EXAMPLE-BASED AUDIO EDITING

BY

RAMIN ANUSHIRAVANI

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Adviser:

Associate Professor Paris Smaragdis

ABSTRACT

Traditionally, audio recordings are edited through digital audio workstations (DAWs), which give users access to different tools and parameters through a graphical user interface (GUI) without prior knowledge in coding or signal processing. The complexity of working with DAWs and the undeniable need for strong listening skills have made audio editing unpopular among novice users and time consuming for professionals. We propose an intelligent audio editor (EBAE) that automates major audio editing routines with the use of an example sound and efficiently provides users with high-quality results. EBAE first extracts meaningful information from an example sound that already contains the desired effects and then applies them to a desired recording by employing signal processing and machine learning techniques.

To my Mom, Dad,
Amir, Omid,
and Pishi
for being there for me.

ACKNOWLEDGMENTS

I would like to express my deep appreciation to my adviser, Prof. Smaragdis, for his continuous support of my master's study and research, patience, and immense knowledge in the world of audio processing. Without his guidance, the completion of my master's degree would not have been possible. I am grateful for the opportunity to learn from one of the brightest audio scientists in the world over the past two years.

My sincere thanks also goes to Prof. Douglas Jones, who provided me with advice and unconditional support throughout my undergraduate and graduate studies.

I would like to also thank my fellow lab-mates in the computational audio lab at the University of Illinois at Urbana-Champaign (U of I) for interesting discussions. In particular, I am grateful to Prof. Minje Kim for being a good friend and providing me with guidance and support.

I would also like to thank the National Science Foundation for supporting this work (Grant No. 1451380).

I would like to say a big thank you to my friends Charles Larson and Peggy Olson for being there in hard times and supporting me throughout my life in Urbana-Champaign.

Last, but not least, I would also like to thank my parents and two brothers for supporting me daily, from tens of thousands of miles away, throughout my study at the U of I.

TABLE OF CONTENTS

CHAPTER 1
EXAMPLE-BASED AUDIO EDITING

1.1 Introduction

This chapter motivates example-based audio editing (EBAE) with numerous examples and analogies. In Chapter 2, we review common audio signal processing tools that are frequently used in this thesis. In Chapters 3, 4, and 5, we discuss background work on equalization, noise, and reverberation as well as proposing the example-based editing scheme for each, respectively. Each chapter begins by reviewing the classical and the state-of-the-art algorithms in enhancing audio recordings in the mentioned criteria. We then discuss how those techniques can be modified and applied for the proposed example-based editing. In Chapter 6, details of a user study that was conducted for evaluating the example-based audio editor subjectively is discussed. In Chapter 7, we conclude this thesis and discuss possible ways for continuing and improving this work.

1.2 What Is Example-Based Editing?

Taking pictures and recording sounds have never been easier. Nowadays, most people are equipped with smart phones, tablets, and various wearable technologies that can perform specific tasks very efficiently. For example, most smart phones have options for applying different types of image filters to any photo that has been taken or saved on the device. These filters are usually depicted on other pictures as a demo, along with a reference picture that has no filter applied to it (see Fig. 1.1). The user does not need to understand the signal processing behind designing the image filters or have man of steel's eyes to find a filter that creates the desired effect. The user

can select a filter by simply trying out different filters and comparing the result with the example reference picture. What if the device or the App does not have the desired filter? What if there is another picture that has the desired filter embedded within it? Is there a way to extract that filter from this picture (i.e. the user provided example picture) and use it to induce the desired filter into the input picture? Can we do that efficiently and in a user-friendly manner?

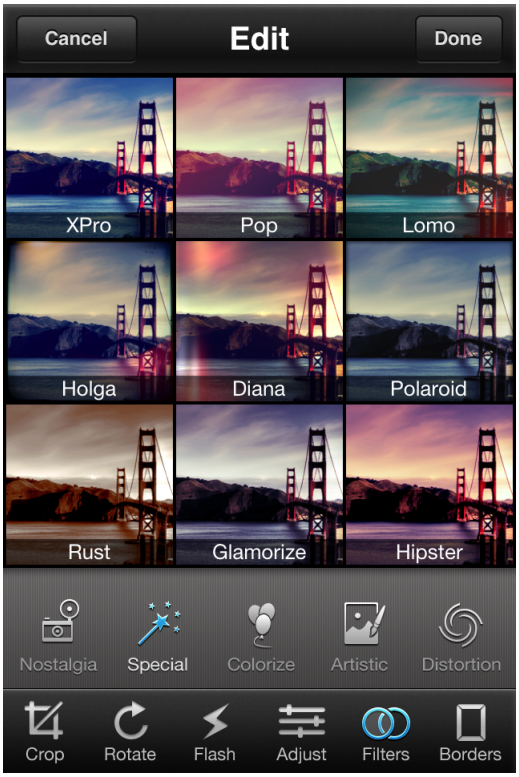

Figure 1.1: An image editing App on iPhone 3 and later models [1].

Yes, there is a way to do this efficiently and accurately with an example-based editing interface. Here is how an example-based image editing could work. The user provides the system with two images: an input image and an example image. The example-based editor decomposes the filtered/wet images into clean/dry images (i.e. the image before it was filtered) and their corresponding filters. The example-based editor then mixes the example filter with the clean input image. This procedure is described in Fig. 1.2 where the user gives the editor a regular image as the input image and a blurry image as the example image. The blurry effect in the example image is the desired filter. The desired output image would be the input image blurred

with the example picture blurry effect.

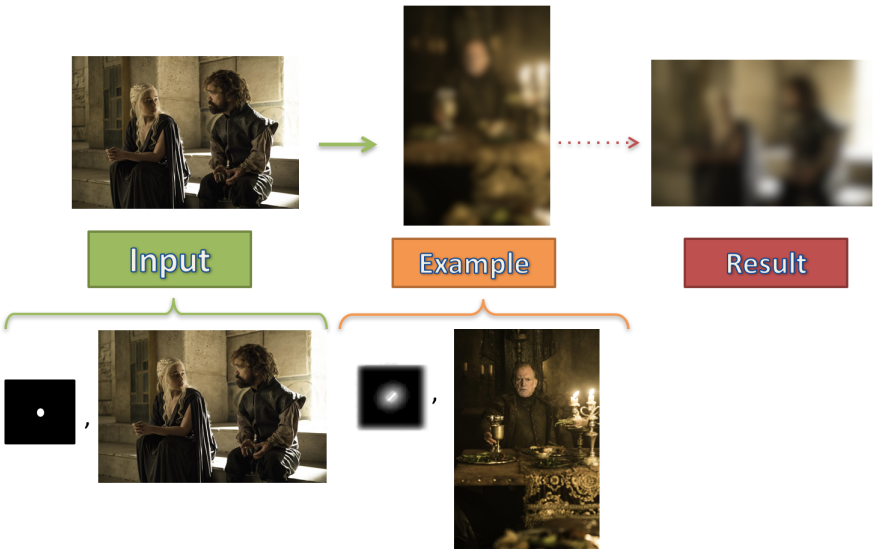

Figure 1.2: Matching the blurry effect of the input image to that in an example image [2].

Now, replace images with audio recordings and the blurry effect with reverberation and you have an example-based audio editor (EBAE).

1.3 Acoustics Matching

An idea similar to example-based audio editing is matching the acoustics of multiple recordings. The goal of an acoustic matching system is to match the acoustics of an input recording to that in an example sound. That is, you place an input recording in the acoustical conditions of an example recording. In essence, you modify the input recording as if it were recorded with the same microphone and equalizer settings used to record the example sound, and under the same background noise and room acoustics.

To motivate the idea behind acoustics matching, consider the following example: Pocasters [3], documentary makers, etc. usually combine multiple recordings which are made in different environments and with different devices. When stitching these recordings into one piece, the change in reverberation, equalization, and background noise from one recording to another becomes more noticeable. This inconsistency can be distracting to the listeners. Manually matching the reverberation, equalization, and noise is humanly

impossible. Our proposed system allows a recording creator to choose one of the recordings as an example sound and automatically match the equalization, noise, and reverberation characteristics of all the other recordings.

Acoustics matching can be used to match the acoustics of a recording to an example sound with a click of a button. The advantages of such a system is that it is very efficient and the need for prior knowledge is minimal.

Consider another scenario for an acoustics matching system in the movie production scheme. Sounds from two consecutive scenes could sound very different [4]. Usually, big movies have enough budget to hire professional audio editors to edit each recording to create consistent, intelligible, and appropriate recordings for each scene. The editing process, however, is very time consuming and unavailable to independent productions and novice users who do not have years of experience in editing audio recordings. An acoustics matching system could help these users to produce acoustically consistent sounds efficiently and accurately. An acoustic matching system can also provide a good starting point for professional audio editors. Figure 1.3 depicts a block diagram for the proposed acoustic matching system. Our proposed acoustic matching system uses the techniques in EBAE to first decompose the sounds into dry sounds and the corresponding equalization, noise, and reverberation effects. The proposed acoustic matching system then transforms these effects on the input sounds to match those in the example sounds. Finally, an input sound is constructed using the example sound effect and the dry input sound.

1.4 Digital Audio Workstations

Consider the graphical equalizers commonly found in Digital Audio Workstations (DAWs) like Adobe Audition, iZotope, Garageband, Reaper, and Audacity as well as home audio systems. As shown in Fig. 1.4, a graphic equalizer consists of a set bars at selected frequencies. The user can adjust the gain for each bar by moving the bar up and down. The gain is in decibels, dB, as shown in Eq. 1.1.

$$\text{Gain}_{\text{dB}} = 20 * log10(\frac{V}{V_0}) \tag{1.1}$$

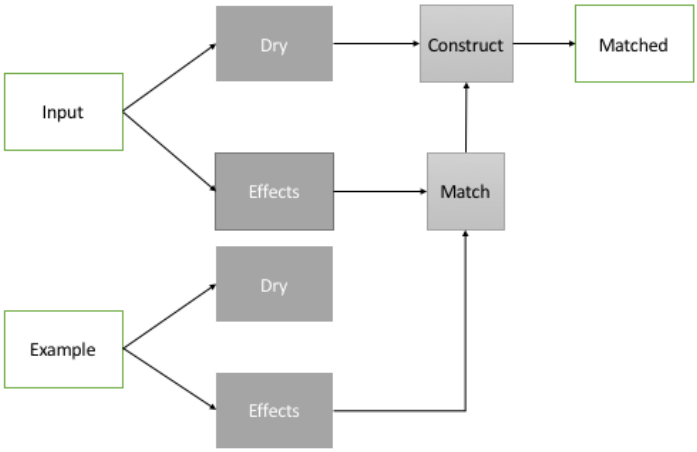

Figure 1.3: Block diagram for the proposed acoustic matching system. A dry sound denotes a recording that has no effects.

where V is the voltage being measured at a microphone which is related to the sound pressure level captured at the microphone, and V_0 is a reference voltage defined for a specific standard (the reference voltage is assumed to be 1 in this thesis).

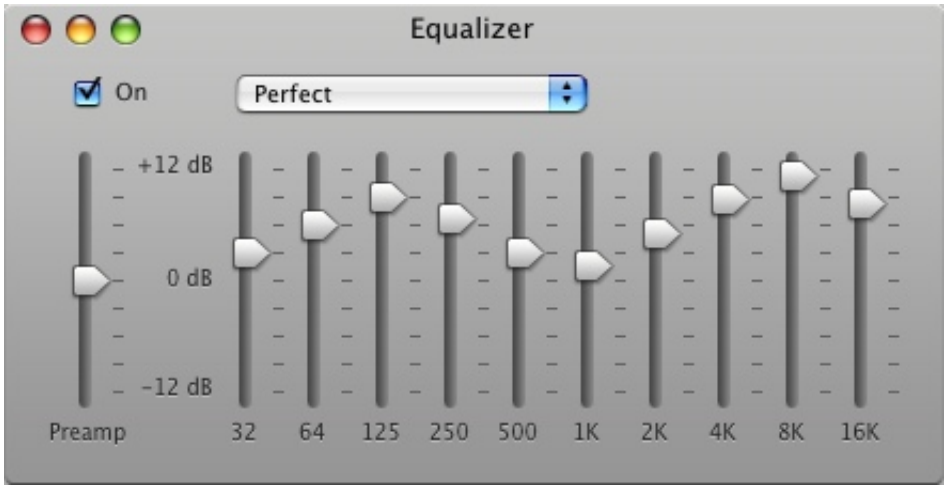

Figure 1.4: The graphic equalizer in GarageBand [5].

The graphical equalizer, one of simplest toolbox found in DAWs, makes a number of assumptions about the user's background. The user needs to be able to detect sounds at different gains and frequencies, either by hearing

it or by analyzing the sound using another toolbox. In order to achieve an appropriate equalizer, it is necessary to understand the exact gain that needs to be adjusted at each selected frequency. In Chapter 6, we discussed that most users struggle with adjusting the equalizer settings even with undoing simple digital filters. Example-based editing can match the equalizer settings of an input sound more efficiently and accurately, given an example sound that has the desired equalizer settings already contained.

In additions to editing the equalization of a recording, other areas of audio editing such as noise and reverberation can also be greatly simplified using the proposed example-based editing system.

1.4.1 How Does Example-Based Audio Editing Work?

Example-based audio editing has three stages:

1. Loading input and example sounds
2. Extracting features (i.e. effects)
3. Matching features

EBAE requires two recordings: an input sound and an example sound. Once the recordings are given to the system, the user can choose one of the available matching schemes (e.g. equalization, noise, and reverberation) to edit the input recording. EBAE prepares the recordings automatically (e.g. voice activity detection, normalizing the gain, click removal) and then extracts corresponding features. The input sound is then modified accordingly to re-create the example feature in the input sound. The user can then hear the final matched sound concatenated with the example sound and choose a different example sound until the desired effect is achieved.

CHAPTER 2
FOUNDATION

2.1 Introduction

This chapter covers some background information on signal processing in a more intuitive manner. This chapter reviews techniques for visualizing sounds in different domains, signal processing concepts such as linear time-invariant (LTI) systems, convolution, aliasing, filtering, and discrete Fourier transforms.

2.2 Visualizing Sound

The first step in analyzing any signal is to visualize it. Sound pressure is a continuous quantity that needs to be sampled by an analog-to-digital converter (ADC) before it can be manipulated on a computer. Sampling is an important step in collecting sound samples and if it is not done correctly the recording could be damage permanently. Waveform is perhaps the most common way of depicting audio recordings digitally. Waveforms represent audio as sound pressure amplitudes sampled in time. Time-domain representations like waveforms are hard to interpret because they do not convey meaningful information about the signal. Consider Fig. 2.1. Both waveforms represent the same speech recordings of "I do!", but they look different.

2.3 LTI Systems

Engineers usually employ LTI systems to model physical systems. An LTI system consists of an input signal, an impulse response, and an output signal. LTI systems have attractive qualities that make them easy to work with, as

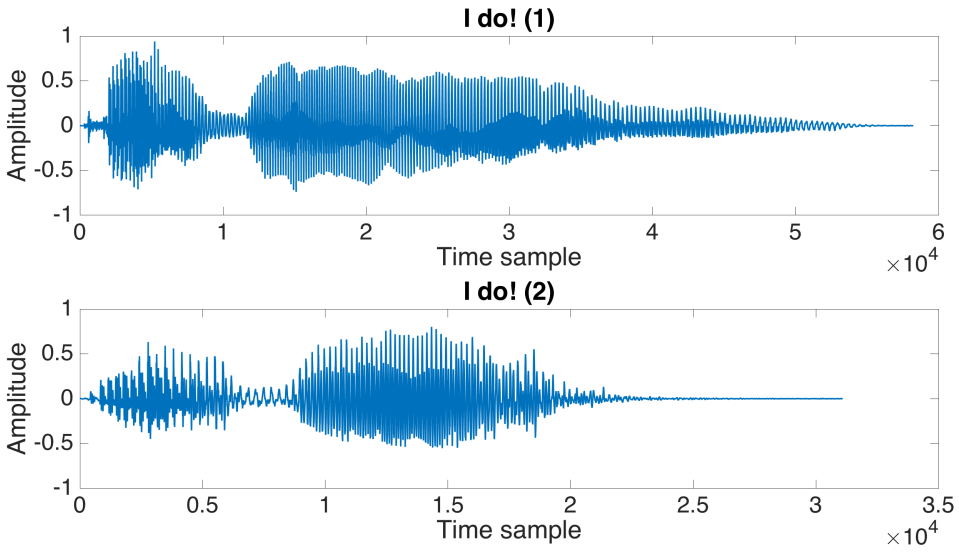

Figure 2.1: These waveforms corresponds to two recordings of "I do!".

described in Eq. 2.1.

$$\begin{aligned} H\{x_1[n]\} &= y_1[n] \\ H\{x_2[n]\} &= y_2[n] \\ H\{a \cdot x_1[n]\} &= a \cdot y_1[n] \\ H\{a \cdot x_1[n] + b \cdot x_2[n]\} &= a \cdot y_1[n] + b \cdot y_2[n] \end{aligned} \tag{2.1}$$

where x, y, and $H\{.\}$ correspond to the input, output, and the LTI system response (i.e. impulse response), respectively.

A system is linear if it meets the following criterion:

$$H\{x_1[n] + x_2[n]\} = y_1[n] + y_2[n] \tag{2.2}$$

A system is time-invariant if delaying the input signal delays the output signal by the same number of time samples, as shown in Eq. 2.3.

$$H\{x[n-r]\} = y[n-r] \tag{2.3}$$

For the rest of this thesis, we assume an LTI system can describe the proposed models unless otherwise noted.

2.4 Convolution

Convolution is perhaps the most important concept in signal processing. We can predict the output to an LTI system by analyzing its impulse response using convolution. Convolution is defined as follows:

$$\begin{aligned} y[n] &= \sum_{k=0}^{L} x[n]h[n-k] \\ y &= x * h \end{aligned} \tag{2.4}$$

where x is the input, h is the impulse response, y is the output, n is the time-domain sample, and k is a time shift sample. For convenience, we denote convolution as $*$.

When convolving an input signal with an impulse response, the input signal is copied at each time index and each copy is multiplied by the amplitude of the impulse response. The final result is achieved by summing up all of the resulting copies.

Convolution is a linear, commutative, associative, and distributive operation. The commutative property states:

$$x[n] * v[n] = v[n] * x[n] \tag{2.5}$$

The associative property denotes:

$$x[n] * (v[n] * w[n]) = (x[n] * v[n]) * w[n] \tag{2.6}$$

Finally, the distributive property expresses the following:

$$x[n] * (v[n] + w[n]) = (x[n] * v[n]) + (x[n] * w[n]) \tag{2.7}$$

2.5 Frequency Domain

As depicted in Fig. 2.1, the representation for the speech recording "I do!" made it hard to analyze the recordings. Frequency domain representation delivers more meaningful information about the contents of these recordings. A frequency domain representation can depict the energy of the sound at different frequencies, and tell us about the vowels and consonants in a speech

recording as well as the frequency content about the background noise, harmonics, artifacts etc. Frequency domain representation can represents an audio recordings more uniquely and compactly. For example, a sinusoidal signal in the time domain is represented by many time domain samples, whereas in the frequency domain, a signal can be represented using only frequency, amplitude, and phase information.

Fourier and Z transforms can be used to convert the time domain signals, such as waveforms, into the frequency domain. Throughout this thesis, uppercase letters are used to denote frequency domain signals and lowercase letters to denote time domain signals. Signals with a [.] extension denote discrete time and those with a (.) denote continuous time signals.

2.6 Discrete Fourier Transform

In this thesis, we focus on the Discrete Fourier Transform (DFT) which is discrete in both time and frequency. Although using discrete time eliminates the complexity of integrals, it brings periodicity in the other domain which brings many potential artifacts. The DFT of a time domain signal and an inverse DFT of a frequency domain signal are shown in Eq. 2.8.

$$\begin{aligned} X[k] &= \sum_{n=0}^{N} x[n] e^{(\frac{-2\pi jkn}{N})} \quad , \; N-1 \leq k \leq 0 \\ x[n] &= \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{(\frac{j2\pi kn}{N})} \quad , \; N-1 \leq n \leq 0 \end{aligned} \tag{2.8}$$

where k, n, and N denote the frequency bin, the sample index, and the total number of time samples, respectively.

DFT is periodic due to the exponential term in Eq. 2.8. The periodicity of an exponential term is shown in Eq. 2.9.

$$exp(j2\pi) = exp(j2\pi k) \tag{2.9}$$

More intuitively, DFT reveal the frequency components of a signal by correlating a time-domain signal with real and complex sinusoidal signals of different frequencies and presents the corresponding correlation value at each

frequency. A good question is to ask where these sinusoidal signals are in Eq. 2.8. They are inside the exponential term. This exponential term can be expressed as a combination of sinusoidal signals using the Euler's formula as follows:

$$\exp(\frac{-2\pi jkn}{N}) = \cos(\frac{-2\pi jkn}{N}) + j\sin(\frac{-2\pi jkn}{N}) \tag{2.10}$$

Another way of representing DFT is by multiplying the DFT basis with the time domain signal. The DFT basis (both real and imaginary basis) for a 64-points DFT are shown in Fig. 2.2. Notice two things about these bases: their symmetry (conjugate symmetry for imaginary basis) and the oscillation between white and black colors that represents frequencies. The first row is the DC component; it is consistently white. Progressing down the rows, you see a more rapid pattern of whites and blacks (i.e. higher frequencies) until the middle row, which is the point of symmetry.

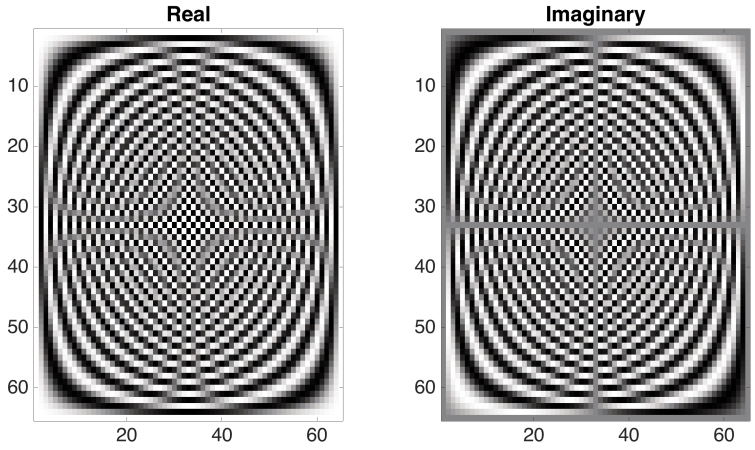

Figure 2.2: DFT basis - real and imaginary components.

The complexity of DFT is $O(N^2)$, the same as convolution. The DFT's symmetric property, however, can be used to speed up the Fourier transform. This faster DFT is called the Fast Fourier Transform, FFT. FFT does not approximate DFT; it is exactly DFT. The complexity of FFT turns out to be $O(Nlog(N))$. This reduction in the number of operations is very noticeable when dealing with a large number of samples. This is achieved by dividing the signal to half and taking the DFT of the smaller segments. Since the signals are halved, FFT works best with signals that are a power of two in

size. To take advantages of FFT efficiency, signals that are not a power of two in size are padded with zeros. For more information on FFT refer to Chapter 12 in [6].

2.7 DFT and Convolution

It can be shown that multiplication in the DFT domain is equivalent to circular convolution in the time domain (i.e. you need to use DTFT if you want to use linear convolution directly in the time domain) [7]. Circular convolution performs a periodic linear convolution, as shown in Eq. 2.11.

$$x[n] \otimes h[n] = \sum_{m=0}^{N-1} x[m]h[n-m]_N \Leftrightarrow X[k]H[k] \tag{2.11}$$

where $\otimes$ denotes circular convolution and $[.]_N$ denotes modulo integer N. DFT requires a circular convolution in the time domain. DFT is periodic, and that is due to the nature of sampling. Sampling in one domain causes periodicity in the other domain. Think of sampling a continuous signal as multiplying it by a set of pulse trains that are spaced by one period. This is equivalent to convolving a set of pulse trains (i.e. DFT of a pulse train is still a pulse train) that are spaced by one sampling frequency. The resulting signal contains the copies of the frequency domain replicas that are one sampling frequency apart. This spacing has to be big enough so that the periodic components do not overlap (e.g. avoiding aliasing). To avoid this problem, we use circular convolution. Circular convolution, however, can be made equivalent to the linear convolution if both input and impulse responses are zero padded to the linear convolution length before taking their DFTs.

2.8 Aliasing

Aliasing in audio describes an artifact that can take place in both the time and frequency domains. Aliasing permanently damages the information contained in the signal and cannot be recovered once it occurs. Again, the periodicity nature of DFT has to be taken into account, or there will be aliasing in both the time and frequency domains. That is, if the modified

time domain or frequency domain signal does not fit in the space provided, it will alias (i.e. replica in the other domain will overlap).

The most popular case of aliasing is breaking the Nyquist criterion. If the sampling rate is less than half the highest frequency (i.e. bandwidth) in a signal, then the low-frequency components leak into the higher frequencies.

$$\text{bandwidth} > \frac{sr}{2} \rightarrow \text{aliasing}$$

where sr denotes the sampling rate. As mentioned earlier, aliasing can also occur if the the input and impulse response are not zero padded before taking their DFT (i.e. converting circular convolution to linear convolution). For more information on aliasing refer to Chapters 3 and 11 in [6].

2.9 Windowing

Windowing was a great area of research in signal processing. Nowadays, a window that is already optimized for a particular application can be chosen off the shelf. Nonetheless, it is important to know a little bit about them.

2.9.1 Rectangular Window

Truncating a signal is the same as windowing it with a finite rectangular window. There are many variations of the rectangular windows. We discuss a zero-phase, center about zero, odd-length rectangular window shown in Eq. 2.12.

$$w[n] = \begin{cases} 1, & |n| < M/2 \\ 0, & else \end{cases} \tag{2.12}$$

A rectangular window has a very narrow main lobe (see Fig. 2.3) and a sharp transition band. Rectangular windows, however, have high side lobes right next to the main lobe, which causes spectral leakage, meaning that some information outside the pass-band leaks inside the main lobe.

