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

Parker Corbitt

December 12, 2025

Abstract

This project visualizes the relationship between loudness and amplitude in popular music using a dual-mode system that operates at both micro and macro scales. Audio features from representative tracks are converted into dense 3D point clouds, which are then voxelized on the GPU into scalar fields and visualized using two high-level computer graphics methods: isosurface extraction and GPU-based ray-marched volume rendering. In this formulation, each volume is treated as an audio density field, and ray marching integrates that density along viewing rays to reveal where particular loudness behaviors are persistent or dominant. Isosurfaces extracted from these point-cloud-derived volumes are color-mapped using additional audio attributes (such as harmonic–percussive balance), encoding a fourth data dimension into hue. The system supports an individual Song Mode that reveals fine-scale loudness structure within a single track, and an aggregate Timeline Mode that lays many songs out along a chronological axis to show how amplitude, loudness, and dynamics behavior evolve over time. This provides a comprehensive visualization framework for analyzing and exploring audio dynamics through advanced rendering techniques. The results show notable trends in loudness and dynamics across decades of popular music, illustrating the effectiveness of the proposed visualization methods.

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 at multiple scales, 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 or energy-related measures 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.

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 from a faint, wispy representation to a dense, nebula-like appearance, revealing 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 corpus 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

- Qualitative visuals: song vs. timeline; dual contour vs. marching tetra.

  In the case of comparing dual contouring and marching tetrahedra for isosurface extraction, dual contouring generally produces smoother surfaces with fewer artifacts. This is because dual contouring generates a single vertex per cell based on the average position of intersections, leading to a more coherent surface representation. In contrast, marching tetrahedra can exhibit more jagged edges due to its discrete case handling, especially in regions with high scalar field variation. It's worth noting that dual contouring produces a more consistent low amplitude region, while marching tetrahedra doesn't create the same region for the same range of isovalues. This behavior is exhibited below.

- Observations about dynamics/crest factor over years.
- Observations about ray-marched volume structures.

  In song mode, isosurfaces at low to moderate loudness levels tend to form branching structures that reflect the temporal evolution of dynamics across frequencies within the track. Higher isovalues isolate peak loudness events, often resulting in smaller, disconnected surface components that highlight moments of intensity. Dual contouring generally produces smoother surfaces with fewer artifacts compared to marching tetrahedra, which can exhibit more jagged edges due to its discrete case handling.

  In timeline mode, the aggregated point cloud reveals long-term trends in harmonic and percussive content across decades. The visualizations show how these characteristics evolve over time, with some decades exhibiting more pronounced harmonic structures and others more percussive ones.

  Volume renderings effectively capture the overall loudness distribution, with different transfer functions emphasizing various aspects of the data. The Classic transfer function highlights mid-range loudness levels, while Ember and Glacial accentuate high and low extremes, respectively. Adjusting density and opacity scales allows for fine-tuning the visibility of internal structures versus broader envelopes, providing insights into the persistence and dominance of certain loudness behaviors over time.

6 Discussion

- Interpretation: what the shapes/colors imply about energy variation.
- Limitations: data coverage, hp as hue, crest factor assumptions, grid resolution costs.
- Ablations or failure cases (e.g., sparse regions, banding).

  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. These regions should only be considered "meaningful" if they have variation in amplitude, which signifies the presence of a strong low frequency signal rather than just noise, and thus creating an isosurface with obvious variation.

  /sectionConclusion

- Recap insights and what the timeline vs. song views reveal.