} \underline{\mathbf{c}}\right)\left(\underline{\mathbf{c}}^H \mathbf{R}^{-1}_{\mathbf{xx}^H} \underline{\mathbf{c}}\right)} = \frac{1}{\underline{\mathbf{c}}^H \mathbf{R}^{-1}_{\mathbf{xx}^H} \underline{\mathbf{c}}} \tag{2.27}$$

The required number of operations per type of operation for a frame of *M* samples is given in Table 2-1 in which *(R)* and *(C)* stand for a real and a complex operation respectively. It is assumed that an M-point Fast Fourier Transform (FFT) requires $M \log_2(M)$ real multiplications and additions when using real input data [9]. An NxN-matrix inversion is assumed to require approximately $N^3 + N^2$ complex multiplications and additions and N^2 complex divisions when applying Gauss-Jordan elimination (see appendix B).

Operation	**MUL**	**ADD**	**DIV**
FFT	$NM\log_2(M)$ *(R)*	$NM\log_2(M)$ *(R)*	-
Inverse	$F(N^3+N^2)$ *(C)*	$F(N^3+N^2)$ *(C)*	FN^2 *(C)*
Output	$FB(N^2+N)$ *(C)*	$FB(N^2+N)$ *(C)*	FB *(C)*
Total *(R)*	$\mathbf{F[4(N^3+N^2)+8N^2+4B(N^2+N)+8B]+NM\log_2(M)}$	$\mathbf{F[4(N^3+N^2)+3N^2+4B(N^2+N)+3B]+NM\log_2(M)}$	$\mathbf{F(B+N^2)}$

Table 2-1 Number of operations for computing MVDR coefficients.

F represents the number of frequency bands, *B* the number of beams and *N* the number of sensors. To strip the operations further down to basic operations, a complex multiplication consists of four real multiplications and two real additions, when keeping the real and imaginary part separated, according to:

$$(a+bi)(c+di) = (ac-bd)+(ad+bc)i \tag{2.28}$$

A complex division, on its turn, corresponds to eight real multiplications, three real additions and one real division according to:

$$\frac{a+bi}{c+di} = \frac{(a+bi)\overline{c+di}}{(c+di)\overline{c+di}} = \frac{(ac+bd)+i(bc-ad)}{c^2+d^2} \tag{2.29}$$

A complex addition, finally, corresponds to two real additions. Table 2-1 also lists the final number of real operations.

2.2.2 *Constrained Maximum Signal to Interference Plus Noise Ratio*

After having discussed a method that is derived based on the criterion to obtain minimal output power (MVDR), constrained to obtain unit gain in the look direction, this method is developed in order to obtain a Maximum Signal to Interference Plus Noise Ratio (MSINR).

Consider the input signal vector $\underline{\mathbf{x}}$ of a beamformer, ignoring frame and frequency indices for surveyability, which consists of the original source signal, interfering sources and noise:

$$\underline{\mathbf{x}} = \underline{\mathbf{d}}s + \underline{\mathbf{i}} + \underline{\mathbf{n}} \tag{2.30}$$

With $\underline{\mathbf{d}}$ containing the transfer functions between the source and each sensor, $\underline{\mathbf{i}}$ being the vector containing the interferences from other signals on each sensor and $\underline{\mathbf{n}}$ containing the noise at each sensor, all applying for one frequency bin. Applying the weight vector of the beamformer on its input vector results in the following output signal:

$$y = \underline{\mathbf{w}}_c{}^H \underline{\mathbf{x}} = \underline{\mathbf{w}}_c{}^H \underline{\mathbf{d}} s + \underline{\mathbf{w}}_c{}^H (\underline{\mathbf{i}} + \underline{\mathbf{n}}) \tag{2.31}$$

The Signal to Interference Plus Noise Ratio (SINR) then can be defined as [27]:

$$SINR = E\left\{ \left| \frac{\underline{\mathbf{w}}_c{}^H \underline{\mathbf{d}} s}{\underline{\mathbf{w}}_c{}^H (\underline{\mathbf{i}} + \underline{\mathbf{n}})} \right|^2 \right\} = E\{|s|^2\} \frac{\left| \underline{\mathbf{w}}_c{}^H \underline{\mathbf{d}} \right|^2}{E\{\left| \underline{\mathbf{w}}_c{}^H (\underline{\mathbf{i}} + \underline{\mathbf{n}}) \right|^2\}} = S^2 \frac{\left| \underline{\mathbf{w}}_c{}^H \underline{\mathbf{d}} \right|^2}{E\{\left| \underline{\mathbf{w}}_c{}^H (\underline{\mathbf{i}} + \underline{\mathbf{n}}) \right|\}} \tag{2.32}$$

With S^2 representing the average signal power. As the nominator remains constant, the SINR can be maximized by minimizing the denominator, while at the same time preserving the constraint in the look direction, leading to the following minimization problem:

$$\underline{\mathbf{w}}_c = \max_{\underline{\mathbf{w}}_c} SINR = \min_{\underline{\mathbf{w}}_c} E\{\left| \underline{\mathbf{w}}_c{}^H (\underline{\mathbf{i}} + \underline{\mathbf{n}}) \right|^2\} \quad \text{subject to } \underline{\mathbf{c}}^H \underline{\mathbf{w}} = 1 \tag{2.33}$$

When again solving this problem using Lagrange multipliers the solution for the optimal weight vector to minimize the SINR becomes [27]:

$$\underline{\mathbf{w}}_c = \mathbf{R}_{\mathbf{uu}^H}{}^{-1} \underline{\mathbf{c}} (\underline{\mathbf{c}}^H \mathbf{R}_{\mathbf{uu}^H}^{-1} \underline{\mathbf{c}})^{-1} \tag{2.34}$$

In which $\boldsymbol{R}_{\boldsymbol{uu}}{}^H$ represents the autocorrelation matrix of the interference plus noise. A direct issue is that this autocorrelation matrix is not present and therefore making this method impractical for implementation purposes.

Two methods have been discussed and derived now. Both of them try to solve a different minimization problem subject to a constraint. The resulting formulas for the weight coefficients look similar; the only difference is the use of a different autocorrelation matrix in the formula for the weight coefficients. [28] provides a proof of the equivalence of these formulas. The use of the autocorrelation matrix of the interference plus noise in the CMSINR method makes it very impractical. The DMVDR method provides an easier implementable solution. Therefore CMSINR will be discarded from now on.

2.2.3 Linear Constraint Minimum Variance

The MVDR beamformer discussed earlier, is a special form of the method known as Linear Constraint Minimum Variance (LCMV). This method aims to minimize the average output power subject to a *number* of constraints. These constraints are formulated for a specific spatial angle and define an exact gain in that direction. In this way, it is possible to create nulls (very high attenuation) at directions from which unwanted sources are interfering. On the other hand it is possible to obtain unit gain in one or more directions. The difference with regard to the MVDR beamformer is that multiple constraints can be postulated instead of only one for the look direction.

Consider a number of M constraint direction vectors that together form the $N \times M$ matrix $\mathbf{C} = \left(\underline{\mathbf{c}}_0, \underline{\mathbf{c}}_1 \cdots \underline{\mathbf{c}}_{M-1}\right)$ with [18] $\underline{\mathbf{c}}_m = \left(1, e^{-j\xi_m} \cdots e^{-j(N-1)\xi_m}\right)^T$ and

$\xi_m = \frac{2\pi F_c d}{c}\sin(\theta_m)$. θ_m represents the angle for which we would like to apply a constraint. The constraint equation then can be written as [18]:

$$\mathbf{C}^H \underline{\mathbf{w}}_c = \underline{\mathbf{f}} \tag{2.35}$$

With $\underline{\mathbf{f}}$ being the *M*-vector containing the desired gain value for each particular direction, from now on called the response vector. The final constrained optimization problem that has to be solved conclusively becomes:

$$\min_{\underline{\mathbf{w}}_c}\{P\} = \min_{\underline{\mathbf{w}}_c}\left\{\underline{\mathbf{w}}_c{}^H \mathbf{R}_{\mathbf{xx}^H} \underline{\mathbf{w}}_c\right\} \quad \text{subject to } \mathbf{C}^H \underline{\mathbf{w}}_c = \underline{\mathbf{f}} \tag{2.36}$$

Solving this problem in a similar way as for the MVDR beamformer, the equation for the output power becomes:

$$P = \underline{\mathbf{w}}_c{}^H \mathbf{R}_{\mathbf{xx}^H} \underline{\mathbf{w}}_c + \underline{\lambda}^H (\mathbf{C}^H \underline{\mathbf{w}}_c - \underline{\mathbf{f}}) \tag{2.37}$$

When setting the derivative of (2.37) to $\underline{\mathbf{w}}_c$ equal to zero again and solving it, the final equation for the constrained weight vector becomes [18, 29]:

$$\underline{\mathbf{w}}_c = \mathbf{R}^{-1}_{\mathbf{xx}^H} \mathbf{C}(\mathbf{C}^H \mathbf{R}^{-1}_{\mathbf{xx}^H} \mathbf{C})^{-1} \underline{\mathbf{f}} \tag{2.38}$$

The constraint matrix of the LCMV beamformer provides the ability to impose constraints in multiple directions, depending on the number of microphones. These directions, however, are not known in advance for this particular application. This could require active interference tracking. However, with relation to memory, required amount of hardware and processing time to achieve this in real-time for all possible directions of a scanned surface by means of a portable implementation, this seems unachievable. Furthermore, time limits the possibility of an extensive implementation process. As mentioned in the requirement which the beamformer should meet and referring to the above, these degrees of freedom in the LCMV beamformer can come to better use for overall suppression of interfering signals like illustrated for the DMVDR beamformer, using actual input signals. Computational complexity therefore will not be considered here. However, it must be emphasized that in a future combination with other systems, the other applications might be used to advantage in providing interference locations.

The next paragraph will discuss a slightly different method than DMVDR/LCMV that tries to reduce the complexity of the DMVDR beamformer.

2.2.4 Generalized Sidelobe Canceller

A disadvantage of the LCMV/MVDR beamformer is that on every new computation of the weight vector, the full, constrained formula has to be considered, leading to a high computational complexity as indicated in Table 2-1. An easier way to recompute weight coefficients, provided by the Generalized Sidelobe Canceller (GSC), is outlined in this paragraph. Goal of the proposal of this method is trying to reduce computational complexity by finding an easier way of computing the filtering coefficients.

A GSC turns a constrained minimization problem, like that of the LCMV/MVDR beamformer, into an unconstrained one by dividing the constraint matrix in a

constrained and an unconstrained part. Consider Figure 2-17, where the system representation is given for the problem that is addressed. In this figure, $\mathbf{B}_f$ represents the so-called Blocking Matrix (BM), $\underline{\mathbf{w}}_{c,f}^{H}$ the weight vector for the specified constraint(s) and $\underline{\mathbf{w}}_{u,f}^{H}[k]$ the unconstrained weight vector. Note the frame independency of the constrained weight vector, which is the principle of the GSC beamformer. $s_{c,f}[k]$ represents the output of the fixed, constrained beamformer and $s_{b,f}[k]$ is the estimated noise and interference that is going to be removed. $\underline{\mathbf{z}}_f[k]$, finally, must contain solely interference and noise.

Assume a number of $M \leq N$ independent constraints, together constructing the constraint matrix $\mathbf{C}_f$, leading to the constraint equation formulated as in Eq. (2.35). To exploit the number of degrees of freedom, determined by the number of sensors, $\mathbf{B}_f$ can be filled with $(N-M)$ independent, unconstrained vectors.

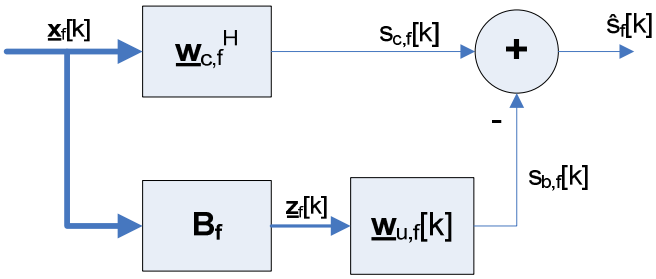


Figure 2-17 System representation GSC.

Ignoring frequency index f and frame index k, without loss of generality, the optimal weight coefficients then are constructed by [30]:

$$\underline{\mathbf{w}} = \underline{\mathbf{w}}_c - \mathbf{B}\underline{\mathbf{w}}_u = \mathbf{C}\underline{\mathbf{v}} - \mathbf{B}\underline{\mathbf{w}}_u \tag{2.39}$$

Using (2.35), this leads to [30]:

$$\underline{\mathbf{f}} = \mathbf{C}^H\underline{\mathbf{w}} = \mathbf{C}^H\mathbf{C}\underline{\mathbf{v}} - \mathbf{C}^H\mathbf{B}\underline{\mathbf{w}}_u \tag{2.40}$$

With $\underline{\mathbf{f}}$ representing the response vector of the complete system and $\underline{\mathbf{v}}$ being a temporary vector. As the blocking matrix $\mathbf{B}$ should block the desired signal $s[k]$, it must be constructed such that it is orthogonal to the constraint matrix $\mathbf{C}$ to avoid blocking (part of) the signal from the desired look direction:

$$\mathbf{C}^H\mathbf{B} = \underline{\mathbf{0}} \tag{2.41}$$

Using (2.40) together with (2.39) and (2.41) the constrained weight vector becomes:

$$\underline{\mathbf{w}}_c = \mathbf{C}\underline{\mathbf{v}} = \mathbf{C}\left(\mathbf{C}^H\mathbf{C}\right)^{-1}\underline{\mathbf{f}} \tag{2.42}$$

To solve the system for $\underline{\mathbf{w}}_u$, we again solve the minimization problem, but now unconstrained and to $\underline{\mathbf{w}}_u$, using Eq. (2.39):

$$\min_{\underline{\mathbf{w}}_u}\{P_{\hat{s}}\} = \min_{\underline{\mathbf{w}}_u}\left\{(\underline{\mathbf{w}}_c - \mathbf{B}\underline{\mathbf{w}}_u)^H\mathbf{R}_{\mathbf{x}\mathbf{x}^H}(\underline{\mathbf{w}}_c - \mathbf{B}\underline{\mathbf{w}}_u)\right\} \tag{2.43}$$

Setting the derivative equal to zero:

$$\frac{d}{d\underline{\mathbf{w}}_u} = -2\mathbf{B}^H\mathbf{R}_{\mathbf{x}\mathbf{x}^H}\underline{\mathbf{w}}_c + 2\mathbf{B}^H\mathbf{R}_{\mathbf{x}\mathbf{x}^H}\mathbf{B}\underline{\mathbf{w}}_u = 0 \tag{2.44}$$

This leads to the final solution for the unconstrained weight vector $\underline{\mathbf{w}}_u$ [30]:

$$\underline{\mathbf{w}}_u = (\mathbf{B}^H\mathbf{R}_{\mathbf{x}\mathbf{x}^H}\mathbf{B})^{-1}\mathbf{B}^H\mathbf{R}_{\mathbf{x}\mathbf{x}^H}\underline{\mathbf{w}}_c \tag{2.45}$$

Note that, when comparing this method to MVDR beamforming, the actual noise reduction performances of these two methods can be proven to be equivalent [4, 29]; it is solely the coefficient computation that differs.

There are several ways of constructing the blocking matrix **B**. One of them is provided by the well-known Griffith-Jim Beamformer (GJBF) [4, 10,16], In the GJBF the blocking matrix is created by subtracting each two adjacent channels after they are phase-aligned for the target direction in order to create a null in that target direction. For a 4-sensor ULA the blocking matrix then looks like:

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 \\ -e^{-j\zeta} & e^{-j\zeta} & 0 \\ 0 & -e^{-j2\zeta} & e^{-j2\zeta} \\ 0 & 0 & -e^{-j3\zeta} \end{pmatrix} \tag{2.46}$$

With again $\xi = \frac{2\pi F_c d}{c}\sin(\theta_0)$ according to Eq. (2.7). An other example is the Walsh blocking matrix [4, 31] which utilizes multiple sensors per constraint for steering a null in the target direction. Multiple combinations are possible as long as all the columns are linearly independent and sum up to zero. For the same 4-sensor ULA, an example of this matrix is:

$$\mathbf{B} = \begin{pmatrix} 1 & -1 & -1 \\ e^{-j\zeta} & e^{-j\zeta} & -e^{-j\zeta} \\ -e^{-j2\zeta} & e^{-j2\zeta} & e^{-j2\zeta} \\ -e^{-j3\zeta} & -e^{-j3\zeta} & e^{-j3\zeta} \end{pmatrix} \tag{2.47}$$

When trying to block the signal, all of these matrices, however, contain an important disadvantage. Due to the sharpness of the created null in the blocking response (see Figure 2-18), a slight mismatch between steering direction and actual target direction will at least partly filter away the requested target signal. Especially in speech enhancement and speaker tracking algorithms, this is an unacceptable disadvantage due to the establishment of only a single beam. Still, in our particular application this disadvantage is not of that much influence: if the target is accidentally filtered away in one steering direction, it will be detected in one of its adjacent steering directions. A thing that is often neglected, however, is the presence of coherent[2] sources or the creation of virtual, coherent sources due to a reverberant environment. In most

[2] Sources of which the exerted waves possess a constant phase difference between each other.

applications the standard version of the GSC namely is utilized solely to suppress background noise. Figure 2-18 depicts two plots of the outputs of the Griffiths-Jim blocking filter. The left image reflects the situation in which only one source is present, the right image shows the pitfall addressed in the case of a second, equal-strength, coherent source. It is clear the blocking matrix in the latter case is not able to place adequate nulls in the target directions.

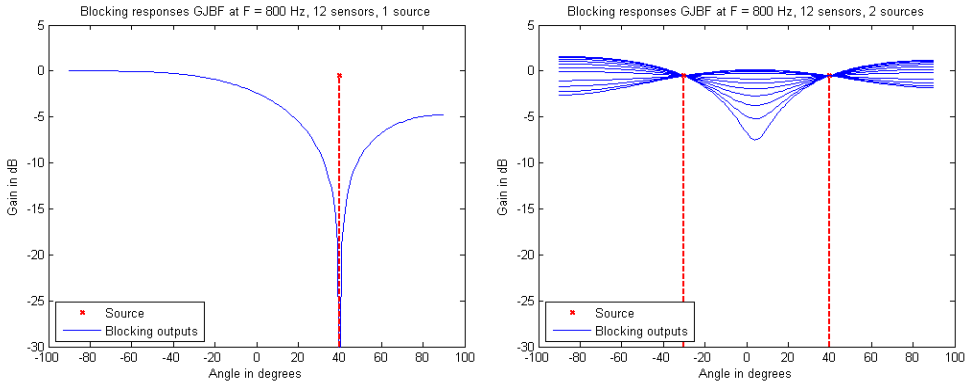


Figure 2-18 Output responses GJBF blocking filter. Left: one source. Right: two sources.

Other blocking matrices provide somewhat better results, but still more than far from acceptable. As the system is required to be applied in public, crowded environments like, for example, soccer stadiums, the presence of coherent sources may be considered a certainty. When still wanting to search for an improved beamforming algorithm with regard to relaxed computational demands, in the next paragraph a solution is tried to be found by means of an adaptive and more robust construction of the GSC.

2.2.5 *Adaptive Generalized Sidelobe Canceller*

The previous paragraph gave an introduction on the principle of the GSC and its accompanying disadvantages for this particular application. A way of coping with the mentioned problems, while keeping the goal of alleviating computational complexity in mind, is to use a form of adaptive beamforming. The slightly changed system representation of this beamformer in its frequency-domain implementation is depicted in Figure 2-19.

