UNIVERSITY OF CALIFORNIA SAN DIEGO

Perceptually-inspired signal processing for modulation in musical sound mixtures

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Music

by

Jeremy Hyrkas

Committee in charge:

Professor Tamara Smyth, Chair
Professor Sarah Creel
Professor Thomas Erbe
Professor Miller Puckette

2025

The Dissertation of Jeremy Hyrkas is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2025

TABLE OF CONTENTS

Dissertation Approval Page iii

Table of Contents iv

List of Figures vi

List of Tables xi

Acknowledgements xii

Vita xiv

Abstract of the Dissertation xv

Introduction 1

Part I Controlling auditory roughness 5

Chapter 1 Auditory roughness, tuning and its relation to sound mixtures 6
1.1 Introduction 7
1.2 Auditory roughness, musical consonance and tuning 8
1.2.1 Roughness and critical bandwidth 8
1.2.2 Computational models of roughness 10
1.3 Applications of roughness models in computer music 12
1.3.1 Roughness models applied to music theory 12
1.3.2 Roughness models applied to music making 14
1.4 Conclusion 17

Chapter 2 Algorithms to control roughness by changing the parameters of sound partials 18
2.1 Introduction 19
2.2 Roughness control algorithms: the shared model 20
2.3 Changing roughness by partial tuning 20
2.4 Changing roughness by amplitude adjustment 25
2.5 Changing binaural roughness using spatial panning 30
2.6 Conclusion 33

Chapter 3 Implementations and evaluations of roughness control algorithms 34
3.1 Introduction 35
3.2 Offline implementations on stored audio signals 35
3.2.1 Offline application to audio mixing 38
3.3 Real-time implementations for additive synthesis 41
3.3.1 Online algorithms applied to composition 44
3.4 Conclusion 47

Part II Vibrato matching . . . 50

Chapter 4 Perceptual motivation for vibrato matching and existing vibrato models . . . 51
4.1 Vibrato and source separation . . . 52
4.2 Vibrato analysis and sinusoidal resynthesis . . . 54
4.3 Vibrato using a time-varying delay line . . . 56
4.4 Conclusion . . . 60

Chapter 5 Methods for vibrato transfer . . . 61
5.1 Introduction . . . 62
5.2 Vibrato Transfer: methods for FM . . . 62
5.2.1 Deriving a delay function from the instantaneous fundamental frequency 62
5.2.2 Estimating instantaneous frequency in real signals . . . 63
5.3 Vibrato Transfer: incorporating AM . . . 70
5.4 The vibrato transfer algorithm and sound blending . . . 74
5.5 Conclusion . . . 76

Chapter 6 Methods for vibrato suppression . . . 81
6.1 Introduction . . . 82
6.2 Vibrato suppression: frequency demodulation . . . 82
6.2.1 Further Vibrato Reduction . . . 86
6.2.2 Deriving delay functions and demodulating real signals . . . 89
6.3 Vibrato suppression: amplitude demodulation . . . 92
6.3.1 Bandpass Plus Spectral Envelope (BPSE) . . . 92
6.3.2 Spectrogram smoothing (SS) . . . 93
6.3.3 Comparison of AM demodulation methods . . . 94
6.4 Combining vibrato suppression and vibrato transfer . . . 98
6.4.1 Vibrato matching for sound blending . . . 99
6.4.2 Vibrato matching and unison source separation . . . 102
6.5 Conclusion . . . 107

Conclusion . . . 109

Bibliography . . . 112

LIST OF FIGURES

Figure 1.1. Idealized consonance curve of two sinusoids based on their frequency difference modeled as critical bandwidth. Reprinted with permission from Plomp, R., & Levelt, W. J. M. (1965). 9

Figure 1.2. Equations (1.1) and (1.2), which model dissonance based on Plomp and Levelt's dissonance curve. The equations model the area of maximum dissonance slightly differently. Dissonance values are unitless and scaled to have a maximum value of 1. 11

Figure 2.1. Frequency bashing pairs of partials with frequencies (440 Hz, 470 Hz) on the left and (880 Hz, 910 Hz) on the right. 22

Figure 2.2. The spectrum of a major chord using sawtooth waves. The original spectra (left) is modified using either frequency bashing (center) or Adaptive Tuning (right). 24

Figure 2.3. The spectrum of a major chord using stretched inharmonic notes. The original spectra (left) is modified using either frequency bashing (center) or Adaptive Tuning (right). 25

Figure 2.4. The estimated masking curve of a sinusoid with frequency 440 Hz is shown on the left and 910 Hz on the right, each with an SPL of 40 dB. 27

Figure 2.5. Amplitude whacking pairs of partials with frequencies (440 Hz, 470 Hz) on the left and (880 Hz, 910 Hz) on the right. 28

Figure 2.6. **Left**: the spectrum of sawtooth waveforms forming a major triad in equal temperament tuning. **Center**: a zoomed region of the spectrum after frequency bashing partials. Three partials have been moved. **Right**: the same zoomed region of the spectrum after amplitude whacking. 29

Figure 2.7. Example per-ear gain multipliers for the amplitude of a signal played from different locations around a listener's head. 31

Figure 3.1. Partials of a signal x are adjusted by removing them using notch filters, and in parallel, isolating them from x using amplitude-complementary peaking bandpass filters. 38

Figure 3.2. A tonally dissonant chord has its roughness reduced by the offline amplitude whacking algorithm. The spectral difference between the original (left) and processed track (middle) is shown in the right, showing a level adjustment of eight partials. 39

Figure 3.3. **Left**: a spectrogram of a dynamic horn line mixture featuring audio of three players performing. **Right**: the spectral difference of an amplitude whacked version of the horn line and the original. 40

Figure 3.4. Frequency bashing to create dissonance as an audio effect. 41

Figure 3.5. Tremolo at 3 Hz (left) is compared to hard bashing to a fixed difference of 3 Hz (right). 42

Figure 3.6. Initial placement of sound sources in "The Hearing Test." Sources 1–25 are sinusoids and are panned algorithmically around the circle as the piece progresses. Source 26 plays samples from an instructional video and is manually panned throughout the piece. 45

Figure 3.7. The spectrum of the opening drone in "The Hearing Test." The piece algorithmically slides between the original spectrum (left) and the amplitude whacked spectrum (right) while dialogue from an eye exam instructional video plays. 46

Figure 4.1. The signal created by modulating a sum of harmonically-related sinusoids (4.9) using a TVDL has a factor of k increase in the spectral spread (left) and time-varying instantaneous frequency modulation depth (right) in the kth harmonic as seen by (4.10). 58

Figure 4.2. The spectrum and frequency deviation of harmonics of a vocalist with vibrato. 58

Figure 4.3. The spectrum and frequency deviation of harmonics of a saxophone with vibrato. 58

Figure 4.4. The spectrum and frequency deviation of harmonics of a flute with vibrato. 58

Figure 4.5. Harmonic peak picking versus f_0 signal times harmonic number. The frequency contour of upper harmonics closely matches those of the first harmonic, with visible deviations more likely to be caused by analysis error than true differences in the frequency tracks. 59

Figure 5.1. First "fundamental" spectral magnitude peak of a sawtooth wave with a sinusoidal vibrato applied. Every 100^{th} frame is offset to show time evolution (from bottom to top) of the peak position as it follows a sinusoidal vibrato track. 64

Figure 5.2. The estimated instantaneous frequency f_i^p and modulating delay function calculated from a vocal signal with vibrato. The delay function is detrended due to its linear growth. 66

Figure 5.3. A signal is processed using a bandpass filter to isolate the first harmonic and use its analytic signal to create the estimated delay function. A frequency domain analysis is depicted to determine the signal's f_0, but time domain methods could be applied instead. 67

Figure 5.4. The estimated instantaneous frequency f_i^b and modulating delay function calculated from a guitar signal with vibrato. 68

Figure 5.5. Estimated instantaneous frequency for sawtooth (top) and clarinet (bottom) using peak picking and bandpass filter methods. 69

Figure 5.6. The signal flow for calculating the AM envelope for both harmonics and the spectral envelope of the non-harmonic portions of a signal. 72

Figure 5.7. Spectral envelope modulation involves down-sampling in the frequency domain to d distinct points, tracking the amplitude over time and creating a modulating envelope of each point. 73

Figure 5.8. Vibrato transfer between flute signals. 75

Figure 5.9. Vibrato transfer between vocal signals. 75

Figure 5.10. Vibrato transfer between saxophone signals. 76

Figure 5.11. Mixing signals with and without vibrato. 77

Figure 5.12. The top plot shows a mix of two signals: a flutist playing a note with vibrato and playing the same note without vibrato. The bottom shows the same mix after vibrato patterns from the first signal have been imparted onto the static note using vibrato transfer. 78

Figure 5.13. The top plot shows a mix of two signals: a saxophonist playing a note with vibrato and playing the same note without vibrato. The bottom shows the same mix after vibrato patterns from the first signal have been imparted onto the static note using vibrato transfer. 79

Figure 6.1. Spectral magnitude of the delay line output $z_d(n)$ (6.2) and its approximation $\hat{z}_d(n)$ (6.13) for FM parameters $I = 1$, $\omega_c = 2\pi 200$ and $\omega_m = 2\pi 5$. . . 85

Figure 6.2. Spectral magnitude of the delay line output $z_{dd}(n)$ showing no sidebands greater than -60dB and thus greater vibrato reduction than $z_d(n)$. 88

Figure 6.3. Using a TVDL to demodulate $z_{k,m}(n)$. $D_d(n)$ removes most sidebands but leaves some audible. Using the recursive delay $D_{dd}(n)$ shows stronger demodulation in both the fundamental and upper harmonics. 88

Figure 6.4. Frequency demodulation of the first harmonic of a flutist and a vocalist playing with heavy vibrato. 90

Figure 6.5. FM demodulating a flute. The demodulating delay function based on the first harmonic also demodulates the upper harmonics. Discontinuities the plot are due to analysis error and not present in the audio signals. 90

Figure 6.6. The ratio of harmonics (ω_k/ω_1) when suppressing vibrato. Errors tracking the frequency of quieter harmonics can lead to a loss of harmonicity in sinusoidal resynthesis (left). The TVDL treats all harmonics collectively and the original harmonicity is preserved (right). 91

Figure 6.7. Removing AM from a signal using the RMS envelopes of its harmonics and the spectral envelope of the residual signal. 93

Figure 6.8. Removing AM from a signal by smoothing the amplitude of each frequency bin of its spectrogram. 94

Figure 6.9. Amplitude demodulation of upper harmonics of a flute. Both the BPSE method and the SS method result in harmonics with less amplitude modulation than the original signal. The low-passed amplitude track used in sinusoidal resynthesis is more demodulated in all cases. 95

Figure 6.10. Spectrograms of a flute sample, original and vibrato suppressed using: sinusoidal resynthesis, TVDL FM demodulation and BPSE AM demodulation, and TVDL FM demodulation and SS AM demodulation. 96

Figure 6.11. Spectrograms of a saxophone sample, original and vibrato suppressed using: sinusoidal resynthesis, TVDL FM demodulation and BPSE AM demodulation, and TVDL FM demodulation and SS AM demodulation. . . 97

Figure 6.12. Spectrograms of a vibraphone sample, original and vibrato suppressed using: sinusoidal resynthesis, TVDL FM demodulation and BPSE AM demodulation, and TVDL FM demodulation and SS AM demodulation. . . 97

Figure 6.13. Two signals of a vocalist singing the same note with different vibrato patterns. The top and middle figures show the source and target signals, and the bottom figure shows the target signal after its vibrato was suppressed and vibrato from the source was transferred onto it. 100

Figure 6.14. The source and target vocal signals from Figure 6.13 are shown in a mix. The pre-vibrato-matched mix (top) appears visually as two sources, where the post-vibrato-matched mix (bottom) is more ambiguous. 101

Figure 6.15. The pre- and post-vibrato-matched sound mixtures of a flutist and a vocalist playing different notes. 103

Figure 6.16. The pre- and post-vibrato-matched sound mixtures of two vocalists singing different notes. 104

Figure 6.17. Using unison source separation based on vibrato patterns on two signals from the same vocalist. 105

Figure 6.18. Unison source separation on bassoon and bass oboe signals with vibrato. Synchronizing vibrato patterns reduces the effectiveness of unison source separation. Degradation is observed in the mixed portion of the first signal and the silent section of the second signal. 106

Figure 6.19. Using unison source separation based on vibrato patterns on vocal and trumpet signals. Synchronizing vibrato patterns reduces the effectiveness of unison source separation, primarily in the silent portion of the second source. 107

LIST OF TABLES

Table 1.1. The total roughness of three intervals with different harmonic structures as calculated by four variations of roughness models. 13

Table 3.1. Per-ear roughness for the closing drone of The Hearing Test. The piece algorithmically switches between panning partials for consonance and dissonance. The table shows three iterations of switches and the resulting roughness calculations. 47

ACKNOWLEDGEMENTS

I would like to acknowledge my committee for their generous advice and support. Tamara Smyth, thank you for chairing my committee, for our many meetings, and for sticking through my rambling drafts to help me become a better writer. Tom Erbe, thank you for your guidance on implementation, practical matters and for geeking out over musical gear. Thank you to Miller Puckette for your advice and guidance on so many topics over the years and for agreeing to take on another committee role in retirement. Thank you to Sarah Creel for lending your valuable expertise in the cognitive and perceptual.

I would also like to acknowledge my other mentors and colleagues in and outside of UCSD. Hannes Gamper at Microsoft Research hired me as an acoustics research intern despite (or perhaps because of) my profile as a music researcher and, together with John Tang and Andy Wilson, taught me so much about research at the intersection of spatial audio and human-computer interaction. Xiao Quan and Jeff Huang at Bose allowed me to continue spatial audio and HCI research in a more product- and user-focused internship. I would like to thank King Britt for his continued support and championing of my teaching. Finally, I would like to acknowledge my friends and research colleagues Pablo Dodero Carrillo, Teresa Díaz de Cossio Sánchez and Sasha Ishov - thank you for the most fun and personally satisfying collaboration of my career so far. I hope we continue working together long after graduation.

Lastly, would like to thank those who helped me personally in these last five years. To my family, whose love and support allowed me to pursue a future in music; to my friends, who ears and words got me through the hardest times; to Caiti, who moved to California for me. From Brooklyn to San Diego all the way back to North Jersey: thank you.

This dissertation is compiled from a variety of published and unpublished manuscripts, in addition to new findings, methods and results. In Part I, Chapter 1 contains material from my Qualifying Exam, submitted to the Music Department and defended as part of my advancement to Ph.D. candidacy. Chapters 2 and 3 contain material from the research paper *Algorithms for Roughness Control Using Frequency Shifting and Attenuation of Partials in Audio*, published in

the Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR 2023). The dissertation author was the primary investigator and author of this work.

In Part II, Chapters 4, 5 and 6 contain material adapted from the following research papers: *On Vibrato and Frequency (De)Modulation in Musical Sounds*, published in the Proceedings of the International Conference on Digital Audio Effects (DAFx 2024), *Real-time implementation of vibrato transfer as an audio effect*, published in the Proceedings of the International Computer Music Conference (ICMC 2025), and *Vibrato Suppression by Time-Varying Delay and Spectral Magnitude Demodulation*, published in the International Symposium On Musical and Room Acoustics (ISMRA 2025), respectively. The dissertation author was the primary investigator and author of these publications, and Dr. Tamara Smyth was coauthor on the publications accepted to DAFx 2024 and ISMRA 2025.

VITA

2012	Bachelor of Science, Computer Science, Colorado State University
2015	Master of Science, Computer Science and Engineering, University of Washington
2020	Master of Music, Music Technology, New York University
2025	Doctor of Philosophy, Music, University of California San Diego

PUBLICATIONS WHILE AT THE UNIVERSITY OF CALIFORNIA SAN DIEGO

Dodero Carrillo, P., Díaz de Cossio Sánchez, T., Hyrkas, J. "Reassembling Russek's Summermood: Revitalizing Mexico's Electroacoustic Legacy Through Modern Performance." Organised Sound, Volume 31, Number 1 (2026).

Hyrkas, J., Dodero Carrillo, P., Díaz de Cossio Sánchez, T. "Preserving Russek's Summermood Using Reality Check and a DeltaLab DL-4 Approximation." PureData Max Conference (2025).

Hyrkas, J., Smyth, T. "Vibrato Suppression by Time-Varying Delay and Spectral Magnitude Demodulation." International Symposium On Musical and Room Acoustics (2025).

Hyrkas, J. "Real-time Implementation of Vibrato Transfer as an Audio Effect." International Computer Music Conference (2025).

Hyrkas, J., Smyth, T. "On Vibrato and Frequency (De)Modulation in Musical Sounds." International Conference on Digital Audio Effects (2024).

Hyrkas, J. "Algorithms for Roughness Control Using Frequency Shifting and Attenuation of Partials in Audio." International Symposium on Computer Music Multidisciplinary Research (2023).

Hyrkas, J., Wilson, A.D., et al. "Spatialized Audio and Hybrid Video Conferencing: Where Should Voices be Positioned for People in the Room and Remote Headset Users?" ACM CHI Conference on Human Factors in Computing Systems (2023).

Hyrkas, J. "WaVAEtable Synthesis." International Symposium on Computer Music Multidisciplinary Research (2021).

ABSTRACT OF THE DISSERTATION

Perceptually-inspired signal processing for modulation in musical sound mixtures

by

Jeremy Hyrkas

Doctor of Philosophy in Music

University of California San Diego, 2025

Professor Tamara Smyth, Chair

Signal processing techniques for audio often operate independently on incoming signals. However, it is increasingly useful to analyze and process signals in the context of other signals in an audio mix. This practice is most commonly employed in manual or automated audio engineering, but there are other applications in which signals might be processed so that a particular auditory or musical effect is achieved upon their combination. This work presents two such applications focusing on modulation control in a mix, outlines signal processing methods for each application and evaluates them both analytically and creatively using examples of music composition and sound design.

Part I explores auditory roughness as a controllable parameter in music. Roughness most

often manifests in mixtures of sounds with sinusoidal partials that are close in frequency. While roughness control is often linked to musical tunings and intervals, the methods presented here control the roughness of sound mixtures by changing the frequency, amplitude and spatial panning of partials in each signal based on the spectral content of all signals in the mix. Roughness control is implemented offline using audio files and in real-time using additive synthesis parameters. Creative applications are presented in the contexts of audio engineering, audio effects, and electronic music composition.

Part II introduces the problem of vibrato matching, in which vibrato patterns from one signal are matched in another signal. Vibrato patterns are highly relevant to the perception of multiple sound sources for both human and machine listeners. As a result, signals with identical vibrato are less likely to be perceived as multiple sound sources. Methods are presented to transfer vibrato patterns from one signal to another, and also to suppress vibrato in a signal so that a new pattern can be applied to it. Vibrato matching offers a novel approach to sound blending, in which signals from multiple sources are used to create a more perceptually fused sound.

Part I and II, while independent in principle, utilize shared signal processing architectures and perceptual features. The solutions to these problems point to potential approaches to the broader processing of signals in sound mixing scenarios.

Introduction

Processing audio signals and mixing them by overlaying them in time is the basis of audio engineering, live sound mixing and electronic music practices dating back to musique concrète. Signal processing techniques for musical audio can have a number of motivations, and in many cases the goal is to produce a particular sonic effect on an input signal without accounting for other signals. However, in the case where multiple signals are mixed to form a cohesive sonic experience, the processing of one signal is often informed by the content of other signals based on a desired effect in the final mix. The most common example is likely that of audio equalization, where specific frequency ranges are boosted or cut in one signal, often so that the signal dominates in specific frequency ranges over other signals in the mix. While audio engineers perform equalization manually, there is a long history of automatic audio mixing [1] that applies audio analysis to signals that are to be combined in a final mix. Audio effects and parameterization for each signal are chosen so that the blend of signals creates a satisfying musical experience.

Automatic mixing parameters are determined by modeling decisions made by audio engineers, which are explicitly or implicitly informed by human auditory perception. There are several other examples of cases when signal processing of one signal is determined by a perceptual goal when mixed with another signal. A common concern in speech processing is the spatial release from masking, where signals that mask each other are more easily heard by a listener by spatially separating them [2]. It has been known since the 1950s that separating voices into different ears makes it easier for a listener to follow either speaker (or both speakers) [3]; this is now done automatically using binaural spatialization in some video teleconferencing systems [4]. Though speech panning in teleconferencing and automatic mixing for music production may seem like unrelated tasks, they share a common analysis and processing pipeline: analyze signals that will be combined and make choices about their processing so that some perceptual goal is achieved upon their combination.

An underexplored area of sound mixing lies in concatenative synthesis. Target-based concatenative synthesis systems analyze a target audio signal and attempt to recreate the sound

using audio sampples from a database that best match audio descriptive features of the input signal [5]. Some concatenative synthesis systems allow multiple sounds from the audio database to overlap in time to better approximate the target signal [6]. This problem is closely related to automatic orchestration, in which a target sound (usually a sound effect or non-instrumental sound) is analyzed and a score is created for a given orchestra to approximate it in a live setting [7, 8, 9].

In each of these problems, audio files from (potentially) disparate sources are mixed in time to approximate a target sound, which is often a single sound source. The initial motivation for the algorithms presented in this dissertation was in service of a concatenative synthesis system that would select audio sources to overlap in time and additionally process them so that their combination was more perceptually blended. One approach is to use sound morphing, in which multiple sounds are analyzed and placed into a perceptual space. A new sound is then resynthesized from some middle ground between the sounds in their perceptual embedding [10, 11]. However, resynthesis can lead to a loss of fidelity or other unwanted artifacts, and sound quality is highly dependent on the analysis and underlying synthesis algorithms. Instead, it may be preferable to use signal processing techniques on each audio source so that the mixture more closely resembles a single source, or at minimum, so that the mixture does not contain unwanted auditory effects such as rapid modulation or obvious timing mismatches.

It is with this application in mind that multiple studies into perceptually-motivated signal processing for sound mixtures were undertaken. This document outlines each study in separate sections. Part I focuses on control of auditory roughness in sound mixtures. Roughness occurs when sinusoidal partials of a sound are close enough in frequency that rapid beating occurs. It is often present in, and considered the primary reason for, musical intervals that are perceived as dissonant. Roughness can easily be introduced when sounds of close (but not exact) fundamental frequencies are mixed. Part I outlines the historical study of auditory roughness, its applications to tuning, and previous methods of roughness control that are based on pitch-shifting. New methods are proposed to control the roughness of sound mixtures without modifying the fundamental

frequency of a signal, offering a new method for controlling roughness as a perceptual auditory effect.

Part II focuses on the musical technique of vibrato, in which a musical note is modulated in pitch and volume by a musician. Vibrato patterns are known to be an important perceptual cue for listeners when determining the presence of multiple sound sources. This section focuses on computational models of vibrato and methods for altering patterns of vibrato in sounds. These methods are presented, evaluated and combined into an algorithm for vibrato matching, in which multiple sound sources are processed so that their vibrato patterns match in time to reduce a listener's perception of multiple sources in a mix.

Each part can be viewed as both an isolated study of a particular problem and as an expansion of the larger problem space of signal processing for sound mixtures. Ultimately, signals are analyzed and processed for an effect that is more important in the context of a mix than in isolation. While the initial motivation stems from the author's interest in concatenative synthesis, the algorithms presented have wider applications in musical composition and sound design. Real musical examples, including a composition by the author, are presented as evaluations. Shared architectures between each problem, particularly the use of filter banks to isolate elements of the sound, connect the algorithms to the history of automatic mixing and beyond. All audio examples from each chapter, as well as additional supporting examples, are available on a supplemental website associated with this dissertation [1].

[1] https://jeremyhyrkas.com/dissertation

Part I

Controlling auditory roughness

Chapter 1

Auditory roughness, tuning and its relation to sound mixtures

1.1 Introduction

Psychoacoustic studies of musical intervals often hypothesize that the perception of consonance and dissonance is related to the number of sinusoidal components of sound (called partials) that are close in frequency in the combined spectra when two or more notes are played simultaneously. The presence of partials relatively close in frequency causes a fluctuation that is sonically perceived as a periodic change in amplitude or as a "buzziness" in the sound. These fluctuations are referred to interchangeably as tonal dissonance, sensory dissonance, or auditory roughness. The difference in frequency that causes maximum roughness is not consistent across the frequency range, but rather is related to the critical bandwidth of the human ear. While musical dissonance is more subjective, sensory dissonance (due to roughness) may be more readily quantified.

The relationship between roughness and consonance makes roughness a salient perceptual feature in music. Various models have been proposed to quantify and computationally model roughness so that it can be automatically analyzed. Sometimes these models are used in creative applications, where sounds are individually modified to minimize the estimated roughness of their mixture. Some examples include the dynamic tuning of electronic instruments, pitch shifting tracks in the context of DJing, and resynthesizing signals of musical chords formed by acoustic instruments.

Models of roughness are quantitatively modeled using time- and frequency-domain analyses of a signal, and are often independent from the source (or sources) that make up the signal. However, most inquiries into roughness as a musical parameter explicitly or implicitly assume that the signal of interest is a mixture of sound sources. Roughness is therefore a musical phenomenon most associated with the combination of several sounds in time. While previous applications of roughness control focus on pitch shifting sources to increase consonance, roughness can be quantified outside the context of tuning and thus may be an interesting perceptual feature to modify without accounting for tuning or musical intervals.

Part I of this manuscript focuses on auditory roughness as it arises in combinations of sounds, and how individual sources might be modified so that roughness is increased or decreased when the sources are mixed. In this chapter, we will review the history of the psychoacoustic study of roughness and its models, and previous applications of these models in music. Chapter 2 introduces new algorithms for modifying roughness in sound mixtures by controlling the frequency, amplitude or spatial panning of sound partials within each source, with implementations and evaluations following in Chapter 3. These methods, when applicable, are compared against previous uses of roughness control using signal re-tuning. The methods presented in Part I offer a new perspective on auditory roughness as a musical result of sound mixtures, with roughness being decoupled from musical intervals and instead treated as a primary musical parameter for composers to manipulate directly.

1.2 Auditory roughness, musical consonance and tuning

1.2.1 Roughness and critical bandwidth

Critical bands are the bandwidths of auditory filters in the human ear, loosely defined as the frequency ranges in which one sinusoidal tone interferes with the perception of another. When two tones fall within the same critical band, either one tone will mask the other or the two tones will be perceived as one fused sound with some degree of amplitude modulation (AM). Tones that are very close in frequency (i.e. < 10 Hz) are heard as a single tone with slow beating and produce a sound generally considered consonant. As the difference in frequency increases, the beating becomes rapid, at which point it is considered *auditory roughness* and is generally considered to be more dissonant than slow beating. As the difference further increases and approaches the critical bandwidth, the listener begins to recognize the sound as separate tones.

Critical bandwidth is smaller in the lower range of audible frequencies and increases at higher frequencies. One proposal for a unit of critical bandwidth is the Bark [12], which determines a critical bandwidth as a nearly constant 100 Hz for center frequencies up to ap-

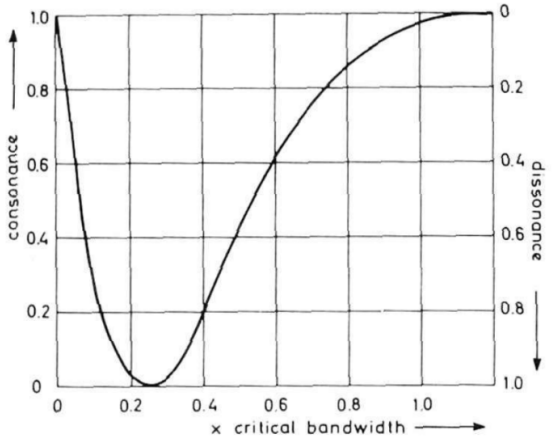

Figure 1.1. Idealized consonance curve of two sinusoids based on their frequency difference modeled as critical bandwidth. Reprinted with permission from Plomp, R., & Levelt, W. J. M. (1965). "Tonal consonance and critical bandwidth." *The Journal of the Acoustical Society of America*, 38(4), 548-560. Copyright 1965, Acoustical Society of America.

proximately the 500 Hz range, after which bandwidth increases somewhat proportionally. The constant bandwidth for lower frequencies has been criticized, and alternative characterizations of auditory filters, such as the Equivalent Rectangular Bandwidth (ERB) [13], propose a decrease in critical bandwidth for frequencies below 500 Hz. The studies that resulted in Barks, ERBs, and other models of critical bandwidth are beyond the scope of this chapter, but computational models of these and other approximations are used throughout Part I.

Theories relating musical consonance and difference in frequency between sound partials date at least as far back as Helmholtz, who credited consonance to the lack of beating partials when harmonic instruments play intervals related by integer ratios [14]. Helmholtz hypothesized the maximum unpleasantness of beating occurs when the frequency difference of partials is in the range of 30–40 Hz. However, this hypothesis was disproved by Plomp and Levelt, who confirmed that maximum roughness is determined by critical bandwidth as opposed to musical intervals or beating at a fixed rate [15]. Plomp and Levelt aggregated data of perceived consonance across frequency ranges to produce their "standard curve," an idealized consonance curve of two tones based on critical bandwidth which is reprinted in Figure 1.1 from the original report [15].

Kameoka and Kuriyagawa extended Plomp and Levelt's experiments and investigated the role sound pressure level (SPL) plays in the perception of consonance [16]. In their experiments, it was confirmed that consonance is affected by SPL, as louder beating pairs are perceived as more dissonant than quieter pairs. Masking also contributes to the perception of consonance, as the roughness of beating pairs decreased as the amplitude of one tone decreased in comparison to the other. No roughness was reported when one tone was completely masked by the other.

1.2.2 Computational models of roughness

Plomp and Levelt's standard curve allows roughness to be calculated as a function of critical bandwidth. Because the curve is idealized and critical bandwidth can be estimated in multiple ways, there are several parameterizations of Plomp and Levelt's curve that have been used in various contexts. In most cases, the dissonance axis of Figure 1.1 is modeled as opposed to the consonance axis. For sounds with more than two partials, roughness is typically defined as the sum of roughness between all pairs of partials.

Sethares [17] models the dissonance curve as

$$PL_s(x) = e^{-b_1 x} - e^{-b_2 x} \tag{1.1}$$

where $b_1 = 3.5$ and $b_2 = 5.75$, and x is critical bandwidth. The values b_1 and b_2 are chosen so that the dissonance peaks at roughly $x = 0.24$, roughly the point of maximum dissonance in Plomp and Levelt's curve. Parncutt [18] alternatively models Plomp and Levelt's curve as

$$PL_p(x) = (e \cdot \frac{x}{a} \cdot e^{\frac{-x}{a}})^2 \tag{1.2}$$

where $a = 0.25$ so that dissonance is maximized at a critical bandwidth of 0.25. The models are largely interchangeable, but they exhibit slightly different shapes, particularly after the point of maximum dissonance as shown in Figure 1.2.

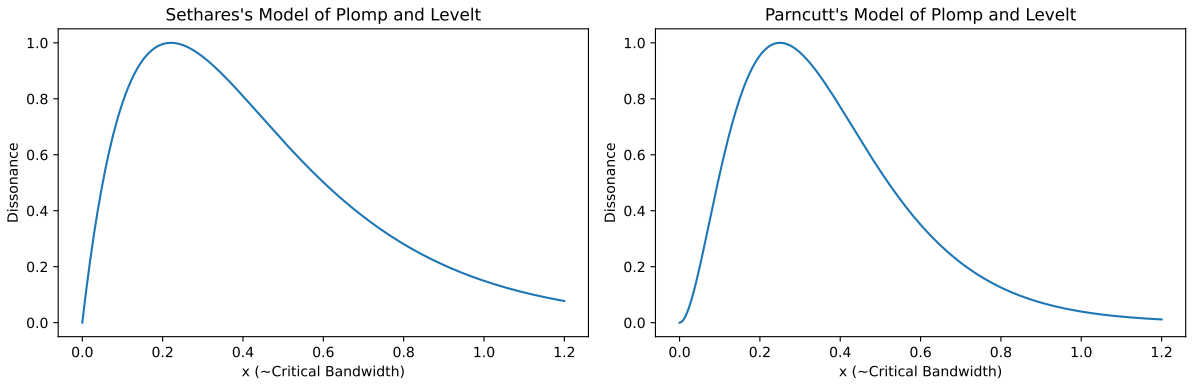

Figure 1.2. Equations (1.1) and (1.2), which model dissonance based on Plomp and Levelt's dissonance curve. The equations model the area of maximum dissonance slightly differently. Dissonance values are unitless and scaled to have a maximum value of 1.

In any model based on Plomp and Levelt's curve, the difference in frequency between two tones must be expressed in critical bandwidth. Parncutt's original model used Hutchinson and Knopoff [19]'s equation for estimating critical bandwidth. Later implementations of Parncutt's model replace this estimation with the Bark scale (defined so that each critical band has a width of one Bark) or the ERB scale, another model of auditory filter bandwidth where filters are generally narrower than critical bands.

Sethares's model includes a critical bandwidth conversion based on the data from Plomp and Levelt's original study. Equation (1.1) was reparameterized for two partials with frequencies f_1 and f_2 having amplitudes a_1 and a_2, respectively, as

$$r_s(f_1, a_1, f_2, a_2) = a_1 a_2 \cdot \left[e^{-b_1 s x} - e^{-b_2 s x}\right] , \tag{1.3}$$

where $b_1 = 3.5$ and $b_2 = 5.75$ as before. In (1.3), $x = f_2 - f_1$ (where $f_1 < f_2$) is the frequency difference between the two partials and s scales the difference using

$$s = \frac{d^*}{s_1 f_1 + s_2} , \tag{1.4}$$

where $d^* = 0.24$ is the position of maximum dissonance and $s_1 = 0.021$ and $s_2 = 19$ are

obtained using a least-squares fit of the data from Plomp and Levelt. Equation (1.3) accounts for partials having different amplitudes, so softer components contribute less to dissonance.

Vassilakis extended (1.3) to account for the role of time-domain fluctuation on roughness, which can be an alternative explanation for the sensation of auditory roughness [20]. Vassilakis models the roughness of partials with frequencies f_1 and f_2 and amplitudes a_1 and a_2 as

$$r_v(f_1, a_1, f_2, a_2) = X^{0.1} 0.5 (Y^{3.11}) Z \ . \tag{1.5}$$

Vassilakis's model largely derives from Sethares's model, with $X = a_1 a_2$ as in (1.3) and $Z = |f_2 - f_1| s$ using (1.3) and (1.4). The new term Y accounts for the depth of amplitude modulation and is defined as

$$Y = \frac{2 \min(a_1, a_2)}{a_1 + a_2} \ , \tag{1.6}$$

so that Y is maximized when $a_1 = a_2$. The exponents 0.1 and 3.11 in (1.5) are chosen to balance the contributions of X, Y and Z.

1.3 Applications of roughness models in computer music

Computational models of roughness have been used in a variety of computer music tasks. The ability to calculate a roughness metric for sounds has been used in computational music theory, integrated into tools for audio engineers and used for automated sound mixing and synthesis systems. Applications to audio engineering and synthesis are particularly relevant to compare with new methods for modifying the roughness of sound mixes that are proposed in Chapters 2 and 3.

