KNOWN-ARTIST LIVE SONG ID: A HASHPRINT APPROACH

TJ Tsai[1] **Thomas Prätzlich**[2] **Meinard Müller**[2]
[1]University of California Berkeley, Berkeley, CA
[2]International Audio Laboratories Erlangen, Erlangen, Germany
tjtsai@berkeley.edu, thomas.praetzlich,meinard.mueller@audiolabs-erlangen.de

ABSTRACT

The goal of live song identification is to recognize a song based on a short, noisy cell phone recording of a live performance. We propose a system for known-artist live song identification and provide empirical evidence of its feasibility. The proposed system represents audio as a sequence of hashprints, which are binary fingerprints that are derived from applying a set of spectro-temporal filters to a spectrogram representation. The spectro-temporal filters can be learned in an unsupervised manner on a small amount of data, and can thus tailor its representation to each artist. Matching is performed using a cross-correlation approach with downsampling and rescoring. We evaluate our approach on the Gracenote live song identification benchmark data set, and compare our results to five other baseline systems. Compared to the previous state-of-the-art, the proposed system improves the mean reciprocal rank from .68 to .79, while simultaneously reducing the average runtime per query from 10 seconds down to 0.9 seconds.

1. INTRODUCTION

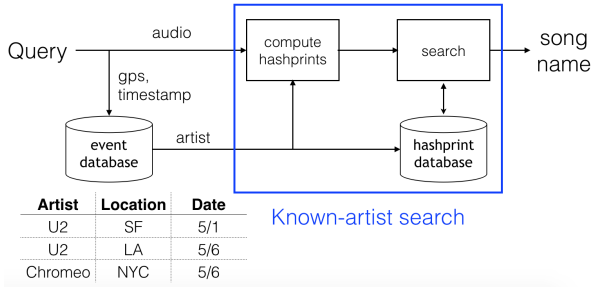
This paper tackles the problem of song identification based on short cell phone recordings of live performances. This problem is a hybrid of exact-match audio identification and cover song detection. Similar to the exact-match audio identification problem, we would like to identify a song based on a short, possibly noisy query. The query may only be a few seconds long, and might be corrupted by additive noise sources as well as convolutive noise based on the acoustics of the environment. Because song identification is a real-time application, the amount of latency that the user is willing to tolerate is very low. Similar to the cover song detection problem, we would like to identify different performances of the same song. These performances may have variations in timing, tempo, key, instrumentation, and arrangement. In this sense, the live song identification problem is doubly challenging in that it inherits the challenges and difficulties of both worlds: it is given a short, noisy query and expected to handle performance variations and to operate in (near) real-time.

To make this problem feasible, we must reduce the searchable set to a tractable size. One way to accomplish this is shown by the system architecture in Figure 1. When a query is submitted, the GPS coordinates of the cell phone and the timestamp information are used to associate the query with a concert, which enables the system to infer who the musical artist is. Once the artist has been inferred, the problem is reduced to a known-artist search: we assume the artist is known, and we would like to identify which song is being played. The known-artist search is more tractable because it constrains the set of possible songs to the musical artist's studio recordings. In this work, we will focus our attention on the known-artist search.

One important assumption in Figure 1 is that the musical artist or group is popular enough that its concert schedule (dates and locations) can be stored in a database. So, for example, this system architecture would *not* work for an amateur musician performing at a local restaurant. It *would* work for popular artists whose concert schedule is available online.

Exact-match audio identification and cover song detection have both been explored fairly extensively (e.g. [25] [1] [22] [20] [19] [7]). There are several successful commercial applications for exact-match music identification, such as Shazam and SoundHound. Both tasks have benefited from standardized evaluations like the TRECVid content-based copy detection task [13] and the MIREX cover song retrieval task [4]. There have also been a number of works on identifying related musical passages based on query fragments [8] [10] [2], but most of these works assume a fragment length that is too long for a real-time application (10 to 30 seconds). Additionally, these works mostly focus on classical music, where performed works are typically indicated on a printed program and 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 challenge for this task, as with many other tasks, is collecting a suitable data set. Rafii et al. [15] collect a set of cell phone recordings of live concerts for 10 different bands, and they propose a method for song identification based on a binarized representation of the constant Q transform. In this work, we propose an approach based on a binarized representation of audio called hashprints coupled with an efficient, flexible method for matching hashprint sequences, and we explore the performance of such an ap-



Artist	Location	Date
U2	SF	5/1
U2	LA	5/6
Chromeo	NYC	5/6

Figure 1. System architecture of the live song identification system. Using GPS and timestamp information, queries are associated with a concert in order to infer the artist.

