AUDIO FINGERPRINTING: COMBINING COMPUTER VISION & DATA STREAM PROCESSING

Shumeet Baluja & Michele Covell

Google, Inc.
1600 Amphitheatre Parkway, Mountain View, CA. 94043
{shumeet,covell}@google.com

ABSTRACT

In this paper, we present *Waveprint*, a novel system for audio identification. Waveprint uses a combination of computer-vision techniques and large-scale-data-stream processing algorithms to create compact fingerprints of audio data that can be efficiently matched. The resulting system has excellent identification capabilities for small snippets of audio that have been degraded in a variety of manners, including competing noise, poor recording quality, and cell-phone playback. We measure the tradeoffs between performance, memory usage, and computation through extensive experimentation. The system is more efficient in terms of memory usage and computation, while being more accurate, when compared with previous state of the art systems.

***Index Terms*—** Acoustic Applications, Acoustic Signal Processing, Pattern Recognition, Music

1. INTRODUCTION & BACKGROUND

Audio fingerprinting provides the ability to link short, unlabeled, snippets of audio content to corresponding data about that content. There are an immense number of applications for audio fingerprinting, ranging from recognizing music based on cell-phone playback [11], duplicate detection (for example as the popularity of music and video sharing rises [6,13]) to enhanced television [4].

Because of lossy compression of audio, as well as the numerous playback options, aurally similar sounds may have largely different encodings, making simple matching insufficient for this task. Numerous approaches have been attempted in the past for approximate matching. One of the most widely used systems [7] uses overlapping windows of audio from which to extract interesting features. In [7], overlapping windows must be used to maintain time-shift invariance for the cases in which exact time alignment is not known. 33 BFCC bands covering the 300-2000 Hz range are used for the spectral representation. Every 11.6 milliseconds, a sub-fingerprint is generated that covers a frame of 370ms. The large overlap ensures that the sub-fingerprints vary slowly over time. The sub-fingerprints are a vector of 32-bits that indicate whether the difference in successive bands increases or decreases in consecutive frames. These sub-fingerprints are largely insensitive to small changes in the audio signal since no actual difference values are kept; instead, only the signs over consecutive frames compose the sub-fingerprint. Comparisons with these fingerprints are efficient; they are simply the Hamming distance of the fingerprints.

A recent extension to the above work was presented in [9]. Based on [7], Ke introduced a learning approach into the feature selection process. An important insight provided by [9] is that the 1-D audio signal can be processed as an image when viewed in a 2-D time-frequency representation. Their learning system finds features that integrate the energy in selected frequencies over time via AdaBoost [12]. The basis of feature selection is the discriminative power of the region in differentiating between when 2 frames are the same (within a distortion set) and when they are different. Thirty-two "boxlet" features are selected, each yielding a binary value. These 32 bits are then used in an analogous procedure to the 32-bit features found by [7]. Temporal coherence is measured by a simple transition model.

An approach based on Distortion Discriminant Analysis (DDA) is explored in [2]. The features used are more complex than in the studies by [7,9], but summarize longer segments of audio. DDA is based on a variant of LDA called Oriented Principal Components Analysis (OPCA). OPCA selects a set of directions for modeling the subspace that maximizes the signal variance while minimizing the noise power. Their experiments found that the fingerprints are resistant to problems with alignment and types of noise not found in the training set.

