Audio Retrieval by Rhythmic Similarity

Jonathan Foote
FX Palo Alto Laboratory, Inc.
3400 Hillview Ave.
Building 4
Palo Alto, CA 94304 USA
foote@fxpal.com

Matthew Cooper
FX Palo Alto Laboratory, Inc.
3400 Hillview Ave.
Building 4
Palo Alto, CA 94304 USA
cooper@fxpal.com

Unjung Nam
CCRMA
Department of Music
Stanford University
Stanford, CA 94305 USA
unjung@stanford.edu

ABSTRACT

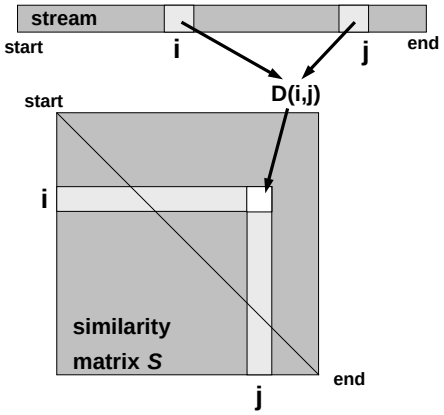
We present a methods for characterizing both the rhythm and tempo of music. We also presentways to quantitatively measure the rhythmic similarity between two or more works of music. This allows rhythmically similar works to be retrieved from a large collection. A related application is to sequence music by rhythmic similarity, thus providing an automatic "disc jockey" function for musical libraries. Besides specific analysis and retrieval methods, we present small-scale experiments that demonstrate ranking and retrieving musical audio by rhythmic similarity.

1. INTRODUCTION

Recently, many computer users are amassing increasingly large numbers of music files. The advent of compressed formats and peer-to-peer file sharing services allows even casual users to build substantial digital music collections. An informal poll on the "Slashdot" website (http://www.slashdot.org) indicated that 24% of nearly 70,000 respondents had collected more than nine gigabytes of MP3 format audio. At typical compression ratios, this corresponds to roughly 150 hours of music, or several thousand popular songs. While song retrieval by metadata (artist, song title, album title) is well supported by current technologies, content-based retrieval is not. We hypothesize that users would like to rank music by rhythmic similarity for browsing and searching, and for sequencing music played in the background. This functionality is not well-supported by existing metadata; while there is often some notion of "genre," it is clear that there can be a wide variations in the tempo or "feeling" of music even in the same genre. For example, Amazon.com places recordings by Serge Gainsbourg, Cheap Trick, and Peaches & Herb in the same "pop rock" category.

We present audio analysis algorithms that can automatically rank music by rhythmic and tempo similarity. Our assumption is that the feeling or mood of a musical work is highly correlated with tempo and rhythm, and that users will find value in systems that can organize existing music collections or discover new music based on similarity. It is hypothesized that a music vendor would find value in a "find me more music like this" service: even if it yields results no better than random, users would likely listen to, and perhaps purchase, music they would not have encountered otherwise.

Music in a user's collection is analyzed using the "beat spectrum," a novel method of automatically characterizing the rhythm and tempo of musical recordings [1]. The beat spectrum is a measure of acoustic self-similarity as a function of time lag. Highly structured or repetitive music will have strong beat spectrum peaks at the repetition times. This reveals both tempo and the relative strength of particular beats, and therefore can distinguish between different kinds of rhythms at the same tempo. Unlike previous approaches to tempo analysis, the beat spectrum does not depend on particular attributes such as energy, pitch, or spectral features, and thus will work for any music or audio in any genre. In particular, the method is still robust (if not very informative) for audio with little or no rhythmic characteristics.



Similarity matrix calculation

The beat spectrum is calculated for every music file in the user's collection. The result is a collection of "rhythmic signatures" for each file. We present methods of measuring the similarity between beat spectra, and thus between the original audio. Given a similarity measure, files can be ranked by similarity to one or more selected query files, or by similarity with any other musical source from which a beat spectrum can be measured. This allows users to search their music collections by rhythmic similarity, as well an enable novel applications. For example, given a collection of files, an application could sequence them by rhythmic similarity, thus functioning as an "automatic DJ."

2. RELATED WORK

Many researchers have made contributions to tempo tracking. Influential early research was done by Dannenberg and Mont-Reynaud [3]. In this work, primarily intended for real-time use, a "confidence" score of a beat occurrence is updated from MIDI note-on events. No audio analysis is performed.

