Fachhochschule Kärnten
Carinthia University of Applied Sciences
Systems Design

FACHHOCHSCHULE KÄRNTEN

MASTER THESIS

Evaluation of Beamforming Algorithms in Voice Assistant Devices

using Microcontroller-Powered Uniform Circular Microphone Arrays

Submitted in Partial Fulfillment of the Requirements of the Academic Degree

Master of Science in Engineering, MSc

Author: Michael Adolf Lamp, BSc

Registration Number: 1610528003

Supervisor: FH-Prof.in Dipl.-Ing.in Dr.in Ulla Birnbacher
Carinthia University of Applied Sciences

Second supervisor: Dipl.-Ing. Andreas Gaich
Johannes Kepler University Linz

Klagenfurt, July 2018

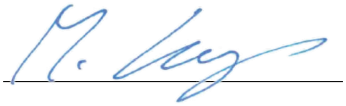
Statutory Declaration

I hereby declare that

- the Master's Thesis has been written by myself without any external unauthorized help and that it has not been submitted to any institution to achieve an academic grading.
- I have not used sources or means without citing them in the text; any thoughts from others or literal quotations are clearly marked.
- the enclosed Master's Thesis is the same version as the version evaluated by the supervisors.
- one copy of the Master's Thesis is deposited and made available in the CUAS library (§8 Austrian Copyright Law [UrhG]).

I fully understand that I am responsible myself for the application for a patent, trademark or ornamental design and that I have to prosecute any resultant claims by myself.

Klagenfurt, 31st July 2018



(Signature)

Abstract

The purpose of this thesis is to implement a typical signal processing chain on a microcontroller-powered cloud-based voice assistant device. The main focus is on finding a beamforming algorithm with the best trade-off between computational costs and noise suppressing performance.
The implemented beamforming algorithms are evaluated using the developed Cerebro-MultiMicBoard and the Google Assistant. Additionally, two different types of microphones are used for the evaluation to find potential advantages of using high SNR MEMS microphones in such devices.
The results revealed that beamforming algorithm definitively improve the performance of voice recognition systems. Especially, the Generalized-Sidelobe-Canceler Beamformer showed a good trade-off between noise source suppression and computational costs. Furthermore, it was concluded that the usage of microphones with low self-noise (high SNR) in voice assistant devices yield better performance in situations where the demanded sound source has a low Sound-Pressure-Level. These benefits vanish if the SPL of the noise source increases.

Keywords: Beamforming Algorithm, Voice Assistant, High SNR Microphone, Signal Processing, GSC, MVDR, Google Assistant, Uniform Circular Microphone Array, Array Processing, Computational Costs, Microcontroller.

Acknowledgement

First of all I would like to thank my main supervisor FH-Prof.in Dipl.-Ing.in Dr.in Ulla Birnbacher for her guidance throughout the education and this thesis.
I would also like to thank my second supervisor Dipl.-Ing. Andreas Gaich from the Johannes Kepler University for his technical guidance in the field of microphone array signal processing.
A big thank you goes to Dr. Dipl.-Ing Siegfried Krainer and Infineon Technologies Austria AG to support my education by giving me the opportunity to gain practical knowledge as a working student besides my theoretical education at the university.
A special thank you goes to all my colleges who I met during my time in the IFAT PMM TI COP-Team at Infineon for their constructive feedback during this work. I want to mention especially Raffael Tschinder, BSc who has accompanied me for many years on my educational and private journey and who was always available for helpful discussions during this thesis.
Last but not least, I want to thank my family and loved ones for their support during my pursuit of my master's degree in engineering.

Contents

1 Introduction to Voice-Assistant-Interfaces

The way humans interact with technology changed a few times over the last century. It started with simple buttons and switches, which were used to control machines. The machine feedback was given by lights and mechanic gauges.
The development of the first computers added punch cards and printers to the well known buttons and switches. This way of interaction with computers changed with the invention of keyboard and display in 1970. Now it was possible to control computers via command-lines and get the feedback via displays. This way of interaction was efficient, but, since the user had to know the exact syntax of the commands, it was nothing for the general public.
This changed with the invention of personal computers in the late seventies of the last century. Now the focus was on making the computer accessible for more people by implementing graphical user interfaces which displaced the command-line interaction. Around 1980 the success story of personal computers continued with the introduction of the mouse and operating systems. Through this, it was even easier to interact with what was displayed on the monitor. The user could now interact with the computer by simply pointing and clicking on windows, icons and menus instead of typing in command-lines. This way of interacting with personal computers was further improved but not really changed until the first decade of the 21st century. [1] [2]
At this time, mobile phones where replaced by smart phones which became more and more powerful. Touchscreens became the new standard of interacting with mobile devices such as mobile phones and tablets. In these devices, the mouse was superseded by the fingers which was a more intuitive way of interacting with a machine. Although this way of interaction is accepted by the generic consumer, the desire of making the interaction with modern computers more intuitive, led to the invention of new interaction possibilities such as gestures and voice.
For human beings, voice is the most natural way of communicating with each other since thousands of years. That's why the desire of interacting with computers by voice was always present in the last 50 years. The performance of modern computers combined with some kind of artificial intelligence made it possible in the last years, to perceive commands per voice. Cloud based voice assistants, such as Amazons Alexa or Goggle Assistant, are conquering our households, helping us with controlling electrical household devices, gathering information from the Internet and setting up all kind of reminders.
The bridge between such cloud based voice assistants and humans are so called voice assistant devices. These interfaces use microphones which capture the spoken commands

and transfer them into the cloud. Unfortunately a microphone captures the spoken words as well as a lot of other disturbing sounds. This makes it hard for voice recognition algorithms to extract the spoken words from the demanded direction. Therefore, many voice assistant devices are using multiple microphones to capture sounds. These so called microphone arrays in combination with beamforming algorithms allow to reduce the unwanted interferer and extract sounds from demanded sources before sending the sound to the voice assistant. This preprocessing improves the number of correctly recognized words even in noisy environments and helps making the interaction between the user and a computer more natural and easier.

The scope of this thesis is the evaluation of a typical signal processing chain for voice assistant devices. Modern array processing algorithms are often computational expensive and hence, not implementable on low cost microcontroller. This thesis focuses on the trade-off between computational costs and the performance of microphone array algorithms. A suitable signal processing chain is implemented on an inexpensive ARM® Cortex®-M4 Microcontroller and evaluated by using normal (65 dBA) and high (69 dBA) SNR MEMS-Microphones in different scenarios.

Figure 1.1: Voice Assistant Device [3]

2 Array Processing Fundamentals

This chapter introduces the fundamentals of digital audio signal processing in terms of beamforming with microphone arrays. First the basics of sound wave generation and propagation are expounded followed by a short introduction about microphones and their application in microphone arrays. In the third part a typical signal processing chain for voice assistant devices is examined with the main focus on Beamformer (BF).

Notation

Within this thesis the following notation is used if not stated otherwise

- Scalar variables are written with normal letters
- Vector/Matrix variables are written with bold letters
- Variables in time-domain are written in lowercase letters
- Variables in frequency-domain are written in uppercase letters
- (t) denotes the continuous time variable
- (k) denotes the frame number (Total number of frames is K)
- $[n]$ denotes the discrete time variable (Total number of time samples per frame k is N)
- (ω) denotes the frequency variable
- m denotes the microphone number (Total number of microphones is M)
- v is the speed of sound (343 $^{m}/_{s}$)

2.1 Sound Generation and Propagation

2.1.1 Sound Generation of Human Speech

Speech sound waves are generated by complex actions of the human body. The voice generation starts in the lungs which are building up air pressure. This pressure brings the vocal folds in the larynx to vibrate. The vibrations produce sounds which are controlled by a set of laryngeal muscles. In that way voiced sounds are produced which humans perceive as speech [4].
Human speech is a complex mixture of a sequence of voiced, unvoiced, plosive and silence signal segments. Most important are the voiced segments which are used by humans to communicate with each other. Voiced segments are constructed by a fundamental frequency ("pitch") and its harmonics. Depending on the gender these pitch

frequencies vary for female voices from 120 to 500Hz and for male voices from 50 to 250Hz. The harmonics have at least two well-defined magnitude maxima, which are defined as formant frequencies. While male voices can have up to three formant frequencies, female voices can even have up to five. The first two maxima determine the vowel. All higher formant frequencies are different for each human and allow people to recognize other humans by there voice [5]. Human speech in general contains frequencies from 100 to 7000Hz. This fact is important to know for audio signal processing because it means that according to the Nyquist–Shannon sampling theorem it is possible to reconstruct a sampled signal if at least a sampling frequency of 14kHz is used. Although studies have shown that doubling the bandwidth from 100-7000Hz to 100-14000Hz improves understandability and user satisfaction often a sampling rate of 16kHz or even lower is used in voice capturing devices such as microphones to save computational power [5].

2.1.2 Sound Propagation

Once the generated speech leaves the human body it propagates as pressure differences, so called air waves, through the environment to the receiver (human ear or capturing device). The planar wave propagation can be described by a simple linear propagation model with the acoustic wave equation [6]:

$$\nabla^2 s(x, y, z, t) = \frac{\partial^2 s(x, y, z, t)}{v^2 \partial t^2} \tag{2.1}$$

where v is the speed of sound ($v = 343\ {}^{m}/_{s}$) and s is the instantaneous sound pressure fluctuation which is dependent on the space variables of the Cartesian coordinate system x, y, z and on the time variable t. For simplification and without loss of generality the acoustic pressure field can be written as $s(\mathbf{r}_0, t)$, where $\mathbf{r}_0 = [x, y, z]^T$ represents the position of the observer (e.g. microphone or human ear) and the sound source is at the origin of the coordinate system as can be seen in the following figure.

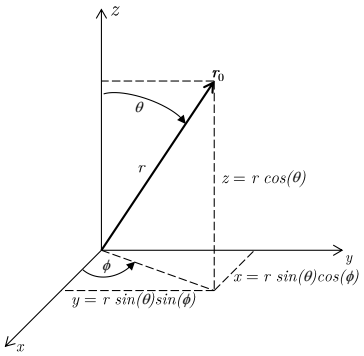


Figure 2.1: Cartesian and spherical coordinate system used within this work

According to [7] one solution of Equation 2.1 is the monochromatic plane wave which can be described by

$$s_0(\mathbf{r}_0, t) = A_0\ e^{i(\omega_0 t - \mathbf{k}_0^T \mathbf{r}_0)} \tag{2.2}$$

where A_0 represents the amplitude of the wave, ω_0 is the temporal frequency and $(\cdot)^T$ denotes the transpose of a matrix or vector. The so called wavenumber vector $\mathbf{k}_0$ is given by

$$\mathbf{k}_0 = \frac{2pi}{\lambda}\mathbf{a}(\theta, \phi) = \frac{\omega_0}{v}\begin{pmatrix} -sin\theta\ cos\phi \\ -sin\theta\ sin\phi \\ -cos\theta \end{pmatrix} \tag{2.3}$$

where λ is the wavelength and $\mathbf{a}(\theta, \phi)$ denotes a unit vector in spherical coordinates which points from the sound source towards the direction of propagation of the monochromatic plane wave. In this context monochromatic means that $s(\mathbf{r}_0, t)$ consists of only one single harmonic with frequency ω_0. In practice an observer captures an infinite number of monochromatic waves from all directions and from different sources [7]. For simplification only monochromatic plane waves are considered within this work.

2.1.2.1 Near-Field Model

The area around a sound source is denoted as near-field. In this area the velocity of the sound pressure and of the sound particle are not in phase. The sound signal captured by an observer within this area without reverberation can be described by

$$X(\omega, \mathbf{r}_m, \mathbf{r}_s) = D_m(\omega, \mathbf{r}_m, \mathbf{r}_s) \cdot S(\omega) + N_m(\omega) \tag{2.4}$$

where $S(\omega)$ is the source signal in frequency domain. N_m is the noise of a microphone with the index m (in case if there are more than just one microphone). The noise is usually modeled as a zero-mean Gaussian random process. In practice this noise consist of two main components: correlated (external) and uncorrelated (from the microphone itself) [5]. $\mathbf{D}_m(\omega, \mathbf{r}_m, \mathbf{r}_s)$ is the sound capturing model for each microphone m which can be written as

$$\mathbf{D}_m(\omega, \mathbf{r}_m, \mathbf{r}_s) = \frac{1}{\|\mathbf{r}_s - \mathbf{r}_m\|} A_m(\omega) U_m(\omega) e^{-i\left(\frac{w}{v}\|\mathbf{r}_s - \mathbf{r}_m\|\right)} \tag{2.5}$$

where $\|\mathbf{r}_s - \mathbf{r}_m\|$ describes the euclidean distance between the source at $\mathbf{r}_s$ and the observer m at $\mathbf{r}_m$. $A_m(\omega)$ is the frequency response of the system preamplifier and the Analog-Digital-Converter (ADC) and $U_m(\omega)$ represents the characteristics of a microphone with the index m. The near-field model is on the one hand geometrically precise and always valid regardless of the distance between source and microphone but requires on the other hand accurate knowledge of the source direction and the distance between source and observer [5] [6].

2.1.2.2 Far-Field Model

With increasing distance between the sound source and the microphones of an array the velocity of sound pressure and the sound particle are in phase. For distances larger than 5-10 times the microphone array diameter the individual distances between each microphone and the sound source can be generalized to one distance from the center of the array to the sound source. This distance can be described by the direction ϕ and the distance ρ [5]. With this generalization the sound capturing model from Equation 2.5 can be transformed to

$$D_m(\omega, \mathbf{r}_m, \mathbf{r}_s) = \frac{1}{\rho} A_m(\omega) U_m(\omega) e^{-i\frac{w}{v}(\|\mathbf{r}_m\| cos(\phi_s - \phi))} \tag{2.6}$$

which decreases the complexity of the model due to the fact that the distance ρ can be assumed constant for every direction. The exponential term represents the phase shifts of the sound wave at each microphone m of the array. These phase differences depend only on the direction of the sound source and the geometry of the array. Within this work only the far-field model is considered due to the fact that sound source voice assistant devices is assumed to be always more than 5-10 times the microphone array size away [5].

2.2 Sound Capturing with Microphone Arrays

The sound capturing in state of the art voice assistant devices is mainly done with Micro-Electro-Mechanical System (MEMS) microphones placed in linear or circular arrays. The scope of this section is to show the different types of microphones and microphones arrays.

2.2.1 Microphones

Microphones are sensors which convert sound waves into electrical signals. Sound waves, which are basically changes in the air pressure, are converted with an flexible membrane (diaphragm) into mechanical movements. These mechanical vibrations can be measured by different measuring principles which classify the microphone. Beside piezoelectric and electrodynamic sensors the most widely used are capacitive microphones [5]. Especially the need for an always smaller form-factor in consumer devices has driven microphone manufacturer to develop small microphones with high performance. Modern MEMS microphones are made of silicon and provide high Signal-To-Noise Ratio (SNR) and good sensitivity while being available in very small packages with low power consumption.

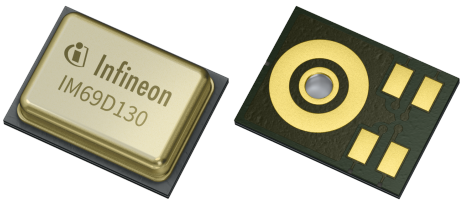


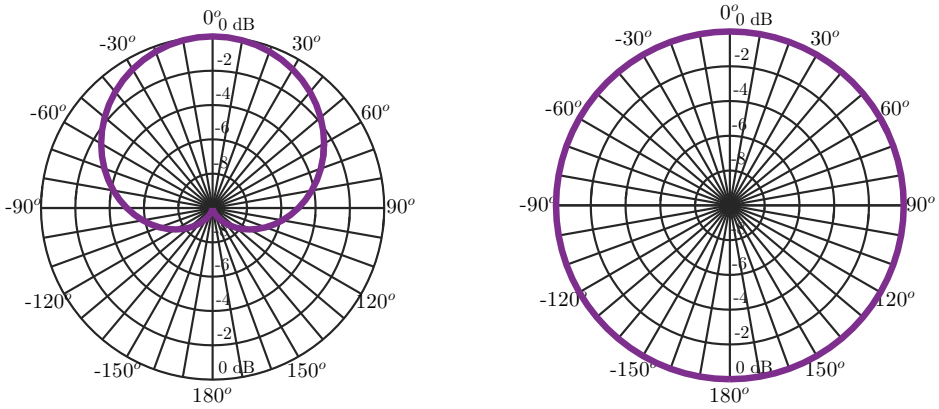
Figure 2.2: Infineon MEMS Microphone IM69D130 [8]

Capacitive microphones measure the relative movement of the diaphragm to a fixed parallel placed membrane. These two membranes form a capacitor which is used to convert the mechanical movements to electrical signals [5]. Modern high performance MEMS microphones such as the one shown in Figure 2.2 are available with analog or digital output and a SNR up to 70 dBA.

2.2.1.1 Types of Microphones

There are two main types of microphones used for array processing, omnidirectional and unidirectional. These two types are distinguished by their directivity pattern/response. The directivity response of a microphone describes how sound waves from

different directions are captured.
Figure 2.3 shows the ideal directivity responses without frequency dependencies as would occur for ideal microphones. Unidirectional microphones are built to pick up sounds waves coming from only one direction. These microphones are often used in headsets and directional microphones to capture only the sound from one desired direction. Sounds from other directions are attenuated as can be seen in the Figure 2.3a. The directivity pattern of this microphone shows a cardioid shape. This microphone attenuates all signals which are not received from ±30°. Real unidirectional microphones have directivity responses which are dependent on frequency and distance between source and microphone.



(a) Unidirectional Microphone **(b)** Omnidirectional Microphone

Figure 2.3: Directivity Pattern

Figure 2.3b shows the directivity pattern of an omnidirectional microphone. Sound waves from all directions are captured evenly without any attenuation.
In voice assistant devices with microphones placed on the top of the device mostly omnidirectional microphones are used since sounds from all directions should be captured. Therefore, only omnidirectional microphones are employed within this thesis.

2.2.2 Microphone Arrays

Besides the source signal an omnidirectional microphone captures also many other signals produced by reverberation, noise or competing sound sources from all directions. Therefore, it is nearly impossible to attenuate these unwanted signal components from the captured signal. Using more than one microphone enables the opportunity to apply specific signal processing algorithms to focus on the sound source and attenuate interferences. A so called microphone array can have different shapes (in 2D or 3D)

with different amounts of microphones. The geometry is usually known and can be generally divided into linear and circular arrays.

