A UNIFIED FRAMEWORK FOR REAL-TIME CONCATENATION-DRIVEN CONVOLUTION

Niccolo Abate

Center for Computer Research in Music and Acoustics
Stanford University
Stanford, California, USA
nlabate@stanford.edu

Brian Hansen

Department of Computational Media
University of California, Santa Cruz
Santa Cruz, California, USA
brmhanse@ucsc.edu

ABSTRACT

This work introduces a novel framework for Concatenation-Driven Convolution (CDC), unifying concatenative synthesis and real-time convolution into a single integrated audio processing paradigm. While concatenative synthesis has traditionally been used for corpus-based sound generation and convolution has served as a largely static filtering technique, the proposed approach reconceptualizes impulse responses (IRs) as dynamic, navigable sonic material. In the CDC framework, a corpus of audio segments is analyzed using perceptual features and organized via a self-organizing map (SOM), enabling intuitive, gesture-based traversal of a structured timbral space; the resulting concatenative output is treated as a continuously evolving impulse response and injected directly into a partitioned convolution engine. Its central technical contribution is single-engine frequency-domain kernel interpolation: rather than crossfading the outputs of two convolution engines, the FFT-domain kernels of the current and target IRs are interpolated within a single engine, preserving the internal convolution state across IR transitions and avoiding the warm-up energy loss inherent to dual-engine crossfading. Implemented as a real-time audio plugin, the system is evaluated in terms of computational efficiency and output-energy stability, demonstrating lower computational cost under continuous IR updates and consistent output energy across all interpolation conditions, with audio-thread CPU usage well within real-time constraints. The CDC framework enables continuous, expressive control over reverberant and textural transformations, opening new possibilities for corpus-driven sound design and convolution-based audio effects across a wide range of audio material — from measured acoustic spaces to arbitrary recordings and synthesized textures.

1. INTRODUCTION

Concatenative synthesis is a sound synthesis technique in which short segments of audio drawn from a corpus are selected and recombined according to perceptual features or target descriptors. Conceptually related to granular synthesis, it differs in that the units are typically drawn from a structured database of audio events rather than from a single source recording. Over the past two decades, concatenative synthesis has been used primarily for textural sound design and environmental sound generation, including applications such as environmental soundscapes, Foley design, and dynamic audio textures in film and games. In these contexts, it functions primarily as a sound generation technique rather than as a signal transformation process.

In parallel, convolution has become one of the most widely used signal processing techniques in audio production. Convolution enables a signal to be filtered by an impulse response (IR), allowing the spectral and temporal characteristics of acoustic spaces or hardware devices to be imposed on an input signal. Convolution reverbs are ubiquitous in music production, film post-production, and game audio, where they provide highly realistic simulations of physical spaces and device responses.

Traditional convolution systems, however, operate under a relatively fixed paradigm: a single impulse response is convolved with an incoming signal. While this approach has proven extremely effective, it constrains the creative possibilities available to sound designers. In particular, most convolution implementations assume a static IR or allow only discrete switching between impulse responses. Continuous manipulation or recombination of impulse responses during playback is rarely supported, due to computational cost and artifacts from abrupt IR changes.

From a creative perspective, sound designers often seek to explore hybrid or evolving sonic spaces. In physical sound design practices such as Foley, artists routinely experiment with collections of materials and objects to discover new textures and acoustic interactions. Digital sound synthesis offers similar opportunities for exploration: granular and concatenative techniques allow the recombination of audio fragments into new sonic structures. Extending this philosophy to convolution raises an intriguing question: what new textures emerge when impulse responses themselves become recombinable elements of a corpus?

1.1. Proposed Framework

This paper introduces a Concatenation-Driven Convolution (CDC) framework that combines concatenative synthesis with a dynamically updating convolution engine. In the CDC framework, a corpus of source audio is analyzed and organized using techniques drawn from concatenative synthesis, enabling relationships between segments to be visualized and navigated by the user. Rather than selecting a single impulse response, the system allows users to traverse the corpus, generating evolving impulse responses through controlled concatenation and interpolation of segments.

The resulting impulse responses are fed into a live convolution engine capable of updating the IR continuously during playback. This architecture allows the convolution process itself to become a dynamic synthesis mechanism rather than a static effect. By merging concatenative corpus navigation with real-time convolution processing, the CDC framework provides a means of

exploring hybrid and evolving reverberant textures derived from combinations of impulse responses. Crucially, the corpus need not consist of measured impulse responses — any audio material can serve as the navigable source, extending the framework's applicability beyond spatial simulation into timbral processing and cross-synthesis.

A key technical challenge in this approach is ensuring that transitions between impulse responses occur smoothly and without artifacts. Existing approaches to IR morphing — including output crossfading and DFT-domain filter exchange [1, 2, 3, 4] — address this by blending the outputs of two simultaneously running convolution engines. However, such dual-engine architectures run two convolutions in parallel during a transition and maintain two independent convolution states: the incoming engine begins processing from a zero state and requires a warmup period roughly equal to the IR length before its output reaches full energy. When transitions are rapid or continuous, this warmup gap produces a consistent loss of output energy. The proposed system instead employs a single-engine frequency-domain kernel interpolation strategy, in which the FFT partition representations of the current and target IRs are blended directly within the active convolution engine. By preserving a single continuous convolution state throughout the transition, the approach avoids this warm-up energy loss and supports continuous IR updates at any rate, all within a single convolution engine.

