

Improving Music Information Retrieval using Segmentation-related Statistics

Stephen Travis Pope

Graduate Program in Media Arts and Technology, Dept. of Music
UC Santa Barbara, Santa Barbara, California, USA
stephen@create.ucsb.edu

ABSTRACT

The next generation of music databases will be required to search on musical genre, mood, song style and user preference. For this, we need analyzer programs that create song feature vectors in a multi-stage process of signal processing, data smoothing and statistics, and song segmentation. This paper discusses the high-level features that can be derived from music segmentation, i.e., finding the break-points between the musical verses and choruses. We address segmentation and related statistics, and present a software implementation and test data for discussion and evaluation.

Keywords

Music information retrieval, multimedia databases, audio analysis, music segmentation, data-mining, machine-learning

1. INTRODUCTION

Music information retrieval (MIR) software uses signal analysis for feature extraction from song data in applications such as genre/mood classification, finger-printing, thumb-nailing and playlist generation (recommendation). MIR analyzers perform detailed signal analysis and post-analysis data processing and statistics, creating feature vectors with a variety of time- and frequency-domain fields [5].

This paper describes the use of a music segmenter to deliver a set of database features that improve the quality of MIR applications. A number of high-level statistics are gathered based on the output of the song segmenter, and these prove useful in a variety of data-mining and machine learning algorithms and a playlist-generation application. We introduce music segmenters and then the array of segmentation-derived features that can be added to the feature vector. Results are evaluated of the role that these new features play in tasks like principal component analysis, rule-learning, and similarity metrics.

The MIR song analysis process consists of a windowed feature extraction stage that reads audio data and performs signal analysis routines on short windows (1-50 msec = 64-2000

samples), yielding a set of raw time- and frequency-domain (and other) features. The second-pass analysis smoothes the raw data and derives higher-level features (e.g., tempo or spectral tracks), followed by pruning and reduction based on numerical/statistical analysis such as histograms or Gaussian mixture models. The third (application-specific) stage involves machine-learning or data-mining techniques such as clustering, classification or structure-learning [5].

2. MUSIC SEGMENTATION

Automatic music segmentation means finding the break-points between the verses and choruses of a song (or the structural segments of a classical composition). For a pop song with clear instrumental breaks, or a jazz piece with tempo changes at the start of each solo, a segmenter would simply have to collect the feature vector data, and look for regularly spaced peaks in the weighted distance between windows. In fact, for many simple cases, this technique is sufficient to locate segment boundaries, and successful segmentation systems have been reported [1, 6].

Finding a musically relevant segmentation means using distance metrics and inter-segment-boundary detection and working either bottom-up (grouping short segments into subsets of the song) or top-down (dividing the song into a string of segments) to pick a song segmentation expressed as a list of "verse/chorus" break-points. Our approach [4] is to find the peaks in the inter-window distance function, build a tree of them based on the expected repetition of short segments, and find the best matches for the song.

We start by creating the peak-index list from a distance function, and guess at a peak period (segment length) from the function's auto-correlation for which to search. We traverse the peak-index list looking for peaks with a periodicity at the selected period; if we find a strong periodicity (regularly spaced distance function peaks over a long interval), we nominate the chosen peak and period as a part of a segmentation candidate. We can quickly generate a list of proposed segmentations based on different starting peaks and periods, and combine these to cover the song. To evaluate a collection of proposed segmentations, a polynomial confidence value is computed based on factors such as:

- * # of segments per song (2-8 preferred)
- * % of song accounted for (aside from intro/outro)
- * % of peaks accounted for (% off-beats)
- * # of peaks per segment (hope for many)
- * Which weighting was used (prefer broad weightings?)
- * Peak-match tolerance (prefer stricter timing?)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MAST '09, Santa Barbara, California USA, January, 2009

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

In our experience, small changes in this weighting influence the segmenter success rate, but the cases of having too many segments (i.e., finding 2-bar sections rather than whole verses) or too few (1st-half/2nd-half), can both be controlled. For complex music, or heavily compressed productions (either in the sense of dynamic range compression in post-production, or of lossy compressed encoding), segmentation techniques are required that incorporate an adaptive distance weighting and/or a multi-pass distance peak-finding algorithm. The specific challenges to segmentation include tempo changes, intro/outro sections, click-track tempo, and songs with very compressed dynamic range.

3. SEGMENTATION-DERIVED FEATURES

Assuming a segmenter with a high confidence for the majority of songs in a database, the question arises of what new high-level features can be derived from the segmentation data to assist specific applications. The output of the segmenter will be represented as a list of segment breakpoints; from this data, we can derive a whole array of related (and semantically relevant) musical features, although the segmentation itself is all that's required for some applications (e.g., thumb-nailing or summarization).

First, we look at the first and last regions (whether they're in-segment or not) and see if they look like fade-in or fade-out sections, or more like contrasting intro or "outro" sections for the song (coda, repeat of the chorus, etc.). Next, we compute the per-segment averaged (or mean value) feature vectors and try to identify the "typical" and "solo" segments, i.e., the most average-sounding segment that starts at regular period, and the segment at the same period that's most different and is not the intro/outro section.

