Insert here your thesis' task.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master’s thesis

Exploring use of non-negative matrix factorization for lossy audio compression

Bc. Tomáš Drbota

Department of Theoretical Computer Science
Supervisor: doc. Ing. Ivan Šimeček, Ph.D.

April 4, 2019

TODO

Acknowledgements

TODO

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on April 4, 2019 ....

Citation of this thesis

Drbota, Tomáš. *Exploring use of non-negative matrix factorization for lossy audio compression.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

TODO

Klíčová slova TODO

Abstract

Non-negative matrix factorization has been successfully applied in various scenarios, mostly for analyzing large chunks of data and finding patterns in them for later use. Due to the nature of NMF, it has also seen some use in the field of image compression.

The purpose of this thesis is to research possible uses of non-negative matrix factorization in the problem of audio compression. A reference audio encoder and decoder using NMF will be implemented and various experiments using this encoder will be conducted. The results will be measured and compared to existing audio compressing solutions.

Keywords lossy, audio, compression, processing, nmf, encoding

Contents

List of Figures

List of Tables

Introduction

In today's age of smartphones and other portable electronic devices capable of connecting to the internet, nearly everyone has access to this giant (and still growing) library of various media, including music and other audio. However, to transmit or store all of this data in its raw uncompressed form, a large amount of bandwidth and storage would be required.

.. need for compression ..

.. common methods of audio compression ..

.. mp3 opus ..

.. this work tries nmf ..

.. state of art ..

.. then design and implement ..

.. measure results ..

Part I

Background

Chapter 1

Digital audio

Sound as we know it can be defined as a physical wave travelling through air or another means. [1] It can be measured as change in air pressure surrounding an object. Once we have this electrical representation of the wave, we can convert it back and consequently play using speakers.

In the real world, these sound waves are generally composed of many different kinds of waves, with differing frequencies and amplitudes. The human ear can tell the difference between high (whistling) and low frequencies (drums), and knowledge of this will be useful later when we are discussing audio encoding.

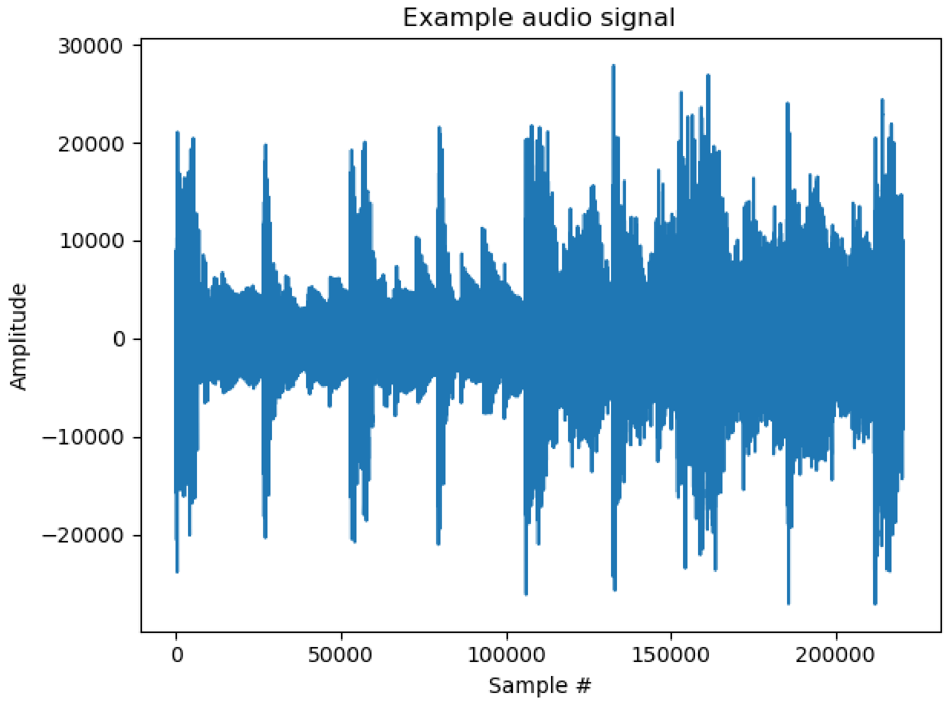
1.1 Notation

Below is a summary of the notation this chapter will be using.

Table 1.1: Digital audio notation

t	symbol representing a time value in seconds
τ	symbol representing a "slow" time, time index with a lower resolution than t
ξ	symbol representing a frequency value in hertz
F_s	symbol representing a sampling rate of an audio signal
$x(t)$	function representing the amplitude of a continuous signal at a time t
x_n	sequence representing the amplitude of a discrete signal indexed by n
$w(t)$	continuous windowing function at a time t
w_n	discrete windowing function indexed by n
$S(\xi)$	the Fourier transform of a continuous signal
S_k	the discrete Fourier transform of a discrete signal
$S(\tau, \xi)$	the short-time Fourier transform of a continuous signal
$S_{k,\xi}$	the discrete short-time Fourier transform of a discrete signal
M_k	the Modified discrete cosine transform of a discrete signal

Figure 1.1: An example of an audio signal represented in PCM form.



1.2 Important terms

In this section several terms used throughout this thesis will be described.

1.2.1 Sampling

Sampling in this context refers to the way we convert an analogue signal into a digital one. When we use something like a microphone, what's happening is that it measures the pressure waves generated by the sound around it at regular intervals. The result of this operation is the amplitude of the wave, or the size of the vibration at that point in time.

To obtain an actual usable sound wave of various frequencies and amplitudes, we will need many samples - generally tens of thousands per second. The rate at which we collect these samples is called the *sampling rate*, denoted F_s.

1.2.2 Nyquist frequency

It is proven that we can fully represent an arbitrary signal $x(t)$ by its samples x_n [2], but there are some conditions to this. If we don't sample a given signal enough times per second, there might be frequencies high enough that our sampler won't be able to record them correctly.

This is called the *Sampling Theorem* and the idea is that we can only accurately record all the frequencies in a signal if our sampling rate F_s is at least twice as large as the largest frequency contained in the signal F_{max}. [3]

In essence, a continuous signal can only be fully represented by its samples if:

$$F_s \geq 2F_{max} \tag{1.1}$$

Looking at it from another angle, it also means that given a sampling rate F_s, the highest frequency we can record is equal to $\frac{F_s}{2}$, and that is what's called the *Nyquist frequency* or *Nyquist limit.*

1.2.3 Quantization

Quantization is a process where we restrict a large set of values, possibly continuous, into a smaller set, generally discrete.

For example, when we sample an analogue signal, what we get back is the amplitude represented by a voltage, which can be considered a set of infinitely many real numbers. In order to use these values on our computers for digital processing, these voltages must first be quantized. [2] An example of a common choice would be signed 16-bit PCM, where each sample is converted to an integer between $-2^{15} = -32768$ and $2^{15} - 1 = 32767$, representing the quantized amplitude for the given sample.