2.2.2.1 Uniform Linear Array

Uniform Linear Microphone Arrays (ULAs) typically use 2-8 microphones which are evenly distributed in one line. This type of array is easy to implement because of its small one dimensional form factor and commonly found in computer monitors or headsets.
ULAs can only be used in one half of the xy-plane. This is due to the fact that it is not possible to distinguish between sound waves which are coming from the front or back of the array. This effect is called front-back ambiguity which arises when sound waves are reaching the microphones in the same order and with the same delay as it is depicted in to following figure .

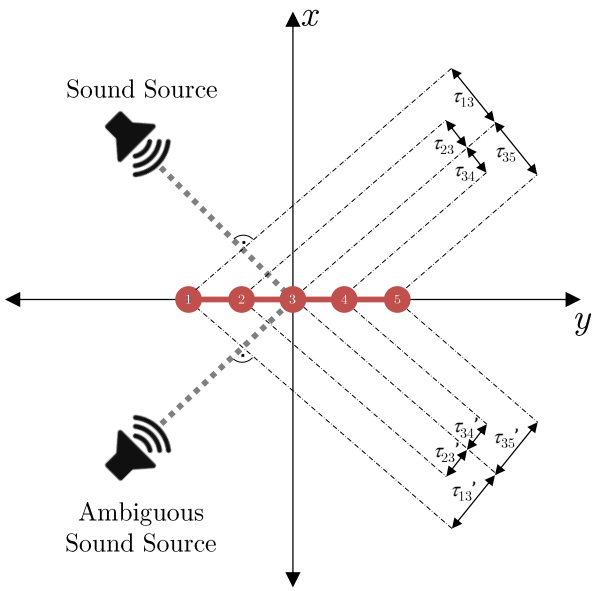


Figure 2.4: Front-back Ambiguity of ULA

Figure 2.4 shows the spatial ambiguity of an ULA. Although the two speakers are on opposite sides of the y-axis the delays are the same for both sides. Sound Source Localization (SSL) algorithms which are examining the position of sound sources by comparing the delays are not able to distinguish between both sides [5]. Further information about SSL can be found in section 3.1. To avoid the ambiguity effect microphones need to be positioned in a two dimensional way for example in an circle.

2.2.2.2 Uniform Circular Array

Uniform Circular Microphone Arrays (UCAs) are typically equipped with 3 - 16 microphones evenly distributed in a circular shape. These kind of arrays can usually be found in conference telephones and voice assistant devices due to the fact that no front-back ambiguity effect arises because of the geometry.

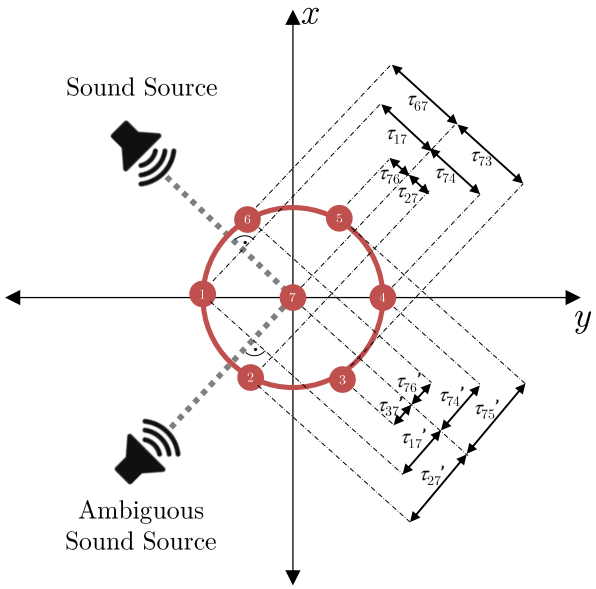


Figure 2.5: Delays caused by two Sound Sources

One can see in Figure 2.5 that in contrast to Figure 2.4 the delays (τ'_{ij}), which are caused by the ambiguous sound source are not the same as the delays (τ_{ij}) caused by the other sound source. The fact that all sound source positions would result in individual delays enables the opportunity to locate sound sources with a SSL 360° in the xy-plane [5].
The delays τ_{ij} between each sensor m_i and the reference sensor m_j are described by the Time-Difference-Of-Arrival (TDOA) [9]

$$\tau_{ij}(\varphi, \theta) = \frac{\varsigma(\mathbf{r}_{m_i} - \mathbf{r}_{m_j})}{v} \tag{2.7}$$

$$= \begin{bmatrix} cos(\theta)cos(\varphi) \\ cos(\theta)sin(\varphi) \\ sin(\theta) \end{bmatrix} \cdot \left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} - \begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} \right) \cdot \frac{1}{v} \tag{2.8}$$

where φ and θ describe the position of the sound source in the spherical coordinate

system and $\mathbf{r_{m_i}}$ and $\mathbf{r_{m_j}}$ denote the position of the microphone in the Cartesian coordinate system.
Within this thesis only UCAs are used due to the fact that the focus of this thesis is about algorithms for voice assistant devices which usually work 360° in the xy-plane.

2.3 Beamforming

Beamforming has a long history in areas such as radar, sonar or communication systems. Basically beamforming can be described as a spatial filter which is applied to the output of a sensor array. Originally, beamforming was developed for narrow-band signals which can be characterized by one single frequency. For audio signal processing these spatial filters had to be adapted to work without disturbing the broadband speech with multiple important frequency components [10].

The goal of beamforming is to find and apply weights $\mathbf{W}$ which change the sound frames in a way, that demanded sound sources are extracted and unwanted components are attenuated without adding audible distortions. This can be achieved by minimizing the so called noise gain $\mathbf{G_N}$ with the constraint of unity gain and zero phase shift towards the direction of the sound source. In other words the goal is to capture the sound waves from the desired sound source with the highest possible SNR [5].
Mathematically the design goal can be described by:

$$\mathbf{W} = arg\min_{\mathbf{W}}(\mathbf{G_N}) \tag{2.9}$$

with the constraint

$$\mathbf{B}(\omega, \phi, \theta) \equiv 1 \tag{2.10}$$

where $\mathbf{W}$ is the weight matrix, $\mathbf{G_N}$ is the noise gain vector. $\mathbf{B} = \mathbf{W}(\omega)\mathbf{D}(\omega, \mathbf{r_m}, \mathbf{r_s})$ is the directivity pattern where $\mathbf{W}$ is the filter coefficient and $\mathbf{D}$ is the steering vector. $\mathbf{B}$ defines the relationship between the actual microphone array output and an ideal observer in the center of the array.

There are two types of Beamformers (BFs): data-independent and data-dependent. The difference between those two is that, in contrast to data-dependent, which can also be denoted as adaptive BF, data-independent BF are not relative to the audio frame.

2.3.1 Time-Invariant Beamformers

The weights of time-invariant BF can be calculated for each listening direction in advance depending on the geometry and noise assumptions. This means, that the weights of the BF cannot be changed with the listening environment. The BF relies on the assumption that only isotropic noise (same noise spectrum everywhere) and spherical noise (uncorrelated noise sources from all directions) are present. In practice the precalculated filter weights for a defined set of directions are saved in memory. This enables a very efficient real-time implementation as a changing direction results in switching the memory address only. The simplest implementation of such an time-invariant BF is the Delay-And-Sum Beamformer (D&S-BF) [5].

2.3.1.1 Delay-And-Sum Beamformer

The D&S-BF can be divided into two processing steps. In the first step the captured data frames of all microphones are individually shifted in time according to the arrays geometry. The goal is to align all frames to a reference point (reference microphone) in a way that the TDOA-Differences described by Equation 2.7 vanish. After that step the individual frames can be summed up and scaled by the number of microphones. Figure 2.6 shows the D&S-BF principle.

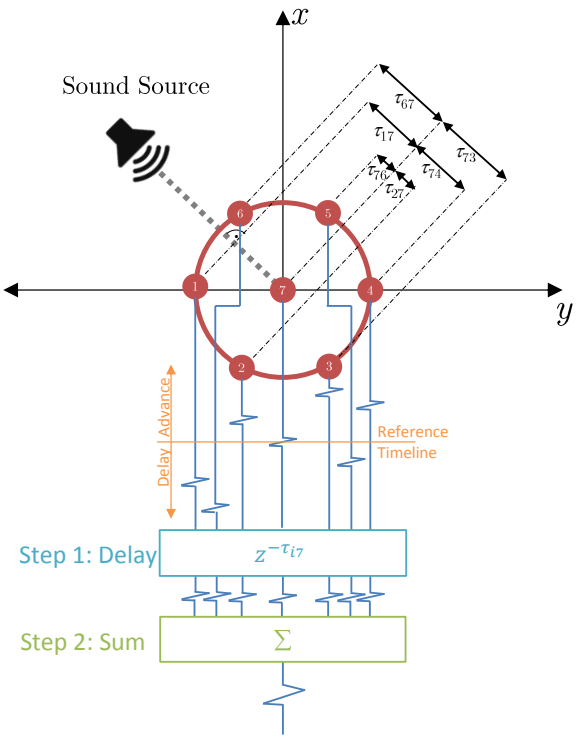


Figure 2.6: Delay-and-Sum Beamformer - Principle

With this technique it is possible to extract sound waves which are coming from the listening direction and attenuate noise from other directions [10].

The D&S-BF can either be implemented in time or frequency domain.

Time-Domain Implementation

Usually the D&S-BF is implemented in time-domain due to the fact that no transformation in frequency domain is needed. The first step can be easily done by delaying or advancing the input frame.

$$y_m[n] = x_m[n - \tau_{i7}] \tag{2.11}$$

where y_m is the aligned frame, x_m the captured frame for each microphone m and τ_{i7} is the delay between microphone i and the center microphone 7.
To get the output frame all aligned microphone frames are summed up and scaled by the number of microphones M.

$$y[n] = \frac{1}{M} \sum_{m=1}^{M} y_m[n] \tag{2.12}$$

Frequency-Domain Implementation

The shifting can also be done in frequency-domain which is usually the case if the pre- or post-processing of the sound frame is also done in the frequency domain. The D&S-BF in frequency domain can be written as

$$Y(\omega) = \frac{1}{M} \sum_{m=1}^{M} X_m(\omega) e^{-i\frac{\omega}{v}(\|\mathbf{r}_m\| cos(\phi_m - \phi))} \tag{2.13}$$

where Y is the output frame and X_m the captured frame for each microphone m in frequency domain. $\|\mathbf{r}_m\|$ is the euclidean distance between the center of the microphone array and each microphone m. ϕ_m describes the angle between the y-axis and the microphone and ϕ is the angle between the y-axis and the incoming sound source wave.

The D&S-BF is an efficient BF in case of uncorrelated noise across all microphones. If this noise is modeled as a zero-mean Gaussian process, one can say that the uncorrelated noise is decreasing by $\sqrt{M}$. This results in an increase of 3 dB per microphone pair. However, in real environments with reverberation and noise sources which are captured by all microphones and are therefore correlated to each other, the performance of the D&S-BF decreases significantly [5].

The fact that in voice assistant devices the competitive noise sources as well as reverberation corrupts the captured signal heavily makes the D&S-BF unattractive for the demanded application.
Therefore, another approach - named Filter-And-Sum BF - was introduced. These BFs filter each microphone channel individually followed by summing up to receive a single output stream.

$$Y(\omega) = \sum_{m=1}^{M} W_m(\omega) X_m(\omega) \tag{2.14}$$

The difference between the D&S-BF and Filter-And-Sum BFs is that the weights W_m have not the same magnitude and linear phase which results from the constant time delays. Consequently, Filter-And-Sum Beamformer have more degrees of freedom and hence allow better coefficient designs for microphone arrays [5]. The following algorithms can all be considered as Filter-And-Sum BF.

2.3.1.2 Minimum-Variance-Distortionless-Response (MVDR) Beamformer

The Minimum-Variance-Distortionless-Response Beamformer (MVDR-BF) was originally designed for antenna arrays and can be denoted as one of the first Filter-And-Sum BF. For the use in microphone arrays the original BF has to be changed to work with broadband sound signals.
The following mathematical derivation is taken from [5] and [6]. According to them the input vector of a MVDR-BF can be described as:

$$\mathbf{X^{(k)}}(\omega) = \mathbf{D_c^{(k)}}(\omega) S_{\mathbf{c}}^{(k)}(\omega) + \mathbf{N^{(k)}}(\omega) \tag{2.15}$$

The input vector has a size of $M \times 1$ and is composed by the capturing vector D_c depending on the sound source position $\mathbf{c} = [x, y, z]$. Without loss of generality the frame index $(\cdot)^{(k)}$ can be omitted.
The output of the MVDR-BF can be written as

$$Y(\omega) = \mathbf{W_c}(\omega)\mathbf{X}(\omega) \tag{2.16}$$

where $\mathbf{W}_c$ is the weight vector with length $1 \times M$. Under no-noise conditions

$$\mathbf{W_c}(\omega)\mathbf{D_c}(\omega) = 1 \tag{2.17}$$

would lead to a undistorted sound capture

$$Y(\omega) = S(\omega) \tag{2.18}$$

With noise the goal is to minimize the noise variance in the BF output

$$Y(\omega) = \mathbf{W_c}(\omega)\mathbf{D_c}(\omega)S(\omega) + \mathbf{W_c}(\omega)\mathbf{N} = S(\omega) + \mathbf{W_c}(\omega)\mathbf{N}(\omega) = S(\omega) + Y_n(\omega) \tag{2.19}$$

with the constrained optimization criterion $\mathbf{Q}_{constr}$:

$$\mathbf{Q}_{constr} = E\left\{|Y_n|^2\right\} = \mathbf{W}_c(\omega)\mathbf{\Phi}_{NN}(\omega)\mathbf{W}_c^H(\omega) \tag{2.20}$$

where $E\{\cdot\}$ is the expectation operator, $(\cdot)^H$ denotes the Hermitian transpose which is the conjugate transpose of a matrix and $\mathbf{\Phi}_{NN} = E\left\{N(\omega)N(\omega)^H\right\}$ is the noise cross-power spectral matrix which sometimes is also termed as array covariance matrix. This leads to the constrained minimization problem

$$\mathbf{W_c}(\omega) = arg\min_{\mathbf{W_c}} \mathbf{W_c}(\omega)\mathbf{\Phi}_{NN}\mathbf{W_c}^H(\omega) \tag{2.21}$$

with the constrain that

$$\mathbf{W_c}(\omega)\mathbf{D_c}(\omega) = 1 \tag{2.22}$$

which let the MVDR-BF minimize its output signal power under the constraint that signals from the listening direction are maintained. This problem can be analytically solved by using the Lagrange multipliers λ:

$$\mathbf{Q} = \mathbf{W_c}(\omega)\mathbf{\Phi}_{NN}\mathbf{W_c}^H(\omega) + \lambda(\omega)[\mathbf{W_c}(\omega)\mathbf{D_c}(\omega) - 1] + \lambda^*(\omega)[\mathbf{W_c}^H(\omega)\mathbf{D_c}^H(\omega) - 1] \tag{2.23}$$

where $(\cdot)^*$ denotes the complex conjugate. The optimal weights can be obtained by taking the complex gradient of $\mathbf{Q}$ with respect to $\mathbf{W_c}$:

$$\mathbf{W_c}_{opt}(\omega) = -\lambda(\omega)\mathbf{D_c}^H(\omega)\mathbf{\Phi}_{NN}^{-1}(\omega) \tag{2.24}$$

If Equation 2.24 is extended by multiplying $\mathbf{D_c}$ to both right sides, λ can be found with the help of 2.17 to be:

$$\lambda(\omega) = -\frac{1}{\mathbf{D_c}^H(\omega)\mathbf{\Phi}_{NN}^{-1}(\omega)\mathbf{D_c}(\omega)} \tag{2.25}$$

Hence the constrained optimal weights for the MVDR-BF can be found with

$$\mathbf{W}_{MVDR}(\omega) = \frac{\mathbf{D_c}^H(\omega)\mathbf{\Phi}_{NN}^{-1}(\omega)}{\mathbf{D_c}^H(\omega)\mathbf{\Phi}_{NN}^{-1}(\omega)\mathbf{D_c}(\omega)} \tag{2.26}$$

For the time-invariant MVDR-BF the noise cross-power spectral matrix $\mathbf{\Phi}_{NN}$ can only be assumed since the real noise of a room is not constant. Therefore, the noise will be modeled to be spatially homogeneous (meaning the correlation function is not dependent on position of the observer) and spherically isotropic. Hence uncorrelated noise sources are assumed to be perceived from all directions [5]. With this assumptions the noise cross-power spectral matrix can be described by

$$\mathbf{\Phi}_{NN}(\omega) = \mathbf{N}(\omega)\mathbf{N}(\omega)^H = \begin{pmatrix} \mathbf{\Phi}_{11}(\omega) & \mathbf{\Phi}_{12}(\omega) & \cdots & \mathbf{\Phi}_{1M}(\omega) \\ \mathbf{\Phi}_{21}(\omega) & \mathbf{\Phi}_{22}(\omega) & \cdots & \mathbf{\Phi}_{2M}(\omega) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{\Phi}_{M1}(\omega) & \mathbf{\Phi}_{M2}(\omega) & \cdots & \mathbf{\Phi}_{MM}(\omega) \end{pmatrix} \tag{2.27}$$

with

$$\mathbf{\Phi}_{ij}(\omega) = N_0(\omega) sinc\left(\frac{\omega d_{ij}}{v}\right) \tag{2.28}$$

where N_0 is the ambient noise spectrum captured by the omnidirectional microphone and d_{ji} is the distance between sensor i and j [5].
In the ideal case this would be enough to find the optimal coefficients for the MVDR-BF. For a more realistic design also the microphone's self-noise should be considered since Equation 2.24 would lead to high instrumental noise gain.
To overcome this problem, either the self-noise of each microphone can be included when calculating the microphone covariance matrix or an adaptive approach for the MVDR-BF can be considered

2.3.2 Time-dependent or adaptive Beamformers

Time-invariant BFs have the disadvantage that noise of a room can only be modeled in advance. Therefore, the BF cannot adjust its weights to a changing environment with competing sound sources and reverberation. In contrast, adaptive BF are calculating the filter coefficients in dependence of the current input signal and in respect to the microphone array's geometry. This yields the advantages to achieve better noise suppression under changing conditions and consequently a lower attenuation of the desired signal from the listening direction. The disadvantage of such BFs is that, in contrast to

time-invariant BF, adaptive BFs are computationally more expensive and need more memory space to adjust the weights of the spatial filter [5].

