Content-Based Retrieval of Music and Audio

Jonathan T. Foote
Institute of Systems Science
National University of Singapore
Heng Mui Keng Terrace, Kent Ridge
Singapore 119597

ABSTRACT

Though many systems exist for content-based retrieval of images, little work has been done on the audio portion of the multimedia stream. This paper presents a system to retrieve audio documents by acoustic similarity. The similarity measure is based on statistics derived from a supervised vector quantizer, rather than matching simple pitch or spectral characteristics. The system is thus able to learn distinguishing audio features while ignoring unimportant variation. Both theoretical and experimental results are presented, including quantitative measures of retrieval performance. Retrieval was tested on a corpus of simple sounds as well as a corpus of musical excerpts. The system is purely data-driven and does not depend on particular audio characteristics. Given a suitable parameterization, this method may thus be applicable to image retrieval as well.

Keywords: Multimedia music audio content-based retrieval

1. INTRODUCTION

This paper presents an audio search engine that can retrieve sound files from a large corpus based on similarity to a query sound. Sounds are characterized by "templates" derived from a tree-based vector quantizer trained to maximize mutual information (MMI). Audio similarity can be measured by comparing templates, which works both for simple sounds and complex audio such as music. Discriminative training allows desired differences to be detected while ignoring unimportant sources of variability. Similar measures have proved successful for talker identification and audio classification.[1,2] Unlike other approaches based on perceptual criteria,[3–7] this technique is purely data-driven and makes no attempt to extract subjective acoustic parameters (like "brightness" or "pitch"). Unlike hidden Markov modeling, this method is computationally inexpensive, yet is robust even with only a small amount of data. Thus it is inexpensive both in computation and storage requirements. These factors indicate that this technique would probably scale quite well to extremely large audio collections. Because the method does not depend on any particular audio characteristics, it should work on other similar problems such as image retrieval.

2. OVERVIEW

The basic operation of the retrieval system is as follows. First, a suitable corpus of audio examples must be accumulated and parameterized into *feature vectors*. The corpus must contain examples of the kinds (*classes*) of audio to be discriminated between, e.g. speech and music, or male and female talkers. Next, a tree-based quantizer is constructed using the methods of Section 2.2. This is a "supervised" operation and requires requires the training data to be *labeled*, i.e. each training example must be associated with a *class*. This is all the human input required. The tree automatically partitions the feature space into regions ("cells") which have maximally different class populations.

To generate an audio template for subsequent retrieval, parameterized data is quantized using the tree. An audio file can be characterized by finding into which cells the input data vectors are most likely to fall. A template is just an estimate of the vector counts for each cell; in other words a histogram. This template captures the salient characteristics of the input audio, because sounds from different classes will have very different counts in the various histogram bins, while similar audio files should have similar counts. To retrieve audio by similarity, a template is constructed for the query audio. Comparing the query template with corpus templates will yield a similarity measure for each audio file in the corpus. These can be sorted by similarity and the results presented as a ranked list as in conventional text retrieval. Section 3.1 discusses distance measures that may be used to estimate the similarity between templates, and thus between the corresponding audio files. Section 4 presents experimental retrieval results on a corpus of sounds and another of music. Quantitative measures of retrieval performance are shown, as well as comparative results from another retrieval method on the same corpus.

