Computer Vision for Music Identification

Yan Ke[1], Derek Hoiem[1], Rahul Sukthankar[1,2]
[1]School of Computer Science, Carnegie Mellon; [2]Intel Research Pittsburgh
{yke,dhoiem,rahuls}@cs.cmu.edu
http://www.cs.cmu.edu/~yke/musicretrieval/

Abstract

We describe how certain tasks in the audio domain can be effectively addressed using computer vision approaches. This paper focuses on the problem of music identification, where the goal is to reliably identify a song given a few seconds of noisy audio. Our approach treats the spectrogram of each music clip as a 2-D image and transforms music identification into a corrupted sub-image retrieval problem. By employing pairwise boosting on a large set of Viola-Jones features, our system learns compact, discriminative, local descriptors that are amenable to efficient indexing. During the query phase, we retrieve the set of song snippets that locally match the noisy sample and employ geometric verification in conjunction with an EM-based "occlusion" model to identify the song that is most consistent with the observed signal. We have implemented our algorithm in a practical system that can quickly and accurately recognize music from short audio samples in the presence of distortions such as poor recording quality and significant ambient noise. Our experiments demonstrate that this approach significantly outperforms the current state-of-the-art in content-based music identification.

1. Introduction

At first glance, problems in the audio domain may appear to have little relevance to computer vision. The former deals with processing 1-D signals over time while computer vision tends to focus on the interpretation of one or more 2-D images (typically captured from a 3-D scene). However, we believe that certain problems in the audio domain transform very naturally into a form that can be effectively tackled by computer vision techniques. This belief is motivated by the observation that audio researchers commonly employ 2-D time-frequency representations, such as spectrograms, when analyzing sound or speech. Traditionally, these representations are treated as images only for visualization purposes. Our approach is to apply current computer vision techniques, such as boosted classifiers [17] and local-descriptor based object recognition [11], to these "images". This paper evaluates the merits of this approach in the context of a real-world audio application: music identification.

The goal of music identification is to reliably recognize a song from a small sample of noisy audio. For instance, a user may wish to identify the music playing on her car radio or in the background at a party. She could send a few seconds of the audio using her mobile phone to a music identification server and receive a text message with the title of the song. This problem is challenging for several reasons. First, the query can be significantly corrupted by the distortions induced by typical portable recording devices or due to noise from ambient sounds. Second, the audio sample from the query will typically match only a small portion of the target song, such that a traditional digital signature computed over the query is unlikely to match the signature of the entire song. Third, a practical music identification system should scale, both in accuracy and speed, to databases containing hundreds of thousands of songs. Recently, the music identification problem has attracted considerable attention, both from commercial companies [1–3] and researchers [4, 8]. However, the task remains challenging, particularly for noisy real-world queries.

We cast music identification into an equivalent sub-image retrieval framework: identify the portion of a spectrogram image from the database that best matches a given query snippet. We first transform the audio data into a spectrogram image and compute local descriptors over the spectrogram for overlapping time intervals. Using these descriptors, we then efficiently retrieve candidate matches for the query. For each candidate, we use RANSAC [5] to temporally align the candidate to the query and compute the likelihood that the aligned candidate matches the query. In Section 2, we describe a broad set of filters suitable for our task. By applying a novel pairwise boosting algorithm, we select a compact subset of these filters, as described in Section 3. Section 3, also describes our model for computing the likelihood of a candidate match and our use of the EM algorithm to accommodate noise and interference. We discuss our retrieval process in Section 4 and provide implementation details in Section 5. Section 6 describes our experiments and compares our method to recent work in music identification. Section 7 summarizes our contributions and outlines directions for future work.

2. Representing Audio as Images

Given a short segment of distorted audio data, we would like the music identification system to quickly find the matching segment of undistorted audio in a large database. The system should meet the following performance requirements: high recall, high precision, query using short audio clips, and fast retrieval. To achieve high recall, its representation must be sufficiently descriptive to distinguish between similar-sounding songs. High recall, on the other hand, demands that the representation also be highly resistant to distortions caused by background noise or poor recording quality. For instance, a song played over low-quality speakers and recorded using a laptop's built-in microphone will sound significantly different from the same song played over high-fidelity speakers and recorded using a professional microphone. Since we want the ability to identify a song based on only a few seconds of audio sampled at an arbitrary point in the song, the representation should be local and robust to small shifts in time. Furthermore, a music identification system should scale to large music databases, returning accurate responses in a few seconds on queries against hundreds of thousands of songs. This scaling requirement indicates that the representation should be computationally inexpensive and efficiently indexable.

