A RE-ORDERING STRATEGY FOR ACCELERATING INDEX-BASED AUDIO FINGERPRINTING

Hendrik Schreiber
tagtraum industries incorporated
hs@tagtraum.com

Peter Grosche
Saarland University and MPI Informatik
pgrosche@mpi-inf.mpg.de

Meinard Müller
Saarland University and MPI Informatik
meinard@mpi-inf.mpg.de

ABSTRACT

The Haitsma/Kalker audio fingerprinting system [4] has been in use for years, but its search algorithm's scalability has not been researched very well. In this paper we show that by simple re-ordering of the query fingerprint's sub-prints in the index-based retrieval step, the overall search performance can be increased significantly. Furthermore, we show that combining longer fingerprints with re-ordering can lead to even higher performance gains, up to a factor of 9.8. The proposed re-ordering scheme is based on the observation that sub-prints, which are elements of n-runs of identical consecutive sub-prints, have a higher survival rate in distorted copies of a signal (e.g. after mp3 compression) than other sub-prints.

1. INTRODUCTION

In 2002 Jaap Haitsma and Ton Kalker proposed their audio fingerprinting system [4], which today is still in use at Gracenote [3], competing with other commercial systems like Shazam [7, 8]. In this system, identity of two songs is established by comparing so called fingerprints. These fingerprints correspond to ca. 3 seconds of audio and are comprised of 256 sub-prints, each representing 11.6 milliseconds of audio with 32-bits. For a general overview of audio fingerprinting systems, we refer to [1].

To identify an unknown audio fragment (the query), fingerprints are extracted from the query and compared with fingerprints stored in a database. Typically, as the fragment is exposed to distortions such as additive noise or compression artifacts, one cannot assume to find an identical fingerprint in the database. Therefore, the similarity of two fingerprints is expressed in terms of the bit error rate (BER). The lower the BER, the more likely two fingerprints belong to the same song. If the BER is below a certain threshold ($\tau = 0.35$), both fingerprints are assumed to stem from the same song.

Comparing all query fingerprints for a song with all reference fingerprints is only feasible for databases containing a very limited number of recordings. Therefore, Haitsma/Kalker proposed an efficient two-step hashing scheme. In the first step, indexing techniques are employed to detect "anchor points" in the database. The idea is, that even though there typically is not an exact match for a whole query fingerprint in the database, at least one of the 256 sub-prints occurs unaltered (without any bit error) in query and reference fingerprints. Exploiting this idea, Haitsma/Kalker propose to use one 32-bit long sub-print of the query fingerprint at a time and query the reference database for identical sub-prints. The positions of exact matches of sub-prints in the database then serve as said anchor points. In the second step, the BER for entire fingerprints (consisting of 256 sub-prints) around these anchor points is calculated. If it turns out to be lower than the threshold, the search is terminated and the identified song returned.

As the number of reference fingerprint lookups and BER computations is considerably reduced by this strategy, this way of searching is multiple orders of magnitude faster than the naïve approach of comparing fingerprints with all reference fingerprints in the database. The lookup of potentially matching songs using unaltered sub-prints is a crucial step in this approach. To find them, the system maintains a lookup table with entries for each of the possible 32-bit sub-prints. Each entry points to a linked list of songs the given sub-print occurs in and the position of the sub-print within this song (Figure 1). Obviously the system is faster, if it finds a matching fingerprint in as few sub-print lookups as possible.

In this paper, we propose an extension to the original algorithm. Our main idea is to re-order the lookup of unaltered sub-prints in such a way that those sub-prints more likely to survive compression distortions are looked up first. In our experiments, we show that this is the case for sub-prints, which are elements of n-runs of identical consecutive sub-prints (Figure 2). Exploiting this property in a sim-

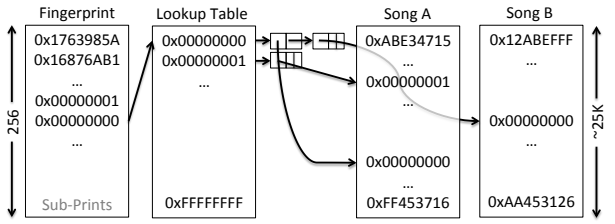

Figure 1. Lookup strategy for potentially matching songs and their reference sub-prints as suggested in [4].

