Research on characteristics of sound and its application for music recognition

First Author
Goi Chi Trung

Second Author
Luong The Van

Third Author
Nguyen Ngoc Phong

Abstract

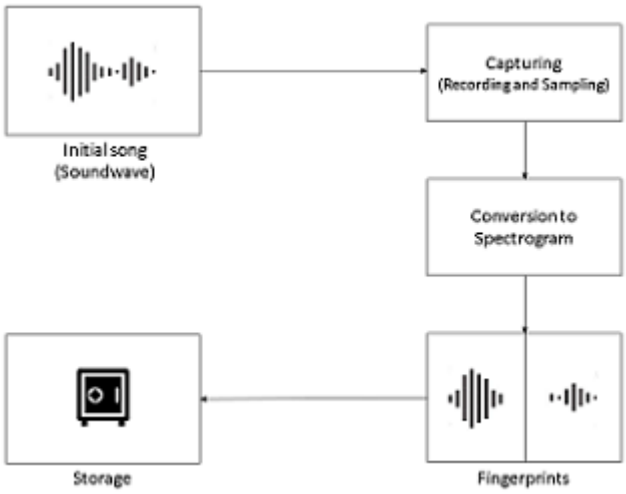
In this topic we present an overview of characteristics of sound, how computers can understand melody by turning sound into audio fingerprint. An audio fingerprint is a content-based compact signature that summarizes an audio recording. Within the context of audio fingerprint, we introduce an algorithm of recognizing which song is playing and who the singer is. For this study, music data is collected according to an experimental design, which enables statements about performance differences with respect to specific music characteristics. Future work concentrates on improving famous songs recognition applications, such as Shazam, to recognize a song played by different singers, instruments, etc.

1 Intruduction

Music plays an important role in our daily life. According to Health Fitness Revolution [2], music helps raising IQ and academic functioning, increasing workout endurance, reducing stress and anxiety, decreasing pain, road rage, and making you sleep better. Forty percent of global time spent listening to music in a typical week is spent listening to radio, twenty percent is spent listening to purchased songs. Same amount of percentage for listening to video streaming and eighteen percent is spent listening to audio streaming, according to Statista.com [2]. Music impacts us in ways that other sounds dont, and the benefits are undeniable.

In this project, our aim is to build a program which receives audio files input by users, reduces noise tolerence, represents audio by spectogram form, match checking with the databases by acknowledging characteristics of sound. Real-life applications are finding songs, singers, license checking on any media platforms, etc.

Our model requires a sound recording from user. Because the computer can only recognize the sound when comparing with other sounds from the dataset, we will convert the sound input into spectrogram [3], a type of 3D graph of time and sound amplitude. We then use hash function to search and index these sounds to the tag including name, artist, time. If the sound input match with dataset, the output result will be the songs information.



2 Background and Related work

When it comes to finding information, searching by text is one of the most common searching algorithms. The searching engine system like Google, Bing, Facebook has had a dramatic development for many years. But those system encounter a limitation when searching for non-text information and data. Especially when it comes to music, whose true nature is rhythm, searching by text clearly show its inefficiency. That is why we come up with an idea to search for songs by the sound inputting by user. This method has showed a great deal of benefit compared with

searching for music by text (usually searching for songs names, artists...).

There are some approaches to build a music recognition model with different algorithms.

2.1 Music Identification System Using MPEG-7 Audio Signature Descriptor [4]

In 2011, Stop online piracy act (SOPA) bill [5] was proposed to protect the copyrighted intellectual property, such as digital content. The first step toward the protection of copyrighted property is to indentify if the property is copyrighted or not. Soundtracks are one type of content which is easily to be illegally reproduced.

Shingchern D. You, Wei-Hwa Chen, and Woei-Kae Chen, from Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei 104, Taiwan, propose a system using multiresolution strategy to identify whether a piece of unknown music is matched with one of the pieces in the database. In the system, high-resolution descriptors are MPEG-7 [6] audio signature descriptors, and the low-resolution descriptors are obtained from high-resolution descriptors with the aid of PCA to reduce their dimensionality. Results show that low-resolution descriptors still have high identification accuracies. Since not every piece of query input is within the database, they also proposed two methods to determine the distance thresholds. Experimental results show that the proposed method II provides an accuracy of 99.4%. Therefore, the proposed system can be used in real applications, such as identifying copyrighted audio files circulated over the Internet.

