HARMONIZING EFFECT USING SHORT-TIME TIME-REVERSAL

Hyung-Suk Kim
CCRMA,
Stanford University
Stanford, CA, USA
hskim08@ccrma.stanford.edu

Julius O. Smith
CCRMA,
Stanford University
Stanford, CA, USA
jos@ccrma.stanford.edu

ABSTRACT

A prior study of short-time time-reversal showed sideband modulation occurs for short time durations, creating overtones for single sinusoid signals. In this paper, we examine the overtones created by short-time time-reversal and the tonal relation between the overtones and the input signal. We present three methods of using short-time time-reversal for harmonizing audio signals. Then modifications to the previous short-time time-reversal needed to implement the proposed methods are described.

1. INTRODUCTION

In a previous paper [1], the general analysis of short-time time-reversal (STTR) was covered. It was shown that for very short window lengths, less than 30 ms, STTR has a sideband modulation effect, a form of spectral aliasing, that creates overtones for single sinusoidal signals and the possibility of exploiting this property for use as a harmonizing effect was briefly mentioned. In this paper, we will examine the overtones created by STTR on sinusoidal signals and propose methods for using STTR as a harmonizing effect.

Harmonizing effects, an application of pitch-shifting, is a widely used audio effect [2] popularized by the Eventide Harmonizer series. Previous implementations of harmonizing effects include delay-line modulation [3, 4, 5] and phase-vocoders [6, 7].

STTR can be categorized as a delay-line modulation. However, it is not a direct application of pitch shifting. The overtones occur naturally from a form of sideband modulation that arises by STTR. Because of this, a harmonizing effect using STTR can be implemented very efficiently. For 50% overlap-add STTR, only a single delay line per channel and two read pointers are needed [1].

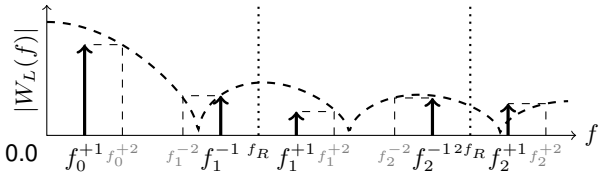
2. SHORT-TIME TIME-REVERSAL

STTR is a linear time-variant filter where the audio signal is windowed and the signal under the window is time reversed in place. In this section, we review the STTR equations and specify the parameters that will be used in the following sections.

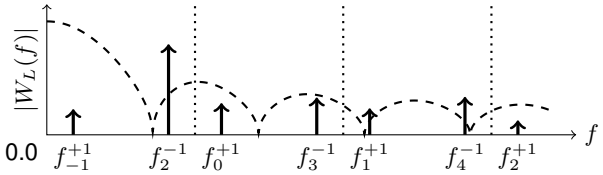
2.1. The STTR Equations

In [1], the equations for STTR in both the time-domain and the frequency-domain were derived. For input signal $x(t)$, the output of STTR $y(t)$ and its frequency response $Y(f)$ is as below, where $w_L(t)$ is the windowing function of length L used for time reversal and R is the step size used for overlap-add.

$$y(t) = \sum_{m=-\infty}^{\infty} x(-t + 2mR) w_L(t - mR) \tag{1}$$



(a) $f_0 < \frac{f_R}{2}$



(b) $f_0 > f_R$

Figure 1: A visualization of equation (4), the STTR output for a single sinusoid. The dotted vertical lines mark multiples of f_R. 1a shows the simple case where $f_0 < \frac{f_R}{2}$. The amplitude of the impulse at $f_k^{\pm 1}$ is found by sampling the window function $|W_L(f)|$ at $f_k^{\pm 2}$. 1b shows the generic case where $f_0 > f_R$. In such cases, the fundamental frequency, $f_0 = f_0^{+1}$, might not have the greatest amplitude, which can be used to harmonize the original signal.

$$Y(f) = \frac{1}{R} \sum_{k=-\infty}^{\infty} X\left(f - \frac{k}{R}\right) W_L\left(2f - \frac{k}{R}\right) \tag{2}$$

As an added constraint to preserve signal power, $w_L(t)$ and R must satisfy constant overlap-add.

