KEYFRAME TIME STRETCHING VIA EXTREMA SAMPLING

Matthew Nielsen

Independent Researcher
Portland, Oregon, USA
heavylight.industries@gmail.com

ABSTRACT

Overlap-add (OLA) is the simplest approach to audio time stretching. Methods like the phase vocoder (PV) and waveform-similarity OLA (WSOLA) offer higher quality results but require operations like the FFT or cross-correlation. On low power embedded hardware, this cost adds up quickly. We present a content-adaptive OLA method, an order of magnitude cheaper than PV or WSOLA, whose dominant artifacts are added saturation and some spectral contrast loss. Our method reduces uniformly sampled signals to sets of timestamped local extrema, a sparse representation where the distance between points encodes the signal's information density directly into the buffer. In OLA, the crossfade duration is fixed, but no one value suits both transients and sustained sounds. We use the extrema density to inform the crossfade duration, adapting it to the signal's local content on a sample-by-sample basis. We compare our method against OLA, WSOLA, and PV using objective metrics and a listening test. Our method coherently stretches audio, preserving transients across a wide range of stretch ratios and capturing dense, layered material cleanly.

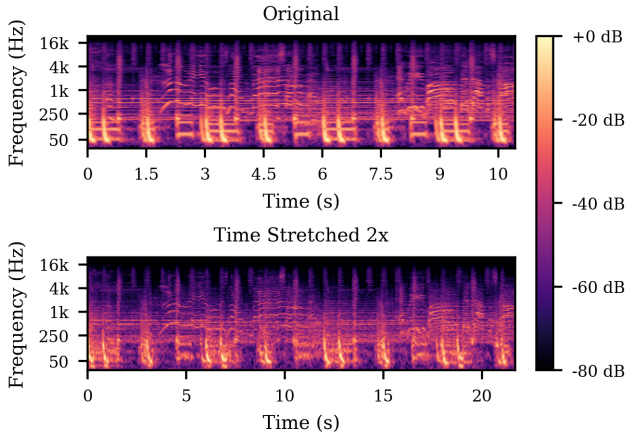

Figure 1: Spectrogram comparison of an excerpt of Lauryn Hill's song *Ex-Factor* stretched by 2×.

1. INTRODUCTION

Time stretching is a fundamental element of modern music production, and in most digital audio workstations it's available in high quality on as many stereo tracks as the user's PC can handle. On a low-power embedded musical platform there's less computational budget available. As such, high quality time stretching quickly eats into CPU headroom, especially as track count increases.

The algorithms that are most computationally feasible on a microcontroller each carry their own compromises. Overlap-add (OLA) [1] is the cheapest. It simply windows and re-spaces short segments of audio. With no mechanism to align them, periodic content degrades into either amplitude modulation and comb filtering for short segments, or audible repeats and altered timing for long ones. WSOLA [2] restores periodicity with a cross-correlation search but struggles with polyphonic content. The phase vocoder [1] handles harmonically rich material well but smears transients and imparts a reverberant, phasey quality to sounds. In PV or WSOLA, the size of the analysis window sets the cost of processing and the frequency range tradeoff.

Our method's compromise is of a different kind. Because we treat time stretching as a creative effect rather than transparent restoration, we happily trade some softening of spectral detail for a large reduction in cost and a coherent, organic sounding time stretch. Figure 1 previews this: harmonic and transient structure survives a 2× stretch largely intact.

Our method begins with the observation that uniformly sampled signals use the same number of samples per second on a bass note as they do for a cymbal crash, despite the two signals having vastly different spectra. Reducing a waveform to only its local extrema and the times at which they occur yields a sparse format whose local sample rate adapts to the signal's instantaneous bandwidth [3]. A bass note produces few, widely spaced extrema. A cymbal crash produces many tightly packed ones. The spacing between points is therefore a free, per-sample estimate of local information density available with no further analysis. Extrema are not an arbitrary choice of sample set. Theoretical work on reconstruction from zero crossings [4], from the joint timing of zeros and extrema [5, 6], and on topology-preserving signal deformations [7] points to extrema as locations where the signal is information-rich.

Prior work on extrema and level-crossing sampling [8, 9] has used sparse representations of this kind chiefly for data compression and event-driven acquisition. Our contribution is to treat the spacing between points itself as a control signal, using it to drive an overlap-add time stretching engine whose crossfade duration adapts to the local information density of the signal.

2. ALGORITHM DESCRIPTION

Our method has three stages. Analysis reduces a uniformly sampled signal to a set of non-uniform extrema we call *keyframes*, named after the discretely stored, continuously interpolated poses of computer animation [10]. Reconstruction interpolates a continuous signal back from those keyframes. Time stretching performs that

reconstruction with an adaptive overlap-add splice driven by the keyframe spacing. Algorithms 1 and 2 give analysis and reconstruction; the rest of this section develops each step in turn.

