Concatenated convolution framelets in audio compression

- Master Thesis -

Hansen, Thomas Rune
Maniragaba, Hoza Benjamin
Pedersen, Mathias Bach

Aalborg University
Mathematical Engineering

AALBORG UNIVERSITY
STUDENT REPORT

Department of Electronics and IT
& Department of Mathematical Sciences
Aalborg University
http://www.aau.dk

Title:
Concatenated convolution framelets in audio compression

Theme:
Master's Thesis

Project Period:
Spring semester 2018

Project Group:
Group 5.218

Participants:
Hansen, Thomas Rune
Maniragaba, Hoza Benjamin
Pedersen, Mathias Bach

Supervisors:
Østergaard, Jan
Morten, Nielsen
Jensen, Jesper

Copies: 4

Page Numbers: 70

Date of Completion:
June 6, 2018

Synopsis:

This project concerns convolution framelets, [Yin et al., 2017], a recently proposed class of redundant tight frames. Specifically, the use of convolution framelets in audio compression is explored. In order to adapt convolution framelets to the problems of compression, we propose methods for controlling the degree of redundancy. Rather than using a single convolution framelet with a high degree of redundancy, a concatenation of less redundant convolution framelets is used. This allows for the use of multiple patch sizes, at the cost of having fewer patches of each size. The compression scheme in this project is inspired by an audio coding scheme proposed by [Ravelli et al., 2008]. The compression scheme consists of finding a sparse set of frame coefficients, and coding these coefficients. To find sparse coefficients, the orthogonal matching pursuit algorithm, [Foucart and Rauhut, 2013], is used. The coding algorithm is inspired by that in [Ravelli et al., 2008], which relies on bitplane run-length coding, and uses an interleaving scheme. The compression scheme is tested on a set of music excerpts, and the resulting quality, estimated by perceptual evaluation of audio quality, are compared to an MP3 coder and the non-psychoacoustic results in [Ravelli et al., 2008]. This comparison shows that our compression scheme does not perform as well for low bit rates.

Preface

This Master's thesis report is written in the period from 01-02-2018 to 07-06-2018 at the 10th semester of Mathematical Engineering at Aalborg University. The Master's thesis was written in cooperation with the Department of Electronic Systems and the Department of Mathematical Sciences, hence two of the project supervisors are connected to each of the two departments, respectively. The third supervisor is from Oticon A/S and works in collaboration with the Department of Electronics Systems.

The topic of interest in this Master's thesis is *concatenated convolution framelets in audio compression.*

The authors would like to thank Jan Østergaard, Morten Nielsen and Jesper Jensen for supervising this Master's thesis. Furthermore, the authors would like to thank Thomas Arildsen at the Department of Electronic Systems for providing access to the servers used for running the simulations carried out in this work. Python implementation of the used algorithms in this Master's thesis is found in the file "*python_scripts.zip*".

Aalborg University, June 6, 2018

Hansen, Thomas Rune
<thanse10@student.aau.dk>

Maniragaba, Hoza Benjamin
<hmanir12@student.aau.dk>

Pedersen, Mathias Bach
<mapede13@student.aau.dk>

Danish abstract

Dette projekt omhandler convolution framelets, som er en klasse af tight frames præsenteret for nylig af [Yin et al., 2017]. I denne forbindelse, er anvendelsen af convolution framelets til lydkomprimering blevet undersøgt.

Vi foreslår metoder til at kontrollere niveauet af redundans for at tilpasse convolution framelets til komprimering. Ideen er at substituere den enkle convolution framelet, der har et højt niveau af redundans, med en sammenkædning af mindre redundante convolution framelets. Dette giver mulighed for at anvende patches med forskellige længder med den konsekvens at der bliver færre patches af hver længde. Komprimeringsmetodikken i dette projekt er inspireret af en lydkomprimeringsmetode forslået af [Ravelli et al., 2008]. Komprimeringsmetoden består i at finde en tyndt besat mængde af frame koefficienter og kode disse koefficienter. For at finde tyndt besatte koefficienter, anvendes ortogonal matching pursuit algoritmen i takt med [Foucart and Rauhut, 2013]. Kodningsalgoritmen er inspireret af algoritmen fra [Ravelli et al., 2008], som er baseret på run length kodning og en interleaving metodik.

Komprimeringsmetodikken er testet på forskellige udklip af musik. Kvaliteten af komprimering er evalueret ved hjælp af et perceptuelt mål af lydkvalitet, kaldet for Perceptual Evaluation of Audio Quality (PEAQ). Vores komprimeringsmetodik sammenlignes med metodikken fra [Ravelli et al., 2008] og en MP3 koder ved at bruge PEAQ værdier. Sammenligning viser at vores komprimeringsmetodik ikke præsterer lige så godt for lavere bitrater. På trods af resultaterne, har sammenkædede convolution framelets potentialet som signalrepræsentation givet deres fleksibilitet med hensyn til domæne og redundansniveau.

Contents

1 | Introduction

Sound can be described as propagating oscillations in air pressure. When recording sound digitally, the air pressure is measured and stored as a time series. In this representation, sound can be thought of as a function of time, hence it is called the time domain representation. The time domain is practical for recording and playback, but it makes little sense to humans, and it turns out to be inefficient for digital storage [Bosi and Goldberg, 2002, ch. 1.1], [You, 2010, ch. 10.1]. The human inner ear decomposes sound into its frequency components, which allows us to hear sound as a spectrum of tones. The way the human ear interprets sound can be thought of as a function of frequency, at least over short spans of time [Bosi and Goldberg, 2002, ch. 6.9]. This is the time-frequency domain representation of sound. It turns out that real sound is often efficiently represented in the time-frequency domain [You, 2010, ch. 5.4]. This observation forms the basis of many audio compression techniques.

To introduce audio compression consider source coding. In source coding, the problem lies in determining the distribution of the source, so that an appropriate entropy coder can be designed. Real audio is not well described as a stationary source, since real sound changes characteristics over time, which means that the source would change distribution over time [You, 2010, ch. 5.4]. In practice a widely used approach is transform coding, in which the sound is transformed into a different domain for coding. Transform coding usually takes advantage of the efficient representation of sound that comes with frequency transforms, such as the Discrete Cosine Transform [DCT]. First of all the advantage is that the transform compacts most of the sounds energy into a few coefficients. Using an appropriate bit allocation strategy, sound can thus be effectively compressed in this domain. Secondly, the transform is typically chosen to be similar to the way the ear decomposes sound, i.e. a time-frequency transform. This means that coefficients, which are imperceptible to the human ear can be neglected in the coding. In particular, the psychoacoustic effect known as auditory masking is used to identify imperceptible coefficients [Bosi and Goldberg, 2002, ch. 6.6].

Transform coding is mainly concerned with orthogonal transforms, since these are energy preserving and easily invertible. Research has also been done on redundant transforms, e.g. [Ravelli et al., 2008], where a union of modified discrete cosine transforms are used. The advantage of using redundant transforms is that the representation of a sound is no longer unique, which allows one to search for representations with better energy compaction, or sparsity. There exist methods for finding sparse representations in redundant transforms. Algorithms from compressive sensing are useful for this purpose, e.g. matching pursuit, [Mallat and Zhang, 1993], [Foucart and Rauhut, 2013, ch. 3]. These methods are computationally costly, which slows down the coding in comparison to orthogonal transforms.

Convolution framelets, introduced by [Yin et al., 2017], are a recent class of redundant tight frames. Frames are redundant generalizations of bases. Tight frames scale all vectors equally [Christensen, 2008, ch. 1.1]. The framelets are constructed by element-wise convolution between two existing tight frames (or orthonormal bases), $\boldsymbol{\Phi}$ and $\boldsymbol{V}$. The canonical coefficients of a given signal in a framelet can be found efficiently. Building a Hankel matrix $\boldsymbol{F}$, the so-called patch matrix, from the time domain signal, the two frames $\boldsymbol{\Phi}$ and $\boldsymbol{V}$ can be applied as transforms to the left and right, respectively, of this patch matrix. This results in a matrix, which contains exactly the canonical coefficients of the convolution framelet.

This project explores the use of convolution framelets for audio compression. The basic idea is to let convolution framelets take the place of orthonormal transforms in a transform coding scheme. Some adaptations are made to the convolutional framelets, to better suit the audio compression context. The structure of the patch matrix causes a high degree of redundancy. The patch matrix is therefore generalized to a subset of rows from the full patch matrix, which allows the redundancy to be reduced. It turns out that since convolution framelets correspond to tight frames, they can be concatenated and preserve tightness, which allows for a unification of differently structured patch matrices for the same signal. As a point of reference, this is comparable to analyzing a signal using multiple window lengths simultaneously.

Once a convolution framelet has been constructed, it is desired to find an efficient representation of a given audio signal in this framelet. Here, 'efficient' should be understood as efficient for coding; practically sparse representations will be sought. The canonical frame coefficients will not do for compression, as they are typically not sparse in concatenated convolution framelets. The approach taken is to choose certain frame elements that describe as much of the signal as possible, resulting essentially in a signal adaptive representation. This is where compressive sensing is of interest, as this field is focused on finding sparse signal representations in redundant systems.

Once sparse coefficients have been found, these have to be encoded into a compressed bit stream. This involves quantization and is thus described as a lossy coding scheme. The convolution framelets in this project will be chosen such that the frame is comparable to a union of bases used by [Ravelli et al., 2008], wherein a redundant system is also used for audio coding. The coding is based on run length coding, since zero-bits are much more common than one-bits in the sparse coefficients. This essentially allows the coder to compress long runs of consecutive zero-bits very effectively, while spending a few extra bits on every one-bit. This coding strategy is implemented in a bit plane coder, which separates the raw coefficient bits into two categories, where the so-called significance bits are the most compressible and the so-called refinement bits are typically not compressible at all.

Finally the method is tested using varying bit rates on a number of music segments and compared to both MP3 compressed versions of the same music segments and the non-psychoacoustic results from [Ravelli et al., 2008].

2 | Framelet Theory

This chapter is dedicated to the theory for the understanding and usage of framelets necessary for this project. Section 2.1 contains the definitions and propositions required for the expansion of framelet theory presented by this project. In Section 2.2 we generalize the convolution framelet to patches with less redundancy along with proof that a reduced (less redundant) framelet can still be tight. In Section 2.3 we present a method for constructing convolution framelets by concatenating multiple orthonormal bases and using the resulting frames for the construction, along with proof of when this method produces tight frames.

2.1 Frames and convolution framelets

In order to define and describe the new methods presented in this report some definitions are required. First circular convolution,

Definition 2.1
The circular convolution between two vectors $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^N$ is defined as

$$(\boldsymbol{v} \circledast \boldsymbol{w})[n] = \sum_{m=0}^{N-1} \boldsymbol{v}[n-m]\boldsymbol{w}[m], \tag{2.1}$$

where periodic boundary conditions are assumed, i.e. $\boldsymbol{v}[N+k] = \boldsymbol{v}[k], \quad \forall k \in \mathbb{Z}$. [Yin et al., 2017, p. 5]

This is expanded to include vectors of differing sizes using zero-padding,

Definition 2.2 (Circular convolution)
The circular convolution in $\mathbb{R}^N$ of any two vectors $\boldsymbol{v} \in \mathbb{R}^{N_1}$ and $\boldsymbol{w} \in \mathbb{R}^{N_2}$ with $N_1, N_2 \leq N$ is defined as

$$\boldsymbol{v} \circledast \boldsymbol{w} = \boldsymbol{v}^0 \circledast \boldsymbol{w}^0, \tag{2.2}$$

where $\boldsymbol{v}^0 = \begin{bmatrix} \boldsymbol{v}^\top & \mathbf{0}_{N-N_1}^\top \end{bmatrix}^\top$ and $\boldsymbol{w}^0 = \begin{bmatrix} \boldsymbol{w}^\top & \mathbf{0}_{N-N_2}^\top \end{bmatrix}^\top$ denote the length-N zero-padded versions of $\boldsymbol{v}$ and $\boldsymbol{w}$. $\mathbf{0}_{N-N_1}$ is the zero vector in $\mathbb{R}^{N-N_1}$ and $\mathbf{0}_{N-N_2}$ is the

zero vector in $\mathbb{R}^{N-N_2}$.
[Yin et al., 2017, p. 5]

Another important vector operation is the flip of a vector,

Definition 2.3 (Vector flip)
For any $\boldsymbol{v} \in \mathbb{R}^{N_1}$ with $N_1 \leq N$, define the flip of $\boldsymbol{v}$ as

$$\overleftarrow{\boldsymbol{v}}[n] = \boldsymbol{v}^0[-n]. \tag{2.3}$$

[Yin et al., 2017, p. 6]

Next, frames and tight frames are defined,

Definition 2.4 (Frame)
A countable family of elements $\{\boldsymbol{\psi}_i\}_{i=0}^{M-1}$ in an inner product space V is a frame for V, if there exist constants $l, k > 0$ such that

$$k\|\boldsymbol{f}\|^2 \leq \sum_{i=0}^{M-1} |\langle \boldsymbol{f}, \boldsymbol{\psi}_i \rangle|^2 \leq l\|\boldsymbol{f}\|^2, \quad f \in V. \tag{2.4}$$

[Christensen, 2008, p. 3]

Definition 2.5 (Tight frame)
A frame $\{\boldsymbol{\psi}_i\}_{i=0}^{M-1}$ of an inner product space V, is defined as tight if the following holds,

$$\sum_{i=0}^{M-1} |\langle \boldsymbol{f}, \boldsymbol{\psi}_i \rangle|^2 = k\|\boldsymbol{f}\|^2, \tag{2.5}$$

for all $\boldsymbol{f} \in V$, where k, known as the frame bound, is a constant.
[Christensen, 2008, p. 5]

The following proposition regarding tight frames is useful for showing that a given frame is tight.

Proposition 2.1
$\{\boldsymbol{\psi}_i\}_{i=1}^m$ is a tight frame for an inner product space V with frame bound k if and only if

$$\boldsymbol{f} = \frac{1}{k}\sum_{i=1}^{m}\langle \boldsymbol{f}, \boldsymbol{\psi}_i\rangle \boldsymbol{\psi}_i, \quad \forall \boldsymbol{f} \in V. \tag{2.6}$$

[Christensen, 2008, p. 5]

Now that all the necessary definitions and propositions have now been introduced, the convolution framelet can be defined.

Definition 2.6
A convolution framelet $\boldsymbol{\Psi}$ is defined as the set of vectors

$$\boldsymbol{\psi}_{ij} = \frac{1}{\sqrt{l}}\boldsymbol{\phi}_i \circledast \boldsymbol{v}_j, \quad i = 1, ..., N, \quad j = 1, ..., l, \tag{2.7}$$

where $\boldsymbol{\phi}_i$ and $\boldsymbol{v}_j$ are orthonormal bases for $\mathbb{R}^N$ and $\mathbb{R}^l$ respectively.
[Yin et al., 2017, p. 6]

To describe convolution framelets using matrix multiplication as in [Yin et al., 2017, p. 6], the concept of the patch matrix is required. Consider a one-dimensional signal-vector $\boldsymbol{f} = \begin{bmatrix}\boldsymbol{f}[0] & \boldsymbol{f}[1] & ... & \boldsymbol{f}[N-1]\end{bmatrix}^\top \in \mathbb{R}^N$ and assume periodic boundary conditions for $\boldsymbol{f}$. The patch size l is a fixed integer between 1 and N, and for any integer $m \in [0, ..., N-1]$, the vector $\boldsymbol{F}_m = \begin{bmatrix}\boldsymbol{f}[m] & ... & \boldsymbol{f}[m+l-1]\end{bmatrix}^\top \in \mathbb{R}^l$ is called the patch of $\boldsymbol{f}$ at m with length l. The patch matrix of $\boldsymbol{f}$ is constructed by vertically stacking the patches according to their order of appearance in the original signal $\boldsymbol{f}$ and is denoted $\boldsymbol{F} \in \mathbb{R}^{N\times l}$:

$$\boldsymbol{F} = \begin{bmatrix}\boldsymbol{F}_0 & ... & \boldsymbol{F}_{N-1}\end{bmatrix}^\top. \tag{2.8}$$

The matrix-vector product of $\boldsymbol{F}$ with any vector $\boldsymbol{v} \in \mathbb{R}^l$ can be written as a convolution:

$$\boldsymbol{F}\boldsymbol{v} = \boldsymbol{f} \circledast \overleftarrow{\boldsymbol{v}}. \tag{2.9}$$

From [Yin et al., 2017, p. 9] the coefficient matrix can be constructed using the patch matrix $\boldsymbol{F}$ and two orthonormal matrices $\boldsymbol{\Phi}$ and $\boldsymbol{V}$:

$$\boldsymbol{C} = \boldsymbol{\Phi}^\top \boldsymbol{F}\boldsymbol{V}, \tag{2.10}$$

where $\boldsymbol{F} \in \mathbb{R}^{N\times l}$, $\boldsymbol{\Phi} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{V} \in \mathbb{R}^{l\times l}$. This (2.10) is the practical method of calculating $\boldsymbol{C}$ used in this report. This is because it requires no convolutions and does not explicitly require the potentially large amount of frame elements to be calculated and stored.

Finally, down and up sampling are defined in order to control the level of redundancy in (2.10).

Definition 2.7 (down and up sampling)
Let $\boldsymbol{f} \in \mathbb{R}^N$, and let $h < N$ be a positive integer. Define the down sampling of $\boldsymbol{f}$ by h as,

$$(\boldsymbol{f} \downarrow h)[n] = \boldsymbol{f}[nh], \quad n = 0, \dots, \left\lfloor \frac{N-1}{h} \right\rfloor. \tag{2.11}$$

Define the up sampling of $\boldsymbol{f}$ by h as,

$$(\boldsymbol{f} \uparrow h)[n] = \begin{cases} \boldsymbol{f}[\frac{n}{h}] & \text{for } \frac{n}{h} \in \mathbb{Z} \\ 0 & \text{o.w.} \end{cases}, \quad n = 0, \dots, Nh-1. \tag{2.12}$$

2.2 Reduced patch matrix

The goal in this section is to generalize the convolution framelet, [Yin et al., 2017], to patches with less overlap, and thus less redundancy. The patch matrix is originally constructed from patches that are shifted one sample relative to their neighbour. We will show that for suitable choices of hop, patch and block size, one will obtain a tight frame when applying orthonormal transforms to the left and right of a reduced patch matrix. To see that this is a generalization of convolution framelets as defined in 2.6, we will show that each frame coefficient corresponds to a convolution between an element from each of the orthonormal bases, where the left element has been up sampled.

Let $\boldsymbol{f} \in \mathbb{R}^N$ and let l be a positive integer denoting patch size. Consider the patch matrix, $\boldsymbol{F} \in \mathbb{R}^{N \times l}$, see (2.8), defined by [Yin et al., 2017, p. 5]. Denote the rows of this matrix as the full ordered set of patches of $\boldsymbol{f}$. Let h be a positive integer, and select an ordered subset of the patches, starting with the first, followed by every h'th patch in order. h then denotes the hop size of the reduced patch matrix, $\boldsymbol{F}_{\downarrow h}$, constructed by vertically stacking the selected subset of patches. Say for example that $h = 2$, then every second patch is selected for the reduced patch matrix. $h = 1$ corresponds to the full patch matrix. We use the $\downarrow h$ notation, because $\boldsymbol{F}_{\downarrow h}$ can be considered a down sampled version of $\boldsymbol{F}$ in terms of patches.

Let $\boldsymbol{\Phi}$ and $\boldsymbol{V}$ be orthonormal matrices. The question is, for which values of h, does the following equation (2.13) yield a tight frame for $\boldsymbol{f}$?

$$\boldsymbol{C}_{\downarrow h} := \boldsymbol{\Phi}^T \boldsymbol{F}_{\downarrow h} \boldsymbol{V}. \tag{2.13}$$

The reduced patch matrix is constructed exclusively from entries of $\boldsymbol{f}$. If the reduced patch matrix contains each entry of $\boldsymbol{f}$ exactly k times, where k is some positive integer, the reduced patch matrix will be called balanced. For a balanced reduced patch matrix it follows that the squared Frobinius norm of the reduced patch matrix will be equal to the squared 2-norm of $\boldsymbol{f}$ scaled by k, that is

$$k\|\boldsymbol{f}\|_2^2 = \|\boldsymbol{F}_{\downarrow h}\|_F^2, \tag{2.14}$$

for all $\boldsymbol{f} \in \mathbb{R}^N$. Since the Frobinius norm is invariant under orthonormal transforms, (2.14) extends to

$$k\|\boldsymbol{f}\|_2^2 = \|\boldsymbol{F}_{\downarrow h}\|_F^2 = \|\boldsymbol{C}_{\downarrow h}\|_F^2. \tag{2.15}$$

This expression corresponds to taking the inner product between (2.6) and $\boldsymbol{f}$ in Proposition 2.1. A given entry of $\boldsymbol{f}$ appears in $\boldsymbol{F}_{\downarrow h}$ as many times as it is covered by a patch. The following lemma places conditions on N and l, given h, which result in a balanced $\boldsymbol{F}_{\downarrow h}$.

Lemma 2.1
Let $\boldsymbol{f} \in \mathbb{R}^N$, and $h \in \{1, 2, 3, \ldots\}$, then the reduced patch matrix $\boldsymbol{F}_{\downarrow h}$ is balanced, containing each entry of $\boldsymbol{f}$ exactly k times, if and only if $N > l$ are both integer multiples of h, with $k := \frac{l}{h}$.

Proof. First it will be shown that each entry of $\boldsymbol{f}$ is contained in the same number of patches, under the assumptions of the lemma.

- Assume that l and N are integer multiples of h. Denote the shift between the first entries of two patches, in terms of $\boldsymbol{f}$ with circularity, as the relative shift between those patches. Note, that the relative shifts between patches in the reduced patch matrix must be the integer multiples of h, because neighbouring patches have a relative shift of h. The fact that N is divisble by h, ensures that this holds even when the shift passes the boundary of $\boldsymbol{f}$.