Creating a feature representation that meets all of these criteria is a challenging task. The original 1-D audio signal varies widely with small distortions, and perceptual information is difficult to extract directly (see Figure 1a). Our approach is to convert the audio signal into a 2-D time-frequency image (spectrogram), using the short-term Fourier transform (STFT) [8]. This spectrogram represents the power contained in 33 logarithmically-spaced frequency bands, measured over 0.372 s windows in 11.6 ms increments. In the spectrogram image, corrupted audio bears some visual similarity to its original, and the signal differences due to different audio sources are more apparent (see Figure 1b). This motivates our assertion that well-developed computer vision techniques should enable us to build good descriptors for our task.

Although the process of converting the time-domain signal into a spectrogram image illuminates important similarities and differences in the audio, simply comparing spectrograms using correlation would be inaccurate and slow. Instead, we advocate learning a small set of filters whose responses are robust to expected distortions while preserving the information needed to distinguish between different songs. Rather than attempting to manually engineer such a suitable set of filters, we define a broad class of candidate filters and apply machine learning techniques to identify a small subset that performs well together. To determine an appropriate family of filters for this task, it is helpful to examine the characteristics of spectrogram images that are distinctive (sensitive to the particular song) while being resistant to expected distortions. These characteristics include: (a) differences of power in neighboring frequency bands at a particular time; (b) differences of power across time within a particular frequency band; (c) shifts in dominant frequency over time; (d) peaks of power across frequencies at a particular time; and (e) peaks of power across time within a particular frequency band. The proposed filter family should be able to capture these aspects while operating along different frequency bands with different bandwidths and over different extents of time. Filters with large bandwidths and time-widths are more robust to certain distortions, but filters with short bandwidths and time-widths can capture discriminative information that the former filters cannot. If we view the spectrogram as a simple 2-D grayscale image, we can see that the class of Haar wavelet-like filters introduced by Viola and Jones for face detection [17] meets these requirements (see Figure 2).

In our system, each filter type can vary in band location from 1 to 33, in bandwidth from 1 to 33, and in time from 1 frame (11.6 ms) to 82 frames (951 ms) in exponential steps of 1.5, resulting in a set of roughly 25,000 candidate filters. From this large candidate set, we select M discriminative filters and corresponding thresholds to generate an M-bit vector that represents overlapping segments of audio (see Figure 1c). This vector, called the descriptor, can be quickly computed using integral images [17] and is suffi-

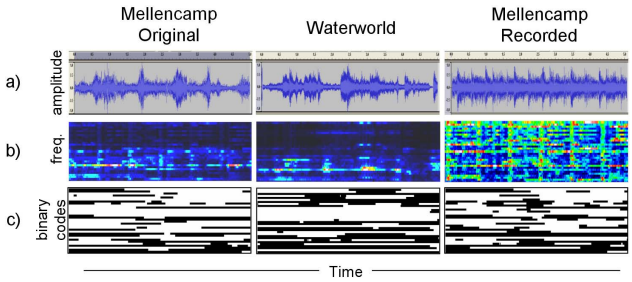

Figure 1: Representing audio. Three 10-second snippets of audio are shown: John Mellencamp original, Waterworld soundtrack, and John Mellencamp recorded. It is difficult to determine which snippet matches the song in the waveform audio representation (a). In the spectrogram images (b), certain similarities between the two Mellencamp snippets and distinguishing differences between the Mellencamp and Waterworld snippets become noticeable. Matching snippets are easily identified using our learned descriptions (c).

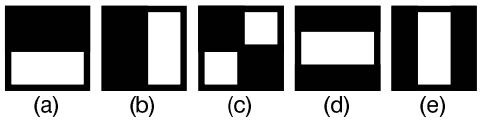

Figure 2: Candidate filter set. We select a compact set of filters from the filter class designed for object detection by Viola and Jones. When applied to spectrogram images, these filters capture important time-frequency characteristics of audio.

ciently stable across distortions to enable retrieval by direct hashing in the database, as described in Section 4. We describe how to learn the description in the next section.

Of course, a single descriptor cannot contain enough information to accurately identify the song matching the given query from among hundreds of thousands of full-length songs. To represent several seconds of an audio snippet, we compute descriptors for overlapping windows of audio every 11.6 ms. Thus, for a ten-second snippet of audio, our signature consists of 860 descriptors. This signature is the basis for matching and retrieval, as described in the following sections.

