PANAKO - A SCALABLE ACOUSTIC FINGERPRINTING SYSTEM HANDLING TIME-SCALE AND PITCH MODIFICATION

Six Joren, Marc Leman
Institute for Psychoacoustics and Electronic Music (IPEM),
Department of Musicology, Ghent University
Ghent, Belgium
joren.six@ugent.be

ABSTRACT

This paper presents a scalable granular acoustic fingerprinting system. An acoustic fingerprinting system uses condensed representation of audio signals, acoustic fingerprints, to identify short audio fragments in large audio databases. A robust fingerprinting system generates similar fingerprints for perceptually similar audio signals. The system presented here is designed to handle time-scale and pitch modifications. The open source implementation of the system is called Panako and is evaluated on commodity hardware using a freely available reference database with fingerprints of over 30,000 songs. The results show that the system responds quickly and reliably on queries, while handling time-scale and pitch modifications of up to ten percent.

The system is also shown to handle GSM-compression, several audio effects and band-pass filtering. After a query, the system returns the start time in the reference audio and how much the query has been pitch-shifted or time-stretched with respect to the reference audio. The design of the system that offers this combination of features is the main contribution of this paper.

1. INTRODUCTION

The ability to identify a small piece of audio by comparing it with a large reference audio database has many practical use cases. This is generally known as *audio fingerprinting* or *acoustic fingerprinting*. An acousic fingerprint is a condensed representation of an audio signal that can be used to reliably identify identical, or recognize similar, audio signals in a large set of reference audio. The general process of an acoustic fingerprinting system is depicted in Figure 1. Ideally, a fingerprinting system only needs a short audio fragment to find a match in large set of reference audio. One of the challenges is to design a system in a way that the reference database can grow to contain millions of entries. Another challenge is that a robust fingerprinting should handle noise and other modifications well, while limiting the amount of false positives and processing time [5]. These modifications typically include dynamic range compression, equalization, added background noise and artifacts introduced by audio coders or A/D-D/A conversions.

Over the years several efficient acoustic fingerprinting methods have been introduced [1, 6, 8, 13]. These methods perform well, even with degraded audio quality and with industrial sized reference databases. However, these systems are not designed to handle queries with modified time-scale or pitch although these distortions can be present in replayed material. Changes in replay speed can occur either by accident during an analog to digital conversion or they are introduced deliberately.

Accidental replay speed changes can occur when working with physical, analogue media. Large music archive often consist of wax cylinders, magnetic tapes and gramophone records. These media are sometimes digitized using an incorrect or varying playback speed. Even when calibrated mechanical devices are used in a digitization process, the media could already have been recorded at an undesirable or undocumented speed. A fingerprinting system should therefore allow changes in replay speed to correctly detect duplicates in such music archives.

Deliberate time-scale manipulations are sometimes introduced as well. During radio broadcasts, for example, songs are occasionally played a bit faster to make them fit into a time slot. During a DJ-set pitch-shifting and time-stretching are present almost continuously. To correctly identify audio in these cases as well, a fingerprinting system robust against pitch-shifting and time-stretching is desired.

Some fingerprinting systems have been developed that take pitch-shifts into account [3, 7, 11] without allowing time-scale modification. Others are designed to handle both pitch and time-scale modification [10, 14]. The system by Zhu et al [14] employs an image processing algorithm on an auditory image to counter time-scale modification and pitch-shifts. Unfortunately, the system is computationally expensive, it iterates the whole database to find a match. The system by Malekesmaeili et al [10] allows extreme pitch- shifting and time-stretching, but has the same problem. To the best of our knowledge, a description of a practical acoustic fingerprinting system that allows sub-

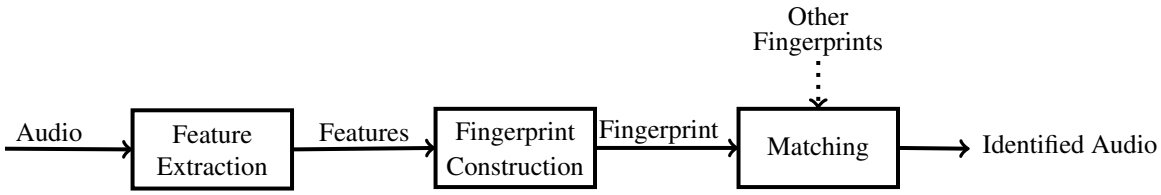

Figure 1: **A generalized audio fingerprinter scheme**. Audio is fed into the system, features are extracted and fingerprints constructed. The fingerprints are consecutively compared with a database containing the fingerprints of the reference audio. The original audio is either identified or, if no match is found, labeled as unknown.

stantial pitch-shift and time-scale modification is nowhere to be found in the literature. This description is the main contribution of this paper.