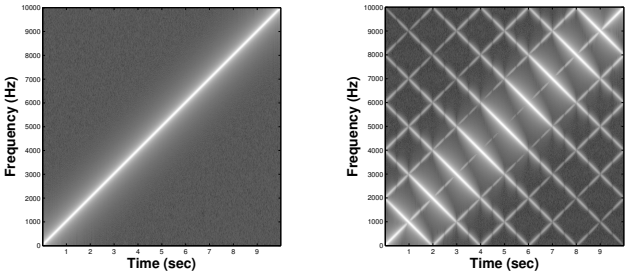
$$\sum_{m=-\infty}^{\infty} w_L(t - mR) = 1 \tag{3}$$

2.2. STTR for Single Sinusoid Input

For a single sinusoid input $x(t) = \cos(2\pi f_0 t + \phi)$, the STTR output is

$$Y(f) = \frac{1}{2R} \sum_{k=-\infty}^{\infty} \Big\{ e^{i\phi} W_L(f_k^{+2}) \delta(f - f_k^{+1}) + e^{-i\phi} W_L(f_k^{-2}) \delta(f - f_k^{-1}) \Big\} \tag{4}$$

where $f_R = \frac{1}{R}$ and $f_k^{\pm a} = k f_R \pm a f_0$. f_R is called the frame rate.



(a) Input signal (b) Window length 1 ms

Figure 2: Spectrogram plots for a 10 second linear sine sweep from 0 Hz to 10 kHz (2a) and the STTR output with a Hann window of length 1 ms and 50% overlap-add (2b). The frame rate f_R is 2 kHz.

We see that STTR creates overtones at frequencies $f_k^{\pm 1}$, f_0 above and below multiples of f_R. Note that the fundamental frequency of the input is $f_0 = f_0^{+1}$. The amplitude for each overtone is proportional to $W_L(f_k^{\pm 2})$, the window spectrum sampled at $2f_0$ above and below multiples of f_R. This is illustrated in Figure 1a.

Figure 1b illustrates a general case where the overtone with the greatest amplitude is not the fundamental frequency f_0. This is further visible in Figure 2, STTR output for a linear sine sweep. The main diagonal is the fundamental frequency (Figure 2a), however the main diagonal of the output (Figure 2b) does not always have the greatest magnitude. This gives rise to the possibility of using STTR to harmonize an input signal.

2.3. Fixed Overlap-add Ratio

To simplify implementation, we use a fixed overlap-add ratio. In effect, the window function becomes dependent of the step size R. For a fixed overlap-add ratio $\alpha = R/L$, the window length is $L = R/\alpha = \frac{1}{\alpha f_R}$. The window spectrum $W_L(f) = W_{\frac{R}{\alpha}}(f)$ stretches in the frequency domain relative to f_R.

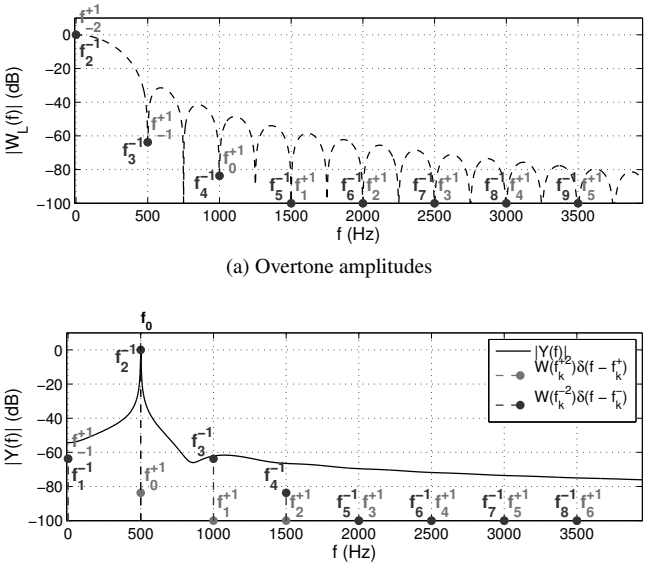
An interesting example is when the window function is of the generalized Hamming window family. In this case, the window spectrum is zero at frequencies $f = k \times 1/L = k \times \alpha f_R$, where k is an integer other than $-1, 0, 1$. For 50% overlap-add, the window spectrum will be zero at multiples of $f_R/2$ except for $f = 0, \pm f_R/2$. This means that for octaves of $f_R/2$, $f_0 = nf_R/2$, where n is a natural number, no overtones will occur.

3. OVERTONE ANALYSIS

In this section, we take a look at examples to understand the overtones created by STTR, then generalize the results to obtain an overtone map. For the examples, we will use 50% overlap-add ($\alpha = 0.5$) with a Hann window.

