ANALYZING SONG STRUCTURE WITH SPECTRAL CLUSTERING

Brian McFee
Center for Jazz Studies
Columbia University
brm2132@columbia.edu

Daniel P.W. Ellis
LabROSA
Columbia University
dpwe@ee.columbia.edu

ABSTRACT

Many approaches to analyzing the structure of a musical recording involve detecting sequential patterns within a self-similarity matrix derived from time-series features. Such patterns ideally capture repeated sequences, which then form the building blocks of large-scale structure.

In this work, techniques from spectral graph theory are applied to analyze repeated patterns in musical recordings. The proposed method produces a low-dimensional encoding of repetition structure, and exposes the hierarchical relationships among structural components at differing levels of granularity. Finally, we demonstrate how to apply the proposed method to the task of music segmentation.

1. INTRODUCTION

Detecting repeated forms in audio is fundamental to the analysis of structure in many forms of music. While small-scale repetitions — such as instances of an individual chord — are simple to detect, accurately combining multiple small-scale repetitions into larger structures is a challenging algorithmic task. Much of the current research on this topic begins by calculating local, frame-wise similarities over acoustic features (usually harmonic), and then searching for patterns in the all-pairs self-similarity matrix [3].

In the majority of existing work on structural segmentation, the analysis is *flat*, in the sense that the representation does not explicitly encode nesting or hierarchical structure in the repeated forms. Instead, novelty curves are commonly used to detect transitions between sections.

1.1 Our contributions

In this paper, we formulate the structure analysis problem in the context of spectral graph theory. By combining local consistency cues with long-term repetition encodings and analyzing the eigenvectors of the resulting graph Laplacian, we produce a compact representation that effectively encodes repetition structure at multiple levels of granularity. To effectively link repeating sequences, we formulate an optimally weighted combination of local timbre consistency and long-term repetition descriptors.

To motivate the analysis technique, we demonstrate its use for the standard task of flat structural annotation. However, we emphasize that the approach itself can be applied more generally to analyze structure at multiple resolutions.

1.2 Related work

The structural repetition features used in this work are inspired by those of Serrà *et al.* [11], wherein structure is detected by applying filtering operators to a lag-skewed self-similarity matrix. The primary deviation in this work is the graphical interpretation and subsequent analysis of the filtered self-similarity matrix.

Recently, Kaiser *et al.* demonstrated a method to combine tonal and timbral features for structural boundary detection [6]. Whereas their method forms a novelty curve from the combination of multiple features, our feature combination differs by using local timbre consistency to build internal connections among sequences of long-range tonal repetitions.

Our general approach is similar in spirit to that of Grohganz *et al.* [4], in which diagonal bands of a self-similarity matrix are expanded into block structures by spectral analysis. Their method analyzed the spectral decomposition of the self-similarity matrix directly, whereas the method proposed here operates on the graph Laplacian. Similarly, Kaiser and Sikora applied non-negative matrix factorization directly to a self-similarity matrix in order to detect blocks of repeating elements [7]. As we will demonstrate, the Laplacian provides a more direct means to expose block structure at multiple levels of detail.

2. GRAPHICAL REPETITION ENCODING

Our general structural analysis strategy is to construct and partition a graph over time points (samples) in the song. Let $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{d \times n}$ denote a d-dimensional time series feature matrix, *e.g.*, a chromagram or sequence of Mel-frequency cepstral coefficients. As a first step toward detecting and representing repetition structure, we form a *binary recurrence matrix* $R \in \{0, 1\}^{n \times n}$, where

$$R_{ij}(X) := \begin{cases} 1 & x_i, x_j \text{ are mutual } k\text{-nearest neighbors} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

and $k > 0$ parameterizes the degree of connectivity.

Ideally, repeated structures should appear as diagonal stripes in R. In practice, it is beneficial to apply a smoothing filter to suppress erroneous links and fill in gaps. We apply a windowed majority vote to each diagonal of R, resulting in the filtered matrix R':

$$R'_{ij} := \operatorname{maj}\left\{R_{i+t,j+t} \mid t \in -w, -w+1, \ldots, w\right\}, \quad (2)$$

where w is a discrete parameter that defines the minimum length of a valid repetition sequence.

