Delft University of Technology
Faculty Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Signal-processing of audio for speech-recognition

Thesis in favor of
Delft Institute of Applied Mathematics
as part of obtaining

a degree in

BACHELOR OF SCIENCE
in
APPLIED MATHEMATICS

by

JOEP DE JONG

Delft, Netherlands
June 2021

BSc thesis APPLIED MATHEMATICS

"Signal-processing of audio for speech-recognition"

JOEP DE JONG

Delft University of Technology

Supervisor

Dr. ir. R. van der Toorn

Selection committee

Dr. ir. R. van der Toorn

Dennis den Ouden-van der Horst

June, 2021 Delft

Preface

This thesis was written by an Applied Mathematics student at the Delft University of Technology. For the final quarter of my third year, I investigated the possibilities of separating vocals from songs.

In my spare time, I listen a lot to hip-hop music. I am highly fascinated by the texts that some wordsmiths write in these songs. However, sometimes the lyrics are hard to understand due to the accompaniment. Often I look up a lyrics of a song on `genius.com`, a site containing a lot of transcriptions of songs and their meaning or background information.

Transcribing newly released songs is a chore that is often done by fans of an artist. The first person that publishes the lyrics of a song on `genius.com` is likely to receive a large number of internet points. As a collector of these obviously important points, I was thinking if it is possible to automate some parts of the transcription process.

Speech assistants that are controlled by neural networks containing speech-to-text algorithms are used more frequently during the last years. I started with applying these algorithms to songs to obtain their lyrics, but I ran into the problem that these algorithms are bad in distinguishing vocals from the accompaniment. This inspired me to research the possibilities of separating vocals from accompaniment in songs.

I would like to thank my supervisor Dr. ir. R. van der Toorn for his advice, proposals, and enthusiasm during the process of this thesis.

Delft, 30 June 2021
Joep de Jong

Summary

The transcription of voice using neural networks is a technique that deserves attention, as speech assistants are becoming increasingly popular. Neural networks have often difficulty with determining the differences between a talking person and noise. Humans have a much better understanding of this and could possibly apply their knowledge of the structure of the signals to improve the understanding of the neural network. A problem that is extremely difficult for a neural network is understanding and transcribing the lyrics of a song.

This thesis analyzes signal-processing techniques that can be applied to a song to improve the understanding of a speech-recognition algorithm. It is mainly focused on filtering the foreground lyrics from the accompaniment. Some basic filtering methods are described including a low-amplitude filter and a band-pass filter. But also two more complicated filters which make use of the periodicity of the background music will be treated.
The first filter is a method of voice separation using the two-dimensional Fourier transform. This method, proposed by Prem Seetharaman, Fatemeh Pishdadian, Bryan Pardo in 2017 [15], combines techniques of signal-processing and image-processing by finding periodic repetitions in a signal by identifying peaks in the two-dimensional Fourier transform of the spectrogram of the signal.
The second filter is a newly proposed method that can be used for the separation of foreground from background music. The algorithm compares columns in the spectrogram and classifies columns as overlapping if there are multiple occurrences of columns similar to the selected column (repetitions). The frequency components, the different frequencies obtained from a discrete short-time Fourier transform, of overlapping columns are afterward compared with components of the same frequency in other columns. Under certain circumstances, overlapping frequency components are subtracted from components in other columns of the spectrogram. This removes repetitions of that frequency throughout the song. The components of the spectrogram that remain after several iterations of this method are most likely to correspond to the least repetitive parts of the song.

The decisions that are made while constructing the method of comparing spectrogram columns are discussed and are compared with steps performed in the method that uses the two-dimensional Fourier transform. An implementation and demonstration are also attached. From the research it is expected that the two-dimensional Fourier transform perform better on strict periodic accompaniment, while the method that compares spectrogram columns is more likely to perform better on songs with a less tight rhythm.

Table of contents

Chapter 1

Introduction

In an era where the interest in the technology of artificial intelligence such as self-driving cars and robotics is growing, can speech-assistants certainly not miss. An important functionality of a speech assistant is understanding what a user says.

When looking at the possibility to automatically transcribe the lyrics of a song with a neural network, some problems occurred. Besides that a neural network has a long training time to recognize all the words from a language from spoken segments, it seemed also very difficult for the network to understand the differences between the voice and (background) music. Meanwhile, humans have no problems of understanding the difference between a voice and the accompaniment in a song. This raises the question if we can use our knowledge about the human voice and characteristics of songs to help a neural network understand the voice.

An interesting concept that can be used in understanding speech is the separation of voice from noise in an audio signal. This thesis is focused on the separation of a foreground singer from the accompaniment in a song.

The separation methods that will be discussed are simple amplitude and band-pass filters and more sophisticated separation methods that make use of the periodicity of the accompaniment. The latter separation method gave inspiration for a new separation method, which is also proposed in this thesis, that may outperform the method on songs that have a less tight rhythm.

It is assumed that the reader has basic knowledge of both mathematics and signal processing. More complicated or less trivial theory, which is helpful for understanding the thesis, is provided in Chapter 2. In Chapter 3 a total of three filters is introduced. The first two filters are straightforward signal-processing filters. The latter is a more sophisticated filter that uses the periodicity of the signal. Inspired by the last filter, I propose a new filter in Chapter 4. The implementation and a small demonstration of the algorithm are attached in Chapter 5. The comparison of the filters and discussion of the proposed filter is done in Chapter 6.

Chapter 2

Theory of signals, sampling, and spectrograms

To perform calculations on audio signals, it is important to understand what audio signals precisely are. In this chapter, it will be clear how audio is measured and discretized.

2.1 Theory of signals

As you might know, audio can be recorded with a microphone. Audio moves as a wave through the air, by vibrating molecules in the field it moves through. A microphone measures the continuous variations in air pressure and gives an analog output in Voltage. The analog signal will be converted by an analog-to-digital converter (ADC) into a digital discrete-time signal.

We will take a look at some properties of an analog audio signal $s(t)$ as a function of time.

2.1.1 Properties of analog signals

From physics, we know that an infinite source of energy is uncommon or even impossible. With the same reasoning, the total amount of power (energy per unit time) is also bounded. The same holds for audio waves: there can not be an infinite amount of energy in an audio wave, and the waves have to be bounded in power, which means that the variations in air pressure have to be bounded too.

The power $P(t)$ of a signal $s(t)$ at a certain point in time is defined[6] as

$$P(t) = s^2(t) \tag{2.1}$$

The total energy E of a signal $s(t)$ is defined as follows.

$$E = \int_{-\infty}^{\infty} P(t)dt = \int_{-\infty}^{\infty} s^2(t)dt \tag{2.2}$$

Assuming that we work with real-world audio signals, we can apply the bounded-energy laws of physics to equations 2.1 and 2.2 to conclude that both power and energy have to be finite, so for some $M, N \in \mathbb{R}$ we have

$$P(t) = s^2(t) \leq M$$

and

$$E(t) = \int_{-\infty}^{\infty} P(t)dt \leq N$$

2.1.2 Properties of digital signals

Digital signals have completely changed the world we live in. Can you remember the last time you recorded a photographic film? The days that you had to bring your photographic film to a specialist or the Hema to develop 12 to 36 photos? It seems all a while ago, maybe the Hema does not even exist anymore when you are reading this. And how revolutionary was the first iPod, that was able to store more than 1000 songs? How is it even possible to store that many songs on a pocket-size device, while previously only 22 minutes[16] of music could be stored on a single side of a 12-inch vinyl record?

The advantages[13] of digital signals are huge. A physical vinyl record is vulnerable and loses quality over time, especially when used. On contrary, a digital signal ensures consistent playback quality. Digital signals are also highly portable and have a wider dynamic range.

Since computers can not work with continuous signals[2], the first step in digitizing a signal is discretization. One can discretize a time series by taking samples of the continuous-time signal at certain several timestamps.

Nyquist rate

The signal will be sampled at a certain sample rate. The sample rate is the rate of how many samples are taken every second ($\text{Hz} = \text{s}^{-1}$). An interesting question to ask is what the lowest sampling rate is at which it is still possible to recover the original signal from its samples. When using this sampling rate, the memory needed is minimum, but we do not lose any important information. A well-known theorem from signal processing tells us that if we want to reconstruct the signal exactly from our samples, the sample rate should be at least twice the bandwidth (highest frequency) of the signal. This rate is also called the **Nyquist rate**.

When using a sample rate that is higher or equal to the Nyquist rate, one prevents aliasing in the digitized, discrete signal. Frequencies higher than half the sample rate are sampled incorrectly and distort the signal.

Frequency range of human speech

Since the focus of this project is on human speech and hearing, we only have to take into account the frequencies that lie in the scope of human speech and hearing. The frequency band of the human voice ranges from 300 Hz to 3400 Hz. The frequency band of human hearing covers a range from 20Hz to 20KHz [5]. From the Nyquist rate theorem, it follows that the minimum sample rate is around 40KHz. The most commonly used sample rates for sampling audio waveforms lay a bit above 40Khz. For example, 44.1kHz is the standard sampling rate for consumer audio (like CDs). The Nyquist frequency is in this case a bit higher than the range of human hearing, which makes it easier to prevent the aliasing of audio signals.

2.1.3 Theory of sampling

To work with digital signals, we need to make our continuous signal discrete. But can we still apply all the Fourier theory to these signals? Is there a simple, mathematical way to make a continuous signal discrete?

Dirac delta function

A function that is commonly used to discretize signals is the Dirac delta function. This function is zero everywhere, except at the origin, where it is infinite:

$$\delta(t) = \begin{cases} +\infty, & t = 0 \\ 0, & t \neq 0 \end{cases}$$

Shifting the Dirac delta function with s gives an impulse at $t = s$:

$$\delta(t - s) = \begin{cases} 1, & t = s \\ 0, & t \neq s \end{cases}$$

Impulse train

When we sum over multiple, distinct Dirac delta functions, we get impulses at all the shifted points. E.g.

$$s_T(t) = \sum_{k \in \mathbb{Z}} \delta(t - kT)$$

is a signal with impulses at $t = 0$, $t = T$, $t = -T$, $t = 2T$, ... This type of signal is called an **impulse train**, since it is like an train of consequent impulses. The period of the train is T as the impulse repeats itself every T steps.

