This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

Known-Artist Live Song Identification Using Audio Hashprints

TJ Tsai, *Member, IEEE,* Thomas Prätzlich, *Student Member, IEEE,* and Meinard Müller, *Senior Member, IEEE*

***Abstract*—The goal of live song identification is to allow concertgoers to identify a live performance by recording a few seconds of the performance on their cell phone. This article proposes a multi-step approach to address this problem for popular bands. In the first step, GPS data is used to associate the audio query with a concert in order to infer who the musical artist is. This reduces the search space to a dataset containing the artist's studio recordings. In the next step, the known-artist search is solved by representing the audio as a sequence of binary codes called hashprints, which can be efficiently matched against the database using a two-stage cross-correlation approach. The hashprint representation is derived from a set of spectro–temporal filters that are learned in an unsupervised, artist-specific manner. On the Gracenote live song identification benchmark, the proposed system outperforms five other baseline systems and improves the mean reciprocal rank of the previous state-of-the-art from .68 to .79, while simultaneously reducing the average runtime per query from 10 seconds to 0.9 seconds. We conduct extensive analyses of major factors affecting system performance.**

***Index Terms*—song identification, live performance, fingerprinting, cover song, audio matching.**

I. Introduction

THIS article tackles the problem of live song identification. A person goes to a live concert, hears a song that he or she likes, and wants to know: "What song is this?" Ideally, the person can simply open an app on his or her cell phone, record a few seconds of the performance, and get an answer. Even if the song is already known, such an app could provide a convenient way for concertgoers to purchase music instantly. While there are several commercially successful apps like Shazam and SoundHound that can identify pre-recorded music playing on the radio, the technology for identifying live music is lagging behind. This work offers a step towards bridging that gap.

Live song identification is a hybrid of two well-studied problems: audio fingerprinting and cover song detection.[1] Audio fingerprinting attempts to uniquely identify a segment of audio in a database of clean recordings. This is what Shazam does. Cover song detection attempts to identify cover versions of the same song. Each of these well-studied problems has certain factors that make the problem challenging and certain factors that make the problem easier. For audio fingerprinting, one major factor that makes the problem challenging is that the queries are short and noisy. Also, audio fingerprinting applications like Shazam often have to run in real-time, so runtime latency can be a big challenge. The factor that makes audio fingerprinting much easier is that such systems typically assume an *exact* match in the underlying source signals, possibly with some simple systematic distortions (like tempo change or room acoustics) or other additive sounds in the environment. For cover song detection, the factor that makes the problem challenging is that the match is very fuzzy — cover versions can differ in key, tempo, arrangement, and instrumentation, and these differences can make the matching problem much more subjective. One factor that makes cover song detection easier is that the problem is offline, so runtime latency is less of a factor. Also, cover song systems typically assume that the "query" is a clean studio recording of an entire song, so length of query and other additive sound sources and distortion are typically not an issue.

Live song identification is a hybrid of these two problems in the sense that it inherits the challenges from both. Like audio fingerprinting, the queries are short and noisy, and the system must run in real-time. Like cover song detection, the match is somewhat fuzzy. While the differences between an artist's live performance and studio recording may be less drastic than the differences between two different cover versions, the live song identification problem must nonetheless cope with differences in key, tempo, arrangement, and instrumentation that are typical of live performances.

Despite these similarities, the live song identification scenario violates the assumptions of these two problems. For example, audio fingerprinting systems are not designed to handle variations like a difference in instrumentation or a vocalist improvising on a melody. Likewise, cover song detection systems typically assume that a recording of the entire song is available, so that it is possible to analyze chord progressions and harmonies. Such an analysis is unlikely to work well on a short query that lasts only a few seconds long.

There has been a lot of previous work in audio fingerprinting and cover song detection. Audio fingerprinting has received a lot of interest in industry, with work coming out of companies such as Philips [2] [3], Google [4][5], Telefonica [6][7], and Gracenote [8]. There are several commercial applications for exact-match audio identification, such as Shazam, SoundHound, Viggle, and MusicID. Both tasks have also benefited from organized evaluations in the academic community. The TRECVID content based copy detection task [9], for example, spurred a lot of work in audio (and video)

T. Tsai is with the Department of Engineering at Harvey Mudd College, 301 Platt Blvd., Claremont, CA 91711. E-mail: ttsai@hmc.edu

T. Prätzlich and M. Müller are with the International Audio Laboratories Erlangen, Am Wolfsmantel 33, 91058 Erlangen, Germany. E-mail: thomas.praetzlich@audiolabs-erlangen.de, meinard.mueller@audiolabs-erlangen.de

Manuscript received April 19, 2005; revised August 26, 2015.

[1]Audio fingerprinting is also often referred to as audio identification or exact-match audio search. Cover song detection is also often referred to as version identification. See chapter 7 in [1] for a broad overview of both problems.

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

fingerprinting (see [9] for a list of participating teams and a comparison of all submitted systems). Likewise, the MIREX cover song evaluation [10] spurred progress on cover song detection [11][12][13][14]. The million song dataset [15] has also allowed researchers to explore cover song retrieval at a large scale [16][17][18][19][20].

There have also been a number of works in audio matching [21][22][23][24]. Audio matching is another hybrid of audio fingerprinting and cover song detection, where the goal is to find segments of audio that are musically similar to a short audio query. There are a few major differences between these approaches and the live song identification scenario considered in this article. One difference is that many of these works are offline tasks, where the fragment length is too long for a real-time application (10 to 30 seconds) or runtime latency is not considered. Another difference is that the query fragments are often extracted from clean studio recordings, rather than from noisy cell phone recordings. Yet another significant difference is that these approaches mostly focus on classical music, where the audience is generally very quiet (unlike at a rock concert).

In contrast, live song identification based on short cell phone queries is relatively new and unexplored. One major reason for this is the difficulty of collecting and annotating a suitable data set. Rafii et al. [25] collect a set of cell phone recordings of live concerts for ten popular bands, and they propose a method for song identification based on a binarized representation of the constant Q transform. To the best of our knowledge, this work is the only previous work that directly addresses our problem of interest. This previous work will serve as a baseline comparison to our proposed system.

In this work, we propose a solution to the live song identification problem that has two main components: (1) a binary representation of audio called hashprints, which are based on learning a set of spectro–temporal filters in an unsupervised, artist-specific manner, and (2) a simple, flexible matching algorithm that allows one to trade off accuracy for efficiency in order to accommodate the size of each artist's searchable database.[2]

The rest of the paper is organized as follows. Section II explains the proposed system. Section III discusses its relation to previous work. Section IV describes the evaluation of our system. Section V presents various analyses of interest. Section VI concludes the work. Note that the structure of this paper is a bit unusual in the sense that we describe the proposed system before discussing its relation to previous work. In discussing previous work, we explain how specific components of the proposed system fit into the existing literature. This ordering allows us to make these connections in a much more concrete and specific way.

[2]This article extends our earlier preliminary work [26]. Whereas our earlier work simply proposed the system and measured the overall results, this article provides insight into *why* the system works well. The analyses in V-A, V-B, V-C, V-D, V-E, and V-F are new contributions, and they tease apart the impact of various design choices on overall system performance. Another contribution of this work is the release of open source code to foster progress on this problem.

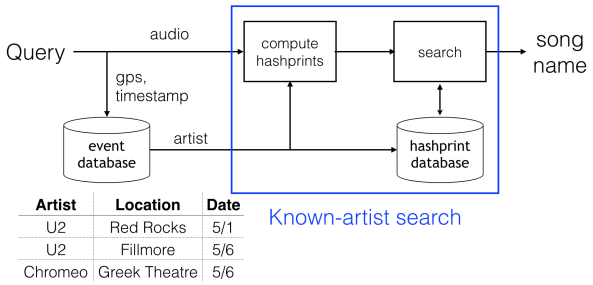


Fig. 1. System architecture of the live song identification system. Using GPS coordinates of the phone, the query is associated with a concert in order to infer who the artist is.

II. System Description

The proposed live song identification system will be described in three parts: the system architecture, the hashprint representation, and the search algorithm.[3]

A. System Architecture

Figure 1 shows the system architecture for the proposed system. When a query is first submitted, the GPS coordinates of the cell phone and the timestamp information are used to associate the query with a concert in order to infer who the artist is. Once the artist has been inferred, the problem is reduced to a known-artist search: we assume we know who the artist is, and we are trying to determine which song is being played. The known-artist assumption is critical to reduce the search space down to a manageable size. Whereas the space of all possible songs may contain millions of possibilities, the set of songs released by a single musical artist is at most a few hundred.[4] In many situations, it is possible to reduce the search space using metadata that is readily available. For the cell phone-based system shown in figure 1, we can infer the artist's identity by simply comparing the GPS coordinates of the phone to a database of current concert events. Since our evaluation data comes from concerts that have occurred in the past, in this article we will simply assume that such a database is available. This work will thus primarily focus on addressing the known-artist search problem.