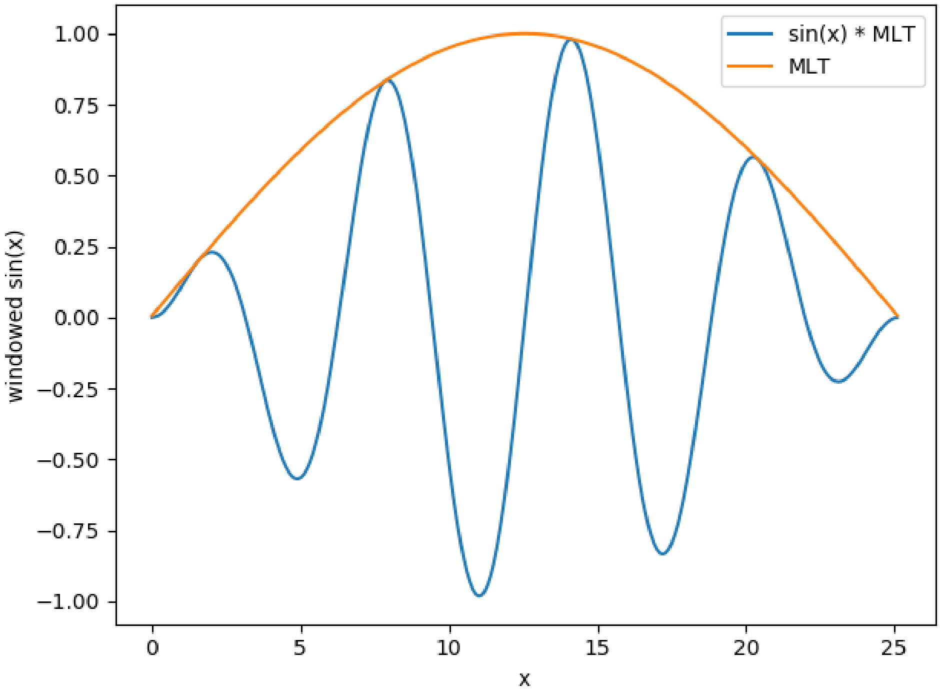
Quantization is also common in digital audio processing when we already have discrete values, e.g. when quantizing MDCT values (described later) in order to further reduce the range and allow for more efficient compression.

1.2.4 Windowing

A windowing function w_n is a function that is equal to 0 outside of some chosen interval, where its value is defined by an expression. It is generally symmetrical, though this is not a rule. Windowing functions are often applied to audio signals by multiplying them with the windowing function before further processing to avoid various artifacts.

There is no one "best window", there's usually a trade off, for example a window used to prevent aliasing will lower the accuracy of frequency identification (as is the case in the Hann window). [2] In fact, most modern audio codecs use different kinds of windows depending on the musical characteristics of the current block and the ones that follow. [4]

Figure 1.2: An example of windowing: a sine function windowed by MLT.



1.2.4.1 Rectangular window

The rectangular window is the simplest one, as it does not modify the original signal at all, and as such is often used to represent e.g. a part of a periodic signal. It's equivalent to:

$$w_n^R = 1 \tag{1.2}$$

1.2.4.2 Hann window

A Hann window is a simple sine-based window, defined as:

$$w_n^H = \sin^2\left(\frac{\pi n}{N}\right) \tag{1.3}$$

This window is what we'll be using most of the time with the short-time Fourier transform.

1.2.4.3 Modulated lapped transform

Modulated lapped transform (MLT) is another sine-based window, used for example in the MP3 codec. It has a comparatively low computational complexity and is simple to implement. [5] It's defined as:

$$w_n^M = \sin\left[\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\right] \tag{1.4}$$

It is what we will be using with the MDCT due to its simplicity of implementation compared to the results it provides.

1.2.4.4 Kaiser-Bessel derived window

The Kaiser-bessel derived window (KBD) is designed to be used with the modified discrete cosine transform (MDCT), and is used for example in the AC-3 codec. It's defined as follows [2]:

$$w_n^D = \begin{cases} \sqrt{\frac{\sum_{i=0}^{n} w_i}{\sum_{i=0}^{\frac{N}{2}} w_i}} & \text{if } 0 \leq n < \frac{N}{2} \\ \sqrt{\frac{\sum_{i=0}^{N-1-n} w_i}{\sum_{i=0}^{\frac{N}{2}} w_i}} & \text{if } \frac{N}{2} \leq n < N-1 \\ 0 & \text{otherwise} \end{cases} \tag{1.5}$$

1.2.5 Transient

A transient is a high amplitude sound with a short duration. It's significant because during block-based audio encoding, if a transient occurs on the border between two blocks it may cause what's called *pre-echo* - essentially a type of artifact where a sound is being heard before it should occur.

To mitigate artifacts caused by transients, it's important to use proper windowing.

1.2.6 Aliasing

Aliasing is a type of artifact that occurs when the sampling rate is too low. Due to the way sampling works, frequencies above the *Nyquist limit* are mirrored to lower frequencies, creating a noticeable distortion. [2]

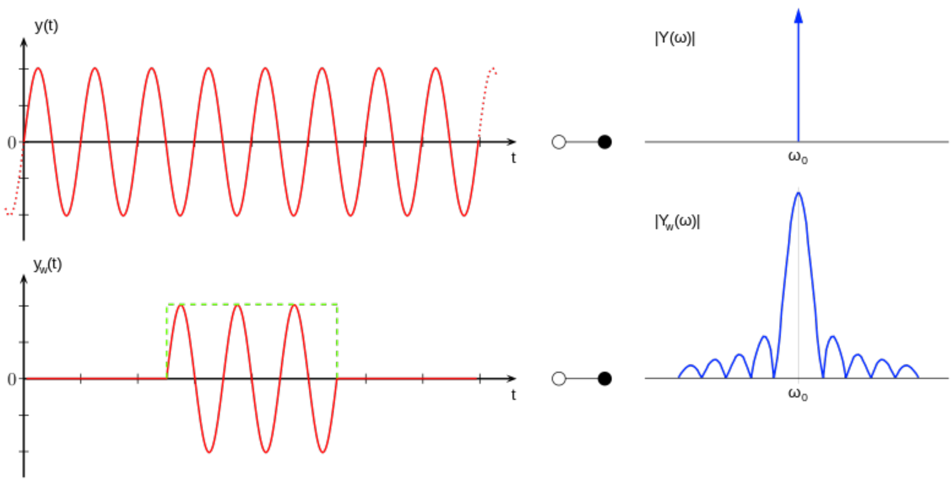
This can be eliminated by either choosing an appropriately higher sampling rate, or by passing the signal through a low-pass filter to eliminate the higher frequency content that would cause aliasing.

1.2.7 Spectral leakage

As we will see later, the Fourier transform (and other Fourier-related transforms) effectively projects the signal into infinity, as if it was a periodic signal. If we apply the transform to an unmodified signal, it's similar to using a rectangular window on a periodic function, where the signal ends abruptly at the edges. This in turn leads to *spectral leakage* - the resulting Fourier transform will produce non-zero values even for frequencies that aren't contained in the original signal at all.

The way to combat spectral leakage is again, by using proper windowing. By smoothing out the function using a window so that it approaches 0 along the edges of the signal, we can mitigate some spectral leakage, although not all of it.

Figure 1.3: An example of a sine function's spectral leakage after application of a rectangular window. Image by Walter Dvorak published under the CC BY 3.0 license.



1.2.8 Scalloping

A phenomena that occurs when using a discrete Fourier (or Fourier-related) transform. Since the frequencies are split into ranges rather than exact values, it's possible that a single frequency may fall into two adjacent bins, providing inaccurate frequency content readings.

To deal with scalloping, a large enough block size in relation to the sampling rate must be chosen to lower the ranges of the bins.