2. METHOD

The proposed method is inspired by three works. Combining key components of those works results in a design of a granular acoustic fingerprinter that is robust to noise and substantial compression, has a scalable method for fingerprint storage and matching, and allows time-scale modification and pitch-shifting.

Firstly, the method used by Wang [13] establishes that local maxima in a time-frequency representation can be used to construct fingerprints that are *robust to quantization effects, filtering, noise and substantial compression*. The described exact-hashing method for *storing and matching fingerprints has proven to be very scalable*. Secondly, Artz et al. [2] describe a method to align performances and scores. Especially interesting is the way how triplets of events are used to search for performances with different timings. Thirdly, The method by Fenet et al. [7] introduces the idea to extract fingerprints from a Constant-Q [4] transform, a time-frequency representation that has a constant amount of bins for every octave. In their system a *a fingerprint remains constant when a pitch-shift occurs*. However, since time is encoded directly within the fingerprint, the method does not allow time-scale modification.

Considering previous works, the method presented here uses local maxima in a spectral representation. It combines three event points, and takes time ratios to form time-scale invariant fingerprints. It leverages the Constant-Q transform, and only stores frequency differences for pitch-shift invariance. The fingerprints are designed with an exact hashing matching algorithm in mind. Below each aspect is detailed.

2.1 Finding Local Maxima

Suppose a time-frequency representation of a signal is provided. To locate the points where energy reaches a local maximum, a tiled two-dimensional peak picking algorithm is applied. First the local maxima for each spectral analysis frame are identified. Next each of the local maxima are iterated and put in the center of a tile with $\Delta T \times \Delta F$ as dimensions. If the local maximum is also the maximum within the tile it is kept, otherwise it is discarded. Thus,

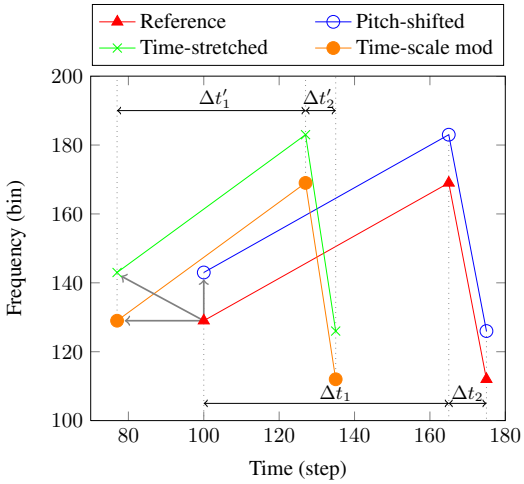

Figure 2: The effect of time-scale and pitch modifications on a fingerprint. It shows a single fingerprint extracted from reference audio (—▲—) and the same fingerprint extracted from audio after pitch-shifting (—○—), time-stretching (—●—) and time-scale modification (—×—).

making sure only one point is identified for every tile of $\Delta T \times \Delta F$. This approach is similar to [7, 13]. This results in a list of event points each with a frequency component f, expressed in bins, and a time component t, expressed in time steps. ΔT and ΔF are chosen so that there are between 24 and 60 event points every second.

A spectral representation of an audio signal has a certain granularity; it is essentially a grid with bins both in time as in frequency. When an audio signal is modified, the energy that was originally located in one single bin can be smeared over two or more bins. This poses a problem, since the goal is to be able to locate event points with maximum energy reliably. To improve reliability, a post processing step is done to refine the location of each event point by taking its energy and mixing it with the energy of the surrounding bins. The same thing is done for the surrounding bins. If a new maximum is found in the surroundings of the initial event point, the event point is relocated accordingly. Effectively, a rectangular blur with a 3×3 kernel is applied at each event point and its surrounding bins.