Several approaches exist for estimating the tempo of recorded music. Canonical work by Eric Schierer is described in [4], where

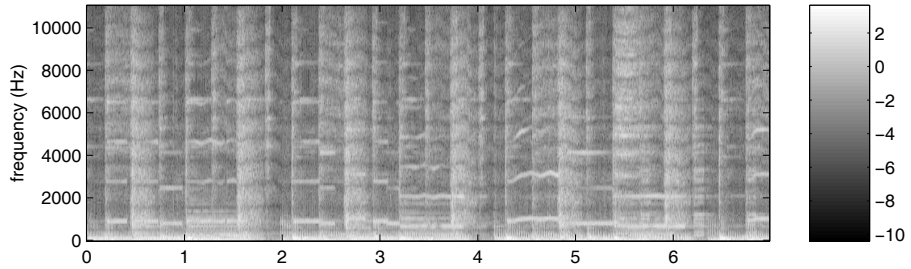


Figure 1. Spectrogram of "Musica Si" excerpt

energy peaks across frequency sub-bands are detected and correlated. This approach will work best on music with a strong percussive element, that is, short-term periodic wideband sources such as drums. Another approach is designed for music in 4/4 time with a bass drum on the downbeat [7]. These systems universally attempt to measure one dominant tempo, and are thus not robust to "beat doubling" effects, where the tempo is misjudged by a factor of two, or confused by energy peaks that do not occur in tempo or are insufficiently strong. Typically this is constrained by a number of ad-hoc methods that include averaging over many beats, rejecting out-of-band results, or Kalman filtering as in [6].

Work done at Musclefish, Inc. computes rhythmic similarity for a system for searching a library of rhythm loops [8]. Here, a "bass loudness time-series" is generated by weighting the short-time Fourier transform (STFT) of the audio waveform. A peak in the power spectrum of this time series is chosen as the "fundamental" period. The Fourier result is normalized and quantized into durations of 1/6 of a beat, so that both duplet and triplet subdivisions can be represented. This serves as a feature vector for tempo-invariant rhythmic similarity comparison. This approach works for drum-only tracks (the application it was designed for) but is likely to be less robust on music with significant low frequency energy not due to drums.

An interesting system has been proposed by Dave Cliff of the HP Research in Bristol, UK. This system is intended to serve as an "Automatic DJ" and covers both track selection and cross-fading [9]. The system is designed for the relatively narrow genre of dance music, where the tempo of musical works is relatively simple to detect because of its repetitive and percussive nature, and is usually constant across a work. Cliff's system for track selection is based on a tempo "trajectory," or a function of tempo versus time. This is quantized into time "slots" based on the number of works available. Both slots and works are then ranked by tempo, and assigned in a 1:1 fashion -- for example, the second highest slot gets the track with the second fastest tempo. Absolute tempos are not considered, but this is not a serious drawback as dance music is generally confined to a limited range of acceptable tempos.

Recent work at Princeton has resulted in a rhythmic characterization called the "beat histogram." Here, an autocorrelation is performed on the amplitudes of wavelet-like features, across multiple windows so that many results are available. Major peaks in each autocorrelation are detected and accumulated in a histogram. The lag time of each bin is inverted to yield a tempo (bpm) axis for the histogram. The result is a measure of periodicity versus tempo. For genre classification, features are derived from the beat histogram including the tempo of the major peaks and amplitude rations between them [5]. This approach is similar to the "beat spectrum" presented here, in that both attempt to represent salient periodicities versus lag time (the beat spectrum) or, equivalently, tempo (the beat histogram). Our approach differs in that we compare the beat spectra directly, without relying on peak-picking or related features which may be less than robust.

3. BEAT SPECTRUM CALCULATION

Details of the beat-spectral analysis are presented in [1]; we include a short version here for completeness. The beat spectrum is calculated from the audio using three principal steps. First, the audio is parametrized into using a spectral or other representation. This results in a sequence of feature vectors. Second, a distance measure is used to find the similarity between all pairwise combinations of feature vectors, hence times in the audio. This is embedded into a two-dimensional representation called a similarity matrix. The beat spectrum results from finding periodicities in the similarity matrix, using diagonal sums or autocorrelation. The following sections present each step in more detail.

3.1 Audio parameterization

The methods presented here are all based on the distance matrix, which is a two-dimensional embedding of the audio self-similarity. The first step is to parameterize the audio. This is typically done by windowing the audio waveform. Various window widths and overlaps can be used; in the present system windows ("frames") are 256 samples wide, and are overlapped by 128 points. For audio sampled at 16kHz, this results in a 16 mS frame width and a frame rate of 125 per second. A fast Fourier transform is performed on each frame, and the logarithm of the magnitude of the result estimates the power spectrum. The result is a compact vector of parameters that characterizes the spectral content of the frame. Many compression techniques such as MPEG-II Layer 3 use a similar spectral representation, which could be used directly in for a distance measure. This would avoid the cost of decoding the audio and reparameterizing, as in [12]. Note that the actual parameterization is not crucial as long as "similar" sounds yield similar parameters. Other parameterizations could be used, including those based on linear prediction, Mel-Frequency Cepstral Coefficient (MFCC) analysis, or psychoacoustic considerations.

