Effekt – Implementation of Different Filters

Lucan Cristian, Maracine Razvan

February 5, 2026

Contents

1 Abstract

This paper presents different types of audio filters, the math and physics behind their implementation, possible variations and their effect on diverse audio samples using different parameters.

2 Samples used

- Crawling, Linkin Park (20s–34s).

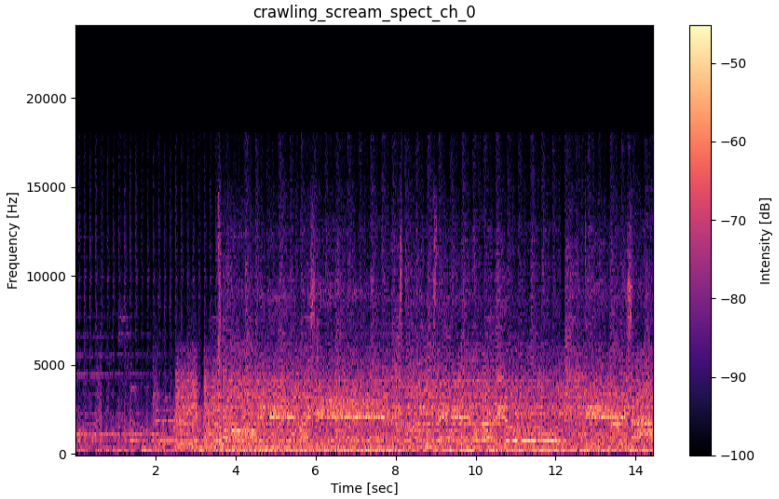

Figure 1: Spectogram for crawling

- Electronic guitar interpretation of Seven Nation Army(first 10s)

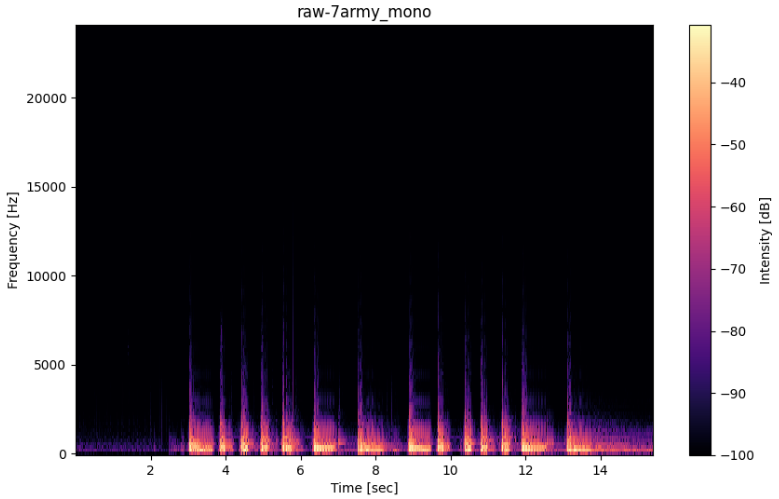

Figure 2: Spectogram for raw electronic guitar Seven Nation Army

- Calm Guitar downloaded from the internet.

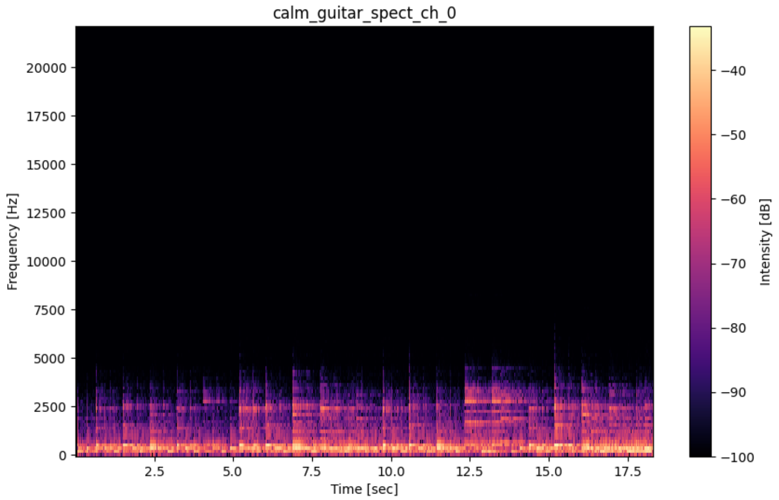

Figure 3: Spectogram for calm guitar

3 State Variable Filter (SVF)

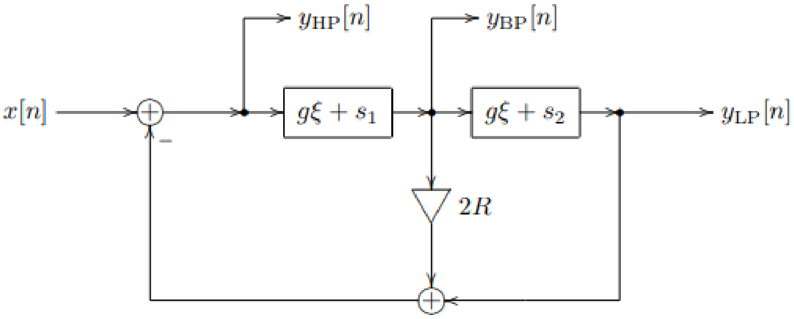

Figure 4: SVF Discrete-Time Model (ZDF Topology)

3.1 Description

The State Variable Filter (SVF) is a highly versatile topology that simultaneously computes three primary filter outputs: low-pass (y_{LP}), high-pass (y_{HP}), and band-pass (y_{BP}).

