AUTHOR GUIDELINES FOR ICASSP 2013 PROCEEDINGS MANUSCRIPTS

Aneesh Vartakavi, Cameron Summers

Georgia Tech. Center for Music Technology (GTCMT)

ABSTRACT

This paper details the implementation of a standalone desktop application in C++ for retrieval of audio files by rhythmic and timbral similarity, notable for being, to our knowledge, the only application of its type under the GNU General Public License (GPL). The implementation focuses on audio loops - short audio files intended to restart immediately upon completion for compositional purposes - to address the difficulty of navigating large loop libraries manually. The audio similarity measure utilizes tempo detection and statistical features of a spectral frame periodicity to provide a ranking of most similar audio files to a particular reference file. Although similarity measures are somewhat subjective and difficult to evaluate, we propose two evaluations: 1) A kNN classification with 63% accuracy on a ground truth data set developed by the authors using real-world audio loops and 2) A synthetic ground truth data set developed by the authors using related beat patterns at different tempos that demonstrates tempo invariance in retrieving rhythmically similar audio. [1]

Index Terms— One, two, three, four, five

1. INTRODUCTION

With the advances in technology, large libraries of audio loops on a single hard drive have become commonplace. Navigating these libraries becomes increasingly difficult because knowledge about the files must be manually tagged or inspected by ear. A computational approach to audio retrieval via a similarity measure dramatically reduces the time spent navigating these libraries.

In this paper, we detail an implementation of a standalone desktop application in C++ for retrieval of audio loops by rhythmic and timbral similarity. A common task is finding an audio sample to sound simultaneously or in succession with a particular reference sample. The application allows a user to filter loops outside an acceptable range of tempo along with rhythmic and timbral distance given a particular reference loop. The tempo detection uses a simple approach based on characteristics of audio loops and the rhythmic distance is based on features derived from spectral periodicity. The timbral measure is based on euclidean distance between features derived from the MFCCs.

2. RELATED WORK

Several studies exist on audio retrieval by rhythmic similarity using statistical features since this approach requires limited or no human input to operate. One successful instance uses correlated energy peaks across frequency sub-bands [REF-Scheirer] while another uses a feature vector from a low frequency-weighted short-time fourier transform (STFT) [REF-Wold]. Another approach called the Beat Spectrum is utilized for rhythmic similarity in this application, because it is based on self-similarity of the audio, no particular features are required of the audio such as silence or periodic energy peaks [REF-Foote] - Only repetitive events are required, which are common in audio loops. We derive a periodicity measure from the similarity matrix similar to [REF - Foote] for our application.

Mel Frequency Cepstral Coefficient based timbre space has been found to be a good model for perceptual timber space (REF - Thirteen). MFCCs have since been used to create similarity measures (REF - Quantitative Analysis) for genre recognition (REF - lightweight measures), (REF - evaluation). We derived a simple timbral distance measure based on the euclidean distance between a feature vector derived from the MFCC's.

Several closed-source applications exist for determining audio similarity in libraries of audio files such as Similarity [REF-Similarity]. However, Similarity and most others are meant primarily for file library organization (e.g. removing duplicate files) and not for rapid navigation for compositional purposes. Another open-source command line application under a BSD-style license written in Python, AudioCompare [REF-AudioCompare], utilizes indices of dominant frequencies of the STFT for a general similarity measure, but this is not a musically descriptive measure and unreliable for our use case.

3. METHODOLOGY

This section describes...

3.1. Rhythmic Similarity

The rhythmic similarity recommendation algorithm is composed of two primary components that can be applied together

or independently in the application: tempo estimation and distance between feature vectors of the Beat Spectrum. For tempo estimation of an audio file, we use a simple method shown in [Equation 1] where T is the tempo, n is the number of expected beats in the loop, fs ¡format!¿ is the sampling frequency of the audio, and N is the number of samples. This method is based on an assumption that the length of the audio loops in beats will always be a power of 2 of a 4/4 measure - a group of 4 beats - which is generally true for many loop libraries. Additionally, we scale the tempo to a range of 50-160 eats per minute.

