Audio Engineering Society

Convention Paper

Presented at the 141st Convention
2016 September 29–October 2 Los Angeles, USA

An Efficient Algorithm for Clipping Detection and Declipping Audio

Christopher Laguna[1], Alexander Lerch[1]

[1] Center For Music Technology, Georgia Institute of Technology, Atlanta, Georgia, 30332, United States of America
claguna3@gatech.edu, alexander.lerch@gatech.edu

ABSTRACT

We present an algorithm for end to end declipping, which includes clipping detection and the replacement of clipped samples. To detect regions of clipping, we analyze the signal's amplitude histogram and the shape of the signal in the time-domain. The sample replacement algorithm uses a two-pass approach: short regions of clipping are replaced in the time-domain and long regions of clipping are replaced in the frequency-domain. The algorithm is robust against different types of clipping and is efficient compared to existing approaches. The algorithm has been implemented in an open source JavaScript client-side web application. Clipping detection is shown to give an f-measure of 0.92 and is robust to the clipping level.

1. INTRODUCTION

Audio quality enhancement is a topic of growing relevance. Traditionally, audio quality enhancement addresses restoration of historical recordings corrupted by artifacts of old recording technology (e.g., tape hiss in Digital Audio Tape) [1]. Recently, however, the need to enhance the quality of mobile phone recordings has emerged. Mobile phone recordings often contain poor quality audio; this is prevalent in videos of live music concerts taken by audience members. One frequently occurring artifact in these mobile phone recordings is clipping.

Clipping occurs when an audio signal's level rises above a microphone's or AD converter's maximum input level. As more audio and video recordings are being taken on mobile devices (sometimes in high sound level conditions such as live concerts), clipping has become an issue that users encounter frequently. Although choosing correct microphone placement and reasonable input levels can often prevent clipping, there is a need to improve the quality of existing recordings, especially when the damaged recordings are irreplaceable.

Declipping is the process of removing the perceptual effect of clipping from clipped audio. Declipping is a two-step process: 1) detection of the sample indices in the signal where clipping occurs (clipping detection), and 2) replacement of the signal values at these indices with estimates that eliminate or reduce the perception of clipping without introducing new artifacts (sample replacement).

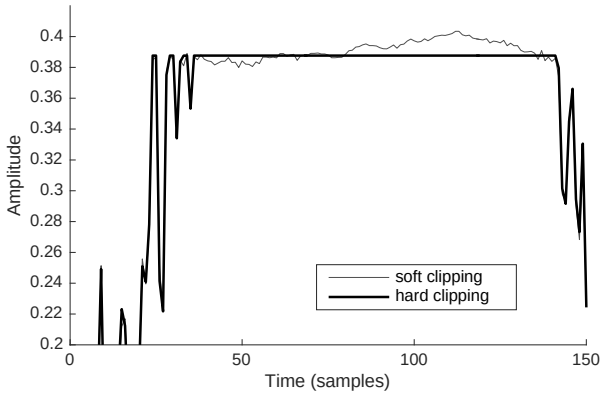

Figure 1: Soft clipping versus hard clipping.

This paper presents a novel approach to declipping audio. The proposed algorithm is computationally efficient compared to existing algorithms and has been implemented in JavaScript by the authors [2].

The remainder of this paper is structured as follows: Section 2 explains some properties of clipping, Section 3 describes existing declipping algorithms, Section 4 describes the proposed declipping algorithm, and Section 5 presents our evaluation and results.

2. PROPERTIES OF CLIPPING

There are two types of clipping: hard clipping and soft clipping. In hard clipping, any signal values above the clipping threshold are set to the clipping level. While easy to detect, this type of clipping is uncommon and usually occurs in the digital domain. In soft clipping, signal values above the clipping threshold are driven near the clipping threshold. If clipping is modeled as a system mapping the non-clipped input signal to the clipped output signal, the relation between input and output is nonlinear and sometimes time-variant. Figure 1 shows the difference between hard clipping and soft clipping.