Algorithm 1 Extrema Analysis

Require: Uniform input buffer $x[n]$, threshold ϵ
Ensure: Sparse extrema buffer $x[m]$
$d_{\text{prev}} \leftarrow 0,\ \ v_{\text{prev}} \leftarrow x[0]$
Save extremum $(0,\ x[0])$ ▷ anchor
for each sample n **do**
  $d[n] \leftarrow \mathcal{B}'(x,\ n)$ ▷ derivative
  if $\text{sign}(d[n]) \neq \text{sign}(d_{\text{prev}})$ **then**
    if $|x[n] - v_{\text{prev}}| > \epsilon$ **then** ▷ above threshold
      $\alpha \leftarrow \frac{|d[n-1]|}{|d[n-1]|+|d[n]|}$
      $n_m \leftarrow (n-1) + \alpha$ ▷ subsample index
      $v_m \leftarrow \mathcal{B}(x,\ n_m)$ ▷ subsample value
      Save extremum $(n_m,\ v_m)$
      $v_{\text{prev}} \leftarrow v_m$
    end if
  end if
  $d_{\text{prev}} \leftarrow d[n]$
end for

Algorithm 2 Sparse Reconstruction

Require: Sparse buffer $x[m]$ of $(n_m,\ v_m)$ pairs, output length N
Ensure: Uniform output buffer $y[n]$
$m \leftarrow 0$
for each output sample $n = 0 \ldots N-1$ **do**
  while $n_{m+1} < n$ **do** ▷ advance playhead
    $m \leftarrow m + 1$
  end while
  $t \leftarrow (n - n_m) \ / \ (n_{m+1} - n_m)$
  $y[n] \leftarrow v_m\, h_{00}(t) + v_{m+1}\, h_{10}(t)$ ▷ $T_0 = T_1 = 0$
end for

2.1. Bandlimited Derivative Estimation

Detecting an extremum means finding where the signal's derivative changes sign, so the analysis requires a derivative of the input. The naive choice, a finite difference $x[n] - x[n-1]$, has its peak gain at Nyquist and no high-frequency attenuation, so it passes the noise and quantization energy concentrated there and reports spurious extrema. Empirical mode decomposition shares this sensitivity, identifying extrema by direct comparison of neighboring samples [11]. We instead estimate the derivative with a B-spline kernel whose zero at the Nyquist frequency suppresses these spurious extrema.

Windowed sinc interpolation is a kernel convolution method where a window function is applied to the sinc function, confining its infinite support to a finite interval. This kernel is then convolved with the signal. Cubic Hermite splines are an approximation of a windowed sinc kernel. Cubic basis splines, or B-splines, are an approximation of the window itself. When convolved with the signal they offer C^2 continuity and a strong zero at the Nyquist frequency at the cost of some high-frequency rolloff.

In practice, we implement B-splines as 4-tap FIR filters where the coefficients are defined by the data itself, requiring no evaluation of trig functions or lookup table interpolation at runtime. The interpolated value in the interval $t \in [0, 1]$ between control points p_{-1}, p_0, p_1, and p_2 is defined as

$$p(t) = p_{-1}b_{-1}(t) + p_0 b_0(t) + p_1 b_1(t) + p_2 b_2(t) \tag{1}$$

where the cubic equation for each segment is:

$$b_{-1}(t) = \frac{(1-t)^3}{6}, \qquad b_0(t) = \frac{3t^3 - 6t^2 + 4}{6},$$
$$b_1(t) = \frac{-3t^3 + 3t^2 + 3t + 1}{6}, \quad b_2(t) = \frac{t^3}{6}. \tag{2}$$

Just as a bandlimited derivative can be obtained by windowing the derivative of the sinc function, we take the derivative of each B-spline segment to obtain a bandlimited derivative kernel:

$$b'_{-1}(t) = \frac{-3(1-t)^2}{6}, \qquad b'_0(t) = \frac{9t^2 - 12t}{6},$$
$$b'_1(t) = \frac{-9t^2 + 6t + 3}{6}, \qquad b'_2(t) = \frac{3t^2}{6}. \tag{3}$$

Figure 2 shows the resulting impulse and frequency responses for both the kernel and its first derivative.

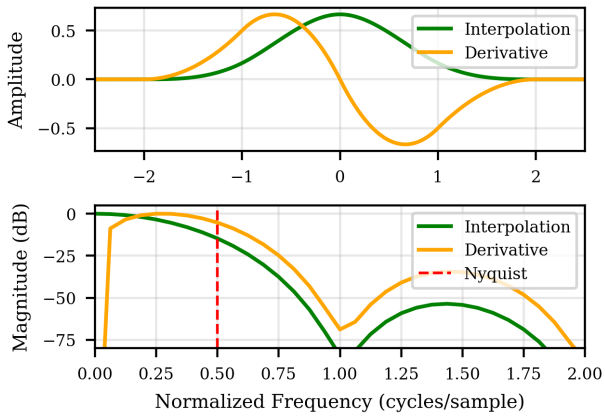

Figure 2: The impulse and frequency responses of the B-spline kernel and its first derivative.

2.2. Difference Thresholding

The sign-change test of the previous section fires at every extremum, including the countless low-level ripples of the noise floor and fine texture. Keeping them all works against the purpose of the sparse representation, so we keep an extremum only when it changes the signal meaningfully. We save the first sample as an anchor, then for each subsequent candidate we take the absolute difference between its amplitude and that of the last saved extremum. If the difference does not exceed a threshold ϵ, the candidate is discarded.

This thresholding is state-dependent. Each saved extremum resets the reference that the next candidate is judged against, giving the decision a hysteresis-like deadband. In the case of digital audio, the signal's amplitude exists in the range -1.0 to 1.0. We found that an amplitude difference threshold of 0.001 (-60 dB) provides a faithful reconstruction of the input. At threshold values below this, the extrema of the noise floor are the only components added and the apparent quality remains the same.

