Computer Vesion Based Music Information Retrieval

Philippe De Wagter
pdewagte@andrew.cmu.edu

Quan Chen
quanc@andrew.cmu.edu

Yuqian Zhao
yuqianz@andrew.cmu.edu

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

This project relies on computer vision techniques to build a practical music retrieval system. Our approach tackles traditional music identification as a corrupted sub-image retrieval problem from the 2-D spectrogram representation of the original songs. More specifically, a query snippet spectrogram is matched against our database using its descriptor representation.

We utilize a novel pairwise boosting method to learn a robust and compact subset of Viola-Jones filters. This subset of filters captures distinctive information of each spectrogram while being resistant to distortion. By applying these filters, we can transform a spectrogram into a low-dimension binary representation. During the query phase, we search for all the candidates that locally match the descriptors of the query snippet. Then, we select the most likely candidate using an occlusion model based on a 2-state HMM. In order to test the performance of our music retrieval system, we rely on a case study: the Reggae music genre of the iTunes Store. The recognition is both fast and accurate, even with noisy background and a short song snipped as query.

We believe that this project can have a significant impact on the way people organize and research music. Indeed, this system tags songs automatically, adding music meta data such as album, artist, genre, and art cover. Furthermore, this project also opens up new avenues for live music search and identification.

1 Introduction

Traditionally, music identification has little to do with computer vision as the former focuses on the processing and recognition of 1-D signals over time while the latter concentrates on 2-D images. However, with the popular use of spectrogram(actually a 2-D image) in audio research, solving this problem in 2-D using computer vision techniques becomes a rising and effective way. In our implementation, we apply computer vision techniques like boosted classifiers and local-descriptor based object recognition to these images and receive impressive results.

The goal of music identification is to quickly and correctly recognize a song given a short and usually noisy sample of it. For example, if one wants to identify the name of a song he is listening to, he could use a recording device, such as a smart phone, to record a snippet of that song and then sent it to the retrieval server. After a few seconds of transmission and processing, the server would send him back a text message with the name of the song. However, this real-world application is

quite challenging. Indeed, most of the time, the query sample represents only a small portion of the original target song. So, before asserting what the original song is, we need to determine to which part of the song the query snippet corresponds. In addition, the snippet can often be distorted due to the low quality of the recording device and the background noise.

As we convert music identification into an equivalent corrupted sub-image retrieval problem, the goal is now to find the portion of a spectrogram image from our database that best matches the spectrogram of the query snippet. When a query snippet is received, we compute the local descriptors of its spectrogram. Then, we query our database of original songs to find out which songs shares descriptors with the query snippet. Once we have a subset of good candidates original songs, we compute the likelihood that the query snippet comes from each original song. Finally, the song which is most likely is identified as the target original song, and this song is return back to the user.

In the subsequent sections, we explain how we realize this procedure. More specifically, section 2 addresses why spectrograms are a good representation for music identification and how we can express a robust representation of these spectrogram by applying filters. Section 3 describes how we select the ”best” filters for our task using a machine learning algorithm: the pairwise boosting. Section 4 details the occlusion model we use to compute the likelihood of a match between a query snippet and an original song. In section 5, we detail the whole process of retrieving a song using our system. Finally, in the section 6, we present our implementation and the results we have obtained.

2 Audio Representation

Given a short noisy song snippet, our goal is to find the matching undistorted audio segment in the database. For this, we need an audio representation as descriptive as possible in order to be able to differentiate songs. However, we also want the representation to be resistant to distortions caused by a noisy background or a poor-quality recording device. In addition, since the query is only a short snippet of the whole song, the representation should be robust to small shifts in time.

The traditional time-amplitude representation does not satisfy all these requirements. For example, we can hardly see any explicit similarities between an original song and its recorded version (see Figure 1a). After applying the FFT to them, we obtain their spectrograms showing the power carried by 33 logarithmically-spaced bands over time, measured by 0.372s windows with 11.6ms increments (see Figure 1b). With the spectrogram representation, we can discover some visual similarities between spectrogram from the same audio source.