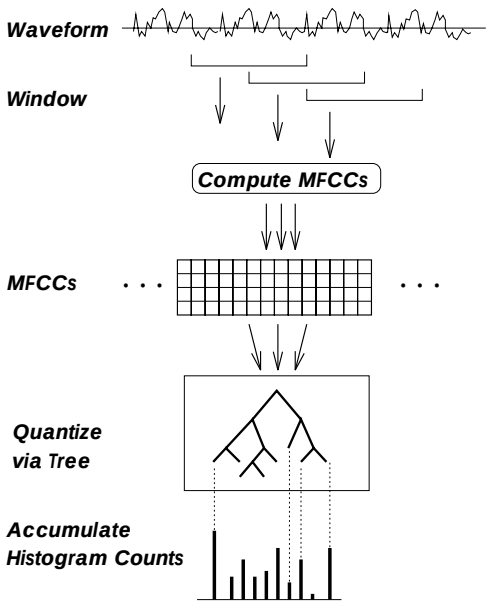

Figure 1. Audio template construction

2.1. Audio Parameterization

Before a tree can be trained and templates generated, audio files are parameterized into mel-scaled cepstral coefficients (MFCCs) plus an energy term.[8] The audio waveform, sampled at 16 kHz, is thus transformed into a sequence of 13-dimensional feature vectors (12 MFCC coefficients plus energy). This parameterization has been shown to be quite effective for speech recognition and speaker ID (though other methods might be better for more general audio, like music). Figure 2 shows the steps in the parameterization. First, the audio is Hamming-windowed in overlapping steps. Each window is 25 mS wide and are overlapped so there are 500 windows, hence feature vectors, in a second of audio*. For each window, the log of the power spectrum is computed using a discrete Fourier transform (DFT). The log spectral coefficients are perceptually weighted by a non-linear map of the frequency scale. This operation, called Mel-scaling, emphasizes mid-frequency bands in proportion to their perceptual importance. The final stage is to further transform the Mel-weighted spectrum (using another DFT) into "cepstral" coefficients. This results in features that are reasonably dimensionally uncorrelated, thus the final DFT is a good approximation of the Karhunen-Loeve transformation for the mel spectra. The audio waveform, sampled at 16 kHz, is thus transformed into 13-dimensional feature vectors (12 MFCC coefficients plus energy) at a 500 Hz rate.

2.2. Tree-structured Quantization

A tree-structured quantizer[9] is at the heart of the distance measure. Once data has been parameterized, a quantization tree (Q tree) is grown off-line using as much training data as practical. Such a tree is a vector quantizer; discriminative training ensures that it attempts to label feature vectors from different classes with a different label. Section 2.3 discusses tree construction in more detail.

Unlike the more common K-means vector quantization (VQ), the tree-based quantization is supervised, which means the feature space may be profitably discretized into many more regions than the conventional minimum-distortion vector quantizers. Supervised training means the quantizer is able to learn the critical distinctions between classes, while ignoring other variability. For example, in speaker identification, the system learns to discriminate the subtle vocal-tract differences between speakers while ignoring the huge (though unimportant) variations between spoken phones (e.g. vowels vs. fricatives).

The tree structure presented here can arguably handle the "curse of dimensionality" better than many other methods, because only one dimension is considered at each decision node. Dimensions that do not help class

*This is a more rapid rate than for speech processing; the advantage is that more data is available for very short audio files.

Figure 2. Audio parameterization into mel-frequency cepstral coefficients

discrimination are ignored, in contrast to other measures which must be computed for all dimensions. Another advantage of handling high-dimensional feature spaces is the ability to explicitly model time variation: if adjacent feature vectors are concatenated into a "super-vector," the tree can discover time-dependent features such as slopes or trends. For music retrieval, this means the technique can find similarity between similar slides, intervals or scales, despite lumping all the time-dependent feature vectors into one (time-independent) template.

2.3. Tree construction

A quantization tree partitions the feature space into a number of discrete regions (analogous to the Voronoi polygons surrounding VQ reference vectors). Each decision in the tree involves comparing the vector with a fixed threshold, and going to the left or right child depending on whether the value is greater or lesser. Each threshold is chosen to maximize the mutual information $I(X;C)$ between the data X and the associated class labels C that indicate the class of each datum.

Because the construction of optimal decision trees is NP-hard, they are typically grown using a greedy strategy.[10] The first step of the greedy algorithm is to find the decision hyperplane that maximizes the mutual information metric. While other researchers have searched for the best general hyperplane using a gradient-ascent search,[11] the approach taken here is to consider only hyperplanes normal to the feature axes, and to find the maximum mutual information (MMI) hyperplane from the optimal one-dimensional split. This is computationally reasonable, easily optimized, and has the advantage that the search cost increases only linearly with dimension.

To build a tree, the best MMI split for all the training data is found by considering all possible thresholds in all possible dimensions. The MMI split threshold is a hyperplane parallel to all feature axes except dimension d, which it intercepts at value t. This hyperplane divides the set of N training vectors X into two sets $X = \{Xa, Xb\}$, such that

$$Xa: \quad x_d \geq t_d \tag{1}$$

$$Xb: \quad x_d < t_d. \tag{2}$$

This first split corresponds to the root node in the classification tree. The left child then inherits Xb, the set of training samples less than the threshold, while the right child inherits the complement, Xa. The splitting process is repeated recursively on each child, which results in further thresholds and nodes in the tree. Each node in the tree corresponds to a hyper-rectangular region or "cell" in the feature space, which is in turn subdivided by its descendants. Cells corresponding to the leaves of the tree completely partition the feature space into non-overlapping regions, as shown in Figure 3.