3. Filter Selection and Modeling

The previous section described how we can treat the time-frequency representation of an audio signal as an image and outlined the set of candidate filters that operate on the spectrogram image. This section details our method for selecting a subset of those filters (and corresponding thresholds) to create a compact representation for each local region of the spectrogram image. The goal is to build a representation in which an original audio segment and its distorted versions will generate highly-similar descriptors, while audio segments from two different songs will generate dissimilar descriptors. Section 3.1 details how this description can be learned using an algorithm that we call *pairwise boosting*.

The descriptors capture only the *local* similarity between a pair of short segments of audio. To correctly evaluate the match between the query snippet and a song in the database, we need to compute the probability that an entire signature (the series of descriptors computed on overlapping audio windows) matches the other.

Additionally, we account for "occlusion" due to background noise that drowns out the signal or due to a poor mobile phone connection. We assume that each descriptor in the signature was generated either by the original song or by an occluding signal. Section 3.2 describes how we employ the Expectation Maximization (EM) algorithm [13] and a simple dependency model to automatically determine whether a given descriptor in a sequence corresponds to the song or an occlusion and to compute the likelihood that one signature matches another.

3.1. Learning Compact Audio Descriptions

Our goal is to build a description that enables us to determine the probability that two (potentially distorted) audio snippets were both sampled from the same position of the same song. Formally, this entails learning a classifier $H(x_1, x_2) \rightarrow y\{-1, 1\}$, where x_1 and x_2 are two spectrogram images and the label y denotes whether the images derive from the same original audio source (y=1) or not (y=-1). One popular method of building a description for object recognition is to define a large class of filters and use Adaboost [6, 15] to select a small subset of those filters for classification. We apply a novel pairwise variant of this method. Our classifier is an ensemble of M weak classifiers, $h_m(x_1, x_2)$, each with an associated confidence, c_m. Our weak classifiers are composed of a filter f_m and a threshold t_m, such that $h_m(x_1, x_2)$=$sgn[(f_m(x_1) - t_m)(f_m(x_2) - t_m)]$. In other words, if two examples generate filter response values on the same side of the threshold, they are labeled by the weak classifier as deriving from the same portion of audio; otherwise, they are labeled as coming from different audio snippets. Note that this formulation differs from standard Adaboost in that labels are assigned to *pairs* of filter responses. Once the weak classifiers are learned, any spectrogram x can be transformed into an M-bit vector, allowing fast indexing through hashing techniques (see Section 4).

One way to learn these weak classifiers would be through the standard Adaboost framework in which, iteratively, weak classifiers are learned and all of the data is reweighted. Such an approach, however, would produce poor results in this case for the following reason: no weak classifier can perform better than chance, on average, on the *non-matching* example pairs! This may seem an odd assertion, but the proof is summarized as follows. Suppose we have x randomly drawn from distribution D, a filter f_m, and a threshold t_m, such that $P(f_m(x) < t_m) = p$, with $0 \leq p \leq 1$. If we independently and randomly draw two non-matching examples x_1 and x_2 from D, then the probability that x_1 and x_2 fall on different sides of t_m is given by

$$P\left(h_m(x_1, x_2) = -1\right) = 2p(1-p) \leq 0.5. \qquad (1)$$

Thus, a pair of non-matching (y=-1) examples will incorrectly be classified as matching at least half of the time for a sufficiently large sample size, violating the weak classifier condition of Adaboost. We resolve this issue by employing an asymmetric pairwise boosting algorithm, in which only the matching (y=1) pairs are re-weighted and the weights of matching pairs and non-matching pairs are normalized such that the sum of each is equal to one-half. Our algorithm is detailed in Figure 3.

From Equation 1, we also note that we can explicitly calculate the probability of error for non-matching pairs for a particular filter and threshold if we know the distribution of the filter responses. We observe that this distribution can be estimated from the single members of the matching pairs, providing two results: (1) the median is the optimal threshold for non-matching pairs; and (2) when the filters are loosely correlated, we do not need non-matching pairs — providing a two-fold speed-up in training or the ability to employ a larger training set of matched pairs at no additional computational cost. Experiments reveal that