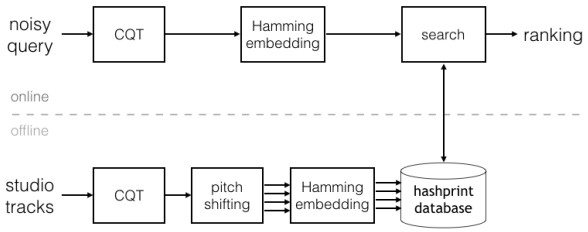


Figure 2. Block diagram for a known-artist search. Multiple pitch-shifted versions of the original studio tracks are considered to handle the possibility that the live performance is performed in a different key.

proach on the live song identification task.

This paper is organized as follows. Section 2 describes the proposed system. Section 3 describes the evaluation of the system. Section 4 presents some additional analyses of interest. Section 5 concludes the work.

2. SYSTEM DESCRIPTION

Figure 2 shows a block diagram of the proposed known-artist search system. There are four main system components, each of which is described below.

2.1 Constant Q Transform

The first main system component is computing a constant Q transform (CQT). The CQT computes a time-frequency representation of audio using a set of logarithmically spaced filters with constant Q-factor. [1] This representation is advantageous for one very important reason: the spacing and width of the filters are designed to match the pitches on the Western musical scale, so the representation is especially suitable for considering key transpositions. In our experiments, we used the CQT implementation described by Schörkhuber and Klapuri [18]. Similar to the work by Rafii et al. [15], we consider 24 subbands per octave between C3 (130.81 Hz) and C8 (4186.01 Hz). To mimic the nonlinear compression of the human auditory system, we compute the log of the subbands' local energies. At the end of this processing, we have 121 log-energy values every 12.4 ms.

2.2 Hamming Embedding

The second main system component is computing a Hamming (binary) embedding. Using a Hamming representation has two main benefits. First, it enables us to store fingerprints very efficiently in memory. In our implementation, we represent each audio frame in a 64-dimensional Hamming space, which allows us to store each hashprint in memory as a single 64-bit integer. Second, it enables us to compute Hamming distances between fingerprints very efficiently. We can compute the Hamming distance between two hashprints by performing a single logical xor operator on two 64-bit integers, and then counting the number of bits in the result. This computation offers significant savings compared to computing the Euclidean distance between two vectors of floating point numbers. These computational savings will be important in reducing the latency of the system.

Our Hamming embedding grows out of two basic principles: compactness and robustness. Compactness means that the binary representation is efficient. This means that each bit should be balanced (i.e. 0 half the time and 1 half the time) and that the bits should be uncorrelated. Note that any imbalance in a bit or any correlation among bits will result in an inefficient representation. Robustness means that each bit should be robust to noise. In the context of thresholding a random variable, robustness means maximizing the variance of the random variable's probability distribution. To see this, note that if the random variable takes a value that is close to the threshold, 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 minimize the probability of this occurring by maximizing the variance of the underlying distribution. [2]

The Hamming embedding is determined by applying a set of 64 spectro-temporal filters at each frame, and then encoding whether each spectro-temporal feature is increasing or decreasing in time. The spectro-temporal filters are learned in an unsupervised manner by solving the sequence of optimization problems described below. These filters are selected to maximize feature variance, which maximizes the robustness of the individual bits. Consider the CQT log-energy values for a single audio frame along with its context frames, resulting in a $\mathbb{R}^{121w}$ vector, where w specifies the number of context frames. We can stack a bunch of these vectors into a large $\mathbb{R}^{M\times 121w}$ matrix A, where M corresponds (approximately) to the total number of audio frames in a collection of the artist's studio tracks. Let $S \in \mathbb{R}^{121w\times 121w}$ be the covariance matrix of A, and let $x_i \in \mathbb{R}^{121w}$ denote the coefficients of the i^{th} spectro-temporal filter. Then, for $i = 1, \ldots, 64$, we solve the following sequence of optimization problems:

[1] The Q-factor refers to the ratio between the filter's center frequency and bandwidth, so a constant Q-factor means that each filter's bandwidth is proportional to its center frequency.