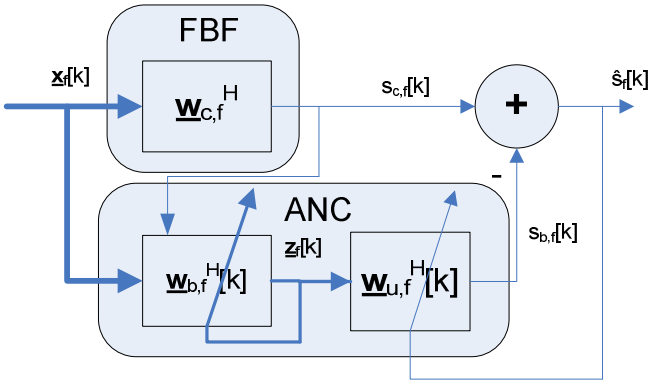


Figure 2-19 System representation of an adaptive beamformer.

The complete beamforming system can be divided into two paths. The upper path still is constituted by the constrained, fixed beamformer (FBF), delivering the output $s_{c,f}[k]$. The lower path consists of two parts. The first part remains containing

the blocking filter defined by its filter coefficients $\underline{\mathbf{w}}_{b,f}[k]$, but now in an adaptive fashion. The blocking matrix coefficients will be updated using a cost function that will try to remove correlation between the output of the FBF and the output of the BM. The right filter, defined by its filter coefficients $\underline{\mathbf{w}}_{u,f}[k]$ is still designed to filter away interference from the output of the FBF but also in an adaptive fashion. By using an update mechanism, it is possible to reduce computational intensity by calculating a new (adaptive) unconstrained weight vector for frame index *k+1*, based on the weight vector of frame *k* and an adaptation criterion. There are several algorithms that provide an equation for this last criterion, all having their advantages and disadvantages related to performance and computational complexity. A few familiar ones will be discussed here.

Until now, beamforming has been discussed in a frequency domain context, which is most common and clarifying for the considered methods. Adaptive filtering, however, is a method that finds its applications executed in frequency domain as well as in time domain. In order to obtain the best overview regarding complexity, both domains will be considered. First, a time domain filter derivation and elaboration will be done after which the scheme will be translated to frequency domain for assessing the quality and complexity of both approaches.

2.2.5.1 *Time-domain adaptive filtering*

In the next paragraphs an introduction is given on adaptive filtering. Several schemes will be described, leading to a final basic method of filtering applied in the AGSC. In the first discussions, only a single-channel filter is considered to derive the filtering equations. In these discussions, Figure 2-20 will be used, representing the schematic of adaptive filtering applied to the ANC in Figure 2-19 in which $\hat{e}[m]$ represents the estimated noise plus interference, $e[m]$ the actual noise plus interference and $\varepsilon[m]$ the difference between these two.

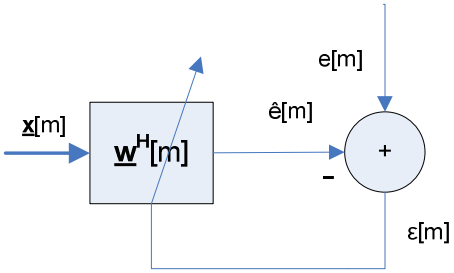


Figure 2-20 Adaptive filtering structure.

2.2.5.1.1 *Steepest Gradient Descent*

Consider the model in Figure 2-20 which represents a single channel time-domain filter. The input is considered to be real-valued. Each time instant *m*, an extra sample $x[m]$ is added to the filter input vector:

$$\underline{\mathbf{x}}[m] = [x[m], x[m-1], \ldots, x[m-L+1]] \tag{2.48}$$

With *L* being the filter length. Using input vector $\underline{\mathbf{x}}[m]$ and by updating the filter coefficient vector:

$$\underline{\mathbf{w}}[m] = [w_0[m], w_1[m], \ldots, w_{L-1}[m]] \tag{2.49}$$

there is tried to estimate each sample $e[m]$ by computing:

$$\hat{e}[m] = \underline{\mathbf{x}}[m]\underline{\mathbf{w}}^T[m] \tag{2.50}$$

The remaining error is then defined as:

$$\varepsilon[m] = \hat{e}[m] - e[m] \tag{2.51}$$

The Steepest Gradient Descent algorithm is a method to update the weight vector for the following time instant *m* by adjusting it with a computed gradient vector $\underline{\mathbf{v}}$ of the weight vector curve [32]:

$$\underline{\mathbf{w}}[m+1] = \underline{\mathbf{w}}[m] - \alpha\underline{\mathbf{v}} \tag{2.52}$$

In which α denotes a factor that represents the adaptation speed. The gradient vector is defined as the derivative of a cost function, *J*[m], to the weight vector. A commonly used criterion for this cost function is the MSE such that (2.52) can be rewritten as (considering Figure 2-20) [32]:

$$\underline{\mathbf{w}}[m+1] = \underline{\mathbf{w}}[m] - \alpha\frac{\partial J[m]}{\partial \underline{\mathbf{w}}[m]} = \underline{\mathbf{w}}[m] - \alpha\frac{\partial E\left\{\varepsilon^2[m]\right\}}{\partial \underline{\mathbf{w}}[m]} \tag{2.53}$$

When proceeding with the last part of (2.53), we can rewrite this as:

$$\begin{aligned}\frac{\partial E\left\{\varepsilon^2[m]\right\}}{\partial \underline{\mathbf{w}}[m]} &= E\left\{\frac{\partial \varepsilon^2[m]}{\partial \underline{\mathbf{w}}[m]}\right\} = 2E\left\{\varepsilon[m]\frac{\partial \varepsilon[m]}{\partial \underline{\mathbf{w}}[m]}\right\} \\ &= 2E\left\{\varepsilon[m]\frac{\partial\left(e[m] - \underline{\mathbf{x}}[m]\underline{\mathbf{w}}^T[m]\right)}{\partial \underline{\mathbf{w}}[m]}\right\} = -2E\left\{\varepsilon[m]\underline{\mathbf{x}}[m]\right\}\end{aligned} \tag{2.54}$$

Writing out the expectation operator leads to:

$$E\left\{\varepsilon[m]\underline{\mathbf{x}}[m]\right\} = E\left\{\left(e[m] - \underline{\mathbf{x}}[m]\underline{\mathbf{w}}^T[m]\right)\underline{\mathbf{x}}[m]\right\} = \underline{\mathbf{r}}_{e\mathbf{x}} - \mathbf{R}_{\mathbf{xx}}\underline{\mathbf{w}}^T \tag{2.55}$$

Combining these formulas, we obtain the final weight vector update equation:

$$\underline{\mathbf{w}}[m+1] = \underline{\mathbf{w}}[m] + 2\alpha\left(\underline{\mathbf{r}}_{e\mathbf{x}} - \mathbf{R}_{\mathbf{xx}^H}\underline{\mathbf{w}}^T\right) \tag{2.56}$$

An issue that can be seen directly is that we require the cross correlation between the input vector $\underline{\mathbf{x}}[m]$ and the actual interference plus noise signal $e[m]$ of which the last one is not present. A measure to solve this issue is by making use of an instantaneous estimate of the gradient. The Least Mean Squares (LMS) algorithm, outlined in the next paragraph, defines an often used method.

Before elaborating on the LMS algorithm, it is useful to determine how to estimate a value for α as this parameter will also return in the LMS algorithm. To start with, a

large value of α will increase the convergence speed to an optimal weight vector. When choosing this value too large, however, convergence will not be guaranteed. When wanting to ensure convergence, α must be limited to a maximum value for which convergence still is achieved.

The derivation of this maximum starts from the principle that the error between the computed weight vector and the optimal weight vector must reduce every iteration. When this error is defined as:

$$\underline{\mathbf{d}}[m] = \underline{\mathbf{w}}[m] - \underline{\mathbf{w}}_o \tag{2.57}$$

with $\underline{\mathbf{w}}_o$ being the optimal weight vector, the error at the next iteration can be recursively defined by [33]:

$$\underline{\mathbf{d}}[m+1] = \underline{\mathbf{d}}[m] + 2\alpha\left(\underline{\mathbf{r}}_{ex} - \mathbf{R}_{\mathbf{xx}}\underline{\mathbf{w}}^T[m]\right) \tag{2.58}$$

Writing out this equation leads to:

$$\begin{aligned}\underline{\mathbf{d}}[m+1] &= \underline{\mathbf{d}}[m] + 2\alpha\left(\mathbf{R}_{\mathbf{xx}}\underline{\mathbf{w}}_o^T - \mathbf{R}_{\mathbf{xx}}\underline{\mathbf{w}}^T[m]\right)\\ &= \underline{\mathbf{d}}[m] - 2\alpha\mathbf{R}_{\mathbf{xx}}\underline{\mathbf{d}}[m]\\ &= \left(I - 2\alpha\mathbf{R}_{\mathbf{xx}}\right)\underline{\mathbf{d}}[m]\end{aligned} \tag{2.59}$$

An eigenvalue decomposition on $\mathbf{R}_{\mathbf{xx}}$ such that $\mathbf{R}_{\mathbf{xx}} = \mathbf{Q}\Lambda\mathbf{Q}^T$ gives [33]:

$$\underline{\mathbf{d}}[m+1] = \left(I - 2\alpha\mathbf{Q}\Lambda\mathbf{Q}^T\right)\underline{\mathbf{d}}[m] = \mathbf{Q}\left(I - 2\alpha\Lambda\right)\mathbf{Q}^T\underline{\mathbf{d}}[m] \tag{2.60}$$

This can be rewritten to:

$$\mathbf{Q}^T\underline{\mathbf{d}}[m+1] = \left(I - 2\alpha\Lambda\right)\mathbf{Q}^T\underline{\mathbf{d}}[m] \tag{2.61}$$

When defining:

$$\underline{\tilde{\mathbf{d}}}[m] = \mathbf{Q}^T\underline{\mathbf{d}}[m] \tag{2.62}$$

We can rewrite (2.61) recursively as:

$$\underline{\tilde{\mathbf{d}}}[m+1] = \left(I - 2\alpha\Lambda\right)\underline{\tilde{\mathbf{d}}}[m] \tag{2.63}$$

The direct formula becomes:

$$\underline{\tilde{\mathbf{d}}}[m+1] = \left(I - 2\alpha\Lambda\right)^{m+1}\underline{\tilde{\mathbf{d}}}[0] \tag{2.64}$$

The maximum eigenvalue in this case determines the allowable value of α in order to ensure convergence:

$$\left|1-2\alpha\lambda_{\max}\right|<1 \quad \Rightarrow \quad 0<\alpha<\frac{1}{\lambda_{\max}} \tag{2.65}$$

2.2.5.1.2 Least Mean Squares

One of the most familiar and widely used adaptation schemes is the Least Mean Squares (LMS) algorithm. As we will see, it is especially famous due to its low computational complexity and easy implementation.

The difference in approach of the LMS algorithm compared to the SGD algorithm is that in its cost function $J[m]$, the LMS algorithm does not make use of the expectation of the MSE but the instantaneous least-squares error, i.e. an estimation of the MSE, leading to a simplified expression [34]:

$$\underline{\mathbf{w}}[m+1]=\underline{\mathbf{w}}[m]-\alpha\frac{\partial J[m]}{d\,\underline{\mathbf{w}}[m]}=\underline{\mathbf{w}}[m]-\alpha\frac{\partial\varepsilon^2[m]}{\partial\underline{\mathbf{w}}[m]} \tag{2.66}$$

With:

$$\begin{aligned}\frac{\partial\varepsilon^2[m]}{\partial\underline{\mathbf{w}}[m]}&=\frac{\partial\varepsilon^2[m]}{\partial\underline{\mathbf{w}}[m]}=2\varepsilon[m]\frac{\partial\varepsilon[m]}{\partial\underline{\mathbf{w}}[m]}\\&=2\varepsilon[m]\frac{\partial\left(e[m]-\underline{\mathbf{x}}[m]\underline{\mathbf{w}}^T[m]\right)}{\partial\underline{\mathbf{w}}[m]}=-2\underline{\mathbf{x}}[m]\varepsilon[m]\end{aligned} \tag{2.67}$$

Hence the update equation of the weight vector is defined as [11, 17, 34]:

$$\underline{\mathbf{w}}[m+1]=\underline{\mathbf{w}}[m]+2\alpha\underline{\mathbf{x}}[m]\varepsilon[m] \tag{2.68}$$

Considering the update equation of the weight vector, an important characteristic is the easy computation of the weight vector for the next time instance: only $L+1$ multiplications and L additions are required to compute the next weight vector. Still, everything comes with its compromises and hence the convergence rate to the optimal weight vector $\underline{\mathbf{w}}_o$ is relatively low.

An issue encountered is that the convergence strongly depends on σ_z^2: the power of the input vector $\underline{\mathbf{x}}[k]$. Non-stationary input signals, which are very common for this particular application, can lead to amplification of gradient noise when possessing substantial power. I.e. the absolute values of the weight coefficients can increase strongly, leading to noise amplification. The next paragraph explains the widely used solution to this problem.

2.2.5.1.3 Normalized Least Mean Squares

The Normalized Least Mean Squares method is a slightly modified version of the LMS algorithm. In order to avoid explosion of the weight coefficients, the weight update parameter α is normalized with the norm of the power of the input signal of the adaptive filter, leading to the following update equation of the weight vector [6, 16, 17, 34]:

$$\underline{\mathbf{w}}[m+1]=\underline{\mathbf{w}}[m]+\frac{2\alpha}{\left\|\underline{\mathbf{x}}[m]\right\|^2+\mu}\underline{\mathbf{x}}[m]\varepsilon[m] \tag{2.69}$$

With μ being a small constant to avoid divergence when the input power is zero. The constraint for α now changes to:

$$0 < \alpha < 1 \tag{2.70}$$

It would be an intuitive decision to choose a high value of α in order to achieve fast convergence. A higher value of α, however, leads to a larger offset in the steady state of the output of the filter. Based on the type of application a balance must be found between these two properties: for relatively non-stationary input signals the final filter output will profit more from a higher convergence rate while, on the other hand, with stationary input signals a more accurate long term steady-state of the output signal will lead to better performance. Of course the decision for this factor will also be influenced by the update rate of the filter coefficients as a filter may not have to be updated at each time sample which will be seen in the following paragraphs.

2.2.5.2 *Time-domain NLMS-NLMS Noise Canceller*

After having discussed the derivation and properties of a single-channel NLMS filter, it is incorporated in the complete AGSC structure as according to Figure 2-21 [10]. Note that we are still considering a time domain implementation.

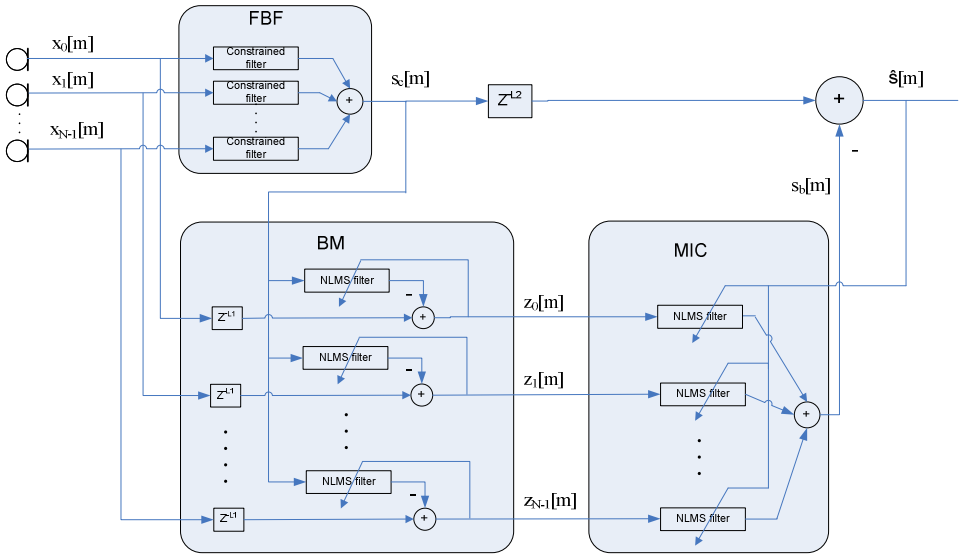


Figure 2-21 Overview of AGSC beamformer with NLMS adaptive filters.

The structure still consists of a fixed beamformer (FBF) and an adaptive noise canceller (ANC). The ANC contains the two before-mentioned filters but now in an adaptive fashion: the adaptive blocking matrix (ABM) and the multiple input canceller (MIC). Both of these two adaptive filtering structures contain NLMS-filters. The ABM tries to filter away all correlation between every sensor output $x_0[m]...x_{N-1}[m]$ and the output $s_c[m]$ of the FBF. The MIC uses all of the outputs $z_0[m]...z_{N-1}[m]$ of the ABM as inputs in order to filter away correlation between these inputs and the total output $\hat{s}[m]$ of the complete AGSC. The number of sample delays L_1 and L_2 are incorporated for causality.

Based on Figure 2-19 and Eq. (2.69), the NLMS equations of the ABM become as follows:

$$z_n[m] = x_n[m - L_1] - \underline{\mathbf{s}}_c[m]\underline{\mathbf{w}}_{b,n}{}^T[m] \tag{2.71}$$

$$\underline{\mathbf{w}}_{b,n}[m+1] = \underline{\mathbf{w}}_{b,n}[m] + \frac{2\alpha}{\left\|\underline{\mathbf{s}}_c[m]\right\|^2 + \mu}\underline{\mathbf{s}}_c[m]z_n[m] \tag{2.72}$$

With *n=0...N-1* being the channel number. The equations for the MIC are:

$$\hat{s}[m] = s_c[m - L_2] - \sum_{n=0}^{N-1}\underline{\mathbf{z}}_n[m]\underline{\mathbf{w}}_{u,n}{}^T[m] \tag{2.73}$$

$$\underline{\mathbf{w}}_{u,n}[m+1] = \underline{\mathbf{w}}_{u,n}[m] + \frac{2\alpha}{\left\|\underline{\mathbf{z}}_n[m]\right\|^2 + \mu}\underline{\mathbf{z}}_n[m]\hat{s}[m] \tag{2.74}$$

When applying this approach to the same situations as depicted in Figure 2-18, the outputs of the ABF show a significant improvement (Figure 2-22).

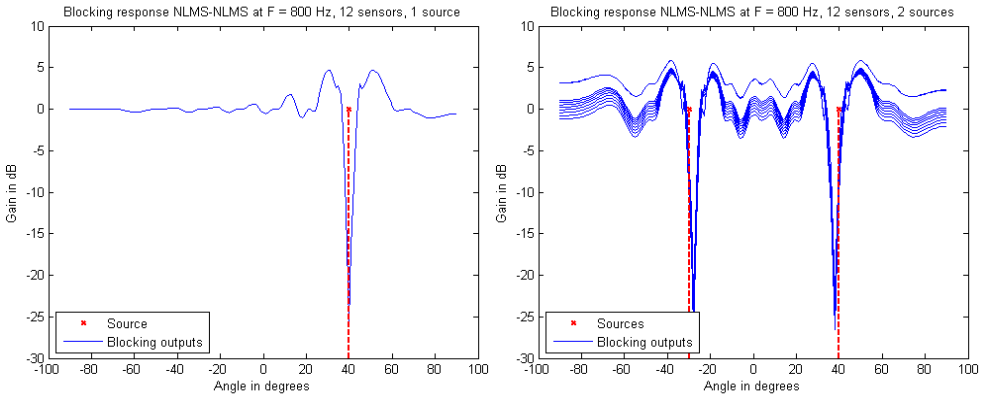


Figure 2-22 Output responses NLMS-NLMS blocking filter. Left: one source. Right: two sources.

Important to notice, still, is that the created nulls are very sharp. Again, this forms a problem in the situation of coherent sources (right figure): the exact position of the null is slightly displaced, making the ABM leak part of the target signal into the MIC. Furthermore, noise exerted from a source right next to the viewing direction is amplified. This on itself would not seem a bad thing as it is used for removing correlation from the output of the AGSC. The amplification, however, causes artifacts in the final output. Hoshuyama and Sugiyama [10] came with a solution directly addressing both problems at once. Next paragraph will outline this solution.

2.2.5.3 Time-domain CCAF-NCAF Noise Canceller

The issues listed in the previous paragraph are attempted to be diminished by a slightly different filter construction in the AGSC.

