A New Audio Compression Scheme that Leverages Repetition in Music

by

Matin Lotfaliei
B.Sc., Islamic Azad University, 2015

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

We acknowledge with respect the Lekwungen peoples on whose traditional territory the university stands and the Songhees, Esquimalt, and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

A New Audio Compression Scheme
that Leverages Repetition in Music

by

Matin Lotfaliei
B.Sc., Islamic Azad University, 2015

Supervisory Committee

Dr. George Tzanetakis, Supervisor
(Department of Computer Science)

Dr. Miguel Nacenta, Departmental Member
(Department of Computer Science)

Dr. Peter Driessen, Outside Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. George Tzanetakis, Supervisor
(Department of Computer Science)

Dr. Miguel Nacenta, Departmental Member
(Department of Computer Science)

Dr. Peter Driessen, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Music frequently exhibits regular repeating structure because of musical beats, vamps, and rhythms. In this thesis, an audio compression algorithm that takes advantage of this structure is described. After providing some background, a simplified implementation of a perceptual audio compression algorithm is described and then modified to explore this idea. The newly proposed algorithm is evaluated using a publicly available dataset. The results show that the proposed algorithm can improve the compression ratio while retaining audio quality on rhythmic mixtures and background music. Better audio compression has potential applications in virtual music performance.

Table of Contents

List of Figures

List of Algorithms

List of Tables

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor **Dr. George Tzanetakis** for supporting me throughout my whole education at the University of Victoria. His kind, flexible, and positive behaviours during my stressful times were definitely very helpful. He always presented me with his excellent knowledge, ideas, references, and experiences. More specifically, there was a time when I gave up on my idea for this thesis and he encouraged me to continue simply by saying: "I have done enough research to be able to tell if there is hope or not, even if it is not visible in the results yet. We just need to polish it so it shows itself.". So without him, I probably wouldn't have stayed consistent to finally experience success.

Then, I would like to thank **Dr. Bill Bird** who allowed me to satisfy my dream of understanding and implementing different types of compression schemes in his Data Compression course. It was his excellent and carefully polished course design, videos, presentations, and assignments that made me more interested and able to satisfy my childhood curiosity about this subject.

Being a self-funded student, I would also like to thank **Leigh Christie**, my work supervisor at MistyWest company who supported me in my half-work half-school life. He cheered me and admired me for staying focused despite being in this splitted lifestyle. I believe studying would have been much more difficult without his acknowledgement, flexibility, and understanding.

Being far away from my home country Iran, I would like to thank my beloved family in Coquitlam, **Azadeh and Ali Hariri**. They treated me like my genuine mom and dad and they helped me with all different kinds of emotional and financial trauma during the last 5 years of being in Canada. I believe without their help, this thesis would have been delayed even more, I couldn't have the peace of mind, and I couldn't pass through many difficulties that I experienced. I will be in debt for their exceptional support for as long as I live.

Lastly, I would like to thank **my mom and dad and all my friends** in Iran and in Canada who encouraged me to stay strong, stay consistent, and be hopeful at the times that I felt lonely far away from home and to be mindful of the bright future ahead.

DEDICATION

I would like to dedicate this work to Michael Jackson.

I have used his unforgettable song "Billie Jean"
for my initial thought experiments in this thesis
and listened to it millions of times.

May he rest in eternal peace.

Chapter 1

Introduction

1.1 Motivation

The need for data compression, more specifically entropy coding, was introduced when television transmissions and mobile telephone services started becoming commercial as early as the 1940s [55]. Then in the 1950s, with the birth of limited and expensive digital storage devices, researchers (such as David Huffman) have started to develop methods to minimize the redundancy of messages[22]. Since then, many data compression schemes have been studied, proposed, and used in all kinds of data including text, audio, and video. With the rapid growth of computing power and the use of different compression schemes, we are now able to send and receive simultaneous real-time high-quality audio and video information in virtual group conferences held on platforms such as Zoom, Google Meet, Skype, etc.

Despite all the mentioned advancements, having cheaper storage devices, and fast internet access, the need for consuming less storage and improving transfer latency is still ongoing. According to Speedtest Global Index, 72 countries out of 182 ($\sim 40\%$) have a slower than 25 Mbps median internet speed over a fixed broadband connection[6]. The 25 Mbps is FCC's recommended connection speed for telecommuting and virtual classrooms[11]. This statistic is even worse for mobile connections, i.e. 68 countries out of 141 ($\sim 48\%$). In addition to the speed, it has been reported that users mostly have a limited amount of download/upload capacity which requires them to pay per their usage. Having said that, especially during the pandemic of COVID-19, it was very difficult for the government and health organizations of these countries to enforce remote working and virtual classes. Hence, all such issues result

in the investment of tech companies in the improvement of compression schemes to be able to provide services to more countries and to a wider range of people.

1.2 Virtual Music Performance

Even though developed countries, with faster and unlimited internet connections, could support remote working and virtual classes during the pandemic of COVID-19, some types of virtual practices are still very difficult or even impossible[2, 8]. One example of such practice is virtual music performances. Unlike a group audio call, in which people can still communicate with one second of latency, having even 100ms latency for a group to perform a music piece together can easily make them out of sync[9, 2]. This is due to the fact that in a group audio call, one person usually waits and listens to another and then speaks when they hear silence. Having one second of latency simply means one second of extra silence between two speakers which is not annoying. In virtual music performance, however, everyone plays simultaneously and the activity of listening and playing happens at the same time. Therefore, the need for reducing the latency of audio is still in high demand and one of the approaches to achieve this reduction is by improving audio compression schemes.

1.3 Domain-Specific Audio Compression

One approach to improve audio compression even further is to use a domain-specific scheme. Unlike the **MPEG Audio Layer III (MP3)** and **Advanced Audio Coding (AAC)** schemes that are designed to be general-purpose audio compression schemes, a domain-specific audio compression focuses on more restricted use cases. For example, in the virtual music performance scenario, an audio compression scheme which is specifically tailor designed for a single musical instrument can potentially achieve better compression, and consequently, less latency in transmission because the audio signal patterns are more limited and easy to recognize.

For example, a drum player can produce a limited number of sounds with the drum set that are typically repeated in predictable patterns. These repeating signals can be at the beat level based on rhythmic events, or in a decaying sound related to one single hit. We believe that an audio compression scheme can be specifically designed so that is able to understand and identify all such limited signals and rhythmic patterns and instead of sending repeating signals, should be able to simply refer

to the known previously played audio signal. This can become more difficult for a more sophisticated musical instrument like a piano which can play any mixture of 88 different notes with different intensities and duration. In addition, for a musical instrument that can play continuous notes, like a violin or a cello, it would be much more difficult to find patterns.

In this thesis, I will be investigating whether a domain-specific audio compression schema can be designed in order to improve the compression of music with repeating structure without introducing annoying perceptible differences.

1.4 Contributions

The main contributions of the thesis are:

- A simplified perceptual audio compression scheme that is based on well-known schemes such as MP3 and AAC but that can be easily implemented and understood in order to explore how audio compression works and experiments variations and modifications without having to deal with the implementation complexities of established audio coding algorithms.
- A method for leveraging repetition in music to improve audio compression is described, implemented, and tested showing the potential of domain-specific audio compression.
- For someone who is interested in evaluating audio compression schemes that leverage music repetition, an experimental methodology is proposed and applied using a publicly available dataset.
- For programmers who would like to explore the new audio compression scheme closely, the Python source code of the encoder, decoder, experiment, and figures are provided in a public GitHub repository[36].
- For someone who is interested to follow up on this thesis and investigate approaches for its improvements, some potential ideas for future works are provided in the final chapter.

1.5 History of the Thesis

Entering the University of Victoria, following my previous experience and passion for Virtual Reality (VR) and Human-Computer Interaction, I had plans to work around the Music in Virtual Reality as the subject for my thesis. Sadly, with the start of the pandemic of COVID-19, it was very difficult to conduct user studies and potentially expose participants to the virus as they would need to use expensive VR equipment that is not widely available. Therefore, I decided to choose my second life-long interest, audio compression, as my thesis without the need to run user studies.

I grew up, experiencing the excitement of compressing my audio and video files to MP3 and MP4. As a curious child, I always wondered how this is possible and if someday, I could do the same. My curiosity grew more when I got older and started working in a video conferencing service provider company. There, I noticed how a carefully chosen audio and video encoder can affect the whole experience of a video conference, and how it plays a major part in a business to be built upon. So I found this opportunity to fulfil my curiosity at the University of Victoria by taking Data Compression courses, taking Directed Studies about Audio Compression Schemes, and working on Audio Compression Schemes leveraging repetition of tones for my thesis.

1.6 Thesis Structure

To help the reader understand the flow and structure of the thesis, the aim and expected contents of each one of the chapters are described below:

Chapter 2 points out some of the published research related to the subject of my thesis in subjects of audio & video compression, evaluation of audio quality, network music performance, and repetition in sound source separation.

Chapter 3 describes some mathematics, techniques, and approaches for the background knowledge and empirical investigations that are required before proposing the new idea in the audio compression.

Chapter 4 explains the main idea of the new audio compression scheme, then it provides technical considerations and algorithmic implementations, and then it illustrates some results of its execution on a single well-known song.

Chapter 5 describes an experiment over a diverse music dataset. Results are reported using descriptive and inferential statistical analysis aiming to achieve a more generalizable understanding of the new compression scheme

Chapter 6 provides an overview of the knowledge achieved from chapter 5 and then, by referring to some of the mentioned research in chapter 2, some potential improvements are proposed that can be done in the future studies.

Chapter 2

Related Work

There is a large amount of related work to the topic of this thesis. In this chapter, an overview of related work is provided grouped into particular topics relevant to the thesis and references to the associated literature are provided.

2.1 Lossless Compression

One of the earliest and still widely used forms of entropy coding is **Huffman Coding**. In a digital storage device, string characters are represented using a fixed number of bits. When a string is converted using Huffman encoding, frequently used characters will be stored with fewer bits and not-so-frequently occurring characters will be stored with more bits, resulting in fewer bits used in total [22]. For example, the text "aab" when saved in ASCII code, uses 24 bits. By generating a Huffman Tree, it can be determined to replace the character "a" with the binary "0" and the character "b" with the binary "10". The Huffman coding of the text becomes the binary "0010" which can be stored in 4 bits.

An obvious improvement on Huffman coding is Arithmetic Coding in which instead of replacing each character with a code, the entire message can be encoded into a single number[50]. Both of these encodings can compress a message significantly, however, they are not suitable for texts with repeated content. For example, Huffman coding of the text "aabaab" is the binary "0010 0010". Similarly, in Arithmetic Coding, the encoded binary of a repeated text consumes almost double the number of bits compared to its singular form.

Therefore, before using the Huffman or Arithmetic Coding, it is typically proposed

to reduce the repetition with **Run-length encoding (RLE)** in which repeating consequent characters are replaced with the character and the length of its repetition[51]. So it changes "aabaab" into "a(×2)ba(×2)b". To reduce the repetition further, it was proposed in LZ77 and LZ78 to look back in the sequence of characters to see if it is a match and reuse it[65, 66]. So two numbers are required to mention how many characters to go back and how many of them to use. For example, "aabaab" changes into "a(1,1)b(3,3)". Even though this approach is very promising for lossless compression, the challenging part is implementing a match-finding algorithm as a poor implementation can lead to a lot of process and memory usage in large files. Welch's high-performance implementation of LZ78 (a.k.a LZW) became widely used in UNIX operating system with the `compress` command (`.Z` files) and `.gif` image format[63]. Later, by combining the LZ77 and Huffman coding, **DEFLATE algorithm** was born which is still being used in `.gz`, `.zip`, and `.png` files[18]. Some of the more space-efficient, but slower compression schemes are LZMA algorithm, which uses LZ77, Markov Chains, and Arithmetic Coding[23], is being used in `.7z`— files and Burrows-Wheeler algorithm, which uses a clever text sorting technique followed by RLE and Huffman coding[5], is being used in `.bz` files.

Even though these algorithms are highly effective on human-readable text files, such as HTML, CSS, and programming source codes, they are far less effective on raw binary data because they such data is less likely to contain repeating patterns in its raw form. Hence, the algorithms mentioned above are used after different types of lossy transforms and manipulations, described in section 2.2 and with even more details in sections 3.1 and 3.5.

2.2 Audio Compression

Studies on audio compression began before digital storage devices became widespread. Shannon proposed a logarithmic transformation of the original communication message in the year 1948[55]. Later, the Discrete Fourier transform (DFT) was proven to be very useful for audio compaction due to the wave nature of audio signals, more specifically, Discrete Cosine Transformation (DCT) was shown to be very useful in pattern recognition, and scalar-type Wiener filtering[1]. Then, using Time-Domain Aliasing Cancellation (TDAC), a **Modified version of Discrete Cosine Transform (MDCT)** was designed that is still being used in almost every audio compression scheme today[42]. Because of MDCT's importance, it will be described in more

detail in section 3.5.

Since then, many researchers focused on improvements in the MDCT. For example, the MDCT is reported to produce severe artifacts at lower bitrates. To fix this, Sinha and Johnston proposed combining it with wavelet filterbank[56], Dietz et al. developed Spectral Band Replication (SBR)[19], Brinker et al. studied Parametric Coding[4], and Ravelli et al. proposed using a redundant union of eight MDCT bases[48], which then he showed its benefits in transform-domain audio indexing[49].

