SynthVis Explanation

Kush Parmar
University of Waterloo

May 8, 2026

Abstract

This paper details the mathematical models and backend data pipeline for a generative audio visualization engine. Unlike standard amplitude-based visualizers, this system constructs a continuous, persistent parametric curve mapping discrete time-domain audio signals to a multi-variable state space. The architecture utilizes Digital Signal Processing (DSP) for feature extraction, linear algebra for deterministic visual mapping, and low-pass filtering for kinematic smoothing. By injecting time-variant entropy, the system successfully balances deterministic mathematical rules with organic, non-repeating artistic output.

1 Introduction

The conceptual foundation for this project was inspired by introductory linear algebra coursework during my 1B mathematics term. Driven by a desire to bridge theoretical concepts with practical application, this engine was developed as an active learning tool to implement foundational matrix operations while exploring the field of Digital Signal Processing.

The primary objective of this system is to translate a one-dimensional audio waveform into a continuous, two-dimensional organic path. The core engineering challenge lies in bridging two different domains: discrete, localized audio data and continuous, fluid graphical rendering. To achieve this, the pipeline is divided into three distinct algorithmic phases: Feature Extraction, Matrix Mapping & Bounding, and Temporal Dynamics.

2 Feature Extraction and Signal Processing

The engine begins by ingesting a discrete time-domain audio signal. To simulate real-time processing and extract instantaneous features, the signal is chunked into localized frames corresponding to 1/60th of a second.

2.1 Root Mean Square (RMS) Envelope

To determine the global perceived loudness of a given frame, the algorithm calculates the Root Mean Square (RMS). For a discrete audio frame containing N samples, where $s[n]$

represents the amplitude of the n-th sample, the RMS volume V_{rms} is defined as:

$$V_{rms} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} s[n]^2} \tag{1}$$

This approach is chosen over a simple peak-amplitude measurement because RMS provides a mathematically robust representation of the signal's continuous power, which directly translates to a smoother visual scaling factor.

2.2 The Real Fast Fourier Transform (RFFT)

To isolate specific musical characteristics (such as a kick drum versus a hi-hat), the time-domain frame must be converted into the frequency domain. The system employs the Real Fast Fourier Transform (RFFT).

Standard FFT algorithms return both positive and negative frequencies, which is redundant for audio waveforms since they consist entirely of real-valued data. The RFFT exploits the conjugate symmetry of real signals to compute only the positive half of the frequency spectrum. This decision halves the computational overhead and memory footprint while yielding the exact magnitude spectrum required for audio analysis.

2.3 Spectrum Partitioning

Once the signal is in the frequency domain, the spectrum is partitioned into specific "buckets" using boolean masks. Let F be the set of frequencies obtained from the RFFT. The spectrum is divided logically based on standard acoustic ranges:

$$\begin{aligned} \text{Bass} &: f \in [20, 250] \text{ Hz} \\ \text{Mid} &: f \in (250, 4000] \text{ Hz} \\ \text{Treble} &: f \in (4000, 20000] \text{ Hz} \end{aligned}$$

For each frame, the mean magnitude of the frequencies within these masks is calculated, yielding three independent frequency envelopes.

2.4 Min-Max Normalization

A fundamental design requirement is that the visualizer must react consistently regardless of whether the uploaded audio track is mastered quietly or loudly. To achieve this, each extracted envelope vector E is normalized to a strictly bounded range of $[0, 1]$:

$$\hat{E} = \frac{E - \min(E)}{\max(E) - \min(E) + \epsilon} \tag{2}$$

where $\epsilon = 10^{-6}$ is added to the denominator to prevent division by zero in intervals of absolute silence. This maps all audio features to a uniform algebraic space, preparing them for matrix multiplication.

3 The Linear Transformation Model

The core translation from audio features to visual parameters is achieved via matrix mapping and bounding. This method was chosen because it allows for rapid, deterministic scaling and shifting of the normalized domains into specific graphical boundaries (e.g., pixel constraints or 8-bit color limits).

