A SUPERVISED APPROACH FOR DETECTING BOUNDARIES IN MUSIC USING DIFFERENCE FEATURES AND BOOSTING

Douglas Turnbull[1], **Gert Lanckriet**[2]
Computer Science & Engineering[1]
Electrical & Computer Engineering[2]
University of California, San Diego
La Jolla, CA 92093, USA

Elias Pampalk, Masataka Goto
National Institute of Advanced
Industrial Science and Technology (AIST)
Tsukuba, Ibaraki 305-8568, Japan

ABSTRACT

A musical boundary is a transition between two musical segments such as a verse and a chorus. Our goal is to automatically detect musical boundaries using temporally-local audio features. We develop a set of *difference* features that indicate when there are changes in perceptual aspects (e.g., timbre, harmony, melody, rhythm) of the music. We show that many individual difference features are useful for detecting boundaries. By combining these features and formulating the problem as a supervised learning problem, we can further improve performance. This is an alternative to previous work on music segmentation which has focused on unsupervised approaches based on notions of self-similarity computed over an entire song. We evaluate performance using a publicly available data set of 100 copyright-cleared pop/rock songs, each of which has been segmented by a human expert.

1 INTRODUCTION

Most pop/rock songs have a standard structure: an introduction followed by alternating verses, choruses, and solos/bridges segments, and concluding with an outro. We define a *musical boundary* as the point in time where a song transitions between two of these segments. A boundary is often associated with one or more perceptual cues such as the introduction of a new instrument, a key change, or a drum fill. Our goal is to first design low-level features that encode information related to such cues, and second, use these features to develop a system that can automatically detect musical boundaries. Automatic boundary detection is useful for generating music thumbnails [10], efficient music browsing [9], music information retrieval [16], and as a front-end for semantic music analysis systems [14, 11].

There are three main contributions of our work. First, we describe a set of local *difference* features each of which is a time series that is designed to indicate (e.g., 'peak') when some aspect of the music changes. Each feature is calculated by sliding a *difference window* across the audio signal and comparing the information extracted from the first half of the window with the second half of the window. Each low-level feature is loosely related to a high-level musical concept such as timbre, harmony, melody, and rhythm. We individually evaluate each difference feature and report how well each feature can be used to detect musical boundaries.

Second, we combine these features and propose a supervised learning approach based on the AdaBoost algorithm [3, 15, 5]. For each of our features, we create a large number of additional features by first smoothing the feature and then calculating the instantaneous derivatives of the smoothed versions of the feature. We then simultaneously sample each feature in this enlarged set of features to create a high-dimensional ($\sim$ 800 dimensions) *vector of feature values* for each sample [1] . The samples that correspond to musical boundaries are labeled as belonging to the 'boundary' class while all others are labeled as belonging to the 'non-boundary' class. We learn a *boosted decision stump* (BDS) classifier using training data and report performance on test data. Viola and Jones [15] pioneered this technique and Dollar et al. [3] applied this to image boundary detection.

Third, we propose two evaluation metrics, *Median Time Difference* and *Boundary Hit Rate*, in order to quantitatively evaluate musical boundary detection. We quantitatively evaluate both individual difference features and BDS classifiers using the RWC music database of 100 pop songs [6]. This data set now includes human annotated segmentations and can serve as a common test bed for future research [8].

Although we are unaware of previous work that explicitly addresses musical boundary detection, there has been a significant amount of work on music segmentation [9, 13, 10, 1, 11, 2, 4]. These two closely related problems are different in that musical boundary detection involves finding musical boundaries from temporally-local features whereas segmentation involves finding coherent segments within a song using notions of self-similarity (e.g. finding repetitive structures [9, 2, 4] or identifying similar spectral characteristics [13, 10, 1]). Our features are designed to indicate dissimilarity in the audio signal. In addition, our supervised approach has the benefit of allowing the user to explicitly specify their definition of a 'boundary' through the labels that they provide for the training data.

[1] Throughout this paper, we will consider a sample to be a 'feature sample' rather than a 'audio sample' that is calculated 10 per second.

2 MUSICAL BOUNDARY FEATURES

In this section, we describe various features that are useful for detecting musical boundaries. We loosely relate each to a high-level music concept for clarity, though these features are largely derived from a low-level spectral representation (e.g., short-time Fourier transforms) and often can be related to more than one high-level concept.

