Real-time implementation of non-linear signal-dependent acoustic beamforming

Oliver Merilaid

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 6.4.2016

Thesis supervisor:

Prof. Ville Pulkki

Thesis advisor:

M.Sc. (Tech.) Symeon Delikaris-Manias

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Oliver Merilaid

Title: Real-time implementation of non-linear signal-dependent acoustic beamforming

Date: 6.4.2016 Language: English Number of pages: 7+41

Department of Signal Processing and Acoustics

Professorship: Acoustics and Audio Signal Processing Code: S-89

Supervisor: Prof. Ville Pulkki

Advisor: M.Sc. (Tech.) Symeon Delikaris-Manias

A real-time acoustical beamforming system incorporating the cross pattern coherence (CroPaC) post filtering method is implemented in this thesis. The real-time implementation consists of a signal-independent beamformer that is used for spatial discrimination of a sound field. The signal of the beamformer is post filtered by modulating it with a parameter that is derived from the cross-spectrum of two directional microphone signals. The post filter is implemented to enhance performance of beamforming (increase in signal-to-noise ratio), because beamformers are not efficient in environments with high level of reverberation.

The post filtering method has been previously implemented in MATLAB for non-real-time use, and this system is the first real-time implementation of an acoustical beamforming system utilizing it. The implementation is programmed in the programming language C for the graphical signal processing program Max developed by Cycling '74. It utilizes a time-frequency domain processing, and the spherical Fourier transform for a decomposition of a sound field into spherical harmonic signals. The implementation can be used with microphone arrays with maximum of 32 microphone capsules, which are laid over rigid sphere with uniform or nearly-uniform arrangements. The real-time implementation can be utilized in many applications, which require algorithm to work in real-time, such as teleconferencing and acoustical cameras.

Keywords: Microphone array signal processing, beamforming, spherical harmonics, post filtering

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Oliver Merilaid

Työn nimi: Epälineaarisen signaaliriippuvan akustisen keilanmuodostajan reaaliaikaimplementaatio

Päivämäärä: 6.4.2016 Kieli: Englanti Sivumäärä: 7+41

Signaalinkäsittelyn ja akustiikan laitos

Professuuri: Akustiikka ja äänenkäsittely Koodi: S-89

Työn valvoja: Prof. Ville Pulkki

Työn ohjaaja: M.Sc. (tech.) Symeon Delikaris-Manias

Tässä diplomityössä implementoidaan reaaliaikainen akustinen keilanmuodostusjärjestelmä signaalien väliseen koherenssiin perustuvalla (CroPaC) jälkisuodatuksella. Reaaliaikaimplementaatio koostuu signaaliriippumattomasta keilanmuodostajasta, jota käytetään äänikentän spatiaaliseen suodatukseen. Keilanmuodostajan signaalia jälkisuodatetaan moduloimalla sitä parametrilla, joka johdetaan kahden suuntamikrofonin signaalin välisestä koherenssista. Jälkisuodatus implementoidaan keilanmuodostajan suorituskyvyn parantamiseksi (signaali-kohina-suhteen kasvu), sillä keilanmuodostajat eivät ole tehokkaita kaiuntaisissa ympäristöissä.
Jälkisuodatusmetodi on aikaisemmin implementoitu MATLABissa ei-reaaliaikakäyttöä varten. Tämän työn implementaatio on ensimmäinen reaaliaikainen akustinen keilanmuodostusjärjestelmä, joka hyödyntää CroPaC-jälkisuodatusta. Implementaatio on ohjelmoitu C-ohjelmointikielellä graafiselle signaalinprosessointityökalulle Max, jonka on kehittänyt Cycling '74. Prosessointi tapahtuu aika-taajuustasossa ja siinä hyödynnetään äänikentän dekompositiota palloharmonisiin signaaleihin. Implementaatiota voidaan käyttää mikrofoniryhmällä, jossa on korkeintaan 32 mikrofonikapselia, jotka on asetettu jäykän pallon päälle tasavälein tai lähes tasavälein. Reaaliaikaimplementaatiota voidaan hyödyntää lukuisissa sovelluksissa, jotka edellyttävät algoritmin reaaliaikaista toimintaa, esimerkiksi puhelinkokouksissa ja akustisissa kameroissa.

Avainsanat: Mikrofoniryhmäsignaalinkäsittely, keilanmuodostus, palloharmoniset, jälkisuodatus

Preface

I would like to thank professor Ville Pulkki for giving me the opportunity to do my master's thesis on this interesting subject, and my advisor Symeon Delikaris-Manias for all of his great help and patience. I would also like to thank all the AkuLab colleagues who have helped me on this topic, and all who I have had fun with playing foosball during my stay in the lab. Last but not least I would like to thank my family for all the support and encouragement over the course of my studies.

Otaniemi, 6.4.2016

Oliver Merilaid

Contents

Abbreviations

COLA	Constand Overlap-Add
CroPaC	Cross Pattern Coherence
CQMF	Complex-Quadrature Mirror Filterbank
DFT	Discrete Fourier Transform
DI	Directivity Index
DOA	Direction-Of-Arrival
ERB	Equivalent Rectangular Bandwidth
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
LCMV	Linearly Constrained Minimum Variance
MP3	MPEG-1 Audio layer 3
MVDR	Minimum Variance Distortionless Response
OLA	Overlap-Add
PQMF	Pseudo-Quadrature Mirror Filterbank
QMF	Quadrature Mirror Filterbank
SDK	Software Development Kit
SNR	Signal-to-Noise Ratio
STFT	Short-Time Fourier Transform
WNG	White Noise Gain

1 Introduction

Microphone arrays have been studied for several decades [1], but possibilities of using them in real-time applications have increased more recently. Computational processing power has increased enabling real-time processing of arrays with large amounts of sensors, and also low cost and small size provided by the development of the microphones arrays [2] has made them interesting. There are different types of arrays such as linear, planar, and more recently cylindrical and spherical microphone arrays.

A major application for microphone arrays is beamforming. It is a method that exploits spatial discrimination of sound sources in space by concentrating the directional beam on certain direction, while suppressing other sound sources coming from other directions. For example speech signal could be recorded with this method when a traffic noise is interfering from the other direction. By beamforming, the speech signal is enhanced while the traffic noise is attenuated.

Beamformers have originally been developed for telecommunication use, such as for radars, but later adapted to acoustics and further developed to deal with audio related phenomena such as reverberation and noise [1]. Beamformers can be used both for reception and transmission [3], but in this thesis only reception is discussed. They can be used for speech enhancement and for source localization applications [1]. For example, beamformers are used in hearing aids (speech enhancement) and in acoustical cameras (source localization).

However, the problem with beamformers in acoustics is that their performance suffers greatly in reverberant conditions and they do not suppress diffuse noise sufficiently [4]. Because of this, different multichannel post filters have been developed [5][6][7] to improve beamformer's performance in real acoustical conditions, with reverberation, and diffuse as well as incoherent noise [4].

The purpose of this master's thesis is to implement an efficient real-time acoustic beamformer system that incorporates state-of-the-art cross pattern coherence algorithm post filter. This algorithm has been previously implemented only in non-real-time and it has never been tested in real-time. In the implementation, an external is programmed in the programming language C for the graphical real-time digital signal processing program Max (version 7) developed by Cycling '74. Additionally Max graphical patcher is implemented for real-time and non-real-time, pre-recorded audio use of the external.

The external transforms multichannel microphone array signals to the time-frequency domain and they are further transformed to the spherical harmonic domain with the spherical Fourier transform. Signal-independent beamformers are calculated using spherical harmonic signals and a post filtering parameter is estimated using a cross-spectrum based measure between two directional microphone signals. Beamformer output is modulated with the post filtering parameter and the output is inverse transformed to the time domain.

Chapters 2–4 describe the theory part of this study. Chapter 2 describes time-frequency processing and different methods for time-frequency analysis, which are based on the Fourier transform and filterbank summation. Chapter 3 goes through

spatial encoding with the spherical harmonic framework, calculation of the spherical harmonic signals, and beamforming in the spherical harmonic domain. Post filtering with the cross pattern coherence algorithm is being discussed in Chapter 4. The implementation of the real-time acoustical beamformer system with the cross pattern coherence post filter is described step by step in Chapter 5. The thesis is concluded in Chapter 6.

2 Time-frequency processing

Time-frequency analysis is frequency analysis that is conducted in short time windows in audio applications. It is motivated by the fact that the auditory system of a human works alike – the inner ear of a human divides broadband acoustic stimuli into multiple narrowband neural signals. [8]

Time-frequency analysis is nowadays widely used in signal processing since it provides a way to learn about variations in frequency content of a signal with time. This is especially important with non-stationary signals such as audio signals. It provides also a way to alter the spectral energy of a desired frequency band of the signal much easier than time domain signal processing tools, such as finite impulse response (FIR) or infinite impulse response (IIR) filters. Time-frequency processing can be used in data compression, audio effects, and enhancement of audio quality [8]. For example, many perceptual audio signal coding techniques exploit information about time-frequency (for example frequency resolution and temporal resolution) hearing of humans to compress data bitrate, such as audio coding MPEG-1 Audio Layer 3 (MP3) [8].

Time-frequency analysis methods can be divided into two different approaches: the Fourier transform methods and filterbank summation methods. These two different approaches are discussed and compared next in Section 2.1 and Section 2.2. The differences in the methods in terms of aliasing and other artefacts are summarized in Table 1. The described methods are perfect reconstruction methods which means that the signals are reconstructed with no added aliasing or phase distortion [9] and near-perfect reconstruction methods which means the signals are reconstructed with negligible or inaudible aliasing components [8].

Table 1: Aliasing and other artefacts that are introduced to different Fourier transform based time-frequency transform methods and filterbank summation based time-frequency transform methods. It is assumed that the Fourier transform methods satisfy the COLA property and filterbank summation methods satisfy the prototype filter criterion. These criteria are defined later in this chapter.

		Artefacts	
Analysis method		Signal is altered	Signal is not altered
Fourier transform	OLA	wideband transients	none
	WOLA	modulation	none
	afSTFT	none	none
Filterbank summation	PQMF	negligible	none
	CQMF	none	none

2.1 Fourier transform methods

Most conventional way for time-frequency analysis is the Fourier transform based approach. Overlap-add (OLA) and weighted overlap-add (WOLA) methods are

widely used short-time Fourier transform (STFT) analysis methods. These methods are discussed next.

2.1.1 Short-time Fourier transform

OLA and WOLA STFT transforms are calculated as follows [8]:

$$X(k) = \sum_{n=0}^{N-1} w_a(n)x(n)e^{-i2\pi kn/N}, \tag{1}$$

where $w_a(n)$ is the analysis windowing function, $x(n)$ is the input signal, i is the imaginary unit, k is a wave number, n is a sample index, and N is a sample window length.

Inverse STFT transforms are calculated as follows [8]:

$$y(n) = \frac{1}{N} w_s(n) \sum_{k=0}^{N-1} X(k)e^{i2\pi kn/N}, \tag{2}$$

where $w_s(n)$ is the synthesis windowing function and $X(k)$ is the input signal. The difference between OLA and WOLA is that OLA does not use a synthesis window ($w_s(n) = 1$), whereas WOLA does.

In practice STFT is calculated by applying a fast Fourier transform (FFT) algorithm to a windowed frame [8], because FFT is much more efficient compared to the discrete Fourier transform (DFT).

2.1.2 Window functions

In order to retain perfect reconstruction, the constant-overlap-add constraint (COLA) (3) must be satisfied [10]. If it is not completely satisfied, aliasing and other artefacts will be introduced in the signal. Depending on the application, these artefacts may not be a problem. Such cases include for example spectrum analyzer [10], but when signal quality is essential, the COLA property must be satisfied. The COLA property [10]:

$$\sum_{N} w_a(n - NR)w_s(n - NR) = 1, \tag{3}$$

where R is the hop size. There are multiple windowing functions that satisfy the COLA property (3)[10], for OLA: rectangular window with no overlap (COLA(N)) and the Bartlett window with 50 % overlap COLA($N/2$) satisfies the COLA property [10]. Rectangular window with no overlap is the easiest solution for STFT transform and it can perfectly reconstruct the signal. However, with overlap better temporal resolution is obtained compared to no overlap. Better temporal resolution is needed if the signal is altered in the time-frequency domain to avoid aliasing in adjacent frequency bands. For WOLA, usually analysis and synthesis windows are chosen to be same ($w_a = w_s$). Therefore the easiest way to construct analysis and synthesis windows for WOLA STFT is to take square root of suitable non-negative OLA window that satisfies the COLA property, for example root-Hamming or root-Hann.

When using overlapping frames or zero-padding in the windowing, the transform is said to be oversampled [8]. [10]

The COLA property ensures perfect reconstruction only when the signal is not modified in the frequency domain. When the signal is modified in the frequency domain, inverse transform does not reconstruct the signal completely [8]. It happens because the signal might not be zero in the edges of a frame, which causes wideband transients in the output signal [8]. This can be avoided by using a synthesis window (WOLA) that truncates the outermost samples of the window to zero. However, with the truncation, part of the data is lost [8] and the modulation effects [11] could be introduced.

Because of the introduced artefacts when signals are altered, alias free STFT has been introduced [11] that eliminates circular effects completely with the cost of increased computational load. Circular effects are unwanted effects that occur because of circular convolution, for example above-mentioned non-zero samples in edges of the window. The main difference to conventional STFT method is that the time-frequency processing multipliers are first preprocessed before multiplication of transformed signal. First, multipliers are inverse transformed and delayed by half of the frame. Then, the resultant is windowed and zero-padded. Next, it is transformed back to the frequency domain and multiplied with the input signal that is transformed to the time-frequency domain with STFT transform method. Last, the resultant frequency domain signal is transformed back to the time-frequency domain. In this method, synthesis windowing is not necessary. This method suppresses circular convolution effects on altered signal completely when zero padded parts of the windowed frame are at least same length as signal parts. [8]

Although completely alias free STFT is possible, it is computationally more demanding than OLA and WOLA STFT methods. These methods however introduce artefacts in the signal if the signal is modified in the frequency domain, and cannot be used in applications where signal quality is important.