To calculate the mutual information $I(X;C)$ of a split, consider a threshold t in dimension d. The mutual information from the split is easily estimated from the training data in the following manner. Over the volume of the current cell, count the relative frequencies:

$$\begin{aligned} N_{ij} &= \text{Number of data points in cell } j \text{ from class } i \\ N_j &= \text{Total number of data points in cell } j \\ &= \sum_i N_{ij} \\ A_i &= \text{Number of data points from class } i : x_d \geq t_d \end{aligned}$$

In the region of cell j, define $Pr(c_i)$ to be the probability of class i and $Pr(a_i)$ as the probability that a member of class i is above the given threshold. These probabilities are easily estimated as follows:

$$\Pr(c_i) \approx \frac{N_{ij}}{N_j}, \quad \Pr(a_i) \approx \frac{A_i}{N_{ij}}. \tag{3}$$

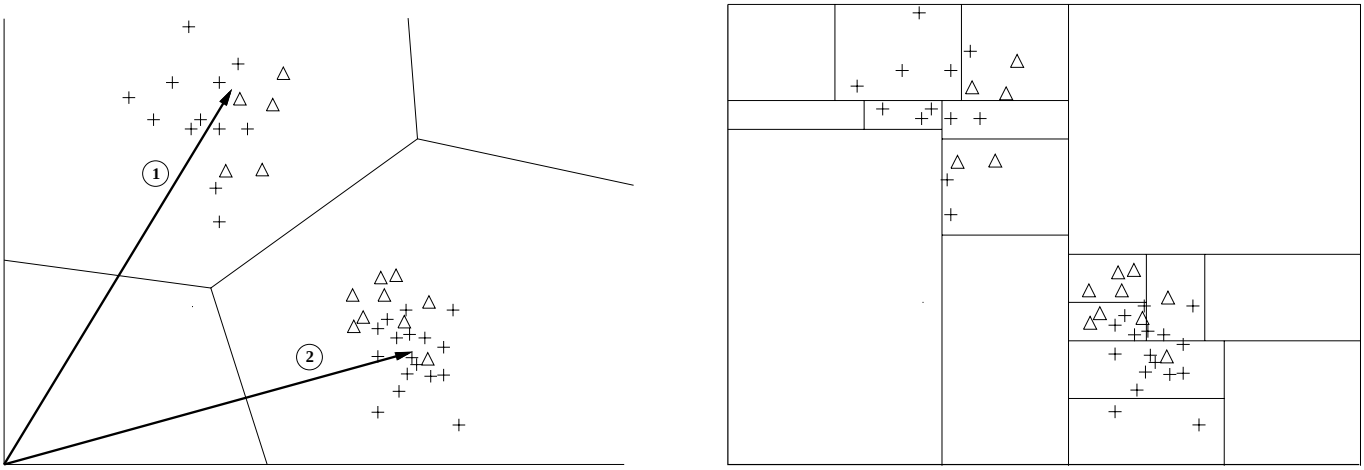

Figure 3. nearest-neighbor VQ (left) and MMI tree (right) feature space partitions

With these probabilities, the mutual information given the threshold may be estimated in the following manner (for clarity of notation, conditioning on the threshold is not indicated):

$$I(X;C) = H(C) - H(C|X) \tag{4}$$

$$= -\sum_i \Pr(c_i) \log_2 \Pr(c_i) + \sum_i \Pr(c_i) H_2\left(\Pr(a_i)\right) \tag{5}$$

$$\approx -\sum_i \frac{N_{ij}}{N_j} \log_2 \frac{N_{ij}}{N_j} + \sum_i \frac{N_{ij}}{N_j} H_2\left(\frac{A_i}{N_{ij}}\right), \tag{6}$$

where H_2 is the binary entropy function

$$H_2(x) = -x \log_2(x) - (1-x) \log_2(1-x). \tag{7}$$

Equation 6 is a function of the (scalar) threshold t, and may be quickly optimized by either an exhaustive or region-contraction search.

This splitting process is repeated recursively on each child, which results in further thresholds and nodes in the tree. At some point, a stopping rule decides that further splits are not worthwhile and the splitting process is stopped. The MMI criterion works well for finding good splits, but is a poor stopping condition because it is generally non-decreasing. (Imagine a tiny cell containing only two data points from different classes: any hyperplane between the points will yield an entire bit of mutual information. Bigger cells with overlapping distributions generally have less mutual information.) Also, if the number of training points in a cell is small, the probability estimates for that cell may be unreliable. This motivates a stopping metric where the best-split mutual information is weighted by the probability mass inside the cell l_j to be split:

$$\text{stop}(l_j) = \left(\frac{N_j}{N}\right) I_j(X;C) \tag{8}$$

where N is the total number of available training points. Further splits are not considered when this metric falls below some threshold. This mass-weighted MMI criterion thus insures that splitting is not continued if either the split criterion is small, or there is insufficient probability mass in the cell to reliably estimate the split threshold.