2.1 Internal connectivity

The filtered recurrence matrix R' can be interpreted as an unweighted, undirected graph, whose vertices correspond to samples (columns of X), and edges correspond to equivalent position within a repeated sequence. Note, however, that successive positions $(i, i+1)$ will not generally be connected in R', so the constituent samples of a particular sequence may not be connected.

To facilitate discovery of repeated sections, edges between adjacent samples $(i, i+1)$ and $(i, i-1)$ are introduced, resulting in the *sequence-augmented graph* R^+:

$$\Delta_{ij} := \begin{cases} 1 & |i-j| = 1 \\ 0 & \text{otherwise} \end{cases}, \quad (3)$$

$$R^+_{ij} := \max(\Delta_{ij}, R'_{ij}). \quad (4)$$

With appropriate normalization, R^+ characterizes a Markov process over samples, where at each step i, the process either moves to an adjacent sample $i \pm 1$, or a random repetition of i; a process exemplified by the Infinite Jukebox [8].

Equation (4) combines local temporal connectivity with long-term recurrence information. Ideally, edges would exist only between pairs $\{i, j\}$ belonging to the same structural component, but of course, this information is hidden. The added edges along the first diagonals create a fully connected graph, but due to recurrence links, repeated sections will exhibit additional internal connectivity. Let i and j denote two repetitions of the same sample at different times; then R^+ should contain sequential edges $\{i, i+1\}$, $\{j, j+1\}$ and repetition edges $\{i, j\}$, $\{i+1, j+1\}$. On the other hand, unrelated sections with no repetition edges can only connect via sequence edges.

2.2 Balancing local and global linkage

The construction of eq. (4) describes the intuition behind combining local sequential connections with global repetition structure, but it does not balance the two competing goals. Long tracks with many repetitions can produce recurrence links which vastly outnumber local connectivity connections. In this regime, partitioning into contiguous sections becomes difficult, and subsequent analysis of the graph may fail to detect sequential structure.

If we allow (non-negative) weights on the edges, then the combination can be parameterized by a weighting parameter $\mu \in [0, 1]$:

$$R^\mu_{ij} := \mu R'_{ij} + (1-\mu)\Delta_{ij}. \quad (5)$$

This raises the question: how should μ be set? Returning to the motivating example of the random walk, we opt for a process that on average, tends to move either in sequence or across (all) repetitions with equal probability. In terms of μ, this indicates that the combination should assign equal weight to the local and repetition edges. This suggests a balancing objective for all frames i:

$$\mu \sum_j R'_{ij} \approx (1-\mu) \sum_j \Delta_{ij}.$$

Minimizing the average squared error between the two terms above leads to the following quadratic optimization:

$$\min_{\mu \in [0,1]} \frac{1}{2} \sum_i \left(\mu d_i(R') - (1-\mu) d_i(\Delta)\right)^2, \quad (6)$$

where $d_i(G) := \sum_j G_{ij}$ denotes the degree (sum of incident edge-weights) of i in G. Treating $d(\cdot) := [d_i(\cdot)]_{i=1}^n$ as a vector in $\mathbb{R}^n_+$ yields the optimal solution to eq. (6):

$$\mu^* = \frac{\langle d(\Delta), d(R') + d(\Delta)\rangle}{\|d(R') + d(\Delta)\|^2}. \quad (7)$$

Note that because Δ is non-empty (contains at least one edge), it follows that $\|d(\Delta)\|^2 > 0$, which implies $\mu^* > 0$. Similarly, if R' is non-empty, then $\mu^* < 1$, and the resulting combination retains the full connectivity structure of the unweighted R^+ (eq. (4)).