3.1. Example 1: $f_0 = f_R$

We first look at the simple case where the frame rate is equal to the input frequency. For $f_0 = f_R$, $W_{2R}(f) = 0$ at $f = k \times f_R/2$ for integer k other than $-1, 0, 1$. Using this fact, equation



(a) Overtone amplitudes

(b) Ideal overtones with STTR output spectrum

Figure 3: Plots illustrating the STTR overtones for $f_0 = f_R = 500$ Hz. Figure 3a shows the window spectrum and the points $W_L(f_k^{+2})$. The points are labeled with the corresponding overtone frequency f_k^{+1}. Figure 3b shows the resulting overtones (dotted) given by equation (4) together with the spectrum of the STTR output, $Y(f)$ (solid). Since the window function is a Hann window, most of the overtone amplitudes fall on the nulls.

(4) simplifies to,

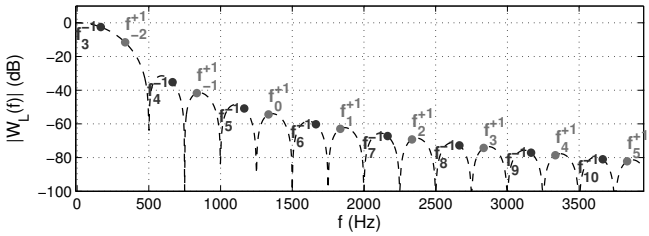
$$\begin{aligned} Y(f) =& \frac{1}{2R}\Big\{ e^{i\phi} W_{2R}(f_{-2}^{+2})\delta(f - f_{-2}^{+1}) \\ &+ e^{-i\phi} W_{2R}(f_2^{-2})\delta(f - f_2^{-1}) \Big\} \\ =& \frac{1}{2}\Big\{ e^{i\phi}\delta(f + f_0) + e^{-i\phi}\delta(f - f_0) \Big\} \\ =& \overline{X(f)} \end{aligned} \tag{5}$$

The output in this case is the conjugate of the input, $\overline{X(f)}$, in the frequency domain, which is the time reversed version of the input signal in the time domain. This makes sense since we are reversing two full periods of $x(t)$ which in effect does nothing to the fundamental frequency. However, from an STTR perspective, it is an overtone at $f_2^{-1} = 2f_R - f_0$, not the original input $f_0 = f_0^{+1}$, that represents the signal (Figure 3).

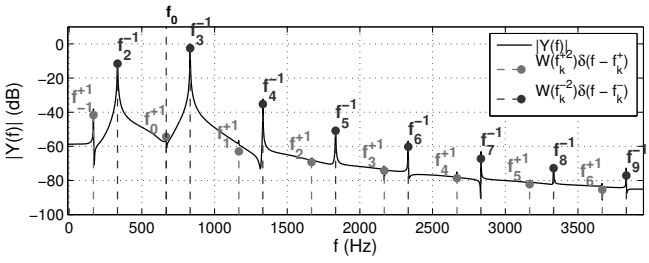
3.2. Example 2: 5 Semitone Difference

We next look at an example where the input frequency is a perfect 4th (5 semitones) above the frame rate, i.e. $f_0 = 2^{5/12} \times f_R$. For this case, most of the overtones are not zero (Figure 4) and thus equation (4) will not simplify like the previous case.

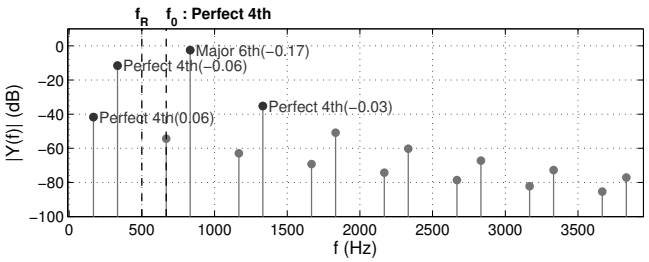
Figure 4b shows the annotated peaks along with the spectrum of the STTR output. We see that f_0 is not the most prominent frequency. To make tonal sense of the overtones, the peaks are annotated with the closest musical interval relative to f_R in Figure 4c. The most prominent overtone is approximately a major 6th



(a) Overtone amplitudes



(b) Ideal overtones with STTR output spectrum



(c) Overtones mapped to musical intervals

Figure 4: Plots illustrating the STTR overtones for $f_0 = 2^{5/12} \times f_R$ with a Hann window where $f_R = 500\,\text{Hz}$. $W(f_k^{\pm 2})$ is mostly non-zero in this case (Figure 4a), and as a result there are many overtones in the output signal (Figure 4b). Figure 4c shows the peaks greater than $-45\,\text{dB}$ annotated with the musical interval with f_R as the key. The greatest frequency is neither f_0 nor f_R.

