Learning Generative Models from Acoustic Sound Sources

Bachelor's Thesis
Lukas Middelberg
August 16, 2022

Supervisor:	Dr. Thomas Hermann
1st reviewer:	Dr. Thomas Hermann
2nd reviewer:	Dr. Jiajun Yang

Cognitive Informatics
Faculty of Technology, Bielefeld University

Contents

1 Motivation

Acoustic musical instruments can be played in numerous ways, resulting in a large variety of timbres. For instance, a trumpet will sound more “airy” depending on the looseness of the player’s embouchure and the amount of air delivered by the player. Similiarly, a guitar string sounds more brilliant the more strongly it is plucked. We can view EMBOUCHURE, AMOUNT OF AIR, and PLUCKING STRENGTH as instances of *control parameters* that influence the qualia of a sound such as AIRINESS and BRILLIANCE, and result in an overall different *timbre*.

Traditional keyboard controllers expose only a limited set of control parameters: Usually, they only allow to control pitch and the strength of attack. Multi-parameter keyboard controllers such as the ROLI Seaboard Block (Figure 1) offer more control parameters. The ROLI controller provides full control in the X, Y and Z direction. A note can be triggered by pressing a key in an arbitrary X and Y position with arbitrary pressure Z. After triggering the note it can be morphed by holding the finger down and moving over the squashy silicone surface. Thus, multi-parameter keyboards allow finer control of an instrument’s parameters and can lead to greater musical expressiveness.

To make use of the potential of multi-parameter keyboard controllers, we need instrument models that can be finely controlled by several parameters. In this thesis, we explore modelling an instrument’s timbre with respect to a set of control parameters by learning from audio recordings of acoustic musical instruments.

2 Introduction

Let us define a configuration of control parameters as an M-dimensional vector $\boldsymbol{\alpha}$ in *control space* $\mathbb{A} := \mathbb{R}^M$, and timbre as a vector $\mathbf{x}$ in *timbre space* $\mathbb{X}$ from we can resynthesize an actual audio signal. (We should model timbre in a way that allows us to easily recover a timbre description from an audio recording, and vice versa, synthesize an audio signal from a timbre. We will discuss the modelling of timbre in section 3.1)

Coined by Dr. Thomas Hermann, *Principal Harmonics* is the idea to use *Generative Models* (*Principal Component Analysis*, PCA in particular) to model an instrument’s timbre with respect to a set of control parameters. Generative Models learn a *continuous* probability distribution $P(X|A = \boldsymbol{\alpha})$ from a dataset of audio recordings, leading to a continuous mapping $f : \mathbb{A} \mapsto \mathbb{X}$.

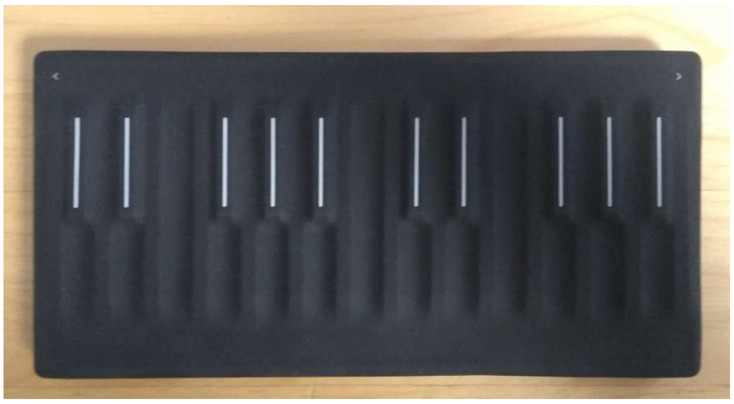

Figure 1: The “ROLI Seaboard Block” is a multi-parameter keyboard controller with a squashy silicone surface.

Both supervised and unsupervised methods are conceivable: For supervised methods, we would use a labeled dataset where all control parameters have a known semantic. For unsupervised methods (such as PCA), we would analyze our dataset of timbres without considering any labels, and learn the semantic of each control parameter in the process. Hence, the challenge with unsupervised methods is not only to reconstruct timbres accurately: We also need to find control parameters that are either motivated by the instrument modelled, or are otherwise musically useful.

For this thesis, we use a dataset provided by the Philharmonia Orchestra (year unknown). We focus on the cello (played *arco*, i.e. with a bow) and the guitar: The cello as an example for instruments that have a rather constant timbre, and the guitar as a representant for instruments with transient sounds that are rich in overtones at first and decay over time.

- A flexible timbre analysis toolkit was implemented based on Serra's *Sinusoids plus Noise* algorithm (Serra 1997) that reads an audio sample and generates a time-varying timbre representation (Chapter 3).
- PCA was applied to the timbres of one sample as well as to an ensemble of samples, yielding an instrument model (Chapter 4).
- Finally, the capabilities of the model were explored (Chapter 5). Originally, we planned to implement an interactive demonstration using a ROLI keyboard controller. However, we did not follow through due to technical problems.

Supplementary Material Accompanying materials are available on Bielefeld University's GitLab instance[1]. The repository contains the Python package developed for this thesis, called `principal_harmonics`. Audio samples are provided, referenced in this report with a 🔊 symbol. Finally, Jupyter notebooks reproduce the results and plots.

Conventions

- The terms *sample* and *frame* may be ambiguous. Where necessary, we write *audio sample* to refer to a full audio recording, i.e. a one-note recording of an instrument. We write *sample* to denote the instantaneous value of a sampled signal. For the timbre analysis, the signal is divided into small slices that are analyzed separately. We call these slices *frames* or *analysis frames*, indexed with n.
- Functions of a continous time variable are written as $f(t)$, and functions of discrete time values as $f[t]$. Vectors and vector-valued functions are printed bold ($\mathbf{x}[t]$).
- *Decibels* express the *ratio* of an amplitude measurement to a reference amplitude on a logarithmic scale (dB $= 10 \log_{10} \frac{A}{A_0}$) (Moore 1990, p. 72). We will be dealing with dimensionless amplitudes, so we choose the arbitrary reference value $A_0 = 1$.

[1]`https://gitlab.ub.uni-bielefeld.de/lukas.middelberg/principal_harmonics`

3 Gathering Data

To train a Generative Model, we first need to compile a timbre dataset. The Philharmonia Sound Samples dataset (Philharmonia Orchestra year unknown) forms the basis of our analyses and can be downloaded online. The dataset contains single-note audio samples of standard orchestral instruments and a number of percussion instruments. It also includes instruments not standard to a classical orchestra, such as the guitar. For the cello and the guitar, the Philharmonia Dataset provides different lengths of audio samples, and especially for the cello, a number of additional playing modes, such as MOLTO VIBRATO, NON-VIBRATO, ARCO-COL-LEGNO-BATTUTO. It also contains multi-note phrases. The guitar also includes samples played using HARMONICS, i.e. using flageolet technique. For the sake of clarity, we constrained us to the VERY-LONG, NORMAL guitar samples and the 1.5-second ARCO-NORMAL cello samples.

In this chapter, we describe our timbre model and the analysis procedure to obtain a time-discrete timbre $\mathbf{x}[n] = [A_1[n], A_2[n], \dots A_D[n]]^T$ from an audio signal. This chapter remains agnostic to the type of Generative Model to be used later. Additional preprocessing steps specific to Principal Component Analysis will be discussed in the next chapter.

3.1 Modelling timbre

In Psychoacoustics, the term *timbre* is not precisely defined. The American National Standarts Institute (ANSI) defines timbre as the "set of properties that allow to distinguish between two tones of equal loudness and pitch" (Weinzierl 2014, p. 5). For our analyses, we need a precise definition of a timbre's representation. In particular, we would like a timbre representation that allows us to easily calculate a time-varying timbre $\mathbf{x}(t)$ from a signal $s(t)$, and vice versa. Furthermore, for most data analysis algorithms we need our representation to be *constant-dimensional*[2] algorithm that tracks the peaks of a spectrum over time: (Serra 1997, p. 10f.). To that end, we will represent timbre simply as the amplitudes of the sound's overtones $\mathbf{x}(t) = [A_1(t), A_2(t), \dots, A_D(t)]^T$. This model, though convenient for further data analysis, makes some strong assumptions. In the following, we will motivate this choice and take note of the assumptions and simplifications that were made:

3.1.1 Psychoacoustic definition of timbre

According to Ellermeier, Hellbrück, and Schlittenlacher (2014, p. 70f), the perception of timbre depends on two factors: **Spectral patterns.** A timbre is partially characterized by its magnitude spectrum. However, analysing the spectrum alone is not sufficient to reproduce a timbre - this can be seen by listening to a recording of a piano note played in reverse: The reversed recording has an accordeon-like quality (Ellermeier, Hellbrück, and Schlittenlacher 2014, 🔊 1). Instead, we also need to consider the evolution of the spectrum with time: **Temporal patterns.** We should also consider the evolution of the spectrum with time. Long-term trends, continuity and periodicity are factors that could be looked at.

Since a suitable representation for temporal patterns is not immediately clear, we will focus on analyzing spectral patterns only; the behaviour of the model with respect to time (in our case: the decay of the plucked guitar string, and the vibrato-oscillation of the cello's timbre) is then recreated manually. To model spectral patterns, we use Serra's model of *Sinusoids plus Noise*:

[2]Other representations are also conceivable: Instead of a constant-dimensional representation, for more complex sounds it might be more useful to think in terms of *frequency trajectories* that can turn on and off at any time. See the TRACKING peak matcher in section 3.2.4

3.1.2 The Sinusoids plus Noise Model

Serra proposes to model musical signals $s(t)$ by separating them into a deterministic and a stochastic component (Serra 1997, p. 94). The deterministic component captures the "main modes of vibration of the system", and can be described as a sum of sines. The stochastic component captures "the energy produced by the excitation mechanism which is not transformed by the system into stationary vibrations" (i.e. the plucking of the guitar string, or the scratching of the bow on a cello string) "plus any other energy component that is not sinusoidal in nature". This leads to the model

$$s(t) = \sum_{i=1}^{N} A_i(t) \cos\left(\varphi_i(t)\right) + e(t), \qquad \varphi_i(t) = \int_0^t \omega_i(\tau) d\tau \tag{1}$$

where $A_i(t)$ denotes the time-varying amplitude and $\varphi_i(t)$ denotes the instantaneous phase of the sinusoid, coupled to its time-varying frequency $\omega(t)$. To derive a constant-dimensional representation from this model, let us make two more assumptions:

- Since the exact phase of a signal is usually indifferent to human perception, we discard the phase information for our timbre model.
- Furthermore, since we are dealing with single-note, single-instrument, harmonic sounds, we can expect that most of the energy of the sound will be concentrated in its overtones, i.e. in the frequencies that are integer multiples of its fundamental frequency. We assume that the signal contains no non-harmonic sinusoidal components.