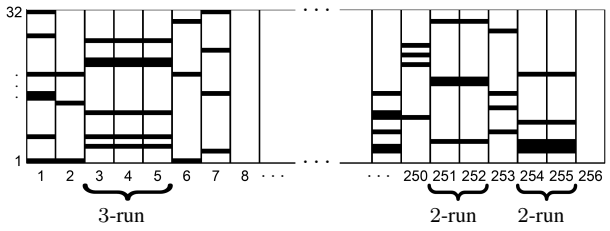

Figure 2. Illustrative example showing one fingerprint consisting of 256 sub-prints. The fingerprint exhibits runs of identical sub-prints.

ple re-ordering scheme leads to significant speed-ups of the search algorithm. In a second step, we apply the re-ordering scheme to fingerprints longer than 256 sub-prints achieving even higher improvements up to a factor of 9.8.

The remainder of this paper is organized as follows. In Section 2 we motivate our re-ordering scheme by investigating the distribution of sub-prints and their likelihood of surviving compression distortions. Then, in Section 3, as our main contribution, we introduce in detail the re-ordering scheme. In Section 4 we give experimental evidence for the speed-up of our approach in a real-world runtime analysis. Finally, conclusions and outlook on future work are given in Section 5.

2. SUB-PRINT PROPERTIES

In this section, we explain in detail the computation of the fingerprints as proposed in [4] (Section 2.1). Then, in Section 2.2, we show that these fingerprints are strongly correlated over time by analyzing audio recordings of three datasets of different genres. Finally, in Section 2.3, we show that the temporal correlation can be exploited for identifying more robust sub-prints.

2.1 Computation

Following [4], we compute the sub-prints from a given audio signal in three steps. In the first step, a spectrogram is derived from the audio. To this end, discrete Fourier transforms are computed over Hann-windowed frames corresponding to 0.37 sec of the audio. These frames overlap by a factor of $31/32$ yielding one frame for every 11.6 ms. In the second step, by suitably pooling spectral coefficients, the frequency axis of the spectrogram is adapted to the human auditory system. More precisely, energy values are computed for 33 non-overlapping spectral bands. These bands are logarithmically spaced and cover the frequency range from 300 Hz to 2000 Hz. Finally, in the third step, fingerprints are derived. Given the energy in frame $t \in [0:T] := \{0,1,2,\ldots,T\}$ for some $T \in \mathbb{N}$ and spectral band $k \in [1:33]$ denoted by $E(t,k)$, we first compute energy differences $\Delta(t,k)$ along the frequency axis $\Delta(t,k) = E(t,k) - E(t,k+1)$ for all $t \in [0:T]$ and $k \in [1:32]$. Then, the energy values are quantized in order to obtain a binary representation $X \in \{0,1\}^{T\times 32}$ by determining the sign of energy differences along the time axis

$$X(t,k) = \begin{cases} 1 & \text{if } \Delta(t,k) > \Delta(t-1,k) \\ 0 & \text{otherwise}\,, \end{cases}$$

for $t \in [1:T]$. Let $X[t] \in \{0,1\}^{32}$ denote the t^{th} column of X. Following [4], such a binary vector is also referred to as *sub-print*. Furthermore, fixing a length parameter K (in the following we use $K = 256$), a binary block $F \in \{0,1\}^{K\times 32}$ is referred to as *fingerprint* consisting of the sub-prints $F[k]$, $k \in [1:K]$. Each of the sub-prints represents 11.6 ms of the audio with 32-bit, see Figure 2 for a schematic illustration of a fingerprint.