Clipping modifies the signal's amplitude distribution. Non-clipped audio signals have predictable amplitude distributions: the distributions are high towards the mean and near-monotonically decrease towards both ends. Since hard and soft clipping both drive high amplitudes down to near the clipping level, a clipped signal will have a probability distribution where amplitudes near the clipping level have an unnaturally high probability. Figure 2 shows the amplitude

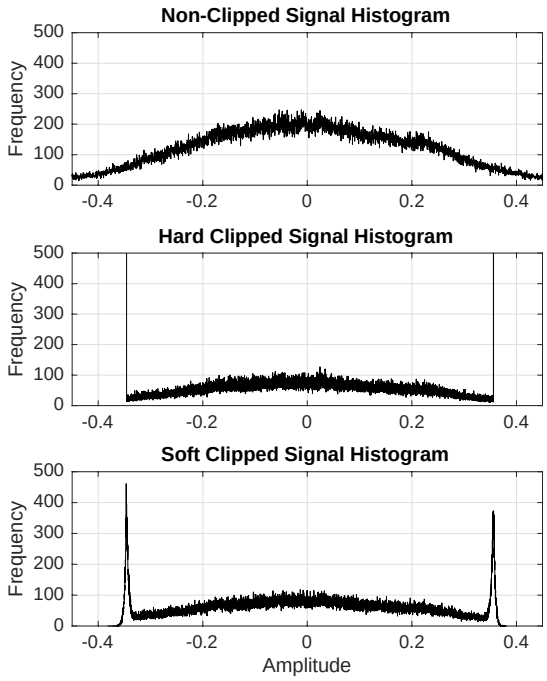

Figure 2: Histograms of clipped and non-clipped signals.

histogram of a signal before clipping, after hard clipping, and after soft clipping. Notice that in hard clipping, all of the amplitudes above the clipping level get placed in a single bin (the bin containing the clipping level), while in soft clipping, the amplitudes above the clipping level get placed into a group of bins, creating a "bump" in the histogram.

Clipping modifies the frequency content of a signal. A clipped sinusoidal can be modeled as a sinusoidal plus harmonics occurring at integer multiples of the frequency of the sinusoidal. In this simple case, clipping has not modified the frequency content of the signal below the fundamental frequency. Signals with more than one frequency component are less easy to analyze due to intermodulation of frequency components, which generates inharmonic components at frequencies below either component. Still, for signals with high bandwidth, most of the distortion produced by clipping occurs at and above the original frequency content of the signal. Clipping is most perceptible when the distortion occurs in the frequency bands where human hearing is most sensitive.

The phase distortion caused by clipping is not very perceptible. We performed an informal listening test to study the perceptual effects of the phase distortion of clipping. We first clipped 7 audio signals such that 25% of all samples clipped. Then, we computed the Discrete Fourier Transform (DFT) of the clipped audio and original audio (in blocks). We then created a reconstructed audio file using the magnitudes of the original audio and the phases of the clipped audio file. The audio files are available online [2]. The reconstructed audio files sound nearly identical to the originals, and we conclude that most of the perceptual effects of clipping are caused by frequency distortion (rather than phase distortion).

3. RELATED WORK

3.1. Clipping Detection

While a fair amount of patents on clipping detection exist, there are relatively few scientific publications on clipping detection. One possible reason for this is that it is difficult to formally evaluate a clipping detection algorithm, as we discuss in the evaluation section.

There are two main approaches to detecting clipping: histogram analysis [3,4] and time-domain analysis [5]. Histogram-based approaches attempt to locate abnormalities in the histograms caused by clipping. Aleinik and Matveev detect soft clipping by searching for a large section of consecutive increasing values of histogram bins from middle outward [3]. The algorithm was evaluated by hard clipping an input signal with the clipping level set to 75% of the maximum signal level and simulating soft clipping by low-pass filtering the signal using forward-backward exponential smoothing. The authors report a "False Alarm" rate of 0.096 and a "Right Detection" rate of 0.978. Otani et al. detect clipping by measuring the difference between the clipped signal histogram and models of natural signal histograms based on Laplace and Gamma distributions [4]. Histogram domain methods work well when the signal is clipped uniformly over time, but can fail if the clipping level changes over time or if the number of clipped samples is much lower than the number of total samples.

Time-domain approaches rely on the fact that the slope of the signal during intervals of clipping is relatively flat and/or that clipping results in discontinuities at the endpoints of each clipping interval. Since these approaches analyze the signal sample by sample, they tend to have fine time resolution. However, false positives can occur because non-clipped signals sometimes exhibit the aforementioned characteristic properties of clipping (relatively sharp increase/decrease in slope and sections of flatness). These false positives can be reduced if assumptions can be made about the shape of the signal. Riemer et al. detect clipping by applying a differentiator to the signal and detecting clipping when 1) the derivative is above a certain threshold and 2) the signal amplitude is close to the global maximum or minimum [5]. The derivative threshold is chosen according to the expected frequency content of the signal. Assumptions such as this cannot be made for music, which can contain any audible frequency.

