Algorithms for Roughness Control Using Frequency Shifting and Attenuation of Partials in Audio

Jeremy Hyrkas*

University of California San Diego
jhyrkas@ucsd.edu

Abstract. Though synthesis algorithms frequently use parameters to change the produced sound, it is not always the case that these parameters have a direct (or intuitive) correlation to the change made to the perceptual attributes—the more meaningful sound descriptors for the listener and/or musician. In this work we explore two strategies by which a perceptual descriptor, roughness, can be parameterized directly on a scale, much like how interactive sound allows for control of pitch and/or loudness. Here, roughness (often tied to dissonance) is controlled by changing either the frequency or amplitude of partials that lie within a critical band. Audio examples are provided to demonstrate use in audio mixing, sound (re)synthesis and audio effects, with two implementations made available: one for offline use and another for real-time interactive synthesis using Max/MSP.

Keywords: auditory roughness, additive synthesis, assistive audio production

1 Introduction

This work presents two algorithms for controlling the roughness of sound by adjusting the frequency or amplitude of individual partials or harmonics of the sound. Auditory roughness is a perceptual feature of sound that is often linked with an experience of dissonance, making it a particularly salient sonic parameter that a composer may want to manipulate. Roughness has historically been studied in relation to musical consonance with regards to interval choice and tuning. Consequently, previous approaches that manipulate the roughness of sound often utilize pitch shifting. The approaches described here avoid pitch shifting, favoring subtle changes to the spectrum to control roughness while maintaining as much of the original timbre as possible. A brief history of roughness and musical applications is presented in Section 2. Algorithms for roughness control are described generally in Section 3. Finally, implementations for audio files and additive synthesis are presented in Section 4. Implementations of the algorithms and audio examples are available online.

* Special thanks to Miller Puckette and Tamara Smyth for advising portions of this project.

2 Auditory roughness, musical consonance and tuning

Auditory roughness is a psychoacoustic and physical phenomena that occurs when sinusoidal components of sound fall within the critical band of the ear. Sinusoidal components that are very close in frequency (i.e. separated by less than 10 Hz), are heard as a single partial with slow beating, a special case of amplitude modulation where the modulating sinusoid (the sound's amplitude envelope) is sufficiently low in frequency that the sound is brought in and out of prominence on a perceptible time scale. While such sounds are generally considered to be *consonant*, increasing the modulation frequency or, equivalently, the frequency difference between the two components, increases the rate of beating, eventually leading to the listener being no longer able to track the beats; the tone takes on a more steady amplitude but with a distinct quality known as *roughness*, an attribute often associated with *dissonance*. Increasing the modulation frequency still, so that the frequency separation approaches the critical bandwidth, the listener begins to recognize the sound as separate tones, at which point discomfort decreases and eventually disappears. Below, we briefly present the history of the study of auditory roughness, its relation to musical consonance and its use in music systems.

2.1 Psychoacoustic models of roughness

Auditory roughness relates the experience of dissonance to the presence of sound partials that fall within a critical band of the human ear. Theories relating musical consonance and distance 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 [1].

Plomp and Levelt tested the perception of dissonance and proposed a general model of dissonance of sinusoidal tones based on critical bandwidth [2]. The resulting data informed a model by Sethares [6] who calculates the dissonance between two partials with frequencies f_1 and f_2 having amplitudes a_1 and a_2, respectively, as

$$r(f_1, a_1, f_2, a_2) = a_1 a_2 \cdot [e^{-b_1 sx} - e^{-b_2 sx}] \,, \tag{1}$$

where $b_1 = 3.5$ and $b_2 = 5.75$ are chosen by fitting sx to Plomp and Levelt's model.

In (1), $x = f_2 - f_1$ (where $f_1 < f_2$) is the frequency difference between the two partials and s scales the frequency difference to fit Plomp and Levelt's standard curve, which was originally plotted with respect to critical bandwidth. The frequency difference is scaled by

$$s = \frac{d^*}{s_1 f_1 + s_2} \,, \tag{2}$$

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. Equation 1 accounts for partials having different amplitudes, so softer components contribute less to dissonance. Figure 1 demonstrates how (1) varies across the frequency domain. We will later use (1) to calculate the roughness of sound partials and choose new frequencies to reduce roughness.