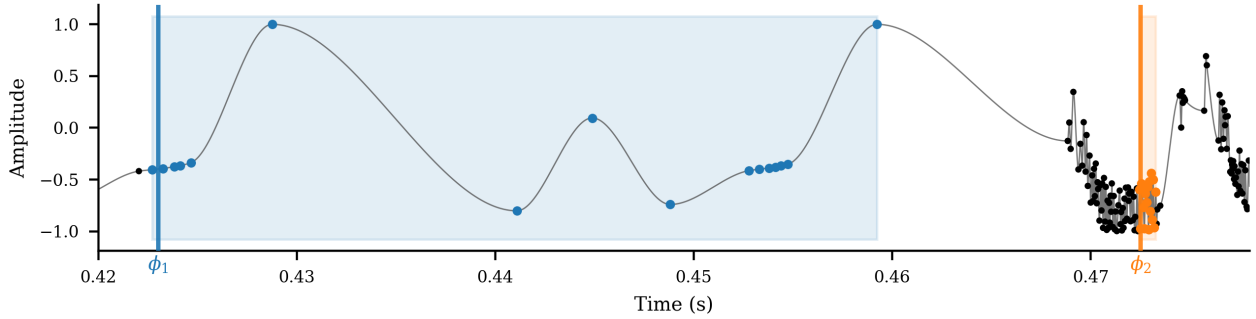

Figure 3: Demonstrating the adaptive splice duration on a bass note followed by a snare hit. The span across the 16 keyframes ahead of playhead ϕ_1 is much longer than the span across the 16 keyframes ahead of playhead ϕ_2.

A higher threshold means more low amplitude elements of the signal are discarded. High frequencies are preserved as long as their amplitude is large enough. However, in practice higher frequencies tend to have lower amplitudes. Therefore, raising the threshold has an amplitude-dependent low-pass filter effect.

2.3. Subsample Extrema Location

Once we've determined that an extremum is worth keeping, we can't simply treat the current sample itself as the extremum. Doing so provides a decent approximation for low frequencies but causes aliasing as frequency increases.

We utilize reverse linear interpolation to solve for the subsample location where the derivative equals zero. The reduction in aliasing between grid snapping and reverse linear interpolation is significant, but we found diminishing returns when using a secondary linear refinement or a quadratic root finding approach. The algorithm for reverse linear interpolation of the derivative signal $d[n]$ is as follows:

$$\alpha = \frac{|d[n-1]|}{|d[n-1]| + |d[n]|} \tag{4}$$

The resulting subsample index n_m is then defined as:

$$n_m = (n-1) + \alpha \tag{5}$$

The refined value v_m is obtained by evaluating the B-spline function $\mathcal{B}$ at the refined index n_m:

$$v_m = \mathcal{B}(n_m) \tag{6}$$

2.4. Sparse Buffer Playback

Before we reconstruct the original signal from its sparse representation we must establish a mental model of their relationship. $x[n]$ is a uniform buffer of length N, and $x[m]$ is a sparse buffer of length M, where $N > M$ but both buffers represent the same signal over the same amount of time, and $x[m]$ is a subset of $x[n]$. Each extremum is defined by the pair (n_m, v_m), where n_m is the extremum's subsample index and v_m is its value.

We use an analog playhead analogy to conceptualize playback, where playing the sparse buffer like tape means incrementing a playhead through its indices. For a particular continuous index, there is a corresponding pair of keyframes needed to produce an output. We call this pair a window. Each w contains:

- The window's keyframe index, denoted $w.m$
- Start and end indices n_m and n_{m+1}
- Duration $\Delta = n_{m+1} - n_m$
- Phase increment $\dot{\phi} = 1/\Delta$

Each playhead consists of:

- A continuous uniform index ϕ
- The corresponding window w
- Sparse subsample position $t = (\phi - n_m) \cdot \dot{\phi}$
- Playback speed σ

Because the sparse index is no longer correlated with the uniform time index, finding the window for a given ϕ requires a search for the pair where $n_m \leq \phi < n_{m+1}$, rather than the direct indexing a uniform buffer allows. During normal playback the playhead moves sequentially, so a linear scan from the last valid m stays local and resolves in a few iterations, an amortized $O(1)$ per output sample. For true random access, a binary search over the buffer locates the window in $O(\log M)$, an advantage that grows with buffer size.

Consider the example of pitched up playback where $\sigma = 1.25$. Each tick, sparse subsample position t is computed from ϕ and the current window and used to interpolate between the two keyframes that make up the window at that continuous index. When $t > 1$, a local search finds the next valid window.

With this analogy, we only need to consider the playhead moving along the virtual tape at some speed. All of the sparse-to-uniform mapping is handled behind the scenes, which allows us to more easily expand on this idea.

2.5. Non-Uniform Cubic Interpolation

Now that we've established the connection between the uniform and sparse domains, it's time to describe how we actually generate interpolated values for a particular ϕ. To do so, we again turn to cubic splines, but this time with a non-uniform spacing between samples. For a given window and t, the interpolated value is defined as:

$$p(t) = v_m h_{00}(t) + T_0 h_{01}(t) + v_{m+1} h_{10}(t) + T_1 h_{11}(t) \tag{7}$$

where $t \in [0, 1]$ is the normalized time between the two keyframes, and the basis functions are

$$\begin{aligned} h_{00}(t) &= 2t^3 - 3t^2 + 1, & h_{01}(t) &= t^3 - 2t^2 + t, \\ h_{10}(t) &= -2t^3 + 3t^2, & h_{11}(t) &= t^3 - t^2. \end{aligned} \tag{8}$$