[2] Since the random variable is a linear combination of many CQT values, the distribution will generally be roughly bell-shaped due to the central limit theorem.

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

The objective function is simply the variance of the features resulting from filter x_i. So, this formulation maximizes the variance (i.e. robustness) while ensuring that the filters are uncorrelated (i.e. compactness). The above formulation is exactly the eigenvector problem, for which very efficient off-the-shelf solutions exist.

Each bit in the hashprint representation encodes whether the corresponding spectro-temporal feature is increasing or decreasing in time. We first compute delta spectro-temporal features at a separation of approximately one second, and then we threshold the delta features at zero. The separation of one second was determined empirically, and the threshold at zero ensures that the bits are balanced. Note that if we were to threshold the spectro-temporal features directly, our Hamming representation would not be invariant to volume changes (i.e. scaling the audio by a constant factor would change the Hamming representation). Because we threshold on delta features, each bit captures whether the corresponding spectro-temporal feature is increasing or decreasing, which is a volume-invariant quantity.

2.3 Search

The third main system component is the search mechanism: given a query hashprint sequence, find the best matching reference sequence in the database. In this work, we explore the performance of two different search strategies. These two systems will be referred to as hashprint1 and hashprint2 (abbreviated as hprint1 and hprint2 in Figure 3). In both approaches, we compute hashprints every 62 ms and using $w = 20$ context frames. These parameters were determined empirically.

The first search strategy (hashprint1) is a subsequence dynamic time warping (DTW) approach based on a Hamming distance cost matrix. The subsequence DTW is a modification of the traditional DTW approach which allows one sequence (the query) to begin anywhere in the other recording (the reference) with no penalty. One explanation of this technique can be found in [11]. We allow $\{(1,1),(1,2),(2,1)\}$ transitions, which allows live versions to differ in tempo from studio versions by a factor up to two. We perform subsequence DTW of the query with all sequences in the database, and then use the alignment score (normalized by path length) to rank the studio tracks.

The second search strategy (hashprint2) is a cross-correlation approach with downsampling and rescoring. First, the query and reference hashprint sequences are downsampled by a factor of B. For example, when $B = 2$ every other hashprint is discarded. Next, for each reference sequence in the database, we determine the frame offset that maximizes bit agreement between the downsampled query sequence and the downsampled reference sequence. The bit agreement at this offset is used as a match score for the reference sequence. After sorting all of the sequences in the database by their downsampled match score, we identify the top 10 candidate sequences. We then rescore these top 10 candidate sequences using the full hashprint sequence (i.e. without downsampling). Finally, we resort the top 10 candidate sequences based on their refined match score. The resulting ranking is the final output of the system. The advantage of the second search strategy is computational efficiency: we first do a rough scoring of the sequences, and only do a more fine-grained scoring on the top few candidate sequences.

2.4 Pitch Shifting

The fourth main system component is pitch shifting. A band might perform a live version of a song in a slightly different key than the studio version, or the live version may have tuning differences. To ensure robustness to these variations, we considered pitch shifts up to four quarter tones above and below the original studio version. So, the database contains nine hashprint sequences for each studio track. When performing a search, we use the maximum alignment score from the nine pitch-shifted versions as the aggregate score for a studio track. We then rank the studio tracks according to their aggregate scores.

2.5 Relation to Previous Work

It is instructive to interpret the above approach in light of previous work. Using multiple context frames in the manner described above is often referred to as shingling [2] or time delay embedding [20], a technique often used in music identification and cover song detection tasks. It allows for greater discrimination on a single feature vector than could be achieved based only on a single frame. The technique of thresholding on projections of maximum variance is called spectral hashing [26] in the hashing literature. It can be thought of as a variant of locality sensitive hashing [3], where the projections are done in a data-dependent way instead of projecting onto random directions. So, we can summarize our approach as applying spectral hashing to a shingle representation, along with a modification to ensure invariance to volume changes (i.e. thresholding on delta features). This approach was first proposed in an exact-match fingerprinting application using reverse-indexing techniques [23]. Here, instead of using the Hamming embedding to perform a table lookup, we instead use the Hamming distance between hashprints as a metric of similarity in a non-exact match scenario.