The system in Figure 1 makes two assumptions. The first assumption is that the artist's concert schedule is available online and can be stored in the database of concert event information. This is necessary to correctly associate the query with a concert. The second assumption is that the artist performs a song from a recorded studio album. These two assumptions generally hold true for popular bands giving live performances. This system would *not* work, however, with an amateur musician performing at a local restaurant, since the concert event information would probably not be available online. The system would also fail if the artist debuts a new song at a live concert, since the system only searches for songs

[3]Source code can be found at http://pages.hmc.edu/ttsai/.

[4]This statement generally holds true for the genres of music considered in this work. For other genres such as jazz or classical music, the number of performed works may be larger.

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

from existing studio albums. This latter limitation is common to general fingerprinting systems.

B. Hashprint Representation

We represent audio as a sequence of binary fingerprints which we call hashprints. We will explain hashprints in three parts: the motivation behind the design, the mechanics of the computation, and the formulation of the filter learning problem.

Motivation. Similar to many other works (e.g. [27][2][6]), we use a binary representation of audio. Using a binary feature representation has two important benefits. The first benefit is compactness in memory. In our implementation we represent each hashprint as a single 64-bit integer containing up to 64 bits of information. Since we may have to store a large amount of audio data in the searchable database, compactness in memory is an important consideration. The second benefit is efficient distance computations. We can compute the Hamming distance between two hashprints very efficiently by peforming a single logical xor operation between two 64-bit integers, and then counting the number of 1 bits in the result. This can be done much faster than computing (say) the Euclidean distance between two vectors of floating point numbers. These computational savings will be important in reducing the latency of the live song identification system.

The hashprint representation grows out of two principles of good fingerprint design: compactness and robustness. Compactness means that the representation is efficient. Note that any imbalance in a bit or correlation between bits will result in an inefficient representation. So, each bit should be balanced (i.e. 0 half the time and 1 half the time) and the bits should be uncorrelated with each other. Robustness means that the individual bits are robust to noise. Within the context of thresholding a random variable, robustness means maximizing the variance of the random variable's probability distribution.[5] To see this, consider the scenario where the random variable takes on a value very close to the threshold (which will be at the median of the distribution in order to ensure a balanced bit). A little bit of noise may cause the random variable to fall on the wrong side of the threshold, resulting in an incorrect bit. We can reduce the probability of this happening by increasing the spread of the random variable's probability distribution. Robustness means high variance.

Hashprints are also designed to be volume-invariant. This means that an audio signal will yield the same hashprint representation even if it is multiplied by a constant factor. This is an important characteristic to have in the live song identification scenario, since the volume of the cell phone recording may not match the volume of the corresponding studio track.

Mechanics. Figure 2 shows the mechanics of how hashprints are computed. There are four steps, which are each described below.

[5]As we will see, the random variable will be a linear combination of many spectrogram log-energy values. Due to the central limit theorem, the probability distribution will thus be roughly bell-shaped.

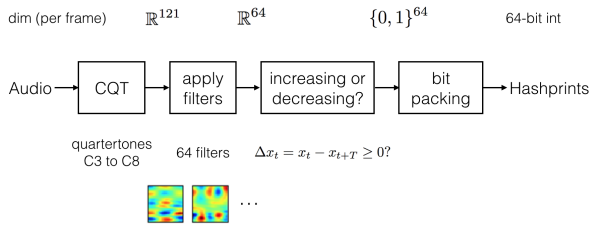


Fig. 2. Block diagram of hashprint computation.

The first step is to compute a constant Q transform (CQT). The CQT is a time-frequency representation of audio that uses a set of logarithmically spaced filters with constant Q factor.[6] One benefit of the CQT is that the filters can be selected to match the pitches of the Western musical scale. Using the CQT thus allows our representation to capture musically meaningful pitch-related information, and it also allows us to efficiently compute pitch-shifted versions of a signal. In our experiments, we used the CQT implementation by Schörkhuber and Klapuri [28]. Similar to the work by Rafii et al. [25], we consider 24 subbands per octave between C3 (130.81 Hz) and C8 (4186.01 Hz). Using a quarter-tone resolution allows us to handle slight tuning differences as well as key changes. To mimic the nonlinear processing of the human auditory system, we take the logarithm of the subband energy values. At the end of this step, we have 121 subband log-energy values every 12.4 ms.

The second step is to apply a set of filters. At each frame, we apply a set of $N = 64$ spectro–temporal filters in order to generate N real-valued spectro–temporal features. Each spectro–temporal feature is a linear combination of the CQT log-energy values from the current audio frame and several neighboring context frames. The weights of this linear combination are specified by the spectro–temporal filters. Note that many other fingerprinting approaches also use spectro–temporal filters (e.g. [2][29][6]). The difference between our proposed approach and previous work is that these spectro–temporal filters are learned in an *unsupervised* manner and adapted to each artist's studio recordings (i.e. the filters are learned per artist). The filters are selected to yield a set of uncorrelated random variables with maximum variance, which (as we discussed previously) yields robust bits. The formulation of this filter learning problem will be discussed in detail in the next subsection. At the end of this second step, we have N features per frame.

The third step is to determine whether each spectro–temporal feature is increasing or decreasing in time. This is done by computing delta spectro–temporal features at a separation of T seconds, and then thresholding the delta features at zero to ensure that the bits will be balanced. In our experiments, we used a value of $T = .992$ seconds, which was determined empirically (see section V-B). The purpose of thresholding on delta features is to achieve volume-invariance. Note that if we were to threshold on the spectro–temporal

[6]The Q factor of a filter is the ratio between the filter's center frequency and its bandwidth.

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

features directly, the resulting representation would not be volume-invariant. Some works achieve volume-invariance by using filters that are symmetric in time (e.g. [2][6]). We experimented with per-frame normalization as well, but found that delta features yield better performance. The reason for this is because, given a large enough separation T, a delta feature will effectively be the sum of two independent random variables. The delta feature will have twice the variance of a single random variable and thus be more robust to noise. Delta features also have the benefit of being extremely efficient to compute — assuming the spectro–temporal features have already been computed, we only need to perform one additional subtraction operation per feature per frame. At the end of this third step, we have N binary values per frame.

The fourth step is to package the $N = 64$ binary values per frame into a single 64-bit integer. This representation allows us to store hashprints very compactly in memory, and also to compute Hamming distances very efficiently using logical bitwise operators. The output of this fourth and final step is a sequence of hashprints, where each hashprint is represented by a single 64-bit integer.

Before moving on, it is useful to summarize what each of the four steps is doing. The first step transforms the audio signal into a time-frequency representation that allows us to access pitch-related information. The second step takes the pitch-related information and generates a set of uncorrelated random variables whose distributions have maximum variance. The third step maps these random variables to binary 0-1 values in a way that ensures that the bits will be balanced and the representation will be volume-invariant. The last step simply packages these binary values in memory in a compact way.

The only part of Figure 2 that still needs to be explained is how we learn the spectro–temporal filters. This will be explained in the next subsection.

Filter Learning Problem. The filters are selected to maximize the variance of the resulting spectro–temporal features, while ensuring that these features are uncorrelated. The filter learning problem can be formulated as a series of optimization problems, which are described below. Consider a vector $\in \mathbb{R}^{121w}$ containing the CQT log-energy values for an audio frame and its neighboring context frames, where w specifies the number of context frames. We can stack a bunch of these vectors into a large matrix $A \in \mathbb{R}^{M\times 121w}$, where M corresponds (approximately) to the total number of audio frames in a set of audio recordings. Let $S \in \mathbb{R}^{121w\times 121w}$ be the covariance matrix of A, and let $x_i \in \mathbb{R}^{121w}$ specify the coefficients of the i^{th} spectro–temporal filter. Given these definitions, note that applying a filter simply corresponds to an inner product between two $\mathbb{R}^{121w}$ vectors. Then, for $i = 1, \ldots, N$, we solve the following:

$$\begin{aligned} \text{maximize} \quad & x_i^T S x_i \\ \text{subject to} \quad & \|x_i\|_2^2 = 1 \\ & x_i^T x_j = 0,\ j = 1, \ldots, i-1. \end{aligned} \tag{1}$$

The objective function represents the variance of the spectro–temporal features generated by filter x_i. The second constraint guarantees that the filters will be uncorrelated with each other. This is exactly the eigenvector problem [30], for which very efficient off-the-shelf routines exist.

To summarize the filter learning problem, we first construct the data matrix A containing CQT log-energy values from the artist's studio tracks. We then compute the covariance matrix of A, and then compute the eigenvectors of the covariance matrix. These eigenvectors specify the spectro-temporal filters that we apply in the hashprint computation. We will compare this formulation to other learning-based approaches in section III.

