Poster: Audio Fingerprint Application for the Media Industry

Andrew Putra Kusuma
Nanyang Technological University
50 Nanyang Avenue,
Singapore 639798
AKUSUMA002@e.ntu.edu.sg

Owen Noel Newton Fernando
Nanyang Technological University
50 Nanyang Avenue,
Singapore 639798
OFERNANDO@ntu.edu.sg

Vajisha U. Wanniarachchi
Nanyang Technological University
50 Nanyang Avenue,
Singapore 639798
Vajisha.uw@ntu.edu.sg

Ng Wee Keong
Nanyang Technological University
50 Nanyang Avenue,
Singapore 639798
AWKNG@ntu.edu.sg

UbiComp/ISWC'18 Adjunct, October 8–12, 2018, Singapore, Singapore

ACM ISBN 978-1-4503-5966-5/18/10.
https://doi.org/10.1145/3267305.3267615

Abstract
Advertising and media industry has shown rapid growth in past few decades by aligning with the increased popularity of mobile phones. As a result, advertising firms tend to try new technologies to capture the target audience attractively. Since its method of identification is independent of the application, audio fingerprinting has been used for various purposes including content-based audio retrieval. This project aims to demonstrate one potential application of audio fingerprinting algorithm in the media industry in the form of a mobile application which is able to detect and identify advertisement tracks and notify the user of related details and offers. The audio fingerprinting algorithm extracts different attributes from an audio file, processes them into audio fingerprints, and compares them with a database of audio fingerprints to find the closest-matching audio file. The proposed application has tested on a small-scale database and shown a respectable performance.

Author Keywords
signal processing; algorithm; mobile application development.

ACM Classification Keywords
H.5.1. [Information interfaces and presentation]: Multimedia Information System – Audio Input/Output;

H.4.m. [Information Systems Applications]: Miscellaneous.

Introduction

The main objective of the audio fingerprinting algorithm is to generate a digital summary of an audio signal which can be used to correctly identify similar audio files in a database [1]. A slight variation that occurs in an audio signal (such as noise and compression) will drastically change its binary representation; hence, bitwise fingerprint algorithm (e.g. checksum) is impractical. Many existing audio fingerprinting techniques exploit perceptual characteristics (such as average spectrum and prominent tone) of audio signals in their algorithm such that insignificant variations to the signal can be tolerated.

Audio fingerprinting has been used in practice for song identification and copyright monitoring [2]. Additionally, audio fingerprinting also demonstrates potential use in the media industry, particularly in advertising. However, there are still other possible uses of audio fingerprinting in the media industry—with potentially higher commercial value—yet to be explored.

This project aims to demonstrate one potential application of audio fingerprinting algorithm in the media industry in the form of an Android application which is able to detect and identify advertisement tracks and notify the user of related details and offers. The application is tested on a small-scale database and assessed on its performance, in particular, the computation time and accuracy.

```
[8, 38, 4] [10, 0]
[8, 13, 5] [10, 0]
[8, 35, 5] [10, 0]
[9, 13, 3] [11, 0]
[9, 38, 3] [11, 0]
[9, 13, 4] [11, 0]
[9, 35, 4] [11, 0]
```

Figure 1: Example of audio fingerprints

Background Theory and Development

In general, the audio fingerprinting algorithm can be grouped into three stages: analysis, hashing, and matching. In the analysis stage, different attributes (such as amplitude, frequency, and time) are extracted from an audio file, which is organized into useful information. Then, in hashing stage, this information is processed into audio fingerprints or unique hashes to the audio file. Finally, in the matching stage, these hashes are compared with a database of audio fingerprints to find the closest-matching audio file.

Analysis

In the analysis stage, the audio format used for the input is PCM 44.1 kHz of unsigned 16-bit depth mono format, which is represented by a short array. FFT, an efficient implementation of DFT is used to convert the time-domain signal into a frequency-domain spectrum. The window function being used is the Hamming window, which is sufficient for many cases [3].

Since the audio will mostly be speech-based, the grouping of frequency bins into bands will emphasize more on the voice frequency range (i.e. 300–3400 Hz, which is the frequency band used in telephony [4]). The logarithmic width of the frequency bands is used to mimic the human ear's non-linear pitch perception [5]. For each band, the frequency with the highest amplitude is saved to obtain the "constellation map" (represented by a two-dimensional array).

The "constellation map" is then filtered using a sliding window filtering. For each window, the mean of every spectrogram peaks within the window are computed, and those peaks with lower amplitude than the mean or a certain threshold value (to remove ambient sounds)

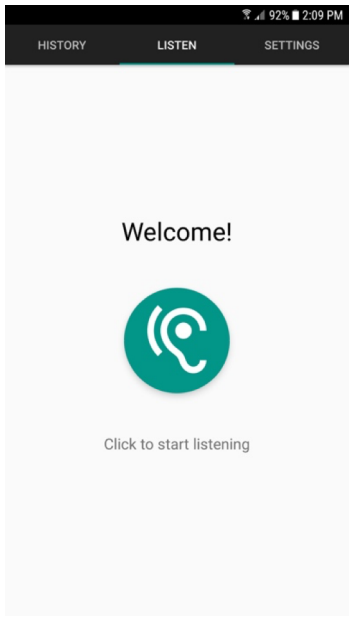

Figure 2: Welcome Screen of the Mobile Application

Attribute	Value
Average time taken (seconds)	6.181 ± 0.822
Recall	0.706 ± 0.223
Precision	0.814 ± 0.267

Table 1: Summary of Testing Results

will be filtered out. The final result is a filtered "constellation map" represented by a list of integer arrays.