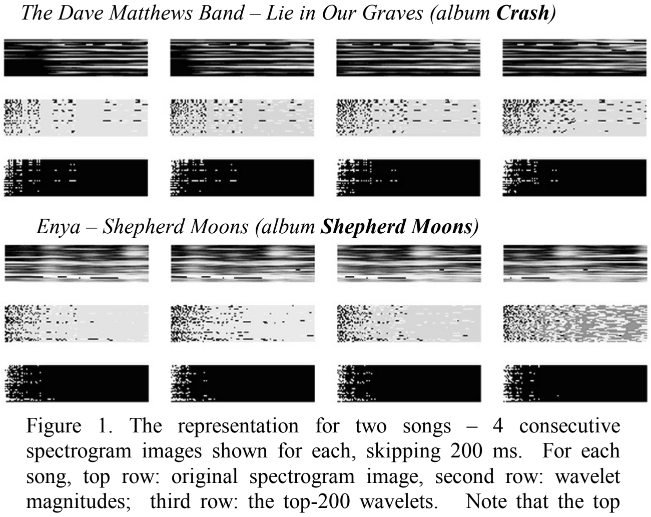
2. THE WAVEPRINT SYSTEM

Our system builds on the insights from [9]: computer vision techniques can be a powerful method for analyzing audio data. However, instead of a learning approach, we examine the applicability of a wavelet-based approach used for efficiently querying by image into large databases [8]. To make the algorithm scale, in terms of both computation and memory, we employ the hashing work from the field of large-scale data-stream processing [3,5]. The sub-fingerprints that we develop are more comprehensive than used in Haitsma's or Ke's work since they represent a longer time period; they are closer to those presented in [2].

We start our processing by converting the audio input into a spectrogram. We use slices that are 371 ms long, taken every 11.6ms, reduced to 32 logarithmically spaced frequency bins between 318 Hz and 2 kHz, as have been used in previous studies [7]. From this, we create overlapping 1.4s (128 slice) spectral images. From these spectral images, we extract the top Haar-wavelets according to their magnitude; this follows the image matching system presented in [8]. In [8], rather than comparing images directly in the pixel space, they first decomposed the image through the use of multi-resolution Haar wavelets. In our system, for each spectrogram image, a wavelet signature is computed. By itself, the wavelet-image is not resistant to noise or audio degradations. To reduce the effects of noise, while maintaining the major characteristics of the image, we select the t top wavelets (by magnitude) and discard the rest. When we look at the wavelets for successive images for two songs, we see easily identifiable patterns both in the wavelet space and even more clearly when the *top-t* wavelets are kept (see Figure 1).

For the task of image retrieval, Jacobs determined that after keeping only the top wavelets, the coefficients were not needed for effective retrieval [8]. Instead, they only kept the sign of the wavelet. As memory usage is a primary concern in this system, this same wavelet-sign representation is used in this study. The sparsity of the resulting vector makes it amenable to further reduction using a technique often used in large-scale data processing: Min Hash [3].

The Min-Hash technique works with binary vectors as follows: Select a random, but known, reordering of all the vector positions. Then, for each vector permutation, measure in which position the first '1' occurs. It is important to note for two vectors, *v1 & v2*, the probability that first_1_occurrence(*v1*) = first_1_occurrence(*v2*) is the same as the probability of finding a row that has a 1 in both *v1* and *v2,* from the set of rows that have 1 in either *v1* or *v2*. Therefore, for a given permutation, the hash values agree if the first position with a 1 is the same in both bit vectors, and they disagree if the first such position is a row where one but not both, vectors contained a 1. *Note that this is exactly what is required; it measures the similarity of the sparse bit vectors based on matching "on" positions.*


*The Dave Matthews Band – Lie in Our Graves (album **Crash**)*

*Enya – Shepherd Moons (album **Shepherd Moons**)*

Figure 1. The representation for two songs – 4 consecutive spectrogram images shown for each, skipping 200 ms. For each song, top row: original spectrogram image, second row: wavelet magnitudes; third row: the top-200 wavelets. Note that the top wavelets have a distinctive pattern for each of the songs.

We can repeat the above procedure multiple times, each time with a new position-permutation. If we repeat the process p times, with p different permutations, we get p largely independent projections of the bit vector. Together, these p values are the signature of the bit vector and replace the original, sparse, representation. The similarity of the bit vectors is computed by counting the exact matches in the p-length signatures. In our system, we store the Min-Hash computed signatures; this is the final sub-fingerprint of the audio-image, and is computed on each spectral image.

Each spectral image is represented by a series of p 8-bit integers, the sub-fingerprint: We empirically measured that 8-bits were sufficient. In the majority of the cases tried (described in the next section), the probability of the first-1-occurrence being after position 255 was insignificant. Even with this compression, efficiently finding near-neighbors in a $p>50$ dimensional space is not a trivial task; naive comparisons are not practical. Instead, we use a technique, termed Locality-Sensitive Hashing (LSH) [5]. It is efficient in the number of comparisons that are required and provides noise-robustness properties.

