Geometric Models for Musical Audio Data

Paul Bendich[1], Ellen Gasparovic[2], John Harer[3], and Christopher Tralie[4]

1 **Department of Mathematics, Duke University**
Geometric Data Analytics, Inc.
Durham, NC USA bendich@math.duke.edu
2 **Department of Mathematics, Union College**
Schenectady, NY USA gasparoe@union.edu
3 **Department of Mathematics and Electrical and Computer Engineering, Duke University**
Geometric Data Analytics, Inc.
Durham, NC USA harer@math.duke.edu
4 **Department of Electrical and Computer Engineering, Duke University**
Durham, NC USA chris.tralie@gmail.com

Abstract

We study the geometry of sliding window embeddings of audio features that summarize perceptual information about audio, including its pitch and timbre. These embeddings can be viewed as point clouds in high dimensions, and we add structure to the point clouds using a cover tree with adaptive thresholds based on multi-scale local principal component analysis to automatically assign points to clusters. We connect neighboring clusters in a *scaffolding* graph, and we use knowledge of stratified space structure to refine our estimates of dimension in each cluster, demonstrating in our music applications that choruses and verses have higher dimensional structure, while transitions between them are lower dimensional. We showcase our technique with an interactive web-based application powered by Javascript and WebGL which plays music synchronized with a principal component analysis embedding of the point cloud down to 3D. We also render the clusters and the scaffolding on top of this projection to visualize the transitions between different sections of the music.

1998 ACM Subject Classification H.5.5 Sound and Music Computing: Modeling, I.5.3 Clustering: Algorithms

Keywords and phrases Geometric Models, Audio Analysis, High Dimensional Data Analysis, Stratified Space Models

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.65

1 Music Features

We are interested in automatically finding structure in musical audio data by first converting it to a point cloud and then invoking geometric tools we have developed for building structure on top of point clouds sampled from stratified spaces. Musical audio data is typically sampled around 44100 hz, so its raw form is often unwieldy, though it is possible to summarize some of the most important perceptual information at a much lower data rate which is more amenable to structural analysis. A variety of lossy audio features have been hand-designed on top of the Short-Time Fourier Transform (SFTF) to pull out perceptual information about small chunks of audio in non-overlapping 30 millisecond "analysis windows," and a sequence of these features may then be used in place of the original audio. Two important

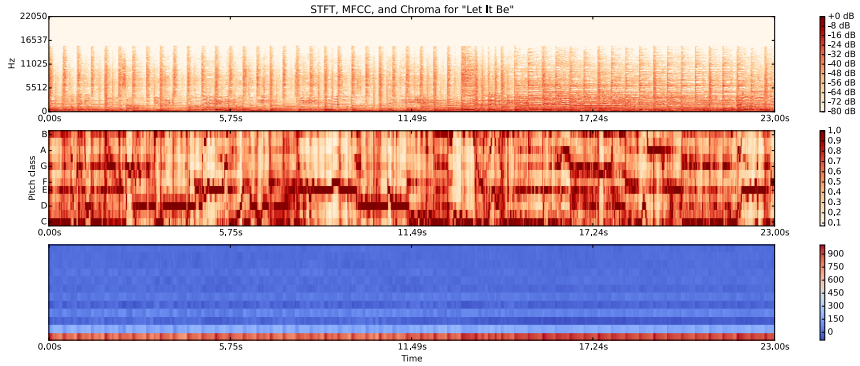

Figure 1 Audio features on top of a 23 second excerpt of "Let It Be" by The Beatles. The top figure shows the log Short-Time Fourier Transform, the middle figure shows chroma features, and the bottom figure shows MFCC features. These features were computed with the help of the "librosa" library in Python (`https://github.com/bmcfee/librosa`). [5]