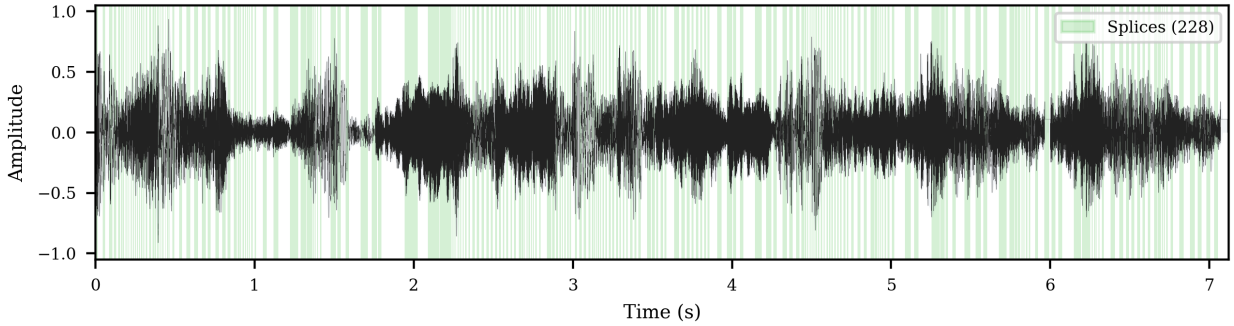

Figure 4: Demonstrating the adaptive splice duration on a musical sample with a pitch rate of 1.65, a time rate of 0.84, a splice duration spanning 113 keyframes, and a total of 228 splices.

In a non-uniform cubic Hermite spline, the derivatives, commonly called the tangents in this context, are estimated using finite differences across varying time intervals [12]:

$$T_0 = \frac{1}{2}\left(\frac{v_{m+1} - v_m}{n_{m+1} - n_m} + \frac{v_m - v_{m-1}}{n_m - n_{m-1}}\right) \tag{9}$$

$$T_1 = \frac{1}{2}\left(\frac{v_{m+2} - v_{m+1}}{n_{m+2} - n_{m+1}} + \frac{v_{m+1} - v_m}{n_{m+1} - n_m}\right) \tag{10}$$

The derivative calculations are computationally expensive, requiring two additional keyframes besides m and $m+1$ and four floating point divisions. However, our analysis already ensured that every extremum we saved has a derivative of zero.

By setting T_0 and T_1 to zero, the two tangent terms in $p(t)$ vanish, leaving us with the following equation:

$$p(t) = v_m h_{00}(t) + v_{m+1} h_{10}(t) \tag{11}$$

The result depends only on the extremum's value scaled by the cubic equations. The spacing between points is no longer part of the equation. This means that C^1 continuous non-uniform interpolation is possible using only two extrema, no division, and less than half the multiplications. This is equivalent to the smoothstep function commonly used in computer graphics [13, p. 30]. With this continuous-time representation of the sparse signal we can generate a uniformly sampled output at any desired sample rate, equivalent to indexing a traditional ring buffer.

2.6. Time Stretching

Time stretching is an operation that makes pitch and duration independent of one another. We extend overlap-add [1] so that, rather than splicing at a fixed duration, the keyframe spacing decides both *when* a splice happens and *how long* it lasts. The engine tracks three playheads:

- A **reference playhead** ϕ_{ref}, advancing at the time rate τ, marking where playback *should* be.
- A **playhead** ϕ_{play}, producing audio and advancing at the pitch rate σ.
- A **temporary playhead** ϕ_{temp}, producing audio at pitch rate σ only during a splice.

A useful analogy to describe our approach to time stretching is likening playback to a jogger (the reference playhead) and their leashed dog (the playhead). The reference playhead moves steadily along the signal. The playhead is free to run ahead or lag behind, but only to the end of a leash K keyframes long. Because the leash is measured in keyframes rather than samples, its length in time stretches and contracts with the signal. Across a sparse passage the playhead can roam far before the leash goes taut. Approaching a transient, the keyframes crowd together, the leash shortens, and the playhead is reined in sooner. Reaching the end of the leash triggers a splice. The current playhead fades out as a fresh one fades in at the reference position. Algorithm 3 gives the full procedure.

When $\tau \neq \sigma$, error accumulates between the reference and playhead positions. We measure this distance d as the absolute difference between their keyframe indices $w.m$; when it exceeds the threshold K, a splice begins. Its duration L is the span between the reference's current keyframe and the one K keyframes ahead:

$$L = n_{w_{\text{ref}}.m+K} - n_{w_{\text{ref}}.m} \tag{12}$$

Since the splice phase t_{temp} increments by $\dot{t}_{\text{temp}} \cdot \sigma$ per tick, the crossfade completes in L/σ output samples, after which the playhead adopts the temporary playhead's position and window and normal playback resumes.

Figure 3 shows this adaptation on a bass note followed by a snare hit. At 0.47 seconds, where the signal transitions from sparse to dense, the keyframe-index distance between the playheads exceeds K and a splice of length L is initiated. Figure 4 shows the same mechanism on a longer musical sample at non-unity pitch and time rates.

2.7. Block Processing

Unlike WSOLA or PV, the quality of our method does not depend on block size. The output is generated on a sample-by-sample basis. However, for real time processing it's still expedient to process in blocks because during analysis it's not known when the next extremum might occur. The solution is to save a keyframe at block boundaries, essentially enforcing a baseline uniform sample rate upon the signal. This enables the analysis and the reconstruction to take place one after the other, with a delay of 512 samples for example. Note that this isn't needed for the offline "record then play back" approach.