Each *feature* is time series that is sampled at a given feature sampling rate (e.g., 10 samples/sec). We create the time series by sliding a window (5-10 seconds in length) across the audio signal with a hopsize equal to the feature sampling rate. The audio signal in the window is summarized with a statistic. The statistic may be a scalar, a vector, a matrix, a set of scalars/vectors, or a time series of scalars/vectors. A *difference feature* is time series of scalar values that is derived by sliding a window over an audio signal. A difference feature is computed by comparing the statistic calculated in the first half of the window with the statistic calculated in the second half of the window. When our statistic is a scalar, vector, or matrix, we will compute the Euclidean norm of the difference. When our statistic is a set or times series, we first estimate one Gaussian distribution (with full covariance) from values in the first half of the window and a second Gaussian distribution from the values in the second half of the window. We then compute the symmetric Kullback-Leibler (sKL) divergence between the two Gaussian distributions (See Section 2.2.3.4 of [12] for details). The sKL is a non-negative scalar value that is large when the two estimated distributions greatly differ. One should note that by estimating Gaussians, we discards all temporal information when our statistic is a time series.

We normalize each feature so that, over an entire song, the mean of the samples is equal to 0 and variance of the samples equal to 1. Normalization is needed for generalization of features across the different songs in our corpus. In addition, we find empirically that 'smoothing' a feature by passing a Gaussian window over the time series improves performance.

2.1 Timbre-related features

Timbre describes sound qualities not related to melody or rhythm. For example, the instrumentation of a piece has a strong impact on the perception of timbre. We use a set of six scalar low-level audio statistics to describe the timbre of the audio contents within a window. (Section 2.2.5 of [12] for details.) These features are related to loudness, noisiness, brightness, and percussiveness of the signal. We compute six corresponding features, denoted *Spec*, by sliding a window over the musical signal. We also compute six spectral difference features, denoted Spec-Diff, by sliding a difference window over a musical signal and computing the difference between the values calculated in each window.

We also use Mel-frequency cepstral coefficients to describe timbre. An MFCC vector encodes the spectral shape of a short-time (e.g., 20 msec) audio frame. Given a difference window, we generate two time series of MFCC vectors (one for each half of the difference window) using the frames that are within the window. We estimate a Gaussian distribution using these time series of MFCC vectors and calculate one sKL value per window. A MFCC difference feature is then created by sliding the difference over the song. We generate a set of four MFCC difference features, denoted MFCC-diff, by using different subsets of the MFCCs: the first 5 MFCCs, the first 20 MFCCs, the 2nd to 5th MFCCs, and the 2th to 20th MFCCs. In addition, we will compute four MFCCDelta-Diff and four MFCCDelta2-Diff features using the instantaneous first and second derivatives from the series of MFCC vectors.

2.2 Harmony- and Melody-related features

Chromagrams are often used to encode information about key and major/minor tonality. For each short-time frame, We first computing a chroma vector [8] that measures the relative amount of energy in each pitch class (e.g. A,...,G#). Similar to the computation of the MFCC-diff features, we estimate Gaussian distributions from the chroma vectors that are extracted from each half of a difference windows and compute one sKL per window. We generate a Chromagram difference feature, denoted Chroma-diff, by sliding the difference window over the song. We use 12-, 24-, and 36-dimensional chroma vectors where each element represents 1, 1/2, and 1/3 of a pitch class, respectively. We also compute ChromaDelta-Diff and ChromaDelta2-Diff features using the first and second derivatives from the time series of chroma vectors.

In an attempt to model changes in the melody, we estimate the fundamental frequency (F0) and the cumulative power (F0Power) in the harmonics of the estimated F0 for each short-time frame[7]. For both the time series of FO values and F0Power values, we calculate F0-Diff and F0Power-Diff features using the same technique we used to create MFCC-Diff and Chroma-Diff features.