1.3.1 Roughness models applied to music theory

Due to Helmholtz's interest in musical consonance, it is no surprise that the earliest applications of computational roughness models were applied to music theory in an attempt to understand the psychological basis of Western music theory. Using these models, the roughness

Table 1.1. The total roughness of three intervals with different harmonic structures as calculated by four variations of roughness models. The intervals are the perfect octave, perfect fifth, and major sixth, all with a base tone at A440 and intervals determined using just intonation. Harmonics 1 through 11 are used, with even harmonics skipped in the lower three rows. Roughness totals across columns are not comparable, and values should be examined comparatively instead of absolutely. The calculations show that all models support Kameoka and Kuriyagawa's findings that the major sixth is a more consonant interval than the perfect fifth for tones with odd harmonics, complicating the conclusions of Hutchinson and Knopoff.

Roughness calculations per model and interval		
Interval and Spectrum	Sethares (1.3)	Parncutt w/ ERB
Octave (all harmonics)	1.67	1.67
Perfect fifth (all harmonics)	4.34	3.97
Major sixth (all harmonics)	4.49	4.05
Octave (odd harmonics)	0.38	0.28
Perfect fifth (odd harmonics)	1.46	1.35
Major sixth (odd harmonics)	0.89	0.97

of combinations of two, three or more tones with different harmonic spectra were calculated and compared to conventional consonance rankings in Western tonal music. Often, a ranking of consonant intervals based on the lowest estimated roughness aligned with traditional Western music theory. However, some contradictions were found.

Kameoka and Kuriyagawa [21] first identified the major sixth as a more consonant interval than the perfect fifth when using tones with only odd harmonics. This finding is a major contradiction to traditional tonal theory, in which the perfect fifth is one of the most consonant intervals, behind only unison and the octave. The authors discovered this discrepancy using an early computational estimate for roughness and confirmed it in listening studies. Table 1.1 shows that this result holds using models of roughness from Sethares and Parncutt. Ironically, the relationship between spectrum and interval is overlooked in later studies that use roughness models to rank the perfect fifth to be the third most consonant interval [19]. This finding suggests that roughness and musical consonance, while related, are not synonymous and could therefore be treated independently.

1.3.2 Roughness models applied to music making

Computational models of auditory roughness were driven by the desire to understand and quantify the consonance of intervals, so it follows that these same models can be used in interval choice. Researchers applied the models outlined in Section 1.2.2 to audio processing and synthesis so that intervals or sound mixes are perceived as more consonant. The problems and technical methods are varied, but they share an underlying utilization of the theories of roughness and the computational models that estimate roughness as a perceptual feature of a signal.

Sethares developed Equation (1.3) over a series of studies that examined the relationship between timbre and scale [17]. His desire to develop consonant scales for any given timbre is likely one of his motivators for modeling (1.3) as a function of frequencies (and particularly as a function of frequency ratio), as opposed to a more explicit model based on critical bandwidth. Often, Sethares would further reparameterize (1.3) as a function of i notes, where each note has fundamental frequency f_0^i, a set of scalars α_k^i that define the relationship between the partials and the fundamental frequency, and a set of amplitudes a_k^i that define the amplitude of each partial. In this posing of the problem, the overall timbral roughness D_F for two notes with potentially different fundamental frequencies and timbres is computed as

$$D_F(f_0^1, \{\alpha^1\}, \{a^1\}, f_0^2, \{\alpha^2\}, \{a^2\}) = \frac{1}{2} \sum_{i=1}^{A^1} \sum_{j=1}^{A^2} r_s(\alpha_i^1 f_0^1, a_i^1, \alpha_j^2 f_0^2, a_j^2) \ , \tag{1.7}$$

where $\alpha_1 = 1$ for all timbre and α_k are integers when the timbre consists of true harmonics. r_s is Sethares's original roughness function for two partials defined in Equation (1.3).

By reparameterizing (1.3) as a function of fundamental frequencies and timbres, Sethares demonstrates multiple uses cases relating to scales, timbre and tuning. When all $\alpha_k^1 = \alpha_k^2$ and $a_k^1 = a_k^2$, D_F can be used to calculate a dissonance curve of frequency ratios of a particular timbre. Local minima of this curve can be used to specify the intervals of a scale for a given timbre [17]. Sethares demonstrates that harmonic timbres lead to local minima at integer ratios,

which correspond to intervals used in just intonation. However, "stretched" harmonic timbres and other inharmonic timbres can lead to very different points of local consonance. Additionally, Sethares demonstrates that (1.7) can be used as part of a constraint optimization problem to find timbres that are consonant with a given scale.

Following this study, Sethares further explored if timbral information and initial fundamental frequencies of musical notes could be used to retune the notes automatically for maximum consonance. Again using Equation (1.7) as the underlying model, Sethares developed the Adaptive Tuning algorithm [22], which adjusts the fundamental frequencies of notes based on their spectra to minimize the expected roughness of the sound. Adaptive Tuning calculates the partial derivative of (1.7) with respect to each fundamental frequency and performs a gradient descent search where each f_0 iteratively steps in the direction of the gradient until convergence. Adaptive Tuning can be approximated in real-time on spectra that are known ahead of time [23], with some modifications necessary to reduce CPU load. A more exhaustive implementation computes the algorithm on notes with different spectra in non-real-time [24].

Some elements of Sethares's real-time approximation were likely due to CPU limitations at the time of its development, and the use of outboard processors and synthesizers to generate sound. Modern CPUs are more equipped to perform the algorithm in real-time, particularly when the sounds are being generated additively and thus the sinusoidal parameters are already known. Porres et al. implemented such an approach and applied it to theremin performance, where closeness to the pitch antenna is mapped to valleys in the dissonance curve instead of a continuous fundamental frequency [25]. While this approach removes the expressive vibrato of a theremin, it allows two players to more easily play in tune with each other, particularly when inharmonic spectra are chosen.

DJs frequently mix two or more tracks together and overlay them in time. It is common to pitch shift one track so that the mix of tracks does not cause dissonance introduced by key clashing or other undesirable elements. Gebhardt et al. used roughness as one of the primary measures for determining the best pitch shift of a track meant to be mixed with another in the

context of DJing [26, 27]. Equation (1.2) is used to calculate roughness, with the ERB used as an estimate of critical bandwidth. Roughness is calculated on a sinusoidal models of the tracks, where pairs of partials (one from each track) that overlap in time potentially contribute to the roughness of the mix. The results of various listening tests indicate that roughness is the most important factor in determining the best pitch shift. Incorporating models of harmony caused no statistically significant benefit, and in some cases showed a degradation from determining pitch shifts entirely using roughness minimization [27]. This finding underscores the importance of roughness as a primary component of perceived dissonance.

Roughness annotation can be particularly useful in the studio, as a live roughness meter could be used by a sound engineer to make mixing decisions. Vassilakis first explored this approach in offline mixes by providing software to create roughness annotations on user-provided audio signals [28]. Two real-time estimation algorithms have been developed for the so-called "patcher" programming environments, one using (1.2) in Max/MSP [29] and another using (1.5) in Pure Data [30]. Both algorithms rely on real-time peak-picking in the frequency domain to produce the top N detected partials at each analysis frame. As a result, partials with near-equal frequency cannot be easily resolved and instead appear as a single beating partial.

Although not explored by any of these authors, a more accurate method for live roughness monitoring could be achieved by analyzing each source before they are mixed. Just prior to mixing, a roughness metric could be computed over the frequencies and amplitudes of partials from each source, without the need to disambiguate the frequencies and amplitudes of beating partials after mixing. The obvious downside to this approach is the need for sinusoidal analysis on multiple tracks, increasing computational complexity with each track analyzed. Additionally, in a streaming audio system, analyses from each track would need to be shared with a global roughness calculation process, which could lead to complex implementations using common audio interfaces such as the Virtual Studio Technology (VST) or Apple's Audio Unit (AU). Nevertheless, a similar approach will be explored in the following chapters, but with the goal of controlling roughness (as opposed to measuring it) by modifying each sound source.

1.4 Conclusion

Auditory roughness caused by the presence of sound partials in the same critical band has been studied for decades as an important perceptual feature in music. Although roughness can exist in a single sound source, it is most often understood as an effect caused by the combination of multiple sound sources. The sources may be simple sinusoidal tones, more complex harmonic or inharmonic (but monophonic) "notes," or as complex as polyphonic sound mixtures. As a result, roughness is somewhat unique among perceptual sound descriptors (i.e., loudness, pitch, timbre) in that it is most sensibly analyzed in the context of sound mixtures.

Previous work has largely focused on the effect of roughness as a function of tuning and fundamental frequency, or as general annotation without a mechanism for control. When roughness has been controlled, usually by reducing roughness, it has been done by changing the pitch of one or more sources, leaving the overall timbre unmodified. Developing methods for controlling roughness by modifying timbre instead of tuning is an underexplored area of research. Methods in this vein are proposed and evaluated in Chapters 2 and 3.

Chapter 2

Algorithms to control roughness by changing the parameters of sound partials

2.1 Introduction

Signal processing solutions for roughness minimization have previously been fundamentally tied to tunings and intervals. Such approaches follow the history of the development of theories of sensory dissonance as a means of explaining consonance in Western music theory. Recall that in the Adaptive Tuning algorithm [22] and automated track detuning algorithms for DJs [26], all partials of a sound have their frequencies changed as a result of a change in fundamental frequency or pitch shift. However, if roughness as a perceptual element of sound can be determined as a function of the proximity in critical bandwidth of sound partials, the control of a sound's roughness need not be tied to tuning or intervals at all.

An unexplored approach is the selective adjustment of partials to reduce or increase roughness in some portions of a signal while leaving the majority of the signal intact. Roughness can be reduced (or increased) in a signal by changing the frequency of nearby partials to change their beating pattern. It is possible to destroy the sensation of pitch in a pitched sound if all partials are detuned in non-uniform and extreme amounts, but if most harmonic relationships are preserved, a listener would likely still hear the same pitch but have a different sensation of roughness. Alternatively, the amplitude of neighboring partials could be adjusted to change their contribution to the overall roughness given that both the relative and absolute amplitudes of pairs of partials play into the perception of roughness.

In this chapter, three algorithms for controlling the roughness of a sound based on the adjustment of sound partials are presented. Two algorithms take a greedy approach to minimizing or maximizing the roughness of a signal by changing either the frequency or amplitude of partials that meet certain criteria. The third algorithm attempts to control roughness binaurally by changing the spatial panning of partials to minimize (or maximize) roughness in each ear. After the theory and general flow of the algorithms are presented, implementations of the algorithms on audio signals and additive synthesis parameters are detailed in Chapter 3.

2.2 Roughness control algorithms: the shared model

Changing the roughness caused by two partials can be accomplished by changing either the frequencies or the amplitudes of the partials. The methods presented here are developed using Sethares's original roughness model of two sound partials,

$$r_s(f_1, a_1, f_2, a_2) = a_1 a_2 \cdot [e^{-b_1 sx} - e^{-b_2 sx}] \,, \tag{2.1}$$

with

$$s = \frac{d^*}{s_1 f_1 + s_2} \,, \tag{2.2}$$

redefined here for convenience but previously presented as Equations (1.3) and (1.4). However, other roughness models from Chapter 1 can be substituted. The proposed algorithms control roughness control by changing frequency, amplitude or spatial panning of partials to modify the overall roughness as determined by Equations (2.1)–(2.2).

2.3 Changing roughness by partial tuning

Here we define *frequency bashing*, a greedy algorithm for controlling roughness by changing the frequency of partials that lie within a critical band. The algorithm takes as input a list of pairs of partials and their contribution to the overall roughness of the signal. Partials are assumed to overlap in time and remain relatively centered around a constant frequency (i.e. vibrato is permitted but glissando is not accounted for). Pairs of partials are iteratively changed to adjust roughness, with pairs that cause the most roughness being adjusted first. A partial may contribute to roughness in more than one pair of nearby partials, so as the algorithm iterates, partials that have already been adjusted are skipped.

For a pair of partials with frequencies f_1 and f_2 and amplitudes a_1 and a_2, respectively, their roughness can be determined using Equation (2.1). Trivially, we could reduce the roughness of the sound by picking one of the partials and moving it along the dissonance curve defined

by (2.1) to the point of minimum roughness; similarly, the partial could be moved to the point of maximum roughness to increase the dissonance of the sound. However, it is clear from the shape of (2.1) (see Figure 1.2) that the point of minimum dissonance occurs when $f_1 = f_2$, and the point of maximum dissonance occurs when $|f_1 - f_2|s$ is equal to 0.24, corresponding to the point of maximum dissonance in the Plomp and Levelt curve.

The goal of the frequency bashing algorithm is to control roughness while maintaining as much of the original timbre as possible, so it is not ideal to set partials to the same frequency because it would create gaps and peaks in the spectrum. To address this problem, the frequency bashing algorithm takes as input two parameters, B_L and B_H, which are fractional Bark values. These parameters serve two purposes: first, only partial pairs that are within the fractional Bark range from each other will be considered when frequency bashing (i.e. partials where f_1 and f_2 are less than B_L or more than B_H Barks apart cause minimal beating and should not be adjusted because of their contribution to roughness). Second, when one partial has its frequency adjusted based on roughness caused by its proximity to another partial, B_L and B_H restrict the range of potential new frequency values. Based on the shape of Plomp and Levelt's curve (Figure 1.1) on which (2.1) is based, the difference in Barks is typically set between 0.05 and 0.4 in the experiments presented here, but the difference is exposed as a user parameter in the implementations in Chapter 3.

Frequency bashing for minimum roughness sets the frequency of the quieter partial to

$$f_2^* = \underset{f_{\min} \le f^* \le f_{\max}}{\arg\min} \; r_s(f_1, a_1, f^*, a_2) \; , \tag{2.3}$$

with the constraints

$$f_{\min}, f_{\max} = Hz_B(B_{Hz}(f_1) + B_L), Hz_B(B_{Hz}(f_1) + B_H) \; , \tag{2.4}$$

where Hz_B and B_{Hz} convert from Bark to Hz and vice versa [31]. In (2.3) and (2.4), f_1 is the

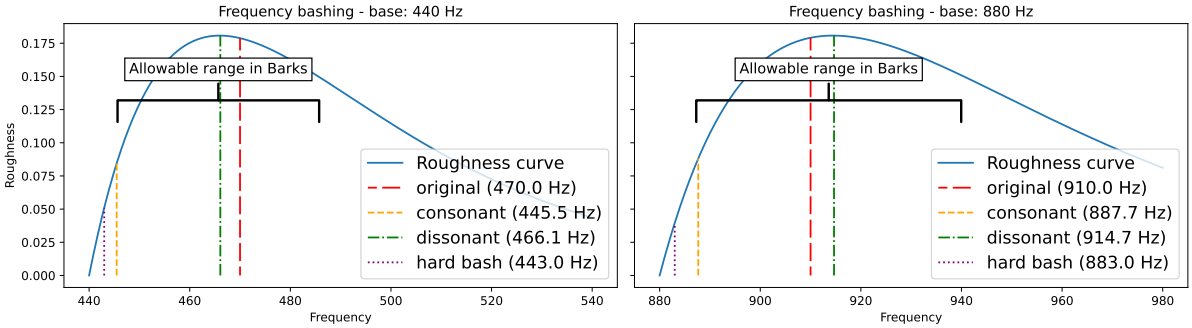

Figure 2.1. Frequency bashing pairs of partials with frequencies (440 Hz, 470 Hz) on the left and (880 Hz, 910 Hz) on the right. The partials with higher frequencies are adjusted. The original frequency of the adjusted partial is depicted in the long-dashed red line, the most consonant in the allowable range of movement in Barks is shown in the short-dashed yellow line and the most dissonant in the dash-dotted green line. Hard-bashing to a difference of 3 Hz is shown in the dotted purple line. Hard-bashing ignores the roughness curve and allowable range.

louder partial whose frequency remains constant, while frequency f_2 is the quieter partial whose frequency will be bashed to a new value. When increasing roughness instead of decreasing, the argmin operation in (2.3) is replaced by argmax. Equation (2.4) assumes $f_1 < f_2$; when $f_1 > f_2$ the Bark range is defined below f_1 instead of above.

Figure 2.1 shows potential adjustments for two sinusoid pairs, one with frequencies 440 Hz and 470 Hz and the other pair at 880 and 910 Hz. Various vertical lines show the original frequency of the higher partial and frequencies it could be moved to for maximum or minimum roughness based on the allowable movement in Barks. Note that even though both examples in the figure have an identical difference in Hz, the horizontal scale of the roughness curve and the solutions for maximum consonance and dissonance are different due to the dissonance curve's dependence on critical bandwidth.

Another option is *hard-bashing*, where the quieter partial is adjusted to be a specified difference in Hz from the unchanged partial. An advantage of hard-bashing is that when multiple partials are adjusted within a sound, they will have identical frequency difference from their neighboring partial, creating a slow-beating tremolo effect, but only in certain frequency ranges of the signal. In Figure 2.1, hard-bashing partials to have a difference of 3 Hz maintains the

equal difference in Hz between examples after bashing and both new adjustments result in lower roughness. However, adjusted partials now fall on different positions along the roughness curve due to their different critical bandwidths. While less informed by roughness models, hard-bashing is computationally cheap and may be aesthetically preferable depending on the musical application.

Frequency bashing over more than two partials is defined in pseudocode in Algorithm 1. The algorithm accepts a list of partial pairs as well as their contributions to the overall roughness of the sound, and greedily adjusts pairs of partials in order of descending estimated roughness. When processing a pair of partials, the quieter partial is bashed while the louder partial is unchanged. Both partials are marked as processed, and later pairs containing either partial will be skipped. This optimization is required to avoid adjusting the same partial multiple times, which could negate previous changes by moving a partial closer to its original frequency.

Algorithm 1: Frequency bashing

Data: $x = \{(p_i, p_j, r_{ij})\}, 0 \leq B_L < B_H < 1$
$x_s \leftarrow$ sort(x) by r_{ij} **desc** `/* Sort partial pairs by descending roughness */`
$P \leftarrow \{\}$ `/* Processed partials */`
$k \leftarrow 1$
while $k \leq |x_s|$ **do**
  $p_i, p_j, r_{ij} \leftarrow x_s[k]$
  $k \leftarrow k + 1$
  if $p_i \in P \vee p_j \in P$ **then**
    continue
  end
  $f_i, a_i \leftarrow$ freq(p_i), amp(p_i)
  $f_j, a_j \leftarrow$ freq(p_j), amp(p_j)
  if $a_i > a_j$ **then**
    freq(p_j) $\leftarrow$ freq_bash(f_i, B_L, B_H) `/* Eq. (2.3) */`
  else
    freq(p_i) $\leftarrow$ freq_bash(f_j, B_L, B_H) `/* Eq. (2.3) */`
  end
  $P \leftarrow P \cup \{p_i, p_j\}$
end

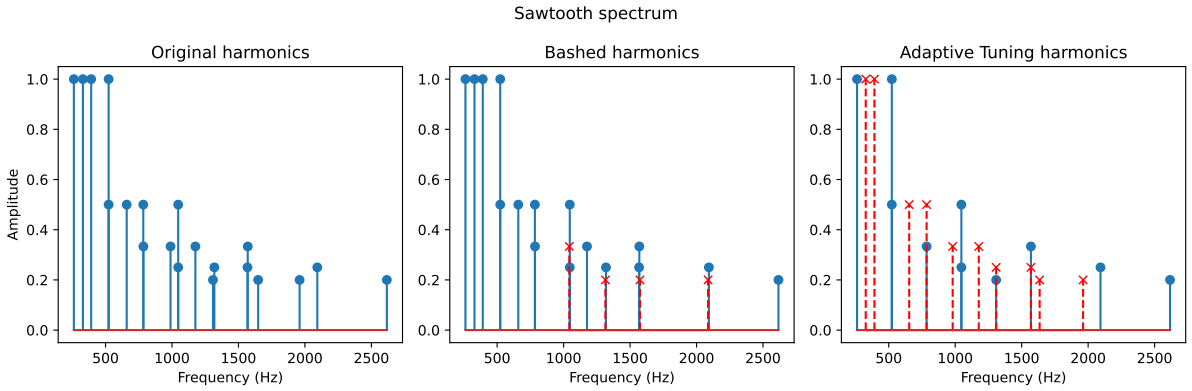

Figure 2.2. The spectrum of a major chord using sawtooth waves. The original spectra (left) is modified using either frequency bashing (center) or Adaptive Tuning (right). Red dashed lines indicate a partial that has been changed. Adaptive Tuning changes the fundamental frequencies of two notes to reduce roughness, changing all partials of each note. Frequency bashing adjusts four partials while leaving the majority of the original spectrum intact.

The algorithm does not guarantee optimality in its reduction of roughness like the Adaptive Tuning algorithm, but it is guaranteed to reduce roughness while being simple enough to implement in real-time. The cases where frequency bashing will struggle to find a minimally rough configuration of partials will be those where many partials (four or greater) lie within a fraction of the critical bandwidth. Such a cluster of tones is by its nature dissonant, and it is not necessarily clear what the correct solution to roughness reduction would entail, so this limitation of the algorithm is acceptable.

Figures 2.2 and 2.3 show the difference between the use of frequency bashing and Adaptive Tuning. In Figure 2.2, the algorithms process an equal temperament major triad (plus octave) of sawtooth wave voices. Frequency bashing adjusts the frequencies of four partials, reducing the beating pattern of the mixture, while Adaptive Tuning changes the fundamental frequency of the major third and fifth notes to just intonated intervals. Red dashed lines indicate partials that have different frequencies than in the original chord.

A similar comparison is shown in Figure 2.3, but now using notes with a stretched timbre. The spectrum of each note consists of partials at 1x, 2.1x, 3.24x, 4.41x, and 5.6x the fundamental frequency. Sethares previously used this example to demonstrate how Adaptive Tuning can

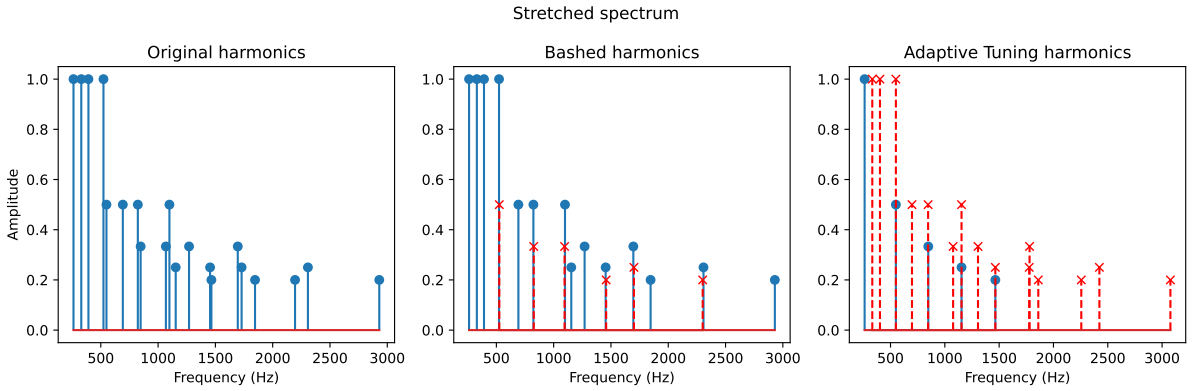

Figure 2.3. The spectrum of a major chord using stretched inharmonic notes. The original spectra (left) is modified using either frequency bashing (center) or Adaptive Tuning (right). Adaptive Tuning changes the fundamental frequencies of three notes to reduce roughness, including the octave. Frequency bashing adjusts six partials. Both adjusted signals have reduced roughness but signally different timbres.

create local minima in dissonance curves that can be used to create tuning systems for inharmonic spectra. In comparison, frequency bashing changes partials individually as opposed to at the note level, and as before, fewer partials have their frequencies changed. Both algorithms reduce roughness compared to the original sound mixture, but the Adaptive Tuning example sounds substantially different. This demonstrates a difference in philosophy, as the goal of frequency bashing is to modify roughness while maintaining as much of the original signal as possible.

2.4 Changing roughness by amplitude adjustment

A different approach to roughness control is to change the amplitude of partials, as the absolute and relative loudness of partials within a critical band contribute to roughness. A simple technique would be fully attenuate the quieter partial in a pair that contributes roughness to the sound. However, lowering the amplitude of a partial will reduce the overall power of the signal, which may change the listener's perception in unintended ways. Instead, the quieter partial should have its amplitude decreased while the louder partial's amplitude is increased to maintain the original signal power. This seesaw effect of amplitude adjustments may remind the reader of the children's arcade game "whack-a-mole," and is therefore named *amplitude whacking*.

Whacking can be performed on a scale between 0.0 (no change) and 1.0 (maximum amplitude change). We will designate this as a user parameter p (or percentage change) in the algorithm. When maximally changing amplitudes, we could consider entirely attenuating the quieter partial and transferring its power to the louder partial. However, this approach is flawed for two reasons: First, adding a substantial amount of power to a partial may drastically change the timbre of the sound. Second, if the partials are close in critical bandwidth, the louder partial will mask the quieter partial well before all power has been transferred, resulting in some range of the parameter p that has no audible effect. Therefore, the algorithm should maximally adjust amplitudes so that the quieter partial is fully masked by the louder partial at $p = 1$.

A simple model [32] to estimate the masking curve of a partial with frequency f and SPL in decibels dB using Barks can be computed using

$$\text{mask}(x|f, dB) = \left\{ \begin{array}{ll} dB - 10 - 27\left[B_{Hz}(f) - B_{Hz}(x)\right], & \text{if } x < f \\ dB - 10 - 15\left[B_{Hz}(x) - B_{Hz}(f)\right], & \text{if } x \geq f \end{array} \right\}, \tag{2.5}$$

where B_{Hz} converts from Hz to Bark as above [31]. Figure 2.4 shows the shape of Equation (2.5). This masking model is greatly simplified compared to actual auditory masking curves, but has been used effectively in real-time applications where minimizing CPU usage is critical [33].

As before, amplitude whacking is first defined over two partials. We consider two sinusoidal partials with frequencies f_1 and f_2 and amplitudes a_1 and a_2, respectively. For this discussion, we will assume that $a_1 > a_2$, and calculate the SPL values $dB_1 = 20\log_{10}(a_1)$ and $dB_2 = 20\log_{10}(a_2)$. The aim is to maintain the power of the signal while adjusting both partials so that the difference in dB falls between the original difference in dB and the theoretical masking threshold determined by (2.5). Specifically, we will define the desired difference in dB as

$$\Delta_{dB} = (dB_1 - dB_2) + p \cdot (\text{mask}(f_2|f_1, dB_1) - (dB_1 - dB_2)), \tag{2.6}$$

so that when $p = 1$, Δ_{dB} is precisely the masking limit defined by the masking curve.

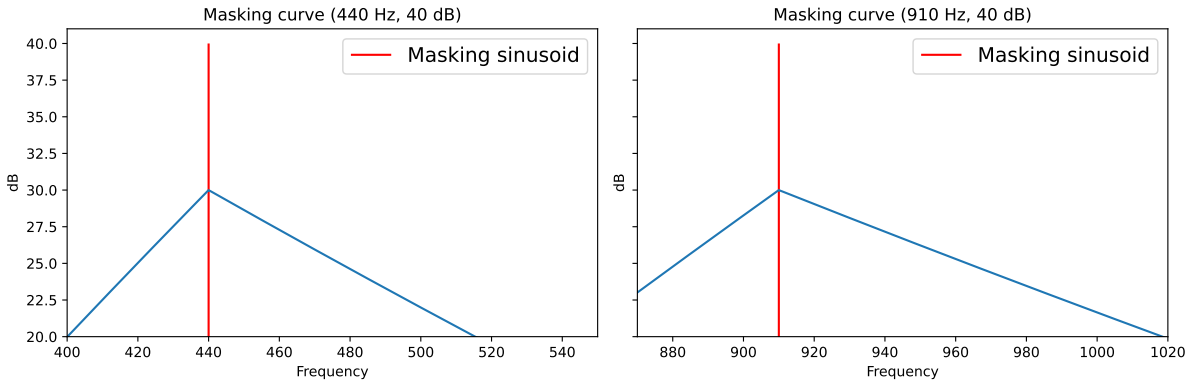

Figure 2.4. The estimated masking curve of a sinusoid with frequency 440 Hz is shown on the left and 910 Hz on the right, each with an SPL of 40 dB. Both masking curves show a characteristic shallower slope for frequencies greater than the sinusoid, demonstrating more masking of higher frequencies than lower frequencies. The x-axes in both figures cover the same range in Hz. The shallower slopes on the right show the effect of critical bandwidth on masking, with higher frequencies masking a larger range than lower frequencies.

Given these quantities, the constraints of power preservation and desired dB difference are defined as

$$20\log_{10}(a_1^*) - 20\log_{10}(a_2^*) = \Delta_{dB}\ , (a_1)^2 + (a_2)^2 = (a_1^*)^2 + (a_2^*)^2 \tag{2.7}$$

respectively, where a_1^* and a_2^* are the new amplitudes of the partials. Solving this system of equations algebraically leads to the following solution for amplitude whacking:

$$a_2^* = \sqrt{\frac{(a_1)^2 + (a_2)^2}{1 + 10^{\frac{\Delta_{dB}}{10}}}},\ a_1^* = \sqrt{(a_1)^2 + (a_2)^2 - (a_2^*)^2}\ . \tag{2.8}$$

As before, we can now define the amplitude whacking algorithm over a collection of partial pairs and their contribution to the overall roughness of the signal. The algorithm proceeds identically to frequency bashing, with pairs of partials that contribute the most roughness processed first and partials that have already been adjusted skipped in later iterations. Pseudocode of the algorithm is shown in Algorithm 2.

Figure 2.5 shows potential adjustments for amplitudes of partial pairs to reduce the

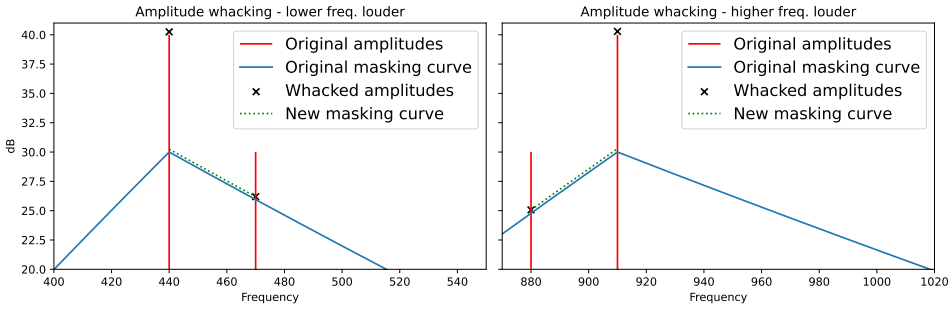

Figure 2.5. Amplitude whacking pairs of partials with frequencies (440 Hz, 470 Hz) on the left and (880 Hz, 910 Hz) on the right. Original amplitudes are shown as red solid lines, with whacked amplitudes depicted as black X's. Despite each pair being equidistant in Hz, the example on the right requires more attenuation of the quieter partial due to the asymmetry of the masking curve.

Algorithm 2: Amplitude whacking

Data: $x = \{(p_i, p_j, r_{ij})\}, 0 \leq B_L < B_H < 1, 0 \leq p \leq 1$
$x_s \leftarrow$ sort(x) by r_{ij} **desc** `/* Sort partial pairs by descending roughness */`
$P \leftarrow \{\}$ `/* Processed partials */`
$k \leftarrow 1$
while $k \leq |x_s|$ **do**
  $p_i, p_j, r_{ij} \leftarrow x_s[k]$
  $k \leftarrow k + 1$
  if $p_i \in P \vee p_j \in P$ **then**
    continue
  end
  $f_i, a_i \leftarrow$ freq(p_i), amp(p_i)
  $f_j, a_j \leftarrow$ freq(p_j), amp(p_j)
  $dB_i, dB_j \leftarrow 20\log_{10}(a_i),\ 20\log_{10}(a_j)$
  if $a_i > a_j$ **then**
    $db_C \leftarrow (dB_i - dB_j)$ `/* Current difference in dB */`
    $dB_M \leftarrow mask(f_j|f_i, dB_i)$ `/* Masking difference in dB (Eq. (2.5)) */`
    $\Delta_{dB} \leftarrow dB_C + p \cdot (dB_M - dB_C)$ `/* Desired difference in dB */`
    amp(p_j), amp(p_i) $\leftarrow$ amp_whack(a_j, a_i, Δ_{dB}) `/* Eq. (2.8) */`
  else
    Same as above, swapping the order of partials and parameters
  end
  $P \leftarrow P \cup \{p_i, p_j\}$
end

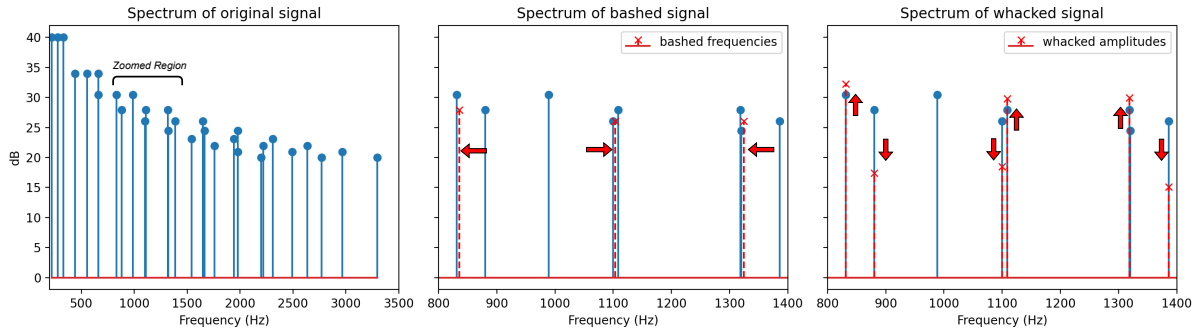

Figure 2.6. Left: the spectrum of sawtooth waveforms forming a major triad in equal temperament tuning. **Center**: a zoomed region of the spectrum after frequency bashing partials. Three partials have been moved. **Right**: the same zoomed region of the spectrum after amplitude whacking. Three pairs of nearby partials (six partials total) have had their amplitudes adjusted.

roughness of a sound. Two properties of note are demonstrated in the figure. First, even though the power of the signal is maintained by increasing the amplitude of the louder partial, the change in dB for the louder partial is far less than the change for the quieter partial, which suggests that in general the algorithm will not greatly boost individual partials in a way that draws attention. Second, due to the asymmetry in the masking curve, we can observe that while both examples have partials of the same amplitude, the quieter partial must be attenuated more when it is lower in frequency than the louder partial to be masked.