2.2 Filterbank summation methods

Filterbank based time-frequency analysis methods divide a broadband time domain signal into multiple narrowband frequency domain signals by using an array of band-pass filters [8]. Quadrature mirror filter (QMF) bank method is a widely used modulation based filterbank summation time-frequency analysis method. Among alias free STFT method, QMF methods are used when the signal is modified in the frequency domain, and high sound quality with no added artefacts is desired. The QMF methods are discussed next.

2.2.1 Quadrature mirror filterbank

Block diagram of a general K-channel quadrature mirror filter (QMF) bank is presented in Figure 1. Spectral resolution in this method depends on the amount of channels or filters in the filterbank. The bandwidths of the filterbank bands are

equal width and they are calculated as follows:

$$\frac{f_s}{2K}, \tag{4}$$

where K denotes the number of filters in the filterbank and f_s denotes the sampling frequency. First, in the analysis part, the signal is filtered with designed low-pass

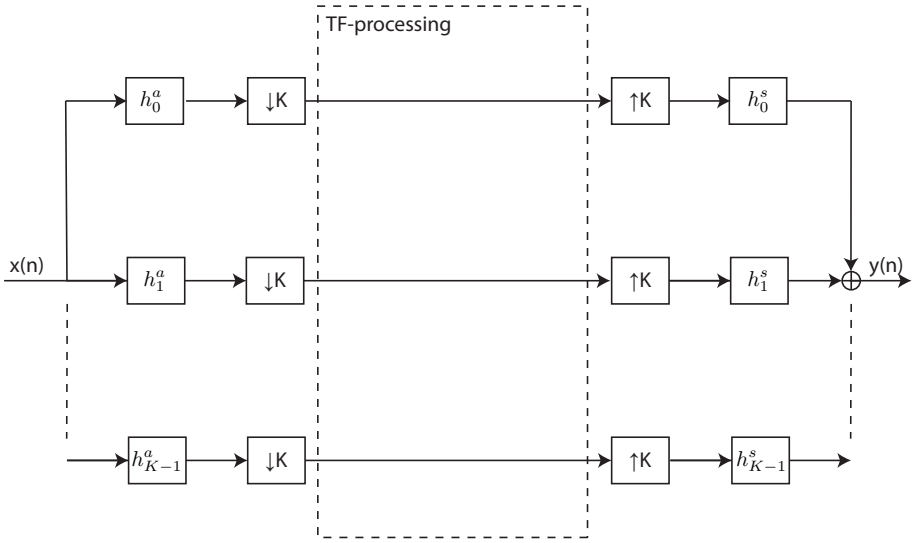

Figure 1: General K-channel QMF filterbank block diagram.

FIR prototype filter $h_p(n)$ that is modulated with a real (5) or complex (6) [8] modulation filter according to angular frequency ω to achieve band-pass filter for certain frequency band for analysis $h_k^a(n)$ and synthesis $h_k^s(n)$. Criterion for the prototype filter is that the energy of adjacent frequency bands has to be preserved and energy of non-adjacent frequency bands must be attenuated enough [12].

$$x_{\mathrm{modR}}(n) = x(n)\cos(\omega n) \tag{5}$$

$$x_{\mathrm{modC}}(n) = x(n)e^{i\omega n} \tag{6}$$

After the filtering, the signal is downsampled by a factor of K. Downsampling needs to be done to reduce complexity and redundancy of the processing. Because of this, QMF filterbanks are said to be critically sampled. However, downsampling and upsampling introduces aliasing components in adjacent frequency bands, but since they have opposite phases compared to each other, the alias components interfere in destructive manner and cancel each other. When using pseudo-quadrature mirror filterbank (PQMF), the modulation filters are real and therefore alias occurs if the signal is modified in the frequency domain. However, complex-quadrature mirror filterbank (CQMF) uses complex valued modulation filter coefficients to cancel out

this effect. Real valued filter coefficients produce negative frequencies to mirror on top of positive frequency, whereas complex valued filters do not map negative frequencies. Therefore they do not have aliasing in adjacent frequency bands although signals are modified in the frequency domain. Because of the complex representation, CQMF is oversampled by the factor of 2. The analysis filter responses for PQMF and CQMF filterbanks are denoted in the equations 7 and 8 respectively. [8]

$$h_k^a(n) = h_p(n) \cos\left[\frac{\pi}{2K}(2k+1)\left(n - \frac{N}{2} - \frac{K}{2}\right)\right] \tag{7}$$

$$h_k^a(n) = h_p(n) e^{\left[i\frac{\pi}{2K}(2k+1)\left(n-\frac{N}{2}-\frac{K}{2}\right)\right]} \tag{8}$$

In the synthesis part, the signal is first upsampled by the factor of K, essentially meaning addition of $K-1$ zeros in between every sample. Then channels are synthesized with synthesis filters $h_k^s(n)$ and summed together to reconstruct the signal. Non-adjacent frequency band alias components are filtered by the synthesis filter. Because of the adjacent frequency band aliasing with real valued filter coefficients, PQMF is called near-perfect reconstruction filterbank and CQMF perfect reconstruction filterbank. The synthesis filter responses for PQMF and CQMF filterbanks are denoted in the equations 9 and 10 respectively. [8]

$$h_k^s(n) = h_p(n) \cos\left[\frac{\pi}{2K}(2k+1)\left(n - \frac{N}{2} + \frac{K}{2}\right)\right] \tag{9}$$

$$h_k^s(n) = h_p(n) e^{\left[i\frac{\pi}{2K}(2k+1)\left(n-\frac{N}{2}+\frac{K}{2}\right)\right]} \tag{10}$$

2.2.2 Tree-structured quadrature mirror filterbank

Tree-structure [9] of a filterbank (sub-sub-band filtering [8]) can be used to obtain better spectral resolution in certain frequency bands. An example of general tree-structured QMF filterbank is presented in Figure 2, where the lowest two frequency bands are further filtered. In the Figure, sub-sub-bands are downsampled, but this is not necessary always, since the computational difference may not be significant depending on the application. Usual spectral resolution in QMF filterbanks is $K = 64$ [8]. The bandwidth of the frequency bands depends on K and sampling frequency f_s. Since QMF bands are equal bandwidth, this leads to bandwidth of 375 Hz with sampling frequency $f_s = 48$ kHz. This is more than enough for higher frequency bands, but for lower frequency bands it is not sufficient. The reason for this is that human hearing spectral resolution follows a logarithmic curve thus frequency resolution is much higher at lower frequencies than at high frequencies [8]. Because of this, tree-structured QMF filterbanks has bas used [13] in time-frequency processing of audio signals. Equivalent rectangular bandwidths (ERB) [8] and Bark scale [14] can be used to determine perceptually sufficient bandwidths for different frequency regions.

Although same result can be obtained with CQMF filterbank and alias free STFT transform method, major advantage over alias free STFT transform is that frequency bandwidths can be non-uniformly divided when using tree-structured QMF

filterbank, while STFT transform methods have always uniform frequency bands. Because of this, QMF filterbank methods are more efficient compared to alias free STFT transform methods when comparing perceptually sufficient time-frequency processing methods.

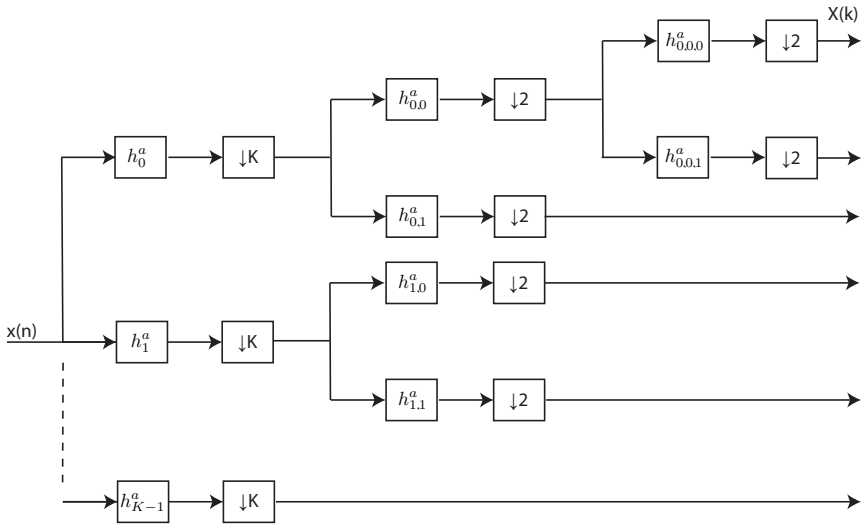

Figure 2: General tree-structured QMF filterbank block diagram.

3 Spatial encoding

In real acoustical environments, sound sources have three-dimensional spatial characteristics in addition to frequency content. These characteristics are for example the direction of the sound and the size of the sound source [15]. Sound field consist of direct sound coming from the direction of the sound source and reflections of it with different phases. These reflections or reverberation depend on the propagation path of the sound in the acoustical environment. For example, in ideal anechoic conditions, there is only the direct sound, but no reflections. In spatial encoding, it is desired to cover these spatial parameters. A microphone capsule captures sound over time samples, whereas a microphone array samples the sound field both spatially and temporally. Microphone array signals are encoded spatially with orthonormal plane-wave decomposition of sound field. This thesis focuses on spatial encoding in three dimensions.

Orthonormal plane-wave decomposition of a sound field is calculated with cylindrical harmonics in two-dimensional space and with spherical harmonics in three-dimensional space. For two-dimensional decomposition, circular microphone arrays are utilized, but in three-dimensional decomposition, a spherical microphone array is required. Spherical harmonics are used in acoustics for example in spatial sound recording, speech communication, sound field analysis, noise control, and entertainment [16][17]. In this thesis, the spherical harmonic framework is used for an acoustical beamforming system and later spherical harmonic signals are utilized in post filtering (Chapter 4). Spherical harmonics have been studied a lot in the last decades, and recently adapted to acoustics [17]. They have many advantages over the spatial domain calculation and linear array processing, for example spherical harmonics provide full three-dimensional rotation compared to linear array processing, and signal processing in the spherical harmonic domain is more efficient than processing in the spatial domain [17]. Section 3.1 defines required coordinate system in spatial encoding, Section 3.2 explains how the spherical harmonic signals are calculated, and Section 3.3 presents how different signal-independent beamformers are calculated in the spherical harmonic domain.

3.1 Spherical coordinate system

In order to understand spatial encoding in the spherical harmonic framework, a spherical coordinate system is described. The spherical coordinate system (r, φ, θ) is illustrated in Figure 3 on top of the Cartesian coordinate system (x, y, z). Spherical coordinates consist of a distance from the origin $r \in (\mathbb{R} \geq 0)$, an azimuth angle $\varphi \in [0, 2\pi)$ describing the angle in the horizontal plane, and an elevation angle $\theta \in [0, \pi]$ describing the angle in the vertical plane.

3.2 Calculation of spherical harmonic signals

Sound field can be decomposed as a sum of spherical harmonic signals. The spherical harmonics are presented in Figure 4 up to the order 4. The zeroth order spherical

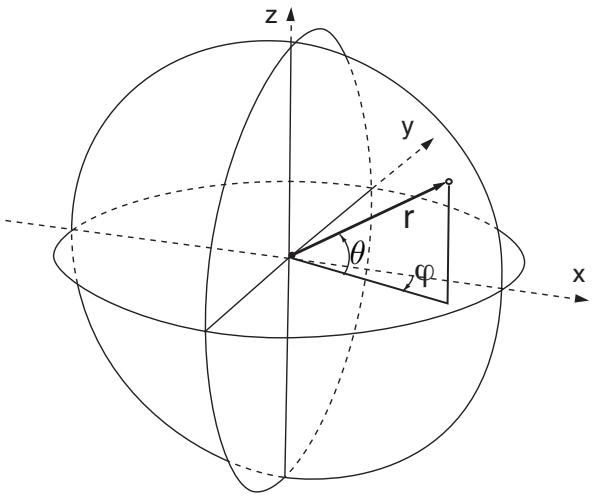

Figure 3: Spherical coordinate system. Adapted from [8].

harmonic present omnidirectional pattern, first order spherical harmonics present dipole patterns, and further orders present more complex directional patterns. In the Figure, yellow color on the patterns represents positive values and blue color negative values.

This section is divided into two separate steps: first, the theory of the spherical Fourier transform is presented, and then it is shown how the spherical harmonic signals are obtained by sampling the sphere and applying a plane wave decomposition. These signals can be calculated with the theoretical approach [17] or with the least squares approach [18]. The least squares approach is commonly used in antenna theory [7] and it has been adapted to acoustics [18]. The theoretical approach is useful since it does not require any preliminary measurements, while least squares approach does. However, measurement-based approach takes into consideration microphone capsule misalignment and capsule mismatch, and therefore has better performance compared to theoretical approach [18]. The theoretical approach is presented in this thesis.