3.2. Sample Replacement

Sample replacement is a popular research topic that spans not only declipping but also many other applications including audio restoration and audio/video streaming (e.g., recovering from packet loss) [1]. Much research focuses on estimating a single 'burst' of unknown samples when there are many known samples on either side of the burst. Approaches include time-domain interpolation [6,8], frequency-domain interpolation [9,11], and sparse reconstruction [12,13].

The most common time-domain interpolation methods model the signal as an autoregressive process and use linear prediction to fill in the burst. Methods differ in how they guarantee that the prediction will be continuous on both sides of the burst. Janssen et al. train a linear predictor on all known samples and the estimates are formed by minimizing the estimation error over all samples in the burst [6]. Esquef et al. train linear predictors on either side of the burst, and the results of forward and backward prediction are crossfaded over the burst [7]. Etter trains linear predictors on either side of the burst, but a single estimation is obtained by creating an objective function that takes both predictors into account [8]. These linear prediction methods tend to work well with bursts that are under 20 milliseconds [8]. However, the order of the autoregressive model highly influences reconstruction accuracy and is difficult to estimate.

Frequency-domain approaches create estimates for time-frequency bins instead of samples. Some

approaches estimate separate autoregressive models for each sub-band of the signal [9,10]. Lagrange et al. handle signals containing vibrato by identifying partials on each side of the burst and training linear predictive models on the partials [11]. Some methods, including Lukin and Todd's approach, estimate tonal and non-tonal components of the signal and treat them separately [10]. Frequency-domain methods are often used to estimate long bursts, where time-domain methods are insufficient. One common challenge with using frequency-domain approaches is that to get a large enough number of time-frequency bins to do proper interpolation requires a large number of known samples on either side of the burst. Locations of clipping are unpredictable, so these methods are difficult to apply to declipping.

Recent approaches to declipping use concepts from compressive sampling [12,13]. Clipped samples can be ignored and the signal can be interpreted as being sampled at non-uniform intervals. Then, the best sparse representation of a signal (in an accordingly sparse basis, such as the DCT) can be obtained using iterative optimization such as orthogonal matching pursuit [14]. These methods are relatively robust to different clipping scenarios, but unfortunately are computationally expensive, usually having a complexity of $O(n^3)$ where n is the block size.

4. PROPOSED METHOD

We present both a clipping detection algorithm and a sample replacement algorithm. The algorithms work for both hard clipped and soft clipped signals. For multichannel audio, each channel is processed independently.

4.1. Clipping Detection

The clipping detection algorithm is divided into two parts: clipping level detection and clipping interval detection. In clipping level detection, we search for positive and negative clipping thresholds by looking for bumps in the clipped signal amplitude histogram (as depicted in Figure 2). Given these clipping levels, we then look for sample-accurate locations of clipping intervals by analyzing the signal in the time-domain.

Clipping level detection occurs in non-overlapping blocks. This allows us to detect time-varying clipping levels.

4.1.1. Clipping Level Detection

First, the amplitude histogram is computed with 6000 equally spaced bins that span the amplitude range of the signal. To remove small, noisy fluctuations in the histogram, we smooth the histogram by low-pass filtering it with a forward-backward exponential smoothing filter. We then compute an adaptive threshold by filtering the smoothed histogram with a forward-backward exponential filter with a significantly lower cut-off frequency. Note that the first pass of filtering aims to smooth the histogram, while the second pass aims to create a very slow-changing threshold.

In order to locate the bumps in the histogram, we compute a novelty function by subtracting the threshold from the smoothed histogram. The novelty function will be above zero at the locations of the bumps. The novelty function might also be slightly above zero at other locations depending on the input signal. We search the outermost 10% of the novelty function and track all intervals of consecutive positive values. These intervals are candidates which might correspond to the bumps. If the candidate does correspond to a bump, then it will have a large area compared to the other candidates. Therefore, a bump is detected when a candidate has an area that is 3 standard deviations above average. The clipping level is determined to be the amplitude corresponding to the innermost bin in the bump.

If neither a positive nor a negative clipping level was found, then the algorithm reports that the signal contains no clipping. This means that this method naturally handles cases where no clipping occurs. Note that histogram normalization is not required because the method only relies on relative values.

4.1.2. Clipping Interval Detection

Given the clipping levels, we then search for the clipping intervals by analyzing the signal in the time-domain. Each local maximum above the clipping level is assigned a clipping interval. The initial endpoints of this interval are both set to the location of the local maximum. We attempt to extend the left endpoint by moving one sample to the left and checking two halting criteria (explained below). If none of the halting criteria are met, we continue to move one sample to the left. The same process is applied to the right endpoint. Any

overlapping intervals are merged after all clipping intervals are computed.