C. Search Algorithm

During the offline portion of the search, we collect audio from the artist's studio albums, extract hashprints, and store the hashprint sequences into a database. In order to handle the possibility that the live performance may be performed in a different key (or a slightly different tuning) than the original studio track, we also consider pitch-shifted versions of the original studio tracks. In our experiments we consider up to four quartertones above and below the original key. So, for each song in the database, there will be nine hashprint sequences corresponding to the nine different pitch-shifted versions of the song.

During the online portion of the search, the query hashprint sequence is compared to the database to find a match. This online search is done by performing cross-correlation matching with downsampling and rescoring. Each of these three components is described below.

Cross correlation. For each hashprint sequence in the database, we determine the offset that maximizes the bit agreement rate between the query hashprint sequence and the corresponding portion of the reference hashprint sequence. This maximum bit agreement rate is used as the match score with the whole sequence. The maximum match score among the various pitch-shifted versions of a song is taken as the aggregate match score for the song. The songs in the database are then ranked by their aggregate match scores. Note that this approach assumes an approximately 1-to-1 tempo correspondence between the query and the studio album. We will investigate the validity of this assumption and also compare our results with a dynamic time warping approach in section V-A.

Downsampling. In order to speed up the online search, we downsample both the query and reference hashprint sequences by a factor B. So, for example, when $B = 2$ we only consider every other hashprint when computing bit agreement scores. Downsampling by a factor B thus reduces the amount of search computation by a factor of B^2. Our hope is that we can reduce the search time without affecting the accuracy too much. The effect of this downsampling will be investigated in section V-E.

Rescoring. After sorting the reference sequences by their rough (downsampled) match scores, we can rescore the top L sequences using the full hashprint sequences without downsampling. We can then resort these top L sequences by their fine-grained match scores. The idea of rescoring is to keep

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

the computation savings of a downsampling approach while retaining the reliability and performance of an exhaustive match without downsampling. The effect of this rescoring will be investigated in section V-E.

III. Relation to Previous Work

The design of the proposed system uses ideas and techniques that are drawn from a rich literature in audio fingerprinting, cover song detection, machine learning, and other related fields. In this section, we will explain how various aspects of the system design draw inspiration from and fit into the context of previous work. We will approach this from five different angles, each in its own section.

A. Binary Representation: Threshold-Based vs Value-Based

One key design decision is using a binary representation of audio. Binary representations of audio have been explored extensively in the audio fingerprinting literature. They are an excellent design decision for fingerprinting applications because they are compact in memory and thus allow for efficient storage of large databases, and they are also suitable for use with indexing techniques. The large body of works in audio fingerprinting can be clustered along several different schemas. In this subsection, we provide one such schema, and we will offer an alternative schema in the following subsection.

One way to cluster binary fingerprint representations is to divide them into what we will call threshold-based and value-based methods. A threshold-based method is one in which each bit of the representation is derived by computing some feature of interest and then applying a hard threshold. This feature of interest could be, for example, the change in subband energies [2], spectral subband moments [31], or chroma [32]. A value-based method is one in which some features of interest are computed, and the values of the features themselves are directly encoded in the binary representation. One very commonly adopted approach is to encode the location of maxima, such as the absolute or relative location of spectral peaks [33][34][35][36], the location of maxima in wavelet coefficients [4][5], or the location of local spectral luminance maxima [37].

Given the above schema, it is useful to point out that hashprints are an example of a threshold-based approach where the features of interest are delta spectro–temporal features. Also, note that these two approaches are not necessarily incompatible. The fingerprints proposed by Anguera et al. [6], for example, encode the location of a spectral peak for part of the representation, and adopt a threshold-based approach for the other bits.

B. Binary Representation: Design Method

Another way to cluster binary fingerprint representations is to separate them by their design method: manual design, supervised learning, or unsupervised learning.

The vast majority of fingerprinting works fall into the first two categories. The first category includes binary representations that are the result of manual design. These approaches often use features that have proven to be useful in other contexts, have useful mathematical properties, or carry some intuitive advantage. Some examples include methods based on spectral peaks [33][38], modulation frequency features [39], chroma [32][40], spectral flatness [41][42], spectral subband centroids and moments [43][44], and symmetric subband energy differences [2][6]. Note that value-based methods that encode the location of a maxima are almost all manually designed representations, since such quantities are generally not very amenable to mathematical optimization procedures. The second category includes approaches that incorporate some form of supervised learning into the fingerprint design process. Several works, for example, define a family of features and use boosting techniques to select the features that yield a maximally robust fingerprint [29][45][31].

There is a recent work that proposes a highly adaptive method for learning a binary fingerprint representation in an *unsupervised* manner [46]. This characteristic is very attractive in the live song identification scenario because it means that the binary representation can be tailored to each artist's music without having to collect and annotate training data for each artist. Accordingly, the hashprints described in this work adopt the same general framework as [46]. A comparison of these two works is given in the next subsection. In the vision literature, there are also works that explore unsupervised learning of binary representations using deep neural networks [47], and these will be discussed separately in a subsection below.

C. Shingling & Hashing

Another key design decision is to have each hashprint describe a relatively long temporal context containing multiple audio frames. Using multiple context frames in the manner described above is often referred to as shingling [21] or time delay embedding [13]. By considering a much higher dimensional space, this technique allows for greater discrimination on a single feature vector than could be achieved with only a single audio frame. Not surprisingly, it has proven to be very useful in handling cover song variations [21][13][14][23].

The hashprint uses hashing techniques to convert each audio shingle into a set of binary values. Given an audio shingle represented as a point in some high-dimensional space, it projects the point onto a direction of maximum variance, and then thresholds the projection to generate a bit.[7] In the hashing literature, this technique is called spectral hashing [48]. It can be thought of as a variant of locality sensitive hashing [49], where the projections are done in a data-dependent way instead of projecting onto random directions.

Hashprints can thus be thought of as applying spectral hashing to a shingle representation, along with a slight modification (computing delta features) to ensure volume-invariance. This general framework was first introduced in the context of an exact-match audio fingerprinting scenario [46]. The main differences between these works are that (1) live song identification is not an exact-match scenario, (2) we consider

[7]More precisely, the hashprint thresholds the *delta* of the projection. The main text is left as is to make the connection to spectral hashing more clear.

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

pitch-like information by using a CQT rather than a mel spectrogram, and (3) instead of using the binary representation for indexing, we use the Hamming distance between hashprints as a metric of similarity.

D. Relation to DNNs

Given the recent surge of interest in deep neural networks (DNNs), it is instructive to consider how hashprints relate to such work. In the machine learning community, many recent works have explored binary encodings learned through DNN architectures [50][51][52]. In the music information retrieval community, Raffel and Ellis [27] propose such an approach for matching MIDI and audio files. Compared to these approaches, hashprints offer two advantages in the live song identification scenario. One advantage is that it learns the binary representation in an unsupervised manner. This is particularly helpful for our scenario of interest, since collecting noisy cell phone data and annotating ground truth is very time-consuming and laborious. The second advantage is that it requires relatively little data to learn a reasonable representation. This allows hashprints to "divide up" the search space in an artist-specific way, even if the artist only has tens of tracks. In cases like these, a deep autoencoder [47][53] may not have sufficient data to converge to a good representation. So, our method straddles two extremes: it is adaptive to the data (unlike the fixed representation proposed in [25]), but it works well with small amounts of data (unlike representations based on DNNs).

Hashprints are similar to DNNs in that both are distributed representations. In fact, hashprints can be considered a single layer neural network, where each output node corresponds to a spectro–temporal filter, the weights for each output node correspond to the filter coefficients, and the output nonlinearity is a hard threshold.[8] So, in a sense, hashprints can be interpreted as a neural network that is tuned to work well with small amounts of data and forced to obey certain constraints (e.g. uncorrelated).

E. Search

The proposed search mechanism is not particularly novel or complex, but it is nonetheless useful to compare it to other approaches. Most search mechanisms generally fall into one of two groups: index-based approaches and exhaustive approaches.

Index-based approaches use binary representations to look up matches in a table or index. This lookup can be improved in a variety of ways, including doing lookups on binary codes that are close in Hamming distance [2][29], prioritizing more reliable hash values [54], using probabilistic methods like locality-sensitive hashing (LSH) and min-hash to improve retrieval performance [21][5], or accumulating lots of local fingerprint matches in an efficient data structure such as a histogram of offsets [33][6]. The advantage of index-based approaches is that they offer the ability to scale to large databases. The disadvantage is that performance drops dramatically when moving from an exact-match problem to a nonexact-match problem like live song identification.

Exhaustive approaches simply do an exhaustive search through the database. The most common example of this is to perform a full dynamic time warping (DTW) between the query and all reference sequences (e.g. [27][55]). Many works explore ways to speed up, optimize, or improve exhaustive searches. Some examples include assuming a constant global tempo difference and considering a number of tempo-adjusted queries [24], using a two-pass approach that first does a rough scoring in order to identify a set of promising candidates for a more fine-grained rescoring [32], or using techniques to deduce and skip computations that are unnecessary [56]. The advantage of exhaustive approaches is that they yield better results for nonexact-match problems. The disadvantage is the computational cost of exhaustively searching through the entire database.