Pairwise Boosting

input: sequence of n examples $\langle(x_{11}, x_{21})\rangle..\langle(x_{1n}, x_{2n})\rangle$, each with label $y_i \in \{-1, 1\}$

initialize: $w_i = \frac{1}{n}, i = 1..n$

for m = 1..M

1. find the hypothesis $h_m(x_1, x_2)$ that minimizes weighted error over distribution w, where $h_m(x_1, x_2) = sgn[(f_m(x_1) - t_m)(f_m(x_2) - t_m)]$ for filter f_m and threshold t_m
2. calculate weighted error: $err_m = \sum_{i=1}^{n} w_i \cdot \delta(h_m(x_{1i}, x_{2i}) \neq y_i)$
3. assign confidence to h_m: $c_m = \log(\frac{1-err_m}{err_m})$
4. update weights for matching pairs: if $y_i = 1$ and $h_m(x_{1i}, x_{2i}) \neq y_i$, then $w_i \leftarrow w_i \cdot exp[c_m]$
5. normalize weights such that $\sum_{i:y_i=-1}^{n} w_i = \sum_{i:y_i=1}^{n} w_i = \frac{1}{2}$.

final hypothesis:

$H(x_1, x_2) = sgn(\sum_{m=1}^{M} c_m h_m(x_1, x_2))$

Figure 3: Summary of our pairwise boosting algorithm for learning a hypothesis that can determine whether the members of a pair, x_1 and x_2, belong to the same class (match) or belong to different classes. Note that the algorithm is asymmetric in that only the matching example pairs are boosted. This is necessary because our simple classifiers cannot achieve better accuracy than chance on the non-matching pairs and, thus, fail to meet the Adaboost weak classifier criterion.

all thresholds learned by the pairwise boosting are approximately at the median of the filter response distribution and that approximating non-matching error in this manner has minimal impact on classification accuracy.

We note that several other researchers have proposed related pairwise methods for pose estimation, face recognition, and object recognition. Shakhnarovich *et al.* [16] independently select the filters that most preserve similarity. Ren *et al.* [14] learn features for identifying human motion from silhouette images using this technique. Jones and Viola [10] select a set of filters using Adaboost, with weak classifiers based on thresholding the difference of responses for same-face and different-face pairs. Mahamud and Hebert [12] model the distance between two data points as the probability that the points have different labels and estimate that probability.

Our learned set of filters (M=32 in our implementation) greatly improves upon the descriptors recently developed by Haitsma and Kalker for music identification [8]. The Haitsma-Kalker filters compute the difference between neighboring frequencies at neighboring times. These filters are equivalent to the diagonal Viola-Jones filters (Figure 2c)

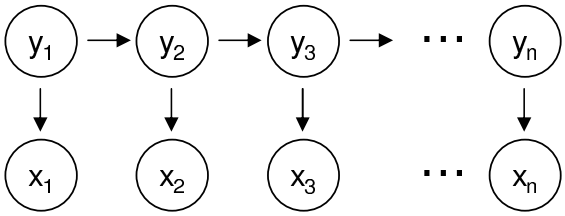

Figure 4: The simple dependency model assumed by our system when determining whether the query signature matches a signature in our database. Each x_i is a vector of bit differences in each descriptor of the signatures. We assume that each x_i is generated either by the music (y_i=1) or by an "occlusion" (y_i=0).

with a bandwith of 2 bands and a time-width of 2 frames. After learning our description, we noticed several commonalities among the filters. One is that the time-widths tend to be large (usually 54 frames or longer out of a maximum of 82 frames). Filters that have a smaller time-width tend to have a large band-width. These characteristics support our belief that filters that have a large extent in a particular dimension can "average out" much of the noise and distortion induced by poor-quality recordings. We also noticed that, out of the 32 filters, 31 either measure the difference in two sets of frequency bands at a particular time interval or a peak across frequency bands at a particular time interval. Thus, the learned filters are highly robust to noise that affect all bands intermittently but are more susceptible to distortions that affect a particular frequency range over long durations.

3.2. Learning an Occlusion Model

The previous section detailed how we learn a description and a classifier that allow us to determine whether a pair of descriptors is likely to be generated by distorted versions of the same original audio. We now describe a method for matching based on a signature, composed of a series of descriptors, that takes into account that some portions of the audio may be dominated by noise or interference. We assume that the likelihood that a recorded snippet matches an original snippet depends on the bit differences of the descriptors in each of the signatures. A snippet from a distorted version of a song in our database is unlikely to be completely attributed to interference, since some descriptors are much more likely to be generated by a distortion of the original than by background noise. We use a simple model, illustrated in Figure 4, to determine whether a descriptor most likely expresses a song from the database or some other distraction, which we term an "occlusion". Under this model, we assume that the likelihood of descriptor being generated by an occlusion depends only on the data and on whether the preceding (in time) descriptor was generated by an occlusion.