2.2.5.3.1 Description

Firstly, by constraining the value of the filter coefficients of the BM, a specific target direction error can be allowed. By doing this, filter coefficients will not explode to extremely high values, thereby avoiding the mentioned noise amplification and restraining null depth. In addition to this, the influence of remaining target signal leakage into the MIC, is diminished by applying a norm constraint to the filter coefficients of the MIC. Of course there is a disadvantage: due to enforcing a wider allowable target direction, and thus blocking a larger angle interval, less noise remains to be filtered away by the MIC.

The layout of the AGSC now changes to the one in Figure 2-23. The NLMS filters in the ABM are replaced by so-called Coefficient Constrained Adaptive Filters (CCAF) [10, 13]. The filters in the MIC are replaced by Norm Constrained Adaptive Filters (NCAF) [10, 13].

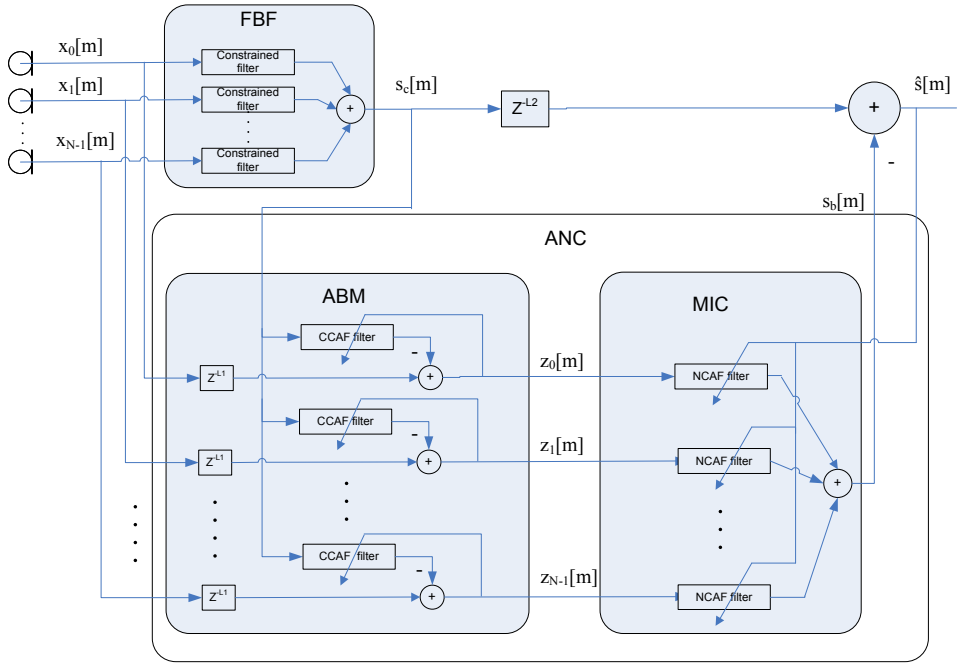


Figure 2-23 Overview of AGSC beamformer with CCAF and NCAF adaptive filters.

For both the ABM and the MIC, the update of the filters is slightly changed in order to meet the proposed constraints. For the ABM the filter coefficients for the next iteration now are defined by [10]:

$$\underline{\tilde{\mathbf{w}}}_{b,n}[m+1] = \underline{\mathbf{w}}_{b,n}[m] + \frac{2\alpha}{\left\|\underline{\mathbf{s}}_c[m]\right\|^2 + \mu}\underline{\mathbf{s}}_c[m]z_n[m] \tag{2.75}$$

$$\underline{\mathbf{w}}_{b,n}[m+1] = \begin{cases} \psi_n, & \underline{\tilde{\mathbf{w}}}_{b,n}[m+1] > \psi_n \\ \varphi_n, & \underline{\tilde{\mathbf{w}}}_{b,n}[m+1] < \varphi_n \\ \underline{\tilde{\mathbf{w}}}_{b,n}[m+1], & otherwise \end{cases} \tag{2.76}$$

With:

$$\underline{\psi}_n = \left[\psi_{n.0}, \psi_{n.1}, \ldots, \psi_{n.P1-1}\right] \tag{2.77}$$

$$\underline{\varphi}_n = \left[\varphi_{n.0}, \varphi_{n.1}, \ldots, \varphi_{n.P1-1}\right] \tag{2.78}$$

And P_1 being the ABM filter order. The values of these constraint vectors are derived based on applying a sinc-window with respect to the target direction interval, mapping the values of this window to the filter coefficients. This means a larger allowable target direction error corresponds to lower absolute filter coefficients. In mathematical notation this translates itself to [14]:

$$\underline{\psi}_{n,p} = -\underline{\varphi}_{n,p} = \frac{1}{\pi \max\{0.1,(p-L_1)-T_n,-(p-L_1)-T_n\}} \tag{2.79}$$

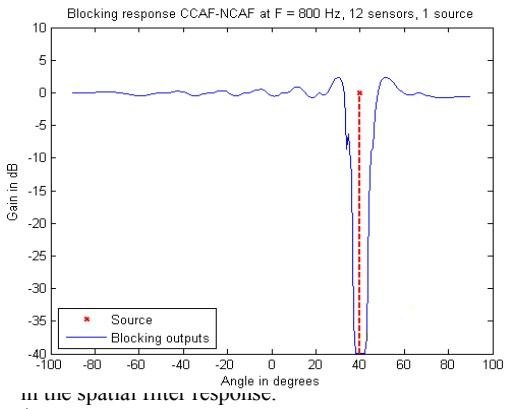
With $p=0...P_1-1$ representing the filter tap and T_n being defined as [14]:

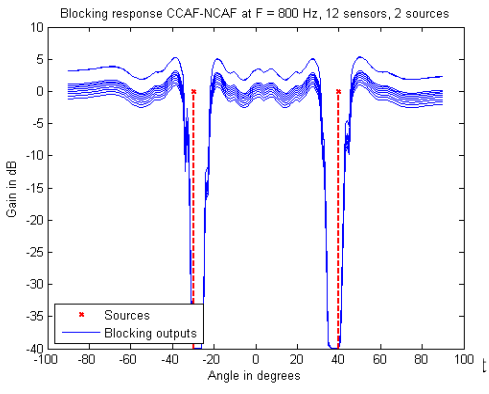
$$T_n = \frac{d_n F_s}{c}\sin(\Delta\theta) \tag{2.80}$$

In which F_s is the sampling frequency in [Hz], d_n the distance in [m] between sensor n and a reference point (e.g. the array centre or a reference sensor), c the speed of sound in [m/s] and $\Delta\theta$ the allowed target direction error in [º]. Note the similarity with Eq. (2.7) which defines the phase shift in frequency domain for a particular sensor. Eq. (2.80) defines the maximum allowable time error/shift in time domain. Eq. (2.79) translates this maximum allowable time error to a maximum value of the filter coefficients such that sources at positions of which the inter-sensor signal arrival time differences are within this maximum time window, are not filtered away. The extra value of 0.1 in Eq. (2.79) makes sure filter coefficients will not diverge.

For our particular application, the goal is to avoid the problem that is introduced in the situation of multiple coherent sources. The maximum allowable target direction error is not dependent on a target direction error in the case of a single source: as long as the complete scanned grid/area is fully covered by all constructed steering vectors, it will be observed. The maximum allowable target direction error therefore is dependent on the influence of the summation of multiple coherent sources. When defining this maximum allowable steering error equal to the maximum angular resolution[3] of a beam, a possible target signal cancellation for a particular beam will be 'picked up' at one of its adjacent beams.

The resolution of the beamformer is dependent on signal frequency and number of sources, as well as array structure and steering angle. Sources with higher frequencies can be resolved with a higher resolution (keeping in mind the maximum allowable frequency with respect to spatial aliasing). Furthermore, small[4] steering angles achieve higher spatial resolution than large steering angles. In either case, the worst-case situation has to be dealt with, which is a single-source situation with a frequency of F=1500 [Hz] at a steering angle of 0 [º]. Due to the non-regular structure of the used cross-array, this maximum resolution[5] is determined by measurements as opposed to deriving a complex resolution equation using an exact geometric model of the array.





in the spatial filter response.

[4] Small when the steering angle is defined as the offset from the line perpendicular to the axis of a broadside array.

[5] Measured maximum resolution of used cross-array under mentioned conditions is 5 [º].

Figure 2-24 Output responses CCAF-NCAF blocking filter. Left: one source. Right: two sources.

Figure 2-24 shows the output of the ABM using the same multi-source example when applying a steering vector error of +/-3 degrees.

The modified equation for the NCAF coefficients is defined by [10]:

$$\underline{\tilde{\mathbf{w}}}_{u,n}[m+1] = \underline{\mathbf{w}}_{u,n}[m] + \frac{2\alpha}{\left\|\underline{\mathbf{z}}_n[m]\right\|^2 + \mu}\underline{\mathbf{z}}_n[m]\hat{s}[m] \tag{2.81}$$

$$\underline{\mathbf{w}}_{u,n}[m+1] = \begin{cases} \sqrt{\frac{K}{P}}\underline{\tilde{\mathbf{w}}}_{u,n}[m+1], & P > K \\ \underline{\tilde{\mathbf{w}}}_{u,n}[m+1], & otherwise \end{cases} \tag{2.82}$$

With:

$$P = \underline{\tilde{\mathbf{w}}}_{u,n}[m+1]\underline{\tilde{\mathbf{w}}}_{u,n}^{T}[m+1] \tag{2.83}$$

K is a constant, the norm, which is defined by trial and error. It defines the maximum allowed power of a filter coefficient. A filter coefficient with a power exceeding this value is scaled down to have the maximum allowed power K. In this way, noise is filtered away up to a certain level which is a flaw of this method. The advantage of course is that a possible, slight, target signal leakage will not tremendously affect the obtained target signal estimation in the final output signal of the AGSC.

2.2.5.3.2 Computational complexity

As mentioned in the introduction of the AGSC, the search for a different approach is driven by the possibility of finding an algorithm with a lower complexity compared to data-driven MVDR beamforming. In order to assess the improvement or deterioration, an estimate is made on the computational complexity in the sense of the number of mathematical operations, similarly as done for MVDR beamforming.

To be able to compare this analysis to the frequency domain MVDR implementation, again consider a frame of M samples per each of N sensors used to create B beams. First of all, no FFTs on the sensor outputs have to be computed. As mentioned in the introduction of this report, however, the final beamformed output is required to be a frequency domain representation for simplifying the computation of sound parameters.

This means FFTs have to be computed for all B beams, which is less efficient because the number of beams will definitely exceed the number of sensors. Still, computational advantages may lie in other parts of the algorithm. Eq. (2.71) to (2.74) correspond to *ABM convolution* and *output/error*, *ABM correlation* and *update*, *MIC/AGSC convolution* and *output/error* and *MIC*/*AGSC correlation* and *update* in Table 2-2, respectively. Note that each equation has to be performed on each sensor channel and for each beam. For now, constraining the coefficients will be left out of analysis due to the non-deterministic number of operations. Consider a filter order of P and an FBF that applies conventional Delay and Sum beamforming; so without weighing the sensor channels.

Operation	**MUL**	**ADD**	**DIV**
FBF	-	BNM *(R)*	-
ABM			
Convolution	BNMP *(R)*	BNMP *(R)*	-
Output/error	-	BNM *(R)*	-
Correlation	BNM(P+2) *(R)*	-	BN *(R)*
Update	-	BNMP *(R)*	-
MIC/AGSC			
Convolution	BNMP *(R)*	BNMP *(R)*	-
Output/error	-	BM(N+1) *(R)*	-
Correlation	BNM(P+2) *(R)*	-	BN *(R)*
Update	-	BNMP *(R)*	-
FFT	$BM\log_2(M)$ *(R)*	$BM\log_2(M)$ *(R)*	-
Total *(R)*	**$BM[4NP+4N+\log_2(M)]$**	**$BM[4NP+3N+\log_2(M)+1]$**	**2BN**

Table 2-2 Computational complexity of time-domain AGSC.

The formulas do not provide a clear overview yet so the comparison with MVDR beamforming will be supported with a realistic example. The number of sensors in the used array is fixed to 32. With relation to the frequency bandwidth of the output of the FFT, used in MVDR in order to avoid a large steering vector error discussed in paragraph *2.1.1.3 Narrowband assumption*, the number of samples M is chosen to be 512. The number of beams B is set to 225 (15 x 15) for now, for scanning a complete grid. The decision for this will be explained in more detail later in this report. The time-domain filter order P is defined as twice the number of frequencies F processed by the MVDR beamformer. To process all the frequencies of the FFT-output, F equals $\frac{1}{2}M$, making P equal M. In this example a 32-bit real floating point multiplication is assumed to take twice as long as an addition [43]. A division is assumed to take five times as long as an addition. Filling in these example numbers, DMVDR needs about 4% the number of normalized operations the time-domain AGSC needs.

Conclusively, it is embarrassingly clear the time-domain AGSC does not at all contribute to a computational speed-up. But, using this as an intermediate step, there still might be a way to reduce the complexity of the AGSC by considering a frequency domain implementation. In the search for finding a more efficient algorithm, we keep in mind that the time-domain version of the AGSC must be speeded up about twenty-five times in order to be a reasonable competitor for DMVDR beamforming.

2.2.5.4 Frequency-domain adaptive filtering

Before considering a frequency-domain version of the AGSC, two schemes of frequency domain adaptive filtering (FDAF) will be discussed. To change from a time-domain implementation of adaptive filtering to a frequency-domain implementation, an intermediate step will be taken to provide an algorithm readily transferrable to the frequency domain. This step translates the standard NLMS algorithm to a block NLMS version.

2.2.5.4.1 Block NLMS adaptive filtering

The standard NLMS algorithm is not directly suited to transform to a frequency domain version. This paragraph will very briefly describe the change to a block NLMS algorithm.

Remind the equations that define the output error and filter update of the standard NLMS algorithm of a single-channel filter, which are repeated here for convenience:

$$\varepsilon[m] = e[m] - \underline{\mathbf{x}}[m]\underline{\mathbf{w}}^T[m]$$

$$\underline{\mathbf{w}}[m+1] = \underline{\mathbf{w}}[m] + \frac{2\alpha}{\left\|\underline{\mathbf{x}}[m]\right\|^2 + \mu}\underline{\mathbf{x}}[m]\varepsilon[m]$$

With:

$$\underline{\mathbf{w}}[m] = \left[w_0[m], w_1[m], ..., w_{L-1}[m]\right]$$

$$\underline{\mathbf{x}}[m] = \left[x[m], x[m-1], ..., x[m-L+1]\right]$$

With L being the filter length. A first step is to write out the recursive version of the filter update to an update over an interval of M time samples [5, 11, 35]:

$$\underline{\mathbf{w}}[m+M] = \underline{\mathbf{w}}[m] + 2\alpha\sum_{n=0}^{M-1}\frac{\underline{\mathbf{x}}[m+n]\varepsilon_{\underline{\mathbf{w}}[m+1]}[m+n]}{\left\|\underline{\mathbf{x}}[m+n]\right\|^2 + \mu} \tag{2.84}$$

The subscript $\underline{\mathbf{w}}[m+1]$ added to the output error means the error is computed with the filter vector computed at time instant $m+1$. The principle behind block based filtering is assuming the filter vector will only change slightly within a certain time interval, so therewith keeping the filter vector constant for a number of M samples. This means there will be an update of this vector every multiple of M samples. The value of M is almost always chosen equal to the filter length L. So the output errors of intermediate samples are computed using the latest update k of the filter coefficients, or in mathematical form [5, 11, 35]:

$$\underline{\mathbf{w}}[kM+M] = \underline{\mathbf{w}}[kM] + 2\alpha\sum_{n=0}^{M-1}\frac{\underline{\mathbf{x}}[kM+n]\varepsilon_{w[kM]}[kM+n]}{\left\|\underline{\mathbf{x}}[kM+n]\right\|^2 + \mu} \tag{2.85}$$

Note the change of the subscript of the output error. As the coefficients are only changed at every time instant kM, above equation can be written as [34]:

$$\underline{\mathbf{w}}[k+1] = \underline{\mathbf{w}}[k] + 2\alpha\sum_{n=0}^{M-1}\frac{\underline{\mathbf{x}}[kM+n]\varepsilon_{w[k]}[kM+n]}{\left\|\underline{\mathbf{x}}[kM+n]\right\|^2 + \mu} \tag{2.86}$$

$$= \underline{\mathbf{w}}[k] + 2\alpha\hat{\underline{\mathbf{v}}}_M \tag{2.87}$$

The output of the filter remains updated every time instant n but with an older version of the filter vector:

$$\hat{e}[kM+n] = \underline{\mathbf{x}}[kM+n]\underline{\mathbf{w}}^T[k] \tag{2.88}$$

The change to a block-based algorithm involves a property that must be taken into account. Because the filter is updated only every M samples, the response time of the filter to variations in the input signal is extended to a maximum of M samples. In other words: it is harder to keep track of non-stationary signals because the filter is only updated after a fixed time. The degree of stationarity of the input signal together with the filter length determines the ability to keep track of the signal. Still, for more wide-sense stationary signals, the steady-state response is proven to remain the same [5].

2.2.5.4.2 Overlap-save method

Description

The implementation of a frequency domain adaptive filter starts with being aware of the difference between the properties of a time-domain (N)LMS filter and a Discrete Fourier Transform (DFT). The time-domain filter applies a *linear* convolution and *linear* correlation for the computation of the output of the time-domain (N)LMS filter and update of the filter coefficients respectively, by multiplying the appropriate vectors/scalars. A multiplication of two DFT sequences, on the other hand, results in a circular convolution or circular correlation between the vectors/scalars in time domain. There is an overlap, however, between circular and linear convolution/correlation as illustrated in Figure 2-25. In the plots, an eight-point and a four-point sequence are correlated and convolved in a circular (periodic) and linear way. The plots show the differences between the linear and circular operations, but also show an overlap in values, indicated by the green, solid bars.

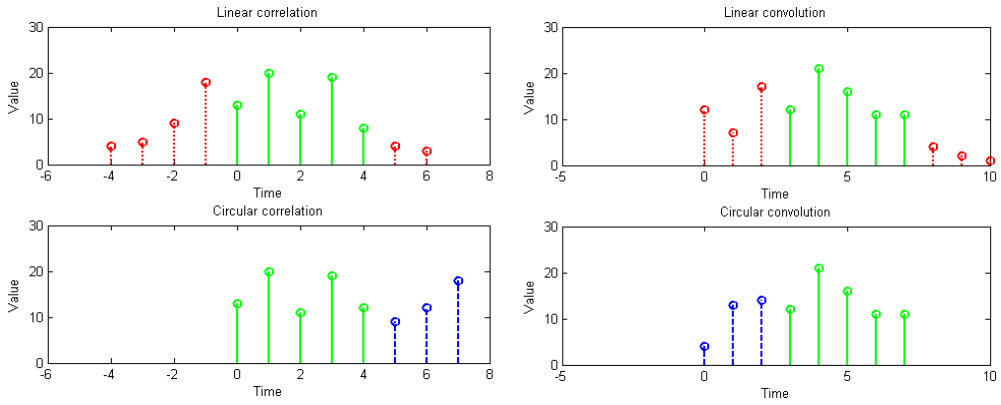


Figure 2-25 Matching values between linear and circular correlation (left) and linear and circular convolution (right) of an eight-point and four-point sequence.

If the lengths of the sequences are defined as S_1 and S_2, with $S_1 \geq S_2$, the number of overlapping values for both operations is:

$$M = S_1 - S_2 + 1 \tag{2.89}$$

The overlap between a circular and linear correlation applies for the first part of the output of the circular correlation. Because a convolution applies the same operation as a correlation, but with a reversely ordered sequence, the overlap between a circular and linear convolution applies for the last part of the output of the circular convolution. Knowing this relation, it can be used to our advantage in the frequency-domain filter implementation: by constructing a 2M-input data frame with the first half being a new part of data and the second half being the last half of the previous frame, we can work in the frequency domain while still obtaining the required linear correlation/convolution.

