Globally Optimal Audio Partitioning

Eric Nichols
Dept. of Computer Science, Indiana Univ.
epnichol@indiana.edu

Christopher Raphael
School of Informatics, Indiana Univ.
craphael@indiana.edu

Abstract

We present a technique for partitioning an audio file into maximally-sized segments having nearly uniform spectral content, ideally corresponding to notes or chords. Our method uses dynamic programming to globally optimize a measure of simplicity or homogeneity of the intervals in the partition. Here we have focused on an entropy-like measure, though there is considerable flexibility in choosing this measure. Experiments are presented for several musical scenarios. [1]

Keywords: audio partitioning, dynamic programming

1. Introduction

The signal-to-score problem seeks to represent symbolically the content of a music audio file. While there are many ways in which the problem can be posed, perhaps the most ambitious seeks a representation that captures the most important elements of Western music notation including pitch, rhythm, and voice. Given the significant difficulty of this goal, several simpler problems have been considered including signal-to-midi, which does not attempt to attribute rhythm or voice to the recognized pitches; monophonic; or polyphonic single-instrument transcription [1], [2], [3], [4]. While the choice of problem is a difficult one, the relevant tradeoffs seem clear: simpler problem statements reduce the complexity of what we seek to estimate and are usually easier; on the other hand one hopes that with a richer interpretation of the musical content, some elements which are less ambiguous will help to reinforce a correct interpretation of those that are more ambiguous.

We explore here a relatively simple problem statement which could as easily be viewed as a preprocessing technique as recognition. We seek to partition the audio into segments having nearly constant spectral properties and thus likely to represent single notes or chords.

Our goal is motivated, in part, by the well-known tradeoffs between time and frequency resolution faced when choosing the length of a Fourier transform: to make precise estimates of underlying frequencies, and hence pitch, one wants to use longer analysis segments. However, if a segment overlaps note boundaries, our interpretation of the spectral content is confounded. Thus in this analysis we seek the longest variable-length segments possible that do not overlap note boundaries. In exploring tradeoffs within this effort we hope to oversegment rather than undersegment the data, so that each recognized segment does not cross note boundaries. While such a segmentation could be used to provide useful pitch and rhythmic analysis of the audio data, it could also be used simply as a means of locating reasonable "frames" for more sophisticated analysis. Such frames would provide better frequency resolution due to their maximally long nature.

[1] This material is based upon work supported by the National Science Foundation under Grant No. IIS-0534694 and the Center for Research on Concepts and Cognition at Indiana University.

2. Partitioning through Optimization

Our basic approach is to choose a partition of the data that optimizes some objective function measuring the overall appropriateness of the partition. In particular, our approach is based on dynamic programming, and requires our objective function to be expressed as a sum of contributions, each depending only on a single interval of the partition. To be specific, suppose we view our audio data as a collection of M frames, each of length N samples with actual samples $s_0, \ldots, s_{NM-1}$. Let $b_0 < b_1 < \ldots < b_n$ be a sequence of breakpoints partitioning the audio data into a collection of segments. We write (b_{k-1}, b_k) for the sequence of samples $s_{Nb_{k-1}}, \ldots, s_{Nb_k-1}$ and we assume $b_0 = 0$ and $b_n = M$. If $H(b_{k-1}, b_k)$ is our measure of the quality of the particular interval, then the overall measure of the partition is given by

$$H(b_0, \ldots, b_n) = \sum_{k=1}^{n} H(b_{k-1}, b_k)$$

A simple dynamic programming argument shows that we can compute the optimal partition recursively. Let $H^*(b)$ be the score of the *optimal* partition of the audio data $(0, b)$. The optimal partition of $(0, b)$ must either be the unpartitioned interval or it must be composed of the optimal partition of of $(0, a)$ with the additional interval (a, b) appended for some $a < b$. The usual idea of dynamic programming then leads to

$$H^*(b) = \min_{a=0}^{b-1} H^*(a) + H(a, b) \qquad (1)$$

where the case of a single interval is accounted for by the $a = 0$ case (we define $H^*(0) = 0$).

The recursion can be modified to find the best partition into intervals (a, b) where $l_{\min} \leq b - a \leq l_{\max}$ by redefining $H(a, b) = \infty$ when $b - a < l_{\min}$ or $b - a > l_{\max}$. Restricting the minimum size of intervals has the benefit of excluding intervals too short to reasonably be a note, while restricting the maximum size decreases the number of intervals on which we must compute $H(a, b)$ — overwhelmingly the main cost in implementing the algorithm.