3. TREE-BASED TEMPLATE GENERATION

The tree partitions the feature space into L non-overlapping regions or "cells," each of which corresponds to a leaf of the tree. Though the tree can be used as a classifier, by labeling each leaf with a particular class. Such a classifier

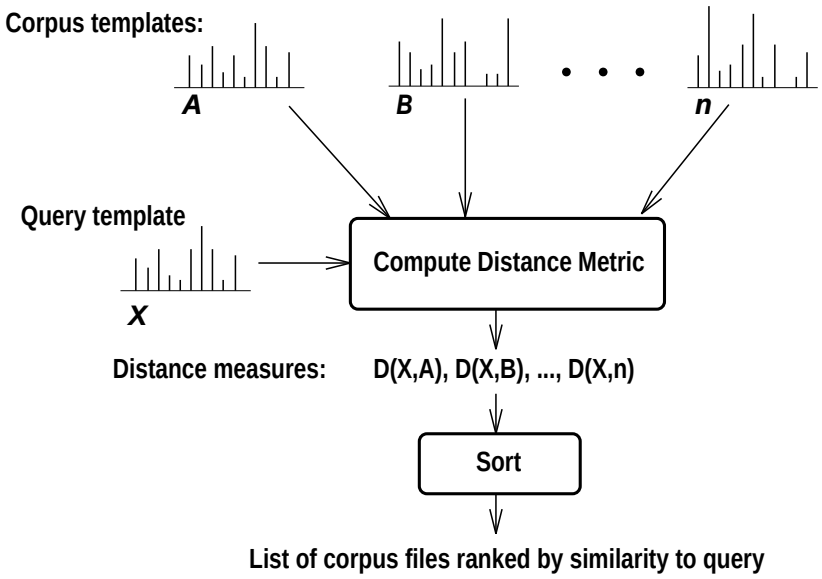

Figure 4. Audio classification using histogram templates

will not be robust, as in general classes will overlap such that a typical leaf will contain data from many different classes. A better way to capture class attributes is to look at the the ensemble of leaf probabilities from the quantized class data. Two hundred milliseconds of data will result in 100 feature vectors (ignoring window effects), and thus 100 different leaf labels. If a histogram is kept of the leaf probabilities, such that if, say, 14 of the 100 unknown vectors are classified at leaf j then leaf j is given a value of 0.14 in the histogram. The resulting histogram captures essential class qualities, and thus serves as a reference template against which other histograms may be compared.

A tree-structured quantizer is especially practical because it may be pruned to different sizes depending on the amount of data. Because there is one histogram bin for each leaf in the tree, the tree size directly determines the size of the histogram template. If data is sparse, a large histogram will be suboptimal as many bin counts will be zero. Pruning the tree will result in fewer bins, which may be able to better characterize the data. In this fashion the number of free parameters can be adjusted to suit the application. Also, quantization is extremely rapid: for a N-leaf tree, quantizing a vector takes only $log(N)$ 1-dimensional compares.

3.1. Distance Metrics

Once templates have been computed for different audio sources, measuring the similarity between the templates yields a measure of acoustic similarity. Though it is not obvious how to choose an appropriate distance measure to compare the templates, simple approaches work well in practice. Several distance measures have been used in implementation, including symmetric relative entropy[1] (the *Kullback-Liebler* metric). Two distance measures were investigated in the experiments of Section 4:

- *Euclidean distance*

$$D_E^2(p,q) = \sum_{i=1}^{N} [p(i) - q(i)]^2 \tag{9}$$

  This measure treats the histograms as vectors in N-dimensional space, and computes the L2 (Euclidean) distance between them. Though there is no true probabilistic justification for this measure, it is closely related to to the χ^2 (chi-squared) measure, and has been used successfully for speaker ID.[2]

- *Cosine distance*

$$D_C(p,q) = \frac{\sum_{i=1}^{N} p(i)q(i)}{\sqrt{\sum_{i=1}^{N} p(i)^2 \times \sum_{i=1}^{N} q(i)^2}} \tag{10}$$

This measure again treats the histograms as N-dimensional vectors, and computes the cosine of the angle between them.[12] It is thus insensitive to the relative magnitudes of the vectors, which are a function of the audio file length. This measure, along with weighted variations, has been used quite successfully for measuring the similarity of text documents.[13]

4. EXPERIMENTS

Two sets of retrieval experiments were performed. The first involves retrieving audio files from a corpus of sounds that can be considered "simple," that is, from only once source. Examples include laughter, musical and percussive instrument notes played individually, spoken words, animal cries, and thunder. Another experiment attempts to retrieve sounds from a corpus of musical clips. This is a more difficult task because the audio consists of multiple notes coming from (usually) multiple sources, often with singing, reverberation, and/or distortion. Analyses used for simple sounds (such as estimates of pitch or brightness) are less appropriate for music because of its inherently greater complexity.