Based on the Zero-Delay Feedback (ZDF) structure, the band-pass output is derived as:

$$y_{BP} = g(x - 2Ry_{BP} - gy_{BP} - s_2) + s_1$$

Solving for y_{BP} yields the stable implementation formula:

$$y_{BP} = \frac{g(x - s_2) + s_1}{1 + g(g + 2R)}$$

From this, the low-pass and high-pass outputs are extracted:

$$y_{LP} = g \cdot y_{BP} + s_2$$
$$y_{HP} = x - 2R \cdot y_{BP} - y_{LP}$$

This filter utilizes two trapezoidal integrators, represented as $g\xi + s_i$, where:

- g: The zero-delay feedback gain.
- ξ: The instantaneous input of the integrator.
- s_1, s_2: The internal states (memory) of the integrators.

The parameter g represents the cutoff frequency after **frequency prewarping**, which maps analog frequencies (0 to ∞) to digital frequencies (0 to Nyquist) using:

$$g = \tan\left(\frac{\text{cutoff} \cdot \pi}{2}\right)$$

3.2 Results

Below are the spectrograms for the outputs of the SVF filter applied to the crawling sample from Figure 1

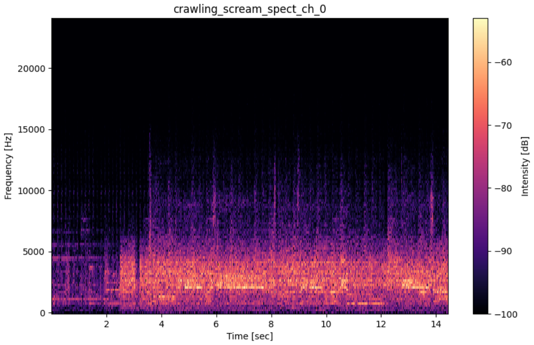

Figure 5: Band-pass output (Cutoff: 0.15, Resonance: 0.5)

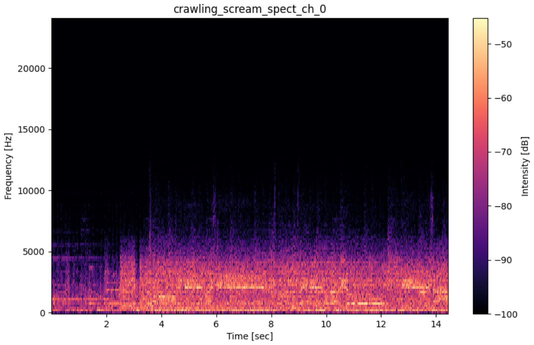

Figure 6: Low-pass output (Cutoff: 0.15, Resonance: 0.5)

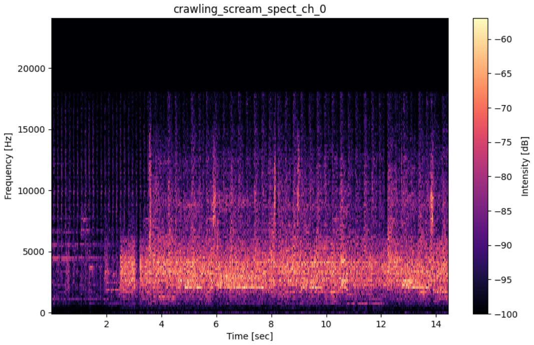

Figure 7: High-pass output (Cutoff: 0.15, Resonance: 0.5)

3.3 Resonance and Damping

The variable R represents the **damping factor** (the mathematical inverse of resonance). It controls the amount of band-pass signal fed back into the summing junction. While the cutoff controls the frequency boundary, R controls the amplitude peak at that boundary.

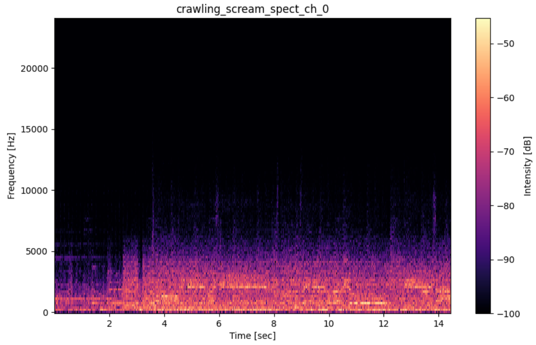

Figure 8: Low-pass (Resonance: 0.01)

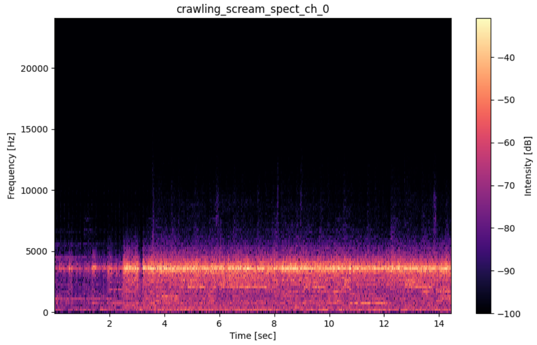

Figure 9: Low-pass (Resonance: 0.99) – Note the resonant peak.

When resonance is high, an intense line appears at the cutoff frequency.

3.4 Secondary Filter Types

By linear combination of the primary SVF outputs, we can derive specialized filters.