2.3 Rhythm-related features

To model rhythm, we use the fluctuation patterns (FPs) which describe modulations in the loudness for a set of frequency bands (see [12] for details). For each Mel-scale frequency band, an FFT is used to compute modulation frequencies of the loudness in dB within that band. Modulations around 4Hz are emphasized using a model of perceived fluctuation strength. The FP is a matrix where the rows correspond to the Mel-bands and the columns correspond to modulation frequencies. The values of a cell in the matrix describe strength of the specific modulation for a specific frequency band.

From an FP computed on each window, we calculate seven statistics denoted by *Flux* (see Section 2.2.5.4 of [12] for details). These statistics are related to the perceived tempo, strength of bass beats, overall strength of beats, etc. To calculate difference features, we compute one FP for the first half of the difference window and one FP for the second half of the difference window. Similar to the Spec-Diff features, we compute the seven *Flux-Diff* features by measuring the difference in values from the first half to the second half of the window. In addition, we compute an eighth Flux-diff feature by calculating the Frobenius norm between the two FPs.

	Set Feature	#	Comments
Timbre	Spec	6	Spectral features
	Spec-Diff	6	Differences between Spec Features
	MFCC-Diff	4	MFCCs 1-5,1-20,2-5,2-20
	MFCCDelta-Diff	4	1st Derivatives of MFCCs
	MFCCDelta2-Diff	4	2nd Derivatives of MFCCs
Harmony	Chroma-Diff	3	Chroma Vectors - 12, 24, 36 Pitch Classes
	ChromaDelta-Diff	3	1st Derivatives of chroma Vectors
	ChromaDelta2-Diff	3	2nd Derivatives of chroma Vectors
Melody	F0	2	F0 and F0Power
	F0-diff	2	Differences between F0 Features
Rhythm	Flux	7	Fluctuation Pattern (FP) features
	Flux-Diff	8	Difference of FP features, Frobenius norm of difference between FPs

Table 1. Summary of features: For each song, we extract 37 *difference* features and 15 *non-difference* features.

3 PEAK PICKING FOR BOUNDARY DETECTION

Our various difference features are designed to peak when one or more timbral, harmonic, melodic or rhythmic cues occurs in a song. Since these cues are often associated with musical boundaries, we estimate the location of musical boundaries by 'picking the local peaks' of a difference feature. The quality of the local peaks can be evaluated by comparing them with a human-generated 'ground-truth' segmentation of the music. Since difference features are often noisy and simple peak picking results in estimating too many boundaries, we propose a technique called *local-peak-local-neighborhood* (LPLN). First, we smooth the feature (see Section 2) and then pick local peaks. For each local peak, we examine at a local neighborhood (e.g., +/- 0.5 seconds) and find the sample with the largest unsmoothed feature value. For each song, we rank all of these LPLN samples according to the unsmoothed feature value. Finally, we output the times corresponding to the top N LPLN samples, where N is a user-specified number. Empirically, we find that this technique does (slightly) improve performance over simple peak picking.

Peak picking can be considered 'unsupervised' since we assume, rather than learn, a relationship between peaks and boundaries. The two main drawbacks of this approach are that it is not obvious how to combine multiple features and that we estimate boundaries only at the peaks of our difference features. In the next section, we will propose a 'supervised' approach that uses the 'ground truth' boundaries from human segmentations to learn models which incorporate a large number of difference and non-difference features. This approach is flexible in that it does not depend on the assumption that the boundaries occur at the local peaks of an individual feature.

4 SUPERVISED BOUNDARY DETECTION USING BOOSTING

We can frame our musical boundary detection problem as a supervised learning problem. The label for a sample is 1 if a boundary occurs at that sample, and 0 otherwise. (Given a feature sample rate of 10 samples/second, there will be between 7-15 'boundary' samples and about 2000 total samples for each song.) We create a 832-dimensional *vector of feature values* for each sample by calculating additional features from the 52 features that are described in Section 2. We smooth each of these features three times using different smoothing window widths (e.g., 1.6 seconds, 6.4 seconds and 25.6 seconds) to encode different time resolutions. For each of the resulting 156 smoothed features, we calculate the first and second instantaneous derivatives and the absolute values of the first and second instantaneous derivatives. The derivative features are intended to encode information about the local optima (e.g., peaks) of the features. We also include their absolute values since decision stumps are linear classifiers and detection of local optima depends on the magnitude of the derivatives. For example, a peak occurs when the magnitude of the first derivative is close to zero. We then simultaneously sample each of these $52 + 156 + 156 * 4 = 832$ features in order to generate a series of 832-dimensional vectors of feature values. Our training data is the set of labeled samples from a corpus of songs, however, in practice we will use only a small random subset of the non-boundary samples.