Semitones	Interval	Overtones			
-12	**P1**	**P1 (0 +0.00)**			
-11	m2	m7 (-2 -0.06)	TT (18 -0.33)	P1 (0 +0.00)	
-10	**M2**	m6 (-4 -0.26)	**M3 (16 +0.30)**	P1 (0 +0.00)	
-9	m3	**P4 (-7 +0.37)**	m3 (15 -0.11)	P1 (0 +0.00)	P1 (24 +0.20)
-8	**M3**	m3 (-9 -0.21)	m2 (13 +0.45)	P1 (0 +0.00)	M7 (23 -0.06)
-7	**P4**	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	P1 (0 +0.00)	m7 (22 -0.34)
-6	TT	**M6 (-15 -0.26)**	m7 (10 +0.45)	P1 (0 +0.00)	
-5	**P5**	**P4 (-19 +0.06)**	**M6 (9 -0.13)**		
-4	m6	**P5 (7 +0.25)**	m2 (-23 -0.33)	TT (18 -0.30)	
-3	**M6**	TT (6 -0.44)	**P5 (-29 +0.18)**	M3 (16 +0.33)	P1 (0 +0.00)
-2	m7	**M3 (4 -0.21)**	**P1 (-36 -0.36)**	m3 (15 -0.08)	P1 (0 +0.00)
-1	**M7**	**M2 (2 -0.05)**	**M7 (-49 +0.14)**	m2 (13 +0.48)	
0	**P1**	**P1 (0 +0.00)**			
1	m2	m7 (-2 -0.06)	m7 (10 +0.48)	m7 (-50 +0.14)	
2	**M2**	m6 (-4 -0.26)	**M6 (9 -0.09)**	m7 (-38 -0.36)	M3 (16 +0.30)
3	m3	**P4 (-7 +0.37)**	**P5 (7 +0.28)**	M3 (-32 +0.18)	m3 (15 -0.11)
4	**M3**	TT (6 -0.41)	m3 (-9 -0.21)		
5	**P4**	**M3 (4 -0.17)**	**P1 (-12 -0.06)**	P1 (12 -0.03)	P1 (-24 +0.06)
6	TT	**M2 (2 -0.02)**	**M6 (-15 -0.26)**	m7 (10 +0.45)	m3 (-21 -0.26)
7	**P5**	**P1 (0 +0.04)**			
8	m6	m7 (-2 -0.02)	**P5 (7 +0.25)**	m2 (-23 -0.33)	m2 (13 +0.25)
9	**M6**	m6 (-4 -0.22)	TT (6 -0.44)	P5 (-29 +0.18)	P1 (12 -0.24)
10	m7	**M3 (4 -0.21)**	**P4 (-7 +0.42)**	m7 (10 +0.24)	
11	**M7**	**M2 (2 -0.05)**	m3 (-9 -0.16)	M6 (9 -0.34)	M7 (-49 +0.14)
12	**P1**	**P1 (0 +0.00)**			

Figure 5: Visualization of an STTR overtone table. The STTR settings are 50% overlap-add using a Hann window. In this instance, the tonic is equal to f_R. The rows span 1 octave above and below the tonic. The table only shows overtones greater than -45 dB with respect to the input signal. The overtones are annotated with the closest musical interval with the actual semitone difference and cent offsets in parenthesis. The text is scaled linearly by the overtone amplitude. Overtones that fall on a major scale with f_R as the tonic are shown in bold.[1]

(9 semitones) above the frame rate and a major 3rd (4 semitones) above the input frequency. Other prominent overtones are octaves of the input frequency. Playing a major scale in the key of f_R, the STTR output of a perfect 4th will approximately be in harmony. From the perspective of the input signal, setting the frame rate to be 5 semitones lower than the input frequency will result in a major 3rd harmony.

3.3. STTR Overtone Table

Using the method in the example above, we can map overtones greater than a reasonable threshold to the closest musical interval and find harmonizing notes for each semitone difference between f_0 and f_R .

Figure 5 shows an example STTR overtone table with overtones on a major scale in bold. The first column lists the semitone difference relative to f_R. The second column shows the abbreviated musical interval with respect to the tonic. The abbreviation indicates the type of interval (Major, minor, Perfect) and the semitone difference from the key. TT denotes a tritone.