  Consider the last entry, $\boldsymbol{f}[i]$, of any patch, then the first entry of this patch is $\boldsymbol{f}[i + 1 - l]$. The patch with $\boldsymbol{f}[i + 1]$ as the first entry, must be included in the reduced patch matrix, because these two patches have a relative shift of l, which was assumed to be an integer multiple of h. Thus a new patch always begins where another ends, meaning that the total number of patches containing any entry of $\boldsymbol{f}$ is constant. Figure 2.1 provides an illustration to help visualize the patches in relation to $\boldsymbol{f}$.

Now it will be shown that under the assumptions of the lemma, the number of patches covering any entry of $\boldsymbol{f}$ is exactly k. As an illustrative aid, see Figure 2.1 (A)

- Consider again the last entry, $\boldsymbol{f}[i]$, of any patch, then it follows that this is the first patch to contain $\boldsymbol{f}[i]$. Since $\boldsymbol{f}[i]$ is the last entry of the patch, all following patches with relative shift less than a full patch size, l, must contain $\boldsymbol{f}[i]$ as well.

Because each patch is shifted by h relative to its neighbour, there are $\frac{l}{h} = k$ patches following this patch, including itself, with relative shift less than l. Finally, by the construction of the reduced patch matrix, no further patches contain $f[i]$. Thus, since all entries were already shown to be contained in the same number of patches, all entries must be contained in exactly k patches.

The implication has been shown to hold in one direction and the converse will now be shown. First, it will be shown that there must exist two entries of $\boldsymbol{f}$ contained in a different number of patches, when l is not an integer multiple of h. For an illustration of these cases, see Figures 2.1 (B) and (D).

- Assume that l is not an integer multiple of h and consider the first entry $\boldsymbol{f}[i]$ of the last patch. $\boldsymbol{f}[i-1]$ can not be the last entry of any non boundary wrapping patch. This is because a relative shift, not passing the boundary, w.r.t. the last patch of l would be required. This is not possible, since relative shifts, not passing the boundary, must be an integer multiple of h and l was assumed not to be an integer multiple of h. For $f[i-1]$ to be the last entry of any boundary wrapping patch, it would be required that $l \geq N$, which is also against the assumptions. As such, $\boldsymbol{f}[i]$ and $\boldsymbol{f}[i-1]$ must be contained in a different number of patches.

Finally it remains to be shown that there must exist two entries of $\boldsymbol{f}$ contained in a different number of patches, when l is an integer multiple of h while N is not. This final case is illustrated in Figure 2.1 (C).

- Assume that l is an integer multiple of h and that N is not. Consider the last entry $\boldsymbol{f}[i]$ of the last boundary wrapping patch. Since l is an integer multiple of h and N is not, the number of entries after wrapping around the boundary in this patch must not be an integer multiple of h. This is because the number of samples before wrapping around the boundary can not be an integer multiple of h, since this would require N to be an integer multiple of h. This means that $\boldsymbol{f}[i+1]$ can not be the first entry of any patch. Thus $\boldsymbol{f}[i]$ and $\boldsymbol{f}[i+1]$ must be contained in a different number of patches.

Since the patch matrix is made by stacking the patches, this proves the lemma. $\square$

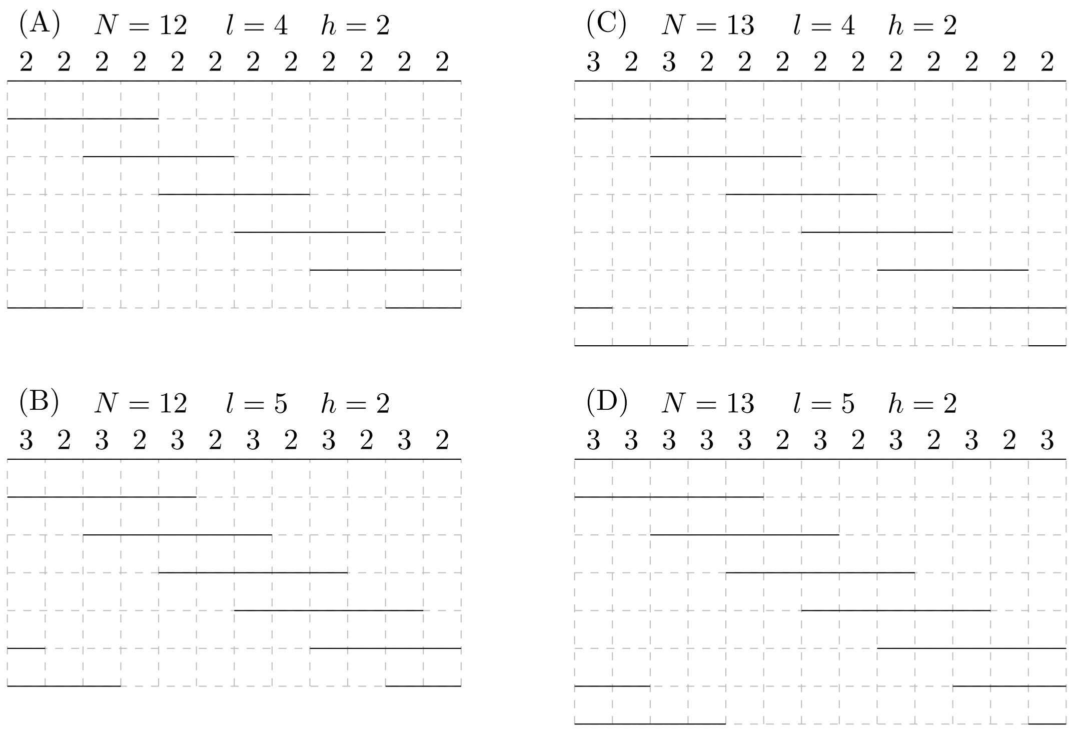


Figure 2.1: Four case illustrations of how many patches contain each entry of $\boldsymbol{f}$. The lines at the top in the four figures represent $\boldsymbol{f}$, and the numbers above each entry of $\boldsymbol{f}$ show how many patches cover that entry. The patches are represented by the shorter lines below, ordered from top to bottom, and aligned horizontally with the corresponding entries of $\boldsymbol{f}$. The parameters are shown above each figure. Note in (A) how each entry of $\boldsymbol{f}$ is covered by exactly $\frac{l}{h} = 2$ patches. In (B) how there are discrepancies wherever a patch ends. In (C) how the boundary wrapping patches are misaligned with the first. Finally in (D) how the boundary wrapping patches are only able to fix the issue seen in (B) for the first few patches.

So $\boldsymbol{F}_{\downarrow h}$ is balanced if and only if N and l are integer multiples of h. Now it will be shown that applying orthonormal transforms to the left and right of a balanced reduced patch matrix, as in (2.13), yields the coefficients of $\boldsymbol{f}$ in a tight frame.

Proposition 2.2
Choose positive integers N, l and h, such that N and l are integer multiples of h and $N > l$. Let $\boldsymbol{f} \in \mathbb{R}^N$ and let $\boldsymbol{F}_{\downarrow h} \in \mathbb{R}^{\frac{N}{h} \times l}$ be the reduced patch matrix of $\boldsymbol{f}$. Let $\boldsymbol{\Phi} \in \mathbb{R}^{\frac{N}{h} \times \frac{N}{h}}$ and $\boldsymbol{V} \in \mathbb{R}^{l \times l}$ be orthonormal matrices, then

$$\Psi_{\downarrow h} = \left\{ (\boldsymbol{\phi}_i \uparrow h) \circledast \boldsymbol{v}_j \right\}, \quad i = 0, \dots, \frac{N}{h} - 1, \quad j = 0, \dots, l - 1. \tag{2.16}$$

is a tight frame for $\mathbb{R}^N$ with frame bound $\frac{l}{h}$, where $\boldsymbol{f}$ has the canonical frame coefficients,

$$\boldsymbol{C}_{\downarrow h}[i, j] = \boldsymbol{\phi}_i \boldsymbol{F}_{\downarrow h} \boldsymbol{v}_j = \boldsymbol{f}^\top \left[(\boldsymbol{\phi}_i \uparrow h) \circledast \boldsymbol{v}_j \right]. \tag{2.17}$$

Proof. By Lemma 2.1 and the choice of N, l and h, $\boldsymbol{F}_{\downarrow h}$ is balanced. Vector multiplication with $\boldsymbol{F}_{\downarrow h}$ can be described as a convolution down sampled by h as follows,

$$\begin{aligned}(\boldsymbol{F}_{\downarrow h}\boldsymbol{v})[n] &= \sum_{m=0}^{N-1} \boldsymbol{f}[nh+m]\boldsymbol{v}[m] \quad n = 0,\ldots,\frac{N}{h}-1 \\ &= ((\boldsymbol{f} \circledast \overleftarrow{\boldsymbol{v}}) \downarrow h)[n].\end{aligned} \tag{2.18}$$

Using this observation, the following identity is derived,

$$\begin{aligned}\boldsymbol{\phi}^\top [(\boldsymbol{f} \circledast \overleftarrow{\boldsymbol{v}}) \downarrow h] &= \sum_{n=0}^{\frac{N}{h}-1} \boldsymbol{\phi}[n] \sum_{m=0}^{N-1} \boldsymbol{f}[nh+m]\boldsymbol{v}[m] \\ &= \sum_{n=0}^{\frac{N}{h}-1} \boldsymbol{\phi}[n] \sum_{m=0}^{N-1} \boldsymbol{f}[m]\boldsymbol{v}[m-nh] \\ &= \sum_{m=0}^{N-1} \boldsymbol{f}[m] \sum_{n=0}^{\frac{N}{h}-1} \boldsymbol{\phi}[n]\boldsymbol{v}[m-nh] \\ &= \boldsymbol{f}^\top [(\boldsymbol{\phi}\uparrow h) \circledast \boldsymbol{v}].\end{aligned} \tag{2.19}$$

From (2.18) and (2.19), it can be seen that

$$\boldsymbol{\phi}^\top \boldsymbol{F}_{\downarrow h}\boldsymbol{v} = \boldsymbol{f}^\top [(\boldsymbol{\phi}\uparrow h) \circledast \boldsymbol{v}], \tag{2.20}$$

which means that $\boldsymbol{C}_{\downarrow h}[i,j] = \boldsymbol{f}^\top [(\boldsymbol{\phi}_i \uparrow h) \circledast \boldsymbol{v}_j]$. Finally, by (2.15), the frame, $[(\boldsymbol{\phi}_i \uparrow h) \circledast \boldsymbol{v}_j]$, is tight, with a frame bound $k = \frac{l}{h}$. □

This result means that the level of redundancy in convolution framelets can be controlled. This does come with a cost. The dimensions of $\boldsymbol{\Phi}$ depend on h, which means that for lower redundancy, this dimension is also lower. Thus, whichever basis is chosen as $\boldsymbol{\Phi}$ will have lower resolution, i.e. fewer coefficients to describe non local characteristics of $\boldsymbol{f}$. It is also worthwhile to note that Proposition 2.2 with $h = 1$ reduces to the case of using orthonormal bases in [Yin et al., 2017, Proposition 1]. The proof here is an alternate approach, which provides additional insight into why the convolutional framelet is tight.

2.3 Framelet concatenation

It is possible to concatenate multiple orthonormal bases, and use the resulting frames to construct convolution framelets.

Concatenating two or more orthonormal bases of an N-dimensional inner product space, V, results in a tight frame for that space. To verify this result, consider any vector, $\boldsymbol{f} \in V$, and let $\boldsymbol{A}$ and $\boldsymbol{B}$ be orthonormal bases of V, then $\boldsymbol{f}$ can be expressed in terms of $\boldsymbol{A}$ or $\boldsymbol{B}$ as follows,

$$\boldsymbol{f} = \sum_{i=0}^{N-1} \langle \boldsymbol{f}, \boldsymbol{a}_i \rangle \boldsymbol{a}_i = \sum_{j=0}^{N-1} \langle \boldsymbol{f}, \boldsymbol{b}_j \rangle \boldsymbol{b}_j, \tag{2.21}$$

where $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$ denote the columns of $\boldsymbol{A}$ and $\boldsymbol{B}$. Now denote the concatenation of $\boldsymbol{A}$ and $\boldsymbol{B}$ by $\boldsymbol{\Psi} = \left[\boldsymbol{A}^\top\ \boldsymbol{B}^\top\right]^\top$, then from (2.21), it follows that

$$\begin{aligned} 2\boldsymbol{f} &= \sum_{i=0}^{N-1} \langle \boldsymbol{f}, \boldsymbol{a}_i \rangle \boldsymbol{a}_i + \sum_{j=0}^{N-1} \langle \boldsymbol{f}, \boldsymbol{b}_j \rangle \boldsymbol{b}_j \\ &= \sum_{k=0}^{2N-1} \langle \boldsymbol{f}, \boldsymbol{\psi}_k \rangle \boldsymbol{\psi}_k, \end{aligned} \tag{2.22}$$

where $\boldsymbol{\psi}_k$ is the k'th element of $\boldsymbol{\Psi}$. By Proposition 2.1, (2.22) shows that Ψ is a tight frame for V with the frame bound 2.

Of course this result extends to concatenating more than two bases. Consider a series of M orthonormal bases, $\{\Phi_m\}_{m=0}^{M-1}$, of V, then once again the concatenation, $\boldsymbol{\Psi} = \left[\boldsymbol{\Phi}_0^\top\ \dots\ \boldsymbol{\Phi}_{M-1}^\top\right]^\top$, of these bases can be seen to be a tight frame for V.

$$\begin{aligned} M\boldsymbol{f} &= \sum_{m=0}^{M-1} \sum_{i=0}^{N-1} \langle \boldsymbol{f}, \boldsymbol{\phi}_{m_i} \rangle \boldsymbol{\phi}_{m_i} \\ &= \sum_{k=0}^{MN-1} \langle \boldsymbol{f}, \boldsymbol{\psi}_k \rangle \boldsymbol{\psi}_k. \end{aligned} \tag{2.23}$$

The frame bound of this frame is M. Proposition 2.2 will now be generalized from using orthonormal bases to tight frames in transforming the patch matrix.

Proposition 2.3
Choose positive integers N, l and h, such that N and l are integer multiples of h and $N > l$. Let $\boldsymbol{f} \in \mathbb{R}^N$ and let $\boldsymbol{F}_{\downarrow h} \in \mathbb{R}^{\frac{N}{h} \times l}$ be the reduced patch matrix of $\boldsymbol{f}$. Let $\boldsymbol{\Phi} \in \mathbb{R}^{\frac{N}{h} \times N'}$ and $\boldsymbol{V} \in \mathbb{R}^{l \times l'}$ be tight frames such that $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \boldsymbol{I}_{\frac{N}{h}}$ and $\boldsymbol{V}\boldsymbol{V}^\top = \boldsymbol{I}_l$, $\boldsymbol{I}$ being the identity, then

$$\Psi_{\downarrow h} = \left\{ (\boldsymbol{\phi}_i \uparrow h) \circledast \boldsymbol{v}_j \right\}, \quad i = 0, \dots, N'-1, \quad j = 0, \dots, l'-1 \tag{2.24}$$

is a tight frame for $\mathbb{R}^N$ with frame bound $\frac{l}{h}$, where $\boldsymbol{f}$ has the canonical frame coefficients,

$$\boldsymbol{C}_{\downarrow h}[i,j] = \boldsymbol{\phi}_i \boldsymbol{F}_{\downarrow h} \boldsymbol{v}_j = \boldsymbol{f}^\top \left[(\boldsymbol{\phi}_i \uparrow h) \circledast \boldsymbol{v}_j \right]. \tag{2.25}$$

Proof. It only needs to be shown that (2.15) still holds, when $\boldsymbol{\Phi}$ and $\boldsymbol{V}$ are tight frames.

That is,

$$\begin{aligned}\|\boldsymbol{C}_{\downarrow h}\|_F^2 &= \operatorname{trace}\left(\boldsymbol{V}^\top \boldsymbol{F}_{\downarrow h}^\top \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \boldsymbol{F}_{\downarrow h}\boldsymbol{V}\right)\\ &= \operatorname{trace}\left(\boldsymbol{V}\boldsymbol{V}^\top \boldsymbol{F}_{\downarrow h}^\top \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \boldsymbol{F}_{\downarrow h}\right)\\ &= \operatorname{trace}\left(\boldsymbol{F}_{\downarrow h}^\top \boldsymbol{F}_{\downarrow h}\right)\\ &= \|\boldsymbol{F}_{\downarrow h}\|_F^2. \end{aligned} \tag{2.26}$$

The proof of Proposition 2.2 can then be applied to Proposition 2.3. □

Now if either $\boldsymbol{\Phi}$ or $\boldsymbol{V}$ are constructed by concatenating M orthonormal bases, they can be made to fulfil the condition $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \boldsymbol{I}_{\frac{N}{h}}$ or $\boldsymbol{V}\boldsymbol{V}^\top = \boldsymbol{I}_l$, by scaling with $\frac{1}{M}$.

Convolution framelets can be further combined by constructing multiple framelets, with the same $\boldsymbol{f}$ and therefore N, with different combinations of l and h. Concatenating multiple tight frames results in a new tight frame, with the sum of individual frame bounds as the new frame bound. In this way, one can construct a single tight frame using any number of orthonormal bases from several spaces with different dimension.

2.3.1 Sparse coefficients

Having constructed these redundant frames, they can now be considered as systems of the form,

$$\boldsymbol{\Psi}^\top \boldsymbol{c} = \boldsymbol{f}. \tag{2.27}$$

Provided that $\boldsymbol{\Psi}$ is redundant, this system is underdetermined, and there exist infinitely many $\boldsymbol{c}$, solving it. For compression purposes it makes sense to look for a sparse solution. Naively that is a $\boldsymbol{c}$ with as many zero entries as possible. Sparsity of $\boldsymbol{c}$ is desired, because it means that optimal bit allocation will be more effective, as is discussed in Section 3.2. Formally, the ideal problem can be stated as such

$$\min_{\boldsymbol{c}} \|\boldsymbol{c}\|_0 \quad \text{s.t. } \boldsymbol{\Psi}^\top \boldsymbol{c} = \boldsymbol{f}, \tag{2.28}$$

where $\|\cdot\|_0$ denotes the l_0-"norm", which counts the number of non zero entries in a vector. This has been studied in the field of compressive sensing, which is concerned with finding the solution, $\boldsymbol{c}^\star$, to problems as stated in (2.28). The sparsity of the desired $\boldsymbol{c}^\star$ means that it can be possible to recover $\boldsymbol{c}^\star$ from (2.27) despite the underdetermined nature of the system. One of the main questions is, for which $\boldsymbol{c}^\star$ and $\boldsymbol{\Phi}$ the problem can be solved. The problem in (2.28), is shown by [Foucart and Rauhut, 2013] to be NP complete, and the l_0 definition of sparsity is sensitive to noise in practice. In general, one practical approach, which has been shown to work well with high probability, is to generate random $\boldsymbol{\Psi}$ matrices, called libraries. Using these libraries, $\boldsymbol{c}^\star$ is sought after using e.g. l_1-optimization or matching pursuit, [Foucart and Rauhut, 2013, Ch. 3 and 9]. To understand what properties a library should have in order to work well in compressive sensing, coherence and restricted isometry properties will be considered. Starting with the coherence of a library, defined as such.

Definition 2.8 (coherence)
Let $\boldsymbol{\Phi} \in \mathbb{C}^{N \times l}$ be a matrix with l_2-normalized columns, $\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{N-1}$. The coherence, $\mu(\boldsymbol{\Phi})$, of $\boldsymbol{\Phi}$ is defined as

$$\mu(\boldsymbol{\Phi}) = \max_{0 \leq i \neq j \leq N-1} |\langle \boldsymbol{\phi}_i, \boldsymbol{\phi}_j \rangle|. \tag{2.29}$$

[Foucart and Rauhut, 2013, p. 111].

Coherence can be useful to get an idea of whether a library will be useful or not. Libraries with low coherence, or informally incoherent libraries, will generally be more successful in recovering sparse solutions from (2.27), which is stated by [Foucart and Rauhut, 2013, p. 111]. Unfortunately coherence does not explain everything. There are examples of libraries that work well, despite not having low coherence. It can be intuited, that high coherence can arise locally in a library, from a few coherent elements, while the library might globally be incoherent. This description fits e.g. the random matrices, described in [Foucart and Rauhut, 2013].

To better explain the properties that make good libraries, the restricted isometry constant is defined.

Definition 2.9 (restricted isometry constant)
The restricted isometry constant $\delta_s(\boldsymbol{A})$ of order s for a matrix $\boldsymbol{A} \in \mathbb{C}^{m \times N}$ is defined as the smallest $\delta > 0$ satisfying the inequality

$$(1-\delta)\|\boldsymbol{x}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq (1+\delta)\|\boldsymbol{x}\|_2^2, \tag{2.30}$$

for all s-sparse vectors, $x \in \mathbb{C}^N$. [Foucart and Rauhut, 2013, p. 133]

The constant, δ_s, gives a better idea of whether a library will work well for $\boldsymbol{c}^\star$ with sparsity up to s. Unfortunately the definition does not give an easy method of computing δ_s. [Foucart and Rauhut, 2013, ch. 9] shows that random libraries have low restricted isometry constants with high probability, but for any given library this is not easy to check. This is one of the major open problems in compressive sensing.

In the practical Chapter 4 of this project, concatenated convolution framelets will be used as libraries, $\boldsymbol{\Psi}$, along with audio signals $\boldsymbol{f}$. Orthogonal matching pursuit will be used to search for sparse sets of coefficients $\boldsymbol{c}^\star$. Since audio is typically sparse in well known frequency transforms, [You, 2010, Ch. 5], the convolution framelets will mainly be constructed using frequency transforms.