Figure 2.6 depicts the frequency bashing and amplitude bashing algorithms applied to an equal tempered major triad of sawtooth waves with 10 partials. The original spectrum is shown on the left side of the figure, alongside the adjusted partials in each algorithm in a zoomed region of the spectrum. Three pairs of partials are found to be nearby in a critical band and contribute to roughness. Although the algorithms only adjusts a handful of partials, there is a noticeable difference in beating between the original and processed examples while the characteristic of the sound is largely intact. As previously seen in Figures 2.2 and 2.3, fewer partials are removed than solutions such as Adaptive Tuning. Subjectively, all solutions reduce are successful in removing roughness but create different sonic experiences. Preference of one algorithm to the others is left to composers and sound designers depending on their artistic intent.

2.5 Changing binaural roughness using spatial panning

Models and applications of roughness in music, including the algorithms outlined in Sections 2.3 and 2.4, typically treat roughness as a function of partials in a single channel. Roughness as a binaural perceptual effect is an underexplored area of research. Assuming a listener is in a situation where the left and right ear receive different signals, a simple model of binaural roughness could be calculated as the sum of roughness experienced in each ear. It is unclear whether the overall roughness experienced by a listener can be accurately modeled as a sum of independent calculations for each ear or whether binaural roughness is better modeled by a nonlinear metric or using some shared information between signals arriving to each ear.

Few previous methods model the roughness of sound projected through multichannel systems. In perhaps the only existing model, Hansen provides a thorough perceptual model of sine tones in space that accounts for roughness curves, power loss, masking, phase cancellation, and the effects of equal loudness curves [34]. This approach, while accounting for many acoustic and psychoacoustic properties, is complex and prohibitively expensive for real-time control. The model of binaural roughness used in this chapter is simpler and cheaper to compute.

Here, we propose an algorithm to control roughness by changing the roughness in each ear using spatial panning. Panning will be considered by placing partials equidistantly from a centered listener along a circle. Changes in distance and elevation are not considered. Using Equations (2.1) and (2.2), an estimate of roughness can be calculated for each ear. The difference between this binaural calculation and the monaural original is the amplitude of each partial, which needs to be scaled for each ear. A simple approach is to scale the amplitude of the partial by power-preserving panning equations. If a partial with amplitude a_1 is placed at azimuth d_1, where $d_1 = 0^\circ$ is directly in front of the listener and $d_1 = 90^\circ$ is directly to the right, the amplitude received by the left ear can be approximated as:

$$a_1^L = a_1 \cdot \left| \cos\left((\frac{d}{2} + 45) \cdot \frac{\pi}{180} \right) \right|, \tag{2.9}$$

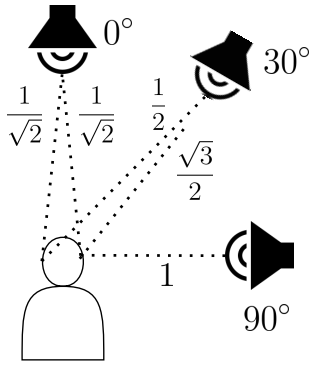

Figure 2.7. Example per-ear gain multipliers for the amplitude of a signal played from different locations around a listener's head.

with the right ear amplitude a_1^R calculated using (2.9) with the cosine operation replaced by sine. The sine/cosine pair is typically used in stereo panning, but here is extended into two dimensions.

Figure 2.7 shows the amplitude scaling factor of a signal placed somewhere along a circle surrounding the listener at ear height. The roughness of each ear can then be computed using (2.1) and (2.2) with a_i^L amplitudes used for the left ear and a_i^R for the right. Using Equation (2.9), a sound source is modeled as equally loud in each ear when at $0°$ or $180°$ azimuth. Sound sources at $90°$ or $-90°$ have a gain of 1 in one ear and a gain of 0 in the other.

Algorithm 3 outlines pseudocode for a simple optimization method for binaural roughness reduction. The algorithm pans each partial independently, with each partial initially placed around a circle surrounding a listener. The initial roughness is calculated using Equation (2.1) on each ear using the modified amplitudes a_i^L for left ear roughness and a_i^R for right ear roughness. The algorithm then proceeds for m_{it} steps (the maximum number of optimization steps). At each step, two partials are randomly selected and a new roughness measure is calculated based on swapping the locations of partials. If the new binaural roughness calculation is lower, the azimuth panning of the partials are switched; if not, the partials remain at their previous location.

This optimization algorithm has a handful of drawbacks. It is somewhat computationally inefficient and does not guarantee optimality. The algorithm pans each partial separately, which is perhaps only appropriate for synthetic examples, as breaking apart the partials of a real harmonic

Algorithm 3: Roughness-based panning

Data: $x = \{p_i\}, N = |x|, m_{it}$
$d_i \leftarrow \frac{360^\circ}{N} \cdot [0, 1, ..., N-1]$ /* Initial panning around a circle */
$\theta_i \leftarrow (\frac{d}{2} + 45) \cdot \frac{\pi}{180}$
$f_i \leftarrow$freq(p_i)
$a_i \leftarrow$ amp(p_i)
$a_i^L \leftarrow |\cos(\theta_i)| \cdot a_i$ /* Left-ear amplitudes */
$a_i^R \leftarrow |\sin(\theta_i)| \cdot a_i$ /* Right-ear amplitudes */
$r_c \leftarrow \sum_{j=1}^{N-1} \sum_{k=j+1}^{N} r_s(f_j, a_j^L, f_k, a_k^L) + r_s(f_j, a_j^R, f_k, a_k^R)$ /* Current roughness */
$it \leftarrow 1$
while $it \leq m_{it}$ **do**
  $j \leftarrow$ random($\{0, ..., N-1\}$) /* Random index */
  $k \leftarrow$ random($\{0, ..., N-1\} \setminus \{j\}$) /* Different random index */
  $\theta_j^* \leftarrow \theta_k, \theta_k^* \leftarrow \theta_j$ /* Consider swapping partials j and k */
  $r_p^* = ...$ /* Potential new roughness */
  if $r_p^* < r_c$ **then**
    $r_c = r_p^*$
    $\theta_j \leftarrow \theta_k, \theta_k \leftarrow \theta_j$ /* Keep the swap */
  end
  $it \leftarrow it + 1$
end

sound source could be perceptually distracting to a listener. However, as will be discussed in Chapter 3, the algorithm has musical applications despite these limitations.

Regarding the accuracy of the proposed binaural measure, it should be noted that in a real binaural or multichannel setup, a signal with a 90° pan would still reach the left ear of a listener with a gain higher than 0. However, 90° still represents the circular angle at which the left ear would receive the signal with the lowest amplitude. Here we operate on the assumption that this simple equation is sufficient to model the relative per-ear gain of a circular pan, and that roughness curves calculated using more complex gain estimates (such as a head-related transfer function) would find similar local minima. However, this assumption is currently unverified.

2.6 Conclusion

This chapter outlines three algorithms for changing the roughness of a signal by modifying the frequency, amplitude or spatial panning of sound partials. Previous algorithms for roughness control operate by pitch shifting input sounds, treating roughness as necessarily tied to tuning. The algorithms here treat roughness as a perceptual parameter that can be modified independently of tuning. All algorithms presented here are theoretical. To evaluate their efficacy in music, it is necessary to extend them into several musical applications. In the process, algorithmic modifications are required to overcome analytical difficulties or to operate in real-time. Concrete implementations are described and evaluated in Chapter 3.

Chapter 2 contains material from the research paper *Algorithms for Roughness Control Using Frequency Shifting and Attenuation of Partials in Audio*, published in the Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR 2023). The dissertation author was the primary investigator and author of this work.

Chapter 3

Implementations and evaluations of roughness control algorithms

3.1 Introduction

The frequency bashing, amplitude whacking and roughness-based panning algorithms defined in Chapter 2 can be performed offline on audio signals or in real-time in the context of additive synthesis. Adapting the algorithms to each approach requires further modifications, tweaks, and optimizations. In this chapter, implementations of the algorithms are defined and evaluated using spectral analysis and demonstrations in both short musical examples and an original multichannel composition.

3.2 Offline implementations on stored audio signals

Auditory roughness occurs most often when partials fall very close to each other (i.e., on the order of 20-40 Hz difference). Finding separate partials with this resolution in one audio signal is difficult using only FFT-based methods [29], although there has been some success in reverse engineering the frequency and amplitude of nearby partials based on amplitude modulation patterns [35]. Roughness control algorithms aim to isolate partials from a signal to modify them, and isolating one partial in a digital filter without affecting a nearby partial in the same signal is very difficult, or in many cases functionally impossible. It is therefore unlikely the algorithms presented in Chapter 2 can be applied to a signal that already contains roughness. However, we can leverage the observation that roughness models are most often applied to mixtures of sounds [23, 26, 35] and follow this approach by applying these algorithms to a collection of signals that are to be combined in a mix. Partials from different signals that overlap in time will be adjusted using the algorithms described in Chapter 2, without making any attempt to modify the roughness that already exists in any individual signal.

Given a collection of audio files, each file is analyzed using sinusoidal modeling [36] to find the top N sinusoids of each frame of audio. Sinusoids are then combined across frames so that each signal is associated with a collection of sinusoidal partials with amplitudes and frequencies that are tracked across time. N can be a small number (i.e., on the order of 10)

because the sinusoidal tracks will not be used for resynthesis, but are instead used to identify the most prominent partials at a given time. Partials from each signal that are found to overlap in time and fall within a critical band are collected as potential candidates for processing. These pairs of partials are then processed using the algorithms and equations defined in Chapter 2. Algorithm 4 depicts the pseudocode for this analysis.

Algorithm 4: Evaluate roughness of partial pairs in two signals

Data: $s_1, s_2, N > 0, B_\Delta > 0$
Result: $y = \{(p_i, p_j, r_{ij})\}$
$y \leftarrow \{\}$
$P_1 \leftarrow$ sinusoidal_analysis(s_1, N) `/* Max of` N `partials per frame */`
$P_2 \leftarrow$ sinusoidal_analysis(s_2, N)
$i, j \leftarrow 1$
while $i \leq |P_1|$ **do**
  $p_i \leftarrow P_1[i]$
  $f_i, a_i \leftarrow$ freq(p_i), amp(p_i)
  while $j \leq |P_2|$ **do**
    $p_j \leftarrow P_1[j]$
    $f_j, a_j \leftarrow$ freq(p_j), amp(p_j)
    $c_{ij} \leftarrow$ overlap_in_time(p_i, p_j) `/* Partials must overlap in time */`
    $c_{ij} \leftarrow c_{ij} \wedge |B_{Hz}(f_i) - B_{Hz}(f_j)| \leq B_\Delta$ `/* ...and be close in Barks */`
    if c_{ij} **then**
      $r_i j \leftarrow r_s(f_i, a_i, f_j, a_j)$ `/* Eq. (1.3) (or alternative) */`
      $y \leftarrow y \cup (p_i, p_j, r_{ij})$ `/* Consider` p_i `and` p_j `for parameter adjustment */`
    end
    $j \leftarrow j + 1$
  end
  $i \leftarrow i + 1$
end

Frequency bashing and amplitude whacking define the changes that we would like to make to the frequencies and amplitudes of partial pairs identified in Algorithm 4. However, in signals we cannot simply change the frequency or amplitude directly as we will in additive synthesis in Section 3.3. Instead, we must remove partials from their original signals and then

mix adjusted signals back in. To adjust a partial in a signal x, the partial is removed from the signal and, in parallel, isolated. The mean frequency and frequency range of the partial are used to determine a center frequency f_c and bandwidth Bw of an amplitude-complementary pair of H_N, a notch filter, and H_P, a bandpass filter, such that

$$H_N(f_c, Bw) + H_P(f_c, Bw) = 1 \ . \tag{3.1}$$

Butterworth filters are used for their flat response in the passband region.

After constructing the filters H_N and H_P they are applied to the signal x. The output of H_N contains the entirety of x except the portion corresponding to the filtered partial. The output of H_P is largely sinusoidal and contains the partial whose amplitude or frequency we will modify. When amplitude whacking, amplitude is determined as the mean amplitude across the lifetime of the partial. Gain is applied to the partial so that the mean amplitude is the adjusted amplitude a^* determined by (2.8). Frequency is determined by the mean frequency of the partial across its lifetime, and the new frequency f^* is determined by (2.3). Single sideband modulation is used to frequency shift the entire output of the peaking filter up or down. The processed output of H_P is mixed with the output of H_N to create the signal with a now modified partial.

In the case where a signal x has k partials to be adjusted, the frequency domain representation of the output signal x' is represented by

$$X' = X \cdot \prod_{i=1}^{k} H_{N_k}(f_{c_i}, Q_i) + \sum_{i=1}^{k} \Delta_i(H_{P_i}(f_{c_i}, Q_i) \cdot X) \ , \tag{3.2}$$

where Δ_i changes either the amplitude or frequency of the partial isolated in the peaking filter. Figure 3.1 depicts the audio processing of partials, with notch filters applied in series and bandpass filters applied in parallel so that their outputs can be adjusted before summing together. Forward and backward filtering is used for zero-phase filtering. Cross-fades are applied between the original signal x and the modified signal x' so that filtered signals with adjusted partials are

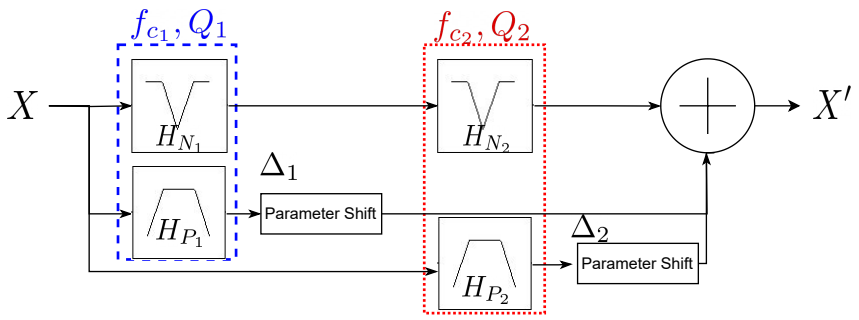

Figure 3.1. Partials of a signal x are adjusted by removing them using notch filters, and in parallel, isolating them from x using amplitude-complementary peaking bandpass filters. Complementary filters share the same center frequency f_{c_k} and quality factor Q_k. The output of the peaking filters are processed to adjust either the frequency or the amplitude by some amount Δ_k.

only heard during the lifetime of the partials that cause roughness.

This analysis and filter-bank processing approach is appropriate for the frequency bashing and amplitude whacking algorithms, where the final mix is monophonic and a majority of the signals are unaltered. Chapter 2 also introduced an algorithm for controlling roughness using two-dimensional panning on all partials in a mix. As this algorithm requires separating all partials from each signal and has no mechanism for non-sinusoidal noise, it is more applicable to experimental composition using additive synthesis parameters than processing high fidelity sound files. As such, no offline implementation for audio signals is attempted here. An implementation for real-time spatialization of sinusoids is provided in Section 3.3, with speculations on roughness-based panning algorithms for audio files left to the conclusion of this chapter.

3.2.1 Offline application to audio mixing

The offline methods for frequency bashing and audio whacking can be applied to mixing, sound design and experimental music composition. Several examples are presented and analyzed here, with associated audio available on the supplemental site[1].

[1] https://jeremyhyrkas.com/dissertation

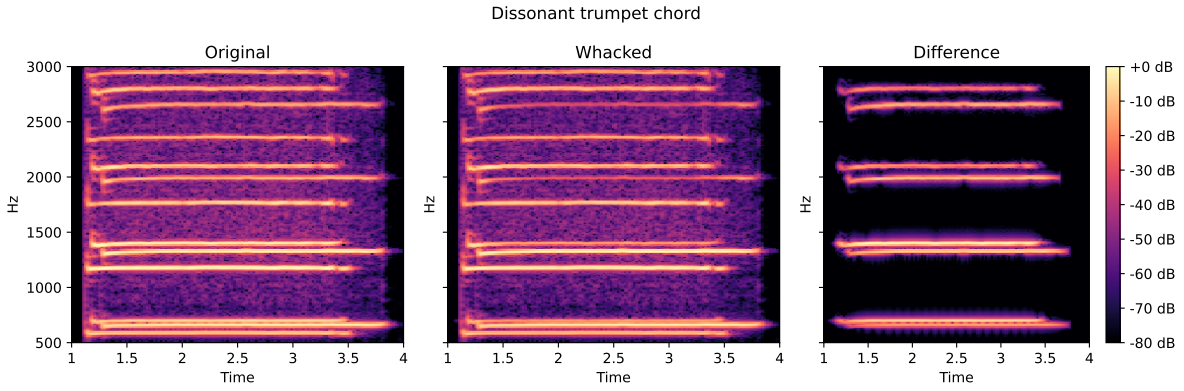

Figure 3.2. A tonally dissonant chord has its roughness reduced by the offline amplitude whacking algorithm. The spectral difference between the original (left) and processed track (middle) is shown in the right, showing a level adjustment of eight partials.

Roughness control as a mixing tool

The first use case explores roughness minimization as a tool in audio engineering, where the algorithms would be applied to signals that are to be combined in a mix after each channel has been independently processed. These examples demonstrate a potential of reducing roughness when mixing without the use of tuning or manually intensive EQs applied to each track.

One creative application of roughness control is to reduce the sensory dissonance of a tonally dissonant chord, allowing a mix to contain complex and discordant intervals while reducing the physical discomfort caused by auditory roughness. This can be accomplished using amplitude whacking to automatically EQ each signal in the chord and changing the amplitude of select partials that cause the most beating. One example is shown in Figure 3.2, in which three trumpet signals play the interval of a root, major second and minor third. The chord is tonally dissonant and contains significant roughness in the original mix. The processed mix maintains tonal dissonance while reducing beating significantly. The right side of the figure shows the spectrogram of the difference between the original and processed signals. Four pairs of partials have their amplitudes adjusted, amounting to eight filters and gains that would otherwise need to be manually tuned and applied by an engineer to create the same sonic effect.

Figure 3.3 demonstrates the dynamic nature of the offline algorithms. Amplitude whack-

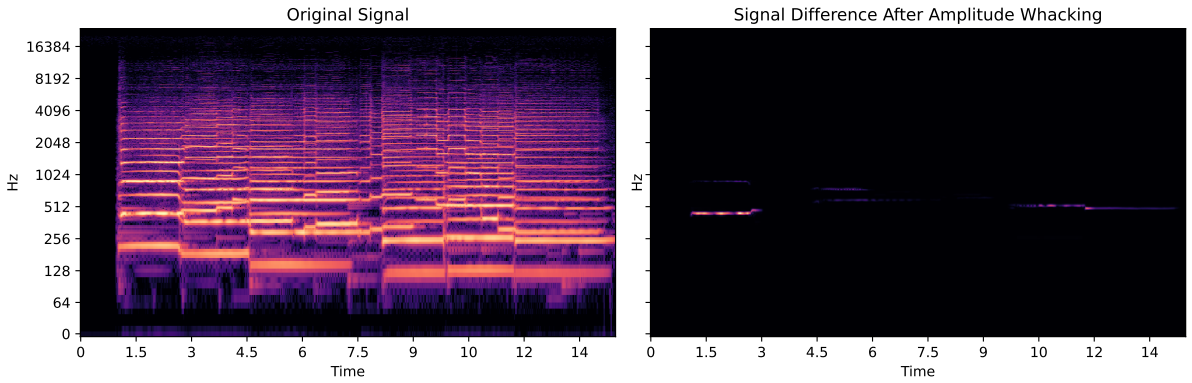

Figure 3.3. Left: a spectrogram of a dynamic horn line mixture featuring audio of three players performing. **Right**: the spectral difference of an amplitude whacked version of the horn line and the original. Amplitudes of partials are only adjusted in areas of roughness, after which point the original signals are faded back in. This example is time-varying and demonstrates subtle changes that achieve a reduction in roughness without retuning any notes.

ing is applied to a melody and harmony played by three horn players. The spectrogram of the original signal is depicted on the left side of the figure, and the spectrogram of the difference signal between the original and modified signal is shown on the right. Most of the amplitude whacked signal is identical (i.e., no energy in the difference signal), but subjectively there is noticeably less modulation than the original when listening comparatively. This example demonstrates the very subtle changes made to the signal, as well as the preservation of the original signal during periods of time where no roughness is present.

Roughness control as an audio effect

While thus far the focus has been on roughness minimization, introducing roughness into a signal can also be used as a creative tool. Figure 3.4 shows spectrograms of a 16-voice choir before and after frequency bashing for increased roughness. In this version of the algorithm, nearby partials are adjusted so that they are 0.24 critical bands apart, the point of maximum dissonance in the Plomp and Levelt curve. A significant amount of modulation is introduced, similar to ring modulation, but without destroying the perception of the major triad of the original chord. This use case shows frequency bashing as an audio effect that may be useful for a

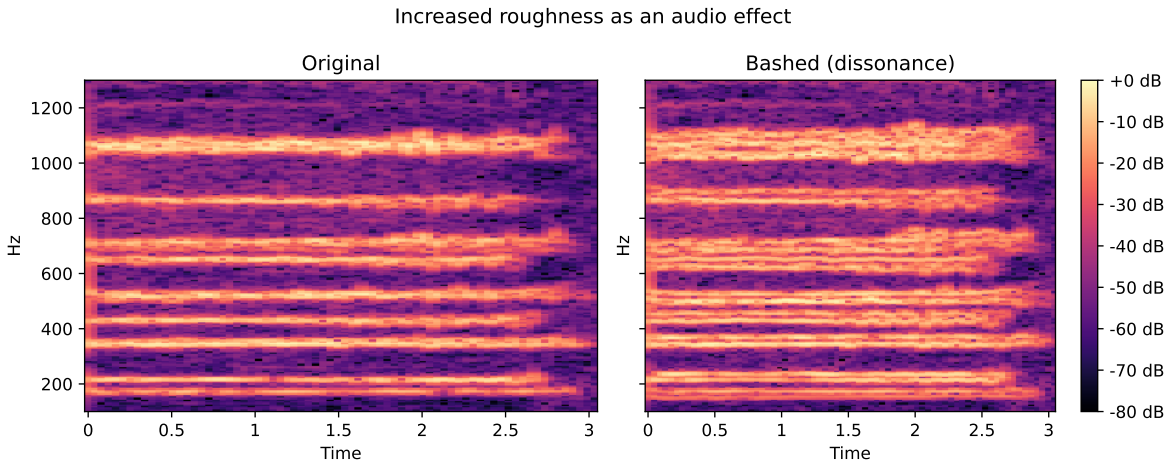

Figure 3.4. Frequency bashing to create dissonance as an audio effect. 16 voices of a choir are bashed for maximum roughness, causing a spread in partials that is particularly noticeable in the 600–700 Hz and 1000–1100 Hz regions. While the mix maintains the tonal consonance of the original chord, perceptually the sense of tuning is perturbed by the creation of non-integer spacing between some harmonics.

composer who wishes to introduce dissonance without detuning or modulating the entire signal.

Hard bashing is another form of frequency bashing where all adjusted partials are set to a fixed difference in Hz rather than using the Plomp and Levelt curve to determine new frequencies. The result is similar to the tremolo audio effect, but the modulation is only heard when roughness is present in the original signal and only in certain bands of the frequency spectrum. Figure 3.5 shows such an example, comparing a 3 Hz tremolo to hard bashing with a 3 Hz difference. While the time-domain waveform of hard bashing does not show the obvious modulation of the tremolo signal, the spectrum shows 3 Hz modulation around select partials. This use of frequency bashing recasts it as a subtle, shimmering tremolo-like effect that can be used in place of full signal amplitude modulation when preferable to a composer.

3.3 Real-time implementations for additive synthesis

The algorithms in Chapter 2 can be performed in a straightforward manner on additive synthesis parameters, as no audio analysis is required. In additive synthesis, frequencies and amplitudes are used to control an oscillator bank, with each oscillator receiving one frequency

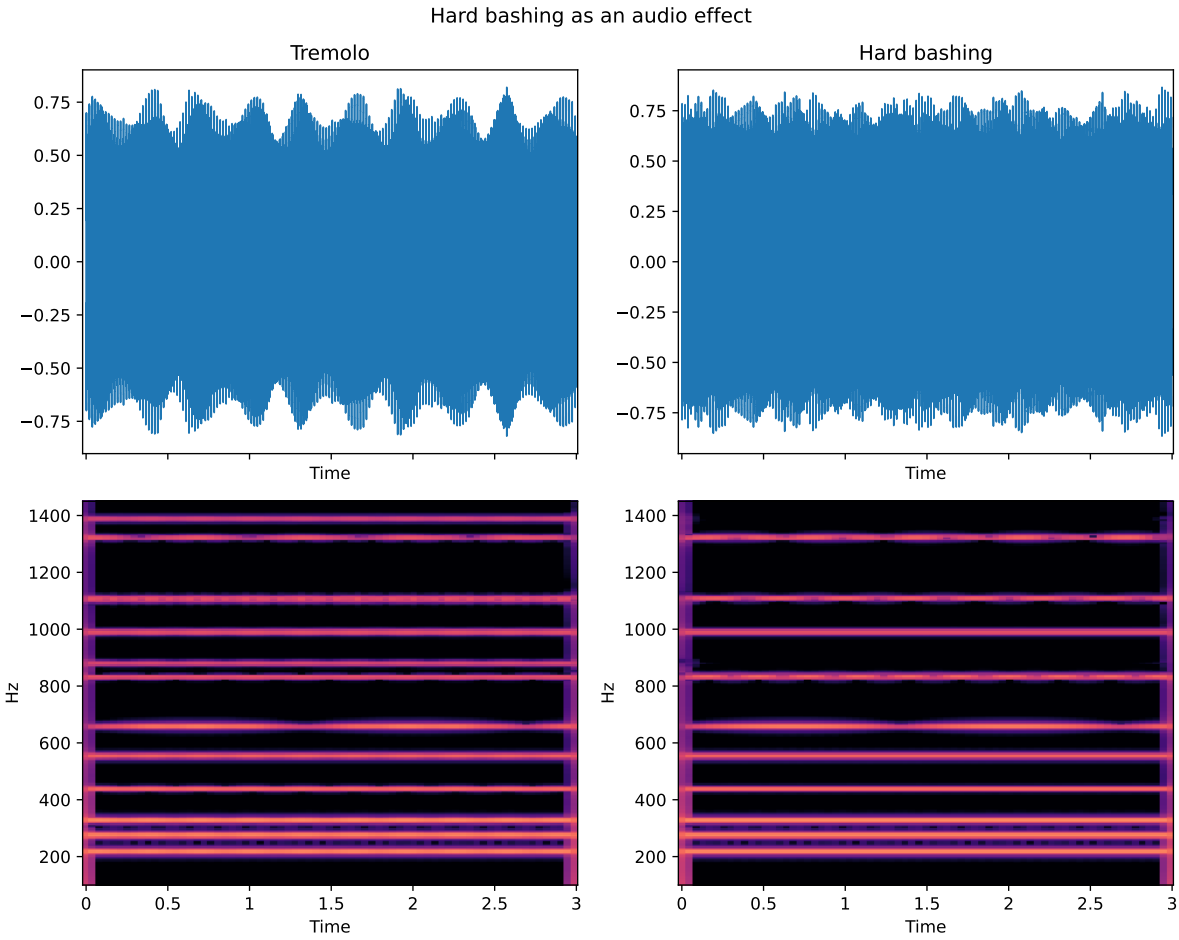

Figure 3.5. Tremolo at 3 Hz (left) is compared to hard bashing to a fixed difference of 3 Hz (right). The time-domain waveform of hard bashing does not show the obvious modulation pattern of tremolo, but the spectrogram reveals 3 Hz beating in regions that previously contained roughness (see the partials at roughly 800 Hz, 1100 Hz, 1300 Hz).

and amplitude per synthesis step. The algorithms for roughness control are run at the block rate (as opposed to the sample rate) due to their use of sorting and iteration. For each DSP block, frequency and amplitude pairs are intercepted and adjusted before they reach the oscillator bank, making the algorithms real-time with respect to the control rate of the system. However, some modifications are necessary to reduce the computational load, particularly in cases where a large number of oscillators (and therefore parameters) are used in synthesis.

When changing amplitudes in real time, it can be expensive to compute the masking curve using Equation (2.4) at every step. The real-time implementation transfers all amplitude from one partial to another when $p = 1$ instead of just enough amplitude to meet the masking curve. This solution has the potential to boost the level of partials by an unwanted amount, causing more timbral change than the offline algorithm. It also results in a portion of the parameter space of p being theoretically inaudible once one partial masks another. This issue could be avoided by pre-computing and quantizing masking curves.

While sinusoidal modeling in the offline method tracks the lifetime of partials, the real-time implementation instead evaluates parameters at every step in a memoryless fashion. As a result, there can be a rapid fluctuation in parameter adjustment in the case where two nearby partials are nearly the same amplitude and fluctuate between which one is louder. While not currently implemented, a short-memory system or a streaming sinusoidal model could be adapted to commit to the choice of adjusted partial.

Algorithm 3 for roughness-based panning can be implemented largely as defined in Chapter 2. Because the algorithm does not change the frequency and amplitude parameters of synthesis and instead assigns an azimuth panning value, it can be used in conjunction with the other algorithms and does not need to be evaluated at every synthesis step. The main consideration for real-time use is the choice of parameter m_{it}, which determines the number of iterative search steps for each evaluation of the algorithm.

The real-time algorithms described in this section are implemented as externals for the Max/MSP computer music environment. Algorithms 1 and 2 are implemented in the *basher*

and *whacker* objects, which take input a list of frequency/amplitude pairs and output one list of frequencies and one list of amplitudes. The basher creates a new list of frequencies while amplitudes are passed through unchanged, while amplitudes are adjusted by whacker with the frequencies unchanged. The outputs of each external can be connected to multichannel Max objects for additive synthesis. The *disspanner2d* (dissonance panning in two dimensions) external additionally takes azimuth panning for each pair as an input and outputs a new pair with each *bang* it receives, allowing panning to take place at a different rate than the other externals.

The input list of sinusoidal parameters can come from an analysis-resynthesis system for sinusoidal modeling [36, 37], but composers are free to use any method for generating sinusoidal parameters. Algorithmic parameters of the externals include the Bark range of search and adjustment (B_L and B_H), the percentage change parameter p, and a toggle to change from decreasing roughness to increasing roughness. These parameters can be fixed or adjusted by the user algorithmically or manually on-the-fly as a musical effect.

3.3.1 Online algorithms applied to composition

The real-time implementations of roughness control algorithms are intended for use in experimental music composition. In contrast to the evaluations of the offline algorithms, the real-time implementations were used by the author to create a multichannel composition. What follows is a brief description of the composition and subjective observations about the usefulness and drawbacks of the algorithms for music making.

It has been the experience of the author that it is beneficial to perform roughness control on slowly evolving sonic material, or in an extreme manner (such as introducing significant roughness to an otherwise smooth signal), for the effect to be immediately obvious to the listener. Reducing roughness in dynamic signals, as done in Section 3.2.1, is often noticeable only in comparison, making it more suited to mixing applications than as a salient musical effect in a composition. To showcase roughness control in a musical piece, it is useful to apply roughness control algorithms on droning sinusoids that allow for slow changes in user parameters so that

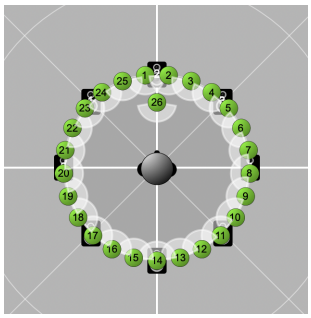

Figure 3.6. Initial placement of sound sources in "The Hearing Test." Sources 1–25 are sinusoids and are panned algorithmically around the circle as the piece progresses. Source 26 plays samples from an instructional video and is manually panned throughout the piece.

the listener can perceive the changes.

Preliminary versions of the Max externals described in Section 3.3 were used in a composition that musically investigates roughness, dissonance, and the experience of listening tests. The sound material of the piece largely consists of inharmonic drones created using additive synthesis. The piece, titled "The Hearing Test,"[2] uses externals to control the amplitude, frequency and spatial panning of partials to increase or reduce auditory roughness. Samples from an instructional video on eye exams are used to simulate an ear exam where the listener is asked to rate the comfort of several adjustments, presumably for some imaginary auditory device that attempts to smooth out the listening experience of the wearer.

To begin, sine tones are placed equidistant around a listener's head (see Figure 3.6). The piece opens and closes with droning chords made up of inharmonic bell tones. The first 24 sources in Figure 3.6 are bell tone partials, with a 25th source reserved for intermittent sine tone sweeps and beeps, and a 26th source for narration from the instructional video. In the opening drone, roughness is controlled using amplitude whacking, with adjustments between unmodified and roughness reduced partials triggered alongside prompts from the dialogue. Figure 3.7 shows the spectrum of the unmodified and modified drones. Modifications are mostly done below 500 Hz, where many partials build up and cause significant beating due to the smaller critical bandwidth in the lower frequency range.

[2]Presented at the 2024 International Computer Music Conference

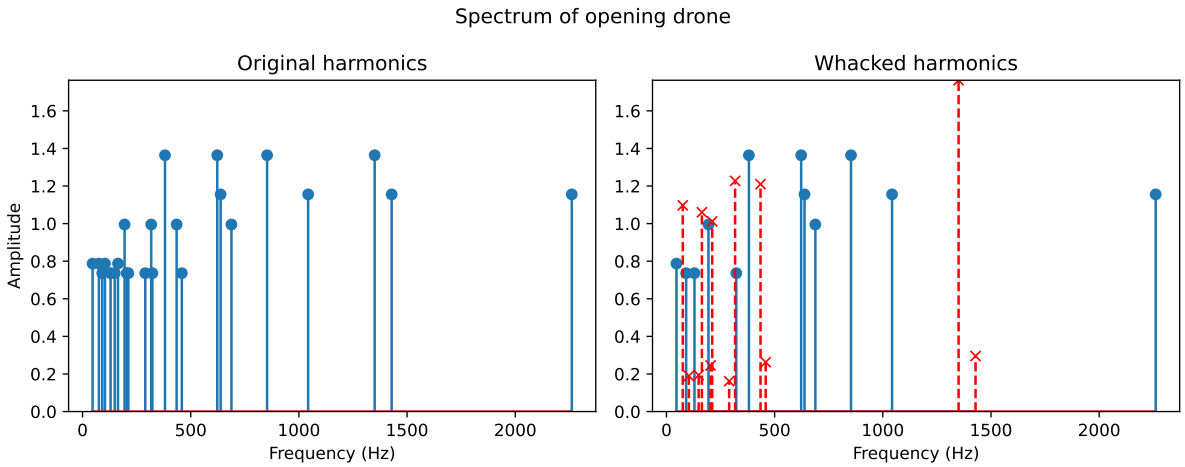

Figure 3.7. The spectrum of the opening drone in "The Hearing Test." The piece algorithmically slides between the original spectrum (left) and the amplitude whacked spectrum (right) while dialogue from an eye exam instructional video plays.

The closing drone maintains a constant spectrum, but the spatial placement of partials around the listener are swapped between panning for minimum and maximum roughness. Each pan is triggered by 100 optimization steps (i.e., swaps in Algorithm 3) with slow panning applied to avoid instantaneous jumps in placement. In the multichannel version of the piece, panning is implemented using the Spat [38] package's implementation of the VBAP [39] algorithm.