3.2.1 Spherical harmonics

The spherical harmonic transform or the spherical Fourier transform is defined over two-dimensional sphere S^2 as follows [19]:

$$p_{lm}(k,r) = \int_{\Omega \in S^2} p(k,r,\Omega) Y_l^{m^*}(\Omega) d\Omega, \tag{11}$$

where $p(k,r,\Omega)$ is the sound pressure, k denotes a wave number, r and $\Omega = [\varphi, \theta]$ spherical coordinates, and $Y_l^{m^*}$ is the complex conjugate of the spherical harmonic Y_l^m. The inverse spherical Fourier transform is calculated as follows [19]:

$$p(k,r,\Omega) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} p_{lm}(k,r) Y_l^m(\Omega). \tag{12}$$

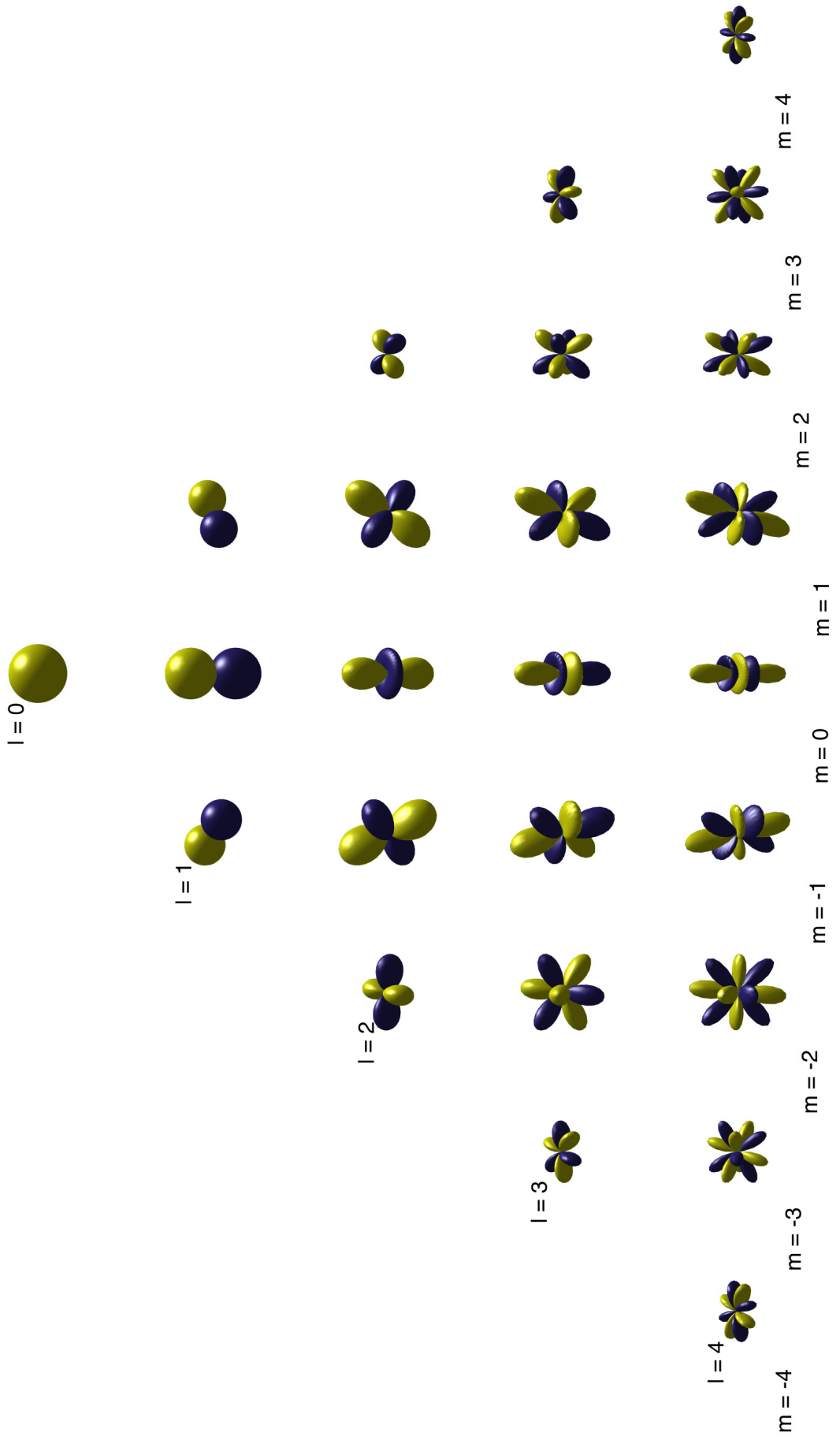

Figure 4: Spherical harmonics of orders 0 to 4.

Spherical harmonics are defined as follows [19]:

$$Y_l^m(\Omega) \equiv \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}} P_l^m(\cos\theta) e^{im\phi}, \tag{13}$$

where $l \in \mathbb{R}$ denotes order, $m \in \mathbb{Z}, |m| > l$ denotes mode, and $P_l^m(x)$ is associated Legendre polynomial function [17]:

$$P_l^m(x) = (-1)^m (1-x^2)^{m/2} \frac{d^m}{dx^m} P_l(x), \tag{14}$$

and $P_l(x)$ is Legendre polynomial function. It can be calculated with Rodriques' formula [17]:

$$P_l(x) = \frac{1}{2^l l!} \frac{d^l}{dx^l} (x^2-1)^l. \tag{15}$$

If equation 14 and equation 15 are combined, associated Legendre polynomial function for a degree of $m \geq 0$ can be presented as:

$$P_l^m(x) = \frac{(-1)^m}{2^l l!} (1-x^2)^{m/2} \frac{d^{l+m}}{dx^{l+m}} (x^2-1)^l, \tag{16}$$

and for a negative degree of m as [17]:

$$P_l^{-m}(x) = (-1)^m \frac{(l-m)!}{(l+m)!} P_l^m(x). \tag{17}$$

Sound pressure on a sphere p_{lm} caused by single plane-wave coming from the direction of (φ_0, θ_0) and amplitude $A_0(k)$ is calculated as follows [20]:

$$p_{lm}(k,r) = A_0(k) b_l(kr) Y_l^{m^*}(\varphi_0, \theta_0), \tag{18}$$

where $b_l(kr)$ is the effect of radial dependency and scattering. It is caused by sound field relation to the pressure on a sphere, and scattering is caused by sound field scattering from the microphone array [17]. The effect attenuates certain frequencies for different order spherical harmonic signals, and the effect needs to be equalized in order to have a flat frequency response. b_l for an open sphere with omnidirectional and cardioid microphone capsules, and for a rigid sphere with omnidirectional microphone capsules are defined as follows [16][17]:

$$b_l(kr) = \left\{ \begin{array}{ll} 4\pi i^l \left[j_l(kr) \right], & \text{open sphere, omni capsules} \\ 4\pi i^l \left[j_l(kr) - i j_l^{'}(kr) \right], & \text{open sphere, cardioid capsules} \\ 4\pi i^l \left[j_l(kr) - \frac{j_l^{'}(kr_a)}{h_l^{(2)'}(kr_a)} h_l^{(2)}(kr) \right], & \text{rigid sphere, omni capsules} \end{array} \right\}, \tag{19}$$

where $j_l(x)$ is a spherical Bessel function of the first kind, $h_l^{(2)}(x)$ is a spherical Hankel function of the second kind, and $'$ denotes a derivative. A spherical Bessel function of the first kind is calculated as follows [17]:

$$j_l(x) = (-1)^l x^l \left(\frac{1}{x} \frac{d}{dx} \right)^l \frac{\sin(x)}{x}. \tag{20}$$

A spherical Hankel function of the second kind is derived from a spherical Bessel function of the first kind and of the second kind as follows [17]:

$$h_l^{(2)}(x) = j_l(x) - iy_l(x), \tag{21}$$

where y_l is a spherical Bessel function of the second kind [17]:

$$y_l(x) = -(-1)^l x^l \left(\frac{1}{x}\frac{d}{dx}\right)^l \frac{\cos(x)}{x}. \tag{22}$$

If equations 20, 21, and 22 are combined, resultant of a spherical Hankel function of the second kind is [17]:

$$h_l^{(2)}(x) = i(-1)^l x^l \left(\frac{1}{x}\frac{d}{dx}\right)^l \frac{e^{-ix}}{x}. \tag{23}$$

The problem with equalization arises when $b_l(kr)$ has zero value in the audible frequency range. In the open sphere case, b_l has zeros regularly over the audible frequency range because of the zeros in the spherical Bessel function of the first kind (Figure 5) [18]. This can be avoided by using directive microphone capsules in the array or omnidirectional capsules on a rigid sphere to add a scattering effect. Although there are no zeros in these cases, the values are still relatively small in lower frequency regions [18]. Because of the zeros, if broadband frequency response is desired, only two latter array types are usable in practice.

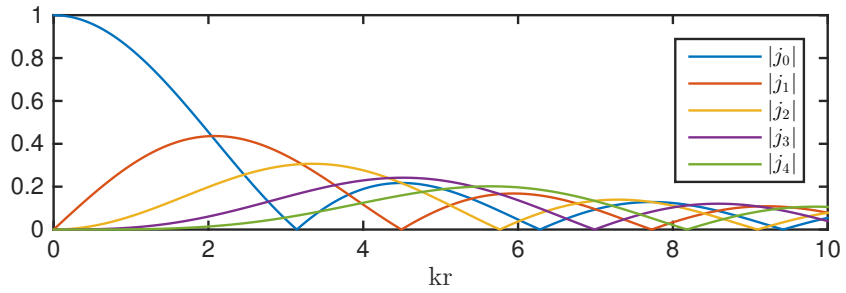

Figure 5: Absolute value of a spherical Bessel function of the first kind $|j_l|$ for orders $l \in [0, 4]$.

Because equalizing directly with $1/b_l(kr)$ leads to infinite amplification in frequency regions where $b_l(kr) = 0$, and in excessive noise amplification where $b_l(kr) \approx 0$, the inverse radial filter has to be regularized to avoid it. Regularization of the inverse radial filter can be done using for example the Tikhonov regularization method [21][18]:

$$\mathrm{EQ}_l(kr) = \frac{1}{b_{l,\mathrm{REG}}(kr)} = \frac{b_l^*(kr)}{|b_l(kr)|^2 + \beta^2}, \tag{24}$$

where $b_{l,\mathrm{REG}}(kr)$ denotes the regularized radial filter, $\mathrm{EQ}_l(kr)$ the regularized inverse radial filter, and β a regularization parameter. The regularization parameter can be

chosen so that it limits the amplification to certain level g in decibels, in a way that takes the signal-to-noise ratio (SNR) increase provided by the array into account [18]:

$$\beta^2 = \frac{1 - \sqrt{1 - \frac{10^{-g/10}}{Q}}}{1 + \sqrt{1 - \frac{10^{-g/10}}{Q}}}, \tag{25}$$

where Q denotes number of microphone capsules in the array. Regularized and non-regularized inverse filter coefficients are illustrated in Figure 6, where two different regularization parameters are presented for the Eigenmike [22]: allowed gain of $g = 0\,\text{dB}$ and $g = 20\,\text{dB}$. The Eigenmike is a rigid nearly-uniform spherical microphone array developed by mh acoustics LLC. These gain coefficients exceed allowed values by 15 dB because the Eigenmike has the array gain of $10 \cdot \log_{10}(32) \approx 15\,\text{dB}$.

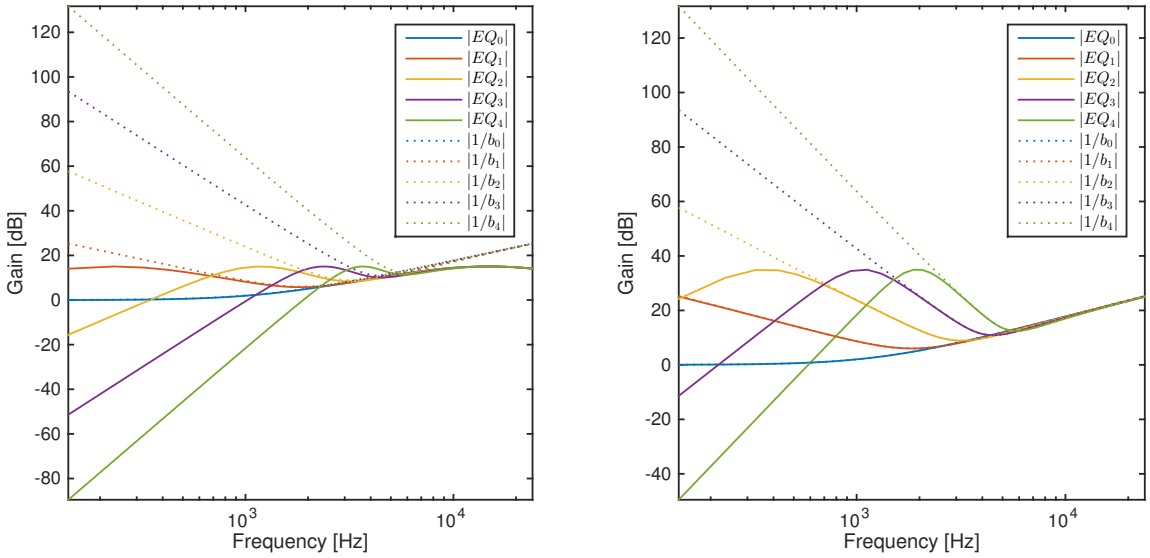

Figure 6: Absolute value of regularized inverse radial filter gain coefficients $|\text{EQ}_l|$ and non-regularized inverse filter coefficients $|1/b_l|$ for the Eigenmike [22] (rigid microphone array). Allowed gain g in the first figure is 0 dB and in the second figure 20 dB.

3.2.2 Sampling the sphere

Because the number of microphone capsules, or discrete sampling points is limited in practice, approximation of sound pressure function on the sphere is done. Different sampling methods have been proposed in the literature: placing the sensors in equal-angle arrangements, Gaussian arrangements, and (nearly-)uniform arrangements [17]. These samplings are illustrated in Figure 7. The sampling weights and the highest perfectly reconstructable spherical harmonic order depend on the array sampling method, as well as number of microphone capsules [17].