The proposed search algorithm is a very simple exhaustive approach that has one very useful characteristic: we can control the tradeoff between accuracy and computational cost. Provided that the database is not too big, this approach offers the good results that come with an exhaustive search while ensuring that the runtime latency will be kept to an acceptable level.

[8]More precisely, the weights for each output node would reflect the filter coefficients for the current frame and its corresponding delta frame. The main text is again left as is for clarity of illustration.

Artist Name	ID	Genre	Songs	Dur (hrs)
Big K.R.I.T.	Big	hip hop	71	4.2
Chromeo	Chr	electro-funk	44	3.0
Death Cab for Cutie	Dea	indie rock	87	6.0
Foo Fighters	Foo	hard rock	86	6.1
Kanye West	Kan	hip hop	92	6.6
Maroon 5	Mar	pop rock	66	4.0
One Direction	One	pop boy band	60	3.4
Taylor Swift	Tay	country, pop	71	4.9
T.I.	TI	hip hop	154	10.8
Tom Petty	Tom	rock, blues rock	193	12.1

TABLE I
OVERVIEW OF THE GRACENOTE LIVE SONG IDENTIFICATION DATA. THE DATABASE CONTAINS FULL TRACKS TAKEN FROM ARTISTS' STUDIO ALBUMS. THE QUERIES CONSIST OF 1000 6-SECOND CELL PHONE RECORDINGS OF LIVE PERFORMANCES (100 QUERIES PER ARTIST).

IV. EVALUATION

We will describe the evaluation of the proposed system in three parts: the data, the evaluation metric, and the experimental results.

A. Data

We evaluated the proposed system on the Gracenote live song identification benchmark. This is a proprietary data set used for internal benchmarking purposes at Gracenote. The data comes from 10 different musical artists spanning a range of genres including pop, rock, country, and rap. Table I shows the 10 musical artists, along with a brief description and the number of studio recordings per artist. For each artist, there is a clean database and a set of noisy queries. The database consists of full audio tracks taken from the artist's released studio albums. The noisy queries are short cell phone recordings of live performances and were prepared in the following manner.

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

ID	Ref	System	Description
Ell	[12]	cover song	max cross-correlation, beat-level chroma
Hyd	[11]	cover song	system combination of [12], [57], and [58]
Sha	[33]	fingerprint	number of matching spectral peak pairs
Pan	[59]	fingerprint	number of matching spectral peak triples
Raf	[25]	live song	Hamming similarity, binarized CQT

TABLE II
SUMMARY OF THE FIVE SYSTEMS USED AS A BASELINE COMPARISON.

Ten Youtube videos of live performances were downloaded for each artist. These ten videos came from 10 different songs and were all recorded on consumer cell phones. The videos were manually pre-processed to cut out non-music material at the beginning and end of the video (e.g. applause, introducing the song, etc). Finally, ten 6-second segments evenly spaced throughout the recording were extracted. There are thus 100 queries per artist, and 1000 queries in total.[9]

B. Evaluation Metric

The evaluation metric we use is the mean reciprocal rank (MRR) [60], which is given by the following formula:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{R_i}$$

Here, N refers to the total number of queries in the benchmark, and R_i refers to the rank of the correct item for the i^{th} query. Recall that the task is a *known-artist* search, so we are only searching a single artist's database on any given query. For example, if the i^{th} query is a clip from a Tom Petty concert, then R_i can only range between 1 and 193, since there are 193 Tom Petty songs. In cases where there is more than one correct item (i.e. the artist has more than one studio recording of the same song), the best rank is used. This only applies to two songs in our entire data set, so that for almost all queries there is only one true match. Note that MRR ranges between 0 and 1, where a low value close to zero corresponds to poor performance and 1 corresponds to perfect performance. Higher MRR is better.

It is useful to point out that MRR is a more suitable metric than precision and recall for this task. Note that, when there is only one relevant item, one can deterministically draw the entire precision-recall curve given the rank of the true match for each query. In these cases, the entire precision-recall graph really only contains a single type of information (the true rank) but in a format that contains a lot of redundant, vacuous information. For this reason, the MRR is a more straightforward and appropriate metric to measure system performance on this task.

C. Results

Figure 3 shows the performance of the proposed system along with five other baseline systems. Figure 4 shows the same results broken down by artist.

[9]For more details on the data set, please contact Zafar Rafii at Gracenote (zrafii@gracenote.com).

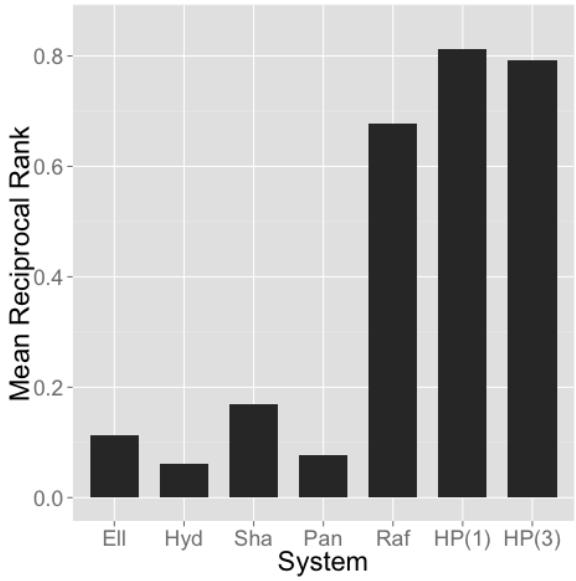


Fig. 3. Performance of proposed hashprint ('HP') system compared to five other baseline systems. The number in parenthesis indicates the downsampling factor of the cross-correlation search.

The first two baselines are cover song detection systems. Note that many cover song detection approaches use very long windows for analysis and cannot be applied directly to this problem. For example, the approaches by Bertin-Mahieux and Ellis [16] and Humphrey et al. [17] both accumulate statistics over a sliding window of 75 beat-synchronous features, corresponding to a window size of 30-45 seconds. Clearly, such an approach cannot be applied to the six-second queries in the live song identification benchmark. We selected two open-source cover song detection systems that were able to process short queries. The first baseline system ('Ell' [12]) measures similarity as the maximum cross-correlation between two sequences of beat-synchronous chroma features. The second baseline ('Hyd' [11]) is a system-level combination of three different cover song detection approaches: two variants of the 'Ell' system just described [12][57] and another that measures similarity based on dynamic programming between two sequences of beat-synchronous chroma features [58].

The next two baselines are audio fingerprinting systems. The third baseline ('Sha' [33]) is an open-source implementation [61] of the well-known Shazam algorithm. This approach identifies the location of spectral peaks in a spectrogram and encodes the relative locations of peak pairs. Similarity is computed by counting the number of matching peak pairs that are aligned in time. The fourth baseline ('Pan' [59]) is a variant of the Shazam algorithm that is invariant to pitch-shifting and tempo changes. This approach identifies the location of spectral peaks in a spectrogram based on the constant Q transform, and then encodes the relative locations of peak triples.

The fifth baseline ('Raf' [25]) is the previously proposed live song identification system at Gracenote. This system represents audio in a binary format by first computing a spectrogram based on the constant-Q transform, and then

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

binarizing the spectrogram using local adaptive thresholding. The similarity between the query and a reference track is then determined by generating a similarity matrix based on the Hamming distance between frames, and then finding the best alignment using a Hough transform. This system is the only baseline that is specifically designed for the live song identification task, so we expect it to have the best performance among the five baseline systems. Note that the two rightmost bars ('HP(1)', 'HP(3)') in figure 3 show the performance of the proposed system at downsampling rates of 1 and 3, respectively. Table II shows a brief summary of the key features from all five baseline systems.

There are four things to notice about the results in figures 3 and 4. First, the audio fingerprinting and cover song detection baselines perform poorly. The first four baselines suggest that existing cover song detection and audio fingerprinting approaches may not be suitable solutions to the live song identification problem. Audio fingerprinting approaches typically assume that the underlying source signal is identical, and may not be able to cope with the variations in live performances. (Note that the Shazam algorithm performs okay for some artists like Chromeo that use digital sounds in live performances that may be identical to the original studio recordings.) On the other hand, cover song detection systems typically assume that an entire clean studio recording is available, and may not be able to cope with short, noisy queries. Second, the proposed system substantially improves upon the previous state-of-the-art. Comparing the three rightmost systems, we see that the two versions of the proposed system improve the MRR from .68 ('Raf') to .81 ('HP(1)') and .79 ('HP(3)'). Given the reciprocal nature of the evaluation metric, this amounts to a substantial improvement in performance. Third, large computational savings can be achieved for a modest sacrifice in accuracy. Comparing the two proposed systems, we see that it is possible to reduce the search computation by a factor of $B^2 = 9$ while only reducing the MRR from .81 to .79. We will investigate the tradeoff between accuracy and search time in section V-E. Fourth, performance varies by artist. Looking at figure 4, we can see that the performance of the various systems varies a lot from artist to artist. The systems generally agree on which artists are "hard" and which are "easy." One big factor that affects these results is how similar or different an artist's live performance is compared to the original studio recording. Another big factor is how many studio tracks the artist has. Note that the artists with highest and lowest scores (Chromeo and Tom Petty, respectively) correspond to the artists with the smallest and largest databases.

