EigenSound: Song Visualization for Edition Purposes

Marcelo Cicconet, Paulo Cezar Carvalho

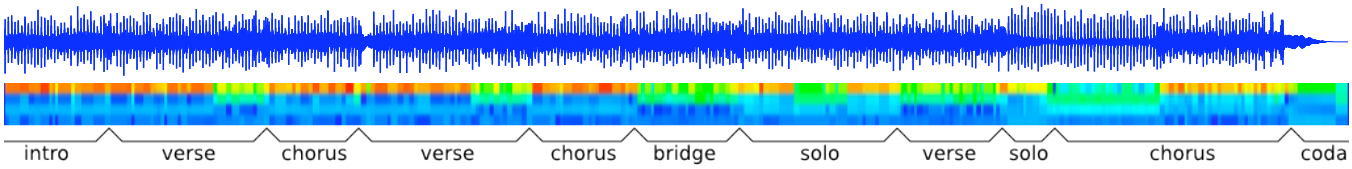

Figure 1. Top: waveform. Botton: eigensound.

***Abstract*—In this work we propose a representation of audio files more suitable for the task of editing audio and video than the traditional waveform graph, in the sense that it helps finding clusters and onsets visually, due to the array of colors associated to each previously computed segment. Segmentation and color-mapping are done by means of well known image processing techniques: bidimensional convolution and spectral decomposition, respectively.**

***Keywords*-Audio Visualization; Spectral Decomposition; Self-Similarity Matrix; Music Information Retrieval.**

I. Introduction

The waveform plot of a sound is the simplest representation of uncompressed digital audio. Supposing the audio is mono (one channel only), that representation consists of the graph of timestamp versus amplitude. When the audio samples are taken from timestamps equally spaced, it's called Pulse Code Modulation (PCM).

This way of seeing songs is widely used in all sort of audio and video editors, a somehow surprisingly fact, since such a low level description of audio in most cases gives no sufficient indication of segment onsets nor hints about regions similar to each other. As an example, if you look at the top part of figure 1 you will find it difficult to gess what could be the solo region.

So we started researching for an audio visualization more helpfull to people working with audio and video, and this text describes some progress we have made in that direction. The method developed is based on the literature of audio summarization, especially [1]. Audio summarization, as our problem, demands finding onsets/offsets and locate important regions of the sound: verse, refrain, chorus, solo, bridge, coda, etc. We will not, however, try to name those regions. Instead, we will make their differences/simmilarities more visible.

In section II some low level audio features used will be introduced. Audio features are extracted from equally spaced and overlapping windows. Section III describes how they are used to find segment onsets/offsets, by looking at a diagonal band of the corresponding Self-Similarity Matrix. After segmentation, a small Self-Similarity Matrix is built, and its spectral components are computed to construct representative images like that of figure 1 - botton. Section IV will explain that process. Results and final comments will take place at section V.

II. Audio Features

Our process begins by transforming the raw audio data into a sequency of feature vectors. A feature vector is usually computed from the entries absolute values of the Fast Fourier Transform (FFT) of successive overlaping windows, and describes a (sometimes perceptually related) property of the sound. The vector of squared absolute values of the FFT is called the *frequency spectrum* of the audio window.

The chosen feature, the window size and the overlap (hop) size are tunable parameters. We have experimented with 8 different one-dimensional features (see [2] for details): loudness, spectral flux, spectral centroid, spectral spread, spectral skewness, spectral roll-off, spectral flatness and spectral crest.

One-dimensional features can be computed for different bands of the frequency spectrum, defining a feature whose dimension is the number of bands. We have used loudness and spectral crest in that way, with 10 non-overlapping frequency bands. Finally there are the *pre-chroma* and the *chroma* vectors. The first consist of the energies corresponding to each one of the 84 MIDI notes ranging from 24 to 107. By packing the energies of notes with the same name we have the second ([3]).

III. Segmentation

Given feature vectors v_i and v_j, the similarity between them is defined by $S(i,j) = 1 - \|v_i - v_j\|^2/M$ (where $M = \max_{k,l} \|v_k - v_l\|$) and the matrix S so built is called Self-Similarity Matrix (SSM). See figure 2 (left) for an example.