3 | Data Compression

Data compression is a process used to obtain a compact representation of a signal, meaning a representation that requires fewer bits than the original representation of the signal, also called the source signal. The compact representation can then be used for different purposes such as processing, storage or transmission. In general, the data compression consists of two parts; the encoding giving the compact representation and the decoding scheme, which is the inverse of encoding, giving a reconstructed version of the source signal [Pu, 2005, p. 3]. In digital audio coding, the source signal is a digital sound signal, which goes through an encoder, giving a compact representation with fewer bits. The new representation is then delivered through a communication or storage channel to a decoder, producing a reconstruction of the source sound signal from the received compact representation [You, 2010, p. 5]. If the compact representation is obtained such that the decoder can perfectly reconstruct the source signal, then the compression is classified as *lossless compression.* However if some information is lost under encoding, giving an approximate reconstruction of the source signal, then it is called *lossy compression.* The compact representation is referred to as a *code* defined as a rule or mapping that specifies how source symbols or groups of such symbols are transformed into or represented by a new set of symbols [Gersho and Gray, 1993, p. 3].

Lossless compression

Lossless compression is based on only removing the elements in the source signal that are statistically redundant through encoding and thus obtaining a perfectly reconstructed signal by the decoder.

Lossy compression

Lossy compression also removes the statistically redundant elements of the source signal, but also removes the elements that are perceptually irrelevant or insignificant. Thus it causes distortion in the reconstructed signal from the decoder, although with the advantage that this removal can be done such that the distortion is imperceptible. The removal of the perceptually irrelevant elements is done by a quantizer [You, 2010, pp. 6-7]. There exist different set-ups of lossy compression. In this project, before applying the compression to the source signal, a transform is applied to the signal in order to obtain a representation of the source signal suitable for compression and then applying a lossy compression. This is referred to as transform coding [You, 2010, ch. 5]. The overall scheme of the lossy compression used can be summarized in Figure 3.1.

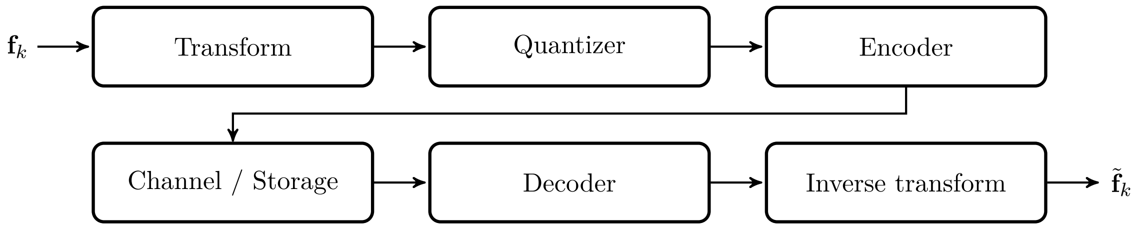


Figure 3.1: This figure shows the different steps in lossy compression of a signal $\boldsymbol{f}_k$

After the source signal, $\boldsymbol{f}_k$, has been transformed, the quantization of the transformed source signal is done through a quantizer. The quantized values are then coded in the encoder using a lossless coding scheme. A compact representation is then obtained from the encoder and can then be stored or transmitted through a channel, which is assumed in this project to be noiseless. The compact representation is then decoded by the decoder, thus giving back the quantized values. Since some information has been irreversibly lost from the quantization process, the inverse transform gives a reconstructed signal $\tilde{\boldsymbol{f}}_k$, which is an approximation of the original source signal.

In this chapter, the basics of quantization will firstly be introduced followed by bit allocation, which is a generalized version of quantization. Further the theory of lossless source coding will be presented. The compression of audio signals will be briefly introduced via audio coding. Lastly a type of binary data representation, called bitplane will be presented.

3.1 Quantization

The quantization is mainly used in the case of the conversion of analogue signals to digital signals. However in this project, the processed signals are digital and so have already been quantized. Thus, the quantization process here is a re-quantization in order to reduce the number of bits used to describe the digital source signal even more. The following description of the quantization process is inspired by [You, 2010] and [Gersho and Gray, 1993], with a focus on discrete time signals. This section covers essential definitions and notation related to quantization and quantization error. Furthermore quantization is presented as a rate-distortion optimization problem, where it is possible to fix a given rate, and optimize the distortion or vice versa.

Definition 3.1 (Scalar quantizer)
A scalar quantizer Q of size N is defined to be a mapping from an input set $\mathcal{X}$, into a finite set $\mathcal{C}$ called the codebook, with size M corresponding to the resolution of the quantizer. Thus,

$$Q\colon \mathcal{X} \mapsto \mathcal{C},$$

where $\mathcal{C} = \{\hat{x}_q\}_{q=1}^{M}$ and $\hat{x}_q \in \mathbb{R}$, for $q = 1, \ldots, M$, are called output values or representative values. [Gersho and Gray, 1993, p. 133]

The scalar quantization is applied as follows. Consider a digital source signal, modelled as a discrete random variable X, since the source signal is already a quantized signal. Let $p(X)$ denote its probability mass function (pmf). In order to quantize this source signal, choose M decision intervals defined by the following $M + 1$ endpoints

$$\{b_q\}_{q=0}^{M}, \tag{3.1}$$

called decision boundaries.

A source sample value x is quantized to the quantization index q if and and only if x falls into the qth decision interval

$$\delta_q = [b_{q-1}; b_q). \tag{3.2}$$

This operation is called forward quantization and is expressed as follows.

Definition 3.2 (Forward quantization)
Forward quantization is defined as the mapping $\mathcal{E}\colon \mathcal{X} \mapsto \mathcal{I}$, where $\mathcal{I} = \{1, 2, 3, \ldots, M\}$. This is,

$$\mathcal{E}(x) = q, \quad \text{if and only if} \quad b_{q-1} \leq x < b_q. \tag{3.3}$$

[You, 2010, p. 21]

Thus, after obtaining the quantization index, the next step is to obtain a quantized value from the quantization index. This operation is called backward quantization and can be defined mathematically as the following.

Definition 3.3 (Backward quantization)
Backward quantization is defined as the mapping $\mathcal{D}\colon \mathcal{I} \mapsto \mathcal{C}$, where $\mathcal{C} = \{\hat{x}_q\}_{q=1}^{M}$. This is

$$\mathcal{D}(\mathcal{E}(x)) = \mathcal{D}(q) = \hat{x}_q. \tag{3.4}$$

[Gersho and Gray, 1993, p. 137]

The quantized value $\hat{x}_q$ and the source sample value x are different to each other and induce a loss of information in the quantized source signal. This distortion is called quantization noise or quantization error and is defined as

$$d(x, \hat{x}_q) = x - \hat{x}_q. \tag{3.5}$$

In order to analyse the loss of information caused by the quantization process, the mean squared quantization error (MSQE) is usually used [You, 2010, p. 22]. Consider the discrete random source signal X, with range $x_1, \ldots, x_N$ and pmf $p(X)$ and the quantizer $Q(X)$ from Definition 3.1. Using (3.5), the MSQE, denoted D, can be written as

$$D = \mathrm{E}\left[(X - Q(X))^2\right] = \sum_{k=1}^{N} (x_k - Q(x))^2 p(x_k). \tag{3.6}$$

For each quantized value $\hat{x}_q$ obtained by quantizing x_k according to M decision intervals defined as in (3.2), the MSQE in (3.6) is rewritten as

$$D = \sum_{x_k \in \delta_q} \sum_{q=1}^{M} (x_k - \hat{x}_q)^2 p(x_k), \quad k = 1, 2, \ldots \tag{3.7}$$

Analysing (3.7) shows that the smaller the quantization error in (3.5) is, the smaller MSQE will be. If the decision intervals are chosen to be small, the quantized value $\hat{x}_q$ will be closer to x, which will also reduces the MSQE. The problem of choosing the optimal quantizer in terms of MSQE can be stated in different ways. The goal is to obtain a shorter representation of the digital source signal, thus reducing the number of bits used to describe the digital source signal. Consider the number of bits needed to describe the quantized value as the resolution or code rate R [Gersho and Gray, 1993, p. 134]. If the quantized values are described by binary codewords of *fixed* size, then the code rate R is

$$R = \lceil \log_2 M \rceil, \tag{3.8}$$

where M is the size of the codebook or the number of the quantized values [Sayood, 2005, p. 231]. However if the binary codewords are *variable*-length, then the code rate will depend on the probability of occurrence of the quantized values [Sayood, 2005, p. 232]. The code rate will thus be the expected length of the codeword needed to describe the quantized value

$$R = \sum_{q=1}^{M} l_q p(\hat{x}_q), \tag{3.9}$$

where M is the number of quantized values, l_q, the length of the codeword corresponding to the quantized value $\hat{x}_q$ and $p(\hat{x}_q)$, the probability of occurrence of $\hat{x}_q$. Since the quantized values are chosen according to the decision boundaries in (3.1), the probabilities $\{p(\hat{x}_q)\}$ will depend on the decision boundaries [Sayood, 2005, p. 232]. Thus the probability of occurrence of $\hat{x}_q$ is

$$p(\hat{x}_q) = \sum_{\hat{x}_k \in \delta_q} p(x_k). \tag{3.10}$$

Using (3.10), the code rate in (3.9) is rewritten as

$$R = \sum_{\hat{x}_k \in \delta_q} \sum_{q=1}^{M} l_q p(x_k). \tag{3.11}$$

The code rate of the quantizer is thus dependent on the chosen intervals and the quantized values. As described in [Sayood, 2005, p. 233] and [You, 2010, p. 23] the quantization process can be stated as one of the following optimization problems.

Find the decision boundaries and quantized values, such that the code rate is minimized, given a constraint on the MSQE, this is

$$\begin{aligned} \underset{b_i,\hat{x}_q}{\text{minimize}} \quad & R = \sum_{\hat{x}_k \in \delta_q} \sum_{q=1}^{M} l_q p(x_k), \quad k = 1,2,\dots \text{ and } i = 0,1,\dots,M. \\ \text{subject to} \quad & D \leq D^*. \end{aligned} \tag{3.12}$$

In this minimization problem, the code rate used is for *variable*-length codewords. For the code rate of quantized value described by codewords of same length as in (3.8), one can see that when M decreases, the code rate R will decrease. The optimization problem will thus be equivalent to finding the decision boundaries and quantized values such that M is minimized given a constraint on the MSQE

$$\begin{aligned} \underset{b_i,\hat{x}_q}{\text{minimize}} \quad & M \\ \text{subject to} \quad & D \leq D^*. \end{aligned} \tag{3.13}$$

The other optimization problem is stated as finding the decision boundaries and quantized values, such that the MSQE is minimized, given a constraint on the code rate R. This is,

$$\begin{aligned} \underset{b_i,\hat{x}_q}{\text{minimize}} \quad & D = \sum_{x_k \in \delta_q} \sum_{q=1}^{M} (x_k - \hat{x}_q)^2 p(x_k), \quad k = 1,2,\dots \text{ and } i = 0,1,\dots,M. \\ \text{subject to} \quad & R \leq R^*. \end{aligned} \tag{3.14}$$

Equivalently for the code rate defined in (3.8), this optimization problem can be stated as finding the decision boundaries and quantized values that minimize the MSQE given a fixed number of quantized values M

$$\begin{aligned} \underset{b_i,\hat{x}_q}{\text{minimize}} \quad & D \\ \text{subject to} \quad & M = c, \end{aligned} \tag{3.15}$$

where c is some constant.

The problem of minimizing distortion with a fixed code rate will be of most practical use here. While the problem in (3.14) will not be solved directly, the notion of restricting the code rate and seeking to minimize distortion will be seen in the practical approach of this project.

3.1.1 Optimal quantizer

In order to have an optimal quantizer, some necessary conditions have to be fulfilled. Since the quantization process is divided into two parts, forward quantization, i.e. choosing the decision intervals or cells, and backward quantization, i.e. obtaining the codebook. Optimal quantizers will be important in deriving optimal bit allocation in Section 3.2.

The first necessary condition for optimality is to find the best partition cells for a given codebook [Gersho and Gray, 1993, p. 175]. This condition corresponds to the nearest neighbour condition presented in the following proposition.

Proposition 3.1 (Nearest neighbour condition)
Let $\mathcal{X}$ be an input set. For a given codebook, $\mathcal{C} = \{\hat{x}_q\}_{q=1}^{M}$, the partition cells satisfy

$$\delta_j = \{x \in \mathcal{X} \colon |x - \hat{x}_j| \leq |x - \hat{x}_i| \quad \text{for all} \quad j \neq i\}\,. \tag{3.16}$$

[Gersho and Gray, 1993, pp. 175-193]

The nearest neighbour condition says that the j'th cell of the partition should consist of all input values closer to $\hat{x}_j$ than to any other output value [Gersho and Gray, 1993, p. 175]. Thus the output value is chosen such that for a given input value, the absolute distortion between the input value and the output value is minimized.

The second necessary condition for optimality is to find the optimal codebook given a cell partition. This condition corresponds to the centroid condition. Since the distortion measure used is the MSQE, the centroid condition w.r.t. MSQE is presented in the following proposition.

Proposition 3.2 (Centroid condition)
Given a nondegenerate partition $\{\delta_q\}$, i.e. $\delta_i \neq \delta_j$ for $i \neq j$, the unique optimal codebook for a random variable X w.r.t. MSQE is given by

$$\hat{x}_q = \mathrm{E}\left[X | X \in \delta_q\right]. \tag{3.17}$$

[Gersho and Gray, 1993, pp. 177-193]

The centroid condition says that the optimal output value $\hat{x}_q$ for a given q'th cell of the partition is the one corresponding to the mean of all the points belonging to the q'th cell partition, i.e. the center of mass of the given partition.

The third condition, which is only necessary when the input values are discrete random variables, is mentioned in [Gersho and Gray, 1993, p. 193]. It states that the input random variable must have zero probability of occurring at a boundary between nearest neighbor cells.

If a quantizer satisfies those 3 necessary conditions, it will be qualified to be an optimal quantizer. The centroid condition in Proposition 3.2 implies some results on the quantizer that are summarized in the following lemma.

Lemma 3.1
Given a discrete random variable X. Let $Q(X)$ be a quantizer whose codebook satisfies the centroid condition. Then

$$\mathrm{E}\left[Q(X) - X\right] = 0 \tag{3.18}$$

$$\mathrm{E}\left[Q(X)(Q(X) - X\right] = 0 \tag{3.19}$$

$$\mathrm{E}\left[(X - Q(X))^2\right] = \mathrm{E}\left[X^2\right] - \mathrm{E}\left[Q(X)^2\right] = \sigma_X^2 - \sigma_{Q(x)}^2, \tag{3.20}$$

where σ_X^2 and $\sigma_{Q(x)}^2$ are the variances of X and $Q(x)$ respectively.
[Gersho and Gray, 1993, pp. 180-181]

Lemma 3.1 states that the mean of a quantizer that satisfies the centroid condition is the same as the mean of the input signal. The output value, i.e. the quantizer output is uncorrelated with the distortion and the MSQE is the difference between the variances of the input signal and quantized output [Gersho and Gray, 1993, pp. 180-181].

3.2 Bit allocation

Compression of time domain signals, is often done in a different domain related by a transform, further details in Section 3.4.1. It is often the case in practice that each coefficient in this transform domain has its own distribution. When quantizing the new signal representation, one will use different quantizers for different coefficients. Since coefficients have different quantizers, a bit allocation strategy is necessary, which corresponds to assigning a number of bits to each of quantizers, such that the distortion of the quantization scheme is optimized [Gersho and Gray, 1993, p. 225].

3.2.1 The bit allocation problem

Let $\boldsymbol{x} = \begin{bmatrix} x_1, & x_2, & \dots, & x_{N_f} \end{bmatrix}$ be some transform coefficients. Recall from Section 3.1, that the quantization process gives a quantization error. Let m_i, be the number of bits available to the quantization of x_i. Using (3.7), the MSQE resulting by quantizing x_i with m_i bits of resolution is given by

$$d_i = \mathrm{E}\left[(x_i - Q_i(x_i, m_i))^2\right]. \tag{3.21}$$

Since the number of coefficients is N_f, the average distortion of the quantization process is defined from (3.21) as

$$D = \frac{1}{N_f}\sum_{i=1}^{N_f} d_i. \tag{3.22}$$

As presented in [Gersho and Gray, 1993, p. 227] and [Spanias et al., 2006, p. 70], the bit allocation problem corresponds to finding the optimal way to allocate m_i bits across the different transform coefficients given a fixed number of available bits M, such that the average distortion in (3.22) is minimized. This is,

$$\begin{aligned} &\underset{m_i}{\text{minimize}} \quad D = \frac{1}{N_f}\sum_{i=1}^{N_f} d_i, \quad i = 1, 2, \ldots, N_f. \\ &\text{subject to} \quad \sum_{i=1}^{N_f} m_i \leq M, \\ &\text{and} \quad m_i \geq 0, \quad i = 1, 2, \ldots, N_f. \end{aligned} \tag{3.23}$$

One possible solution to the minimization problem in (3.23) can be obtained as shown in [Gersho and Gray, 1993, pp. 227-231]. In this approach, the following assumptions are made:

- The transform coefficients are a set of N_f random variables, with known variances σ_i^2, for $i = 1, 2, \ldots, N_f$.
- Each quantizer, Q_i, satisfies the necessary conditions mentioned in Section 3.1.1 for it to be an optimal quantizer [Gersho and Gray, 1993, pp. 175-193].

Furthermore, the solution to the optimization problem will be developed in the case of high resolution quantization, that is many decision intervals with a small width [Gersho and Gray, 1993, p. 161]. The high resolution quantization case permits one to treat the minimization problem as a continuous optimization problem allowing the use of calculus to solve the problem, without having to take into consideration the constraint of having only integers bits. Combining the high resolution case with the two assumptions mentioned above, the MSQE for each optimal quantizer is given by the distortion variance for the i'th quantizer as

$$d_i \approx h_i \sigma_i^2 2^{-2m_i}, \tag{3.24}$$

where the constant h_i depends on the probability distribution of the normalized random variable x_i/σ_i. Using (3.24), the bit allocation problem in (3.23) is explicitly written as

$$\begin{aligned} &\underset{m_i}{\text{minimize}} \quad D = \frac{1}{N_f}\sum_{i=1}^{N_f} h_i \sigma_i^2 2^{-2m_i}, \quad i = 1, 2, \ldots, N_f. \\ &\text{subject to} \quad -(\sum_{i=1}^{N_f} m_i - M) \geq 0, \\ &\text{and} \quad m_i \geq 0, \quad i = 1, 2, \ldots, N_f. \end{aligned} \tag{3.25}$$

The bit allocation problem is a constrained minimization problem. For now the constraint on the positivity of the bits is ignored. Thus the problem can be solved using the Lagrange multipliers theorem, which states that a set $\{m_i\}$ solving the minimization problem must satisfy that

$$\begin{aligned}
\frac{\partial L\left(\{m_i\},\mu\right)}{\partial m_k} &= 0, \quad k = 1,\dots,N_f \\
\frac{\partial L\left(\{m_i\},\mu\right)}{\partial \mu} &= 0, \\
\mu &\geq 0, \\
-\mu\left(\sum_{i=1}^{N_f} m_i - M\right) &\geq 0,
\end{aligned}$$

where μ is the Lagrange multiplier associated with the inequality constraint and $L\left(\{m_i\},\mu\right)$ is the Lagrangian given by

$$L\left(\{m_i\},\mu\right) = \frac{1}{N_f}\sum_{n=1}^{N_f} h_n\sigma_n^2 2^{-2m_n} + \mu\left(\sum_{j=1}^{N_f} m_j - M\right). \tag{3.26}$$

The partial derivatives of the Lagrangian in (3.26), are given by

$$\begin{aligned}
\frac{\partial L\left(\{m_i\},\mu\right)}{\partial m_k} &= \frac{\partial}{\partial m_k}\left(\frac{1}{N_f}\sum_{n=1}^{N_f} h_n\sigma_n^2 2^{-2m_n} + \mu\left(\sum_{j=1}^{N_f} m_j - M\right)\right) \\
&= \frac{1}{N_f}\sum_{n=1}^{N_f}\frac{\partial}{\partial m_k}\left(h_n\sigma_n^2 2^{-2m_n}\right) + \mu\left(\sum_{j=1}^{N_f}\frac{\partial m_j}{\partial m_k} - \frac{\partial M}{\partial m_k}\right) \\
&= -\frac{2}{N_f}\ln(2)h_k\sigma_k^2 2^{-2m_k} + \mu.
\end{aligned} \tag{3.27}$$

Next, (3.27) is now equated to zero and solved for m_i, this is

$$\begin{aligned}
-\frac{2}{N_f}\ln(2)h_k\sigma_k^2 2^{-2m_k} + \mu = 0 \\
2^{-2m_k} &= \frac{\mu N_f}{2\ln(2)h_k\sigma_k^2} \\
-2m_k &= \log_2\left(\frac{N_f}{2\ln(2)h_k\sigma_k^2}\right) + \log_2(\mu) \\
m_k &= \frac{1}{2}\log_2\left(\frac{2\ln(2)h_k\sigma_k^2}{N_f}\right) - \frac{1}{2}\log_2(\mu).
\end{aligned} \tag{3.28}$$

The expression of μ is needed in order to have a final expression of m_k. The partial derivative w.r.t to μ is found and equated to zero by solving the following equation

$$\begin{aligned}\frac{\partial}{\partial\mu}\left(\frac{1}{N_f}\sum_{n=1}^{N_f}h_n\sigma_n^2 2^{-2m_n}+\mu\left(\sum_{j=1}^{N_f}m_j-M\right)\right)&=0\\ \sum_{j=1}^{N_f}m_j-M&=0 \qquad (3.29)\\ \sum_{j=1}^{N_f}m_j&=M.\end{aligned}$$