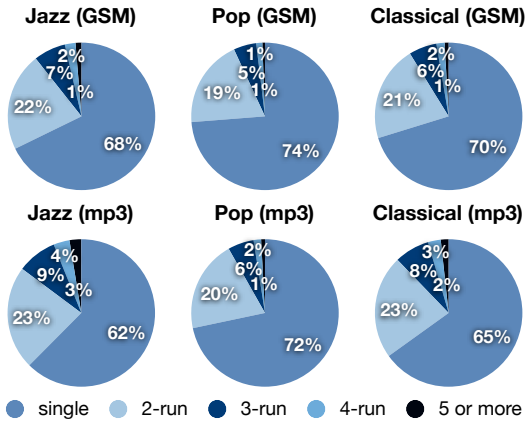

Figure 3. Percentages of sub-prints occurring as singles, or in higher runs in mp3 and GSM files from genre-specific RWC collections (first 3 min). The distributions for the WAV encoded reference files are almost identical.

2.2 Temporal Correlation

As pointed out in [4], because of the high amount of overlap between adjacent frames, the sub-prints are temporally correlated. In fact, they are correlated so strongly that often one

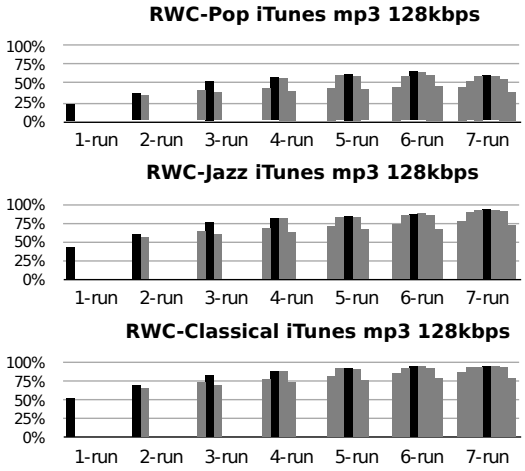

Figure 4. Probability of an iTunes mp3 encoded sub-print at a given position m in an n-run having an identical counterpart in a reference fingerprint depending on the length of the n-run it belongs to.

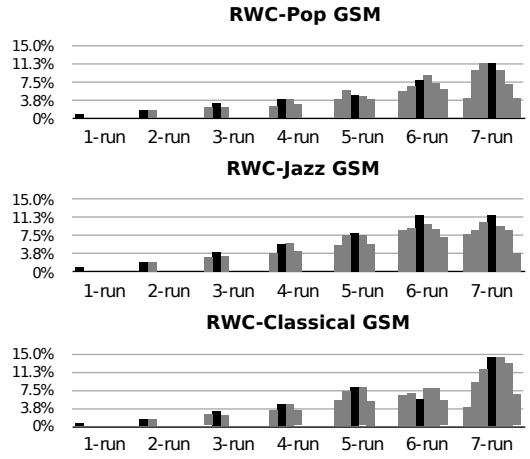

Figure 5. Probability of a GSM encoded sub-print at a given position in an n-run having an identical counterpart in a reference fingerprint depending on the length of the n-run it belongs to.

sub-print is followed by one or more identical sub-prints. We call such a sequence of identical consecutive sub-prints an n-run (see Figure 2). For the remainder of this paper, n-runs with $n = 1$ will also be called *singles* and n-runs with $n > 1$ are referred to as *higher* runs.

To better understand the temporal correlation of sub-prints, we analyzed a large collection of audio recordings of various genres with respect to the occurrence of n-runs. Specifically, we used the sub-collections RWC-Jazz, RWC-Pop, and RWC-Classical provided by the RWC Music Database [2]. Overall, there are 211 recordings with a total duration of 16 hours. We consider each of these recordings in three different versions of different quality. Firstly, we refer to the CD quality versions denoted as *reference* versions. Furthermore, we consider two encoded (*distorted*) versions derived from the reference employing lossy audio codecs. As a mildly compressed version, we use an mp3 version encoded with 128 kbps using iTunes. This version can be regarded to be of "standard" quality. Finally, as a heavily compressed version of poor audio quality, we encode the reference versions using the (full rate) 13kbps GSM Voice codec. Originally intended for the compression of speech signals, this codec introduces severe audible distortions to music signals. This version is included in our analysis as an extreme case.