Figure 2-26 shows the complete diagram of this approach. Because of the use of both time- and frequency-domain signals in this method, variables will be denoted with the subscript $(.)_t$ or $(.)_f$ respectively. As already mentioned, the frequency domain input of frame *k* of length *2M* is constructed by overlapping two frames and applying an FFT [17]:

$$\underline{\mathbf{x}}_f[k] = FFT\left(x_t[kM-M], x_t[kM-M+1], \ldots x_t[kM+M-1]\right) \tag{2.90}$$

In which *FFT(.)* stands for the Fast Fourier Transform.

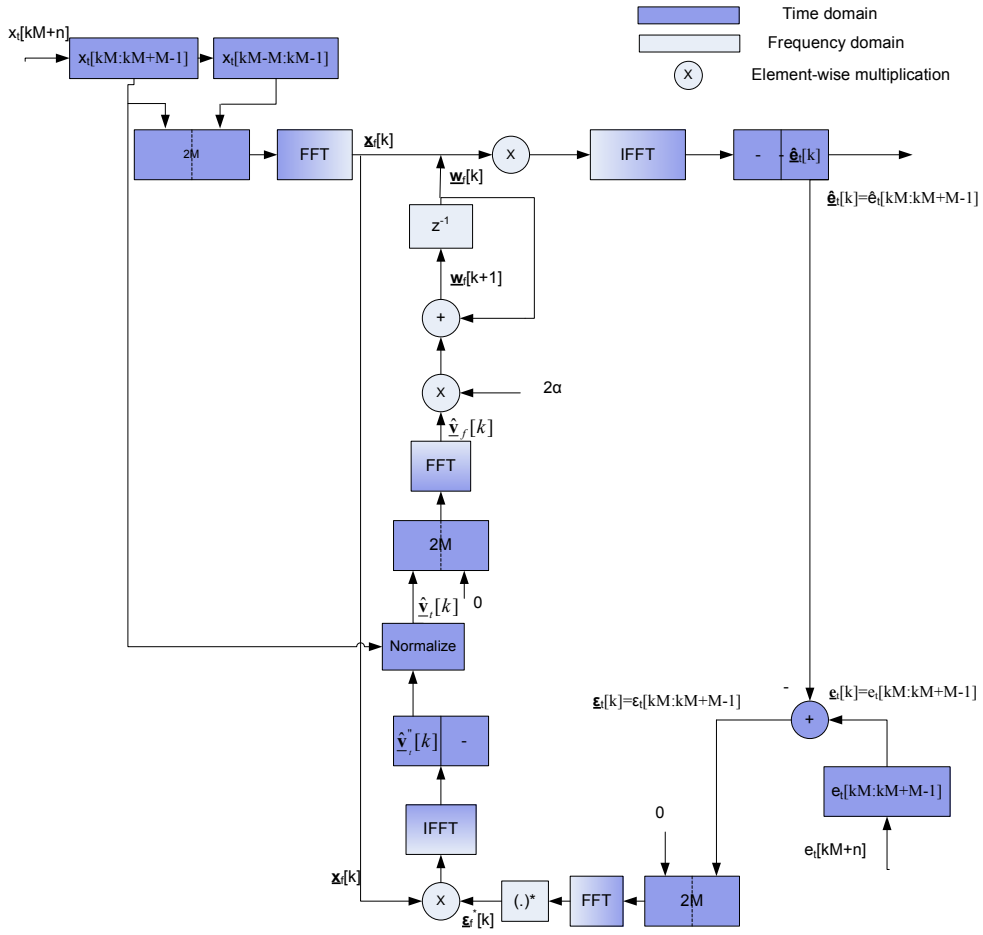


Figure 2-26 Overview FDAF filter structure of overlap-save method.

To match the linear correlation of the computation of the weight vector in time domain with the circular correlation of that same operation in frequency domain, the time domain weight vector is extended with zeros [17, 35], in accordance with Figure 2-25:

$$\underline{\mathbf{w}}_f[k] = FFT\left(\underline{\mathbf{w}}_t[k], \underline{\mathbf{0}}_M\right) \tag{2.91}$$

With:

$$\underline{\mathbf{0}}_M = [0_0, 0_1, \ldots, 0_{M-1}] \tag{2.92}$$

The frequency-domain output then is defined by [7, 11, 17, 35]:

$$\underline{\hat{\mathbf{e}}}_f[k] = \underline{\mathbf{x}}_f[k] \times \underline{\mathbf{w}}_f^T[k] \tag{2.93}$$

Where $\times$ stands for element-wise multiplication. The opposite holds for the time domain output vector: to account for the linear correlation, the time domain output signal is defined by the last part of the inverse FFT (IFFT) of the output $\underline{\hat{\mathbf{e}}}_f[k]$ [17, 35]:

$$\underline{\hat{\mathbf{e}}}_t[k] = \left[\underline{\hat{\mathbf{e}}}_t[kM],...,\underline{\hat{\mathbf{e}}}_t[kM+M-1]\right] \tag{2.94}$$

$$= \text{ last } M \text{ components of } IFFT\left(\underline{\hat{\mathbf{e}}}_f^T[k]\right) \tag{2.95}$$

The error $\underline{\varepsilon}_t[k]$ then becomes:

$$\underline{\varepsilon}_t[k] = \underline{\hat{\mathbf{e}}}_t[k] - \underline{\mathbf{e}}_t[k] \tag{2.96}$$

For the same before mentioned reasons, the error vector is augmented with zeros such that the frequency-domain error becomes [7, 17, 35]:

$$\underline{\varepsilon}_f[k] = FFT\left(\underline{\mathbf{0}}_M, \underline{\varepsilon}_t[k]\right) \tag{2.97}$$

Following Figure 2-26, the gradient vector estimation $\underline{\hat{\mathbf{v}}}_t[k]$ becomes [7, 35]:

$$\underline{\hat{\mathbf{v}}}_t^{''}[k] = \text{ first } M \text{ components of } IFFT\left(\underline{\varepsilon}_f^*[k] \times \underline{\mathbf{x}}_f[k]\right) \tag{2.98}$$

$$\underline{\hat{\mathbf{v}}}_t[k] = \frac{\underline{\hat{\mathbf{v}}}_t^{''}[k]}{\left\|\underline{\mathbf{x}}_t[kM...kM+M-1]\right\|^2 + \mu} \tag{2.99}$$

Leading to the final weight vector update [7, 35]:

$$\underline{\mathbf{w}}_f[k+1] = \underline{\mathbf{w}}_f[k] + 2\alpha FFT\left(\underline{\hat{\mathbf{v}}}_t[k], \underline{\mathbf{0}}_M\right) \tag{2.100}$$

To assess the advantage of this method with respect to the time-domain implementation, in the next paragraph again an estimate will be made on the computational complexity.

Computational complexity
For the computation of the complexity of the complete beamformer, Figure 2-23 of the time-domain CCAF-NCAF Noise Canceller will be reused, but now considering the frequency domain implementation. Each time-domain filter channel will be replaced by its frequency-domain equivalent, meaning the structure will only change inside the filters themselves.

Table 2-3 lists the number of operations per component with *M* being the frame length, *N* the number of sensors and *B* the number of beams. The number of output frequencies is ½*M*, making the comparison equal to the example comparison between DMVDR and time-domain AGSC beamforming in paragraph *2.2.5.3.2 Computational complexity*.

Using the same parameters as in this example, the frequency-domain implementation turns out to be an improvement of the time-domain variant: it is about 10 times faster. Still, it not yet outperforms DMVDR which remains 2½ times more efficient. A last attempt to speed things up will be made in the next paragraph.

Operation	**MUL**	**ADD**	**DIV**
FBF	-	BNM (R)	-
ABM			
(I)FFTs	$10BNM\log_2(2M)$ *(R)*	$10BNM\log_2(2M)$ *(R)*	-
Convolution	BNM *(C)*	-	-
Output/error	-	BNM *(R)*	-
Correlation	BNM *(C)*	-	-
Normalization	BNM *(R)*	BNM *(R)*	BN *(R)*
Filter update	-	BNM *(C)*	-
MIC/AGSC			
(I)FFTs	$10BNM\log_2(2M)$ *(R)*	$10BNM\log_2(2M)$ *(R)*	-
Convolution	BNM *(C)*	-	-
Output/error	-	BM(N+1) *(R)*	-
Correlation	BNM *(C)*	-	-
Normalization	BNM *(R)*	BNM *(R)*	BN *(R)*
Filter update	-	BNM *(C)*	-
FFT Output	$BM\log_2(M)$ *(R)*	$BM\log_2(M)$ *(R)*	-
Total *(R)*	**$BM[18N+(20N+1)\log_2(M)]$**	**$BM[13N+(20N+1)\log_2(M)+1]$**	**2BN**

Table 2-3 Computational complexity linear convolution AGSC.

2.2.5.4.3 Circular Convolution method

The linear convolution method delivers a scheme for applying adaptive filtering in frequency domain. It does not yet provide the desired reduction of computational complexity though. The circular convolution method disobeys the property of linear convolution in the update of the filter coefficients [8]. In this way it wins in on computational relaxation but at the cost of performance degradation related to convergence and steady-state. This paragraph will describe the changes compared to the linear convolution method and evaluate the decrease in complexity together with its performance degradation.

Figure 2-27 shows the reduced diagram of the circular convolution method. A couple of things have changed as a consequence of applying circular convolution in the filter update. Because the time domain representation of the filter is not extended to twice the frame length *M* for enforcing linear convolution, the filter update can be performed directly in frequency domain without intermediate conversions from time domain to frequency domain and vice versa. This means an input frame of *N* samples corresponds to an output frame of *N* samples. The most important change in the filter update is [8]:

$$\hat{\underline{\mathbf{v}}}_f[k] = \frac{\underline{\varepsilon}_f^*[k] \times \underline{\mathbf{x}}_f[k]}{\left\|\underline{\mathbf{x}}_f[k]\right\|^2 + \mu} \tag{2.101}$$

$$\underline{\mathbf{w}}_f[k+1] = \underline{\mathbf{w}}_f[k] + 2\alpha\hat{\underline{\mathbf{v}}}_f[k] \tag{2.102}$$

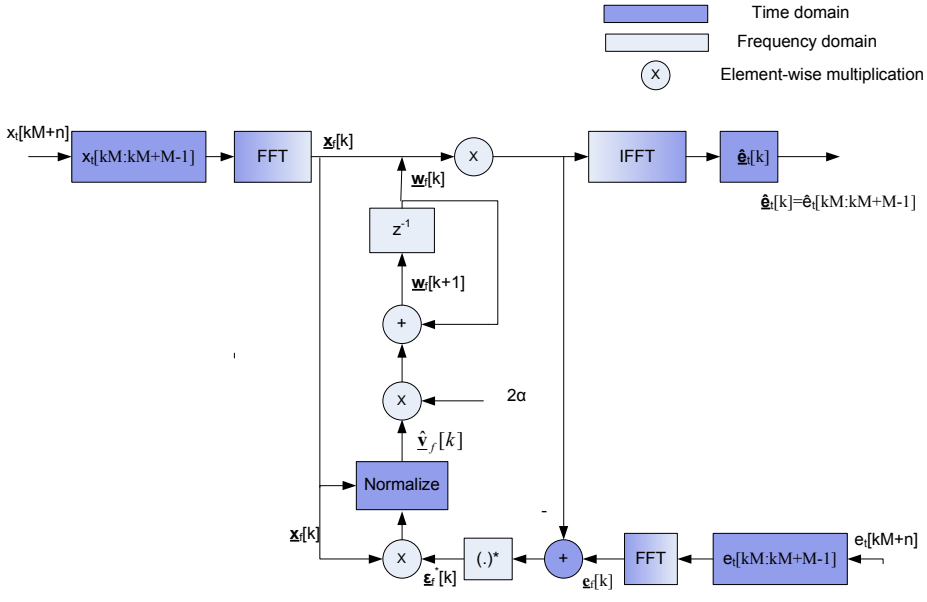


Figure 2-27 Overview FDAF filter based on circular convolution method.

Computational complexity

Figure 2-27 indicates a number of three (I)FFTs required to implement the filter. This is already a significant reduction compared to the linear convolution method which uses five. The filter assumes the reference signal to be a time domain sequence that has to be converted to frequency domain. Applying the filter in the adaptive beamforming scheme learns that the reference signal is the output of the FBF which already can be computed in frequency domain. This saves one FFT, but because we will then have the reference signal (the sensor output) also already in frequency domain, the last FFT can be removed as well. Moreover, the output of the filter also was assumed to be converted to time domain. For our particular use, the input of the NCAF-filter must be transferred to frequency domain as well, meaning we can remove the output IFFT from the CCAF-filter and the input FFT of the NCAF-filter. Because we require the output of the complete AGSC to be in frequency domain, we can also remove the remaining two (I)FFTs from the NCAF-filter. This means the only FFT that rests is the one on the sensor outputs. In Table 2-4 the complexity is listed by means of the amount of required operations.

Operation	**MUL**	**ADD**	**DIV**
FFT sensors	$NM\log_2(M)$ *(R)*	$NM\log_2(M)$ *(R)*	-
FBF	½BNM *(C)*	½BNM *(C)*	-
ABM			
(I)FFTs	-	-	-
Convolution	½BNM *(C)*	-	-
Output/error	-	½BNM *(C)*	-
Correlation	½BNM *(C)*	-	-
Normalization	½BNM *(C)*	½BNM *(C)*	BN *(C)*
Filter update	-	½BNM *(C)*	-
MIC/AGSC			
(I)FFTs	-	-	-

Convolution	½BNM *(C)*	-	-
Output/error	-	½BM(N+1) *(C)*	-
Correlation	½BNM *(C)*	-	-
Normalization	½BNM *(C)*	½BNM *(C)*	BN *(C)*
Filter update	-	½BNM *(C)*	-
Total *(R)*	**NM[14B+log$_2$(M)]+16BN**	**NM[14B+log$_2$(M)]+BM+6BN**	**2BN**

Table 2-4 Computational complexity circular convolution AGSC.

Applying the example used throughout this paragraph, confirms the seemingly reduced complexity in Table 2-4: the circular convolution method achieves a speed-up of approximately 14 times with respect to the linear convolution method. Moreover, the new method is about 6 times as fast as DMVDR. The question remains whether this is a fair comparison.

Degradation

Because the gradient $\underline{\mathbf{v}}_f[k]$ is estimated in frequency domain now, wrap-around effects are introduced due to the assumption of a periodic input, because of the application of circular convolution. Since the application environment of the system is not at all long- nor short-term stationary, this will lead to poorer convergence. The CCAF blocking filter must ensure reasonable convergence, however, to remove as much target signal as possible. The convergence of the MIC is of lower interest: less convergence will just lead to a decreased ability of filtering away interference. But, a combined use of both methods (circular and linear convolution) will still not contribute to a speed-up of DMVDR.

Though it took a lot of effort in coming to a computationally relieved solution, it has to be concluded that this method can not be used as a substitute method in dynamic beamforming.

2.3 Evaluation

In the previous paragraphs the principles and aspects of beamforming have been discussed. Several methods have been outlined and evaluated, all having their advantages but also their flaws. This paragraph will shortly describe the decision for a suitable method that will be used for implementation and will define some restrictions that apply for the array used in this particular project.

2.3.1 Conclusion beamforming methods

The application of conventional DAS beamforming provides a reasonable solution but lacks directivity at lower frequencies. Conventional, static, frequency-invariant window functions are able to change the response pattern to create a flatter pattern or one with deeper nulls. This does not improve the response at lower frequencies though. Neither will it be able to assist in active noise and interference suppression.

SMVDR beamforming is able to solve the first problem by deriving filter coefficients based on an assumption of the noise coherence matrix at a certain frequency. In this way it is able to preserve the spatial response of the DAS beamformer over a larger frequency interval. The cost of a decreased WNG can be diminished by making the autocorrelation matrix more robust, leading to a balance between directivity and noise amplification. Furthermore, it still provides equal complexity due to its static nature. A negative characteristic, still, is the inability of active interference suppression and the fact that the angular beam resolution at higher frequencies is not an improvement compared to DAS.

DMVDR beamforming is able to cope with both problems: it has high resolution at both high and low frequencies in the presence of a single source and also delivers a reasonable improvement in a coherent, multisource environment. This will be one of the most suitable methods to use, though we have to take into account its computational complexity.

Finally, an attempt has been made to find an adaptive beamforming algorithm of which the spatial performance approaches that of DMVDR but with a decreased complexity. A couple of adaptive implementations have been discussed, but the sacrifices that have to be made in order to reach a more efficient solution are not considered acceptable. The next chapter will discuss how the system is going to deal with the complex algorithms.

2.3.2 Array specific constraints

The beamforming methods and examples were applied for linear arrays in order to keep comparisons easy. The array used for this project, however, is a 2D-cross with four side branches (see appendix A). The array was originally designed for speech detection and therefore has a linear sensor spacing of 0.13 [m]. The side branches were added to improve the spatial resolution in the corners of a grid. Some constraints will be imposed on the frequency range that can be accurately covered with this array. Due to the unusual structure, this will be done based on simulation instead of computation.

2.3.2.1 Cut-off frequencies

According to paragraph *2.1.1.1 Aliasing*, the sensor spacing of 0.13 [m] would suggest an upper cut-off frequency of approximately 1310 [Hz], in order to avoid spatial aliasing. The side branches, however, all have an added value due to their virtual projection on the main branches (appendix A). By taking one main branch with the projected sensors of the side branches, the required upper cut-off frequency can be slightly increased to 1500 [Hz].

Because the array has a fixed aperture, there is also a lower bound on the allowed frequency range. Below this limit it is not possible anymore to locate multiple sources at their actual position. Figure 2-28 illustrates this using the conventional DAS method. Still, a linear array is considered, but now with the mentioned projection of the side branch sensors; so a realistic example. In the left plot, only one source is present. Though the spatial response is worse for lower frequencies, the source location is correct. The right plot shows the case in a multi-source environment. For lower frequencies (< 300 [Hz]) it is clear the coherence between the two sources and the limited array aperture make the array 'see' the two sources as a combined source in the middle of the two actual sources and slightly amplified.

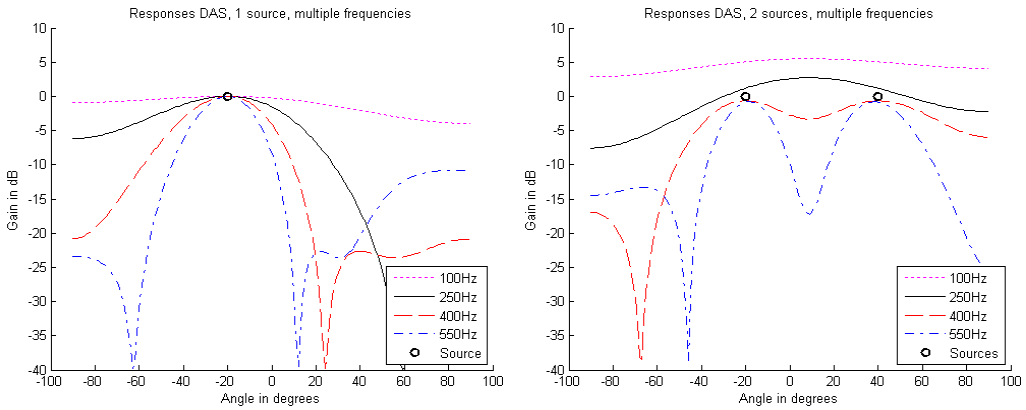


Figure 2-28 Responses DAS at multiple frequencies with 1 source (left) and 2 coherent sources (right).

