Music Visualizer: Point Cloud, Mesh, and Volume Rendering of Audio Dynamics

Parker Corbitt

December 12, 2025

Abstract

This project presents an interactive visualization system for exploring audio dynamics in popular music at both micro and macro scales. Audio features are extracted from representative tracks and encoded as dense 3D point clouds, which are voxelized on the GPU into scalar fields and visualized using two complementary techniques: isosurface extraction and ray-marched volume rendering. In Song Mode, each STFT time-frequency bin becomes a point in a (time, log-amplitude, frequency) space, with A-weighted loudness serving as the primary scalar and harmonic-percussive balance mapped to hue. In Timeline Mode, per-song features are downsampled and stitched into a chronological point cloud whose axes encode global time, amplitude, and crest factor, enabling macro-scale analysis of loudness and dynamics behavior across decades. The renderer supports point visualization, marching-tetrahedra and dual-contouring surface extraction with an interactive isovalue, and real-time volume rendering with multiple transfer-function presets. Qualitative experiments across tracks from the 1970s through the 2010s show clear decade-level structure, including changes in harmonic-percussive emphasis, shifts toward higher amplitudes, and evidence of increased dynamics compression in later decades. Together, these results demonstrate that voxelized audio feature fields provide a useful and expressive basis for visual analysis of loudness and dynamics trends in music.

1 Introduction

Popular music over the last several decades has undergone a well-documented "loudness war," in which average levels have steadily increased while dynamic range has often decreased. These changes are typically quantified with scalar metrics such as RMS level or integrated loudness, but such measures do not convey the structure of how loudness, amplitude, and dynamics behave over time, frequency, or across large collections of songs. The motivation behind this project is to build an interactive visualization system that makes those structures visible and explorable.

The system takes audio features derived from individual tracks and from an aggregate timeline of many tracks, and maps them into 3D volumetric representations. At the micro scale, a Song Mode converts each short-time Fourier transform (STFT) time–frequency bin of a single song into a point in a 3D space whose axes are time, log amplitude, and frequency, with additional attributes encoding human perceived loudness and harmonic–percussive balance. At the macro scale, a Timeline Mode stitches per-song features into a single, long point cloud laid out along a chronological axis, with other dimensions encoding amplitude and crest factor. In both cases, the resulting point clouds are voxelized on the GPU into scalar fields that can be visualized as points, extracted as isosurfaces, or rendered as ray-marched volumes.

The main contributions of this project are threefold. First, it provides an end-to-end data pipeline that converts raw audio into dense 3D point clouds and fitted scalar fields for both

per-song and aggregate timeline views. Second, it implements a unified GPU rendering pipeline that supports point-cloud rendering, marching-tetrahedra and dual-contouring isosurface extraction, and real-time ray-marched volume rendering from the same underlying data. Third, it demonstrates how these representations can reveal trends in loudness, amplitude, crest factor, and harmonic–percussive balance, giving insight into both individual tracks and broader historical behavior.

2 Data Pipeline

2.1 Per-Song Features

Each song is first processed in Python to extract frame-wise audio features that feed the visualization. The audio waveform is converted to mono and analyzed with a short-time Fourier transform (STFT) using a Hann window, typically with an FFT size of 2048 samples and a hop size of 1024 samples. From the STFT magnitude, several derived quantities are computed.

First, basic energy measures are calculated, including the frame-wise root-mean-square (RMS) amplitude and a log-amplitude representation in decibels. To better match human hearing, an A-weighting curve is applied in the frequency domain; the resulting A-weighted magnitudes are converted to decibels and normalized to produce a perceptual loudness value in the range $[0, 1]$ for each time–frequency bin. This A-weighted loudness becomes the primary scalar field used later for isosurface extraction and volume rendering.

Second, harmonic–percussive source separation is applied to the complex STFT using a standard HPSS algorithm. The magnitudes of the harmonic and percussive components are used to compute a harmonic–percussive ratio

$$\text{hp\_ratio} = \frac{|H|}{|H| + |P| + \varepsilon},$$

which lies in $[0, 1]$ and encodes how "tonal" versus "percussive" each time–frequency bin is. This ratio is stored per vertex and later drives the color mapping in the renderer.