Inserting (3.28) in (3.29) gives

$$\begin{aligned}\sum_{i=1}^{N_f}\left(\frac{1}{2}\log_2\left(\frac{2\ln(2)h_i\sigma_i^2}{N_f}\right)-\frac{1}{2}\log_2(\mu)\right)&=M\\ \frac{N_f}{2}\log_2(\mu)&=\sum_{i=1}^{N_f}\frac{1}{2}\log_2\left(\frac{2\ln(2)h_i\sigma_i^2}{N_f}\right)-M\\ \log_2(\mu)&=\frac{1}{N_f}\sum_{i=1}^{N_f}\log_2\left(\frac{2\ln(2)h_i\sigma_i^2}{N_f}\right)-\frac{2M}{N_f} \qquad (3.30)\\ \log_2(\mu)&=\log_2\left(\left(\prod_{i=1}^{N_f}\frac{2\ln(2)h_i\sigma_i^2}{N_f}\right)^{\frac{1}{N_f}}\right)-\frac{2M}{N_f}\\ \mu&=\left(\prod_{i=1}^{N_f}\frac{2\ln(2)h_i\sigma_i^2}{N_f}\right)^{\frac{1}{N_f}}\cdot 2^{-\frac{2M}{N_f}}.\end{aligned}$$

The obtained expression of μ in (3.30) is inserted in (3.28) giving

$$m_i=\frac{1}{2}\log_2\left(\frac{2\ln(2)h_i\sigma_i^2}{N_f}\right)-\frac{1}{2}\log_2\left(\left(\prod_{i=1}^{N_f}\frac{2\ln(2)h_i\sigma_i^2}{N_f}\right)^{\frac{1}{N_f}}\cdot 2^{-\frac{2M}{N_f}}\right). \qquad (3.31)$$

The two logarithm terms in the above equation can be decomposed in a sum of logarithms. Firstly the logarithm to the left is equal to,

$$\log_2\left(\frac{2\ln(2)}{N_f}\right)+\log_2(h_i)+\log_2\left(\sigma_i^2\right), \qquad (3.32)$$

and secondly the logarithm to the right, is equal to

$$\log_2\left(\frac{2\ln(2)}{N_f}\right)+\log_2\left(\left(\prod_{i=1}^{N_f}h_i\right)^{\frac{1}{N_f}}\right)+\log_2\left(\left(\prod_{i=1}^{N_f}\sigma_i^2\right)^{\frac{1}{N_f}}\right)-\frac{2M}{N_f}. \qquad (3.33)$$

The logarithm terms in (3.32) and (3.33) are thus used in (3.31) to obtain the final expression of m_i

$$m_k = \frac{M}{N_f} + \frac{1}{2}\log_2\left(\frac{\sigma_k^2}{\left(\prod_{i=1}^{N_f}\sigma_i^2\right)^{\frac{1}{N_f}}}\right) + \frac{1}{2}\log_2\left(\frac{h_k}{\left(\prod_{i=1}^{N_f}h_i\right)^{\frac{1}{N_f}}}\right), \tag{3.34}$$

where $\frac{M}{N_f}$ corresponds to the average rate or average number of bits per coefficient [Gersho and Gray, 1993, pp. 226-234], [Sayood, 2005, pp. 407 - 408] and [Bosi and Goldberg, 2002, pp. 205-210]. The optimal bit allocation, in the case of high resolution quantization, is thus given by (3.34) for nonidentically distributed random variables. If the normalized coefficients are identically distributed random variables, the term h_i will be identical for all random variables, since it depends on the normalized distribution, and the optimal bit allocation in (3.34) will be reduced to

$$\begin{aligned} m_k &= \frac{M}{N_f} + \frac{1}{2}\log_2\left(\frac{\sigma_k^2}{\left(\prod_{i=1}^{N_f}\sigma_i^2\right)^{\frac{1}{N_f}}}\right) + \frac{1}{2}\log_2\left(\frac{h}{\left(\prod_{i=1}^{N_f}h\right)^{\frac{1}{N_f}}}\right) \\ &= \frac{M}{N_f} + \frac{1}{2}\log_2\left(\frac{\sigma_k^2}{\left(\prod_{i=1}^{N_f}\sigma_i^2\right)^{\frac{1}{N_f}}}\right). \end{aligned} \tag{3.35}$$

The values of the optimal bit allocation obtained by (3.34) and (3.35) could be negative and non integers, since the restriction, in the minimization problem in (3.25), on the non negativity of the value of m_k was ignored. There exist different methods to ensure that all optimal allocated bits are positive. One approach mentioned in [Bosi and Goldberg, 2002, p. 213] and [You, 2010, p. 81] is the water-filling method. The idea is to set the negative bits to zero and then to iteratively enforce that the total number of bits allocated does not exceed M. Rewriting (3.35) gives

$$\begin{aligned} m_k &= -\frac{1}{2}\log_2\left(2^{-\frac{M}{N_f}}\right) + \frac{1}{2}\log_2\left(\frac{\sigma_k^2}{\left(\prod_{i=1}^{N_f}\sigma_i^2\right)^{\frac{1}{N_f}}}\right) \\ &= \frac{1}{2}\log_2\left(\frac{\sigma_k^2}{\left(\prod_{i=1}^{N_f}\sigma_i^2\right)^{\frac{1}{N_f}} 2^{-\frac{M}{N_f}}}\right). \end{aligned} \tag{3.36}$$

From (3.36), it can be seen that

$$m_k \leq 0 \quad \text{if} \quad \sigma_k^2 \leq \left(\prod_{i=1}^{N_f}\sigma_i^2\right)^{\frac{1}{N_f}} 2^{-\frac{M}{N_f}}. \tag{3.37}$$

A positive threshold parameter λ, is now chosen such that it is less than the average MSQE, i.e. $0 \leq \lambda \leq D$. The bits are allocated such that the coefficients with variances greater than λ are described, while the others are set to zero. That is

$$m_i = \begin{cases} \frac{1}{2}\log_2\left(\frac{\sigma_i^2}{\lambda}\right) & ,\lambda < \sigma_i^2 \\ 0 & ,\lambda \geq \sigma_i^2. \end{cases} \tag{3.38}$$

The next step is to ensure that the total number of bits allocated is not higher than M. This is done by iteratively reoptimizing the sum of bits allocated for the remaining coefficients. It has been shown in [Cover and Thomas, 2006, p. 314] that the bit allocation obtained in (3.38) is optimal if the MSQE d_i for each coefficient is define such that

$$d_i = \begin{cases} \lambda & ,\lambda < \sigma_i^2 \\ \sigma_i^2 & ,\lambda \geq \sigma_i^2. \end{cases} \tag{3.39}$$

The optimal bit allocation by water-filling method is given by (3.38) and (3.39) as a function of the threshold λ, i.e.

$$m_i(\lambda) = \max\left\{\frac{1}{2}\log_2\left(\frac{\sigma_i^2}{\lambda}\right), 0\right\}, \tag{3.40}$$

with the individual MSQE given by $d_i(\lambda) = \min\left\{\lambda, \sigma_i^2\right\}$, in the case where the coefficients are normally distributed random variables.

3.2.2 Coding gain and measure of spectrum flatness

The purpose of this section is to look at the gain of quantization, while using an optimal bit allocation scheme compared to quantizing with a uniform distribution of bits. This gain is referred to as *coding gain* and corresponds to the ratio between the distortion measures derived under uniform bit allocation and optimal bit allocation. Furthermore the measure of the spectral flatness is introduced, since it is related to the coding gain as it has been shown in [Gersho and Gray, 1993], [Bosi and Goldberg, 2002] and [You, 2010].

Recall the case of high resolution optimization and assume for now that the normalized transform coefficients are identically distributed random variables. If the bits are allocated uniformly to the transform coefficients, then each coefficient will be allocated m_i bits, for $i = 1, \ldots, N_f$, corresponding to the average number of bits per coefficient $\frac{M}{N_f}$. Using (3.24) , the average MSQE will be

$$D_{\text{bit allocation}}^{\text{uniform}} = \frac{1}{N_f}\sum_{i=1}^{N_f} h\sigma_i^2 2^{-2\frac{M}{N_f}}. \tag{3.41}$$

Alternatively, using the optimal bit allocation strategy given by (3.35), the average MSQE will correspond to

$$
\begin{aligned}
D^{\text{optimal}}_{\text{bit allocation}} &= \frac{1}{N_f}\sum_{i=1}^{N_f} h\sigma_i^2 2^{-2\frac{M}{N_f}\frac{\left(\prod_{j=1}^{N_f}\sigma_j^2\right)^{\frac{1}{N_f}}}{\sigma_i^2}} \\
&= \frac{1}{N_f}\sum_{i=1}^{N_f} h2^{-2\frac{M}{N_f}}\left(\prod_{j=1}^{N_f}\sigma_j^2\right)^{\frac{1}{N_f}}.
\end{aligned}
\tag{3.42}
$$

The coding gain of using the optimal bit allocation strategy over the uniform bit allocation is thus defined as the ratio of the distortions defined in (3.41) and (3.42),

$$
\begin{aligned}
G &= \frac{\frac{1}{N_f}\sum_{i=1}^{N_f} h\sigma_i^2 2^{-2\frac{M}{N_f}}}{\frac{1}{N_f}\sum_{i=1}^{N_f} h2^{-2\frac{M}{N_f}}\left(\prod_{j=1}^{N_f}\sigma_j^2\right)^{\frac{1}{N_f}}} \\
&= \frac{\frac{h2^{-2\frac{M}{N_f}}}{N_f}\sum_{i=1}^{N_f}\sigma_i^2}{\frac{h2^{-2\frac{M}{N_f}}}{N_f}N_f\left(\prod_{j=1}^{N_f}\sigma_j^2\right)^{\frac{1}{N_f}}} \\
&= \frac{\frac{1}{N_f}\sum_{i=1}^{N_f}\sigma_i^2}{\left(\prod_{j=1}^{N_f}\sigma_j^2\right)^{\frac{1}{N_f}}},
\end{aligned}
\tag{3.43}
$$

where $\frac{1}{N_f}\sum_{i=1}^{N_f}\sigma_i^2$ and $\left(\prod_{j=1}^{N_f}\sigma_j^2\right)^{\frac{1}{N_f}}$ are respectively the arithmetic and geometric mean of the transform coefficients variances. The arithmetic mean-geometric mean inequality [You, 2010, p. 77] states that

$$
\frac{1}{N_f}\sum_{i=1}^{N_f}\sigma_i^2 \geq \left(\prod_{i=1}^{N_f}\sigma_i^2\right)^{\frac{1}{N_f}}, \tag{3.44}
$$

with equality if and only if

$$
\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_{N_f}^2.
$$

Thus the coding gain in (3.43) is such that

$$
G \geq 1. \tag{3.45}
$$

This leads to the definition of the spectrum flatness measure.

Definition 3.4 (Spectrum flatness measure)
Given a random process $X = x_1, \dots, x_{N_f}$. The spectrum flatness measure denoted γ_x is defined as the ratio between the arithmetic and geometric mean of the power

spectral density of the random process

$$\gamma_x = \frac{\left(\prod_{i=1}^{N_f} S_X(i)\right)^{\frac{1}{N_f}}}{\frac{1}{N_f}\sum_{i=1}^{N_f} S_X(i)}, \tag{3.46}$$

where S_X is the power spectral density of X. [Bosi and Goldberg, 2002, p. 211]

The spectrum flatness measure takes values between 0 and 1, where the values close to 1 indicate that the spectrum of the signal is very flat and the values near 0 indicate that the spectrum is less flat. A relation between the coding gain in (3.43) and the spectral flatness measure is given by [You, 2010, p. 85] as

$$\lim_{N_f \to \infty} G = \frac{1}{\gamma_x}. \tag{3.47}$$

The coding gain defined in (3.43) is thus asymptotically equivalent to the inverse of the spectrum flatness measure. As the spectrum flatness measure decreases, the coding gain increases for using an optimal bit allocation strategy relative to uniform bit allocation. Applying the optimal bit allocation strategy to a signal representation which is less flat, i.e. with strong peaks [Bosi and Goldberg, 2002, p. 216], will increase the coding gain. Hence it is advantageous to choose a transform that concentrates most of the signal energy in a few coefficients. The optimal bit allocation strategy derived above will permit reducing the number of bits necessary to describe the coefficients, when applied to a sparse signal representation, i.e. a signal having many coefficients close to zero. Fewer or zero bits will be allocated to the coefficients close to zero because of their low variances. This will reduce the total number of bits used to describe the coefficients in comparison to a uniform distribution of the bits through coefficients. In practice this strategy inspires the coding algorithm presented in Algorithm 3.

3.3 Lossless source coding

The information in a digital random source signal may be represented by a source code, usually a binary code with a given length in bits. In an attempt to compress the signal, one may use a code that has a smaller code length than the original in order to reduce the number of bits. The overall process involves two parts that have to be taken into consideration: The encoding part, where a code is used to describe the information in the signal, and the decoding part, where the information in the signal is recovered from the codes, without loss. The compression scheme will thus involve finding the optimal encoding scheme, meaning the one giving the shortest code length on average, while fulfilling the conditions for decodability.

Definition 3.5 (Source code)
Given a random variable X, with range $\mathcal{A}$. A source code C for X is defined as the mapping

$$\mathcal{A} \to \mathcal{D}^*, \tag{3.48}$$

where $\mathcal{A}$ denote the input or source alphabet and $\mathcal{D}^*$ is the set of finite-length strings of symbols from a D-ary alphabet. [Cover and Thomas, 2006, p. 103]

An important characteristic of a source code is the average or expected code length, which is presented in the following.

Definition 3.6 (Expected length)
Let X be a random variable with input alphabet $\mathcal{A}$ and pmf $p(X = a) = p(a)$. The expected length $L(C)$ of a source code $C(a)$ for X is defined as

$$L(C) = \sum_{a \in \mathcal{A}} p(a)l(a), \tag{3.49}$$

where $l(a)$ is the length of the codeword $C(a)$ associated with a. [Cover and Thomas, 2006, p. 104]

When assigning a code to an element, it is important to ensure that two different source-elements have different corresponding codes. This property is referred to as the nonsingularity of the code.

Definition 3.7 (Nonsingular code)
A code is said to be nonsingular if every element of $\mathcal{A}$, the range of X, maps into a different string in $\mathcal{D}^*$; that is

$$a \neq a' \Rightarrow C(a) \neq C(a'). \tag{3.50}$$

[Cover and Thomas, 2006, p. 105]

In Definition 3.5, the codeword associated with a single realisation of the random variable has been introduced. When a finite sequence of realisations is coded, then the resulting codeword is a concatenation of the singles codewords. This concatenation is called code extension.

Definition 3.8 (Code extension)
The extension C^M of a code C is the mapping from length M strings of $\mathcal{A}$ to finite-length strings of $\mathcal{D}^*$, defined by

$$C^M(a_0a_1 \dots a_{M-1}) = C(a_0)C(x_1)\dots C(a_{M-1}), \tag{3.51}$$

where $C(a_0)C(a_1)\dots C(a_{M-1})$ denotes the concatenation of the corresponding codewords. [Cover and Thomas, 2006, p. 105]

As mentioned above, one major desired property of source coding, is the ability to decode the source code without loss. The code fulfilling this condition is called uniquely decodable.

Definition 3.9 (Uniquely decodable code)
A code C is said to be uniquely decodable if its extension C^* is nonsingular. [Cover and Thomas, 2006, p. 105]

An often used uniquely decodable code is the prefix code [Gersho and Gray, 1993, p. 269] and [Cover and Thomas, 2006, pp. 105-106]. When the starting point of the prefix code is known, a codeword will be decodable as soon as the end of the codeword is reached. Thus the prefix code permits one to decode a codeword independently from the future codewords.

Definition 3.10 (Prefix code)
A code is called a prefix code or an instantaneous code if no codeword is a prefix of any other codeword. [Cover and Thomas, 2006, p. 106]

The main goal so far is to construct codes having the shortest expected length, while being able to decode them. The uniquely decodable codes ensure the decodability of the code; however the lengths of the codewords are restricted by the Kraft inequality. In this project, the codewords are chosen from a binary alphabet, thus the Kraft inequality will be restricted to binary codewords.

Theorem 3.1 (Kraft-McMillan inequality)
Given any uniquely decodable code with source alphabet $\mathcal{A} = \{a_0, \dots, a_{M-1}\}$ and codeword lengths $l_i = l(a_i)$, $i = 0, 1, \dots, M-1$. The set of codeword lengths $l_i = l(a_i)$, $i = 0, 1, \dots, M-1$ must satisfy the Kraft inequality

$$\sum_{i=0}^{M-1} 2^{-l_i} \leq 1. \tag{3.52}$$

Conversely if the set of codeword lengths satisfies the Kraft inequality, then there exists a uniquely decodable code for a source alphabet $\mathcal{A} = \{a_0, \ldots, a_{M-1}\}$, having these codeword lengths. [Cover and Thomas, 2006, p. 116]

Proof. Given a uniquely decodable code C with source alphabet $\mathcal{A} = \{a_0, \ldots, a_{M-1}\}$ and codeword lengths $l_i = l(a_i)$, $i = 0, 1, \ldots, M-1$. Let C^k be the k'th extension of the code C. Thus the length of C^k is given by

$$l(\boldsymbol{b}) = \sum_{i=0}^{k-1} l(b_i), \tag{3.53}$$

where $\boldsymbol{b} = [b_0, b_1, \ldots, b_{k-1}]$ is any length k input string. Consider the sum in (3.52) and apply the k'th power to it. That is,

$$\begin{aligned} \left(\sum_{i=0}^{M-1} 2^{-l_i}\right)^k &= \left(\sum_{a\in\mathcal{A}} 2^{-l(a)}\right)^k \\ &= \sum_{b_0\in\mathcal{A}} \sum_{b_1\in\mathcal{A}} \cdots \sum_{b_{k-1}\in\mathcal{A}} 2^{-l(b_0)} 2^{-l(b_1)} \ldots 2^{-l(b_{k-1})} \qquad (3.54) \\ &= \sum_{b_0, b_1, \ldots, b_{k-1}\in\mathcal{A}^k} 2^{-(l(b_0)+l(b_1)+\cdots+l(b_{k-1}))} \\ &= \sum_{\boldsymbol{b}\in\mathcal{A}^k} 2^{-l(\boldsymbol{b})}, \qquad (3.55) \end{aligned}$$

where $\mathcal{A}^k = \{(a_0, a_1, \ldots, a_{k-1}) | a_i \in \mathcal{A} \text{ for } i = 0, 1, \ldots, k-1\}$. The terms in (3.54) are obtained by using a summation identity and (3.55) results from (3.53). Let $l_{\max} = \max\{l_i; i = 0, 1, \ldots, M-1\}$ be the maximum codeword length and denote by $N(m)$ the total number of source sequences $\boldsymbol{b}$ mapping into codewords of length $l(\boldsymbol{b}) = m$ given by (3.53). For a total of k codewords per sequence, $m \leq kl_{\max}$. Thus (3.55) can be written as

$$\sum_{\boldsymbol{b}\in\mathcal{A}^k} 2^{-l(\boldsymbol{b})} = \sum_{m=1}^{kl_{\max}} N(m) 2^{-m} \leq \sum_{m=1}^{kl_{\max}} 2^m 2^{-m} = kl_{\max}. \tag{3.56}$$

The inequality is provided by the property of uniquely decodable codes in Definition 3.9. Precisely, all $N(m)$ source sequences are mapped into a different code of overall length m. Thus $N(m) \leq 2^m$, since there are at most 2^m binary codes of length m. It follows from (3.54) and (3.56) that

$$\begin{aligned} \left(\sum_{i=0}^{M-1} 2^{-l_i}\right)^k &\leq kl_{\max} \\ \sum_{i=0}^{M-1} 2^{-l_i} &\leq (kl_{\max})^{\frac{1}{k}} = 2^{\frac{1}{k}\log(k)+\frac{1}{k}\log(l_{\max})}, \text{ for all } k \in \mathbb{Z}. \qquad (3.57) \end{aligned}$$

Taking the limit as $k \to \infty$, results in $2^{\frac{1}{k}\log(k)+\frac{1}{k}\log(l_{\max})} \to 1$, thus

$$\sum_{i=0}^{M-1} 2^{-l_i} \leq 1, \tag{3.58}$$

which is the Kraft inequality. The second part of the proof can be found in [Cover and Thomas, 2006, p. 117] and [Gersho and Gray, 1993, pp. 265-266]. □

The overall process of the source coding scheme can now be established as an optimization problem, i.e. it is desired to find the set of codeword lengths that satisfy the Kraft inequality, while minimizing the expected codeword length (3.49).

$$\begin{aligned} \underset{l_i}{\text{minimize}} \quad & L = \sum_{i=0}^{M-1} p_i l_i \quad \text{for all } l_0, l_1, \dots, l_{M-1} \in \mathbb{Z} \\ \text{subject to} \quad & \sum_{i=0}^{M-1} 2^{-l_i} \leq 1. \end{aligned} \tag{3.59}$$

This optimization problem is a constrained minimization problem, and the Lagrange multipliers theorem is used to solve it. The integer constraint on l_i is ignored for now. The set of $\{l_i\}$ that solve the minimization problem satisfy that

$$\frac{\partial J\left(\{l_i\},\mu\right)}{\partial l_k} = 0, \quad k = 0, \dots, M-1 \tag{3.60}$$

$$\frac{\partial J\left(\{l_i\},\mu\right)}{\partial \mu} = 0, \tag{3.61}$$

$$\mu \geq 0, \tag{3.62}$$

$$-\mu \left(\sum_{j=0}^{M-1} 2^{-l_j} - 1 \right) \geq 0,$$

where μ is the Lagrange multiplier associated with the inequality constraint and $J\left(\{l_i\},\mu\right)$ is the Lagrangian given by

$$J\left(\{l_i\},\mu\right) = \sum_{h=0}^{M-1} p_h l_h + \mu \left(\sum_{j=0}^{M-1} 2^{-l_j} - 1 \right). \tag{3.63}$$

Using this Lagrangian, (3.60) is solved firstly by

$$\begin{aligned} \frac{\partial J\left(\{l_i\},\mu\right)}{\partial l_k} &= \frac{\partial}{\partial l_k} \left(\sum_{h=0}^{M-1} p_h l_h + \mu \left(\sum_{j=0}^{M-1} 2^{-l_j} - 1 \right) \right) \\ &= \sum_{h=0}^{M-1} \frac{\partial}{\partial l_k} \left(p_h l_h\right) + \mu \left(\sum_{j=0}^{M-1} \frac{\partial}{\partial l_k} \left(2^{-l_j} - 1\right) \right) \\ &= p_k - \mu 2^{-l_k} \ln(2), \end{aligned} \tag{3.64}$$

and secondly by equating (3.64) to zero and solving for l_k,

$$\begin{aligned} p_k - \mu 2^{-l_k} &= 0 \\ 2^{-l_k} &= \frac{p_k}{\mu \ln(2)}. \end{aligned} \tag{3.65}$$

Identically (3.61) is solved by finding firstly the partial derivative associated with μ,

$$\begin{aligned} \frac{\partial J\left(\{l_i\},\mu\right)}{\partial \mu} &= \frac{\partial}{\partial \mu}\left(\sum_{h=0}^{M-1} p_h l_h + \mu\left(\sum_{j=0}^{M-1} 2^{-l_j} - 1\right)\right) \\ &= \sum_{j=0}^{M-1} 2^{-l_j} - 1. \end{aligned} \tag{3.66}$$

Next, (3.65) is substituted in (3.66) and equated to zero,

$$\begin{aligned} \sum_{i=0}^{M-1} \frac{p_i}{\mu \ln(2)} - 1 = 0 \\ \mu = \frac{1}{\ln(2)}. \end{aligned} \tag{3.67}$$

Substituting (3.67) in (3.65) gives

$$2^{-l_i} = p_i. \tag{3.68}$$

Thus the optimal codeword length under the condition that l_i is not necessarily an integer, is

$$l_i^* = -\log_2(p_i). \tag{3.69}$$

The expected code length given by (3.69) is,

$$L^* = \sum_{i=0}^{M-1} p_i l_i^* = -\sum_{i=0}^{M-1} p_i \log_2(p_i) = H(X), \tag{3.70}$$

where $H(X)$ is called the entropy of the random variable X described by the pmf $p_i = p(X = a_i)$ [Cover and Thomas, 2006, pp. 110-111] and [Gersho and Gray, 1993, pp. 267 - 268].