Formally, we have a signature $x^r = (x_1^r, x_2^r, \ldots, x_n^r)$

composed of n descriptors that is computed from a recorded or otherwise distorted audio snippet. We model the likelihood that the signature was generated by a distortion of a particular original snippet with signature $x^o = (x_1^o, x_2^o, ..., x_n^o)$ as follows:

$$P(x^r|x^o) = P(x^{r-o}) = \prod_{i=1}^{n} P(x_i^{r-o}|y_i)P(y_i|y_{i-1}). \quad (2)$$

x_i^{r-o} denotes the bit differences between the recorded (x_i^r) and original (x_i^o) descriptors. $y_i = 1$ when the descriptor x_i^r is due to a distortion of the original audio, and $y_i = 0$ when the descriptor is due to an occlusion that obfuscates the original audio. The descriptor difference $x_i^{r-o} \in \{0,1\}^M$ is an M-bit vector that denotes whether the thresholded filter outputs of the descriptor from the recording and the original have the same value. We model the distribution of x_i^{r-o} as a product of independent, non-identically distributed Bernoulli random variables. In our implementation, we select M=32. Thus, we have 66 parameters to estimate: 32 Bernoulli parameters for each of $P(x_i^{r-o}|y_i = 0)$ and $P(x_i^{r-o}|y_i = 1)$, and one Bernoulli transition parameter for each of $P(y_i|y_{i-1} = 0)$ and $P(y_i|y_{i-1} = 1)$. Since, in our training data, we do not know whether a particular descriptor is generated by the music or by noise, we need a method to simultaneously estimate the labels of the data y_i and the parameters. The EM algorithm is the obvious choice. Under the assumptions of our model, the E- and the M-steps are straightforward to derive, but we do not display the equations here due to space restrictions.

For a given query signature x^r, we find the signature x^o in our database that maximizes $P(x^r|x^o)$, as in Equation 2. Finally, we decide if the query signature matches the most likely database signature based on

$$P(x^r|x^o) > T, \quad (3)$$

where the threshold T controls the precision-recall tradeoff. This is approximately equivalent to a decision based on the posterior $P(x^o|x^r)$, since all x^r are approximately equally likely and since we assume that all database songs are equally likely to be queried.

4. Retrieval

Using the representation described in the previous sections, we build signatures for all of the songs in the database. During retrieval, we perform a similarity search for each of the query snippet's descriptors against this signature database. The large size of our database and the high number of queries required for each snippet motivates us to seek efficient schemes for similarity search in high-dimensional (typically 32-bit) descriptor space. A natural choice is locality-sensitive hashing (LSH) [9], a technique that enables approximate similarity searches in sub-linear time, particularly since it is so well-suited for the Hamming distance metric [7]. Our initial experiments using LSH gave excellent results, but, somewhat surprisingly, we discovered that our descriptors are so robust that *direct indexing*, using a classical hash table, greatly reduced running time without significantly impacting accuracy. We describe this indexing approach in the remainder of this section.

We hash all of the signatures into a standard hash table (keyed by appropriate M-bit descriptors). We define those descriptors within a Hamming distance of 2 from the given query to be near-neighbors. These are retrieved with the following sequence of exhaustive probes. First, we probe the hashtable with the query; this retrieves all matches within a Hamming distance of 0. Next, we make M additional probes in the hash, each consisting of the query descriptor with a single bit flipped; this finds matches at a Hamming distance of 1. Finally, we repeat this process with every combination of two-bit flips to retrieve those descriptors at a Hamming distance of 2. While such an approach may initially appear to be inefficient, we have observed that it is significantly faster than LSH for our application because each probe is so inexpensive and it returns exact rather than approximate results. We have observed that the use of unweighted Hamming distance instead of classifier confidence as a basis for descriptor similarity is a reasonable approximation, since we found the confidence values for different weak classifiers to be nearly equal in our experiments.

Once all of the near neighbors have been found, we need to identify the song that best matches the set of descriptors in the query. Rather than simply voting based on the number of matches, we employ a form of geometric verification that is similar to that used in object recognition using local features [11]. For each candidate song, we determine whether the matched descriptors are consistent over time. For this, we use RANSAC [5] to iterate through candidate time alignments and use the EM score, the likelihood of the query signature being generated by the same original audio as the candidate signature (Equation 2), as the distance metric. We have explored two alignment models. The first assumes that the query can be aligned to the original once a single parameter (temporal offset) has been determined. In this case, the minimal set is a single pair of matching descriptors. The second model assumes that the query could be a temporally scaled (linearly stretched or compressed) version of the original. This model is defined by two parameters (speed ratio and offset) and requires a minimal set of two matching descriptors. More complicated temporal distortion models are certainly possible. In practice, we have found that the first model gives accurate results, particularly since our query snippets are short. We find that RANSAC converges in fewer than 500 iterations even in the presence of significant occlusion. Once all of the retrieved candidates have been aligned, we select the song with the best

EM score, assuming that it passes a minimum threshold.