Kameoka and Kuriyagawa extended Plomp and Levelt's experiments and investigated the role of masking [3]. The perception of dissonance when two partials have different amplitudes was found to differ from the pattern found in masking curves. While

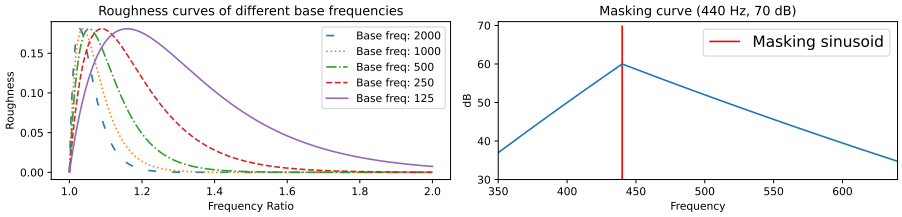

Fig. 1. Left: the roughness of two partials with equal amplitude by their frequency ratio (1). **Right**: a plot of (3), a simple model for estimating the masking curve of a tone given its frequency and sound pressure level in dB. Tones on or below the sloped lines are masked by the original.

a louder sinusoid will more easily mask a quieter sinusoid of higher frequency, pairs of sinusoids were found to be more dissonant when the tone with lower frequency was louder than the tone of higher frequency. Nevertheless, perceived dissonance dropped when one was completely masked by the other, suggesting that masking plays a role in perceived dissonance. A simple model for estimating the masking curve of a partial based on difference in Barks [8] can be computed as

$$mask(x|f, dB) = \left\{ \begin{array}{l} dB - 10 - 27\left[B_{Hz}(f) - B_{Hz}(x)\right], \text{if } x < f \\ dB - 10 - 15\left[B_{Hz}(x) - B_{Hz}(f)\right], \text{if } x \geq f \end{array} \right\}, \quad (3)$$

which we will later use to computationally control roughness by amplitude changes informed by masking. B_{Hz} converts from Hz to Bark using Traunmüller's model [4]. A plot of this masking curve is shown on the right side of Figure 1.

Concepts of consonance and dissonance have many definitions in the context of music. The sensation caused by beating partials in a critical band, which can also be described as the presence of amplitude modulation within a critical band, is now referred to as *roughness* in psychoacoustics to disambiguate it from compositional definitions. Based on the work of Plomp and Levelt, more sophisticated models of roughness have been developed. Sethares's model of Plomp and Levelt accounts for the amplitude of partials (see (1)). Vassilakis [14] additionally accounted for the role of amplitude modulation in an extension to (1) [1]. The roughness of a signal is usually defined as the sum of roughness between all pairs of partials.

2.2 Applications to tuning, mixing and synthesis

While roughness can be measured outside the context of musical applications, it has been supposed to be related to musical consonance [1] and has been used as a metric in tuning, mixing and composition. Sethares developed the Adaptive Tuning algorithm, which adjusts the fundamental frequencies of notes based on their spectra to minimize

[1] Vassilakis's model more thoroughly accounts for the role of loudness and amplitude modulation to roughness. However, this equation is more complex and was not found to improve synthesis in this study, so the simpler model from Sethares is used.

the expected roughness of the sound. Sethares's algorithm can be approximated in real-time on spectra that are known ahead of time [9] or computed exactly on mixtures of spectra in the full implementation [10]. Adaptive Tuning is based on minimizing (1) with respect to fundamental frequency (as opposed to individual partial frequency) and has been used in other musical contexts (for example, to control the pitch of a Theremin implementation in real-time using Pure Data [13]).

The roughness of an audio mix can be a useful metric to measure and control for in sound engineering. Vassilakis used an extension of (1) to analyze and annotate the roughness of sound files [14]. Real-time roughness estimators have been implemented in Pure Data [15][16], each of which use sinusoidal modeling to estimate partials of an incoming mix to determine the roughness. While the ability to analyze roughness in real-time can be helpful, these algorithms crucially will have difficulty detecting sinusoidal peaks that are nearby (i.e., less than 20 Hz difference) using solely FFT-based analyses. Vassilakis's software, on the other hand, uses frequency reassignment methods to obtain finer resolution of nearby partials.