There are, of course, many other ways to derive a Hamming embedding. The previous work by Rafii et al. [15] performs the Hamming embedding by comparing each CQT energy value to the median value of a surrounding region in time-frequency. Many recent works have explored Hamming embeddings learned through deep neural network architectures [17] [12], including a recent work by Raffel and Ellis [14] proposing such an approach for matching MIDI and audio files. One advantage of our proposed method is that it learns the audio fingerprint rep-

Artist Name	Genre	# Tracks
Big K.R.I.T.	hip hop	71
Chromeo	electro-funk	44
Death Cab for Cutie	indie rock	87
Foo Fighters	hard rock	86
Kanye West	hip hop	92
Maroon 5	pop rock	66
One Direction	pop boy band	60
Taylor Swift	country, pop	71
T.I.	hip hop	154
Tom Petty	rock, blues rock	193

Table 1. 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).

resentation in an unsupervised manner. This is particularly helpful for our scenario of interest, since collecting noisy cell phone queries and annotating ground truth is very time-consuming and labor-intensive. Our proposed method also has the benefit of requiring relatively little data to learn a reasonable representation. This can be helpful if, for example, the artist of interest only has tens of studio tracks. In such cases, a deep auto-encoder [9] may not have sufficient training data to converge to a good representation. So, our method straddles two different extremes: it is adaptive to the data (unlike the fixed representation proposed in [15]), but it works well with small amounts of data (unlike representations based on deep neural networks).

3. EVALUATION

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

3.1 Data

We use the Gracenote live song identification data set. This is a proprietary data set that is used for internal benchmarking of live song identification systems at Gracenote. The data comes from 10 bands spanning a range of genres, including rock, pop, country, and rap. There are two parts to the data set: the database and the queries. The database consists of full tracks taken from the artists' studio albums. Table 1 shows an overview of the database, including a brief description of each band and the number of studio tracks. Note that the number of tracks per artist ranges from 44 (for newer groups like Chromeo) up to 193 (for very established musicians like Tom Petty). The queries consist of 1000 short cell phone recordings of live performances, and were generated in the following fashion. For each band, 10 live audio tracks were extracted from Youtube videos, each from a different song. The videos were all recorded from smartphones during actual live performances. For each cell phone recording, the audio was

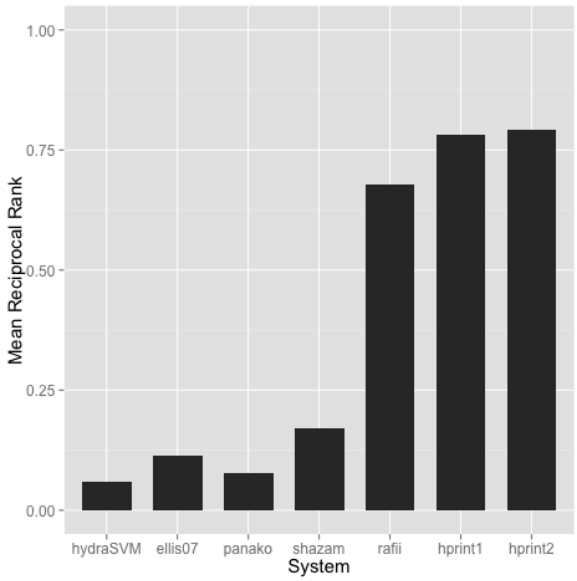


Figure 3. Mean reciprocal rank for five baseline systems and the two proposed systems (hprint1, hprint2).

cropped to exclude any non-music material at the beginning or end (e.g. applause, introducing the song, etc). Finally, ten 6-second segments evenly spaced throughout the cropped recording were extracted. Thus, there are 100 6-second queries for each band, totaling 1000 queries.

3.2 Evaluation Metric

We use mean reciprocal rank (MRR) as our evaluation metric [24]. This measure is defined by the equation

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

where N is the number of queries and R_i specifies the rank of the correct answer in the i^{th} query. When a song has two or more studio versions, we define R_i to be the best rank among the multiple studio versions. The MRR is a succinct way to measure rankings when there is an objective correct answer. Note that when a system performs perfectly — it returns the correct answer as the first item every time — it will have an MRR of 1. A system that performs very poorly will have an MRR close to 0. Higher MRR is better.