Figure 7: Spherical microphone array arrangements that can reconstruct spherical harmonic signals up to the order $L = 4$ illustrated: equal-angle (100 samples), Gaussian (50 samples), and nearly-uniform (36 samples) [23].

Order limited spherical harmonics signals can be reconstructed as follows:

$$p_{lm}(k, r) \approx \sum_{q=0}^{Q} \alpha_q p(k, r_q, \Omega_q) Y_l^{m^*}(\Omega_q), \tag{26}$$

where α_q is sampling weight and $p(k, r_q, \Omega_q)$ is the pressure of the microphone capsule q. When radial dependency and scattering is taken into account:

$$a_{lm}(k) \approx \mathrm{EQ}_l(kr) p_{lm}(k, r). \tag{27}$$

The sampling weights for equal-angle (EA), Gaussian (G), and (nearly-)uniform ((N)U) arrangements are [17]:

$$\alpha_q = \left\{ \begin{array}{ll} \frac{2\pi}{(L+1)^2} \sin(\theta_q) \sum_{\hat{q}=0}^{L} \frac{1}{2\hat{q}+1} \sin([2\hat{q}+1]\theta_q), & q \in [0, 2L+1], \quad \mathrm{EA} \\ \frac{\pi}{L+1} \frac{2(1-\cos^2\theta_q)}{(L+2)^2 P_{L+2}^2(\cos\theta_q)}, & q \in [0, L], \quad \mathrm{G} \\ \frac{4\pi}{Q}, & \quad (\mathrm{N})\mathrm{U} \end{array} \right\}, \tag{28}$$

When choosing the sampling arrangement, one criterion is to find a sampling method that can perfectly reconstruct spherical harmonic order L with the least number of sampling points Q. For equal-angle, Gaussian, and (nearly-)uniform arrangements, sampling criteria are [17][24]:

$$Q \geq \left\{ \begin{array}{ll} 4(L+1)^2, & \mathrm{EA} \\ 2(L+1)^2, & \mathrm{G} \\ \approx 1.5(L+1)^2, & (\mathrm{N})\mathrm{U} \end{array} \right\}, \tag{29}$$

where the uniform sampling can perfectly reconstruct the highest order spherical harmonic L with the least number of sampling points Q. The only way to construct uniformly distributed array is to lay sampling points over convex regular polyhedron or Platonic solid [17]:

$$L = \lfloor \tau/2 \rfloor, \tag{30}$$

where τ denotes τ-design of Platonic solid or regular convex polyhedron, and $\lfloor \cdot \rfloor$ is floor function. However, since there are only 5 different Platonic solids (tetrahedron, hexahedron, octahedron, dodecahedron, icosahedron), and the largest τ-design is 5, only spherical harmonic order up to $L = 2$ can be perfectly reconstructed with uniform sampling arrangement. Because of this, many nearly-uniform sampling arrangements have been developed that can reconstruct higher order spherical harmonics with negligible error [25]. For example τ-designs have been extended to $\tau = 12$ [26], therefore providing sampling method that can reconstruct spherical harmonics up to the order $L = 6$ with equal sampling weights. An example of different arrangements with different number of sample points that is needed to reconstruct equal order is illustrated in Figure 7. The sampling in (nearly-)uniform cases is always more efficient than in Gaussian sampling cases, but the lowest achievable Q is never under $(L+1)^2$ [26]. The lowest sampling criterion can be used with all arrangements, but if it does not satisfy the sampling criterion of the arrangement, spatial aliasing occurs [27]. When spatial aliasing occurs, higher order spherical harmonics are aliased to lower order spherical harmonics [17].

Because of the least number of sampling points for the order L is desired, near-uniform sampling methods are usually most suitable. However, (nearly-)uniform arrangements are much complex, and they are much harder to construct compared to other methods, but they have equal sampling weights. In addition, Gaussian and equal-angle sampling can utilize FFT in the calculation of spherical Fourier transform to improve calculation efficiency [25], since the elevation and azimuth coordinates are equally distributed in two-dimensional plane.

3.3 Beamforming in spherical harmonic domain

One essential application of spherical harmonics is spatial filtering. Spatial filters attenuate sound coming from the non-desired direction-of-arrival (DOA), and enhance sound coming from the desired DOA. Beamforming is a simple way to perform spatial filtering, and it is a method to separate signals interfering in the frequency domain when the signals do not interfere spatially [3]. Calculating the beamformer in the spherical harmonic domain offers several advantages: it is more efficient to calculate beamformer in the spherical domain compared to the spatial domain since there are usually more signals in the spatial domain compared to the spherical harmonic domain. Also, it is simpler to calculate beamformers in the spherical domain, because array configuration has to be always taken into account when beamformer is calculated in the spatial domain. [17]

There are two different types of beamformers: signal-independent and signal-dependent. In signal-independent beamforming, beamformer weights do not depend on the array data, whereas signal-dependent beamformer uses known (statistically optimum signal-dependent beamforming) or unknown (adaptive beamformer) second order statistics (for example auto-spectral density) from the array data to generate optimum beamformer weights, usually by steering nulls towards undesired DOA angles [3]. In this thesis, three different signal-independent beamformers were implemented:

- Regular beamformer [17],

- Minimum sidelobe beamformer (in-phase beamformer) [28], and
- Maximum directivity beamformer (Max r_e beamformer, beamformer that maximizes the energy in the look direction) [29].

The directional patterns of these beamformers are presented in Figure 8 where the differences in the lobes are illustrated. Signal-independent beamformers are computationally efficient and the beamformer weights can be calculated offline. The differences to regular beamformer is defined by the size of mainlobe and sidelobes: minimum sidelobe beamformer eliminates completely sidelobes, but increases the size of mainlobe, while maximum directivity beamformer maximizes the directivity of the beam, but does not decrease the size of sidelobes as much as minimum sidelobe beamformer [18]. The choice of beamformer is essentially a trade off between the size and gain of sidelobes' and maximum directivity of the main lobe. The right selection of the beamformer type depends highly on the application and sound field condition where it is used. The directivity of the beamformer's mainlobe can be made narrower by increasing the orders of spherical harmonic signals. However, increasing the orders increases the minimum sensor number and therefore more sensor channels need to be processed.

The beamformer output in the frequency domain is calculated as follows [17]:

$$y(k) = \sum_{l=0}^{L} \sum_{m=-l}^{l} w_{lm}^{*}(k) p_{lm}(k, r), \tag{31}$$

where $w_{lm}^{*}(k)$ denotes beamformer weights. Beamformer weights in axis-symmetric [30] case can be calculated as follows [17]:

$$w_{lm}^{*}(k) = \frac{d_l(k)}{b_l(kr)} Y_l^m(\theta_l, \phi_l), \tag{32}$$

where (θ_l, ϕ_l) denotes the look direction and $d_l(k)$ is real valued axis-symmetric weights. For example for regular beamformer, $d_l(k) = 1$ [17]:

$$y(\theta_l, \phi_l) = \sum_{l=0}^{L} \sum_{m=-l}^{l} \frac{p_{lm}(k, r_a)}{b_l(kr_a)} Y_l^m(\theta_l, \phi_l). \tag{33}$$

Minumum sidelobe beamformer weight calculation has been presented in [28], and maximum directivity beamformer weight calculation has been presented in [29].

There are two parameters that describe beamformer performance: directivity index (DI) [31], and white noise gain (WNG) [31]. These parameters are dependent on the choice of $d_l(k)$ [16]. DI gives the ratio between unity gain and overall average gain, which is given in decibels. Essentially it measures SNR increase provided by the beamformer. WNG denotes frequency dependent SNR improvement that array provides compared to single sensor input. [17]

Beamformer spatial discrimination performance depends on microphone array radius. If the sensors are arranged denser with a respect to a half wavelength, it is said that the array has a large spatial aperture [3]. If the array is small, low frequency

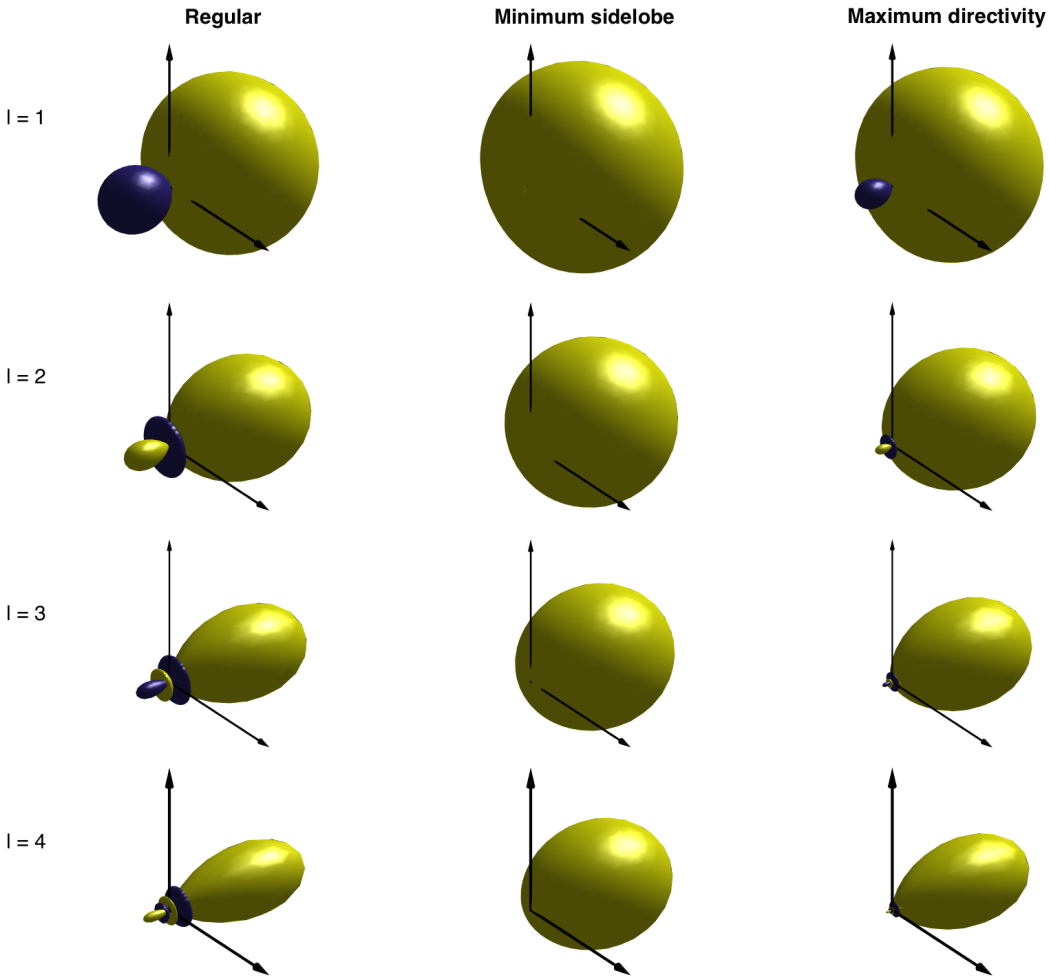

Figure 8: Different data independent beamformers for orders 1 to 4.

performance suffers. Secondly, if the array spacing is not dense enough compared to wavelength, spatial aliasing occurs. This effect occurs because spherical harmonics signals get deformed in higher frequencies. The spatial aliasing of the spherical array can be calculated with an approximated equation [18] as follows:

$$f_{al} = \frac{c}{2r\gamma}, \tag{34}$$

where c denotes the speed of sound, r denotes the radius, and γ the angle between two microphone capsules in the array.

4 Cross pattern coherence algorithm

Although beamforming might be sufficient in ideal cases when using narrow enough beamformers, reverberation and noise are present in real acoustical conditions. Acoustical beamformers do not perform properly in acoustically challenging situations where diffuse noise and reverberation are present [32]. The reverberation increases diffuseness of the sound field, therefore weakening the performance of the beamformer. Beamformer diffuse noise suppression depends on the number of microphone capsules, but coherence between sensors in low frequencies impairs the performance [32]. Because of this, different post filters have been developed to enhance dereverberation and noise attenuation of beamformer output [6][7], and they are also used for dereverberation and noise attenuation of omnidirectional microphone signal [5].

4.1 Post filtering

Post filtering methods are mostly based on the use of coherence-based measures on the microphone array signals, such as cross-spectral and auto-spectral densities [6]. Parameters derived from these measures are used to modulate either omnidirectional or beamformer signal [6]. Zelinski's method [5] assumes that noise is incoherent while speech signal is correlated between microphone signals. However, especially when using dense array configuration, noise correlation can be large between microphone channels at low frequencies [6]. McCowan proposed improved method [6] for Zelinski's method by using known noise field coherence model, for example spherically isotropic or cylindrically isotropic noise field model. Although these methods perform effectively in high frequency region, they suffer from poor performance at low frequencies [32].

Such a post filtering method overcoming poor performance on low frequency noise attenuation has been proposed [7] that also works with highly correlated noise. This method is called cross pattern coherence algorithm (CroPaC) [7] and it has proven to give better results for low frequency noise attenuation [7]. In addition, the method performs additional spatial filtering. This method is described next in Section 4.2.

4.2 Cross pattern coherence post filter

The block diagram of acoustic beamformer incorporating CroPaC post filter is presented in Figure 9. First, multi-channel time domain signal is transformed to the time-frequency domain with STFT transform or filterbank summation method. Second, microphone array signals are transformed to spherical harmonic signals with the spherical Fourier transform, and the effect of the microphone array is inverse filtered. Third, beamformer is calculated in the spherical harmonic framework, and cross-spectrum between two spherical harmonic signals is calculated. The cross-spectrum measure is normalized between $[0, 1]$ to obtain unity gain in the look direction. Since the measure introduces artefacts to the signal, the measure is interpolated and spectral floor is added. Fourth, beamformer is modulated with the obtained attenuation value, and in the end, the output signal is inverse transformed back to the time domain. Time-frequency transform, spherical harmonic signal

calculation, spherical harmonic beamforming, and inverse transform have been described in the previous chapters. This section is divided into five subsections: cross-spectrum calculation, spectral smoothing, spectral floor, synthesis, and performance.

