Masterarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Künstliche Intelligenz und Robotik

Localization of Traffic Objects using Acoustic Data

Amjad Saadeh
Matrikelnummer: 4554300
amjad.saadeh@fu-berlin.de

Eingereicht bei: Prof. Dr. Daniel Göhring
Zweitgutachter: Prof. Dr. Raúl Rojas

Berlin, 18.04.2016

Amjad Saadeh

Abstract

Hearing is a crucial sense for our daily life. We are communicating by sound, classify situations and even locate objects by their sound emissions.

A typical scenario for sound source localization is traffic. Pedestrians may 'hear' the direction of an incoming vehicle, even though they are watching the other way. Locating vehicles by sound is pretty important, so the U.S. Department of Transportation National Highway Traffic Safety Administration proposed a rule for minimum sound requirements for electric vehicles. Therefore, information provided by sound may be also valuable for drivers assistance systems and autonomous vehicles.

In this thesis, a method, which utilizes acoustic data, for the determination of the directions of traffic objects is introduced and analysed. This method bases on the Multiple Signal Classification (MUSIC) algorithm and should determine the direction of all types of traffic sound.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Berlin, 18.04.2016

Amjad Saadeh

Contents

1 Introduction

1.1 Motivation

Hearing is a crucial sense for our daily life. We are communicating by sound, use sounds to raise attention and to classify situations. Above all of that, sound is not only used to determine the type of an object, it is also pretty useful to locate it.
A typical scenario for *sound source localization (SSL)* is traffic. Every car emits sounds, e.g. from its engine or the friction between the tires and the road. Locating such sounds is especially important for pedestrians and bicyclists, but also for drivers. The importance of sound is shown by its absence. Electric and hybrid vehicles got a much lower noise level at low speed, so they are harder to locate. To prevent crashes, the *U.S. Department of Transportation National Highway Traffic Safety Administration* proposed in [1] a rule for minimum sound requirements for such vehicles.
Information provided by sound are also valuable for drivers assistance systems and autonomous vehicles. Modern infotainment systems and noise shielding influence the perception of traffic noise, making it more difficult to determine the position of other traffic participants that are not in sight. Knowing the sound and its *direction of arrival (DOA)* can be a useful assistance to classify data provided by other sensors, e.g. lidar point clouds.
At last, one information is accessible through sound much earlier than to other sensors: the presence a siren. Knowing its direction is crucial to initiate proper behaviour, like clearing the way.

1.2 Related Work

Improving the human ability to locate sound sources has been an objective since the early 20th century. During World War I 'acoustic radar', like the *Japanese war tuba*, was used to determine the direction of attacking Zeppelins or planes [2]. After the invention of radar, these kind of devices got obsolete.
SSL is widely used for naval applications. In the 1920s the first ancestors of *SONAR (SOund Navigation And Ranging)*, a technique to localize sound sources in water, were developed [3]. Originally, SONAR was used to detect and track hostile submarines, but also civil applications, e.g. mapping sea ground, became popular.
In our daily life, SSL is mostly set in closed rooms. Typical devices are equipment for distributed meetings. Zhan et al. implemented *maximum likelihood (ML)* SSL for such a system [4, 5]. By knowing the location of the sound source, the signal can be improved by focusing into the determined direction (*beamforming*).
In mobile robotics, sound is one way people and robots can communicate. Robots use SSL to turn towards speaking persons, to indicate that they

are paying attention. While the research on speech recognition was quite active in the early aughts, SSL fell behind [6]. So Valin et al. [6] developed an SSL system, consisting of eight microphone, mounted on a small robot. The developed algorithm determines the *time delay of arrival (TDOA)* of a sound by *generalized cross-correlation (GCC)*. Since then, some interesting efforts were made by Ishi et al., Li et al. and Hu et al. to build SSL into mobile robots [7, 8, 9].
In traffic, SSL is mostly used to determine sources of noise, to reduce noise emission like in [10]. For such applications, sound information is used to form an overlay on camera images. These images visualize the location of noise like in figure 1.1 (page 2). The *Deutsche Bahn* uses such *acoustic cameras* to determine ICE components [11] which emit most noise. In [12] a technique is shown which maps sound information onto 3D models.

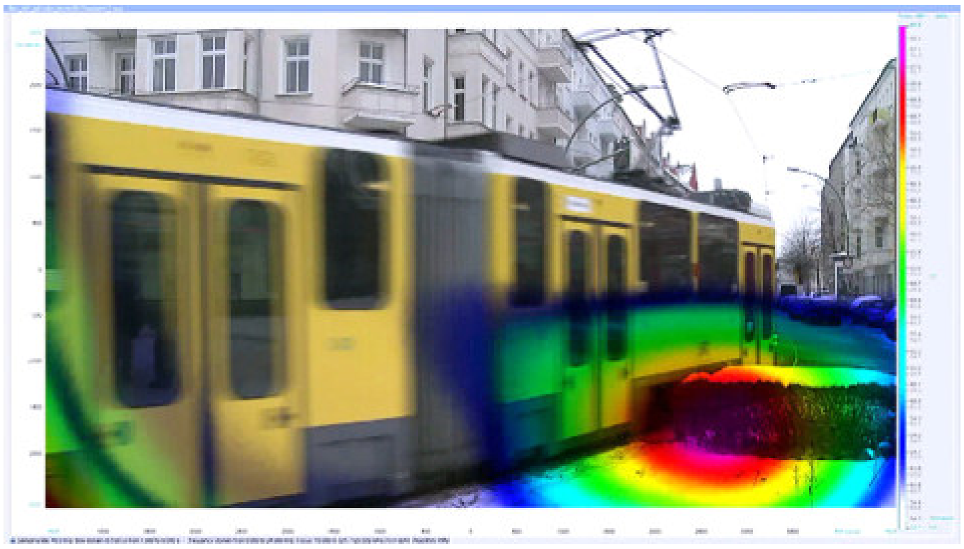


Figure 1.1: Acoustic image [13] - magenta/red regions indicate much noise, blue ones or regions without overlay are less active.

Classifying environmental sounds is widely investigated, e.g the detection of sirens [14, 15, 16, 17]. Fazenda et al. introduced in [18] a system that detects and determines the direction of sirens, utilizing GCC. Roseveare proposed an approach for tracking vehicles by their sound emission in [19]. He compared a *capon beamformer* to *multiple signal classification (MUSIC)* and extended the used beamforming algorithm to track wideband sources. In [20, 21], Marmaroli et al. used a GCC-method to track the axes of vehicles. With these information the wheelbase of the vehicles was estimated.
The starting point for my work is set by Goehring et al. in [22]. They used two *Kinects*, orthogonally placed to each other, and a simple cross-correlation approach to determine the direction of passing cars.

1.3 Structure of this Thesis

In this thesis, a method, which utilizes acoustic data, for the determination of the directions of traffic objects is introduced and analysed. While tracking sirens is the most interesting application, the proposed method should to track all types of traffic sound.
Section 2 (page 4) explains some fundamentals and additional methods needed throughout this thesis.
The basic problem is described in section 2.7 (page 22) and a short overview of some SSL techniques is given.
Afterwards, the MUSIC algorithm is introduced in section 3 (page 29). The derivation is shown and properties are mentioned.
Section 4 (page 37) presents some implementations of the MUSIC algorithm. Used tools and hardware are introduced and major pitfalls shown.
Experimental evaluation is done in section 5 (page 43). It starts with an evaluation of the microphone array of the Kinect and continues with manual analysis of traffic recordings. At the end of the section the MUSIC implementation will be evaluated with simulated and real world data.
In the end, a conclusion is given in section 6 (page 60). This section includes a discussion and some starting points for future work.

2 Theoretics

2.1 Notation

In this section, some mathematical notations are described.

2.1.1 Continuous and Discrete Functions

A continuous function is denoted with parentheses, i.e. $f : S \mapsto T$ with $S \in \{\mathbb{R}, \mathbb{C}\}$ and $T \in \{\mathbb{R}, \mathbb{C}, \mathbb{N}, \mathbb{Z}\}$ is denoted by $f(x)$.
Discrete functions or series are denoted by brackets, so $f : S \mapsto T$ with $S \in \{\mathbb{Z}, \mathbb{N}\}$ and $T \in \{\mathbb{R}, \mathbb{C}, \mathbb{N}, \mathbb{Z}\}$ is denoted by $f[x]$.

2.1.2 Complex Numbers

Given the polar coordinates (β, φ) of the complex number z, the *exponential form* is given by:

$$z = \beta e^{\varphi j}$$

with j as imaginary unit. The imaginary part of z denoted by $\Im(z)$, $\Re(z)$ is used to denote the real part.
The absolute value and argument of $z \in \mathbb{C}$ are denoted by the following formulas:

$$|z| = \beta \qquad \text{(absolute value)}$$
$$\arg(z) = \varphi \qquad \text{(argument)}$$

The *complex conjugate* is denoted by a superscript $*$ and is defined by:

$$z^* = \beta e^{-\varphi j}$$

An illustration of all parameters is given by figure 2.1 (page 5).
Scalars, no matter whether they are complex or real, are denoted by lower-case letters.

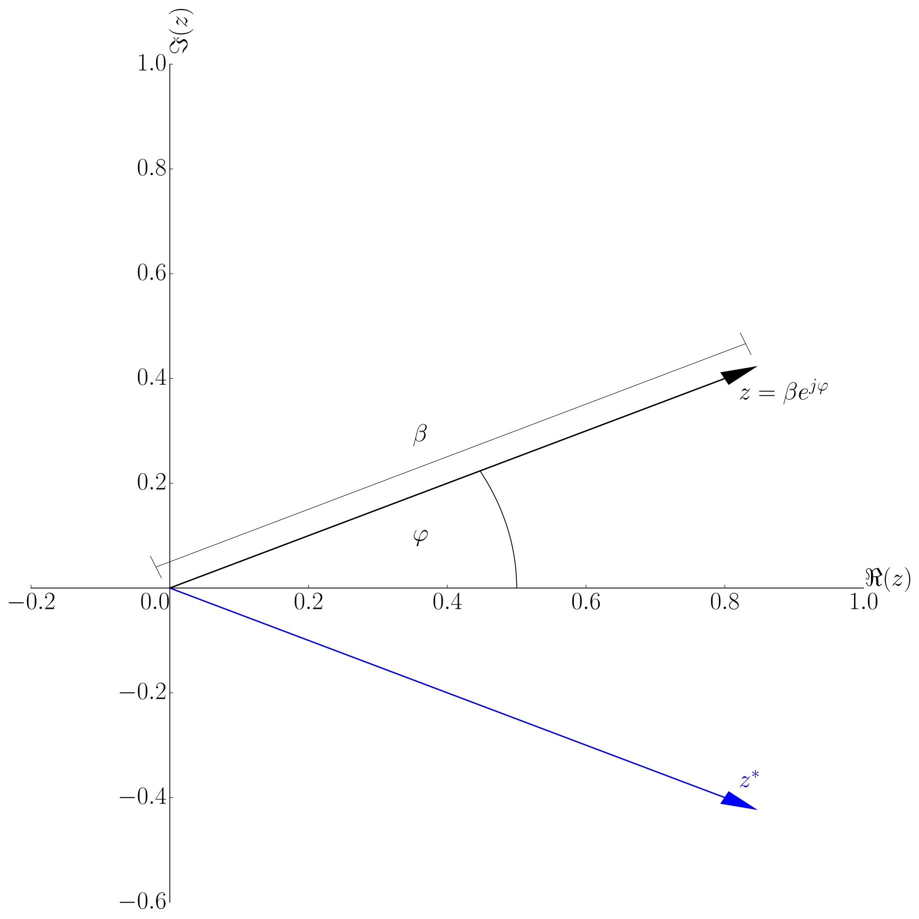


Figure 2.1: Illustration of a complex number in c plane.

2.1.3 Matrices and Vectors

Matrices are denoted by bold uppercase letters and can be conjugated, either:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

$$\mathbf{A}^* = \begin{pmatrix} a^*_{11} & a^*_{12} & \dots & a^*_{1n} \\ a^*_{21} & a^*_{22} & \dots & a^*_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a^*_{m1} & a^*_{m2} & \dots & a^*_{mn} \end{pmatrix} \qquad \text{(complex conjugate)}$$

The *Hermitian transpose* [23] is denoted by a superscript H and describes the transpose of a complex conjugate of a matrix:

$$\mathbf{A}^H = (\mathbf{A}^*)^T$$

Vectors are denoted by letters with an arrow on top, e.g. $\vec{a}$. I refer to them as column vector, i.e. they are $m \times 1$ matrices.

2.1.4 Convolution and Cross-correlation

The discrete convolution '$*$' of two discrete functions $f[x]$ and $h[x]$ is defined by [24]:

$$(f * h)[k] = \sum_{i=-\infty}^{\infty} f[i] \cdot h[k-i] \qquad k, m, n \in \mathbb{Z}$$

The cross-correlation '$\star$' of these functions is given by:

$$(f \star h)[k] = \sum_{i=-\infty}^{\infty} f[i]^* \cdot h[k+i] \qquad k, m, n \in \mathbb{Z}$$

The cross-correlation is related to the convolution by the following equation [25]:

$$(f \star h) = (f_-^* * h)$$

with $f_-[k] = f[-k]$.
For real computations proper boundaries for the sums have to be chosen.

2.2 Physical Foundation

What we sense as *sound* is the periodic change of pressure within a medium [26]. Sometimes this medium is water (e.g. SONAR) or a solid material like stone (e.g. in seismic applications), but in our daily life this medium is air. The periodic nature of sound is modelled by waves. A *sound source* - e.g. a sound box - changes the pressure with vibration, compressing the air in its direct neighbourhood. This change is propagated by kinetic interaction between particles and can be detected by a receiver - e.g. the human ear. Imagine a particle moving back and forth, exchanging kinetic energy with its neighbours. Such a wave is called *longitudinal wave*. In contrast to a *transversal wave*, which oscillates orthogonal to its direction of propagation, a longitudinal wave oscillates along the direction of its propagation. A comparison of both types of waves is shown in figure 2.2 (page 7).
Because of the kinetic interaction, the propagation speed c depends on the density of the medium. Therefore, speed of sound is higher in solid mediums than in gas because the particles are closer to each other and can exchange kinetic energy faster. The speed of sound in air at sea level at 20 °C is $c = 343\,\frac{\mathrm{m}}{\mathrm{s}}$ [28, 29].

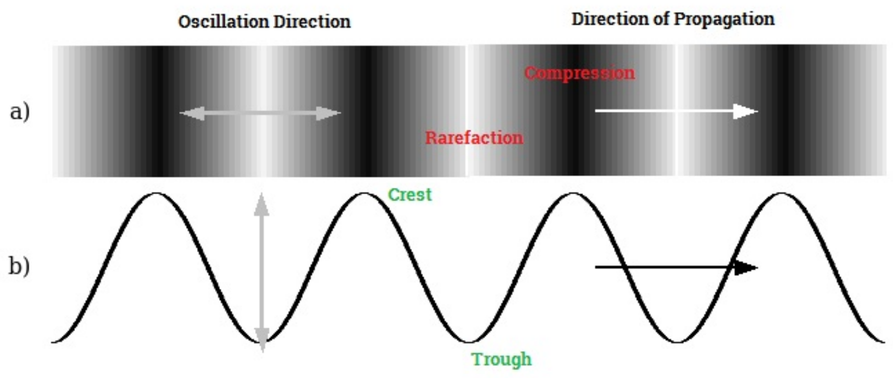


Figure 2.2: a) longitudinal wave, b) transversal wave - [27]

2.2.1 Mathematical Description

A wave is modeled by a *wave equation*, which depends on time and location. For my purpose only the time parameter is needed. A *pure tone* [30] is modelled by a sinusoidal wave:

$$s_r(t) = \beta \cos(-2\pi f t - \varphi) \tag{2.1}$$

The relation in equation 2.1 (page 7) describes the elongation of *signal* $s_r : \mathbb{R} \to \mathbb{R}$ over continuous time t in seconds. The natural frequency is denoted by f and given in Hz, φ is the phase shift in radians and the amplitude is β. In the following, the term 'frequency' is used as synonym for 'natural frequency'.
In some cases alternatives to f are useful:

$$\begin{aligned} \omega &= 2\pi f && \text{(angular frequency)} \\ T &= \frac{1}{f} && \text{(period)} \\ \lambda &= \frac{c}{f} = T \cdot c && \text{(wavelength)} \end{aligned} \tag{2.2}$$

with c as speed of sound.
An illustration of the variables of a wave function is given in figure 2.3 (page 8).
In many cases, equation 2.1 (page 7) makes things more complicated. Thanks to the *Euler's identity*, we can model a pure tone by complex numbers:

$$s_r(t) = \Re\left(\beta e^{j\cdot(-2\pi f + \varphi)}\right) \tag{2.3}$$

s_r is also called *real signal* because $s_r(t) \in \mathbb{R}$.

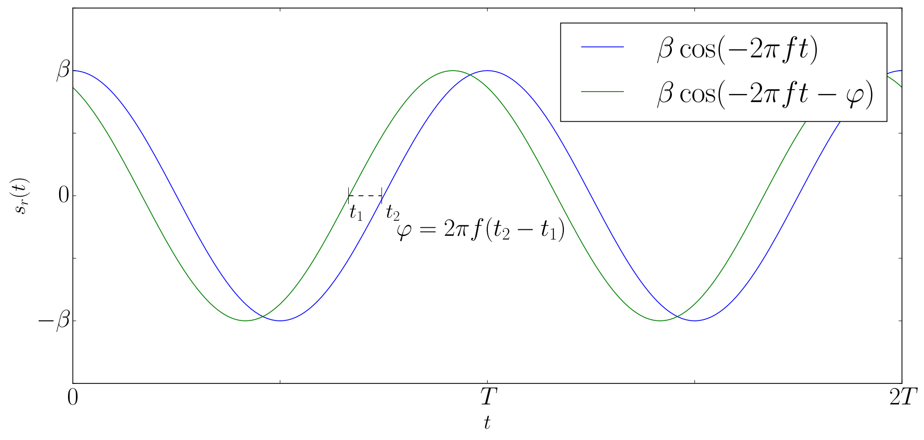


Figure 2.3: Illustration of basic variables of a wave.

A more handy notation omits the $\Re$:

$$\begin{aligned} s(t) &= \beta e^{j\cdot(-2\pi f t+\varphi)} \\ &= \beta e^{j\varphi} \cdot e^{-2\pi f t\cdot j} \\ &= a e^{-2\pi f t\cdot j} \end{aligned} \tag{2.4}$$

where s is the *complex signal* and a is called *complex amplitude*. The *magnitude* is given by $|a|$. Typically, s cannot be sampled directly, but it can be approximated by an *analytic signal* (section 2.5 (page 19)).
An alternative formulation of equation 2.3 (page 7) utilizes two complex sinusoids to form a real one [31]:

$$s_r(t) = \frac{a}{2} e^{-2\pi f t\cdot j} + \frac{a^*}{2} e^{2\pi f t\cdot j} \tag{2.5}$$

This equation will be important in section 2.4 (page 16).
The negative sign in the exponent is convention. Imagine equation 2.4 (page 8) as a vector in the c plane (see figure 2.1 (page 5)). To make the vector move clockwise, when moving forward in time, the angular frequency has to be negative.
Real applications are not able to work with continuous signals. So, continuous signals have to be sampled. The discrete signal is defined by [32]:

$$s[m] = s\left(\frac{m}{f_s}\right)$$

with m as discrete time and f_s as sampling frequency.

2.2.2 Harmonics

Mostly, real sounds are more complex than a pure sound [33]. A sound may consist of a *fundamental*, the pure sound itself, and additional *harmonics*. The frequencies of the harmonics are integer multiples of the fundamental, i.e.:

$$f_{Hn} = n \cdot f \qquad\qquad (n\text{th harmonic})$$

with f as frequency of the fundamental and $n > 1$.
The number of harmonics depends on the sending characteristics of a source.

2.2.3 Doppler Effect

When a sound source moves relative to the receiver, the received frequency differs from the sent one. This effect is called *Doppler effect* [28]. In our daily life, you can hear the impact of the Doppler effect, when an ambulance with activated siren passes you. In figure 2.4 (page 9) the Doppler effect is illustrated.

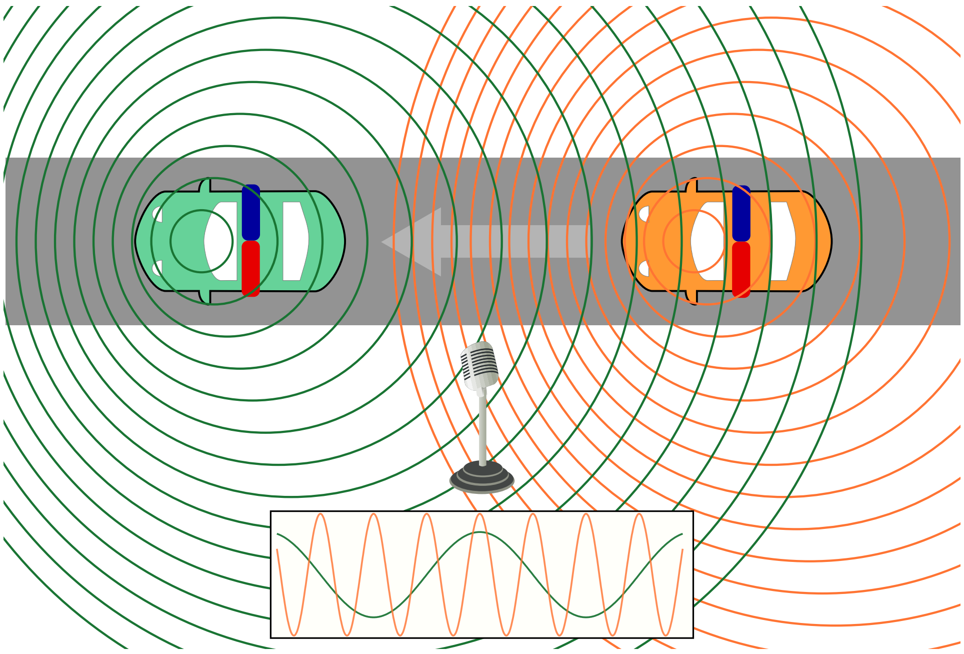


Figure 2.4: Doppler effect - the arrow describes the direction of movement - [34]

When the source moves towards the receiver the peaks of the wave are compressed. Therefore, the wavelength is compressed, either, resulting in a higher frequency. When the source moves away the waves get stretched, so the frequency drops.