Due to time constraints, we did not consider the noise component $e(t)$ of the signal. Hence, for these analyses, we will simply represent the deterministic part of a timbre as a vector $\mathbf{x}(t) = [A_1(t), A_2(t), \ldots, A_D(t)]^T$, where the i-th component denotes the amplitude found in the i-th partial of the sound. The next section describes the algorithm proposed by Serra to obtain timbre vectors at discrete time frames $\mathbf{x}[n] = [A_1[n], A_2[n], \ldots, A_D[n]]^T$.

3.2 Serra's Sinusoids Plus Noise Algorithm

Serra's algorithm uses the *Short-time Fourier Transformation* (STFT) (Miranda 2002, p. 51ff.) to take spectral measurements of the signal $s[t]$ at *discrete time frames* n. In a nutshell, it detects peaks in the magnitude spectrum of each frame, yielding a frequency, amplitude and a phase (ω, A, φ) for each peak. It is assumed that the spectrum of the deterministic component varies little with time. Hence, the peaks belonging to the deterministic component will be those that persist for a large amount of frames and whose frequency only varies a little with each frame n. Extracting these peaks $(\omega_i, A_i, \varphi_i)$, the signal of the deterministic component can be resynthesized in the time domain in a phase-correct fashion by interpolating between frames. The resulting deterministic signal $\hat{s}[t]$ is then subtracted from the original signal to generate the signal of the stochastic residue $e[t] = s[t] - \hat{s}[t]$. If the analysis worked, there should be no pitch audible in the stochastic residue, but just the noisy components of the signal, i.e. the plucking of the guitar string, or the scratching of the cello's bow. In the following, we briefly discuss each step of the algorithm, and point out the alterations we made.

3.2.1 Pitch detection

Detecting a signal's instantaneous fundamental frequency can inform the spectrum computation and the tracking of partials (see above). Serra proposes using Maher and Beauchamp's

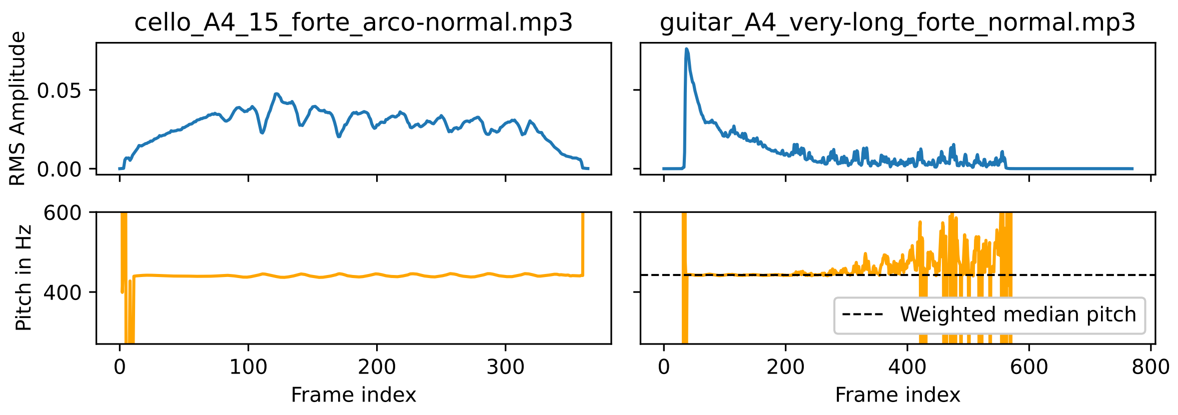

Figure 2: Pitch estimation using the YIN pitch tracker

fundamental frequency detection algorithm (Maher and Beauchamp 1994) as an on-line procedure that computes the most likely fundamental frequency from the set of peaks of the previous frame (Serra 1997, pp. 101–103). It requires four hyperparameters to be set. Because it contains a potential feedback loop (the fundamental estimation depends on the spectral peaks from the previous frame, which in turn depend on the fundamental estimation in the previous frame, and so on), it requires complex stepping logic to reevaluate a frame in case its fundamental estimate does not match the previous frame. Instead, I opted to use the YIN pitch detector proposed by Cheveigné and Kawahara (2002) which is readily available in `librosa`. It does not require setting hyperparameters, and can be run as a preliminary step before performing the spectral analysis, eliminating the feedback loop. Looking at Figure 2, we can see that the YIN pitch tracker performs well when the signal is loud. We can observe the cello's vibrato, and the comparatively constant fundamental frequency of the guitar string. However, the quality of the estimation degrades dramatically with decreasing amplitude, which is problematic for the guitar since its signal decays exponentially. But since the samples in the dataset are labeled with a note, we have an estimate of the fundamental that can be expected.[3] We clip the fundamental estimation one semitone above and below the expected frequency. Then we apply a weighted median, where the weight of each instantaneous frequency is a Gaussian probability density centered around our expected frequency. This way, we can get a reliable estimate for the signal's actual, constant fundamental.

3.2.2 Spectrum Computation

The *Short-time Fourier Transform* (STFT) is used to compute an amplitude and phase spectrum for each analysis frame. We obtain the amplitude and phase of the sound's partials by calculating the local maxima in the magnitude spectrum and reading the corresponding phases from the phase spectrum.[4]

The STFT works by applying a sliding *window function* to the signal. It smoothly pushes the signal to silence towards the window's edges. This way, the Fast Fourier Transform can be used to compute the spectrum of *only the portion of the signal that was not pushed to zero*. See Figure 3 for an illustration. The spectrum is complex-valued, and can be interpreted as a

[3]We cannot just use the calculated from the note label because of tuning inaccuracies

[4]For gathering our timbre vectors, we would only need to consider the magnitude spectrum. However, for the phase-correct resynthesis in section 3.2.6, the signal's phases are needed as well.

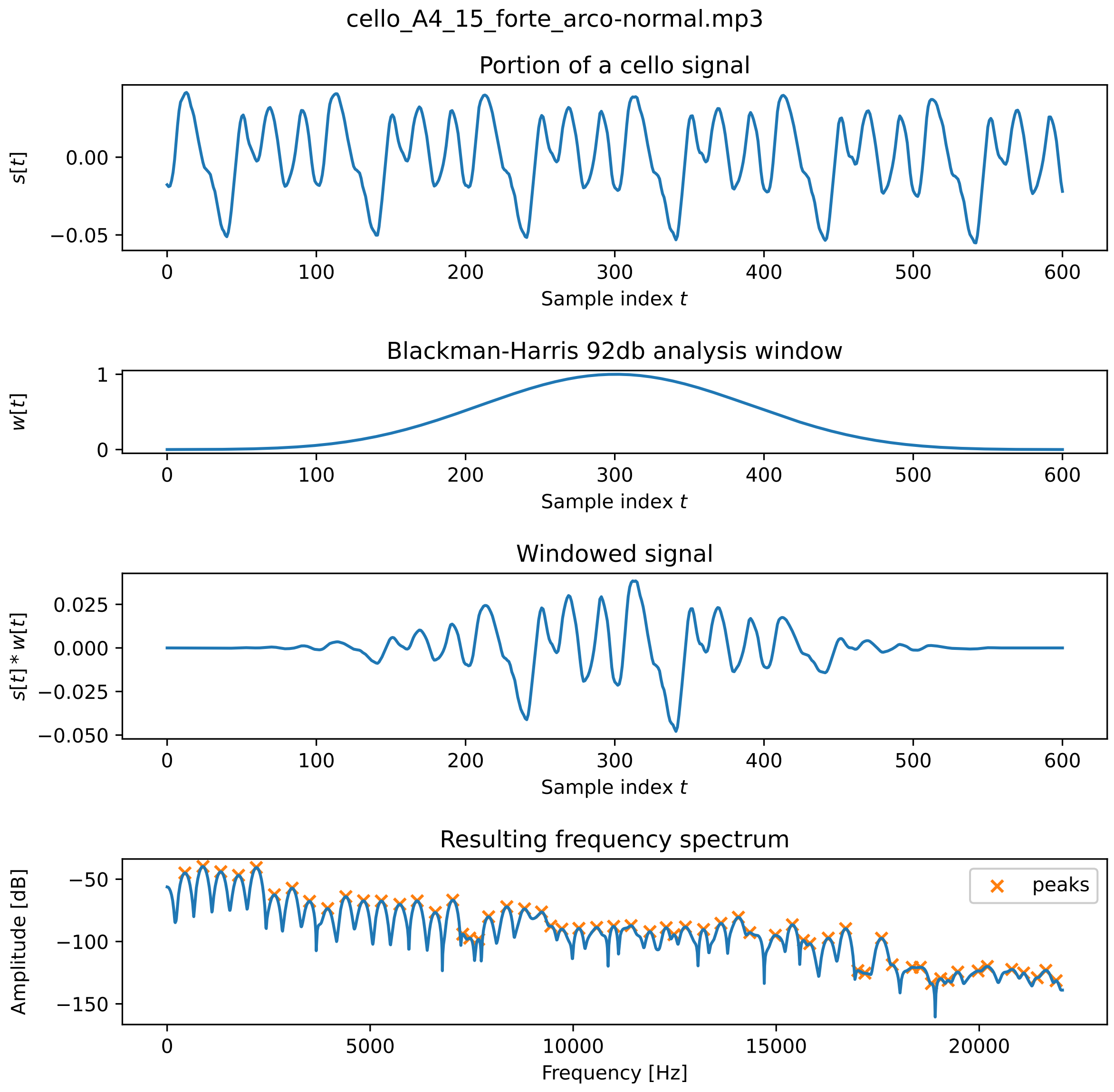

Figure 3: A typical spectrum analysis frame of a cello signal. Figure adapted from Serra (1997, p. 6).

magnitude- and a phase spectrum. The STFT has several analysis parameters (type of window function, hop size, window size, signal padding). For the sake of brevity, we will not discuss the choice of parameters. Refer to (Serra 1997, pp. 5ff.) and (Harris 1978) for a more detailed discussion.