Figure 2-29 gives a better overview of the multisource situation. Here, the response for all frequencies is illustrated. The array is able to distinguish two sources starting at approximately 350 [Hz]. This value will be used as the lower cut-off frequency. A thing that must be emphasized, however, is that for this example two sources with an angular distance of 60 [°] have been used. When this difference decreases or when multiple coherent sources are present, it will become harder to distinguish them. Ideally, one would like to increase the cut-off frequency up to the point where the ability of the array in distinguishing two sources approaches the maximum angular beam resolution. This would drastically reduce the frequency range available for sound classification; therefore a balance between the two properties is chosen. A realistic conclusion actually is that this array may not be really suited for this purpose. Testing must point out whether it will still provide an acceptable solution. At the end of this report a recommendation for a different array structure will be given.

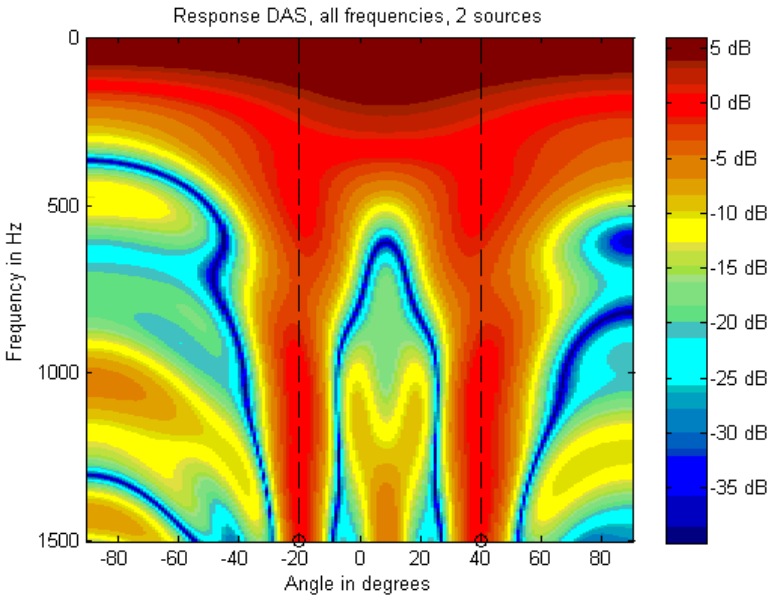


Figure 2-29 Full response plot DAS with two coherent sources.

2.3.2.2 Resolution

The resolution of the array is a measure for the sharpness of a constructed beam. Here it is defined as the absolute angular interval around the target direction where the attenuation is less than 3 [dB]. The resolution is highest at the maximum beamforming frequency (1500 [Hz]) in a single-source environment with a target direction of 0 [°] (perpendicular to the array). By means of simulation, the maximum resolution is determined to be approximately 5 [°]. For covering a complete 2D-grid, this is the safe value that must be used for computing the number of required beams in order not to miss any spot in the grid. Figure 2-30 shows a grid covered by all required adjacent beams.

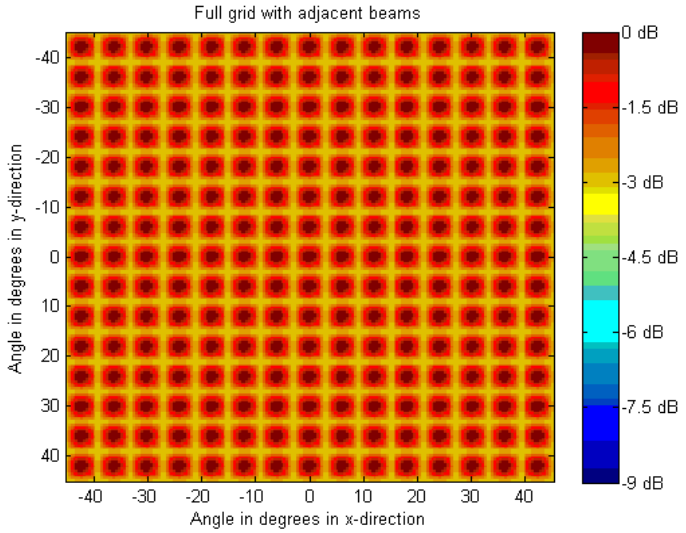


Figure 2-30 Area coverage of a complete grid of beams.

We must be aware that the resolution of DMVDR in a single-source environment is higher than that of DAS. This could mean a source is missed when using the number of beams, determined based on a DAS approach. The maximum resolution of DMVDR is approximately 1 [°]. For an average grid with an angular focusing interval of the array of 90 [°], this would require a number of 8100 beams to construct, which is computationally undoable. The diagonal of the autocorrelation matrix in robust DMVDR beamforming therefore is loaded with a higher value for higher frequencies to stick to the resolution of 5 [°]. This seems like destroying the advantage of DMVDR, but the slopes of the main beam remain significantly steeper than those of DAS. Also the directivity (suppression of interference over the whole angular range) remains a great improvement. Furthermore, for lower frequencies the loaded value is chosen to be lower such that the maximum resolution is retained over a considerably larger frequency interval than for DAS.

3 Implementation

Now a number of different beamforming methods have been discussed and compared, the chosen algorithms are implemented and incorporated in a real-time application. This final system must comprise all functionalities like data streaming, synchronization, beamforming and user interfacing in such a way that a robust design is obtained. This chapter will discuss the architecture of the system and things that are involved in heading to such a solution.

First, an implementation platform will be determined, after which an overview of the system will be given to refer to in the rest of this chapter. Specific parts of the system will be outlined in more detail.

3.1 Implementation platform

The array comprises, besides microphones, a NIOS II development board, connected to an Altera Stratix II FPGA. On the FPGA, a design is already present that just samples the AD-converters and fills the appropriate buffers. We may consider an implementation of the system directly on this FPGA, with the goal of achieving a high-speed solution. A few things have to be taken in consideration then.

First of all, let us look at the type of operations that have to be performed. Only for DMVDR-beamforming itself, 32 FFTs must be computed; a 32x32 complex matrix inversion is needed per frequency bin and a number of complex vector multiplications have to be executed. The computation of MFCCs involves matrix multiplications for creating the mel-scale spectrum; taking a logarithm and computing a discrete cosine transform (DCT). Considering only the required amount of hardware while not yet focusing on memory constraints, the number of required logic elements (LEs) and fixed hardware blocks may increase the amount provided by the FPGA. Table 3-1 and Table 3-2 list the available amount of hardware at the Stratix II EP2S60 [41] and the required amount of hardware for the mentioned types of single-precision floating-point operations [40], respectively.

Stratix II EP2S60			
LEs	**M4K**	**RAM bits**	**DSP – 18x18**
60.440	255	2.544.192	144

Table 3-1 Available hardware and memory resources at FPGA.

Operation	**LEs**	**M4K**	**RAM bits**	**DSP – 18x18**
Sine/cosine	1.830	-	2.190	16
Logarithm	1.387	-	1.904	8
FFT (512-pnt)	60.440	255	2.544.192	128
Real inverse (32x32)	15.655	200	699.164	118

Table 3-2 Required hardware and memory resources of basic elements.

It is clear that the amount of required hardware does not match the properties of the used FPGA, left aside the fact that the current design already uses 24% of the available hardware resources. Of course it is possible to look for a different FPGA that does meet the required properties. However, the limited time to come with a working solution provides very little space in overheads like ordering time, conversion of the current design to a different FPGA etc. Also note that the hardware requirements are highly related with the beamforming algorithm and therefore were known in a later stadium of

this project. Another issue is the ability of applying changes to the design. The generated sound parameters, for example, are a preliminary set of parameters used for sound classification. If other parameters seem to be more useful in a later stadium, changing the design is a time demanding process. The FPGA implementation reduces flexibility, which can be an important criterion in the development of the classification algorithm.

Taking above arguments into consideration, the FPGA as a platform for developing the system would be a rather inconvenient choice. A PC implementation seems the best solution referring to flexibility. When choosing a PC implementation, for this particular project we have to take into account the possible deployment of the application on a master system. As mentioned in the introduction, this master will deal with a number of other applications. Though we may require certain hardware facilities, this might not guarantee a fixed amount of memory and CPU resources for our application as this is dependent on the hardware utilizations of the other running applications, which might not be that strictly formulated. Further, it is possible the application may be used independently for other goals, outside the purpose of the project it is dedicated to. Conclusively, to still meet the real-time requirement without demanding strict hardware facilities, a flexible, robust system must be designed that takes care of changing processing circumstances. A system is developed taking this requirement into account. The next paragraph describes the architecture and composition of the system.

3.2 System overview

To get an overall picture of the different elements in the application, this paragraph will briefly describe the functionalities of the different parts in the final solution. Parts with more significant functionality will be described in more detail in their own paragraph.

Figure 3-1 depicts a rough overview of the system.

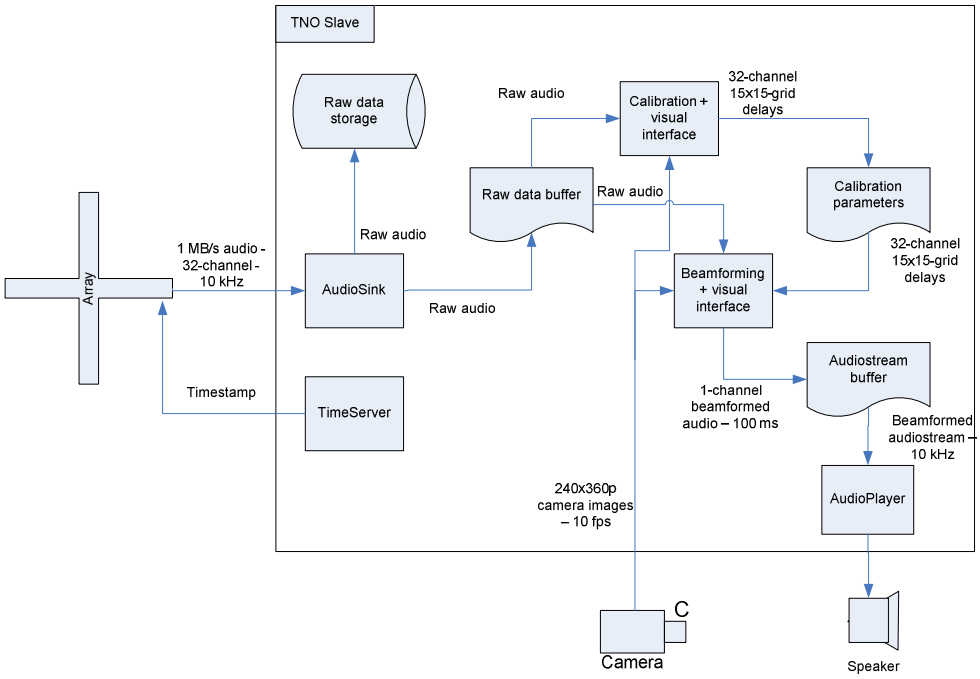


Figure 3-1 System overview.

3.2.1 Array

The 32-element array comprises an FPGA that samples audio with a sample frequency of 10 [kHz] and a resolution of 24 bits per sample. It buffers data packets of 100 samples per microphone using a ping-pong[6] buffer and signals the microcontroller on a full buffer. The microcontroller, on its turn, sends the data in the buffer to the audio slave over an Ethernet connection.

3.2.2 TimeServer

The TimeServer synchronizes the time of the array with the time of the audio slave. Every ten seconds it sends a timestamp to the array with which the array corrects its clock. In this way, an accurate sample frequency of 10 [kHz] is met. The TimeServer is implemented in C++.

3.2.3 AudioSink

The AudioSink represents the audio data interface between the array and the audio slave. It continuously listens to the Ethernet connection established with the array and receives its data packets. On every reception of a data packet, it writes the raw data to a harddisk for permanent storage and to the raw data buffer that is meant for using in the beamforming and calibration application. The AudioSink is implemented in C++.

3.2.4 Raw data storage

The raw data storage ensures that all data is saved in order to be able to reprocess all the recorded audio data afterwards. In this way, it is possible to look/hear back any events that occurred during a recording session. Also, there is the opportunity of reprocessing the data with a different beamforming algorithm than the one used during the real-time execution.

3.2.5 Raw data buffer

The raw data buffer temporarily stores 32 streams of raw audio data, one for every sensor. The AudioSink writes every newly received data packet and signals its reception by updating the latest received stream number in the buffer. In this way the applications using this data, get notice of the arrival of a new stream. The buffer has a length of two seconds; after this, old data is overwritten. By choosing a length of two seconds, the calibration and beamforming applications are able to fetch ‘older’ data when they might not be able to keep up with the data stream due to a decrease in execution speed. This decrease can be caused by other applications temporarily requiring more CPU-usage.

3.2.6 Calibration and visual interface

The calibration takes care of delivering the correct filter vectors for the beamforming application. This procedure is executed only once for every deployment location. It consists of an interface with which the user is able to determine all the relative delays between the sensors and each position in the area to scan. Due to the highly interfaced structure of the calibration procedure, it is implemented in MatLab. Later in this chapter, a more detailed description on this procedure is given.

---

[6] A buffer construction using two buffers of which only one is filled at a time. After one buffer is full, it is ready for getting its data transmitted while the other of the two buffers is filled. In this way there is always a complete version of the data and possible transmission delays (up to a certain level) do not cause data loss.

3.2.7 Calibration parameters

The calibration parameters are stored in a file that is used by the beamforming application. The parameters consist of all relative sensor-delays of the complete scan area, together with the grid size. A user can decide to use these parameters or overwrite them to recalibrate for a new location.

3.2.8 Beamforming and visual interface

The beamforming application contains the major part of the system. It consists of a user interface and the implementations of the applied beamforming algorithms. It constructs all the beams for the complete scan-area and computes the required sound classification parameters based on this information. This all is packed in a developed shell that takes care of its real-time operation. The entity contains a user interface that is able to show an overlay of the acoustic field and camera images and lets the user set a number of parameters. Moreover, it keeps the user up to date about the performance of the system. The beamforming application also writes the constructed beam of one specific, user-selected location or the output of one sensor to an audio stream buffer and signals it with the new stream number. Control and (user) interfacing is implemented in MatLab for convenient interface control. The demanding beamforming algorithms, on the other hand, are implemented in C-functions and are provided a MatLab interface. Later in this chapter, the beamforming application will be discussed in more detail.

3.2.9 Audiostream buffer

The audio stream buffer contains a single-channel 16-bit audio stream that represents the reconstructed audio data of a beam or a sensor. It is a buffer of only 100 milliseconds that is updated by the beamforming application.

3.2.10 AudioPlayer

The developed AudioPlayer is provided an interface with the sound card of the audio slave. It continuously checks the audio stream buffer for a change in stream number. On every change of stream number, it reads the data from the file together with the accompanying requested playing volume. Based on the requested playing volume and a clipping value, it scales the audio and fills the audio buffer of the sound card, which plays the sound with a sample frequency of 10 [kHz].

3.2.11 Camera

The connected Ethernet-camera delivers camera images that are used in the calibration and beamforming procedures. The camera is not a required part of the system but is indispensable in calibration and provides a visual advantage in the beamforming user interface. It has a user-defined resolution which is downgraded in the beamforming application for performance reasons.

3.3 Calibration and visual interface

As indicated in the system overview, calibration of the array with respect to the deployment location is necessary before actual beamforming can be applied. Calibration in this sense means that the inter-sensor delays have to be determined for every grid point. We derive and compare two methods for doing this, after which the calibration user interface for the chosen method will be described.

3.3.1 Approach

It is possible to determine the calibration delays geometrically. Figure 3-2 [15] depicts the Cartesian coordinate system with a beam of length R that is defined with an azimuth

and elevation angle θ and ϕ respectively. In this figure, the array is located in the XZ-plane.

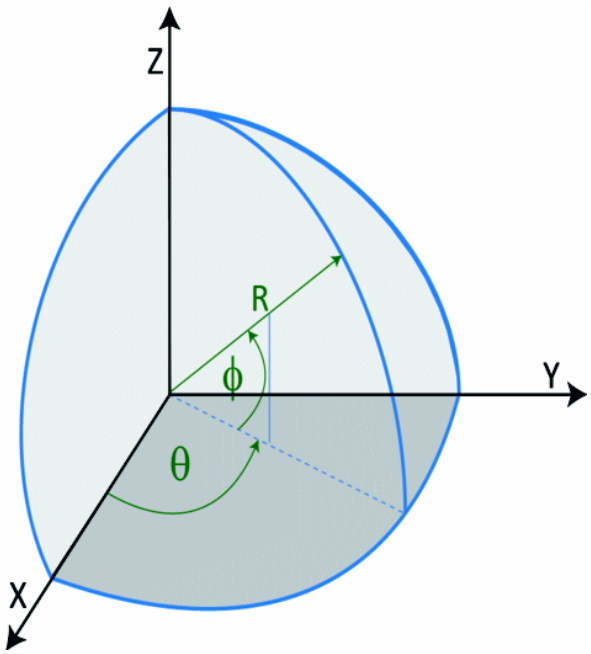


Figure 3-2 Azimuth and elevation in Cartesian coordinate system.

Again, the purpose is to determine the delays like done for a ULA in paragraph *2.1.1 Delay and Sum* but now for a 2D cross-array scanning a surface instead of a line. By using the discussed far-field assumption, only the beam angles with respect to the center of the array are used. Define a sensor position as an x-, y- and z-coordinate with, for our 2D-array, the y-coordinate being zero for each sensor. To determine the relative inter-sensor distance, we project the sensors on the beam from the source through the center of the array. The virtual, 1D sensor position on this beam then is given by:

$$pos = -x\cos\theta\cos\phi + z\sin\phi \tag{3.1}$$

Applying above formula comes with a practical issue. To determine the mentioned projections, the exact sensor positions have to be known. Small differences between the assumed and actual positions will have a negative influence on the spatial response. The same goes for a possibly fixed phase offset between sensors. Moreover, the exact position and orientation of the array have to be known, making the system less portable for, for example, hanging constructions.

A solution to this is to actively determine the delays in-field. In this case the array can be maneuvered in any position and orientation as long as it stays fixed during application. Phase differences between the sensors are directly taken account for. The idea is that a sound source located at a grid point creates a sharp impulse that is detected by the sensors. By determining the differences between the arrival times of the signal peak between the sensors, the grid point can be associated with that set of delays. By doing this for a number of grid points and interpolating the delays for the rest of the grid, the complete area can be covered. Note that a possible echoic environment does not influence this type of calibration as the signal peak/trigger in the (fastest) direct path is detected first. Figure 3-3 shows the signals and trigger points of three sensors on exertion of an impulse from a grid point; the derivation of the delays speaks for itself.

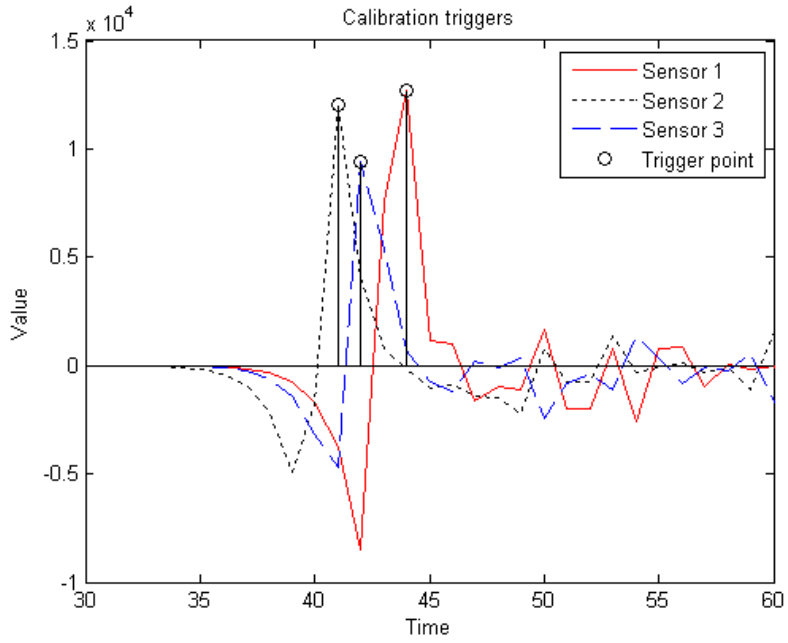


Figure 3-3 Triggers of three sensors when generating impulse at a random grid point.