The larger the size of the window, the more localized its spectral peaks are going to be (i.e. narrower main lobe). Side lobes do not change with the

size of the window and are more closely related to the transition band. For more information on rectangular window refer to "The Effect of Windowing on DFT" in [8] and Chapter 16 in [6].

2.9.2 Hanning Window

Hanning windows are commonly used in speech and audio processing. The Hanning window has a main lobe twice as wide as the one in the rectangular window. Hanning windows can be calculated using Eq. 2.13. Side lobes are about -32 dB and decay at 60 dB. The trade-off is between the width of the main lobe and the energy of the side lobes; the narrower the main lobe, the higher the energy of the side lobes.

$$w(n) = 0.5(1 - cos(\frac{2\pi n}{N-1})) \tag{2.13}$$

The square root of the Hanning window has a sharper main lobe at the cost of having slightly higher side lobes. The square root of the Hanning window is commonly used to re/construct the short-time Fourier transform of a recording.

2.9.3 Hamming Window

The Hamming window is similar to the Hanning window. The Hamming window's first side lobe is at -43 dB which is smaller than for the Hanning window, but it decays slower than Hanning window at 20 dB. You may not care about the average magnitude of the side lobes, but only the side lobes closest to the main lobe. Hamming windows can be designed using Eq. 2.14.

$$0.54 - 0.46\ cos(\frac{2\pi n}{N-1}) \tag{2.14}$$

Figure 2.3 compare rectangular, Hamming, and Hanning windows in the frequency domain.

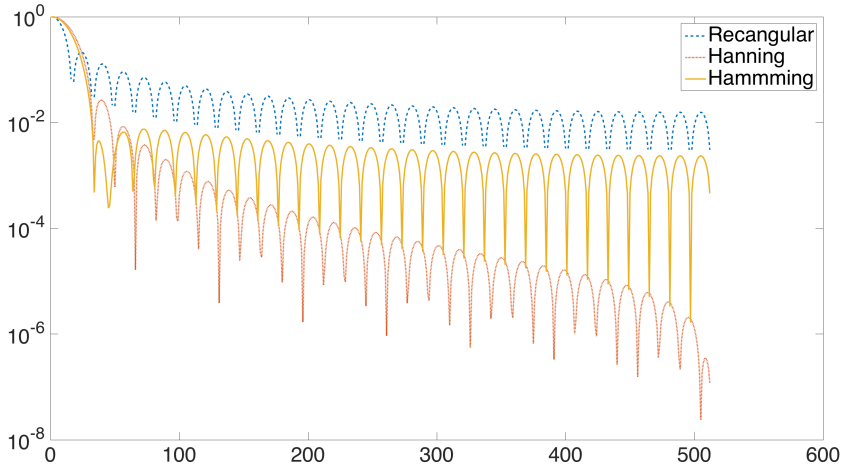

Figure 2.3: The rectangular window has the narrowest main lobe and the highest side lobe. The Hamming window has a narrower main lobe in the frequency domain due to the slower roll off in time domain. The Hamming window's first side lobe is smaller than that of the Hanning window.

2.10 Phase

The DFT of a signal is complex valued. There are a number of ways to denote the DFT of a signal, such as follows:

$$X[k] = |X[k]|\frac{X[k]}{|X[k]|} = |X[k]|e^{j\angle X[k]} = |X[k]|e^{j \cdot tan^{-1}\frac{imag(X[k])}{real(X[k])}} \tag{2.15}$$

where $\angle X[k]$, $imag$, and $real$ denote the phase, imaginary, and real part of the DFT of a signal. $tan^{-1}\frac{imag(X[k])}{real(X[k])}$ also denotes the phase of the signal.

Studying the phase of a signal can reveal how different frequencies are delayed relative to each other and how it affects an LTI system. It is well known that human hearing is not sensitive to the phase of a sound (at least in speech and audio enhancement applications), as discussed in Chapter 3 of [9]. Sign changes in the amplitude of a signal are captured by the phase as a discontinuity. Keep in mind that this type of discontinuity does not make the phase non-linear. Phase is 2π periodic. Phase is usually wrapped between $-\pi <$ARG$\{X[k]\} < \pi$ to make it easier to visualize. A wrapped phase is called the principle phase. The unwrapped phase is usually denoted as arg$\{X[k]\}$.

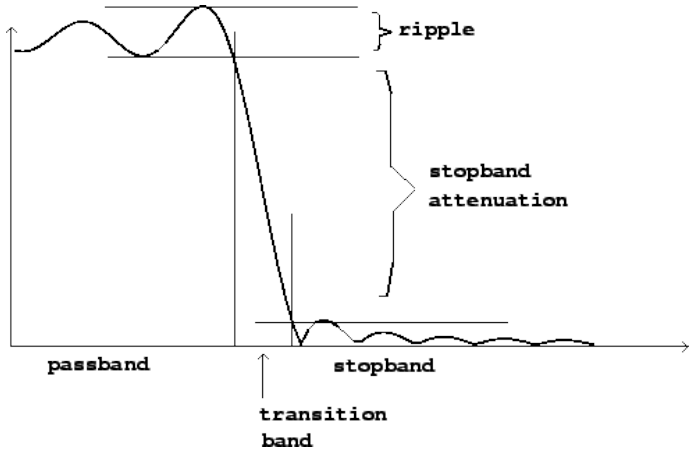

Figure 2.4: Tolerance diagram of a selective filter.

2.11 Selective Filters

Filters are used to select frequencies in a signal to operate on. Some of the common frequency selective filters are low-pass (i.e. pass low-frequencies), high-pass (i.e. pass high-frequencies), band-pass (i.e. pass mid frequencies), and band-stop (i.e. reject mid-frequencies). As shown in Fig. 2.4, practical filters have ripples in both the pass-band and stop-band regions, and a transition band. Figure 2.4 is called a tolerance diagram (i.e. denoting the magnitude response of a filter). The filter's phase, as discussed earlier, is not as important for speech and enhancement applications. We discuss zero-phase filtering in Section 2.11.1. Zero-phase filtering technique is used to removes any concerns regarding phase from non-real-time audio enhancement applications. In a nutshell, zero-phase filtering does not change the phase of the LTI system, but makes it non-causal.

There are two main categories for designing selective filters: FIR (finite impulse response) and IIR (infinite impulse response) filters. FIR filters are easier to implement. As a filter designer, you have control over the magnitude and the phase of FIR filters. FIR filters require a lot of coefficients, and they usually end up having wider main lobes with a lot of ripples. IIR filers are harder to implement, as they involve recursion (you can design FIR filters using recursion, too, but that is just making a simple design complicated). The phase response of an IIR filter cannot be specified, only the magnitude can be specified. IIR filters can have a sharp transition band using only a few coefficients.

Mathematically, FIR and IIR filters can be represented as expressed in Eq. 2.16.

$$y[n] = \sum_{l=0}^{L-1} x[n-l]b[l] - \sum_{m=1}^{M-1} y[n-m]a[m] \tag{2.16}$$

where b and a are the filter coefficients. For FIR filters $a = 0$ (i.e. FIR filters do not depend on the past output values). Table 2.1 compares the characteristics of FIR and IIR filters.

Table 2.1: Comparison of FIR and IIR filters.

Filter	**FIR**	**IIR**
Design	Optimization based	Analog transformation based
Magnitude	No constraint	Only frequency selective gain
Phase	Linear- full control	Non-linear - no control
Coefficients	Large number of coefficients	Few number of coefficients

2.11.1 Zero-Phase Filtering

Phase is an important characteristic of an impulse response that can be ignored in most enhancement applications, due to its complexity and the fact that change in phase is usually inaudible. There are, however, cases where the phase is important (e.g. dereverberation and virtual reality audio applications for 3D audio [10]). As mentioned earlier, linear phase filters are designed using an FIR filters, but consider what happens if we want to use an IIR filter. Another option to get a linear phase filter is to use a zero-phase filter. Zero-phase filters do not distort the phase or even introduce delays. Zero-phase filters can be obtained from any filters. In order to make an LTI system insensitive to the filter's phase, the following steps are performed:

1. $W(z) = X(z) \cdot H(z)$
2. $W^*(z) = H^*(z) \cdot X(z)^*$
3. $V(z) = H(z) \cdot H^*(z) \cdot X(z)^*$
4. $V^*(z) = X(z) \cdot H^2(z)$

$H_{\text{zero-phase}} = H^2(z)$

The zero-phase filter is squared in magnitude (i.e. no imaginary component, no phase), which also doubles the cut-off frequency and squares the magnitude of the ripples. Zero-phase filtering is a non-causal operation, because it requires time reversing the signal.

2.12 Resampling

EBAE requires two sounds, an input sound and an example sound. In order to be able to transfer the information from an example sound to the input sound, they need to be sampled at the same sampling rates. The same sampling rate can be achieved by resampling the signals using downsampling and upsampling techniques.

2.12.1 Downsampling

Downsampling, as the name suggests, decreases the sampling rate. Downsampling is achieved by simply removing some of the samples. Downsampling a signal of length L by a factor M means removing $\frac{L}{M}$ samples in between each sample, specifically those that are not indexed at multiples of ML. For example, downsampling a signal by 2 is equivalent to removing every other samples. This corresponds to expanding the frequency domain replicas by two. If the signal is not filtered properly before downsampling, the periodic component will overlap (i.e. aliasing). A low-pass filter with a cut-off at π/M can be used to cut the aliasing frequencies before the signal is downsampled.

The expansion in the frequency brings the ending frequencies closer to each other, as shown in Eq. 2.17 [11].

$$\omega = \Omega T \rightarrow \omega = \Omega MT \tag{2.17}$$

where Ω is the analog frequency. Downsampling stretches and squashes the signal in frequency domain.

2.12.2 Upsampling

Upsampling a signal of length M by a factor L adds $L-1$ zeros between each sample. Fortunately, there is no aliasing associated with upsampling. Upsampling compresses the frequency domain replicas (i.e. put more spacing between the replicas). After upsampling the signal, a low-pass filter can be used to smooth (i.e. interpolate) the signal in the time domain (i.e. pick one of the replicas in the frequency domain). Upsampling can be expressed as convolving a signal with delta functions located at multiples of L with zeros in between them as shown in Eq. 2.18 [11].

$$x[n] = \sum_{0}^{N} x[n]\delta[n - kL] \tag{2.18}$$

Taking the DFT of Eq. 2.18 gives:

$$X[k] = \sum_{0}^{K} x[n]e^{-j2\pi fnL/N} = X[kL] \tag{2.19}$$

2.13 Trimming

Most digital recorders tend to leave artifacts upon starting and finishing the recording (e.g. clicking noise). In EBAE, we automatically trim the sound to remove these artifacts. We applied a strong (i.e. big cut-off frequency), zero-phase, low-pass filters on the absolute value of the time-domain signal, as shown in Eq. 2.20.

$$y = |x| * h \tag{2.20}$$

where x is the input signal, h is a zero-phase low-pass filter, and y is the resulting signal. We then took the gradient of y. We then track two major peaks, where the voice activity begins and ends, and extract all the samples in between the indexes of the two peaks as the trimmed signal.

2.14 Scaling

Input and example sounds might have different volumes. The volume difference is going to be problematic if it influences the estimated filters which could result in clipping the matched sound (i.e. a signal is said to be clipped when the volume of the signal goes over the highest possible value defined in the system). We scaled both recording after they are trimmed to be in a range between -1 to 1 before processing them using Eq. 2.21.

$$\begin{aligned} t &= \frac{y}{std(y)} \\ z &= \frac{y}{max(|y|)} \end{aligned} \tag{2.21}$$

where y is the normalized recordings in the time domain and z is the normalized recording between -1 and 1.

CHAPTER 3
EQUALIZATION MATCHING

3.1 Introduction

In this chapter, we discuss graphical equalizers, power spectra, short-time Fourier transforms, and the proposed example-based equalizer matching to automate the equalization matching process.

3.2 Graphic Equalizer

Graphic equalizers, also known as a parametric equalizer, let the user adjust the gain on selected frequencies either through knobs on a hardware, digitally in a DAW, or a media player.

Graphic equalizers, interpolate the user-selected gains over all possible frequencies in the recording and then create a filter that adjusts the frequency gains of the recording. Alternatively, one can look at equalizers as a filter bank. Each selected frequency represents a frequency band, and changing the gain in each selected frequency denotes a change in the gain for all of the frequencies in that frequency band.

3.3 User Story

Adjusting the gain on a graphic equalizer by moving a bar up and down might seem easy, but getting the right gain at the right frequency is difficult. Some DAWs provide the user with a real-time listening tool so the user can hear the change in the equalizer while adjusting the gains. This procedure makes for an iterative process where a user employs listening skill to interpret how the frequency content in a desired recording is changing and uses these

cues to try to do better at the next iteration. Most DAWs provide the users with a spectrogram of the recording as well to help the more experienced users to detect the area of interest faster.

My imaginary friend, Jimmy, is going to help us motivate the propose of equalization matching further. Jimmy made a recording with his smart phone, but while recording, his phone was facing away from the sound source and the equalizer setting on the phone was also set to some unknown default. As a result, some frequencies are not getting the full gain they should be receiving. In signal processing terms, his recording has inappropriate gains at different frequencies. Depending on the situation, we might say the recording is, *muffled* or *squeaky*. Let us assume that Jimmy's recording sounds like it was passed through a band-pass filter. Jimmy detects an abnormality in the frequency content of his recording; however, detecting the exact frequencies that needs adjusting is difficult. This makes sense because detecting the exact frequencies and the amount of gains requires strong listening skills. Jimmy has poor listening skills, and frankly, does not have the time to use a graphical equalizer to iteratively solve this problem, as he has made this problem with all the recordings he has made over the past few months.

Jimmy and many other users do not have all the necessary knowledge or the patience to learn the ways of a DAW. Editing the equalization of a recording could be a very slow optimization problem, in a sense; the user needs to adjust the parameters on an equalizer until convergence of the true equalization setting is achieved. In the next section, we discuss a system that helps Jimmy fix the equalization on his recordings with a touch of a button.

3.4 Equalization Matching

Example-based equalizer matching is going to help relax some of these requirements, like strong listening skills, prior knowledge on using a DAW, and prior background in signal processing.

In order to understand how an example-based equalization matching works, consider Jimmy's case. Jimmy has another recording made using professional equipment and it has already been edited by a professional recording engineer. EBAE takes advantage of this recording to edit the equalization on Jimmy's previous recording automatically (i.e. undo the inappropriate

band-pass gains). In a nutshell, Jimmy provides his bad recording as an input sound and the better recording as an example sound, and then clicks on a button called "Match Equalization". EBAE automatically matches the equalization of the input sound to that in the example sound.

Before discussing the details on the proposed equalization matching, we must first have a good understanding of power spectra and how sound is represented in the time-frequency domain.

3.5 Power Spectral Density

Power spectral density or power spectra represents the power of a signal at each frequency bin. There are parametric and non-parametric ways of estimating the power spectra.

3.5.1 Parametric Spectrum Estimation

Parametric estimations are model based. They need data. Some parametric models that are used often with speech recordings are the moving average and the autoregressive models (i.e. all-pole model). The parameters for the autoregressive models are usually estimated based on some observed data. For more information, refer to "Parametric vs. Nonparametric Spectrum Estimation" in [8].

3.5.2 Non-Parametric Estimation

Periodogram is computed based on the magnitude squared of the DFT coefficients. It can also be calculated by taking the DFT of the autocorrelation vector in the time domain, where autocorrelation is defined in Eq. 3.1.

$$\begin{aligned} r_{xx}[k] &= \frac{1}{m}\sum_{n=0}^{L-1} x[n]x[n-m] \\ S[z] &= \mathrm{DFT}(r_{xx}) = \frac{1}{L}|X(z)|^2 \end{aligned} \tag{3.1}$$

where r_{xx} and $S[z]$ denote the autocorrelation vector and power spectra, respectively. A more familiar approach is to window the signal, take its

DFT, and then square it. It can be shown that the mean of a periodagram approaches the true value as the length of the data increases to infinity, but that also increases the variance of the estimation. For more information refer to the chapter "The Periodogram" in [8] and [12].

3.5.3 Average Periodogram

Welch's method can be used to calculate an average periodagram. Welch's method alleviates the poor variance of the periodagram dicussed earlier. In Welch's method, the signal is divided into overlapping segments. The periodagram of overlapping segments are then calculated and averaged. The more segments, the less is the variance. We used a similar approach to Welch's method when calculating the power spectrum of a sound using the Short-Time Fourier Transform (STFT).

3.6 Short-Time Fourier Transform

Speech signals are non-stationary. That means their first- and second-order statistics (e.g. mean and autocorrelation) change over time. A signal is said to be stationary if its frequency content does not change with time [13]. Taking a Fourier transform of a speech recording ignores its non-stationary nature and it is also inefficient. It is inefficient because if you have a lengthy signal, the FFT size is also large. STFT attacks both the non-stationary and inefficiency by dividing the audio recording into overlapping time frames and operating at one time-frame at a time. Audio signals are generally assumed to be quasi-periodic (i.e. almost stationary) in a 20-30 ms time frame [14]. The DFT of these time frames are then aligned next to each other. The magnitude DFT of these time frames can then be mapped to a colormap. This visualization is called the spectrogram of the sound. STFT is defined in Eq. 3.2.

$$X[n,k] = \sum_{m=0}^{M} x[m]w[n-m]e^{(\frac{-j2\pi kn}{N})} \tag{3.2}$$

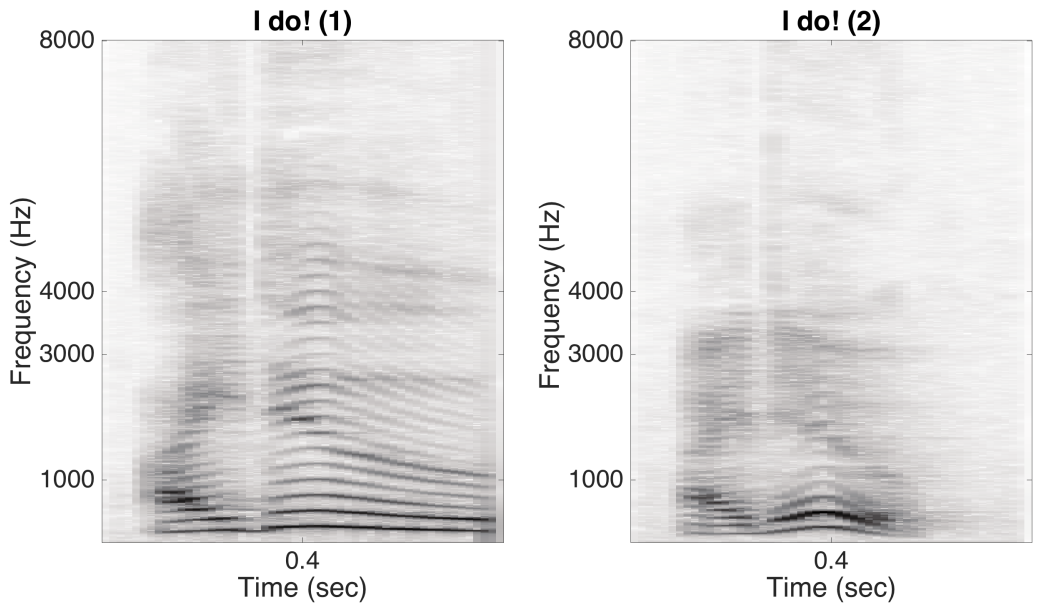

Figure 3.1: The spectrogram represents information about the vowels and consonants in a speech recording. The vowels in "I do!" are clearly represented here. We can also see that speech signals are more active in the lower frequencies.

where w is a sliding window of size M. Hamming and Hanning windows taper the signal at the edge of each time frame to make overlapping time frames. The tapering effect can be undone by dividing each time frame by that window function. That is, however, a computational hazard, as we are dividing the time frames by zero or smaller values. A more common way of recovering the time domain information is to use overlap-add and overlap-save methods.

In this thesis spectrograms are represented in a false colormap unless otherwise noted (i.e. black colors denotes higher energy). Figure 3.1 depicts a spectrogram for the waveforms in Fig. 2.1.

Overlap-Add

Overlap-add can be used to take the STFT and covert it back to time domain by using the following steps:

1. Divide the input signal in the time domain into blocks of size L tailed with M zeros, length of the impulse response, to avoid the need for aliasing later.

2. Apply a window to each segment before it is zero padded.

3. Take the DFT of each segment (and multiply them with the frequency domain of the window function).

4. Take the IDFT of the resulting segments.

5. Add all segments together. The zero tailed of the first segments is added to the first M samples of the second segment. If a *perfect reconstruction window* is used, then the tapering effect is neutralized.

The overlap-add method can also be mathematically defined as follows [15, 16]:

$$y(n) = \sum_{r=0}^{M} \frac{1}{N} \sum_{k=0}^{N-1} X(rR, k) e^{(\frac{j2\pi kn}{N})} \tag{3.3}$$

where R is the number of samples in each window and r is the frame index. x is the time domain signal and w is the window function [17].

Overlap-Save

Overlap-save is similar to overlap-add with one difference. The first segment is zero padded at the *beginning* of the recording by M zeros. The first M samples of all other segments contain the last M samples of the previous segment. The rest of the steps are similar to overlap-add until the reconstruction. Before summing all the segments the first M samples of the signal is discarded.

Perfect Reconstruction

In order to be able to perfectly reconstruct a time domain signal from its STFT, the overlapping windows must meet the *perfect reconstruction* criteria. Otherwise, the synthesized time domain signals suffer from amplitude modulation, which, if excessive, can turn into very audible artifacts. Let us now look at the overlapping windows in the time domain as follows:

$$y(n) = \sum_{r=0}^{M} y_r(n) = x(n) \sum_{r=0}^{M} w(rR - n) \tag{3.4}$$

where r denotes the frame number. In order for overlap-add and overlap-save to achieve perfect reconstruction, the overlap between time frames should meet the following constraint [18]:

$$\sum_{r=0}^{M} w(rR - n) = C \tag{3.5}$$

where C is a constant. Equation 2.32 is also known as the Constant Overlap-Add (COLA) condition [17]. In the context of speech enhancement, a 50% overlap between each time frame using Hamming windows makes for a perfect reconstruction [17]. Another case of perfect reconstruction of windows is that of Weighted Overlap-Add (WOLA) windows, where the synthesis and analysis windows are the same. WOLA windows are used commonly in audio compression applications [18]. The WOLA constraint is expressed in Eq. 3.6.

$$\sum_{r=0}^{M} w^2(rR - n) = C \tag{3.6}$$

One example of a WOLA window is a 75% overlapping square root Hanning window. In Matlab, the periodic option for Hanning window must be used to assure that the overlapping windows sum to a constant value [19]. One easy way to know if the STFT can perfectly reconstruct the time domain signal is to apply the overlapping window on a sequence of ones, take its STFT, and then its I-STFT. If a sequence of ones (or constant multiplies of it) is retrieved then the COLA condition is met. Figure 3.2 shows the COLA condition.

3.7 How to Match?

Our example-based equalization matching first extracts the power spectra from input and example sounds. EBAE then calculates an equalization filter using the estimated power spectra to manipulate the input sound frequency content such that its power spectrum matches the example sound power

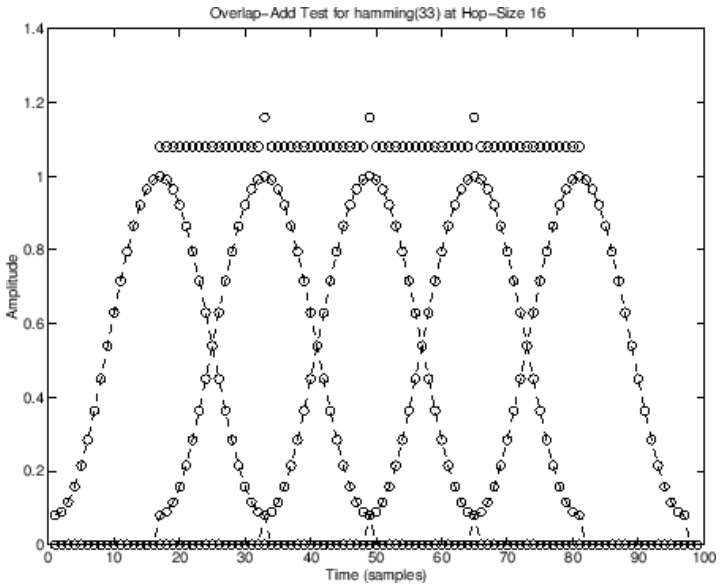

Figure 3.2: COLA - Hamming windows with 50% overlap [15].

spectrum. The equalization filter is expressed in Eq. 3.7.

$$E[k] = \frac{P_{ex}[k]}{P_{in}[k] + \beta[k]} \tag{3.7}$$

where in, ex, $P(.)$, k, and E denote input, example, power spectra, frequency index, and the time-invariant equalization filter, respectively. The $\beta[k]$ is a regularization parameter to avoid introducing ill-conditioned frequencies to the equalization filter (e.g. dividing by a small value). The equalization filter is then element-wise multiplied by every time frames in the input sound power spectrogram as follows:

$$Y_{mat}[t,k]^2 = E[k] \cdot X_{in}[t,k]^2 \tag{3.8}$$

where Y_{mat} and X_{in} denote the equalized matched sound and input sound spectrogram, respectively. t denotes the time frame index. The time domain signal is retrieved using the input sound phase through inverse STFT (I-STFT) using Eq. 3.9 and Eq. 3.10.

$$\phi_{Y\,mat}[t,k] = \frac{X_{in}[t,k]}{|X_{in}[t,k]|} \tag{3.9}$$

Time domain signal is shown in Eq. 3.10.

$$y_{mat} = \text{I-STFT}(|Y_{mat}[t,k]|\phi_{Y_{mat}}[t,k]) \tag{3.10}$$

Figure 3.3 depicts the block diagram for the proposed equalizer matching system.