2.3.2.1 Adaptive Minimum-Variance-Distortionless-Response Beamformers

The adaptive MVDR-BF is based on the time-invariant MVDR-BF which is introduced in chapter 2.3.1.2. Hence it uses the constrained optimal weights of the MVDR-BF (Equation 2.24) but with the difference that the noise cross-power spectral matrix is recalculated if no sound source signal from the listening direction is present. This enables the opportunity to adjust the MVDR-BF weights on a changing noise field to attenuate interfering signals and changing sound source directions to emphasize the demanded signal from the listening direction [5] [6]. In real-time implementations the covariance matrix has to be calculated within one noise frame. A covariance matrix estimator can be used to estimate the noise cross-power spectral matrix $\mathbf{R}_{NN}$

$$\mathbf{R}_{NN}(\omega) = \frac{1}{K}\sum_{k=0}^{K-1} x_k x_k^H \tag{2.29}$$

where $x_k x_k^H$ results in a $N \times N$ matrix for each noise input frame k. K is the number of frames with N samples per frame. To prevent the covariance matrix from changing to fast a so called forgetting factor $0 < \lambda_{MVDR} < 1$ (typically ≈ 0.8) is introduced and the estimator is changed in the following way:

$$\mathbf{R}_{NN}(\omega) = \lambda_{MVDR}\mathbf{R}_{NN} + (1 - \lambda_{MVDR})x_k(\omega)x_k^H(\omega) \tag{2.30}$$

which is calculated per frequency bin for each noise frame k. According to Equation 2.26, $\mathbf{R}_{NN}$ has to be inverted which is prone to ill-conditioned matrices. $\mathbf{R}_{NN}$ can get ill-conditioned for low frequencies. Therefore, diagonal loading (regularization) is mandatory before inverting the cross-power spectral matrix to get an robust BF [11] which can be done by:

$$\tilde{\mathbf{R}}_{NN} = \mathbf{R}_{NN} + \varepsilon\mathbf{I} \tag{2.31}$$

where ε is the diagonal loading factor ($\approx 10^{-2}$ *to* 10^{-4}) and $\mathbf{I}$ denotes the $N \times N$ identity matrix. If Equation 2.31 and Equation 2.26 are combined, the new weights can be calculated for the robust adaptive MVDR-BF with

$$\mathbf{W}_{MVDR}(\omega) = \frac{\mathbf{D_c}^H(\omega)(\mathbf{R}_{NN} + \varepsilon\mathbf{I})^{-1}(\omega)}{\mathbf{D_c}^H(\omega)(\mathbf{R}_{NN} + \varepsilon\mathbf{I})^{-1}(\omega)\mathbf{D_c}(\omega)} \tag{2.32}$$

The MVDR-BF still has the problem that the cross-power spectral matrices have to be inverted which results in a computational complexity of $O(N^3)$ [11] which is very extensive especially for real-time application. Therefore, an alternative is to compute the MVDR-BF weights iteratively which is basically an improved Frost Beamformer and is described in [11].

2.3.2.2 Generalized-Sidelobe-Canceler (GSC) Beamformer

A different approach to overcome the computational expensive MVDR weight computation is to use a Generalized-Sidelobe-Canceler Beamformer (GSC-BF)-Structure as proposed by Griffiths and Jim in [12].

Griffiths and Jim Time-Domain GSC Beamformer

The proposed GSC-BF divides the constrained minimization problem of the MVDR-BF into a unconstrained one which can be computed more efficiently [4].

Griffiths and Jim's time-domain GSC-BF architecture consists of a conventional time-invariant BF, a blocking matrix **b** and a Adaptive-Interference-Canceler (AIC) as can be seen in the Figure 2.7.

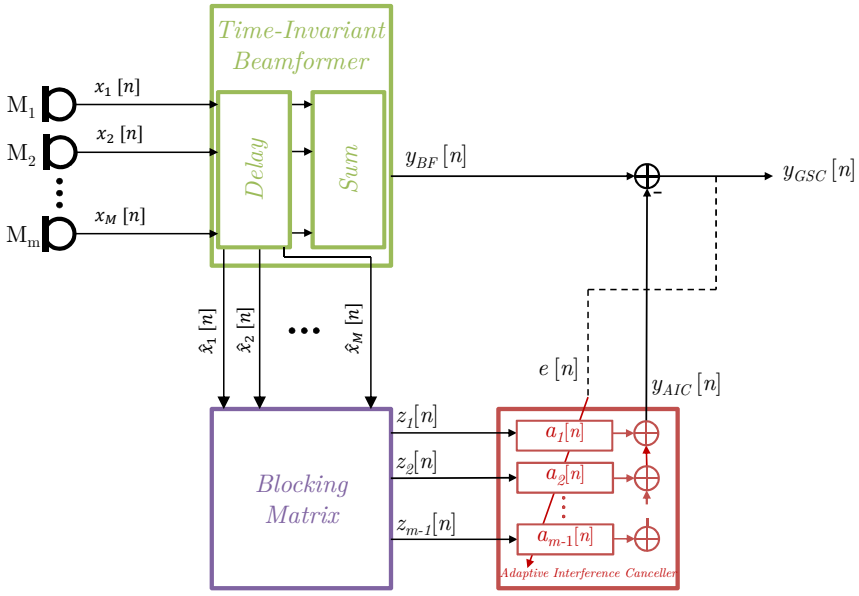


Figure 2.7: Blockdiagram of Griffiths-Jim Generalized Sidelobe Canceler Implementation

The time-invariant BF is typically a D&S-BF. As described in section 2.3.1.1, the D&S-BF aligns all M input streams according to Equation 2.13. The aligned input stream

vector $\hat{\mathbf{x}} = [\hat{x}_1[n], \hat{x}_2[n], \ldots, \hat{x}_M[n]]^T$ is subsequently multiplied with a blocking matrix $\mathbf{b}$ to remove all demanded components from the input frame

$$\mathbf{z}[n] = \mathbf{b}\hat{\mathbf{x}}[\mathbf{n}] \tag{2.33}$$

Griffiths and Jim proposed the following $(M-1) \times M$ - blocking matrix

$$\mathbf{b} = \begin{bmatrix} 1 & -1 & 0 & \ldots & 0 \\ 0 & 1 & -1 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \tag{2.34}$$

which subtracts each two adjacent input streams. This results in a cancellation of the sound components from the steered direction. Consequently the $M-1$ output streams $\mathbf{z}[n] = [z_1, z_2, \ldots, z_{M-1}]^T$ are holding only the remaining noise components which are feed into the AIC [13].
In the last stage of the GSC-BF the AIC adaptively subtracts the remaining interferences from the D&S-BF output y_{BF}

$$y_{GSC}[n] = y_{BF}[n] - y_{AIC}[n] = e[n] \tag{2.35}$$

The AIC consists of $M-1$ LMS adaptive filters. The input of the AIC $\mathbf{z}[n]$ is multiplied with the filter weights $\mathbf{w}_{AIC}[n] = [w_{AIC,1}, w_{AIC,2}, \ldots, w_{AIC,M-1}]^T$ of the adaptive filter and summed up to form y_{AIC}.

$$y_{AIC}[n] = \mathbf{z}[n]^T \mathbf{w}_{AIC}[n] \tag{2.36}$$

The Least-Mean-Squares (LMS) algorithm uses y_{GSC} to update the filter weights for the following input sample in a way that the Mean-Squared-Error (MSE) of the GSC-BF output is minimized.

$$\mathbf{w}_{AIC}[n+1] = \mathbf{w}_{AIC}[n] + \alpha \; y_{GSC}[n] \; \mathbf{z}[n] \tag{2.37}$$

The step size parameter α is constant and influences the convergence as well as the sound quality. According to [14], α must be in the range of $0 < \alpha < \dfrac{2}{total\ input\ power}$ to ensure convergence. Small step sizes lead to slow convergence and slow tracking which would result in a "breathing" noise in the output. Large α on the other hand would lead to misadjustments in the output which emerges in a cancellation of the desired signal components [15]. Therefore, the step size must be chosen dependent on

the application and environment.
The LMS algorithm has the downside that the convergence depends, beside the step size parameter also strongly on the signal power of the input sample $z_m[n]$. To prevent large filter weights, which would lead to instability of the adaptive filter, a Normalized-Least-Mean-Squares (NLMS) algorithm can be used to compute the filter coefficients [13]. The NLMS uses the variable step size μ [14]

$$\mu = \frac{\alpha}{\|\mathbf{z}[n]\|^2 + \gamma} \tag{2.38}$$

where $\|\mathbf{z}[n]\|^2 = \mathbf{z}[n]^T\mathbf{z}[n]$ is the Euclidean norm, which can also be denoted as total input power [4]. γ is a small positive constant ($\approx 10^{-3} - 10^{-6}$) to avoid large step sizes when the input power of the current sample is very small [14]. The filter weight update equation for a NLMS adaptive filter can be expressed as

$$\mathbf{w}_{AIC}[n+1] = \mathbf{w}_{AIC}[n] + \mu\ y_{GSC}[n]\ \mathbf{z}[n] \tag{2.39}$$

$$= \mathbf{w}_{AIC}[n] + \frac{\alpha\ y_{GSC}[n]\ \mathbf{z}[n]}{\|\mathbf{z}[n]\|^2 + \gamma} \tag{2.40}$$

The Griffiths and Jim GSC-BF with a Blocking Matrix is a computational efficient BF but has the downside that the performance relies on the accuracy of the SSL (see section 3.1). If the Direction-Of-Arrival (DOA) is not detected accurately, signal components from the listening direction are leaking into the input of the AIC. Consequently these components are also canceled from y_{BF} and will not be present in the output of the GSC-BF. This would lead to a degradation of the sound quality of the BF output.
To address this issue, the Blocking Matrix can be replaced by several adaptive filters which is then denoted as Adaptive-Blocking-Matrix (ABM). This filter tries to subtract all demanded signal components from the microphone inputs. In a way that only the interfering components remain in the signals which are feed into the AIC [6]. This type of GSC-BF with an ABM is alternatively called Robust-Generalized-Sidelobe-Canceler (RGSC).

Robust Time-Domain GSC Beamformer

The RGSC is derived from Griffiths and Jim's GSC-BF, but with the difference that the blocking matrix is replaced by M - Coefficient-Constrained-Adaptive-Filters (CCAFs) [15], as depicted in Figure 2.8.

The CCAFs are using the enhanced signal from the time-invariant BF to minimize

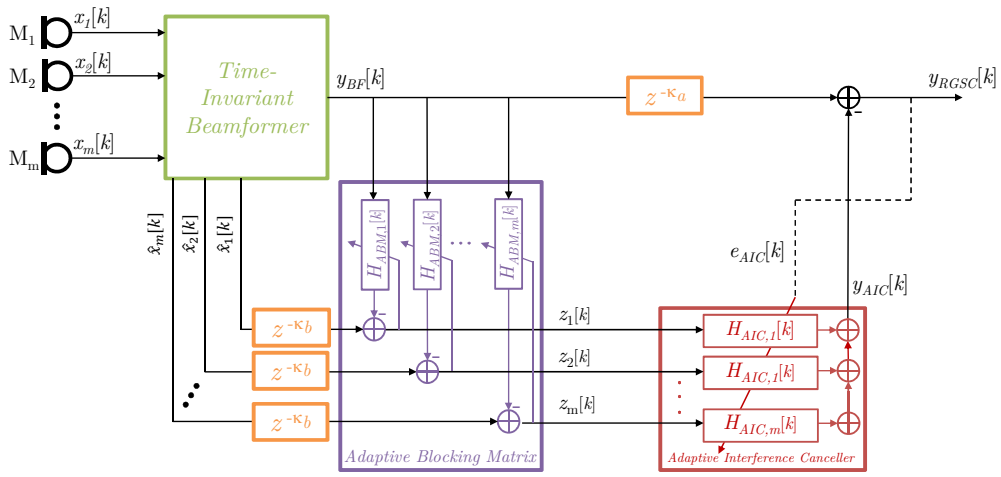


Figure 2.8: Blockdiagram of a Time-Domain Implementation of the Robust GSC with a Coefficient Constrained Adaptive Filter (CCAF) - Adaptive Blocking Matrix

the target signal from the steering direction in the blocking matrix outputs $\mathbf{z}[n]$. The coefficients of each CCAF are first updated in the same NLMS way as described in Equation 2.39 and then constrained in the following way

$$w_{ABM,m}[n+1] = \begin{cases} \phi_m & for\ w_{ABM,m}[n+1] > \phi_m \\ -\phi_m & for\ w_{ABM,m}[n+1] < -\phi_m \\ w_{ABM,m}[n+1] & otherwise \end{cases} \tag{2.41}$$

The constrained coefficients allow to have a DOA error up to a specific degree. The constraint ϕ_m depends on the following equation

$$\phi_m = \frac{1}{\pi \max\left(0.1,\ (M_{ABM} - \kappa_b) - T_m,\ -(M_{ABM} - \kappa_b) - T_m\right)} \tag{2.42}$$

where κ_b is the number of delay samples which is needed to ensure causality of the input stream and M_{ABM} is the filter order of the CCAF . T_m is the group delay in samples which is dependent on the geometry of the array and the maximum allowable DOA error in degree [15]

$$T_m = \frac{r_m f_s}{v} \sin(\Delta\varphi) \tag{2.43}$$

where r_m is the distance between the center and each microphone, f_s is the used sampling frequency and $\Delta\varphi$ is the maximum allowable steering vector error.

The described BF can either be realized in time or frequency domain. For real-time processing it is advisable to realize the RGSC in frequency domain with Frequency-

Domain Adaptive Filters (FDAF) which are computational more efficient but consumes more memory [16].

Robust Frequency-Domain GSC Beamformer

Figure 2.9 shows the previously described time-domain RGSC converted to work in the frequency domain. Within this implementation, the overlap-and-save method should be used for the block processing. This method avoids acoustic artifacts due to circular convolution effects in the reconstructed time-domain output signal. The Overlap-and-save block processing technique overlaps two consecutive frames by a factor α and applies the signal processing algorithm on the overlapped frames. After transforming the processed frame back into time domain the first N samples are discarded to get the N output samples [16].

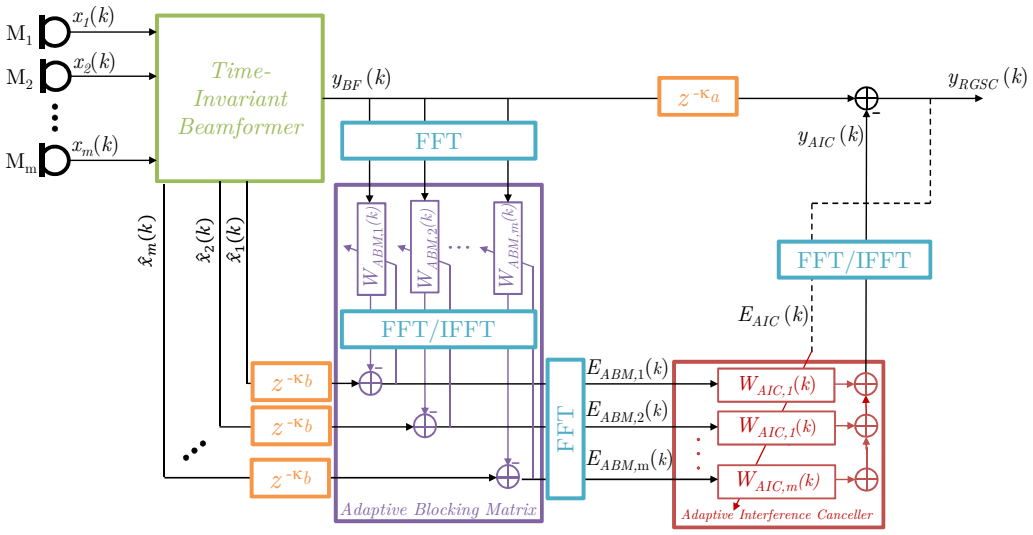


Figure 2.9: Blockdiagram of a Frequency-Domain Implementation of the Robust GSC

Similar to the Griffiths-Jim GSC-BF the first stage of the RGSC is a Fixed Beamformer (FBF) (green box in Figure 2.9) which aligns all input frames in respect to a reference point/microphone. The output is gathered by summing up all aligned frames and scaling by the number of microphones as described in Equation 2.13. This BF can be realized either in frequency- or time-domain [16].
The output of the time-invariant BF has a length of N samples per frame k. These N samples are overlapped with the previous N samples, which results in a $2N \times 1$ input frame y_{BF} for the ABM.

The purple box in Figure 2.9 depicts the Adaptive-Blocking-Matrix (ABM). It consist of M - Frequency-Domain Adaptive Filters (FDAF) which adaptively subtract the

demanded signal component from the listening direction from each microphone input. A time delay κ_b delays each microphone input to ensure the causality of the adaptive filters.
For optimization purposes the overlapped beamforming output y_{BF} is transformed into frequency domain and saved in an diagonal matrix $\mathbf{Y}_{BF}$.
The output signal of the FDAF $\mathbf{Y}_{ABM,m}$ can be calculated by

$$\mathbf{Y}_{ABM,m}(k) = \mathbf{Y}_{BF}(k)\ \mathbf{W}_{ABM,m}(k) \tag{2.44}$$

where $\mathbf{W}_{ABM,m}$ are the adaptive filter coefficient for each microphone m. Subsequently $\mathbf{Y}_{ABM,m}$ is transformed back into time domain and multiplied with $\mathbf{h}_{01}$, which is a $2N \times 2N$ diagonal matrix where the first N values of the diagonal are zeros and the other N values are ones. The error vector $\mathbf{e}_{ABM,m}$ is then subtracted from the reference signal $\hat{\mathbf{x}}_m(k - \kappa_b)$ (aligned input frame) which results in the error vectors $\mathbf{e}_{ABM,m}$

$$\mathbf{e}_{ABM,m}(k) = \hat{\mathbf{x}}_m(k - \kappa_b) - \mathbf{y}_{ABM}(k); \tag{2.45}$$

These vectors are on the one hand fed into the AIC and on the other hand used to update the weights of the adaptive filters from the ABM using

$$\mathbf{W}_{ABM,m}(k+1) = \mathbf{W}_{ABM,m}(k) + \boldsymbol{\mu}(k)\mathbf{Y}^H_{BF}(k)\mathbf{E}_{ABM,m}(k) \tag{2.46}$$

where $\mathbf{E}_{ABM,m}$ are the error vectors in frequency domain and $\boldsymbol{\mu}$ is the normalized step size defined as

$$\boldsymbol{\mu}(k) = 2\mu\ diag([P_0^{-1}(k),\ \ldots\ , P_{2N-1}^{-1}(k)]) \tag{2.47}$$

where μ is the fixed step size parameter and P_l being the power estimate of the l^{th} frequency bin which can be calculated by

$$P_l(k) = \lambda_{GSC} P_l(k-1) + (1 - \lambda_{GSC})\,|Y_{BF,l}(k)|^2 \tag{2.48}$$

where λ_{GSC} is the forgetting factor and $Y_{BF,l}$ denotes the l^{th} frequency bin of $\mathbf{Y}_{BF}$ [16].