The first time an audio compression scheme was standardized was in 1993 by the Moving Picture Experts Group (MPEG), consisting of an alliance of groups established by ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission). **The third layer of MPEG-1 (MP3)** standard consists of Polyphase Quadrature Filters (PQF filterbank), MDCT, and cancelling Aliases caused by PQF filterbanks[25]. The successor of MP3, **the Advanced Audio Coding (AAC)** standard was introduced in MPEG-2 [26, 3] and then used in MPEG-4 standard in 1999 [27]. AAC doesn't use the PQF filterbank like MP3. Rather, it uses a pure MDCT followed by Temporal Noise Shaping (TNS) and Perceptual Noise Substitution (PNS). Even though AAC is reported to encode with a better audio quality compared to MP3 in lower bitrates[40], both of these two widely known standards are able to generate a compressed file with less than 20% size of the PCM audio signal while the quality difference is guaranteed to be imperceptible.

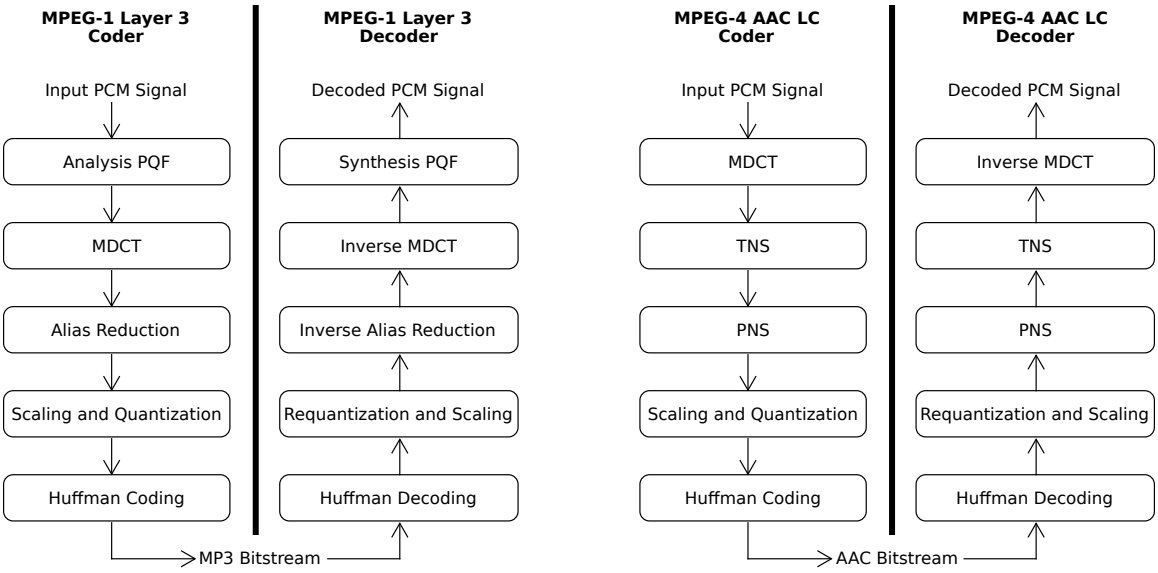


Figure 2.1: Steps taken for encoding and decoding MP3 and AAC bitstreams.

Looking at Figure 2.1, we can see that both MP3 and AAC coders receive a PCM

audio signal as input, after passing five steps they generate the compressed audio bitstream. The compressed bitstream can be decoded back to the PCM audio signal by passing through inverted versions of those five steps. Three steps are similar in both encoders: MDCT, Scaling & Quantization, and Huffman Coding. Having said that, I will be focusing more on the first two, specially MDCT, in chapter 3.

Despite being less popular, lossless audio coding has also been receiving attention. Since 2001, the Free Lossless Audio Codec (FLAC) has been an open format developed by Coalson and the Xiph.Org Foundation. Some years later, in 2005, Audio Lossless Coding (ALS) has become standardized in MPEG-4. Unlike MP3 and AAC, both FLAC and MPEG-4 ALS use different types of polynomial or linear models to achieve approximations on small blocks of PCM signal and then they store them alongside their residual difference using the Rice coding[10, 35].

2.3 Evaluation of Audio Quality

Many lossy audio encoders, including MP3 and AAC, were assessed subjectively to be able to guarantee an "indistinguishable" audio quality [25, 27]. An experimental design was required for a generalizable result with a limited number of participants who listened to different outputs of these lossy audio codecs and reported their perceived differences among them. One of the most well-known assessment methods that is still being used is called "MUlti Stimulus test with Hidden Reference and Anchor (MUSHRA)" recommended by the International Telecommunication Union (ITU)[29]. It involves a series of tests that are recognized to be one of the most reliable ways to measure the quality of audio.

Since subjective quality assessment is expensive and very time-consuming, many studies were made to develop an objective measurement method that is able to achieve an estimate of the perceived audio quality automatically from the signal. Because many of the introduced objective methods such as PESQ[31] and PSQM[30] were never thoroughly validated, ITU studied six methods and integrated their promising tools into one single method called **Perceptual Evaluation of Audio Quality (PEAQ)** [28]. PEAQ was carefully validated and it was proven to generate reliable and useful information for some applications. There have been some doubts that PEAQ represents a realistic and valid model of auditory perception because it is a composite of multiple different auditory models, several secondary feature extraction techniques and an artificial neural network. Therefore, even though the PEMO-Q

method was proposed which simplifies and slightly improves the PEAQ[21], PEAQ is still considered to be useful enough to predict subjective audio quality ratings.

PEAQ was examined and implemented by Kabal and other graduate students at McGill University as part of a course project[33] and their implementation was used for this thesis. In chapter 3, I will be describing this method and its implementation in more detail.

2.4 Repetition in Sound Source Separation

After efficient algorithms were proposed for **Non-negative Matrix Factorization (NMF)**[34] and showed its ability to identify components with temporal structure[58], attempts were made to use it to separate audio objects (such as tones)[57] and even vocal tracks[62] in an audio signal. Because this approach was dependent on particular initialization, features, and prior training for each type of audio, Rafii and Pardo proposed a new method that is based on "self-similarity" and works on any audio as long as it has a repeating structure (such as a piece of music).

At first, they named their method **REpeating Pattern Extraction Technique (REPET)** and proposed to segment the audio at the found period of the repeating structure. Then, by averaging segments, they create a repeating segment model which is compared to the original audio to label it as the foreground audio (vocal) or the background (instrumental companions)[44]. Later, Rafii and Pardo improved their method using a similarity matrix[45], noise estimation with an "online" sliding buffer(REPET-SIM)[46], and by taking the element-wise median of all the periodically repeating segments[47]. Later, supervised by Pardo, Seetharaman proposed to use a 2D Fourier Transform of the audio spectrogram[54].

There are many different REPET implementations on the internet, one of which is in the nussl python package implemented by Manilow, Seetharaman, and Pardo[38] which I used for this thesis. In chapter 3, I will be describing this algorithm in more detail.

2.5 Network Music Performance

As already mentioned in section 1.2, having even 100 milliseconds of delay between two music performers can easily make them out of sync [9, 2]. In a Network Music Performance (NMP), the time delay is split into 12 different steps starting from the

music instrument, transmitting to the network, and to the ear of the other person. These steps are known as Over-all One-way Source-to-Ear (OOSE) and each step contributes differently to the amount of delay between two performers[7, 52].

It is reported that choosing between different audio encoders can significantly affect the OOSE delay. Lutzky has reported a comparison of the algorithmic delay between MP3 and AAC audio encoders which shows at least 20 ms of delay[37]. This is hardly acceptable for the OOSE delay budget. Therefore, attempts were made to develop low-delay audio encoders such as ultra-low-delay (ULD)[53], CELT[61], and OPUS[60] encoders which were able to achieve delays as low as 4 ms.

2.6 Similarity-based Audio Compression

There has been limited work directly related to leveraging repetition for audio compression which is the main idea of the thesis. A related approach has been termed similarity-based audio compression. Cunningham et al. developed an audio compression scheme named ACER (Audio Compression Exploiting Repetition) by finding repeating matches in the audio[15], and then they tried different approaches for its improvement such as: by using static and dynamic block searches[12] and a similarity matrix [14]. They developed a file format[13] and evaluated it objectively and subjectively[17, 16]. The achievements of Cunningham et al. show promising results in the improvement of audio compression. However, they believed more improvements are needed in the search for matching signals as their approaches were mostly not efficient. Recently, Tarjano and Pereira proposed an efficient approach to segment "quasi-periodic" audio signals and despite not seeing improvements in the compression rate, they believe this can be beneficial in lossy audio compression applications[59].

Chapter 3

Background and Empirical Investigations

In this chapter, I will be describing topics that are important for understanding the main ideas behind this thesis. This background material was crucial for me to learn well and it was acquired through courses, directed studies, and empirical investigations. In each section, the necessary background in terms of mathematics, techniques, and approaches is presented targeting a reader who is not necessarily familiar with audio compression.

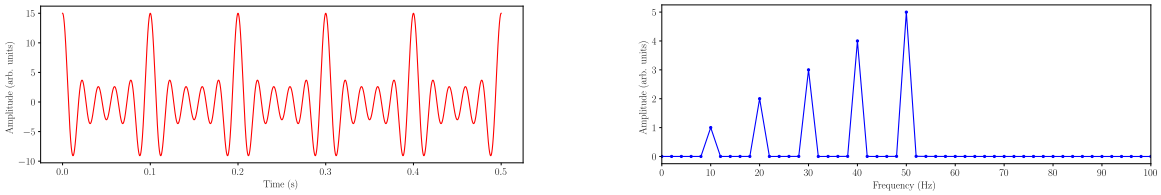
3.1 Fourier Transformation (FT)

The Fourier Transform is fundamental in audio signal processing. Raw audio signals consist of waves similar to figure 3.1a that are difficult (sometimes impossible) to interpret and process without transforming into the frequency domain. The Fourier Transform is one of the most well-known initial steps for processing audio signals. A **Fourier Transform (FT)** decomposes functions depending on time into functions depending on frequency. It is based on the Fourier Series expansion shown in equation 3.1, It uses an integral (or "continuous sum") that exploits properties of sine and cosine (with period P) to recover the amplitude (A) and phase (φ) of each sinusoid in a Fourier series (Equation 3.1). The inverse Fourier transform recombines these waves using a similar integral to reproduce the original function.

$$s_N(t) = \frac{A_0}{2} + \sum_{n=1}^{N} A_n \cdot \cos\left(\tfrac{2\pi}{P} nt - \varphi_n\right) \tag{3.1}$$

In digital signal processing, the **Discrete Fourier Transform (DFT)** converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the **Discrete-Time Fourier Transform (DTFT)**, which is a complex-valued function of frequency. Focusing on real values only, the **Discrete Cosine Transform (DCT)** is a widely used transformation technique in signal processing and data compression [1]. For example, figure 3.1a shows a sum of cosine waves at 10, 20, 30, 40, and 50 Hz (equation 3.2) which is difficult to process and store. Figure 3.1b is the DCT of the same function which clearly shows its compression potential by having many zeroes.

$$A(t) = \sum_{n=1}^{5} n \cos(nwt), \quad w = 10 \times 2\pi \tag{3.2}$$



(a) Raw Audio Signal

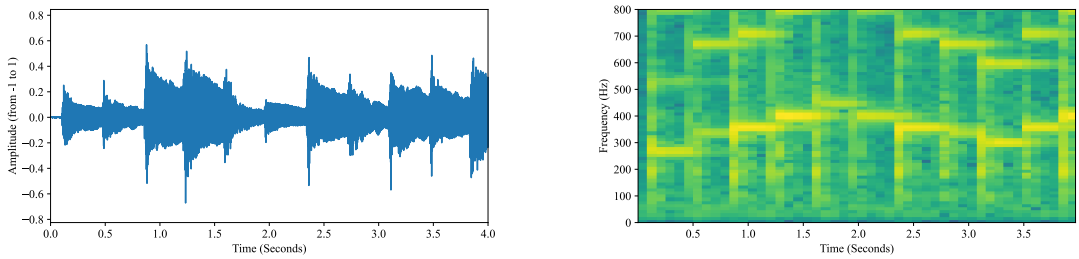
(b) Discrete Cosine Transform

Figure 3.1: Sum of cosine waves at 10, 20, 30, 40, and 50 Hz

Fourier transforms can also treat non-periodic functions as periodic with an infinite period. This can generate approximate frequency domain representations of non-periodic functions, allowing a waveform to be converted between its time-domain representation and its frequency domain representation. One realistic example can be seen in figure 3.2 which shows a recording of an acoustic piano which plays DEFGAGFEF notes in a raw waveform (3.2a) and a heat map of the time-frequency domain, a.k.a Audio Spectrogram (3.2b).

3.2 Non-Negative Matrix Factorisation

During our initial investigations on instrument-specific audio compression schemes we explore sound source separation and transcription approaches based on Non-Negative Matrix Factorisation (NMF). NMF factorizes a matrix V into two matrices W and H without any negative elements. When used in MIR, these matrices are spectrograms or some other form of time-frequency representation.