Figure 3 shows the percentage of sub-prints that occur in an n-run of identical sub-prints for versions encoded with iTunes mp3 128kbps or GSM. As our results show, the sub-prints are temporally correlated. For all three datasets and both encodings about 30% of all sub-prints occur in higher runs. For example, in the case of RWC-Jazz (mp3), 23% are a member of a 2-run, 9% of a 3-run, 4% of a 4-run, and 3% of a 5-run or even longer run.

2.3 Robustness

We hypothesize that such sub-prints occurring in n-runs are more likely to have unaltered counterparts in distorted versions of the same song than sub-prints occurring on their own, i.e. as singles. In other words, as the survival rate of sub-prints in higher runs is higher, they are more robust.

To test this hypothesis we measure the probability of a sub-print extracted from a distorted version being identical to its reference counterpart. This is done by computing sub-prints of the first 3 min for both reference and distorted versions. Because some distortions introduce a minor, linear frame-shift (± 1), we then align both sub-print lists so that as many as possible sub-prints are directly opposite an identical counterpart. This we call optimal alignment. Subsequently, we categorize the distorted sub-prints as members of n-runs along with their position $m \in [1 : n]$ in the run and record how many sub-prints of each category have unaltered, aligned counterparts in the reference sub-print list.

Figures 4 and 5 show the probabilities of sub-prints that are members of an n-run (column group) at position m (column) having an unaltered counterpart in the reference fingerprint, i.e. their survival probabilities. Our results for iTunes encoded mp3 audio (Figure 4) clearly show that sub-prints belonging to higher runs are more likely to have an unaltered counterpart in the reference fingerprint than sub-prints occurring as singles. Taking the RWC-Pop values as example, a single sub-print has a relatively low survival

probability of 23%, while a member of a 6-run has a survival probability between 45% and 65%. Note that, because of the low value for singles, the relative survival rate gain from singles to higher runs is larger for the pop songs than for jazz or classical music. For example, the maximal possible gain factor from single to 6-run for RWC-Pop is $\times 2.8$, while the gain factor for the RWC-Classical from the single with 52% to the maximal 6-run with 95% is only $\times 1.8$.

For GSM encoded files the effect becomes even more significant (Figure 5). Here, the survival probabilities are much lower, e.g., 0.7% for singles in the case of RWC-Classical. For 5-runs this probability increases to values between 5.1% and 8.0%, a gain factor of $\times 11.4$.

As a second important result, we observe, that in most cases those members of n-runs, that take a central position in their run, are even more likely to have an identical reference counterpart than n-run elements at edge positions. For example, in the case of RWC-Pop (mp3), see Figure 4, sub-prints at edge positions of a 6-run have a survival probability of 45%. For sub-prints at center position, however, this probability is significantly higher at 65%. In the case of RWC-Classical (GSM), see Figure 5, sub-prints at edge positions of a 5-runs survive with 5.7% probability, sub-prints at the center position, however, with 8.0%. We define this central position in an n-run as $m_{central} = \lfloor n/2 \rfloor + 1$. In Figures 4 and 5 it is shown in black.

3. IMPROVING THE SEARCH ALGORITHM

In this section, we first explain the original Haitsma/Kalker lookup algorithm. Then we describe our proposed improvements, which are based on the increased robustness of sub-prints contained in higher runs. Finally, we present an experiment that measures actually achieved overall speedups validating our chosen approach.