The Adaptive-Interference-Canceler (AIC) adaptively subtracts interfering signal components from the FBF output which leads to the beamformed GSC-BF output. The time-delay k_a is introduced to ensure causality of the adaptive filters.
The input of the AIC is formed by the current and the previous outputs of the ABM

in the following way:

$$\mathbf{X}_{AIC,m}(k) = diag(\mathbf{E}_{ABM,m}(k) + \mathbf{J}\ \mathbf{E}_{ABM,m}(k-1)) \tag{2.49}$$

where $\mathbf{J} = diag([1,-1,1,-1]_{1\times 2N})$ is multiplied with frequency domain representation of the previous ABM output $\mathbf{E}_{ABM,m}(k-1)$ which results in an circular shift of N samples in the frequency domain. The resulting input vectors are multiplied with the adaptive filter coefficients and summed up to obtain the output of the AIC.

$$\mathbf{Y}_{AIC}(k) = \sum_{m=1}^{M} \mathbf{X}_{AIC,m}(k)\mathbf{W}_{AIC,m}(k) \tag{2.50}$$

where $\mathbf{W}_{AIC,m}$ are the adaptive filter weights for each microphone m. The resulting AIC output vector is transformed back into the time domain and multiplied with $\mathbf{h}_{01}$. The resulting error of the AIC can be gathered by

$$\mathbf{e}_{AIC}(k) = \mathbf{y}_{BF}(k) - \mathbf{y}_{AIC}(k) \tag{2.51}$$

which is used to update the filter weights of the AIC as defined by:

$$\mathbf{W}_{AIC,m}(k+1) = \mathbf{W}_{AIC,m}(k) + \boldsymbol{\mu}(k)\mathbf{X}^{H}_{AIC,m}(k)\mathbf{E}_{AIC}(k) \tag{2.52}$$

where the variable step size μ is defined by 2.47 with the estimated power for each frequency bin l

$$P_l(k) = \lambda_{GSC}P_l(k-1) + (1-\lambda_{GSC})\sum_{m=1}^{M}|X_{AIC,l}(k)|^2 \tag{2.53}$$

Consequently the output of the GSC-BF-BF is obtained by taking the last K samples of $\mathbf{e}_{AIC}$ [16].

3 Implementation of a Microcontroller-Powered Signal Processing Chain

This section is dedicated to the implementation of a typical microphone array signal processing chain for microcontroller powered voice assistant devices. It first introduces a typical signal processing chain followed by an introduction of the developed Cerebro-MultiMicBoard which is used for evaluating the suitable beamforming algorithms. The third part of this chapter comprises the analysis of the computational costs of the implemented algorithms with the focus on needed processing power and memory usage.

3.1 Voice Assistant Signal Processing Chain

Modern voice assistant devices usually process the captured audio data in a specific manner. Figure 3.1 shows a typical signal processing chain with the microphones (M_m), buffer (red box), Sound Source Localization (SSL) (orange box), Voice Activity Detector (VAD) (purple box), BF (green box) and an interface to the cloud based voice assistant (blue box).

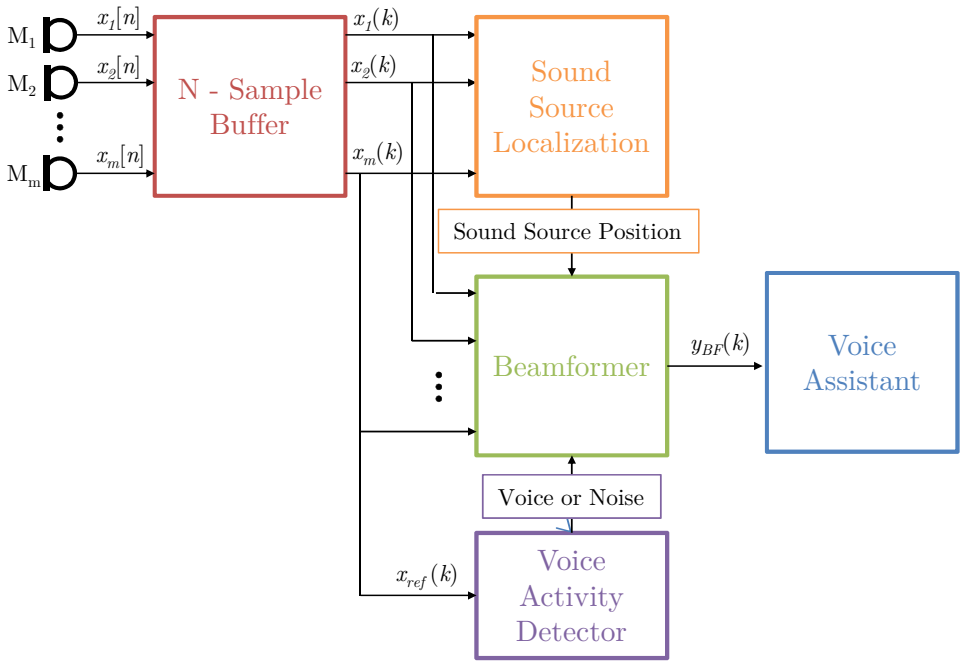


Figure 3.1: Signal Processing Chain

N-Sample Buffer

Audio signal processing is primarily done with so called audio frames which are composed by N successively captured audio samples. These audio frames are usually 10-100 ms long and are sampled with a frequency of 8-192 kHz. Within this thesis audio frames of a length 16 ms are captured with 16 kHz, which results in 256 values per frame. These frames contain the desired signal from the sound source corrupted by noise, reverberation and competing sound sources. To collect these frames of N samples a buffer of length $M \times N$ is used where M is the number of microhones. K denotes the number of frames with k as the current processed frame.
If not mentioned otherwise all BFs discussed in this work process audio frames in the frequency domain with a method called Overlap-and-Add (OLA). This procedure ensures that all processed frames merge together with the succeeding frame. Otherwise discontinuities will arise which can be heard. Instead of processing frame by frame always $\alpha \cdot N$ samples from the previous frame are merged with the current frame. α is called the overlapping factor and is typically ≈ 0.5. Therefore, always a part of the previous frame and the current frame is processed together. The extracted frame must be windowed (typically with a Root-Hanning window) before transforming into the frequency domain to avoid spectral effects. After processing the result is transformed back into time-domain and windowed again. The windowed result is overlapped by $\alpha \cdot N$ samples with the previous frames. The overlapping samples from the previous and the current processing step are added up which prevents acoustic artifacts [5].

Voice Activity Detector (VAD)

A Voice Activity Detector (VAD) is used within voice assistant devices to label if an incoming frame k is either noise or voice. A frame is only used for further processing if it is labeled as voice. Otherwise it is either discarded or used for noise floor calculations which might be used by the beamformer.
There are several VAD approaches available: Starting from computationally efficient energy based algorithms, up to extensive machine learning algorithms which detect only specific keywords such as "Alexa", "Hey Google" or "Hey Siri". Within this thesis a simple energy based VAD with internal logic is implemented where only data from one reference microphone (x_{ref}) is used. As the focus of this thesis is on beamforming, the VAD is not described more precisely but can be looked up in [17].

Sound Source Localization (SSL)

The performance of BFs is highly dependent on steering the beam into the correct source direction. If the beam would be steered into the wrong direction, a BF would

attenuate the signal components from the desired sound source. Therefore, it is mandatory to localize the demanded sound source accurately to ensure that the BF can work properly.
Voice assistant devices can estimate the DOA of demanded sources with so called Sound Source Localization (SSL) algorithms. There are several algorithms available. In this thesis a GCC-PHAT algorithm is implemented to localize the position of the speaker with a resolution of 30°.
The GCC-PHAT is based on TDOA defined in Equation 2.7. The time shift between each microphone and a reference microphone (x_{ref}) is calculated with the Generalized Cross-Correlation (GCC) [9]:

$$\mathbf{r}_{ij}(k) = iFFT(\mathbf{\Psi} \cdot \mathbf{X}_i(\omega)\mathbf{X}_j(\omega)) \tag{3.1}$$

where $\mathbf{X}_i(\omega)\mathbf{X}_j(\omega) = \mathbf{G}_{X_iX_j}$ is the cross correlation between the two frequency domain microphone frames, iFFT is the inverse Fourier transformation which converts the result from frequency domain to time domain and $\mathbf{\Psi}$ is the generalizing weighting function. $() \cdot ()$ denotes the per-element matrix multiplication. The weighting function is needed because the peak width of the cross-correlation is frequency dependent. This would result in a flat and smeared peak for low-frequency components [5]. Therefore, the GCC uses a frequency weighting function to have a higher precision of the TDOA. There are several weightings available. For SSLs most often the PHAT weighting is used, which stands for Phase-Transformation (PHAT).

$$\mathbf{\Psi}_{PHAT}(\omega) = \frac{1}{\left|\mathbf{G}_{X_iX_j}(\omega)\right|} \tag{3.2}$$

If $\mathbf{\Psi}$ is inserted in Equation 3.1 the GCC-PHAT can be obtained by:

$$\mathbf{r}_{ij,PHAT}(k) = iFFT\left(\frac{1}{\left|\mathbf{G}_{X_iX_j}(\omega)\right|} \cdot \mathbf{G}_{X_iX_j}(\omega)\right) \tag{3.3}$$

The weighting function eliminates the magnitude and equally weights the phases for each frequency bin.

The time shifts between each microphone pair can be examined by finding the index of the highest peak in $abs(\mathbf{r}_{ij,PHAT})$. Furthermore, the TDOAs can be calculated by Equation 2.7. To find the direction of the sound source the calculated delays are compared with a precalculated 360° delay grid by calculating the Root-Mean-Squared-

Error (RMSE)

$$\mathbf{e}(\varphi) = \sqrt{(\hat{\tau}(\varphi) - \tau_{ij})^T (\hat{\tau}(\varphi) - \tau_{ij})} \tag{3.4}$$

where $\hat{\tau}$ is the precalculated delay for each angle φ. The direction of the sound source can now be found by searching the minimum in the RMSE vector [9].

$$\varphi = arg \min_{\varphi} \mathbf{e}(\varphi) \tag{3.5}$$

Beamforming

Beamformers are using multiple microphones in so called microphone arrays to spatially filter the incoming sound waves. Beamforming enables the possibility to enhance sounds from specific directions and attenuated disturbing noise sources from all other directions. The focus of this thesis is on computationally efficient beamformers. Therefore, chapter 2.3 (page 11ff) is dedicated to this topic.

Voice Assistant

Modern cloud based voice assistants such as Amazon Alexa or Google Assistant are controlled by voice and can perform specific task or answer arbitrary questions. An interface connects the voice assistant device, where the signal processing is done, with the voice assistants. Google recommends to transfer 100 ms of processed audio data at once via 16 bit PCM-Samples with a sampling rate of 16 kHz to a server [18]. At the server the data is analyzed and the corresponding action (e.g. answer to a question, command for other devices etc.) are sent back to the voice assistant devices.

3.2 Cerebro - MultiMicBoard

The Cerebro-MultiMicBoard, which has been developed within this thesis, is designed to evaluate the capabilities of high SNR microphones in combination with array processing algorithms for voice assistants such as Amazon Alexa or Google Assistance. The board is equipped with an ARM® Cortex®-M4 XMC4700 Microcontroller and two Uniform Circular Microphone Array (UCA) with different digital MEMS microphone types from Infineon, as can be seen in Figure 3.2.

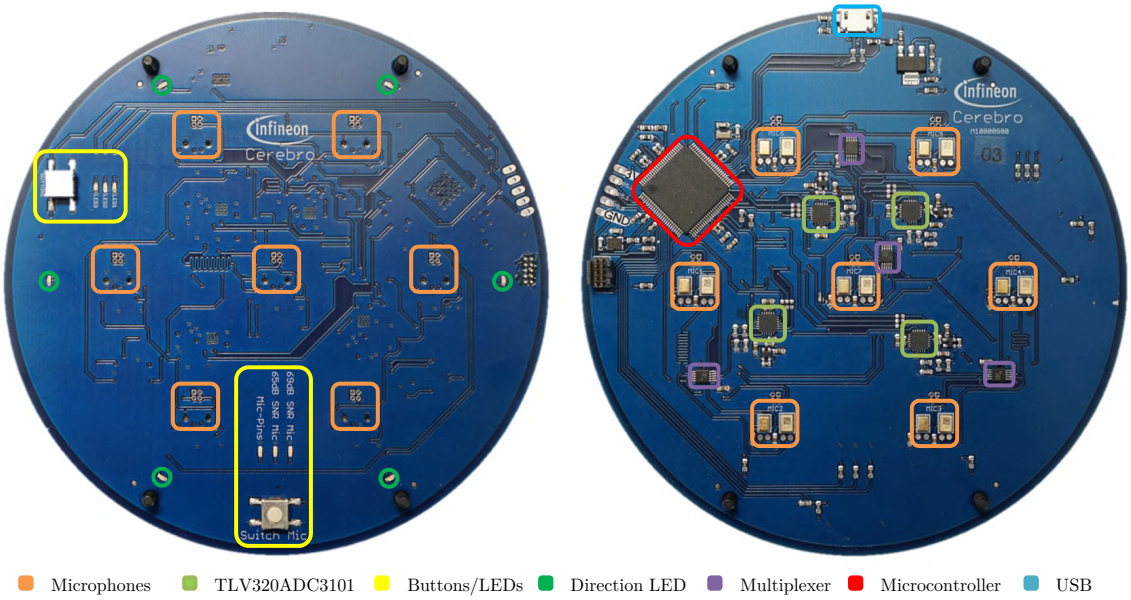


Figure 3.2: Cerebro-MultiMicBoard

The UCAs are respectively equipped with six microphones evenly distributed on a circular path with a diameter of 70 mm and one microphone in the center of the array. The two UCAs can be differentiated by the SNR of the placed microphones. One array is assembled with microphones with a SNR of 69 dBA (Infineon IM69D120). The other array is equipped with microphones with a SNR of 65 dBA (Gettop Acoustic GD-HRA263 261 which uses the Infineon ASIC). Additionally to the two microphone arrays, a third circular array with pins for external microphones is envisaged on the board. On the topside of the Cerebro-Board are six LEDs which are evenly distributed around in a circular path. These LEDs can be used to display the direction of the speaker. Further detailed information about the dimensions of the board can be found in the appendix on page XV.

Figure 3.3 depicts the concept of the Cerebro-Board. Each microphone samples the incoming sound waves with an oversampling factor of 128 ($16\ kHz \cdot 128 = 2.048\ MHz$).

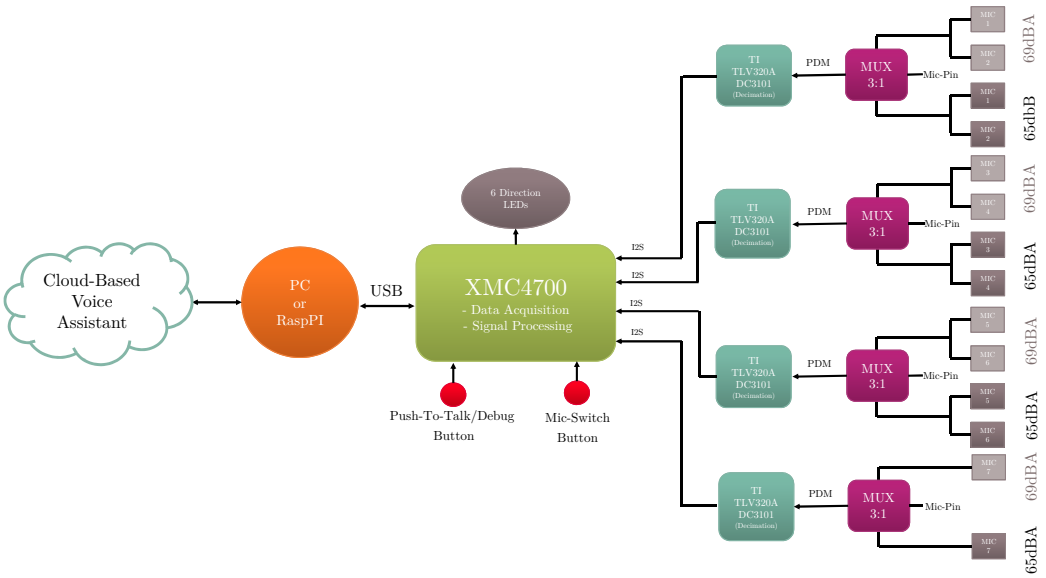


Figure 3.3: Cerebro Concept

The digital output is a Pulse-Density-Modulated (PDM) stream which is defined to transmit two channels (R/L) concurrently. The output of always two microphones of the same UCA are connected with each other and used as one input channel for one of the four 3:1 Multiplexer (MUX). The MUX enables the possibility to switch between the three different UCAs via the *Mic-Switch-Button* or software. The output channel of each MUX is connected with one of the four TI TLV320ADC3101. This Integrated Circuit (IC) decimates the incoming 2.048 MHz PDM data stream to a 16 kHz PCM data stream which is sent via an I2S-Bus to the microcontroller. The microcontroller can operate in two modes:

1.) **Data Acquisition Mode**

In this mode, the XMC is used to simply sample all seven microphone channels and send it via USB to a PC or Raspberry Pi®. This mode can be used to read and post process the recorded data with e.g. Matlab™ or Python. A companion Matlab™ class has been developed to interface between a PC and the Cerebro-MultiMicBoard via the serial port.

