A NOTE ON THE IMPLEMENTATION OF AUDIO PROCESSING BY SHORT-TERM FOURIER TRANSFORM

James A. Moorer
1111 Alligator Drive
Panacea, FL 32346, USA

jamminpower@earthlink.net

ABSTRACT

Short-term Fourier Transform (STFT) forms the backbone of a great deal of modern digital audio processing. A number of published implementations of this process exhibit time-aliasing distortion. This paper reiterates the requirements for alias-free processing and offers a novel method of reducing aliasing.

Index Terms— DFT, STFT, Convolution, Aliasing

1. INTRODUCTION

A number of DSP processing techniques employ the Short-term Fourier Transform (STFT). These include Non-negative Matrix Factorization [1][2], binary filtering [3], broadband denoising [4] and many others. Some implementations of these algorithms exhibit significant audible artifacts. Some of the published implementations of these techniques show a lack of attention to issues of time-aliasing distortion, which is one form of artifact that can be audible and annoying. We show that all (synchronous) STFT processing is a form of linear filtering. This makes it straightforward to identify when time-aliasing is introduced and suggests ways to eliminate it. We introduce the concept of "brick-wall" windowing, implemented by a frequency-domain convolution to limit implied impulse response lengths. For binary filtering, we introduce the concept of an "atom" representing a single point in the frequency domain that has limited extent in time, thus allowing alias-free binary filtering.

2. HISTORICAL NOTE

In 1966, Thomas G. Stockham, Jr., published a seminal paper [5] describing the implementation of digital convolution by use of the Fast Fourier Transform [6]. The point of mentioning Stockham is just to note that this technique has been known for at least 50 years.

Since Stockham, a great deal of progress has occurred. Many modern DSP techniques have the computational paradigm of starting with one or more STFT (Short Term Fourier Transform) sequences, then producing a corresponding STFT sequence. The output is then produced by taking the inverse FFT of each frame, then overlapping and adding into the output sequence. In the process, some examples of published implementation code violate, explicitly or implicitly, the basic principles enumerated by Stockham in 1966. In this paper, we identify a common error leading to time-aliasing and suggest a novel way to correct it. This paper is extracted from a more complete discussion in [7].

3. SHORT-TERM FOURIER TRANSFORM

Let us define the padded STFT of an audio signal *x(n)* as follows:

$$X_k(n) = \sum_{m=0}^{N-1} w(m)x(n-m)e^{2\pi jmk/M} \quad (1)$$

where

n	Time index
k	Frequency index, $0 \le k < M$
N	Number of input points per frame
M	Transform size ($M \ge N$)
$w(m)$	Window function

This calculation is performed at certain fixed time intervals that is called the "hop size" or the "frame rate".

From one or more of these sequences, we then produce a new STFT sequence $\tilde{X}_k(n)$. We can then synthesize an output sequence, $\tilde{x}(n)$, by taking the inverse transform of each frame, then overlapping and adding to form the output..

4. STFT IS LINEAR FILTERING

Knowing *nothing* about how $\tilde{X}_k(n)$ was formed, what is the relation between $\tilde{X}_k(n)$ and $X_k(n)$? If the input and output frames have the same phases, then their relation is that the output is the result of applying a zero-phase filter to the input. This observation is independent of how the output frames were computed. We can make this explicit by calculating a gain function as follows:

$$G_k(n) \equiv \frac{\tilde{X}_k(n)}{X_k(n)} \quad (2)$$

Assuming we take care not to divide by zero, $G_k(n)$ is the transfer function of a zero-phase filter that will transform that input to the desired output. Since $G_k(n)$ defines a linear filter, it has an impulse response, and what we are performing is a convolution. Note that the input and output frames need not have the same phase. In that case, the filter will not have linear phase, but it is still a linear filter and has an impulse response. In general, the transfer function will be different in each STFT frame and will represent a *time-varying* filter. Limiting time-aliasing in each frame is sufficient to produce an alias-free overall result.