3.1 Original Algorithm

Suppose we are given a database containing a large number of audio documents, which are converted into binary representations as described in Section 2.1. Then, given a query fingerprint $F_Q \in \{0,1\}^{K \times 32}$, the identification task consists of finding a document with binary representation X as well as a position t such that the fingerprint defined by $F_D := (X[t], \ldots, X[t+K-1])$ is similar to F_Q. More precisely, as in [4], we require that the bit error rate (BER) between F_Q and F_D is below a threshold $\tau = 0.35$. We then also say that F_D is a *match* for F_Q.

To avoid an exhaustive fingerprint search in the database, an index-based pre-processing step is used to cut down the search space. Here, the binary representations of all database documents are indexed by means of the 32-bit sub-prints using an index structure that consists of a suitable lookup table as illustrated by Figure 1. Then, based on the assumption that at least one sub-print $F_Q[k]$, $k \in [1:K]$, of the query appears unaltered in the document to be identified, a lookup is performed to first retrieve all sub-prints that coincide with $F_Q[k]$. Each of these retrieved candidate sub-print consists of a document identifier and a position parameter t. Let X be the binary representation of the corresponding document, then the BER is computed between F_Q and $F_D := (X[t-k+1], \ldots, X[t-k+K])$. If the BER falls below the threshold $\tau = 0.35$, the algorithm terminates and returns the associated document identifier. If no such F_D can be found, the algorithm terminates without identifying the query.

Since a position k, that corresponds to an unaltered sub-print in the database, is not known a-priori, in [4] an outer loop is executed querying the index structure for sub-prints $F[k]$ in the order in which they appear in F_Q, i.e. with indices $k = 1, 2, 3, \ldots, K$. This loop is aborted as soon as a matching fingerprint is found. Therefore, the overall running time of the algorithm crucially depends on the position of the index at which an unaltered sup-print of a matching fingerprint occurs for the first time.

3.2 Sub-Print Re-Ordering

To take advantage of the observed sub-print properties, we change the order in which sub-prints are looked up in the database. Instead of simply iterating through the sub-prints of the query fingerprint from beginning to end, we prioritize those sub-prints that are more likely to lead to matching fingerprints. This means that we need to look up the central sub-prints of higher runs first, ordered by the length of the run they belong to in descending order. Then we look up the singles and then all remaining sub-prints, again ordered by the length of the run they belong to.

Figure 6 shows an example for this re-ordering scheme. Because the longest run is the 3-run, we rank its central element (3) first. The second longest run is the 2-run, thus its central element (8) lands on rank 2. Since there are no other higher runs, we then proceed to add all singles in the order in which they appear. And eventually, we add the remaining sub-prints from the two higher runs (2, 4 and 9).

The idea behind this is, that if a central sub-print does not lead to a match, it is more likely that another central sub-print leads to a match (even if it is a member of a shorter n-run) than a non-central sub-print of an n-run we already know of that its central sub-print does not match.

More formally, for a fingerprint F_Q of length $K = 256$ we calculate the $rank(k,m,n)$ with $k,m,n \in [1:K]$ of each sub-print $F_Q[k]$ that is the m^{th} element of an n-run, and order all sub-prints according to their rank in descending order. The $rank$ function is defined as

$$rank(k,m,n) = k + Km + K^2 n + K^3 n \delta_{m-1,\lfloor n/2 \rfloor} \quad (1)$$

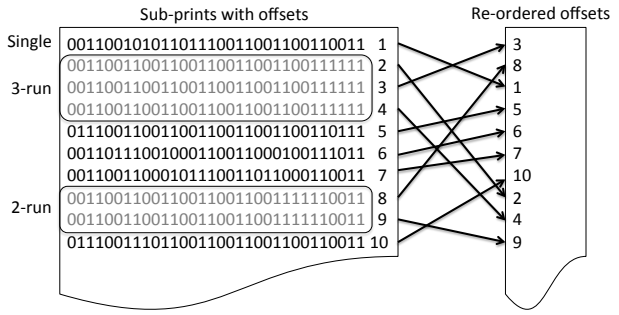

Figure 6. Optimization of the sub-print order.

