Music Informatics Coursework 2: Audio Identification

INTRODUCTION

This project implements audio identification as described in Haitsma, 2002 [1], also known as the Philips algorithm. Often pitched against Wang, 2003 [2] (Shazam algorithm), the Philips approach is known to be robust to pitch shifts, and time-shifts, but cannot work with GSM (Global System for Mobile Communications) compression [3].

While there has been several works done to improve the algorithm in both its fingerprinting [3] and search [4] components, this project aims to implement the original paper to get a solid understanding of the key ideas behind the algorithm.

This report details the approach taken, evaluates its performance, and discusses its merits and room for improvement.

THEORY

The Philips approach has 2 parts:

1. Fingerprinting
2. Search algorithm

Haitsama, 2001 proposed a method for audio fingerprinting based on the idea that each piece of music can be represented by the energy differences between neighbouring Mel-spectrogram bands and frames. The differences are encoded as 1s and 0s, depending on whether they are positive or negative differences. With a proposed number of 33 Mel bands and frames of size 0.37 seconds, each song is represented by a series of 32-bit sub-fingerprints.

To query from a database of fingerprinted songs, the search algorithm first looks for a single frame match (i.e. any 32-bit sub-fingerprint in the query should match one 32-bit sub-fingerprint in the database song). Then, to verify if the song is a match, a fingerprint block of 256 frames containing the matched sub-fingerprint in both the query and database song are compared against each other. If the similarity exceeds a threshold, then the match is returned.

While efficient, the search algorithm is highly reliant on the assumption that there is at least one exact-match sub-fingerprint between the query and the actual song. If this assumption does not hold, there will not be a match returned. To relax this assumption, the paper also proposed iteratively switching bits in the query sub-fingerprints, starting from the most 'unstable' bit (i.e. the bit with energy difference closest to 0, since it could have been encoded differently with a small change in energy difference.)

In this project, we first implement the simple version of the search algorithm, before exploring more complex alterations.

IMPLEMENTATION

Fingerprinting

While the idea behind using energy differences as fingerprints sounds simple, the key in achieving unique but representative fingerprints lies in the parameters of the Mel-spectrogram. In the paper, specific values are set:

1. The audio is downsampled to 5000Hz
2. Window size is 0.37 seconds, weighted by a Hanning window with an overlap factor of 31/32.
3. There are 33 Mel-bands
4. Only lower frequencies from 300Hz to 2000Hz are considered.

In this project's implementation, the Mel-spectrogram parameters follow the ones specified above. However, using an overlap factor of 31/32 was found to insufficient. When testing on a few randomly selected queries, they did not meet the key assumption of having at least one exact matching sub-fingerprint, but were at least 4-5 bits away from the query fingerprint. As a result, the overlap factor was increased to 63/64, essentially halving the hop size. A comparison of performance differences can be found in the Evaluation section.

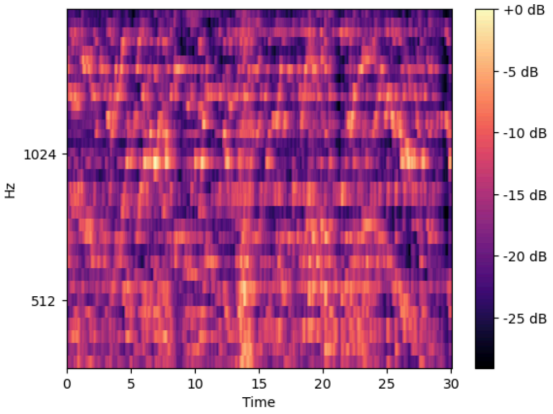

Figure 1: Mel-spectrogram for classical.00000.wav (overlap of 31/32)

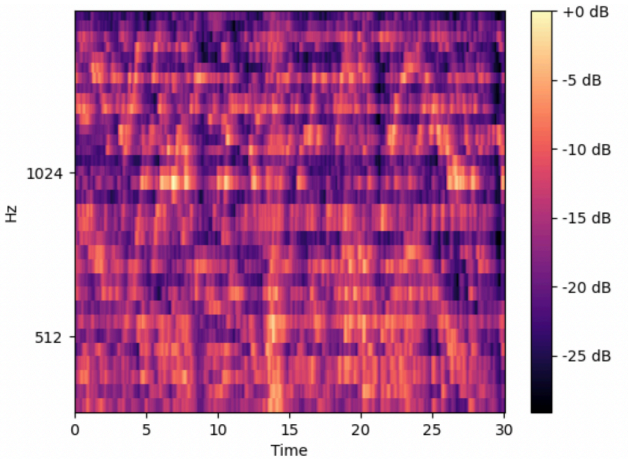

Figure 2: Mel-spectrogram for classical.00000.wav (overlap of 63/64)