2. RELATED WORK

2.1. Concatenative Synthesis

Concatenative synthesis selects and recombines short audio segments from a pre-analyzed corpus according to perceptual descriptors, drawing conceptual foundations from Xenakis' stochastic organization of audio fragments [5]. Early adoption was constrained by the computational cost of corpus analysis, limiting the technique to offline or batch workflows. It gained prominence through speech synthesis in the 1990s [6, 7], and as processing power increased, expanded into musical and sound design applications [8]. Tools such as Mosevius [9] and CataRT [10] established a paradigm centered on projecting a corpus into a 2D descriptor space and navigating it via manual or feature-guided control — an interactive model that has since informed much of the field.

Subsequent work explored new interaction modalities and application domains: Beller [11] introduced gestural control for performance contexts, later extended by Zbyszyński et al. [12] to incorporate machine learning; Mosaic [13] applied corpus-based selection to live audio effects via real-time feature analysis; and further applications have explored rhythmic composition [14], physics-based sound synthesis [15], and immersive virtual reality interfaces [16]. Across these systems, the core workflow remains consistent: a corpus is analyzed offline, organized by descriptor similarity, and navigated interactively to drive synthesis.

The CDC framework adapts this corpus navigation and synthesis machinery not as a direct sound generation technique, but as a means of continuously generating dynamic impulse responses from arbitrary audio material.

2.2. Real-Time Convolution

A longstanding goal in audio production has been to impose the acoustic characteristics of physical spaces and devices onto arbitrary signals. Real-time convolution has become the standard means of achieving this, typically implemented in the frequency domain: input and impulse response are transformed via FFT, multiplied spectrally, and reconstructed via inverse FFT. Partitioned convolution divides the IR into blocks processed incrementally; non-uniform schemes use small early partitions to minimize latency while larger partitions handle the tail asynchronously [17, 18]. The increasing availability of measured impulse responses of rooms, concert halls, and hardware devices has further cemented convolution reverb as a ubiquitous tool in music production and post-production.

A more challenging problem arises when the impulse response changes during processing. Mourjopoulos et al. [1] provided foundational analysis of time-varying digital audio filters and their perceptual implications. Brandtsegg et al. [2] addressed live convolution through time-domain coefficient replacement, fading the input to the outgoing engine while introducing the incoming one. Wefers and Vorländer [3] and Franck [4] proposed more computationally efficient approaches using DFT-domain filter crossfading, exploiting the compact frequency-domain representation of the fade window to reduce cost relative to running two fully independent engines. A dual-engine realization, however, keeps a separate convolution state for the incoming IR, so it begins from a zero state and incurs an energy loss during transitions as it warms up — a limitation that becomes pronounced under rapid or continuous IR updates.

Furthermore, most existing approaches focus on transitions between a small number of predefined impulse responses rather than on the exploration of large corpora. A notable exception is the work of Schwarz et al. [19], which combines corpus-based concatenative synthesis with real-time convolution: grains selected from a 2D descriptor-space corpus serve as impulse responses for a live contact microphone input, with smooth transitions achieved by mixing multiple simultaneous convolution voices in the time domain. This work demonstrates the creative potential of corpus-driven convolution and explicitly identifies single-engine frequency-domain kernel interpolation as a natural next step.

3. SYSTEM OVERVIEW

The proposed CDC system is designed as a general-purpose audio processing framework that bridges the gap between corpus-based concatenative synthesis and real-time convolution audio effects. The framework makes two distinct contributions. Conceptually, it reframes the synthesized audio stream not merely as output, but as a dynamic, time-varying impulse response (IR) that transforms an incoming signal in real time, integrating established corpus-navigation and concatenative-synthesis techniques into the convolution signal path. Its concrete technical contribution is the single-engine frequency-domain kernel interpolation scheme of Section 5.2, which makes such continuous IR updates practical while preserving the internal convolution state.

3.1. System Architecture

The CDC architecture is comprised of two core functional blocks: the **Concatenation and IR Generation Module** and the **Live Convolution Engine**, as illustrated in Figure 1.

1. **Concatenation and IR Generation Module:** This module manages the analysis and retrieval of audio grains from a pre-defined corpus. The grains are organized using a Self-Organizing Map (SOM) and synthesized into a continuous