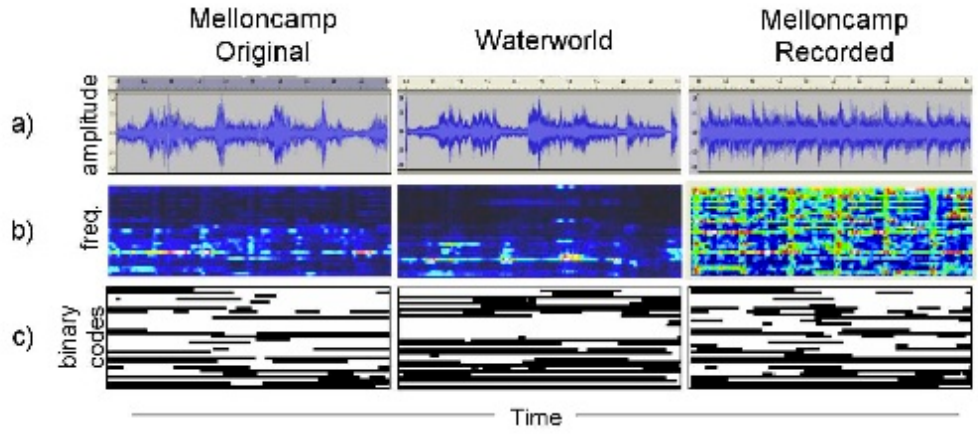

Figure 1: Represneting Audio

However, simply comparing spectrograms is not efficient because of the noise. Ao, we further transform a spectrogram into its binary representation by applying on this spectrogram a compact set of Viola-Jones filters (see Figure 2). The merit of these filters is that they can preserve important distinctive information (essential for distinguishing a song from similar songs) of each song, such

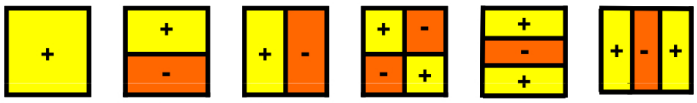

Figure 2: Voila Jones filter set

as differences of power in adjacent frequency bands, differences of power across time in a certain band, peak of power across bands at a particular time, and etc, while being resistant to distortions. The basic candidate filter set includes 6 types and each type can vary in location within a frame, bandwidth and time, so we have a total of about 30,000 candidate filters. Through a novel machine learning technique (pairwise boosting), we have selected 32 filters (and also 32 corresponding thresholds) for the use of our system. By applying these filters to each frame of a spectrogram, and by comparing the corresponding value with the respective threshold of every 32 filters, each frame now can be represented as a 32-bit vector in which each bit stands for the filter response from filter 1 to 32. This 32-bit vector is called as the descriptor hereinafter. As a result, a spectrogram can be transformed into a series of descriptors, which we call the signature (see Figure 1c). The signature of a spectrogram image or audio segment, is the basis unit for song retrieval in our system.

3 Filter Selection

In order to get the most distinguished representation of music, we need to select the filters who can give the best description. The goal is to select a small number of filters from the 29829 filter candidates to locate snippets from a potential distorted song on its original version in our database. So we can learn our weak classifiers:

$$h_m : Data(x_{ori}, x_{rec}) \longrightarrow \{-1, 1\}$$

where x_{ori} and x_{rec} are respectively the original-song and recorded-song samples.

The algorithm we use is based on the classic Adaboost, by which weak classifiers are learned and all the data is re-weighted. However, in our application, there are two main reasons stop us from using Adaboost:

1. It is a multi-class boosting, which means the two pieces of data should be passed as input to get either the output $y = +1$, meaning that the two songs are the same, or $y = -1$, meaning that the two songs are different.

2. It can cause a negative training case possibility of less than 50%. Indeed, if we independently and randomly draw two non-matching examples x_{ori} and x_{rec}:

$$P\left[h_m\left(x_{ori}, x_{rec}\right) = -1\right] = 2p\left(1 - p\right) \leq 0.5$$

So we use a variant of adaboost, called pairwise boosting. The basic idea of pairwise boosting is to judge if the the members of a pair x_1 and x_2 fall to the same or different sides of the threshold, which means they belongs to the same song or not, respectively. And in the re-weight part, both the positive data and negative data are normalized into 50%. The algorithm is detailed in the box below.