978-1-5386-1631-4/17/$31.00 ©2017 IEEE

5. IMPULSE RESPONSE AND CONVOLUTION

How long is the impulse response of this filter? Can it be a single point? It is only a single point if the implied filter is just a gain change. Any other transfer function will have a non-trivial impulse response. Let us say that the impulse response of this filter is K points in length (or that it can be windowed to K points in length without excessive distortion). We know that we need the size of the transform to be at least *N+K-1* in length or we will get time-aliasing due to the circular nature of convolution in the digital domain [5]. Since these transfer functions are generated algorithmically from unknown data, we can generally only loosely bound the length of the impulse responses.

6. LIMITING IMPULSE RESPONSE LENGTH

How do we limit the length of the impulse response implied by the transfer function $G_k(n)$? We can take its inverse transform, and if the significant part of the impulse response is less than *M* points in length, then we know that we can force its length to be less than *M-N+1* by simple windowing in the time domain. If the inverse transform does not seem to die out over *M* points, then it is probably longer than M, and longer transforms (more padding) must be used, else time-aliasing distortion will be introduced. *We must always use transforms long enough to encompass the longest impulse response that the input/output transfer function can exhibit.* Ideally, we would use a transform with enough padding so that the significant portion of the calculated impulse response of $G_k(n)$ is less than *M-N+1* points in length.

7. IMPLEMENTING A "BRICK-WALL" WINDOW

The process of time-limiting an impulse response is a well-known one. We multiply the first *M-N* points by a window function, then we zero the remaining points. This is analogous to a band-limiting filter, except that it is applied in the time domain. Since band-limiting filters are often called "brick-wall" filters, we might call this a "brick-wall" window. Applying a time-domain window to the impulse response of $G_k(n)$ would appear to involve two consecutive FFTs – one to inverse transform $G_k(n)$ to the time domain for windowing, then one more to transform it back to the frequency domain for application. We can implement an approximation to this process by a convolution in the frequency domain using the transform of the desired time-domain window function. The approximation comes by limiting the number of non-zero points in the frequency-domain convolution kernel. The more terms we use in the convolution kernel, the closer we come to the ideal brick-wall window, but at some point, the computational demands exceed what would be required for the direct implementation involving two transforms noted above.

8. AN EXAMPLE WINDOW KERNEL

Let us illustrate the issue with a concrete example. We will take *M = 2N*, so that our input sequence will be taken *N* points at a time, and we will have to limit the length of the impulse response to *N+1* points. For simplicity, let us round that down to just *N* points. In the time domain, we might choose a window for the impulse response to be *N* points of a Hamming window, followed by *N* zeros. Figure 1 shows a half-length Hamming window implemented directly, versus an implementation as a frequency domain convolution of $G_k(n)$ with a 5-point kernel and a 7-point kernel. The exact values for the 7-point kernel are (-0.9854, 3.68, -7.0597, 8.64, -7.0597, 3.68, -0.9854). For convenience, these kernels were generated by simply applying a 5 and 7-point rectangular (Fourier) window to the exact transform of the half-length Hamming window. There are surely better ways to produce suitable kernels.

This simple 5-point convolution kernel gives us about 23-dB of rejection of the impulse response beyond *N* points. This is generally enough for most audio applications, unless the transfer function exhibits sharp slopes. The 7-point kernel gives over 30 dB of rejection.

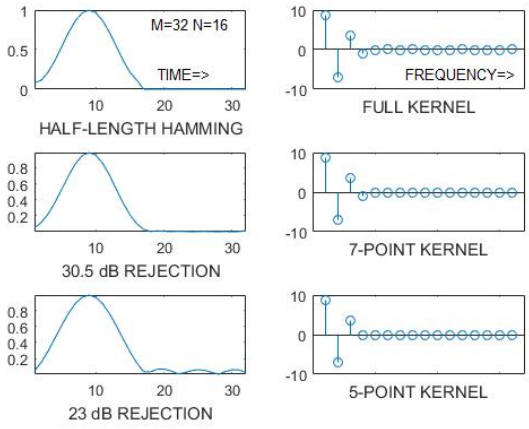

Figure 1: Half-length Hamming window used to produce a frequency-domain convolution kernel to implement some amount of time windowing to suppress the impulse response beyond *N* points. Only the upper wing of the symmetric kernel is shown.