A simple modification of the algorithm instead minimizes

$$H(b_0, \ldots, b_n) = \sum_{k=0}^{n-1} H(b_k, b_{k+1}) + n\lambda$$

where we have added the penalty λ for each interval in the partition, thereby encouraging explanations involving as few intervals as possible. The penalized version of the objective function is minimized with the recursion

$$H^*(b) = \min_{a=0}^{b-1} H^*(a) + H(a, b) + \lambda 1_{a \neq 0} \qquad (2)$$

where the latter term is 1 only when $a \neq 0$.

Our algorithm for finding optimal partitions is well-known in the literature [5] and is almost a canonical example of dynamic programming. The time complexity of the algorithm is clearly $O(M^2)$ since each possible interval must be considered once in Eqn.1. However, if we restrict the maximal interval size, then the algorithm is $O(M)$.

3. Minimizing Entropy

Our measure of the quality of a partition is defined in terms of Fourier transforms. Suppose our sequence of M frames is partitioned into several contiguous sections $f_1, \ldots f_n$ of lengths $l_1, \ldots, l_n$ with $l_1 + \ldots l_n = M$. Here the f_k are the actual sample vectors corresponding to the intervals. Suppose we take the finite Fourier transform of each f_k yielding k transforms $F_1, \ldots, F_k$. The Parseval relation assures us that the total energy in is preserved between each (f_k, F_k) pair:

$$\sum_i f_k^2(i) = \sum_j |F_k(j)|^2 \qquad (3)$$

Consequently, if we sum the energy over all cells of the partition we see

$$\sum_{i,k} f_k^2(i) = \sum_{j,k} |F_k(j)|^2 \qquad (4)$$

Since the left hand side is simply a constant — the total sum of squares of the audio data, which is independent of choice of partition — the right hand side must also be independent of the choice of partition.

Our goal is to choose a partition so that F_k cluster their energy in as compact a way as possible. For this purpose we choose to minimize the *entropy* over all partitions defined as

$$H = -\sum_{j,k} |F_k(j)|^2 \log_2(|F_k(j)|^2)$$

While it might seem more natural to define the entropy on the version of $|F_k(j)|^2$ normalized to sum to 1, as the usual definition does, the result of minimizing either entropy over the partition choice will be the same due to Eqn. 4.

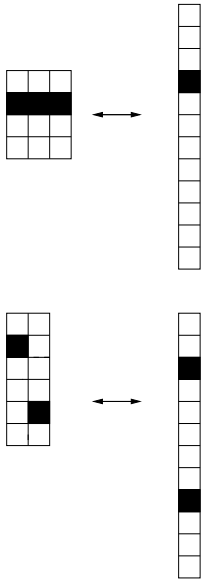

Figure 1. Top: Merging similar frames results in reduced entropy. Bot: Merging dissimilar frames results in increased or equivalent entropy.

Figure 1 gives an intuitive explanation of why minimizing entropy might lead to reasonable partitions. In the top panel of this figure, the left hand member shows a cartoon-like picture of $|F_k(j)|^2$ over a series of several consecutive original (unconcatenated) frames, all contained within a single musical note. While there may well be many harmonics in the spectra, for simplicity's sake we focus on a single one, the horizontal strip in the left-hand-side of the figure. Now suppose we concatenate these frames into one large frame as in the right-hand member of the figure. Clearly the entropy will be smaller for the right-hand member since we have concentrated all the energy on a single "pixel." Note that the number of pixels is the same in both cases. Thus the energy minimizing partition tends to merge frames corresponding to the same note or chord, since this leads to lower entropy.

In the bottom panel of Figure 1 we contemplate the merge of neighboring frames corresponding to *different* notes or chords. While each of these are represented horizontally as "one-pixel-wide" spectra, each frame might be the result of several merges at some earlier stage. The result of merging these two frames results in the spectrum on the right. Note that both cases concentrate the energy on the

same number of pixels, suggesting that the entropy measure is indifferent to whether or not we merge here. However, a more lifelike version of this situation would reveal that peaks in the right-hand spectrum will be more spread out than in the left-hand side, due to the finer resolution of the Fourier transform. For this reason the entropy criterion will tend to favor the situation on the left, as is consistent with our goal.

While the experiments presented herein use the entropy measure as the objective function to be minimized, other reasonable choices are possible and worthy of further study. One of the other objective functions we have considered is based on *conditional* entropy — the partition that minimizes

$$G(b_0, \ldots, b_n) = \sum_{k=1}^{n} H'(b_{k-1}, b_k) E(b_{k-1}, b_k) + n\lambda$$

where $H'(b_{k-1}, b_k)$ is the entropy of the squared frequency energy of segment (b_{k-1}, b_k), *normalized* to sum to one, and $E(b_{k-1}, b_k)$ is the total squared energy in the segment.

The other measure we have considered is based on autoregression and is defined by

$$G(b_0, \ldots, b_n) = \sum_{k=1}^{n} A(b_{k-1}, b_k) + n\lambda$$

where $A(b_{k-1}, b_k)$ is the total residual squared error when an autoregressive model of fixed order is fit to the data in (b_{k-1}, b_k).

