SOUND ANALYSIS USING MPEG COMPRESSED AUDIO

George Tzanetakis

Computer Science Department
Princeton University
35 Olden Street, Princeton NJ 08544,USA
gtzan@cs.princeton.edu

Perry Cook

Computer Science and Music Deptartment
Princeton University
35 Olden Street, Princeton NJ 08544,USA
prc@cs.princeton.edu

ABSTRACT

There is a huge amount of audio data available that is compressed using the MPEG audio compression standard. Sound analysis is based on the computation of short time feature vectors that describe the instantaneous spectral content of the sound. An interesting possibility is the calculation of features during the decompression process. Since the bulk of the feature calculation is performed during the encoding stage this process has a significant performance advantage if the available data is compressed. Combining decoding and analysis in one stage is also very important for audio streaming applications.

In this paper, we describe the calculation of features directly from MPEG audio compressed data. Two of the basic processes of analyzing sound are: segmentation and classification. To illustrate the effectiveness of the calculated features we have implemented two case studies: a general audio segmentation algorithm and a Music/Speech classifier . Experimental data is provided to show that the results obtained are comparable with sound analysis algorithms working directly with audio samples.

1. INTRODUCTION

In order to handle the increasing amounts of audio data available, more structure-extracting tools are necessary. Although there has been a lot of work on video analysis, the work on audio analysis has been more limited. Recently, a number of techniques for automatic analysis of audio information have been proposed [1]. All these approaches involve the calculation of short time feature vectors that describe the instantaneous content of the sound.

MPEG audio compression has become the most common way to store audio files (the well-known .mp3 files). In this work we explore the possibility of calculating short time features vectors directly from the compressed data. This calculation can be done at the same time as the audio samples decompression or by itself without having to output audio samples. That way significant time can be saved since the bulk of the feature calculation is performed during the encoding stage. Combining decoding and analysis in one stage is also very important for audio streaming applications. Using traditional analysis techniques that work with the audio samples, the MPEG data would have to be decompressed and then reprocessed for performing the analysis.

Auditory scene analysis is the process by which the auditory system builds mental descriptions of complex auditory environments by analyzing mixtures of sounds [2]. From an ecological viewpoint, we try to associate events with sounds in order to understand our environment. Classification, i.e what made this sound, and segmentation, i.e how long did it last, are two basic processes of auditory analysis. The decisions for sequential and simultaneous integration of sound are based on multiple cues. Although, our feature calculation does not attempt to model the human auditory system, it does use multiple features as a basis for classifying and detecting segmentation boundaries. As an example of segmentation a general audio segmentation system similar to [3] has been implemented. As an example of classification a Music Speech discriminator similar to [4] has been implemented. The experiments indicate that the features selected contain enough information to be useful for automatic classification and segmentation and are comparable with algorithms that operate directly on audio samples.

2. SHORT MPEG OVERVIEW

The MPEG audio coding standard is an example of a perception based coder that exploits the characteristics of the human auditory system to compress audio more efficiently. Since there is no specific source model, like for example in speech compression algorithms, it can be used to compress any type of audio. In the first step the audio signal is converted to spectral components via an analysis filterbank. Each spectral component is quantized and coded with the goal of keeping the quantization noise below the masking threshold. Simultaneous masking is a frequency domain phenomenon where a low level signal (the maskee) can be made inaudible (masked) by a simultaneous occuring stronger signal (the masker) as long as masker and maskee are close enough to each other in frequency. Therefore, the dynamic bit allocation in each subband is controlled by a psychoacoustic model. Layer I,II,III offer different tradeoffs between complexity and compression ratio. More details can be found in [5] and at the complete ISO specification [6] [7].

Like most modern coding schemes there is an assymetry between the encoder and the decoder. The encoder is more complicated, slower, and there is some flexibility in the psychoacoustic model used. On the other hand, decoding is simpler and straightforward. The subband sequences are reconstructed on the basis of blocks taking into account the bit allocation information. Each time the subband samples of all the 32 subbands have been calculated, they are applied to the synthesis filterbank, and 32 consecutive 16-bit, PCM-format audio samples are calculated. The feature calculation described in the following section is performed before the filterbank synthesis which is common to all layers.

