Audio Engineering Society

Convention Paper

Presented at the 133rd Convention
2012 October 26–29 San Francisco, USA

A Tilt Filter in a Servo Loop

John Lazzaro[1] and John Wawrzynek[1]

[1] *Computer Science Division, UC Berkeley, Berkeley, CA, 94720-1776, USA*

Correspondence should be addressed to John Lazzaro (`lazzaro@cs.berkeley.edu`)

ABSTRACT

Tone controls based on the tilt filter first appeared in 1982, in the Quad 34 Hi-Fi preamp. More recently, tilt filters have found a home in specialist audio processors, such as the Elysia *mpressor*. This paper describes a novel dynamic filter design based on a tilt filter. A control system sets the tilt slope of the filter, in order to servo the spectral median of the filter output to a user-specified target. Users also specify a tracking time. Potential applications include single-instrument processing (in the spirit of envelope filters) and mastering (for subtle control of tonal balance). Although we have prototyped the design as an AudioUnit plug-in, the architecture is also a good match for analog circuit implementation.

1. INTRODUCTION

The Quad 34 Hi-Fi preamp [1], released in 1982, introduced the tilt filter to the audio world. The filter has two parameters. One parameter, the center frequency, is fixed at 800 Hz. The second parameter, *tilt*, is under user control. Tilt is a signed parameter, with units of dB/octave.

In the Quad 34 design, the filter has unity gain at 800 Hz, for all tilt values. The transfer function of the filter is a first-order shelving response. The tilt knob controls the slope of the shelf at its steepest point, which also occurs at 800 Hz. Positive tilt yields a high pass shelf; negative tilt, a low pass shelf; and zero tilt, a flat response.

The power of the tilt filter stems from the wide range of spectral control that is available by turning just one knob. More recent versions of the tilt filter, such as the filter in the *mpressor* audio processor [2] by the Elysia Corporation, make the tilt parameter even more powerful. This is done by morphing the shelving response into a 6 dB/octave low-pass or high-pass response at large tilt values.

We wondered if one could make an interesting audio effect by creating a dynamic tilt filter that places the tilt knob under servo control. This design concept taps the spirit of audio level compressors (from the professional audio world) and envelope-tracking filters (from the guitar effects world).

Taking an experimental approach, we created an AudioUnit plug-in implementation of a dynamic tilt filter. In this paper, we explain how the plug-in works, and analyze how the filter responds to audio examples of an isolated instrument source (a trombone loop) and complete program material (an excerpt from the Suzanne Vega song *Caramel* [3]). Although we prototyped the design as a plug-in, we constrained the design space so that the filter would be a good match for analog circuit implementation.

Fig. 1 shows the user interface for the plug-in. The user sets the center frequency (f_c) of the tilt filter with the *Spectral Center* slider. The *Treble Meter* slider is a real-time meter, that shows how the servo system varies the tilt over time. The *Tracking Time* and *Threshold* sliders let the user set the time response of the servo loop.

In normal operation, the servo system controls the tilt parameter so that the loudness of the filter output in the 20 Hz to f_c band equals the loudness in the f_c to 20 kHz band. Loudness is calculated using a simple perceptual model (linear equal-loudness filtering, temporal integration, and energy detection).

Under certain conditions, the servo law defined above is impossible to meet. This situation occurs when the input audio has minimal energy in the [20 Hz, f_c] and/or [f_c, 20 kHz] bands. The servo system detects this condition, and targets a flat filter response in this case. For reasonable assumptions about the tilt filter transfer function, we show that this modified servo law results in a locked loop.

The forward transfer function of our tilt filter implementation is a model of the filter in the Elysia *mpressor* [2]. The model was developed by Ivanov [4].

In practice, it is common for the gain of the servo-controlled tilt filter output to vary over ±20 dB. To tame these loudness transients, the plug-in includes a feedforward makeup gain system that gently levels the tilt filter output.

Our design was developed incrementally. Two sets of listening materials (single-instrument loops and full program material) were auditioned, to hear how different tilt filters, servo control laws, make-up gain blocks, and perceptual loudness models influence the audio output.

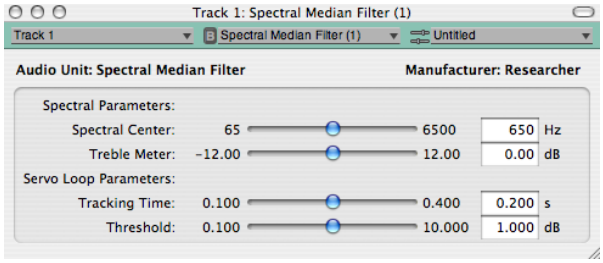

Fig. 1: The dynamic tilt filter. **Upper section:** Input slider for tilt filter center frequency (*Spectral Center*), output meter showing the tilt amount (*Treble Meter*). **Lower section:** Input sliders for servo control (*Tracking Time* and *Threshold*).

In addition to musical aesthetics, our other design constraint was amenability to analog circuit implementation. Our plug-in uses many of the tricks that an analog implementation would employ. For example, the analysis filters in the servo loop are 6 dB/octave low pass and high pass filters, that are available as internal signals in the tilt filter signal flow diagram.

Although our design was developed empirically, our work has links to research that takes a first-principles approach to audio effects, such as perceptual morphing [5] [6] and perceptually adaptive effects [7].

Viewed in this light, our servo system lets the user set the spectral median value of the filter output with the center frequency slider. As the spectral median is not a perceptual metric of timbre [8] [9] [10] [11], our design is not a perceptually-based effect. However, one could replace the spectral median with a perceptual metric, such as the spectral centroid. We did not take this approach, as we were interested in servo laws that would work well as analog circuits.

The rest of the paper is organized as follows. Section 2 introduces the architecture of the dynamic tilt filter. Sections 3-6 describe the operation of functional blocks in the design. Sections 7-9 present and analyze data from filter operation. Section 10 is a critical analysis of the performance of the plug-in, and Section 11 is the conclusion.

2. ARCHITECTURE

We begin with a formal definition of the spectral median. Consider a continuous-time signal $i(t)$ whose spectrum we denote as $I(f)$. $I(f)$ is band-limited to the range $0 < f < f_{\max}$. In this paper, we take $f_{\max}$ to be 20 kHz.

We define $\bar{I}(f)$ to be the root-mean-square (rms) energy of $I(f)$, measured over a short time period. We assume $\bar{I}(f) > 0$, for all f within the band-limit. Under this assumption, the spectral median f_i of $\bar{I}(f)$ has a unique value defined by the constraint:

$$\int_0^{f_i} \bar{I}(f)\mathrm{d}f = \int_{f_i}^{f_{\max}} \bar{I}(f)\mathrm{d}f. \tag{1}$$

Next, we introduce the dynamic tilt filter. The filter takes $i(t)$ as input and generates the continuous-time signal $m(t)$ as output. We denote the spectrum of $m(t)$ as $M(f)$. $\bar{M}(f)$ is the rms energy of $M(f)$, measured over a short time period, and f_m is the spectral median of $M(f)$.

The forward path of the filter is defined as:

$$M(f) = H(f, f_c, tilt_{\mathrm{db}})I(f) \tag{2}$$

where $H(f, f_c, tilt_{\mathrm{db}})$ is the transfer function of a parametric filter, with scalar parameters f_c and $tilt_{\mathrm{db}}$.

f_c is the spectral median target, set by the user via the *Spectral Center* slider. $tilt_{\mathrm{db}}$ is a variable under the control of the servo system. The *Treble Meter* slider displays the instantaneous value of $tilt_{\mathrm{db}}$.

