THE IMPORTANCE OF SEQUENCES IN MUSICAL SIMILARITY

Michael Casey

Goldsmiths College, University of London
New Cross, London SE14 6NW
m.casey@gold.ac.uk

Malcolm Slaney

Yahoo! Research
Sunnyvale, CA 94089
malcolm@ieee.org

ABSTRACT

This paper demonstrates the importance of temporal sequences for passage-level music information retrieval. A number of audio analysis problems are solved successfully by using models that throw away the temporal sequence data. This paper suggests that we do not have this luxury when we consider a more difficult problem: that is finding musically similar passages within a narrow range of musical styles or within a single musical piece. Our results demonstrate a significant improvement in performance for audio similarity measures using temporal sequences of features, and we show that quantizing the features to string-based representations also performs well, thus admitting efficient implementations based on string matching.

1. OVERVIEW

This paper describes methods for efficiently assessing musical similarity. Our goal is to perform music retrieval, a temporal matching problem, with the same ease that search engines retrieve text on the world-wide web. This paper addresses the issues inherent in temporal matching of musical signals, and describes approaches that are amenable to fast hashing techniques.

There is a range of modeling techniques for describing and finding musical matches. Audio fingerprinting tools [8] find the most salient portions of the musical signal and use the most detailed models to then look for exact matches over time. The text equivalent is looking for the words "Four score and seven years ago" to identify Lincoln's Gettysburg address. Genre-recognition efforts [19], artist recognition [6], musical key identification [12], and speaker identification [14] use much more general models such as histograms or probability densities of acoustic features. These so-called bag-of-feature models ignore the temporal ordering inherent in the signal. The textual equivalent would use the word frequencies of a paragraph of text to identify the document as a 19th century speech.

This paper describes models with an intermediate level of detail so we can find musical passages that are similar to a requested song based on sequential harmony information. In our study, we consider passages to be equivalent if a human listener judges them similar, even if they are acoustically distinct. An example application is the retrieval of all the thematic repeats (recurring melodies) in classical works or popular music tracks, while being robust to changes in the lyrics, instrumentation, tempo, rhythm, chord voicings and so forth. The applications of such similarity methods are far-reaching, and have immediate relevance to music browsing, computational musicology, audio thumbnailing, music structure identification and audio synthesis by *musaicing* [18].

This research was supported, in part, by EPSRC grant GR/S84750/01 (Hierarchical Segmentation and Semantic Markup of Musical Signals).

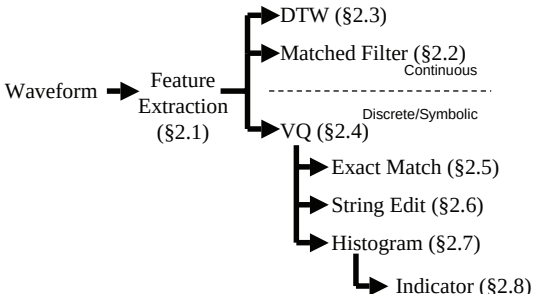

Fig. 1. Overview of the audio feature extraction, temporal modeling and evaluation of quantized and continuous features. Section numbers refer to the text.

We measure the performance of the algorithms in this paper by identifying repeated sections of a song. It is difficult to establish a ground truth for melodic repetition, especially with popular works. A value judgment needs to be made on whether repetitions are perceptually distinct or not. We restricted use of the popular music ground truth to those repeated melodic segments that occurred within the chorus sections of the works. This is because the instrumentation in popular music is often not established in first verse, thus repeats of the verse melodic phrases are subject to more acoustic variation than repeats of melodic materials in the choruses. Using a broader definition of repeats will lower our absolute performance, but we do not think it changes our conclusions.

This paper continues with a description of our method for generating quantized features for audio, similarity matching and evaluation in section 2; we present empirical results in section 3, and draw conclusions in section 4.

2. METHOD

Figure 1 gives an overview of the techniques we use in this paper. We make a distinction between conventional methods that use continuous variables (and their quantized approximation) and the discrete or symbolic representations amenable to hashing approaches. The subsections that follow talk about the acoustic features we use, how we quantize the analog data to form a symbolic representation, and how we perform matching in the continuous and symbolic domains.