4.1. Measuring Retrieval Performance

The output of the retrieval process is a list of audio files, ranked by their similarity to the query file. (This will be familiar to anyone who has used AltaVista or other web search engines.) Ideally, the most similar files will be at the head of the ranked list: if one's query is an oboe sound, other oboe sounds should be highly ranked, while dissimilar sounds, such as bells or thunder, should be lower. Quantitatively measuring retrieval performance requires *relevance assessments*, that is, judgments by a human of how relevant (similar) each file in the corpus is to a particular query. For audio, of course, this is a rather subjective area (see Section 4.3); my approach has been to use the filenames as a restrictive, though quasi-objective, indication of similarity. Thus the files `oboe23` and `oboe19` are regarded as relevant to each other (as are all files of the form `oboe*`); all other files are regarded as irrelevant (dissimilar).

Once relevance classes have been defined, retrieval performance is can be measured by *precision*, the proportion of retrieved messages that are relevant to a particular query at a certain position in the ranked list. An accepted single-number performance figure is the *average precision* or AP. For each query, the precision values are averaged for each relevant document in the ranked list. The results are then averaged across the query set, resulting in the average precision. For example, given only one file relevant (similar) to the query, an AP of 1.0 means that the retrieval system ranked it first. The AP can be loosely interpreted as the percentage of top-ranked files that are actually relevant, e.g. an AP of 0.3 means roughly 3 out of the 10 top-ranked files are indeed relevant. The average precision figures presented here were calculated using the TREC evaluation software[14] developed for evaluating the performance of text retrieval systems.

4.2. Experiment 1: retrieving simple sounds

For this experiment, a corpus of 409 sounds was used. The sounds were in ".au" format consisting of 8-bit (companded) samples at a sampling rate of 8kHz. These were up-sampled to 16kHz and 16-bit linear format for the MFCC parameterization described above in Section 2.2. Two basic experiments were performed. The first was "quasi-unsupervised": each sound was taken as an instance of a separate audio class, hence the tree was trained to differentiate between the 409 different audio samples. Note that though the tree training is still supervised, there is no real meaning to the class labels (other than that the files were different), hence the designation "quasi-unsupervised." A large tree was grown using "supervectors" of 5 concatenated feature vectors for a total dimensionality of 65. This tree was pruned to the various sizes used in the experiments. For the second supervised experiment, audio files were grouped into 41 different classes on the basis of their file names. (For example, all oboe samples having a filename of the form `oboe*` were grouped into one class.) A second tree was similarly trained to separate these classes. Though it can be argued that this is not good experimental practice, as it involves training on testing data, it can be justified for several reasons: it allows direct comparison with the Muscle Fish results[4]; there was no obvious additional training corpus available; there was insufficient data for meaningful cross-validation experiments; and it probably reflects the "real-world" application where one would wish to use all the data available (including any prior knowledge about classes). Certainly additional experiments are warranted to determine how well the retrieval works on unseen files.

Retrieval results were measured for 6 of the 409 sounds, using both cosine and Euclidean distance measures, and both supervised and quasi-unsupervised trees. The chosen six sounds come from the examples on the Muscle Fish