The received wavelength λ' is described by the following equation [28]:

$$\lambda' = \lambda\left(1 - \frac{v_s}{c}\right)$$

with λ as the original wavelength, v_s as speed of the source and c as speed of sound. By using equation 2.2 (page 7) the received frequency f' is given by:

$$\begin{aligned} & \lambda = \frac{c}{f} && \text{from equation 2.2 (page 7)} \\ \Rightarrow \quad & \frac{c}{f'} = \frac{c}{f}\left(1 - \frac{v_s}{c}\right) \\ \Rightarrow \quad & f' = f\frac{1}{\left(1 - \frac{v_s}{c}\right)} && (2.6) \end{aligned}$$

with f as source frequency.

2.2.4 Sound Field

The space around a source is divided in *fields*, which define the properties of a received signal.
The *near field* reaches from the sound source to about twice the wave length [35] or 'three times the largest dimension of the sound source' [29]. The sound level does not obey a simple distance law and *particle velocity* is not in phase with pressure.
The *far field* begins where the near field ends and reaches to infinity [29]. In the far field the pressure obeys the *Inverse Square Law* [35], i.e. if the distance to the source doubles the pressure is decreased by the factor of four.
The *reverberant field* is the result of *multipath propagation*. In this field the direct signal and at least one of its reflections are received [29].
The opposite of the reverberant field is the *direct field*, the space of the sound field without reflections.
The *free field* is a space with no reflecting surfaces. In practice, a free field can be assumed when the level of the direct sound is at least twice the level of its reflections [29].

2.2.5 Receiver Model

Due to physical effects, sent signals differ from their received instances. These differences are caused by medium properties, the distance between sender and receiver or additional signal sources. Typically, the presence of a free field or far field is assumed. According to this assumption, the following receiver model is used:

$$x(t) = \sum_{i=0}^{q} a_i s_i(t) \tag{2.7}$$

with $x(t)$ as the *microphone response* - i.e. the data provided by the microphone - q as the number of present signal sources, s_i as ith sources signal and a_i models the impact of the medium and the relative position between sender and receiver. For example the bigger distance between source and receiver, the lower is $|a_i|$ because of damping. Reflections are not explicitly modelled, but they can be treated as additional sources.
The principle behind equation 2.7 (page 10) is called *superposition*, i.e. when multiple signals arrive at the same time, their elongations are summed. This behaviour is also called *interference*. It is important to note that signals interfere *at the receiver*. Consider a situation where you got two directed sources with a narrow propagation path and their paths cross each other in a limited space. Only in that space interference occurs, while outside this particular area only one signal is received.
With equation 2.7 (page 10) the impact of noise is not addressed. This is done by the following equation:

$$x(t) = \sum_{i=0}^{q} a_i s_i(t) + n(t) \tag{2.8}$$

The appended $n(t)$ term is a stochastic process representing white noise (see section 2.2.6 (page 11)).

2.2.6 Noise

Undesired sounds or disturbances are called noise [29].
Some sources of noise are electronic hissing, because of temperature changes [36], and other sound sources. The latter one is *ambient noise*, noise form the environment. The first one is *instrumental noise*, i.e. it depends on the quality of the receiver.
Noise can be also classified by its *noise color*, which describe roughly how the energy is distributed over the frequencies, analogous to light [37]. I.e. red noise got more energy in lower frequency bands, blue noise has more energy in higher frequency bands.
Typically noise is modelled by additive *white noise*. White noise has a constant power spectrum over all frequencies (like white light), so it affects all frequency bands the same way. One way to generate white noise a stochastic process. If the stochastic process follows a temporal normal distribution, i.e. it is normal distributed over time, noise is called *Additive White Gaussian Noise* or just *Gaussian noise*.
In the following, the term white noise is used as synonym for Gaussian noise.
A common measure for noise is the *signal-to-noise ratio (SNR)*. It is defined the following way [38]:

$$\mathrm{SNR} = \frac{P_s}{P_n}$$

with P_s and P_n as power of the signal and noise, respectively. An easier definition utilizes the complex magnitude a of a signal and the standard deviation σ_n:

$$\mathrm{SNR} = \frac{|a|}{\sigma_n}$$

The SNR is dimensionless, so it does not have a unit, but usually it is given in *decibel (*dB*)*:

$$\mathrm{SNR_{dB}} = 10 \log_{10} \left(\frac{|a|}{\sigma_n} \right) \mathrm{dB}$$

Due to the logarithm, the SNR is greater than 0 if the signal is dominant. If noise outweighs the signal, SNR is lower than 0.

2.2.7 Microphones

Microphones are passive sensors which convert the change of pressure into an electric signal. Microphones can be also seen as 'acoustic antennas'. Several microphone designs exist [39], but most of them use a membrane which oscillates when a sound wave impinges.
Piezoelectric microphones use crystal structures attached to a membrane, which change their electric state on pressure changes.
The membrane of a *magnetic microphone*, also called dynamic microphone, is attached to a magnet within a coil. When a wave impinges the membrane, the magnet swings. Due to electromagnetic induction, a current is induced, which is the electric equivalent to the sound waves elongation.
Condenser microphones got an electrically charged backplate behind the membrane. They form a capacitor and when the membrane oscillates, the distance to the backplate changes. Therefore, the capacity and voltage changes.
Only some microphone designs were mentioned because they do not affect the further work. I assume the electric signal, provided by any microphone, is properly digitalized by an *analog-digital converter (ADC)*, so a PC or a microcontroller is able to use it. In the following, I refer to a system consisting of a microphone and an ADC chip as a microphone.
Because of circuit design and used materials, magnitude and phase of a received input signal may differ from the output of the microphone. The dimension of this difference depends on the frequency, so it is called *frequency response* [40].
Another property of a microphone is the *directivity pattern*. It defines the frequency response of a sound which impinges from a specific direction. Determining the frequency response and the directivity pattern of a microphone is quite complex and many measurements are needed.

A *microphone array*, or just *array*, is a composition of microphones. Arrays are used to localize sound sources and to enhance signal quality. The *array geometry*, i.e. the position of the microphones relative to each other, affects the localization. An array of linearly placed omnidirectional microphones cannot determine, whether the source is placed before or behind it, similar to figure 2.5 (page 13). Placing at least one microphone off the axis would overcome this problem. For localization in 3D, the array must not form a plane.

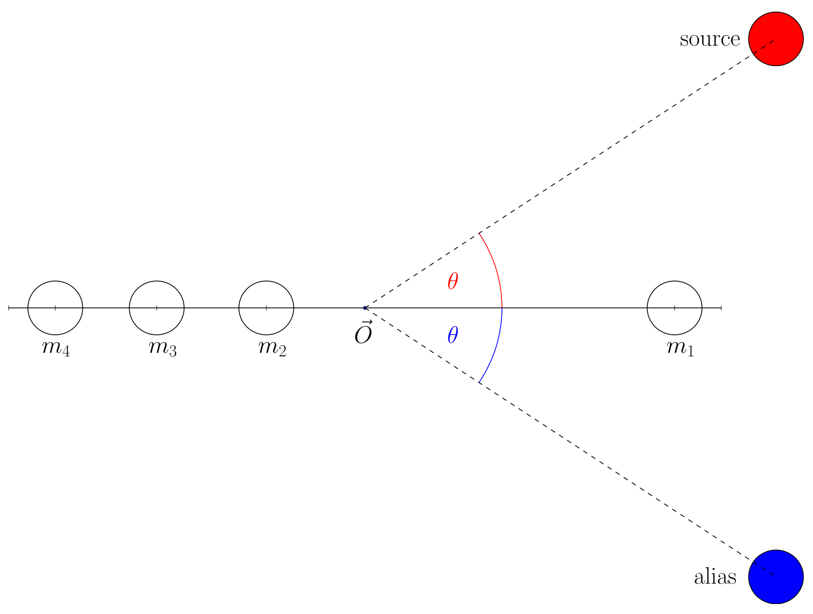


Figure 2.5: Spacial aliasing with a linear array - there are two possible angles of arrival, so the location of the source cannot be determined clearly

Another aspect, affected by array geometry, is the quality of SSL depending on the source frequency. Smaller spacings make localization of higher frequencies more reliable, but also reduce spacial resolution, since time differences of arrival are smaller.
If you do not consider array geometry, *spacial aliasing* will lead to ambiguous estimations.

2.3 Time Domain and Frequency Domain

A signal or a microphone response can be denoted in two ways. The first one is the *time domain*. It is defined as series of *samples* over time:

$$x[m_s], x[m_s+1], \dots, x[m_s+n] \in \mathbb{R}$$

with m_s as the discrete start time (number of the first sample) and n as the number of samples. Data provided by a microphone are usually in time domain.
Another way to describe the response is the *frequency domain*. In the frequency domain the response or signal is considered as a composition of waves with different frequencies over a fixed time interval $[m_s, m_s + n]$. The frequency components are denoted by $X[f, m_s] = a_{m_s}$, where $a_{m_s} \in \mathbb{C}$ is the complex amplitude of equation 2.4 (page 8).
Frequency domain and time domain are related to each other the following way:

$$x[m] = \sum_{k=0}^{K} X[f_k, m_s] \cdot \exp\left(-2\pi f_k \underbrace{\frac{m}{f_s}}_{=t} \cdot j\right) \qquad m_s \leq m \leq n$$

with K as the number of frequency components, m as the discrete time (t is the continuous time), f_s as the sampling rate and f_k as the frequency of the kth component.

2.3.1 Spectrum

The series of frequency components

$$X[f_0, m_s], X[f_1, m_s], \ldots, X[f_N, m_s] \in \mathbb{C}$$

is called *spectrum* and can be obtained from time domain data, e.g. by using the Fourier transform (section 2.4 (page 16)) or wavelet transform [41]. The spectrum gives the intensity of every frequency and can be interpreted in different ways. The first interpretation is the magnitude spectrum $|X[f_0]|, |X[f_1]|, \ldots, |X[f_N]|$, which gives the magnitude of each wave. With this type of spectrum the power and energy of the frequency components can be computed. The second interpretation is the phase spectrum $\arg(X[f_0]), \arg(X[f_1]), \ldots, \arg(X[f_N])$, giving the phase shift for each wave. To simplify the notation $m_s = 0$ is assumed, so m_s can be omitted.

2.3.2 Spectrogram

The intensities of several frequencies do not remain constant over time because sources move (see section 2.2.3 (page 9)), appear out of nowhere and disappear in the next moment. For such situations one spectrum is not suitable, but a series spectrums, which can show the change of frequency components over time, is useful. Such a series is a *spectrogram*:

$$\vec{X}_{\text{spec}}[0], \vec{X}_{\text{spec}}[1], \ldots, \vec{X}_{\text{spec}}[n_{\text{intervals}}]$$

with

$$\vec{X}_{\text{spec}}[i] = (X[f_0, m_s + io], X[f_1, m_s + io], \dots, X[f_K, m_s + io])$$

as the spectrum of the ith interval, $n_{\text{intervals}}$ as number of intervals and o as interval offset in samples.
Typically, spectrograms are visualized as heat maps. The y-axis represents the frequencies, the x-axis the time. This visualization can either give information about magnitude, power or phase shift of the frequency components at a given time. In magnitude spectrograms warmer colors, like red or orange, mean higher intensity, colder colors, like green or blue mean lower intensity. In phase spectrograms warmer colors mean positive phase, colder colors stand for negative phase, turquoise means zero phase shift. Examples for spectrograms are given in figure 2.6 (page 15).

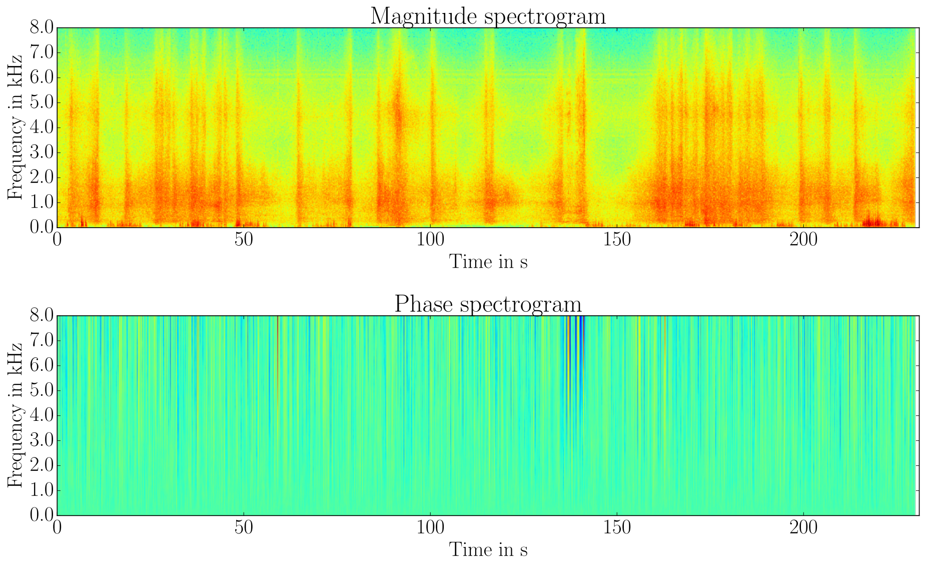


Figure 2.6: Spectrograms of a recording of traffic sounds. Warmer colors mean larger numbers at this time and at this frequency, colder colors mean smaller numbers.

2.3.3 Nyquist-Shannon Sampling Theorem

According to the Nyquist-Shannon Sampling Theorem [42], a signal with frequency f can only be correctly converted to frequency domain if it obeys the following inequation:

$$f < \frac{f_s}{2}$$

with f_s as sampling rate.

The frequency $f_s/2$ is called Nyquist frequency and is highest detectable frequency.
If the frequency is $f > f_s/2$, the signal will be *undersampled*, causing *spectral aliasing* [32, 43]. Aliasing means that the samples are interpreted as another frequency, which is lower than the sampling rate, like in figure 2.7 (page 16).

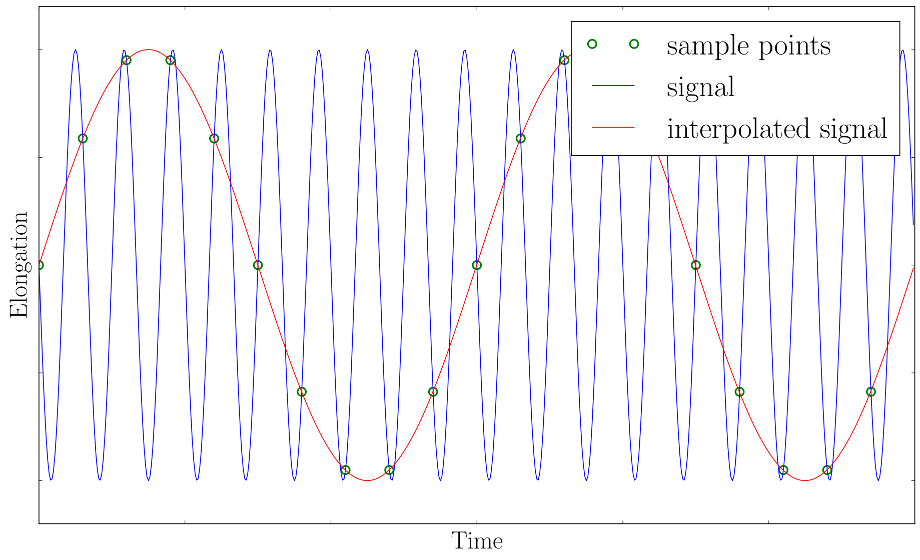


Figure 2.7: Result of aliasing - Due to low sampling frequency, the original signal (blue curve) is misinterpreted (red curve).

To avoid aliasing, *oversampling* can be applied, i.e. the wave is sampled at a higher sampling rate. Then the sampled signal is filtered and downsampled, to obtain a correct signal at a lower sampling rate.

2.4 Discrete Fourier Transform

A popular representation of the frequency domain is the *Discrete Fourier Transform (DFT)*. The DFT is a series of *Fourier coefficients*, which contain the phases and magnitudes of of a set of frequencies. The kth component is defined as [43]:

$$\begin{aligned} X[k] &= \mathcal{F}\left\{x\right\}[k] \\ &= \sum_{m=0}^{M-1} x[m] \cdot e^{-2\pi j m \frac{k}{M}} \qquad 0 \leq k < M \end{aligned} \tag{2.9}$$

with $X[k] \in \mathbb{C}$, M as the number of samples and $x[m] \in \mathbb{C}$ as the mth sample in the time domain. $X[k]$ is the complex amplitude of the wave that performs k cycles per M samples. Phase, magnitude and natural frequency are obtained with following formulas:

$$
\begin{aligned}
|a_k| &= \frac{|X[k]|}{M} = \frac{\sqrt{\Re(X[k])^2 + \Im(X[k])^2}}{M} && \text{(amplitude)} \\
\varphi_k &= \arg(X[k]) = -j \ln\left(\frac{X[k]}{|X[k]|}\right) && \text{(phase)} \\
f_k &= \frac{k}{T_s} = \frac{k}{M/f_s} = \frac{k}{M} f_s && \text{(natural frequency)} \qquad (2.10)
\end{aligned}
$$

with f_s as the sampling rate and T_s as sampling period (reciprocal sampling rate).
As you may see in equation 2.10 (page 17), the considered frequencies are discrete and limited:

$$
\begin{aligned}
f_{\max} &= \frac{f_s}{2} && \text{(maximum/Nyquist frequency)} \\
f_{\min} &= \frac{f_s}{M} && \text{(minimum frequency)}
\end{aligned}
$$

So, more samples are needed if lower frequencies are desired.
An illustration of the result of the DFT is given in figure 2.8 (page 17).

$$f(t) = \underbrace{5}_{\text{dc}} + \underbrace{2\cos(2\pi t - 90\,\mathrm{i}^\circ)}_{1\,\text{Hz}} + \underbrace{3\cos 4\pi t}_{2\,\text{Hz}}$$

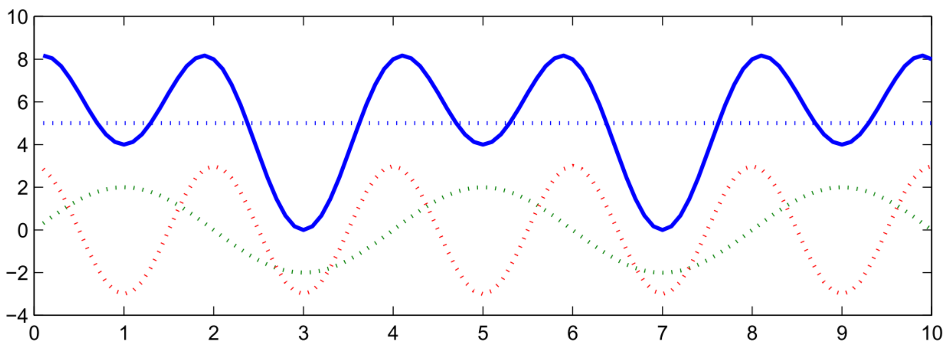


Figure 2.8: Decomposition of a response given by the f (blue curve) into atomic waves (dashed curves) [43].

The DFT is invertible [44] by the *Inverse Discrete Fourier Transform (IDFT)*:

$$
\begin{aligned}
x[m] &= \mathcal{F}^{-1}\{X\}[m] \\
&= \frac{1}{M}\sum_{k=0}^{M-1} X[k] e^{2\pi jk\frac{m}{M}}
\end{aligned}
$$

The first component, i.e. $X[0]$, is the constant level of a signal, $X[\lfloor M/2 \rfloor]$ is the component of the Nyquist frequency. So, the components $X[\lfloor M/2 \rfloor + 1], \ldots, X[M]$ belong to frequencies which are virtually not detectable (see section 2.3.3 (page 15)), but what does it mean?

2.4.1 Symmetry of Spectrum

In equation 2.9 (page 16) the DFT is defined on complex samples - i.e. $x[0], \ldots, x[M-1] \in \mathbb{C}$. If the samples are real - i.e. $x[0], \ldots, x[M-1] \in \mathbb{R}$ -, the DFT is symmetric about the component $M/2$ [43], i.e. $X[k] = X^*[M-k]$. If you remember equation 2.5 (page 8), you will also remember that a real signal can be considered as sum of two complex sinusoids. Therefore, $X[n-k]$ can be considered as the complex amplitude of the second complex sinusoid of equation 2.5 (page 8). So, the formula for the magnitude changes:

$$|a_k| = 2\frac{|X[k]|}{M} = \frac{\sqrt{\Re(X[k])^2 + \Im(X[k])^2}}{M} \qquad 0 < k < \frac{M}{2}$$

The constant component and the Nyquist component are not affected. This is just an intuitive approach to explain the symmetry of the DFT. A proof is given in [44].

2.4.2 Convolution Theorem

The convolution theorem [45] of the DFT says that a convolution of two series, f and h, is equivalent to the multiplication of their DFTs, $F = \mathcal{F}\{f\}$ and $H = \mathcal{F}\{h\}$:

$$(f * h)[m] = \mathcal{F}^{-1}\{F[0] \cdot H[0], F[1] \cdot H[1], \ldots, F[n-1] \cdot H[n-1]\}[m]$$

with n as number of samples. This theorem is important for efficient implementation of convolution.

2.4.3 Short Time Fourier Transform

The DFT describes a spectrum over all samples, but sometimes a spectrogram is needed. Spectrograms can be obtained by *Short Time Fourier Transform (STFT)*:

$$X[k,l] = \sum_{m=0}^{M-1} x[m]w[m-l] \cdot e^{-2\pi j m \frac{k}{M}} \qquad 0 \leq k < M$$

where $X[k,l]$ is the kth Fourier coefficient at time l (lth sample) and w is a *window function*. The window function defines which samples are used and how they are weighted, e.g. a rectangular window of length r is defined the following way:

$$w_{\text{rect}}[k] = \begin{cases} 1 & 0 \leq k < r \\ 0 & \text{else} \end{cases}$$

If n is large, the amount of components is large as well. So, the number of used samples should be limited, like a sliding window:

$$X[k,i] = \sum_{m=i\cdot o}^{i\cdot o+r-1} x[m] \cdot e^{-2\pi jm\frac{k}{M}}$$

with $X[k,i]$ as the kth component of the ith window, r the window size in samples, o the offset between the beginnings of two windows in samples. The offset can be also expressed by an overlap parameter, which describes the number of samples two consecutive windows should overlap.
An important parameter is the number of samples M. It defines the lowest detectable frequency as well as the frequency and time resolution. For better frequency resolution, i.e. smaller steps between the detected frequencies, more samples are needed, but time resolution would suffer.