Third, dynamic-range-related features such as crest factor are computed at the frame level as the ratio between peak amplitude and RMS level. Time indices are normalized to $[-1, 1]$ after discarding frames that are completely silent, and frequency bin indices are similarly normalized to $[-1, 1]$. For Song Mode, each surviving time–frequency bin is finally written out as a five-float vertex

$$[x, y, z, \text{scalar}, \text{hp\_ratio}],$$

where x is normalized time, y is normalized log amplitude, z is normalized frequency index, scalar is normalized A-weighted loudness, and hp_ratio is the harmonic–percussive balance.

2.2 Timeline Construction

To build the aggregate timeline dataset, per-song features are downsampled and repacked into a single, chronological point cloud. Rather than representing a full time–frequency grid for each track, a fixed number of frames is sampled per song, and each sampled frame contributes a lower-dimensional point whose coordinates are designed to emphasize macro-scale dynamics.

Along the horizontal axis, points are laid out in song order so that the x coordinate encodes global timeline position, with optional offsets based on track year or ordering within a decade. The vertical coordinate y is derived from amplitude for that frame, capturing how strong the signal is at that moment in normalized units. The depth coordinate z encodes crest factor, mapping peak-to-RMS behavior into $[-1, 1]$ using the 1st and 99th percentiles of the crest-factor distribution

to clamp outliers. As in Song Mode, a scalar value derived from A-weighted loudness is stored to act as a density measure, and a hue channel is derived from harmonic–percussive statistics aggregated over the frame or a small neighborhood.

The result is a single timeline point cloud in the same $[x, y, z, \text{scalar}, \text{hp\_ratio}]$ layout as the per-song point clouds, but with different semantics: x reflects global chronology rather than local song time, y reflects frame-wise amplitude, and z reflects crest factor. This point cloud is then consumed directly by the renderer, which voxelizes it into a scalar grid for Timeline Mode visualizations.

3 Rendering Pipeline

3.1 Modes

At runtime, the Metal renderer operates in two data modes and three visualization modes. The data modes correspond to the two point-cloud datasets: Song Mode uses a per-song point cloud derived from the full STFT of a single track, while Timeline Mode uses the stitched timeline point cloud built from many songs. For each active dataset, the renderer can display the data as a raw point cloud, as an extracted isosurface, or as a ray-marched volume.

For volumetric operations, the point cloud is first voxelized into a regular grid whose resolution depends on the current mode. Song Mode typically uses a 128^3 grid, while Timeline Mode uses a higher 224^3 resolution to preserve detail along the longer horizontal axis. The grid is tightly fit to the bounds of the active point cloud by computing its axis-aligned bounding box and expanding it slightly to avoid clipping at the edges. A trackball-style camera allows the user to orbit, pan, and zoom, and a "fit to data" behavior recenters and reframes the view around the active point cloud, particularly when switching into Timeline Mode.

3.2 Surface Extraction

Once the scalar grid has been constructed from the point cloud, the system supports two GPU isosurface extraction algorithms: marching tetrahedra and dual contouring. In both cases, the underlying scalar field is the normalized loudness density produced by voxelization. Marching tetrahedra subdivides each grid cell into six tetrahedra and applies a local case table to generate triangles wherever the scalar field crosses the chosen isovalue. Dual contouring instead computes a single representative vertex per cell whose scalar range straddles the isovalue, then connects these dual vertices across edges whose endpoints span the isovalue.

Both algorithms are implemented as compute shaders in Metal and operate entirely on the GPU. The isovalue is exposed as an interactive control, allowing the user to sweep through different loudness levels and see how the extracted surface grows, shrinks, and changes topology. Per-vertex normals are either computed from triangle geometry (for marching tetrahedra) or estimated from local scalar gradients (for dual contouring), and a simple directional light above the scene provides shaded, three-dimensional appearance. For dual contouring, normals are flipped to maintain consistent orientation and lighting. At the top of the results section are examples of each method applied to the same scalar field.

3.3 Volume Rendering

For volume rendering, the same scalar grid used for isosurface extraction is sampled by a full-screen ray marcher. A lightweight vertex shader renders a single full-screen triangle, and the fragment

shader reconstructs a view-space ray for each pixel. That ray is intersected with the axis-aligned bounding box of the volume, and if an intersection exists, the shader marches along the ray in fixed steps, sampling the scalar field at each point.