1.3 Digital audio representation

Most commonly, the amount of air pressure is sampled many times a second and after being processed this information is stored as a discrete-time signal using numerical representations - this is what's known as a *digital audio signal*. This entire process is called *digital audio encoding*.

By sampling the audio signal, we will potentially be losing out on some information, but given a high enough sampling rate, the result will be imperceptible to the human ear. For general purpose audio and music, the standard sampling rate is 48 kHz, alternatively 44.1 kHz from the compact disk era.

Once we have our digital signal, there are two distinct kinds of ways we can represent, or, encode it. Both of them have many different data models for encoding [1], but in this work I am only going to focus on the most relevant ones.

1.3.1 Time domain representation

In the time domain, the signal is simply represented as a function of time, where t is the time and $x(t)$ is the raw amplitude, or air pressure, at that point. [2]

This is the most straightforward representation since it directly correlates to how the signal is being captured in the first place. However, as we will see later, this format is not ideal for storing audio data with any sort of compression.

1.3.1.1 PCM

In the time domain, the most basic encoding we can use is PCM (Pulse Code Modulation). After sampling a signal at uniform intervals, the discrete values are quantized; that is, each range of values is assigned a symbol in (usually) binary code.

For example using 16-bit signed PCM, each sample will be represented as a 16-bit signed integer, or in the case of multiple channels, N 16-bit signed integers, where N is the amount of channels.

PCM serves as a good base for what we are going to talk about next - Frequency domain representation and encoding.

1.3.2 Frequency domain representation

While it's simple to understand and work with for the computer with samples in the form of a sequence of amplitudes, it's difficult to run any sort of meaningful analysis on such data. To better grasp the structure of the audio we're working with, it would be helpful to be able to decompose it into its basic building blocks, so to speak. And that's where frequency based representation comes in.

The goal here is to represent the signal as not a function of time, but rather a function of frequency $X(\xi)$. That is, instead of having a simple sequence of amplitudes, we will have information about the magnitude for each component from a set of frequency ranges. This description alone is generally more compact than the PCM representation [2] on top of providing us with useful information about the signal, so it will serve as a good entry point to our compression schemes.

1.3.2.1 Fourier transform

Fourier transform is the first and arguably the most used tool for converting a signal from a function of time $x(t)$ into a function of frequency $X(\xi)$.

It is based on the *Fourier series*, which is essentially a representation of a periodic function as the linear combination of sines and cosines. [6] However, the main difference is that our function need not be periodic.

The Fourier transform of a continuous signal x is defined as: [7]

$$S(\xi) = \int_{-\infty}^{\infty} x(t)e^{-2\pi it\xi}dt \tag{1.6}$$

If we inspect the formula, we can notice that Fourier transform essentially projects our signal into infinity - this wouldn't be a problem if it was a periodic signal, but sampled audio is generally constrained by time. To prevent spectral leakage as a result, we must window the signal before processing it. [8]

The output is a complex number, which provides us with the means to find the magnitude and phase offset for the sinusoid of each frequency ξ.

The Fourier transform can also be inverted, providing us with an easy way to obtain the original signal back from its frequency components. The inverse transform is defined as:

$$x(t) = \int_{-\infty}^{\infty} S(\xi)e^{2\pi it\xi}d\xi \tag{1.7}$$

However, seeing as our samples are discretely sampled, we will need to modify our transform accordingly.

The discrete Fourier transform of a discrete signal $x_0, x_1, ..., x_{N-1}$ is: [**?**]

$$S_k = \sum_{n=0}^{N-1} x_n e^{-2\pi ikn/N} \tag{1.8}$$

And our inverse is:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} S_k e^{2\pi ikn/N} \tag{1.9}$$

In the discrete form, rather than finding the frequency content for a specific frequency ξ, we find the content of the k-th frequency bin. Since we have fewer values to work with, the frequency range is quantized to a degree and each bin contains an uniformly sized range of frequencies rather than a specific one.

The way that works is as following: if we for example run the discrete Fourier transform on 1152 samples recorded with a sampling rate $F_s = 44100$, we will end up with 1152 frequency bins, each containing the amplitude of a frequency range $\frac{F_s}{N} = \frac{44100}{1152} \approx 38$ Hz. However, due to the *Nyquist limit*, the only useful frequencies are up until $\frac{F_s}{2} = \frac{44100}{2} = 22050$ Hz, and so we only need to be concerned with the first half of the bins, i.e. $\frac{1152}{2} = 576$ bins, as the latter half mirrors the first and does not contain any useful information.

Due to the nature of this process, if we run the Fourier transform on our whole signal, we will only be able to analyse it as a whole. That means we won't be able to tell which parts of for example a song are quiet or if there are any parts with very high frequencies - we lose our temporal data.

To alleviate this problem, we can run Fourier transform on smaller chunks of the signal, analyse them separately and later join them back into the original signal. That is the essence of the Short-time Fourier transform.

1.3.2.2 Short-time Fourier transform

When using Short-time Fourier transform, or STFT for short, we first split the signal into smaller segments of equal size and then run Fourier transform on those separately. As such, our output can be projected into two dimensions - specifically a frequency spectrum as a function of time, a spectrogram.

Doing it this way will let us see how the frequency components change over time instead of taking the spectrum of the entire signal.

As with regular Fourier transform, we'll need to window each segment of the signal, but there is a caveat. Since we have windowed segments, we may be losing some information at the edge of each segment leading to artifacts, and furthermore we may be losing information about transients. To solve this, we'll need to introduce overlapping windows - however, having an overlap will increase the amount of coefficients required.

The continuous version is defined as: [**?**]

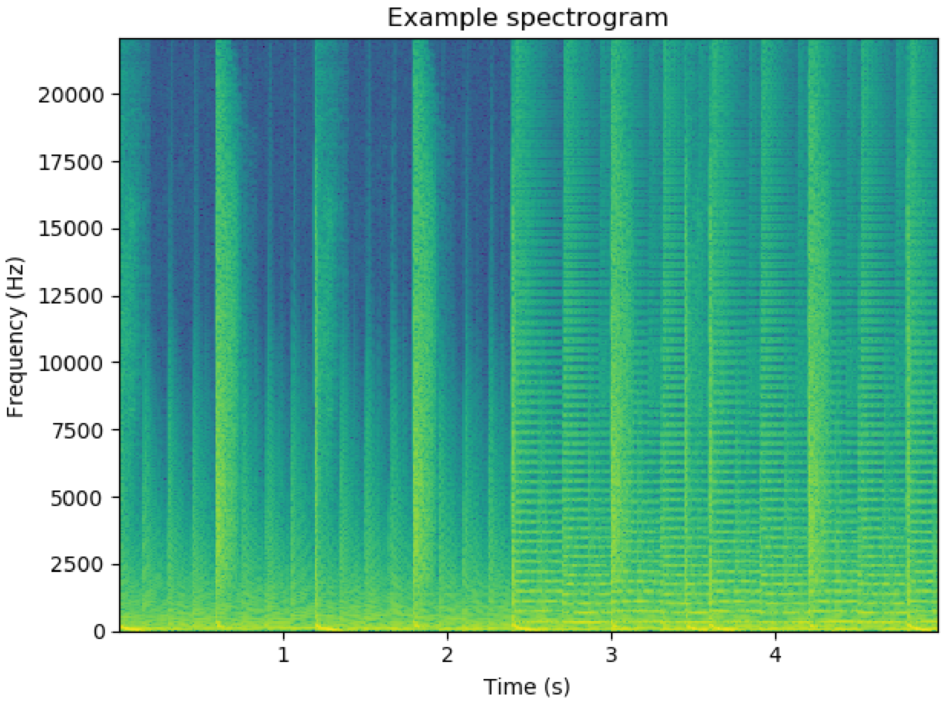
$$S(\tau, \xi) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-2\pi i t\xi}dt \tag{1.10}$$

where w is the window function.