The servo system adjusts $tilt_{\mathrm{db}}$ to satisfy the constraint:

$$\int_0^{f_c} \bar{M}(f)\mathrm{d}f = \int_{f_c}^{f_{\max}} \bar{M}(f)\mathrm{d}f. \tag{3}$$

A locked servo loop implies that $f_m = f_c$, realizing the design goal of the spectral median filter.

Next, we place the two constraints on the structure of $H(f, f_c, tilt_{\mathrm{db}})$:

$$|H(f, f_c, 0)| = 1 \tag{4a}$$

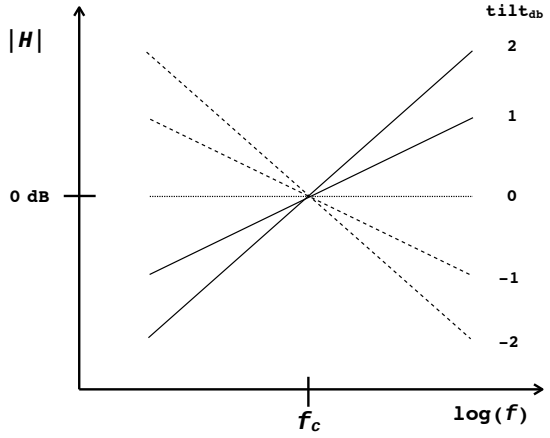

Fig. 2: Tilt filter transfer function $|H(f, f_c, tilt_{\mathrm{db}})|$.

$$\frac{\partial |H(f, f_c, tilt_{\mathrm{db}})|}{\partial f} = g(f, tilt_{\mathrm{db}}) \tag{4b}$$

where $g()$ is a strictly *increasing* monotonic function of $tilt_{\mathrm{db}}$ for $tilt_{\mathrm{db}} > 0$ and a strictly *decreasing* monotonic function of $tilt_{\mathrm{db}}$ for $tilt_{\mathrm{db}} < 0$. In addition, $g(f, 0) = 0$.

These constraints partition the $tilt_{\mathrm{db}}$ value space into three regions. The first region is the single point $tilt_{\mathrm{db}} = 0$, and corresponds to the condition $f_i = f_c$. In this region, Eqn. (4a) ensures that the loop locks, as it causes Eqns. (1) and (3) to be identical if and only if $f_i = f_c$.

The second region is defined by $tilt_{\mathrm{db}} > 0$, and corresponds to $f_i < f_c$. In this region, Eqn. (4b) forces $|H|$ to be a monotonically increasing function of f whose local slope monotonically increases with $tilt_{\mathrm{db}}$. Thus, the servo will always be able to find the $tilt_{\mathrm{db}}$ value that boosts high frequencies relative to low frequencies in such a way that $f_m = f_c$, and at this $tilt_{\mathrm{db}}$ value the loop locks. The third region, $tilt_{\mathrm{db}} < 0$ corresponding to $f_i > f_c$, is symmetrical to the second region, and the same arguments apply.

Fig. 2 sketches $|H(f, f_c, tilt_{\mathrm{db}})|$ for the case $g(f, tilt_{\mathrm{db}}) = g(tilt_{\mathrm{db}})$, with $g(tilt_{\mathrm{db}}) = -g(-tilt_{\mathrm{db}})$. The name *tilt filter* has been applied to transfer function families of this form.

Fig. 3 shows how the servo loop controls the tilt parameter in the dynamic tilt filter. In the waveform sketch, for times $t < t_o$ the loop is locked: $f_i = f_c$, and so $f_m = f_c$ and $tilt_{\mathrm{db}} = 0$.

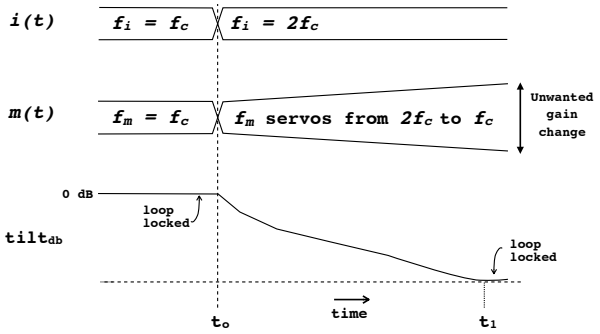

Fig. 3: Sketch of a servo action. At $t < t_o$, loop is locked. At $t = t_o$, the spectral median of the input undergoes a step change. Servo action re-locks the loop at $t = t_1$.

At $t = t_o$, the spectral median of $i(t)$ undergoes a step change to $f_i = 2f_c$. As a result, the loop loses lock: the value of $tilt_{\rm db}$ is still 0, and so $f_i = 2f_c$. To restore lock, the servo adjusts $tilt_{\rm db}$ downward until $f_m = f_c$, at time $t = t_1$.

In Fig. 4 we show the architecture of the dynamic filter. The *Tilt Filter* block is a realization of H. The *Listener Model* block uses a psychoacoustic model to generate time-averaged control signals from $i(t)$ and $m(t)$. The *Servo Control* block implements a control law for adjusting the *Tilt Filter* transfer function. The *Make-Up Gain* block implements a feed-forward algorithm for restoring $m(t)$ to unity gain (relative to $i(t)$).

The next several sections of the paper explain the mathematics that underlie these blocks. To create the plug-in software, we began by coding the equations in Structured Audio [12]. Structured Audio is a domain-specific language for audio that is a part of the MPEG-4 standard. We used the Structured Audio compiler `sfront` [13] to convert our program [14] into C-language source code for a common plug-in format (AudioUnits, Apple OS X operating system).

The plug-in is capable of both mono and stereo operation. In stereo operation, the *Listener Model* analyzes summed-to-mono versions of its stereo inputs, and the *Servo Control* and *Make-Up Gain* blocks run in mono. Thus, at any moment, the left and right channels of the tilt filter receive the same $tilt_{\rm db}$ value, and the make-up gain for the left and right channels is identical.

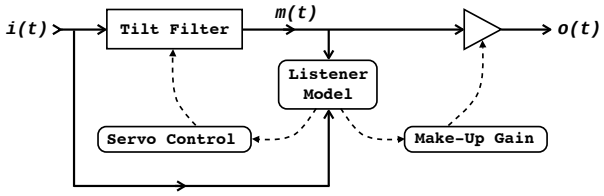

Fig. 4: Block diagram of the dynamic tilt filter. Dashed lines indicate control paths.

3. TILT FILTER DESIGN

Tone controls based on the tilt filter may be found in several audio products. An early example is the Quad 34 pre-amp, a consumer hi-fi component released in 1982 by Quad Electroacoustics Ltd. [1]. The Quad 34 features a single-pole tilt filter. A rotary control lets the listener apply ±3 dB of spectral tilt around a fixed center frequency of 800 Hz.

More recently, tilt filters have appeared in professional audio signal processing devices. For example, the *mpressor* [2], a multi-function effects device from the Elysia Corporation, contains a tilt filter.

Ivanov [4] created a discrete-time software model of the *mpressor* tilt filter, as shown in Fig. 5 and documented in the Appendix. We based the tilt filter in our plug-in on Ivanov's design. A simplified s-domain transfer function for this filter is:

$tilt_{\rm db} > 0$:

$$H(s) = \frac{[s/s_c]e^{tilt_{\rm db}/a} \; + \; e^{-\kappa(tilt_{\rm db}/a)}}{1 + [s/s_c]} \qquad \text{(5a)}$$

$tilt_{\rm db} < 0$:

$$H(s) = \frac{[s/s_c]e^{\kappa(tilt_{\rm db}/a)} \; + \; e^{-tilt_{\rm db}/a}}{1 + [s/s_c]} \qquad \text{(5b)}$$

$tilt_{\rm db} = 0$:

$$H(s) = 1 \qquad \text{(5c)}$$

where s_c is the root associated with f_c, $a = 6/\ln(2)$, and κ is a gain factor whose value we take as 5.