If we succeed at finding these, we create a new set of segmentation features based on the ratios of selected features between the "typical" and the "solo" segments. A third statistic of interest is the segment-level dynamic range, expressed as the percentage of segments that are significantly louder or quieter than the typical segment.

The solo/verse segment data can be used to select short sections (e.g., 1 beat or 1 second) for detailed analysis, for example to identify the main voice(s) or the spectral signature of the solo instrument. The segmentation data is useful for pruning the main song feature vector, summing the core features over the chosen verse (or just a few seconds of it), and storing the mean and variance (or GMM or histogram) for each feature. We are exploring other segmentation-derived features; this list represents the current feature set.

Collecting the outputs of the segmenter and the derived features together, we have:

- * Segmenter confidence (range 0 - 1; value < 0.2 → ignore)
- * Number of segments (2 - 16 in normal music)
- * Verse length (10 - 60 seconds)
- * First verse start (length of intro/prelude)
- * Typical/Solo start (start of the verse/solo section)
- * Typical/Solo index (1-16, rarely 1 or 16)
- * Quiet/Loud sections (% of quiet/loud sections)
- * Fade-In/Out (# secs to reach avg. amplitude)
- * Solo/Verse ratios: RMS, Tempo, SpectralCentroid, SpectralVariety, DynamicRange (more possible here)

As an example of a successful segmentation, the text below is the output of a database query for some of the segmenter features of a pop/dance song that segmented well (numerical features are normalized to the range 0-1).

```
Database fields:
| title | segmentweight | numsegments
| I Believe In Love | 0.923772 | 0.24

| verselength | typicalstart | solostart
| 0.631119 | 0.280232 | 0.590672

| s_centroid | s_variety | s_tempo | s_dynrange
| 0.4991 | 0.001422 | 0.3360 | 0.654455
```

Note first that the confidence (segmentweight) is high (0.924); the numsegments of 0.24 corresponds to 7 segments (normalized to a maximum of 30 per track). The verselength (0.63, normalized to 0-30 seconds), typicalstart (0.28 of the way through the song), and solostart (0.59 of the way through the song) values are each correct. The solo/verse ratio features (s_centroid, s_variety, s_tempo, s_dynrange and especially the s_variety, the solo-verse spectral variety ratio) are all quite low because this song has an acoustic piano as the solo instrument, i.e., it's much "darker," less dynamic in timbre, and perceptually "slower" than the vocal verses.

4. EVALUATION AND DISCUSSION

An analyzer for this feature vector has been implemented over several generations in the form of the Music Analysis Kernel (MAK) in C++, Smalltalk, MATLAB and SQL [3, 4]. The MAK feature vector incorporates a rich set of time- and frequency-domain features and uses segmentation-informed statistics to store a single song-typical mean and variance value for each of the 26-40 core features. A multi-output tempo estimator delivers tempo-related features alongside a bass pitch follower and key/chroma analysis. Beyond this, we store several beat histogram features [6], including the low/high peak amplitudes, BPM tempi and octave sums. We add the fluctuation-pattern (FP) measures fp_gravity, fp_bass and fp_focus [2], and the set of 17-20 segmentation features introduced above.

The segmenter is a multi-tolerance peak-tree blackboard algorithm with the complex confidence measure given above; for our (eclectic) test set of 15000 songs, allowing up to 16 segments per song, it fails about 3% of the time (about equal to the ratio of spoken-word content in the test set). If we reduce the limit to 12 segments per song, the failure rate only rises to 7%.

How can we quantify the contribution of the new features in real-world applications (their "applied information gain" as it were)? We turn to dimensionality reduction, information theory and data mining techniques, and start with the principal component analysis (PCA) of the song data, which delivers polynomial equations that describe the dimensions of decomposition (the principal components) of the data set. The quality of the analysis is reflected in the number of components whose weight exceeds a given threshold. To test this, we "prune" the data and repeat PCA decomposition with three subsets of the feature vector:

- * FV1 = 26 base-features and their variances
- * FV2 = FV1 + beat histogram and flux-patterns
- * FV3 = FV2 + 17 segmentation-derived features

Figure 1 shows the weights of the PCA dimensions contributing more than a threshold of 0.09. For FV1 (left-most and lowest data series), we see that only 9 PCA dimensions were identified, while FV2 (adding the advanced features), gave us both higher PCA dimension weights and more of

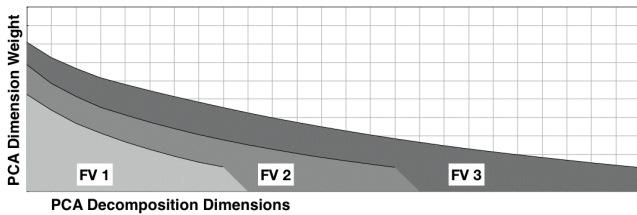


Figure 1: PCA dimension weights for simple (FV1) and extended (FV2, FV3) feature vectors showing the contribution of the new segmentation-derived features to the decomposition.

them (16). For the vector that includes the segmentation features (FV3), we observe both higher values and more dimensions (26) for the PCA processing.