The sharp peak is generated by hitting two triplex plates onto each other.

3.3.2 Procedure

The method used for generating a calibration peak is integrated in a real-time user interface that combines the detection of the peaks and the association with grid points by the provision of camera images. This procedure will be discussed using the interface screenshot in Fig. C-1 in appendix C.

The software provides a continuous audio stream of all 32 sensors. In *A*, a single channel (sensor output) is plotted to get an idea of how the sensor signal looks like and to verify whether there are non-functioning sensors. The sensor that is plotted can be selected by the user. Together with the sensor output, the current threshold level of that particular sensor is plotted.

The software continuously checks for each sensor stream whether it exceeds its threshold level. If this is the case, it locates the exact peak and generates a trigger for this particular sensor. A generated trigger will be indicated as a green bullet in *B*. Here, the total number of triggered sensors is visualized. The user is able to define the minimum number of triggered sensors before a 'complete trigger' is forced. This number is not fixed to the total number of sensors, because this allows ignoring defect sensors. If no complete trigger is forced, the interface reminds the triggers for a few seconds to enable the user to observe which sensors are lagging. The sensor number indicated in red is the so-called reference sensor. All delays are computed relative to this sensor. This sensor is selected by the user in advance. This means that an extra requirement for a complete trigger, is that at least the reference sensor has to be triggered; otherwise no relative delays can be computed.

A thing already mentioned before, is that the array consists of low-cost sensors. A consequence of this is that they differ in sensitivity. To account for this and for the overall generated calibration sound level, the user can adjust the threshold level for all sensors or for only one particular sensor in *C*. If one or more sensors do not get triggered, in *A* it is easy to see how high the threshold level should be.

If a trigger is detected, the application is 'frozen' and the screen in Fig. C-2 in appendix C will show up. This screen shows plots of all the sensor outputs in the near

vicinity of the timestamp of its trigger, together with the assumed trigger point. This extra screen is added for safety purposes: the user can verify whether the triggers have been detected correctly and decide to use or ignore the trigger.

The possibility exists that a clap with the sound source exceeds the maximum output range of a sensor. The application notifies the user with the sensors that clipped. Figure 3-4 shows why an overshoot makes the exact determination of the trigger time inaccurate: the location of the exact peak cannot be determined, leading to an offset in the relative delays.

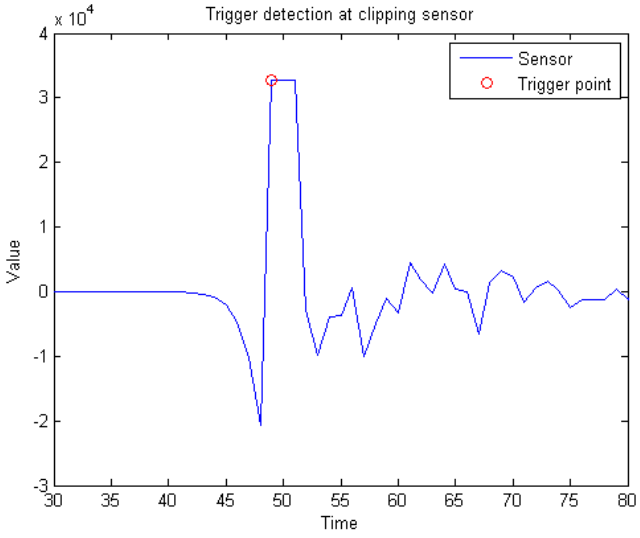


Figure 3-4 A clipping sensor with its detected trigger point.

If the user confirms the trigger, it can assign the exact location of the calibration point in the frozen camera screen *D*. The application then saves the coordinates of the calibration point, indicated by the red cross, and computes all the relative sensor delays. For convenience, all the calibration points remain plotted on the camera screen to obtain a clear overview of the calibrated grid.

E is added to account for a slightly delayed camera stream. On a trigger, the software creates a one-second buffer of camera images. With this buffer, the user can forward the camera stream to the exact point of triggering to obtain the correct calibration location.

After a number of calibration points have been obtained, the user can decide to finish calibration. At this point, the software will compute the relative delays of all the other points in the grid by means of interpolation. The grid will match and cover the complete camera screen. This means the grid points and therewith the beams are related to the camera position and orientation used with calibration. It may be possible to relate them with a camera in an other position by projecting the grid points on a virtual surface with known, fixed locations and translating it to an other camera.

When the calibration is finished, the real-time beamforming application is ready to start.

3.4 Beamforming and visual interface

3.4.1 Implementation and interfacing

The execution of real-time beamforming is the main element in the system. It uses the computed delay vectors of the calibration procedure for reconstructing the power spectra of the sound emitted from every grid point in the scan area. The current

application uses 225 (15 x 15) filter vectors to recover the sound field of a 2D-grid. With the obtained power spectra, the MFCC-coefficients, pitch- and RMS-values are computed for every beam. These parameters will be the input of the sound classification algorithms that will be developed in a later stage of the project. This paragraph will give a more detailed description of the beamforming module.

Besides the software that takes care of all the computations and control, a simple user interface is integrated with the application to provide some user interaction and audible and visual information. Like in the description of the calibration procedure, this interface, depicted in appendix D, is used along with the explanation of the beamforming module.

The application provides three beamforming methods: Delay and Sum (DAS), static MVDR (SMVDR) and dynamic MVDR (DMVDR), having increased resolution respectively. DAS and SMVDR both have equal execution time as the appropriate filter coefficients are computed independent of data and therefore can be computed in advance. DMVDR has the highest computational demand. In *A,* the user is able to select the desired beamforming method. Applications or test sites that require only little spatial resolution or of which only information of higher frequencies is necessary, can make use of only DAS beamforming. If particular sound classification algorithms, for example, need higher resolution at lower frequencies, one could decide to enable SMVDR or DMVDR beamforming.

The software continuously checks the buffer file for incoming, raw audio streams. The streams are cut in frames; decoded to reconstruct the actual audio data and finally an FFT is performed to translate them to a frequency domain representation after which they are cut off to a frequency range of 350 to 1500 [Hz]. The length of a used audio frame is fixed to 100 milliseconds. It is cut in two frames of 50 milliseconds on which the FFT is performed. The spectra then are added together to form the spectrum of the 100 millisecond audio frame. The decision for this length is based on a balance between execution time and signal stationarity. As discussed earlier, DMVDR beamforming is a computationally complex algorithm. By increasing the frame length, there is more time for executing the algorithm. Because an increase in the amount of samples also gives an increase in the number of frequencies, the frame is split and added afterwards to remain with the same frequency resolution. This frequency resolution or bandwidth is chosen in accordance with an allowable maximum absolute beam angle deviation of 1.5 [°], as discussed in paragraph *2.1.1.3 Narrowband assumption*. This deviation accounts for the worst case: the lowest processed frequency at the maximum absolute viewing angle of 45 [°].

Using a frame of 100 milliseconds is not optimal. It depends on the type of sounds we are interested in, but, for example, speech would rather be considered stationary for an interval of 25 to 50 milliseconds. The same goes for other sounds like, for example, breaking glass. The effect of using a processing interval that is too long is that the obtained frequency spectrum is less specific/detailed. In other words: the influence of a frequency that is present for only 50 milliseconds is diminished in the obtained, beamformed signal due to the decrease in relative share in the complete frequency spectrum. This decreases the differentiation of a specific signal which will influence the anomaly detection in a negative way. Still, preliminary results have shown that the obtained frequency spectra of beamformed audio frames containing anomalous sounds, do possess sufficient differentiation with regard to non-anomalous sound frames. Extra research and testing must point out the validity of this decision. An option is to utilize a platform with more processing power to compensate for the loss in execution time.

Returning to the user interface, in *B* the user can select to plot the incoming video stream in *C*, possibly overlaid with a transparent plot of the reconstructed, acoustic field. The plotted acoustic field is represented by the RMS-values of the beams plotted in a dynamically regulated color distribution, ranging from red (high value) to blue (low value).

The slide bars in *D* are assisting in user-specific desires by defining the color scale interval. The threshold slide bar defines the default upper scale limit. This means that RMS-values equal to this limit will get the highest (red) intensity. The lowest intensity (blue) then is defined by this limit minus the user-defined range. RMS-values in between will be plotted by a color matching linearly with the color scale. RMS-values lower than the lower limit will all be plotted with the lowest intensity. RMS-values exceeding the upper level will dynamically lift the scaling interval. The new scaling interval then will be reminded for one extra frame to visually distinguish lower-intensity echoes. A higher RMS-value, however, will still overrule the reminded scale. Choosing a small range will narrow the visual area of a source, but also reduce the ability to see lower-intensity sources. Figure 3-5 summarizes above elucidation.

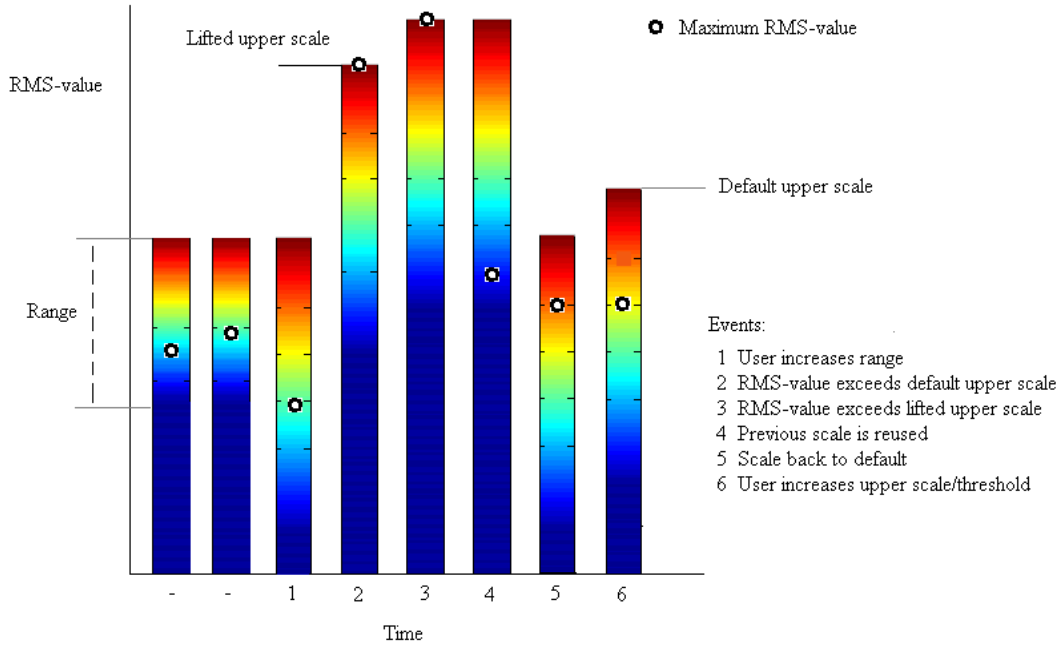


Figure 3-5 Dynamic RMS-colour scaling.

Dynamic scaling is applied to obtain more visual information, while on the other hand provide the ability to filter away ambient noise levels and only visualize very exceptionally high sound intensities. Up till now, this is the only visual part. After incorporating sound event detection, extra visual effects can be added in, indicating the occurrence and location of an event.

In the beginning of this chapter, we already introduced that the decision for a PC as implementation platform involves the possibility of a changing amount of available processing hardware and memory; both platform-varying and time-varying. Consider the use of DMVDR beamforming and the sudden decrease in available memory and/or assigned CPU-time due to other applications with (temporarily) increased demands. If the application would continue with its intensive tasks, it would not be able to keep up with the data rate and data would be lost after a while. Of course, the data buffer could be enlarged, but this would mean we would build up an increasing input-output delay, which is not allowed according to the system requirements. A solution to this could be

to switch to DAS or SMVDR beamforming. A more elegant way, however, is to dynamically adjust the number of frequencies processed by DMVDR and leave the rest to a simpler beamforming method. In this way, the system will deliver the best possible solution with the current provisions, instead of applying a binary decision for processing all frequencies with DMVDR or none. In addition to this, the system can run on almost every system, regardless of its hardware constraints. Of course, if a user provides a system meeting the proposed hardware constraints, the application can provide full DMVDR if the user prefers that.

Block *E* in the user interface shows the frequencies that are processed by DMVDR, defined as an interval F_l - F_h in [Hz]. The user can choose a fixed frequency interval by hand or decide to let the application drive the number of frequencies to its maximum. In the latter case it is still possible to shift the interval. In either case, the application is always in control of the number of processed frequencies. By determining the ratio between execution time and data rate, it is able to assess its performance. Performance ratios up to a certain limit will allow or enforce the addition of frequencies that are processed by DMVDR. When exceeding an upper limit, it will automatically switch back the number of frequencies. The system will drive the performance ratio up to about 90% to obtain some interaction time on changing circumstances. A 'rest' interval is used to avoid oscillation (see Figure 3-6). If the user does not select a fixed interval, the application will start adding in low frequencies because here the advantage of DMVDR is biggest. For user convenience, the ratio between execution time and data rate is visible in *F*.

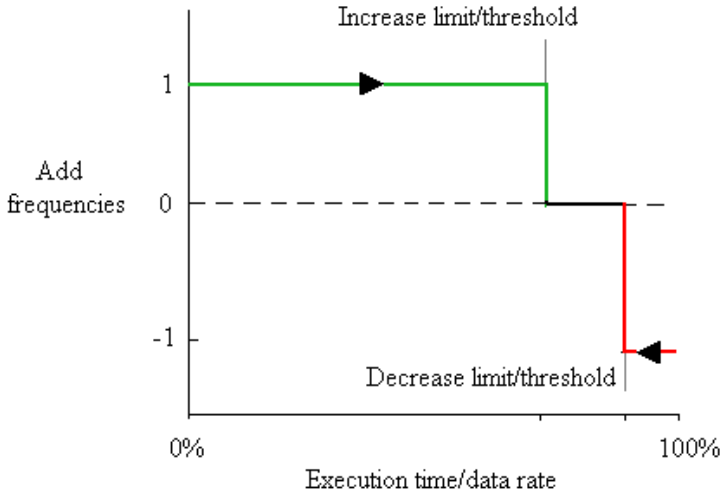


Figure 3-6 DMVDR frequency processing scheme.

With the last part of the interface, block *G*, the user is able to determine the type and volume of the audio stream that is constructed for the AudioPlayer. The usage speaks for itself. When deciding to listen to a beam, the user can press a point in the screen, indicated by the red cross. The reconstruction of the complete audio signal is performed by beamforming over the complete frequency spectrum (up to the Nyquist frequency) for that particular beam, disobeying the spatial aliasing constraint. The reason for this is the appearance of audible artifacts when cutting the signal at 1500 [Hz]. Still, alleviation is that listening to a beam is introduced for providing the ability of eavesdropping persons, i.e. listening to speech. Because most speech is within a frequency range of 500-1500 [Hz], the speech signal will only be marginally distorted by other speakers. Of course, there remains the possibility of interference by higher frequency sources at spatial aliasing positions. DAS is used as the beamforming method for audio reconstruction. The high resolution of DMVDR beamforming can filter the

desired speaker signal completely away on small speaker movements. Furthermore, the relatively low resolution of DAS beamforming allows listening to a small group of adjacent people which was indicated to be more useful than listening to only one person.

3.4.2 Performance and speed-up

Although the system is able to cope with an undesired or changing availability of hardware facilities, we still would like to provide an adequate solution on systems with 'average' hardware properties. Average properties in this sense can be defined as the properties of a relatively conventional personal computer. As a baseline platform, the complete system is tested independently on a 2.99 [GHz] dual-core Intel E6850 computer with 3.25 [GB] of RAM and on an Intel E8600 computer with a slightly faster dual-core of 3.33 [GHz]. On the 'slowest' of the two, the Intel E6850, the system was able to process the full beamforming spectrum using DMVDR and to still provide all user interfacing (plotting camera stream, plotting acoustic fields etc.) as outlined before. At this point, the system uses 74% of its maximum allowed execution time. Achieving such a performance can not be called quite effortless. Therefore, this paragraph will outline a few optimization highlights.

The part of the software that is most demanding is the beamforming procedure itself, which accounts for about 90% of the mentioned execution time. Therefore, the discussed optimizations will relate to this part. The DMVDR algorithm is implemented in C for speed-up purposes as MatLab as a script language lacks performance. In the discussion about complexity of DMVDR beamforming in paragraph *2.2.1.4 Computational complexity*, an estimate of computational complexity was made. To make this more concrete, in the current application we need about 8×10^7 multiplications, 8×10^7 additions and 3×10^4 divisions to process the frame of 100 milliseconds, neglecting the FFTs. The trick is to remove as much overhead as possible; make the operations that are executed most, take the least number of execution cycles and to make optimal use of the arithmetic facilities of the CPU. Furthermore, attempting to make efficient use of cache memory can considerably reduce memory latencies. Here are a few optimizations.

Loop unrolling

A simple way of reducing the amount of control overhead is (partly) unrolling for-loops. In this way loop-index incrementation and the number of checks for re-entering the loop are diminished. To remain having a balance between generality and optimization, a loop that has to be traversed 32 times, one time for each sensor, is unrolled to be traversed two times: one time for each 16 sensors. This makes the use of larger, acoustic arrays easier, though still requiring the array to contain a multiple of 16 sensors. Code 3-1 shows an example code snippet.

```
for(n=0; n<numberOfSensors; n++){
  <operation>;
}
```

```
assert(numberOfSensors%16==0);

for(n=0; n<numberOfSensors/16; n++){
  <operation>; /*1*/
  <operation>; /*2*/
        …
  <operation>; /*16*/
}
```

Code 3-1 Change from generic loop (above) to unrolled loop (below).

Enforcing column-major accessing
The delay vector matrix used for filtering has three dimensions: the number of sensors, frequencies and beams. They are passed through a MatLab interface to the C-function where the other side of the interface treats them as one long array. By choosing a convenient arrangement of the dimensions of the matrix in advance, we can step through the array without performing index translations. This saves index computations, but also makes efficient use of cache memory as almost all caches fetch complete blocks of memory-adjacent variables from memory.

Output pre-allocation
Instead of allocating memory for the DMVDR output in the C-function on every call, memory is pre-allocated at start-up of the system and the (empty) output variable is passed to the C-function by reference. This relieves the processor from searching for and allocating available memory on every function call.

Arithmetic distribution
The processor contains multiple units in the ALU (Arithmetic Logic Unit) that can perform the same specific operation, e.g. multiplication, addition etc. Ideally one would like all the necessary ALUs to be filled continuously. It may be possible that the compiler optimizes the code in such a way that this is already achieved. Still, the code tries to enforce this in two ways: by enforcing operation parallelism and at the same time avoid pipeline stalling due to structural[7] and data hazards[8]. Operation parallelism is enforced by including multiple operations of the same type in one statement. Finally, by mixing different types of operations in one statement, all the different types of ALU-units are used at the same time. If, on the other hand, first all additions would be executed, the addition units would be overloaded (structural hazard) and the other units must wait for the results of those operations to use as their inputs (data hazard).

[7] Structural hazards are hazards that occur when part of the hardware is needed by multiple instructions at the same time.
[8] Data hazards are hazards that occur when one or more instructions have to wait for data generated by a previous instruction.

```
for(n=0; n<noIters; n++){
  value1 = a[n] * a[n];
  value2 = b[n] * b[n];
      …           …
  value12 = m[n] * m[n];

  out[n] = value1 + value2;
  out[n] += value3;
      …     …
  out[n] += value13;
}
```

```
for(n=0; n<noIters; n++){
    out[n] = a[n] * a[n] + b[n] * b[n] + .. + h[n] * h[n];
    out[n] += i[n] * i[n] + j[n] * j[n] + .. + m[n] * m[n];
}
```

Code 3-2 Enforcing execution parallelism and avoiding data hazards. Above: before. Below: after.

Code 3-2 gives a very simple example to point out the approach to achieve this.