At each sample, the scalar value is mapped through a transfer function that produces color and opacity. The implementation provides several presets (Classic, Ember, and Glacial) that emphasize different portions of the scalar range and use different color palettes to highlight structures of interest. Two user-controlled parameters, density and opacity scale, modulate how aggressively the scalar field contributes to the accumulated color and alpha along the ray. Together, they allow the volume to be tuned to reveal different aspects of the underlying loudness distribution.

3.4 Color Mapping

Across all modes, color encodes harmonic–percussive structure and scalar magnitude in a consistent way. For point rendering, each vertex carries its hp_ratio value, which is mapped into a five-stop color palette ranging from red through orange and white to light and deep blue. This maps predominantly percussive regions to one end of the palette (red) and predominantly harmonic regions to the other (blue), with mixed regions appearing in between.

For mesh rendering, voxelization also accumulates harmonic–percussive statistics into a secondary grid. During isosurface extraction, each mesh vertex inherits a color scalar derived from the average hp_ratio of its surrounding cells. The mesh fragment shader then maps this scalar onto a warm–cool gradient, optionally modulated by Phong lighting. In the current implementation, volume rendering uses scalar-only transfer functions without explicit hp-based modulation, emphasizing the overall loudness density; extending the ray marcher with multi-field transfer functions that incorporate hp information is left as future work.

4 Experiments

To evaluate the behavior of the system and the expressiveness of the visualizations, a series of qualitative experiments were conducted on a small set of commercially released tracks spanning multiple decades. The dataset includes songs taken from the billboard top year end track list, from the beginning of the 1970s to the end of the 2010s. For each track, a per-song point cloud is generated and visualized in Song Mode, and all tracks are also aggregated into a single timeline point cloud for Timeline Mode.

Several parameter sweeps were performed to understand how rendering choices affect the resulting images. In mesh mode, the isovalue was varied across the full $[0, 1]$ range to explore how constant-loudness surfaces emerge, merge, and disappear. Side-by-side comparisons of marching tetrahedra and dual contouring highlight their differing behavior on the same scalar field. In volume mode, the Classic, Ember, and Glacial transfer functions were compared under different density and opacity scales to see which settings best reveal internal structure versus large-scale envelopes.

5 Results and Discussion

When comparing dual contouring and marching tetrahedra for isosurface extraction in song mode, both methods reveal branching structures at low to moderate loudness that reflect the track's temporal evolution of dynamics across frequency bands, while higher isovalues isolate peak loudness events and produce smaller, often disconnected components highlighting moments of intensity. Both techniques also expose a noticeable noise floor, but dual contouring generally yields smoother, more

coherent surfaces with fewer artifacts since it places a single representative vertex per cell from the intersection geometry. Marching tetrahedra, by contrast, can produce more fragmented surfaces in regions with sparse data or steep gradients due to its local case-based triangulation. The choice between the two methods may depend on the desired balance between surface smoothness and fidelity to local variations in the scalar field.