Definition 3.11 (Entropy)
The entropy $H(X)$ of a discrete random variable X with range $\mathcal{A}$ and pmf $p(X = a) = p(a)$, is defined by

$$H_b(X) = -\sum_{a\in\mathcal{A}} p(a)\log_b(p(a)) = \mathrm{E}\left[\log_b\left(\frac{1}{p(X)}\right)\right]. \tag{3.71}$$

[Cover and Thomas, 2006, p. 14]

The entropy $H(X)$ of a random variable X is seen as a measure of the uncertainty of the random variable X. Thus the entropy is considered as the amount of information necessary in order to describe the random variable on the average. Furthermore, there is the relative entropy which corresponds to the distance between two probability distributions [Cover and Thomas, 2006, p. 19].

Definition 3.12 (Relative entropy)
Let X be a discrete random variable with range $\mathcal{A}$. The relative entropy between two probability mass functions $p(a)$ and $q(a)$ is defined as

$$D(p||q) = -\sum_{a\in\mathcal{A}} p(a)\log_b\left(\frac{p(a)}{q(a)}\right) = \mathrm{E}\left[\log_b\left(\frac{p(X)}{q(X)}\right)\right]. \tag{3.72}$$

[Cover and Thomas, 2006, p. 19]

The relative entropy is seen as a measure of the inefficiency of assuming that the probability distribution is q, when the true probability distribution is p [Cover and Thomas, 2006, p. 19]. An important property about relative entropy is stated in the following lemma.

Lemma 3.2 (Information inequality)
Let $p(a), q(a), a \in \mathcal{A}$ be two probability mass functions. Then

$$D(p||q) \geq 0, \tag{3.73}$$

with equality if and only if $p(a) = q(a)$ for all $a \in \mathcal{A}$. [Cover and Thomas, 2006, p. 28]

The fact that the codewords length must be an integer in the minimization problem means that (3.69) cannot always be used as the optimal codeword length. However it has been shown that the expected length of the shortest description of the source signal cannot be smaller than the entropy [Cover and Thomas, 2006, p. 10]. This is stated in the following theorem.

Theorem 3.2
The expected length of a uniquely decodable binary code for a random variable X with input alphabet $\mathcal{A} = \{a_0, \ldots, a_{M-1}\}$ and marginal pmf $p_i = p(X = a_i)$, cannot be smaller than the entropy $H(X)$; that is,

$$L \geq H(X), \tag{3.74}$$

with equality if and only if

$$2^{-l_i} = p_i, \text{ for all } a \in \mathcal{A}.$$

[Gersho and Gray, 1993, p. 268]

Proof. From (3.49) and (3.71), the difference between the expected length and the entropy is given by

$$\begin{aligned}
L - H(X) &= \sum_{i=0}^{M-1} p_i l_i + \sum_{a\in\mathcal{A}} p_i \log_2(p_i) \\
&= -\sum_{i=0}^{M-1} p_i \log_2(2^{-l_i}) + \sum_{i=0}^{M-1} p_i \log_2(p_i) \\
&= \sum_{i=0}^{M-1} p_i \log_2(p_i) - p_i \log_2(2^{-l_i}) \\
&= \sum_{i=0}^{M-1} p_i \Big(\log_2(p_i) - \log_2(2^{-l_i}) + \log_2(s) - \log_2(s)\Big) && (3.75) \\
&= \sum_{i=0}^{M-1} p_i \left(\log_2\left(\frac{p_i}{q_i}\right) - \log_2(s)\right) \\
&= \sum_{i=0}^{M-1} p_i \log_2\left(\frac{p_i}{q_i}\right) + p_i \log_2\left(\frac{1}{s}\right) \\
&= D(p||q) + p_i \log_2\left(\frac{1}{s}\right), && (3.76)
\end{aligned}$$

where

$$s = \sum_{j=0}^{M-1} 2^{-l_j} \text{ and } q_i = \frac{2^{-l_i}}{\sum_{j=0}^{M-1} 2^{-l_j}}.$$

From Lemma 3.2, $D(p||q) \geq 0$ with equality if and only if $p_i = q_i$ and from Kraft inequality in (3.52), $s \leq 1$ with equality if $p_i = 2^{-l_i}$. Thus using (3.76), the following result holds

$$L - H(X) \geq 0, \tag{3.77}$$

with equality if and only if $p_i = 2^{-l_i}$. □

The entropy is the global minimum for the expected length and thus the codes that have an expected length near the entropy can be qualified as good codes [Gersho and Gray, 1993, p. 268].

The following theorem shows that there exist uniquely decodable codes that have an expected length very close to the entropy.

Theorem 3.3 (Noiseless coding theorem)
Let $l_0^*, l_1^*, \ldots, l_{M-1}^*$ be optimal codeword lengths for a source distribution $p(X)$ with input alphabet $\mathcal{A} = \{a_0, \ldots, a_{M-1}\}$ and a binary alphabet, and let L^* be the associated expected length of an optimal code. Then

$$H(X) \leq L^* < H(X) + 1. \tag{3.78}$$

[Cover and Thomas, 2006, p. 113]

Proof. Recall the minimization problem in (3.59) leading to the optimal choice of codeword length given by (3.69) as

$$l_i^* = \log_2\left(\frac{1}{p_i}\right). \tag{3.79}$$

Since the l_i^* satisfying (3.79) are not necessarily integers, the codewords length l_i are chosen such that

$$l_i = \left\lceil \log_2\left(\frac{1}{p_i}\right) \right\rceil. \tag{3.80}$$

Furthermore

$$\sum_{i=0}^{M-1} 2^{-\left\lceil \log_2\left(\frac{1}{p_i}\right)\right\rceil} \leq \sum_{i=0}^{M-1} 2^{-\log_2\left(\frac{1}{p_i}\right)} = \sum_{i=0}^{M-1} p_i = 1. \tag{3.81}$$

Thus the codeword length l_i satisfies the Kraft inequality. However since l_i^* is smaller than l_i,

$$\log_2\left(\frac{1}{p_i}\right) \leq l_i < \log_2\left(\frac{1}{p_i}\right) + 1$$

$$\sum_{i=0}^{M-1} p_i \log_2\left(\frac{1}{p_i}\right) \leq \sum_{i=0}^{M-1} p_i l_i < \sum_{i=0}^{M-1} p_i \log_2\left(\frac{1}{p_i}\right) + \sum_{i=0}^{M-1} p_i \tag{3.82}$$

$$H(X) \leq L < H(X) + 1. \tag{3.83}$$

Since L^* is the optimal expected code length,

$$L^* < L,$$

and applying Theorem 3.2,

$$H(X) \leq L^* < H(X) + 1. \qquad \square$$

The minimum expected length necessary for a uniquely decodable code to describe a random variable X is within 1 bit of the entropy of the random variable.

In Chapter 4 lossless coding will be combined with quantization and bit allocation in a lossy audio coder. The lossless part of this coder will be based on the Golomb coder, which is an entropy coder for exponentially decaying distributions on positive integers [Spanias et al., 2006, p. 82]. In this case the fact that zero occurs with a much higher probability than one is used.

3.4 Audio coding

The purpose of audio coding is to obtain a more compact digital representation of an audio signal, which uses fewer bits to describe the signal. There exist various methods to compress an audio signal; one method is transform coding, introduced in Section 3.4.1, which consists of firstly representing the original or source audio signal in another domain using a transform. Secondly quantization and a lossless coding scheme are applied to the transformed audio signal to obtain a compressed bitstream representation. An important characteristic of the compressed audio signal is that the audio signal should be perceptually close to the original audio. A perceptual model is thus often included in audio coding [You, 2010, p. 73]. In this section, the application of transform coding to audio signals will be presented, based on the properties mentioned in Chapter 2 and Section 3.2. Finally the concept of perceptual models for audio signals will be briefly touched upon.

3.4.1 Transform coding

In transform coding, an audio signal $\boldsymbol{x} = \begin{bmatrix}\boldsymbol{x}[0] & \boldsymbol{x}[1] & \dots & \boldsymbol{x}[N-1]\end{bmatrix}^\top \in \mathbb{R}^N$ is first divided into blocks each having a fixed number of samples N_f, also called block size. Applying an orthogonal transform $\boldsymbol{T} \in \mathbb{R}^{N_f \times N_f}$ to each block, gives a new audio signal representation $\boldsymbol{y} \in R^N$ with each block corresponding to a signal-vector given by

$$\boldsymbol{y}_k = \boldsymbol{T}\boldsymbol{x}_k, \tag{3.84}$$

where $\boldsymbol{x}_k$ is the signal vector corresponding to the original block. Since, it is desired to obtain a more compact representation of $\boldsymbol{x}$, the choice of the transform should be in accordance with the optimal bit allocation presented in Section 3.2. This means that the optimal choice of a transform is the one giving the best coding gain, when using an optimal bit allocation strategy. As shown in Section 3.2.2, the desired transformed signal should have most coefficients equal to or near zero, i.e. the optimal transform is the one having the best energy compaction.

Recall the coding gain of using the optimal bit allocation strategy relative to the uniform bit allocation in (3.43) was given by

$$G = \frac{\frac{1}{N_f}\sum_{i=1}^{N_f}\sigma_i^2}{\left(\prod_{i=1}^{N_f}\sigma_i^2\right)^{\frac{1}{N_f}}}. \tag{3.85}$$

Since $\boldsymbol{T}$ is an orthogonal transform, it has the energy conservation property [You, 2010, p. 75]. This means that the arithmetic mean of the variances of the transform coefficients, given by the numerator of (3.85), is constant for a given signal [You, 2010, pp. 79-82]. In Section 3.2.2, it is shown that a better coding gain was achieved by a transform that concentrates most of the signal energy in a few coefficients. In that case the product of the variances of the transform coefficients will be smaller, since a lot of transform coefficients will have variances near zero. The best coding gain is thus obtained when the denominator of (3.85), i.e. the geometric mean of the variances of the transform coefficients, is as small as possible. In other words, the maximal coding gain is obtained by solving the following minimization problem:

$$\underset{\sigma_i^2}{\text{minimize}} \quad \left(\prod_{i=1}^{N_f} \sigma_i^2 \right)^{\frac{1}{N_f}}. \tag{3.86}$$

Using (3.84), it follows that

$$\boldsymbol{y}_k \boldsymbol{y}_k^\top = \boldsymbol{T} \boldsymbol{x}_k \boldsymbol{x}_k^\top \boldsymbol{T}^\top. \tag{3.87}$$

The covariance matrix of the transform coefficients $\boldsymbol{R}_{yy}$ is obtained by applying the expected value operator on (3.87). That is

$$\boldsymbol{R}_{yy} = \boldsymbol{T} \boldsymbol{R}_{xx} \boldsymbol{T}^\top, \tag{3.88}$$

where $\boldsymbol{R}_{yy} = \mathrm{E}\left[\boldsymbol{y}_k \boldsymbol{y}_k^\top\right]$ and $\boldsymbol{R}_{xx} = \mathrm{E}\left[\boldsymbol{x}_k \boldsymbol{x}_k^\top\right]$ is the covariance matrix of the audio signal. The covariance matrix of the transformed audio signal is a symmetric matrix having the diagonal entries corresponding to variances of the different transform coefficients [You, 2010, p. 83], thus the variance of the ith transform coefficient is given by

$$\sigma_i^2 = \mathrm{E}\left[\boldsymbol{y}_k[i] \boldsymbol{y}_k[i]\right] = [\boldsymbol{R}_{yy}]_{ii}, \quad i = 1, \dots N_f, \tag{3.89}$$

where $\boldsymbol{y}_k[i]$ is the i'th component of the transformed block $\boldsymbol{y}_k$ and $[\boldsymbol{R}_{yy}]_{ii}$ is the i'th diagonal element of the transform covariance matrix. Using (3.89), the problem in (3.86) can be written as

$$\underset{\boldsymbol{T}}{\text{minimize}} \quad \left(\prod_{i=1}^{N_f} [\boldsymbol{R}_{yy}]_{ii} \right)^{\frac{1}{N_f}}. \tag{3.90}$$

Assume for now that the covariance matrix of the audio signal, $\boldsymbol{R}_{xx}$ is positive definite. It has been shown in [You, 2010, p. 83] that

$$\prod_{i=1}^{N_f} [\boldsymbol{R}_{yy}]_{ii} \geq \det(\boldsymbol{R}_{xx}), \tag{3.91}$$

with equality if and only if $\boldsymbol{R}_{yy}$ is a diagonal matrix. In order to minimize the geometric mean of the variances, one might find a transform $\boldsymbol{T}$ that makes $\boldsymbol{R}_{yy}$ a diagonal matrix. From [You, 2010, p. 84], it is shown that the transform having those properties is the Karhunen-Loeve Transform (KLT), which has an orthonormal matrix with rows corresponding to the eigenvectors of the covariance matrix of the audio signal $\boldsymbol{R}_{xx}$. When applying the KLT to the audio signal, the obtained transform coefficients will be equal to the eigenvalues of $\boldsymbol{R}_{xx}$. Thus the KLT maximizes the coding gain G in (3.85).

The KLT is optimal in terms of the optimal bit allocation strategy. However, in practice the KLT is not usually used in transform coding. The KLT is signal dependent.

The statistics of audio signals are for the most part time-dependent, and will thus require a real-time computation of the covariance matrix, eigenvectors and eigenvalues when using the KLT. In general because of this signal dependency of the KLT, it will be necessary to send information on the original signal to the decoder in order to reconstruct the signal, which will require additional bits. This is not suitable in practice when dealing with audio coding [You, 2010, p. 85]. Instead of using the KLT, one may use other orthogonal transforms that achieve a coding gain near the maximal coding gain achieved by the KLT. The class of orthogonal sinusoidal transforms is known to achieve a coding gain close to the maximum [You, 2010, p. 86]. One state of the art used transform is the discrete cosine transform (DCT), see Section 4.1, which is not data dependent and which can achieve a coding gain close to the maximum, when the block size goes to the infinity [You, 2010, p. 86].

3.4.2 Perceptual model

A lossy compression applied to an audio signal is based on reducing the number of bits used to describe the audio signal by using quantization and lossless source coding. The quantization process consists of reducing the number of bits used to describe the audio signal by removing the information in the signal, which is perceptually irrelevant [You, 2010, p. 173]. It is thus needed to apply a perceptual model to the audio signal in order to find which components of the audio signal are perceptually irrelevant, i.e. the components of the audio signal which are inaudible for the human ear. A perceptual model is based on the response of the human auditory system to sounds. This falls under the theory of psychoacoustics [Bosi and Goldberg, 2002, p. 149]. In this section only some aspects of psychoacoustics will be presented.

Since the quantization process produces quantization noise, the idea is to shape quantization in a way that makes this noise inaudible. The perceptually irrelevant parts of the audio signal with quantization noise. The quantization noise is not perceived by the human ear if its power is less than the hearing threshold [You, 2010, p. 173]. The power of the sound can be described by the sound pressure level, which is defined as follows.

Definition 3.13 (Sound pressure level)
The sound pressure level (SPL) is the ratio between the sound pressure of a sound relative to a reference sound pressure.

$$\mathrm{SPL} = 20\log_{10}\left(\frac{p}{p_0}\right)\mathrm{dB}, \tag{3.92}$$

where p is the variation of the atmosphere pressure in time called sound pressure and p_0 is a reference sound pressure of 2×10^{-5} Pa corresponding to the best hearing sensitivity of an average listener for tone frequencies around 1kHz. [You, 2010, p. 174]

The hearing threshold is thus defined as follows.

Definition 3.14 (Hearing threshold)
The hearing threshold is defined as the lowest SPL of a pure tone that can be heard, by an average listener with normal hearing capability, in an absolutely quiet environment. [You, 2010, p. 175]

The part of the audio signal having an SPL below the hearing threshold may be assumed to be perceptually irrelevant. The hearing threshold varies with the frequency of the used tone. In [Bosi and Goldberg, 2002, pp. 152-156] and [You, 2010, pp. 174-175], the analysis of frequencies against hearing threshold shows the frequency levels for which the human ear is more and less sensitive. The idea of using hearing threshold in coding an audio signal, is to identify the frequency components of the audio signal, which are inaudible for the human ear. Those will be quantized by allocating them few bits with the advantage that the quantization noise will not be heard in the coded audio signal. However it has been shown in [You, 2010] that this method has some disadvantages. Instead of directly using the hearing threshold, one may use the fact that the hearing threshold of a pure tone increases significantly in the presence of louder sounds in the environment because of the phenomenon of auditory masking [Bosi and Goldberg, 2002, p. 179] and [You, 2010, p. 173]. Masking is defined formally in the following.

Definition 3.15 (Masking)
Masking is a phenomenon where loud sounds can cause a weak sound to be less audible or totally inaudible, when the power of the weak sound is under a certain frequency dependent threshold jointly determined by the characteristics of both sounds. [You, 2010, p.173]

There exist different types of masking which can be used in audio coding, [Bosi and Goldberg, 2002, pp. 156-164] and [You, 2010, pp. 186-198]. One type of masking used in state of the art audio coding is simultaneous masking, where large sound pressure levels of the quantization noise localized in frequencies of the signal become inaudible, because the quantization noise is masked by frequency components of the signal that occur at the same time as the quantized frequency components [You, 2010, p. 173]. The general idea with building a perceptual model of an audio signal, is to combine the hearing threshold and masking, e.g. simultaneous masking, in order to find an estimate of a so-called masking threshold. The audio signal components having a power below the masking threshold will according to the model be inaudible. In audio coding, the quantization process will thus be designed such that all the quantization noises have powers that are below the masking threshold [You, 2010, p. 173]. The masking threshold can for example be included in the bit allocation problem presented in Section 3.2 to obtain a perceptual bit allocation, [Bosi and Goldberg, 2002, pp. 216-218] and [You, 2010, pp. 194-195].

Building a perceptual model is based on combining the properties of the used audio signal representation with a psychoacoustic approach. Psychoacoustic models are usually empirically determined. Applying a psychoacoustic model with the convolution framelet introduced in Chapter 2 is not explored further in this project due to time restrictions.

3.5 Bitplane

When encoding a signal, a practical way of showing the bit representation is with bitplanes. The basic idea is to construct a vector for each coefficient containing the specific binary expansion representing the coefficient. This means that a coefficient matrix $\boldsymbol{C}$ is represented with a 3D bitarray. An example slice of a bitarray can be seen on Figure 3.2. The slice contains several bitplanes as well as a plane containing information concerning the sign of the coefficient.