2.2 Embedding watermarks

A typical method to attach the copyright information to a piece of music is by embedding watermarks [7]. Digital watermarking is the process of attaching the identity of a copyright owner to a digital audio work that is difficult to remove. The technique is analogous to a stain on a dress that is impossible to get out or the mark of floodwater upon a building after the tide has receded. The goal of digital audio watermarking is to give the artistic work owner the capability to prove the work is theirs.

There are many challenges to audio watermarking. The technical requirements are that the watermark does not add distortion, be audible in the signal, be robust against further audio processing, be not removable by attacks designed to separate the watermarking from the audio illegally and it must also be easily identifiable by a decoding mechanism.

Though effective, this method has some limitations, such as the watermarks must be embedded into the source soundtracks before release. Therefore, it is not possible to identify the rights owner of a piece of music without watermarks.

2.3 Shazam

Shazam [8] is a company which dedicate to music indentification. Its database contains around eleven million soundtracks.

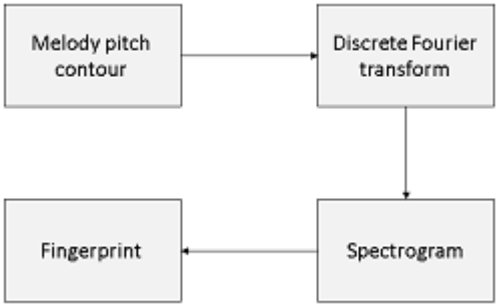
On the server side:

- Shazam precomputes fingerprints [9] from a very big database of music tracks. All those fingerprints are put in a fingerprint database which is updated whenever a new song is added in the song database.

On the client side:

- When a user uses the Shazam app, the app first records the current music with the phone microphone. The phone applies the same fingerprinting algorithm as Shazam on the record. The phone sends the fingerprint to Shazam. Shazam checks if this fingerprint matches with one of its fingerprints. If no it informs the user that the music cant be found. If yes, it looks for the metadata associated with the fingerprints (name of the song, ITunes url, Amazon url) and gives it back to the user

3 Method



3.1 Melody Pitch Contour

One of the most basic attribute of music is the pitch of note, melody and rhythm. Melody of a song is a component of notes with different pitch. Dowling has come up with the new approach to perform the strain of pitch which is called Melody Pitch Contour..

3.1.1 Turn analog signal into digital signal

Sound is a wave that is deliver by vibration to our ear. This sound wave is actually a continuous signal. When we record sound by our microphone, this signal is translated into analog signal. Then because of more efficient storing, it is then turned into the digital discrete signal. This process is called sampling.

3.2 Discrete Fourier Transform and Fast Fourier Transform

3.2.1 Discrete Fourier Transform

The Discrete Fourier Transform (DFT) [11] will help to convert digital audio signals into discrete spectrum. The formula including the size of window number of samples that composed the signals, frequencies. DFT transforms the digital signal into frequencies and then perform the discrete spectrum based on the frequecies according to time.

$$X(n) = \sum_{k=0}^{N-1} x[k] e^{-j(2\pi kn/N)}$$

The DFT is a tool to perform Fourier analysis on a signal. One of the way to calculate DFT is Fast Fourier Transform (FFT).

3.2.2 Compute Discrete Fourier Transform using Fast Fourier Transform

TThe Fast Fourier Transform (FFT) [12] is one of the most important algorithms in signal processing and data analysis. The FFT is a fast, O[Nlog N] algorithm to compute the Discrete Fourier Transform (DFT), which naively is an O[N2] computation.

Due to the importance of FFT in various of fields, Python contains many standard tools and wrappers to compute this. Both NumPy and SciPy have wrappers of the extremely well-tested FFTPACK library, found in the submodules numpy.fft and scipy.fftpack respectively. Set aside these implementations, lets compute FFT from scratch.

3.2.3 Compute Fast Fourier Transform

Divide the DFT computation into two smaller parts:

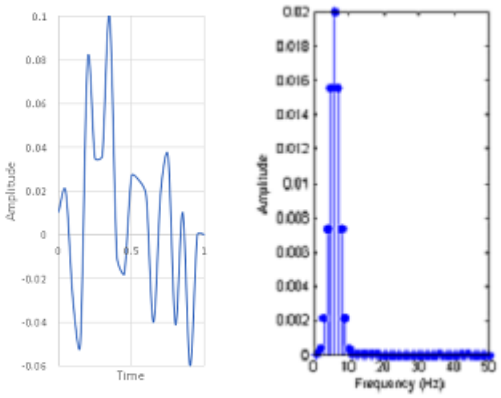
$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i\,2\pi\,k\,n\,/\,N} \tag{1}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-i\,2\pi\,k\,(2m)\,/\,N} + \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-i\,2\pi\,k\,(2m+1)\,/\,N} \tag{2}$$

$$= \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-i\,2\pi\,k\,m\,/\,(N/2)} + e^{-i\,2\pi\,k\,/\,N} \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-i\,2\pi\,k\,m\,/\,(N/2)} \tag{3}$$