2.3 Edge weighting and feature fusion

The construction above relies upon a single feature representation to determine the self-similarity structure, and uses constant edge weights for the repetition and local edges. This can be generalized to support feature-weighted edges by replacing R' with a masked similarity matrix:

$$R'_{ij} \mapsto R'_{ij} S_{ij}, \quad (8)$$

where S_{ij} denotes a non-negative affinity between frames i and j, *e.g.*, a Gaussian kernel over feature vectors x_i, x_j:

$$S^{\text{rep}}_{ij} := \exp\left(-\frac{1}{2\sigma^2}\|x_i - x_j\|^2\right)$$

Similarly, Δ can be replaced with a weighted sequence graph. However, in doing so, care must be taken when selecting the affinity function. The same features used to detect repetition (typically harmonic in nature) may not capture local consistency, since successive frames do not generally retain harmonic similarity.

Recent work has demonstrated that local timbre differences can provide an effective cue for structural boundary detection [6]. This motivates the use of two contrasting feature descriptors: harmonic features for detecting long-range repeating forms, and timbral features for detecting local consistency. We assume that these features are respectively supplied in the form of affinity matrices S^{rep} and S^{loc}. Combining these affinities with the detected repetition structure and optimal weighting yields the *sequence-augmented affinity matrix* A:

$$A_{ij} := \mu R'_{ij} S^{\text{rep}}_{ij} + (1-\mu)\Delta_{ij} S^{\text{loc}}_{ij}, \quad (9)$$

where R' is understood to be constructed solely from the repetition affinities S^{rep}, and μ is optimized by solving (7) with the weighted affinity matrices.

3. GRAPH PARTITIONING AND STRUCTURAL ANALYSIS

The Laplacian is a fundamental tool in the field of spectral graph theory, as it can be interpreted as a discrete analog of a diffusion operator over the vertices of the graph, and its spectrum can be used to characterize vertex connectivity [2]. This section describes in detail how spectral clustering can be used to analyze and partition the repetition graph constructed in the previous section, and reveal musical structure.

3.1 Background: spectral clustering

Let D denote the diagonal *degree matrix* of A:

$$D := \operatorname{diag}(d(A)).$$

The *symmetric normalized Laplacian* L is defined as:

$$L := I - D^{-1/2} A D^{-1/2}. \tag{10}$$

The Laplacian forms the basis of spectral clustering, in which vertices are represented in terms of the eigenvectors of L [15]. More specifically, to partition a graph into m components, each point i is represented as the vector of the ith coordinates of the first m eigenvectors of L, corresponding to the m smallest eigenvalues. [1] The motivation for this method stems from the observation that the multiplicity of the bottom eigenvalue $\lambda_0 = 0$ corresponds to the number of connected components in a graph, and the corresponding eigenvectors encode component memberships amongst vertices.

In the non-ideal case, the graph is fully connected, so λ_0 has multiplicity 1, and the bottom eigenvector trivially encodes membership in the graph. However, in the case of A, we expect there to be many components with high intra-connectivity and relatively small inter-connectivity at the transition points between sections. Spectral clustering can be viewed as an approximation method for finding normalized graph cuts [15], and it is well-suited to detecting and pruning these weak links.

Figure 1 illustrates an example of the encoding produced by spectral decomposition of L. Although the first eigenvector (column) is uninformative, the remaining bases clearly encode membership in the diagonal regions depicted in the affinity matrix. The resulting pair-wise frame similarities for this example are shown in Figure 2, which clearly demonstrates the ability of this representation to iteratively reveal nested repeating structure.

To apply spectral clustering, we will use k-means clustering with the (normalized) eigenvectors $Y \in \mathbb{R}^{n \times M}$ as features, where $M > 0$ is a specified maximum number of structural component types. Varying M — equivalently, the dimension of the representation — directly controls the granularity of the resulting segmentation.