Now lets take a look at the fourier coefficients of the Fourier series of the Dirac delta function over an interval of length T:

$$\begin{aligned} \hat{\delta}(n) &= \frac{1}{T} \int_T \delta(t) e^{-itn} dt \\ &= \frac{1}{T} \delta(0) e^{-i0n} \\ &= \frac{1}{T} \end{aligned}$$

To calculate the Fourier series coefficients of the impulse train, we first look at the translation property of what happens to the Fourier series of a translated function. From Fourier analysis[1], we know that for a 2π-periodic $f \in L^1(\mathbb{T})$ and $\tau \in \mathbb{T}$, the translated function is defined as

$$f_\tau(t) = f(t - \tau), t \in \mathbb{T}$$

Then the Fourier coefficients of the translated function can be calculate with a simple substitution of $s = t - \tau$:

$$\begin{aligned} \hat{f_\tau}(n) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t - \tau) e^{-int} dt \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f(s) e^{-ins} e^{-in\tau} ds \\ &= e^{-in\tau} \hat{f}(n) \end{aligned}$$

Using this property, we can derive the Fourier series coefficients of the periodic impulse train:

$$\begin{aligned} S_T[n] &= \frac{1}{2\pi}\int_{-\pi}^{\pi} s_T(t)e^{-itn}dt \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \sum_{k\in\mathbb{Z}} \delta(t-kT)e^{-itn}dt \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \delta(t)e^{-itn}dt \\ &= \frac{1}{T} \end{aligned}$$

The Fourier transform is thus:

$$\hat{s_T}(n) = \sum_{k\in\mathbb{Z}} \frac{1}{T}e^{-ikn} = \frac{1}{T}\sum_{k\in\mathbb{Z}} \delta\left(n - \frac{k}{T}\right)$$

Thus if a function is an impulse train in the time domain, its Fourier transform is an impulse train in the frequency domain.

Discretizing

With the help of an impulse train, we can make a continuous signal $x(t)$ discrete. Let T be the sampling time and $\omega = \frac{1}{T}$ be the sampling rate. The first step in discretizing the signal is multiplying the signal with the impulse train $s_T(t)$. We will obtain a sampled version $f(t)$ of the signal $x(t)$. The distance between the samples is T.

$$f(t) = x(t) \cdot s_T(t)$$

The signal that remains is zero everywhere, except at the impulses that occur every ω unit.

$$f(t) = \begin{cases} x(t), & \exists k \in \mathbb{Z} : t = k\omega \\ 0, & \forall k \in \mathbb{Z} : t \neq k\omega \end{cases}$$

Note that the continuous-time sampled version of a signal has no information about the original signal at points that are not a multiple of the sampling rate. In the last step of discretizing the signal, we will leave out these points. The points that remain are the points that contain information about the signal. For some $k \in \mathbb{Z}$ the discretised signal of $x(t)$ is thus:

$$x[k] = x(k\omega) = x(k\frac{1}{T})$$

Sampling rate

Note that if T is smaller, the distance between the samples is smaller, which means that the signal is sampled faster. When the sampling time T increases, the sampling rate decreases and the signal is sampled slower. When sampling, there is a trade-off between the quality of the sampled signal and the memory that is needed: increasing the sampling rate increases the quality, but also increases the memory needed to store the signal.

2.1.4 Bounded frequencies

A key difference between a continuous and discrete signal is the bound of frequency components. The frequencies of a continuous-time signal are unbounded, but the frequencies of a discrete (sampled) signal are linked to the sampling rate. The bound of frequency can lead to loss of (higher frequency) information about the original signal in the sample's signal.

The sampling rate bounds the number of different frequency components that the Fourier transform of the discrete signal has. When sampling, the highest possible frequency is the frequency that corresponds to a change in sign at every sample. The minimum period of the signals for accurate sampling is thus twice the distance between the samples. It is impossible to distinguish frequencies higher than half the sample rate from frequencies that are lower. This explains also why the Nyquist rate is twice the bandwidth of the signal.

Continuous-time version of a discrete signal

It is possible to extend a discrete-time signal to a continuous-time signal. In the last step of discretizing a signal, we erased all the points that did not have any information about the signal. We can add these points again by setting the function equal to zero between the impulses. This continuous-time signal will probably not be continuous (only if it is identical to zero). But it is clearly measurable and thus an $L^1(\mathbb{R})$ function, so we can apply the Fourier theory to it!

It is also possible to draw a sinusoidal between the impulses. In some cases, this can result in reconstructing the original signal. But, as discussed before, sampling a signal can result in losing information about higher frequency components and these components cannot be reconstructed.

2.2 Periodicity of signals

2.2.1 Periodicity of continuous signals

Suppose we have a continuous, complex-valued signal $x(t)$ which is composed of (unknown) sine waves. Then we can decompose this signal into its sine waves using a Fourier transformation.

Let $L^p(\mathbb{R})$ be the space of measurable functions:

$$L^p(\mathbb{R}) = \{f : \mathbb{R} \to \mathbb{C} : ||f||_p := \left(\int_{\mathbb{R}} |f(t)|^p dt\right)^{1/p} < \infty\}$$

For a function $f \in L^1(\mathbb{R})$ and $\xi \in \mathbb{R}$ we define[1] the Fourier transform of f as

$$\hat{f}(\xi) := \int_{\mathbb{R}} f(x) e^{-it\xi} dt \tag{2.3}$$

In section 2.1.1 we have seen that the total energy of an analog signal is bounded. So our analog signals are in L^2. Since L^2 is contained in L^1, we can use the Fourier transform on our analog signals.

Fourier transform

The Fourier transform is an integral over the whole real axis. The audio signals we work with are of finite length, e.g. l. So our signal $x(t)$ takes real values in the interval $[0, l]$. Outside the interval, the signal is zero-padded.

$$s(t) = \begin{cases} x(t), & t \in [0, l] \\ 0, & otherwise \end{cases} \tag{2.4}$$

Fourier series

Apart from Fourier transforms, one could also use the Fourier series. Fourier series are easy to work with when a signal is 2π-periodic. Our signal is clearly not periodic, and not of length 2π. However, we can make it of length 2π by transforming the signal and we can make it periodic by taking a periodic extension of the function outside $[0, l]$:

$$\tilde{x}(t) = \sum_{k\in\mathbb{Z}} x\left(\frac{2\pi}{l}t + 2\pi k\right) \tag{2.5}$$

The Fourier series of a function is a trigonometric series that uses the n-th Fourier coefficients as weight. The n-th Fourier coefficient is similar to the Fourier transform (2.3) but has only discrete frequencies:

$$\hat{f}(n) := \int_{\mathbb{R}} f(x)e^{-itn}dt \tag{2.6}$$

Differences between Fourier transform and Fourier series

The difference between the Fourier series and Fourier transforms is that the Fourier series describes a periodic function by a discrete sum of complex exponential functions. Fourier transforms are used in a more general case: they describe non-periodic functions as an integral (continuous) of complex exponential functions.

2.2.2 Periodicity of discrete signals

A discrete signal $x[n]$ is said to be periodic when

$$x[n] = x(n + p) \tag{2.7}$$

for some non-zero integer p.
Let N be the smallest possible natural number for p, then N is the length of the period (the number of samples). The **fundamental frequency** (ω_0) of a signal is the lowest frequency of a periodic waveform, it can be interpreted as the **pitch**.

When a discrete signal is periodic, the ratio of its fundamental frequency divided by 2π must be a rational number. So for M, $N \in \mathbb{Z}$:

$$\frac{\omega_o}{2\pi} = \frac{M}{N}$$

In this case, M is the number of full cycles.

Example 1

Denote the discrete sine function with frequency ω and phase ϕ as

$$\sin(\omega n + \phi)$$

We will look at the periodicity of the discrete signal

$$\sin\left(\frac{n}{2} - \frac{\pi}{2}\right)$$

The frequency (ω) of this signal is $\frac{1}{2}$. The factor $\frac{\omega}{2\pi} = \frac{1}{4\pi}$ is not rational, and thus the signal is not periodic.

Example 2

Next, we will look at the periodicity of the discrete signal

$$\cos\left(\frac{7\pi n}{6} + 2\right)$$

The frequency of this signal is $\frac{7\pi}{6}$. Dividing this by 2π gives $\frac{7}{12}$, which is a rational number. So the signal is periodic and has a period of 12 samples. The shift of the signal does not impact the periodicity.

2.3 Short-time Fourier transform

It is also possible to perform a Fourier transform on a short segment of a signal. This is called the short-time Fourier transform (STFT).

2.3.1 Window functions

The short-time Fourier transform is easy to compute with the help of a window. The name window is self-explanatory: by looking through a window at the signal, one can only see the parts of the signal that are visible through the window. A window is a function that takes only values in a certain interval, outside the interval the function is always zero. By multiplying a signal with a window function, only parts of the signal that are in this interval remain. Usually, window functions are symmetric and centered around $x = 0$, but they can also be shifted with a factor τ.