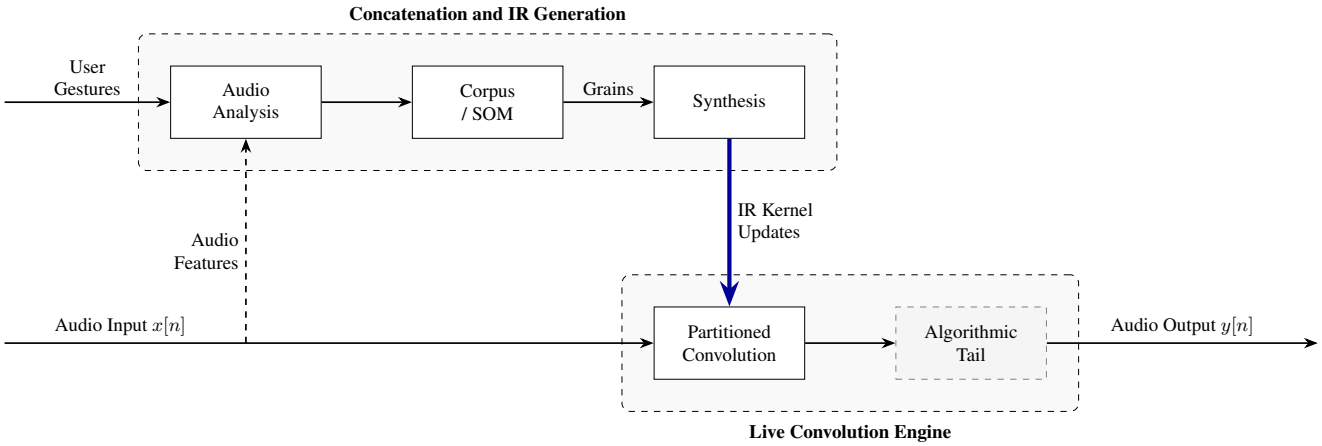

Figure 1: Block diagram of the CDC architecture. The Concatenation and IR Generation module analyzes audio features from the live input, driving navigation of a static audio corpus via user gestures, and generates grain-based streams of impulse responses. These kernel updates are fed continuously into the Live Convolution Engine. An optional algorithmic tail stage (dashed) extends the late reverberation decay in series after the convolution output.

stream of audio based on user interaction (gestures) and real-time audio analysis.

2. **Live Convolution Engine:** This module performs real-time partitioned convolution on a live input signal. Unlike traditional convolution processors that use static IRs, this engine is optimized to accept a continuous stream of kernel updates from the Concatenation and IR Generation module. An optional Freeverb-based algorithmic tail stage extends the late reverberation decay beyond the synthesized IR length.

3.2. The Integration Strategy

The concatenative synthesis output is coupled directly to the convolution kernel buffers: whereas a synthesizer and a convolution reverb would normally operate in series, here the "synthesis" stage instead produces the filter coefficients for the convolution stage. As the Concatenation and IR Generation module outputs new audio segments, they are transformed into the frequency domain and injected into the engine, so the convolution evolves continuously with corpus navigation, affording an instrument-like responsiveness in which reverb design becomes an act of corpus exploration rather than IR selection.

4. CONCATENATION AND IR GENERATION

The Concatenation and IR Generation Module is responsible for synthesizing the stream of impulse responses piped into the convolution engine. By utilizing a corpus-based approach [8], the system allows for the navigation of complex timbral spaces, transforming traditional concatenative synthesis from a direct sound-production method into a control source for the convolution engine.

4.1. Corpus Analysis and SOM Organization

A static corpus of audio segments is analyzed and organized offline to facilitate real-time navigation. Each segment is characterized by a multi-dimensional feature vector.

4.1.1. Feature Extraction

Each grain is analyzed using a sliding-window FFT, with a frame size of 1024–4096 samples depending on sample rate, a Hamming window, and a hop factor of 4. Audio is resampled to a common rate prior to analysis using windowed sinc interpolation. Features are averaged across the grain's timeline to produce a single fixed-length descriptor vector per grain. The following features are extracted: **RMS energy**, **zero-crossing rate** (with a small DC offset to suppress noise-induced crossings near zero), **spectral centroid**, **spectral flatness**, **spectral slope** and **intercept** (from a band-limited linear fit to the spectrum), **spectral energy ratio**, **spectral bandwidth**, and **fundamental frequency** (f_0, via the Harmonic Product Spectrum).

4.1.2. Self-Organizing Map (SOM) Topology

To project the high-dimensional feature space onto a navigable 2D grid, the system employs a Self-Organizing Map (SOM) [20] — an unsupervised neural network that organizes grains such that neighboring nodes correspond to segments with similar acoustic features. This topological structure supports gesture-based traversal of the corpus, producing gradual timbral variation in the generated IR as the user moves across the map.

Feature vectors are normalized to $[0, 1]$ (per-corpus min–max for the relative descriptors; frequency-based features such as f_0 by a fixed reference), and users may assign per-feature weights prior to training to emphasize relevant dimensions for a given corpus. The SOM may alternatively be trained directly on raw DFT magnitude bins. The map is an 80×76 rectangular lattice trained online, one grain presented per iteration, for approximately 25 passes over the corpus. The Gaussian neighborhood anneals from a global scale down to a floor of roughly one cell while the learning rate decays toward zero, giving an initial ordering phase followed by fine convergence; flooring the neighborhood keeps updates spread over neighboring nodes rather than collapsing onto the winning node alone, preventing the late-stage folding that would otherwise inflate topographic error [21].

4.2. Gesture-Based Matching and Traversal

Interaction with the corpus is defined by "gestures" — user-defined paths or regions drawn directly onto the SOM grid. These gestures are then traversed based on real-time audio analysis of the input audio, live user input, or fixed parameters (such as LFOs).

- **Path Traversal:** Grains are selected from the SOM along the drawn path, with optional randomization within a variable range. The traversal position may be driven by live input features or advanced at a fixed rate.
- **Regional Selection:** Grains are sampled stochastically from the selected map region, producing a consistent but evolving IR texture. Live input may optionally modulate the sampling radius with the region.