The two halting criteria check whether the clipping interval meets general assumptions about clipping. The first criterion checks whether the absolute value of the derivative of the signal at the current location is above a threshold. The derivative is computed by taking the difference between the current sample and the previous sample (when moving left, the previous sample is located one sample to the right). This criterion checks if the slope is flat enough for the sample to be included in the interval. The second criterion checks whether the current signal value deviates from the average value of the samples within the working interval by more than a threshold. This criterion guards against cases where the signal has a near-flat slope that never changes, such as a very low frequency triangle wave.

To set the thresholds for these criteria, we refer to bumps detected in the histogram during clipping level detection. Both thresholds are set to half of the width (units are amplitude) of each bump. Determining these thresholds based on the bump width is appropriate because the bump width indicates how much the signal deviates during clipping.

4.2. Sample Replacement

An overview of the entire replacement process is shown in Figure 3. To remove clipping, we first divide the clipping intervals into short intervals and long intervals. We interpolate the short intervals in the time-domain using cubic spline interpolation. The signal is then split into a low-frequency band and a high-frequency band. Samples in the high-frequency band are replaced by linearly interpolating time-frequency bin magnitudes. The low-frequency band remains untouched. Finally, we recombine the low-frequency band and the processed high-frequency band.

A frequency-domain approach to declipping presents challenges because frequency transformations require many consecutive samples and there are no guarantees about the number of consecutive non-clipped samples surrounding each clipping interval. On the other hand, time-domain approaches require parametrization, for example, the order of an autoregressive model, which, if set incorrectly, often causes large reconstruction error.

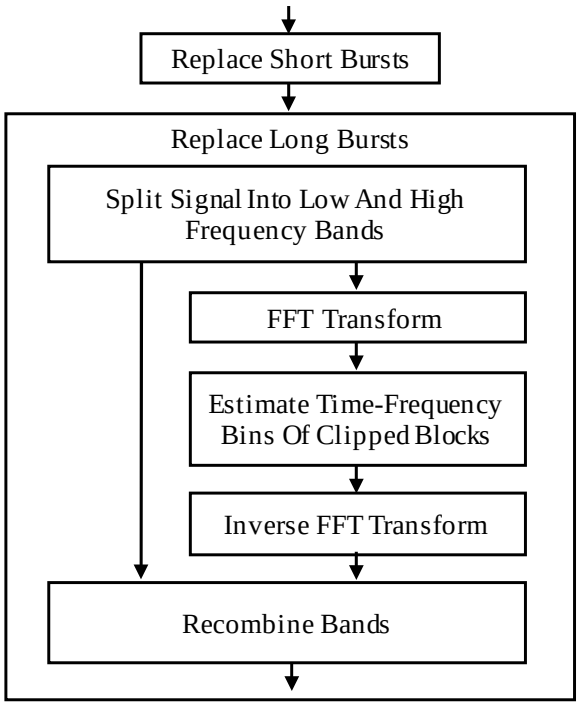

Figure 3: Block Diagram of sample replacement

Since our algorithm must handle any musical signal, it is desirable for our algorithm to be as general as possible. Therefore, we prefer a frequency-domain approach over a time domain approach. We consider sparse reconstruction approaches too computationally expensive for our purposes (see Section 5.3.2); therefore, our approach uses frequency domain analysis.

To facilitate a frequency-domain approach to declipping, we first replace short clipping intervals in the time domain. This results in more consecutive non-clipped samples, which increases the number of locations where clean frequency transforms are possible.

4.2.1. Replacing Short Intervals

We replace short intervals using cubic spline interpolation. We give the cubic spline interpolator at most 20 samples on either side of the clipping interval. If either side of the clipping interval contains less than 3 samples, we do not interpolate.

4.2.2. Replacing Long Intervals

Splitting Signal into Bands

As described in Section 2, the distortion introduced by clipping mostly resides in high frequencies. Thus, we do not process the low frequencies; we split the signal into a low-frequency band (0-100 Hz) and a high-frequency band (100 Hz-Nyquist) and only process the high-frequency band. To split the audio signal into bands, we first create the low-frequency band by applying a low-pass filter to the signal (note that the signal has already had its short clipping intervals replaced). We create the high-frequency band by subtracting the low-frequency band from the original signal. After processing, we combine bands by adding the processed high-frequency band to the low-frequency band.