Hann window

A function that is often used in the calculation of short-time Fourier transforms is the Hann window function. It is a sinusoidal shaped window that has smooth transitions to zero at the endpoints of its interval. The standard Hann window (figure 2.1) of length L is given by a squared cosine function centered around $\tau = 0$:

$$w_0(x) = \begin{cases} \cos^2\left(\frac{\pi x}{L}\right), & |x| \leq \frac{L}{2} \\ 0, & |x| > \frac{L}{2} \end{cases} \tag{2.8}$$

In signal processing, we usually work with sampled signals. For the discrete version of the Hann window, the length L is replaced by $N + 1$, where N is the number of samples in the window. The shift τ is replaced with an integer m. Outside the window, the function still is zero. To make sure that the window function starts at 0, we shift the cosine function a quarter of a period. The standard discrete window function (figure 2.2) can thus be written using a sine.

$$w[n] = \sin^2\left(\frac{\pi n}{N}\right), 0 \leq n \leq N \tag{2.9}$$

2.3.2 Continuous short-time Fourier transform

The short-time Fourier transforms calculates the Fourier transform of the product of the signal $x(t)$ and the window function $w(t)$. This transform can be calculated for every position τ of the window along the time-axis.

$$STFT\{x(t)\}(\tau, \omega) \equiv X(\tau, \omega) = \int_{-\infty}^{\infty} x(t) w(t - \tau) e^{-i\omega t} dt \tag{2.10}$$

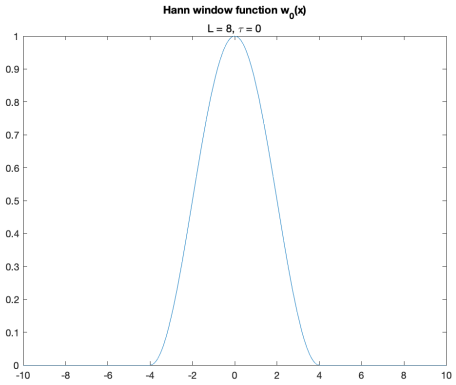

Figure 2.1: Continuous Hann window function $w_0(x)$ with length $L = 8$ and centered at $\tau = 0$.

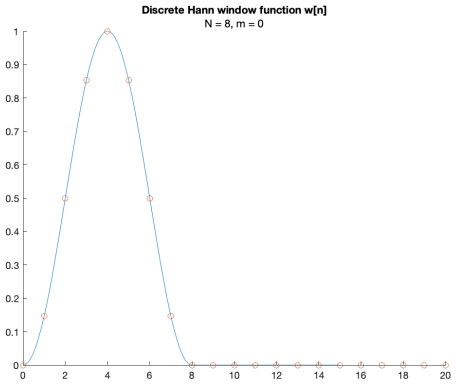

Figure 2.2: Discrete Hann window function $w[n - m]$ with length $N = 8$ and starting at $m = 0$.

2.3.3 Discrete short-time Fourier transform

We are especially interested in the discrete short-time Fourier transform. Similar to the continuous case we will use a window function. The discretized Hann window as stated in equation 2.9 can be used to obtain a chunk of samples from the signal, it can also be shifted, but only by a discrete value m. This will give us again a two-dimensional representation:

$$STFT\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \tag{2.11}$$

The frequencies ω can be continuous, but we will only use discrete values for ω.

2.4 Two-dimensional Fourier transform

The one-dimensional Fourier transform decomposes a signal into the magnitudes and phases of its frequency components. It decomposes a one-dimensional function into the functions $e^{int} = \cos(nt) + i\sin(nt)$, which form an orthogonal basis. In a similar way, this is possible for higher dimensional functions. Instead of a one-dimensional signal, these transforms can be done on higher dimensional data such as a two-dimensional image. A technique that is often used in image-processing is the 2-dimensional Fourier transform (2DFT)[17]. Similar to the 1DFT it makes use of an orthogonal basis:

$$e^{i(ux+vy)} = \cos(ux + vy) + i\sin(ux + vy)$$

Note that the extreme values of $\cos(ux + vy)$ occur when $ux + vy = n\pi$ for some $n \in \mathbb{Z}$.

The solution of this equation is a set of uniformly spaced parallel lines with normal vector u and wavelength $|x|^{-1}$. In the direction orthogonal to the lines, the solution is a sinusoid. The magnitude $|x|$ of x is thus the frequency of the sinusoid. The normal vector u points as usual in the direction of the lines (perpendicular to the sinusoid). To summarize, the 2DFT decomposes images into a summation of weighted and phase-shifted 2-dimensional sinusoids.

Like the one-dimensional Fourier transform, the two-dimensional Fourier transform has several forms that depend on whether the function or signal that is transformed is periodic and/or discrete.

2.4.1 Aperiodic, continuous transform

The most general case is the continuous two dimensional Fourier transform of an aperiodic, continuous function $f(x, y)$, which is defined as

$$F(u, v) = \int \int_{-\infty}^{\infty} f(x, y) e^{-i(ux+vy)} dxdy$$

with inverse

$$f(x, y) = \int \int_{-\infty}^{\infty} F(u, v) e^{i(ux+vy)} dudv$$

where u and v are spatial frequencies in the x and y directions.

2.4.2 Aperiodic, discrete transform

Later on, we will use the 2DFT for aperiodic, discrete functions $f[m, n]$:

$$F[k, l] = \frac{1}{\sqrt{MN}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[m, n] e^{-i(\frac{mk}{M} + \frac{nl}{N})}$$

with inverse

$$f[m, n] = \frac{1}{\sqrt{MN}} \sum_{l=0}^{N-1} \sum_{k=0}^{M-1} F[k, l] e^{i(\frac{mk}{M} + \frac{nl}{N})}$$

where M and N are the number of samples in x- and y-direction respectively.

Note that the Fourier transform on a discrete signal with finite length is similar to the Fourier series of the periodic extension of the signal.

2.5 Theory of spectrograms

Audio moves as a wave through the air and an audio signal describes the variation in air pressure at a certain time. It can be interesting to visualize an audio signal. Besides a simple time-amplitude plot, one could also look at the spectrogram of a signal. The spectrogram is an intensity plot. It represents the spectrum of frequencies and their intensity at different points in time.

The x-axis of a spectrogram is the time-axis. The y-axis shows the pitch. The third component, volume, is showed by the color.

A spectrogram can be used to analyze the timbre (quality/texture) of a signal. Since it visualizes the intensity of different frequencies, one could see whether or not overtones occur. The spectrogram of a whistling person has a pure pitch. But the spectrogram of people at the dentist shouting the letter A has a lot more lines: the human voice vibrates at different frequencies.

The timbre of a signal is said to be dark when there are only a few overtones. When there are a lot of overtones, the timbre is bright.

2.5.1 Making a spectrogram

In section 2.3.3 we have seen how to calculate the discrete short-time Fourier transform of a signal. This STFT can be used to construct a spectrogram of a signal.

The following input is required to make a spectrogram.

- A discrete signal $x[n]$
- A window $w[n]$ with window size m
- The number of samples that overlap between adjoining segments k
- The number of frequencies used in the discrete Fourier transform f

The first step is segmenting the signal into chunks of length m that overlap k samples:

Let $x[n]$ be a discrete signal of length N. Then, using the window, we can split up this signal in smaller, overlapping segments of length m with overlap k. Assume that $m \ll N$ and put all the segments in a matrix X. The dimensions of this matrix will be $m \times M$ where M is $\frac{N}{m-k}$ rounded down. Both rows and columns of the matrix are indexed by time. The first column of X contains all the samples in the first segment, the second column contains the samples of the second segment, and so on. The segments overlap with k samples and thus the original signal x is clearly represented by the matrix X.

The next step is to transform the columns of the matrix from the time domain to the frequency domain. Let $\hat{X}_k$ be the discrete Fourier transform (DFT) of the k-th column of X. Note that the Fourier transform uses points in frequency instead of time. Since we only have a finite number of samples, the total number of frequencies is bounded. Every $s \geq N$ will work, but $s = 2N$ seems to be a good candidate.

Now let $\hat{X}$ be the matrix containing all the Fourier transforms as the columns:

$$\hat{X} = [\hat{X}_1^T, \hat{X}_2^T, \ldots, \hat{X}_M^T]$$

This complex matrix contains all the data we need for constructing the spectrogram of $x[n]$. The used window w has size m. The values in the matrix $\hat{X}$ are the intensities at a certain time and frequency.

The **spectrogram** is the visualization of the matrix in a two-dimensional image with a color scale. It has a 2-dimensional structure with time on the x-axis and frequency on the y-axis. The colors correspond to the magnitude of the real part of the short-time Fourier transform. The magnitude is the square of the amplitude. The imaginary part of the Fourier transform, the phase, is not displayed.

It is possible to filter audio using the phase spectrogram using a 3-dimensional Transform [11]. But in this project, all the calculations will be performed on the frequency spectrogram.

In 1943 Georg Ohm already claimed[12] that the human ear is not able to hear differences in phase. Twenty years later came Helmholtz to the same conclusion [7]. This seems to hold for almost every audio signal[10], for this reason, the phase spectrogram will be neglected in this project.

2.5.2 Inverting a spectrogram

An interesting question to ask is whether we can reconstruct the original signal from its spectrogram. For this, we will look at the columns separately. The discrete Fourier transform that was applied to the product of the signal with a window can simply be inverted. We now remain with the product of the signal and a window. Depending on the type of window that we initially used, we can find an inverse window. The initial window that is applied to the signal is called the **analysis window**. The window that is used during the inversion is called the **synthesis window**. The synthesis window can be applied to the raw output of the inverse short-time

Fourier transform. In our case a window that is solely 1 is sufficient[9]. After applying the synthesis window, the segments of the signal are summed using an overlap-add synthesis method[3]. The details of the overlap-add synthesis method are out of the scope of this project, but one thing is noticeable: the original signal can easily be reconstructed from its short-time Fourier transform if the short-time Fourier transform has no overlap between the segments. This can be done by inverting the short-time Fourier transforms of consecutive segments and putting the inverted segments into a sequence.

The above procedure describes how to invert the short-time Fourier transform. However, when making a spectrogram an additional step is taken: only the squared amplitude remained in the image, the phase was neglected. This makes it nearly impossible to reconstruct the exact signal from its spectrogram since for example, the phase information lacks. If interested, the reconstruction of an audio signal from its spectrogram is studied in a recent research project from Rémi Decorsière[4].

Chapter 3

Filtering

Throwing raw data in a neural network is of course possible, but not preferred. For a machine-learning algorithm, it is sometimes hard to understand the structure of the data and the types of fields. The person who provides the dataset has probably a lot of pre-knowledge about its structure, which can be used to do some transformations on the data to improve the algorithm.

An example of this is when you hear a song, you recognize the (background) music and a (foreground) voice. You can feed this song as input to a machine-learning algorithm to recognize some characteristics of the voice. However, the algorithm will get a hard time making good predictions, since it does not know if it has to listen to the (background) music, the (foreground) voice, or both.