2.) **Signal Processing Mode**

This mode can be used to implement microphone array signal processing algorithms directly on the microcontroller by using the C programming language. The resulting mono audio channel can be read in via a companion Python class for the Raspberry Pi®. The Raspberry Pi® acts as gateway to a cloud based voice assistant. Alternatively, the data can also be read in via the provided Matlab class and analyzed on the PC.

3.3 Computational Costs and Memory Analysis

Modern beamforming algorithms are often designed with the focus on the noise reduction performance regardless of the computational costs. This is fine as long as a modern computer with high processing power and large RAM can be used. Since voice assistant devices are mostly powered by inexpensive microcontrollers, a more realistic performance evaluation can be achieved by taking the needed computational costs as well as the used memory of the signal processing algorithms into account.

3.3.1 Hardware Resources Infineon XMC4700

For a long time the first choice for Digital-Signal-Processing (DSP) tasks were digital signal processors. These types of processors are highly optimized for mathematical operations and even capable of executing some of them concurrently in so called Multiply-Accumulate (MAC) units. However, this kind of processors struggle when it comes to general purpose tasks such as interfacing with sensors [19]
Microcontrollers, on the other hand, are excellent for general purpose tasks and connectivity. They are built to interface with sensors, control actors or handle user-interfaces while consuming minimal power. However, microcontrollers are very limited when it comes to performing mathematical operations due to their lack of special registers and computation modules [19].
Previously, when systems similar to voice assistant devices were developed, always a complex combination of a digital signal processor with a microcontroller had to be used. The Microcontroller (µC) interfaced with the sensors, and the digital signal processor computed mathematical tasks. This changed with the invention of microcontrollers with DSP capabilities, making it possible to interface with sensors and compute mathematical operations in one single IC [19].
The Cerebro-MultiMicBoard is equipped with an XMC4700 F100K2048 Microcontroller from Infineon. This controller is powered by an 144MHz ARM® Cortex®-M4 processor with a built-in DSP instruction set. The instruction set in combination with a single-precision Floating-Point-Unit (FPU) and a Direct-Memory-Access (DMA) feature makes it suitable for computing digital signal processing algorithms on it.
In contrast to modern PCs with gigabytes of Random-Access-Memory (RAM), microcontrollers have to get along with a very limited amount of memory. The used µC has 2 MB of flash storage for constants e.g. coefficient tables and 352 kB RAM for the executable program and variables [20]. It is mandatory to design a signal processing chain with a small memory footprint to prevent running out of memory space.

3.3.2 Required Hardware and Memory Resources for Basic Mathematical Operations

There are various ways of calculating the computational costs of an algorithm. Within this thesis the focus is on mathematical operations and memory usage. Therefore, the used algorithms are analyzed by comparing the amount of Additions (ADDs), Multiplications (MULs) and Divisions (DIVs) needed.
The XMC4700 provides dedicated hardware (MAC and division/inversion units) for these operations. Therefore, it is assumed that these mathematical operations take the same amount of time to be calculated. To simplify the analyses, assignments as well as the initialization of variables are not considered. Furthermore the overhead produced by function calls, loops or conditional statements are not taken into account. Also some mathematical operations with low computational costs such as matrix transposition or complex conjugate are neglected.
Since some algorithms are working with complex data the complex mathematical operations can be reduced to real operations in the following way [13].

- 1 Complex ADDs = 2 Real ADDs

  $$(a + bi) + (c + di) = (a + c) + i(b + d)$$

- 1 Complex MULs = 2 Real ADDs and 4 Real MULs

  $$(a + bi) \cdot (c + di) = (ac - bd) + i(ad + bc)$$

- 1 Complex DIVs = 3 Real ADDs + 8 Real MULs + 1 Real DIV

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

Most of the algorithms are computed using blocks/frames with N audio samples. If an algorithm works on each incoming sample individually, also N audio samples are considered for one processing step to be comparable. Block processing algorithms are often using matrix operations which can be broken down to real operations [13] [21].

- Vector Dot Product $(1 \times n \cdot n \times 1)$
  - ADDs: $n - 1$
  - MULs: n
  - DIVs : 0
- Scalar $\cdot$ Matrix $(n \times m)$
  - ADDs: 0
  - MULs: nm
  - DIVs : 0

- Matrix Multiplication ($n \times l \cdot l \times m$)
  - ADDs: $nm(l-1)$
  - MULs: nml
  - DIVs : 0
- Matrix Addition/Subtractions ($n \times l \pm n \times l$)
  - ADDs: nl
  - MULs: 0
  - DIVs : 0
- Matrix Inversion ($n \times n^{-1}$)
  - ADDs: $n^3 + n^2 - 3n + 2$
  - MULs: $n^3 + n^2 - 3n + 2$
  - DIVs : n

Some of the algorithms are computed in frequency domain. Therefore, the audio frame of N samples has to be transfered from time to frequency domain using a Fast-Fourier-Transformation (FFT). From literature one can state that $N \log_2(N)$ real ADDs and real MULs are needed to compute the FFT of N real input samples [13].

To address the limited storage problem of microcontroller also an assessment of the needed memory space for each algorithm is examined. The datatype of all input samples are considered to be FLOATs with a word length of 32bit (or 4 bytes) each.

3.3.3 Delay and Sum Beamformer

The D&S-BF is the most efficient implementation of a beamforming algorithm. The input samples are first delayed according to their geometrical position in the array and the DOA, and are then summed up and scaled by the number of microphone channels. This procedure can be done either in time or frequency domain.

3.3.3.1 Time-Domain Implementation

The time-domain implementation is, due to its simplicity and efficiency, the most common implementation of a D&S-BF. The pseudo-code of a time-domain D&S-BF implementation can be seen in Algorithm 3.1

Algorithm 3.1 Time-Domain Delay-and-Sum Beamformer

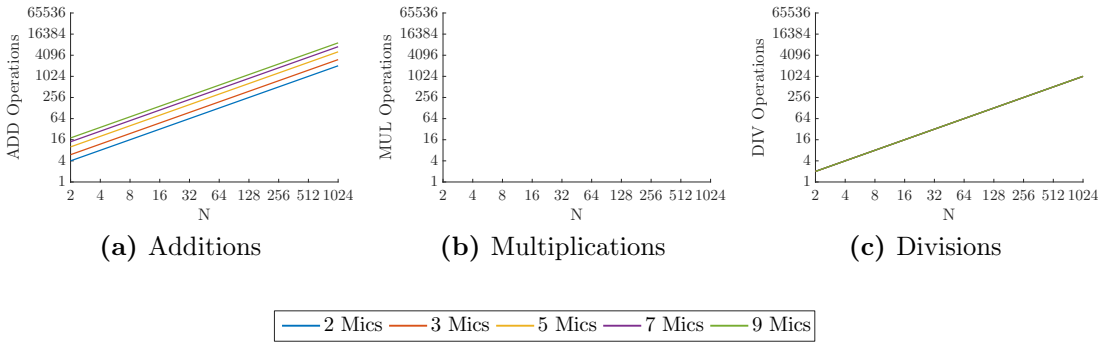
```
for all samples of frame do
    for all microphones do
        Delay or advance sample according to DOA
        Sum up samples
    end for
    Scale sum by number of microphones
end for
```

Table 3.1 lists the needed mathematical operation for a Delay-And-Sum Beamformer.

Table 3.1: Computational Cost: Time-Domain Delay-and-Sum Beamformer

Operation	ADD	MUL	DIV
Delay&Sum	NM	0	0
Scale	0	0	N
Total	**NM**	**0**	**N**

One can clearly see that the needed computations are low compared to other beamformer implementations which are analyzed on the following pages.



(a) Additions **(b)** Multiplications **(c)** Divisions

Figure 3.4: Computational Cost: Time-Domain Delay-and-Sum Beamformer when using different microphone quantities (axis are log_2 scaled)

Figure 3.4 shows the computational costs of the time-domain D&S-BF in dependence of the frame size N and the amount of used microphones M. The figures were plotted using a log_2 scaling of the y and x axis. One can see in Figure 3.4a that the dominating operations are the additions which are linearly dependent on the number ob microphones and the frame size. The divisions, pictured in Figure 3.4c, are only dependent on the frame size. The D&S-BF needs no multiplications.

The TD-D&S-BF is also from the memory perspective a very efficient BF. It basically needs just one variable where the shifted samples of each microphone are summed up.

3.3.3.2 Frequency-Domain Implementation

The frequency-domain implementation is sometimes used, if data is already available in frequency domain or the output is needed in frequency domain. For comparison reasons, the transformations into frequency and time domain have to be considered in the computational costs analysis. The pseudo code of a frequency implementation of the D&S-BF can be seen in Algorithm 3.2

Algorithm 3.2 Frequency-Domain Delay-and-Sum Beamformer

```
Apply FFT
for each frequency bin do
    for all microphones do
        Delay or advance sample according to DOA
        Sum up sample
    end for
    Scale sum by number of microphones
end for
Apply IFFT
```

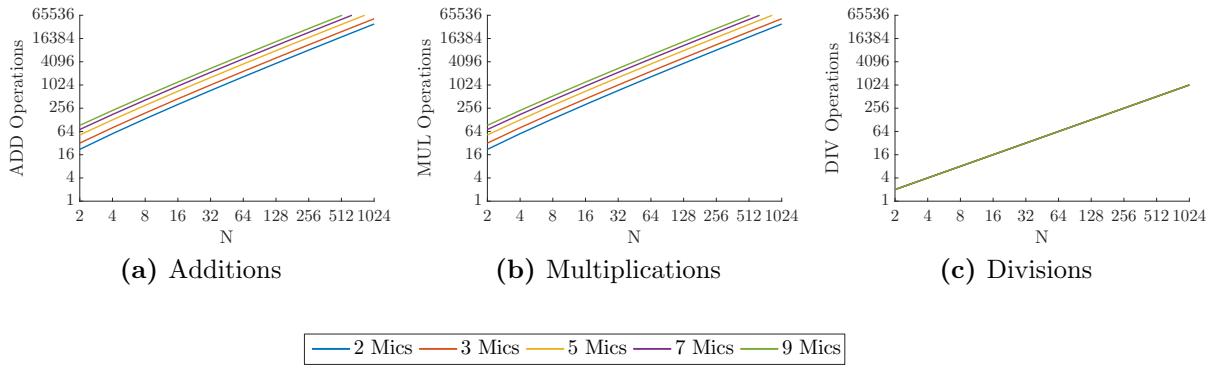
Table 3.2: Computational Cost: Frequency-Domain Delay-and-Sum Beamformer

Operation	ADD	MUL	DIV
FFT	$MNlog_2(N)$	$MNlog_2(N)$	0
Delay&Sum	$2NM$	NM	0
Scale	0	0	N
IFFT	$Nlog_2(N)$	$Nlog_2(N)$	0
Total	$\mathbf{N(log_2(N)(M+1)+4M)}$	$\mathbf{N(log_2(N)(M+1)+4M)}$	**N**

The time delay for each microphone can be precalculated in a grid of different DOAs. The frequency-domain D&S-BF is less efficient than the time-domain implementation since all the Delay&Sum operations have to be implemented as complex operations which need at least twice as many real operations.

Figure 3.5 shows the computational costs of the frequency-domain D&S-BF. The figures were plotted using a log_2 scaling of the y and x axis. One can see that the number of operations for ADDs and MULs increase equally while the number of divisions rises with the frame size.

The memory usage of the FD-D&S-BF is significantly higher than the TD-D&S-BF. When the ARM FFT from the CMSIS-DSP library is used the BF needs two arrays with a length of N. In addition to that, again one sum variable is needed. The time delay according to the DOA can be precalculated and saved as constant in the flash



(a) Additions **(b)** Multiplications **(c)** Divisions

Figure 3.5: Computational Cost: Frequency-Domain Delay-and-Sum Beamformer when using different microphone quantities (axis are log_2 scaled)

memory. It depends on the resolution of the precalculated grid how much memory is used for that.

3.3.4 MVDR Beamformer

The MVDR-BF is always applied in the frequency domain and can either be implemented as time-independent or adaptive BF. In contrast to the static MVDR-BF implementation, the adaptive MVDR-BF can adjust its weights with a changing noise environment. This is good for the noise reduction performance, but results in an increase of the needed computations.

3.3.4.1 Static MVDR

The time-independent MVDR-BF uses precalculated weights which are convoluted with the input samples. This is done by multiplying each frequency bin of the microphone channels with the weights in frequency domain as can be seen in Algorithm 3.3

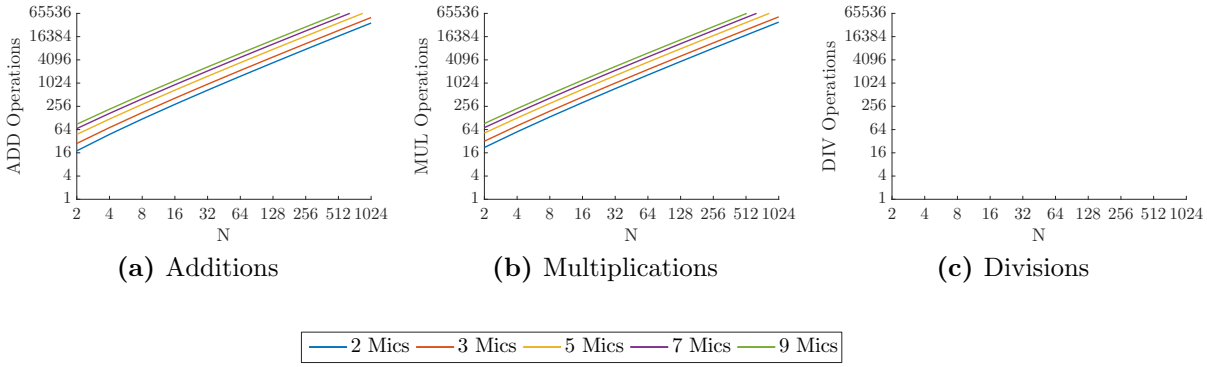
Algorithm 3.3 Static MVDR Beamformer

```
1: Apply FFT
2: for each frequency bin do
3:     for all microphones do
4:         Multiply frequency bin with precalculated filter weight
5:     end for
6: end for
7: Apply IFFT
```

Table 3.3: Computational Cost: Static MVDR

Operation	ADD	MUL	DIV
FFT	$MNlog_2(N)$	$MNlog_2(N)$	0
Convolution	$4NM - 2N$	$4NM$	0
IFFT	$Nlog_2(N)$	$Nlog_2(N)$	0
Total	$\mathbf{N(log_2(N)(M+1)+4M-2)}$	$\mathbf{N(log_2(N)(M+1)+4M)}$	**0**

Table 3.3 shows the computational costs of a static MVDR-BF implementation. One can see that the number of operations are similar the costs of a frequency domain D&S-BF.



(a) Additions **(b)** Multiplications **(c)** Divisions

Figure 3.6: Computational Cost: Static MVDR Beamformer when using different microphone quantities (axis are log_2 scaled)

Figure 3.6 depicts the computational costs in dependence of the frame size N and the amount of used microphones M. The figures were plotted using a log_2 scaling of the y and x axis. The major part of the MULs are computed concurrently with the ADDs in the MAC unit of the µC. Except for the DIVs, the computational costs are similar to the costs of the FD-D&S-BF.

Also the memory footprint is very similar to the FD-D&S-BF. The NM precalculated filter weights can be stored in the flash memory. The FFT needs two N sized arrays to transfer a audio frame into frequency domain.

3.3.4.2 Adaptive MVDR

The time-dependent MVDR-BF can adjust its weights according to the environment. It updates the filter coefficient when there is no speech detected through the VAD. Therefore, the BF has the same computational costs as the static MVDR-BF during speech phases. If no speech is detected, the computational footprint increases dramatically. In these frames the weights are updated which needs among other operations

N computational heavy $M \times M$ matrix inversions. The pseudo-code of an adaptive MVDR-BF implementation can be seen in Algorithm 3.4.

Algorithm 3.4 Adaptive MVDR Beamformer

```
Apply FFT
if Frame is labeled as noise then
    for each frequency bin do
        Calculate covariance matrix
        Apply Diagonal Loading to covariance matrix
        Invert covariance matrix
        Update filter weight
    end for
end if
Apply filter weight
Apply IFFT
```

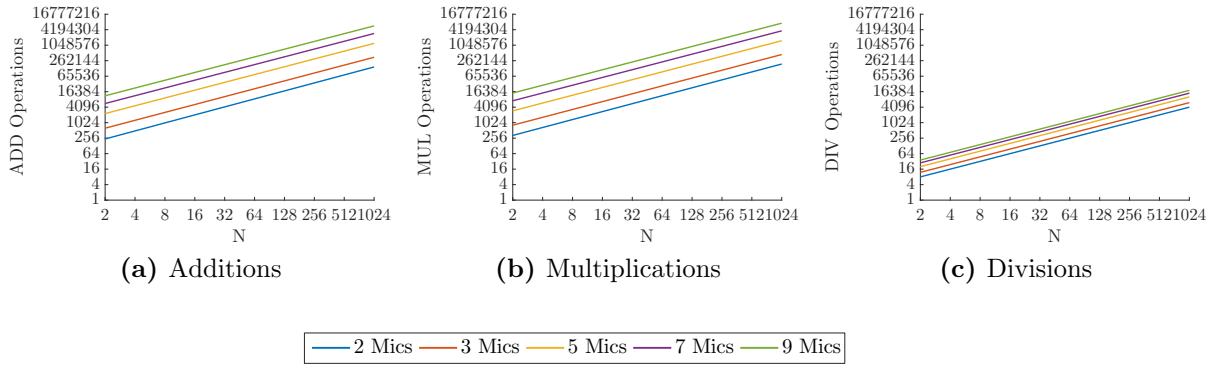
The analysis of the computational costs of the beamformer when a noise frame is detected and the filter weights are updated can be seen in Table 3.4.