The result is that the DFT turns into two terms which look similar to a smaller DFT, one on the odd-numbered values, and one on the even-numbered values. Each term consists of (N/2)* N computations, for a total of N^2. Because the range of k is $0 \leq k < N$, while the range of m is $0 \leq m < M \equiv N/2$, we see from the symmetry properties of DFT, we only need to perform half the computations for each sub-problem. Our O[N^2] computation has become O[M^2], with M is half the size of N. As long as our smaller Fourier transforms have an even-valued M, we can reapply this divide-and-conquer approach, halving the computational cost each time, until our arrays are small enough that the strategy

is no longer beneficial. Recursive algorithm is approached, and it is asymptopical O[NlogN], which means Fast Fourier Transform has been implemented.



3.3 Spectrogram

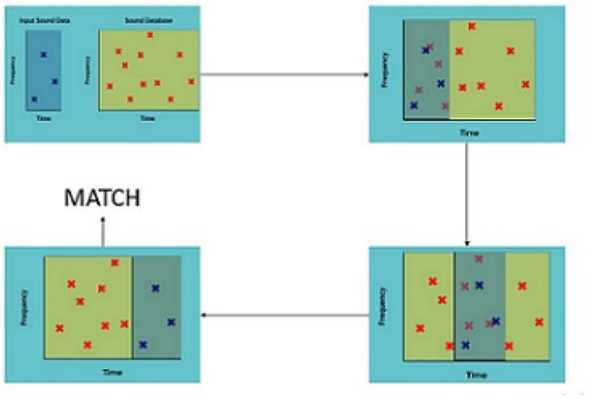
Storing songs data as well as input audio file is less expensive when the database is converted to spectrogram format. Spectrogram is a 3-Dimension chart (3D plot) which includes the x axis as time (second), y axis as frequency (Hz), and color tempature as amplitude of sound (dB).

In order to make searching and matching audio data more efficient, the solution is to turn spectrogram into fingerprints, as mentioned in [9]. Since every note created usually oscillates in a certain frequency range, capturing points which have highest frequency (higest peaks) in a unit of time will simplify the spetrogram, inferring the complexity of searching and matching. Filtering highest peaks also enables us to avoid noise tolerant and change the spetrogram into 2D plot.

However in one song, there are usually many ranges of strong frequencies (from C 30.47Hz to C8 4186Hz). Because of a huge range of interval, we can choose smaller intervals instead and analyze them separately. Now for each interval we will have the highest peak. Those peak become a signature for the sound input and is used to compare with the dataset.

3.3.1 Matching with database

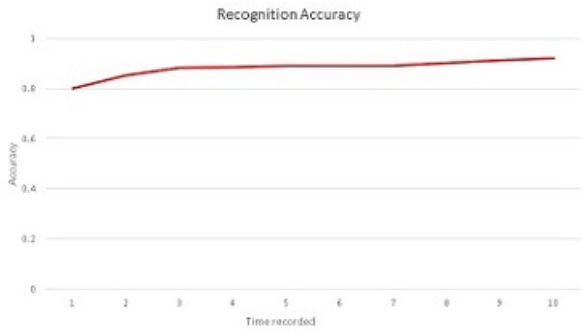
Each point marked in the spectrogram have a coordinate (x, y). The absolute value of these coordinates will be compared with those in the database (which are also converted into spectrogram). The accurate matching result returns the song data.



4 Result

Initial testing starts by inputting several five-second audio .wav files into the model due to the lack of microphone, so the result is expected to be high since there is no noise disturbance. The result gives 90% of accuracy when the model returns the correct songs data.