(a) Raw Audio Recording (b) Audio Spectrogram

Figure 3.2: A recording of an acoustic piano playing DEFGAGFEF notes

The relationship between these three matrices is shown in the equation 3.3. One of the applications of NMF in digital audio processing is that when the output of FT of audio containing repeating tones, as a matrix consisting of frequencies as rows and time as columns ($V_{f\times t}$), is provided to this algorithm with the right number of extraction tones (n), it will generate a matrix containing the frequencies of each tone ($W_{f\times n}$) and a matrix containing the activation time of each tone ($H_{n\times t}$). Multiplying the tones and activation matrices will recreate a matrix similar to V. Saving the W and H matrices will consume less space as compared to saving the V and it has the potential to be used as a compression scheme.

$$f\big(V_{f\times t} \quad , \quad n\big) \quad = \quad W_{f\times n}H_{n\times t} \quad \simeq \quad V_{f\times t} \tag{3.3}$$

So to try the scheme, using LibRosa in Python[39], I provided 12 seconds of acoustic piano sound (similar to figure 3.2b) to DFT to obtain a spectrogram matrix $V_{2048\times 572}$. Then NMF was applied using the *decompose* function from the *librosa* library with $n = 15$ to generate two matrices of dimensions $W_{2048\times 15}$ and $H_{15\times 572}$. This means the total number of elements to save after the factorization would be 39,300 as compared to 1,171,456 elements after the DFT ($Ratio = 3.35\%$). This would be a significant improvement in compression ratio compared to MP3 ($Ratio = 15.9\%$). However, listening to the reconstructed V was very annoying ($ODG = -3.793$). Using $n = 150$, listening to the reconstructed V was still very annoying ($Ratio = 33.5\%, ODG = -3.793$). This means that to factorize two matrices of tones and activations, many important pieces of audio are removed due to careless approximations of the NMF approach. The **Objective Difference Grade** ODG is a way to automatically computed the perceived quality of an audio signal and is explained in the section 3.3. Although the initial investigation of using an NMF-based

approach for audio compression was not successful, it provided insights that led to the development of the proposed algorithm.

3.3 Perceptual Evaluation of Audio Quality

Perceptual Evaluation of Audio Quality (PEAQ) [28] is a standardized approach used for objective measurements of perceived audio quality, and is based on generally accepted psycho acoustic principles. The overall goal of this algorithm is to obtain a quality comparison measure between two audio files, one as a reference and one decoded from the compressed version of the audio, similar to a **Subjective Difference Grade (SDG)** acquired from listening tests with human participants. This output is called the **Objective Difference Grade (ODG)**. The ODG ranges from 0 to -4 and is defined as follows:

0 = Imperceptible
-1 = Perceptible, but not annoying
-2 = Slightly annoying
-3 = Annoying
-4 = Very annoying

In cases that the ODG is higher than zero, the difference is still considered as imperceptible. This can happen because PEAQ is based on a machine learned regression model that outputs a decimal number with correlations to the aforementioned interpretations of their closest integer number.

Unfortunately, a complete implementation of PEAQ in Python could not be found. Instead, we utilized Kabal's original MATLAB implementation of the algorithm [33] and loaded the results of the evaluation to integrate with the Python code.

When running the PEAQ algorithm, one important thing to remember is that the audio shouldn't be shifted. Some audio codecs, specifically AAC, apply some paddings in their calculations. For our case, I noticed that there is 1024 sample padding in the decoded signal from the AAC codec. So I had to shift the signal back to be able to get the true ODG result.

3.4 Repeating Pattern Extraction Technique

As already mentioned in section 2.1, most general-purpose lossless compression schemes such as DEFLATE[18] are not able to compress unmodified variations of DFT or DCT because they are not able to find any repeating patterns. Another idea that was investigated early during my research was to use the DEFLATE algorithm after the audio is split using **REpeating Pattern Extraction Technique (REPET)**[47]. REPET is able to create two masks separating the foreground and background audio (M_{fg}, M_{bg}) based on identifying repeating patterns. Rafii and Pardo believe that background audio consists of many repeating tones related to instruments, drums, and bases while foreground audio doesn't have repetitions and is mostly related to vocal sounds. Therefore, REPET can be used to remove the sound of the singer and generate suitable karaoke pieces of music. More importantly, as shown in equation 3.4 foreground audio (A_{fg}) and background audio (A_{bg}) can regenerate the original audio (A_{orig}) using a simple add operation. I will refer to this regenerated audio as **the sum** of REPET's foreground and background.

$$\begin{aligned} A_{fg} &= A_{orig} \times M_{fg} \\ A_{bg} &= A_{orig} \times M_{bg} \\ A_{fg} + A_{bg} &= A_{orig} \end{aligned} \tag{3.4}$$

I hypothesized that the DEFLATE algorithm should be able to compress REPET's foreground more than the sum on songs that contain fewer vocals as compared to instrumental music. Moreover, I hypothesized that the DEFLATE algorithm should be able to compress REPET's background more than the sum because they contain more obvious repeating patterns. Later in chapter 5, these two hypotheses are being tested with an experimental study.

3.5 Modified Discrete Cosine Transform

The modified version of Discrete Cosine Transform (MDCT), as already mentioned in section 2.1, is a common step in both MP3 and AAC compression schemas. One important reason for not using DCT in audio compression schemes is that, after the inverse transformation, a quantization error appears in the time domain (figure 3.4b). Therefore, the MDCT was developed to have time-domain aliasing cancellation (TDAC) using an overlapping pre-filter technique[42]. In a lapped transformation,

as shown in figure 3.3, subsequent wave blocks are overlapped so that the last half of one block coincides with the first half of the next block. When lapping is paired with DCT, it prevents energy leakage into its higher-frequency bins (figure 3.4c), and as a result, more energy is compacted into the DC component. As a result of these advantages, the MDCT is the most widely used time-frequency transformation used in lossy compression techniques in audio data compression.

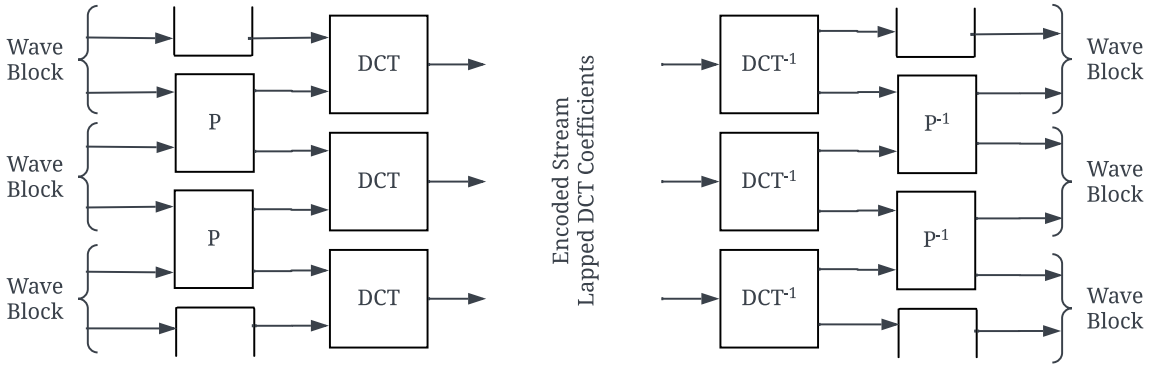


Figure 3.3: A lapped transform, consisting of the DCT with pre-filters (P) and post-filters (P^{-1}) straddling block boundaries[41]

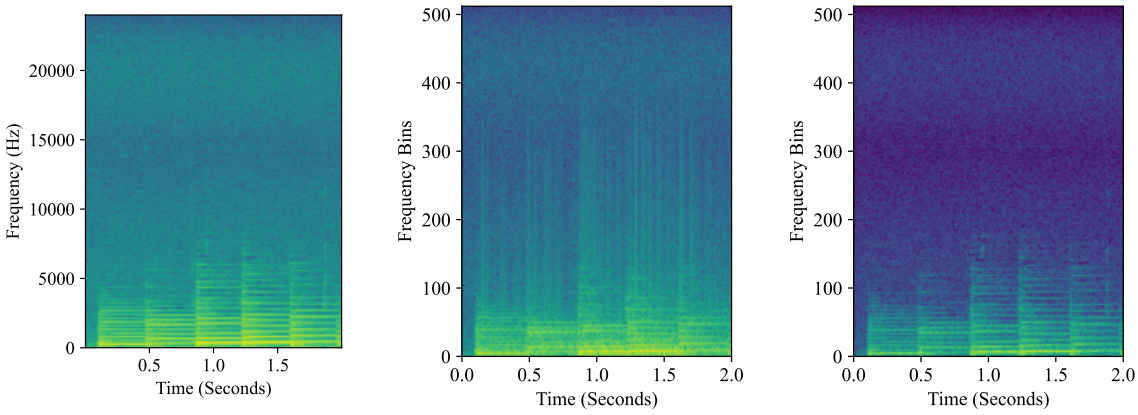
.

Figure 3.4 shows the advantage of MDCT over DCT for audios signals clearly. In 3.4b specifically, quantization errors are visible as vertical lines that repeat throughout the time domain. In 3.4c however, not only the time-domain aliasing is cancelled, but it can also be seen that energies in frequency bins are more concentrated in lower frequencies, making the figure darker as compared to the Spectrum and DCT. As a result of all these benefits, the MDCT is very suitable for audio data compression.

3.6 A Simple Lossy Audio Compression Scheme

In order to better understand the building blocks of perceptual lossy audio compression, a simple stream-lined lossy audio compression scheme was designed and implemented. This allowed more easy experimentation without having to deal with all the implementation complexities of a fully developed audio coder while still retaining the main characteristics of such audio coders.

Once the MDCT is applied to an audio signal, the resulting data is typically encoded using some form of entropy-based encoding in order to compress it while still retaining the critical and perceptual parts of the signal. In section 2.2, many



(a) Spectrum (b) DCT (c) MDCT

Figure 3.4: DCT and MDCT applied on a recording of an acoustic piano playing DEFGAGFEF notes

different approaches for such encoding were introduced. These techniques are carefully designed and modified to find correct parameters that would keep the compressed difference imperceptible. Unfortunately, I didn't have enough time to go through each one of the steps individually to be able to implement a full-blown MP3 or AAC encoder. Instead, I decided to implement multiple simple proof-of-concepts steps and report the Compression Size Ratio and Audio Quality of each step. For checking the Audio Quality, unlike MP3 and AAC that recruited humans for scoring, I used the ODG in the PEAQ method (described in section 3.3). My aim during such streamlined steps was to modify frequency bins so that when M is given to a standard DEFLATE algorithm, I could achieve a better compression ratio. It reasonable to assume that any observed differences in compression ratios in the streamlined scheme would most likely result in similar improvements.

The outcome of each step is shown in Table 3.1 and the decoded wave audio of each step can be found in the Github repository of the project[36].

Step 1: Thanks to Nils Werner's MDCT Python package[64], I had no problem getting the coefficients matrix (M). The MDCT function receives raw wave audio signals (W) and returns the matrix of coefficients with 512 frequency bins per frame in the time domain (t). It is important to mention that in Python, each element of W is 16-bits and the element size of M is 64-bits. Even though it is nice to have high precision, storing the matrix would require a lot more

space compared to W. Therefore, I changed the data type of M back to 16-bit numbers. As a result of this step, I achieved only 91% of the original size while changes in the decoded sound quality were imperceptible.

Step 2 One of the most simple techniques that can improve the compression ratio, is to zero out all high frequencies. It was mentioned earlier that human ears can not perceive extreme high frequencies. This frequency limit is different for each individual, and different lossy audio compression schemes have chosen different parameters and utilize more complex psychoacoustic models. Instead, as a simple approximation I chose to zero out half of the frequency bins in each time frame (from bin 250 to 511). As a result of this step, I simply achieved 46.8% of the original size (exactly half of Step 1) while changes in the decoded sound quality were still imperceptible. Obviously, if I was using a more complex audio signal (like a voice), I would have removed many critical parts of the audio and the decoded sound quality would most probably be perceptible. However, for a simple audio signal such as an acoustic piano, this step seems to be suitable.

Step 3 Another simple technique to improve the compression ratio, is to remove decimal points by utilizing quantization. It was already mentioned that initially, the matrix M has been reduced from 64-bit floating-point numbers (52 decimal digits) to 16-bit floating-point numbers (10 decimal digits) without any perceptible damage. I believe human ears are still not able to perceive that level of details and they most probably consider two signals of 0.3211 and 0.3212 amplitudes as similar. I personally chose to round all numbers in the matrix M to 3 digits of precision. As a result of this step, I achieved 15.4% of the original size (almost one-third of Step 2). However, such careless modification was slightly perceptible in the decoded audio quality. Hearing the differences is difficult and when noticed, it was not annoying.

Step 4 The last simple technique to improve the compression ratio, is to remove frequencies that have a very low amplitude. I believe human ears are not able to perceive subtle and silent noises like signals of 0.001 amplitudes. So I personally chose to zero out and threshold all numbers in the matrix M that are lower than 0.01 amplitude. As a result of this step, I achieved 6.6% of the original size (almost one-third of Step 3). Obviously, such careless modification was more perceptible in the decoded audio quality. Even though notes and the melody

being played on the piano are still perceivable in the decoded signal, one can get slightly annoyed when the raw and the decoded signals are compared together. This means many critical audio signals were removed despite winning over the output sizes of MP3 and AAC encoders.