Once the event points with local maximum energy are identified, the next step is to combine them to form a fingerprint. A fingerprint consists of three event points, as seen in Figure 2. To construct a fingerprint, each event point is combined with two nearby event points. Each event point can be part of multiple fingerprints. Only be-

tween 8 and 20 fingerprints are kept every second. Fingerprints with event points with the least cumulative energy are discarded. Now that a list of fingerprints has been created a method to encode time information in a fingerprint hash is needed.

2.2 Handling Time Stretching: Event Triplets

Figure 2 shows the effect of time stretching on points in the time-frequency domain. There, a fingerprint extracted from reference audio (Fig.2, —▲—) is compared with a fingerprint from time stretched audio (Fig.2, —●—). Both fingerprints are constructed using three local maxima e_1, e_2, e_3 and e'_1, e'_2, e'_3 . While the frequency components stay the same, the time components do change. However, the ratios between the time differences are constant as well. The following equation holds [1] :

$$\frac{t_2 - t_1}{t_3 - t_1} = \frac{t'_2 - t'_1}{t'_3 - t'_1} \tag{1}$$

With event point e_n having a time and frequency component (t_n, f_n) and the corresponding event points e'_n having the components (t'_n, f'_n). Since $t_3 - t_1 \geq t_2 - t_1$, the ratio always resolves to a number in the range $]0, 1]$. This number, scaled and rounded, is a component of the eventual fingerprint hash (an approach similar to [2]).

Now that a way to encode time information, indifferent of time-stretching, has been found, a method to encode frequency, indifferent to pitch-shifting is desired.

2.3 Handling Pitch-Shifts: Constant-Q Transform

Figure 2 shows a comparison between a fingerprint from pitch shifted audio (—○—) with a fingerprint from reference audio (—▲—). In the time-frequency domain pitch shifting is a vertical translation and time information is preserved. Since every octave has the same number of bins [4] a pitch shift on event e_1 will have the following effect on it's frequency component f_1, with K being a constant, $f'_1 = f_1 + K$. It is clear that the difference between the frequency components remains the same, before and after pitch shifting: $f_1 - f_2 = (f'_1 + K) - (f'_2 + K)$ [7]. In the proposed system three event points are available, the following information is stored in the fingerprint hash: $f_1 - f_2; f_2 - f_3; \tilde{f}_1; \tilde{f}_3$

The last two elements, $\tilde{f}_1$ and $\tilde{f}_3$ are sufficiently coarse locations of the first and third frequency component. They are determined by the index of the frequency band they fall into after dividing the spectrum into eight bands. They provide the hash with more discriminative power but also limit how much the audio can be pitch-shifted, while maintaining the same fingerprint hash.

2.4 Handling Time-Scale Modification

Figure 2 compares a fingerprint of reference audio (Fig.2, —▲—) with a fingerprint from the same audio that has been sped up (Fig.2, —×—). The figure makes clear that speed change is a combination of both time-stretching and pitch-shifting. Since both are handled in with the previous measures, no extra precautions need to be taken. The next step is to combine the properties into a fingerprint that is efficient to store and match.

2.5 Fingerprint Hash

A fingerprint with a corresponding hash needs to be constructed carefully to maintain aforementioned properties. The result of a query should report the amount of pitch-shift and time-stretching that occurred. To that end, the absolute value of f_1 and $t_3 - t_1$ is stored, they can be used to compare with f'_1 and $t'_3 - t'_1$ from the query. The time offset at which a match was found should be returned as well, so t_1 needs to be stored. The complete information to store for each fingerprint is:

$$\left(f_1 - f_2; f_2 - f_3; \tilde{f}_1; \tilde{f}_3; \frac{t_2 - t_1}{t_3 - t_1}\right); t_1; f_1; t_3 - t_1; id \tag{2}$$

The hash, the first element between brackets, can be packed into a $32bit$ integer. To save space, f_1 and $t_3 - t_1$ can be combined in one $32bit$ integer. An integer of $32bit$ is also used to store t_1. The reference audio identifier is also a $32bit$ identifier. A complete fingerprint consists of $4 \times 32bit = 128bit$. At eight fingerprints per second a song of four minutes is reduced to $128bit \times 8 \times 60 \times 4 = 30kB$. An industrial size data set of one million songs translates to a manageable $28GB$ [2] .