4.3. Synthesis

The selected grains are reconstructed into a continuous audio stream using overlap-add (OLA) synthesis. Each grain is windowed with a Tukey (tapered cosine) window, whose flat-top region preserves the grain's energy while the cosine tapers smooth the boundaries. The overlap is set equal to the taper length, approximating constant-power summation across grains and minimizing amplitude modulation artifacts that would otherwise propagate into the convolution engine as periodic fluctuations. Additional synthesis parameters — including grain size, density, pitch, and playback direction — allow further control of the synthesized IR stream independently of corpus navigation.

4.4. Operational Modes: Live vs. Baked

The system supports two IR-management workflows, catering to different performance and compositional contexts.

- **Live Mode (continuous update):** the Concatenation Module streams audio directly into the convolution engine, with the impulse response updated continuously as gestures are traversed — driven by real-time user interaction, live audio analysis, or automated parameters, enabling both performative and generative use. This is the primary mode of the CDC architecture.
- **Baked Mode (on-demand update):** the user "freezes" a gesture, rendering it to a single static IR that updates the convolution kernel only upon an explicit user trigger — useful for traditional reverb applications where a stable, predictable spatial profile is desired after an initial exploration phase.

4.5. Special Considerations for Reverb Effects

While the framework is general-purpose, two reverb-oriented options shape the synthesized IR stream before injection into the convolution engine.

- **Exponential decay envelope:** since the grains are drawn from a corpus without enforced temporal structure, the IR stream does not inherently decay like natural reverberation; an arbitrary envelope $h'[n] = h[n]\, e^{-\lambda n / F_s}$ (λ setting the decay rate) shapes the energy profile into a perceptually natural reverberation decay, regardless of the timbral content of the grains.
- **Stereo width control:** a hybrid parameter combines per-grain panning with a global mid/side (M/S) scaling of the IR's stereo difference; together these allow independent control over both the local spatial texture and the global width of the generated IR.

5. LIVE CONVOLUTION ENGINE

The Live Convolution Engine convolves the input signal with a dynamically updated impulse response using a non-uniform partitioned FFT scheme, employing frequency-domain kernel interpolation to support continuous IR transitions within a single engine at low real-time CPU overhead.

5.1. Partitioned Convolution

Let $x[n]$ be the input and $h[n]$ a causal impulse response of length L. Linear convolution is

$$y[n] = \sum_{k=0}^{L-1} x[n-k]h[k]. \tag{1}$$

For large L, the system employs partitioned FFT convolution [17, 18]. The impulse response is decomposed into P partitions of length N:

$$h[n] = \sum_{i=0}^{P-1} h_i[n - iN], \tag{2}$$

with each $h_i[n]$ supported on $0 \le n < N$.

Processing the input in blocks of length N, the frequency-domain accumulation is

$$Y_k(\omega) = \sum_{i=0}^{P-1} X_{k-i}(\omega) H_i(\omega), \tag{3}$$

where $X_k(\omega)$ are input spectra stored in a circular buffer. Time-domain reconstruction is performed using overlap-add (OLA). The per-block complexity is $O(N \log N)$.

To reduce latency, a two-stage non-uniform scheme is used:

$$h[n] = h_{\text{head}}[n] + h_{\text{tail}}[n]. \tag{4}$$

The head is processed with small partitions to minimize latency; the tail uses larger partitions computed asynchronously.

5.2. Frequency-Domain Kernel Interpolation

The impulse response used by the convolution engine is continuously updated by the concatenative synthesis module.

A common approach to impulse-response interpolation employs two independent convolution engines and crossfades their outputs [1, 2]:

$$y[n] = (1-\alpha)(x * h_1)[n] + \alpha(x * h_2)[n]. \tag{5}$$

In a stateless convolution model, where the full direct-form sum is evaluated independently at each output sample, this formulation is algebraically equivalent to convolving the input with an interpolated impulse response. However, in practical real-time implementations using partitioned FFT convolution, the convolution

process maintains internal state including overlap buffers and input partition histories. In a dual-engine architecture these states evolve independently, so transitions occur only at the output stage rather than within the convolution process itself: the prior engine's contribution is damped as it fades out, while the incoming engine must "warm up."

The proposed approach instead performs direct interpolation of convolution kernels within a single convolution engine. Let $H_{i,1}$ and $H_{i,2}$ denote the FFTs of partition i of impulse responses $h_1[n]$ and $h_2[n]$. Interpolation is defined as

$$H_i^{(\alpha)} = (1-\alpha)H_{i,1} + \alpha H_{i,2}. \tag{6}$$

The parameter α advances linearly from 0 to 1 over a user-configurable interpolation duration T, updated once per processing block.

By linearity of the Fourier transform,

$$h^{(\alpha)}[n] = (1-\alpha)h_1[n] + \alpha h_2[n], \tag{7}$$

and the resulting output is

$$y[n] = x * h^{(\alpha)}[n]. \tag{8}$$

Because interpolation occurs within a single convolution engine, the internal state of the partitioned convolution process remains continuous, preserving the convolution history as the impulse response evolves and providing improved output-energy stability during transitions. Frequency-domain filter-crossfading methods [3, 4] apply a related blend, but embed a time-varying fade window into the convolution to effect a bounded transition between filters; our method instead overwrites the kernel with a static linear blend each block, leaving the convolution unmodified and imposing no distinct transition state — so continuous morphing, freezing, and redirection are the same per-block operation.