In contrast to standard hashing, LSH performs a series of hashes, each of which examines only a portion of the sub-fingerprint. The goal is to partition the feature vectors (in this case the Min-Hash signatures) into l subvectors and to hash each subvector into l separate hash tables. Each hash table uses only a single subvector as input to the hash function. Candidate neighbors can be efficiently retrieved by partitioning the probe feature vector and collecting the set of entries in the corresponding hash bins. The final list of potential neighbors can be created by vote counting, with each hash casting votes for the entries of its indexed bin, and retaining the candidates that receive some minimum number of votes, v. If $v = 1$, this takes the union of the candidate lists. At the other extreme, if $v= l$, this takes the intersection (and is equivalent to standard hashing).

2.1. Retrieval

The overall retrieval process is shown in Figure 2. Rather than dividing the song in uniformly overlapping segments, as was done in the fingerprint generation process, the song is divided into randomly overlapping segments. Randomizing the stride amount avoids problems of unlucky alignments. If the sampling of the probe had been kept constant, it might

```
1. Create spectral images of 11.6*$w$ duration from the
   spectrogram, with random spacing averaging $d$-ms apart.

   For each spectral image:
      2. Compute the wavelets on the spectral image
      3. Extract the top-t wavelets.
      4.Create a binary vector of the top-t wavelets.
      5.Use min-hash to create a sub-fingerprint of the top-t
         wavelets (note that the same permutations used in the
         database-creation portion are used here)
      6. Using LSH, with $b$ bins, $l$ hash tables, find the sub-
         fingerprint segments that are close matches.
      7. Discard sub-fingerprints with less than $v$ matches
      8. Compute the Hamming distance from the remaining
         candidate sub-fingerprints to the query sub-fingerprint.

9. Use Dynamic Prog. to combine the matches across time
```

Figure 2: Retrieval process.

repeatedly sample uniformly large offsets from the sample positions that were used to create the database. After each audio snippet is created, its signature is computed in the same manner as described in the database generation process (Steps 2-5). The next steps 6-8 describe an efficient mechanism for finding matches in the database and measuring their distances from the query. When using LSH (steps 6-7), we require that the sub-fingerprints have at least v votes to be compared with the query sub-fingerprint. For the sub-fingerprints that have at least the minimum number of votes, we then compare them fully: because each byte of the sub-fingerprint is a Min-Hash signature, we simply look at the number of bytes (out of p) that match exactly. The sub-fingerprint with the maximum score is the best match on that spectral image.

The simplest method to combine evidence across all of the probe's sub-fingerprints is voting without temporal constraints. We also explored using dynamic-time warping [10] to impose "tempo" constraints in mapping the probe sequence onto the database songs. We use both global-slope constraints (there can be on 10% change in tempo from the probe to the candidate match) and local-slope constraints (no single probe can match more than one song location within a single track and no local-time inversions are allowed).

3. ACCURACY, MEMORY & COMPUTATION

One of the intrinsic difficulties in designing a large-scale system is conducting a thorough exploration of the parameter settings. In this section, we report the outcome of extensive testing of parameter sets. *Over 50,400 different parameter combinations were tried to ensure that we select the best settings and quantify the tradeoffs with each parameter.* The parameter combinations included varying all of the parameters listed, as well as using dynamic programming for temporal coherence measurement. The full details of this exploration can be found in [1]. The results from those experiments varied greatly along every axis: accuracy, memory and computation, depending on the settings. Only 122 of the 50,400 settings achieved a retrieval accuracy of >97.5% on a probe set of 1000 samples. The computation and memory requirements within the 122 settings varied drastically also, with a range that spanned 2-3 orders of magnitude in computation and memory. From the full set of experiments, we selected 2 operating points:

Waveprint 1: 97.9% accuracy; settings: t=200 top wavelets retained, l=20 hashes, 0.9-sec DB stride, d=46-ms probe stride, v= 2

Waveprint 2: 97.5% accuracy; settings: t=200 top wavelets retained, l=25 hashes, 0.9-sec DB stride, d=46-ms probe stride, v=5