3.2.3 Peak detection

Using the magnitude spectra from the previous step, peaks are detected by finding locally maximal amplitudes. Physical audio recordings usually contain some amount of white noise generated by electrical interference and other physical phenomena. This noise is visible in the magnitude spectrum as a *noise floor*. This is relevant to us because the noise floor will drown out very quiet parts of the spectrum. Essentially, it acts as a lower bound below which we cannot detect any spectral peaks. If we detect peaks in the vicinity of the noise floor, we cannot be entirely sure if they belong to the deterministic component of our signal. Although we perform filtering later on, many false positives can be avoided by only considering peaks 6dB over the noise floor (i.e. approximately double the amplitude). The samples in our dataset are not all recorded with

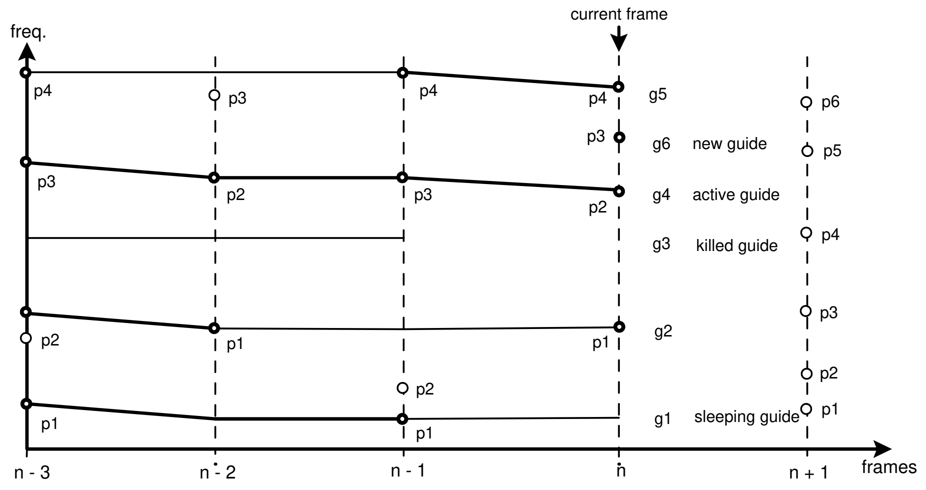

Figure 4: Illustration of tracking peak matching. Extracted from Serra (1997, p. 11). Used with permission.

a consistent loudness, leading to inconsistent signal to noise ratios and differing levels of the noise floor. However, since peaks show up as narrow spikes in the magnitude spectrum (before scaling to decibels), we can just compute the median magnitude to get a rough estimate of the noise floor in each analysis frame.

The accuracy of the peak frequencies depends on the resolution of the spectrum. After having detected the peaks, Smith and Serra (1987, pp. 10f) propose an interpolation scheme to improve the estimate for the peak frequencies. When scaling the magnitude spectrum to decibels, peaks look approximately like parabolas. Hence, by fitting a parabola to the magnitude peak and its two neighboring data points, we can get an improved estimate for the signal amplitude A and frequency ω. The same interpolation coefficient is used to linearly interpolate the phase φ.

3.2.4 Peak matching

Not all peaks detected in the previous step belong to the deterministic component of the signal. Peaks of the deterministic component move slowly with time, i.e. we can expect that similiar peaks will exist in a large number of consecutive frames, allowing us to *track* the peak's trajectories over time. Furthermore, as discussed in section 3.1, we expect that we only need to consider frequencies that are integer multiples of the signal's fundamental frequency.

SIMPLE peak matching For our analysis, we do not perform tracking. We just use those peaks that are sufficiently close to our *expected overtone frequencies* and assemble these into our constant-dimensional vectors. If a peak for an expected frequency is not found, the vector component is left blank. Peaks that do not match our expectation stay unused at this stage. After the spectral analysis is complete, a cleanup operation: We remove peaks that only exist for few consecutive frames (we consider these as false positives that do not belong to the deterministic component) and interpolate when a partial drops out for a few frames. This strategy is very simple in principle, however we need to make sure that it does not miss a deterministic vibration that is not part of the harmonic series. Also, it will inevitably degrade if we get an unreliable estimate of the fundamental frequency. To validate our assumptions, we also implemented a tracking peak matcher as proposed by Serra:

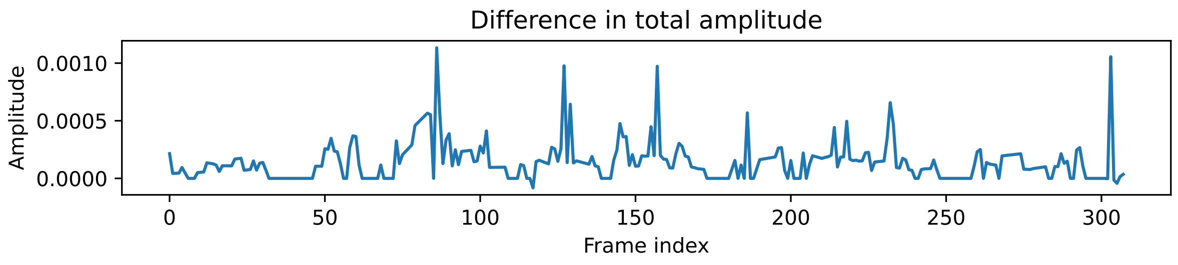

Figure 5: Difference of total amplitude of the two peak matching strategies: (SIMPLE - TRACKING).

TRACKING peak matching The *tracking peak matcher* (Serra 1997, p. 10f.) manages a fixed number of "guides" that follow the trajectory of each peak over the course of the analysis. Guides may turn on and off at any time. When a new set of peaks comes in, the guides will consume them appropriately: From loudest to quitest, each guide consumes the peak whose frequency is closest to its frequency in the previous frame. If no suitable frequency is found, the guide is turned off. After all guides have been continued or turned off, new guides are created until the maximum number of guides is reached, consuming the loudest peaks that have not been consumed yet. As before, the trajectories are cleaned up by removing short ones and interpolating over small holes. Figure 4 provides an illustration. This method is more flexible and might be useful for analyzing nonharmonic sounds. However, it bears a lot more complexity.

As we expected, we cannot find a qualitative difference between the the results of the two peak matching procedures. Looking at Figure 5, The difference in amplitude extracted from the signal is at least one order of magnitude smaller than the original signal's amplitude. We conclude that the SIMPLE peak matcher does not lose significant parts of the signal. Our assumption that we only need to consider integer multiples of the signal's pitch as candidates for the deterministic component was correct. Hence, we use the SIMPLE peak matcher. When the analysis of an audio sample is complete and the cleanup has been performed, the SIMPLE peak matcher will have gathered timbre vectors $\mathbf{x}[n] = [A_1[n], A_2[n], \dots A_D[n]]^T$ for each analysis frame. They will form the basis on which we train the Generative Models in the next chapter.

3.2.5 Visualizing the Timbre Representation

Timbre vectors can be visualized in two different ways: *Plotting the detected peak frequencies vs. time.* Color can be used to denote the decibel-scaled amplitude of the peak. Similiar in spirit to an STFT spectrogram (Weinzierl 2014, p. 18f.), this plot can give a holistic view of the structure of the deterministic component we extracted, including vibrato, jumping partials, missing values, etc. It is also useful to judge the overall performance of the analysis and to tune analysis parameters to the data at hand. However, finer structures in the peak amplitudes cannot be identified visually. *Only plotting the detected peak amplitudes vs time* (or a subset of them), their behaviour with time can be observed more precisely. See Figure 6 for exemplary plots. We can recognize the cello's vibrato and the relatively consistent distribution of amplitudes over time, and the rapid decay of the guitar's overtones.

Before moving on, let us briefly touch on the extraction of the deterministic component:

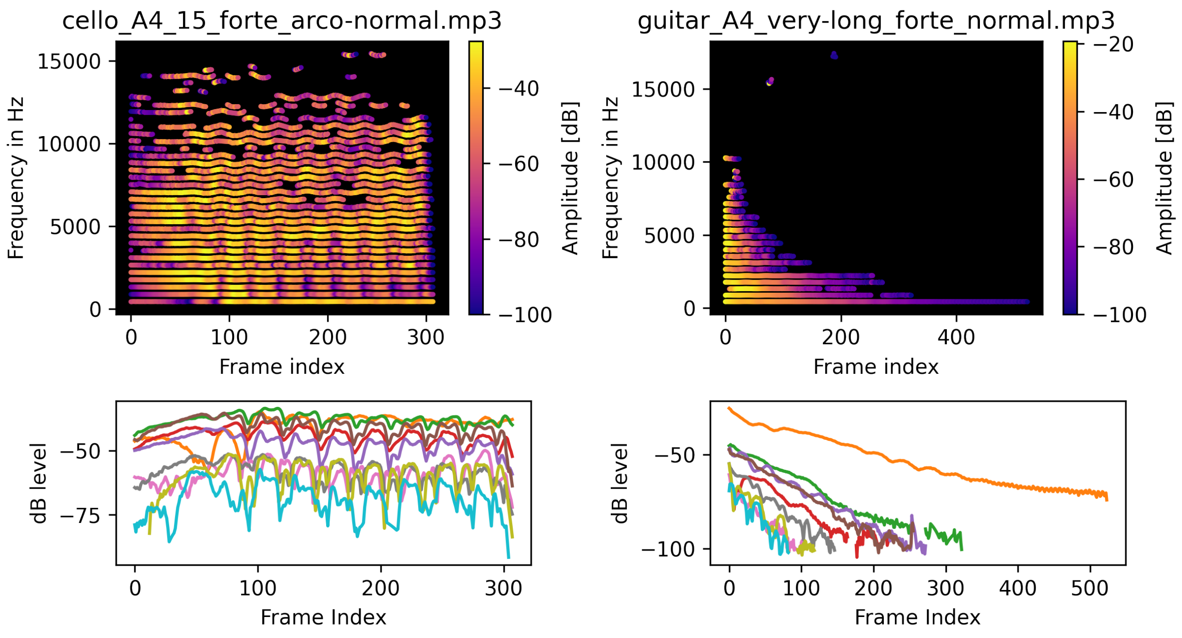

Figure 6: Two visualizations of the timbre vectors gathered for a typical cello sample (left) and guitar sample (right). The top plot shows the peak frequencies over time, color indicating peak amplitudes. The bottom plot shows the behaviour of the amplitudes of the guitar's first ten overtones.

3.2.6 Phase-correct resynthesis, Evaluation, Noise subtraction

To extract the stochastic component $e[t]$ of the signal, we need to resynthesize the deterministic component in a *phase-correct* fashion and subtract it from the original signal. We do this using an oscillator-bank technique as described by McAulay and Quatieri (1986): Each oscillator's instantaneous amplitude is determined by linearly interpolating between the amplitudes in the analysis frames $A_i[n], A_i[n+1]$. Interpolating between the phases is a little more difficult, since phases are 2π-periodic. We only know the phases $\varphi_i[n]$ and $\varphi_i[n+1]$, but it is not clear if or how often the phase should wrap around 2π between the analysis frames:

$$\varphi_i[n] + 2\pi k + \Delta\varphi_i = \varphi_i[n+1] \mod 2\pi$$

McAulay and Quatieri (1986) derive a set of formulas[5] that optimize k so that the oscillator's phase trajectory is "maximally smooth". With that, the deterministic component $\hat{s}[t]$ can be resynthesized in a phase-correct way. The stochastic residue $e[t]$ is obtained by simply subtracting the deterministic synthesis from the original signal. Due to time constraints, I did not perform any more analyses on the noise component.