The interpolation is performed as linear interpolation in the Cartesian domain. By the linearity of the DFT established above, this is algebraically identical to a time-domain crossfade of the two impulse responses, requiring no additional windowing or special treatment within the overlap-add framework. A consequence is that bins whose phases are near-antiphase across the two IRs will exhibit an energy reduction during the transition. This reduction is equivalent to the phase cancellation inherent in any linear time-domain crossfade. Polar interpolation (interpolating magnitude linearly and phase along the shortest arc) would avoid this reduction but does not correspond to a simple time-domain operation, requiring additional care within the overlap-add framework, and carries a significantly higher per-bin computational cost.

5.3. Block-Synchronous Updates and Length Adaptation

Kernel updates occur only at block boundaries, yielding a piecewise time-invariant approximation of an otherwise time-varying system:

$$h(n,t) \approx h_{\alpha_k}(n), \quad t \in [kN, (k+1)N). \tag{9}$$

That is, the impulse response is held constant within each processing block and updated discretely between blocks, ensuring consistency with FFT-based convolution and overlap-add reconstruction.

Transitions between impulse responses of different lengths are also supported. When the number of partitions differs between impulse responses, additional partitions are introduced as zero-padded kernels during expansion, while existing partitions are removed only after their delayed contributions have completed. This preserves causal consistency and prevents truncation artifacts.

5.4. Computational Cost

The proposed method uses a single convolution engine, requires no additional FFT passes, and adds only linear interpolation per partition. The overhead relative to standard partitioned convolution is negligible.

5.5. Optional Late Reverberation Stage

Although architecturally downstream of the core convolution engine, an optional late reverberation stage completes the signal chain. The output of the partitioned convolution, representing the early reflection structure of the synthesized IR, is fed into a second-stage algorithmic reverb unit. This serial arrangement reflects the physical structure of natural reverberation, in which early reflections give way to a dense, diffuse late tail, and allows the late decay to be extended well beyond the length of the synthesized IR.

The algorithmic stage is based on a Freeverb architecture [22], a well-established feedback delay network (FDN) design employing parallel comb filters followed by series all-pass filters. The size and decay characteristics of the algorithmic stage can be controlled independently of the concatenative IR content, allowing the timbral character of the reverb to be decoupled from its overall decay length and room impression. This enables a wider range of spatial effects than either stage could achieve in isolation.

6. IMPLEMENTATION AND EVALUATION

6.1. Implementation

We implemented the proposed CDC framework in C++ using the JUCE framework [23] as a cross-platform audio plugin supporting the VST3 and Audio Unit (AU) formats. Corpus analysis and SOM organization are performed offline during an initial loading phase, with memory allocation and file I/O confined to this initialization stage. All runtime processing — grain synthesis, kernel updates, and partitioned convolution — runs from pre-allocated buffers, with each grain synthesized and injected into the convolution kernel within a single audio block, keeping the signal path allocation-free and well within the real-time budget.

The user interface (Figure 2) centers on a 2D SOM corpus panel onto which the analyzed grains are projected. Gestures (paths or regions) are drawn directly on this map and define the IR generation trajectory, with additional controls for impulse response generation, kernel injection, and convolution parameters.

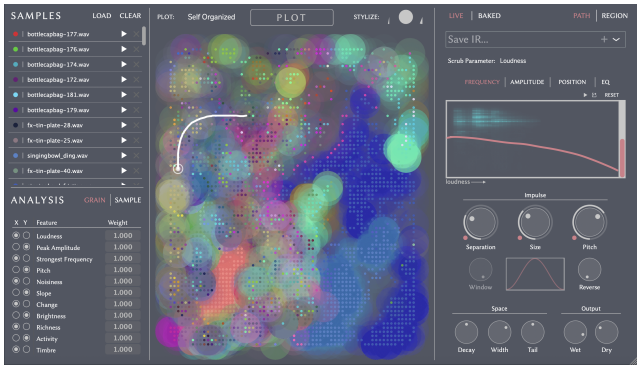

Figure 2: Interface of the CDC plugin implementation, showing the analysis, SOM corpus navigation, and controls panels.

6.2. Evaluation

The proposed frequency-domain interpolation engine was evaluated against a baseline employing dual-engine time-domain crossfading, a widely-implemented representative of the dual-state IR-morphing methods. Tests were conducted at 44.1 kHz with a block size of 512 samples, using a two-stage partition scheme with head and tail sizes of 512 and 1024 samples respectively. Impulse responses of approximately 3–4 seconds (Church, Hall, and Spring reverbs) were tested across four interpolation regimes: full (∼2 s), medium (∼0.67 s), short (∼0.16 s), and continuous (switching every ∼0.16 s, so the engine is perpetually interpolating).

6.2.1. *Computational Efficiency*

The computational cost was measured as mean CPU time per block (μs, summed across all threads), in two conditions: non-real-time (continuous processing) and simulated real-time (sleeping between normally spaced audio callbacks, with background threads continuing to work). The simulated real-time condition exercises background thread scheduling under realistic timing constraints. As shown in Figure 3, both engines perform similarly outside of interpolation regions, but the CDC engine exhibits lower cost during active interpolation in both modes.