V. Analysis

In this section, we investigate seven different questions of interest. These analyses are chosen to provide insight into *why* the system works well. Note that the experimental setup for all seven analyses is identical to the setup described in IV-A and IV-B, unless otherwise noted.

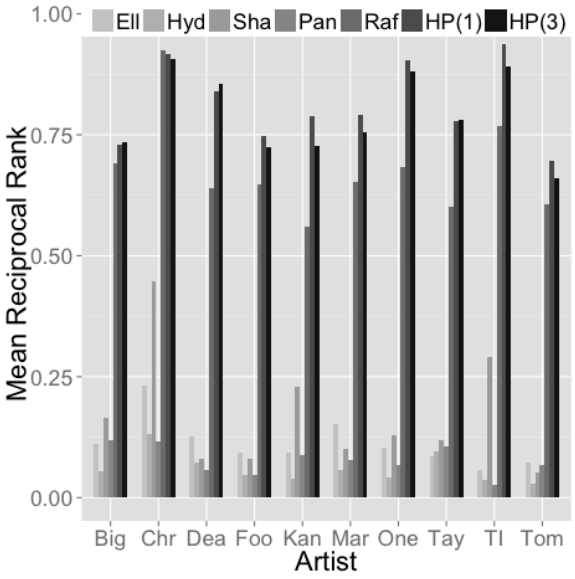


Fig. 4. Breakdown of results by artist. The first three letters of the artist's name is shown at bottom.

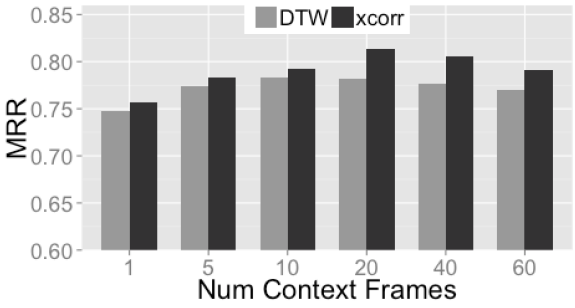


Fig. 5. Comparison of DTW and cross correlation approaches. The horizontal axis refers to the number of context frames spanned by each hashprint.

A. Effect of Tempo Mismatch

The first question of interest is, "How much do tempo mismatches affect our results?" As mentioned earlier, the cross-correlation matching approach assumes that the live performance is at approximately the same tempo as the original studio recording. Clearly, this assumption is violated if the artist performs a song faster or slower than the studio version. We would like to measure how reasonable or unreasonable this assumption is.

In order to answer this question, we compared the cross-correlation search algorithm with subsequence DTW. DTW is a common way to align two feature sequences with local tempo differences, and subsequence DTW is a variant of DTW that allows one sequence to begin at any offset in the other sequence. A brief explanation of subsequence DTW is given below, and the reader is referred to [1] (chapter 3) for a more detailed explanation. For each reference sequence in the database, we compute a Hamming distance cost matrix between the query and reference hashprint sequences, and then we determine the alignment path with lowest score using dynamic programming techniques. We use

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

the set $\{(1,1),(1,2),(2,1)\}$ as possible step sizes, which allows for tempo differences up to a factor of 2. We then use the normalized alignment path score as a match score for the reference sequence. Comparing the cross-correlation and subsequence DTW approaches will indicate how important it is to handle local tempo differences.

Figure 5 shows the comparison between the cross-correlation and subsequence DTW approaches. Since the amount of temporal context described by each hashprint can affect its ability to absorb timing mismatches, we also show this comparison across a range of context values w. There are three things to notice about Figure 5. First, cross correlation performs better than subsequence DTW. Across a wide range of context values, the cross correlation approach shows consistently better results than subsequence DTW. This indicates that the extra degrees of freedom offered by the DTW algorithm are not necessary or helpful, and may even hurt system performance. Second, more context helps up to a certain point. For both matching algorithms, using more context frames (up to 20) improves results. Third, context helps cross-correlation more than it helps subsequence DTW. Note that the MRR scores using subsequence DTW range from .75 to .78, while the MRR scores using cross correlation range from .76 to .81. With the subsequence DTW approach, tempo mismatches can be handled by the matching algorithm even if the context of each hashprint is small. But with the cross correlation approach, tempo mismatches *cannot* be handled by the matching algorithm. In this case, the only factor that can mitigate the effect of tempo mismatch is the amount of temporal context described by each hashprint, so context has a proportionally larger impact on system performance.

In summary, the tempo mismatches are small enough that across a short 6-second segment, we can approximately assume that the tempos are equal. Furthermore, increasing the amount of context in the hashprint representation allows individual hashprints to absorb slight misalignments that result from tempo mismatch. Of course, these conclusions can only generalize to the extent that these 10 musical artists are representative of other live song identification scenarios. For this data set, however, the conclusion is clear: cross correlation wins.

B. Effect of Context & Delta

The second question of interest to us is, "How do context and delta separation affect system performance?" To answer this question, we ran experiments across a range of context values w and delta separation values T. For these experiments, we used cross correlation matching with a downsampling rate of 1 (i.e. no downsampling).

Figure 6 shows the effect of context and delta separation on system performance. We see that using more context helps up to a certain point, as discussed previously. We also see that using greater delta separation (up to 992 ms) also improves results. For very small values of T, the delta spectro–temporal feature is the difference between two spectro–temporal features that are located close together in time and thus highly correlated. Because the difference between two highly correlated features is small, the variance of the resulting delta features will be small, resulting in bits that are not robust. For large values of T, the delta spectro–temporal feature will be the difference between two independent features, which effectively yields double the variance. As T increases, however, we also effectively lose T seconds of data from our 6 second query. For example, when $T = 2$ seconds, we effectively have 4 seconds of time-varying data. Thus, there is a tradeoff in selecting the optimal delta separation value T. We can see from Figure 6 that $T = 992$ ms and $w = 20$ context frames yield the best results. These settings are used in all of the reported results in this paper, unless otherwise noted.

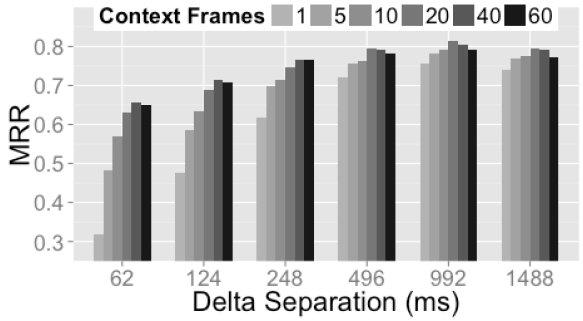


Fig. 6. Effect of hashprint context window length and delta separation value on system performance.

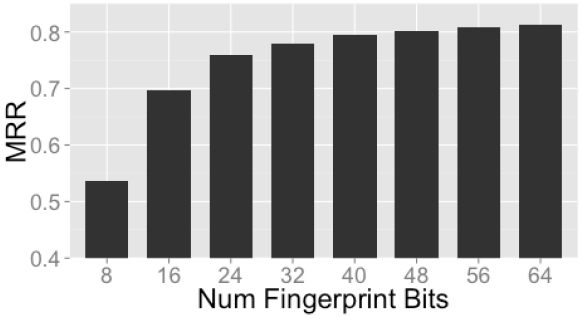


Fig. 7. Effect of the number of hashprint bits on system performance.

C. Effect of Number of Bits

The third question of interest to us is, "How does the number of hashprint bits affect the results?" Figure 7 shows the effect of varying the number of bits in the hashprint representation. For these experiments, we used a cross correlation matching with a downsampling factor of 1. As we increase the number of bits in the hashprint representation, the system performance improves in an asymptotic fashion. By the time we reach 64 bits, the system performance has largely leveled off. We would expect only marginal improvement in performance if we went beyond 64 bits. This is quite convenient for our system design, since going beyond 64 bits would require twice the storage and computation on a 64-bit machine. We use a 64-bit hashprint representation for all other results reported in this paper.