Step 5 Finally, we need to apply a classic Run-Length encoding and Huffman coding on the matrix M. Gladly, Numpy Python Package[20] already has implemented the DEFLATE algorithm in its `savez_compressed` function. The function creates a compressed file with `npz` extension which can later be loaded with Numpy's `load` function. Note that other techniques can be used to write a compressed output. But I used this function for its simplicity. To report the outcome of each of the above steps in table 3.1, I repeated Step 5 so that I could check the relative improvement of the compression size. But for the final outcome of my improvised simple encoding scheme, it can only be used once as the last step. Algorithm 1 is the pseudo-code of the encoding scheme.

Algorithm 1 A simple lossy audio compression scheme

function ENCODER(W)
  $M_{t\times b} \leftarrow \text{MDCT}(W)$ ▷ Step 1: Transformation
  $M \leftarrow \begin{cases} 0, & \text{for all } b > 250 \\ M, & \text{otherwise} \end{cases}$ ▷ Step 2: High Frequency Cut
  $M \leftarrow \lfloor M \times 10^3 \rfloor \times 10^{-3}$ ▷ Step 3: Round Decimals
  $M \leftarrow \begin{cases} 0, & \text{if } |M| < 10^{-3} \\ M, & \text{otherwise} \end{cases}$ ▷ Step 4: Zero Threshold
  return DEFLATE(M)
end function

function DECODER(Z)
  $M_{t\times b} \leftarrow \text{DEFLATE}^{-1}(Z)$
  return $\text{MDCT}^{-1}(M)$
end function

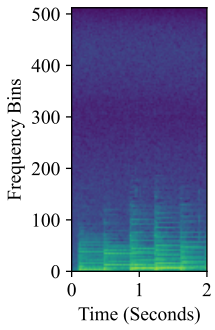
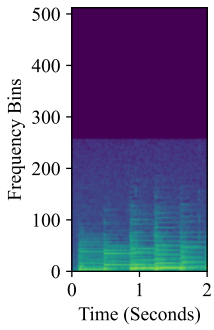
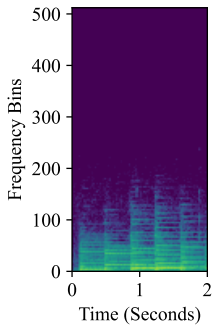
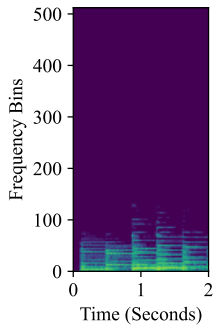
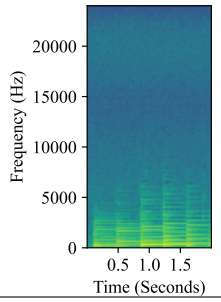
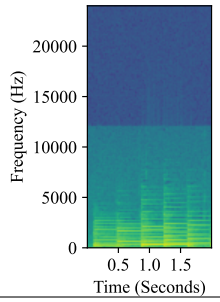
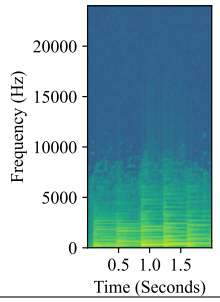
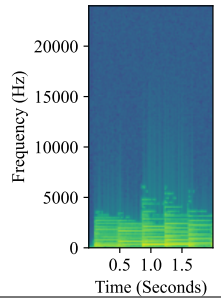
	Step 1 Only MDCT	Step 2 High Freq. Cut	Step 3 Dec. Round	Step 4 Zero Threshold
Size Ratio	91.1%	46.8%	15.4%	6.66%
MDCT				
iMDCT				
Audio Quality (ODG)	0.122 Imperceptible	0.067 Imperceptible	−1.097 Perceptible, Not Annoying	−2.138 Slightly Annoying

Table 3.1: Steps of lossy audio compression using MDCT on a recording of an acoustic piano playing DEFGAGFEF notes

3.7 Summary

In this chapter, the main building blocks behind perceptual audio compression and audio quality evaluation were described. In addition, we discussed sound sound separation using Non-Negative Matrix Factorization and repeating musical structure. Finally, a simplified audio compression algorithms that resembles more complicated existing algorithms was described. This simplified algorithm is extended and tuned to leverage repeating structure. It is the main topic of this thesis and described in Chapter 4.

Chapter 4

Proposed Approach

Following the objective of the sections 3.2 and 3.4, I would like to narrow my focus to only one example that I believe has an obvious instrumental repetition, and I (and maybe others) have listened to it enough that can relate and interpret signals more effectively. Figure 4.1 shows the spectrum of the first 4 seconds of the song "Billie Jean" by Michael Jackson[32]. The song starts with 5 seconds of repeating drum hits before other instruments join and the singer starts singing after 25 seconds. This repeating drum continues for 4 minutes until the end of the song. So in this chapter, I am investigating if I can find a way to reuse this drum sound during the whole music and as a result, achieve an acceptable lossy audio compression.

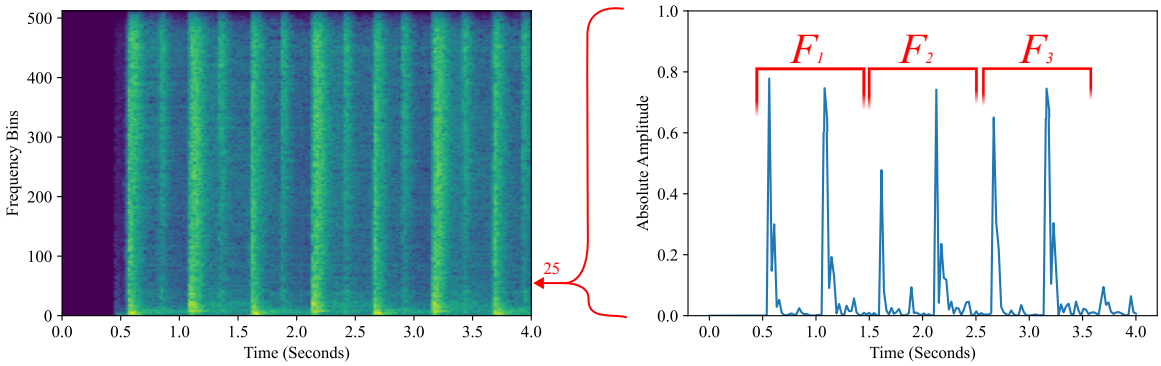


Figure 4.1: Illustration of repeating beats in the song "Billie Jean" by Michael Jackson[32] starting at the beginning.

4.1 Main Idea

To analyze the repeating signal easier, I focus on MDCT's 25th frequency bin which is shown on the right side of figure 4.1. As can be seen in this Amplitude-Time plot, repeating drum sounds are visible as sudden spikes jumping from 0 to .8 in the amplitude. Knowing the song and seeing this plot, I can identify 3 repeating frames: first from 0.5 to 1.5 seconds (F_1), second from 1.5 to 2.5 seconds (F_2), and third from 2.5 to 3.5 seconds (F_3). As for a computer to identify matching frames, it can pick a frame (for example F_3) and loop through a shifting frame that is going backward in time while calculating the sum of absolute differences between these two frames. As can be seen in figure 4.2, the output of this calculation has the minimum amount at 1 second. This means that the best candidate for F_3 to reuse is a frame starting 1 second earlier, i.e F_2.

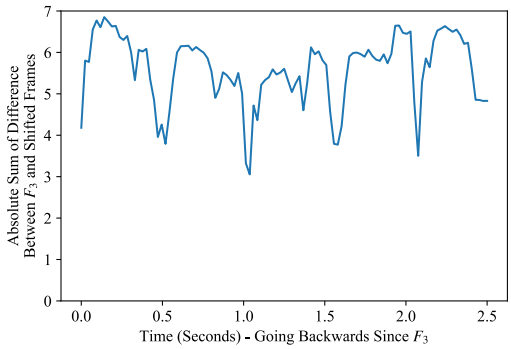
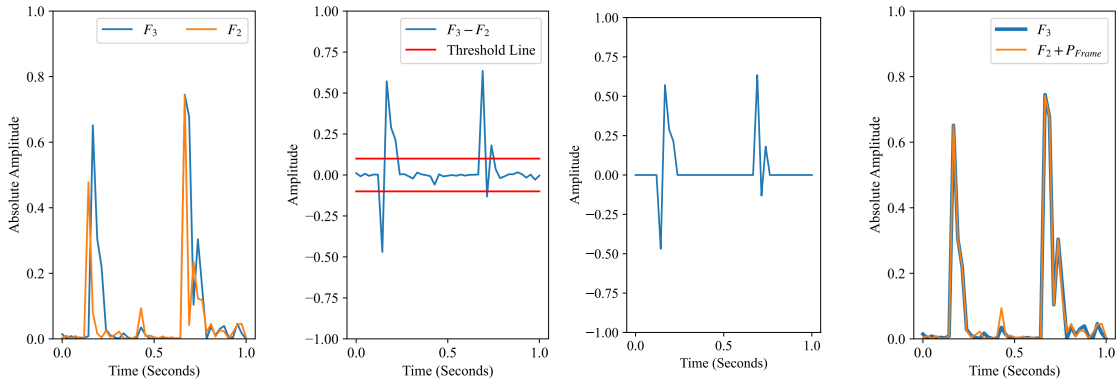


Figure 4.2: Sum of Absolute Differences between F_3 and a shifting frame going backwards in time.

The minimum amount in figure 4.2 is almost always not zero. That is because even though two audio signals sound identical to a human ear, they have some level of imperceptible differences. Figure 4.3a shows these differences more clearly by overlapping both F_3 and F_2 on each other. When subtracting these two frames, as can be seen in figure 4.3b, the output is not zero, but it has some noises around the zero amplitude. This can be evidence of the reason the DEFLATE algorithm can not perform its lossless compression over songs with repeating notes because the repetition is not exact.

So because human ears can not perceive minor differences between F_3 and F_2, I propose that by manipulating signals we can help the DEFLATE algorithm and achieve a better compression size. This manipulation can be done by removing noises

around the zero amplitude of differences between two frames using a threshold. Figure 4.3b shows this threshold should be able to separate perceptible differences from imperceptible ones. Applying this **Zero Threshold** would result in many zeros (figure 4.3c) which can be useful for the DEFLATE algorithm to identify as a reusable signal. The resulting frame has similarities with P-Frames in the video encoding layer of MPEG-1 standard[24].



(a) F_3 and F_2 (b) $F_3 - F_2$ (c) P_3 (d) F_3 and $F_2 + P_3$

Figure 4.3: Illustration of P-Frame logic

The rest of the properties of this audio compression idea are inspired by the MPEG-1 video encoder in which two types of frames are defined. **Intra-Frames or Independent-Frames (I-Frames)** are the ones that can be decoded independently and they contain a complete JPEG image. **Inter-Frames or Predicted-Frames (P-Frames)** are the ones that are dependent on another I-Frame or P-Frame to be decoded and they only contain the difference between the two images.

So similar to the MPEG scheme, a frame that is made from the differences of audio frames can be named a P-Frame, while an independent audio frame can be named an I-Frame. The proposed encoding saves I-Frames alongside P-Frames. P-Frames also hold a number to indicate how many frames are they referring back in time while they are keeping only the differences between the frame at their place and the referred frame (as shown with arrows in figure 4.4). The decoder would reconstruct frames by adding the referenced frame with the stored difference. Figure 4.3d shows the similarity of the original F_3 and the reconstructed signal. Obviously, the similarity between the original and the reconstructed signal is dependent on the threshold parameter. If the threshold parameter is chosen carelessly, it can produce

perceptible differences in the decoded signal.

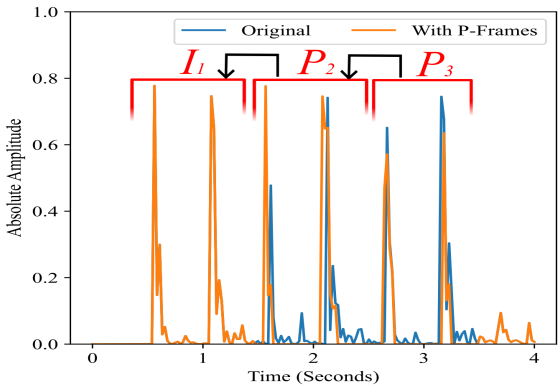


Figure 4.4: Illustration of P-Frame logic and its comparison outcome with the original

To make this proposed approach more efficient, some conditions need to be defined, managed, and clarified:

- A P-Frame should be used only if it has less information compared to the corresponding I-Frame. In cases where a repeating frame can not be found, using a P-Frame would contain a greater or equal amount of information and therefore it would not useful for the compression size.

- Each decoding from a P-Frame produces a signal that is imperceptible compared to the original. However, after continuous use of P-Frames, all these minor imperceptible differences can accumulate to some quiet but perceptible noises. Therefore, similar to the MPEG scheme, I-Frames should be forced to be used on a constant frequency. The frequency of I-Frame enforcement can be called **I-Rate**.

- It would be very inefficient to find a suitable similarity between two frames that are 1 minute apart. Finding a suitable reference for a P-Frame should have a limit. Without this limit, the processing time would increase exponentially. We can use the same I-Rate parameter and enforce P-Frames not to look further behind an enforced I-Frame.