Having obtained a synthesized signal for the deterministic component, we can perform a sanity check: The stochastic residue will be relatively quiet compared to the deterministic component. Hence, by overlaying the resynthesis and the original signal, we can validate that our analysis actually worked (see Figure 7). Numerical evaluation is possible as well: For a signal of length T, let $\overline{s} = \frac{1}{T}\sum_{t=1}^{T} s[t]$. Using the *coefficient of determination* R^2 (Pedregosa et al. 2022,

[5]Serra cites McAulay and Quatieri's formulas. Note that there is a sign error in Serra's paper: Compare the calculation of ι in (Serra 1997, p. 14) to the equivalent formula for $\beta(M)$ in (McAulay and Quatieri 1986, p. 70, formula 34).

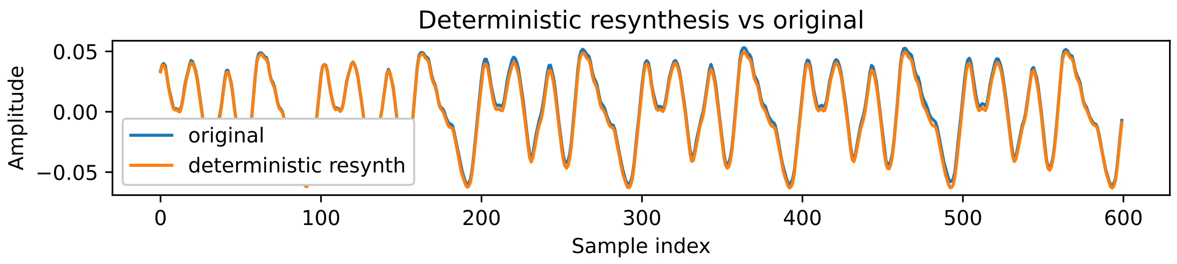

Figure 7: Phase-correct resynthesis from timbre vectors. (Compare to Figure 3)

Section 3.3.4.1), we get an estimate of the proportion of energy extracted from the signal:

$$R^2 = 1 - \frac{\sum_{t=1}^{T}(s[t] - \hat{s}[t])^2}{\sum_{t=1}^{T}(s[t] - \overline{s})^2}$$

This measure will range from negative infinity to 1, greater scores indicating that more energy was extracted. This is by no means a precise measure of the quality of extraction, but it can alert us when the analysis fails somehow and less energy than expected is extracted. In the next section, we evaluate the results of our procedure on the cello and guitar datasets:

3.3 Results

The analysis procedure outputs three audio signals: The original signal $s[t]$, normalized and clipped to note boundaries, the deterministic resynthesis $\hat{s}[t]$, and the stochastic residue $e[t]$. In general, we expect that our procedure separates the signal into a deterministic and stochastic component, as discussed in section 3.1.2. Because we normalize the source signal before the analysis, the recordings and synthesized signals will have the same loudness, regardless of the loudness of the audio source. However, quieter samples will have their noise floor more strongly elevated, which we expect to be noticeable in our results.

Let us first look at the analysis results of the cello. Listening to 🔊 2 (C♯3 ***pp***), our impression is that the cello is clearly recognizable in the resynthesis of the deterministic component. Overall, the deterministic component sounds a little more dull to us than the original signal. The stochastic residue contains a scratching noise which we recognize as the scratching sound created by the bow. In addition, we also notice a high-pitched "shimmering" noise. Let us compare these results to the same note note played fortissimo: 🔊 3 (C♯3 ***ff***). In the source file, the rate of the cello's vibrato is faster, which we also notice in the deterministic resynthesis. Furthermore, the scratching sound in the stochastic residue seems stronger than the pianissimo sample. We can observe some other phenomena: In 🔊 4 (E4 ***mp***), the shimmering noise seems more pronounced to us, and in 🔊 5 (A4 ***f***) it appears as if there is still some part of the original pitch audible in the stochastic residue. We believe that the shimmering noises originate from peaks that were misclassified in the peak matching step and made it through the cleanup operation (see section 3.2.4). We call these "false positive" and "false negative" peaks.[6]

The median R^2 of ≈ 0.991 confirms that our analysis is reliable (see Table 1 for a full reference). Notice that small errors do not find expression in the R^2 score: the separation of 🔊 5 (A4 ***f***) was sub-par, yet with $R^2 \approx 0.992$ it still achieved above-median scoring. For the lowest few notes in

[6]False negative peaks are peaks belonging to the deterministic component and should have been extracted from the source signal but are wrongly left in the stochastic residue. False positive peaks are peaks that are attributed to a deterministic oscillation that does not exist in the source signal. Because the oscillation will exist in the resynthesis, and $e[t] = s[t] - \hat{s}[t]$, it is noticeable in the stochastic residue with a flipped sign.

Table 1: R^2 scores of the deterministic synthesis, in quantiles.

	$q = 0$ (min)	$q = 0.25$	$q = 0.5$ (median)	$q = 0.75$	$q = 1$ (max)
Cello	0.000184	0.978829	0.991228	0.996171	0.999061
Cello > D2	0.794196	0.983347	0.992441	0.996351	0.999061
Guitar	0.198769	0.549556	0.761197	0.865428	0.958670
Guitar ***p***	0.198769	0.446254	0.606756	0.751304	0.928321
Guitar ***f***	0.268161	0.772095	0.856637	0.900786	0.958670

the dataset (C2, C♯2, D2), we observe that the analysis failed completely, achieving R^2 scores lower than 0.4. We have not yet found an explanation. However, the rest of the samples seems unaffected, as can be seen when we exclude the degraded samples from the R^2 statistics. In the following, we will exclude all cello samples with $R^2 < 0.5$.

Let us move on to the guitar. Listen to 🔊 6 (A♯4 ***f***): Again, to us the deterministic resynthesis sounds very close to the original signal. With the cello, we observed that the resynthesis sounded muffled compared to the original. In our impression, this is not the case for the guitar. The stochastic residue contains a noise that we identify as the plucking and oscillation of the string. We notice the same shimmering noise we found in the cello recordings, however its pitch seems to follow a downward motion. Over time, the guitar's partials decrease in amplitude, and typically drop out from highest to lowest frequency (see Figure 6). Just before their dropout, the distinction between noise floor and deterministic peaks is especially difficult. This supports the thesis that the shimmering originates from "false positive / false negative" peaks. Listen to 🔊 7 (D♯3 ***p***) - this sample has a higher noise floor. To us, the shimmering noise we noticed in the other sample's stochastic residue is similiarly noticeable in both the stochastic residue and the deterministic resynthesis. This might point to a larger classification error: The shimmering is so strong that it cannot be drowned out by the rather quiet deterministic signal.

Table 1 shows that the deterministic / stochastic separation for the guitar performs worse than for the cello. We attribute this to the rapid decay of the guitar's overtones, that demands a precise separation between deterministic peaks and the noise floor, which we might not achieve with the SIMPLE peak matcher. Also, the overall worse signal to noise ratio in the guitar samples might be detrimental to our results: The dataset only contains *piano* and *forte* samples. Notice in Table 1 that the score slightly improves when only considering *forte* (***f***) samples, which should tend to have a better signal to noise ratio than *piano* (***p***) samples. In our opinion, the timbre vectors still lead to a decent approximation of the guitar sounds.

Let us summarize: The timbre vectors we gathered appear to be reliable enough to resynthesize a signal that is, in our impression, close to the source signal. They seem to capture most of the deterministic vibrations, as we find little or no deterministic content left in the residue signal. The shimmering noise seems to originate from peaks that are very close to the noise floor and wrongly end up in either the deterministic or the stochastic component. To improve the analysis, we might use a more advanced method of estimating the noise floor. Alternatively, we could employ a more robust peak matching strategy such as the TRACKING peak matcher that does not need noise filtering. For our purposes however, we think that our simpler procedure is sufficient. Concerning the cello, it is unfortunate that the resynthesis sounds more dull than the original. However, since we did not find deterministic content left in the stochastic residue, it appears that the more brilliant part of the cello's timbre is attributed solely to the scratching noise of the bow. Leaving out the stochastic component might be an over-simplification in this case.

4 PCA on the Timbre Dataset

Having gathered a timbre dataset, we would now like to find an instrument model that is able to generate a continuum of timbres from a set of control parameters. Particularly, we would like to find control parameters that allow us to move within the continuum in a musical fashion. The model should be *intuitive to control*. If possible, we would like to be able to *exaggerate* the control parameters beyond the realm found in the original dataset, to produce musically interesting sounds. To that end, we explore using *Principal Component Analysis* (PCA) on our dataset.

PCA is an unsupervised method that finds the "axes" in a dataset that explain most of its variance. These axes are called *Principal Components*. Figure 8 provides an illustration. Applying PCA to our timbre dataset, we would like to find axes in the timbre continuum.

In the following, we describe the PCA algorithm itself. PCA exploits the linear relationships betweeen coordinates in a dataset. It is not intrinsically robust against outliers and cannot deal with missing data. For that reason, we motivate some further preprocessing techniques needed to apply PCA successfully. Finally, we evaluate the results of our experiments.