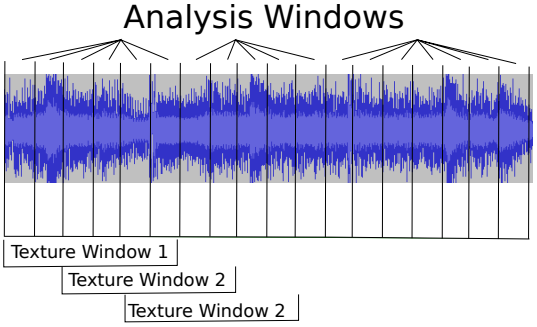

Figure 2 Features are extracted in short time 30ms windows of audio called analysis windows, and the features in blocks of these windows are summarized in texture windows.

complementary sets of such features are the Mel-Frequency Cepstral Coefficients (MFCC) [4] and chroma features [1]. MFCC retains coarse information about the entire smoothed spectrum of each time frequency window, while chroma summarizes pitch in 12 equivalence classes of frequencies across all octaves in a time frequency window, one feature for each halfstep in the Western scale. (Note that other spacings are also possible.) There are a number of other features in addition to chroma and MFCC that can be used to summarize information from the STFT. For an overview of these "timbral features," see [6].

If each feature is viewed as a dimension, then the features in each time frequency window can be thought of as a point in some Euclidean space. But since all of these features are computed over very short time windows, it is possible that many windows will be similar even in very different sections of the song. This problem is readily addressed by using much larger overlapping blocks of analysis windows called "texture windows" to capture more time evolution of sound in each audio chunk, as in [6]; see Figure 2. We simply take the mean and standard deviation of each feature over all of the analysis windows in a texture block. In particular, we take 5 timbral features, 12 MFCC coefficients, and 12 chroma features in each analysis window, for a total of 29 features, which we then summarize over 7 second blocks with their mean and standard deviation (233 30ms analysis windows for each 7 second texture window). We also add one "low energy" feature [6] for a total of 59 features so that the points live in $\mathbb{R}^{59}$. Finally, since we are fusing a collection of heterogeneous features, we normalize each dimension so that its standard deviation is 1 over the data cloud.

2 Geometric Models

Now that we have a scheme for turning audio into a point cloud, we turn to techniques we have developed for inferring stratified space structure from sampled points. More details can be found in [2], but we briefly summarize our techniques here. We first build a cover tree [3] on our point cloud X, which is a multi-scale way of organizing the point cloud. Each node in the tree is associated with a particular "center," which is a point in X. Each level l of the tree has an associated subset X_l of X and a radius $R_l = R_0/2^l$ so that the union of the balls of radius $2R_l$ centered at points in X_l cover all of the points in X. This condition makes it so that the subset of points at each level l goes from coarse to fine (and eventually includes all points in X). To ensure that the tree is balanced and that the collection of points of X at each level are evenly distributed, it is also the case that two centers at a particular level are each at least $2R_l$ apart from each other.

For a more parsimonious representation of our data, we choose a subtree of the cover tree so that the leaf nodes summarize geometrically homogeneous regions. In order to automatically choose which nodes to keep, we perform a version of principal component analysis (PCA) on the set of points within a certain radius of each center point at each level, and we do this at many radii. This process is known as *multi-scale local PCA* (MLPCA). Using a criterion based on the "eigenmetric" introduced in [2], for each node, we decide to either continue moving down the tree constructing subtrees, or to stop subdividing if the associated points have sufficiently similar eigenvalue profiles at multiple scales. The result of this adaptive process is a set of nodes at varying levels whose member points are geometrically similar to one another.

Our next step is to build a *scaffolding* graph on the resulting nodes, so that an edge is drawn between two nodes if the Euclidean distance between them is below some distance threshold. The distance threshold can be chosen automatically or manually in various ways; see [2]. Then, we use a local dimension estimation process based on MLPCA to estimate the intrinsic dimension locally near each one of the clusters. Namely, for each cluster, we perform PCA on the set of all points in that cluster together with all points in neighboring clusters connected to it in the scaffolding graph. Then, we compute the square roots of the eigenvalues we obtain from PCA, and use the largest gap between successive eigenvalues as an indication of what to choose as an initial dimension guess for that cluster. This initial guess may not be accurate, so we use knowledge from stratified space theory to inform and refine our local dimension estimate. A description of this refinement process may be found in [2]. Although this is the technique we chose to use in [2] and the one we used to create the examples in our Javascript demo, we point out that our framework for building the geometric models is flexible enough so that one can instead employ any local dimension estimation method of one's choice.