2.4.4 Fast Fourier Transform

Using equation 2.9 (page 16) requires $\mathcal{O}(M^2)$ time, but there exist some algorithms which compute the DFT in $\mathcal{O}(M \log M)$ time. These algorithms are generally called *Fast Fourier Transform (FFT)* and *Inverse Fast Fourier Transform (IFFT)*, respectively.
The first FFT algorithm was introduced by Cooley and Tukey [26], which is a reinvention of an algorithm of Gauss [46]. This algorithm is a divide-and-conquer algorithm, which recursively splits the computation into smaller parts. Even though the number of samples has to be a power of two, this algorithm is widely used to compute the DFT.

2.5 Analytic Signal

All received signals are real by nature, but sometimes the complex representation is needed. Such a reconstructed complex signal is called *analytic signal* [47].
Since the real part is known, only the imaginary part, also called *quadrature component*, of a sample has to be estimated. This can be done by *Hilbert transform*, which is effectively a convolution with the following function [47]:

$$h_{\text{HT}}[m] = \begin{cases} 0 & m \text{ is even} \\ \frac{2}{\pi m} & m \text{ is odd} \end{cases}$$

Another method was proposed by Marple in [48]. Instead of estimating the quadrature component separately, the captured data are transferred to frequency domain by FFT. With this decomposition complex samples for each atomic frequency can be obtained. Summing the complex samples at a time, like in equation 2.7 (page 10), gives the desired result. This is fast done by IFFT. The pseudocode for this algorithm is given in algorithm 2.1 (page 20).

Algorithm 2.1 Analytic Signal via FFT

```
Require: $x$: series of $M$ samples
1: $X \leftarrow \mathrm{FFT}(x)$
2: initialize $Y \in \mathbb{C}^M$
3: $k \leftarrow 1$
4: for $k \leq \frac{M}{2}$ do
5:     $Y[k] \leftarrow X[k] + X[M-k]$
6:     $k \leftarrow k+1$
7: end for
8: for $k \leq M$ do
9:     $Y[k] \leftarrow 0$
10:    $k \leftarrow k+1$
11: end for
12: return $\mathrm{IFFT}(Y)$
```

2.6 Finite Impulse Response Filter

Typically, only a limited frequency band is desired, so a bandpass filter is needed. This can be done in the frequency domain, e.g. by applying FFT, setting the undesired components to zero and transferring the modified DFT back into the time domain, with IFFT. This is a simple approach, but it is generally a bad idea to filter in frequency domain [49]. The spectrograms in figure 2.9 (page 21) show the impact of a bandpass filter which used FFT. Even though the drawbacks of frequency domain filtering are avoidable, filtering in the time domain is more suitable. A popular approach are *finite impulse response filters (FIR filters)*. The *impulse response* of such a filter has a limited duration, i.e. if you wait long enough (feed the filter with zeros), the filter will provide only zeros, until non-zero input samples will arrive. The impulse response of a system is its output when an impulse, a single peak in time domain (e.g. a 1 just for one sample and the other samples are zeroes), is given as input.
Such filters are defined by [50]:

$$y[m] = \sum_{i=0}^{M} b[i]x[m-i] \tag{2.11}$$

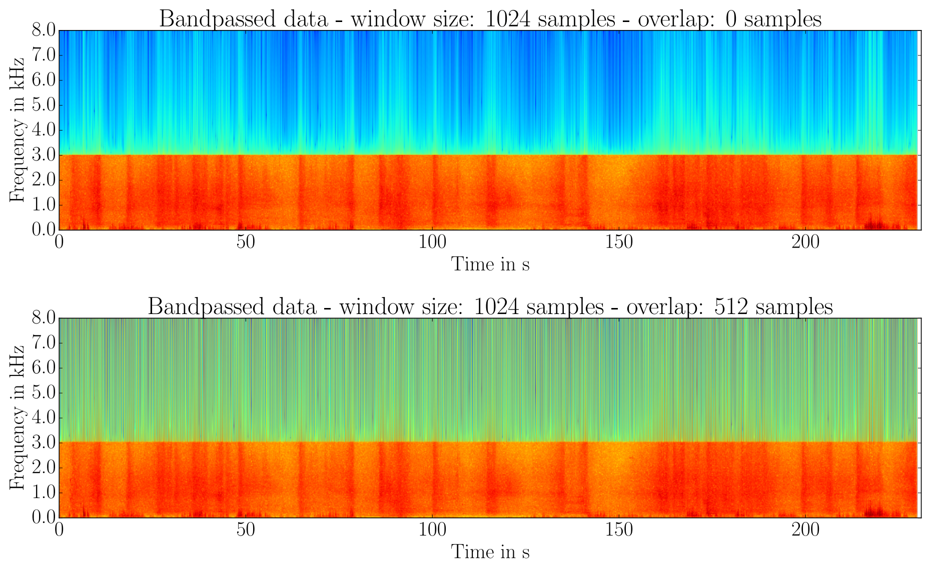


Figure 2.9: Spectrograms, with different overlaps, of low pass filtered data. Filtering has been done with FFT with window size of 1024 samples and an overlap of 0. As you may see in the lower spectrogram, an FFT bandpass filter may affect the frequencies in an undesired way.

where $y[m]$ is the filtered sample at time m (mth sample), M the number of samples, x is the series of input samples and b the series of *filter coefficients*. Maybe you already noticed that equation 2.11 (page 20) is a convolution.
The behaviour of a filter is defined by its filter coefficients. Since FIR filters have to have a finite impulse response, the number of non-zero filter coefficients is limited, i.e. $\forall i < 0 \vee i > N : b[i] = 0$ where N is the *order* of the FIR filter. As consequence of this property, $y[m]$ depends only on the samples $x[m], x[m-1], \dots, x[m-N]$.
A possible implementation works like a shift register, similar to figure 2.10 (page 22).
Implementing an FIR filter is quite simple, but estimation of filter coefficients is not trivial, especially when it comes to frequency filtering. Since equation 2.11 (page 20) is a polynomial, its coefficients can be approximated. A common algorithm for FIR filter design is the *Parks-McClellan algorithm* [51], which utilizes *Remez algorithm* [52]. The objective is to minimize the following error function [53]:

$$||E(\omega)||_\infty = \max_{\omega \in [0,\pi]} |W(\omega)(A(\omega) - D(\omega))| \tag{2.12}$$

with $W(\omega) \geq 0$ as the weighting function, $D(\omega) \in \mathbb{R}$ as the desired response of the filter, $A(\omega) \in \mathbb{R}$ as the actual filter response and $\omega \in [0, \pi]$ as the normalized angular frequency. The frequency is normalized for simplification

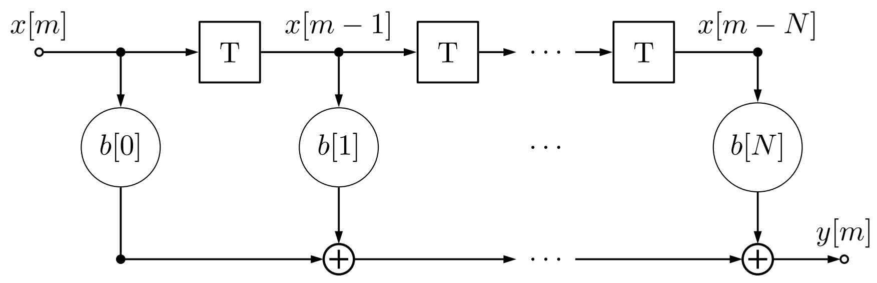


Figure 2.10: FIR filter with N taps, which delay the input samples. The boxes with the T are tap units. Illustration is inspired by [50].

and the interval $[0, \pi]$ can be mapped to the interval $[0, f_{\text{max}}]$ of natural frequencies. The actual response is given by [53]:

$$A(\omega) = \sum_{m=0}^{N} a(m) \cos(m\omega)$$

with N as the order of the FIR filter.
Equation 2.12 (page 21) is also called *weighted Chebyshev error* [53] and describes the maximum difference of the desired and the actual response. So, the Parks-McClellan algorithm minimizes the maximum difference of the desired and the actual response. Implementations of Parks-McClellan or Remez algorithm are widely available, e.g. for Matlab or Python.

2.7 Sound Source Localization

SSL is a fundamental part of human perception. Since almost every human has this ability, it is widely studied and modeled.
Computational Auditory Scene Analysis (CASA) tries to replicate the human ability to analyse situations by sound. Therefore, it is limited to monaural and binaural models, i.e. only one or two microphones are used [54]. However, CASA provides many interesting entry points and models.
Human SSL works with two basic factors [55]. The first one is the *interaural time difference (ITD)*. It is the time a signal needs from one ear to the other. For sounds consisting of a single pulse, e.g. a signal-horn, ITD corresponds to a *time delay of arrival (TDOA)*, for pure tones the ITD results in a phase shift. The ITD is used for frequencies below about 1.5 kHz. For higher frequencies a phase shift cannot be mapped clearly to a direction because of the dimensions of the head. Therefore, the *interaural intensity difference (IID)*, caused by shadowing of the head, is used for frequencies above 1.5 kHz.
However, technical systems are not limited to two microphones. In the following, I refer to systems with more microphones.

2.7.1 Array Receiver Model

The receiver model in equation 2.8 (page 11) is extended to multiple microphones by:

$$x_k[m] = \sum_{i=1}^{q} a_{ki} s_i[m] + n_k[m] \qquad\qquad 0 < k \leq q$$

with $x_k[m]$ as response of the kth microphone, $s_i[m]$ as ith source signal, $n_k[m]$ as the noise component, m as the discrete time, p as the number of microphones and q as the number of sources. The factor a_{ki} models the phase shift and the decrease of intensity of the ith signal to the kth microphone. This model can be written in vector notation:

$$\begin{pmatrix} x_1[m] \\ x_2[m] \\ \vdots \\ x_p[m] \end{pmatrix} = \begin{pmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{pi} \end{pmatrix} \cdot s_i[m] + \begin{pmatrix} n_1[m] \\ n_2[m] \\ \vdots \\ n_p[m] \end{pmatrix}$$
$$\vec{x}[m] = \vec{A}_i \cdot s_i[m] + \vec{n}[m]$$

where $\vec{A}_i \in \mathbb{C}^p$ is the *steering vector* of the ith source and $\vec{x}$ is the array response, i.e. all microphone responses as a vector.
For all q sources the model can be written with a matrix:

$$\vec{x}[m] = \begin{pmatrix} \vec{A}_1 & \vec{A}_2 & \dots & \vec{A}_q \end{pmatrix} \cdot \vec{s}[m] + \vec{n}[m]$$
$$\begin{pmatrix} x_1[m] \\ x_2[m] \\ \vdots \\ x_p[m] \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1q} \\ a_{21} & a_{22} & \dots & a_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1} & a_{p2} & \dots & a_{pq} \end{pmatrix} \cdot \begin{pmatrix} s_1[m] \\ s_2[m] \\ \vdots \\ s_q[m] \end{pmatrix} + \begin{pmatrix} n_1[m] \\ n_2[m] \\ \vdots \\ n_p[m] \end{pmatrix}$$
$$\vec{x}[m] = \mathbf{A} \cdot \vec{s}[m] + \vec{n}[m] \tag{2.13}$$

where $\mathbf{A} \in \mathbb{C}^{p \times q}$ is called *steering matrix* and the columns of $\mathbf{A}$ are the *steering vectors*.
In the frequency domain equation 2.13 (page 23) is given by:

$$\vec{X}[f] = \mathbf{A} \cdot \vec{S}[f] + \vec{N}[f] \tag{2.14}$$

It is important to note that the frequency domain notation does not have to be the DFT. So, f is the natural frequency, which can be mapped to an index of a Fourier coefficient as and when needed.
The entry a_{ki} of $\mathbf{A}$ is the complex amplitude of the ith signal, which depends on the relative position of the sound source to the kth microphone. Therefore, $\mathbf{A}$ encodes the positions of sound sources.

2.7.2 Cross-Correlation

A simple method to determine the TDOA is the cross-correlation (section 2.1.4 (page 6)) of the recording of two microphones. The assumed TDOA is given by [56]:

$$\begin{aligned} r_{x1x2}[\tau] &= (x_1 \star x_2)[\tau] && \text{(cross-correlation)} \\ \hat{\tau} &= \underset{\tau}{\operatorname{argmax}}(r_{x1x2}[\tau]) && \text{(TDOA)} \end{aligned}$$

with x_1 and x_2 as series of samples of the first and second microphone, respectively, $\hat{\tau}$ is the TDOA in samples and argmax returns the argument where the given function, $r_{x1x2}[\tau]$, has its maximum. An example is given in figure 2.11 (page 24).

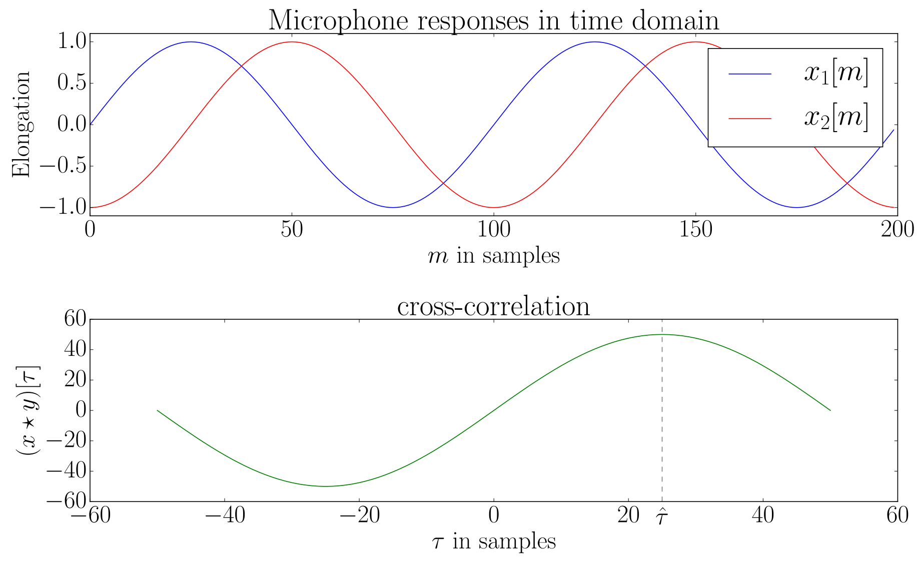


Figure 2.11: Cross-correlation of two simulated microphone responses without noise.

To keep the values of r_{x1x2} comparable, the number of used samples has to be the same. Therefore, proper interval boundaries have to be chosen. According to the convolution theorem of the DFT (section 2.4.2 (page 18)), cross-correlation can be also computed in frequency domain [56]:

$$r_{x1x2}[\tau] = \frac{1}{M} \sum_{k=0}^{M} X_1[k] X_2^*[k] e^{j \frac{2\pi k}{M} \tau} \tag{2.15}$$

with M as the number of samples for the DFT, $X_1 = \mathcal{F}\{x_1\}$ and $X_2 = \mathcal{F}\{x_2\}$.

The corresponding angle is determined by:

$$\begin{aligned} \Delta s &= m\cos\theta && \text{(projection on microphone axis)} \\ \Delta s &= \underbrace{\frac{\hat{\tau}}{f_s}}_{\text{time in seconds}} c && \text{(equation of motion)} \\ \Rightarrow \quad m\cos\theta &= \frac{\hat{\tau}}{f_s}c \\ \Rightarrow \quad \theta &= \arccos\left(\frac{\hat{\tau}c}{f_s m}\right) && \text{(AOA)} \end{aligned}$$

where θ is the AOA, c the speed of sound in $\frac{\mathrm{m}}{\mathrm{s}}$, m the distance between the microphones in meters and f_s the sampling rate in Hz.
Knapp and Carter introduced the *Generalized Cross Correlation (GCC)* in 1976, which extends equation 2.15 (page 24) with a weighting function:

$$g_{x1x2}[\tau] = \frac{1}{M}\sum_{k=0}^{M}\psi(k)X_1[k]X_2^*[k]e^{j\frac{2\pi k}{M}\tau}$$

with $\psi(k)$ as the weighting function. There are several weighting functions, but an empirically well working function is the *Phase Transform (PHAT)*:

$$\psi_{\text{PHAT}}(k) = \frac{1}{|X_1[k]||X_2[k]|}$$

This approach is limited to two microphones, but it is possible to extend it to more than two microphones [57].
The cross-correlation approach is quite simple and intuitive and it also works with noise. However, in environments with multiple sources it suffers from a lack of reliability [56].

2.7.3 Maximum Likelihood

The *Maximum likelihood (ML)* approach works with the model of equation 2.14 (page 23):

$$\vec{X}[f] = \vec{A}(\hat{\theta}) \cdot S[f] + \vec{N}[f]$$

where $\vec{A}(\hat{\theta})$ is the steering vector with the AOA $\hat{\theta}$, f is the natural frequency, S is the source signal in the frequency domain, $\vec{X}$ the array response vector in the frequency domain and $\vec{N}$ the noise component in the frequency domain, which includes ambient noise and reverberation. For simplicity, the entries of $\vec{N}$ are the result of a stochastic process, which is normal distributed in time domain.

The aim of this approach is to maximize a likelihood of a set of parameters, in this case only the AOA θ, given several observations, i.e. an array response vector:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}(\vec{X}[f]|\theta) \tag{2.16}$$

with $\mathcal{L}(\vec{X}[f]|\theta)$ as the likelihood of the observation $\vec{X}[f]$ under the condition that the AOA is θ. The right angle can be determined by *hypothesis testing*, i.e. calculating $\mathcal{L}$ for each θ and picking the θ, which yielded the highest score.
The likelihood is commonly modeled by a multivariate Gaussian distribution [58]:

$$P(\vec{X}[f]|\theta) = \frac{1}{\pi^p \det(\mathbf{R}_n(f))} e^{-\frac{1}{2}(\vec{X}[f]-\vec{A}(\theta)S[f])^H \mathbf{R}_n^{-1}(f)(\vec{X}[f]-\vec{A}(\theta)S[f])} \tag{2.17}$$

with $P(\vec{x}[m]|\theta)$ as the *probability density function (PDF)*, $\mathbf{R}_n(f)$ as the covariance matrix of the noise, det as determinant operator and p as the number of microphones. $\mathbf{R}_n$ can be approximated by the sample covariance matrix:

$$\begin{aligned} \mathbf{R}_n(f) &= E\{\vec{N}[f,m]\vec{N}[f,m]^H\} && \text{(expectation value)} \\ &= \frac{1}{M}\sum_{m=0}^{M} \vec{N}[f,m]\vec{N}[f,m]^H && \text{(sample covariance matrix)} \end{aligned} \tag{2.18}$$

where M is the number of windows used for STFT, and $N[f,m]$ is the complex amplitude for frequency f within the mth time windows.
Most terms of equation 2.17 (page 26) are constant, only the term in the exponent contains parameters. Therefore, the likelihood for a Gaussian distribution can be expressed by [58]:

$$\mathcal{L}_G(X[f]|\theta) = -(\vec{X}[f] - \vec{A}(\theta)S[f])^H \mathbf{R}_n^{-1}(f)(\vec{X}[f] - \vec{A}(\theta)S[f]) \tag{2.19}$$

The appearance of $\mathcal{L}$ depends on the used distribution and the set of parameters to be determined. Even though a Gaussian distribution is a common way to model the likelihood, other distributions may be more appropriate in specific situations, e.g. a multivariate Laplacian distribution [58].
The ML approach natively supports more than two microphones, but it has some drawbacks. Compared to the cross-correlation approach, the determination of the likelihood is computational expensive. The determination of $\mathbf{R}_n$ is nearly impossible. Ambient noise can be measured in quiet periods, but the reverberation cannot be included reliably. Therefore, the performance in real world applications is relatively poor [5].

2.7.4 Beamforming

Beamforming wants to improve signal quality by focusing to a specific direction or even position. So, beamforming is complementary to SSL.
The simplest beamformer is a delay-and-sum beamformer, similar to figure 2.12 (page 27). The samples of each microphone are delayed by an individual amount of time and then summed up. As consequence, the signal from sound sources within the virtual beam should amplify (by constructive interference), while unwanted sounds interfere in a destructive manner.

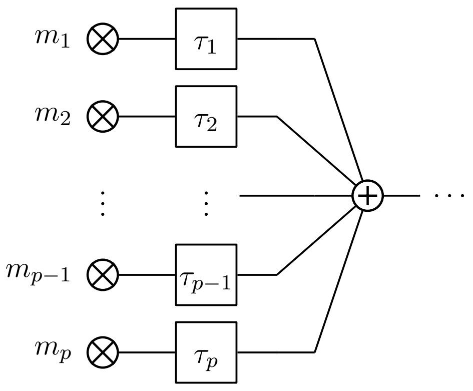


Figure 2.12: Delay-and-sum beamformer with p microphones - m_i is a microphone and τ_i the corresponding delay component ($0 < i \leq p$). The positions of the microphones are almost arbitrary and do not have to conform with this illustration.

More generally, beamforming is formulated the following way [59]:

$$Y[f] = \vec{W}(f,\theta)^H \vec{X}[f] \tag{2.20}$$

with $\vec{W}(f,\theta) \in \mathbb{C}^p$ as the weight vector, which represents the delays and the sum, θ as the focusing angle, $\vec{X}[f]$ as the microphone response vector in frequency domain, f as the natural frequency and p as the number of microphones. The result, $Y[f]$, is the improved signal.
Typically, $Y[f] \approx S[f]$ is desired, so equation 2.20 (page 27) results in:

$$\hat{S}[f] = \vec{W}(f,\theta)^H \vec{X}[f]$$

with $\hat{S}[f]$ as one of the present signals.
To find the right $\vec{W}(f,\theta)$, several beamforming algorithms exist. One of them is the *minimum variance distortionless response (MVDR)* beamformer. For an MVDR beamformer the optimal weight vector is given by [60]:

$$\vec{W}(f,\hat{\theta}) = \underset{\vec{W}(f,\theta)}{\operatorname{argmin}} \vec{W}(f,\theta)^H \mathbf{R}_n(f) \vec{W}(f,\theta) \tag{2.21}$$

$$\text{with constraint} \quad \vec{W}(f,\hat{\theta})^H \vec{A}(\hat{\theta}) = 1$$

with $\vec{W}(f,\hat{\theta})$ as the optimal weight vector for angle $\hat{\theta}$, $\mathbf{R}_n(f)$ as the noise covariance matrix from equation 2.18 (page 26) and $\vec{A}(\theta)$ as the steering vector for angle θ. The closed-form solution for equation 2.21 (page 27) is given by [60]:

$$\vec{W}(f,\hat{\theta}) = \frac{\mathbf{R}_n(f)^{-1}\vec{A}(\hat{\theta})}{\vec{A}(\hat{\theta})^H\mathbf{R}_n(f)^{-1}\vec{A}(\hat{\theta})}$$

DOA estimation with beamforming is done by calculating the SNR for multiple beamformers with different directions. Using MVDR beamformers for such an approach is equivalent to ML-SSL with a Gaussian PDF [60].

