Soundz

ACOUSTIC FINGERPRINTING FOR MUSIC IDENTIFICATION

July 6, 2018

Jake Runzer

University of Victoria

CSC 461 Project Report

Contents

List of Figures

1 Introduction

Audio has become an increasingly popular form of multimedia. Fast and available Internet allows anyone to stream music to their devices or place of business. Hearing a new song you like while on the go is now common. Fortunately, the rise of smartphones have lead to mobile applications that can quickly and accurately identify an unknown song with only a few seconds of audio. These applications use audio fingerprinting techniques to create short compact summaries belonging to known audio sources. These summaries have many uses, one of them being music identification. Other applications of audio fingerprinting include copyright detection and broadcast monitoring.

In this report I cover how several audio fingerprinting algorithms and music identification systems work. I have also implemented a small program which demonstrates identification using peak-pair hashing, a process I explain in detail.

2 Audio Fingerprinting

Audio fingerprints are compact signatures that summarize the audio signal they were generated from. Some techniques of fingerprinting produce a single unique representation of the audio. Others produce hundreds or thousands of small representations that when looked at individually are not unique, but the collection uniquely corresponds to a single audio source. An important property that all fingerprinting algorithms used for music identification have, is that a fingerprint represents how humans hear the audio. The binary representation is not directly looked at when analyzing the audio. A fingerprint is not a hash of the audio file and a single bit flip or small distortion should not have a large affect on the generated result. This is very important when analyzing audio generated from a mobile phone in a public setting as noise or distortions are often introduced.

3 Music Identification

Music identification is the process of using a database of fingerprints belonging to known sources to identify a fingerprint belong to an unknown source. A database of fingerprints is generated using any audio that should be identified. When an identification needs to be made, the unknown audio sample is fingerprinted and used to query the database. A match indicates that the unknown audio sample originated from the matching source sample. This process can be see in Figure 1. A common application for music identification algorithms is allowing a user to record a short amount of audio on their mobile device and the source song is identified. Typically these applications require 5 - 15 seconds of audio for accurate identification.

The audio used to create the fingerprint database is normally from high quality and controlled sources with little to no imperfections. The audio source of the unknown sample that needs to be identified is unknown and the sample is often noisy and full of distortions.

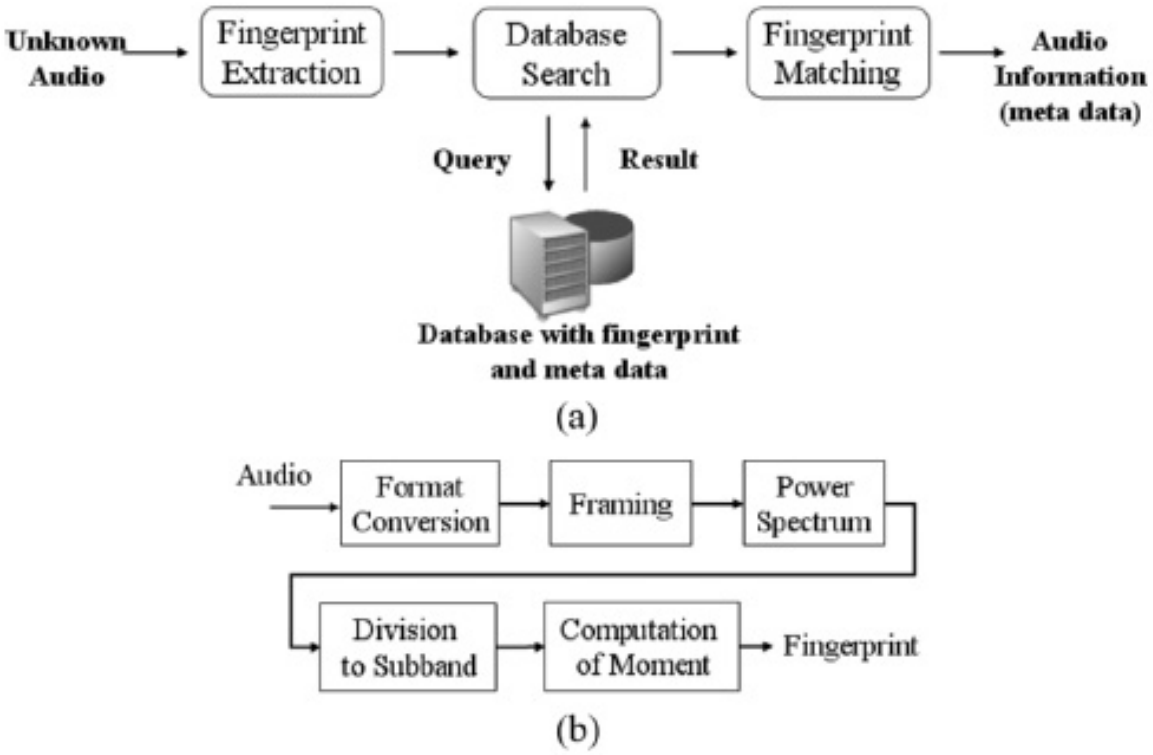

Figure 1: Example music identification process [1]

3.1 Requirements

Many music identification applications are deployed for use on mobile devices and will be used in noisy environments such as coffee shops and restaurants. High quality audio fingerprinting algorithms are robust to these distortions and at the same time respect the memory and computational limitations of a users device. Additionally, the identification process should take as short as possible, typically in the range of 10 ± 5 seconds.