3.3 Results

Figure 3 compares the performance of the proposed hashprint1 and hashprint2 systems with five different baselines. The first two baselines (HydraSVM [16] and Ellis07 [6]) are open-source cover song detection systems. The next two baselines (Panako [21] and Shazam [25]) are open-source audio fingerprinting systems.[3] The fifth baseline is the previously proposed live song identification system by Rafii et al. [15]. In order to allow for a more fair comparison, we also ran this baseline system with four quarter tone pitch shifts above and below the original studio

[3] For the Shazam baseline, we used the implementation by Ellis [5].

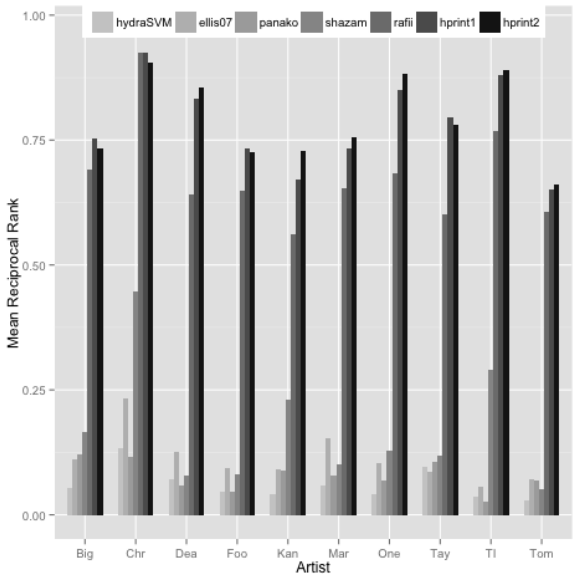


Figure 4. Breakdown of results by artist. The first three letters of the artist's name is shown at bottom.

recording. The two rightmost bars in Figure 3 show the performance of the hashprint1 and hashprint2 systems, respectively. Figure 4 shows the same results broken down by artist.

There are four things to notice in Figures 3 and 4. First, cover song and fingerprinting approaches perform poorly. The first four baseline systems suggest that existing cover song detection and existing 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 found in live performances. On the other hand, cover song detection systems typically assume that an entire clean studio track is available, and may not cope well with short, noisy queries. Second, the proposed systems improve upon the previous state-of-the-art. Comparing the three rightmost systems, we see that the two proposed systems improve the MRR from .68 (rafii) up to .78 (hashprint1) and .79 (hashprint2). Given the reciprocal nature of the evaluation metric, this amounts to a major improvement in performance. Third, the more computationally efficient version of the proposed system (hashprint2) has the best performance. In system design, we often sacrifice accuracy for efficiency. But in this case, we observe no degradation in system performance while reducing computational cost. The reason for this, as we will see in Section 4, is because the extra degrees of freedom in the DTW matching are not necessary. We will also investigate the runtime performance of these systems in the next section. Fourth, performance varies by artist. We see a wide variation in MRR from artist to artist, but all three live song identification systems generally agree on which artists are 'hard' and which are 'easy'. One major factor determining this difficulty level is how much variation there is between an artist's studio recording and live performance. The other major factor, of course, is how many studio tracks are in the database. Note that the best performance (Chromeo) and worst performance (Tom Petty) correlate with how many studio tracks the artist had.

Matching	Downsample	MRR	Runtime (s)
DTW	-	**.78**	**29.3**
xcorr	1	.81	3.43
xcorr	2	.80	1.26
xcorr	**3**	**.79**	**.90**
xcorr	4	.77	.76
xcorr	5	.73	.69

Table 2. Effect of downsampling on a cross-correlation matching approach. The third and fourth columns show system performance and average runtime required to process each 6-second query. The top row shows the performance of a DTW matching approach for comparison. The first and fourth rows correspond to the hashprint1 and hashprint2 systems shown in Figure 3.

4. ANALYSIS

In this section, we investigate two different questions of interest about the proposed systems.

4.1 Runtime

The first question of interest to us is "What is the runtime of the proposed systems?" Since live song identification is a real-time application, the amount of latency is a very important consideration. Table 2 shows the average runtime of a cross-correlation approach across a range of downsampling rates. This is the average amount of time required to process each 6-second query.[4] The runtime for a subsequence DTW approach is also shown for reference. The first and fourth rows (highlighted in bold) correspond to the hashprint1 and hashprint2 systems shown in Figure 3.