We learn a *boosted decision stumps* (BDS) classifier from the training data [3, 15]. A decision stump is a 'weak' classifier that operates on one feature by setting a threshold for that feature. For each sample, if the value of the feature is greater than the threshold, then it is classified as one class, otherwise it is classified as the other class. At each iteration of our *boosted* learning algorithm, we pick one decision stump with an associated weight and add it to our existing ensemble of decision stumps and weights. We use the AdaBoost algorithm to determine the weights [5] for each decision stump. When classifying a novel vector, we evaluate each of the stumps and combine their 'votes' using the learned weights.

Given a novel song, we extract features, generate the time series of vectors for feature values, and classify each vector using our BDS classifier. The classifier outputs both a binary decision and a confidence score of each sample belonging to the 'boundary' class. Often the classifier will predict many 'boundary' samples near one true boundary. We create a smoothed time series from the series of confidence scores and pick peaks using the LPLN technique.

5 EXPERIMENTAL SETUP

We evaluate both unsupervised (picking the peaks of individual difference features) and supervised (combining multiple features and learning a BDS classifier) approaches for detecting musical boundaries. Our data set is 100 pop songs from the RWC music database (RWC-MDB-P-2001) [6]. Each song has been manually segmented by a music graduate student [8]. We define a true musical boundary as a transition between 'intro', 'verse', 'chorus', 'bridge/solo', and 'outro' segments. There are between 7-15 boundaries per song and each song is between 2 and 6 minutes long.

We compare 260 unsupervised (52 features x 5 smoothing window widths) and 4 supervised BDS models. For each model, we estimate the location of the top 10 musical boundaries per song using the LPLN technique. We explicitly fix the number of boundaries since some (less smoothed) models tend to over segment songs. To evaluate a model, we use two sets of metrics: median times

Model	Median Time (seconds)		Boundary Hit Rates (ranges between 0.0 and 1.0)		
	guess-to-true	true-to-guess	precision	recall	Hit-F
Baselines					
Deterministic	8.85 (0.39)	6.40 (0.46)	0.039 (0.006)	0.052 (0.008)	0.043 (0.007)
Stochastic	9.67 (0.42)	8.80 (0.38)	0.043 (0.007)	0.056 (0.009)	0.048 (0.007)
Unsupervised Approaches: Picking the peaks of individual difference features					
MFCCDelta-Diff(1-20), 3.2 sec	**5.14** (0.49)	3.72 (0.56)	**0.258** (0.015)	**0.358** (0.022)	**0.295** (0.017)
f0-Diff(Power), 1.6 sec	8.07 (0.84)	6.36 (0.59)	0.140 (0.012)	0.195 (0.017)	0.160 (0.013)
Chroma-Diff(36), 1.6 sec	8.19 (0.71)	9.91 (0.92)	0.120 (0.010)	0.163 (0.014)	0.136 (0.011)
Spec-Diff(Harm), 0.8 sec	7.11 (0.73)	15.87(1.53)	0.109 (0.011)	0.143 (0.017)	0.121 (0.012)
Flux-Diff(Norm), 12.8 sec	6.87 (0.52)	**3.71** (0.29)	0.096 (0.011)	0.131 (0.015)	0.109 (0.012)
Supervised Approaches: training a boosted decision stump (BDS) classifier					
BDS 200	**4.29** (0.47)	1.82 (0.30)	**0.327** (0.016)	**0.462** (0.022)	**0.378** (0.017)
BDS 100	4.63 (0.46)	**1.58** (0.24)	0.320 (0.014)	0.452 (0.020)	0.370 (0.016)
BDS 50	5.06 (0.50)	2.19 (0.33)	0.300 (0.016)	0.424 (0.022)	0.346 (0.018)
BDS 20	5.35 (0.53)	2.49 (0.37)	0.295 (0.016)	0.422 (0.023)	0.342 (0.018)