3.2 Identification Pipeline

A general pipeline of mobile music identification is as follows

- Audio is captured on the users mobile device using the built-in microphone
- A fingerprint or set of fingerprints is generated on the device and sent to a matching server
- The unknown fingerprint(s) is used to query a database of fingerprints that all belong to know sources
- A matching step is performed on the results to select the most similar matches
- The best matched song is sent back to the users mobile device

3.3 Production Apps

Several apps that perform this identification are Shazam [3] and SoundHound [4].

4 Spectrograms

Spectrograms are a critical part of any audio fingerprinting algorithm. The algorithm needs a digital representation of the audio as it is heard by humans and spectrograms more closely represent this, especially compared to the raw binary representation.

A spectrogram is a visual representation of the spectrum of frequencies of sound as they vary with time. An example of a spectrogram of the song Kids by MGMT can be seen in Figure 2.

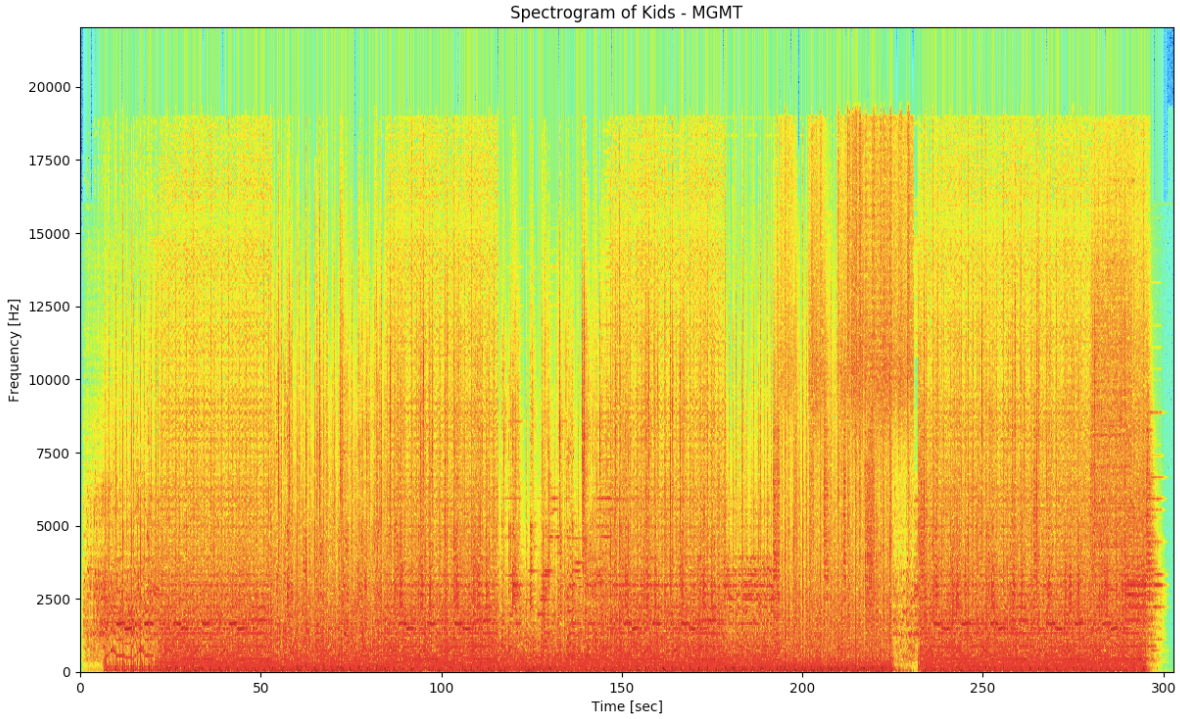

Figure 2: Example of a spectrogram for the song Kids - MGMT

4.1 Short-Time Fourier Transform

The spectrogram is created by using the short-time Fourier Transform (STFT). STFT takes overlapping windows of an audio signal in the time domain, and converts it to frequency domain using the Fourier Transform. For discrete data this transform is represented as [5],

$$\mathbf{STFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n}$$

where $w[t]$ is the window function and $x[t]$ is the signal to be transformed. The Fast Fourier Transform (FFT) is typically used. The spectrogram is the square of the magnitude of the STFT.

$$\text{spectrogram}\{x(t)\}(\tau, \omega) \equiv |X(\tau, \omega)|^2$$

4.1.1 STFT Parameters

Several parameters can be configured when generating the spectrogram. These are

- Window type
- Window length
- Overlap amount
- FFT length

The window type controls the window curve function, length controls the number of samples that are used in a single FFT calculation, and overlap controls how much each window overlap with the next. The FFT length controls the frequency resolution [6].