2.1. The filterbank

The digital audio input is mapped into 32 subbands via equally spaced bandpass filters. A polyphase filter structure is used for the frequency mapping; its filters have 512 coefficients. The filters are equally spaced in frequency. At a 22050 Hz sampling rate, each band has a width of 11025 / 32 = 345 Hz. The subsampled filter outputs exhibit significant overlap. The impulse response of subband k, $h_k(n)$ is obtained by multiplication of a single prototype low-pass filter, $h(n)$, by a modulating function that shifts the lowpass responce to the appropriate subband frequency range:

$$h_i(n) = h(n)cos(\frac{(2i-1)}{2M} + \phi(i));$$
$$M = 32; i = 1, 2, ..32; n = 1, 2, ...512; \qquad (1)$$

Although the actual calculation of the samples is performed differently for performance reasons; the 32-dimensional subband vector can be written into a convolution equation:

$$S_t[i] = \sum_{n=0}^{n=511} x[t-n] * h_i[n] \qquad (2)$$

where $h_i[n]$ are the individual subband band-pass filter responces. Details about the coefficients of the prototype filter and the phase shifts $\phi(k)$ are given in the ISO/MPEG standard [6], [7].

3. FEATURE CALCULATION

For the experiments MPEG-2, Layer III, 22050Hz sampling rate, mono, files were used. A similar approach can be used with the other layers and sampling rates as well as any filterbank-based perceptual coder. The analysis is performed on blocks of 576 samples (about 20msec at 22050Hz) that correspond to one MPEG audio frame. A root mean squarred subband vector is calculated for the frame as:

$$M[i] = \sqrt{\frac{\sum_{t=1}^{t=18}(S_t[i]^2)}{18}}, \; i = 1..32; \qquad (3)$$

S_t are the 32-dimensional subband vectors. The resulting M is a 32-dimensional vector that describes the spectral content of the sound for that frame. From this vector characteristic features are calculated. The features are:

Centroid is the balancing point of the vector. It can be calculated using

$$C = \frac{\sum_{i=1}^{i=32} iM[i]}{\sum_{i=1}^{i=32} M[i]} \qquad (4)$$

Rolloff is the the value R such that

$$\sum_{i=1}^{R} M[i] = 0.85 \sum_{i=1}^{i=32} M[i] \qquad (5)$$

Spectral Flux is the 2-norm of the difference between normalized M vectors evaluated at two successive frames.

RMS is a measure of the loudness of the frame. It can be calculated as

$$RMS = \sqrt{\frac{\sum_{i=1}^{i=32}(M[i]^2)}{32}} \qquad (6)$$

This feature is unique to segmentation since changes in loudness are important cues for new sound events. In contrast, classification algorithms must be loudness invariant.

The actual features used for the analysis are the mean and variances of those features in a larger window (40 frames about 1sec) of past frames. A circular buffer of the past 40 feature vectors is used for faster performance. In addition to the means and variances of those features, a feature called *LowEnergy* is used. It is defined as the percentage of frames, in a 1sec window, that have less than the average power. This feature is used for Music/Speech discrimination since speech has large variations in energy levels. Finally, log-transforms of the features are used to reduce the dynamic range and make the clusters in classification more compact.

4. SEGMENTATION

In this work segmentation refers to the process of breaking audio into regions based on what could be called "texture" of sound. Some examples are a piano entrance after the orchestra in a concerto, a rock guitar solo, a change of speaker etc. There are no assumptions about the type of audio and no statistical class model of the data is made. For segmenting we follow the methodology described in [3]. The method can be broken into four stages:

1. A time series of feature vectors V_t is calculated by iterating over the sound file.
2. A distance signal $\Delta_t = ||V_t - V_{t-1}||$ is calculated between successive frames of sound. In our implementation we use a Mahalonobis distance. It is defined by

   $$D(x, y) = (x-y)^T \Sigma^{-1}(x-y) \qquad (7)$$

   where Σ is the feature covariance matrix calculated from the whole sound file. This distance rotates and scales the feature space so the contribution of each feature is equal. Other distance metrics, possibly using relative feature weighting, can also be used.
3. The derivative $\frac{d\Delta_t}{dt}$ of the distance signal is taken. The derivative of the distance will be low for slowly changing textures and high during sudden transitions. Therefore the peaks roughly correspond to texture changes.

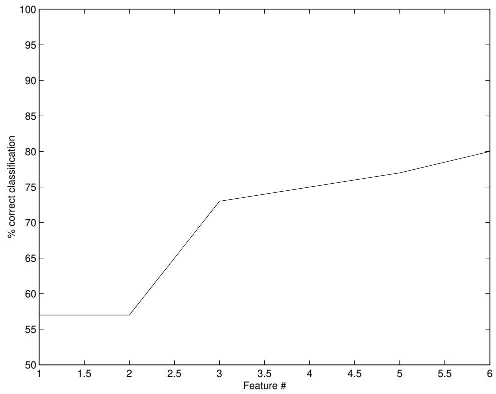

Figure 1: Increase in classification performance as features are added. Gaussian MAP classifier.