Pairwise Boosting input: sequence of N song pairs and T filters $\langle(x_{11_{ori}}, x_{11_{rec}})\rangle$... $\langle(x_{nt_{ori}}, x_{nt_{rec}})\rangle$, each song pair with class $y_i \in \{-1, 1\}$
initialize: $\omega_{ij} = \frac{1}{N}, i = 1..N, j = 1..T$
for m = 1..M

1. find the hypothesis $h_m(x_{m_{ori}}, x_{m_{rec}})$ that minimizes weighted error over distribution ω, where $h_m(x_{m_{ori}}, x_{m_{rec}}) = sgn\left[\left(f_{m(x_{m_{ori}})-t_m}\right)\left(f_{m(x_{m_{rec}})-t_m}\right)\right]$ for filter f_m and threshold t_m
2. calculate weighted error: $err_m = \sum_{i=1}^{n} \omega_i \delta\left(h_m(x_{im_{ori}}, x_{im_{rec}}) \neq y_i\right)$
3. assign confidence to h_m : $\alpha_m = 0.5log\left(\frac{1-err_m}{err_m}\right)$
4. update weights $\omega_i = \omega_i \exp(-\alpha_i)$
5. normalize weights: $\sum_{i:y=1}^{n} \omega_i = \sum_{i:y=-1}^{n} \omega_i = \frac{1}{2}$

Finally, we learnt 32 filters from the pairwise boosting. We observe that filters with small time-dimension width have a large band-dimension width. This means that filters cover a large extent in the same snippet and can therefore average noise and distortion. These learnt filters either measure the differences between the two frequency or a peak across the frequency domain that the particular snippet. Finally, these filters are highly robust to noise and distortion.

4 Occlusion Model

We have built a simple occlusion model to evaluate the likelihood for a given query signatures to be generated by a particular original song. In the query snippet, the frames in the spectrogram are either generated by the song, or by an occlusion. An occlusion can be for instance some one talking while the music is being recorded. any descriptor we observation we observe can be generated either from the state music, or from the occlusion state. Intuitively, we can understand that if the recording was generated from the occlusion in the frame i, it is very likely that the recording will also be generated from the occlusion in the frame $i + 1$.

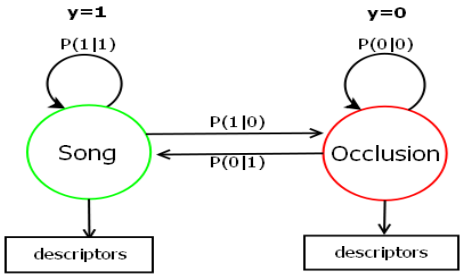

Figure 3: HMM status diagram

The sequence of observations for which we want to compute the likelihood using our HMM, is the bit-difference between the query snippet and the original song. This means that if the song was not distorted at all, the bit-difference would be made of 32 zero bits. Moreover, we model each bit as an independent Bernoulli random variable.

So, we have to define 32 parameters for the emission probabilities in the state song, and another 32 for the emission probabilities in the state occlusion. We also need to define 2 parameters for the transition probabilities, and another 2 parameters for the initial probabilities. This means we have to define $32 + 32 + 2 + 2 = 68$ parameters for our model.

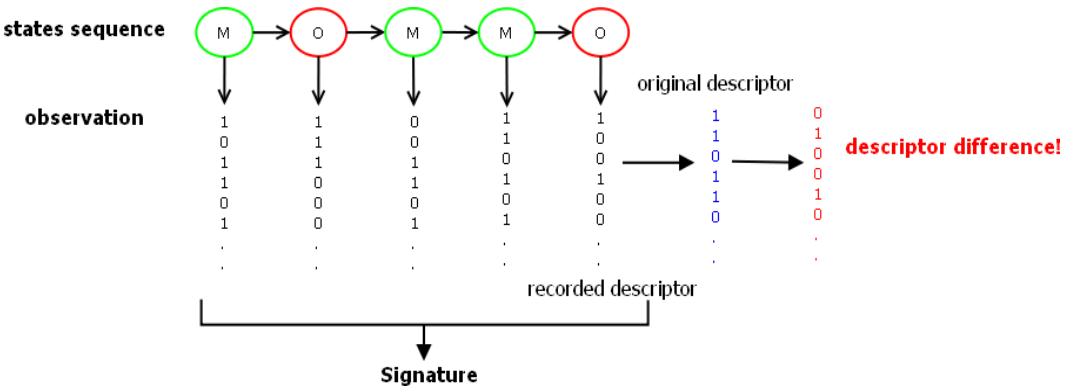

Figure 4: Descriptor observation status diagram

5 Song Retrieval

Using the filters and their thresholds learned from the pairwise boosting, we build the signatures of all the original songs in our database. To implement the retrieval process, we build two hash tables. The first one is keyed by all individual descriptors present in our songs and maps to a list of frames, where a frame consists of a pair ¡Id of the song, offset of the descriptor in the song¿, values. The second table is keyed by by the song Ids, and it stores the whole signatures of the corresponding original song.