The roughness of a sound can be used as a parameter to be increased or reduced in some synthesis or resynthesis methods. Molina et al. modeled audio signals of chords using sinusoid-plus-residual analysis [5] and reduced beating partials in resynthesis by forcing partial frequencies be an integer multiple of one of the fundamental frequencies of the chord [18]. Roughness was found to be the most impactful feature when pitch shifting one track to be consonant with another in the context of DJing [19][20]. When synthesizing impact sounds using additive synthesis, roughness has been used as a parameter to create convincing sound examples [17]. Finally, Park et al. created software to manipulate psychoacoustic features such as spectral slope or inharmonicity when composing using sound material [12]; auditory roughness control was not implemented in the software but may be another candidate for compositional control.

3 Roughness control by individual partial adjustment

Previous approaches to roughness control have focused on changing fundamental frequencies (and therefore all partials) of notes [9], pitch shifting audio tracks [20], or quantizing partial frequencies to strict integer harmonics [18]. An unexplored approach is the selective adjustment of partials independent of tuning to control roughness while leaving the majority of the signal intact. Such an approach gives the composer options for reducing or increasing roughness in subtle ways that preserve the essence of the original material as much as possible. The methods developed here explore this approach in both a real-time and offline process. Real-time methods change either the frequency or amplitude of additive synthesis parameters in the Max/MSP computer music environment, while offline approaches selectively filter and resynthesize certain partials in audio files while preserving the remaining portions of audio, which may produce higher fidelity output than methods that use sinusoidal resynthesis [18]. Section 3.1 presents an algorithm for frequency adjustment and Section 3.2 presents an algorithm for amplitude adjustment. Offline and online implementations are described in Section 4.

3.1 Changing roughness by partial tuning

Here we define *frequency bashing*, a greedy algorithm for reducing or increasing roughness by changing the frequency of partials that lie within a critical band. Pairs of partials that overlap in time and cause roughness are selected and iteratively changed to adjust roughness. 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.

In the algorithm, partials pairs are analyzed for their contribution to the overall roughness of the sound using (1), with the most rough partial pairs processed first. The quieter partial has its frequency changed to be the frequency that either minimizes or maximizes roughness with respect to the louder partial within a predefined distance. We restrict movement to a specified range in Barks due to the the shape of (1), which would otherwise cause partials to always move to an identical frequency when bashing for minimized roughness; we also aim to keep partials within their original critical band to avoid drastically altering the nature of the original sound. Based on the maximum dissonance of (1), distance in Barks is typically set to between 0.05 and 0.4 in our experiments, but the distance is exposed as a user parameter.

Frequency bashing for consonance is computed mathematically as

$$f_2^* = \underset{f_{\min} \le f^* \le f_{\max}}{\arg\min} \; r(f_1, a_1, f^*, a_2) \tag{4}$$

with constraints defined as

$$f_{\min}, f_{\max} = Hz_B(B_{Hz}(f_1) + B_L), Hz_B(B_{Hz}(f_1) + B_H) \; . \tag{5}$$

Hz_B and B_{Hz} convert from Bark to Hz and vice versa [4], and B_L and B_H define the allowable distance in Barks. In (4) and (5), f_1 is the 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 (4) is replaced by argmax. Equation 5 assumes $f_1 < f_2$; when $f_1 > f_2$ the Bark range is defined below f_1 instead of above.

Figure 2 shows potential adjustments for sinusoid pairs. Note that for partials with identical differences in frequency, their position along the roughness curve and solutions for maximum consonance and dissonance will change depending on their location in the frequency domain. The partial pairs depicted (440 Hz and 470 Hz versus 880 Hz and 910 Hz) have unequal differences in their maximally consonant and dissonant solutions, even though they begin with the same difference of 30 Hz per pair.

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. However, consonance and dissonance as defined by roughness models are disregarded. In Figure 2, hard-bashing partials to have a difference of 3 Hz maintains the equal difference between partial pairs after bashing and both new frequencies result in lower roughness. However, they now fall on different positions along the roughness curve due to their different critical bandwidths.