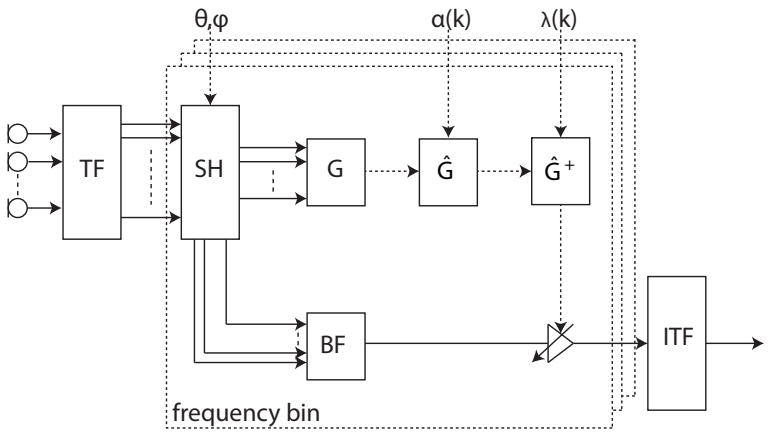

Figure 9: Block diagram of the beamforming system incorporating CroPaC post filter.

4.2.1 Cross-spectrum

The main principle of the CroPaC post filter lays on the cross-spectral density between two different order spherical harmonic signals that have the same look direction. Cross-spectral density is calculated as follows [7]:

$$\Phi_{b_{lm}b_{\hat{l}\hat{m}}}(k,t) = E\left[a^*_{lm}(k,t)a_{\hat{l}\hat{m}}(k,t)\right], \tag{35}$$

where $a^*_{lm}(k,t)$ is a complex conjugate of a spherical harmonic signal of order l and mode m in the time-frequency domain, and $a_{\hat{l}\hat{m}}(k,t)$ is correspondingly an another spherical harmonic signal with the same look direction, but of another order. In addition, k denotes a frequency band and t denotes a time index. The modulation gain is the real valued part of the cross-spectral density normalized by the auto-spectral densities of the spherical harmonic signals linked to used orders [7]:

$$\Phi_{b_{lm}}(k,i) = E[|(a_{lm})^2(k,t)|]. \tag{36}$$

The estimate is normalized in order to obtain unity gain in the look direction. For example if the used orders are 1 and 2, normalization is calculated using all three components of the order 1 ($m \in [-1,1]$) summed with all five components of the order 2 ($\hat{m} \in [-2,2]$). After normalization, the estimate is half-wave rectified since negative cross-correlation indicates that the sound is not coming from the look

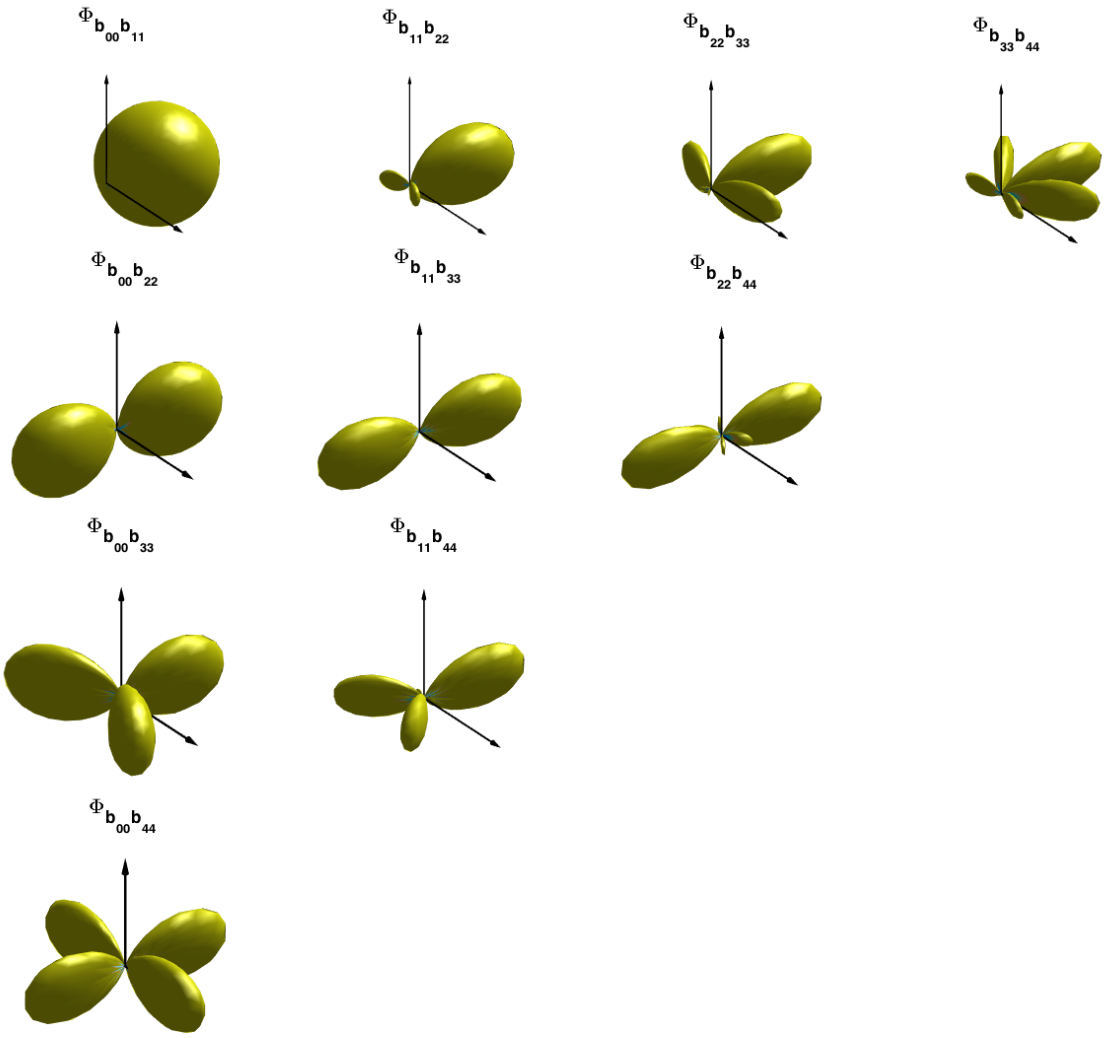

Figure 10: Resulting beampatterns from the multiplication of different real spherical harmonics. Only positive and real valued part of the multiplication is considered.

direction. Therefore, real part of normalized cross-spectrum is calculated as follows [7]:

$$G(k,t) = \max\left(0, \frac{C\Re[\Phi_{b_{lm}b_{\hat{l}\hat{m}}}(k,t)]}{\sum_{m}^{M,-M} \Phi_{b_{lm}} + \sum_{\hat{m}}^{\hat{M},-\hat{M}} \Phi_{b_{\hat{l}\hat{m}}}}\right), \tag{37}$$

where M denotes maximum mode of the spherical harmonic order, $\Re$ denotes real valued part of the value, and C is a normalization multiplier. Since the sum of energies in the denominator corresponds ideally to two omnidirectional signals, the nominator is multiplied by $C = 2$ to obtain unity gain in the look direction. However, this value depends on how the directional beampatterns have been derived, and in practice the multiplier value need to be calculated offline using theoretical spherical harmonics to get the exact normalization multiplier for the unity gain in the look

direction. In the practical case, also noise is present and should be taken into account. Resulting beampatterns from the multiplication of different order spherical harmonics are presented in Figure 10. The beampatterns illustrated use orders of 1 to 4 thus 10 different patterns are illustrated.

4.2.2 Spectral smoothing

Because this algorithm introduces large variation in the modulation gain G value, musical noise [33] is introduced to the signal. Musical noise is often described as a bubbly-like, distracting artefact that is introduced by large steps in amplitude [7]. Because of this, spectral smoothing or interpolation is used to mitigate the effect [7]:

$$\hat{G}(k,t) = (1-\alpha(k))G(k,t) + \alpha(k)\hat{G}(k,t-1), \tag{38}$$

where $\alpha(k) \in [0,1]$ is an interpolation value that can be frequency variant or invariant. Applying more interpolation to low frequencies is important because by listening the result, it can be noticed that low frequency musical noise is more audible than high frequency musical noise. Therefore, more smoothing is needed for lower frequencies than for higher frequencies. With less smoothing, better attenuation is achieved, but sound quality is usually degraded because of audible musical noise. The α value can also be calculated adaptively based on the SNR of the signal. This method however assumes that SNR can be estimated accurately [34]. In addition, α has to be truncated between $[0,1]$, because otherwise the interpolation would not be stable [34]:

$$\alpha(k,t) = \frac{1}{1+\xi^2(k,t)}, \tag{39}$$

where ξ is SNR. When SNR is large, α is near 0 and when SNR is low, α is near 1 [34].

4.2.3 Spectral floor

Since interpolation might not mitigate enough musical noise in challenging acoustical conditions, additional spectral floor is applied. These conditions compromise for example multiple simultaneous talkers and background noise. The spectral floor defines the minimum value for the modulation gain G. It can be added frequency dependently or independently as follows [7]:

$$\hat{G}^{+}(k,t) = \left\{ \begin{array}{ll} \hat{G}(k,t), & \text{if } \hat{G}(k,t) \geq \lambda(k) \\ \lambda(k), & \text{if } \hat{G}(k,t) < \lambda(k) \end{array} \right\}, \tag{40}$$

where $\lambda(k) \in [0,1]$ is the spectral floor value. When the spectral floor value is 0, there is no spectral floor and when the value is 1, beamformer output is not modulated at all with CroPaC gain value. Spectral floor is always a trade off between attenuation and sound quality [7]. If maximum attenuation is the main goal, spectral floor should not be used, but if maximum sound quality is desired, spectral floor should be used. If the goal is somewhere in between, optimal spectral floor value should be tuned carefully with frequency dependent values according to acoustical conditions.

4.2.4 Synthesis

Post-filtered signal is synthesized by modulating beamformer or omnidirectional output with CroPaC value as follows [7]:

$$Y(k,t) = \hat{G}^{+}(k,t)y(k,t), \tag{41}$$

where $y(k,t)$ is the beamformer or omnidirectional microphone signal. The beamformer can be any kind of beamformer, for example previously described signal-independent beamformer or signal-dependent beamformer such as minimum variance distortionless response (MVDR) beamformer. Best spatial resolution with negligible noise is achieved with cross-spectrum calculated from spherical harmonic signals of orders $l = L$ and $\hat{l} = L - 1$, where L is the maximum order that can be reconstructed with the microphone array [7]. For example, if a spherical array with maximum reconstructable order $L = 3$, best performance with inaudible noise is achieved by using the spherical harmonic signals order of $l = 3$ and $\hat{l} = 2$ looking at the same direction for cross-spectrum calculation.

4.2.5 Performance

CroPaC post filter output has been compared to McCowan post filter output in the first version of CroPaC post filter system [7]. In this version, omnidirectional output was modulated with CroPaC post filter calculated using least squares beamforming with cylindrical microphone array consisting of 8 microphone capsules [7]. In this experiment, CroPaC post filter outperformed McCowan post filter in most tests [7]. Several improvements have been proposed to the algorithm: in the different version of proposed CroPaC post filter [35], spherical harmonic framework with spherical microphone array consisting of 8 microphone capsules was used. In this version, MVDR beamformer output was modulated instead of omnidirectional output. Experiment was conducted for this version where MVDR beamformer output was compared to proposed algorithm using different reverberation values in simulated test cases [35]. For higher reverberation times improvement was lower compared to less reverberant acoustical conditions [35].

Taking into consideration of the above-mentioned test results, improvements on noise cancellation and robustness on reverberation is evident in the CroPaC post filter systems compared to other coherence-based post filtering methods. The performance can be further improved by using optimized microphone array geometry and by increasing the amount of microphone capsules in the array. Additionally, sidelobe positions and sidelobe gain of the beamformer affects the performance of CroPaC post filter system. [7]

5 Implementation

In this chapter, a real-time non-linear acoustic beamformer system based on the CroPaC algorithm is implemented. The algorithm is implemented in the programming language C as an external for graphical real-time signal processing programming language Max (version 7) developed by Cycling '74. Additionally a Max patcher is implemented for using a real stream of audio from the Eigenmike [22] and from other rigid spherical (nearly-)uniform microphone arrays or pre-recorded signals. By using Max and the standard libraries of C, the resulting implementation has multi operating system support and it works on Apple OS X and Microsoft Windows operating systems that support Max.

This implementation proves that it is possible to implement efficient acoustical beamforming system with the CroPaC post filter that can be used in real-time. This implementation can be used for the research of the CroPaC post filter system by easing tuning of the different parameters in real-time. Secondly, it is also possible to research functionality of different microphone arrays with this implementation. This kind of beamforming system could be utilized in various applications in the future where efficient spatial discrimination along with effective dereverberation and diffuse noise suppression is needed.

The implementation was tested informally in several acoustical cases in an anechoic chamber and a listening room with reverberation time RT_{60} about 500 ms. Five different sound samples were recorded with the Eigenmike [22] in the listening room for non-real-time demonstration of the implementation. All talkers were at 0 degree of elevation angle and divided with 90 degree in azimuth angle. These five samples were one talker, two talkers, three talkers, and two talkers with added low or high level diffuse white noise.