Table 2. Results: The reported value is the average of the metric over the 100 songs. The standard error is reported in parenthesis. For unsupervised models, we only report the best individual model (feature and smoothing width) from each of the five sets of difference features.

between estimated and true boundaries, and boundary hit rates. Two median time metrics, true-to-guess and guess-to-true, respectively measure the median time between each true boundary to the closest estimate, and the median time between each estimate to the closed true boundary. For the boundary hit rate, we consider an estimate a 'hit' when it is within half a second of a true boundary. We calculate the precision (the percentage of estimates that hit a true estimate), the recall (the percentage of true boundaries that are hit), and Hit-F (the harmonic average of precision and recall). Note that each true boundary can be hit by at most one estimate since we explicitly prevent estimates from being close to one another using the LPLN technique.

For all metrics, performance is averaged over the 100 songs in the data set. For each supervised model, we use 10-fold cross validation. In addition to providing results for both our unsupervised and supervised approaches, we also report the performance of two baselines for comparison. The *deterministic* baseline uniformly spaces 10 estimates over the course of the song. The *stochastic* baseline involves picking 10 samples randomly from each song to use as our estimates for the boundaries. All results are given in Table 2.

6 RESULTS

For the unsupervised approach, we compare 520 individual models by smoothing each of the 52 features presented in Section 2 with one of ten smoothing window with widths ranging from 0.1 seconds (no smoothing) to 51.2 seconds. We rank (by Hit-F) the best performing individual feature (and associated smoothing width) in each set of difference features. If we consider Hit-F metric, the best model for each difference feature set performs significantly better than the random baselines. All 12 MFCC-based difference features outperform all other difference features with respect to the hit rate metrics.

For the supervised approach, we produce results for BDS classifiers with 20, 50, 100, or 200 decision stumps. All BDS classifiers significantly out perform all unsupervised model. By examining the first 20 features selected during boosting, we find that MFCC, F0, Spec, Chroma and Flux difference features are all represented. This suggests that these features are encoding different cues and that, when combined, can produce superior performance. Another interesting finding is that the three smoothing widths and four derivatives are all represented in the first 20 features that are selected. This suggests that it is important to consider multiple time resolutions and to encode 'peaks' when designing a supervised framework that uses difference features.

REFERENCES

[1] S. A. Abdallah, K. Noland, M. Sandler, M. Casey, and C. Rhodes. Theory and evaluation of a bayesian music structure extractor. In *ISMIR*, pages 420–425, 2005.

[2] R. Dannenberg and N. Hu. Discovering musical structure in audio recordings. In *ICMAI*, pages 43–57, London, UK, 2002.

[3] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, June 2006.

[4] J. Foote. Visualizing music and audio using self-similarity. In *ACM MULTIMEDIA*, pages 77–80, New York, NY, USA, 1999.

[5] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, pages 23–37, 1995.

[6] M. Goto. Development of the RWC music database. In *International Congress on Acoustics*, pages 553–556, 2004.

[7] M. Goto. A real-time music-scene-description system: Predominant-f0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, 43(4):311–29, 2004.

[8] M. Goto. AIST annotation for RWC music database. In *ISMIR*, October 2006.

[9] Masataka Goto. A chorus-section detection method for musical audio signals and its application to a music listening station. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5), September 2006.

[10] M. Levy, M. Sandler, and M. Casey. Extraction of high-level musical structure from audio data and its application to thumbnail generation. In *ICASSP*, volume 5, pages V13–V16, May 2006.

[11] N. C. Maddage, C. Xu, M. S. Kankanhalli, and Xi Shao. Content-based music structure analysis with applications to music semantics understanding. In *ACM MULTIMEDIA*, pages 112–119, New York, NY, USA, 2004. ACM Press.

[12] E. Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, Vienna, Austria, 2006.

[13] C. Rhodes, M. Casey, S. Abdallah, and M. Sandler. A markov-chain monte-carlo approach to musical audio segmentation. In *ICASSP*, volume 5, pages V797–V800, May 2006.

[14] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet. Towards musical query-by-semantic description using the CAL500 data set. In *SIGIR '07*, 2007.

[15] P. Viola and M. J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004.

[16] K. West and S. Cox. Finding an optimal segmentation for audio genre classification. In *ISMIR*, pages 680–685, 2005.