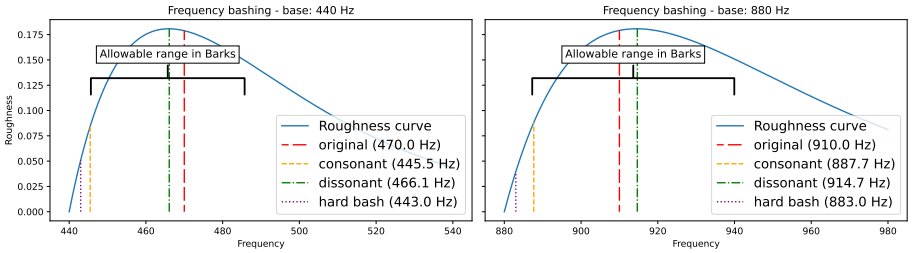

Fig. 2. Frequency bashing two pairs of partials with frequencies (440 Hz, 470 Hz) on the left and (880 Hz, 910 Hz) on the right. The partials with higher frequencies are adjusted. The original frequency of the higher partial is depicted in the long-dashed red line, the most consonant in the specified 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 the partial to a difference of 3 Hz is shown in the dotted purple line. Hard-bashing ignores the roughness curve and allowable range.

Frequency bashing is implemented as described for offline processing of audio files, with further modifications described in Section 4.1. Simplifications made to this algorithm for real-time implementation are described in Section 4.2.

3.2 Changing roughness by amplitude adjustment

Another approach to roughness control is changing the amplitude of partials, as the absolute and relative loudness of partials contributes to roughness. A simple technique would be to lower the amplitude of the quieter partial in a pair that contributes roughness to the sound. However, lowering the amplitude of a partial will reduce the power of the signal. Instead, the quieter partial must 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). The algorithm maximally adjusts amplitudes so that the quieter partial is fully masked by the louder partial, as masking plays a role in perceived roughness [3]. When adjusting the amplitudes of a pair of partials to reduce roughness, the resulting difference in dB (Δ_{dB}) should be the whacking percentage of the masking threshold specified by (3); additionally, the power of the signal should be retained. If a_1 is the amplitude of the louder partial and a_2 the amplitude of the quieter partial, these constraints are specified as

$$20\log_{10}(a_1^*) - 20\log_{10}(a_2^*) = \Delta_{dB}\ , (a_1)^2 + (a_2)^2 = (a_1^*)^2 + (a_2^*)^2 \tag{6}$$

respectively, where a_1^* and a_2^* are the new amplitudes of the partials. Solving the system of equations algebraically leads to the solutions:

$$a_2^* = \sqrt{\frac{(a_1)^2 + (a_2)^2}{1 + 10^{\frac{\Delta_{dB}}{10}}}},\ a_1^* = \sqrt{(a_1)^2 + (a_2)^2 - (a_2^*)^2}\ . \tag{7}$$

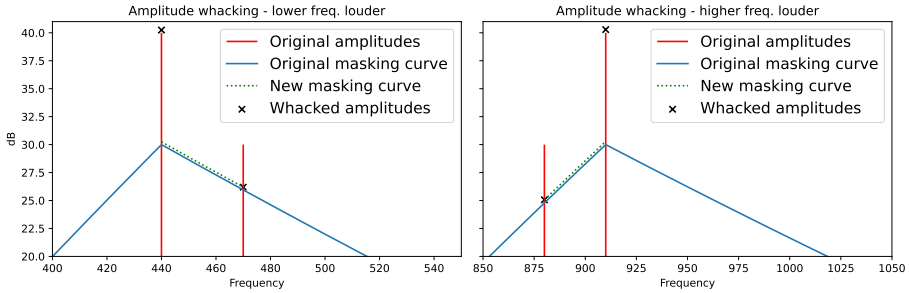

Fig. 3. Amplitude whacking pairs of partials with frequencies (440 Hz, 470 Hz) on the left and (880 Hz, 910 Hz) on the right. Original amplitudes are shown as red solid lines, with whacked amplitudes depicted as black X's. Despite each pair being equidistant in Hz, the example on the right requires more attenuation of the quieter partial due to the asymmetry of the masking curve.