It is important to note that filtering the signal modifies the locations where clipping occurs: the clipping intervals get smeared out and could be extended by up to the length of the filter's impulse response. With this in mind, we designed an FIR filter with an impulse response length of 25. The filter's cutoff frequency is 100 Hz, and the filter is designed to be as steep as possible given its order. We apply the filter forwards and backwards to ensure that the filter is zero phase. Because the filter is zero phase, the midpoints of the clipping intervals do not change. Since the support of the filter's impulse response spans from sample -25 to 25, we extend the clipping intervals by 25 samples on either side after filtering.

Frequency Magnitude Interpolation

After replacing short intervals, there are generally enough non-clipped samples to analyze the signal in the frequency-domain. However, if we use a standard approach to blocking the signal, there might be very few blocks that contain no clipped samples. Figure 4 illustrates a case where not very many samples are clipped, yet with a desired 50% overlap between blocks, there is only one block that has no clipped samples. However, if we allow block locations to be anywhere and zero pad shorter blocks to a standard FFT size, there are many locations where it is possible to take an FFT.

To replace long intervals in the high-frequency band, we first block the signal with a block size N = 512 and

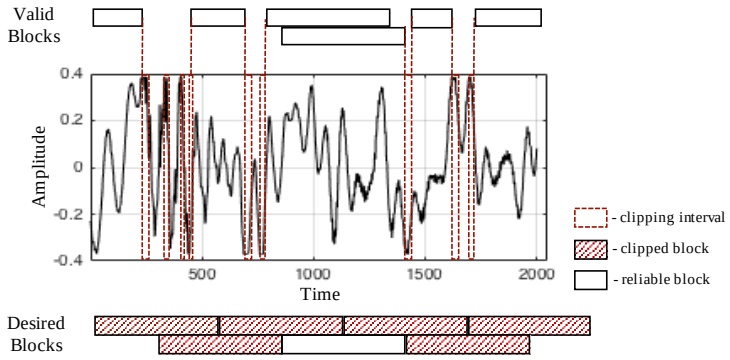

Figure 4: Example of linear interpolation with an overlap of 50%.

75% overlap. We aim to estimate the magnitudes and phases for each block containing any clipped samples. After obtaining the estimates, we can inverse FFT the estimates and overlap-add to reconstruct the time domain signal.

As mentioned in Section 2, clipping has a negligible perceptual effect on the phase content of the signal, so we use the phase of the current clipped block in our reconstruction.

The magnitude of a clipped block is estimated by linearly interpolating between the magnitudes of the two closest reliable blocks. Reliable blocks are located by searching for the nearest interval of at least N / 4 consecutive non-clipped samples. To take the FFT of a reliable block with M < N samples, we first window the reliable block and then zero pad. We normalize the resulting FFT by multiplying by the following scalar:

$$s = e^{log_e(\|w_a\|) - log_e(\|w_b\|)} \quad (1)$$

The L_2-norm is used in this equation; w_a is the N-point window function, w_b is the M-point window function.

There are occasions when linear interpolation of magnitudes causes artifacts. Most notably, overestimation of magnitudes during blocks containing clipped transients can cause the transients to sound tonal, resulting in 'blips' during high hat and snare hits. To reduce this artifact, we upper bound each estimated magnitude by the magnitude of the corresponding bin in the clipped block.

Clipping Detection Results

Algorithm	Clipping Level	Histogram Block Length	Precision	Recall	F-measure
Histogram	95th Percentile	File	0. 937	0. 889	0. 911
Histogram	95th Percentile	3 Seconds	0.937	0.892	0.914
Histogram	90th Percentile	File	0.942	0.840	0.881
Histogram	90th Percentile	3 Seconds	0.947	0.880	0.912
Combined	95th Percentile	File	0.941	0.910	0.925
Combined	95th Percentile	3 Seconds	0.940	0.908	0.922
Combined	90th Percentile	File	0.950	0.902	0.925
Combined	90th Percentile	3 Seconds	0.945	0.862	0.894

Table 1: Clipping detection results.

5. EVALUATION

5.1. Clipping Detection

5.1.1. Evaluation Methodology

Evaluating the detection of soft clipping is problematic because it is difficult to record soft clipping with corresponding sample-accurate ground truths of clipping indices. Existing research either simulates soft clipping in software [3] or does not include a formal evaluation of their algorithm [5]. To properly evaluate with soft clipped signals, a method would be required that is able to record clean audio data and corresponding soft clipped audio time-aligned and phase-aligned to near-sample accuracy in order to reliably annotate clipping interval locations.