Algorithm 1 Boundary detection

Input: Laplacian eigenvectors $Y \in \mathbb{R}^{n \times m}$,
Output: Boundaries b, segment labels $c \in [m]^n$

```
function BOUNDARY-DETECT($Y$)
    $\hat{y}_i \leftarrow Y_{i,\cdot} / \|Y_{i,\cdot}\|$        // Normalize each row $Y_{i,\cdot}$
    Run $k$-means on $\{\hat{y}_i\}_{i=1}^{n}$ with $k = m$
    Let $c_i$ denote the cluster containing $\hat{y}_i$
    $b \leftarrow \{i | \; c_i \neq c_{i+1}\}$
    return $(b, c)$
```

3.2 Boundary detection

For a fixed number of segment types m, segment boundaries can estimated by clustering the rows of Y after truncating to the first m dimensions. After clustering, segment boundaries are detected by searching for change-points in the cluster assignments. This method is formalized in Algorithm 1. Note that the number of segment *types* is distinct from the number of *segments* because a single type (*e.g.*, verse) may repeat multiple times throughout the track.

3.3 Laplacian structural decomposition

To decompose an input song into its structural components, we propose a method, listed as Algorithm 2, to find boundaries and structural annotations at multiple levels of structural complexity. Algorithm 2 first computes the Laplacian as described above, and then iteratively increases the set of eigenvectors for use in Algorithm 1. For $m = 2$, the first two eigenvectors — corresponding to the two smallest eigenvalues of L — are taken. In general, for m types of repeating component, the bottom m eigenvectors are used to label frames and detect boundaries. The result is a sequence of boundaries B^m and frame labels C^m, for values $m \in 2, 3, \ldots, M$.

Note that unlike most structural analysis algorithms, Algorithm 2 does not produce a single decomposition of the song, but rather a sequence of decompositions ordered by increasing complexity. This property can be beneficial in visualization applications, where a user may be interested in the relationship between structural components at multiple levels. Similarly, in interactive display applications, a user may request more or less detailed analyses for a track. Since complexity is controlled by a single, discrete parameter M, this application is readily supported with a minimal set of interface controls (*e.g.*, a slider).

However, for standardized evaluation, the method must produce a single, flat segmentation. Adaptively estimating the appropriate level of analysis in this context is somewhat ill-posed, as different use-cases require differing levels of detail. We apply a simple selection criterion based on the level of detail commonly observed in standard datasets [5, 12]. First, the set of candidates is reduced to those in which the mean segment duration is at least 10 seconds. Subject to this constraint, the segmentation level $\tilde{m}$ is selected to maximize frame-level annotation entropy. This strategy tends to produce solutions with approximately balanced distributions over the set of segment types.

[1] An additional length-normalization is applied to each vector, to correct for scaling introduced by the symmetric normalized Laplacian [15].