2.6 Matching Algorithm

The matching algorithm is inspired by [13], but is heavily modified to allow time stretched and pitch-shifted matches. It follows the scheme in Figure 1 and has seven steps.

1. Local maxima are extracted from a constant-Q spectrogram from the query. The local maxima are combined by three to form fingerprints, as explained in Sections 2.1, 2.3 and 2.4.

2. For each fingerprint a corresponding hash value is calculated, as explained in Section 2.5.

3. The set of hashes is matched with the hashes stored in the reference database, and each exact match is returned.

4. The matches are iterated while counting how many times each individual audio identifier occurs in the result set.

5. Matches with an audio identifier count lower than a certain threshold are removed, effectively dismissing random chance hits. In practice there is almost always only one item with a lot of matches, the rest being random chance hits. A threshold of three or four suffices.

[1] It is assumed that the time stretch factor is constant in the time interval $t'_3 - t'_1$. A reasonable assumption since $t'_3 - t'_1$ is small.

[2] Depending on the storage engine used, storage of fingerprints together with an index of sorts introduces a storage overhead. Since the data to store is small, the index can be relatively large.

6. The residual matches are checked for alignment, both in frequency and time, with the reference fingerprints using the information that is stored along with the hash.

7. A list of audio identifiers is returned ordered by the amount of fingerprints that align both in pitch and frequency.

In step six, frequency alignment is checked by comparing the f_1 component of the stored reference with f_1', the frequency component of the query. If, for each match, the difference between f_1 and f_1' is constant, the matches align.

Alignment in time is checked using the reference time information t_1 and $t_3 - t_1$, and the time information of the corresponding fingerprint extracted from the query fragment t_1', $t_3' - t_1'$. For each matching fingerprint the time offset t_o is calculated. The time offset t_o resolves to the amount of time steps between the beginning of the query and the beginning of the reference audio, even if a time modification took place. It stands to reason that t_o is constant for matching audio.

$$t_o = t_1 - t_1' \times \frac{(t_3 - t_1)}{(t_3' - t_1')} \tag{3}$$

The matching algorithm also provides information about the query. The time offset tells at which point in time the query starts in the reference audio. The time difference ratio $(t_3 - t_1)/(t_3' - t_1')$ represents how much time is modified, in percentages. How much the query is pitch-shifted with respect to the reference audio can be deduced from $f_1' - f_1$, in frequency bins. To convert a difference in frequency bins to a percentage the following equation is used, with n the number of cents per bin, e Eulers number, and ln the natural logarithm: $e^{((f_1' - f_1) \times n \times ln(2)/1200)}$

The matching algorithm ensures that random chance hits are very uncommon, the number of false positives can be effectively reduced to zero by setting a threshold on the number of aligned matches. The matching algorithm also provides the query time offset and the percentage of pitch-shift and time-scale modification of the query with respect to the reference audio.

3. RESULTS

To test the system, it was implemented in the Java programming language. The implementation is called Panako and is available under the *GNU Affero General Public License* on `http://panako.be`. The DSP is also done in Java using a DSP library [12]. To store and retrieve hashes, Panako uses a key-value store. Kyoto Cabinet, BerkeyDB, Redis, LevelDB, RocksDB, Voldemort, and MapDB were considered. MapDB is an implementation of a storage backed B-Tree with efficient concurrent operations [9] and was chosen for its simplicity, performance and good Java integration. Also, the storage overhead introduced when storing fingerprints on disk is minimal. Panako is compared with Audfprint by Dan Ellis, an implementation of a fingerprinter system based on [13].

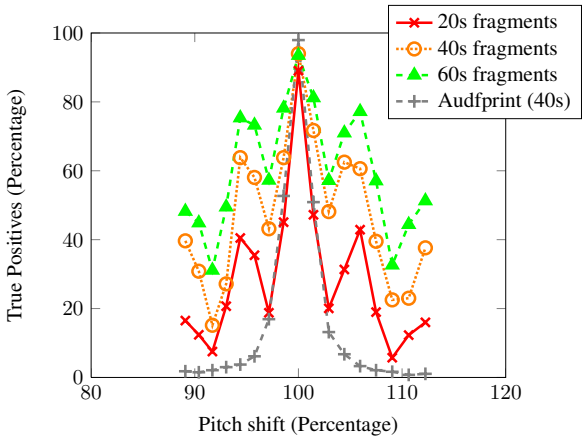

Figure 3: True positive rate after pitch-shifting. Note the fluctuating effect caused by the Constant-Q frequency bins.