But again, as we have discrete samples, we will need to use a discrete short-time Fourier transform, specifically:

$$S_{k,\xi} = \sum_{n=-\infty}^{\infty} x_n w_{n-k} e^{-2\pi i \xi n} \tag{1.11}$$

Figure 1.4: An example of an audio spectrogram for the signal from Figure 1.



And similarly to the regular Fourier Transform, short-time Fourier Transform is also invertible. [9]

STFT is commonly used for audio analysis (e.g. for generating spectrograms) but in this case it will be used as a means for our NMF compression.

1.3.2.3 Modified discrete cosine transform

Modified discrete cosine transform, or MDCT for short, has become the dominant means of lossy high-quality audio coding. [10]

It is what's known as a *lapped transform*. This means that when transforming a block into its MDCT coefficients, the basis function overlaps the block's boundaries. [11] In practice, what this means is that while we have blocks with overlapping windows as in the short-time Fourier transform, the number of coefficients remains the same as without while retaining the relevant properties.

As the name suggests, MDCT is based on the Discrete cosine transform, namely *DCT-IV*, where the main difference is the addition of lapping mentioned above.

What makes MDCT simpler to work with compared to Fourier transform is that not only do we not need more coefficients despite overlapping, they are also real numbers as opposed to complex numbers, lowering the amount of bytes necessary to store them.

It is a linear function $f : \mathbf{R}^{2N} \to \mathbf{R}^{N}$, defined as: [12]

$$M_k = \sum_{n=0}^{N-1} x_n \cos\left\{\frac{(2n+1+\frac{N}{2})(2k+1)\pi}{2N}\right\} \tag{1.12}$$

for $k = 0, 1, \ldots, \frac{N}{2} - 1$.

It is assumed that $x(n)$ is already windowed by an appropriate windowing function w.

MDCT is also invertible, and its inversion is defined as:

$$\bar{x}_n = \sum_{k=0}^{\frac{N}{2}-1} M_k \cos\left\{\frac{(2n+1+\frac{N}{2})(2k+1)\pi}{2N}\right\} \tag{1.13}$$

for $n = 0, 1, \ldots, N - 1$.

It's important to note that the inverted transformed sequence $\bar{x}_n$ by itself does not correspond to the original signal x_n [13]. To achieve perfect invertibility, we must add subsequent overlapping blocks of the inverted MDCT (IMDCT). This method is called *time domain aliasing cancellation* [14], or TDAC for short. As the name suggests, it mainly helps remove artifacts on the boundaries between transform blocks.

1.4 Psychoacoustics

Apart from time-frequency representations being generally more compact, they also give us the ability to analyse, isolate or modify the frequency composition of a given signal. This comprises a large chunk of the audio compressing process.

The field of psychoacoustics studies sound perception - that is, how our ears work and how we perceive different kinds of sounds. There are many different characteristics to sound that need to be taken into account for a proper psychoacoustic analysis [15], split into several categories, namely:

tonal includes pitch, timbre, melody harmony

dynamic based on loudness

temporal involves time, duration, tempo and rhythm

qualitative represents harmonic constitution of the tone

For music, it's important to balance these four qualities appropriately. For compression, the most important qualities for us in scope of this work are going to be tonal (pitch) and dynamic (loudness).

1.4.1 Pitch

Pitch is a characteristic that comes from a frequency. The difference between the two is that pitch is our subjective perception of the tone whereas a frequency is an objective measure. Despite this fact, pitch is often quantified as a frequency using Hertz as its unit.

The lower bound of human hearing is around 20 Hz whereas the upper bound is most commonly cited as 20 000 Hz, or 20 kHz. [16] In a laboratory environment, people have been found to hear as low as 12 Hz. As people age, our hearing gets progressively worse and a healthy adult younger than 40 years can generally perceive frequencies only up to 15 kHz. [15]

The human ear is capable of distinguishing different frequencies fairly accurately, though accuracy gets lower with increasing frequency. It's easier for our ears to tell a difference between 500 Hz and 520 Hz compared to the difference between 5000 Hz and 5020 Hz. [17]

Furthermore, if we hear two different tones simultaneously, but their frequencies are close enough to one another, we may perceive them as a combination of tones rather than separate tones. Frequency ranges, or bands, where this phenomenon happens, are called *critical bands*. [18] It's also possible for one tone to mask the other entirely, and then we get what's called *auditory masking*. [19]

Based on the knowledge of the existence of these critical bands, it's possible to devise a system that specifies the range of each band in human hearing. One such scale that is commonly used is called the *Bark scale*.

1.4.1.1 Bark scale

The Bark scale ranges from 1 to 24 Barks, where each Bark corresponds to a single critical band of human hearing. [20] The perceived difference in pitch between each band should be the same, despite the scale not growing linearly in terms of frequency ranges. Specifically, until around 500 Hz, the scale is roughly linear, but above that it has a more logarithmic growth. [21]

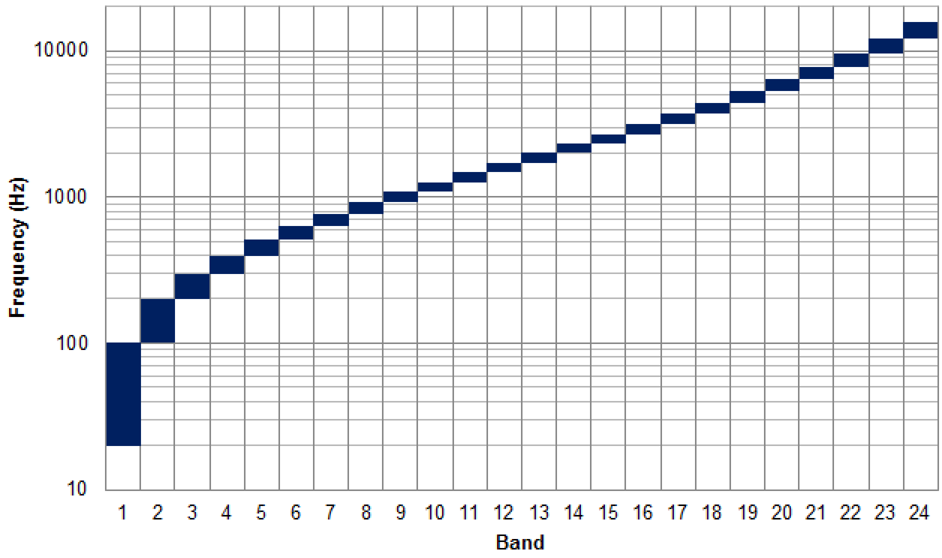
The Bark scale is commonly used as reference for audio encoding codecs, as we will see later. Knowledge of these critical bands allows for more educated byte allocation during the quantization process when compressing a frequency domain representation.

1.4.2 Loudness

What people often decide as loudness is really called *sound pressure level* and it's measured in decibels (dB), however it has some shortcomings when it comes to psychoacoustic analysis.