3.1 Low-amplitude filter

In a song, the softer sounds in the background are usually not the singer. We could filter all these sounds out since they are not of interest. Due to the waveform of sound, it is not possible to simply filter out low amplitudes in the time domain. For this reason, we filter out frequencies that have a low magnitude in the frequency domain:

$$\hat{f}[n] = \begin{cases} \hat{x}[n], & \hat{x}[n] \geq M \\ 0, & \hat{x}[n] < M \end{cases}$$

$$f[k] = \sum_{n=0}^{N-1} \hat{f}[n] e^{ink}$$

The result is quite significant. But if we listen to the part of the signal that is subtracted, we can also hear the voice in it. It seems that some overtones of the voice are subtracted using this method, which is not ideal.

3.2 Bandpass filter

Our main goal is to help a speech-to-text algorithm accurately transcribe the lyrics of a song. The separation of a human voice from an audio signal could contribute a lot to this. We already noticed before that the human voice ranges from 300Hz to 3400Hz. With this knowledge, one type of filter is quite trivial: filtering high and low frequencies out of the signal.

The results of the bandpass filter are as expected: when lower and upper frequencies are chosen correctly, only background music is filtered. However, no filtering is applied to the band

between these frequencies. This filter is thus unable to filter out accompaniment with the same frequency as a human voice.

3.3 Separating voice using a two-dimensional Fourier transform

A more sophisticated method of filtering that helps the speech-recognition algorithm is separating the voice from background music that lies in the same frequency range.

As seen in section 2.4 the two-dimensional Fourier transform decomposes images into a summation of weighted and phase-shifted sinusoids. In an article[15] by Prem Seetharaman, Fatemeh Pishdadian, Bryan Pardo an interesting method is described: they apply well-known image processing techniques to the spectrogram of an audio signal. Usually, only one-dimensional Fourier transforms were of interest for audio signals, but by looking at the 2-dimensional spectrogram a whole world of possibilities opens up. The method of Prem Seetharaman, Fatemeh Pishdadian, Bryan Pard is applied to the spectrogram of an audio signal and consists of 5 steps.

Step 1: Constructing a spectrogram

The method takes a magnitude spectrogram as input. This spectrogram can be constructed from a signal using the method described in section 2.5.1. Only the magnitudes are placed in the matrix, to turn the spectrogram into a magnitude spectrogram. The phase is left out.

Step 2: Transforming to the scale-rate domain

The spectrogram shows which frequencies occur at a certain time interval. As described in section 2.5 it is actually a matrix consisting of several short-term Fourier transforms of the signal. This matrix containing all the intensities of the signal can be visualized in a two-dimensional image. This makes it possible to borrow techniques from other studies. A two-dimensional Fourier transform (2DFT) can be performed on this image. The result is a complex matrix containing the phases and weights of the sinusoids that form the spectrogram. We refer to this matrix as the scale-rate representation of the spectrogram.

Let $X(m,\omega)$ be the discrete short-time Fourier transform of the signal. $X(m,\omega)$ defines a function on the time-frequency domain and the plot of its magnitude is the spectrogram that is used as input. The scale-rate representation $\tilde{X}(s,r)$ of the spectrogram is given by

$$\tilde{X}(s,r) \equiv FT_{2D}\{|X(m,\omega)|\} \tag{3.1}$$

$$= \frac{1}{\sqrt{MN}} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} |X(m,n)| e^{-i(\frac{mk}{M}+\frac{nl}{N})} \tag{3.2}$$

The scale, on the vertical axis, can be interpreted as the spread of spectral energy as well as frequency-domain repetitions (e.g. overtones). The rate is plotted against the horizontal axis and corresponds to temporal repetitions.

Step 3: Locating peaks and masking in the scale-rate domain

The second step in their method is locating the peaks in the scale-rate representation of the spectrogram. From image-processing, we know that these peaks correspond to the most repeating patterns. Thus in our case, the peaks correspond to the patterns in the spectrogram that periodically repeat.

The peaks can be located by looking at small neighborhoods in the spectrogram. We calculate the **range** α_c of values in a neighbourhood around an arbitrary point $c = (s_c, r_c)$ by subtracting its maximum and minimum magnitude.

$$\alpha_c = \max_{N(c)} |\tilde{X}(s,r)| - \min_{N(c)} |\tilde{X}(s,r)|$$

A point is selected as a peak when it is a **local maximum** in its neighbourhood. A threshold γ is introduced to ensure the distance between the points in the neighbourhood is large enough to talk of a peak. A background mask which takes values in $0, 1$ is created. Only points that are peaks are masked with a 1:

$$M_{bg}(s_c, r_c) = \begin{cases} 1 & \alpha_c > \gamma, |\tilde{X}(s_c, r_r)| = \max_{N(c)} |\tilde{X}(s,r)| \\ 0 & otherwise \end{cases} \tag{3.3}$$

The foreground mask is the mask containing the points that are not peaks:

$$M_{fg}(s,r) = 1 - M_{bg}(s,r) \tag{3.4}$$

The background mask can be applied to the scale-rate domain representation of the signal to obtain the repetitive accompaniment in the scale-rate domain. The Hadamard (element-wise) product of the mask and signal matrices gives the desired masked signal:

$$M_{bg}(s,r) \circ \tilde{X}(s_c, r_r) \tag{3.5}$$

Shape of neighborhood

The shape of the neighborhood can be arbitrary. But we are only interested in repetitions along the time axis of the spectrogram. The time axis (horizontal) shows the changes in the spectrum throughout time. Periodic repetitions of frequency components in this direction, show that these frequencies occur several times in the song and that is what we are interested in. The periodic repetitions along the (horizontal) time-axis of the spectrogram correspond, after applying the two-dimensional Fourier transform, to (local) maxima along the (horizontal) rate-axis in the scale-rate domain.

On the other hand, periodic repetitions along the frequency axis, the vertical direction of the spectrogram, is not of our interest. This is because these repetitions correspond to overtones of the frequency. Periodic repetitions in the vertical direction of the spectrogram correspond to (local) maxima in the vertical direction of the scale-rate representation.

For what we want to achieve, it thus is best to use a neighborhood that is more horizontal than vertical. The neighborhood that is suggested has a height (scale) of 1 frame and a width (rate) of 15 to 100 frames.

There is a trade-off in the width of the neighborhood: a small width results in more leakage from the foreground to the background music, while a larger number of frames results in more leakage from the background music to the foreground. This can be explained by the mask in equation 3.3: only the points in the scale-rate domain that are a local maximum in a neighborhood around them are masked as background points in the scale-rate domain. Increasing the size of the neighborhood reduces the chance that these points are local maxima and results in classifying fewer points as background music. Meanwhile, decreasing the width of the neighborhood masks more points as foreground. The threshold γ also enhances the effect of classifying points in the scale-rate domain as foreground in the case of a small neighborhood, since in this case, the range of frequencies α_c around c is less likely to exceed γ.

Step 4: Masking in the time-frequency domain

In the previous step, we found a masked signal for the accompaniment in the scale-rate domain. By applying the inverse of the two-dimensional Fourier transformation we obtain the magnitude spectrogram of the repetitive accompaniment in the time-frequency domain:

$$|X_{bg}(m,\omega)| = IFT_{2D}\{M_{bg}(s,r) \circ \tilde{X}(s_c, r_r)\} \tag{3.6}$$

The absolute value around $X_{bg}(m,\omega)$ is introduced in the notation to make it clear we calculated the magnitudes of the complex-valued spectrogram $X_{bg}(m,\omega)$. We can only calculate the magnitudes of the background and foreground signals since the two-dimensional Fourier transform was initially applied (3.1) to the magnitude of the original spectrogram.

The magnitude spectrogram of the foreground vocals is obtained by applying the inverse transformation to the scale-rate representation that is masked with the foreground mask:

$$|X_{fg}(m,\omega)| = IFT_{2D}\{M_{fg}(s,r) \circ \tilde{X}(s_c, r_r)\} \tag{3.7}$$

A new mask is introduced in the time-frequency domain, which compares the magnitude spectrograms of the foreground and background parts of the signal. Only points in the time-frequency domain that have a higher background than foreground magnitude remain in the background mask:

$$M_{bg}(m,\omega) = \begin{cases} 1 & |X_{bg}(m,\omega)| > |X_{fg}(m,\omega)| \\ 0 & otherwise \end{cases} \tag{3.8}$$

The foreground mask is again the complement of the background mask:

$$M_{bg}(m,\omega) = 1 - M_{bg}(m,\omega) \tag{3.9}$$

It may be unclear why this second mask is introduced. In the previous step, we already classified points in the scale-rate domain as background or foreground using the mask described in equation 3.3. A single point in the scale-rate domain corresponds to several, equidistant, points in the time-frequency domain. We can, of course, use these points for the foreground and background spectrogram, but that may not be optimal: A signal with a soft periodic background part, e.g. a tone of a certain frequency every m samples, and a loud aperiodic foreground part, e.g. a saxophone solo, with some frequencies in common with the background part, will, in this case, not be filtered as desired. Due to the highly repetitive background music, the frequency of the tone will be masked as the background for every m-th sample. However, the saxophone solo also has this frequency at an m-th sample which will thus be filtered out. This is undesired since the solo was louder and thus more important than the background music. To prevent this, a second mask (3.8) is introduced which compares the magnitudes of the foreground and background spectrogram, and only classifies a point in the spectrogram as background if it is softer than the foreground.

Step 5: Obtaining signals from the spectrograms

The last step is transforming the spectrograms back into time signals. As discussed in section 2.5.2, it is possible to invert the short-time Fourier transform using the overlap-add synthesis method to obtain the separated background and foreground signals:

$$x_{bg}(t) = ISTFT\{M_{bg}(m,\omega) \circ X_{bg}(m,\omega)\}, \tag{3.10}$$

$$x_{fg}(t) = ISTFT\{M_{fg}(m,\omega) \circ X_{bg}(m,\omega)\} \tag{3.11}$$

3.3.1 Discussion

The method described looks at repetitions along the time axis of the spectrogram. This can be roughly interpreted as the "frequency" at which frequencies of the signal occur. This frequency, the so-called rate, has a high amplitude for components of the signal that are periodic. The periodicity of a signal is discussed in section 2.2. Remember that a discrete signal is only periodic when the number of samples in a period is constant. For this reason, only frequencies that occur every $n \in \mathbb{N}$ samples are masked in the time-frequency domain. In the time-amplitude domain, this translated to the components of the audio that have a tight rhythm.