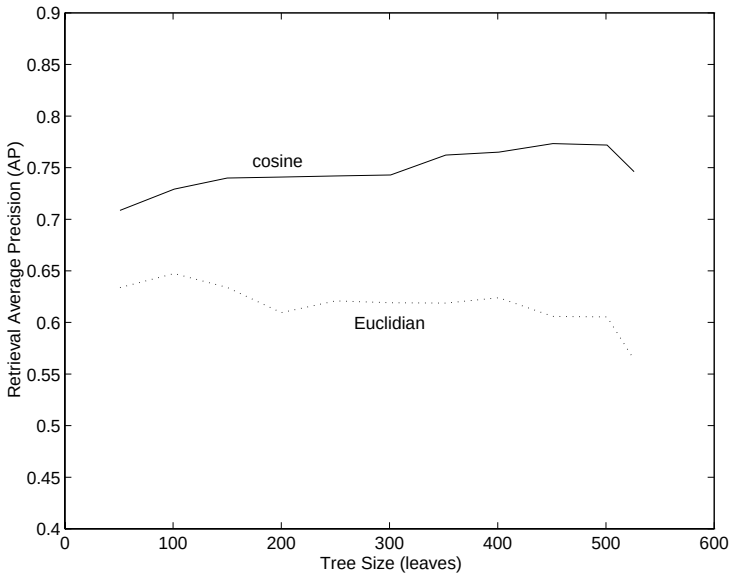

Figure 5. Retrieval performance vs. tree size (number of histogram bins) for cosine and Euclidean distance measures

demonstration page[†], and can thus be expected to show at least the Muscle Fish retrieval approach in reasonable form. Figure 5 shows retrieval performance versus tree size for both distance metrics. The cosine distance measure performed better than the Euclidean for all experiments, though it is not clear why. The cosine measure was reasonably insensitive to the number of histogram bins, though performance suffers at the extremes: too few bins, and dissimilar vectors are lumped into the same bin, hurting performance, while too many bins similar vectors are placed into different bins, which also hurts retrieval. Muscle Fish retrieval results (from their web demonstration) were also analyzed for comparison. Table 1 summarizes the results from experiments using 500-bin templates. (Euclidean results are not shown because the cosine distance measure consistently outperformed the Euclidean measure by 10%–20%)

4.3. Comparative Results

It is interesting to compare the performance of the Muscle Fish retrieval results with those from the quantization (Q) tree. The Muscle Fish results are not supervised, though they do make use of statistics from the entire database[‡]. The Muscle Fish retrieval system includes additional weighting by duration, pitch, and loudness. Results in the columns marked "no DPL" and "+ DPL" respectively ignore and include the additional weighting. The "no DPL" column is probably a better comparison with the Q tree results, as the latter method does not explicitly try to match any of these quantities. Simple duration or loudness matching would be a relatively straightforward addition to the algorithm and would doubtless improve performance (especially on tasks like the oboe sounds as these are very similar in duration and loudness). The Muscle Fish results, which are based on psychoacoustically-derived measures, seem to do a better job of capturing the timbre of a particular sound. This is particularly apparent in the retrieval of oboe sounds, especially when duration, pitch, and loudness are taken into account. The Q tree approach, on the other hand, doesn't retrieve oboe sounds nearly as well. One reason for this is the Q tree retrieval tends to rank similar-pitched sounds higher than sounds with similar timbres but different pitches. There are many other instrument sounds in the database (trombone, 'cello, etc.) and samples of these with pitches near the query oboe pitch were ranked higher in the Q tree results than other oboe samples. Because these similar-pitch sounds were not considered relevant (while all oboe samples were), the Muscle Fish retrieval scored much better. This underlines the subjective nature of audio similarity: it is not clear which criterion is more important—the appropriate choice is probably application-dependent. (Similarly for image retrieval, the relative importance of shape vs. color is not clear.)

[†]These sounds may be heard at the Muscle Fish web site `http://www.musclefish.com/cbrdemo.html`

[‡]Erling Wold, personal communication.

Distance	Q Tree (D_C) "unsupervised"	Q Tree (D_C) supervised	Muscle Fish (no DPL)	Muscle Fish (+ DPL)
Laughter (M)	0.68	0.82	1.00	1.00
Oboe	0.11	0.43	0.69	0.94
Agogo	1.00	1.00	0.53	0.58
Speech (F)	0.77	0.87	0.69	0.94
Touchtone	0.61	1.00	0.44	0.73
Rain/thunder	0.22	0.35	0.30	0.42
Mean AP	0.580	0.772	0.608	0.768

Table 1. Retrieval Average Precision (AP) for different retrieval schemes. Quantization tree results used unweighted cosine distance measure (D_C) with 500-bin histogram templates.

4.4. Experiment 2: retrieving music

The Q tree approach has the advantage that it can handle very complicated data, essentially by ignoring variation not germane to the classification task. As a demonstration, another retrieval experiment was performed on a corpus of recorded music. The corpus consisted of 255 7-second clips of recorded music; roughly five clips apiece from each of 40 artists or styles (a few styles, apparently from compilation albums, had 10 clips). Representative genres include jazz, pop, rock, rap, and techno, as well as Brazilian music, plainsong, solo piano, guitar, and "easy listening." All music from each artist was considered a class. For retrieval assessment, all clips from each artist were considered relevant to that artist only. (This assumption is almost certainly too narrow: songs by the same artist can sound less similar than other songs by different artists.) Table 2 summarizes some preliminary results. As in the previous experiment, the supervised cosine distance measure performed the best. Though the actual Average Precision values appear low in comparison, this is to be expected given the unrealistic relevance assessments. In practice the music retrieval seems quite effective. It is interesting to examine the system's errors, which are usually sensible: for example reggae is often ranked similar to blues, as both are rhythmic, bass-heavy, and have relatively clean guitar accompanying a prominent male vocal. The reader is invited to judge the performance of the retrieval system: an on-line demonstration is available[§].