Because each step is an unoptimized search, every iteration that pans for consonance or dissonance results in a slightly different panning scheme with different per-ear roughness calculations. Table 3.1 lists the roughness calculation for three iterations using Equation 2.9 to determine per-ear amplitudes and Equation 2.1 to calculate the roughness for each ear. In each iteration, the roughness is decreased overall and in each ear when panning for consonance over dissonance. It appears that there is more variation in the dissonant panning positions. This is likely due to the sonic material, which has many clashing partials that can be placed close to each other, causing significant roughness in one ear.

The Hearing Test sits conceptually somewhere between a composition, a listening test and a live demonstration of roughness control algorithms. The choice to merge these three elements was partially chosen due to the novelty of roughness control as a musical parameter

Table 3.1. Per-ear roughness for the closing drone of The Hearing Test. The piece algorithmically switches between panning partials for consonance and dissonance. The table shows three iterations of switches and the resulting roughness calculations. Because the algorithm does not perform an optimized search, different local minima (and therefore panning values for each partials) are found with each iteration.

Roughness calculations per ear			
Panning Goal	Iteration	Left Ear	Right Ear
Consonance	1	0.52	0.50
Dissonance	1	0.96	0.88
Consonance	2	0.55	0.50
Dissonance	2	1.1	0.72
Consonance	3	0.52	0.51
Dissonance	3	0.83	1.01

and an uncertainty around listeners' perception of the sonic effects. In anecdotal remarks from early listeners of the piece, it was noted that the change in roughness was often experienced as a change in ear pressure (i.e., a sense of build-up and relief); this was particularly true in sections where roughness-based panning was used. Because there are other elements that affect the general pleasantness or tolerability of sound, such as differing power levels in each ear and the effect of frequency on perceived volume [40], a scientific verification of the effectiveness of roughness modification in a listening test would require careful control of these parameters. It is possible that independent control of roughness independent of tuning is unfamiliar enough to listeners that tests would reveal mixed or muddled results. Regardless, The Hearing Test acts as a demonstration of the creative usefulness of real-time roughness control algorithms, albeit without proving the algorithms' ability to reduce perceive roughness for the listener.

3.4 Conclusion

Models of auditory roughness have endured as a research topic in psychoacoustics and computer music due to the strong link between roughness and perceived dissonance. The most common connection between roughness and music is the use of roughness models to determine pitch shifts in the context of note intervals or sound mixes, which likely dates back to the

origin of studies into sensory dissonance as it relates to consonant intervals in Western music theory. However, three algorithms described in Part I demonstrate that roughness can be treated as a musical parameter that can be controlled without accounting for tuning or pitch. Sound engineering, audio effects, and compositional control of additive synthesis parameters have been identified as three potential use cases for these algorithms.

Roughness as a musical parameter is often explored implicitly in compositions through the choice of tunings and intervals that either lack or prominently feature beating in the harmonics of the sounding instruments. Few compositions are explicitly composed to emphasize the sensation of roughness, and even fewer explore roughness as a spatial feature. Brian Hansen has composed two such pieces: *Arch*, in which sine tones that are 25% of a critical band apart are placed in space so listeners can freely move and experience different dissonant points, and *Sensations of Tone*, where users control the frequency, amplitude and spatial parameters of three tones in space to explore their perceived dissonance with relation to a computed dissonance curve [41]. Where Hansen's pieces allow the listener to explore roughness interactively, The Hearing Test provides a more curated approach while still introducing moments of algorithmic control and uncertainty. All pieces sit at the intersection of art music and scientific experiment, perhaps the most natural technique when compositionally exploring psychoacoustic phenomena.

There are several possible directions to push this research agenda forward. First, while the algorithms proposed here will reduce roughness of sounds as modeled mathemetically by Equations (1.2), (1.3), or (1.5), a listener's experience of dissonance is complex and involves more factors than just the roughness of sound. Listening tests would be informative to confirm that listeners perceive the modified signals as more or less dissonant in the sensory context, although careful control of perceived loudness is critical. Second, it is possible to improve the implementations of both the offline audio-based and real-time additive synthesis-based algorithms to improve usability. Real-time estimation of sinusoidal partials (particularly methods that maintain tracks across time) could be incorporated into the audio processing algorithms to allow for real-time control of roughness at the audio block rate, similar to the real-time implementation

in Max. Track memory could also be incorporated into the Max externals to avoid the rapidly changing parameter issue described in Section 3.3. Finally, while spatially panning for roughness control is only explored here using sinusoids, spatial control of roughness could be applied to automatic panning of audio signals without breaking them apart into sinusoidal partials. It is well known that listeners have difficulty identifying spatial placement of sinusoids, but more complex signals could be analyzed and placed throughout a space to control the roughness in each ear for a centered listener. The application of computational models of roughness to spatialization is an underexplored area of research and could have application in audio engineering, sound design and composition.

Composers and performers often introduce or reduce roughness in sound mixtures intentionally, even if indirectly by considering the tuning of instruments. However, there are other compositional contexts where signals that were not originally created to be combined are nonetheless mixed together, such as a DJ combining tracks in a live mix. A related problem of interest to the author is concatenative synthesis systems where sounds from an audio corpus are combined and overlaid in time. Overlapping out-of-context sound sources, particularly those that are of the same pitch class but perhaps not perfectly in tune, can easily introduce roughness to the combined signal. Incorporating signal-based roughness reduction algorithms could be used to reduce roughness in overlapped signals in a concatenative synthesis system.

Chapter 3 contains material from the research paper *Algorithms for Roughness Control Using Frequency Shifting and Attenuation of Partials in Audio*, published in the Proceedings of the International Symposium on Computer Music Multidisciplinary Research (CMMR 2023). The dissertation author was the primary investigator and author of this work.

Part II

Vibrato matching

Chapter 4

Perceptual motivation for vibrato matching and existing vibrato models

4.1 Vibrato and source separation

Musical performances featuring acoustic instruments often contain moments of musicians playing in unison or harmony to form complex timbres that may be perceived as a fused sonic source, as opposed to multiple instruments. Players may adjust their volume, pitch, vibrato, or other performance techniques to more closely match those of the other performers. There are several analogous practices in electronic music where audio samples are overlaid in time to approximate the timbre of a target sound; examples of such practices include, but are not limited to: target-based concatenative synthesis [6], automated orchestration [42] and audio mosaicing [42]. Unlike in live performance, samples from an audio database exist without musical context and may need to be edited or processed to blend together more naturally into a more fused sound. One common approach is sound morphing, where multiple signals are analyzed and resynthesized into a single signal that contains some timbral qualities of each individual signal [11]. However, the fidelity and potential richness of multiple overlaid sounds may be lost due to the reduction of signals or artifacts of the analysis or resynthesis. As in Part I, this collection of chapters presents methods for processing multiple signals in the interest of creating a perceptual effect upon their mixing (here, the perception of a more uniform sound mixture) while still maintaining the presence of two or more separate source signals.

Vibrato is an effect found in many musical styles whereby the musician modulates the frequency of a sustained, usually harmonic, tone. While pitch modulation is the musical goal, natural vibrato is often accompanied by changes in amplitude, and in many instruments, the changes in amplitude are more perceptually salient than changes in pitch. Frequency and amplitude modulation (FM/AM) patterns in vibrato tend to be characteristic of, and unique to, a particular instrument and performer.

Humans are remarkably good at counting the number of instruments in a mix, even when playing in unison, up to a limit of three to four instruments [43]. These results hold for controlled laboratory settings and more general online surveys [44]. In the case of musicians playing in

unison, one explanation for a listener's ability to detect multiple instruments is the presence of vibrato and its connection to Gestalt's Common Fate principle applied to psychoacoustics [45], where related sound partials have related perceived movement. The vibrato of one source gives its partials a similar rate of change in frequency and amplitude. Partials from different instruments have different rates of change, even when they overlap in frequency, giving listeners a cue to the presence of multiple sources.

Modulation patterns can be also be leveraged in automatic source separation. Source separation algorithms take a signal that contains several sources and try to separate the sources into individual signals. Traditional methods for source separation struggle in the presence of unison signals due to their overlapping harmonics. Unison source separation algorithms [46] use a two-dimensional Fourier Transform to capture not only the frequency content of a signal but also modulation patterns in the frequency domain. The state-of-the-art method, called the Common Fate Transform [47] after the Gestalt principle, has been shown to be successful at separating sources playing in unison by analyzing their vibrato patterns and separating partials based on the AM and FM patterns of each source.

The methods presented throughout Part II are informed by the inverse of audio source separation - that is, we look to process signals so that when they are mixed, a source separation algorithm (and ideally a listener) will have difficulty separating the mix back to individual sources. Based on the psychoacoustic theory of Common Fate and the success of unison source separation algorithms in isolating signals based on vibrato patterns, it stands to reason that two signals (particularly when playing in unison) could be made to sound more uniform and fused if their vibrato patterns were matched exactly. The remainder of this chapter focuses on methods for analyzing a signal's vibrato so that modification for sound blending can be applied. The methods for transferring, suppressing and matching vibrato patterns are left to Chapters 5 and 6.

4.2 Vibrato analysis and sinusoidal resynthesis

Vibrato analysis is the subject of many previous studies, likely as a result of its perceptual salience, variance between players and instruments, and its perceived musicality in some genres and styles. Typically, a signal with vibrato is analyzed for both its AM and FM patterns. One illustrative example is the Vibrato Analysis Toolbox [48]. In this package, a monophonic signal is analyzed to determine its fundamental frequency (f_0). Bandpass filters are centered at integer multiples of f_0 to isolate each harmonic. Butterworth filters are chosen due to their relatively flat passband. Finally, the frequency and amplitude signal of each harmonic are extracted using the instantaneous amplitude and frequency from the analytic signal of each harmonic.

In a pair of studies using this toolbox [48, 49], the authors draw several conclusions about the natural of acoustic vibrato. FM patterns are largely identical per harmonic across various instruments. Intuitively, this is expected and welcome, as it implies that the pitch bending caused by vibrato does not cause signals to become inharmonic. However, patterns in AM can differ between harmonics, both in the extent of modulation and the phase of modulation. Additionally, FM and AM modulation patterns can be out of phase, likely due a difference in the acoustic mechanisms that drive frequency and amplitude modulation in an instrument. To further complicate matters, AM patterns can exist in the residual (non-harmonic) components of the signal, and these patterns can also differ across the frequency spectrum [50]. These differing patterns in AM perhaps complicate the assumptions of the Common Fate transform for source separation, as differences in modulation depth and phase could disrupt the grouping of partials from a source (although the modulation rate is shared across harmonics).

While vibrato analysis remains an area of study, vibrato manipulation of a recorded signal is a less explored topic. One state-of-the-art method is to analyze a signal, model its vibrato and resynthesize the signal entirely [50]. In this method, the time-varying frequency and amplitude signal (track) of each harmonic are modeled, as well as the amplitude of the spectral envelope of the residual noise. Each parameter track is extracted and modeled as the

sum of a low-passed signal and a residual high-passed signal, where the residual contains the modulation patterns caused by vibrato. The authors use a high-order FIR lowpass filter and forward-backward filtering to perform zero-phase filtering.

In this setup, the frequency of the kth harmonic is modeled as

$$\omega_k(n) = \overline{\omega}_k(n) + \alpha\tilde{\omega}_k(n), \tag{4.1}$$

where $\overline{\omega}_k(n)$ is the low-passed frequency track and $\tilde{\omega}_k(n)$ is the high-passed residual. The parameter α controls the vibrato extent. Setting $\alpha = 1$ maintains the original signal's vibrato, while $\alpha = 0$ suppresses vibrato entirely. Vibrato can be lessened from the original using $0 < \alpha < 1$, and $\alpha > 1$ increases the extent of vibrato when compared to the original.
This framework similarly computes the amplitude of the kth harmonic as

$$A_k(n) = \overline{A}_k(n) + \alpha\tilde{A}_k(n),^{1} \tag{4.2}$$

and the spectral envelope of the residual noise as

$$N(\omega, m) = \overline{N}(\omega, m) + \alpha\tilde{N}(\omega, m). \tag{4.3}$$

where $N(\omega, m)$ contains the envelope for time m and frequency ω.

Vibrato control using this framework is performed by resynthesizing the original signal using a sine-plus-noise model [51]. The results are generally positive but rely on several factors. As with all sinusoidal modeling, parameters such as the number of harmonics to model, amplitude thresholds, time thresholds, and quality of initial analysis play a large role in the sonic quality of the resynthesis. There are times when it may be preferable for a user to not resynthesize a signal using sinusoidal modeling, either for aesthetic reasons or to avoid a costly analysis in

[1]This formulation is rewritten from the original [50], where $S(\omega, m)$ is the spectral envelope of the signal for time step m and frequency ω and the amplitude track is $A_k(n) = S(\omega_k(n), m)$ interpolated to signal length.

both CPU time and data size for a large number of signals (such as in the motivating case of concatenative synthesis using a database of sounds). This framework, therefore, will be used as a point of comparison for other techniques for vibrato control explored in the rest of Part II.

4.3 Vibrato using a time-varying delay line

Another common model of vibrato (strictly treated as a change in frequency) is as a particular parameterization of FM. In this model, vibrato is described as a wavering of pitch that oscillates about a central sounding frequency. While oscillation is not strictly sinusoidal in most acoustic vibrato, in this simplified model the instantaneous frequency of a discrete-time (with sampling rate $f_s = 1/T$) sinusoidal component with vibrato is represented by

$$\omega_i(n) = \omega_c - d\cos(\omega_m nT) \text{ rad/s}, \tag{4.4}$$

a familiar FM expression where a center frequency ω_c is modulated by a sinusoid of frequency ω_m and amplitude d that controls the depth of the modulation (maximum deviation from the center frequency). The corresponding instantaneous phase of the sinusoid is then given by integrating (4.4) with respect to discrete time nT to yield

$$\theta_i(n) = \int_0^{nT} \omega_i(n)\, dnT = \omega_c nT - I\sin(\omega_m nT) + \phi_c, \tag{4.5}$$

where ϕ_c is an added phase term (a constant of integration) and $I = d/\omega_m$ is the *index of modulation*, the well-known synthesis parameter used to control the spectral spread (approximate number of sidebands) produced by the modulation. When I and ω_m are small, the modulation is perceived as a time-varying sounding frequency rather than as a change in sound spectrum - in other words, small I and ω_m can be thought of as controlling vibrato instead of timbre.

If operating on an existing signal (and not synthesizing directly from sinusoids), the above modulation can be accomplished using a time-varying delay line (TVDL). If, for example,

the complex exponential sinusoid $z(n) = e^{j\omega_c nT}$ with frequency ω_c is input into a delay line with time-varying delay function

$$D_m(n) = \frac{I}{\omega_c}\sin(\omega_m nT)f_s + \frac{I}{\omega_c}fs + D_0, \tag{4.6}$$

where D_0 is an offset added to the delay function to ensure it is positive and (4.6) has a maximum value of $\max\{D_m(n)\}$, then the modulated output is given by

$$z_m(n) = z(n - D_m(n)) = e^{j\omega_c(n-D_m(n))T}, \tag{4.7}$$

having instantaneous phase

$$\theta_m(n) = \arg\{z_m(n)\} = \omega_c nT - \omega_c D_m(n)T = \omega_c nT - I\sin(\omega_m nT) - I, \tag{4.8}$$

equivalent to Eq. (4.5) with $\phi_c = I$.

An example more closely related to acoustic signals with vibrato, which typically have harmonically related sound partials, can be seen with an input signal represented by

$$z_k(n) = \sum_{k=1}^{K} e^{jk\omega_c nT}, \tag{4.9}$$

which, when modulated by delay function (4.6), yields output signal

$$z_{k,m}(n) = \sum_{k=1}^{K} e^{j(k\omega_c nT - k\omega_c D_m(n)T)} = \sum_{k=1}^{K} e^{j(k\omega_c nT - kI\sin(\omega_m nT) - kI)}, \tag{4.10}$$

showing an amplitude of kI for the sinusoidal term in the phase and thus an effective index of modulation that increases by a factor of k for upper harmonics at frequencies $k\omega_c$. As shown in Figure 4.1, this increase in the index of modulation results in a greater spectral spread in higher harmonics (left) and, equivalently, a larger deviations in instantaneous frequencies (right).

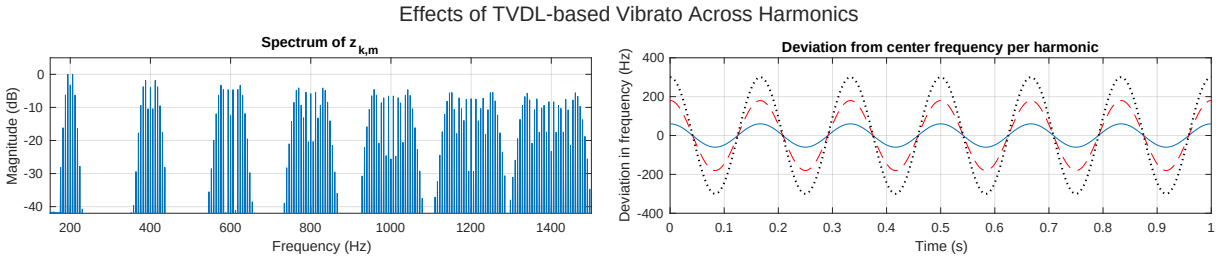

Figure 4.1. The signal created by modulating a sum of harmonically-related sinusoids (4.9) using a TVDL has a factor of k increase in the spectral spread (left) and time-varying instantaneous frequency modulation depth (right) in the kth harmonic as seen by (4.10).

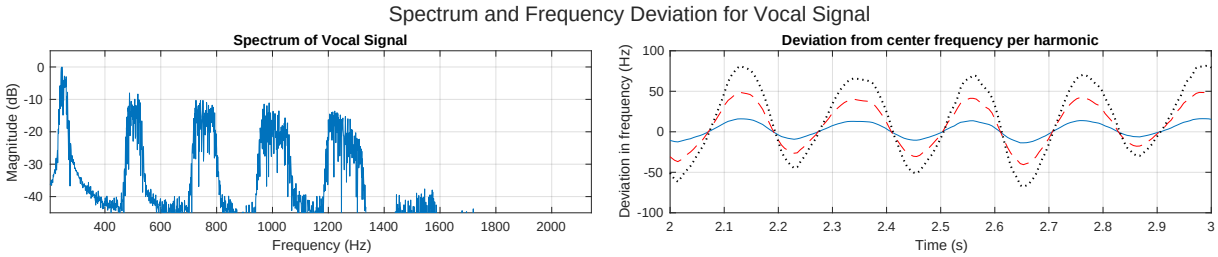

Figure 4.2. The spectrum and frequency deviation of harmonics of a vocalist with vibrato.

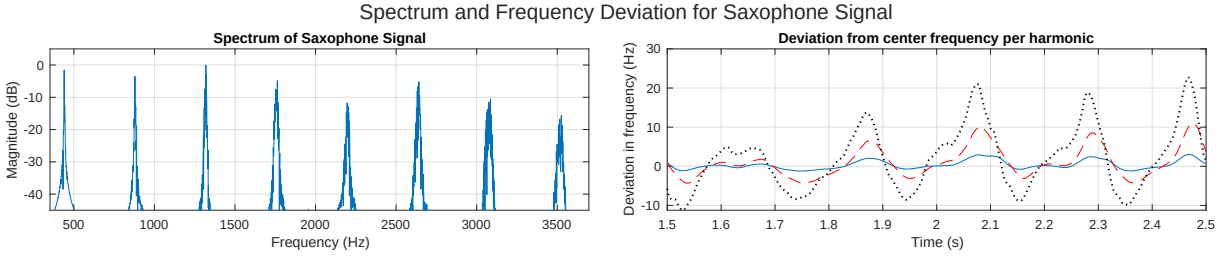

Figure 4.3. The spectrum and frequency deviation of harmonics of a saxophone with vibrato.

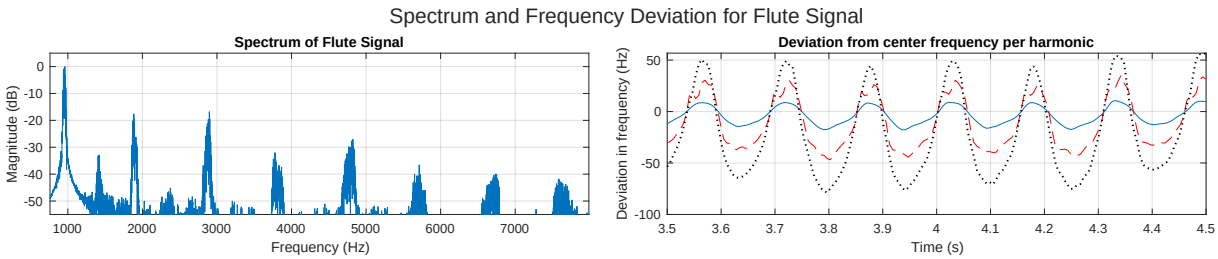

Figure 4.4. The spectrum and frequency deviation of harmonics of a flute with vibrato.

The increase of modulation depth in the upper harmonics imparted by the delay line is consistent with the behavior of upper harmonics imparted by vibrato in real musical instruments, a validation of the TVDL as an appropriate method for applying the FM component of vibrato. Figures 4.2–4.4 show the spectra of vocal, saxophone, and flute[2] signals with vibrato, as well as the frequency deviation of several harmonics of each signal. The frequency of each harmonic is derived by taking the Short-time Fourier transform (STFT) of each signal and picking peaks in the frequency domain close to harmonic values based on each signal's f_0, which is known a priori. Frequency deviations are calculated by subtracting the mean frequency of each harmonic.

Similar to the patterns in Figure 4.1, these spectra show an increase in sidebands in the upper harmonics and an increase in frequency deviation with each successive harmonic. However, the FM patterns are not sinusoidal or precisely periodic. Figure 4.5 shows the instantaneous frequencies of the first six harmonics of the flute signal shown in Figure 4.4. The frequency of the kth harmonic is approximately equal to k times the fundamental frequency contour, the same frequency deviation that would be imparted in a TVDL. The similarities between natural vibrato and vibrato caused by a TVDL validate the use of the TVDL to impart vibrato, but point to the need for more complex modulating delay functions $D_m(n)$ to approximate real vibrato.

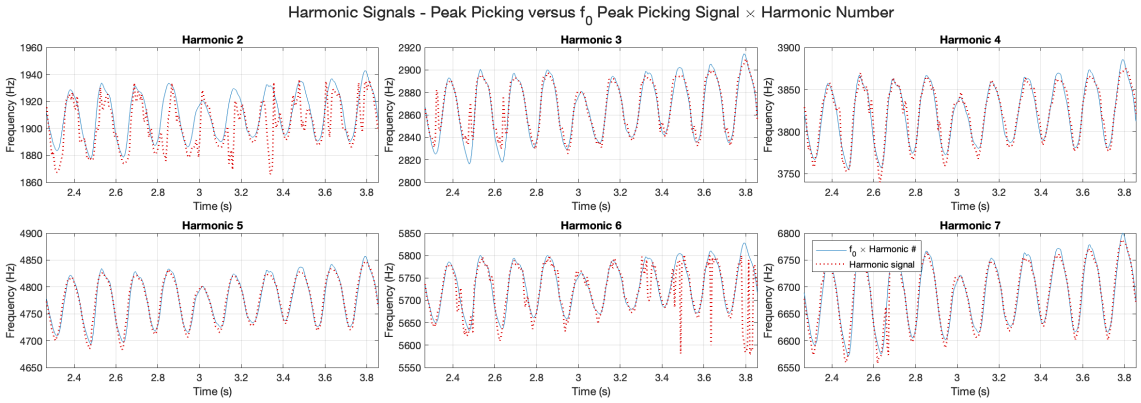

Figure 4.5. Harmonic peak picking versus f_0 signal times harmonic number. The frequency contour of upper harmonics closely matches those of the first harmonic, with visible deviations more likely to be caused by analysis error than true differences in the frequency tracks.

[2]This sample generously provided by Axel Roebel. All other samples were recorded in-house.

4.4 Conclusion

Vibrato is a perceptually salient musical characteristic in terms of human and machine source separation. Its complexity in terms of AM and FM patterns has been at the heart of several research projects in acoustics and music information retrieval, often with the goal of modeling natural vibrato and sometimes with the goal of modifying it. Sinusoidal modeling is the current gold standard for vibrato control, allowing for the extent of vibrato to be expanded or reduced (down to full removal), but the sonic results depend on a full resynthesis of the signal and an accurate analysis of all relevant harmonics and spectral noise. Vibrato imparted by a delay line causes similar pitch modulation and spectral affects as natural vibrato, although it traditionally lacks amplitude modulation. Delay-line vibrato is typically implemented using basic sinusoidal modulation as an audio effect, but could be extended to create more naturalistic vibrato with more complex delay functions.

The remaining chapters of Part II explore the problems of vibrato transfer and vibrato suppression. Both sinusoidal modeling and a delay line will be used, in conjunction with new methods for modifying AM on an existing signal. Ultimately, the goal is to match vibrato patterns of multiple signals so that they are more perceptually blended following the theory of Common Fate as applied to psychoacoustics.

Chapter 4 contains material adapted from the following research papers: *On Vibrato and Frequency (De)Modulation in Musical Sounds*, published in the Proceedings of the International Conference on Digital Audio Effects (DAFx 2024), *Real-time implementation of vibrato transfer as an audio effect*, published in the Proceedings of the International Computer Music Conference (ICMC 2025), and *Vibrato Suppression by Time-Varying Delay and Spectral Magnitude Demodulation*, published in the International Symposium On Musical and Room Acoustics (ISMRA 2025), respectively. The dissertation author was the primary investigator and author of these publications, and Dr. Tamara Smyth was coauthor on the publications accepted to DAFx 2024 and ISMRA 2025.

Chapter 5

Methods for vibrato transfer

5.1 Introduction

This chapter focuses on *vibrato transfer*, in which vibrato patterns from one signal (the source) are imparted onto another (the target). The target signal is assumed to have no vibrato of its own, which would otherwise complicate the transfer (this problem will be addressed in Chapter 6). A method for vibrato transfer using a delay-line, as well as two implementations, is developed in Section 5.2. While this method exclusively handles FM caused by vibrato, a method for AM transfer is presented in Section 5.3. Several examples and potential applications are presented, including applications in sound design and perceptually-inspired sound blending.

5.2 Vibrato Transfer: methods for FM

5.2.1 Deriving a delay function from the instantaneous fundamental frequency

Here, let us return to the theoretical example of a sinusoid $z(n)$ that has been modulated using a time-varying delay line with delay function $D_m(n)$ to create the phase modulated sinusoid $z_m(n)$ (see Chapter 4, equations (4.4)–(4.8)). As with the phase-frequency relationship in (4.5), the instantaneous frequency of $z_m(n)$ modulated by delay function $D_m(n)$ is the derivative of the phase (4.8) with respect to time (or nT for the discrete-time signal here) and is given by

$$\begin{aligned}\omega_i(n) = \frac{d}{dnT}\theta_m(n) &= \omega_c - I\omega_m \cos(\omega_m nT)) \\ &= \omega_c \left(1 - \dot{D}_m(n)\right)\end{aligned} \tag{5.1}$$

where the derivative of the delay function with respect to n

$$\dot{D}_m(n) = \frac{d}{dn} D_m(n) = I\omega_m/\omega_c \cos(\omega_m nT) \tag{5.2}$$

is the *relative frequency shift* (RFS) imparted by the delay function $D_m(n)$. From (5.1), it may be easily seen that the transposition factor (or momentary transposition [52]), i.e. that value which, when multiplying an input frequency ω_c, results in a transposition of sounding frequency $\omega_i(n)$, is given by

$$\frac{\omega_i(n)}{\omega_c} = 1 - \dot{D}_m(n) \tag{5.3}$$

and thus the RFS may be expressed as

$$\dot{D}_m(n) = 1 - \frac{\omega_i(n)}{\omega_c}, \tag{5.4}$$

showing (5.4) may be estimated given frequency $\omega_i(n)$ and carrier frequency ω_c. It follows from (5.2) that the delay function is obtained by integrating the RFS, an operation that may be approximated by the discrete-time cumulative sum of (5.4)

$$\hat{D}_m(n) \approx \sum_{l=0}^{n} \dot{D}_m(l) \approx n - \frac{\theta_m(n)}{\omega_c T}. \tag{5.5}$$

Obtaining the delay function is, then, a matter of first estimating either the instantaneous phase $\theta_m(n)$ or frequency $\omega_i(n)$. In the following section, two commonly-used such techniques, one that estimates frequency and one that estimates phase, are briefly described and evaluated for their accuracy in examples estimating vibrato delay functions for 1) a contrived broadband signal (sawtooth wave) with known vibrato, 2) a musical signal (clarinet) with known vibrato (delay-line modulation with $D_m(n)$) and finally 3) for musical sounds with unknown vibrato.

5.2.2 Estimating instantaneous frequency in real signals

In this section two methods often used for obtaining $\omega_i(n)$ and/or $\theta_m(n)$ are explored and evaluated. The first method uses the STFT followed by momentary peak tracking to obtain $\omega_i(n)$ and the second uses a bandpass filter followed by the Hilbert Transform to obtain the analytic signal with angle $\theta_i(n)$, from which $\omega_i(n)$ can be derived.

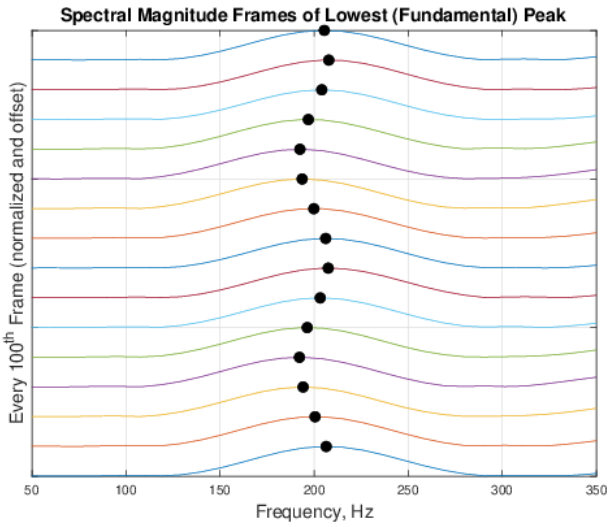

Figure 5.1. First "fundamental" spectral magnitude peak of a sawtooth wave with a sinusoidal vibrato applied. Every 100^{th} frame is offset to show time evolution (from bottom to top) of the peak position as it follows a sinusoidal vibrato track.

In the first method, a signal with vibrato $y(n)$ of length-N is analyzed using the STFT to obtain the $N \times N_f$ matrix of N_f spectral magnitude frames

$$\mathbf{Y} = \begin{bmatrix} |Y_0(\omega_0)| & |Y_1(\omega_0)| & |Y_2(\omega_0)| & \dots & |Y_{N_f-1}(\omega_0)| \\ |Y_0(\omega_1)| & |Y_1(\omega_1)| & |Y_2(\omega_1)| & \dots & |Y_{N_f-1}(\omega_1)| \\ |Y_0(\omega_2)| & |Y_1(\omega_2)| & |Y_2(\omega_2)| & \dots & |Y_{N_f-1}(\omega_2)| \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ |Y_0(\omega_N)| & |Y_1(\omega_N)| & |Y_2(\omega_N)| & \dots & |Y_{N_f-1}(\omega_N)| \end{bmatrix} \tag{5.6}$$

where the m^{th} column, $m = 0, 1, 2, \dots, N_f - 1$, is the magnitude of the DFT

$$|Y_m(\omega_k)| = \left| \sum_{n=0}^{N-1} x(n) w(n - mN_h) e^{-j2\pi kn/N} \right| \tag{5.7}$$

at time starting from sample mN_h for hopsize N_h and window $w(n)$. Because both time and frequency resolution are needed for estimating the vibrato signal, the hop size N_h is kept small.

From each momentary spectral magnitude $|Y_m(\omega_k)|$ frame, the "fundamental" frequency,

defined here as the lowest frequency with a significant peak amplitude, is tracked from frame to frame to create the frequency vector $\big[f_p(0), f_p(1), \ldots, f_p(N_f - 1)\big]$. To find the true peak, a quadratic interpolation is used. Figure 5.1 depicts the process of fundamental frequency tracking across time using the peak-picking method.

The vector f_p is resampled at a rate equal to N_h to yield a length-N vector $f_i^p(n)$[1] where

$$f_i^p(mN_h) = f_p(m), \tag{5.8}$$

along with interpolated frequency values at intermediate indeces. The estimated instantaneous frequency $\omega_i(n)$ is given by multiplying (5.8) by 2π. The center frequency ω_c is chosen as the mean of $\omega_i(n)$, and the estimated modulating delay function $\hat{D}_m(n)$ is derived using Equations (5.4)–(5.5). Figure 5.2 shows the instantaneous frequency and delay function of a vocal signal derived using the peak-picking method.

The second method proposed here uses a bandpass filter to isolate the first harmonic, from which the instantaneous frequency can be determined directly. This method is inspired by the bandpass analysis used in the Vibrato Analysis Toolbox [48]. It is required to have an external estimate of the signal's f_0 for this method; f_0 can be derived from a $f_i^p(n)$ defined as above, or using an external pitch estimation algorithm. In the case where $f_i^p(n)$ is extracted from the signal, f_0 can be set as the mean of $f_i^p(n)$ or as

$$f_0 = \frac{1}{N}\sum_{n=0}^{N-1} f_i^p(n) \approx \frac{\max\{f_i^p(n)\} + \min\{f_i^p(n)\}}{2}, \tag{5.9}$$

whichever is preferable for the signal. ω_c is created by multiplying f_0 by 2π.

A sixth order Butterworth bandpass filter is constructed centered at f_0. The Butterworth filter is chosen for its flat passband region, similar to the Vibrato Analysis Toolbox. The bandwidth can be set as a threshold (for example, 20% of the estimated f_0), or from quantiles of

[1]The superscript p denotes a $f_i(n)$ derived using spectral peak-picking.