In Section 5.1, the external and the patcher are presented. In Section 5.2, the implementation decisions of the system are described step by step divided into five related sections: time-frequency processing, spherical harmonics calculation, beamforming, post filtering, and artefacts.

5.1 Max

Max is a graphical signal processing programming language meant for artists, educators, and researchers. It is used in the field of audio, visual media, and physical computing [36]. The program comes with a software development kit (SDK) that can be used to program third party externals in the programming language C. The selection criteria for the platform was mainly the ability to process signals in real-time, but there are also other platforms that fulfill the criterion such as Virtual Studio Technology plugins and Pure Data graphical signal processing software.

The block diagram of the implemented acoustic beamformer system is presented in Figure 11. The bottom side of the figure describes the external part of the implementation and the upper side describes the patcher part of the implementation. There are one multichannel audio input, keyboard or mouse and joystick input, and multichannel output. The patcher is used to control the external with multiple

parameters presented in the figure. These parameters are used for various adjustment and control of the algorithm.

5.1.1 Max external

Max is a modular programming language. These modules are connected graphically together with cords. The modules are called objects or externals and they have data inlets and data outlets. The external implemented in this thesis has 32-channel audio input and 3-channel output. Audio input channel number is 32, because external was implemented to be used with the Eigenmike [22], but also lower amount of input channels is supported. Output signals are omnidirectional signal output, beamformer signal output, and post filtered beamformer output. In addition, the external takes multiple control parameter values and arrays as an input. These control parameters are presented in Figure 11. The parameters consist of:

- microphone array coordinates,
- microphone capsule amount,
- spherical harmonics regularization method and limitation values,
- post filter version,
- frequency dependent interpolation values,
- frequency dependent spectral floor values,
- beamformer type, and
- beamformer order.

These parameters are discussed next.

Spherical harmonics calculation block

First set of parameters consists of the microphone array data. In order to calculate spherical harmonic signals for any microphone array, the microphone array data is needed. The data consists of microphone capsule coordinates in spherical coordinate system (r, θ, φ), where r denotes radius of the spherical microphone array in meters, $\theta \in [0, \pi]$ denotes the elevation array of microphone capsules in radians, and $\varphi \in [0, 2\pi)$ denotes the azimuth array of the microphone capsules in radians. In this implementation, it is assumed that the microphone capsule positions have the same radius as the array. The external supports up to 32-channel rigid spherical microphone arrays that have (nearly-)uniform arrangement, because equal weighting is used. However, less dense array can also be used, depending on how high order of spherical harmonics is needed. Minimum number of microphone capsules is calculated with:

$$Q \geq (L+1)^2, \tag{42}$$

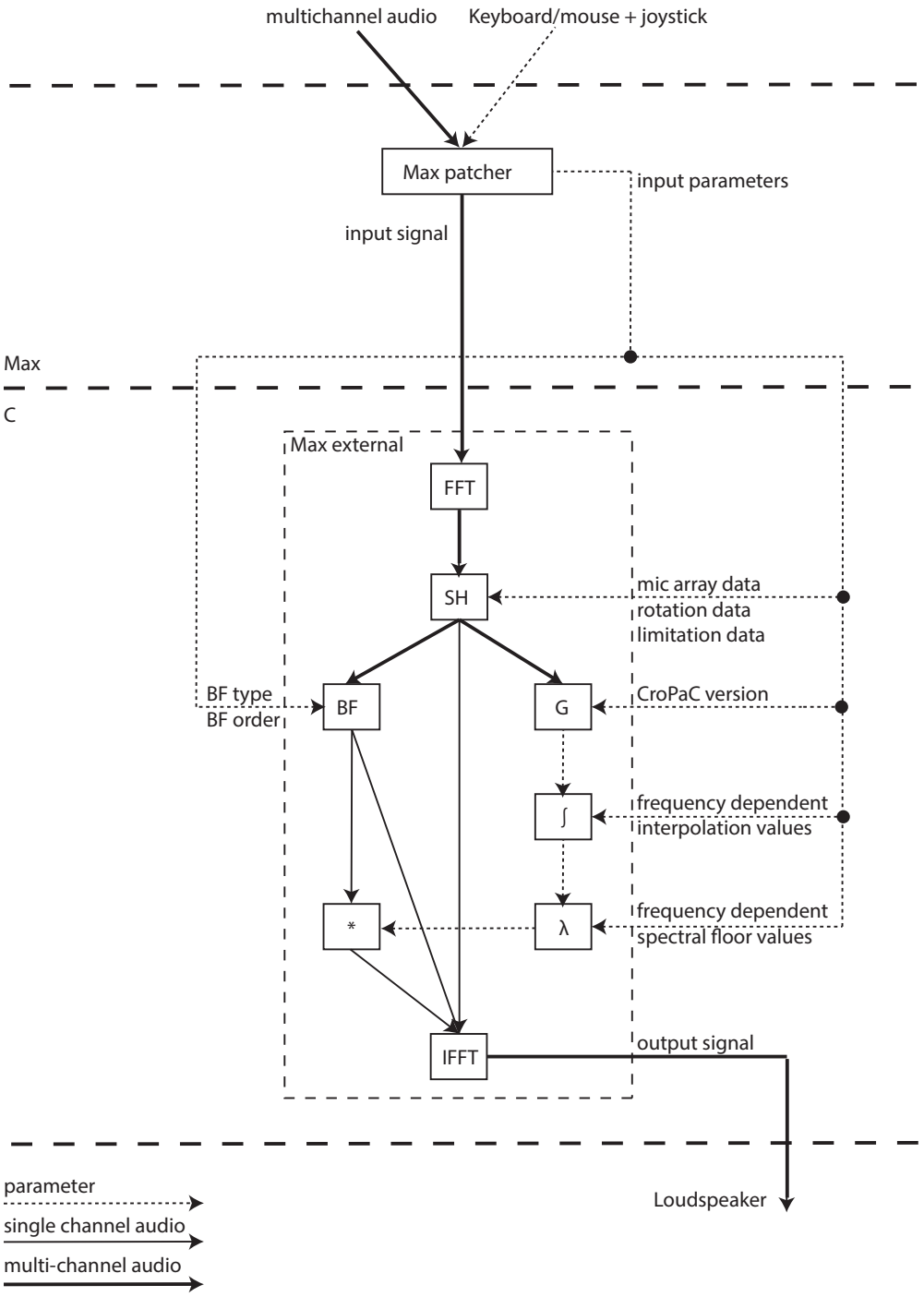

Figure 11: Block diagram of the implementation.

where Q denotes number of microphone capsules and L denotes order of the spherical harmonics. For this implementation, order of the spherical harmonics must be equal or greater than 1:

$$(1+1)^2 \leq 4. \tag{43}$$

However, depending on the array arrangement and allowed amount of spatial aliasing, the minimum capsule amount can also be higher. With more capsules than in (43), better performance is achieved.

Rotation data is needed to be able to rotate the beamformers and perform spatial filtering for different directions in space. The rotation of the beamformer is essential in real-time applications, because the beam can be rotated in full three dimensions in real-time. The external takes elevation value $\theta_{\mathrm{ROT}} \in [-90, 90]$ in degree and azimuth $\varphi_{\mathrm{ROT}} \in [-180, 180)$ value in degree as an input parameter.

Limitation data includes array that controls regularization parameters of the inverse radial filters as well as the regularization method number. The limitation data limits the self-noise amplification allowed for different order spherical harmonic signals in order to inverse the effect of the array to the sound field.

The parameters are sent to the external in the following way:

```
prepend radius int[1]
prepend micsCount int[1]
prepend micsAzimuth float[micsCount]
prepend micsElevation float[micsCount]
prepend rotation int[2]
prepend limit int[5]
prepend regMethod int[1]
```

Beamformer calculation block

In order to use different signal-independent beamformers, the external takes beamformer type and beamformer order as an input parameter. The external supports three different beamformer types: regular beamformer, minimum sidelobe beamformer, and maximum directivity beamformer. Since beamformers are formed by mixing spherical harmonics signals with real weights, the order of the beamformer can be modified from 1 to 4 depending on maximum spherical harmonic order supported by the array. For example, generally to use order 1 beamformer, at least minumum of 4 microphone capsules are needed, but for order 4, minimum of 25 capsules are needed (42).

The parameters are sent to the external in the following way:

```
prepend type int[1]
prepend order int[1]
```

Post filtering block

Since the system consists of many different CroPaC cross-spectrum calculation patterns, the external takes number of the cross-spectrum calculation pattern as

an input. The external supports 10 different CroPaC patterns based on the cross-spectrum between different spherical harmonic signals: $\Phi_{b_{lm}b_{\hat{l}\hat{m}}}$, where $l \in [0, 4]$ and $m \in [-l, l]$. Use of the higher order CroPaC patterns is limited by the number of microphone capsules available (42). For example with 4 capsules only $\Phi_{b_{00}b_{11}}$ can be used, but with at least 25 capsules, all 10 patterns can be used.

The parameters are sent to the external in the following way:

```
prepend CropacVersion int[1]
```

Interpolation and spectral floor calculation block

Post filtering usually results in the musical noise caused by too quick variation of the post filter value estimates that are used to modulate a beamformer signal. Because of this, frequency dependent interpolation or smoothing coefficients are needed. The external takes an array of interpolation values $\alpha \in [0, 1]$ as an input. However, when background noise and several talkers are present, interpolation might not decrease musical noise enough. In this case, external takes spectral floor value array $\lambda \in [0, 1]$ as an input.

The parameters are sent to the external in the following way:

```
prepend interpolation float[133]
prepend spectralFloor float[133]
```

5.1.2 Max patcher

The structure of the patcher is described in Figure 11 and a screenshot of the user interface of the patcher is presented in Figure 12. The user interface is controlled with a keyboard and a mouse or optionally with a joystick. Only initial setups are controlled by a mouse and a keyboard such as microphone array type, signal selection, gain in, the order of beamformer, the type of beamformer, type of the CroPaC pattern, regularization method, order dependent limitation values, frequency dependent interpolation and frequency dependent or frequency independent spectral floor. There is also preset selection, where all of these selections can be saved. By clicking on different presets, many of the above-mentioned input parameters can be changed at once. Some of the typical cases have been tuned in the real cases and these parameters are predetermined. The joystick is added for easier control and increased speed for constantly changed input parameters such as rotation. It is also utilized to switch between output signals (post filtered beamformer signal, beamformer signal, omnidirectional signal, microphone signal). Frequency independent spectral floor can be controlled through the joystick's lever and the system can be switched on or off. This kind of setup provides better user experience compared to fully keyboard and mouse provided control.

5.2 Implementation decisions

Many decisions on the implementation of acoustical beamforming system were made during the process, because of different reasons, such as computational load and time

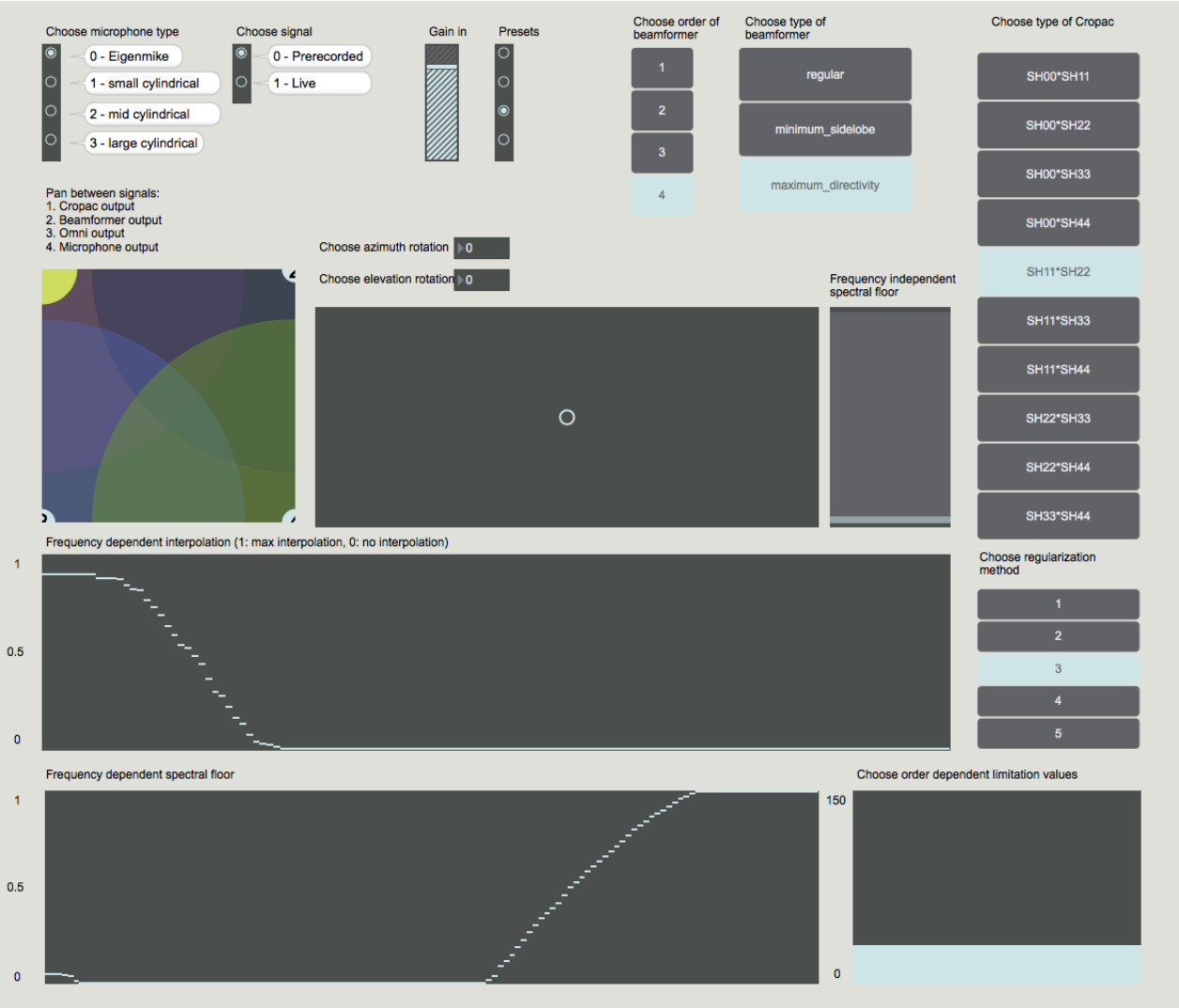

Figure 12: Max patcher for the implemented external.

related issues. These implementation decisions are discussed next.