A further extension to the distance measures of Section 3.1 was investigated. To motivate the discussion, consider the following thought experiment. Imagine a two templates trained on different classes of audio. To emphasize the difference, let's use classical music for one class and heavy metal for another. Any arbitrary audio file can then be characterized by a two-element vector consisting of its similarity to each template. This vector can then serve as a "second-level" template: similar files should hopefully have similar distance vectors, regardless of classicality or heaviness. For experimental verification, each music file was given a vector consisting of the Euclidean distances to all other files (one element of which will be zero). File similarity was calculated as the Euclidean distance between distance vectors, and retrieval performance, shown in the last column of Table 2, was respectably close to that of the unsupervised Euclidean template measure. This approach might be appropriate for a large corpus of music files, for example maintained by an on-line music vendor. Given good templates of musical genres, users would be able to query the archive based not only on audio similarity but by genre as well, and files could be easily added to the corpus without additional tree training. Again, more experiments are needed to see how performance extends to unseen files or larger corpora.

5. CONCLUSIONS AND FUTURE DIRECTIONS

Preliminary experiments suggest there is some value in "bin weighting." This idea is motivated by text-based information retrieval, where query words are typically weighted according to their frequency in the corpus. A word occurring in many documents is assumed to have less information and is accordingly weighted less. Though a histogram bin is not a good analogue to a word, there may be a similar effect: certain bins may be more "important"

[§] `http://repulse.iss.nus.sg:8080/cgi-bin/audio-cbr`

Distance:	Euclidean (D_E) supervised	Euclidean (D_E) "unsupervised"	Cosine (D_C) supervised	Vector distance
AP	0.35	0.32	0.40	0.31

Table 2. Retrieval Average Precision (AP) for music retrieval experiment

for discrimination and might usefully be weighted higher in the distance measure. If the distribution of counts in a bin is reasonably uniform across the corpus, then that bin may be less important than another bin having many counts for certain files and few for others. The entropy of the cross-corpus bin counts might be a good way to measure uniformity (the entropy of a uniform distribution is maximum); experiments weighting bins by inverse entropy are underway.

This technique offers perhaps a way to measure subjective perceptual qualities of sounds, often described in terms like "brightness" and "harmonicity." Rather having to define and compute an actual measure of these relatively subjective terms, it is possible to train a template with a number of example files deemed to have (or not have) the given quality. The resultant distance from that template may be used as a measure of the particular quality, without having to explicitly define or calculate it.

Another more difficult application is to automatically segment multimedia sources by audio *changes*, such as between different speakers,[15–17] pauses, musical interludes, fade-outs, etc. Because the identification technique works well enough on a sub-second time scale, it could be used to detect these changes simply by looking at the histogram generated by a short running window over a longer audio stream. Comparing the window histogram with pre-trained templates would allow detection and segmentation of speech, particular speakers, music, and silence. Another approach would be to compute the distance between the histogram of a short window with a longer window, which might yield a measure of audio novelty by the degree that short-term statistics differ from a longer-term average.

An interesting side-effect of tree construction is that the relative importance of features can be estimated. The relative "importance" of each feature dimension can be judged by looking at the structure of the tree. If a dimension is never used for a split, it gives no information and may be safely ignored. Conversely, a dimension used for many or most of the splits must carry important information about the class distributions. This allows other parameterizations to be easily investigated by adding extra feature dimensions and looking at the structure of the resulting tree. There is probably room for experiments of this sort because the MFCC parameterization has been tweaked for speech recognition and there is no real evidence it is optimal for general audio.

A large motivation for using MFCC parameterization is because the resulting features are reasonably uncorrelated. Since the tree quantizer can usefully model correlation, it may be possible to find parameterizations that better capture speaker-dependent features, especially when the importance of additional features can be judged by the tree. Additional features such as pitch or zero-crossing rate (as in 18) would probably aid classification. An interesting possibility, yet unexplored, is to use compressed audio (for example MPEG encoded parameters) directly. This would eliminate the need for the parameterization step as well as decoding and would thus be extremely rapid.