Algorithm 3 Sparse-Domain Time Stretching

Require: Sparse buffer $x[m]$, time rate τ, pitch rate σ, threshold K
Ensure: Output sample y per tick

$\phi_{\text{ref}}, \phi_{\text{play}} \leftarrow 0$, splicing $\leftarrow$ false
for each output tick **do**
  Update w_{ref}, w_{play} if $t > 1$ ▷ window search
  $d \leftarrow |w_{\text{play}}.m - w_{\text{ref}}.m|$ ▷ measure distance
  if ¬ splicing **and** $d > K$ **then** ▷ initialize splice
    $\phi_{\text{temp}} \leftarrow \phi_{\text{ref}}$, $w_{\text{temp}} \leftarrow w_{\text{ref}}$
    $L \leftarrow n_{w_{\text{ref}}.m+K} - n_{w_{\text{ref}}.m}$
    $t_{\text{temp}} \leftarrow 0$, $\dot{t}_{\text{temp}} \leftarrow 1/L$
    splicing $\leftarrow$ true
  end if
  if splicing **and** $t_{\text{temp}} \leq 1$ **then** ▷ conduct splice
    $y_{\text{play}} \leftarrow \text{Interp}(x[m], \phi_{\text{play}})$
    $y_{\text{temp}} \leftarrow \text{Interp}(x[m], \phi_{\text{temp}})$
    $y \leftarrow y_{\text{temp}} \cdot t_{\text{temp}} + y_{\text{play}} \cdot (1 - t_{\text{temp}})$
    $\phi_{\text{play}} \mathrel{+}= \sigma$, $\phi_{\text{temp}} \mathrel{+}= \sigma$
    $t_{\text{temp}} \mathrel{+}= \dot{t}_{\text{temp}} \cdot \sigma$
    $\phi_{\text{ref}} \mathrel{+}= \tau$
    return y
  else if splicing **and** $t_{\text{temp}} > 1$ **then** ▷ finish splice
    $\phi_{\text{play}} \leftarrow \phi_{\text{temp}}$, $w_{\text{play}} \leftarrow w_{\text{temp}}$
    splicing $\leftarrow$ false
  end if
  $y \leftarrow \text{Interp}(x[m], \phi_{\text{play}})$ ▷ normal playback
  $\phi_{\text{play}} \mathrel{+}= \sigma$, $\phi_{\text{ref}} \mathrel{+}= \tau$
  return y
end for

2.8. Computational Complexity

We compare the per output sample operation counts and an estimated Cortex M7 cycle cost against overlap-add (OLA), waveform-similarity OLA (WSOLA) [2], and the phase vocoder (PV), using the MATLAB implementations and default parameters of the TSM toolbox [1]: window $N_w = 1024$ and synthesis hop $H_s = 512$ for OLA and WSOLA, correlation tolerance $\Delta_{\max} = 512$ for WSOLA, and FFT length $N_f = 2048$ for PV and the FFT-assisted WSOLA variant.

Our cost splits into a one-time analysis pass and per-sample playback, and Table 1 reports both. Offline (Ours_{O}), a signal is sparsified once and replayed many times, so only playback is charged. During live processing (Ours_{L}), analysis and playback run together and their costs sum. The analysis pass, a 4-tap derivative FIR with thresholding and subsample refinement, runs per input sample and scales with the extrema density M/N, which peaks at 1 for white noise at the Nyquist rate; we adopt $M/N \approx 1/2$ as a conservative bound for musical material. The playback figures are the worst case of an active splice: two interpolations and a crossfade.

Table 1 summarizes the results. The time-domain WSOLA search correlates an N_w-tap window across $2\Delta_{\max} + 1$ offsets, on the order of $N_w \cdot 2\Delta_{\max} \approx 10^6$ multiply–adds per frame; with $\Delta_{\max} = 512$ this amortizes to roughly 2050 per output sample over the $H_s = 512$ hop, three orders of magnitude above OLA. The same correlation can instead be computed with three length-$2N_w$ FFTs [1]. Counting each length-N FFT at $\frac{N}{2}\log_2 N$ complex multiplications and $N \log_2 N$ complex additions, this FFT-assisted variant falls into the same cost class as the phase vocoder. The phase vocoder additionally computes a per-bin $\arctan$ for the phase; without hardware support this transcendental costs around a hundred cycles. Both frequency-domain methods sit well below time-domain WSOLA, yet remain an order of magnitude above our method.

Table 1: Per-output-sample operation counts and an estimated Cortex-M7 cycle cost, with per-frame costs amortized over the synthesis hop H_s. ADD includes additions, subtractions, absolute values, and comparisons; DIV includes divisions and square roots; TRIG counts $\sin$, $\cos$, and $\arctan$. WSOLA_{T} uses time-domain correlation, WSOLA_{F} FFT-based correlation; Ours_{O}/Ours_{L} are offline/live (playback only vs. analysis plus playback).

Method	MUL	ADD	DIV	TRIG	Cycles
OLA	2	2	0	0	≈ 4
WSOLA_{T}	2050	2052	0	0	≈ 4100
WSOLA_{F}	280	407	0	0	≈ 690
PV	190	275	2	6	≈ 850
Ours_{O}	15	19	0	0	≈ 34
Ours_{L}	21	26	1	0	≈ 61

Cycle estimate: multiplies, adds, and comparisons at one cycle each; divisions and square roots at 14; $\sin$ and $\cos$ at ≈40; and $\arctan$ at ≈100, a conservative figure for an optimized implementation. PV evaluates one $\arctan$, $\sin$, $\cos$, and square root per bin; the $\arctan$, its costliest transcendental, is roughly 200 of its ≈850 cycles.