For this case, playing the notes 0, 5, 7, 12 semitones above f_R will keep the harmonies on a major scale. Not surprisingly, the aforementioned intervals are known as the perfect intervals, intervals that occur as harmonics.

One point to note is that the overtones may vary over octaves, though overall the overtones between octaves are similar. For example, in Figure 5 the overtones of a major third -8 semitones below f_R are different from those of a major third 4 semitones above f_R.

For more general use, we can pre-compute the prominent overtone frequencies and save the ratio between the overtone frequency and the frame rate, then use this as a lookup table for harmonizing.

This is an example of using STTR to harmonize an input signal. In the next section, different methods of using STTR as a harmonizing effect are covered.

[1] An interactive version of the STTR overtone table can be found at `https://ccrma.stanford.edu/~hskim08/sttr/harmonize/`. Source code, compiled audio plug-ins and sound examples are also available at the URL.

4. HARMONIZING METHODS

From equation (4), we can express the frequencies and amplitudes of the overtones as

$$f_k^{\pm 1} = f_R(k \pm \beta) = f_0(k/\beta \pm 1). \quad (6)$$

$$W_L(f_k^{\pm 2}) = W_L(f_R(k \pm 2\beta)) = W_L(f_0(k/\beta \pm 2)) \quad (7)$$

where $\beta = f_0/f_R$. The overtones can be expressed relative to the input frequency f_0 or the frame rate f_R.

Since f_0 will be determined by the input signal, to use STTR as a harmonizing effect we have three choices, a) fix f_R and choose f_0 respectively, b) fix β by changing f_R with respect to f_0 or c) change both f_R and β according to some rule.

4.1. Fixed f_R

In Section 3.3, we covered an example of harmonizing a signal with the tonic f_{tonic} to be equal to the frame rate f_R. An extension of this approach is to separate the tonic from the frame rate by specifying an offset factor n_{offset} such that $f_{\text{tonic}} = 2^{n_{\text{offset}}/12} f_R$.

Figure 6 is an overtone table with $n_{\text{offset}} = 5$. With this setting, most of the notes on a major scale will loosely sound harmonized with a few out-of-scale overtones.

Semitones	Interval	Overtones			
-7	**P1**	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (0 +0.00)**	m7 (22 -0.34)
-6	m2	**M6 (-15 -0.26)**	m7 (10 +0.45)	**P1 (0 +0.00)**	
-5	**M2**	**P4 (-19 +0.06)**	**M6 (9 -0.13)**		
-4	m3	**P5 (7 +0.25)**	m2 (-23 -0.33)	TT (18 -0.30)	
-3	**M3**	TT (6 -0.44)	**P5 (-29 +0.18)**	**M3 (16 +0.33)**	**P1 (0 +0.00)**
-2	**P4**	**M3 (4 -0.21)**	**P1 (-36 -0.36)**	m3 (15 -0.08)	**P1 (0 +0.00)**
-1	TT	**M2 (2 -0.05)**	**M7 (-49 +0.14)**	m2 (13 +0.48)	
0	**P5**	**P1 (0 +0.00)**			
1	m6	m7 (-2 -0.06)	m7 (10 +0.48)	m7 (-50 +0.14)	
2	**M6**	m6 (-4 -0.26)	**M6 (9 -0.09)**	m7 (-38 -0.36)	**M3 (16 +0.30)**
3	m7	**P4 (-7 +0.37)**	**P5 (7 +0.28)**	**M3 (-32 +0.18)**	m3 (15 -0.11)
4	**M7**	TT (6 -0.41)	m3 (-9 -0.21)		
5	**P1**	**M3 (4 -0.17)**	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (-24 +0.06)**
6	m2	**M2 (2 -0.02)**	**M6 (-15 -0.26)**	m7 (10 +0.45)	m3 (-21 -0.26)
7	**M2**	**P1 (0 +0.04)**			
8	m3	m7 (-2 -0.02)	**P5 (7 +0.25)**	m2 (-23 -0.33)	m2 (13 +0.25)
9	**M3**	m6 (-4 -0.22)	TT (6 -0.44)	**P5 (-29 +0.18)**	**P1 (12 -0.24)**
10	**P4**	**M3 (4 -0.21)**	**P4 (-7 +0.42)**	m7 (10 +0.24)	
11	TT	**M2 (2 -0.05)**	m3 (-9 -0.16)	**M6 (9 -0.34)**	**M7 (-49 +0.14)**
12	**P5**	**P1 (0 +0.00)**			
13	m6	m7 (-2 -0.06)	**P4 (5 +0.32)**	**M6 (-15 -0.19)**	m7 (10 +0.48)
14	**M6**	m6 (-4 -0.26)	**M3 (4 -0.45)**		
15	m7	**M2 (2 -0.31)**	**P4 (-7 +0.37)**	**P5 (7 +0.28)**	m2 (-23 -0.23)
16	**M7**	**P1 (0 -0.27)**	TT (6 -0.41)	m3 (-9 -0.21)	
17	**P1**	m7 (-2 -0.35)	**M3 (4 -0.17)**	**P1 (-12 -0.06)**	m6 (8 +0.37)