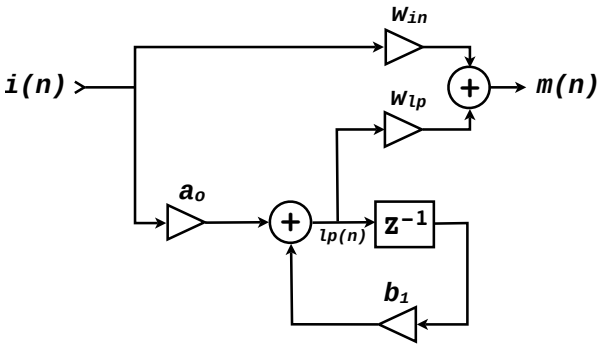

Fig. 5: Discrete-time tilt filter implementation, from [4]. a_o and b_1 are functions of f_c. $w_{\rm in}$ and $w_{\rm lp}$ are functions of $tilt_{\rm db}$. See Appendix.

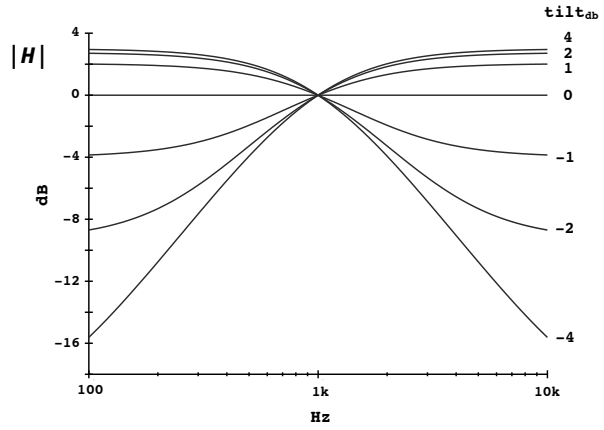

Fig. 6: Tilt filter transfer function, shown for $f_c =$ 1 kHz. The function is normalized so that $|H| = 1$ if $f = f_c$, for all values of $tilt_{\rm db}$.

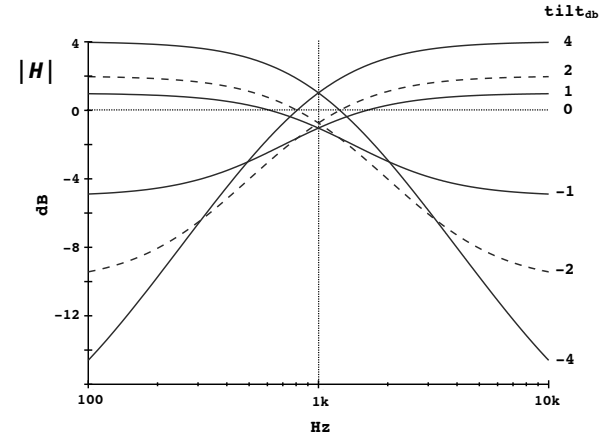

Fig. 7: Filter transfer function (no normalization).

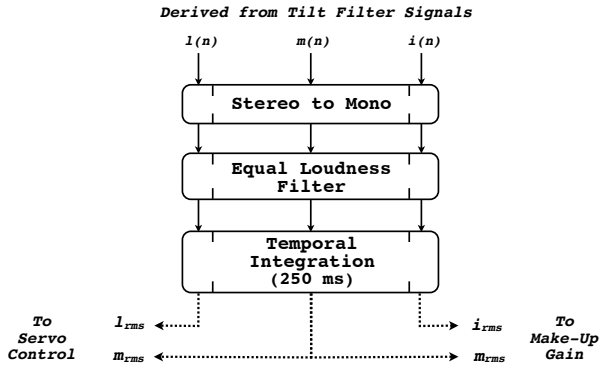

Fig. 8: Block diagram for the *Listener Model*.

A curve family for Eqn. (5) is shown in Fig. 6. All curves are normalized to be unity gain at $f = f_c =$ 1 kHz.

The $|tilt_{\rm db}| = 1$ curves are gentle shelf responses that are roughly symmetrical about the $tilt_{\rm db} = 0$ line. The $|tilt_{\rm db}| = 4$ curves are single-pole high pass and low pass responses. Intermediate $|tilt_{\rm db}|$ curves interpolate between the two response types.

The curves show why H is a good transfer function for servo control. The shelving response at small $|tilt_{\rm db}|$ values lets the servo respond to small $i(t)$ spectral changes in a subtle way. The low pass and high pass responses at large $|tilt_{\rm db}|$ values let the servo recapture lock after large $i(t)$ spectral changes.

In the plug-in, we use the tilt filter as defined in Eqn. (5), without the normalization used in Fig. 6, because we found that the unnormalized filter sounds better when operated under servo control. In Fig. 7 we plot the unnormalized curves, to show how the gain changes with $|tilt_{\rm db}|$.

4. LISTENER MODEL

The *Listener Model* (Fig. 8) monitors signals in the audio data path, and calculates signal statistics for use by the *Servo Control* and *Make-Up Gain* blocks.

Three audio signals are monitored: the tilt filter input $i(n)$, the tilt filter output $m(n)$, and $l(n)$. $l(n)$ is a low pass filtered version of $m(n)$, at a cutoff frequency of f_c. The respective outputs $i_{\rm rms}$, $m_{\rm rms}$ and $l_{\rm rms}$ are updated at a rate of 10 ms.

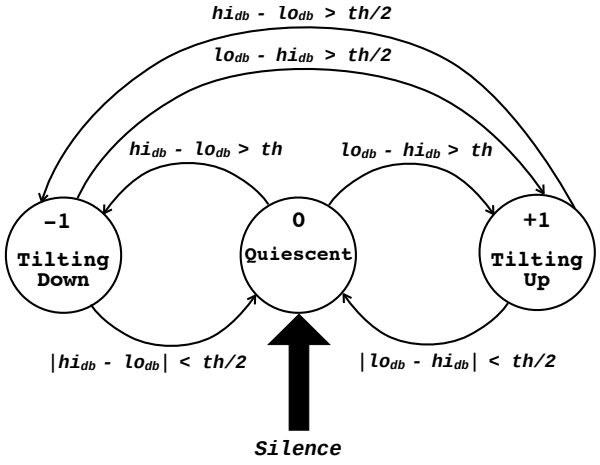

Fig. 9: State machine for the *Servo Control* block. If *Silence* is true, the thick arrow fires, superseding all other rules.

The *Servo Control* block uses $m_{\rm rms}$ and $l_{\rm rms}$ to estimate the spectral median of $m(t)$. The *Make-Up Gain* block uses $i_{\rm rms}$ and $m_{\rm rms}$ to determine the gain change of the tilt filter.

The slope of the low pass filter that generates $l(n)$ determines the accuracy of the spectral median estimation. In the example design, we use the single-pole low pass filter contained in the tilt filter to compute $l(n)$, via the approximation $l(n) \approx [w_{in} + w_{lp}]\, lp(n)$. Note that $lp(n)$, w_{in}, and w_{lp} appear in Fig. 5, and w_{in} and w_{lp} are defined in the Appendix.

The *Listener Model* processes all of its inputs in an identical fashion, as shown in Fig. 8. Stereo audio signals are first summed to mono. Then, the mono signals are filtered to model the frequency dependence of human loudness perception. We use two single-pole filters to form a simple approximation to the 80 dB SPL equal loudness curve [15]: a high pass with a 235 Hz cutoff, and a low pass with a 2 kHz cutoff.