These counts favor our method most clearly on the embedded targets we are concerned with. On desktop CPUs the FFT is heavily accelerated by wide vector units, and the frequency-domain methods close the performance gap. Our targets have no such hardware. Optimized software FFT and fast-math routines such as those in the CMSIS-DSP library help, but the cost stays $O(N \log N)$ per frame, and the per-bin transcendentals that dominate the phase vocoder have no hardware acceleration. Software lookup tables reduce $\sin$ and $\cos$ to interpolated reads, but $\arctan$ and its unbounded domain resists tabulation. More fundamentally, our method emits each output sample independently, with no frame to buffer and no block latency, whereas every frame-based method must accumulate and transform a full window before producing output. That structural property, as much as the operation count, is what suits the method to low-latency, sample-by-sample use on constrained hardware.

3. RESULTS AND DISCUSSION

3.1. Test Methodology

To quantify performance we run our method on the Electrosmith Daisy Patch SM Eurorack module, built around a 480 MHz STM32 H7 processor with a 24-bit 48 kHz codec and 64 MB of SDRAM.

Our evaluation has two parts. The first characterizes the artifacts intrinsic to our method, independent of any time stretching. Namely, the aliasing and harmonic distortion of the reconstruction and the fidelity lost to sparsification at unity time and pitch. The second compares our time stretching head-to-head against OLA, WSOLA, and the phase vocoder, both subjectively through a listening test and objectively through transient preservation.

Our corpus of 10 samples comes from the TSM toolbox website [1], a set of short clips curated to stress-test TSM algorithms. The authors processed each at three stretch factors through OLA, WSOLA, and the phase vocoder, which we adopt as our comparison conditions.

3.2. Aliasing and Harmonic Distortion

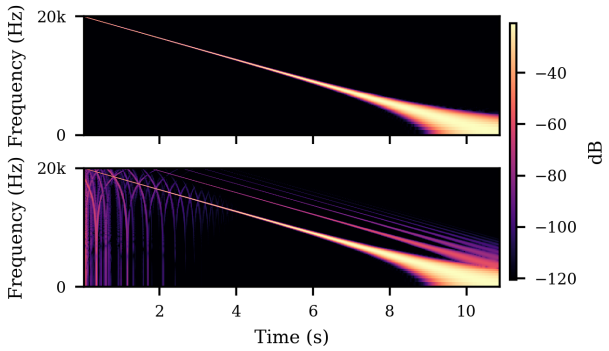

Figure 5: Sine sweep before and after sparsification. Frequency axis is log-scaled.

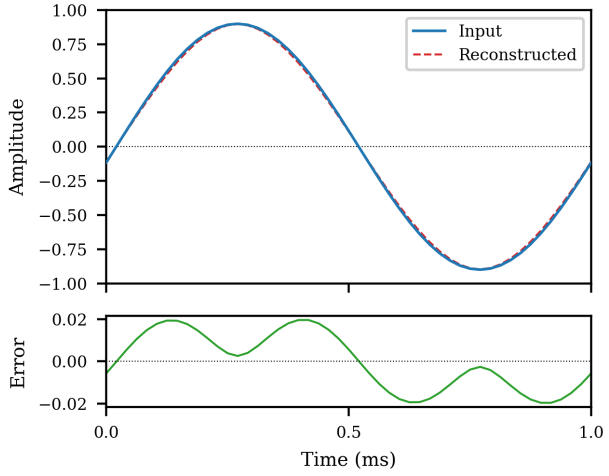

Figure 6: Time domain difference for a single cycle of a sine wave.

To measure aliasing of the sparse sampling system as a whole, we processed a sine wave frequency sweep from 20 kHz to 20 Hz locally on the Patch SM. The reconstruction's spectrogram in Figure 5 shows the harmonics caused by the cubic spline's inability to reproduce a perfect sine wave.

To quantify the spline's total harmonic distortion, we processed a 1 kHz sine wave and used the Goertzel algorithm [14] to measure the presence of even and odd harmonics. The total THD is -38.1 dB for odd and -85.8 dB for even. For reference, a tanh saturator at unity gain has a THD of -25 dB for odd harmonics and none for even. Figure 6 shows the time-domain difference between the sparsified and original single cycle.

3.3. Sparsification Artifacts

Extrema sampling, which is a form of lossy data compression, results in an unavoidable quality loss that's independent of the time stretching and highly dependent upon the source material and reconstruction interpolation. To quantify this, we utilize several metrics.

- Short time objective intelligibility (STOI) [15]
- Log-attack-time (LAT) deviation [16]
- Spectral contrast loss [17]

STOI produces a score from 0 to 1, where a score of 1 is considered completely transparent. LAT deviation measures differences in the timing of transients where 0 is a perfect score. Spectral contrast loss measures the amount of blurring in the contours of the frequency domain, with 0 dB being a perfect score. We ran these tests on each of the 10 test samples with no time or pitch modification applied and an analysis threshold of 0.001 or -60 dB. Our goal is for the results to be as transparent as possible.

Table 2: Passthrough fidelity per clip at unity time/pitch.

Clip	STOI	LAT dev.	Contrast loss (dB)
Bongo	0.98	0.07	2.55
Castanets/Violin	0.85	0.01	5.13
Drum Solo	0.97	0.02	2.41
Glockenspiel	0.75	0.41	3.84
Jazz	0.99	0.05	2.70
Pop	0.99	0.02	2.58
Singing Voice	0.92	0.15	4.54
Indie Pop	0.99	0.08	2.79
Synth Mono	0.85	0.05	2.67
Synth Poly	0.97	0.02	2.44
Mean	0.91	0.09	3.08

The results indicate that in most cases, the overall intelligibility of the sample is preserved along with the transients. However, it's evident that certain material (like the Glockenspiel) isn't captured effectively by the sparse sampling. In the case of the singing voice, the STOI and LAT deviation are acceptable but the level of contrast loss is significant. This contrast loss fills the valleys between spectral peaks, heard as a wideband, noise-like haze around the signal's partials. Across the board, some blurring of the spectrum is present in all samples.