It is defined as following: [22]

Figure 1.5: The Bark scale. Image by "Swpb" licensed under CC BY-SA 4.0.



$$L_p = 20 \log_{10}\left(\frac{p}{p_0}\right) \mathrm{dB} \tag{1.14}$$

where p is a sound's sound pressure and p_0 is a reference sound pressure, also called the threshold of human hearing.

While this metric is very popular, it doesn't account for the fact that different frequencies have a different perceived loudness for a person's ears. [15] There is a lot of research in recent years into how different frequencies impact our perception and hearing [23], but that is out of scope of this work. For more information about the exact definitions of loudness, refer to [15].

1.4.3 Auditory masking

As mentioned above, when it comes to audio masking, and therefore audio compression, we must not only take into account the critical bands as per e.g. the Bark scale, but also their intensity.

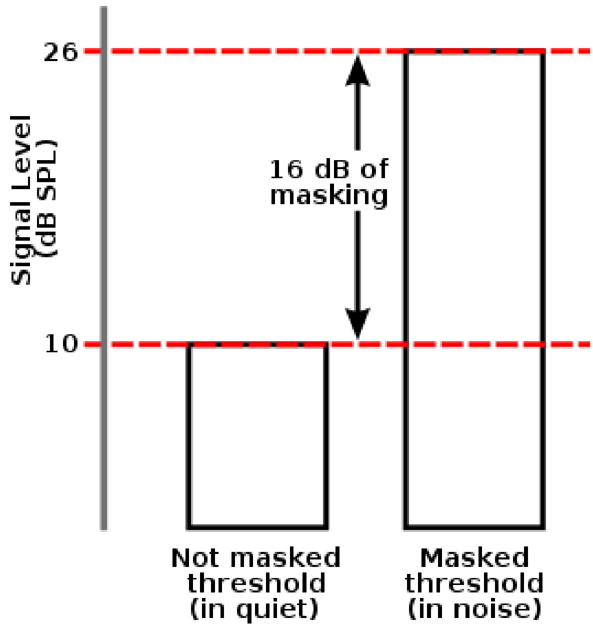
For example a lower frequency sound may mask one of a higher frequency, but the other way around does not apply. [19] Modern audio encoders take this into account and using this knowledge are able to eliminate sounds that exist in the original signal, but are not perceivable by humans.

There are two important different kinds of masking effects - *simultaneous* masking and *temporal* masking. [4]

Simultaneous masking is what I have hinted at above - when there are two sounds within the same critical band, the dominant one may mask other frequencies within the same band. This can be compensated to a degree by increasing the volume of the masked sound.

Temporal masking does not occur in the frequency domain, but the time domain. The essence is that a stronger tonal component may mask a weaker one if they appear within a small window of time in succession.

Figure 1.6: A hypothetical example of how a masker can shift the hearing threshold of a signal by 16 dB. Image created by Alan De Smet and published in the public domain.



Chapter **2**

Non-negative matrix factorization

In today's age of big data, machine learning and various other fields, it's important to have ways to quickly analyse these datasets and ideally find patterns within. Non-negative matrix factorization is one of the paradigms suitable for that task.

In layman's terms, what NMF does is that when we are given a large set of data, for example a matrix representing books and their review scores from people, we can extract certain hidden "features" from it using NMF, in this case representing e.g. various genres (basis matrix) and how prominent they are in a given book (weight matrix). And then, using these two matrices, we are able to estimate or even predict what kind of books a user would like - this is a simple example of a possible recommendation algorithm.

2.1 Linear dimensionality reduction

Non-negative matrix factorization, or NMF, falls under *linear dimensionality reduction* techniques. These are used widely for noise filtering, feature selection or compression, among others.

LDR can be defined as following: [24]

$$x_j \approx \sum_{k=1}^{r} w_k h_j(k) \qquad \text{for some weights } h_j \in \mathbf{R}^r \tag{2.1}$$

where given a data set of size n, we define $x_j \in \mathbf{R}^p$ for $1 \leq j \leq n$, $r < \min(p, n)$, and $w_k \in \mathbf{R}^p$ for $1 \leq k \leq r$.

What this effectively means is that we represent p-dimensional data points in a r-dimensional linear subspace, with basis elements w_k and data coordinates given by vectors h_j. LDR defined in this manner is equivalent to low-rank matrix approximation, which is the essence of non-negative matrix factorization.

2.2 NMF definition

Non-negative matrix factorization solves the following NP-hard problem:

Given a non-negative matrix V, find non-negative matrix factors W and H such that:

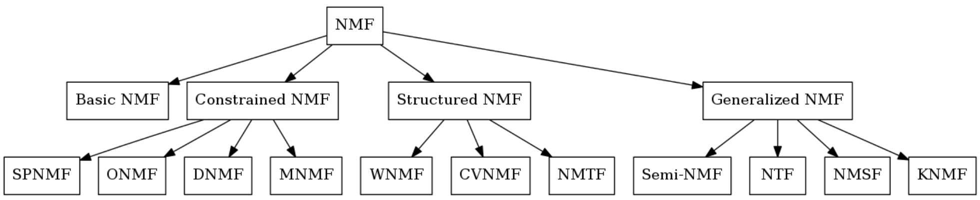
$$V \approx WH \tag{2.2}$$

That is, given a set of multivariate n-dimensional data vectors, we place these vectors in the columns of a $n \times m$ matrix V, where m is the amount of examples we have. We then approximately factorize this matrix into two different matrices: a $n \times r$ matrix W and a $r \times m$ matrix H. We generally choose $r < \min(n, m)$ (though this is not required) so that the two matrices are smaller than the original matrix V, essentially compressing it. [25]

2.3 Classification

NMF is as of currently still a relevant research topic, and has been explored by researchers from many different fields including mathematicians, statisticians, computer scientists or biologists. Given the wide range of use, over time it lead to different variations and additional constraints on the algorithms. Therefore, a taxonomy system was proposed in [26], outlined below.

Figure 2.1: The NMF classification as per [26].



2.3.1 Basic NMF

This is the basic model which only enforces non-negativity, and which all the following ones build upon.

However, due to its unconstrained nature, without any other constraints there are many possible solutions which may lead to the algorithm's performance to vary. Further constraints outlined below help in the search of unique solutions and optimizing for specific scenarios.

2.3.2 Constrained NMF (CNMF)

Constrained NMF imposes additional constraints on the resulting matrices, namely:

Sparse NMF SPNMF, sparseness constraint

Orthogonal NMF ONMF, orthogonality constraint

Discriminant NMF DNMF, couples discriminant information along with the decomposition

NMF on manifold MNMF, preserves local topological properties

2.3.3 Structured NMF (SNMF)

Structured NMF modifies standard factorization formulations:

Weighed NMF WNMF, attaches weights to different elements relative to their importance

Convolutive NMF CVNMF, considers time-frequency domain factorization

Non-negative Matrix Trifactorization NMTF, decomposes the data into three matrices

2.3.4 Generalized NMF (GNMF)

Generalized NMF can be considered a broader variant of Basic NMF, where conventional data types or factorization modes may be replaced with something different. It's split as follows:

Semi-NMF relaxes the non-negativity constraint on a specific factor matrix

Non-negative Tensor Factorization NTF, generalizes the model to higher dimensional tensors

Non-negative Matrix-set Factorization NMSF, extends the data sets from matrices to matrix-sets

Kernel NMF KNMF, non-linear model of NMF

2.4 Properties

The additional constraint of non-negativity is important, as results show that it leads to a natural higher sparseness in both the basis matrix (W) and the encoding matrix (H). Additionally, non-negativity leads to a parts-based representation, which is similar to how our brains are presumed to work, basically combining parts in an additive manner to form a whole instead of subtracting. [27] This sparseness makes it even easier to further compress the resulting matrices, saving us more space.