4. Peaks are picked using simple heuristics and used to create the segmentation of the signal into time regions. As a heuristic example, adaptive thresholding can be used. A minimum duration between successive peaks can be set to avoid small regions.

The features used for segmentation are *Low Energy* and the mean and variances of *Centroid, Rolloff, Flux, Rms.*

5. CLASSIFICATION

In order to evaluate the proposed features a multifeature statistical Music/Speech classifier has been implemented. It is based on the discriminator described in [4]. Two standard statistical pattern recognition classifiers were used for the experiments. For a more complete description of these classifiers and statistical pattern recognition in general refer to [8].

The Gaussian (MAP) clasifier assumes each class can be represented as a multi-dimensional normal distribution in feature space. A labeled data set is used to train the classifier by calculating the parameters for each particular class. This classifier is typical of parametric statistical classifiers that assume a particular form for the underlying class probability density functions.

Unlike parametric classifiers, the K-Nearest neighbor classifier directly uses the training set for classification without assuming any mathematical form for the underlying class probability density functions. Each sample is classified according to the class of its nearest neighbor in the training data set. In K-NN, the K nearest-neighbors are calculated and voting is used to determine the class. More compicated classifiers like Gaussian Mixture and Neural Network classifiers can be used to improve the performance of the system. However, it is a known result that no matter what classifier we use, we can never do better than to cut the error rate in half over the nearest-neighbor classifier, assuming the training and testing data represent the underlying feature space topology. The features used for classification are *Low Energy* and the mean and variances of *Centroid, Rolloff* and *Flux.*

	GMAP	K-NN(1)	K-NN(5)
MPEG	$82.2 \pm 2.1\%$	$84.6 \pm 4.4\%$	$86.2 \pm 3.8\%$
AUDIO	$84.8 \pm 1.4\%$	$89.5 \pm 2.0\%$	$90.0 \pm 1.2\%$
Different Data Set			
OTHER1	$94.0 \pm 2.6\%$	$94.2 \pm 3.6\%$	$95.7 \pm 3.5\%$
Different Data Set & Classifiers			
OTHER2	77.1%	90.0%	94.2%

Table 1: Music/Speech percentages of frame-based classification performance

	Free	4 ± 1	8 ± 2
MPEG FB	55%, 31/56	58%, 25/43	54%, 38/70
AUDIO FB	62%, 35/56	51%, 22/43	62%, 44/70
MPEG BE	71%, 40/56	74%, 32/43	65%, 46/70
AUDIO BE	85%, 48/56	86%, 37/43	75%, 53/70

Table 2: Comparison of segmentation algorithms

6. EVALUATION

6.1. Classfication Evaluation

The data used for evaluating the Music/Speech classification consists of about 2 hours of audio data. There are 45 minutes of speech, 45 minutes of music, and about 30 minutes of mixed audio. Radio, live recordings of speech, compact disks and movies representing a variety of speakers and music styles were used as data sources.

To ensure that the results would not be affected by the choise of training/testing set, a robust evaluation was done using a random 10% of the data for testing and 90% for training. The process was repeated 100 times and the mean and the standard deviation of the classification accuracy were collected. In addition, soundfiles were not split into training and testing data. This is important since there is a good deal of frame-to-frame correlation in vectors of the same file so splitting the file would give an incorrect estimate of classifier performance for truly novel data. The results are calculated on a frame basis without any integration. The regions detected by the segmentation algorithm can be used for integration using a malority filter to achieve better classification performance.

Further improvements in classification performance can be achieved by integrating the results. For integration the region detected by the segmentation algorithm can be used.

Figure 1, shows the improvement in classification performance by the gradual addition of features. The order used is *mean Centroid,mean Rolloff, mean Flux, variance Centroid, variance Rolloff, variance Flux and Low Energy.* Table 5 compared the performance of the MPEG based classification with audio sample based systems. For the AUDIO line the same dataset and a similar set of features calculated using short time Fourier Transform were used. The OTHER1 line is based on results by [4] and uses a different data set. The OTHER2 line is based on results by [9] and uses a different data set and different classifiers. The results indicate that the proposed features can be used for classification without significant decrease in accuracy.