4.1 The PCA Algorithm

Bishop (2007, p. 561ff.) formulates PCA as a variance-driven method for dimensionality reduction: PCA finds the axes in a high-dimensional dataset that best explain its variance by calculating the eigenvector decomposition of its covariance matrix. Assuming the latent space has a lower dimensionality $M < D$, we can use only the M eigenvectors that explain the largest share of variance to form a basis of the latent space. By projecting onto that space we get a lower-dimensional approximation of the data. The calculation is as follows:

Let $\mathbf{X} = (\mathbf{x}_n)_{n \in \{1,\ldots N\}}$, $\mathbf{x}_n \in \mathbb{R}^D$ be a D-dimensional dataset in Euclidean space with sample mean and covariance matrix

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n, \quad \mathbf{C} = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$$

We calculate the eigenvectors $\mathbf{u}_i$ of $\mathbf{C}$ by solving $\mathbf{C}\mathbf{u}_i = \lambda_i \mathbf{u}_i$ for $\mathbf{u}_i$. Since $\mathbf{C}$ is a covariance

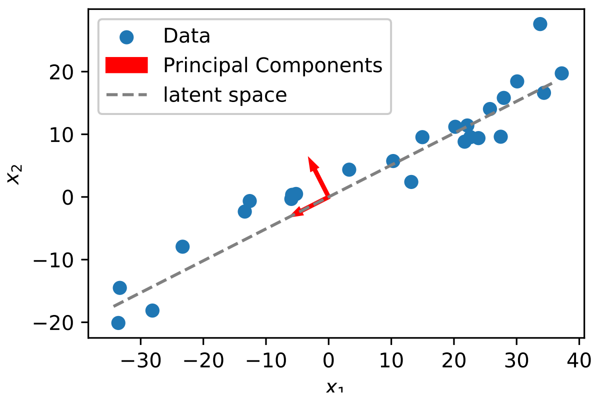

Figure 8: PCA on an artificial 2D dataset. We can reduce the dimension by one by projecting only on one of the principal components.

matrix, it is symmetric and positive semidefinite. Hence, the eigenvectors will be pairwise orthogonal. Assuming we are not dealing with a degenerate case where the geometric multiplicity of an eigenvalue is greater than one, we will get D pairwise orthogonal eigenvectors that can be used as a new basis. Performing a basis change, we express each sample $\mathbf{x}_n$ as a linear combination of the eigenvectors. The eigenvalues λ_i represent the variance explained by the principal component u_i. By sorting the eigenvalues from largest to smallest, we can use only the M most significant eigenvectors to get a simpler approximation for the data:

$$\mathbf{x}_n = \overline{\mathbf{x}} + \sum_{i=1}^{D} (\mathbf{x}_n - \overline{\mathbf{x}})^T \mathbf{u}_i =: \overline{\mathbf{x}} + \sum_{i=1}^{D} \alpha_{ni} \mathbf{u}_i \approx \overline{\mathbf{x}} + \sum_{i=1}^{M} \alpha_{ni} \mathbf{u}_i$$

We then interpret $\boldsymbol{\alpha}_n = [\alpha_{n1}, \ldots, \alpha_{nM}]^T \in \mathbb{R}^M$ as vectors in the latent M-dimensional space.

In the context of our analysis, we use PCA to reduce high-dimensional timbre vectors to a simpler linear combination of eigenvectors $\sum_i \alpha_{ni} \mathbf{u}_i$. We will interpret the eigenvectors $\mathbf{u}_i$ as *axes* along which we can move through the timbre continuum, and and α_{ni} as the *control parameters* required to resynthesize an approximation of the source signal. Looking at the trajectory of the control parameters over the course of one sample will give us some insight as to the semantics of each eigenvector.

For building electronic musical instruments, we are mostly interested in the eigenvectors $\mathbf{u}_i$ and less in the actual dimensionality-reduced data vectors $\boldsymbol{\alpha}_n$. By using artificial control parameters $\tilde{\alpha}_i$ chosen instantaneously by the musician, we can calculate new timbres and resynthesize them using additive synthesis, as explained in the previous chapter.

4.2 Preprocessing

Some further preprocessing is required to transform our dataset from chapter 3. As we have seen in the previous chapter, PCA assumes the data contains no holes, and it makes no attempt to be robust against outliers. In the following, we illustrate some techniques to adapt our data so that it is suitable for PCA:

4.2.1 Decibel Scaling

Amplitudes are *positive semi-definite*. Physically, the amplitude of an oscillation denotes its maximum positive deflection (Tipler and Mosca 2015, p. 415). A sinusoid with a negative amplitude, though mathematically possible, does not make physical sense because it is equivalent to a positive-amplitude sinusoid shifted by π: $-A \sin(\omega t) = A \sin(\omega t + \pi)$. And since absolute phase is not relevant in the human perception of sounds (Serra 1997, p. 17), the sinusoid with negative amplitude will be perceived the same as the sinusoid with positive amplitude.

This is problematic because PCA is not intrinsically constrained to the positive realm. For some sets of control parameters, the model will yield negative amplitudes, making it less intuitive to control. We get especially undesirable results if we exagerrate a control parameter α_i beyond its original realm: If $\alpha_i \to +\infty \Rightarrow A_j \to +\infty$, then $\alpha_i \to -\infty \Rightarrow A_j \to -\infty$. Both extremes are however perceived the same, which might be musically undesirable. A solution might be to constrain the model to only those combinations of control parameters that yield positive amplitudes, or to clip negative amplitudes to zero. Both solutions are however detrimental to the intuitiveness of our model: For the former, it is unclear what should happen if the musician selects an “invalid” combination of control parameters. For the latter, clipping some parts of the signal to silence while exaggerating others might yield unpleasant or unexpected results.

Instead, it might be more practical to scale the data to decibels before applying PCA. ($A_{dB} = 10\log_{10} A \ \text{dB}$) Since "$\log -\infty = 0, \log +\infty = \infty$", we can constrain our model to operate purely in the realm of positive amplitudes. Decibel scaling has another advantage: the perceived loudness of a signal does not scale linearly with its amplitude. By applying decibel scaling we get a quantity that is better correlated with human perception of loudness (Weinzierl 2014, p. 14).

4.2.2 Dealing with Missing Values

Overtones for which no peaks could be detected will appear as holes in the timbre vectors. Cello and guitar datasets are both affected by dropouts of partials, however the effect is more extreme for the guitar, where instead of temporary dropouts, partials vanish completely due to the decay of the signal. For higher partials, we might have as little as a handful data points at the beginning of the sample. PCA cannot deal with missing values, so we need to employ some imputation strategy. Simple approaches could be to fill the holes with the estimated mean *amplitude of the noise floor*, or to calculate the *mean amplitude for each partial* and fill the holes accordingly. However, since vanilla PCA is not very robust against outliers, these strategies will heavily skew the axes found by PCA. Instead, we propose two strategies specific to the instrument that is analyzed:

4.2.3 Cello: Filtering Timbres

Looking back to Figure 6, the cello has a more consistent distribution of overtones than the guitar. To get a dataset free of holes, we can first of all use linear interpolation. However, since the amplitudes oscillate rapidly, we cannot sensibly interpolate over more than ten frames without losing important detail. Instead, after interpolating over holes up to ten frames long, we apply a filtering technique: We consider those partials for which a value is present at least 80% of the time as "reliable" partials. Unreliable partials are silienced by filling them with a constant value of $-240dB$. This way, they are inaudible and have zero variance, and thus will not influence the PCA. Looking at Figure 6, unreliable overtones will typically be in the upper third of the frequency spectrum, where the differentiation between noise and deterministic component is difficult anyway. Subsequently, all frames where a reliable overtone is missing are dropped, leaving us with data free of holes.

4.2.4 Guitar: Sub-Noise Imputation

Let us look at Figure 6 again. We can see that for a typical guitar sample, the amplitudes tend to descend with time in a regular fashion. Our goal is to find a reasonable extrapolation below the noise floor. Since the extrapolation below the noise floor is a blind guess, we should prefer the simplest possible model: We extrapolate each feature using a linear regression as a function of time. (More precisely, a Ridge regression to get a little more robustness). We consider features with fewer than ten data points too unstable for regression. Again, we remove them from the analysis by filling the features with silence ($-240dB$). Figure 9 shows the results of the imputation.

We also considered two other approaches: `sklearn.impute.IterativeImputer` implements a more complex imputation scheme that imputes missing data iteratively: For each feature, an arbitrary regressor is fit *using the other features as inputs*. This step is repeated until the maximum number of iterations is reached or the imputation converges. The advantage of this approach is that the imputed features do not need to be linear with respect to time, so we might

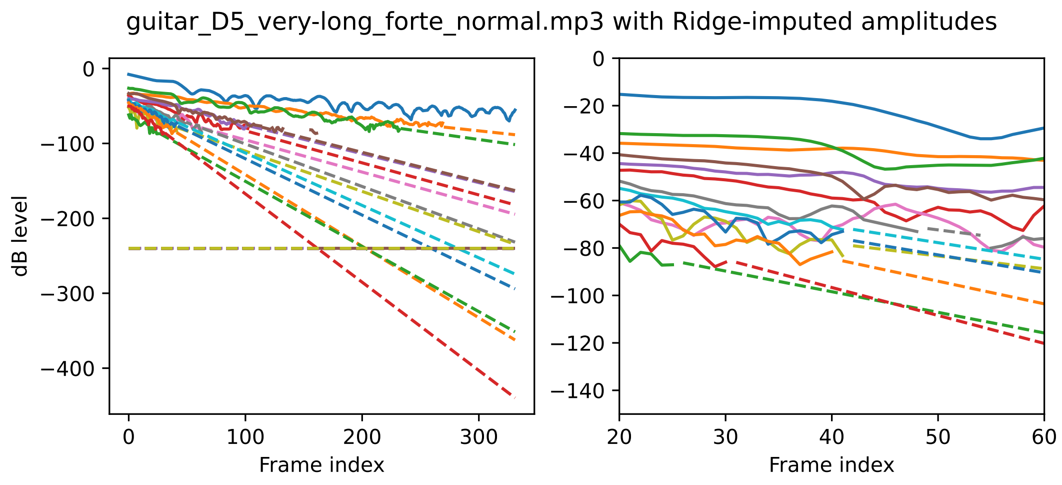

Figure 9: Extrapolation of guitar overtones below their lower bound. Dashed lines are the imputed parts. Some partials are silenced completely (-240dB) because the do not contain enough data points, explaining the horizontal line in the left plot. The right plot is a zoomed view of the left plot.

be able to express more detail. However, this approach is both computationally expensive and unstable - we did not achieve an imputation that converged reliably for all inputs.

Lastly, we tried slicing the signal into multiple parts and applying the imputation and PCA separately. This way, we do not need to impute as many values: We do not need to impute a partial over the course of the entire sample, but just to the border of the slice, after which we can consider it empty. Indeed, this approach stabilized the analyses. However, it seems flawed, because it is not clear what to do with the multiple PCA results: In the end, we need a unified instrument model, if not for the whole instrument, then at least to play a single note. However, it seems impractical to switch out the different eigenvectors $\mathbf{u}_i$ in the middle of playing a note, or to interpolate between them.

4.3 Evaluation Criteria

Since PCA is an unsupervised method, we need some way to judge how well the decomposition worked. Four factors come to mind:

- *Performance of dimensionality reduction:* To build a musical instrument that is intuitive to control, we would like to obtain a model that explains the most amount of variance with the smallest number of axes necessary.
- *Describability:* We can look at the trajectories of the control parameters over the course of the sample. A control parameter might behave in any number of ways; as long as its behaviour is describable in a simple way (i.e. steady movement, random jittering, sine-like oscillation), or can be motivated by the characteristics of the instrument being analyzed, it will likely be musically useful. The only behaviour that is undesirable are sudden jumps, as they are likely to point to an outlier in the dataset that might skew the the whole decomposition.
- *Independence:* Furthermore, we can look at combinations of the most significant control parameters. Ideally, we would like each control parameter to be responsible for one and only one kind of movement in the timbre continuum - we would like the control pa-

rameters to be independent. PCA guarantees uncorellatedness, but does not guarantee independence.