The test data set consists of freely available music downloaded from Jamendo [3]. A reference database of about 30,000 songs, about 10^6 seconds of audio, was created. From this data set random fragments were selected, with a length of 20, 40 and 60 seconds. Each fragments was modified 54 times. The modifications included: pitch-shifting (-200 tot 200 cents in steps of 25 cents), time-stretching (-16% to $+16\%$, in steps of 2%), time-scale modification (-16% to $+16\%$, in steps of 2%), echo, flanger, chorus and a band-pass filter [4]. Another set of fragments were created from audio not present in the reference database, in order to measure the number of correctly unidentified fragments. In total 3 $(durations)$ $\times$ 600 $(excerpts)$ $\times$ 54 $(modifications)$ $= 97,200$ fragments were created.

Each fragment is presented to both Panako and Audfprint and the detection results are recorded. The systems are regarded as binary classifiers of which the amount of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN) are counted. During the experiment with Panako no false positives (FP) were detected. Also, all fragments that are not present in the reference database were rejected correctly (TN). So Panako's specificity is $TN/(TN + FP) = 100\%$. This can be explained by the design of the matching algorithm. A match is identified as such if a number of hashes, each consisting of three points in a spectrogram, align in time. A random match between hashes is rare, the chances of a random match between consecutively aligned hashes is almost non-existent, resulting in 100% specificity.

The sensitivity $FP/(TP + FN)$ of the system, however, depends on the type of modification on the fragment. Figure 3 shows the results after pitch-shifting. It is clear that the amount of pitch-shift affects the performance, but

[3] `http://jamendo.com` is a website where artists share their work freely, under various creative commons licenses. To download the data set used in this paper, and repeat the experiment, please use the scripts provided at `http://panako.be`.

[4] The effects were applied using SoX, a command line audio editor. The scripts used to generate the queries can be found at the website `http://panako.be`

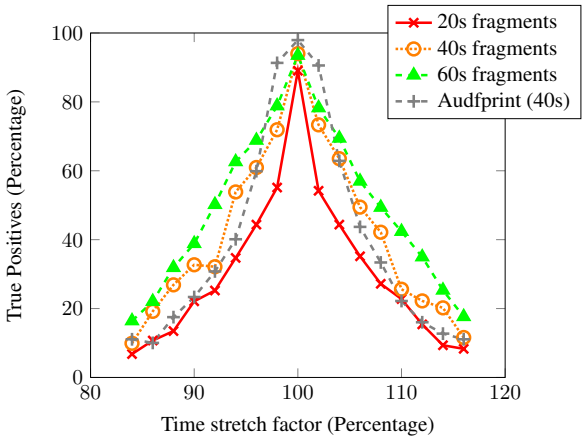

Figure 4: The true positive rate after time-stretching.

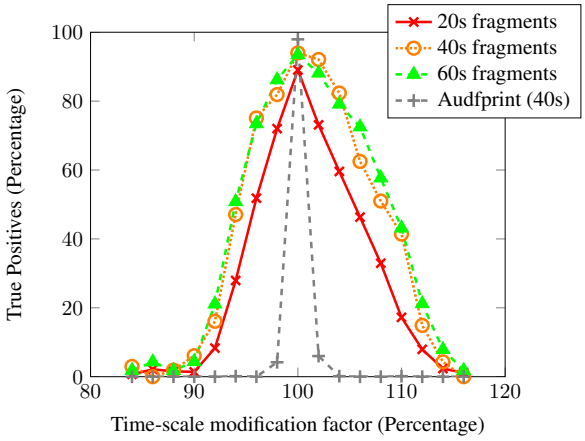

Figure 5: True positive rate after time-scale modification

in a fluctuating pattern. The effect can be explained by taking into account the Constant-Q bins. Here, a bin spans 33 cents, a shift of $n \times 33/2$ cents spreads spectral information over two bins, if n is an odd number. So performance is expected to degrade severely at ± 49.5 cents (3%) and ± 148.5 cents (9%) an effect clearly visible in figure 3. The figure also shows that performance is better if longer fragments are presented to the system. The performance of Audfprint, however, does not recover after pitch-shifts of more than three percent.