When a query is received, our system follows the steps below:

1. Probe the first table for the first descriptor of the query signature to obtain a list of all containing with this descriptors.
2. Repeat step 1 for every descriptor in the query snipped.
3. Extract all the frames from the lists, and count how many times a song id appears.
4. Choose the top 10 song ids which appear most frequently. These 10 songs have the most descriptors in common with the query among all the original songs.
5. Since we know for the 10 best candidates all the offset values from step 3, we can align the entire query signature to a snippet of the song signature according to the offset.
6. Compute the likelihood that the query signature is generated by the aligned partition of the song.
7. Repeat step 6 for all possible alignments of the 10 song ids.
8. Choose the song id which is the most likely according to our occlusion model.

6 Inplementation

6.1 Data Preparation

To create the database, we have written a java program able to automatically download the songs preview from the iTunes Store. More specifically, this program contains 100 threads downloading in parallel the songs along with their meta-information (cover art, album name, artist name, etc) for the Reggae style, which represent a $100,000$ of songs.

The next step in the data preparation part is the transcoding from m4a to wav format. For that, we have written a VLC script.

Since, the transcoding was so slow, we had to restrain our final dataset to $22,000$ of songs.

Concerning the training set, we have prepared ten distorted songs, consisting of the superposition of the original song, and one local radio from Pittsburgh to act as random background noise.

6.2 Pairwise Boosting

For each original song in our training set, we create a recording version to pairing it by adding radio noise. The original song name format is #_.wav, and its recording pair name format is #_#.wav, where # should be replace by an actual song id.

For each original song, we create ten pairs, the first five are from this original song adding noise, the last five are from other distorted song. Then, we choose different part of the radio to add noise to make sure the background noise vary across the training set. For the negative samples, we choose distorted songs randomly across the training.

The next step is to apply some signal processing techniques on each of these songs. First, we convert a song into a single-channel and downsample it to 5512.5HZ. Then we apply the FFT on it with an analysis window of 0.372s (2048 samples) and a hop size of 11.6ms (64 samples) to get float samples. Then we add all candidate filters at a fixed snippet of the song and get the a set of data for pairwise boosting.

After the pairwise boosting, we get the 32 filters with its threshold and weight. The output file contains the filter id, the threshold, and the weight, for each of the 32 filters.

6.3 HMM

We have implemented the Baum-Welch algorithm to train our HMM on the same training data as for the pairwise boosting. However, we have found that an intuitive occlusion model where the probability of 0 and 1 is 50% in the occlusion state, while it is respectively 80% and 20% in the song state, leads to approximatively the same results.

7 Results

To test our implementation, we query 100 distorted audio snippet to retrieve the original out of our $22,000$ songs database all extracted from the Reggae genre of the iTunes Store.

When we were using a 30 seconds snippet, we were able to correctly identify 99 songs among the 100, just by choosing the song with the most descriptors in common, and without using the occlusion model at all. This result proves the quality (both robustness and discriminative power), of our set of descriptors we learned.

To push even further our system, we account only for only 100 frames. Since the hop size is $11.6ms$, and since the descriptor computation spread across 100 frames, we actually use 2 seconds of audio to query our our system. This time, using the HMM to compute the likelihood of each observation, we obtain 95 correctly identified songs, while 5 songs are evaluated as unlikely to be generated by any of our original songs.

This results shows how computer based vision techniques can be used to efficiently tackle the problem of song identification. The next step we are working on is to extend this algorithm to the entire iTunes Store music library, and provide a desktop application able to automatically identify and tag songs in everyone's music library.

Acknowledgments

We want to express our thanks to Prof. Bhiksha Raj, Prof. Rahul Sukthankar and Yan Ke for their valuable guidance during our project.

References

[1] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Proceedings of Computer Vision and Pattern Recognition, 2001.

[2] Y. Ke, D. Hoiem, and R. Sukthankar. In Proceedings of Computer Vision and Pattern Recognition, 2005.

[3] FFTW http://www.fftw.org/

[4] LibTag http://developer.kde.org/ wheeler/taglib.html