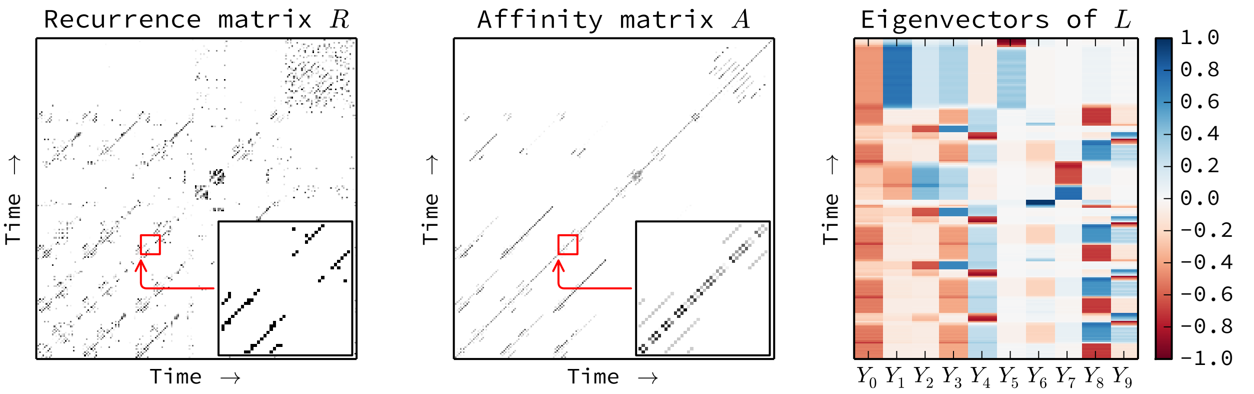

Figure 1. Left: the recurrence matrix R for *The Beatles — Come Together*. Center: the sequence-augmented affinity matrix A; the enlarged region demonstrates the cumulative effects of recurrence filtering, sequence-augmentation, and edge weighting. Right: the first 10 basis features (columns), ordered left-to-right. The leading columns encode the primary structural components, while subsequent components encode refinements.

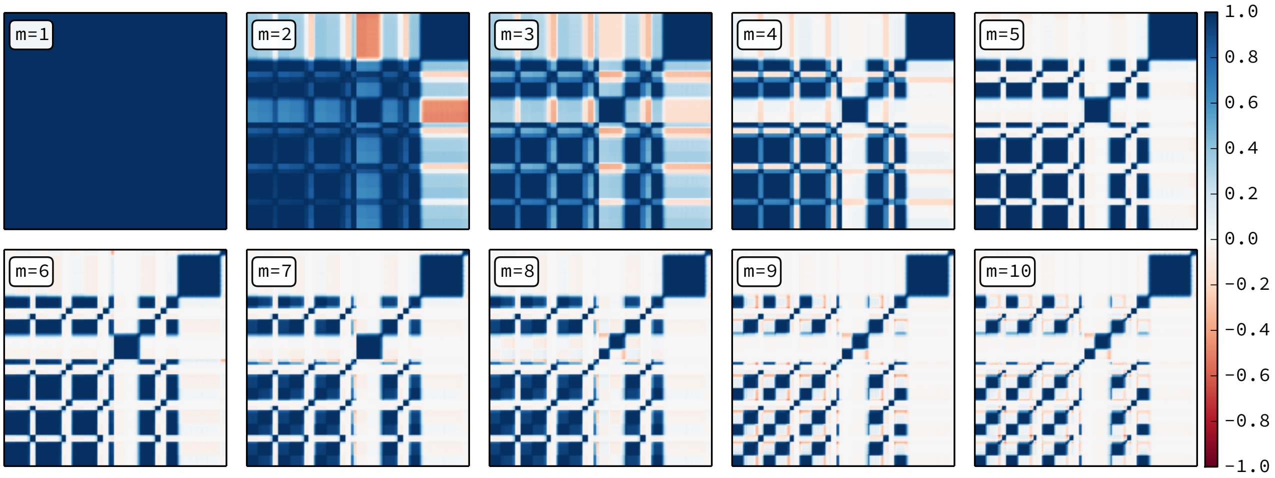

Figure 2. Pair-wise frame similarities (YY^T) using the first 10 components for *The Beatles — Come Together*. The first (trivial) component ($m = 1$) captures the entire song, and the second ($m = 2$) separates the outro (final vamp) from the rest of the song. Subsequent refinements separate the solo, refrain, verse, and outro, and then individual measures.

4. EXPERIMENTS

To evaluate the proposed method quantitatively, we compare boundary detection and structural annotation performance on two standard datasets. We evaluate the performance of the method using the automatic complexity estimation described above, as well as performance achieved for each fixed value of m across the dataset.

Finally, to evaluate the impact of the complexity estimation method, we compare to an oracle model. For each track, a different m^* is selected to maximize the evaluation metric of interest. This can be viewed as a simulation of interactive visualization, in which the user has the freedom to dynamically adapt the level of detail until she is satisfied. Results in this setting may be interpreted as measuring the best possible decomposition within the set produced by Algorithm 2.

4.1 Data and evaluation

Our evaluation data is comprised of two sources:

Beatles-TUT: 174 structurally annotated tracks from the Beatles corpus [10]. A single annotation is provided for each track, and annotations generally correspond to functional components (*e.g.*, *verse*, *refrain*, or *solo*).