Finally, to approximate the loudness integration time of the auditory system [16], the most recent 250 ms of each signal is stored in a circular buffer. Every 10 ms, the rms energy of each buffer is calculated to produce the output control value.

5. SERVO CONTROL BLOCK

The goal of the servo controller is to satisfy the spectral median constraint defined by Eqn. (3). At 10 ms intervals, the controller examines the l_{rms} and m_{rms} values produced by the *Listener Model*, and runs an algorithm to update the tilt filter parameter $tilt_{\rm db}$. We refer to this process as the *control cycle*.

The servo controller is organized as a state machine, as shown in Fig. 9. At any given time, the controller is executing an action to increase $tilt_{\rm db}$ (state *Tilting Up*), decrease $tilt_{\rm db}$ (state *Tilting Down*), or is in the *Quiescent* state. The variable `state` takes on values of +1, 0, or -1 to encode *Tilting Up*, *Quiescent*, and *Tilting Down*, respectively. To reset the servo controller, we set `state` $= 0$ and $tilt_{\rm db} = 0$.

At the start of a control cycle, the controller estimates the integrals of Eqn. (3), as:

$$lo_{\rm db} = 90 + 20\log_{10}(l_{rms} + \texttt{fl}) \qquad (6a)$$

$$hi_{\rm db} = 90 + 20\log_{10}(m_{rms} - l_{rms} + \texttt{fl}) \qquad (6b)$$

where $lo_{\rm db}$ estimates the left-hand side of (3), $hi_{\rm db}$ estimates the right-hand side of (3), and `fl` is the silence floor, expressed as a rms energy value. Eqn. (6) uses a dB scale where 90 dB corresponds to a rms energy of 1. The silence floor is at -30 dB.

We use $lo_{\rm db}$ and $hi_{\rm db}$ to update the state machine, as shown in Fig. 9. In the expressions shown in Fig. 9, *th* refers to the value of the *Threshold* slider. The *Threshold* slider lets the user control how tightly the servo meets the constraint of Eqn. (3).

The thick arrow in Fig. 9 forces the controller into the *Quiescent* state. This action occurs when the variable *Silence* is true:

$$Silence = (lo_{\rm db} < -27\,{\rm dB}) \;||\; (hi_{\rm db} < -27\,{\rm dB}). \quad (7)$$

In this equation, || is the logical OR function. Note that if the thick arrow fires, it supersedes all other arcs in the state machine

To complete the control cycle, the controller updates $tilt_{\rm db}$. In normal operation, *Silence* evaluates to false, and the controller updates $tilt_{\rm db}$ using the rule:

$$tilt_{\rm db} = tilt_{\rm db} + \frac{10\,{\rm ms}}{\tau} \times \texttt{state} \qquad (8)$$

where τ refers to the value of the *Tracking Time* slider.

When *Silence* evaluates to true, the constraint of Eqn. (3) is impossible to meet. In this case, the controller updates $tilt_{\mathrm{db}}$ to move closer to the $tilt_{\mathrm{db}} = 0$ value, using the rule:

$$tilt_{\mathrm{db}} = tilt_{\mathrm{db}} \; - \; \frac{10\,\mathrm{ms}}{\tau} \; \times \; \mathrm{sgn}[tilt_{\mathrm{db}}] \qquad (9)$$

where the sgn[x] function takes the value 1 if $x > 0$, 0 if $x = 0$, and -1 if $x < 0$.

The tilt filter implementation interpolates $tilt_{\mathrm{db}}$ values between control cycles, in order to reduce zipper noise.

Finally, we note the update rule defined in Eqn. 8 only works well in the regime where changes in $|tilt_{\mathrm{db}}|$ have a significant effect on $|H|$. For the $|H|$ shown in Fig. 7, this regime corresponds to $|tilt_{\mathrm{db}}| < 4$. In practice, this condition can be met by restricting the minimum value of *Tracking Time*. In our implementation, we use the restriction $\tau >= 100\,\mathrm{ms}$.

6. MAKE-UP GAIN SYSTEM

The average signal energy of $m(t)$, relative to $i(t)$, may vary over ±20 dB in normal operation. The *Make-Up Gain* block (shown in Fig. 4) is a feedforward gain control system that corrects this imbalance. The *Make-Up Gain* block operates on a 10 ms control cycle, in synchrony with the *Servo Control* control cycle.

At the start of a control cycle, the *Make-Up Gain* block calculates the gain imbalance as:

$$mi_{\mathrm{db}} = 20\log_{10}(m_{rms} + \texttt{fl}) - 20\log_{10}(i_{rms} + \texttt{fl}) \qquad (10)$$

where m_{rms} and i_{rms} are produced by the *Listener Model*, and `fl` is the silence floor (as in Eqn. (6)).

Next, the *Make-Up Gain* block updates a running histogram of mi_{db} values. The center of mass of the histogram is calculated, as the variable cm_{db}. This variable is used to rescale $m(t)$, via the equation:

$$o(t) = m(t) \; \times \; 10^{cm_{\mathrm{db}}/20}. \qquad (11)$$

To prevent zipper noise, the scaling factor in Eqn. (11) is interpolated between control cycles.

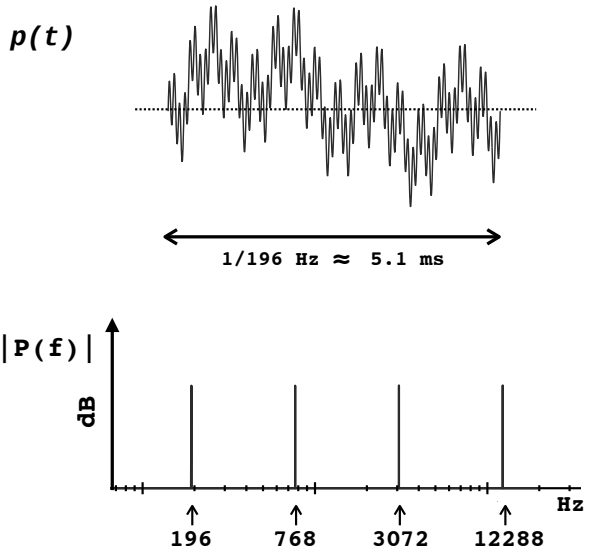

Fig. 10: Top: One cycle of the sparse harmonic tones time-domain waveform. **Bottom:** Spectrum of the sparse harmonic tones signal.

Under normal conditions, the mi_{db} histogram length is 1200 ms. However, when the *Silence* logic value (Eqn. (7)) transitions from false to true on successive control cycles, the histogram length is temporarily reduced to 320 ms. This change allows for faster adaptation to new program material.

7. STEADY-STATE RESPONSE

We measured the steady-state frequency response of the plug-in, as a function of the spectral median target value f_c. To do so, we used the periodic signal $p(t)$, shown in Fig. 10.

The harmonics of $p(t)$ are widely spaced in frequency (f_1, $4f_1$, $16f_1$, and $64f_1$, with $f_1 = 192$ Hz). Phases are chosen to reduce the crest factor of the waveform (0, $3\pi/2$, $\pi/8$, and $\pi/13.\bar{3}$ radians, respectively). All harmonics are of equal amplitude.

$p(t)$ has a multi-valued spectral median. Any spectral median value f_i over the range 768 Hz $< f_i <$ 3072 Hz fulfils Eqn. (1). This behavior occurs because $P(f) = 0$ over 768 Hz $< f_i <$ 3072 Hz, thereby violating the uniqueness condition for Eqn. (1). We construct $p(t)$ in this way to illuminate how the response of the plug-in differs from that of an ideal spectral median filter.