2.7.5 Eigenspace Methods

Eigenspace methods are algorithms that estimate the DOA by eigenvalue decomposition of the sample covariance matrix. Pisarenko introduced the first method in 1973 [61]. Schmidt generalized this approach to *Multiple Signal Classification (MUSIC)* [62], which can estimate the DOA of multiple sources, as long as the number of sources is lower than the number of microphones. The array geometry is almost arbitrary.
Another popular technique is *Estimation of Signal Parameters Via Rotational Invariance Techniques (ESPRIT)* [61]. In contrast to MUSIC, ESPRIT limits the array geometry.
Due to the eigenvalue decomposition, eigenspace methods are generally computational intensive, but are robust in multi source environments [63]. However, to avoid restrictions in array geometry I decided to investigate the MUSIC algorithm. So, section 3 (page 29) proceeds with a detailed description of MUSIC.

3 Multiple Signal Classification

3.1 Introduction

MUSIC was invented by Schmidt in 1977. It is designed to estimate multiple parameters - e.g. the location - of multiple signal. Since then, it has been widely investigated and many variations have been developed. It is used for DOA estimation in radio systems, for radar and to locate sound sources. However, MUSIC is not limited to DOA estimation, it is also applicable for spectral estimation [64] or for estimation of signal polarization [62].
This section introduces the MUSIC algorithm for DOA estimation in time domain and frequency domain. A detailed derivation is shown and the algorithm is formulated. At the end, additional aspects, relevant for implementation, are described.

3.2 Mathematical Description

Before I start with the mathematical description, some essential assumptions are made:

1. Different AOAs result in different steering vectors, i.e. $\theta_i \neq \theta_h \Rightarrow \vec{A}(\theta_i) \neq \vec{A}(\theta_h)$.
2. The number of sources is smaller than the number of microphones, i.e. $p > q$.
3. All AOAs are different from each other, therefore, $\mathbf{A}(\vec{\theta})$ has rank q.
4. The entries of $\vec{n}[m]$ are temporal normal distributed with $E\{n_i[m]\} = 0$ (entries have zero mean).
5. The normal distributions behind the entries of $\vec{n}[m]$ all have the same variance σ_n^2 and are independent.
6. Entries of $\vec{x}[m]$ are zero mean.
7. All sources are independent from each other.

These assumptions are important to make MUSIC work.

3.2.1 Time Domain

MUSIC utilizes the array receiver model in equation 2.13 (page 23):

$$\vec{x}[m] = \mathbf{A}(\vec{\theta}) \cdot \vec{s}\,[m] + \vec{n}[m] \tag{3.1}$$

with $\vec{x}[m] \in \mathbb{C}^p$ as the array response, $\mathbf{A}(\vec{\theta}) \in C^{p\times q}$ as the steering matrix, $\vec{s}[m] \in \mathbb{C}^q$ as the vector of source signals, $\vec{n}[m] \in \mathbb{C}^p$ as the noise, $p \in \mathbb{N}$ as

the number of microphones and q as the number of sources. The angles of the sources are encoded by $\mathbf{A}(\vec{\theta})$ and given by $\vec{\theta} = (\theta_1, \theta_2, \dots, \theta_q)$, where θ_i is the AOA of the ith sources, i.e. it corresponds to $s_i[m]$.
Let $\mathbf{R}_x = E\{\vec{x}[m]\vec{x}[m]^H\}$ be the covariance matrix of the array response vectors. $\mathbf{R}_x$ can be approximated by the sample covariance matrix [61]:

$$\mathbf{R}_x = \frac{1}{M} \sum_{m=0}^{M-1} \vec{x}[m]\vec{x}[m]^H \tag{3.2}$$

with M as the number of samples. Please remember that $x[m]$ has zero mean, such that the mean $\mu = 0$. So, the difference of a typical covariance matrix can be omitted.
Equation 3.1 (page 29) is a linear map, so $\mathbf{R}_x$ follows [62]:

$$\begin{aligned}\mathbf{R}_x &= \mathbf{A}(\vec{\theta})E\{s[m]s[m]^H\}\mathbf{A}(\vec{\theta})^H + E\{\vec{n}[m]\vec{n}[m]^H\} \\ &= \mathbf{A}(\vec{\theta})\mathbf{R}_s\mathbf{A}(\vec{\theta})^H + \mathbf{R}_n \end{aligned} \tag{3.3}$$

Due to the independence of noise, equation 3.3 (page 30) can be written as [65]:

$$\begin{aligned}\mathbf{R}_x &= \mathbf{A}(\vec{\theta})\mathbf{R}_s\mathbf{A}(\vec{\theta})^H + \sigma_n^2\mathbf{I} \qquad (3.4)\\ \Rightarrow \mathbf{R}_x - \sigma_n^2\mathbf{I} &= \mathbf{A}(\vec{\theta})\mathbf{R}_s\mathbf{A}(\vec{\theta})^H\end{aligned}$$

with $\mathbf{I}$ as *identity matrix*, i.e.:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}$$

Due to the second and third assumption, $\mathbf{A}(\vec{\theta})$ spans a q dimensional subspace within a p dimensional space. Therefore, $\mathbf{A}(\vec{\theta})$ is not of full rank. So, $\mathbf{A}(\vec{\theta})\mathbf{R}_s\mathbf{A}(\vec{\theta})^H$ is not of full rank, either, so:

$$\begin{aligned} \det(\mathbf{A}(\vec{\theta})\mathbf{R}_s\mathbf{A}(\vec{\theta})^H) &= 0 \\ \Rightarrow \det(\mathbf{R}_x - \sigma_n^2\mathbf{I}) &= 0 \end{aligned} \tag{3.5}$$

were det is the determinant operator.
The relation in equation 3.5 (page 30) is the *characteristic polynomial*, so σ_n^2 has to be an eigenvalue of R_x [66].
At this point we know that $\mathbf{R}_x \in \mathbb{C}^{p\times p}$ has at least one eigenvalue with value σ_n^2. We also know, that $\mathbf{R}_s$ and $\mathbf{A}(\vec{\theta})$ have rank q, so $\mathbf{A}(\vec{\theta})\mathbf{R}_s\mathbf{A}(\vec{\theta})^H \in \mathbb{C}^{p\times p}$ also has rank q and $\mathbf{A}(\vec{\theta})\mathbf{R}_s\mathbf{A}(\vec{\theta})^H \in \mathbb{C}^{p\times p}$ is positive semidefinite [61]. Therefore, the largest q eigenvalues of $\mathbf{A}(\vec{\theta})\mathbf{R}_s\mathbf{A}(\vec{\theta})^H$ are $\tilde{\lambda}_1 > \tilde{\lambda}_2 > \dots >$

$\tilde{\lambda}_q > 0$ and the $p-q$ smallest eigenvalues are $\tilde{\lambda}_{q+1} > \tilde{\lambda}_{q+2} > \ldots > \tilde{\lambda}_p = 0$. Remembering equation 3.4 (page 30), the noise eigenvalues are additive [65]. So, the eigenvalues of $\mathbf{R}_x$ are:

$$\lambda_i = \begin{cases} \tilde{\lambda}_i + \sigma_n^2 & 0 < i \leq q \\ \sigma_n^2 & q < i \leq p \end{cases}$$

where λ_i is the ith largest eigenvalue of R_x.
Maybe you are wondering why it is possible to order the eigenvalues, although they have to be complex as long as their matrix is complex. Because of equation 3.2 (page 30), $\mathbf{R}_x$, $\mathbf{R}_s$ and $\mathbf{A}(\vec{\theta})\mathbf{R}_s\mathbf{A}(\vec{\theta})^H$ are hermitian matrices, i.e. $\mathbf{R}_x = \mathbf{R}_x^H$. According to [67], the eigenvalues of a hermitian matrix are real. Therefore, all eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$ of $\mathbf{R}_x$ are real.
The eigenvectors with the q largest eigenvalues of $\mathbf{R}_x$ span the same space like $\mathbf{A}(\vec{\theta})$, the signal space. The eigenvectors with the $p-q$ smallest eigenvalues span the noise space. Let $\mathbf{U}_n$ be the concatenation of the $p-q$ eigenvectors that span the noise space:

$$\mathbf{U}_n = \begin{pmatrix} \vec{w}_{q+1} & \vec{w}_{q+2} & \ldots & \vec{w}_p \end{pmatrix}$$

with $\vec{w}_i$ as the eigenvector which corresponds to the ith largest eigenvalue. $\mathbf{U}_n$ spans the noise space, so $\mathbf{A}(\vec{\theta})$ and $\mathbf{U}_n$ are orthogonal, i.e.:

$$\vec{A}(\theta_i)^H \mathbf{U}_n \mathbf{U}_n^H \vec{A}(\theta_i) = 0 \qquad \text{for } 0 < i \leq q \tag{3.6}$$

By testing all possible steering vectors with equation 3.6 (page 31) the right steering vectors can be obtained.
In real life only a finite amount of samples is available, which results in variations of the eigenvalues. Therefore, the $p-q$ smallest eigenvalues are not equal, but cluster around σ_n^2. Because of these variations and the fact that the calculations suffer from discrete computation, equation 3.6 (page 31) does not work. Alternatively, steering vectors as orthogonal as possible are searched with:

$$P_{MU}(\theta) = \frac{1}{\vec{A}(\theta)^H \mathbf{U}_n \mathbf{U}_n^H \vec{A}(\theta)}$$

where P_{MU} is a spacial spectrum [68], the *MUSIC spectrum*. By picking the q largest peaks of P_{MU} the AOAs of all q sources are determined.
This is the classical MUSIC approach introduced by Schmidt [62]. It is also called *spectrum MUSIC*.
This formulation assumes that all sources are sending at the same frequency, so the performance of time domain MUSIC highly depends on a proper guess of it. However, multiple sources sending on the same frequency would result in unresolvable interference.

3.2.2 Frequency Domain

The derivation for MUSIC in the frequency domain is similar to section 3.2.1 (page 29). It starts with the array receiver model in the frequency domain from equation 2.14 (page 23):

$$\vec{X}[f,i] = \mathbf{A}(\vec{\theta}, f) \cdot \vec{S}[f,i] + \vec{N}[f,i] \tag{3.7}$$

with f as the natural frequency, i as the index of the time window which is used to obtain the frequency domain and $\mathbf{A}(\vec{\theta}, f)$ as the steering matrix for frequency f. The assumptions remain the same.
In equation 3.7 (page 32) it is assumed that all sources send at the same frequency. Typically, this would result in unresolvable interference. To overcome this problem, it is assumed that one source is dominant in a frequency band at a time.
Instead of using the covariance matrix in equation 3.2 (page 30), the *cross-spectral matrix* is used [69]:

$$\begin{aligned}\mathbf{R}_X[f] &= E\{X[f,i]X[f,i]^H\}\\ &= \frac{1}{R}\sum_{i=0}^{R-1} X[f,i]X[f,i]^H\end{aligned}$$

with R as number of time windows.
For the eigenvectors of $\mathbf{R}_X[f]$ and the eigenvectors of $\mathbf{R}_x$ the same properties apply. Therefore, $\mathbf{U}_n[f]$ is the matrix consisting of the eigenvectors with the $p-q$ smallest eigenvalues of $\mathbf{R}_x[f]$. The MUSIC spectrum is described by:

$$P_{MU}(\theta, f) = \frac{1}{\vec{A}(\theta,f)^H \mathbf{U}_n[f]\mathbf{U}_n[f]^H \vec{A}(\theta,f)} \tag{3.8}$$

The spectrum given by equation 3.8 (page 32) is only for one frequency, so the spectrums have to be merged. This can be done using the geometric mean [70]:

$$P_{MU}(\theta) = \left(\prod_{k=0}^{K} P_{MU}(\theta, f_k)\right)^{\frac{1}{K}} \tag{3.9}$$

with f_k as kth frequency and K as number of frequencies.
This variation of MUSIC is also called *independent frequency bin (IFB) MUSIC* [71].
In contrast to the time domain formulation, the frequency domain formulation does not have to assume that sources send on a specific frequency, because the processing is done for each frequency bin separately. So, it is more suitable to locate sources which send on multiple frequency bands or send white noise. Since pure sounds are almost not observable in nature, frequency domain MUSIC is usually used for such applications.

3.3 Implementation Details

The mathematical formulation is pretty generic, but for implementation some details have to be considered. Examples of the algorithms are illustrated in figure 3.1 (page 34).
Samples provided by an array are usually real, but MUSIC in time domain requires complex samples. Therefore, an analytic signal generator [72] has to be used to obtain $\vec{x}[0], \dots, \vec{x}[M-1] \in \mathbb{C}$.
In the frequency domain the analytic signal generator is replaced by a method to obtain the frequency domain. A common approach is using STFT with FFT [7, 73], but other method, e.g. wavelet transform [74], are also suitable. For each frequency bin the MUSIC algorithm is performed. At the end, the MUSIC spectrums are merged to one, typically by taking the mean of all spectrums. The geometric mean in equation 3.9 (page 32) [70], but also the arithmetic mean [7], is applicable.

3.3.1 Steering Vectors

The performance of MUSIC highly depends on the steering vectors. To simplify their generation, following assumptions are made:

1. The array is in far field to all sources.
2. Sounds are plane waves, i.e. wavefronts emitted by the same source are parallel and linear.

With these assumptions, a real localization of sound sources is not possible, but the direction can be determined. Fortunately, in traffic the direction of a sound source is enough.
As a consequence of these assumptions, the propagation paths are parallel. Therefore, the DOA for each microphone is the same, similar to figure 3.2 (page 35).
Due to the propagation paths being parallel, a distance Δs can be computed with a simple orthogonal projection. For an array similar to figure 3.2 (page 35) the propagation distance for the kth microphone is given by:

$$\Delta s_k(\theta) = m_k \cos\theta \tag{3.10}$$

with m_k as the distance of the kth microphone from the origin $\vec{O}$ and θ as angle of arrival. This distance can be determined more generally:

$$\begin{aligned} \Delta s_k(\theta) &= \frac{-\vec{m}_k \cdot \vec{v}(\theta)}{\vec{v}(\theta) \cdot \vec{v}(\theta)} && \text{or} \\ \Delta s_k(\theta) &= -\vec{m}_k \cdot \vec{v}(\theta) && \text{for } |\vec{v}(\theta)| = 1 \end{aligned}$$

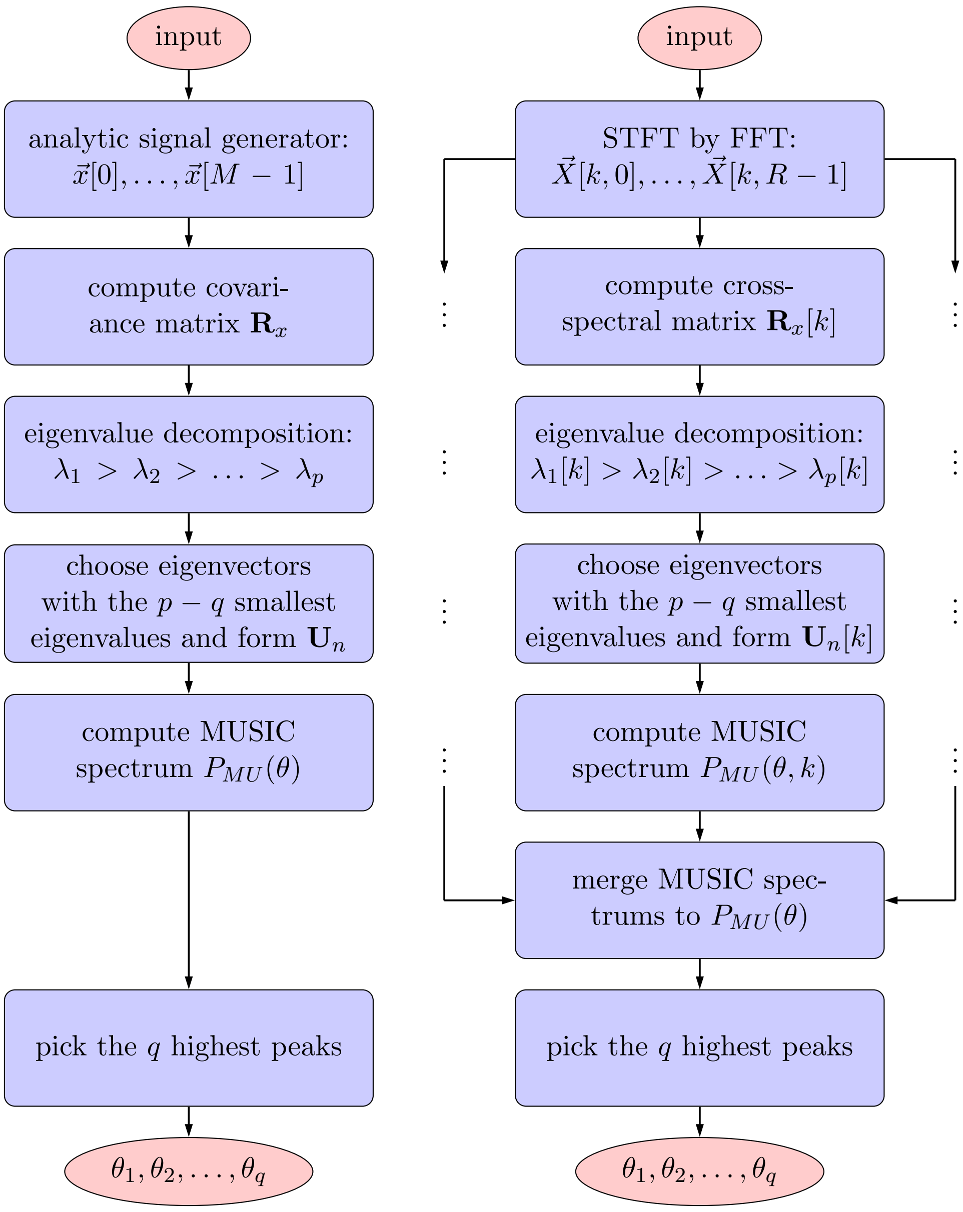

input
analytic signal generator: $\vec{x}[0], \ldots, \vec{x}[M-1]$
compute covariance matrix $\mathbf{R}_x$
eigenvalue decomposition: $\lambda_1 > \lambda_2 > \ldots > \lambda_p$
choose eigenvectors with the $p-q$ smallest eigenvalues and form $\mathbf{U}_n$
compute MUSIC spectrum $P_{MU}(\theta)$
pick the q highest peaks
$\theta_1, \theta_2, \ldots, \theta_q$
input
STFT by FFT: $\vec{X}[k,0], \ldots, \vec{X}[k,R-1]$
compute cross-spectral matrix $\mathbf{R}_x[k]$
eigenvalue decomposition: $\lambda_1[k] > \lambda_2[k] > \ldots > \lambda_p[k]$
choose eigenvectors with the $p-q$ smallest eigenvalues and form $\mathbf{U}_n[k]$
compute MUSIC spectrum $P_{MU}(\theta, k)$
merge MUSIC spectrums to $P_{MU}(\theta)$
pick the q highest peaks
$\theta_1, \theta_2, \ldots, \theta_q$

Figure 3.1: MUSIC in the time domain (left) and in the frequency domain (right). You can see that both implementations are pretty similar, but the execution branches at the very beginning and is done for each DFT frequency bin.

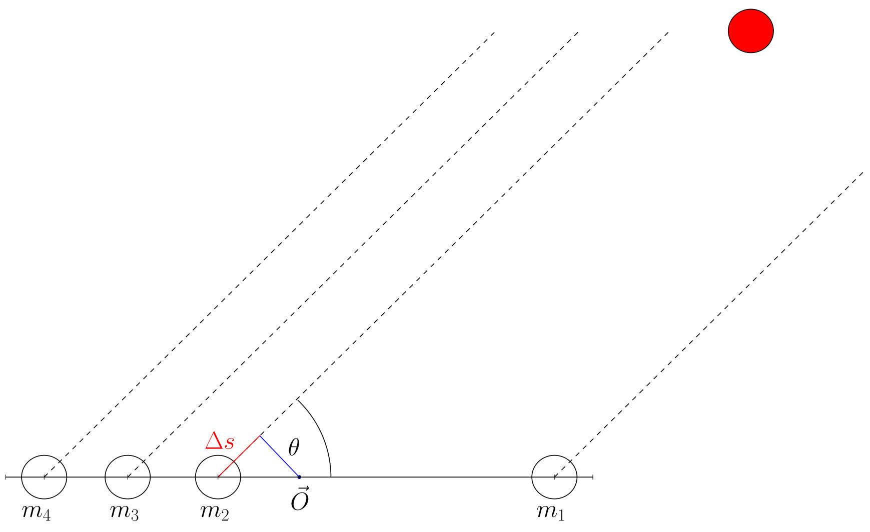


Figure 3.2: Illustration of array receiver model. $\vec{O}$ is the reference point of the array, the red filled circle is the source and $m_1, \ldots, m_4$ are the microphones. The dashed lines are the directions of arrival of the sound and θ is the corresponding *angle of arrival (AOA)*. The blue line is the orthogonal projection of $\vec{O}$ to the propagation path of sound.

with $\vec{m}_k$ as the position of the kth microphone relative to $\vec{O}$ and $\vec{v}(\theta)$ as the direction of arrival. The relation of DOA and angle of arrival is defined by:

$$\vec{v}(\theta) = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$$

The distance results in a TDOA and a phase shift:

$$\tau_k(\theta) = \frac{\Delta s_k(\theta)}{c} \qquad \text{(TDOA)}$$
$$\varphi_k(\theta, f) = 2\pi f \tau_k(\theta) \qquad \text{(phase shift)}$$

with $\tau_k(\theta)$ as the time delay to the kth microphone, c as the speed of sound and f as the frequency of the source.
With the phase shift, the steering vector $\vec{A}_i$ for the ith source is determined by:

$$\vec{A}(\theta, f) = \begin{pmatrix} \beta_1(\theta, f)e^{j\cdot\varphi_1(\theta,f)} \\ \beta_2(\theta, f)e^{j\cdot\varphi_2(\theta,f)} \\ \vdots \\ \beta_p(\theta, f)e^{j\cdot\varphi_p(\theta,f)} \end{pmatrix}$$

with p as the number of microphones, θ as the AOA and f as frequency.