9. SYNTHESIS FROM NMF IS LINEAR FILTERING

NMF ("Non-Negative Matrix Factorization") is a useful tool for a number of tasks. I will discuss one use of it here, which is the so-called "source separation" problem. In general, the idea is that you have two sounds that have largely independent statistics. We will use NMF to select spectral components thought to represent one source or another [1].

In NMF, we approximate a vector (such as a magnitude-spectrum of a bit of audio) by a weighted sum of other example vectors. We constrain the weights to be non-negative. Magnitude spectra of real sequences are, by definition, non-negative. We represent this factorization symbolically as follows:

$$V \approx WH \qquad (3)$$

Where

V	vector, such as a magnitude spectrum
H	matrix of many such vectors
W	matrix of non-negative weights

An example of this source separation by NMF might be separating a mixture of male and female speakers talking simultaneously. You start by building to libraries, H_m and H_f, consisting of many, many magnitude spectra of male speech and female speech. We then take the speech mixture to be separated and represent it by a sequence of transforms taken at some frame rate and the same frame size as those in the libraries. For each frame, we then approximate the magnitude spectrum first using the male library then the female library. The phases are generally taken directly from the input mixture. There are many techniques available for calculating the weights in clever and efficient ways [2]. As noted in [1], constraints can be applied to guide the process. Ultimately, however the resulting magnitude spectrum is computed, we end up with a sequence of frames, $\tilde{X}_k(n)$. Consequently, for each frame, we can compute the transfer ratio of a zero-phase filter, $G_k(n)$, as noted in equation (2). This transfer function has an impulse response that may or may not be longer than *M-N+1* points in length. Figure 2 shows an example of the inverse transform of one frame of $\tilde{X}_k(n)$ taken from a NMF source-separation task with *M=2N*.

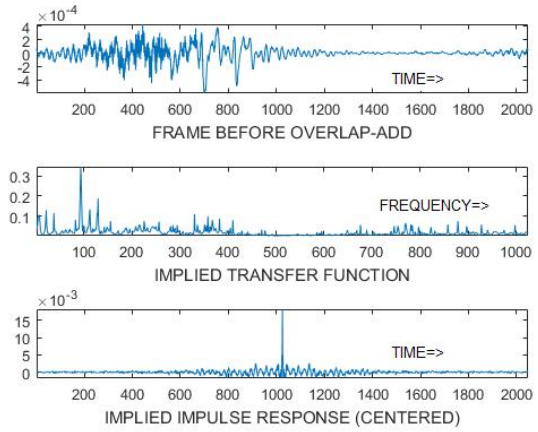

Figure 2: An example of a synthesized output frame from an NMF separation task with 2:1 padding (*M*=2048, *N*=1024). Middle plot is implied magnitude transfer function. Bottom is centered impulse response of implied transfer function

In the case shown, the length of the impulse response corresponding to $G_k(n)$ is clearly greater than or equal to *N+1* points. Without padding or windowing, there will be time-aliasing that may be audible.

Note also that the impulse response of $G_k(n)$ extends both backwards and forwards in time. This is a natural characteristic of a zero-phase filter. *In addition to adding sufficient padding and possibly limiting the length of the impulse response of $G_k(n)$, it is necessary to circularly shift the synthesized frame by (M-N)/2 points.* Adding padding by itself *without* circularly shifting the result before the overlap-add step leaves a discontinuity at the beginning and end of the frame that will be audible. Note that the shift can be done on input. That is, rather than padding the input data by annexing zeros, one may imbed the windowed input data in the middle of the analysis window with an equal number of zeros before and after the input data. The centering has to be done somewhere – either on the input side or the output side. *Note that in many cases, it is sufficient to add centering and padding to NMF to reduce time aliasing to inaudible levels*. That is, it is often not necessary to explicitly window the impulse response of the implied transfer function, but simply to use sufficient padding (*e.g.* 4:1 or 8:1) and centering the data in the window, trusting that the filters produced will not be too sharp. Much of the time, this will be the case.