There are three things to notice about Table 2. First, cross-correlation is unilaterally better than DTW. When we compare the first two rows of Table 2, we see that switching from DTW to cross-correlation drastically reduces the runtime (from 29.3s to 3.43s) while simultaneously improving the performance (from .78MRR to .81MRR). These results are an indication that the extra degrees of freedom in the DTW matching are not beneficial or necessary. Across a short 6-second query, it appears that we can simply assume a 1-to-1 tempo correspondence and allow the context frames in each hashprint to absorb slight mismatches in timing. Of course, this conclusion only generalizes to the extent that these 10 artists are representative of other live song identification scenarios.

Second, downsampling trades off accuracy for efficiency. When we compare the bottom five rows of Table 2, we see a tradeoff between MRR and average runtime: as downsampling rate increases, we sacrifice performance for efficiency. For a downsampling rate of 3 (the hashprint2

[4] Note that the runtime scales linearly with the size of the database. So, for example, the runtime for Tom Petty will be longer than for Chromeo.

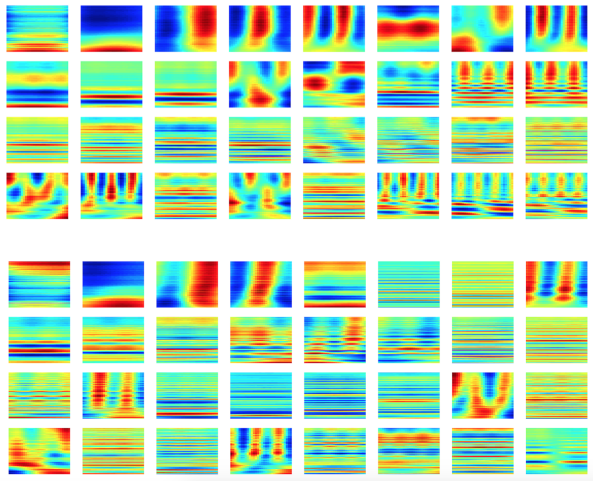


Figure 5. Learned filters for Big K.R.I.T. (top four rows) and Taylor Swift (bottom four rows). The filters are ordered first from left to right, then from top to bottom. Each filter spans .372 sec and covers a frequency range from C3 to C8.

system), we can reduce the average runtime to under a second, while only sacrificing a little on accuracy (MRR falls from .81 to .79). Note that the previously proposed system by Rafii et al. [15] has a self-reported runtime of 10 seconds per query, so the hashprint2 system may offer substantial improvement in runtime efficiency. [5]

Third, there is a floor to the runtime. Note that using a downsampling rate higher than 3 only benefits the average runtime marginally. This is because there is a fixed cost (about .5 seconds) for computing the CQT. The downsampling can only improve the time spent searching the database, but the time required to compute the query hashprints is a fixed cost. In a commercial application, however, the CQT could be computed in a streaming manner, so that the effective latency experienced by the user is determined by the search time. Such an optimization, however, is beyond the scope of this work.

4.2 Filters

The second question of interest to us is "What do the learned filters look like?" This can provide intuition about what type of information the hashprint is capturing. Figure 5 shows the top 32 learned filters for Big K.R.I.T. (top four rows) and Taylor Swift (bottom four rows). The filters are arranged first from left to right, and then from top to bottom. Each filter spans .372 sec (horizontal axis) and covers a frequency range from C3 to C8 (vertical axis).

There are three things to notice about the filters in Figure 5. First, they contain both temporal and spectral modulations. Some of the filters primarily capture modulations in time, such as filters 3, 4, 5, and 8 in the first row. Some filters primarily capture modulations in frequency, such as the filters in row 3 that contain many horizontal bands. Other filters capture modulations in both time and frequency, such as filters 15 and 16 (in row 2), which seem to capture temporal modulations in the higher frequencies and spectral modulations in the lower frequencies. The important thing to notice is that both types of modulations are important. If our hashprint representation only considered the CQT energy values for a single context frame, we would hinder the representational power of the hashprints.