4. Experimental Results

We implemented the dynamic programming algorithm described above in the C language. All experiments were performed on a 2.1 GHz Linux PC. Audio inputs were recordings of musical performances subsequently sampled down to 8 kHz mono audio. We fixed the frame size N to 256 samples for all experiments. The Fast Fourier Transform (FFT) was used to speed up entropy calculations.

Because the FFT algorithm used requires the number of sample points to be a power of two, whereas our partition segments are not so constrained, we use the standard technique of zero-padding the data to the required size. Adding these additional data points has the effect of interpolating the FFT over the entire set of frequency bins as well as shrinking the magnitudes so that the squared norm (Eqn. 3) is preserved. Entropy is not preserved during zero-padding; however, we can approximate the desired non-zero-padded entropy by

$$H = \sum_{j,k} |F_k(j)|^2 \log_2 \left(|F_k(j)|^2 \frac{J'}{J} \right) \qquad (5)$$

where J is the number of samples and $J' = 2^{\lceil \log_2 J \rceil}$ is the size of the zero-padded data.

To speed up experimentation, the program precomputes $H(a, b)$ for all possible pairs of breakpoints subject to $a < b$ and $b - a \leq l_{\max}$ and stores the results on disk. For two minutes of audio, this computation would take approximately 18 minutes with l_{max} set to 100. If desired the performance could be improved by approximating longer FFTs based on shorter FFT results. Once the entropy values were recorded, they were used in multiple experiments by the dynamic programming algorithm for a range of values for parameters λ and $l_{\min}$. Each computation of an optimal partition via the dynamic programming algorithm took a negligible amount of time when using the precomputed $H(a, b)$ values.

We used the program to compute optimal partitions for several polyphonic audio examples, including recordings of two Chopin Preludes (Op. 28, Nos. 7 and 20), and Mvt. 1 of the Schostakovich String Quartet No. 3, Op. 73, henceforth `chopin7`, `chopin20`, and `schostakovich`. Figures 2, 3, and 4 show partitions generated by the algorithm for each of these examples.[2] Each figure displays a spectrogram of the data with vertical lines indicating breakpoints found by the algorithm. Particular values of λ selected for each file to obtain these results were 0, 0.1, and 0.01, respectively.

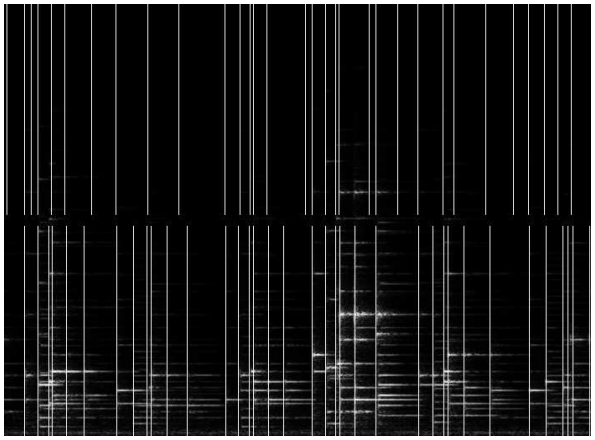

Figure 2. Section of Chopin Prelude No. 7, Op. 28

Figure 2 compares the algorithm results (partitions in the top half of the figure) with the *true* note onset times (bottom half of the figure). Many of the note onsets correspond exactly with the generated partitions. Repeated chords are a common omission in our results, as are note onsets where previous notes are sustained over the new entry. In the `chopin20` results we find that overpartitioning often breaks chords up into a segment during the attack portion and a segment after the chord has decayed substantially. For the `schostakovich` partition we see that some quieter notes in the strings are grouped together into underpartitioned segments.

[2] Audio files of these results in .wav format are available online at `http://xavier.informatics.indiana.edu/~craphael/ismir06/`. The partitions generated by the algorithm are represented by audible clicks in these files.