Continue with this process, the model receives audio files with the length ranging from six to ten seconds. The result is getting higher and reach 95% accuracy when the input is ten-second audio file. Here are the results as the function of time:



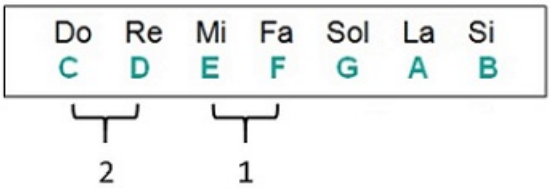
Using the method above, the result is only high when the input is cut exactly from the song which is in the database. If with the exact same song but the performer is another singer or song performed by instruments, the model does not work. Due to the difference of the frequency of different singers, our next approach to implement an algorithm to improve this problem is represented in the next section.

5 Discussion

The main objective of the model is to help users find the songs information like name, singer, published date,

But as mentioned above, the problem when using this model is that it cant find the songs data which is already in the database because the input audio file is a cut song perform by a different singer or instruments. Our future approach is to determine in the physics of music note [13].

There are seven notes in a chord: Do, Re, Mi, Fa, Sol, La, Si, signed as C, D, E, F, G, A, B respectively, called octave [14]. The distance between two adjacent notes is an interval, marked as 2, except for the distance of E and F is half an interval, marked as 1. The distance from the last note of one octave, B, to the first note of the next octave, C, is also half an interval.



Our approach will concentrate in evaluating the databases note distances and storing them into an ordered 1D array (vector). The distance of note is the same even when the songs is performed by different singers or instruments. Therefore, by storing the database into note-distance-ordered array, the model is expected to find the songs data while the input song is not the original one. The input audio file is also evaluated for note distance, compared with database to find matching. The period of time to find the song will be longer due to the fact that many songs might have the same note distance for the first few seconds, but it guarantees to return the songs data which various performers.

6 Conclusion

In this paper, a song recognition model has been built using the act of capturing and matching audio fingerprints from the spectrogram. Our database contains hundred of songs spectrogram with captured fingerprints. Then by inputting audio file into the model, it will also convert the audio file into spectrogram and capture the fingerprints. Next the model will start to compare the input audio files spectrogram with ones in the database. If there is a match, the model is expected to return the songs informations. In our experiment, the model reaches around 90% accuracy when the input audio file is five seconds long, and 95% when it is 10 seconds long.

Since our model cant not find the song which is already in the database due to different performers, our future work will concentrate on music physics as mentioned in the Discussion section. The demo of the algorithm is already implemented in C++ to calculate the distance between the notes. Further research features collecting data for songs note lyrics for database and how to convert input audio files into note lyrics for finding and matching database.

References

[1] Health Fitness Revolution: *Top 10 Benefits of Listening to Music*, January 2nd, 2019. `http://www.healthfitnessrevolution.com/top-10-benefits-listening-music/`

[2] `Statista.com`: *Weekly time spent listening to music in the United States as of August 2017, by type of consumer (in hours)*,August 2017.

[3] Spectrogram, `https://en.wikipedia.org/wiki/Spectrogram`.

[4] Shingchern D. You, Wei-Hwa Chen, and Woei-Kae Chen: *Music Identification System Using MPEG-7 Audio Signature Descriptors*, The Scientific World Journal Volume 2013, 2013.

[5] Stop Online Piracy Act, `https://en.wikipedia.org/wiki/Stop_Online_Piracy_Act`.

[6] F. Nack and A. T. Lindsay, *Everything you wanted to know about MPEG-7: part 1 and 2*, IEEE Multimedia, vol. 6, no. 3, pp. 6577 and vol. 6, no.4, pp.64-73, 1999.

[7] R. Eklund, *Audio watermarking techniques*, `http://www.musemagic.com/papers/watermark.html`.

[8] Shazam, https://www.shazam.com/.

[9] Pedro Cano and Eloi Battle, Ton Kalker and Jaap Haitsma:*A Review of Audio Fingerprinting*, Journal of VLSI Signal Processing 41, 271284, 2005.

[10] Justin Salamon, Geoffroy Peeters and Axel Robel:*Statistical Characterisation Of Melodic Pitch Contours and Its Application for Melody Extraction*.

[11] Discrete Fourier Transform, https://en.wikipedia.org/wiki/Discrete$_Fourier_transform$.

[12] Fast Fourier Transform,https://en.wikipedia.org/wiki/Fast$_Fourier_transform$.

[13] B. H. Suits: *Physics of Music-Notes*, hysics Department, Michigan Technological University, copyright 1998-2019. https://pages.mtu.edu/~suits/Physicsofmusic.html

[14] Octave, https://en.wikipedia.org/wiki/Octave.