Band-Shelving Filter

Using a gain parameter K, we can boost or cut the band-pass region:

$$y_S = x + 2RK \cdot y_{BP}$$

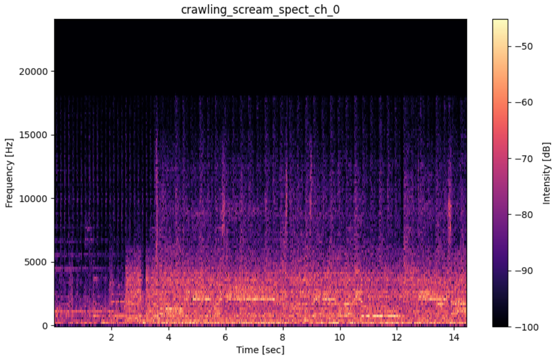

Figure 10: Band-shelving (Cutoff: 0.15, Res: 0.99, K: 0.5)

Notch Filter

A notch filter is achieved by setting $K = -1$, effectively subtracting the resonant peak from the input:

$$y_N = x - 2R \cdot y_{BP}$$

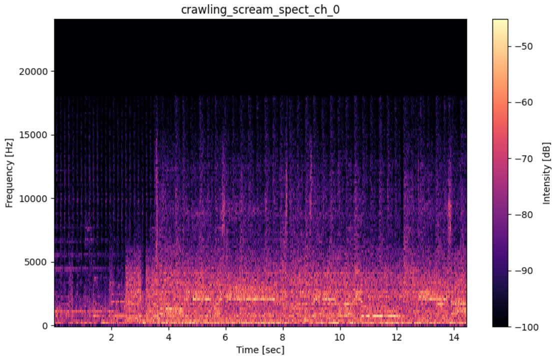

Figure 11: Notch filter (Cutoff: 0.15, Res: 0.99)

All-Pass Filter

Setting $K = -2$ creates an all-pass response, affecting only phase:

$$y_{AP} = x - 4R \cdot y_{BP}$$

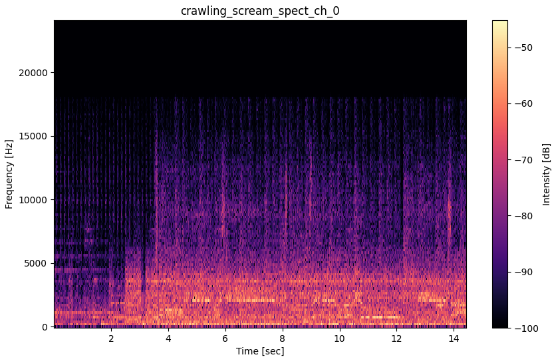

Figure 12: All-pass filter (Cutoff: 0.15, Res: 0.99)

Peaking Filter

The peaking (or phase-flip) filter is derived from the difference of the low-pass and high-pass signals:

$$y_P = y_{LP} - y_{HP}$$

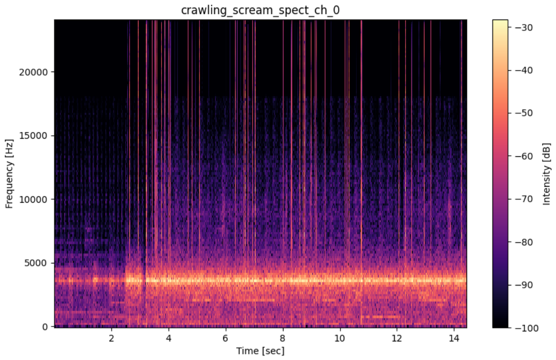

Figure 13: Peaking filter output

4 Binaural Rotation

This implementation is not a spectral filter, but rather a spatialization technique designed to simulate a monaural sound source moving horizontally (360° azimuth) around a listener's head. By transforming a mono input into a stereo output, we simulate the two primary cues used in human sound localization: Interaural Time Difference (ITD) and Interaural Level Difference (ILD).

4.1 Interaural Time Difference (ITD)

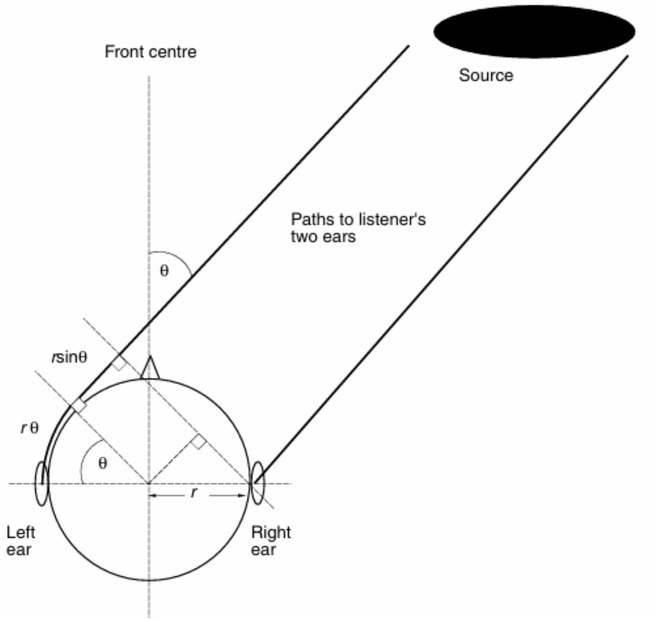

Figure 14: Geometric representation of ITD based on the spherical head model.

ITD represents the time delay incurred because the signal must travel different distances to reach each ear based on the source's position. As seen in Figure 14, a signal originating from the right will reach the right ear significantly earlier than the left.