Table 3.4: Computational Cost: Adaptive MVDR

Operation	**ADD**	**MUL**	**DIV**
FFT	$MNlog_2(N)$	$MNlog_2(N)$	0
Covariance Matrix	$2M^2N(M+1.5)$	$4M^2N(M+1)$	0
Diagonal Loading	M^2N	0	0
Inverting Matrix	$N(4M^3+4M^2-9M+8)$	$N(4M^3+4M^2-4M+8$	NM
Calculate Weights	$N(8M^2+3M+2)$	$MN(8M+12)$	NM
Convolution	$4NM-2N$	$4NM$	0
IFFT	$Nlog_2(N)$	$Nlog_2(N)$	0
Total	$\mathbf{N(6M^3+16M^2-2M}$ $\mathbf{+8+log_2(N)(M+1))}$	$\mathbf{N(8M^3+16M^2+12M}$ $\mathbf{+8+log_2(N)(M+1))}$	**2NM**

The computational costs depending on frame size N and used microphones M can be seen in the Figure 3.7. The figures were plotted using a log_2 scaling of the y and x axis. One can see that the computational costs are significantly higher than the previously analyzed BFs. Also the dependency of the used microphone is much higher than before. More microphones results in large increasing steps in the required ADD and MUL operations.

The adaptive MVDR-BF needs N $(M \times M)$ covariance matrices. This in combination with the needed memory for the FFT and the NM filter coefficient array makes this BF very expensive. Since all the variables are needed to be changeable during noise frames no variables can be saved to the flash memory. Therefore, this BF is not implementable on the used microcontroller.



(a) Additions **(b)** Multiplications **(c)** Divisions

Figure 3.7: Computational Cost: Adaptive MVDR Beamformer when using different microphone quantities (axis are log_2 scaled)

3.3.5 GSC Beamformer

The GSC-BF was introduced as an alternative to the computational expensive adaptive MVDR-BF. It consist of a time-invariant Beamformer (e.g. D&S-BF) which enhances the wanted signal components in the input stream and a canceling path (Blocking Matrix + Interference Canceler) which cancels unwanted components from the beamformer output. A GSC-BF can be implemented in various ways. Within this thesis the following three types of a GSC-BF are analyzed:

- Time Domain
  - Griffiths and Jim GSC-BF
  - Robust GSC-BF
- Frequency Domain
  - Robust GSC-BF

3.3.5.1 Time-Domain Implementation

Griffiths and Jim GSC

Griffiths and Jim were one of the first who came up with the idea of using a GSC-BF for array processing. Their implementation consists of a D&S-BF, a Blocking Matrix, and an Adaptive-Interference-Canceler which is basically a Multiple-Input-Single-Output (MISO) adaptive filer. This constellation makes it possible to implement an computational efficient BF which is capable of running in real-time on voice assistant devices. The pseudo-code of an GJGSC-BF can be seen in Algorithm 3.5.

Algorithm 3.5 Griffiths and Jim GSC Beamformer

```
for all samples of frame do
    for all microphones do
        Apply TD-D&S-BF
        Apply BlockingMatrix to aligned samples
        Apply AIC Weights
        Update NLMS Stepsize
        Update AIC Weights (NLMS-Algorithm)
    end for
end for
```

The adaptive filter order for the computational cost analysis is kept to one. If a higher filter order is needed, the adaptive filter computations will increase by the filter order. The computational footprint of a Griffiths and Jim GSC-BF is listed in Table 3.5.

Table 3.5: Computational Cost: Griffiths and Jim GSC

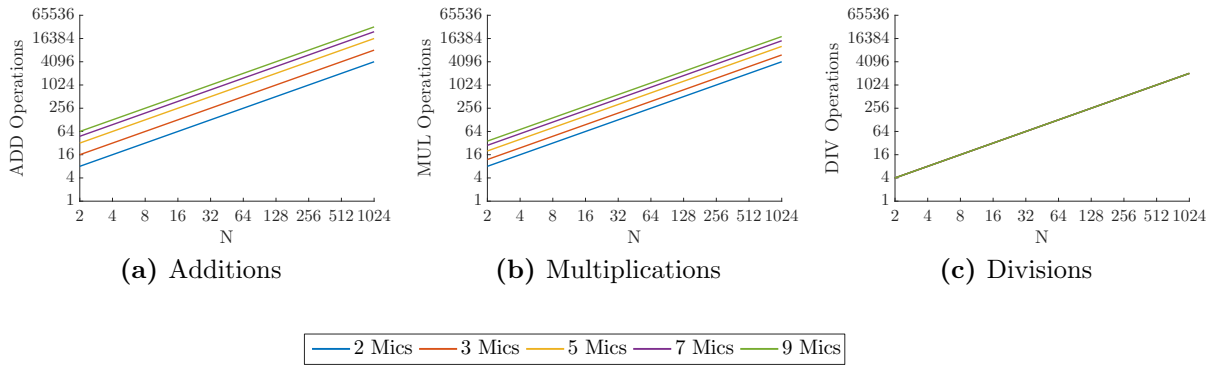
Operation	**ADD**	**MUL**	**DIV**
TD-D&S-BF	NM	0	N
Blocking Matrix	$N(M-1)$	0	0
Apply AIC Weights	$N(M-2)$	$N(M-1)$	0
Obtain Error/Output Signal	N	0	0
Calculate Step Size	N	N	N
Calculate AIC Weights	$N(M-1)$	$N+N(M-1)$	0
Total	$2N(2M-1)$	$2NM$	$2N$

The Blocking Matrix can be implemented as simple subtractions of each two adjacent input streams. Hence, $N(M-1)$ ADDs are taken into account for the blocking matrix step. The dependency of the computational costs in respect to the frame size and used microphone number can be seen in Figure 3.8. The figures were plotted using a log_2 scaling of the y and x axis.

The memory usage can be split up in three parts. First is the time-invariant BF which is usually a TD-D&S-BF. According to section 3.3.3.1 the D&S-BF needs only one sum variable which is very effective.

The second part is the blocking matrix. This part needs $M-1$ temporary arrays with a size of N to store the $M-1$ blocking matrix output streams.

The last part is the AIC which consists of an MISO adaptive filter. It needs the four variables y, e, μ and E which hold the filter output, the produced error, the step size and the input energy, respectively. In addition to that an array of M filter weights is needed. Since this BF is processed in time domain no additional arrays for a FFT is needed.



(a) Additions **(b)** Multiplications **(c)** Divisions

Figure 3.8: Computational Cost: Griffiths and Jim GSC when using different microphone quantities (axis are log_2 scaled)

Although the Griffiths and Jim GSC-BF is solving the same minimization problem as the adaptive MVDR-BF, the GSC-BF has a significantly smaller computational footprint. Also from the memory perspective, this BF is more applicable for the usage in microcontrollers. The Blocking Matrix uses M temporary arrays with a size of N and five variables for the AIC.

Robust GSC

The Griffiths and Jim GSC-BF has the downside that it relies on a precise SSL. A DOA Error would result in a degradation of GSC-BF output quality. The Robust GSC-BF addresses this issue by replacing the Blocking Matrix with an Adaptive-Blocking-Matrix which consists out of M adaptive filters. Algorithm 3.6 shows the pseudo-code for a RGSC. The compuational cost analysis assumes that all adaptive filters have a filter order of one.

Algorithm 3.6 Robust Time-Domain GSC Beamformer

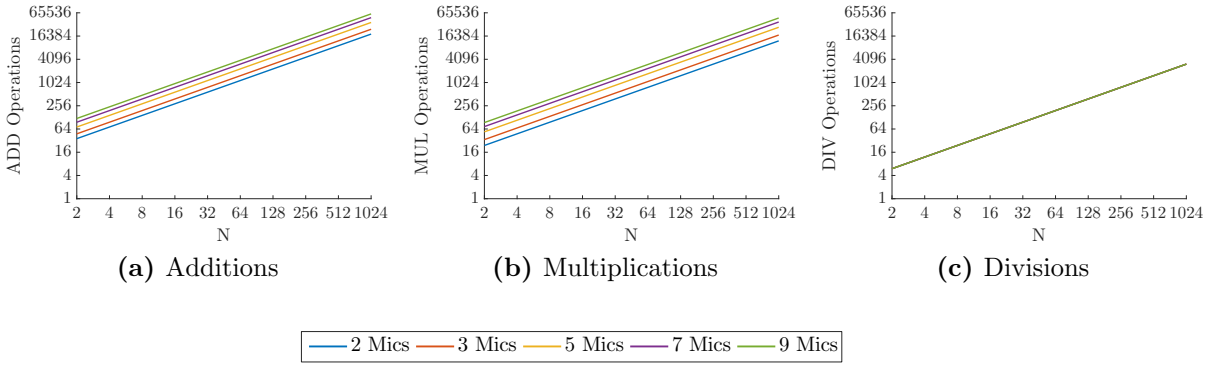
```
for all samples of frame do
    for all microphones do
        Apply TD-D&S-BF
        Apply ABM
        Apply AIC
        Optain GSC output signal
    end for
end for
```

This implementations enhances the sound quality of the output signal but with the costs of a higher computational footprint as can be seen in Table 3.6.

Table 3.6: Computational Cost: Robust Time-Domain GSC

Operation	ADD	MUL	DIV
TD-D&S-BF	NM	0	N
Apply ABM Weights	0	NM	0
Obtain ABM Error Signal	NM	0	0
Calculate ABM Step Size	$N + N(M-1)$	NM	N
Calculate ABM Weights	NM	$N(M+1)$	0
Apply AIC Weights	$N(M-1)$	NM	0
Obtain AIC Error/Output Signal	N	0	0
Calculate AIC Step Size	N	N	N
Calculate AIC Weights	NM	$N(M+1)$	0
Total	$N(6M+1)$	$N(5M+2)$	$3N$

The computational costs in respect to the frame size and the number of used microphones can be seen in Figure 3.9. The figures were plotted using a log_2 scaling of the y and x axis.



(a) Additions **(b)** Multiplications **(c)** Divisions

Figure 3.9: Computational Cost: Robust Time-Domain GSC when using different microphone quantities (axis are log_2 scaled)

One can see that the added second adaptive filter increases the computational costs slightly but with the advantage of being more robust against DOA errors.

From the memory perspective this BF is more efficient than the previously examined GJGSC-BF. This results from the replacement of the blocking matrix with M adaptive filters. These filters need only five variables (y,e, μ, E and filter weight) per filter which is much less than the Blocking Matrix requires.

3.3.5.2 Frequency Domain Implementation

In [16] Herbordt and Kellermann proposed a frequency-domain implementation of the Robust GSC-BF. They replaced all adaptive filters with frequency-domain adaptive

filters which are computationally more efficient with the costs of requiring more memory space.

Algorithm 3.7 Robust Frequency-Domain GSC Beamformer

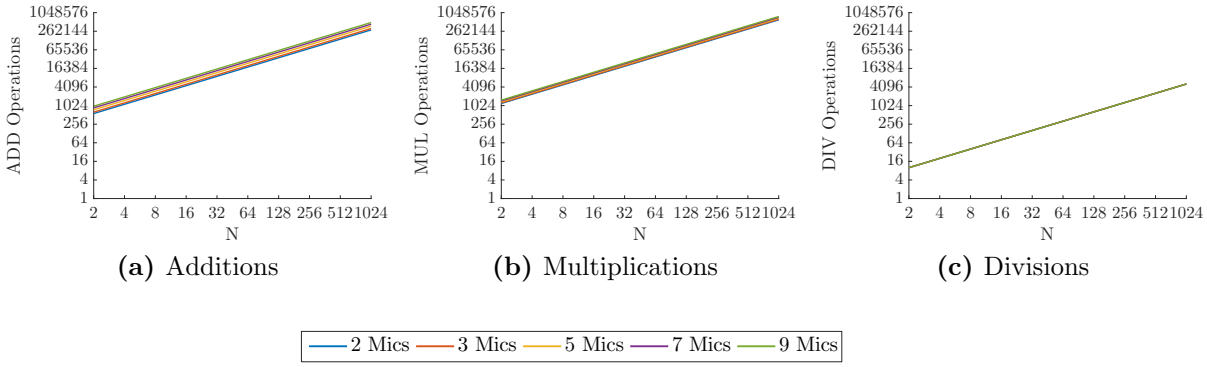
```
for all microphones do
    Apply TD-D&S-BF
    FFT
    for each frequency bin do
        Apply ABM
    end for
    IFFT
    Obtain ABM output
    FFT
    for each frequency bin do
        Apply AIC
    end for
    IFFT
    Optain GSC output signal
end for
```

The computational costs of each processing step are listed in Table 3.7.

Table 3.7: Computational Cost: Robust Frequency-Domain GSC

Operation	**ADD**	**MUL**	**DIV**
TD-D&S-BF	NM	0	N
$FFT(y_{BF})$	$Nlog_2(N)$	$Nlog_2(N)$	0
Apply ABM Weights	$2NM$	$4NM$	0
$IFFT(Y_{ABM})$	$MNlog_2(N)$	$MNlog_2(N)$	0
Obtain ABM Error Signal	$^{NM}/_2$	0	0
$FFT(e_{ABM})$	$Nlog_2(N)$	$Nlog_2(N)$	0
Calculate ABM Power	$4N$	$8N$	0
Calculate ABM Step Size	0	$2N$	$2N$
Calculate ABM Weights	$6NM$	$8NM$	0
Circular Shift	$2NM$	$4NM$	0
Apply AIC Weights	$4NM$	$4NM$	0
$IFFT(Y_{AIC})$	$Nlog_2(N)$	$Nlog_2(N)$	0
Obtain AIC Error/Output Signal	$^{N}/_2$	0	0
Calculate AIC Power	$2N(2M+1)$	$4N(M+1)$	0
$FFT(e_{AIC})$	$Nlog_2(N)$	$Nlog_2(N)$	0
Calculate AIC Step Size	N	$2N$	$2N$
Calculate AIC Weights	$6NM$	$8NM$	0
Total	$N(25.5M+6.5+log_2(M+4)))$	$N(21M+24+log_2(M+4)))$	$5N$

Figure 3.10 shows the relationship between the computational costs and the frame size as well as number of used microphones. The figures were plotted using a log_2 scaling of the y and x axis.



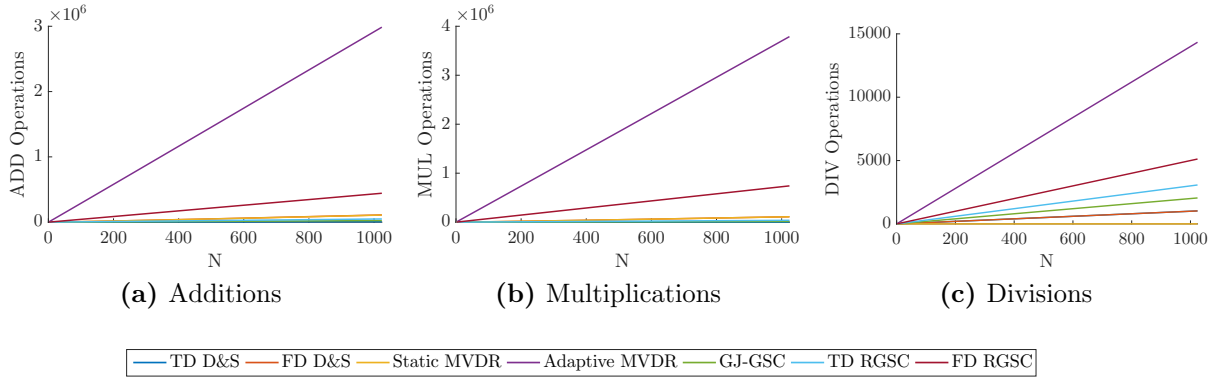
(a) Additions **(b)** Multiplications **(c)** Divisions

Figure 3.10: Computational Cost: Robust Frequency-Domain GSC when using different microphone quantities (axis are log_2 scaled)

The FD-RGSC requires clearly more memory space then the time domain implementations of the GSC-BF. This is because several FFTs are needed. In addition to that, an array of N filter weights for each microphone and adaptive filter is needed.
One can see that the implementation is more efficient than the adaptive MVDR-BF approach but cannot outperform the efficiency in terms of computational costs and memory usage of the TD-RGSC.

3.3.6 Comparison of the evaluated Beamformers

The previous sections examined each beamforming algorithm individually. Figure 3.11 shows the examined beamforming algorithms together. For better comparability the different algorithms are examined in the case of a seven microphone constellation. The figures are plotted using linear scale for the x and y scale for a better visibility of the computational cost differences of the examined beamforming algorithms. One can see that the adaptive MVDR-BF (purple lines) is clearly the computational most expensive BF due to the updating of its filter weights during noise frames. The dark red line depicts the frequency domain RGSC. It needs less computations then the adaptive MVDR-BF but is still significantly more expensive then all the other BFs. This comes due to the several FFT/IFFT which are needed at the ABM and AIC to cancel the interfering sound components from the time-invariant BF. Therefore, it depends on the used processor, how fast the FFT can be calculated. Special processors are nowadays available that have dedicated FFT hardware units which are sometimes capable to



(a) Additions **(b)** Multiplications **(c)** Divisions

Figure 3.11: Computational Costs of different Beamformer Implementations when using 7 Microphones

calculate even one frequency bin per clock cycle (e.g. Infineon Aurix TC3XX). Within this thesis an ordinary low cost microcontroller is used which does not have that feature. Hence, it needs the examined computations for each FFT/IFFT.
All the other BF are from the computational cost perspective similar. To depict this result even more, the computational costs for each beamformer when using a frame size of 256 samples and 7 microphones can be seen in the Table 3.8.

Table 3.8: Computational Cost: Example with N=256, M=7

Beamformer	**ADD**	**MUL**	**DIV**
TD-D&S-BF	1792	0	256
FD-D&S-BF	23552	23552	256
Static MVDR-BF	23040	23552	0
Adaptive MVDR-BF	742400	943104	3584
GJ-GSC-BF	6144	3584	512
TD-RGSC	12288	9472	768
FD-RGSC	110711	185253	1280
Total	**NM**	**0**	**N**

One can see in the example that the adaptive MVDR-BF is 5/6.7 times (MUL/ADD) more computational expensive as the FD-RGSC. Compared to the TD-RGSC it has a even 60/100 times heavier computational footprint. The best trade off between computational cost and noise canceling performance when using the XMC microcontroller is given by the time-domain RGSC which addresses the same minimization problem as the adaptive MVDR-BF with a much lower computational footprint.
When comparing the memory consumption of all BF, it can be seen that all time domain BF need fewer memory space since no FFTs are needed. The most expensive BF in terms of memory usage is clearly the adaptive MVDR-BF followed by the FD-RGSC. The TD-RGSC provides from the memory perspective the best trade off.

