Audio Fingerprinting for Recognition

Michael Mandel

January 26, 2005

Abstract

Shazam's audio features, consisting of pairs of spectral peaks with their associated difference in time, form a useful representation for identifying identical audio clips in the presence of noise and distortion. This project implements a shazam feature extractor and attempts to generalize it from the very specific identity detector to a less specific auditory similarity measure. This generalization unfortunately did not meet with much success, but we have created a number of reduced-data songs from the shazam representation that are still recognizeable even with no additional information from the original song.

1 Introduction

Recently, the company Shazam has begun a service in England for identifying musical recordings from their audio content alone. Users submit audio queries over the telephone and their service responds with a text message containing the artist and title of the particular song. Not only is the audio *fingerprint*, as they call it, robust to the distortions and band-limitation of cellular telephone communication, it is also powerful enough to identify music in the background of a newscast, to recognize simulatneously up to eight songs mixed together, and to distinguish between seemingly identical renditions of the same song.

While one of its most valuable features for track identification, this specificity of the fingerprint also prevents its use in other, more flexible musical similary measurement situations. This project aims to loosen the strict constraints on matching audio content and to use a modified version of this fingerprint feature for musical similarity assessment.

I will first describe the shazam fingerprint features in more detail before describing its generalization as an audio feature.

2 Shazam Feature Extraction

For an audio feature to survive the coding and decoding of cell phone transmission, it must closely align with those characteristics of a sound that the codec preserves, namely those to which the human ear is most closely attuned. We hypothesize that such features are onsets in the spectral domain, henceforce referred to as *peaks*. Furthermore, while one peak

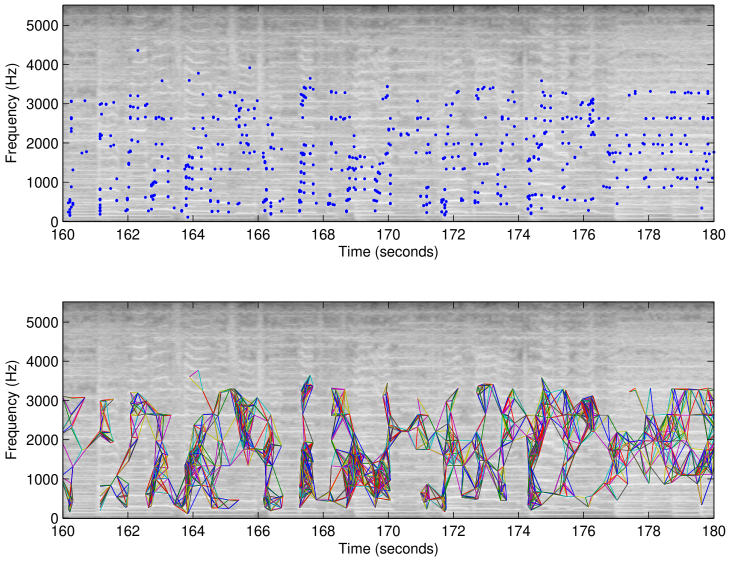

Figure 1: The peaks (top) and hash keys (bottom) generated from a portion of the Beatles' "Lucy in the Sky with Diamonds" overlaid on top of the spectrogram.

at a certain frequency is indistinguishable from another at the same frequency, pairs of peaks with their associated temporal separations are much more unique. This section will describe a means for finding peaks and pairs of peaks and for matching a distorted audio clip to the entire sound from which it originated.

2.1 Finding Peaks

This algorithm functions on the short-time fourier transform of an audio clip, also known as the spectrogram. We hypothesize the most noticeable features for human listeners are those areas of the spectrum that both contain a lot of new acoustic energy, i.e. energy that has just appeared. In order to find such points, we process spectral windows of 100ms at a time, looking for both the largest energies and the largest increases in energy. Points that fall into both of these groups are labelled as peaks. See Figure 1 for an example of a spectrogram overlayed with its peaks.

2.2 Finding Pairs of Peaks

Once the peaks are extracted, we would like to pair proximate points to each other to create more specific features as described above. To accomplish this, we search in a specified time-frequency window around each peak. To avoid duplicate pairs and self-pairing, this window actually begins slightly after each peak and extends typicaly 400ms after the peak and 400Hz above and below each peak in frequency.

One may adjust the size of this window in order to obtain more or fewer pairs of peaks and the types of pairs found. For example, a window extending two seconds beyond each peak would group together separate auditory events that occur in sequence, while a window extending just 100ms would miss some of the structure in longer events. Similarly, the wider the frequency band searched, the more specific the resulting pairs, but also the increase in spurious pairs created by grouping coincident but independent sources together.

2.3 Recognizing Matches

Each pair of peaks is stored as the key to hash, the value of which is the first peak's time of occurrence and the song ID, when training. Thus a library of hashes can be built for known songs. When an unknown song is presented to the system, it is hashed in the same way, and then each pair of peaks is looked up in the hash table of known songs. Even though the occurrence of these peak pairs is likely offset from the original song, that offset will be the same for all of the matching hash keys. So to test for a matching song, the matched offsets from each song are examined for a statistically significant spike at the offset at which the unknown clip began. See Figure 2 for an example of the offsets recorded for both a matching and an unmatching database song.