- It is not efficient for a frame to have the size of 1 second. Even though the 1 second was used in the example above for simpler analysis, in fact, a smaller frame size can increase the chances of finding similarities. I propose that we use the smallest frame size possible which is one single MDCT time block or one

single column of MDCT containing only 512 frequency bins. It an audio file with sample rate of 44.1 KHz, it would be the equavalent of .023 of a second ($= 1/(44100/512/2)$).

- Because the frame size is proposed to be only one single MDCT columns, our frame comparisons will only be operating in the frequency domain and not the time domain. Therefore, even though the comparison method might be similar to the correlator or the matched filter, the comparison is fundamentally different because of the use of MDCT.

4.2 Implementation

Algorithm 2 shows a function that is responsible to find a suitable P-Frame using the conditions mentioned above. It creates an I-Frame as a comparison to all shifting P-Frames and chooses the one that would create the most number of zeros. The function then returns the generated frame (F) and the relative reference of a frame that was used to generate it (r). If the function decides on using an I-Frame, it would use zero inside the r variable.

As can be seen in algorithm 3, by calling the defined function repeatedly over the whole MDCT matrix, we can achieve a modified matrix of frames and an array of all used references. These two data structures can then be given to the DEFLATE algorithm to store on a hard drive. Later for decoding, these two data structures can be reconstructed back from DEFLATE and P-Frames, identifiable by their non-zero reference, can be changed back into I-Frames by summing the referenced frame with the P-Frame. Lastly, the matrix can use the inverse of MDCT to produce raw audio signals.

4.3 Exploration Tests

Now with the new P-Frame scheme defined, it is time to observe how it behaves on our chosen example, the "Billie Jean" song. The parameters that I used were: 2 seconds for I-Rate, 10^{-3} for P-Frame Threshold, and 0 for I-Frame Threshold. Looking at the histogram of generated references in figure 4.5, it can be seen that more than 98% of frames were chosen to be P-Frames ($h(0) = 1.5\%$) which means for each one of them, at least one element (2 bytes) became zero and helped to improve the compression.

Algorithm 2 Finding the most similar frames for the encoding (P-Frame)

function ENCODEFRAME($M_{t\times 512}, t, iRate, iThreshold, pThreshold$)
  $P \leftarrow$ ZEROS($iRate$)
  $F \leftarrow$ ZEROS($iRate, 512$)

  $N \leftarrow$ MAX($M[t]$)
  $F[0] \leftarrow$ THRESHOLD($M[t], iThreshold \times N$) ▷ The I-Frame
  $P[0] \leftarrow$ COUNT_ZEROS(F_0)

  for $i = 1$ **to** $iRate$ **do**
    if $(t - i) \mod iRate = 0$ **then**
      break ▷ The I-Rate Limit
    end if

    $dF \leftarrow M[t] - M[t - i]$
    $F[i] \leftarrow$ THRESHOLD($dF, pThreshold$) ▷ P-Frames
    $P[i] \leftarrow$ COUNT_ZEROS($F[i]$)
  end for

  $r \leftarrow$ ARGMAX(P) ▷ Pick the Best
  return $(r, F[r])$
end function

Interestingly, the histogram seems to be having jumps around 44 frames (1 second), 33 frames (.75 of a second), 22 frames (.5 of a second), and 11 frames (.25 of a second). This means that the algorithm is able to find the 1-second repeating beats and fractions of it based on the song's 4/4 time signature.

Most noticeably in Figure 4.5, we can see that more than 13.2% of P-Frames refer to their most immediate previous frame ($\sim$ .023 of a second) and more than 5% refer to their second most immediate previous frame ($\sim$ .045 of a second). This probably means even though repeating beats and notes can have similar frames, those frames related to the same note are even more similar. Looking at the differences between I-Frame only and P-Frame spectrum in figure 4.6, we can see that 4.6b has darker areas compared to the 4.6a, especially in areas that the drum sound is decaying and the red line is higher. The red line shows the increase in the number of zero elements that were added after applying the P-Frame scheme. This is an indication of how many bytes were saved and more compression each P-Frame.

Looking at the used references in Figure 4.6c, we can see that between 0 to 2 seconds, the distance of references increases linearly except for areas where an unfa-

Algorithm 3 Proposed new P-Frame scheme

```
function ENCODE(W, iRate, iThreshold, pThreshold)
    M_{t×512} ← MDCT(W)
    R_t ← ZEROS(t)
    for i = 0 to t do
        R[i], M[i] ← ENCODEFRAME(M, i, iRate, iThreshold, pThreshold)
    end for
    return DEFLATE(R), DEFLATE(M)
end function

function DECODE(Z_r, Z_m)
    M_{t×b} ← DEFLATE^{-1}(Z_m)
    R_t ← DEFLATE^{-1}(Z_r)
    for i = 0 to t do
        if R[i] > 0 then
            M[i] ← M[i] + M[i − R[i]]
        end if
    end for
    return MDCT^{-1}(M)
end function
```

miliar or new signal is played. At 2 seconds, because the parameter I-Rate was set, references suddenly drop to zero and they would not refer to any frames before 2 seconds. This will be repeated every 2 seconds.

In a different part of the song where more musical instruments are being played and Michael Jackson starts singing, we can see in figure 4.7b shows less increase in the number of zero elements as compared to 4.7a. Expectedly, at 29.5 seconds where vocal signals seem to look more chaotic and less repetitive, we can see that the algorithm has failed to find similarities with previous frames. Similarly, in figure 4.7c, we can see that the distance of references suddenly drops at 29.5 seconds before I-Rate enforces an I-Frame at 30 seconds. This would be an indication that the P-Frame schema behaves differently in different parts of music depending on its ability to detect and reuse similar or repeated information.

Overall, with the specified parameters, the P-Frame encoder is able to compress the "Billie Jean" song to more than 89.5 % as compared to the lossless I-Frame scheme which is 95.1 %. Because no other perceptual techniques were active, this means the P-Frame scheme alone has a 5.6 % improvement in compression ratio. Evaluating the perceptual audio quality of the decoded signal, I got -.858 for the objective difference

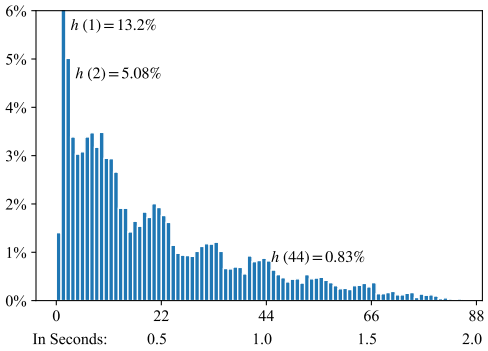
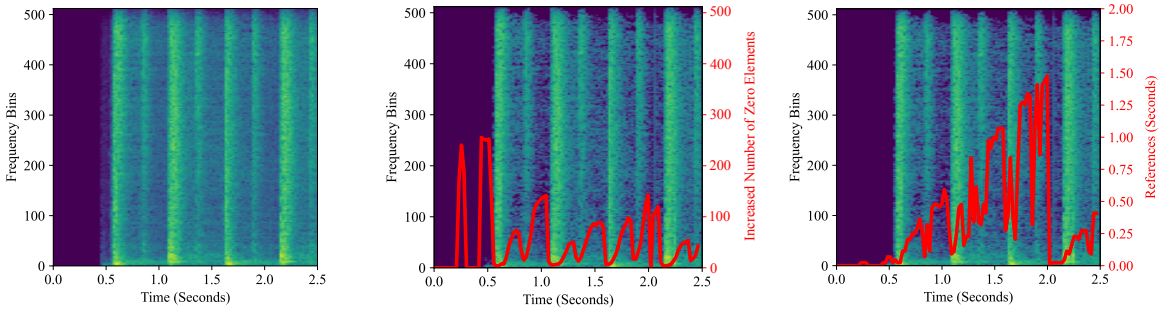


Figure 4.5: Histogram of References from P-Frame encoding over the song "Billie Jean" by Michael Jackson

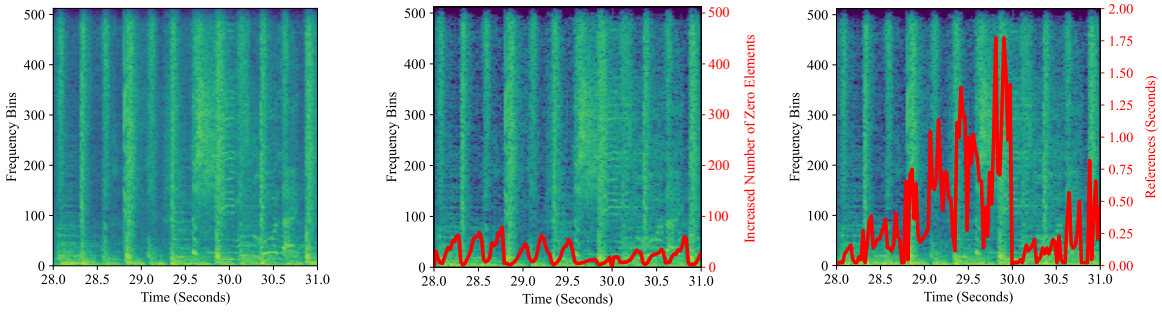


(a) I-Frames Only (b) P-Frames & Zeros (c) P-Frames & References

Figure 4.6: Effectiveness of P-Frames over the drums in the beginning song "Billie Jean" by Michael Jackson

degree which means the difference between the original and the decoded signal is hard to perceive by the human ears.

The aforementioned results for this specific example indicate that the new approach is feasible and does a good job of compressing "Billie Jean" song. Moreover, more songs are required to be tested and analyzed in order to consider these results more generally applicable. To investigate this, I run the P-Frame scheme on a larger dataset and I analyze these results statistically in chapter 5.



(a) I-Frames Only (b) P-Frames & Zeros (c) P-Frames & References

Figure 4.7: Effectiveness of P-Frames over the vocals in the song "Billie Jean" by Michael Jackson

Chapter 5

Experiments

In this chapter, I try to answer the research question: Can the described P-Frame schema improve the compression of pieces of music with repeated instrumental tones without adding perceptible differences? My hypothesis is that by reusing portions of audio that are being repeated (P-Frames), as long as a piece of music has repeating structure, the scheme should be able to compress better as compared to the compression of every single frame individually (I-Frames). Therefore, to answer the research question quantitatively, it is required to have a statistical analysis of numerical outputs (Dependent Variables) of the scheme that is run over a random collection of pieces of music while controlling all different variables to isolate our conditions (Independent Variables). The following describes my experimental design:

Population: To have an unbiased and random set of pieces of music, I used the "test" set in MUSDB-HQ which consists of 50 full-length songs (∼3h duration) with different genres along with their isolated drums, bass, vocals and others stems [43].

Dependent Variables (DVs): After running the compression scheme on each audio file, I measure the following:

- **Audio Quality:** the audio quality was measured using Objective Difference Grade (ODG) of Perceptual Evaluation of Audio Quality (PEAQ) [28] to ensure that the compression scheme does not add perceptible artifacts. The range is between 0 and -4.
- **Compression Ratio:** As each song has a different length, the output size of files is compared by measuring the ratio, so the range is between 0 and

1. It is calculated as: Compressed File Size / Uncompressed File Size

- **Desirability:** As ODG and Compression-Ratio are correlated, I need a new variable that can help me interpret the effectiveness of the scheme. This measure is named Desirability, has a range between 0 and -1, and is calculated as: (ODG/4) / Compression Ratio
- **P-Ratio:** To measure the number of frames that are reused in the compressed file. P-Ratio has a range between 0 and 1 and is calculated as: N(used P-Frames) / N(all frames)

Control Variables: To ensure that we measure only the effectiveness of reused frames, all other techniques of audio compression should be disabled and non-zero parameters should be set to the lowest amount. Therefore, after running the MDCT algorithm, the following parameters are used:

- **High Frequency Cut** = 0
- **Rounding Decimals** = 0
- **P-Frame Zero Threshold** = 10^{-5}; this value can not be zero. So because MDCT numbers are between -1 and 1 with 12 digits of decimal points, we use the mentioned small value.
- **I-Frame Zero Threshold** = 10^{-5}; to be equal to the other threshold.
- **I-Rate** = 2 seconds; Only in section 5.2.

Independent Variables (IVs): The following variables are purposely set and the scheme is executed once per each of them. So each song is compressed 42 ($= 6 \times 3 + 6 \times 4$) times.

- **Audio Type** (6 levels)
  1. **Mixture:** The original song including vocals, drums, and the rest of the instruments.
  2. **Vocals:** Only the vocal sound of the song which normally doesn't have repeating tones.
  3. **Drums:** Only the drums sound of the song which normally repeats with the beat.
  4. **REPET's Foreground:** The masked foreground of the REPET algorithm which is determined based on lack of repetition.

5. **REPET's Background:** The masked background of the REPET algorithm which is determined based on having repetitions.
6. **Sum:** A mixture of the latter two which holds the concatenation of REPET's foreground with only I-Frames and REPET's background with P-Frames.

- **Scheme** (3 levels); Only used in section 5.2.
- **I-Rate** (4 levels); Only used in section 5.1.