However this isn't without any downsides. While the concept of adding parts together seems to make a lot of sense, there is an issue. Since NMF employs a holistic approach, the additive parts learned by it in an unsupervised mode only considers features on a global level, and does not allow for representation of spatially localized features. [28]

So while on paper NMF might seem better than PCA or SVD for a parts-based representation, it only comes at a cost of increased complexity, and since both PCA and SVD have a more compact spectrum than NMF, we must consider if this is worth the trade-off. [26]

2.5 Algorithms

For the purposes of this thesis, we will only consider algorithms for Basic NMF. While NMF has been widely used in sound analysis etc. as we'll see below, its use for audio compression specifically is rare and there are limited resources to provide insight into utilizing possible constraints, therefore we will only be using the standard version.

Finding a decomposition of a matrix V into matrices W and H is an NP-hard problem, and as such, the resulting matrices are generally only approximated over a number of iterations of an optimization algorithm. What this means in practice is that it's likely a result we'll find is sub-optimal or a local minimum.

2.5.1 Cost function

When using iterative updates, in each step of the process we need to evaluate the quality of the approximation. The function that does this is called the *cost function*, or *objective function*.

There are two simple commonly used functions. Firstly, we can use squared Euclidean distance: [29]

$$||A - B||^2 = \sum_{ij}(A_{ij} - B_{ij})^2 \tag{2.3}$$

This is lower bounded by 0, which it only is equal to if $A = B$.

Another metric we can use is based on Kullback-Leibler divergence, and is defined as such: [25]

$$D(A||B) = \sum_{ij} \left(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right) \tag{2.4}$$

2.5.2 Update rules

With the cost function in place, we now need a function to apply each iteration to try and minimize the value of the cost function. It has been found that a good compromise between speed and ease of implementation is to use what's called *multiplicative update rules*. [25] Despite being over 15 years old, they are still very commonly used exactly for this reason.

For non-increasing Euclidean distance $||V - WH||$, if W and H are at a stationary point of distance, we may use these rules:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{(W^T V)_{a\mu}}{(W^T W H)_{a\mu}} \tag{2.5}$$

$$W_{ia} \leftarrow W_{ia} \frac{(V H^T)_{ia}}{(W H H^T)_{ia}} \tag{2.6}$$

And for non-increasing divergence $D(V||WH)$, if W and H are at a stationary point of divergence, we can use this:

$$H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i W_{ia} V_{i\mu} / (WH)_{i\mu}}{\sum_k W_{ka}} \tag{2.7}$$

$$W_{ia} \leftarrow W_{ia} \frac{\sum_\mu H_{a\mu} V_{i\mu} / (WH)_{i\mu}}{\sum_v H_{av}} \tag{2.8}$$

2.5.3 Initialization

Before we can use our update rules to iteratively optimize the cost function, we need some initial value to initialize the matrices W and H to, first. In practice, different initializations generally yield different solutions, so this is worth considering. [30]

The most basic way of initialization is to simply initialize the matrices with random values. This generally works decently but you somewhat lose out on controlling the composition of the matrices. A small way to improve this is to generate random matrices a couple times and pick the one with the lowest cost function value, but the issue remains.

As per [30], there are tons of different ways to initialize the matrices, usually relating to constraints on the matrices, e.g. initialize H in such a way that none of the values are above a certain threshold etc. But given the low amount of research on audio compression using NMF, it's difficult to gauge which of these might work better for audio than others, and as such this work will not elaborate on different ones further.

2.6 Use in digital audio processing

In digital audio, Non-negative matrix factorization is mostly used as a tool for analysis rather than compression. The ability to extract hidden features from a given signal is useful for certain things.

For example NMF sees use in audio separation tasks. [31] The gist of this lies in first creating a spectrogram of the signal using STFT and then using NMF decomposition to isolate different kinds of sounds or instruments from it, roughly represented by the basis matrix.

Another example of use is musical transcription. [7] The idea here is to process an input signal via STFT, further filtering and NMF, to isolate individual notes from e.g. a piano piece.

Both of these methods have seen some success, but as we will see later, applying NMF to compression rather than analysis is not a simple task and might not even be worth it.

Chapter 3

Lossy audio compression

Compression can be split into two kinds - lossy and lossless. Using ”lossless” in the context of audio is a bit misleading, since sampling itself is a lossy process, but using a high enough sampling rate, we will not notice any difference, so sampled audio without any lossy compression will be our baseline.

For audio, lossless compression generally means taking some form of digital audio representation and losslessly compressing this data. This will preserve the signal in its entirety with a reduced bit-rate. However, due to size of such audio (an audio CD could only fit about 80 minutes of such music sampled at 44.1 kHz), it’s become more common to use a lossy format.

Lossy compression implies that there will be loss of data, and while this is true, thanks to the application of various psychoacoustic principles size of audio can be greatly reduced without altering human perception, leading to vastly smaller bit-rates for no real cost.

This work focuses on lossy audio compression, therefore only lossy codecs will be considered for comparison.

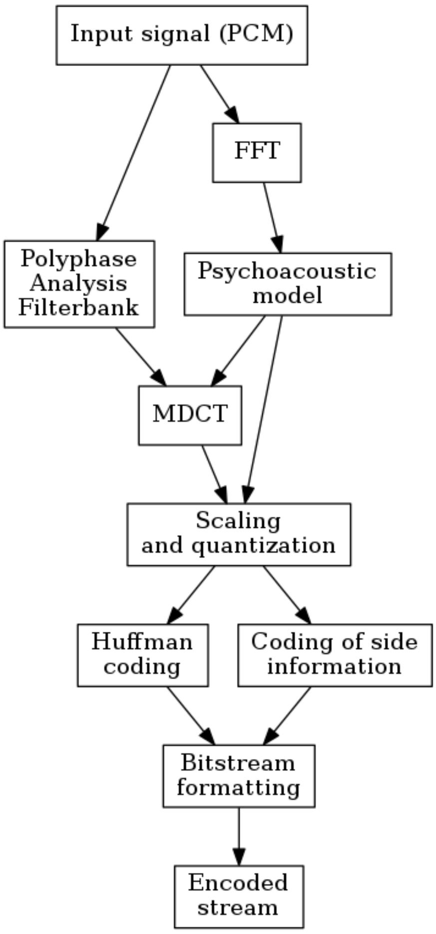
3.1 Overview

3.2 State of the art

Due to its qualities of efficiently compacting energy and mitigating artifacts at block boundaries, MDCT is the most commonly used transformation in modern lossy audio coding, and is employed in the most popular audio formats including MP3, Opus, Vorbis or AAC.

In this section, I will elaborate on some of the more popular ones to get an idea of what considerations go into writing a modern audio codec. They are also the ones that will be used for comparison with my own encoding scheme.

Figure 3.1: The MP3 encoding scheme, simplified.



3.2.1 MP3

MP3, or MPEG-1 Layer III has been standardized in 1991 and has since become widespread throughout a multitude of electronic devices as the de-facto standard for music storage. Its simplified encoding scheme can be seen in Figure 3.1.