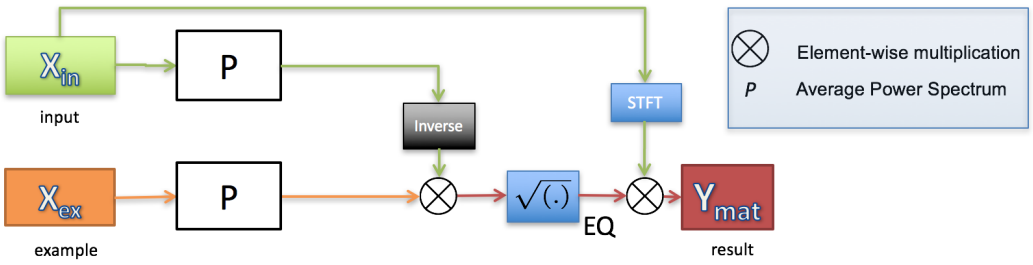

Figure 3.3: Equalization matching block diagram.

3.7.1 The Importance of the Example Sound

Is matching the power spectra going to fix the maladjusted frequencies in all scenarios? What if the user only wants to adjust the higher frequencies rather than adjusting the gains in all the frequency bins, as suggested in Eq. 3.7 and Eq. 3.8? These are reasonable critiques on equalization matching. We suggest two solutions for such issues.

1. Find an example recording that has frequencies similar to the input sound in the area we do not want to change, but the desired gain at frequencies we would like to change. This may seem like an unreasonable requirement, but for most scenarios an exact match is not necessary. For example, using a recording of a guitar as an example sound to adjust the low-frequencies of a speech recording does not work. Example sounds should be similar in their contents to the input sounds. If the input sound is a speech recording, then the example sound should be a speech recording.

2. A more advance solution on the back-end is to employ multiple equalizers for each frequency band and provide the user with multiple matched sounds (e.g. one with low-frequencies matched, one with mid-frequencies matched, etc.). We can still use Eq. 3.8 to apply the equalization filter. However, after finding the equalization filter using Eq. 3.6, all frequency bins

that are going to stay unchanged should be replaced with ones, as shown in Eq. 3.11.

$$\begin{aligned} E_{mod}[k] &= 1 \qquad b_i < k < c_i \\ E_{mod}[k] &= E[k] \quad d_i < k < e_i \end{aligned} \tag{3.11}$$

where E_{mod} is the modified equalization filter. The b_i and c_i are the beginning and ending frequency bins for those frequencies that are going to stay unchanged, and d_i and e_i are those that needs to be modified. It is then up to the user which recording sounds better. As discussed in Chapter 7, EBAE can be improved by learning from the user's common editing routines.

Another critique to equalization matching is the problem of sampling rates. What if the sampling rate of the example sound is lower than the input sound? Should we upsample the example sound or downsample the input sound? A similar solution is to, again, provide the users with multiple matched sounds that cover different scenarios and let the user select the desired recording.

3.8 Toy Example

A toy example (i.e. Jimmy's recordings discussed in Section 3.3) for equalization matching is shown Fig. 3.4. The input sound has noticeably higher gain, around $500 - 1500$ Hz. The example sound has a more consistent frequency content. As shown in Fig. 3.4, the matched sound was successful in undoing the maladjusted frequencies in the mid-frequency region and created a more consistent frequency content in the equalized matched sound.

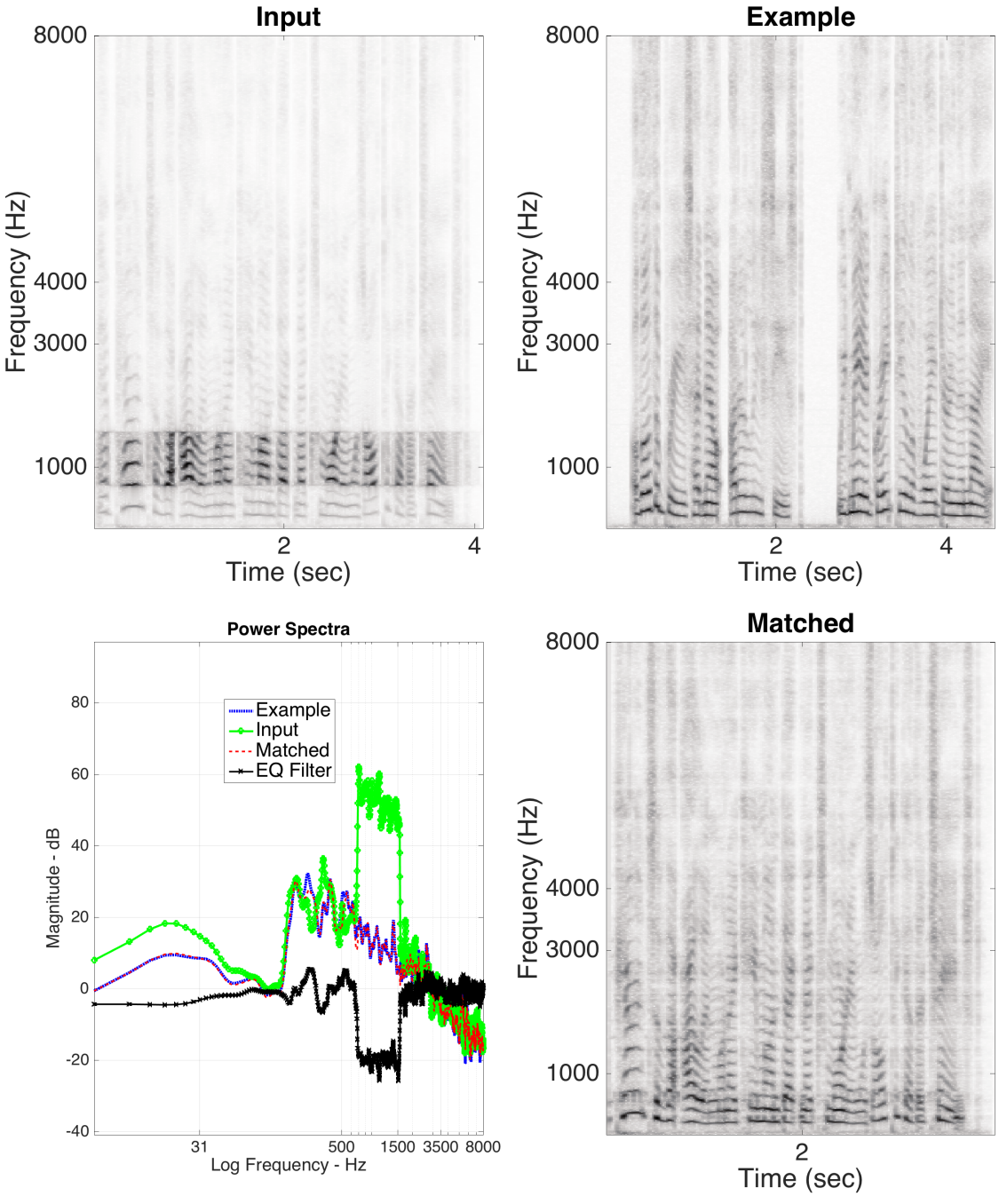

Figure 3.4: Toy example for the proposed equalization matching.

CHAPTER 4
NOISE MATCHING

4.1 Introduction

We live in a noisy world. Many efforts have gone toward denoising audio and speech recording using signal processing, and, more recently, using machine learning methods. In this chapter, we discuss the current noise reduction toolbox in DAWs, a high-level description of the EBAE noise matching system, and how current speech enhancement algorithms can be used in EBAE to estimate noise and clean sources in a noisy recording for the purpose of noise matching.

4.2 Modeling a Noisy Recording

A noisy signal can be represented as the summation of the clean and noise signals, as shown in Eq. 4.1.

$$\begin{aligned} y[n] &= x[n] + v[n] \\ Y[k,t] &= X[k,t] + V[k,t] \end{aligned} \tag{4.1}$$

where n, t, and k denote time sample, time frame, and frequency bin, respectively. The x, v, and y denote time domain clean, noise, and noisy signals, respectively. X, V, and Y denote the corresponding spectrograms. In this model, noise is assumed to be an additive white Gaussian noise (AWGN). Most noise reduction systems estimate the noise by making statistical assumptions about the nature of the noise and the clean signal (e.g. speech recordings have been studies for decards). Later in this chapter, we cover spectral subtraction [20] and Wiener filtering [21]. We also briefly discuss a non-ngative matrix factorization [22, 23] framework for removing non-

stationary noise signals.

4.3 Noise Reduction in Audition

Consider Adobe Audition's noise reduction toolbox. The first step in removing noise in Audition is to capture a *noise profile*. A noise profile summarizes the statistics of a background noise of a noisy recording. In order to capture the noise profile, the user is asked to highlight a few time frames, either in a time domain or a time-frequency domain, that represents the background noise. This could be time frames where there are no voice activities, such as the first few frames of the noisy recording. After selecting these time frames, the user is asked to adjust a number parameters, such as noise reduction gain, spectral decay, and smoothing, to adjust the parameters of the noise profile (i.e. how much of the noise is going to be subtracted from the noisy recording). The user can adjust these parameters while listening to the captured noise (or the estimated clean signal if desired) to optimize these parameters. Figure 4.1 depicts a snapshot of the Adobe Audition noise reduction toolbox [24].

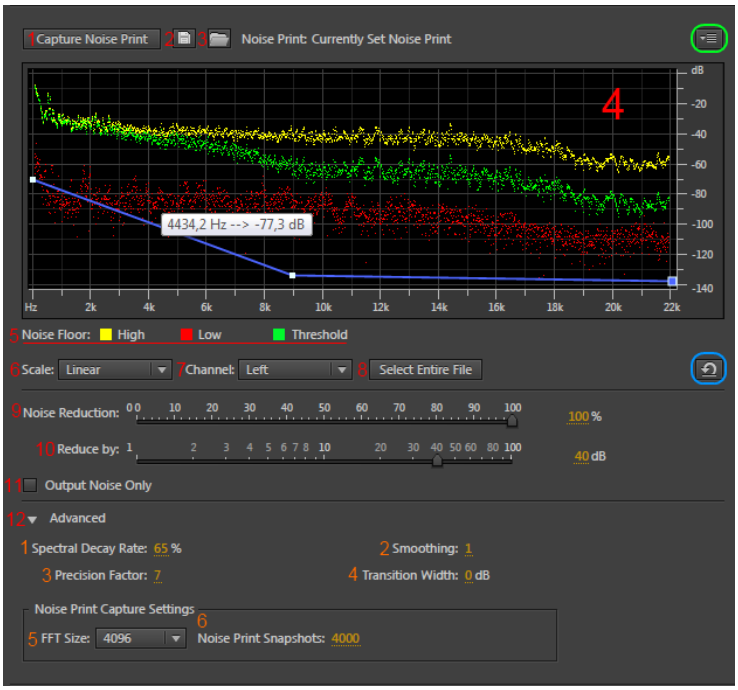

Figure 4.1: Adobe Audition noise reduction toolbox.

4.4 User Story

Most noise reduction algorithms leave behind some type of artifacts, and the final clean recording might not still sound natural. This is due to the fact that the algorithm is designed to remove all the noise. Therefore, the user is usually provided with numerous parameters to alleviate the artifacts by keeping some of the noise. The proposed noise matching system embraces noise, and as a result, the recording sounds more natural without the artifact.

Consider my imaginary friend Jimmy again. Jimmy is in charge of the sound production for an independent movie. In one of the scenes, the sound scene changes from a room with an air conditioner on (e.g. input sound with AWGN) to another room that has minor band-pass noise around 3-5 kHz (e.g. example sound) (see Fig. 4.3). Jimmy is asked to remove the air conditioner noise from the input sound and re-create the band-pass noise from the example recording to create consistency between the different scenes. The matched sound is created by adding the clean input sound with the example noise. Jimmy, however, is unable to find all the right parameters to estimate the clean and noise components and has many other scenes that require the same attention. Next, we discuss a number of denoising techniques for automatically extracting the clean and noise signals from a noisy recording.

4.5 Spectral Subtraction

Spectral subtraction subtracts an estimated noise profile from the noisy signal spectrum to estimate the clean sound spectra, as shown in Eq. 4.2 [25].

$$|\hat{X}[t,k]| = |Y[t,k]| - |\hat{V}[t,k]| \tag{4.2}$$

where $\hat{.}$ depicts an estimated quantity. Spectral subtraction estimates the *magnitude* spectrogram of the clean signal; as a result, it is important to avoid negative values when subtracting the noise spectrum from the noisy signal spectrum, as shown in Eq. 4.3.

$$|\hat{X}[k]| = \begin{cases} 0 & , |Y[k]| < |\hat{V}[k]| \\ |Y[k]| - |\hat{V}[k]| & , \text{else} \end{cases} \tag{4.3}$$

There are better ways for ensuring these non-negativity constraints as oppose to zeroing all the negative values. It turns out that Eq. 4.6 introduces a very annoying and audible artifact, called musical noise, at low SNR regions. We will discuss an algorithm for removing musical noise in the next section. It is easier to work with the power spectra as opposed to the magnitude spectra, so we rewrite Eq. 4.2 as the following:

$$|\hat{X}[k]|^2 = |Y[k]|^2 - \alpha\Delta|\hat{V}[k]|^2 \tag{4.4}$$

where α is an over-subtraction factor which is usually necessary to account for the underestimation in the noise profile. Δ is an additional over-subtraction factor which is usually provided to users for customization. Where are the cross-terms in Eq. 4.7? We assume that the clean and noise signals are uncorrelated (this assumption is not true for all SNRs). Equation 4.5 can also be written in the time domain as follows:

$$r_{xx} = r_{yy} - r_{vv} \tag{4.5}$$

where r is the autocorrelation vector of a signal. We do not like subtracting terms in signal processing. We like multiplication and convolutions, since DSP processors are optimized for these type of operations. Therefore, we express Eq. 4.5 as the following:

$$|\hat{X}[t,k]|^2 = H[t,k]|Y[t,k]|^2 \tag{4.6}$$

where Y and X are the STFT of the noisy and clean signals, respectively. H is the noise suppression transfer function shown in Eq. 4.7.

$$H[t,k] = \sqrt{1 - \frac{|\hat{V}[t,k]|^2}{|Y[t,k]|^2}} \tag{4.7}$$

The block diagram for spectral subtraction is shown in Fig. 4.2.

The time domain signal can then be reconstructed by taking the I-STFT of $X[t,k]$ using the noisy signal phase, ϕ_Y. Even though human hearing is not sensitive to the change in phase, using the noisy signal phase could introduce artifacts. The noisy phase is only accurate to use in the higher SNR regions (e.g. if SNR $>$ 10 dB regions, then the difference between noisy and clean

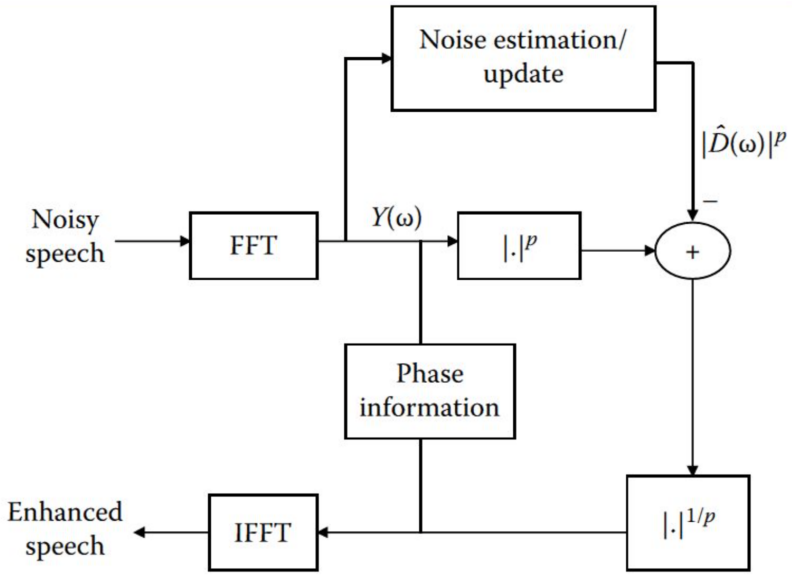

Figure 4.2: Spectral subtraction block diagram. Note that this diagram denotes the estimated noise as $\hat{D}$. p is set to 2 [26].

signal phase should not exceed $\frac{\pi}{10}$ or there will be artifacts) [27]. There are several ways for estimating the clean signal phase [27]; however, in this thesis we assume that the noisy recording phase is going to provide the estimated clean recording with an accurate phase information.

4.5.1 Musical Noise Suppression

As it turns out, the cross-term product is not zero [26], and that introduces musical noise to the enhanced recording. Geometric spectral subtraction is a variation of spectral subtraction that consider the cross-terms and operates on the complex values as oppose to the magnitude values. Since geometric spectral subtraction is a complex-valued algorithm, the phase of the estimated clean signal is also embodied in the estimation. The details of geometric spectral subtraction can be found in [20]. Subjective results do not, however, show any noticeable improvement using geometrical spectral subtraction in suppressing musical noise.

As shown in Fig. 4.3, musical noise looks like random frequencies scattered in time, especially in the low SNR regions. Musical noise is a very audible artifact, and most listeners prefer the noisy signal over the enhanced recording. Luckily, there are post-processing methods for removing the musical

noise [28, 29].

The musical noise reduction system used in EBAE is implemented based on the algorithm discussed in [28]. This algorithm starts by detecting the low SNR regions. The next step is to create a post-filter for smoothing the gain of the spectral subtraction algorithm result. In order to do that, two different SNRs are estimated: *a posteriori* SNR $\gamma(t,k)$ and *a priori* SNR $\zeta(t,k)$, as shown in Eq. 4.8.

$$\begin{aligned} \gamma[t,k] &= \frac{|Y[t,k]|^2}{|\hat{V}[t,k]|^2} \\ \zeta[t,k] &= \frac{|\hat{X}[t,k]|^2}{|\hat{V}[t,k]|^2} \end{aligned} \tag{4.8}$$

where $\hat{X}$ is the spectral subtraction result from Eq. 4.6. The next step is to detect the low SNR regions in the noisy recording. The following power ratio is used to detect these regions:

$$\eta(k) = \frac{\sum_{k=0}^{K-1} |\hat{X}[t,k]|^2}{\sum_{k=0}^{K-1} |Y[t,k]|^2} \tag{4.9}$$

The low SNR regions can then be found by setting a threshold on η_{thr}, as depicted in Eq. 4.10.

$$\eta_T[k] = \begin{cases} 1, & \text{if } \eta(k) \geq \eta_{thr} \\ \eta[k], & \text{else} \end{cases} \tag{4.10}$$

Now that we detected the low SNR regions, we can calculate a postfilter using Eq. 4.11.

$$G[t,k] = \begin{cases} \frac{1}{\text{round}[(1-\frac{\eta_T[k]}{\eta_{thr}}).\beta]}, & \text{if } \ |\hat{X}[t,k]|^2 \leq \eta_T(k) \\ 0, & \text{else} \end{cases} \tag{4.11}$$

where β is an extra gain smoothing factor that could be provided to the user. The final post-filtered signal is expressed in Eq. 4.12.

$$|\hat{S}[t,k]|^2 = |\hat{X}[t,k]|^2 G[t,k] \tag{4.12}$$

where $\hat{S}$ is the post-filtered enhanced recording. A block diagram for spectral

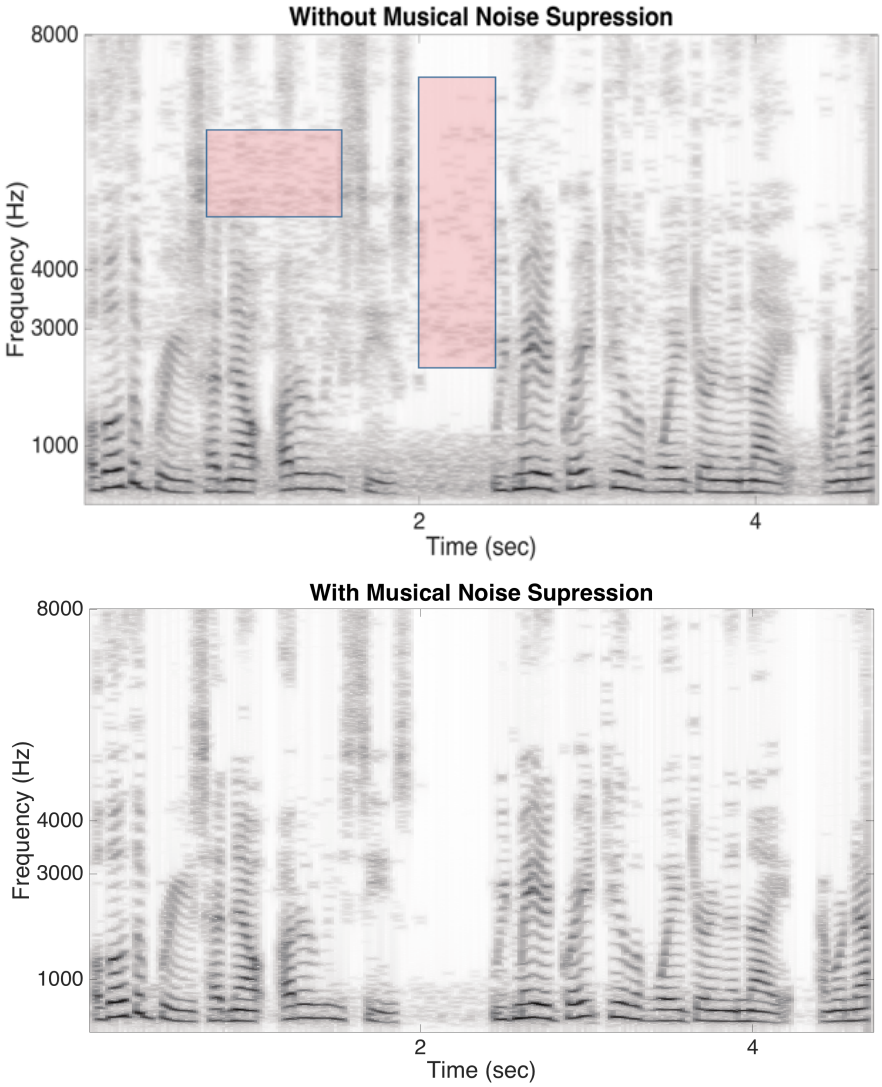

Figure 4.3: Typical enhanced speech recording using spectral subtraction. Musical noise is highlighted. Spectral subtraction with musical noise suppression post-filtering is shown in the bottom image.

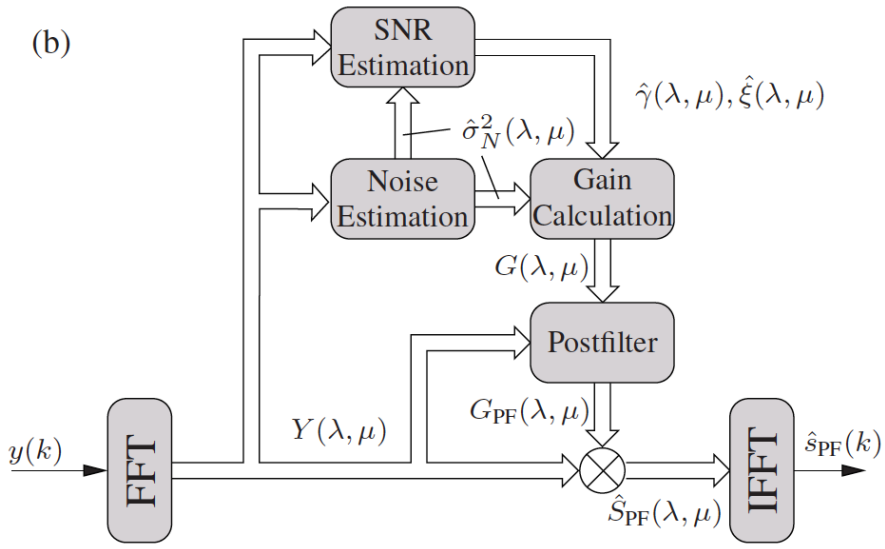

Figure 4.4: Spectral subtraction with musical noise suppression post-filtering block diagram. The $\hat{\xi}$ in the block diagram is denoted as ζ in the text. We dropped the $\hat{(.)}$ for simplicity [28].

subtraction with the musical noise post-processing filter is shown in Fig. 4.3.

Once the clean input sound and example noise are estimated Eq. 4.3 can be used to perform noise matching.

4.5.2 Multiband Spectral Subtraction

In order to better estimate the noise profile at each frequency band, we extended the spectral subtraction and the musical noise post-filter to multiple frequency bands. The mathematics for the multiband spectral subtraction is similar to the earlier method. The only difference is that quantities such as SNRs, noise profile, and enhance recordings are calculated individually for each frequency band. This technique is useful if the background noise is more stationary within each frequency band as opposed to a whole.

4.6 Wiener Filtering

Wiener filtering undoes the effect of an LTI system from an output signal in order to estimate the input signal by minimizing the error between the

estimated and the true input signals. A Wiener filter can be as simple as predicting the coefficients for an FIT filter, as shown in Eq. 4.13.

$$\hat{d}[n] = \sum_{k=0}^{M-1} h_k y[n-k] \tag{4.13}$$

where h_k represents the FIR filter coefficients (i.e. Wiener coefficients), M is the number of coefficients, y is the output signal, and $\hat{d}$ is the estimated input signal.