Figure 6: Visualization of an STTR overtone table (50% overlap-add, Hann window) with f_{tonic} 5 semitones above f_R. The first column shows the semitone difference of f_0 and f_R while the second column shows the musical interval of f_0 relative to f_{tonic}.[2]

The advantage of this approach is that it requires no additional computation apart from that of standard STTR. It also presents a characteristic harmonizing scheme determined by the window function. However, finding a setting that works for all notes on a desired musical scale becomes a nontrivial search problem.

4.2. Fixed β

Another approach would be to keep the ratio between f_0 and f_R constant. This will keep the overtone intervals constant relative to f_0. This can be viewed as a form of pitch shifting, especially in the case where there is a single dominant overtone. However for most cases, there are many overtones so it will be a harmonizing effect.

f_R now becomes a function of f_0 by $f_R(f_0) = f_0/\beta$. This approach requires a fundamental frequency (F0) estimator. F0 estimation is a well studied area with many proposed solutions [8, 9, 10, 11].

We also need to decide what value of β to use, or what set of overtones to use. In Figure 7, the overtones are mapped to the musical interval relative to f_0. We can use this table to select β and in turn choose the overtones to use.

Semitones	Overtones			
-12	**P1 (0 +0.00)**			
-11	m7 (-2 -0.06)	TT (18 -0.33)	**P1 (0 +0.00)**	
-10	m6 (-4 -0.26)	**M3 (16 +0.30)**	**P1 (0 +0.00)**	
-9	**P4 (-7 +0.37)**	m3 (15 -0.11)	**P1 (0 +0.00)**	**P1 (24 +0.20)**
-8	m3 (-9 -0.21)	m2 (13 +0.45)	**P1 (0 +0.00)**	**M7 (23 -0.06)**
-7	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (0 +0.00)**	m7 (22 -0.34)
-6	**M6 (-15 -0.26)**	m7 (10 +0.45)	**P1 (0 +0.00)**	
-5	**P4 (-19 +0.06)**	**M6 (9 -0.13)**		
-4	**P5 (7 +0.25)**	m2 (-23 -0.33)	TT (18 -0.30)	
-3	TT (6 -0.44)	**P5 (-29 +0.18)**	**M3 (16 +0.33)**	**P1 (0 +0.00)**
-2	**M3 (4 -0.21)**	**P1 (-36 -0.36)**	m3 (15 -0.08)	**P1 (0 +0.00)**
-1	**M2 (2 -0.05)**	**M7 (-49 +0.14)**	m2 (13 +0.48)	
0	**P1 (0 +0.00)**			
1	m7 (-2 -0.06)	m7 (10 +0.48)	m7 (-50 +0.14)	
2	m6 (-4 -0.26)	**M6 (9 -0.09)**	m7 (-38 -0.36)	**M3 (16 +0.30)**
3	**P4 (-7 +0.37)**	**P5 (7 +0.28)**	**M3 (-32 +0.18)**	m3 (15 -0.11)
4	TT (6 -0.41)	m3 (-9 -0.21)		
5	**M3 (4 -0.17)**	**P1 (-12 -0.06)**	**P1 (12 -0.03)**	**P1 (-24 +0.06)**
6	**M2 (2 -0.02)**	**M6 (-15 -0.26)**	m7 (10 +0.45)	m3 (-21 -0.26)
7	**P1 (0 +0.04)**			
8	m7 (-2 -0.02)	**P5 (7 +0.25)**	m2 (-23 -0.33)	m2 (13 +0.25)
9	m6 (-4 -0.22)	TT (6 -0.44)	**P5 (-29 +0.18)**	**P1 (12 -0.24)**
10	**M3 (4 -0.21)**	**P4 (-7 +0.42)**	m7 (10 +0.24)	
11	**M2 (2 -0.05)**	m3 (-9 -0.16)	**M6 (9 -0.34)**	**M7 (-49 +0.14)**
12	**P1 (0 +0.00)**			