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

D. Effect of Learning

The fourth question of interest to us is, "How much does the unsupervised learning actually help?" It could be the case, for example, that the benefit in our system performance simply comes from using more context frames, rather than from learning the filter coefficients. In order to answer this question, we repeated the 'HP(1)' experiment in Figure 3 with one major change: instead of using the learned filters, we use filters with random coefficients. Using random coefficients corresponds to a locality senstive hashing (LSH) approach, which was described in section III-C. When we ran this experiment, the MRR of the system falls from .81 ('HP(1)') to .65 (LSH-based approach).

There are two things to note about this result. First, using random coefficients still performs quite well. This LSH-based approach (MRR .65) performs almost as well as the previous state-of-the-art system proposed by Rafii et al. [25] (MRR .68). So, even if the unsupervised learning was omitted, the resulting system would still perform relatively well. The effectiveness of the LSH technique has been validated by its widespread adoption in practice. Second, the learning helps a lot. The structural elements of the hashprint approach (e.g. spectro–temporal filters, amount of context, delta separation, etc) yields a performance that is on par with the previous state-of-the-art, but the unsupervised learning of filter coefficients provides substantial improvement beyond previous results. The unsupervised learning does indeed help a lot.

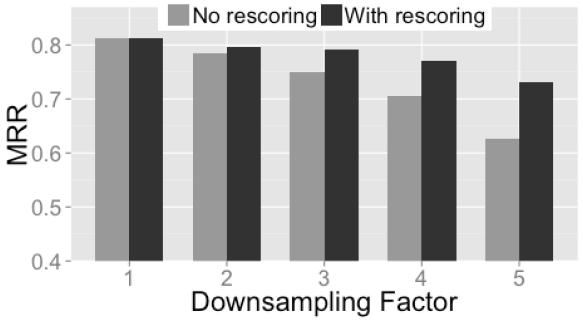


Fig. 8. Effect of downsampling and rescoring on system performance.

Downsample	Rescoring	CQT	Search	Total
1	no	.51	2.87	3.43
2	no	.50	.68	1.23
3	no	.49	.31	.85
4	no	.52	.18	.74
5	no	.49	.12	.66
1	yes	.53	2.90	3.48
2	yes	.50	.72	1.26
3	yes	.51	.35	.90
4	yes	.50	.21	.76
5	yes	.49	.15	.69

TABLE III
EFFECT OF DOWNSAMPLING AND RESCORING ON AVERAGE PROCESSING TIME PER QUERY. COLUMNS 3 AND 4 SHOW AVERAGE TIME IN SECONDS REQUIRED TO COMPUTE THE CQT AND PERFORM THE SEARCH, RESPECTIVELY. COLUMN 5 SHOWS THE AVERAGE TOTAL PROCESSING TIME PER QUERY. EXPERIMENTS WERE RUN ON A SINGLE CORE OF A 2.2 GHZ INTEL XEON PROCESSOR.

E. Effect of Downsampling & Rescoring

The fifth question of interest to us is, "How do downsampling and rescoring affect system performance?" The purpose of introducing downsampling and rescoring is to reduce the search latency, so we must consider both the accuracy and the efficiency of the system. Figure 8 shows the effect of downsampling and rescoring on system accuracy. The five groups of bars show the MRR of the system as the downsampling rate increases from 1 to 5. Each pair of bars compares the performance with and without rescoring. Table III shows the effect of downsampling and rescoring on system efficiency. To quantify the efficiency of the system, we measure the average amount of time it takes to process each 6 second query. We ran the experiments on a single core of a 2.2 GHz Intel Xeon processor. The average amount of time required to compute the CQT and search the database is shown in columns 3 and 4, and the total average runtime is shown in column 5.

There are three things to notice about the results in Figure 8 and Table III. First, downsampling provides a tradeoff between accuracy and efficiency. As we increase the downsampling rate, the MRR gets worse (Figure 8) and the average runtime gets better (Table III). We can choose a downsampling rate to achieve the desired tradeoff. Second, rescoring substantially improves MRR with low impact on runtime. Comparing the two bars in each group in Figure 8, we can see that rescoring helps to recoup a lot of the MRR that is lost as a result of downsampling. At the same time, this rescoring has very little impact on average runtime. Comparing the runtime of systems with and without rescoring in Table III, for example, we see that rescoring only adds 2 to 5 ms to the total runtime. Third, we can substantially reduce the runtime with only a modest sacrifice to MRR. With a downsampling rate of 3 with rescoring, for example, we can reduce the runtime from 3.48 seconds to 0.90 seconds while only decreasing the MRR from .81 to .79. This is the 'HP(3)' system shown at the far right of Figure 3. Note that the system proposed by Rafii et al. [25] had a (self-reported) average runtime of 10 seconds per query, so our proposed system may offer substantial savings in search computation.

One other important factor to mention is the cost of computing the CQT. We can see from Table III that the CQT is a fixed cost to the system, regardless of the downsampling rate. Downsampling can reduce the runtime of the search algorithm, but it can only asymptotically reduce the total runtime down to this fixed cost. In a real commercial system, the CQT could be computed on the query in a streaming manner, so that the latency experienced by the user effectively depends only on the search algorithm. Such an implementation, however, is beyond the scope of this work.

F. Effect of Database Size

The sixth question of interest to us is, "How does the database size affect system performance?" Since the probability of randomly selecting a true match from a collection of N possible songs is $\frac{1}{N}$, we expect the results to get worse as N increases. We would like to quantify how much the results change as a function of the artist's database size. In order to

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

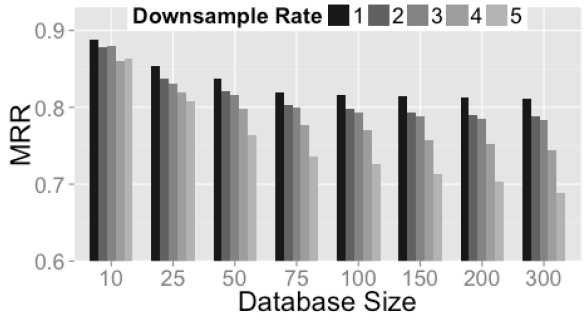


Fig. 9. Effect of database size on system performance.

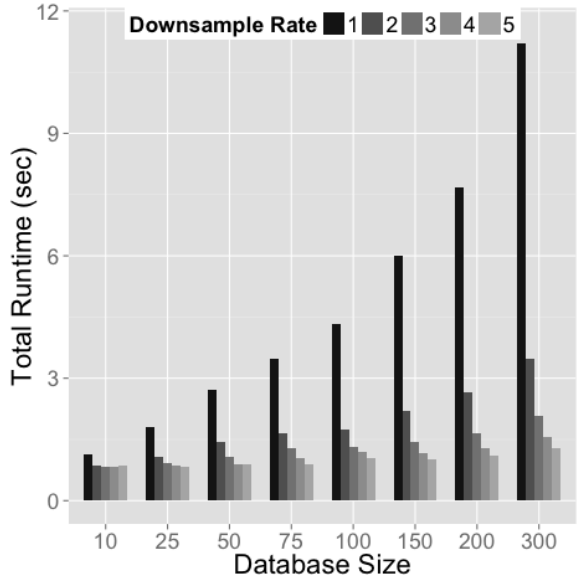


Fig. 10. Effect of database size on average processing time per query.

answer this question, we ran a set of controlled experiments in which we artificially fixed the size of the database (measured by the number of songs). We considered a range of database sizes that would be realistic for an artist of the given genres. (Note that a very established musician like Tom Petty has about 200 songs.) When the fixed database size is less than the artist's actual database size, we removed songs from the database to achieve the desired size. When the fixed database size is greater than the artist's actual database size, we padded the database with randomly selected songs from the other musical artists in order to achieve the desired size. Since the songs from other artists may result in a database containing more variation in style and genre than is typical within the artist's own repertoire, these results should be interpreted with caution. Nonetheless, such an analysis can provide a rough indicator of how database size would affect results. Figure 9 shows the MRR of the system across different database sizes and different downsampling rates. Figure 10 shows the corresponding average runtimes. Note that the database size here refers to the number of original studio recordings, so the actual number of pitch-shifted reference sequences in the database will be nine times greater.

There are three things we can observe about the results in Figures 9 and 10. First, MRR decreases asymptotically in the database size. As database size increases beyond 75, there is only a marginal decline in MRR. For example, with a downsampling factor of 1, the MRR falls from .89 to .82 as the database size increases from 10 to 75, but it only falls from .82 to .81 as the database size increases from 75 to 300. Similarly, with a downsampling factor of 3, the MRR falls from .88 to .80 as the database size increases from 10 to 75, but it only falls from .80 to .78 as the database size increases from 75 to 300. Second, the runtime increases linearly in the database size. (Note that the database sizes shown in the figures are not spaced linearly.) Since the cross-correlation search is doing a linear scan across the database, the search time will be proportional to the size of the database. The total runtime thus consists of fixed costs (such as computing the CQT) and a variable cost that is proportional to the database size. Though actual runtime will of course depend on memory usage and CPU load, a rough guideline is that there are about .6 seconds of fixed costs and each hashprint sequence takes about 4 ms to score (without downsampling). Third, the downsampling rate can be selected to stay below a maximum acceptable runtime latency. One advantage of downsampling is that we can control the tradeoff between accuracy and efficiency in an artist-specific way. For artists with a small database — where latency is not an issue — we can use a downsampling rate of 1 to maximize the reliability of the results. For artists with a large database — where latency will be an issue — we can use a higher downsampling rate to guarantee that the average latency stays below an acceptable threshold. In our experiments, for example, we can guarantee an average MRR of .75 and average runtime latency of 1.3 seconds for databases up to size 200, and we can guarantee an average MRR of .74 and average runtime latency of 1.6 seconds for databases up to size 300. The downsampling rate can thus be tailored to each artist to achieve the desired tradeoff between accuracy and efficiency.