2.1. Feature extraction

We investigated two audio representations in this work. LFCC (log-frequency cepstral coefficients) is a simplification of MFCC (mel-frequency cepstral coefficients) that is used to represent timbre in speech recognition and some music tasks [11]. The chromagram representation captures the musical qualities of the sound by collapsing notes across octaves.

We extract features using 375ms windows every 100ms. We used a constant-Q power spectrum [4] with 1/12th octave resolution, aligned with and corresponding to notes in western tonal music. Each element of this spectrum is compressed to approximate loudness perception using a logarithm. We collapse the 86-D spectrum to a low-dimensional representation using one of two methods. We approximate the shape of the spectrum in LFCC with a discrete cosine transform, in the same way that MFCC works to reduce the dimensionality of the auditory spectrum. We collapse each note in the chromagram representation to the base octave, A1–G#2 (55Hz–104Hz), to give an octave-independent measure of the harmonicity of the music.

2.2. Matched Filter

The matched filter is a linear operator for matching a waveform. Whether we do this in the waveform or the cepstral domain, a matched filter looks for an exact match over the given temporal window. The disadvantage of the matched filter is that we can not expect it to find an exact match with human variation, especially as we look at longer windows.

2.3. Dynamic Time Warping

Dynamtic time warping (DTW) is the conventional solution to the tempo-variation problem [5]. Given two different temporal waveforms, DTW finds the time course that generates the best match between the two signals and returns the corresponding matching errors. This computation is expensive [16].

2.4. Vector quantization

Vector quantization (VQ) has been successfully employed in a wide-range of applications in automatic speech recognition and music information retrieval—aside from passage-level music retrieval which is the subject of our study. In this work we do not use VQ with a small quantization interval to find a nearly exact representation of the original signal: instead we are using VQ to convert the signal into a string of relatively small set of symbols for which text tools can be used for processing.

We train vector quantization (VQ) models using unsupervised learning over a fixed third of the training data. Four K-means models with 8, 16, 32 and 64 clusters generate sequences of cluster indices using nearest-cluster assignment; each feature vector is assigned to the nearest of K cluster centers using Euclidean distance. We group the sequence of cluster assignments into short-term windows with durations between 0.5s and 4s and a hop size of 0.1s. For each work in the data set, each windowed VQ sequence was represented in a database as fixed-length string using the base-64 ASCII encoding. The VQ strings are a new feature with the same dimensionality as the window length used to produce them.

2.5. Exact string matching

A discrete equivalent to a matched filter is an exact string comparison. (While the string comparison is exact, the result is an approximate match because of quantization.) When quantizing a low-dimensional signal we can talk about neighboring symbols and we expect some errors are more acceptable than others. But in high dimensions, the curse of dimensionality means that all errors are equally likely [3]. Because of the small number of symbols we are using, we do not expect that exact-string matching to produce very robust results: false positives will be higher.

2.6. String-edit distance

The string edit distance is a good approximation to DTW for discrete symbols. We compute similarity for strings of states using the Levenshtein distance metric, or *string edit distance*, which counts the minimum number of insertions and deletions (indels) and substitutions (swaps or replacements) required to make a query string match the target string. The Levenshtein distance that we use treats all substitutions as equally erroneous.

The Levenshtein distance is the minimum number of insertions, deletions and swaps required to make a test sequence into a query sequence. For example, lev(abc,abc)$=0$, lev(abc,abbc)$=1$, and lev(abc,cba)$=2$. Efficient computation of the Levenshtein distance is the subject of much research across computational disciplines, and has been explored in great detail for applications such as biological sequence comparison and text-based information retrieval [10, 16]. In most applications, a dynamic-programming algorithm finds the minimum possible distance, with implementations being worst-case polynomial order 2 with respect to the string length and linear with respect to the number of strings in the database. Exact string matching by hash table lookup, in contrast, has computational complexity on the order of constant time with respect to the number of strings in the database.

