Single-sample PCA

In this notebook, we apply PCA to single audio samples. We'll use the following pipelines for analyzing single samples:

In [1]:
```
%matplotlib inline
%load_ext autoreload
%autoreload 2
from pathlib import Path

# Enter the locations of the sample directories
CELLO_PATH  = Path("/home/lukas/BA/philharmonia-samples/cello")
GUITAR_PATH = Path("/home/lukas/BA/philharmonia-samples/guitar")

# Output directories for figures and wavfiles
GFX_PATH    = Path("/home/lukas/BA/report/gfx/")
WAVS_PATH   = Path("/home/lukas/BA/report/wavs/")

# Whether to generate graphs for all samples. Image files will be
# written to the dataset directories.
GENERATE_ALL_GRAPHS = False
```

In [2]:
```
# Initialization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import librosa
import pya
import random

import principal_harmonics as ph

for path in [GFX_PATH, WAVS_PATH]:
    if path.exists() and not path.is_dir():
        raise NotADirectoryError(path)
    if not path.exists():
        path.mkdir()
```

In [3]:
```
def plot_single_pca(coefs, pipeline, fig, n_vecs,
                    filename, logscale_explained_variance=False):
    colors = ['tab:blue', 'tab:orange', 'tab:green', 'tab:red', 'tab:purple',
              'tab:brown', 'tab:pink', 'tab:pink', 'tab:gray', 'tab:olive', 'tab:cyan']

    fig.suptitle(filename)
    left_fig, right_fig = fig.subfigures(1, 2)
    var_ax, vec_ax = left_fig.subplots(2, 1)
    alpha_axs = right_fig.subplots(n_vecs, 1, sharex=True)

    trans = pipeline.fit_transform(np.abs(coefs))
    pca = pipeline[-1]

    var_ax.set_title("Explained Variance Ratio")
    var_ax.plot(pca.explained_variance_ratio_, ds='steps-post')
    var_ax.set_xlabel("Eigenvector index $i$")
    var_ax.set_ylabel("$\lambda_i / \sum_k \lambda_k$")
    if logscale_explained_variance:
        var_ax.semilogy()
        var_ax.set_ylim(bottom=1e-7)

    vec_ax.set_title("Eigenvectors")
    vec_ax.set_xlabel("Vector component index $j$")
    vec_ax.set_ylabel("$u_{i,j}$")
    for i, vec in enumerate(pca.components_[:n_vecs]):
        vec_ax.plot(vec, label=f'$u_{i}$', ds='steps-post')
    vec_ax.legend(loc='right')

    left_fig.align_ylabels()

    alpha_axs[0].set_title("Resulting control parameters")
    alpha_axs[-1].set_xlabel("Frame index $n$")

    for i, (alpha, color, ax) in enumerate(zip(trans.T[:n_vecs],
                                               colors,
                                               alpha_axs)):
        ax.plot(alpha, color=color)
        ax.set_ylabel(f'$\\alpha_{i}[n]$')
    right_fig.align_ylabels()
```

In [4]:
```
from principal_harmonics.models import *
from sklearn.pipeline import make_pipeline

guitar_single_pipeline = make_pipeline(
    DropDCTransformer(),
    DBTransformer(),
    HoleImputer(hole_size_limit=10),
    RidgeSubNoiseImputer(constant_value=-240),
    PCA()
)

cello_single_pipeline = make_pipeline(
    DropDCTransformer(),
    DBTransformer(),
    HoleImputer(hole_size_limit=10),
    FilterIncompleteTimbresTransformer(),
    PCA()
)
```