5. Implementation

The music retrieval application consists of two parts — the music identification server and the graphical user interface (GUI), which can run on different machines and communicate over sockets. We first describe how we generate the audio signature database, then describe the query phase, and finally touch on the user interface.

For each song in the database, we build an audio signature as described below; the same preprocessing is later used on the query snippet. We first compute the spectrogram image, as in [8]. We convert each song into mono and downsample to 5512.5 KHz. For a CD quality audio signal sampled at 44.1 KHz, we convolve the signal with a low pass filter and take every 8th sample. Next, we apply a short-term Fourier transform with a window size of 2048 samples (0.372 s) with successive windows offset by 64 samples (11.6 ms). We divide the power between 300 Hz and 2000 Hz into 33 logarithmically spaced bands. This frequency range corresponds to the range that can be easily transmitted over mobile phones. We use logarithmic spacing since the power distribution of typical audio is approximately logarithmic across frequency. Finally, we apply the 32 learned filters and thresholds to get a 32-bit descriptor for every time step (11.6 ms) of the signal; this series of descriptors is known as the signature. For an average-length song of 200 seconds, the storage requirement for this representation is approximately 70KB. We load all of the descriptors into memory and put them into a hash table, along with the song id and frame id (temporal offset). Although the main memory implementation works well for a few thousand songs, other methods may be necessary for larger databases.

During the query phase, the music identification server builds a similar set of 32-bit descriptors for the audio snippet. For each descriptor, it finds all descriptors that are within a Hamming distance of 2 bits (empirically determined to be a good balance between accuracy and matching speed). Finally, we perform geometric alignment using RANSAC and find the best match according to the EM score (Equation 3).

The GUI frontend records audio clips from the microphone and sends the waveform to the server for identification. Figure 5 shows a screenshot of our GUI. The bottom left and right panels displays the spectrogram of the recorded and original songs, respectively. Although the raw spectrograms look different due to noise and occlusion, one can see similar structures. The text panel gives the name of the the song, correctly identified.

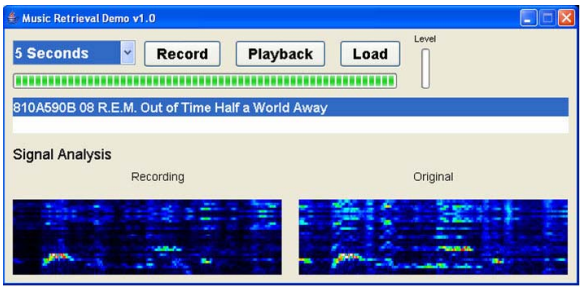

Figure 5: Spectrograms for the corrupted query and the corresponding region from the correctly-identified song.

6. Evaluation

We present three sets of experiments that evaluate the performance of our system. First, we present results at the descriptor level, showing that our representation far outperforms the descriptors described in [8]. Second, we provide results that demonstrate our system's accuracy at the song level. Finally, we show results that explore the effect of various design decisions and parameter settings.

6.1 Experimental setup

We need to train the two parts of our system: the filters for extracting descriptors and the EM noise model. Both requires training data consisting of aligned pairs of filter outputs. This poses a chicken-and-egg alignment problem: how can we accurately align noisy recordings to original songs before learning good descriptions? Our solution is to bootstrap the learning process with synthetically-distorted songs, for which the alignment is known. From these we learn a set of filters that, while insufficiently accurate for music identification in noisy environments, is suitable for training data alignment.

The training data consists of 78 songs played through low-quality speakers and recorded using low-quality microphones, aligned to the originals using the bootstrap filters. We learned the 32-bit filters and the EM noise parameters as described in section 3. Next, we recorded test data in a completely different environment using different microphones, speakers, computers and recording rooms. The experiments use two challenging real-world test sets designed to exemplify worst-case scenarios. The first consists of 71 songs played at a low volume and recorded with a distorted microphone (denoted as "Test A"). The second more difficult set contains 220 songs captured with a very noisy recording setup (denoted as "Test B"). In many cases, the noise drowns out the music to such a degree that the song is barely audible to humans. These recordings were drawn from a database of 145 albums with 1862 songs spanning a large variety of music genres including classical, vocal, rock and pop. Since each query could match up to 1861 false positives, the baseline accuracy of the tasks was only 0.05%.