With these two operating points, we perform a comprehensive set of tests with a large number of signal degradations, and compare our system to that of [9][1]. The results are shown in Table I. For each task, we lookup randomly selected, and degraded, samples from a database of 10,000 songs. We evaluate recognition accuracy with 10-, 30-, and 60-second snippets. The accuracy is the percentage of times the correct song was found as the top match, measured over 1,000 trials for each of the 33 tasks (11 degradations * 3 probe lengths) The following signal degradations are evaluated on the probe set:

- Time Offset Only: no signal degradation, only time-offset is unknown.
- Echo: 90% of the original signal level and arrives 100ms after original.
- Equalizer: passes the signal through an equalizer with the settings of [7].
- MPEG2 layer-3 at 32 Kbps CBR: encodes and decodes the test probes.
- GSM-Adaptive Multi-Rate AMR at 4.75 Kbps CBR: encodes and decodes to 4.75-kbps-mode AMR audio (cell-phone emulation).
- Loud structured noise: adds Enya's *Watermark I* or To Die For's *Veil of Tears Epilogue* to the probe at a fixed volume. Perceptually, this ranged from ½–2× the volume of the probe content recording.
- Linear Speed Modification (LSM)[2]: changed the playback speed ± 2%.
- Time-Scale Modification (TSM): tempo ± 10% w/o changing the pitch.

The results show uniformly improved performance over the compared system. Further, when we look at the computation and memory requirements of *Waveprint*, we again find a benefit. We can express memory usage in terms of the parameters of the system. For N songs stored in our database, with an average length of M sec, memory usage is on the order of $\boldsymbol{8*l*b + 4*l*N*M/s + p*N*M/s}$. The first term is the b-bin hash tables (b=100,000 for Waveprint 1 & Waveprint 2), using a simple array based representation. l is the number of hash tables used. In each bin, the following is stored: one pointer to its content array and one counter of the number of elements in the content array. The second term represents the terms that are pointed to in the hash-

[1] We used the publicly available code base for Ke's system from http://www.cs.cmu.edu/~yke/musicretrieval/. We modified their amplitude normalization, by using a smoothly varying normalization computed on a sliding window of the surrounding 5 seconds. This *uniformly* improved their results across all experiments.

[2] Since the extension of Ke's temporal model presented in [9] to include timing variations should be possible, we omit the pessimistic performance numbers of their system for time-modification tests (LSM & TSM).

Table I: Results

Degradation Source	Recognition System	10 Sec Query	30 Sec Query	60 Sec Query
Time-Offset Only	Waveprint 1	***100.00***	***100.00***	***100.00***
	Waveprint 2	***100.00***	***100.00***	***100.00***
	Ke	99.60	99.61	99.60
Echo-90%	Waveprint 1	***99.90***	***99.59***	***99.68***
	Waveprint 2	97.98	***99.59***	***99.68***
	Ke	99.00	99.11	99.25
Equalizer	Waveprint 1	***99.80***	99.69	***99.79***
	Waveprint 2	98.18	***100.00***	***99.79***
	Ke	99.13	99.14	99.30
MP3-32K	Waveprint 1	***97.47***	***98.76***	***99.15***
	Waveprint 2	97.37	98.44	98.50
	Ke	94.34	93.55	93.17
Noise-Enya	Waveprint 1	***86.97***	95.96	98.40
	Waveprint 2	85.45	***96.17***	***98.51***
	Ke	60.49	71.72	75.86
Noise-Veil of Tears	Waveprint 1	***82.29***	***84.37***	85.18
	Waveprint 2	81.01	84.16	***85.25***
	Ke	73.09	75.61	77.99
AMR: GSM Codec	Waveprint 1	***95.86***	***99.28***	***99.79***
	Waveprint 2	86.16	96.69	99.04
	Ke	94.87	96.97	97.65
LSM-102	Waveprint 1	***80.30***	***92.32***	***96.15***
	Waveprint 2	60.30	80.81	90.17
LSM-98	Waveprint 1	***93.74***	***95.56***	***96.95***
	Waveprint 2	90.81	94.21	94.78
TSM-90	Waveprint 1	***99.40***	***99.69***	***99.79***
	Waveprint 2	90.81	94.21	94.78
TSM-110	Waveprint 1	***99.09***	99.69	***99.78***
	Waveprint 2	98.89	***100.00***	99.68