Amplitude whacking 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. Figure 3 shows potential adjustments for amplitudes of partial pairs to reduce the roughness of a sound. Offline and real-time implementations of the algorithm are further described in Sections 4.1 and 4.2, respectively.

4 Algorithm Implementations

The algorithms for frequency bashing and amplitude whacking can be performed offline on mixes of audio files or in real-time in additive synthesis. Each approach requires certain tweaks and optimizations. The implementations are described below, followed by a discussion of potential applications with accompanying examples available online.

4.1 Offline implementations on audio files

Auditory roughness occurs most when partials fall very close to each other (i.e., on the order of 20-40 Hz difference). Finding partials with such resolution in a single audio file is difficult using only FFT-based methods [14], and isolating one partial in a digital filter without affecting the nearby partial also presents issues. Taking these limitations into account, and following previous application of roughness to combinations of sounds [9][18][19], we use the algorithms defined in Section 3 to control the roughness of audio files that are to be combined in a mix.

Given a collection of audio files, each file is analyzed using sinusoidal modeling [5] to find the top N partials of each frame of audio. N can be a small number (i.e., on the order of 10) because the sinusoidal tracks will not be used for resynthesis and are instead used to identify the most prominent partials at a given time. When a partial from signal x_i and a partial from signal x_j overlap in time and cause roughness, the partials are collected as candidates for frequency bashing or amplitude whacking with new frequencies or amplitudes determined using (4) and (7).

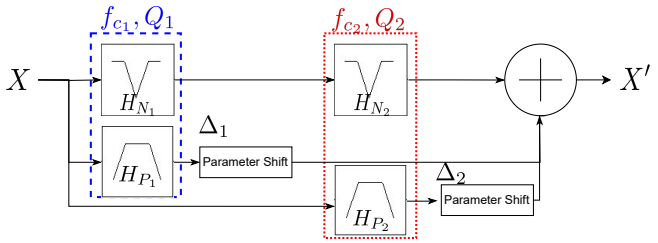

Fig. 4. Partials of a signal x are adjusted by removing them using notch filters, and in parallel, isolating them from x using amplitude-complimentary peaking bandpass filters. Complimentary filters share the same center frequency f_{c_k} and quality factor Q_k. The output of the peaking filters are processed to adjust either the frequency or the amplitude by some amount Δ_k.

To adjust partials of a signal x, the offending partials are removed from x and in parallel isolated, altered, and added back in. The mean frequency and frequency range of a partial are used to set the center frequency f_c and quality factor Q of an amplitude-complimentary pair of H_N, a notch filter, and H_P, a peaking bandpass filter, such that

$$H_N(f_c, Q) \cdot X + H_P(f_c, Q) \cdot X = X \ . \tag{8}$$

If a signal has k partials to be adjusted, the output signal x' will be computed as

$$X' = X \cdot \prod_{i=1}^{k} H_{N_i}(f_{c_i}, Q_i) + \sum_{i=1}^{k} \Delta_i(H_{P_i}(f_{c_i}, Q_i) \cdot X) \ , \tag{9}$$

where Δ_i changes either the amplitude or frequency of the partial isolated in the peaking filter. H_N and H_P are standard second-order IIR filters defined in [7] and implemented as the *iirnotch* and *iirpeak* functions in MATLAB and scipy.

Figure 4 depicts the audio processing of partials. Forward and backward filtering is used for zero-phase filtering. When frequency bashing, the output of a peaking filter is frequency shifted using single sideband modulation; when amplitude whacking, gain is applied to the output so that the mean amplitude matches the intended value. The processed signals are then added to the output of the series of notch filters. Additionally, cross-fades are applied so that filtered signals with adjusted partials are only heard during the lifetime of the partial. When partials that cause roughness are not present, the original sound files are used unaltered.