The string-edit distance only approximates the continuous DTW distance, since the discrete symbols are only an approximation to the exact sound, and the Levenshtein metric does not take into account which symbols are easily confused. There are, however, versions of the Levenshtein distance that incorporate non-uniform penalties for swaps [16].

2.7. VQ State Histogram

We use a histogram of the symbols within the matching window as an analogue to the bags of frames approach. We count the number of times each VQ state is found in the matching window. This representation approximates a distribution of the states, with weightings calculated from the occupancies in the match window.

2.8. Indicator String

Our primary motivation is to find a means of scalable audio-similarity matching for music applications with a large number of documents. In pursuit of this goal, we recognize that a possible implementation is a hash-table lookup over the VQ strings. However, our target task consists of repetitions of melodic segments subject to variations in timing, timbre, pitch voicing, instrumentation, rhythmic content and lyrics (voice content) due to natural variation in musical performance. Thus, we expect that melodic repetitions will not result in literal string repetitions, but we must consider temporal re-orderings (swaps), insertions and deletions.

We desire a method to represent strings where we collapse such likely confusions within the string representation itself, rather than by a similarity computation at query time, which is computationally expensive. To this end, we made a new feature consisting of the set of unqiue state labels that occurred in each VQ string ordered deterministically by alphabetical ordering of cluster labels. For example, we represent the length 10 string (zzaaabbzbc) by the indicator function (abcz), as is the sequence (zcbcaczcbca).

Exact matches in this representation are invariant to temporal ordering of the VQ strings, thus temporal information is eliminated in our indicator function representation.

3. RESULTS

3.1. Data

In order to permit evaluation over contrasting musical styles, we collected two sets of data. The first was a corpus of classical music works performed on a real piano and recorded in a reasonably reverberation-free acoustic space. Human performances of three classical works by Bach and Beethoven were recorded into a MIDI file via a Clavinova and played back on a Yamaha Disklavier, an electronically-driven acoustic piano. Each performance was then marked up for repeated melodic content using an audio editor. The pieces were selected for their considerable use of repetition.

The second dataset consisted of seven popular music tracks with markup for melodic repetition provided by the MPEG-7 Audio group [9]. Each repeat of a melody was indicated by its start time. The classical and popular music data were marked up into a total of 68 repeat segments. We generated all possible pairs of ground-truth query/result segments for each song; i.e. C_2^m combinations for n repeated instances of a given melody, in our experiments there were 183 repeats in the dataset.

3.2. Evaluation by results rank

Figure 2 shows the raw-matching similarity as calculated on one piece using the matched-filter approach. There is a strong diagonal component indicating that each segment of the sound is a good match for itself. There are five copies of the same chorus in this piece. Within each box, there is a strong diagonal match, indicating that this portion of the song (time along the x-axis) matches another portion (time along the y-axis) of the song.

We evaluate each of our algorithms by measuring the average rank of each query. Each window in a repeated section of the song is used as a query to find similar sections in the rest of the song. Each query produces distances, dependent on the algorithm being tested, for the entire song, and we sort and rank these distances looking to measure the rank of the correct answers. The absolute closest match for each query is always with itself—we do not try to remove these.

For each window in a repeated section, we record the rank of the appropriate window in each of the other repeated sections. The appropriate window in each repeated section is calculated by linear interpolation from the beginning and end of each section (because each repeat is not played at the same rate.) If there are 3 repeats of a section in a song, each 10 seconds long, and we are using 1 second matching windows, then there will be 90 valid queries per repeat $(10-1)/0.1$, or 270 queries total. There are 3 correct answers per query (one for each repeat, including itself) and the best ranks in a slightly broadened window (+/- 1 frame) around each "correct" position are averaged to get the score per query. The 183 query rankings are summarized by their mean.