Chapter 4

Separating voice by comparing spectrogram columns

The voice-separation method using the two-dimensional Fourier transform is a good method for signals with constant periods. But what if the period of signals (of a certain frequency) changes in time? Could we find a method that performs better on less periodic signals?

4.1 Motive

A human drummer does not drum nearly as tight as a computer-generated drum. Even the rhythm of a professional drummer is imperfect. The error in the rhythm of a professional Ghanaian drummer is studied[8] and his rhythm seems to be on average 16 milliseconds off.

For a signal that is sampled at a sample rate of 48kHz, these 16 milliseconds correspond to an average error of 768 samples. The error in the samples in the time-amplitude domain has also an impact on the time-frequency domain: the short-time Fourier transform will have some frequencies that are slightly off or have even some samples that end up in another segment.

In the previous section, we discussed a method that analyses the periodicity with the help of a two-dimensional Fourier transform. This method has some disadvantages for signals that are off the beat. The disturbances in the time-frequency domain are carried over to the scale-rate when taking the two-dimensional Fourier transform of the spectrogram. This can cause larger problems in the scale-rate domain.

Indeed, the two-dimensional Fourier transform decomposes the signal in its horizontal (scale) and vertical (rate) repetitions. The horizontal repetitions (e.g. overtones) are hardly disturbed, only by some samples that ended up in the wrong segment. However, the vertical repetitions (repeating frequencies) may be off a lot: the frequencies that were slightly off in the time-frequency domain end up in different rows and are thus not counted as repetitions anymore. The peaks in the scale-rate domain will be smaller or occur at different places.

As a result, the masking method may end up masking peaks that are not mainly caused by periodic parts of the accompaniment. The separated signal of the human drummer will have more leakage from the vocals to the accompaniment, and vice-versa than the computer-generated drum. The drummer is a striking example, of course, the same reasoning holds for periodic signals of other instruments, even if they are less periodic than a drum.

4.2 Introduction

Inspired by the 2DFT-method, we will also look at the spectrogram. The idea is quite intuitive: we will not take a Fourier transform, but we will compare the columns of the spectrogram with the other columns and determine their overlap. When a column overlaps often with other columns, we subtract its most important frequency components from all the other columns. The four questions that remain are:

- When do columns overlap?
- How often should they overlap?
- When is a frequency component of a column important?
- How to subtract the most important frequency components from other columns?

Before answering those questions, we will look at the general structure of the spectrogram. The spectrogram is a matrix in $\mathbb{C}^{n\times m}$ where n is the number of frequencies and m the number of segments. The values are complex since the Fourier transform transforms the signal into an orthogonal basis of sines and cosines. The real part corresponds with the even parts of the signal and the imaginary part with the odd parts. The magnitude of the complex number corresponds to the amplitude of the signal. The angle that it makes in the complex plane corresponds to the phase.

For this method, it is recommended to use a spectrogram with no overlaps in segments of the spectrogram. Overlapping segments results in overrepresenting the time signal. Since some data is duplicated, it increases the chance of consecutive columns of the spectrogram to be overlapping. Besides that, it complicates the inversion of the spectrogram as discussed in section 2.5.2. Overlap of segments is thus undesired.

To simplify calculations, we will look at the real and imaginary parts separately. We will thus perform our calculations on two $\mathbb{R}^{n\times m}$ matrices.

4.3 The algorithm

In this section, the algorithm of comparing columns for the separation of foreground from background music is described. The algorithm is an iterative method applied to the spectrogram of a signal. The iterations consist of at most 5 steps. After iterating, the spectrogram is transformed back to a time signal.

In the description of every step, an example is provided. The method is intended for spectrograms of audio, which have high dimensions, but it can be unclear to see the effect of a single step in these matrices. For this reason, we introduced (variants of) a 13×10 binary matrix, the simplified spectrogram, (figure 4.1) to illustrate the steps in the examples.

4.3.1 Step 1: Determining overlap of columns

The first question we want to answer is when the $\mathbb{R}^n$ columns do overlap. For this, we will look at the distance between the column vectors. Let $X_{*,k}, X_{*,l} \in \mathbb{R}^n$ be the k-th and l-th column of the matrix $X \in \mathbb{R}^{n\times m}$. We say that the columns k and l (for $k \neq l$) overlap when their distance in the L_1-norm is less than a chosen δ:

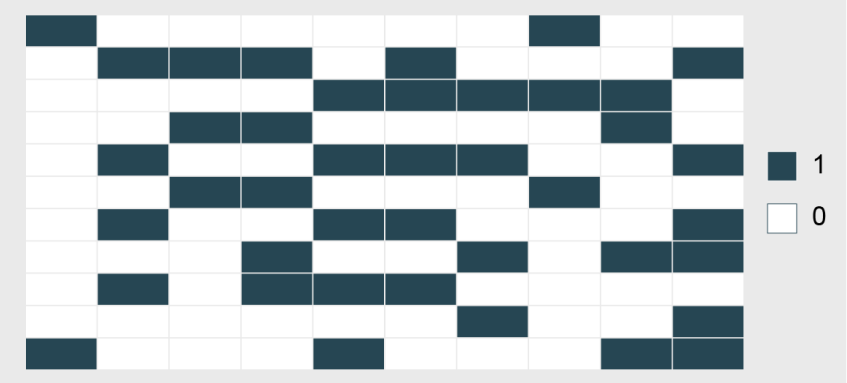

Figure 4.1: Simplified spectrogram used for illustrating the steps of the algorithm.

$$O(k,l) = \begin{cases} 1, & ||X_{*,k} - X_{*,l}||_1 < \delta \\ 0, & ||X_{*,k} - X_{*,l}||_1 \geq \delta \end{cases} \tag{4.1}$$

From now on, we will assume that overlapping columns are less likely to contain the vocals since vocals, with their non-percussive character, and their variation in lyrics and intonation, tend to be highly variable and hence non-repetitive, by their very nature.

Later on, in section 4.4.1, we will conclude that δ has to depend on the dimensions of matrix X and the average norm of its columns.

Example

Since the simplified spectrogram contains binary values, the distance between column k and l in the L_1-norm is the number of elements between the columns that do not match. This can of course be extended to real-valued columns by comparing the total distance between the elements of the columns with δ. In figure 4.2 and 4.3 the columns of the simplified spectrogram that overlap with the first ($k = 1$) and second ($k = 2$) column with $\delta = 4$ and $\delta = 5$ respectively, are indicated in pink.

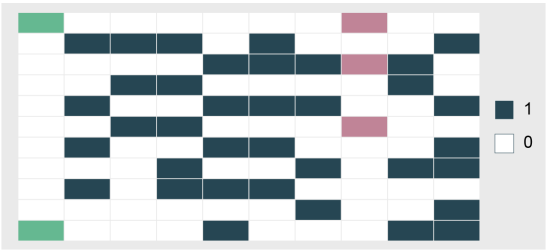

Figure 4.2: Simplified spectrogram indicating the overlap of column $k = 1$ (green) with other columns (pink) for $\delta = 4$

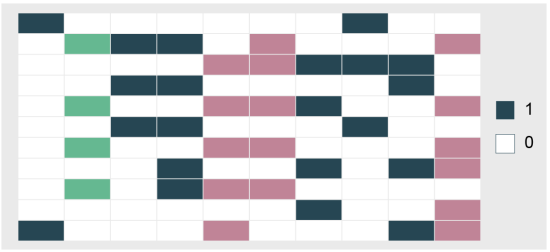

Figure 4.3: Simplified spectrogram indicating the overlap of column $k = 2$ (green) with other columns (pink) for $\delta = 5$

4.3.2 Step 2: Setting the minimum number of overlaps

The next step is determining a threshold for how often a column should overlap if we want to use it for filtering. We will use a shifted Heaviside step function as an activator on the sum of

overlaps minus 1 (for the case $l = k$):

$$A(k) = \begin{cases} 1, & (\sum_{l=1}^{m} O(k,l)) - 1 \geq N \\ 0, & (\sum_{l=1}^{m} O(k,l)) - 1 < N \end{cases} \tag{4.2}$$

The columns that activate have at least N occurring overlaps. Since the chance of an overlap increases as the duration of the song increases, N should depend on the duration of the song. Furthermore, when δ increases, the number of overlaps does too. So N should be chosen wisely after choosing δ. The first column that activates will now be used in further calculations.

Example

To clarify this step of the algorithm, we will again look at the simplified spectrogram. In our example, we only want to continue with a column that has at least three overlaps, so $N = 3$. In figure 4.2 and 4.3 we indicated these overlaps for column 1, with $\delta = 4$, and 2, with $\delta = 5$, respectively. It is clear that for $\delta = 4$ the number of overlaps of column 1 is not high enough to activate the function. Even for $\delta = 5$, column 1 only overlaps twice: with the columns $l = 8$ and $l = 9$. The second column, however, will activate the step function for $\delta = 5$ and $N = 3$. Since this is the first column for which this happens, the algorithm will continue with this column.

4.3.3 Step 3: Finding important frequency components

To prevent filtering out the whole signal, we are only interested in suppressing the repetitive frequencies that are of importance. In the previous step, we already determined which columns are of importance by counting their overlaps with other columns and applying an activator function. Now we will compare the amplitudes of every frequency in the selected important column to the average amplitudes of the row the frequency is in. By only continuing with frequencies in the column that have amplitudes larger than a certain value, we make sure we only subtract the frequencies of the column that affect the signal significantly and thus do not lose too much important information by subtracting small values or noise from every cell. We introduce a constant $r \in \mathbb{R}_{>0}$ for the minimum ratio between the amplitude of the column and the average amplitude of the row it is in.

$$|X_{m,k}| > r\frac{1}{M}||X_{m,*}||_1 \tag{4.3}$$

Increasing r results in filtering with frequencies that have a larger magnitude. As long as $r \gg 0$ we get what we desire: ignoring the frequencies that have magnitudes really close to 0. Larger magnitudes affect the filtering process the most, so a slight difference in r does not change the results significantly. But magnitudes around the average can be really important, especially for constant signals, so we want r to be less than 1. We choose $r = 0.5$ in further calculations.