4.2 Real-time implementations for additive synthesis parameters

Frequency bashing and amplitude whacking can be performed more straightforwardly on additive synthesis parameters, as no audio analysis is inherently required. In additive synthesis, frequency and amplitude pairs are used to control an oscillator bank, with each oscillator receiving one frequency and amplitude per synthesis step. Roughness control on additive synthesis parameters can be more easily performed in real-time unlike the previous methods for audio processing, although some simplifications to the algorithms are required to reduce computational load.

Two externals for the Max/MSP computer music environment were created to control roughness of additive synthesis parameters. The externals are control rate objects that take as input a list of frequency/amplitude pairs and output one list of frequencies and one list of amplitudes. In the *basher* object, frequencies are adjusted using (4) while amplitudes are passed through unchanged, while amplitudes are adjusted by the *whacker* object using (7) with the frequencies unchanged. The outputs can be connected to multichannel Max objects for additive synthesis (see the associated code for examples). The input list of sinusoidal parameters can come from an analysis-resynthesis system for sinusoidal modeling [5][11], but composers are free to use any method for generating sinusoidal parameters. Other algorithmic parameters include the Bark range of search and adjustment (see Equation 5), a toggle to change from decreasing roughness to increasing roughness, and the percent of movement from the original sinusoidal parameters to the adjusted parameters. These parameters can be fixed or adjusted by the user algorithmically or manually on-the-fly as a musical effect.

Equation 1 is expensive to compute in real-time for every potential pair-wise combination for every frame. When searching for potential candidates for parameter adjustment in the Max objects, partial pairs are sorted by frequency instead of by overall roughness to reduce computational load. Pairs are processed in ascending order of frequency, short-cutting whenever pairs start to fall outside of the defined Bark range. In many cases there will not be many partials lying within a fraction of a critical band of each other, but in general results may differ from the offline implementation.

When changing amplitudes in Max, the user parameter for whacking amount maximally transfers all power from the quietest partial to the loudest partial to avoid computing (3) at every frame. As a result, the audible effect of the parameter will stop before reaching the maximum value once quieter partials are fully masked. This issue could be avoided by precomputing and quantizing masking curves. Finally, while offline methods change the parameters of partials across time, sinusoidal parameters here are adjusted at every frame in a memoryless fashion. This change reduces computation and computer memory necessary to track previous changes, but makes the objects more susceptible to rapid fluctuation in the case where two nearby partials are nearly the same amplitude and fluctuate between which one is louder.

4.3 Audio examples and observations

Examples of frequency bashing and amplitude whacking on audio files and additive synthesis parameters are available on the associated supplemental website. [2] The basic algorithms are demonstrated on the sinusoid examples in Figures 2 and 3 where the effect of each algorithm and parameter choice is most obvious. More complex examples showcase various use cases of the algorithms in more realistic musical contexts.

The effects of frequency bashing and amplitude whacking on synthesized notes is shown in Figure 5. Each algorithm is 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

[2] `https://jeremyhyrkas.com/cmmr2023`

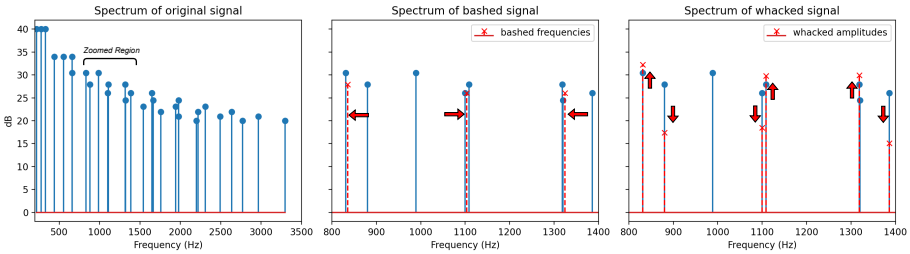

Fig. 5. Left: the spectrum of a major triad of sawtooth waveforms in equal temperament tuning. **Center**: zoomed region of the spectrum after frequency bashing. Three partials have been moved. **Right**: the same zoomed region of the spectrum after amplitude whacking. Three pairs of nearby partials (six partials total) have had their amplitudes adjusted.