Based on Woodworth's formula, the time delay for a spherical head is calculated as:

$$ITD = \frac{r}{c}(\theta + \sin\theta) \tag{1}$$

Where:

- r is the radius of the listener's head (typically ≈ 0.0875 m).
- c is the speed of sound (approximately 343 m/s).
- θ is the azimuth angle in radians (0 radians being directly in front).

The $\sin\theta$ term accounts for the direct line-of-sight path to the near ear, while the θ term represents the additional distance the sound wave must travel as it diffracts around the curved surface of the head to reach the "shadowed" far ear. In this implementation, the delay is applied only to the ear further from the source; for example, if $\sin(\theta) > 0$ (source in the right hemisphere), a delay is applied to the left channel.

4.2 Interaural Level Difference (ILD)

ILD models the variation in sound pressure level (amplitude) reaching each ear. The ear closer to the source perceives a higher amplitude, while the further ear perceives a lower amplitude due to the "head shadow" effect. To maintain a natural perception of volume during rotation, a Constant Power Panning law is utilized:

$$G_{left} = \cos\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \tag{2}$$

$$G_{right} = \sin\left(\frac{\theta}{2} + \frac{\pi}{4}\right) \tag{3}$$

These formulas leverage the trigonometric identity $\sin^2\theta + \cos^2\theta = 1$ to ensure that the total perceived power remains constant regardless of the angle. A 45° ($\pi/4$) phase shift is applied to the coordinate system to ensure that at $\theta = 0$ (center), both gains are equal to ≈ 0.707 (-3 dB), creating a stable phantom center.

4.2.1 Acoustic shadow

The human head blocks high frequencies and lets lower frequencies pass just like a low pass filter when the sound is further from the ear. This is especially true when it comes from behind(most notably the pina blocks back sounds by design).

For this two low pass filters (coming from a SVF) are used, one for the left, one for the right ear. The cutoffs for the filters are defined as follows:

- **normal cutoff**: 0.99 (transparent)
- **shadow cutoff**: 0.35 (shadow effect based on whether the sound is left or right)
- **rear cutoff**: 0.2 (amplified shadow effect when in the rear side)

The cutoff "sweeped" from normal to shadow and from shadow to rear based on the left factor($\sin\theta$), right factor ($-\sin\theta$) and the rear factor ($(1 - \cos\theta) * 0.5$) using linear interpolation. First sweeping from normal cutoff to side cutoff using the side factor and then from the result of that to the rear cutoff using the rear factor.

4.3 Results

Below are the spectrograms for the outputs of the binaural rotation filter applied to the crawling sample from Figure 1

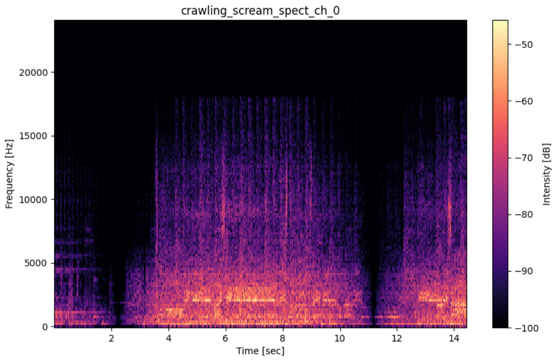

Figure 15: Binaural rotation without svf low pass filtering left channel

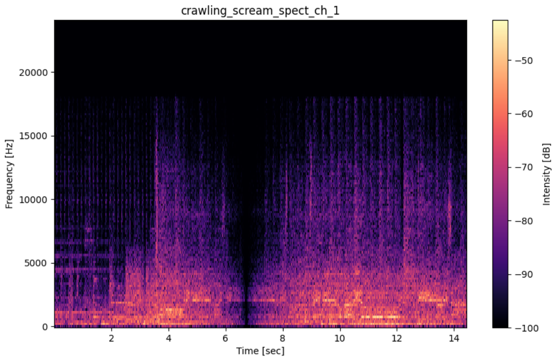

Figure 16: Binaural rotation without svf low pass filtering right channel

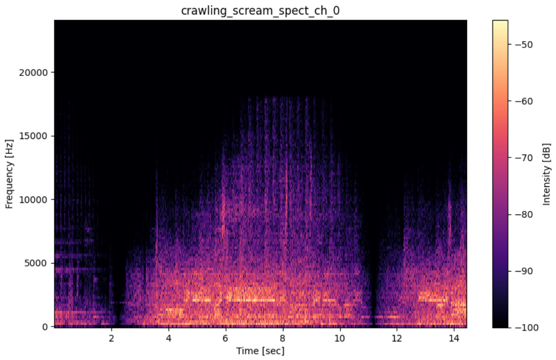

Figure 17: Binaural rotation with svf low pass filtering left channel

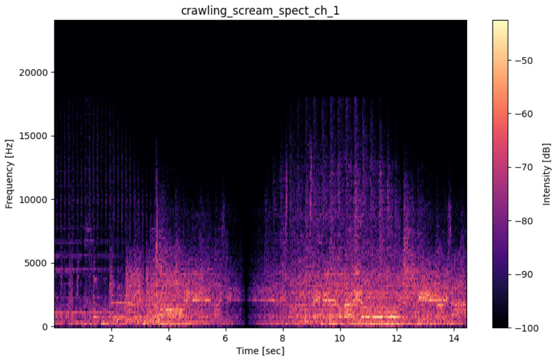

Figure 18: Binaural rotation with svf low pass filtering right channel

Most notable are the difference in phase between the channels and that for the version that uses low pass filtering we see the frequencies steadily decrease and increase.