3 Using the Features

The specificity of these hash keys makes them unsuitable for use in general audio similarity measurement. Since identical audio excerpts produce identical hash keys even under distortion, we might also presume that similar excerpts will produce similar hash keys to one another. If this assumption holds, then these hash keys can serve as a robust audio feature for use in distance measurements and so forth. To take advantage of this characteristic of the hash keys, we cluster similar has keys together and then analyze the occurrence of members of these clusters over time looking for patters.

3.1 Clustering Feature Vectors

The feature vectors (hash keys) were clustered using standard k-means clustering with varying numbers of cluster centers, which were initialized with randomly chosen feature vectors. Since the hashing window need not treat time and frequency evenly, some sort of normalization is needed before clustering to ensure the even spreading of the cluster centers in feature space. We chose to normalize the features to all have the same variance, although direcly taking into account the size of the peak-finding window might serve the same purpose. The variance normalization was chosen to avoid discretization artifacts when there aren't many FFT frames or frequency channels in the peak-finding window.

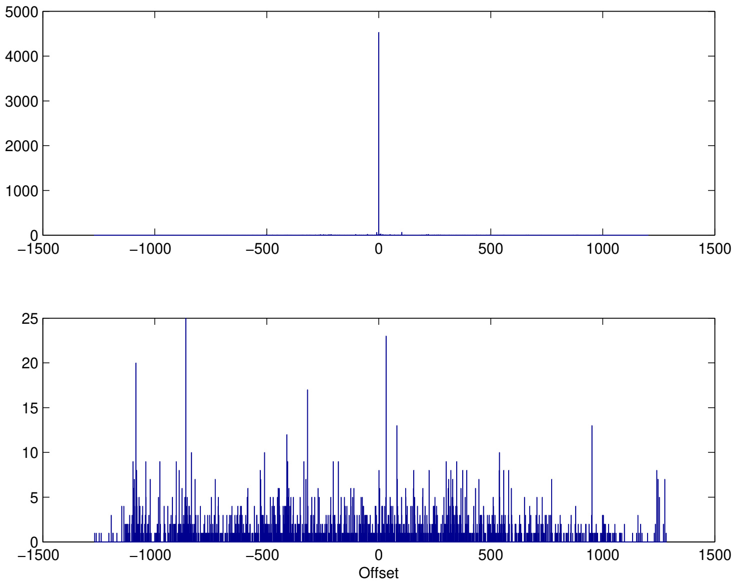

Figure 2: Histogram of distances between matching hash keys for matching original music with 0 dB white noise added to the same music (top), and unmatching music (bottom). Note the differences in scales of 2 orders of magnitude.

3.2 Finding Patterns

Once the feature vectors have been clustered, we may look for patterns in their co-ocurrence. Henceforth, the cluster with which the hash keys are associated and the times at which those keys will be treated as the features, creating one feature dimension per cluster center and one measurement per STFT "frame." This matrix of feature occurrences versus time is the co-occurrence matrix. Principal components analysis (PCA) can provide a lower rank approximation to this matrix. Since there can never be a negative number of ocurrences of an event, nonnegative matrix factorization (NMF)[2] is even more appropriate for this approximation.

4 Results

Preliminary tests do not show very good generalization ability of shazam's features. We were able to produce sounds with varying amounts of information removed, including first sound around the peaks, then the phase and magnitude, and then only using the peaks from the clustered hash keys instead of the originals. In all of these cases enough of the information from the original song is preserved to resemble it to some degree and be recognizeable (perhaps with slight prompting).

Testing the performance of the shazam system on its original task, identifying sounds through noise, indicated success on identifying sounds through a signal-to-noise ratio of greater than 0 dB. Also, on the toy data set at http://www.ee.columbia.edu/ dpwe/digits, the system was able to successfully identify practically all of the example sentences through the provided distortions.

5 Conclusions

The shazam representation for audio is very well-suited to the task of retrieving sounds identical to those on which it was trained through gross distortion. Unfortunately, it is harder to generalize this representation to matching similar audio than originally expected. Improvements in the peak-picking procedure might improve robustness by finding the perceptually most relevant onsets in the sound, although the current scheme works reasonably well.

5.1 Future Work

There are many ideas from the computer vision literature that could be applied directly to the problem of finding interesting parts of a spectrogram, finding the relationships and relative positions of these parts, and comparing such "constellations" of parts. Most notably, Harris feature detectors are used in machine vision to find "interesting" points in images for the purposes of focusing computational effort. Non-maximal suppression ensures that these points cover an image well without clumping together in the same area. Pietro Perona's work on image feature constellations[3] uses an entropy-based feature extractor, then learns the relationships between these features using EM, and finally can perform recognition of new images using a Bayesian classifier.

References

[1] A. Wang. "An Industrial Strength Audio Search Algorithm." Unpublished paper.

[2] P. Smaragdis, J.C. Brown. "Non-negative Matrix Factorization for Polyphonic Music Transcription," IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp. 177-180, October 2003

[3] R. Fergus, P. Perona, and A. Zisserman. "Object Class Recognition by Unsupervised Scale-Invariant Learning." Proc. of the IEEE Conf on Computer Vision and Pattern Recognition 2003