10. BINARY FILTERING IS STILL FILTERING

It is sometimes interesting to construct a "binary" filter. That is, a filter with a magnitude transfer spectrum that consists of only ones and zeros [3]. Although this is somewhat of an artificial constraint, as any natural sound may have contributions to each time-frequency point from a number of sources, the restriction to ones and zeros greatly simplifies the separation algorithm design. Without interpreting the ones and zeros, we can independently raise the question of audio quality. That is, given a desired transfer function in the form of a binary mask, how do we make it sound as good as possible? We start by examining the sources of distortion. In this paper, we look at only one type of distortion, which is time aliasing.

The previous discussion about padding assumes that the implied transfer function impulse response decays to zero, so that making the analysis window larger (*i.e.* increasing the amount of padding) will always result in an impulse response that decays below any arbitrary threshold level. It is reasonable to ask when this is true and when it is not.

Consider the impulse response of a single time-frequency point. This can be considered the "atom" of binary filtering. A single point in the frequency domain, surrounded by zeros, is the only signal that does *not* decay with time. If we even have two consecutive non-zero points, the impulse response will decay and increasing padding will always reveal an impulse response that is effectively time-limited. Unfortunately, the selection algorithms for source separation involving binary masks often produce isolated frequency points. How can we deal with this? There are several choices:

1. We could always require 2 or more adjacent frequency points to be non-zero and work that requirement into the selection algorithm. If more precision in the frequency domain is required, additional padding can always be used to increase the number of frequency points.
2. Although we compute a binary mask, we could synthesize a slightly different transfer function to actually perform the filtering just for auditioning. For example, after the binary mask is computed, we could then re-analyze the input using a 3:1 padding. At every frequency in the original mask that is one, we could place a 3-point Hamming window kernel – that is, (-.23, .54, -.23), centered on the frequency of the non-zero point in the original mask. This changes the "atom" to a function that is known to decay. This technique can be expanded to 5-point kernels, such as Blackman windows, or even higher-order kernels [4]. We can make time-limited functions to approximate sequences of any number of consecutive unit frequency bins. As noted in [4], these functions have closed-form formulas both in the time

and frequency domains. This suggestion is, in fact, another implementation of windowing through frequency-domain convolution as previously described.

3. Any other method could be used to "smooth" the frequency response. The smoothing need not be done using a linear frequency scaling, but could be adjusted to be a Bark scale, equal-octave, or any other grouping. The more smoothing that is used, the more the response deviates from the original binary mask, but this can be remediated by increasing the padding which increases the frequency precision accordingly. By increasing the padding, the resulting frequency response can be made to approximate the original binary mask arbitrarily precisely.

Figure 3 shows one example of a time-limited "atom" that can be used to implement each non-zero frequency point. This atom was produced by the use of 1:8 padding, and consists of five consecutive 5-point Blackman window kernels. This is an approximation to a bandpass filter with a width corresponding roughly to that of a single point in the original binary mask. Although this is sufficient to guarantee freedom from time-aliasing, other artifacts due to sharp band-edges will dominate. Some amount of frequency-domain smoothing would be necessary to reduce this remaining form or artifact.