Hashing

One of the techniques used to generate audio fingerprints from spectrogram peaks is combinatorial hashing—the technique used by Shazam [6] in its algorithm. During the hashing stage, an order relation between peaks in the spectrogram is necessary to be able to generate reproducible target zones.

After the spectrogram peaks are time-frequency ordered, anchor points are selected among the spectrogram peaks. An anchor point is defined as the strongest peak within each time frame (i.e. one FFT window). For each combination of an anchor point and a point within a target zone it is associated with, a hash can be generated using the format:

$$key: couple = [f_a : f_p : \Delta] : [t_a : n]$$

where f_a is the frequency of the anchor point, f_p is the frequency of the chosen point, Δ is the time difference between the anchor point and the chosen point, t_a is the absolute time of the anchor point in the audio file, and n is the unique ID of the audio file.

Target zones for every anchor points can be determined using two parameters: anchor distance (i.e. the distance between the anchor point and the point in the target zone with the lowest order value) and target zone size (i.e. the number of points in the target zone). Figure 1 shows an example of generated fingerprints.

Matching

After audio fingerprints have been obtained from the audio sample, the audio fingerprints will be sent to the server. To find a match in the database, firstly, for every address-couple pair obtained from the audio file, the algorithm searches the database for all possible couples such that their address is the same. After that, the offset for every matching pair is obtained by $\delta = t_a - t_a'$ where δ is the offset between the audio file and the audio in the database record, t_a is obtained from the matching fingerprint in the database, and t_a' is obtained from the fingerprint. For each audio ID appearing in the couples, the largest value of appearance of offsets (i.e. the maximum number of matches that are time-coherent) is kept. The audio ID with the largest number is the matching record.

Testing

The mobile application records audio samples using the phone's built-in microphone. Short, repeated recordings are taken continuously with the length of 2 seconds per interval. The recording process stops when a match is found, or no matches are found after 5 cycles of recording (i.e. 10 seconds). In order to test the robustness, recordings are done in a noisy environment, which is simulated by playing an audio record of road traffic noise [7] alongside the advertisement being tested. The audio fingerprint database contains 36 advertisement records of varying lengths. Parameters being tested are time taken to find a match, as well as precision and recall. Tests are done by running each record in the database 10 times with arbitrary starting time.

Results and Discussion

The testing results are summarized in Table 1. The results show that the mobile application demonstrates respectable performance in terms of speed and accuracy. Despite being tested in a noisy environment, the algorithm still managed to obtain F-measure of 0.756. The average time taken is also satisfactory, with a low degree of variation.

One observation found is that relatively high variance was noticed in the precision and recall. Two possible causes are: the sub-par audio quality due to low source video bit rate and sampling frequency; and unsuitable parameters, particularly the ranges of the frequency bands and the FFT window size, due to the characteristics of the audio track. Low performance is observed when the advertisement being tested either has a low dynamic range of frequencies—which causes not much information obtained—or is more music-heavy with a relatively high pitch.

Future Work

The mobile application developed for this project currently runs on a small-scale database. In the future, the mobile application can be tested running on a large-scale remote database. One possible extension of the algorithm is to utilize ultrasound region (i.e. 20–22 KHz), which in theory should work better due to lesser noise. Moreover, tweaking algorithm parameters in accordance to its practical usage may also improve the performance in terms of precision and recall.

Conclusion

A robust and scalable audio fingerprinting algorithm has been implemented in this project. The algorithm makes use of prominent notes in the audio signal and their correlations to generate digital fingerprints of the signal. The algorithm is demonstrated in a functional Android mobile application, which also shows the use of the audio fingerprinting algorithm in the advertisement industry. The mobile application has been tested and displays respectable performance (in terms of speed and accuracy) and robustness against noise.

References

1. ISO/IEC TR 21000-11:2004 Information technology – Multimedia framework (MPEG-21) – Part 11: Evaluation Tools for Persistent Association Technologies.
2. A. Abajian, "Method for identifying copyright infringement violations by fingerprint detection", US20030061490A1, 2003.
3. National Instruments. (2016, Dec. 30). Understanding FFTs and Windowing. [Online]. Available: http://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf.
4. P. Burk, *Music and computers*. Emeryville, CA: Key College Pub., 2005.
5. Huawei Enterprise. (2014, Sep. 24). *Technical White Paper for Videoconferencing Audio Protocols*. [Online]. Available: http://e-file.huawei.com/en-US/marketing-material/onLineView?MaterialID=%7B4687CF1A-7C56-4FF8-AA0B-AEA7D121A724%7D.
6. C. Kalenzaga, "How does Shazam work", *Coding Geek*, 2015.
7. TrafficNoiseDanger. (2009, Nov. 16). Traffic Noise in India [Video file]. Available: https://www.youtube.com/watch?v=s3BzHO0mT5s.