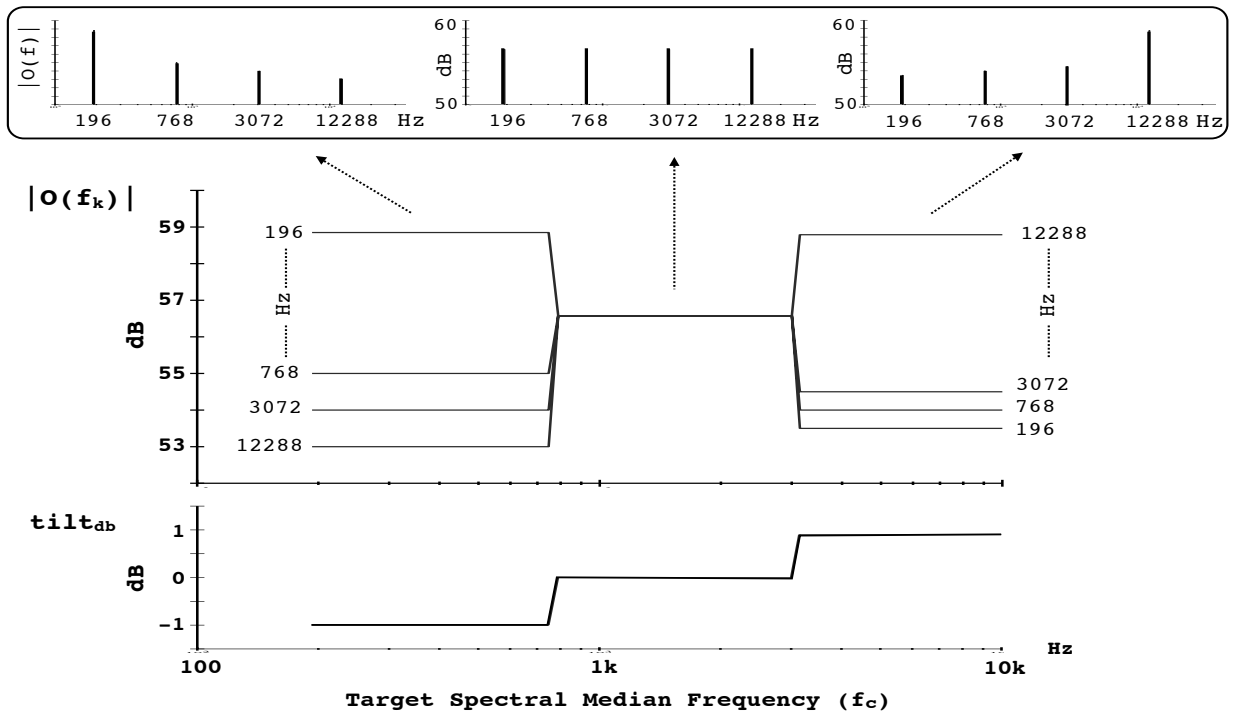

Fig. 11: Response of an ideal spectral median filter to $p(t)$. See main text for details.

To measure the steady-state filter response at a particular f_c, we set the *Spectral Center* slider to f_c, and applied $p(t)$ to the filter input. After waiting for the *Servo Control* and *Make-Up Gain* control systems to settle, we measured the rms energy of each harmonic of $o(t)$, using a lattice of notch filters. Unless otherwise noted, we disabled the equal loudness filters in the *Listener Model* for the measurements shown in this Section, in order to simplify the interpretation of the results.

Fig. 11 sketches the steady-state response of an ideal spectral median filter to $p(t)$. The $|O(f_k)|$ plot shows the output energy of the four stimulus harmonics ($k = 1$, 4, 16, and 64), as a function of f_c. The insets above the main plot show the complete output spectrum $|O(f)|$ for selected f_c values. The $tilt_{\mathrm{db}}$ plot provides insight into the state of the control system.

Examining the ideal response, for all f_c over the range 768 Hz $< f_c <$ 3072 Hz we find that $|O(f)| = |P(f)|$ and $tilt_{\mathrm{db}} = 0$. This response is a consequence of the multi-valued spectral median of $p(t)$. For f_c over the range 192 Hz $< f_c <$ 768 Hz, the control system of an ideal filter finds the $tilt_{\mathrm{db}} < 0$ value that balances the energy of the fundamental versus that of the upper harmonics, yielding an $|O(f)|$ with a downward tilt (shown in left inset). A symmetrical result is seen for f_c in the 3072 Hz $< f_c <$ 12228 Hz range (shown in right inset).

Fig. 12 shows the response of the plug-in. The inset $|O(f)|$ responses show a qualitative match with those of the ideal filter: a downward slope for a low f_c, an upward slope for a high f_c, and a flat spectrum for a mid-point f_c.

However, the main $|O(f_k)|$ plot departs from the ideal response in several ways. The $|O(f)| = |P(f)|$ condition corresponds to a single spectral median value ($f_c = 556\,\mathrm{Hz}$) that does not lie within the range of spectral median values for $p(t)$. These differences are due to the finite slope of the low pass filter used to generate $l(n)$, and the chosen definition of f_c as the -3dB point of the low pass response. In addition, all curves in the main $|O(f_k)|$ plot exhibit a characteristic non-monotonic response with f_c, due

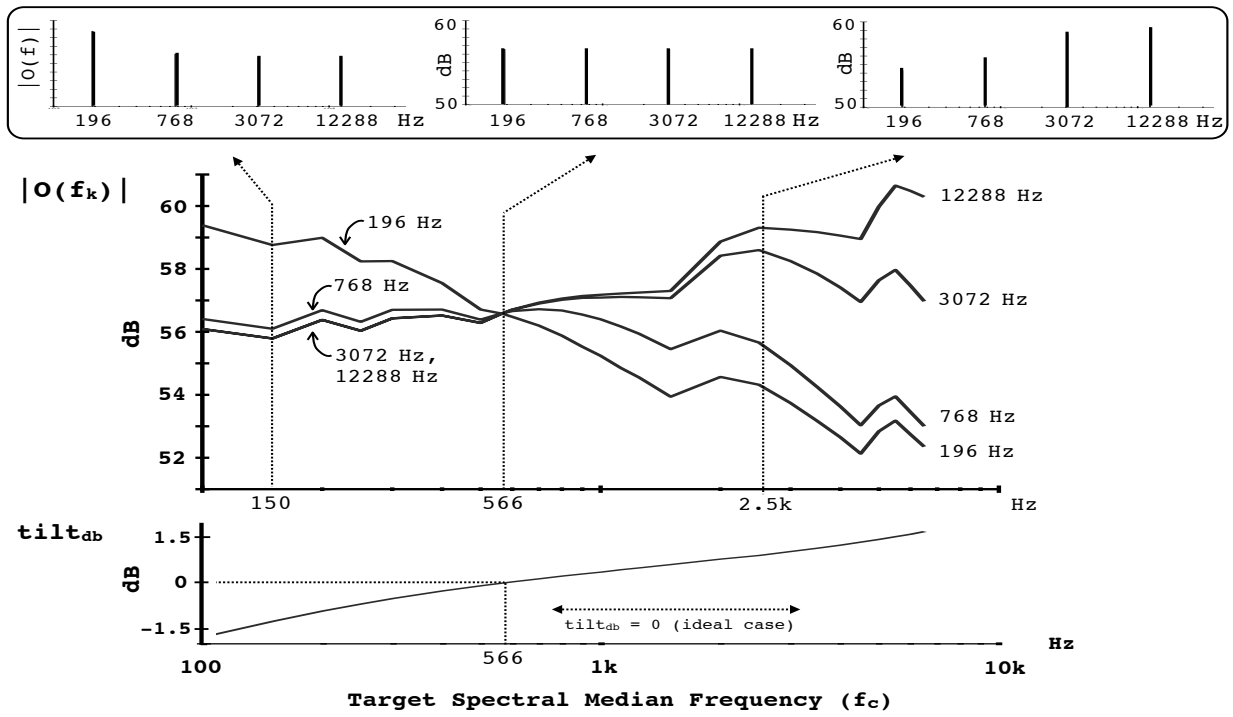

Fig. 12: Response of the plug-in to $p(t)$. Data layout as in Fig. 11.

to the action of the *Make-Up Gain* system. We discuss these differences in Section 10.