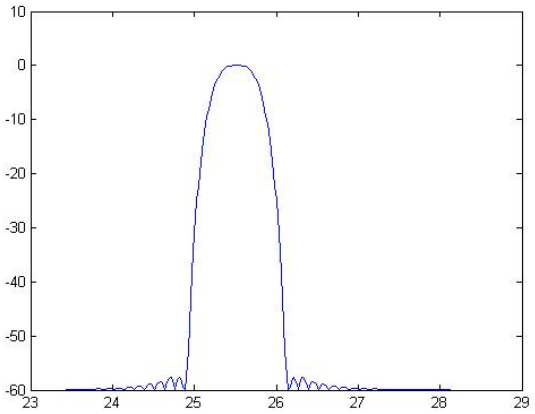

Figure 3: Windowed and time-limited approximation to a spectral impulse at frequency bin 25. This was produced by starting with a spectrum of an impulse response padded 1:8. Five consecutive sets of 5-point Blackman window coefficients were summed to produce a filter roughly one unit wide in the original (unpadded) domain.

The frame rate interacts with the window chosen for the approximation. We know that the Hamming and Hann windows can be used the a hop size of *N/2* without introducing any time-domain amplitude modulation. Use of a 2nd-order window, such as Blackman, requires us to reduce the hop size to *N/4* to eliminate amplitude modulation.

Some researchers are using "soft masks" rather than strict binary filtering, *e.g.* [8]. This relaxes the constraint that each frequency point be either 0 or 1. By itself, of course, this does not guarantee freedom from time-aliasing. Steps must still be taken to limit the implied impulse response length.

11. SUMMARY

Processing using the short-term Fourier transform can be formulated as a linear filter, regardless of how the output magnitude spectra are generated. As such, the transfer function for each frame can be analyzed to determine if it introduces time-aliasing distortion. We introduce a novel frequency-domain convolution that implements an approximation to a "brick-wall" window function. In many cases, simple padding and "centering" of the analysis data is sufficient to reduce time-aliasing distortion to inaudibility. For more radical filters, such as binary filters, time-aliasing can only be reduced in general by some kind of smoothing in the frequency domain. One solution involves replacing the "atom" with a kernel that is known to decay to zero and increasing the padding size to accommodate the chosen kernel. The frame rate may also need to be increased to avoid amplitude-modulation distortion.

In all cases, however, formulating STFT processing as a linear filtering operation leads us directly to these guidelines and techniques such that time-aliasing distortion can *always* be avoided with relatively modest penalties in time and/or complexity.

12. ACKNOWLEDGMENT

Many thanks to Paris Smaragdis and Gautham Mysore for numerous suggestions and encouragement in the creation of this paper, and to the reviewer's constructive recommendations.

13. REFERENCES

[1] Tuomas Virtanen "Monaural Sound Source Separation by Nonnegative Matrix Factorization With Temporal Continuity and Sparseness Critera" *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 15, No. 3, March 2007, pp. 1066-1074

[2] Lee and Seung "Algorithms for Non-Negative Matrix Factorization" in *Advances in Neural Information Processing Systems* 13: *Proceedings of the 2000 Conference*. MIT Press pp. 556–562

[3] Pedersen *et al.* "Two-Microphone Separation of Speech Mixtures" *IEEE Transactions on Neural Networks,* Volume 19, No. 3, March 2008, pp. 475-492

[4] James Moorer and Mark Berger "Linear Phase Bandsplitting" *Audio Engineering 76th Conference*, New York, 1984

[5] Thomas G. Stockham, Jr., "High-Speed Convolution and Correlation", *1966 Spring Joint Computer Conference, AFIPS Conference Proceedings*, Volume 28, pp 229-233

[6] J.W. Cooley and J.W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Comutation*, Volume 19, Number 90, April 1965, pp. 297-301

[7] James Moorer, "NMF, WOLA, And Binary Filtering: Avoiding the Curse of Time-Aliasing". http://www.jamminpower.com/PDF/NMF_WOLA_Binary_Filtering.pdf

[8] Antoine Liutkus, Roland Badeau. Generalized Wiener _ltering with fractional power spectrograms. 40th International Conference on Acoustics, Speech and Signal Processing (ICASSP), Apr 2015, Brisbane, Australia. IEEE, 2015.