G. Filters

The seventh question of interest to us is, "What do the learned filters look like?" This analysis can help us gain a deeper intuition about what type of information the hashprints are capturing. Figures 11 and 12 show the top 32 learned filters for two different musical artists, Big K.R.I.T. and Taylor Swift. The filters are arranged first from left to right, and then from top to bottom. The filters cover the frequency range C3 (130.81 Hz) to C8 (4186.01 Hz) and span .372 seconds.[10] There are four things to notice about the filters in these figures.

First, the filters contain both temporal and spectral modulations. Some filters primarily capture modulations along the temporal dimension, such as filters 3, 4, and 5 in figure 11 that contain vertical bands. Some filters primarily capture modulations along the spectral dimension, such as the filters on row 3 that contain many horizontal bands. Other filters capture

[10]Note that here we use a slightly larger amount of context ($w = 30$) to make the filters easier to visualize. We can see from figure 5 that using between 20 and 40 context frames yields good performance.

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

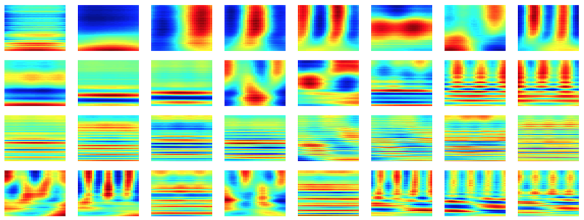


Fig. 11. Top 32 learned filters for Big KRIT. The filters are arranged first from left to right, and then from top to bottom. Each filter spans .372 seconds and covers a frequency range from C3 to C8.

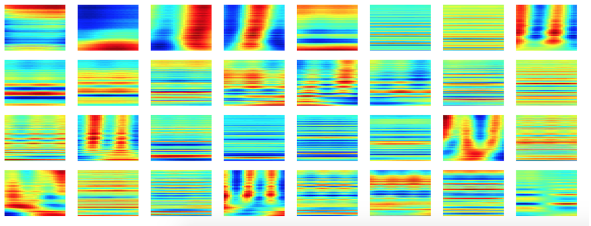


Fig. 12. Top 32 learned filters for Taylor Swift.

both temporal and spectral modulations, such as filters 15 and 16 in figure 11, which seem to capture primarily temporal modulations in the higher frequencies and primarily spectral modulations in the lower frequencies. The important thing to point out is that both types of modulations are important: if the hashprint representation were constrained to describing only a single audio frame, for example, it would be unable to characterize an important aspect of the signal.

Second, the filters capture both broad and fine spectral detail. Some filters describe the broad shape of the spectrum, such as filters 2 and 6 in figure 11. Other filters capture fine spectral details such as filters 22, 23, and 24 in the same figure. Often feature representations like chroma or MFCCs tend to focus on either fine spectral structure or broad spectral shape, but here we see that the hashprints are able to capture both types of information in a simple, unified framework.

Third, there is a progression from lower modulation frequencies to higher modulation frequencies. When we look at filters 3, 4, 5, and 8 on the first row of figure 11, we can see a progression from lower temporal modulation frequencies to higher and higher modulation frequencies. In the same figure, we see a similar progression in filters 2, 6, 9, 10, and 11 for spectral modulation frequencies. It appears that lower modulation frequencies are more useful in the sense that they yield features with greater variance.

Fourth, the filters are artist-specific. When we compare the learned filters for Big K.R.I.T. (figure 11)with the learned filters for Taylor Swift (figure 12), we see that the first four filters are quite similar. After these first four, however, the filters begin diverging and reflecting the unique characteristics of each artist's music. We notice, for example, that many more of the filters for Big K.R.I.T. emphasize temporal modulations, perhaps a reflection of the fact that rap music tends to be more percussion and rhythm-based. In contrast, the filters for Taylor Swift seem to primarily capture pitch-related information, which perhaps reflects the fact that country and pop music tends to be more harmony-based. The fact that the learned filters diverge so quickly is also an argument for using a representation that is adaptive to each artist. For a known-artist search, it may be advantageous to use a highly adaptive representation that is specific to each artist, rather than using a single unified representation based on a huge training set.

VI. Conclusion

We have introduced a known-artist live song identification system that is characterized by two main components: (1) a binary representation of audio called hashprints, which are derived from a set of spectro–temporal filters that are learned in an unsupervised, artist-specific manner, and (2) a cross correlation-based matching algorithm that can be easily tuned to achieve a desired runtime latency. On the Gracenote live song identification benchmark, the proposed system improves the mean reciprocal rank of the previous state-of-the-art from .68 to .79, while simultaneously reducing the runtime latency from 10 seconds down to 0.9 seconds. We conduct extensive analyses to understand the capabilities and limitations of the hashprint representation as well as the search mechanism. One main underlying theme of this work is approaching similarity search in a highly flexible and adaptive way: both the audio representation and matching algorithm are artist-specific, adapted to the specific characteristics of the artist's music and database size.

The proposed system has two underlying assumptions: it assumes that a database of concert event information is available, and it assumes that the query is a live performance of a song on a recorded studio album. One area of future work is to gracefully handle situations where these assumptions are violated. For example, when the query is a novel song that does not exist in the database, it would be better to switch to a different type of retrieval, such as returning a list of songs that are similar in some musically meaningful way. Another area of future work includes expanding the database to include a greater number and variety of artists, styles, and genres. Having an expanded data set would enable us to study the relationship between the specificity of learning and performance. For example, we could gain insight on this topic by comparing the performance when filters are learned per artist, per genre, and across a diverse musical database.

Acknowledgment

We would like to thank Zafar Rafii and Markus Cremer at Gracenote for generously providing the data set, and Brian Pardo for helpful discussions. Thomas Prätzlich has been supported by the German Research Foundation (DFG MU 2686/7-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

References

[1] M. Müller, *Fundamentals of Music Processing*. Springer, 2015.
[2] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2002, pp. 107–115.

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

[3] J. Haitsma, T. Kalker, and J. Oostveen, "Robust audio hashing for content identification," in *International Workshop on Content-Based Multimedia Indexing*, vol. 4, 2001, pp. 117–124.

[4] S. Baluja and M. Covell, "Audio fingerprinting: Combining computer vision & data stream processing," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, pp. 213–216.

[5] ——, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern Recognition*, vol. 41, no. 11, pp. 3467–3480, May 2008.

[6] X. Anguera, A. Garzon, and T. Adamek, "MASK: Robust local features for audio fingerprinting," in *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2012, pp. 455–460.

[7] E. Younessian, X. Anguera, T. Adamek, N. Oliver, and D. Marimon, "Telefonica research at TRECVID 2010 content-based copy detection," in *Proc. of TRECVID*, 2010.

[8] B. Coover and J. Han, "A power mask based audio fingerprint," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 1394–1398.

[9] P. Over, G. Awad, J. Fiscus, B. Antonishek, M. Michel, A. Smeaton, W. Kraaij, and G. Quénot, "TRECVID 2011 — An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics," in *Proc. of TRECVID*, 2011.

[10] J. Downie, M. Bay, A. Ehmann, and M. Jones, "Audio cover song identification: MIREX 2006-2007 results and analyses," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2008, pp. 468–474.

[11] S. Ravuri and D. Ellis, "Cover song detection: from high scores to general classification," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010, pp. 65–68.

[12] D. Ellis and G. Poliner, "Identifying 'cover songs' with chroma features and dynamic programming beat tracking," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, pp. 1429–1432.

[13] J. Serra, X. Serra, and R. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, no. 9, p. 093017, 2009.

[14] J. Serra, E. Gómez, and P. Herrera, "Audio cover song identification and similarity: background, approaches, evaluation, and beyond," in *Advances in Music Information Retrieval*, 2010, pp. 307–332.

[15] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2011, pp. 591–596.

[16] T. Bertin-Mahieux and D. Ellis, "Large-scale cover song recognition using the 2D fourier transform magnitude," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2012, pp. 241–246.

[17] E. J. Humphrey, O. Nieto, and J. P. Bello, "Data driven and discriminative projections for large-scale cover song identification," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2013, pp. 149–154.