3.2 Calculating frame similarity

Once the audio has been parameterized, it is then embedded in a 2-dimensional representation. A (dis)similarity measure $D(i,j)$ between feature vectors is calculated from audio frames i and j. A simple measure is the Euclidean distance in the parameter space. Another useful metric is the scalar (dot) product of the vectors. This will be large if the vectors are both large and similarly oriented. To remove the dependence on magnitude (and hence energy,

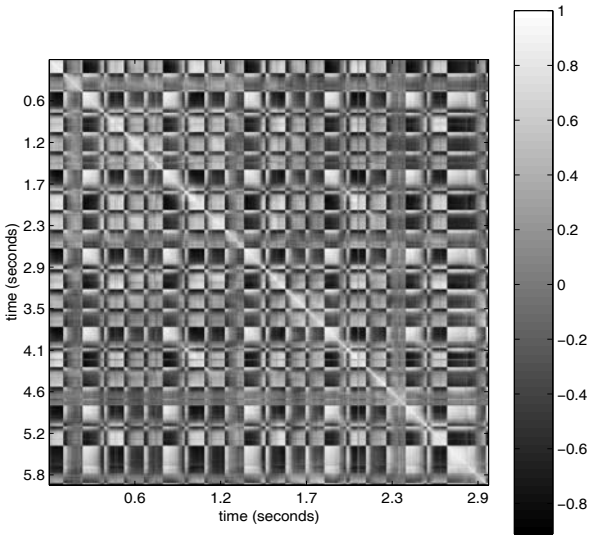


Figure 2. Distance matrix visualization for "Musica Si" theme

given our features), the product can be normalized to give the cosine of the angle between the parameter vectors. The cosine measure ensures that windows with low energy, such as those containing silence, can still yield a large similarity score, which is generally desirable. This is the distance measure used here.

3.3 Distance Matrix Embedding

A distance matrix conveniently represents the similarity between all possible instants in a signal. This is done by *embedding* the distance measure in a two-dimensional representation, as shown in the figure on the front page. The matrix $\mathbf{S}$ contains the similarity measure calculated for all frame combinations, hence time indexes i and j such that the i,jth element of $\mathbf{S}$ is $D(i,j)$. In general, $\mathbf{S}$ will have maximum values on the diagonal (because every window will be maximally similar to itself); furthermore if D is symmetric then $\mathbf{S}$ will be symmetric as well.

$\mathbf{S}$ can be visualized as a square image such that each pixel i, j is given a gray scale value proportional to the similarity measure $D(i,j)$, and scaled such that the maximum value is given the maximum brightness. The resulting image provides a visualization of the audio structure. Regions of high self-similarity appear as bright squares on the diagonal. Repeated sections will be visible as bright off-diagonal rectangles. If the work has a high degree of repetition, this will be visible as diagonal stripes or checkerboards, offset from the main diagonal by the repetition time.The diagonal line at $i = j$ indicates that each frame is maximally similar to itself. Figure 2 shows an example similarity matrix derived from the spectrogram of Figure 1. Note that the periodicity visible in the spectrogram (slightly greater than one second) is also visible in the similarity matrix. More details about the distance matrix embedding can be found in [1].

To simplify computation, the similarity can be represented in the "lag" domain $L(i, j)$ where the lag $l = j - i$. This is particularly helpful here, as the similarity is not needed for all combinations of *i* and *j*, only those within a few seconds of each other (thus small *l*). This reduces the algorithmic complexity from $\mathbf{O}(n^2)$ for a full similarity matrix to a much more manageable $\mathbf{O}(n)$, and in practice the beat spectrum may be computed several times faster than real-time.

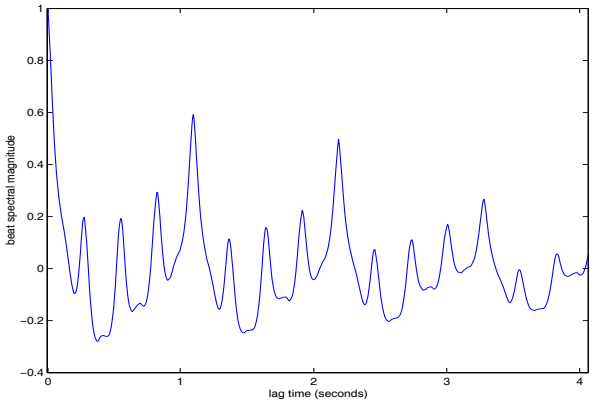


Figure 3. Beat spectrum of "Musica Si" example. Note peak at periodicity slightly greater than one second.