SALAMI: 735 tracks from the SALAMI corpus [12]. This corpus spans a wide range of genres and instrumentation, and provides multiple annotation levels for each track. We report results on *functional* and *small-scale* annotations.

In each evaluation, we report the F-measure of boundary detection at 0.5-second and 3-second windows. To evaluate structural annotation accuracy, we report pairwise frame classification F-measure [9]. For comparison purposes, we report scores achieved by the method of Serrà *et*

Algorithm 2 Laplacian structural decomposition

```
Input: Affinities: $S^{\text{rep}}, S^{\text{loc}} \in \mathbb{R}_+^{n \times n}$, maximum number of segment types $0 < M \leq n$
Output: Boundaries $B^m$ and frame labels $C^m$ for $m \in 2 \ldots M$
function LSD($S^{\text{rep}}, S^{\text{loc}}, M$)
    $R \leftarrow$ eq. (1) on $S^{\text{rep}}$        // Recurrence detection
    $R' \leftarrow$ eq. (2) on $R$        // Recurrence filtering
    $A \leftarrow$ eq. (9)        // Sequence augmentation
    $L \leftarrow I - D^{-1/2} A D^{-1/2}$
    for $m \in 2, 3, \ldots, M$ do
        $Y \leftarrow$ bottom $m$ eigenvectors of $L$
        $(B^m, C^m) \leftarrow$ BOUNDARY-DETECT($Y$)
    return $\{(B^m, C^m)\}_{m=2}^{M}$
```

al., denoted here as SMGA [11].

4.2 Implementation details

All input signals are sampled at 22050Hz (mono), and analyzed with a 2048-sample FFT window and 512-sample hop. Repetition similarity matrices S^{rep} were computed by first extracting log-power constant-Q spectrograms over 72 bins, ranging from $C2$ (32.7 Hz) to $C8$ (2093.0 Hz).

Constant-Q frames were mean-aggregated between detected beat events, and stacked using time-delay embedding with one step of history as in [11]. Similarity matrices were then computed by applying a Gaussian kernel to each pair of beat-synchronous frames i and j. The bandwidth parameter σ^2 was estimated by computing the average squared distance between each x_i and its kth nearest neighbor, with k set to $1 + \lceil 2 \log_2 n \rceil$ (where n denotes the number of detected beats). The same k was used to connect nearest neighbors when building the recurrence matrix R, with the additional constraint that frames cannot link to neighbors within 3 beats of each-other, which prevents self-similar connections within the same measure. The majority vote window was set to $w = 17$.

Local timbre similarity S^{loc} was computed by extracting the first 13 Mel frequency cepstral coefficients (MFCC), mean-aggregating between detected beats, and then applying a Gaussian kernel as done for S^{rep}.

All methods were implemented in Python with NumPy and librosa [1, 14].

4.3 Results

The results of the evaluation are listed in Tables 1 to 3. For each fixed m, the scores are indicated as L_m. L indicates the automatic maximum-entropy selector, and L^* indicates the best possible m for each metric independently.

As a common trend across all data sets, the automatic m-selector often achieves results comparable to the best fixed m. However, it is consistently outperformed by the oracle model L^*, indicating that the output of Algorithm 2 often contains accurate solutions, the automatic selector does not always choose them.

Table 1. Beatles (TUT)