It's a very powerful compression/decompression scheme capable of reducing the bit-rate of an audio stream by up to a factor of 12 without any noticeable (to humans) quality degradation. In other words, to transmit CD quality audio, it needs a bitrate of 128 kbps. [4]

The core of MP3 compression is the Modified discrete cosine transform. The signal represented in its PCM form is first split into 32 subbands using an analysis polyphase filterbank, and each of those is further split into 18 MDCT bins, so overall we end up with 576 MDCT frequency bins per frame.

These bins are then sorted into 22 scalefactor bands, which roughly correlate to the 24 bands of human hearing. The point of these bands is that you may individually scale each of them up or down depending on how much precision you need for that specific frequency range. This usually done by dividing and rounding the values in the band, losing a certain amount of information; this process will be reversed during decoding.

The signal is also analyzed using the Fourier transformation, which gives us frequency information for the signal in the same frame, and we can use this information to determine how much to scale each scalefactor band - e.g. if there's some weak sound that will be masked by another, we can assign the band it's in lower precision, saving data. [32]

Once we have the scaled and quantized data, we use the Huffman encoding to losslessly compress these values, and format this output into the final bitstream, encoding our audio.

3.2.2 Opus (CELT)

The Opus codec has been standardized fairly recently, in 2012 [33] compared to other widespread audio codecs. Opus was created from two core technologies - Skype's SILK codec based on linear prediction, and Xiph.Org's CELT codec, based on the MDCT. [34]

As a result, Opus is capable of performing in three different modes:

SILK used for speech signals

CELT used for high-bitrate speech and music

Hybrid both SILK and CELT used simultaneously

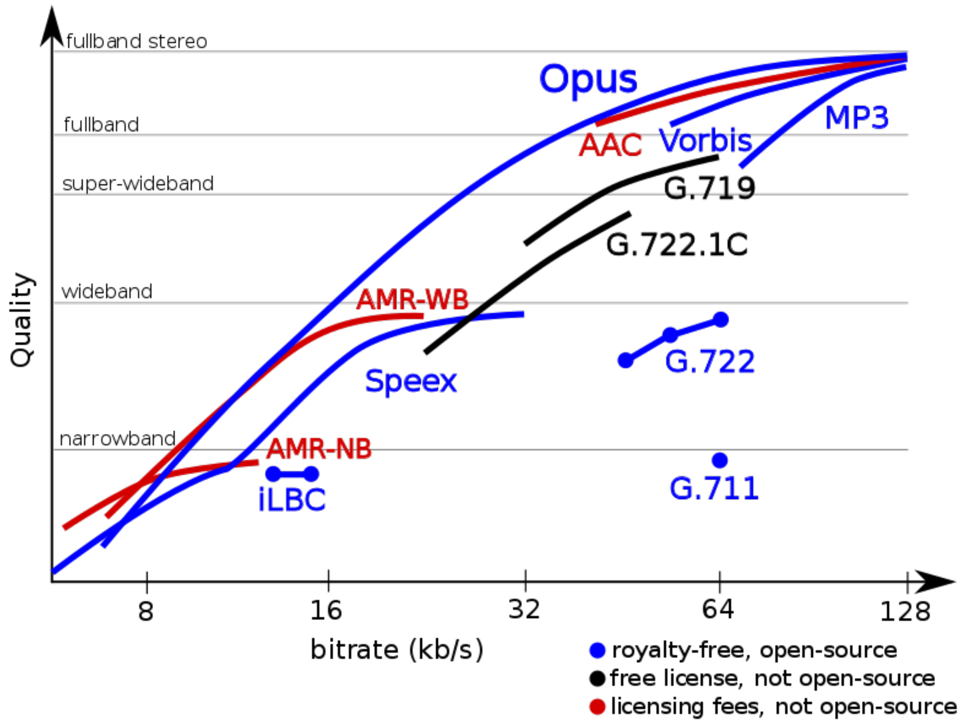
This thesis will mainly focus on Opus CELT due to it being more general purpose than SILK, which will make it easier to use for comparison with a NMF-based codec. SILK uses a technique known as Linear predictive coding, whose complexity and difficulty of implementation exceeds the scope of this work.

Opus is designed with real-time constraints in mind, that is, for example network music performances which require very low delays. Despite that, however, its compression is comparable to codecs with higher delays intended for streaming or storage. This makes it a good candidate to test against in this work, alongside MP3.

CELT (Constrained Energy Lapped Transform) is based on MDCT similar to MP3, but the main difference is that Opus uses specifically sized bands with ranges similar to the Bark scale in order to preserve the spectral envelope of the signal.

Similar to MP3, Opus CELT works on the basis of quantizing MDCT coefficients, but utilizes various kinds of prediction and focuses more on handling transients, and is much more complex in general. Through various optimizations Opus achieves similar results but at a reduced bitrate.

Figure 3.2: This figure attempts to summarize the results of a collection of listening tests, comparing various lossy audio codecs. Image attributed to Jean-Marc Valin and licensed under CC BY 3.0.



3.3 Compression using NMF

This section is separate from the state of the art, as NMF is not really used for audio compression in practice. There is one notable example that will be covered and to an extent implemented here, for more details refer to [35].

The proposed method uses a metric previously formulated in [36] called *noise-to-mask ratio*, or NMR for short. They successfully apply this metric onto non-negative matrix factorization, establishing a custom cost function to represent NMR and also propose update rules based on Euclidean distance for minimizing this cost function.

Results showed that NMR performed better than other, more general cost functions. Audio compressed using NMF + NMR had better perceptual quality and managed to encode signals with lots of transient sounds in better quality. However, NMR relies on proprietary algorithms such as PEAQ [37] and an implementation is not readily available.

This modified NMF is then used in what the paper refers to as *object-based audio coding*, and the principle is fairly simple to understand. Instead of using MDCT for conversion to the frequency domain, they use STFT, obtaining the frequency spectrum as a set of complex values. They then take the magnitude and phase of each of these complex values to obtain the *magnitude spectrogram* and *phase information*.

The magnitude spectrogram is then compressed using NMF (+NMR) among other things, whereas the phase information is entropy encoded separately and does not utilize NMF at all.

The conclusion is that while object-based audio coding done this way achieves reasonable bit-rate and quality on the magnitude spectrogram, it is difficult to efficiently encode the phase information and as such the bit-rate of this approach proved to be too high to be usable in practice compared to other, more common, MDCT based methods.

Part II

Audio compression using NMF

Chapter 4

Design

.. time domain compression ..

compressing raw

.. frequency domain compression ..

- compressing STFT

STFT

NMF

mu-law quantization, non-uniform, formula in p128 wrong (missing sgn)

- compressing MDCT

unusable, needs to be compressed consistently / losslessly

.. file structure ..

diagram for both time domain and frequency domain compression

Chapter 5

Implementation

5.1 Encoder

.. process of encoding ..
.. application of NMF ..
.. variables ..

5.2 Decoder

Chapter 6

Evaluation

.. how is audio evaluated ..

.. gstpeaq ..

.. how does gstpeaq work ..

.. how and what did I test ..

.. comparison to other formats ..

Conclusion

.. what went right ..

.. what went wrong ..

.. what could be improved .. add psychoacoustics try compressing LPC/SILK

Bibliography

[1] You, Y. *Audio Coding Theory and Applications*. Springer US, 2010.