Figure 4 shows the results after time stretching. Due to the granularity of the time bins, and considering that the step size stays the same for each query type, time modifications have a negative effect on the performance. Still, a more than a third of the queries is resolved correctly after a time stretching modification of 8%. Performance improves with the length of a fragment. Surprisingly, Audfprint is rather robust against time-stretching, thanks to the way time is encoded into a fingerprint.

Figure 5 shows the results after time-scale modification. The performance decreases severely above eight percent. The figure shows that there is some improvement when comparing the results of 20s fragments to 40s fragments, but going from 40s to 60s does not change much. Audiofprint is unable to cope with time-scale modification due to the changes in both frequency and time.

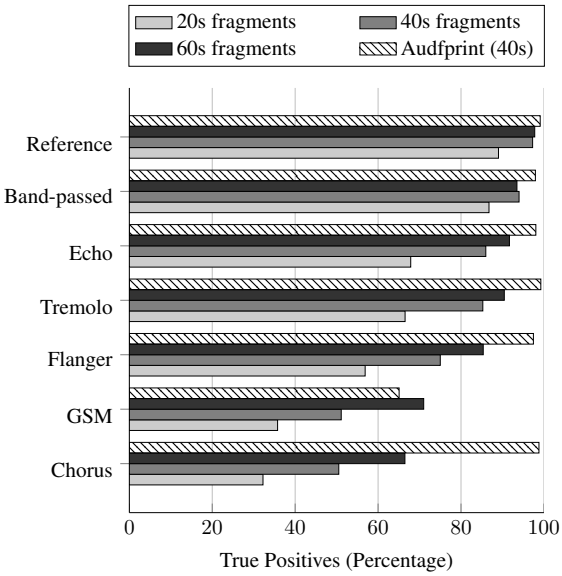

Figure 6: Effect of several attacks on true positive rate.

In Figure 6, the results for other modifications like echo, chorus, flanger, tremolo, and a band pass filter can be seen. The parameters of each effect are chosen to represent typical use, but on the heavy side. For example the echo effect applied has a delay line of 0.5 seconds and a decay of 30%. The system has the most problems with the chorus effect. Chorus has a blurring effect on a spectrogram, which makes it hard for the system to find matches. Still it can be said that the algorithm is rather robust against very present, clearly audible, commonly used audio effects. The result of the band pass filter with a center of 2000Hz is especially good. To test the systems robustness to severe audio compression a test was executed with GSM-compressed queries. The performance on 20s fragments is about 30% but improves a lot with query length, the 60s fragment yields 65%. The results for Audfprint show that there is room for improvement for the performance of Panako.

A practical fingerprinting system performs well, in terms of speed, on commodity hardware. With Panako extracting and storing fingerprints for 25s of audio is done in one second using a single core of a dated processor [5] The test data set was constructed in $30,000 \times 4 \times 60s/25 = 80$ processor hours. Since four cores were used, it took less than a full day. After the feature extraction, matching a $40s$ query with the test database with $30,000$ songs is done within $75ms$. The complete matching process for a $40s$ fragment takes about one second. Monitoring multiple streams in real-time poses no problem for the system. Building a fingerprint dataset with Audfprint is faster since fingerprints are extracted from an FFT which is less demanding than a Constant-Q transform. The matching step performance, however, is comparable.

[5] The testing machine has an Intel Core2 Quad CPU Q9650 @ 3.00GHz introduced in 2009. The processor has four cores.

Failure analysis shows that the system does not perform well on music with spectrograms either with very little energy or energy evenly spread across the range. Also extremely repetitive music, with a spectrogram similar to a series of dirac impulses, is problematic. Also, performance drops when time modifications of more than 8% are present. This could be partially alleviated by redesigning the time parameters used in the fingerprint hash, but this would reduce the discriminative power of the hash.