- *Stability*: Ideally, a control parameter should always perform the same function, regardless of the audio sample being analyzed. The eigenvectors extracted by the PCA analysis are sorted by their explained variance. If two eigenvectors usually explain a similiar amount of variance, it will be difficult to identify which one belongs to which function just based on their index. We might have to perform some clustering.

4.4 PCA Results

Using the preprocessing techniques discussed above, we can now apply PCA to our data. We tried applying PCA to the timbres of a single sample, as well as on a whole set of timbres. In the following, we will discuss the numerical results and plots. Also, get a sense of the function of the principal components learned by PCA, we resynthesize the timbres along each eigenvector:

We illustrate an principal component's function by synthesizing a signal from an artificial timbre that moves in a sinusoidal pattern along the eigenvector:

$$\tilde{\mathbf{x}}[t] = \bar{\mathbf{x}} + a\sigma_i \sin(2\pi f t) \cdot \mathbf{u}_i$$

f is a low frequency (0.5 Hz in our case), σ_i is the standard deviation of the ith control parameter, and a is a scaling factor. We use $a = 4$ to exaggerate the timbre range.

Let us now begin with the analysis of single audio samples:

4.4.1 Single-Sample PCA

Since applying PCA to each audio sample separately leads to as many PCA models as there are audio samples, we need to concentrate on a few examples. We tried to pick samples that seemed typical.[7] Let us start with the cello:

Cello Let us look at the analysis of the G3 fortissimo sample (🔊 8), depicted in the first set of plots in Figure 10: We can see that the explained variance is rather spread out, with less than 30% of the variance explained by the most significant eigenvector. This points to a *low-quality dimensionality reduction*. To get a model that can reproduce the same variability in timbre as the source sample, we would need to use a large number of control parameters, which is undesirable because it makes our model more difficult to control by a musician. Looking at the trajectory of the control parameters, we can observe an oscillation which looks approximately sinusoidal. The oscillation is especially regular for the first two control parameters. Notice that $\alpha_1[n]$ seems to be almost a mirror image of $\alpha_0[n]$. We reckon that the first three control parameters are *not completely independent*. Let us listen to the resynthesis: 🔊 9. The three signals sound similiar to us - in the extreme points of the oscillation, the same parts of the spectrum seem to stand out, regardless of the principal component being looked at. Our conclusion is that the three principal components we looked at are *describable*, because their trajectory in the original sample can be described by a simple oscillation, but they seem to have approximately the same function. This leaves more to be desired.

[7] All analysis results can be generated using the notebooks in the supplimentary material.

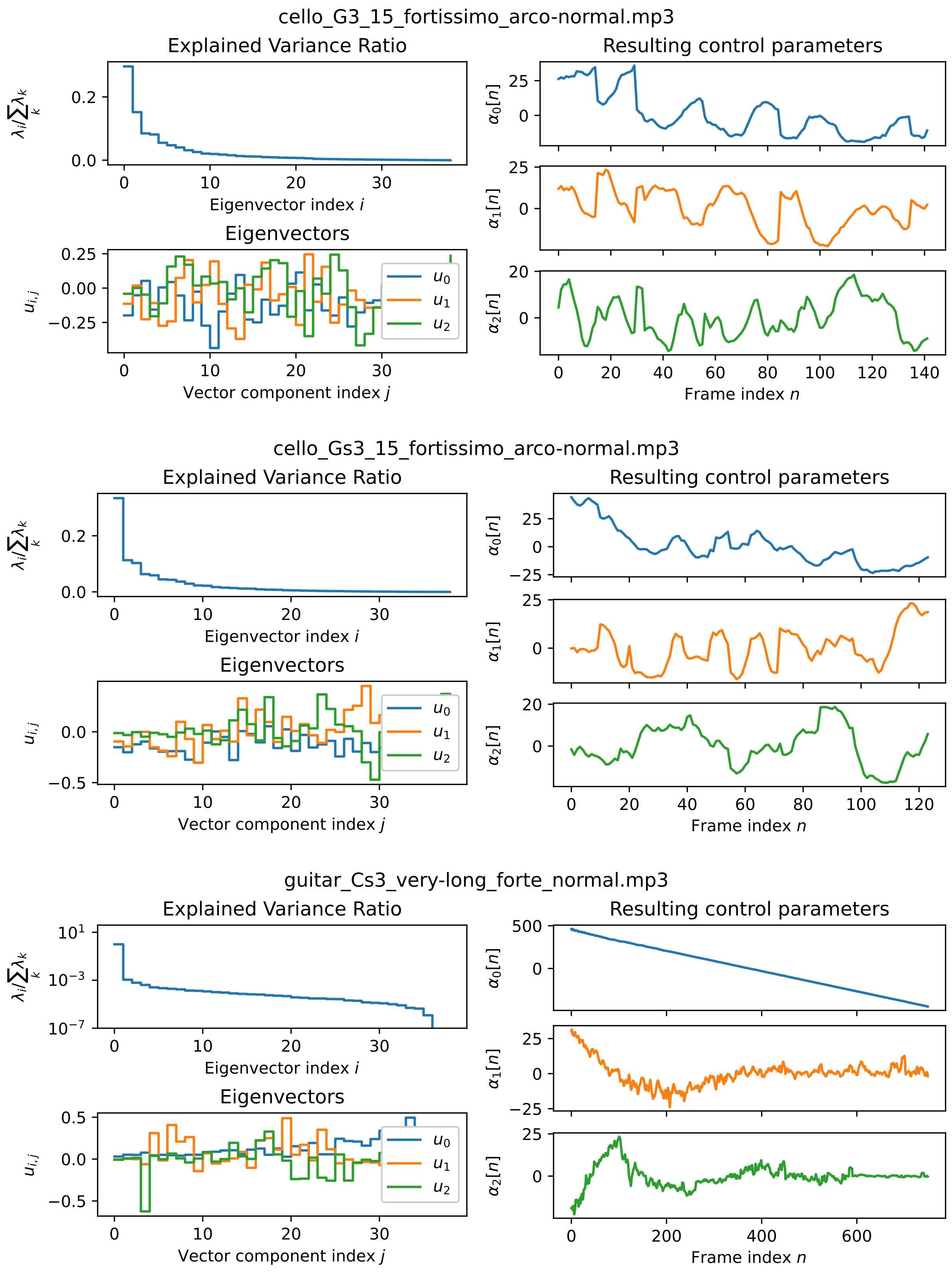

Figure 10: Results of PCA applied to typical single audio samples. For each sample, the variance explained by each eigenvector is shown as a proportion of the total variance (top left plots). The three most significant eigenvectors are shown (bottom left plots). Finally, their corresponding control parameters are plotted (right-hand side plots)

To judge the stability of the analysis, we consider the sample one semitone above, G♯3 fortissimo (🔊 10). In our opinion, it sounds very similiar[8] to the G3 sample (🔊 8), so we should be able to expect similiar PCA results. In fact, the trajectory of the control parameters is similiar to the G3 sample, as shown in the middle set of plots in Figure 10. However, looking at the contents of the most significant eigenvector u_0, we can observe that the first eleven coefficients remain entirely negative, while the corresponding coefficients for the G3 sample are both positive and negative. Similiarly, the first eleven coefficients of u_2 are close to zero for the G♯3 sample, while they are much more spread out for the G3 sample. Finally, let us listen to the artificial timbres created by the eigenvectors of the G♯3 sample (🔊 11, synthesized with a pitch of G3 for easier comparison with 🔊 9). Our impression is that G♯3's principal components emphasize a narrow band of the spectrum, similiar to G3's principal components. However, the equilibrium of the oscillation, i.e. the mean timbre $\bar{\mathbf{x}}$ sounds quite different compared to G3. For now, we conclude that the analysis is *not stable enough* to reliably produce simliar timbres from similiar samples.

Guitar For the guitar, the PCA results look different: We will use the C♯3 forte (🔊 12) sample as an example. Looking at the bottom set of plots in Figure 10, the explained variance is strongly concentrated on the 0th principal component (notice the logarithmic scaling in the Explained Variance plot). We notice that the coefficients of the eigenvector u_0 are entirely positive. The value of the corresponding control parameter $\alpha_0[t]$ decreases almost linearly with time. We interpret that u_0 is responsible for the dampening of the guitar sound. Listening to the resynthesis 🔊 13 confirms this interpretation: u_0 pulls the signal to silence. The effect of u_1 is less dramatic in our impression. As for u_2, its $j = 4$th component stands out in the eigenvector plot, and is, in our impression, clearly noticeable in the synthesized signal.

Looking beyond our exemplary sample, we can notice a large number of "degraded" samples, where for some components j no eigenvector explains any variance. Remember that in our preprocessing routine (section 4.2.4) we filled holes in the timbre vectors using a linear regression with respect to time. Overtones that contained less than 10 values over the course of the sample were silenced. Hence, they will not contain any variance that could be modeled, so the corresponding eigenvector components will be zero. This entails that samples with unstable peaks will likely be unusable; the single-sample guitar anlysis is not stable.

Dimensionality reduction works well, and the 0th principal component can even be motivated with the mechanics of the instrument at hand. However we might say that dimensionality reduction performs *too well*: If we can reduce the whole evolution of the timbre onto only one eigenvector, that means that the timbre evolution itself is not very complex, potentially leading to a model that is not musically interesting.

To mitigate the problems discussed above, we tried training a PCA model using multiple samples:

4.4.2 Ensemble PCA

Analyzing all samples with a single PCA model has a number of advantages: When analyzing multiple cello samples, we observed that their mean timbre was inconsistent. We sidestep this problem by only computing a single mean timbre. Also, we observed that the imputation process might fail, which would be fatal for single-sample PCA. By analyzing all samples at the same

[8]Note that judging the dissimiliarity of these types of samples is very subjective and may depend on numerous variables such as the type of loudspeaker being used for playback, the physical condition of the listener, environmental noises, etc.