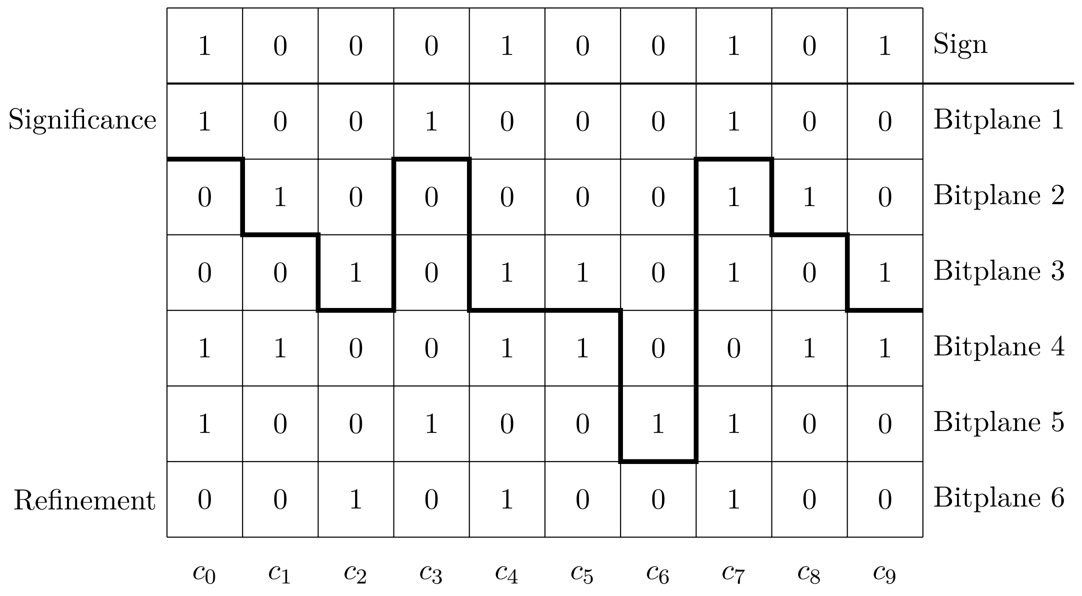
1	0	0	0	1	0	0	1	0	1	Sign
1	0	0	1	0	0	0	1	0	0	Bitplane 1
0	1	0	0	0	0	0	1	1	0	Bitplane 2
0	0	1	0	1	1	0	1	0	1	Bitplane 3
1	1	0	0	1	1	0	0	1	1	Bitplane 4
1	0	0	1	0	0	1	1	0	0	Bitplane 5
0	0	1	0	1	0	0	1	0	0	Bitplane 6
c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	

Figure 3.2: Example of bitplanes for 10 coefficients from $\boldsymbol{C}$.

An important aspect of the bitplanes is the order of significance. The most significant bitplane is the first, i.e. Bitplane 1, and as the order of bitplanes increase the significance decreases. This means that it is possible to compress the signal by truncating the bitplanes starting with the highest order bitplane. This of course introduces distortion as the signal loses refinement.

Based on [Ravelli et al., 2008, p. 6], the j'th most significant bit of the coefficient c_i is given by $b_{i,j} = mod(floor(\frac{abs(c_i)\times 2^j}{A}))$, where $A = \max(abs(c_i))$. The vector of bits of same significance j is the j'th bitplane $\boldsymbol{B}_j = \{b_{i,j}\}$. A coefficient c_i is said to be significant at level j if $abs(c_i)/A \geq 2^{-j}$.

When encoding bitplanes the process is divided in two passes, firstly a significance pass and secondly a refinement pass. The significance pass will as the name implies encode the most significant part of the bitplanes. It notes the bitplane, where the first 1 is encountered for each coefficient, when searching from the most significant bitplane down to the least significant. This generally results in a higher occurrence of zeros in the significance pass than ones. The refinement pass includes the remaining bits, which were not included in the significance pass. This results in a more even distribution of zeroes and ones in the refinement pass. Generally, if the spectrum flatness measure, Definition 3.4, is close to zero, the occurrence of zeros in the significance pass is high, which is ideal. Figure 3.2 is modified to include a line, which separates the bits in the significance pass and the bits in the refinement pass, this results in Figure 3.3.



	1	0	0	0	1	0	0	1	0	1	Sign
Significance	1	0	0	1	0	0	0	1	0	0	Bitplane 1
	0	1	0	0	0	0	0	1	1	0	Bitplane 2
	0	0	1	0	1	1	0	1	0	1	Bitplane 3
	1	1	0	0	1	1	0	0	1	1	Bitplane 4
	1	0	0	1	0	0	1	1	0	0	Bitplane 5
Refinement	0	0	1	0	1	0	0	1	0	0	Bitplane 6
	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	

Figure 3.3: Example of bitplanes with the thick line representing the divide between the significance pass (above the line) and the refinement pass (below the line).

The passes are then coded separately starting with the significance pass. An example of a bitplane coder, which exploits the differences between the significance pass and refinement pass, is given in Section 4.3. It is possible during the encoding to truncated the refinement pass due to for example a constraint on the bit rate, this is equivalent to a further quantization of the signal.

4 | Practical Methods

This chapter details the algorithms used in the project, and how they are applied. The first part of this chapter introduces some of the methods used in the simulations, including the discrete cosine transform, orthogonal matching pursuit and a bitplane coding algorithm. Beginning from Section 4.5 simulation descriptions are provided. Simulation results and observations will be collected at the end of the chapter in Section 4.6.

4.1 Discrete cosine transform

The Discrete Cosine Transform [DCT] will be the primarily used orthonormal transform for constructing the convolutional framelets in the simulations. The DCT-II, read as type two DCT, is defined here.

Definition 4.1 (DCT-II)
Let $\boldsymbol{x}$ be a vector in $\mathbb{R}^N$. The DCT-II of $\boldsymbol{x}$ is defined as

$$\boldsymbol{X}_{ct}[k] = C[k] \sum_{n=0}^{N-1} \boldsymbol{x}[n] \cos\left[\frac{\pi}{2N}(2n+1)\,k\right], \quad k = 0, 1, \ldots, N-1, \tag{4.1}$$

where

$$C[k] = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } k = 0 \\ \sqrt{\frac{2}{N}} & \text{for } k = 1, 2, \ldots, N-1. \end{cases} \tag{4.2}$$

[Sundararajan, 2001, p. 305]

The DCT-II is the most commonly used type of DCT. It is distinguished by its boundary condition, which is symmetric. This property means that it does not give rise to discontinuities at its boundaries, resulting in better energy compaction [You, 2010, p. 88.]. The DCT-III, which is the inverse of the DCT-II is stated here.

Proposition 4.1 (Inverse DCT-II)
Given a vector $\boldsymbol{x}$ in $\mathbb{R}^N$, the inverse DCT-II is

$$\boldsymbol{x}[n] = \sum_{k=0}^{N-1} C[k]\boldsymbol{X}_{ct}[k] \cos\left[\frac{\pi}{2N}(2n+1)k\right], \quad n = 0, 1, \dots, N-1, \tag{4.3}$$

where $C[k]$ is as defined in (4.2). [Sundararajan, 2001, p. 305]

It is shown in [Sundararajan, 2001, pp. 305-309.] that the DCT-II has an efficient implementation based on the fast Fourier transform. The DCT-II from Definition 4.1, can be rewritten to the expression,

$$\boldsymbol{X}_{ct}[k] = C[k] \sum_{n=0}^{N-1} \boldsymbol{y}[n] \cos\left[\frac{2\pi}{N}\left(n + \frac{1}{4}\right)k\right], \tag{4.4}$$

where $\boldsymbol{y}[n]$ is defined as follows,

$$\boldsymbol{y}[n] = \begin{cases} \boldsymbol{x}[2n] & \text{if } n = 0, 1, \dots, \frac{N}{2} - 1 \\ \boldsymbol{x}[2N - 2n - 1] & \text{if } n = \frac{N}{2}, \frac{N}{2} + 1, \dots, N - 1. \end{cases} \tag{4.5}$$

Using Eulers formula, $\exp(ix) = \cos(x) + i\sin(x)$, and the fact that $\boldsymbol{y}$ is real-valued, the cosine in (4.4) can be rewritten as the real part of a complex exponential,

$$\begin{aligned} \boldsymbol{X}_{ct}[k] &= C[k]\mathrm{Re}\left\{\sum_{n=0}^{N-1} \boldsymbol{y}[n] e^{-i\frac{2\pi}{N}\left(n+\frac{1}{4}\right)k}\right\} \\ &= C[k]\mathrm{Re}\left\{e^{-i\frac{2\pi k}{4N}} \sum_{n=0}^{N-1} \boldsymbol{y}[n] e^{-i\frac{2\pi}{N}nk}\right\}. \end{aligned} \tag{4.6}$$

In (4.6), the discrete Fourier transform of $\boldsymbol{y}$ is recognized, and can be computed via the FFT, to efficiently compute the DCT.

4.2 Matching pursuit

Recall the tight frame, $\boldsymbol{\Psi}$, constructed as a concatenation of convolution framelets discussed in Section 2.3. The canonical coefficients of this frame,

$$\boldsymbol{c} = \boldsymbol{\Psi}^\top \boldsymbol{f}, \tag{4.7}$$

are very redundant. Every subset of $\boldsymbol{c}$ corresponding to one of the convolution framelets from the concatenation, is sufficient to recover $\boldsymbol{f}$, and even the individual framelets may be redundant. Thus, with the desire of a sparse set of coefficients in mind, non-canonical sets of frame coefficients are of interest. Ideally $\boldsymbol{f}$ should be described completely by as few non-zero coefficients as possible. This is, however, a difficult problem to solve. As explained in [Foucart and Rauhut, 2013, pp. 53-56], the problem is NP-hard.

Matching pursuit is an algorithm, which can find a sub-optimal sparse solution. Matching pursuit has been demonstrated to work well, despite its sub-optimality, see for instance [Mallat and Zhang, 1993] and [Ravelli et al., 2008, p. 3]. The matching pursuit algorithm, detailed in Algorithm 1 below, works by iteratively selecting the coefficient, which accounts for the most residual energy, and then updating the residual.

Algorithm 1 Matching pursuit

```
input: $\boldsymbol{f}$, $\boldsymbol{\Psi}$, $\epsilon_{dB}$
$\boldsymbol{r} = \boldsymbol{f}$
$\boldsymbol{c}_{opt} = \boldsymbol{0}$
repeat
    $i_{opt} = \underset{i}{\operatorname{argmax}} |\langle \boldsymbol{r}, \boldsymbol{\psi}_i \rangle|$
    $c = \langle \boldsymbol{r}, \boldsymbol{\psi}_{i_{opt}} \rangle$
    $\boldsymbol{r} = \boldsymbol{r} - c\boldsymbol{\psi}_{i_{opt}}$
    $\boldsymbol{c}_{opt}[i_{opt}] = \boldsymbol{c}_{opt}[i_{opt}] + c$
until $10 \log_{10} \left( \frac{P(\boldsymbol{r})}{P(\boldsymbol{f})} \right) \leq \epsilon_{dB}$
output: $\boldsymbol{c}_{opt}$
```

Here, $P(\boldsymbol{f})$ denotes the average power of $\boldsymbol{f}$. Since matching pursuit only updates one coefficient each iteration, any distortion introduced by a given coefficient, can only be corrected by future coefficients.

4.2.1 Orthogonal matching pursuit

A more advanced version of matching pursuit, known as orthogonal matching pursuit [Foucart and Rauhut, 2013, p. 65], is available. Orthogonal matching pursuit differs from matching pursuit in the residual update step. Where matching pursuit updates the residual with respect to a single element of $\boldsymbol{\Psi}$ each iteration, orthogonal matching pursuit updates with respect to all elements selected so far. This is accomplished by the orthogonal projection of $\boldsymbol{f}$ onto the orthogonal complement of the span of selected elements. Let $\boldsymbol{\Psi}_{[S]}$ denote the restriction of $\boldsymbol{\Psi}$ to the columns indexed by the set S.

Algorithm 2 Orthogonal matching pursuit

```
input: $\boldsymbol{f}$, $\boldsymbol{\Psi}$, $\epsilon_{dB}$
$\boldsymbol{r} = \boldsymbol{f}$
$S = \emptyset$
$\boldsymbol{c}_{opt} = \boldsymbol{0}$
repeat
    $i_{opt} = \underset{i}{\operatorname{argmax}} \, |\langle \boldsymbol{r}, \boldsymbol{\psi}_i \rangle|$
    $S = S \cup i_{opt}$
    $\boldsymbol{P} = \boldsymbol{\Psi}_{[S]} \left( \boldsymbol{\Psi}_{[S]}^\top \boldsymbol{\Psi}_{[S]} \right)^{-1} \boldsymbol{\Psi}_{[S]}^\top$
    $\boldsymbol{r} = \boldsymbol{f} - \boldsymbol{P}\boldsymbol{f}$
until $10 \log_{10} \left( \frac{P(\boldsymbol{r})}{P(\boldsymbol{f})} \right) \leq \epsilon_{dB}$
$\boldsymbol{c}_{opt[S]} = \left( \boldsymbol{\Psi}_{[S]}^\top \boldsymbol{\Psi}_{[S]} \right)^{-1} \boldsymbol{\Psi}_{[S]}^\top \boldsymbol{f}$
output: $\boldsymbol{c}_{opt}$
```

The iterative expansion of the span of $\boldsymbol{\Psi}_{[S]}$, means that much of the computational effort implied by Algorithm 2, i.e. computing the orthogonal projection matrix $\boldsymbol{P}$ and subsequently the residual $\boldsymbol{r}$, can be saved. First of all, using the Gram-Schmidt procedure to orthonormalize each new element of $\boldsymbol{\Psi}_{[S]}$, one can iteratively construct and update an orthonormal basis, $\boldsymbol{Q}_S$, for the span of $\boldsymbol{\Psi}_{[S]}$, reducing the problem of computing $\boldsymbol{P}$ to

$$\boldsymbol{P} = \boldsymbol{Q}_S \boldsymbol{Q}_S^\top. \tag{4.8}$$

The Gram-Schmidt procedure is detailed in [Trefethen and Bau III, 1997, pp. 56-58]. Furthermore, as a result of the orthogonality of $\boldsymbol{Q}_S$, it is only necessary to project $\boldsymbol{f}$ onto the newest element of $\boldsymbol{Q}_S$ and add the result to the previous projection. The Gram-Schmidt orthogonalization step required in each iteration can be stated as follows,

$$\boldsymbol{q}_{i_{opt}} = \boldsymbol{\psi}_{i_{opt}} - \sum_{j=0}^{i-1} \frac{\langle \boldsymbol{q}_j, \boldsymbol{\psi}_{i_{opt}} \rangle}{\langle \boldsymbol{q}_j, \boldsymbol{q}_j \rangle} \boldsymbol{q}_j. \tag{4.9}$$

This orthogonalization step makes the orthogonal matching pursuit more computationally demanding than matching pursuit, since the orthogonalization requires an increasing number of computations each iteration. Orthogonal matching pursuit will however be demonstrated to give more sparse solutions. This should be clear, since orthogonal matching pursuit reoptimizes all coefficients each iteration, they can correct more distortion per iteration.

Matching and orthogonal matching pursuit will be used to find sparse frame coefficients for audio signals in the practical simulations.

4.3 Bitplane coder

The coder used in this project is based on the bitplane coder in [Ravelli et al., 2008, Algo. 3], which is a run-length coder called a Golomb-coder. Remembering the introduction to bitplane coding in Section 3.5, the encoding is separated into a significance pass, and a refinement pass. The significance pass of this coder uses an adaptive run length, to efficiently code long runs of zeros; remember that a sparse set of coefficients is found using OMP, which is why long runs of zeros are expected. The refinement bits are expected to be close to a uniform distribution between ones and zeros, so they are simply appended to the end of each significance pass.

Algorithm 3 Bitplane coder

```
input: $\boldsymbol{c}$, $q$
code maximum amplitude $A = \max_i(\text{abs}(\boldsymbol{c}_i))$
$\boldsymbol{z}_i = 0$ for all $i$
$j = 0$
repeat
    $b_i = \text{mod}(\lfloor \text{abs}(\boldsymbol{c}_i) 2^j / A \rfloor)$ for all $i$
    $S = S \cup \{b_i | \boldsymbol{z}_i = 0\}$
    $R = R \cup \{b_i | \boldsymbol{z}_i = 1\}$
    $\boldsymbol{z}_i = 1$ for all $i$ such that $b_i \in S$ and $b_i = 1$
    $j = j + 1$
until $j = q$
$k = 2$
repeat
    if sequence of $2^k$ zeros in $S$ then
        write a zero to the bit stream
        $k = k + 1$
    else
        write a one to the bit stream
        write $k$ bits describing the number of zeros before the one
        write the sign bit of the corresponding coefficient
        $k = k - 1$
    move to the next bits in $S$
until the end of $S$
write $R$ to the bit stream
output: the bit stream
```

The input q in Algorithm 3 is the quantization level, i.e. how many bitplanes should be coded. The vector $\boldsymbol{z}$ keeps track of which coefficients are in significance and which are in refinement. The variable k ensures that the coder looks for runs of zeros of incrementing powers of 2; decrementing if a one appears. The coder is thus adaptive, and spends fewer and fewer bits to code zeros if no ones appear, at the cost of making ones more expensive to code once they do appear.

Consider the optimal bit allocation presented in Section 3.2. This coder does not use that exact bit allocation, since the variances of coefficients are not considered. However, it does draw on some of the ideas behind optimal bit allocation. Remember that optimal bit allocation is dependent on the spectrum flatness of the coefficients, achieving a better coding gain when most bits can be allocated to a few large coefficients. This coder takes a sparse set of coefficients as input, which supports that there is a coding gain to be had. This coder only compresses significance bits, and large coefficients will have a larger ratio of refinement bits to significance bits, than small coefficients. This means that large coefficients will be represented by more bits in the coded bit stream than small coefficients. This is in line with the optimal bit allocation, which allocates more bits to coefficients with larger variance.

4.3.1 Coefficient interleaving

The concatenated convolution framelets used in simulations, will be constructed from DCT and identity bases with varying patch sizes. This effectively means that the coefficients will consist of concatenated time-frequency spectra, with variable time-frequency resolution. This is comparable to [Ravelli et al., 2008], in which a union of modified discrete cosine transforms, with variable window lengths are used. In [Ravelli et al., 2008, Section III.] an interleaving scheme is introduced, which groups coefficients by their relative closeness in the time-frequency domain. Effectively this combines the individual spectra into one. The interleaving scheme assumes that each time-frequency spectrum has 2 times the window length of the previous, and can be described as follows.

Let M denote the number of framelets concatenated, P_m denote the number of patches in the m'th framelet and l_m denote the patch size of the m'th framelet. Define the recursive function,

$$r(p,m) = \begin{cases} p, & \text{if } m = M-1. \\ r\left(\frac{p}{2}, m+1\right), & \text{if } m < M-1 \text{ and } p \text{ is even.} \\ r\left(\frac{p-1}{2}, m+1\right) + P_{m+1}, & \text{if } m < M-1 \text{ and } p \text{ is odd.} \end{cases} \tag{4.10}$$

Using r, the coefficients $\boldsymbol{c}$ are mapped to a vector $\boldsymbol{v}$ as such,

$$\boldsymbol{v}_i = \boldsymbol{c}_{m,r(p,m),k}, \quad \text{for } m = \{0,\dots,M-1\}, p = \{0,\dots,P_m-1\}, k = \{0,\dots,l_m\}, \tag{4.11}$$

where

$$i = M(kP_m + p) + m. \tag{4.12}$$

The benefit of using this scheme, is that it groups coefficients from active time-frequency areas together, leaving longer runs of zeros in $\boldsymbol{v}$, than in $\boldsymbol{c}$, which makes the bitplane coder more efficient.

Example 4.1
The first coefficient matrix:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \tag{4.13}$$

The second coefficient matrix:

$$\begin{bmatrix} 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\ 25 & 26 & 27 & 28 & 29 & 30 & 31 & 32 \end{bmatrix} \tag{4.14}$$

Interleaving these two matrices results in the following vector:

$$\begin{gathered}[1, 17, 9, 25, 5, 18, 13, 26, 2, 19, 10, 27, 6, 20, 14, 28, ... \\ ...3, 21, 11, 29, 7, 22, 15, 30, 4, 23, 12, 31, 8, 24, 16, 32]\end{gathered} \tag{4.15}$$

Note the line break is only done for readability.

4.4 Evaluation

The simulation results are evaluated differently depending on the simulation. The first measure of evaluation is the speed of OMP convergence, i.e. how many iterations it takes before the residual power becomes sufficiently small. This also corresponds to the achieved sparsity of coefficients, and can thus be useful in gauging how well the signal is represented in the framelet domain.

Secondly, results are evaluated by compression factor. That is the ratio of uncompressed number of bits to average compressed code length,

$$\frac{16 \times N}{\text{average code length}} = \text{compression factor}. \tag{4.16}$$

Note that the compression factor will be calculated for a quantization level of $q = 16$ bitplanes in the coder, whereas the compression factor is also set as a criterion for the coding algorithm in certain simulations. This requires that the stop criterion of the coder be changed to consider the bit rate of the coded bit stream rather than coding a fixed number of bit planes. This is done by coding bit planes until the bit stream becomes too long for the desired bit rate R, at which point the coder stops. The bit stream in this case should be shorter than

$$N\frac{R}{f_s}, \tag{4.17}$$

where f_s is the sampling frequency.

Likely the most important result is the perceptual quality of the quantized signals. To evaluate the quality, the Objective Difference Grade [ODG] scale is used. This scale goes from 1 to 5, with 5 being the best quality. The ODG is estimated using the Perceptual Evaluation of Audio Quality [PEAQ] standard; Matlab implementation by [Kabal, 2003]. PEAQ requires the raw uncompressed audio signal as reference.

4.5 Simulations

Several simulations are performed to verify and evaluate the performance of the compression algorithm, Algorithm 3. The simulations are detailed in this section and the results will be presented in each subsection and summarized in Section 4.6.

The following is a small example of concatenated framelets to quickly recap the established parameters for their construction.

Example 4.2
Figure 4.1 is an example of a concatenation of two framelets, both constructed from a single block of $N = 2^4 = 16$ samples. The blue framelet has a patch size of 4 and hop size of 2, meaning the resulting patch matrix has dimensions 8×4. The red framelet has a patch size of 8 and hop size of 4, resulting in a patch matrix with dimensions 4×8.