3.4 Deriving the beat spectrum

Both the periodicity and relative strength of musical beats can be derived from the similarity matrix. We call a measure of self- similarity as a function of the lag the *beat spectrum B(l).* Peaks in the beat spectrum correspond to repetitions in the audio. A simple estimate of the beat spectrum can be found by summing $\mathbf{S}$ along the diagonal as follows:

$$B(l) \approx \sum_{k \subset R} \mathbf{S}(k, k + l)$$

Here, $B(0)$ is simply the sum along the main diagonal over some continuous range R, $B(1)$ is the sum along the first superdiagonal, and so forth. An example of the beat spectra for different tempos of music is shown in Figure 4. Music with 120 beats per minute (bpm[1]) should have a strong beat spectral peak at a lag of 0.5 s, as indicated in the figure. A more robust estimate of the beat spectrum comes from the autocorrelation of $\mathbf{S}$:

$$B(k, l) = \sum_{i,j} \mathbf{S}(i, j)\mathbf{S}(i + k, j + l)$$

Because $B(k,l)$ is symmetrical, it is only necessary to sum over one variable, giving the one-dimensional result $B(l)$. This approach has been shown to work well across a range of musical genres, tempos, and rhythmic structures [1].

4. MEASURING RHYTHMIC SIMILARITY

We present methods to determine the similarity between beat spectra computed from different musical works. Given two works, we can compute two beat spectra $B_1(l)$ and $B_2(l)$; both are 1-dimensional functions of lag time *l*. In practice, *l* is discrete and finite, so an obvious approach is to truncate the beat spectra to some number L of discrete values. This yields L-dimensional vectors, from which the Euclidean or other distance functions can be computed. Though there are many possible distance measures, it is not obvious that any will be at all correlated with perceptual differences. Thus it will be important to show that small "distances" correspond to rhythmically similar music, and that larger distances are correlated with decreasing rhythmic similarity. The following section presents small-scale experiments to demonstrate this.

[1]In musical scores, beats per minute is often denoted "MM" for Mälzel's Metronome, after the eponymous inventor of the clockwork metronome.

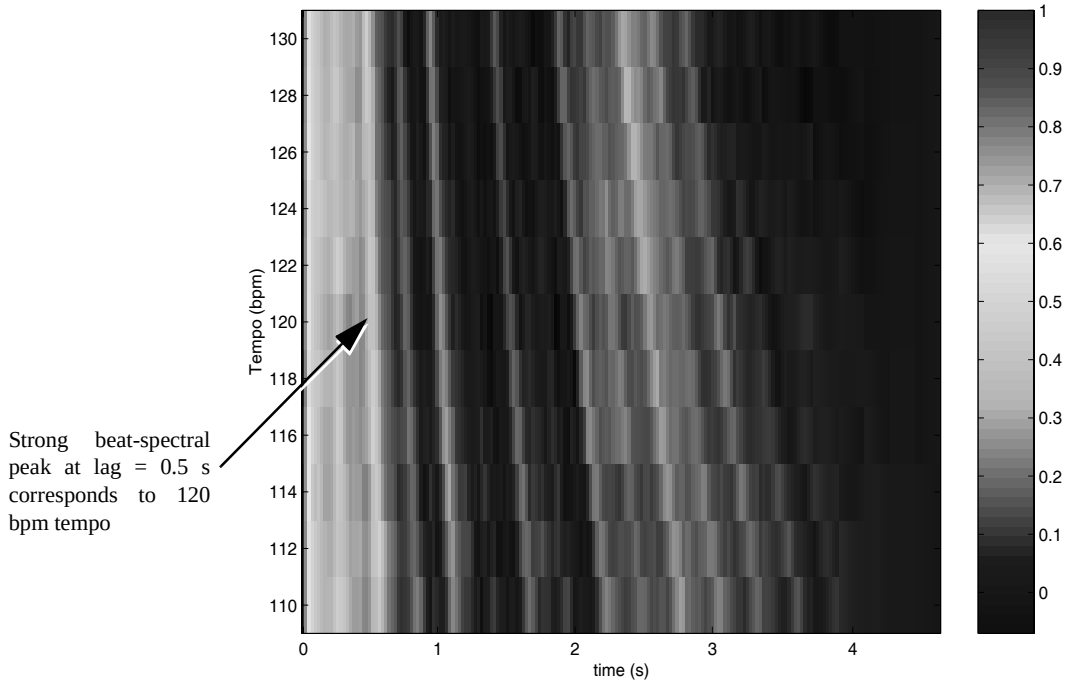


Figure 4. Beat spectra of similar music at different tempos

4.1 Experiment 1:

In this experiment, we determine how well Euclidean distance between beat spectra measures tempo difference. To isolate the effect of tempo on the measurement, we generated different-tempo versions of the identical musical excerpt ("Tangerine" by Apostrophe Ess, © Sean Householder 2001). This is easily done using commercially available music editing software, which can change the duration of a musical waveform without altering the pitch. This musical excerpt consists of 16 4/4 bars of live vocals and instrumentals over a rhythmic beat, and is thus far more realistic than a synthesized MIDI realization. The original excerpt was played at 120 beats per minute (bpm; also denoted MM). Ten tempo variations were generated at 2 bpm intervals from 110 to 130 bpm. Thus the test corpus consists of 11 musical excerpts identical save for the tempo. It should be noted that a two bpm tempo difference is rather subtle, and may not be perceptible to many listeners (a fact we have exploited for a watermarking scheme). This audio data can be found at: http://www.fxpal.com/people/foote/musicr/tempor.html