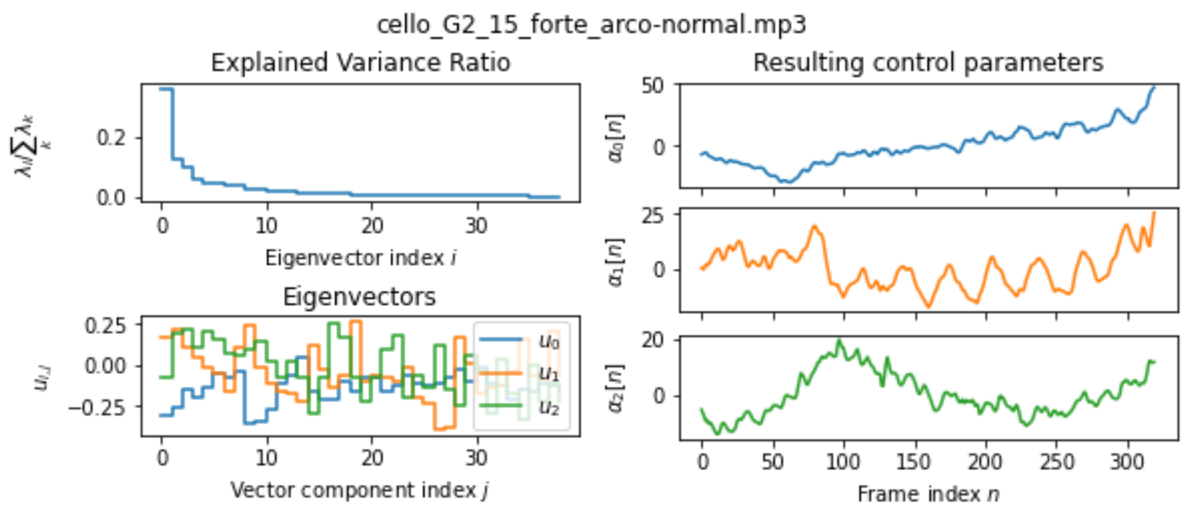
Let's try the Cello first:

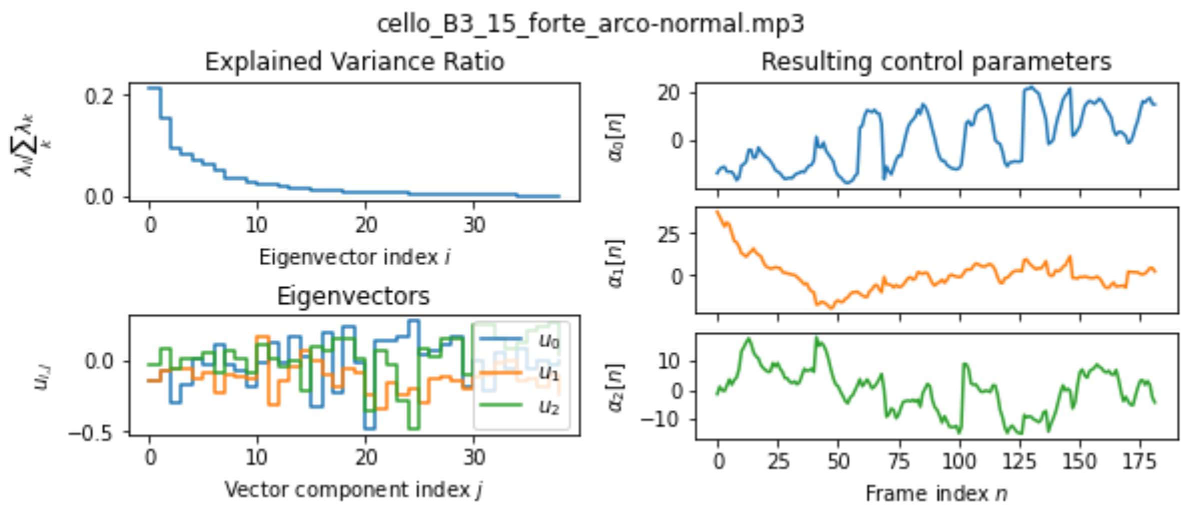
In [5]:
```
cello_df  = ph.dataset.open_dataset(CELLO_PATH)
# let's exclude the degraded samples (See section 3.3 of the report)
cello_df = cello_df[cello_df.midi > librosa.note_to_midi('D2')]
guitar_df = ph.dataset.open_dataset(GUITAR_PATH)
```