6.2. Segmentation

The data used to evaluate segmentaion consists of 10 sound files about 1 minute long. A variety of music styles,textures,speech and singing are represented. The comparison is based on the user experiments for audio segmentation described in [3]. In those experiments, nine subjects were asked to segment each sound file using standard audio editing tools in 3 ways. The first way, called free, is breaking up the file into any number of segments. The second and third way constrain the users to a specific budget of total segments 4 ± -1 and 8 ± 2. The segments that more than 4 of the 9 subjects agreed upon were retained for comparison with the automatic segmentation algorithm. In table 6.1 the performance of the MPEG-based segmentation algorithm is compared with a similar algorithm based on short time Fourier Transform analysis. The table shows the number of regions that were marked by humans that were captured by the system.FB (fixed-budget) refers to automatic segmentation by requesting the same number of segments as the salient human segments. BE (best effort) refers to the best automatic segmentation achieved by incrementally increasing the number of regions up to a maximum of 16. Although the performance is lower than the audio-based algorithm still a large number of segmentation boundaries are captured by the algorithm. Most of the cases where the algorithm missed boundaries compared to the audio based algorithm were in soundfiles containing rock or jazz music. The reason is that the audio based algorithm uses time domain zerocrossings that capture the transition from noise-like signals to more harmonic signals.

7. IMPLEMENTATION

For the feature calculation an MP3 decoder was implemented in C++. The source code is loosely based on the Fraunhaufer institute reference software implementation and other open source implementations. The feature calculation, classification and segmentation componenents were integrated using MARSYAS [10] an object-oriented framework for building audio analysis tools written in C++ and JAVA. The combined decoding and classification/segmentation runs real-time on a Pentium II PC. The source code for the feature calculation is available upon request.

8. FUTURE WORK

For statistical pattern recognition large amounts of training data need to be collected. By doing the analysis on MPEG compressed data the enormous resources available on the Web can become available for training. We are planning to implement a Web crawler to automatically gather .mp3 files from the Web for this purpose. Other feature-based statistical pattern recognition applications that could use a similar front-end are: speaker identification, male/female discrimination, content based retrieval and instrument classification. It might even be possible to use MPEG based features for speech recognition directly on compressed data. The features described in this work are not the only ones possible and further investigation is needed in order to come up with better features tailored to specific applications. In addition to the subband analysis other types of information can be utilized. For example in MPEG layer III a window switching scheme is used where the 32 subband signals are further subdivided in frequency content by applying, to each subband, a 6-point or 18-point modified DCT block transform. The 18-point block transform is normally applied because it provides better frequency resolution, whereas the 6-point block transform provides better time resolution. Therefore it is more likely to find semgentation boundaries in short blocks and this information could be used to enhance the segmentation algorithm.

9. SUMMARY

A short time feature calculation scheme that operates directly on MPEG audio compressed data during the decompression stage is described. The resulting features have been used successfully for automatic Music/Speech classification and general audio segmentation. The decoding and feature calculation are performed real-time since a large part of the feature calculation has already been done during the encoding stage.

10. REFERENCES

[1] J. Foote, “An overview of audio information retrieval,” *ACM Multimedia Systems*, vol. 7, pp. 2–10, 1999.

[2] A. Bregman, *Auditory Scene Analysis*, MIT Press, 1990.

[3] G. Tzanetakis and P. Cook, “Multifeature audio segmentation for browsing and annotation,” in *Proc.1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA99*, New Paltz, NY, 1999.

[4] E. Scheirer and M. Slaney, “Construction and evaluation of a robust multifeature speech/music discriminator,” *IEEE Transactions on Acoustics, Speech and Signal Processing (ICASSP'97)*, pp. 1331–1334, 1997.

[5] Noll Peter, “Mpeg digital audio coding,” *IEEE Signal Processing Magazine*, pp. 59–81, September 1997.

[6] ISO/IEC JTC1/SC29, *Information Technology-Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s-IS 11172 (Part 3, Audio)*, 1992.

[7] ISO/IEC JTC1/SC29, *Information Technology-Generic Coding of Moving Pictures and Associated Audio Information-IS 13818 (Part 3, Audio)*, 1994.

[8] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.

[9] S. Rossignol, X. Rodet, et al., “Features extraction and temporal segmentation of acoustic signals,” *Proc. ICMC 98*, pp. 199–202, 1998.

[10] G. Tzanetakis and P. Cook, “A framework for audio analysis based on classification and temporal segmentation,” in *Proc.25th Euromicro Conference. Workshop on Music Technology and Audio Processing*, Milan, Italy, 1999, IEEE Computer Society.