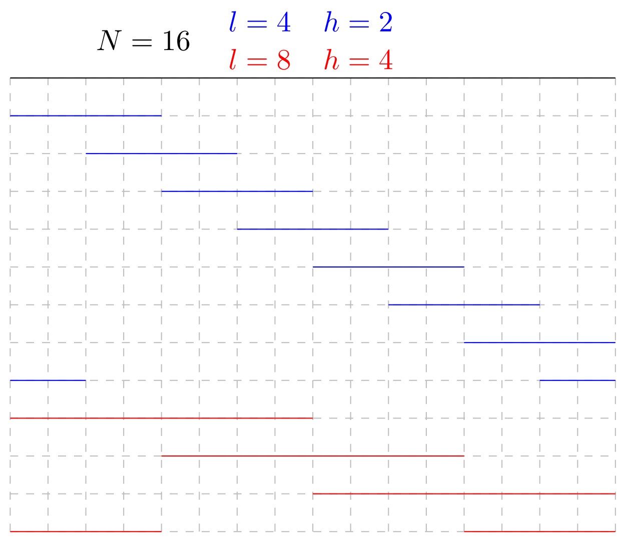


Figure 4.1: An example of concatenated framelets, with block size N, patch size l and hop size h. The colors for l and h correspond to the colors of the lines illustrating the differing patches.

The simulations are based on the concatenated convolution framelet decomposition described in Section 2.3, which will be referred to as $\boldsymbol{c} = \boldsymbol{\Psi f}$ in this section. The procedure is to decompose an N sample block, $\boldsymbol{f}_k$, of an audio signal. The framelets are constructed from DCT bases, identity bases and patch matrices with variable patch size. The frame coefficients are found using OMP. These coefficients are rearranged according to the interleaving scheme in Section 4.3.1, and passed to the bitplane coder, which outputs a quantized and compressed bitstream. Finally the performance is evaluated as described in Section 4.4. This procedure is summarized in Figure 4.2.

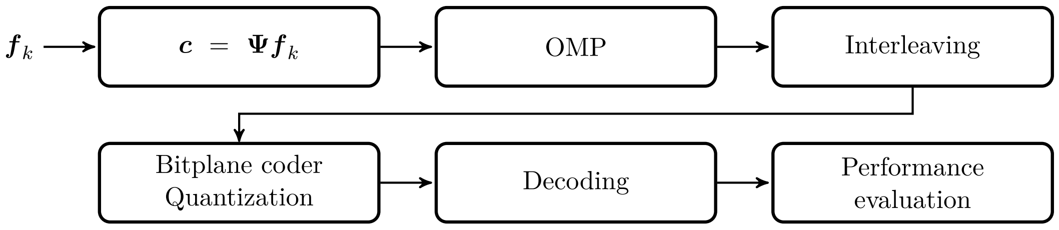


Figure 4.2: The simulation procedure is summarized by this flow chart.

The standard simulation parameters are listed here, and any deviation from these will be specified under the appropriate simulation description.

- N : 2^{12} samples.
- Audio : *Jazz.wav* [Reid, 2016], 44.1 kHz, samples $[20 \cdot N; 40 \cdot N]$.
- Framelet bases : I & DCT.
- Patch size : $\{2^7, 2^8, \ldots, 2^{12}\}$ samples.
- Hop size : half of patch size.
- OMP stop criterion : residual -50 dB.
- Coder stop criterion : 16 bitplanes.

A spectrogram of the *Jazz.wav* segment can be found in Figure 4.3,

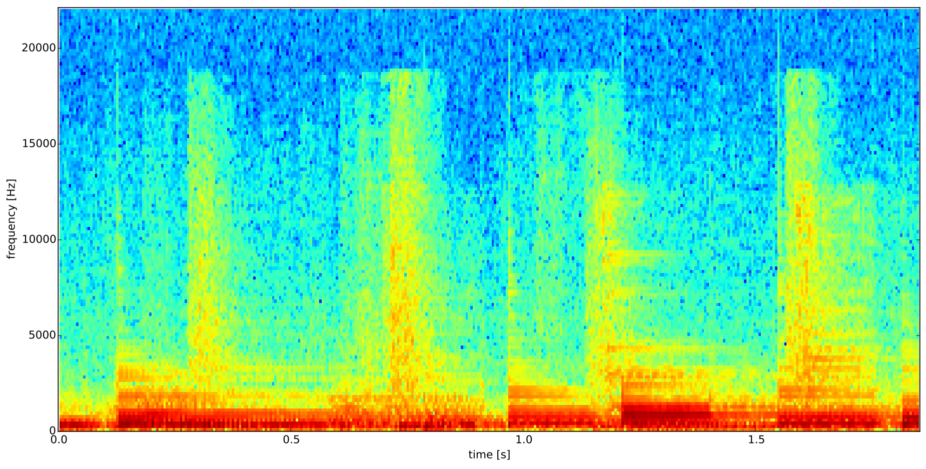


Figure 4.3: Log magnitude spectrogram of the relevant segment of *Jazz.wav*.

4.5.1 Matching pursuit vs. orthogonal matching pursuit

The matching pursuit, Algorithm 1, and orthogonal matching pursuit, Algorithm 2, are tested. The algorithms are set to run N iterations, where N is the length of the audio block. The residual power after each iteration is stored and averaged over the audio blocks at the end. This allows comparison of convergence between the two algorithms. The results can be found in Figure 4.4. MP is seen to converge considerably slower than OMP. Just before N iterations, the residual power plummets for OMP, indicating that it has reached a near perfect reconstruction.

This simulation uses the standard parameters, with the following exceptions.

- Patch size : $\{2^7, 2^8, 2^9, 2^{10}\}$ samples.
- MP and OMP stop criterion : N iterations.

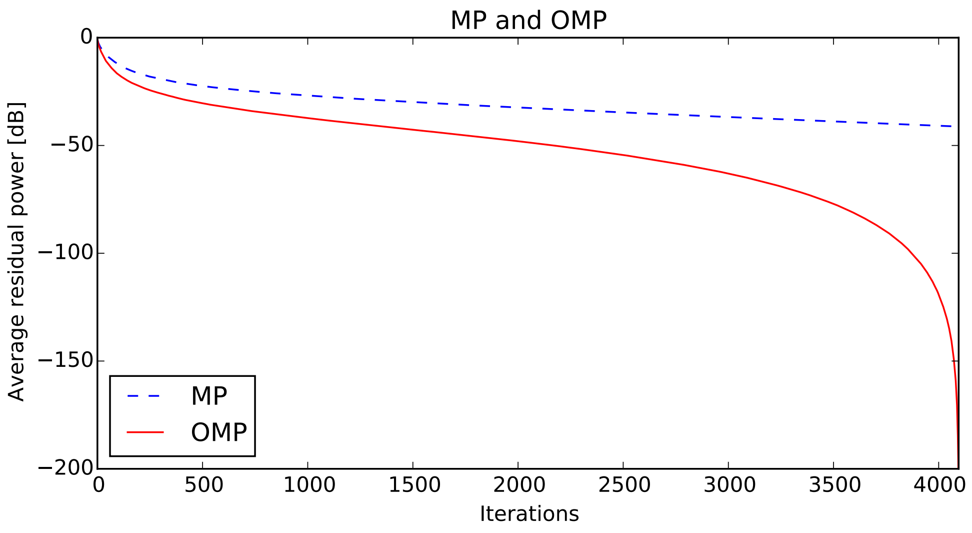


Figure 4.4: Average residual power as the MP and OMP algorithms run iterations.

In Appendix A, spectrograms of 4 partially reconstructed blocks, at eight points during the OMP and MP algorithms, are included. Figure A.2 shows spectrograms for the OMP algorithm, and Figure A.3 for MP. These spectrograms are included to give the reader an idea of the order in which the time-frequency components of a signal are chosen by OMP and MP.

4.5.2 Patch size combinations

There are many choices in constructing concatenated convolution framelets. The simulations in this project are restricted to convolution framelets using the DCT and identity bases. One interesting feature, which is explored in this simulation, is the redundancy level of $\boldsymbol{\Psi}$. This simulation will test the performance of concatenations of a variable number of framelets, with different patch sizes. Combinations of patch sizes from 2^7 samples to 2^{12} are tested. Starting with 2^7 to 2^{10} adding one additional patch size at a time up until 2^{12}. The framelets are evaluated by the convergence of OMP, seen in Figure 4.5, and compression factor in Figure 4.6.

The non-standard parameters in this simulation are,

- Patch size : $\{2^7, 2^8, \ldots, 2^{10}\}$, $\{2^7, 2^8, \ldots, 2^{11}\}$, $\{2^7, 2^8, \ldots, 2^{12}\}$ samples.

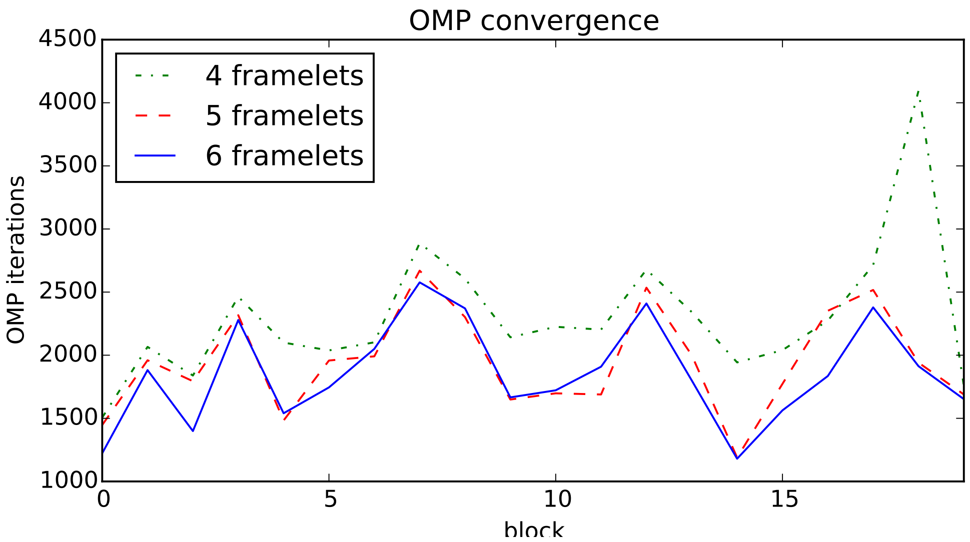


Figure 4.5: Number of iterations before OMP convergence by block. Average OMP iterations, 2301, 1947 and 1854 for 4, 5 and 6 framelets respectively.

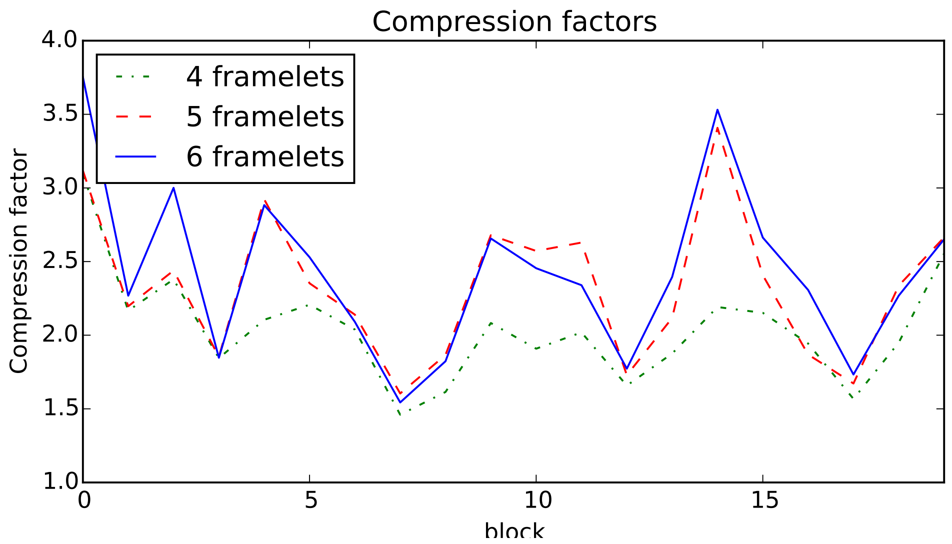


Figure 4.6: Compression factor by block. Average compression factors, 2.04, 2.32 and 2.42 for 4, 5 and 6 framelets respectively.

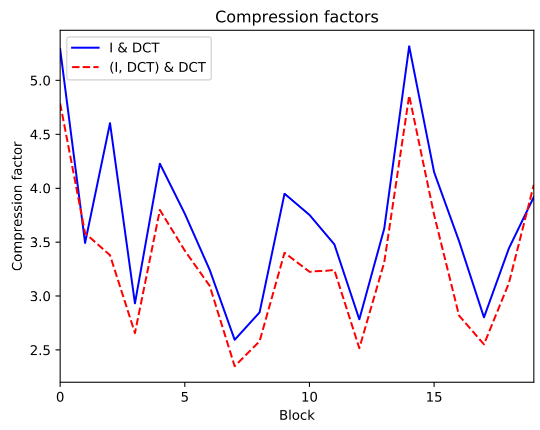
From the average values, an improvement in both OMP convergence and compression factor is observed for larger sets of framelets. It can also be seen that a larger compression factor is achieved, when the OMP converges faster, which makes sense, as fewer iterations of OMP means fewer non-zero coefficients to be coded. Finally in Appendix A, The coefficients of a single block from each of the 6 component framelets are plotted, to give an idea of what OMP selects from each of them, see Figure A.4.

4.5.3 Number of Bases

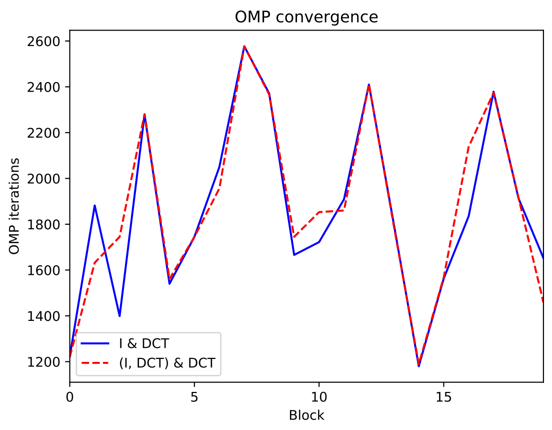
This simulation is constructed to study the effect the number of bases has on the compression factor and the rate of OMP convergence. This simulation uses the standard parameters, with the following exceptions.

- Framelet bases : I & DCT and (I, DCT) & DCT

Figures 4.7a and 4.7b show the results from the simulation.



(a) Compression factor for each block.



(b) Number of OMP iterations for each block.

Figure 4.7: Bases simulation results.

The average compression factor is calculated for both setups according to (4.16).

- I & DCT : 3.546.
- (I, DCT) & DCT : 3.199.

The additional DCT basis does not seem to be helpful in terms of OMP convergence and thus it has a negative impact on the compression factor, since the coefficient matrix is larger.

4.5.4 Interleaving

This experiment is constructed in order to determine the effectiveness of the interleaving method described in section 4.3.1 compared to simply vectorizing the coefficient matrix C along either the rows or the columns, when used in conjunction with the bitplane coder. The simulation uses the default parameters, with the exception of various vector reshaping during encoding.

Table 4.1: Minimum, maximum and mean code lengths in bits over 20 blocks.

Reshaping Type	Minimum	Maximum	Mean
Interleaving	12329	25263	18481.6
Reshape - along rows	12810	25736	19038.0
Reshape - along columns	12904	25874	19202.5

It is clear from the results of the experiment shown on table 4.1 that interleaving the coefficients improves the effectiveness of the coding scheme.

4.5.5 Compressed audio quality

In this simulation, the compression scheme is used on a larger set of musical audio. The performance is evaluated by rate of compression and quality in Objective Difference Grade [ODG]. The music used in this simulation is CD quality, i.e. 16 bit pulse code modulation with a sampling frequency of 44.1 kHz.

The non-standard parameters in this simulation are,

- Audio : various music track excerpts by various artists, 44.1 kHz, $100 \cdot N$ samples, see Appendix B.

The average OMP convergence over blocks varies between tracks, as seen in Figure 4.8.

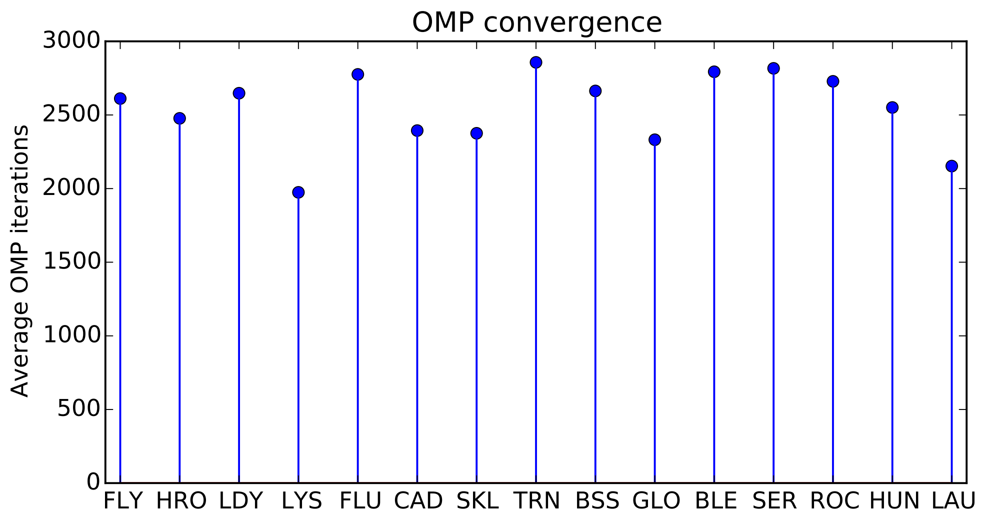


Figure 4.8: Average number of iterations before OMP convergence by track; find the list of keys in Table B.1. Average OMP iterations, 2543, across all tracks.

In the same way, the average compression factor also varies between tracks; see Figure 4.9. This is to be expected since the more sparse tracks can be more efficiently coded.

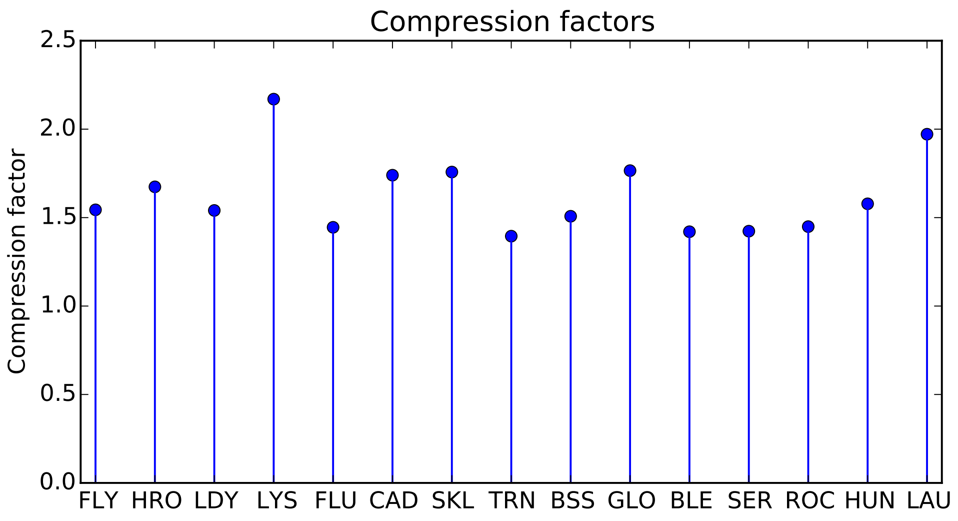


Figure 4.9: Average compression factor using 16 bitplane quantization by track; find the list of keys in Table B.1. Average compression factor, 1.625, across all tracks.

Finally the ODG as measured by PEAQ, for different degrees of quantization can be seen in Figure 4.10. The 128 and 64 kbps bit rate ODG's are compared to the ODG's after mp3 compression at the same bit rate.

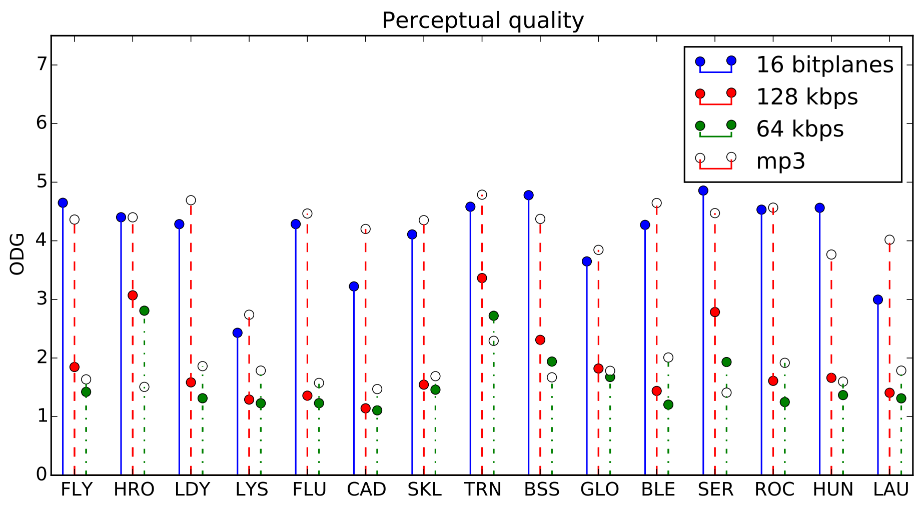


Figure 4.10: ODG for multiple quantization levels; find the list of keys in Table B.1. Average ODG's, $4.107, 1.881, 1.597$, for 16 bitplanes, 128 and 64 kbps across all tracks. Coloured markers indicate our compression scheme, and white markers indicate mp3. Note, 16 bitplanes roughly corresponds to 434 kbps given the average compression factor and uncompressed CD quality corresponds to 706 kbps.