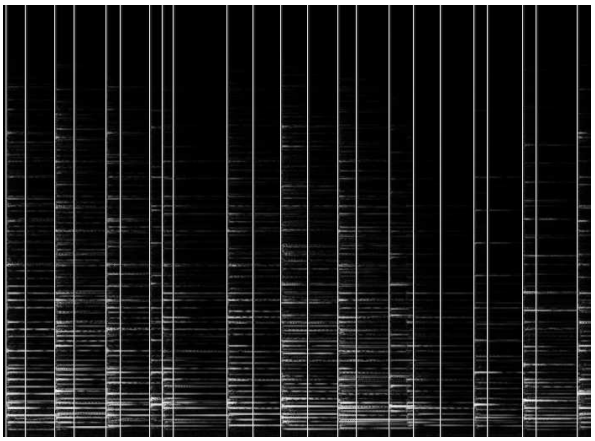

Figure 3. Section of Chopin Prelude No. 20, Op. 28

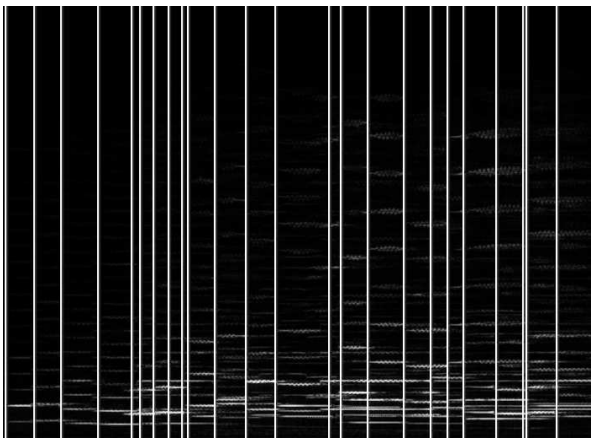

Figure 4. Section of Schostakovich String Quartet No. 3

For each experiment we have an important choice of the penalty parameter λ as well as selection of the allowed range of partition sizes specified by $l_{\min}$ and $l_{\max}$. Typical values of $l_{\max}$ for our experiments were 100 or 200 frames, selected to reduce computation time while ensuring that the longest notes in each experiment would still fit within a single interval of a partition. $l_{\min}$, on the other hand, was set to 4 frames for each experiment, corresponding to the shortest sixteenth notes in the `chopin7` example.

Different audio recordings yield different characteristic entropy values. To choose a "good" λ for a particular input, we would search for a value that resulted in neither serious overpartitioning (suggested by a prevalence of breakpoints separated by $l_{\min}$) or underpartitioning (indicated by breakpoints separated by $l_{\max}$). We would aim for a value of λ that tended to slightly overpartition the data. Serious overpartitioning was not always a possibility; for example, the "best" λ turned out to be 0 for `chopin7`. Typically we would find a good result with $\lambda < 1$. The problem of λ selection merits further study, although we know that the relationship between λ and the number of intervals in the partition may be understood by thinking of λ as specifying a "geometric prior" on the number of breakpoints [5].

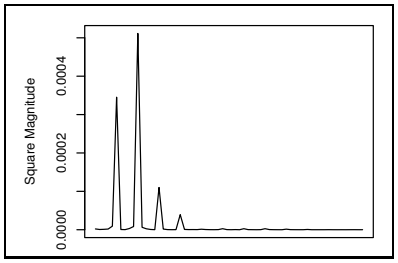

Figure 5. Spectrum of frame 240.

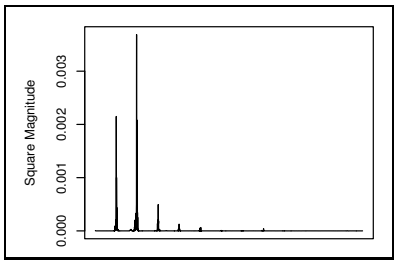

Figure 6. Spectrum of frames 227-259.

5. Conclusion

The goal of entropy-driven audio partitioning is to segment audio input in a manner that maximizes the amount of consistent information within a segment. Fourier transforms of larger segments should provide a more precise characterization of frequency distribution. As an example of the utility of our technique, consider the partial spectrum from one frame of the `schostakovich` data in Figure 5. This single frame is included in a 33-frame long segment in the partition computed above; Figure 6 shows the spectrum computed from this longer segment. The peaks are much more well-defined due to the improved frequency resolution, and as such may be more amenable to analysis by other audio processing algorithms.

References

[1] Saito S., Kameoka H., Nishimoto T., Sagayama S., "Specmurt Analysis of Multi-Pitch Music Signals with Adaptive Estimation of Common Harmonic Structure," *Proc. Int. Symp. Music Info. Retrieval, 2005*, pp. 84–91, London UK.

[2] Klapuri A., "Multiple Fundamental Frequency Estimation by Harmonicity and Spectral Smoothness," *IEEE Trans. Speech and Audio Processing*, 11(6), 804-816, 2003.

[3] Cemgil A. T., Kappen H. J. , Barber D. "A Generative Model for Music Transcription," *IEEE Transactions on Audio, Speech and Language Processing* 14(2), March 2006.

[4] Kapanci E., Pfeffer A., "Signal-to-Score Music Transcription Using Graphical Models," *Proc. 19th Int. Joint Conf. on Artif. Intel (IJCAI), 2005*, Edinburgh, UK.

[5] B. Jackson, J. D. Scargle, D. Barnes, et al., "An algorithm for optimal partitioning of data on an interval", *IEEE Signal Processing Letters*, vol 12, no. 2, pp. 105–108, Feb. 2005.