SSM's usually have a checkerboard pattern. So we can segment the song by convolving a diagonal band of the SSM with a checkerboard-shaped kernel and look for peaks of the

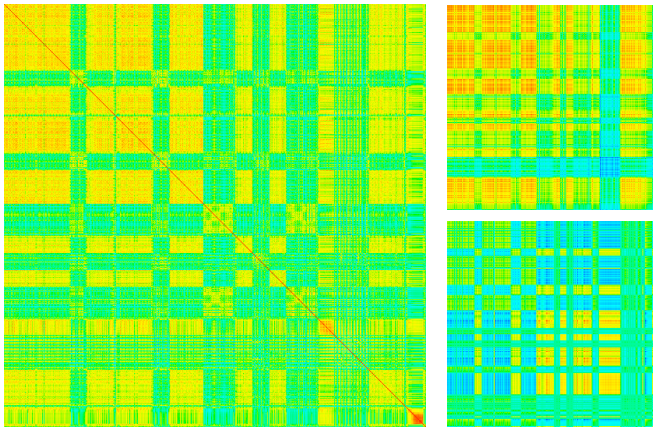

Figure 2. Left: SSM of ACDC's Anything Goes song, relative to the the pre-chroma feature, with a window resp. overlap of about 743 resp. 372 mili-seconds. The colormap goes from blue (low similarity) to red throw the HSV colorspace. Right: Normalized first (top) and second components of the spectral decomposition of the segment-indexed SSM. Feature is pre-chroma, window resp. hop size are about 93 resp. 23 mili-seconds long.

resulting curve (the so called *novelty score*) or of a smoothed version of it (fig. 3, top).

Peaks above a threshold define onsets, and a segment is the set of windows between consecutive onsets. The mean feature vector of each segment will represent the segment, and a new Self-Similarity Matrix is computed using that means. This matrix is called *segment-indexed* SSM.

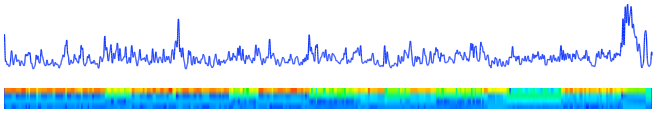

Figure 3. Top: novelty score of ACDC's Anything Goes song, using the pre-chroma feature, window resp. hop size of about 93 resp. 23 mili-seconds, checkerboard kernel and smoothing kernel widths being equal to 32 windows. Botton: Corresponding EigenSound image, using 4 componentes of the segment-indexed SSM's spectral decomposition.

IV. Clustering

SSM's are symmetric matrices, thus the spectral decomposition applies, i.e., an SSM can be written as $\sum \lambda_i v_i v_i^{\mathsf{T}}$, where λ_i and v_i are eigenvalue and eigenvector pairs. Sorting the eigenvalues in decreasing order, the first components represent most of the SSM information, different components describing de contribution of different parts to the total image. Figure 2 (right) presents two components of a segment-indexed SSM.

In the particular case of SSM's, summing components along columns will show song regions with high self-similarity. Therefore a song representative image can be build by stacking along column sums of spectral component images, the line number being equal to the component index (figure 3, botton).

That image can halp finding onsets and clusters by visual inspection, making easier the task of editing audio and video. Refer to figure 1 for an example.

V. Conclusion

In this work we have proposed a method to represent a song by means of an image based upon previous segmentation (via convolution along a diagonal band of the SSM) and clustering (using spectral decomposition of the segment-indexed SSM).

The tricky point of this project is the selection of the proper set of parameters, especially the feature vector, in view of the fact that the only way to evaluate the goodness of the result is by eyes/ears inspection. For that reason we have implemented an app to facilitate searching for good sets of parameters (fig. 4), which can be downloaded from the project website [4].

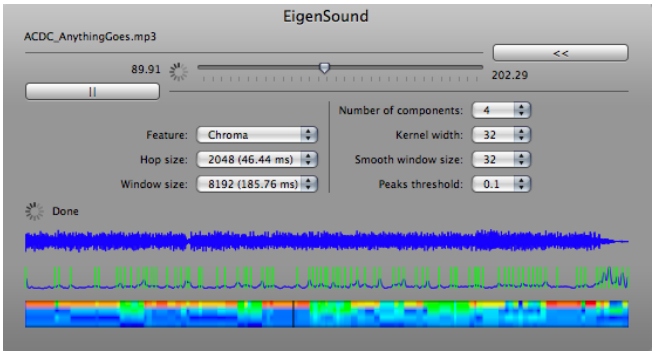

Figure 4. Screenshot of the app we have implemented to halp finding good sets of parameters.

Pre-chroma and chroma behave better for songs with strong harmonic base, since they are designed to be pitch and chord sensitive. On the other hand, for songs which present phases of high and low intensity, loudness has been a good choise of feature.

As future work we intent to combine one feature from each of the three perceptual fields (intensity, pitch and timbre) and search for optimum weights representing them.

References

[1] M. Cooper and J. Foote, "Summarizing popular music via structural similarity analysis," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[2] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," IRCAM, Tech. Rep., 2004.

[3] T. Jehan, "Creating music by listening," Ph.D. dissertation, MIT Media Laboratory, 2005.

[4] http://w3.impa.br/~cicconet/thesis/eigensound/.