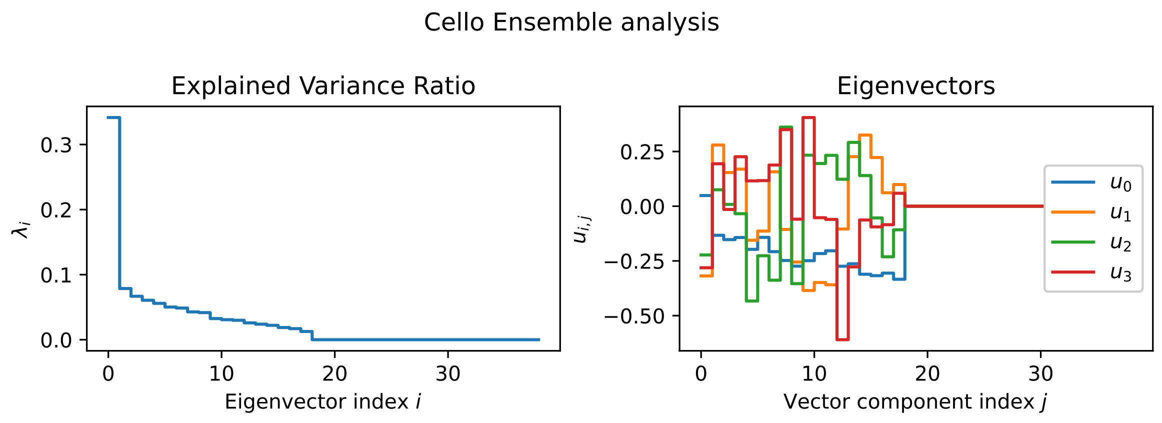

Figure 11: Explained Variance and Eigenvectors for the cello ensemble analysis. Behaviour of the control parameters is not shown as they seem to behave similar to single-sample PCA.

time, we can afford to lose a little more data and exclude the samples for which the preprocessing failed.

In preprocessing, we need to be careful about whether or not to preprocess samples together, or each one separately: The cello timbres all need to be processed together. This is because when we perform Timbre Filtering (section 4.2.3), we decide which overtones to consider for analysis. If we performed this step for each audio sample separately, we might get overtones that are active for in some audio samples, and inactive in others - essentially, we would end up with the same problem as we had with the guitar in the previous section. On the other hand, the Guitar's Sub-Noise Imputation (section 4.2.4) needs to be done for each sample on its own. Since we exploit the linear movement of the overtone amplitudes with time, we cannot simply concatenate all samples and feed them into the same regression. Instead, we need to run a regression for each sample separately. Having discussed preprocessing details, we will now outline the results of the ensemble PCA:

Cello Figure 11 shows the results of applying PCA to the whole Cello timbre dataset. We can see that the concentration of variance has improved slightly - the first principal component now captures a little more variance than in single-sample PCA. Also notice that u_0 now has an interesting characteristic: Its first component is positive, while all other components are negative. Listening to 🔊 14, we can observe that the perceptual difference between the principal components is greater than for single-sample PCA (compare to 🔊 9): Our impression is that u_0 influences the timbre to be more "aggressive", while moving along u_1 produces a more muffled sound. u_2 and u_3's influence on the timbre sound similar. Both seem to produce a "softer" timbre with little brilliance. Projecting the original timbres onto the eigenvectors, we find that the resulting control parameters show a similiar behaviour as in single-sample PCA - see the Jupyter notebooks in the supplimentary material for more plots.

Guitar Moving on to the guitar, we see that the Explained Variance is less concentrated than for the single-sample PCA (Figure 12): Whereas for the single-sample PCA, u_0 explained almost 100% of the variance, here it explains a little less than 60%, with u_1, u_2 and u_3 explaining the largest share of the remaining variance. This is not necessarily a bad thing: As we said before, an overly simple model might not be interesting from a musical perspective. We might have found a sweet spot between complexity and simplicity. Comparing the eigenvectors to single-

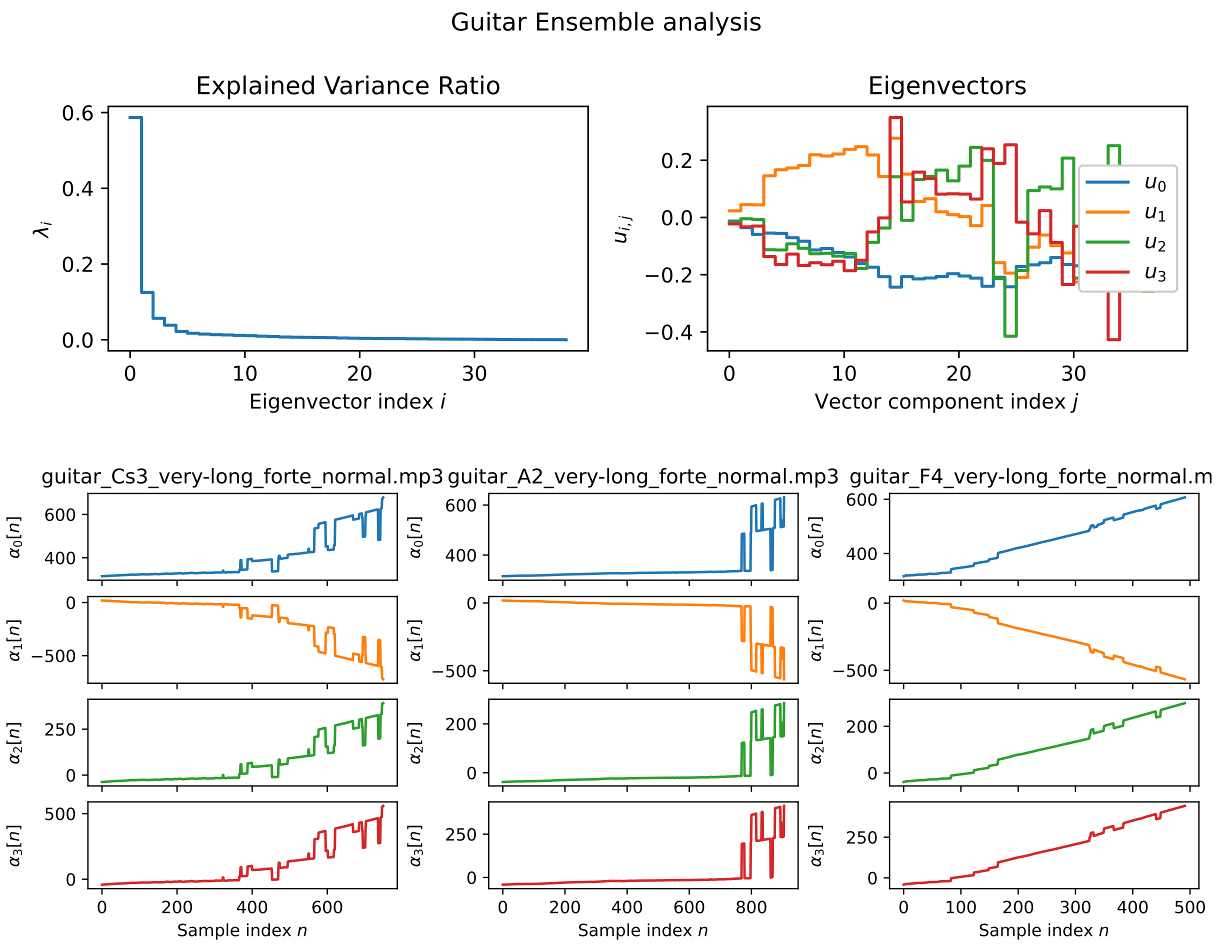

Figure 12: Top: Explained Variance and Eigenvectors for the guitar ensemble analysis. Bottom: Behaviour of control parameters for randomly chosen samples.

sample PCA, we see that their coefficients are distributed more evenly: There are fewer large jumps from component to component. Looking at the control parameters, we observe something curious: They behave differently from their single-sample counterparts: All parameters follow an approximately linear trajectory that contains sudden jumps (see the bottom part of Figure 12 for reference). The control parameters of one sample seem to all behave in unison: either they move in the same direction, or in the inverse direction, but they all contain the same kinks and jumps. We have not found an explanation for this behaviour. Finally, let us listen to the timbres produced with the principal components: 🔊 15. As with the cello, we notice a greater perceptual difference between the principal components. The difference between u_0 and u_1 seems especially pronounced to us.

4.5 Summary

In this chapter, we discussed applying PCA to our dataset. We have outlined preprocessing techniques necessary to transform the data into a form that is usable by PCA. We have set some criteria for evaluating the analyses, and finally applied PCA to single samples and on datasets of multiple samples. Single-sample PCA gave us a gauge on what to expect from the analyses, but ultimately fell short because it was too unstable. Ensemble PCA partially side-steps the instability. Ensemble PCA seems to yield a greater variety of principal components, which may be beneficial for building musical instruments. However, to judge if the ensemble analysis provides timbre axes that are actually useful in a musical sense, we have to put them to use in a musical instrument. We will briefly discuss this in the next chapter.

5 Building a Multi-Parameter Synthesizer

Using the principal components gathered in the previous chapter, a musical instrument can be implemented that produces timbres according to a set of control parameters. Originally, we planned to implement such a scheme using the SuperCollider[9] real-time synthesizer engine and a ROLI Blocks MIDI controller. Due to technical difficulties with the controller's firmware, I was unable to complete the synthesizer.

To get a sense of the instrument model's capabilities, listen to two artificial samples generated by the cello and guitar Ensemble models: 🔊 16 In both cases, the samples were synthesized from the mean timbre and the two most significant eigenvectors:

$$\mathbf{x}[t] = \bar{\mathbf{x}} + \alpha_0[t]\mathbf{u}_0 + \alpha_1[t]\mathbf{u}_1$$

For the cello, we use two phase-shifted low-frequency sinusoids as control parameters:

$$\alpha_0[t] = a_0 \sin(\omega_0 t), \;\; \alpha_1[t] = a_1 \sin(\omega_1 t + \pi/2)$$

For the guitar, we use a linear function and another low-frequency sinusoid:

$$\alpha_0[t] = a_0 t + b, \;\; \alpha_1[t] = a_1 \sin(\omega_1 t)$$

In our impression, the cello sample is not recognizable as a cello. The guitar sample seems a little more realistic, but it is also a very rough approximation of the guitar's timbre in our opinion.

6 Conclusion and Outlook

The goal of this thesis was to explore modelling an instrument's timbre continuum using generative models. To that end, we implemented a signal-processing analysis procedure devised by Serra. We have seen that we can reliably separate a signal into its deterministic and stochastic component. The procedure still has difficulty with peaks that are very close to the noise floor. A better estimation of the noise floor or a more flexible peak matching procedure might be needed here. Using the timbres we gathered from Serra's analysis, we applied Principal Component Analysis. We concluded that the models we have gathered so far are not sufficient to accurately model the variability of timbre of a complete instrument. The axes found in the analysis may however be useful in implementing artificial instruments that do not try to resemble an acoustic instrument.