5.2.1 Time-frequency transform

The implementation uses filterbank summation based time-frequency transform method. The type of the algorithm is 128-channel tree-structured PQMF filterbank. It is used because it provides better temporal resolution with equal spectral resolution compared to STFT transform methods. CQMF filterbank is by definition alias free, while Fourier transform based methods require an extra step in the calculation to be alias free. However, PQMF filterbank that is used here is not completely alias free, but the level of aliasing is inaudible [37]. PQMF filterbank is chosen over CQMF filterbank because it has real filter coefficients compared to complex coefficients, which makes it more efficient in comparison.

The block diagram of this frequency transform method is presented in Figure

13. The signal bands are divided in this implementation into 128 linearly uniform frequency bands. The bandwidth is based on bark scale critical bands, but because the bandwidth is 187.5 Hz with sampling frequency of 48 kHz, the spectral resolution is perceptually sufficient only above 1 kHz [14]. Because of this, the lower frequency bands of 187,5... 750 Hz are each further filtered into two separate frequency bands. According to bark scale, this is sufficient spectral resolution for the acoustical beamforming system implemented in this thesis.

The PQMF filterbank C library used here can be found from GitHub [37], and it can be used with the following commands:

```
1 afSTFTinit(&(h->afHandle), h->hopSize, h->inChannels, h->
    outChannels, h->LDmode, h->hybridMode);
2 afSTFTforward(h->afHandle, h->inTDtemp, h->inFD[i]);
3 afSTFTinverse(h->afHandle,h->outFD[i], h->inTDtemp);
4 afSTFTfree(h->afHandle);
```

The parameters used in the implementation are following: `hopSize` = 128, inChannels $\in [4, 32]$, `outChannels` = 3, `LDmode` = 0, and `hybridMode` = 1. `LDmode` denotes low delay mode and `hybridMode` denotes tree-structure in low frequency bands extending the total resolution to 133 frequency bands.

5.2.2 Spherical harmonics calculation

The spherical harmonic signal calculation in this implementation is based on the theoretical approach. It provides more convenient way for real-time calculation compared to the measurement based least squares approach where all the microphone arrays have to be measured first. However, the least squares approach may give more precise results, because real microphone arrays have imperfections such as sensor misalignment, sensor noise levels, and sensor directivity problems. This implementation supports spherical harmonics up to order $L = 4$ because it is maximum order spherical harmonic that can be reconstructed with the Eigenmike [22][27]. The code used to generate spherical harmonic weights is translated from the advisor's MATLAB code that is used for non-real-time calculation.

The code calculates spherical harmonic weights for rigid spherical arrays with (nearly-)uniform arrangement. The code is divided into 3 functions:

```
1 void Y_gen(double ***Y_large,double rot_az_deg, double
    rot_el_deg, float *azang, float *elang, int micsCount);
2 void complex2real(double ***Y_large, double **Y, int micsCount);
3 void calculateRadialDependency(double ***weightType, float *
    limiterLevel, float radius);
```

The first function is used to calculate the complex spherical harmonic weights for spherical rigid microphone array signals. This is also the function where rotation is implemented. Rotation is performed by rotating the microphone array capsule coordinates with rotation matrices as follows:

$$R_{ROT} = R_0(R(\chi)R(\theta)R(\varphi))^\intercal, \tag{44}$$

where (44), R_{ROT} denotes rotated microphone array in the Cartesian coordinates, R_0 denotes the initial, non-rotated microphone array in the Cartesian coordinates,

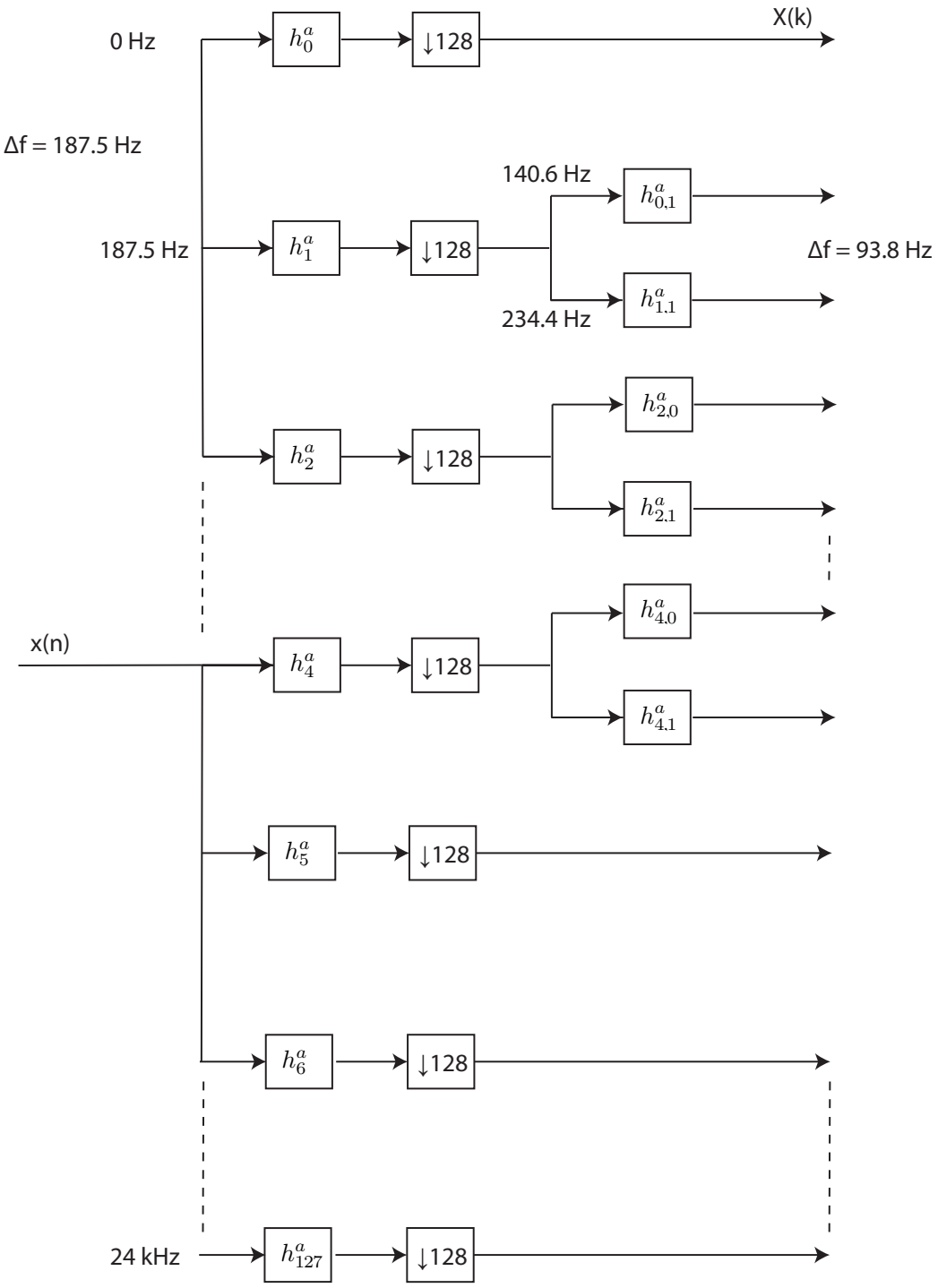

Figure 13: Block diagram of the PQMF filterbank type time-frequency transform used in the implementation.

and $R(\chi)$, $R(\theta)$, and $R(\varphi)$ denotes the z-y-z rotation matrices for the Euler angles. The initial microphone array (45) and the rotation matrices (46,47,48) are defined as follows:

$$R_0 = \begin{bmatrix} x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix} \tag{45}$$

$$R(\chi) = \begin{bmatrix} \cos(\chi) & \sin(\chi) & 0 \\ -\sin(\chi) & \cos(\chi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{46}$$

$$R(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\chi) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{47}$$

$$R(\varphi) = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{48}$$

where $\chi = 0$, θ is elevation rotation angle in radians, and φ is azimuth rotation angle in radians. With $\chi = 0$, $R(\chi) = \mathrm{diag}(\mathbf{I})$, so therefore equation 44 simplifies to:

$$R_0(R(\theta)R(\varphi))^\intercal. \tag{49}$$

This method of rotation is efficient and it is real valued. However, rotation errors may be produced that depend on the microphone distribution in the array [38].

The second function translates complex spherical harmonic weights to real form with the following relation:

$$Y_l^m = \left\{ \begin{array}{ll} (-1)^m\sqrt{2}\Im[Y_l^{|m|}], & m < 0 \\ Y_l^m, & m = 0 \\ (-1)^m\sqrt{2}\Re[Y_l^m], & m > 0 \end{array} \right\}, \tag{50}$$

where $\Re$ denotes the real valued part of the value, $\Im$ denotes the imaginary part of the value, $\sqrt{2}$ is the normalization multiplier, and $(-1)^m$ is the Condon-Shortley phase convention [39].

The third function calculates the inversion coefficients and regularizes them depending on the chosen regularization method and the regularization parameter. The regularization parameters are applied order dependently to the regularization method. The inversion coefficients are implemented for equalizing the array radial dependency and the scattering effect. Additionally several auxiliary functions are implemented since the standard C library does not include several functions that are present in MATLAB, such as factorial function, coordinate conversion function for converting coordinates from the Cartesian coordinates to the spherical coordinates

and back, the associated Legendre polynomial function, the spherical Bessel function of the first kind, and the spherical Hankel function of the second kind. There are libraries available for these calculations, but the libraries may not work in all operating systems. The standard library is used to maintain operating system independency. Also all the complex number calculations are calculated analytically and separated to real and imaginary part. The reason is that it was found out that the C standard library complex.h for complex number calculations is not efficient enough with the current implementation of the system in order to run in real-time.

Additionally, precalculated spherical harmonic weights were tried during the implementation process, but it was decided to calculate the weights in real-time to obtain more flexible system that can utilize different microphones and different input parameters without need to calculate the weights first offline. By calculating the weights offline, processing power may be conserved, but memory usage is increased. Precalculated weights could be considered when single microphone array and parameters are used or when less processing power is available.

5.2.3 Beamforming

It was decided to use signal-independent beamformers in the implementation. Signal-dependent beamformers provide more attenuation than signal-independent beamformers. However, since they were not implemented because the increase of needed processing power due to matrix calculations, it is unknown whether they will work in practical system. The mixing weights of three different beamformers were calculated offline in MATLAB and utilized in the implementation. This is convenient since the orders can be changed in according to used array configuration. An example implementation is provided below:

```
1 for(int time = 0; time < h->t; time++) {
2     for(int freq=0; freq < 133; freq++) {
3         h->outFD[time][0].re[freq] = (0.1410) * h->outFD[time][3].
              re[freq] + (0.2443) * h->outFD[time][6].re[freq];
4         h->outFD[time][0].im[freq] = (0.1410) * h->outFD[time][3].
              im[freq] + (0.2443) * h->outFD[time][6].im[freq];
5     }
6 }
```

This is the first order regular beamformer, where `h->outFD[time][0]` denotes beamformer signal, `h->outFD[time][3]` spherical harmonic signal of order $l = 0$, `h->outFD[time][6]` spherical harmonic signal of order $l = 1$ and mode $m = 1$ in each time index. `h->t` is the size of frame and `.re[freq]` denotes real valued part and `.im[freq]` imaginary part of each frequency bin.

5.2.4 Post filtering

Post filtering algorithm used in the implementation is the CroPaC algorithm. It is used since it has proven to have high performance in low frequency attenuation compared to other post filtering methods available [7]. An example of cross-spectrum is calculated as follows:

```
1  h->CS[time][freq]=2.3851 * (h->outFD[time][6].re[freq] * h->
       outFD[time][11].re[freq] + h->outFD[time][6].im[freq] * h->
       outFD[time][11].im[freq]);
```

where 6 is spherical harmonic signal of order $l = 1$ and mode $m = 1$, and 11 is spherical harmonic signal of order $l = 2$, and mode $m = 2$. Cross-spectrum is normalized with energies of spherical harmonic signals before modulation in order to obtain unity gain at look direction ($G \in [0, 1]$). This ensures that post filter would not affect the level of the signal. The spherical harmonic signal energies are calculated as follows:

```
1  h->E[time][sh][freq] = powf(h->outFD[time][sh].re[freq], 2) +
       powf(h->outFD[time][sh].im[freq], 2);
```

where sh denotes spherical harmonic signal. An example CroPaC gain value is then calculated as follows:

```
1  for(int time = 0; time < h->t; time++) {
2     for(int freq = 0; freq < 133; freq++) {
3        h->G_new[time][freq] = (h->C[time][freq]) / (h->E[time
             ][4][freq] + h->E[time][5][freq] + h->E[time][6][freq]
             + h->E[time][7][freq] + h->E[time][8][freq] + h->E[time
             ][9][freq] + h->E[time][10][freq] + h->E[time][11][freq
             ] + 1.0E-20);
4     }
5  }
```

where $4 \ldots 6$ denote the spherical harmonic signals of order $l = 1$ and mode $m = -1 \ldots 1$, $7 \ldots 11$ denote the spherical harmonic signals of order $l = 2$ and mode $m = -2 \ldots 2$, and 1.0E-20 is small value to prevent division by zero. This is important since division by zero is undefined action and the program might crash if it happens.