4.6.1 Wiener Filter in Time Domain

The difference between the true and the estimated input signal is calculated in Eq. 4.14.

$$e[n] = d[n] - \hat{d}[n] = d[n] - h^T y \tag{4.14}$$

where y contains the past M samples and h contains M FIR coefficients that we would like to find. The mean squared error (MSE) cost function is shown in Eq. 4.15.

$$J = E(e^2[n]) = E(d^2[n]) - 2h^T r_{yd} + h^T R_{yy} h \tag{4.15}$$

where $r_{yd} \in \mathbb{R}^{M\times 1}$ is the cross-correlation vector between y and the desired signal and $R_{yy} \in \mathbb{R}^{M\times M}$ is the autocorrelation matrix. $E(.)$ is the expectation operator. In order to find the optimal filter coefficient, we take the gradient of the cost function and set it equal to zero as follows:

$$\begin{aligned} \frac{dJ}{dh} &= -2r_{yd} + 2h^T R_{yy} = 0 \\ \rightarrow h^* &= R_{yy}^{-1} r_{yd} \end{aligned} \tag{4.16}$$

where h^* is the optimal FIR coefficients also known as the *Wiener-Hopf* solution [21, 30]. Keep in mind that, in most applications, we do not have access to the true desired signal, and consequently, r_{yd}.

4.6.2 Wiener Filters in Frequency Domain

Wiener filter can also be expressed in frequency domain, as shown in Eq. 4.17.

$$\hat{d}[n] = h[n] * y[n] \leftrightarrow \hat{D}(z) = H(z)Y(z) \tag{4.17}$$

The estimated error in the frequency domain is shown in Eq. 4.18.

$$E(z) = D(z) - H(z)Y(z) \tag{4.18}$$

Taking the gradient of the MSE in frequency domain and setting it equal to zero, we have:

$$H(z) = \frac{P_{dy}(z)}{P_{yy}(z)} \tag{4.19}$$

where P_{dy} denotes cross-power spectrum between the noisy signal and true clean signal, which is generally complex. We generally do not have access to P_{yd}. For more information refer to Chapter 6 of [9].

4.6.3 Wiener Filters for Noise Reduction

Wiener filter can be modified to reduce noise in noisy recordings. Autocorrelation of the noisy recording is shown in Eq. 4.20.

$$R_{yy} = E(yy^T) = E((x+v)(x+v)^T) = R_{xx} + R_{nn} \tag{4.20}$$

We assume that the clean and noise signals are uncorrelated. The optimal filter can then be calculated as follows [31]:

$$h^* = (R_{xx} + R_{vv})^{-1} r_{xx} \tag{4.21}$$

The noise reduction Wiener filter in frequency domain is:

$$H(z) = \frac{P_{xx}(z)}{P_{xx}(z) + P_{nn}(z)} \tag{4.22}$$

We can rewrite Eq. 4.25 using *a priori* SNR as the following:

$$\begin{aligned} \zeta_z &= \frac{P_{xx}(z)}{P_{nn}(z)} \\ \rightarrow H(z) &= \frac{\zeta_z}{\zeta_z + 1} \end{aligned} \tag{4.23}$$

$H(z)$ is directly proportionate to the SNR (i.e. at low SNR, the Wiener filter is almost zero and at high SNR is almost 1). That is the "attenuation portion of the spectrum is where the SNR is low" [31]. The musical noise reduction algorithm discussed earlier can also be applied to the Wiener filtered enhanced sound as well. Figure 4.4 depicts a block diagram for creating a Wiener filter for noise reduction.

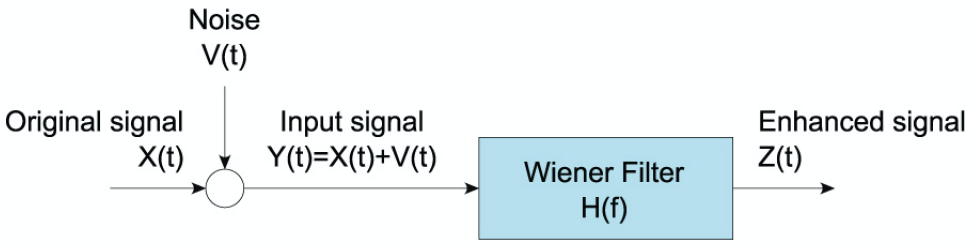

Figure 4.5: Wiener filter block diagram for noise reduction. The enhanced signal is denoted as $Z(t)$ in this diagram [32].

For more information about the performance of spectral subtraction and Wiener filtering, along with other classical speech enhancement algorithms, see [33].

4.6.4 Noise Estimation

There are several ways to estimate the noise profile, some which can be found in Chapter 3 of [9] and in [34]. One simple way is by averaging the power spectrum of the first few frames in a recording or more accurately detecting silence frames in a recording using voice activity detection techniques [35]. In this thesis, we calculated the 10-20th percentile of the power spectrogram and assign that as the noise profile. Next, we discuss a framework for denoising non-stationary noise from noisy recording. In order to understand this framework, we first need to have a good understanding of under- and

over-determined system among a few other background works discuss in the next few sections.

4.7 Non-Negative Matrix Factorization

There are number of techniques to discover latent structures in data, such as Principle Component Analysis (PCA), Independent Component Analysis (ICA), and Non-Negative Matrix Factorization (NMF). For example, ICA maximizes the statistical measure between different components [36] to reveal indpendant latent components. In order to approximate distinct sound sources in a recording, a number of constraints can be enforced on each hidden source for estimating sound sources more accurately [37, 23, 22]. The empirical results shows that NMF works better than ICA in source separation [38], and that is why we used it in this thesis to separate noise from a clean sound source.

NMF does not make statistical assumptions about latent sound sources; it only enforces non-negativity on the hidden components to achieve more meaningful results. Figure 4.5 depicts face decomposition using PCA and NMF [23]. As you can see, the eigen faces [39] achieved using PCA, all look like faces, some with negative values; however, NMF components depict actual face parts like nose and eyebrows due its non-negativity constraints. NMF has interesting properties that makes it applicable to audio signals. Since NMF expects positive values, we used the magnitude spectrogram of the recording as the input to NMF (i.e. ignoring the phase of the recording). NMF has been used in the audio field in many different applications, such as automatic music separation, denoising, and dereverberation [37, 40, 41, 42].

NMF approximates a non-negative matrix, $V \in \mathbb{R}^{M \times N}$, as a product of two other matrices, $W \in \mathbb{R}^{M \times K}$ which contains the basis for the data and $H \in \mathbb{R}^{K \times N}$, which contains the activation matrix for each basis [43].

$$V \approx W \cdot H \tag{4.24}$$

In order to approximate W and H, we need to minimize a cost function

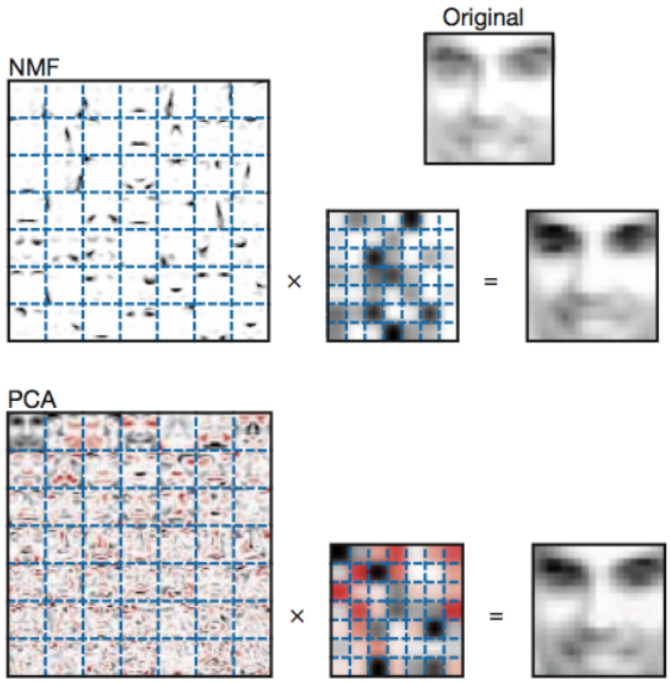

Figure 4.6: PCA vs. NMF in face decomposition [23].

between the true signal and its approximation, as shown in Eq. 4.25.

$$D(V||W,H) = ||W \odot log\frac{V}{W \cdot H} - V + W \cdot H|| \tag{4.25}$$

where D denotes the Kullback-Leibler divergence (KLD). Division is an element-wise operation here. In order to approximate W and H, a variation of gradient descent [43, 23] is used to update W and H individually. The multiplicative update rules for W and H are shown in Eq. 4.26. This update rule is convex in W and H individually [43].

$$\begin{aligned} H &= H \odot \frac{W^T \cdot \frac{V}{WH}}{W^T \cdot 1} \\ W &= W \odot \frac{\frac{V}{WH} \cdot H^T}{1 \cdot H^T} \end{aligned} \tag{4.26}$$

where $1 \in \mathbb{R}^{M \times N}$ is a matrix with all its elements equal to one. W and H are usually initialized with positive matrices.

A great example for how NMF can separate sources in an audio recording is

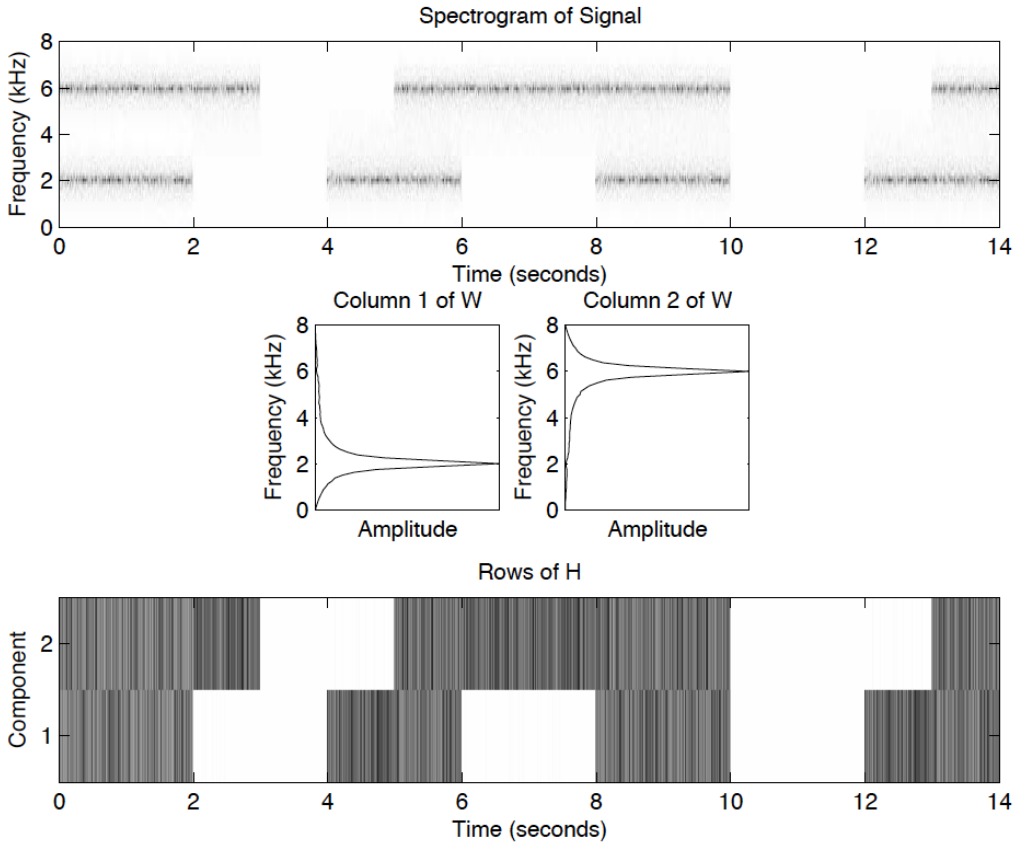

Figure 4.7: NMF decomposition for non-varying frequencies for the recording shown in the top image [43].

mentioned in [44], which we used here to motivate the application of NMF in separating audio sources in a recording. Consider having an audio recording with two bursts at 2 kHz and 6 kHz, as shown in Fig. 4.6. The goal here is to extract these two burst using NMF. If we set the number of components, $K = 2$, and perform the update rules in Eq. 4.26, we get the results shown in Fig. 4.6. The basis matrix has captured the frequency for each source and the activation matrix expresses when each frequency is activated throughout the recording. In this example, the number of sources are assumed to be known. There is, however, a way to estimate the number of components in an NMF framework when the number of sources is unknown to us [45].

Now, consider a more practical audio recording where the frequency sweeps over time, as shown in Fig. 4.7. Using the same parameters for the toy example mentioned in Fig. 4.6 gives the results shown in Fig. 4.7. The estimated W and H, however, do not capture the bursts. "NMF is not

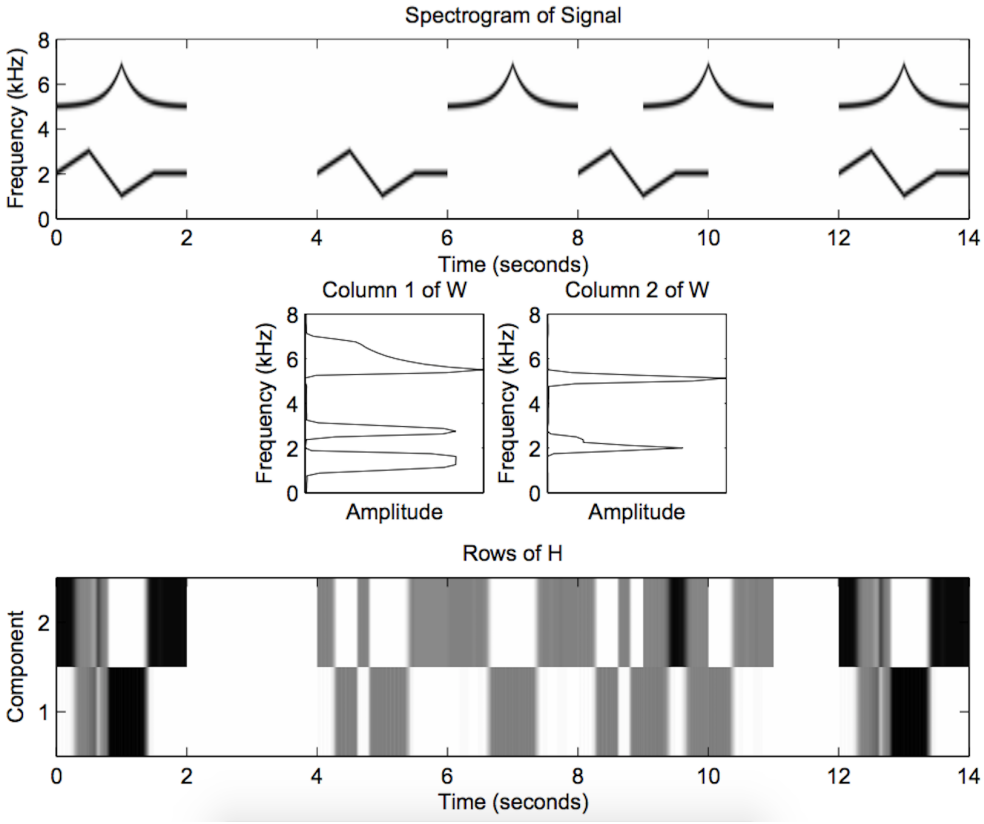

Figure 4.8: NMF decomposition for a recording with sweeping frequencies for the recording shown in the top image [43].

expressive enough to reveal this temporal structure" [43].

4.7.1 NMF for Speech Denoising

We discussed two algorithms for removing stationary noises. What if the noise is non-stationary (e.g. phone ringing in the middle of a concert recording). NMF can be used to attack non-stationary noises by separating the noise and clean sounds [40, 46, 47]. Speech enhancement algorithms in an NMF framework consist of two steps: training and denoising. Training is done on only clean speech and only noise recordings. This is done by minimizing the cost functions in Eq. 4.27.

$$\begin{aligned} &D(Y_{\text{speech}}||W_{\text{speech}}H_{\text{speech}}) \\ &D(Y_{\text{noise}}||W_{\text{noise}}H_{\text{noise}}) \end{aligned} \tag{4.27}$$

where D is KLD between the true magnitude spectrogram and its NMF reconstruction. $Y_{\text{speech}} \in \mathbb{R}^{M \times S}$ is the clean speech magnitude spectrogram, $W_{\text{speech}} \in \mathbb{R}^{M \times S_b}$ is the clean speech basis, and $H_{\text{speech}} \in \mathbb{R}^{S_b \times S}$ is the corresponding clean activation matrix. S_b is the number of components. $Y_{\text{noise}} \in \mathbb{R}^{M \times N}$ is the only noise magnitude spectrogram, $W_{\text{noise}} \in \mathbb{R}^{M \times N_b}$ is the noise basis, $H_{\text{noise}} \in \mathbb{R}^{N_b \times S}$ is the activation matrix for noise signals, and N_b is the number of components. Once the training stage is completed, the denoising stage can begin. W_{speech} and W_{noise} are fixed and assumed to be good bases for factorizing the noisy signal. For simplicity, we represented this process as follows:

$$H_{\text{noisy}} = \text{NMF}(Y_{\text{noisy}}, [W_{\text{clean}}, W_{\text{noise}}]) \tag{4.28}$$

where the items on the right-hand side are known and fixed during the factorization and items on the left-hand side are estimated. The estimated clean sound is then reconstructed using Eq. 4.29.

$$\hat{Y}_{\text{clean}} = W_{\text{clean}} H_{\text{noisy},1:S_b} \tag{4.29}$$

4.8 How to Match?

After estimating the noise and clean components of a noisy signal from Eq. 4.1, we can match the background noise of two noisy recordings. In this work, we used spectral subtraction with musical noise suppression from Section 4.5 to match noisy recordings as it provided better quality results in comparison to other methods.

A simple way is to add the estimated example noise the estimated clean input sound as follows:

$$y_{mat} = x_{in} + v_{ex} \tag{4.30}$$

A good estimation of a clean input and example sound's noise are, therefore, required. Both clean and noise components might contain some residue from their counter components, which could introduce some artifacts to the final matched sound based on the severity. In order to avoid these artifact, we propose matching the equalization on the estimated input noise to that in the

example noise and use the equalized matched noise instead, as shown in Eq. 4.31. The reason we can employ the equalization matching here is because the noise is stationary. This procedure can also be extended to matching noisy recordings by matching the equalization on the clean recordings as well.

$$\begin{aligned} v_{mat} &= \mathrm{EQ}(v_{in}, v_{ex}) \\ y_{mat} &= \mathrm{SNR}_{in}\ x_{in} + v_{mat} \end{aligned} \tag{4.31}$$

where EQ is the equalization algorithm discussed in Chapter 3 (Eq. 3.1 and Eq. 3.2). SNR denotes the signal to noise ratio. We decided to use the input SNR, but based on the application, one can use the example SNR in the same fashion. SNR is calculated using Eq. 4.32.

$$\mathrm{SNR} = \frac{\sigma\{x\}}{\sigma\{v\}} \tag{4.32}$$

where $\sigma\{.\}$ denotes the standard deviation. Keep in mind that both x and v are estimated through speech enhancement algorithms that are discussed later in this chapter. A block diagram for the proposed matching system is shown in Fig. 4.8.

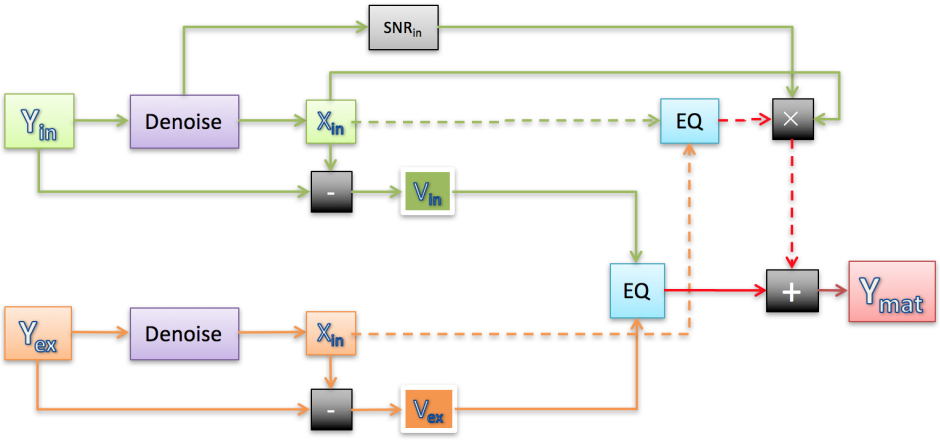

Figure 4.9: Noise matching block diagram.

The procedure for matching the noise in NMF framework is slightly different as we are attacking non-stationary noise signals. The matched noise

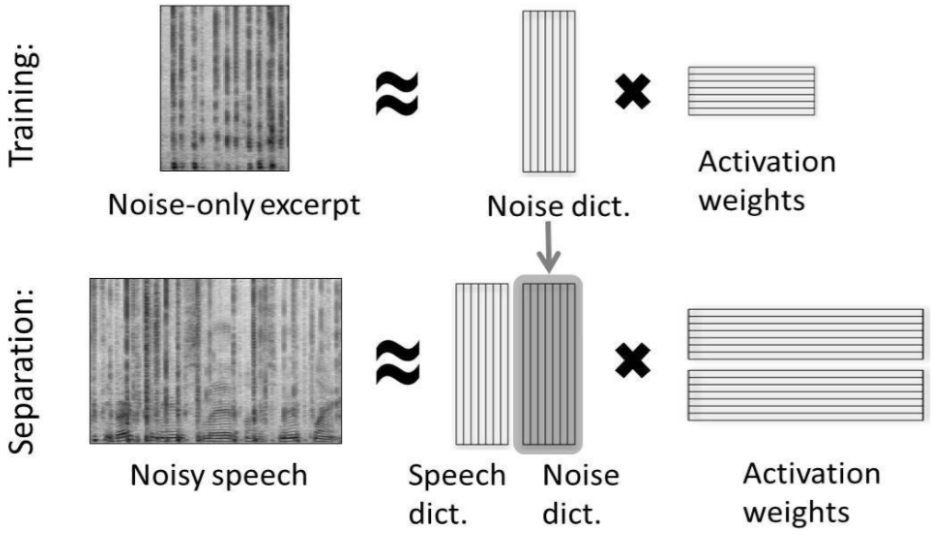

Figure 4.10: Speech enhancement with NMF. Clean speech training is not shown in this figure [47].

signal can then be estimated as follows:

$$\hat{Y}_{\text{mat}} = (W_{\text{clean,in}} + W_{\text{noise,ex}})H_{\text{noisy,in},1:S_b} \tag{4.33}$$

The phase of the matched recording for taking the signal back to time domain is shown in Eq. 4.34.

$$\phi_{\hat{Y}_{\text{clean}}} = \frac{Y_{\text{noisy,cpx}}}{\hat{Y}_{\text{noisy}}} \tag{4.34}$$

where *cpx* denotes a complex-valued signal and $\hat{Y}_{\text{noisy}}$ denotes the NMF reconstruction of the noisy signal. This trick is useful in compensating for any misestimation in the magnitude spectrogram that might have occurred when factorizing the signal. Figure 4.9 depicts how NMF is used to denoise a noisy speech.

4.9 Toy Example

Figure 4.10 depicts Jimmy's input and matched recordings concatenated with the example recording from Section 4.4. The proposed noise matching system

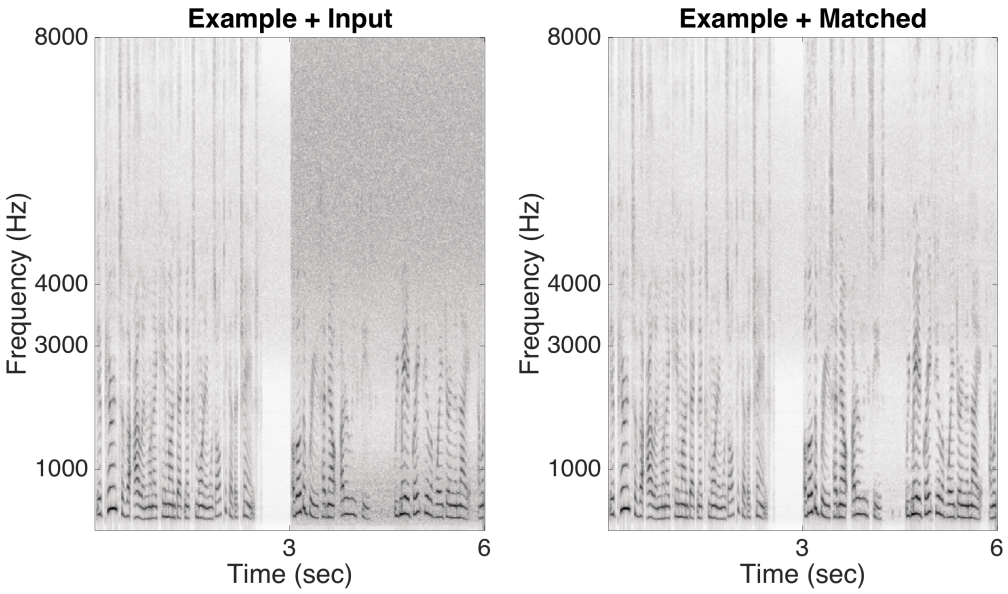

Figure 4.11: Noise matching toy example.

was successful in transforming the input noise to that in the example sound.

CHAPTER 5
REVERBERATION MATCHING

5.1 Introduction

The noisy world we live in is also surrounded by trees, walls, and objects. Sounds bounce of objects in a room and scattered through the air before reaching us. Researchers have spent decades working on algorithms for dereverberating speech recording and developing acoustic echo cancellation techniques to make recordings more intelligible. Signal processing community along with the musicians have spent decades finding ways to reverberate recordings synthetically for aesthetic reasons or blending musical notes. In this chapter, we will first discuss echo cancellation techniques, acoustical properties of reverberation (reverb), dereverberation techniques, and finally the proposed acoustic matching system for matching the reverb on multiple recordings.