5 History

Music identification algorithms have been in development since the 90s but their popularity has increased exponentially in the last 5 years due to an increased number of smartphones and processing power. Several different successful ideas have been explored [7].

5.1 Computer Vision for Music Identification

In 2005, Ke *et al.* [8] represented the audio as an image and used computer vision techniques to create a unique fingerprint. The intuition in this algorithm is that a 1 dimensional audio signal can be analyzed as a conventional 2 dimensional image when viewed in the time-frequency spectrogram representation. The spectrogram is used to train a machine learning model to learn compact audio descriptions with the goal that the probability that two noisy or distorted descriptions which were sampled from the same position of the same song can be determined.

Adaptive Boosting classifiers [9] are trained using box-filters on the spectrogram. The output of the classifiers, a binary value, is concatenated and used as the fingerprint.

5.2 Waveprint: Efficient Wavelet-Based Audio Fingerprinting

A similar approach, in combination with data-stream processing techniques, was used by Baluja and Covell in 2008 [10]. Instead of a machine learning approach, *Waveprint*, uses wavelets to extract features from the audio spectrogram. Wavelets are a "mathematical tool for hierarchically decomposing functions" [10]. Their approach is as follows

1. Divide the spectrogram of an audio sample into smaller spectral images
2. Compute the wavelets on the spectral images
3. Extract the top wavelets, measured by magnitude
4. Create a binary representation of the top wavelets using their sign

In order to allow efficient nearest-neighbour indexing, Min-Hash [11] is used to create a more compact representation of the binary output. The result is a set of p bytes (typically > 100) that can be directly compared by computing the Hamming distance. In an analysis by Chandrasekhar *et al.* [7], this wavelet based approach to music identification performed the most accurately for 5, 10, and 15 second queries achieving $> 90\%$ accuracy after a temporal alignment step. However, this process requires significantly more memory and computational resources than the other methods for computing the fingerprints.

5.3 An Industrial-Strength Audio Search Algorithm

In 2003, Avery Wang, released a paper [2] detailing Shazam's ingenious approach to music fingerprinting. The algorithm only looks at high energy peaks in the spectrogram. Peaks are more likely to survive ambient noise as a peak analysis of music and noise together will contain spectral peaks due to the music and noise as if they were analyzed separately. Pairs of peaks are used as fingerprints for an audio sample, of which there can be thousands.

It is this algorithm I implemented for my project and will go into greater detail below.

5.4 Robust Audio Fingerprinting Using Peak-Pair-Based Hash of Non-Repeating Foreground Audio in a Real Environment

In 2016, Kim *et al.* [12], improved upon Wang's peak-pair algorithm by extracting non-repeating foreground audio from background audio. Background audio, for example, in an unknown audio sample would be the background chatter at a coffee shop. A modulated complex lapped transform (MCLT) is used instead of STFT to convert the 1D audio signal into a format suitable for peak extraction. Adaptive temporal thresholding is then applied to the high energy peaks computed from the MCLT spectrogram. Finally, pairs of nearby peaks are combined into a 32-bit fingerprint hash. This process can be seen in Figure 3. This algorithm is more robust to noise and echo conditions than the original one proposed by Wang.

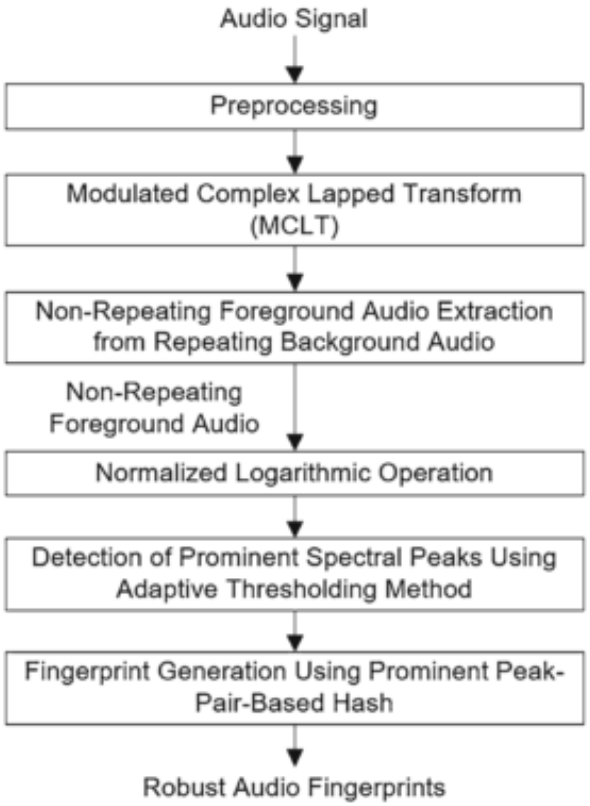

Figure 3: An improvement to Shazam's original peak-pair hashing identification algorithm

6 Implementation

For this project I have built a small program which demonstrates using peak-pair hashing for fingerprint generation and music identification. I make heavy use of the Numpy [13] and SciPy [14] Python libraries.