Depending on the type of processor, this may already be partly taken care of by static and dynamic instruction scheduling[9]. Still, to try to make the execution time independent of processor type (except for the amount of hardware facilities), this is already tried to be manipulated.

Compiler type

The integrated compiler of MatLab is able to compile C-functions with a MatLab-interface. It is not very good in performing code optimizations though. The use of a different compiler provides the ability to scrape away the last code deficiencies, though the influence is not very significant anymore. The GCC-compiler with MatLab integration and a preset maximum optimization level is used for this purpose. A compiler dedicated for the type of processor can optimize the code even further, but is deliberately not chosen. A negative property of these compilers is that when deploying the compiled code on a different type of processor, execution time can increase dramatically. Because we are not yet aware of the final deployment platform and to keep the solution available for other platforms in a later stadium, this could be risky.

3.5 Evaluation

The developed system has been verified and tested in three different environments: in an anechoic room, on the street and in a reverberant foyer. This paragraph will first briefly reflect on the requirements, postulated in the introduction and will finally summarize the observations made concerning beamforming performance.

The delivered system meets the requirements mentioned in the introduction, by providing MFCCs, pitch frequencies and RMS-values, considering the complete beamforming frequency spectrum of 350-1500 [Hz]. All these parameters are computed 10 times per second for every one of 225 beams, covering a grid defined by the position

[9] Instruction scheduling is used to put the instructions in such an order that control, structural and data hazards are avoided. It can be performed statically (by the compiler) or dynamically at runtime (by the processor).

of the array in combination with an acoustical (and visual) azimuth and elevation opening angle of 90 [º].

The user is provided with an interface, showing a continuous stream of camera images, overlaid with the acoustic field, constructed by the computed RMS-values. The user has the ability to select a sensor output or a specific point in space to which it can listen. Extra interface buttons provide audio volume control and visual acoustic field scaling capabilities.

Finally, referring to the important real-time aspect, the system is able to keep up with all incoming data from the array used for this particular project. Regarding the maximum allowed average input-output delay of 500 [ms], the system meets the requirements as can be seen in Table 3-3.

Availability of:	**Average I/O delay [ms]**	**Maximum I/O delay [ms]**
Parameters	170	360
RMS- and camera-plot	190	380
Audible audio signal	390	580

Table 3-3 Measured average and maximum input-output delays.

The delays are measured on the reference systems, listed in paragraph *3.4.2 Performance and speed-up*. The maximum input-output delays listed in the table, are observed very rarely due to the non-real-time properties of the deployment platform. This maximum delay may slightly increase and may occur more frequently on systems running other applications with strongly varying processing demands. Still, because the system dynamically increases the number of frequencies processed by DMVDR up to a 'safe' execution time limit, the influence of this is decreased to a minimum. Due to the adaptive structure of the system, the average delay will not increase when deploying the application on systems possessing less optimal hardware conditions.

The system is tested in an anechoic room. Scenarios with one and multiple coherent and non-coherent sources have been demonstrated giving beamforming results similar to those obtained in simulations. The only observed difference is that, due to the use of slightly noisy sensors, the maximum angular resolution decreases from 5 [º] to approximately 5.5 [º].

The application of the system in an asphalted area introduced one or two reverberation paths, depending on source position. The beamforming algorithms are still able to separate this total of three, possibly virtual, sources depending on source location, signal frequency and signal stationarity.

The application of the system in a foyer with a large number of reflective elements such as walls, floor and ceiling, deteriorate the ability of spatial filtering. The type of signal seemed to play an important role: highly stationary signals cause large variations in constructive and destructive interference between the direct source signal and reflections at particular points in space, sometimes making the created virtual source stronger than the original source. On the other hand, for less stationary signals like, for example, breaking glass, this effect vanishes. Reverberations are still noticeable, but highly attenuated and at a later point in time, providing the ability of filtering them away in sound classification and detection.

4 Conclusions and recommendations

This chapter will describe the conclusions that can be drawn from the previous chapters and will address some possible recommendations for future work.

4.1 Conclusions

A number of beamforming methods have been outlined and evaluated with respect to performance, complexity and practical implementation. These are some conclusions that can be drawn:

- Compared to conventional, frequency-invariant beamforming methods like DAS, it is clear that MVDR beamforming delivers a significantly increased, spatial resolution, especially at lower frequencies (<1000 [Hz]).

- Static MVDR has the advantage of having an equal computational complexity as conventional DAS beamforming, but at the cost of less interference reduction in noisy or multi-source environments.

- Dynamic MVDR actively tries to filter away interference by making use of the output signals of the sensors. This dynamic filter increases spatial resolution but also increases computational complexity.

- LCMV beamforming can deliver a contribution in actively filtering away interfering noise sources, but needs additional information on interferer locations which, in this application, is an impractical requirement.

- The use of adaptive filtering methods (AGSC) in order to alleviate complexity is not suitable: measures that have to be taken for reducing computational complexity, have a substantial negative effect on required convergence rates, making the application of it unacceptable. Especially with regard to the blocking filter, used as a first step in adaptive noise cancelling, high convergence must almost be a certainty for avoiding target signal cancellation.

The spatial structure of the current array limits the application of beamforming and parameter generation to a frequency bandwidth of 350-1500 [Hz]. The upper cut-off frequency is derived from the spatial aliasing constraint. The lower cut-off frequency relates to the ability of the array of distinguishing multiple coherent sources.

The developed system uses the optimal DMVDR method for beamforming, taking into account its complexity and the varying amount of available hardware resources by dynamically changing the number of frequency bands that are processed with this algorithm. Still, tests on a regular, 4 GB, 2.99 [GHz] dual-core PC, show full bandwidth processing ability with DMVDR beamforming, while still providing all user interfacing and parameter generation.

In-field system calibration avoids dependencies on array structure, positioning and orientation and consequently increases system portability. Also, the negative influence of flaws in future array designs, with respect to exact sensor positioning and phase differences between sensors, is prevented by using in-field calibration.

Finally, a system is delivered that meets the requirements postulated at the start of the project and is ready for use. It has retained both deployment flexibility and extendibility to specific purposes, making it a solid base for further development. The initial prototype with the present functionality already provides sufficient user information for utilizing it in a range of applications that solely aim at obtaining a preliminary image of the acoustic field and its parameters and require the ability of eavesdropping.

4.2 Recommendations

As indicated in the evaluation of the beamforming methods reflected on the currently used array, the structure of the array is not optimal. The spatial aliasing frequency of 1500 [Hz] may be too low for sufficiently supporting the classification of specific sounds. To increase this frequency, the sensor spacing must be decreased. A disadvantage of this is that the total array aperture will also decrease when the same number of 32 sensors is used. This will decrease resolution at lower frequencies. To keep a balance between the resolution at higher and at lower frequencies, a logarithmic sensor distribution is recommended: smaller sensor spacings near the center of the array and larger spacings towards the outside of the array. The required distance between the sensors at the center of the array of course depends on the desired upper cut-off frequency.

The application of the system in reverberant environments will have a negative effect on the spatial filtering ability. Signals exerted from one source will arrive at multiple time instances at the array and from different directions. This means that the same signal is detected from multiple directions at different points in time, depending on the room impulse response. It is also possible that the reverberations of relatively stationary signals will amplify or attenuate each other, such that one or more (possibly stronger) virtual sources are created at different locations.

To slightly diminish this effect, the use of directional microphones is advised. The current microphones are omni-directional, which introduces spatial aliasing at the backside of the array and increases the sensitivity to reverberations from the backside of the array. If there are no reflective elements behind the array, the use of omni-directional microphones will not cause any problems. In other cases, reducing the opening angle of the microphones to about 90 [°], will contribute in the decrease of the influence of reverberations. When doing this, it must be kept in mind that the array position and orientation still ensure coverage of the complete surface to scan.

If reverberations are still an issue, special anti-reverberation beamforming algorithms may provide a solution. These methods, however, mostly are computationally intensive, which is a burden for this particular application.

The current system is able to process the complete frequency spectrum with DMVDR on a regular PC and to provide all user interfacing and parameter generation. A possible reformation of the array will increase the total frequency bandwidth and therewith the required amount of computations. Also, routines added for sound classification and the possible reduction of the processing window of 100 [ms] will increase computational demands. If the application of one or more of these measures will significantly increase required execution time, with the consequence of a decreased number of frequencies processed by DMVDR, the beamforming algorithm can possibly be speeded up. The way of doing this is by making use of the Graphical Processing Unit (GPU). The GPU architecture is strongly structured towards parallel computing. It is used in applications that need to perform large amounts of the same operations as in, for example, video processing. The important difference with regard to a CPU is that a CPU uses more

hardware for memory as opposed to arithmetic while a GPU is structured oppositely (Figure 4-1 [42]).

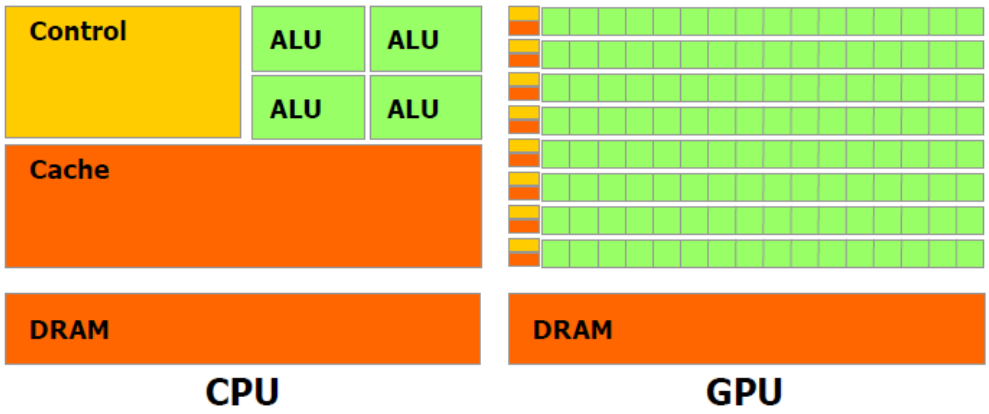


Figure 4-1 Processing architectures. Left: CPU. Right: GPU.

Compute Unified Device Architecture (CUDA) is a general purpose computing architecture that supports the use of the GPU for compute-intensive tasks. Dedicated libraries provide the interface between CUDA and applications written in C, C++, Fortran etc.

The early and experimental stage of this project did not yet allow the somewhat more complicated use of GPU programming. The currently achieved processing performance for this particular application neither demands a faster solution yet. However, in a future stage this may be desired.

References

1. Ryan, J.G., *Criterion for the minimum source distance at which plane-wave beamforming can be applied.* The Journal of the Acoustical Society of America, Volume 104, Issue 1, July 1998, pp. 595-598.
2. Abutalebi, H.R., Sheikhzadeh, H., Brennan, R.L., Freeman, G.H., *A Hybrid Subband Adaptive System for Speech Enhancement in Diffuse Noise Fields*. IEEE Signal Processing Letters, Volume 11, No. 1, Jan. 2004.
3. Bitzer, J., Simmer, K.U., *Superdirective Microphone Arrays*. Microphone Arrays: signal processing techniques and applications, May 2001, Springer, pp. 19-37.
4. Bitzer, J., Kammeyer, K.-D., Simmer, K.U., *An alternative implementation of the superdirective beamformer*. Proceedings IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, Oct. 17-20, 1999, New Paltz, New York.
5. Clark, G.A, Mitra, S.K, Parker, S.R., *Block Implementation of Adaptive Digital Filters*. IEEE Transactions on Acoustics, Speech and Signal Processing, Volume ASSP-29, No. 3, June 1981.
6. Romoli, L., Squartini, S., Piazza, F., *A Variable Step-size Frequency-domain Adaptive Filtering Algorithm for Stereophonic Acoustic Echo Cancellation.* 18th European Signal Processing Conference, Aug. 23-27, 2010, Aalborg, Denmark.
7. Lepauloux, L., Scalart, P., Marro, C., *Computationally Efficient and Robust Frequency-domain GSC.* International Workshop on Acoustic Echo and Noise Control, Aug. 30-Sep. 2, 2010, Tel Aviv, Israel.
8. Hertz, D., Mansour, D., Engel, I., *On Least Square Frequency-domain Adaptive Filters.* IEEE Transactions on Circuits and Systems, Vol. CAS-33, No. 3, March 1986.
9. *Fast Fourier Transform*. [cited Dec. 2011]; Available from: http://www.mathworks.nl/help/techdoc/math/brentm1-1.html.
10. Hoshuyama, O., Sugiyama, A., *Robust Adaptive Beamforming*, Microphone Arrays: signal processing techniques and applications, May 2001, Springer, pp. 19-37.
11. Clark, G.A, Mitra, S.K, Parker, S.R., *A unified approach to time- and frequency-domain realization of FIR adaptive digital filters*. IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-31, No. 5, Oct.1983.
12. *Planar waves.* [cited Aug. 2011]. Available from: http://en.wikipedia.org/wiki/Plane_wave
13. Cheng, J., Phua, K., Shue, L., Sun, H., *A Robust Adaptive Cross Microphone Array.* IEEE Transactions on Signal Processing, Vol. 47, No. 10, Oct.1999.
14. Hoshuyama, O., Sugiyama, A., Hirano, A.,*A Robust Adaptive Beamformer with a Blocking Matrix Using Coefficient-constrained Adaptive Filters*, IEICE Transaction Fundamentals, Vol. E82-A, No. 4, Apr. 1999.
15. *Cartesian coordinate system.* [cited Dec. 2011]; Available from: http://www.mathworks.com/help/toolbox/phased/ug/sphericalcoordinateshading.gif
16. Chang, S.-H., Chang, C.-C., *The Application of Wavelet-based Least Mean Square Algorithm in Adaptive Beamforming*, Journal of Marine Science and Technology, Vol. 5, No. 1, 1997.
17. Joho, M., Moschytz, G. S., *Adaptive Beamforming with Partitioned Frequency-domain filters*, Applications of Signal Processing to Audio and Acoustics, Oct. 19-22, 1997, New Paltz, USA.
18. Okuma, Y., Suzuki, Y., Murakami, T., Ishida, Y., *A Fast Directionally Constrained Minimization of Power Algorithm for Extracting a Speech Signal Perpendicular to a Microphone Array*, International Journal of Information and Communication Engineering, Vol. 4, No. 2, 2008.

19. Koretz, A., Rafaely, B., *Dolph-Chebyshev Beampattern Design for Spherical Arrays,* IEEE Transactions on Signal Processing, Vol. 57, No. 6, June 2009.
20. Elminowicz, A., *Efficient Wideband Beamformer in the Frequency Domain for Linear Array Sonar*, Molecular and Quantum Acoustics, Vol. 32, 2002.
21. *Nyquist sampling theorem.* [cited, Nov. 2011]; Available from: http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem
22. Dmochowski, J., Benesty, J., Affes, S., *On Spatial Aliasing in Microphone Arrays*, IEEE Transactions on Signal Processing, Vol. 57, No. 4, Apr. 2009.
23. Luszczyk, M., Mucha, D., *Kaiser-Bessel Window Weighting Function for Polyphase Pulse Compression Code*, IEEE Microwaves, Radar and Wireless Communications, May 19-21, 2008.
24. Zhang, L., Liu, W., Yu, L., *Performance Analysis for Finite Sample MVDR Beamformer with Forward Backward Processing*, IEEE Transactions on Signal Processing, Vol. 59, No. 5, May 2011.
25. McCowan, I. A., Bourlard, H., *Microphone Array Post-filter Based on Noise Field Coherence*, IEEE Transactions on Signal Processing, Vol. 11, No. 6, Nov. 2003.
26. Rafaely, B., *The spherical-Shell Microphone Array*, IEEE Transactions on Audio, Speech and Language Processing, Vol. 16, No. 4, May 2008.
27. Choi, S., Choi, J., Im, H.-J., Choi, B., *A Novel Adaptive Beamforming Algorithm for Antenna Array CDMA Systems with Strong Interferers*, IEEE Transactions on Vehicular Technology, Vol. 51, No. 5, Sep. 2002.
28. *Optimal beamforming methods.* Available at: *http://www.comm.utoronto.ca/~rsadve/Notes/BeamForming.pdf*
29. Breed, B.R., Strauss, J., *A short proof of the equivalence of LCMV and GSC Beamforming,* IEEE Signal Processing Letters, Vol. 9, No. 6, June 2002.
30. Lee, J.-H., Cho, C.-L., *GSC-based Adaptive Beamforming with Multiple-beam Constraint Under Random Array Position Errors*, Elsevier Computer Science, Signal Processing 84, 2004.
31. Joo, I., Choi, S., Kim, K., *Performance of Adaptive Beamforming using the Split RLS Algorithm*, Proceedings of the IEEE Region 10 Conference, Vol. 2, Sep 15-17, 1999.
32. Mandic, D.P, Hanna, A.I., Razaz, M., *A Normalized Gradient Descent Algorithm for Nonlinear Adaptive Filters Using a Gradient Adaptive Step Size*, IEEE Signal Processing Letters, Vol. 8, No. 11, Nov. 2001.
33. Juntti, M., [cited Nov. 2011], *Communication Signal Processing – Matrix Algorithms*, University of Oulu, Dept. Electrical and Information Engineering.
34. Gu, Y., Jin, J., Mei, S., *Norm Constraint LMS Algorithm for Sparse System Identification*, IEEE Signal Processing Letters, Vol. 16, No. 9, Sep 2009.
35. Shynk, J.J., *Frequency-domain and Multirate Adaptive Filtering,* IEEE Signal Processing Magazine, Vol. 9, No. 1, Jan. 1992.
36. Jiang, Z., Huang, H., Yang, S., Lu, S., Hao, Z., *Acoustic Feature Comparison of MFCC and CZT-based Cepstrum for Speech Recognition,* 5th International Conference on Natural Computation, Aug. 14-16, 2009, Tianjin.
37. Kelkar, S.S., Grigsby, L.L., Langsner, J., *An Extension of Parseval's Theorem and Its Use in Calculating Transient Energy in the Frequency Domain,*IEEE Transactions on Industrial Electronics, Vol. IE-30, No. 1, Feb. 1983.
38. Hasan, R., Jamil, M., Rabbani, G., Rhaman, S., *Speaker Identification Using Mel Frequency Cepstral Coefficients,* 3rd International Conference on Electrical & Computer Engineering, Dec. 28-30, 2004, Dhaka, Bangladesh.
39. *Mel-frequency Cepstral Coefficients*, [cited Jan. 2012], Available from: *http://en.wikipedia.org/wiki/Mel-frequency_cepstrum*
40. Altera Corporation, *Altera Floating-point Megafunctions - User Guide 5.0*, May 2011, San Jose, USA.

41. Altera Corporation, *Nios Development Board Stratix II Edition – Reference Manual 1.3,* May 2007, San Jose, USA.
42. NVIDIA Corporation, *Cuda C Programming Guide 4.0*, June 2011, Santa Clara, USA.
43. Intel Corporation, *Intel 64 and IA-32Architectures Optimization Reference Manual*, Vol. 025, June 2011, Santa Clara, USA.