We also measured the plug-in response with the equal loudness filters of the *Listener Model* enabled. In Fig. 13, the lower plot $|O_w(f_k)|$ is the plug-in response with the equal loudness filters enabled. The upper plot $|O_u(f_k)|$) reproduces data from Fig. 12, for reference. In practice, the equal loudness filter acts to rescale the taper of the *Spectral Center* slider, to better match the expectations of a human listener.

8. DYNAMIC RESPONSE

In this Section, we use an isolated instrument recording to examine the dynamic behavior of the *Servo Control* and *Make-Up Gain* blocks of the plug-in. The stimulus [14] is a short recording of a trombone playing a steady eighth-note cadence in a low register.

We set the *Spectral Center* and *Threshold* sliders to values corresponding to $f_c = 65\,\text{Hz}$ and $th = 0.1\,\text{dB}$, and varied the *Tracking Time* slider over its range. The fastest *Tracking Time* (corresponding to $\tau = 100\,\text{ms}$) muted the high frequencies of the attack of alternating trombone notes. The slowest *Tracking Time* ($\tau = 400\,\text{ms}$) left the attack timbres unaffected, and mainly acted to decrease the high frequency energy of the sustained portion of the trombone notes.

Fig. 14 shows how the plug-in responds to the stimulus. Plots of $tilt_{\text{db}}$ are shown for several τ values. The envelope of the trombone recording is shown as $i(t)$.

For all τ, the average value of $tilt_{\text{db}}$ stabilizes to a value close to -3 dB (shown by the horizontal dotted lines), reducing the high frequency energy of the sustained portion of the trombone notes.

The triangle waveform superimposed on the average $tilt_{\text{db}}$ value is responsible for the spectral envelope shaping of note attacks. The amplitude of the triangle waveform corresponds to the degree of timbral shaping. The fastest *Tracking Time* ($\tau = 100\,\text{ms}$) has the highest-amplitude triangle waveform.

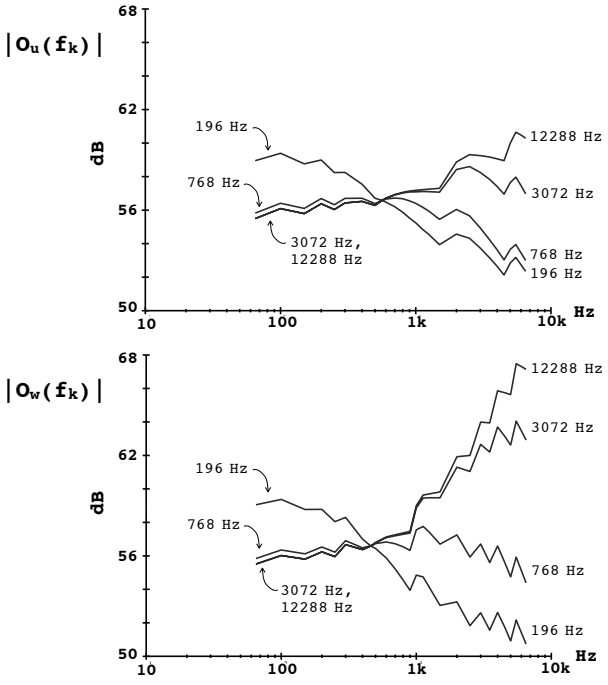

Fig. 13: **Top:** Plug-in response with the equal loudness filters of the *Listener Model* disabled. **Bottom:** Response with the equal loudness filters re-enabled.

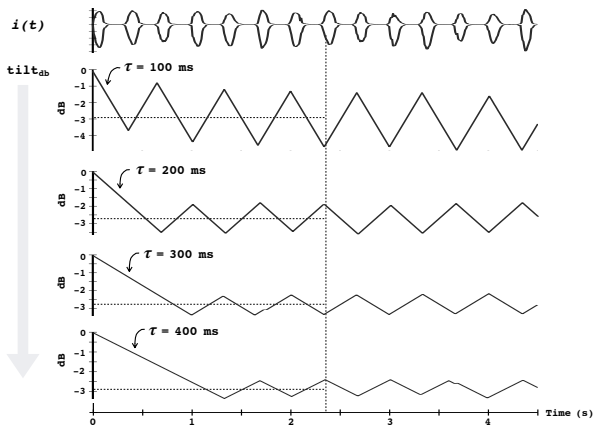

Fig. 14: Trombone recording [14] response, showing the effect of the *Tracking Time* slider.

After the $tilt_{\mathrm{db}}$ curves reach their stable average values, their triangle waveforms are synchronized. The τ value determines how quickly the $tilt_{\mathrm{db}}$ curve stabilizes. The relative phase of the $tilt_{\mathrm{db}}$ triangle waveforms depends on τ, as indicated by the vertical dotted line in Fig. 14. In a musical sense, the phase determines if the even notes or the odd notes of the cadence are muted.

Fig. 15 shows the internal operation of the *Servo Control* block for the $\tau = 200\,\mathrm{ms}$ condition. The $\int dt$ double-arrow near the bottom of the plot shows the 250 ms loudness integration time of the *Listener Model*. The temporal integration process acts to remember the energy value of most recent trombone attack. The flat tops of the hi_{db} and lo_{db} waveforms in the plot are indicative of this process.

When a trombone attack occurs, hi_{db} and lo_{db} update to reflect the amount of energy above and below f_c in the attack, respectively. Once $tilt_{\mathrm{db}}$ reaches a stable average value, each note attack causes the `state` variable to flip sign, due to the state machine triggering rules shown in Fig. 9. Following each `state` variable sign flip, the $tilt_{\mathrm{db}}$ triangle waveform changes direction, due to Eqn. (8).

Finally, Fig. 16 shows the internal operation of the *Make-Up Gain* block for the $\tau = 200\,\mathrm{ms}$ condition. The $mi_{\mathrm{db}} + gain$ plot shows the end-to-end gain of the plug-in. The *gain* plot shows the make-up gain that was added to $m(t)$.

A few seconds into the musical phrase, the $mi_{\mathrm{db}} + gain$ curve settles to an average value near 0 dB, meeting the goal of unity end-to-end gain. For the remainder of the cadence, the curve shows ± 2 dB gain steps, corresponding to alternating muted and normal trombone note attacks.

The gain steps, which lend an authenticity to the dynamic filtering effect, reflect the use of an unnormalized *Tilt Filter* transfer function (Fig. 7). The gain steps are not reduced by the make-up gain system, because the time between trombone notes is short compared to the length of the histogram used by the system (see "normal histogram" arrow in Fig. 16).