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. In contrast, an accompanying audio example demonstrates the chord in just intonation, which an algorithm such as Adaptive Tuning [9] may offer as a solution. The retuned example also contains less roughness than the original but sounds fundamentally different. This example demonstrates the difference in philosophy between our methods for roughness control versus tuning-based approaches, as the goal of our algorithms is to only change the perceived auditory roughness while maintaining as much of the original signal as possible.

More examples of roughness reduction include frequency bashing a slightly detuned major chord and amplitude whacking a horn line. The horn line is depicted in Figure 6, with a spectrogram of the original on the left and a spectrogram of the difference signal after amplitude whacking on the right. This example again 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. Another example demonstrates the effect of filtering a partial that contributes to roughness without resynthesizing it back in with a different frequency or amplitude. This approach will reduce the power of the signal and can cause the sound to feel hollow when too many partials are removed, but can be effective on audio examples where very few partials contribute to roughness. These examples demonstrate a potential use case of reducing roughness when mixing a track without the use of retuning or manually intensive EQing.

Two examples are presented where frequency bashing is used to introduce more roughness into a sound mix than was originally present. A sawtooth major seventh chord is used, as is a recording of a choir singing a chorale. This use case shows frequency bashing as an audio effect that may be useful for a composer who wishes to introduce dissonance without detuning or modulating the entire signal. Finally, an example is presented using hard-bashing where pairs that contribute roughness are moved to be exactly 3 Hz apart from one another. The effect is similar to a tremolo audio effect, but the modulation is only heard when roughness is present and only in some parts of the frequency domain. All examples described can be found on the accompanying website.

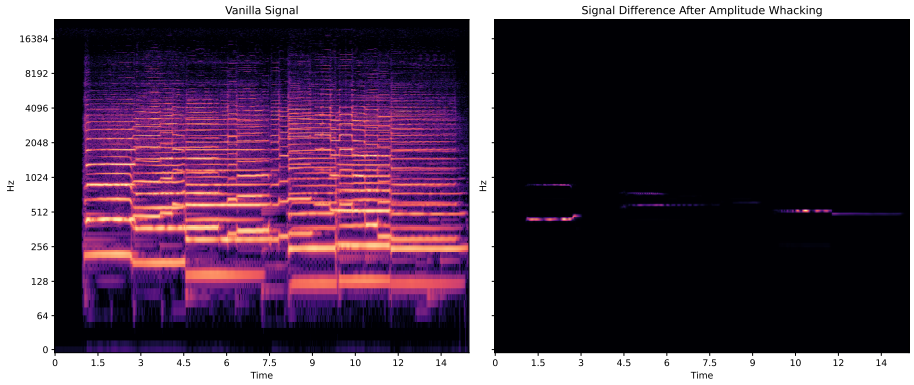

Fig. 6. Left: a spectrogram of a dynamic horn line featuring audio of three players performing. **Right**: the spectral difference of an amplitude whacked version of the horn line and the original. Amplitudes of partials are only adjusted in areas of roughness, after which point the original signals are faded back in. This example is time-varying and demonstrates subtle changes that achieve a reduction in roughness without retuning any notes.

The Max externals described in Section 4.2 are demonstrated in videos using harmonic and inharmonic drones, as well as a dynamic sinusoidal reconstruction of the horn line described previously. The reconstruction requires playback of an offline analysis using the SPEAR modeling software [11]. Preliminary versions of these Max externals that control the amplitude, frequency and spatial panning of partials to increase or reduce auditory roughness were used by the author to create a composition that musically investigates roughness, tuning and listening tests. [3]

5 Conclusion and Future Work

We present two algorithms for controlling auditory roughness by targeted sound partial adjustment. Frequency bashing modifies the frequencies of neighboring partials, while amplitude whacking modifies their amplitude. Considerations are made based on previous work modeling listener perception of roughness, auditory masking, and previous approaches to roughness reduction. Offline implementations of both algorithms are provided for audio files intended to be mixed in time[4], and control-rate objects for additive synthesis are provided for the Max/MSP environment[5]. The accompanying audio examples demonstrate potential use cases in mixing and composition.