Second, the filters capture both broad and fine spectral structures. Many of the filters capture pitch-like quantities based on fine spectral structure, which appear as thin horizontal bands. But other filters capture very broad spectral structure (such as filter 6, row 1) or treat broad ranges of frequencies differently (such as filters 15 and 16, previously mentioned). Whereas many other feature representations often focus on only fine spectral detail or only broad spectral structure, the hashprint seems to be capturing both types of information.

Third, the filters are artist-specific. When we compare the filters for Big K.R.I.T. and the filters for Taylor Swift, we can see that the hashprint representation adapts to the characteristics of the artist's music. The first four filters of both artists seem to be very similar, but thereafter the filters begin to reflect the unique characteristics of each artist. For example, more of the filters for Big K.R.I.T. seem to emphasize temporal modulations, perhaps an indication that rap tends to be more rhythmic and percussion-focused. In contrast, the filters for Taylor Swift seem to have more emphasis on pitch-related information, which may indicate music that is more based on harmony.

5. CONCLUSION

We have proposed a system for a known-artist live song identification task based on short, noisy cell phone recordings. Our system represents audio as a sequence of hashprints, which is a Hamming embedding based on a set of spectro-temporal filters. These spectro-temporal filters can be learned in an unsupervised manner to adapt the hashprint representation for each artist. Matching is performed using a cross-correlation approach with downsampling and rescoring. Based on experiments with 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 average runtime per query from 10 seconds down to 0.9 seconds. Future work will focus on characterizing the effect of various system parameters such as number of context frames, Hamming dimension, and database size.

6. ACKNOWLEDGMENTS

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.

[5] Since we re-implemented this baseline system without optimizing for runtime efficiency, we rely on the self-reported runtime in [15].

7. REFERENCES

[1] S. Baluja and M. Covell. Waveprint: Efficient wavelet-based audio fingerprinting. *Pattern Recognition*, 41(11):3467–3480, May 2008.

[2] M. Casey, C. Rhodes, and M. Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5):1015–1028, 2008.

[3] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational Geometry*, pages 253–262, 2004.

[4] J. Downie, M. Bay, A. Ehmann, and M. Jones. Audio cover song identification: MIREX 2006-2007 results and analyses. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, pages 468–474, 2008.

[5] D. Ellis. Robust landmark-based audio fingerprinting. Available at `http://labrosa.ee.columbia.edu/matlab/fingerprint/`, 2009.

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

[7] D. Ellis and B. Thierry. Large-scale cover song recognition using the 2d fourier transform magnitude. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, pages 241–246, 2012.

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

[9] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[10] F. Kurth and M. Müller. Efficient index-based audio matching. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):382–395, 2008.

[11] M. Müller. *Fundamentals of Music Processing*. Springer, 2015.

[12] M. Norouzi, D. Blei, and R. Salakhutdinov. Hamming distance metric learning. In *Advances in neural information processing systems*, pages 1061–1069, 2012.

[13] P. Over, G. Awad, J. Fiscus, B. Antonishek, M. Michel, A.F. Smeaton, W. Kraaij, and G. Quénot. TRECVID 2011 - An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics. In *TRECVID 2011 - TREC Video Retrieval Evaluation Online*, Gaithersburg, Maryland, USA, December 2011.

[14] C. Raffel and D. Ellis. Large-scale content-based matching of midi and audio files. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, pages 234–240, 2015.

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

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

[17] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[18] C. Schörkhuber and A. Klapuri. Constant-q transform toolbox for music processing. In *Sound and Music Computing Conference*, pages 3–64, 2010.

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

[20] J. Serra, X. Serra, and R. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9):093017, 2009.

[21] J. Six and M. Leman. Panako: a scalable acoustic fingerprinting system handling time-scale and pitch modification. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, 2014.

[22] R. Sonnleitner and G. Widmer. Quad-based audio fingerprinting robust to time and frequency scaling. In *Proceedings of the International Conference on Digital Audio Effects*, 2014.

[23] T. Tsai and A. Stolcke. Robust and efficient multiple alignment of unsynchronized meeting recordings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):833–845, 2016.

[24] E.M. Voorhees. The TREC-8 question answering track report. In *Proceedings of the 8th Text Retrieval Conference*, pages 77–82, 1999.

[25] A. Wang. An industrial-strength audio search algorithm. In *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, pages 7–13, 2003.

[26] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems 21 (NIPS'09)*, pages 1753–1760, 2009.