For our evaluation, we simulate soft clipping by digitally hard clipping a signal (and saving the clipping locations), and then encoding the clipped signal using a lossy codec. When a hard clipped signal goes through a lossy encoder, the clipped sections of the signal are modified, as these segments of audio are the most difficult to encode (the frequency-domain is least sparse at sharp edges). The soft clipped signal in Figure 1 was generated by encoding the hard clipped signal in Figure 1 with FFmpeg's AAC encoder using a variable bit rate (VBR) quality of 5.

To evaluate, we run our clipping detection on the resulting audio files and take the precision, recall, and f-measure of the clipping indices. We use the dataset published by Homburg et al. for evaluation [15]. This dataset contains 1886 10-second excerpts of songs from 9 musical genres. The audio files are encoded as MPEG-1 layer 3 files at 44.1 kHz/128 kb. This dataset is chosen because it covers a wide range of musical styles.

We run the evaluation with different amounts of clipping, different block sizes, and with different versions of the algorithm. The clipping levels should be chosen such that the amount of clipping is comparable between audio files. Clipping based on a percentage of the maximum amplitude does not normalize for amount of clipping because different signals have different amounts of dynamic variation. Instead, we choose our clipping level based on a percentile of the amplitude distribution in the signal. This guarantees that the same number of samples clip in each audio file. Because the amount of perceptual clipping also depends on the frequency content of the signal, choosing the clipping level based on amplitude percentile does still not guarantee equal amounts of perceptual clipping. Preprocessing such as filtering the input by the inverse spectral envelope might be worth investigating, but was considered excessive for this evaluation.

Running the evaluation with different block sizes allows us to verify that the system can be used when the clipping level is time-variant.

We also compare results using the clipping interval detection (combined) vs. only using clipping level detection (histogram). When only using clipping level

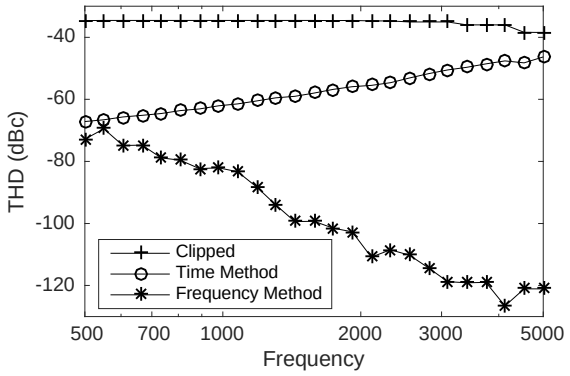

Figure 5: Total harmonic distortion of reconstruction on sinusoidal input.

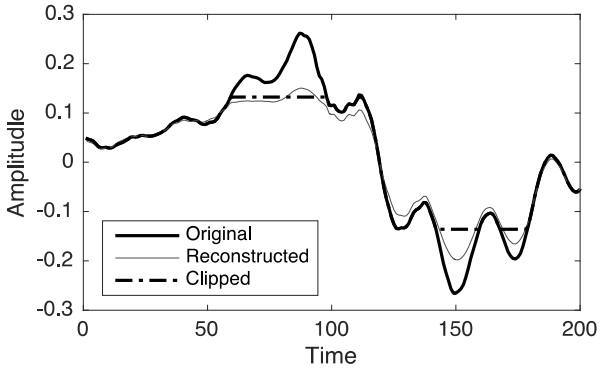

Figure 6: Clean, clipped, and reconstructed speech signal waveforms.

detection, clipping is detected at each sample with an amplitude above the clipping level.

5.1.2. Evaluation Results

Results from our clipping detection evaluation are shown in Table 1. The algorithm performs with an f-measure of 0.91 in most cases.

In all experiments, precision (~0.94) is higher than recall (~0.90). One explanation for this is that we choose the clipping level to be the innermost bin in the bump. It is possible that the amplitudes placed in bins on the inner half of the bump sometimes correspond to clipped regions of the signal and other times correspond to non-clipped regions of the signal. If this is the case, then there is an inherent tradeoff between precision and recall for these regions of the signal. Here, precision is considered more important than recall because we wish to replace all clipped regions of the signal, which can be done without knowledge of all non-clipped regions of the signal.

The following observations can be made by comparing results of different experiments. Firstly, experiments with different amounts of clipping have similar precision, recall, and f-measure. This indicates that the clipping level estimate is equally reliable regardless of the size of the bumps in the histogram. Secondly, precision, recall, and f-measure are similar between experiments using block-level histograms and experiments using file-level histograms. This indicates that the clipping detection algorithm can be used on time varying clipping. Thirdly, recall tends to improve when using clipping interval detection. This reinforces our claim that time-domain analysis can help find the accurate boundaries of clipping, which is especially useful for reducing false positives.