Insert EQUATION

While there can be errors with this method when the assumptions do not hold, it worked consistently well for the real-world EarSketch [REF-EarSketch] loops used for evaluation. Improvements to the application could incorporate a more robust tempo estimate.

The second component of rhythmic similarity comes from a calculation of statistical features of spectral periodicity, called the Beat Spectrum by [REF-Foote] who provides a detailed overview. We calculate the spectral periodicity as described in [REF-Foote] and include some processing steps prior to computing features. The spectral periodicity of an audio file is derived from a similarity matrix S where Si,j ¡format!¿ is the cosine distance between the STFT of i-th and j-ith block of the audio file. An example of S is shown in [FIGURE 1]. A block size of 1024 and hop size of 512 were used for the STFT. At a typical sampling frequency of 44100Hz for the audio loops we tested, this provides resolution in time of 23ms, which is sufficient for detecting rhythms even at faster tempos that adequately describe the audio file. Variations on these values could be tested in future research to determine if optimization is possible.

Next each super diagonal of S is summed as in [Equation 2] to give a representation of the spectral periodicity in the lag domain, where l is lag and k is the block index. An example of the result is shown in [FIGURE 2], where the periodicity of events in the audio is evident. Alternatively, the spectral periodicity can be viewed as an autocorrelation of a vector of elements where each element is the STFT of a block of samples in the audio file.

Insert EQUATION 2

In the implementation in application, the spectral periodicity was calculated using a single similarity matrix with a length equal the number frames in the entire audio file. The audio file could alternatively be segmented and the spectral periodicity calculated on each segment as described by the Beat Spectrogram in [REF-Foote], but since audio loops are often taken as a single unit for the compositional use case, our approach seems more appropriate for the application. Future implementations might explore this alternative and compare the performance for similarity ranking.

Prior to calculating features on Beat Spectrum some preprocessing steps were taken to increase robustness of the measure. First, since the Beat Spectrum is an autocorrelation and the number of elements summed decreases as the lag increases, a line was fit using least squares and subtracted from the signal. Next, the signal was inverted so that lags of lower distances had higher values by subtracting the signal from an equally long signal with every value a 1. Finally, a three point moving average filter was used on the Beat Spectrum for smoothing.

After the preprocessing steps, a feature vector was constructed to describe the Beat Spectrum signal. These features are shown in [Table 1]. And lastly, the algorithm determines a rhythmic distance measure by calculating the euclidean distance between Beat Spectrum feature vectors of two audio files

¡INSERT TABLE 1¿

3.2. Timbral Similarity

The timbral distance measure was based on thirteen dimensional MFCCs, which were computed as described in (REF - Quantitative). Additionally, we also computed the first derivative of the MFCC coefficients, storing a 52 dimensional feature vector of the mean and standard deviation of the MFCCs and their derivatives for each audio file. The timbral distance measure was formulated as the Euclidean distance between the aforementioned feature vector, scaled from 0-1 for each audio file.

3.3. Implementation

Processing Beat Spectrum features on large libraries of files requires speed and this informed the application design decisions. We chose the C++ framework JUCE [REF] because C++ code can be optimized efficiently for time and memory and JUCE is highly regarded in the audio software development sphere for its user interface styling and audio utilities. Additionally, JUCE allows for multi-platform development and deployment, making greater the potential future dissemination of the application. We also use the Eigen Library, which provides optimized linear algebra routines and an FFT wrapper.

3.4. User Interface

Given the difficulty of navigating large loop libraries, the user interface of the application seeks to assist the algorithm in making this process simple and fast. A user can select a reference file in a list box on the left on which to compare other files in the library and has the option of listening to that file. Next there are two range sliders available, one for tempo and one for beat spectrum feature vector distance, that filter all files outside of the specified range. Then Similar files are automatically shown or removed in a left list box according to the slider positions and those can also be played back by the

user. The application also writes previously calculated features for a directory in a cache and allows the user to load and reuse that cache to avoid calculating features on the directory again.

4. EVALUATION

5. REFERENCES

[1] Finger C. Er Z. Connel C. Dang, A., "Audiocompare," https://github.com/charlesconnell/AudioCompare, December 7 2013.