The function $\beta_k(\theta, f)$ models the decrease of the intensity and depends on the directivity patterns of the microphones. For simplicity, an omnidirectional directivity pattern is assumed and the distances of all sources are big enough, such that the distance between the microphones is too small to have significant decrease of registered magnitude. Therefore, $\beta_k(\theta, f) = 1$ is assumed. This results in a simpler steering vector:

$$\vec{A}(\theta, f) = \begin{pmatrix} e^{j\cdot\varphi_1(\theta,f)} \\ e^{j\cdot\varphi_2(\theta,f)} \\ \vdots \\ e^{j\cdot\varphi_p(\theta,f)} \end{pmatrix}$$

3.3.2 Determination of the Number of Sources

Another important aspect is the determination of the number of sources q. Guessing the wrong number of sources affects the reliability of DOA estimation in a negative way.

There are several approaches to solve this problem. One approach is the *independent component analysis (ICA)* [75]. It separates signals by their sending characteristic, i.e. distribution of samples over time. Other approaches utilize the eigenvalues of the covariance matrix or the cross-spectral matrix, respectively. In [76] two information theoretic criteria were proposed, which rely on the ratio of the geometric mean and the arithmetic mean of the eigenvalues.

The simplest method picks the number of eigenvalues, which exceed a static threshold. The problem with static thresholds is that they do not adapt. In situations with a high sound level too many sources are detected, in quiet situations too few sources are found. A more robust solution is given by the ratio of the sum of the n largest eigenvalues to the sum of all eigenvalues:

$$q = \min \left\{ n \;\middle|\; \forall n : 1 \leq n \leq p \wedge \frac{\sum_{i=1}^{n} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \geq \gamma \right\} \tag{3.11}$$

with p as the number of microphones, $\lambda_1 > \lambda_2 > \ldots > \lambda_p$ as the eigenvalues of the covariance matrix or cross-spectral matrix, respectively, and $0 < \gamma < 1$ as relative threshold, e.g. $\gamma = 0.95$. In equation 3.11 (page 36) the smallest number of dominant eigenvalues is determined. The sensitivity is defined by γ. For quiet situations a bigger γ can be chosen, for noisy environments smaller γ may provide better results. For $q = p$ the presence of too much sources or no source can be assumed.

4 Implementation

4.1 Kinect

In November 2010, the *Kinect* was released by Microsoft as an extension for the *Xbox 360*. It is a 'webcam-like' composition of sensors, which originally enables the user to interact with his Xbox by gestures and sound. So, the whole body becomes the controller. Because of its low price and of its sensors, especially its infrared depth sensor, the Kinect became a popular system for human-computer-interfaces or mobile robotics. An illustration of the sensor positions is given in figure 4.1 (page 37).

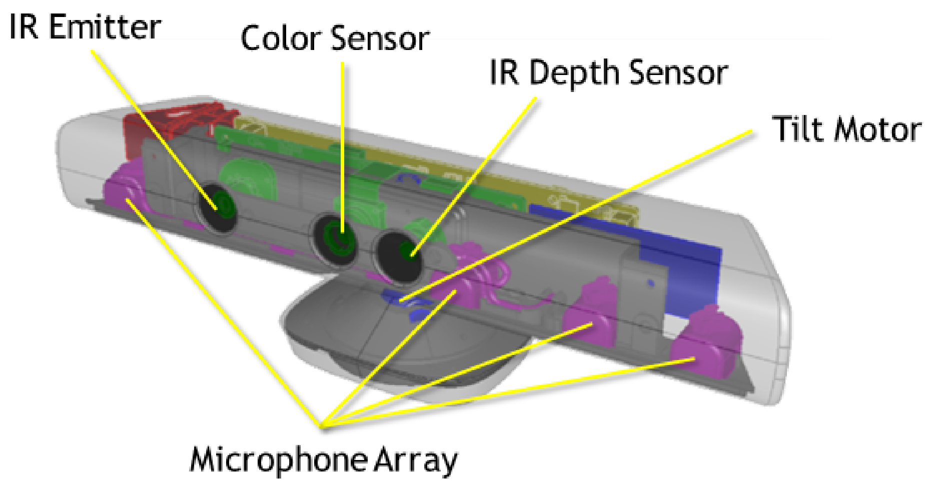


Figure 4.1: Illustration of Kinect with description of sensor [77]

For my purpose the microphone array is relevant. It consists of four linearly aligned microphones, which face downward [78]. The array geometry is given in figure 4.2 (page 37).

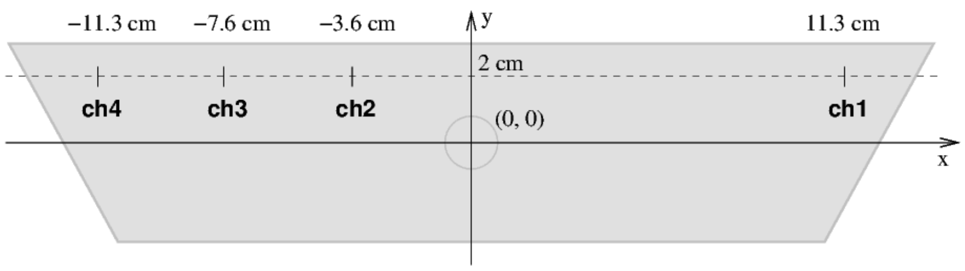


Figure 4.2: Kinect microphone array geometry [79]

The conversion from analog signals to digital data is done by two Wolfson Microelectronics WM8737G stereo ADC chips [78]. According to [80], one chip can capture the responses of two microphones simultaneously. To re-

duce the impact of spectral aliasing, oversampling is applied. Sampling rates up to 96 kHz are supported.
The Kinect provides 24 bit PCM data, i.e. the signal is given as series of 24 bit signed integers. The samples are given as three byte, little-endian integer and the sampling rate is 16 kHz [77].
The directivity patterns of the microphones are cardioid [81], i.e. they are more sensitive to sounds coming from the front than to sounds coming from the back. A visualization of the directivity pattern is given in figure 4.3 (page 38).

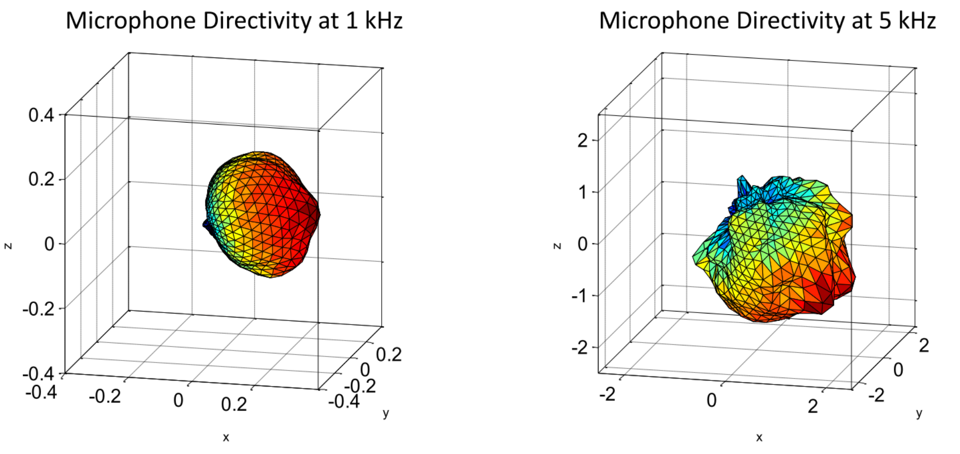


Figure 4.3: Directivity pattern of a microphone of the Kinect for 1 kHz (left) and 5 kHz (right) [81]. Red means that the ratio of received power from this direction to the averaged overall power is high (high directivity), blue means low directivity.

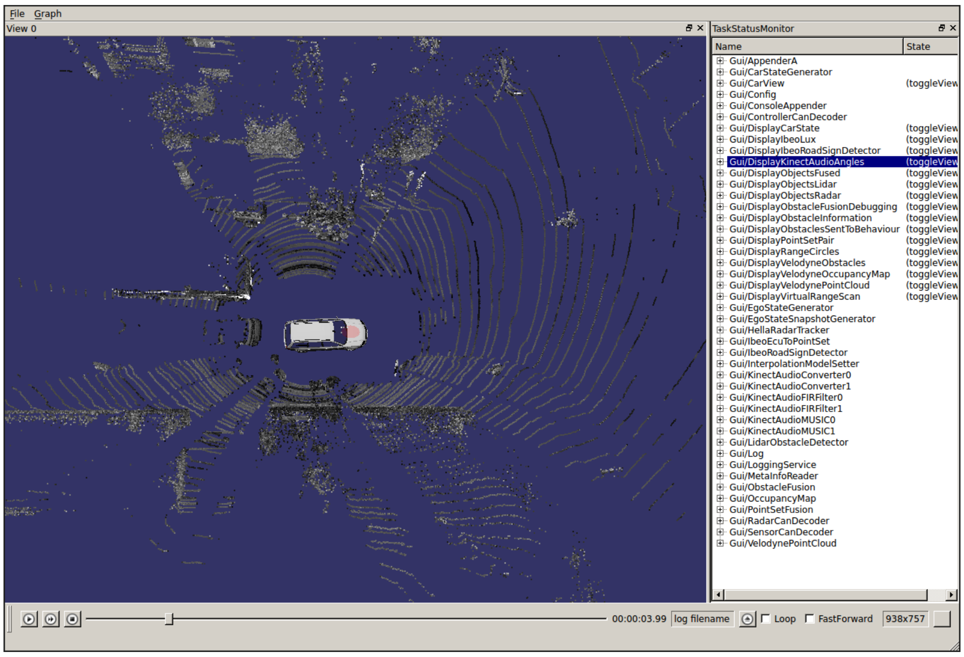
According to figure 4.3 (page 38), the microphones of the Kinect are almost completely insensitive to sounds coming from behind. The sensitivity from 0° to 180° seems to be almost uniform.

4.2 Software

4.2.1 Log Player

The *log player* is the foundation for development for the autonomous cars of the *Autonomous Labs*[1]. It is written in C++ and implements a flexible module system, which bases on *OROCOS* [82] for real time processing. Modules implement behaviours, visualizations, etc. and communicate by message-passing. The output of modules can be logged by the `Logger` module and saved in files, e.g. when sensor data are recorded. These files can also be replayed and the logged output can be used as input for other modules. Running modules on life data is possible, either. A screenshot of the log player GUI is given in figure 4.4 (page 39).

[1]`http://autonomos.inf.fu-berlin.de/`



Name	State
Gui/AppenderA	
Gui/CarStateGenerator	
Gui/CarView	(toggleView
Gui/Config	
Gui/ConsoleAppender	
Gui/ControllerCanDecoder	
Gui/DisplayCarState	(toggleView
Gui/DisplayIbeoLux	(toggleView
Gui/DisplayIbeoRoadSignDetector	(toggleView
Gui/DisplayKinectAudioAngles	(toggleView
Gui/DisplayObjectsFused	(toggleView
Gui/DisplayObjectsLidar	(toggleView
Gui/DisplayObjectsRadar	(toggleView
Gui/DisplayObstacleFusionDebugging	(toggleView
Gui/DisplayObstacleInformation	(toggleView
Gui/DisplayObstaclesSentToBehaviour	(toggleView
Gui/DisplayPointSetPair	(toggleView
Gui/DisplayRangeCircles	(toggleView
Gui/DisplayVelodyneObstacles	(toggleView
Gui/DisplayVelodyneOccupancyMap	(toggleView
Gui/DisplayVelodynePointCloud	(toggleView
Gui/DisplayVirtualRangeScan	(toggleView
Gui/EgoStateGenerator	
Gui/EgoStateSnapshotGenerator	
Gui/HellaRadarTracker	
Gui/IbeoEcuToPointSet	
Gui/IbeoRoadSignDetector	
Gui/InterpolationModelSetter	
Gui/KinectAudioConverter0	
Gui/KinectAudioConverter1	
Gui/KinectAudioFIRFilter0	
Gui/KinectAudioFIRFilter1	
Gui/KinectAudioMUSIC0	
Gui/KinectAudioMUSIC1	
Gui/LidarObstacleDetector	
Gui/Log	
Gui/LoggingService	
Gui/MetaInfoReader	
Gui/ObstacleFusion	
Gui/OccupancyMap	
Gui/PointSetFusion	
Gui/RadarCanDecoder	
Gui/SensorCanDecoder	
Gui/VelodynePointCloud	

Figure 4.4: Screenshot of log player GUI - the gray point clouds are the visualization of lidar data.

To capture the data from the Kinect *libfreenect*[2] is used. It is an open source C/C++ library which provides an API and drivers to make Kinects accessible for multiple platforms. Captures were made by the default WAV recorder of libfreenect or by the `KinectAudioGrabber` module.
For linear algebra operations in C++, e.g. vector operations or eigenvalue decomposition, the *Eigen* [83] template library is used. It provides functions for FFT and IFFT, either.
The log player was used to test MUSIC implementations on real world sound data. In conjunction with lidar point clouds, qualitative observations of DOA accuracy were made.

4.2.2 Analysis and Simulation Environment

Sound data analysis and simulations were done in Python[3]. To simplify implementation, SciPy [84] was used, a Python ecosystem for scientific computing. SciPy consists of several packages, partly implemented in C/C++ for faster processing. Especially, the following packages were used:

- SciPy library[4]: This library provides several packages for many nu-

[2] https://github.com/OpenKinect/libfreenect
[3] https://www.python.org/
[4] https://www.scipy.org/scipylib/index.html

merical tasks. Primarily, the `signal` package[5] and `stats` package[6] were used.

- NumPy[7]: A C/C++ based package for linear algebra, FFT and random number generation.
- matplotlib [85]: A library for visualization by plots and spectrograms. Some of the illustrations are also made with matplotlib.

Basing on these libraries, a simple simulation environment was implemented, `pySoundToolbox` [86]. It implements the array receiver model of equation 2.13 (page 23) for pure sounds, sources with statistically distributed sending characteristic and sources from WAV files.

4.2.3 MUSIC Implementations

Several MUSIC implementations were created.
I started with a time domain MUSIC implementation as log player module, which never really worked. At this point I did not know whether my implementation was wrong or the provided sound data (sound of passing cars) were not suitable for DOA estimation.
So, I proceeded with a time domain MUSIC implementation for simulations, to investigate the properties of MUSIC in a controllable environment. During the tests on artificially generated data, I found out that an analytic signal has to be generated before computing the covariance matrix. This was probably the major mistake in the first implementation.
The third implementation was IFB MUSIC in Python. It utilizes parts of the time domain MUSIC implementation and can also estimate the DOA of sources which do not emit pure sound.
The Python implementations are available as a package in [87].
With the experience from the simulations, an IFB MUSIC module for the log player was created. The structure is shown in figure 4.5 (page 41).
The `KinectAudioGrabber` module is part of a library which makes the data from the Kinect accessible to other log player modules. It utilizes the libfreenect API to fetch the raw PCM data from one Kinect. Its output is either logged by `Logger` or directly processed.
`KinectAudioConverter` converts the samples from 3 byte little-endian to 4 byte big-endian integer and stores them in C++ `std::vectors`. These vectors are the time domain data.
To focus on significant frequency bands, an FIR filter is applied. `KinectAudioFIRFilter` is a simple implementation, which loads the filter coefficients from a plain text file. Depending on the number of filter coefficients,

[5] http://docs.scipy.org/doc/scipy/reference/signal.html
[6] http://docs.scipy.org/doc/scipy/reference/stats.html
[7] http://www.numpy.org/

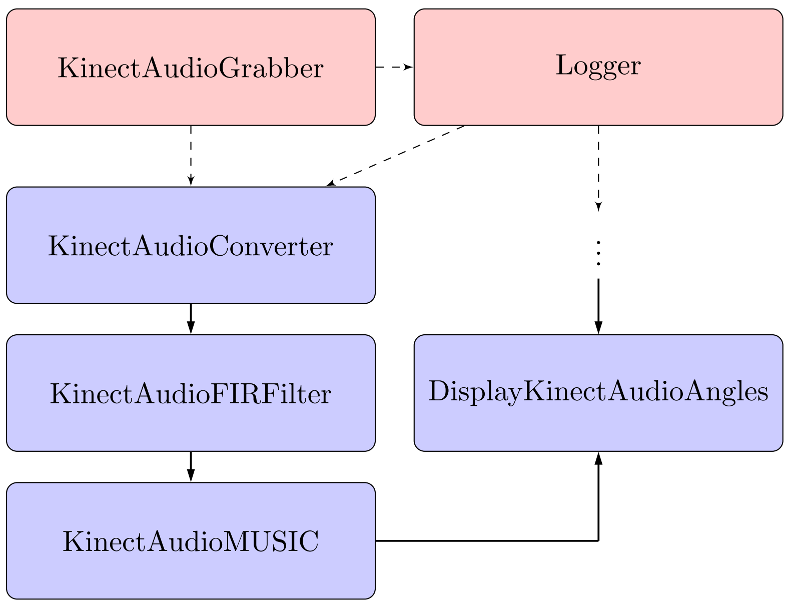


Figure 4.5: Structure of the IFB MUSIC implementation for the log player. For each Kinect an instance of each class which starts with '`Kinect`' is created. Either data are processed life from `KinectAudioGrabber` or are logged by `Logger` and replayed later. `DisplayKinectAudioAngles` visualizes the angles for all Kinects.

the received data are buffered, so always enough samples for filtering are available. The filter coefficients are determined by SciPys `signal.remez` function.
`KinectAudioMUSIC` implements the IFB MUSIC algorithm illustrated on the right side of figure 3.1 (page 34). It provides several options to make processing faster and DOA estimation more flexible. The most important are:

- number of samples used for the FFT
- range of the Fourier coefficients to use
- angle range to test, to avoid spacial aliasing
- number of angle steps to define angle resolution
- used time range for calculation of the cross-spectral matrix

Generating steering vectors on the fly made execution of `KinectAudioMUSIC` instances unreasonably slow. So, steering vectors are generated and saved at the beginning of execution, to overcome this performance issues.
The number of eigenvectors which form the noise matrix is estimated for each frequency bin by equation 3.11 (page 36). The MUSIC spectrums are merged by the geometric mean (equation 3.9 (page 32)) and the three highest

peaks are picked. An additional estimation of the number of sources is not done.
At the end, the estimated angles are passed to `DisplayKinectAudioAngles` for visualization. It can receive the angles from up to two `KinectAudioMUSIC` instances and draws the DOAs as colored lines.

5 Experimental Results

5.1 Evaluation of Kinect Microphone Array

At first, the frequency response of the microphone array of a Kinect was investigated. This is needed to correctly interpret recordings made by the Kinects and to determine the impact of spectral aliasing.
A sound with increasing frequency was generated with Audacity[8]. The frequency increased linearly from 5 Hz to 20 kHz over 150 s (= 2:30 minutes). This sound was played by a speaker and recorded by a Kinect with the libfreenect WAV file recorder. The distance between speaker and Kinect was about 0.5 m.

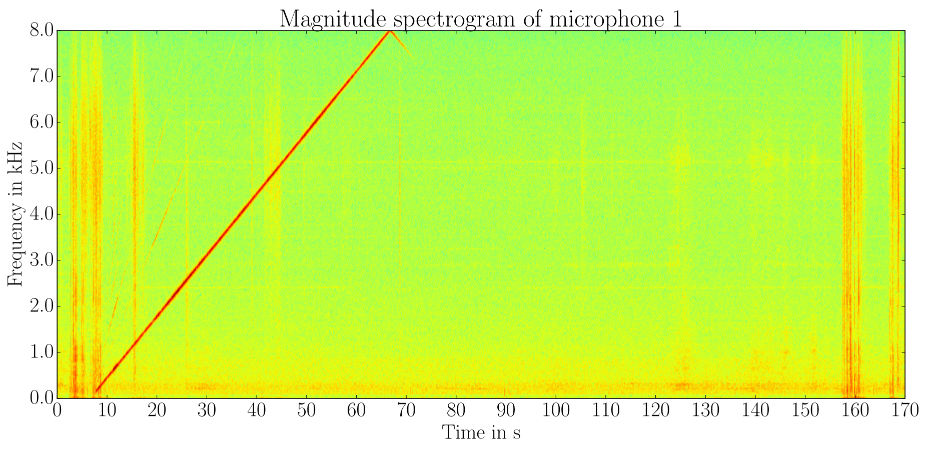


Figure 5.1: Spectrogram of a recording from the first microphone. The dominant red line is the increasing frequency of the source, until it reaches the Nyquist frequency.

The magnitude spectrogram in figure 5.1 (page 43) looks almost as expected. The frequency increases linearly, until it reaches the Nyquist frequency at about 65 s. After that, some aliasing occurs. At about 72 s the anti-aliasing mechanisms of the Kinect start to work. Therefore, frequencies up to 7 kHz seem to be detected correctly. The activity at the beginning and the end of the recordings are just noise made by myself, e.g. when I started or stopped the recording.
According to [80], at least the response of two microphones connected to the same ADC are simultaneously sampled. In addition, it is plausible that both ADC chips are synchronized, so it is reasonable to assume that the responses of all microphones are simultaneously sampled.
The recording was not done in an anechoic chamber, so reflections might influence it. However, this influence should be negligible, since the registered frequencies conform with the generated ones.

[8]http://www.audacityteam.org/

5.2 Traffic sound analysis

To learn more about the characteristics of traffic sounds, recordings were manually analysed. These recordings were made with two Kinects for [22] in 2013 and include data from lidar and radar sensors as baseline. The Kinects were placed on the cars roof like in figure 5.2 (page 44).

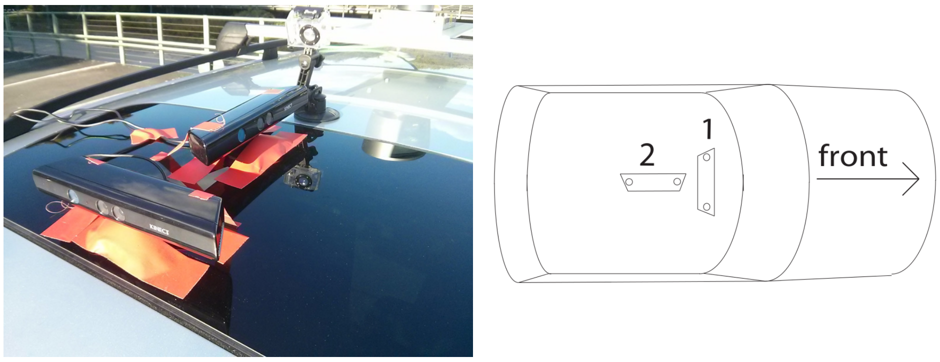


Figure 5.2: Image (left) and illustration of placement (right) of the Kinects [22].