5 Envelope Follower / Peak Detector

While not a filter in the spectral sense, the Envelope Follower is a circuit that automatically detects and traces the contour (peaks) of the audio signal. This component is essential for dynamic effects, such as the automatic wah (crybaby) effect, where the filter cutoff frequency needs to respond to the playing intensity.

The behavior of the envelope follower is divided into two distinct phases, **Attack** and **Release**. In both phases, the envelope moves towards the instantaneous amplitude of the input signal, but usually at significantly different speeds:

- **Attack Phase:** Occurs when the absolute value of the input $|x[n]|$ is *greater* than the current envelope $e[n-1]$. The envelope rises quickly to catch the transient.

- **Release Phase:** Occurs when the input drops *below* the current envelope. The envelope decays slowly, smoothing out the signal to prevent "zipper noise" or distortion.

In digital implementation, these speeds are defined by a time constant τ (measured in seconds), which represents the time it takes for the value to reach approximately 63.2% of the target step. To simulate the physics of a real capacitor charging and discharging in an analog RC circuit, this time constant is converted into a filter coefficient using an exponential function dependent on the sampling rate f_s:

$$\alpha = e^{-\frac{1}{\tau \cdot f_s}} \tag{4}$$

Where:

- α is the resulting Attack or Release coefficient (between 0 and 1).
- τ is the desired time constant in seconds.
- f_s is the sampling rate in Hz.

This coefficient α is then used to perform a "leaky integration" (a weighted average), which linearly interpolates between the previous envelope state and the current input. The logic branches based on the signal direction:

$$e[n] = \begin{cases} \alpha_{att} \cdot e[n-1] + (1 - \alpha_{att}) \cdot |x[n]| & \text{if } |x[n]| > e[n-1] \quad \text{(Attack)} \\ \alpha_{rel} \cdot e[n-1] + (1 - \alpha_{rel}) \cdot |x[n]| & \text{if } |x[n]| \leq e[n-1] \quad \text{(Release)} \end{cases} \tag{5}$$

Where $e[n]$ is the current envelope value and $|x[n]|$ is the rectified (absolute) input signal.

6 Crybaby Effect (Wah-Wah)

The Crybaby effect is digitally implemented by applying a band-pass filter to the signal while simultaneously sweeping the cutoff frequency and resonance (Q-factor). The control mechanism creates the characteristic vocal-like sound.

The control signal, often represented as a "knob" value between 0 and 1, is typically mapped to a **triangular sweep**. This is done to simulate the physical motion of a real Wah pedal: as the player rocks the pedal back and forth, the value rises linearly to the maximum and falls linearly back to the minimum, creating a smooth, continuous modulation rather than an abrupt jump.

6.1 Resonance Sweep

For a convincing Crybaby effect, a relatively **high resonance** is preferred. This is because the "Wah" sound mimics human speech formants (specifically the transition from vowel 'u' to 'a'). A high Q-factor creates a sharp, narrow peak in the frequency spectrum, which is necessary to emulate these resonant vocal cavities.

The resonance is dynamically calculated using the following linear relationship:

$$r[n] = r_{start} + r_{factor} \cdot s[n] \tag{6}$$

Where:

- $r[n]$ is the current resonance (Q) value.
- r_{start} is the base resonance value (0.85)
- r_{factor} is the **Resonance Sensitivity Factor**. This parameter defines the *depth* of the modulation, determining how much the sharpness of the filter increases as the sweep reaches its peak.
- $s[n]$ is the current value of the triangular sweep (0.0 to 1.0).

6.2 Cutoff Sweep

The core of the effect lies in moving the center frequency of the band-pass filter. I explored two distinct methods for controlling this movement.

6.2.1 Periodic Exponential Cutoff

My first implementation used an automatic Low Frequency Oscillator (LFO) to move the filter. The triangular sweep control signal $s[n]$ was mapped to the frequency domain using an exponential curve:

$$f_c[n] = f_{start} \cdot \left(\frac{f_{end}}{f_{start}}\right)^{s[n]} \tag{7}$$

The exponential function is used because human hearing perceives pitch logarithmically. If a linear sweep were used (e.g., adding a fixed number of Hertz per step), the filter would appear to move very slowly through the bass frequencies and rush instantly through the high frequencies. An exponential sweep ensures that the musical interval (octaves per second) remains constant, resulting in a smooth, natural-sounding transition across the entire range.

6.2.2 Envelope Follower Cutoff (Auto-Wah)

The periodic LFO attempt successfully created a wah sound, but it was robotic and disconnected from the musical performance. To resolve this, I switched to an **Auto-Wah** architecture.

Instead of a pre-calculated triangular wave, this method uses the **Envelope Follower** described in the previous section to drive the cutoff frequency. This allows the effect to react to playing dynamics in real-time: picking a string harder generates a higher envelope value, which opens the filter to a higher frequency immediately.

This dynamic interaction creates the expressive "quack" sound associated with funk music.

