PANAKO 2.0 - UPDATES FOR AN ACOUSTIC FINGERPRINTING SYSTEM

Joren Six
IPEM, Ghent University, Belgium

ABSTRACT

This work presents updates to Panako, an acoustic fingerprinting system that was introduced at ISMIR 2014. The notable feature of Panako is that it matches queries even after a speedup, time-stretch or pitch-shift. It is freely available and has no problems indexing and querying 100k seashanties. The updates presented here improve query performance significantly and allow a wider range of time-stretch, pitch-shift and speed-up factors: e.g. the top 1 true positive rate for 20s query that were sped up by 10 percent increased from 18% to 83% from the 2014 version of Panako to the new version.

The aim of this short write-up is to reintroduce Panako, evaluate the improvements and highlight two techniques with wider applicability. The first of the two techniques is the use of a constant-Q non-stationary Gabor transform: a fast, reversible, fine-grained spectral transform which can be used as a front-end for many MIR tasks. The second is how near-exact hashing is used in combination with a persistent B-Tree to allow some margin of error while maintaining reasonable query speeds.

1. INTRODUCTION

Acoustic fingerprinting solves the problem to efficiently find short audio fragments in large audio archives. It has many uses cases ranging from copyright management, music identification, smart-costume designing to data synchronization. Acoustic fingerprinting that also works when a query is time-stretched, pitch-shifted or sped up allows, for example, DJ-set analysis.

A classical approach is to use combinations of peaks in a spectral representation as a fingerprint. This feature has been used in several systems. Panako, a system that was later found to be conceptually similar to a patented method.

Panako received updates in 2021 after close inspection of an efficient implementation of a similar algorithm for embedded systems. The underlying concepts from the original paper still stand but two changes improve the system considerably.

2. CHANGES

The first change replaces the frequency transform from a classical constant-Q transform to a constant-Q non-stationary Gabor transform. The latter has more efficient and allows a finer frequency resolution at equal computational cost. See for a detailed comparison. The spectral peak detection in Panako improves by the use of this finer frequency resolution. In Panako, JGaborator is used: a wrapper around the Gaborator library which implements a constant-Q non-stationary Gabor transform in C++11.

The second change replaces an exact hash matching technique by a near-exact hash matching approach. Some background helps understand this change. The first step in the Panako algorithm is a transform a one dimensional waveform into a two dimensional time/frequency grid. Each fingerprint hash consists of a combination of three local peaks in this grid. Each peak has a frequency, time and magnitude component. Each bin in the time/frequency grid has a very short duration and a small frequency dimension. The exact dimensions of these bins are determined by the spectral transform parameters and can be small but they remain discrete. This means that when a query and a match differ by about half the duration of a bin, energy is spread over neighbouring bins. Since time-frequency coordinates of peak magnitude bins are used in fingerprint hashes, off-by-one errors are to be expected, both in time and frequency. A design is needed to cope with such cases.

For indexing and matching a hash is constructed from the components mentioned below. A hash combines fingerprint information into a single integer. The additional information contains an audio identifier used to tally matches. Below, the components are ordered from most to least significant. By only including approximate frequency information and having the time ratio in the least significant bytes, range queries become possible. The last couple of bits can be ignored during a search in ordered hashes: effectively dealing with off-by-one errors.

\[
\begin{align*}
|f_3 - f_2|/4 : |f_2 - f_1|/4 : \tilde{f}_1 \\
|t_3 - t_2| > |t_2 - t_1| \\
m_3 > m_2 ; m_3 > m_1 ; m_1 > m_2 \\
f_3 > f_1 ; f_3 > f_2 ; f_1 > f_2 \\
(t_2 - t_1)/(t_3 - t_1)
\end{align*}
\]

This idea of gracefully handling off-by-one errors needs to be reflected in the matching scheme as well. The fingerprints extracted from a query are matched with the
Figure 1. A linear regression from the first matches to the last (the blue range) is used to allow some small margin in which matches are accepted (green area). This allows off-by-one matches and linear time-stretching or speed modifications.

index. A list of matching prints is returned and needs to be filtered: hashes might randomly collide or short fragments might match for a very short duration. To filter true positive matches from false positives the difference in time ($\Delta t$) between each reference and query hash. For a true match $\Delta t$ is either a constant or changes linearly over time. In the original paper [8] a true positive is only accepted if $\Delta t$ is a fixed constant. Here, we calculate a linear regression from the first matches to the last and allow some small margin in which matches are accepted, the blue range in 1). In this manner off-by-one matches and linear time-stretching/tempo modifications are supported.

In [5] a different trade-off was made. There, four spectral peaks are combined in a fingerprint which are indexed in a multi-dimensional R-tree. This allows to execute a range query. A multidimensional range query in a large data set is quite a bit more costly than a near-exact hash query in a single dimension. Effectively, query performance is traded for higher retrieval accuracy [5].

When larger archives are indexed, the characteristics of the key-value store become more and more important. The key-value store stores a hashes together with some additional information. The 2021 Panako version stores ordered fingerprints using a persistent, compact, high performance, B-Tree [14]. Now, LMDB (Lightning Memory-Mapped Database Manager, http://lmdb.tech) is used. The speed, small storage overhead and performance allow more beneficial trade-offs between query performance: it facilitates storing more fingerprints per second of audio or large while still maintaining a similar query speed.

3. EVALUATION AND CONCLUSION

Panako contains an evaluation script. The script takes a folder with audio files as input and indexes 80% of the files. The remaining 20% are used to check true negatives. After indexing a number of query files are created. These are randomly selected 10 and 20 second fragments which are modified in several ways: time-stretched, pitch shifted, sped up, filtered, ... The Panako system is presented with the modified queries and it is checked whether the expected match is returned. The script then calculates a range of metrics. Figure 2 contains the most relevant metrics which have been achieved using the Free Music Archive [15]. Note that text describes and evaluates Panako as is in the following commit found on GitHub: 6cf936730131d71c94c562a06a1a791e09b4c520. Figure 2 shows a significant improvement in robustness in modifications with respect to the Panako 2014 version. Query speeds are around 40 times real-time both for query as indexing using a single thread on an older CPU (early 2015 macBook pro Dual Core i5). Readers are encouraged to re-run the evaluation using their specific music data sets.

To conclude: the updated version of Panako has been introduced. It contains two techniques with wider applicability: the use of a constant-Q non-stationary Gabor transform and a near-exact hashing technique for range queries. The evaluation of the updated system shows significant improvements in true positive rates after speed-up, time-stretching and pitch-shifting while maintaining reasonable query speeds.
4. REFERENCES