The comparison has shown that the optimal beamformer for a implementation on the used microcontroller is the TD-RGSC. It provides from the computational costs as well from the required memory space the best trade off.
Within this thesis all mentioned BF are implemented in Matlab. Additionally to that, the TD-D&S-BF as well as the TD-RGSC are implemented directly on the microcontroller of the Cerebro-MultiMicBoard.

3.3.7 Optimization Possibilities

Sound signal processing algorithms are in general computational expensive. The combination of high sampling frequencies and the need of distortion-less processing of succeeding sound frames makes it essential to find possibilities to reduce the needed computations.
The simplest way of optimizing algorithms when using C and microcontrollers is to turn the compiler optimization to the highest level (-O3). This forces the compiler to rearrange the written code in a way that the resulted machine code is optimized for the used processor architecture [19].
Another way to speed up the algorithms is to use optimized math libraries for operations such as FFT, dot product or matrix/vector multiplications. ARM provides the CMSIS-DSP library for its processor architecture. This library is perfectly fitted to the built in features (e.g. MAC Unit, HW Division Unit) of the ARM processor and makes it possible to calculate for example multiple addition and multiplications within one clock cycle. [19]. The algorithms itself can also be optimized. Some algorithms can be represented in frequency domain more efficiently than in time domain. The speed up can be achieved, if the whole signal processing chain can be transfered into the frequency domain and only one FFT at the beginning and one IFFT at the end is needed. Reducing the amount of microphones is also a way of reducing computations as well as memory usage. It has to be declared if this may reduces the noise canceling capabilities or the SSL accuracy. The influences of reducing microphones is not scope of this thesis and therefore not evaluated.
Reducing the frame size also results in a decrease of the needed computations. All the algorithms need to be real-time capable and must provide low-latency [5]. The smaller the frame size, the lower the latency. However, when using too small frames, the needed time for the mathematical operations and its overhead, may not be long enough to finish all operations before a new frame is coming. Hence, no real-time processing would be possible. Although longer frames would need also more computations, the time needed because of overhead (e.g. variable initialization, function calls etc.) would decrease.

4 Evaluation of Beamforming Algorithms for Voice Assistant Devices

In this chapter the implemented beamforming algorithms are evaluated using the two different microphone types on the Cerebro-MultiMicBoard. The processed audio data is transmitted to the Google Assistant which returns the recognized words. These words can further be used to examine the Word-Error-Rate (WER).

4.1 Measurement Setup

4.1.1 Measurement Room

The measurements are examined in a specific room which is designed as a semi-anechoic chamber as can be seen in Figure 4.1. This means that the walls as well as the ceiling are able to highly absorb sound waves. Only the floor is solid and can reflect energy. Therefore, this room is nearly reverberation free with a T60 less than 0.1s. Artificial reverberations as well as reflections can be controlled via the sound interface of the room [22].

Figure 4.1: Measurement Room for Voice Assistant Devices [22]

4.1.2 Speaker Setup

The measurement room is equipped with four Dynacord D8 loudspeakers which are controlled with a RME Fireface UFX Audio Interface. The speakers are chosen to have a small directivity to replicate the directivity of the human mouth. The frequency response of the speakers is corrected to vary only $\pm 3\ dB$ from 100 Hz up to 20 kHz. Three speakers are used to produce noise and the remaining fourth speaker is used as demanded sound source. The position of the speakers is depicted in Figure 4.2.

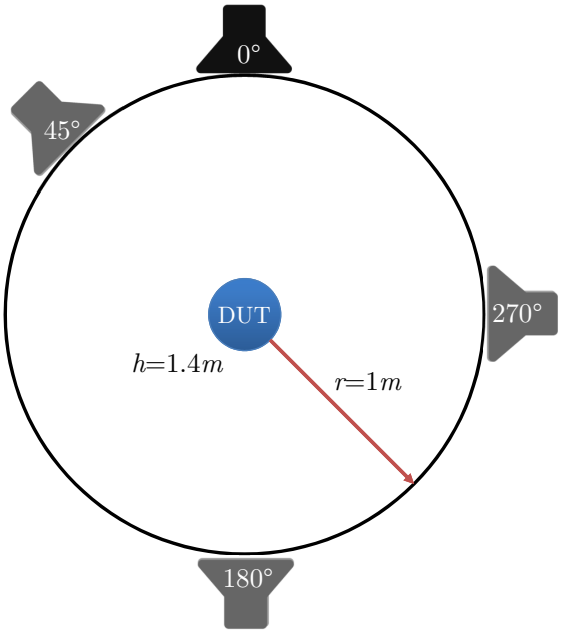


Figure 4.2: Used Speaker Setup for Evaluation of Beamforming Algorithms

The black speaker in Figure 4.2 depicts the position of the sound source at 0°. The gray speakers display the position of the noise sources at 45°, 180° and 270°. All speakers are 1 m away from the Device-Under-Test (DUT) which is positioned in the middle of the measurement setup in about 1.4 m elevation.

4.1.3 Device-Under-Test

The Cerebro-MultiMicBoard described in section 3.2 is used as DUT within this evaluation. The Cerebro-Board uses two UCAs with different microphone types (7 microphones each) to capture frames of 256 16 bit-samples with a sample rate of 16 kHz. The microphone types can be differentiated by their SNR (65 dBA and 69 dBA).
All evaluations are executed with both microphone types using the following three capturing modes:

- **Raw**
  - The audio from all 7 microphone channels is recorded without further signal processing.
- **Delay-And-Sum Beamformer (D&S-BF)**
  - The audio from all 7 microphone channels is fed in a time-domain D&S-BF and the resulting beamformed output stream is recorded.
- **Generalized-Sidelobe-Canceler Beamformer (GSC-BF)**
  - The audio from all 7 microphone channels is fed in a time-domain GJGSC-BF and the resulting beamformed output stream is recorded.

4.1.4 Noise Scenarios

The different capturing modes are evaluated by using the following three measuring scenarios.

- **No Noise with decreasing sound source volume level**
  - The Sound-Pressure-Level (SPL) (=volume) of the sound source is varied from 60dB down to 10dB without any additional noise.
- **Babble Noise with constant sound source level**
  - The SPL of the sound source stays constant at 60 dB. The SPL of the babble noise, which comes from the three noise speakers, is varied from 70 dB down to 40 dB. Hence, the noise with a SPL of 70 dB can be considered to be 10 dB louder than the sound source and 20 dB quieter if the noise is at 40 dB.
- **IEC-268 (= Pink Noise) with constant sound source level**
  - The SPL of the sound source stays constant at 60 dB. The SPL of 60 dB is chosen as this is a typical volume level for a human during a conversation [23]. The SPL of the pink noise, which comes from the three noise speakers, is varied from 70 dB down to 40 dB.

The four speakers are calibrated with a SPL-Meter (NTi Audio XL2) and a microphone (NTi M2015 (class 1) (Nr. 1045)) at the DUT to guarantee a correct SPL prior the measurements.

4.1.5 Sound Source Sentences

For the evaluation, eight non-identical sentences with 5-10 words from 4 different speakers from the TIMIT Corpus [24] are used as sound source. Always one female and one

male speaker from 2 different USA regions (Western and New England) with preferable distinct pitches are used. The following sentences from the test database of the TIMIT Corpus [24] are used:

1.) **felc0:** Dialect Region: 1 (New England); Sentences: sx126 + sx216; Gender: Female
   - *Artificial intelligence is for real. The small boy put the worm on the hook.*

2.) **msjs1:** Dialect Region: 1 (New England); Sentences: sa1 + si1899; Gender: Male
   - *She had your dark suit in greasy wash water all year. She took me by surprise.*

3.) **flas0:** Dialect Region: 7 (Western); Sentences: sx138 + si1026; Gender: Female
   - *The clumsy customer spilt some expensive perfume. Careful, don't disturb her.*

4.) **mkjl0:** Dialect Region: 7 (Western); Sentences: sx110 + si470; Gender: Male
   - *The best way to learn is to solve extra problems. A quick touch down resulted.*

Further information about the TIMIT Corpus can be found in [24].

4.1.6 Google Assistant

For the evaluation a Python class was developed during this thesis to send the recorded audio streams to the Google Assistant API. This API supports a transcript feature that returns the recognized words by the Goggle Assistant which can be used to examine the Word-Error-Rate (WER). The WER-Analysis uses the known spoken words and compares them with the recognized words. The lower the WER the more words have been recognized correctly by the cloud based voice assistant.
According to the Google Assistant API in [18], the input volume of voice samples has to be in the region between -10 and -20 dBFS. Therefore, the recorded audio streams with noise are amplified by 20 dB to meet the requirements. The data streams from the scenario where no noise is present are normalized to an amplitude of ± 0.9 to see the impact of the self-noise of the microphones.
Google allows only one input stream at a time. Therefore, for the evaluation of the raw data only microphone 7, which is placed in the center of the UCA, is considered.

4.2 Results

The Word-Error-Rate (WER) of the 4 sentences are evaluated individually and averaged to get one overall WER for each measurement.

For the raw microphone evaluation, only the microphone in the center of the UCA (Microphone 7) was used without further processing.

Within the D&S-BF and the GSC-BF evaluation, the captured data from all seven microphones is processed with the beamformer directly on the Cerebro-MultiMicBoard. The SSL and VAD was disabled during the measurements to prevent unwanted influences from other algorithms before the BF. The DOA was fixed to the speech speaker location.

The following parameters where used for the GSC-BF.

- Blocking Matrix:
  - $\mathbf{b} = \begin{bmatrix} 1 & 0 & 0 & \dots & -1 \\ 0 & 1 & 0 & \dots & -1 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$

    Which basically means that the center microphone (Microphone 7) is subtracted from the other input streams.
- Filter Order AIC:
  - $M_{AIC} = 3$ (tap length = 4)
- Stepsize constant
  - $\alpha_{AIC} = 0.0005$
- AIC Shift:
  - $shift_{aic} = 6$

4.2.1 No Noise

Within the "No Noise" scenario the SPL of the speech speaker is varied from 10 to 60 dB. No additional noise was played within this evaluation. To see the influences of the microphone self noise the amplitude of the recorded data is normalized to ± 0.9.

4.2.1.1 Microphone SNR: 65 dBA

The results from the 65 dBA SNR microphone in the no noise scenario can be seen in Figure 4.3.

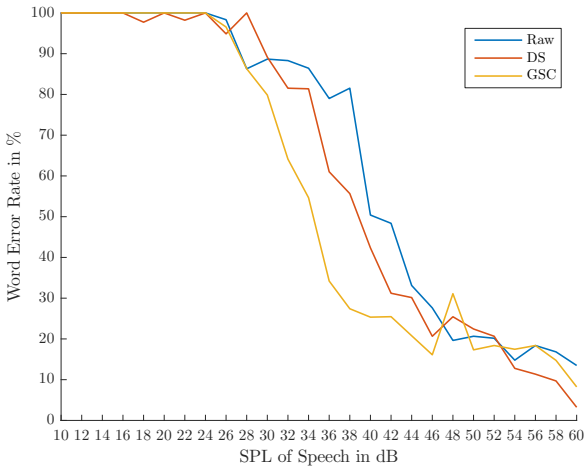


Figure 4.3: WER: Microphone: 65 dB SNR; Scenario: No Noise

It can be seen that up to 22 dBSPL the error rate is always around 100%. The D&S-BF is the first to get a WER beneath 100%. It can also be seen that overall the D&S-BF and GSC-BF perform better than the raw microphone which is not surprising due to the fact that the D&S-BF is used in both BFs. The D&S-BF algorithm has the ability of suppressing uncorrelated noise. In the no noise scenario the uncorrelated self-noise of the microphone is the only present noise.

4.2.1.2 Microphone SNR: 69 dBA

Figure 4.4 depicts the results of the 69 dBA SNR microphone in the no noise scenario.

It is shown that the WER decreases a few dBSPL earlier than at the 65dBA SNR microphones due to the lower self-noise of the 69dBA SNR microphone. The GSC-BF shows the best performance in this scenario especially in the low SPL region.

4.2.2 Babble Noise

Babble Noise is sometimes also called Coffee House Noise. This noise is generated when many people are talking in the same SPL.
In this scenario the SPL of the speech speaker is constant at 60 dB and the volume of the noise speakers is varied from 40 dBdB to 70 dB. Afterwards the recorded data is amplified by 20 dB to meet the requirements of the Goggle Assistant.

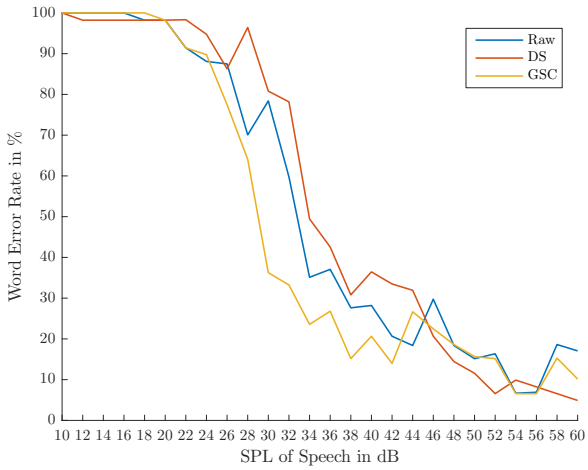


Figure 4.4: WER: Microphone: 69 dB SNR; Scenario: No Noise

4.2.2.1 Microphone SNR: 65 dBA

The results of the 65 dBAdBA SNR microphones captured in babble noise environment are displayed in Figure 4.5. The purple-doted line depicts the SPL level of the demanded sound source. All results which are lower than 60 dBdB mean that the noise volume is lower than the sound source. Results which are higher than 60 dB SPL are captured with a noise which is louder than the speech sound.
The babble noise has the same frequency components as clean voice. It can be seen that both beamformers do not respond well to this kind of noise. The sidelobe canceling path of the GSC-BF further reduces the noise canceling performance because it fails to adjust the adaptive filter coefficients perfectly. The performance of the GSC-BF is only a little bit better than from the raw microphone if the noise is louder than the 60 dBSPL of the demanded sound source.

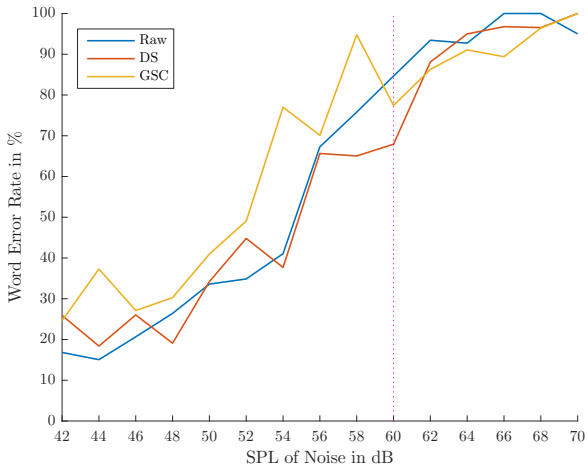


Figure 4.5: WER: Microphone: 65 dB SNR; Scenario: Babble Noise

4.2.2.2 Microphone SNR: 69 dBA

Figure 4.6 shows the results for data captured with 69 dB SNR microphones in a babble noise environment.

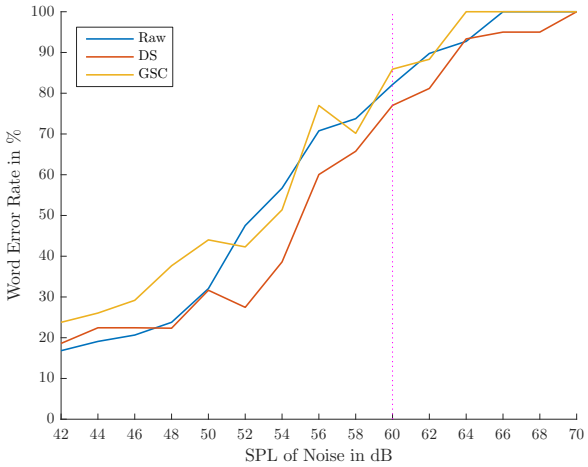


Figure 4.6: WER: Microphone: 69 dB SNR; Scenario: Babble Noise

It is shown that the D&S-BF has, especially in the case when the noise is louder than the sound source, a slightly better performance than both other capturing modes. It can be seen that the results from both microphones are quite similar. All three capturing modes yield almost the same WERs. Since this does not agree with the results from the other scenarios, it is assumed that the speech recognition algorithm of the Google Assistant is trained particular for this scenario. Babble noise is one of the most common noise for voice assistant devices. It may be possible that further babble noise reduction takes place within the cloud which corrupts the results as well.

4.2.3 IEC-268/Pink Noise

The IEC-268 is a pink noise with a crest factor of 6dB. Its noise power is higher in low frequency regions and low in high frequency regions [22].
In this scenario, the SPL of the speech speaker is constant at 60 dB and the volume of the noise speakers is varied from 40 dB to 70 dB. Afterwards the recorded data is amplified by 20 dB to meet the requirements of the Goggle Assistant.

4.2.3.1 Microphone SNR: 65 dBA

Figure 4.7 depicts the WER for different SPL of pink noise captured with 65 dBA SNR microphones.
Especially when the noise is louder than the sound source, the GSC-BF performs the

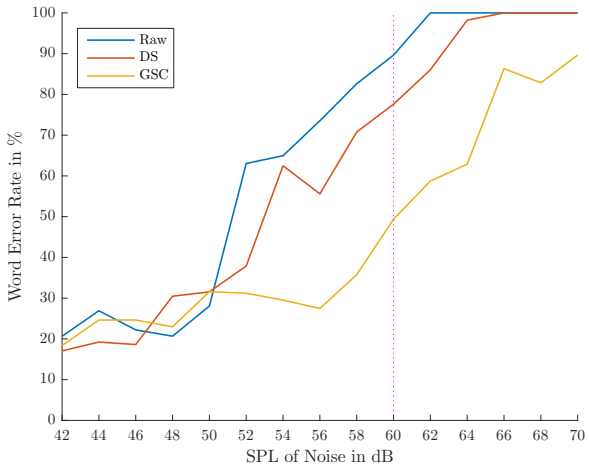


Figure 4.7: WER: Microphone: 65 dBA SNR; Scenario: IEC-268/Pink Noise

best as can be seen in Figure 4.7. In scenarios where the sound source is clearly louder than the noise all three BFs achieve nearly the same WER.