At first, the lidar points clouds were analyzed to determine when vehicles passed. After that, I listened to the sound recordings to classify different sounds and associated them with the point clouds, so samples of different situations were obtained. In the end, this knowledge was used to get information about the spectral composition, by spectrograms, and the temporal distributions of sample values, by histograms.
A recording with moderate traffic was used because it contains different situations. The spectrogram is given in figure 5.3 (page 44).

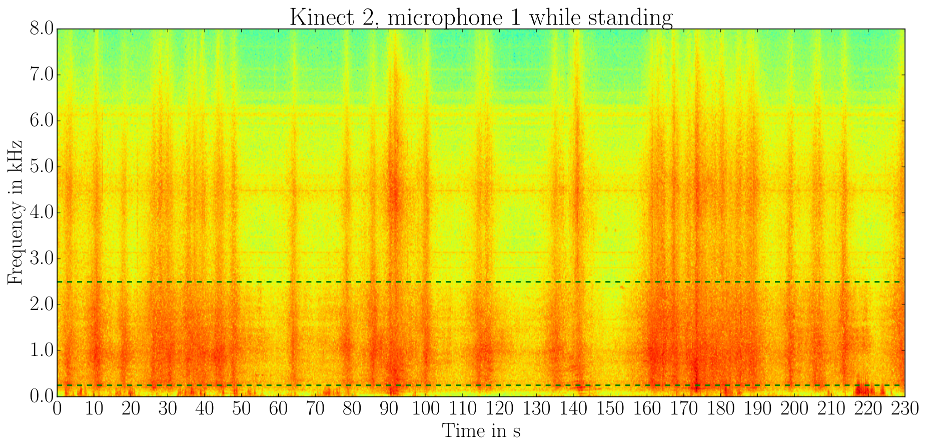


Figure 5.3: Spectrogram of a sound recording from microphone 1 of Kinect 2 of moderate traffic. The band within the green dashed lines got the most activity.

At about 3.2 kHz, 4.5 kHz and 6.2 kHz vertical lines are visible, indicating constant intensity in the band around these frequencies. I could not determine where they come from. The spectrogram of Kinect 1 did not show such lines, so they probably belong to the characteristics of the array of Kinect 2. Another attempt to explain this behaviour might be that at the position of Kinect 2 (in the middle of the rooftop window) the sound of the car itself (e.g. caused by the engine) and the sound of its interior got a bigger impact. However, this behaviour is only visible in this specific recording.

5.2.1 Noise

Section 3.2 (page 29) assumes that background noise follows a normal distribution in time domain. To verify this assumption, some quiet periods were analysed, e.g. the period from 147 s to 151 s. In figure 5.4 (page 45) the spectrogram and the histogram of sample values are shown. For comparison the PDF of a normal distribution is overlaid. Its parameters, mean and variance, are determined by the `stats.norm.fit`[9] function.

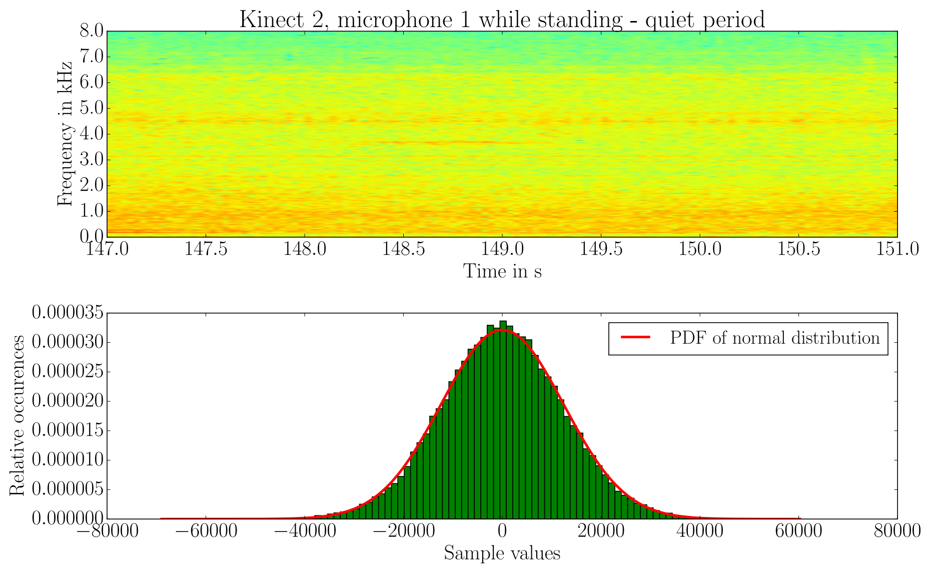


Figure 5.4: Spectrogram (top) and normalized histogram of sample values (bottom) of the period from 147 s to 151 s. The red line is the PDF of a normal distribution with mean and variance determined by data of this period.

According to the spectrogram in figure 5.4 (page 45), the power is not really uniformly distributed over the frequencies. There is more activity in the band from 0 kHz to 1 kHz, the band from 6.5 kHz to 8 kHz is less active.

[9] http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html

However, the histogram seems to form a normal distribution, at least the red curve fits quite well. Therefore, the assumption that background noise follows a normal distribution can be considered applicable.

5.2.2 Wind

In figure 5.3 (page 44) you may see that the band from 0 kHz to about 0.2 kHz does not behave like the others. The intensity in this band changes independently from whether vehicles pass or not. By listening, activity in this frequency band could be classified as wind noise. In figure 5.5 (page 46) the spectrogram of a period with dominant wind noise is shown.

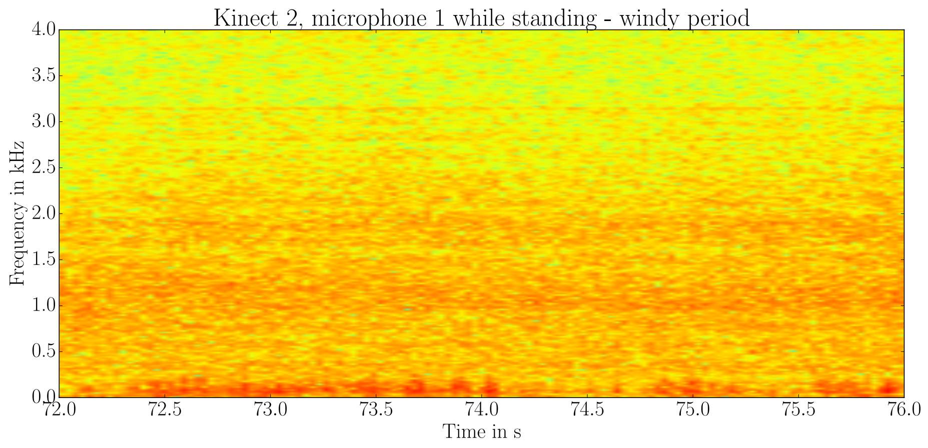


Figure 5.5: Spectrogram of a windy period from 72 s to 76 s. The spectrogram is cropped at 4 kHz to make the band from 0 kHz to 0.2 kHz more visible.

The wind sound is dominant while driving because of the air stream. The influence of wind is clearly illustrated in figure 5.6 (page 47).
By using a bandpass filter, e.g. FIR filter, wind noise can be almost removed, as long as it is not too dominant.

5.2.3 Passing Vehicles

The sound characteristics of passing vehicles are most interesting, since these are the objects that should be localized. These situations were analyzed:

- one vehicle passing on the own lane
- one vehicle passing on the opposite lane
- multiple vehicles passing

Spectrogram analysis showed that it does not matter on which lane one vehicle passes. Multiple vehicles increase the overall intensity. I could not

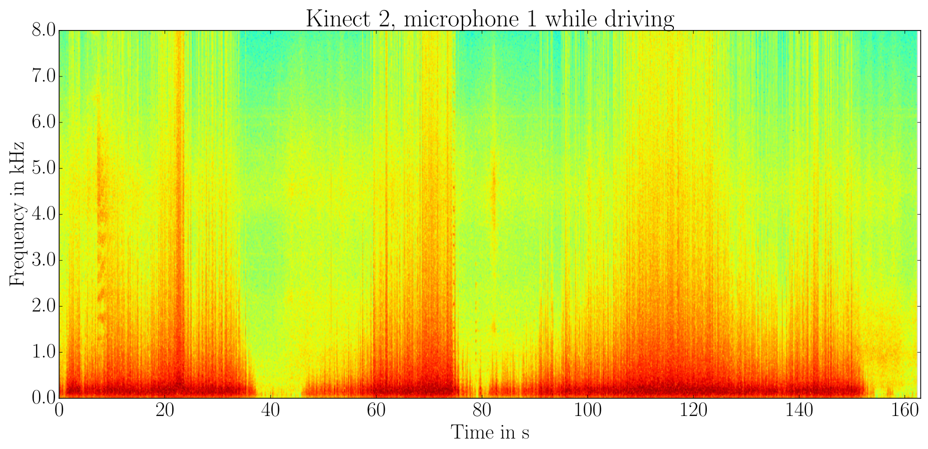


Figure 5.6: Spectrogram while driving. The car stands, e.g. at traffic lights, in the periods with less activity. Due to the air stream, the band from 0 kHz to 0.2 kHz is dominant.

determine a relation between vehicle type (car, truck or vehicle) and occupied frequency band.
Whenever one or more vehicles pass the activity in the spectrogram increases over almost all frequencies, but not all frequencies are affected the same way. The band from about 6.5 kHz to 8 kHz is less active than the others. The band from about 0.2 kHz to 2.5 kHz (between the green dashed lines in figure 5.3 (page 44)) is the most active one. With this knowledge in mind, I decided to use this band for DOA estimation.
Another aspect which was investigated, is the distribution of sample values when a car passes. As you can see in figure 5.7 (page 48), a normal distribution (red curve) does not fit as well as for the noise in figure 5.4 (page 45). Recordings of passing cars were tested against several distributions, which look similar to a normal distribution. The most promising candidate was the *logistic distribution* (blue curve). The parameter determination for the logistic distribution was done by `stats.logistic.fit`[10].
Neither the normal distribution nor the logistic distribution seem to fit well. For more significant results *Q-Q plots* (quantile-quantile plot) [88] were used. They are shown in figure 5.8 (page 49). These plots were generated by `stats.probplot`[11].
The Q-Q plots show that a passing car rather follows normal distribution than a logistic distribution.

[10]`http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.logistic.html`
[11]`http://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.probplot.html`

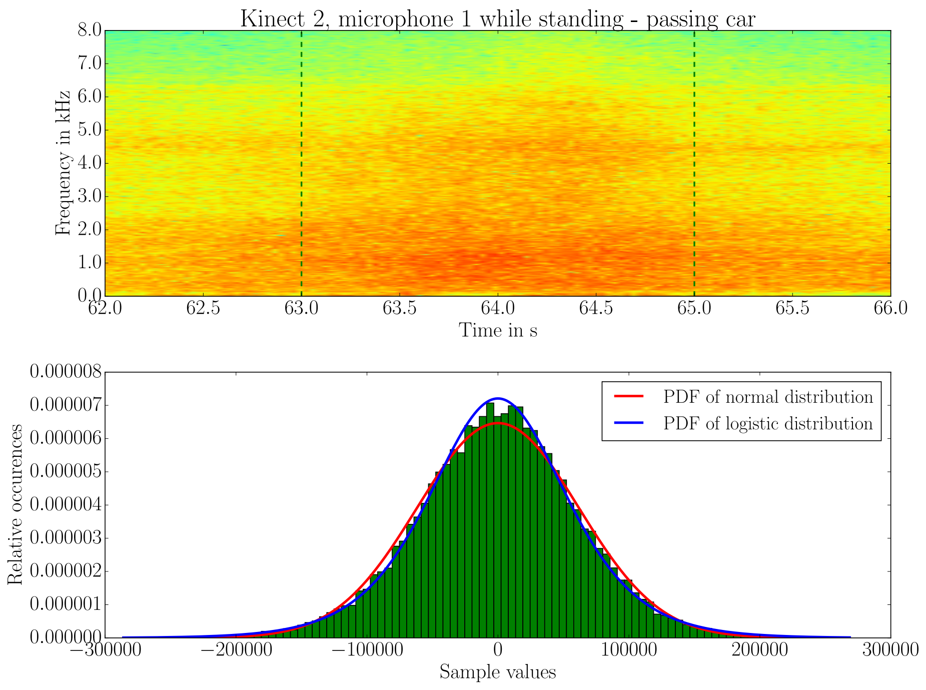


Figure 5.7: Spectrogram (top) and normalized histogram (bottom) of a passing car in the period from 62 s to 66 s. The period within the green dashed lines is used for the histogram. The PDFs of a normal distribution (red) and a logistic distribution (blue) are overlaid.

5.2.4 Sirens

Even more interesting traffic sounds are sirens. Because of regulations in many countries, cars have to give way to vehicles with activated siren. According to German law [89], sirens have to consist of a series of sounds, but no specific frequencies are mentioned. In DIN 14610 the range of the fundamental frequencies is defined and should be between 360 Hz and 630 Hz [90]. It also defines different frequency subranges for rural and urban environments.
In figure 5.9 (page 49) the spectrogram of a siren of a German ambulance is shown. Its fundamental frequencies are about 400 Hz and 500 Hz, many harmonics are visible. The sound of the siren is pretty dominant, even with the wind at about 2 s.

5.3 MUSIC with simulated Data

To investigate the behaviour of the MUSIC algorithms in a controlled environment, they were implemented in Python and tested with artificially generated array responses, generated by `pySoundToolbox` [86]. After that,

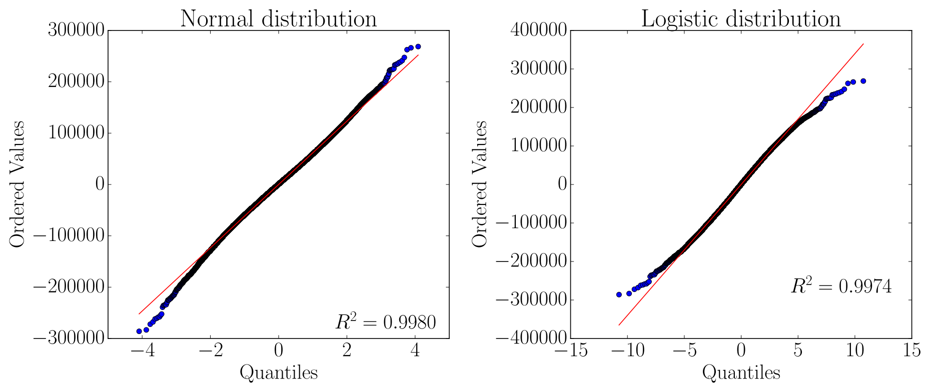


Figure 5.8: Q-Q plots to test the sample value distribution of a passing car against a normal distribution (left) and a logistic distribution (right). The better the blue points fit to the red line, the better the sample value distribution fits to the assumed distribution.

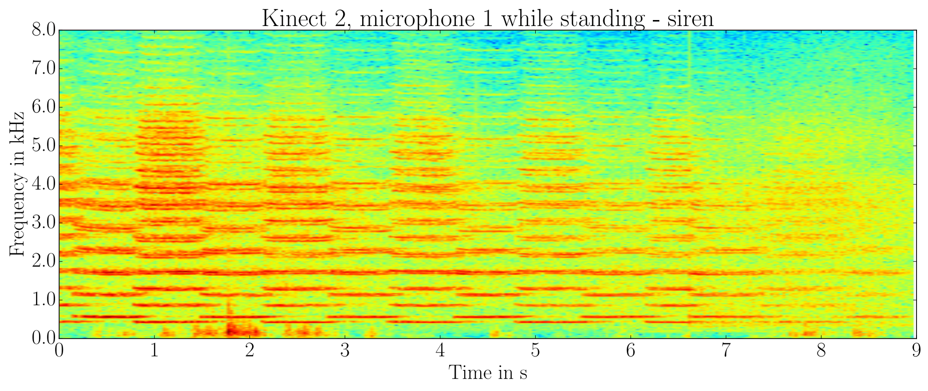


Figure 5.9: Spectrogram of a German ambulance siren. The alternation of frequencies is clearly visible and multiple harmonics are present.

the plots of the MUSIC spectrums were empirically evaluated.
Simulations were done under different conditions, e.g. different noise levels. The geometry of the simulated array is the same like the composition of two Kinect positioned the same way like in section 5.2 (page 44). The results were evaluated in a qualitative way.
This section presents the results of the most interesting simulations.

5.3.1 Time Domain MUSIC

Influence of Noise and Recording Duration

The first simulations should determine the robustness of MUSIC in the presence of noise. Therefor the array response of a pure sound, sent by one source, was generated. The SNRs, duration (i.e. the number of samples

to generate the covariance matrix) and the frequency of the source have been varied. Due to the stochastic behaviour of the noise, simulations were repeated several times.
Figure 5.10 (page 50) shows the MUSIC spectrums for a duration of 0.1 s and a source with a frequency of 600 Hz. For this recording time and frequency up to an SNR of -7 dB peaks were quite stable. Lower SNRs result in slightly wrong, unstable or undetectable peaks.

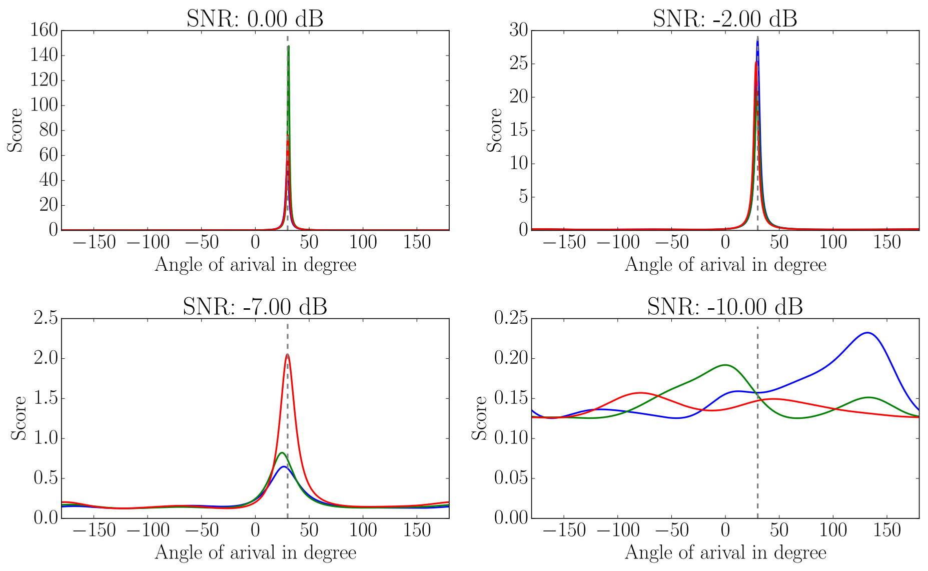


Figure 5.10: Multiple time domain MUSIC spectrums with different SNRs. Each color represents the spectrum of one trial. The source is placed at 30° (marked by the gray dashed line) and sends at 600 Hz. Duration was 0.1 s. Since low SNRs result in low scores, the y-axis is scaled individually, so peaks are visible.

The results of these simulations can be summarized as follows:

- The lower the SNR, the worse the quality of DOA estimation.
- The more samples are available, the more robust is the DOA estimation, as long as the source angle remains the same.
- For low source frequencies more samples are needed for robust DOA estimation.

Impact of Doppler effect

Another interesting aspect is the influence of the Doppler effect (section 2.2.3 (page 9)), since sources typically move and time domain MUSIC needs to know the sources frequency. Therefor the shifted frequency f' of a source sending at frequency f and relative velocity $v_s = 60\,\frac{\text{km}}{\text{h}} \approx 16.7\,\frac{\text{m}}{\text{s}}$ has been calculated with equation 2.6 (page 10). After that, the array response, with f' as source frequency, and an SNR of 1 dB has been generated. Time domain MUSIC with f as assumed source frequency was applied. An example of such spectrums is given in figure 5.11 (page 51).

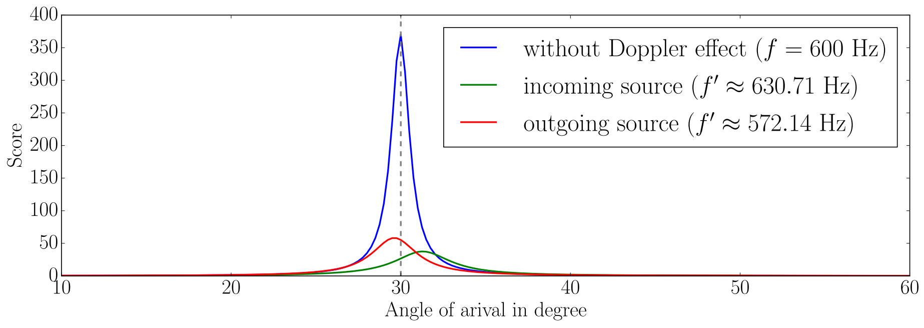


Figure 5.11: Example of MUSIC spectrums without the impact of the Doppler effect (blue), with an incoming source (green), i.e. positive frequency shift, and an outgoing source (red). SNR is 1 dB, duration is 0.1 s, the source is placed at 30° and sends at 600 Hz. Spectrum is cropped for better visibility.

The MUSIC spectrums in figure 5.12 (page 52) show that it is impossible to determine the direction of such sources with time domain MUSIC. The Doppler effect shifts the estimated DOA in a similar way it shifts the frequency of the received signal. When source and receiver move away from each other the estimate is left from the actual source. The spectrum is affected analogously when the source moves towards the receiver. So, the faster the source, the bigger the error of the estimate.
These experiments did not simulate a moving source, they rather simulated the impact of a wrong source frequency guess. However, these results show that without knowledge of the sender frequency, time domain MUSIC tends to be erroneous. It is also unsuitable to sources which change their frequency over time, e.g. sirens.