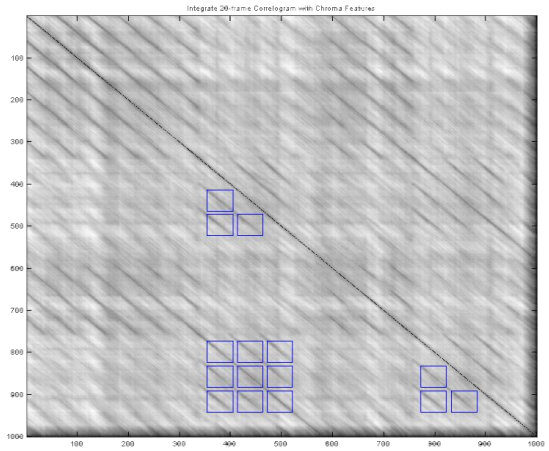

Fig. 2. Intra-song matched-filter temporal similarity matrix: dark pixels correspond to windows of the song that match the repeat within a 30-frame window. The five repeated sections are indicated on the lower-triangular portion matrix with boxes.

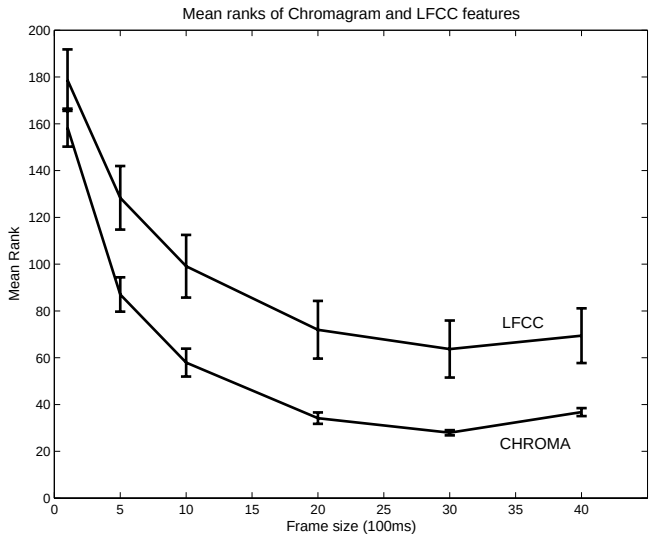

Fig. 3. Mean retrieval performance, as measured by average query rank, of LFCC and chromagram features.

Figure 3 shows the results of matched filter retrieval for the continuous features with window lengths of 1–40 frames. Here, the chromagram features have significantly higher mean rank than the LFCC features, the standard deviation is also lower for the chromagram features. This result justifies our choice of chromagram features for the next experiment.

Figure 4 compares the performance of the string features and the continuous features. The string-edit distance performed the best, with the lowest mean rank. The histogram VQ score is significantly worse than the string-edit distance, indicating that the 'bag-of-frames' approach is not appropriate for this task. The indicator

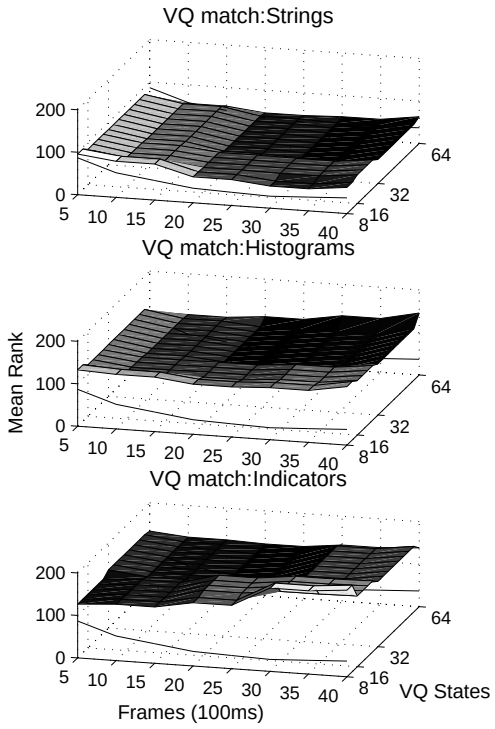

Fig. 4. Mean retrieval performance for the symbolic features: string-edit distance, histograms and indicator functions. For reference, the performance of the continuous chromagram features from Figure 3 are shown at the front and back of the plot.

function shows similar performance to the histogram.