6.3 Results

For all results the following parameters were kept constant:

- **Base resonance: 0.85**
- **Resonance factor: 0.12**

Below are the spectrograms for the outputs of the crybaby effect applied to the raw electronic guitar interpretation of Seven Nation Army, Figure 2

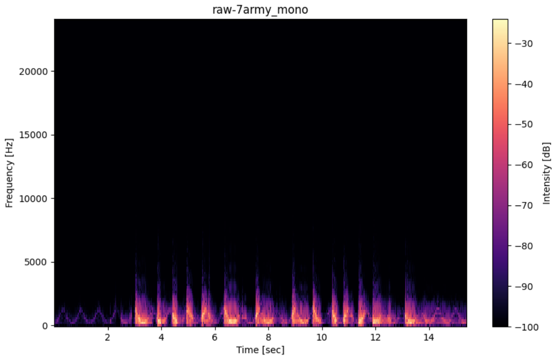

Figure 19: Crybaby effect using periodic exponential sweep, sweep speed=1.5 Hz/s, cutoff start=450Hz, cutoff end=2500Hz

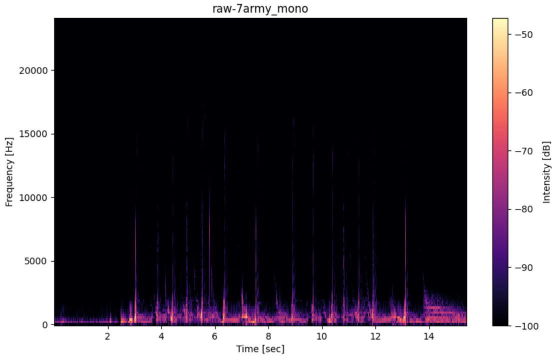

Figure 20: Crybaby effect using envelope follower, attack period=10ms, release period=100ms

7 Bitcrusher

The Bitcrusher is a digital-only effect that intentionally degrades audio quality by reducing both the vertical resolution (amplitude) and horizontal resolution (time) of the signal.

7.1 Quantization (Bit Depth Reduction)

The "bit" in "bitcrusher" refers to the binary representation of audio. In digital systems, dynamic range is defined by bit depth. We use **bits** as the parameter rather than direct integer levels for two primary reasons:

- **Perceptual Scaling:** Human hearing is logarithmic. Adding 1 bit of resolution consistently provides a 6dB improvement in the signal-to-noise ratio, making bits a more musically intuitive unit than raw steps.
- **Hardware Emulation:** Using bits allows the filter to accurately mimic specific vintage hardware, such as 8-bit samplers or 12-bit drum machines, which are defined by their hardware architecture.

The formula for quantization of a normalized signal $x \in [-1, 1]$ is:

$$y = \frac{\text{round}(x \cdot 2^{b-1})}{2^{b-1}}$$

- y: output signal
- x: input signal
- b: number of bits (resolution)

7.2 Dry/Wet Interpolation

At the end of the processing chain, an interpolation is performed between the original input (x) and the crushed output (y). This is implemented to provide a "Mix" control because:

- **Parallel Processing:** Bitcrushing is extremely destructive. Mixing in the "Dry" signal allows the listener to retain the fundamental clarity and transient punch of the original audio while adding the "crushed" texture as an overlay.
- **Harmonic Control:** Since bitcrushing adds significant high-frequency noise and aliasing, interpolation acts as a manual intensity knob, making the effect usable in a musical mix without over-powering other instruments.

7.3 Results/Observations

Playing around with the number of bits, I observed that leaving it between 0 and 4 does not offer much change because at such low resolution, the signal essentially becomes a square wave or pure noise. The same can be said about going after 16; since 16-bit is CD quality, the human ear struggles to hear further increases in resolution in a typical listening environment.

Changing the sampling rate shows a more rapid change that "worsens the quality of the signal" through aliasing. Interestingly, as I decreased the sample rate, Chester's voice became more aggressive. This is due to the aliasing adding metallic, grit-like frequencies to the mid-range, which emphasizes the "fry" in the vocal delivery.

Here is the spectogram for the one that sounded "the best" on crawling (spectogram for initial sample: Figure 1):

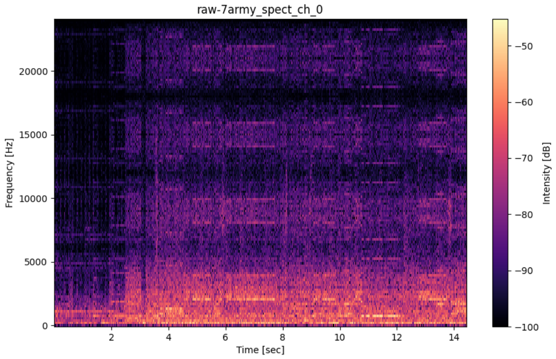

Figure 21: Bitcrusher on crawling, downsampling=8 samples, bit count=8

But I think this effect was better suited for the calm guitar sample (spectogram for initial sample: Figure 3):

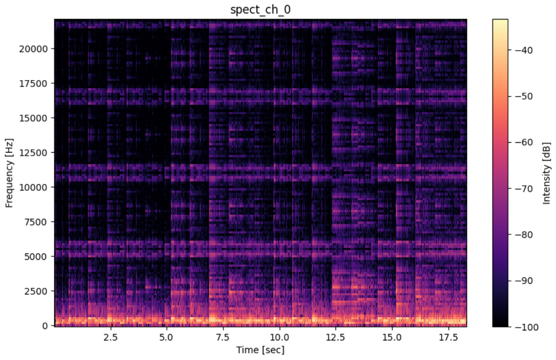

Figure 22: Bitcrusher on calm guitar, downsampling=8 samples, bit count=8

8 Cabinet Simulation via Convolution

8.1 Concept

To implement a realistic guitar cabinet simulation in software, we utilize **Convolution**. This way we can apply the acoustic characteristics of a physical speaker cabinet to a raw digital audio signal.