5.2.5 Artefacts

When beamformer signal is post filtered, musical noise is introduced because of quick fluctuation of modulation gain. Two ways were implemented to reduce musical noise enough to be inaudible. These parameters were chosen to be changed in real-time in order to help tuning. First, frequency dependent interpolation is added:

```
1  if(time != 0)
2     Gvalue = (h->interpolation[freq]) * (h->G_new[time - 1][freq
          ]) + (1.f - (h->interpolation[freq])) * (h->G_new[time][
          freq]);
3  else //if first index -> have to use previous frame's last value
        for interpolation
4     Gvalue = (h->interpolation[freq]) * (h->G_old[(h->t) - 1][
          freq]) + (1.f - (h->interpolation[freq])) * (h->G_new[time
          ][j]);
```

These values were decided to be manually tuned since SNR estimation is a computationally costly operation. After interpolation, spectral floor is added:

```
1  if(h->G_new[time][freq] < h->spectralFloor[freq])
2     h->G_new[time][freq] = h->spectralFloor[freq];
```

In the original design of the CroPaC post filter [7], interpolation is frequency dependent, but spectral floor is frequency independent. In this implementation, spectral floor was implemented also to support frequency dependent spectral floor since it was found out empirically that there is no need for spectral floor for all frequency bands. Also where spatial aliasing occurs, use of post filter should be avoided, that is to say, spectral floor should be raised to 1. When the Eigenmike [22] is used, spatial aliasing occurs approximately at 7.3 kHz according to equation 34. Because this implementation might attenuate low frequencies when using higher order CroPaC patterns, spectral coloration is introduced and spectral floor should be added to raise lower frequencies to obtain better sound quality.

6 Conclusions and future work

The real-time non-linear acoustic beamforming system was implemented in this thesis. The implementation was programmed in the C programming language for graphical signal processing program Max (version 7) developed by Cycling '74. The acoustic beamforming system works in the time-frequency domain with signal-independent beamforming and it incorporates state-of-the-art post filter based on cross pattern coherence (CroPaC [7]). The post filter improves spatial attenuation as well as low frequency denoising and dereverberation. The implementation is the first real-time implementation of a CroPaC post filter, and it facilitates research on spatial filtering, CroPaC post filtering, and parameter tuning. In addition, it allows the demonstration of the post filter capabilities in real-time. The implementation is flexible, enabling use of different spherical arrays with rigid uniform and near-uniform configuration. The starting point for the implementation was the Eigenmike [22], since it can be used for reconstructing high order spherical harmonic signals with relatively small amount of microphones. Because of this, maximum supported microphone capsule number is 32 and spherical harmonic order 4 [27].

During the implementation of the system, several implementation decisions were made. Max was chosen since it is widely used in the field of audio signal processing providing a well-documented SDK and programming language C is supported. A time-frequency transform was chosen to be made with PQMF filterbank since it provides efficient processing with inaudible aliasing. Compared to STFT methods, filterbanks provide non-uniform frequency band division. If perceptually sufficient STFT frequency bandwidth would be used, much more bands would be needed for processing while higher spectral resolution in high frequencies would not provide any additional information that would be beneficial. Complex QMF filterbank has no aliasing, but PQMF is more efficient. Spherical arrays and spherical harmonic framework was chosen over different configurations mainly since spherical configuration provides full three dimensional rotation of the directional beams which is not possible with other array types. Additionally spherical harmonics provide more flexible way for beamforming. Spherical harmonics were calculated with the theoretical approach instead of the least squares approach because it is not reasonable to measure all the microphone arrays beforehand, when the implementation supports any kind of array. However, the least squares approach may provide better result since the microphone arrays may have imperfections in capsules and their positioning. Beamformers were chosen as data independent since it provides a way to efficiently synthesize beamforming by mixing spherical harmonic signals with real weights. Since spherical harmonic beamforming is used, the array configuration does not have to be taken into account. Because of this, beamformer weights can be calculated offline beforehand to increase the efficiency of the system. The CroPaC post filter was chosen since it has been proven to have better low noise attenuation performance compared to other coherence-based post filtering methods.

In addition, user interface was programmed in Max for use of the external. Real-time audio stream from the array as well as prerecorded, non-real-time audio stream can be used in the user interface. It supports modifying all the control

and tuning parameters in real-time with keyboard or mouse. Additionally the most critical controls are implemented to support also use with joystick, for example the rotation of the directional beam.

The implementation was tested informally in several real acoustical conditions for verification. These test scenarios consisted of the department lab anechoic chamber and the listening room, with reverberation time RT_{60} about 500 ms. However, formal perceptual evaluations were not conducted in this thesis and for future, the system should be compared to previous CroPaC systems in [7], [34], and [35].

Although working efficient real-time system was implemented, several future improvement ideas emerged during the implementation process, most of which are calculation efficiency and attenuation performance related, such as optimization of the code and optimal beamforming:

- **Optimal beamformer** such as linearly constrained minimum variance (LCMV) beamfomer and especially the special case of LCMV, minimum variance distortionless response (MVDR) beamformer.
- **Optimization** of the code. The most critical code parts to be optimized are spherical harmonics calculation where all the weights are calculated again in each frame, which is not necessary if the parameters do not change. Additionally multiple thread calculation could be considered in order to share the workload on multiple cores.
- **Filterbank** with higher spectral resolution in low frequency region. This decreases temporal resolution because of increased frame size, but it might still improve the performance of the post filter on low frequencies. More complex tree-structures could be used to implement efficiently denser filterbank in low frequencies.
- **Least squares** approach functionality to spherical harmonics calculation to obtain better results with real arrays that might have imperfections such as capsule positional error or angle error.
- **Extend microphone array support.** For example variable radii support, open sphere with cardioid capsules support, and support for different shaped arrays such as cylindrical and hemispherical arrays with different sampling arrangements. Additionally support for any number of microphone capsules and any order spherical harmonic signals could be added.
- **Adaptive interpolation** that adjusts with the SNR of the signal with no need to manually find the best interpolation values for different situations [34].
- **Optimal regularization** for the inversion of radial dependency and scattering caused by the array. Research about optimal regularization parameters and method to obtain better low frequency performance when using higher order beampatterns in cross-spectral density calculation.

- **Camera** support for various applications, such as teleconferencing and security monitoring. Also possibility to use with acoustical cameras, increasing spatial resolution in low frequency bands compared to acoustical cameras used in the industry. For example 3-D camera could be used along with spherical microphone array.
- **Spatial aliasing** detection. Spectral floor could be increased automatically above spatial aliasing frequency bands.
- **Formal perceptual evaluation** with the current implementation should be conducted where this implementation is compared to previous CroPaC implementations in [7], [34], and [35].

References

[1] M. Brandstein and D. Ward, *Microphone arrays: signal processing techniques and applications*. Springer-Verlag, 2001.

[2] D. B. Ward, R. A. Kennedy, and R. C. Williamson, "Constant directivity beamforming," in *Microphone arrays: signal processing techniques and applications*, pp. 3–17, Springer-Verlag, 2001.

[3] B. D. Van Veen and K. M. Buckley, "Beamforming: A versatile approach to spatial filtering," *IEEE assp magazine*, vol. 5, no. 2, pp. 4–24, 1988.

[4] K. U. Simmer, J. Bitzer, and C. Marro, "Post-filtering techniques," in *Microphone arrays: signal processing techniques and applications*, pp. 39–60, Springer-Verlag, 2001.

[5] R. Zelinski, "A microphone array with adaptive post-filtering for noise reduction in reverberant rooms," in *International Conference on Acoustics, Speech, and Signal Processing*, pp. 2578–2581, IEEE, 1988.

[6] I. McCowan, H. Bourlard, *et al.*, "Microphone array post-filter based on noise field coherence," *IEEE Transactions on Speech and Audio Processing*, vol. 11, no. 6, pp. 709–716, 2003.

[7] S. Delikaris-Manias and V. Pulkki, "Cross pattern coherence algorithm for spatial filtering applications utilizing microphone arrays," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 11, pp. 2356–2367, 2013.

[8] V. Pulkki and M. Karjalainen, *Communication Acoustics: An Introduction to Speech, Audio and Psychoacoustics*. John Wiley & Sons, 2015.

[9] S. K. Mitra and Y. Kuo, *Digital signal processing: a computer-based approach*, vol. 2. McGraw-Hill New York, 2006.

[10] J. O. Smith, *Spectral audio signal processing*. W3K, 2011.

[11] E. Vickers, "Frequency-domain implementation of time-varying fir filters," in *Audio Engineering Society Convention 133*, Audio Engineering Society, 2012.

[12] C. D. Creusere and S. K. Mitra, "A simple method for designing high-quality prototype filters for m-band pseudo qmf banks," *IEEE Transactions on Signal Processing*, vol. 43, no. 4, pp. 1005–1007, 1995.

[13] J. Breebaart, S. van de Par, A. Kohlrausch, and E. Schuijers, "Parametric coding of stereo audio," *EURASIP Journal on Applied Signal Processing*, vol. 2005, pp. 1305–1322, 2005.

[14] E. Zwicker, G. Flottorp, and S. S. Stevens, "Critical band width in loudness summation," *The Journal of the Acoustical Society of America*, vol. 29, no. 5, pp. 548–557, 1957.

[15] M.-V. Laitinen *et al.*, "Techniques for versatile spatial-audio reproduction in time-frequency domain," 2014.

[16] B. Rafaely, Y. Peled, M. Agmon, D. Khaykin, and E. Fisher, "Spherical microphone array beamforming," in *Speech Processing in Modern Communication*, pp. 281–305, Springer-Verlag, 2010.

[17] B. Rafaely, *Fundamentals of Spherical Array Processing*, vol. 8. Springer-Verlag, 2015.

[18] S. Bertet, J. Daniel, and S. Moreau, "3d sound field recording with higher order ambisonics-objective measurements and validation of spherical microphone," in *Audio Engineering Society Convention 120*, Audio Engineering Society, 2006.

[19] E. G. Williams, *Fourier acoustics: sound radiation and nearfield acoustical holography*. Academic press, 1999.

[20] B. Rafaely, "Plane-wave decomposition of the sound field on a sphere by spherical convolution," *The Journal of the Acoustical Society of America*, vol. 116, no. 4, pp. 2149–2157, 2004.

[21] A. Tikhonov and V. Y. Arsenin, *Methods for solving ill-posed problems*. John Wiley and Sons, Inc, 1977.

[22] mh Acoustics LLC, "Eigenmike microphone - digital signal processing, acoustics and product design." http://www.mhacoustics.com/products [Online; accessed 22-January-2015].

[23] R. H. Hardin and N. J. Sloane, "des.3.36.8." http://neilsloane.com/sphdesigns/dim3/des.3.35.6.txt [Online; accessed 7-March-2016].

[24] D. Alon and B. Rafaely, "Efficient sampling for scanning spherical array," in *Second International Symposium on Ambisonics and Spherical Acoustics*, 2010.

[25] B. Rafaely, "Analysis and design of spherical microphone arrays," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 1, pp. 135–143, 2005.

[26] R. H. Hardin and N. J. Sloane, "Mclaren's improved snub cube and other new spherical designs in three dimensions," *Discrete & Computational Geometry*, vol. 15, no. 4, pp. 429–441, 1996.

[27] S. Brown and D. Sen, "Error analysis of spherical harmonic soundfield representations in terms of truncation and aliasing errors," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 360–364, IEEE, 2013.

[28] D. G. Malham, "Experience with large area 3-d ambisonic sound systems," in *Proceedings of the Institute of Acoustics*, Institute of Acoustics, 1992.

[29] J. Daniel, J.-B. Rault, and J.-D. Polack, "Ambisonics encoding of other audio formats for multiple listening conditions," in *Audio Engineering Society Convention 105*, Audio Engineering Society, 1998.

[30] J. Meyer and G. Elko, "A highly scalable spherical microphone array based on an orthonormal decomposition of the soundfield," in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. II–1781, IEEE, 2002.

[31] H. Van Trees, "Optimum array processing, ser. detection, estimation, and modulation theory (part iv)," 2002.

[32] J. Bitzer, K. U. Simmer, and K.-D. Kammeyer, "Multichannel noise reduction—algorithms and theoretical limits," in *9th European Signal Processing Conference*, pp. 1–4, IEEE, 1998.

[33] T. Esch and P. Vary, "Efficient musical noise suppression for speech enhancement system," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4409–4412, IEEE, 2009.

[34] S. Delikaris-Manias and V. Pulkki, "Parametric spatial filter utilizing dual beamformer and snr-based smoothing," in *Audio Engineering Society Conference: 55th International Conference: Spatial Audio*, Audio Engineering Society, 2014.

[35] S. Delikaris-Manias and V. Pulkki, "Cross spectral density based spatial filter employing maximum directivity beam patterns," in *The 5th International Conference on Information, Intelligence, Systems and Applications*, pp. 1–6, IEEE, 2014.

[36] Cycling '74, "About us and contact information." https://cycling74.com/company/ [Online; accessed 21-January-2016].

[37] J. Vilkamo, "Alias-free short-time fourier transform - a robust time-frequency transform for audio processing." https://github.com/jvilkamo/afSTFT [Online; accessed 10-September-2015].

[38] J. Atkins, "Robust beamforming and steering of arbitrary beam patterns using spherical arrays," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 237–240, IEEE, 2011.

[39] E. U. Condon and G. H. Shortley, *The theory of atomic spectra*. Cambridge University Press, 1951.