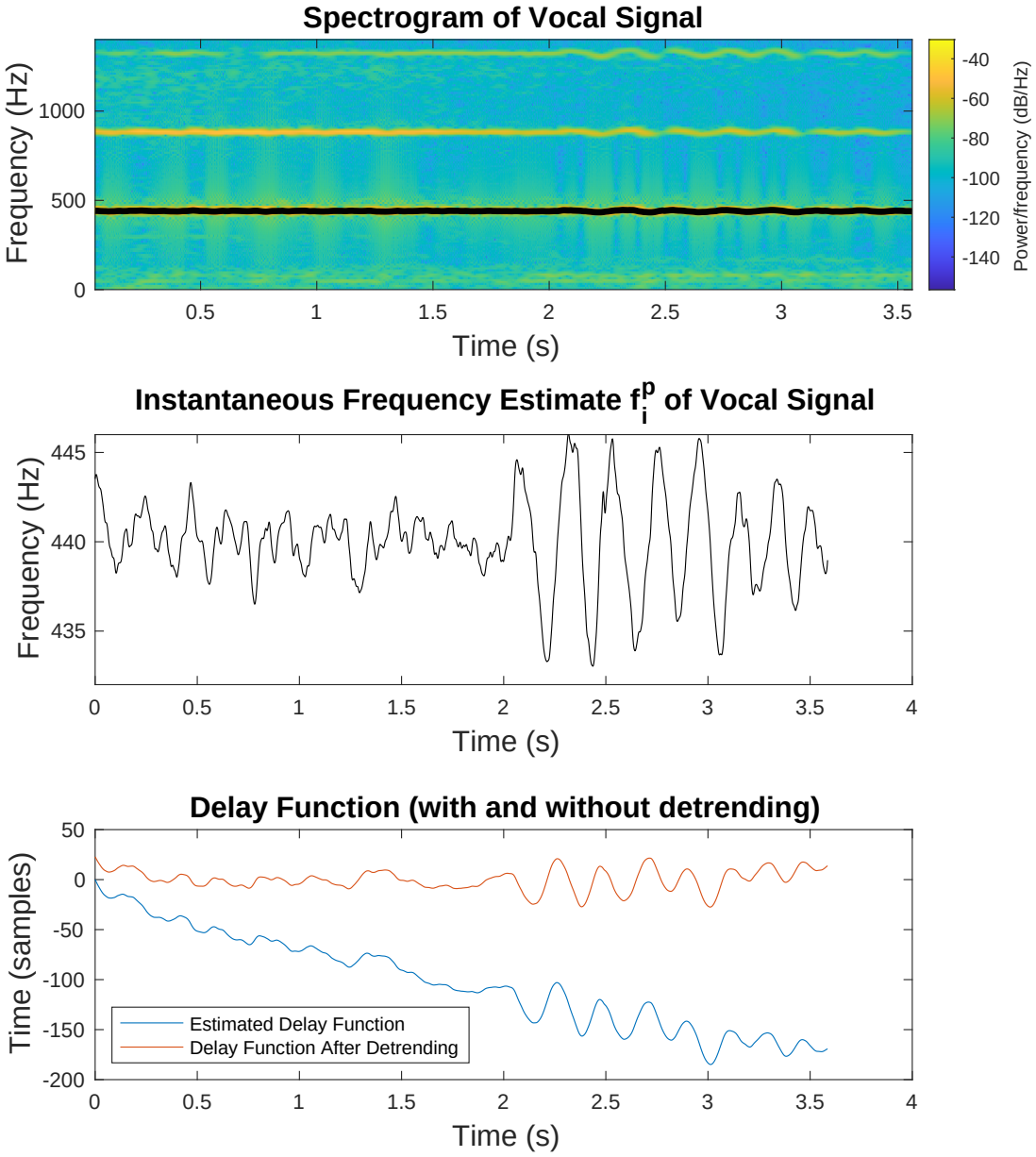

Figure 5.2. The estimated instantaneous frequency f_i^p and modulating delay function calculated from a vocal signal with vibrato. The delay function is detrended due to its linear growth.

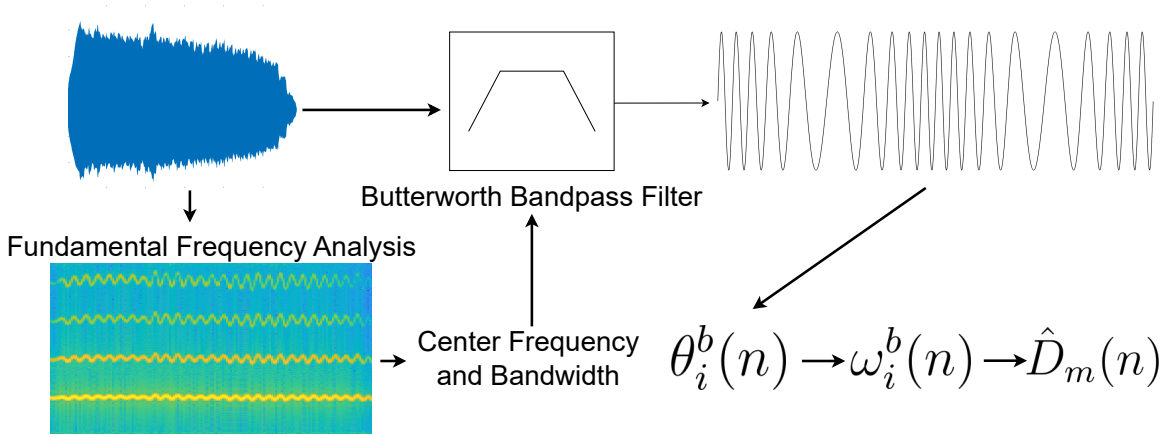

Figure 5.3. A signal is processed using a bandpass filter to isolate the first harmonic and use its analytic signal to create the estimated delay function. A frequency domain analysis is depicted to determine the signal's f_0, but time domain methods could be applied instead.

the f_i^p signal. The goal is to extend the lower and upper band edges of the filter to ensure that all FM sidebands pass with minimal error while not overlapping with the second harmonic (and its possible sidebands) or introducing any other unwanted signal noise.

If the modulated signal is passed through bandpass filter H_b, it is assumed the output is nearly sinusoidal, having only the first "fundamental" harmonic along with any sidebands from the vibrato's low-frequency modulation. For this reason, the angle of the complex analytic signal of the filter output yields the instantaneous phase $\theta_m^b(n)$ (with the b superscript indicating the bandpass method) and instantaneous frequency by its derivative, approximated by the finite difference

$$\omega_i^b(n) = \frac{\theta_m^b(n+1) - \theta_m^b(n)}{T}, \quad \text{and} \quad f_i^b(n) = \frac{\omega_i^b(n)}{2\pi}. \tag{5.10}$$

This analysis pipeline is depicted in Figure 5.3, and Figure 5.4 shows the instantaneous frequency and delay function of a guitar signal derived using the bandpass filter method.

Once the instantaneous frequency $\omega_i^p(n)$ or phase $\theta_m^b(n)$ is estimated, the corresponding delay function may be constructed by (5.4) and/or (5.5). Figure 5.5 shows both the f_i^p and f_i^b for a sawtooth and clarinet with known instantaneous frequencies. Both signals show good correspondence with the ground truth, with the exception of the onset of the bandpass filter signal,

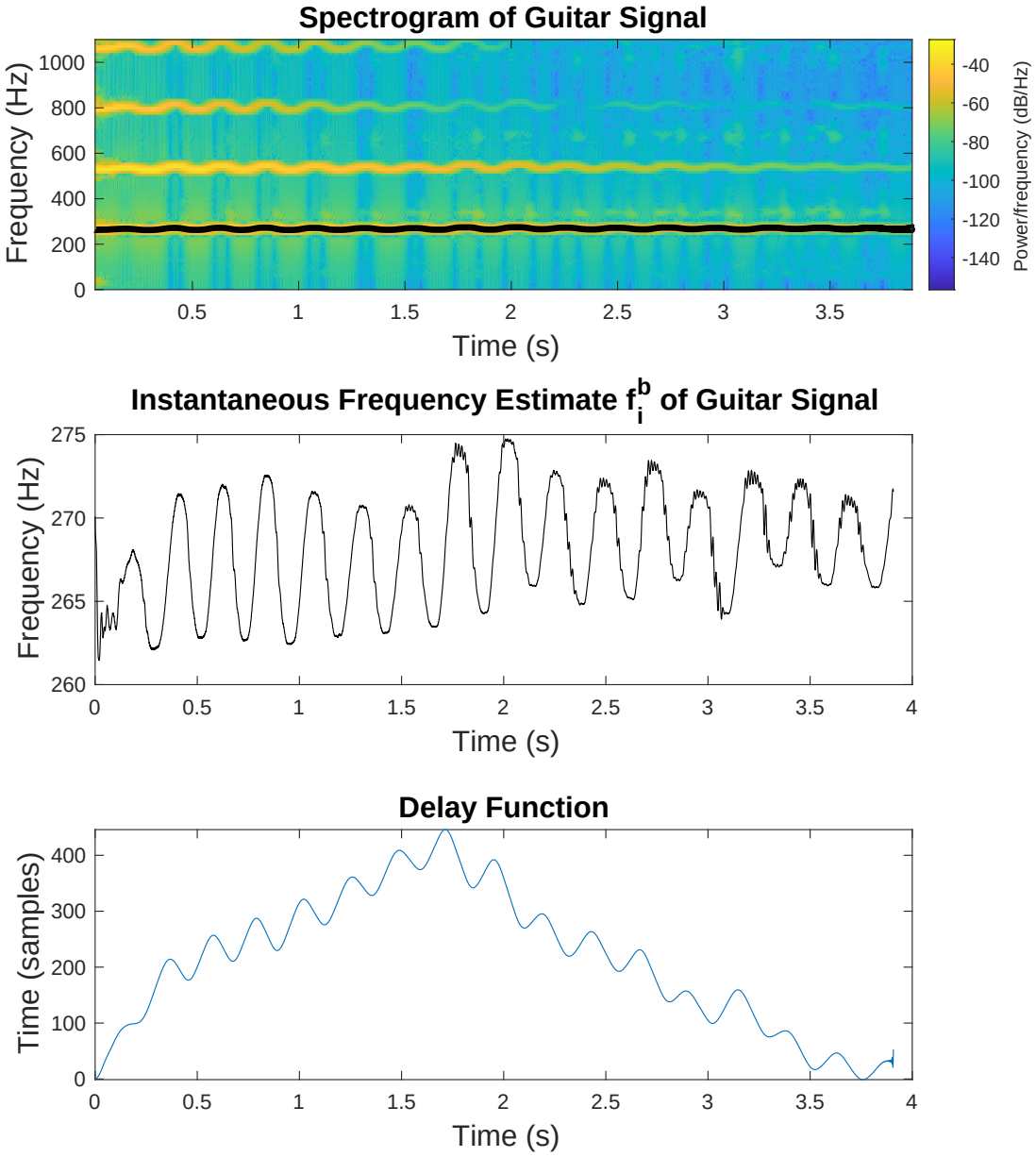

Figure 5.4. The estimated instantaneous frequency f_i^b and modulating delay function calculated from a guitar signal with vibrato.

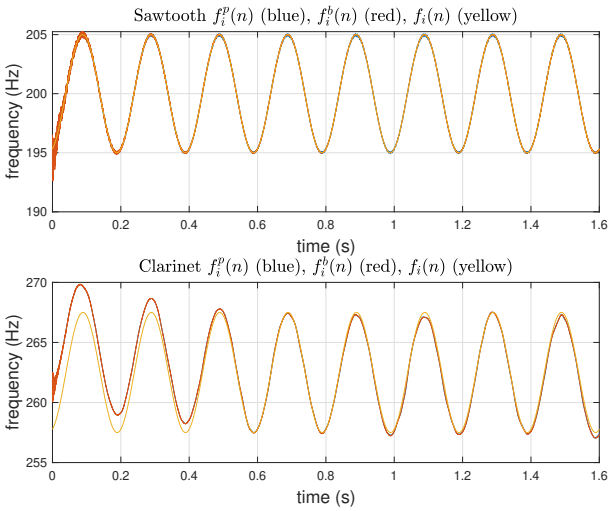

Figure 5.5. Estimated instantaneous frequency for sawtooth (top) and clarinet (bottom) using peak picking and bandpass filter methods. Each is modulated with parameters $f_m = 5$ and $I = 1$ and compared to known frequency modulator $f_i(n) = I f_m \cos(\omega_m nT)$. For the sawtooth $f_0 = 200$, but for the clarinet, f_0 is time varying (with a slight downward trend).

which takes time to stabilize due to being an IIR filter. Both peak-picking and bandpass methods show good agreement across multiple examples. The benefit of the bandpass filter method is that when combined with a real-time f_0 estimation algorithm, the instantaneous frequency (and derived modulating delay function) can be estimated in real-time at a very low computational cost [53]. However, the bandpass filter method can become unstable in signals with a very high noise floor due to a low signal-to-noise ratio (SNR) in the output of the bandpass filter. The peak-picking method is more robust to signal noise at the expense of a more computationally expensive analysis.

It should be noted that in real instrument sounds, f_0 may not be constant throughout the note but rather time varying with a (non-oscillating) upward or downward trend that is distinct from the vibrato (see clarinet Figure 5.5, bottom). In the case of vibrato transfer, where the derived delay function is used to modulate an external signal, it is sufficient to keep f_0 fixed. However, in Chapter 6 it will be necessary to account for a time-varying fundamental frequency.

Even in the presence of a fixed f_0, delay functions corresponding to certain pitch shifts are not practical for use in real delay settings. Consider a constant transposition of 1/2, for

example. Using (5.3) and (5.5) yields a delay function

$$\sum_{l=0}^{n}\left(1-\frac{1}{2}\right)=\sum_{l=0}^{n}\frac{1}{2}=\frac{1}{2}n,$$

having an upward linear trend with increasing sample index n. In an example more relevant to vibrato, consider a pitch shift that bends further downward from the center pitch than above. The RFS signal of this pitch shift will likely contain a DC component, resulting in a linear growth when integrated to create the delay function. To solve this issue in offline vibrato transfer, the derived modulating delay functions can be detrended using a linear or polynomial detrending function. In real-time vibrato transfer[2], delay drift can be mitigated using a bandpass filter on the delay function [53] at the expense of a slightly less accurate delay function.

5.3 Vibrato Transfer: incorporating AM

While the delay-line methods in Section 5.2 are effective at transforming the pitch modulation caused by vibrato, a more convincing vibrato transfer requires AM to be transferred as well. Because it is known that AM caused by vibrato can differ per-harmonic, it follows that transferring the AM of each harmonic should be used for a perceptually convincing result. Vibrato can also cause AM in the non-harmonic portions of the signal, and this AM can also differ across the frequency range. Therefore, modulating the spectral envelope of the non-harmonic component of a signal could improve vibrato transfer.

Consider a source signal $x(n)$ with vibrato, and the target signal $t(n)$ that will receive the vibrato from $x(n)$. The harmonics of $x(n)$ can be isolated using a bank of Butterworth bandpass filters centered at integer multiples of x's fundamental frequency (analogous to the filter bank used in the Vibrato Analysis Toolbox [48]). The residual noise is isolated by cascading x through a series of complementary Butterworth bandstop filters whose center frequencies are

[2]Real-time estimation of vibrato transfer is outside of the scope of this dissertation, but more information can be found in the cited report by the author.

also centered at integer multiples of the fundamental frequency; this approach is analogous to the filtering used in Part I to isolate partials and the residual signal in frequency bashing - see Equation (3.2). In the following algorithm, the harmonic signals and residual signal are used to determine AM patterns that will transfer to the harmonics and residual of $t(n)$, which are isolated using a similar set of filters centered at integer multiples of $t(n)$'s fundamental frequency.

The AM pattern of each harmonic is determined using its envelope and a low-passed version of the envelope (i.e. the envelope without vibrato). This approach is inspired by the low- and high-passed amplitude tracks used in sinusoidal resynthesis in Equation (4.2). The kth harmonic in $x(n)$ captured using the kth bandpass filter as

$$h_k(n) = x \circledast BP_k(n). \tag{5.11}$$

The envelope $E_{hk}(n)$ is found using the root-mean-square (RMS) of $h_k(n)$ with a 1024-sample window and 75% window overlap. Its low-passed counterpart $\overline{E}_{hk}(n)$ is obtained by processing $E_{hk}(n)$ with a high-order FIR lowpass filter and using forward-backward filtering to maintain the original phase of the envelope, exactly as done in the resynthesis method [50]. The AM transfer envelope of the kth harmonic is then computed as

$$AM_k(n) = \frac{E_{hk}(n)}{\overline{E}_{hk}(n)}. \tag{5.12}$$

The signal flow for this process is depicted in Figure 5.6.

Similar to using sinusoidal resynthesis, it is necessary to choose the number of harmonics to capture using bandpass filters. The remaining harmonics, as well as the non-harmonic components of the input, will be contained in the residual

$$x_r(n) = x \circledast (BS_1 \circledast BS_2 \circledast \ldots \circledast BS_k). \tag{5.13}$$

Again taking inspiration from the resynthesis approach, the spectral envelope of the residual is

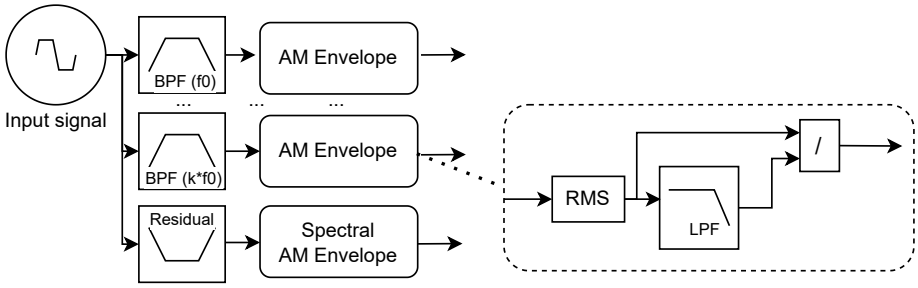

Figure 5.6. The signal flow for calculating the AM envelope for both harmonics and the spectral envelope of the non-harmonic portions of a signal. The envelope of each harmonic is calculated as a ratio of the RMS of the harmonic and a low-passed version of the RMS signal. Creating the spectral AM envelope is shown in Figure 5.7

.

modeled using a fixed number of points d and then creating a frequency-dependent AM envelope. First, the STFT of $x_r(n)$ is computed to obtain $X_R(\omega, m)$. The frequency range of X_R is divided into d non-overlapping regions, and for each frame at time m the frequency with maximum magnitude in each region

$$\omega_d^*(m) = \underset{\omega * \in [\omega_d, \omega_{d+1})}{\arg\max} |X_R(\omega^*, m)| \tag{5.14}$$

is used to create the spectral envelope

$$X_{RS}(d, m) = 20 \log_{10} \Big(|X_R(\omega_d^*(m), m| \Big). \tag{5.15}$$

As before, a lowpass filter is applied along the time axes to obtain the spectral envelope $\overline{X}_{RS}(d, m)$. Then both X_{RS} and $\overline{X}_{RS}$ are linearly interpolated back to the dimensions of X_R and converted back to linear amplitude to create $X_{RS}^*(\omega, m)$ and $\overline{X}_{RS}^*(\omega, m)$. Finally, the AM envelope is constructed as

$$AM_R(\omega, m) = \frac{X_{RS}^*(\omega, m)}{\overline{X}_{RS}^*(\omega, m)}. \tag{5.16}$$

An example of the spectral envelope for one frame, the time-varying envelope for one spectral point, and the AM envelope for that point are shown in Fig. 5.7.

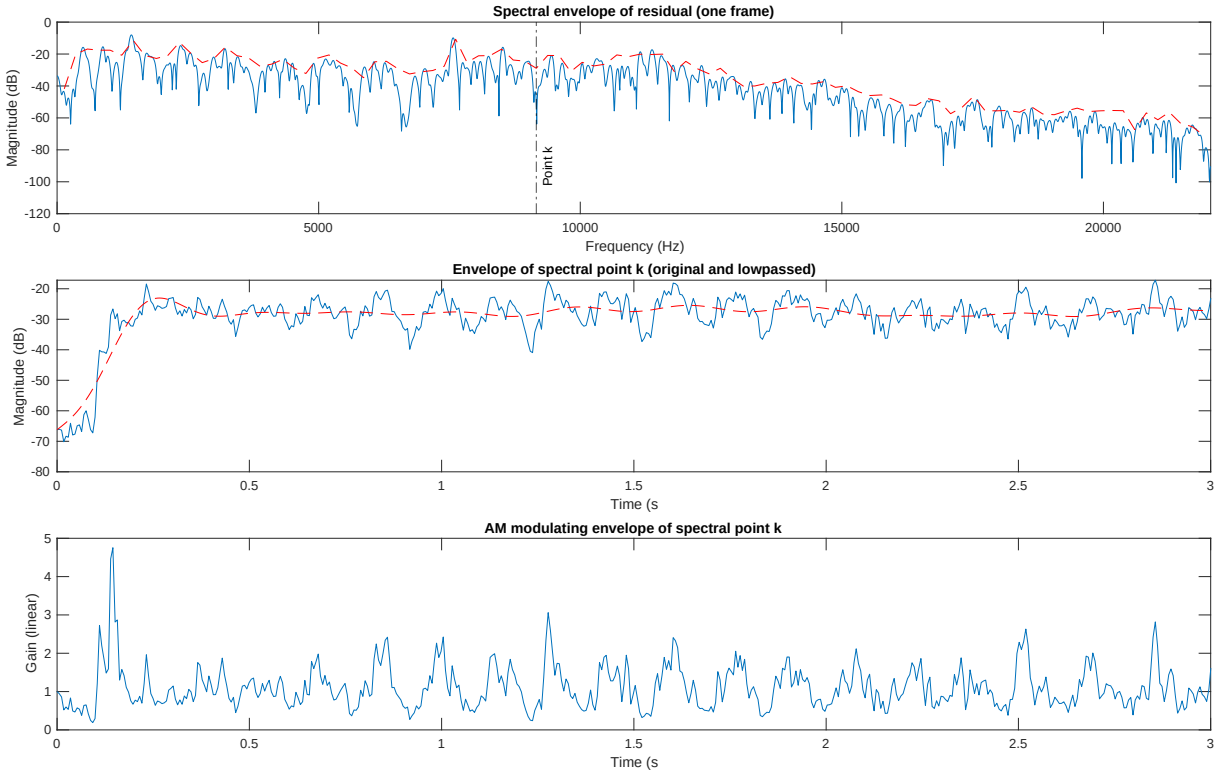

Figure 5.7. Spectral envelope modulation involves down-sampling in the frequency domain to d distinct points, tracking the amplitude over time and creating a modulating envelope of each point. The spectral envelope (top) is estimated using the surfboard method. The original and low-passed amplitude track of a single point is shown over time (middle). The modulating envelope of the analyzed point (bottom) is created using the ratio of the original amplitude track and its low-passed version.

5.4 The vibrato transfer algorithm and sound blending

The transfer of vibrato from $x(n)$ to $t(n)$ proceeds as follows. First, the harmonics and residual signals of each signal are isolated. The kth harmonic of $t(n)$ are multiplied in the time-domain by the corresponding AM envelope $AM_k(n)$ from (5.12). If performing residual AM transfer, the STFT of the residual signal $t_s(n)$ is multiplied by the spectral AM envelope $AM_{RS}(\omega, m)$ from (5.16). Then, the harmonic signals are summed and FM is transferred using an estimated modulating delay $\hat{D}_m(n)$ as derived in Section 5.2. The residual signal is excluded from delay-line modulation so that broadband noise is not modulated, which can make the delay function audible. Finally, the signal of harmonics, now with the FM and AM from $x(n)$, are mixed with the residual signal to a create the final vibrato transferred signal.

Figures 5.8–5.10 show vibrato transfer for three instruments (flute, vocals, and saxophone, respectively). In each figure, both AM transfer with and without treating the residual noise is shown, and FM transfer is performed using a $D_m(n)$ derived using peak-picking. The effectiveness of treating residual noise largely depends on the signal. The flute signal in Figure 5.8 benefits the most, particularly in noise above 10 kHz, whereas the vocal and saxophone signals in Figures 5.9 and 5.10 do not substantially change when the residual noise is left unmodulated.

The transfers in Figures 5.8–5.10 were created using recordings of an instrumentalist playing the same note with and without vibrato. Turning back to the motivating application of sound blending, it is instructive to compare sound mixtures of these notes before and after transferring patterns to the previously unmodulated note. While these mixes are not formally evaluated in a listening test, it is clear that the mixing of the signals post-vibrato transfer results in a substantial increase in perceptual blending.

Figures 5.9–5.13 show mixtures of the flute, vocal, and saxophone signals from Figures 5.8–5.10. In the top portion of each figure, a mixture of the signal with and without vibrato is shown. On the bottom, the mixture is shown after modulation from the signal with vibrato was transferred onto the unmodulated signal. The spectrograms show that the mixes pre-vibrato

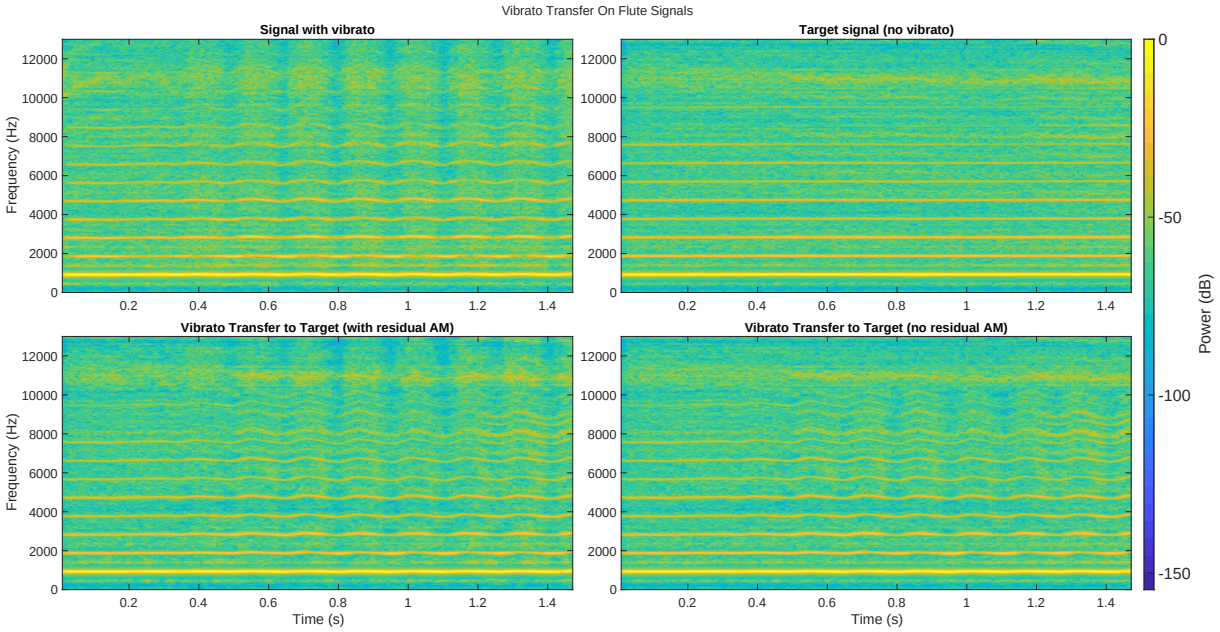

Figure 5.8. Vibrato transfer between flute signals. The vibrato from a flute signal (top left) is transferred to a signal without vibrato (top right). Incorporating AM to the residual noise (bottom left) creates a more accurate transfer than only transferring AM to the harmonics of the signal (bottom right). This is particularly visible in the AM in the upper (> 10 kHz) frequencies.

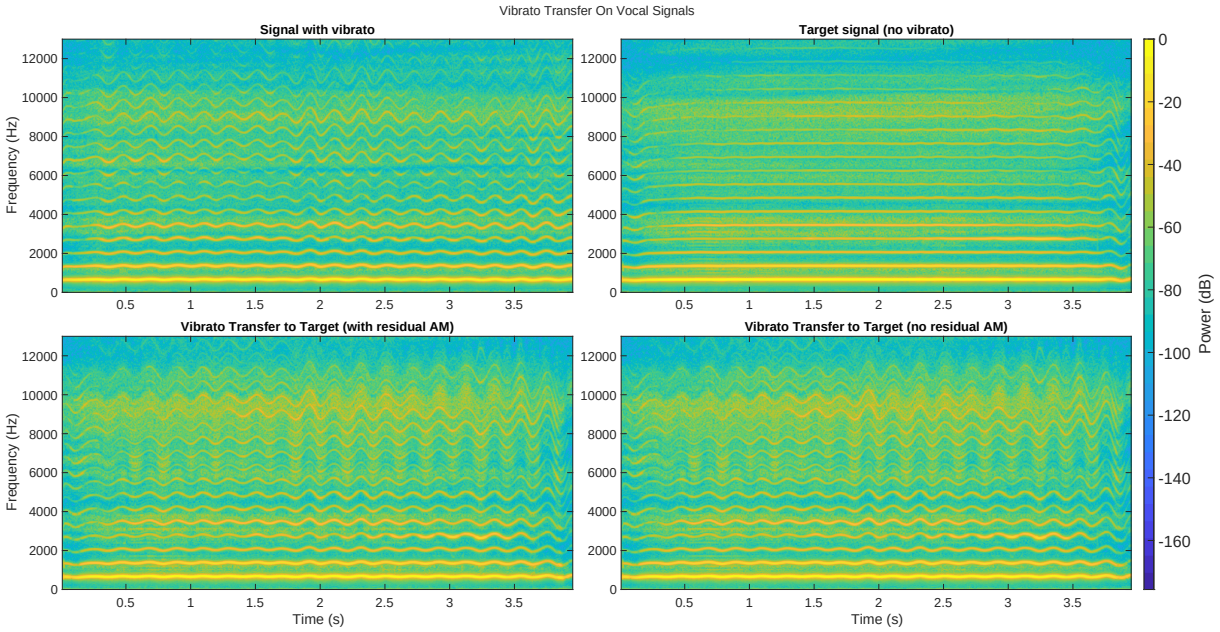

Figure 5.9. Vibrato transfer between vocal signals.

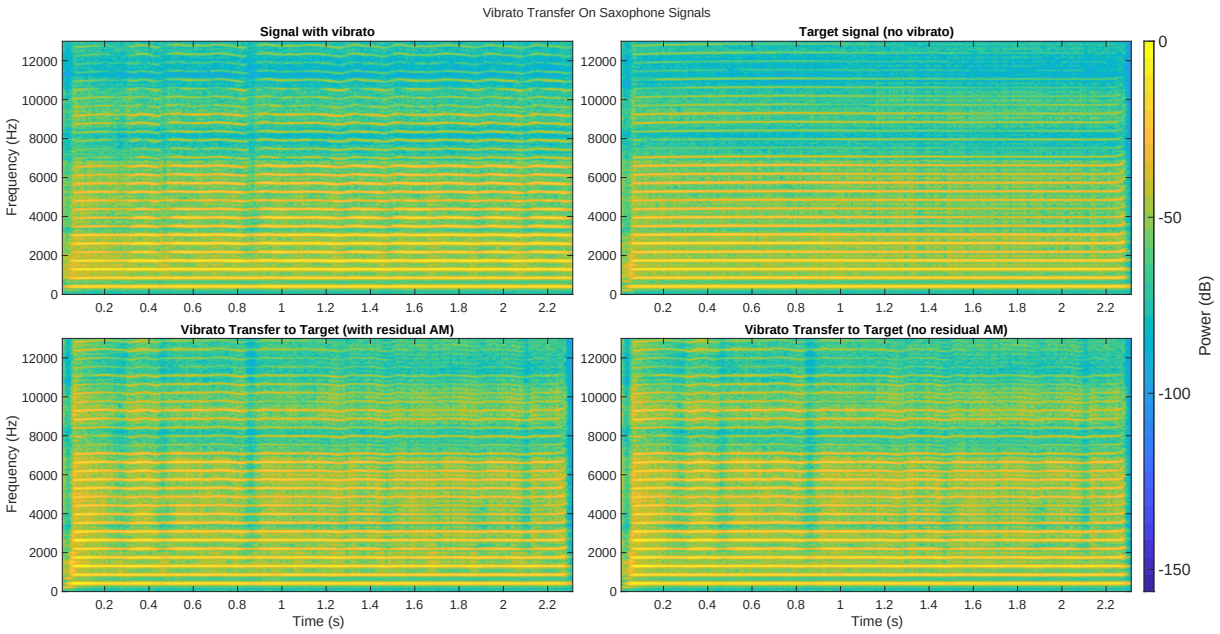

Figure 5.10. Vibrato transfer between saxophone signals.

transfer can be identified as a mixture of two signals, as the harmonics of the vibrato and static signals are both clearly visible and produce audible beating (this is particularly true in Figure 5.9). Conversely, the mixture of signals after vibrato transfer overlay substantially and appear spectrally as one signal.

The audio examples of each mix are available on the supplemental website. Subjectively, mixes post-vibrato transfer audibly resemble double-tracked signals or signals processed with a double or chorusing audio effect, while the pre-vibrato transfer mixes are less blended. The main component where signals remain perceptually distinct is in the attack region, suggesting that attack modification could be used for further perceptual blending. Nevertheless, vibrato transfer seems to be a promising effect for blending unison signals and may additionally have use cases as a multitrack processing tool in audio engineering.

5.5 Conclusion

Vibrato transfer has multiple applications in music making and sound design. The vibrato audio effect, a popular tool for guitar and synthesizer, can be supplemented with real-time control

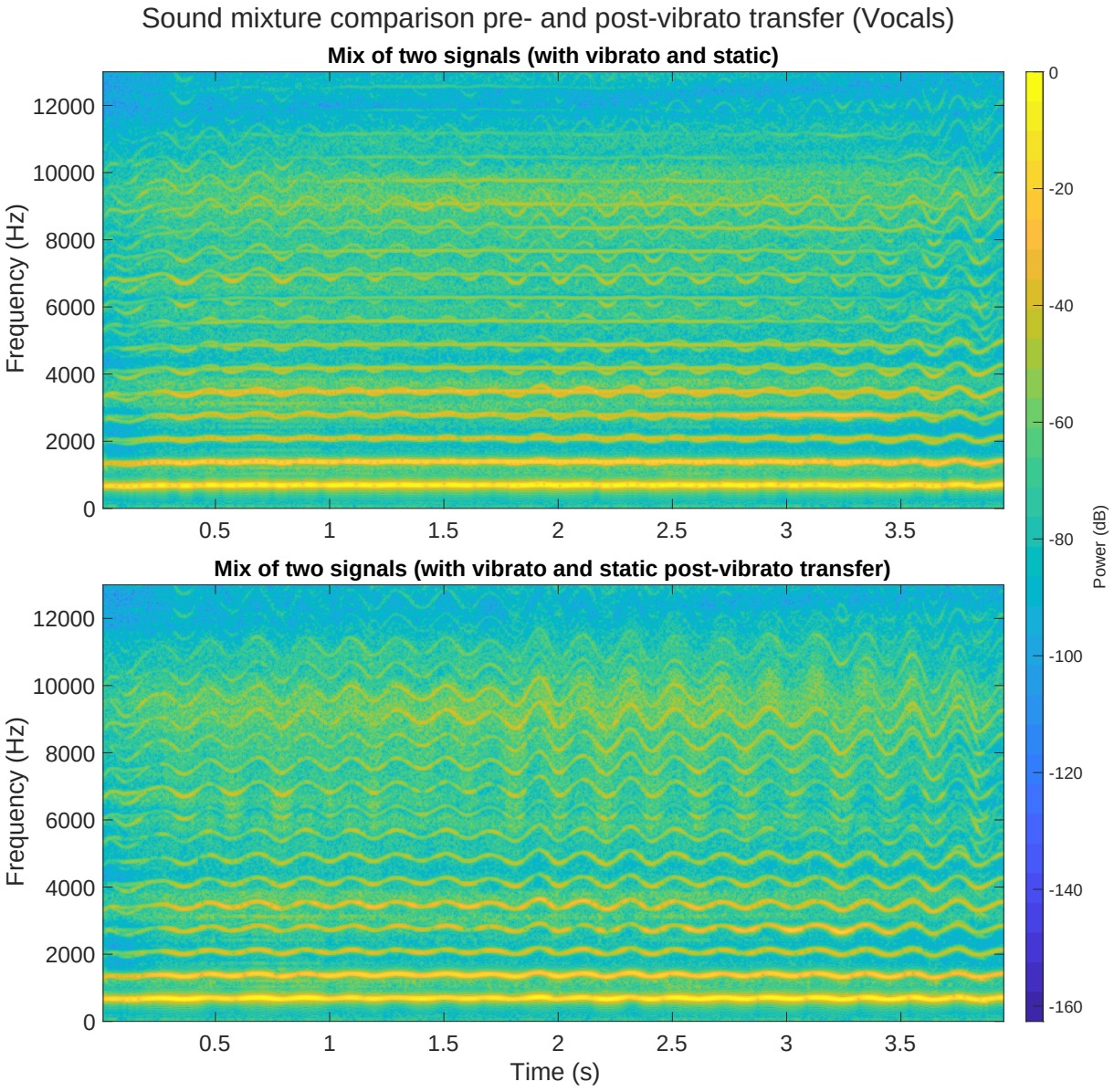

Figure 5.11. Mixing signals with and without vibrato. The top plot shows a mix of two signals: a vocalist singing a note with vibrato and singing the same note with as little wavering as possible. The bottom shows the same mix, but after vibrato patterns from the first signal have been imparted onto the static note using vibrato transfer. Outside of the attack region, the mixture on the bottom looks remarkably like a single signal (compare to the top left plot in Figure 5.9).