Query	iTunes 128kbps	Lame 32kbps	GSM
Orig.	4.15	36.59	100.84
256	1.41 (×2.9)	12.82 (×2.9)	67.62 (×1.5)
512	1.31 (×3.2)	8.90 (×4.1)	60.58 (×1.7)
1024	1.25 (×3.3)	6.48 (×5.6)	43.79 (×2.3)
2048	1.22 (×3.4)	5.06 (×7.2)	27.04 (×3.7)
4096	1.20 (×3.5)	4.17 (×8.8)	18.08 (×5.6)
8192	1.24 (×3.3)	3.73 (×9.8)	14.30 (×7.0)

Table 1. Average number of sub-print lookups until a sub-print match is found, depending on query distortion and fingerprint length (based on 100,000 randomly selected queries). Denoted in parentheses are the factors between the optimized and the original approach.

and consists of four terms, each containing a weight factor based on K. The last term is only $\neq 0$ if and only if the sub-print is central. In that case, the term dominates the outcome of the function. If it is not central, the length n of the run becomes the deciding factor, as $K^2 n$ will be greater than the two remaining terms k and Km. Amongst sub-prints belonging to the same run length n, position m within the run and k in the fingerprint become tie-breakers.

Additionally to re-ordering, in a second optimization step, we also use a longer query fingerprint. This did not make sense before optimizing the lookup order, as we were not able to recognize more robust sub-prints. But with the suggested re-ordering scheme, enlarging the fingerprint increases our chances of finding more and longer n-runs, therefore significantly increasing our chances of finding a surviving sub-print counterpart in the reference data.

A side effect of this strategy is the necessity of computing a larger query fingerprint, which puts some additional computational burden on the client and requires a longer audio fragment. For the BER computation we still only use 256 sub-prints as there is nothing to be gained by using more sub-prints.

3.3 Experimental Verification

To test both approaches, sub-print re-ordering and fingerprint enlargement, we measured the average number of sub-print lookups needed to find a matching fingerprint in a database of 200 songs for 100,000 randomly selected queries.

Note, that in this experiment we focus on sub-print comparisons, not full fingerprint comparisons. Therefore, the number of songs in the database is irrelevant. Nevertheless, sub-print matches lead to fingerprint comparisons. In order to measure the total search time, those also need to be taken into account, if the number of song pointers per sub-print is not distributed uniformly. For the purpose of this experiment we assume a uniform distribution, in particular one that is independent from the used $rank$ function.

The results in Table 1 show that with our optimization scheme between 1.5 and 9.8 times fewer lookups are necessary. Even without enlarging the query fingerprint, we were still able to achieve 2.9 times fewer iterations for mp3 files encoded at 128kbps. This equates to only 1.41 sub-print lookups on average.

It also deserves to be mentioned that for audio data with stronger distortions (e.g. GSM, mp3 32kbps) our techniques tend to yield larger benefits. One reason for this is that for strongly distorted audio material many more lookups are necessary when no re-ordering is used (100.84 for GSM as opposed to 4.15 for mp3 128kbps).

4. RUNTIME ANALYSIS

Why do we care so much about the sub-print lookups? Realistically, a large scale audio fingerprinting system will have to be able to manage not just 10,000 songs [4], but rather 100 million songs—perhaps even more. [1] Assuming $\pm 25,000$ sub-prints per song, this results in a total of $25,000 \cdot 10^8 = 25 \cdot 10^{11}$ sub-prints. This means that the lookup table proposed by Haitsma/Kalker is not sparsely populated as they claim, but on average each entry contains a list of pointers to 582 ($= 25 \cdot 10^{11}/2^{32}$) songs. Assuming that each of these pointers has at least a size of 4 bytes to reference a song, plus an offset into the song's reference fingerprint of 2 bytes, we must manage roughly $2^{32} \cdot (4+2) \cdot 582$ bytes $= 15$ terabytes for the pointer lists alone. Obviously, with current technology, we cannot simply load the data-structure into the main memory of a regular PC.