6.1 Architecture

The architecture of my implementation closely follows the music identification process explained above. However, instead of the unknown audio coming from a mobile device, it is captured through my computer speakers using a microphone. Capturing the audio through the microphone allows me to introduce artificial noise. This more closely represents a real-world application compared to using segments of the audio files that were used to create the fingerprint database. The architecture of the program can be seen in Figure 4.

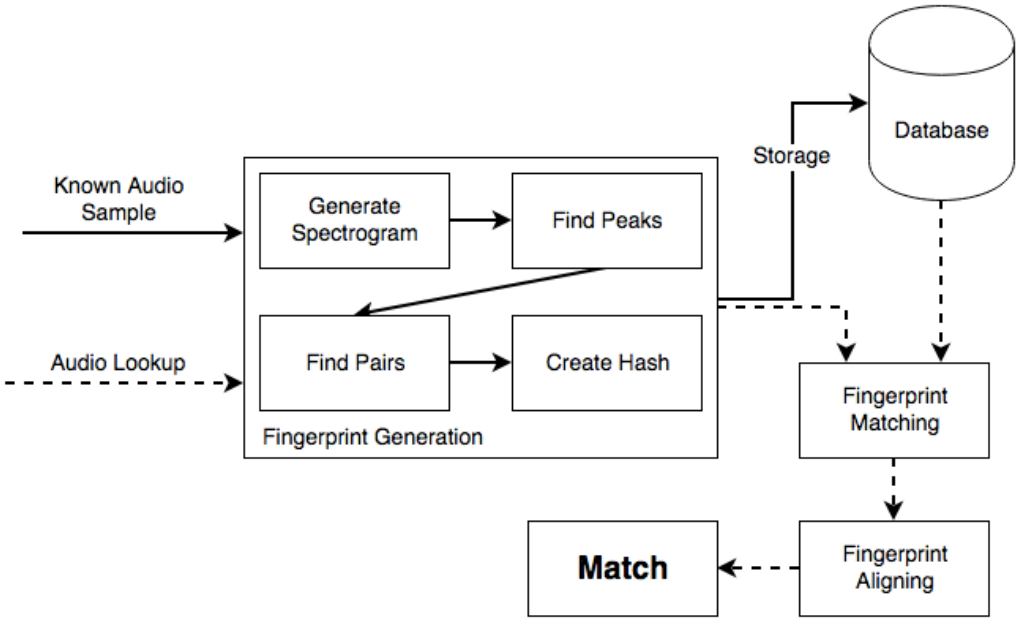

Figure 4: Architecture of demo program

6.2 Spectrogram Creation

To create the spectrogram I useed a Hamming window function with 1024 samples per window. The windows overlapped by 50% and I used 1024 bins for the FFT input. The spectrogram generated with these parameters can be seen in Figure 2.

6.3 Constellation Maps

Time-frequency peaks are found using an image local maxima filter with a neighbourhood of 15 pixels. Pixels corresponds to milliseconds in the time domain and hertz in the frequency domain. In Wang's original paper [2], the spectrogram peaks can be plotted to create a "constellation map". The constellation map for the song Kids by MGMT can be seen in Figure 5