If we look at the actual PCA equations (listed in [4]), we see that the first dimension consists exclusively of high-level spectral variables (spectral measures, MFCC, spectral tracker data and their variances), while dimension two is a mix of fluctuation pattern, beat histogram, tempo, segmentation, and solo ratio features (SegmentWeight, QuietSections, SoloStart, SoloTempo, FadeIn and LoudSections). The third dimension combines a different set of segmentation data (VerseLength, QuietSections, LoudSections and FadeOut). This is very encouraging.

As another test, we derive PART decision lists, a data-mining technique that "discovers" weighted rules describing the data set; for our test data, the segmentation features play a role in many PART rules, such as,

```
ZeroCrossingsVar <= 0.115844 AND
fp_bass <= 0.219083 AND
LoudSections <= 0.003411 AND
NumSegments <= 0.04 AND
HPRMS <= 0.681665 --> New Age (3.0)
```

A series of "blind" playlist-generation tests were run, meaning that we used only content-derived features in the search similarity metric. We compared these play-lists to the output of recommender systems that use human-supplied collaborative filtering metadata, and can say that the content-only solution is quite competitive with the best of them [4]. We have also observed the power of the new features in both CURE and Oracle database clusterers, an SVM-based classifier, and similarity-based search.

The next step in improving the segmentation algorithm will look into the effect of the confidence measure and the lower and upper limits on the number of segments on the segmentation process. Some components of the confidence should be obvious, such as preferring longer segments, and hoping that the list of segments found does account for most of the musical content. For other measures such as the distance peak-matching tolerance or the percentage of peaks accounted for, it is not at all obvious whether they should factor in to the evaluation of a given segmentation.

There is a debate on the limit on the number of segments allowed in a segmentation; for many applications (esp. summarization), it is appropriate to allow up to 30 (or even more) segments per song, meaning that we may well be identifying 4-bar phrases as segments. In the case of our primary application (automatic playlist-generation), we chose to allow up to 30 segments, since our segmentation-derived

features still work well; we'll generally label the start of the refrain as the "typical" section, and some part of the instrumental solo or bridge as the "solo" segment, though this would be problematic for some applications.

Even in cases of segmentation failure, we can determine the fact of the failure by the low confidence value, too many segments, or extreme solo ratio features. We use a simple SQL script to prune the database, setting all of the segmentation features to NULL for these (since there are so few of them). These songs are still not counted as "outliers" in the database, since our distance metric (earth-mover's distance) does not punish records with missing feature values.

5. CONCLUSIONS

It is the goal of this effort to improve music information retrieval (MIR) software by proposing new analysis features derived from the musical segmentation of the song content. Our process starts by segmenting a song to find the verse/chorus boundaries and identify the typical verse and solo section break-points.

We define 15-20 new audio features that we derive from the song segmentation, including fade-in/out and intro/outro statistics, identification of the most average and least average (solo) verses of the song, and computation of the ratios of several spectral and tempo measures between the verse/solo segments. Lastly, we use the segmentation data to prune the base feature vector, storing each feature's weighted average/variance within the typical verse as our reference data.

The analysis processes has been implemented and tested in the MAK software, and we were able to demonstrate a robust segmenter whose output lead to significant improvements in the quality of the resulting data set, e.g., the weights of the first few PCA dimensions and the contribution of our new features to these dimensions. We evaluated the data set using a range of statistical data-mining processes (rule-learning, clustering, classification).

A content-based playlist-generation system was built using the new features, with no human-supplied metadata such as playlist co-occurrence; it arguably performs better than the best "user-informed" recommenders in the field.

We expect that there are many applications yet to be discovered in other domains of MIR that can use this kind of high-quality segmentation statistics and segmentation-derived higher-level song data, and hope that others will extend and refine this initial list of semantically relevant music-structural database features.

6. REFERENCES

- [1] J. Foote and M. Cooper. Media segmentation using self-similarity decomposition. In *Proc SPIE*, 2002.
- [2] E. Pampalk. Audio-based music similarity and retrieval. In *Proc MIR eXchange (MIREX)*, 2006.
- [3] S. Pope. *The FASTLab Music Analysis Kernel*. <http://FASTLabInc.com/FMAK.pdf>, 2001.
- [4] S. Pope. *Music Information Retrieval using High-level Audio Features and Segmentation Statistics*. http://FASTLabInc.com/MAT_MIR.pdf, 2008.
- [5] S. Pope, F. Holm, and A. Kouznetsov. Feature extraction and database design for music software. In *Proc International Computer Music Conference*, 2004.
- [6] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. In *IEEE Trans on Sp-Audio*, 2002.