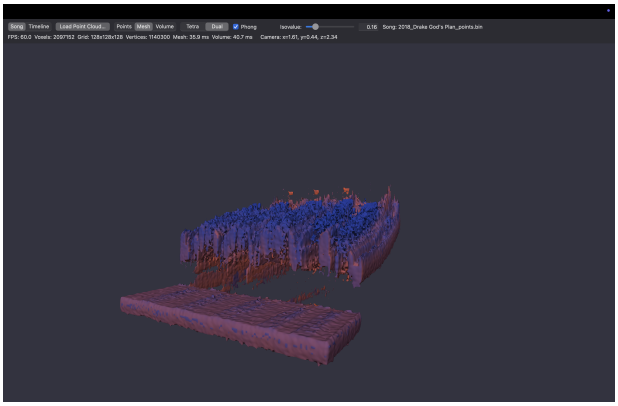

Figure 1: Dual Contouring

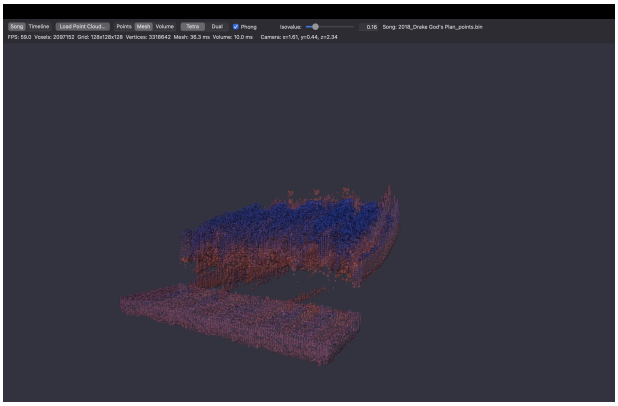

Figure 2: Marching Tetrahedra

The runtime of either algorithm is dominated by the voxelization step, which scales with the number of points in the point cloud and the grid resolution. Higher-resolution grids capture more detail but incur greater computational cost and memory usage. Marching tetrahedra was timed to have an average of 13.4 ms per extraction at 128^3 (song mode) resolution, while dual contouring averaged 16.2 ms under the same conditions. Under the 224^3 (timeline mode) resolution, Marching tetrahedra took on average 14.7 ms while dual contouring took 17.2 ms. Both methods scale roughly linearly with the number of occupied cells, so denser point clouds or higher resolutions lead to longer runtimes.

The data displayed by the volume renderer reveals similar structure to the isosurfaces, but with the added benefit of showing continuous density variations throughout the volume. Lower density and opacity settings produce a more transparent effect that highlights internal structures and gradual transitions in loudness, while higher settings yield a denser, more opaque appearance that emphasizes dominant loudness regions. The choice of transfer function also significantly affects the visualization: The Classic preset applies a mostly linear color and opacity mapping, producing a balanced view of the density field. Ember reshapes the density to make low-to-mid values contribute more visibly and yields a warmer orange-ish look, while Glacial suppresses low densities and ramps opacity more aggressively at higher values, emphasizing denser cores with a cooler blue.

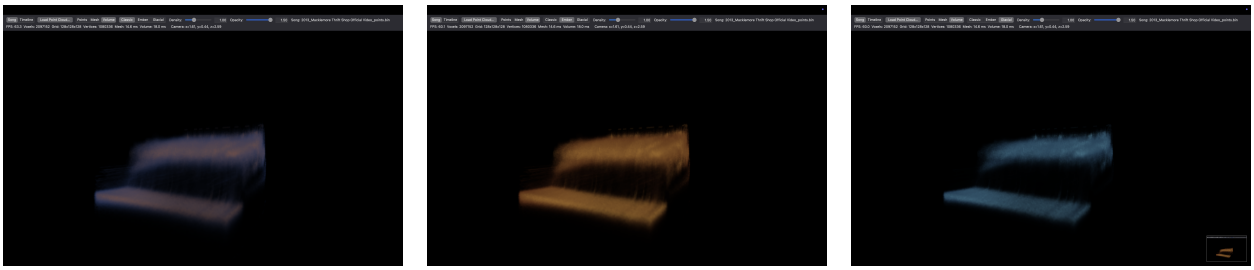

Figure 3: Classic (left), Ember (center), and Glacial (right) transfer functions applied to the same volume data.

Building the volumes was overall quicker, with an average of 11.3 ms at 128^3 for song mode and 10 ms at 224^3 for timeline mode.

Timeline mode indicates that percussive content was more prevalent from the late 1970s through the 1990s, while the 2000s and 2010s show a shift toward more harmonic material. This pattern likely reflects evolving production techniques and changes in dominant musical styles over time.

This percussive emphasis in the later 1970s, 1980s and 1990s may also help explain the higher perceived loudness in those decades. Drum-kit–centric production in this era often featured sharper transients and more high-frequency energy, especially from snares and cymbals. By contrast, the more harmonic-heavy textures common in the 2000s and 2010s often rely on sustained tones and chordal beds, which may contribute less to perceived loudness than aggressive percussive attacks.