An effective method for audio retrieval has been presented, showing that useful retrieval can be performed even for complex audio. Moreover, audio retrieval is very modest in computation and storage requirements. Given the very modest storage requirements (a few hundred integer bin counts per template), this method should be practical for even extremely large archives. One can imagine an "AudioVista[¶]" application, where a web spider roams the net searching for audio files. Once located, suitable files are downloaded, and templates are computed. A large collection of templates can then be amassed in reasonable storage space, because the actual audio can be discarded provided the source URL is saved with the template.

[¶] Apologies to Digital Equipment Corporation.

ACKNOWLEDGEMENTS

This work was primarily funded by a J. William Fulbright research fellowship, administrated by the Committee for the International Exchange of Scholars. Thanks to the staff at the Institute for Systems Science for additional support, and to Erling Wold at Muscle Fish for discussions and permission to use portions of the Muscle Fish audio corpus.

REFERENCES

1. J. T. Foote and H. F. Silverman, "A model distance measure for talker clustering and identification," in *Proc. ICASSP '94*, vol. S1, pp. 317–32, IEEE, (Adelaide, Australia), Apr. 1994.
2. J. T. Foote, "Rapid speaker ID using discrete MMI feature quantisation," in *Proc. Pacific Asian Conference on Expert Systems*, (Singapore), Feb. 1997.
3. S. Pfeiffer, S. Fischer, and W. Effelsberg, "Automatic audio content analysis," Tech. Rep. TR-96-008, University of Mannheim, D-68131 Mannheim, Germany, April 1996. `ftp://pi4.informatik.uni-mannheim.de/-pub/techreports/1996/TR-96-008.ps.gz`.
4. E. Wold, T. Blum, D. Keslar, and J. Wheaton, "Content-based classification, search, and retrieval of audio," *IEEE Multimedia* , pp. 27–36, Fall 1996.
5. T. Blum, D. Keslar, J. Wheaton, and E. Wold, "Audio analysis for content-based retrieval," tech. rep., Muscle Fish LLC, 2550 Ninth St., Suite 207B, Berkeley, CA 94710, USA, May 1996. `http://www.musclefish.com/cbr.html`.
6. L. Wyse and S. Smoliar, "Toward content-based audio indexing and retrieval and a new speaker discrimination technique," in *Proc. ICJAI '95*, (Montreal), 1995.
7. B. Feiten and S. Günzel, "Automatic indexing of a sound database using self-organizing neural nets," *Computer Music Journal* **18**(3), pp. 53–65, 1994.
8. L. R. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
9. J. T. Foote, *Decision-Tree Probability Modeling for HMM Speech Recognition*. Ph.D. thesis, Brown University, Providence, RI, 1993.
10. L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth International Group, Belmont, Calif., 1984.
11. M. Anikst *et al.*, "The SSI large-vocabulary speaker-independent continuous-speech recognition system," in *Proc. ICASSP '91*, pp. 337–340, IEEE, (Toronto, Canada), 1991.
12. G. Salton and C. Buckley, "Term weighting approaches in automatic text retrieval," Tech. Rep. TR87-881, Department of Computer Science, Cornell University, Ithaca, New York 14853-7501, November 1987. `http://cs-tr.cs.cornell.edu/Dienst/UI/2.0/Describe/ncstrl.cornell%2fTR87-881`.
13. W. Frakes and R. Baeza-Yates, *Information Retrieval: data structures and algorithms*, Prentice Hall, New Jersey, 1992.
14. G. Salton and C. Buckley, "TREC evaluation software distribution." `ftp://ftp.cs.cornell.edu/pub/smart`, 1991.
15. L. Wilcox, F. Chen, and V. Balasubramanian, "Segmentation of speech using speaker identification," in *Proc. ICASSP 94*, vol. S1, pp. 161–164, Apr. 1994.
16. D. Kimber and L. Wilcox, "Acoustic segmentation for audio browsers," in *Proc. Interface Conference*, (Sydney, Australia), July 1996. `http://www.fxpal.xerox.com/abstracts/kim96.htm`.
17. D. Roy and C. Malamud, "Speaker identification based text to audio alignment for an audio retrieval system," in *Proc. ICASSP 97*, IEEE, (Munich, Germany), Apr. 1997.
18. J. Saunders, "Real-time discrimination of broadcast speech/music," in *Proc. ICASSP '96*, vol. II, pp. 993–996, IEEE, (Atlanta), May 1996.