4.2.3.2 Microphone SNR: 69 dBA

Figure 4.8 depicts the resulting WER for data captured in pink noise environment with a 69 dBA SNR microphone.

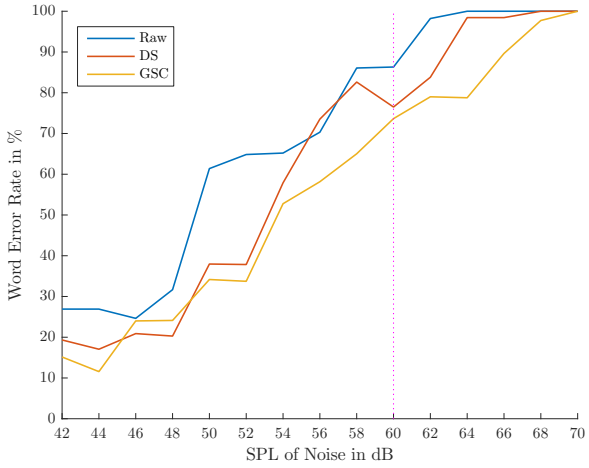


Figure 4.8: WER: Microphone: 69 dBA SNR; Scenario: IEC-268/Pink Noise

The GSC-BF, similar to the 65 dBA SNR microphones, archives also in this examination the best results. Especially in the case when noise is louder than voice the unprocessed raw microphone yields the worst WER results.

4.2.4 Comparison of 65 dBA and 69 dBA SNR Microphones

The Cerebro-MultiMicBoard can capture data with two types of microphones which can be differentiated by their Signal-To-Noise Ratio (SNR). The following results compare the 65 dBA and 69 dBA SNR microphones in different noise environments with each other.

4.2.4.1 No Noise Scenario

The following figure depicts the differences between the 65 dBA and 69 dBA microphone when no noise is present and the sound source SPL is varied. The captured data is processed with a GSC-BF.

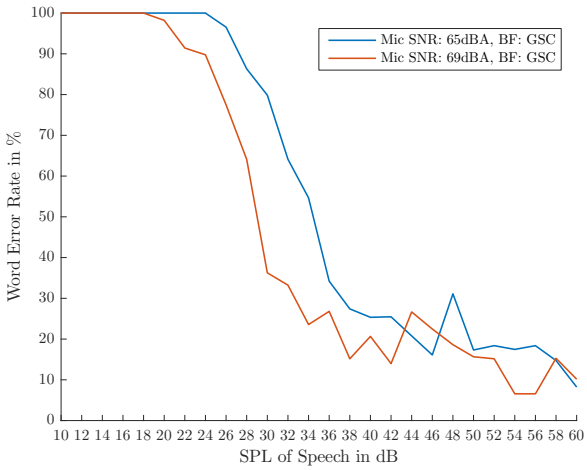


Figure 4.9: Comparison 65 dBA vs. 69 dBA in *No Noise* Scenario processed with GSC

It can be seen that when the sound source has a low SPL the 69 dBA microphones yield better WER than the 65 dBA microphones. This comes due to the lower self-noise of the 69 dBA SNR microphones. Therefore, silent commands are better recognizable for the voice assistant when recorded with the 69 dBA SNR microphones.

4.2.4.2 Babble Noise Scenario

Figure 4.10 compares the WER of the captured sentences when recorded with 65 dBA and 69 dBA SNR microphones in babble noise environment. The babble noise is varied from 40 dBSPL to 70 dBSPL. The sound source is fixed at a SPL of 60 dB. The captured data is processed with a D&S-BF.
It is clearly visible that in this scenario both WER curves are quiet similar. There are no noteworthy differences between 65 dBA and 69 dBA SNR microphones.

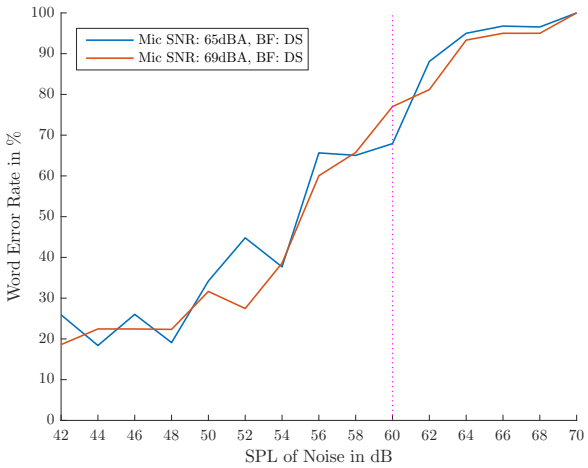


Figure 4.10: Comparison 65 dBA vs. 69 dBA in *Babble Noise* Scenario processed with DS

4.2.4.3 IEC-268/Pink Noise Scenario

In Figure 4.11 the WER curves of the 65 dBA and 69 dBA SNR microphones are displayed. The data was captured in an pink noise environment and is processed with a D&S-BF. The SPL of the noise is varied while holding the sound source on a constant level of 60 dBSPL.

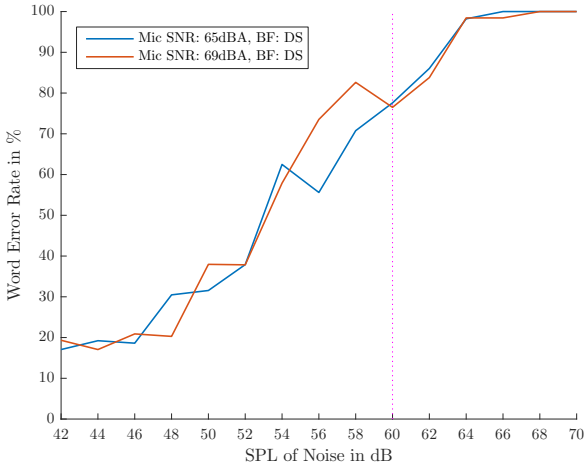


Figure 4.11: Comparison 65 dBA vs. 69 dBA in *Pink Noise* Scenario processed with DS

In the pink noise scenario the higher SNR microphones yield similar results as the 65 dBA SNR microphones. This result proofs the statement from the babble noise scenario that there are no considerable differences regarding WER-Analysis between 65 dBA and 69 dBA SNR microphones.

5 Summary & Conclusion

Within this thesis a typical signal processing chain for cloud-based voice assistant devices was developed. The focus thereby was on different beamforming algorithms and the trade-off between computational costs and noise source suppressing performance. The whole signal processing chain was first implemented using Matlab. After examining the computational costs of each beamforming algorithm, two realizable beamformers (D&S-BF and GSC-BF) where chosen to be implemented on an inexpensive low-power microcontroller.
As evaluation platform the Cerebro-MultiMicBoard, which was developed within this thesis, was used. It is equipped with two uniform circular microphone arrays. These two arrays hold seven microphones each which can be differentiated by their SNR. While one array uses 65 dBA SNR microphones, the other uses microphones with a SNR of 69 dBA.
To evaluate the noise suppressing performance of each beamformer, when using different microphones, the Google Assistant API was used to develop a Python-based voice assistant interface which sends the recorded audio streams to the Goggle Assistant. The transcript feature of the Goggle Assistant API was used to compare which words where recognized correctly which was measured as Word-Error-Rate (WER).
For the measurements eight non-identical sentences with 5-10 words from four different speakers from the TIMIT Corpus were used. Always one female and one male speaker from two different regions in the USA with preferable distinct pitches where chosen.
The measurement setup included one loudspeaker which was used for the main sound source and three loudspeaker in different positions around the DUT which were used as additional noise sources. The three different noise scenarios *No Noise*, *Babble Noise*, and *IEC-268/Pink Noise* where used within the measurements.

The evaluation has shown that using a beamformer definitely improves the WER. The two evaluated beamformer D&S-BF and GSC-BF showed in all scenarios better results than using just one single microphone without further signal processing. The GSC-BF, in particular, outperforms the other capturing modes in almost all scenarios. Only in the *Babble Noise* scenario it achieves no improvement to the other two modes. This may be because of the Google Assistant. It is assumed that, due to the fact that this kind of noise is the most common noise for voice assistants, Google trained its speech recognition algorithms in a way that this specific noise is further suppressed. Therefore, the results of all three capturing modes are almost the same.
The usage of microphones with a higher SNR improves the WER only under specific conditions. The results show that there are only noticeable improvements if the sound

source has a very low sound pressure level. The babble and pink noise evaluation confirmed this finding as the WER where almost the same for both microphone types.

The evaluation method used within this thesis was good but can be improved in future. It is not known what exactly happens with the data which is sent to the Google Assistant. E.g. the babble evaluation was eventually corrupted by some further noise suppression which is performed in the cloud.
For future evaluation alternatives such as dedicated speech-to-text software or a speech intelligibility measurement (ABC-MRT) (available as plug-in for the audio precision APx500) may be considered.

Finally, it results that a beamformer should be definitely considered when developing a signal processing chain for cloud based voice assistant devices. The Generalized-Sidelobe-Canceler Beamformer is an easy way to suppress noise coming from other directions than the sound source. It requires low computational costs as well as memory space. This kind of beamformer can be implemented on low-power microcontroller and helps to improve the WER of cloud-based voice assistant devices.

References

[1] J. Grudin, *A Moving Target—The Evolution of Human-Computer Interaction*. Taylor & Francis Group, January 2012.

[2] M. Turk, *Perceptual User Interfaces*, pp. 39–51. Springer London, 2001.

[3] Infineon Technologies AG, "Infineon brand portal - voice assistant." Accessed on 04 July 2018: https://www.infineon-brandportal.com/media-pool/asset/list?q=voice%20assistant.

[4] J. Benesty *et al.*, *Springer Handbook of Speech Processing*. Springer, 2008.

[5] I. J. Tashev, *Sound Capture and Processing: Practical Approaches*. John Wiley & Sons, 2009.

[6] H. Pessentheiner, "Beamforming using uniform circular arrays for distant speech recognition in reverberant environment and double-talk scenarios."

[7] W. Herbordt, *Sound Capture for Human/Machine Interfaces*. Springer, 2005.

[8] Infineon Technologies AG, *IM69D130 - High performance digital XENSIV MEMS microphone*, 2017.

[9] P. Schober, "Source localization with a small circular array."

[10] Y. H. J. Benesty, Jingdong Chen, *Microphone Array Signal Processing*. Springer, 2008.

[11] G. Doblinger, "An adaptive microphone array for optimum beamforming and noise reduction," in *2006 14th European Signal Processing Conference*, pp. 1–5, Sept 2006.

[12] L. Griffiths and C. Jim, "An alternative approach to linearly constrained adaptive beamforming," *IEEE Transactions on Antennas and Propagation*, vol. 30, pp. 27–34, Jan 1982.

[13] J. van de Sande, "Real-time beamforming and sound classification parameter generation in public environments."

[14] Inc. Computer Product Div. Analog Devices, *Adaptive Filters*, ch. 6. Adsp 21000 Family Application Handbook, Analog Devices Incorporated, 1994.

[15] O. Hoshuyama, A. Sugiyama, and A. Hirano, "A robust adaptive beamformer for microphone arrays with a blocking matrix using constrained adaptive filters," *IEEE Transactions on Signal Processing*, vol. 47, pp. 2677–2684, Oct 1999.

[16] W. Herbordt and W. Kellermann, "Efficient frequency-domain realization of robust generalized, sidelobe cancellers," in *2001 IEEE Fourth Workshop on Multimedia Signal Processing (Cat. No.01TH8564)*, pp. 377–382, 2001.

[17] S. Kirill, V. Ekaterina, and B. Simak, "Approach for energy-based voice detector with adaptive scaling factor," vol. 36, 11 2009.

[18] Google LLC, "Google assistant sdk - best practices for audio." Accessed on 9 March 2018: https://developers.google.com/assistant/sdk/guides/service/python/best-practices/audio.

[19] J. Yiu, *The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors*. Newnes, 2013.

[20] Infineon Technologies AG, *XMC4700 / XMC4800 Reference Manual*, 2016.

[21] D. Pearce, "Vectors, matrices and linear algebra - counting the costs," in *Getting Started with Communications Engineering*, ch. 2, 2007.

[22] J. Fehringer, "Measurement setup for voice recognition devices," 2017.

[23] E. Sengpiel, "Decibel table − spl − loudness comparison chart." Accessed on 1 July 2018: http://www.sengpielaudio.com/TableOfSoundPressureLevels.htm.

[24] J. S Garofolo, L. Lamel, W. M Fisher, J. Fiscus, D. S. Pallett, N. L. Dahlgren, and V. Zue, "Timit acoustic-phonetic continuous speech corpus," 11 1992.

Acronyms

µC . . . . . Microcontroller

ABM . . . . . Adaptive-Blocking-Matrix

ADC . . . . . Analog-Digital-Converter

ADD . . . . . Addition

AIC . . . . . Adaptive-Interference-Canceler

API . . . . . Application-Programming-Interface

ASIC . . . . . Application-Specific-Integrated-Circuit

BF . . . . . Beamformer

CCAF . . . . . Coefficient-Constrained-Adaptive-Filter

D&S-BF . . . . . Delay-And-Sum Beamformer

DIV . . . . . Division

DMA . . . . . Direct-Memory-Access

DOA . . . . . Direction-Of-Arrival

DSP . . . . . Digital-Signal-Processing

DUT . . . . . Device-Under-Test

FBF . . . . . Fixed Beamformer

FDAF . . . . . Frequency-Domain Adaptive Filters

FFT . . . . . Fast-Fourier-Transformation

FPU . . . . . Floating-Point-Unit

GCC . . . . . Generalized Cross-Correlation

GCC-PHAT . . . . . Generalized Cross-Correlation with Phase-Transform

GSC-BF . . . . . Generalized-Sidelobe-Canceler Beamformer

I2S . . . . . Inter-IC Sound

IC . . . . . Integrated Circuit

LMS . . . . . Least-Mean-Squares

MAC . . . . . Multiply-Accumulate

MEMS . . . . . Micro-Electro-Mechanical System

MISO . . . . . Multiple-Input-Single-Output

MSE . . . . . Mean-Squared-Error

MUL . . . . . Multiplication

MUX . . . . . Multiplexer

MVDR-BF . . . . . Minimum-Variance-Distortionless-Response Beamformer

NLMS . . . . . Normalized-Least-Mean-Squares

OLA . . . . . Overlap-and-Add

PCM . . . . . Pulse-Code-Modulated

PDM . . . . . Pulse-Density-Modulated

PHAT . . . . . Phase-Transformation

RAM . . . . . Random-Access-Memory

RGSC . . . . . Robust-Generalized-Sidelobe-Canceler

RMSE . . . . . Root-Mean-Squared-Error

SNR . . . . . Signal-To-Noise Ratio

SPL . . . . . Sound-Pressure-Level

SSL . . . . . Sound Source Localization

TDOA . . . . . Time-Difference-Of-Arrival

UCA . . . . . Uniform Circular Microphone Array

ULA . . . . . Uniform Linear Microphone Array

VAD . . . . . Voice Activity Detector

WER . . . . . Word-Error-Rate

XMC . . . . . Cross-Market Microcontroller from Infineon

List of Figures

List of Tables

List of Algorithms

Appendices

A Cerebro-Board Dimensions

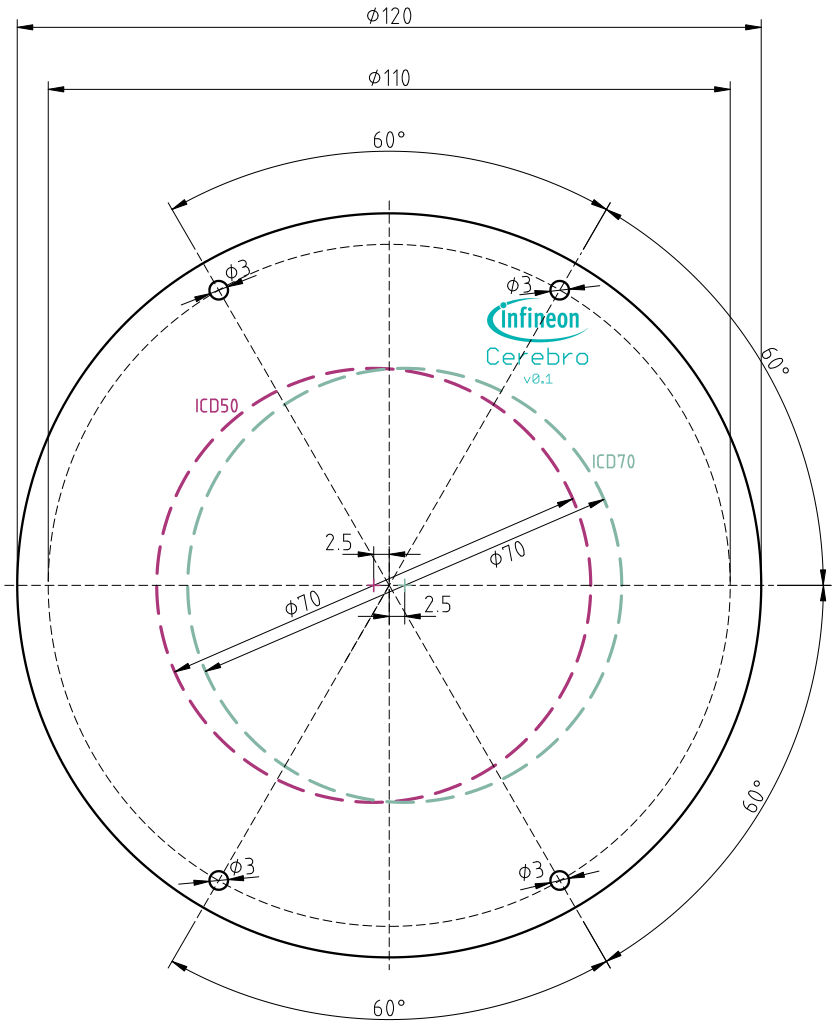


Figure A.1: Cerebro-MultiMicBoard Dimensions