A first test of measuring beat spectral difference is a simple Euclidean distance between beat spectra. To this end, beat spectra were computed for each excerpt, and the squared Euclidean distance computed for all pairwise combinations. Figure 5 shows the result. Each line shows the Euclidean distance between one source excerpt and all other files. The source file is easily identified as the tempo where each line has a distance of zero. This graphically demonstrates that the Euclidean distance increases relatively monotonically for increasing tempo differences. This indicates that the Euclidean distance can be used to rank music by tempo. Of course, this is a highly artificial case in that examples of the same music at different tempos are relatively rare. Still, it serves as a "sanity check" that the beat spectrum does in fact capture useful rhythmic information. Our next experiments examine beat spectral similarity across different kinds of music.

4.2 Experiment 2

The corpus for this experiment are 10-second excerpts taken from the audio track of "Musica Si," a pop-music variety show produced by RTVE (Spain), and available as item V22 of the MPEG-7 Content Set [2]. This is a good source for these experiments as it both contains a variety of popular musical styles, and has been released under a copyright that allows for research use. Excerpts were 10 seconds long and are labeled with the start time from the beginning of the video. (Excerpt 15, taken five seconds into the start of the theme music, is the source for Figures 1, 2, and 3.) Table 2 summarizes the data excerpted from the soundtrack (which again, can be found at http://www.fxpal.com/people/foote/musicr/tempor.html). There were four songs that were long enough to extract multiple ten-second samples. Each song is represented by three ten-second excerpts, save for a pop/rock song whose chorus and verse are each represented by three excerpts respectively. Although judging relevance for musical purposes is generally a complex and subjective task, in this case it was fairly straightforward: each excerpt was assumed to be relevant to other excerpts from the same tune, and not relevant to all other excerpts. The one exception is that the verse and chorus of the pop/rock song were markedly different in rhythm and so are assumed to not be relevant to each other. Thus we have three ten-second excerpts from each of five relevance classes (three songs plus two song sections), for a total of 15 excerpts.

The raw beat spectra were first processed in the following manner. Each was normalized by scaling so the peak magnitude (at zero lag) was unity. Next, the mean was subtracted from each vector. Finally the beat spectra were truncated in time. Because the short-lag spectra is similar across all files and thus not informative, the first 116 ms was truncated. Also lags longer than 4.75 s were also truncated. The result was a zero-mean vector having a length of 200 values, representing lags from 116 ms to 4.75 s for each musi-