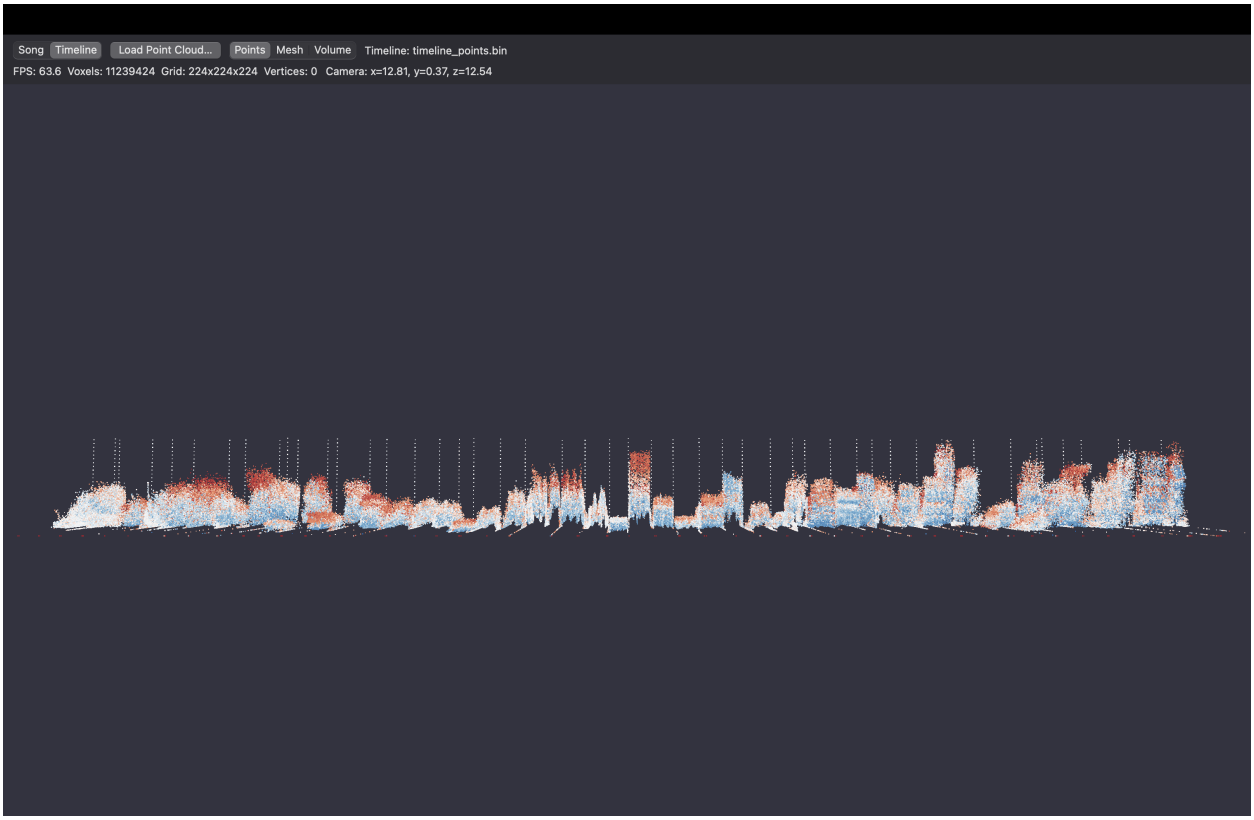

Figure 4: Data over the decades from the 1970s (left) to the 2010s (right), showing a trend toward higher amplitudes in later decades and percussive dominance throughout the 70s, 80s, and 90s.

A similar trend is also discovered in the crest factor dimension, where the 1980s and 1990s show a wider spread of crest factors, indicating more dynamic range and transient activity. The 2000s and 2010s exhibit a narrower crest factor distribution, suggesting more compressed dynamics and less variation between peak and average levels, with the late 2010s showing the most pronounced compression. The 1970s show a moderate crest factor spread, reflecting a balance between dynamic range and compression. Consistent with the trend of the "loudness war," the 2000s and the 2010s show a clear shift in less dynamic range with higher average amplitude levels compared to previous decades.