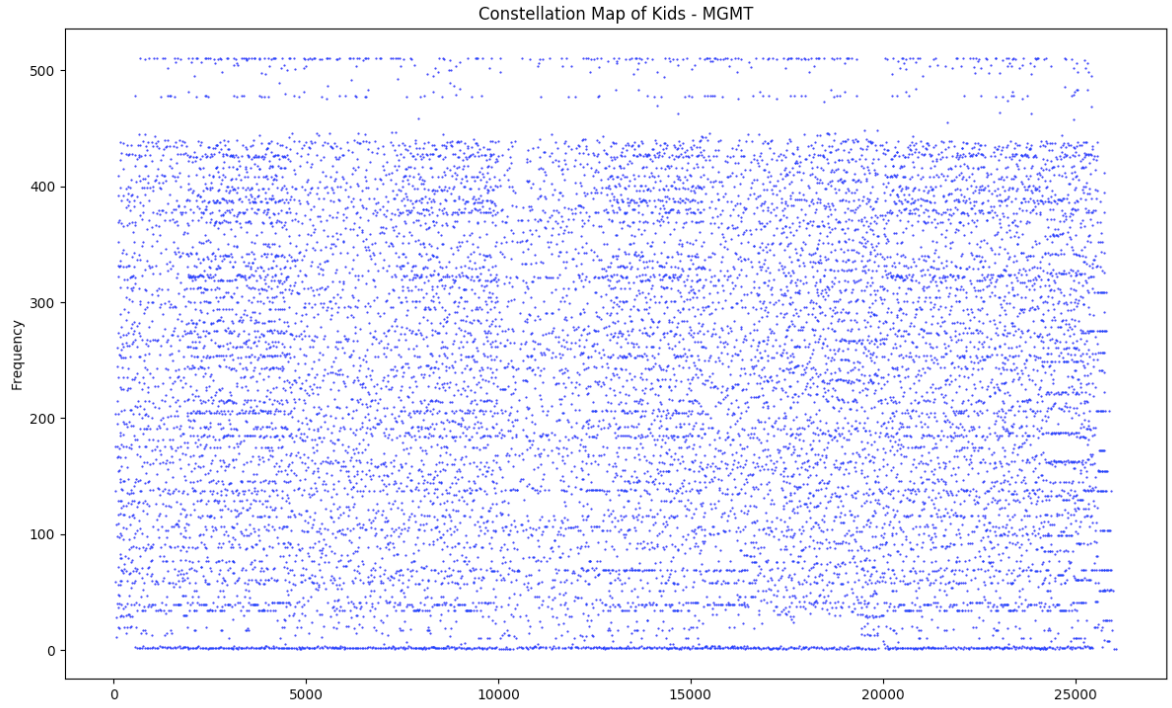

Figure 5: Constellation map of Kids - MGMT

The total number of peaks found depends on the song length and the amount of energy in the song. Generally, more upbeat and dance-y songs will have more peaks in them. In Figure 5 there were 14425 peaks found.

6.4 Finding Pairs

Pairs are found by looking at each peak and the closest 15 neighbouring peaks within 200 seconds. In Figure 5 there were 8514 pairs.

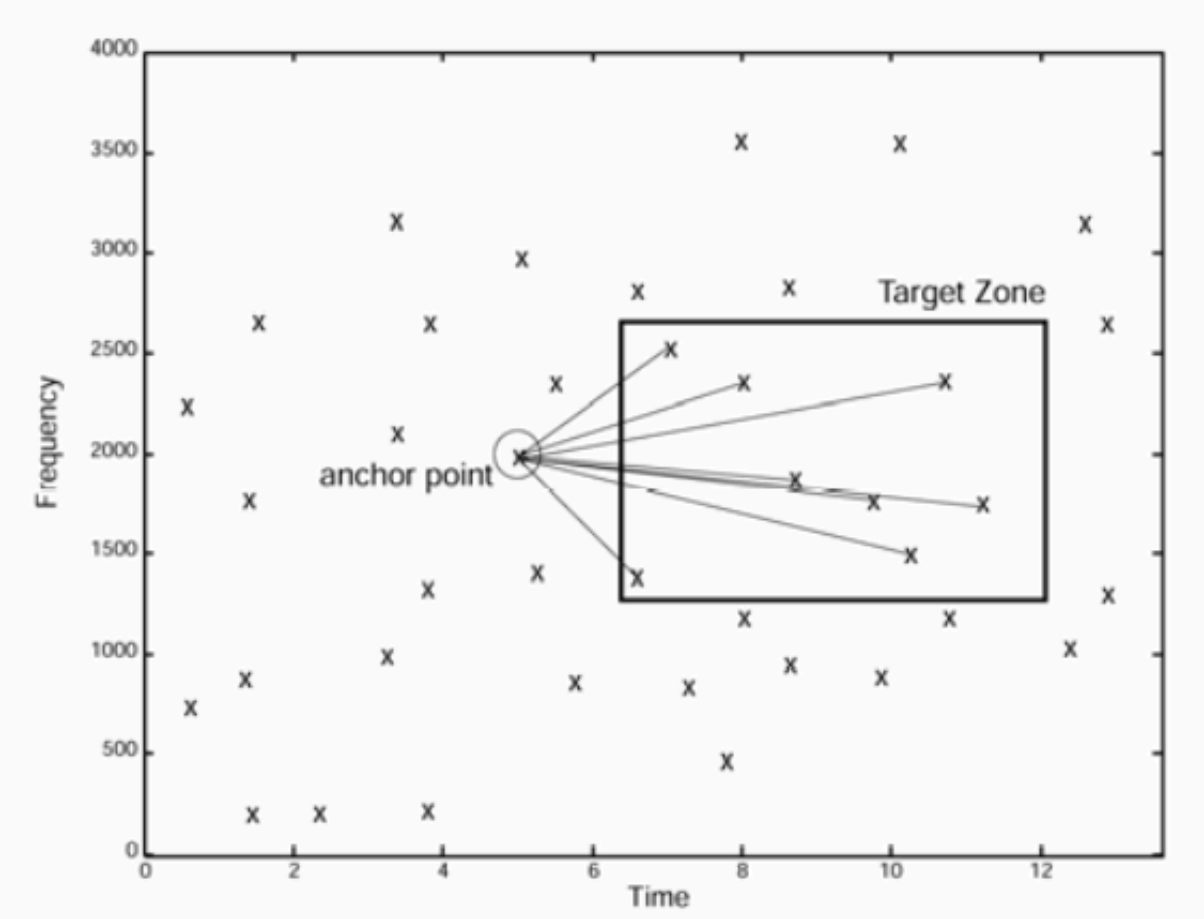

Figure 6: Finding pairs in a constellation map [2]

6.5 Creating Hashes

A fingerprint hash is create for each pair of peaks. It should be noted that these are not cryptographic hashes. Each hash is composed of the frequency of the first point, the frequency of the second point, and the difference in times. The hash is combined with the time offset of the first point, as it will be used in the fingerprinting aligning step, to create a single fingerprint.