The indicator function is, in essence, a quantized histogram and performs similar to the histogram except in two cases. For short window lengths, with a large number of VQ states the decrease in performance is due to (the natural) correlations between successive frames of the data. When the windows are short, we expect to see most of the positions occupied by the same state. The indicator function for that state is fixed at one, while the histogram can grow to reflect the correlation. In the second case, the window length is much greater than the number of states, so the histogram does a better job of representing the multiple hits per state.

4. CONCLUSIONS

We have demonstrated the importance of temporal features in a music-similarity task. We looked at several different forms of musical representation and distance measures. We showed that temporal queries were more effective at retrieving musically similar segments of our music library. We are encouraged that string-based methods work so well, and we now wish to study efficient methods to perform efficient string matching so we can apply our ideas to today's million-song libraries.

5. ACKNOWLEDGEMENTS

The authors gratefully acknowledge many valuable insights from Christophe Rhodes at Goldsmiths College.

6. REFERENCES

[1] Mark A. Bartsch and Gregory H. Wakefield, "To Catch a Chorus: Using Chroma-Based Representations for Audio Thumbnailing," in *Proc. WASPAA*, 2001.

[2] J. R. Bellegarda and D. Nahamoo, "Tied mixture continuous parameter models for large vocabulary isolated speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pp. 13–16., 1999.

[3] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press, 1995, section 1.4.

[4] J.C. Brown and M.S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *J. Acoust. Soc. Am.*, 92(5):2698–701, November 1992.

[5] R. B. Dannenberg, H. Hu, "Polyphonic Audio Matching for Score Following and Intelligent Audio Editors," *Proc. ICMC-03*, Singapore, October 2003.

[6] D. Ellis, B. Whitman, A. Berenzweig, S. Lawrence, "The Quest for Ground Truth in Musical Artist Similarity," *Proc. ISMIR-02*, pp. 170–177, Paris, October 2002.

[7] Jonathan Foote, "Visualizing music and audio using self-similarity," in *ACM Multimedia (1)*, 1999, pp. 77–80.

[8] J. Herre, E. Allamanche, O. Hellmuth, T. Kastner, "Robust identification/fingerprinting of audio signals using spectral flatness features," *Journal of the Acoustical Society of America*, Volume 111, Issue 5, pp. 2417–2417 (2002).

[9] International Standards Organization, *Coding of Moving Pictures and Audio 15938-4(Audio)*, 2002.

[10] V. I. Levenshtein, "A binary code capable of correcting spurious insertions and deletions of ones," *Cybernetics and Control Theory*, 10(8):707–710, (1966).

[11] B. Logan, and S. Chu, "Music Summarization Using Key Phrases," in *Proc.IEEE ICASSP*, Turkey, 2000.

[12] S. Pauws, "Musical Key Extraction from Audio," in *Proc. ISMIR*, Barcelona, 2004.

[13] Jan Puzicha, Thomas Hofmann, and Joachim M. Buhmann, "Histogram clustering for unsupervised image segmentation," in *Proc. of CVPR*, 1999.

[14] Douglas A. Reynolds, "Speaker identification and verification using Gaussian mixture speaker models," *Speech Commun.*, 17 (1-2):91–108, 1995.

[15] C. J. van Rijsbergen, *Information Retrieval*, Butterworth, 1979.

[16] David Sankoff and Josepth Kruskal, *Time warps, String Edits and Macromolecules. The Theory and Practice of Sequence Comparison*, Mass.: Addison-Welsey, 1983.

[17] Eric Scheirer and Malcolm Slaney, "Construction and evaluation of a robust multifeatures speech/music discriminator," *IEEE Transactions on Acoustics, Speech, and Signal Processing* (ICASSP'97), 1997, pp. 1331–1334.

[18] D. Schwartz, "The Caterpillar System for Concatenative Sound Synthesis," in *Proc. DAFx*, London, 2003.

[19] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals.," "*IEEE Transactions on Speech and Audio Processing*", 10(5):293–302, 2002.