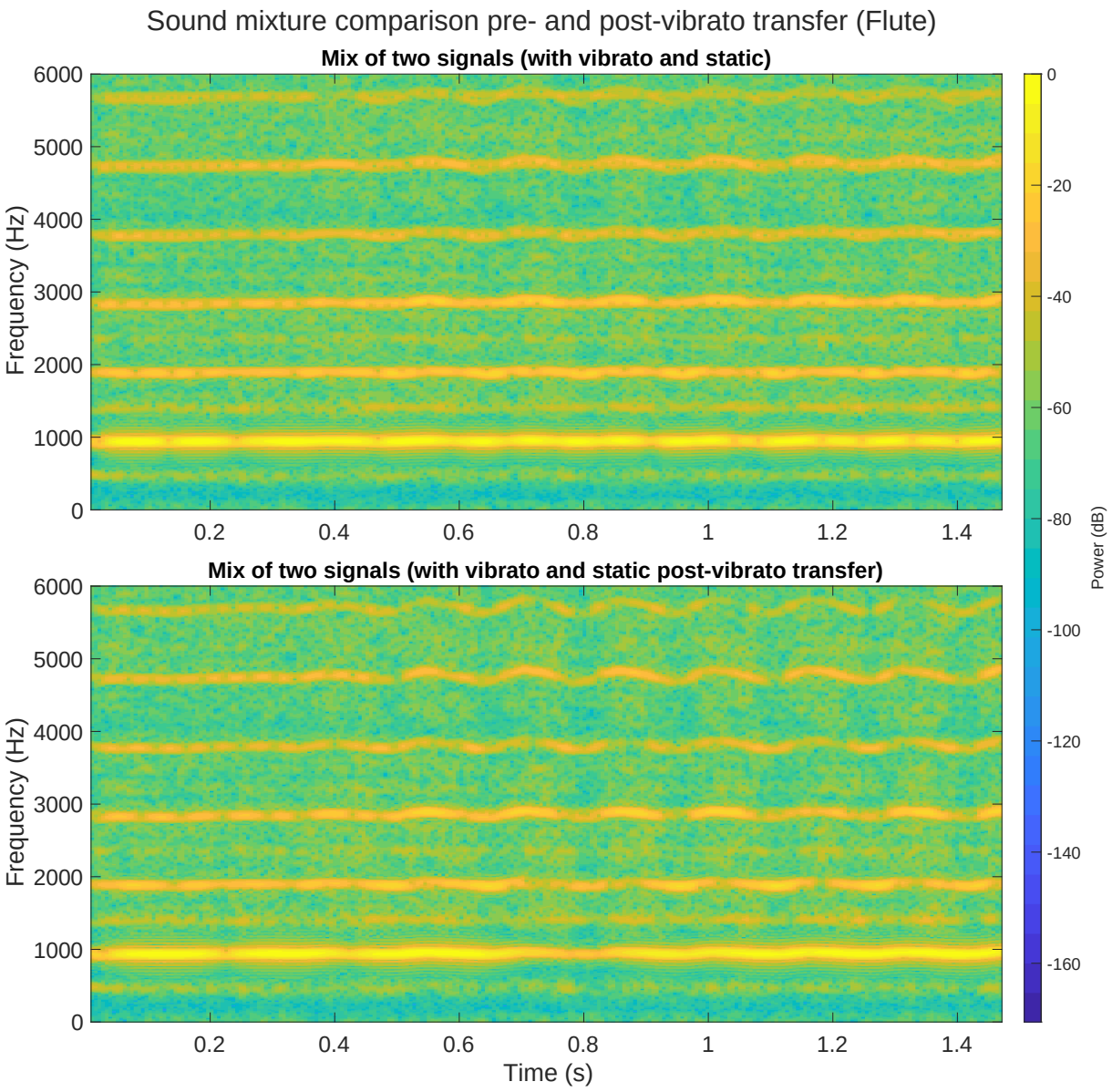

Figure 5.12. The top plot shows a mix of two signals: a flutist playing a note with vibrato and playing the same note without vibrato. The bottom shows the same mix after vibrato patterns from the first signal have been imparted onto the static note using vibrato transfer.

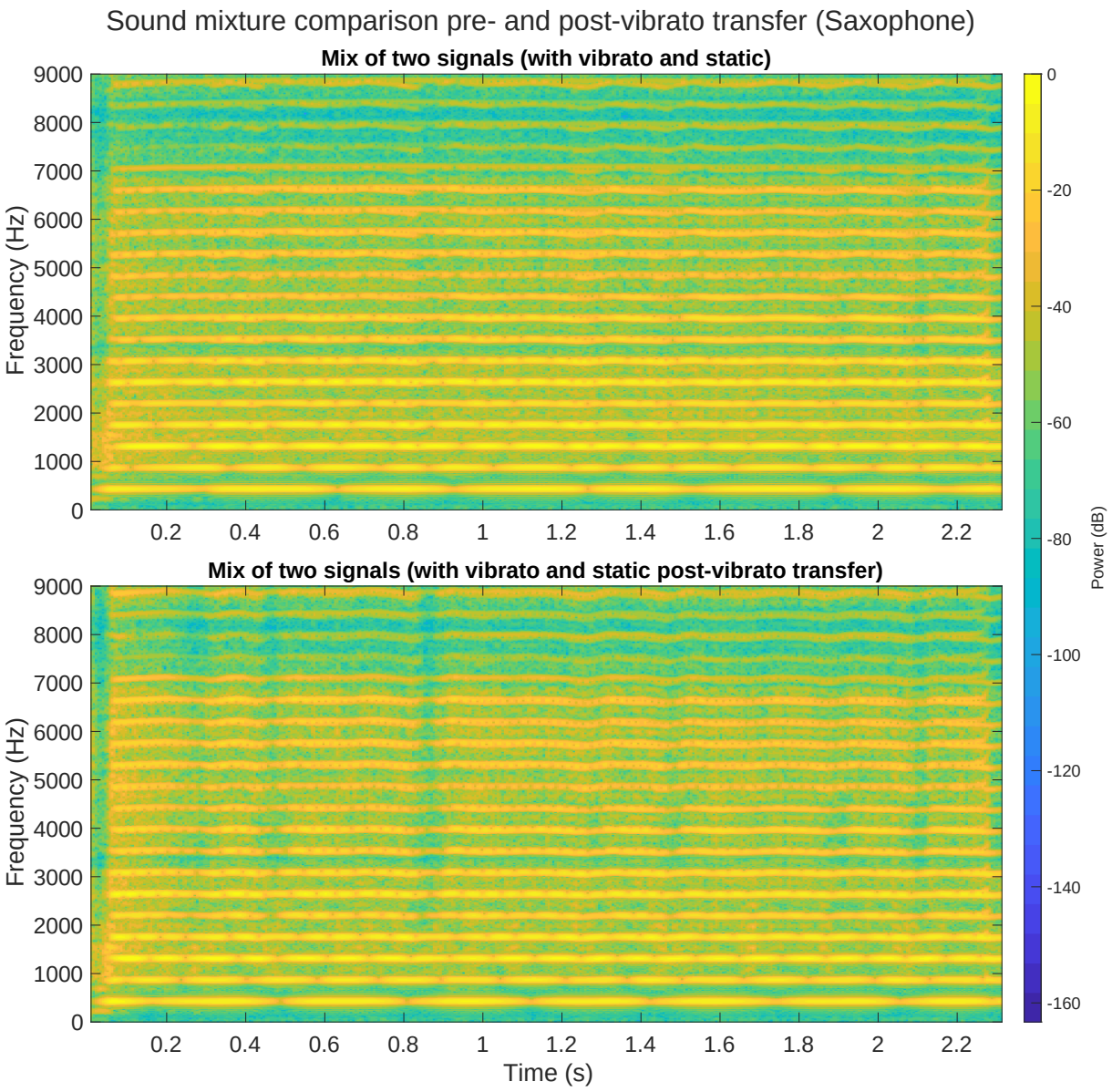

Figure 5.13. The top plot shows a mix of two signals: a saxophonist playing a note with vibrato and playing the same note without vibrato. The bottom shows the same mix after vibrato patterns from the first signal have been imparted onto the static note using vibrato transfer.

from an external signal to create a more expressive and realistic vibrato. The incorporation of amplitude modulation supplements the typical delay-line implementation to impart a vibrato closer in timbre to acoustic instruments. Transferring vibrato from recordings of acoustic instruments may be used to complement sounds synthesized using physical modeling, which may not account for vibrato. Finally, since distinct vibrato patterns are perceptually relevant to source separation [46], matching vibrato patterns of unison signals offers a unique approach to blending signals in sound design.

The FM patterns of a signal with real-vibrato can be transferred to a new signal using a delay line. AM patterns of vibrato are more complex and require a combination of filter banks and spectral processing to fully transfer. Applying amplitude modulation to each harmonic of a monophonic signal, and optionally modulating the spectral envelope of its residual noise, can result in a vibrato pattern that very closely matches a real signal with vibrato.

The vibrato transfer algorithms operate on the assumption that the signal receiving vibrato is otherwise unmodulated. If applying external vibrato to a signal with its own vibrato, the patterns of modulation will mix and interfere, resulting in a less (or not at all) convincing transfer. It may be necessary to first remove vibrato from a target signal before imparting new modulation onto it; this problem is addressed in the following chapter.

Chapter 5 contains material adapted from the following research papers: *On Vibrato and Frequency (De)Modulation in Musical Sounds*, published in the Proceedings of the International Conference on Digital Audio Effects (DAFx 2024), *Real-time implementation of vibrato transfer as an audio effect*, published in the Proceedings of the International Computer Music Conference (ICMC 2025), and *Vibrato Suppression by Time-Varying Delay and Spectral Magnitude Demodulation*, published in the International Symposium On Musical and Room Acoustics (ISMRA 2025), respectively. The dissertation author was the primary investigator and author of these publications, and Dr. Tamara Smyth was coauthor on the publications accepted to DAFx 2024 and ISMRA 2025.

Chapter 6

Methods for vibrato suppression

6.1 Introduction

As explored in Chapters 4 and 5, natural vibrato causes complex FM and AM patterns in a signal. Vibrato is often considered an expressive musical technique; however, at times, it may necessary to remove vibrato from a recorded signal. The process will henceforth be referred to as *vibrato suppression* [1]. In the context of sound blending by matching the vibrato patterns in signals, it is necessary to suppress vibrato from a signal before applying a new vibrato pattern. Other contexts in which vibrato might be suppressed include informatics tasks in which vibrato causes analysis errors or creative tasks where precise vibrato control of a previously recorded sound is desired by the composer. Finally, while not strictly related to vibrato in the traditional sense, frequency demodulation at vibrato rate could be used to de-warp audio obtained from phonographs [54] or other physical mediums.

This chapter focuses on algorithms for vibrato suppression. They are direct corollaries (and sometimes inverses) of vibrato transfer methods presented in Chapter 5, with some modifications. Methods for demodulating both FM and AM caused by vibrato are presented and tested against vibrato suppression using sinusoidal resynthesis. Lastly, sound blending a mixture of signals is explored, with several examples of sound mixtures presented where both vibrato suppression and vibrato transfer are used to obscure the presence of multiple sources.

6.2 Vibrato suppression: frequency demodulation

Let us return to the synthetic example of a sinusoid, $z(n)$, which has been modulated using a time-varying delay line (TVDL) and delay function $D_m(n)$ to create the modulated sinusoid $z_m(n)$[2]. Using $z_m(n)$ as the input to the TVDL with delay function

$$D_d(n) = -\frac{I}{\omega_c}\sin(\omega_m nT)f_s + \frac{I}{\omega_c}fs, \tag{6.1}$$

[1] As will be shown, suppression is preferable to *demodulation*, which suggests a removal of modulation patterns. Here, the goal is perceptual removal of modulation even if the signal is not fully demodulated.

[2] See Chapter 4, Equations(4.6)–(4.8).

the phase inverse of $D_m(n)$, the output of the delay line is given by

$$z_d(n) = z_m(n - D_d(n)) = e^{j\theta_m(n - D_d(n))}, \tag{6.2}$$

with instantaneous phase given by substituting instances of n in $\theta_m(n)$ with $n - D_d(n)$ to yield

$$\begin{aligned} \theta_d(n) &= \omega_c(n - D_d(n))T - \omega_c D_m(n - D_d(n))T \\ &= \omega_c nT + I\sin(\omega_m nT) - 2I - I\phi(n), \end{aligned} \tag{6.3}$$

where the final expression has the frequency modulated sinusoid

$$\phi(n) = \sin(\omega_m nT + I_1 \sin(\omega_m nT) - I_1), \tag{6.4}$$

for $I_1 = I\omega_m/\omega_c$. Because $I\phi(n) \neq I\sin(\omega_m nT)$, these two terms do not cancel in (6.3) and $z_d(n) \neq z(n)$, suggesting the delay function (6.1) does not completely demodulate the vibrato. It is worthwhile, however, to examine the extent to which $I\phi(n) \approx I\sin(\omega_m nT)$ and whether the vibrato is audibly present in $z_d(n)$.

A Fourier series expansion of (6.4) expresses it as a weighted sum of sinusoids

$$\phi(n) = \sum_{k=-\infty}^{\infty} \phi_k(n) = \sum_{k=-\infty}^{\infty} J_k(I_1)\sin((1+k)\omega_m nT - I_1), \tag{6.5}$$

where $J_k(I_1)$ is the kth-order Bessel function of the first kind indexed by I_1. Assuming for vibrato that $\omega_m << \omega_c$ and I is a low value when compared to typical settings in synthesis, then I_1 will be small and very few harmonics (sidebands) of ω_m will have significant amplitude. For example, the sideband of $\phi(n)$ at $k = -1$ given by

$$\phi_{-1}(n) = J_{-1}(I_1)\sin(-I_1) = J_1(I_1)\sin(I_1) \approx 0 \tag{6.6}$$

yields a DC component of negligible amplitude for small I_1. The same approximation for sideband $k = 1$ yields

$$\begin{aligned}\phi_1(n) &= J_1(I_1)(\cos(I_1)\sin(2\omega_m nT) - \sin(I_1)\cos(2\omega_m nT) \\ &\approx J_1(I_1)\cos(I_1)\sin(2\omega_m nT) \end{aligned} \tag{6.7}$$

and the sideband at $k = 0$ by

$$\begin{aligned}\phi_0(n) &= J_0(I_1)(\cos(I_1)\sin(\omega_m nT) - \sin(I_1)\cos(\omega_m nT)) \\ &\approx \sin(\omega_m nT) - J_0(I_1)\sin(I_1)\cos(\omega_m nT), \end{aligned} \tag{6.8}$$

where the unit-amplitude sinusoid arises since $J_0(I_1)\cos(I_1) \approx 1$ for small I_1. Adding (6.7) and (6.8) gives an approximation to (6.4) as the sum

$$\phi(n) \approx \sin(\omega_m nT) - \frac{I_2}{I}\cos(\omega_m nT) + \frac{I_3}{I}\sin(2\omega_m nT), \tag{6.9}$$

where

$$I_2 = IJ_0(I_1)\sin(I_1) \quad \text{and} \quad I_3 = IJ_1(I_1)\cos(I_1). \tag{6.10}$$

Substituting (6.9) into (6.3), yields

$$\theta_d(n) \approx \omega_c nT - 2I + I_2\cos(\omega_m nT) - I_3\sin(2\omega_m nT) \tag{6.11}$$

and a cancellation of the original modulating sinusoid in (6.3).

Finally, substituting (6.11) into (6.2) yields

$$z_d(n) \approx e^{j(\omega_c nT - 2I)}e^{jI_2\cos(\omega_m nT)}e^{-jI_3\sin(2\omega_m nT)} = \hat{z}_d(n), \tag{6.12}$$

showing that $\hat{z}_d(n)$ is an FM sinusoid with two low amplitude modulating sinusoids.

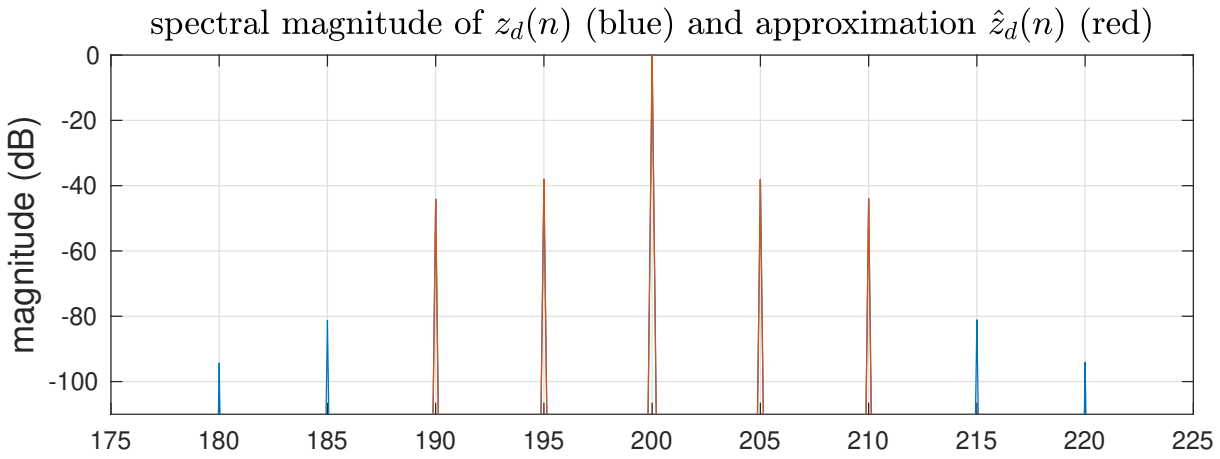

Figure 6.1. Spectral magnitude of the delay line output $z_d(n)$ (6.2) and its approximation $\hat{z}_d(n)$ (6.13) for FM parameters $I = 1$, $\omega_c = 2\pi 200$ and $\omega_m = 2\pi 5$. Strong agreement is visible for 2 sidebands above and below the carrier frequency greater than -60dB: for $k = \pm 1$ the amplitude is $J_1(I_2) = 0.0125 \approx -38$ dB and for $k = \pm 2$ the amplitude is $J_2(I_3) = 0.0062 \approx -44$ dB.

For small $1 \ll I_2, I_3$, approximations may be made using only sidebands $k = -1, 0, 1$ to yield

$$\begin{aligned} e^{jI_2\cos(\omega_m nT)} &= \sum_{k=-\infty}^{\infty} J_k(I_2) e^{jk(\omega_m nT + \pi/2)} \\ &\approx 1 + J_1(I_2) j \left(e^{j\omega_m nT} + e^{-j\omega_m nT}\right), \\ e^{-jI_3\sin(2\omega_m nT)} &= \sum_{k=-\infty}^{\infty} J_k(I_3) e^{-jk2\omega_m nT} \\ &\approx 1 - J_1(I_3) \left(e^{j2\omega_m nT} - e^{-j2\omega_m nT}\right), \end{aligned}$$

so that their product in (6.12) yields the final approximation of the delay-line output given by

$$\begin{aligned} \hat{z}_d(n) \approx \quad & e^{j(\omega_c nT - 2I)} + J_1(I_2) j e^{-j2I} \left(e^{j(\omega_c+\omega_m)nT} + e^{j(\omega_c-\omega_m)nT}\right) \\ & - J_1(I_3) e^{-j2I} \left(e^{j(\omega_c+2\omega_m)nT} - e^{j(\omega_c-2\omega_m)nT}\right), \end{aligned} \tag{6.13}$$

showing a unit-amplitude sinusoidal component at carrier frequency ω_c along with two upper and lower sidebands at frequencies $\omega_c \pm \omega_m$ and $\omega_c \pm 2\omega_m$ having amplitudes $J_1(I_2)$ and $J_1(I_3)$, respectively (see Figure 6.1). (Note that (6.13) omits a final combination term that contributes a negligible amplitude of $J_1(I_2)J_1(I_3)$ to the first and third upper and lower sidebands).

6.2.1 Further Vibrato Reduction

In many natural instruments, the frequency modulation induced by vibrato is mild enough that the first approximation demodulating delay $D_d(n)$ is enough to reduce sidebands below the threshold of audibility. However, some instruments (such as the human voice, as will be shown later) can produce vibrato with sidebands that are audible after demodulation using $D_d(n)$. To deal signals with high vibrato extent, we consider the *recursive* demodulating delay function

$$\begin{aligned} D_{dd}(n) &= D_d(n - D_d(n)) \\ &= -\frac{I}{\omega_c}\sin(\omega_m(n - D_d(n)T)f_s + \frac{I}{\omega_c}f_s \\ &= -\frac{I}{\omega_c}\phi(n)f_s + \frac{I}{\omega_c}f_s, \end{aligned} \tag{6.14}$$

in which (6.1) is delayed by itself. Passing $z_m(n)$ through a TVDL using $D_{dd}(n)$ as the delay function yields the output

$$z_{dd}(n) = z_m(n - D_{dd}(n)) = e^{j\theta_m(n - D_{dd}(n))}, \tag{6.15}$$

having instantaneous phase given by substituting n with $n - D_{dd}(n)$ in (4.8) to yield

$$\begin{aligned} \theta_{dd}(n) &= \omega_c nT - \omega_c D_{dd}(n)T - I\sin\left(\omega_m nT - \omega_m D_{dd}(n)T\right) - I \\ &= \omega_c nT - 2I + I\phi(n) - I\sin(\omega_m nT + I_1\phi(n) - I_1). \end{aligned} \tag{6.16}$$

Expressing the last term in (6.16) as the imaginary part of an analytic signal,

$$I\sin(\omega_m nT + I_1\phi(n) - I_1) = I\Im\left\{e^{j(\omega_m nT - I_1)}e^{jI_1\phi(n)}\right\}, \tag{6.17}$$

and substituting the approximation for $\phi(n)$ in (6.9), yields intermediate factor

$$e^{jI_1\phi(n)} \approx e^{jI_1 \sin(\omega_m nT)} e^{-jA_1 \cos(\omega_m nT)} e^{jA_2 \sin(2\omega_m nT)}, \tag{6.18}$$

showing again an FM sinusoid with very low-amplitude modulators. Since $A_1 = I_1 I_2 / I$ and $A_2 = I_1 I_3 / I$ are small, the approximations

$$\begin{aligned} e^{-jA_1 \cos(\omega_m nT)} &= j \sum_{k=-\infty}^{\infty} J_k(A_1) e^{jk\omega_m nT} \approx J_0(A_1) \approx 1, \\ e^{jA_2 \sin(\omega_m nT)} &= \sum_{k=-\infty}^{\infty} J_k(A_2) e^{jk2\omega_m nT} \approx J_0(A_2) \approx 1, \end{aligned} \tag{6.19}$$

may be made because the zeroth sideband ($k = 0$) is the only term in the infinite sum (6.19) having non-negligible amplitude (and it's amplitude is almost exactly equal to one). Substituting (6.19) into (6.18) yields the approximation

$$e^{jI_1\phi(n)} \approx e^{jI_1 \sin(\omega_m nT)} \tag{6.20}$$

and substituting (6.20) into the analytic signal (6.17) in turn yields

$$\begin{aligned} \sin(\omega_m nT + I_1\phi(n) - I_1) &\approx \Im \left\{ e^{j(\omega_m nT + I_1 \sin(\omega_m nT) - I_1)} \right\} \\ &\approx \phi(n). \end{aligned} \tag{6.21}$$

Finally, when substituting the approximation of (6.21) into (6.16), there is a cancellation of the sinusoidal terms so that

$$\theta_{dd}(n) \approx \omega_c nT - 2I, \tag{6.22}$$

showing that the resulting delayed signal

$$z_{dd}(n) \approx z(n) \tag{6.23}$$

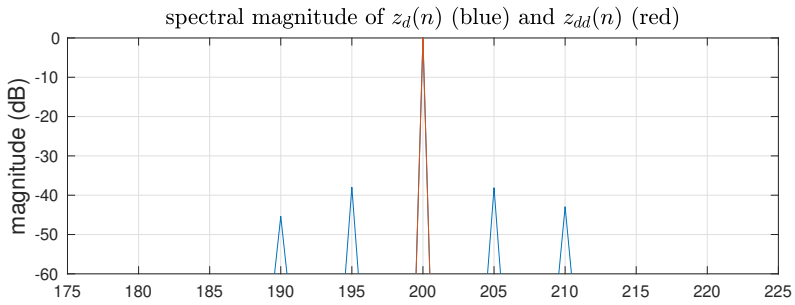

Figure 6.2. Spectral magnitude of the delay line output $z_{dd}(n)$ showing no sidebands greater than -60dB and thus greater vibrato reduction than $z_d(n)$.

is nearly equal to the original unmodulated sinusoid (omitting a pure delay) with improved vibrato reduction, as sidebands are attenuated well below the threshold of audibility (see Figure 6.2).

As in Chapter 4, it is instructive to examine the effectiveness of $D_{dd}(n)$ on a signal with multiple harmonics. As before, we will consider the signal $z_k(n)$ from (4.9), a sum of equal amplitude harmonics, modulated using a delay $D_m(n)$ to create the signal $z_{k,m}(n)$ (4.10), a sum of modulated sinusoids. Demodulating $z_{k,m}(n)$ using a recursive delay function attenuates all sidebands of the first harmonic, which is expected given the preceding analysis. The upper harmonics are also demodulated, as seen in Figure 6.3. In general, upper harmonics are modulated more strongly (with an index of modulation of kI for the kth harmonic), so the demodulating effectiveness of $D_{dd}(n)$ should be reduced for each successive harmonic. Fortunately, in acoustic signals, upper harmonics typically have a much quieter amplitude than the first harmonic and therefore less audible upper sidebands, so $D_{dd}(n)$ derived from the first harmonic should still be effective in suppressing audible sidebands in the upper harmonics of the signal.

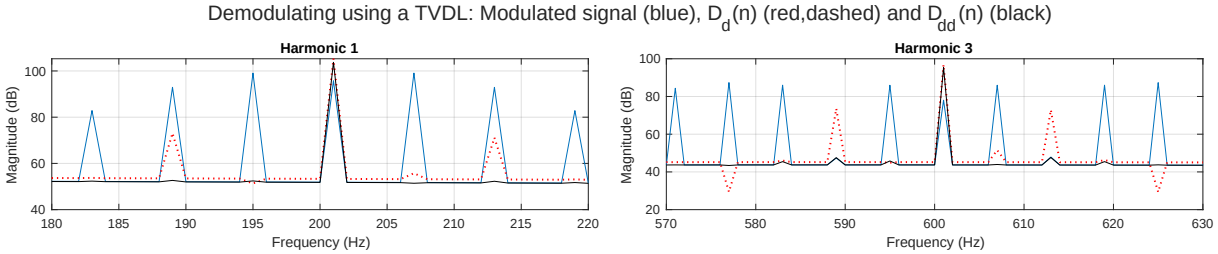

Figure 6.3. Using a TVDL to demodulate $z_{k,m}(n)$. $D_d(n)$ removes most sidebands but leaves some audible. Using the recursive delay $D_{dd}(n)$ shows stronger demodulation in both the fundamental and upper harmonics.

6.2.2 Deriving delay functions and demodulating real signals

Let us now consider demodulating the FM of an acoustic signal with vibrato. To demodulate a signal with vibrato $x(n)$ using a TVDL, the recursive demodulation function $D_{dd}(n)$ is derived by first deriving the modulating delay function $D_m(n)$ using the peak-picking method from Chapter 5. As a reminder, each time frame m of the spectral magnitude $|X(\omega_m, m)|$ is analyzed to find the instantaneous fundamental frequency signal $2\pi f_i^p(m)$, now labeled $\omega_1(m)$ (i.e., the instantaneous frequency of the first harmonic) for consistency with the sinusoidal resynthesis method for vibrato suppression (see Equation 4.1).

In a slight modification to the algorithm in Section 5.2, instead of determining a static ω_c to estimate the delay functions, $\omega_1(m)$ is smoothed using the low-pass filter (the same filtering from [50] and Section 5.3) to yield a time-varying fundamental frequency with suppressed modulation $\overline{\omega}_1(m)$. Both modulated and smoothed frequency patterns are then upsampled to create corresponding signals $\omega_1(n)$ and $\overline{\omega}_1(n)$, respectively, at the given sampling rate. $\overline{\omega}_1(n)$ is a time-varying fundamental frequency without vibrato and is used here place of a static ω_c.

With known modulated and smoothed time-varying frequency, the *momentary transposition* may be defined by their ratio

$$t_m(n) = \frac{\omega_1(n)}{\overline{\omega}_1(n)} \tag{6.24}$$

As before in Equation (5.4), the relative frequency shift (RFS) is the derivative of the modulating delay function and is given by

$$\frac{d}{dn} D_m(n) = 1 - t_m(n). \tag{6.25}$$

Integrating Eq. 6.25 numerically using a cumulative sum with respect to n yields $D_m(n)$, which may be phase inverted to obtain the demodulating delay function

$$D_d(n) = -D_m(n) = \sum_{l=0}^{n} \left(t_m(l) - 1\right), \tag{6.26}$$

with the final recursive demodulation function $D_{dd}(n)$ obtained by delaying (6.26) by itself.

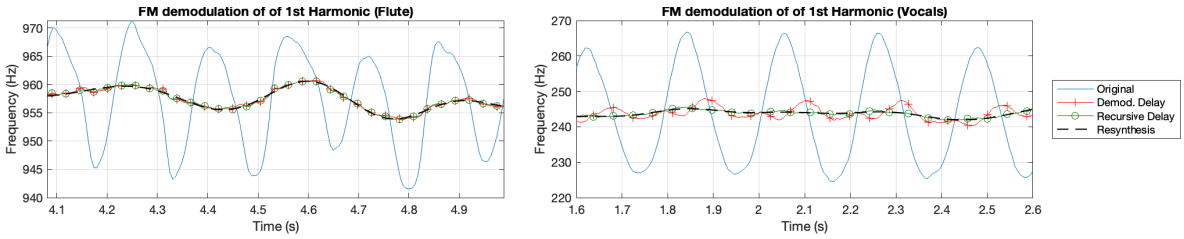

Figure 6.4. Frequency demodulation of the first harmonic of a flutist and a vocalist playing with heavy vibrato.

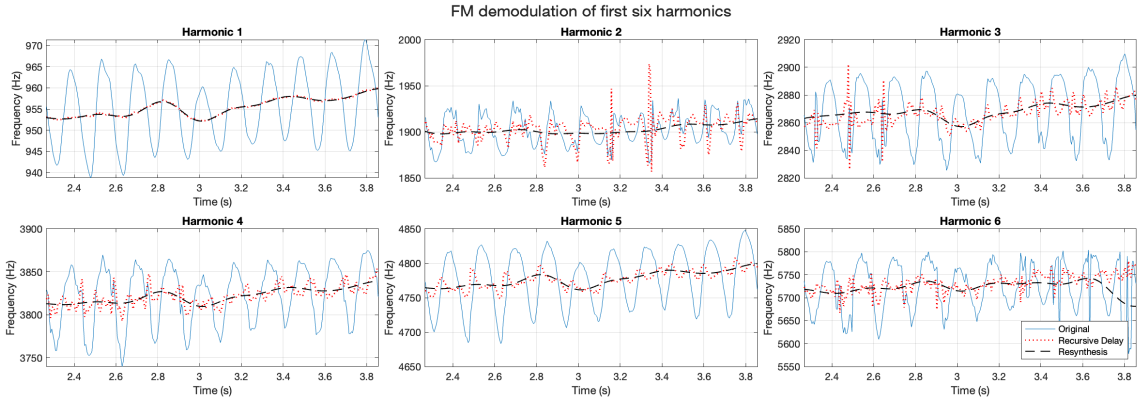

Figure 6.5. FM demodulating a flute. The demodulating delay function based on the first harmonic also demodulates the upper harmonics. Discontinuities the plot are due to analysis error and not present in the audio signals.

Although it was shown that $D_{dd}(n)$ sufficiently demodulates an FM sinusoid with a known modulating function, Figure 6.4 illustrates the extent to which that theory holds in practice. The first harmonic of a flute and vocal signal with vibrato are shown, as well as the first harmonic the signals demodulated using $D_d(n)$ and $D_{dd}(n)$, and for further comparison, a sinusoidal resynthesis using $\overline{\omega}_1(n)$ (see Section 4.2). The flute has a typical FM extent and the delay function $D_d(n)$ appears adequate for suppressing FM in the first harmonic. On the other hand, the vocal signal has a vibrato with a greater FM extent that is not sufficiently demodulated using $D_d(n)$. The recursive delay function $D_{dd}(n)$ is more successful at achieving an FM suppression comparable to the sinusoidal model but without the need for resynthesis.