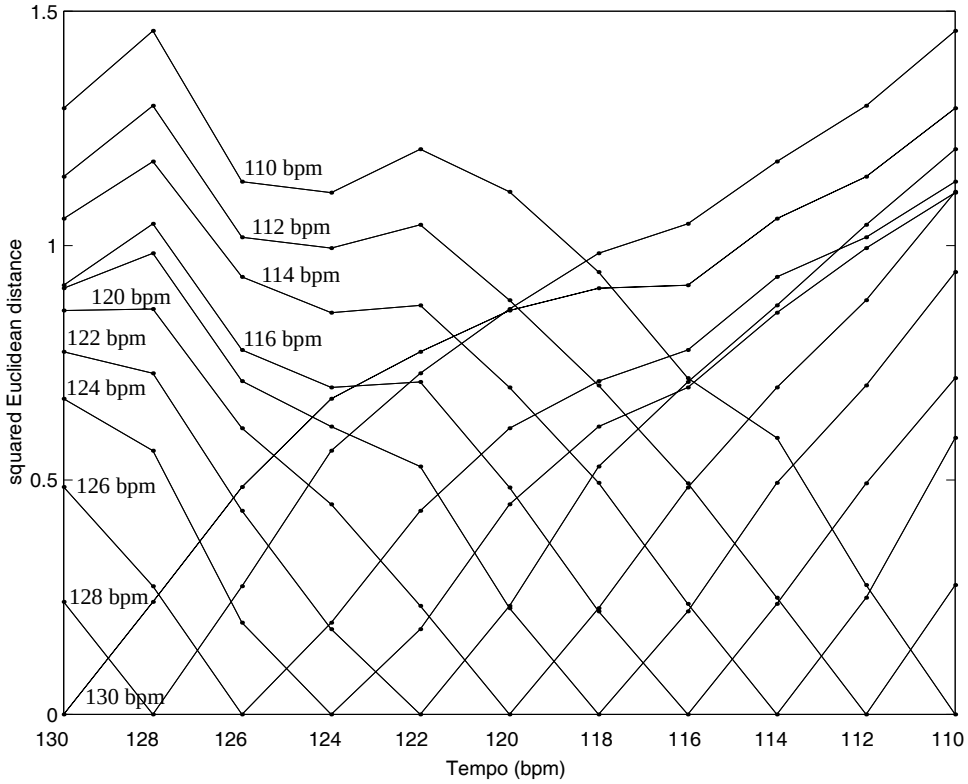


Figure 5. Euclidean Distance vs. Tempo

cal excerpt. (The effect of varying the truncated regions was not examined, and it is not unlikely that other values may result in better retrieval performance.)

4.1.1 Euclidean Distance

Three different distance measures were used. The first was straightforward squared Euclidean distance measure, or the sum of the squares of the element-by-element differences between the values, as used in Experiment 1. For evaluation, each excerpt was used as a query. Each of the 15 corpus documents was then ranked by similarity to each of the 15 queries using the squared Euclidean distance. (For the purposes of ranking, the squared distance serves as well as the distance, as the square root function is monotonic.) Each query had 2 relevant documents in the corpus, so this was chosen as the cutoff point for measuring retrieval precision. Thus there were 30 relevant documents for this query set. For each query, documents were ranked by increasing Euclidean distance from the query. Using this measure, 24 of the 30 possible documents were relevant (i.e. from the same relevance class), giving a retrieval precision of 80%. (More sophisticated analyses such as ROC curves, are probably not warranted due to the small corpus size.)

4.1.2 Cosine Distance

The second measure used is a cosine metric, similar to that described in the previous section. This distance measure may be preferable because it is less sensitive to the actual magnitudes of the vectors involved. This measure proved to perform significantly better than the Euclidean distance. Using this measure, 29 of the 30 documents retrieved were relevant, giving a retrieval precision of 96.7% at this cutoff.

4.1.3 Fourier Beat Spectral Coefficients

The final distance measure is based on the Fourier coefficients of the beat spectrum, because they can represent the rough spectral shape with many fewer parameters. A more compact representation is valuable for a number of reasons: for example, fewer elements speeds distance comparisons and also reduces the amount of data that must be stored to represent each file. To this effect, the fast Fourier transform was computed for each beat spectral vector. The log of the magnitude was then determined, and the mean subtracted from each coefficient. Because high “frequencies” in the beat spectra are not rhythmically significant, the transform results were truncated to the 25 lowest coefficients. Additionally the zeroth coefficient was ignored, as the DC component is insignificant for zero-mean data. The cosine distance metric was computed for the 24 zero-mean Fourier coefficients, which served as the final distance metric. Experimentally, this measure performed identically to the cosine metric, yielding 29 of 30 relevant documents or 96.7% precision. Note that this performance was achieved using an order of magnitude fewer parameters.

Though this corpus is admittedly very small, there is no reason that the methods presented here could not be scaled to thousands or even millions of works. Computing the beast spectrum is computationally quite reasonable and can be done several times faster than real time, and even more rapidly if spectral parameters can be derived directly from MP3 compressed data as in [12] and [13]. Additionally, well-known database organization methods can dra-