Method	$F_{0.5}$	F_3	F_{pair}
L_2	0.307 ± 0.14	0.429 ± 0.18	0.576 ± 0.14
L_3	0.303 ± 0.15	0.544 ± 0.17	0.611 ± 0.13
L_4	0.307 ± 0.15	0.568 ± 0.17	0.616 ± 0.13
L_5	0.276 ± 0.14	0.553 ± 0.15	0.587 ± 0.12
L_6	0.259 ± 0.14	0.530 ± 0.15	0.556 ± 0.12
L_7	0.246 ± 0.13	0.507 ± 0.14	0.523 ± 0.12
L_8	0.229 ± 0.13	0.477 ± 0.15	0.495 ± 0.12
L_9	0.222 ± 0.12	0.446 ± 0.14	0.468 ± 0.12
L_{10}	0.214 ± 0.11	0.425 ± 0.13	0.443 ± 0.12
L	0.312 ± 0.15	0.579 ± 0.16	0.628 ± 0.13
L^*	0.414 ± 0.14	0.684 ± 0.13	0.694 ± 0.12
SMGA	0.293 ± 0.13	0.699 ± 0.16	0.715 ± 0.15

Table 2. SALAMI (Functions)

Method	$F_{0.5}$	F_3	F_{pair}
L_2	0.324 ± 0.13	0.383 ± 0.15	0.539 ± 0.16
L_3	0.314 ± 0.13	0.417 ± 0.16	0.549 ± 0.13
L_4	0.303 ± 0.12	0.439 ± 0.16	0.547 ± 0.13
L_5	0.293 ± 0.12	0.444 ± 0.16	0.535 ± 0.12
L_6	0.286 ± 0.12	0.452 ± 0.16	0.521 ± 0.13
L_7	0.273 ± 0.11	0.442 ± 0.16	0.502 ± 0.13
L_8	0.267 ± 0.12	0.437 ± 0.16	0.483 ± 0.13
L_9	0.260 ± 0.11	0.443 ± 0.16	0.464 ± 0.14
L_{10}	0.250 ± 0.11	0.422 ± 0.16	0.445 ± 0.14
L	0.304 ± 0.13	0.455 ± 0.16	0.546 ± 0.14
L^*	0.406 ± 0.13	0.579 ± 0.15	0.652 ± 0.13
SMGA	0.224 ± 0.11	0.550 ± 0.18	0.553 ± 0.15

In the case of SALAMI (small), the automatic selector performs dramatically worse than many of the fixed-m methods, which may be explained by the relatively different statistics of segment durations and numbers of unique segment types in the small-scale annotations as compared to Beatles and SALAMI (functional).

To investigate whether a single m could simultaneously optimize multiple evaluation metrics for a given track, we plot the confusion matrices for the oracle selections on SALAMI (functional) in Figure 3. We observe that the m which optimizes F_3 is frequently larger than those for $F_{0.5}$ — as indicated by the mass in the lower triangle of the left plot — or F_{pair} — as indicated by the upper triangle of the right plot. Although this observation depends upon our particular boundary-detection strategy, it is corroborated by previous observations that the 0.5-second and 3.0-second metrics evaluate qualitatively different objectives [13]. Consequently, it may be beneficial in practice to provide segmentations at multiple resolutions when the specific choice of evaluation criterion is unknown.

5. CONCLUSIONS

The experimental results demonstrate that the proposed structural decomposition technique often generates solutions which achieve high scores on segmentation evaluation metrics. However, automatically selecting a single "best" segmentation without a priori knowledge of the evaluation criteria