Example

In previous examples, the steps were illustrated with the help of the simplified spectrogram, a binary matrix, from figure 4.1. Since every frequency component with a value of 1 is larger or equal to the average of the row it is in, we need to allow more values in the illustration to show the effect of this step. Figure 4.4 shows an extended version of the simplified spectrogram. Some

m	$\frac{1}{M}\|\|X_{m,*}\|\|_1$
1	0.114
2	0.450
3	0.322
4	0.251
5	0.412
6	0.161
7	0.221
8	0.312
9	0.352
10	0.201
11	0.311

Table 4.1: Table containing the average amplitudes of the rows in the extended spectrogram of figure 4.4.

values are changed from 1 to 0.5 or 0.1 and at certain points, a noise of 0.01 is added. It is possible that these slight changes affect the selection procedure of the columns in the previous steps, but, for simplicity and illustration purpose, we will neglect that and continue with the example for column $k = 2$.

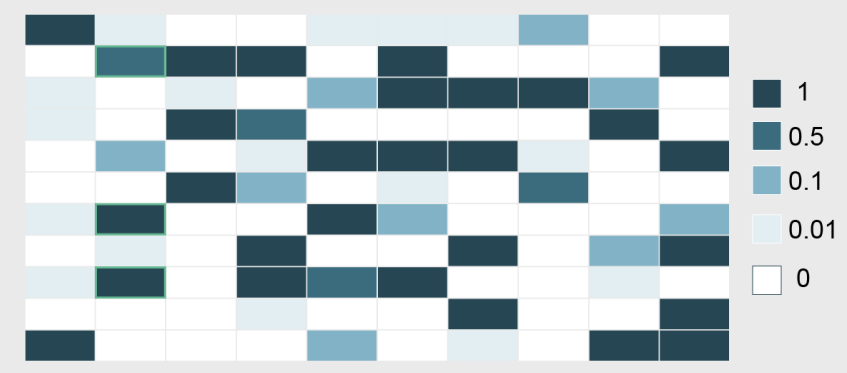

Figure 4.4: Extended spectrogram with noise, indicating the important frequency components of column $k = 2$ for $r = 0.5$ with a green border.

In table 4.3.3 the average amplitude of every row m in the extended spectrogram is denoted. These values are used to determine which frequency components of the second column activate. After multiplying the average amplitudes of the rows with $r = 0.5$ and comparing them to the frequency components of the second column, we obtain the important frequency components which are indicated with a green border in figure 4.4. Only the frequency components of the second column in row $m = 2$, $m = 7$ and $m = 9$ are of importance. The noises in row $m = 1$ and $m = 8$ are, correctly, not classified as important. The frequency component in column $m = 5$ is ignored since its amplitude is not high enough compared to the average amplitude of the row it is in.

4.3.4 Step 4: Subtracting frequency components

Only certain important frequencies of the overlapping column k remained from the previous step. These components will be compared with the other columns and subtracted under certain circumstances.

Notice that we decided to perform the calculations on the real and imaginary parts of the $\mathbb{C}^{n\times m}$ separately. This comes in handy for the comparison of the amplitudes: comparing complex numbers is difficult since the complex numbers are not an ordered field [18]. It is possible to compare their magnitudes and phases separately. But the real numbers can simply be compared using their ordering or their sign and absolute value.

For every remaining frequency component m of column k, we compare the amplitude with the amplitude of the m-th component of every other column l. When the frequency components $X_{m,k}$ and $X_{m,l}$ have the same sign, we set $X_{m,l}$ to zero if the magnitude of the component of the k-th (overlapping) column is larger than the magnitude of the corresponding frequency component in the l-th column. Otherwise, the m-th component of the overlapping column is subtracted from column l. Frequency components $X_{m,k}$ and $X_{m,l}$ with opposite signs are ignored. They are likely covered by a future overlapping column. Ignoring them now also reduces the chance of incorrectly subtracting frequency components.

In total there are thus 3 cases for subtracting the real-valued amplitudes. When $X_{m,k}$ and $X_{m,l}$ have the same sign and $X_{m,l} \neq 0$, we subtract or set $X_{m,l}$ to zero:

$$X^*_{m,l} := \begin{cases} X_{m,l} - X_{m,k}, & |X_{m,k}| \leq |X_{m,l}| \\ 0, & |X_{m,k}| > |X_{m,l}| \end{cases} \tag{4.4}$$

And when $X_{m,k}$ and $X_{m,l}$ are of opposite sign or $X_{m,l} = 0$, the values remain unchanged:

$$X^*_{m,l} := X_{m,l} \tag{4.5}$$

The frequency components in the row of a component of column k that is not of importance remain also unchanged.

$$X^*_{m,l} := X_{m,l}$$

for all k such that $|X_{m,k}| \leq r\frac{1}{M}||X_{m,*}||_1$.

The reason why we have to set some frequency components to zero is that subtracting the important frequency components from all other non-zero components in their row, has devastating consequences for the structure of the signal.
For example, subtracting an important frequency component with an amplitude of 1 from a component with an amplitude of 0.1 would result in a filtered component with an amplitude of 0.9. This is undesired since we do not want the amplitudes to become larger in our filtered spectrogram. Setting these columns to zero means only subtracting 0.1 and thus remaining with a filtered frequency component of 0.
In theory, it is possible that the foreground and background music have components in antiphase and thus the amplitude of a component in the filtered (foreground) spectrogram should be higher than in the original spectrogram, but it is highly unlikely that this occurs often, including through the assumption that the accompaniment is highly periodic and the vocals are not.

Example

In the previous step we classified the frequency components in row $m = 2$, $m = 7$ and $m = 9$ from column $k = 2$ of the extended spectrogram as important. Now we will apply the steps of subtraction out of this section to these components. Since this step of the algorithm leaves frequency components with opposite signs unchanged, we can again use the extended spectrogram (with only positive values) to demonstrate the effects of this step. The resulting filtered spectrogram is shown in figure 4.5.

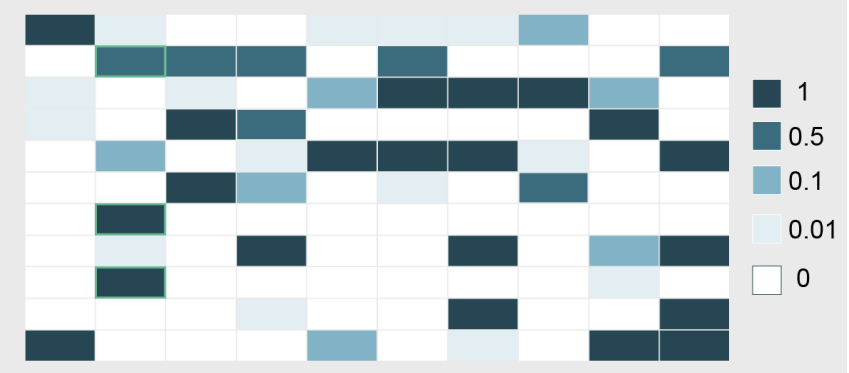

Figure 4.5: Filtered extended spectrogram obtained after subtracting the important frequency components $X_{2,2}$, $X_{7,2}$ and $X_{9,2}$ (indicated in green) from the extended spectrogram in figure 4.4.

First we will look at the second row. Notice that $|X_{2,2}| = 0.5 < |X_{2,l}| = 1$ for $l \in \{3, 4, 6, 10\}$. This corresponds to the first case of equation 4.4, in which we subtract $X_{2,2}$ from the other components:

$$X^*_{2,3} = X^*_{2,4} = X^*_{2,6} = X^*_{2,10} = 1 - 0.5 = 0.5 \tag{4.6}$$

Since $X_{2,l} = 0$ for $l \in \{1, 5, 7, 8, 9\}$, the values of the other components in the second row remain unchanged, as stated in equation 4.5:

$$X^*_{2,1} = X^*_{2,5} = X^*_{2,7} = X^*_{2,8} = X^*_{2,9} = 0 \tag{4.7}$$

Note that the components of the rows $m = 7$ and $m = 9$ of column $k = 2$ are all a maximum in their row:

$$X_{2,7} = X_{2,9} = 1 \tag{4.8}$$

Thus for these rows, the first case of equation 4.4 is only satisfied when components in other columns have an amplitude equal to this maximum of 1. In this case we subtract the frequency components $X_{7,2}$ or $X_{9,2}$:

$$\begin{aligned} X^*_{7,5} &= X_{7,5} - X_{7,2} = 1 - 1 = 0 \\ X^*_{9,4} &= X_{9,4} - X_{9,2} = 1 - 1 = 0 \\ X^*_{9,6} &= X_{9,6} - X_{9,2} = 1 - 1 = 0 \end{aligned}$$

As discussed in this section, subtracting $X_{7,2}$ and $X_{9,2}$ from all the other non-zero components would disturb the signal significantly. For this reason, we set these components to zero such

that their filtered frequency component has no magnitude as described in the second case of equation 4.4. This means that $X_{7,2}$ will set $X_{7,1}$, $X_{7,6}$ and $X_{7,10}$ to zero, and that $X_{9,2}$ will set $X_{9,1}$, $X_{9,5}$ and $X_{9,9}$ to zero to obtain their filtered values.

$$X^*_{7,1} = X^*_{7,6} = X^*_{7,10} = 0$$
$$X^*_{9,1} = X^*_{9,5} = X^*_{9,9} = 0$$

All the other frequency components in the matrix of the extended spectrogram remain unchanged.