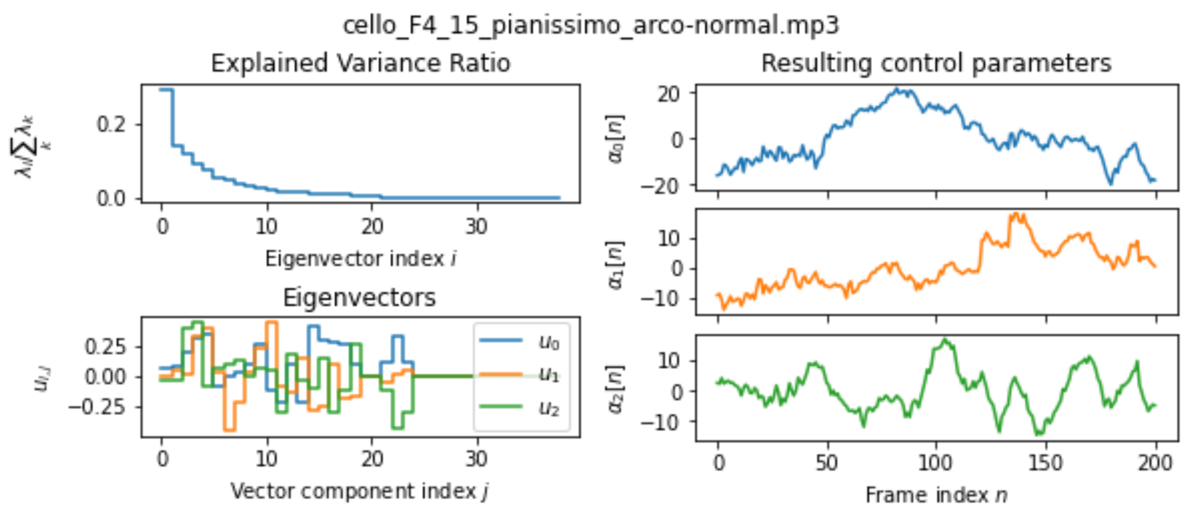
In [6]:
```
if GENERATE_ALL_GRAPHS:
    for filename, row in cello_df.iterrows():
        fig = plt.figure(figsize=(8.2, 3.5), constrained_layout=True)
        plot_single_pca(row.coefs, cello_single_pipeline, fig, 3, filename)
        plt.savefig(CELLO_PATH / (filename + '-SINGLE_PCA.png'))
        plt.close()
```

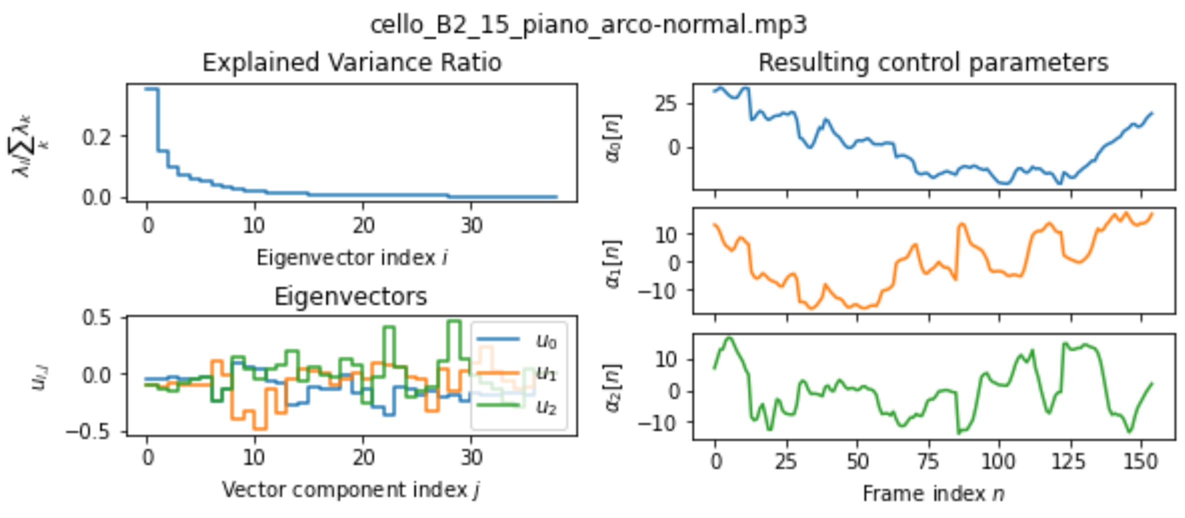
Let us first look at some randomly chosen samples to get a gauge of what the analysis results will look like:

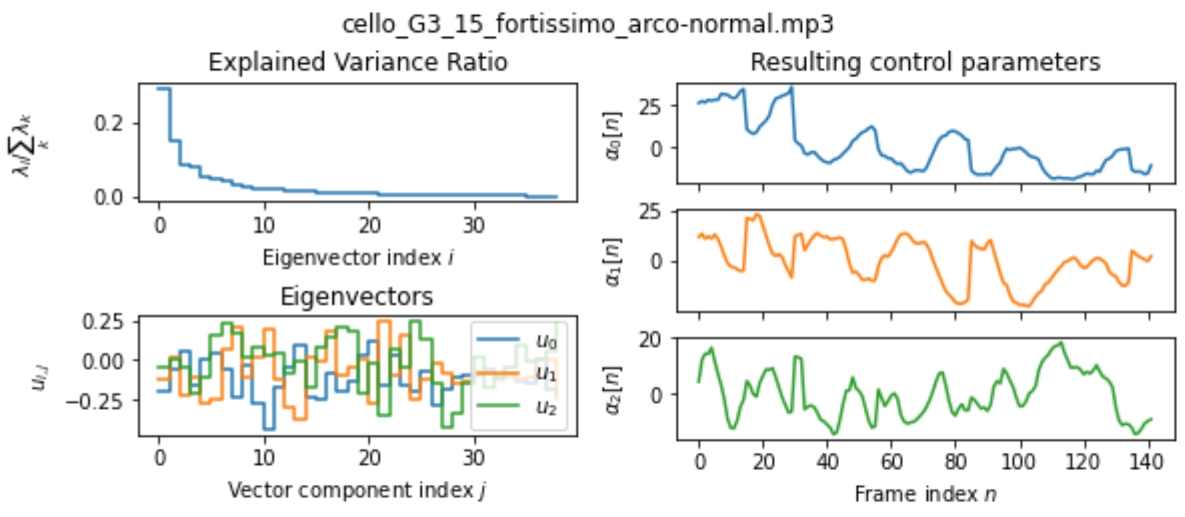
In [7]:
```
for filename, row in cello_df.sample(n=5, random_state=42).iterrows():
    fig = plt.figure(figsize=(8.2, 3.5), constrained_layout=True)
    plot_single_pca(row.coefs, cello_single_pipeline, fig, 3, filename)
```











Notice that the first control parameter α_0 behaves very differently among the analyses. Let's confirm this by looking at two samples that are perceptually very close to each other:

In [8]:
```
%%capture
fnames = ['cello_G3_15_fortissimo_arco-normal.mp3', 'cello_Gs3_15_fortissimo_arco-normal.mp3']
for i, fname in enumerate(fnames):
    coefs = cello_df.loc[fname].coefs
    fig = plt.figure(figsize=(8.2,3.5), constrained_layout=True)
    plot_single_pca(coefs, cello_single_pipeline, fig, 3, fname)
    plt.savefig(GFX_PATH / f'3-cello-single-{i}.eps')
```