Statistical Analysis: For each DV, I provide figures that show the mean of outputs in each one of the IVs ($N = 50$). Error bars in graphs are all showing 95% confidence intervals. In addition to the graphs, descriptive and inferential statistics are provided. Data is analyzed using a 2-way repeated-measures ANOVA and Tukey HSD post-hoc test. For every ANOVA test, the assumptions of Normality and Sphericity of Distributions were checked. Despite some mild violations of Normality, there was no violation of the Sphericity test. This means degrees of freedom never needed to be corrected. As for violations of the Normality test, it is believed that the ANOVA test is robust in groups of the same size.

5.1 Effects of I-Rate on the Compression Scheme

As described in section 4.1, the I-Rate parameter is the rate of forced I-Frames that P-Frames are not allowed to refer to any frame before them. A larger number in the I-Rate means P-Frames are allowed to look more back in time to find a suitable referenced frame to reuse. Additionally, it means more comparisons and more CPU time. In this section, I try to answer the question: What is the suitable amount of time for P-Frames to look back? My hypothesis is that, in the worst case, beats are repeated once per second. Therefore, 2 seconds for the I-Rate should be sufficient and there is no point in going back more in time to find a reference.

To test this hypothesis, DVs are measured as described at the beginning of chapter 5 with an additional 4-level IV named I-Rates: 0.1 of a second, 1 second, 2 seconds, and 5 seconds. The following sections are reports for each measured variable.

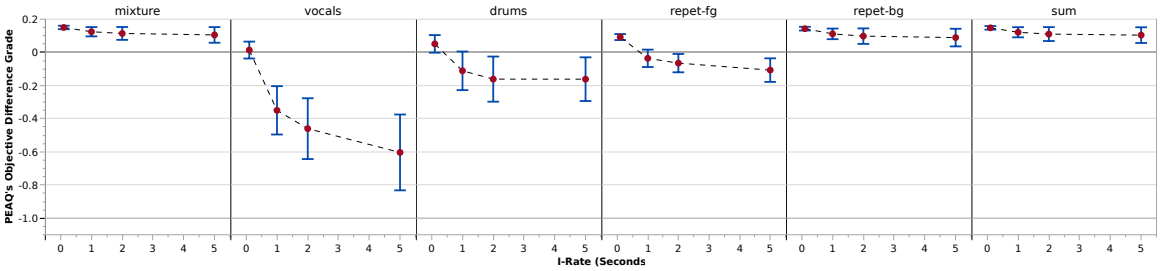


Figure 5.1: Comparison of Audio Quality in different types of audio and I-Rates. Audio Quality is measured using Objective Difference Grade (ODG) in Perceptual Evaluation of Audio Quality (PEAQ) (0 = Imperceptible, -1 = Perceptible, but not annoying)

I-Rates	0.1		1		2		5	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Mixture	.15	.037	.125	.099	.114	.137	.105	.166
Vocals	.014	.18	-.349	.513	-.46	.645	-.603	.805
Drums	.051	.187	-.111	.41	-.161	.479	-.162	.464
REPET's Foreground	.092	.064	-.036	.185	-.065	.194	-.107	.25
REPET's Background	.143	.039	.112	.098	.098	.165	.089	.187
Sum	.148	.038	.121	.107	.111	.148	.104	.167

Table 5.1: Table of Audio Quality in different types of audio and I-Rates. Audio Quality is measured using Objective Difference Grade (ODG) in Perceptual Evaluation of Audio Quality (PEAQ) (0 = Imperceptible, -1 = Perceptible, but not annoying)

5.1.1 Audio Quality

It can be seen in figure 5.1 and table 5.1 that mixture, REPET's background, and the Sum audio inputs are imperceptible in all I-Rates. In drums and REPET's foreground, however, with the increment of I-Rate, the audio quality is slightly reduced. Most noticeably, with the increment of I-Rate, the audio quality in vocals are highly reduced. Within-subject ANOVA showed that the audio quality is significantly affected by the used input audio type ($F(5, 245) = 31.4, p < .000, \eta_p^2 = .39$), the I-Rate ($F(3, 147) = 44.8, p < .000, \eta_p^2 = .477$), and the interaction between them ($F(15, 735) = 18.7, p < .000, \eta_p^2 = .276$). Tukey HSD post-hoc showed that the audio quality is significantly higher in I-Rate of 0.1 of a second ($M = .1, SD = .122$), as compared to other used I-Rates (all ps $< .001$) while it showed no significant reduction of audio quality for the I-Rate of 1 second ($M = -.023, SD = .333$), 2 seconds ($M = -.06, SD = .407$), and 5 seconds ($M = -.096, SD = .479$). In vocals specifically, Tukey HSD showed no significant reduction of audio quality between I-Rates of 1 second and 2 seconds, and I-Rates of 2 seconds and 5 seconds. However, the audio quality between I-Rates of 1 second and 5 seconds are significantly lowered ($p = .011$).

In summary, the results confirmed that, except for vocals, the lossy compression remains imperceptible with any amount of I-Rate and it stabilizes when it is higher than 1 second.

5.1.2 Compression Ratio

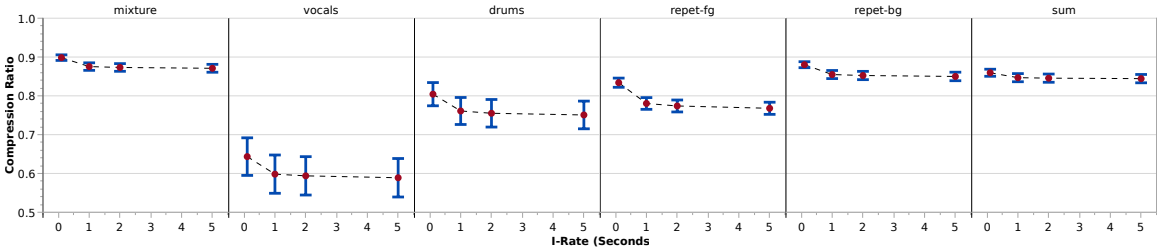


Figure 5.2: Comparison of Compression Ratio in different types of audio and I-Rates

I-Rates	0.1		1		2		5	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Mixture	.898	.025	.875	.034	.873	.035	.871	.036
Vocals	.643	.17	.598	.174	.594	.174	.589	.174
Drums	.804	.105	.761	.122	.755	.125	.751	.126
REPET's Foreground	.834	.042	.78	.053	.774	.054	.768	.055
REPET's Background	.88	.027	.855	.037	.852	.038	.85	.039
Sum	.859	.032	.847	.037	.846	.037	.844	.038

Table 5.2: Table of Compression Ratio in different types of audio and I-Rates

It can be seen in figure 5.2 and table 5.2 that all audio inputs are slightly compressed in I-Rate of 0.1 of a second and then a bit more above one second. Most noticeably, with the increment of I-Rate, vocal audio input is compressed more than the other. Within-subject ANOVA showed that Compression Ratio is significantly affected by the used input audio type ($F(5, 245) = 72.4, p < .000, \eta_p^2 = .596$), the I-Rate ($F(3, 147) = 292, p < .000, \eta_p^2 = .856$), and the interaction between them ($F(15, 735) = 48, p < .000, \eta_p^2 = .495$). Tukey HSD post-hoc showed that the compression is significantly worse in I-Rate of 0.1 of a second ($M = .82, SD = .12$), as compared to other used I-Rates (all ps $< .001$) while it showed no significant improvement of compression for the I-Rate of 1 second ($M = -.786, SD = .131$), 2 seconds ($M = .782, SD = .133$), and 5 seconds ($M = .779, SD = .134$). In vocals specifically, Tukey HSD showed no significant reduction of audio quality in I-Rates of 0.1 of a second, 1 second, 2 seconds and 5 seconds (all ps $> .319$).

In summary, the results confirmed that the Compression Ratio doesn't improve with I-Rates higher than 1 second.

5.1.3 Desirability

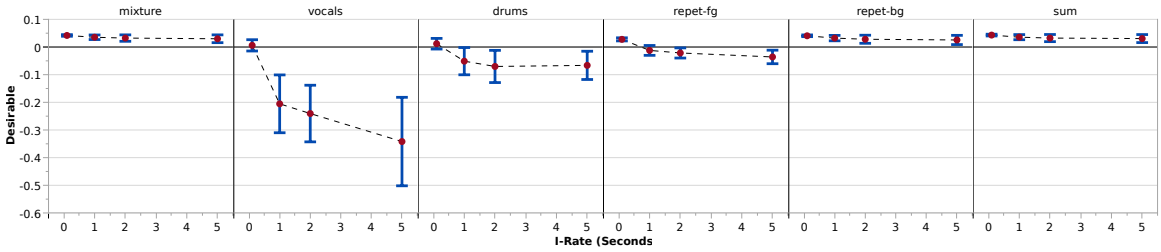


Figure 5.3: Comparison of Desirability in different types of audio and I-Rates

I-Rates	0.1		1		2		5	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Mixture	.042	.01	.035	.029	.032	.041	.03	.05
Vocals	.006	.072	-.205	.368	-.241	.36	-.342	.563
Drums	.012	.068	-.051	.173	-.07	.205	-.066	.18
REPET's Foreground	.028	.02	-.012	.063	-.022	.065	-.036	.086
REPET's Background	.041	.011	.032	.034	.028	.051	.026	.059
Sum	.043	.011	.036	.033	.032	.045	.031	.051

Table 5.3: Table of Desirability in different types of audio and I-Rates

It can be seen in figure 5.3 and table 5.3 that mixture, REPET's background, and the Sum audio inputs are the most desirable. In drums and REPET's foreground audio inputs, the increment of I-Rate makes the result slightly less desirable. Most noticeably, in vocals, the increment of I-Rate makes the algorithm least desirable. Within-subject ANOVA showed that audio quality is significantly affected by the used input audio type ($F(5, 245) = 21.8, p < .000, \eta_p^2 = .308$), the I-Rate ($F(3, 147) = 28.5, p < .000, \eta_p^2 = .368$), and the interaction between them ($F(15, 735) = 12.6, p < .000, \eta_p^2 = .205$). Tukey HSD post-hoc showed that the Desirability is significantly higher in I-Rate of 0.1 of a second ($M = .029, SD = .044$), as compared to other used I-Rates (all ps $< .001$) while it showed no significant reduction of Desirability for the I-Rate of 1 second ($M = -.028, SD = .189$), 2 seconds ($M = -.04, SD = .198$), and 5 seconds ($M = -.06, SD = .278$). In vocals specifically, Tukey HSD showed no significant reduction of Desirability between I-Rates of 1 second and 2 seconds, and I-Rates of 2 seconds and 5 seconds. However, the I-Rate of 5 seconds is significantly less desired than the I-Rate of 1 second ($p = .02$).

In summary, the results confirmed that, except for vocals, the lossy compression remains desirable with any amount of I-Rate and it stabilizes when it is higher than 1 second.

5.1.4 P-Ratio

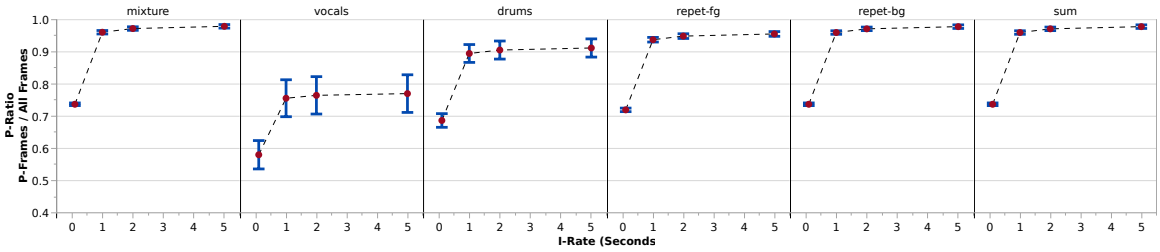


Figure 5.4: Comparison of P-Ratios in different types of audio and I-Rates

I-Rates	0.1		1		2		5	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Mixture	.737	.014	.961	.019	.972	.019	.979	.019
Vocals	.58	.155	.756	.202	.765	.204	.77	.206
Drums	.687	.075	.895	.098	.906	.099	.912	.1
REPET's Foreground	.72	.019	.938	.025	.949	.026	.956	.026
REPET's Background	.737	.015	.96	.019	.972	.019	.978	.02
Sum	.737	.015	.96	.019	.972	.019	.978	.02

Table 5.4: Table of P-Ratios in different types of audio and I-Rates

It can be seen in figure 5.4 and table 5.4 that in mixture, REPET's foreground, REPET's background, and the sum audio inputs, P-Ratio starts with a lower value in 0.1 of a second, and then it increases. In drums audio input, P-Ratio had a similar trend but was slightly lower compared to the former. Most noticeably in vocals, P-Ratio was highly lower compared to the rest. Within-subject ANOVA showed that P-Ratio is significantly affected by the used input audio type ($F(5, 245) = 40.6, p < .000, \eta_p^2 = .453$), the I-Rate ($F(3, 147) = 20263, p < .000, \eta_p^2 = .998$), and the interaction between them ($F(15, 735) = 40.6, p < .000, \eta_p^2 = .453$). Tukey HSD post-hoc showed that P-Ratio is significantly lower in I-Rate of 0.1 of a second ($M = .7, SD = .09$), as compared to other used I-Rates (all ps $< .001$) while it showed no significant improvement for the I-Rate of 1 second ($M = .912, SD = .118$), 2 seconds ($M = .923, SD = .119$), and 5 seconds ($M = .929, SD = .12$). Moreover, Tukey HSD showed that P-Ratio is significantly lower in vocals ($M = .718, SD = .207$), as compared to other used I-Rates (all ps $< .001$) while it showed no significant change

for the mixture ($M = -.912, SD = .103$), drums ($M = .85, SD = .132$), REPET's foreground ($M = .891, SD = .102$), REPET's background ($M = .912, SD = .103$), and the sum ($M = .912, SD = .103$).