The algorithms reduce roughness as calculated by the roughness models by their definition, but listening tests would be beneficial to confirm the intended effect on listeners, as the perception of consonance and dissonance is affected by context in ways not accounted for in these models. Additionally, a VST implementation of the audio

[3] The piece described here was submitted for consideration to CMMR 2023.
[4] `https://github.com/jhyrkas/sms-tools-audio-bashing`
[5] `https://github.com/jhyrkas/basher_max`

processing algorithms would assist sound engineers in incorporating them into their workflow. While these algorithms do not currently work on incoming audio streams, a plug-in implementation may be beneficial for use on processed tracks before final mixing and mastering. Finally, simplifying certain portions of the audio implementations may make them viable for use in real-time applications.

References

1. Helmholtz, H.: On the Sensations of Tone as a Physiological Basis for the Theory of Music. Longmans, Green and Co. (1877)
2. Plomp, R., Levelt, W.: Tonal Consonance and Critical Bandwidth. In: The Journal of the Acoustical Society of America, 38 #4 (1965), 548–560
3. Kameoka, A., Kuriyagawa, M.: Consonance Theory Part I: Consonance of Dyads. In: The Journal of the Acoustical Society of America, 45 #6 (1969), 1451–1459
4. Traunmüller, H.: Analytical Expressions for the Tonotopic Sensory Scale. In: The Journal of the Acoustical Society of America, 88 #1 (1990), 97–100
5. Serra, X.: A System for Sound Analysis/Transformation/Synthesis Based on a Deterministic Plus Stochastic Decomposition. Dissertation, Stanford University (1990)
6. Sethares, W.: Local Consonance and the Relationship Between Timbre and Scale. In: The Journal of the Acoustical Society of America, 94 #3 (1993), 1218–1228
7. Orfanidis, S. J.: Introduction to Signal Processing. Prentice Hall Upper Saddle River, 1996
8. Lagrange, M., Marchand, S.: Real-time Additive Synthesis of Sound by Taking Advantage of Psychoacoustics. In: Proc. of the International Conference on Digital Audio Effects (DAFx) (2001)
9. Sethares, W.: Real-time Adaptive Tunings Using Max. In: The Journal of New Music Research, 31 #4 (2002), 347–355
10. Sethares, W.: Tuning, Timbre, Spectrum, Scale. Springer Science & Business Media, 2005
11. Klingbeil, M.: Software for spectral analysis, editing, and synthesis. In: Proc. of the International Computer Music Conference (ICMC) (2005)
12. Park, T. H., Biguenet, J., Li, Z., Richardson, C., Scharr, T.: Feature Modulation Synthesis (FMS). In: Proc. of the International Computer Music Conference (ICMC) (2007)
13. Porres, A., Manzolli, J.: A Roughness Model in Pd for an Adaptive Tuning Patch Controlled by Antennas. In: Proc. of the Pure Data Convention (2007)
14. Vassilakis, P., Fitz, K.: SRA: A Web-based Research Tool for Spectral and Roughness Analysis of Sound Signals. In: Proc. of the Sound and Music Computing Conference (SMC) (2007)
15. MacCallum, J. and Einbond, A.: 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 (2007)
16. Villegas, J., Cohen, M.: "Roughometer": Realtime Roughness Calculation and Profiling. In: Proc. of the Audio Engineering Society Convention 125 (AES) (2008)
17. Aramaki, M., Gondre, C., Kronland-Martinet, R., Voinier, T., Ystad, S.: Thinking the Sounds: An Intuitive Control of an Impact Sound Synthesizer. In: Proc. of the 15th International Conference on Auditory Display (ICAD) (2009)
18. Molina, E., Barbancho, A., Tardón, L., Barbancho, I.: Dissonance Reduction In Polyphonic Audio Using Harmonic Reorganization. In: The IEEE/ACM Transactions on Audio, Speech, and Language Processing, 22 #2 (2013), 325–334
19. Gebhardt, R., Davies, M., Seeber, B.: Harmonic Mixing Based on Roughness and Pitch Commonality. In: Proc. of the International Conference on Digital Audio Effects (DAFx) (2015)
20. Gebhardt, R., Davies, M., Seeber, B.: Psychoacoustic Approaches for Harmonic Music Mixing. In: The Journal of Applied Sciences, 6 #5 (2016), 123–143