In Wangs paper, each fingerprint was a 32 bit integer as this was a convenient way to store them.

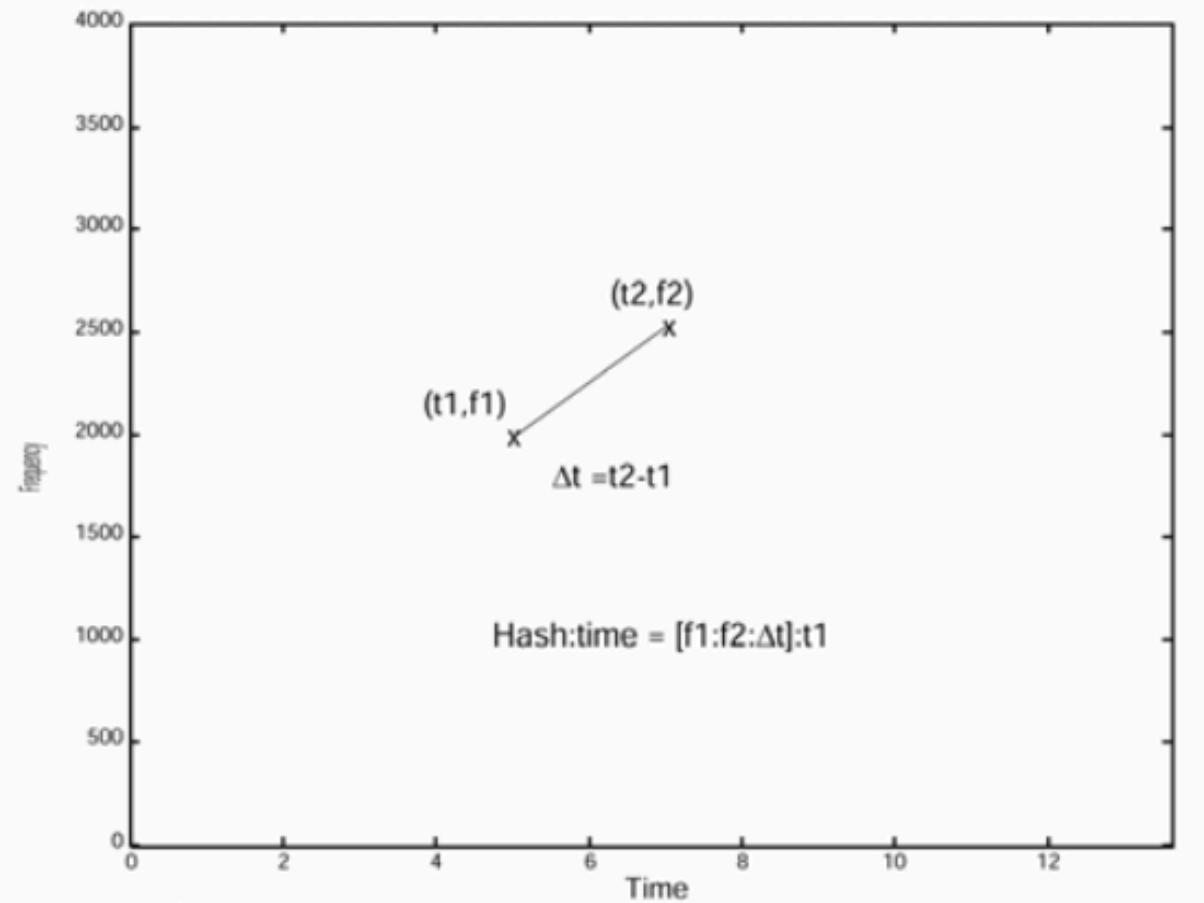

Figure 7: Creating a fingerprint from a peak-pair [2]

6.6 Database

Storing the fingerprints that belong to known audio sources is a critical step in a production music identification system. Fingerprints for potentially millions of songs need to be efficiently stored and indexed for fast lookup. In my implementation speed was not required, so I used a simple PostgreSQL database with a table that held information about each source song, and a table that held information about each fingerprint. Each fingerprint holds a relationship to its source song.

6.7 Identification

Unknown audio samples that need to be identified are first fingerprint in the same process as above. The set of fingerprints generated is used to query the database and matching fingerprints are retrieved.

6.7.1 Fingerprint Alignment

A naive approach to selecting the best matching source song could be to select the song with the highest number of matching fingerprints. However, this does not take into account the order in which the fingerprints occured. We cannot know the time offset the unknown audio started recording at, but we do know the order and time offsets that the fingerprints occured relative to each other. This information is used to find matched fingerprints that occur successively after each other in with the same relative offsets. This is computed by simply subtracting the time offsets originating from the known fingerprints by the time offsets originating from the unknown fingerprints. This can be visually seen in Figure 8, where a diagonal is present when looking at the matching fingerprint time offsets. A peak in the histogram of time differences indicates a matching song.