5.2 Echo

Echo is the reflection of sounds from objects that arrive in smaller amplitude after the direct sound has arrived at the listener's ears, as shown in Fig. 5.1. In signal processing terms, echo is an impulse response with a set of delayed and decayed delta functions, as shown in Eq. 5.1.

$$y = (x * h) + v \tag{5.1}$$

where x, h, y, v and $*$ are the direct sound, echo response, wet sound (i.e. sound that contains the echo), noise, and the convolution operator, respectively. An arbitrary echo impulse response is depicted in Fig. 5.2. In this

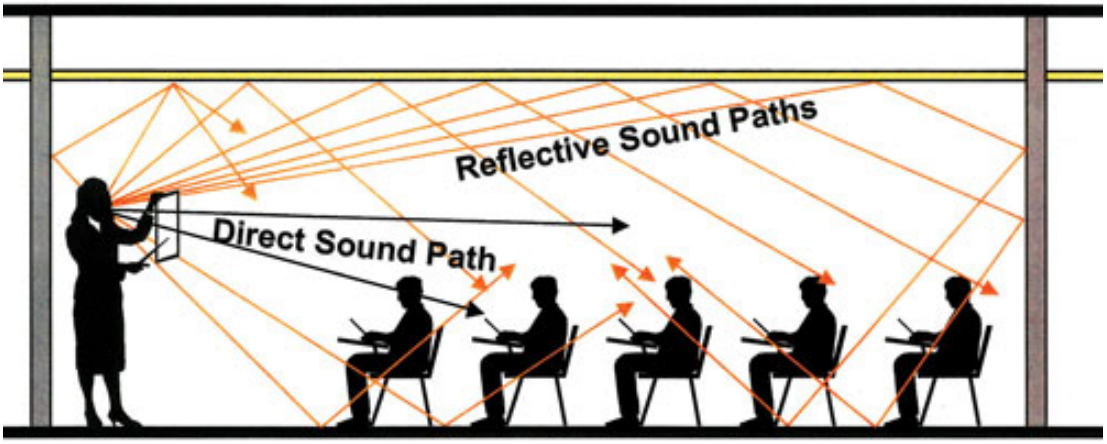

Figure 5.1: Demonstration of reflective and directive sounds in a classroom [48].

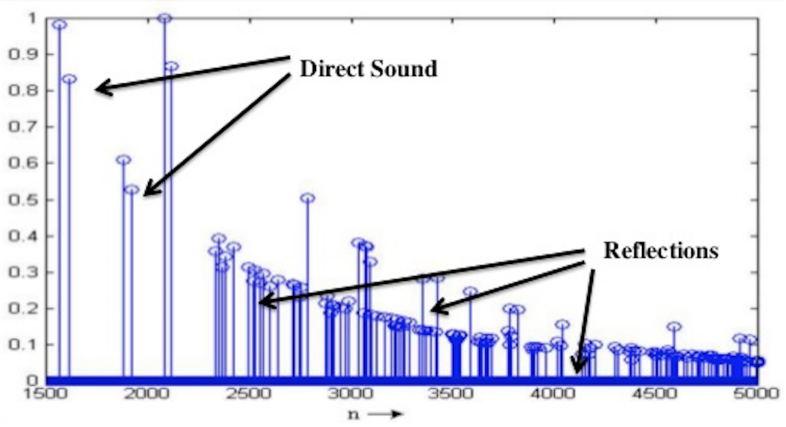

Figure 5.2: Echo impulse response [49].

chapter, we assume that there is no noise (i.e. $v = 0$).

Acoustic echo cancellation is a popular area in the speech recognition and telecommunication communities. Echo can distort speech recordings and degrade computer listening applications and video conferencing performance. The goal of echo cancellation is to estimate the direct sound, $\hat{x}$, by undoing the echo response from the echoed sound. Next, we discuss a few methods on echo cancellation and a scheme for matching the echos automatically.

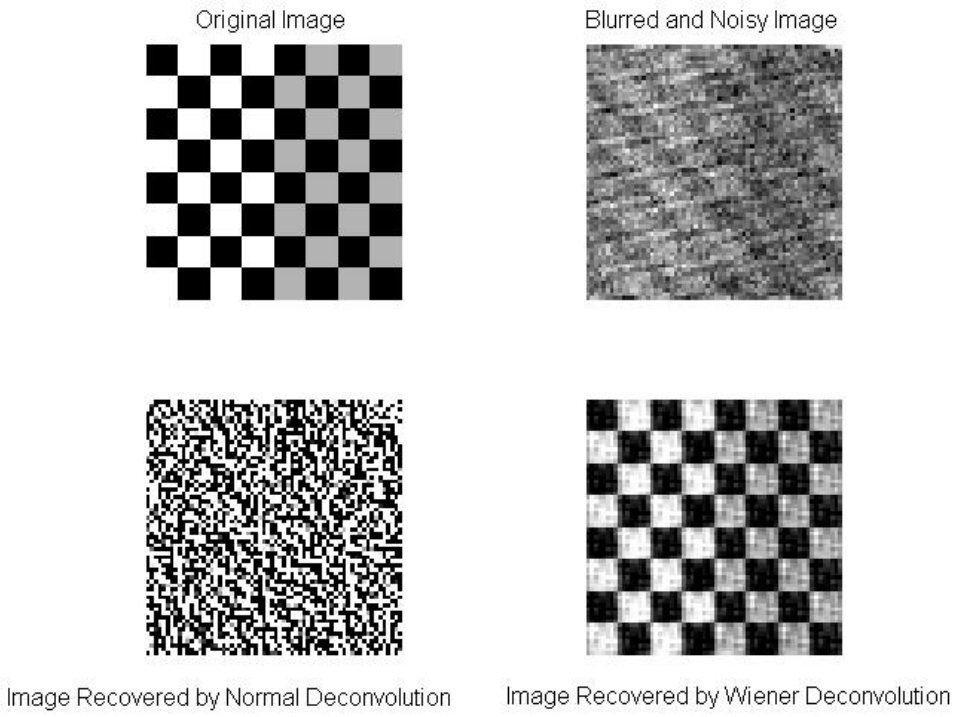

Figure 5.3: Wiener deconvolution vs. normal deconvolution [54].

5.3 Wiener Deconvolution

Wiener deconvolution is a technique for deconvolving an impulse response from a wet sound in frequency domain. Wiener deconvolution is commonly used for deblurring blurry images. Blurring an image is similar to adding reverb to a sound in a sense that they both involve convolution of a kernel with a signal [50, 51]. Figure 5.3 depicts the result of deblurring a blurry image using Wiener deconvolution and the regular division in frequency domain (i.e. impulse response deconvolution without regularization) [52, 53]. As shown in Fig. 5.3, the normal deconvolution is unable to restore the original image. This is due to the small noise components in the blurry image. These components turn into very large values when inverted (i.e. resulting in an unstable numerical situation).

Equation 5.2 expresses the Wiener deconvolution filter in frequency do-

main.

$$\begin{aligned} G(z) &= \frac{H^*(z)P_x(z)}{|H(z)|^2X(z) + P_v(z)} \\ G(z) &= \frac{1}{H(z)}[\frac{|H(z)|^2}{|H(z)|^2 + \frac{1}{SNR(z)}}] \end{aligned} \tag{5.2}$$

where P_x and P_v are the power spectral density for the dry and the noise signals and $\text{SNR}(z) = \frac{P_x(z)}{P_v(z)}$. The estimated dry sound is shown in Eq. 5.3.

$$\hat{X}(z) = G(z)Y(z) \tag{5.3}$$

The derivation of Wiener deconvolution is similar to the Wiener denoising derived in Chapter 4. The first step is to find the error between the true and the estimated signal as follows:

$$\begin{aligned} e(z) &= E|X(z) - \hat{X}(z)|^2 \\ &= E|X(z) - G(z)[H(z)X(z) + V(z)]|^2 \end{aligned} \tag{5.4}$$

where E denotes expectation. The error is then differentiated with respect to $G(z)$ and set equal to zero which gives us the result in Eq. 5.2.

5.4 Least Mean Squares

Least Mean Square (LMS) is a form of adaptive filtering that is commonly applied toward echo cancellation [55]. LMS is a gradient descent-based algorithm that converges to the optimal Wiener solution shown in the last section. The filter coefficients for LMS are updated according to Eq. 5.5.

$$w[n+1] = w[n] + 2\mu e[n]x[n], \quad n = 0, 1, ..., N \tag{5.5}$$

where μ is the step size, w is the adaptive FIR coefficient, and x is the input vector. The derivation for the LMS algorithm is similar to the gradient descent approach shown earlier [55]. Within each iteration of the LMS algorithm, three steps are performed:

1. Find the output of the echo cancellation system at each iteration.

$$y = w * x \tag{5.6}$$

2. Estimate the error between the desired and the estimated signals.

$$e[n] = d[n] - y[n] \tag{5.7}$$

3. Update the filter coefficient using Eq. 5.5 and repeat until convergence.

One important assumption that was made using both Wiener deconvolution and LMS is having access to the echo response or the desired signal. However, in most practical cases, we only have access to the wet sound.

5.5 Echo Matching

The purpose of echo matching is to extract the echo response from an example sound and apply it to the dry input sound. Since EBAE does not have access to the echo response and there are no reliable techniques to extract only the echo response from a recording in a blind fashion, we employed a user interface to provide a solution to echo matching. EBAE requires the user to mimic the echo using short sounds or words (e.g. saying "Hello.....Hello...Hello" where the first sound is used as a reference sound for normalization purposes and the second two sounds are the actual echos). In order to extract the echo response from the example sound, we use a simple peak tracking algorithm [56] on the log power spectrum of the recording. We determined the loudness and the time index of each echo normalized by the amplitude and the index of reference sound. We then synthesize an echo response using decayed and delayed delta functions.

5.6 Reverberation

Reverb is a similar concept to echo. Reverb is created when many reflections are build up in an environment, some may be absorbed by surface objects in the space. Echos can be heard distinctly (each echo separated by about

$50 - 100$ ms). Reverb, on the other hand, is usually perceived as a decaying white noise [57, 58]. Reverberation can be represented using Eq. 5.1 as well, where h is the reverberation kernel (also known as the room response) instead of the echo response.

5.6.1 Reverb Kernel

Reverb kernel describes how sounds are going delayed and decayed in a specific space. Room response consists of the direct sound impulse, early reflection impulses (those in charge of creating the echos), and a tail that represents a set of exponential decays commonly known as the reverb kernel. Figure 5.4 depicts a room response in the magnitude time domain. Early reflections and the tail of the reverb can be separated to individual impulses, as shown in Eq. 5.8. The phase response of the room is not studied in this thesis.

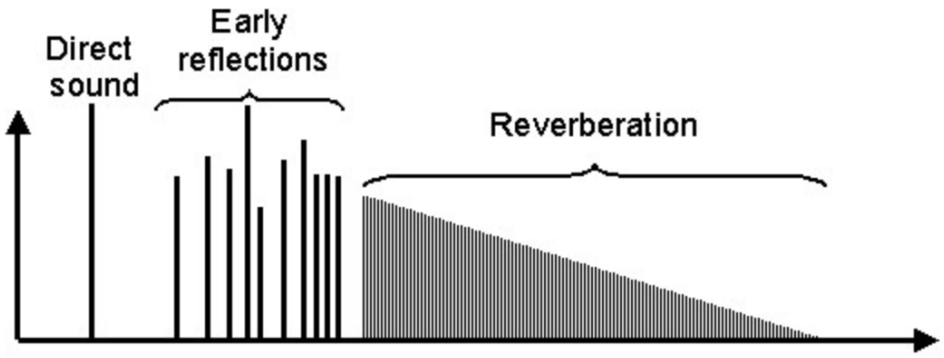

Figure 5.4: Room response consists of the direct sound, early reflections, and the reverb tail [59].

$$y = x * h_i + h_l * x[n - n_d] \tag{5.8}$$

where h_i and h_l are the early reflection and the reverb tail responses (i.e. late reflections), respectively. n_d denotes the time sample index for the tail. Reverb kernel (also goes as kernel) is visualized in time-frequency using STFT, as shown in Fig. 5.5.

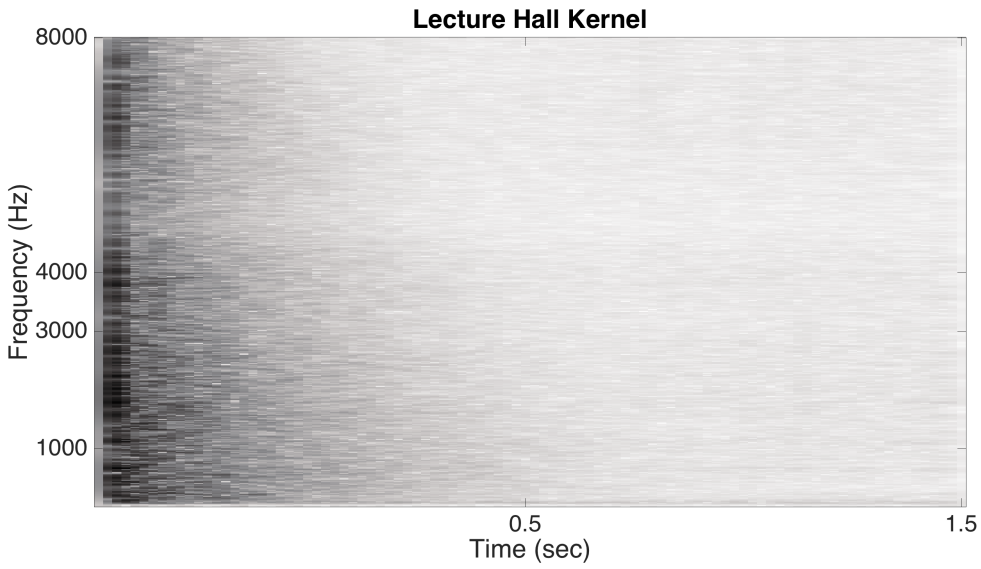

Figure 5.5: Reverb kernel spectrogram. Kernel decays faster in the higher frequencies than in the lower-frequency region.

Reverberation Time

Reverberation time, T_{60}, is traditionally known as the time it takes for a reverb sound to decay below 60 dB. More intuitively, T_{60} is the when the remained energy in the reverb sound is equal to the background noise. T_{60} is frequency-dependent (i.e. it takes longer for the sound to decay in lower frequencies than in higher frequencies). As shown in Eq. 5.9, T_{60} can be estimated using Sabine's reverb equation.

$$T_{60} = 0.116 \frac{V}{S \cdot a} \tag{5.9}$$

where V is the volume of the room in $meter^3$, S is the total surface area in $meter^2$, and a is the average absorption coefficient for all room surfaces. Keep in mind that Sabine's T_{60} does not consider room temperature, objects in the room, or the frequency-dependent nature of the room response.

A close concept to T_{60} is the critical distance. Critical distance is the distance between the source and the listener at which the air pressure level of the direct sound is equal to reverb sound. Critical distance can be calculated using Eq. 5.10. If the listener is farther than this distance, then all they hear

is reverb.

$$d_c = 0.057\sqrt{\frac{V}{T_{60}}} \tag{5.10}$$

Energy Decay Curve

A more practical way of estimating the T_{60} is by analyzing the room energy decay curve (EDC).

Schroeder introduced EDC as the tail integral of the room response as follows [60]:

$$\mathrm{EDC}(t) = \int_t^\infty h^2(\tau)d\tau \tag{5.11}$$

EDC represents the total amount of energy remaining in the room response. When EDC is represented in a time-frequency visualization such as spectorgram, the EDC is called the energy decay relief (EDR). EDR can be calculated using Eq. 5.12 and shown in Fig. 5.6.

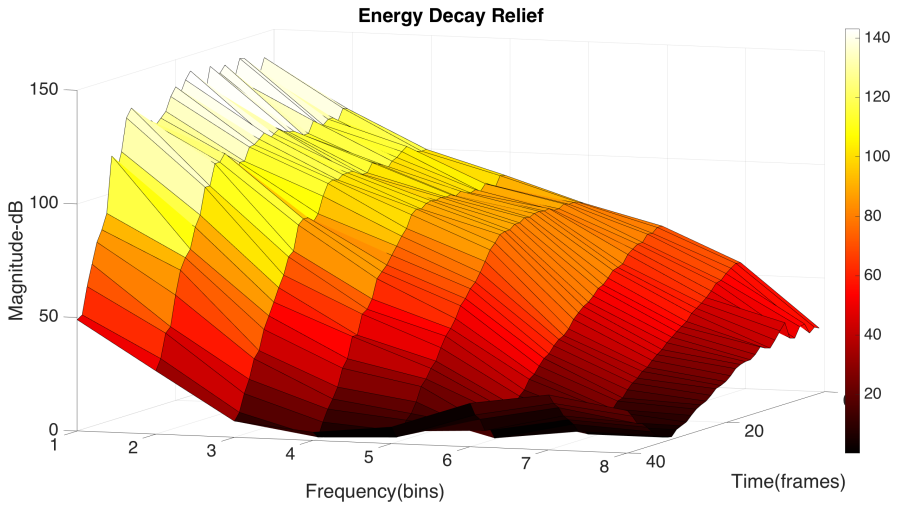

Figure 5.6: Energy decay relief of a dungeon.

$$EDR[t,k] = \sum_{m=n}^{M} |H[m,k]|^2 \tag{5.12}$$

EDR can be used to estimate T_{60} at each frequency bands by calculating its decaying slope over time.

There are also methods for blind estimation of T_{60} (i.e. without having access to the kernel). One particular method model the reverb tail as decaying Gaussian white noise processes [61]. This model estimates T_{60} using maximum-likelihood techniques by finding the most likely T_{60} that could construct the estimated tail in a desired reverb sound.

5.7 Measuring Room Responses

We discussed different aspects of a room response, but how can we measure a room response in an actual physical environment like a concert hall? We consider the room to be an LTI system (Eq. 5.1); therefore, if we use a delta function as an input to an LTI system (i.e. the room), then the output should be the LTI system impulse response (i.e. room response). Delta functions, however, are difficult to create in practice. We can approximate delta functions by popping a balloon and recording it. As shown in Fig. 5.7, one way to estimate the reverb kernel of a concert hall is to pop a balloon at different locations and average the extracted kernels. There are multiple problems with this approach. For example, the dynamic range and the frequency range of popping a balloon are not broad enough to represent the kernel with a good amplitude and frequency resolution. Experimental results also show that such methods are not robust and vary too much.

Another theoretical approach is to play the exponential function of different frequencies through a loudspeakers in a room and record them (i.e. an exponential input to an LTI system also outputs an exponential function [63]). Exponential functions, however, are not realizable in practice (i.e. exponential functions are unstable).

A practical approach to estimating the room response is to use chirp signals (e.g. a sweeping sound over all frequencies and over time), as shown in Fig. 5.8. A maximum length sequence (MLS), a pseudorandom binary sequence, can also be used. Chirp and MLS can cover a bigger range of frequencies and amplitudes and work well in practice. In order to extract the room response from MLS and chirp signals, the recorded signal is correlated with the original signal (i.e. before recording). An impulse response can then

Figure 5.7: Measuring the impulse response of the Moss Art Center by popping a balloon [62].

extracted from the correlated result at each frequency band [10].

One problem still remains: it involves the amplifier, ADC, loudspeakers, and the microphone frequency response included in measured response? Two solutions can be provided: (1) use flat frequency response devices and (2) undo the frequency response of each device using deconvolution methods. This is usually done by comparing the frequency response of the devices with a reference device that has a flat frequency response in the designated region.

There are a number of online room response databases that have considered many of these aspects when extracting the room response [65, 66].

5.8 Modeling Reverb Kernels

Reverb kernels can be modeled using a Gaussian noise and exponential decays, as shown in Eq. 5.13 [67].

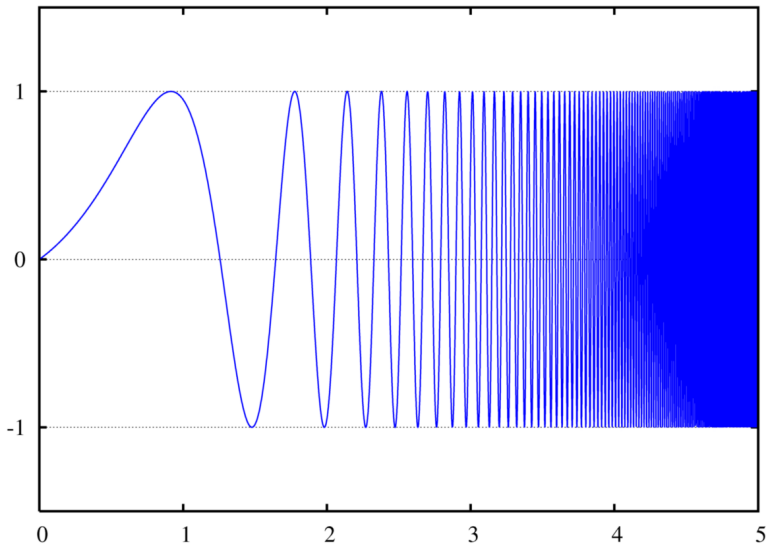

Figure 5.8: Chirp signal sweeps from lowest to highest frequencies in 5 seconds [64].

$$h[n] = b[n]e^{-\zeta n} \tag{5.13}$$

where b is coefficients for Gaussian noise, $\zeta = 3\frac{\ln(10)}{T_{60} \cdot f_s}$, and f_s is the sampling rate. The Polack model represents the kernel as exponentially decaying white noise, which is more accurate for modeling the room response tail. We used a similar model to match the reverberation kernels. A more accurate method for modeling reverberation is the image method [68].

5.9 Dereverberation

As mentioned earlier, when a sound propagates in a room, the room linearly distorts the sound. Too much reverb makes speech unintelligible. Specifically, people with hearing loss have a hard time understanding speech in reverberant environment. Regular hearing aids are unable to detect the direct sound from all the reflections. Beamforming techniques have successfully been employed in a number of hearing aids and headsets to attack this problem in real time [69]. Beamforming techniques use multiple microphones to form a beam toward the sound source; therefore, suppressing the reflections from

other directions [70, 71]. Reverb can also severely degrade speech recognition and other computer listening tasks.

Researchers have spent decades finding ways to dereverberate single channel audio using signal processing and machine learning techniques [72, 73, 74, 75, 76, 77, 78, 79, 41, 80, 81]. We first discuss a number of dereverberation techniques to dereverberate a wet sound, which is then escalated to methods that are capable of decomposing a wet sound to its dry components and kernel, therefore, capable of matching the reverberation between two wet recordings.

5.9.1 Room Response Inversion

Good questions to ask are: "If we have access to the room response, can we just invert it?" or "Can we use Wiener deconvolution to suppress reverberation?" The answer to each is no. Wiener deconvolution is only useful if the room response is, *minimum phase*.

A room response is characterized in the frequency domain as follows:

$$H(z) = |H(z)|e^{j\phi(z)} \tag{5.14}$$

An impulse response is said to be the minimum phase if all of its poles and zeros are in a unit circle (i.e. stable and causal). A typical room response, however, is not minimum phase. They can be factored into a minimum phase and an all-pass filter, however, as shown in Eq. 5.15.

$$H(z) = M(z)A(z) \tag{5.15}$$

where $M(z) = |H(Z)|e^{j\phi_m(z)}$. ϕ_m is the minimum phase component of the room response. $A(z) = exp^{j\phi_a(z)}$ is the all-pass component with its magnitude set to one for all frequencies; $\phi_a(z)$ is zero. Once the minimum phase component of the room response is extracted [82], it can be used to perform deconvolution using Eq. 5.2 and Eq. 5.3. This is all nice, but room responses are not accessible in most applications; as a result, blind dereverberation techniques are highly desired.

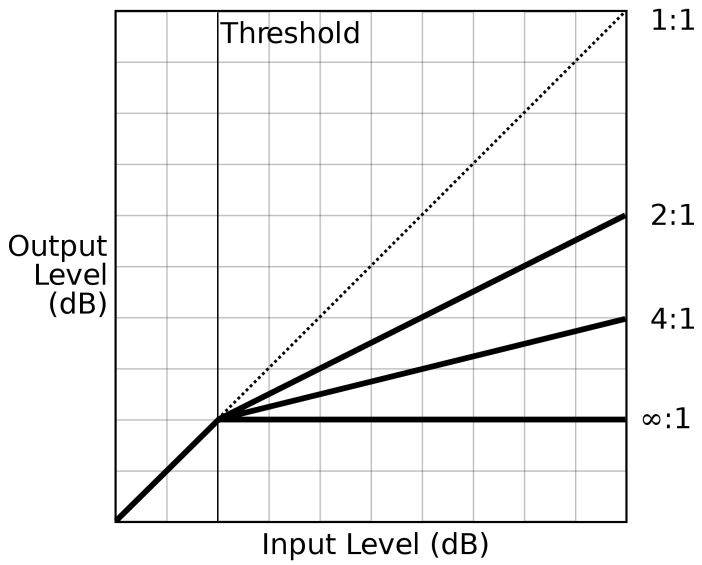

Figure 5.9: Noise gate compression.

5.9.2 Dynamic Multiband Noise Gate Compressor

Noise gates (also known dynamic range compressors) are traditionally used to control the volume of an audio recording during transients [83]. We modified noise gates to control the reverb in a recording. Figure 5.9 represents the relationship between the input and output of a noise gate system. When the amplitude of the recording crosses a threshold, the noise gates decrease the amplitude by a fixed ratio. Equation 5.16 depicts the hypothesis function for the noise gate used in Fig. 5.10. Figure 5.10 depicts the power spectrum of a drum recording before and after the noise gate is applied.

$$Z = \frac{X}{1 + aX^p} \tag{5.16}$$

where X is the power spectrum for the input reverb sound. If a is set to zero, input and output are equal to each other. p controls how much the input is going to be decayed. In order to count for the room response frequency-dependent decay rate, Eq. 5.16 can be customized and applied to individual frequency bands. Figure 5.11 shows the STFT of the drum sounds shown in Fig. 5.10. The dereverberated sound using the multiband noise gate algorithm in Fig. 5.11 is shown on the right-hand side.