3 Javascript Demo

We created a GUI in Javascript/WebGL that enables interactive exploration of the geometric models we built, which can be run live at `http://www.ctralie.com/Research/GeometricModels`. We show a projection of the 59 dimensional point cloud onto the subspace determined by the first three principal components. Although information is lost in the projection (e.g., in the examples we made available, about 60% of the variance is explained by the first three principal components), we can still render the geometric model that we computed in high dimensions, as well as the dimension estimates of each cluster. The GUI plays music synchronized with the geometric model, highlighting the point that has its first

analysis window at the current time as it goes along, which causes it to trace out a trajectory through the geometric model over time.

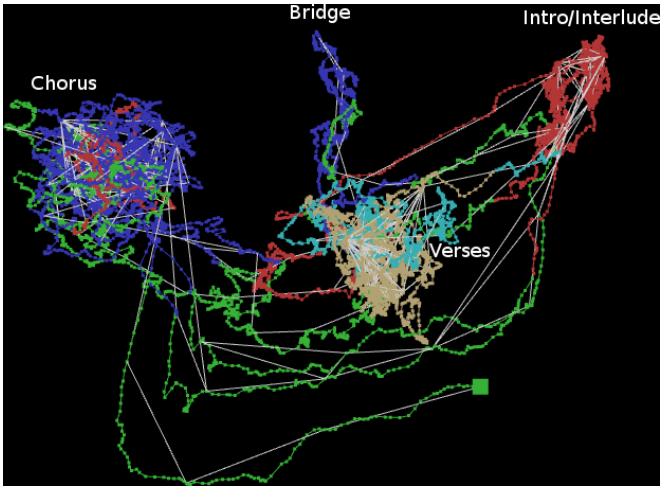

Figure 3 A screenshot from our Javascript GUI depicting our 3D visualization of the embedding of Michael Jackson's "Bad." The large green point indicates the visualization is at the beginning of the song, and this point moves through the model as the song plays.

Figure 3 shows a screenshot of our GUI on Michael Jackson's "Bad," with labels superimposed to mark different sections of the song. White line segments show edges in the scaffolding graph, i.e., connections between different clusters found in the cover tree. Each cluster is colored by its dimension estimate; specifically, green is dimension 1, blue is dimension 2, red is dimension 3, cyan is dimension 4, and tan is dimension 6. We notice in this example that the verses, interlude, and the chorus demonstrate areas of higher dimensionality, and transitions between them are clearly visible as 1 dimensional paths. Additionaly, a bridge, or a deviation from the main pattern of the song, constitutes its own cluster.

Acknowledgements: The first/third authors were partially supported by NSF award BIGDATA 1444791. The first was also partially supported by NSF award WBSE 3331753. The fourth author was supported by an NSF Graduate Fellowship NSF DGF 1106401. The second author thanks the Department of Mathematics at Duke University for hosting her during fall 2015.

References

1 Mark A. Bartsch and Gregory H. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, pages 15–18. IEEE, 2001.

2 Paul Bendich, Ellen Gasparovic, John Harer, and Christopher Tralie. Scaffoldings and spines: Organizing high-dimensional data using cover trees, local principal component analysis, and persistent homology, 2016. http://arxiv.org/abs/1602.06245.

3 Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104. ACM, 2006.

4 Bruce P. Bogert, Michael J.R. Healy, and John W. Tukey. The quefrency analysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. In *Proceedings of the symposium on time series analysis*, volume 15, pages 209–243. chapter, 1963.

5 Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in Python. In *Proceedings of the 14th Python in Science Conference*, 2015.

6 George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE transactions on Speech and Audio Processing*, 10(5):293–302, 2002.