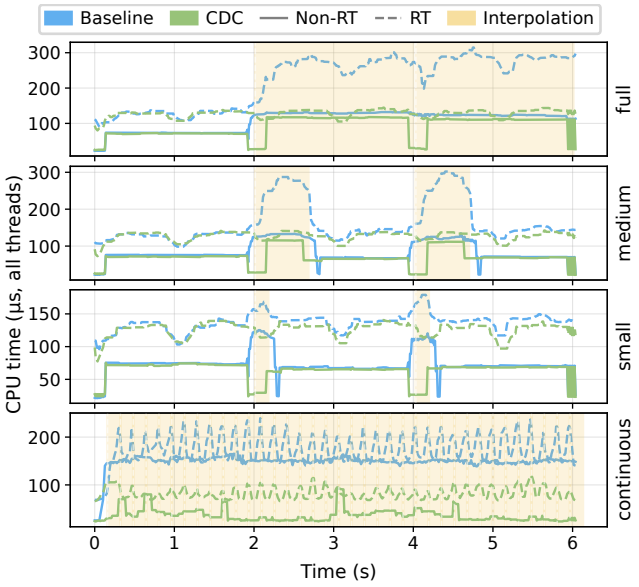

Figure 3: Mean CPU per block by engine and interpolation length. Solid = non-RT; dashed = simulated RT. Blue = baseline; green = CDC. Yellow marks interpolation regions.

To further characterize the relationship between interpolation activity and cost, we swept interpolation occupancy (i.e. the fraction of blocks during which interpolation was active) continuously from 5% to 100%. As shown in Figure 4, the baseline cost increases roughly linearly with occupancy in both modes, while the CDC engine remains nearly flat across the full occupancy range. This is consistent with the single-engine architecture avoiding the redundant FFT and IFFT passes required by the dual-engine approach during interpolation. On an Apple M4 Pro, mean CPU per block for the proposed engine stayed under ∼150 μs even under continuous interpolation (summed across all threads) — about 1% of the 11.6 ms block period, indicating ample real-time headroom.

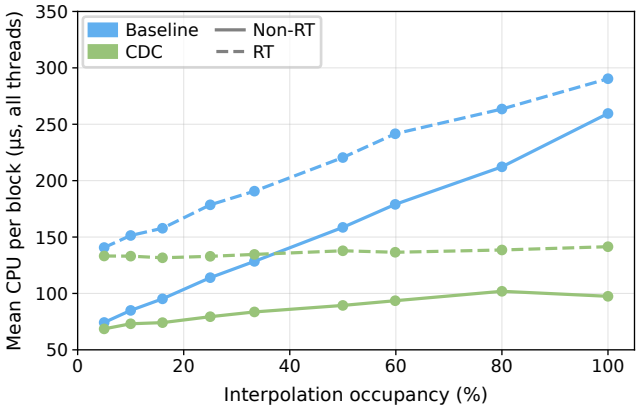

Figure 4: Mean CPU per block vs. interpolation occupancy. Solid = non-RT; dashed = simulated RT. Blue = baseline; green = CDC.

6.2.2. *RMS Energy Stability*

Figure 5 shows the RMS loudness for both engines across interpolation conditions. The baseline dual-engine architecture exhibits a noticeable deadzone (dip in energy) during the transition period. This occurs because the second convolution engine begins processing the input from a zero-state at the start of the crossfade, requiring a warmup period of roughly the length of the impulse response before its internal convolution state is fully populated.

When the interpolation length is short relative to the IR length, the baseline's output energy drops, as the first engine is faded out before the second has reached steady state. In contrast, the proposed CDC engine maintains a single continuous convolution state. By interpolating the kernels directly, the internal state remains valid throughout the transition, resulting in stable RMS energy regardless of interpolation speed. This is most apparent in the continuous interpolation case, where the baseline suffers from consistent energy loss while the CDC engine remains stable.

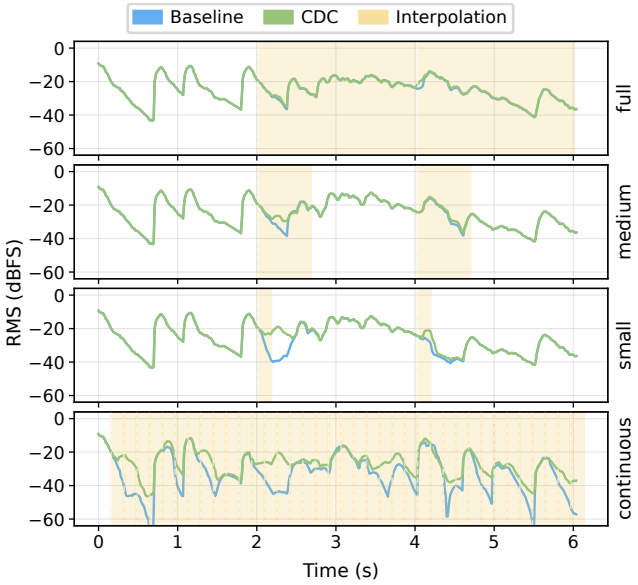

Figure 5: RMS loudness by engine and interpolation length. Blue = baseline; green = CDC. Yellow marks interpolation regions.