[2] Bosi, M.; Goldberg, R. E. *Introduction to digital audio coding and standards*. Kluwer Academic Publishers, 2003.

[3] Marks Robert J., I. *Introduction to Shannon Sampling and Interpolation Theory*. 01 1991, ISBN 0387973915, doi:10.1007/978-1-4613-9708-3.

[4] Raissi, R. The Theory Behind Mp3. 2002.

[5] Malvar, H. S. Lapped transforms for efficient transform/subband coding. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, volume 38, no. 6, June 1990: pp. 969–978, ISSN 0096-3518, doi:10.1109/29.56057.

[6] Shatkay, H. The Fourier Transform - A Primer. Technical report, Providence, RI, USA, 1995.

[7] Recoskie, D.; Mann, R. Constrained Nonnegative Matrix Factorization with Applications to Music Transcription. *University of Waterloo Technical Report CS-2014-27*, Dec 2014. Available from: `https://cs.uwaterloo.ca/sites/ca.computer-science/files/uploads/files/cs-2014-27.pdf`

[8] Heinzel, G.; Rüdiger, A.; et al. Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new flat-top windows. *Max Plank Inst*, volume 12, 01 2002.

[9] Selesnick, I. W. Short-Time Fourier Transform and Its Inverse. Apr 2009. Available from: `http://eeweb.poly.edu/iselesni/EL713/STFT/stft_inverse.pdf`

[10] Wang, Y.; Vilermo, M. Modified discrete cosine transform - Its implications for audio coding and error concealment. *Advances in Engineering Software - AES*, volume 51, 01 2012.

[11] Malvar, H. S. *Signal Processing with Lapped Transforms*. Norwood, MA, USA: Artech House, Inc., 1992, ISBN 0890064679.

[12] Babu, S. P. K.; Subramanian, K. Fast and Efficient Computation of MDCT / IMDCT Algorithms for MP 3 Applications. 2013.

[13] Princen, J.; BRADLEY, A. Analysis/Synthesis filter bank design based on time domain aliasing cancellation. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, volume 34, 11 1986: pp. 1153 – 1161, doi:10.1109/TASSP.1986.1164954.

[14] Princen, J.; Johnson, A.; et al. Subband/Transform coding using filter bank designs based on time domain aliasing cancellation. 05 1987, pp. 2161 – 2164, doi:10.1109/ICASSP.1987.1169405.

[15] Olson, H. *Music, Physics and Engineering*. Dover Books, Dover Publications, 1967, ISBN 9780486217697. Available from: `https://books.google.cz/books?id=RUDTFBbb7jAC`

[16] Rosen, S.; Howell, P. Signals and Systems for Speech and Hearing. *Acoustical Society of America Journal*, volume 94, 12 1993: p. 163, doi:10.1121/1.407176.

[17] ”smacdon”. Critical Bands in Human Hearing. Oct 2018. Available from: `https://community.plm.automation.siemens.com/t5/Testing-Knowledge-Base/Critical-Bands-in-Human-Hearing/ta-p/416798`

[18] Fletcher, H. Auditory Patterns. *Rev. Mod. Phys.*, volume 12, Jan 1940: pp. 47–65, doi:10.1103/RevModPhys.12.47. Available from: `https://link.aps.org/doi/10.1103/RevModPhys.12.47`

[19] Gelfand, S. *Hearing: An Introduction to Psychological and Physiological Acoustics*. 01 1990, 187 pp., doi:10.1201/b14858.

[20] Fastl, H.; Zwicker, E. *Psychoacoustics: Facts and Models*. Berlin, Heidelberg: Springer-Verlag, 2006, ISBN 3540231595.

[21] Hermes, D. J. The auditory filter. Available from: `http://home.ieis.tue.nl/dhermes/lectures/soundperception/04AuditoryFilter.html`

[22] Behar, A. Intensity and sound pressure level. *Journal of The Acoustical Society of America - J ACOUST SOC AMER*, volume 76, 08 1984, doi:10.1121/1.391117.

[23] Kuwano, S.; Namba, S.; et al. Advantages and Disadvantages of A-weighted Sound Pressure Level in Relation to Subjective Impression of Environmental Noise. *Noise Control Engineering Journal - NOISE CONTR ENG J*, volume 33, 11 1989, doi:10.3397/1.2827748.

[24] Gillis, N. The Why and How of Nonnegative Matrix Factorization. *Regularization, Optimization, Kernels, and Support Vector Machines*, volume 12, 01 2014.

[25] Lee, D.; Seung, H. Algorithms for Non-negative Matrix Factorization. *Adv. Neural Inform. Process. Syst.*, volume 13, 02 2001.

[26] Wang, Y.-X.; Zhang, Y.-J. Nonnegative Matrix Factorization: A Comprehensive Review. *IEEE Transactions on Knowledge and Data Engineering*, volume 25, no. 6, 2013: p. 1336–1353, doi:10.1109/tkde.2012.51.

[27] Lee, D.; Sebastian Seung, H. Learning the Parts of Objects by Non-Negative Matrix Factorization. *Nature*, volume 401, 11 1999: pp. 788–91, doi:10.1038/44565.

[28] Li, S.; Hou, X.; et al. Learning Spatially Localized, Parts-Based Representation. 01 2001, pp. 207–212, doi:10.1109/CVPR.2001.990477.

[29] Paatero, P. Least Squares Formulation of Robust Non-Negative Factor Analysis. *Chemometrics and Intelligent Laboratory Systems*, volume 37, 05 1997: pp. 23–35, doi:10.1016/S0169-7439(96)00044-5.

[30] Naik, G. *Non-negative Matrix Factorization Techniques: Advances in Theory and Applications*. 09 2015, ISBN 978-3-662-48331-2, doi:10.1007/978-3-662-48331-2.

[31] Févotte, C.; Vincent, E.; et al. *Single-channel audio source separation with NMF: divergences, constraints and algorithms*. 01 2017.

[32] Wilburn, T. The AudioFile: Understanding MP3 compression. Oct 2007. Available from: `https://arstechnica.com/features/2007/10/the-audiofile-understanding-mp3-compression/2/`

[33] Valin, e. a. Definition of the Opus Audio Codec. RFC 6716, RFC Editor, September 2012. Available from: `http://www.rfc-editor.org/rfc/rfc6716.txt`

[34] Valin, J.-M.; Maxwell, G.; et al. High-Quality, Low-Delay Music Coding in the Opus Codec. *135th Audio Engineering Society Convention 2013*, 01 2013: pp. 73–82.

[35] Nikunen, J.; Virtanen, T. Object-based audio coding using non-negative matrix factorization for the spectrogram representation. 05 2010.

[36] Nikunen, J.; Virtanen, T. Noise-to-mask ratio minimization by weighted non-negative matrix factorization. *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010: pp. 25–28.

[37] BS, ITU-R. Method for objective measurements of perceived audio quality. 01 2006.

Appendix A

Acronyms

todo TODO

Appendix **B**

Contents of enclosed CD

```
│ readme.txt ....................... the file with CD contents description
├── exe ....................................... the directory with executables
├── src ......................................the directory of source codes
│   ├── wbdcm ...................................... implementation sources
│   └── thesis ..............the directory of LaTeX source codes of the thesis
└── text ......................................... the thesis text directory
    ├── thesis.pdf ..........................the thesis text in PDF format
    └── thesis.ps ............................the thesis text in PS format
```