3.4. Listening Test

We conducted a time stretching comparison survey ($N = 15$ participants) using 5 of the 10 test samples, with the aim of capturing a variety of percussive and harmonic material at different stretch factors without fatiguing the listener. We utilized the webMUSHRA framework to create the test, although there is no baseline hidden stimulus within the conditions as is normally the case in a MUSHRA [18] test because of the change in time scale.

We matched the test conditions that TSM toolbox used for their example dataset. Each of the samples is stretched by a factor of 0.5, 1.2, and 1.8 (a factor of 0.5 results in the sample playing twice as fast as the original). Table 3 shows the rating averaged across the three stretch factors.

The survey results show WSOLA as the overall winner, topping four of five clips. Phase vocoder scored well for harmonic samples and poorly for transient heavy ones. OLA scores exceptionally well for purely percussive material and very badly for anything harmonic. Our method was rated "fair" overall, dropping to "poor"

Table 3: MUSHRA ratings per clip, averaged across stretch factors 0.5, 1.2, and 1.8.

Clip	KEYFRAME	OLA	PV	WSOLA
Bongo	51.1	68.3	19.2	54.4
Castanets/Violin	42.5	17.2	43.7	73.3
Drum Solo	48.8	54.6	24.1	55.9
Singing Voice	31.1	17.7	58.9	76.0
Indie Pop	46.8	11.8	52.6	54.2

only on the singing voice. This ranking tracks sparsification fidelity, not stretch quality. For instance, the two clips with the most spectral contrast loss in the passthrough test (Table 2), the singing voice and castanets/violin, are exactly the two lowest-scoring here, while the cleanly sparsified clips come within several points of WSOLA. The listening test thus measures the cost of our representation as much as the time stretching itself.

3.5. Transient Preservation

To complement the subjective listening test with some objective measure of accuracy, we utilize LAT deviation to measure the alignment and preservation of transients. The results show our method has the best LAT for stretch factor of 1.8x, is a close second for 1.2x, and performs the worst of the four for the 0.5x stretch ratio. See §3.7 for further discussion.

Table 4: Log-Attack-Time (LAT) deviation per stretch factor, averaged across all 10 clips. Lower is better.

Stretch	KEYFRAME	OLA	PV	WSOLA
0.50	0.875	0.755	0.704	**0.635**
1.20	0.520	0.540	0.817	**0.489**
1.80	**0.502**	0.575	0.648	0.531

We acknowledge that a complementary objective measure of spectral fidelity would round out this picture, but we found that such metrics remain an open problem for TSM. Frame-aligned distances (MCD, log-spectral distance) and perceptual metrics (PEMO-Q, ViSQOL) all assume a temporal alignment that the TSM operation itself disturbs.

3.6. Summary

Measured against its goal, a creative time stretch on low-power hardware, our method delivers where it counts. It's an order of magnitude cheaper than the phase vocoder or FFT-assisted WSOLA (Table 1), runs sample-by-sample with no block latency, and preserves transients competitively with the time-domain methods, leading on log-attack-time at the 1.8x stretch (Table 4). Its main concession is quality on spectrally nuanced material. As the listening test shows, this tracks the representation rather than the stretching.

In practice our method is at its best on dense, layered material. Full mixes with drums, bass, vocals, and keys at once capture cleanly, which is a pattern the passthrough scores confirm. The busiest clips (jazz, pop, indie) lose the least contrast, while the most nuanced (glockenspiel, solo voice) lose the most. Percussion is a particular strength, where the varying splice length becomes a texture in its own right. Across all material, the adaptive splice duration imparts a naturalistic quality to the result.

3.7. Limitations

This approach is not without issues. We address what we consider the most important ones below:

1. Time compression (0.5x) underperforms compared to the 1.2x and 1.8x stretch ratios. We attribute this to the fact that we prioritized longer splice durations for 0.5x to maintain audible clarity and avoid robotic-sounding inharmonic distortion, thus lowering the timing scores.
2. Spectrally nuanced material shows audible spectral contrast loss from the sparsification, independent of the time scale modification. This negatively impacts clarity of the output.
3. Long periods of silence or otherwise sparse keyframes followed by transients result in a very long splice duration, causing many audible repeating segments leading up to the transient. We partially mitigate this by enforcing a maximum splice duration of a few hundred milliseconds.

3.8. Further Work

One method to improve the quality of the output we explored was saving the B-spline's second derivative with each keyframe during analysis. The polarity of the second derivative describes whether that extremum is max or a min, and its magnitude describes the signal's acceleration. Using it as an additional term in the cubic reconstruction interpolation may improve the spectral clarity and reduce the contrast loss.

Another direction to explore is using the time difference between extrema as the setpoint for the highest frequency band in a wavelet filter bank, eliminating unoccupied frequency space before the wavelet transform. Within each wavelet band, the sparsification can then be run and the resulting local extrema spacing used as a measure of instantaneous frequency, phase, and magnitude in the spirit of the Hilbert-Huang Transform [11].

Transient detection is another promising direction for this sparse medium. One might infer the location of transients by searching for significant changes in duration between keyframes. Another approach would be to use the second derivative of the extrema to calculate the Teager-Kaiser energy operator [19], and treat threshold crossings in that signal as transient events.