It is worth mentioning that lossy encoding might slightly alter the locations of clipping, which could impact the results of our evaluation.

5.2. Sample Replacement

5.2.1. Evaluation Methodology

Objectively evaluating sample replacement is also problematic because the goal is to measure the perceptual difference between the clipped and reconstructed signals. Here, we base our evaluation off of the standard audio distortion measurement metric tonal harmonic distortion (THD), and we provide some declipping listening examples online [2].

We compare the total harmonic distortion (THD) of a clipped sinusoidal before and after sample replacement. We soft clip (as in Section 5.1.1) the sinusoidal input at the 90th percentile and run sample replacement using the ground truth clipping locations. The signal is 2 seconds long and sampled at 44.1 kHz. We test the performance of replacing short intervals (cubic spline interpolation) and replacing long intervals (frequency magnitude interpolation) separately. When testing long interval replacement, we also pad the input with one second of a non-clipped sinusoidal before and after the clipped region. Without this padding, there would be nowhere in the signal where a clean FFT is possible (note that using our algorithm, a clipped sinusoidal would normally be declipped in the time domain, but here we are interested in evaluating the frequency domain interpolation). The padding is removed before measuring the THD.

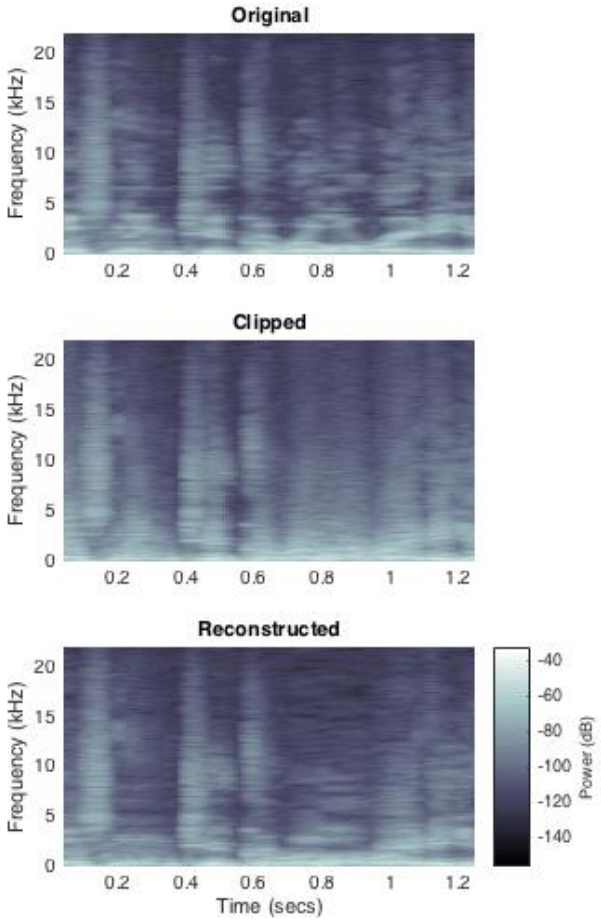

Figure 7: Clean, clipped, and reconstructed speech signal spectrograms.

5.2.2. Evaluation Results

THD results are shown in Figure 5. Both time domain and frequency domain methods are shown to improve the signal THD. The time domain reconstruction THD increases as frequency increases. This is because the number of samples in a single period decreases as frequency increases, and therefore a greater interpolation accuracy is necessary for higher frequencies to maintain a constant THD. This is an artifact of sampling; if the experiment is run with a high enough sample rate (e.g., 192 kHz), then the time domain reconstruction THD is constant with respect to frequency, remaining near -70 dB.

The frequency domain reconstruction THD decreases as frequency increases. This is due to phase differences between the clipped and the original sinusoidal. In order to verify this, the original phase of the sinusoidal was used, and the resulting THD was relatively constant with respect to frequency (approximately -90 dB).

The frequency domain approach obtains a lower THD than the time domain approach across frequencies. However, the frequency domain approach required extra information (a clean signal before and after clipping) in order to function properly. This validates our two-stage approach to sample replacement.

Results for sinusoidals are not necessarily generalizable to high bandwidth signals. The next section briefly illustrates the algorithm performance on real world signals.

5.3. End to End System Evaluation

5.3.1. Declipping On Real World Signals

To illustrate the performance of our algorithm on a real-world signal, we visualize the waveform and spectrogram of a clean, clipped, and declipped speech signal. Figure 6 shows the signal waveforms. The waveform of the declipped signal matches the general shape of the clean signal, although the reconstruction peaks tend to have a lower magnitude than the clean peaks. Figure 7 shows the signal spectrograms. From the spectrogram of the clipped signal, the high-frequency distortion caused by clipping is clearly visible. It can be seen from the reconstruction spectrogram that this distortion is mostly removed; however, the higher partials of speech are occasionally missing.