Figure 7: Visualization of an STTR overtone table (50% overlap-add, Hann window) with the overtones annotated with musical intervals relative to f_0. The "Semitones" column in this table can be translated to β, the ratio between f_0 and f_R.[3]

[2] https://ccrma.stanford.edu/~hskim08/sttr/harmonize/vis.html?offset=5.

[3] https://ccrma.stanford.edu/~hskim08/sttr/harmonize/vis.html?relative=1.

4.3. Variable f_R and β

One possibility of solving the limitations of the two previous examples, would be a hybrid approach where for a given f_0 we would find a value β that keeps the overtones within a given key with tonic f_{tonic}. In this case, β becomes a function of f_0. Thus $f_R(f_0) = f_0/\beta(f_0, f_{\text{tonic}})$. This becomes an extension of the search problem briefly mentioned in Section 4.1. Here we need to find a function or mapping $\beta(f_0, f_{\text{tonic}})$ such that the resulting overtones mostly fall on the target scale. This is an open-ended design problem that requires further study.

5. IMPLEMENTATION

The harmonizing methods covered in the previous section work by setting the frame rate f_R. It does not affect the STTR implementation itself. Only a front-end for controlling f_R needs to be added. For STTR, the real-time implementation presented in the previous paper [1] can be used. For STTR with 50% overlap-add, it was shown that the implementation only requires one delay line per channel and two reader pointers that are shared between the channels.[4]

5.1. Fixed f_R

To implement the STTR harmonizing effect with fixed f_R, the frame rate needs to be restricted to a reasonable range. The STTR implementation in [1] allowed for step sizes from 0.5 ms to 0.5 s. For user convenience, we changed the *window length* parameter, which is in milliseconds, to a MIDI note number ranging from 48 to 72, one octave below and above middle C (MIDI number 60). This corresponds to a step size of 1.9111 ms to 7.6445 ms. We add a *Fine Tune* parameter to allow a user to adjust f_R by $\pm$ 50 cents.

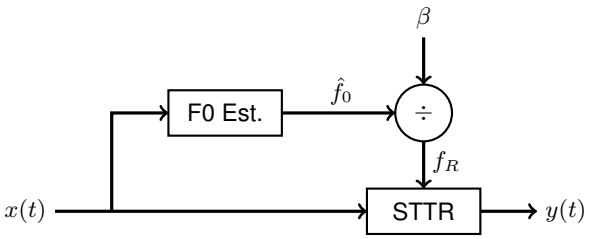
Figure 8: Audio plug-in for fixed f_R harmonizing effect. f_R parameter is labeled *Key*.

5.2. Fixed β

For the fixed β implementation, an F0 estimator is required to control the frame rate. For our implementation, we use the YIN algorithm [9]. We expose β, the semitone offset of between f_0 and f_R, as a user parameter. We add a *Fine Tune* parameter to adjust β by $\pm$ 50 cents.

While the YIN algorithm works well for single note input, it does not work well with signals with multiple fundamental frequencies. The decay from a previous note will often cause momentarily erroneous F0 estimates. While using a multiple F0 estimator [10, 11] can fix this problem, we are left with the problem of deciding the harmonizing strategy for multiple fundamental frequencies.

[4]Sound examples using the implementations explained can be found at `https://ccrma.stanford.edu/~hskim08/sttr/harmonize/examples.html`.



(a) Fixed β schematic

(b) Audio plug-in for fixed β harmonizing effect

Figure 9: Schematic and audio plug-in implementation of fixed β harmonizing effect. In Figure 9b, β parameter is labeled *Harmonize*. The F0 estimation is shown below the controls.

Each time the estimated f_0 changes, f_R and, in turn, the window length L needs to be updated. This change is accommodated by adjusting the positions of the windows and readers with respect to the writer position (Figure 10). This keeps the average lag between the input buffer readers and the output buffer writer to one window length. As a side effect, when the window length changes, there is a momentary pitch shift caused by speed change of the readers. To reduce discontinuity artifacts, L is gradually changed with a set slew rate.