The resulting sound has a smaller reverb tail. The noise gate fails to reduce

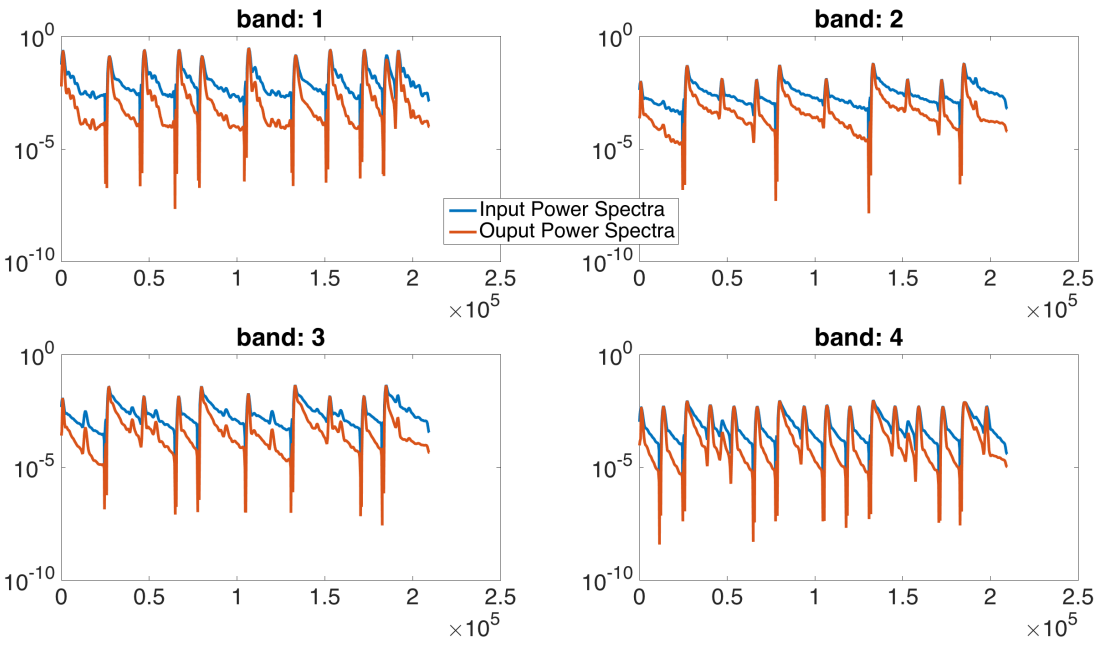

Figure 5.10: Input and output power spectra of drum sounds for the four-band noise gate shown in Eq. 5.16.

reverb when the exponential decay is not visible in the power spectra (e.g. speech recordings do not have the clear exponential decay of a drum sound). The abrupt change in the EDC can also introduce *pumping* and *phasiness* artifacts [84] to the reduced reverb recording.

5.9.3 Complex Cepstrum

Homomorphic Transform

As mentioned in Sections 2.3 and 2.4, the relationship between an input and output of an LTI system is represented as linear convolution in the time domain and as DTFT multiplication in the frequency domain. Homomorphic (homomorphic means same structure) systems are useful when an interfering signal is mixed with the source in a non-linear way. A non-linear system processed through a homomorphic process can be treated as a regular LTI system, as discussed in Chapter 34 of [6].

The non-linearity used in homomorphic systems is a complex logarithm. A complex logarithm operates on the absolute value of the signal and then

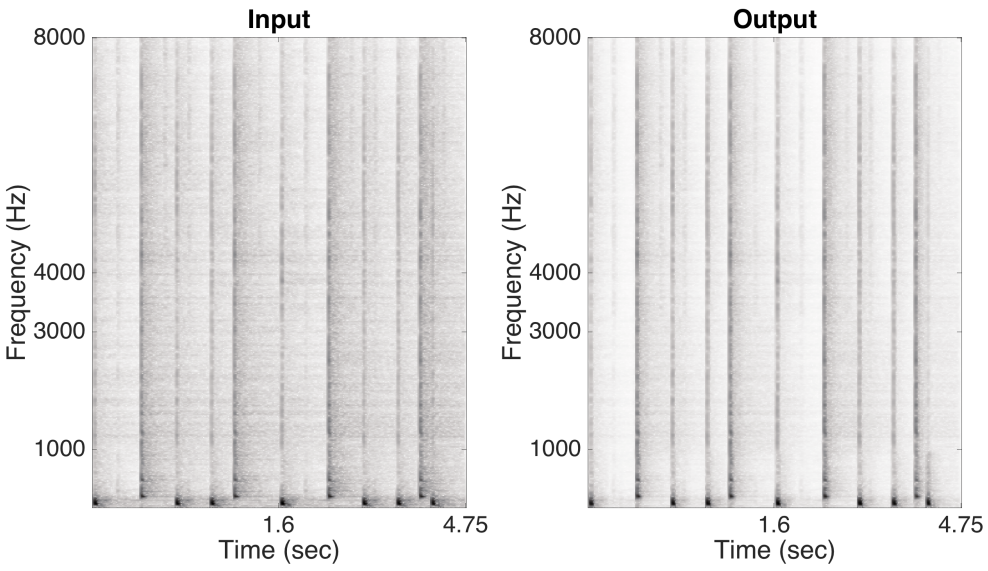

Figure 5.11: Drum sounds before and after the noise gate is applied.

the sign of the original signal is applied to the signal after filtering. Figure 5.12 depicts homomorphic transformation and its inverse.

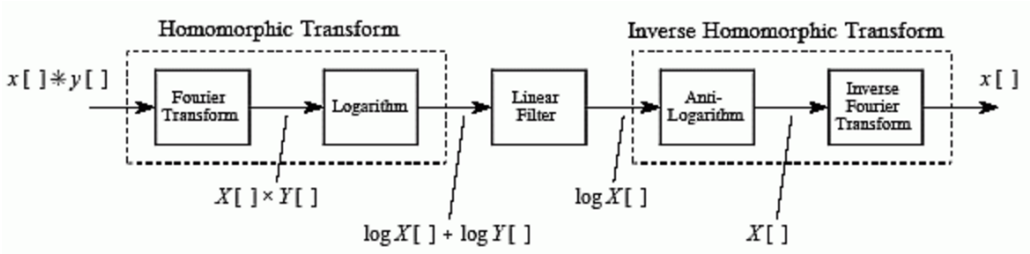

Figure 5.12: Homomorphic system [85].

Homomorphism has interesting vocabularies such as cepstrum and quefrency, which correspond to the spectra and frequency for an LTI system. Cepstrum represents the variation in the frequency. The lower quefrencies are associated with the lower variations, the envelop and the higher quefrencies are associated with the excitation signals. Cepstrums and their variations (e.g. Mel frequency cepstral coefficients known as MFCC) are used in modern speech recognizer systems as speech features.

Complex Cepstrum Dereverberation

Consider the following model for reverb:

$$Y(z) = X(z)H(Z) \tag{5.17}$$

where X, H, Y are the Z transform of the dry sound, the room response, and the reverb sound, respectively. $H(z)$ can formulated as follows:

$$H(z) = 1 + \alpha_i z^{-t_i}$$

where α_i and t_i control the gain and time delay of each decayed and delayed copy of the original sound. Taking the log of both sides of Eq. 5.17 gives:

$$\hat{Y}(z) = \hat{X}(z) + \hat{H}(z) \tag{5.18}$$

Taking the inverse Z transform of Eq. 5.18, we get:

$$\hat{y}(t) = \hat{x}(t) + \hat{h}(t) \tag{5.19}$$

where $\hat{y}(t)$, $\hat{x}(t)$, and $\hat{h}(t)$ are called the complex cepstra of $y(t)$, $x(t)$, and $h(t)$, respectively. It is believed that the low qufrencies correspond to the dry speech components, whereas the room responses are the ripples and peaks seen in the higher quefrencies. By low-time liftering (i.e log spectral filtering in the low quefrencies) the complex cepstra of dry speech recording can be estimated [86, 87, 88]). Deconvolution can then be done in the complex cepstra domain by assigning a deconvolution filter as $\hat{w}(t) = -\hat{h}(t)$ [88]. The time domain signal can then be synthesized by taking the Z transform, undoing the complex *log*, and then taking the inverse Z transform. This method, however, has little use in practical cases as it introduces too many artifacts and is unable to effectively suppresses the kernel.

5.9.4 Linear Predictive Coding

Linear predictive (LP) coding was initially developed for modeling speech [73, 74, 89]. It can be shown that there is a relationship between a reverb speech and its residual LP coefficients. By modifying the LP residuals one

can potentially dereverberate the recording [89]. The LP coefficients, b, of a reverb speech, y, are obtain using Eq. 5.20.

$$b = R_{yy}^{-1} r_{xx} \tag{5.20}$$

where R_{yy} is the autocorrelation matrix and r_{xx} is the autocorrelation vector. One method suggests computing the LP residuals of a complex cepstrum representation and clipping peaks in the higher quefrency regions. The LP residual-based dereverberation techniques suffer from the same problem as the complex cepstra dereverberation.

5.10 NMF for Dereverberation

As mentioned earlier, reverberation can be modeled as a linear combination of decayed and delayed copies of the dry sound using the convolution operator as follows:

$$\begin{aligned} y(n) &= \sum_{p=0}^{P-1} x(p) l(n-p) \\ y &= x * l \end{aligned} \tag{5.21}$$

The variables x, l, and y represent the dry recording, reverb kernel, and reverberated sound, respectively; p, n, and P are the time lag, time index, and the length of the kernel (in samples). The operator $*$ denotes time domain convolution.

Reverberation can also be approximated as the convolution between the input and kernel spectra [77, 75, 41, 90, 81] (known as the non-negative convolutive transfer function, N-CTF) using Eq. 5.22.

$$\begin{aligned} Y[k] &\approx \sum_{\tau=0}^{L_r-1} X[\tau, k] L[t-\tau, k] \\ Y &\approx X \star L \end{aligned} \tag{5.22}$$

The variables X, Y, and L represent the magnitude STFT of x, y, and l, respectively. The t, τ, and L_r are the time frame index, time frame lag, and the length of the kernel (in time frames). Operator $\star$ denotes convolution

between corresponding rows of X and L at each frequency bin for all time frames.

Kumar et al. [77] proposed a dereverberation technique for estimating the dry sound in an NMF framework by imposing sparsity on a Gammatone-like spectrogram [91] of the reverberant speech recording and minimizing the MSE between $G(Y)$ and $G(X) \star G(L)$ using Eq. 5.23.

$$\min(\sum_{i}(G(Y[t,k]) - \sum_{m}(G(X[m,k])G(L[t-m,k]))^2 + \lambda \sum_{t} G(X[t,k]) \tag{5.23}$$

where λ is an L_1-norm sparsity parameter enforced on the estimated dry speech recording. G represents the Gammatone transformation. Kumar et al. show empirically that using customized kernels for different frequency bands results in a better estimation of X [77]. The clean recording can be synthesized in time domain using inverse Gammatone transformation and the reverb sound phase.

More recent studies [75, 41] have extended Kumar's approach one step further by employing prior information on dry speech recordings (i.e. minimizing $D(Y||(W \cdot H) \star L)$ in the STFT domain). These studies rewrote Eq. 5.22 as the following:

$$Y \approx (W_d \cdot H_d) \star L = \hat{Y} \tag{5.24}$$

where $W_d \in \mathbb{R}^{\geq 0, M \times K}$ is a set of non-negative dry speech bases of K components, $H_d \in \mathbb{R}^{\geq 0, K \times N}$ is the corresponding non-negative activation matrix (referred to as encoding), and $L \in \mathbb{R}^{\geq 0, M \times L_r}$ is the reverb kernel.

Mohammadiha et al. derived update equations for these matrices in an NMF framework [41]. They proposed learning a dictionary of dry speech sounds offline by imposing the low-rank structure model of speech to better estimate the dry sound and the kernel.

Liang et al. employed a similar model with the addition of a stationary noise to Eq. 5.24 [75]. They proposed a probabilistic version of NMF by modeling W_d from dry speech recordings as exponential distributions. Liang then derived update rules for all hidden components: clean speech, kernel, and the noise parameters using a variational EM algorithm.

5.11 Reverberation Matching

We propose an algorithm that matches the reverberation of an input recording to that of a reference recording such that the matched recording sounds as if it is placed in the same room as the reference [92]. We propose a convolutive non-negative matrix factorization scheme that decomposes a reverberated sound into a dry sound and a reverberation kernel, along with a reconstruction scheme, for applying the example's reverberation characteristics onto the input recording.

5.11.1 Why and How to Match?

In reverb matching the goal is to change the room effect in a recording to resemble the effect of a different room (e.g. editing a sound recorded in a small room as if it was recorded in a concert hall). In a perfect world where the estimations are exact, reverb matching can be done by simply convolving a dereverberatd input sound with the example's kernel, as shown in Eq. 5.25. In most practical scenarios, however, we do not have access to the dry sound or the example's kernel, and estimating them reliably is hard.

$$y_{mat} = x_{in} * r_{ex} \tag{5.25}$$

5.11.2 Convolutive Non-Negative Matrix Factorization

In order to reveal the temporal structure in an audio recording (See Section 4.7) more accurately, NMF is extended to a convolutive model known as Concolutive Non-Negative Matrix Factorization (CNMF). CNMF can be seen as performing T number of NMFs, where T is the length of each spectrum of W [44]. We would like the relationship between W and H to be convolutive to capture the temporal relationship more expressively. If $T = 1$, CNMF reduces to NMF [44]. That is, each basis gets scaled and shifted by the activation pattern. CNMF can be expressed as follows:

$$V \approx \sum_{t=0}^{T-1} W_t \cdot H^{t\rightarrow} = \Lambda \tag{5.26}$$

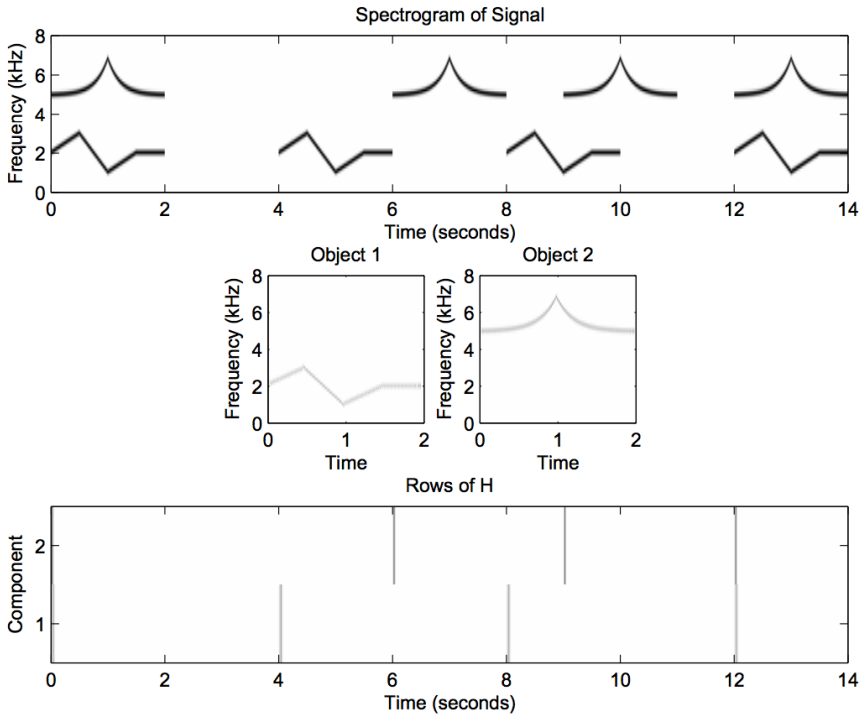

Figure 5.13: CNMF decomposition for the recording shown on the top image [43].

where $V \in \mathbb{R}^{M \times N}$, $W_t \in \mathbb{R}^{M \times K}$ and $H \in \mathbb{R}^{K \times N}$. W_t at the ith column denotes the spectrum at time t. The $(.)^{t\rightarrow}$ denotes a column shift operator that moves the columns i places to the right and fill the leftmost columns with zeros and $(.)^{\leftarrow t}$ does the same thing in the opposite direction. The resulting CNMF update rules are shown in Eq. 5.27.

$$\begin{aligned} H &= H \odot \frac{W_t^T \cdot \frac{V}{\Lambda}^{\leftarrow t}}{W_t^T \cdot 1} \\ W_t &= W_t \odot \frac{\frac{V}{\Lambda} \cdot (H^{t\rightarrow})^T}{1 \cdot (H^{t\rightarrow})^T} \end{aligned} \tag{5.27}$$

H is averaged to all of updates after updating all W_t's. Consider the previous example showed in Fig. 4.11 where NMF failed to factorize a sweeping frequency signals correctly. We set $R = 2$ components and $T = 2$ seconds. The CNMF results are shown in Fig. 5.13. W captures each time varying audio object correctly and H identifies the beginning time index for each basis.

There are many variations of NMF and CNMF with additional constraints, such as sparsity [93]. For example, one can enforce sparsity on an activation

matrix by modifying the cost function as follows:

$$G(V||\Lambda) = D(V||\Lambda) + \lambda \sum_{i,j} H_{i,j} \tag{5.28}$$

New update rules can then be found for this cost function using variants of gradient descent.

5.11.3 Proposed Decomposition Method

When a speech signal is reverberated, the reverberation process does not occur during the production of the speech signal (i.e. through the speech bases), but occurs when it resonates inside the recording space. We use this fact to propose a model that applies the reverberation kernel solely in the activation matrix using CNMF.

We can move the parentheses in Eq. 5.24, since convolution is an associative operator, as follows:

$$Y \approx W_d \cdot (H_d \star R) = W_d \cdot H_r \tag{5.29}$$

where $W_d \in \mathbb{R}^{\geq 0, M \times K}$ and $H_d \in \mathbb{R}^{\geq 0, K \times N}$ are the dry speech bases and dry activation matrix, respectively. $R \in \mathbb{R}^{\geq 0, K \times L_r}$ is the kernel and $H_r \in \mathbb{R}^{\geq 0, K \times L_{hr}}$ is the estimated reverberated signal's activation matrix (note the different dimension for L and R). $L_{hr} = N + L_r - 1$ is the convolution length. Equation 5.26 represents reverberation as the spectral convolution between convolutive speech bases with its corresponding reverb activation matrix, H_r. We propose the following steps for estimating the kernel and other hidden components:

1. A dictionary of speech bases can be learned offline from dry speech recordings. We can use NMF to represent these recordings as follows:

$$Y_d \approx W_d \cdot H_d = \hat{Y_d} \tag{5.30}$$

   where Y_d is the magnitude STFT of multiple concatenated dry speech recordings that we use as training examples. W_d and H_d are the dry speech bases and its corresponding activation matrix, respectively. These two matrices can be estimated iteratively using NMF update

rules [37, 43, 44].

We enforced a low-rank structure of the speech bases through exponentiating the activation matrix to $\alpha \in [0.9, 0.98]$ at each iteration (a process that sparsifies the activations, thus making the basis set more compact). For simplicity we will refer to this NMF update rule as follows:

$$(W_d, H_d) = \text{NMF}(Y_d, \alpha) \tag{5.31}$$

where the input Y_d, sparsity factor α, and other fixed items are shown on the right-hand side. The first item on the left-hand side corresponds to the result of the update equation for the bases (W_d) and the second item corresponds to the activation matrix (H_d).

2. Fix W_d from Eq. 5.31 and estimate the reverb encoding H_r for a desired reverb sound Y.

$$H_r = \text{CNMF}(Y, W_d, \alpha) \tag{5.32}$$

where CNMF denotes the update rules for convolutive matrix factorization [44] same format as Eq. 5.31.

3. Initialize R with monotonically decaying envelopes:

$$\begin{aligned} R(t, k_i) &= exp(\lambda_i \cdot t) \\ L_r \geq t \geq 1 \ , \ b_i \leq k_i \leq e_i \end{aligned} \tag{5.33}$$

where k denotes the frequency bins between the beginning (b_i) and ending frequency (e_i) bins in the ith frequency band. $\lambda_i \in [-0.9, -1.5]$ is an arbitrary decaying factor that can be adjusted to resemble how fast the reverb decays at different frequency bands.

4. Estimate R and the clean activation matrix by factorizing H_r as follows:

$$(H_d, R) = \text{CNMF}(H_r, \beta) \tag{5.34}$$

where $\beta \in [0.8, 0.9]$ is the sparsity parameter enforced on the estimated kernel iteratively. H_c and R are treated as W_d and H_d in the CNMF

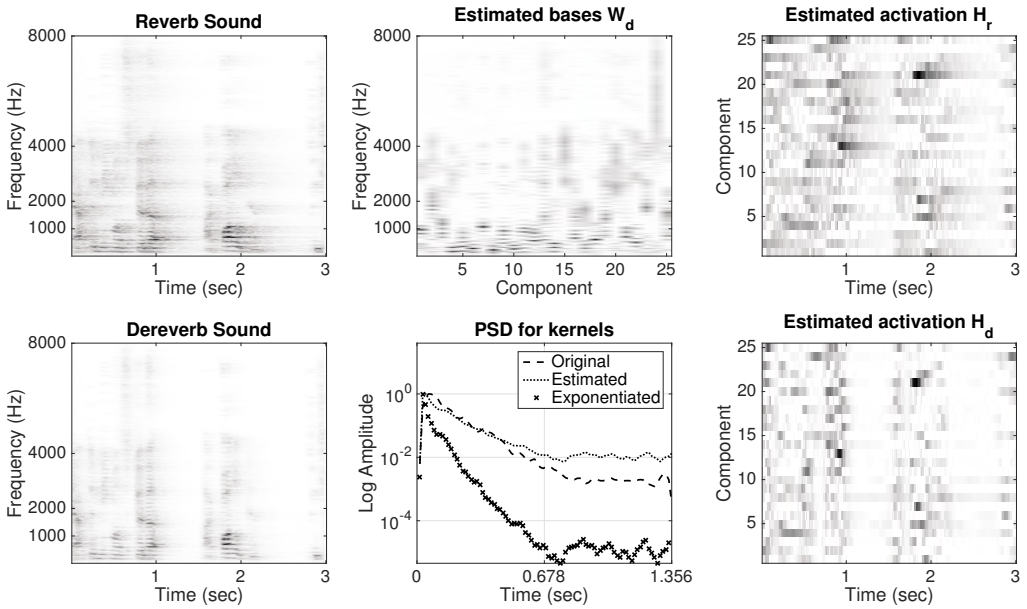

Figure 5.14: Example decomposition and reverb modification. The depicted reverb sound is a 10 second long recording in a big lecture hall. We zoomed in on the first 200 time frames for visual clarity. H_r was estimated by decomposing the reverberant speech sound using Eq. 5.32. CNMF is then applied on H_r to estimate H_d and R using Eq. 5.34. For illustrative purposes, R was averaged over all components. We reconstructed a less reverberated sound by setting $\beta = 2.7$ in Eq. 5.37. For reverb matching with a reference sound, we would estimate β from a another recording so that we can match the reverberation amounts.

update rules, respectively. The size of the kernel can also be approximated using blind T_{60} estimation techniques to get better prediction accuracy [61].

Figure 5.14 depicts an illustrative example in which a reverberant sound is decomposed to a reverb kernel and a dry sound, and is then modified to have a different amount of reverberation.

5.12 How to Match?

Once the input and example sounds are decomposed we can start the matching process. A naive way to match the reverb is to replace the input kernel

with the example kernel, as suggested in Eq. 5.35.

$$Y_{mat} \approx W_d \cdot (H_{d,in} \star R_{ex}) \tag{5.35}$$

where d, in denotes that H belongs to the dry estimate of the input sound. However, Eq. 5.36 results in considerable artifacts in the output. This is primarily an artifact of the decomposition process. The convolutions between each component activation and its corresponding element in R are not guaranteed to be synchronized. As is visible in the right plots in Fig. 5.16, the reverb kernels for each component are randomly staggered in time. This shift is compensated by a tie-shifting of the activations, but this shifting will not be the same for both decompositions making a reverb transfer infeasible. To bypass this issue, we will instead attempt to transform the existing reverb to the desired one.

This will be done by approximating the estimated reference kernel using exponentiation on the estimated input kernel. The resulting matched kernel is designed to have a similar decay rate as the reference kernel. Equation 5.36 depicts the cost function for finding the appropriate exponent for component j, β_j.

$$\text{argmin}_{\beta_j} D(R_{ex,j} || R_{in,j}^{\beta_j}) \tag{5.36}$$

where $\beta_j \in [0.5, 2]$. The ex, j subscript on R denotes the example kernel at component j, and D denotes the KL divergence distance. The matched kernel for each component can be calculated as follows:

$$R_{mat,j} = R_{in,j}^{\beta_{opt,j}} \tag{5.37}$$

where $\beta_{opt,j}$ is the optimal exponent for the kernel in the jth component, as determined by Eq. 5.37. The magnitude STFT of reverb matched sound can then be constructed as follows:

$$Y_{mat} \approx W_d \cdot (H_{d,in} \star R_{mat}) \tag{5.38}$$

The phase of the reverb matched sound is calculated in Eq. 5.39.

$$\Phi_{mat} = \frac{Y_{in,cpx}}{\hat{Y}_{in}} \tag{5.39}$$

where *cpx* denotes the complex STFT of the reverb sound. We normalized the STFT of the input reverb by its CNMF reconstruction to adjust the gains in magnitude STFT more properly. The retrieval of the time domain sound can be done using an inverse STFT. A general block diagram for matching reverb is shown in Fig. 5.15.