Instead, just like the songs' sub-prints, the data-structure also has to live in secondary storage (e.g. flat files, a relational database management system (RDBMS), a no-SQL database, or a simple Berkeley DB). In the case of an

[1] In May 2011 MusicBrainz [6] stated on its website to have more than 10 million tracks in its database. This number is probably going to increase significantly as the years go by.

Subprints	
int	id
int	subprint
smallint	offset

Figure 7. Trivial table design for the reference song lookup with an RDBMS.

RDBMS, a single table containing the columns (song-)*id* (at least 4 bytes), *sub-print* (4 bytes) and (sub-print-)*offset* (at least 2 bytes) is sufficient (Figure 7). One database index on the *sub-print* column and another on *id* and *offset* ensure fast access. [2] Assuming the aforementioned setup, the measurable runtime behavior of the algorithm is governed by three main factors:

1. Number of songs in the database.

2. Speed of lookup from secondary storage.

3. Probability of a query sub-print having an identical reference counterpart.

Note that only the number of fingerprint lookups and BER computations depend directly and linearly on the number of songs in the database. This means that the overall runtime is linear with respect to the size of the database.

As for the secondary storage, even though solid state drives and the decreasing price of RAM slightly blur the lines, accessing secondary storage still takes much more time than performing relatively simple arithmetic operations like computing a BER. Therefore we can safely assume that each SQL-*select* operation to look up a fingerprint in the database takes orders of magnitude longer than the associated BER computation. Besides the collection's size, secondary storage access is therefore a determining factor for the absolute runtime of the algorithm.

Finally, how many times we have to look up complete fingerprints and access secondary storage depends highly on the probability that a given query sub-print has an identical reference counterpart. As shown above, we can significantly increase the probability of finding an identical subprint quickly by re-ordering the sub-prints. This is a deciding factor for the runtime of this algorithm and unlike the other two mentioned factors it has nothing to do with available hardware or the size of the problem.

[2] For 100 million songs, this database design leads to the impressive storage requirement of 25 terabytes ($= (4 + 4 + 2) \cdot 25 \cdot 10^{11}$ bytes), plus additional space for the indices. Not surprisingly, Haitsma/Kalker attempted to reduce this by sub-sampling reference fingerprints [5].

5. CONCLUSION

In this paper we presented an optimization scheme of the Haitsma/Kalker audio fingerprinting search algorithm. The suggested approach exploits strong temporal correlations between sub-prints as an indicator for sub-print robustness. This can lead to significant savings in the number of required lookups leading to a significant overall speed-up for the identification task.

Future research may focus on applying the proposed strategy on other existing algorithms or creating new ones, in which only reliable sub-prints are taken into account to begin with, which may lead to shorter, more robust fingerprints and reduced overall storage requirements. Also the combination of our re-ordering strategy with the reliability considerations proposed by Haitsma/Kalker is subject for future research.

Acknowledgement. P. Grosche and M. Müller are supported by the Cluster of Excellence on Multimodal Computing and Interaction at Saarland University.

6. REFERENCES

[1] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. A review of algorithms for audio fingerprinting. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 169–173, St. Thomas, Virgin Islands, USA, 2002.

[2] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical and jazz music databases. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 2002.

[3] Gracenote. `http://www.gracenote.com/`.

[4] Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 107–115, Paris, France, 2002.

[5] Jaap Haitsma, Ton Kalker, and Steven Schimmel. Efficient storage of fingerprints. US Patent 7,477,739, January 2009.

[6] MusicBrainz. `http://musicbrainz.org/`.

[7] Shazam. `http://www.shazam.com/`.

[8] Avery Wang. An industrial strength audio search algorithm. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 7–13, Baltimore, USA, 2003.