Loudness trends are also evident, with the 1980s and 1990s showing higher overall loudness levels, while the 2000s and 2010s exhibit slightly lower loudness on average. The 1970s display moderate loudness levels, consistent with the crest factor observations. This suggests that the peak

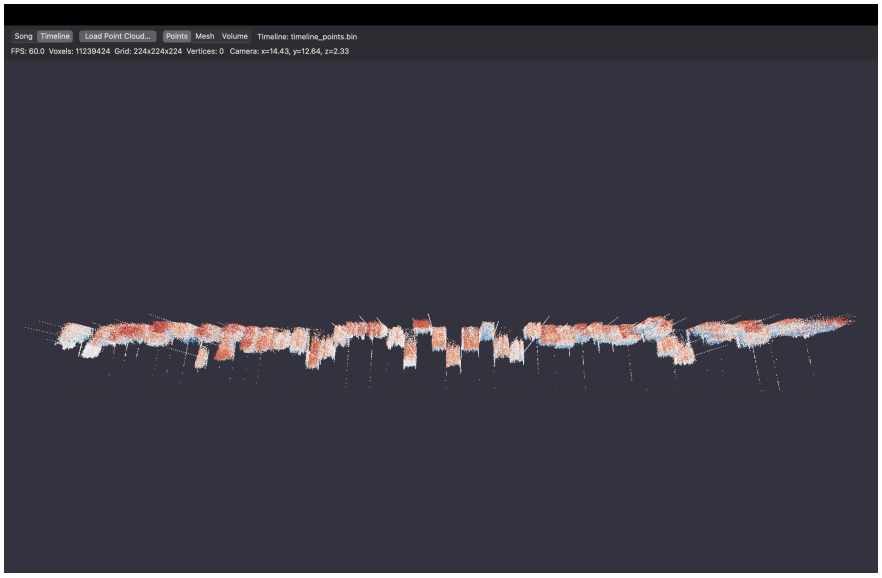

Figure 5: Crest factor data over the decades from the 1970s (left) to the 2010s (right), showing a trend toward lower crest factors (more compression) in later decades and a wider spread of crest factors in the 80s and 90s.

loudness levels were more aggressively pushed in the 1980s and 1990s, yet still retaining a sense of musicality with a moderate dynamic range. In the 2000s and 2010s, while average loudness remains high, the reduced dynamic range may have led to a perceived decrease in loudness impact. For lack of a better phrase, the music in the 2010s is just "loud" rather than "loud and dynamic." The volume renderings effectively illustrate these trends, with denser regions corresponding to higher loudness levels and more compressed dynamics in the later decades, as shown in the figure below.

Loudness densities across the timeline are well shown via isosurface/volume extractions. A volumetric render shown in figure 6 shows higher loudness density in the 80s and 90s with a significant gap in the 2000s and a slight return in the 2010s. The 1970s are moderate in loudness density. The density and opacity scales were increased to better reveal internal structures. Due to the longer timeline axis, the volume resolution was set to 224^3 to attempt to preserve detail. Figures 7 and 8 show a capture of dual contouring and marching tetraheda respectively, giving greater context to the volume rendering. Taking both figures 6, 7 and 8 together, it is clear that the 1980s and 1990s had the highest average loudness density, with a notable dip in the 2000s and a slight recovery in the late 2010s. Once again, the 1970s show moderate loudness density.