In [9]:
```
cello_g3_source_asig = pya.Asig(str(cello_df.loc['cello_G3_15_fortissimo_arco-normal.mp3'].clipped_wav))
cello_g3_source_asig.save_wavfile(WAVS_PATH / 'ch4_cello_G3_15_fortissimo_arco-normal_clipped.wav')
cello_gs3_source_asig = pya.Asig(str(cello_df.loc['cello_Gs3_15_fortissimo_arco-normal.mp3'].clipped_wav))
cello_g3_source_asig.save_wavfile(WAVS_PATH / 'ch4_cello_Gs3_15_fortissimo_arco-normal_clipped.wav')
# cello_g3_source_asig.play()
# cello_gs3_source_asig.play()
```

Out[9]:
```
Asig('/home/lukas/BA/philharmonia-samples/cello/cello_G3_15_fortissimo_arco-normal_clipped.wav'): 1 x 80384 @ 44100
Hz = 1.823s cn=['0']
```

In [10]:
```
def make_axis_asig_oscillating(mean, ui, scale, note, f=0.5) -> pya.Asig:
    ts = np.linspace(0, 4, 4*int(44100 // 256))
    dbs = mean.reshape(1, -1) + ui.reshape(1, -1) * scale * np.sin(2*np.pi * ts * f).reshape(-1, 1)
    ampls = pya.dbamp(dbs)
    #ampls = ampls / ampls.sum(axis=1).reshape(-1, 1) * 0.1
    return ph.pvoc.additive_resynth(librosa.note_to_hz(note), ampls)

def make_all_axes_asigs(pipeline, df, fname, note, output_prefix):
    trans = pipeline.fit_transform(np.abs(df.loc[fname].coefs))
    pca = pipeline[-1]
    mean = pca.mean_
    asigs = []
    for i in range(3):
        ui = pca.components_[i]
        scale = 2*np.std(trans[:, i])
        asig = make_axis_asig_oscillating(mean, ui, scale, note)
        asig.save_wavfile(str(WAVS_PATH / f'{output_prefix}_u{i}.wav'), dtype='float32')
        asigs.append(asig)
    return asigs

cello_g3_asigs  = make_all_axes_asigs(cello_single_pipeline, cello_df, fnames[0], 'G3', 'ch4_cello_single_pca_G3')
# for easier comparison, we'll use G3 as the pitch for both resyntheses
cello_gs3_asigs = make_all_axes_asigs(cello_single_pipeline, cello_df, fnames[1], 'G3', 'ch4_cello_single_pca_Gs3')
```

```
(1, 1) (1, 1) (688, 39)
(1, 1) (1, 1) (688, 39)
(1, 1) (1, 1) (688, 39)
(1, 1) (1, 1) (688, 39)
(1, 1) (1, 1) (688, 39)
(1, 1) (1, 1) (688, 39)
```