[18] M. Khadkevich and M. Omologo, "Large-scale cover song identification using chord profiles," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2013, pp. 233–238.

[19] J. Osmalsky, J.-J. Embrechts, P. Foster, and S. Dixon, "Combining features for cover song identification," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2015, pp. 462–468.

[20] T. Bertin-Mahieux and D. P. Ellis, "Large-scale cover song recognition using hashed chroma landmarks," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2011, pp. 117–120.

[21] M. Casey, C. Rhodes, and M. Slaney, "Analysis of minimum distances in high-dimensional musical spaces," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 5, pp. 1015–1028, 2008.

[22] P. Grosche and M. Müller, "Toward characteristic audio shingles for efficient cross-version music retrieval," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 473–476.

[23] F. Kurth and M. Müller, "Efficient index-based audio matching," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 382–395, 2008.

[24] M. Müller, F. Kurth, and M. Clausen, "Audio matching via chroma-based statistical features," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2005, pp. 288–295.

[25] Z. Rafii, B. Coover, and J. Han, "An audio fingerprinting system for live version identification using image processing techniques," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 644–648.

[26] T. Tsai, T. Prätzlich, and M. Müller, "Known-artist live song ID: A hashprint approach," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2016, to appear.

[27] C. Raffel and D. Ellis, "Large-scale content-based matching of midi and audio files," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2015, pp. 234–240.

[28] C. Schörkhuber and A. Klapuri, "Constant-q transform toolbox for music processing," in *Sound and Music Computing Conference*, 2010, pp. 3–64.

[29] Y. Ke, D. Hoiem, and R. Sukthankar, "Computer vision for music identification," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 597–604.

[30] S. Haykin, *Adaptive Filter Theory*. Prentice Hall, 1996.

[31] S. Kim and C. D. Yoo, "Boosted binary audio fingerprint based on spectral subband moments," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, pp. 241–244.

[32] S. Fenet, Y. Grenier, and G. Richard, "An extended audio fingerprint method with capabilities for similar music detection," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2013, pp. 569–574.

[33] A. Wang, "An industrial-strength audio search algorithm," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2003, pp. 7–13.

[34] J. George and A. Jhunjhunwala, "Scalable and robust audio fingerprinting method tolerable to time-stretching," in *IEEE International Conference on Digital Signal Processing (DSP)*, 2015, pp. 436–440.

[35] S. Fenet, G. Richard, and Y. Grenier, "A scalable audio fingerprint method with robustness to pitch-shifting," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2011, pp. 121–126.

[36] R. Sonnleitner and G. Widmer, "Quad-based audio fingerprinting robust to time and frequency scaling," in *Proc. International Conference on Digital Audio Effects*, 2014, pp. 173–180.

[37] Y. Shi, W. Zhang, and J. Liu, "Robust audio fingerprinting based on local spectral luminance maxima scheme," in *Proc. Interspeech*, 2011, pp. 2485–2488.

[38] R. Sonnleitner and G. Widmer, "Robust quad-based audio fingerprinting," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 3, pp. 409–421, 2016.

[39] S. Sukittanon and L. E. Atlas, "Modulation frequency features for audio fingerprinting," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2002, pp. 1773–1776.

[40] M. Malekesmaeili and R. K. Ward, "A local fingerprinting approach for audio copy detection," *Signal Processing*, vol. 98, pp. 308–321, 2014.

[41] E. Allamanche, J. Herre, O. Hellmuth, B. Fröba, T. Kastner, and M. Cremer, "Content-based identification of audio material using MPEG-7 low level description," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2001.

[42] J. Herre, E. Allamanche, and O. Hellmuth, "Robust matching of audio signals using spectral flatness features," in *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Platz, New York, USA, Oct. 2001, pp. 127–130.

[43] J. S. Seo, M. Jin, S. Lee, D. Jang, S. Lee, and C. D. Yoo, "Audio fingerprinting based on normalized spectral subband centroids," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2005, pp. 213–216.

[44] ——, "Audio fingerprinting based on normalized spectral subband moments," *IEEE Signal Processing Letters*, vol. 13, no. 4, pp. 209–212, 2006.

[45] D. Jang, C. D. Yoo, S. Lee, S. Kim, and T. Kalker, "Pairwise boosted audio fingerprint," *IEEE Trans. Inf. Forensics Security*, vol. 4, no. 4, pp. 995–1004, 2009.

[46] T. Tsai and A. Stolcke, "Robust and efficient multiple alignment of unsynchronized meeting recordings," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 5, pp. 833–845, 2016.

[47] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[48] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 1753–1760.

[49] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. Annual Symposium on Computational geometry*, 2004, pp. 253–262.

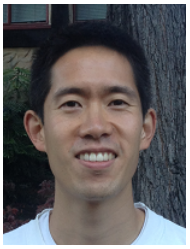
[50] M. Norouzi, D. Blei, and R. Salakhutdinov, "Hamming distance metric learning," in *Advances in neural information processing systems (NIPS)*, 2012, pp. 1061–1069.

[51] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.
The final version of record is available at http://dx.doi.org/10.1109/TMM.2017.2669864

[52] V. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2475–2483.
[53] L. Deng, M. Seltzer, D. Yu, A. Acero, A. Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *Proc. Interspeech*, 2010, pp. 1692–1695.
[54] H. Schreiber and M. Müller, "Accelerating index-based audio identification," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1654–1664, 2014.
[55] N. Hu, R. Dannenberg, and G. Tzanetakis, "Polyphonic audio matching and alignment for music retrieval," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2003.
[56] H. Nagano, R. Mukai, T. Kurozumi, and K. Kashino, "A fast audio search method based on skipping irrelevant signals by similarity upper-bound calculation," in *Proc. IEEE Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 2324–2328.
[57] D. Ellis and C. Cotton, "The 2007 labrosa cover song detection system," in *Music Information Retrieval Evaluation Exchange (MIREX)*, 2007.
[58] J. Serra and E. Gomez, "A cover song identification system based on sequences of tonal descriptors," in *Music Information Retrieval Evaluation Exchange (MIREX)*, 2007.
[59] J. Six and M. Leman, "Panako: a scalable acoustic fingerprinting system handling time-scale and pitch modification," in *Proc. International Society for Music Information Retrieval (ISMIR)*, 2014.
[60] E. Voorhees, "The TREC-8 question answering track report," in *Proc. 8th Text Retrieval Conference*, 1999, pp. 77–82.
[61] D. Ellis. (2015) Robust landmark-based audio fingerprinting. [Online]. Available: http://labrosa.ee.columbia.edu/matlab/fingerprint/



TJ Tsai completed his BS and MS in electrical engineering at Stanford University in 2006 and 2007. From 2008 to 2010, he worked at SoundHound, a startup that allows users to search for music by singing, humming, or playing a recorded track. He completed his Ph.D. at the University of California Berkeley in May 2016, and is now an assistant professor of engineering at Harvey Mudd College.

Thomas Prätzlich (S'13) received his B.Sc. (2008) in bioinformatics and his M.Sc. in computer science from Saarland University, Saarbrücken, Germany. He completed his PhD in December 2016 in the Semantic Audio Processing Group headed by Prof. Meinard Müller at the International Audio Laboratories Erlangen, which are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer-Institut für Integrierte Schaltungen IIS. He has been working as a researcher in the field of music information retrieval since 2012. His research interests are music segmentation, synchronization, and source separation.

Meinard Müller Meinard Müller studied mathematics (Diplom) and computer science (Ph.D.) at the University of Bonn, Germany. In 2002/2003, he conducted postdoctoral research in combinatorics at the Mathematical Department of Keio University, Japan. In 2007, he finished his Habilitation at Bonn University in the field of multimedia retrieval. From 2007 to 2012, he was a member of the Saarland University and the Max-Planck Institut für Informatik leading the research group *Multimedia Information Retrieval and Music Processing* within the Cluster of Excellence on *Multimodal Computing and Interaction*. Since September 2012, Meinard Müller holds a professorship for Semantic Audio Processing at the International Audio Laboratories Erlangen, which is a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Fraunhofer-Institut für Integrierte Schaltungen IIS. His recent research interests include music processing, music information retrieval, audio signal processing, multimedia retrieval, and motion processing. Meinard Müller has been a member of the IEEE Audio and Acoustic Signal Processing Technical Committee from 2010 to 2015 and a member of the Board of Directors of the International Society for Music Information Retrieval (ISMIR) since 2009. He has co-authored more than 100 peer-reviewed scientific papers, wrote a monograph titled *Information Retrieval for Music and Motion* (Springer, 2007) as well as a textbook titled *Fundamentals of Music Processing* (Springer, 2015, www.music-processing.de).

Copyright (c) 2017 IEEE. Personal use is permitted. For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.