4.3.5 Step 5: Clearing the overlapping column and iterate

Finally, after comparing (and possibly subtracting) the frequency components of the overlapping column k with all the other columns, we set the column itself to zero. This is in line with the assumption that the most overlapping columns are less likely to contain the voice.

$$X^*_{*,k} = 0$$

Now we are done with the k-th column. We can continue to the next column $(k+1)$ and perform the same steps as we did for column k, but now we take the filtered spectrogram X^* as input. Repeat this process for all the columns of the spectrogram.

To make sure that the algorithm described in this section works for multiple iterations, some slight changes have to be made. These are discussed in the next section.

Example

In our example, we performed calculations on column $k = 2$. This was the first occurrence of a column that overlapped at least $N = 3$ times using an overlap threshold of $\delta = 5$. In the last step of this iteration, we clear the k-th column. The results are shown in figure 4.6.

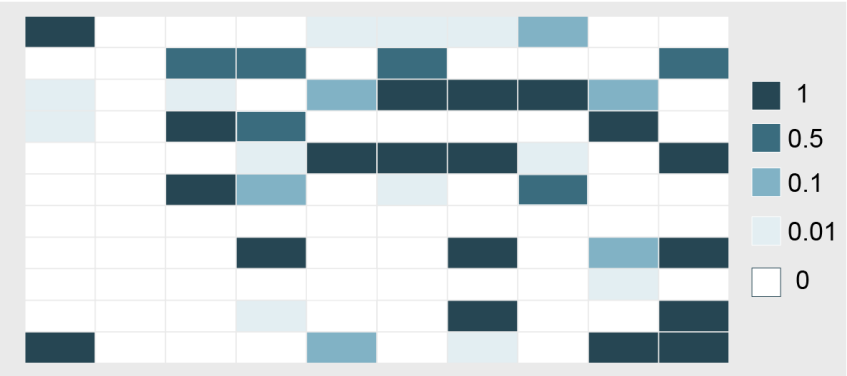

Figure 4.6: Filtered extended spectrogram after clearing the overlapping column k.

Now we are done with the overlapping column $k = 2$ and can thus continue to the next overlapping column. We start over at step 1 (section 4.3.1 of the algorithm, but perform all the calculations on the filtered spectrogram X^* from this iteration instead of the original spectrogram X. The next iteration will thus be performed on the first column $k > 2$ that has at least $N = 3$ overlaps for $\delta = 5$.

4.3.6 Step 6: Obtaining time signals from the spectrogram

After iterating over the columns of the matrix and subtracting overlapping columns, we can calculate the time signal of the foreground by applying the inverse spectrogram techniques, which are discussed in section 2.5.2. These calculations are straightforward since our original spectrogram was constructed without any overlap.

4.4 Clarification of thresholds

4.4.1 Maximum distance for overlap: δ

The threshold δ determines whether or not columns of the spectrogram overlap. There is no rule of thumb for what the value of δ has to be for every spectrogram. Indeed, it depends on the dimensions (resolution) of the spectrogram. A spectrogram with smaller short-time Fourier transform segments has a higher dimension of columns (time segments) axis and a lower dimension of rows (frequencies), and is thus more likely to overlap than a matrix with more segments.

Besides the choice of δ depending on the dimensions of the spectrogram, we have to take into account that the filter is an iterative process. In every step components of an overlapping column are subtracted from other columns, which decreases the total magnitude of the spectrogram and thus the total distance between columns. Since the distance between columns determines whether or not they overlap, it is a good idea to decrease the constant δ after each iteration. If we do not scale δ, it is possible that after some iterations every column overlaps. Scaling δ with the average norm of the columns solves this issue. We will thus replace the constant δ with a function that depends on the magnitudes of X:

$$\delta^*(X) = \delta \frac{1}{m} \sum_{k=1}^{m} ||X_{*,k}||_1$$

In figure 4.7 and 4.8 the filtered spectrogram of "Blondie - Call me" for respectively the constant and scaled δ is shown.

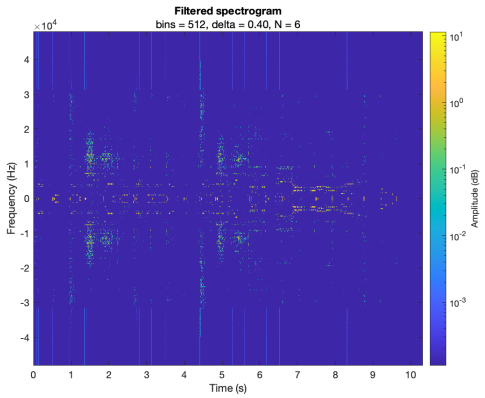

Figure 4.7: Filtered spectrogram of "Blondie - Call me" with constant δ

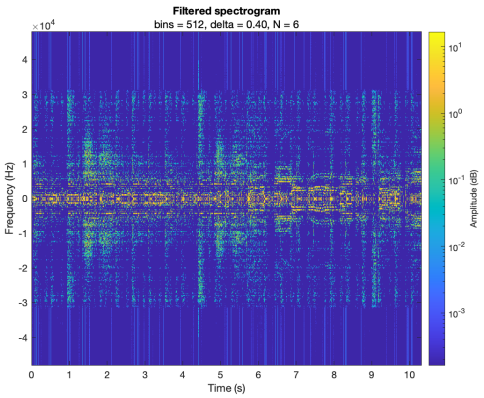

Figure 4.8: Filtered spectrogram of "Blondie - Call me" with scaled δ^*

Example

The issue that is raised and the solution that is proposed are illustrated by the filtered extended spectrogram in figure 4.6. After subtracting the important frequency components from other

components in its row and clearing the overlapping column, the distance between columns in terms of the L_1-norm is reduced or remained the same.

As seen in figure 4.2, column $k = 1$ does not overlap with column $l = 5$ for $\delta = 4$. Even in the extended spectrogram (figure 4.4) they do not overlap, since $||X_{*,1} - X_{*,5}||_1 = 4.47 > 4 = \delta$. However, after filtering the overlapping column $k = 2$, the distance between the third and second column is reduced: $||X^*_{*,1} - X^*_{*,5}||_1 = 2.99 < 4 = \delta$. A small disclaimer: the second column is classified as overlapping for $\delta = 5$, while figure 4.4 uses $\delta = 4$. The algorithm for $\delta = 5$ will thus perform slightly different, but the idea remains the same: columns that are not indicated as overlapping at previous iterations, can be classified as overlapping in a further iteration due to filtering. Redefining when columns overlap after every iteration, for example by scaling δ with the average norm of the columns, could resolve this problem.

4.4.2 Minimum ratio of frequency components: r

Similar to the δ, the comparison that determines the importance of a frequency component (equation 4.3) is also dependant on magnitudes in the spectrogram. We introduced this comparison to prevent subtracting frequencies that are or become too small. Scaling this comparison after every iteration with the new magnitudes contradicts the motivation we had to introduce the comparison: determining the importance of frequency components such that low amplitude components or noise are ignored. The right-hand side of equation 4.3 can thus be calculated before iteration, on the unfiltered spectrogram X and remains constant. For the ratio r, a constant can be used too.

4.4.3 Subtraction of all frequency components

In the step of subtracting the frequency components (section 4.3.4), we not only subtract the overlapping column k from the columns it overlaps with, but also from all the other columns. This can feel counter-intuitive. What do other columns have to do with the k-th column, if they do not even overlap? Do we need another threshold to prevent filtering these columns?

In section 4.3.1, we assumed that overlapping columns are less likely to contain the vocals. Subtracting only the overlapping columns will thus result in filtering primarily the background music. If the filter works optimally, the amplitude of the filtered signal will be zero everywhere, except at times where there are (loud enough) vocals.

Having a spectrogram that is zero everywhere, except at times the vocals occur could have a useful purpose in signal-processing for speech recognition, but it is unfortunately not directly helpful for our current case of separating the foreground vocals from the background music. By filtering out these columns, all the other information that we have about the parts of the background music without vocals will be filtered too and cannot be used in a later iteration of our method.

But those parts are exactly what we need for separating the vocals. When we are able to subtract just the background music parts from the background and foreground parts, we possibly remain with the foreground parts. It just so happens that by our assumption the overlapping column k is most likely a column consisting of background music only. At a different time, the background music can be almost identical, but vocals can be added. When the vocals have frequencies that are also in the background music, the amplitudes for these frequencies increase. Subtracting at most the amplitude of the background music from this signal should leave the

remaining foreground music intact.

Filtering the background music from non-vocal parts could for example be used in the determination of the end of words or sentences. With a bit of creativity, the non-vocal background music parts can have a use case in separating the voice: interpolation of missing samples [14] can be used to reconstruct the background music under the vocals. The autoregressive model may even be improved since we have knowledge (the combined signal of the foreground and background) about the signal for the missing samples. If the interpolation is accurate, the vocals could be found by subtracting the interpolated background music signal from the original signal. This is out of the scope of this project, but maybe a reader is interested in looking further into this.

Chapter 5

Implementation and demonstration

5.1 Implementation in Matlab

The implementation in Matlab played an important role during the construction of the method of separating voice by comparing spectrogram columns. Without seeing the spectrograms or listening to the filtered audio fragments, one will have a hard time constructing a method that is applied to spectrograms to filter an audio fragment. The construction of the method consisted of constantly switching between the theory and the implementation. The algorithm requires several input parameters:

- the filename of the .wav file of the song,
- the window type, e.g. `hamming(512, 'periodic')`,
- the number of samples that overlap between segments of the spectrogram,
- the number of frequencies in the spectrogram,
- a range of values for the parameter δ,
- a range of values for the minimum number of overlaps N,
- the ratio r as discussed in section 4.3.3.

The final version of the implementation of the separation algorithm can be found on GitHub in the file `compare_columns.m`: `https://github.com/JoepdeJong/comparing-spectrogram-columns`.

5.2 Demonstration

An online demonstration is supplied for the readers that prefer perceiving the results of the method of comparing spectrogram columns with their eyes and ears over reading about it. The demonstration contains the filtered spectrograms and audio signals of different songs. The results are calculated using the implementation described in section 5.1. The script (`demo.m`) for generating all the results for the demo is also provided in the GitHub repository of the project.

The demonstration can be found on `https://joepdejong.com/bep`.