tables (represented in the first term). This is the number of hash-table pointers to the sub-fingerprints. *(M / s)* is the average number of sub-fingerprints for each song. Each hash table has a pointer to all of the sub-fingerprints. The final term is the actual sub-fingerprint storage: for each of the *(N * M / s)* spectral images in the database, there are *p* 1-byte elements (where *p* is the number of Min-Hash permutations, *p*=100 for both Waveprint 1 & Waveprint 2). To make this concrete, we can store 10,000 songs in approximately 0.42-0.46×10^9 bytes of memory (*Waveprint-1 & 2* respectively). Approximately 45,000-50,000 songs can be placed on a 2 GB machine without touching disk for retrieval. In contrast, using the approach reported in [7][3], we estimate that the previous best approach would use 0.7×10^9 bytes for a 10,000 3.5-minute song database, allowing 31,000 songs in 2 GB of memory.

A similar analysis for the computational load shows that *Waveprint-1* and *Waveprint-2* both have average case performances that are either lower than or within the bottom 1% of the range of performances expected in [9], in terms of the expected number of byte-comparisons. In terms of real-time performance, on a 3.4 Ghz Pentium, Waveprint is approximately 13.6× faster than the probe duration *(e.g.* a 10 second probe takes approximately 0.73 seconds to find in our database). Effectively, if the goal is to retrieve a 3.5 minute song, when we use a 10-sec probe, our system is 286× faster than real time (since the full song is not examined); for a 30-sec probe, 94× faster; for a 60-sec probe, 47× faster. These numbers reflect unoptimized performance; it is possible to speed up the most computationally expensive portion of the process (computing and sorting the wavelets, which account for 90% of the computation) by a factor of ~16-32x. This can be done by reusing partial results across successive sub-fingerprints, since much of the computation is repeated across the time-window being examined.

4. CONCLUSIONS

In this work, we have presented the *Waveprint* audio identification system. The system builds on the insight of [9]: the task of audio recognition can be effectively addressed through computer-vision techniques. In this work, we extended the computer-vision work presented in [8] for retrieving near-duplicate images from a large corpus of image data to the task of audio retrieval. Efficiencies were obtained through using data-stream processing techniques such as Min Hash. For the size databases explored, the resulting system is more efficient in terms of memory usage than the state-of-the-art competing system while providing better recognition accuracy. Immediate next steps include scaling the database. We have seen preliminary promising results, both in terms of accuracy and speed, especially in the *Waveprint-2* setting. Other future work includes exploring applications beyond music matching, such as using the system for matching television broadcasts.

5. REFERENCES

[1] Baluja, Covell. Content fingerprinting using wavelets, *Proc. CVMP* (2006).

[2] Burges, Platt, Jana. Distortion Discriminant Analysis for Audio Fingerprinting, *IEEE-PAMI* 11(3):165–174, 2003.

[3] Cohen, *et al*. Finding interesting associations without support pruning. *Knowledge and Data Engineering*, 13(1):64–78, 2001.

[4] Fink, Covell, Baluja. Social- and Interactive-TV Application Based on Ambient-Audio Id, *Proc. Euro-ITV,* 2006.

[5] Gionis, Indyk, Motwani. Similarity search in high dimensions via hashing. *Proc. VLDB*, pp. 518–529, 1999.

[6]Google Video http://video.google.com/video_about.html, 2005.

[7] Haitsma, Kalker. A Highly Robust Audio Fingerprinting System. *Proc. ISMIR,* 2002.

[8] Jacobs, Finkelstein, Salesin. Fast Multiresolution Image Querying. *Proc SIGGRAPH,* 1995.

[9] Ke, Hoiem, Sukthankar. Computer Vision for Music Identification. *Proc. CVPR*, 2005.

[10] Myers, Rabiner. A comparative study of several dynamic time-warping algorithms for connected word recognition. *The Bell System Technical Journal*, 60(7):1389-1409, 1981.

[11]Shazam Ltd. http://www.shazam.com/, 2006.

[12] Viola, Jones.Robust Real-time Object Detection. *ICCV,* 2001.

[13] YouTube, http://www.youtube.com/t/fact_sheet, 2005.

[3] Since [7] gave better bounds for memory/computation than [9], we used [7] for this comparison.