As observed in Figures 1 and 2 above, halving the hop size in Figure 2 leads to a higher temporal resolution in the Mel-spectrogram. This translates to each sub-fingerprint corresponding to a hop size of 5.8ms instead of 11.6ms.

From the Mel-spectrogram, each frame in the Mel-spectrogram is encoded into a 33-bit sub-fingerprint based on the following equation (1):

$$F(n,m)=\begin{cases}1 \text{ if } E(n,m)-E(n,m+1)-\left(E(n-1,m)-E(n-1,m+1)\right)>0\\ 0 \text{ if } E(n,m)-E(n,m+1)-\left(E(n-1,m)-E(n-1,m+1)\right)\le 0\end{cases} \quad (1)$$

Where F(n, m) is the value for the m-th bit of the n-th frame, and E(n, m) is the energy value in the m-th band of the n-th frame.

In the more complex search strategy, this encoding was not performed, and instead, the intermediate sub-fingerprints were represented by the raw values of the energy differences instead of encoded bits. This was to enable the switching of 'unstable' bits based on the energy difference values in the complex version of the search strategy.

Search Algorithm

A database of audio files are fingerprinted and encoded as described above. They are then stored in a hash-map, represented by a dictionary in Python, where the keys are all possible 32-bit sub-fingerprint values in the database. Their corresponding values are the file ids of the audio files containing the sub-fingerprint, alongside the frame index of the sub-fingerprints.

Given a query audio file, the query is similarly fingerprinted and encoded as described above.

In the simple search strategy, a query performs the following steps:

1. Loop through all sub-fingerprints in the query audio file
2. For each sub-fingerprint, look for its match among the keys of the hash-map
3. If there is a match, look through all the audio files containing that key (as identified by the file ids in the values of the dictionary)
4. For each audio file from step (3), get its 256-frame fingerprint block from frame p to p-256, where p is the index where the matching sub-fingerprint occurs.
5. Calculate the Bit Error Rate (BER) between the fingerprint block of the audio file and the fingerprint block of the query file. BER is defined as the ratio of differing bits between the fingerprint of the audio file and the query file.

In the complex search strategy, the steps above are conducted, and then extended as follows:

6. For each sub-fingerprint, swap the bit with energy difference closest to 0, and run steps (2) to (5) on the edited sub-fingerprint. In the code implementation, this step is done a maximum of 3 times (i.e. 3 bits are swapped in each sub-fingerprint).

After getting the BERs for each fingerprint block, each potential match is represented by its smallest BER. Then, the results are ranked according to BER, with the smallest BER having the highest rank. This differs form the original paper, which only returns a match if the BER is below a threshold of 0.35. However, since our system allows for Top 3 predictions, we simply use the BER as ranking scores, instead of applying a fixed threshold.

EVALUATION

The beat tracking approach described above is then evaluated on the all samples in the GTZAN dataset. Before looking into the details of evaluation, we first look at the nature of the dataset.

Dataset

The GTZAN dataset consists of 200 database audio files, and 213 query audio files that have added noise. Some query files are so noisy that it is almost impossible to hear the music in the background.

Fingerprints

256 sub-fingerprints make up a 'fingerprint block', and an example can be shown in Figures 3 and 4 below. From a first glance, the 2 fingerprint blocks are not exactly visually similar. This would give us an idea of the level of accuracy we can expect in the next section.