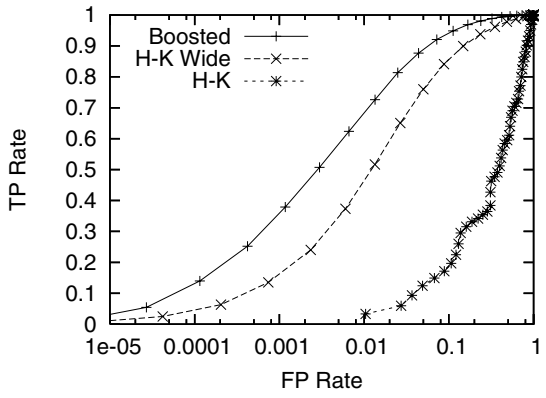

Figure 6: ROC curve comparing descriptor performance. Boosted dominates H-K and H-K Wide. Note semi-log scale.

6.2 Descriptor Performance

To test the descriptor performance in isolation, we generated a data set with approximately one hundred thousand positive and one million negative examples. The positive examples were pairs of matching descriptors (from originals and their corresponding noisy recordings), sampled in 15 second snippets in each of 71 songs from Test A. The negative examples were pairs of non-matching descriptors drawn from the same audio data. Three types of descriptors were tested: Haitsma and Kalker's descriptor from [8], denoted "H-K"; our improvement on H-K as described below, denoted "H-K Wide"; and our descriptor learned using pairwise boosting (denoted "Boosted"). Since all of the descriptors are the same length (32-bits), we varied the Hamming distance threshold from 0 to 32 to generate the ROC curves shown in Figure 6. We note that Boosted dramatically outperforms H-K over the entire ROC curve. We hypothesised that a reason for the H-K descriptors' unexpectedly poor performance is that the filters span only a short time interval and may therefore be highly susceptible to noise and slight misalignments. We constructed H-K Wide by extending the filter width from 2 to 54 frames. As can be seen, our H-K variant performs significantly better than the original H-K, but is still consistently dominated by Boosted. These results validate our belief that pairwise boosting can improve on manually-engineered descriptors by learning from data.

To enable fast retrieval through direct hashing, we need to find candidate descriptors using a Hamming distance of no more than 2 bits. For a 10 second query containing 860 descriptors, this requires a recall at the descriptor level of at least several percent to ensure that we find the matching signature. Table 1 shows our results for Hamming distances of 3 bits or fewer. At these distances, Boosted achieves recall rates that are 1–2 orders of magnitude higher than the recall rates of the engineered H-K Wide description. Consequently, with a Hamming distance threshold of 2 on 10 second queries from Test A, the H-K Wide descriptors cannot obtain a recall of higher than 6%, while Boosted achieves 92% recall at 97% precision.

	Distance Threshold			
	0	1	2	3
Boosted	1.1%	5.4%	14.0%	25.2%
H-K Wide	$< 0.01\%$	0.09%	0.64%	2.5%
H-K	$< 0.01\%$			

Table 1: Recall of descriptors with various Hamming distance thresholds.

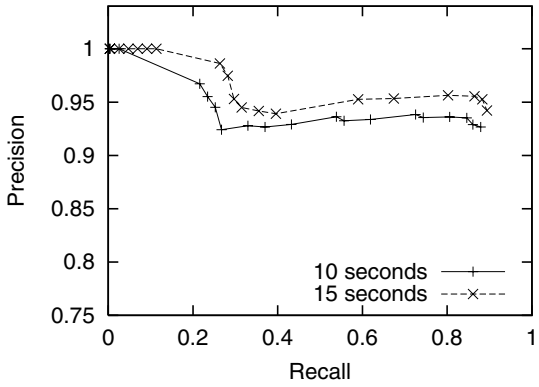

Figure 7: Song retrieval rate on combined dataset, with different length audio query snippets.

6.3 System Performance

Figure 7 presents song retrieval rates on the combined test sets A and B, with query snippets of 10 and 15 seconds, corresponding to 860 and 1290 descriptors, respectively. We see that the longer query slightly improves retrieval results. In a practical system, users wish to achieve the desired accuracy using as short a query as possible.

Figure 8 shows results for the individual datasets on 10 second recordings. For Test A, we achieve 90% recall at 96% precision. Test B is more challenging, but we are still able to achieve 80% recall at 93% precision.