The core component of this simulation is the **Impulse Response (IR)**. An IR is an audio recording that captures how in this case a guitar cabinet reacts to instantaneous burst of energy.

When we convolve a dry guitar signal with this IR, we are essentially triggering that recorded response for every single sample of the input guitar.

This filters the frequency content—removing the harsh, fizzy high frequencies typical of direct-recorded distortion—and introduces the natural resonance and short reverberations of the wooden cabinet structure.

8.2 Source Material

For this specific implementation, the impulse response captures a classic rock configuration:

- **Cabinet:** Marshall 1960
- **Speakers:** Celestion G12M-25 "Greenbacks"
- **Microphone:** Shure SM57
- **Position:** Cap Edge, 0.5 inches distance

8.3 Results

Here is the resulted spectogram after applying the filter on crawling (spectogram for initial sample: Figure 1)

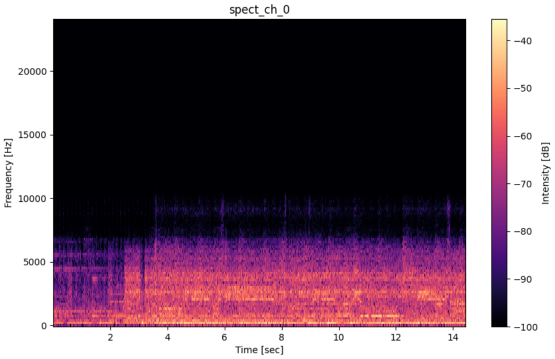

Figure 23: Cabinet applied on crawling

9 Overdrive Effect (Diode Clipper)

9.1 Concept

Overdrive creates distortion by "clipping" the peaks of an audio signal. In the analog world, this is often achieved using diodes. A diode allows current to flow easily in one direction but restricts it in the other, and most importantly, it has a non-linear relationship between voltage and current.

When the signal volume is low, the diode acts like an open circuit, and the audio passes unchanged. As the volume increases, the diode begins to conduct, "squashing" the amplitude. This squares off the tops of the waveforms, resuling in signals that we perceive as "distortion."

9.2 Implementation: Wave Digital Filters (WDF)

Instead of processing simple voltages, this implementation uses **Wave Digital Filters**. WDFs model circuits using wave variables (incident and reflected waves) rather than Kirchhoff variables (voltage and current).

The relationship between physical variables (v, i) and wave variables (a, b) at a port with port resistance R is defined as:

$$\begin{aligned} a &= v + R \cdot i \quad \text{(Incident Wave)} \\ b &= v - R \cdot i \quad \text{(Reflected Wave)} \end{aligned}$$

This approach decouples the non-linear element (the diode) from the rest of the circuit, allowing us to solve the non-linear equation locally without affecting the stability of the entire system.

9.3 The Physics: Shockley Diode Equation

The behavior of the diode is governed by the **Shockley Diode Equation**:

$$I_d = I_s(e^{V_d/V_t} - 1)$$

Where:

- I_d: Diode current.
- V_d: Voltage across the diode.
- I_s: **Saturation Current**. The code uses 1×10^{-9} (1nA) for positive voltage and 5×10^{-8} (50nA) for negative. This asymmetry creates **Asymmetric Clipping**, which produces even-order harmonics, often associated with smoother, "tube-like" distortion.
- V_t: **Thermal Voltage**. Approximately 25.85mV at room temperature (300K).

9.4 The Solver: Newton-Raphson Method

Because the Shockley equation is implicit (current depends on voltage, but voltage depends on current in the circuit), we cannot solve it algebraically. We must find a value for v that satisfies the WDF constraints.

We rearrange the WDF incident wave equation $a = v + R \cdot i(v)$ into a function $f(v)$ that equals zero:

$$f(v) = v + R \cdot I_s(e^{v/V_t} - 1) - a_{in} = 0$$

We solve for v using the **Newton-Raphson method**, an iterative algorithm that approximates the root of a function:

$$v_{n+1} = v_n - \frac{f(v_n)}{f'(v_n)}$$

Once v is found, the reflected wave b is calculated via $b = 2v - a$, which is returned to the circuit loop.

9.5 Results

Here is the resulted spectogram after applying the filter on the calm guitar sample (spectogram for initial sample: Figure 3)

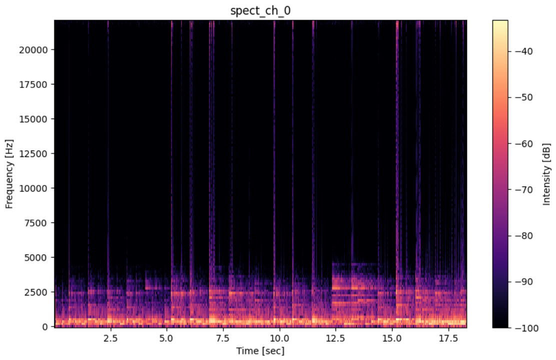

Figure 24: Overdrive applied on calm guitar

10 Pitch Shifter Effect (Granular Synthesis)

10.1 Concept

This implementation utilizes **Granular Synthesis** logic to achieve pitch shifting without altering the playback duration. The core principle mimics the **Doppler Effect**.

- Playing audio back faster than it was recorded increases the pitch.
- Playing it back slower decreases the pitch.