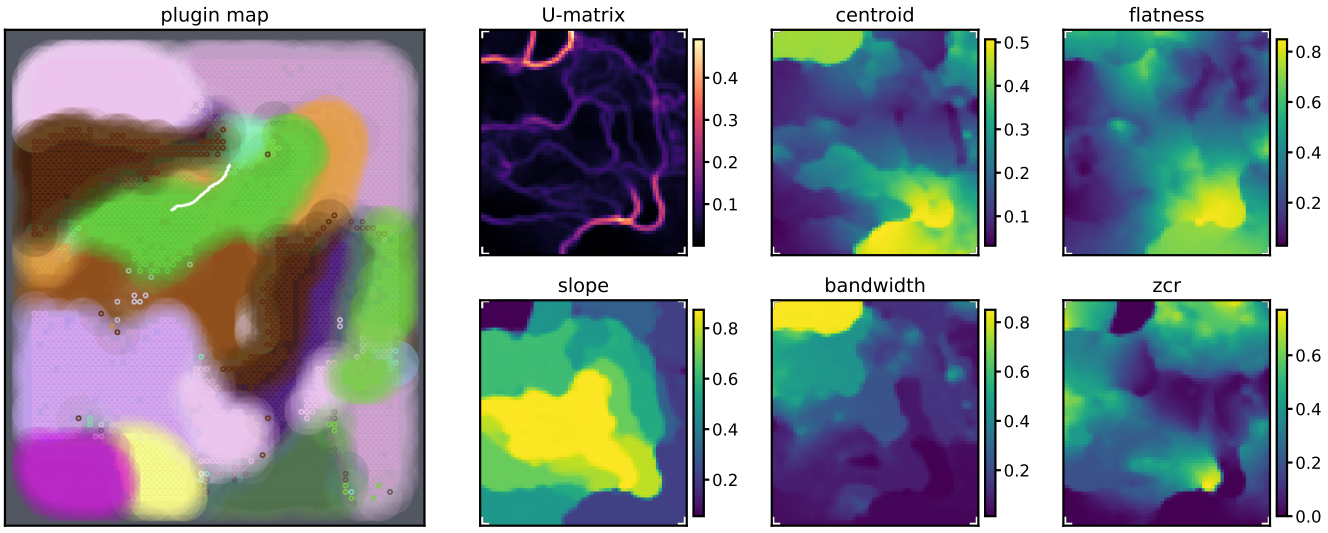

Figure 6: SOM organization for the chimes corpus. *Left:* the plugin map, grains colored by source file, with an example gesture path. *Right:* the U-matrix (neighbor distance; bright ridges are cluster boundaries) and five component planes showing smooth feature gradients.

6.2.3. SOM Organization

We also assess whether the SOM produces a well-ordered map of the corpus, using two standard quality measures [24]: the *quantization error* (QE; the mean distance from a grain to its best-matching node, in normalized feature units) and the *topographic error* (TE; the fraction of grains whose first and second best-matching nodes are not adjacent on the lattice [21]). Table 1 reports both for three corpora spanning percussive, resonant, and acoustic-space material that densely populate the map (∼4–6 grains per node), where these measures are meaningful.

Both measures are low across all three corpora (Table 1): QE is only 3–8% of the mean distance between grains, so each grain lies close to its representing node, and TE is 0.02–0.04, meaning for 96–98% of grains the two best-matching nodes are immediate neighbors, the signature of a smooth, unfolded map. The per-feature component planes (Figure 6) accordingly show smooth, coherent gradients.

Table 1: SOM organization quality (8 descriptor features, ∼4700-node map), for corpora that densely populate the map.

Corpus	Grains	QE	TE
Struck metal	27,665	0.031	0.030
Impulse responses	22,085	0.045	0.037
Chimes	18,063	0.022	0.022

6.3. Application Examples

The following examples illustrate representative use cases of the CDC framework.

6.3.1. Reverb Design and Spatial Exploration

When the corpus consists of measured impulse responses of acoustic spaces (rooms, halls, cathedrals, etc.), the SOM organizes them by perceptual and spectral similarity. Rather than selecting a single fixed reverb, the user traces paths through the space of reverberant environments, morphing continuously between them in real time. The single continuous convolution state preserves output energy across these transitions, enabling a style of reverb performance not possible with traditional static IR switching.

6.3.2. Timbral and Resonant Processing

When the corpus consists of recordings of physical objects or materials (struck metals, plastics, water, vocal textures, etc.), the resulting IR shapes the live input with the spectral coloration of that material. Navigating the SOM functions as gestural timbre morphing, with live audio features optionally driving the traversal to produce feature-reactive spectral processing. With more abstract or noise-like corpus material, smooth paths through the SOM reveal gradual spectral evolutions, while stochastic regional sampling produces evolving, grain-level variation — extending the system into generative sound design territory.

6.3.3. Cross-Synthesis and Live Performance

When timbrally rich input (noise, percussion, vocalizations, etc.) is used as the convolution input, the system can function as a real-time cross-synthesis instrument. The input's transient and spectral energy excites the resonant structure of the current IR, producing hybrid timbres that evolve as the performer navigates the corpus. This positions the CDC framework as a tool for live performance, where the boundary between reverberation and synthesis is intentionally blurred.

Audio examples using corpora of measured impulse responses, water recordings, struck metals, plastics, and vocal textures, along with supplementary materials, are available at:
`https://github.com/NiccoloAbate/concatenation-driven-convolution`.