White Noise Sources

At the end, time domain MUSIC with a source which emits white Gaussian noise was tested. This was especially interesting because passing cars emit similar sounds when passing (section 5.2.3 (page 46)). The array response for one source, which emits temporal normal distributed sound, has been generated. The source was placed at 30°, the duration was 1 s and no background noise was present. MUSIC with different frequency assumptions was applied. The MUSIC spectrums in figure 5.12 (page 52) show that it is impossible to determine the direction of such sources with time domain MUSIC.
Time domain MUSIC correctly determines the direction of one source, as

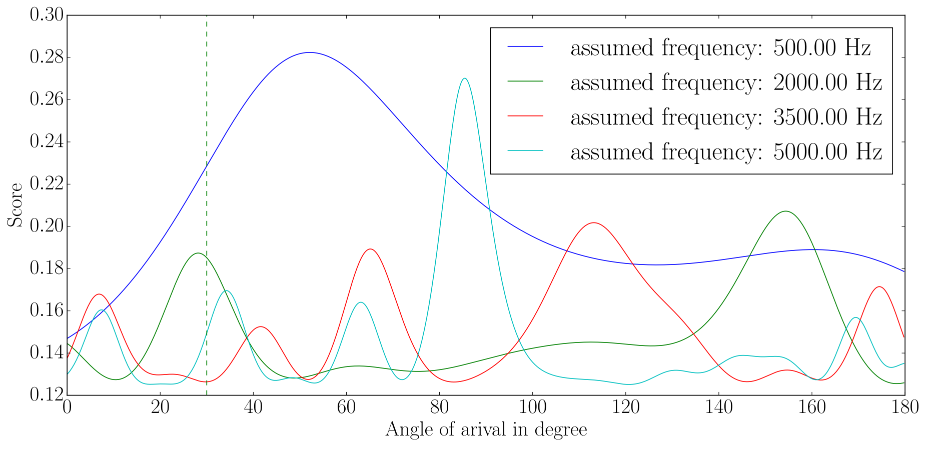


Figure 5.12: MUSIC spectrums with different frequency assumptions. The source sent white noise and has been placed at 30° (green dashed line), no background noise has been simulated. Duration of the record was 1 s. As you may see, DOA estimation is not possible.

long as the source sends a pure tone at a known frequency. Under this conditions even quite low SNRs provide good results. The results get worse when the frequency changes or a wrong frequency is assumed. So, the direction of multiple sources cannot be correctly determined, since they have to send on different frequencies. When it comes to stochastic distributed sounds, time domain MUSIC becomes unsuitable for DOA estimation.

5.3.2 Frequency Domain MUSIC

Influence of Parameters

Since traffic sounds have most activity in the band from 0.2 kHz to 2.5 kHz, primarily the Fourier coefficients which belong to frequencies within this band were used. Similar to section 5.3.1 (page 49), several parameters were varied, to investigate their impact on DOA estimation. The investigated parameter were:

- Duration of recording
- SNR
- Number of samples used to FFT
- Number of sources

Due to the stochastic nature of the signals and the sources, multiple trials were run.

In contrast to prior simulations, two or more sources, which emitted white Gaussian noise, were simulated. An example is given in figure 5.13 (page 53). The results are similar to the results in section 5.3.1 (page 49):

- The longer the duration, the better DOA estimation.
- Low SNRs result in bad DOA estimation.

However, frequency domain MUSIC is more sensitive to noise, e.g. the SNR has to be at about 10 dB for a useful estimation of the directions of two sources.

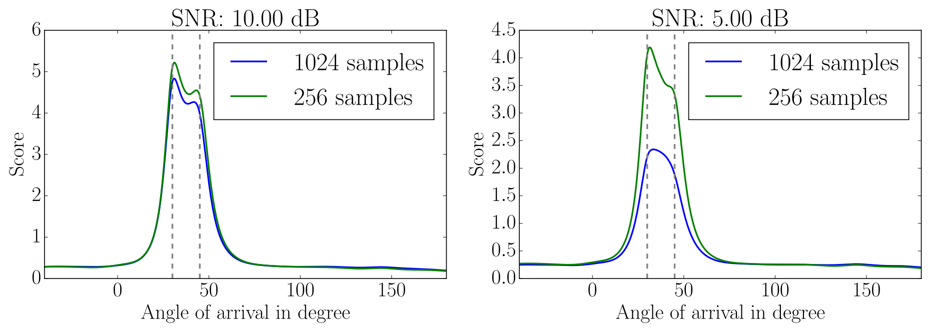


Figure 5.13: Spectrum of frequency domain MUSIC with different SNRs. Duration was 0.2 s, two sources were present, one at 30° and one at 45° (marked by the gray dashed lines). Each color represents another number of samples used for FFT. For higher SNRs the number of samples for FFT seems to be neglected, but for low SNRs FFT with less samples provide better results.

Additionally, the following results have been obtained:

- More sources require higher SNRs for correct DOA estimation.
- Higher frequencies make spacial separation of source directions more reliable, i.e. sources can be closer to each other and will also be recognized as different sources.
- If too high frequencies are used, MUSIC tends to spacial aliasing (also depends on array geometry).
- FFT with less samples tend to provide better results.

Especially, the last result was surprising. I expected that more samples for the FFT will result in better estimation because the frequency resolution is better. Probably, multiple DFTs are needed to compute a, in terms of stochastic, better estimate of the cross-spectral matrix. Therefore, FFT with less samples require less duration and are more robust to noise.

Wrong Number of Sources
Another interesting aspect was the behaviour of IFB MUSIC, when a wrong number of present sources is assumed. In figure 5.14 (page 54) you can see that a wrong assumption of the source number decreases accuracy of DOA estimation and makes spacial separation of sources sometimes impossible.

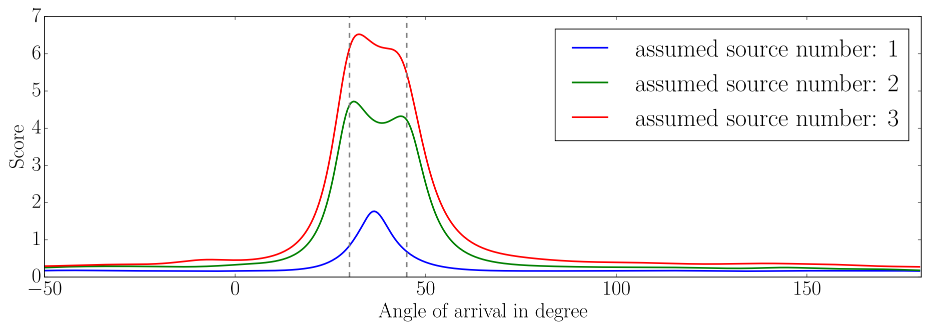


Figure 5.14: MUSIC spectrums of two sources, sending white noise, placed at 30° and 45° (marked by gray dashed lines). Duration is 0.2 s, SNR is 20 dB. Each color is the spacial spectrum for one assumption of the source number. The green curve shows the MUSIC spectrum with correct number of sources.

Even though I expected wrong DOA estimations, I was surprised that an assumption of too few sources results in a peak at an angle between the two actual angles. I expected that at least the direction of one source will be estimated nearly correct. However, if one of the sources is dominant, this peak moves towards the angle of the dominant source.

Sirens
At last, the estimation of direction of a siren has been investigated. Therefor recordings of German firefighter sirens from [91] were used. Array responses with different SNRs has been generated. These array responses were used for DOA estimation. An example of the quality of DOA estimation of a siren can be seen in figure 5.15 (page 55).
The estimation of the direction of a siren works pretty well in simulations. In contrast to source which emit white noise, the direction is also found when the SNR is relatively low.

5.4 Real Environments

5.4.1 Empiric evaluation

At first, the behaviour of the MUSIC implementation was empirically evaluated. Therefor the estimated directions of sound sources has been compared with lidar point clouds in the log player. The estimated directions were visualized by colored lines. The logs contained moderate traffic and

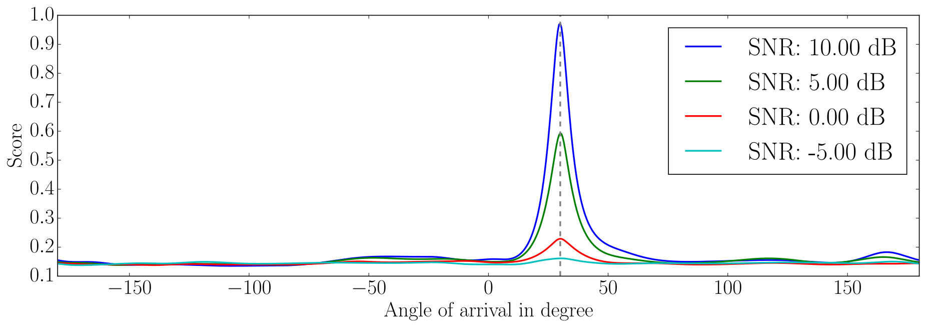


Figure 5.15: MUSIC spectrums of one siren placed at 30° (gray dashed line). Each color is a MUSIC spectrum with another SNR.

a standing car because wind sound being too dominant while driving, no matter whether it was filtered.

Kinects as one Array

In contrast to section 4.2.3 (page 40), both Kinects were interpreted as one array with 'T' shape, at first. Using this array geometry direction estimation was unreliable, as long as vehicles were behind Kinect 1, which is aligned orthogonally to the street. Additionally, an unexpected symmetry between the first and the second direction has been observed. Variations of microphone placement and the threshold γ for equation 3.11 (page 36) did not significantly change this behaviour.

Kinects as independent Arrays

The same logs like in the previous experiment were used. To avoid spacial aliasing, only angles within the range from 0° to 180° were used to search for peaks.
It turned out that DOA estimation of Kinect 1 did not behave as expected. The estimated directions were not symmetric along the axis the microphones are placed along, e.g. an actual source angle of $-20°$ did not result in an estimated angle of 20°. Instead, the estimated directions jumped in a 'chaotic' manner. Maybe this behaviour can be explained by the directivity pattern of a single microphone (see figure 4.3 (page 38)). Due to the lack of backward directivity, the impact of background noise is higher, making DOA estimation more error prone. However, as long as the sound source was in front of the Kinect, DOA estimation was quite good. So, Kinect 2 provided relatively reliable estimations of directions.
Situations with one passing vehicle were uncritical. They were tracked by their sound emission, as long as they were not too far away and the actual angles were not too flat.

Two vehicles were tracked simultaneously, but less reliably than for one vehicle. Tracking two vehicles worked best, when they drove in the same direction. When they moved in opposite directions and got too close, they were recognized as one sound source.
More than two vehicles at a time resulted in jumping directions, but they pointed to a vehicle, almost always. However, this could be also luck, since the vehicles almost completely covered the side.
The directions of sirens were reliably tracked. Typically, the direction with the largest score in the MUSIC spectrum was the direction of the siren. In contrast to passing cars, the acoustic reflection of the siren was detected first. When the ambulances were near enough, their direction was estimated directly.

Threshold
Section 3.3.2 (page 36) introduced a simple approach to determine the number of eigenvectors to use to form the noise matrix. Therefor a threshold γ (equation 3.11 (page 36)) is needed, which had to be estimated. So, different values for γ has been tried and the behaviour of DOA estimation has been observed.
Too high values as well as too low values for γ resulted in unstable estimation of direction. The best results were observed with $\gamma \approx 0.75$.

5.4.2 Quantitative Evaluation

Due to the problems with sound sources behind a Kinect, only the direction estimations of Kinect 2 were used for quantitative evaluation. The same logs like in section 5.4.1 (page 54) were used.
The object detection module of the log player has been used as baseline. It utilizes the output of the lidar and multiple radar sensor and fusions them to discrete objects. Additionally, it computes the velocity each detected object. These informations were used to automatically match estimated directions to moving objects. To reduce mismatches, e.g. pedestrians, only objects on the left hand side, relative to the cars orientation, were used.
The DOA estimations were made with an FFT with 256 samples, 10 DFTs were used to generate the cross-spectral matrix and the threshold was $\gamma = 0.75$. The three highest peaks in the MUSIC spectrum were picked.

Angle Error of Passing Vehicles
An interesting aspect is the angle error between the actual object angle and the angle estimated by MUSIC. The angle error is the difference between the actual object angle and the angle of the estimated direction. The actual object angle is the angle between the line, which goes through the center of the car and the center of the object, and the axis of the microphones. Given an angle provided by MUSIC, the moving object with the smallest angle

error has been chosen. The angle error as function of distance is shown in figure 5.16 (page 57).

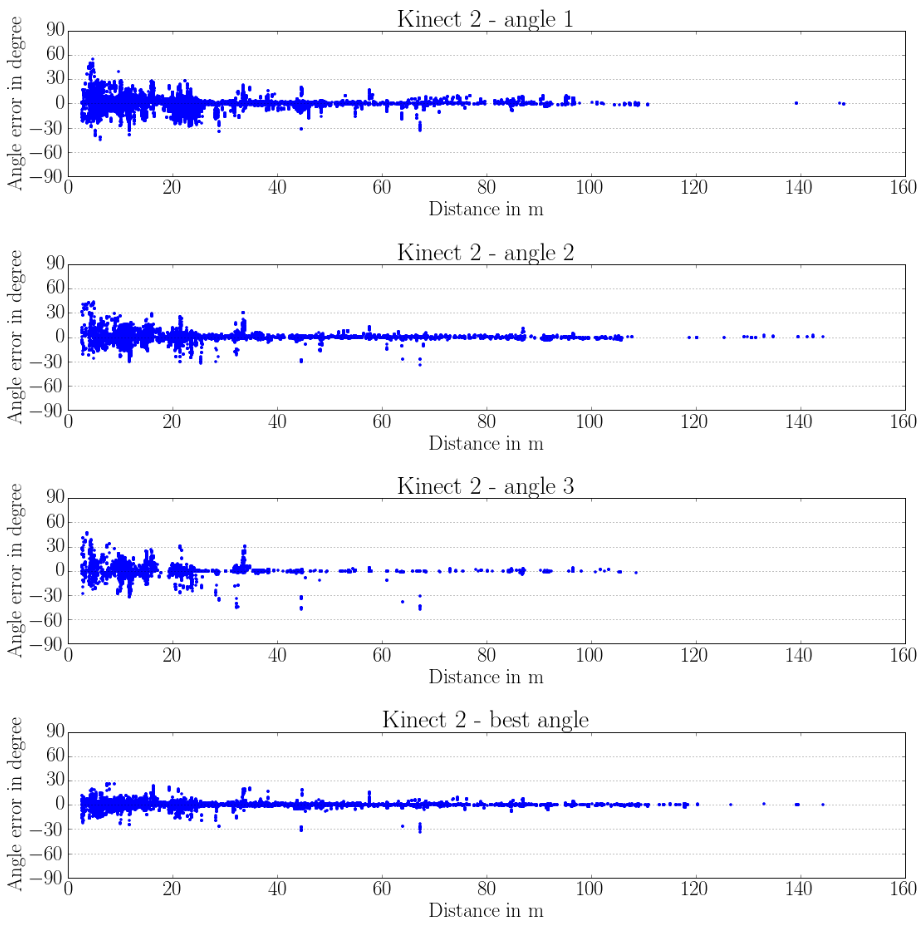


Figure 5.16: Angle error as function of object distance. Angle 1 (first) is the estimated angle with the highest MUSIC score, angle 2 (second) is the angle with the second highest MUSIC score and angle 3 (third) is the angle with the third highest MUSIC score. The best of the three angles is chosen in the fourth plot.

As you may see, the angle error does not exceed 60° or −60°, respectively. Most angle errors are within the interval from −30° to 60°.
In figure 5.17 (page 58) the mean absolute angle errors decrease with object distance. The maximum mean angle error was about 9°, but the maximum angle error was about 60°. The standard deviations, illustrated by error bars, show the large fluctuation of the absolute angle errors and are almost as large as the mean itself.
These results differ from my expectations and the results of my empirical observations. Surprisingly, the highest angle errors were detected for close objects. I expected smaller errors for near objects, since their sound emis-

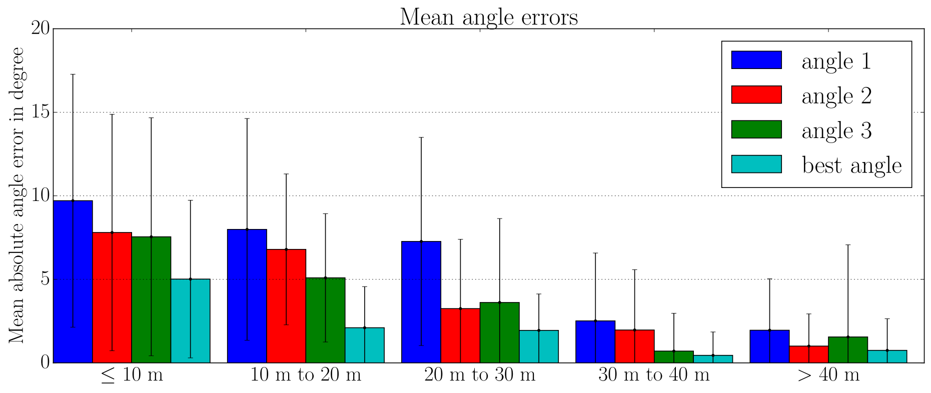


Figure 5.17: Mean absolute angle errors with standard deviation (black error bars) for several distances.

sions should be more intensive. As mentioned in section 5.2.3 (page 46), most sounds emitted by a passing vehicle come from the friction between tire and road. So, the estimated directions did not directly point to the center of the objects. They pointed to the tires, which are about half the objects size away from the center. When a vehicle was close, this displacement resulted in a higher angle error.
However, these numbers should be used with caution. Probably, some angle errors were the result mismatches, since the object detection did not classify the objects. So, it is possible that the angle error to a passing bicyclist could be used because he was the moving object which was closest to the estimated direction.

Sirens
Recordings of sirens of German ambulances were made near a firefighter station. The ambulances came from behind and turned at the crossroads right in front of the car. So, the provided angles of both Kinects were evaluated. Due to the dominance of sirens, only the first angle, which is the angle with with highest peak in the MUSIC spectrum, was used. Figure 5.18 (page 59) shows the angle error over time in a recording with present siren. Kinect 2 tracked the reflection of the siren until about 5 s. However, at this point the ambulance was quite close, so the angle error remained high. At about 7 s the estimated direction started jumping, indicating that the border of the detection range of Kinect 2 has been reached. Due to the turn, the angle errors of Kinect 2 from 8 s to 13 s belong to other objects.
At about 7 s Kinect 1 started to track the siren, but the angle error did not significantly change. At about 11 s object detection failed because of the turn.
Generally, these results were disappointing. The promising results from

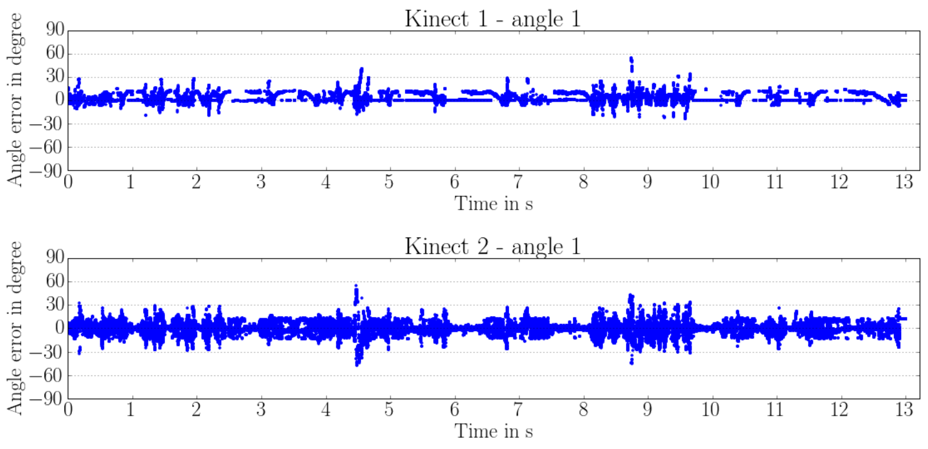


Figure 5.18: Angle error of the first estimated angle of Kinect 1 (top) and Kinect 2 (bottom) over time. The ambulance turned right from 8 s to 11 s. As you may see, the siren did not significantly influence the angle error.

empirical observations were not confirmed by quantitative results. Therefor more experiments, maybe in a more controlled environment, are needed for significant results.

5.5 Runtime Performance

Generating steering vectors on the fly made the implementation unusable. Generating them at the start and saving them dramatically reduced the runtime.
Another aspect which affected the runtime, was the number of Fourier coefficients used to compute the MUSIC spectrums. There are two ways to reduce the number of Fourier coefficients:

1. Reduce the number of samples for the FFT.

2. Only use Fourier coefficients which belong to a limited frequency range.

The first option reduces the number of MUSIC spectrums to compute, but also reduces frequency resolution. However, simulations showed that less samples for FFT may improve DOA estimation.
A simple runtime was test made. An FFT with 256 samples has been applied and 35 Fourier coefficients, resulting in 35 MUSIC spectrums, has been used. The vector operations were not parallelized. With an Intel Core i5-2540M with two cores at 2.6 GHz and 16 GiB RAM about 25 s were needed to replay a log with a duration of 10 s.
I did not exhaustively investigate runtime performance, but the implementation clearly was far from real-time. So, major improvements have to be made.

6 Conclusion

6.1 Discussion

Estimating the direction of vehicles by sound emission is difficult for several reasons. Traffic is inherently noisy and many sound sources are present. Many parts of a vehicle emit different sounds. Urban areas are highly reverberant because of dense building development. Additionally, other objects may shadow sound. Variations in array geometry and in directivity patterns of the microphones make DOA estimation even more challenging.
However, under certain conditions, estimating the direction of traffic objects by their sound emission is possible.
In this thesis the behaviour of MUSIC under different conditions was investigated. Simulations showed that time domain MUSIC provides stable estimations, even with low SNR. However, since it requires a stable sender frequency, it is unsuitable for traffic environments.
In contrast to time domain MUSIC, IFB MUSIC does not rely on a known sender frequency. It was less robust to noise in simulations, but it successfully determined the direction of a simulated siren.
The results on real world data were not as good as expected. Generally, estimating the direction of traffic objects by acoustic data is not very precise. Especially, when vehicles are close, their axes may be detected as different sound sources. On the other hand, the presence of many vehicles make DOA estimation almost impossible. However, under certain conditions vehicles were reliably tracked by sound.
Empiric observations showed that sirens can be tracked, but quantitative evaluation did not provide sufficient results. For reliable conclusions, more experiments are needed. For such experiments, a more controlled environment should be preferred.

6.2 Future Work

The introduced implementation is clearly not mature. It has some problems with precision and it is pretty slow. So, in this section some starting points for future work and improvements are proposed.