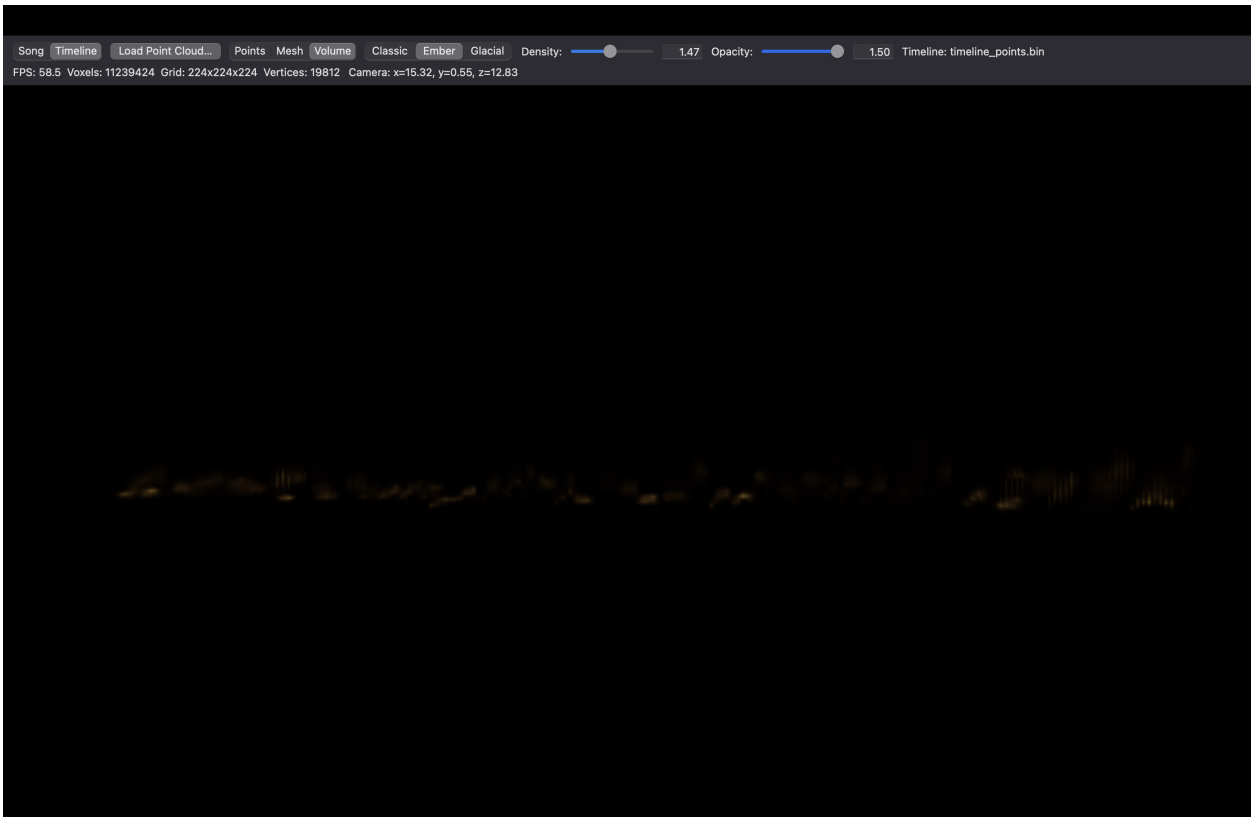

Figure 6: Volume rendering of loudness over the decades from the 1970s (left) to the 2010s (right)

Figure 7: Dual Contouring isosurface extraction of loudness over the decades from the 1970s (left) to the 2010s (right)