In summary, the results confirmed that, except for vocals, the lossy compression can find P-Frames with any amount of I-Rate and it stabilizes when it is higher than 1 second.

5.2 Comparison of Compression Schemas

It can be concluded from section 5.1 that 2 seconds is a suitable value to use for the I-Rate parameter and we only use this constant value for the P-Frame schema. Now it is time to answer the main research question: i.e "Can the described P-Frame schema improve the compression of pieces of music with repeated instrumental tones without adding perceptible differences?". To study the answer, DVs are measured as described at the beginning of chapter 5 with an additional 3-level IV named **Schema**:

1. **Lossless:** The base condition which saves raw MDCT output.
2. **Only I-Frames:** The MDCT output with all I-Frames zero thresholds applied.
3. **With P-Frames:** The proposed new compression scheme.

Following sections are reports for each measured variable.

5.2.1 Audio Quality

Input Type	Mixture		Vocals		Drums		REPET's Foreground		REPET's Background	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Lossless	.164	.009	.151	.015	.149	.03	.15	.016	.159	.009
All I-Frames	.152	.035	.146	.017	.113	.088	.14	.022	.146	.039
With P-Frames	.114	.137	-.46	.645	-.161	.479	-.065	.194	.098	.165

Table 5.5: Table of Schemas in Audio Quality amongst different types of audio. Audio Quality is measured using Objective Difference Grade (ODG) in Perceptual Evaluation of Audio Quality (PEAQ) (0 = Imperceptible, -1 = Perceptible, but not annoying)

It can be seen in figure 5.5 and table 5.5 that lossless encoding has an ODG above zero (imperceptible) and very similar in all input types. Similarly, all I-frame encoding

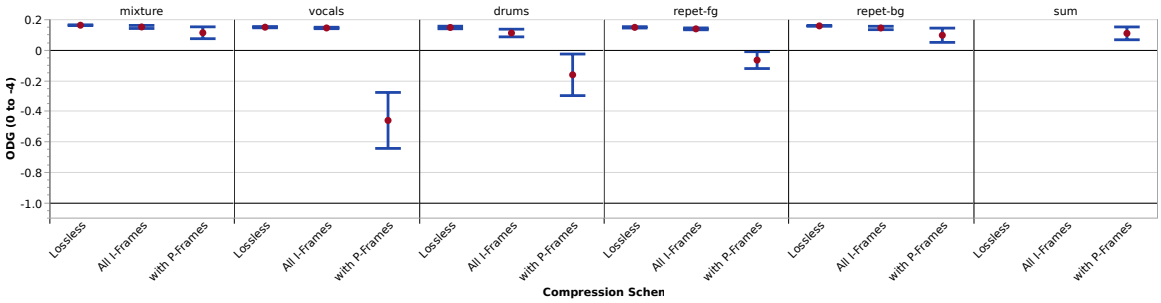


Figure 5.5: Comparison of Schemas in Audio Quality amongst different types of audio. Audio Quality is measured using Objective Difference Grade (ODG) in Perceptual Evaluation of Audio Quality (PEAQ) (0 = Imperceptible, -1 = Perceptible, but not annoying)

is imperceptible with a slight decrease in audio quality in all input types. However, with P-frame encoding can have an ODG below zero depending on the audio type with even more decrease in all input types.

Within-subject ANOVA showed that the audio quality is significantly affected by the type of audio ($F(4, 196) = 24.4, p < .000, \eta_p^2 = .332$), the used schema ($F(2, 98) = 56.4, p < .000, \eta_p^2 = .535$), and the interaction between them ($F(8, 392) = 25.1, p < .000, \eta_p^2 = .339$). Tukey HSD post-hoc showed that the audio quality for vocals ($M = -.054, SD = .469$) was significantly lower than mixture ($M = .144, SD = .083$), drums ($M = .033, SD = .313$), REPET's foreground ($M = .075, SD = .15$), and REPET's background ($M = .134, SD = .1$), (all ps $< .007$). Beside that, the audio quality for drums was significantly lower than the mixture, and REPET's background (all ps $< .001$). It is important to mention that Tukey HSD showed no significant reduction of audio quality for having P-frames ($M = -.06, SD = .407$), all I-frames ($M = .139, SD = .049$), and lossless ($M = .155, SD = .018$).

In summary, the results confirmed that having P-Frames can significantly reduce the audio quality on vocals, and not the background sounds such as drums. However, it might be confusing why mixtures have a better sound quality with P-Frames scheme as compared to drums. This can probably be because drums audio type contains more silence and human ears can perceive differences in silence better as compared to the mixture with many accompanying instruments.

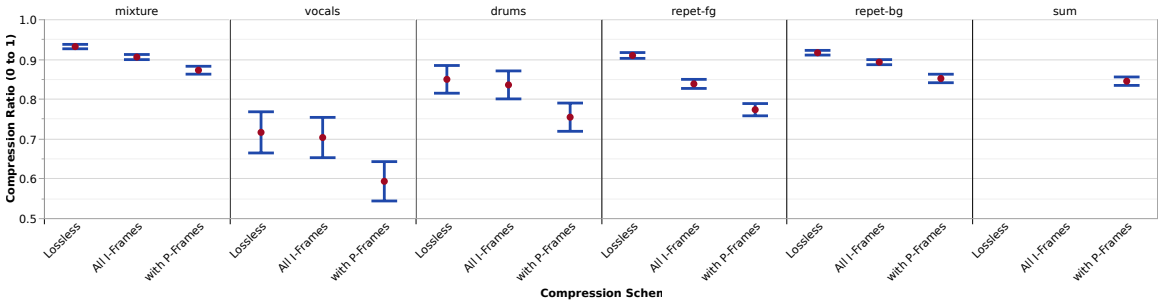


Figure 5.6: Comparison of Schemas in Compression Ratio amongst different types of audio

Input Type	Mixture		Vocals		Drums		REPET's Foreground		REPET's Background	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Lossless	.993	.019	.717	.182	.85	.123	.91	.026	.917	.021
All I-Frames	.906	.023	.704	.178	.836	.124	.839	.04	.893	.021
With P-Frames	.873	.035	.594	.174	.755	.125	.774	.054	.893	.022

Table 5.6: Table of Schemas in Compression Ratio amongst different types of audio

5.2.2 Compression Ratio

It can be seen in figure 5.6 and table 5.6 that lossless encoding has a very similar compression ratio in the mixture, REPET's foreground, and REPET's background. However, lossless encoding had a lower compression ratio in the vocals, and drums. Compared to the lossless encoding, fully I-frame encoding reduced the compression ratio in all input types. Having P-frames reduced the compression ratio even further in all input types. Within-subject ANOVA showed that compression ratio is significantly affected by the used input audio type ($F(4, 196) = 51.8, p < .000, \eta_p^2 = .514$), the used schema ($F(2, 98) = 164, p < .000, \eta_p^2 = .77$), and the interaction between them ($F(8, 392) = 20.3, p < .000, \eta_p^2 = .293$). Tukey HSD post-hoc showed that the compression ratio with P-frame encoding ($M = -.782, SD = .133$) was significantly lower than the compression ratio for fully I-frame encoding ($M = .836, SD = .122, p < .0001$), while the fully I-frame encoding was not significantly lower than the lossless encoding ($M = .033, SD = .313, p = 1$). As for audio inputs, Tukey HSD post-hoc showed that the compression ratio for vocals ($M = .672, SD = .185$) was significantly lower than the compression ratio for mixture ($M = .904, SD = .036$), drums ($M = .814, SD = .13$), REPET's foreground ($M = .841, SD = .07$), and REPET's background ($M = .888, SD = .039$). Beside that, the compression ratio for drums was significantly lower than the compression ratio for mixture, REPET's

foreground, and REPET's background (all ps $< .0005$). However, the compression ratio for drums was not significantly different than REPET's foreground ($p = 154$). Similarly, the compression ratio for the mixture was not significantly different than the compression ratio for REPET's background ($p = 696$).

In summary, the results confirmed that having P-Frames can significantly reduce the compression ratio. Furthermore, the compression ratio is highly dependent on the type of input audio. Specifically, having more silence in the audio input will reduce the compression ratio. This explains the reason that vocals and drums audio inputs have better compression ratios.

5.2.3 Desirability

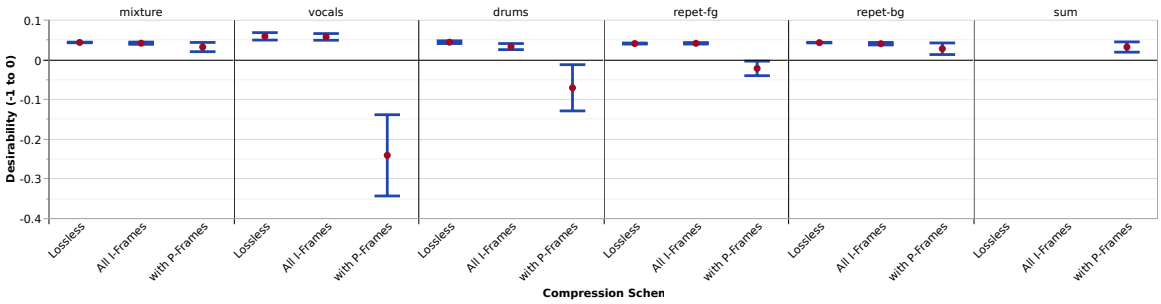


Figure 5.7: Comparison of Compression Schemas in Desirability amongst different types of audio

Input Type	Mixture		Vocals		Drums		REPET's Foreground		REPET's Background	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Lossless	.044	.003	.059	.033	.045	.012	.041	.005	.043	.003
All I-Frames	.042	.01	.058	.03	.034	.027	.042	.007	.041	.011
With P-Frames	.032	.041	-.241	.36	-.07	.205	-.022	.065	.028	.051

Table 5.7: Table of Compression Schemas in Desirability amongst different types of audio

It can be seen in figure 5.7 and table 5.7 that desirability is not much being reduced in mixture and REPET's background audio inputs with all compression schemes. However, with P-frame encoding, desirability is slightly reduced in drums and REPET's foreground audio and it is reduced the most in vocals audio input. Within-subject ANOVA showed that desirability is significantly affected by the used input audio type ($F(4, 196) = 16.3, p < .000, \eta_p^2 = .25$), the used schema ($F(2, 98) =$

$46.7, p < .000, \eta_p^2 = .488$), and the interaction between them ($F(8, 392) = 21.9, p < .000, \eta_p^2 = .309$). Tukey HSD post-hoc showed that the desirability was significantly reduced in vocals with P-frame encoding and in drums with P-frame encoding compared to all other cases (all ps $< .0008$). However, the desirability of having P-frames is not significantly reduced in the mixture, REPET's foreground, and REPET's background.

In summary, the results confirmed that having P-frames can be significantly undesirable in vocals. Having P-frames can also significantly reduce desirability in drums but the mean difference shows it is not as undesirable as it is for vocals.

5.3 Conclusions of Experiments

To summarize, it can be concluded from section 5.1 that 2 seconds is a suitable value to use for the I-Rate parameter for the P-Frame schema. Afterwards, from section 5.2 it can be concluded that having P-Frames can significantly reduce the Audio Quality and Desirability on vocal tracks, while the background sound is not affected as much. Furthermore, having P-Frames can have a significant improvement effect on the compression ratio by 3 to 8 percent.

Chapter 6

Conclusions and Future Work

Virtual Music Performances can benefit from domain-specific audio compressions. One approach to improve audio compression is to leverage the repeating aspect of musical beats, vamps, rhythms, and decays in one single musical note. In this thesis, after providing some background, a simple implementation of perceptual audio compression is described that was used for the exploration of ideas. Then, inspired by the MPEG-1 video encoder, the idea of using I-Frame and P-Frames over the MDCT of audio signals is described, implemented, and tested in one example. And lastly, an experimental methodology is designed using the MUSDB-HQ dataset and results were reported with descriptive and inferential statistical analysis.

The proposed audio compression scheme shows promising results on mixture and background music in which musical rhythms can be found but for a vocal track, it is rather harmful. Even though these results were found before I read about Cunningham et al. at the time of writing this thesis, their findings confirm the experimental results of this thesis. We both found improvements in compression ratio and similar effects on vocal sounds. To find similarities, they used the resource heavy similarity matrix and sliding windows over FFT, while I used MDCT column comparison over a fixed I-Rate limit. Additionally, they tested their approach on a non-public selected audio, while I used a public library with an experimental methodology with statistical analysis to show generalizability. On a big contrast, my work was not compared with actual MP3 and AAC encoders, but rather it was compared to a simplified implementation while all other lossy techniques were disabled. So because the idea was implemented with the Python programming language, the processing time was not analyzed or compared. Therefore, for future work, I suggest that the proposed method is implemented using the C++ programming language, all lossy techniques

for perceptual audio compression are applied, and then it would be compared with both MP3 and AAC encoders in audio quality, compression ratio, and latency. This can allow us to see if the idea is beneficial for a real NMP scenario.