Chapter 6

Discussion and conclusion

6.1 Comparison with the 2DFT method

Periodicity of the signals

The separation method using the two-dimensional Fourier transform and the method of comparing spectrogram columns have a lot in common. There are, however, some significant differences. The main difference between the separation method using the two-dimensional Fourier transform and the method of comparing spectrogram columns is the way they determine periodic components of the song.

The method using the two-dimensional Fourier transform decomposes the spectrogram into sinusoidals with strict periods. Stricly periodic parts of a song are transformed into an individual sinusoidal in the scale-rate domain. These parts give clear peaks that can be located using the method. Parts of a song that are slightly off-beat are technically non-periodic, these parts will be represented by several sinusoidals with different frequencies in the scale-rate domain. Similar to the vocals, the energy of these parts will be spread out across their spectrum in the scale-rate domain, which makes it hard to distinguish them.

Instead of using the two-dimensional Fourier transform, the proposed method compares spectrogram columns with each other. The comparison does not only happen between columns that have a constant number of columns between them, which could ensure periodicity. But every column is compared with every other column. This makes the comparing column method seem to be better suited for slightly off/non-periodic, bit repetitive, signals.

Finding the repetitions

The Fourier transform already takes care of finding the number of repetitions by transforming to frequencies. The local maxima correspond exactly to the most repetitive frequencies. When comparing columns an additional step is introduced to find repetition: counting the number of overlaps (columns that are in the L_1-norm closer than δ to the current column, equation 4.1) and comparing (equation 4.2) them to a threshold N, which indicates the minimum number of overlaps. The number of overlaps is, in terms of the criterion that is used for comparing the columns, exactly the number of repetitions of a column throughout a song. The threshold N, used for selecting columns, is thus a threshold in the horizontal direction, along the time axis, of the spectrogram.

Importance of frequencies

The constant r, which defines the minimum ratio between the amplitude of a column and the average amplitude of its row, is introduced for the same reason that the threshold γ is: ensuring that only frequencies with a high enough amplitude are filtered. The thresholds r and γ can thus be interpreted as thresholds in the vertical direction, along the frequency axis, of the spectrogram. Scaling the threshold r with the average amplitude can be seen as an improvement in comparison to the 2DFT-method.

Masking versus subtracting

The 2DFT-method uses a binary mask to determine whether or not a frequency component is in the background or foreground. Because of this, the separation of foreground and background signals that partly consist of the same frequencies at a certain time is impossible. The human voice consists of a wide range of frequencies, which makes it plausible that frequencies overlap with the accompaniment. This is not desired. The comparing column method takes care of this problem by not subtracting more than the value of the important frequency components from the repetitive columns from other columns. Variation of the amplitude caused by the human voice can thus remain if the amplitude of its frequency components throughout the song is larger than the amplitude of the repeating important frequency component from the overlapping column that is compared to them.

To prevent filtering out non-periodic frequency components that are louder than the background components, the 2DFT-method uses a second mask (section 3.3) that compares the magnitudes. It thus makes a decision between classifying a point in the spectrogram as foreground or as background, a combination is not possible.

6.2 Suggested improvements

Method for the selection of columns

Currently, the first column that has more than N overlaps is used in filtering. The important frequencies of this column are subtracted from all the other columns. After that, the first column after column with at least N overlaps is used for subtraction, and so on.
It is possible that initially there was a column that overlapped with a lot more columns than the column that was used first by the algorithm. To find this column we have to calculate in every iteration the distance between all columns. The time complexity of this calculation is an order higher than finding the next column with more than N overlaps.

Knowing that we are working with a lot of samples, increasing the order of time complexity is undesired. If time is not a problem for the reader, it could of course be implemented. But the question remains if it performs better:

The above procedure for selecting the most overlapping column changes of course also the order of the used columns. By looking further at the overlapping columns of the original procedure, one can notice that they are mainly positioned at the start and end of the song. These columns seem to correspond with the instrumental intro and outro of a song. Using the above procedure does not take advantage of the instrumental intro and outro, but it does suggest using another order: alternating between the sequences of increasing and decreasing columns 1, 2, 3, ... and m, $m-1$, $m-2$, ...

Changes in accompaniment throughout a song

The important frequency components of the overlapping column are subtracted from every other column. It could be possible that the background music changes throughout the song. If this is the case, then the chance increases that the method of shifting columns subtracts the frequency components from columns that are not part of the accompaniment. The vocals will be corrupted.

The suggested solution for this problem is looking at the density of the overlapping columns. When the number of columns between the overlapping columns is large and thus less repetitive, it is less certain that the columns are part of the repetitive background signal. A threshold may be introduced to ensure that this distance remains small enough. If, after some steps, the distance between the columns exceeds the threshold, then we assume that the accompaniment has changed.

A drawback of this threshold is that background sounds with a large period will not be filtered anymore. However, this can maybe be resolved by scaling the threshold with the average distance between the overlaps in the first place, which is a similar solution as scaling δ with r with the average magnitude.

Filtering the magnitude-phase representation of the spectrogram

At the end of section 4.3.4 it is stated that it is in theory possible that components of the foreground and background music are in antiphase. This raises the question of looking at the real and imaginary parts of these components separately is the best practice for filtering the accompaniment using the method of comparing spectrogram columns. Filtering spectrograms, that are in magnitude-phase representation, separately on their magnitude and phase parts will probably give different results and is expected to have more in common with the 2DFT-method, which filters on the magnitude spectrogram.

Combining filters

In this thesis, all the filters are discussed separately. For follow-up research, it would be interesting to look at the combination of the filters discussed in this topic. Does applying a bandpass filter before using the two-dimensional Fourier transform method improve the accuracy? And what about filtering frequencies with low amplitudes before applying the method of comparing spectrogram columns? From the demonstrations of both the 2DFT and comparing spectrogram columns methods, one could also notice that the preserved vocals for the same song are quite similar, but the noise in the background differs a lot. Investigating the combination of these filters could be interesting too.

6.3 Conclusion

The purpose of this thesis was to analyze the possibilities of separating the voice of a singer from a song to improve the understanding of a neural network. To do so, several filters were evaluated. Two filters are treated in detail: The existing method of separating voice from a repetitive background accompaniment using the two-dimensional Fourier transform and the proposed method of comparing spectrogram columns.

The proposed method is compared with the existing method in several ways to substantiate the choices that are made while constructing this new method. The differences in the expected

behavior are also described.

The motivation behind the construction of the method of comparing spectrogram columns was finding a method that performs well on time signals that have background accompaniment with slightly off periods. The method that is proposed takes this into account and the first results are truly promising.

Bibliography

[1] Alex Amenta. *Lecture notes: Fourier analysis*. 2018.

[2] Luis F. Chaparro and Aydin Akan. "Chapter 8 - Sampling Theory". In: *Signals and Systems Using MATLAB (Third Edition)*. Ed. by Luis F. Chaparro and Aydin Akan. Third Edition. Academic Press, 2019, pp. 449–485. ISBN: 978-0-12-814204-2.

[3] R. Crochiere. "A weighted overlap-add method of short-time Fourier analysis/Synthesis". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.1 (1980), pp. 99–102. DOI: 10.1109/TASSP.1980.1163353.

[4] Remi Julien Blaise Decorsière. "Spectrogram inversion and potential applications for hearing research". English. PhD thesis. 2013.

[5] W.R. Dieter, S. Datta, and Wong Key Kai. "Power reduction by varying sampling rate". In: (2005), pp. 227–232. DOI: 10.1145/1077603.1077658.

[6] Marco F. Duarte. *Signals and Systems*. 2017.

[7] H.L.F. von Helmholtz. "Die Lehre von den Tonempfindungen als physiologische Grundlage für die Theorie der Musik". In: *F. Vieweg Sohn* (1863).

[8] Holger Hennig, Ragnar Fleischmann, and Theo Geisel. "Musical rhythms: The science of being slightly off". In: *Physics Today* 65 (July 2012), pp. 64–65. DOI: 10.1063/PT.3.1650.

[9] Hilmar. *Inverse Short Time Fourier Transform algorithm described in words*. 2012. URL: https://dsp.stackexchange.com/questions/2757/inverse-short-time-fourier-transform-algorithm-described-in-words.

[10] M.-V Laitinen, Sascha Disch, and Ville Pulkki. "Sensitivity of Human Hearing to Changes in Phase Spectrum". In: *AES: Journal of the Audio Engineering Society* 61 (Nov. 2013), pp. 860–877.

[11] Laurent Navarro, Guy Courbebaisse, and Jean-Charles Pinoli. "Continuous frequency and phase spectrograms : A study of their 2D and 3D capabilities. Application to musical signal analysis". In: *Journal of Zhejiang University SCIENCE A* 9 (Feb. 2008), pp. 199–206. DOI: 10.1631/jzus.A072140.

[12] G. S. Ohm. "Ueber die Definition des Tones, nebst daran geknüpfter Theorie der Sirene und ähnlicher tonbildender Vorrichtungen". In: *Annalen der Physik* 135.8 (1843), pp. 513–565.

[13] Matt Ottewill. "Advantages disadvantages of analogue digital audio". In: *Planet of Tunes* (2015).

[14] Laurent Oudre. "Interpolation of Missing Samples in Sound Signals Based on Autoregressive Modeling". In: *Image Processing On Line* 8 (Oct. 2018), pp. 329–344. DOI: 10.5201/ipol.2018.23.

[15] Prem Seetharaman, Fatemeh Pishdadian, and Bryan Pardo. "Music/Voice separation using the 2D fourier transform". In: (2017), pp. 36–40. DOI: 10.1109/WASPAA.2017.8169990.

[16] Standard Vinyl. "Full-length LP records on 150 and 180 gram vinyl". In: *Standard Vinyl* (2018).

[17] Ruye Wang. *Two-Dimensional Fourier Transform*. 2007. URL: http://fourier.eng.hmc.edu/e101/lectures/Image_Processing/node6.html.

[18] Richard C. Weimer. "Can the Complex Numbers Be Ordered?" In: *The Two-Year College Mathematics Journal* 7.4 (1976), pp. 10–12. DOI: 10.1080/00494925.1976.11974455.