6.2.1 Hardware

One way to improve quality of DOA estimation is improving the hardware. To reduce the impact of wind noise, microphone windscreens should be utilized. Blimps are required, when the car is moving. However, the influence of windscreens has to be investigated. Maybe two separate arrays, one with blimps for driving and one with simple windscreens for standing, might be useful.

The Kinects sampling rate is relatively low, e.g. the sound of a source placed at 0° just needs 5 samples from the outer right microphone to the outer left microphone. A higher sampling rate could improve DOA estimation, but would also increase computational cost, since more samples for FFT will be needed to detect lower frequencies.
Using an array with more microphones would improve stability of DOA estimation and would raise the number of detectable sources. Arrays with certain geometries help to avoid spacial aliasing, may improve the quality of DOA estimation and enable the usage of more efficient algorithms, e.g. ESPRIT [61]. Additionally, more knowledge about the frequency response and directivity pattern would be useful.

6.2.2 Software

In section 3.3.1 (page 33) a simple model for steering vectors has been described. Better steering vectors, e.g. steering vectors which include the decrease of intensity or the directivity pattern of the microphones, may improve quality of DOA estimation. Obtaining steering vectors by calibration of the array may be more reliable than modeling them.
In this thesis classical spectrum MUSIC has been implemented and evaluated. It worked pretty well, but better performance and less computational expense would be preferable. More reliable methods to determine the number of noise eigenvectors, e.g. independent component analysis [75], would improve stability of DOA estimation. Additionally, other spectrum merging techniques may lower the computational expense.
Different variations of MUSIC may improve accuracy and runtime.
Smooth-MUSIC [65] overcomes the assumption that all sources have to be uncorrelated. Therefor the array has to consist of L overlapping subarrays, i.e. subarrays share at least on microphone. This allows the computation of a 'smoothed' covariance matrix, which allows the localization of up to $L-1$ signals.
Using crystal-shaped arrays enable the usage of *Crystal-MUSIC* [70]. According to [70], accuracy of DOA estimation would be improved compared to classical MUSIC.
No special array geometry is required by *Self-Consistent MUSIC (SC-MUSIC)* [69]. It utilizes *Recursively Applied and Projected (RAP) MUSIC* [92], which had originally been designed to locate sources in EEG, to improve sound source localization. The results in [69] showed that also sources close to each other and reverberations can be located.
ESPRIT [61] is less computationally expensive, but it requires the array to consist of N doublets. A doublet is a subarray of two microphones, so $2N$ microphones are needed. The doublets have to be identical, i.e. the microphones of a doublet have to have the same distance to each other, and they have to be parallelly aligned. To reduce the number of microphones,

doublets can overlap. With these restrictions in array geometry, up to $N-1$ sound sources can be localized with less computational expense.

References

[1] U. D. o. T. National Highway Traffic Safety Administration, “Minimum Sound Requirements for Hybrid and Electric Vehicles,” Tech. Rep. January, 2013.

[2] C. Wild, “Giant WWI ”war tubas” were pretty much exactly what they sound like.” [Online]. Available: http://mashable.com/2015/02/16/war-tubas-radar-wwi/#om4DopHLZkqS

[3] L. S. Howeth, *Maximum likelihood localization of multiple sources by alternating projection*, 1963. [Online]. Available: http://earlyradiohistory.us/1963hw39.htm

[4] C. Z. C. Zhang, Z. Z. Z. Zhang, and D. Florencio, “Maximum Likelihood Sound Source Localization for Multiple Directional Microphones,” *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP ’07*, vol. 1, 2007.

[5] C. Zhang, D. Florencio, D. E. Ba, and Z. Zhang, “Maximum likelihood sound source localization and beamforming for directional microphone arrays in distributed meetings,” in *IEEE Transactions on Multimedia*, vol. 10, no. 3, 2008, pp. 538–548.

[6] J.-m. Valin, J. Rouat, and L. Dominic, “Robust Sound Source Localization Using a Microphone Array on a Mobile Robot,” *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1228–1233, 2003. [Online]. Available: http://jmvalin.ca/papers/iros.pdf

[7] C. T. Ishi, O. Chatot, H. Ishiguro, and N. Hagita, “Evaluation of a MUSIC-based Real-time Sound Localization of Multiple Sound Sources in Real Noisy Environments,” *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2027 – 2032, 2009.

[8] X. Li, M. Shen, W. Wang, and H. Liu, “Real-time Sound source localization for a mobile robot based on the guided spectral-temporal position method,” *International Journal of Advanced Robotic Systems*, vol. 9, pp. 1–8, 2012. [Online]. Available: http://cdn.intechopen.com/pdfs-wm/39306.pdf

[9] J.-S. Hu, C.-Y. Chan, C.-K. Wang, M.-T. Lee, and C.-Y. Kuo, “Simultaneous Localization of a Mobile Robot and Multiple Sound Sources Using a Microphone Array,” *Advanced Robotics*, vol. 25, no. 1-2, pp. 135–152, 2011. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1163/016918610X538525

[10] D. Farrell, "Using Acoustic Beamforming for Pass-By Noise Source Detection," p. 15, 2013.

[11] A. Martens, J. Wedemann, N. Meunier, and A. Leclere, "High Speed Train Noise - Sound Source Localization At Fast Passing Trains," *Deutsche Balm AG, Sociedad Espanola de Acustica, SEA.*, 2009. [Online]. Available: http://webistem.com/acoustics2008/acoustics2008/cd1/data/fa2002-sevilla/forumacusticum/archivos/noi03007.pdf

[12] A. Meyer and D. Döbler, "Noise source localization within a car interior using 3D-microphone arrays," *Proceedings of the BeBeC*, pp. 1–7, 2006. [Online]. Available: http://bebec.eu/Downloads/BeBeC2006/Papers/BeBeC-2006-17_Meyer_Doebler.pdf

[13] H. J. von der Burchard, "Akustische Kamera," 2014. [Online]. Available: http://www.3sat.de/page/?source=/ard/wissenaktuell/175742/index.html

[14] S.-w. Park and J. Trevino, "Automatic Detection of Emergency Vehicles for Hearing Impaired Drivers," *DSPS Fest 2000 Proceedings*, no. 361, 2000. [Online]. Available: http://www.ti.com/sc/docs/general/dsp/festproceedings/fest2000/biomedical.htm

[15] H. Martin, "ASD - Automatic Siren Detection A mixed signal ASIC for Detection of Emergency Signals in general Traffic Situations," p. 14610, 2009. [Online]. Available: http://www.date-conference.com/files/file/10-ubooth/ub-3.2-p09.pdf

[16] T. Miyazaki, Y. Kitazono, and M. Shimakawa, "Ambulance Siren Detector using FFT on dsPIC," *1st IEEE/IIAE International Conference on Intelligent Systems and Image Processing*, pp. 266–269, 2013. [Online]. Available: https://www2.ia-engineers.org/iciae/index.php/icisip/icisip2013/paper/viewFile/247/184

[17] Jens Schröder, S. Goetze, V. Grützmacher, and J. Anemüller, "Automatic Acoustic Siren Detection In Traffic Noise By Part-Basedmodels." [Online]. Available: http://www.ear-it.eu/sites/default/files/sgga13.pdf

[18] B. Fazenda, H. Atmoko, F. G. F. Gu, L. G. L. Guan, and A. Ball, "Acoustic based safety emergency vehicle detection for intelligent transport systems," *2009 Iccas-Sice*, no. 1, pp. 4250–4255, 2009. [Online]. Available: http://usir.salford.ac.uk/9390/3/Emergency_Vehicle_Detection_for_ITS-Fazenda-FINALv3.pdf

[19] N. J. Roseveare, "Wideband Direction-of-Arrival estimation methods for unattended acoustic sensors," Ph.D. dissertation, 2007.

[Online]. Available: http://homepages.uni-paderborn.de/nickr/files/Roseveare07-Thesis-WBDOAestimationForAcousticSensors.pdf

[20] P. Marmaroli, M. Carmona, J. M. Odobez, X. Falourd, and H. Lissek, "Observation of vehicle axles through pass-by noise: A strategy of microphone array design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1654–1664, 2013. [Online]. Available: http://www.idiap.ch/$\sim$odobez/publications/MarmaroliCarmonaOdobezFalourdLissek-IEEE-TITS-2013.pdf

[21] X. F. P. Marmaroli J.-M. Odobez and H. Lissek, "A Bimodal Sound Source Model for Vehicle Tracking in Traffic Monitoring," *European Signal Processing Conference (EUSIPCO), Barcelona*, no. Eusipco, pp. 1327–1331, 2011.

[22] H. Tadjine and D. Goehring, "Acoustic/Lidar Sensor Fusion for Car Tracking in City Traffic Scenarios," 2015. [Online]. Available: http://www.drgoehring.de/bib/tadjine15fastzero/tadjine15fastzero.pdf

[23] E. W. Weisstein, "Conjugate Transpose." [Online]. Available: http://mathworld.wolfram.com/ConjugateTranspose.html

[24] P. Baldwin, "Response to an Entire Signal," p. 41, 2006. [Online]. Available: http://situs.biomachina.org/hn06/talks/Baldwin/convolution_filters_new.pdf

[25] R. K. Rao Yarlagadda, "Chapter 2: Convolution and Correlation," in *Analog and Digital Signals and Systems*, 2010, pp. 39–69.

[26] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Computation of the Complex Fourier Series," *Mathematics of Computation*, vol. 19, p. 297, 1965. [Online]. Available: http://attach3.bdwm.net/attach/0Announce/groups/GROUP_3/MathTools/D6714701A/D69595345/M.1089260001.A/CooleyJ_AlgMCC.pdf

[27] Nipun, "Difference Between Transverse and Longitudinal Waves," 2015. [Online]. Available: http://pediaa.com/difference-between-transverse-and-longitudinal-waves/

[28] D. C. Giancoli, *Physics: Principles with Applications*, 2010, vol. 53.

[29] C. H. Hansen, "Fundamentals of Acoustics," 2010. [Online]. Available: http://www.who.int/occupational_health/publications/noise1.pdf

[30] A. C. Quillen, "Pure Tones and the Sine Wave," 2014. [Online]. Available: http://astro.pas.rochester.edu/$\sim$aquillen/phy103/Lectures/B_Sine.pdf

[31] M. Wickert, "Chapter 3: Spectrum Representation," in *ECE 2610 Signal and Systems*, 2011, vol. 3, pp. 1–46. [Online]. Available: http://www.eas.uccs.edu/$\sim$mwickert/ece2610/lecture_notes/ece2610_chap3.pdf

[32] ——, "Chapter 4: Sampling and Aliasing," in *ECE 2610 Signal and Systems*, 2013, vol. 1, pp. 11–18.

[33] P. White, "Sound Synthesis, Part 1," 1994. [Online]. Available: https://www.soundonsound.com/sos/1994_articles/feb94/soundsynthesis.html

[34] Inkwana, "Datei:DopplerEffectCars.svg," 2011. [Online]. Available: https://de.wikipedia.org/wiki/Datei:DopplerEffectCars.svg

[35] Gracey and Associates, "Sound Fields - Definitions, Terms, Units and Measurements." [Online]. Available: http://www.acoustic-glossary.co.uk/sound-fields.htm

[36] R. J. Mohr, "Mohr on Receiver Noise Characterization, Insights and Surprises."

[37] Rane, "Pro Audio Reference," 2016. [Online]. Available: http://www.rane.com/par-n.html#noise_color

[38] D. Johnson, "Signal-to-noise ratio," *Scholarpedia*, vol. 1, no. 12, p. 2088, dec 2006. [Online]. Available: http://www.scholarpedia.org/article/Signal-to-noise_ratio

[39] PCB Piezotronics, *Microphone Handbook*, 2013. [Online]. Available: www.pcb.com

[40] Mike Brookes, "11: Frequency Responses," 2015. [Online]. Available: http://www.ee.ic.ac.uk/hp/staff/dmb/courses/ccts1/01100_Freqresp.pdf

[41] D. M. Bujakovic, M. S. Andric, and M. D. Antonic, "Analysis of Sound Signals Using Wavelet Transform," 2010. [Online]. Available: PM:10892260

[42] C. E. Shannon, "Communication In The Presence Of Noise (Reprint)," *Proceedings of the IEEE*, vol. 86, no. 2, pp. 447–457, 1998. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=659497

[43] S. Roberts, "Lecture 7 - The Discrete Fourier Transform," pp. 82–96, 2000. [Online]. Available: http://www.robots.ox.ac.uk/$\sim$sjrob/Teaching/SP/l7.pdf

[44] J. Fessler, "The Discrete Fourier Transform," pp. 1–31, 2004. [Online]. Available: http://www.worldscientific.com/worldscibooks/10.1142/4610

[45] D. W. Jacobs, "Correlation and Convolution," p. 17, 2005. [Online]. Available: http://www.cs.umd.edu/$\sim$djacobs/CMSC426/Convolution.pdf

[46] M. Heideman, D. Johnson, and C. S. Burrus, "Gauss and the history of the Fast Fourier Transform," *IEEE Signal Processing Magazine*, vol. 1, no. 3, pp. 14–21, 1984. [Online]. Available: http://www.cis.rit.edu/class/simg716/Gauss_History_FFT.pdf

[47] G. V. Tcheslavski, "Lecture 04 : Analytic signal generation and Hilbert transformers," pp. 1–9, 2008.

[48] S. L. Marple, "Computing the Discrete-Time Analytic Signal via FFT," pp. 2600–2603, 1999. [Online]. Available: http://classes.engr.oregonstate.edu/eecs/winter2009/ece464/AnalyticSignal_Sept1999_SPTrans.pdf

[49] B. Roche, "Why EQ Is Done In the Time Domain," 2012. [Online]. Available: http://blog.bjornroche.com/2012/08/why-eq-is-done-in-time-domain.html

[50] C. Roppel, "Realisierung digitaler Filter in C," p. 16, 2009. [Online]. Available: http://www.hs-schmalkalden.de/schmalkaldenmedia/Realisierung_Digitaler_Filter_in_C-p-419.pdf

[51] J. McClellan, T. Parks, and L. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," *IEEE Transactions on Audio and Electroacoustics*, vol. 21, no. 6, pp. 506–526, dec 1973. [Online]. Available: http://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/062_computerprogram.pdfhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1162525

[52] A. Shadrin, "Approximation Theory - Lecture 5," 2005. [Online]. Available: http://www.damtp.cam.ac.uk/user/na/PartIIIat/b05.pdf

[53] I. Selesnick, "EL 713 Lecture Notes 1," pp. 1–33, 2012. [Online]. Available: http://eeweb.poly.edu/iselesni/EL713/remez/remez.pdf

[54] D. W. Guy J Brown, G. J. Brown, and D. Wang, "Fundamentals of Comutational Auditory Scene Analysis," in *Computational Auditory Scene Analysis*, 2006, pp. 1–44. [Online]. Available: http://www.cs.northwestern.edu/$\sim$pardo/courses/casa/papers/casa_chapter_1.pdf

[55] D. Wang and B. G. J., “Binaural Sound Localization,” in *Guy J Brown, DeLiang Wang Brown, Guy J Wang, DeLiang*, 2006, pp. 1–34. [Online]. Available: https://www.cs.cmu.edu/$\sim$robust/Papers/SternWangBrownChapter.pdf

[56] M. I. Mandel, “Binaural Model-Based Source Separation & Localization,” Ph.D. dissertation, 2010. [Online]. Available: http://m.mr-pc.org/work/dissertation_ch4.pdf

[57] R. Duraiswami, “Microphone Arrays and Time Delay Estimation Microphone Arrays,” 2006. [Online]. Available: http://www.umiacs.umd.edu/$\sim$ramani/cmsc828d_audio/MicrophoneArrays.pdf

[58] B. Lee, T. Kalker, and R. W. Schafer, “Maximum-Likelihood Sound Source Localization With A Multivariate Complex Laplacian Distribution.”

[59] K. Varma, “Time-Delay-Estimate Based Direction-of-Arrival Estimation for Speech in Reverberant Environments,” Ph.D. dissertation, 2002.

[60] D. E. Ba, D. Florêncio, and C. Zhang, “Enhanced MVDR Beamforming For Arrays Of Directional Microphones.”

[61] R. Roy and T. Kailath, “ESPRIT - Estimation of Signal Parameters Via Rotational Invariance Techniques,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 984–995, jul 1989. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=32276

[62] R. Schmidt, “Multiple emitter location and signal parameter estimation,” *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, mar 1986. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1143830

[63] J. D. Reed, “Approaches to Multiple-source Localization and Signal Classification,” Ph.D. dissertation, 2009.

[64] S. Ehrhard, M. Fischer, M. Fuhr, M. Mouazzen, M. Müller, and M. L. Schulz, “Spektralschätzung mit MUSIC und ESPRIT,” Tech. Rep., 2011. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/1678008

[65] A. Raviraj, “Direction of Arrival Estimation,” 2007. [Online]. Available: http://www.comm.utoronto.ca/$\sim$rsadve/Notes/DOA.pdf

[66] L. Papula, *Mathematische Formelsammlung für Naturwissenschaftler und Ingeneure*. Vieweg + Teubner, 2009.

[67] E. W. Weisstein, "Hermitian Matrix." [Online]. Available: http://mathworld.wolfram.com/HermitianMatrix.html

[68] G. Systems, "An Introduction to MUSIC and ESPRIT," 2012. [Online]. Available: http://www.girdsystems.com/pdf/GIRD_Systems_Intro_to_MUSIC_ESPRIT.pdf

[69] F. S. Avarvand, A. Ziehe, and G. Nolte, "Self-Consistent MUSIC algorithm to localize multiple sources in acoustic imaging," in *4th Berlin Beamforming Conference 2012*, 2012, pp. 1–9.

[70] N. Ito, E. Vincent, and N. Ono, "Crystal-MUSIC : Accurate Localization of Multiple Sources in Diffuse Noise Environments Using Crystal-Shaped Microphone Arrays," *Signals*, pp. 1–8, 2010.

[71] M. A. Alrmah, S. Weiss, and S. Lambotharan, "An extension of the music algorithm to broadband scenarios using a polynomial eigenvalue decomposition," *19th European Signal Processing Conference*, no. Eusipco, pp. 629–633, 2011. [Online]. Available: http://www.eurasip.org/Proceedings/Eusipco/Eusipco2011/papers/1569428327.pdf

[72] S. Park, S. Jeong, and M. Han, "Target Signal Detection Using MUSIC Spectrum in Noise Environment," *World Academy of Science, Engineering and Technology*, vol. 6, no. 10, pp. 838–842, 2012. [Online]. Available: http://waset.org/publications/13869/target-signal-detection-using-music-spectrum-in-noise-environment

[73] S. Argentieri and P. Danes, "Broadband variations of the MUSIC high-resolution method for sound source localization in Robotics," *Intelligent Robots and Systems, 2007. . . .*, vol. 1, no. 1, pp. 2009–2014, 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4399422

[74] X. Mao and H. Pan, "An Improved DOA Estimation Algorithm Based on Wavelet Operator," *Journal of Communications*, 2013. [Online]. Available: http://www.jocm.us/uploadfile/2014/0102/20140102012454491.pdf

[75] H. Sawada, R. Mukai, S. Araki, and S. Makino, "Estimating the number of sources using independent component analysis," *Acoustical Science and Technology*, vol. 26, no. 5, pp. 450–452, 2005. [Online]. Available: http://joi.jlc.jst.go.jp/JST.JSTAGE/ast/26.450?from=CrossRef

[76] M. Wax and T. Kailath, "Detection of signals by information theoretic criteria," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 2, pp. 387–392, 1985. [Online]. Available: http://www.redes.unb.br/lasp/files/subjects/AASP/WAX_KAILATH_Detection_of_signals_by_information_theoretic_criteria.pdf

[77] Microsoft, "Kinect for Windows Sensor Components and Specifications." [Online]. Available: https://msdn.microsoft.com/en-us/library/jj131033.aspx

[78] IFixit, "Xbox 360 Kinect Teardown - iFixit." [Online]. Available: https://www.ifixit.com/Teardown/Microsoft+Kinect+Teardown/4066

[79] G. Salvi, "ms-kinect-microphone-array-geometry @ giampierosalvi.blogspot.de," 2013. [Online]. Available: http://giampierosalvi.blogspot.de/2013/12/ms-kinect-microphone-array-geometry.html

[80] Wolfson Microelectronics, "Stereo ADC with Microphone Preamplifier," 2004. [Online]. Available: http://www.datasheet.hk/download.php?id=1094716&pdfid=16B7FB91E51D604C1F202938F72B209A&file=0006\wm8737_51814.pdf

[81] M. R. P. Thomas, "Application of Measured Directivity Patterns to Acoustic Array Processing."

[82] P. Soetens, "RTT: Real-Time Toolkit." [Online]. Available: http://www.orocos.org/rtt

[83] G. Guennebaud, B. Jacob, and Others, "Eigen v3," 2010. [Online]. Available: http://eigen.tuxfamily.org

[84] E. Jones, T. Oliphant, P. Peterson, and Others, "SciPy: Open source scientific tools for Python." [Online]. Available: http://www.scipy.org/

[85] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90—-95, 2007.

[86] A. Saadeh, "pySoundToolbox," 2016. [Online]. Available: https://github.com/Trion/pySoundToolbox

[87] ——, "pyMUSIC," 2016. [Online]. Available: https://github.com/Trion/pyMUSIC

[88] NIST/SEMATECH, "NIST/SEMATECH e-Handbook of Statistical Methods," 2013. [Online]. Available: http://www.itl.nist.gov/div898/handbook/eda/section3/probplot.htm

[89] Bundesministerium für Verkehr und digitale Infrastruktur, "Â§ 55 Einrichtungen für Schallzeichen," 2012. [Online]. Available: http://www.gesetze-im-internet.de/stvzo_2012/__55.html

[90] T. Hoger, "Dynamische Wahrnehmbarkeitsanalyse eines Martinshorns im Frequenzspektrum," *Verkehrsunfall - und Fahrzeugtechnik*, vol. 09, pp. 272–278, 2010. [Online]. Available: http://www.ureko.de/downloads/veroeffentlichungen/220

[91] Freiwillige Feuerwehr Gerolfing, “Downloads Feuerwehr,” 2011. [Online]. Available: http://www.feuerwehr-gerolfing.de/d11.html

[92] J. C. Mosher and R. M. Leahy, “Source localization using recursively applied and projected (RAP) MUSIC,” *Signal Processing, IEEE Transactions* . . . , vol. 47, no. 2, pp. 332–340, 1999. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=740118