Due to the similarity in vibrato extent across harmonics in natural signals and when using a TVDL (as shown in Section 4.3), demodulating using a TVDL based on the first harmonic

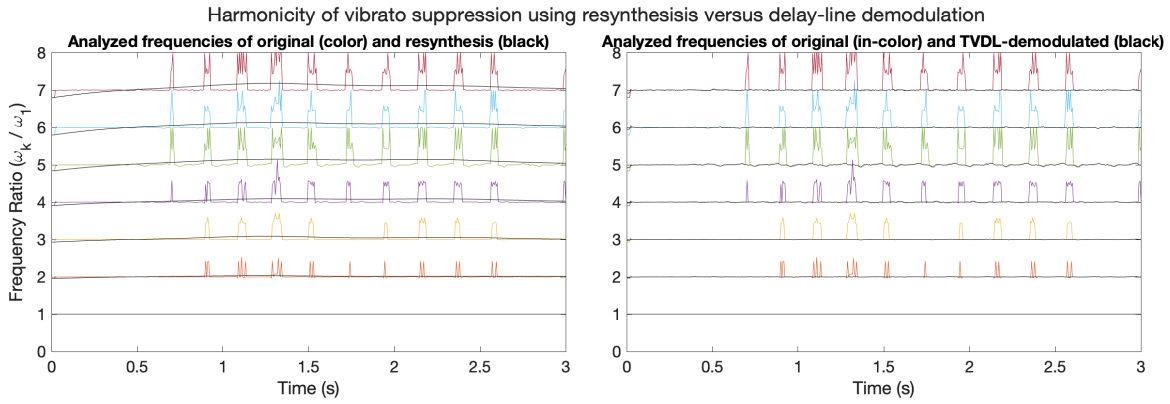

Figure 6.6. The ratio of harmonics (ω_k/ω_1) when suppressing vibrato. Errors tracking the frequency of quieter harmonics can lead to a loss of harmonicity in sinusoidal resynthesis (left). The TVDL treats all harmonics collectively and the original harmonicity is preserved (right).

of a signal is also successful as demodulating the upper harmonics. Figure 6.5 shows the time-varying frequency of the first six harmonics of the flute signal, the harmonics from the delay-line demodulated signal, and the low-passed signals $\overline{\omega}_k(n)$. The general shape of the demodulated signal's harmonics are comparable to $\overline{\omega}_k(n)$, although some modulation remains.

Because all signals in Figure 6.5 are derived using frequency-domain peak-picking, they are subject to analysis error due to the low amplitude and/or SNR. The difficulty of peak-picking in the upper harmonics points to an advantage of the TVDL method, which only requires the first harmonic to be tracked accurately. When resynthesizing a vibrato-suppressed signal sinusoidally, errors in tracking $\omega_k(n)$ can cause the low-passed resynthesis parameters $\overline{\omega}_k$ to become inharmonic. Figure 6.6 shows the ratio between frequency tracks $\omega_k(n)$ and $\omega_1(n)$ and those demodulated using either $\overline{\omega}_k$ or $D_{dd}(n)$ for a vocalist singing with heavy vibrato. Analysis errors in the upper harmonics lead to inharmonic partials when resynthesizing using low-passed frequency tracks. The TVDL-based demodulation appears to maintain the original signals harmonicity, subject to analysis errors that are not audible in the demodulated audio.

6.3 Vibrato suppression: amplitude demodulation

AM patterns caused by vibrato have different patterns across harmonics and residual noise. Demodulating AM, therefore, requires handling each component independently. Two methods for AM demodulation are presented here, one inspired by (and the inverse of) the vibrato transfer method presented in Section 5.3 and the other with no parallel in vibrato transfer.

Suppressing AM using sinusoidal modeling uses only the low-passed harmonic amplitudes and residual spectral envelope $\overline{A}_k(n)$ and $\overline{N}(\omega, m)$ in the resynthesis step (see Chapter 4, Equations (4.2)–(4.3)). Based on the success of this approach, the methods proposed here demodulate AM of a signal by conceptually applying the low-passed amplitude and spectral envelopes analogous to $\overline{A}_k(n)$ and $\overline{N}(\omega, m)$, but without resynthesizing the signal sinusoidally. Both methods rely on analyzing a signal, obtaining envelopes, and then dividing the envelopes out while applying low-passed envelopes in their place.

6.3.1 Bandpass Plus Spectral Envelope (BPSE)

The first proposed method is conceptually similar to vibrato suppression using sinusoid plus noise resynthesis and is the inverse of the AM transfer algorithm in Chapter 5. First, a signal $x(n)$ is FM demodulated using a TVDL, ensuring that all harmonics are relatively stable in frequency for the duration of the signal. The process then proceeds identically to the harmonic processing in Section 5.3, with the signal of the kth harmonic $h_k(n)$ isolated by a Butterworth bandpass filter centered at frequency $k\omega_c$. The envelope of the kth harmonic $E_{hk}(n)$ is created from the RMS signal of $h_k(n)$. The AM demodulated harmonic $\overline{h}_k(n)$ is then obtained as

$$\overline{h}_k = h(k) \cdot \frac{\overline{E}_{hk}(n)}{E_{hk}(n)}, \tag{6.27}$$

where the multiplying term $\frac{\overline{E}_{hk}(n)}{E_{hk}(n)}$ is the inverse of the AM transfer envelope from Equation 5.12.

The residual signal $x_r(n)$ is again derived using cascaded complementary Butterworth

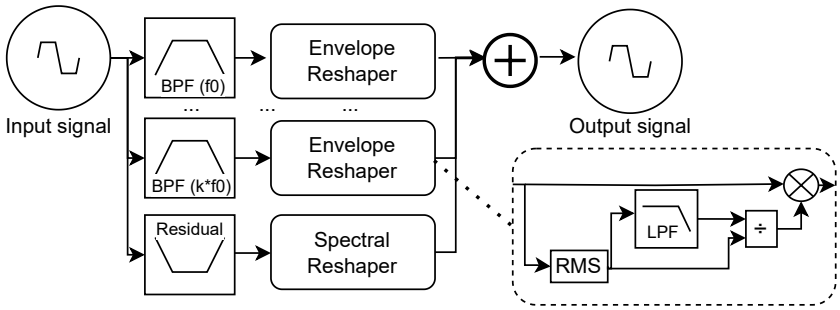

Figure 6.7. Removing AM from a signal using the RMS envelopes of its harmonics and the spectral envelope of the residual signal.

bandstop filters using Equation (5.13). We apply the STFT to the residual signal $x_r(n)$ to form $X_R(\omega, m)$, divide the frequency range into d non-overlapping segments for analysis, and finally obtain the spectral envelopes $X^*_{RS}(\omega, m)$ and $\overline{X}^*_{RS}(\omega, m)$ using Equations (5.14)–(5.15). This procedure now inverts the AM transfer method, with the vibrato suppressed residual is recovered by first creating

$$\overline{X}_R(\omega, m) = X_R(\omega, m) \cdot \frac{\overline{X}^*_{RS}(\omega, m)}{X^*_{RS}(\omega, m)} \tag{6.28}$$

and then taking the inverse Short-time Fourier transform (ISTFT) of $\overline{X}_R(\omega, m)$. The harmonic and residual signals are summed to create the final AM demodulated signal. The combination of bandpass and spectral envelope (BPSE) demodulation is depicted in Figure 6.7.

Mathematically, BPSE divides away the original envelopes of each harmonic and frequency range in the spectrum of the original, and then multiplies by the low-passed envelopes that are be used in the sinusoid plus noise resynthesis method. This method of envelope adjustment was previously proposed more broadly in Feature Modulation Synthesis [55].

6.3.2 Spectrogram smoothing (SS)

An alternative approach to isolating harmonics and noise is to treat the entire spectrogram of the input signal. After FM suppression using a TVDL, the STFT $X(\omega, m)$ is created from a signal $x(n)$ using windows that overlap by 75%. Each segment is multiplied by a periodic Hann window to ensure the constant overlap-add (COLA) constraint is met.

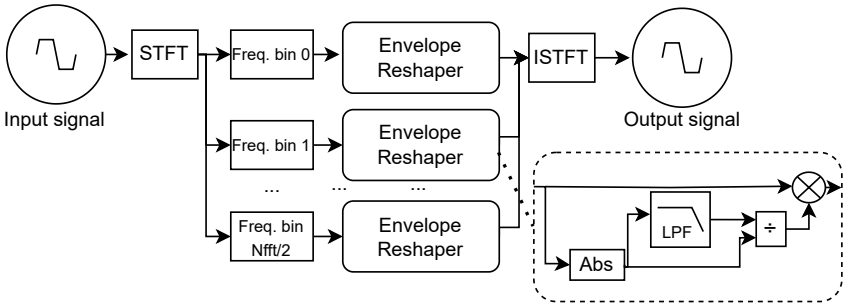

Figure 6.8. Removing AM from a signal by smoothing the amplitude of each frequency bin of its spectrogram.

We create an envelope $E_X(\omega, m) = |X(\omega, m)|$ and its low-passed counterpart $\overline{E}_X(\omega, m)$ by applying a lowpass filter along the time dimension for each frequency bin ω. Each bin is then envelope corrected along the time axis, creating the modified spectrogram

$$\overline{X}(\omega, m) = X(\omega, m) \cdot \frac{\overline{E}_X(\omega, m)}{E_X(\omega, m)}. \tag{6.29}$$

$\overline{X}$ is then inverted using the ISTFT to create the AM demodulated signal. This process of spectrogram smoothing (SS) (see Figure 6.8) is similar to the treatment of the spectral envelope in BPSE, but with no separation of harmonic and residual signals or reduction in spectral bins.

6.3.3 Comparison of AM demodulation methods

The importance of treating the amplitude modulation in both the harmonics and spectral envelope of the non-harmonic noise in a signal with vibrato was heavily emphasized by Roebel et al. when presenting the sinusoidal resynthesis method for vibrato extent control [50]. The examples used throughout their study were of a flute player, due to the prominent AM in the breath noise of the signals. As a point of comparison, the same flute signal (previously analyzed in Figures 4.4, 4.5 and 6.5) is demodulated using both the BPSE and SS methods.

Figure 6.9 shows the amplitudes of the first six harmonics of the flute signal. The BPSE and SS methods demodulate the amplitude of each harmonic comparably and reduce most audible

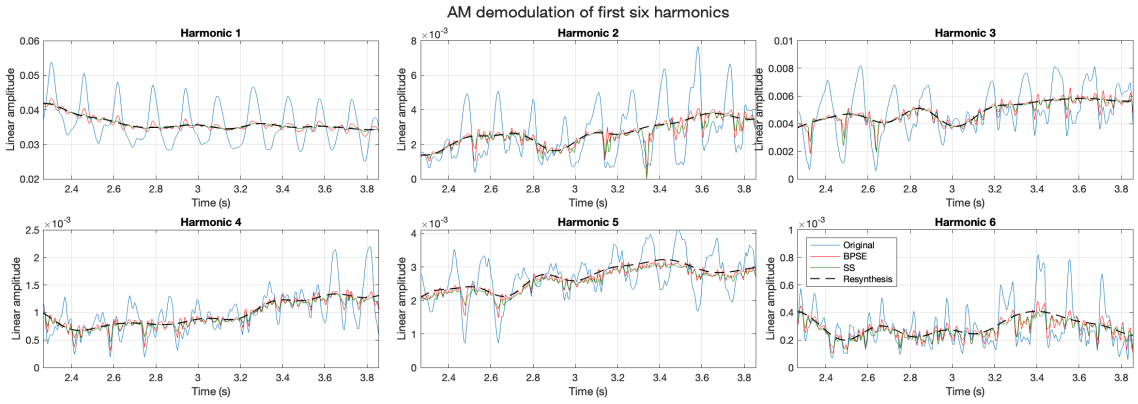

Figure 6.9. Amplitude demodulation of upper harmonics of a flute. Both the BPSE method and the SS method result in harmonics with less amplitude modulation than the original signal. The low-passed amplitude track used in sinusoidal resynthesis is more demodulated in all cases.

AM in practice. Both methods contain more residual vibrato than the low-passed amplitude tracks $\overline{A}_k$ used in resynthesis. As before, amplitude tracks are found spectral peak-picking and are subject to analysis error due to the high SNR in this signal, so visible jumps in the analysis may not be present in the actual signal.

Demodulation of the residual noise is most easily examined using spectrograms. Figure 6.10 shows the spectrograms of the flute signal before and after demodulation using Roebel's resynthesis method, and demodulation using the TVDL to suppress FM and either BPSE or SS to suppress AM in both the harmonics and the residual noise. Sinusoid plus noise resynthesis successfully demodulates the flute signal, with some artifacts. The attack of the signal is more diffuse than the original due to the lowpass filter used to create $\overline{A}_k(n)$. The residual noise, while demodulated, is more diffuse than the original. Examining the original signal reveals the presence of a subharmonic one octave below the original signal, which has its own harmonics with reasonably significant amplitude (namely, the third and fifth harmonic of the subharmonic, i.e., 1.5x and 2.5x the true fundamental frequency). These components are more sinusoidal in both BPSE and SS due to the residual signal being processed directly, as opposed to resynthesized. BPSE shows the worst performance at AM demodulating the residual noise; however, it is better at preserving the attack of the residual signal.

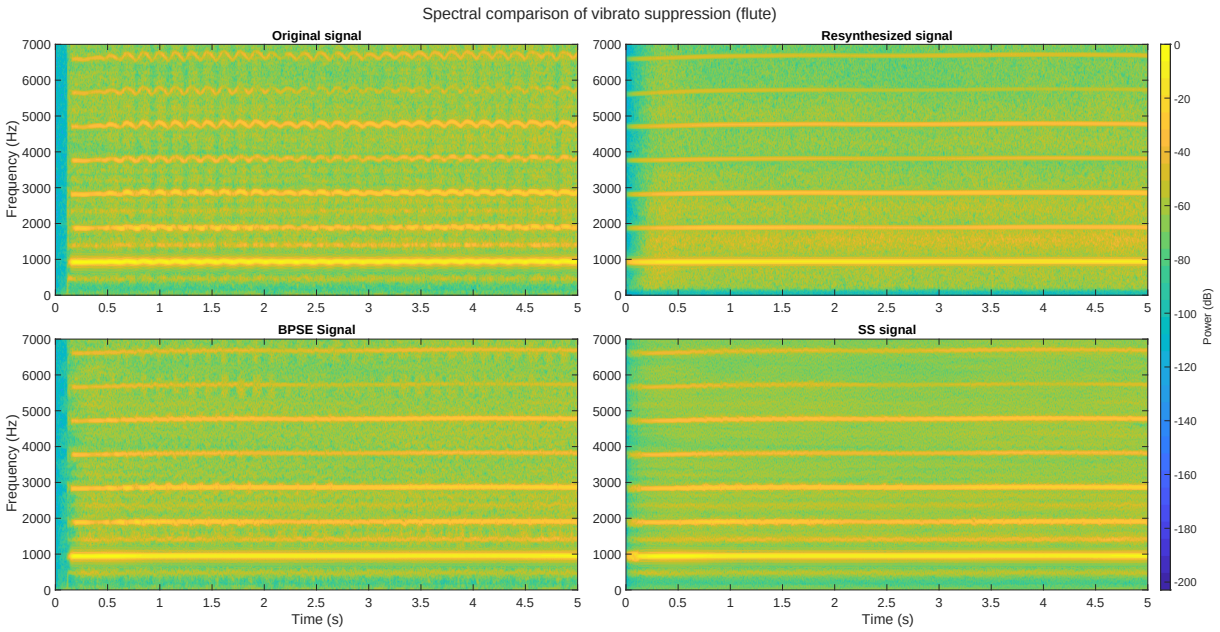

Figure 6.10. Spectrograms of a flute sample, original and vibrato suppressed using: sinusoidal resynthesis, TVDL FM demodulation and BPSE AM demodulation, and TVDL FM demodulation and SS AM demodulation.

Figures 6.11 and 6.12 show additional examples of a saxophone and vibraphone. These examples, along with the flute signal analysis in Figures 6.9–6.10, reveal several trends. First, the low-passed amplitude tracks used in resynthesis appear to show the most demodulation; however, preliminary listening tests suggest that there is negligible audible difference between techniques with regards to harmonic demodulation. BPSE shows the poorest demodulation of the spectral envelope, while best preserving the attack of the original signal. Both resynthesis and SS smear the attack (this is particularly noticeable in Figure 6.12).

The main benefit of the SS method is the ability to perform vibrato suppression without the need to accurately model the amplitude of quieter upper harmonics. As we have already shown in Figure 6.6, difficulty in tracking the upper harmonics can cause the low-passed frequency tracks $\overline{\omega}_k(n)$ to become inharmonic. Here we also observe that deriving imprecise amplitude tracks $A_k(n)$ causes the residual spectral envelope $N(\omega, m)$ to contain peaks corresponding to harmonics, which remain in the low-passed spectral envelope $\overline{N}(\omega, m)$. As a result, the SNR between harmonics and noise is different in the resynthesis compared to the original signal.

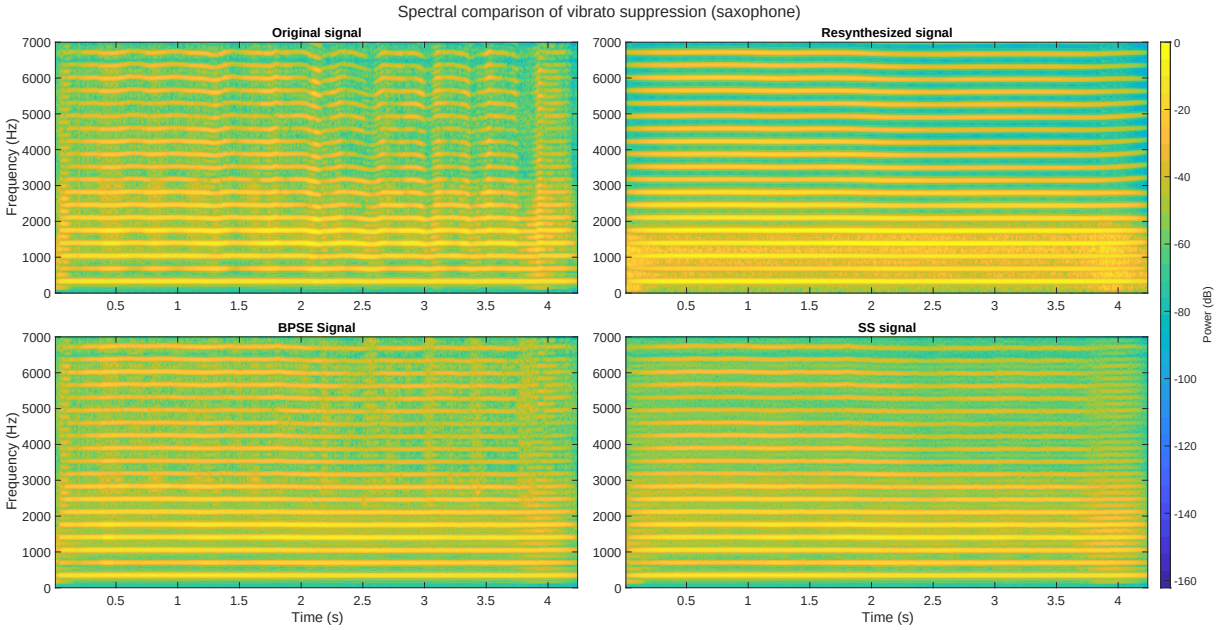

Figure 6.11. Spectrograms of a saxophone sample, original and vibrato suppressed using: sinusoidal resynthesis, TVDL FM demodulation and BPSE AM demodulation, and TVDL FM demodulation and SS AM demodulation.

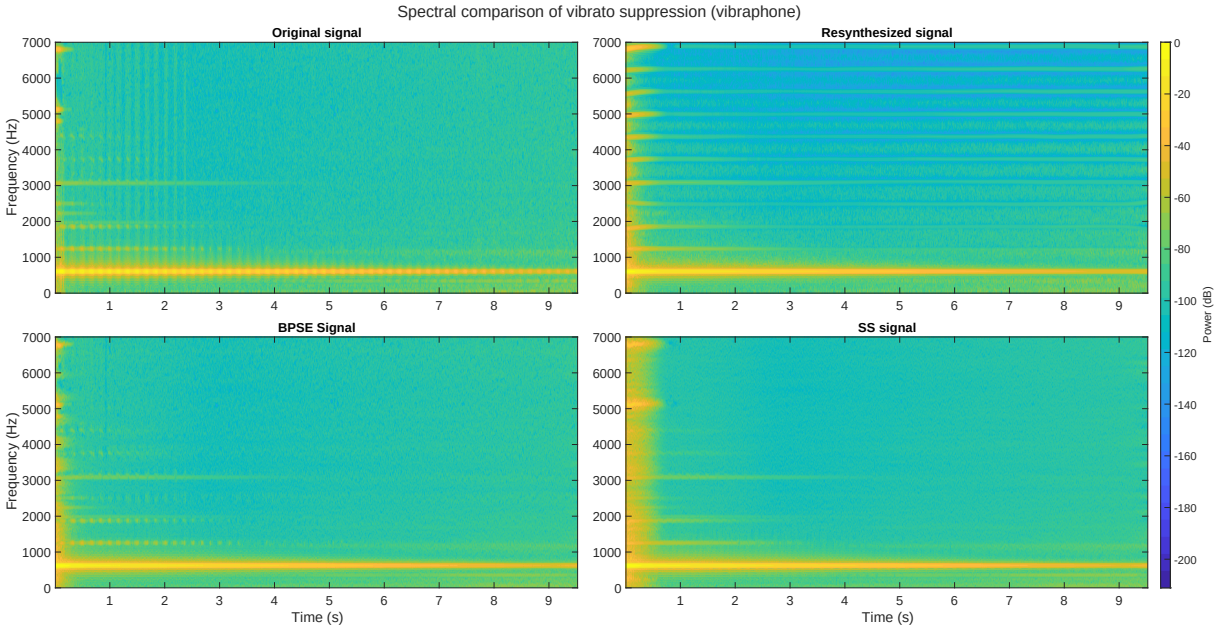

Figure 6.12. Spectrograms of a vibraphone sample, original and vibrato suppressed using: sinusoidal resynthesis, TVDL FM demodulation and BPSE AM demodulation, and TVDL FM demodulation and SS AM demodulation.

6.4 Combining vibrato suppression and vibrato transfer

When transferring vibrato-induced AM and FM from a source signal $s_s(n)$ onto a target signal $s_t(n)$, vibrato patterns will not match if $s_t(n)$ contains its own vibrato. Instead, existing patterns in $s_t(n)$ will be additionally modulated. In the motivating case of sound blending using vibrato matching, it is preferable for the vibrato patterns of sounds in a mix to match as closely as possible, so vibrato suppression is a complementary pre-processing step to vibrato transfer.

Algorithm 5: Vibrato matching two signals

Data: $s_s(n), s_t(n), N_h, f_0^s, f_0^t, 0 < bw_s < \frac{1}{3}, 0 < bw_t < \frac{1}{3}$

/* Demodulate target */

$\omega_1^t(n) \leftarrow$ peak_pick($s_t(n)$) /* Eqs. (5.6)–(5.8) */

$\overline{\omega}_1^t(n) \leftarrow$ LPF($\omega_1^t(n)$) /* Eq. (4.1) */

$D_d^t(n) \leftarrow$ get_Dd($\omega_1^t(n), \overline{\omega}_1^t(n)$) /* Eqs. (6.24)–(6.26) */

$D_{dd}^t(n) \leftarrow$ delay($D_d^t(n), D_d^t(n)$) /* Eq. (6.14) */

$\widehat{s}_t(n) \leftarrow$ delay($s_t(n), D_{dd}^t(n)$) /* FM demodulate target */

$\overline{s}_t(n) \leftarrow$ SS($\widehat{s}_t(n)$) /* AM demodulate target, Eq. (6.29) */

/* Get modulating functions from source */

$h_k^s(n), r^s(n) \leftarrow$ get_h_and_r($s_s(n), N_h, f_0^s, bw_s$) /* Isolate N_h harmonics and residual, Eqs. (5.11),(5.13) */

$AM_k(n) \leftarrow$ harm_envs($h_k^s(n)$) /* Compute $E_{hk}(n), \overline{E}_{hk}(n)$ using filters with bandwidths $k \cdot f_0^s \cdot bw_s$ for kth harmonic, Eq. (5.12) */

$AM_R(\omega, m) \leftarrow$ res_env($r^s(n)$) /* Eqs. (5.15),(5.16) */

$\omega_1^s(n) \leftarrow$ peak_pick($s_s(n)$)

$\overline{\omega}_1^s(n) \leftarrow$ LPF($\omega_1^s(n)$)

$D_m^s(n) \leftarrow$ get_Dm($\omega_1^s(n), \overline{\omega}_1^s(n)$) /* Eqs. (6.24),(6.25) */

/* Transfer source vibrato to demodulated target */

$h_k^t(n), r^t(n) \leftarrow$ get_h_and_r($\overline{s}_t(n), N_h, f_0^t, bw_t$)

$\widetilde{h}_k^t(n) \leftarrow h_k^t(n) \cdot AM_k(n)$ /* Apply AM per-harmonic */

$\widetilde{h}_t(n) = \sum^{N_h} \widetilde{h}_k^t(n)$ /* Sum harmonics */

$h_t^* \leftarrow$ delay($\widetilde{h}_t(n), D_m^s(n)$) /* Apply FM to harmonic sum */

$R_t(\omega, m) \leftarrow$ STFT($r^t(n)$)

$R_t^*(\omega, m) \leftarrow R_t(\omega, m) \cdot AM_R(\omega, m)$ /* Apply AM to residual */

$r_t^* \leftarrow$ ISTFT($R_t^*(\omega, m)$)

$s_t^* \leftarrow h_t^* + r_t^*$ /* Mix harmonics and residual */

Algorithm 5 outlines a vibrato matching algorithm that uses both vibrato suppression and the offline vibrato transfer algorithm detailed in Section 5.4. The basic logic of the vibrato matching algorithm is as follows. The target signal $s_t(n)$ is FM and AM demodulated using a TVDL and SS to create $\overline{s}_t(n)$. The harmonics and residual signals of the vibrato source signal $s_s(n)$ and $\overline{s}_t(n)$ are isolated, and AM envelopes are extracted from the harmonics and residual of $s_s(n)$. AM is applied to the harmonics and residual of $\overline{s}_t(n)$, and the harmonics are summed and frequency modulated using the modulating delay derived from $s_s(n)$. Finally, the modulated harmonics and residual are summed to create the vibrato-matched signal $s_t^*(n)$.

The algorithm is parameterized by N_h, the number of harmonics to process in AM transfer, the f_0 of each signal (determined using any f_0 estimation algorithm), and two fractional bandwidths bw_s and bw_t. Bandpass and bandstop filters are used to isolate the harmonics and residual for each signal, with filters centered at frequency $k \cdot f_0$ for the kth harmonic. The bandpass width parameters determine the width of the passband and stopband respectively, with bandwidths set to $bw \cdot k \cdot f_0$ The harmonics of $s_s(n)$ contain vibrato, so it is expected that the harmonics will vary in frequency over time. Conversely, the harmonics of $\overline{s}_t(n)$ are isolated after vibrato suppression, so the frequency of each harmonic should be somewhat static. For this reason, it is useful for bw_s to be wider than bw_t in practice.

6.4.1 Vibrato matching for sound blending

To illustrate the effectiveness of vibrato matching, it is instructive to process recordings of an instrumentalist playing a note with differing amounts of vibrato and examine the extent to which the vibrato patterns are matched after processing. Previously, Figures 5.11–5.13 showed the effect of mixing signals of the same note with and without vibrato using the vibrato transfer algorithm. Below, the problem is complicated by mixing two signals from the same instrumentalist, each with their own vibrato rate and width. Figure 6.13 demonstrates that the vibrato matching algorithm can successfully match vibrato patterns on vocal signals by first suppressing the target signal and then transferring vibrato from the source to the target. The

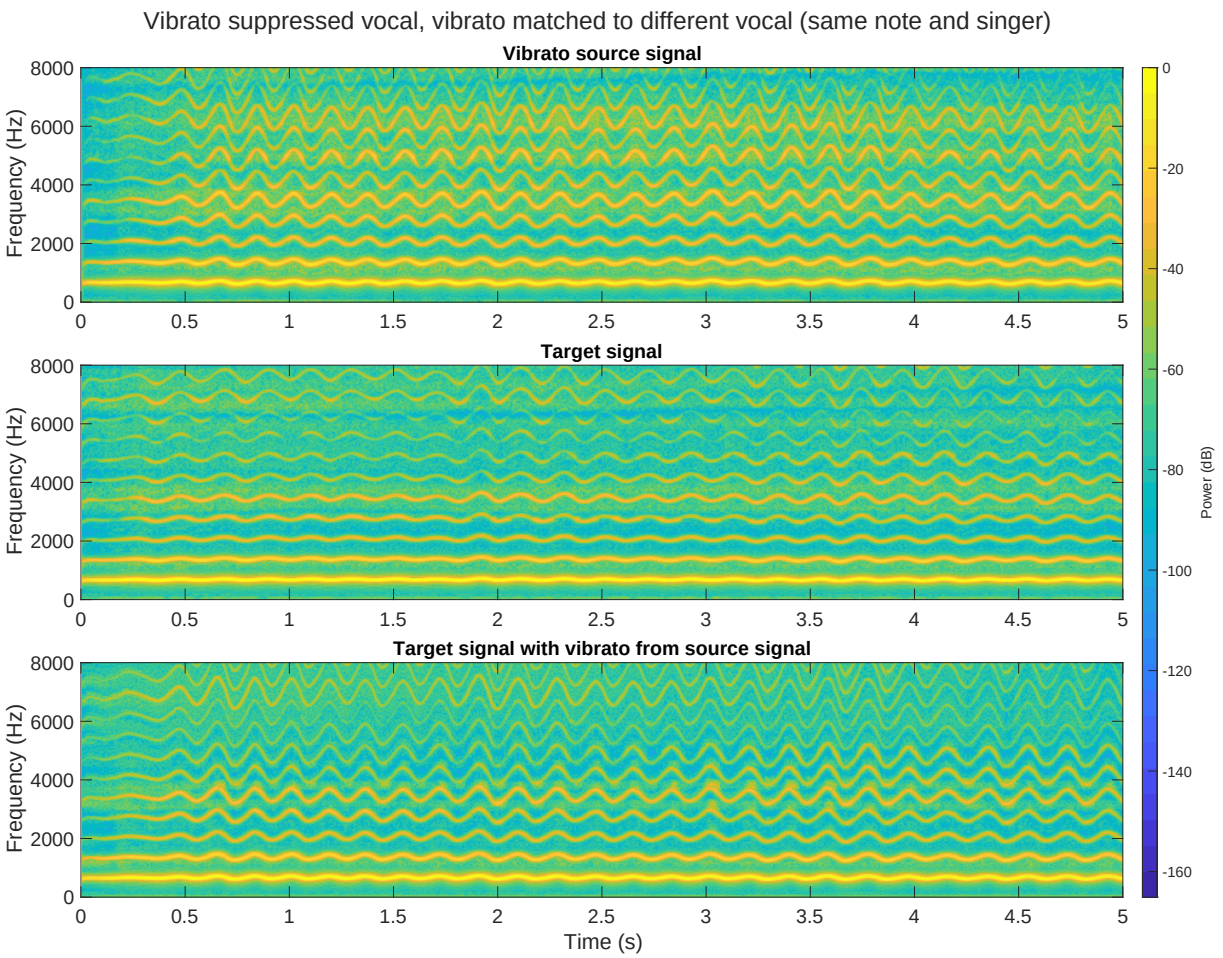

Figure 6.13. Two signals of a vocalist singing the same note with different vibrato patterns. The top and middle figures show the source and target signals, and the bottom figure shows the target signal after its vibrato was suppressed and vibrato from the source was transferred onto it. The bottom figure shows very similar modulation patterns as the source signal.

mixture of these signals pre- and post-vibrato match are shown in Figure 6.14. Similar to the example in Figure 5.11, the spectrogram of the post-match mix visually appears to be one source, and the sonic result is again more fused. Additional spectrograms and sonic examples of different instruments playing in unison are available on the supplemental website. Applying vibrato matching to signals playing in unison often obscures the presence of different sources, pointing to a potential use in hybrid instrument sound design.

Figures 6.15 and 6.16 move beyond unison signals to sources from different instrumentalists playing different notes. The figures show signals from a flutist and the previously analyzed

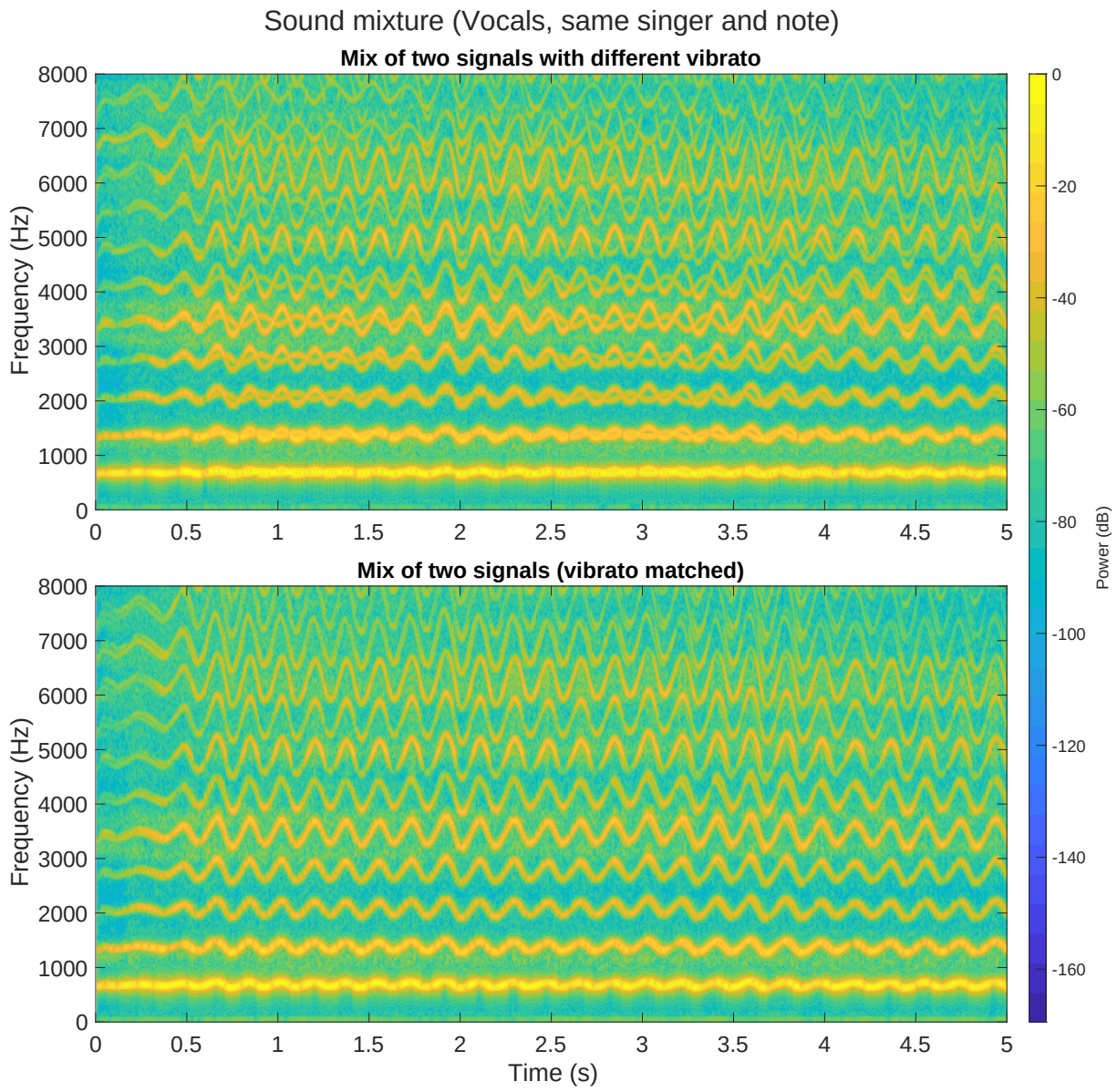

Figure 6.14. The source and target vocal signals from Figure 6.13 are shown in a mix. The pre-vibrato-matched mix (top) appears visually as two sources, where the post-vibrato-matched mix (bottom) is more ambiguous.

vocalist and two different vocalists, respectively. Mixes created before vibrato matching show different modulation patterns that can be visually identified as separate sources. Post-vibrato matching, the situation is more opaque. The spectrograms clearly show the presence of different notes based on the harmonic patterns, but the shared modulation suggests either that the notes originate from the same source or that they modulate harmoniously. The strong visual match of modulation patterns suggests that vibrato matching can be used as a creative tool to produce a sonic perception of common fate.