One way that the proposed P-Frame scheme can shine is in digital mixers and drum machines. These devices normally sum the repeating content with some randomized numbers to make the generated audio more natural to the human ears. By separating and saving these repeating content in I-Frames and randomized numbers in P-Frames, the proposed scheme seems to have a great potential to be integrated and used in these devices. This can potentially add popularity to these devices as well as the proposed compression scheme because the former can support audio compression as a built-in feature and the latter can achieve a better compression ratio by eliminating the need for estimating suitable P-Frames and references. In other words, there would be no need to have the encoder algorithm implemented in these devices and instead, they can be programmed to generate their output directly in the P-Frame scheme format which can then be converted into the PCM wave signals by the defined decoder algorithm.

Another direction for future work stems from the finding that P-Frames are mostly referenced to their closes first or second frames (see figure 4.5). This probably means there is more similarity between frames that contain the continuous or decaying signal of the same note as compared to another note that is being played at a different time. So I suggest the P-Frame be modified in favour of this decaying signal. One possible modification can be to save the ratio of the absolute sum of two frames ($r = |F_2|/|F_1|$) and use the ratio when calculating the difference between two frames ($P_2 = F_2 - rF_1$). This might cause the P-Frame to contain more zero elements after the threshold. The list of all ratios are needed to be stored in the encoded stream alongside the list of references, but overall I hypothesize that it would improve the compression ratio of the encoded bitstream.

Bibliography

[1] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. https://doi.org/10.1109/T-C.1974.223784.

[2] Christopher Bartlette, Dave Headlam, Mark Bocko, and Gordana Velikic. Effect of network latency on interactive musical performance. *Music Perception*, 24(1):49–62, 2006.

[3] marina bosi, karlheinz brandenburg, schuyler quackenbush, louis fielder, kenzo akagiri, hendrik fuchs, and martin dietz. ISO/IEC MPEG-2 Advanced Audio Coding. *journal of the audio engineering society*, 45(10):789–814, october 1997.

[4] Den Brinker, C. Albertus, Erik Schuijers, and Werner Oomen. Parametric coding for high-quality audio. *Journal of the audio engineering society*, april 2002.

[5] Michael Burrows and David J Wheeler. A block-sorting lossless data compression algorithm. *Citeseer*, 1994.

[6] Speedtest© by Ookla©. Speedtest Global Index – Internet Speed around the world. https://www.speedtest.net/global-index. Online; Accessed: 2022-06-27.

[7] Alexander Carôt and Christian Werner. Fundamentals and principles of musical telepresence. *Journal of Science and Technology of the Arts*, 1(1):26–37, 2009.

[8] Chris Chafe, Juan-Pablo Caceres, and Michael Gurevich. Effect of temporal separation on synchronization in rhythmic performance. *Perception*, 39(7):982–992, 2010.

[9] Chris Chafe, Michael Gurevich, Grace Leslie, and Sean Tyan. Effect of time delay on ensemble accuracy. In *Proceedings of the international symposium on musical acoustics*, volume 31, page 46. ISMA Nara, 2004.

[10] Josh Coalson and Xiph.Org Foundation. FLAC - documentation. `https://xiph.org/flac/documentation_format_overview.html`, 2001. Online; Accessed: 2022-06-30.

[11] Federal Communications Commission. Broadband Speed Guide. `https://www.fcc.gov/consumers/guides/broadband-speed-guide`, August 2011. Online; Accessed: 2022-06-27.

[12] Stuart Cunningham and Vic Grout. Advances in similarity-based audio compression. In *SEIN 2007: Proceedings of the Third Collaborative Research Symposium on Security, E-Learning, Internet and Networking*, page 129. Lulu. com, 2007.

[13] Stuart Cunningham and Vic Grout. Audio compression exploiting repetition (acer): Challenges and solutions. 2009.

[14] Stuart Cunningham and Vic Grout. Data reduction of audio by exploiting musical repetition. *Multimedia tools and applications*, 72(3):2299–2320, 2014.

[15] Stuart Cunningham, Vic Grout, and John McGinn. Play it again, babbage!–a framework to exploit musical repetition for high-quality audio compression. In *Proceedings of IADIS-International Conference on WWW/Internet, Lisbon, Portugal, 19th-22nd October*, 2005.

[16] Stuart Cunningham and Iain McGregor. Subjective evaluation of music compressed with the acer codec compared to aac, mp3, and uncompressed pcm. *International Journal of Digital Multimedia Broadcasting*, 2019, 2019.

[17] Stuart Cunningham, Jonathan Weinel, Shaun Roberts, Vic Grout, and Darryl Griffiths. Initial objective & subjective evaluation of a similarity-based audio compression technique. In *Proceedings of the 8th Audio Mostly Conference*, pages 1–6, 2013.

[18] L. Peter Deutsch. DEFLATE Compressed Data Format Specification version 1.3. Technical Report 1951, RFC Editor, May 1996. `https://www.rfc-editor.org/info/rfc1951`.

[19] Martin Dietz, Lars Liljeryd, Kristofer Kjorling, and Oliver Kunz. Spectral band replication, a novel approach in audio coding. In *Audio Engineering Society Convention 112*. Audio Engineering Society, 2002.

[20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. `https://doi.org/10.1038/s41586-020-2649-2`.

[21] Rainer Huber and Birger Kollmeier. PEMO-Q—A new method for objective audio quality assessment using a model of auditory perception. *IEEE Transactions on audio, speech, and language processing*, 14(6):1902–1911, 2006.

[22] David A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. `https://doi.org/10.1109/JRPROC.1952.273898`.

[23] Igor Pavlov. LZMA SDK (Software Development Kit). `https://www.7-zip.org/sdk.html`, 1998. Online; Accessed: 2022-06-30.

[24] ISO/IEC JTC 1/SC 29. Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 2: Video. Standard 35.040.40, International Organization for Standardization, August 1993.

[25] ISO/IEC JTC 1/SC 29. Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 3: Audio. Standard 35.040.40, International Organization for Standardization, August 1993.

[26] ISO/IEC JTC 1/SC 29. Information technology — Generic coding of moving pictures and associated audio information — Part 3: Audio. Standard 35.040.40, International Organization for Standardization, May 1995.

[27] ISO/IEC JTC 1/SC 29. Information technology — Coding of audio-visual objects — Part 3: Audio. Standard 35.040.40, International Organization for Standardization, December 2001.

[28] ITU-R. 1387: Method for objective measurements of perceived audio quality (PEAQ). *International Telecommunication Union*, 1387, 2001.

[29] ITU-R. 1534-1: Method for the subjective assessment of intermediate quality levels of coding systems (MUSHRA). *International Telecommunication Union*, 2003.

[30] ITU-T. P.861 : Objective quality measurement of telephone-band (300-3400 Hz) speech codecs. *International Telecommunication Union*, page 34, 1996.

[31] ITU-T. P.862 : Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. *International Telecommunication Union*, 2001.

[32] Michael Jackson. Billie Jean, Thriller Album. `https://www.youtube.com/watch?v=OZGtRvYF-A4`, 1982. YouTube Video; Accessed: 2022-06-20.

[33] Peter Kabal et al. An examination and interpretation of ITU-R BS. 1387: Perceptual evaluation of audio quality. Technical report, TSP Lab, Dept. Electrical & Computer Engineering, McGill University, 2002.

[34] Daniel Lee and H. Sebastian Seung. Algorithms for Non-negative Matrix Factorization. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000.

[35] Tilman Liebchen, Takehiro Moriya, Noboru Harada, Yutaka Kamamoto, and Yuriy A Reznik. The MPEG-4 Audio Lossless Coding (ALS) Standard - Technology and Applications. *New York*, page 14, 2005.

[36] Matin Lotfaliei. MSc Thesis: A New Audio Compression Scheme that Leverages Repetition in Music. `https://github.com/matinlotfali/audio-compression-repetition-music`, 2022.

[37] Manfred Lutzky, Gerald Schuller, Marc Gayer, Ulrich Krämer, and Stefan Wabnik. A guideline to audio codec delay. In *Audio Engineering Society Convention 116*. Audio Engineering Society, 2004.

[38] Ethan Manilow, Prem Seetharaman, and Bryan Pardo. The northwestern university source separation library. Proceedings of the 19th International Society of Music Information Retrieval Conference (ISMIR 2018), Paris, France, September 23-27, 2018.

[39] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, 2015.

[40] David Meares, Kaoru Watanabe, and Eric Scheirer. Report on the MPEG-2 AAC Stereo Verification Tests. Technical report, International Organization for Standardization, February 1998.

[41] Christopher Montgomery. next generation video: Introducing Daala. `https://people.xiph.org/~xiphmont/demo/daala/demo1.shtml`, June 2013. Online; Accessed: 2022-06-12.

[42] J. Princen and A. Bradley. Analysis/synthesis filter bank design based on time domain aliasing cancellation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1153–1161, 1986. `https://doi.org/10.1109/TASSP.1986.1164954`.

[43] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. MUSDB18-HQ - an uncompressed version of musdb18, December 2019. `https://doi.org/10.5281/zenodo.3338373`.

[44] Zafar Rafii and Bryan Pardo. A simple music/voice separation method based on the extraction of the repeating musical structure. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 221–224. IEEE, 2011.

[45] Zafar Rafii and Bryan Pardo. Music/Voice Separation Using the Similarity Matrix. In *ISMIR*, pages 583–588, 2012.

[46] Zafar Rafii and Bryan Pardo. Online repet-sim for real-time speech enhancement. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 848–852. IEEE, 2013.

[47] Zafar Rafii and Bryan Pardo. REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(1):73–84, 2013. https://doi.org/10.1109/TASL.2012.2213249.

[48] Emmanuel Ravelli, Gal Richard, and Laurent Daudet. Union of MDCT Bases for Audio Coding. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(8):1361–1372, November 2008. Conference Name: IEEE Transactions on Audio, Speech, and Language Processing. https://doi.org/10.1109/TASL.2008.2004290.

[49] Emmanuel Ravelli, Gaël Richard, and Laurent Daudet. Audio Signal Representations for Indexing in the Transform Domain. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):434–446, March 2010. Conference Name: IEEE Transactions on Audio, Speech, and Language Processing. https://doi.org/10.1109/TASL.2009.2025099.

[50] Jorma J Rissanen. Generalized kraft inequality and arithmetic coding. *IBM Journal of research and development*, 20(3):198–203, 1976.

[51] A.H. Robinson and C. Cherry. Results of a prototype television bandwidth compression scheme. *Proceedings of the IEEE*, 55(3):356–364, 1967. https://doi.org/10.1109/PROC.1967.5493.

[52] Cristina Rottondi, Chris Chafe, Claudio Allocchio, and Augusto Sarti. An overview on networked music performance technologies. *IEEE Access*, 4:8823–8843, 2016.

[53] Gerald DT Schuller, Bin Yu, Dawei Huang, and Bernd Edler. Perceptual audio coding using adaptive pre-and post-filters and lossless compression. *IEEE Transactions on Speech and Audio Processing*, 10(6):379–390, 2002.

[54] Prem Seetharaman, Fatemeh Pishdadian, and Bryan Pardo. Music/Voice separation using the 2D fourier transform. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 36–40, October 2017. ISSN: 1947-1629, https://doi.org/10.1109/WASPAA.2017.8169990.

[55] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[56] D. Sinha and J.D. Johnston. Audio compression at low bit rates using a signal adaptive switched filterbank. *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, 2:1053–1056 vol. 2, May 1996. ISSN: 1520-6149, https://doi.org/10.1109/ICASSP.1996.543544.

[57] Paris Smaragdis. Discovering auditory objects through non-negativity constraints. In *ISCA Tutorial and Research Workshop (ITRW) on Statistical and Perceptual Audio Processing*, 2004.

[58] Paris Smaragdis. Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs. In *International Conference on Independent Component Analysis and Signal Separation*, pages 494–499. Springer, 2004.

[59] Carlos Tarjano and Valdecy Pereira. An efficient algorithm for segmenting quasi-periodic digital signals into pseudo cycles: Application in lossy audio compression. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1694–1703, 2022.

[60] Jean-Marc Valin, Gregory Maxwell, Timothy B Terriberry, and Koen Vos. High-quality, low-delay music coding in the opus codec. *arXiv preprint arXiv:1602.04845*, 2016.

[61] Jean-Marc Valin, Timothy B Terriberry, and Gregory Maxwell. A full-bandwidth audio codec with low complexity and very low delay. In *2009 17th European Signal Processing Conference*, pages 1254–1258. IEEE, 2009.

[62] Shankar Vembu and Stephan Baumann. Separation of vocals from polyphonic audio recordings. In *ISMIR*, pages 337–344. Citeseer, 2005.

[63] Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984. https://doi.org/10.1109/MC.1984.1659158.

[64] Nils Werner. MDCT Python Package. http://github.com/nils-werner/mdct, 2020. GitHub Repository; commit: 3eb8915.

[65] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977. https://doi.org/10.1109/TIT.1977.1055714.

[66] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978. https://doi.org/10.1109/TIT.1978.1055934.