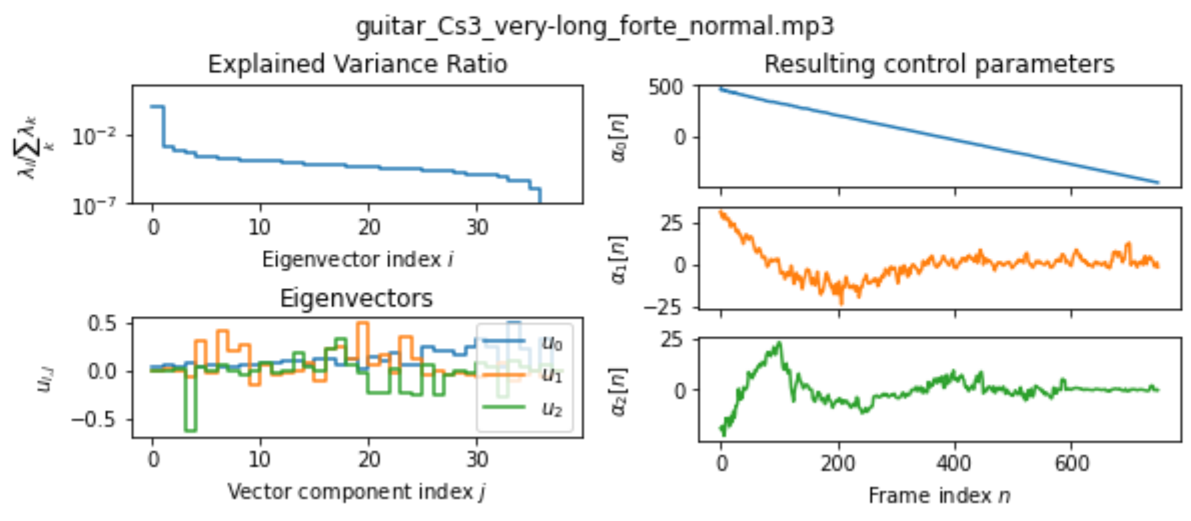
In [11]:
```
# cello_g3_asigs[0].play()
```