It is unfortunate that the stochastic component $e[t]$ is not included in our timbre analysis process. Especially for the cello, the stochastic component is elementary for an accurate resynthesis of its timbre. Adding the stochastic component to the analysis procedure may greatly improve the quality of the results. Also, so far we have only considered modelling spectral patterns. Considering temporal patterns in the timbre analysis might also lead to more interesting timbres.

Overall, using PCA to model a timbre distribution seems flawed. PCA cannot cope with missing values and is *not robust* against outliers. It is best used with clean, curated data. On the other

[9] `https://supercollider.github.io/`

hand, the timbre vectors we gathered in our analysis will always be noisy and contain holes. To make PCA work with our dataset, we had to employ a complex feature engineering pipeline. Especially for the guitar, where often more than half of the amplitude values have to be imputed, it is not clear if our imputation strategy is actually sensible. Looking back at the PCA results we got for the guitar, we usually found that the most significant control parameter followed an almost perfectly straight line over the course of one sample. It seems as if the direction of the first principal component is predetermined by the extrapolation of the amplitude trajectories below the noise floor. If that is the case, the imputation strategy is unsuitable because it skews the covariance too heavily.

Furthermore, PCA *does not make use of locality*: Since the amplitudes of the overtones only change slowly with time, there is a dependence between the probability of a timbre $\mathbf{x}$ occurring at frame n, and a timbre $\mathbf{x}'$ occurring at frame $n + 1$ (or any other frame in the vicinity). PCA does not consider this, and treats the samples as identically independently distributed, potentially throwing away information.

We propose exploring an alternative approach: *Generative Adversarial Networks* (Goodfellow et al. 2014) are a modern strategy for latent parameter extraction. Briefly, they consist of a generative and a discriminative neural network that compete against one another. The generative network creates artificial data from a given set of latent parameters, and the discriminative network distinguishes between real and artificial data. Since GANs are often built using convolutional neural nets, they may be more robust against missing data and outliers. Convolutional layers do not consider single data points, but always look at the data points in the immediate vicinity as well (Khan et al. 2018, p. 43). In our case that means that a convolutional layer could look at a sequence of timbre vectors, instead of one at a time. Coupled with *Atrous Convolution* (Chen et al. 2016, Section 3) (i.e. inflating filter masks with different amounts of zeroes to get multiple sizes of receptive field), the model might be able to capture short-time temporal patterns of different sizes.

Summing up, we have seen that modelling timbres with generative models is generally possible, although PCA in combination with our simplistic timbre model does not seem flexible enough to deliver musically interesting results. More work is needed for accurate and musically interesting models. The timbre analysis procedure and the modelling considerations made in this thesis may serve as a starting point for further exploration.

7 The `principal_harmonics` software package

To facilitate the analyses, I implemented the Python package `principal_harmonics`. It contains an implementation of the *Sinusoids plus Noise* algorithm described in chapter 3, as well as functionality for handling large datasets, plotting and transforming the dataset as described in chapter 4. The package is implemented in pure Python and uses standard scientific libraries.[10]

Timbre analysis: `principal_harmonics.pvoc` This subpackage contains the signal processing needed to gather timbre vectors from audio files, implementing Serra's *Sinusoids Plus Noise* algorithm. Several implementations of this procedure exist - for example the `SprModelAnal` algorithm in the audio processing framework `essentia`.[11] However, to gain a better understanding of the procedure, and to provide an accessible, readable implementation of the algorithm, I decided to reimplement it myself in pure Python. The implementation is both lightweight enough to be included in other software, and flexible enough to facilitate other types of analyses with little additional work. Most Phase-Vocoder type analyses differ only in the way incoming peaks are handled and packed into an appropriate timbre representation - see section 3.2.4 for an outline of peak two matching procedures. A new strategy can easily be realized by implementing the `PeakMatcher` interface and passing it into the analysis. Furthermore, the rest of the procedure consists of components that are sufficiently decoupled to be used as "building blocks" for new timbre analysis schemes.

Analysis Workflow: `principal_harmonics.dataset` This subpackage contains some functionality to handle the analysis of large amounts of audio samples. We consider a *timbre dataset* a directory containing a `labels.csv` file, original audio files, and timbre analyses for each audio file. The labels file contains pitch information for each sample which may be used in the timbre analysis process. Other arbitrary fields can be provided, so that can be used later when exploring the timbre data. If the filenames have a known syntax, `build-labels.py` can build a label file automatically by parsing the filenames - implement a `FilenameParser` to parse a custom filename syntaxes. `build-dataset.py` then builds the complete timbre dataset from the labels and audio source files. To open a dataset, use `principal_harmonics.dataset.open_dataset(PATH)` to get a Pandas dataframe with all analysis results and corresponding labels. This way, Pandas' infrastructure for complex transformations filtering operations is easily accessible.

Timbre Preprocessing: `principal_harmonics.models` This subpackage contains the preprocessing that is used before applying PCA, as outlined in section 4.2. Transformations are implemented as `sklearn`-style transformers so that they can be used in standard `sklearn` pipelines.

Plotting: `principal_harmonics.plots` This subpackage contains some plots that might be useful to visualize timbre vectors. When used in an interactive environment such as a Jupyter notebook, timbres will be resynthesized ad-hoc when the user clicks on the plot, potentially aiding in exploring a timbre dataset interactively.

[10]`principal_harmonics` uses `numpy`, `scipy`, `sklearn` and `matplotlib`. `pya` is used to handle audio signals, and `librosa` provides some functionality not available in `pya`, namely the YIN pitch tracker and the parsing of note names.

[11]`https://essentia.upf.edu/reference/streaming_SprModelAnal.html`

References

Bishop, Christopher M. (2007). *Pattern Recognition and Machine Learning*. Springer. ISBN: 978-0387-31073-2.

Chen, Liang-Chieh et al. (2016). *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. DOI: 10.48550/ARXIV.1606.00915.

Cheveigné, Alain de and Hideki Kawahara (2002). "YIN, a fundamental frequency estimator for speech and music". In: *The Journal of the Acoustical Society of America* 111.4, pp. 1917–1930. DOI: 10.1121/1.1458024.

Ellermeier, Wolfgang, Jürgen Hellbrück, and Josef Schlittenlacher (2014). "Physiologische und psychoakustische Grundlagen des Hörens". In: *Akustische Grundlagen der Musik*, pp. 31–79. ISBN: 978-3-89007-699-7.

Goodfellow, Ian J. et al. (2014). *Generative Adversarial Networks*. DOI: 10.48550/ARXIV.1406.2661.

Harris, F.J. (1978). "On the use of windows for harmonic analysis with the discrete Fourier transform". In: *Proceedings of the IEEE* 66.1, pp. 51–83. DOI: 10.1109/PROC.1978.10837.

Khan, Salman et al. (2018). *A Guide to Convolutional Neural Networks for Computer Vision*. Springer. DOI: 10.2200/S00822ED1V01Y201712COV015.

Maher, Robert C. and James W. Beauchamp (1994). "Fundamental frequency estimation of musical signals using a two-way mismatch procedure". In: *The Journal of the Acoustical Society of America* 95.4, pp. 2254–2263. DOI: 10.1121/1.408685.

McAulay, R. and T. Quatieri (1986). "Speech analysis/Synthesis based on a sinusoidal representation". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34.4, pp. 744–754. DOI: 10.1109/TASSP.1986.1164910.

Miranda, Eduardo Reck (2002). *Computer Sound Design: Synthesis Techniques and Programming*. Focal Press. ISBN: 0-240-51693-1.

Moore, F. Richard (1990). *Elements of Computer Music*. P T R Prentice Hall. ISBN: 0-13-252552-6.

Pedregosa, F. et al. (2022). *sklearn User Guide*. Accessed: 2022-08-10. URL: https://scikit-learn.org/stable/user_guide.html.

Philharmonia Orchestra (year unknown). *Philharmonia Sound Samples*. https://philharmonia.co.uk/resources/sound-samples/. Accessed: 2022-07-05.

Serra, Xavier (Jan. 1997). "Musical Sound modeling with Sinusoids plus Noise". In: URL: https://hdl.handle.net/10230/45776.

Smith, Julius Orion and Xavier Serra (1987). "PARSHL: An Analysis/Synthesis Program for Non-Harmonic Sounds Based on a Sinusoidal Representation". In: URL: https://hdl.handle.net/10230/33794.

Tipler, Paul A. and Gene Mosca (2015). *Physik für Naturwissenschaftler und Ingenieure*. Springer Spektrum. ISBN: 978-3-642-54165-0. DOI: https://doi.org/10.1007/978-3-642-54166-7.

Weinzierl, Stefan (2014). "Schallereignisse und Musik. Typologie, Beschreibung, Analyse". In: *Akustische Grundlagen der Musik*, pp. 3–30. ISBN: 978-3-89007-699-7.

List of Audio Examples

The audio samples listed below can be found in the `audio-samples` directory of the supplementary material. Except for 🔊 1, they were produced by the code in the Jupyter notebooks, and converted to mp3 for smaller file size. Multiple files with the same prefix are abbreviated with an asterisk (`*`). Note that in the nomenclature of the Philharmonia dataset, `s` stands for *sharp* (♯). Thus, `As4` stands for A♯4, not A♭4 as one might expect when used to German note names.

🔊 1 `ch3_piano_note.mp3`
`ch3_piano_note_reversed.mp3`

🔊 2 `ch3_cello_Cs3_15_pianissimo_arco-normal_*.mp3`

🔊 3 `ch3_cello_Cs3_15_fortissimo_arco-normal_*.mp3`

🔊 4 `ch3_cello_E4_15_mezzo-piano_arco-normal_*.mp3`

🔊 5 `ch3_cello_A4_15_forte_arco-normal_*.mp3`

🔊 6 `ch3_guitar_As4_very-long_forte_normal_*.mp3`

🔊 7 `ch3_guitar_Ds3_very-long_piano_normal_*.mp3`

🔊 8 `ch4_cello_G3_15_fortissimo_arco-normal_clipped.wav`

🔊 9 `ch4_cello_single_pca_G3_u*.wav`

🔊 10 `ch4_cello_Gs3_15_fortissimo_arco-normal_clipped.wav`

🔊 11 `ch4_cello_single_pca_Gs3_u*.wav`

🔊 12 `ch4_guitar_Cs3_very-long_forte_normal_clipped.wav`

🔊 13 `ch4_guitar_single_pca_Cs3_u*.wav`

🔊 14 `ch4_cello-ensemble-pca-u*.wav`

🔊 15 `ch4_guitar-ensemble-pca-u*.wav`

🔊 16 `ch5_cello-model.wav`
`ch5_guitar-model.wav`