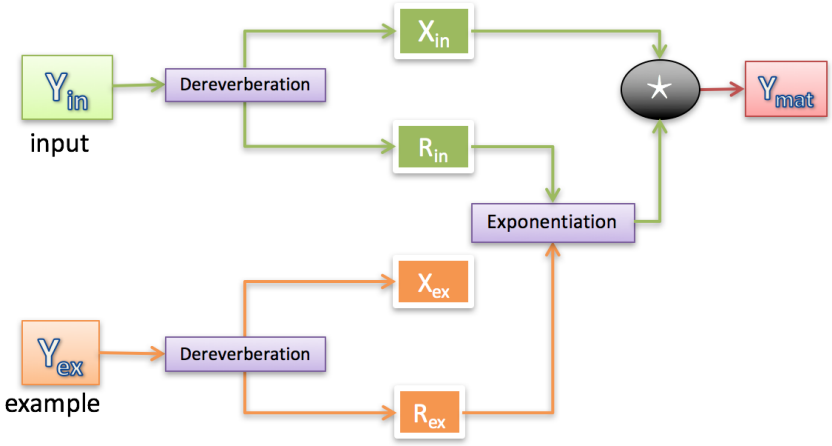

Figure 5.15: Reverb matching block diagram.

5.13 Reverb Matching Results

We evaluated the algorithm on both real and synthesized recordings. We collected 400 real recordings from the DAPS dataset [94] recorded in three different environments: living rooms, bedrooms, and conference rooms. Real recordings were contaminated with a stationary noise, so we used a variation of spectral subtraction [25, 26] (see Section 4.5) to denoise the recordings before decomposition. We synthesized 400 recordings using clean speech recordings from DAPS that are convolved with the following kernels from the AIR database [65, 66]: lecture halls, meeting rooms, and booths descending in their T_{60}, respectively. We then randomly assigned reverb sounds to input and example sounds and created 300 different matching cases for evaluating each real and synthesized set. Each recording is 10 seconds long. We resam-

pled the recordings to 16000 Hz sampling rate. The STFT transformation was done using a 513 samples square root Hanning window with 50% overlap. We trained a 25 element dictionary and set the number of iterations to 500. We used the true kernel size (L_r) for the synthetic dataset. For real recordings we set the bedroom, conference room and living room kernel sizes to 45, 65, and 85 frames, respectively. We generated ground truth recordings for the synthesized dataset and found the corresponding ground truth in DAPS for real recordings.

We evaluated the results using three metrics. We used Speech to Reverberation Modulation Energy Ratio (SRMR) [95] to measure the speech to reverb ratio on one recording. A higher SRMR denotes a dryer sound. We also employed the rating of Overall Quality (Covl) and Speech Distortion (Csig) which are composite metrics [96]. In Csig and Covl, we assigned the example sound as the clean recording and the input and matched sounds as enhanced recordings. Higher Covl and Csig denote a higher quality and a lower distortion in the enhanced file, respectively.

For a clearer representation we are depicting an *improvement score* for all these metrics. A high positive value denotes higher improvement after matching the kenrels. A negative value could denote that the algorithm has introduced artifacts to the matched sound which could be due to a poor estimate of the input or example kernel.

$$\begin{aligned} \text{score}_{\text{Covl}} &= \text{Covl}_{ex,mat} - \text{Covl}_{ex,in} \\ \text{score}_{\text{Csig}} &= \text{Csig}_{ex,mat} - \text{Csig}_{ex,in} \\ \text{score}_{\text{SRMR}} &= log\frac{|\text{SRMR}_{in} - \text{SRMR}_{ex}|}{|\text{SRMR}_{mat} - \text{SRMR}_{ex}|} \end{aligned} \tag{5.40}$$

The raw SRMR for each category is depicted in Fig. 5.17 as a reference. As shown in Fig. 5.16, the proposed algorithm has done a good job in matching the real and synthesized reverb when adding (e.g. BM, BL, and ML in the synthesized set) and removing reverb (e.g. BC and LC in the real set). SRMR score is improved over all synthesized recordings and most of real recordings except those transitioning to a bedroom. Overall quality and speech distortion metrics show improvements in most cases. Overall quality and speech distortion seem to be more closely correlated with each other in

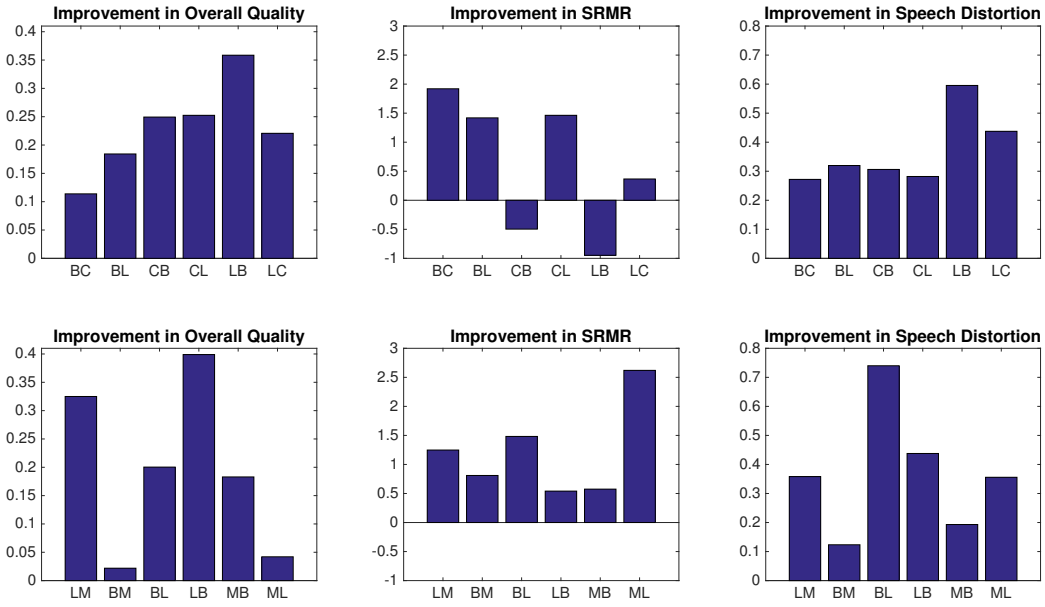

Figure 5.16: Improvement scores. The first and second letters on the x-axis of each bar denote the input and example sound environments, respectively. Top row: Real recordings; B = Bedroom, C = Conference room, L = Living room. Bottom row: Synthesized recordings; L = Lecture hall, B = Booth, and M = Meeting room. As an example, the bar labeled BC on top figures denotes that this is the improvement score for matching input sounds recorded in bedrooms to example sounds recorded in conference rooms.

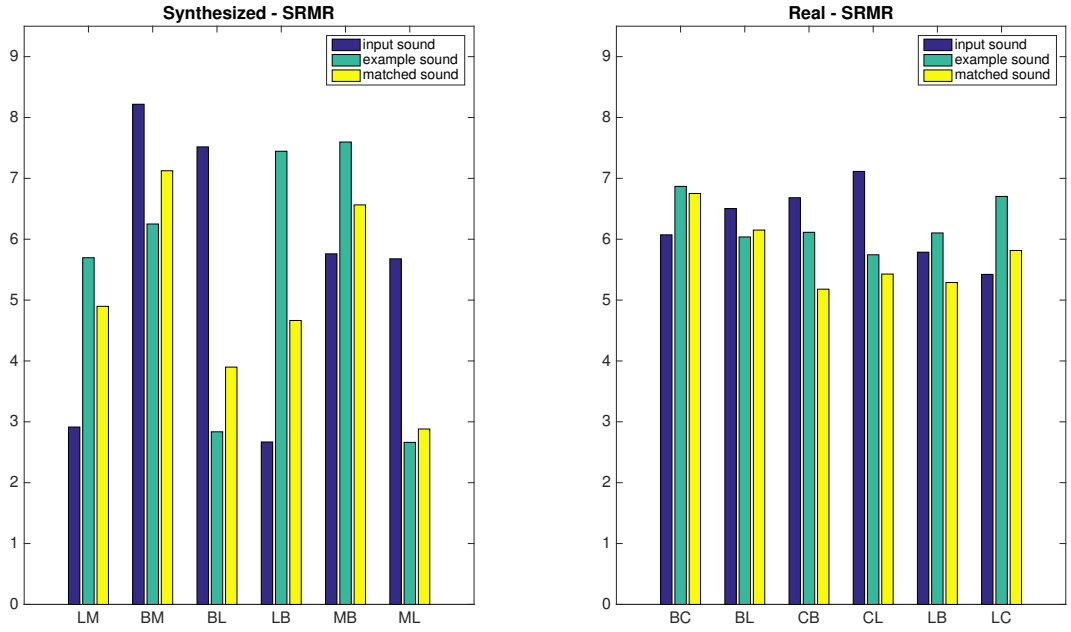

Figure 5.17: Raw SRMR measurements from the experiments shown in Fig. 5.16.

the synthesized dataset. We hypothesize that the negative metrics are due to poor kernel estimations. Other artifacts can also be introduced due to using the reverb sound phase when reconstructing the time domain matched sound.

CHAPTER 6
USER STUDY

6.1 Introduction

In this chapter, we discuss the user study that was conducted to evaluate the proposed acoustic matching system (the algorithms behind the example-based audio editing as well) subjectively.

6.2 User Study

The goal of this user study was to understand if an acoustics matching system is useful in increasing the efficiency and the quality, as oppose to manually editing the acoustics, of the recording. We interviewed 31 people. We categorized the users into three categories: experts, mildly experienced, and novice users. This was done through an initial interview with each user to identify individual skills in music production and audio editing.

We introduced the concept of acoustics matching to each user and allowed the users to get familiarized with the user interface through a set of tasks. Visual and written aids were available to the users that provided direction for completing each task. We used the Matlab Graphical User Interface (GUI) toolbox [97] to design our proposed acoustics matching system, along with a manual editor, to simulate the graphical equalizer and the reverberation toolbox in current DAWs. Users performed the study on a Macbook Air computer and listened to recordings using JVC headphones [98].

6.3 Tasks

Each user was asked to perform 15 tasks manually using a prototype DAW editor and then using the proposed acoustics matching system. Users were given 3 minutes to complete each task. The goal of each task is to modify the input recording until it acquires a specific quality in the example sound. The first nine tasks involved adjusting the equalization, the next three dealt with reverberation matching, and the last three involved both equalization and reverberation. At the end of each task, the user was asked to rate the tasks shown in Table 6.1.

Table 6.1: Metrics for evaluating each task.

Evaluate for / Rating	**1**	**2,3,4**	**5**
Ease of use-manual	hard	$\cdots$	easy
Contentment-manual	frustrated	$\cdots$	straightforward
Confidence in result-manual	not at all	$\cdots$	certain
Automatic quality	bad	$\cdots$	great
Automatic efficiency	too much work	$\cdots$	fast

Users were asked to performed the following steps for completing each task:

1. Load the noted input and example sounds from the manual.
2. Manually edit the input sound using the provided tools.
3. Listen to the input sound concatenated with the example sound.
4. Repeat steps 2 and 3 within 3 minutes or until desired result is achieved.
5. Answer the first three questions in the form (i.e. rows 1 to 3 in Table 6.1).
6. Reset the editor and load the sounds from step 1.
7. Automatically match the input sound acoustic to the example sound by clicking the provided acoustics matching button.
8. Listen to the result and compare it with your manual result.
9. Answer the last two questions on the form (i.e. rows 4 and 5 in Table 6.1).

To make sure exact results can be achieved manually, we used the same content recordings for both the input and the example sounds. We used our GUI to create the desired effects on the input and example sounds. Input and example sounds were each 5 seconds long and were sampled at 1600 Hz. The STFT of the proposed acoustic matching system has the following parameters: periodic square root Hanning window with 513 samples (1024 FFT points) and 75% overlap.

6.3.1 Task 1

In the first task, users were asked to perform equalization matching. We increased the gain on the input sound to 16 dB, 10 dB, and 4 dB over the following frequency regions: $f < 500$ Hz, $f > 2000$ Hz, and 1000 Hz $< f < 4000$ Hz (nine recording in total). The example sounds have a regular equalization pattern. Users were asked to undo the gain from the input sound to match the one in the example sound. Users first started with the boosted low-frequency recordings, starting from 16 dB down to 4 dB. The user then followed the same procedure for the high-frequency and mid-frequency affected recordings. Users were provided with a graphical equalizer, shown in Fig. 6.1, to manually adjust the gain for selected frequencies. The acoustic matching for this task was done by pressing a button called "Match EQ". Table 6.2 depicts the specification on the input recoding.

Table 6.2: Task 1.

Region	**Gain**$_1$	**Gain**$_2$	**Gain**$_3$
f $<$ 500 Hz	16 dB	10 dB	4 dB
f $>$ 2000 Hz	16 dB	10 dB	4 dB
1000 Hz $<$ f $<$ 4000 Hz	16 dB	10 dB	4 dB

The user first manipulated each recording using the graphical equalizer shown in Fig. 6.1. Users employed their listening skills to manually adjust the bars on the graphical equalizers until desired results are achieved. Acoustic matching graphical equalizer interpolates selected frequencies and gain using cubic interpolation. The interpolated filter is then applied to the input sound using Eqs. 3.8-10.

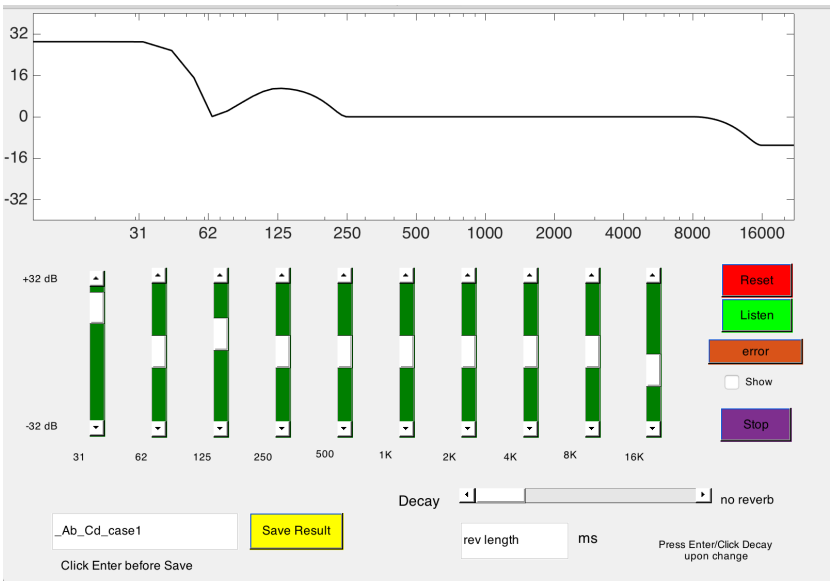

Figure 6.1: Acoustics matching system graphical equalizer.

The automatic equalization matching uses the algorithm described in Chapter 3. The user can automatically match the equalization by pressing a button on the main page of the GUI called "EQ Match", as shown in Fig. 6.2. We counted the first six equalization recordings (i.e. low- and high-frequency recordings) as practice tasks so the user can become familiarize with the interface. We only evaluated the last three equalization recording (mid-frequency recordings) when evaluating the proposed equalization matching.

6.3.2 Task 2

The goal of this task was to reverberate the input recordings (three recordings total) until it contained the same amount of reverb in the provided example sound. Users were asked to adjust the parameter ζ from Eq. 5.13 using a bar called "Decay", as shown in Fig. 6.1 until the desired reverb is achieved. For simplicity other parameters in Eq. 5.13 were set to their true value. We synthesized the kernel based on this parameter and convolved it with the dry input recording to reverberate the sound. Example recordings were reverberated in the same manner using a 0.8 second long room response with $\zeta = [0.25, 0.91, 1.73]$, respectively. We employed the reverb matching algorithm discussed in Section 5.11 to provide the user with matched results upon pressing the "Reverb" button. Table 6.3 summarizes the recordings for

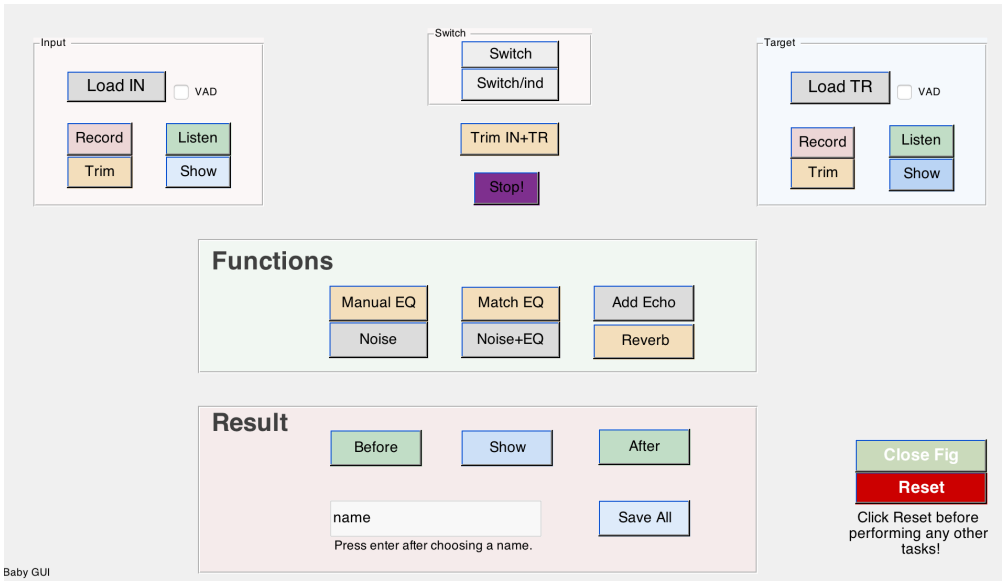

Figure 6.2: Acoustics matching system GUI.

task two.

Table 6.3: Task 2.

Reverb Matching	
ζ	**Translation**
0.25	easy (lots of reverb)
0.91	medium (medium amount of reverb)
1.73	hard (low amount of reverb)

6.3.3 Task 3

The propose of this task was to test the users on more practical cases, where both the equalization and reverb needed adjusting. Input recordings were clean and dry. Users were asked to add equalization filters and reverberating the recordings until they matched the ones in the example sound (three recordings total). We first adjust the equalization for each of the frequency regions in Table 6.2 to 10 dB. We then reverberate the resulting recordings by fixing ζ to 0.6 from Eq. 5.13. We asked the user to use both the equalization and reverberation toolbox at no particular order to recreate these effects in the input sound.

The guideline for automatically matching the equalization and reverb is as follows:

1. Click on automatic equalization matching
2. Click on automatic reverberation matching
3. Click on automatic equalization matching again

Repeating the equalization matching at the end help with compensating for any spectral changes that is caused by the reverb matching algorithm. Table 6.4 summarizes the recordings for task 3.

Table 6.4: Task 3.

Equalization and Reverb Matching			
ζ	**Gain**	**Frequency Regions**	**Translation**
0.6	10 dB	f $<$ 500 Hz	easy
0.6	10 dB	f $>$ 2000 Hz	medium
0.6	10 dB	1000 Hz $<$ f $<$ 4000 Hz	hard

To summarize the user study, each user first performed three equalization matching tasks, starting with more obvious ones, then go on to three reverb matching cases, also starting from more obvious cases, and finally performing both equalization and reverb matching, starting from editing the reverberated low-, high-, and finally mid-frequency recordings.

6.4 Results

We evaluated the proposed acoustic matching system based on user's performance and their answers to Table 6.1 for each of the nine tasks. The population of the user in each category is shown in Table 6.5.

Table 6.5: User's categories.

Users	
Experts	29.03%
Some Experience	35.48%
Novice	32.26%

We evaluated the results of the study based on users' performance and their responses in the form. Figures 6.3-6.5 depicts the ease of use, contentment, and the confidence for performing each task manually. Most users seemed to think similarly of the complexity of each task. Having more experience in audio editing seem to be correlated with more confidence for most of the tasks. Users seem to find reverb matching easier to manipulate (i.e. less complex) and more content (i.e. less frustrating) than other tasks, which, as a result, increased their confidence. The final task seems to be the most frustrating and complex, with the lowest confidence score.

Figures 6.6-6.7 represent the efficiency and the quality of the proposed acoustic matching system. All users agree upon the higher quality and the efficiency of the acoustic matching system over the manual editing. Experts seem to rate the quality slightly lower than others.

We also depicted an improvement score in Figs. 6.8-6.10 by calculating the difference between the Kullback-Leibler distance (KLD) between the normalized power spectra (i.e. sum to one) of the recordings noted in Equation 6.13 for tasks 1 and 3 and the normalized power spectra of the synthesized kernels for task 2 [99].

$$\text{improvement} = \frac{D(\text{in}, \text{ex}) - D(\text{mat}, \text{ex})}{D(\text{in}, \text{ex})} \tag{6.1}$$

where D denotes the KLD distance. We depicted the improvement score for completing each task manually for each category as well the acoustic matching score. As expected, all users performed better when the effect was more audible or less complex to work with (i.e. less parameters to control). More experienced users performed better when matching the reverberation, but struggled when matching both the equalization and reverberation. Experts seem to be performing better than other users for all tasks. The acoustic matching system outperformed users in all tasks.

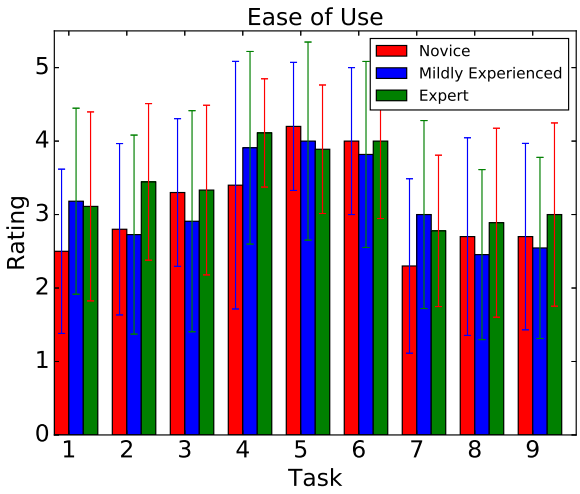

Figure 6.3: Ease of use vs. task.

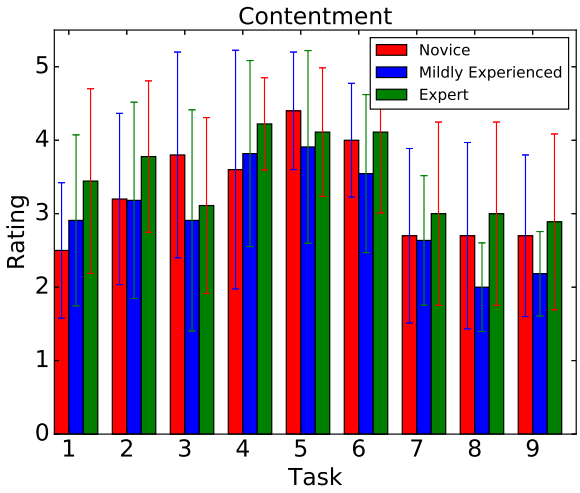

Figure 6.4: Contentment vs. task.

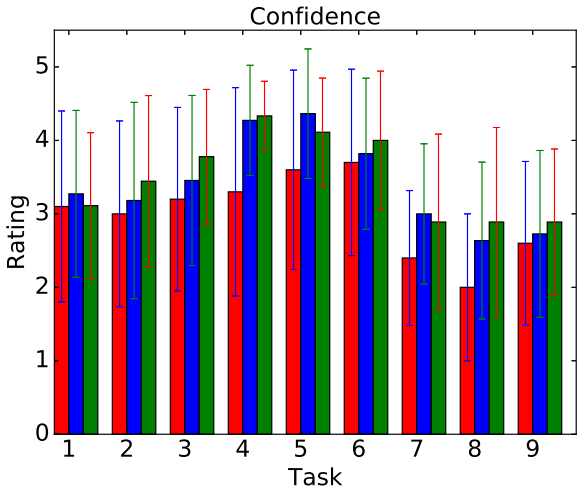

Figure 6.5: Confidence vs. task.

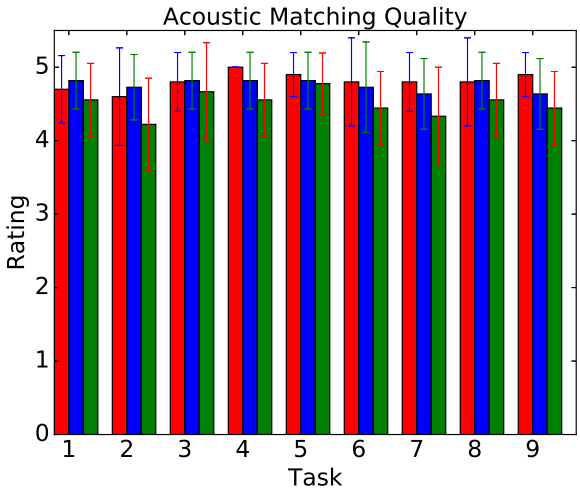

Figure 6.6: Acoustic matching quality vs. task.

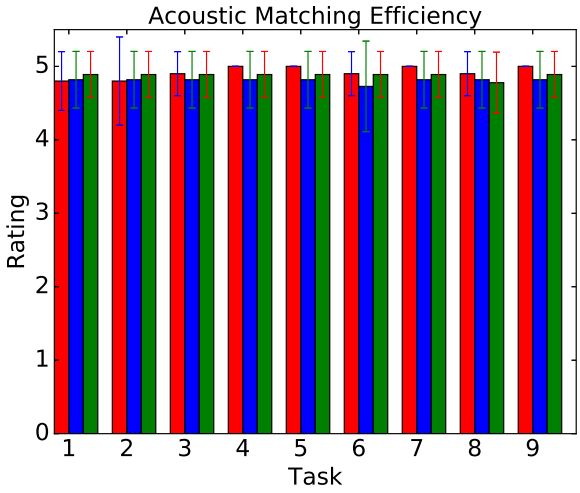

Figure 6.7: Acoustic matching efficiency vs. task.