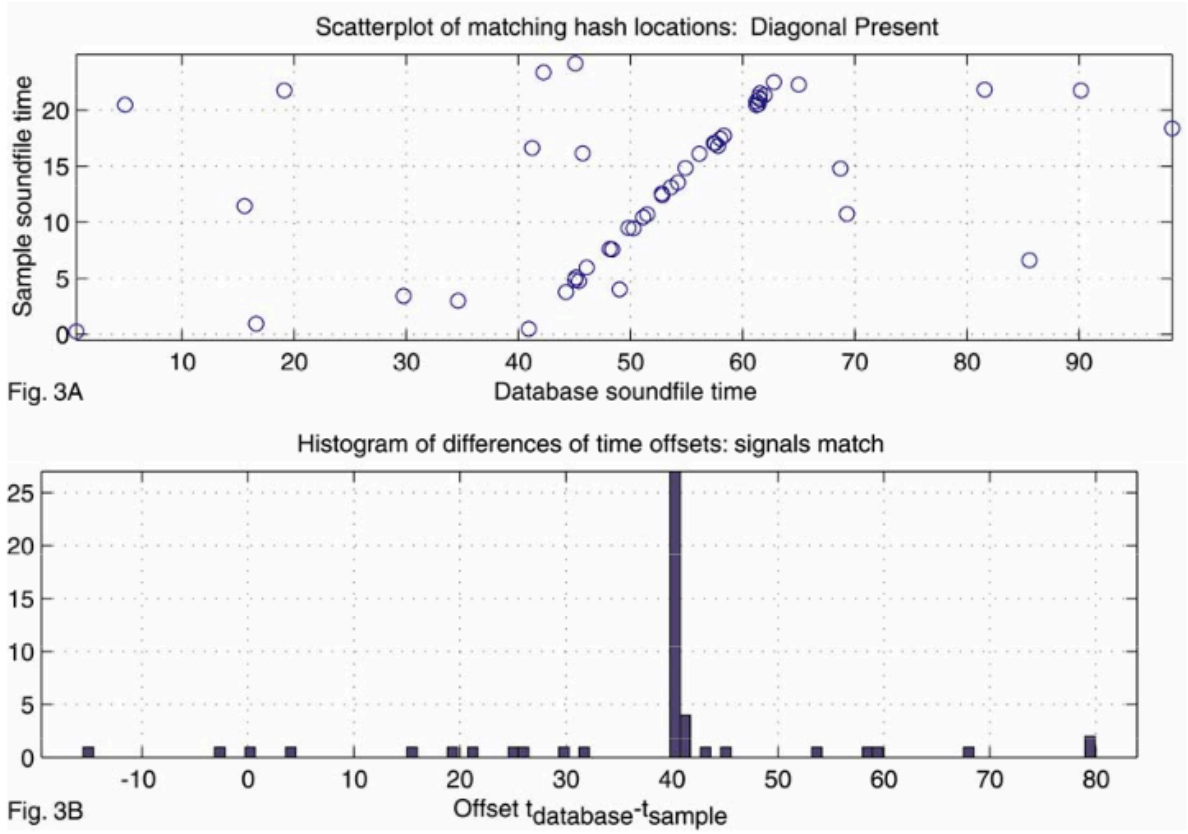

Figure 8: Aligning fingerprints by analyzing fingerprint time offsets [2]

6.8 Results

Music identification systems are notoriously hard to measure and test. The accuracy of the system is affect by the length of the query audio, the offset from which the audio started from, the quality of the query microphone, the quality of the source songs, and the content of the songs. For this reason there is no standardized way to compare identification algorithms.

For my implementation I performed manual testing using a database of 87 songs with 591094 fingerprints belonging to all these songs. The source audio is all 320 Kbps MP3 files with a 44000 Hz sample rate and I choose songs from a variety of genres. The queried audio is recorded using a standard webcam microphone positioned 5 feet away from the output speaker. I queried 30 songs recorded at 5, 8, and 10 seconds and started recording at random times in the song. The results can be seen in Figure 9.