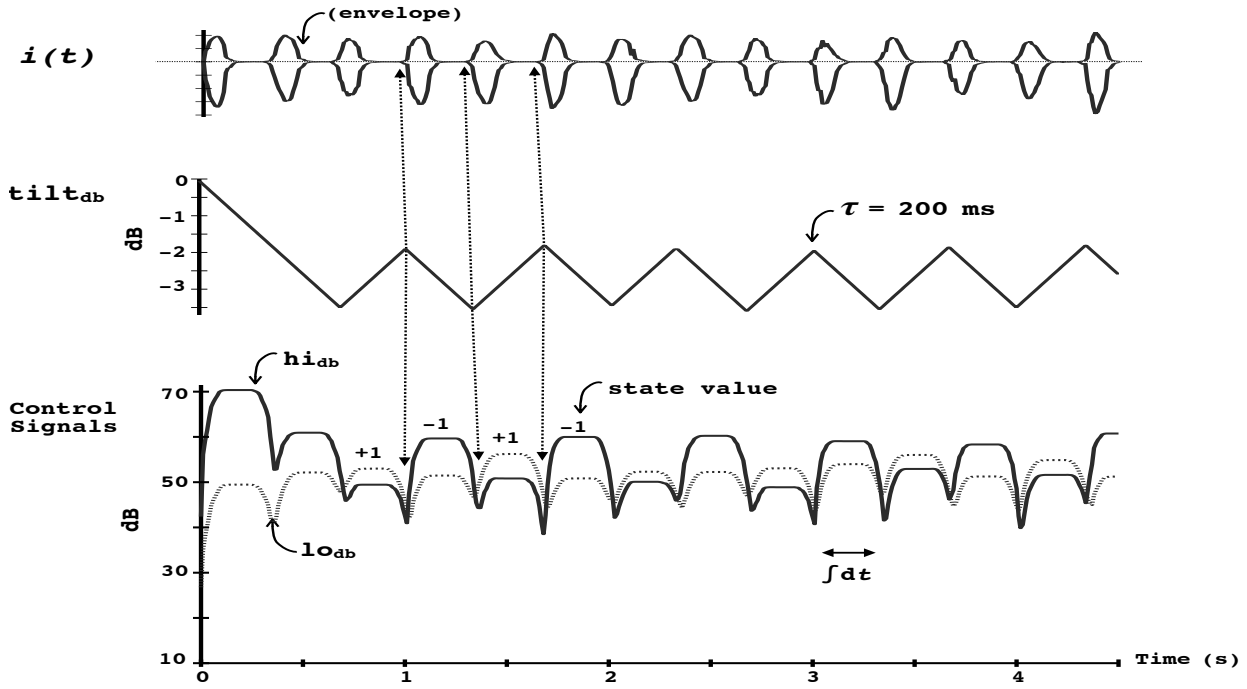

Fig. 15: Trombone recording response, showing signals in the *Servo Control* block. $\int dt$ indicates the loudness integration time of the *Listener Model*.

9. RESPONSE TO PROGRAM MATERIAL

In certain styles of music production, many tracks of isolated instrument recordings are blended together to create a stereo mix. Audio engineers use a variety of techniques to perceptually "glue" the isolated tracks together in a mix.

One technique engineers use to create "glue" is to apply moderate automatic level compression to the stereo mix. This technique works because the compression is usually triggered by a transient in one instrument in the mix, but the resulting gain reduction affects all instruments in the mix.

In this Section, we apply the spectral median filter to a complete musical performance, to show how dynamic filtering may be used in a similar way. We processed a commercial pop music recording, the song *Caramel* by Suzanne Vega [3] [14].

Our goal was to subtly "lift" the vocal out of the mix. We listened to the song as we explored the parameter space, and decided to set the *Tracking Time* slider to 200 ms, the *Spectral Center* slider to 650 Hz, and the *Threshold* slider to 1 dB.

Fig, 17 shows a 10-second excerpt from the song. The $tilt_{\rm db}$ curve has valleys near 0 dB (flat filter response) and peaks that range up to 2 dB (increased high frequency energy). The turn-around points of the $tilt_{\rm db}$ waveform tend to correlate to events in the input envelope $i(t)$. These points occur at a rate that is roughly related to the song's tempo.

Transients with dominant energy below the target spectral median value tend to trigger moves towards higher $tilt_{\rm db}$ values. The upward $tilt_{\rm db}$ moves tend to be reversed by onsets of the vocal we wish to enhance. In the filter output, the vocal onsets tend to have more high frequency energy because they tend to begin at elevated $tilt_{\rm db}$ values.

Fig. 17 also shows the operation of the *Make-Up Gain* system. The $mix_{\rm db} + gain$ curve shows an average end-to-end gain of 0 dB, as desired.

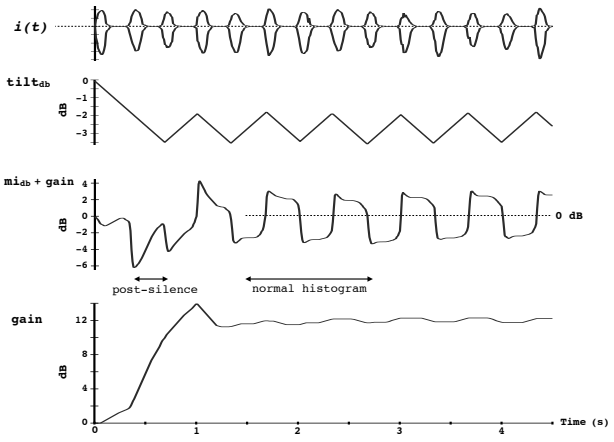

Fig. 16: Trombone recording response, showing *Make-Up Gain* signals. Arrows shows the histogram length in normal operation *(normal histogram)* and after a low-to-high transition of the `Silence` variable *(post-silence)*.

10. DISCUSSION

In this section, we describe ways to improve the plug-in. To begin, we consider the $l(n)$ input of the *Listener Model*. This signal is used to evaluate Eqn. (3). Ideally, $l(n)$ should be a filtered version of the tilt filter output, using a brick-wall low pass filter with a cutoff frequency of f_c.

In practice, the plug-in computes $l(n)$ using a single-pole low pass filter. As a result, some signal energy above f_c is erroneously assigned to $l(n)$. This error results in a $tilt_{\mathrm{db}}$ vs. f_c steady-state response that is closer to a line than a stair-step, as a comparison of the plug-in response (Fig. 12) and the ideal response (Fig. 11) indicates.

In practice, we have found the single-pole design to be satisfactory, based on listening sessions using a variety of musical source materials. However, an improved design would compute $l(n)$ with a Butterworth low pass filter, whose rolloff slope would be an option that the user may select.

Along the same lines, the plug-in could offer a choice of tilt filter transfer functions, each imparting a unique sonic "signature" to the filter output. The *Control Law* block could also offer a choice of algorithms, modeled after the different types of servo algorithms that are used in automatic level compressors.

Another way to enhance the plug-in would be to let the *Servo Control* and *Make-Up Gain* systems adapt to the program material. In fact, the design presented in this paper uses program adaptation in a simple way: the *Silence* variable (Eqn. (7)) is used to adapt to input silence. However, the concept could be taken much further.

The data in Fig. 16 shows the rationale for program adaptation in the *Make-Up Gain* block. An improved gain control system would quickly adapt at the start of this recording, to minimize the transient seen in the $mi_{\mathrm{db}}+gain$ plot, but then transition to longer adaptation times, to avoid gain modulation of the individual notes envelopes.

A similar argument could be made for program adaptation in the *Servo Control* block. Fig. 14 shows an isolated musical instrument recording whose spectral variation has two natural time constants: a musical phrase time constant that spans the example, and a shorter musical note time constant. An adaptive algorithm that senses note and phrase boundaries would yield a more "natural" and "musical" temporal response.

Finally, the plug-in could be improved through the use of more realistic psychoacoustic models in the *Listener Model*.