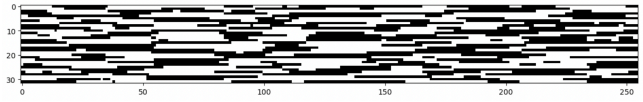

Figure 3: A fingerprint block for query 'classical.00000-snippet-10-0.wav'

Figure 4: A fingerprint block for its corresponding match 'classical.00000.wav'

Audio Identification Metrics

	Accuracy@1	Accuracy@2	Accuracy@3
Simple search (OF=32)	6.1%	7.9%	7.9%
Complex search (OF=32)	7.9%	10.8%	11.0%
Simple search (OF=64)	8.9%	11.27%	11.27%
Complex search (OF=64)	14.1%	19.2%	20.1%

Table 1: Accuracy metrics comparing different overlapping factors and search algorithms.

We measure 3 levels of accuracy@k for audio identification, where accuracy@k is defined as the proportion of correctly identified songs within the top k results. The results were ranked according to the smallest BER in each match.

From the metrics above, we observe that increasing the overlapping factor does improve accuracy metrics. In the case of using the complex search algorithm, the effect is almost double when overlapping factor doubles. Furthermore, the complex search algorithm consistently performs better than the simple search algorithm, which is expected since it has a 'wider' search criteria.

When listening to the audio queries for samples that have been matched, we observe that these queries are naturally 'easier' to

match, since the song can be heard relatively clearer, and there is less noise in the audio. More importantly, the matched samples have fewer instances of human voices, which might have interfered with the fingerprinting of the query audio, especially since the fingerprints are solely based on energy differences.

Time Metrics

The time taken to construct a database of fingerprints is about 1.52 minutes for an overlap factor of 32, 3.03 minutes for an overlap factor of 64, and 13.1 minutes for a overlap factor of 320. Naturally, a larger overlap factor results in more frames, which translates to more sub-fingerprints in the database, causing the increase in time. However, the time increment is not linear to the increase in number of frames, which is great, since it means that if we wanted to make our fingerprints more complex (i.e. more number of bits), it will not affect the fingerprinting time linearly.

The time taken for audio identification is about 0.2s per query for an overlap factor of 32, and 0.5s per query for an overlap factor of 64. Interestingly, there is barely any difference between the query times for the simple search vs the complex search algorithms. Queries for an overlap factor of 320 was not tested in the interest of time.

DISCUSSION

From the experiments, we observe that the Philips algorithm does not work well on GTZAN with noise, especially if the noise is very loud and contains human voices. Its performance is greatly hindered by the assumption that at least one sub-fingerprint will be an exact match, which was hardly the case for our dataset. If the assumption works, the algorithm is very promising, since the runtime performance is fast. However, without the certainty that at least one sub-fingerprint will match, it becomes much more difficult problem to find matches based on the 32-bit fingerprints. A brute-force method for searching that was also discussed in the paper would be highly inefficient.

Furthermore, the parameters provided in the paper might not be optimal for the dataset. We observed that increasing the overlap factor improved the accuracy scores, although it is expected that the improvement in accuracy will only work up to a certain extend. Increasing the time and frequency resolution of the Mel-spectrogram will result in more precise and unique fingerprints, but will come at the cost of compute time. This also suggests that the parameters in the original algorithm might not be the most generalisable, and they might need to be further tuned for different audio qualities - perhaps they are not ‘highly robust’ after all.

FUTURE IMPROVEMENTS

To address these limitations, more tuning can be done to find the optimum parameters. For example, increasing the number of Mel bands might help to differentiate between the harmonic frequencies in music and other noise. It might however, still not work in the case of noise from human voices. It would also mean an increased bit-size for the fingerprint representations, which would result in more resources needed.

If we would like to stick to the parameters in the original paper, perhaps some pre-processing of the audio file is needed before feeding it into the audio identification component. For example, removing noise and speaking voices might help improve the performance of the identification component.

REFERENCES

[1] Haitsma, J., & Kalker, T. (2002, October). A highly robust audio fingerprinting system. In *Ismir* (Vol. 2002, pp. 107-115).

[2] Wang, A. (2003, October). An industrial strength audio search algorithm. In *Ismir* (Vol. 2003, pp. 7-13).

[3] Chu, R., Niu, B., Yao, S., & Liu, J. (2019, September). Peak-based philips fingerprint robust to pitch-shift for massive audio retrieval. In *2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM)* (pp. 314-320). IEEE.

[4] Yao, S., Niu, B., & Liu, J. (2018, September). Enhancing sampling and counting method for audio retrieval with time-stretch resistance. In *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)* (pp. 1-5). IEEE.