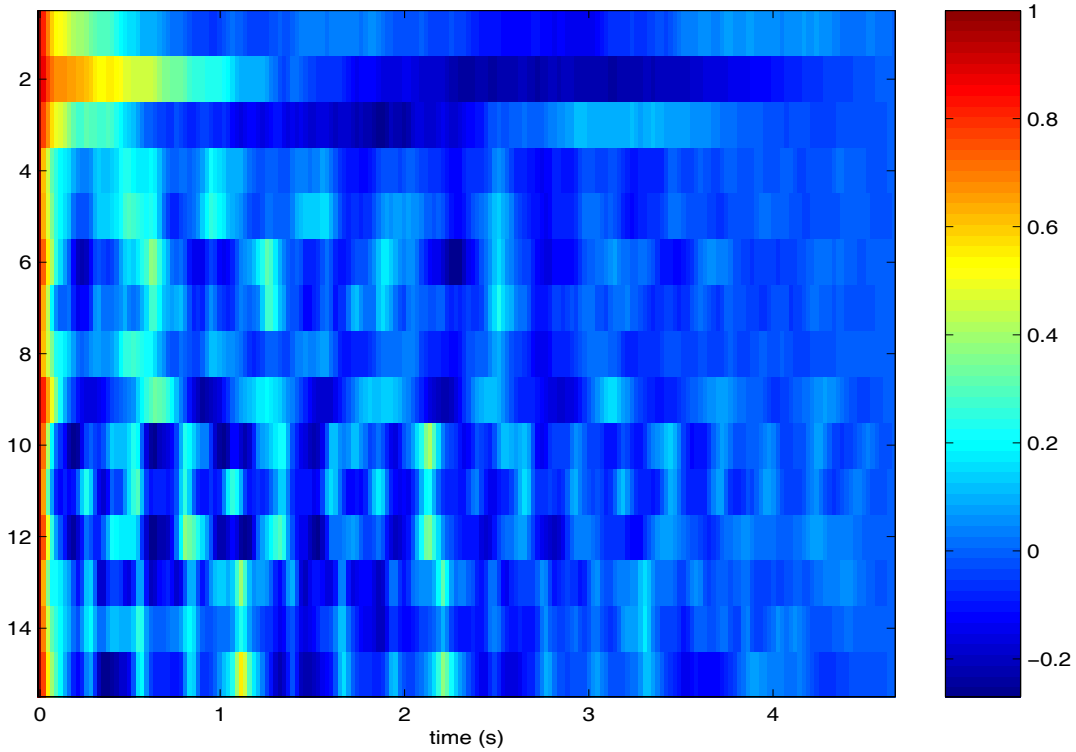


Figure 6. Beat spectra of retrieval data set (see Table 1). Excerpt number 15 (bottom row) is the example of Figure 3.

Table 1. Retrieval data set: 10-second excerpts from "Musica Si" video [2]

Index	Time (mm:ss)	Song Title (approximate)	Description	Relevance Set
1	09:12	"Toto Para Me"	acoustic guitar + vocals	A
2	09:02	"Toto Para Me"	acoustic guitar + vocals	A
3	08:52	"Toto Para Me"	acoustic guitar + vocals	A
4	07:26	"Never Loved You Anyway"	pop/rock chorus	B
5	06:33	"Never Loved You Anyway"	pop/rock chorus	B
6	06:02	"Never Loved You Anyway"	pop/rock verse	C
7	05:52	"Never Loved You Anyway"	pop/rock verse	C
8	05:30	"Never Loved You Anyway"	pop/rock chorus	B
9	04:53	"Never Loved You Anyway"	pop/rock verse	C
10	01:39	"Everybody Dance Now"	dance + rap vocals	D
11	01:29	"Everybody Dance Now"	dance + rap vocals	D
12	01:19	"Everybody Dance Now"	dance + vocals	D
13	00:25	"Musica Si Theme"	theme + vocals	E
14	00:15	"Musica Si Theme"	theme + vocals	E
15	00:05	"Musica Si Theme"	theme intro	E

matically reduce the search time. In particular, high-dimensional indexing techniques can hierarchically cluster beat spectral coefficients so that the search cost increases only logarithmically with the number of documents.

5. ENHANCEMENTS TO THE ALGORITHM

These similarity measures could be extended in several ways. For example, it might be desirable to search for music with similar rhythmic structure but differing tempos. In this case, the beat spectra could be normalized by scaling the lag time. One method might be to scale the lag axis of all beat spectra so that the largest peaks coincide. Using the above distance measures on the scaled spectra would find rhythmically similar music regardless of the tempo.

Because the beat spectra and its corresponding Fourier coefficients inhabit a vector space, many common classification and machine-learning techniques can be used, including both supervised and unsupervised methods. For example, given example classes of music, a statistical classifier can be constructed that might categorize unknown music into the given classes or genres, as in [5]. Example classification methods include linear discriminant functions, Mahalonobis distances, Gaussian mixture models, or non-parametric methods like K-nearest neighbors. Unsupervised clustering could automatically determine genre or other classifications.

6. OPTIMAL MUSIC SEQUENCING

Given a measure of rhythmic similarity, a related problem is to sequence a number of music files in order to maximize the similarity between adjacent files. This allows for smoother 'segues' between music files, and has several applications. If the user has selected a number of files to put on a CD or recording media of limited duration. then the files can be arranged by rhythmic similarity. For example, one method is to create a 'template' of works with a particular rhythm and sequence, for example slow-moderate-fast (The commercial Muzak™ service is known to vary the tempo of its music in 15-minute cycles, as this has been shown to improve worker productivity [10].) Given a template, an algorithm can automatically sequence a larger collection of music according to similarity to the template, possibly with a random element so that the sequence is unlikely to repeat exactly.