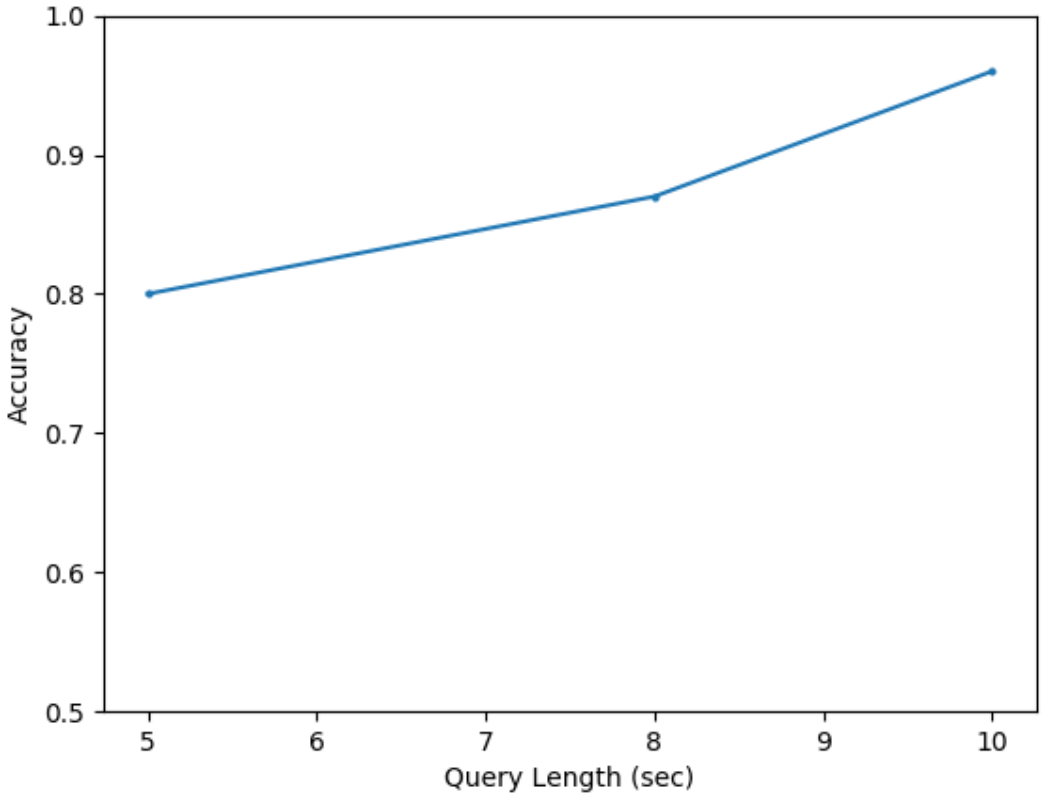

Figure 9: Accuracy Identifying 5, 8, and 10 seconds of audio over 30 trials

This test is far from robust but shows that the algorithm works. When 10 seconds of

audio is queried, mis-identifications are normally due to the audio being recorded from a very quiet or uncommon part of the song.

7 Conclusion

Audio fingerprinting algorithms create compact signatures of an audio signal that represent how humans hear it. Fingerprints can be used in a music identification service, which stores large amount of fingerprints in a database, to identify the source of an unknown sample. Several techniques of music identification have been researched including using computer vision, wavelets, and peak-pair hashing. The demo program I created used a simplified version of the original Shazam's algorithm, but was able to correctly identify the source song with 96% accuracy using 10 second query samples.

References

[1] "Audio Fingerprinting | ACRCloud Docs," Jul 2018, [Online; accessed 6. Jul. 2018]. [Online]. Available: https://www.acrcloud.com/docs/acrcloud/introduction/audio-fingerprinting

[2] A. L. chun Wang and T. F. B. F, "An industrial-strength audio search algorithm," in *Proceedings of the 4 th International Conference on Music Information Retrieval*, 2003.

[3] "Shazam - Music Discovery, Charts & Song Lyrics," Jul 2018, [Online; accessed 6. Jul. 2018]. [Online]. Available: https://www.shazam.com

[4] "SoundHound Inc." Jul 2018, [Online; accessed 6. Jul. 2018]. [Online]. Available: https://soundhound.com

[5] "Short-time Fourier transform - Wikipedia," Jul 2018, [Online; accessed 6. Jul. 2018]. [Online]. Available: https://en.wikipedia.org/wiki/Short-time_Fourier_transform

[6] "Introduction - FFT Size," Jul 2012, [Online; accessed 6. Jul. 2018]. [Online]. Available: http://support.ircam.fr/docs/AudioSculpt/3.0/co/FFT%20Size.html

[7] V. Chandrasekhar, M. Sharifi, and D. Ross, "Survey and evaluation of audio fingerprinting schemes for mobile query-by-example applications," in *12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011.

[8] Y. Ke, D. Hoiem, and R. Sukthankar, "Computer vision for music identification," pp. 597– 604 vol. 1, 07 2005.

[9] "AdaBoost - Wikipedia," Jul 2018, [Online; accessed 6. Jul. 2018]. [Online]. Available: https://en.wikipedia.org/wiki/AdaBoost

[10] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern Recognition*, vol. 41, no. 11, pp. 3467 – 3480, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320308001702

[11] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang, "Finding interesting associations without support pruning," vol. 13, pp. 64 – 78, 02 2001.

[12] H.-G. Kim, H.-S. Cho, and J. Y. Kim, "Robust audio fingerprinting using peak-pair-based hash of non-repeating foreground audio in a real environment," *Cluster Computing*, vol. 19, no. 1, pp. 315–323, Mar. 2016. [Online]. Available: http://dx.doi.org/10.1007/s10586-015-0523-z

[13] "NumPy — NumPy," Apr 2018, [Online; accessed 6. Jul. 2018]. [Online]. Available: http://www.numpy.org

[14] "SciPy.org — SciPy.org," Jun 2018, [Online; accessed 6. Jul. 2018]. [Online]. Available: https://www.scipy.org