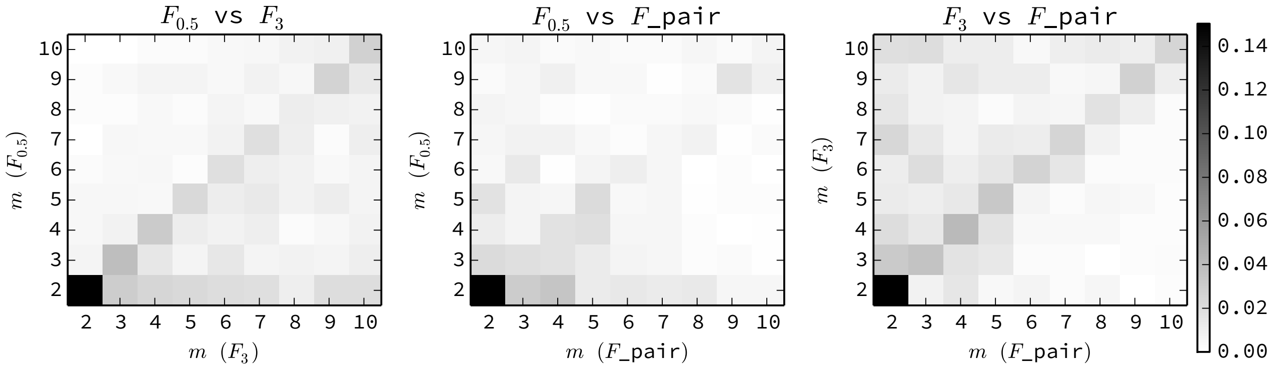

Figure 3. Confusion matrices illustrating the oracle selection of the number of segment types $m \in [2, 10]$ for different pairs of metrics on SALAMI (functional). While $m = 2$ is most frequently selected for all metrics, the large mass off-diagonal indicates that for a given track, a single fixed m does not generally optimize all evaluation metrics.

Table 3. SALAMI (Small)

Method	$F_{0.5}$	F_3	F_{pair}
L_2	0.151 ± 0.11	0.195 ± 0.13	0.451 ± 0.19
L_3	0.171 ± 0.12	0.259 ± 0.16	0.459 ± 0.17
L_4	0.186 ± 0.12	0.315 ± 0.17	0.461 ± 0.15
L_5	0.195 ± 0.12	0.354 ± 0.17	0.455 ± 0.14
L_6	0.207 ± 0.12	0.391 ± 0.18	0.452 ± 0.13
L_7	0.214 ± 0.12	0.420 ± 0.18	0.445 ± 0.13
L_8	0.224 ± 0.12	0.448 ± 0.18	0.435 ± 0.13
L_9	0.229 ± 0.12	0.467 ± 0.18	0.425 ± 0.13
L_{10}	0.234 ± 0.12	0.486 ± 0.18	0.414 ± 0.13
L	0.192 ± 0.11	0.344 ± 0.15	0.448 ± 0.16
L^*	0.292 ± 0.15	0.525 ± 0.19	0.561 ± 0.16
SMGA	0.173 ± 0.08	0.518 ± 0.12	0.493 ± 0.16

remains a challenging practical issue.

6. ACKNOWLEDGMENTS

The authors acknowledge support from The Andrew W. Mellon Foundation, and NSF grant IIS-1117015.

7. REFERENCES

[1] Librosa, 2014. https://github.com/bmcfee/librosa.

[2] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[3] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 1, pages 452–455. IEEE, 2000.

[4] Harald Grohganz, Michael Clausen, Nanzhu Jiang, and Meinard Müller. Converting path structures into block structures using eigenvalue decomposition of self-similarity matrices. In *ISMIR*, 2013.

[5] Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, University of London, 2010.

[6] Florian Kaiser and Geoffroy Peeters. A simple fusion method of state and sequence segmentation for music structure discovery. In *ISMIR*, 2013.

[7] Florian Kaiser and Thomas Sikora. Music structure discovery in popular music using non-negative matrix factorization. In *ISMIR*, pages 429–434, 2010.

[8] P. Lamere. The infinite jukebox, November 2012. http://infinitejuke.com/.

[9] Mark Levy and Mark Sandler. Structural segmentation of musical audio by constrained clustering. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(2):318–326, 2008.

[10] Jouni Paulus and Anssi Klapuri. Music structure analysis by finding repeated parts. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 59–68. ACM, 2006.

[11] J. Serrà, M. Müller, P. Grosche, and J. Arcos. Unsupervised music structure annotation by time series structure features and segment similarity. *Multimedia, IEEE Transactions on*, PP(99):1–1, 2014.

[12] Jordan BL Smith, John Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie. Design and creation of a large-scale database of structural annotations. In *ISMIR*, pages 555–560, 2011.

[13] Jordan BL Smith and Elaine Chew. A meta-analysis of the mirex structure segmentation task. In *ISMIR*, 2013.

[14] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[15] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.