A particular application of this paper is to automatically sequence a selected number of musical works. We hypothesize that a satisfying sequence of arbitrary music can be achieved by minimizing the beat-spectral difference between successive songs. This ensures that song transitions are not jarring, for example following a particularly slow or melancholic song with a rapid or energetic one. In this application, two beat spectra are computed for each work, one near the beginning of the work and one near the end. The goodness of a particular transition can be inferred from the beat spectral distance between the ending segment of the first work and the starting segment of the second. Given N works, we can construct a distance matrix whose i,j^{th} entry is the beat spectral distance between the end of work i and the start of work j. Note that this distance matrix is not symmetrical, because in general the distance between end of work i and the start of work j is not identical to the distance between work j's start and work i's end.

The task is now to order the selected songs such that the sum of the intersong distances is a minimum. In matrix formulation, we wish to find the permutation of the distance matrix that will minimize the sum of the superdiagonal. Though this is effectively the Travelling Salesman problem, A greedy algorithm will work to find a reasonable sequence. Variations on this method include constraints such as the sequence must start or end with a particular work.

7. CONCLUSION

We have presented an approach to finding rhythmically similar music and audio.In contrast to other approaches, the beat spectrum does not depend on assumptions such as silence, periodic peaks, or particular time signatures in the source audio. Because it is based on self-similarity, all that is necessary to detect rhythm is repetitive events (even silence) in the audio. (In fact, we expect these methods to work for non-music audio such as speech or industrial sounds if there were an application requiring rhythmic analysis.) Practical applications include an "automatic DJ" for personal music collections, and we are currently prototyping such a system. We are also investigating how well these methods scale to larger collections of hundreds or thousands of songs. This system could usefully be combined with other systems that retrieve music by pitch or timbral similarity, such as [12]. Such a hybrid retrieval engine might allow users to trade off spectral and rhythmic similarity to suit their particular information needs.

8. ACKNOWLEDGEMENTS

Thanks to Sean Householder for the music of Experiment 1. Jorge Licea and Sean Householder also assisted with music title identification.

9. REFERENCES

[1] Foote, J. and Uchihashi, S. "The Beat Spectrum: A New Approach to Rhythm Analysis," in *Proc. International Conference on Multimedia and Expo* 2001. http://www.fxpal.com/people/foote/papers/ICME2001.htm

[2] MPEG Requirements Group. "Description of MPEG-7 Content Set," Doc. ISO/MPEG N2467, MPEG Atlantic City Meeting, 1998. (http://www.darmstadt.gmd.de/mobile/MPEG7/Documents/N2467.html)

[3] Roger B. Dannenberg and Bernard Mont-Reynaud. "Following an Improvisation in Real Time," in *Proceedings of the 1987 International Computer Music Conference*, pp.241-248 (1987)

[4] Scheirer, E., "Tempo and Beat Analysis of Acoustic Musical Signals," in *J. Acoust. Soc. Am.* **103**(1), Jan. 1998, pp 588-601.

[5] G. Tzanetakis, P. Cook, "Automatic Musical Genre Classification of Audio Signals," in *Proc.International Symposium for Audio Information Retrieval* (ISMIR 2001)

[6] Cemgil A.T., Kappen, B. Desain, P. and Honing, H. "On Tempo Tracking: Tempogram Representation and Kalman Filtering." In P*roceedings of 2000 International Computer Music Conference*. pp. 352-355. September 2000.

[7] Goto, M., and Muraoka, Y., "A Beat Tracking System for Acoustic Signals of Music," in *Proc. ACM Multimedia 1994*, San Francisco, ACM.

[8] Wold, E., Blum, T., Keislar, D., and Wheaton, J., "Classification, Search and Retrieval of Audio," in *Handbook of Multimedia Computing*, ed. B. Furht, pp. 207-225, CRC Press, 1999.

[9] Cliff, David, "Hang the DJ: Automatic Sequencing and Seamless Mixing of Dance Music Tracks," *HP Technical Report HPL-2000-104*, Hewlett-Packard Laboratories Bris-

tol (http://hpl.hp.com/techreports/2000/HPL-2000-104.html)

[10] Tagg, P., “Understanding ‘Time Sense’: Concepts, Sketches, Consequences.” In *Tvärspel: 31 artiklar om musik: Festskrift till Jan Ling*. Göteborg: Skrifter från Musikvetenskapliga institutionen, pp 11-43., 1984. (http://www.theblackbook.net/acad/tagg/articles/timesens.pdf)

[11] Foote, J., “Content-Based Retrieval of Music and Audio.” In *Multimedia Storage and Archiving Systems II, Proc. SPIE*, Vol. 3229, Dallas, TX. 1997.

[12] Pye, D., “Content-based Methods for the Management of Digital Music,” in *Proc. ICASSP 2000*, vol. IV pp 2437, IEEE

[13] Pfeiffer, S., Robert-Ribes, J., Kim, D., “Audio Content Extraction from MPEG-encoded sequences.” In *Proc. First International Workshop on Intelligent Multimedia Computing and Networking (MMCN 2000*), a constituent of JCIS 2000, Atlantic City, New Jersey