5.3.2. Computation Speed

The proposed algorithm is computationally efficient. We ran the web app implementation of the algorithm on a 2-minute long stereo audio file sampled at 44100 kHz. The experiment was run using a 2012 MacBook Pro with a 2.9 GHz quad-core processor and Google Chrome. The algorithm took 7 minutes to finish. This is a major improvement over existing declipping algorithms: we implemented a method similar to [9] in Matlab, and it took roughly 2.5 hours to process 5 seconds of audio.

6. CONCLUSION

We present an efficient algorithm for end to end declipping. The algorithm works for hard clipping and

soft clipping as well as for clipping thresholds changing over time. The clipping detection algorithm supplements histogram analysis with time domain analysis in order to reduce false positives. Our two-pass sample replacement algorithm demonstrates that frequency domain approaches can be applied to declipping even though locations of clipping are unpredictable. Frequency domain approaches to declipping are inherently less computationally expensive than sparse reconstruction approaches, which cannot take advantage of FFT optimizations; therefore, it is worthwhile to continue investigation of frequency domain approaches to declipping.

7. REFERENCES

[1] S. Godsill, P. Rayner, "Digital audio restoration," New York: Springer, 1998.

[2] C. Laguna, A. Lerch, "ClipAway," http://cplaguna-audio.github.io/ClipAway, 2016 (accessed 01 May 2016).

[3] S. Aleinik, Y.N. Matveev, "Detection of clipped fragments in speech signals," in Int. J. Electr. Electron. Sci. Eng., 2014, pp.74–80.

[4] T. Otani, M. Tanaka, Y. Ota, S. Ito, "Clipping Detection Device and Method," US Patent 20100030555 A1, Int. Cl. G10L 21/02, 2010.

[5] T. E. Riemer, M. S. Weiss, M. W. Losh, "Discrete clipping detection by use of a signal matched exponentially weighted differentiator," in Southeastcon'90. Proceedings., IEEE, 1990, pp. 245–248.

[6] A. Janssen, R. Veldhuis, L. Vries, "Adaptive interpolation of discrete-time signals that can be modeled as autoregressive processes," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 2, pp. 317–330, Apr. 1986.

[7] P. A. Esquef, V. Välimäki, K. Roth, I. Kauppinen, "Interpolation of long gaps in audio signals using the warped burg's method," in *Proc. of the 6th International Conference on Digital Audio Effects (DAFx-03)*, 2003, pp. 08–11.

[8] W. Etter, "Restoration of a discrete-time signal segment by interpolation based on the left-sided and right-sided autoregressive parameters," *IEEE Transactions on Signal Processing,* vol. 44, no. 5, pp. 1124–1135, 1996.

[9] L. W. P. Biscainho, P. S. R. Diniz, P. A. A. Esquef, "ARMA processes in sub-bands with application to audio restoration," in *Proc. of the 2001 IEEE International Symposium on Circuits and Systems, 2001, ISCAS 2001*, 2001, vol. 2, pp. 157–160 vol. 2.

[10] A. Lukin, J. Todd, "Parametric Interpolation of Gaps in Audio Signals," *Audio Engineering Society Convention 125,* pp. 3-6, 2008.

[11] M. Lagrange, S. Marchand, J.-B. Rault, "Long Interpolation of Audio Signals Using Linear Prediction in Sinusoidal Modeling," Journal of the Audio Engineering Society, 2005, vol. 53, no. 10, pp. 891–905.

[12] A. Adler, V. Emiya, M. G. Jafari, M. Elad, R. Gribonval, M. D. Plumbley, "Audio Inpainting," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 922–932, Mar. 2012.

[13] B. Defraene, N. Mansour, S. De Hertogh, T. van Waterschoot, M. Diehl, M. Moonen, "Declipping of Audio Signals Using Perceptual Compressed Sensing," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 12, pp. 2627–2637, Dec. 2013.

[14] Y. C. Pati, R. Rezaiifar, P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *1993 Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers, 1993*, 1993, pp. 40–44 vol.1.

[15] H. Homburg, I. Mierswa, B. Moller, K. Morik, M. Wurst, "A Benchmark Dataset for Audio Classification and Clustering," *Proc. of the International Symposium on Music Information Retrieval 2005*, pp. 528-531, 2005.