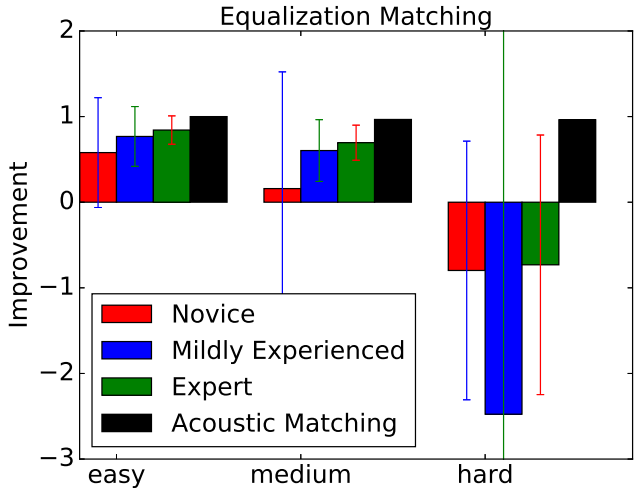

Figure 6.8: Task 1 improvement.

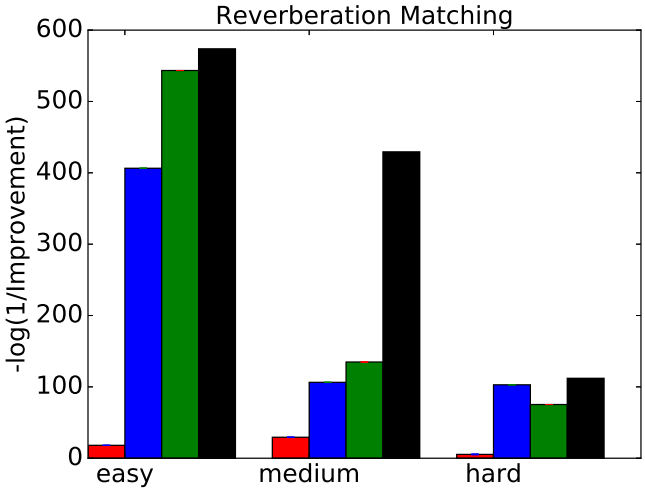

Figure 6.9: Task 2 improvement.

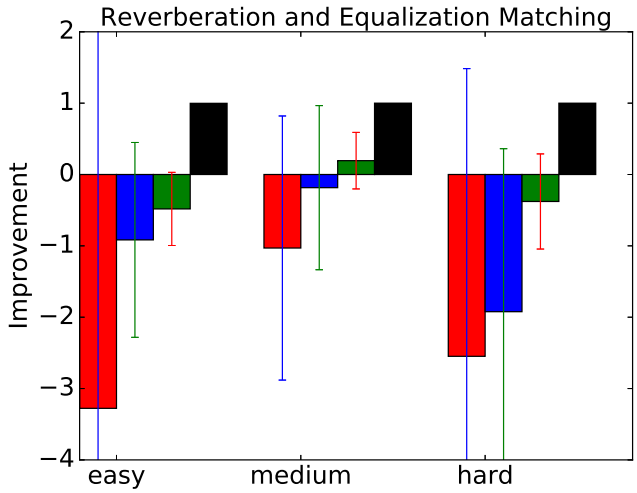

Figure 6.10: Task 3 improvement.

Figure 6.11 confirms that user found the reverb matching task easier than the other two tasks. Figures 6.13-6.16 show that users thought the manual editor was more complex to work with than they initially rated after completing each task.

To summarize, all users perform better when the effect is more audible or less complex to edit (e.g. recording numbers 1, 2, 4, 5, and 6). Users perform better when matching the reverb, which also correlates with their confidence in performing better, as shown previously in Fig. 6.5. Users seem to struggle when adjusting both equalization and reverberation at the same time. Some users, however, found out by the end of this task that reverberating a recording could also affect its equalization and that was the reason it made it harder for them to adjust. Most users, especially the novice and mildly experienced users, struggled when the equalization filter was harder to detect. Experts seem to perform better than other users overall.

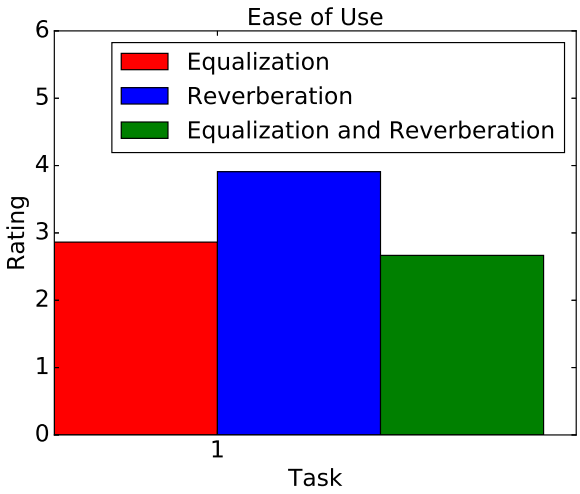

Figure 6.11: Average complexity vs. task.

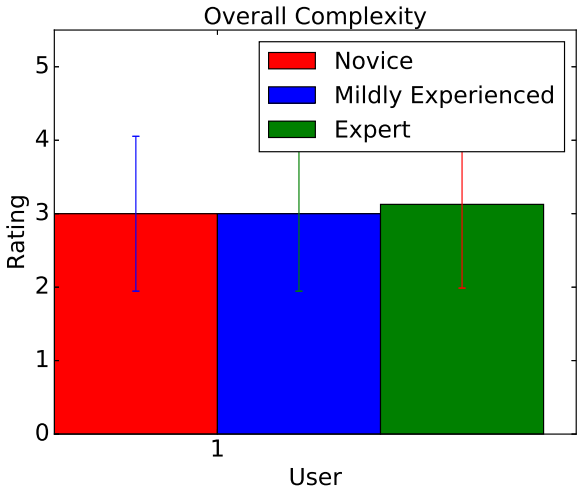

Figure 6.12: Overall complexity vs. user.

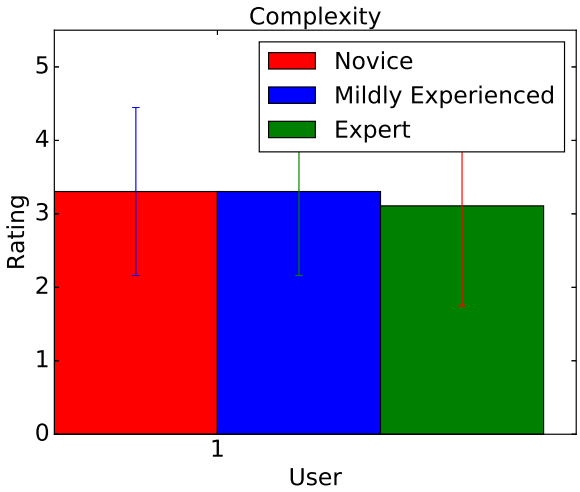

Figure 6.13: Average complexity vs. user.

CHAPTER 7
RESULTS

7.1 Conclusion

In this thesis, we proposed a new approach to editing audio recordings through example sounds. EBAE is an intelligent audio editor that learns from an example that already contains the effect. EBAE automatically adds user's desired effects to any input recording with the touch of a button. EBAE can also create higher-quality recordings more efficiently than the manual editing procedures in current DAWs. EBAE focuses on automating three tasks of equalization, noise, and reverberation which characterise the acoustics of most audio recordings accurately.

In order to match the equalization, we proposed matching the estimated power spectra of the recordings by manipulating their magnitude STFTs.

In order to match the noise, we proposed a decomposition technique using classical denoising approach in signal processing and source separation using NMF for non-stationary noise. The estimated noise components are then matched and added to the clean input sound.

In order to match the reverberation, we employed CNMF to decompose input and example sounds to dry activation matrix and a kernel. We then reconstructed the match recording using an exponentiated input kernel.

We conducted a user study to evaluate the proposed audio editor subjectively. Overall, regardless of their experience in the field of audio editing and music production, most users agreed that EBAE creates higher-quality results more efficiently when matching the acoustics between two recording.

7.2 Future Work

There are a number of ways to improve upon EBAE. Besides the improvements that can be made to the matching algorithms in terms of efficiency for running on mobile devices and the user interface, we think the following features could contribute to an improved user experience.

EBAE only focused on matching the acoustics of the recordings. There are, however, a number of audio editing routines that can be easily automated as well. For example, users can mimic an undesired noise in a recording to be removed from a recording [100]. Another useful feature would be to automatically match the dynamic range of two recordings [83].

We believe that using the input sound phase as the phase for the matched sound is responsible for some of the artifacts introduced to the matched sound when matching the noise and the reverb for some of the recordings. Any improvement to on estimating the phase of the matched recording could lead to better quality results.

Another improvement would be to automatically understand the features a user would like to apply as oppose to having the user choosing which aspects of the sound they would like to modify.

It would also be beneficial if the the example-based audio editor is able to learn from user's common editing routines and optimize the parameters for matching the acoustics of the two recordings that is personalized for that user.

We also think it would interesting if the user can provided the system with input, example, and a desired edited input sounds and have the system coming up with an automated procedure to perform that particular editing task for other recordings as well.

REFERENCES

[1] G. Delight, "Camera plus pro v4.0: A new dawn in iphoneography," 2011. [Online]. Available: http://www.globaldelight.com/blog/camera-plus-pro-v4/

[2] W. of the Wall, "New photos from the game of thrones season 6 finale the winds of winter," June 2016, Pictures. [Online]. Available: http://watchersonthewall.com/new-photos-game-thrones-season-6-finale-winds-winter/

[3] Apple, "Podcast makers," 2016, Resource. [Online]. Available: http://itunespartner.apple.com/en/podcasts/faq

[4] R. F. School, "Sound vs. Picture (RJFS Field Test)," July 2016, Video. [Online]. Available: https://www.youtube.com/watch?v=SDLm_q5J7D8

[5] H. Fork, "How to: A perfect reconstruction graphic equalizer," 2007. [Online]. Available: http://blog.hostilefork.com/perfect-reconstruction-equalizer/

[6] P. Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, 1st ed. California Technical Publication, 1997.

[7] B. J. Bazuin, "Lecture Notes on Discrete-Time Signal Processing," 2015. [Online]. Available: http://kilyos.ee.bilkent.edu.tr/~ee424/EE424.pdf

[8] B. V. Veen, Online Signal Processing Course. [Online]. Available: http://allsignalprocessing.com

[9] P. C. Loizou, *Speech Enhancement: Theory and Practice*, 2nd ed. CRC Press, 2013.

[10] R. Anushiravani, "3D audio playback through two loudspeakers," B.S. Thesis, Universiry of Illinois at Urbana Champaign, Urbana, 2014.

[11] M. F. Duarte, "Changing Sampling Rates in Discrete Time," 2014. [Online]. Available: https://cnx.org/exports/5c879a76-e347-48c1-ba53-ef7e3225ec67@2.pdf/changing-sampling-rates-in-discrete-time-2.pdf

[12] A. V. Oppenheim and G. C. Verghese, *Power Spectral Density.* Introduction to Control and Signal Processing, 2010.

[13] iitg.vlab.co.in, "Non-stationary nature of speech signal," Virtual Lab. [Online]. Available: http://iitg.vlab.co.in/?sub=59&brch=164&sim=371&cnt=1104

[14] M. Bosi and R. E. Goldberg, *Introduction to Digital Audio Coding and Standards.* Springer Science, 2003.

[15] J. B. Allen and L. R. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Acoustics, Speech, Signal Processing*, vol. 13-25, 1977.

[16] Jont B. Allen, "Application of the short-time Fourier transform to speech processing and spectral analysis," in *International Conference on Acoustics, Speech, and Signal Processing*, 1982.

[17] Julius O. Smith, "Mathematical definition of the STFT," Spectral Audio Signal Processing, 2011. [Online]. Available: https://ccrma.stanford.edu/~jos/sasp/Mathematical_Definition_STFT.html#19930

[18] R. Crochiere, "A weighted overlap-add method of short-time Fourier analysis/synthesis," *Acoustics, Speech, Signal Processing*, vol. 12-28, 1980.

[19] DSP-Related, "Matlab for the Hann window," Online DSP Course, 2011. [Online]. Available: https://www.dsprelated.com/freebooks/sasp/Matlab_Hann_Window.html

[20] Y. Lu and P. C. Loizou, "A geometric approach to spectral subtraction," in *Interspeech Conference*, 2008.

[21] M. A. A. El-Fattah, M. I. Dessouky, S. M. Diab, and F. E. A. El-Samie, "Speech enhancement using an adaptive Wiener filtering approach," *Progress Electromagnetic Research*, vol. 4, pp. 167–184, 2008.

[22] T. Virtanen, "Non-negative matrix factorization and its application to audio," Slide, 2011. [Online]. Available: https://www.lsv.uni-saarland.de/fileadmin/teaching/dsp/ss15/nmf.pdf

[23] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, vol. 13, 2000.

[24] Adobe, "Adobe audition cc: Removing noise," Video. [Online]. Available: http://tv.adobe.com/watch/companion-videos-for-inspire/adobe-audition-cc-removing-noise/

[25] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-squared error short-time spectral amplitude estimator," in *Signal Processing Letters*, 1984.

[26] P. C. Loizou, *Speech Enhancement: Theory and Practice*, 2nd ed. CRC Press, 2013.

[27] Y. Liao, "Phase and frequency estimation: High-accuracy and low-complexity techniques," M.S. thesis, Worcester Polytechnic Institute, May 2011. [Online]. Available: https://web.wpi.edu/Pubs/ETD/Available/etd-042511-144644/unrestricted/YLiao.pdf

[28] T. Esch and P. Vary, "Efficient musical noise suppression for speech enhancement systems," *Institute of Communication Systems and Data Processing RWTH Aachen University, Germany*, 2009.

[29] A. Lukin and J. Todd, "Efficient musical noise suppression for speech enhancement systems," in *International Conference on Acoustics, Speech and Signal Processing*, 2009.

[30] S. V. Vaseghi, *Wiener Filters*. John Wiley and Sons LTD, 2000.

[31] P. C. Loizou, *Speech Enhancement: Theory and Practice*, 2nd ed. CRC Press, 2013.

[32] MathWorks, "Create apps with graphical user interfaces in Matlab." [Online]. Available: http://www.mathworks.com/discovery/matlab-gui.html

[33] M. Dubey, "Evaluation of signal processing methods for speech enhancement," B.S. Thesis, University of Illinois at Urbana Champaign, Urbana, 2016.

[34] S. Rangachari and P. C. Loizou, "Noise-estimation algorithm for highly non-stationary environments," in *Speech Communication*, 2005.

[35] M. H. Moattar and M. M. Homayounpour, "A simple but efficient real-time voice activity detection algorithm," in *European Signal Processing Conference*, 2009.

[36] A. Hyvrinen and E. Oja, "Independent component analysis: Algorithms and applications," in *Neural Networks*, 2000.

[37] P. Smaragdis and B. Raj, "Shift-invariant probabilistic latent component analysis," Tech Report. [Online]. Available: https://pdfs.semanticscholar.org/8490/7a76b16df13a7bf5bd099e42ba98c0f710c4.pdf

[38] A. Mirzal, "Independent component analysis: Algorithms and applications," in *Advanced Data Analysis Classification*, 2014.

[39] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, pp. 71–86, 1991.

[40] K. W. Wilson, B. Raj, P. Smaragdis, A. D. Heusdens, and J. Jensen, "Speech denoising using nonnegative matrix factorization with priors," in *International Conference on Acoustics, Speech and Signal Processing*, 2008.

[41] N. Mohammadiha, P. Smaragdis, and S. Doclo, "Joint acoustic and spectral modeling for speech dereverberation using non-negative representations," in *International Conference on Acoustics, Speech and Signal Processing*, 2015.

[42] K. Kumar, "A spectro-temporal framework for compensation o reverberation for speech recognition," Ph.D. dissertation, Carnegie Mellon University, 2011.

[43] P. D. O'Grady and B. A. Pearlmutter, "Convolutive non-negative matrix factorisation with sparseness constraint," in *Machine Learning for Signal Processing*, 2006.

[44] P. Smaragdis, "Convolutive speech bases and their application to supervised speech separation," in *Speech And Audio Processing*, 2007.

[45] M. Kim and P. Smaragdis, "Efficient model selection for speech enhancement using a deflation method for nonnegative matrix factorization," in *Global Conference on Signal and Information Processing IEEE*, 2014.

[46] K. W. Wilson, B. Raj, and P. Smaragdis, "Regularized non-negative matrix factorization with temporal dependencies for speech denoising," in *Interspeech Conference*, 2008.

[47] Z. Duan1, G. J. Mysore, and P. Smaragdis, "Speech enhancement by online non-negative spectrogram decomposition in non-stationary noise environments," in *Interspeech Conference*, 2012.

[48] I. Acoustical Surfaces and I. Bonded Logic, "Whole system acoustical treatments." [Online]. Available: https://continuingeducation.bnpmedia.com/course.php?L=217&C=893&P=4

[49] C. A. Joshi, "Acoustic echo cancellation," 2012. [Online]. Available: http://www.slideshare.net/chintanajoshi/acoustic-echo-cancellation

[50] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," in *Special Interest Group on Computer Graphics and Interactive Techniques*, 2008.

[51] S. Cho and S. Lee, "Fast motion deblurring," in *Special Interest Group on Computer Graphics and Interactive Techniques*, 2009.

[52] M. Maule and A. L. B. da Ros, "Wiener filter applied to deconvolution," Universidade Federal do Rio Grande do Sul Instituto de Informtica. [Online]. Available: http://www.inf.ufrgs.br/~mmaule/classes/CMP562/wiener.pdf

[53] I. P. Toolbox, "Deblurring with the Wiener filter," Matlab. [Online]. Available: https://www.mathworks.com/help/images/examples/deblurring-images-using-a-wiener-filter.html?requestedDomain=www.mathworks.com

[54] W. Harper, "Explanation of Wiener deconvolution," 2007, OpenStax-CNX. [Online]. Available: https://archive.cnx.org/contents/eed4dcd4-1b95-4a66-9fae-e16add05cc13@1/explanation-of-wiener-deconvolution

[55] S. K. Nagendra and V. Kumar.S.B, "Echo cancellation in audio signal using LMS algorithm," in *National Conference on Recent Trends in Engineering Technology*, 2011.

[56] MathWorks, "Peak analysis," 2002. [Online]. Available: http://www.mathworks.com/help/signal/examples/peak-analysis.html

[57] P. Classroom, "Echo vs. reverberation," 2002, tutorial. [Online]. Available: http://www.physicsclassroom.com/mmedia/waves/er.cfm

[58] G. Davis, "The sound reinforcement handbook," in *National Conference on Recent Trends in Engineering Technology*, 1987.

[59] D. M. Howard, J. Angus, F. Press, and Butterworth-Heinemann, "Surrounded by sound - a sonic revolution," 2000. [Online]. Available: http://www-users.york.ac.uk/~dtm3/RS/RSweb.htm

[60] Julius O. Smith, "Energy decay curve," Physical Audio Signal Processing, 2010. [Online]. Available: https://ccrma.stanford.edu/~jos/pasp/Energy_Decay_Curve.html

[61] R. Ratnam, D. L. Jones, B. C. Wheeler, W. D. OBrien, C. R. Lansing, and A. S. Feng, "Blind estimation of reverberation time," in *Acoustical Society of America*, 2003.

[62] J. Tuel, "The sounds of science," 2013. [Online]. Available: http://www.vtmag.vt.edu/winter14/sounds-of-science.html

[63] A. Turlapaty, "LTI system: response to a complex exponential," February 2015, Video. [Online]. Available: https://www.youtube.com/watch?v=SDLm_q5J7D8

[64] Wikiwand, "Chirp." [Online]. Available: http://www.wikiwand.com/en/Chirp

[65] M. Jeub, M. Schfer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," in *Proceedings of International Conference on Digital Signal Processing*, 2009.

[66] M. Jeub, M. Schafer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," in *Proceedings of International Conference on Digital Signal Processing*, 2009.

[67] J.-M. Jot, L. Cerveau, and O. Warusfel, "Analysis and synthesis of room reverberation based on a statistical time-frequency model," in *Institute for Research and Coordination in Acoustics/Music*, 1997.

[68] J. Allen and D. Berkley, "Image method for efficiently simulating small-room acoustics," in *Journal of the Acoustical Society of America*, 1979.

[69] Apple, "Airpods," 2016, Resource. [Online]. Available: http://www.apple.com/airpods/

[70] E. A. Habets and J. Benesty, "Joint dereverberation and noise reduction using a two-stage beamforming approach," in *Joint Workshop on Hands-Free Speech Communication and Microphone Arrays*, 2011.

[71] T. Dietzen, N. Huleihel, A. Spriet, W. Tirry, S. Doclo, M. Moonen, and T. van Waterschoot, "Speech dereverberation by data-dependent beamforming with signal pre-whitening," in *European Signal Processing Conference*, 2015.

[72] N. D. Gaubitch and P. A. Naylor, "Analysis of the dereverberation performance of microphone arrays," in *International Workshop on Acoustic Echo and Noise Control*, 2005.

[73] N. D. Gaubitch, P. A. Naylor, and D. B. Ward, "On the use of linear prediction for dereverberation of speech," in *International Workshop on Acoustic Echo and Noise Control*, 2013.

[74] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B.-H. Juang, "Blind speech dereverberation with multi-channel linear prediction based on short time Fourier transform representation," in *International Conference on Acoustics, Speech and Signal Processing*, 2008.

[75] D. Liang, M. D. Hoffman, and G. J. Mysore, "Speech dereverberation using a learned speech model," in *International Conference on Acoustics, Speech and Signal Processing*, 2015.

[76] P. A. Naylor and N. D. Gaubitch, "Single-microphone spectral enhancement," *Speech Dereverberation*, pp. 64–71, 2009.

[77] K. Kumar, B. Raj, R. Singh, and R. M. Stern, "An iterative least-squares technique for dereverberation," in *International Conference on Acoustics, Speech and Signal Processing*, 2011.

[78] K. Han, Y. Wang, D. Wang, W. S. Woods, I. Merks, and T. Zhang, "Learning spectral mapping for speech dereverberation and denoising," in *ACM Transaction on Audio, Speech,and Language Processing*, 2015.

[79] X. Feng, Y. Zhang, and J. Glass, "Speech feature denoising and dereverberation via deep autoencoders for noisy reverberant speech recognition," in *International Conference on Acoustics, Speech and Signal Processing*, 2014.

[80] Y. Lin, J. Chen, Y. Kim, and D. D. Lee, "Blind channel identification for speech dereverberation using l_1 norm sparse learning," in *Advances in Neural Information Processing Systems*, 2008.

[81] A. Jukic, T. van Waterschoot, T. Gerkmann, and S. Doclo, "Speech dereverberation with convolutive transfer function approximation using map and variational deconvolution approaches," in *International Workshop on Acoustic Signal Enhancement*, 2014.

[82] A. V. Oppenheaim and R. W. SchaferD, *Digital Signal Processing*. Prentice-Hall, 1975.

[83] D. Giannoulis, M. Massberg, and J. D. Reiss, "Digital dynamic range compressor design a tutorial and analysis," 2012. [Online]. Available: https://www.eecs.qmul.ac.uk/~josh/documents/GiannoulisMassbergReiss-dynamicrangecompression-JAES2012.pdf

[84] D. Dorran, E. Coyle, and R. Lawlor, "An efficient phasiness reduction technique for moderate audio timescale modification," in *Digital Audio Effects*, 2004.

[85] P. Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, 1st ed. California Technical Pub, 1997.

[86] Y. Huang, J. Benesty, and J. Chen, *Speech Enhancement*, 1st ed. Springer, 2005.

[87] T. Drugman, B. Bozkurt, and T. Dutoit, "Complex cepstrum-based decomposition of speech for glottal source estimation," in *Interspeech Conference*, 2009.

[88] S. Xizhong and M. Guang, "Complex cepstrum based single channel speech dereverberation," in *Computer Science and Education*, 2009.

[89] H. Padaki, K. Nathwani, and R. M. Hegde, "Single channel speech dereverberation using the LP residual cepstrum," *Indian Institute of Technology Kanpur*, 2013.

[90] R. Talmon, I. Cohen, and S. Gannot, "Relative transfer function identification using convolutive transfer function approximation," *Audio, Speech, and Language Process*, vol. 17, pp. 546 – 555, 2009.

[91] D. P. W. Ellis, "Gammatone-like spectrograms," 2009. [Online]. Available: http://www.ee.columbia.edu/~dpwe/resources/matlab/gammatonegram/

[92] R. Anushiravani, P. Smaragdis, and G. J. Mysore, "Long reverberation matching," U.S. Patent, 2015.

[93] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," in *Journal of Machine Learning Research*, 2004.

[94] G. J. Mysore, "Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech? A Dataset, Insights, and Challenges," in *Signal Processing Letters*, 2014.

[95] T. H. Falk, C. Zheng, and W.-Y. Chan, "A non-intrusive quality and intelligibility measure of reverberant and dereverberated speech," in *Audio, Speech, and Language Processing*, 2010.

[96] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement," in *Audio, Speech, and Language Processing*, 2008.

[97] Lectrosonic, "Articles," 2014. [Online]. Available: http://www.lectrosonics.com/Support/Table/White-Papers/ASPEN/

[98] MathWorks, "JVC over-the-ear comfortable stereo headphones." [Online]. Available: https://amzn.com/B008JFM7E0

[99] D. Pinto, "The Kullback-Leibler distance," in *International Conference on Intelligent Text Processing and Computational Linguistics*, 2007.

[100] P. Smaragdis, "User guided audio selection from complex sound mixtures." in *ACM Symposium on User Interface Software and Technology*, 2009.