6. CONCLUSION

For short window lengths, STTR creates overtones through sideband modulation. This property allows STTR to be used as a harmonizing effect without an explicit pitch shifting implementation. Thus, STTR can be used as an efficient method of implementing a harmonizing effect. We investigated the tonal relation of the overtones with respect to the frame rate and the frequency of a single sinusoidal input. These tonal relations can be saved as an STTR overtone table which then can be used as a guide when using STTR as a harmonizing effect. Three methods were proposed for harmonizing a signal using STTR of which two were covered in detail. The first method, fixed f_R, directly used an STTR overtone table for finding intervals that work for a given musical scale. The second method, fixed $\beta = f_0/f_R$, kept the overtones at a constant interval relative to the input signal. Implementation of the two methods requires little modification to the general STTR implementation. A front-end that controls the frame rate needs to be added which in the first case is trivial and in the latter case requires an F0 estimator.

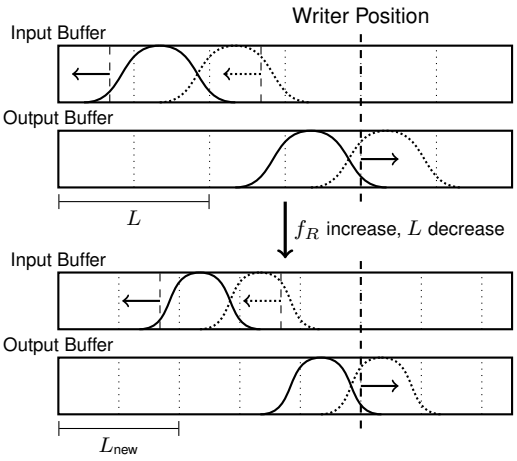


Figure 10: Illustration of STTR adjusting to an increase in the frame rate f_R. The reader and window positions of the input buffer are adjusted relative to the writer position of the output buffer, keeping an average lag of one window length. The arrows in the input/output buffers depict the reader/writer positions and directions respectively.

The analysis in this paper was focused on single sinusoids signals. The next step would be to analyze the effects of signals with harmonics. From our observations for STTR with 50% overlap-add using a Hann window, octaves generally had similar overtone relations to the input signal, though not strictly identical. However this cannot be said for the partials.

The requirements for constant overlap-add windows for the 50% overlap-add case was previously covered, outlining the possibility of designing windows [1]. By designing window functions tailored to STTR harmonizing, we could further control the overtones.

In Section 4.3, a variable β harmonizing scheme was briefly covered. Together with window design, this opens possibilities of extending STTR harmonizing such that it can harmonize input signals to a given scale without the limitations of the two implemented methods.

7. REFERENCES

[1] H. S. Kim and J. O. Smith, "Short-time time-reversal on audio signals," in *Proc. of 17th Int. Conf. on Digital Audio Effects (DAFx-14)*, Erlangen, Germany, September 2014.

[2] P. Dutilleux, G. De Poli, and U. Zölzer, "Time-segment processing," *DAFX: Digital Audio Effects*, pp. 201–236, 2002.

[3] A. Agnello, "Method and apparatus for producing two complementary pitch signals without glitch," Patent US 4 369 336, January 18, 1983.

[4] K. Bogdanowicz and R. Belcher, "Using multiple processors for real-time audio effects," in *Audio Engineering Society Conference: 7th Int. Conf.: Audio in Digital Times*. Audio Engineering Society, 1989.

[5] S. Disch and U. Zölzer, "Modulation and delay line based digital audio effects," in *2nd Workshop on Digital Audio Effects DAFx*. Citeseer, 1999.

[6] J. Laroche and M. Dolson, "New phase-vocoder techniques for pitch-shifting, harmonizing and other exotic effects," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 1999, pp. 91–94.

[7] A. Röbel and X. Rodet, "Efficient spectral envelope estimation and its application to pitch shifting and envelope preservation," in *Proc. of 8th Int. Conf. on Digital Audio Effects (DAFx-05)*, Madrid, Spain, September 2005.

[8] A. von dem Knesebeck and U. Zölzer, "Comparison of pitch trackers for real-time guitar effects," in *Proc. of 13th Int. Conf. on Digital Audio Effects (DAFx-10)*, Graz, Austria, September 2010.

[9] A. De Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[10] A. de Cheveigné, "Multiple f0 estimation," *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, pp. 45–72, 2006.

[11] A. P. Klapuri, "Multiple fundamental frequency estimation based on harmonicity and spectral smoothness," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 6, pp. 804–816, 2003.