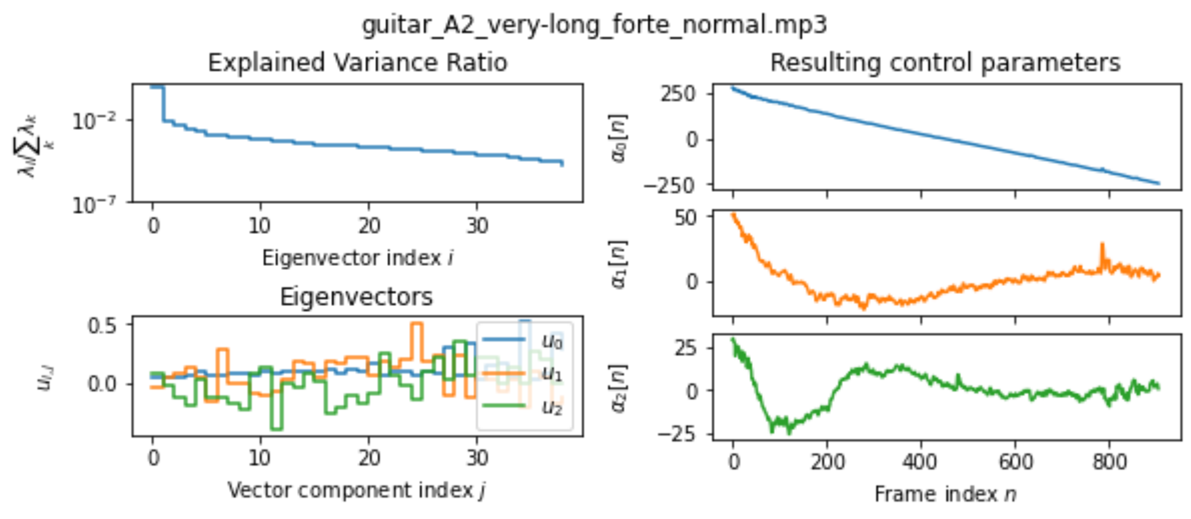
Moving on to the guitar

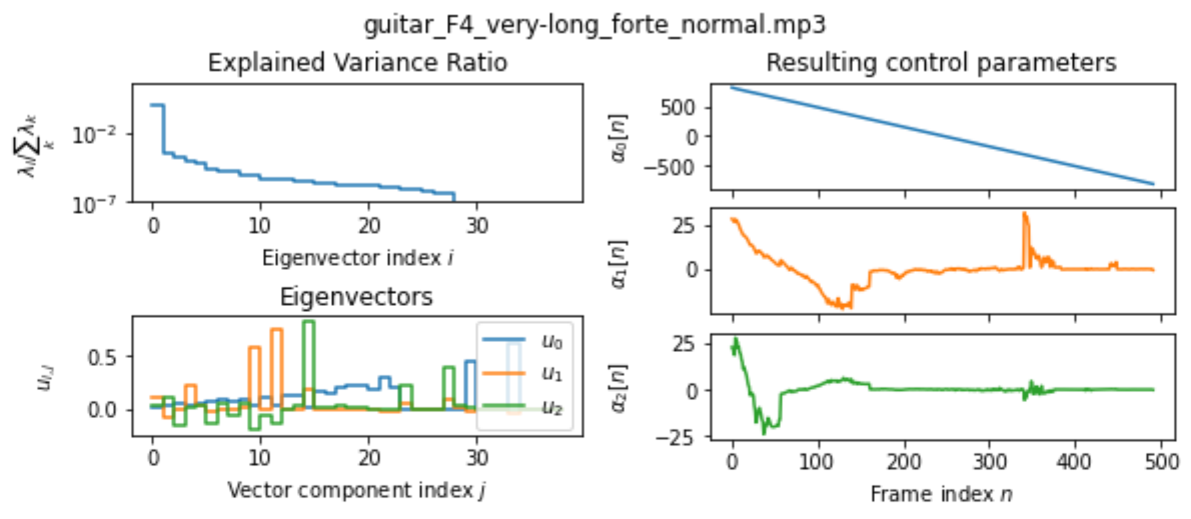
In [12]:
```
if GENERATE_ALL_GRAPHS:
    for filename, row in guitar_df.iterrows():
        fig = plt.figure(figsize=(8.2, 3.5), constrained_layout=True)
        plot_single_pca(row.coefs, guitar_single_pipeline, fig, 3, filename, logscale_explained_variance=True)
        plt.savefig(GUITAR_PATH / (filename + '-SINGLE_PCA.png'))
        plt.close()
```

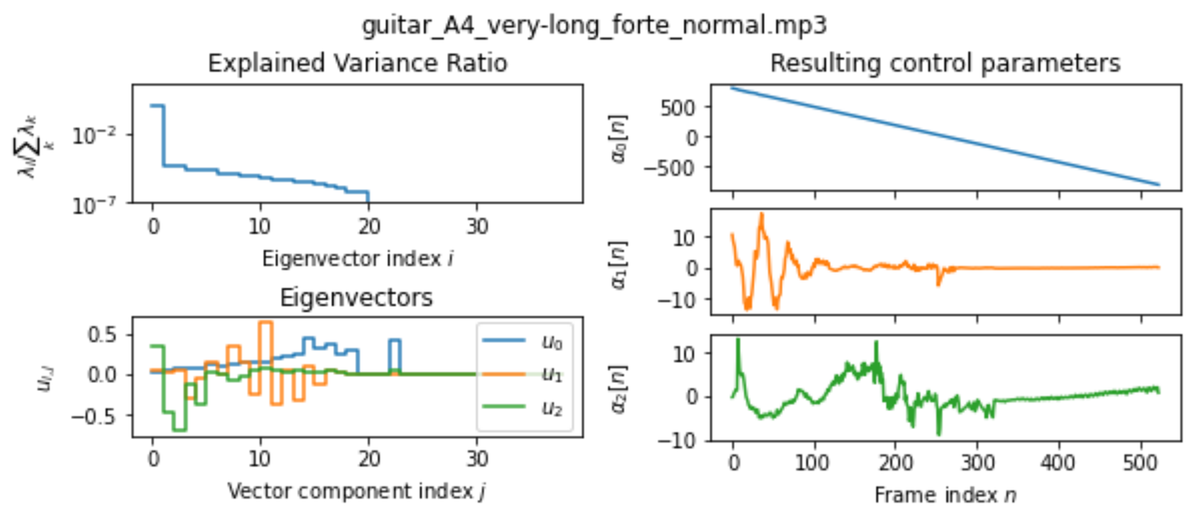
As before, let's first look at some randomly chosen samples.