11. CONCLUSIONS

In this paper, we presented a dynamic filter that lets the user specify a target timbre using an audio descriptor (the spectral median). The paper is intended to serve as a starting point for the design of dynamic filters based on other audio descriptors, including some of the MPEG-7 descriptors listed in [11].

We limited the scope of the paper to signal processing for professional audio applications. However, the internal state variables of dynamic filters (for the spectral median plug-in, the $tilt_{\mathrm{db}}$, `state`, and *Silence* variables) have potential as sparse representations for audio pattern recognition systems. We hope to explore this application area in future work.

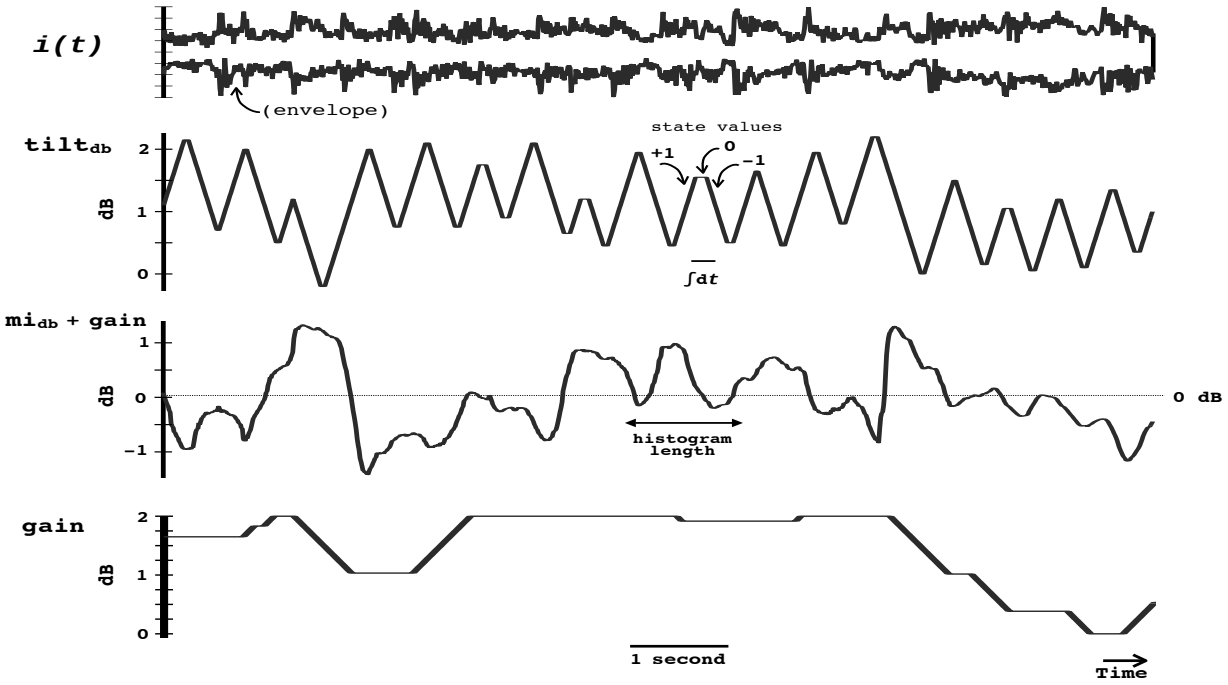

Fig. 17: Response to a 10 second excerpt of the song *Caramel* by Suzanne Vega [3] [14].

12. APPENDIX

In this Appendix, we document the tilt filter design used in the plug-in. The signal flow graph of the filter, shown in Fig. 5, is adapted from [4]. The a_o and b_1 coefficients in this graph are defined as:

$$a_o = \frac{2w_c}{3\,\mathtt{sr} + w_c}, \quad b_1 = \frac{3\,\mathtt{sr} - w_c}{3\,\mathtt{sr} + w_c}$$

where $w_c = 2\pi f_c$ and $\mathtt{sr}$ is the audio sampling rate. In the plug-in, $\mathtt{sr} = 44100$Hz.

The w_{in} and w_{lp} coefficients are defined in a piecewise fashion. For $tilt_{\mathrm{db}} = 0$, we set $w_{lp} = w_{in} = 1$. Otherwise:

$tilt_{\mathrm{db}} > 0$:

$$w_{lp} = e^{-\kappa(tilt_{\mathrm{db}}/a)} - e^{tilt_{\mathrm{db}}/a}, \quad w_{in} = e^{tilt_{\mathrm{db}}/a}$$

$tilt_{\mathrm{db}} < 0$:

$$w_{lp} = e^{-tilt_{\mathrm{db}}/a} - e^{\kappa(tilt_{\mathrm{db}}/a)}, \quad w_{in} = e^{\kappa(tilt_{\mathrm{db}}/a)}$$

where $a = 6/\ln(2)$, and κ is a gain factor we set to 5.

To derive an approximate s-domain transfer function for the filter, we formulate $\hat{H}(z)$ from the signal flow graph, and apply a bilinear transform to $\hat{H}(z)$ to generate $\hat{H}(s)$. Finally, we neglect a high-frequency zero in $\hat{H}(s)$ to produce the $H(s)$ function shown in Eqn. (5) in the main text.

13. ACKNOWLEDGEMENTS

We thank the reviewers of an earlier version of this manuscript for helpful suggestions.

14. REFERENCES

[1] Quad Electroacoustics Ltd, "Quad 34 Service Data Manual," 1982.

[2] Elysia GmbH, "mpressor User Manual," 2007.

[3] S, Vega, "Caramel," *Nine Objects of Desire*, A&M Records, track 3, 1996.

[4] L. I. Ivanov, "Simple Tilt Equalizer," *Music DSP Source Code Archive.* `http://www.musicdsp.org/showone.php?id=267`

[5] D. Williams and T. Brooks, "Perceptually motivated audio morphing: brightness," *122nd Convention of the Audio Engineering Society,* Vienna, Austria, May 5-8, 2007.

[6] D. Williams and T. Brooks, "Perceptually motivated audio morphing: softness," *126nd Convention of the Audio Engineering Society,* Munich, Germany, May 7-10, 2009.

[7] V. Verfaille, U. Zolzer,and D. Arfib "Adaptive Digital Audio Effects (A-DAFx): A New Class of Sound Transformations" *IEEE Trans. Audio, Speech and Language,* **14**:5, 2006.

[8] D. L. Wessel, "Psychoacoustics And Music: A Report From Michigan State University," *Bulletin of the Computer Arts Society,* 1973.

[9] J. M. Grey, "Multidimensional perceptual scaling of musical timbres," *The Journal of the Acoustical Society of America,* **61**:5, 1977.

[10] J. C. Brown, O. Houix, and S. McAdams, "Feature dependence in the automatic identification of musical woodwind instruments," *The Journal of the Acoustical Society of America,* **109**:3, 2001.

[11] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," *CUIDADO IST Project Report,* 2004.

[12] ISO 14496 (MPEG-4), Part 3 (Audio), Subpart 5 (Structured Audio), 1999.

[13] J. Lazzaro and J. Wawrzynek, "Compiling MPEG 4 structured audio into C," *Proceedings of the 2nd IEEE MPEG-4 Workshop and Exhibition,* June 2001.

[14] Structured Audio source code for the plug-in, and audio files for Fig. 14 and Fig. 17, are available at `http://www.cs.berkeley.edu/~lazzaro/tilt`

[15] ISO 226:2003, "Acoustics – Normal equal-loudness-level contours," *International Organization for Standardization,* 2nd edition, 2003.

[16] N. F. Viemeister, "Temporal integration and multiple looks," *The Journal of the Acoustical Society of America,* **90**:2, 1991.