A Array structure

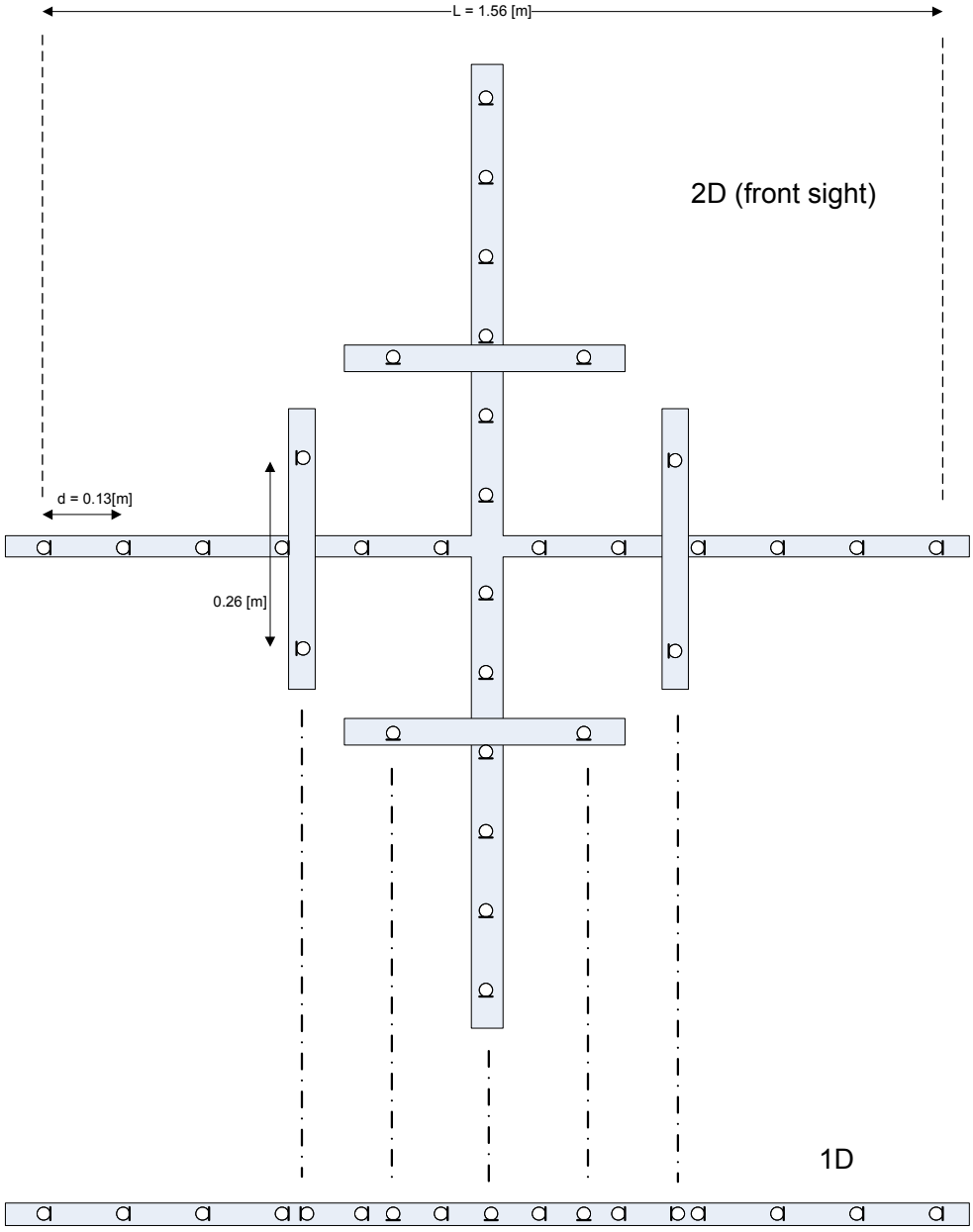


Figure D-1 Array structure. Above: complete 2D structure. Below: 1D-projection.

B Gauss-Jordan inversion complexity

Gauss-Jordan elimination for the construction of a matrix inverse is performed by applying a set of elementary row operations both on the matrix to invert and the identity matrix, such that the original matrix is transformed to the identity matrix. Mathematically:

$$\left(A\middle|I\right)\rightarrow\left(I\middle|B\right) \text{ with } B=A^{-1} \tag{B.1}$$

With A being the matrix to invert, I the identity matrix and B the inverse of A, all having the same (square) dimensions. To derive the number of operations involved, an example will be described. Consider the inversion of matrix A like in Eq. (B.1) such that the initial augmented matrix is defined as:

$$\left(A\middle|I\right)=\left(\begin{array}{ccc|ccc}1&1&2&1&0&0\\2&4&-3&0&1&0\\3&6&-5&0&0&1\end{array}\right)=B[0] \tag{B.2}$$

With $B[n]$ denoting the augmented matrix after step n. For every step, the number and types of operations will be given next to the equation. To reduce the number of operations in the next step, the first step is to divide row R1 by its main diagonal entry. Though this diagonal entry is equal to one in this particular example, this is always the initial step and has to be catered for:

$$B[0]\xrightarrow{R1=\frac{R1}{1}}\left(\begin{array}{ccc|ccc}1&1&2&1&0&0\\2&4&-3&0&1&0\\3&6&-5&0&0&1\end{array}\right)=B[1] \tag{B.3}$$

ADD: -
MUL: -
DIV: N

The second step is to obtain zeros at the entries in the first column that are not on the main diagonal:

$$B[1]\xrightarrow[R3=R3-3R1]{R2=R2-2R1}\left(\begin{array}{ccc|ccc}1&1&2&1&0&0\\0&2&-7&-2&1&0\\0&3&-11&-3&0&1\end{array}\right)=B[2] \tag{B.4}$$

ADD: N(N-1)
MUL: N(N-1)
DIV: -

Again, the next row is divided by its diagonal entry:

$$B[2]\xrightarrow{R2=\frac{R2}{2}}\left(\begin{array}{ccc|ccc}1&1&2&1&0&0\\0&1&-\frac{7}{2}&-1&\frac{1}{2}&0\\0&3&-11&-3&0&1\end{array}\right)=B[3] \tag{B.5}$$

ADD: -
MUL: -
DIV: N

$$B[4] \xrightarrow{R3=R3-3R2} \left(\begin{array}{ccc|ccc} 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & -\frac{7}{2} & -1 & \frac{1}{2} & 0 \\ 0 & 0 & -\frac{1}{2} & 0 & -\frac{3}{2} & 1 \end{array}\right) = B[4] \qquad \begin{array}{c}\text{ADD: } N(N\text{-}2) \\ \text{MUL: } N(N\text{-}2) \\ \text{DIV: -}\end{array} \tag{B.6}$$

By dividing the last row by its diagonal entry, the row echelon form becomes:

$$B[4] \xrightarrow{R3=\frac{R3}{-1/2}} \left(\begin{array}{ccc|ccc} 1 & 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & -\frac{7}{2} & -1 & \frac{1}{2} & 0 \\ 0 & 0 & 1 & 0 & 3 & -2 \end{array}\right) = B[2N-1] \qquad \begin{array}{c}\text{ADD: -} \\ \text{MUL: -} \\ \text{DIV: } N\end{array} \tag{B.7}$$

To generalize the above for an *NxN*-matrix, the number of additions equals the number of multiplications, defined by:

$$MUL/ADD = N\sum_{n=1}^{N-1}\left(N-n\right) = N\sum_{n=1}^{N-1} n = N\,\frac{N\left(N+1\right)}{2} = \frac{N^3}{2} + \frac{N^2}{2} \tag{B.8}$$

The number of divisions up till now can be defined by:

$$DIV = N^2 \tag{B.9}$$

To reduce matrix $B[n]$ further to the reduced row echelon form, first the non-diagonal entries of column C3 are forced to zero:

$$B[2N-1] \xrightarrow[R2=R2+\frac{7}{2}R3]{R1=R1-2R3} \left(\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & -6 & 4 \\ 0 & 1 & 0 & -1 & 11 & -7 \\ 0 & 0 & 1 & 0 & 3 & -2 \end{array}\right) = B[2N] \qquad \begin{array}{c}\text{ADD: } N(N\text{-}1) \\ \text{MUL: } N(N\text{-}1) \\ \text{DIV: -}\end{array} \tag{B.20}$$

The same step is applied for column C2:

$$B[2N] \xrightarrow{R1=R1-1R2} \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 2 & -17 & 11 \\ 0 & 1 & 0 & -1 & 11 & -7 \\ 0 & 0 & 1 & 0 & 3 & -2 \end{array}\right) = B[3N-2] \qquad \begin{array}{c}\text{ADD: } N(N\text{-}2) \\ \text{MUL: } N(N\text{-}2) \\ \text{DIV: -}\end{array} \tag{B.31}$$

The total number of multiplications and additions for this second part is then again defined as:

$$MUL/ADD = N\sum_{n=1}^{N-1}\left(N-n\right) = \frac{N^3}{2} + \frac{N^2}{2} \tag{B.42}$$

The combined total number of multiplications and additions then becomes:

$$MUL/ADD = N^3 + N^2 \tag{B.53}$$

The combined total number of division remains:

$$DIV = N^2 \tag{B.64}$$

This example is illustrated using real numbers and real operations. The possibility of complex numbers and operations has to be taken into account.

C Calibration interface

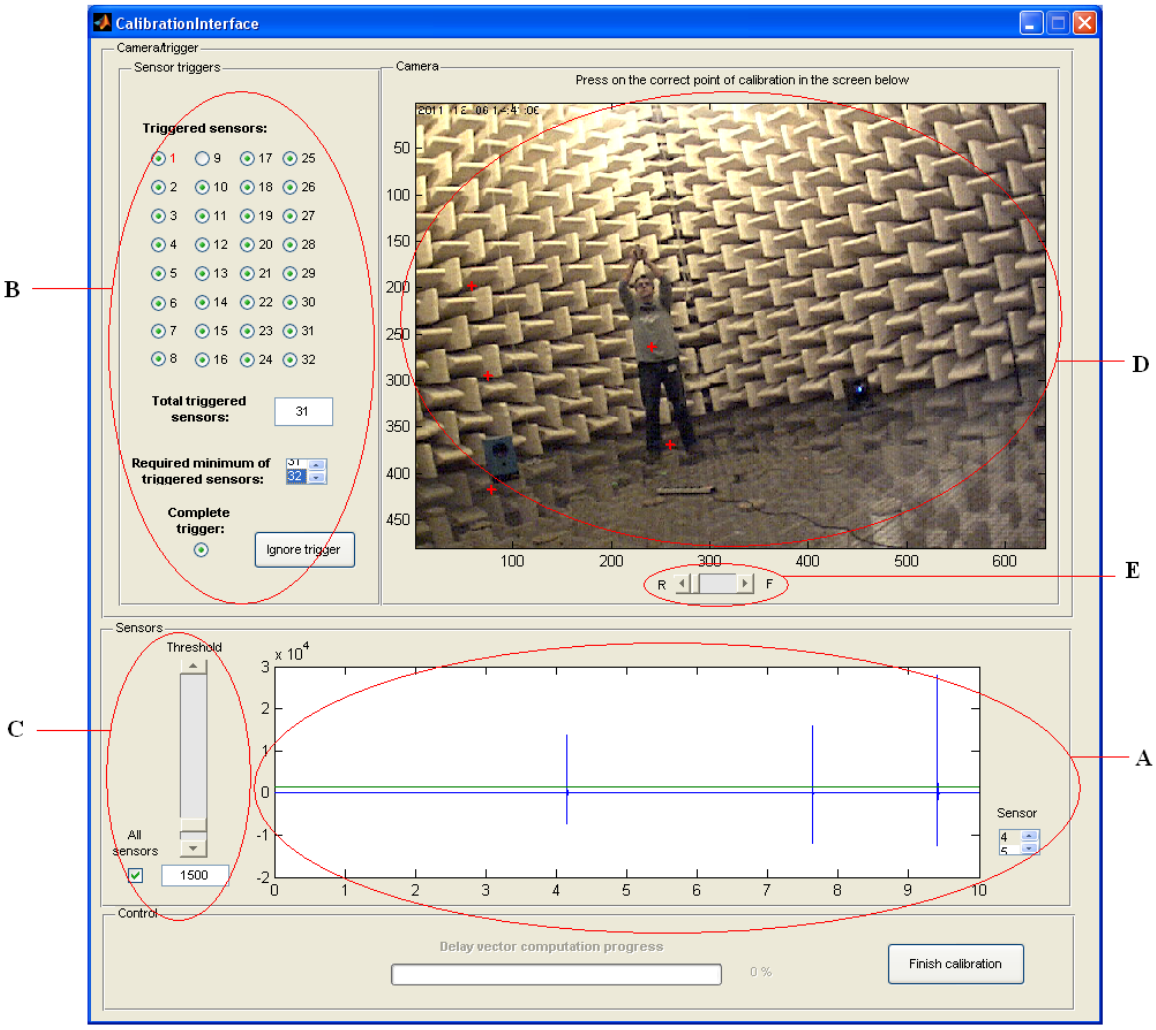


Figure C-1 User interface calibration screen.

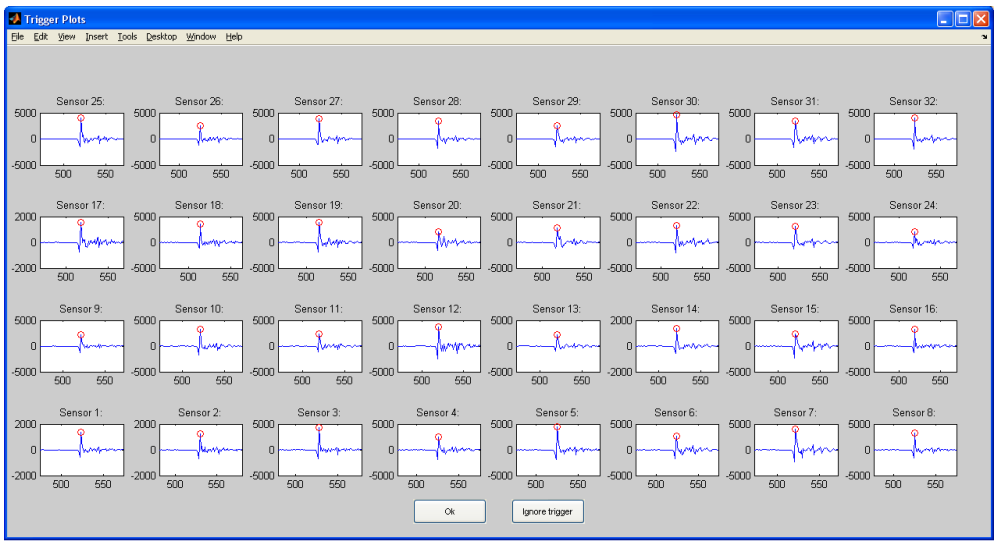


Figure C-2 Pop-up screen with generated triggers.

D Beamforming interface

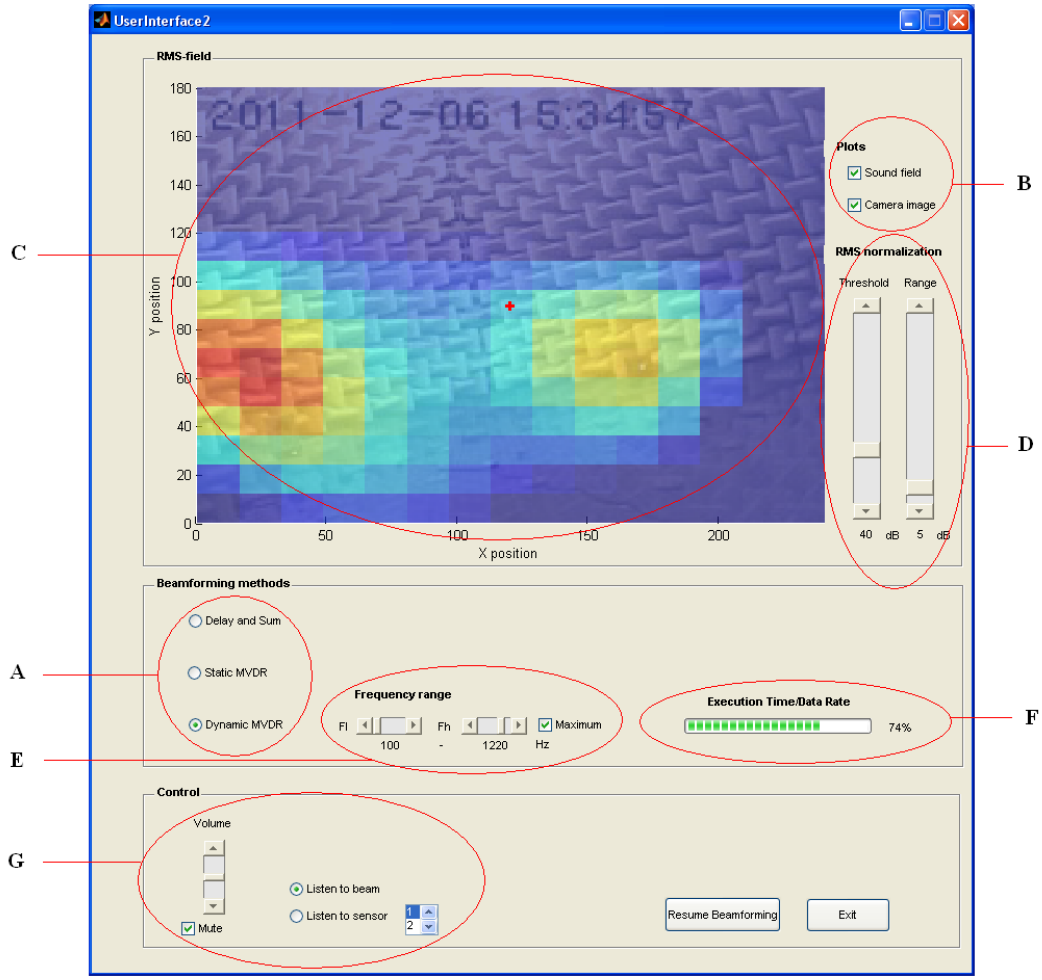


Figure D-1 User interface beamforming.

E Parameter computation

Mel-frequency Cepstral Coefficients
Mel-frequency Cepstral Coefficients (MFCCs) are coefficients that are based on exploiting the logarithmic property of the human auditory system. They are often used in speech and speaker recognition [36, 38]. For a general time-domain (windowed) audio frame, they are computed as follows [36, 38, 39]:

1. Take the Fast Fourier Transform.
2. Map the powers of (1) onto the Mel-scale, by filtering them with triangular overlapping windows with a logarithmic distribution as depicted in Fig. E-1. Integrating the outputs per filter delivers the Mel-scale powers.
3. Take the logarithm of the powers of (2).
4. Take the Discrete Cosine Transform of (3).
5. The amplitudes of (4) are the MFCCs.

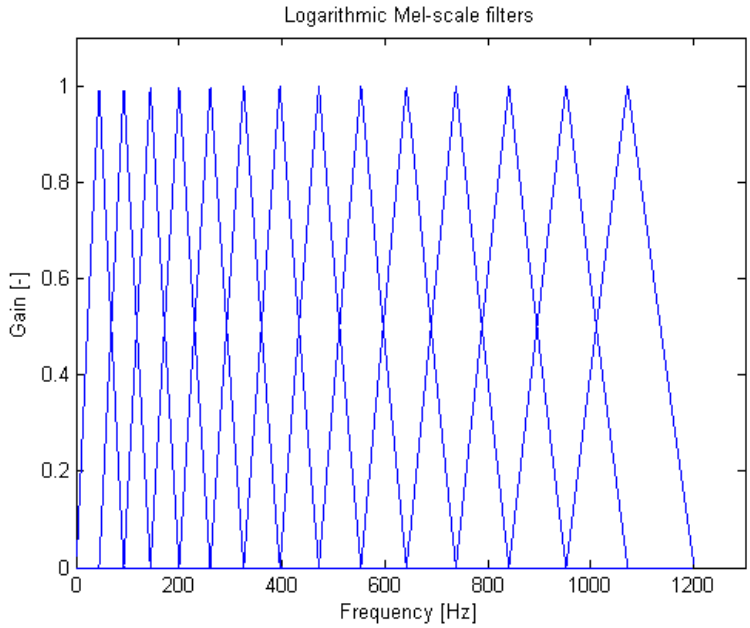


Figure E-1 Representation of the Mel-scale filters.

Pitch frequencies
The pitch frequency indicates the frequency with the highest power within the considered range of frequencies.

Root Mean Square-values
According to Parseval's theorem [37], the RMS-value of a signal can be determined in frequency domain according to:

$$RMS = \sqrt{\frac{1}{N_f}\sum_{n=0}^{N_f-1}\left|X(f_n)\right|^2} \tag{E.1}$$

With N_f being the number of outputs of the Fourier Transform and $X(f_n)$ being the output of the Fourier Transform of the signal at frequency f_n.