4. CONCLUSION

In this paper a practical acoustic fingerprinting system was presented. The system allows fast and reliable identification of small audio fragments in a large set of audio, even when the fragment has been pitch-shifted and time-stretched with respect to the reference audio. If a match is found the system reports where in the reference audio a query matches, and how much time/frequency has been modified. To achieve this, the system uses local maxima in a Constant-Q spectrogram. It combines event points into groups of three, and uses time ratios to form a time-scale invariant fingerprint component. To form pitch-shift invariant fingerprint components only frequency differences are stored. For retrieval an exact hashing matching algorithm is used.

The system has been evaluated using a freely available data set of 30,000 songs and compared with a baseline system. The results can be reproduced entirely using this data set and the open source implementation of Panako. The scripts to run the experiment are available as well. The results show that the system's performance decreases with time-scale modification of more than eight percent. The system is shown to cope with pitch-shifting, time-stretching, severe compression, and other modifications as echo, flanger and band pass.

To improve the system further the constant-Q transform could be replaced by an efficient implementation of the non stationary Gabor transform. This is expected to improve the extraction of event points and fingerprints without effecting performance. Panako could also benefit from a more extensive evaluation and detailed comparison with other techniques. An analysis of the minimum , most discriminative, information needed for retrieval purposes could be especially interesting.

5. REFERENCES

[1] Eric Allamanche. Content-based identification of audio material using mpeg-7 low level description. In *Proceedings of the 2nd International Symposium on Music Information Retrieval (ISMIR 2001)*, 2001.

[2] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. Fast identification of piece and score position via symbolic fingerprinting. In Fabien Gouyon, Perfecto Herrera, Luis Gustavo Martins, and Meinard Mller, editors, *Proceedings of the 13th International Symposium on Music Information Retrieval (ISMIR 2012)*, pages 433–438, 2012.

[3] Carlo Bellettini and Gianluca Mazzini. Reliable automatic recognition for pitch-shifted audio. In *Proceedings of 17th International Conference on Computer Communications and Networks (ICCCN 2008)*, pages 838–843. IEEE, 2008.

[4] Judith Brown and Miller S. Puckette. An Efficient Algorithm for the Calculation of a Constant Q Transform. *Journal of the Acoustical Society of America*, 92(5):2698–2701, November 1992.

[5] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. A review of audio fingerprinting. *The Journal of VLSI Signal Processing*, 41:271–284, 2005.

[6] Dan Ellis, Brian Whitman, and Alastair Porter. Echoprint - an open music identification service. In *Proceedings of the 12th International Symposium on Music Information Retrieval (ISMIR 2011)*, 2011.

[7] Sébastien Fenet, Gaël Richard, and Yves Grenier. A Scalable Audio Fingerprint Method with Robustness to Pitch-Shifting. In *Proceedings of the 12th International Symposium on Music Information Retrieval (ISMIR 2011)*, pages 121–126, 2011.

[8] Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *Proceedings of the 3th International Symposium on Music Information Retrieval (ISMIR 2002)*, 2002.

[9] Philip L. Lehman and s. Bing Yao. Efficient locking for concurrent operations on b-trees. *ACM Transactions Database Systems*, 6(4):650–670, 1981.

[10] Mani Malekesmaeili and Rabab K. Ward. A local fingerprinting approach for audio copy detection. *Computing Research Repository (CoRR)*, abs/1304.0793, 2013.

[11] M. Ramona and G. Peeters. AudioPrint: An efficient audio fingerprint system based on a novel cost-less synchronization scheme. In *Proceedings of the 2013 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2013)*, pages 818–822, 2013.

[12] Joren Six, Olmo Cornelis, and Marc Leman. TarsosDSP, a Real-Time Audio Processing Framework in Java. In *Proceedings of the 53rd AES Conference (AES 53rd)*. The Audio Engineering Society, 2014.

[13] Avery L. Wang. An Industrial-Strength Audio Search Algorithm. In *Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR 2003)*, pages 7–13, 2003.

[14] Bilei Zhu, Wei Li, Zhurong Wang, and Xiangyang Xue. A novel audio fingerprinting method robust to time scale modification and pitch shifting. In *Proceedings of the international conference on Multimedia (MM 2010)*, pages 987–990. ACM, 2010.