4.6 Results

This section is a summation of the results from the simulations in Section 4.5.

- Orthogonal matching pursuit converges considerably faster compared to matching pursuit. When the number of iterations nears the block length, the residual power shows that OMP reaches a near perfect reconstruction.
- There is an improvement on average in both OMP convergence and compression factor as the number of framelets increases.
- The bases I & DCT are similar to (I, DCT) & DCT, when comparing OMP convergence. I & DCT does however perform better regarding compression factor.
- Interleaving the coefficients prior to coding is shown to improve the overall compression factor.
- When applying the method on 15 music excerpts, the results show an average compression factor of 1.625 and an average ODG of 4.107. If the coefficients are quantized during coding the ODG's drop to an average of 1.881 for 128 kbps and 1.597 for 64 kbps, indicating that artefacts are introduced to the signal. It is also observed that MP3 performs much better in terms of ODG at 128 kbps, while the ODG's are more comparable at 64 kbps.

5 | Discussion

The simulations in this project have been set up to be comparable to those in [Ravelli et al., 2008], which are also based on finding and coding a sparse set of coefficients in a redundant frame.

The observations in Section 4.6, Results, indicate that the compression scheme in this project degrade the audio quality more than the scheme described in [Ravelli et al., 2008]. Our algorithm introduce rather heavy quantization noise for bit rates of 128 and 64 kbps, with a large impact on audio quality. Comparing our algorithm with a standard MP3 coder, [Spanias et al., 2006, ch. 10.4], shows that the degradation of the audio quality is significantly higher than for MP3, especially for 128 kbps. There are a number of differences between this project and [Ravelli et al., 2008], which can factor in to explaining why this is. Firstly, this project uses convolution framelets based on the DCT rather than modified discrete cosine transforms, which is deliberate, as the concatenated convolution framelet is the novel part of the compression scheme. Secondly, [Ravelli et al., 2008] works on whole signals as a single block, which is computationally feasible, when using MP rather than OMP. Working on a signal in several blocks can break up long runs of zeros in the coding algorithm, thereby wasting bits, resulting in further need for quantization. The block size also limits the patch size, leading to the next difference. This project uses six framelets, while [Ravelli et al., 2008] uses eight bases. Using bases or framelets with wider time windows or patch sizes can result in faster convergence of MP and OMP, which also improves coding efficiency. Finally, the data set in this project consists mostly of music performed by multiple instruments and vocals, while [Ravelli et al., 2008] uses a data set, which mainly contains single instruments, and a few multi instrumental signals.

For finding sparse coefficients, OMP works well, but since the required computations grow each iteration, this limits the size of data that can feasibly be worked on at once. It might be better to use an algorithm, which exploits the statistics of audio or includes a perceptual model, to converge faster for long signals. It may also be desirable to perform the quantization during each iteration here, making it possible to cancel out quantization effects to a degree. As it is OMP optimizes distortion without taking quantization into consideration. Performing quantization during OMP effectively include quantization noise in this optimization. Of course this requires the quantization level to be known before coding. Another aspect of OMP is the convergence criterion. The lower this criterion the more non-zero coefficients will have to be coded. Depending on the quantization level, it may be better for audio quality to raise the convergence criterion. Essentially searching for a better balance between distortion from OMP and quantization.

The comparison to [Ravelli et al., 2008] is made to their results without psychoacoustic modelling, same as ours, so the comparison is fair in that regard. That being said, psychoacoustics are used in most modern audio coders, so it makes sense to discuss it. A psychoacoustic model could be included in either the MP/OMP or the coder. For example, a psychoacoustically based OMP might not select coefficients that would be masked by already selected coefficients. [Ravelli et al., 2008] proposes a psychoacoustic coder, which can likely be adapted to the concatenated convolution framelets used in this project.

In terms of audio compression, there is still work to be done with concatenated convolution framelets. It would be of interest to explore a more computationally efficient way of finding sparse coefficients, so that longer audio signals can be compressed at once. A psychoacoustic model, that can be included in the coder, to reduce perceptible quantization noise. Concatenated convolution framelets show potential as audio signal representations by way of their flexibility in both domain and redundancy level. The bases used in their construction can be chosen to fit the signal characteristics, and the redundancy can be controlled by the number of framelets, and their individual number of patches. In this project, concatenated convolution framelets were demonstrated to construct a system similar to the union of bases presented by [Ravelli et al., 2008], which was usable for audio compression, baring the problems discussed above.

Bibliography

M. Bosi and R. E. Goldberg. *Introduction to Digital Audio Coding and Standards*, volume 721. Springer Science & Business Media, 2002.

O. Christensen. *Frames and bases: An introductory course*. Springer Science & Business Media, 2008.

T. M. Cover and J. A. Thomas. Elements of information theory 2nd edition. 2006.

S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

A. Gersho and R. M. Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 1993.

P. Kabal. Pqevalaudio,”, 2003.

S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

I. M. Pu. *Fundamental data compression*. Butterworth-Heinemann, 2005.

E. Ravelli, G. Richard, and L. Daudet. Union of mdct bases for audio coding. *IEEE Transactions on audio, speech, and language processing*, 16(8):1361–1372, 2008.

C. Reid. Acoustic jazz quartet no1, 2016. URL `https://commons.wikimedia.org/wiki/File:Acoustic_Jazz_Quartet_no1_(exploration).flac`.

K. Sayood. *Introduction to data compression*. Elsevier, 2005.

A. Spanias, T. Painter, and V. Atti. *Audio signal processing and coding*. John Wiley & Sons, 2006.

D. Sundararajan. *The discrete Fourier transform: theory, algorithms and applications*. World Scientific, 2001.

L. N. Trefethen and D. Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

R. Yin, T. Gao, Y. M. Lu, and I. Daubechies. A tale of two bases: Local-nonlocal regularization on image patches with convolution framelets. *SIAM Journal on Imaging Sciences*, 10(2):711–750, 2017.

Y. You. *Audio Coding: Theory and Applications.* Springer Science & Business Media, 2010.

A | Spectrograms

This appendix contains spectrograms and plots of coefficients for Section 4.5.1 and Section 4.5.2. The spectrograms in Figures A.2 and A.3 display the time frequency content of a short segment from *Jazz.wav* during eight points in the OMP and MP algorithms. All 16 spectrograms use the same color scale. Figure A.1 contains a clean reference spectrogram. It can be seen in Figures A.2a and A.3a that both algorithms initially chose coefficients that describe the powerful low frequency content, which apparently introduces large local-time errors across all frequencies. The following figures indicate that both algorithms continue by constructing the most powerful signal content, corresponding to yellow and orange colors, from low frequencies to higher frequencies. After 1000 iterations both algorithms appear to have reconstructed the most important signal content, again indicated by warm colours, as seen in Figures A.2g and A.3g. After this point, as seen in Figures A.2h and A.3h, the algorithms focus on correcting the errors that were introduced in the earlier iterations. This appears to be where OMP begins to perform much better than MP, as there are almost no visible artefacts left in Figure A.2h.

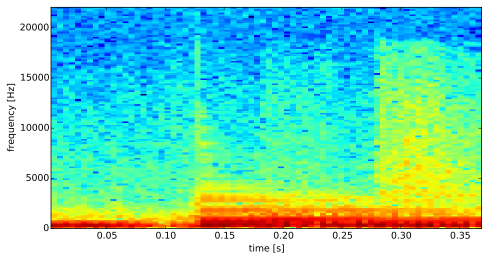
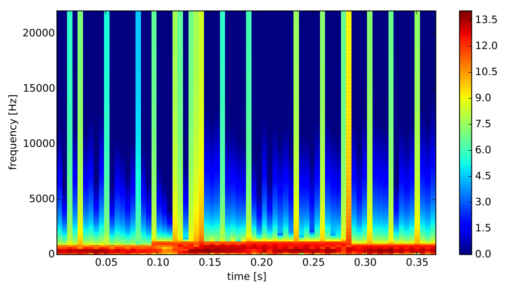
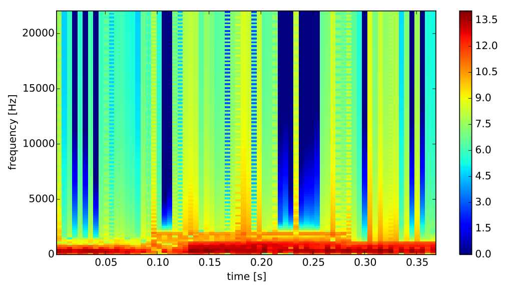


Figure A.1: Spectrogram of 4 blocks from *Jazz.wav*.

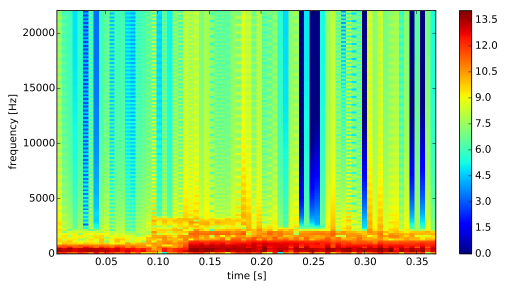
Figure A.4 contains the plotted coefficients of a single block from *Jazz.wav*, after the OMP converged to -50 dB. The coefficients are separated into 6 graphs corresponding to the 6 component framelets in the concatenated convolution framelet. From the graphs it can be seen that the longer patch size framelets contain larger coefficients. Indicating that the OMP prioritized long time patches with large energy to approximate the block, and uses lower energy short time patches to capture the finer details in the block.



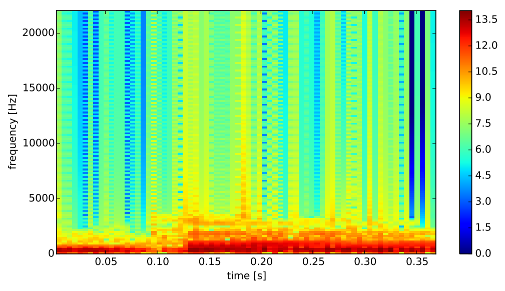
(a) Spectrogram after 10 iterations.



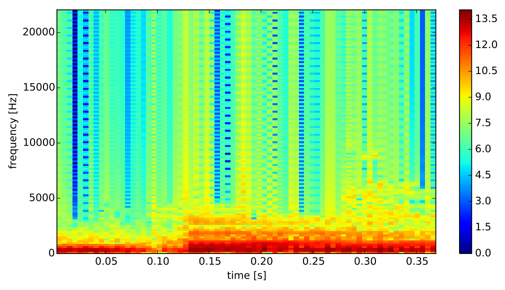
(b) Spectrogram after 50 iterations.



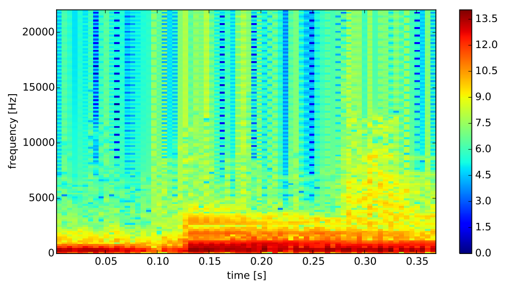
(c) Spectrogram after 100 iterations.



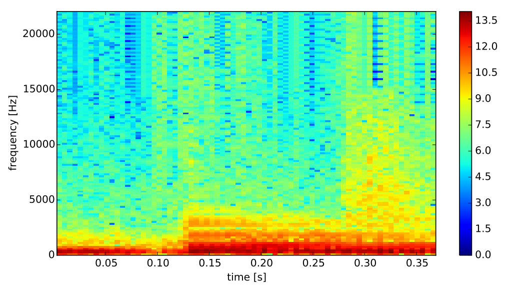
(d) Spectrogram after 150 iterations.



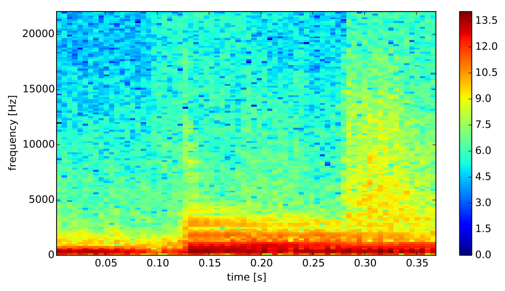
(e) Spectrogram after 250 iterations.



(f) Spectrogram after 500 iterations.

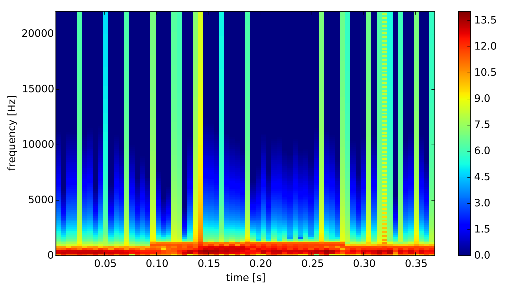


(g) Spectrogram after 1000 iterations.

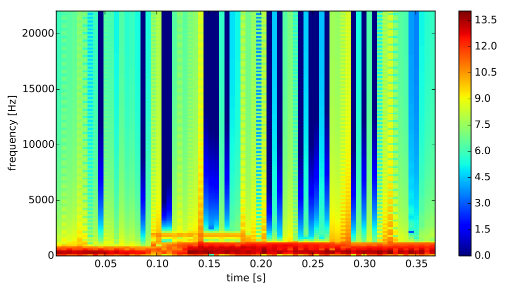


(h) Spectrogram after 2000 iterations.

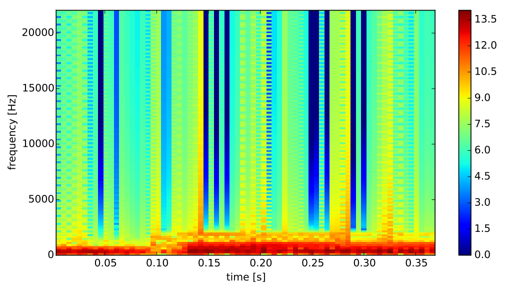
Figure A.2: Spectrograms of 4 blocks from *Jazz.wav*, at certain points during orthogonal matching pursuit.



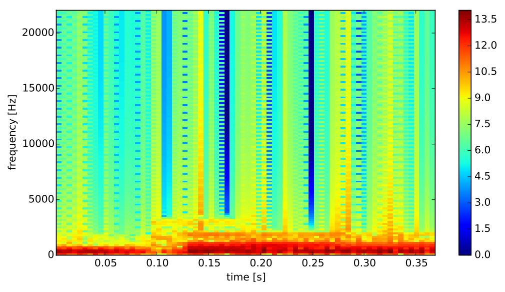
(a) Spectrogram after 10 iterations.



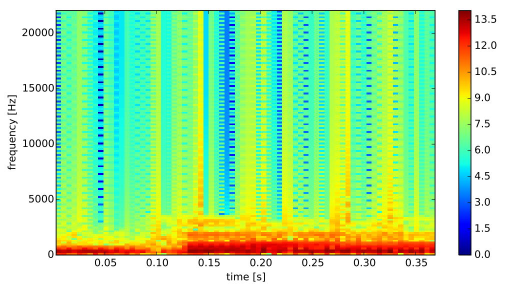
(b) Spectrogram after 50 iterations.



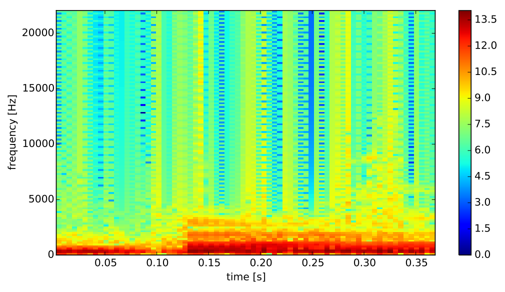
(c) Spectrogram after 100 iterations.



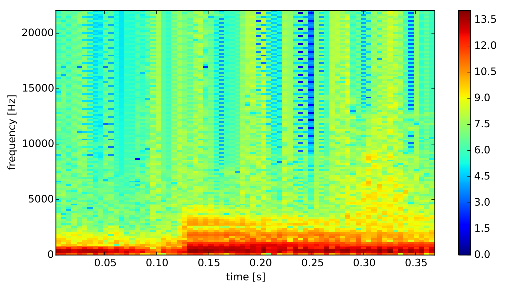
(d) Spectrogram after 150 iterations.



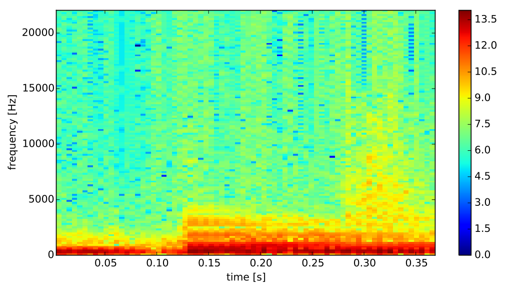
(e) Spectrogram after 250 iterations.



(f) Spectrogram after 500 iterations.

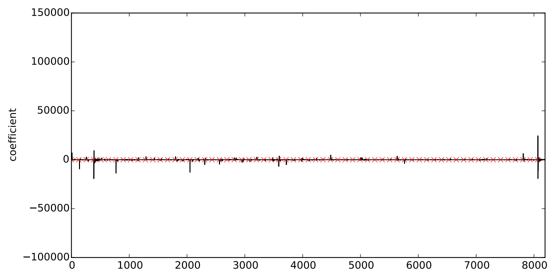


(g) Spectrogram after 1000 iterations.

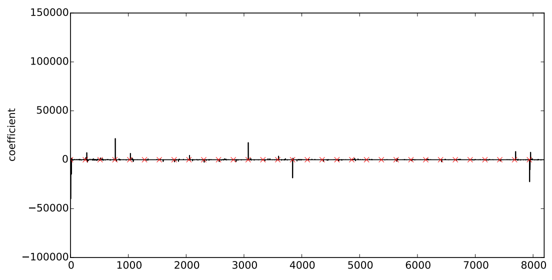


(h) Spectrogram after 2000 iterations.

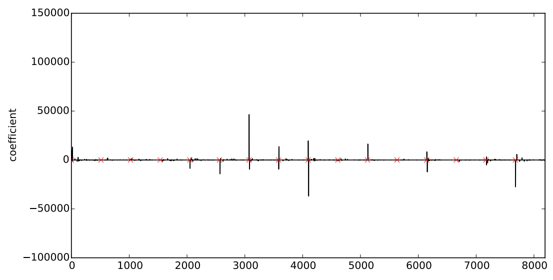
Figure A.3: Spectrograms of 4 blocks from *Jazz.wav*, at certain points during matching pursuit.



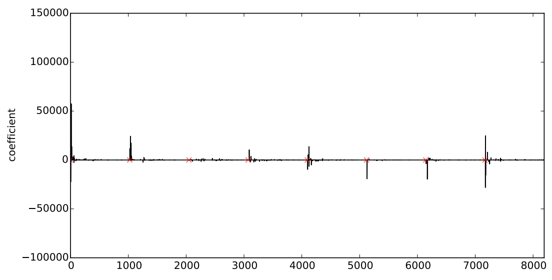
(a) Coefficients from framelet with $l = 2^7$.



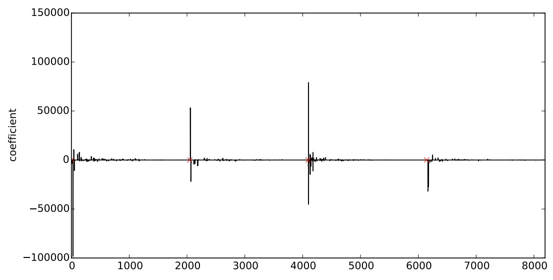
(b) Coefficients from framelet with $l = 2^8$.



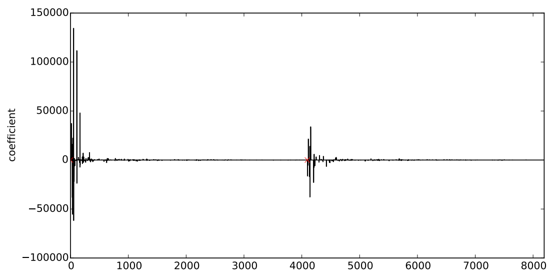
(c) Coefficients from framelet with $l = 2^9$.



(d) Coefficients from framelet with $l = 2^{10}$.



(e) Coefficients from framelet with $l = 2^{11}$.



(f) Coefficients from framelet with $l = 2^{12}$.

Figure A.4: Vectorized coefficients from each framelet in the concatenated framelet decomposition of a single block of *Jazz.wav*. l denotes the patch size and red x'es denote boundaries between patches.

B | Soundtracks

Key	Title	Artist(s)	Album
FLY	Come Fly with Me	Frank Sinatra	Nothing But the Best: The Frank Sinatra Collection
HRO	Hr. Oluf	Asynje	Færd
LDY	Lady (radio edit)	Modjo	Absolute Dance 30
LYS	Nu Tændes Tusind Julelys	Per Nielsen	Christmas Time
FLU	Flute (radio edit)	Barcode Brothers	Absolute Dance 30
CAD	Guitar Gangsters & Cadillac Blood	Volbeat	Guitar Gangsters & Cadillac Blood
SKL	Skáll	Asynje	Færd
TRN	I turn to you (Hex Hector radio mix)	Melanie C	Absolute Dance 30
BSS	Phatt bass (Aquagen short mix)	Warp Brothers	Absolute Dance 30
GLO	Gloria i excelsis	Per Nielsen	Christmas Time
BLE	Die Blechtrommel (single cut)	Taiko	Absolute Dance 30
SER	Serengeti (radio mix)	Infernal	Absolute Dance 30
ROC	What is Rock	Arone Dyer; Blue Man Group; Peter Moore	The Complex
HUN	Ungarsk Dans	Brahms	Klassiske Mesterværker
LAU	Laudale Dominum	Per Nielsen	Christmas Time
jazz.wav	Acoustic Jazz Quartet no1	Claude Reid	

Table B.1: The different soundtracks used under the simulations.