6.4.2 Vibrato matching and unison source separation

No formal listening tests were conducted on vibrato-matched signals, so it cannot be confirmed emperically that listeners perceive the sound to be more fused or if it is more difficult to determine the presence of multiple sources. However, because vibrato patterns have been shown to aid in source separation in mixes of instruments playing in unison [46, 47], it follows that matching vibrato patterns should degrade the quality of source separation. The use of vibrato to aid in automated source separation is based on listening tests where vibrato was shown to aid human listeners in counting the number of instruments in a mix [43, 44]. It stands to reason that degradations in automated source separation could correspond to a degradation in a listener's ability to perceive multiple instruments.

To evaluate this theory, mixes of unison instruments with different vibrato patterns were processed before and after matching vibrato patterns using Algorithm 5. Source separation was performed using a Python package associated with the Common Fate Transform (CFT) algorithm for unison source separation [47]. The CFT is calculated as a two-dimensional Discrete Fourier Transform over an STFT split into grids. The transform is meant to capture modulation in both amplitude and frequency so that mixes of signals with the same fundamental frequency (and therefore overlapping harmonics in the STFT) can be isolated by modulation patterns.

Three example mixes were tested using the CFT. In each case, sound mixes were prepared analogously to sound examples provided by the CFT package's authors. Each sound source

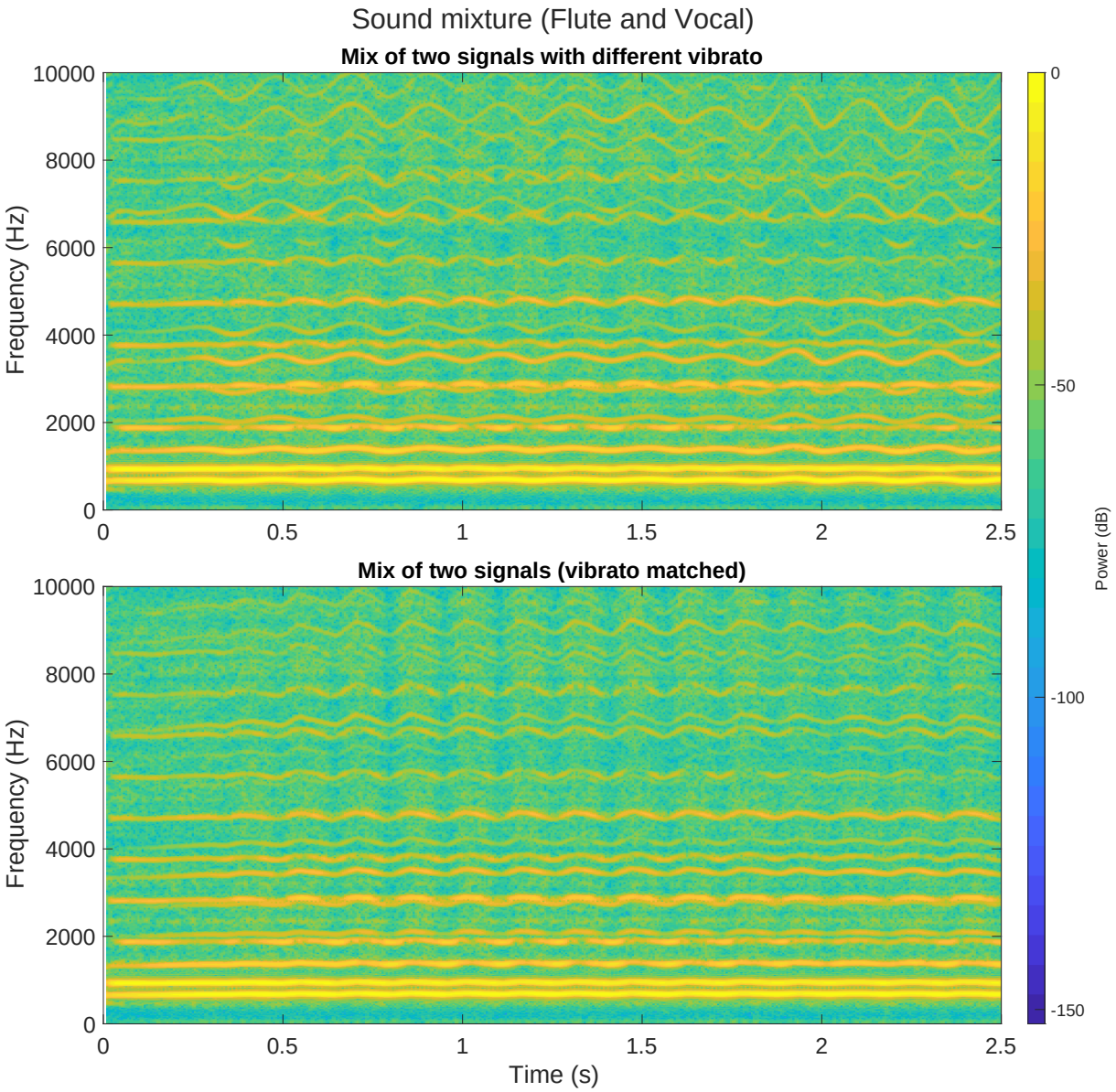

Figure 6.15. The pre- and post-vibrato-matched sound mixtures of a flutist and a vocalist playing different notes.

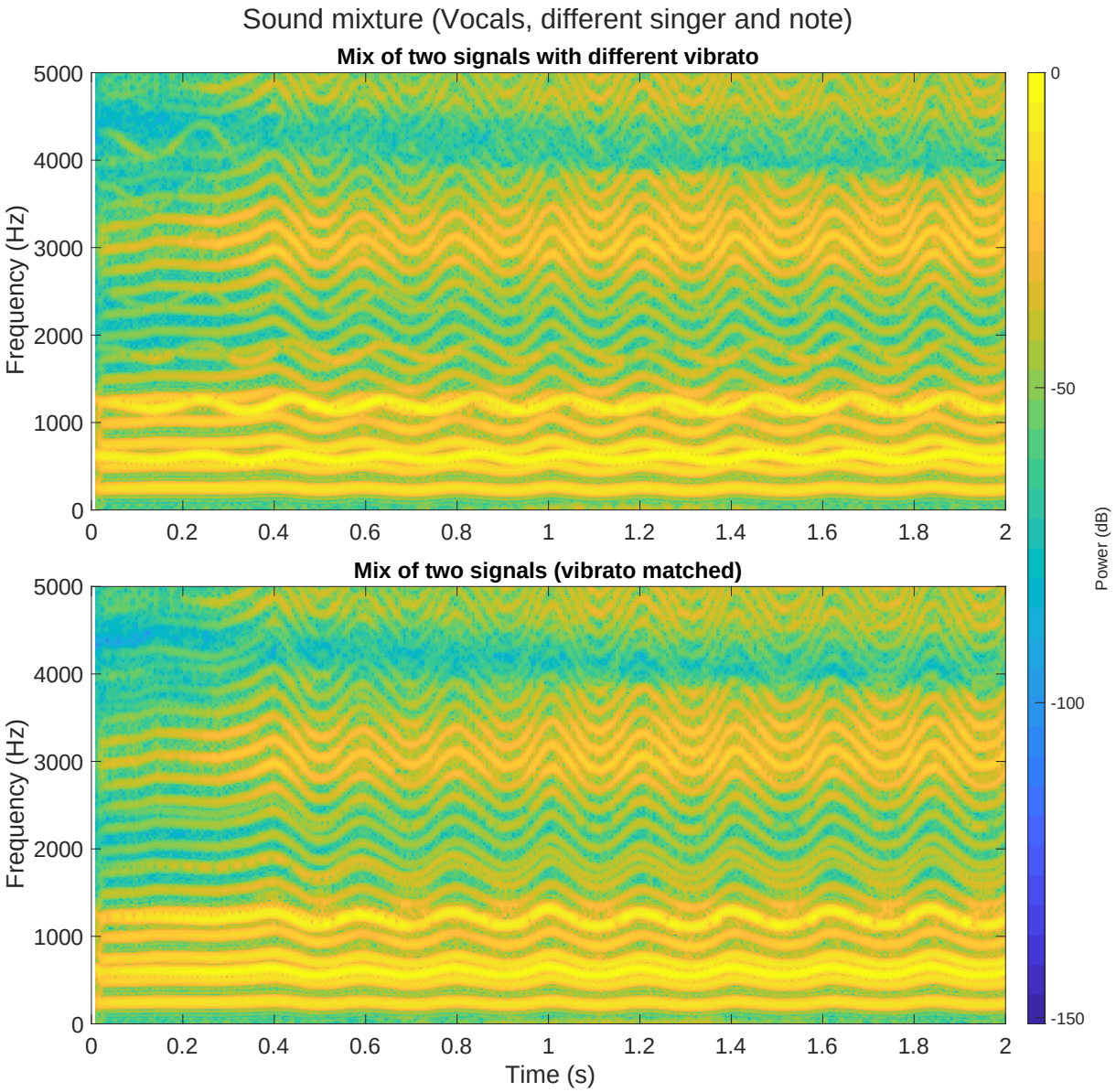

Figure 6.16. The pre- and post-vibrato-matched sound mixtures of two vocalists singing different notes.

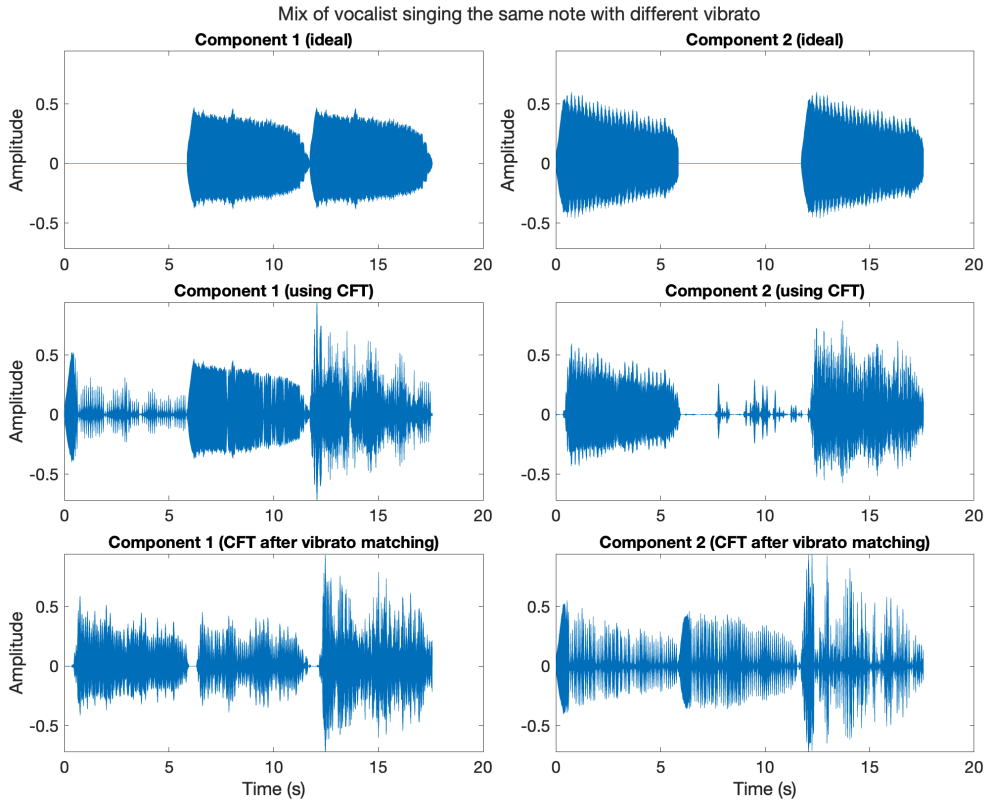

Figure 6.17. Using unison source separation based on vibrato patterns on two signals from the same vocalist. The ideal separation is shown in top row, while the center row shows the actual separation. When matching the vibrato patterns using Algorithm 5 before mixing, source separation is much poorer as seen in the bottom row.

is played once in isolation (the first source followed by the second), and then both sources are played at the same time. While it is unlikely that source separation algorithms would be conditioned on isolated sound sources in a real musical example, this sequencing was followed in an attempt to maximize the separation quality using the CFT algorithm.

In the first example, two signals of the same vocalist are used. Each signal has a unique vibrato pattern (rate and depth over time). This example is likely the hardest example for a source separation algorithm, as modulation patterns are the only indicator of multiple sources due to the spectral similarity of the two signals. Figure 6.17 shows the ideal outputs of the source separation algorithm, as well as the signals isolated before and after vibrato matching. The middle row demonstrates the source separation ability of the CFT. The second signal is isolated

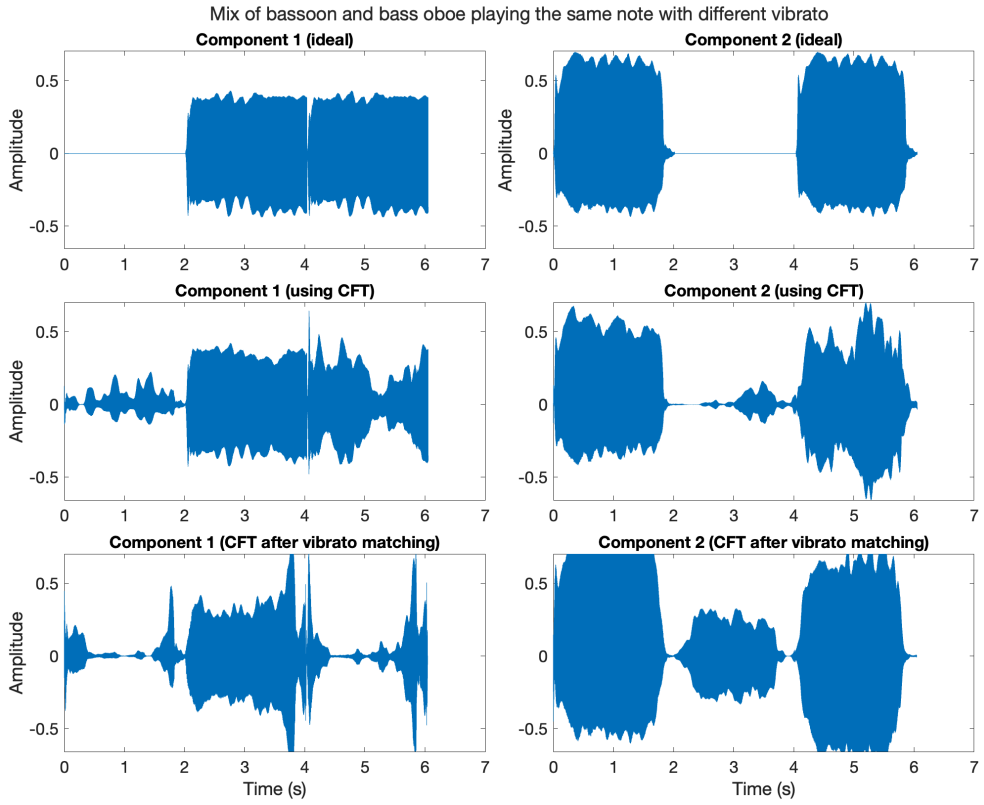

Figure 6.18. Unison source separation on bassoon and bass oboe signals with vibrato. Synchronizing vibrato patterns reduces the effectiveness of unison source separation. Degradation is observed in the mixed portion of the first signal and the silent section of the second signal.

admirably, while the first signal is isolated somewhat well but with noticeable leakage. The bottom row shows that vibrato matching has completely degraded the CFT's ability to separate these signals. This is a positive result in the context of vibrato matching as a means of sound blending; however, this result is somewhat expected as the two signals, once vibrato-matched, are remarkably similar as shown in Figures 6.13 and 6.14

Figures 6.18 and 6.19 show the same experiment using an bassoon and bass oboe pair and a trumpet and vocal pair, respectively. In the both examples, the CFT is relatively successful at separating the signals when they have different vibrato patterns and shows poorer performance after vibrato matching. The results are not as striking as Figure 6.17, but degradations are observed in each source signal in Figure 6.18 and in the second source in Figure 6.19. The ability of the CFT to isolate some portions of each source even when their pitch and modulation

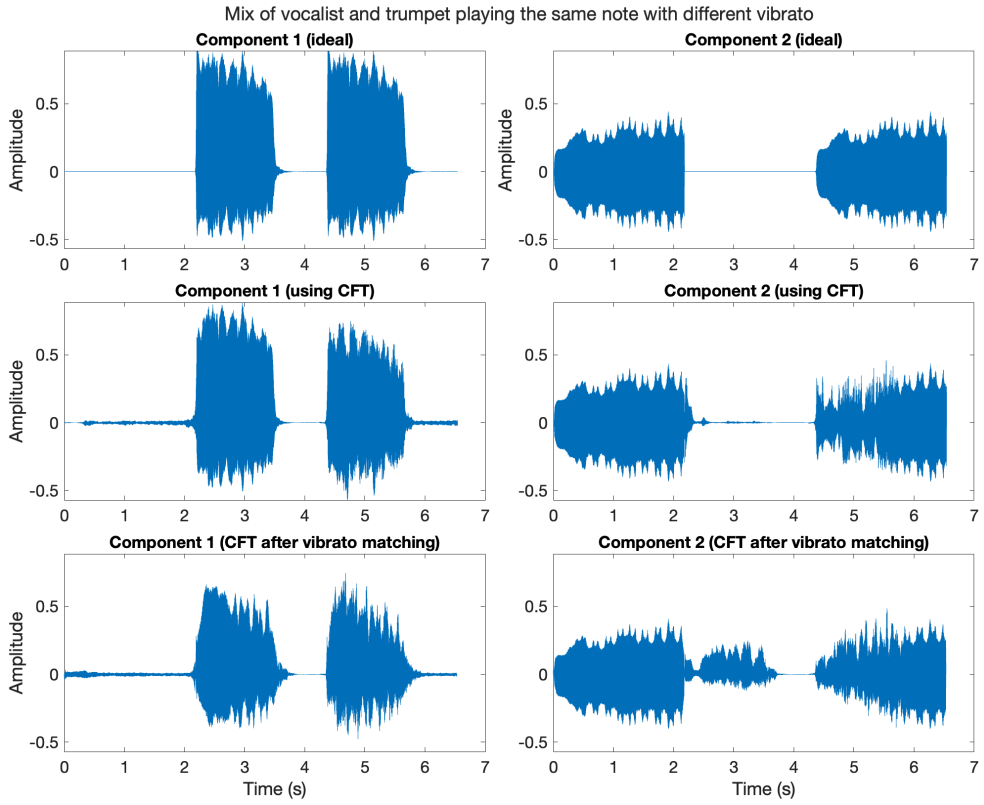

Figure 6.19. Using unison source separation based on vibrato patterns on vocal and trumpet signals. Synchronizing vibrato patterns reduces the effectiveness of unison source separation, primarily in the silent portion of the second source.

are nearly identical suggests that other spectral features play a role in unison source separation. Nevertheless, the results point toward a reduced ability in a machine (and perhaps human) listener to detect the presence of multiple sources after vibrato patterns are matched in each source.

6.5 Conclusion

In this chapter, we have shown the viability of suppressing vibrato in a recorded sound using a TVDL with a recursive demodulating delay function to suppress FM and two methods (BPSE and SS) to suppress AM. Because artificial vibrato using a TVDL has effects on the sidebands of a harmonic signal similar to that of natural vibrato, demodulation using a delay derived from the first harmonic is sufficient to demodulate upper harmonics. The TVDL method is also effective due to the high correlation between FM patterns across harmonics, as seen in

Chapter 4.

Of the proposed methods, SS is more successful at amplitude demodulation. Sinusoidal modeling and resynthesis remains a viable option to suppress vibrato in a signal, so long as parameters are chosen correctly and analysis errors are kept to a minimum. However, inaccuracies in parameter choice and analysis can lead to loss of fidelity. Using a TVDL and SS to demodulate has the benefit of maintaining as much information from the original signal as possible while suppressing vibrato, with very little parameterization required.

While BPSE is less successful at AM demodulation, its counterpart is successful at AM transfer in the vibrato transfer algorithm proposed in Chapter 5. Vibrato suppression paired with vibrato transfer is used to create an effective vibrato matching algorithm, in which vibrato patterns from two signals are matched regardless of whether both signals have their own pre-existing vibrato. This algorithm has potential in sound design and experimental music practices. Experiments with unison source separation show that matching the vibrato of unison signals degrades the ability to automatically separate the signals, which may suggest a similar effect for listeners. A formal listening test would be helpful to establish whether vibrato-matched sound mixtures are perceived as more fused or even as a single source.

Chapter 6 contains material adapted from the following research papers: *On Vibrato and Frequency (De)Modulation in Musical Sounds*, published in the Proceedings of the International Conference on Digital Audio Effects (DAFx 2024), *Real-time implementation of vibrato transfer as an audio effect*, published in the Proceedings of the International Computer Music Conference (ICMC 2025), and *Vibrato Suppression by Time-Varying Delay and Spectral Magnitude Demodulation*, published in the International Symposium On Musical and Room Acoustics (ISMRA 2025), respectively. The dissertation author was the primary investigator and author of these publications, and Dr. Tamara Smyth was coauthor on the publications accepted to DAFx 2024 and ISMRA 2025.

Conclusion

Each section of this dissertation presents unique problems related to the perceptual qualities of sound mixtures. Part I focuses on the control of auditory roughness caused by sound partials that are close in frequency, which most often occurs in sound mixtures. Three algorithms were developed to control roughness independently of pitch shifting or retuning, the traditional approach to roughness control in music. Individual audio signals are modified using a filter bank to isolate partials, with the frequency, amplitude or spatial panning of each partial controlled algorithmically. Using these methods, the roughness of the sound combination is changed even though the roughness within each audio signal is essentially unchanged.

Part II focused on vibrato patterns that manifest as modulation in both the frequency and amplitude of musical signals. Vibrato can be modeled, transferred and removed using a delay line, a filter bank with envelope control and spectral envelope reshaping. It is known that vibrato is a powerful perceptual indicator when counting the number of instruments and in source separation. Matching vibrato patterns in a mix seems to greatly reduce perception of multiple instruments based on spectral analyses and tests using automatic source separation. Vibrato matching is therefore useful not only as an individual signal shaping algorithm but also as a creative mixing tool.

Both roughness control and vibrato matching focus on modulation, be it in one signal or in a mix. This connection emphasizes the perceptual importance of modulation as a sonic feature, and suggests that control of modulation in sounds mixtures is a powerful tool in composition, audio engineering and sound design. Spatialized approaches to modulation in mixes remains an open research area. The panning technique explored in Part I, in which two partials that cause rapid beating are positioned to one per ear, is related to binaural beating, in which two sine waves of similar frequency are played one per channel in headphones. It is known that the beating perceived by a listener is different than the perception of roughness due to the lack of nearby partials in the critical bands of the each ear [56]. Recent work has explored spatializing tones and the effect of spatialization on binaural beating [57], but there is still a need to connect this work to spatialization of broader signals with modulation in sound mixtures.

While each problem presented here is solved using a unique algorithmic approach, there are shared architectures and themes. Solutions to roughness control and vibrato matching use a bank of bandpass filters to isolate sound partials and cascaded bandstop filters to capture the residual signal. The filter bank approach connects each method to the broader problem of automatic mixing, where each signal is processed using a multiband equalizer. However, while in audio engineering equalization is applied to broader frequency regions, the solutions in Parts I and II use highly targeted filters that isolate individual components of the sound. Such targeted filters would be prohibitively intensive and time-consuming for engineers mixing by ear, but future applications of automatic mixing could consider this more granular approach.

There are several architectural issues in current audio Application Programming Interfaces (APIs) that inhibit the implementation of these mix-based algorithms in a manner that can be easily incorporated into Digital Audio Workstations. Plugin interfaces such as VST or AU typically focus on processing inputs independently. Sidechain inputs can be used for analysis, but sharing information between plugins is not natively supported at the API level, and the use of shared global memory is ill-advised because plugin hosts are not guaranteed to run plugins in the same process. The shared architecture and processes highlighted in these use cases (i.e., shared information between analyses, filter bank to isolate sound partials) suggest that a new API for analyzing and processing signals using shared information between multiple tracks could be used to implement these and other mix-informed signal processing algorithms.

Bibliography

[1] B. De Man, J. Reiss, and R. Stables, "Ten years of automatic mixing," in *Proceedings of the 3rd Workshop on Intelligent Music Production (WIMP)*, 2017.

[2] R. Y. Litovsky, "Spatial release from masking," *Acoust. Today*, vol. 8, no. 2, pp. 18–25, 2012.

[3] E. C. Cherry, "Some experiments on the recognition of speech, with one and with two ears," *Journal of the acoustical society of America*, vol. 25, pp. 975–979, 1953.

[4] J. Hyrkas, A. D. Wilson, J. Tang, H. Gamper, H. Sodoma, L. Tankelevitch, K. Inkpen, S. Chappidi, and B. Jones, "Spatialized audio and hybrid video conferencing: Where should voices be positioned for people in the room and remote headset users?" in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–14.

[5] D. Schwarz, "Concatenative sound synthesis: The early years," *Journal of New Music Research*, vol. 35, no. 1, pp. 3–22, 2006.

[6] B. Hackbarth, N. Schnell, and D. Schwarz, "Audioguide: a framework for creative exploration of concatenative sound synthesis," *Musical research residency report. Paris, Institut de Recherche et Coordination Acoustique-Musique*, 2010.

[7] G. Carpentier, G. Assayag, and E. Saint-James, "Solving the musical orchestration problem using multiobjective constrained optimization with a genetic local search approach," *Journal of Heuristics*, vol. 16, pp. 681–714, 2010.

[8] M. Caetano, A. Zacharakis, I. Barbancho, and L. J. Tardón, "Leveraging diversity in computer-aided musical orchestration with an artificial immune system for multi-modal optimization," *Swarm and Evolutionary Computation*, vol. 50, p. 100484, 2019.

[9] C.-E. Cella, "Orchidea: A comprehensive framework for target-based computer-assisted dynamic orchestration," *Journal of New Music Research*, vol. 51, no. 1, pp. 40–68, 2022.

[10] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[11] M. Caetano, "Morphing musical instrument sounds with the sinusoidal model in the sound morphing toolbox," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2019, pp. 481–503.

[12] E. Zwicker, "Subdivision of the audible frequency range into critical bands (frequenzgruppen)," *The Journal of the Acoustical Society of America*, vol. 33, no. 2, pp. 248–248, 1961.

[13] B. C. Moore and B. R. Glasberg, "Suggested formulae for calculating auditory-filter bandwidths and excitation patterns," *The journal of the acoustical society of America*, vol. 74, no. 3, pp. 750–753, 1983.

[14] H. von Helmholtz, *On the Sensations of Tone as a Physiological Basis for the Theory of Music*. Longmans, Green, 1877.

[15] R. Plomp and W. J. M. Levelt, "Tonal consonance and critical bandwidth," *The Journal of the Acoustical Society of America*, vol. 38, no. 4, pp. 548–560, 1965.

[16] A. Kameoka and M. Kuriyagawa, "Consonance theory part i: Consonance of dyads," *The Journal of the Acoustical Society of America*, vol. 45, no. 6, pp. 1451–1459, 1969.

[17] W. A. Sethares, "Local consonance and the relationship between timbre and scale," *The Journal of the Acoustical Society of America*, vol. 94, no. 3, pp. 1218–1228, 1993.

[18] R. Parncutt, "Parncutt's implementation of hutchinson & knopoff (1978)," 2015.

[19] W. Hutchinson and L. Knopoff, "The acoustic component of western consonance," *Journal of New Music Research*, vol. 7, no. 1, pp. 1–29, 1978.

[20] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and models*. Springer Science & Business Media, 2013, vol. 22, ch. 11.

[21] A. Kameoka and M. Kuriyagawa, "Consonance theory part ii: Consonance of complex tones and its calculation method," *The Journal of the Acoustical Society of America*, vol. 45, no. 6, pp. 1460–1469, 1969.

[22] W. A. Sethares, "Adaptive tunings for musical scales," *The Journal of the Acoustical Society of America*, vol. 96, no. 1, pp. 10–18, 1994.

[23] ——, "Real-time adaptive tunings using max," *Journal of New Music Research*, vol. 31, no. 4, pp. 347–355, 2002.

[24] ——, *Tuning, timbre, spectrum, scale*. Springer Science & Business Media, 2005.

[25] A. T. Porres and J. Manzolli, "A roughness mmdel in pd for an adaptive tuning patch controlled by antennas," in *Pure Data Convention 2007*, 2007.

[26] R. Gebhardt, M. Davies, and B. Seeber, "Harmonic mixing based on roughness and pitch commonality," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2015.

[27] ——, "Psychoacoustic approaches for harmonic music mixing," *Applied Sciences*, vol. 6, no. 5, p. 123, 2016.

[28] P. N. Vassilakis and K. Fitz, "Sra: A web-based research tool for spectral and roughness analysis of sound signals," in *Proceedings of the Sound and Music Computing Conference (SMC)*, 2007, pp. 319–325.

[29] J. MacCallum and A. Einbond, "Real-time analysis of sensory dissonance," in *Computer Music Modeling and Retrieval. Sense of Sounds. 4th International Symposium, CMMR 2007, Copenhagen, Denmark, August 2007, Revised Papers*. Springer, 2008, pp. 203–211.

[30] J. Villegas and M. Cohen, "Roughometer: Realtime roughness calculation and profiling," in *Audio Engineering Society Convention 125 (AES)*. Audio Engineering Society, 2008.

[31] H. Traunmüller, "Analytical expressions for the tonotopic sensory scale," *The Journal of the Acoustical Society of America*, vol. 88, no. 1, pp. 97–100, 1990.

[32] M. Lagrange and S. Marchand, "Real-time additive synthesis of sound by taking advantage of psychoacoustics," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2001.

[33] G. Garcia, "Data compression of sinusoidal modeling parameters based on psychoacoustic masking," in *Proceedings of the International Computer Music Conference (ICMC)*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 1999.

[34] B. Hansen, "Spatial utilization of sensory dissonance and the creation of sonic sculpture," in *Proceedings of the International Computer Music Conference (ICMC)*, 2014.

[35] E. Molina, A. M. Barbancho, L. J. Tardón, and I. Barbancho, "Dissonance reduction in polyphonic audio using harmonic reorganization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 2, pp. 325–334, 2013.

[36] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[37] M. Klingbeil, "Software for spectral analysis, editing, and synthesis," in *Proceedings of the International Computer Music Conference (ICMC)*, 2005.

[38] T. Carpentier, M. Noisternig, and O. Warusfel, "Twenty years of ircam spat: looking back, looking forward," in *41st International Computer Music Conference (ICMC)*, 2015, pp. 270–277.

[39] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, no. 6, pp. 456–466, 1997.

[40] H. Fletcher and W. A. Munson, "Loudness, its definition, measurement and calculation," *Bell System Technical Journal*, vol. 12, no. 4, pp. 377–430, 1933.

[41] B. M. Hansen, *Part One: Compositional Utilization of Sensory Dissonance. Part Two: A Portfolio of Compositions*. University of California, Santa Barbara, 2013.

[42] J. Driedger, T. Prätzlich, and M. Müller, "Let it bee-towards nmf-inspired audio mosaicing." in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 350–356.

[43] F.-R. Stöter, M. Schoeffler, B. Edler, and J. Herre, "Human ability of counting the number of instruments in polyphonic music," *Proceedings of Meetings on Acoustics*, vol. 19, no. 1, 2013.

[44] M. Schoeffler, F.-R. Stöter, H. Bayerlein, B. Edler, and J. Herre, "An experiment about estimating the number of instruments in polyphonic music: A comparison between internet and laboratory results." in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 389–394.

[45] E. Terhardt, "Gestalt principles and music perception," in *Auditory processing of complex sounds*. Routledge, 2016, pp. 157–166.

[46] F.-R. Stöter, S. Bayer, and B. Edler, "Unison source separation." in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2014, pp. 235–241.

[47] F.-R. Stöter, A. Liutkus, R. Badeau, B. Edler, and P. Magron, "Common fate model for unison source separation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 126–130.

[48] M. Zhang, M. Bocko, and J. Beauchamp, "Temporal analysis, manipulation, and resynthesis of musical vibrato," *Proceedings of Meetings on Acoustics*, vol. 22, no. 1, 2014.

[49] ——, "Measurement and analysis of musical vibrato parameters," *Proceedings of Meetings on Acoustics*, vol. 23, no. 1, 2015.

[50] A. Roebel, S. Maller, and J. Contreras, "Transforming vibrato extent in monophonic sounds," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2011.

[51] X. Serra, "Musical sound modeling with sinusoids plus noise," in *Musical signal processing*. Routledge, 2013, pp. 91–122.

[52] M. S. Puckette, *The Theory and Technique of Electronic Music*. World Scientific Publ., 2007.

[53] J. Hyrkas, "Real-time implementation of vibrato transfer as an audio effect," in *Proceedings of the International Computer Music Conference (ICMC)*, 2025.

[54] N. Lederer, "The preservation and restoration of sound recordings," *The Journal of Library History*, 1982.

[55] T. H. Park, J. Biguenet, Z. Li, C. Richardson, and T. Scharr, "Feature modulation synthesis (fms)," in *Proceedings of the International Computer Music Conference (ICMC)*. International Computer Music Association, 2007, pp. 368–372.

[56] J. C. R. Licklider, J. Webster, and J. Hedlun, "On the frequency limits of binaural beats," *The journal of the acoustical society of america*, vol. 22, no. 4, pp. 468–473, 1950.

[57] S. Sudre, R. Kronland-Martinet, L. Petit, J. Rozé, S. Ystad, and M. Aramaki, "A new perspective on binaural beats: Investigating the effects of spatially moving sounds on human mental states," *PloS one*, vol. 19, no. 7, p. e0306427, 2024.