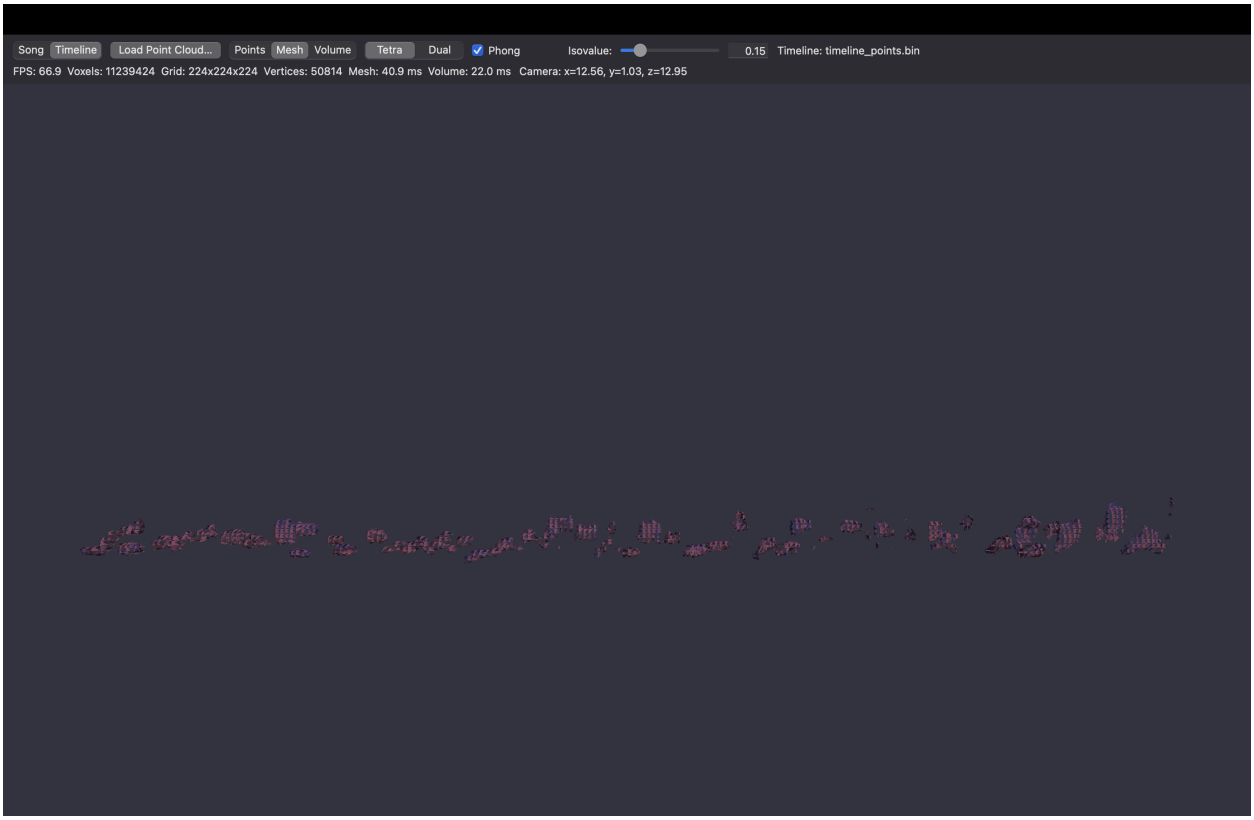

Figure 8: Marching Tetrahedra isosurface extraction of loudness over the decades from the 1970s (left) to the 2010s (right)

6 Limitations

A problem with the generated point cloud data in this experiment are the regions where the log amplitude is very low, which is most obvious for lower frequency bins. These low frequency regions exhibit a sharp "cliff" in the log amplitude values, creating a sort of gap in the resulting isosurfaces and volume renderings. This is likely due to the noise floor of the audio signal, where very low amplitude values are dominated by noise rather than meaningful signal content. As a result, these low-amplitude low-frequency regions have an observable discontinuity in the generated surfaces and volumes. Addressing this issue may require more sophisticated noise reduction or thresholding techniques during the feature extraction phase to ensure that only meaningful signal content contributes to the point cloud.

Another limitation lies within the interpretability of the results. Adding in some form of audio playback synchronized with the visualizations would greatly enhance the interpretability of the results. Being able to hear the audio while observing the corresponding visual structures would provide a more intuitive understanding of how specific musical elements map to features in the point clouds, isosurfaces, and volumes. This could help validate interpretations of the visual data and make it easier to connect observed patterns with actual sonic characteristics.

7 Conclusion

This work demonstrates a unified pipeline for converting audio-derived features into voxelized scalar fields that can be explored through point rendering, isosurface extraction, and ray-marched volume rendering. By supporting both per-song and aggregate timeline representations, the system enables analysis of loudness and dynamics structure within individual tracks and across decades of popular music. Marching tetrahedra and dual contouring provide complementary surface reconstructions of constant-loudness (or loudness-density) level sets, while volume rendering reveals continuous internal structure that is not visible in extracted meshes alone. The harmonic-percussive mapping further adds interpretability by encoding a fourth dimension into hue across rendering modes.

The experiments indicate clear macro-scale patterns consistent with known production trends: later decades exhibit higher amplitudes and narrower crest-factor distributions suggestive of stronger compression, while timeline visualizations also show changes in harmonic-percussive emphasis over time. At the micro scale, Song Mode surfaces and volumes expose branching loudness structures at moderate levels and isolate transient peak events at higher isovalues. Together, these observations support the central claim of the project: treating loudness-related behavior as a voxelized field yields visualizations that are both technically rich and musically informative.

Several limitations remain. Low-amplitude regions, especially at low frequencies, can create discontinuities likely driven by the signal noise floor, suggesting the need for improved thresholding or denoising during feature extraction. Interpretability would also benefit from synchronized audio playback to directly connect visual structures to perceptual events. Future extensions could incorporate multi-field transfer functions that combine loudness with harmonic-percussive information during volume rendering, and further refinements to data sampling and normalization in the timeline representation.