However, simply changing the playback speed would cause the audio to drift out of sync with the input stream (either running out of data or falling infinitely behind). To solve this, the audio is sliced into small segments called **"grains"** (typically 50–100ms). The read pointer moves at the desired pitch speed within a grain but periodically "jumps" back to align with the current write position.

10.2 Implementation: Dual Delay Line

A naive granular implementation creates audible discontinuities ("clicks") whenever the read pointer jumps. To eliminate these artifacts, this implementation uses a **Dual Delay Line** approach with cross-fading.

We employ a circular buffer to store a history of the input signal. Two separate read heads traverse this buffer:

- **Head A**: The primary read pointer.
- **Head B**: A secondary pointer offset by exactly 180° (0.5 phase).

As one head reaches the end of its grain (the discontinuity point), its volume is faded to silence, while the other head is at the peak of its volume in the middle of a grain. This ensures a continuous output signal.

10.3 Mathematical Formulation

10.3.1 Phase Accumulator

The position of the read heads is driven by a normalized phasor $\phi(n)$ that ranges from 0.0 to 1.0. The rate of change ($\Delta\phi$) determines the pitch shift:

$$\Delta\phi = \frac{1.0 - P}{N_{grain}}$$

Where:

- P: The pitch factor (e.g., 2.0 for an octave up, 0.5 for an octave down).
- N_{grain}: The size of a grain in samples.

The phasor for the current sample n is updated as:

$$\phi[n] = (\phi[n-1] + \Delta\phi) \pmod 1$$

10.3.2 Fractional Delay (Linear Interpolation)

Since the read pointer often lands between integer sample indices (e.g., index 405.7), we must approximate the value using **Linear Interpolation** to avoid quantization noise (aliasing):

$$y_{interp} = x[i] + \alpha \cdot (x[i+1] - x[i])$$

Where:

- i: The integer part of the read index ($\lfloor index \rfloor$).
- α: The fractional part ($index - i$).

10.3.3 Windowing Function

To smooth the transition between grains, we apply a **Triangle Window** to both read heads. The weight w for a given phase ϕ is calculated as:

$$w(\phi) = 1.0 - 2.0 \cdot |\phi - 0.5|$$

This function creates a triangular amplitude envelope:

- At $\phi = 0.0$ or 1.0 (grain edges), $w = 0$ (Silence).
- At $\phi = 0.5$ (grain center), $w = 1$ (Full Volume).

10.3.4 The Final Summation

The final output $y[n]$ is the sum of the two overlapping heads. Because Head B is offset by phase 0.5, the sum of their weights is constant ($w_A + w_B = 1$), resulting in unity gain:

$$\begin{aligned} \phi_A &= \phi[n] \\ \phi_B &= (\phi[n] + 0.5) \pmod 1 \\ y[n] &= \underbrace{(y_{\phi_A} \cdot w(\phi_A))}_{\text{Head A}} + \underbrace{(y_{\phi_B} \cdot w(\phi_B))}_{\text{Head B}} \end{aligned}$$

10.4 Results

Here is the resulted spectogram after applying the filter on the crawling sample (spectogram for initial sample: Figure 1)

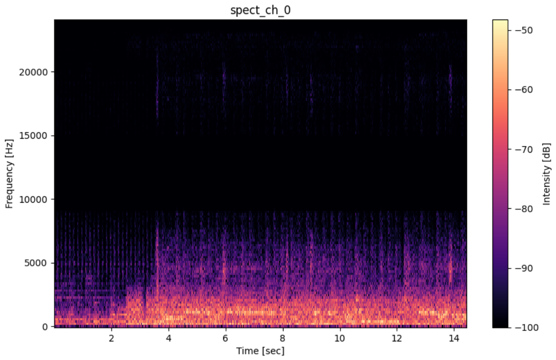

Figure 25: Pitch-shifter low pitch applied on crawling

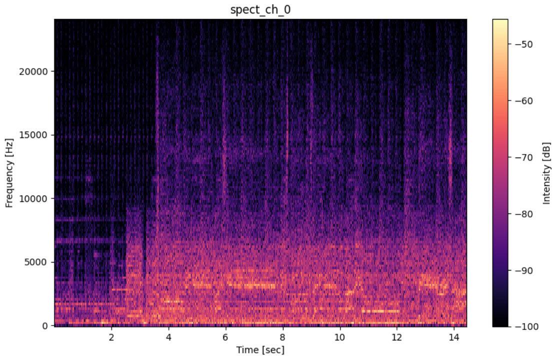

Figure 26: Pitch-shifter high pitch applied on crawling

11 References

- https://people.montefiore.uliege.be/josmalskyj/files/Gardner1995Efficien
  pdf
- https://redwirez.com/pages/the-marshall-1960a-ir-pack
- https://www.native-instruments.com/fileadmin/ni_media/downloads/pdf/VAFilterDesign_2.1.0.pdf
- Spatial Audio by Francis Rumsey
- https://ccrma.stanford.edu/~dtyeh/papers/yeh10_taslp.pdf
- https://www.musicdsp.org/en/latest/Analysis/97-envelope-detector.html
- https://www.uncini.com/dida/tsa/mod_tsa/Chap_08.pdf
- https://www.melaudia.net/zdoc/sweepSine.PDF
- https://re.public.polimi.it/bitstream/11311/1203185/1/WD_NR_TASLP_Bernardini_etal_2021_RG.pdf
- DAFX: Digital Audio Effects, 2nd Edition Udo Zölzer