3.1 The State Vectors and Transformation Matrix

At any time step t, the extracted normalized audio features form the input column vector $\vec{x}_t \in \mathbb{R}^4$:

$$\vec{x}_t = \begin{bmatrix} \text{Volume}_t \\ \text{Bass}_t \\ \text{Mid}_t \\ \text{Treble}_t \end{bmatrix} \tag{3}$$

The behavior of the visualizer is governed by a user-defined weight matrix $W \in \mathbb{R}^{6\times4}$. Matrix multiplication projects the 4-dimensional audio space into a 6-dimensional target parameter space (Target Size, Target Speed, Target Angle, Red, Green, Blue).

3.2 Matrix Mapping & Bounding

To map the normalized projection into usable boundaries, a scaling vector $\vec{s}$ and a bias vector $\vec{b}$ are applied. The target state vector $\vec{y}_t$ is computed as:

$$\vec{y}_t = (W\vec{x}_t) \odot \vec{s} + \vec{b} \tag{4}$$

where $\odot$ denotes the element-wise product.

4 State Space and Temporal Dynamics

A major logical flaw in naive audio visualizers is the assumption that raw audio data maps perfectly to screen rendering. Using the target vector $\vec{y}_t$ directly results in discontinuous, highly jittery movement that fails to produce a cohesive visual curve. To resolve this, inertia must be mathematically simulated.

4.1 Low-Pass Filtering (Inertia)

A linear interpolation (LERP) function acts as a low-pass filter. Instead of snapping instantly to new values, the current kinematic state is forced to "chase" the target state, thereby simulating physical mass and momentum. Let v_t be the current speed and $\hat{v}_t$ be the target speed from $\vec{y}_t$. The smoothed velocity is updated iteratively:

$$v_t = v_{t-1} + \alpha_v(\hat{v}_t - v_{t-1}) \tag{5}$$

where $\alpha_v \in (0, 1]$ is the speed inertia coefficient. A parallel equation governs the steering angle utilizing an angular inertia coefficient α_θ.

4.2 Cartesian Kinematics

Once the smoothed velocity v_t and steering angle θ_t are calculated, the Cartesian coordinates of the path are derived using Euler integration:

$$x_t = x_{t-1} + v_t \cos(\theta_t) \tag{6}$$

$$y_t = y_{t-1} + v_t \sin(\theta_t) \tag{7}$$

5 Mathematical Entropy and Time-Variance

A critical challenge in generating a persistent parametric curve is the "Steady-State Problem." If a track contains a highly repetitive sonic profile (e.g., electronic dance music), a purely deterministic algorithm will eventually settle into a localized, repetitive geometric loop.

To counteract this and maintain aesthetic viability over long durations, a time-variant perturbation is injected into the kinematics. Let $p = t/T$ represent the normalized progress of the audio track, where $p \in [0, 1]$. An entropy multiplier iteratively loosens the inertia, and a time-dependent drift alters the steering:

$$\Delta\theta_{\text{drift}} = \sin(t_{\text{step}} \cdot (1 + p)) \cdot (k_1 + p \cdot k_2) \tag{8}$$

By scaling both the angular drift and the color palette offsets against p, the mathematical rules of the environment subtly shift over time. This continuous evolution ensures the generated art piece maintains global asymmetry and visual complexity from the first frame to the last.

6 Conclusion

The architecture detailed in this paper successfully bridges the gap between one-dimensional signal processing and two-dimensional kinematic rendering. By partitioning the pipeline, the system handles data in distinct, mathematically sound phases: the RFFT and RMS calculations extract normalized, reliable acoustic features; linear matrix projections bound these features into a strict parameter space; and finally, iterative low-pass filtering translates these discrete points into a fluid, continuous path.

Ultimately, this engine functions not merely as a reactive display, but as a simulated kinematic system where audio frequencies act as driving physical forces. By introducing time-variant entropy to disrupt steady-state loops, the algorithm demonstrates how rigid, deterministic mathematical constraints can be leveraged to generate highly organic, unstructured artistic outputs.