4. CONCLUSION

We set out to time-stretch audio on low-power hardware, where the phase vocoder and WSOLA are too costly to run at scale. Our method provides a per-sample estimate of the signal's local bandwidth. The analysis is paid once, and from then on adaptive time stretching uses that estimate to produce uniform-rate output. As a side benefit the representation provides mild data compression, an inherent advantage on the limited RAM of embedded devices.

Beyond being less expensive than FFT and cross-correlation methods, our engine puts every parameter under continuous real-time control. The time and pitch rates modulate freely, and the splice threshold K acts as a macro over splice duration. Raising it lengthens the splices while leaving their adaptation to transients intact.

Time stretching is only the first operation we've built with this representation. The same spacing that drives the splice has potential to locate transients and estimate instantaneous frequency and phase. We see the sparse, content-adaptive medium itself as our contribution, more than any single operation performed with it. An open-source implementation and audio examples are available on our project page.[1]

5. ACKNOWLEDGMENTS

We would like to thank Jatin Chowdhury, Mateo Aboy, and John Costella for their engagement with our work. The time and effort they and the anonymous reviewers freely offered helped turn this from an idea into reality. We would also like to thank our family and friends for supporting us throughout this journey.

6. REFERENCES

[1] J. Driedger and M. Müller, "TSM toolbox: MATLAB implementations of time-scale modification algorithms," in *Proc. 17th Int. Conf. Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sept. 1–5, 2014, pp. 249–256.

[2] W. Verhelst and M. Roelands, "An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech," *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, pp. 554–557, 1993.

[3] D. Rzepka and M. Miśkowicz, "Recovery of varying-bandwidth signal from samples of its extrema," in *Proc. Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, Poznań, Poland, Sept. 26–28, 2013, pp. 143–148.

[4] J. B. F. Logan, "Information in the zero crossings of bandpass signals," *Bell Syst. Tech. J.*, vol. 56, no. 4, pp. 487–510, 1977.

[5] R. Kumaresan and Y. Wang, "On representing signals using only timing information," *J. Acoust. Soc. Am.*, vol. 110, no. 5, pp. 2421–2439, 2001.

[6] A. Ignjatović, C. Wijenayake, and G. Keller, "Chromatic derivatives and approximations in practice—Part II: Nonuniform sampling, zero-crossings reconstruction, and denoising," *IEEE Trans. Signal Process.*, vol. 66, no. 6, pp. 1513–1525, Mar. 2018.

[7] G. Essl, "Topology-preserving deformations of digital audio," in *Proc. 27th Int. Conf. Digital Audio Effects (DAFx24)*, Guildford, UK, Sept. 3–7, 2024.

[8] B. Premanand and V. S. Sheeba, "Compressed encoding of vibration signals using extremum sampling," *SN Applied Sciences*, vol. 2, no. 7, p. 1261, Jun. 2020.

[9] N. Sayiner, H. V. Sorensen, and T. R. Viswanathan, "A level-crossing sampling scheme for A/D conversion," *IEEE Trans. Circuits Syst. II*, vol. 43, no. 4, pp. 335–339, Apr. 1996.

[10] N. Burtnyk and M. Wein, "Computer-generated key-frame animation," *J. Soc. Motion Picture Telev. Eng.*, vol. 80, no. 3, pp. 149–153, 1971.

[11] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N.-C. Yen, C. C. Tung, and H. H. Liu, "The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis," *Proc. Royal Soc. London A*, vol. 454, pp. 903–995, 1998.

[12] E. Catmull and R. Rom, "A class of local interpolating splines," in *Computer Aided Geometric Design*, R. E. Barnhill and R. F. Riesenfeld, Eds. Academic Press, 1974, pp. 317–326.

[13] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley, *Texturing and Modeling: A Procedural Approach*, 3rd ed. Morgan Kaufmann, 2003.

[14] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *The American Mathematical Monthly*, vol. 65, no. 1, pp. 34–35, 1958.

[15] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, 2011.

[16] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," IRCAM, Paris, France, Tech. Rep., 2004.

[17] D.-N. Jiang, L. Lu, H.-J. Zhang, J.-H. Tao, and L.-H. Cai, "Music type classification by spectral contrast feature," in *Proc. IEEE Int. Conf. Multimedia and Expo (ICME)*, vol. 1, Lausanne, Switzerland, 2002, pp. 113–116.

[18] International Telecommunication Union, "Method for the subjective assessment of intermediate quality level of audio systems," ITU-R, Geneva, Switzerland, Tech. Rep. Recommendation BS.1534-3, 2015.

[19] J. F. Kaiser, "On a simple algorithm to calculate the "energy" of a signal," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, Albuquerque, NM, USA, Apr. 1990, pp. 381–384.

7. APPENDIX: REDUCED COMPLEXITY VARIANTS

We propose two ways to reduce the computational complexity of our algorithm:

The first is to omit the subsample interpolation and simply treat the sample where the zero crossing was detected as the extremum. This removes the need for reverse linear interpolation and subsample value interpolation, and allows us to store the index as an integer rather than a floating-point value. The downside is increased aliasing in the output.

The second is to use linear interpolation for the analysis interpolator, a finite difference derivative instead of cubic, and linear interpolation for reconstruction. Increased aliasing is the main drawback to this optimization.

The time stretching behavior of the engine is unaffected by these optimizations, even when the fundamental operator of every step is just linear interpolation.

[1] `https://github.com/heavylight-industries/capicola`