Figure 9 shows song retrieval rates for Hamming distance thresholds of 0, 1, and 2 bits on Test A. Performance improves as we allow more bit errors, but the marginal gain drops after a distance of 1 while query time continues to increase exponentially, hence our design decision to restrict near-neighbor descriptor searches to be within 2 bits.

7. Conclusion

This paper demonstrates that computer vision techniques can make a significant contribution to topical problems in the audio domain. Our research contributions can be summarized as follows. First, we show how certain audio tasks,

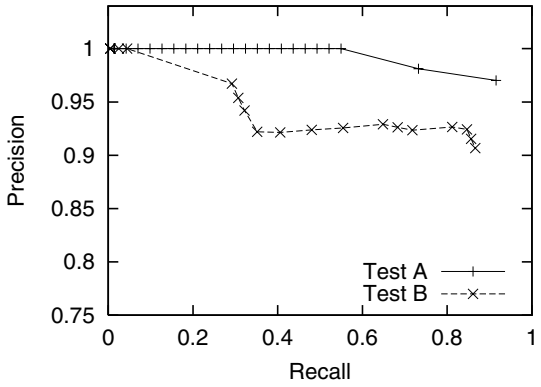

Figure 8: P-R curves for song-level retrieval on individual datasets, with 10 second queries.

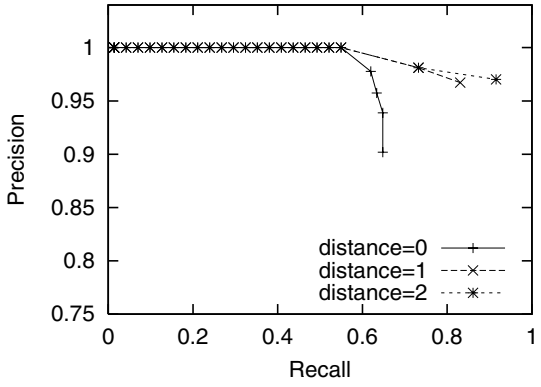

Figure 9: Impact of Hamming threshold on song retrieval.

such as music identification, can be transformed into familiar 2-D computer vision problems. Second, we propose a pairwise variant of boosting to learn discriminative descriptors and apply it to a classification problem with thousands of classes. Third, we incorporate our algorithms into a practical system that quickly and accurately identifies songs from a few seconds of noisy, distorted audio. Finally, we demonstrate, using rigorous experiments on real data, that our system significantly outperforms current approaches in content-based music identification.

In future work, we will study the scaling properties of our system by indexing larger music collections. We also hope to explore the problem of recognizing variants on a piece of music, such as recordings of the same song by different artists. Additionally, we plan to improve the accuracy of our algorithm through improved likelihood models. We believe that computer vision ideas have immediate applicability and direct relevance to many other domains and hope that this paper encourages computer vision researchers to explore such opportunities.

Acknowledgments

We would like to thank L. Huston, P. Pillai, J. Campbell, G. Shakhnarovich, R. Dannenberg and P. Robinson for their useful feedback on this research.

References

[1] AT&T Wireless. `http://www.attwireless.com/`.

[2] Musikube. `http://www.musikube.com/`.

[3] Shazam Entertainment. `http://www.shazam.com/`.

[4] P. Cano, E. Batlle, T. Kalker, and J. Haitsma. A review of algorithms for audio fingerprinting. In *Workshop on Multimedia Signal Processing*, 2002.

[5] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 1981.

[6] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Proceedings of International Conference on Machine Learning*, 1996.

[7] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of International Conference on Very Large Databases*, 1999.

[8] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. In *Proceedings of International Conference on Music Information Retrieval*, 2002.

[9] P. Indyk and R. Motwani. Approximate nearest neighbor – towards removing the curse of dimensionality. In *Proceedings of Symposium on Theory of Computing*, 1998.

[10] M. Jones and P. Viola. Face recognition using boosted local features. Technical Report MERL-TR-2003-25, Mitsubishi Electric Research Laboratory, 2003.

[11] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of International Conference on Computer Vision*, 1999.

[12] S. Mahamud and M. Hebert. Minimum risk distance measure for object recognition. In *Proceedings of International Conference on Computer Vision*, 2003.

[13] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics, 1997.

[14] L. Ren, G. Shakhnarovich, J. Hodgins, P. Viola, and H. Pfister. Learning silhouette features for control of human motion. Technical Report CMU-CS-04-165, Carnegie Mellon University, 2004.

[15] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3), 1999.

[16] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter sensitive hashing. In *Proceedings of International Conference on Computer Vision*, 2003.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of Computer Vision and Pattern Recognition*, 2001.