7. CONCLUSIONS

This paper has presented the Concatenation-Driven Convolution (CDC) framework, a unified architecture that treats corpus-based concatenative synthesis as a real-time source of dynamic impulse responses for a partitioned convolution engine. By directly coupling grain synthesis output to the convolution kernel buffers, the system enables continuous, gesture-driven navigation of a timbral corpus as a form of expressive spatial and spectral processing, spanning conventional reverb design, resonant processing, cross-synthesis, and generative sound design.

A key technical contribution is a frequency-domain kernel interpolation scheme that operates within a single convolution engine. Unlike dual-engine crossfading approaches, interpolation occurs directly on the partitioned FFT kernels, preserving the internal convolution state throughout IR transitions. This was shown to avoid the energy loss that affects dual-engine architectures during rapid or continuous IR updates, while achieving lower computational cost under high interpolation occupancy. Objective evaluation further indicates that the SOM organizes diverse corpora into well-ordered, low-distortion maps with smooth feature gradients.

Several directions remain for future work. Our evaluation relies on narrow objective metrics; formal perceptual listening studies would strengthen the framework's claims — both comparing the single-engine kernel interpolation against dual-engine crossfading under controlled conditions, and relating SOM map distance to perceived timbral similarity. Polar interpolation, which avoids the energy reduction (phase cancellation) from near-antiphase bin pairs, remains implemented but underexplored. Finally, more advanced traversal strategies, such as interpolation between gestures, physical controller mappings, and tighter audio-driven navigation, present further opportunities for expressive control.

8. REFERENCES

[1] J. Mourjopoulos, E. D. Kyriakis-Bitzaros, and C. E. Goutis, "Theory and real-time implementation of time-varying digital audio filters," *Journal of the Audio Engineering Society*, vol. 38, pp. 523–536, 1990.

[2] Ø. Brandtsegg, S. Saue, and V. Lazzarini, "Live convolution with time-varying filters," *Applied Sciences*, vol. 8, p. 103, 2018.

[3] F. Wefers and M. Vorländer, "Efficient time-varying FIR filtering using crossfading implemented in the DFT domain," in *Proc. Forum Acusticum*, 2014.

[4] A. Franck, "Efficient frequency-domain filter crossfading for fast convolution with application to binaural synthesis," in *Proc. 55th AES International Conference on Spatial Audio*, 2014.

[5] I. Xenakis and M. J. Challifour, *Formalized Music: Thought and Mathematics in Composition*. Bloomington: Indiana University Press, 1971.

[6] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, vol. 1, pp. 373–376 vol. 1, 1996.

[7] J. P. Olive, "Section introduction. concatenative synthesis," in *Progress in Speech Synthesis*. Springer, 1997, pp. 261–262.

[8] D. Schwarz, "Corpus-based concatenative synthesis," *IEEE Signal Process. Mag.*, vol. 24, no. 2, pp. 92–104, Mar. 2007.

[9] A. Lazier and P. Cook, "MOSIEVIUS: Feature driven interactive audio mosaicing," in *Digital Audio Effects (DAFx)*. Citeseer, 2003.

[10] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-time corpus-based concatenative synthesis with CataRT," in *Proceedings of the 9th International Conference on Digital Audio Effects, DAFx 2006*, 2006.

[11] G. Beller, "Gestural control of real time concatenative synthesis," *ICPhS*, 09 2011.

[12] M. Zbyszyński, B. D. Donato, F. G. Visi, and A. Tanaka, "Gesture-timbre space: Multidimensional feature mapping using machine learning and concatenative synthesis," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2019, pp. 600–622.

[13] "Echobit," https://echobit.myshopify.com, accessed: 2022-05-01.

[14] C. Ó Nuanáin, P. Herrera, and S. Jordà, "Rhythmic concatenative synthesis for electronic music: Techniques, implementation, and evaluation," *Computer Music Journal*, vol. 41, pp. 21–37, 06 2017.

[15] S. An, D. James, and S. Marschner, "Motion-driven concatenative synthesis of cloth sounds," *ACM Transactions on Graphics - TOG*, vol. 31, 07 2012.

[16] C. Moore and W. Brent, "Interactive real-time concatenative synthesis in virtual reality," in *Proc. ICAD*, 2019, pp. 317–320.

[17] W. G. Gardner, "Efficient convolution without input/output delay," *Journal of The Audio Engineering Society*, vol. 43, pp. 127–136, 1995.

[18] F. Wefers, "Partitioned convolution algorithms for real-time auralization," Ph.D. dissertation, RWTH Aachen University, 2015.

[19] D. Schwarz, P.-A. Tremblay, and A. Harker, "Rich contacts: Corpus-based convolution of audio contact gestures for enhanced musical expression," in *Proc. New Interfaces for Musical Expression (NIME)*, London, UK, 2014.

[20] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[21] K. Kiviluoto, "Topology preservation in self-organizing maps," in *Proc. IEEE International Conference on Neural Networks (ICNN)*, vol. 1, 1996, pp. 294–299.

[22] J. O. Smith, *Physical Audio Signal Processing*. W3K Publishing, 2010, online book, accessed 2026-03-15. [Online]. Available: https://ccrma.stanford.edu/~jos/pasp/

[23] JUCE Team, "Juce: C++ framework for audio application and plugin development," 2025, accessed 2025-04-01. [Online]. Available: https://juce.com

[24] F. Forest, M. Lebbah, H. Azzag, and J. Lacaille, "A survey and implementation of performance metrics for self-organized maps," *arXiv preprint arXiv:2011.05847*, 2020.