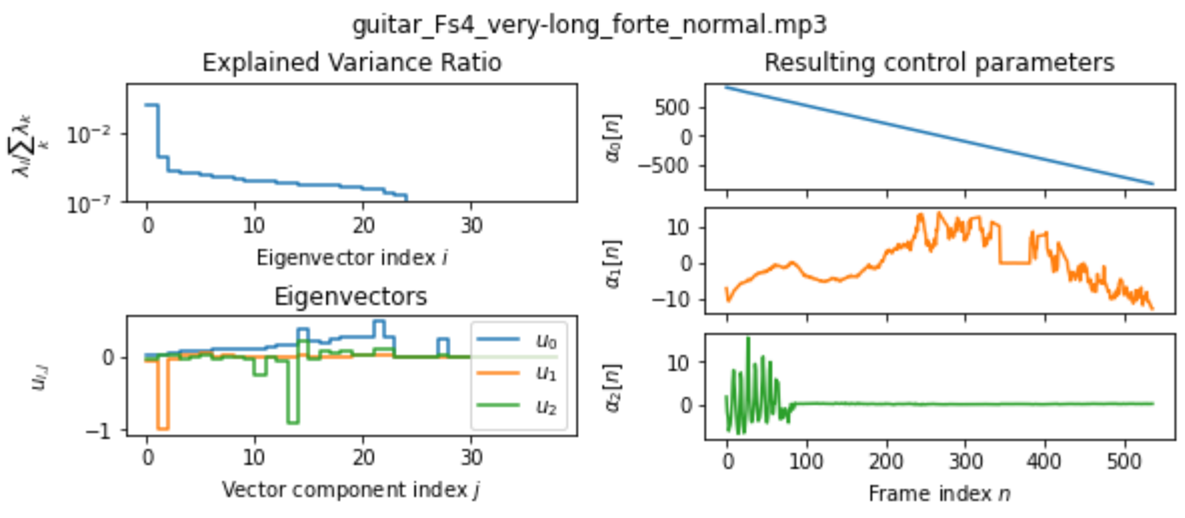
In [13]:
```
for filename, row in guitar_df.sample(n=5, random_state=42).iterrows():
    fig = plt.figure(figsize=(8.2, 3.5), constrained_layout=True)
    plot_single_pca(row.coefs, guitar_single_pipeline, fig, 3, filename, logscale_explained_variance=True)
```











In [14]:
```
%%capture
fig = plt.figure(figsize=(8.2, 3.5), constrained_layout=True)
fname = 'guitar_Cs3_very-long_forte_normal.mp3'
plot_single_pca(np.abs(guitar_df.loc[fname].coefs), guitar_single_pipeline, fig, 3, fname, logscale_explained_varia
plt.savefig(GFX_PATH / '3-guitar-single.eps')
```

In [15]:
```
guitar_cs3_source_asig = pya.Asig(str(guitar_df.loc[fname].clipped_wav))
guitar_cs3_source_asig.save_wavfile(WAVS_PATH / 'ch4_guitar_Cs3_very-long_forte_normal_clipped.wav')
# guitar_cs3_source_asig.play()
```

Out[15]:
```
Asig('/home/lukas/BA/philharmonia-samples/guitar/guitar_Cs3_very-long_forte_normal_clipped.wav'): 1 x 192000 @ 4410
0Hz = 4.354s cn=['0']
```

In [16]:
```
guitar_asigs = make_all_axes_asigs(guitar_single_pipeline, guitar_df, fname, 'C#3', 'ch4_guitar_single_pca_Cs3')
```

```
(1, 1) (1, 1) (688, 39)
(1, 1) (1, 1) (688, 39)
(1, 1) (1, 1) (688, 39)
```

In [17]:
```
# guitar_asigs[0].play()
```